Implantation d'un assistant graphique amélioré pour Microsoft Excel: l'assistant PostGraphe 2

Francis Fauteux et Guy Lapalme

Août 1996

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Université de Montréal

TABLE DES MATIÈRES

1 Introduction

1.1 Présentation

Le rapport statistique joue un rôle crucial dans notre ère d'information et de communication: il transmet les messages importants contenus dans des vastes et confus bassins de données. Ces messages se trouvent parfois dans les données individuelles, mais le plus souvent dans les tendances générales; or l'expression compréhensible de ces tendances requiert certains moyens visuels, particulièrement deux: le tableau et le graphique.

Le première structure, le tableau, permet d'organiser les données brutes par catégorie. Par exemple, pour les résultats d'une recherche effectuée sur une race d'oiseau. Nous reprendrons maintes fois cet exemple dans notre rapport; six sujets ont été testés: Frio, Kwak, Saru, Pouf, Bafa et Torr. On a choisi des oiseaux nés différentes années, respectivement 1980, 1983, 1985, 1982, 1981, 1984. On les a nourri avec des fruits et des grains. Leur consommation en grains et en fruits par semaine a été respectivement, en grammes: 12.2, 1.2, 8, 4.6, 9.7 et 11.25, et 8.25, 18.7, 11, 14.9, 11.2, 8.4. Présentés ainsi, les résultats sont peu éloquents et difficiles à déchiffrer. Un tableau (Tableau 1.1) aide à avoir une meilleure vue d'ensemble:

Oiseau	Naissance	g de grain	g de fruit
Frio	1980	12.2	8.3
Kwak	1983	1.2	18.7
Saru	1985	8.0	11.0
Pouf	1982	4.6	14.9
Bafa	1981	9.7	11.2
Torr	1984	11.3	8.4

Tableau 1.1: Résultats de la recherche sur les oiseaux

Mais si ce tableau permet de classer efficacement ces données, leurs tendances générales restent difficiles à percevoir. Le graphique permet de clarifier ces tendances, de déchiffrer visuellement les points importants des résultats. Par exemple, nous voulons déterminer l'existence d'une corrélation entre les consommations des deux aliments, un graphique permet d'afficher clairement la relation (Figure 1.1). Si nous voulons plutôt comparer les consommations de grain entre les spécimens, nous utiliserons un graphique différent (Figure 1.2).

Bien sûr, avant de créer ces figures élaborées, il faut conserver, organiser et traiter des données. C'est pourquoi on a créé le **tableur**. Ce logiciel spécialisé génère, stocke, traite et analyse les informations. Son concept consiste à diviser une "feuille" virtuelle en cellules adressables, à l'instar d'une matrice bidimensionnelle. Chaque cellule, vide à l'origine, peut accueillir soit une donnée (e.g. numérique, textuelle, booléenne, etc.), soit une **formule**. La formule génère automatiquement des données à partir de valeurs fixes et du contenu d'autres

cellules. Par exemple, une cellule peut contenir une formule calculant la moyenne du contenu d'une **plage** de cellules. Ainsi, lorsqu'une donnée est modifiée, tous les résultats des formules dépendant de cette donnée sont instantanément recalculés, ce qui procure un avantage indéniable au tableur en facilitant l'analyse des conséquences des changements de données.

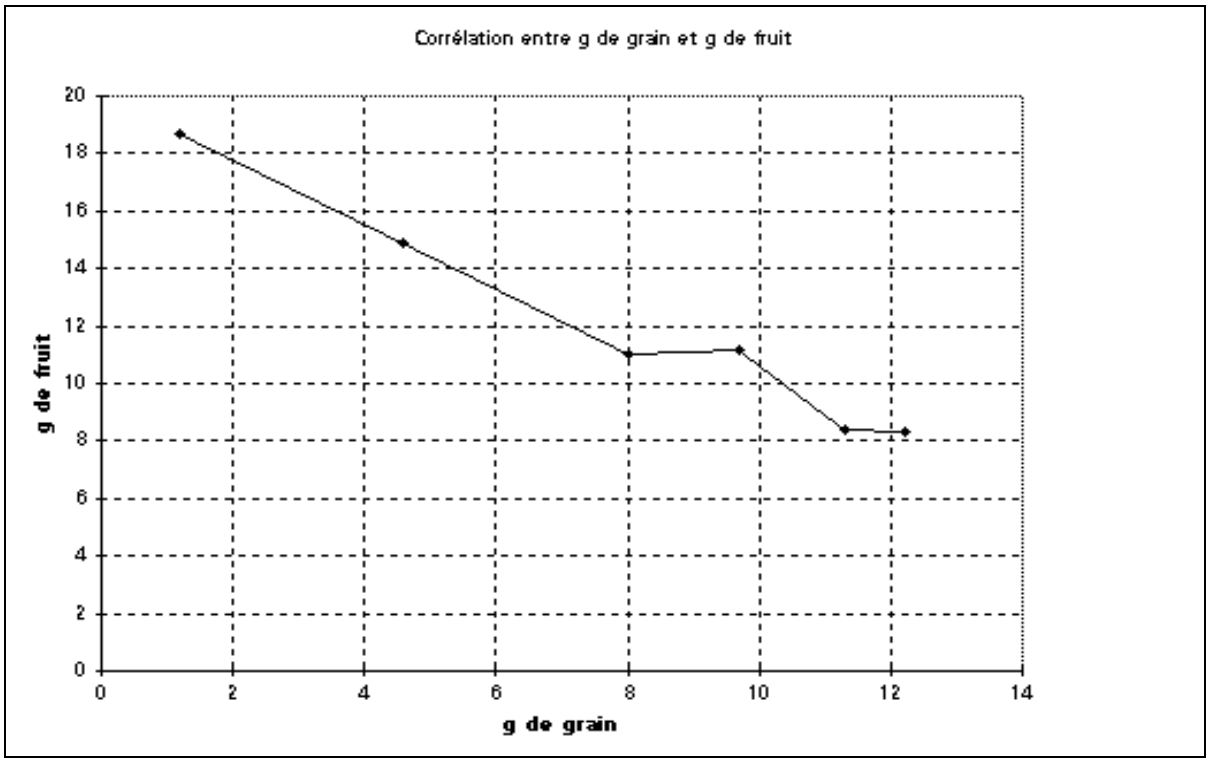

Figure 1.1: Graphique montrant la corrélation

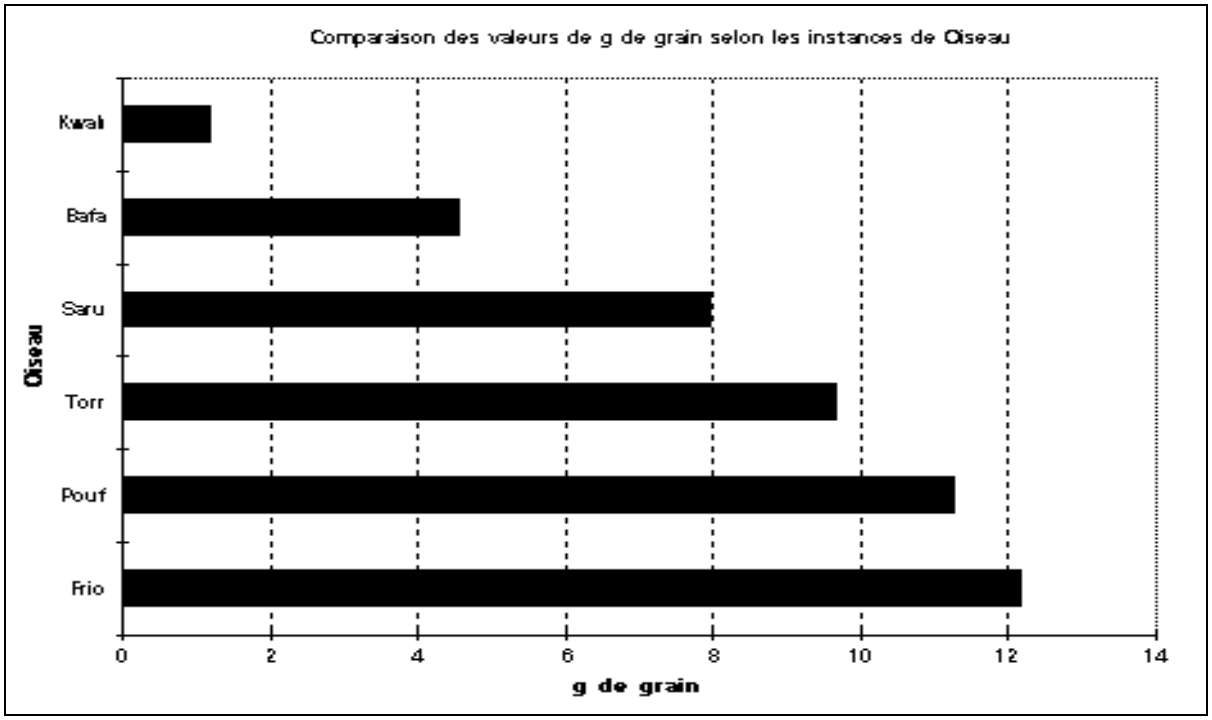

Figure 1.2: Graphique représentant la comparaison

La génération automatique de graphiques est un autre avantage du tableur. L'usager spécifie le type de graphique et les modalités de ses éléments (légende, titre, etc.); le graphique est dessiné automatiquement à partir des données de la plage de cellules sélectionnée.

Excel de Microsoft est un exemple connu de tableur, proposant une interface polyvalente et un vaste jeu de fonctions. Ses concepteurs l'ont même nanti de systèmes d'aide à l'usager appelés **assistants** pour les tâches ardues. L'assistant permettant de générer des graphiques est nommé **assistant graphique**. Celui-ci présente à l'usager une série de menus pour la sélection de la forme du graphique, de la façon dont il devra afficher les données (axes, nombre de courbes, contenu de la légende, etc.), et des étiquettes des éléments (titre, légende, axes, etc.). Ainsi, l'usager a pratiquement les pleins pouvoirs sur le type de graphique à afficher.

Pourtant, ce pouvoir est à la fois bienfaiteur et dangereux pour l'usager. Rien ne garantit que ses choix seront les meilleurs s'il ne sait pas quel mode d'affichage servira le mieux son intention, et souvent, l'assistant lui proposera des graphiques inappropriés. Reprenons l'exemple où le rédacteur veut démontrer la corrélation entre les consommations de grains pour chaque oiseau. Le graphique par défaut généré par Excel (Figure 1.3) s'éloigne beaucoup du graphique "idéal" présenté à la Figure 1.1. On peut évidemment changer le graphique par défaut mais ceci ne change pas le problème fondamental d'adaptation selon l'intention du rédacteur. Donner à l'usager des choix formels seulement peut entraîner des résultats médiocres. Voulant remédier à cela, nous avons réalisé notre propre assistant graphique: l'*assistant Postgraphe 2*, dont la fonction sera de construire des graphiques selon les **intentions** de l'usager.

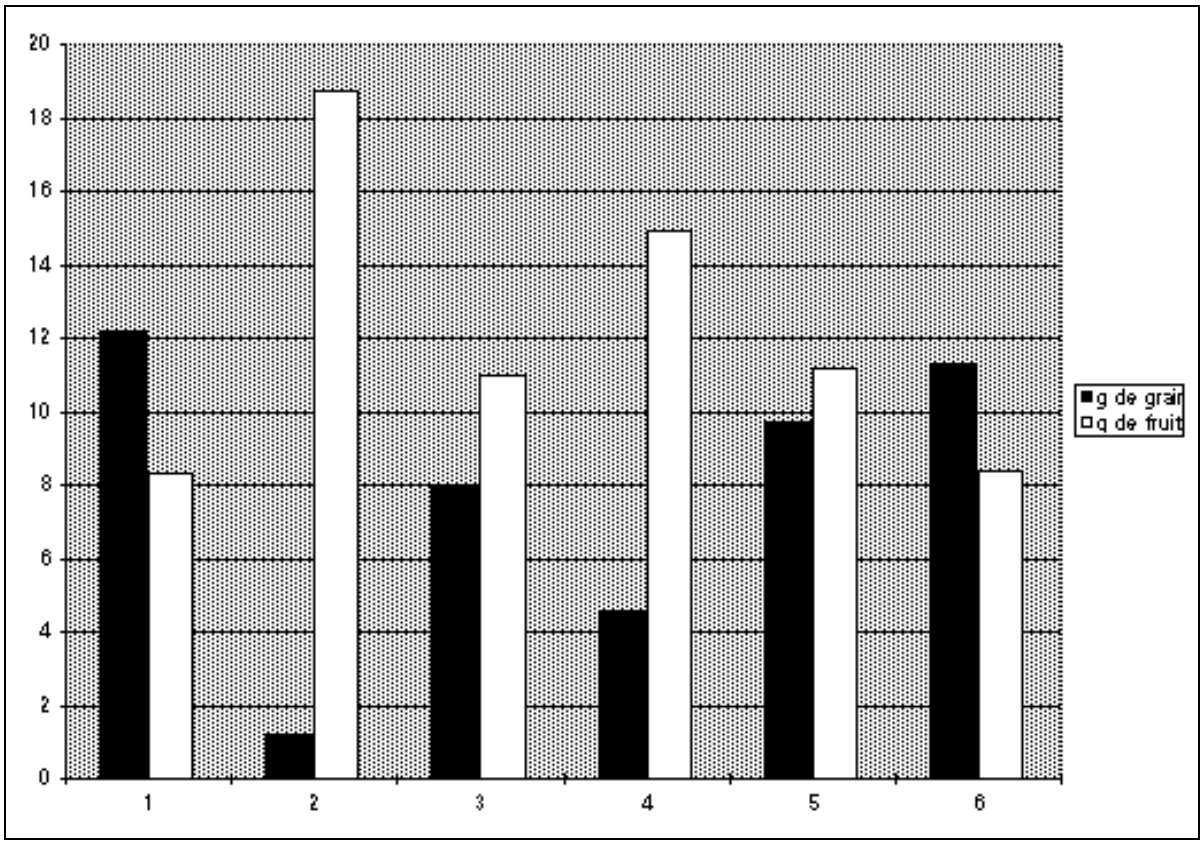

Figure 1.3: Graphique par défaut d'Excel

Notre projet continue les travaux entrepris par Guy Saint-Denis durant l'été 1996, eux-mêmes basés sur la recherche de Massimo Fasciano [Fasciano, 1996] portant sur la génération automatique du rapport statistique. Elle a mené au développement du prototype *PostGraphe*, un système programmé en Prolog qui génère des figures (tableaux et graphiques) et des textes à partir de données brutes selon les intentions multiples exprimées par l'usager.

Guy Saint-Denis, durant l'été 1996, a créé une première version de PostGraphe pour l'utilisation avec Excel: l'*assistant PostGraphe*, qui reprend essentiellement les résultats de la recherche de Fasciano mais s'éloigne un peu de PostGraphe. Son interface est basée sur les fenêtres de dialogue, comme celle des assistants d'Excel. De plus, il ne génère que des graphiques répondant à une intention simple, non des rapports complets satisfaisant plusieurs intentions. Cette version de l'assistant, programmée en Visual Basic, n'a pu être complétée par manque de temps; ainsi, les graphiques générés manquent parfois de clarté et plusieurs types de graphiques n'ont pu être implantés. Les problèmes ont été pour la plupart causés par les contraintes de l'environnement Excel vis-à-vis les outils graphiques.

Nous avons complété les travaux entrepris par Guy Saint-Denis en créant un assistant à l'image de l'assistant graphique d'Excel, mais qui demande l'entrée de paramètres semblables à ceux demandés par PostGraphe, le plus important d'entre eux étant l'**intention** de l'usager. Nos travaux sont orientés vers l'implantation d'une interface qui, comme PostGraphe, ne génère que des résultats valides et logiques.

Pourtant, nos travaux ne s'arrêtent pas là. Notre assistant graphique est également bilingue et permet d'affecter des fonctions aux données. Aussi, il vérifie toutes les données en entrée pour s'assurer d'éviter tout conflit.

Tout cela nous mène à l'implantation de l'*assistant PostGraphe 2*, programmé en Visual Basic. Les modalités du langage seront présentées plus loin dans le rapport. Pour commencer, nous présenterons le mode d'utilisation de l'assistant.

Nous ne mentionnerons pas explicitement les travaux de Guy Saint-Denis dans notre rapport, puisque l'essentiel de ses travaux se retrouve dans notre implantation. L'organisation générale et plusieurs sections de ce rapport sont fortement inspirées du rapport de Guy St-Denis [St-Denis 96].

1.2 Utilisation de AP2

Cette section sert en quelque sorte de manuel de l'usager. Nous y décrivons les étapes à suivre pour que se génère le graphique.

1.2.1 Installation de l'assistant

L'assistant a la forme d'une macro complémentaire. On l'installe en sélectionnant "Macro complémentaire..." du menu "Outils". Dans la boîte de dialogue, on sélectionne la macro correspondant à l'assistant et on appuie sur OK.

L'assistant peut ensuite être appelé à partir du menu "Outils" de la feuille de calcul. On peut également demander l'affichage de la boîte à outils PostGraphe à partir du menu d'affichage. Cette boîte contient deux boutons. Le premier appelle l'assistant graphique. Le deuxième permet d'afficher les messages et les informations de l'assistant soit en anglais ou en français.

1.2.2 Construction du tableau d'entrée

AP2 n'accepte qu'une forme de tableau bien précise. Ce tableau doit faire partie d'une feuille de calcul, comme n'importe quel tableau Excel. Un exemple de tableau d'entrée est présenté à la Figure 1.4.

D'abord, les données doivent être arrangées sous forme de colonnes, c'est-à-dire que chaque colonne doit contenir les informations et valeurs d'une seule variable. La première ligne du tableau contient les noms des variables. Ceux-ci doivent être différents entre eux, et certains termes utilisés dans le programme sont à éviter: "Série", "(aucune)", "elimine" et un mot vide (""). La deuxième ligne contient la classe de la variable, dont les possiblités sont décrites dans la section 2.1.1.Il est préférable mais non obligatoire de respecter la langue de l'assistant dans le choix des classes. La troisième ligne contient une valeur booléenne (vrai/faux) qui indique si la variable est candidate de clé, c'est-à-dire si le programme peut la considérer comme une clé potentielle de la relation telle que décrite à la section 2.1.2. Plusieurs variables peuvent être candidates de clé; la clé véritable de la relation dépendra des choix de l'usager. Si la variable n'est pas candidate, la cellule peut rester vide. Les lignes suivantes contiennent les valeurs (tuples) de la variable.

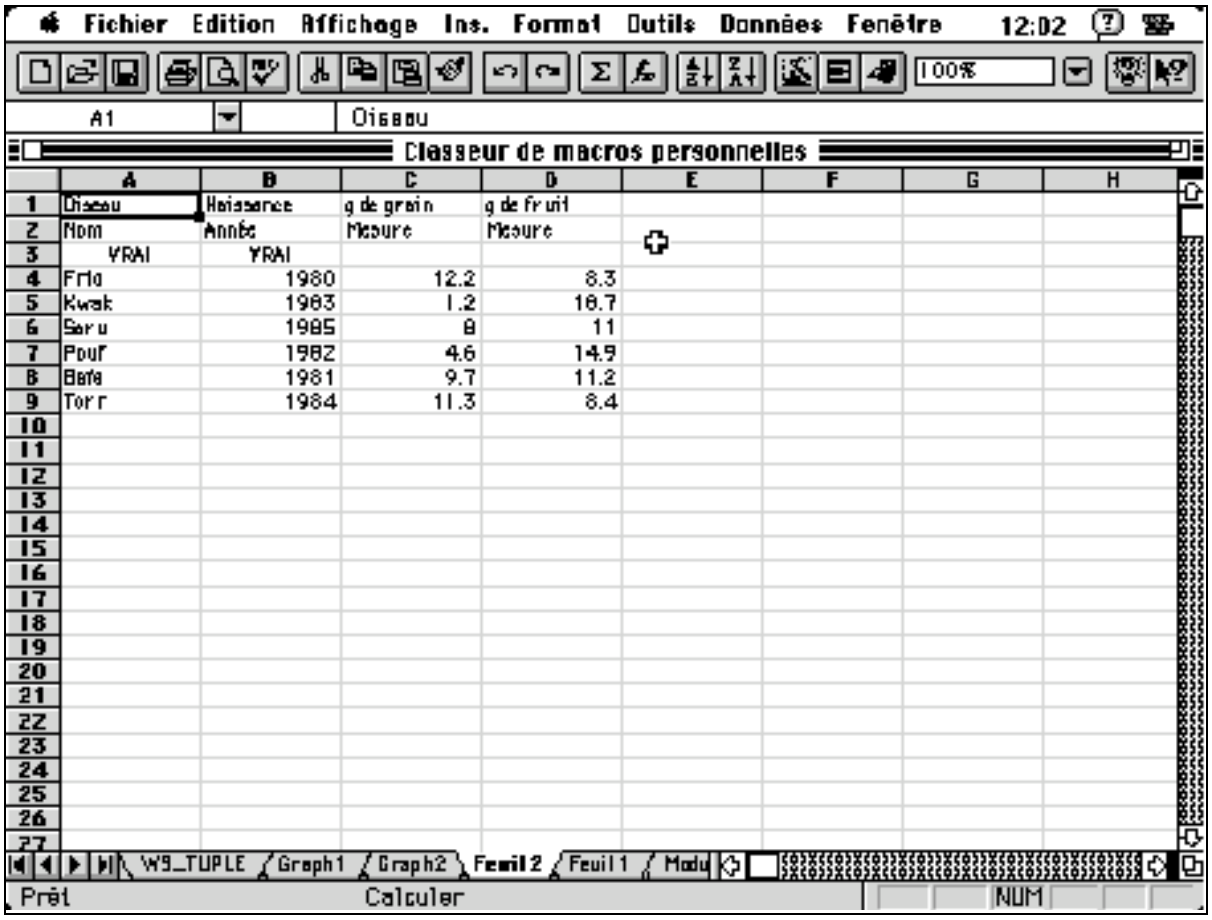

Figure 1.4: Tableau d'entrée de AP2

1.2.3 Sélection du tableau d'entrée

La première étape de AP2 imite l'assistant graphique d'Excel. Une boîte de dialogue (Figure 1.5) demande de confirmer la sélection de la plage qui a été faite. Les boîtes représentées sont les boîtes de la version française. Les boîtes de la version anglaises sont semblables, seulement le texte a été traduit.

La sélection peut se faire en écrivant les coordonnées de la plage dans la boîte, ou en sélectionnant la plage directement sur la feuille de calcul. Si l'adresse donnée est invalide ou comporte des erreurs de syntaxe, Excel demande une correction. L'usager peut sélectionner des colonnes de données non contiguës et dans n'importe quel ordre, pourvu qu'elles aient toutes la même hauteur et que celle-ci dépasse trois cellules. Un avantage de ce type de sélection est qu'il permet de trouver une relation particulière dans un tableau à plusieurs variables. On n'a pas à sélectionner tout le tableau, mais seulement les données intéressantes. Aussi, ce mode d'entrée est plus simple que celui de PG, mais rappelons-nous que le but de PG insistait d'avantage sur la génération efficace d'un rapport que sur la simplicité de l'interface.

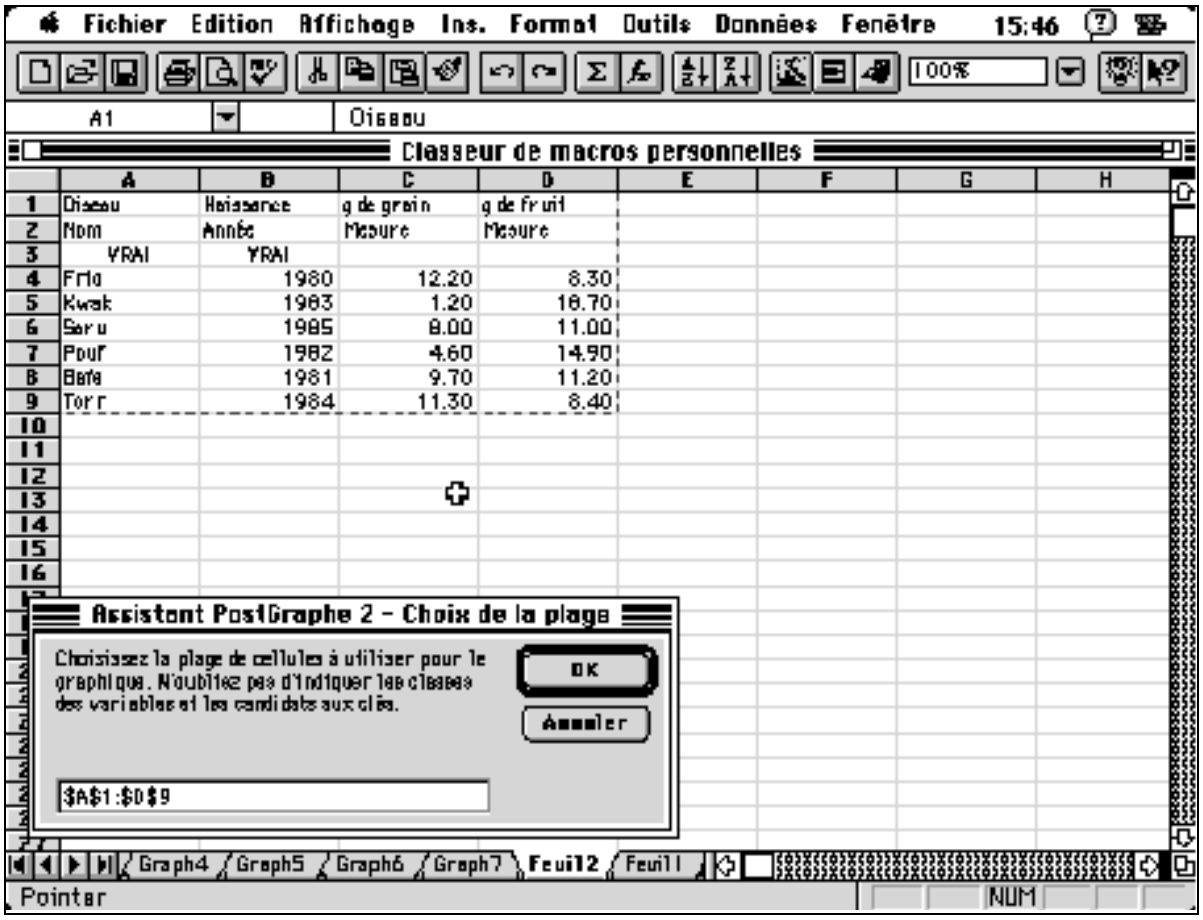

Figure 1.5: Première étape de AP2

Si la plage n'est pas valide, la boîte de dialogue revient en indiquant l'erreur. Les sources d'erreur possibles sont les suivantes:

- Si une plage de forme invalide est sélectionnée. Une plage invalide contient soit des colonnes trop petites (trois colonnes et moins), soit des colonnes de tailles différentes entre elles.
- Si les noms de variables se répètent ou utilisent les termes interdits décrits à la section précédente.
- Si une variable contient des données invalides (par exemple, des mots pour une variable numérique) ou des cellules vides.
- Si la sélection de variables ne peut être arrangée de façon valide. Il faut noter que toutes les variables sélectionnées seront automatiquement utilisées dans le graphique.

Les structures de données décrites à la section 2.3.2 sont les seuls arrangements valides. Si une seule série est considérée, on peut traiter de deux à quatre variables. Si un groupe sériel est considéré (i.e. toutes les variables sérielles potentielles ont le même type), l'arrangement peut comprendre une ou deux variables non sérielles et jusqu'à cinq séries. Si deux groupes sériels sont considérés, les deux doivent contenir deux variables; sinon, le plus petit n'est pas considéré. Dans l'arrangement final, un seul groupe sériel sera choisi, mais deux choix peuvent être présentés.

1.2.4 Sélection de l'intention et des arguments

La deuxième étape de AP2 consiste à choisir une intention en appuyant sur le bouton d'option correspondant. L'intention par défaut est toujours la présentation. La sélection des arguments de l'intention s'effectue dans la boîte de liste. Les arguments sont chacun entre parenthèses et sont séparés par une virgule. Les groupes sériels sont représentés par la liste des variables sérielles. La boîte de dialogue est montrée à la Figure 1.6.

Les intentions incompatibles avec les données ont été préalablement identifiées et désactivées dans la boîte. Tous les arrangements de variables réalisables sont proposés pour chaque intention. Pour aider l'usager, le libellé de l'intention sélectionnée est présente au bas de la fenêtre.

Comme sur la plupart des fenêtres, il se trouve trois boutons principaux. Le bouton *Suite* mène à l'étape suivante, le bouton *Précédent* effectue un retour à l'étape précédente et le bouton *Annuler* permet de quitter instantanément l'assistant sans générer de graphique.

Sur cette fenêtre se trouve toutefois un bouton supplémentaire, le bouton *Fonctions*. Ce bouton mène à la boîte de dialogue de sélection des fonctions. Dans certains cas, aucune fonction ne sera applicable et le bouton sera désactivé. Dans un autre cas, il faudra absolument appliquer une fonction; le bouton *Suite* sera alors désactivé donc, pour continuer, il faudra aller à la fenêtre des fonctions.

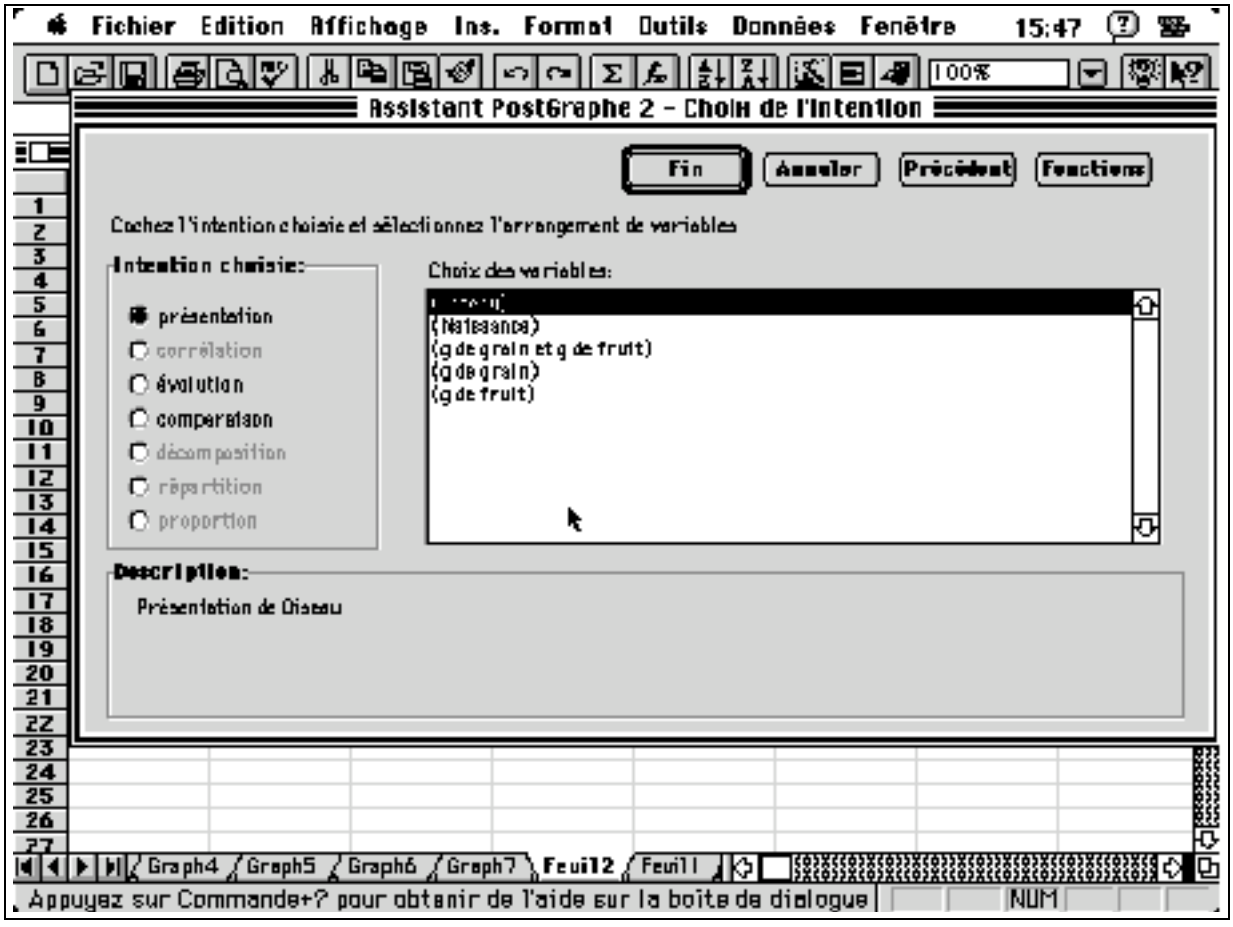

Figure 1.6: Deuxième étape de AP2

1.2.5 Sélection des fonctions

Cette étape est accessible en appuyant sur le bouton *Fonction* de la deuxième étape. Elle utilise la variété de fonctions offertes par Excel. Elle permet d'assigner une fonction à une des variables non sérielles quantitatives et d'assigner une fonction au groupe sériel. La boîte de dialogue de cette étape est montrée à la Figure 1.7.

Les fonctions appliquables à une variable sont des fonctions mathématiques prédéfinies dans Excel et applicables à un nombre simple. Par exemple, si nous voulions arrondir dans notre exemple les consommations de grain, il suffirait de sélectionner la variable et la fonction d'arrondissement.

Les fonctions appliquables au groupe sériel forment en quelque sorte une seule variable avec toutes les variables sérielles. Par exemple, on peut prendre la moyenne ou la somme des variables sérielles au lieu de prendre chaque série individuellement. Cela permet parfois d'accepter le traitement de plusieurs variables sérielles en tant qu'une variable pour une intention qui n'accepte pas de séries multiples (la proportion, par exemple). Dans ce cas, il devient impossible de ne pas assigner de fonction au groupe sériel.

Si les fonctions sélectionnées ne sont pas applicables aux données, une boîte de message apparaît pour en aviser l'usager. Le programme retourne ensuite à la boîte de choix des fonctions.

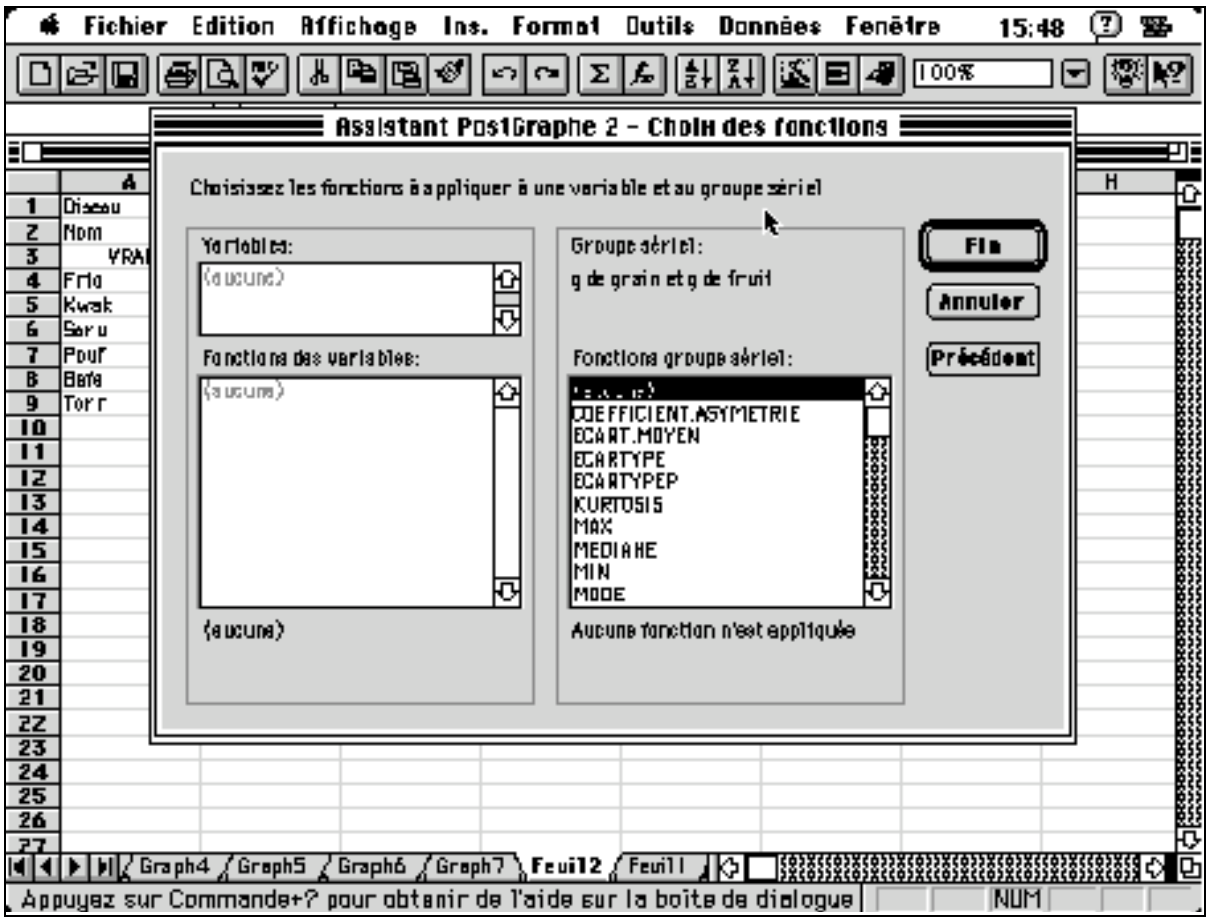

Figure 1.7: Choix des fonctions dans AP2

Après toutes ces étapes, le graphique est construit sur une feuille de graphique séparée. Un exemple de résultat est présenté à la Figure 1.8.

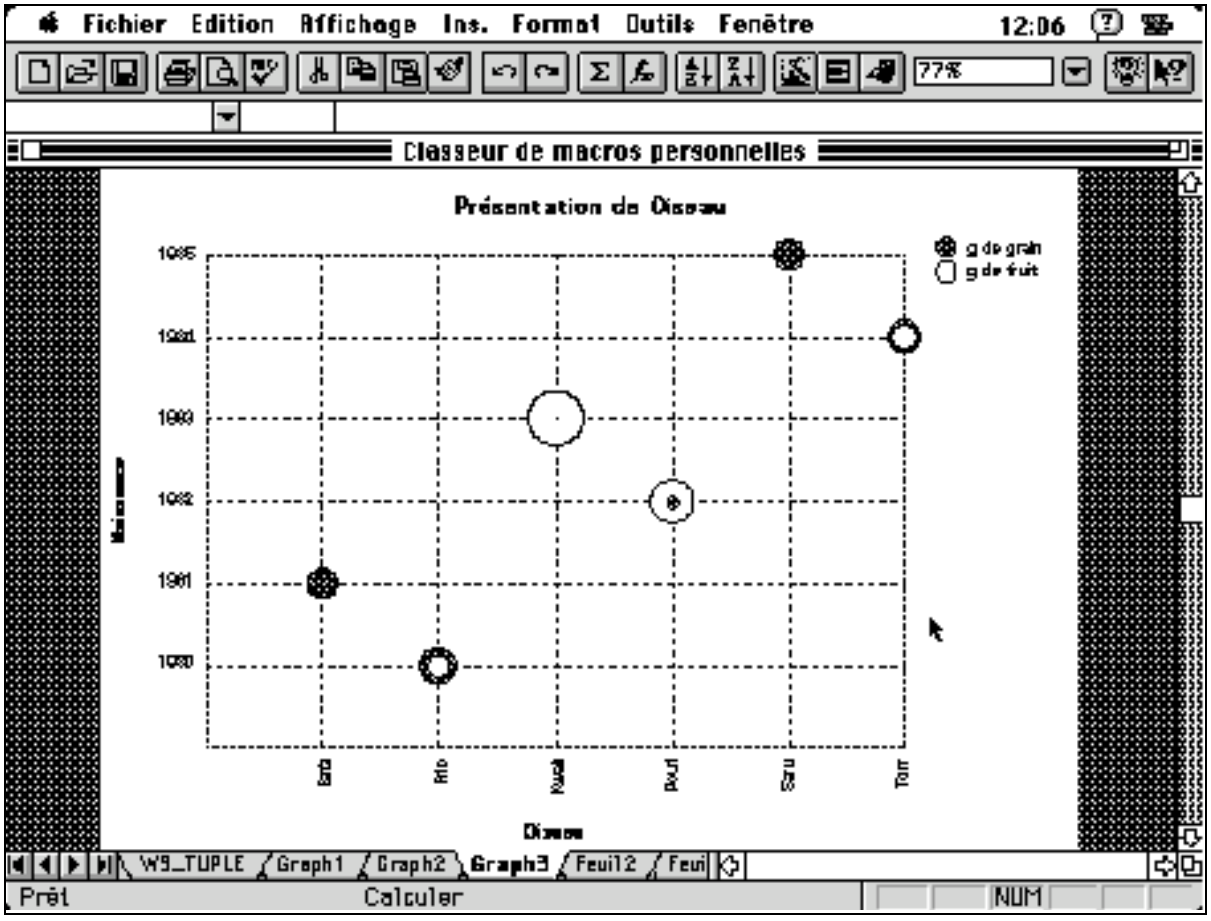

Figure 1.8: Résultat de AP2

Certaines notions utilisées dans AP2 peuvent paraître abstraites. La prochaine section effectue un survol des théories développées par Massimo Fasciano [Fasciano, 1996], et de la manière dont nous les avons modifiées pour subvenir à nos besoins.

2 Rappels théoriques

Nous résumons dans cette section les notions théoriques présentées dans la thèse de Massimo Fasciano et utilisées dans PostGraphe (PG), en précisant les modifications et les limitations que nous avons imposées pour les adapter à notre programme.

2.1 Propriété des données

Les données possèdent une multitude de propriétés. Une première classification possible est de les regrouper en **variables**. Dans notre exemple présenté au Tableau 1.1, les données *Pouf*, *Kwak*, *Saru*, *Bafa*, *Torr* et *Frio* sont associées à la variable *Oiseau*. Les données *1980, 1983, 1985, 1982, 1981, 1984* se rapportent à la variable *année*, et ainsi de suite.

Ces propriétés sont déterminées par l'usager, qui s'en sert pour organiser proprement les données. Cette organisation se fait la plupart du temps par rangées ou par colonnes.

2.1.1 Nature des variables

Chaque variable possède des propriétés collectivement connues sous le nom de **nature** dans la thèse de Fasciano. Celui-ci a répertorié six catégories de propriétés, reliées entre elles selon des liens d'héritage: l'organisation, le domaine, le format, les mesures, le temps et les entités spécifiques.

Pourtant, si ces propriétés sont importantes pour la rédaction d'un rapport statistique complet (figures et texte), elles sont moins nécessaires pour la génération simple de graphiques. De plus, nous désirons que l'usager agisse plutôt par instinct: ainsi, au lieu de lister les propriétés possibles des variables, nous avons répertorié plusieurs **classes** de variables (Tableau 2.1), qui contiennent des combinaisons de valeurs pour trois propriétés. Ces classes ont été créées pour qu'un usager puisse instinctivement classer ses variables. Aussi, nous avons tenté d'exprimer à travers notre liste les différentes combinaisons de propriétés que pouvait prendre une variable dans le modèle de Fasciano.

Propriétés des classes

Nom: nom de la classe

Organisation: la variable est soit **ordonnée**, soit non ordonnée. Cela permet de déterminer si les valeurs peuvent être triées et comparées.

Domaine: la variable est soit **énumérée**, soit **quantitative**, ou continue. Cela permet de déterminer si l'on peut appliquer des opérations mathématiques sur les valeurs, et si l'on doit utiliser des axes continus (section 4.2).

Tri: certaines classes contiennent une liste pré-ordonnée de valeurs (par exemple, les jours de la semaine). Le type de tri identifie la liste prédéfinie associé à la classe (voir section 4.4.3). Cela permet de déterminer la façon de trier les valeurs et de rajouter les valeurs manquantes dans les listes de valeurs.

Tri des classes

Numérique	Ce tri sert à ordonner des données numériques continues.
Alphabétique	Ce tri ordonne des étiquettes de façon alphabétique
Naturel	Ce tri ordonne des étiquettes numériques qui se suivent à l'unité.
Autres tris	Le tri portant le nom de sa classe indique que cette classe possède une liste propre et ordonnée de valeurs.

Classe/ Traduction	**Ordonnée**	**Énumérée**	**Description**	**Exemple**	**Tri**
Ordinal	X	X	Variable ordonnée triée au préalable	Notes musicales (do, ré, mi)	Aucun
Intervalle/ Interval	X	X	Intervalles numériques	[2, 1[, [3, 7]	Aucun
Réel/Real	X		Nombre réel	3.23, 345.0	Numérique
Entier/Integer	X		Nombre entier	..., -1, 0, 1,...	Numérique
Naturel/ Natural	X		Nombre naturel	0, 1, 2, 3,...	Numérique
Numéro/ Number		X	Étiquette numérique	Numéro de joueur	Numérique
Mesure/ Measure	X		Mesure continue	Poids, pression	Numérique
Position	X		Coordonnée continue	34°	Numérique
Date	X	X	Date	96/11/01	Numérique
Nom/Name		X	Appellation d'une entité ou personne	Charlie, O.N.U.	Alphabétique
Objet/Object		X	Appellation d'un objet inanimé	Table, chaise	Alphabétique
Pourcentage/ Percentage	X		Nombre exprimé en pourcentage	3%, 4.23%	Numérique
Territoire/ Territory		X	Appellation d'une entité géographique	Ontario, Montreal	Territoire/ Territory
Mois/Month	X	X	Mois	Janvier, Février	Mois/Month
Année/Year	X	X	Année exprimée en nombre	1997, 1982	Naturel
Dollar	X		Nombre exprimé en dollars	12.34$, 12$	Numérique
Rang/Rank	X	X	Rang (premier, deuxième) exprimé en nombre	1, 2, 3, 4	Naturel
Qualité/ Quality		X	Adjectif	Beau, intelligent	Alphabétique
Faveur/ Support		X	Opinion	Pour, contre, ne sais pas	Faveur/ Support

Temps/Time	X		Mesure du temps en base 10	12 (minutes), 2 (heures)	Numérique
Jour/Day	X	X	Jour de la semaine	Lundi, Mardi	Jour/Day
Note/Grade	X	X	Note scolaire	a+, b, c-	Note

Tableau 2.1: Classes des variables dans AP2

2.1.2 Dépendances entre les variables

Les variables d'un système sont souvent dépendantes entre elles. Ces dépendances sont définies en s'inspirant du modèle des bases de données relationnelles proposé par C. J. Date [Date, 1988]. Dans le contexte de ce modèle, le tableau contenant les données du système représente la **relation**, les colonnes de ce tableau se nomment **attributs** (ce sont en fait, les variables) et ses lignes portent le nom de **tuples**. Une **clé minimale** est définie comme étant un ensemble minimal d'attributs qui identifie chaque tuple de façon unique. En d'autres mots, en prenant chacune des variables tour à tour, on vérifie la relation par rapport aux valeurs de cette variable: dès qu'il est possible d'identifier chaque tuple selon la valeur de une ou d'une combinaison de plusieurs variables, cette ou ces variables forment une clé minimale pour la relation. On dira alors que les autres variables de la relation sont dépendantes de cette clé minimale. Une clé **non-minimale** est obtenue en ajoutant une des variables dépendantes à la clé minimale d'une relation. Un bon exemple de clé minimale dans notre exemple est la variable *Oiseau*.

La dépendance entre les variables est limitée dans notre programme, pour des raisons de simplicité. Les clés minimales ne peuvent contenir qu'une variable, ce qui nous permet d'utiliser des tableaux strictement unidimensionnels. Prenons un exemple du modèle de Massimo Fasciano. Pour deux villes, pour trois années, on a calculé le taux de chômage (Tableau 2.2: Relation avec clé à deux variables). Les données sont présentées en tuples de cette forme: (année, ville, taux). Le couple (année, ville) constitue une clé minimale de cette relation et les données pourraient être représentées par un tableau bidimensionnel (Tableau 2.3). Dans notre implantation, il faudrait, selon l'intention, construire un tableau différent, qui aurait une clé minimale simple, soit la ville (Tableau 2.4), soit l'année (Tableau 2.5). Cette structure est nécessaire car dans Excel lit les données soit en lignes, soit en colonnes. Dans notre implantation, tous les tableaux sont lus en colonnes. Ainsi, dans notre rapport, la clé de la relation fera référence à une variable. Cette variable ne doit pas avoir deux valeurs identiques dans la relation.

Année	Ville	Taux de chômage
1987	Toronto	12.2
1987	Montréal	5.2
1988	Toronto	13.4
1988	Montréal	6.1
1989	Toronto	12.3
1989	Montréal	5.3

Tableau 2.2: Relation avec clé à deux variables

Année Ville	1987 Taux de chômage	1988 Taux de chômage	1989 Taux de chômage
Toronto	12.2	13.4	12.3
Montréal	5.2	6.1	5.3

Tableau 2.3: Tableau bidimensionnel dans PG

Ville	Taux de chômage 1987	Taux de chômage 1988	Taux de chômage 1989
Toronto	12.2	13.4	12.3
Montréal	5.2	6.1	5.3

Tableau 2.4: Premier arrangement d'une clé à deux variables dans AP2

Année	Taux de chômage de Montréal	Taux de chômage de Toronto
1987	12.2	5.2
1988	13.4	6.1
1989	12.3	5.3

Tableau 2.5: Deuxième arrangement d'une clé à deux variables dans AP2

2.2 Intentions

Après avoir compilé et choisi les données, la chose la plus importante dans le choix d'un graphique est l'**intention** du rédacteur.

L'intention représente le but communicatif du rédacteur, c'est-à-dire le message que le rédacteur souhaite tirer des données et transmettre au lecteur. Dans le modèle de Fasciano, les intentions sont classifiées en cinq catégories élémentaires, auxquelles sont rajoutées certaines intentions modificatrices permettant de faire ressortir un aspect particulier d'une intention. Notre modèle ne traite pas les modificateurs. Certains modificateurs de PG sont devenus des

intentions à part entière, les autres ont été délaissés. Les correspondances entre les deux intentions sont présentées dans le Tableau 2.6.

Intention/traduction	**Appellation dans le modèle de Fasciano**
Présentation/ Presentation	Lecture
Comparaison/ Comparison	Comparaison
Décomposition/ Decomposition	Comparaison, avec modificateur décomposition
Évolution/ Evolution	Évolution, avec ou sans les modificateurs augmentation, diminution, stagnation, récapitulation
Répartion/ Distribution	Répartion
Proportion/ Proportion	Répartition, avec modificateur proportion

Tableau 2.6: Intentions dans AP2 et PG

Mais une plus grande différence entre PG et AP2 se situe au niveau du nombre d'intentions exprimées. L'usager de PG pouvait, à chaque utilisation, demander d'exprimer plusieurs intentions à partir d'une plage de donnée, ce qui menait à la génération de plusieurs graphiques ou d'un graphique qui exprimait correctement plusieurs intentions. À ce niveau, AP2 se rapproche plus de l'assistant graphique que du générateur de rapport statistique, car il ne permet de choisir qu'une intention à chaque utilisation. Ainsi, nous parlons dans notre implantation d'**intention simple**.

Chaque intention nécessite un ou deux **arguments**, c'est-à-dire des variables impliquées dans le message à transmettre. Le nombre d'arguments, dans le modèle de Fasciano, est appelé l'**arité**. Aussi, notons que puisque l'usager doit choisir l'intention, nous avons traduit chaque intention (Tableau 2.6).

Présentation

La présentation (lecture dans le modèle de Fasciano) montre les valeurs individuelles avec détail plutôt que la tendance globale. Cette intention ne prend qu'un argument et s'applique à n'importe quel type de donnée. Dans notre exemple, on pourrait présenter la consommation de grain chez les oiseaux. PG, dans certains cas, satisfait la présentation par un tableau. AP2 étant un assistant graphique, il satisfait chaque intention seulement en générant un graphique.

Évolution

Cette intention permet d'observer la variation d'une ou plusieurs variables à travers une variable temporelle. Elle nécessite donc deux arguments, l'un qui évolue en fonction de l'autre. Le deuxième argument doit être une clé de la relation et temporel (temps, date, année, mois, jour). Dans notre exemple, on pourrait montrer comment la consommation de fruits évolue en fonction de l'année de naissance.

Corrélation

La corrélation démontre l'existence d'une relation entre deux variables. Cette intention prend deux arguments quantitatifs, comme nous parlons ici de corrélation mathématique. Dans notre exemple, nous pourrions montrer une corrélation entre la quantité de grain consommée et la quantité de fruit consommée, comme cela a été fait à la Figure 1.1: Graphique montrant la corrélation.

Comparaison

Le but de la comparaison est de démontrer la position, ou le classement, de différentes valeurs entre elles. Cette intention prend deux arguments. On compare les valeurs du premier selon les valeurs du deuxième; le premier doit donc être ordonné et le deuxième, clé de la relation. Dans notre exemple, nous pourrions comparer la consommation de grain pour chaque oiseau, comme cela a été fait à la Figure 1.2. Dans PG, plusieurs modificateurs de cette intention ont de l'impact exclusivement sur le texte généré. Comme cette version ne génère pas de texte, nous délaisserons ces modificateurs.

Décomposition

La décomposition, dans PG, est un modificateur de la comparaison qui exprime les valeurs comparées en fractions de la somme des valeurs de la variable. Nous le considérons ici comme une intention particulière. Nous pourrions par exemple décomposer la consommation de grain parmi les oiseaux différents.

Répartition

La répartition divise le domaine d'une variable en intervalles ou valeurs pour une variable énumérée et compte pour chaque intervalle le nombre d'échantillons qui s'y trouvent. Cette intention prend deux arguments; le deuxième doit être clé de la relation. Nous pourrions dans notre exemple répartir la consommation de grain par rapport aux oiseaux. Notons que cette intention et la suivante sont les seules qui modifient les données originales avant de construire le graphique.

Proportion

La proportion, dans le modèle de Fasciano, est un modificateur de la répartition qui exprime le nombre d'échantillons en pourcentage du nombre total d'échantillons. Nous pourrions voir dans notre exemple les proportions des consommations de grain par rapport aux oiseaux.

2.3 Propriétés des données dans un graphique

Les données dans un tableau possèdent des propriétés spécifiques, mais peuvent être traitées de multiples façons lorsqu'on les place dans des graphiques. Deux choses à considérer lorsqu'on construit un graphique: l'existence de plusieurs séries et la structure de données.

Ces propriétés sont déterminées une fois que l'usager a fait son choix de données et d'intention. Même si elles sont dépendantes des choix de l'usager, celui-ci n'a aucune emprise directe dessus.

2.3.1 Série

La **série** n'est pas définie explicitement par Fasciano [Fasciano, 1996]. Notre implantation nous oblige à la définir pour faciliter le traitement des données par Excel.

Une série représente une relation, c'est à dire un ensemble de tuples complets permettant de porter sur un graphique un ensemble de points.

Dans Excel, tout graphique possède une série ou plus. Chaque série peut être représentée selon un type graphique différent (section 2.4). En pratique, on utilise le même type pour toutes les séries, ou au plus une combinaison de quelques types compatibles entre eux.

Dans notre modèle, les séries sont identifiées par une **variable sérielle**, c'est-à-dire une variable propre à chaque série. Dans notre exemple, les relations (oiseau, g de grain) et (oiseau, g de fruit) sont deux séries que l'on pourrait porter sur un même graphique. Les variables sérielles sont *g de grain* et *g de fruit*.

Dans notre implantation, nous supposons que les variables sérielles font partie de la même classe. Cela permet aux séries d'être portées dans un même graphique, avec des axes communs et des types graphiques semblables. Les graphiques contenant plusieurs types différents ne sont pas utilisés dans notre projet par souci de clarté.

2.3.2 Structure de données

Nous employons l'expression **structure de données** pour désigner à la fois le nombre de variables par tuple et l'existence de séries multiples dans les données brutes.

Le Tableau 2.7 résume la notation utilisée dans notre projet. Nous utilisons ces symboles pour spécifier la structure de donnée pour le choix et la construction d'un graphique. Par souci de simplicité, nous nous limitons à un maximum de quatre variables par tuple. La variable générique X est associée à la première position, Y est associée à la deuxième position, et ainsi de suite pour Z et T. Chaque structure doit au moins avoir deux variables (X et Y).

Pour bien comprendre l'information de ce tableau, regardons deux entrées en détail. La structure de donnée [X, Y] indique que les données sont constituées de deux variables formant un tuple d'arité 2 (i.e. 2 variables). Si la structure de donnée est [X,YS], cela nous indique que les données contiennent une variable X **non sérielle**, mais aussi plusieurs variables Y **variables** pour ce même X. Il est donc possible de construire plusieurs séries de tuples en réutilisant les valeurs de X, mais avec les valeurs d'un Y différent. Nous appelons un groupe de variables sérielles **groupe sériel**; celui-ci est exprimé comme une variable normale. Toujours par souci de simplicité, nous nous limitons à un maximum de cinq variables par groupe sériel. Aussi, remarquons que dans chaque structure, le groupe sériel occupe la dernière position.

Symbole	Nombre de variables par tuple	Nombre de séries
[X, Y]	2	1
[X, YS]	2	Nombre de Y
[X, Y, Z]	3	1
[X, Y, ZS]	3	Nombre de Z
[X, Y, Z, T]	4	1

Tableau 2.7: Structures de données dans AP2

2.4 Types de graphiques

Toutes les notions vues précédemment mènent au choix du type de graphique. Une fois son intention spécifiée, l'usager n'a aucune emprise directe sur ce choix, puisque cela est la problématique même de notre projet.

Les types de graphiques utilisés dans notre projet (comme dans PG) sont tous bidimensionnels. On associe généralement les deux dimensions à des axes orthogonaux servant de repères dans le plan. Pour notre projet, nous associons l'axe horizontal et la clé de la relation d'un graphique à la variable X, et l'axe vertical avec la variable Y. Les variables Z et T sont associées aux caractéristiques rétiniennes des **points** (section 4.2).

On retrouve dans PG plusieurs types de graphiques dont nous résumons ici les caractéristiques, en présentant leurs limites et possibilités. Certaines caractéristiques ont été rajoutées dans notre projet; elles exploitent les possibilités graphiques d'Excel.

Points

Ce graphique représente un point typiquement par une petite figure géométrique (e.g. un cercle, une triangle, un carré ...) située en X et Y. L'avantage de ce genre de graphique est qu'il n'impose pas de relation prédéterminée entre les variables; il permet justement de trouver si de telles relations existent.

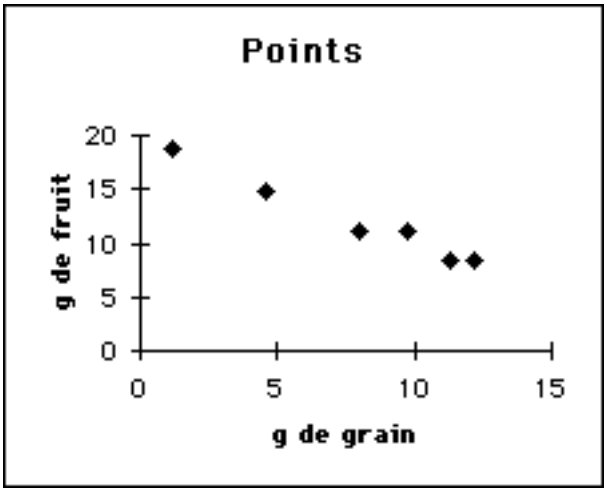

Figure 2.1: Graphique en points

Conséquemment, ce type de graphique est utilisé pour chercher la corrélation entre une variable X et une variable Y. On peut aussi encoder les variables supplémentaires d'un tuple en attribuant aux points des caractéristiques rétiniennes (forme, couleur, grosseur) au besoin ou en étiquetant lorsque possible. C'est ce qui rend ce graphique souvent le graphique par défaut lorsqu'on veut utiliser plusieurs variables. Pour mieux transmettre des intentions spécifiques, on peut relier les points du graphique (évolution, corrélation) ou rajouter un quadrillage qui permet de mieux les situer (comparaison). Les détails de ces instruments graphiques sont donnés à la section 4.2.

Courbe

Ce graphique omet généralement les marqueurs de points individuels qui sont situés dans le plan selon la variable X et la variable Y. Plutôt, la caractéristique importante de ce graphique est la courbe qui relie les points entre eux.

Ce type de graphique est prédéfini dans Excel, mais nous ne l'utilisons pas. Nous préférons utiliser un graphique en points reliés, qui permet un axe horizontal continu. L'aspect du graphique est le même, seulement le transfert des données sur un axe en sera simplifié.

Comme son axe principal est horizontal, la courbe présente très bien l'évolution temporelle de valeurs. L'avantage de la courbe par rapport aux colonnes est qu'il est plus efficace de montrer plusieurs séries de données avec les courbes multiples qu'avec les colonnes multiples.

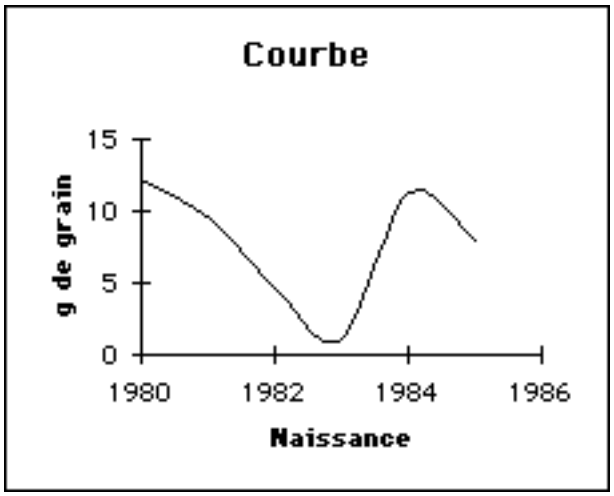

Figure 2.2: Graphique en courbe

Colonnes

Ce type de graphique désigne chaque tuple par un rectangle vertical situé en X et de hauteur Y. La variable X est énumérée et son domaine se limite à douze valeurs tandis que la variable Y est quantitative ou, du moins, ordonnée, et doit être fonction de X; par conséquent, X doit être une clé minimale de la relation.

Même si ce type de graphique est idéal pour exprimer l'évolution de valeurs dans le temps, nous préférons utiliser la courbe pour cette intention. Par contre, on utilisera des colonnes multiples plutôt que les barres pour comparer plusieurs variables entre elles. On peut aussi montrer des répartitions avec une forme particulière du graphique en colonnes, l'**histogramme**, dans lequel les colonnes simples sont collées les unes aux autres.

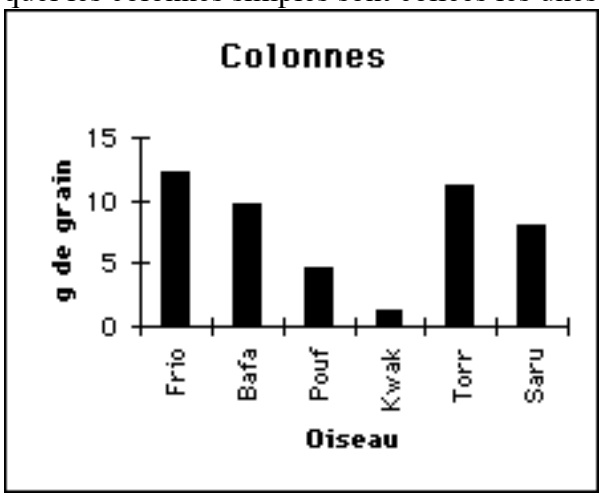

Figure 2.3: Graphique en colonnes

Barres

Nous considérons ce type de graphique comme un graphique "colonnes horizontales". En fait, les axes X et Y sont transposés, c'est-à-dire que l'axe X est vertical et l'axe Y est horizontal. Cela correspond à l'approche adoptée par Excel vis-à-vis ce graphique.

La représentation d'un tuple est réalisée par un rectangle horizontal situé en X et de longueur horizontale Y. La variable X est énumérée tandis que la variable Y est ordonnée. De plus, Y doit être fonction de X. Pour des raisons esthétiques, on emploie rarement ce graphique avec plus de deux variables; on préfère les colonnes dans de tels cas.

On utilise ce graphique surtout pour les comparaisons, puisque les barres orientées horizontalement facilitent l'évaluation des longueurs (et donc leur comparaison). Pour des raisons esthétiques, il est préférable de trier les valeurs de façon à placer la plus longue barre au centre focal, soit le bas du graphique. On évite ainsi un graphique ayant une allure "déséquilibrée".

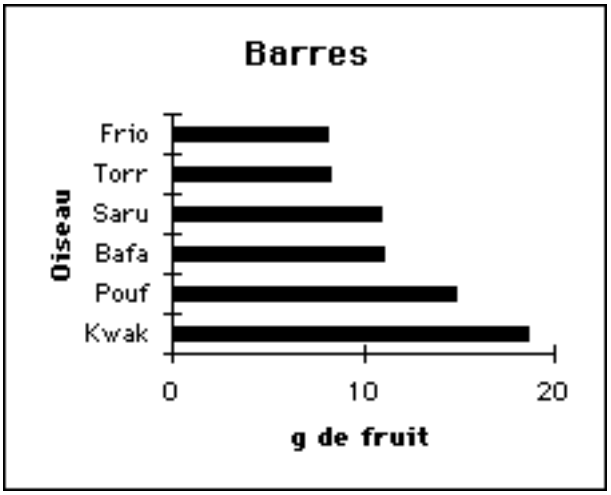

Figure 2.4: Graphique en barres

Tarte

Ce graphique attribue à chaque tuple un secteur de cercle dont l'angle varie en fonction de la valeur de la variable Y du point. Il ne comporte pas d'axes puisqu'on n'utilise pas de caractéristiques spatiales pour exprimer les variables. Il nécessite que la variable X soit énumérée (pour les étiquettes de secteurs) et que la variable Y soit quantitative et fonction de X. Pour assurer une lisibilité acceptable, on limite habituellement le nombre de secteurs à une valeur entre six (6) et huit (8), regroupant les valeurs supplémentaires dans un secteur nommé, par exemple, "Autres".

On utilise surtout la tarte pour exprimer la répartition de fractions par rapport à un tout ou bien pour comparer ces fractions entre elles. Elle ne permet pas de montrer plusieurs séries.

Notons que Excel offre d'autres types de graphiques prédéfinis, tels Aires, Radar et Anneaux. Puisque PG n'en a pas étudié les propriétés et la pertinence, AP2 les ignore également.

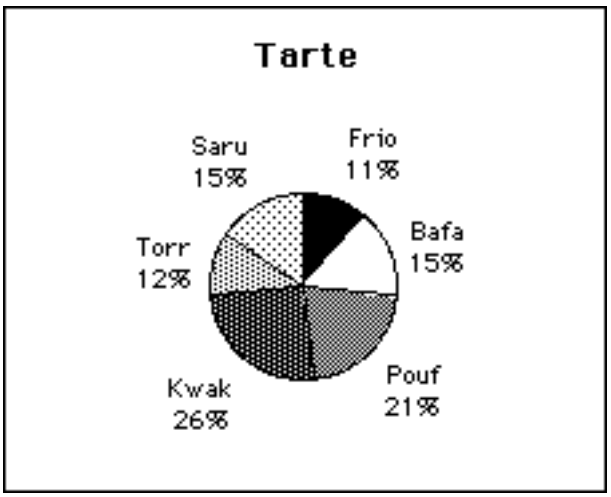

Figure 2.5: Graphique en tarte

2.5 En résumé

Cette section a d'abord expliqué les facteurs permettant d'effectuer le choix d'un graphique: les propriétés des données (classes), les intentions du rédacteur et les arguments de cette intention, et la structure de données utilisée, qui identifie le nombre de séries. Nous avons identifié les choix pour chaque paramètre et les choix laissés à l'usager.

Nous avons aussi identifié les modifications effectuées sur le modèle original de PG: d'abord, les propriétés des données sont répertoriées différemment et leurs relations de dépendance ont été simplifiées. Ensuite, certains choix d'intentions ont été modifiés ou délaissés. Et surtout, nous avons souligné que AP2 ne traite que des intentions simples.

Finalement, nous avons décrit cinq types de graphiques de base qui sont: les colonnes, les barres, la courbe, les points et la tarte. Chacun de ces graphiques comporte des contraintes quant aux données qui peuvent y figurer, et certains sont préférables à d'autres pour exprimer une intention particulière.

Malgré les modifications effectuées, ces notions proviennent en grande partie de la recherche de Massimo Fasciano. La section suivante passe aux notions techniques, qui, elles, n'ont plus rien en commun avec celles de PG, puisque PG et AP2 ont été implantés dans des environnements différents.

3 Rappels Techniques

Cette partie du rapport traite des éléments des environnements d'implantation et d'utilisation. Elle ne décrit pas exhaustivement ces environnements, mais souligne les outils essentiels à notre projet.

3.1 Excel

Dans cette section, nous discutons de l'application Excel pour mieux comprendre cet environnement de travail et pour voir comment il nous a permis de réaliser notre projet.

3.1.1 Interface usager

Excel adopte l'organisation suivante: toutes les données fournies par l'utilisateur occupent des **feuilles** qui sont à leur tour regroupées dans des **classeurs**.

Classeur

Au sommet de la hiérarchie de cette organisation se trouve le classeur, qui sert à regrouper un ensemble de feuilles utiles à un projet donné. Le classeur constitue la plus petite structure logique qu'Excel peut sauver sur disque; il n'est pas possible de sauver des feuilles individuelles.

Feuilles

Les feuilles d'Excel se divisent en quatre groupes: la **feuille de calcul**, la **feuille graphique**, le **module** et la **feuille de dialogue**.

Feuille de calcul

La feuille de calcul est constituée de cellules utilisées pour le stockage et la manipulation des données. L'application Excel repose avant tout sur ce type de feuille puisque les données à traiter y sont contenues.

Feuille graphique

La feuille graphique sert de support à l'affichage de graphiques et de dessins. Nous utilisons souvent ce type de feuille dans notre projet car les graphiques que nous créons occupent ce type de feuille.

Module

Le module joue deux rôles dans Excel: il constitue un environnement de développement du code Visual Basic et agit comme interface entre le code source et l'application Excel.

Pour le développement, le module permet l'édition, le débogage et le stockage du code source Visual Basic. Ce code source sera tantôt une **procédure**, tantôt une **fonction**. La procédure peut être utilisée directement pour contrôler l'application Excel ou elle peut servir à construire d'autres procédures ou fonctions. La fonction sert uniquement à calculer une valeur

dans une feuille de calcul, une procédure ou une autre fonction; elle ne peut pas contrôler les actions d'Excel.

Comme interface, le module permet à Excel de repérer et d'exécuter le code source Visual Basic. En d'autres mots, lorsque nous désirons utiliser une procédure ou une fonction écrite en Visual Basic, Excel cherchera d'abord le code correspondant dans les modules à sa disposition puis il l'exécutera à partir du module approprié en lisant les instructions qui s'y trouvent. Puisque nous réalisons notre projet avec des procédures et des fonctions écrites en Visual Basic, ce type de feuille nous est indispensable.

Feuille de dialogue

La feuille de dialogue permet de construire une boîte de dialogue servant d'interface spécialisée pour interagir avec l'usager au cours de l'exécution d'une tâche.

Dans notre projet, nous utilisons ce genre de feuille pour permettre à l'usager de choisir une valeur parmi un ensemble de valeurs qui varient selon le contexte. Nous en décrirons les instruments plus loin.

3.1.2 Interface de programmation

Pour le programmeur intéressé à contrôler l'application à l'aide de procédures spécialisées, Excel se décompose selon un modèle orienté-objet avec certaines particularités que nous décrivons dans les paragraphes suivants.

Objet

Toutes les composantes fonctionnelles qui constituent le logiciel "Excel" sont en fait des objets, c'est-à-dire des entités autonomes contenant des **données** et des **propriétés**. Lorsqu'il est nécessaire de référer à une composante d'Excel dans un programme Visual Basic, on utilise l'objet correspondant à cette composante.

Par exemple, `Application`, `Workbook`, `Worksheet`, `Chart`, et `Range` sont des exemples de classes d'objets représentant respectivement un classeur, une feuille de calcul, une feuille graphique et une **plage** de cellules. Une plage de cellules est constituée d'une seule ou de plusieurs cellules contiguës ou non. Notons que la classe d'objet représentant une seule cellule n'existe pas; on doit plutôt utiliser un objet de la classe `Range`.

Chaque objet contient des données (propriétés) et des opérations (méthodes) propres que l'on peut appeler ainsi: *`nom_objet.propriété`* ou *`nom_objet.méthode`*.

Propriétés

Dans Excel, certaines propriétés peuvent être modifiées directement pour affecter l'allure ou le comportement d'un objet; ce mode d'accès est dit *Read/Write*. D'autres propriétés ne servent qu'à indiquer l'état d'un aspect particulier d'un objet; leur mode d'accès est *Read-Only* puisqu'on peut seulement lire leur valeur courante. Un exemple de propriété accessible en mode Read/Write est `HasTitle` de l'objet `Chart`, qui représente un graphique; `HasTitle` prendra la valeur `True` si le graphique affiche son titre à l'écran, et `False` dans le cas contraire. Par contre, la propriété `Width`, qui représente la largeur du graphique, ne peut qu'être lue. Il n'est donc pas possible de changer la largeur d'un objet `Chart` en modifiant directement la valeur de sa propriété `Width`.

Méthodes

Les méthodes des objets d'Excel permettent la spécification de leurs paramètres de façon **conventionnelle** (i.e. selon la position des valeurs dans la liste d'arguments) ou **nommée** (i.e. sous forme d'affectation avec le symbole ":=").

Parfois, certains paramètres sont facultatifs, ou même incompatibles entre eux. Pour les paramètres à omettre (dans l'exemple suivant, c'est le deuxième paramètre), on n'aura qu'à ne rien inscrire à leur position dans la liste d'arguments s'il s'agit de la syntaxe "conventionnelle":

```
objet.méthode(p1, , p3, p4)
```

On peut également ne pas les inclure lors de l'appel avec la syntaxe "nommée":

```
objet.méthode(param1:=p1, param3:=p3, param4:=p4)
```

Dans Excel, l'usage des parenthèses pour délimiter la liste d'arguments est régi par une règle: si la valeur ou l'objet retourné par une méthode doit être récupéré (pour une affectation, pour une expression constituée de plusieurs objets, etc.), les parenthèses sont requises, sinon il faut les omettre. On peut aussi omettre les parenthèses lorsque la méthode n'a pas d'arguments. Pour notre projet, nous récupérons toujours la valeur ou l'objet retourné par une méthode. Nous affectons à la variable *voidVal* les valeurs que nous n'utilisons pas réellement, et à la variable *voidObj* les objets non-utilisés. Ces deux variables ont été créées pour notre projet; elles ne sont pas des variables spéciales pré-définies dans Excel.

Conteneur

Dans Excel, un objet peut être constitué d'autres objets de classes différentes: on appelle un objet qui en contient un autre un **conteneur** pour cet objet. Par exemple, `Workbook` est une classe d'objet qui contient des objets de la classe `Worksheet`. À son tour, un objet de la classe `Worksheet` sert de conteneur pour les objets de la classe `Range`, et ainsi de suite.

Les conteneurs permettent de spécifier précisément à quel objet nous voulons référer, car ils indiquent en quelque sorte le "trajet" qu'il faut parcourir dans l'hiérarchie des objets d'Excel pour parvenir à l'objet désiré. Ce trajet débute toujours à l'application Excel elle-même, c'est-à-dire l'objet `Application`.

Pour accéder à un objet d'un conteneur, on appelle le conteneur, suivi d'un point et du nom de l'objet. On peut ainsi passer par tous les étages de la hierarchie pour accéder à un objet. Par exemple, supposons que nous voulions accéder au "Graphique 2" du "Classeur B". Nous l'appelons ainsi:

```
Application.ClasseurB.Graphique2
```

Collection

Excel utilise un genre d'objet spécial, la collection, pour faciliter la gestion des multiples objets d'une même classe (ou de classes similaires) présents dans un conteneur. Une collection est en quelque sorte équivalente à une liste de pointeurs vers chaque objet: à partir de cette liste, on peut accéder soit à un objet individuel, soit à l'ensemble des objets d'un seul coup.

Pour agir sur la collection entière, on n'a qu'à appeler la méthode désirée. Par exemple, la collection `Workbooks` de l'objet `Application` permet l'accès à tous les classeurs ouverts dans une session Excel. L'énoncé suivant permet de fermer tous les classeurs d'un seul coup:

```
voidVal = Application.Workbooks.Close
```

Habituellement, le nom d'une collection est la forme plurielle du nom de la classe d'objet qu'elle contient. A titre d'exemple, `Charts` est la collection d'objets qui appartiennent à la classe `Chart`. Remarquons qu'un objet peut appartenir à plusieurs collections simultanément, car `Chart`, l'objet représentant une feuille graphique, se trouve à la fois dans la collection `Charts` (les feuilles graphiques) et la collection `Sheets` (tous les types de feuilles). On s'aperçoit alors que les collections existent surtout pour simplifier le traitement de plusieurs objets d'un même genre, comme par exemple dans le cas d'énoncés itératifs qui effectuent un traitement similaire sur chaque item d'une collection.

Pour référer à un objet individuel, on utilise la méthode `Item(index)` qui renvoie un objet unique de la collection, en spécifiant en paramètre son index. Cet index peut être soit un nombre entier, soit une chaîne de caractères. Le nombre représente la position du pointeur de l'item recherché dans la collection et est égal ou supérieur à 1, car toutes les collections d'Excel sont indexées à partir de la valeur 1. La chaîne de caractères représente le nom particulier de l'objet, soit la valeur de la propriété `Name` de cet objet.

Si comme dans notre exemple précédent deux classeurs sont ouverts, nommément "ClasseurA" et "ClasseurB", nous pouvons adresser spécifiquement "ClasseurB" avec la syntaxe suivante:

```
Set voidObj = Application.Workbooks.Item("ClasseurB")
```

Pour alléger la syntaxe, toutefois, Excel considère `item(index)` la méthode par défaut des objets. Conséquemment, on peut omettre le point et le mot `item` lors de l'appel d'un membre d'une collection. Ainsi, l'appel de "Graphique2" à partir de "ClasseurB" peut s'effectuer ainsi:

```
Set voidObj = _
Application.Workbooks("ClasseurB").Sheets("Graphique2")
```

Le symbole '_' utilisé dans le code indique à Excel que la première ligne se poursuit sur la ligne suivante. On peut ainsi répartir un énoncé sur plusieurs lignes en ajoutant ce symbole à la fin de chaque ligne précédant la dernière.

Un autre raccourci permis est d'omettre le nom du conteneur par défaut, qu'Excel déduit selon le contexte approprié. Ainsi, on peut omettre par exemple le mot `Application` pour l'appel d'un objet `Workbook`.

La création d'une instance de classe (i.e. un objet) se fait à partir de la collection correspondante avec la méthode `Add`. Par exemple, l'ajout d'une nouvelle feuille graphique dans "ClasseurB" se fait à partir de sa collection `Charts` de la façon suivante:

```
Set voidObj = Workbooks("ClasseurB").Charts.Add
```

Objet actif

Pendant le déroulement d'une session Excel, il est toujours possible d'identifier le **contexte de travail**. Nous entendons par contexte de travail l'ensemble des objets qui peuvent être affectés par la prochaine action exécutée soit par l'usager lui-même ou via un programme

Visual Basic en cours d'exécution. Ces objets sont dits **actifs**, et l'objet `Application` possède des méthodes qui retournent justement une référence à ces objets importants: `ActiveWorkbook`, `ActiveSheet`, `ActiveChart`, et ainsi de suite.

Certaines méthodes renvoyant un objet actif n'utilisent pas le mot "Active"; c'est le cas notamment de `Selection` qui retourne un ou plusieurs objets (de n'importe quelle classe) sélectionnés explicitement par l'usager à l'aide du curseur, ou à la suite d'une instruction au cours de l'exécution d'une procédure Visual Basic. Dans le cadre de notre travail, l'objet retourné par `Selection` est typiquement une plage de cellules.

Pour revenir à notre exemple, si "ClasseurB" est le classeur actif de cette session, c'est-à-dire le classeur dont les feuilles sont présentement affichées à l'écran, la simplification suivante devient possible:

```
ActiveWorkbook.Sheets("Graphique2")
```

Et si en plus d'être le classeur actif, la feuille active de "ClasseurB" est "Graphique2", l'expression atteint son maximum de simplicité:

```
ActiveSheet
```

3.2 Visual Basic pour Applications

Dans cette section, nous décrivons notre usage de l'environnement Visual Basic pour Applications (VBA).

3.2.1 Introduction

Originalement conçu comme un langage de macro-instructions pour une base de données, Visual Basic s'est d'abord transformé en un langage de programmation orienté-objet, pour ensuite être désigné comme le langage de contrôle pour diverses applications permettant cette possibilité.

Plus précisément, la version du langage utilisée pour l'automatisation des applications s'appelle Visual Basic pour Applications: elle retient de la version indépendante un noyau de base qui permet le traitement des objets particuliers des applications contrôlées. Il est important de noter que Visual Basic pour Applications nous fournit les outils pour contrôler ces objets mais ne permet pas de définir de nouvelles classes d'objets. Cette possibilité n'existe qu'avec la version "complète" de Visual Basic.

VBA s'apparente à un langage de programmation structuré comme Pascal ou C. On y retrouve toutes les composantes "classiques": d'une part les déclarations de variables, de procédures et de fonctions, et d'autre part les énoncés d'affectation, de sélection et d'itération. Les paragraphes suivants reprennent ces aspects du langage.

3.2.2 Déclarations

Les identificateurs de variables, de constantes, de procédures et de fonctions doivent être des chaînes de caractères alphanumériques commençant par une lettre. Les lettres minuscules et majuscules ne sont pas distinctes; en d'autres mots, *`voidVal`*, *`VOIDVAL`* et *`voidval`* sont considérés par Excel comme étant le même identificateur.

Variables

La déclaration des variables en Visual Basic n'est pas requise. Pourtant, la possibilité d'exiger la déclaration des variables avant leur usage existe. Nous utilisons ce style de programmation dans ce projet puisqu'il minimise le risque d'erreurs et améliore la compréhension du code. La commande **`Option Explicit`** placée au début d'un module force la déclaration des variables.

Pour déclarer une variable, on utilisera une des quatre formes possibles selon la portée que l'on désire accorder à cette variable. Le Tableau 3.1 résume les cas possibles, en passant de la portée la plus globale à la plus locale.

Déclaration	**Déclarée dans**	**Portée de la variable**
`Public` *`identificateur`*	Module	Globale pour toute procédure dans tout module dans tout classeur
`Public` *`identificateur`*	Module, avec **`Option Private Module`**	Globale pour toute procédure dans tout module dans un classeur unique
`Dim` *`identificateur`* ou **`Private`** *`identificateur`*	Module	Globale pour les procédures dans un module unique
`Dim` *`identificateur`*	Procédure	Locale à une procédure unique

Tableau 3.1: Déclaration de variables

Il est possible de spécifier le type de données que peut contenir une variable, quoique ceci ne soit pas absolument nécessaire: une variable déclarée sans spécification de type est du type Variant par défaut. Le Tableau 3.2 présente les types fondamentaux possibles.

Type de donnée	Valeurs possibles
`Boolean`	`True`, `False`
`Integer`	[-32 768, 32 767], entières
`Long`	[-2 147 484 648, 2 147 483 647], entières
`Single`	[-3.402 823 E28, -1.401 298 E-45] et [1.401 298 E-45, 3.402 823 E28], réelles
`Double`	[-1.797 693 134 862 32 E308, -4.940 656 458 412 47 E -324] et [4.940 656 458 412 47 E -324, 1.797 693 134 862 32 E308], entières
`Currency`	[-922 337 203 685 477.580 8, 922 337 203 685 477.580 7], réelles, 4 décimales
`Date`	1 janvier 1000 au 31 décembre 9999, incluant les heures, les minutes et les secondes
`String`	Chaîne de caractères, de longueur 0 à environ 2 milliard
`Object`	Référence (i.e. pointeur) vers un objet quelconque
`Variant`	N'importe quel type (déterminé selon le contexte)

Tableau 3.2: Types de données

Il est possible de déclarer plusieurs variables avec une seule déclaration, mais chaque nom de variable doit être accompagné de son type, sinon le type Variant sera accordé. Pour illustrer, considérons les deux exemples suivants:

```
Dim X, Y As Integer
Dim X As Integer, Y As Integer
```

Dans le premier énoncé, *X* est du type `Variant` et *Y* est du type `Integer`, tandis que dans le deuxième, les deux variables sont de type `Integer`.

Constantes

La déclaration de constantes est identique à la déclaration de variables, mais le mot clé **`Const`** remplace **`Dim`**, comme suit:

```
Const identificateur = expression
```

On peut aussi spécifier explicitement le type de la constante en ajoutant le mot clé **`As`** entre l'identificateur et le type, comme pour la déclaration de variable. En supposant que l'on veuille définir la constante *OUI* comme type booléen possédant la valeur `True`, on utiliserait l'énoncé suivant:

```
Const OUI As Boolean = True
```

Procédures

La déclaration de procédures est seulement possible au niveau du module, pas à l'intérieur d'une autre procédure ou d'une fonction. Elle s'effectue selon la syntaxe suivante:

```
Sub identificateur(p1 As type1, p2 As type2, ..., pN As typeN)
      (déclarations)
      (énoncés)
End Sub
```

Les spécifications de type des paramètres peuvent être omises; dans tel cas, les paramètres seront attribués le type `Variant`.

Pour appeler une procédure à l'intérieur d'une autre procédure, le mot clé **`Call`** peut être utilisé comme suit:

```
Call procédure(arguments)
```

La liste d'arguments doit alors être entre parenthèses. On doit toutefois omettre ces parenthèses si nous n'utilisons pas le mot clé **`Call`**.

La procédure peut modifier des valeurs ou objets, mais ne prend pas une valeur elle-même. On ne peut pas l'utiliser dans une expression.

Fonctions

La déclaration de fonctions s'effectue selon la syntaxe suivante:

```
Function identificateur(p1 As type1, ..., pN As typeN) As typeF
      (déclarations)
      (énoncés)
identificateur = valeur_de_retour
End Function
```

Les consignes pour la déclaration de procédures s'appliquent également pour les fonctions. De plus, la valeur de retour de la fonction peut aussi être typée selon les mêmes conventions.

3.2.3 Énoncés

Affectation

L'affectation de valeurs se fait avec symbole "=" pour tous les types de variables:

`Variable` = 2 * 3.14159

L'affection de variable de type `Object` requiert le mot clé **`Set`** devant l'énoncé, sinon Excel signalera une erreur à l'exécution:

```
Set objet = ActiveChart
```

Sélection

Les décisions s'effectuent avec une forme légèrement différente du If ... Then ... Else classique. La voici:

```
If condition1 Then
      énoncé(s) 1
ElseIf condition2
      énoncé(s) 2
(...)
ElseIf condition(N-1)
      énoncé(s)(N-1)
Else
      énoncé(s) N
End If
```

Les alternatives **Else** et **ElseIf** peuvent être omises. La sélection parmi les choix multiples a aussi son énoncé propre:

```
Select Case expression
Case valeur1
      énoncé(s) 1
Case valeur2
      énoncé(s) 2
(...)
Case valeurN
      énoncé(s) N
Case Else
      énoncé(s) alternatifs
End Select
```

Là aussi, **Case Else** peut être omis.

Itération

Pour boucler pendant une condition vraie, la structure suivante est proposée:

```
Do While condition_vraie
      enoncé(s)
Loop
```

La construction suivante est semblable mais elle boucle pendant une condition fausse jusqu'à ce que celle-ci devienne vraie:

```
Do Until condition_vraie
      enoncé(s)
Loop
```

On peut quitter la boucle en y plaçant la ligne **Exit Loop**

On peut balayer avec un index dont les bornes sont connues:

```
For i = a To b
      enoncé(s)
Next i
```

On peut aussi bien balayer une collection entière dont on ignore le nombre d'éléments:

```
For Each X In Charts
      énoncé(s)
Next X
```

Autres énoncés

Lorsqu'un bloc de code utilise les propriétés et les méthodes d'un même objet, il peut être fastidieux de toujours répéter les expressions *objet.propriété* et *objet.méthode*. Visual Basic propose une construction qui permet de définir les bornes d'un bloc dans lequel

les propriétés et les méthodes d'un objet peuvent être accédées plus directement et simplement avec la notation *.propriété* et *.méthode*:

```
With objet
      énoncé(s)
End With
```

Constantes prédéfinies

Pour simplifier la tâche de programmation, plusieurs constantes numériques sont prédéfinies. La compagnie Microsoft a adoptée la nomenclature suivante, permettant ainsi d'identifier le contexte d'utilisation de ces constantes: les constantes intrinsèques à l'application Excel sont préfixées des lettres "xl", tandis que les constantes significatives pour Visual Basic ont comme préfixe les lettres "vb".

Par exemple, il est beaucoup plus simple d'utiliser `xlColumn` au lieu de sa valeur correspondante 3 pour spécifier le type d'un graphique. La compréhension du code Visual Basic est d'autant plus claire.

3.3 En Résumé

Nous avons dans cette section présenté le "squelette" de notre environnement de travail, c'est à dire les modalités de base d'Excel et VBA. Dans la prochaine section, nous verrons plus en détail les objet et méthodes les plus utilisés dans notre projet.

4 Objets et méthodes

Dans cette section, nous discutons des objets utilisés dans notre implantation et des méthodes utilisées pour les modifier et les manipuler.

4.1 Plages de cellules

La plage de cellule est un objet de base dans Excel, car c'est l'unité de stockage élémentaire de l'environnement, à l'instar des listes dans Prolog. La plage correspond à l'objet `Range`.

Notons que pour notre rapport, nous ne mentionnons pour les méthodes que les arguments qui importent dans notre traitement. Nous utilisons dans notre programme la syntaxe "nommée" (section 3.1.2), ainsi les arguments omis prennent les valeurs par défaut. Ces valeurs, nous y avons vu, sont toujours valides.

4.1.1 Identification

Pour identifier une cellule d'une plage de données, on utilise le plus souvent la méthode `Cells(RowIndex, ColumnIndex)`. Les deux arguments correspondent à l'index de la rangée et de la colonne à l'intérieur de la plage. Pour accéder à une colonne ou à une rangée, on utilise respectivement les méthodes `Columns(Index)` et `Rows(Index)`.

Pour accéder à une plage dont la position est relative à la première, on utilise la méthode `Offset(Rowoffset, Columnoffset)`. `Rowoffset` et `Columnoffset` correspondent au nombre de cellules de déplacement vers la droite et vers le bas; la valeur 0 reste à la plage originale.

La méthode `Resize(Rowsize,Columnsize)` redimensionne la plage traitée. `Rowsize` indique le nombre de rangées et `Columnsize` le nombre de colonnes à partir de la première cellule de la plage originale.

4.1.2 Tri

On trie une plage de données en utilisant la méthode `Sort(Key1, Order1, Header, OrderCustom, Orientation)`. `Key1` indique la colonne à trier le premier et `Order1`, l'ordre du tri, soit `xlAscending` (croissant) ou `xlDescending` (décroissant). `Header` prend la valeur `xlNo` si la première colonne ou la première ligne est à trier; sinon, `xlYes`. `OrderCustom` indique la liste prédéfinie dans Excel à utiliser (voir section 4.4.3). Pour `Orientation`, `xlTopToBottom` trie les rangées; `xlLeftToRight` trie les colonnes.

4.1.3 Recherche

Pour rechercher une valeur spécifique dans une plage, on utilise la méthode `Find(What, LookIn, LookAt, SearchOrder)`. `What` indique la valeur à trouver. Dans notre implantation, `LookIn` prend la valeur `xlValue` et `LookAt`, la valeur `xlWhole`, ce qui signifie que la recherche doit trouver la valeur intégralement dans les valeurs (non les formules) des cellules. `SearchOrder` prend la valeur `xlByRows` s'il faut chercher ligne par ligne; sinon, il prend la valeur `xlByColumns`.

4.1.4 Format

Le format d'une cellule est le mode d'affichage du nombre qu'elle contient, que ce soit en pourcentage, en nombre réel, en date ou en dollars. Cette propriété est appelée par `Format`. Lorsqu'il faut appliquer un format "neutre", c'est-à-dire sans caractéristiques particulières, nous employons le format "général".

4.1.5 Affectation

Les cellules d'une plage contiennent soit rien du tout, soit une formule, soit une valeur. La valeur est affichée à l'écran, et est le résultat de la formule s'il y en a une. Pour retrouver ou affecter une valeur dans une cellule, on utilise la propriété `Value`. Si l'on affecte une valeur à une cellule, la formule anciennement contenue est effacée.

Si l'on veut affecter la formule d'une plage ou d'une cellule, on utilise la propriété `Formula`. Souvent, les formules affectées feront référence à d'autre plages de cellule. Ces formules doivent contenir l'adresse de la plage concernée. Pour obtenir cette adresse, on utilise la méthode `Address(RowAbsolute, ColumnAbsolute, External)`. `RowAbsolute` est vrai si l'adresse renvoie la rangée en coordonnées absolues. `ColumnAbsolute` est vrai si l'adresse renvoie la colonne en coordonnées absolues. `External` est vrai si on doit inclure le nom de la feuille dans l'adresse.

Les coordonnées absolues font référence à une cellule particulière; elles ne changent pas si l'on copie la formule ailleurs. Les coordonnées relatives font référence à une cellule selon sa position relativement à la cellule qui contient la formule; elles changent si l'on copie la formule ailleurs. Par exemple, si la cellule A1 a une formule qui renvoie à la cellule B1 en coordonnées absolues et que l'on copie la formule en A2, la formule reste la même. Si les coordonnées sont relatives, la formule renvoie à la cellule B2.

Si l'on veut tout simplement copier une plage dans une autre on utilise la méthode `Copy(Destination)`. `Destination` est la plage où placer la copie, si les deux sont de la même grandeur; sinon, `Destination` est simplement la première cellule de cette plage. Cette méthode copie les formules, sinon les valeurs. Notons que si l'on modifie les valeurs dans la plage originale, les valeurs ne changent pas dans la copie.

4.2 Graphiques

Le graphique est situé dans notre implantation sur une feuille graphique appelée `Chart`. Dans notre implantation, toutes les propriétés et méthodes sont celles de l'objet directement renvoyé par la méthode `Charts.Add`.

Les figures Figure 4.1 et Figure 4.2 identifient les principales composantes d'un graphique. Ces composantes sont décrites dans la section qui suit (avec entre parenthèses les classes d'objets correspondantes).

Certains objets contiennent des coordonnées (`Top` et `Left`) calculées en points permettant de les positionner sur la feuille. Ces objet, souvent ont des dimensions (`Height` et `Width`) permettant de régler également leur taille.

Zone de graphique (`ChartArea`)

Cet objet désigne le rectangle qui englobe tous les autres éléments composant le graphique. Aucun élément du graphique ne peut dépasser cette zone. Cet objet a des coordonnées et dimensions non modifiables.

Zone de traçage (`PlotArea`)

Cet objet désigne la surface, typiquement délimitée par les axes, qui contient les marqueurs de points, les courbes ainsi que les étiquettes de valeurs. Nous pouvons modifier la couleur de cette surface en utilisant la propriété `interior.color`. Ses dimensions comprennent les étiquettes des valeurs et ses coordonnées sont variables.

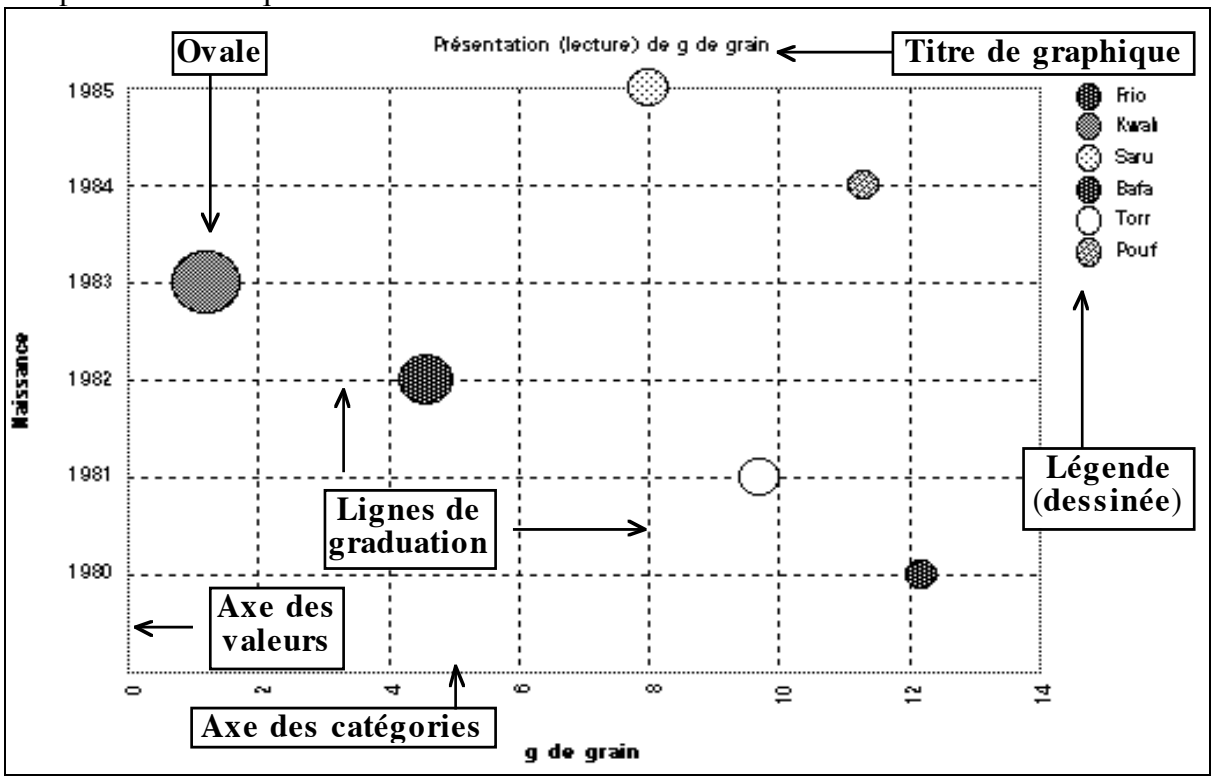

Figure 4.1: Éléments d'un graphique (1)

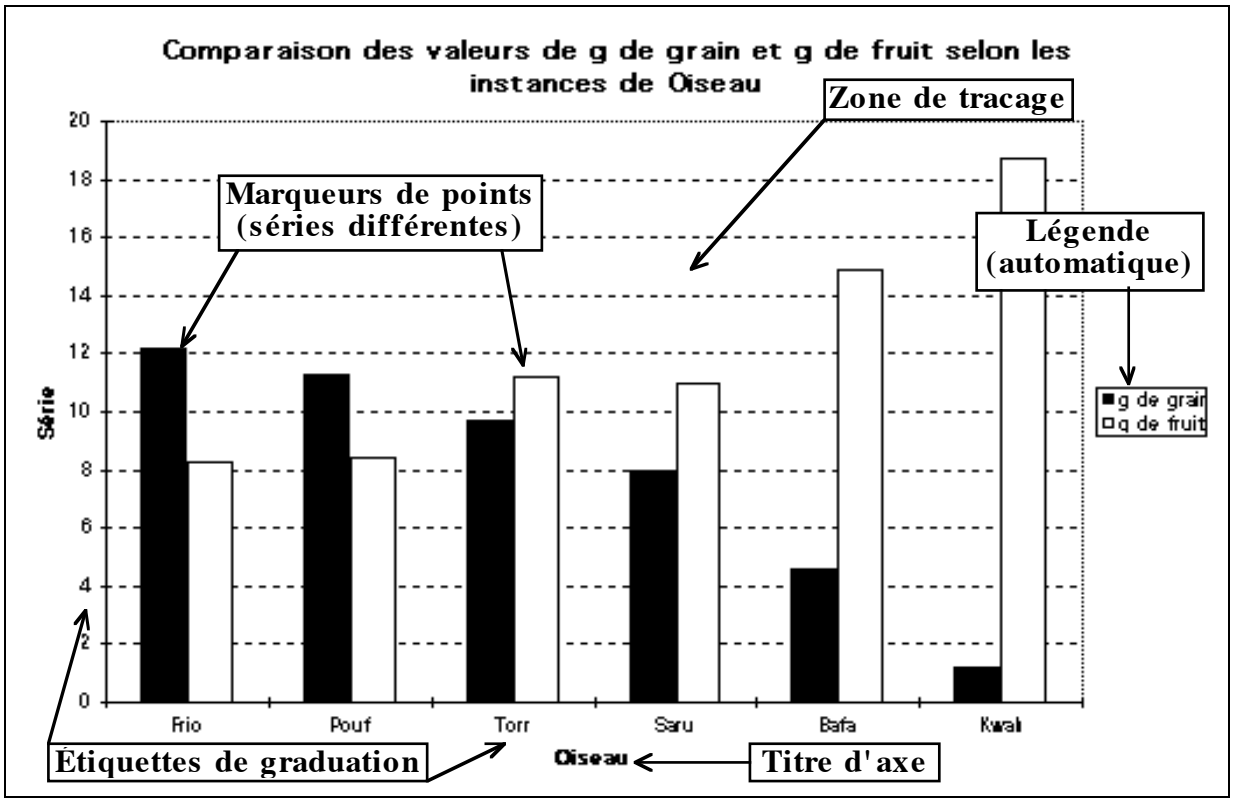

Figure 4.2: Éléments d'un graphique (2)

Série (`SeriesCollection(Index)`)

Cet objet représente une série constituant le graphique. Une feuille graphique contient une collection de séries (l'objet `SeriesCollection`). C'est à travers cette collection que l'on accède à une série particulière sur le graphique. Chaque série possède une propriété `Name` qui détermine comment la légende y fera référence.

Titre de graphique (`ChartTitle`)

Cet objet désigne le titre du graphique, qui apparaît normalement entre le haut de la zone de traçage et le haut de la zone de graphique. On peut changer sa valeur en utilisant sa propriété `Text`.

Légende (`Legend`)

Cet objet désigne la légende, un petit tableau simple qui répertorie les caractéristiques rétiniennes associées aux séries. Excel s'attend à ce que la légende soit située entre la droite de la zone de traçage et la droite de la zone de graphique; il lui réserve automatiquement un espace approprié dans cette région en déplaçant la zone de traçage vers la gauche. Nous ne nous servons de la légende dans notre implantation qu'avec les séries multiples en Y. Dans les autres cas, nous dessinons une légende avec des boîtes de texte.

Axes (`Axes(Index)`)

Cet objet désigne un axe du graphique. Chacun des deux axes, associé à une variable particulière, agit comme repère pour permettre de relever la valeur de la variable correspondante pour un point donné du graphique.

Dans Excel, l'axe X se nomme **axe de catégories** et l'axe Y porte le nom d'**axe de valeur**. On réfère à l'un de ces axes en utilisant la collection `Axes` de l'objet `Chart`. Plutôt que de spécifier des index numériques pour cette collection, on peut utiliser les constantes prédéfinies `xlCategory` et `xlValue` qui retournent automatiquement l'axe correspondant.

Normalement, l'axe de catégories (l'axe X) est horizontal et l'axe de valeurs (l'axe Y) est vertical, mais l'inverse est vrai dans le cas d'un graphique de type "Barres" (`xlBar`). De plus, les axes sont totalement absents dans un graphique de type "Tarte" (`xlPie`). Dans un graphique "Points" (`xlXYScatter`), l'axe X se comporte comme un axe de valeurs. Dans les cas où ce graphique remplace un autre type de graphique (par exemple, lorsque la structure a un arité de quatre), nous traitons ses axes comme ceux de ce type. Les paragraphes suivants décrivent les distinctions fondamentales entre les deux axes.

Axe de catégories

Excel suppose toujours que l'axe de catégories est associé à une variable énumérée . Ces valeurs sont désignées comme étant les **catégories**. De plus, Excel n'effectue aucun tri des valeurs de la variable lorsqu'il crée le graphique; il ne fait que les disposer une à la suite de l'autre sur l'axe.

Autre inconvénient, Excel ne disposera pas sur cet axe les valeurs d'une variable quantitative comme si l'axe représentait un intervalle de nombres réels; plutôt, il les répartit uniformément le long de l'axe en partant de l'origine et progresse uniformément vers la droite, peu importe leur valeur numérique.

Ainsi, nous devons dans notre implantation voir à ce que cet axe ne contienne que des valeurs énumérées, préalablement triées s'il y a lieu.

Axe de valeurs

Pour l'axe de valeurs, Excel suppose que la variable associée est quantitative et que son domaine est un intervalle de nombres réels. Il calcule les bornes de l'intervalle à représenter sur l'axe et détermine l'espacement approprié entre les graduations d'axe en fonction de ce domaine et de la dimension physique du graphique. Les points portés sur le graphique seront situés par rapport à l'axe comme si celui-ci représentait un intervalle de nombres réels.

Si les valeurs de cette variable ne sont pas quantitatives, la valeur zéro sera attribuée à toutes les valeurs, produisant ainsi un graphique complètement inutile. AP2 remédie à cela en indexant les valeurs d'une variable énumérée et en alignant le libellé des différentes valeurs le long de l'axe.

L'axe de valeurs possède des propriétés exclusives qui affectent entre autres l'intervalle réel et les graduations. Par exemple, les propriétés `MaximumScale` et `MinimumScale` permettent de lire ou de modifier les bornes supérieure et inférieure respectivement de l'intervalle réel représenté par l'axe.

Les différences entre l'axe de catégories et l'axe de valeurs que nous venons de décrire nous obligent à porter une attention particulière à la nature des variables X et Y associées aux

axes. Pour certains types de graphiques que nous décrivons prochainement, nous devons transformer les données originales pour parvenir à créer le graphique voulu.

La classe `Axis` contient de nombreux objets qu'il nous faut considérer. Par exemple, l'objet `AxisTitle` qui représente le titre de l'axe. Cet objet peut être redimensionné, déplacé, et son texte peut être modifié par la propriété `Text`.

Les étiquettes de graduation ne peuvent être modifiées, mais l'on peut tout simplement les effacer en affectant la valeur fausse à la propriété `HasTickLabels`. Aussi, on peut savoir le nombre d'étiquettes avec la propriété `NumTickLabels` et leur espacement avec la propriété `TickMarkSpacing`.

Dans certains cas, aussi, il faudra ajouter du quadrillage au graphique. Pour ajouter des lignes perpendiculairement à un axe, il faut assigner la valeur vrai à la propriété `HasMajorGridLines`.

Point (`Points(Index)`)

La collection `Points` n'est pas une propriété de `Chart`, mais plutôt une propriété d'une série. Chaque point, dans notre implantation, représente un tuple du tableau original. Chaque point d'une série est indexé et accessible individuellement.

Nous pouvons modifier la forme de ce point avec la propriété MarkerStyle, qui peut prendre de nombreuses valeurs (`xlSquare`, `xlDiamond`, `xlTriangle`, `xlX`, `xlStar`, `xlDot`, `xlDash`, `xlCircle`, `xlPlus`).

De plus, chaque objet `Point` contient un objet `DataLabel` qui nous permet de modifier l'étiquette de valeur de ce point. Pour afficher cette étiquette, il suffit d'affecter la valeur vraie à la propriété `HasDataLabel`. Initialement placé à côté du point, il est possible dans certains cas de déplacer ces étiquettes pour construire une légende artificielle.

Dans le case d'un graphique tarte, la propriété `Explosion` du point permet de modifier la distance (en points) d'une pointe de tarte au centre de la tarte pour faire ressortir cette pointe.

Boîte de texte (`TextBoxes(index)`)

Les boîtes, dans notre implantation, servent surtout à construire des légendes ou à étiqueter les axes indexés. On les ajoute grâce à la propriété `TextBoxes.Add(Left, Top, Width, Height)`.

Ces boîtes peuvent être dimensionnées en fonction de leur contenu grâce à la propriété `AutoSize`, à laquelle doit être assignée la valeur `True`. On peut également modifier leur orientation avec la méthode `Orientation`, qui peut être `xlHorizontal` (horizontal) ou `xlUpward` (lettres verticales).

Ovale (`Ovals(index)`)

Cet objet est un dessin que l'on crée avec la méthode `Ovals.Add(Top, Left, Height, Width)`. Il est essentiel à certains graphiques qui contiennent des structures de données de trois variables et plus, car certaines informations peuvent être encodées dans la taille et la couleur des ovales qui représentent les points.

Pour qu'un point soit représenté par un ovale préalablement dessiné, il suffit de l'effacer (`Oval.Delete`) et de le coller sur le point (`Points(Index).Paste`).

Format (`AutoFormat(Gallery, Format)`)

Le formatage est une méthode. Il permet d'assigner un type (`Gallery`) et une allure spécifique (`Format`) à l'ensemble du graphique. Les possibilités correspondent aux choix offerts par l'assistant graphique d'Excel.

Les choix de graphiques que nous utilisons sont `xlBar` (barres), `xlColumns` (colonnes), `xlPie` (tarte), `xlXYScatter` (points et courbes).

4.3 Feuilles de dialogue

Les feuilles de dialogues jouent un grand rôle dans AP2 puisque l'accessibilité de l'interface est primordiale. Plusieurs outils sont essentiels pour l'entrée de données et la sélection des choix.

Certaines boîtes de dialogues sont déjà comprises dans l'environnement Excel. Notre programme utilise la méthode `Application.Inputbox(Prompt, Title, Default, Left, Top, Type)`. Cette méthode affiche une boîte de dialogue affichant le message `Prompt`, le titre `Title`, l'entrée par défaut `Default`, les coordonnées `Left` et `Top` et le type d'entrée obligatoire `Type`.Les autres boîtes de dialogue ont été créées "à la main"; un exemple est présenté à la Figure 4.3.

Souvent, au cours de l'utilisation des boîtes de dialogue, la sélection d'un élément aura un effet sur un autre élément. Pour permettre cela, il existe une option dans la boîte des outils de la boîte de dialogue. Cette option affecte une Macro, c'est-à-dire une procédure sans paramètres, à l'objet, ce qui lui permet d'avoir un effet sur d'autres objets lorsqu'il est utilisé.

Boutons (`Buttons(`*`Name`*`)`)

Sur toutes les boîtes de dialogue, il existe un bouton "Ok" qui permet de fermer la boîte en validant les données; ce bouton est le plus souvent le bouton par défaut, soit celui appuyé lorsque l'usager appuie sur la touche de retour. Le bouton "Annuler" ferme la boîte en ignorant les modifications opérées par l'usager. Il est possible de créer d'autres boutons et de modifier leurs effets en y affectant des macros.

Étiquette (`Labels(`*`Name`*`)`)

La seule utilité de l'étiquette est le texte qu'elle affiche. On peut le modifier avec la propriété `Text`.

Boîte d'entrée (`EditBoxes(`*`Name`*`)`)

Cette boîte permet à l'usager d'y entrer du texte. Le programme peut également le modifier grâce à l'objet `Text`.

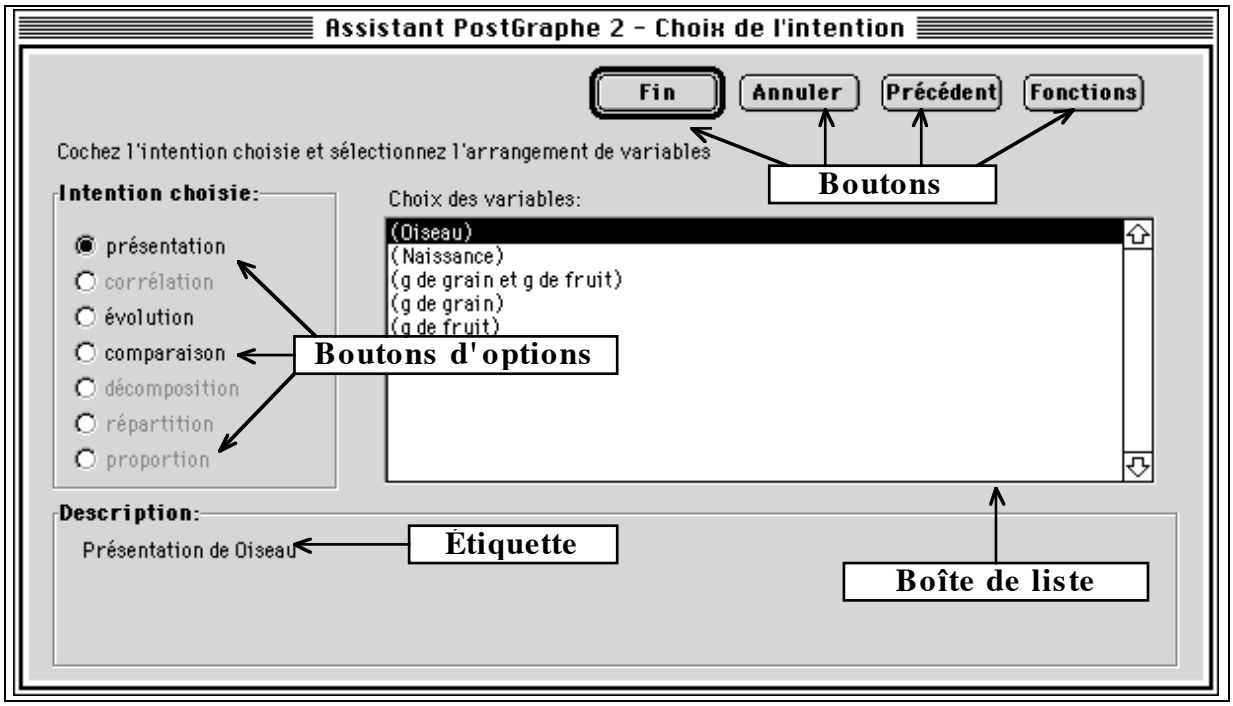

Figure 4.3: Éléments de la boîte de dialogue

Boutons d'option (`OptionButtons(`*`Name`*`)`)

Un seul bouton peut être sélectionné par l'usager parmi tous ceux présents sur la feuille. Cela permet à l'usager de choisir une intention, par exemple. Pour empêcher une option d'être choisie, on assigne la valeur `False` à sa propriété `Enabled`.

Boîte de liste (`ListBoxes(`*`Name`*`)`)

La boîte de liste permet de sélectionner une entrée parmi plusieurs possibilités. Chaque entrée est indexée. Pour accéder à une entrée, on utilise l'objet `List(Index)`. La propriété `ListCount` donne le nombre d'entrée de la boîte et la propriété `ListIndex` donne l'index de l'élément sélectionné. Pour vider la boîte, on utilise la méthode `RemoveAllItems`.

4.4 Méthodes reliées à l'application

Les objets et méthodes décrits ici sont rattachés à l'application elle-même et ne sont pas liés hierarchiquement à d'autres objets. Nous nous en servons pour la plupart pour faire des ajustements à l'environnement d'Excel permettant alors la bonne utilisation d'AP2.

4.4.1 Boîte d'outils

La boîte d'outils, identifiée par l'objet `Toolbars(Index)`, contient des boutons auxquels sont reliés des procédures. Les boutons sont identifiés par le sous-objet ToolbarButtons(Index). Pour ajouter un bouton à la collection, on utilise la méthode `Add(Button, OnAction)`. `Button` fait référence au numéro du bouton tel que défini dans la

liste des boutons d'Excel et `OnAction` est le libellé de la procédure liée au bouton. Pour supprimer une boîte, on utilise la méthode `Delete`.

4.4.2 Menus

La barre de menus est la série de listes déroulantes située en haut de l'écran. Chaque type de feuille possède une barre de menus particulière. Celle que nous modifions correspond à l'objet `MenuBars(xlWorksheet)`. Chaque menu est accessible par l'objet `Menus(Name)`. Chaque ligne d'un menu est accessible par l'objet `MenuItems(Caption)`, Caption étant le libellé de l'instruction. Pour ajouter un item à un menu, on utilise la méthode `Add(Caption, OnAction)`. `OnAction` est le libellé de la procédure liée à l'item du menu. Pour supprimer cet item, on utilise la méthode `Delete`.

4.4.3 Listes prédéfinies

Les listes prédéfinies sont les listes utilisées pour le tri des valeurs particulières (mois, jours, etc.). Ces listes prédéfinies sont utilisées dans notre programme pour certaines classes de données. La méthode `GetCustomListNum(ListArray)` retourne le numéro de la liste prédéfinie contenant les éléments de la table `ListArray` si elle existe; sinon, elle retourne la valeur Faux. Pour ajouter un liste, on utilise la méthode `AddCustomList(ListArray)`.

4.5 En Résumé

Nous avons décrit dans cette section comment sont utilisés les objets d'Excel et de VBA dans notre programme. Nous avons d'abord vu comment manipuler des plages de cellules, ce qui est primordial pour le traitement d'information. Ensuite, nous avons vu comment manipuler des graphiques pour obtenir un résultat final optimal. Enfin, nous avons décrit les décomposantes de la boîte de dialogue.

Maintenant que les environnements ont été explorés, il semble adéquat de décrire en détail l'utilisation et la façon dont agit AP2, ce que nous faisons dans la prochaine section.

5 Implantation de l'assistant PostGraphe 2

Dans la première sous-section, nous décrivons les outils particuliers que nous avons utilisés pour traiter les données. Ensuite, nous décrivons plus en détail le fonctionnement interne du programme au niveau des modules, procédures et fonctions.

5.1 Outils de travail

À part les modules et les boîtes de dialogue, AP2 fait usage de feuilles de calcul invisibles à l'usager: les feuilles de travail et les tableaux prédéfinis.

5.1.1 Feuilles de travail

Notre modèle utilise des **feuilles de travail**, des feuilles de calcul où sont placées plusieurs valeurs et tableaux temporaires qui serviront au traitement des données. Elles remplacent les **listes** utilisées en Prolog pour l'implantation de PG. Ainsi, lorsque AP2 doit traiter des tableaux complexes et variables d'information, il stocke les données sur des feuilles où le tri et la recherche sont des méthodes prédéfinies par VBA.

`WS_WORK` est la feuille de travail principale. Toutes les propriétés des variables, les structures de données, les descriptions des groupes sériels y sont construits pendant le programme. Cette feuille est interne au programme, donc l'usager ne peux pas la consulter. Une fois le graphique construit, toutes ses données peuvent être effacées.

`WS_TUPLE` est la feuille où sont reproduits les tableaux sélectionnés par l'usager et les tableaux de données transformées qui serviront à la construction du graphique. Les valeurs de ces tableaux correspondent directement aux valeurs contenues dans la sélection originale; si ces valeurs changent, les valeurs de cette feuille changeront aussi pour adapter, autant que possible, les valeurs du tableau aux changements. Comme cette feuille contient les données directement reliées à la construction du graphique, le programme la crée pour chaque classeur qui utilise AP2. Elle est donc extérieure au programme et visible pour l'usager. Si l'usager en modifie les données, les graphiques en seront directement affectés.

5.1.2 Tableaux prédéfinis

Ces tableaux contiennent toutes les notions théoriques décrites à la section 2. Les données contenus dans ces tableaux ne changent pas au cours du programme. Les noms des feuilles sont précisés entre parenthèses. Les tableaux eux-mêmes sont des plages de cellules identifiés par la fonction `Range(`*`Nom_du_tableau`*`)`.

Tableau des contraintes sur les intentions (`WS_CTAB`)

Ce tableau, nommé `RNG_CTAB` sert à cerner les arrangements de variables compatibles avec les différentes intentions. Il permet d'empêcher le choix d'arrangements de variables incompatibles. Il est propre à AP2 puisque PG laissant à l'usager le soin de ne pas faire des choix erronés. Un extrait est présenté dans le Tableau 5.1.

La première ligne contient les noms des différentes intentions. Chaque intention comprend trois colonnes: une pour les structures de données, une pour le premier argument, une pour le deuxième (sauf l'intention de présentation).

La quatrième cellule de la première colonne contient le nombre de structures de données acceptables pour l'intention. Si ce nombre est 0, toutes les structures sont acceptables (ceci est aussi vrai pour les deux autres colonnes); sinon, les structures acceptables sont énumérées dans les cellules au-dessous. Les structures acceptables ont été déterminées en examinant les choix offerts par PG.

corrélation		
	FAUX	FAUX
	VRAI	VRAI
4	14	14
[X, Y]	Réel	Réel
[X, Y, Z]	Entier	Entier
[X, Y, ZS]	Naturel	Naturel
[X, Y, Z, T]	Mesure	Mesure
	Position	Position
	Pourcentage	Pourcentage
	Dollar	Dollar
	Temps	Temps
	Real	Real
	Integer	Integer
	Natural	Natural
	Measure	Measure
	Percentage	Percentage
	Time	Time

Tableau 5.1: Extrait du tableau de contraintes sur les intentions

La quatrième cellule des colonnes des arguments indique le nombre de classes possibles pour cet argument, suivi de l'énumération de ces classes. Remarquons que cette énumération comprend les choix en anglais et en français. La deuxième cellule indique VRAI si l'argument doit être candidat de clé. La troisième cellule indique VRAI si l'argument ne peut pas être un groupe sériel. Les choix réflètent les natures des intentions telles que décrites dans la section 2.2.

Tableau des fonctions (`WS_FTAB`)

Ce tableau contient les fonctions applicables aux variables et groupes sériels ainsi que leurs descriptions. Il est utilisé dans la boîte de dialogue de la troisième étape. Il contient deux parties, l'une anglaise et l'autre française. Le Tableau 5.2 en contient un extrait.

COS	Cosinus	MIN	Valeur minimale
COSH	Cosinus hyperbolique	MOD	Valeur la plus fréquente

Tableau 5.2: Extrait du tableau des fonctions

La première colonne contient les noms des fonctions applicables aux variables non sérielles et la deuxième colonne, leurs descriptions; cette partie est nommée `RNG_FN` et `RNG_FN_ANG` pour les langues françaises et anglaises. La troisième colonne contient les noms des fonctions applicables aux groupes sériels et la quatrième colonne, leurs description; cette partie est nommée `RNG_FNSERIE` et `RNG_FNSERIE_ANG` pour les langues françaises et anglaises.

Tableaux des types des variables et des tris (`WS_TTAB`)

Le premier tableau, dont un extrait est présenté dans le Tableau 5.3, est nommé `RNG_TTAB` et est utilisé pour récupérer les renseignements relatifs à la nature d'une variable telle que vue à la section 2.1.1. Son équivalent en anglais est nommé `RNG_TTAB_ANG`. La première colonne de ces tableaux contient le nom des classes. La deuxième colonne contient une valeur booléenne qui indique si une variable de cette classe est ordonnée ou non. La troisième colonne contient une valeur booléenne qui indique si la variable est énumérée ou non (quantitative). La dernière colonne contient le nom du tri à utiliser pour ordonner la variable.

Nom	FAUX	VRAI	Alphabétique
Objet	FAUX	VRAI	Alphabétique
Pourcentage	VRAI	FAUX	Numérique

Tableau 5.3: Extrait du tableau des classes des variables

Le deuxième tableau, dont un extrait est présenté dans le Tableau 5.4, est nommé `RNG_LTAB` et contient les listes de valeurs utilisées pour les tris, en forme de colonne. Les listes pour les deux langues sont contenues dans ce tableau. La première ligne contient le nom de la liste de tri. La deuxième contient le numéro de la liste prédéfinie. La troisième contient le nombre de valeurs contenues dans la liste, et les lignes suivantes contient les valeurs triées de la liste en ordre croissant.

Faveur	Jour
6	3
3	7
Pour	Lundi
Contre	Mardi
Ne sais pas	Mercredi
	Jeudi
	Vendredi
	Samedi
	Dimanche

Tableau 5.4: Extrait du tableau des tris

La feuille contient également la cellule indiquant la langue de l'assistant, nommée `RNG_LANGUAGE`. Cette cellule contient soit le mot "Français", soit le mot "Anglais", dépendemment de la langue choisie par l'usager.

Tableau des choix de graphique (`WS_ITAB`)

Ce tableau est parcouru par AP2 pour obtenir le nom d'un type de graphique que l'on nomme **graphique candidat**. Notre tableau, dont un extrait est présenté dans le Tableau 5.5, correspond à une version simplifiée de la table d'inférence du planificateur dans PG; certains choix ont été délaissés pour respecter les intentions simples et pour éliminer la génération de tableaux.

évolution	[X, Y]	Y	X	courbe
	[X, Y, Z]	Y	X	points étiquetés
	[X, Y, Z]	Y	X	points gris
	[X, Y, Z]	Y	X	points aire
	[X, Y, Z]	Y	X	points forme
	[X, YS]	Y	X	multi-courbes

Tableau 5.5: Extrait du tableau des choix de graphiques

Le graphique candidat est sélectionné en fonction de deux indices qui correspondent aux deux premières colonnes de ce tableau. Le premier index est l'intention de l'usager, le second est la structure de données. A partir de ces deux informations, l'assistant PostGraphe est en mesure de sélectionner des graphiques qui *potentiellement* peuvent exprimer l'intention désirée avec les données en entrée.

Les deux colonnes suivantes servent à indiquer la position du premier et du second argument de l'intention dans les tuples qui constitueront le graphique. La deuxième colonne est ignorée si l'intention n'a qu'un seul argument. Notons qu'il n'y pas d'intention avec plus de deux arguments dans notre modèle (ou dans celui de Fasciano). Nous expliquerons en détail le fonctionnement de ces marqueurs de position dans quelques instants.

La cinquième colonne contient le nom du graphique candidat.

Le processus de sélection va de haut en bas, et les graphiques sont placés en ordre décroissant de qualité. Ce tableau assure que chaque graphique considéré soit compatible avec l'intention et la structure de données. Cependant, pour assurer que les classes des variables puissent être respectées, il nous faut un autre tableau.

Tableau des spécifications du graphique (WS_GTAB)

Ce tableau, dont un extrait est présenté au Tableau 5.6, est nommé RNG_GTAB et contient les contraintes sur les arguments de chaque type de graphique. La première colonne contient les noms des types de graphique, et les colonnes suivantes contiennent les restrictions sur les variables X et Y (ainsi que Z et T selon le graphique). Le rôle de ce tableau est d'indiquer, après avoir choisi un type de graphique selon l'intention et la structure de donnée, si ce type s'applique aux variables considérées et à leurs propriétés.

Chaque variable dans le graphique est soumise à cinq contraintes. La première ligne indique si la variable doit être ordonnée ("ordonnée") ou non ("non ordonnée"). La deuxième ligne indique si la variable doit être énumérée ("énumérée") ou quantitative ("quantitative"). La troisième ligne indique VRAI si la variable doit être clé de la relation. Pour ces paramètres, s'il est indiqué "(aucune)", alors il n'y a aucune contrainte d'appliquée.

tarte triée	Ordonné	(aucune)
	Énuméré	Quantitatif
	VRAI	
	1	-1
	-1	-1

Tableau 5.6: Extrait du tableau de spécifications des graphiques

Les deux lignes suivantes indiquent les nombres minimums et maximums de valeurs que doivent prendre l'argument, s'il est énuméré. Si l'un de ces paramètres est -1, alors il n'y a aucune contrainte d'appliquée.

Tableaux des intentions et des messages à l'usager (WS_TERMS)

Le premier tableau, nommé RNG_INTENT, contient tout simplement les intentions possibles pour l'assistant. Il sert de liste à parcourir pour les procédures qui impliquent l'énumération des intentions.

Le deuxième tableau contient tous les messages pouvant être affichés à l'usager durant l'exécution de l'assistant. La première colonne contient tous les messages en français, la deuxième contient ces messages en anglais.

5.2 Fonctionnement interne de AP2

Dans cette section, nous détaillons les rôles des procédures dans le traitement des données et la construction du graphique, ainsi que les modifications qu'elles opèrent sur les feuilles de calcul. Les opérations extérieures aux procédures mentionnées sont contenues dans le programme principal, soit *pr_PostGraphe3*.

Certaines plages construites sont nommées dans cette section. Ces noms correspondent aux noms des objets utilisés dans le programme lui-même.

5.2.1 Création des outils d'appel

À l'ouverture même de la macro complémentaire d'AP2 se produit une procédure, *Auto_Open*, qui crée les outils d'appel de l'assistant, soit la commande dans le menu "Outils" et la boîte d'outils "PostGraphe" en leur affectant les procédures appropriées. Elle ajoute également les listes prédéfinies nécessaires.

Lorsque le programme est fermé, la procédure *Auto_Close* détruit tous les outils créés par la procédure *Auto_Open*, sauf les listes prédéfinies, qui pourront être utilisées dans d'autres contextes.

5.2.2 Validation de la plage sélectionnée

AP2 montre une boîte d'entrée (section 4.3) pour demander la validation de la plage sélectionnée. Ce type de boîte étant prédéfini dans Excel, aucune procédure n'est requise pour détailler son fonctionnement.

Une fois la plage sélectionnée, la procédure *pr_MakeVarRng* retranscrit les données sur `WS_TUPLE` sous forme de tableau (*VarRng*), en s'assurant que les colonnes sont d'égale hauteur. La procédure *pr_CheckTypes* vérifie la validité des titres, des candidats au clés, des classes et des valeurs.

La procédure, *pr_MakeSerieRng*, construit sur `WS_WORK` le tableau des groupes sériels (Tableau 5.7). Dans ce tableau, appelé *SerieRng*, la première colonne contient les types associés aux groupes sériels, suivi du nombre de variables sérielles contenues dans chaque groupe, puis des noms de ces variables.

Mesure	2	g de grain	g de fruit

Tableau 5.7: Exemple de tableau des groupes sériels

Après avoir identifié les groupes sériels potentiels, AP2 détermine avec *pr_ValidRng* les structures possibles pour l'arrangement des variables sélectionnées. Si aucune structure n'est possible, AP2 fait revenir l'usager à la première étape de sélection; sinon, il construit *StrucRng*, le tableau des différentes structures de données possibles (Tableau 5.8).

[X, Y, ZS]	[X, Y, Z, T]

Tableau 5.8: Exemple de tableau des structures

5.2.3 Construction du tableau des choix possibles

Pour éviter de donner à l'usager des choix inutiles, AP2 compile tous les choix possibles d'intentions et d'arguments. La procédure *pr_MakeChoiceSpec* construit un tableau qui, pour chaque structure possible de chaque intention, détermine tous les arrangements d'arguments possibles (i.e. tous les choix qui seront proposés à l'usager). Pour les structures comprenant une seule série, la procédure appelle la sous-procédure *pr_Choice*. Pour la structure [X,YS], elle appelle *pr_ChoiceXYS*. Pour la structure [X,Y,ZS], elle appelle *pr_ChoiceXYZS*. Cette tâche fait usage du tableau RNG_CTAB. Pour chaque arrangement d'arguments, ces sous-procédures font appel à la procédure *pr_ChLine* pour l'écriture des données.

Les procédures sont plutôt longues en raison de la multitude des choix possibles, surtout avec des séries multiples. Il faut considérer que les variables sérielles potentielles d'une structure de données peuvent souvent être prises comme variables non sérielles.

Le tableau résultant de la sélection, appelé *ChoiceSpecRng* et situé dans WS_WORK, contient neuf colonnes. Un exemple est présenté au Tableau 5.9.

évolution	1	[X, Y, ZS]	g de grain et g de fruit	Naissance			FAUX	VRAI
prèsentation	7	[X, Y, ZS]	Oiseau				VRAI	VRAI
		[X, Y, ZS]	Naissance				VRAI	VRAI
		[X, Y, ZS]	g de grain et g de fruit				VRAI	VRAI
		[X, Y, Z, T]	Oiseau					
		[X, Y, Z, T]	Naissance					
		[X, Y, Z, T]	g de grain					
		[X, Y, Z, T]	g de fruit					

Tableau 5.9: Exemple de tableau de choix d'arguments

La première colonne du tableau contient les noms des intentions. La deuxième colonne contient, pour chaque intention, le nombre de choix possibles. La troisième colonne contient, pour chaque choix, la structure de données considérée. La quatrième colonne contient le premier argument et la cinquième, le deuxième (s'il y a lieu). Si un groupe sériel est pris comme argument, on utilise la fonction *fn_SerieToString* pour exprimer la liste en texte. Les deux colonnes suivantes servent à contenir les variables sérielles potentielles qui, pour le choix particulier, ne seront pas considérées comme sérielles, donc exclues du groupe sériel utilisé. La huitième colonne indique VRAI si le groupe sériel (s'il y a lieu) peut être utilisé tel quel dans le graphique (i.e. si on peut avoir plusieurs séries). La neuvième colonne indique VRAI si une fonction peut être appliquée au groupe sériel (i.e. si les variables sérielles sont quantitatives).

Une fois ces choix effectués, il suffit de modifier la boîte de dialogue pour la deuxième étape de AP2. Cela se fait avec *pr_Getintent*, qui construit et affiche la boîte DLG_INT (DLG_INT_ANG pour la version anglaise) selon les choix possibles de *ChoiceSpecRng*.

Certaines procédures sont reliées aux éléments de DLG_INT et DLG_INT_ANG. La procédure *ActiveIntent* identifie l'intention sélectionnée dans la boîte de dialogue. La procédure *lstChoices_QuandClic* modifie le contenu de l'étiquette de description lorsque la sélection dans la liste change. La procédure *OpButt_QuandClic* modifie les contenus de l'étiquette de description et de la liste d'arguments lorsqu'une intention différente est sélectionnée.

Une fois la fenêtre fermée, la procédure recueille les choix d'intention (*Intent*), et place les arguments choisis dans un une plage horizontale de WS_WORK nommée *ArgRng*. Elle en déduit la structure de données (*DataStruct*).

5.2.4 Saisie des fonctions

Cette tâche est acquittée par la procédure pr_Functions. La première recueille grâce à la boîte de dialogue DLG_FNSERIE (DLG_FNSERIE_ANG dans la version anglaise) la fonction (*Fn*) appliquée à une variable, le nom de cette variable (*FnVar*), et la fonction appliquée au groupe sériel (*FnSerie*). Certaines procédures contrôlent les objets de la boîte de dialogue. La procédure *lstFunction* affiche dans une étiquette la description d'une fonction sélectionnée dans une liste. Les procédures *lstFn_QuandChangement*, *lstFnSerie_QuandChangement1* et *lstFnSerie_QuandChangement2* appellent cette procédure. La première affiche la description d'une fonction appliquée à une variables; les deux autres affichent la description d'une fonction appliquée à un groupe sériel.

La procédure pr_CheckFunc vérifie si les fonctions sont applicables aux données après la confirmation des choix.

La première opération effectuée par le programme sans interaction avec l'usager est de construire le titre par défaut du graphique avec la procédure *pr_GetTitle*. L'application des fonctions choisies sur les variables choisies (s'il y a lieu) s'effectue à l'aide de la procédure *pr_ApplyFunc*, qui crée un nouveau tableau *VarRng* à partir des données du premier, toujours dans WS_TUPLE.

5.2.5 Spécification des variables

La procédure *pr_MakeSpec* construit dans WS_WORK un tableau (*VarSpecRng*) contenant les propriétés de chaque variable et s'il y a lieu, du groupe sériel. Le Tableau 5.10 montre un extrait de ce tableau.

Pour chaque variable, le programme appelle la sous-procédure *pr_VarSpec*, qui crée une partie du tableau. Pour le groupe sériel, il utilise la procédure *pr_SerieSpec*. Dans la première colonne est placé le nom de la variable ou du groupe. La deuxième colonne contient le nombre de lignes du tableau consacrées à cette variable ou au groupe. La troisième colonne contient une valeur booléenne qui indique si la variable est candidate de clé (non la clé véritable de la relation), ce qui indique toujours faux pour le groupe sériel. Les quatres colonnes suivantes contiennent la ligne du tableau des types (section 5.1.2) associée à la classe de la variable ou du groupe. La huitième colonne indique le nombre de valeurs possibles que peut prendre la variable. Si celle-ci n'est pas énumérée, ce nombre devient 0. La colonne suivante contient la liste triée des valeurs de cette variable, ou le minimum et le maximum si celle-ci est quantitative.

Oiseau	6	VRAI	Nom	FAUX	VRAI	Alphabétique	6	Bafa	1
								Frio	2
								Kwak	3
								Pouf	4
								Saru	5
								Torr	6

Tableau 5.10: Extrai du tableau de spécification des variables

Pour créer la liste des valeurs, le programme fait appel à la procédure *pr_EnumVar*, ou *pr_EnumSeries* pour le groupe sériel. Cette procédure insère dans la liste toutes les valeurs de la liste contenue dans RNG_LTAB (section 5.1.2) comprises entre la plus grande et la plus petite, même celles non comprises parmi les données. Cela permettra de créer un graphique équilibré. Dans notre exemple, si il manquait un spécimen pour une année située entre la première et la dernière, le programme inclurait tout de même cette année dans le graphique d'évolution pour mieux exprimer la continuité qui existe dans cette variable.

Parfois, il faudra sur un graphique Excel placer des valeurs énumérées sur une échelle numérique. Pour permettre cela, on crée dans le tableau des spécifications une colonne à côté de celle des valeurs qui contient pour chaque valeur un numéro d'index. Ainsi, dans un graphique à echelle continue, la position d'un point correspondrait à l'index du tableau.

Si la variable ou le groupe est quantitatif, le programme fait appel à la procédure *pr_HiLoVar* ou *pr_HiLoSeries* respectivement. Ces procédures trouvent et arrondissent (à l'aide de la fonction *fn_RoundBnd*) le maximum et le minimum des valeurs. Les valeurs, bien sûr, ne sont pas indexées dans ce cas.

5.2.6 Distribution selon les intervalles

Cette étape fait appel à la procédure *pr_Intervals* et ne s'applique qu'à deux intentions: la répartition et la proportion. La procédure prend d'abord la variable répartie selon ses valeurs individuelles ou, si elle est quantitative, des intervalles de longueur égale. Pour chaque intervalle ou valeur, le programme compte le nombre d'instances dans le tableau original et place le nombre d'instances pour chaque valeur dans un nouveau tableau.

Prenons, dans notre exemple, l'intention de répartition de la consommation de grammes de grain parmis les oiseaux. La procédure construit le Tableau 5.12 à partir du Tableau 5.11.

Bien sûr, après avoir créé ce nouveau tableau, le programme doit refaire le tableau des propriétés des variables ainsi que le tableau des arguments. Cela s'effectue de la même manière que la première fois.

g de grain	Oiseau
Mesure	Nom
	VRAI
12.2	Frio
1.2	Kwak
8.0	Saru
4.6	Bafa
9.7	Torr
11.3	Pouf

Tableau 5.11: Avant la répartition

Valeurs de g de grain	Instances de g de grain
Intervalle	Naturel
VRAI	FAUX
[1, 2[	1
[2, 3[	0
[3, 4[	0
[4, 5[	1
[5, 6[	0
[6, 7[	0
[7, 8[	0
[8, 9[	1
[9, 10[	1
[10, 11[	0
[11, 12[	1
[12, 13[	1
[13, 14[	0
[14, 15[	0
[15, 16[	0

Tableau 5.12: Après la répartition

5.2.7 Choix d'un type de graphique valide

Le choix du graphique se fait à l'aide de la procédure *pr_IdGraph*, qui contient plusieurs sous-procédures.

La procédure *pr_GetGraph*, pour trouver le meilleur graphique possible, parcourt RNG_ITAB selon l'intention et la structure. Pour chaque graphique potentiel, la procédure *pr_ArrVar* fabrique un tableau à deux colonnes (Tableau 5.13) nommé *ArrRng*. La première colonne contient les variables placées selon leurs positions (X, Y, Z, T) assignées par le type de graphique. La deuxième colonne contient une valeur booléenne qui indique si la variable est à sa position prédéfinie. Seuls les arguments de l'intention et les groupes sériels ont des positions prédéfinies; les autres variables peuvent être permutées entre elles pour satisfaire les contraintes d'un type de graphique.

La procédure *pr_GSpecRng* trouve dans RNG_GTAB les spécifications du graphique candidat. La procédure *pr_ValidGraph* vérifie si un type de graphique est valide pour la combinaison de variables. Elle fait appel à la sous-procédure *pr_ValidType* pour vérifier si les propriétés des variables sont compatibles avec le graphique. Tant que l'arrangement est incompatible, la procédure trouve une nouvelle permutation de variables avec *pr_Perm*. Si, après avoir vérifié toutes les permutations, aucun arrangement valide n'est trouvé, le programme passe au graphique potentiel suivant.

Oiseau	VRAI
Naissance	FAUX
g de grain	FAUX
g de fruit	FAUX

Tableau 5.13: Tableau d'arrangement des variables

5.2.8 Construction du tableau d'entrée

Si le programme trouve un graphique valide et un arrangement des variables pour ce graphique, AP2 passe à la procédure *pr_MakeGraphRng*, qui reconstruit le tableau d'entrée (*GraphRng*) selon l'ordre de variables convenable. Les variables ordonnées sont également triées pour clarifier le graphique.

Si une variable énumérée devra être placée sur une échelle numérique, on remplace dans *GraphRng* les valeurs de cette variable par les numéros d'index correspondants. Ainsi, lorsque les points seront portés sur le graphique, il suffira de remplacer les marques de graduation par les valeurs énumérées de la variable.

Entre autre, si une variable est énumérée, il est possible que certaines valeurs du domaine présentes dans le tableau des propriétés de la variable ne soient pas présentes parmi les valeurs de la variable. Si la variable est clé de la relation et exprime une continuité (par exemple, le temps dans une évolution), il faut ajouter ces valeurs. Dans ce cas, la procédure ajoute des tuples au tableau, où seulement la colonne de la clé contiendra une valeur. Les autres colonnes étant vides, aucun point ne sera présent sur le graphique.

5.2.9 Construction du graphique

Le graphique est construit par la procédure *pr_MakeGraph*, qui contient plusieurs sous-procédures. La première, *pr_Format*, trouve le format spécifique du graphique parmi les choix prédéfinis pas Excel.

À sa création, le graphique contient des séries par défaut, tirées de la plage sélectionnée. La procédure *pr_AddSeries* supprime ces séries invalides et crée les séries de points voulues.

Dans plusieurs cas, les points d'un graphique ont des caractéristiques rétiniennes particulières. Plusieurs sous-procédures ont été créées pour leur attribuer ces caractéristiques. La procédure *pr_PntEti* étiquette les points d'un graphique. La procédure *pr_PntFor* applique des formes différentes aux points. La procédure *pr_PntGri* colle sur les points des ovales de couleurs différentes et, s'il y a lieu, de tailles différentes. La procédure *pr_PntSup* fait de même pour les graphiques avec plusieurs séries de données. La procédure *pr_PntAir* colle sur les points des ovales de tailles différentes seulement.

Comme mentionné dans la section 4.2, dans certains graphiques, les valeurs sont indexées. Il faut, dans ces graphiques graduer les axes avec des boîtes de texte. Cette tâche est effectuée par la procédure pr_Ticks.

La fin de la construction du graphique marque la fin de l'assistant.

5.2.10 Choix de la langue

Cette opération est extérieure au déroulement de l'assistant lui-même. Elle est appelée par le bouton de la langue sur la boîte d'outils PostGraphe. Elle modifie la valeur contenue dans la cellule RNG_LANGUAGE (voir section 0) à l'aide de la procédure *pr_ChangeLanguage*.

5.3 En résumé

Nous avons détaillé le déroulement de AP2 du point de vue de l'usager et du point de vue interne, en expliquant les rôles des procédures, des variables importantes, des tableaux pré-définis et des tableaux construits. Bien sûr, les algorithmes sont plus profonds et comprennent des détails trop élaborés pour être approchés ici. Pour une compréhension totale du programme, on se rapportera au programme.

6 Conclusion

6.1 Discussion sur l'assistant PostGraphe

L'assistant PostGraphe 2 est un outil complet en lui-même, non plus une version simplifiée du prototype de Massimo Fasciano [Fasciano, 1996]. Son mécanisme est quasi infaillible et on ne peut plus simple pour l'usager. Ses capacités vis-à-vis les graphiques bidimensionnels dépassent pratiquement les capacités de l'assistant graphique d'Excel.

Bien que AP2 n'offre pas la complexité de PG (intentions multiples) ni le choix de graphiques de l'assistant d'Excel (graphiques tridimensionnels), on peut le considérer comme un hybride qui offre le meilleur des deux programmes, en rajoutant quelques caractéristiques particulières. Ces caractéristiques comprennent les fonctions applicables aux données, le choix de la langue indépendant de l'environnement de travail, et surtout le système de vérification des données qui fait défaut à l'assistant d'Excel. Nous pouvons dire que notre travail est sorti de l'ombre de ses inspirations pour s'affirmer comme réalisation complète en elle-même.

6.2 Appréciation des environnements de travail

6.2.1 Visual Basic pour Applications et Excel

Le language VBA utilisé dans le cadre de notre projet est relativement simple et se prête aisément à l'analyse. Les déclarations, les systèmes de paramétrages sont simples et permettent un débugage rapide.

Un aspect particulièrement apprécié de ce language est la détection automatique des erreurs de syntaxe au fur et à mesure de l'entrée du code. Aussi, la mise en page simplifie la lecture du code.

Nous nous devons d'apprécier la richesse de la collection d'objets qu'offre Excel. Heureusement, nous avions sous la main un volume [Webb, 1996] décrivant longuement et en détail chacun de ces objets. Nous pouvons affirmer avoir touché à la grand majorité des objets que comprend Excel, ce qui a enrichi notre compréhension de ce vaste environnement.

Mais si nous avons eu la chance d'apprécier les possibilités d'Excel, nous avons également relevé certaines limitations, en particulier vis-à-vis la construction de graphiques. Ainsi, plusieurs éléments générés automatiquement dans d'autres générateurs de graphiques ont dû être créés de toute part dans notre programme; par exemple, les légendes, les points de tailles variables, les axes énumérés.

Malgré cela, nous avons vu comment Excel peut devenir un environnement presque entièrement personnalisé grâce à VBA. Son environnement simple à la première approche peut se transformer en un outil adapté à des tâches diverses et divergentes. Remarquons que nous avons travaillé sur la version Excel 5.0. Avec un peu de chance, la nouvelle version offre encore plus de possibilités que nous n'avons pas pu exploité.

6.2.2 Environnement de travail

Notre environnement de travail était constitué d'un Macintosh LC 475 possédant une mémoire vive de 79.8 Mo et un processeur 68040. La vitesse d'exécution n'en était pas toujours entièrement satisfaisante, mais ses capacités ont largement suffi à mener à bien nos projets.

6.3 Travaux futurs

Les travaux sur l'assistant PostGraphe pourraient s'orienter dans plusieurs directions; par exemple en s'éloignant ou en se rapprochant des travaux de Massimo Fasciano [Fasciano, 1996].

Une première approche viserait à aller plus loin que la recherche de Massimo Fasciano en découvrant de nouvelles intentions et en rajoutant des modificateurs plus variés. On s'éloignerait alors des choix élémentaires donnés à l'usager pour se diriger vers une liste plus vaste de possibilités plus spécifiques mais toujours aussi claires. Pourtant, cela ressortirait plus de la recherche théorique que de la recherche technique. En effet, notre implantation se prête facilement à ce genre d'amélioration.

Une autre direction des travaux pourrait viser à modifier la base même de notre programme pour la rapprocher de celle de Fasciano. Notre programme pourrait, lui aussi, générer des tableaux et du texte, ainsi que tous les éléments propres au rapport statistique. Pourtant, cela s'éloignerait de notre but initial de remplacer l'assistant graphique d'Excel par un assistant plus simple et plus rigide.

Bien sûr, cette rigidité vis-à-vis le format d'entrée peut être partiellement éliminée. Nous pouvons simplifier le format d'entrée des données, faire varier les possibilités de format. Mais la souplesse du format d'entrée peut s'avérer dangereuse, comme avec l'assistant d'Excel. Ainsi, si les travaux sont dirigés vers la souplesse du format d'entrée, nous devons redoubler de vigilance quant aux erreurs que pourraient provoquer cette souplesse.

Une autre qualité de l'assistant d'Excel qui fait parfois défaut dans notre implantation est la dépendance des données d'un graphique envers les données de la feuille de calcul. Comme plusieurs caractéristiques du graphique (forme des points, légende, graduation des axes) doivent parfois être construits par notre programme, il en résulte que les résultats affichés ne soient plus liés aux données de la feuille de calcul. Malheureusement, il faudrait que tous nos formats de graphiques soient prédéfinis dans Excel pour assurer un lien solide entre les données et le graphique. Peut-être que les versions futures d'Excel permettront de créer automatiquement toutes les formes de graphiques comprises dans notre implantation.

7 Bibliographie

[Date, 1988] Date, C.J. (1988). *An Introduction to Database Systems*, volume I. Addison Wesley, édition 4

[Fasciano, 1996] Fasciano, M. (1996). *Génération intégrée de textes et de graphiques statistiques*, Thèse de Doctorat, Département d'informatique et de recherche opérationnelle, Université de Montréal, Publication #1039

[Webb, 1996] Webb, J. (1996) *Using Excel Visual Basic for Applications*, Second edition, Que Corporation