**Analysis by Simulation and Graphics – some examples**

**This expository talk is intended to demonstrate the utility of simulation and graphics for four selected situations. The first illustrates the power of smoothing, the second the validity of the bootstrap, the third the power of creative use of simulation and graphics to amplify the utility of meagre data, and the fourth to reveal a surprising property of sports league point spreads.**

**The applications discussed are respectively, gasoline consumption, Body Mass Index, bakery deliveries, and the league table for Premier League football. The overall message in these examples is that there exist simple yet non-standard techniques, using simulation and graphics, that could be included in undergraduate statistics courses.**

Here is a bit more info about what I would do with these examples. I have R programs for everything …

Gas & smoothing demo – I just show how smoothing reveals the seasonal pattern not otherwise discernible.

SEM-bootstrap - I show that simulation can generate the square root law via simulation

Bimbo – I show how a few brave assumptions can produce valuable information from scanty data

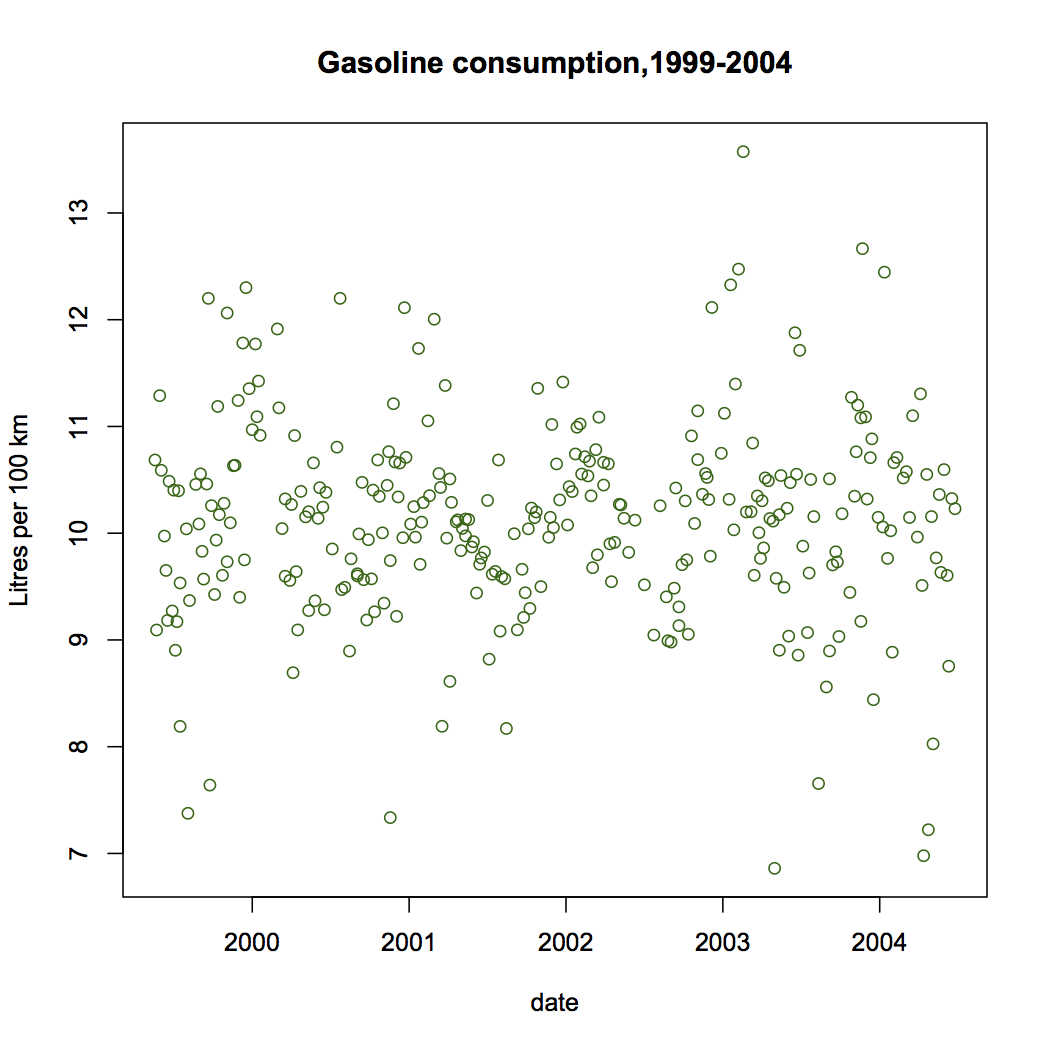
Sports Leagues – I show that the point spread at the end of a Premier League season does not support the widespread assumption that the top teams are actually better than the bottom teams (with some exceptions).

**The Power of Smoothing**

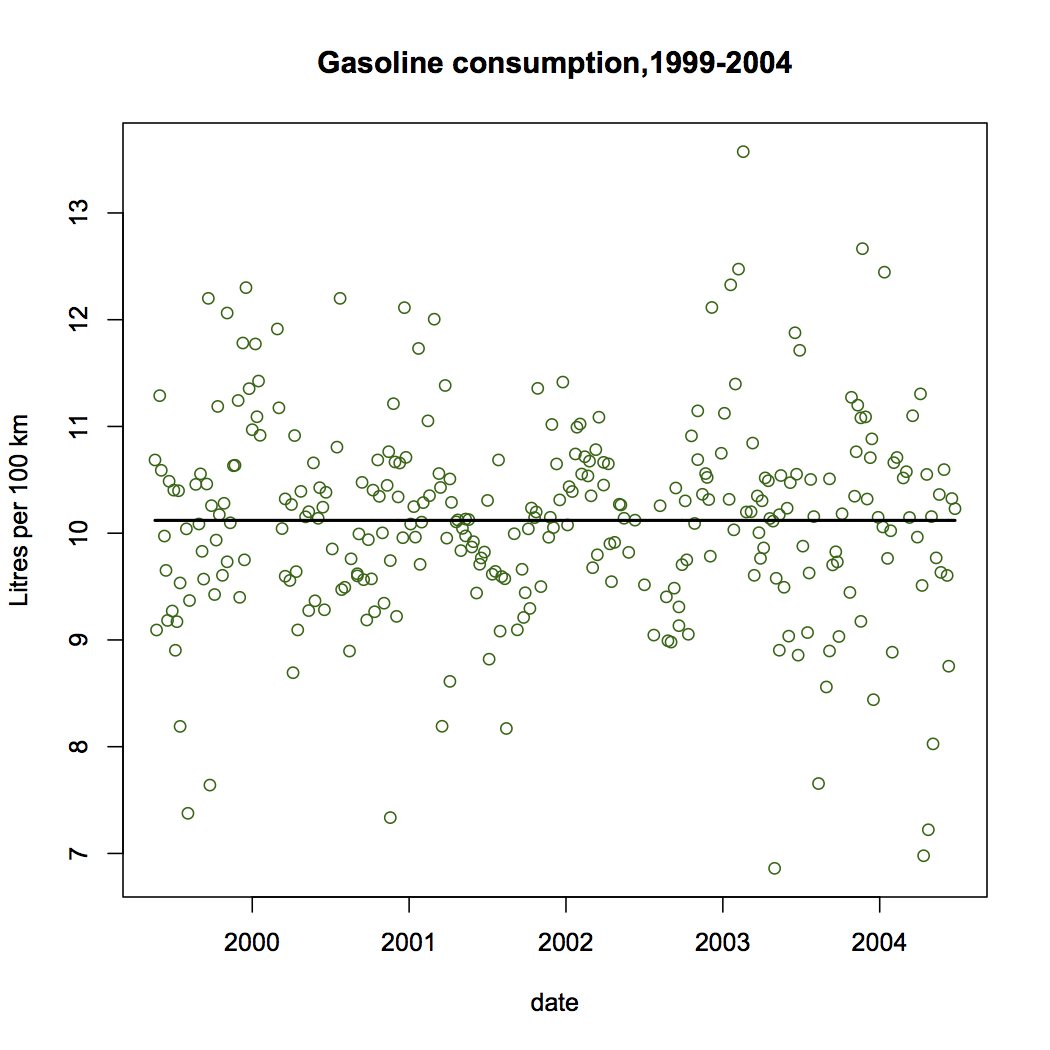
Real statisticians like data, so maybe that is why I collected my gasoline consumption for five years. This involved writing down, and each gasoline refill, the kilometers travelled since the last refill, and the kilometers travelled since the last refill. Here is the data displayed as a time series:

<use gas()



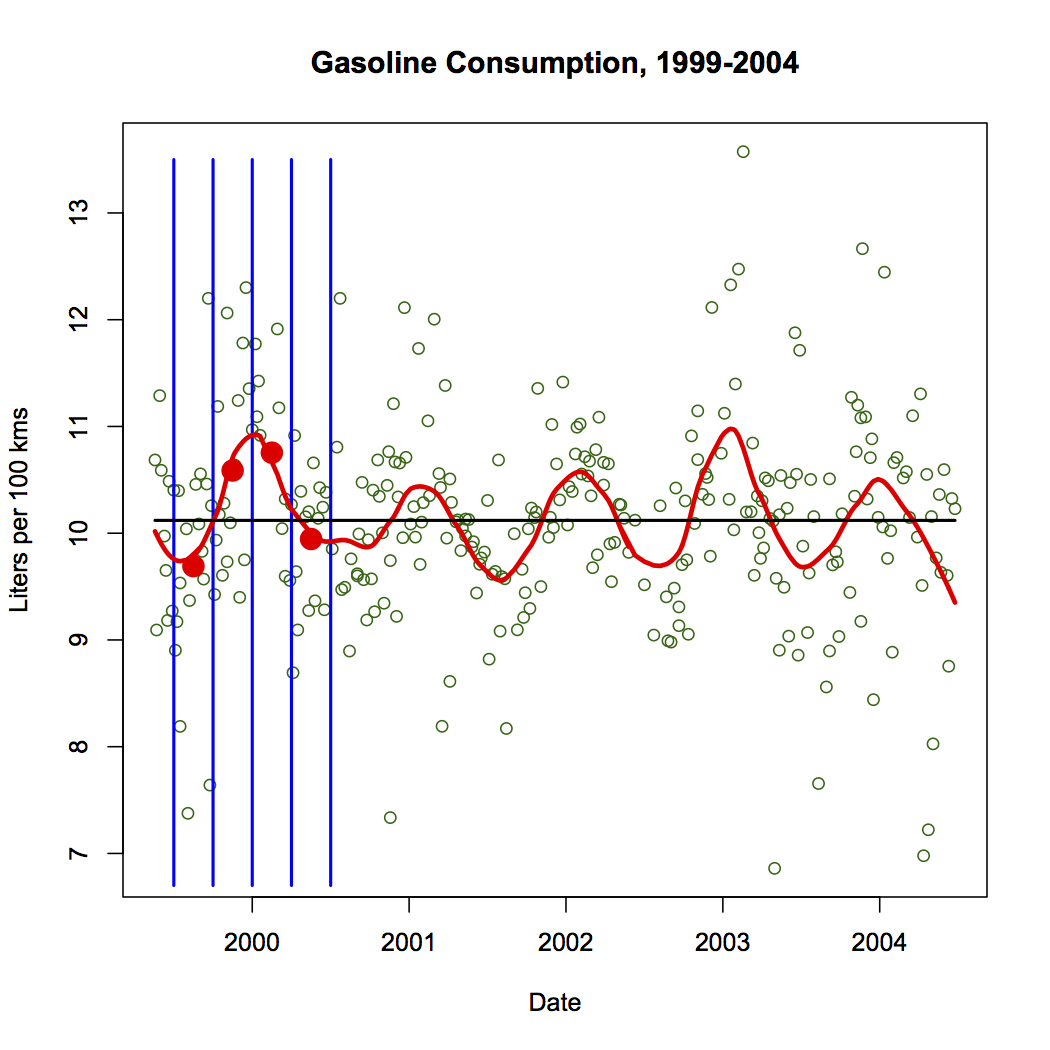


There does not seem to be much trend up or down over time. In fact the regression line would look like this:



You might conclude that the effort to collect this data resulted in a pretty boring result! But note that there is quite a bit of variability in the data, and we would like to explain at least some of that. Actually, the driving situation did not change much over time: I was commuting from Tsawwassen to and from SFU each weekday, and not much other driving with this car. So the program to discover what affects gas consumption is likely to be a dead end. The variability of the gas consumption rate is surprising given the constancy of the driving regime. Perhaps the problem is measurment error. If so, lets do a bit of averaging:

<use gasplot()>



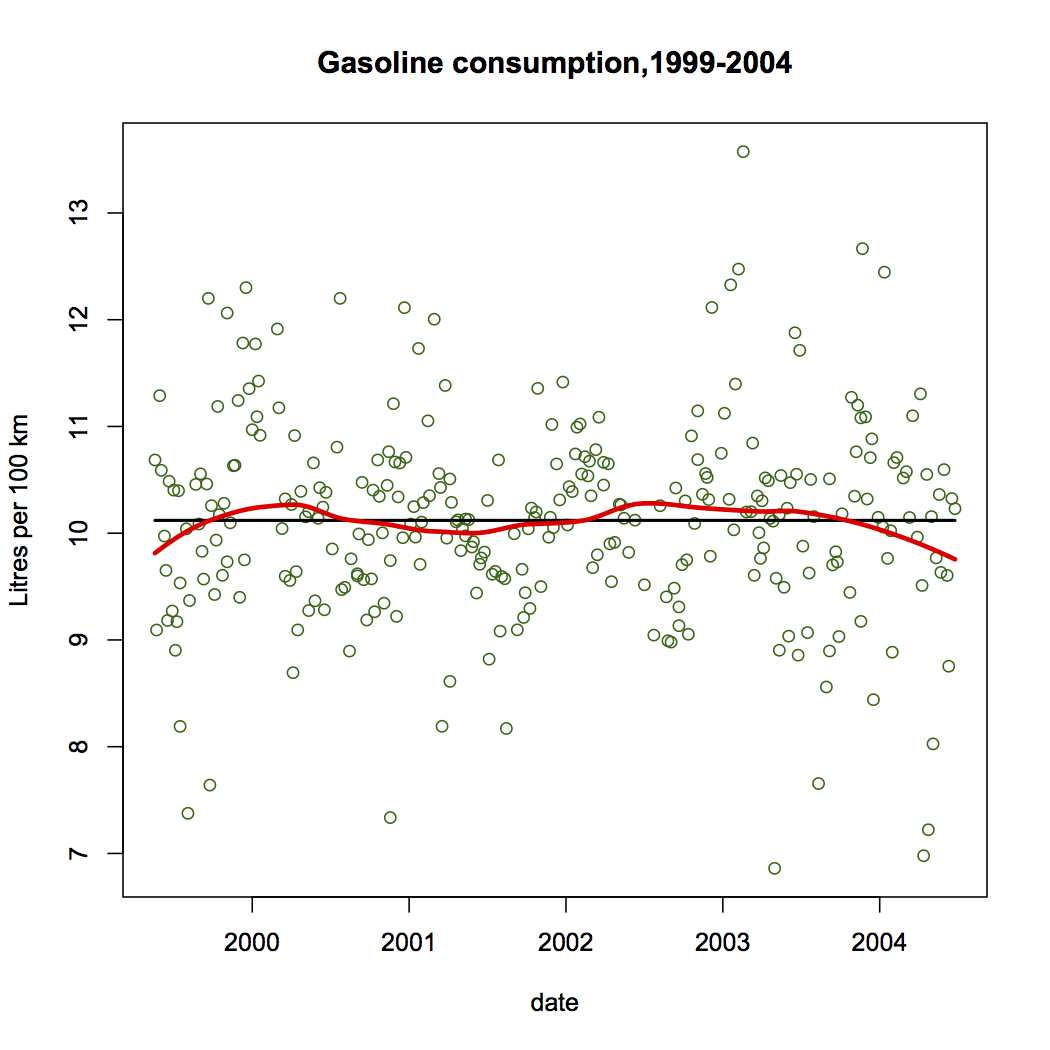
Averaging in vertical strips is the simplest lind of smoothing, and one can see here that it produces a seasonal pattern. Note that the thickness of the strip determines the amount of smoothing applied. Too narrow strips would produce a chaotic pattern like the data itself, while too thick strips would produce an almost flat curve.

<use gas.v.run>

This smooth is not enough:



while this one is over-smoothed:



The criteria that determines the correct amount of smoothing is the believability of features displayed, in the light of general knowledge about the data measurements. That good statistics practice involves subjectivity is a fact that applied statisticians know is true.

There are more sophisticated methods of smoothing, but when simple is good enough, it is better because it is easy to explain to the audience that is interested in the result.

**Confirmation of the Utility of the Bootstrap**

The bootstrap is a method for estimating variability of a statistic based on a single sample (of n data points selected at random from a population). If the statistic is t(x) where x is an n-tuple of data, then the variability of t(x) is estimated by computing the variability of t(x\*), where x\* is a sample selected with replacment from x. Clearly we can sample as many x\* as we need, and as we include more and more x\* in our estimate, the variability estimate will usually stabilize to a single value.

There is one situation where we know what the variability should be, and so let’s see if the bootstrap gives us a good estimate. For a random sample from any population, the variability of a sample mean of n observations from the population will be the boot , where 𝜎 is the population SD. In practice we do not usually know 𝜎 but we estimate it from s, and use s / √n as our estimate of the SD of the sample mean.

Our experiment here is to try to get an estimate close to 𝜎 / √n using only the data in one single sample of n observations, using the bootstrap technique.

use boot.demo.slow()

We start with a population – of course this would not usually be known, but for our experiment, it is known so that we can check the outcome.

Next we select a random sample – and note the sample mean and sample SD.

Often when we take a sample we want to use it to estimate the population mean, but when we do this we also like to know how accurate this estimate of the mean is. The sample SD measure the spread of the sample but what we want is the accuracy of the sample mean as an estimate of the population mean. Now we have theory that tells us to use s / √n to indicate the accuracy of the sample mean but we are going to pretend we do not know this theory, and instead use the bootstrap on this one sample to inform us about the accuracy of the sample mean. Talking re-samples from our original sample, with replacment, produces samples that have many of the same values as were in the original sample, but with varying multiplicity. The result gives an estimate very close to s / √n, which should be surprising!

The following graph shows how this procedure works over several sample sizes, with one experiment at each sample size. Note the bootstrap works as well as s / √n.

use boot.SEM.demo()

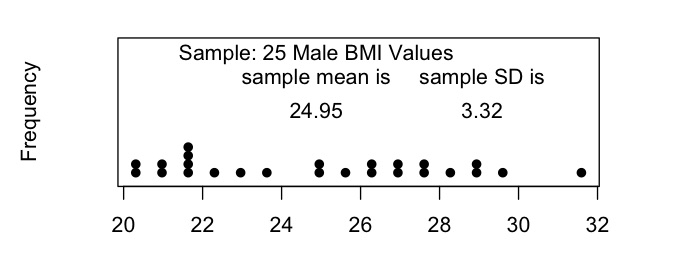
Is there a point here, other than an explanation of the bootstrap? Note that the bootstrap procedure sidestepped the requirement of know the theoretical formula for estimating the variability of a statistic, and suggests that it might work in more complicated situations in which the theoretical approach is not known.

Why does the bootstrap work? The Probability Integral transformation provides a general method for sasmpling from a population defined by the CDF. If we use the ECDF as a proxy for the CDF in this, “random sampling” becomes Bootstrap sampling. So any statistic value computed from a bootstrap sample will approximate the value computed from the actual population sampled. Since the ECDF is the main information we have about the CDF, its use as an estimate is natural and avoids further assumptions – but note the important assumption of “random” sampling,

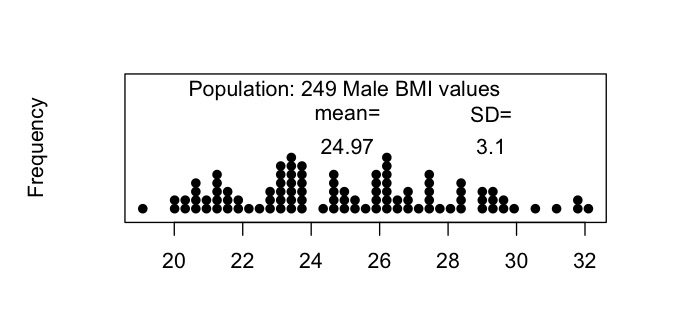
Let’s take a look at a data-based application.

Here is a sample of 25 BMI values for men.

BMI is (weight in kgms)/(height in meters)2 It tries to measure obesity.



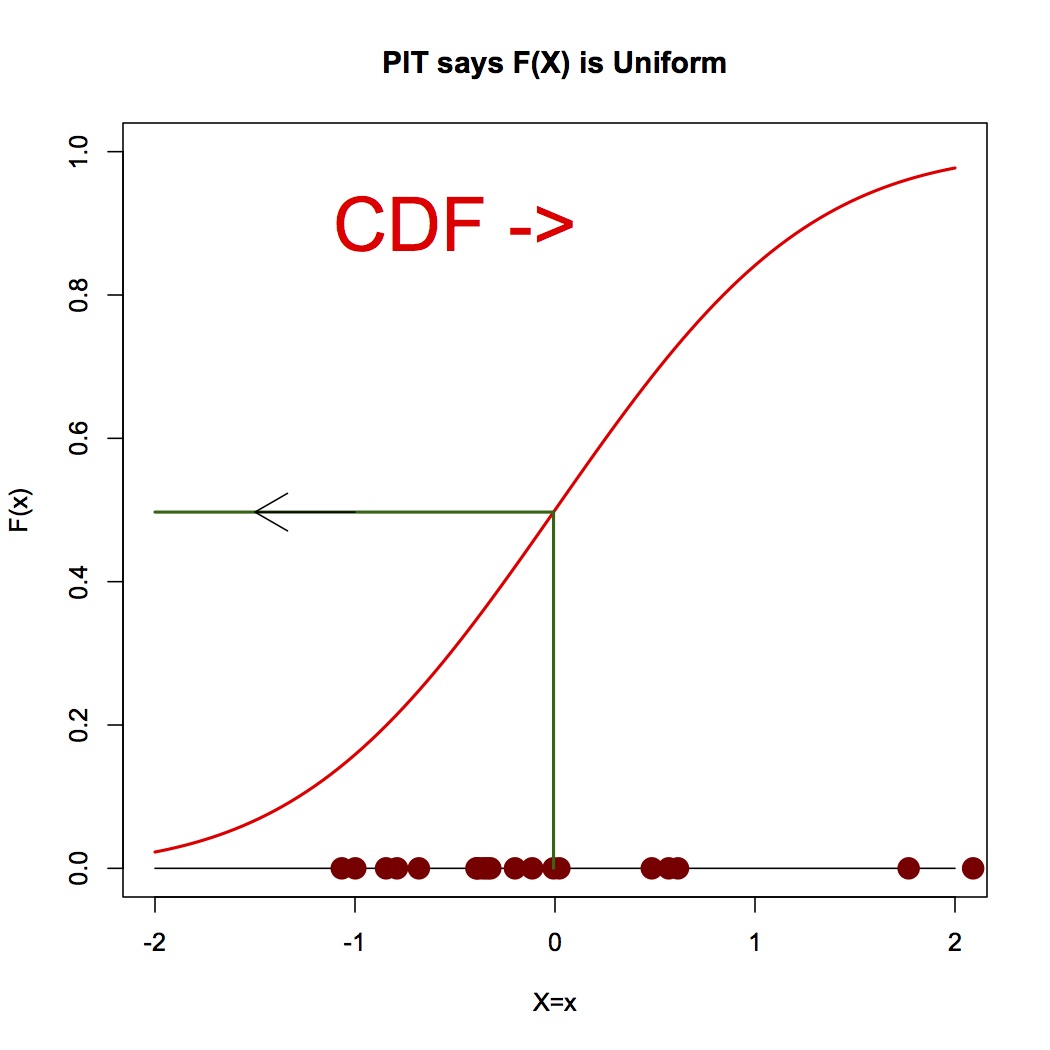
Suppose we want to know how well we have estimated the population mean BMI for the population underlying this random sample. We need to estimate the SEM of the population mean and we now have two ways to do that. The square root formula gives 0.66 and the bootstrap method gives .60 for the SEM of the estimated mean. However, in this experiement we do actually know the true SEM based on the population data:

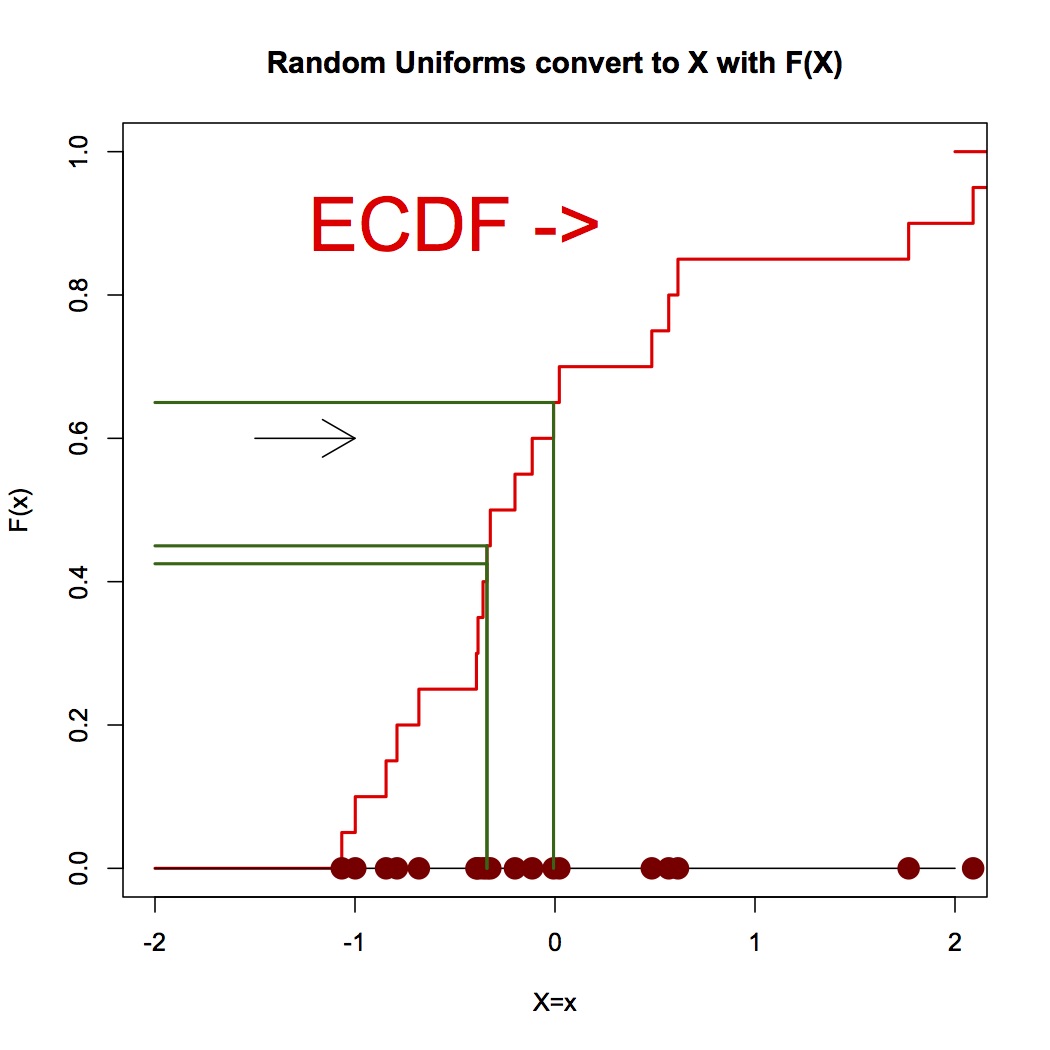


so the population SEM is 3.1/√25 = 0.62. So both estimates (.66 and .60) were very good in this case.

At this point, you may ask, why does the bootstrap work at all?

The following two graphs can be used to explain it. Essentially, the bootstrap is what you would use to generate a sample from a distribution whose true cdf was the ecdf of the sample used.

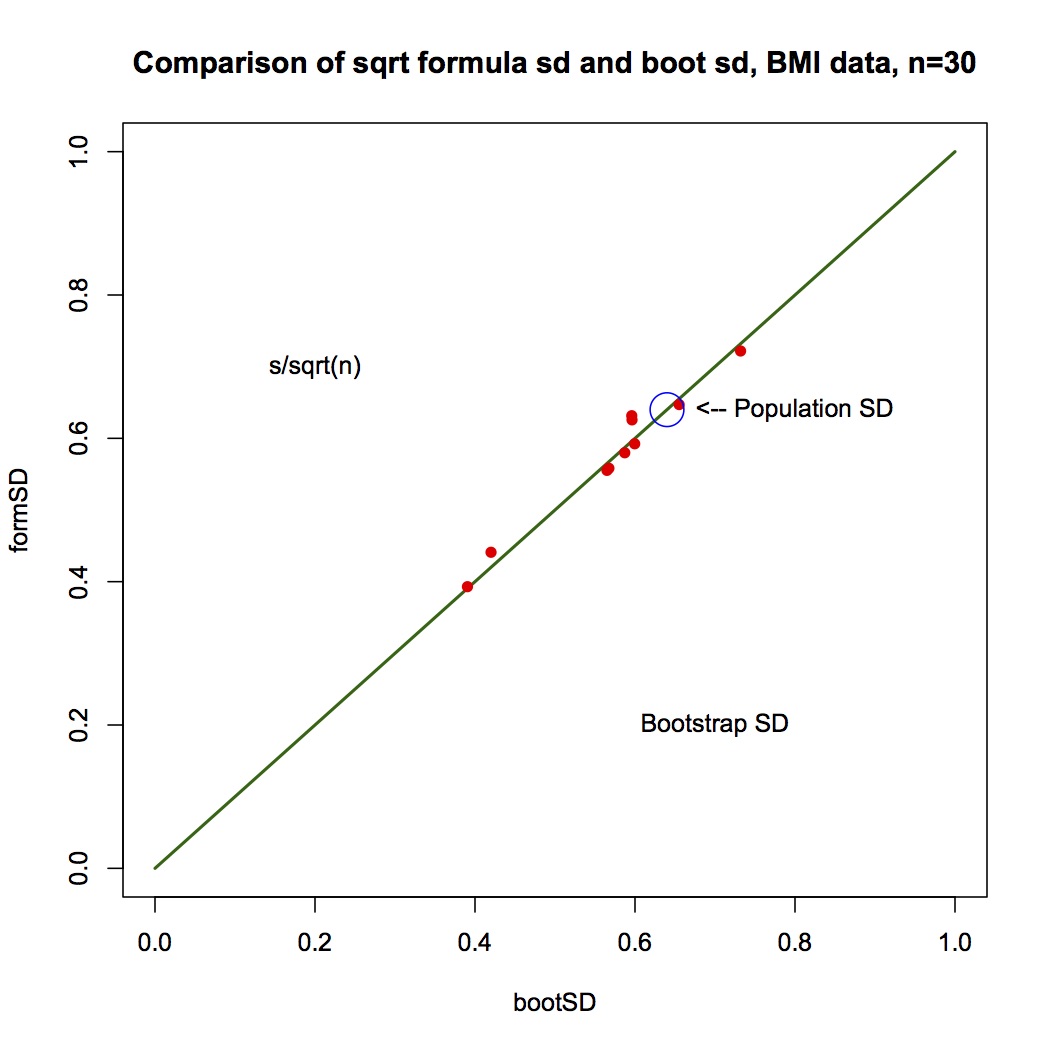




Without making further assumptions about the shape of the distribution, using the empirical cumultive distribution function is easy to justify.

Now we move on to use the bootstrap in a more complicated situation where the distribution SD of the statistic is not known.

Here is one situation where you may not have an easy formula for the SD of an estimate: Consider a sample of size n=30, from the BMI population, and a statistic such as the 4rth largest value (essentially the 90th percentile). The true sd of the estimated percentile is about 0.63. The estimates using bootstrap on samples of size 30 produce values like (in 10 separate experiments) those shown in the graph below:



Not bad for a sample of only 30 values – and, no further theory required!

The bootstrap has an important role of providing the precision of estimates in complex or non-standard applications.

**Exploration of Simulation and Graphics for Operations Research**

(The case of the Bimbo Bakery)

Bimbo Group is an global bakery company based in Mexico. I was once sent some data on a tiny piece of the bakery operations, and asked if there were a way to use it to improve operations. The data concerned one retail outlet, one product (a loaf of bread), the deliveries of that product to the outlet each day for a year, and the sales of that product that day. So the main question to address with this data is whether the delivery amounts could be improved: some days the deliveries are too small to meet demand, and some days the deliveries exceed the demand and there is leftover waste. Of course, crucial to this analysis is the cost of unfulfilled demand, and the cost of leftovers. We may not know this but presumably someone from the Bimbo company could provide these costs. Let’s proceed with some guesses for these costs and see whether there is some advice about deliveries that would flow from our analysis. Lets suppose each loaf sold results in a gross profit of $0.50, each leftover results in a loss of $0.25, and each unfulfilled demand results in a loss of $2.00. The idea behind this last parameter is that unfulfilled demand could result in a loss of future sales, perhaps with a small probability.

The data consists of two numbers for each day of the year, except Sundays. The two numbers are: the number of leftovers loaves - L

the number of loaves sold – S

From this we can deduce the number of loaves delivered – D

We also need a symbol for unfulfilled demand – U. Obviously, U is not observed, and this is where we need to use some creative techniques to judge its influence on the outlet’s daily profit.

There is a fairly obvious weekly pattern, and so we might simplify the problem by concentrating on one day-of-the-week, say Wednesday. So now we have 53 daily delivery quantities and 53 daily sales quantities. Often the two numbers will be equal indicating that there may have been some unfulfilled demand – in this case the leftovers L = 0. When they are unequal, the deliveries D > the sales. On any particular day, the gross profit is S\*0.50 –L\*0.25 – U\*2.00. Each day, either L=0 or U=0.

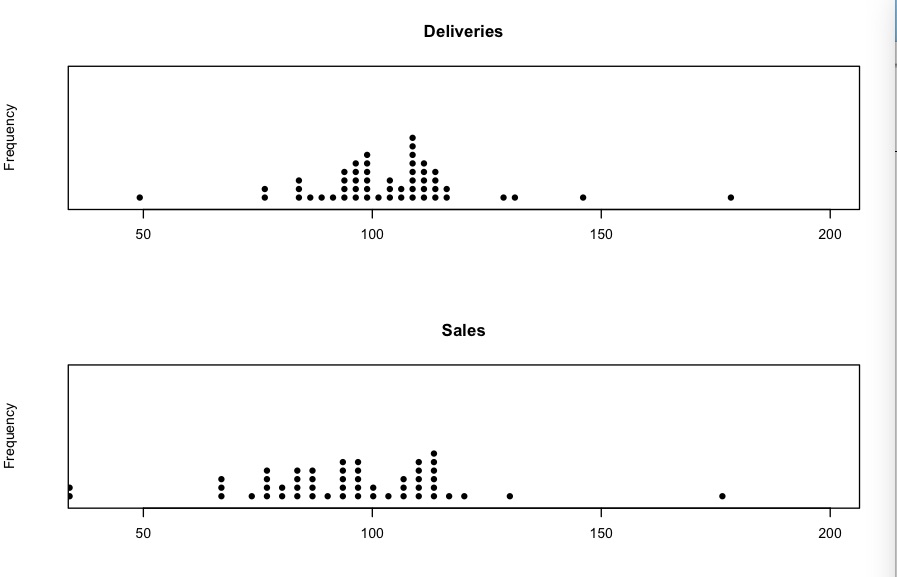
If we could estimate the daily demand distribution, then we could run through a year of observed deliveries and by comparing simulated demand with actual deliveries, we could simulate an annual sales distribution. In fact, if we propose a certain distribution as a demand distribution, we can check our proposal by comparing the simulated sales distribution with the actual sales distribution. We will use this method to estimate the actual demand distribution for a particular day of the week – eg. Wednesday.

Lets first look at the Wednesday data:

(see next page)

It is clear that Wednesday deliveries vary considerably over the year, and that demand is also erratic (when it has a chance to show it). Lets look at the distributions of deliveries and demand:

Use bimbo.guess.plots(115,35,xd=wd,xs=ws)



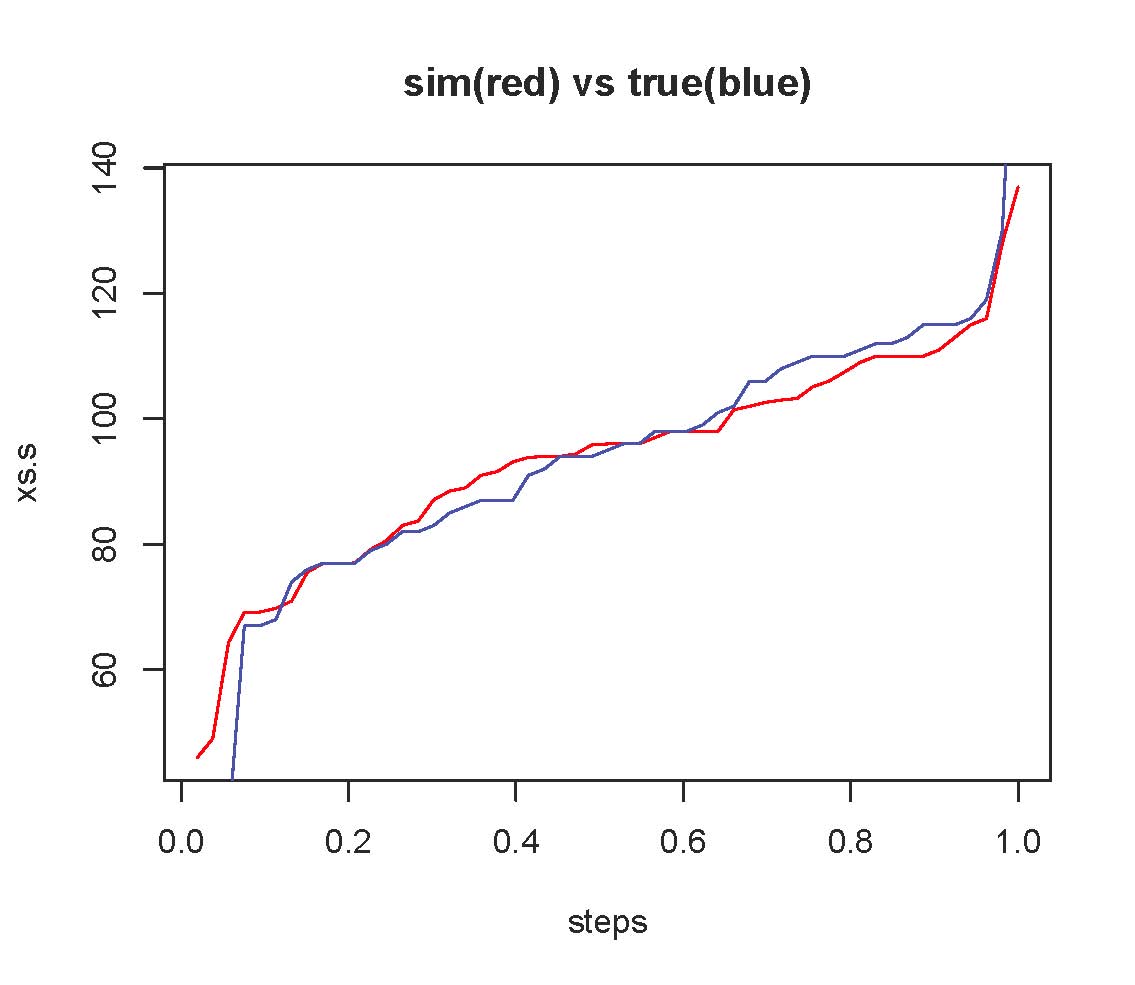
The dotplots suggest that the upper part of the demand distribution is hidden because the sales cannot exceed the deliveries. Our next step is to guess the demand distribution – we can check it by 1. Computing the simulated sales using the simulated demand and the actual deliveries

2. Comparing the simulated sales distribution with the actual sales distribution

<use bimbo.test(m,s,ws,wd) with m=115 and s=35

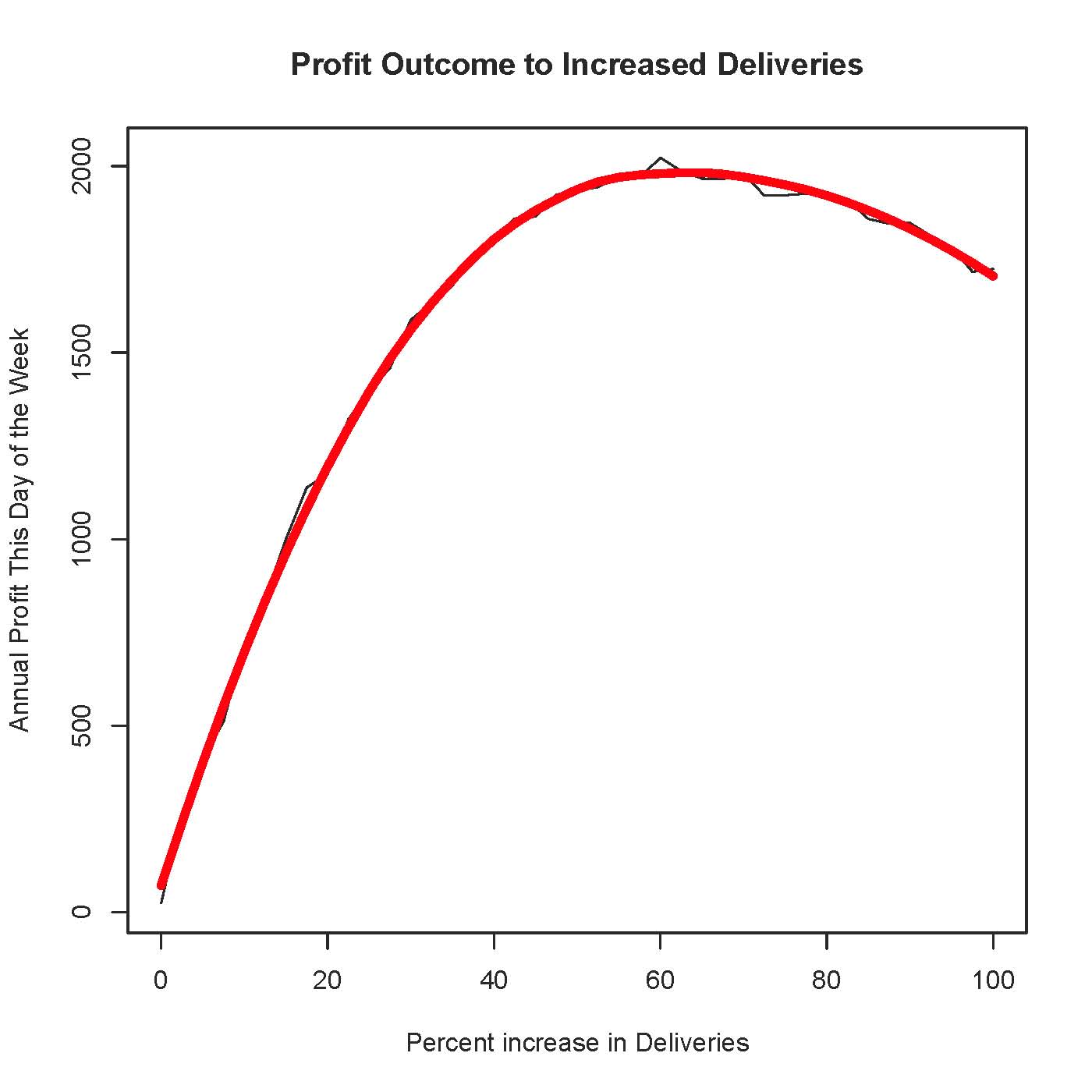
The result by trial and error is that the demand distribution is well-approximated by a N(mean=115, sd = 35) distribution. Here is a typical plot of the ecdfs of the two distributions:

(But try a few bimbo.test(115,35,ws,wd))



So we have found the hidden demand distribution. Now we consider what this implies for advice to the Bimbo delivery plan. Presumably there is someone deciding on the delivery amounts, and local conditions must inform this. But we can see if the delivery amounts tend to be to large or too small by seeing how the profit would change if the delivery amounts were altered by a constant percentage over the year.

Use bimbo.profit.graph(115,35,wd,smooth=F) then with smooth =T



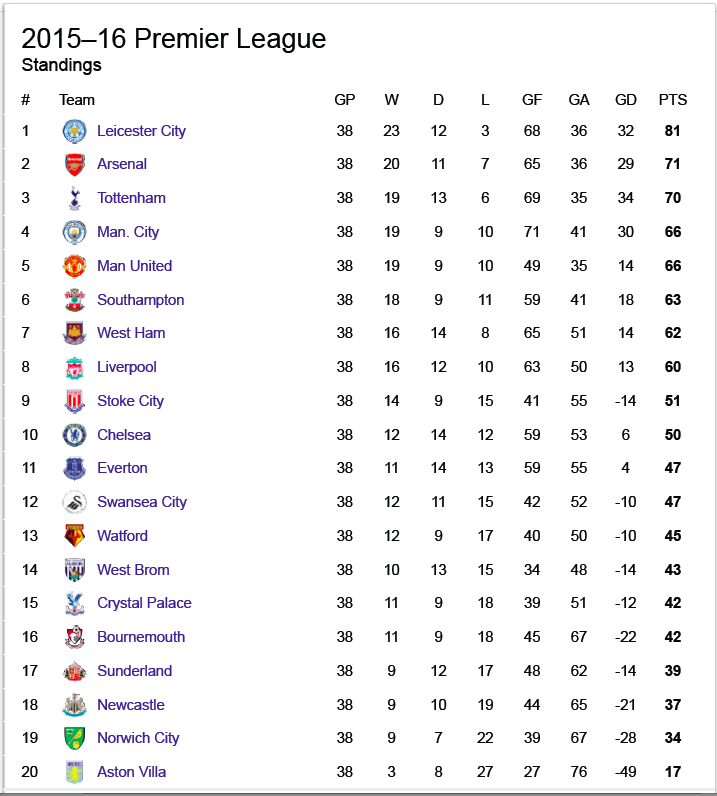
It looks like a 60% increase in delivery amounts will maximize profits, and furthermore, the increase in profit from the previous regime is almost $2000 per annum. (the value of the graph at 0 increase is very small).

Presumably this procedure, including the trial and error fitting of the demand distribution, could be automated, and perfomed on every product at every retail location, and for every day of the week!

**A Surprising Feature of Sports Leagues**

The public generally under-appreciates the effects of random variation. The early development of mathematical statistics was partly motivated by the desire to show that many apparently interesting research results were merely the result of chance or “luck”, and were not reproducible results. However, the existence of the analytical methods to separate luck from reproducible results has had a limited influence among public activites – especially in sport.

Consider the famous Premier League of football (or soccer as we might prefer). Last years league results are shown here:



Many fans believe implicitly in the relationship between the league points and the quality of play realized by the various teams. The particular outcome of the 2015-16 season was in the news even more than usual since Leicester City had ended the previous season in 14th position the previous year. Fans would remember the previous year’s results – see below – in which Leicester City ranked 14th.



From one year to the next, there is no draft system aimed at equalizing the teams. Each team is run as a business. So for a team’s performance to change drastically from one year to the next is unexpected.

This experience suggests the question: How much of the annual league outcome is the result of random events rather than actual changes in the quality of team play?

One way to approach this question is to suppose that all teams are actually the same quality – you could interpret this as the same probability to score a goal in any time interval. Most fans would consider this unrealistic in the extreme, but let’s explore the implications anyway, and consider fixes later. The aim of this assumption is the explore the effect of this assumption on the league point list at the end of a 38 game season.

We might have to consider the effect of HOME vs AWAY, and we need to allow for TIE games, but in essence we could imagine each game to be decided by a random mechanism such as the flip of a coin. What would the league points look like if there were no real differences among the teams?

By running a simulation of the 38 games with 50-50 probabilities, it turns out that the league points typically vary from about 37 to about 68, and do not cluster around 52 as one might expect. (0 for a loss, 1 for a tie, 3 for a win, averages 1.375 times 38 is 52.25). Here is a simulation of 10 seasons of this hypothetical league:

Use lgz.round(ha=0,m=10)

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 69 66 66 63 71 64 66 69 70 69

[2,] 63 64 62 62 66 63 63 67 67 68

[3,] 62 62 61 59 65 62 60 64 67 63

[4,] 59 61 59 59 62 61 58 63 61 62

[5,] 57 60 57 58 61 61 57 62 60 59

[6,] 57 59 56 57 60 58 57 59 59 58

[7,] 56 58 55 57 60 58 56 56 55 55

[8,] 56 58 54 55 59 57 55 56 54 54

[9,] 54 52 53 54 56 55 54 55 54 54

[10,] 52 52 52 52 53 53 53 53 53 51

[11,] 50 48 52 51 52 50 53 51 51 51

[12,] 50 48 51 50 52 50 52 50 50 50

[13,] 49 48 51 50 50 50 51 50 49 50

[14,] 47 47 51 49 48 49 47 46 49 49

[15,] 47 47 50 48 47 48 47 43 47 45

[16,] 47 44 47 46 41 47 47 41 46 45

[17,] 46 44 44 45 41 45 45 39 45 44

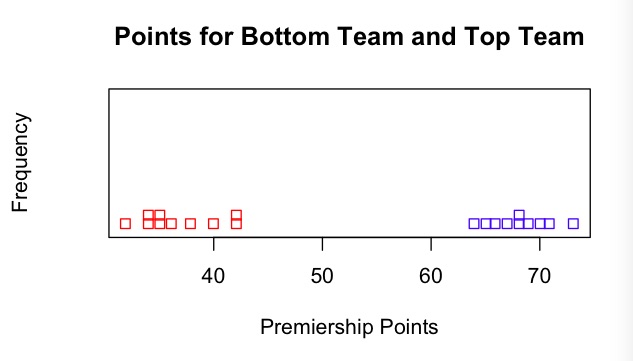
[18,] 44 42 43 43 40 45 43 38 43 44

[19,] 39 42 42 41 39 42 41 38 41 43

[20,] 33 37 31 39 32 33 33 36 38 40

Now compare the actual top and bottom team points from 2015-16:

Use lgz.round(ha=0)



And now note the range of values 17 for Aston Villa and 81 for Leicester City. Clearly these two teams are not average. However, the simulation suggests that the rest of the league does not have a convincing departure from the range of values that would be expected if all games were 50-50.

At this point someone with an historical knowledge of this Premier League would point out that there is a long history of four superior teams: Manchester United, Chelsea, Arsenal and Liverpool. Moreover, huge financial boost has recently been given to Manchester City and Tottenham. Moreover, it is widely known that there is a home team advantage. If we redo the simulation with a two tier league – meaning that 6 teams are 50-50 with each other but 60-40 with the poorer teams, and poorer teams are 50-50 with each other but 40-60 with the better teams – then the result looks like this:

Use lgz.round(ha=.1,nsup=6,hi=1.5,m=10)

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 71 63 69 63 73 83 78 72 70 61

[2,] 71 62 67 63 64 64 72 72 70 61

[3,] 67 61 61 62 61 61 62 66 66 60

[4,] 64 60 59 62 61 59 59 62 62 59

[5,] 62 58 57 61 60 58 59 61 59 58

[6,] 56 58 54 60 58 58 59 56 55 58

[7,] 54 57 54 56 55 57 56 54 54 58

[8,] 54 57 53 55 52 53 54 54 53 58

[9,] 51 53 52 55 51 53 53 51 53 55

[10,] 51 52 51 54 51 52 52 51 53 53

[11,] 51 52 51 52 51 52 51 50 52 52

[12,] 48 50 50 51 50 51 49 47 50 50

[13,] 48 48 50 51 50 49 47 46 48 50

[14,] 47 46 49 48 50 49 47 46 47 49

[15,] 46 46 49 48 49 48 46 45 47 46

[16,] 44 46 49 47 47 45 44 43 45 46

[17,] 43 45 47 47 46 39 42 41 44 45

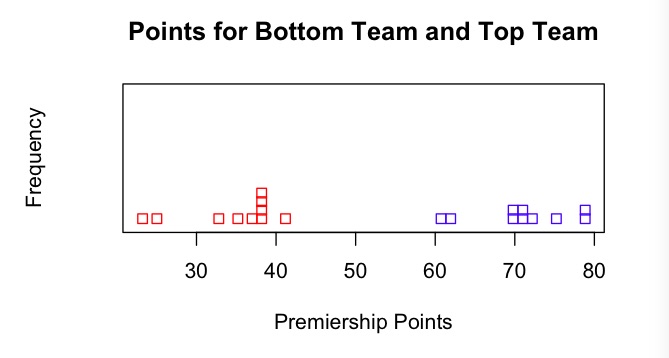
[18,] 43 42 47 47 44 39 41 39 42 43

[19,] 38 42 44 46 41 39 36 38 42 42

[20,] 35 39 38 32 37 29 35 37 32 39

and focussing on the highest and lowest:

use lgz.round.hilo(ha=.1,nsup=6,hi=1.5,m=10)



To demonstrate superior quality, a team would have to have much more than 73 league points. Only Leicester City has this feature, a team that has never figured among the “superior” teams.

What can one conclude from these simulations?

1. The persistence of the “superior” teams to be in the top few ranks over several years suggests that they really are superior and not just lucky. The league appears to have two tiers of teams: 6 elite teams and+14 others.
2. Most of the quality differences among the two tiers can be ascribed to “chance” or “luck”.
3. The supremacy of Leicester City in 2015-16 seems to be a genuine anomaly. While not easily explained by “luck”, it is hard to convince Premier league connoiseurs that incredible luck was not at play in this instance.

(For a completely different approach to the nature of Premier League standings, see the recent article at bit.ly/2j79drn) .

The larger point in this example of simulation and graphics is this:

The example is a real world outcome of a process widely believed to be subject to many complex influences, too complex to sort out in detail for a particular instance. However, we were able to use an easy-to-explain simulation and graphical display which do provide useful information about the interpretation of the outcome.

**Conclusion from Four Examples**

1. Simulation and graphics are not just for fun.
2. These techniques provide powerful tools for data analysis, and are relatively easy to explain to students.
3. Inclusion of such techniques in undergraduate courses opens up the subject to creativity, and may appeal to a wide audience.
4. The approach illustrated in these examples adds the power of subjectivity to a discipline that has been limited by its culture of objectivity.

(KLW 2017/03/08)