SANJIV RANJAN DAS

DATA SCIENCE: THEORIES, MODELS, ALGORITHMS, AND ANALYTICS

S. R. DAS

10
Bidding it Up: Auctions

10.1 Theory

Auctions comprise one of the oldest market forms, and are still a popular mechanism for selling various assets and their related price discovery. In this chapter we will study different auction formats, bidding theory, and revenue maximization principles.

Hal Varian, Chief Economist at Google (NYT, Aug 1, 2002) writes:

"Auctions, one of the oldest ways to buy and sell, have been reborn and revitalized on the Internet.

When I say "old," I mean it. Herodotus described a Babylonian marriage market, circa 500 B.C., in which potential wives were auctioned off. Notably, some of the brides sold for a negative price.

The Romans used auctions for many purposes, including auctioning off the right to collect taxes. In A.D. 193, the Praetorian Guards even auctioned off the Roman empire itself!

We don't see auctions like this anymore (unless you count campaign finance practices), but auctions are used for just about everything else. Online, computer-managed auctions are cheap to run and have become increasingly popular. EBay is the most prominent example, but other, less well-known companies use similar technology."

10.1.1 Overview

Auctions have many features, but the key ingredient is *information asymmetry* between seller and buyers. The seller may know more about the product than the buyers, and the buyers themselves might have differential information about the item on sale. Moreover, buyers also take into

account imperfect information about the behavior of the other bidders. We will examine how this information asymmetry plays into bidding strategy in the mathematical analysis that follows.

Auction market mechanisms are *explicit*, with the prices and revenue a direct consequence of the auction design. In contrast, in other markets, the interaction of buyers and sellers might be more implicit, as in the case of commodities, where the market mechanism is based on demand and supply, resulting in the implicit, proverbial invisible hand setting prices.

There are many *examples* of active auction markets, such as auctions of art and valuables, eBay, Treasury securities, Google ad auctions, and even the New York Stock Exchange, which is an example of a continuous call auction market.

Auctions may be for a *single unit* (e.g., art) or *multiple units* (e.g., Treasury securities).

10.1.2 Auction types

The main types of auctions may be classified as follows:

1. English (E): highest bid wins. The auction is open, i.e., bids are revealed to all participants as they occur. This is an ascending price auction.
2. Dutch (D): auctioneer starts at a high price and calls out successively lower prices. First bidder accepts and wins the auction. Again, bids are open.
3. 1st price sealed bid (1P): Bids are sealed. Highest bidder wins and pays his price.
4. 2nd price sealed bid (2P): Same as 1P but the price paid by the winner is the second-highest price. Same as the auction analyzed by William Vickrey in his seminal paper in 1961 that led to a Nobel prize. See Vickrey (1961).
5. Anglo-Dutch (AD): Open, ascending-price auction till only two bidders remain, then it becomes sealed-bid.

10.1.3 Value Determination

The eventual outcome of an auction is price/value discovery of the item being sold. There are two characterizations of this value determination

process, depending on the nature of the item being sold.

1. Independent private values model: Each buyer bids his own independent valuation of the item at sale (as in regular art auctions).

2. Common-values model: Bidders aim to discover a common price, as in Treasury auctions. This is because there is usually an after market in which common value is traded.

10.1.4 Bidder Types

The assumptions made about the bidders impacts the revenue raised in the auction and the optimal auction design chosen by the seller. We consider two types of bidders.

1. Symmetric: all bidders observe the same probability distribution of bids and stop-out (SP) prices. The stop out price is the price of the lowest winning bid for the last unit sold. This is a robust assumption when markets are competitive.

2. Asymmetric or non-symmetric. Here the bidders may have different distributions of value. This is often the case when markets are segmented. Example: bidding for firms in M&A deals.

10.1.5 Benchmark Model (BM)

We begin by analyzing what is known as the benchmark model. It is the simplest framework in which we can analyze auctions. It is based on 4 main assumptions:

1. Risk-neutrality of bidders. We do not need utility functions in the analysis.

2. Private-values model. Every bidder has her own value for the item. There is a distribution of bidders' private values.

3. Symmetric bidders. Every bidder faces the same distribution of private values mentioned in the previous point.

4. Payment by winners is a function of bids alone. For a counterexample, think of payment via royalties for a book contract which depends on post auction outcomes. Or the bidding for movie rights, where the buyer takes a part share of the movie with the seller.

The following are the results and properties of the BM.

1. D = 1P. That is, the Dutch auction and first price auction are equivalent to bidders. These two mechanisms are identical because in each the bidder needs to choose how high to bid without knowledge of the other bids.

2. In the BM, the optimal strategy is to bid one's true valuation. This is easy to see for D and 1P. In both auctions, you do not see any other lower bids, so you bid up to your maximum value, i.e., one's true value, and see if the bid ends up winning. For 2P, if you bid too high you overpay, bid too low you lose, so best to bid one's valuation. For E, it's best to keep bidding till price crosses your valuation (reservation price).

3. Equilibria types:

   - Dominant: A situation where bidders bid their true valuation irrespective of other bidders bids. Satisfied by E and 2P.
   - Nash: Bids are chosen based on the best guess of other bidders' bids. Satisfied by D and 1P.

10.2 Auction Math

We now get away from the abstract definition of different types of auctions and work out an example of an auctions equilibrium.

Let F be the probability distribution of the bids. And define v_i as the true value of the i-th bidder, on a continuum between 0 and 1. Assume bidders are ranked in order of their true valuations v_i. How do we interpret $F(v)$? Think of the bids as being drawn from say, a beta distribution F on $v \in (0, 1)$, so that the probability of a very high or very low bid is lower than a bid around the mean of the distribution. The expected difference between the first and second highest bids is, given v_1 and v_2:

$$D = [1 - F(v_2)](v_1 - v_2)$$

That is, multiply the difference between the first and second bids by the probability that v_2 is the second-highest bid. Or think of the probability of there being a bid higher than v_2. Taking first-order conditions (from the seller's viewpoint):

$$\frac{\partial D}{\partial v_1} = [1 - F(v_2)] - (v_1 - v_2)F'(v_1) = 0$$

Note that $v_1 \equiv^d v_2$, given bidders are symmetric in BM. The symbol $\equiv^d$ means "equivalent in distribution". This implies that

$$v_1 - v_2 = \frac{1 - F(v_1)}{f(v_1)}$$

The expected revenue to the seller is the same as the expected 2nd price. The second price comes from the following re-arranged equation:

$$v_2 = v_1 - \frac{1 - F(v_1)}{f(v_1)}$$

10.2.1 Optimization by bidders

The goal of bidder i is to find a function/bidding rule B that is a function of the private value v_i such that

$$b_i = B(v_i)$$

where b_i is the actual bid. If there are n bidders, then

$$\begin{aligned} \Pr[\text{bidder } i \text{ wins}] &= \Pr[b_i > B(v_j)], \quad \forall j \neq i, \\ &= [F(B^{-1}(b_i))]^{n-1} \end{aligned}$$

Each bidder tries to maximize her expected profit relative to her true valuation, which is

$$\pi_i = (v_i - b_i)[F(B^{-1}(b_i))]^{n-1} = (v_i - b_i)[F(v_i)]^{n-1}, \tag{10.1}$$

again invoking the notion of bidder symmetry. Optimize by taking $\frac{\partial \pi_i}{\partial b_i} = 0$. We can get this by taking first the total derivative of profit relative to the bidder's value as follows:

$$\frac{d\pi_i}{dv_i} = \frac{\partial \pi_i}{\partial v_i} + \frac{\partial \pi_i}{\partial b_i}\frac{db_i}{dv_i} = \frac{\partial \pi_i}{\partial v_i}$$

which reduces to the partial derivative of profit with respect to personal valuation because $\frac{\partial \pi_i}{\partial b_i} = 0$. This useful first partial derivative is taken from equation (10.1):

$$\frac{\partial \pi_i}{\partial v_i} = [F(B^{-1}(b_i))]^{n-1}$$

Now, let v_l be the lowest bid. Integrate the previous equation to get

$$\pi_i = \int_{v_l}^{v_i} [F(x)]^{n-1}\, dx \tag{10.2}$$

Equating (10.1) and (10.2) gives

$$b_i = v_i - \frac{\int_{v_l}^{v_i} [F(x)]^{n-1}\, dx}{[F(v_i)]^{n-1}} = B(v_i)$$

which gives the bidding rule $B(v_i)$ entirely in terms of the personal valuation of the bidder. If, for example, F is uniform, then

$$B(v) = \frac{(n-1)v}{n}$$

Here we see that we "shade" our bid down slightly from our personal valuation. We bid less than true valuation to leave some room for profit. The amount of shading of our bid depends on how much competition there is, i.e., the number of bidders n. Note that

$$\frac{\partial B}{\partial v_i} > 0, \qquad \frac{\partial B}{\partial n} > 0$$

i.e., you increase your bid as your personal value rises, and as the number of bidders increases.

10.2.2 Example

We are bidding for a used laptop on eBay. Suppose we assume that the distribution of bids follows a beta distribution with minimum value $50 and a maximum value of $500. Our personal value for the machine is $300. Assume 10 other bidders. How much should we bid?

```
x = (1:1000)/1000
y = x*450+50
prob_y = dbeta(x,2,4)
print(c("check=",sum(prob_y)/1000))
prob_y = prob_y/sum(prob_y)
plot(y,prob_y,type="l")

> print(c("check=",sum(prob_y)/1000))
[1] "check="            "0.999998333334"
```

Note that we have used the non-central Beta distribution, with shape parameters $a = 2$ and $b = 4$. Note that the Beta density function is

$$\text{Beta}(x, a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\, x^{a-1}(1-x)^{b-1}$$

for x taking values between 0 and 1. The distribution of bids from 50 to 500 is shown in Figure 10.1. The mean and standard deviation are computed as follows.

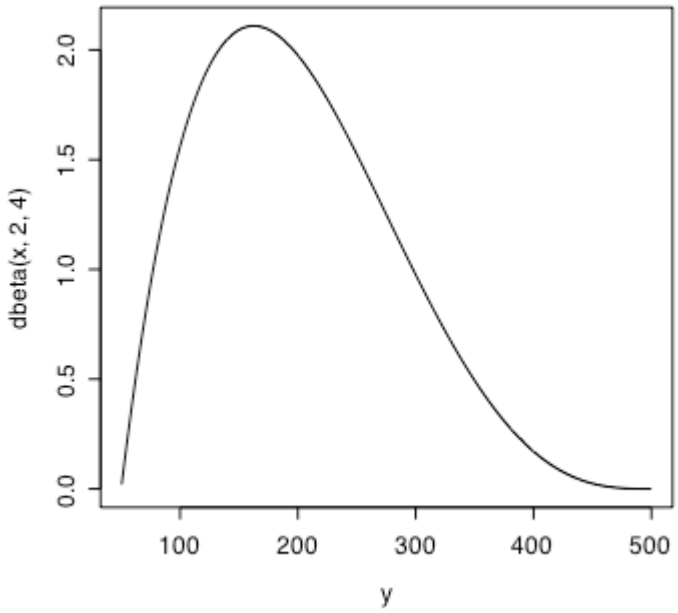


Figure 10.1: Probability density function for the Beta ($a = 2, b = 4$) distribution.

```
> print(c("mean=",sum(y*prob_y)))
[1] "mean="              "200.000250000167"
> print(c("stdev=",sqrt(sum(y^2*prob_y)-(sum(y*prob_y))^2)))
[1] "stdev="             "80.1782055353774"
```

We can take a computational approach to solving this problem. We program up equation 10.1 and then find the bid at which this is maximized.

```
> x = (1:1000)/1000
> y = 50 + 450*x
> cumprob_y = pbeta(x,2,4)
> exp_profit = (300-y)*cumprob_y^10
> idx = which(exp_profit==max(exp_profit))
> y[idx]
[1] 271.85
```

Hence, the bid of 271.85 is slightly lower than the reservation price. It is 10% lower. If there were only 5 other bidders, then the bid would be:

```
> exp_profit = (300-y)*cumprob_y^5
> idx = which(exp_profit==max(exp_profit))
> y[idx]
[1] 254.3
```

Now, we shade the bid down much more, because there are fewer competing bidders, and so the chance of winning with a lower bid increases.

10.3 Treasury Auctions

This section is based on the published paper by Das and Sundaram (1996). We move on from single-unit auctions to a very common multi-unit auction. Treasury auctions are the mechanism by which the Federal government issues its bills, notes, and bonds. Auctions are usually held on Wednesdays. Bids are received up to early afternoon after which the top bidders are given their quantities requested (up to prescribed ceilings for any one bidder), until there is no remaining supply of securities.

Even before the auction, Treasury securities trade in what is known as a "when-issued" or pre-market. This market gives early indications of price that may lead to tighter clustering of bids in the auction.

There are two types of dealers in a Treasury auction, primary dealers, i.e., the big banks and investment houses, and smaller independent bidders. The auction is really played out amongst the primary dealers. They place what are known as *competitive* bids versus the others, who place *non-competitive bids*.

Bidders also keep an eye on the secondary market that ensues right after the auction. In many ways, the bidders are also influenced by the possible prices they expect the paper to be trading at in the secondary market, and indicators of these prices come from the when-issued market.

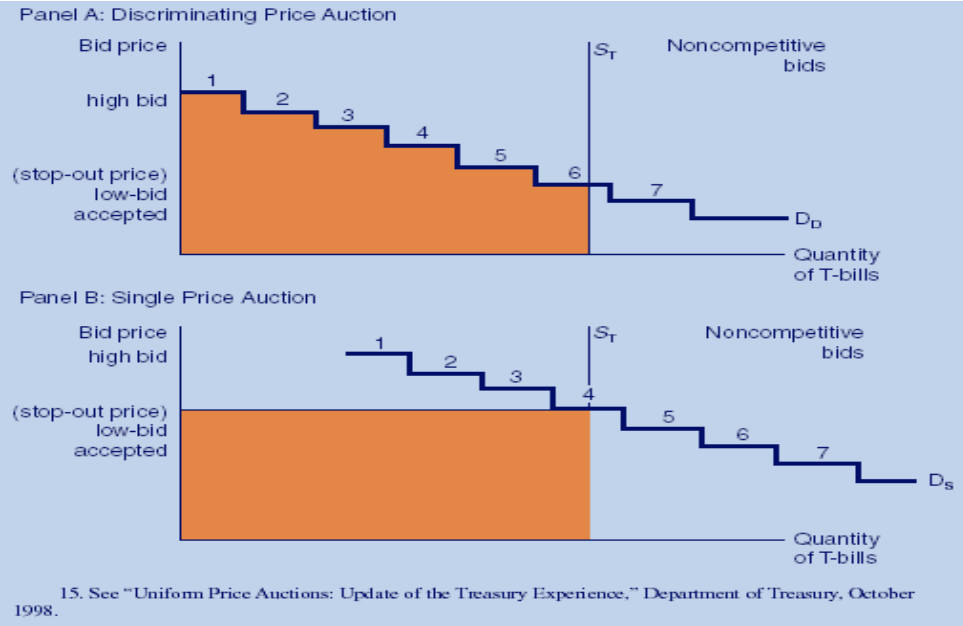
The winner in an auction experiences regret, because he knows he bid higher than everyone else, and senses that he overpaid. This phenomenon is known as the "winner's curse." Treasury auction participants talk amongst each other to mitigate winner's curse. The Fed also talks to primary dealers to mitigate their winner's curse and thereby induce them to bid higher, because someone with lower propensity for regret is likely to bid higher.

10.3.1 DPA or UPA?

DPA stands for "discriminating price auction" and UPA for "uniform price auction." The former was the preferred format for Treasury auctions and the latter was introduced only recently.

In a DPA, the highest bidder gets his bid quantity at the price he bid.

Then the next highest bidder wins his quantity at the price he bid. And so on, until the supply of Treasury securities is exhausted. In this manner the Treasury seeks to maximize revenue by filling each winning at the price. Since the prices paid by each winning bidder are different, the auction is called "discriminating" in price. Revenue maximization is attempted by walking down the demand curve, see Figure 10.2. The shaded area quantifies the revenue raised.



15. See "Uniform Price Auctions: Update of the Treasury Experience," Department of Treasury, October 1998.

Figure 10.2: Revenue in the DPA and UPA auctions.

In a UPA, the highest bidder gets his bid quantity at the price of the last winning bid (this price is also known as the stop-out price). Then the next highest bidder wins his quantity at the stop-out price. And so on, until the supply of Treasury securities is exhausted. Thus, the UPA is also known as a "single-price" auction. See Figure 10.2, lower panel, where the shaded area quantifies the revenue raised.

It may intuitively appear that the DPA will raise more revenue, but in fact, empirically, the UPA has been more successful. This is because the UPA incentivizes higher bids, as the winner's curse is mitigated. In a DPA, bids are shaded down on account of winner's curse – winning means you paid higher than what a large number of other bidders were willing to pay.

Some countries like Mexico have used the UPA format. The U.S., started with the DPA, and now runs both auction formats.

An interesting study examined markups achieved over yields in the when-issued market as an indicator of the success of the two auction formats. They examined the auctions of 2- and 5-year notes from June 1991 - 1994). [from Mulvey, Archibald and Flynn, US Office of the Treasury]. See Figure 10.3. The results of a regression of the markups on bid dispersion and duration of the auctioned securities shows that markups increase in the dispersion of bids. If we think of bid dispersion as a proxy for the extent of winner's curse, then we can see that the yields are pushed higher in the UPA than the DPA, therefore prices are lower in the UPA than the DPA. Markups are decreasing in the duration of the securities. Bid dispersion is shown in Figure 10.4.

Markup regressed on:	Multiple-price	Uniform-price
Intercept	-0.615**	-5.926**
	(0.266)	(1.663)
Dispersion of Bids	0.776**	1.540**
	(0.222)	(0.363)
Duration of auctioned securities	-0.030	-0.273
	(0.052)	(0.152)
R^2	0.30	0.33
D.W.	1.69	2.09

Standard errors are shown in parentheses. Two asterisks indicate statistical significance at the 99% level. The regressions are estimated with ordinary least squares and White's (1980) correction for heteroskedasticity.

Figure 10.3: Treasury auction markups.

10.4 Mechanism Design

What is a good auction mechanism? The following features might be considered.

- It allows easy entry to the game.
- It prevents collusion. For example, ascending bid auctions may be used to collude by signaling in the early rounds of bidding. Different auction formats may lead to various sorts of collusion.
- It induces truthful value revelation (also known as "truthful" bidding).
- Efficient - maximizes utility of auctioneer and bidders.
- Not costly to implement.
- Fair to all parties, big and small.

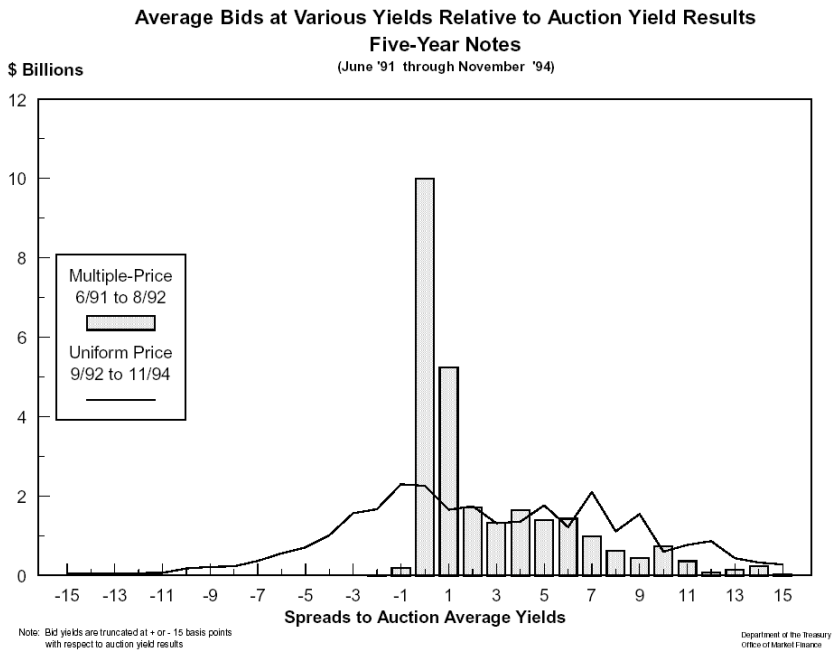


Figure 10.4: Bid-Ask Spread in the Auction.

10.4.1 Collusion

Here are some examples of collusion in auctions, which can be explicit or implicit. Collusion amongst buyers results in mitigating the winner's curse, and may work to either raise revenues or lower revenues for the seller.

- (Varian) 1999: German phone spectrum auction. Bids had to be in minimum 10% increments for multiple units. A firm bid 18.18 and 20 million for 2 lots. They signaled that everyone could center at 20 million, which they believed was the fair price. This sort of implicit collusion averts a bidding war.

- In Treasury auctions, firms can discuss bids, which is encouraged by the Treasury (why?). The restriction on cornering by placing a ceiling on how much of the supply one party can obtain in the auction aids collusion (why?). Repeated games in Treasury security auctions also aids collusion (why?).

- Multiple units also allows punitive behavior, by firms bidding to raise prices on lots they do not want to signal others should not bid on lots they do want.

10.4.2 Clicks (Advertising Auctions)

The Google AdWords program enables you to create advertisements which will appear on relevant Google search results pages and our network of partner sites. See `www.adwords.google.com`.

The Google AdSense program differs in that it delivers Google AdWords ads to individuals' websites. Google then pays web publishers for the ads displayed on their site based on user clicks on ads or on ad impressions, depending on the type of ad.

The material here refers to the elegant paper by Aggarwal, Goel, and Motwani (2006) on keyword auctions in AdWords. We first list some basic features of search engine advertising models. Aggarwal went on to work for Google as they adopted this algorithm from her thesis at Stanford.

1. Search engine advertising uses three models: (a) CPM, cost per thousand views, (b) CPC, cost per click, and (c) CPA, cost per acquisition. These are all at different stages of the search page experience.
2. CPC seems to be mostly used. There are 2 models here:
   (a) *Direct ranking*: the Overture model.
   (b) *Revenue ranking*: the Google model.
3. The merchant pays the price of the "next" click (different from "second" price auctions). This is non-truthful in both revenue ranking cases as we will see in a subsequent example. That is, bidders will not bid their true private valuations.
4. Asymmetric: there is an incentive to underbid, none to overbid.
5. Iterative: by placing many bids and watching responses, a bidder can figure out the bid ordering of other bidders for the same keywords, or basket of keywords. However, this is not obvious or simple. Google used to provide the GBS or Google Bid Simulator so that sellers using AdWords can figure out their optimal bids. See the following for more details on Adwords: `google.com/adwords/`.
6. If revenue ranking were truthful, it would maximize utility of auctioneer and merchant. (Known as auction "efficiency").
7. Innovation: the *laddered auction*. Randomized weights attached to bids. If weights are 1, then it's direct ranking. If weights are CTR (click-through rate), i.e. revenue-based, it's the revenue ranking.

To get some insights about the process of optimal bidding in AdWords auctions, see http://www.thesearchagents.com/2009/09/optimal-bidding-part-1-behind-the-scenes-of-google-adwords-bidding-tutorial/. See the Hal Varian video: http://www.youtube.com/watch?v=jRx7AMb6rZ0.

Here is a quick summary of Hal Varian's video. A merchant can figure out what the maximum bid per click should be in the following steps:

1. *Maximum profitable CPA*: This is the profit margin on the product. For example, if the selling price is $300 and cost is $200, then the profit margin is $100, which is also the maximum cost per acquisition (CPA) a seller would pay.

2. *Conversion Rate (CR)*: This is the number of times a click results in a sale. Hence, CR is equal to number of sales divided by clicks. So, if for every 100 clicks, we get a sale every 5 times, the CR is 5%.

3. *Value per Click (VPC)*: Equal to the CR times the CPA. In the example, we have $VPC = 0.05 \times 100 = \$5$.

4. *Determine the profit maximizing CPC bid*: As the bid is lowered, the number of clicks falls, but the CPC falls as well, revenue falls, but the profit after acquisition costs can rise until the sweet spot is determined. To find the number of clicks expected at each bid price, use the Google Bid Simulator. See the table below (from Google) for the economics at different bid prices. Note that the price you bid is not the price you pay for the click, because it is a "next-price" auction, based on a revenue ranking model, so the exact price you pay is based on Google's model, discussed in the next section. We see that the profit is maximized at a bid of $4.

Just as an example, note that the profit is equal to

$$(VPC - CPC) \times \#\text{Clicks} = (CPA \times CR - CPC) \times \#\text{Clicks}$$

Hence, for a bid of $4, we have

$$(5 - 407.02/154) \times 154 = \$362.98$$

Bid	Clicks	Cost	Revenue	Profit	ICC
$5.00	208	$697.42	$1040.00	$342.58	$5.73
$4.50	190	$594.27	$950.00	$355.73	$5.20
$4.00	154	$407.02	$770.00	$362.98	$4.63
$3.50	133	$309.73	$665.00	$355.27	$3.99
$3.00	113	$230.00	$565.00	$335.00	$3.32
$2.50	86	$140.37	$430.00	$289.63	

As pointed out by Varian, the rule is to compute ICC (Incremental cost per click), and make sure that it equals the VPC. The ICC at a bid of $5.00 is

$$ICC(5.00) = \frac{697.42 - 594.27}{208 - 190} = 5.73 > 5$$

Then

$$ICC(4.50) = \frac{594.27 - 407.02}{190 - 154} = 5.20 > 5$$

$$ICC(4.00) = \frac{407.02 - 309.73}{154 - 133} = 4.63 < 5$$

Hence, the optimal bid lies between $4.00 and $4.50.

10.4.3 Next Price Auctions

In a next-price auction, the CPC is based on the price of the click next after your own bid. Thus, you do not pay your bid price, but the one in the advertising slot just lower than yours. Hence, if your winning bid is for position j on the search screen, the price paid is that of the winning bid at position $j + 1$.

See the paper by Aggarwal, Goyal and Motwani (2006). Our discussion here is based on their paper. Let the true valuation (revenue) expected by bidder/seller i be equal to v_i. The CPC is denoted p_i. Let the click-through-rate (CTR) for seller/merchant i at a position j (where the ad shows up on the search screen) be denoted CTR_{ij}. CTR is the ratio of the number of clicks to the number of "impressions" i.e., the number of times the ad is shown.

- The "utility" to the seller is given by

$$\text{Utility} = CTR_{ij}(v_i - p_i)$$

- Example: 3 bidders A, B, C, with private values 200, 180, 100. There are two slots or ad positions with CTRs 0.5 and 0.4. If bidder A bids 200, pays 180, utility is $(200-180)\times 0.5 = 10$. But why not bid 110, for utility of $(200-100)\times 0.4 = 40$? This simple example shows that the next price auction is not truthful. Also note that your bid determines your ranking but not the price you pay (CPC).
- Ranking of bids is based on $w_i b_i$ in descending order of i. If $w_i = 1$, then we get the Overture direct ranking model. And if $w_i = CTR_{ij}$ then we have Google's revenue ranking model. In the example below, the weights range from 0 to 100, not 0 to 1, but this is without any loss of generality. The weights assigned to each merchant bidder may be based on some qualitative ranking such as the Quality Score (QS) of the ad.
- Price paid by bidder i is $\frac{w_{i+1}b_{i+1}}{w_i}$.
- Separable CTRs: CTRs of merchant $i = 1$ and $i = 2$ are the same for position j. No bidder position dependence.

10.4.4 Laddered Auction

AGM 2006 denoted the revised auction as "laddered". It gives a unique truthful auction. The main idea is to set the CPC to

$$p_i = \sum_{j=i}^{K} \left(\frac{CTR_{i,j} - CTR_{i,j+1}}{CTR_{i,i}} \right) \frac{w_{j+1}b_{j+1}}{w_i}, \quad 1 \le i \le K$$

so that

$$\frac{\#Clicks_i}{\#Impressions_i} \times p_i = CTR_{ii} \times p_i = \sum_{j=i}^{K} \left(CTR_{i,j} - CTR_{i,j+1} \right) \frac{w_{j+1}b_{j+1}}{w_i}$$

The lhs is the expected revenue to Google per ad impression. Make no mistake, the whole point of the model is to maximize Google's revenue, while making the auction system more effective for merchants. If this new model results in truthful equilibria, it is good for Google. The weights w_i are arbitrary and not known to the merchants.

Here is the table of CTRs for each slot by seller. These tables are the examples in the AGM 2006 paper.

	A	B	C	D
Slot 1	0.40	0.35	0.30	0.20
Slot 2	0.30	0.25	0.25	0.18
Slot 3	0.18	0.20	0.20	0.15

The assigned weights and the eventual allocations and prices are shown below.

	Weight	Bid	Score	Rank	Price
Merchant A	60	25	1500	1	13.5
Merchant B	40	30	1200	2	16
Merchant C	50	16	800	3	12
Merchant D	40	15	600	4	0

We can verify these calculations as follows.

```
> p3 = (0.20-0)/0.20 * 40/50 * 15; p3
[1] 12
> p2 = (0.25-0.20)/0.25 * 50/40 * 16 + (0.20-0)/0.25 * 40/40 * 15; p2
[1] 16
> p1 = (0.40-0.30)/0.40 * 40/60 * 30 + (0.30-0.18)/0.40 * 50/60 * 16
+ (0.18-0)/0.40 * 40/60 * 15; p1
[1] 13.5
```

See the paper for more details, but this equilibrium is unique and truthful.

Looking at this model, examine the following questions:

- What happens to the prices paid when the *CTR* drop rapidly as we go down the slots versus when they drop slowly?
- As a merchant, would you prefer that your weight be higher or lower?
- What is better for Google, a high dispersion in weights, or a low dispersion in weights?
- Can you see that by watching bidding behavior of the merchants, Google can adjust their weights to maximize revenue? By seeing a week's behavior Google can set weights for the next week. Is this legal?
- Is Google better off if the bids are more dispersed than when they are close together? How would you use the data in the table above to answer this question using R?

Exercise

Whereas Google clearly has modeled their AdWords auction to maximize revenue, less is known about how merchants maximize their net

revenue per ad, by designing ads, and choosing keywords in an appropriate manner. Google offers merchants a product called "Google Bid Simulator" so that the return from an adword (key word) may be determined.

In this exercise, you will first take the time to role play a merchant who is trying to explore and understand AdWords, and then come up with an approach to maximize the return from a portfolio of AdWords.

Here are some questions that will help in navigating the AdWords landscape.

1. What is the relation between keywords and cost-per-click (CPC)?
2. What is the Quality Score (QS) of your ad, and how does it related to keywords and CPC?
3. What defines success in an ad auction? What are its determinants?
4. What is AdRank. What does a higher AdRank buy for a merchant?
5. What are AdGroups and how do they relate to keywords?
6. What is automated CPC bidding?
7. What are the following tools? Keyword tool, Traffic estimator, Placement tool, Contextual targeting tool?
8. What is the incremental cost-per-click (ICC)?

Sketch a brief outline of how you might go about optimizing a portfolio of AdWords. Use the concepts we studied in Markowitz portfolio optimization for this.

11
Truncate and Estimate: Limited Dependent Variables

11.1 Introduction

Usually we run regressions using continuous variables for the dependent (y) variables, such as, for example, when we regress income on education. Sometimes however, the dependent variable may be discrete, and could be binomial or multinomial. That is, the dependent variable is "limited". In such cases, we need a different approach.

Discrete dependent variables are a special case of *limited dependent* variables. The Logit and Probit[1] models we look at here are examples of discrete dependent variable models. Such models are also often called *qualitative response* (QR) models.

[1] These are common usage and do not need to be capitalized, so we will use lower case subsequently.

In particular, when the variable is binary, i.e., takes values of $\{0, 1\}$, then we get a probability model. If we just regressed left hand side variables of ones and zeros on a suite of right hand side variables we could of course fit a linear regression. Then if we took another observation with values for the right hand side, i.e., $x = \{x_1, x_2, \ldots, x_k\}$, we could compute the value of the y variable using the fitted coefficients. But of course, this value will not be exactly 0 or 1, except by unlikely coincidence. Nor will this value lie in the range $(0, 1)$.

There is also a relationship to classifier models. In classifier models, we are interested in allocating observations to categories. In limited dependent models we also want to explain the reasons (i.e., find explanatory variables) for what results in the allocation across categories.

Some examples of such models are to explain whether a person is employed or not, whether a firm is syndicated or not, whether a firm is solvent or not, which field of work is chosen by graduates, where consumers shop, whether they choose Coke versus Pepsi, etc.

These fitted values might not even lie between 0 and 1 with a linear

regression. However, if we used a carefully chosen nonlinear regression function, then we could ensure that the fitted values of y are restricted to the range $(0,1)$, and then we would get a model where we fitted a probability. There are two such model forms that are widely used: (a) Logit, also known as a logistic regression, and (b) Probit models. We look at each one in turn.

11.2 *Logit*

A logit model takes the following form:

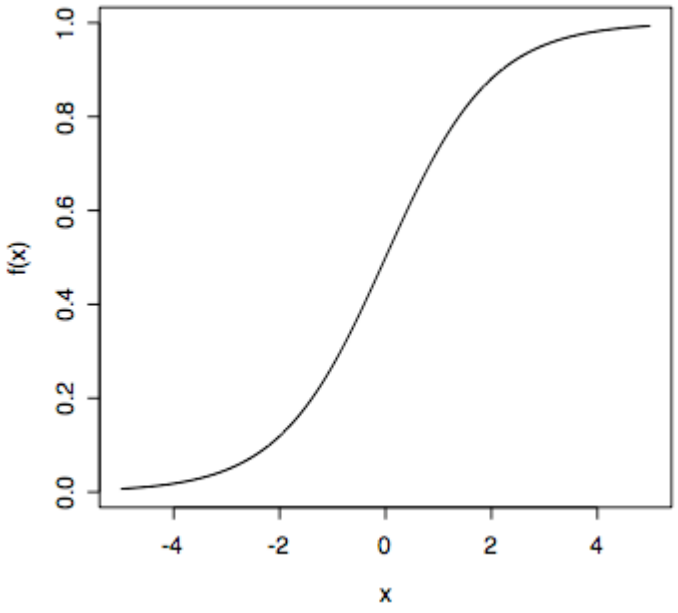
$$y = \frac{e^{f(x)}}{1+e^{f(x)}}, \quad f(x) = \beta_0 + \beta_1 x_1 + \ldots \beta_k x_k$$

We are interested in fitting the coefficients $\{\beta_0, \beta_1, \ldots, \beta_k\}$. Note that, irrespective of the coefficients, $f(x) \in (-\infty, +\infty)$, but $y \in (0,1)$. When $f(x) \to -\infty$, $y \to 0$, and when $f(x) \to +\infty$, $y \to 1$. We also write this model as

$$y = \frac{e^{\beta' x}}{1+e^{\beta' x}} \equiv \Lambda(\beta' x)$$

where Λ (lambda) is for logit.

The model generates a S-shaped curve for y, and we can plot it as follows:



The fitted value of y is nothing but the probability that $y = 1$.

For the NCAA data, take the top 32 teams and make their dependent variable 1, and that of the bottom 32 teams zero.

```
> 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 0 0 0 0 0
[39] 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
> x = as.matrix(ncaa[4:14])
```

Then 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 ‘ ’ 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
```

Suppose we ran this just with linear regression (this is also known as running a linear probability model):

```
> 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:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 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
```

11.3 *Probit*

Probit has essentially the same idea as the logit except that the probability function is replaced by the normal distribution. The nonlinear regression equation is as follows:

$$y = \Phi[f(x)], \quad f(x) = \beta_0 + \beta_1 x_1 + \ldots \beta_k x_k$$

where $\Phi(.)$ is the cumulative normal probability function. Again, irrespective of the coefficients, $f(x) \in (-\infty, +\infty)$, but $y \in (0, 1)$. When $f(x) \to -\infty$, $y \to 0$, and when $f(x) \to +\infty$, $y \to 1$.

We can redo the same previous logit model using a probit instead:

```
> 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 ‘ ’ 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
```

11.4 Analysis

Both these models are just settings in which we are computing binary probabilities, i.e.

$$\Pr[y = 1] = F(\beta' x)$$

where β is a vector of coefficients, and x is a vector of explanatory variables. F is the logit/probit function.

$$\hat{y} = F(\beta' x)$$

where $\hat{y}$ is the fitted value of y for a given x. In each case the function takes the logit or probit form that we provided earlier. Of course,

$$\Pr[y = 0] = 1 - F(\beta' x)$$

Note that the model may also be expressed in conditional expectation form, i.e.

$$E[y|x] = F(\beta' x)(y = 1) + [1 - F(\beta' x)](y = 0) = F(\beta' x)$$

11.4.1 Slopes

In a linear regression, it is easy to see how the dependent variable changes when any right hand side variable changes. Not so with nonlinear mod-

els. A little bit of pencil pushing is required (add some calculus too).

Remember that y lies in the range $(0,1)$. Hence, we may be interested in how $E(y|x)$ changes as any of the explanatory variables changes in value, so we can take the derivative:

$$\frac{\partial E(y|x)}{\partial x} = F'(\beta' x)\beta \equiv f(\beta' x)\beta$$

For each model we may compute this at the means of the regressors:

- In the logit model this is as follows:

```
(C1) F: exp(b*x)/(1+exp(b*x));
                                      b x
                                    %E
(D1)                          ----------
                                 b x
                              %E      + 1
(C2) diff(F,x);
                          b x             2 b x
                       b %E            b %E
(D2)                   ------- - ---------------
                         b x               b x       2
                      %E      + 1       (%E      + 1)
```

Therefore, we may write this as:

$$\frac{\partial E(y|x)}{\partial x} = \beta \left(\frac{e^{\beta' x}}{1 + e^{\beta' x}} \right) \left(1 - \frac{e^{\beta' x}}{1 + e^{\beta' x}} \right)$$

which may be re-written as

$$\frac{\partial E(y|x)}{\partial x} = \beta \cdot \Lambda(\beta' x) \cdot [1 - \Lambda(\beta' x)]$$

```
> h = glm(y~x, family=binomial(link="logit"))
> beta = h$coefficients
> beta
 (Intercept)         xPTS         xREB         xAST          xTO
-45.83315262 -0.06127422   0.49037435   0.16421685 -0.38404689
         xA.T         xSTL         xBLK          xPF          xFG
  1.56351478   0.78359670   0.07867125   0.02602243 46.21373793
          xFT          xX3P
 10.72992472   5.41984900
```

```
> dim(x)
[1] 64 11
> beta = as.matrix(beta)
> dim(beta)
[1] 12  1
> wuns = matrix(1,64,1)
> x = cbind(wuns,x)
> dim(x)
[1] 64 12
> xbar = as.matrix(colMeans(x))
> dim(xbar)
[1] 12  1
> xbar
           [,1]
     1.0000000
PTS 67.1015625
REB 34.4671875
AST 12.7484375
TO  13.9578125
A.T  0.9778125
STL  6.8234375
BLK  2.7500000
PF  18.6562500
FG   0.4232969
FT   0.6914687
X3P  0.3333750
> logitfunction = exp(t(beta) %*% xbar)/(1+exp(t(beta) %*% xbar))
> logitfunction
          [,1]
[1,] 0.5139925
> slopes = beta * logitfunction[1] * (1-logitfunction[1])
> slopes
                    [,1]
(Intercept) -11.449314459
xPTS         -0.015306558
xREB          0.122497576
xAST          0.041022062
xTO          -0.095936529
```

```
xA.T          0.390572574
xSTL          0.195745753
xBLK          0.019652410
xPF           0.006500512
xFG          11.544386272
xFT           2.680380362
xX3P          1.353901094
```

- In the probit model this is

$$\frac{\partial E(y|x)}{\partial x} = \phi(\beta' x)\beta$$

where $\phi(.)$ is the normal density function (not the cumulative probability).

```
> h = glm(y~x, family=binomial(link="probit"))
> beta = h$coefficients
> beta
 (Intercept)         xPTS         xREB         xAST          xTO
-26.28219202  -0.03462510   0.28493498   0.10893727  -0.23742076
        xA.T         xSTL         xBLK          xPF          xFG
  0.71484863   0.45963279   0.03029006   0.01040612  26.58460638
         xFT         xX3P
  6.28277680   3.15823537
> x = as.matrix(cbind(wuns,x))
> xbar = as.matrix(colMeans(x))
> dim(xbar)
[1] 12  1
> dim(beta)
NULL
> beta = as.matrix(beta)
> dim(beta)
[1] 12  1
> slopes = dnorm(t(beta) %*% xbar)[1]*beta
> slopes
                    [,1]
(Intercept) -10.470181164
xPTS         -0.013793791
xREB          0.113511111
xAST          0.043397939
```

xTO	−0.094582613
xA.T	0.284778174
xSTL	0.183106438
xBLK	0.012066819
xPF	0.004145544
xFG	10.590655632
xFT	2.502904294
xX3P	1.258163568

11.4.2 Maximum-Likelihood Estimation (MLE)

Estimation in the models above, using the `glm` function is done by R using MLE. Lets write this out a little formally. Since we have say n observations, and each LHS variable is $y = \{0, 1\}$, we have the likelihood function as follows:

$$L = \prod_{i=1}^{n} F(\beta' x)^{y_i} [1 - F(\beta' x)]^{1-y_i}$$

The log-likelihood will be

$$\ln L = \sum_{i=1}^{n} \left[y_i \ln F(\beta' x) + (1 - y_i) \ln[1 - F(\beta' x)] \right]$$

To maximize the log-likelihood we take the derivative:

$$\frac{\partial \ln L}{\partial \beta} = \sum_{i=1}^{n} \left[y_i \frac{f(\beta' x)}{F(\beta' x)} - (1 - y_i) \frac{f(\beta' x)}{1 - F(\beta' x)} \right] \beta = 0$$

which gives a system of equations to be solved for β. This is what the software is doing. The system of first-order conditions are collectively called the "likelihood equation".

You may well ask, how do we get the t-statistics of the parameter estimates β? The formal derivation is beyond the scope of this class, as it requires probability limit theorems, but let's just do this a little heuristically, so you have some idea of what lies behind it.

The t-stat for a coefficient is its value divided by its standard deviation. We get some idea of the standard deviation by asking the question: how does the coefficient set β change when the log-likelihood changes? That is, we are interested in $\partial\beta / \partial \ln L$. Above we have computed the reciprocal of this, as you can see. Lets define

$$g = \frac{\partial \ln L}{\partial \beta}$$

We also define the second derivative (also known as the Hessian matrix)

$$H = \frac{\partial^2 \ln L}{\partial \beta \partial \beta'}$$

Note that the following are valid:

$$\begin{aligned} E(g) &= 0 \quad \text{(this is a vector)} \\ Var(g) &= E(gg') - E(g)^2 = E(gg') \\ &= -E(H) \quad \text{(this is a non-trivial proof)} \end{aligned}$$

We call

$$I(\beta) = -E(H)$$

the information matrix. Since (heuristically) the variation in log-likelihood with changes in beta is given by $Var(g) = -E(H) = I(\beta)$, the inverse gives the variance of β. Therefore, we have

$$Var(\beta) \to I(\beta)^{-1}$$

We take the square root of the diagonal of this matrix and divide the values of β by that to get the t-statistics.

11.5 Multinomial Logit

You will need the `nnet` package for this. This model takes the following form:

$$\text{Prob}[y = j] = p_j = \frac{\exp(\beta_j' x)}{1 + \sum_{j=1}^{J} \exp(\beta_j' x)}$$

We usually set

$$\text{Prob}[y = 0] = p_0 = \frac{1}{1 + \sum_{j=1}^{J} \exp(\beta_j' x)}$$

To run this we set up as follows:

```
> ncaa = read.table("ncaa.txt",header=TRUE)
> x = as.matrix(ncaa[4:14])
> w1 = (1:16)*0 + 1
> w0 = (1:16)*0
> y1 = c(w1,w0,w0,w0)
> y2 = c(w0,w1,w0,w0)
> y3 = c(w0,w0,w1,w0)
```

```
> y4 = c(w0,w0,w0,w1)
> y = cbind(y1,y2,y3,y4)
> library(nnet)
> res  = multinom(y~x)
# weights:  52 (36 variable)
initial  value 88.722839
iter  10 value 71.177975
iter  20 value 60.076921
iter  30 value 51.167439
iter  40 value 47.005269
iter  50 value 45.196280
iter  60 value 44.305029
iter  70 value 43.341689
iter  80 value 43.260097
iter  90 value 43.247324
iter 100 value 43.141297
final  value 43.141297
stopped after 100 iterations
> res
Call:
multinom(formula = y ~ x)

Coefficients:
   (Intercept)       xPTS       xREB       xAST        xTO       xA.T
y2   -8.847514 -0.1595873  0.3134622  0.6198001 -0.2629260 -2.1647350
y3   65.688912  0.2983748 -0.7309783 -0.6059289  0.9284964 -0.5720152
y4   31.513342 -0.1382873 -0.2432960  0.2887910  0.2204605 -2.6409780
       xSTL        xBLK       xPF       xFG       xFT       xX3P
y2 -0.813519  0.01472506  0.6521056 -13.77579  10.374888 -3.436073
y3 -1.310701  0.63038878 -0.1788238 -86.37410 -24.769245 -4.897203
y4 -1.470406 -0.31863373  0.5392835 -45.18077   6.701026 -7.841990

Residual Deviance: 86.2826
AIC: 158.2826
> names(res)
 [1] "n"           "nunits"      "nconn"       "conn"
 [5] "nsunits"     "decay"       "entropy"     "softmax"
 [9] "censored"    "value"       "wts"         "convergence"
```

```
[13] "fitted.values" "residuals"     "call"          "terms"
[17] "weights"       "deviance"      "rank"          "lab"
[21] "coefnames"     "vcoefnames"    "xlevels"       "edf"
[25] "AIC"
> res$fitted.values
             y1           y2           y3           y4
1  6.785454e-01 3.214178e-01 7.032345e-06 2.972107e-05
2  6.168467e-01 3.817718e-01 2.797313e-06 1.378715e-03
3  7.784836e-01 1.990510e-01 1.688098e-02 5.584445e-03
4  5.962949e-01 3.988588e-01 5.018346e-04 4.344392e-03
5  9.815286e-01 1.694721e-02 1.442350e-03 8.179230e-05
6  9.271150e-01 6.330104e-02 4.916966e-03 4.666964e-03
7  4.515721e-01 9.303667e-02 3.488898e-02 4.205023e-01
8  8.210631e-01 1.530721e-01 7.631770e-03 1.823302e-02
9  1.567804e-01 9.375075e-02 6.413693e-01 1.080996e-01
10 8.403357e-01 9.793135e-03 1.396393e-01 1.023186e-02
11 9.163789e-01 6.747946e-02 7.847380e-05 1.606316e-02
12 2.448850e-01 4.256001e-01 2.880803e-01 4.143463e-02
13 1.040352e-01 1.534272e-01 1.369554e-01 6.055822e-01
14 8.468755e-01 1.506311e-01 5.083480e-04 1.985036e-03
15 7.136048e-01 1.294146e-01 7.385294e-02 8.312770e-02
16 9.885439e-01 1.114547e-02 2.187311e-05 2.887256e-04
17 6.478074e-02 3.547072e-01 1.988993e-01 3.816127e-01
18 4.414721e-01 4.497228e-01 4.716550e-02 6.163956e-02
19 6.024508e-03 3.608270e-01 7.837087e-02 5.547777e-01
20 4.553205e-01 4.270499e-01 3.614863e-04 1.172681e-01
21 1.342122e-01 8.627911e-01 1.759865e-03 1.236845e-03
22 1.877123e-02 6.423037e-01 5.456372e-05 3.388705e-01
23 5.620528e-01 4.359459e-01 5.606424e-04 1.440645e-03
24 2.837494e-01 7.154506e-01 2.190456e-04 5.809815e-04
25 1.787749e-01 8.037335e-01 3.361806e-04 1.715541e-02
26 3.274874e-02 3.484005e-02 1.307795e-01 8.016317e-01
27 1.635480e-01 3.471676e-01 1.131599e-01 3.761245e-01
28 2.360922e-01 7.235497e-01 3.375018e-02 6.607966e-03
29 1.618602e-02 7.233098e-01 5.762083e-06 2.604984e-01
30 3.037741e-02 8.550873e-01 7.487804e-02 3.965729e-02
31 1.122897e-01 8.648388e-01 3.935657e-03 1.893584e-02
32 2.312231e-01 6.607587e-01 4.770775e-02 6.031045e-02
```

```
33 6.743125e-01 2.028181e-02 2.612683e-01 4.413746e-02
34 1.407693e-01 4.089518e-02 7.007541e-01 1.175815e-01
35 6.919547e-04 4.194577e-05 9.950322e-01 4.233924e-03
36 8.051225e-02 4.213965e-03 9.151287e-01 1.450423e-04
37 5.691220e-05 7.480549e-02 5.171594e-01 4.079782e-01
38 2.709867e-02 3.808987e-02 6.193969e-01 3.154145e-01
39 4.531001e-05 2.248580e-08 9.999542e-01 4.626258e-07
40 1.021976e-01 4.597678e-03 5.133839e-01 3.798208e-01
41 2.005837e-02 2.063200e-01 5.925050e-01 1.811166e-01
42 1.829028e-04 1.378795e-03 6.182839e-01 3.801544e-01
43 1.734296e-01 9.025284e-04 7.758862e-01 4.978171e-02
44 4.314938e-05 3.131390e-06 9.997892e-01 1.645004e-04
45 1.516231e-02 2.060325e-03 9.792594e-01 3.517926e-03
46 2.917597e-01 6.351166e-02 4.943818e-01 1.503468e-01
47 1.278933e-04 1.773509e-03 1.209486e-01 8.771500e-01
48 1.320000e-01 2.064338e-01 6.324904e-01 2.907578e-02
49 1.683221e-02 4.007848e-01 1.628981e-03 5.807540e-01
50 9.670085e-02 4.314765e-01 7.669035e-03 4.641536e-01
51 4.953577e-02 1.370037e-01 9.882004e-02 7.146405e-01
52 1.787927e-02 9.825660e-02 2.203037e-01 6.635604e-01
53 1.174053e-02 4.723628e-01 2.430072e-03 5.134666e-01
54 2.053871e-01 6.721356e-01 4.169640e-02 8.078090e-02
55 3.060369e-06 1.418623e-03 1.072549e-02 9.878528e-01
56 1.122164e-02 6.566169e-02 3.080641e-01 6.150525e-01
57 8.873716e-03 4.996907e-01 8.222034e-03 4.832136e-01
58 2.164962e-02 2.874313e-01 1.136455e-03 6.897826e-01
59 5.230443e-03 6.430174e-04 9.816825e-01 1.244406e-02
60 8.743368e-02 6.710327e-02 4.260116e-01 4.194514e-01
61 1.913578e-01 6.458463e-04 3.307553e-01 4.772410e-01
62 6.450967e-07 5.035697e-05 7.448285e-01 2.551205e-01
63 2.400365e-04 4.651537e-03 8.183390e-06 9.951002e-01
64 1.515894e-04 2.631451e-01 1.002332e-05 7.366933e-01
```

You can see from the results that the probability for category 1 is the same as p_0. What this means is that we compute the other three probabilities, and the remaining is for the first category. We check that the probabilities across each row for all four categories add up to 1:

```
> rowSums(res$fitted.values)
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
 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
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

```
 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
51 52 53 54 55 56 57 58 59 60 61 62 63 64
 1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

11.6 Truncated Variables

Here we provide some basic results that we need later. And of course, we need to revisit our Bayesian ideas again!

- Given a probability density $f(x)$,

$$f(x|x > a) = \frac{f(x)}{Pr(x > a)}$$

  If we are using the normal distribution then this is:

$$f(x|x > a) = \frac{\phi(x)}{1 - \Phi(a)}$$

- If $x \sim N(\mu, \sigma^2)$, then

$$E(x|x > a) = \mu + \sigma \frac{\phi(c)}{1 - \Phi(c)}, \quad c = \frac{a - \mu}{\sigma}$$

  Note that this expectation is provided without proof, as are the next few ones. For example if we let x be standard normal and we want $E([x|x > -1]$, we have

```
> dnorm(-1)/(1-pnorm(-1))
[1] 0.2876000
```

- For the same distribution

$$E(x|x < a) = \mu + \sigma \frac{-\phi(c)}{\Phi(c)}, \quad c = \frac{a - \mu}{\sigma}$$

  For example, $E[x|x < 1]$ is

```
> -dnorm(1)/pnorm(1)
[1] -0.2876000
```

- Inverse Mills Ratio: The values $\frac{\phi(c)}{1-\Phi(c)}$ or $\frac{-\phi(c)}{\Phi(c)}$ as the case may be is often shortened to the variable $\lambda(c)$, which is also known as the Inverse Mills Ratio.

- If y and x are correlated (with correlation ρ), and $y \sim N(\mu_y, \sigma_y^2)$, then

$$\begin{aligned} Pr(y, x|x > a) &= \frac{f(y,x)}{Pr(x > a)} \\ E(y|x > a) &= \mu_y + \sigma_y \rho \lambda(c), \quad c = \frac{a - \mu}{\sigma} \end{aligned}$$

This leads naturally to the truncated regression model. Suppose we have the usual regression model where

$$y = \beta' x + e, \quad e \sim N(0, \sigma^2)$$

But suppose we restrict attention in our model to values of y that are greater than a cut off a. We can then write down by inspection the following correct model (no longer is the simple linear regression valid)

$$E(y|y > a) = \beta' x + \sigma \frac{\phi[(a - \beta' x)/\sigma]}{1 - \Phi[(a - \beta' x)/\sigma]}$$

Therefore, when the sample is truncated, then we need to run the regression above, i.e., the usual right-hand side $\beta' x$ with an additional variable, i.e., the Inverse Mill's ratio. We look at this in a real-world example.

An Example: Limited Dependent Variables in VC Syndications

Not all venture-backed firms end up making a successful exit, either via an IPO, through a buyout, or by means of another exit route. By examining a large sample of firms, we can measure the probability of a firm making a successful exit. By designating successful exits as $S = 1$, and setting $S = 0$ otherwise, we use matrix X of explanatory variables and fit a Probit model to the data. We define S to be based on a *latent* threshold variable S^* such that

$$S = \begin{cases} 1 & \text{if } S^* > 0 \\ 0 & \text{if } S^* \leq 0. \end{cases} \tag{11.1}$$

where the latent variable is modeled as

$$S^* = \gamma' X + u, \quad u \sim N(0, \sigma_u^2) \tag{11.2}$$

The fitted model provides us the probability of exit, i.e., $E(S)$, for all financing rounds.

$$E(S) = E(S^* > 0) = E(u > -\gamma' X) = 1 - \Phi(-\gamma' X) = \Phi(\gamma' X), \tag{11.3}$$

where γ is the vector of coefficients fitted in the Probit model, using standard likelihood methods. The last expression in the equation above follows from the use of normality in the Probit specification. $\Phi(.)$ denotes the cumulative normal distribution.

11.6.1 *Endogeneity*

Suppose we want to examine the role of syndication in venture success. Success in a syndicated venture comes from two broad sources of VC expertise. First, VCs are experienced in picking good projects to invest in, and syndicates are efficient vehicles for picking good firms; this is the selection hypothesis put forth by Lerner (1994). Amongst two projects that appear a-priori similar in prospects, the fact that one of them is selected by a syndicate is evidence that the project is of better quality (ex-post to being vetted by the syndicate, but ex-ante to effort added by the VCs), since the process of syndication effectively entails getting a second opinion by the lead VC. Second, syndicates may provide better monitoring as they bring a wide range of skills to the venture, and this is suggested in the value-added hypothesis of Brander, Amit and Antweiler (2002).

A regression of venture returns on various firm characteristics and a dummy variable for syndication allows a first pass estimate of whether syndication impacts performance. However, it may be that syndicated firms are simply of higher quality and deliver better performance, whether or not they chose to syndicate. Better firms are more likely to syndicate because VCs tend to prefer such firms and can identify them. In this case, the coefficient on the dummy variable might reveal a value-add from syndication, when indeed, there is none. Hence, we correct the specification for endogeneity, and then examine whether the dummy variable remains significant.

Greene (2011) provides the correction for endogeneity required here. We briefly summarize the model required. The performance regression is of the form:

$$Y = \beta' X + \delta S + \epsilon, \quad \epsilon \sim N(0, \sigma_\epsilon^2) \tag{11.4}$$

where Y is the performance variable; S is, as before, the dummy variable taking a value of 1 if the firm is syndicated, and zero otherwise, and δ is a coefficient that determines whether performance is different on account of syndication. If it is not, then it implies that the variables X are sufficient to explain the differential performance across firms, or that there is no differential performance across the two types of firms.

However, since these same variables determine also, whether the firm syndicates or not, we have an endogeneity issue which is resolved by adding a correction to the model above. The error term ϵ is affected by censoring bias in the subsamples of syndicated and non-syndicated

firms. When $S = 1$, i.e. when the firm's financing is syndicated, then the residual ϵ has the following expectation

$$E(\epsilon|S = 1) = E(\epsilon|S^* > 0) = E(\epsilon|u > -\gamma'X) = \rho\sigma_\epsilon \left[\frac{\phi(\gamma'X)}{\Phi(\gamma'X)}\right]. \quad (11.5)$$

where $\rho = Corr(\epsilon, u)$, and σ_ϵ is the standard deviation of ϵ. This implies that

$$E(Y|S = 1) = \beta'X + \delta + \rho\sigma_\epsilon \left[\frac{\phi(\gamma'X)}{\Phi(\gamma'X)}\right]. \quad (11.6)$$

Note that $\phi(-\gamma'X) = \phi(\gamma'X)$, and $1 - \Phi(-\gamma'X) = \Phi(\gamma'X)$. For estimation purposes, we write this as the following regression equation:

$$Y = \delta + \beta'X + \beta_m m(\gamma'X) \quad (11.7)$$

where $m(\gamma'X) = \frac{\phi(\gamma'X)}{\Phi(\gamma'X)}$ and $\beta_m = \rho\sigma_\epsilon$. Thus, $\{\delta, \beta, \beta_m\}$ are the coefficients estimated in the regression. (Note here that $m(\gamma'X)$ is also known as the inverse Mill's ratio.)

Likewise, for firms that are not syndicated, we have the following result

$$E(Y|S = 0) = \beta'X + \rho\sigma_\epsilon \left[\frac{-\phi(\gamma'X)}{1 - \Phi(\gamma'X)}\right]. \quad (11.8)$$

This may also be estimated by linear cross-sectional regression.

$$Y = \beta'X + \beta_m m'(\gamma'X) \quad (11.9)$$

where $m' = \frac{-\phi(\gamma'X)}{1-\Phi(\gamma'X)}$ and $\beta_m = \rho\sigma_\epsilon$.

The estimation model will take the form of a stacked linear regression comprising both equations (11.7) and (11.9). This forces β to be the same across all firms without necessitating additional constraints, and allows the specification to remain within the simple OLS form. If δ is significant after this endogeneity correction, then the empirical evidence supports the hypothesis that syndication is a driver of differential performance. If the coefficients $\{\delta, \beta_m\}$ are significant, then the expected difference in performance for each syndicated financing round (i, j) is

$$\delta + \beta_m \left[m(\gamma'_{ij}X_{ij}) - m'(\gamma'_{ij}X_{ij})\right], \quad \forall i, j. \quad (11.10)$$

The method above forms one possible approach to addressing treatment effects. Another approach is to estimate a Probit model first, and then to set $m(\gamma'X) = \Phi(\gamma'X)$. This is known as the instrumental variables approach.

The regression may be run using the `sampleSelection` package in R. Sample selection models correct for the fact that two subsamples may be different because of treatment effects.

11.6.2 Example: Women in the Labor Market

After loading in the package `sampleSelection` we can use the data set called `Mroz87`. This contains labour market participation data for women as well as wage levels for women. If we are explaining what drives women's wages we can simply run the following regression.

```
> library(sampleSelection)
> data(Mroz87)
> summary(Mroz87)
      lfp              hours            kids5           kids618
 Min.   :0.0000   Min.   :   0.0   Min.   :0.0000   Min.   :0.000
 1st Qu.:0.0000   1st Qu.:   0.0   1st Qu.:0.0000   1st Qu.:0.000
 Median :1.0000   Median : 288.0   Median :0.0000   Median :1.000
 Mean   :0.5684   Mean   : 740.6   Mean   :0.2377   Mean   :1.353
 3rd Qu.:1.0000   3rd Qu.:1516.0   3rd Qu.:0.0000   3rd Qu.:2.000
 Max.   :1.0000   Max.   :4950.0   Max.   :3.0000   Max.   :8.000
      age              educ            wage            repwage
 Min.   :30.00   Min.   : 5.00   Min.   : 0.000   Min.   :0.000
 1st Qu.:36.00   1st Qu.:12.00   1st Qu.: 0.000   1st Qu.:0.000
 Median :43.00   Median :12.00   Median : 1.625   Median :0.000
 Mean   :42.54   Mean   :12.29   Mean   : 2.375   Mean   :1.850
 3rd Qu.:49.00   3rd Qu.:13.00   3rd Qu.: 3.788   3rd Qu.:3.580
 Max.   :60.00   Max.   :17.00   Max.   :25.000   Max.   :9.980
     hushrs          husage          huseduc          huswage
 Min.   : 175   Min.   :30.00   Min.   : 3.00   Min.   : 0.4121
 1st Qu.:1928   1st Qu.:38.00   1st Qu.:11.00   1st Qu.: 4.7883
 Median :2164   Median :46.00   Median :12.00   Median : 6.9758
 Mean   :2267   Mean   :45.12   Mean   :12.49   Mean   : 7.4822
 3rd Qu.:2553   3rd Qu.:52.00   3rd Qu.:15.00   3rd Qu.: 9.1667
 Max.   :5010   Max.   :60.00   Max.   :17.00   Max.   :40.5090
     faminc           mtr            motheduc        fatheduc
 Min.   : 1500   Min.   :0.4415   Min.   : 0.000   Min.   : 0.000
 1st Qu.:15428   1st Qu.:0.6215   1st Qu.: 7.000   1st Qu.: 7.000
 Median :20880   Median :0.6915   Median :10.000   Median : 7.000
 Mean   :23081   Mean   :0.6789   Mean   : 9.251   Mean   : 8.809
 3rd Qu.:28200   3rd Qu.:0.7215   3rd Qu.:12.000   3rd Qu.:12.000
 Max.   :96000   Max.   :0.9415   Max.   :17.000   Max.   :17.000
      unem             city            exper          nwifeinc
 Min.   : 3.000   Min.   :0.0000   Min.   : 0.00   Min.   :-0.02906
 1st Qu.: 7.500   1st Qu.:0.0000   1st Qu.: 4.00   1st Qu.:13.02504
 Median : 7.500   Median :1.0000   Median : 9.00   Median :17.70000
 Mean   : 8.624   Mean   :0.6428   Mean   :10.63   Mean   :20.12896
 3rd Qu.:11.000   3rd Qu.:1.0000   3rd Qu.:15.00   3rd Qu.:24.46600
 Max.   :14.000   Max.   :1.0000   Max.   :45.00   Max.   :96.00000
  wifecoll     huscoll         kids
 TRUE:212     TRUE:295    Mode :logical
 FALSE:541    FALSE:458   FALSE:229
                          TRUE :524
> res = lm(wage ~ age + I(age^2) + educ + city, data=Mroz87)
> summary(res)

Call:
lm(formula = wage ~ age + I(age^2) + educ + city, data = Mroz87)

Residuals:
    Min      1Q  Median      3Q     Max
-4.6805 -2.1919 -0.4575  1.3588 22.6903

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -8.499373   3.296628  -2.578   0.0101 *
age          0.252758   0.152719   1.655   0.0983 .
I(age^2)    -0.002918   0.001761  -1.657   0.0980 .
educ         0.450873   0.050306   8.963   <2e-16 ***
city         0.080852   0.238852   0.339   0.7351
---
Signif. codes:  0 Ô***Õ 0.001 Ô**Õ 0.01 Ô*Õ 0.05 Ô.Õ 0.1 Ô Õ 1

Residual standard error: 3.075 on 748 degrees of freedom
Multiple R-squared: 0.1049,     Adjusted R-squared: 0.1001
F-statistic: 21.91 on 4 and 748 DF,  p-value: < 2.2e-16
```

So, education matters. But since education also determines labor force participation (variable `lfp`) it may just be that we can use `lfp` instead. Let's try that.

```
> res = lm(wage ~ age + I(age^2) + lfp + city, data=Mroz87)
> summary(res)

Call:
lm(formula = wage ~ age + I(age^2) + lfp + city, data = Mroz87)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1808 -0.9884 -0.1615  0.3090 20.6810

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.558e-01  2.606e+00  -0.175   0.8612
age          3.052e-03  1.240e-01   0.025   0.9804
I(age^2)     1.288e-05  1.431e-03   0.009   0.9928
lfp          4.186e+00  1.845e-01  22.690   <2e-16 ***
city         4.622e-01  1.905e-01   2.426   0.0155 *
---
Signif. codes:  0 Ô***Õ 0.001 Ô**Õ 0.01 Ô*Õ 0.05 Ô.Õ 0.1 Ô Õ 1

Residual standard error: 2.491 on 748 degrees of freedom
Multiple R-squared: 0.4129,     Adjusted R-squared: 0.4097
F-statistic: 131.5 on 4 and 748 DF,  p-value: < 2.2e-16

> res = lm(wage ~ age + I(age^2) + lfp + educ + city, data=Mroz87)
> summary(res)

Call:
lm(formula = wage ~ age + I(age^2) + lfp + educ + city, data = Mroz87)

Residuals:
    Min      1Q  Median      3Q     Max
-4.9895 -1.1034 -0.1820  0.4646 21.0160

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.7137850  2.5882435  -1.821    0.069 .
age          0.0395656  0.1200320   0.330    0.742
I(age^2)    -0.0002938  0.0013849  -0.212    0.832
lfp          3.9439552  0.1815350  21.726  < 2e-16 ***
educ         0.2906869  0.0400905   7.251 1.04e-12 ***
city         0.2219959  0.1872141   1.186    0.236
---
Signif. codes:  0 Ô***Õ 0.001 Ô**Õ 0.01 Ô*Õ 0.05 Ô.Õ 0.1 Ô Õ 1
```

```
Residual standard error: 2.409 on 747 degrees of freedom
Multiple R-squared: 0.4515,     Adjusted R-squared: 0.4478
F-statistic:   123 on 5 and 747 DF,  p-value: < 2.2e-16
```

In fact, it seems like both matter, but we should use the selection equation approach of Heckman, in two stages.

```
> res = selection(lfp ~ age + I(age^2) + faminc + kids + educ,
   wage ~ exper + I(exper^2) + educ + city, data=Mroz87, method = "2step" )
> summary(res)
--------------------------------------------
Tobit 2 model (sample selection model)
2-step Heckman / heckit estimation
753 observations (325 censored and 428 observed)
       and 14 free parameters (df = 740)
Probit selection equation:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.157e+00  1.402e+00  -2.965 0.003127 **
age          1.854e-01  6.597e-02   2.810 0.005078 **
I(age^2)    -2.426e-03  7.735e-04  -3.136 0.001780 **
faminc       4.580e-06  4.206e-06   1.089 0.276544
kidsTRUE    -4.490e-01  1.309e-01  -3.430 0.000638 ***
educ         9.818e-02  2.298e-02   4.272 2.19e-05 ***
Outcome equation:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.9712003  2.0593505  -0.472    0.637
exper        0.0210610  0.0624646   0.337    0.736
I(exper^2)   0.0001371  0.0018782   0.073    0.942
educ         0.4170174  0.1002497   4.160 3.56e-05 ***
city         0.4438379  0.3158984   1.405    0.160
Multiple R-Squared:0.1264,      Adjusted R-Squared:0.116
Error terms:
              Estimate Std. Error t value Pr(>|t|)
invMillsRatio   -1.098      1.266  -0.867    0.386
sigma            3.200         NA      NA       NA
rho             -0.343         NA      NA       NA
--------------------------------------------
```

11.6.3 Endogeity – Some Theory to Wrap Up

Endogeneity may be technically expressed as arising from a correlation of the independent variables and the error term in a regression. This can be stated as:

$$Y = \beta' X + u, \quad E(X \cdot u) \neq 0$$

This can happen in many ways:

1. *Measurement error*: If X is measured in error, we have $\tilde{X} = X + e$. The the regression becomes

   $$Y = \beta_0 + \beta_1(\tilde{X} - e) + u = \beta_0 + \beta_1\tilde{X} + (u - \beta_1 e) = \beta_0 + \beta_1\tilde{X} + v$$

   We see that

   $$E(\tilde{X} \cdot v) = E[(X + e)(u - \beta_1 e)] = -\beta_1 E(e^2) = -\beta_1 Var(e) \neq 0$$

2. *Omitted variables*: Suppose the true model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + u$$

but we do not have X_2, which happens to be correlated with X_1, then it will be subsumed in the error term and no longer will $E(X_i \cdot u) = 0, \forall i$.

3. *Simultaneity*: This occurs when Y and X are jointly determined. For example, high wages and high education go together. Or, advertising and sales coincide. Or that better start-up firms tend to receive syndication. The *structural form* of these settings may be written as:

$$Y = \beta_0 + \beta_1 X + u, \qquad X = \alpha_0 + \alpha_1 Y + v$$

The solution to these equations gives the *reduced-form* version of the model.

$$Y = \frac{\beta_0 + \beta_1 \alpha_0}{1 - \alpha_1 \beta_1} + \frac{\beta v + u}{1 - \alpha_1 \beta_1}, \qquad X = \frac{\alpha_0 + \alpha_1 \beta_0}{1 - \alpha_1 \beta_1} + \frac{v + \alpha_1 u}{1 - \alpha_1 \beta_1}$$

From which we can compute the endogeneity result.

$$Cov(X, u) = Cov\left(\frac{v + \alpha_1 u}{1 - \alpha_1 \beta_1}, u\right) = \frac{\alpha_1}{1 - \alpha_1 \beta_1} \cdot Var(u)$$

12
Riding the Wave: Fourier Analysis

12.1 *Introduction*

Fourier analysis comprises many different connnections between infinite series, complex numbers, vector theory, and geometry. We may think of different applications: (a) fitting economic time series, (b) pricing options, (c) wavelets, (d) obtaining risk-neutral pricing distributions via Fourier inversion.

12.2 *Fourier Series*

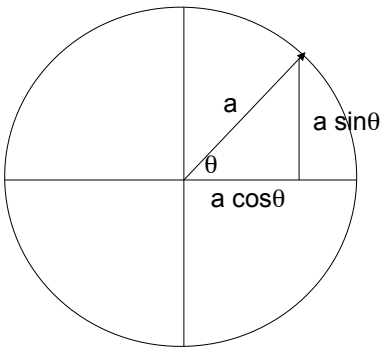
12.2.1 *Basic stuff*

Fourier series are used to represent *periodic* time series by combinations of sine and cosine waves. The time it takes for one cycle of the wave is called the "period" T of the wave. The "frequency" f of the wave is the number of cycles per second, hence,

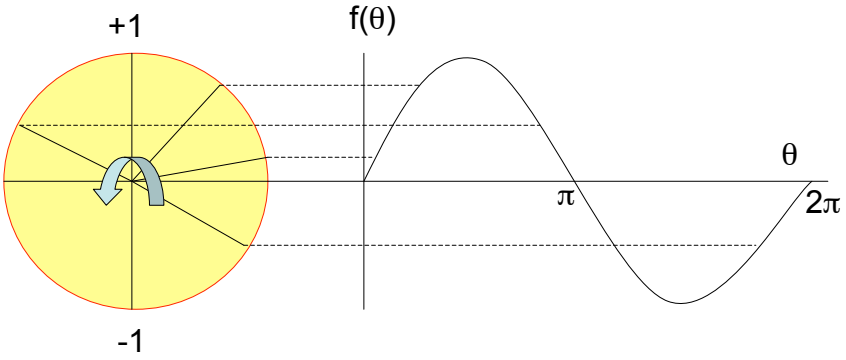
$$f = \frac{1}{T}$$

12.2.2 *The unit circle*

We need some basic geometry on the unit circle.



This circle is the unit circle if $a = 1$. There is a nice link between the unit circle and the sine wave. See the next figure for this relationship.



Hence, as we rotate through the angles, the height of the unit vector on the circle traces out the sine wave. In general for radius a, we get a sine wave with amplitude a, or we may write:

$$f(\theta) = a\sin(\theta) \tag{12.1}$$

12.2.3 *Angular velocity*

Velocity is distance per time (in a given direction). For angular velocity we measure distance in degrees, i.e. degrees per unit of time. The usual symbol for angular velocity is ω. We can thus write

$$\omega = \frac{\theta}{T}, \quad \theta = \omega T$$

Hence, we can state the function in equation (12.1) in terms of time as follows

$$f(t) = a\sin\omega t$$

12.2.4 *Fourier series*

A Fourier series is a collection of sine and cosine waves, which when summed up, closely approximate any given waveform. We can express the Fourier series in terms of sine and cosine waves

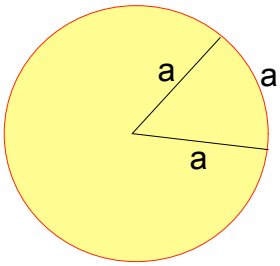
$$f(\theta) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\theta + b_n \sin n\theta)$$

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t)$$

The a_0 is needed since the waves may not be symmetric around the x-axis.

12.2.5 *Radians*

Degrees are expressed in units of radians. A radian is an angle defined in the following figure.



The angle here is a radian which is equal to 57.2958 degrees (approximately). This is slightly less than 60 degrees as you would expect to get with an equilateral triangle. Note that (since the circumference is $2\pi a$) $57.2958\pi = 57.2958 \times 3.142 = 180$ degrees.

So now for the unit circle

$$\begin{aligned} 2\pi &= 360 \text{ (degrees)} \\ \omega &= \frac{360}{T} \\ \omega &= \frac{2\pi}{T} \end{aligned}$$

Hence, we may rewrite the Fourier series equation as:

$$\begin{aligned} f(t) &= a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) \\ &= a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{2\pi n}{T} t + b_n \sin \frac{2\pi n}{T} t \right) \end{aligned}$$

So we now need to figure out how to get the coefficients $\{a_0, a_n, b_n\}$.

12.2.6 *Solving for the coefficients*

We start by noting the interesting phenomenon that sines and cosines are orthogonal, i.e. their inner product is zero. Hence,

$$\int_0^T \sin(nt).\cos(mt)\ dt = 0, \forall n, m \tag{12.2}$$

$$\int_0^T \sin(nt).\sin(mt)\ dt = 0, \forall n \neq m \tag{12.3}$$

$$\int_0^T \cos(nt).\cos(mt)\ dt = 0, \forall n \neq m \tag{12.4}$$

What this means is that when we multiply one wave by another, and then integrate the resultant wave from 0 to T (i.e. over any cycle, so we could go from say $-T/2$ to $+T/2$ also), then we get zero, unless the two waves have the *same* frequency. Hence, the way we get the coefficients of the Fourier series is as follows. Integrate both sides of the series in equation (12.2) from 0 to T, i.e.

$$\int_0^T f(t) = \int_0^T a_0\ dt + \int_0^T \left[\sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t)\ dt\right]$$

Except for the first term all the remaining terms are zero (integrating a sine or cosine wave over its cycle gives net zero). So we get

$$\int_0^T f(t)\ dt = a_0 T$$

or

$$a_0 = \frac{1}{T}\int_0^T f(t)\ dt$$

Now lets try another integral, i.e.

$$\begin{aligned}\int_0^T f(t)\cos(\omega t) &= \int_0^T a_0 \cos(\omega t)\ dt \\ &\quad + \int_0^T \left[\sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t)\cos(\omega t)\ dt\right]\end{aligned}$$

Here, all terms are zero except for the term in $a_1 \cos(\omega t)\cos(\omega t)$, because we are multiplying two waves (pointwise) that have the same frequency. So we get

$$\begin{aligned}\int_0^T f(t)\cos(\omega t) &= \int_0^T a_1 \cos(\omega t)\cos(\omega t)\ dt \\ &= a_1\ \frac{T}{2}\end{aligned}$$

How? Note here that for unit amplitude, integrating $\cos(\omega t)$ over one cycle will give zero. If we multiply $\cos(\omega t)$ by itself, we flip all the wave segments from below to above the zero line. The product wave now fills out half the area from 0 to T, so we get $T/2$. Thus

$$a_1 = \frac{2}{T}\int_0^T f(t)\cos(\omega t)$$

We can get all a_n this way - just multiply by $\cos(n\omega t)$ and integrate. We can also get all b_n this way - just multiply by $\sin(n\omega t)$ and integrate.

This forms the basis of the following summary results that give the coefficients of the Fourier series.

$$a_0 = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\,dt = \frac{1}{T}\int_0^T f(t)\,dt \tag{12.5}$$

$$a_n = \frac{1}{T/2}\int_{-T/2}^{T/2} f(t)\cos(n\omega t)\,dt = \frac{2}{T}\int_0^T f(t)\cos(n\omega t)\,dt \tag{12.6}$$

$$b_n = \frac{1}{T/2}\int_{-T/2}^{T/2} f(t)\sin(n\omega t)\,dt = \frac{2}{T}\int_0^T f(t)\sin(n\omega t)\,dt \tag{12.7}$$

12.3 Complex Algebra

Just for fun, recall that

$$e = \sum_{n=0}^{\infty}\frac{1}{n!}.$$

and

$$e^{i\theta} = \sum_{n=0}^{\infty}\frac{1}{n!}(i\theta)^n$$

$$\begin{aligned}\cos(\theta) &= 1 + 0.\theta - \frac{1}{2!}\theta^2 + 0.\theta^3 + \frac{1}{4!}\theta^2 + \ldots\\ i\sin(\theta) &= 0 + i\theta + 0.\theta^2 - \frac{1}{3!}i\theta^3 + 0.\theta^4 + \ldots\end{aligned}$$

Which leads into the famous Euler's formula:

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{12.8}$$

and the corresponding

$$e^{-i\theta} = \cos\theta - i\sin\theta \tag{12.9}$$

Recall also that $\cos(-\theta) = \cos(\theta)$. And $\sin(-\theta) = -\sin(\theta)$. Note also that if $\theta = \pi$, then

$$e^{-i\pi} = \cos(\pi) - i\sin(\pi) = -1 + 0$$

which can be written as

$$e^{-i\pi} + 1 = 0$$

an equation that contains five fundamental mathematical constants: $\{i, \pi, e, 0, 1\}$, and three operators $\{+, -, =\}$.

12.3.1 *From Trig to Complex*

Using equations (12.8) and (12.9) gives

$$\cos\theta = \frac{1}{2}(e^{i\theta} + e^{-i\theta}) \tag{12.10}$$

$$\sin\theta = \frac{1}{2}i(e^{i\theta} - e^{-i\theta}) \tag{12.11}$$

Now, return to the Fourier series,

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) \tag{12.12}$$

$$= a_0 + \sum_{n=1}^{\infty} \left(a_n \frac{1}{2}(e^{in\omega t} + e^{-in\omega t}) + b_n \frac{1}{2i}(e^{in\omega t} - e^{-in\omega t}) \right) \tag{12.13}$$

$$= a_0 + \sum_{n=1}^{\infty} \left(A_n e^{in\omega t} + B_n e^{-in\omega t} \right) \tag{12.14}$$

where

$$A_n = \frac{1}{T}\int_0^T f(t)e^{-in\omega t}\, dt$$

$$B_n = \frac{1}{T}\int_0^T f(t)e^{in\omega t}\, dt$$

How? Start with

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left(a_n \frac{1}{2}(e^{in\omega t} + e^{-in\omega t}) + b_n \frac{1}{2i}(e^{in\omega t} - e^{-in\omega t}) \right)$$

Then

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left(a_n \frac{1}{2}(e^{in\omega t} + e^{-in\omega t}) + b_n \frac{i}{2i^2}(e^{in\omega t} - e^{-in\omega t}) \right)$$

$$= a_0 + \sum_{n=1}^{\infty} \left(a_n \frac{1}{2}(e^{in\omega t} + e^{-in\omega t}) + b_n \frac{i}{-2}(e^{in\omega t} - e^{-in\omega t}) \right)$$

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left(\frac{1}{2}(a_n - ib_n)e^{in\omega t} + \frac{1}{2}(a_n + ib_n)e^{-in\omega t} \right) \tag{12.15}$$

Note that from equations (12.8) and (12.9),

$$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega t)\,dt \tag{12.16}$$
$$= \frac{2}{T}\int_0^T f(t)\frac{1}{2}[e^{in\omega t}+e^{-in\omega t}]\,dt \tag{12.17}$$
$$a_n = \frac{1}{T}\int_0^T f(t)[e^{in\omega t}+e^{-in\omega t}]\,dt \tag{12.18}$$

In the same way, we can handle b_n, to get

$$b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega t)\,dt \tag{12.19}$$
$$= \frac{2}{T}\int_0^T f(t)\frac{1}{2i}[e^{in\omega t}-e^{-in\omega t}]\,dt \tag{12.20}$$
$$= \frac{1}{i}\frac{1}{T}\int_0^T f(t)[e^{in\omega t}-e^{-in\omega t}]\,dt \tag{12.21}$$

So that

$$ib_n = \frac{1}{T}\int_0^T f(t)[e^{in\omega t}-e^{-in\omega t}]\,dt \tag{12.22}$$

So from equations (12.18) and (12.22), we get

$$\frac{1}{2}(a_n - ib_n) = \frac{1}{T}\int_0^T f(t)e^{-in\omega t}\,dt \equiv A_n \tag{12.23}$$
$$\frac{1}{2}(a_n + ib_n) = \frac{1}{T}\int_0^T f(t)e^{in\omega t}\,dt \equiv B_n \tag{12.24}$$

Put these back into equation (12.15) to get

$$f(t) = a_0 + \sum_{n=1}^{\infty}\left(\frac{1}{2}(a_n - ib_n)e^{in\omega t} + \frac{1}{2}(a_n + ib_n)e^{-in\omega t}\right) = a_0 + \sum_{n=1}^{\infty}\left(A_n e^{in\omega t} + B_n e^{-in\omega t}\right) \tag{12.25}$$

12.3.2 *Getting rid of a_0*

Note that if we expand the range of the first summation to start from $n = 0$, then we have a term $A_0 e^{i0\omega t} = A_0 \equiv a_0$. So we can then write our expression as

$$f(t) = \sum_{n=0}^{\infty} A_n e^{in\omega t} + \sum_{n=1}^{\infty} B_n e^{-in\omega t} \text{ (sum of A runs from zero)}$$

12.3.3 *Collapsing and Simplifying*

So now we want to collapse these two terms together. Lets note that

$$\sum_{n=1}^{2} x^n = x^1 + x^2 = \sum_{n=-2}^{-1} x^{-n} = x^2 + x^1$$

Applying this idea, we get

$$
\begin{aligned}
f(t) &= \sum_{n=0}^{\infty} A_n e^{in\omega t} + \sum_{n=1}^{\infty} B_n e^{-in\omega t} && (12.26)\\
&= \sum_{n=0}^{\infty} A_n e^{in\omega t} + \sum_{n=-\infty}^{-1} B_{(-n)} e^{in\omega t} && (12.27)\\
&\quad \text{where} \\
&\quad B_{(-n)} = \frac{1}{T}\int_0^T f(t) e^{-in\omega t}\, dt = A_n \\
&= \sum_{n=-\infty}^{\infty} C_n e^{in\omega t} && (12.28)\\
&\quad \text{where} \\
&\quad C_n = \frac{1}{T}\int_0^T f(t) e^{-in\omega t}\, dt
\end{aligned}
$$

where we just renamed A_n to C_n for clarity. The big win here is that we have been able to subsume $\{a_0, a_n, b_n\}$ all into one coefficient set C_n. For completeness we write

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) = \sum_{n=-\infty}^{\infty} C_n e^{in\omega t}$$

This is the complex number representation of the Fourier series.

12.4 Fourier Transform

The FT is a cool technique that allows us to go from the Fourier series, which needs a period T to waves that are aperiodic. The idea is to simply let the period go to infinity. Which means the frequency gets very small. We can then sample a slice of the wave to do analysis.

We will replace $f(t)$ with $g(t)$ because we now need to use f or Δf to denote frequency. Recall that

$$\omega = \frac{2\pi}{T} = 2\pi f, \quad n\omega = 2\pi f_n$$

To recap

$$
\begin{aligned}
g(t) &= \sum_{n=-\infty}^{\infty} C_n e^{in\omega t} = \sum_{n=-\infty}^{\infty} C_n e^{i2\pi f t} && (12.29)\\
C_n &= \frac{1}{T}\int_0^T g(t) e^{-in\omega t}\, dt && (12.30)
\end{aligned}
$$

This may be written alternatively in frequency terms as follows

$$C_n = \Delta f \int_{-T/2}^{T/2} g(t) e^{-i2\pi f_n t}\, dt$$

which we substitute into the formula for $g(t)$ and get

$$g(t) = \sum_{n=-\infty}^{\infty} \left[\Delta f \int_{-T/2}^{T/2} g(t) e^{-i2\pi f_n t}\, dt \right] e^{in\omega t}$$

Taking limits

$$g(t) = \lim_{T\to\infty} \sum_{n=-\infty}^{\infty} \left[\int_{-T/2}^{T/2} g(t) e^{-i2\pi f_n t}\, dt \right] e^{i2\pi f_n t} \Delta f$$

gives a double integral

$$g(t) = \int_{-\infty}^{\infty} \underbrace{\left[\int_{-\infty}^{\infty} g(t) e^{-i2\pi f t}\, dt \right]}_{G(f)} e^{i2\pi f t}\, df$$

The dt is for the time domain and the df for the frequency domain. Hence, the **Fourier transform** goes from the time domain into the frequency domain, given by

$$G(f) = \int_{-\infty}^{\infty} g(t) e^{-i2\pi f t}\, dt$$

The **inverse Fourier transform** goes from the frequency domain into the time domain

$$g(t) = \int_{-\infty}^{\infty} G(f) e^{i2\pi f t}\, df$$

And the **Fourier coefficients** are as before

$$C_n = \frac{1}{T} \int_0^T g(t) e^{-i2\pi f_n t}\, dt = \frac{1}{T} \int_0^T g(t) e^{-in\omega t}\, dt$$

Notice the incredible similarity between the coefficients and the transform. Note the following:

- The coefficients give the *amplitude* of each component wave.
- The transform gives the *area* of component waves of frequency f. You can see this because the transform does not have the divide by T in it.
- The transform gives for any frequency f, the rate of occurrence of the component wave with that frequency, *relative* to other waves.
- In short, the Fourier transform breaks a complicated, aperiodic wave into simple periodic ones.

The spectrum of a wave is a graph showing its component frequencies, i.e. the quantity in which they occur. It is the frequency components of the waves. But it does not give their amplitudes.

12.4.1 Empirical Example

We can use the Fourier transform function in R to compute the main component frequencies of the times series of interest rate data as follows:

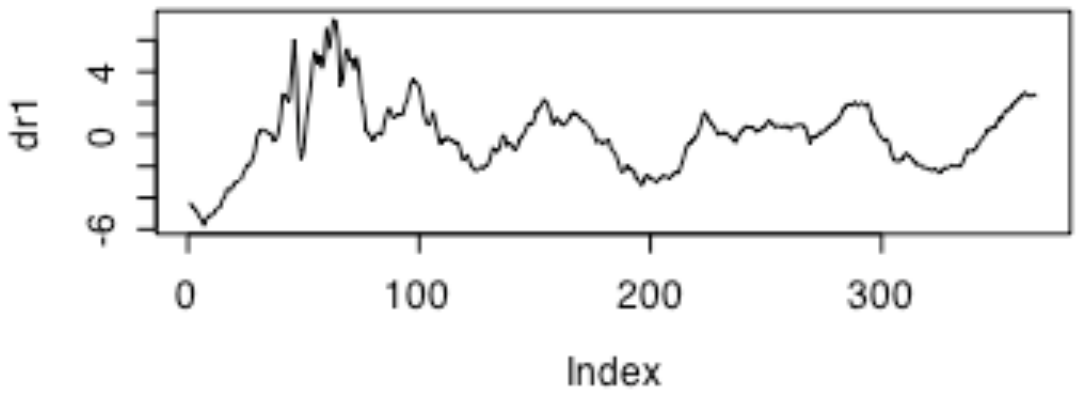
```
> rd = read.table("tryrates.txt",header=TRUE)
> r1 = as.matrix(rd[4])
> plot(r1,type="l")
> dr1 = resid(lm(r1 ~ seq(along = r1)))
> plot(dr1,type="l")
> y=fft(dr1)
> plot(abs(y),type="l")
```

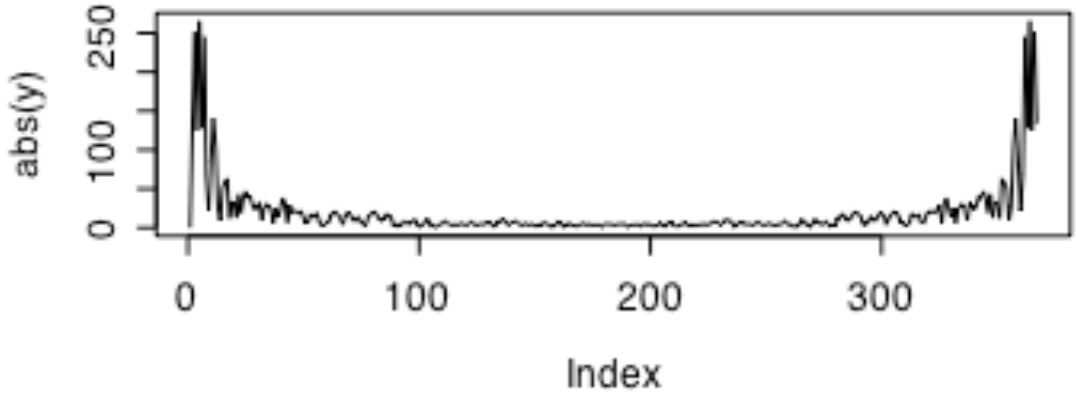
The line with

```
dr1 = resid(lm(r1 ~ seq(along = r1)))
```

detrends the series, and when we plot it we see that its done. We can then subject the detrended line to fourier analysis.

The plot of the fit of the detrended one-year interest rates is here:





Its easy to see that the series has short frequencies and long frequencies. Essentially there are two factors. If we do a factor analysis of interest rates, it turns out we get two factors as well.

12.5 Application to Binomial Option Pricing

To implement the option pricing in Cerny, Exhibit 8.

```
> ifft = function(x) { fft(x,inverse=TRUE)/length(x) }
> ct = c(599.64,102,0,0)
> q = c(0.43523,0.56477,0,0)
> R = 1.0033
> ifft(fft(ct)*( (4*ifft(q)/R)^3) )
[1]  81.36464+0i 115.28447-0i 265.46949+0i 232.62076-0i
```

12.6 Application to probability functions

12.6.1 Characteristic functions

A characteristic function of a variable x is given by the expectation of the following function of x:

$$\phi(s) = E[e^{isx}] = \int_{-\infty}^{\infty} e^{isx} f(x)\, dx$$

where $f(x)$ is the probability density of x. By Taylor series for e^{isx} we have

$$\begin{aligned}
\int_{-\infty}^{\infty} e^{isx} f(x)\, dx &= \int_{-\infty}^{\infty} [1 + isx + \frac{1}{2}(isx)^2 + \ldots] f(x) dx \\
&= \sum_{j=0}^{\infty} \frac{(is)^j}{j!} m_j \\
&= 1 + (is)m_1 + \frac{1}{2}(is)^2 m_2 + \frac{1}{6}(is)^3 m_3 + \ldots
\end{aligned}$$

where m_j is the j-th moment.

It is therefore easy to see that

$$m_j = \frac{1}{i^j} \left[\frac{d\phi(s)}{ds} \right]_{s=0}$$

where $i = \sqrt{-1}$.

12.6.2 Finance application

In a paper in 1993, Steve Heston developed a new approach to valuing stock and foreign currency options using a Fourier inversion technique. See also Duffie, Pan and Singleton (2001) for extension to jumps, and Chacko and Das (2002) for a generalization of this to interest-rate derivatives with jumps.

Lets explore a much simpler model of the same so as to get the idea of how we can get at probability functions if we are given a stochastic process for any security. When we are thinking of a dynamically moving financial variable (say x_t), we are usually interested in knowing what the probability is of this variable reaching a value x_τ at time $t = \tau$, given that right now, it has value x_0 at time $t = 0$. Note that τ is the remaining time to maturity.

Suppose we have the following financial variable x_t following a very simple Brownian motion, i.e.

$$dx_t = \mu\, dt + \sigma\, dz_t$$

Here, μ is known as its "drift" and "sigma" is the volatility. The differential equation above gives the movement of the variable x and the term dz is a Brownian motion, and is a random variable with normal distribution of mean zero, and variance dt.

What we are interested in is the characteristic function of this process. The characteristic function of x is defined as the Fourier transform of x, i.e.

$$F(x) = E[e^{isx}] = \int e^{isx} f(x) ds$$

where s is the Fourier variable of integration, and $i = \sqrt{-1}$, as usual. Notice the similarity to the Fourier transforms described earlier in the note. It turns out that once we have the characteristic function, then we can obtain by simple calculations the probability function for x, as well as all the moments of x.

12.6.3 Solving for the characteristic function

We write the characteristic function as $F(x, \tau; s)$. Then, using Ito's lemma we have

$$dF = F_x dx + \frac{1}{2} F_{xx} (dx)^2 - F_\tau dt$$

F_x is the first derivative of F with respect to x; F_{xx} is the second derivative, and F_τ is the derivative with respect to remaining maturity. Since F is a characteristic (probability) function, the expected change in F is zero.

$$E(dF) = \mu F_x \, dt + \frac{1}{2} \sigma^2 F_{xx} \, dt - F_\tau \, dt = 0$$

which gives a PDE in (x, τ):

$$\mu F_x + \frac{1}{2} \sigma^2 F_{xx} - F_\tau = 0$$

We need a boundary condition for the characteristic function which is

$$F(x, \tau = 0; s) = e^{isx}$$

We solve the PDE by making an educated guess, which is

$$F(x, \tau; s) = e^{isx + A(\tau)}$$

which implies that when $\tau = 0$, $A(\tau = 0) = 0$ as well. We can see that

$$\begin{aligned} F_x &= isF \\ F_{xx} &= -s^2 F \\ F_\tau &= A_\tau F \end{aligned}$$

Substituting this back in the PDE gives

$$\begin{aligned}
\mu isF - \frac{1}{2}\sigma^2 s^2 F - A_\tau F &= 0 \\
\mu is - \frac{1}{2}\sigma^2 s^2 - A_\tau &= 0 \\
\frac{dA}{d\tau} &= \mu is - \frac{1}{2}\sigma^2 s^2 \\
\text{gives: } A(\tau) &= \mu is\tau - \frac{1}{2}\sigma^2 s^2 \tau, \quad \text{because } A(0) = 0
\end{aligned}$$

Thus we finally have the characteristic function which is

$$F(x, \tau; s) = \exp[isx + \mu is\tau - \frac{1}{2}\sigma^2 s^2 \tau]$$

12.6.4 Computing the moments

In general, the moments are derived by differentiating the characteristic function y s and setting $s = 0$. The k-th moment will be

$$E[x^k] = \frac{1}{i^k}\left[\frac{\partial^k F}{\partial s^k}\right]_{s=0}$$

Lets test it by computing the mean ($k = 1$):

$$E(x) = \frac{1}{i}\left[\frac{\partial F}{\partial s}\right]_{s=0} = x + \mu\tau$$

where x is the current value x_0. How about the second moment?

$$E(x^2) = \frac{1}{i^2}\left[\frac{\partial^2 F}{\partial s^2}\right]_{s=0} = \sigma^2\tau + (x + \mu\tau)^2 = \sigma^2\tau + E(x)^2$$

Hence, the variance will be

$$Var(x) = E(x^2) - E(x)^2 = \sigma^2\tau + E(x)^2 - E(x^2) = \sigma^2\tau$$

12.6.5 Probability density function

It turns out that we can "invert" the characteristic function to get the pdf (boy, this characteristic function sure is useful!). Again we use Fourier inversion, which result is stated as follows:

$$f(x_\tau|x_0) = \frac{1}{\pi}\int_0^\infty Re[e^{-isx_\tau}]F(x_0, \tau; s)\, ds$$

Here is an implementation

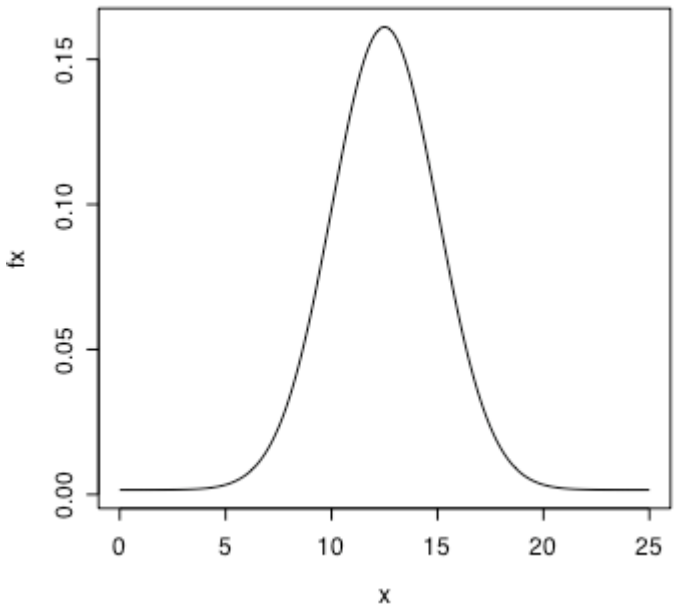
```
#Model for fourier inversion for arithmetic brownian motion

x0 = 10
mu = 10
sig = 5
tau = 0.25
s = (0:10000)/100
ds = s[2]-s[1]

phi =  exp(1i*s*x0+mu*1i*s*tau-0.5*s^2*sig^2*tau)

x = (0:250)/10
fx=NULL
for ( k in 1:length(x) ) {
        g = sum(as.real(exp(-1i*s*x[k]) * phi * ds))/pi
        print(c(x[k],g))
        fx = c(fx,g)
        }

plot(x,fx,type="l")
```



13
Making Connections: Network Theory

13.1 Overview

The science of networks is making deep inroads into business. The term "network effect" is being used widely in conceptual terms to define the gains from piggybacking on connections in the business world. Using the network to advantage coins the verb "networking" - a new, improved use of the word "schmoozing". The science of viral marketing and word-of-mouth transmission of information is all about exploiting the power of networks. We are just seeing the beginning - as the cost of the network and its analysis drops rapidly, businesses will exploit them more and more.

Networks are also useful in understanding how information flows in markets. Network theory is also being used by firms to find "communities" of consumers so as to partition and focus their marketing efforts. There are many wonderful videos by Cornell professor Jon Kleinberg on `YouTube` and elsewhere on the importance of new tools in computer science for understanding social networks. He talks of the big difference today in that networks grow organically, not in structured fashion as in the past with road, electricity and telecommunication networks. Modern networks are large, realistic and well-mapped. Think about dating networks and sites like `Linked In`. A free copy of Kleinberg's book on networks with David Easley may be downloaded at `http://www.cs.cornell.edu/home/kleinber/networks-book/`. It is written for an undergraduate audience and is immensely accessible. There is also material on game theory and auctions in this book.

13.2 Graph Theory

Any good understanding of networks must perforce begin with a digression in graph theory. I say digression because its not clear to me yet how a formal understanding of graph theory should be taught to business students, but yet, an informal set of ideas is hugely useful in providing a technical/conceptual framework within which to see how useful network analysis will be in the coming future of a changing business landscape. Also, it is useful to have a light introduction to the notation and terminology in graph theory so that the basic ideas are accessible when reading further.

What is a graph? It is a picture of a network, a diagram consisting of relationships between entities. We call the entities as vertices or nodes (set V) and the relationships are called the edges of a graph (set E). Hence a graph G is defined as

$$G = (V, E)$$

If the edges $e \in E$ of a graph are not tipped with arrows implying some direction or causality, we call the graph an "undirected" graph. If there are arrows of direction then the graph is a "directed" graph.

If the connections (edges) between vertices $v \in V$ have weights on them, then we call the graph a "weighted graph" else it's "unweighted". In an unweighted graph, for any pair of vertices (u, v), we have

$$w(u,v) = \begin{cases} w(u,v) = 1, & \text{if } (u,v) \in E \\ w(u,v) = 0, & \text{if } (u,v) \ni E \end{cases}$$

In a weighted graph the value of $w(u, v)$ is unrestricted, and can also be negative.

Directed graphs can be cyclic or acyclic. In a cyclic graph there is a path from a source node that leads back to the node itself. Not so in an acyclic graph. The term "dag" is used to connote a "directed acyclic graph". The binomial option pricing model in finance that you have learnt is an example of a dag.

A graph may be represented by its adjacency matrix. This is simply the matrix $A = \{w(u, v)\}, \forall u, v$. You can take the transpose of this matrix as well, which in the case of a directed graph will simply reverse the direction of all edges.

13.3 *Features of Graphs*

Graphs have many attributes, such as the number of nodes, and the distribution of links across nodes. The structure of nodes and edges (links) determines how connected the nodes are, how flows take place on the network, and the relative importance of each node.

One simple bifurcation of graphs suggests two types: (a) random graphs and (b) scale-free graphs. In a beautiful article in the Scientific American, Barabasi and Bonabeau (2003) presented a simple schematic to depict these two categories of graphs. See Figure 13.1.

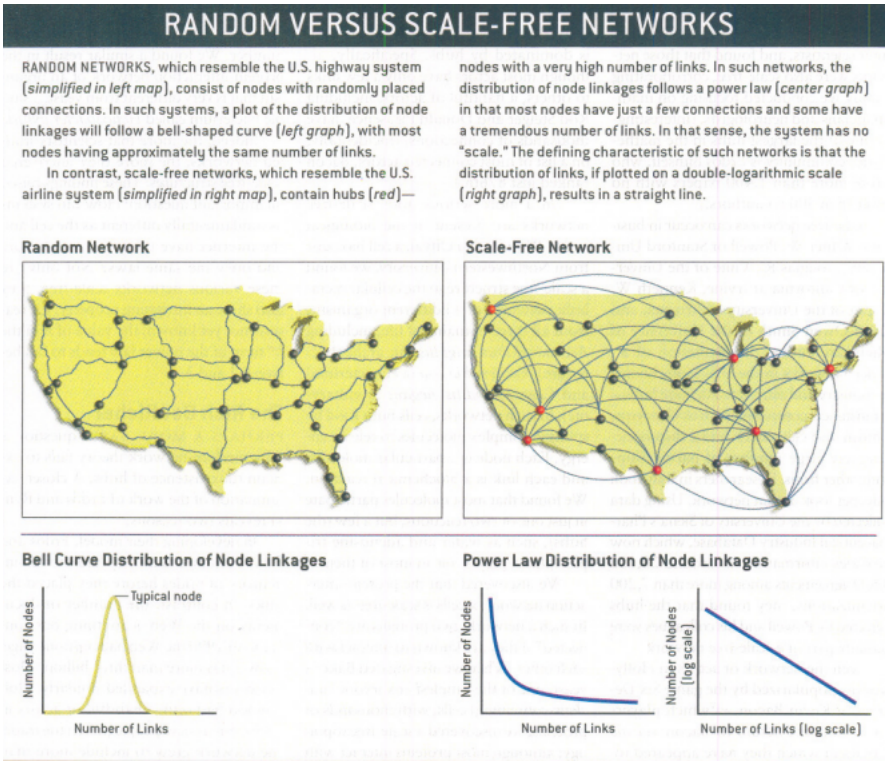


Figure 13.1: Comparison of random and scale-free graphs. From Barabasi, Albert-Laszlo., and Eric Bonabeau (2003). "Scale-Free Networks," *Scientific American* May, 50–59.

A random graph may be created by putting in place a set of n nodes and then randomly connecting pairs of nodes with some probability p. The higher this probability the more edges the graph will have. The distribution of the number of edges each node has will be more or less Gaussian as there is a mean number of edges ($n \cdot p$), with some range around the mean. In Figure 13.1, the graph on the left is a depiction of this, and the distribution of links is shown to be bell-shaped. The left graph is exemplified by the US highway network, as shown in simplified

form. If the number of links of a node are given by a number d, the distribution of nodes in a random graph would be $f(d) \sim N(\mu, \sigma^2)$, where μ is the mean number of nodes with variance σ^2.

A scale-free graph has a hub and spoke structure. There are a few central nodes that have a large number of links, and most nodes have very few. The distribution of links is shown on the right side of Figure 13.1, and an exemplar is the US airport network. This distribution is not bell-shaped at all, and appears to be exponential. There is of course a mean for this distribution, but the mean is not really representative of the hub nodes or the non-hub nodes. Because the mean, i.e., the parameter of scale is unrepresentative of the population, the distribution is scale-free, and the networks of this type are also known as scale-free networks. The distribution of nodes in a scale-free graph tends to be approximated by a power-law distribution, i.e., $f(d) \sim d^{-\alpha}$, where usually, nature seems to have stipulated that $2 \leq \alpha \leq 3$, by some curious twist of fate. The log-log plot of this distribution is linear, as shown in the right side graph in Figure 13.1.

The vast majority of networks in the world tend to be scale-free. Why? Barabasi and Albert (1999) developed the Theory of Preferential Attachment to explain this phenomenon. The theory is intuitive, and simply states that as a network grows and new nodes are added, the new nodes tend to attach to existing nodes that have the most links. Thus influential nodes become even more connected, and this evolves into a hub and spoke structure.

The structure of these graphs determines other properties. For instance, scale-free graphs are much better at transmission of information, for example. Or for moving air traffic passengers, which is why our airports are arranged thus. But a scale-free network is also susceptible to greater transmission of disease, as is the case with networks of people with HIV. Or, economic contagion. Later in this chapter we will examine financial network risk by studying the structure of banking networks. Scale-free graphs are also more robust to random attacks. If a terrorist group randomly attacks an airport, then unless it hits a hub, very little damage is done. But the network is much more risky when targeted attacks take place. Which is why our airports and the electricity grid are at so much risk.

There are many interesting graphs, where the study of basic properties leads to many quick insights, as we will see in the rest of this chap-

ter. Our of interest, if you are an academic, take a look at Microsoft's academic research network. See `http://academic.research.microsoft.com/` Using this I have plotted my own citation and co-author network in Figure 13.2.

13.4 Searching Graphs

There are two types of search algorithms that are run on graphs - depth-first-search (DFS) and breadth-first search (BFS). Why do we care about this? As we will see, DFS is useful in finding communities in social networks. And BFS is useful in finding the shortest connections in networks. Ask yourself, what use is that? It should not be hard to come up with many answers.

13.4.1 Depth First Search

DFS begins by taking a vertex and creating a tree of connected vertices from the source vertex, recursing downwards until it is no longer possible to do so. See Figure 13.3 for an example of a DFS.

The algorithm for DFS is as follows:

```
function DFS(u):
    for all v in SUCC(u):
        if notvisited(v):
            DFS(v)
    MARK(u)
```

This recursive algorithm results in two subtrees, which are:

$$a \to b \to c \to \begin{matrix} \nearrow & f \\ g & \\ \searrow & d \end{matrix}$$

$$e \to h \to i$$

The numbers on the nodes show the sequence in which the nodes are accessed by the program. The typical output of a DFS algorithm is usually slightly less detailed, and gives a simple sequence in which the nodes are first visited. An example is provided in the `graph` package:

```
> library(graph)
```

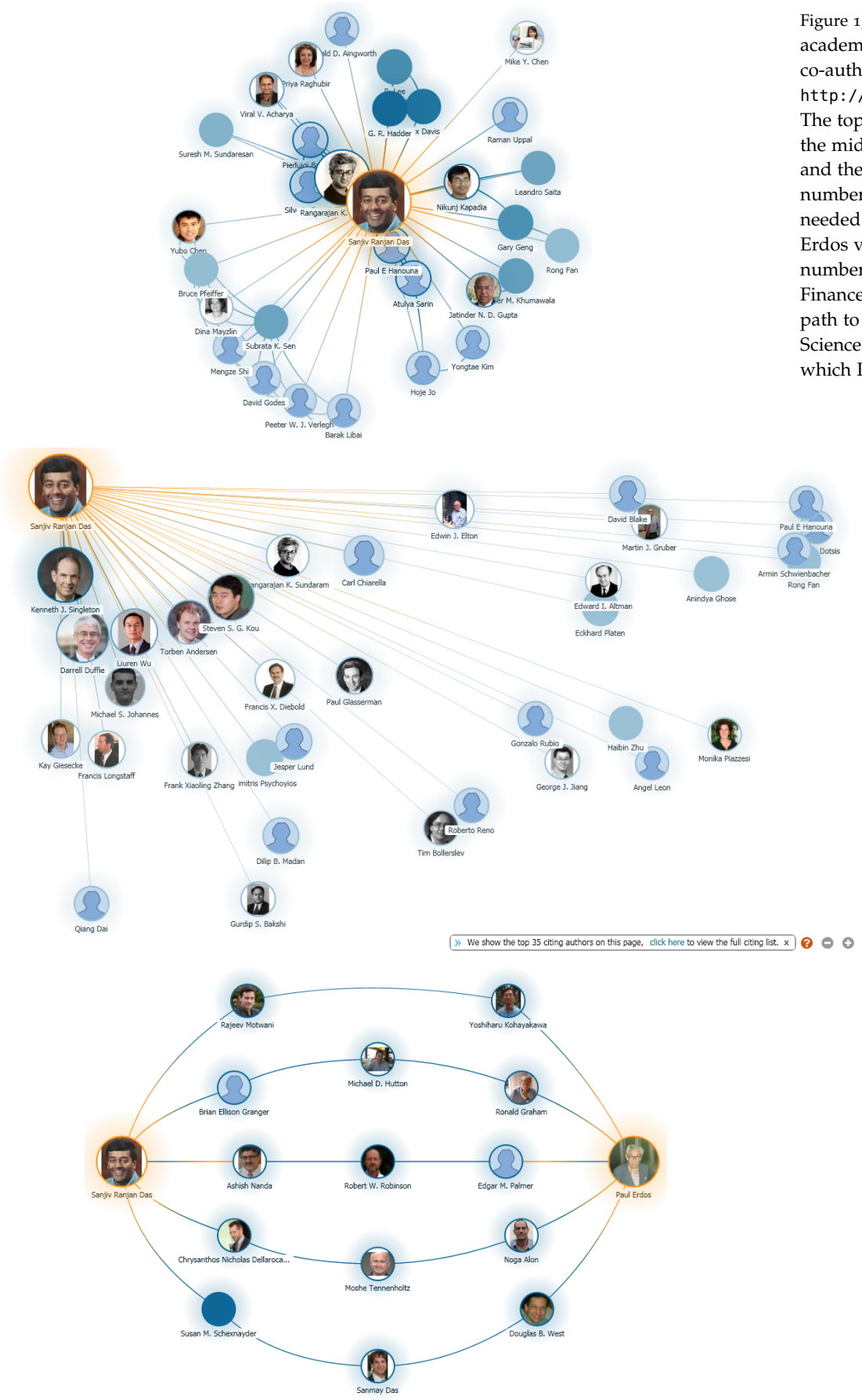


Figure 13.2: Microsoft academic search tool for co-authorship networks. See: `http://academic.research.microsoft` The top chart shows co-authors, the middle one shows citations, and the last one shows my Erdos number, i.e., the number of hops needed to be connected to Paul Erdos via my co-authors. My 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.

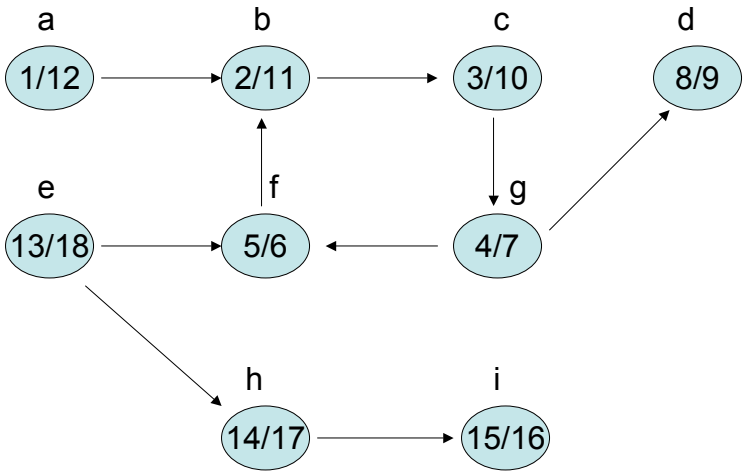


Figure 13.3: Depth-first-search.

```
> RNGkind("Mersenne-Twister")
> set.seed(123)
> g1 <- randomGraph(letters[1:10], 1:4, p=.3)
> g1
A graphNEL graph with undirected edges
Number of Nodes = 10
Number of Edges = 21
> edgeNames(g1)
 [1] "a~g" "a~i" "a~b" "a~d" "a~e" "a~f" "a~h" "b~f" "b~j"
[10] "b~d" "b~e" "b~h" "c~h" "d~e" "d~f" "d~h" "e~f" "e~h"
[19] "f~j" "f~h" "g~i"
> RNGkind()
[1] "Mersenne-Twister" "Inversion"
> DFS(g1,"a")
a b c d e f g h i j
0 1 6 2 3 4 8 5 9 7
```

Note that the result of a DFS on a graph is a set of trees. A tree is a special kind of graph, and is inherently acyclic if the graph is acyclic. A cyclic graph will have a DFS tree with back edges.

We can think of this as partitioning the vertices into subsets of connected groups. The obvious business application comes from first understanding why they are different, and secondly from being able to target these groups separately by tailoring business responses to their characteristics, or deciding to stop focusing on one of them.

Firms that maintain data about these networks use algorithms like this to find out "communities". Within a community, the nearness of connections is then determined using breadth-first-search.

A DFS also tells you something about the connectedness of the nodes. It shows that every entity in the network is not that far from the others, and the analysis often suggests the "small-world's" phenomenon, or what is colloquially called "six degrees of separation." Social networks are extremely rich in short paths.

Now we examine how DFS is implemented in the package `igraph`, which we will use throughout the rest of this chapter. Here is the sample code, which also shows how a graph may be created from a paired-vertex list.

```
#CREATE A SIMPLE GRAPH
df = matrix(c("a","b","b","c","c","g",
              "f","b","g","d","g","f",
              "f","e","e","h","h","i"),ncol=2,byrow=TRUE)

g = graph.data.frame(df,directed=FALSE)
plot(g)

#DO DEPTH–FIRST SEARCH
dfs(g,"a")

$root
[1] 0

$neimode
[1] "out"

$order
+ 9/9 vertices, named:
[1] a b c g f e h i d

$order.out
NULL

$father
NULL
```

```
$dist
NULL
```

We also plot the graph to see what it appears like and to verify the results. See Figure 13.4.

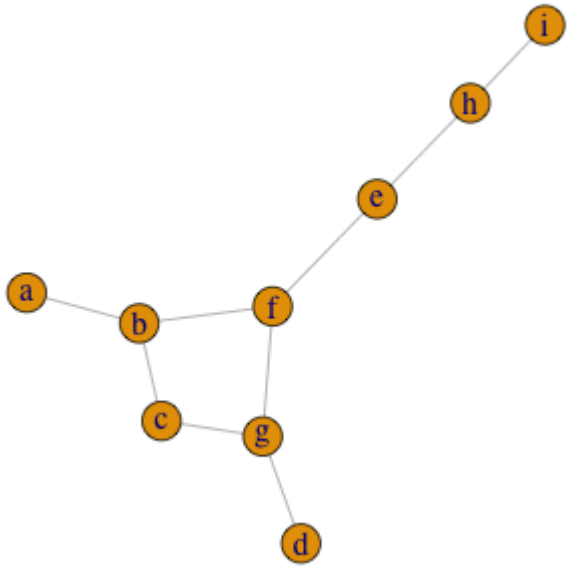


Figure 13.4: Depth-first search on a simple graph generated from a paired node list.

13.4.2 *Breadth-first-search*

BFS explores the edges of E to discover (from a source vertex s) all reachable vertices on the graph. It does this in a manner that proceeds to find a frontier of vertices k distant from s. Only when it has located all such vertices will the search then move on to locating vertices $k+1$ away from the source. This is what distinguishes it from DFS which goes all the way down, without covering all vertices at a given level first.

BFS is implemented by just labeling each node with its distance from the source. For an example, see Figure 13.5. It is easy to see that this helps in determining nearest neighbors. When you have a positive response from someone in the population it helps to be able to target the nearest neighbors first in a cost-effective manner. The art lies in defining the edges (connections). For example, a company like Schwab might be able to establish a network of investors where the connections are based on some threshold level of portfolio similarity. Then, if a certain account

displays enhanced investment, and we know the cause (e.g. falling interest rates) then it may be useful to market funds aggressively to all connected portfolios with a BFS range.

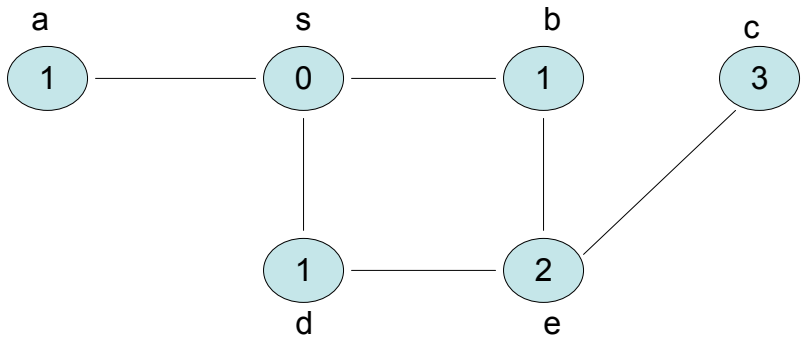


Figure 13.5: Breadth-first-search.

The algorithm for BFS is as follows:

```
function BFS(s)
    MARK(s)
    Q = {s}
    T = {s}
    While Q ne { }:
        Choose u in Q
        Visit u
        for each v=SUCC(u):
            MARK(v)
            Q = Q + v
            T = T + (u,v)
```

BFS also results in a tree which in this case is as follows. The level of each tree signifies the distance from the source vertex.

$$\begin{array}{ccc} & \nearrow & b \searrow \\ s & \rightarrow & d \rightarrow e \rightarrow c \\ & \searrow & a \end{array}$$

The code is as follows:

```
df = matrix(c("s","a","s","b","s","d","b","e","d","e","e","c"),
            ncol=2,byrow=TRUE)
g = graph.data.frame(df,directed=FALSE)
```

```
bfs(g,"a")

$root
[1] 1

$neimode
[1] "out"

$order
+ 6/6 vertices, named:
[1] s b d a e c

$rank
NULL

$father
NULL

$pred
NULL

$succ
NULL

$dist
NULL
```

There is a classic book on graph theory which is a must for anyone interested in reading more about this: Tarjan (1983) – Its only a little over 100 pages and is a great example of a lot if material presented very well.

Another bible for reference is "Introduction to Algorithms" by Cormen, Liserson, and Rivest (2009). You might remember that Ron Rivest is the "R" in the famous RSA algorithm used for encryption.

13.5 Strongly Connected Components

Directed graphs are wonderful places in which to cluster members of a network. We do this by finding strongly connected components (SCCs) on such a graph. A SCC is a subgroup of vertices $U \subset V$ in a graph with

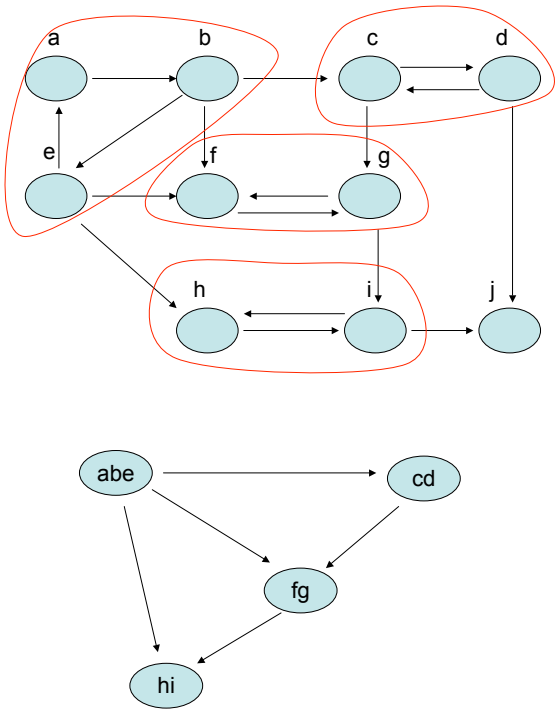


Figure 13.6: Strongly connected components. The upper graph shows the original network and the lower one shows the compressed network comprising only the SCCs. The algorithm to determine SCCs relies on two DFSs. Can you see a further SCC in the second graph? There should not be one.

the property that for all pairs of its vertices $(u, v) \in U$, both vertices are reachable from each other.

Figure 13.6 shows an example of a graph broken down into its strongly connected components.

The SCCs are extremely useful in partitioning a graph into tight units. It presents local feedback effects. What it means is that targeting any one member of a SCC will effectively target the whole, as well as move the stimulus across SCCs.

The most popular package for graph analysis has turned out to be `igraph`. It has versions in R, C, and Python. You can generate and plot random graphs in R using this package. Here is an example.

```
> library(igraph)
> g <- erdos.renyi.game(20, 1/20)
> g
Vertices: 20
Edges: 8
Directed: FALSE
Edges:
```

```
[0]  6 --  7
[1]  0 -- 10
[2]  0 -- 11
[3] 10 -- 14
[4]  6 -- 16
[5] 11 -- 17
[6]  9 -- 18
[7] 16 -- 19
> clusters(g)
$membership
 [1]  0  1  2  3  4  5  6  6  7  8  0  0  9 10  0 11  6  0  8
[20]  6

$csize
 [1] 5 1 1 1 1 1 4 1 2 1 1 1

$no
[1] 12

> plot.igraph(g)
```

It results in the plot in Figure 13.7.

13.6 *Dijkstra's Shortest Path Algorithm*

This is one of the most well-known algorithms in theoretical computer science. Given a source vertex on a weighted, directed graph, it finds the shortest path to all other nodes from source s. The weight between two vertices is denoted $w(u, v)$ as before. Dijkstra's algorithm works for graphs where $w(u, v) \geq 0$. For negative weights, there is the Bellman-Ford algorithm. The algorithm is as follows.

```
function DIJKSTRA(G,w,s)
S = { }
%S = Set of vertices whose shortest paths from
%source s have been found
Q = V(G)
while Q notequal { } :
    u = getMin(Q)
    S = S + u
```

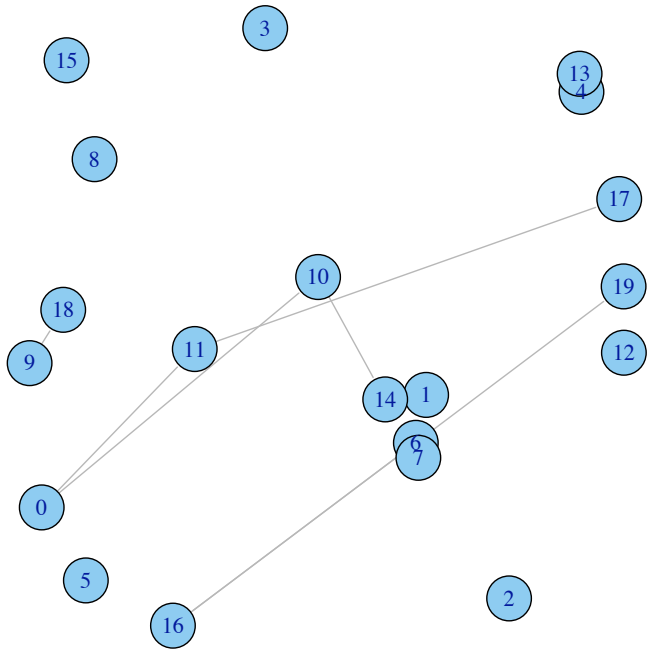


Figure 13.7: Finding connected components on a graph.

```
    Q = Q − u
    for each vertex v in SUCC(u):
        if d[v] > d[u]+w(u,v) then:
            d[v] = d[u]+w(u,v)
            PRED(v) = u
```

An example of a graph on which Dijkstra's algorithm has been applied is shown in Figure 13.8.

The usefulness of this has been long exploited in operations for airlines, designing transportation plans, optimal location of health-care centers, and in the every day use of map-quest.

You can use `igraph` to determine shortest paths in a network. Here is an example using the package. First we see how to enter a graph, then process it for shortest paths.

```
> el = matrix(nc=3, byrow=TRUE, c(0,1,8, 0,3,4, 1,3,3, 3,1,1, 1,2,1,
                                  1,4,7, 3,4,4, 2,4,1))
> el
     [,1] [,2] [,3]
[1,]    0    1    8
[2,]    0    3    4
[3,]    1    3    3
[4,]    3    1    1
[5,]    1    2    1
[6,]    1    4    7
[7,]    3    4    4
```

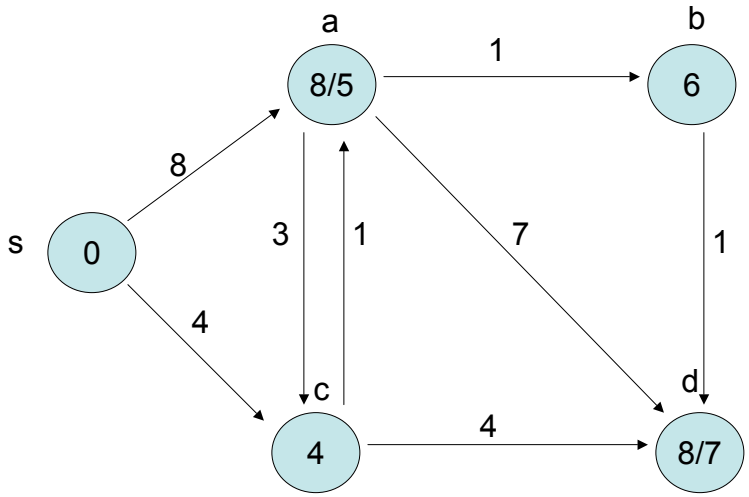


Figure 13.8: Dijkstra's algorithm.

```
[8,]    2    4    1
> g = add.edges(graph.empty(5), t(el[,1:2]), weight=el[,3])
> shortest.paths(g)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    5    6    4    7
[2,]    5    0    1    1    2
[3,]    6    1    0    2    1
[4,]    4    1    2    0    3
[5,]    7    2    1    3    0
> get.shortest.paths(g,0)
[[1]]
[1] 0

[[2]]
[1] 0 3 1

[[3]]
[1] 0 3 1 2

[[4]]
[1] 0 3

[[5]]
[1] 0 3 1 2 4
```

Here is another example.

```
> el <- matrix(nc=3, byrow=TRUE,
              c(0,1,0, 0,2,2, 0,3,1, 1,2,0, 1,4,5, 1,5,2, 2,1,1, 2,3,1,
                2,6,1, 3,2,0, 3,6,2, 4,5,2, 4,7,8, 5,2,2, 5,6,1, 5,8,1,
                5,9,3, 7,5,1, 7,8,1, 8,9,4) )
> el
      [,1] [,2] [,3]
 [1,]    0    1    0
 [2,]    0    2    2
 [3,]    0    3    1
 [4,]    1    2    0
 [5,]    1    4    5
 [6,]    1    5    2
```

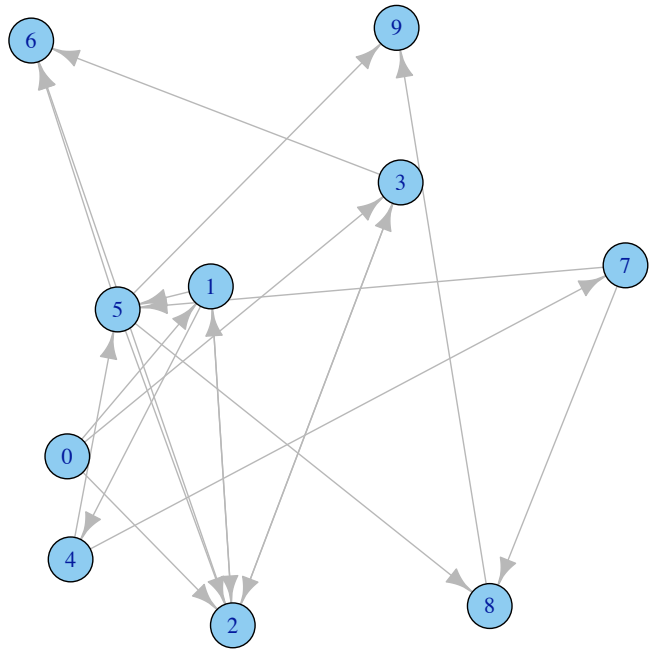


Figure 13.9: Network for computation of shortest path algorithm

```
 [7,]    2    1    1
 [8,]    2    3    1
 [9,]    2    6    1
[10,]    3    2    0
[11,]    3    6    2
[12,]    4    5    2
[13,]    4    7    8
[14,]    5    2    2
[15,]    5    6    1
[16,]    5    8    1
[17,]    5    9    3
[18,]    7    5    1
[19,]    7    8    1
[20,]    8    9    4
> g = add.edges(graph.empty(10), t(el[,1:2]), weight=el[,3])
> plot.igraph(g)
> shortest.paths(g)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    4    2    1    3    3     5
 [2,]    0    0    0    0    4    2    1    3    3     5
 [3,]    0    0    0    0    4    2    1    3    3     5
 [4,]    0    0    0    0    4    2    1    3    3     5
 [5,]    4    4    4    4    0    2    3    3    3     5
 [6,]    2    2    2    2    2    0    1    1    1     3
 [7,]    1    1    1    1    3    1    0    2    2     4
 [8,]    3    3    3    3    3    1    2    0    1     4
 [9,]    3    3    3    3    3    1    2    1    0     4
[10,]    5    5    5    5    5    3    4    4    4     0
> get.shortest.paths(g,4)
[[1]]
[1] 4 5 1 0
```

```
[[2]]
[1] 4 5 1

[[3]]
[1] 4 5 2

[[4]]
[1] 4 5 2 3

[[5]]
[1] 4

[[6]]
[1] 4 5

[[7]]
[1] 4 5 6

[[8]]
[1] 4 5 7

[[9]]
[1] 4 5 8

[[10]]
[1] 4 5 9

> average.path.length(g)
[1] 2.051724
```

13.6.1 *Plotting the network*

One can also use different layout standards as follows: Here is the example:

```
> library(igraph)
> el <- matrix(nc=3, byrow=TRUE,
+               c(0,1,0, 0,2,2, 0,3,1, 1,2,0, 1,4,5, 1,5,2, 2,1,1, 2,3,1,
+                 2,6,1, 3,2,0, 3,6,2, 4,5,2, 4,7,8, 5,2,2, 5,6,1, 5,8,1,
+                 5,9,3, 7,5,1, 7,8,1, 8,9,4) )
> g = add.edges(graph.empty(10), t(el[,1:2]), weight=el[,3])

#GRAPHING MAIN NETWORK
g = simplify(g)
V(g)$name = seq(vcount(g))
l = layout.fruchterman.reingold(g)
#l = layout.kamada.kawai(g)
# = layout.circle(g)
l = layout.norm(l, -1,1,-1,1)
#pdf(file="network_plot.pdf")
plot(g, layout=l, vertex.size=2, vertex.label=NA, vertex.color="#ff000033",
        edge.color="grey", edge.arrow.size=0.3, rescale=FALSE,
        xlim=range(l[,1]), ylim=range(l[,2]))
```

The plots are shown in Figures 13.10.

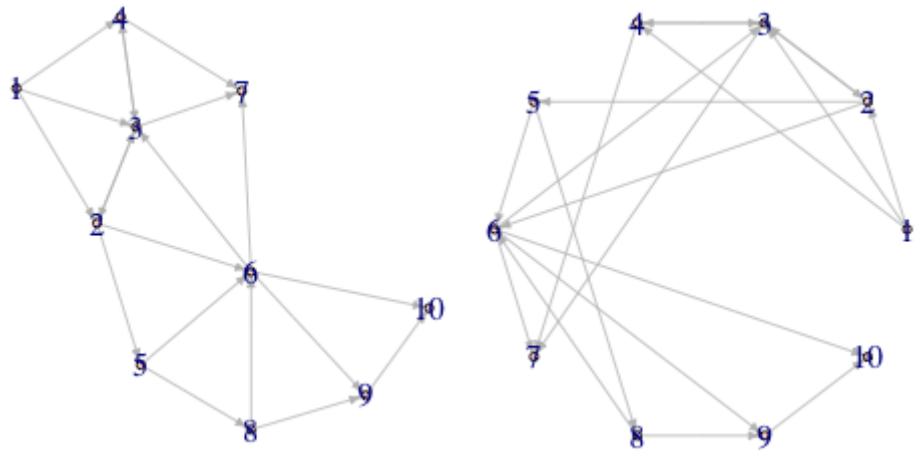


Figure 13.10: Plot using the Fruchterman-Rheingold and Circle layouts

13.7 *Degree Distribution*

The degree of a node in the network is the number of links it has to other nodes. The probability distribution of the nodes is known as the degree distribution. In an undirected network, this is based on the number of edges a node has, but in a directed network, we have a distribution for in-degree and another for out-degree. Note that the weights on the edges are not relevant for computing the degree distribution, though there may be situations in which one might choose to avail of that information as well.

```
#GENERATE RANDOM GRAPH
g = erdos.renyi.game(30,0.1)
plot.igraph(g)
print(g)

IGRAPH U--- 30 41 -- Erdos renyi (gnp) graph
+ attr: name (g/c), type (g/c), loops (g/l), p (g/n)
+ edges:
 [1]  1-- 9  2-- 9  7--10  7--12  8--12  5--13  6--14 11--14
 [9]  5--15 12--15 13--16 15--16  1--17 18--19 18--20  2--21
[17] 10--21 18--21 14--22  4--23  6--23  9--23 11--23  9--24
[25] 20--24 17--25 13--26 15--26  3--27  5--27  6--27 16--27
[33] 18--27 19--27 25--27 11--28 13--28 22--28 24--28  5--29
[41]  7--29
```

```
> clusters(g)

$membership
 [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 2

$csize
[1] 29  1

$no
[1] 2
```

The plot is shown in Figure 13.11.

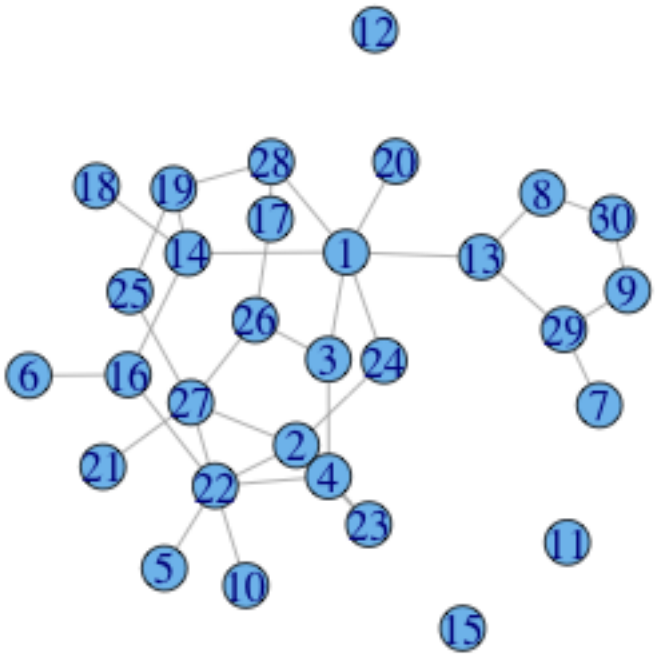


Figure 13.11: Plot of the Erdos-Renyi random graph

We may compute the degree distribution with some minimal code.

```
#COMPUTE DEGREE DISTRIBUTION
dd = degree.distribution(g)
dd = as.matrix(dd)
d = as.matrix(seq(0,max(degree(g))))
plot(d,dd,type="l",lwd=3,col="blue",ylab="Probability",xlab="Degree")

> sum(dd)
```

```
[1] 1
```

The resulting plot of the probability distribution is shown in Figure 13.12.

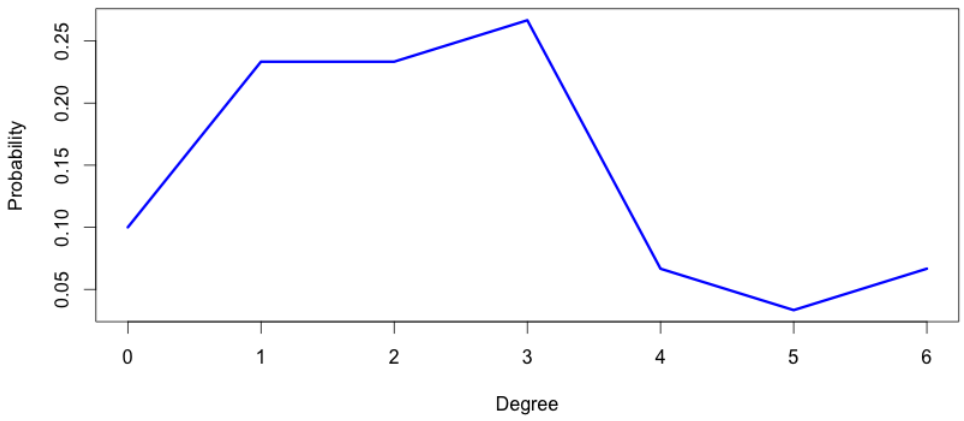


Figure 13.12: Plot of the degree distribution of the Erdos-Renyi random graph

13.8 Diameter

The diameter of a graph is the longest shortest distance between any two nodes, across all nodes. This is easily computed as follows for the graph we examined in the previous section.

```
> print(diameter(g))
[1] 7
```

We may cross-check this as follows:

```
> res = shortest.paths(g)
> res[which(res==Inf)]=-99
> max(res)
[1] 7
> length(which(res==7))
[1] 18
```

We see that the number of paths that are of length 7 are a total of 18, but of course, this is duplicated as we run these paths in both directions. Hence, there are 9 pairs of nodes that have longest shortest paths between them. You may try to locate these on Figure 13.11.

13.9 Fragility

Fragility is an attribute of a network that is based on its degree distribution. In comparing two networks of the same average degree, how do assess on which network contagion is more likely? Intuitively, a scale-free network is more likely to facilitate the spread of the variable of interest, be it flu, financial malaise, or information. In scale-free networks the greater preponderance of central hubs results in a greater probability of contagion. This is because there is a concentration of degree in a few nodes. The greater the concentration, the more scale-free the graph, and the higher the fragility.

We need a measure of concentration, and economists have used the Herfindahl-Hirschman index for many years.

(See `https://en.wikipedia.org/wiki/Herfindahl_index`.)

The index is trivial to compute, as it is the average degree squared for n nodes, i.e.,

$$H = E(d^2) = \frac{1}{n}\sum_{j=1}^{n} d_j^2 \tag{13.1}$$

This metric H increases as degree gets concentrated in a few nodes, keeping the total degree of the network constant. For example, if there is a graph of three nodes with degrees $\{1, 1, 4\}$ versus another graph of three nodes with degrees $\{2, 2, 2\}$, the former will result in a higher value of $H = 18$ than the latter with $H = 12$. If we normalize H by the average degree, then we have a definition for *fragility*, i.e.,

$$\text{Fragility} = \frac{E(d^2)}{E(d)} \tag{13.2}$$

In the three node graphs example, fragility is 3 and 2, respectively. We may also choose other normalization factors, for example, $E(d)^2$ in the denominator. Computing this is trivial and requires a single line of code, given a vector of node degrees (`d`), accompanied by the degree distribution (`dd`), computed earlier in Section 13.7.

```
#FRAGILITY
print((t(d^2) %*% dd)/(t(d) %*% dd))
```

13.10 Centrality

Centrality is a property of vertices in the network. Given the adjacency matrix $A = \{w(u, v)\}$, we can obtain a measure of the "influence" of

all vertices in the network. Let x_i be the influence of vertex i. Then the column vector x contains the influence of each vertex. What is influence? Think of a web page. It has more influence the more links it has both, to the page, and from the page to other pages. Or think of a alumni network. People with more connections have more influence, they are more "central".

It is possible that you might have no connections yourself, but are connected to people with great connections. In this case, you do have influence. Hence, your influence depends on your own influence and that which you derive through others. Hence, the entire system of influence is interdependent, and can be written as the following matrix equation

$$x = A\ x$$

Now, we can just add a scalar here to this to get

$$\xi\ x = Ax$$

an eigensystem. Decompose this to get the principle eigenvector, and its values give you the influence of each member. In this way you can find the most influential people in any network. There are several applications of this idea to real data. This is eigenvector centrality is exactly what Google trademarked as `PageRank`, even though they did not invent eigenvector centrality.

Network methods have also been exploited in understanding Venture Capitalist networks, and have been shown to be key in the success of VCs and companies. See the recent paper titled "Whom You Know Matters: Venture Capital Networks and Investment Performance" by Hochberg, Ljungqvist and Lu (2007).

Networks are also key in the Federal Funds Market. See the paper by Adam Ashcraft and Darrell Duffie, titled "Systemic Illiquidity in the Federal Funds Market," in the *American Economic Review*, Papers and Proceedings. See Ashcraft and Duffie (2007).

See the paper titled "Financial Communities" (Das and Sisk (2005)) which also exploits eigenvector methods to uncover properties of graphs. The key concept here is that of eigenvector *centrality*.

Let's do some examples to get a better idea. We will create some small networks and examine the centrality scores.

```
> A = matrix(nc=3, byrow=TRUE, c(0,1,1, 1,0,1, 1,1,0))
> A
```

```
     [,1] [,2] [,3]
[1,]    0    1    1
[2,]    1    0    1
[3,]    1    1    0
> g = graph.adjacency(A,mode="undirected",weighted=TRUE,diag=FALSE)
> res = evcent(g)
> res$vector
[1] 1 1 1
> res = evcent(g,scale=FALSE)
> res$vector
[1] 0.5773503 0.5773503 0.5773503
```

Here is another example:

```
> A = matrix(nc=3, byrow=TRUE, c(0,1,1, 1,0,0, 1,0,0))
> A
     [,1] [,2] [,3]
[1,]    0    1    1
[2,]    1    0    0
[3,]    1    0    0
> g = graph.adjacency(A,mode="undirected",weighted=TRUE,diag=FALSE)
> res = evcent(g)
> res$vector
[1] 1.0000000 0.7071068 0.7071068
```

And another...

```
> A = matrix(nc=3, byrow=TRUE, c(0,2,1, 2,0,0, 1,0,0))
> A
     [,1] [,2] [,3]
[1,]    0    2    1
[2,]    2    0    0
[3,]    1    0    0
> g = graph.adjacency(A,mode="undirected",weighted=TRUE,diag=FALSE)
> res = evcent(g)
> res$vector
[1] 1.0000000 0.8944272 0.4472136
```

Year	#Colending banks	#Coloans	Colending pairs	$R = E(d^2)/E(d)$	Diam.
2005	241	75	10997	137.91	5
2006	171	95	4420	172.45	5
2007	85	49	1793	73.62	4
2008	69	84	681	68.14	4
2009	69	42	598	35.35	4

(Year = 2005)		
Node #	Financial Institution	Normalized Centrality
143	J P Morgan Chase & Co.	1.000
29	Bank of America Corp.	0.926
47	Citigroup Inc.	0.639
85	Deutsche Bank Ag New York Branch	0.636
225	Wachovia Bank NA	0.617
235	The Bank of New York	0.573
134	Hsbc Bank USA	0.530
39	Barclays Bank Plc	0.530
152	Keycorp	0.524
241	The Royal Bank of Scotland Plc	0.523
6	Abn Amro Bank N.V.	0.448
173	Merrill Lynch Bank USA	0.374
198	PNC Financial Services Group Inc	0.372
180	Morgan Stanley	0.362
42	Bnp Paribas	0.337
205	Royal Bank of Canada	0.289
236	The Bank of Nova Scotia	0.289
218	U.S. Bank NA	0.284
50	Calyon New York Branch	0.273
158	Lehman Brothers Bank Fsb	0.270
213	Sumitomo Mitsui Banking	0.236
214	Suntrust Banks Inc	0.232
221	UBS Loan Finance Llc	0.221
211	State Street Corp	0.210
228	Wells Fargo Bank NA	0.198

Table 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.

In a recent paper I constructed the network graph of interbank lending, and this allows detection of the banks that have high centrality, and are more systemically risky. The plots of the banking network are shown in Figure 13.13. See the paper titled "Extracting, Linking and Integrating Data from Public Sources: A Financial Case Study," by Burdick et al (2011). In this paper the centrality scores for the banks are given in Table 13.1.

Another concept of centrality is known as "betweenness". This is the proportion of shortest paths that go through a node relative to all paths

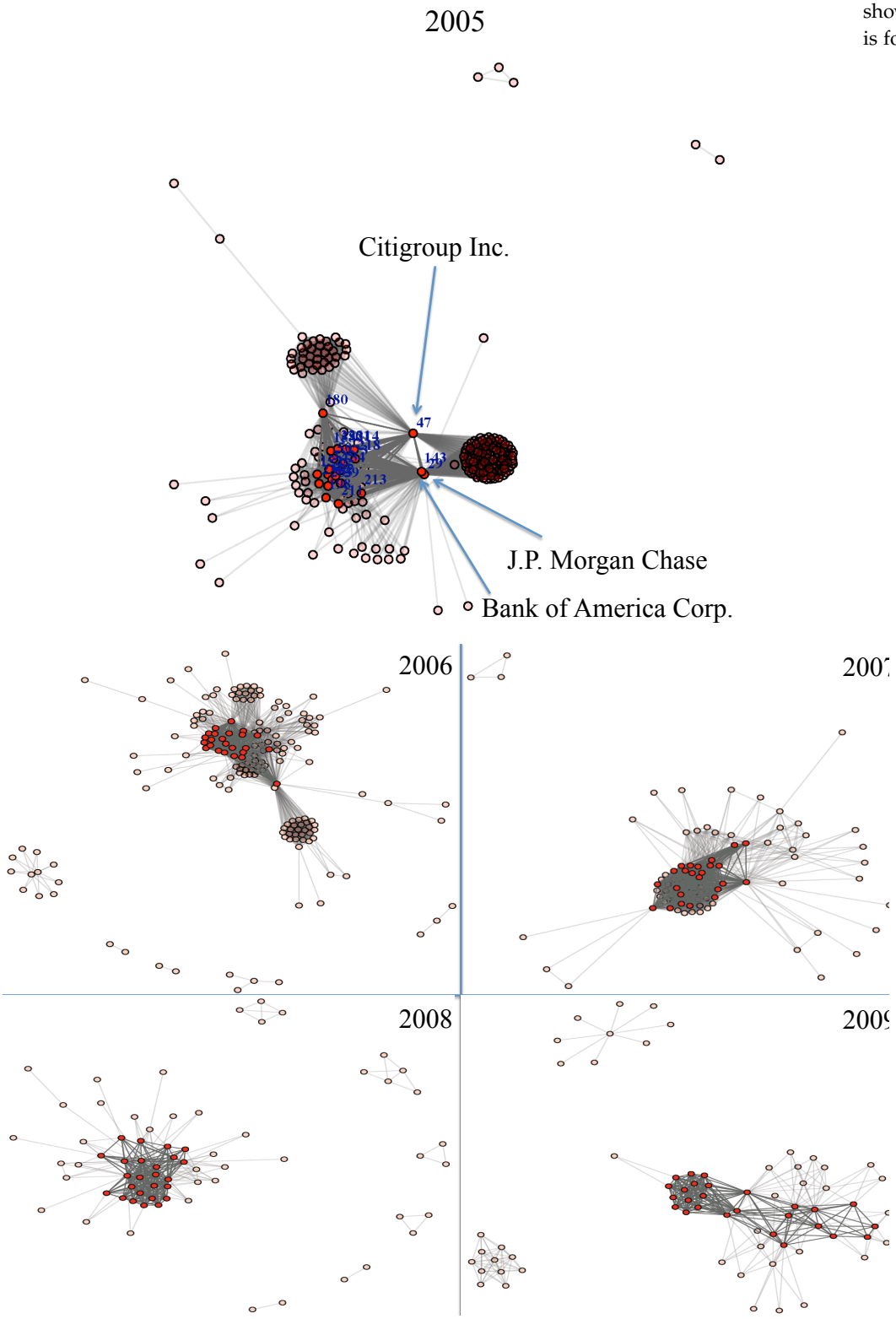


Figure 13.13: Interbank lending networks by year. The top panel shows 2005, and the bottom panel is for the years 2006-2009.

that go through the same node. This may be expressed as

$$B(v) = \sum_{a \neq v \neq b} \frac{n_{a,b}(v)}{n_{a,b}}$$

where $n_{a,b}$ is the number of shortest paths from node a to node b, and $n_{a,b}(v)$ are the number of those paths that traverse through vertex v. Here is an example from an earlier directed graph.

```
> el <- matrix(nc=3, byrow=TRUE,
+              c(0,1,0, 0,2,2, 0,3,1, 1,2,0, 1,4,5, 1,5,2, 2,1,1, 2,3,1,
+                2,6,1, 3,2,0, 3,6,2, 4,5,2, 4,7,8, 5,2,2, 5,6,1, 5,8,1,
+                5,9,3, 7,5,1, 7,8,1, 8,9,4) )
> g = add.edges(graph.empty(10), t(el[,1:2]), weight=el[,3])
> res = betweenness(g)
> res
 [1]  0.0 18.0 17.0  0.5  5.0 19.5  0.0  0.5  0.5  0.0
```

13.11 Communities

Communities are spatial agglomerates of vertexes who are more likely to connect with each other than with others. Identifying these agglomerates is a cluster detection problem, a computationally difficult (**NP**-hard) one. The computational complexity arises because we do not fix the number of clusters, allow each cluster to have a different size, and permit porous boundaries so members can communicate both within and outside their preferred clusters. Several partitions satisfy such a flexible definition. Communities are constructed by optimizing **modularity**, which is a metric of the difference between the number of within-community connections and the expected number of connections, given the total connectivity on the graph. Identifying communities is difficult because of the enormous computational complexity involved in sifting through all possible partitions. One fast way is to exploit the walk trap approach recently developed in the physical sciences (Pons and Latapy (2006), see Fortunato (2010) for a review) to identify communities.

The essential idea underlying community formation dates back at least to Simon (1962). In his view, complex systems comprising several entities often have coherent subsystems, or communities, that serve specific functional purposes. Identifying communities embedded in larger entities can help understand the functional forces underlying larger entities. To make these ideas more concrete, we discuss applications from the physical and social sciences before providing more formal definitions.

In the life sciences, community structures help understand pathways in the metabolic networks of cellular organisms (Ravasz et al. (2002); Guimera et al. (2005)). Community structures also help understand the functioning of the human brain. For instance, Wu, Taki, and Sato (2011) find that there are community structures in the human brain with predictable changes in their interlinkages related to aging. Community structures are used to understand how food chains are compartmentalized, which can predict the robustness of ecosystems to shocks that endanger particular species, Girvan and Newman (2002). Lusseau (2003) finds that communities are evolutionary hedges that avoid isolation when a member is attacked by predators. In political science, community structures discerned from voting patterns can detect political preferences that transcend traditional party lines, Porter, Mucha, Newman, and Friend (2007).[1]

Fortunato (2010) presents a relatively recent and thorough survey of the research in community detection. Fortunato points out that while the computational issues are challenging, there is sufficient progress to the point where many methods yield similar results in practice. However, there are fewer insights on the functional roles of communities or their quantitative effect on outcomes of interest. Fortunato suggests that this is a key challenge in the literature. As he concludes "... What shall we do with communities? What can they tell us about a system? This is the main question beneath the whole endeavor." Community detection methods provide useful insights into the economics of networks. See this great video on a talk by Mark Newman, who is just an excellent speaker and huge contributor to the science of network analysis: `http://www.youtube.com/watch?v=lETt7IcDWLI`, the talk is titled "What Networks Can Tell us About the World".

We represent the network as the square adjacency matrix A. The rows and columns represent entities. Element $A(i,j)$ equals the number of times node i and j are partners, so more frequent partnerships lead to greater weights. The diagonal element $A(i,i)$ is zero. While this representation is standard in the networks literature, it has economic content. The matrix is undirected and symmetric, effectively assuming that the benefits of interactions flow to all members in a symmetric way.

Community detection methods partition nodes into clusters that tend to interact together. It is useful to point out the considerable flexibility and realism built into the definition of our community clusters. We

[1] Other topics studied include social interactions and community formation (Zachary (1977)); word adjacency in linguistics and cognitive sciences, Newman (2006); collaborations between scientists (Newman (2001)); and industry structures from product descriptions, Hoberg and Phillips (2010). For some community detection datasets, see Mark Newman's website http://www-personal.umich.edu/ mejn/netdata/.

do not require all nodes to belong to communities. Nor do we fix the number of communities that may exist at a time, and we also allow each community to have different size. With this flexibility, the key computational challenge is to find the "best" partition because the number of possible partitions of the nodes is extremely large. Community detection methods attempt to determine a set of clusters that are internally tight-knit. Mathematically, this is equivalent to finding a partition of clusters to maximize the observed number of connections between cluster members minus what is expected conditional on the connections within the cluster, aggregated across all clusters. More formally (see, e.g., Newman (2006)), we choose partitions with high modularity Q, where

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i \times d_j}{2m} \right] \cdot \delta(i,j) \tag{13.3}$$

In equation (13.3), A_{ij} is the (i,j)-th entry in the adjacency matrix, i.e., the number of connections in which i and j jointly participated, $d_i = \sum_j A_{ij}$ is the total number of transactions that node i participated in (or, the degree of i) and $m = \frac{1}{2} \sum_{ij} A_{ij}$ is the sum of all edge weights in matrix A. The function $\delta(i,j)$ is an indicator equal to 1.0 if nodes i and j are from the same community, and zero otherwise. Q is bounded in [-1, +1]. If $Q > 0$, intra-community connections exceed the expected number given deal flow.

13.11.1 Modularity

In order to offer the reader a better sense of how modularity is computed in different settings, we provide a simple example here, and discuss the different interpretations of modularity that are possible. The calculations here are based on the measure developed in Newman (2006). Since we used the `igraph` package in R, we will present the code that may be used with the package to compute modularity.

Consider a network of five nodes $\{A, B, C, D, E\}$, where the edge weights are as follows: $A : B = 6$, $A : C = 5$, $B : C = 2$, $C : D = 2$, and $D : E = 10$. Assume that a community detection algorithm assigns $\{A, B, C\}$ to one community and $\{D, E\}$ to another, i.e., only two

communities. The adjacency matrix for this graph is

$$\{A_{ij}\} = \begin{bmatrix} 0 & 6 & 5 & 0 & 0 \\ 6 & 0 & 2 & 0 & 0 \\ 5 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 10 \\ 0 & 0 & 0 & 10 & 0 \end{bmatrix}$$

Let's first detect the communities.

```
> library(igraph)
> A = matrix(c(0,6,5,0,0,6,0,2,0,0,5,2,0,2,0,0,0,2,0,10,0,0,0,10,0),5,5)
> g = graph.adjacency(A,mode="undirected",diag=FALSE)
> wtc = walktrap.community(g)
> res=community.to.membership(g,wtc$merges,steps=3)
> print(res)
$membership
[1] 1 1 1 0 0

$csize
[1] 2 3
```

We can do the same thing with a different algorithm called the "fast-greedy" approach.

```
> g = graph.adjacency(A,mode="undirected",weighted=TRUE,diag=FALSE)
> fgc = fastgreedy.community(g,merges=TRUE,modularity=TRUE,
            weights=E(g)$weight)
> res = community.to.membership(g,fgc$merges,steps=3)
> res
$membership
[1] 0 0 0 1 1

$csize
[1] 3 2
```

The Kronecker delta matrix that delineates the communities will be

$$\{\delta_{ij}\} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The modularity score is

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i \times d_j}{2m} \right] \cdot \delta_{ij} \tag{13.4}$$

where $m = \frac{1}{2} \sum_{ij} A_{ij} = \frac{1}{2} \sum_i d_i$ is the sum of edge weights in the graph, A_{ij} is the (i, j)-th entry in the adjacency matrix, i.e., the weight of the edge between nodes i and j, and $d_i = \sum_j A_{ij}$ is the degree of node i. The function δ_{ij} is Kronecker's delta and takes value 1 when the nodes i and j are from the same community, else takes value zero. The core of the formula comprises the modularity matrix $\left[A_{ij} - \frac{d_i \times d_j}{2m} \right]$ which gives a score that increases when the number of connections within a community exceeds the expected proportion of connections if they are assigned at random depending on the degree of each node. The score takes a value ranging from -1 to $+1$ as it is normalized by dividing by $2m$. When $Q > 0$ it means that the number of connections within communities exceeds that between communities. The program code that takes in the adjacency matrix and delta matrix is as follows:

```
#MODULARITY
Amodularity = function(A, delta) {
    n = length(A[1,])
    d = matrix(0,n,1)
    for (j in 1:n) { d[j] = sum(A[j,]) }
        m = 0.5*sum(d)
        Q = 0
        for (i in 1:n) {
            for (j in 1:n) {
                Q = Q + (A[i,j] - d[i]*d[j]/(2*m))*delta[i,j]
            }
        }
        Q = Q/(2*m)
    }
```

We use the R programming language to compute modularity using a canned function, and we will show that we get the same result as the formula provided in the function above. First, we enter the two matrices and then call the function shown above:

```
> A = matrix(c(0,6,5,0,0,6,0,2,0,0,5,2,0,2,0,0,0,2,0,10,0,0,0,10,0),5,5)
> delta = matrix(c(1,1,1,0,0,1,1,1,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,1,1),5,5)
> print(Amodularity(A, delta))
```

```
[1] 0.4128
```

We now repeat the same analysis using the R package. Our exposition here will also show how the walktrap algorithm is used to detect communities, and then using these communities, how modularity is computed. Our first step is to convert the adjacency matrix into a graph for use by the community detection algorithm.

```
> g = graph.adjacency(A,mode="undirected",weighted=TRUE,diag=FALSE)
```

We then pass this graph to the walktrap algorithm:

```
> wtc=walktrap.community(g,modularity=TRUE,weights=E(g)$weight)
> res=community.to.membership(g,wtc$merges,steps=3)
> print(res)
$membership
[1] 0 0 0 1 1

$csize
[1] 3 2
```

We see that the algorithm has assigned the first three nodes to one community and the next two to another (look at the membership variable above). The sizes of the communities are shown in the size variable above. We now proceed to compute the modularity

```
> print(modularity(g,res$membership,weights=E(g)$weight))
[1] 0.4128
```

This confirms the value we obtained from the calculation using our implementation of the formula.

Modularity can also be computed using a graph where edge weights are unweighted. In this case, we have the following adjacency matrix

```
> A
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    1    0    0
[2,]    1    0    1    0    0
[3,]    1    1    0    1    0
[4,]    0    0    1    0    1
[5,]    0    0    0    1    0
```

Using our function, we get

```
> print(Amodularity(A,delta))
[1] 0.22
```

We can generate the same result using R:

```
> g = graph.adjacency(A,mode="undirected",diag=FALSE)
> wtc = walktrap.community(g)
> res=community.to.membership(g,wtc$merges,steps=3)
> print(res)
$membership
[1] 1 1 1 0 0

$csize
[1] 2 3

> print(modularity(g,res$membership))
[1] 0.22
```

Community detection is an **NP**-hard problem for which there are no known exact solutions beyond tiny systems (Fortunato, 2009). For larger datasets, one approach is to impose numerical constraints. For example, graph partitioning imposes a uniform community size, while partitional clustering presets the number of communities. This is too restrictive.

The less restrictive methods for community detection are called hierarchical partitioning methods, which are "divisive," or "agglomerative." The former is a top-down approach that assumes that the entire graph is one community and breaks it down into smaller units. It often produces communities that are too large especially when there is not an extremely strong community structure. Agglomerative algorithms, like the "walktrap" technique we use, begin by assuming all nodes are separate communities and collect nodes into communities. The fast techniques are dynamic methods based on random walks, whose intuition is that if a random walk enters a strong community, it is likely to spend a long time inside before finding a way out (Pons and Latapy (2006)).[2]

Community detection forms part of the literature on social network analysis. The starting point for this work is a set of pairwise connections between individuals or firms, which has received much attention in the recent finance literature. Cohen, Frazzini and Malloy (2008a); Cohen, Frazzini and Malloy (2008b) analyze educational connections between sell-side analysts and managers. Hwang and Kim (2009) and ChidKed-Prabh (2010) analyze educational, employment, and other links between CEOs and directors. Pairwise inter-firm relations are analyzed by Ishii and Xuan (2009) and Cai and Sevilir (2012), while VC firm connections

[2] See Girvan and Newman (2002), Leskovec, Kang and Mahoney (2010), or Fortunato (2010) and the references therein for a discussion.

with founders and top executives are studied by Bengtsson and Hsu (2010) and Hegde and Tumlinson (2011).

There is more finance work on the *aggregate* connectedness derived from pairwise connections. These metrics are introduced to the finance literature by Hochberg, Ljungqvist and Lu (2007), who study the aggregate connections of venture capitalists derived through syndications. They show that firms financed by well-connected VCs are more likely to exit successfully. Engelberg, Gao and Parsons (2000) show that highly connected CEOs are more highly compensated.

The simplest measure of aggregate connectedness, degree centrality, simply aggregates the number of partners that a person or node has worked with. A more subtle measure, eigenvector centrality, aggregates connections but puts more weight on the connections of nodes to more connected nodes. Other related constructs are betweenness, which reflects how many times a node is on the shortest path between two other nodes, and closeness, which measures a nodes distance to all other nodes. The important point is that each of these measures represents an attempt to capture a node's stature or influence as reflected in the number of its own connections or from being connected to well-connected nodes.

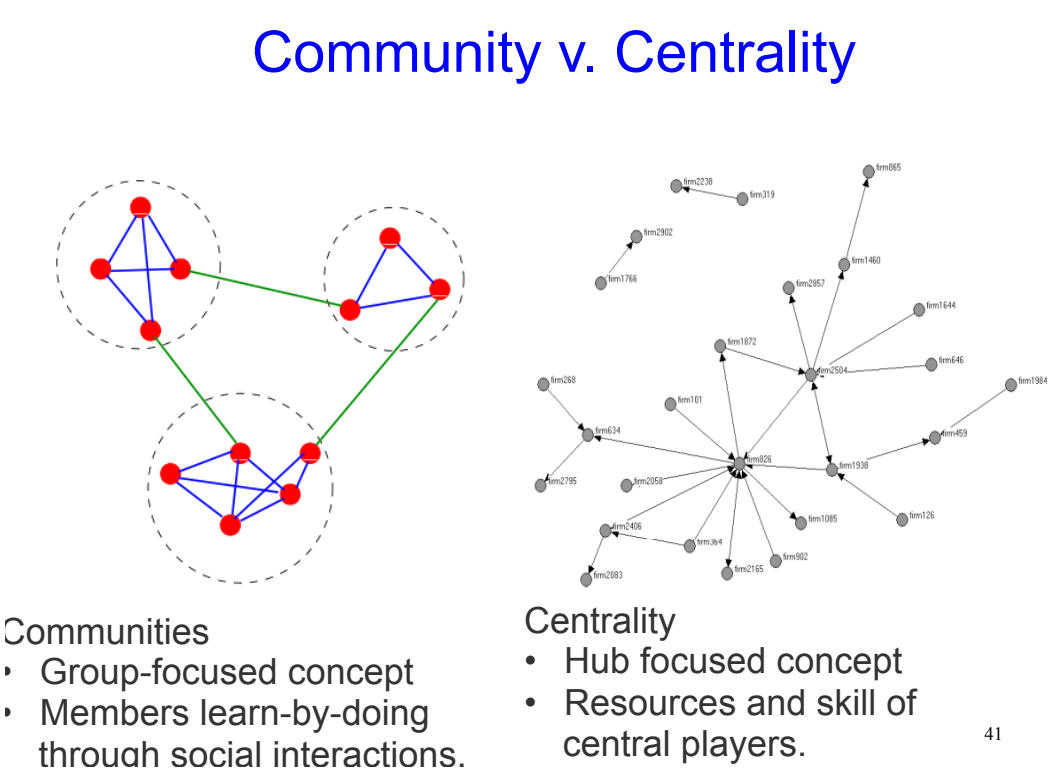
Community membership, on the other hand, is a *group* attribute that reflects whether a node belongs to a spatial cluster of nodes that tend to communicate a lot together. Community membership is a variable inherited by all members of a spatial agglomerate. However, centrality is an individual-centered variable that captures a node's influence. Community membership does not measure the reach or influence of a node. Rather, it is a measure focused on interactions between nodes, reflecting whether a node deals with familiar partners. Neither community membership nor centrality is a proper subset of the other.

The differences between community and centrality are visually depicted in Figure 13.13, which is reproduced from Burdick et al (2011). The figure shows the high centrality of Citigroup, J. P. Morgan, and Bank of America, well connected banks in co-lending networks. However, none of these banks belong to communities, which are represented by banks in the left and right nodes of the figure. In sum, community is a group attribute that measures whether a node belongs to a tight knit group. Centrality reflects the size and heft of a node's connections.[3] For another schematic that shows the same idea, i.e., the difference between

[3] Newman (2010) brings out the distinctions further. See his Sections 7.1/7.2 on centrality and Section 11.6 on community detection.

centrality and communities is in Figure 13.14.

Figure 13.14: Community versus centrality



See my paper titled "Venture Capital Communities" where I examine how VCs form communities, and whether community-funded startups do better than the others (we do find so). We also find evidence of some aspects of homophily within VC communities, though there are also aspects of heterogeneity in characteristics.

13.12 *Word of Mouth*

WOM has become an increasingly important avenue for viral marketing. Here is a article on the growth of this medium. See **?**. See also the really interesting paper by Godes and Mayzlin (2009) titled "Firm-Created Word-of-Mouth Communication: Evidence from a Field Test". This is an excellent example of how firms should go about creating buzz. See also Godes and Mayzlin (2004): "Using Online Conversations to Study Word of Mouth Communication" which looks at TV ratings and WOM.

13.13 Network Models of Systemic Risk

In an earlier section, we saw pictures of banking networks (see Figure 13.13), i.e., the interbank loan network. In these graphs, the linkages between banks were considered, but two things were missing. First, we assumed that all banks were similar in quality or financial health, and nodes were therefore identical. Second, we did not develop a network measure of overall system risk, though we did compute fragility and diameter for the banking network. What we also computed was the relative position of each bank in the network, i.e., it's eigenvalue centrality.

In the section, we augment network information of the graph with additional information on the credit quality of each node in the network. We then use this to compute a system-wide score of the overall risk of the system, denoting this as *systemic risk*. This section is based on [4].

[4]

We make the following assumptions and define notation:

- Assume n nodes, i.e., firms, or "assets."
- Let $E \in R^{n \times n}$ be a well-defined adjacency matrix. This quantifies the influence of each node on another.
- E may be portrayed as a directed graph, i.e., $E_{ij} \neq E_{ji}$.
  $E_{jj} = 1$; $E_{ij} \in \{0, 1\}$.
- C is a $(n \times 1)$ risk vector that defines the risk score for each asset.
- We define the "systemic risk score" as

$$S = \sqrt{C^\top \; E \; C}$$

- $S(C, E)$ is linear homogenous in C.

We note that this score captures two important features of systemic risk: (a) The interconnectedness of the banks in the system, through adjacency (or edge) matrix E, and (b) the financial quality of each bank in the system, denoted by the vector C, a proxy for credit score, i.e., credit rating, z-score, probability of default, etc.

13.13.1 Systemic Score, Fragility, Centrality, Diameter

We code up the systemic risk function as follows.

```
library(igraph)
```

```
#FUNCTION FOR RISK INCREMENT AND DECOMP
NetRisk = function(Ri,X) {
  S = sqrt(t(Ri) %*% X %*% Ri)
  RiskIncr = 0.5 * (X %*% Ri + t(X) %*% Ri)/S[1,1]
  RiskDecomp = RiskIncr * Ri
  result = list(S,RiskIncr,RiskDecomp)
}
```

To illustrate application, we generate a network of 15 banks by creating a random graph.

```
#CREATE ADJ MATRIX
e = floor(runif(15*15)*2)
X = matrix(e,15,15)
diag(X) = 1
```

This creates the network adjacency matrix and network plot shown in Figure 13.15. Note that the diagonal elements are 1, as this is needed for the risk score.

The code for the plot is as follows:

```
#GRAPH NETWORK: plot of the assets and the links with directed arrow
na = length(diag(X))
Y = X; diag(Y)=0
g = graph.adjacency(Y)
plot.igraph(g,layout=layout.fruchterman.reingold,
            edge.arrow.size=0.5,vertex.size=15,
            vertex.label=seq(1,na))
```

We now randomly create credit scores for these banks. Let's assume we have four levels of credit, $\{0,1,2,3\}$, where lower scores represent higher credit quality.

```
#CREATE CREDIT SCORES
Ri = matrix(floor(runif(na)*4),na,1)

> Ri
      [,1]
 [1,]    1
 [2,]    3
 [3,]    0
 [4,]    3
 [5,]    0
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
 [1,]    1    0    1    1    1    1    1    0    0     1     0     0     0     0     1
 [2,]    1    1    1    0    0    1    0    1    1     0     1     1     0     1     1
 [3,]    0    1    1    1    1    1    0    0    1     0     0     0     1     0     0
 [4,]    0    0    1    1    0    0    1    0    0     1     0     1     1     1     0
 [5,]    1    1    0    1    1    0    0    1    0     1     1     1     0     0     1
 [6,]    1    1    1    1    1    1    0    0    0     1     1     1     0     0     0
 [7,]    1    1    1    0    0    1    1    0    1     0     0     0     1     1     0
 [8,]    0    1    1    1    0    1    1    1    1     0     0     1     0     0     1
 [9,]    0    1    0    1    0    0    0    0    1     1     0     0     0     1     0
[10,]    0    0    0    0    0    0    1    1    1     1     1     1     1     1     1
[11,]    1    1    1    1    1    0    0    1    1     1     1     1     1     1     1
[12,]    0    1    0    0    1    1    1    1    0     0     1     1     1     1     1
[13,]    1    1    0    0    0    1    1    0    0     1     0     1     1     0     0
[14,]    1    1    0    1    1    1    1    1    1     0     0     0     1     1     0
[15,]    1    1    1    0    1    1    1    1    0     0     1     0     1     0     1
```

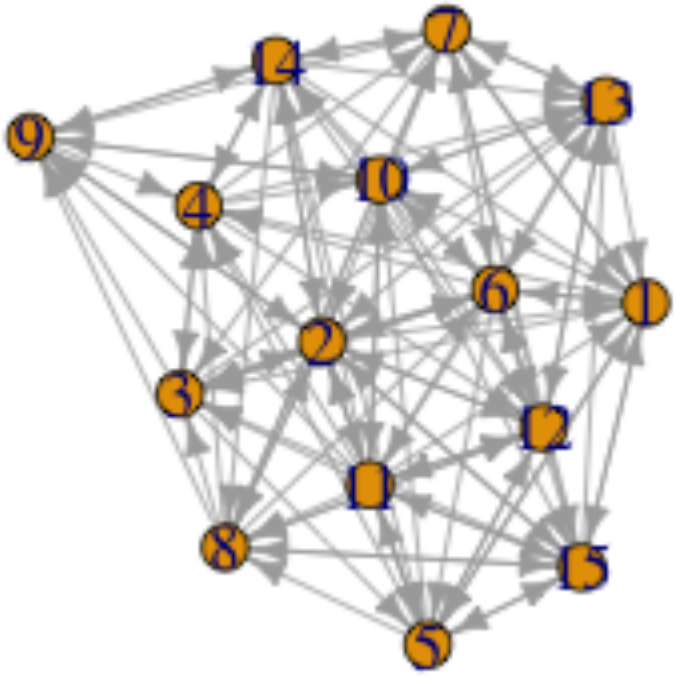


Figure 13.15: Banking network adjacency matrix and plot

```
 [6,]    0
 [7,]    2
 [8,]    0
 [9,]    0
[10,]    2
[11,]    0
[12,]    2
[13,]    2
[14,]    1
[15,]    3
```

We may now use this generated data to compute the overall risk score and risk increments, discussed later.

```
#COMPUTE OVERALL RISK SCORE AND RISK INCREMENT
res = NetRisk(Ri,X)
S = res[[1]]; print(c("Risk Score",S))
RiskIncr = res[[2]]

[1] "Risk Score"        "14.6287388383278"
```

We compute the **fragility** of this network.

```
#NETWORK FRAGILITY
deg = rowSums(X)-1
frag = mean(deg^2)/mean(deg)
print(c("Fragility score =",frag))

[1] "Fragility score =" "8.1551724137931"
```

The **centrality** of the network is computed and plotted with the following code. See Figure 13.16.

```
#NODE EIGEN VALUE CENTRALITY
cent = evcent(g)$vector
print("Normalized Centrality Scores")
print(cent)
sorted_cent = sort(cent,decreasing=TRUE,index.return=TRUE)
Scent = sorted_cent$x
idxScent = sorted_cent$ix
barplot(t(Scent),col="dark red",xlab="Node Number",
        names.arg=idxScent,cex.names=0.75)
```

```
> print(cent)
 [1] 0.7648332 1.0000000 0.7134844 0.6848305 0.7871945 0.8721071
 [7] 0.7389360 0.7788079 0.5647471 0.7336387 0.9142595 0.8857590
[13] 0.7183145 0.7907269 0.8365532
```

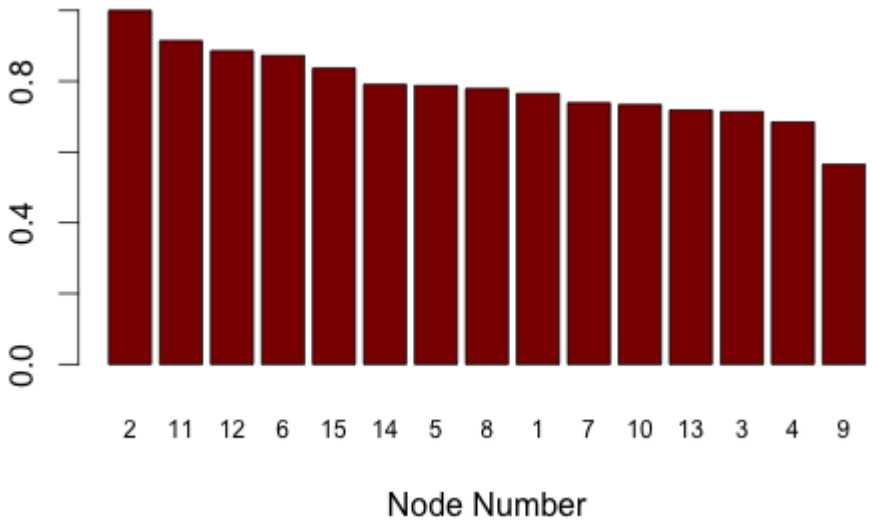


Figure 13.16: Centrality for the 15 banks.

And finally, we compute **diameter**.

```
print(diameter(g))
[1] 2
```

13.13.2 *Risk Decomposition*

Because the function $S(C, E)$ is homogenous of degree 1 in C, we may use this property to decompose the overall systemic score into the contribution from each bank. Applying Euler's theorem, we write this decomposition as:

$$S = \frac{\partial S}{\partial C_1} C_1 + \frac{\partial S}{\partial C_2} C_2 + \ldots + \frac{\partial S}{\partial C_n} C_n \tag{13.5}$$

The risk contribution of bank j is $\frac{\partial S}{\partial C_j} C_j$.

The code and output are shown here.

```
#COMPUTE RISK DECOMPOSITION
RiskDecomp = RiskIncr * Ri
sorted_RiskDecomp = sort(RiskDecomp,decreasing=TRUE,
                         index.return=TRUE)
RD = sorted_RiskDecomp$x
```

```
idxRD = sorted_RiskDecomp$ix
print("Risk Contribution");
print(RiskDecomp);
print(sum(RiskDecomp))
barplot(t(RD),col="dark green",xlab="Node Number",
        names.arg=idxRD,cex.names=0.75)
```

The output is as follows:

```
> print(RiskDecomp);
           [,1]
 [1,] 0.7861238
 [2,] 2.3583714
 [3,] 0.0000000
 [4,] 1.7431441
 [5,] 0.0000000
 [6,] 0.0000000
 [7,] 1.7089648
 [8,] 0.0000000
 [9,] 0.0000000
[10,] 1.3671719
[11,] 0.0000000
[12,] 1.7089648
[13,] 1.8456820
[14,] 0.8544824
[15,] 2.2558336
> print(sum(RiskDecomp))
[1] 14.62874
```

We see that the total of the individual bank risk contributions does indeed add up to the aggregate systemic risk score of 14.63, computed earlier.

The resulting sorted risk contributions of each node (bank) are shown in Figure 13.17.

13.13.3 Normalized Risk Score

We may also normalize the risk score to isolate the network effect by computing

$$\bar{S} = \frac{\sqrt{C^\top E\, C}}{\|C\|} \tag{13.6}$$

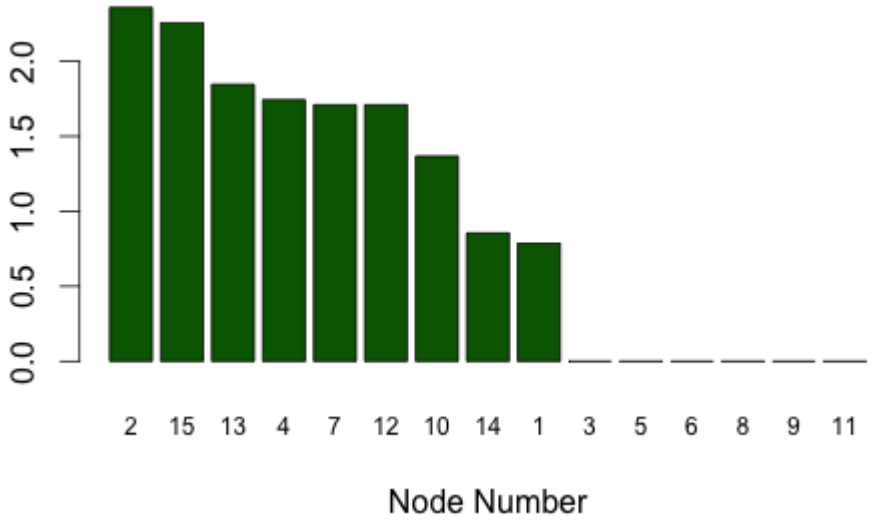


Figure 13.17: Risk Decompositions for the 15 banks.

where $\|C\| = \sqrt{C^\top C}$ is the norm of vector C. When there are no network effects, $E = I$, the identity matrix, and $\bar{S} = 1$, i.e., the normalized baseline risk level with no network (system-wide) effects is unity. As $\bar{S}$ increases above 1, it implies greater network effects.

```
#Compute normalized score SBar
Sbar = S/sqrt(t(Ri) %*% Ri)
print("Sbar (normalized risk score");

> print(Sbar)
          [,1]
[1,] 2.180724
```

13.13.4 Risk Increments

We are also interested in the extent to which a bank may impact the overall risk of the system if it begins to experience deterioration in credit quality. Therefore, we may compute the sensitivity of S to C:

$$\text{Risk increment} = I_j = \frac{\partial S}{\partial C_j}, \quad \forall j \tag{13.7}$$

```
> RiskIncr
           [,1]
 [1,] 0.7861238
```

```
 [2,] 0.7861238
 [3,] 0.6835859
 [4,] 0.5810480
 [5,] 0.7177652
 [6,] 0.8544824
 [7,] 0.8544824
 [8,] 0.8203031
 [9,] 0.5810480
[10,] 0.6835859
[11,] 0.9228410
[12,] 0.8544824
[13,] 0.9228410
[14,] 0.8544824
[15,] 0.7519445
```

Note that risk increments were previously computed in the function for the risk score. We also plot this in sorted order, as shown in Figure 13.18. The code for this plot is shown here.

```
#PLOT RISK INCREMENTS
sorted_RiskIncr = sort(RiskIncr,decreasing=TRUE,
                       index.return=TRUE)
RI = sorted_RiskIncr$x
idxRI = sorted_RiskIncr$ix
print("Risk␣Increment␣(per␣unit␣increase␣in␣any␣node␣risk")
print(RiskIncr)
barplot(t(RI),col="dark␣blue",xlab="Node␣Number",
        names.arg=idxRI,cex.names=0.75)
```

13.13.5 *Criticality*

Criticality is compromise-weighted centrality. This new measure is defined as $y = C \times x$ where x is the centrality vector for the network, and $y, C, x \in \mathcal{R}^n$. Note that this is an element-wise multiplication of vectors C and x. Critical nodes need immediate attention, either because they are heavily compromised or they are of high centrality, or both. It offers a way for regulators to prioritize their attention to critical financial institutions, and pre-empt systemic risk from blowing up.

```
#CRITICALITY
crit = Ri * cent
```

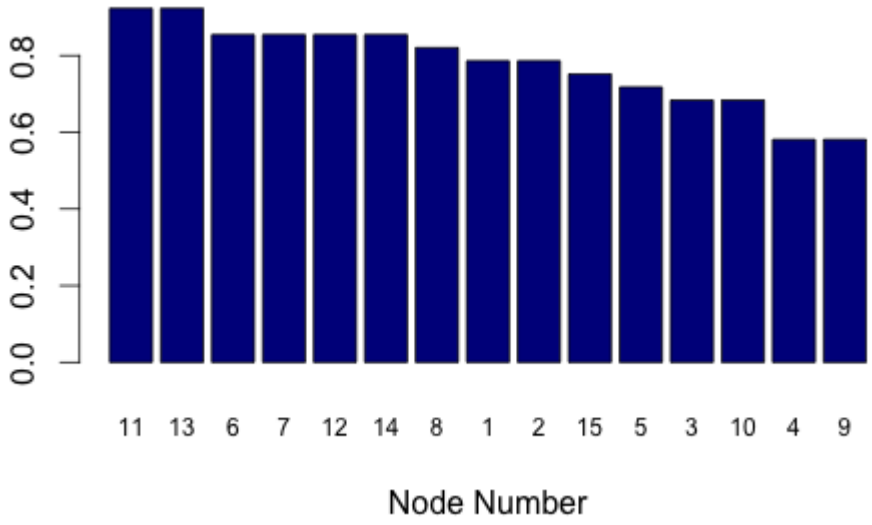


Figure 13.18: Risk Increments for the 15 banks.

```
print("Criticality␣Vector")
print(crit)
sorted_crit = sort(crit,decreasing=TRUE,index.return=TRUE)
Scrit = sorted_crit$x
idxScrit = sorted_crit$ix
barplot(t(Scrit),col="orange",xlab="Node␣Number",
        names.arg=idxScrit,cex.names=0.75)

> print(crit)
           [,1]
 [1,] 0.7648332
 [2,] 3.0000000
 [3,] 0.0000000
 [4,] 2.0544914
 [5,] 0.0000000
 [6,] 0.0000000
 [7,] 1.4778721
 [8,] 0.0000000
 [9,] 0.0000000
[10,] 1.4672773
[11,] 0.0000000
[12,] 1.7715180
[13,] 1.4366291
[14,] 0.7907269
```

```
[15,] 2.5096595
```

The plot of criticality is shown in Figure 13.19.

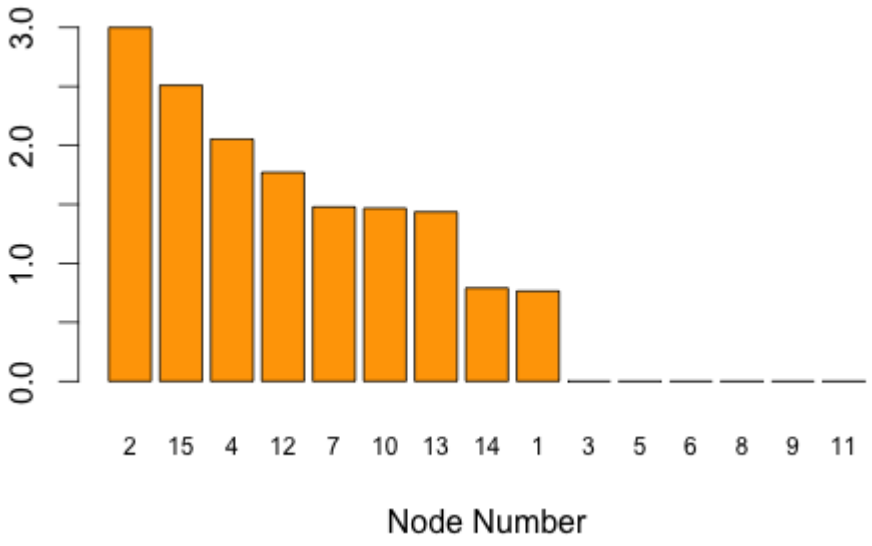


Figure 13.19: Criticality for the 15 banks.

13.13.6 Cross Risk

Since the systemic risk score S is a composite of network effects and credit quality, the risk contributions of all banks are impacted when any single bank suffers credit deterioration. A bank has the power to impose externalities on other banks, and we may assess how each bank's risk contribution is impacted by one bank's C increasing. We do this by simulating changes in a bank's credit quality and assessing the increase in risk contribution for the bank itself and other banks.

```
#CROSS IMPACT MATRIX
#CHECK FOR SPILLOVER EFFECTS FROM ONE NODE TO ALL OTHERS
d_RiskDecomp = NULL
n = length(Ri)
for (j in 1:n) {
  Ri2 = Ri
  Ri2[j] = Ri[j]+1
  res = NetRisk(Ri2,X)
  d_Risk = as.matrix(res[[3]]) - RiskDecomp
  d_RiskDecomp = cbind(d_RiskDecomp,d_Risk)
}
```

```
#3D plots
library("RColorBrewer");
library("lattice");
library("latticeExtra")
cloud(d_RiskDecomp,
      panel.3d.cloud = panel.3dbars,
      xbase = 0.25, ybase = 0.25,
      zlim = c(min(d_RiskDecomp), max(d_RiskDecomp)),
      scales = list(arrows = FALSE, just = "right"),
      xlab = "On", ylab = "From", zlab = NULL,
      main="Change in Risk Contribution",
      col.facet = level.colors(d_RiskDecomp,
          at = do.breaks(range(d_RiskDecomp), 20),
          col.regions = cm.colors, colors = TRUE),
      colorkey = list(col = cm.colors,
          at = do.breaks(range(d_RiskDecomp), 20)),
      #screen = list(z = 40, x = -30)
)
brewer.div <- colorRampPalette(brewer.pal(11, "Spectral"),
                               interpolate = "spline")
levelplot(d_RiskDecomp, aspect = "iso",
          col.regions = brewer.div(20),
          ylab="Impact from", xlab="Impact on",
          main="Change in Risk Contribution")
```

The plots are shown in Figure 13.20. We have used some advanced plotting functions, so as to demonstrate the facile way in which R generates beautiful plots.

Here we see the effect of a single bank's C value increasing by 1, and plot the change in risk contribution of each bank as a consequence. We notice that the effect on its own risk contribution is much higher than on that of other banks.

13.13.7 Risk Scaling

This is the increase in normalized risk score $\bar{S}$ as the number of connections per node increases. We compute this to examine how fast the score increases as the network becomes more connected. Is this growth exponential, linear, or logarithmic? We randomly generate graphs with increasing connectivity, and recompute the risk scores. The resulting

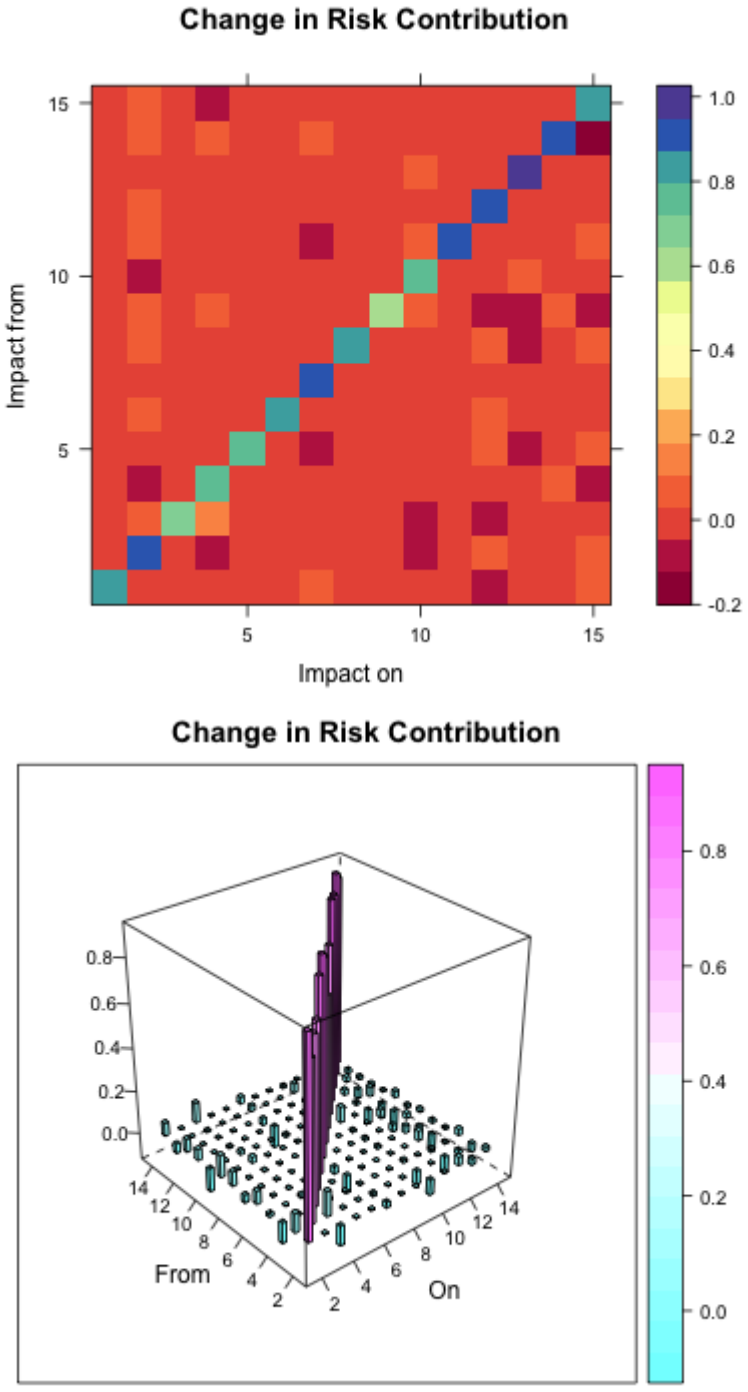


Figure 13.20: Spillover effects.

plots are shown in Figure 13.21. We see that the risk increases at a less than linear rate. This is good news, as systemic risk does not blow up as banks become more connected.

```
#RISK SCALING
#SIMULATION OF EFFECT OF INCREASED CONNECTIVITY
#RANDOM GRAPHS
n=50; k=100; pvec=seq(0.05,0.50,0.05);
svec=NULL; sbarvec=NULL
for (p in pvec) {
  s_temp = NULL
  sbar_temp = NULL
  for (j in 1:k) {
    g = erdos.renyi.game(n,p,directed=TRUE);
    A = get.adjacency(g)
    diag(A) = 1
    c = as.matrix(round(runif(n,0,2),0))
    syscore = as.numeric(sqrt(t(c) %*% A %*% c))
    sbarscore = syscore/n
    s_temp = c(s_temp,syscore)
    sbar_temp = c(sbar_temp,sbarscore)
  }
  svec = c(svec,mean(s_temp))
  sbarvec = c(sbarvec,mean(sbar_temp))
}
plot(pvec,svec,type="l",
     xlab="Prob␣of␣connecting␣to␣a␣node",
     ylab="S",lwd=3,col="red")
plot(pvec,sbarvec,type="l",
     xlab="Prob␣of␣connecting␣to␣a␣node",
     ylab="S_Avg",lwd=3,col="red")
```

13.13.8 Too Big To Fail?

An often suggested remedy for systemic risk is to break up large banks, i.e., directly mitigate the too-big-to-fail phenomenon. We calculate the change in risk score S, and normalized risk score $\bar{S}$ as the number of nodes increases, while keeping the average number of connections between nodes constant. This is repeated 5000 times for each fixed number

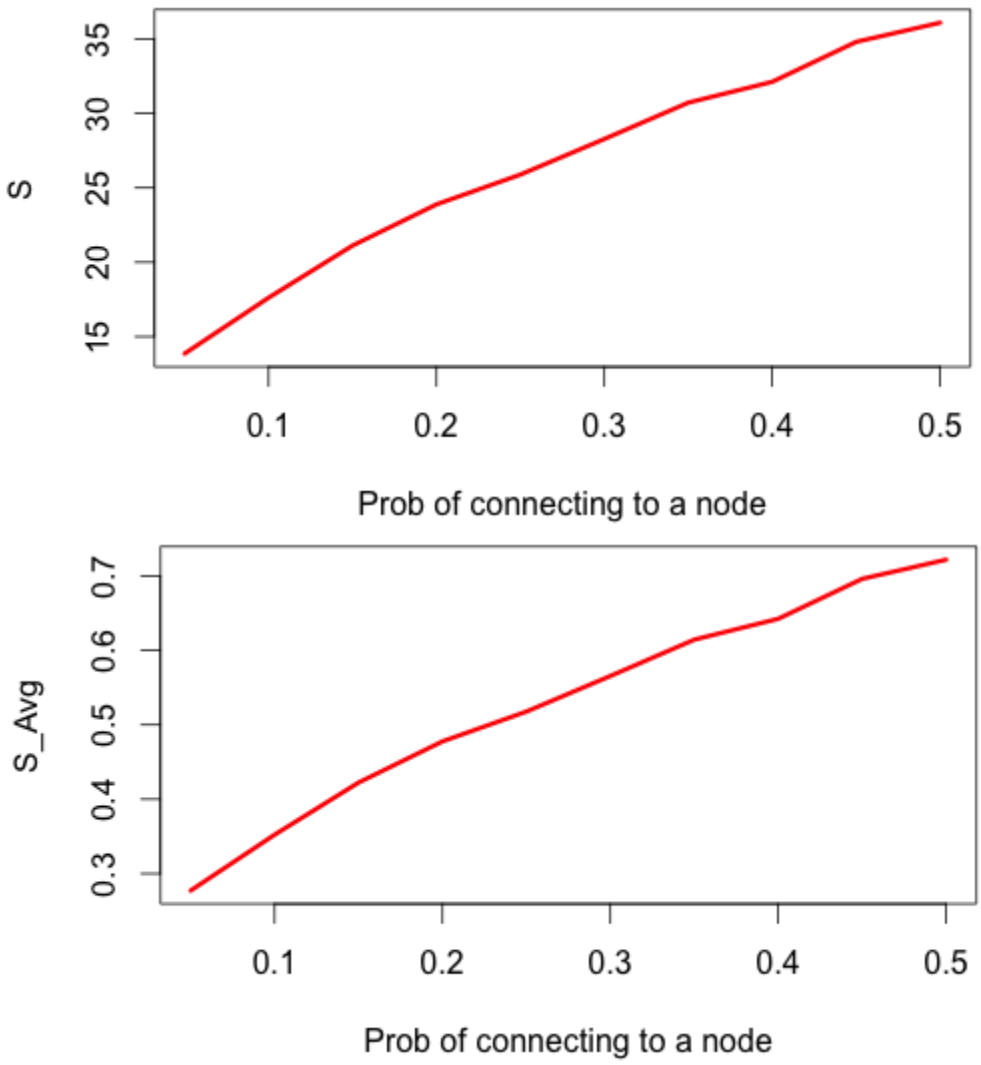


Figure 13.21: How risk increases with connectivity of the network.

of nodes and the mean risk score across 5000 simulations is plotted on the y-axis against the number of nodes on the x-axis. We see that systemic risk increases when banks are broken up, but the normalized risk score decreases. Despite the network effect $\bar{S}$ declining, overall risk S in fact increases. See Figure 13.22.

```
#TOO BIG TO FAIL
#SIMULATION OF EFFECT OF INCREASED NODES AND REDUCED CONNECTIVITY
nvec=seq(10,100,10); k=5000; svec=NULL; sbarvec=NULL
for (n in nvec) {
  s_temp = NULL
  sbar_temp = NULL
  p = 5/n
  for (j in 1:k) {
    g = erdos.renyi.game(n,p,directed=TRUE);
    A = get.adjacency(g)
    diag(A) = 1
    c = as.matrix(round(runif(n,0,2),0))
    syscore = as.numeric(sqrt(t(c) %*% A %*% c))
    sbarscore = syscore/n
    s_temp = c(s_temp,syscore)
    sbar_temp = c(sbar_temp,sbarscore)
  }
  svec = c(svec,mean(s_temp))
  sbarvec = c(sbarvec,mean(sbar_temp))
}
plot(nvec,svec,type="l",
     xlab="Number␣of␣nodes",ylab="S",
     ylim=c(0,max(svec)),lwd=3,col="red")
plot(nvec,sbarvec,type="l",
     xlab="Number␣of␣nodes",ylab="S_Avg",
     ylim=c(0,max(sbarvec)),lwd=3,col="red")
```

13.13.9 *Application of the model to the banking network in India*

The program code for systemic risk networks was applied to real-world data in India to produce daily maps of the Indian banking network, as well as the corresponding risk scores. The credit risk vector C was based on credit ratings for Indian financial institutions (FIs). The net-

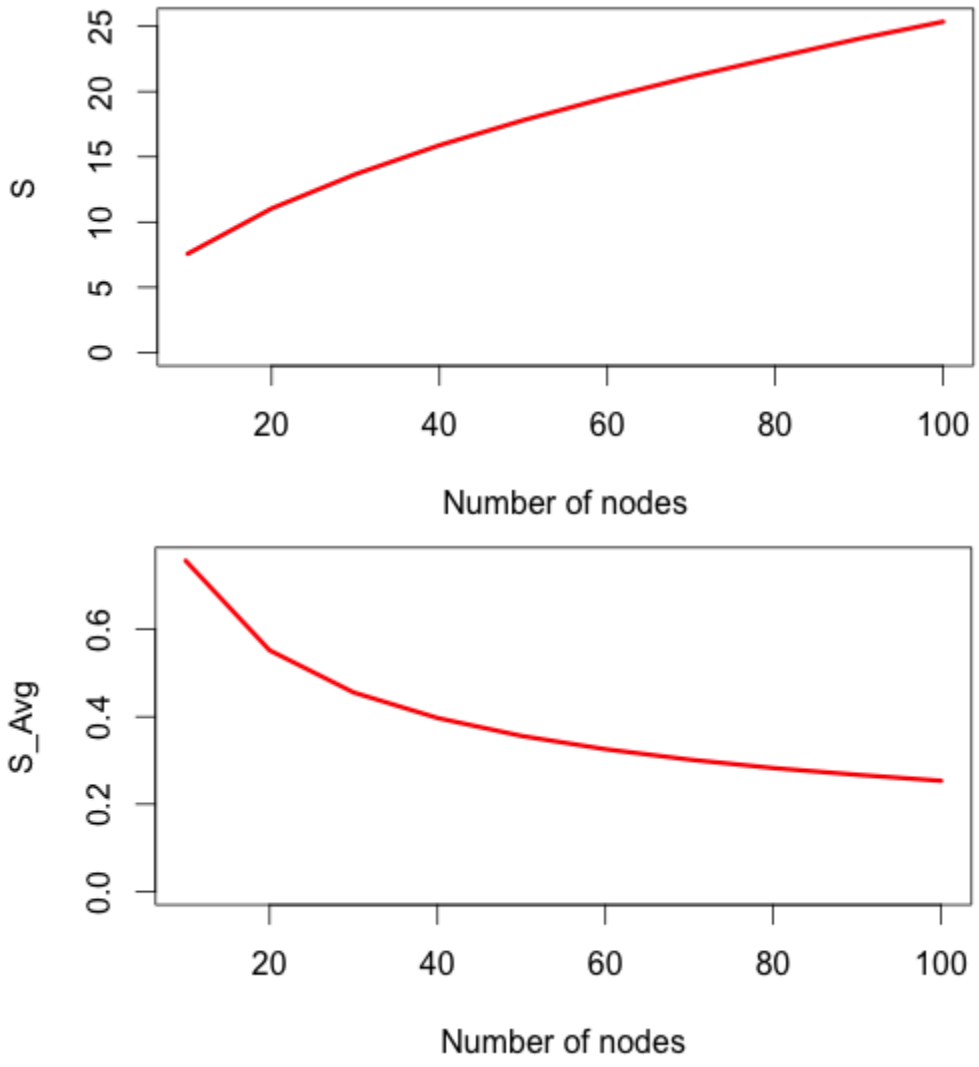


Figure 13.22: How risk increases with connectivity of the network.

work adjacency matrix was constructred using the ideas in a paper by Billio, Getmansky, Lo, and Pelizzon (2012) who create a network using Granger causality. This directed network comprises an adjacency matrix of values $(0, 1)$ where node i connects to node j if the returns of bank i Granger cause those of bank j, i.e., edge $E_{i,j} = 1$. This was applied to U.S. financial institution stock return data, and in a follow-up paper, to CDS spread data from U.S., Europe, and Japan (see Billio, Getmansky, Gray, Lo, Merton, and Pelizzon (2014)), where the global financial system is also found to be highly interconnected. In the application of the Das (2014) methodology to India, the network matrix is created using this Granger causality method to Indian FI stock returns.

The system is available in real time and may be accessed directly through a browser. To begin, different selections may be made of a subset of FIs for analysis. See Figure 13.23 for the screenshots of this step.

Once these selections are made and the "Submit" button is hit, the system generates the network and the various risk metrics, shown in Figures 13.24 and 13.25, respectively.

13.14 Map of Science

It is appropriate to end this chapter by showcasing network science with a wonderful image of the connection network between various scientific disciplines. See Figure 13.26. Note that the social sciences are most connected to medicine and engineering. But there is homophily here, i.e., likes tend to be in groups with likes.

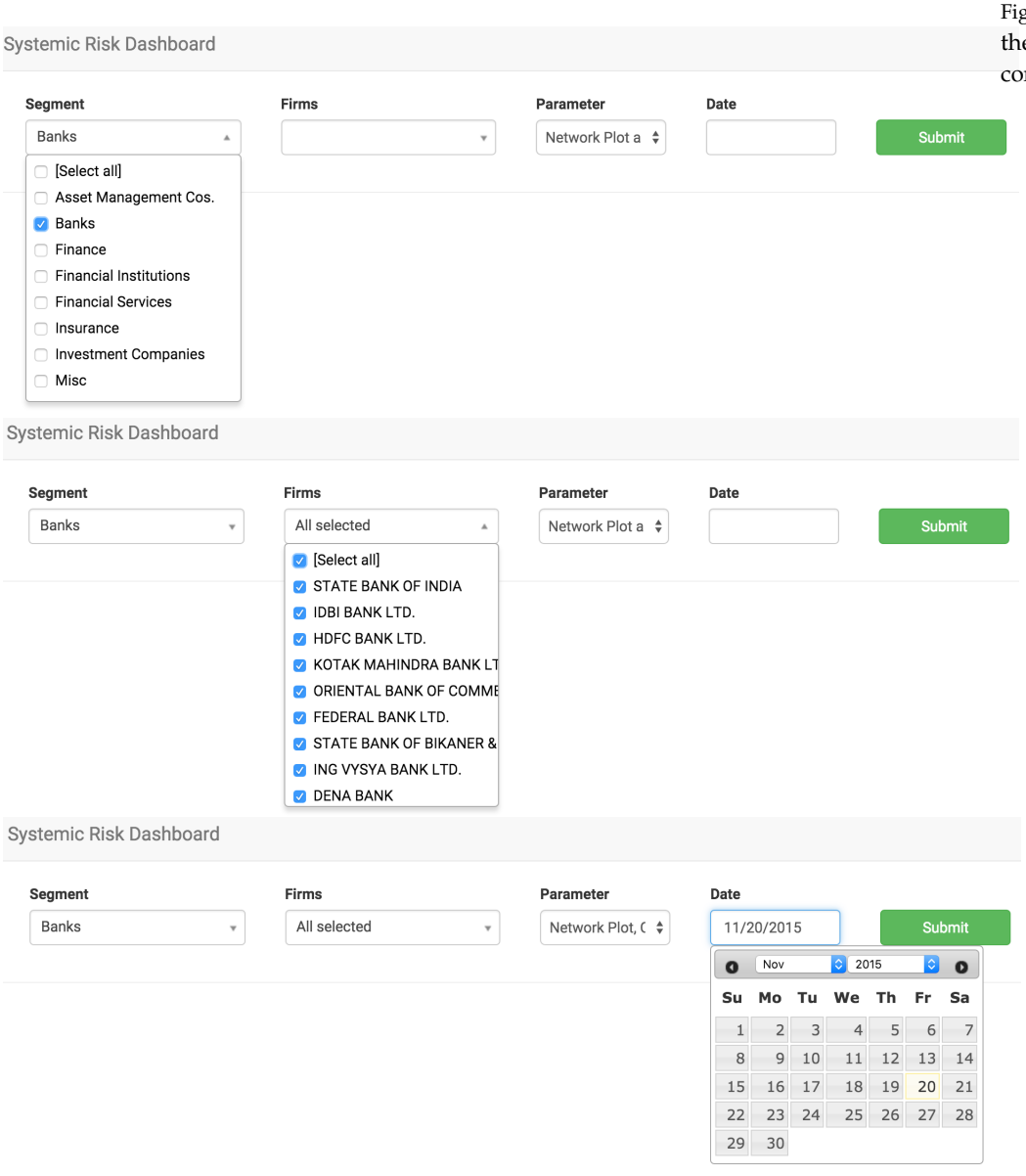


Figure 13.23: Screens for selecting the relevant set of Indian FIs to construct the banking network.

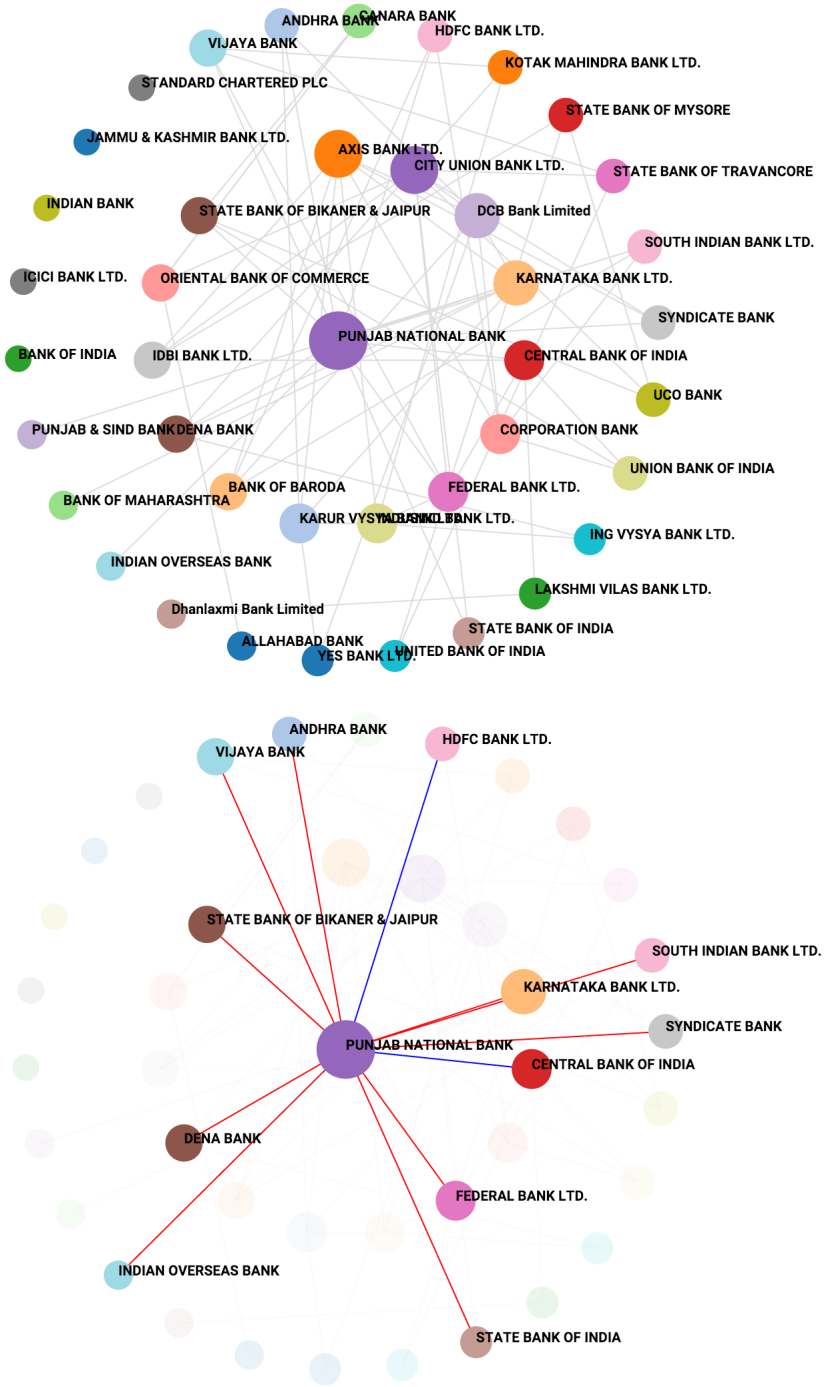
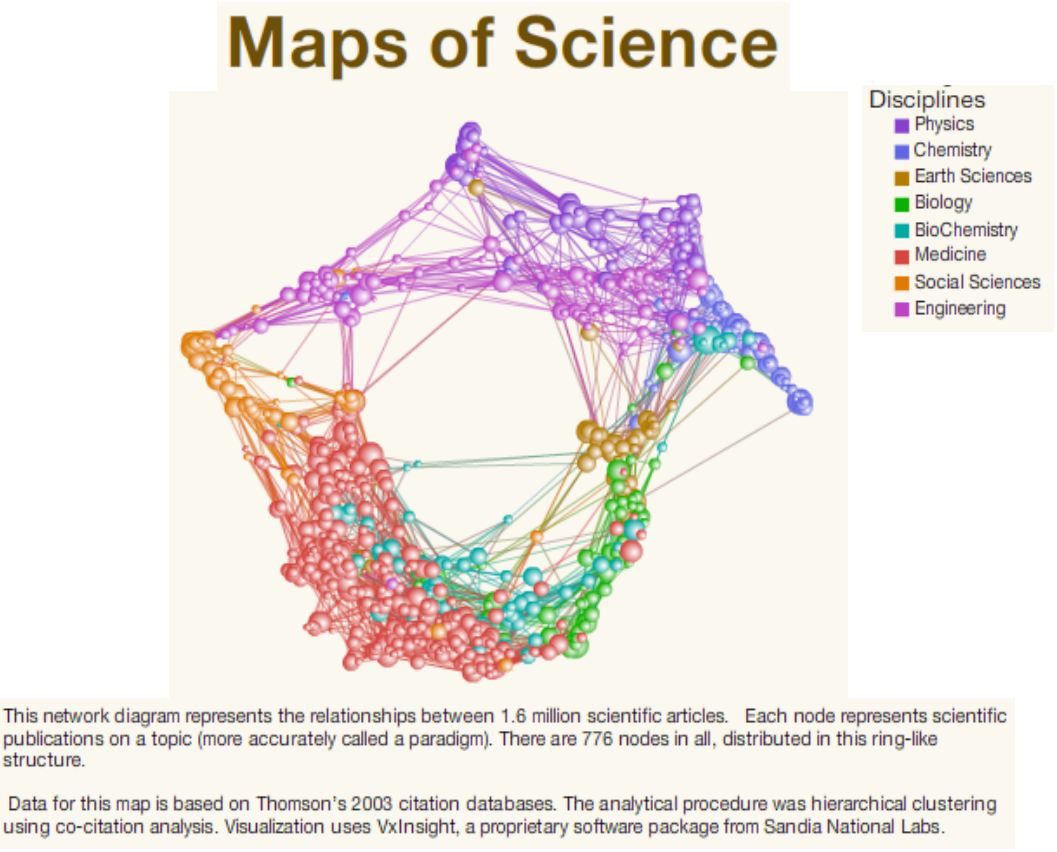


Figure 13.24: Screens for the Indian FIs banking network. The upper plot shows the entire network. The lower plot shows the network when we mouse over the bank in the middle of the plot. Red lines show that the bank is impacted by the other banks, and blue lines depict that the bank impacts the others, in a Granger causal manner.

Figure 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.

Figure 13.26: The Map of Science.



14
Statistical Brains: Neural Networks

14.1 Overview

Neural Networks (NNs) are one form of nonlinear regression. You are usually familiar with linear regressions, but nonlinear regressions are just as easy to understand. In a linear regression, we have

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

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

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

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

Differentiating w.r.t. β gives the following f.o.c:

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

We can examine this by using the `markowitzdata.txt` data set.

```
> data = read.table("markowitzdata.txt",header=TRUE)
> dim(data)
[1] 1507   10
> names(data)
 [1] "X.DATE" "SUNW"   "MSFT"   "IBM"    "CSCO"   "AMZN"   "mktrf"
 [8] "smb"    "hml"    "rf"
> amzn = as.matrix(data[,6])
> f3 = as.matrix(data[,7:9])
```

```
> res = lm(amzn ~ f3)
> summary(res)

Call:
lm(formula = amzn ~ f3)

Residuals:
      Min        1Q    Median        3Q       Max
-0.225716 -0.014029 -0.001142  0.013335  0.329627

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0015168  0.0009284   1.634  0.10249
f3mktrf      1.4190809  0.1014850  13.983  < 2e-16 ***
f3smb        0.5228436  0.1738084   3.008  0.00267 **
f3hml       -1.1502401  0.2081942  -5.525 3.88e-08 ***
---
Signif. codes:  0 ï£¡***ï£¡ 0.001 ï£¡**ï£¡ 0.01 ï£¡*ï£¡ 0.05 ï£¡.ï£¡ 0.1 ï£¡ ï£¡ 1

Residual standard error: 0.03581 on 1503 degrees of freedom
Multiple R-squared: 0.2233,     Adjusted R-squared: 0.2218
F-statistic: 144.1 on 3 and 1503 DF,  p-value: < 2.2e-16

> wuns = matrix(1,length(amzn),1)
> x = cbind(wuns,f3)
> b = solve(t(x) %*% x) %*% (t(x) %*% amzn)
> b
              [,1]
       0.001516848
mktrf  1.419080894
smb    0.522843591
hml   -1.150240145
```

We see at the end of the program listing that our formula for the coefficients of the minimized least squares problem $\beta = (X'X)^{-1}(X'Y)$ exactly matches that from the regression command `lm`.

14.2 *Nonlinear Regression*

A nonlinear regression is of the form

$$Y = f(X;\beta) + e$$

where $f(\cdot)$ is a nonlinear function. Note that, for example, $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + e$ is *not* a nonlinear regression, even though it contains nonlinear terms like X^2.

Computing the coefficients in a nonlinear regression again follows in the same way as for a linear regression.

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

Differentiating w.r.t. β gives the following f.o.c:

$$\begin{aligned} -2\left(\frac{df(X;\beta)}{d\beta}\right)' Y + 2\left(\frac{df(X;\beta)}{d\beta}\right)' f(X;\beta) &= \mathbf{0} \\ \left(\frac{df(X;\beta)}{d\beta}\right)' Y &= \left(\frac{df(X;\beta)}{d\beta}\right)' f(X;\beta) \end{aligned}$$

which is then solved numerically for $\beta \in R^n$. The approach taken usually involves the Newton-Raphson method, see for example:

`http://en.wikipedia.org/wiki/Newton's_method`.

14.3 *Perceptrons*

Neural networks are special forms of nonlinear regressions where the decision system for which the NN is built mimics the way the brain is supposed to work (whether it works like a NN is up for grabs of course).

The basic building block of a neural network is a perceptron. A perceptron is like a neuron in a human brain. It takes inputs (e.g. sensory in a real brain) and then produces an output signal. An entire network of perceptrons is called a neural net.

For example, if you make a credit card application, then the inputs comprise a whole set of personal data such as age, sex, income, credit score, employment status, etc, which are then passed to a series of perceptrons in parallel. This is the first "layer" of assessment. Each of the perceptrons then emits an output signal which may then be passed to another layer of perceptrons, who again produce another signal. This second layer is often known as the "hidden" perceptron layer. Finally, after many hidden layers, the signals are all passed to a single perceptron which emits the decision signal to issue you a credit card or to deny your application.

Perceptrons may emit continuous signals or binary $(0, 1)$ signals. In the case of the credit card application, the final perceptron is a binary one. Such perceptrons are implemented by means of "squashing" functions. For example, a really simple squashing function is one that issues a 1 if the function value is positive and a 0 if it is negative. More generally,

$$S(x) = \begin{cases} 1 & \text{if } g(x) > T \\ 0 & \text{if } g(x) \leq T \end{cases}$$

where $g(x)$ is any function taking positive and negative values, for instance, $g(x) \in (-\infty, +\infty)$. T is a threshold level.

A neural network with many layers is also known as a "multi-layered" perceptron, i.e., all those perceptrons together may be thought of as one single, big perceptron. See Figure 14.1 for an example of such a network

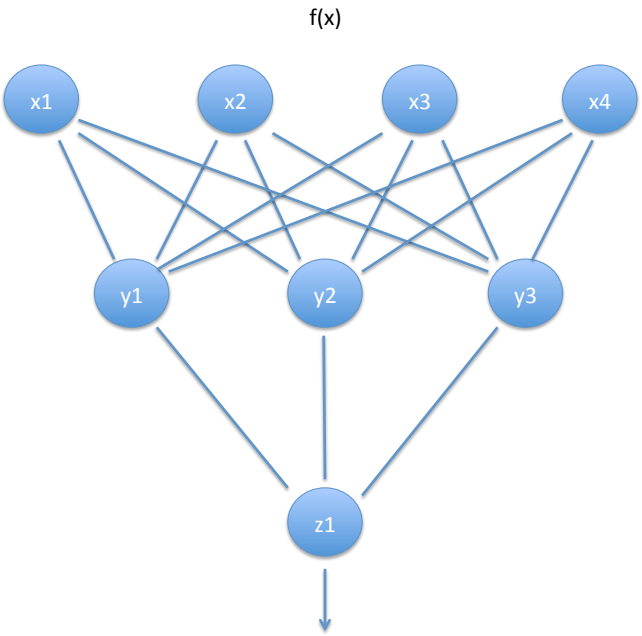


Figure 14.1: A feed-forward multi-layer neural network.

Neural net models are related to **Deep Learning**, where the number of hidden layers is vastly greater than was possible in the past when computational power was limited. Now, deep learning nets cascade through 20-30 layers, resulting in a surprising ability of neural nets in mimicking human learning processes. see: `http://en.wikipedia.org/wiki/Deep_learning`. And also see: `http://deeplearning.net/`.

Binary NNs are also thought of as a category of classifier systems. They are widely used to divide members of a population into classes. But NNs with continuous output are also popular. As we will see later, researchers have used NNs to learn the Black-Scholes option pricing model.

Areas of application: credit cards, risk management, forecasting corporate defaults, forecasting economic regimes, measuring the gains from mass mailings by mapping demographics to success rates.

14.4 Squashing Functions

Squashing functions may be more general than just binary. They usually squash the output signal into a narrow range, usually $(0, 1)$. A common choice is the logistic function (also known as the sigmoid function).

$$f(x) = \frac{1}{1 + e^{-w\,x}}$$

Think of w as the adjustable weight. Another common choice is the probit function

$$f(x) = \Phi(w\ x)$$

where $\Phi(\cdot)$ is the cumulative normal distribution function.

14.5 How does the NN work?

The easiest way to see how a NN works is to think of the simplest NN, i.e. one with a single perceptron generating a binary output. The perceptron has n inputs, with values $x_i, i = 1...n$ and current weights $w_i, i = 1...n$. It generates an output y.

The "net input" is defined as

$$\sum_{i=1}^{n} w_i x_i$$

If the net input is greater than a threshold T, then the output signal is $y = 1$, and if it is less than T, the output is $y = 0$. The actual output is called the "desired" output and is denoted $d = \{0, 1\}$. Hence, the "training" data provided to the NN comprises both the inputs x_i and the desired output d.

The output of our single perceptron model will be the sigmoid function of the net input, i.e.

$$y = \frac{1}{1 + \exp\left(-\sum_{i=1}^{n} w_i x_i\right)}$$

For a given input set, the error in the NN is

$$E = \frac{1}{2} \sum_{j=1}^{m} (y_j - d_j)^2$$

where m is the size of the training data set. The optimal NN for given data is obtained by finding the weights w_i that minimize this error function E. Once we have the optimal weights, we have a calibrated "feed-forward" neural net.

For a given squashing function f, and input $x = [x_1, x_2, \ldots, x_n]'$, the multi-layer perceptron will given an output at the hidden layer of

$$y(x) = f\left(w_0 + \sum_{j=1}^{n} w_j x_j\right)$$

and then at the final output level the node is

$$z(x) = f\left(w_0 + \sum_{i=1}^{N} w_i \cdot f\left(w_{0i} + \sum_{j=1}^{n} w_{ji} x_j\right)\right)$$

where the nested structure of the neural net is quite apparent.

14.5.1 Logit/Probit Model

The special model above with a single perceptron is actually nothing else than the logit regression model. If the squashing function is taken to the cumulative normal distribution, then the model becomes the probit regression model. In both cases though, the model is fitted by minimizing squared errors, not by maximum likelihood, which is how standard logit/probit models are parameterized.

14.5.2 Connection to hyperplanes

Note that in binary squashing functions, the net input is passed through a sigmoid function and then compared to the threshold level T. This sigmoid function is a monotone one. Hence, this means that there must be a level T' at which the net input $\sum_{i=1}^{n} w_i x_i$ must be for the result to be on the cusp. The following is the equation for a hyperplane

$$\sum_{i=1}^{n} w_i x_i = T'$$

which also implies that observations in n-dimensional space of the inputs x_i, must lie on one side or the other of this hyperplane. If above the hyperplane, then $y = 1$, else $y = 0$. Hence, single perceptrons in neural nets have a simple geometrical intuition.

14.6 Feedback/Backpropagation

What distinguishes neural nets from ordinary nonlinear regressions is feedback. Neural nets *learn* from feedback as they are used. Feedback is implemented using a technique called backpropagation.

Suppose you have a calibrated NN. Now you obtain another observation of data and run it through the NN. Comparing the output value y with the desired observation d gives you the error for this observation. If the error is large, then it makes sense to update the weights in the NN, so as to self-correct. This process of self-correction is known as "back-propagation".

The benefit of backpropagation is that a full re-fitting exercise is not required. Using simple rules the correction to the weights can be applied gradually in a learning manner.

Lets look at backpropagation with a simple example using a single perceptron. Consider the j-th perceptron. The sigmoid of this is

$$y_j = \frac{1}{1 + \exp\left(-\sum_{i=1}^{n} w_i x_{ij}\right)}$$

where y_j is the output of the j-th perceptron, and x_{ij} is the i-th input to the j-th perceptron. The error from this observation is $(y_j - d_j)$. Recalling that $E = \frac{1}{2}\sum_{j=1}^{m}(y_j - d_j)^2$, we may compute the change in error with respect to the j-th output, i.e.

$$\frac{\partial E}{\partial y_j} = y_j - d_j$$

Note also that

$$\frac{dy_j}{dx_{ij}} = y_j(1 - y_j)w_i$$

and

$$\frac{dy_j}{dw_i} = y_j(1 - y_j)x_{ij}$$

Next, we examine how the error changes with input values:

$$\frac{\partial E}{\partial x_{ij}} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dx_{ij}} = (y_j - d_j)y_j(1 - y_j)w_i$$

We can now get to the value of interest, which is the change in error value with respect to the weights

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dw_i} = (y_j - d_j)y_j(1 - y_j)x_{ij}, \forall i$$

We thus have one equation for each weight w_i and each observation j. (Note that the w_i apply across perceptrons. A more general case might be where we have weights for each perceptron, i.e., w_{ij}.) Instead of updating on just one observation, we might want to do this for many observations in which case the error derivative would be

$$\frac{\partial E}{\partial w_i} = \sum_j (y_j - d_j)y_j(1 - y_j)x_{ij}, \forall i$$

Therefore, if $\frac{\partial E}{\partial w_i} > 0$, then we would need to reduce w_i to bring down E. By how much? Here is where some art and judgment is imposed. There is a tuning parameter $0 < \gamma < 1$ which we apply to w_i to shrink it when the weight needs to be reduced. Likewise, if the derivative $\frac{\partial E}{\partial w_i} < 0$, then we would increase w_i by dividing it by γ.

14.6.1 Extension to many perceptrons

Our notation now becomes extended to weights w_{ik} which stand for the weight on the i-th input to the k-th perceptron. The derivative for the error becomes

$$\frac{\partial E}{\partial w_{ik}} = \sum_j (y_j - d_j) y_j (1 - y_j) x_{ikj}, \forall i, k$$

Hence all nodes in the network have their weights updated. In many cases of course, we can just take the derivatives numerically. Change the weight w_{ik} and see what happens to the error.

14.7 Research Applications

14.7.1 Discovering Black-Scholes

See the paper by Hutchinson, Lo, and Poggio (1994)), A Nonparametric Approach to Pricing and Hedging Securities Via Learning Networks, *The Journal of Finance*, Vol XLIX.

14.7.2 Forecasting

See the paper by Ghiassi, Saidane, and Zimbra (2005). "A dynamic artificial neural network model for forecasting time series events," *International Journal of Forecasting* 21, 341–362.

14.8 Package `neuralnet` *in R*

The package focuses on multi-layer perceptrons (MLP), see Bishop (1995), which are well applicable when modeling functional relationships. The underlying structure of an MLP is a directed graph, i.e. it consists of vertices and directed edges, in this context called neurons and synapses. [See Bishop (1995), Neural networks for pattern recognition. Oxford University Press, New York.]

The data set used by this package as an example is the `infert` data set that comes bundled with R.

```
> library(neuralnet)
Loading required package: grid
Loading required package: MASS
> names(infert)
[1] "education"       "age"             "parity"          "induced"
[5] "case"            "spontaneous"     "stratum"         "pooled.stratum"
> summary(infert)
   education        age             parity         induced
 0-5yrs : 12   Min.   :21.00   Min.   :1.000   Min.   :0.0000
 6-11yrs:120   1st Qu.:28.00   1st Qu.:1.000   1st Qu.:0.0000
 12+ yrs:116   Median :31.00   Median :2.000   Median :0.0000
               Mean   :31.50   Mean   :2.093   Mean   :0.5726
               3rd Qu.:35.25   3rd Qu.:3.000   3rd Qu.:1.0000
               Max.   :44.00   Max.   :6.000   Max.   :2.0000
      case          spontaneous        stratum      pooled.stratum
 Min.   :0.0000   Min.   :0.0000   Min.   : 1.00   Min.   : 1.00
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:21.00   1st Qu.:19.00
 Median :0.0000   Median :0.0000   Median :42.00   Median :36.00
 Mean   :0.3347   Mean   :0.5766   Mean   :41.87   Mean   :33.58
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:62.25   3rd Qu.:48.25
 Max.   :1.0000   Max.   :2.0000   Max.   :83.00   Max.   :63.00
```

This data set examines infertility after induced and spontaneous abortion. The variables `induced` and `spontaneous` take values in $\{0, 1, 2\}$ indicating the number of previous abortions. The variable `parity` denotes the number of births. The variable `case` equals 1 if the woman is infertile and 0 otherwise. The idea is to model infertility.

As a first step, let's fit a logit model to the data.

```
> res = glm(case ~ age+parity+induced+spontaneous,
            family=binomial(link="logit"), data=infert)
> summary(res)

Call:
glm(formula = case ~ age + parity + induced + spontaneous,
    family = binomial(link = "logit"),
    data = infert)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6281  -0.8055  -0.5298   0.8668   2.6141

Coefficients:
```

```
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.85239    1.00428  -2.840  0.00451 **
age          0.05318    0.03014   1.764  0.07767 .
parity      -0.70883    0.18091  -3.918 8.92e-05 ***
induced      1.18966    0.28987   4.104 4.06e-05 ***
spontaneous  1.92534    0.29863   6.447 1.14e-10 ***
---
Signif. codes:  0 ï£¡***ï£¡ 0.001 ï£¡**ï£¡ 0.01 ï£¡*ï£¡ 0.05 ï£¡.ï£¡ 0.1 ï£¡ ï£¡ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 316.17  on 247  degrees of freedom
Residual deviance: 260.94  on 243  degrees of freedom
AIC: 270.94

Number of Fisher Scoring iterations: 4
```

All explanatory variables are statistically significant. We now run this data through a neural net, as follows.

```
> nn = neuralnet(case~age+parity+induced+spontaneous,hidden=2,data=infert)
> nn
Call: neuralnet(formula = case ~ age + parity + induced + spontaneous,     data = infert, hidden = 2)

1 repetition was calculated.

        Error Reached Threshold Steps
1 19.36463007     0.008949536618 20111
> nn$result.matrix
                                      1
error                     19.364630070610
reached.threshold          0.008949536618
steps                  20111.000000000000
Intercept.to.1layhid1      9.422192588834
age.to.1layhid1           -1.293381222338
parity.to.1layhid1       -19.489105822032
induced.to.1layhid1       37.616977251411
spontaneous.to.1layhid1   32.647955233030
Intercept.to.1layhid2      5.142357912661
age.to.1layhid2           -0.077293384832
parity.to.1layhid2         2.875918354167
induced.to.1layhid2       -4.552792010965
spontaneous.to.1layhid2   -5.558639450018
Intercept.to.case          1.155876751703
1layhid.1.to.case         -0.545821730892
1layhid.2.to.case         -1.022853550121
```

Now we can go ahead and visualize the neural net. See Figure 14.2.

We see the weights on the initial input variables that go into two hidden perceptrons, and then these are fed into the output perceptron, that generates the result. We can look at the data and output as follows:

```
> head(cbind(nn$covariate,nn$net.result[[1]]))
  [,1] [,2] [,3] [,4]          [,5]
1   26    6    1    2 0.1420779618
2   42    1    1    0 0.5886305435
```

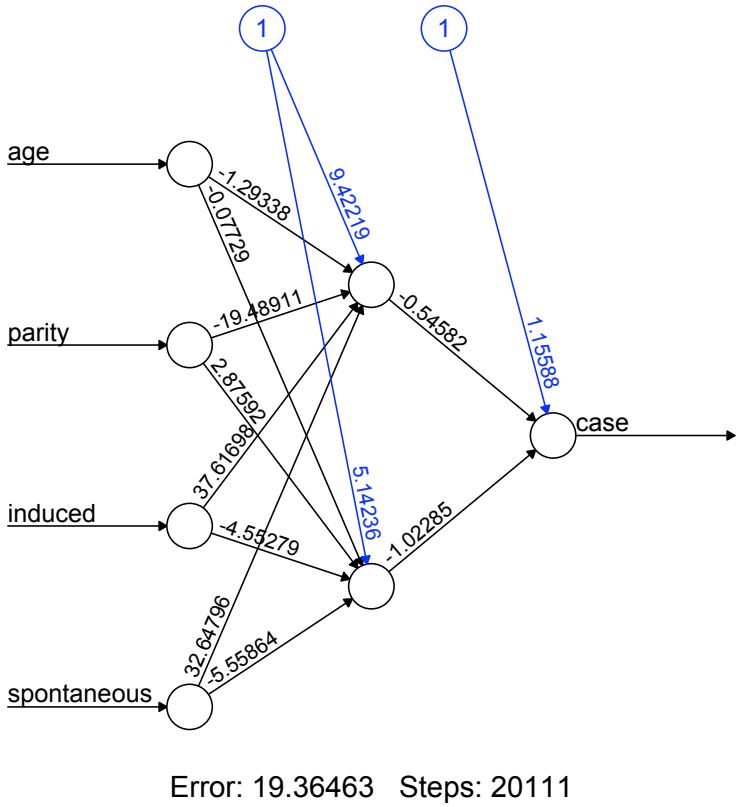


Figure 14.2: The neural net for the `infert` data set with two perceptrons in a single hidden layer.

```
3    39     6     2     0 0.1330583729
4    34     4     2     0 0.1404906398
5    35     3     1     1 0.4175799845
6    36     4     2     1 0.8385294748
```

We can compare the output to that from the logit model, by looking at the correlation of the fitted values from both models.

```
> cor(cbind(nn$net.result[[1]],res$fitted.values))
             [,1]         [,2]
[1,] 1.0000000000 0.8814759106
[2,] 0.8814759106 1.0000000000
```

As we see, the models match up with 88% correlation. The output is a probability of infertility.

We can add in an option for back propagation, and see how the results change.

```
> nn2 = neuralnet(case~age+parity+induced+spontaneous,
             hidden=2, algorithm="rprop+", data=infert)
> cor(cbind(nn2$net.result[[1]],res$fitted.values))
           [,1]       [,2]
[1,] 1.00000000 0.88816742
[2,] 0.88816742 1.00000000
> cor(cbind(nn2$net.result[[1]],nn$fitted.result[[1]]))
```

There does not appear to be any major improvement.

Given a calibrated neural net, how do we use it to compute values for a new observation? Here is an example.

```
> compute(nn,covariate=matrix(c(30,1,0,1),1,4))
$neurons
$neurons[[1]]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1   30    1    0    1

$neurons[[2]]
     [,1]                [,2]         [,3]
[1,]    1 0.00000009027594872 0.5351507372


$net.result
             [,1]
```

```
[1,] 0.6084958711
```

We can assess statistical significance of the model as follows:

```
> confidence.interval(nn,alpha=0.10)
$lower.ci
$lower.ci[[1]]
$lower.ci[[1]][[1]]
              [,1]           [,2]
[1,]   1.942871917   1.0100502322
[2,]  -2.178214123  -0.1677202246
[3,] -32.411347153  -0.6941528859
[4,]  12.311139796  -9.8846504753
[5,]  10.339781603 -12.1349900614

$lower.ci[[1]][[2]]
              [,1]
[1,]  0.7352919387
[2,] -0.7457112438
[3,] -1.4851089618

$upper.ci
$upper.ci[[1]]
$upper.ci[[1]][[1]]
              [,1]          [,2]
[1,] 16.9015132608 9.27466559308
[2,] -0.4085483215 0.01313345496
[3,] -6.5668644910 6.44598959422
[4,] 62.9228147066 0.77906645334
[5,] 54.9561288631 1.01771116133

$upper.ci[[1]][[2]]
              [,1]
[1,]  1.5764615647
[2,] -0.3459322180
[3,] -0.5605981384

$nic
[1] 21.19262393
```

The confidence level is $(1 - \alpha)$. This is at the 90% level, and at the 5% level we get:

```
> confidence.interval(nn,alpha=0.95)
$lower.ci
$lower.ci[[1]]
$lower.ci[[1]][[1]]
              [,1]           [,2]
[1,]   9.137058342   4.98482188887
[2,]  -1.327113719  -0.08074072852
[3,] -19.981740610   2.73981647809
[4,]  36.652242454  -4.75605852615
[5,]  31.797500416  -5.80934975682

$lower.ci[[1]][[2]]
              [,1]
[1,]  1.1398427910
[2,] -0.5534421216
[3,] -1.0404761197

$upper.ci
$upper.ci[[1]]
```

```
$upper.ci[[1]][[1]]
              [,1]           [,2]
[1,]   9.707326836  5.29989393645
[2,]  -1.259648725 -0.07384604115
[3,] -18.996471034  3.01202023024
[4,]  38.581712048 -4.34952549578
[5,]  33.498410050 -5.30792914321

$upper.ci[[1]][[2]]
              [,1]
[1,]  1.1719107124
[2,] -0.5382013402
[3,] -1.0052309806

$nic
[1] 21.19262393
```

14.9 *Package* nnet *in R*

We repeat these calculations using this alternate package.

```
> nn3 = nnet(case~age+parity+induced+spontaneous,data=infert,size=2)
# weights:  13
initial  value 58.675032
iter  10 value 47.924314
iter  20 value 41.032965
iter  30 value 40.169634
iter  40 value 39.548014
iter  50 value 39.025079
iter  60 value 38.657788
iter  70 value 38.464035
iter  80 value 38.273805
iter  90 value 38.189795
iter 100 value 38.116595
final  value 38.116595
stopped after 100 iterations
> nn3
a 4-2-1 network with 13 weights
inputs: age parity induced spontaneous
output(s): case
options were -
> nn3.out = predict(nn3)
> dim(nn3.out)
[1] 248   1
> cor(cbind(nn$fitted.result[[1]],nn3.out))
```

```
      [,1]
[1,]    1
```

We see that package `nnet` gives the same result as that from package `neuralnet`.

As another example of classification, rather than probability, we revisit the IRIS data set we have used in the realm of Bayesian classifiers.

```
> data(iris)
> # use half the iris data
> ir = rbind(iris3[,,1], iris3[,,2], iris3[,,3])
> targets = class.ind( c(rep("s", 50), rep("c", 50), rep("v", 50)) )
> samp = c(sample(1:50,25), sample(51:100,25), sample(101:150,25))

> ir1 = nnet(ir[samp,], targets[samp,],size = 2, rang = 0.1,
        decay = 5e-4, maxit = 200)
# weights:  19
initial  value 57.017869
iter  10 value 43.401134
iter  20 value 30.331122
iter  30 value 27.100909
iter  40 value 26.459441
iter  50 value 18.899712
iter  60 value 18.082379
iter  70 value 17.716302
iter  80 value 17.574713
iter  90 value 17.555689
iter 100 value 17.528989
iter 110 value 17.523788
iter 120 value 17.521761
iter 130 value 17.521578
iter 140 value 17.520840
iter 150 value 17.520649
iter 150 value 17.520649
final  value 17.520649
converged

> orig = max.col(targets[-samp,])
> orig
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
[36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[71] 3 3 3 3 3
> pred = max.col(predict(ir1,ir[-samp,]))
> pred
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 3 3 1 1 1 1 1 1 1
[36] 3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[71] 3 3 3 3 3
> table(orig,pred)
    pred
orig  1  2  3
   1 20  0  5
```

```
2  0 25  0
3  0  0 25
```

15
Zero or One: Optimal Digital Portfolios

Digital assets are investments with returns that are binary in nature, i.e., they either have a very large or very small payoff. We explore the features of optimal portfolios of digital assets such as venture investments, credit assets and lotteries. These portfolios comprise correlated assets with joint Bernoulli distributions. Using a simple, standard, fast recursion technique to generate the return distribution of the portfolio, we derive guidelines on how investors in digital assets may think about constructing their portfolios. We find that digital portfolios are better when they are homogeneous in the size of the assets, but heterogeneous in the success probabilities of the asset components.

The return distributions of digital portfolios are highly skewed and fat-tailed. A good example of such a portfolio is a venture fund. A simple representation of the payoff to a digital investment is Bernoulli with a large payoff for a successful outcome and a very small (almost zero) payoff for a failed one. The probability of success of digital investments is typically small, in the region of 5–25% for new ventures (see Das, Jagannathan and Sarin (2003)). Optimizing portfolios of such investments is therefore not amenable to standard techniques used for mean-variance optimization.

It is also not apparent that the intuitions obtained from the mean-variance setting carry over to portfolios of Bernoulli assets. For instance, it is interesting to ask, ceteris paribus, whether diversification by increasing the number of assets in the digital portfolio is always a good thing. Since Bernoulli portfolios involve higher moments, how diversification is achieved is by no means obvious. We may also ask whether it is preferable to include assets with as little correlation as possible or is there a sweet spot for the optimal correlation levels of the assets? Should all the investments be of even size, or is it preferable to take a

few large bets and several small ones? And finally, is a mixed portfolio of safe and risky assets preferred to one where the probability of success is more uniform across assets? These are all questions that are of interest to investors in digital type portfolios, such as CDO investors, venture capitalists and investors in venture funds.

We will use a method that is based on standard recursion for modeling of the exact return distribution of a Bernoulli portfolio. This method on which we build was first developed by Andersen, Sidenius and Basu (2003) for generating loss distributions of credit portfolios. We then examine the properties of these portfolios in a stochastic dominance framework framework to provide guidelines to digital investors. These guidelines are found to be consistent with prescriptions from expected utility optimization. The prescriptions are as follows:

1. Holding all else the same, more digital investments are preferred, meaning for example, that a venture portfolio should seek to maximize market share.

2. As with mean-variance portfolios, lower asset correlation is better, unless the digital investor's payoff depends on the upper tail of returns.

3. A strategy of a few large bets and many small ones is inferior to one with bets being roughly the same size.

4. And finally, a mixed portfolio of low-success and high-success assets is better than one with all assets of the same average success probability level.

Section 15.1 explains the methodology used. Section 15.4 presents the results. Conclusions and further discussion are in Section 15.5.

15.1 Modeling Digital Portfolios

Assume that the investor has a choice of n investments in digital assets (e.g., start-up firms). The investments are indexed $i = 1, 2, \ldots, n$. Each investment has a probability of success that is denoted q_i, and if successful, the payoff returned is S_i dollars. With probability $(1 - q_i)$, the investment will not work out, the start-up will fail, and the money will be lost in totality. Therefore, the payoff (cashflow) is

$$\text{Payoff} = C_i = \begin{cases} S_i & \text{with prob } q_i \\ 0 & \text{with prob } (1 - q_i) \end{cases} \tag{15.1}$$

The specification of the investment as a Bernoulli trial is a simple representation of reality in the case of digital portfolios. This mimics well for example, the case of the venture capital business. Two generalizations might be envisaged. First, we might extend the model to allowing S_i to be random, i.e., drawn from a range of values. This will complicate the mathematics, but not add much in terms of enriching the model's results. Second, the failure payoff might be non-zero, say an amount a_i. Then we have a pair of Bernoulli payoffs $\{S_i, a_i\}$. Note that we can decompose these investment payoffs into a project with constant payoff a_i plus another project with payoffs $\{S_i - a_i, 0\}$, the latter being exactly the original setting where the failure payoff is zero. Hence, the version of the model we solve in this note, with zero failure payoffs, is without loss of generality.

Unlike stock portfolios where the choice set of assets is assumed to be multivariate normal, digital asset investments have a joint Bernoulli distribution. Portfolio returns of these investments are unlikely to be Gaussian, and hence higher-order moments are likely to matter more. In order to generate the return distribution for the portfolio of digital assets, we need to account for the correlations across digital investments. We adopt the following simple model of correlation. Define y_i to be the performance proxy for the i-th asset. This proxy variable will be simulated for comparison with a threshold level of performance to determine whether the asset yielded a success or failure. It is defined by the following function, widely used in the correlated default modeling literature, see for example Andersen, Sidenius and Basu (2003):

$$y_i = \rho_i \, X + \sqrt{1 - \rho_i^2} \, Z_i, \quad i = 1 \ldots n \tag{15.2}$$

where $\rho_i \in [0, 1]$ is a coefficient that correlates threshold y_i with a normalized common factor $X \sim N(0, 1)$. The common factor drives the correlations amongst the digital assets in the portfolio. We assume that $Z_i \sim N(0, 1)$ and $\text{Corr}(X, Z_i) = 0, \forall i$. Hence, the correlation between assets i and j is given by $\rho_i \times \rho_j$. Note that the mean and variance of y_i are: $E(y_i) = 0, Var(y_i) = 1, \forall i$. Conditional on X, the values of y_i are all independent, as $\text{Corr}(Z_i, Z_j) = 0$.

We now formalize the probability model governing the success or failure of the digital investment. We define a variable x_i, with distribution function $F(\cdot)$, such that $F(x_i) = q_i$, the probability of success of the digital investment. Conditional on a fixed value of X, the probability of

success of the i-th investment is defined as

$$p_i^X \equiv Pr[y_i < x_i|X] \tag{15.3}$$

Assuming F to be the normal distribution function, we have

$$\begin{aligned} p_i^X &= Pr\left[\rho_i X + \sqrt{1-\rho_i^2}\, Z_i < x_i|X\right] \\ &= Pr\left[Z_i < \frac{x_i - \rho_i X}{\sqrt{1-\rho_i^2}}|X\right] \\ &= \Phi\left[\frac{F^{-1}(q_i) - \rho_i X}{\sqrt{1-\rho_i}}\right] \end{aligned} \tag{15.4}$$

where $\Phi(.)$ is the cumulative normal density function. Therefore, given the level of the common factor X, asset correlation ρ, and the unconditional success probabilities q_i, we obtain the conditional success probability for each asset p_i^X. As X varies, so does p_i^X. For the numerical examples here we choose the function $F(x_i)$ to the cumulative normal probability function.

We use a fast technique for building up distributions for sums of Bernoulli random variables. In finance, this *recursion* technique was introduced in the credit portfolio modeling literature by Andersen, Sidenius and Basu (2003).

We deem an investment in a digital asset as successful if it achieves its high payoff S_i. The cashflow from the portfolio is a random variable $C = \sum_{i=1}^n C_i$. The maximum cashflow that may be generated by the portfolio will be the sum of all digital asset cashflows, because each and every outcome was a success, i.e.,

$$C_{max} = \sum_{i=1}^{n} S_i \tag{15.5}$$

To keep matters simple, we assume that each S_i is an integer, and that we round off the amounts to the nearest significant digit. So, if the smallest unit we care about is a million dollars, then each S_i will be in units of integer millions.

Recall that, conditional on a value of X, the probability of success of digital asset i is given as p_i^X. The recursion technique will allow us to generate the portfolio cashflow probability distribution for each level of X. We will then simply compose these conditional (on X) distributions using the marginal distribution for X, denoted $g(X)$, into the unconditional distribution for the entire portfolio. Therefore, we define the

probability of total cashflow from the portfolio, conditional on X, to be $f(C|X)$. Then, the unconditional cashflow distribution of the portfolio becomes

$$f(C) = \int_X f(C|X) \cdot g(X)\, dX \tag{15.6}$$

The distribution $f(C|X)$ is easily computed numerically as follows.

We index the assets with $i = 1 \ldots n$. The cashflow from all assets taken together will range from zero to C_{max}. Suppose this range is broken into integer buckets, resulting in N_B buckets in total, each one containing an increasing level of total cashflow. We index these buckets by $j = 1 \ldots N_B$, with the cashflow in each bucket equal to B_j. B_j represents the total cashflow from all assets (some pay off and some do not), and the buckets comprise the discrete support for the entire distribution of total cashflow from the portfolio. For example, suppose we had 10 assets, each with a payoff of $C_i = 3$. Then $C_{max} = 30$. A plausible set of buckets comprising the support of the cashflow distribution would be: $\{0, 3, 6, 9, 12, 15, 18, 21, 24, 27, C_{max}\}$.

Define $P(k, B_j)$ as the probability of bucket j's cashflow level B_j if we account for the first k assets. For example, if we had just 3 assets, with payoffs of value 1,3,2 respectively, then we would have 7 buckets, i.e. $B_j = \{0, 1, 2, 3, 4, 5, 6\}$. After accounting for the first asset, the only possible buckets with positive probability would be $B_j = 0, 1$, and after the first two assets, the buckets with positive probability would be $B_j = 0, 1, 3, 4$. We begin with the first asset, then the second and so on, and compute the probability of seeing the returns in each bucket. Each probability is given by the following *recursion*:

$$P(k+1, B_j) = P(k, B_j)\ [1 - p^X_{k+1}] + P(k, B_j - S_{k+1})\ p^X_{k+1}, \quad k = 1, \ldots, n-1. \tag{15.7}$$

Thus the probability of a total cashflow of B_j after considering the first $(k+1)$ firms is equal to the sum of two probability terms. First, the probability of the same cashflow B_j from the first k firms, given that firm $(k+1)$ did not succeed. Second, the probability of a cashflow of $B_j - S_{k+1}$ from the first k firms and the $(k+1)$-st firm does succeed.

We start off this recursion from the first asset, after which the N_B buckets are all of probability zero, except for the bucket with zero cashflow (the first bucket) and the one with S_1 cashflow, i.e.,

$$P(1, 0) = 1 - p^X_1 \tag{15.8}$$

$$P(1, S_1) = p^X_1 \tag{15.9}$$

All the other buckets will have probability zero, i.e., $P(1, B_j \neq \{0, S_1\}) = 0$. With these starting values, we can run the system up from the first asset to the n-th one by repeated application of equation (15.7). Finally, we will have the entire distribution $P(n, B_j)$, conditional on a given value of X. We then compose all these distributions that are conditional on X into one single cashflow distribution using equation (15.6). This is done by numerically integrating over all values of X.

15.2 Implementation in R

15.2.1 Basic recursion

Given a set of outcomes and conditional (on state X) probabilities. we develop the recursion logic above in the following R function:

```
asbrec = function(w,p) {
#w: payoffs
#p: probabilities
#BASIC SET UP
N = length(w)
maxloss = sum(w)
bucket = c(0,seq(maxloss))
LP = matrix(0,N,maxloss+1)        #probability grid over losses

#DO FIRST FIRM
LP[1,1] = 1-p[1];
LP[1,w[1]+1] = p[1];

#LOOP OVER REMAINING FIRMS
for (i in seq(2,N)) {
        for (j in seq(maxloss+1)) {
                LP[i,j] = LP[i-1,j]*(1-p[i])
                if (bucket[j]-w[i] >= 0) {
                        LP[i,j] = LP[i,j] + LP[i-1,j-w[i]]*p[i]
                }
        }
}

#FINISH UP
lossprobs = LP[N,]
```

```
print(t(LP))
result = matrix(c(bucket,lossprobs),(maxloss+1),2)
}
```

We use this function in the following example.

```
w = c(5,8,4,2,1)
p = array(1/length(w),length(w))
res = asbrec(w,p)
print(res)
print(sum(res[,2]))
barplot(res[,2],names.arg=res[,1],
      xlab="portfolio value",ylab="probability")
```

The output of this run is as follows:

```
      [,1] [,2] [,3]  [,4]   [,5]    [,6]
 [1,]    0  0.8 0.64 0.512 0.4096 0.32768
 [2,]    1  0.0 0.00 0.000 0.0000 0.08192
 [3,]    2  0.0 0.00 0.000 0.1024 0.08192
 [4,]    3  0.0 0.00 0.000 0.0000 0.02048
 [5,]    4  0.0 0.00 0.128 0.1024 0.08192
 [6,]    5  0.2 0.16 0.128 0.1024 0.10240
 [7,]    6  0.0 0.00 0.000 0.0256 0.04096
 [8,]    7  0.0 0.00 0.000 0.0256 0.02560
 [9,]    8  0.0 0.16 0.128 0.1024 0.08704
[10,]    9  0.0 0.00 0.032 0.0256 0.04096
[11,]   10  0.0 0.00 0.000 0.0256 0.02560
[12,]   11  0.0 0.00 0.000 0.0064 0.01024
[13,]   12  0.0 0.00 0.032 0.0256 0.02176
[14,]   13  0.0 0.04 0.032 0.0256 0.02560
[15,]   14  0.0 0.00 0.000 0.0064 0.01024
[16,]   15  0.0 0.00 0.000 0.0064 0.00640
[17,]   16  0.0 0.00 0.000 0.0000 0.00128
[18,]   17  0.0 0.00 0.008 0.0064 0.00512
[19,]   18  0.0 0.00 0.000 0.0000 0.00128
[20,]   19  0.0 0.00 0.000 0.0016 0.00128
[21,]   20  0.0 0.00 0.000 0.0000 0.00032
```

Here each column represents one pass through the recursion. Since there are five assets, we get five passes, and the final column is the result we are looking for. The plot of the outcome distribution is shown in Figure

15.1.

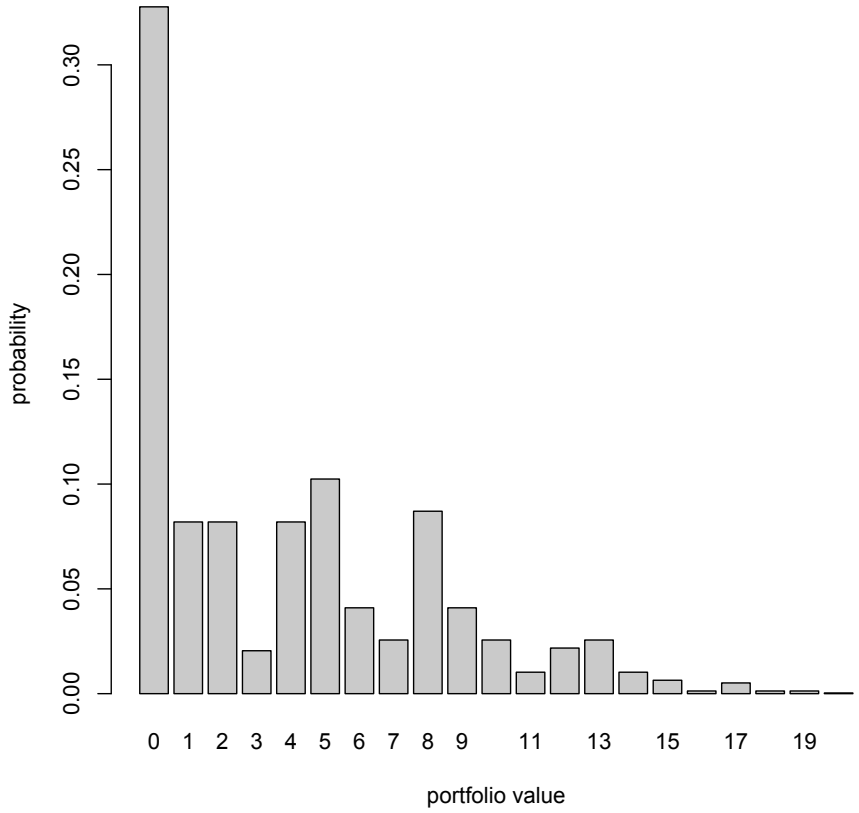


Figure 15.1: Plot of the final outcome distribution for a digital portfolio with five assets of outcomes $\{5, 8, 4, 2, 1\}$ all of equal probability.

We can explore these recursion calculations in some detail as follows. Note that in our example $p_i = 0.2, i = 1, 2, 3, 4, 5$. We are interested in computing $P(k, B)$, where k denotes the k-th recursion pass, and B denotes the return bucket. Recall that we have five assets with return levels of $\{5, 8, 4, 2, 1\}$, respecitvely. After $i = 1$, we have

$$\begin{aligned} P(1,0) &= (1 - p_1) = 0.8 \\ P(1,5) &= p_1 = 0.2 \\ P(1,j) &= 0, j \neq \{0,5\} \end{aligned}$$

The completes the first recursion pass and the values can be verified from the R output above by examining column 2 (column 1 contains the values of the return buckets). We now move on the calculations needed

for the second pass in the recursion.

$$\begin{aligned}
P(2,0) &= P(1,0)(1-p_2) = 0.64 \\
P(2,5) &= P(1,5)(1-p_2) + P(1,5-8)p_2 = 0.2(0.8) + 0(0.2) = 0.16 \\
P(2,8) &= P(1,8)(1-p_2) + P(1,8-8)p_2 = 0(0.8) + 0.8(0.2) = 0.16 \\
P(2,13) &= P(1,13)(1-p_2) + P(1,13-8)p_2 = 0(0.8) + 0.2(0.2) = 0.04 \\
P(2,j) &= 0, j \neq \{0,5,8,13\}
\end{aligned}$$

The third recursion pass is as follows:

$$\begin{aligned}
P(3,0) &= P(2,0)(1-p_3) = 0.512 \\
P(3,4) &= P(2,4)(1-p_3) + P(2,4-4) = 0(0.8) + 0.64(0.2) = 0.128 \\
P(3,5) &= P(2,5)(1-p_3) + P(2,5-4)p_3 = 0.16(0.8) + 0(0.2) = 0.128 \\
P(3,8) &= P(2,8)(1-p_3) + P(2,8-4)p_3 = 0.16(0.8) + 0(0.2) = 0.128 \\
P(3,9) &= P(2,9)(1-p_3) + P(2,9-4)p_3 = 0(0.8) + 0.16(0.2) = 0.032 \\
P(3,12) &= P(2,12)(1-p_3) + P(2,12-4)p_3 = 0(0.8) + 0.16(0.2) = 0.032 \\
P(3,13) &= P(2,13)(1-p_3) + P(2,13-4)p_3 = 0.04(0.8) + 0(0.2) = 0.032 \\
P(3,17) &= P(2,17)(1-p_3) + P(2,17-4)p_3 = 0(0.8) + 0.04(0.2) = 0.008 \\
P(3,j) &= 0, j \neq \{0,4,5,8,9,12,13,17\}
\end{aligned}$$

Note that the same computation work even when the outcomes are not of equal probability.

15.2.2 Combining conditional distributions

We now demonstrate how we will integrate the conditional probability distributions p^X into an unconditional probability distribution of outcomes, denoted $p = \int_X p^X g(X)\ dX$, where $g(X)$ is the density function of the state variable X. We create a function to combine the conditional distribution functions. This function calls the `absrec` function that we had used earlier.

```
#FUNCTION TO COMPUTE FULL RETURN DISTRIBUTION
#INTEGRATES OVER X BY CALLING ASBREC
digiprob = function(L,q,rho) {
        dx = 0.1
        x = seq(-40,40)*dx
        fx = dnorm(x)*dx
        fx = fx/sum(fx)
```

```
        maxloss = sum(L)
        bucket = c(0,seq(maxloss))
        totp = array(0,(maxloss+1))
        for (i in seq(length(x))) {
                p = pnorm((qnorm(q)-rho*x[i])/sqrt(1-rho^2))
                ldist = asbrec(L,p)
                totp = totp + ldist[,2]*fx[i]
        }
        result = matrix(c(bucket,totp),(maxloss+1),2)
}
```

Note that now we will use the unconditional probabilities of success for each asset, and correlate them with a specified correlation level. We run this with two correlation levels $\{-0.5, +0.5\}$.

```
#———————INTEGRATE OVER CONDITIONAL DISTRIBUTIONS———
w = c(5,8,4,2,1)
q = c(0.1,0.2,0.1,0.05,0.15)
rho = 0.25
res1 = digiprob(w,q,rho)
rho = 0.75
res2 = digiprob(w,q,rho)
par(mfrow=c(2,1))
barplot(res1[,2],names.arg=res1[,1],xlab="portfolio value",
    ylab="probability",main="rho = 0.25")
barplot(res2[,2],names.arg=res2[,1],xlab="portfolio value",
    ylab="probability",main="rho = 0.75")
```

The output plots of the unconditional outcome distribution are shown in Figure 15.2. We can see the data for the plots as follows.

```
> cbind(res1,res2)
      [,1]         [,2] [,3]         [,4]
 [1,]    0 0.5391766174    0 0.666318464
 [2,]    1 0.0863707325    1 0.046624312
 [3,]    2 0.0246746918    2 0.007074104
 [4,]    3 0.0049966420    3 0.002885901
 [5,]    4 0.0534700675    4 0.022765422
 [6,]    5 0.0640540228    5 0.030785967
 [7,]    6 0.0137226107    6 0.009556413
 [8,]    7 0.0039074039    7 0.002895774
```

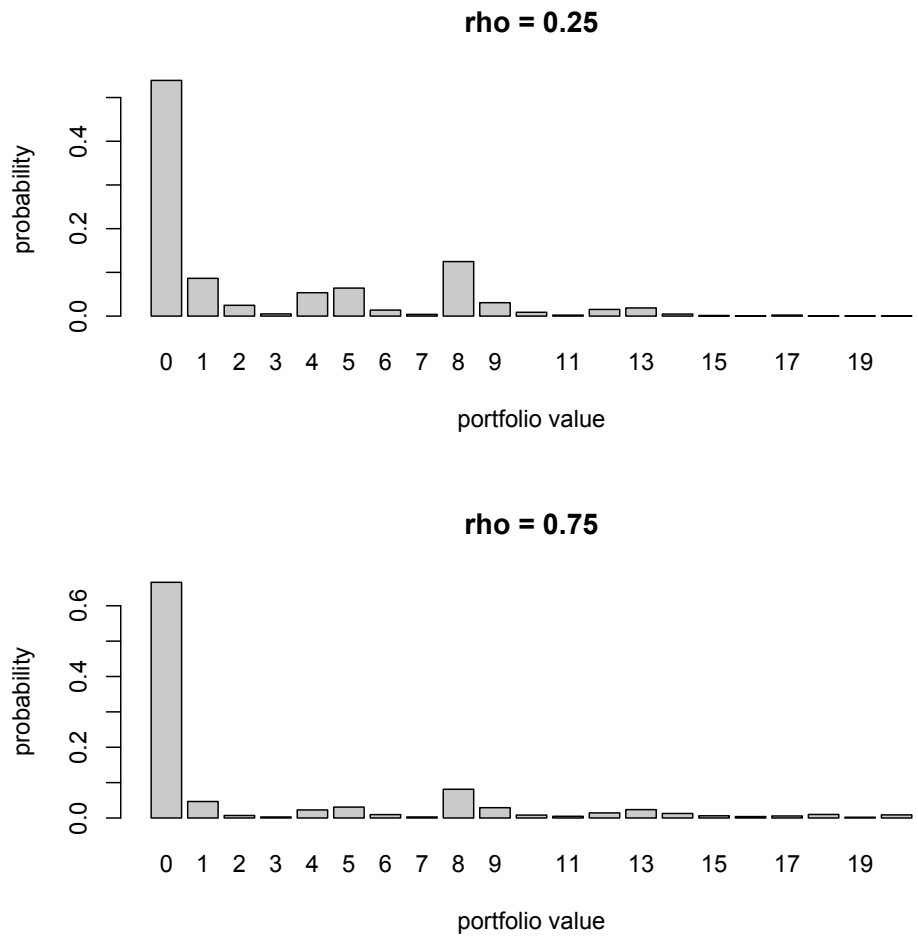


Figure 15.2: Plot of the final outcome distribution for a digital portfolio with five assets of outcomes $\{5, 8, 4, 2, 1\}$ with unconditional probability of success of $\{0.1, 0.2, 0.1, 0.05, 0.15\}$, respecitvely.

```
 [9,]    8 0.1247287209    8 0.081172499
[10,]    9 0.0306776806    9 0.029154885
[11,]   10 0.0086979993   10 0.008197488
[12,]   11 0.0021989842   11 0.004841742
[13,]   12 0.0152035638   12 0.014391319
[14,]   13 0.0186144920   13 0.023667222
[15,]   14 0.0046389439   14 0.012776165
[16,]   15 0.0013978502   15 0.006233366
[17,]   16 0.0003123473   16 0.004010559
[18,]   17 0.0022521668   17 0.005706283
[19,]   18 0.0006364672   18 0.010008267
[20,]   19 0.0002001003   19 0.002144265
[21,]   20 0.0000678949   20 0.008789582
```

The left column of probabilities has correlation of $\rho = 0.25$ and the right one is the case when $\rho = 0.75$. We see that the probabilities on the right are lower for low outcomes (except zero) and high for high outcomes. Why? See the plot of the difference between the high correlation case and low correlation case in Figure 15.3.

15.3 Stochastic Dominance (SD)

SD is an ordering over probabilistic bundles. We may want to know if one VC's portfolio dominates another in a risk-adjusted sense. Different SD concepts apply to answer this question. For example if portfolio A does better than portfolio B in every state of the world, it clearly dominates. This is called "state-by-state" dominance, and is hardly ever encountered. Hence, we briefly examine two more common types of SD.

1. First-order Stochastic Dominance (FSD): For cumulative distribution function $F(X)$ over states X, portfolio A dominates B if $\text{Prob}(A \geq k) \geq \text{Prob}(B \geq k)$ for all states $k \in X$, and $\text{Prob}(A \geq k) > \text{Prob}(B \geq k)$ for some k. It is the same as $\text{Prob}(A \leq k) \leq \text{Prob}(B \leq k)$ for all states $k \in X$, and $\text{Prob}(A \leq k) < \text{Prob}(B \leq k)$ for some k.This implies that $F_A(k) \leq F_B(k)$. The mean outcome under A will be higher than under B, and all increasing utility functions will give higher utility for A. This is a weaker notion of dominance than state-wise, but also not as often encountered in practice.

```
> x = seq(-4,4,0.1)
> F_B = pnorm(x,mean=0,sd=1);
> F_A = pnorm(x,mean=0.25,sd=1);
> F_A-F_B    #FSD exists
 [1] -2.098272e-05 -3.147258e-05 -4.673923e-05 -6.872414e-05 -1.000497e-04
 [6] -1.442118e-04 -2.058091e-04 -2.908086e-04 -4.068447e-04 -5.635454e-04
```

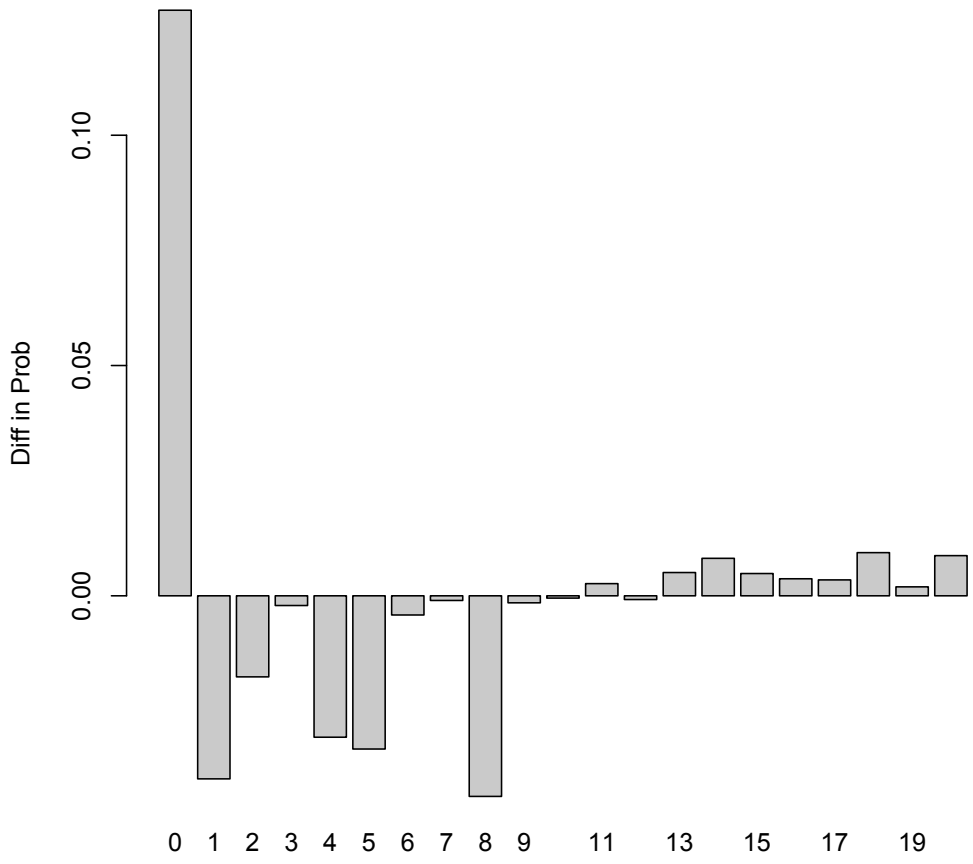


Figure 15.3: Plot of the difference in distribution for a digital portfolio with five assets when $\rho = 0.75$ minus that when $\rho = 0.25$. We use outcomes $\{5, 8, 4, 2, 1\}$ with unconditional probability of success of $\{0.1, 0.2, 0.1, 0.05, 0.15\}$, respecitvely.

```
[11] -7.728730e-04 -1.049461e-03 -1.410923e-03 -1.878104e-03 -2.475227e-03
[16] -3.229902e-03 -4.172947e-03 -5.337964e-03 -6.760637e-03 -8.477715e-03
[21] -1.052566e-02 -1.293895e-02 -1.574810e-02 -1.897740e-02 -2.264252e-02
[26] -2.674804e-02 -3.128519e-02 -3.622973e-02 -4.154041e-02 -4.715807e-02
[31] -5.300548e-02 -5.898819e-02 -6.499634e-02 -7.090753e-02 -7.659057e-02
[36] -8.191019e-02 -8.673215e-02 -9.092889e-02 -9.438507e-02 -9.700281e-02
[41] -9.870633e-02 -9.944553e-02 -9.919852e-02 -9.797262e-02 -9.580405e-02
[46] -9.275614e-02 -8.891623e-02 -8.439157e-02 -7.930429e-02 -7.378599e-02
[51] -6.797210e-02 -6.199648e-02 -5.598646e-02 -5.005857e-02 -4.431528e-02
[56] -3.884257e-02 -3.370870e-02 -2.896380e-02 -2.464044e-02 -2.075491e-02
[61] -1.730902e-02 -1.429235e-02 -1.168461e-02 -9.458105e-03 -7.580071e-03
[66] -6.014807e-03 -4.725518e-03 -3.675837e-03 -2.831016e-03 -2.158775e-03
[71] -1.629865e-03 -1.218358e-03 -9.017317e-04 -6.607827e-04 -4.794230e-04
[76] -3.443960e-04 -2.449492e-04 -1.724935e-04 -1.202675e-04 -8.302381e-05
[81] -5.674604e-05
```

2. Second-order Stochastic Dominance (SSD): Here the portfolios have the same mean but the risk is less for portfolio A. Then we say that portfolio A has a "mean-preserving spread" over portfolio B. Technically this is the same as $\int_{-\infty}^{k}[F_A(k) - F_B(k)]\, dX < 0$, and $\int_X X dF_A(X) = \int_X X dF_B(X)$. Mean-variance models in which portfolios on the efficient frontier dominate those below are a special case of SSD. See the example below, there is no FSD, but there is SSD.

```
> x = seq(-4,4,0.1)
> F_B = pnorm(x,mean=0,sd=2);
> F_A = pnorm(x,mean=0,sd=1);
> F_A-F_B     #No FSD
 [1] -0.02271846 -0.02553996 -0.02864421 -0.03204898 -0.03577121 -0.03982653
 [7] -0.04422853 -0.04898804 -0.05411215 -0.05960315 -0.06545730 -0.07166345
[13] -0.07820153 -0.08504102 -0.09213930 -0.09944011 -0.10687213 -0.11434783
[19] -0.12176261 -0.12899464 -0.13590512 -0.14233957 -0.14812981 -0.15309708
[25] -0.15705611 -0.15982015 -0.16120699 -0.16104563 -0.15918345 -0.15549363
[31] -0.14988228 -0.14229509 -0.13272286 -0.12120570 -0.10783546 -0.09275614
[37] -0.07616203 -0.05829373 -0.03943187 -0.01988903  0.00000000  0.01988903
[43]  0.03943187  0.05829373  0.07616203  0.09275614  0.10783546  0.12120570
[49]  0.13272286  0.14229509  0.14988228  0.15549363  0.15918345  0.16104563
[55]  0.16120699  0.15982015  0.15705611  0.15309708  0.14812981  0.14233957
[61]  0.13590512  0.12899464  0.12176261  0.11434783  0.10687213  0.09944011
[67]  0.09213930  0.08504102  0.07820153  0.07166345  0.06545730  0.05960315
[73]  0.05411215  0.04898804  0.04422853  0.03982653  0.03577121  0.03204898
[79]  0.02864421  0.02553996  0.02271846
> cumsum(F_A-F_B)     #But there is SSD
 [1] -2.271846e-02 -4.825842e-02 -7.690264e-02 -1.089516e-01 -1.447228e-01
 [6] -1.845493e-01 -2.287779e-01 -2.777659e-01 -3.318781e-01 -3.914812e-01
[11] -4.569385e-01 -5.286020e-01 -6.068035e-01 -6.918445e-01 -7.839838e-01
[16] -8.834239e-01 -9.902961e-01 -1.104644e+00 -1.226407e+00 -1.355401e+00
[21] -1.491306e+00 -1.633646e+00 -1.781776e+00 -1.934873e+00 -2.091929e+00
[26] -2.251749e+00 -2.412956e+00 -2.574002e+00 -2.733185e+00 -2.888679e+00
[31] -3.038561e+00 -3.180856e+00 -3.313579e+00 -3.434785e+00 -3.542620e+00
[36] -3.635376e+00 -3.711538e+00 -3.769832e+00 -3.809264e+00 -3.829153e+00
[41] -3.829153e+00 -3.809264e+00 -3.769832e+00 -3.711538e+00 -3.635376e+00
[46] -3.542620e+00 -3.434785e+00 -3.313579e+00 -3.180856e+00 -3.038561e+00
[51] -2.888679e+00 -2.733185e+00 -2.574002e+00 -2.412956e+00 -2.251749e+00
[56] -2.091929e+00 -1.934873e+00 -1.781776e+00 -1.633646e+00 -1.491306e+00
[61] -1.355401e+00 -1.226407e+00 -1.104644e+00 -9.902961e-01 -8.834239e-01
[66] -7.839838e-01 -6.918445e-01 -6.068035e-01 -5.286020e-01 -4.569385e-01
[71] -3.914812e-01 -3.318781e-01 -2.777659e-01 -2.287779e-01 -1.845493e-01
[76] -1.447228e-01 -1.089516e-01 -7.690264e-02 -4.825842e-02 -2.271846e-02
```

```
[81] -2.220446e-16
```

15.4 Portfolio Characteristics

Armed with this established machinery, there are several questions an investor (e.g. a VC) in a digital portfolio may pose. First, is there an optimal number of assets, i.e., ceteris paribus, are more assets better than fewer assets, assuming no span of control issues? Second, are Bernoulli portfolios different from mean-variances ones, in that is it always better to have less asset correlation than more correlation? Third, is it better to have an even weighting of investment across the assets or might it be better to take a few large bets amongst many smaller ones? Fourth, is a high dispersion of probability of success better than a low dispersion? These questions are very different from the ones facing investors in traditional mean-variance portfolios. We shall examine each of these questions in turn.

15.4.1 How many assets?

With mean-variance portfolios, keeping the mean return of the portfolio fixed, more securities in the portfolio is better, because diversification reduces the variance of the portfolio. Also, with mean-variance portfolios, higher-order moments do not matter. But with portfolios of Bernoulli assets, increasing the number of assets might exacerbate higher-order moments, even though it will reduce variance. Therefore it may not be worthwhile to increase the number of assets (n) beyond a point.

In order to assess this issue we conducted the following experiment. We invested in n assets each with payoff of $1/n$. Hence, if all assets succeed, the total (normalized) payoff is 1. This normalization is only to make the results comparable across different n, and is without loss of generality. We also assumed that the correlation parameter is $\rho_i = 0.25$, for all i. To make it easy to interpret the results, we assumed each asset to be identical with a success probability of $q_i = 0.05$ for all i. 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 Figure 15.4, 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.

One approach to determining if greater n is better for a digital portfolio is to investigate if a portfolio of n assets stochastically dominates one with less than n assets. On examination of the shapes of the distribution functions for different n, we see that it is likely that as n increases, we obtain portfolios that exhibit second-order stochastic dominance (SSD) over portfolios with smaller n. The return distribution when $n = 100$ (denoted G_{100}) would dominate that for $n = 25$ (denoted G_{25}) in the SSD sense, if $\int_x x\, dG_{100}(x) = \int_x x\, dG_{25}(x)$, and $\int_0^u [G_{100}(x) - G_{25}(x)]\, dx \leq 0$ for all $u \in (0,1)$. That is, G_{25} has a mean-preserving spread over G_{100}, or G_{100} has the same mean as G_{25} but lower variance, i.e., implies superior mean-variance efficiency. To show this we plotted the integral $\int_0^u [G_{100}(x) - G_{25}(x)]\, dx$ and checked the SSD condition. We found that this condition is satisfied (see Figure 15.4). As is known, SSD implies mean-variance efficiency as well.

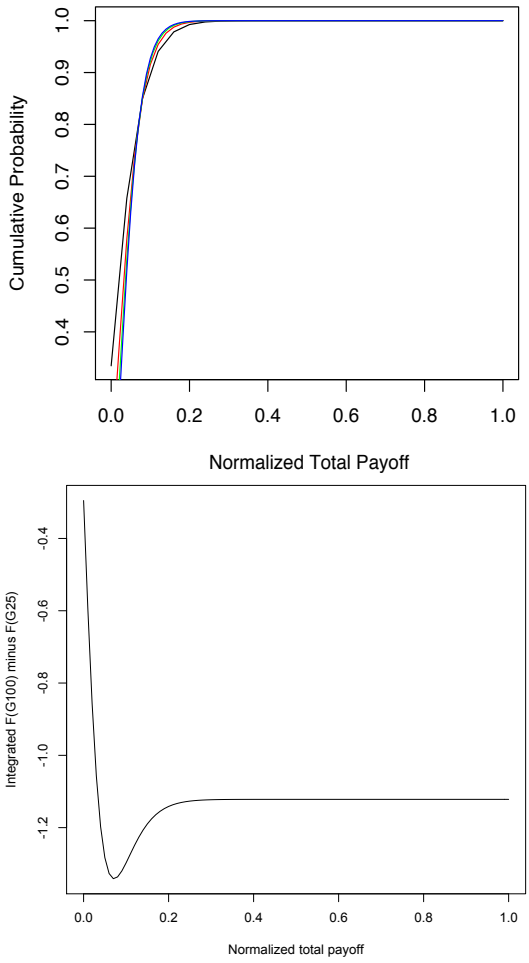


Figure 15.4: Distribution functions for returns from Bernoulli investments as the number of investments (n) increases. Using the recursion technique we computed the probability distribution of the portfolio payoff for four values of $n = \{25, 50, 75, 100\}$. The distribution function is plotted in the 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 $\rho = 0.25$.

We also examine if higher n portfolios are better for a power utility investor with utility function, $U(C) = \frac{(0.1+C)^{1-\gamma}}{1-\gamma}$, where C is the normalized total payoff of the Bernoulli portfolio. Expected utility is given by $\sum_C U(C)\ f(C)$. We set the risk aversion coefficient to $\gamma = 3$ which is in the standard range in the asset-pricing literature. Table 15.1 reports the results. We can see that the expected utility increases monotonically with n. Hence, for a power utility investor, having more assets is better than less, keeping the mean return of the portfolio constant. Economically, in the specific case of VCs, this highlights the goal of trying to capture a larger share of the number of available ventures. The results from the SSD analysis are consistent with those of expected power utility.

n	$E(C)$	$Pr[C > 0.03]$	$Pr[C > 0.07]$	$Pr[C > 0.10]$	$Pr[C > 0.15]$	$E[U(C)]$
25	0.05	0.665	0.342	0.150	0.059	−29.259
50	0.05	0.633	0.259	0.084	0.024	−26.755
75	0.05	0.620	0.223	0.096	0.015	−25.876
100	0.05	0.612	0.202	0.073	0.011	−25.433

Table 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 $U(C) = (0.1 + C)^{1-\gamma}/(1-\gamma), \gamma = 3$.

We have abstracted away from issues of the span of management by investors. Given that investors actively play a role in their invested assets in digital portfolios, increasing n beyond a point may of course become costly, as modeled in Kanniainen and Keuschnigg (2003).

15.4.2 *The impact of correlation*

As with mean-variance portfolios, we expect that increases in payoff correlation for Bernoulli assets will adversely impact portfolios. In order to verify this intuition we analyzed portfolios keeping all other variables the same, but changing correlation. In the previous subsection, we set the parameter for correlation to be $\rho = 0.25$. Here, we examine four levels of the correlation parameter: $\rho = \{0.09, 0.25, 0.49, 0.81\}$. For each level of correlation, we computed the normalized total payoff distribution. The number of assets is kept fixed at $n = 25$ and the probability of success of each digital asset is 0.05 as before.

The results are shown in Figure 15.5 where the probability distribution function of payoffs is shown for all four correlation levels. We find that the SSD condition is met, i.e., that lower correlation portfolios stochastically dominate (in the SSD sense) higher correlation portfolios. We also examined changing correlation in the context of a power utility investor

with the same utility function as in the previous subsection. The results are shown in Table 15.2. We confirm that, as with mean-variance portfolios, Bernoulli portfolios also improve if the assets have low correlation. Hence, digital investors should also optimally attempt to diversify their portfolios. Insurance companies are a good example—they diversify risk across geographical and other demographic divisions.

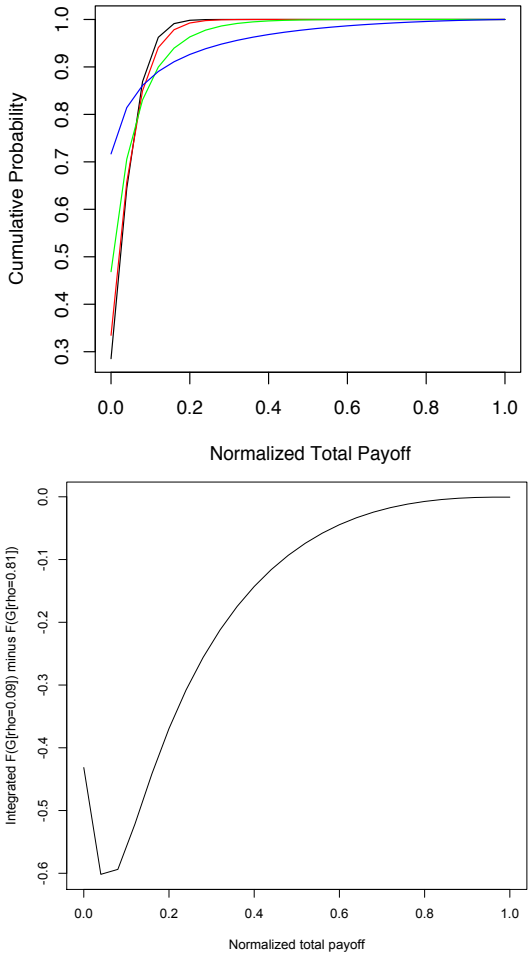


Figure 15.5: Distribution functions for returns from Bernoulli investments as the correlation parameter (ρ^2) increases. Using the recursion technique we computed the probability distribution of the portfolio payoff for four values of $\rho = \{0.09, 0.25, 0.49, 0.81\}$ shown by the black, red, green 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.

ğă

15.4.3 *Uneven bets?*

Digital asset investors are often faced with the question of whether to bet even amounts across digital investments, or to invest with different weights. We explore this question by considering two types of Bernoulli portfolios. Both have $n = 25$ assets within them, each with a success

ρ	$E(C)$	$Pr[C > 0.03]$	$Pr[C > 0.07]$	$Pr[C > 0.10]$	$Pr[C > 0.15]$	$E[U(C)]$
0.3^2	0.05	0.715	0.356	0.131	0.038	−28.112
0.5^2	0.05	0.665	0.342	0.150	0.059	−29.259
0.7^2	0.05	0.531	0.294	0.170	0.100	−32.668
0.9^2	0.05	0.283	0.186	0.139	0.110	−39.758

Table 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 $U(C) = (0.1 + C)^{1-\gamma}/(1-\gamma), \gamma = 3$.

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 $j/325, j = 1, 2, \ldots, 25$. We note that the sum of the payoffs in both cases is 1. Table 15.3 shows the utility of the investor, where the utility function is the same as in the previous sections. We see that the utility for the balanced portfolio is higher than that for the imbalanced one. Also the balanced portfolio evidences SSD over the imbalanced portfolio. However, the return distribution has fatter tails when the portfolio investments are imbalanced. Hence, investors seeking to distinguish themselves by taking on greater risk in their early careers may be better off with imbalanced portfolios.

	$E(C)$	Probability that $C > x$						
Wts	$E[U(C)]$	$x = 0.01$	$x = 0.02$	$x = 0.03$	$x = 0.07$	$x = 0.10$	$x = 0.15$	$x = 0.25$
Balanced	0.05	0.490	0.490	0.490	0.278	0.169	0.107	0.031
	−33.782							
Imbalanced	0.05	0.464	0.437	0.408	0.257	0.176	0.103	0.037
	−34.494							

Table 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 $j/325, j = 1, 2, \ldots, 25$. We note that the sum of the payoffs in both cases is 1. The correlation parameter is $\rho = 0.55$. The utility function is $U(C) = (0.1 + C)^{1-\gamma}/(1-\gamma), \gamma = 3$.

15.4.4 Mixing safe and risky assets

Is it better to have assets with a wide variation in probability of success or with similar probabilities? To examine this, we look at two portfolios of $n = 26$ assets. 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 probability distribution of payoffs and the expected utility for the same power utility investor (with $\gamma = 3$) are given in Table 15.4. We see that mixing the portfolio between investments with high and low probability of success results in higher expected utility than keeping the investments similar. We also confirmed that such imbalanced success probability portfolios also evidence SSD over portfolios with similar investments in terms of success rates. This

result does not have a natural analog in the mean-variance world with non-digital assets. For empirical evidence on the efficacy of various diversification approaches, see Lossen (2006).

	$E(C)$	Probability that $C > x$						
Wts	$E[U(C)]$	$x = 0.01$	$x = 0.02$	$x = 0.03$	$x = 0.07$	$x = 0.10$	$x = 0.15$	$x = 0.25$
Uniform	0.10	0.701	0.701	0.701	0.502	0.366	0.270	0.111
	−24.625							
Mixed	0.10	0.721	0.721	0.721	0.519	0.376	0.273	0.106
	−23.945							

Table 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 $U(C) = (0.1 + C)^{1-\gamma}/(1-\gamma), \gamma = 3$.

15.5 *Conclusions*

Digital asset portfolios are different from mean-variance ones because the asset returns are Bernoulli with small success probabilities. We used a recursion technique borrowed from the credit portfolio literature to construct the payoff distributions for Bernoulli portfolios. We find that many intuitions for these portfolios are similar to those of mean-variance ones: diversification by adding assets is useful, low correlations amongst investments is good. However, we also find that uniform bet size is preferred to some small and some large bets. Rather than construct portfolios with assets having uniform success probabilities, it is preferable to have some assets with low success rates and others with high success probabilities, a feature that is noticed in the case of venture funds. These insights augment the standard understanding obtained from mean-variance portfolio optimization.

The approach taken here is simple to use. The only inputs needed are the expected payoffs of the assets C_i, success probabilities q_i, and the average correlation between assets, given by a parameter ρ. Broad statistics on these inputs are available, say for venture investments, from papers such as Das, Jagannathan and Sarin (2003). Therefore, using data, it is easy to optimize the portfolio of a digital asset fund. The technical approach here is also easily extended to features including cost of effort by investors as the number of projects grows (Kanniainen and Keuschnigg (2003)), syndication, etc. The number of portfolios with digital assets appears to be increasing in the marketplace, and the results of this analysis provide important intuition for asset managers.

The approach in Section 2 is just one way in which to model joint success probabilities using a common factor. Undeniably, there are other

ways too, such as modeling joint probabilities directly, making sure that they are consistent with each other, which itself may be mathematically tricky. It is indeed possible to envisage that, for some different system of joint success probabilities, the qualitative nature of the results may differ from the ones developed here. It is also possible that the system we adopt here with a single common factor X may be extended to more than one common factor, an approach often taken in the default literature.

16
Against the Odds: Mathematics of Gambling

16.1 Introduction

Most people hate mathematics but love gambling. Which of course, is strange because gambling is driven mostly by math. Think of any type of gambling and no doubt there will be maths involved: Horse-track betting, sports betting, blackjack, poker, roulette, stocks, etc.

16.1.1 Odds

Oddly, bets are defined by their odds. If a bet on a horse is quoted at 4-to-1 odds, it means that if you win, you receive 4 times your wager plus the amount wagered. That is, if you bet \$1, you get back \$5.

The odds effectively define the probability of winning. Lets define this to be p. If the odds are fair, then the expected gain is zero, i.e.

$$\$4p + (1-p)(-\$1) = \$0$$

which implies that $p = 1/5$. Hence, if the odds are $x : 1$, then the probability of winning is $p = \frac{1}{x+1} = 0.2$.

16.1.2 Edge

Everyone bets because they think they have an advantage, or an edge over the others. It might be that they just think they have better information, better understanding, are using secret technology, or actually have private information (which may be illegal).

The edge is the expected profit that will be made from repeated trials relative to the bet size. You have an edge if you can win with higher probability (p^*) than $p = 1/(x+1)$. In the above example, with bet size

$1 each time, suppose your probability of winning is not $1/5$, but instead it is $1/4$. What is your edge? The expected profit is

$$(-1) \times (3/4) + 4 \times (1/4) = 1/4$$

Dividing this by the bet size (i.e. $1) gives the edge equal to $1/4$. No edge means zero or negative value betting.

16.1.3 Bookmakers

These folks set the odds. Odds are dynamic of course. If the bookie thinks the probability of a win is $1/5$, then he will set the odds to be a bit less than 4:1, maybe something like 3.5:1. In this way his expected intake minus payout is positive. At 3.5:1 odds, if there are still a lot of takers, then the bookie surely realizes that the probability of a win must be higher than in his own estimation. He also infers that $p > 1/(3.5+1)$, and will then change the odds to say 3:1. Therefore, he acts as a market maker in the bet.

16.2 Kelly Criterion

Suppose you have an edge. How should you bet over repeated plays of the game to maximize your wealth. (Do you think this is the way that hedge funds operate?) The Kelly (1956) criterion says that you should invest only a fraction of your wealth in the bet. By keeping some aside you are guaranteed to not end up in ruin.

What fraction should you bet? The answer is that you should bet

$$f = \frac{\text{Edge}}{\text{Odds}} = \frac{p^* x - (1 - p^*)}{x}$$

where the odds are expressed in the form $x : 1$. Recall that p^* is your privately known probability of winning.

16.2.1 Example

Using the same numbers as we had before, i.e., $x = 4$, $p^* = 1/4 = 0.25$, we get

$$f = \frac{0.25(4) - (1 - 0.25)}{4} = \frac{0.25}{4} = 0.0625$$

which means we invest 6.25% of the current bankroll. Lets simulate this strategy using R. Here is a simple program to simulate it, with optimal Kelly betting, and over- and under-betting.

```
#Simulation of the Kelly Criterion
#Basic data
pstar = 0.25    #private prob of winning
odds = 4                #actual odds
p = 1/(1+odds)  #house probability of winning
edge = pstar*odds - (1-pstar)
f = edge/odds
print(c("p=",p, "pstar=",pstar, "edge=",edge, "f",f))

n = 1000
x = runif(n)
f_over = 1.5*f
f_under = 0.5*f
bankroll = rep(0,n); bankroll[1]=1
br_overbet = bankroll; br_overbet[1]=1
br_underbet = bankroll; br_underbet[1]=1

for (i in 2:n) {
        if (x[i]<=pstar) {
                bankroll[i] = bankroll[i-1] + bankroll[i-1]*f*odds
                br_overbet[i] = br_overbet[i-1] + br_overbet[i-1]*f_over*odds
                br_underbet[i] = br_underbet[i-1] + br_underbet[i-1]*f_under*odds
                }
        else {
                bankroll[i] = bankroll[i-1] - bankroll[i-1]*f
                br_overbet[i] = br_overbet[i-1] - br_overbet[i-1]*f_over
                br_underbet[i] = br_underbet[i-1] - br_underbet[i-1]*f_under
                }
        }

par(mfrow=c(3,1))
plot(bankroll,type="l")
plot(br_overbet,type="l")
plot(br_underbet,type="l")
print(c(bankroll[n],br_overbet[n],br_underbet[n]))
print(c(bankroll[n]/br_overbet[n],bankroll[n]/br_underbet[n]))
```

Here is the run time listing.

```
> source("kelly.R")
[1] "p="      "0.2"     "pstar=" "0.25"    "edge="   "0.25"    "f"
[8] "0.0625" "n="      "1000"
[1] 542.29341  67.64294 158.83357
[1] 8.016999 3.414224
```

We repeat bets a thousand times. The initial pot is $1 only, but after a thousand trials, the optimal strategy ends up at $542.29, the over-betting one yields$67.64, and the under-betting one delivers $158.83. The ratio of the optimal strategy to these two sub-optimal ones is 8.02 and 3.41, respectively. This is conservative. Rerunning the model for another trial with $n = 1000$ we get:

```
> source("kelly.R")
 [1] "p="      "0.2"     "pstar=" "0.25"    "edge="   "0.25"    "f"
 [8] "0.0625" "n="      "1000"
```

```
[1] 6.426197e+15 1.734158e+12 1.313690e+12
[1] 3705.657 4891.714
```

The ratios are huge in comparison in this case, i.e., 3705 and 4891, respectively. And when we raise the trials to $n = 5000$, we have

```
> source("kelly.R")
 [1] "p="      "0.2"     "pstar="  "0.25"    "edge="   "0.25"    "f"
 [8] "0.0625" "n="      "5000"
[1] 484145279169       1837741    9450314895
[1] 263445.8383      51.2306
```

Note here that over-betting is usually worse then under-betting the Kelly optimal. Hence, many players employ what is known as the 'Half-Kelly" rule, i.e., they bet $f/2$.

Look at the resultant plot of the three strategies for the first example, shown in Figure 16.1. The top plot follows the Kelly criterion, but the other two deviate from it, by overbetting or underbetting the fraction given by Kelly.

We can very clearly see that not betting Kelly leads to far worse outcomes than sticking with the Kelly optimal plan. We ran this for 1000 periods, as if we went to the casino every day and placed one bet (or we placed four bets every minute for about four hours straight). Even within a few trials, the performance of the Kelly is remarkable. Note though that this is only one of the simulated outcomes. The simulations would result in different types of paths of the bankroll value, but generally, the outcomes are similar to what we see in the figure.

Over-betting leads to losses faster than under-betting as one would naturally expect, because it is the more risky strategy.

In this model, under the optimal rule, the probability of dropping to $1/n$ of the bankroll is $1/n$. So the probability of dropping to 90% of the bankroll ($n = 1.11$) is 0.9. Or, there is a 90% chance of losing 10% of the bankroll.

Alternate betting rules are: (a) fixed size bets, (b) double up bets. The former is too slow, the latter ruins eventually.

16.2.2 Deriving the Kelly Criterion

First we define some notation. Let B_t be the bankroll at time t. We index time as going from time $t = 1, \ldots, N$.

The odds are denoted, as before $x : 1$, and the random variable denot-

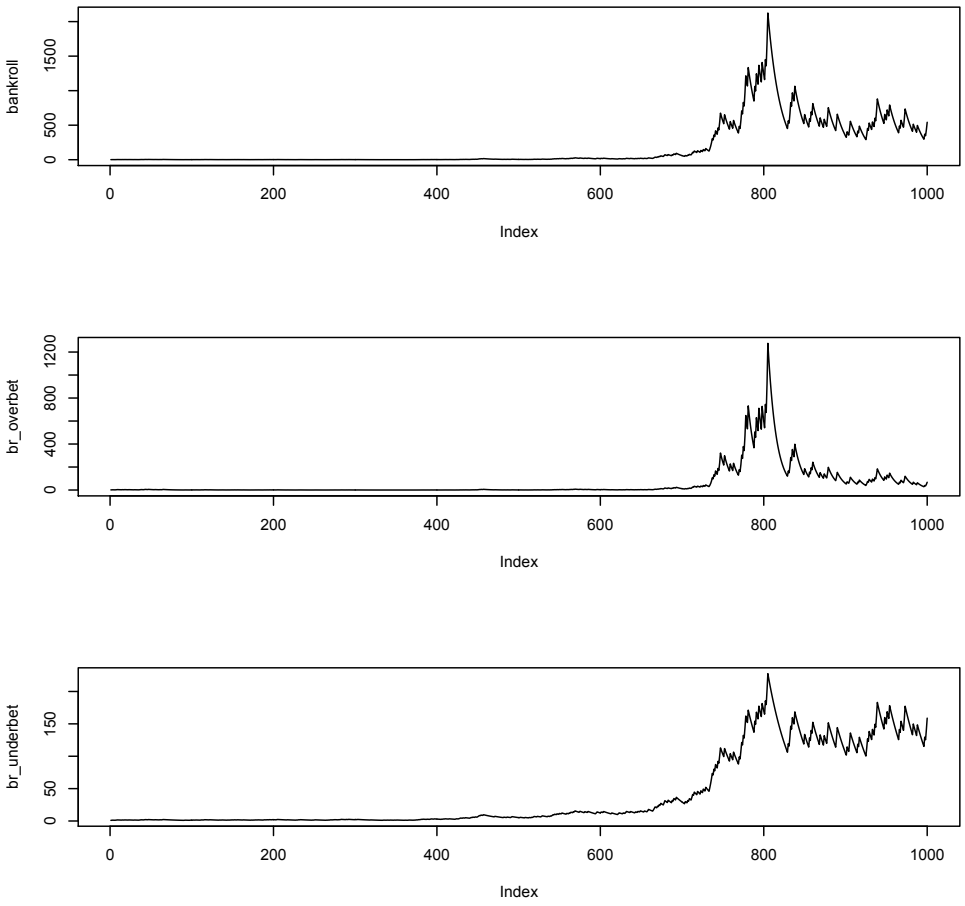


Figure 16.1: Bankroll evolution under the Kelly rule. The top plot follows the Kelly criterion, but the other two deviate from it, by overbetting or underbetting the fraction given by Kelly. The variables are: odds are 4 to 1, implying a house probability of $p = 0.2$, own probability of winning is $p^* = 0.25$.

ing the outcome (i.e., gains) of the wager is written as

$$Z_t = \begin{cases} x & \text{w/p } p \\ -1 & \text{w/p } (1-p) \end{cases}$$

We are said to have an *edge* when $E(Z_t) > 0$. The edge will be equal to $px - (1-p) > 0$.

We invest fraction f of our bankroll, where $0 < f < 1$, and since $f \neq 1$, there is no chance of being wiped out. Each wager is for an amount fB_t and returns fB_tZ_t. Hence, we may write

$$\begin{aligned} B_t &= B_{t-1} + fB_{t-1}Z_t \\ &= B_{t-1}[1 + fZ_t] \\ &= B_0 \prod_{i=1}^{t} [1 + fZ_t] \end{aligned}$$

If we define the growth rate as

$$\begin{aligned} g_t(f) &= \frac{1}{t} \ln \left(\frac{B_t}{B_0} \right) \\ &= \frac{1}{t} \ln \prod_{i=1}^{t} [1 + fZ_t] \\ &= \frac{1}{t} \sum_{i=1}^{t} \ln[1 + fZ_t] \end{aligned}$$

Taking the limit by applying the law of large numbers, we get

$$g(f) = \lim_{t \to \infty} g_t(f) = E[\ln(1 + fZ)]$$

which is nothing but the time average of $\ln(1 + fZ)$. We need to find the f that maximizes $g(f)$. We can write this more explicitly as

$$g(f) = p \ln(1 + fx) + (1-p) \ln(1-f)$$

Differentiating to get the f.o.c,

$$\frac{\partial g}{\partial f} = p \frac{x}{1 + fx} + (1-p) \frac{-1}{1-f} = 0$$

Soving this first-order condition for f gives

$$\text{The Kelly criterion: } f^* = \frac{px - (1-p)}{x}$$

This is the optimal fraction of the bankroll that should be invested in each wager. Note that we are back to the well-known formula of `Edge/Odds` we saw before.

16.3 Entropy

Entropy is defined by physicists as the extent of disorder in the universe. Entropy in the universe keeps on increasing. Things get more and more disorderly. The arrow of time moves on inexorably, and entropy keeps on increasing.

It is intuitive that as the entropy of a communication channel increases, its informativeness decreases. The connection between entropy and informativeness was made by Claude Shannon, the father of information theory. It was his PhD thesis at MIT. See Shannon (1948).

With respect to probability distributions, entropy of a discrete distribution taking values $\{p_1, p_2, \ldots, p_K\}$ is

$$H = -\sum_{j=1}^{K} p_j \ln(p_j)$$

For the simple wager we have been considering, entropy is

$$H = -[p \ln p + (1-p)\ln(1-p)]$$

This is called Shannon entropy after his seminal work in 1948. For $p = 1/2, 1/5, 1/100$ entropy is

```
> p=0.5; -(p*log(p)+(1-p)*log(1-p))
[1] 0.6931472
> p=0.2; -(p*log(p)+(1-p)*log(1-p))
[1] 0.5004024
> p=0.01; -(p*log(p)+(1-p)*log(1-p))
[1] 0.05600153
```

We see various probability distributions in decreasing order of entropy. At $p = 0.5$ entropy is highest.

Note that the normal distribution is the one with the highest entropy in its class of distributions.

16.3.1 Linking the Kelly Criterion to Entropy

For the particular case of a simple random walk, we have odds $x = 1$. In this case,

$$f^* = p - (1-p) = 2p - 1$$

where we see that $p = 1/2$, and the optimal average bet value is

$$\begin{aligned} g^* &= p\ln(1+f) + (1-p)\ln(1-f) \\ &= p\ln(2p) + (1-p)\ln[2(1-p)] \\ &= \ln 2 + p\ln p + (1-p)\ln(1-p) \\ &= \ln 2 - H \end{aligned}$$

where H is the entropy of the distribution of Z. For $p = 0.5$, we have

$$g^* = \ln 2 - 0.5\ln(0.5) - 0.5\ln(0.5) = 1.386294$$

We note that g^* is decreasing in entropy, because informativeness declines with entropy and so the portfolio earns less if we have less of an edge, i.e. our winning information is less than perfect.

16.3.2 Linking the Kelly criterion to portfolio optimization

A small change in the mathematics above leads to an analogous concept for portfolio policy. The value of a portfolio follows the dynamics below

$$B_t = B_{t-1}[1 + (1-f)r + fZ_t] = B_0 \prod_{i=1}^{t}[1 + r + f(Z_t - r)]$$

Hence, the growth rate of the portfolio is given by

$$\begin{aligned} g_t(f) &= \frac{1}{t}\ln\left(\frac{B_t}{B_0}\right) \\ &= \frac{1}{t}\ln\left(\prod_{i=1}^{t}[1 + r + f(Z_t - r)]\right) \\ &= \frac{1}{t}\sum_{i=1}^{t}\ln\left([1 + r + f(Z_t - r)]\right) \end{aligned}$$

Taking the limit by applying the law of large numbers, we get

$$g(f) = \lim_{t\to\infty} g_t(f) = E[\ln(1 + r + f(Z - r))]$$

Hence, maximizing the growth rate of the portfolio is the same as maximizing expected log utility. For a much more detailed analysis, see Browne and Whitt (1996).

16.3.3 Implementing day trading

We may choose any suitable distribution for the asset Z. Suppose Z is normally distributed with mean μ and variance σ^2. Then we just need to

find f such that we have

$$f^* = \text{argmax}_f \; E[\ln(1 + r + f(Z - r))]$$

This may be done numerically. Note now that this does not guarantee that $0 < f < 1$, which does not preclude ruin.

How would a day-trader think about portfolio optimization? His problem would be closer to that of a gambler's because he is very much like someone at the tables, making a series of bets, whose outcomes become known in very short time frames. A day-trader can easily look at his history of round-trip trades and see how many of them made money, and how many lost money. He would then obtain an estimate of p, the probability of winning, which is the fraction of total round-trip trades that make money.

The Lavinio (2000) d-ratio is known as the 'gain-loss" ratio and is as follows:

$$d = \frac{n_d \times \sum_{j=1}^{n} \max(0, -Z_j)}{n_u \times \sum_{j=1}^{n} \max(0, Z_j)}$$

where n_d is the number of down (loss) trades, and n_u is the number of up (gain) trades and $n = n_d + n_u$, and Z_j are the returns on the trades. In our original example at the beginning of this chapter, we have odds of 4:1, implying $n_d = 4$ loss trades for each win ($n_u = 1$) trade, and a winning trade nets $+4$, and a losing trade nets -1. Hence, we have

$$d = \frac{4 \times (1 + 1 + 1 + 1)}{1 \times 4} = 4 = x$$

which is just equal to the odds. Once, these are computed, the day-trader simply plugs them in to the formula we had before, i.e.,

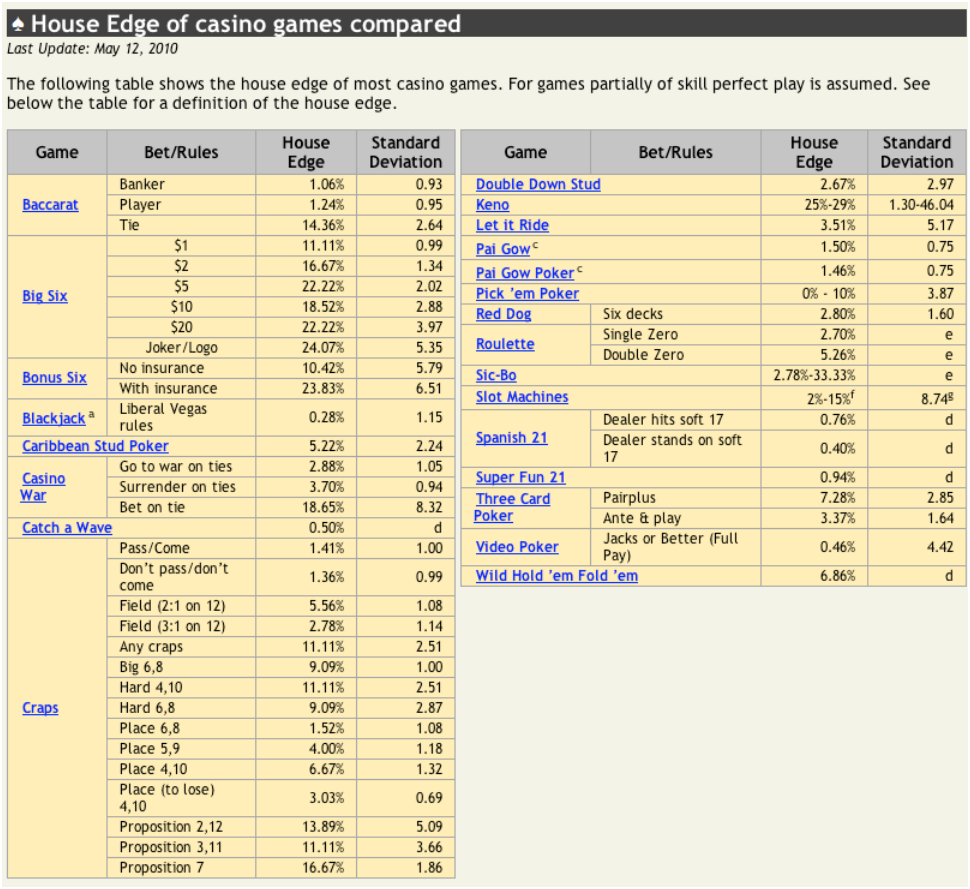
$$f = \frac{px - (1 - p)}{x} = p - \frac{(1 - p)}{x}$$

Of course, here $p = 0.2$. A trader would also constantly re-assess the values of p and x given that the markets change over time.

16.4 Casino Games

The statistics of various casino games are displayed in Figure 16.2. To recap, note that the Kelly criterion maximizes the average bankroll and also minimizes the risk of ruin, but is of no use if the house had an edge. *You* need to have an edge before it works. But then it really works! It is

not a short-term formula and works over a long sequence of bets. Naturally it follows that it also minimizes the number of bets needed to double the bankroll.



♠ House Edge of casino games compared

Last Update: May 12, 2010

The following table shows the house edge of most casino games. For games partially of skill perfect play is assumed. See below the table for a definition of the house edge.

Game	Bet/Rules	House Edge	Standard Deviation
Baccarat	Banker	1.06%	0.93
	Player	1.24%	0.95
	Tie	14.36%	2.64
Big Six	$1	11.11%	0.99
	$2	16.67%	1.34
	$5	22.22%	2.02
	$10	18.52%	2.88
	$20	22.22%	3.97
	Joker/Logo	24.07%	5.35
Bonus Six	No insurance	10.42%	5.79
	With insurance	23.83%	6.51
Blackjack [a]	Liberal Vegas rules	0.28%	1.15
Caribbean Stud Poker		5.22%	2.24
Casino War	Go to war on ties	2.88%	1.05
	Surrender on ties	3.70%	0.94
	Bet on tie	18.65%	8.32
Catch a Wave		0.50%	d
Craps	Pass/Come	1.41%	1.00
	Don't pass/don't come	1.36%	0.99
	Field (2:1 on 12)	5.56%	1.08
	Field (3:1 on 12)	2.78%	1.14
	Any craps	11.11%	2.51
	Big 6,8	9.09%	1.00
	Hard 4,10	11.11%	2.51
	Hard 6,8	9.09%	2.87
	Place 6,8	1.52%	1.08
	Place 5,9	4.00%	1.18
	Place 4,10	6.67%	1.32
	Place (to lose) 4,10	3.03%	0.69
	Proposition 2,12	13.89%	5.09
	Proposition 3,11	11.11%	3.66
	Proposition 7	16.67%	1.86

Game	Bet/Rules	House Edge	Standard Deviation
Double Down Stud		2.67%	2.97
Keno		25%-29%	1.30-46.04
Let it Ride		3.51%	5.17
Pai Gow [c]		1.50%	0.75
Pai Gow Poker [c]		1.46%	0.75
Pick 'em Poker		0% - 10%	3.87
Red Dog	Six decks	2.80%	1.60
Roulette	Single Zero	2.70%	e
	Double Zero	5.26%	e
Sic-Bo		2.78%-33.33%	e
Slot Machines		2%-15% [f]	8.74 [g]
Spanish 21	Dealer hits soft 17	0.76%	d
	Dealer stands on soft 17	0.40%	d
Super Fun 21		0.94%	d
Three Card Poker	Pairplus	7.28%	2.85
	Ante & play	3.37%	1.64
Video Poker	Jacks or Better (Full Pay)	0.46%	4.42
Wild Hold 'em Fold 'em		6.86%	d

Figure 16.2: See `http://wizardofodds.com/gambling/h` 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.

In a neat paper, Thorp (1997) presents various Kelly rules for blackjack, sports betting, and the stock market. Reading Thorp (1962) for blackjack is highly recommended. And of course there is the great story of the MIT Blackjack Team in Mezrich (2003). Here is an example from Thorp (1997).

Suppose you have an edge where you can win $+1$ with probability 0.51, and lose -1 with probability 0.49 when the blackjack deck is "hot" and when it is cold the probabilities are reversed. We will bet f on the hot deck and $af, a < 1$ on the cold deck. We have to bet on cold decks just to prevent the dealer from getting suspicious. Hot and cold decks

occur with equal probability. Then the Kelly growth rate is

$$g(f) = 0.5[0.51\ln(1+f) + 0.49\ln(1-f)] + 0.5[0.49\ln(1+af) + 0.51\ln(1-af)]$$

If we do not bet on cold decks, then $a = 0$ and $f^* = 0.02$ using the usual formula. As a increases from 0 to 1, we see that f^* decreases. Hence, we bet less of our pot to make up for losses from cold decks. We compute this and get the following:

$$\begin{aligned} a = 0 &\rightarrow f^* = 0.020 \\ a = 1/4 &\rightarrow f^* = 0.014 \\ a = 1/2 &\rightarrow f^* = 0.008 \\ a = 3/4 &\rightarrow f^* = 0.0032 \end{aligned}$$

17
In the Same Boat: Cluster Analysis and Prediction Trees

17.1 Introduction

There are many aspects of data analysis that call for grouping individuals, firms, projects, etc. These fall under the rubric of what may be termed as "classification" analysis. Cluster analysis comprises a group of techniques that uses distance metrics to bunch data into categories.

There are two broad approaches to cluster analysis:

1. Agglomerative or Hierarchical or Bottom-up: In this case we begin with all entities in the analysis being given their own cluster, so that we start with n clusters. Then, entities are grouped into clusters based on a given distance metric between each pair of entities. In this way a *hierarchy* of clusters is built up and the researcher can choose which grouping is preferred.
2. Partitioning or Top-down: In this approach, the entire set of n entities is assumed to be a cluster. Then it is progressively partitioned into smaller and smaller clusters.

We will employ both clustering approaches and examine their properties with various data sets as examples.

17.2 Clustering using k-means

This approach is bottom-up. If we have a sample of n observations to be allocated to k clusters, then we can initialize the clusters in many ways. One approach is to assume that each observation is a cluster unto itself. We proceed by taking each observation and allocating it to the nearest cluster using a distance metric. At the outset, we would simply allocate an observation to its nearest neighbor.

How is nearness measured? We need a distance metric, and one common one is Euclidian distance. Suppose we have two observations x_i and x_j. These may be represented by a vector of attributes. Suppose our observations are people, and the attributes are {height, weight, IQ} = $x_i = \{h_i, w_i, I_i\}$ for the i-th individual. Then the Euclidian distance between two individuals i and j is

$$d_{ij} = \sqrt{(h_i - h_j)^2 + (w_i - w_j)^2 + (I_i - I_j)^2}$$

In contrast, the "Manhattan" distance is given by (when is this more appropriate?)

$$d_{ij} = |h_i - h_j| + |w_i - w_j| + |I_i - I_j|$$

We may use other metrics such as the cosine distance, or the Mahalanobis distance. A matrix of $n \times n$ values of all d_{ij}s is called the "distance matrix." Using this distance metric we assign nodes to clusters or attach them to nearest neighbors. After a few iterations, no longer are clusters made up of singleton observations, and the number of clusters reaches k, the preset number required, and then all nodes are assigned to one of these k clusters. As we examine each observation we then assign it (or re-assign it) to the nearest cluster, where the distance is measured from the observation to some representative node of the cluster. Some common choices of the representative node in a cluster of are:

1. Centroid of the cluster. This is the mean of the observations in the cluster for each attribute. The centroid of the two observations above is the average vector $\{(h_i + h_j)/2, (w_i + w_j)/2, (I_i + I_j)/2\}$. This is often called the "center" of the cluster. If there are more nodes then the centroid is the average of the same coordinate for all nodes.

2. Closest member of the cluster.

3. Furthest member of the cluster.

The algorithm converges when no re-assignments of observations to clusters occurs. Note that k-means is a random algorithm, and may not always return the same clusters every time the algorithm is run. Also, one needs to specify the number of clusters to begin with and there may be no a-priori way in which to ascertain the correct number. Hence, trial and error and examination of the results is called for. Also, the algorithm aims to have balanced clusters, but this may not always be appropriate.

In R, we may construct the distance matrix using the `dist` function. Using the NCAA data we are already familiar with, we have:

```
> ncaa = read.table("ncaa.txt",header=TRUE)
> names(ncaa)
 [1] "No"   "NAME" "GMS"  "PTS"  "REB"  "AST"  "TO"   "A.T"  "STL"  "BLK"
[11] "PF"   "FG"   "FT"   "X3P"
> d = dist(ncaa[,3:14], method="euclidian")
```

Examining this matrix will show that it contains $n(n-1)/2$ elements, i.e., the number of pairs of nodes. Only the lower triangular matrix of d is populated.

It is important to note that since the size of the variables is very different, simply applying the `dist` function is not advised, as the larger variables swamp the distance calculation. It is best to normalize the variables first, before calculating distances. The `scale` function in R is simple to apply as follows.

```
> ncaa_data = as.matrix(ncaa[,3:14])
> summary(ncaa_data)
      GMS              PTS              REB              AST              TO
 Min.   :1.000   Min.   :46.00   Min.   :19.00   Min.   : 2.00   Min.   : 5.00
 1st Qu.:1.000   1st Qu.:61.75   1st Qu.:31.75   1st Qu.:10.00   1st Qu.:11.00
 Median :2.000   Median :67.00   Median :34.35   Median :13.00   Median :13.50
 Mean   :1.984   Mean   :67.10   Mean   :34.47   Mean   :12.75   Mean   :13.96
 3rd Qu.:2.250   3rd Qu.:73.12   3rd Qu.:37.20   3rd Qu.:15.57   3rd Qu.:17.00
 Max.   :6.000   Max.   :88.00   Max.   :43.00   Max.   :20.00   Max.   :24.00
      A.T              STL              BLK              PF
 Min.   :0.1500   Min.   : 2.000   Min.   :0.000   Min.   :12.00
 1st Qu.:0.7400   1st Qu.: 5.000   1st Qu.:1.225   1st Qu.:16.00
 Median :0.9700   Median : 7.000   Median :2.750   Median :19.00
 Mean   :0.9778   Mean   : 6.823   Mean   :2.750   Mean   :18.66
 3rd Qu.:1.2325   3rd Qu.: 8.425   3rd Qu.:4.000   3rd Qu.:20.00
 Max.   :1.8700   Max.   :12.000   Max.   :6.500   Max.   :29.00
       FG               FT              X3P
 Min.   :0.2980   Min.   :0.2500   Min.   :0.0910
 1st Qu.:0.3855   1st Qu.:0.6452   1st Qu.:0.2820
 Median :0.4220   Median :0.7010   Median :0.3330
 Mean   :0.4233   Mean   :0.6915   Mean   :0.3334
 3rd Qu.:0.4632   3rd Qu.:0.7705   3rd Qu.:0.3940
 Max.   :0.5420   Max.   :0.8890   Max.   :0.5220
> ncaa_data = scale(ncaa_data)
```

The `scale` function above normalizes all columns of data. If you run `summary` again, all variables will have mean zero and unit standard deviation. Here is a check.

```
> round(apply(ncaa_data,2,mean),2)
GMS PTS REB AST  TO A.T STL BLK  PF  FG  FT X3P
  0   0   0   0   0   0   0   0   0   0   0   0
> apply(ncaa_data,2,sd)
GMS PTS REB AST  TO A.T STL BLK  PF  FG  FT X3P
  1   1   1   1   1   1   1   1   1   1   1   1
```

Clustering takes many observations with their characteristics and then allocates them into buckets or clusters based on their similarity. In finance, we may use cluster analysis to determine groups of similar firms. For example, see Figure 17.1, where I ran a cluster analysis on VC

financing of startups to get a grouping of types of venture financing into different styles.

Unlike regression analysis, cluster analysis uses only the right-hand side variables, and there is no dependent variable required. We group observations purely on their overall similarity across characteristics. Hence, it is closely linked to the notion of "communities" that we studied earlier, though that concept lives in the domain of networks.

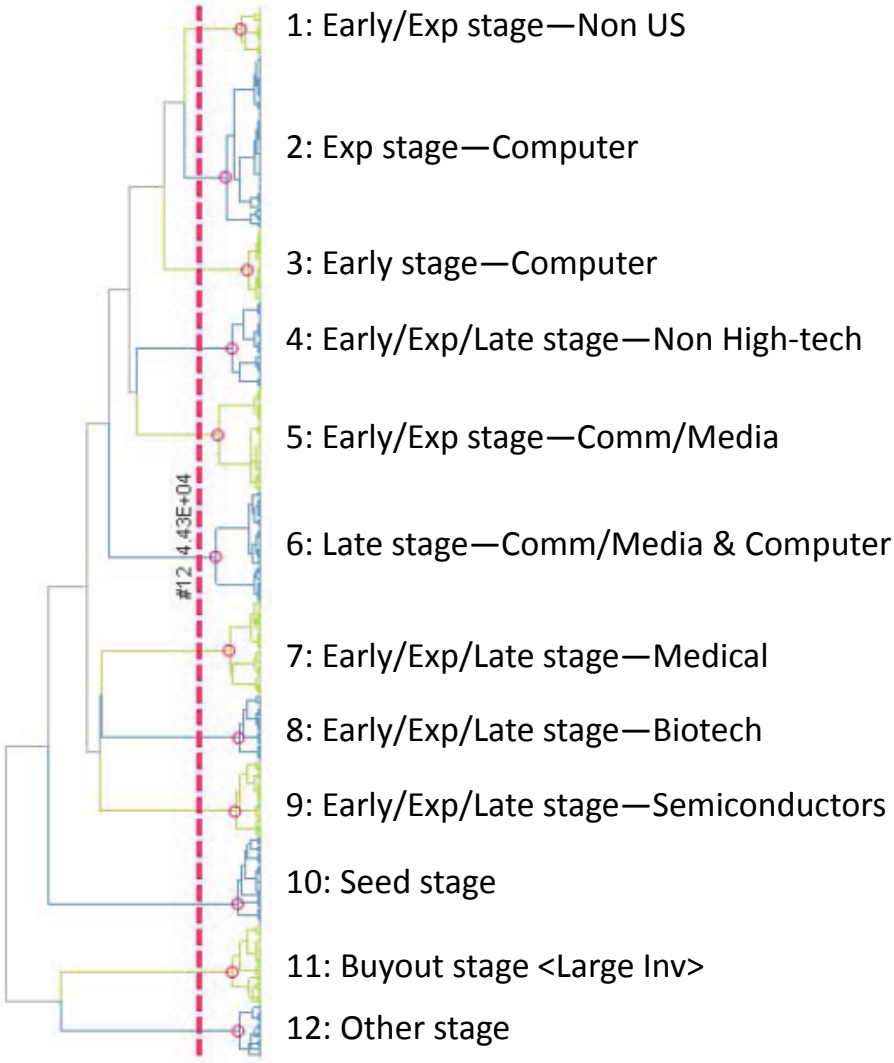


Figure 17.1: VC Style Clusters.

17.2.1 Example: Randomly generated data in kmeans

Here we use the example from the `kmeans` function to see how the clusters appear. This function is standard issue, i.e., it comes with the `stats`

package, which is included in the base R distribution and does not need to be separately installed. The data is randomly generated but has two bunches of items with different means, so we should be easily able to see two separate clusters. You will need the `graphics` package which is also in the base installation.

```
> require(graphics)
>
> # a 2-dimensional example
> x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
+            matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
> colnames(x) <- c("x", "y")
> (cl <- kmeans(x, 2))
K-means clustering with 2 clusters of sizes 52, 48

Cluster means:
           x           y
1  0.98813364  1.01967200
2 -0.02752225 -0.02651525

Clustering vector:
  [1] 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
 [36] 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [71] 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 2

Within cluster sum of squares by cluster:
[1] 10.509092  6.445904

Available components:
[1] "cluster"  "centers"  "withinss" "size"
> plot(x, col = cl$cluster)
> points(cl$centers, col = 1:2, pch = 8, cex=2)
```

The plotted clusters appear in Figure 17.2.

We can also examine the same example with 5 clusters. The output is shown in Figure 17.3

```
> ## random starts do help here with too many clusters
> (cl <- kmeans(x, 5, nstart = 25))
K-means clustering with 5 clusters of sizes 25, 22, 16, 20, 17

Cluster means:
           x          y
1 -0.1854632  0.1129291
2  0.1321432 -0.2089422
3  0.9217674  0.6424407
4  0.7404867  1.2253548
5  1.3078410  1.1022096

Clustering vector:
  [1] 1 2 1 1 2 2 2 4 2 1 2 1 1 1 1 2 2 1 2 1 1 1 1 2 2 1 1 2 2 3 1 2 2 1 2
 [36] 2 3 2 2 1 1 2 1 1 1 1 1 2 1 2 5 5 4 4 4 4 4 4 5 4 5 4 5 5 5 5 3 4 3 3
 [71] 3 3 3 5 5 5 5 5 4 5 4 4 3 4 5 3 5 4 3 5 4 4 3 3 4 3 4 3 4 3

Within cluster sum of squares by cluster:
[1] 2.263606 1.311527 1.426708 2.084694 1.329643

Available components:
[1] "cluster"  "centers"  "withinss" "size"
> plot(x, col = cl$cluster)
> points(cl$centers, col = 1:5, pch = 8)
```

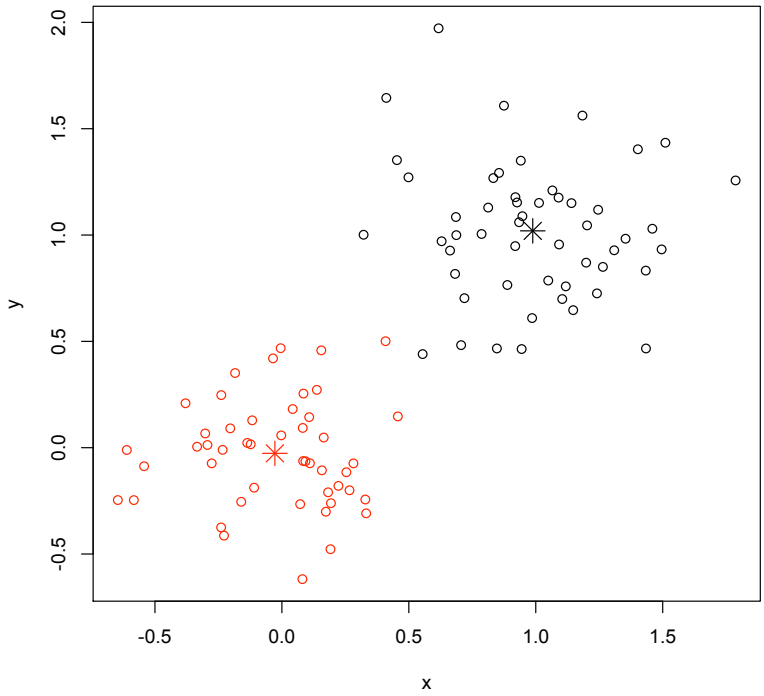


Figure 17.2: Two cluster example.

17.2.2 *Example: Clustering of VC financing rounds*

In this section we examine data on VC's financing of startups from 2001–2006, using data on individual financing rounds. The basic information that we have is shown below.

```
> data = read.csv("vc_clust.csv",header=TRUE,sep=",")
> dim(data)
[1] 3697   47
> names(data)
 [1] "fund_name"          "fund_year"          "fund_avg_rd_invt"
 [4] "fund_avg_co_invt"   "fund_num_co"        "fund_num_rds"
 [7] "fund_tot_invt"      "stage_num1"         "stage_num2"
[10] "stage_num3"         "stage_num4"         "stage_num5"
[13] "stage_num6"         "stage_num7"         "stage_num8"
[16] "stage_num9"         "stage_num10"        "stage_num11"
[19] "stage_num12"        "stage_num13"        "stage_num14"
[22] "stage_num15"        "stage_num16"        "stage_num17"
[25] "invest_type_num1"   "invest_type_num2"   "invest_type_num3"
[28] "invest_type_num4"   "invest_type_num5"   "invest_type_num6"
[31] "fund_nation_US"     "fund_state_CAMA"    "fund_type_num1"
```

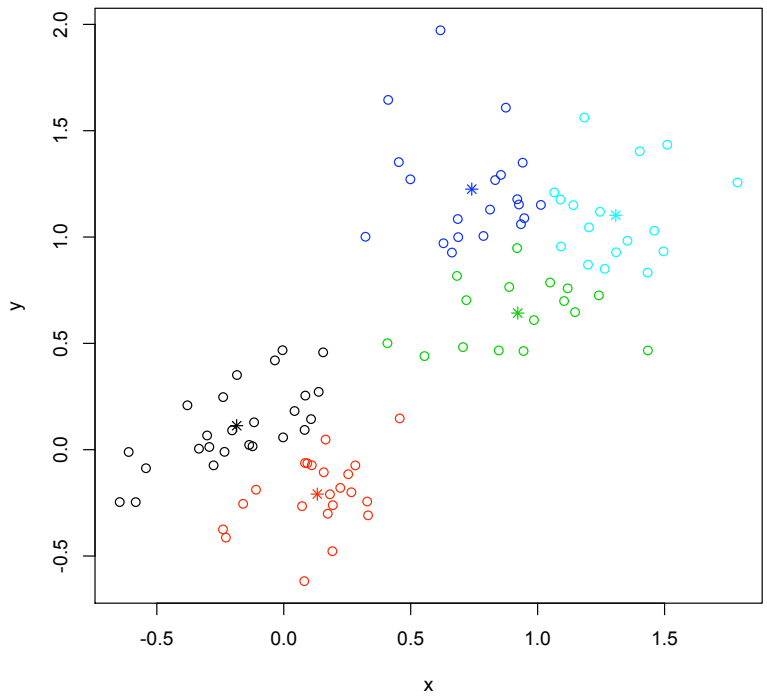


Figure 17.3: Five cluster example.

```
[34] "fund_type_num2"    "fund_type_num3"    "fund_type_num4"
[37] "fund_type_num5"    "fund_type_num6"    "fund_type_num7"
[40] "fund_type_num8"    "fund_type_num9"    "fund_type_num10"
[43] "fund_type_num11"   "fund_type_num12"   "fund_type_num13"
[46] "fund_type_num14"   "fund_type_num15"
```

We clean out all rows that have missing values as follows:

```
> idx = which(rowSums(is.na(data))==0)
> length(idx)
[1] 2975
> data = data[idx,]
> dim(data)
[1] 2975   47
```

We run a first-cut *k*-means analysis using limited data.

```
> idx = c(3,6,31,32)
> cdata = data[,idx]
> names(cdata)
[1] "fund_avg_rd_invt" "fund_num_rds"     "fund_nation_US"
[4] "fund_state_CAMA"
> fit = kmeans(cdata,4)
> fit$size
[1] 2856    2   95   22
> fit$centers
  fund_avg_rd_invt fund_num_rds fund_nation_US fund_state_CAMA
1         4714.894     8.808824       0.5560224       0.2244398
2      1025853.650     7.500000       0.0000000       0.0000000
3        87489.873     6.400000       0.4631579       0.1368421
4       302948.114     5.318182       0.7272727       0.2727273
```

We see that the clusters are hugely imbalanced, with one cluster accounting for most of the investment rounds. Let's try a different cut now. Using investment type = {buyout, early, expansion, late, other, seed} types of financing, we get the following, assuming 4 clusters.

```
> idx = c(25,26,27,28,29,30,31,32)
> cdata = data[,idx]
> names(cdata)
[1] "invest_type_num1" "invest_type_num2" "invest_type_num3"
[4] "invest_type_num4" "invest_type_num5" "invest_type_num6"
[7] "fund_nation_US"   "fund_state_CAMA"
> fit = kmeans(cdata,4)
> fit$size
[1] 2199   65  380  331
> fit$centers
  invest_type_num1 invest_type_num2 invest_type_num3 invest_type_num4
1        0.0000000        0.00000000        0.00000000        0.00000000
2        0.0000000        0.00000000        0.00000000        0.00000000
3        0.6868421        0.12631579        0.06052632        0.12631579
4        0.4592145        0.09969789        0.39274924        0.04833837
  invest_type_num5 invest_type_num6 fund_nation_US fund_state_CAMA
1                0                1      0.5366075       0.2391996
2                1                0      0.7538462       0.1692308
3                0                0      1.0000000       0.3236842
4                0                0      0.1178248       0.0000000
```

Here we get a very different outcome. Now, assuming 6 clusters, we have:

```
> idx = c(25,26,27,28,29,30,31,32)
> cdata = data[,idx]
> fit = kmeans(cdata,6)
> fit$size
[1]   34  526  176  153 1673  413
> fit$centers
  invest_type_num1 invest_type_num2 invest_type_num3 invest_type_num4
1                0        0.3235294                0        0.3529412
2                0        0.0000000                0        0.0000000
3                0        0.3977273                0        0.2954545
4                0        0.0000000                1        0.0000000
5                0        0.0000000                0        0.0000000
6                1        0.0000000                0        0.0000000
  invest_type_num5 invest_type_num6 fund_nation_US fund_state_CAMA
1        0.3235294                0      1.0000000       1.0000000
2        0.0000000                1      1.0000000       1.0000000
3        0.3068182                0      0.6306818       0.0000000
4        0.0000000                0      0.4052288       0.1503268
5        0.0000000                1      0.3909145       0.0000000
6        0.0000000                0      0.6319613       0.1864407
```

17.2.3 NCAA teams

We revisit our NCAA data set, and form clusters there.

```
> ncaa = read.table("ncaa.txt",header=TRUE)
> names(ncaa)
 [1] "No"   "NAME" "GMS"  "PTS"  "REB"  "AST"  "TO"   "A.T"  "STL"  "BLK"
[11] "PF"   "FG"   "FT"   "X3P"
```

```
> fit = kmeans(ncaa[,3:14],4)
> fit$size
[1] 14 17 27  6
> fit$centers
       GMS      PTS      REB      AST       TO      A.T      STL
1 3.357143 80.12857 34.15714 16.357143 13.70714 1.2357143 6.821429
2 1.529412 60.24118 38.76471  9.282353 16.45882 0.5817647 6.882353
3 1.777778 68.39259 33.17407 13.596296 12.83704 1.1107407 6.822222
4 1.000000 50.33333 28.83333 10.333333 12.50000 0.9000000 6.666667
       BLK       PF        FG        FT       X3P
1 2.514286 18.48571 0.4837143 0.7042143 0.4035714
2 2.882353 18.51176 0.3838824 0.6683529 0.3091765
3 2.918519 18.68519 0.4256296 0.7071852 0.3263704
4 2.166667 19.33333 0.3835000 0.6565000 0.2696667
> idx = c(4,6); plot(ncaa[,idx],col=fit$cluster)
```

See Figure 17.4. Since there are more than two attributes of each observation in the data, we picked two of them {AST, PTS} and plotted the clusters against those.

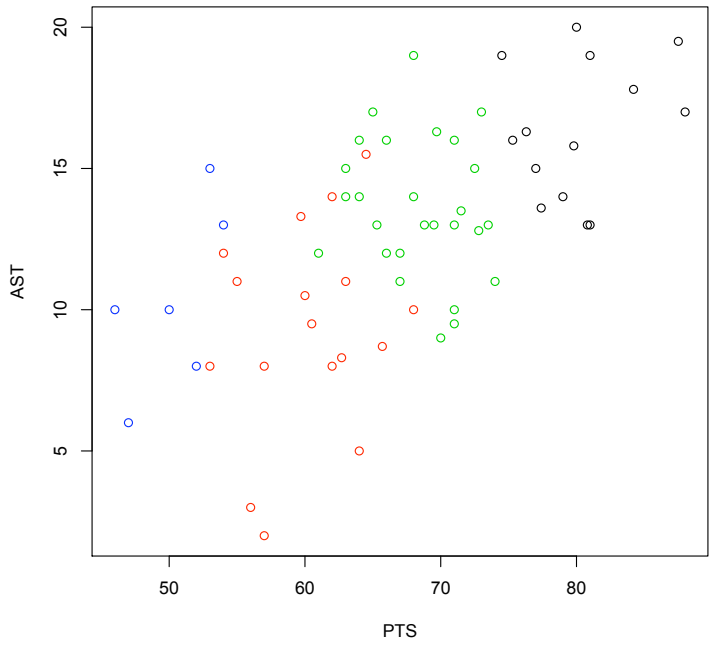


Figure 17.4: NCAA cluster example.

17.3 Hierarchical Clustering

Hierarchical clustering is both, a top-down (divisive) approach and bottom-up (agglomerative) approach. At the top level there is just one cluster. A level below, this may be broken down into a few clusters, which are then further broken down into more sub-clusters a level below, and so on. This clustering approach is computationally expensive, and the divisive approach is exponentially expensive in n, the number of entities being clustered. In fact, the algorithm is $\mathcal{O}(2^n)$.

The function for clustering is `hclust` and is included in the `stats` package in the base R distribution.

We re-use the NCAA data set one more time.

```
> d = dist(ncaa[,3:14], method="euclidian")
> fit = hclust(d, method="ward")
> names(fit)
[1] "merge"       "height"      "order"       "labels"      "method"
[6] "call"        "dist.method"
> plot(fit,main="NCAA Teams")
> groups = cutree(fit, k=4)
> rect.hclust(fit, k=4, border="blue")
```

We begin by first computing the distance matrix. Then we call the `hclust` function and the `plot` function applied to object `fit` gives what is known as a "dendrogram" plot, showing the cluster hierarchy. We may pick clusters at any level. In this case, we chose a "cut" level such that we get four clusters, and the `rect.hclust` function allows us to superimpose boxes on the clusters so we can see the grouping more clearly. The result is plotted in Figure 17.5.

We can also visualize the clusters loaded on to the top two principal components as follows, using the `clusplot` function that resides in package `cluster`. The result is plotted in Figure 17.6.

```
> groups
 [1] 1 1 1 1 1 2 1 1 3 2 1 3 3 1 1 1 2 3 3 2 3 2 1 1 3 3 1 3 2 3 3 3 1 2 2
[36] 3 3 4 1 2 4 4 4 3 3 2 4 3 1 3 3 4 1 2 4 3 3 3 3 4 4 4 4 3
> library(cluster)
> clusplot(ncaa[,3:14],groups,color=TRUE,shade=TRUE,labels=2,lines=0)
```

17.4 Prediction Trees

Prediction trees are a natural outcome of recursive partitioning of the data. Hence, they are a particular form of clustering at different levels.

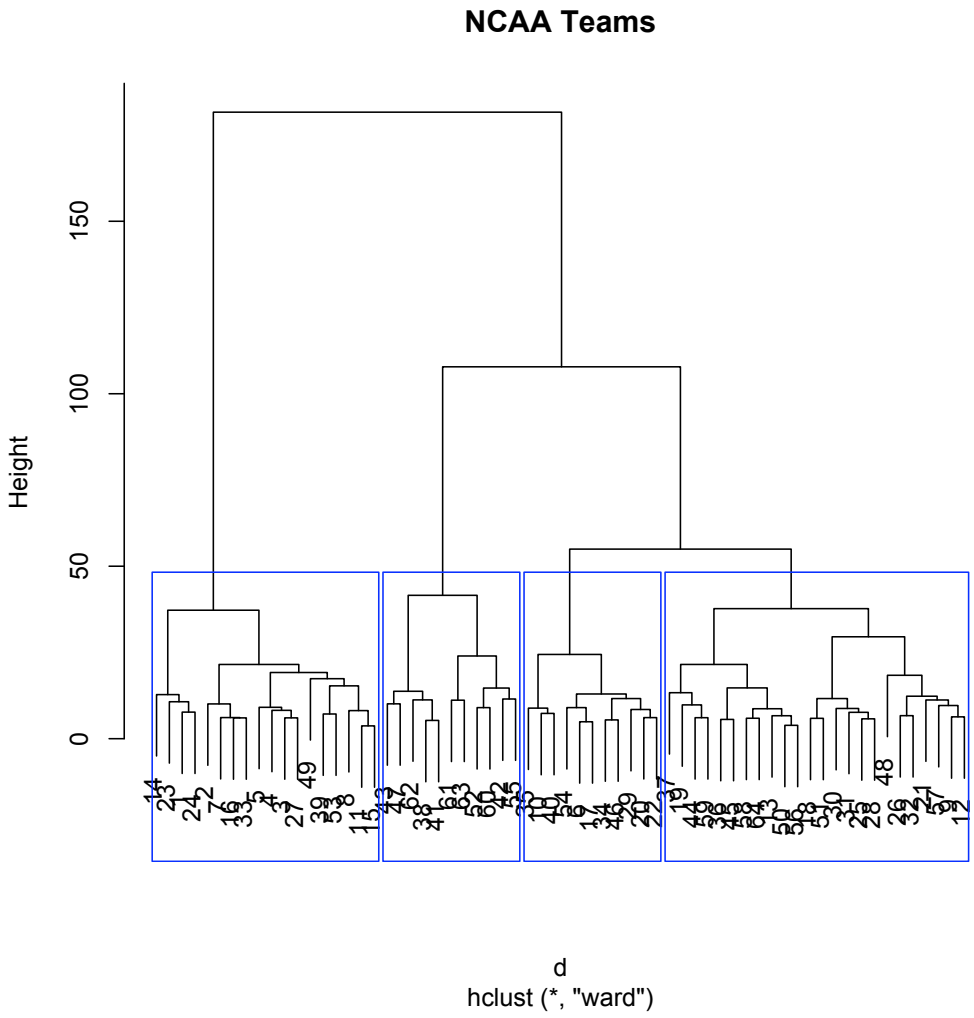


Figure 17.5: NCAA data, hierarchical cluster example.

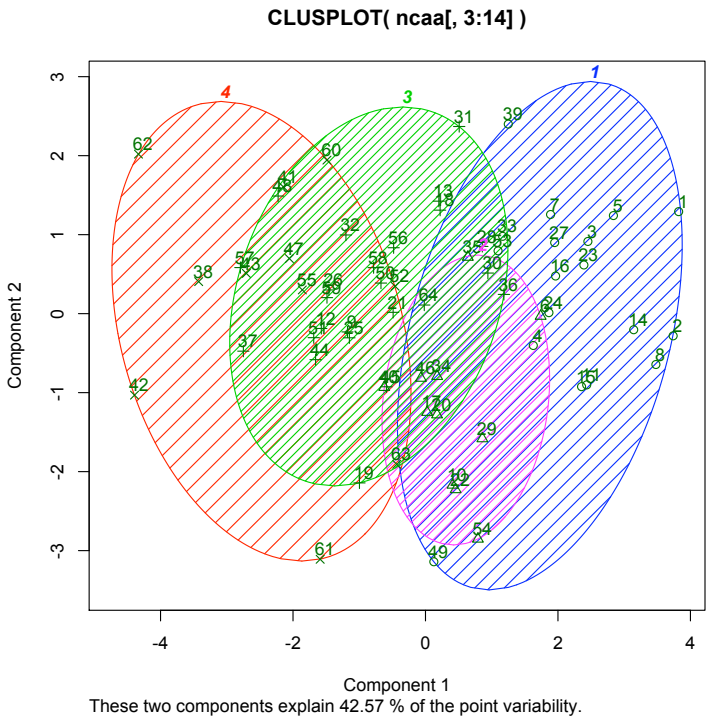


Figure 17.6: NCAA data, hierarchical cluster example with clusters on the top two principal components.

Usual cluster analysis results in a "flat" partition, but prediction trees develop a multi-level cluster of trees. The term used here is CART, which stands for classification analysis and regression trees. But prediction trees are different from vanilla clustering in an important way – there is a dependent variable, i.e., a category or a range of values (e.g., a score) that one is attempting to predict.

Prediction trees are of two types: (a) Classification trees, where the leaves of the trees are different categories of discrete outcomes. and (b) Regression trees, where the leaves are continuous outcomes. We may think of the former as a generalized form of limited dependent variables, and the latter as a generalized form of regression analysis.

To set ideas, suppose we want to predict the credit score of an individual using age, income, and education as explanatory variables. Assume that income is the best explanatory variable of the three. Then, at the top of the tree, there will be income as the branching variable, i.e., if income is less than some threshold, then we go down the left branch of the tree, else we go down the right. At the next level, it may be that we use education to make the next bifurcation, and then at the third level we use age. A variable may even be repeatedly used at more than one level.

This leads us to several leaves at the bottom of the tree that contain the average values of the credit scores that may be reached. For example if we get an individual of young age, low income, and no education, it is very likely that this path down the tree will lead to a low credit score on average. Instead of credit score (an example of a regression tree), consider credit ratings of companies (an example of a classification tree). These ideas will become clearer once we present some examples.

Recursive partitioning is the main algorithmic construct behind prediction trees. We take the data and using a single explanatory variable, we try and bifurcate the data into two categories such that the additional information from categorization results in better "information" than before the binary split. For example, suppose we are trying to predict who will make donations and who will not using a single variable – income. If we have a sample of people and have not yet analyzed their incomes, we only have the raw frequency p of how many people made donations, i.e., and number between 0 and 1. The "information" of the predicted likelihood p is inversely related to the sum of squared errors (SSE) between this value p and the 0 values and 1 values of the observations.

$$SSE_1 = \sum_{i=1}^{n} (x_i - p)^2$$

where $x_i = \{0, 1\}$, depending on whether person i made a donation or not. Now, if we bifurcate the sample based on income, say to the left we have people with income less than K, and to the right, people with incomes greater than or equal to K. If we find that the proportion of people on the left making donations is $p_L < p$ and on the right is $p_R > p$, our new information is:

$$SSE_2 = \sum_{i, Income < K} (x_i - p_L)^2 + \sum_{i, Income \geq K} (x_i - p_R)^2$$

By choosing K correctly, our recursive partitioning algorithm will maximize the gain, i.e., $\delta = (SSE_1 - SSE_2)$. We stop branching further when at a given tree level δ is less than a pre-specified threshold.

We note that as n gets large, the computation of binary splits on any variable is expensive, i.e., of order $\mathcal{O}(2^n)$. But as we go down the tree, and use smaller subsamples, the algorithm becomes faster and faster. In general, this is quite an efficient algorithm to implement.

The motivation of prediction trees is to emulate a decision tree. It also helps make sense of complicated regression scenarios where there are

lots of variable interactions over many variables, when it becomes difficult to interpret the meaning and importance of explanatory variables in a prediction scenario. By proceeding in a hierarchical manner on a tree, the decision analysis becomes transparent, and can also be used in practical settings to make decisions.

17.4.1 Classification Trees

To demonstrate this, let's use a data set that is already in R. We use the `kyphosis` data set which contains data on children who have had spinal surgery. The model we wish to fit is to predict whether a child has a post-operative deformity or not (variable: Kyphosis = {absent, present}). The variables we use are Age in months, number of vertebrae operated on (Number), and the beginning of the range of vertebrae operated on (Start). The package used is called `rpart` which stands for "recursive partitioning".

```
> library(rpart)
> data(kyphosis)
> head(kyphosis)
  Kyphosis Age Number Start
1   absent  71      3     5
2   absent 158      3    14
3  present 128      4     5
4   absent   2      5     1
5   absent   1      4    15
6   absent   1      2    16
> fit = rpart(Kyphosis~Age+Number+Start, method="class", data=kyphosis)
>
> printcp(fit)

Classification tree:
rpart(formula = Kyphosis ~ Age + Number + Start, data = kyphosis,
    method = "class")

Variables actually used in tree construction:
[1] Age   Start

Root node error: 17/81 = 0.20988

n= 81

        CP nsplit rel error xerror    xstd
1 0.176471      0   1.00000 1.0000 0.21559
2 0.019608      1   0.82353 1.1765 0.22829
3 0.010000      4   0.76471 1.1765 0.22829
```

We can now get a detailed summary of the analysis as follows:

```
> summary(fit)
Call:
rpart(formula = Kyphosis ~ Age + Number + Start, data = kyphosis,
    method = "class")
  n= 81
```

```
         CP nsplit rel error   xerror      xstd
1 0.17647059      0 1.0000000 1.000000 0.2155872
2 0.01960784      1 0.8235294 1.176471 0.2282908
3 0.01000000      4 0.7647059 1.176471 0.2282908

Node number 1: 81 observations,    complexity param=0.1764706
  predicted class=absent   expected loss=0.2098765
    class counts:    64    17
   probabilities: 0.790 0.210
  left son=2 (62 obs) right son=3 (19 obs)
  Primary splits:
      Start  < 8.5  to the right, improve=6.762330, (0 missing)
      Number < 5.5  to the left,  improve=2.866795, (0 missing)
      Age    < 39.5 to the left,  improve=2.250212, (0 missing)
  Surrogate splits:
      Number < 6.5  to the left,  agree=0.802, adj=0.158, (0 split)

Node number 2: 62 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.09677419
    class counts:    56     6
   probabilities: 0.903 0.097
  left son=4 (29 obs) right son=5 (33 obs)
  Primary splits:
      Start  < 14.5 to the right, improve=1.0205280, (0 missing)
      Age    < 55   to the left,  improve=0.6848635, (0 missing)
      Number < 4.5  to the left,  improve=0.2975332, (0 missing)
  Surrogate splits:
      Number < 3.5  to the left,  agree=0.645, adj=0.241, (0 split)
      Age    < 16   to the left,  agree=0.597, adj=0.138, (0 split)

Node number 3: 19 observations
  predicted class=present  expected loss=0.4210526
    class counts:     8    11
   probabilities: 0.421 0.579

Node number 4: 29 observations
  predicted class=absent   expected loss=0
    class counts:    29     0
   probabilities: 1.000 0.000

Node number 5: 33 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.1818182
    class counts:    27     6
   probabilities: 0.818 0.182
  left son=10 (12 obs) right son=11 (21 obs)
  Primary splits:
      Age    < 55   to the left,  improve=1.2467530, (0 missing)
      Start  < 12.5 to the right, improve=0.2887701, (0 missing)
      Number < 3.5  to the right, improve=0.1753247, (0 missing)
  Surrogate splits:
      Start  < 9.5  to the left,  agree=0.758, adj=0.333, (0 split)
      Number < 5.5  to the right, agree=0.697, adj=0.167, (0 split)

Node number 10: 12 observations
  predicted class=absent   expected loss=0
    class counts:    12     0
   probabilities: 1.000 0.000

Node number 11: 21 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.2857143
```

```
    class counts:    15     6
   probabilities: 0.714 0.286
  left son=22 (14 obs) right son=23 (7 obs)
  Primary splits:
      Age    < 111  to the right, improve=1.71428600, (0 missing)
      Start  < 12.5 to the right, improve=0.79365080, (0 missing)
      Number < 3.5  to the right, improve=0.07142857, (0 missing)

Node number 22: 14 observations
  predicted class=absent   expected loss=0.1428571
    class counts:    12     2
   probabilities: 0.857 0.143

Node number 23: 7 observations
  predicted class=present  expected loss=0.4285714
    class counts:     3     4
   probabilities: 0.429 0.571
```

We can plot the tree as well using the `plot` command. See Figure 17.7. The dendrogram like tree shows the allocation of the $n = 81$ cases to various branches of the tree.

```
> plot(fit, uniform=TRUE)
> text(fit, use.n=TRUE, all=TRUE, cex=0.8)
```

17.4.2 *The* `C4.5` *Classifier*

This is one of the top algorithms of data science. This classifier also follows recursive partitioning as in the previous case, but instead of minimizing the sum of squared errors between the sample data x and the true value p at each level, here the goal is to minimize entropy. This improves the information gain. Natural entropy (H) of the data x is defined as

$$H = -\sum_{x} f(x) \cdot ln\, f(x) \tag{17.1}$$

where $f(x)$ is the probability density of x. This is intuitive because after the optimal split in recursing down the tree, the distribution of x becomes narrower, lowering entropy. This measure is also often known as "differential entropy."

To see this let's do a quick example. We compute entropy for two distributions of varying spread (standard deviation).

```
dx = 0.001
x = seq(-5,5,dx)
H2 = -sum(dnorm(x,sd=2)*log(dnorm(x,sd=2))*dx)
print(H2)
```

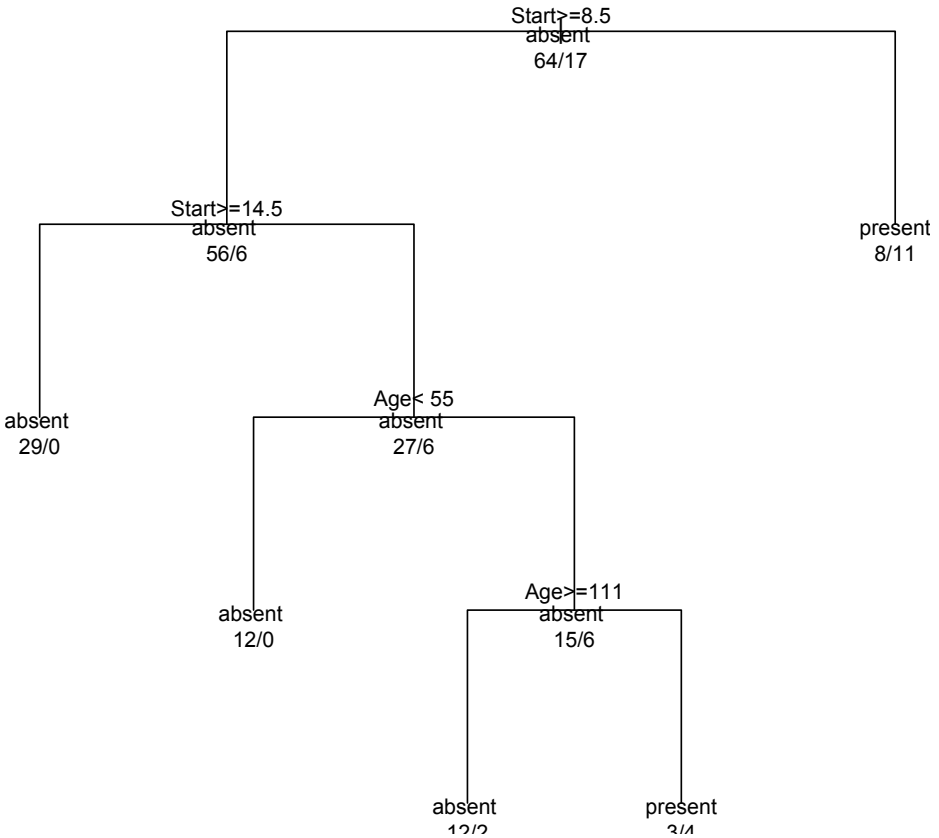


Figure 17.7: Classification tree for the kyphosis data set.

```
H3 = -sum(dnorm(x,sd=3)*log(dnorm(x,sd=3))*dx)
print(H3)

[1] 2.042076
[1] 2.111239
```

Therefore, we see that entropy increases as the normal distribution becomes wider. Now, let's use the C4.5 classifier on the `iris` data set. The classifier resides in the `RWeka` package.

```
library(RWeka)
data(iris)
print(head(iris))
res = J48(Species~.,data=iris)
print(res)
summary(res)
```

The output is as follows:

```
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
J48 pruned tree
------------------


Petal.Width <= 0.6: setosa (50.0)
Petal.Width > 0.6
|   Petal.Width <= 1.7
|   |   Petal.Length <= 4.9: versicolor (48.0/1.0)
|   |   Petal.Length > 4.9
|   |   |   Petal.Width <= 1.5: virginica (3.0)
|   |   |   Petal.Width > 1.5: versicolor (3.0/1.0)
|   Petal.Width > 1.7: virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9
```

```
=== Summary ===

Correctly Classified Instances         147               98       %
Incorrectly Classified Instances         3                2       %
Kappa statistic                          0.97
Mean absolute error                      0.0233
Root mean squared error                  0.108
Relative absolute error                  5.2482 %
Root relative squared error             22.9089 %
Coverage of cases (0.95 level)          98.6667 %
Mean rel. region size (0.95 level)      34       %
Total Number of Instances              150

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = setosa
  0 49  1 |  b = versicolor
  0  2 48 |  c = virginica
```

17.5 Regression Trees

We move from classification trees (discrete outcomes) to regression trees (scored or continuous outcomes). Again, we use an example that already exists in R, i.e., the cars dataset in the `cu.summary` data frame. Let's load it up.

```
> data(cu.summary)
> names(cu.summary)
[1] "Price"       "Country"     "Reliability" "Mileage"     "Type"
> head(cu.summary)
                Price Country Reliability Mileage  Type
Acura Integra 4 11950   Japan Much better      NA Small
Dodge Colt 4     6851   Japan        <NA>      NA Small
Dodge Omni 4     6995     USA  Much worse      NA Small
Eagle Summit 4   8895     USA      better      33 Small
Ford Escort 4    7402     USA       worse      33 Small
```

```
Ford Festiva 4   6319   Korea       better       37 Small
> dim(cu.summary)
[1] 117   5
```

We see that the variables are self-explanatory. See that in some cases, there are missing ($< NA >$) values in the Reliability variable. We will try and predict Mileage using the other variables. (Note: if we tried to predict Reliability, then we would be back in the realm of classification trees, here we are looking at regression trees.)

```
> library(rpart)
> fit <- rpart(Mileage~Price + Country + Reliability + Type,
   method="anova", data=cu.summary)
> summary(fit)
Call:
rpart(formula = Mileage ~ Price + Country + Reliability + Type,
    data = cu.summary, method = "anova")
  n=60 (57 observations deleted due to missingness)

          CP nsplit rel error    xerror       xstd
1 0.62288527      0 1.0000000 1.0322810 0.17522180
2 0.13206061      1 0.3771147 0.5305328 0.10329174
3 0.02544094      2 0.2450541 0.3790878 0.08392992
4 0.01160389      3 0.2196132 0.3738624 0.08489026
5 0.01000000      4 0.2080093 0.3985025 0.08895493

Node number 1: 60 observations,    complexity param=0.6228853
  mean=24.58333, MSE=22.57639
  left son=2 (48 obs) right son=3 (12 obs)
  Primary splits:
      Price       < 9446.5  to the right, improve=0.6228853, (0 missing)
      Type        splits as  LLLRLL,      improve=0.5044405, (0 missing)
      Reliability splits as  LLLRR,       improve=0.1263005, (11 missing)
      Country     splits as  --LRLRRRLL,  improve=0.1243525, (0 missing)
  Surrogate splits:
      Type    splits as  LLLRLL,     agree=0.950, adj=0.750, (0 split)
      Country splits as  --LLLRRLL, agree=0.833, adj=0.167, (0 split)

Node number 2: 48 observations,    complexity param=0.1320606
  mean=22.70833, MSE=8.498264
  left son=4 (23 obs) right son=5 (25 obs)
  Primary splits:
      Type        splits as  RLLRRL,      improve=0.43853830, (0 missing)
      Price       < 12154.5 to the right, improve=0.25748500, (0 missing)
      Country     splits as  --RRLRL-LL,  improve=0.13345700, (0 missing)
      Reliability splits as  LLLRR,       improve=0.01637086, (10 missing)
  Surrogate splits:
      Price   < 12215.5 to the right, agree=0.812, adj=0.609, (0 split)
      Country splits as  --RRLRL-RL,  agree=0.646, adj=0.261, (0 split)

Node number 3: 12 observations
  mean=32.08333, MSE=8.576389

Node number 4: 23 observations,    complexity param=0.02544094
  mean=20.69565, MSE=2.907372
  left son=8 (10 obs) right son=9 (13 obs)
  Primary splits:
      Type    splits as  -LR--L,      improve=0.515359600, (0 missing)
      Price   < 14962   to the left,  improve=0.131259400, (0 missing)
      Country splits as  ----L-R--R,  improve=0.007022107, (0 missing)
```

```
  Surrogate splits:
      Price < 13572   to the right, agree=0.609, adj=0.1, (0 split)

Node number 5: 25 observations,    complexity param=0.01160389
  mean=24.56, MSE=6.4864
  left son=10 (14 obs) right son=11 (11 obs)
  Primary splits:
      Price       < 11484.5 to the right, improve=0.09693168, (0 missing)
      Reliability splits as  LLRRR,       improve=0.07767167, (4 missing)
      Type        splits as  L—RR–,       improve=0.04209834, (0 missing)
      Country     splits as  —LRRR—LL,    improve=0.02201687, (0 missing)
  Surrogate splits:
      Country splits as  —LLLL—LR, agree=0.80, adj=0.545, (0 split)
      Type    splits as  L—RL–,    agree=0.64, adj=0.182, (0 split)

Node number 8: 10 observations
  mean=19.3, MSE=2.21

Node number 9: 13 observations
  mean=21.76923, MSE=0.7928994

Node number 10: 14 observations
  mean=23.85714, MSE=7.693878

Node number 11: 11 observations
  mean=25.45455, MSE=3.520661
```

We may then plot the results, as follows:

```
> plot(fit, uniform=TRUE)
> text(fit, use.n=TRUE, all=TRUE, cex=.8)
```

The result is shown in Figure 17.8.

17.5.1 *Example: Califonia Home Data*

This example is taken from a data set posted by Cosmo Shalizi at CMU. We use a different package here, called `tree`, though this has been subsumed in most of its functionality by `rpart` used earlier. The analysis is as follows:

```
> library(tree)
> cahomes = read.table("cahomedata.txt",header=TRUE)
> fit = tree(log(MedianHouseValue)~Longitude+Latitude, data=cahomes)
> plot(fit)
> text(fit,cex=0.8)
```

This predicts housing values from just latitude and longitude coordinates. The prediction tree is shown in Figure 17.9.

Further analysis goes as follows:

```
> price.deciles = quantile(cahomes$MedianHouseValue,0:10/10)
> cut.prices = cut(cahomes$MedianHouseValue,price.deciles,include.lowest=TRUE)
> plot(cahomes$Longitude, cahomes$Latitude,
   col=grey(10:2/11)[cut.prices],pch=20,xlab="Longitude",ylab="Latitude")
> partition.tree(fit,ordvars=c("Longitude","Latitude"),add=TRUE)
```

The plot of the output and the partitions is given in Figure 17.10.

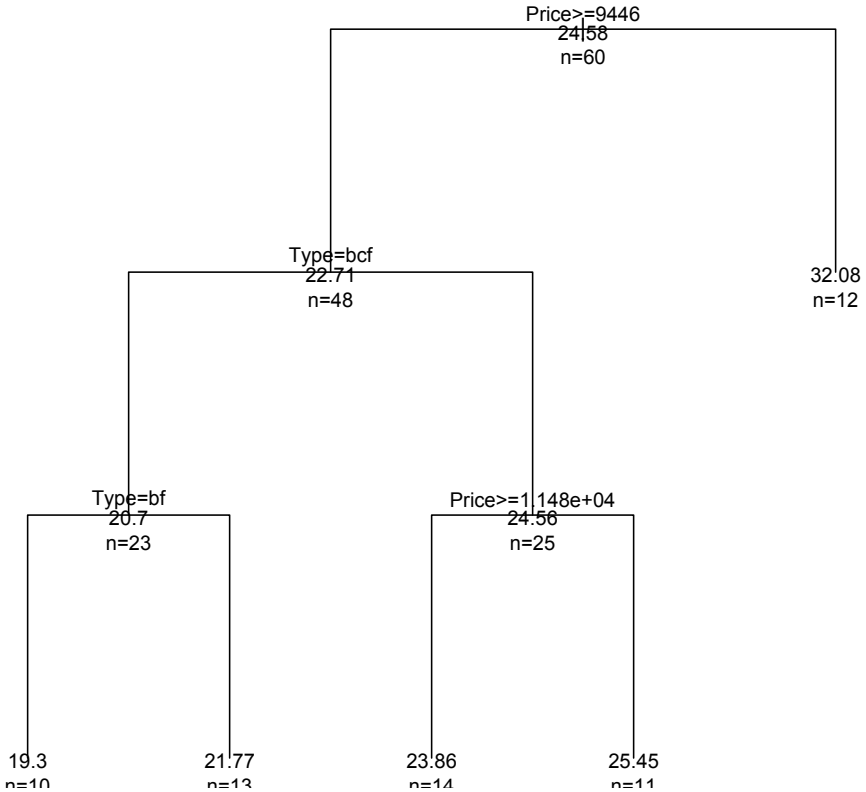


Figure 17.8: Prediction tree for cars mileage.

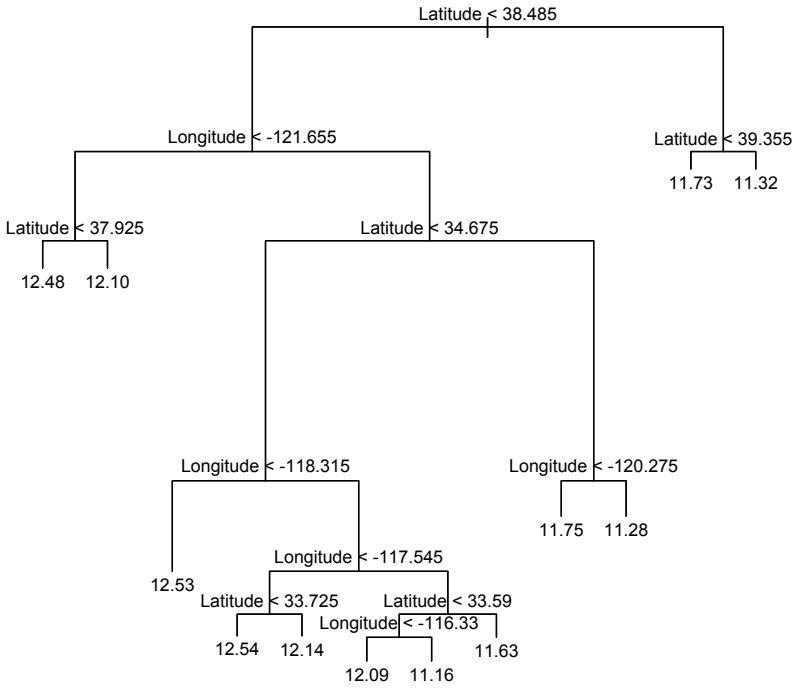


Figure 17.9: California home prices prediction tree.

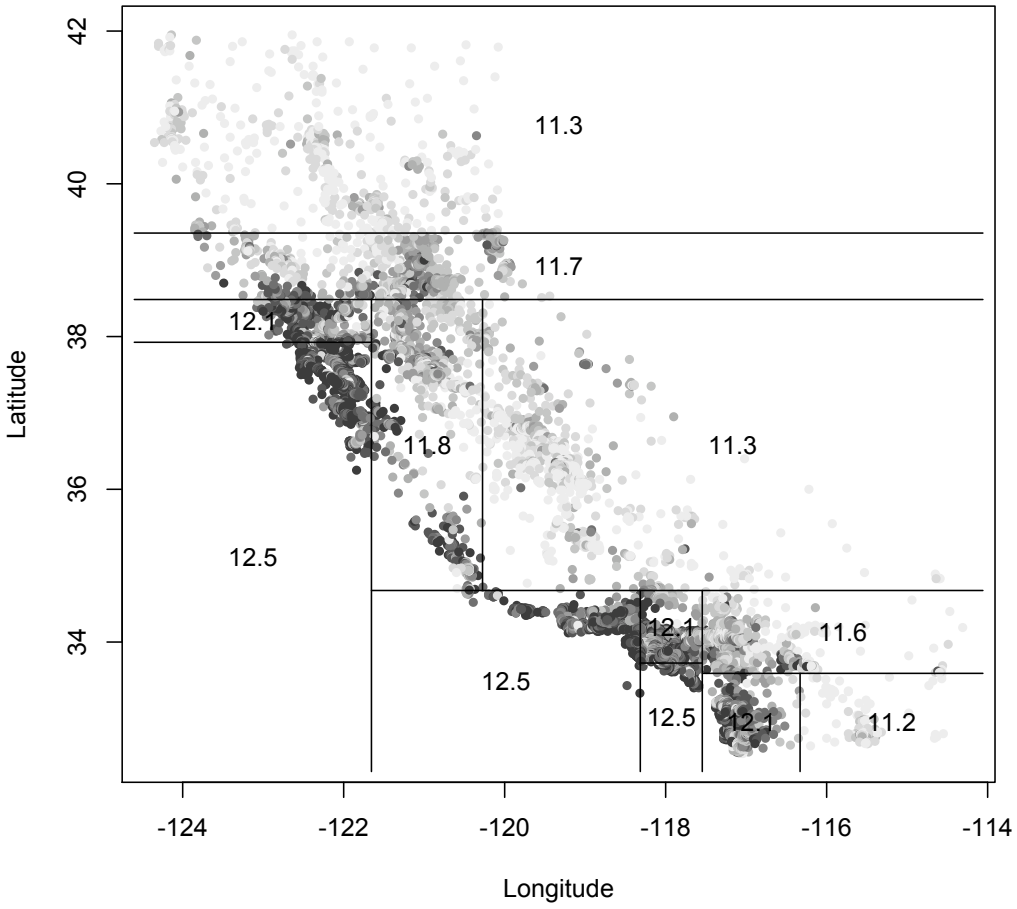
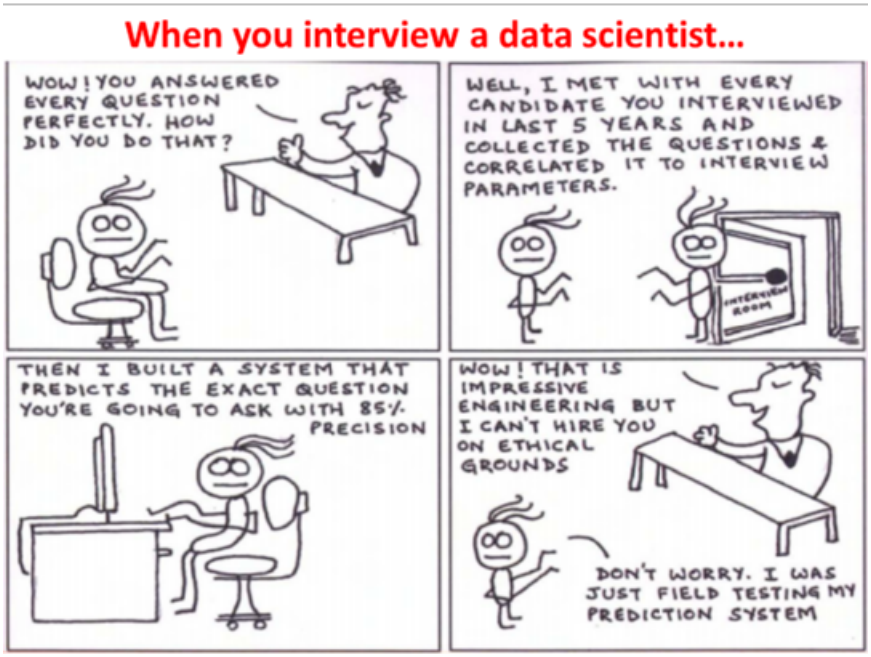


Figure 17.10: California home prices partition diagram.


When you interview a data scientist...
WOW! YOU ANSWERED EVERY QUESTION PERFECTLY. HOW DID YOU DO THAT?
WELL, I MET WITH EVERY CANDIDATE YOU INTERVIEWED IN LAST 5 YEARS AND COLLECTED THE QUESTIONS & CORRELATED IT TO INTERVIEW PARAMETERS.
INTERVIEW ROOM
THEN I BUILT A SYSTEM THAT PREDICTS THE EXACT QUESTION YOU'RE GOING TO ASK WITH 85% PRECISION
WOW! THAT IS IMPRESSIVE ENGINEERING BUT I CAN'T HIRE YOU ON ETHICAL GROUNDS
DON'T WORRY. I WAS JUST FIELD TESTING MY PREDICTION SYSTEM

18
Bibliography

A. Admati, and P. Pfleiderer (2001). "Noisytalk.com: Broadcasting Opinions in a Noisy Environment," working paper, Stanford University.

Aggarwal, Gagan., Ashish Goel, and Rajeev Motwani (2006). "Truthful Auctions for Price Searching Keywords," Working paper, Stanford University.

Andersen, Leif., Jakob Sidenius, and Susanta Basu (2003). All your hedges in one basket, *Risk*, November, 67-72.

W. Antweiler and M. Frank (2004). "Is all that Talk just Noise? The Information Content of Internet Stock Message Boards," *Journal of Finance*, v59(3), 1259-1295.

W. Antweiler and M. Frank (2005). "The Market Impact of Corporate News Stories," Working paper, University of British Columbia.

Artzner, A., F. Delbaen., J-M. Eber., D. Heath, (1999). "Coherent Measures of Risk," *Mathematical Finance* 9(3), 203–228.

Ashcraft, Adam., and Darrell Duffie (2007). "Systemic Illiquidity in the Federal Funds Market," *American Economic Review, Papers and Proceedings* 97, 221-225.

Barabasi, A.-L.; R. Albert (1999). "Emergence of scaling in random networks," *Science* 286 (5439), 509–512. arXiv:cond-mat/9910332. doi:10.1126/science.286.5439.509. PMID 10521342.

Barabasi, Albert-Laszlo., and Eric Bonabeau (2003). "Scale-Free Networks," *Scientific American* May, 50–59.

Bass, Frank. (1969). "A New Product Growth Model for Consumer Durables," *Management Science* 16, 215–227.

Bass, Frank., Trichy Krishnan, and Dipak Jain (1994). "Why the Bass Model Without Decision Variables," *Marketing Science* 13, 204–223.

Bengtsson, O., Hsu, D. (2010). How do venture capital partners match with startup founders? *Working Paper*.

Billio, M., Getmansky, M., Lo. A., Pelizzon, L. (2012). "Econometric Measures of Connectedness and Systemic Risk in the Finance and Insurance Sectors," *Journal of Financial Economics* 104(3), 536–559.

Billio, M., Getmansky, M., Gray, D., Lo. A., Merton, R., Pelizzon, L. (2012). "Sovereign, Bank and Insurance Credit Spreads: Connectedness and System Networks," Working paper, IMF.

Bishop, C. (1995). "Neural Networks for Pattern Recognition," Oxford University Press, New York.

Boatwright, Lee., and Wagner Kamakura (2003). "Bayesian Model for Prelaunch Sales Forecasting of Recorded Music," *Management Science* 49(2), 179–196.

P. Bonacich (1972). "Technique for analyzing overlappingmemberships," *Sociological Methodology* 4, 176-185.

P. Bonacich (1987). "Power and centrality: a family of measures," *American Journal of Sociology* 92(5), 1170-1182.

Bottazzi, L., Da Rin, M., Hellmann, T. (2011). The importance of trust for investment: Evidence from venture capital. *Working Paper*.

Brander, J. A., Amit, R., Antweiler, W. (2002). Venture-capital syndication: Improved venture selection vs. the value-added hypothesis, *Journal of Economics and Management Strategy*, v11, 423-452.

Browne, Sid., and Ward Whitt (1996). "Portfolio Choice and the Bayesian Kelly Criterion," *Advances in Applied Probability* 28(4), 1145–1176.

Burdick, D., Hernandez, M., Ho, H., Koutrika, G., Krishnamurthy, R., Popa, L., Stanoi, I.R., Vaithyanathan, S., Das, S.R. (2011). Extracting, linking and integrating data from public sources: A financial case study, *IEEE Data Engineering Bulletin*, 34(3), 60-67.

Cai, Y., and Sevilir, M. (2012). Board connections and M&A Transactions, *Journal of Financial Economics* 103(2), 327-349.

Cestone, Giacinta., Lerner, Josh, White, Lucy (2006). The design of communicates in venture capital, *Harvard Business School Working Paper*.

Chakrabarti, S., B. Dom, R. Agrawal, and P. Raghavan. (1998). "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies," *The VLDB Journal*, Springer-Verlag.

Chidambaran, N. K., Kedia, S., Prabhala, N.R. (2010). CEO-Director connections and fraud, *University of Maryland Working Paper*.

Cochrane, John (2005). The risk and return of venture capital. *Journal of Financial Economics* 75, 3-52.

Cohen, Lauren, Frazzini, Andrea, Malloy, Christopher (2008a). The small world of investing: Board connections and mutual fund returns, *Journal of Political Economy* 116, 951–979.

Cohen, Lauren, Frazzini, Andrea, Malloy, Christopher (2008b). Sell-Side school ties, forthcoming, *Journal of Finance*.

M. Coleman and T. L. Liau. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology* 60, 283–284.

Cormen, Thomas., Charles Leiserson, and Ronald Rivest (2009). Introduction to Algorithms, MIT Press, Cambridge, Massachusetts.

Cornelli, F., Yosha, O. (2003). Stage financing and the role of convertible securities, *Review of Economic Studies* 70, 1-32.

Da Rin, Marco, Hellmann, Thomas, Puri, Manju (2012). A survey of venture capital research, *Duke University Working Paper*.

Das, Sanjiv., (2014). "Text and Context: Language Analytics for Finance," *Foundations and Trends in Finance* v8(3), 145-260.

Das, Sanjiv., (2014). "Matrix Math: Network-Based Systemic Risk Scoring," forthcoming *Journal of Alternative Investments*.

Das, Sanjiv., Murali Jagannathan, and Atulya Sarin (2003). Private Equity Returns: An Empirical Examination of the Exit of asset-Backed Companies, *Journal of Investment Management* 1(1), 152-177.

Das, Sanjiv., and Jacob Sisk (2005). "Financial Communities," *Journal of Portfolio Management* 31(4), 112-123.

S. Das and M. Chen (2007). "Yahoo for Amazon! Sentiment Extraction from Small Talk on the Web," *Management Science* 53, 1375-1388.

S. Das, A. Martinez-Jerez, and P. Tufano (2005). "eInformation: A Clinical Study of Investor Discussion and Sentiment," *Financial Management* 34(5), 103-137.

S. Das and J. Sisk (2005). "Financial Communities," *Journal of Portfolio Management* 31(4), 112-123.

Das, Sanjiv., and Rangarajan Sundaram (1996). "Auction Theory: A Summary with Applications and Evidence from the Treasury Markets," *Financial Markets, Institutions and Instruments* v5(5), 1–36.

Das, Sanjiv., Jo, Hoje, Kim, Yongtae (2011). Polishing diamonds in the rough: The sources of communicated venture performance, *Journal of Financial Intermediation*, 20(2), 199-230.

Dean, Jeffrey., and Sanjay Ghemaway (2004). "MapReduce: Simplified Data Processing on Large Clusters," OSDI'04: Sixth Symposium on Operating System Design and Implementation.

P. DeMarzo, D. Vayanos, and J. Zwiebel (2003). "Persuasion Bias, Social Influence, and Uni-Dimensional Opinions," *Quarterly Journal of Economics* 118, 909-968.

Du, Qianqian (2011). Birds of a feather or celebrating differences? The formation and impact of venture capital syndication, *University of British Columbia Working Paper*.

J. Edwards., K. McCurley, and J. Tomlin (2001). "An Adaptive Model for Optimizing Performance of an Incremental Web Crawler," Proceedings WWW10, Hong Kong, 106-113.

G. Ellison, and D. Fudenberg (1995). "Word of Mouth Communication and Social Learning," *Quarterly Journal of Economics* 110, 93-126.

Engelberg, Joseph., Gao, Pengjie, Parsons, Christopher (2000). The value of a rolodex: CEO pay and personal networks, Working Paper, University of North Carolina at Chapel Hill.

Fortunato, S. (2009). Community detection in graphs, *arXiv*:0906.0612v1 [physics.soc-ph].

S. Fortunato (2010). "Community Detection in Graphs," *Physics Reports* 486, 75-174.

Gertler, M.S. (1995). Being there: proximity, organization and culture in the development and adoption of advanced manufacturing technologies, *Economic Geography* 7(1), 1-26.

Ghiassi, M., H. Saidane, and D. Zimbra (2005). "A dynamic artificial neural network model for forecasting time series events," *International Journal of Forecasting* 21, 341–362.

Ginsburg, Jeremy., Matthew Mohebbi, Rajan Patel, Lynnette Brammer, Mark Smolinski, and Larry Brilliant (2009). "Detecting Influenza Epidemics using Search Engine Data," *Nature* 457, 1012–1014.

Girvan, M., Newman, M. (2002). Community structure in social and biological networks, *Proc. of the National Academy of Science* 99(12), 7821–7826.

Glaeser, E., ed., (2010). Agglomeration Economics, University of Chicago Press.

D. Godes, D. Mayzlin, Y. Chen, S. Das, C. Dellarocas, B. Pfeieffer, B. Libai, S. Sen, M. Shi, and P. Verlegh. "The Firm's Management of Social Interactions," *Marketing Letters* v16, 415-428.

Godes, David., and Dina Mayzlin (2004). "Using Online Conversations to Study Word of Mouth Communication" *Marketing Science* 23(4), 545–560.

Godes, David., and Dina Mayzlin (2009). "Firm-Created Word-of-Mouth Communication: Evidence from a Field Test", *Marketing Science* 28(4), 721–739.

Goldfarb, Brent, Kirsch, David, Miller, David, (2007). Was there too little entry in the dot com era?, *Journal of Financial Economics* 86(1), 100-144.

Gompers, P., Lerner, J. (2000). Money chasing deals? The impact of fund inflows on private equity valuations, *Journal of Financial Economics* 55(2), 281-325.

Gompers, P., Lerner, J. (2001). The venture capital revolution, *Journal of Economic Perspectives* 15(2), 45-62.

Gompers, P., Lerner, J., (2004). The Venture Capital Cycle, MIT Press.

Gorman, M., Sahlman, W. (1989). What do venture capitalists do? *Journal of Business Venturing* 4, 231-248.

P. Graham (2004). "Hackers and Painters," O'Reilly Media, Sebastopol, CA.

Granger, Clive, (1969). "Investigating Causal Relations by Econometric Models and Cross-spectral Methods". *Econometrica* 37(3), 424–438.

Granovetter, M. (1985). Economic action and social structure: The problem of embeddedness, *American Journal of Sociology* 91(3), 481-510.

Greene, William (2011). Econometric Analysis, 7th edition, Prentice-Hall.

Grossman, S., Hart, O. (1986). The costs and benefits of ownership: A theory of vertical and lateral integration, *Journal of Political Economy* 94(4), 691-719.

Guimera, R., Amaral, L.A.N. (2005). Functional cartography of complex metabolic networks, *Nature* 433, 895-900.

Guimera, R., Mossa, S., Turtschi, A., Amaral, L.A.N. (2005). The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles, *Proceedings of the National Academy of Science* 102, 7794-7799.

Guiso, L., Sapienza, P., Zingales, L. (2004). The role of social capital in financial development, *American Economic Review* 94, 526-556.

R. Gunning. The Technique of Clear Writing. *McGraw-Hill*, 1952.

Halevy, Alon., Peter Norvig, and Fernando Pereira (2009). "The Unreasonable Effectiveness of Data," *IEEE Intelligent Systems* March-April, 8–12.

Harrison, D., Klein, K. (2007). What's the difference? Diversity constructs as separation, variety, or disparity in organization, *Academy of Management Review* 32(4), 1199-1228.

Hart, O., Moore, J. (1990). Property rights and the nature of the firm, *Journal of Political Economy* 98(6), 1119-1158.

Hegde, D., Tumlinson, J. (2011). Can birds of a feather fly together? Evidence for the economic payoffs of ethnic homophily, *Working Paper*.

Hellmann, T. J., Lindsey, L., Puri, M. (2008). Building relationships early: Banks in venture capital, *Review of Financial Studies* 21(2), 513-541.

Hellmann, T. J., Puri, M. (2002). Venture capital and the professionalization of start-up firms: Empirical evidence, *Journal of Finance* 57, 169-197.

Hoberg, G., Phillips, G. (2010). Product Market Synergies and Competition in Mergers and Acquisitions: A Text-Based Analysis, *Review of Financial Studies* 23(10), 3773–3811.

Hochberg, Y., Ljungqvist, A., Lu, Y. (2007). Whom You Know Matters: Venture Capital Networks and Investment Performance, *Journal of Finance* 62(1), 251-301.

Hochberg, Y., Lindsey, L., Westerfield, M. (2011). Inter-firm Economic Ties: Evidence from Venture Capital, *Northwestern University Working Paper*.

Huchinson, J., Andrew Lo, and T. Poggio (1994). "A Non Parametric Approach to Pricing and Hedging Securities via Learning Networks," *Journal of Finance* 49(3), 851–889.

Hwang, B., Kim, S. (2009). It pays to have friends, *Journal of Financial Economics* 93, 138-158.

Ishii, J.L., Xuan, Y. (2009). Acquirer-Target social ties and merger outcomes, *Working Paper*, SSRN: http://ssrn.com/abstract=1361106.

T. Joachims (1999). "Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning," B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press.

Kanniainen, Vesa., and Christian Keuschnigg (2003). The optimal portfolio of start-up firms in asset capital finance, *Journal of Corporate Finance* 9(5), 521-534.

Kaplan, S. N., Schoar, A. (2005). Private equity performance: Returns, persistence and capital flows, *Journal of Finance* 60, 1791-1823.

Kaplan, S. N., Sensoy, B., Stromberg, P. (2002). How well do venture capital databases reflect actual investments?, *Working paper*, University of Chicago.

Kaplan, S. N., Stromberg, P. (2003). Financial contracting theory meets the real world: Evidence from venture capital contracts, *Review of Economic Studies* 70, 281-316.

Kaplan, S. N., Stromberg, P. (2004). Characteristics, contracts and actions: Evidence from venture capital analyses, *Journal of Finance* 59, 2177-2210.

Kelly, J.L. (1956). "A New Interpretation of Information Rate," *The Bell System Technical Journal* 35, 917–926.

Koller, D., and M. Sahami (1997). "Hierarchically Classifying Documents using Very Few Words," *International Conference on Machine Learning*, v14, Morgan-Kaufmann, San Mateo, California.

D. Koller (2009). "Probabilistic Graphical Models," MIT Press.

Krishnan, C. N. V., Masulis, R. W. (2011). Venture capital reputation, in Douglas J. Cummings, ed., *Handbook on Entrepreneurial Finance, Venture Capital and Private Equity*, Oxford University Press.

Lavinio, Stefano (2000). "The Hedge Fund Handbook," Irwin Library of Investment & Finance, McGraw-Hill..

D. Leinweber., and J. Sisk (2010). "Relating News Analytics to Stock Returns," mimeo, Leinweber & Co.

Lerner, J. (1994). The syndication of venture capital investments. *Financial Management* 23, 1627.

Lerner, J. (1995). Venture capitalists and the oversight of private firms, *Journal of Finance* 50 (1), 302-318

Leskovec, J., Kang, K.J., Mahoney, M.W. (2010). Empirical comparison of algorithms for network community detection, *ACM WWW International Conference on World Wide Web*.

S. Levy (2010). "How Google's Algorithm Rules the Web," *Wired*, March.

F. Li (2006). "Do Stock Market Investors Understand the RiskSentiment of Corporate Annual Reports?" Working paper, University of Michigan.

Lindsey, L. A. (2008). Blurring boundaries: The role of venture capital in strategic alliances, *Journal of Finance* 63(3), 1137-1168.

Lossen, Ulrich (2006). The Performance of Private Equity Funds: Does Diversification Matter?, Discussion Papers 192, SFB/TR 15, University of Munich.

T. Loughran and W. McDonald, (2014). Measuring readability in financial disclosures, *The Journal of Finance* 69, 1643–1671.

Loukides, Mike (2012). "What is Data Science?" *O'Reilly*, Sebastopol, CA.

Lusseau, D. (2003). The emergent properties of a dolphin social network, *Proceedings of the Royal Society of London B* 271 S6: 477–481.

Mayer-Schönberger, Viktor., and Kenneth Cukier (2013). "Big Data: A Revolution that will Transform How We Live, Work, and Think," *Houghton Mifflin Harcourt*, New York.

A. McCallum (1996). "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering," `http://www.cs.cmu.edu/~mccallum/bow`.

McPherson, M., Smith-Lovin, L., Cook, J. (2001). Birds of a feather: Homophily in social networks, *Annual Review of Sociology* 27, 415-444.

Mezrich, Ben (2003). "Bringing Down the House: The Inside Story of Six MIT Students Who Took Vegas for Millions," Free Press,

Mitchell, Tom (1997). "Machine Learning," McGraw-Hill.

L. Mitra., G. Mitra., and D. diBartolomeo (2008). "Equity Portfolio Risk (Volatility) Estimation using Market Information and Sentiment," Working paper, Brunel University.

P. Morville (2005). "Ambient Findability," O'Reilly Press, Sebastopol, CA.

Neal, R.(1996). "Bayesian Learning for Neural-Networks," *Lecture Notes in Statistics*, v118, Springer-Verlag.

Neher, D. V. (1999). Staged financing: An agency perspective, *Review of Economic Studies* 66, 255-274.

Newman, M. (2001). Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality, *Physical Review E* 64, 016132.

Newman, M. (2006). Modularity and community structure in networks, *Proc. of the National Academy of Science* 103 (23), 8577-8582.

Newman, M. (2010). Networks: An introduction, Oxford University Press.

B. Pang., L. Lee., and S. Vaithyanathan (2002). "Thumbs Up? Sentiment Classification using Machine Learning Techniques," *Proc. Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Patil, D.J. (2012). "Data Jujitsu," *O'Reilly*, Sebastopol, CA.

Patil, D.J. (2011). "Building Data Science Teams," *O'Reilly*, Sebastopol, CA.

P. Pons, M. Latapy (2006). "Computing Communities in Large Networks Using Random Walks," *Journal of Graph Algorithms Applied*, 10(2), 191-218.

M. Porter, (1980). "An Algorithm for Suffix Stripping," *Program* 14(3), 130?137.

Porter, M.E. (2000). Location, competition and economic development: Local clusters in a global economy, *Economic Development Quarterly* 14(1), 15-34.

Porter, Mason., Mucha, Peter, Newman, Mark, Friend, A. J. (2007). Community structure in the United States House of Representatives, *Physica A: Statistical Mechanics and its Applications* 386(1), 413–438.

Ravasz, E., Somera, A.L., Mongru, D.A., Oltvai, N., Barabasi, A.L. (2002). Hierarchical organization of modularity in metabolic networks, *Science* 297 (5586), 1551.

Robinson, D. (2008). Strategic alliances and the boundaries of the firm, *Review of Financial Studies* 21(2), 649-681.

Robinson, D., Sensoy, B. (2011). Private equity in the 21st century: cash flows, performance, and contract terms from 1984-2010, *Working Paper*, Ohio State University.

Robinson, D., Stuart, T. (2007). Financial contracting in biotech strategic alliances, *Journal of Law and Economics* 50(3), 559-596.

Seigel, Eric (2013). "Predictive Analytics," *John-Wiley & Sons*, New Jersey.

Segaran, T (2007). "Programming Collective Intelligence," O'Reilly Media Inc., California.

Shannon, Claude (1948). "A Mathematical Theory of Communication," *The Bell System Technical Journal* 27, 379–423.

Simon, Herbert (1962). The architecture of complexity, *Proceedings of the American Philosophical Society* 106 (6), 467–482.

Smola, A.J., and Scholkopf, B (1998). "A Tutorial on Support Vector Regression," NeuroCOLT2 Technical Report, ESPIRIT Working Group in Neural and Computational Learning II.

Sorensen, Morten (2007). How smart is smart money? A Two-sided matching model of venture capital, *Journal of Finance* 62, 2725-2762.

Sorensen, Morten (2008). Learning by investing: evidence from venture capital, *Columbia University Working Paper*.

Tarjan, Robert, E. (1983), "Data Structures and Network Algorithmsï£¡ *CBMS-NSF Regional Conference Series in Applied Mathematics*.

P. Tetlock (2007). "Giving Content to Investor Sentiment: The Role of Media in the Stock Market," *Journal of Finance* 62(3), 1139-1168.

P. Tetlock, P. M. Saar-Tsechansky, and S. Macskassay (2008). "More than Words: Quantifying Language to Measure Firm's Fundamentals," *Journal of Finance* 63(3), 1437-1467.

Thorp, Ed. (1962). "Beat the Dealer," Random House, New York.

Thorp, Ed (1997). "The Kelly Criterion in Blackjack, Sports Betting, and the Stock Market," *Proc. of The 10th International Conference on Gambling and Risk Taking*, Montreal, June.

Vapnik, V, and A. Lerner (1963). "Pattern Recognition using Generalized Portrait Method," *Automation and Remote Control*, v24.

Vapnik, V. and Chervonenkis (1964). "On the Uniform Convergence of Relative Frequencies of Events to their Probabilities," *Theory of Probability and its Applications*, v16(2), 264-280.

Vapnik, V (1995). The Nature of Statistical Learning Theory, Springer-Verlag, New York.

Vickrey, William (1961). "Counterspeculation, Auctions, and Competitive Sealed Tenders," *Journal of Finance* 16(1), 8–37.

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand and Dan Steinberg, (2008). "Top 10 Algorithms in Data Mining," *Knowledge and Information Systems* 14(1), 1-37.

Wu, K., Taki, Y., Sato, K., Kinomura, S., Goto, R., Okada, K., Kawashima, R., He, Y., Evans, A. C. and Fukuda, H. (2011). Age-related changes in topological organization of structural brain networks in healthy individuals, *Human Brain Mapping* 32, doi: 10.1002/hbm.21232.

Zachary, Wayne (1977). An information flow model for conflict and fission in small groups, *Journal of Anthropological Research* 33, 452–473.