Real-time computation (or estimation) of the “probability to win” is difficult. We’ve seem that in soccer games, in elections… but actually, as a professor, I see that frequently when I grade my students.

Consider a classical multiple choice exam. After each question, imagine that you try to compute the probability that the student will pass. Consider here the case where we have 50 questions. Students pass when they have 25 correct answers, or more. Just for simulations, I will assume that students just flip a coin at each question… I have students, and 50 questions

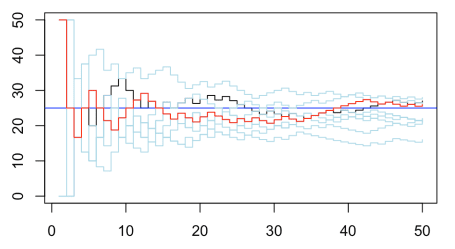
|  |  |
| --- | --- |
| 1  2  3 | **set.seed**(1)  n=10  M=**matrix**(**sample**(0:1,size=n\*50,**replace**=TRUE),50,n) |

Let denote the score of student at question . Let denote the cumulated score, i.e. . At step , I can get some sort of prediction of the final score, using . Here is the code

|  |  |
| --- | --- |
| 1  2 | SM=**apply**(M,2,**cumsum**)  NB=SM\*50/(1:50) |

We can actually plot it

|  |  |
| --- | --- |
| 1  2  3  4 | **plot**(NB[,1],type="s",ylim=**c**(0,50))  **abline**(h=25,**col**="blue")  **for**(i **in** 2:n) **lines**(NB[,i],type="s",**col**="light blue")  **lines**(NB[,3],type="s",**col**="red") |

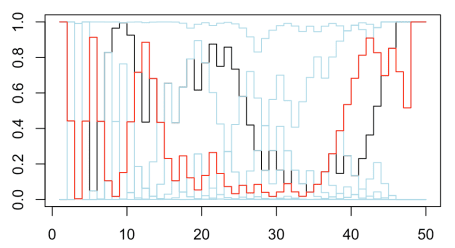
  
But that’s *simply* the prediction of the final score, at each step. That’s not the computation of the probability to pass !

Let’s try to see how we can do it… If after questions, the students has 25 correct answer, the probability should be 1 – i.e. if – since he cannot fail. Another simple case is the following : if after questions, the number of points he can get with all correct answers until the end is not sufficient, he will fail. That means if S\_{i,j}+(50-i+1)< 25[/latex] the probability should be 0. Otherwise, to compute the probability to sucess, it is quite straightforward. It is the probability to obtain at least [latex]25-S\_{i,j}[/latex] correct answers, out of [latex]50-j[/latex] questions, when the probability of success is actually [latex]S\_{i,j}/j[/latex]. We recognize the survival probability of a binomial distribution. The code is then simply

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | PB=NB\*NA  **for**(i **in** 1:50){  **for**(j **in** 1:n){  **if**(SM[i,j]&gt;=25) PB[i,j]=1  **if**(SM[i,j]+(50-i+1)&lt;25) PB[i,j]=0  **if**((SM[i,j]&lt;25)&amp;(SM[i,j]+(50-i+1)&gt;=25)) PB[i,j]=1-**pbinom**(25-SM[i,j],size=(50-i),prob=SM[i,j]/i)  }} |

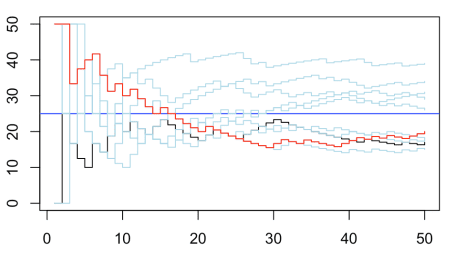
So if we plot it, we get

|  |  |
| --- | --- |
| 1  2  3  4 | **plot**(PB[,1],type="s",ylim=**c**(0,1))  **abline**(h=25,**col**="red")  **for**(i **in** 2:n) **lines**(PB[,i],type="s",**col**="light blue")  **lines**(PB[,3],type="s",**col**="red") |

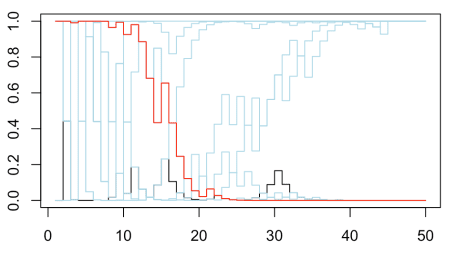


which is much more volatile than the previous curves we obtained ! So yes, computing the “probability to win” is a complicated exercice ! Don’t blame those who try to find it hard to do !

Of course, things are slightly different if my students don’t flip a coin… this is what we obtain if half of the students are good (2/3 probability to get a question correct) and half is not good (1/3 chance),



If we look at the probability to pass, we usually do not have to wait until the end (the 50 questions) to know who passed and who failed



PS : I guess a less volatile solution can be obtained with a Bayesian approach… if I find some spare time this week, I will try to code it…