

Program OPACSA

OPTimal ACceptance Sampling by Attribute of grains

EU research program: **Co-Extra**

Program realized by François-Xavier VIALARD

Guidelines: André KOBILINSKY, François-Xavier VIALARD

November 2006

1 OPACSA, program for the OPTimal ACceptance Sampling by Attributes

The program “OPACSA” allows to put easily in practice the method described in the article [1] to find the cheapest mode of control by attributes of the purity of grain lots. In the examples given here, the purity considered will be the absence of a genetically modified organism (GMO), but as outlined in [1] the program is clearly usable in many other contexts.

Recall that in the considered method of control, an *initial sample* to be sent to the laboratory for analyses is first collected. From this initial sample, the laboratory extracts one or several subsamples and determines the presence or absence of GMO for each of them. It is from these binary responses that the control is made.

The procedure used to get the initial sample is the “*initial sampling protocol*” and the procedure used by laboratories to control this initial sample is called the “*acceptance sampling plan*”. Only the latter is considered here. So it is based on a qualitative detection of the presence or absence of the defect in groups of grains. It is a procedure known to be easy to use and robust in many contexts.

2 Single or double sampling

More precisely, it is proposed to perform either a single, or a double sampling plan. In the single sampling plan, N groups of n grains are separately ground and analysed to determine if they are GMO-positive or not. If there are X positive out of the N groups the GMO proportion is estimated by X/N and the lot is accepted if $X \leq A$, rejected if $X > A$, where A is a predetermined acceptance threshold. Thus the set of parameters defining the plan is (N, n, A) .

It is generally cheaper to perform a double sampling plan, defined as follows. First N_1 groups of n_1 grains are assayed. The lot is accepted if $X_1 \leq A_1$ where X_1 is the number of positive groups and A_1 a predetermined threshold. If $X_1 \geq R_1$, where R_1 is a predetermined rejection threshold, the lot is rejected. In between, that is if $A_1 < X_1 < R_1$, N_2 new groups are assayed. The lot is then accepted if $X_2 \leq A_2(X_1)$, where X_2 is the number of positive among the N_2 new groups and $A_2(X_1)$ a predetermined threshold. The function $A_2 : X_1 \mapsto A_2(X_1)$ giving for each value of X_1 the acceptance threshold at the second step must be a decreasing function of X_1 because the bigger is X_1 , the smaller must be X_2 to compensate. In classical quality control, this function has the form $A_2 - X_1$ where A_2 is a fixed number. The lot is therefore accepted at the second step if $X_2 + X_1 \leq A_2$ and rejected if $X_2 + X_1 \geq R_2 = A_2 + 1$. Here we allow the subsamples examined at step 1 and 2 to have possibly different numbers n_1 and n_2 of grains. It is then natural to adopt a larger frame making possible the use of something less symmetric than the sum $X_1 + X_2$ to base the acceptance at the second step. So the set of parameters to determine in that case is $T = (N_1, n_1, A_1, R_1, N_2, n_2, A_2())$. The brackets following A_2 are put to remind that A_2 is a function.

3 Buyer's and seller's requirements

The buyer wants the proportion of GM grains to be below a *non tolerable* threshold p_{nt} while the seller wants the lot to be accepted if the GM grain proportion does not exceeds a *tolerable* threshold p_t . The tolerable threshold must be smaller than the non tolerable one, that is $p_t < p_{nt}$, otherwise there cannot be any agreement between the buyer and the seller.

For instance, the buyer does not want to accept a lot with a proportion $p > 1\%$ while the seller does not want a lot with $p < 0.2\%$ to be refused.

Statistically speaking, these requirements can be expressed by the following inequalities.

$$\text{Prob}(\text{acceptance} \mid p_{nt}) \leq \beta \tag{1}$$

$$\text{Prob}(\text{rejection} \mid p_t) \leq \alpha \tag{2}$$

The first formalises the buyer's requirement to refuse a lot with a GMO grain proportion $p > p_{nt}$. He does not want the risk of acceptance of such a lot to exceed a predetermined risk β .

The second formalises the seller's requirement to have the lots with $p < p_t$ accepted. He does not want the risk of refusal of such a lot to exceed a predetermined risk α .

The choice of p_{nt} , β , p_t , α depends on many considerations: risk for humans, risk for environment, degree of purity that can be obtained, nature of the grains (commercial seeds, basic seeds, breeder's seeds, grain for human food, for animal feed, etc . . .). Once there is an agreement between contractors on their choice, the problem is to find the acceptance sampling plan leading to a minimal inspection cost among those which satisfies the constraints (1) and (2).

4 Cost of the acceptance sampling plan

This cost has essentially two components. One depending on the number of grains used, one on the number of groups of grains tested. What is important is the ratio between the price of a grain and the price of each test. The latter is taken as the unit, and the price c_g of a grain is thus expressed relatively to it. For instance, $c_g = 0.001$ if the price of a test for one group of grains is equivalent to the price of 1000 grains.

In a double sampling plan, the price is random depending on the probability to have to test the N_2 supplementary groups in the case where the first N_1 tests do not allow to conclude. This probability to have a second step depends on the real proportion p in the lot. So besides the relative price c_g of a grain, one has to provide a value p of this possible expected proportion in order to evaluate the expected cost .

5 How to use the program “OPACSA”

This program being written in the PYTHON software, the first operation, described with more details a little further, is to download this software on the used computer.

The execution is at least four time quicker if the software “*psyco*” is also installed. On Windows you can install its precompiled version which, at the time these guidelines are written (October 2006), works with Python release 2.4, not yet with Python release 2.5. See how to install it a little further.

The following operation is to execute the python module providing the values of the parameters p_{nt} , β , p_t , α , p , c_g and calling the optimization program with these parameters. Maximum values for the numbers N_1 , N_2 have also to be given to avoid in some cases a too long search. In any case, the execution can be stopped by “Ctrl C”.

The functions for the optimization are in the file `OPACSA_funcs.py`. The main one is “`planopt`” which has an argument *params* giving the parameter values

$$\text{params} = (p, p_t, p_{nt}, \alpha, \beta, c_g, N1_max, N2_max)$$

and another “*nf*” name of the output file, for instance `nf="output.txt"`.

There are two ways to introduce “*params*” and “*nf*” and launch the optimization.

The **first** is to execute `OPACSA_win.py`. It produces a window with fields for the *parameters* and one field for the *output file name*. Default values can be provided by the file `OPACSA_win.par`. The results of the search are displayed on the screen and stored in the output file (of name “*nf*”).

The **second** is to define the *parameter values* and the *output file name* in a small program which imports `OPACSA_funcs.py` and launches `planopt`. Two small examples are provided.

The first one `OPACSA_prog1.py` is the simplest possible. It defines *params* (the parameters), *nf* the output file name, import `OPACSA_funcs` and execute `planopt`. The second one `OPACSA_prog2.py` is used to find the optimal acceptance sampling schemes corresponding to table 2 of the reference [1]. Users can adapt these small programs to their own context.

To execute a python module such as *OPACSA_win.py* or *OPACSA_prog1.py*, one way is to double click on its name, which normally executes python with this module. The command “python OPACSA.py” in the command editor produces the same effect and it is also possible to launch the module by IDLE (Python Graphical User Interface) or by the python command line. Note that to get the right version of Python launched, it must appear in the window path.

During the search the program writes in the file *OPACSA_search.txt* the parameter values N1, A1, R1, N2 being explored. The file can be consulted during the execution. Thus the user can get an idea of the time required by the search for each set of values of these 4 parameters. Note that among the parameters of the search are upper bounds for N1 and N2. Standard satisfying values are N1_max= 4 and N2_max= 10. But it is possible to reduce the global time of search by lowering these upper bounds.

At the end of the search, the results appear on the screen and are stored in the output file (*nf*).

6 Python and psyco software

OPACSA is written in the programming language *Python*, which is a freeware. The source of OPACSA being available, users who want to adapt it to their own situation can easily integrate it in their own programs.

The Python software can be found on the net at:

<http://www.python.org/download>

You will find there versions for several systems. “*OPACSA.py*” should work on any of them, but to exemplify, we consider here only the version for Windows XP.

To download it, click on DOWNLOAD, then on “Python Windows Installer” which give you access to the Windows Installer Package. If you want to be sure *psyco* works too (advice given in October 2006), choose to install *python-2.4.msi* rather than *python-2.5.msi*.

If you execute *python-2.4.msi* (for instance by double clicking it after writing it on your hard disk), you get Python installed on your computer.

You can dowload *psyco* on the web site

<http://psyco.sourceforge.net/>

(for Windows, select *download*, then *from precompiled binaries* then click on <http://sourceforge.net/> You get *psyco-1.5.1-win-2.4.zip* on your computer, unzip it and install psyco.

It is not necessary to know Python to use “OPACSA”, but if you want to adapt and consequently modify the source you can have a look at the excellent manual “Learning to Program” of Alan GAULD, available from the net at the address below:

<http://www.freenetpages.co.uk/hp/alan.gauld/>

7 Example

Assume the parameters of the search are $p_t = 0.002$, $\alpha = 0.05$, $p_{nt} = 0.01$, $\beta = 0.05$, $p = p_t/10$, $c_g = 0.001$, $N1_max = 4$, $N2_max = 10$ and that the output is redirected to the file *output.txt*. These parameters and output file are those defined in *OPACSA_prog1.py* and the corresponding optimum acceptance sampling plan can be get by executing it.

Normally *OPACSA_prog1.py* is executed either by a double click on the left button of the mouse, or with the command line

```
python OPACSA_prog1.py
```

executed in the directory where *OPACSA_prog1.py* and *OPACSA_funcs.py* lay. One can also use the following commands in the python command line, assuming *OPACSA_prog1.py* is in *C:/opacsaprog* (respect lower and uppercase letters):

```
from os import chdir
chdir('C:/opacsaprog')
import OPACSA_prog1.py
```

or in IDLE select “File”, then “Open”, then “OPACSA_prog1.py”, and “Run”.

The results of the program, found in “output.txt” are

$$N_1 = 1, \quad n_1 = 333, \quad A_1 = 0, \quad R_1 = 2, \quad N_2 = 6, \quad n_2 = 339, \quad A_2 = (4)$$

The associated cost is 1.85. This explains why the search is only made for $N_1 = 1$, as appears in the file “opti_search.txt” when *optisearch = True*.

The result is here identical to the one in table (2) of [1]. If *cg* were 0.01 instead of 0.001, then n_1 would be found equal to 313 instead of 314. This is because in the computations made for the article the numbers n_1, n_2 were first searched as real numbers, and the integers n_1, n_2 finally taken as the solution were one the two neighbouring integers, but not the right ones in every case.

8 APP number and Artistic licence

The program is referred to at the Agency for the Protection of Program (APP), under the reference number

IDDN.FR.001.500009.000.R.P.2006.000.30100”

It is diffused under the Artistic licence (www.opensource.org/licenses/artistic-license.php).

References

- [1] Kobilinsky A., Bertheau Y. (2005). Minimum cost acceptance sampling plans for grain control, with application to GMO detection. *Chemometrics and intelligent laboratory systems*, **75**, 189–200.

