AUDIT

Le contrôle des comptes à l'aide du langage VBA d'Excel

Par Benoît-René RIVIÈRE
Expert-comptable, COGEDIAC & Associés S.A. à Caen

L'audit de données peut être mené à l'aide de différents outils durant toutes les étapes de la mission du commissaire aux comptes : avec des logiciels spécialisés (Idea, ACL), à l'aide de requêtes SQL [1] ou encore avec des tableurs (Excel, Calc…). C'est souvent cette dernière catégorie d'outils qui a la préférence des auditeurs. Ce sont en effet les outils les plus maniables et les plus flexibles qui soient. Généralement, l'auditeur met en œuvre des fonctionnalités plus ou moins élaborées de son tableur : filtres, tris, formats conditionnels voire tableaux croisés dynamiques, liens intertables (requêtes SQL, « recherchev »)… Une autre possibilité offerte par les tableurs modernes est toutefois fréquemment négligée par méconnaissance : le VBA.

Pourtant le VBA (pour *Visual Basic for Application*) est un puissant langage de programmation qui accompagne toutes les applications de la suite Office de Microsoft. Ce langage de programmation étend à l'infini les capacités de traitement de l'information et d'analyse de données de ces logiciels et notamment d'Excel. Il permet d'automatiser des traitements comme l'analyse de données et de les reproduire de manière identique voire de les systématiser sur l'ensemble des dossiers. Cet article s'efforcera de démontrer l'intérêt que la profession peut tirer de cet outil de développement au travers d'un exemple simple mais efficace.

Résumé de l'article

L'audit de données prend une part de plus en plus importante dans les travaux d'audit du commissaire aux comptes. Pour l'aider à remplir cette tâche, le commissaire aux comptes dispose d'une palette complète d'outils. L'un de ces outils est le VBA (*Visual Basic for Application*). Il est souvent ignoré par la profession alors qu'il est présent sur tous nos ordinateurs. Facile à maîtriser et à mettre en œuvre, il n'attend qu'une seule chose : démontrer aux professionnels que nous sommes les formidables capacités de traitement de données qu'il est en mesure d'apporter à Excel.

Le VBA, un langage de programmation à la portée de tous

Le langage VBA, basé sur le Basic [2], est assez facile à prendre en main. Il étend les capacités de traitement des logiciels hôtes et assure principalement deux fonctions :
• ajout de nouvelles fonctions utilisables directement dans les formules des feuilles de calcul Excel [3],
• création d'applications à l'aide de procédures.
Ce sont plus spécifiquement les procédures qui font l'objet de cet article.
La prise en main du module Visual Basic nécessite :
• de découvrir l'environnement de développement VBA : ce module, semblable sur l'ensemble des applications de la suite Office, assure l'édition des programmes VBA,
• de maîtriser les mots-clefs et la syntaxe du langage : ce langage s'assimile très rapidement, tant les termes (quoique en anglais) sont peu nombreux,
• de comprendre la logique de programmation.
La littérature spécialisée est très fournie et de nombreux sites animés par des passionnés forment une riche mine d'information sur ce sujet [4].

Cet article se veut une initiation et, à ce titre, présente une démarche de mise en œuvre concrète illustrée d'une application d'analyse de données. A l'issue, le lecteur maîtrisera les principales syntaxes de programmation du VBA (boucles, tests…) et saura effectuer les principaux types de manipulations proposés par Excel :
• ouvrir un nouvel onglet dans un classeur, le renommer,
• manipuler des cellules : les lire, y écrire des données y compris des formules de

1. Cf. à ce sujet l'article intitulé "Contrôle des comptes par le commissaire aux comptes à l'aide de l'analyse de données", Revue française de comptabilité *n° 433 de juin 2010 et http://www.auditsi.eu/?p=400. Cet article se proposait de mettre en avant l'intérêt du langage de requêtes SQL dans l'analyse de données tandis que le présent article tente de démontrer l'intérêt du VBA, langage plus généraliste et plus puissant encore.*

2. Extrait de l'encyclopédie Wikipedia (URL : http://fr.wikipedia.org/wiki/BASIC) : « En programmation, Basic est un acronyme pour Beginner's All-purpose Symbolic Instruction Code, qui désigne une famille de langages de programmations de haut niveau. Le Basic a été conçu à la base en 1963 […] pour permettre aux étudiants qui ne travaillaient pas dans des filières scientifiques d'utiliser les ordinateurs. En effet, à l'époque, l'utilisation des ordinateurs nécessitait l'emploi d'un langage de programmation réputé réservé aux seuls spécialistes, en général un langage d'assemblage ou Fortran. Le Basic est indissociable de l'apparition, dans les années 1980, de la micro-informatique grand public. En effet, la plupart des micro-ordinateurs vendus durant cette période étaient fournis avec un interprète Basic […]. »

3. Le tutoriel "Excel : Créer une fonction en VBA" (http://www.auditsi.eu/?p=513) présente la démarche à suivre pour créer une fonction Excel.

4. Le lecteur se référera aux sites internet http://vb.developpez.com/ et http://www.vbfrance.com/ pour plus d'information ainsi qu'aux ouvrages "VBA Excel 2007 - Maîtrisez la programmation sous Excel" (Editions ENI) et "VBA pour Office 2007" (Micro-Application) afin d'approfondir le sujet.

AUDIT

Figure 1 : Etat du stock et bouton de lancement des tests (onglet “STOCKS“)

3	DOSSIER :	STE NORMANDE DE NEGOCE		CLÔTURE :	31/12/2010
4	Collab :	BRR			
5	Date :	24/02/2011			
6					
7					
8					
9					
10	**INSTRUCTIONS :**				
11					
12		1. Coller ci-dessous l'état du stock à analyser			
13					
14		2. Cliquer sur le bouton "Lancer test"	Lancer test		
15					
16					
17	Référence	Désignation	QtéN	PUN	Total
18	120	PALETTE HOUSSEE	0	0	0
19	150	PALETTE	-181	4,4	-796,4
20	300	CHEVRON 6X8	1608	0,64	1029,12
21	512	ELINGUES DIAM 12 MM(F.UT: 500KG)	12	4,42	53,04
22	516	ELINGUES DIAM 16 MM(F.UT:1000KG)	48	5,89	282,72
23	520	ELINGUES DIAM 20 MM(F.UT:1600KG)	64	7,88	504,32
24	524	ELINGUES DIAM 24 MM	15	7,77	116,55

calcul, les formater (couleur, type numérique…),
• activer les filtres automatiques.

Un exemple pour démontrer l’intérêt du VBA pour le commissaire aux comptes

Pour démontrer l’intérêt du VBA pour le commissaire aux comptes, rien ne vaut un exemple concret : l’audit des états de stocks [5]. La difficulté à laquelle se trouve souvent confronté l’auditeur est le volume de données à analyser en des temps réduits : les états d’inventaire sont fréquemment composés de milliers de lignes. Une revue manuelle des états papier ou informatique s’avère rapidement fastidieuse et repose obligatoirement sur des sondages. L’analyse automatisée de données s’affranchit de ces contraintes et permet même des tests exhaustifs.
L’exemple proposé présente un état de stock sur Excel (onglet “STOCK“). Cet état comprend une référence, une désignation, une quantité, un prix unitaire et un total. L’utilisateur colle l’état d’inventaire à auditer dans cet onglet puis clique sur le bouton “Lancer le test“ (figure 1). Le code source VBA va réaliser une série de tests automatiquement sur l’état d’inventaire et créer un nouvel onglet (intitulé “Résultat“) [6].
L’application génère un second onglet nommé “RESULTAT“ qui comprend le résultat des tests. Les calculs ne prennent que quelques instants.

Tests effectués par le programme

L’application effectue les contrôles suivants :
• analyse de la cohérence des quantités (quantités négatives) et des prix unitaires (PU négatifs ou nul avec quantité renseignée),
• contrôle du total de la ligne (quantité x PU)
En fonction des écarts relevés, le programme colorise les anomalies en rouge (PU négatifs par exemple) et attire l’attention de l’auditeur sur de potentielles anomalies (PU nuls) en colorisant certaines cellules en jaune.

Résultat obtenu

Le résultat des tests (cf. figure 2) permet à l’auditeur de se focaliser, dans un premier temps, sur les anomalies réelles et potentielles relevées par le programme

5. Différentes manières d’analyser la cohérence des stocks coexistent notamment à l’aide de requêtes SQL, à titre de comparaison cf. l’article http://www.auditsi.eu/?p=560.

6. Un autre exemple de fonction et de procédure en VBA de retraitement de données (« Audit de données et VBA : automatisation du formatage des données numériques ») est consultable ici : http://www.auditsi.eu/?p=520.

Figure 2 : Résultat des tests (onglet “RESULTAT“)

	A	B	C	D	E	F	G	H	I	J
1	**TESTS SUR LE STOCK**									
2										
3	ANOMALIE									
4	ALERTE									
5										
6			*INFORMATIONS REPRISES DE L'ETAT DE STOCK*			*INFO CALCULEES*				
7	*Référenc*	*Désignation*	*Quantité*	*Prix Unita*	*Total*	*otal CALC*	*CART sur T*	*Qté négat*	*U négatif*	*tal négatif*
8	120	PALETTE HOUSSEE	0,00	0,00	0,00					
9	150	PALETTE	- 181,00	4,40	- 796,40	- 796,40	0,00	X		X
10	300	CHEVRON 6X8	1 608,00	0,64	1 029,12	1 029,12	0,00			X
11	512	ELINGUES DIAM 12 MM(F.UT: 500K(	12,00	4,42	53,04					
12	516	ELINGUES DIAM 16 MM(F.UT:1000K	48,00	5,89	282,72	282,72	0,00			X
13	520	ELINGUES DIAM 20 MM(F.UT:1600K	64,00	7,88	504,32					
14	524	ELINGUES DIAM 24 MM	15,00	7,77	116,55					
15	530	ELINGUES DIAM.30 MM	6,00	26,26	157,56					
16	594	MAINS ARTEON	0,00	0,00	0,00					
17	710	CALE BOIS 20X20X30	414,00	3,48	1 440,72					
18	1111110	F 14X20X3 HR 33 CF20/C20F	0,00	0,00	0,00					
19	1111120	F 20X20X3 HR 33 C26F	0,00	0,00	0,00					
20	1114120	F 20X20X3 HR 33 T400	0,00	0,00	0,00					
21	1131190	F 25X25 ALVEOLE N HR 33	65,00	0,00	0,00				X	

(celles-ci étant mises en évidence à l'aide de couleurs). Pour ce faire, l'auditeur a toute latitude pour filtrer les données par couleur de cellule et/ou en fonction des données chiffrées.

Explication pas à pas du code-source

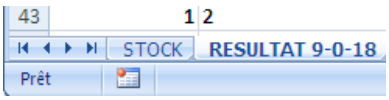
Le code-source du programme présenté ci-avant dans l'article est détaillé dans les grandes lignes dans la présente partie [7]. Il est composé de seulement 140 lignes.

La lisibilité du code-source est améliorée par :
- l'ajout de commentaires (les commentaires sont précédés d'une apostrophe ; ce signe signale à l'interpréteur VBA de ne pas exécuter la ligne) expliquant le fonctionnement du programme ;
- le texte du code est décalé en marge en fonction de la progression dans les boucles et procédures. Cette mise en forme est purement conventionnelle ; le fait de ne pas la respecter n'a aucune incidence sur l'exécution du code mais réduit sensiblement sa compréhension par un programmeur.

Le tableau 1 fournit des extraits commentés du code-source du programme. Des conseils complémentaires et quelques sources d'erreurs communes sont résumés dans l'encadré 2.

7. Le code-source représente un volume trop important pour être reproduit en intégralité dans la revue ; le lecteur se reportera sur le site www.auditsi.eu afin d'en télécharger l'intégralité.

Tableau 1 : Extraits commentés du code-source de l'application

Code VBA	Commentaires
Option Explicit Sub TestStocks() Cette procédure teste la cohérence d'un état de stock	
Option Explicit	Cette clause est facultative ; elle oblige le programmeur à déclarer toutes ses variables => en cas d'erreur de saisie sur le nom d'une variable, le programmeur détectera plus facilement l'erreur
Sub NomProcédure *[…]* End Sub	Délimite une procédure ; la procédure "TestStocks()" est composées d'un ensemble d'instructions cohérentes qui effectuent des actions déterminées.
'	Indique à l'interpréteur VBA de ne pas exécuter la ligne (ce qui permet de documenter le code source avec des commentaires explicatifs).
Dim déb, fin As Variant Dim débanalyse As Boolean	'1ère et dernière lignes utilisées 'tant que FALSE : l'analyse n'est pas commencée
Dim variable as type	Déclaration des variables utilisateur. Les variables permettent de stocker, le temps de l'exécution du programme, des données de toute nature résultant de calculs, de saisie, de lecture de fichiers... Type de variable : Boolean : booléen, type binaire (Vrai ou Faux) Single : réel (maxi : 3,402823E38) Double : réel (maxi : 1,79769313486232E308) Integer : nombre entier (jusqu'à 32 767) Long : nombre entier (jusqu'à 2 147 483 647) String : chaîne de caractères Variant : représente n'importe quel type de variable
Sheets(1).Select 'la feuille n°1 (la plus à gauche) contient le détail du stock ActiveSheet.UsedRange.Select déb = Selection.Cells(1, 1).Row fin = Selection.Cells.Rows.Count + déb - 1 'Crée une FT additionnelle qui contiendra le résultat de l'analyse Sheets.Add FT_Résultat = "RESULTAT" & Hour(Now) & «-» & Minute(Now) & «-» & Second(Now) ActiveSheet.Name = FT_Résultat Sheets(FT_Résultat).Move After:=Sheets(2)	
Sheets (*index ou nom*).select	Indique sur quel onglet du classeur travailler ; l'onglet est précisé à l'aide d'un index (indiquant sa position, 1 étant le plus à gauche) ou de son nom. Par exemple, Sheets(1) ou Sheets(« STOCK ») représentent ici la même feuille de travail. 
Sheets.add	Crée une nouvelle feuille de travail (onglet)
Now Hour(), Minute(), Second()	Cette fonction renvoie la date et l'heure du système ; correspond à la fonction MAINTENANT() d'Excel. Ces fonctions renvoient respectivement l'heure, les minutes et les secondes contenues dans la date/heure renseignée.
Activesheet.	Représente la feuille de travail sélectionnée
Activesheet.Name=	Change le nom de la feuille active (ici, la feuille prend le nom "RESULTAT" ainsi que l'heure à laquelle elle a été générée.
.Move After :=Sheets(2)	Déplace la feuille mentionnée après la feuille 2.
UsedRange.	Représente la plage de cellules utilisée par une feuille de travail
ActiveSheet.UsedRange.Select	Sélectionne la plage de cellules utilisées dans la feuille active
Variable =	Attribue une valeur à une variable, par exemple : a = 1

AUDIT

Code VBA	Commentaires
déb = Selection.Cells(1, 1).Row	Cells(1,1) représente la cellule ligne 1, colonne 1 de la sélection ; Row fournit le numéro de la rangée (ligne), Column celui de la colonne Par exemple : si la sélection = $B12 :$H25, Cells(1,1) est située en B12, Selection.Cells(1,1).Row renvoie 12 Ici les variables déb et fin représentent les premières et dernières utilisées par le tableau Excel
‘Titre feuille de calcul ligneencours = ligneencours + 1 With Range(«A» & ligneencours) .Value = «TESTS SUR LE STOCK» .Font.Size = 18 .Font.Bold = True End With	
ligneencours = ligneencours + 1	Augmente la valeur de la variable « ligneencours » de 1
With… End With	Permet de travailler sur un objet tout en évitant de devoir le répéter à chaque ligne de programme. Ainsi With Range signale que les lignes de code suivantes qui commencent par un point “. “ font référence à la cellule référencée par Range (l’objet). Le code source suivant, bien que plus lourd, est équivalent : Range(“A“ & ligneencours).Value = “TESTS SUR LE STOCK“ Range(“A“ & ligneencours).Font.Size = 18 Range(“A“ & ligneencours).Font.Bold = True
Range (« réf cellules »)	Désigne une plage de cellules
.Value .Formula .FormulaLocal	Renvoie à : la valeur d’une cellule La formule Excel (en anglais) La formule Excel (dans la langue Excel) Formula et FomulaLocal permettent de lire les formules de calcul d’un tableau mais également de les modifier.
.Value = “TESTS SUR LE STOCK“	La cellule désignée comprend le texte désigné entre guillemets. En Basic, les guillemets entourent systématiquement des caractères non numériques.
.Interior.Color = vbYellow	Colore la cellule désignée en jaune
.HorizontalAlignment = xlCenter .Font.Size = 18 .Font.Bold = True .Font.Italic = True .NumberFormat = «# ##0.00»	Centre le contenu des cellules désignées (horizontalement) Met les caractères : en taille 18 en caractères gras en italique (= False pour enlever l’italique) Formate les nombre avec deux décimales et séparateurs de milliers
‘Reprise des info de l’état de stock ligneencours = ligneencours + 1 For j = 1 To 5 Sheets(FT_Résultat).Cells(ligneencours, j).Value = Sheets(1).Cells(i, j).Value If j > 2 Then Sheets(FT_Résultat).Cells(ligneencours, j).NumberFormat = «# ##0.00» Next j	
For var = déb To fin Next var	Les instructions For To font prendre successivement les valeurs de déb à fin à la variable var tout en exécutant les instructions comprises jusque l’instruction Next. Lorsque Next est atteinte, l’interpréteur VBA retourne sur l’instruction For, incrémente la variable var de 1 et recommence l’exécution des instructions jusqu’à ce que var soit égal à fin. La suite d’instructions formée par For To Next s’appelle une boucle. For i = 1 to 5 Next : exécute les instructions cinq fois
Sheets(FT_Résultat).Cells(ligneencours, j).Value = Sheets(1).Cells(i, j).Value	La partie située à gauche du signe égal désigne une cellule (*Cells* en anglais) de la feuille (*Sheets*) codifiée « FT_Résultat » qui prendra la valeur (=) d’une cellule de la feuille 1.
If… Then… ElseIf… End If	Test conditionnel : Si condition1 alors action1 sinon si condition2 alors action2 If a<0 then dette=-a else creance=a : si a est négatif alors la variable dette = -a sinon créance=a

‘Analyse des totaux => anomalie si < 0 ou si qté * PU <> totaux
If (Sheets(1).Range(“E“ & i).Value < 0) Or (Round Sheets(1).Range(“C“ & i).Value * Sheets(1).Range(“D“ & i).Value, 2) <> Round Sheets(1).Range(“E“ & i).Value, 2) Then
Sheets(FT_Résultat).Range(“E“ & ligneencours).Interior.Color = vbRed
Sheets(FT_Résultat).Range(“F“ & ligneencours).Value = Sheets(1).Range(“C“ & i).Value * Sheets(1).Range(«D» & i).Value
Sheets(FT_Résultat).Range(“G“ & ligneencours).Value = Sheets(FT_Résultat).Range(“F“ & ligneencours).Value - Sheets(1).Range(“E“ & i).Value
Sheets(FT_Résultat).Range(“F“ & ligneencours & «:» & «G» & ligneencours).NumberFormat = «# ##0.00»
With Sheets(FT_Résultat).Range(“J“ & ligneencours)
.Value = “X“
.HorizontalAlignment = xlCenter
End With
End If	

Analyse de l’extrait du programme ci-dessus	Si le total est inférieur à zéro ou si qté x PU différent du total : Le total est coloré en rouge (colonne E), Le total calculé est présenté en colonne F, L’écart de totalisation est présenté en colonne G, Une croix est mentionnée en colonne J (colonne des anomalies sur totalisation).
‘Filtres automatiques Sheets(FT_Résultat).Range(“A“ & lignefiltre).AutoFilter	
.AutoFilter	Applique le filtre automatique à la plage de cellules sélectionnée

Attacher le code-source à un bouton

Pour faciliter l'exécution du programme, il est possible de l'associer à un bouton :

Pour ce faire, dessiner un bouton (par exemple avec une des formes proposées dans le menu Insertion).
Puis, faire un clic droit dessus et sélectionner "Affecter une macro", choisir la macro adéquate et le tour est joué !

Pour conclure sur le sujet

Après ce tour d'horizon, l'auditeur connaît maintenant les bases du VBA. Il n'a plus qu'à se l'approprier, à l'utiliser au quotidien et à en faire un outil d'audit… pour que VBA deviennent *VB for Audit*.
L'auditeur aura compris que les traitements offerts par le VBA sont infinis, rapides et sécurisent les calculs dans la mesure où ceux-ci ne dépendent pas des formules de calcul de la feuille de calcul mais du programme. Donc plus de risques liés à des formules effacées ou non dupliquées par l'utilisateur. ■

Encadré 1 : Et les autres langages ?

Le VBA n'est pas le seul langage disponible. D'autres langages plus généralistes existent : Delphi/TurboPascal, C++, Visual Basic… Toutefois, le VBA est un langage attaché à une application donnée (Excel, Access…) et de ce fait est plus aisé à mettre en œuvre même s'il est tout fait possible d'interagir sur des feuilles Excel avec des langages tels que Delphi. Cependant, le VBA présente l'inconvénient de ne pas être compilé, c'est-à-dire que le programme est modifiable par l'utilisateur et est moins rapide d'exécution qu'un programme composé dans un langage compilé (l'opération de compilation rendant toute modification par l'utilisateur impossible)[8].

Encadré 2 : Ecueils à éviter et conseils de programmeur

• Récursivité des procédures et fonctions : la récursivité est la démarche qui consiste pour une procédure ou fonction à s'appeler elle-même. Dans ce cas, si le programmeur n'a pas été suffisamment attentif, le programme risque de tourner en rond sans fin… jusqu'à ce que le programme s'arrête inopinément pour dépassement de pile.
• Boucles sans fin (do… loop while et for… to… next) : une boucle sans fin est une boucle qui de par la définition des critères de sortie définis par le codeur ne peut jamais s'arrêter. L'issue des boucles sans fin est généralement la même que celle d'une récursivité mal gérée.
• Erreur dans les tests conditionnels if… then… elseif… end if : les conditions énumérées par le programmeur doivent être rédigées avec soin. Il est en effet fréquent que des conditions mal rédigées aient pour conséquence des résultats non conformes.
• L'usage de la clause "Option explicit" est vivement recommandé en début de programme. En effet, elle oblige le programme à déclarer l'ensemble de ses variables. Ainsi toute erreur saisie dans les variables lui sera instantanément signalée.
• Documentation du code : il est conseillé de commenter les lignes de code afin d'en comprendre le sens plusieurs mois après sans pour autant se replonger dans la lecture du code…

8. L'article "Les langages de programmation compilés alliés à Excel : outils puissants de traitement de données à disposition de la profession comptable" accessible sur http://www.auditsi.eu/?p=543 approfondit ce sujet.

Abstract

Data analysis is becoming increasingly important in audits carried out by statutory auditors. An extensive range of tools exist to support auditors in this task including VBA (Visual Basic for Application). Although often ignored by the profession, VBA is present in all our computers and is simple to understand and use. Ready and waiting therefore to demonstrate to professionals the incredible data management capacities it can bring to Excel.

Bibliographie et échange avec le lecteur

Mémoire *"Extraction et exploitation des données du système d'information dans le cadre du commissariat aux comptes : méthodologie & outils"*, mémoire d'expertise comptable de Benoît-René Rivière, téléchargeable sur le site bibliotique et sur le site internet : www.auditsi.eu.

Sites internet http://vb.developpez.com/ et http://www.vbfrance.com/.

Ouvrages *"VBA Excel 2007 - Maîtrisez la programmation sous Excel"* (Editions ENI) et *"VBA pour Office 2007"* (Micro-Application).

Blog dédié à l'audit et aux systèmes d'information www.auditsi.eu.

Espaces d'échanges ouverts à tous sur www.pacioli.fr : *"Audit & Systèmes d'Information"* et *"Le Cercle des Développeurs"*.