



Les équations dans DataDrill EXPRESS

Bonnes Pratiques

Olivier PINETTE

STATUT : V1.3 – 2010/10/21 - VALIDE



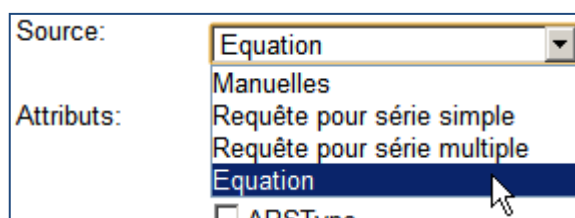
1 Introduction

Pour construire ses indicateurs, l'utilisateur de DataDrill EXPRESS a souvent besoin d'utiliser des équations pour valoriser certaines séries. Régulièrement, nous recevons des questions à ce sujet. Dans ce document, nous avons voulu détailler les principales équations disponibles dans DataDrill EXPRESS (version 4.1), mais aussi les astuces et bonnes pratiques.

2 Généralités

Une équation est une chaîne de texte interprétable et composée d'un ensemble de fonctions, d'opérateurs et/ou de constantes qui retournent une valeur. Les équations sont assez semblables à des formules de cellules dans un tableur, mais elles ne sont pas un langage de programmation structuré.

Les équations sont l'un des types de sources de données disponibles pour une série, au même titre que « manuelles » ou « requête »



Les équations peuvent retourner :

- Une valeur numérique, pour les séries de type « Données »
- Une couleur, pour les séries de type « Alarme »

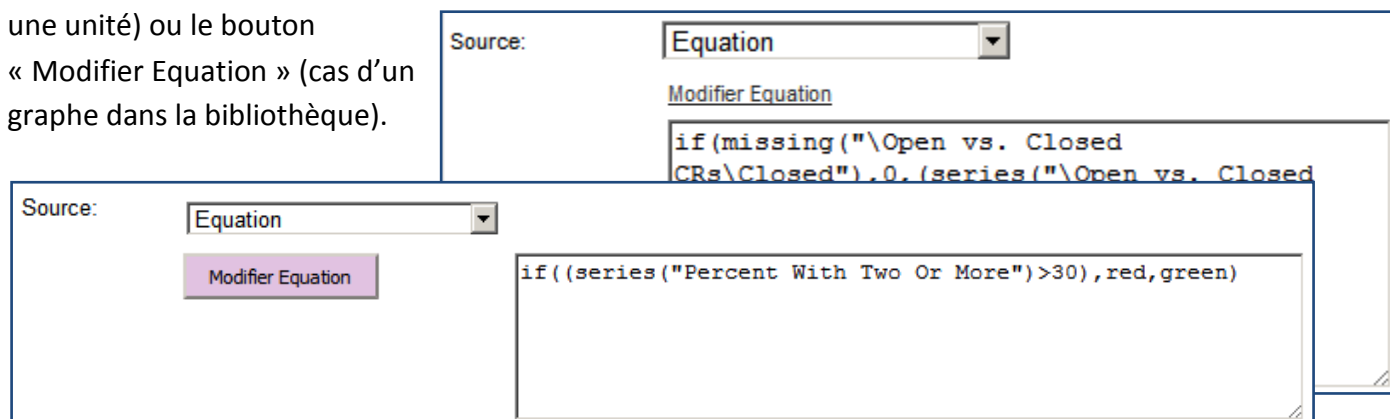
Elles peuvent également être utilisées pour spécifier les valeurs limites (inférieure et supérieure) d'une région.

Chaque équation est évaluée pour chaque période du calendrier utilisé par le graphe.

3 L'éditeur d'équations

L'éditeur d'équation vous permet de saisir et de tester les équations, mais aussi d'accéder à l'assistant (wizard).

Pour les graphes incluant des séries basées sur une équation, l'éditeur d'équations est accessible sur la page « propriété de la série » en suivant le lien « Modifier Equation » (cas d'un graphe dans une unité) ou le bouton « Modifier Equation » (cas d'un graphe dans la bibliothèque).



Rappel du nom du graphe et de la série

Zone de saisie de l'équation

Mémo des noms de séries du graphe courant

Série: \Defect- Defect Trends\Open Defects Alarm
 Equation:

```

if([series("\Open Defects")>400],
    red,
    (if([series("\Open Defects")>300],
        yellow,
        green)
    )
)
    
```

- Séries dans ce graphe :
- Closed Defects
 - Estimated Defects
 - New Defects
 - Open Defects
 - Open Defects Alarm
 - Total Defects

Résultats: 04/01/2010 11/01/2010 18/01/2010 25/01/2010 01/02/2010 08/02/2010 15/02/2010 22/02/2010 01/03/2010 08/03/2010 15/03/2010 22/03/2010 29/03/2010 05/04/2010 12/04/2010
 m m m m m m m m green green green green green green green

Enregistrer Annuler vérifier

catégorie: alarm détails de la fonction :

fonctions:
 alarm1G3s
 alarm2o3G2s
 alarm4o5G1s
 alarm8cl
 alarmAll4
 ryg1s
 ryg2s

Nom:
 Description:
 exemple:
 paramètres:

insérer au début insérer à la fin

Test et évaluation des valeurs de la série pour chaque période du graphe.

« m » = « missing » = indéfini.

L'assistant permet d'accéder à chaque fonction regroupée par catégorie.

Pour voir le détail d'une fonction, sélectionnez une catégorie, puis une fonction dans la liste en dessous. Sur la droite apparaît la description, un exemple de la fonction et les paramètres. Pour certains paramètres, un lien est affiché pour vous aider à choisir un dossier, produit d'information, un graphique ou une série.

4 Les équations

4.1 Les opérateurs

Les opérateurs mathématiques :

Opération	Opérateur	Exemples et remarques
Addition	+	a + b renvoie la somme de a par b <i>Exemple : 2 + 3 renvoie 5</i>
Soustraction	-	a - b renvoie la différence de a par b <i>Exemple : 3 - 2 renvoie 1</i> Voir également « changement de signe »
Multiplication	*	a * b renvoie le produit de a par b <i>Exemple : 2 * 3 renvoie 6</i>
Division	/	a / b renvoie la quotient de a par b <i>Exemple : 2/3 renvoie 0,6667</i> Pour la division entière voir les fonctions fix(exp) et int(exp)
Puissance	^	a ^ b renvoie a^b (a à la puissance b) exemple : 2^3 renvoie 8 Voir également la fonction pow(exp)
Modulo	%	a % b renvoie le reste de la division entière de a par b. Le résultat est du même signe que diviseur (a). <i>exemples : 10%3 renvoie 1</i> <i>-11%3 renvoie -2</i>
Changement de signe	neg(exp)	Neg(exp) renvoie la valeur de l'expression (ou de la constante) en ayant changé son signe. <i>Exemple : neg(2) renvoie -2</i> Attention, le symbole "-" ne peut pas être utilisé devant une expression ou une constante pour en changer le signe. Pour faire un changement de signe, utilisez la fonction neg(exp).

Les opérateurs logiques :

Opération	Opérateur	Exemples et remarques
Égal	=	Les opérateurs logiques sont généralement utilisés dans des fonctions conditionnelles, comme le IF : if((expression_logique), si_vrai, si_faux) Les expressions logiques doivent être entourées par des parenthèses ou des crochets. <i>Exemple : if([series("\Réelle")>0.95],RED,GREEN) renvoie Rouge pour les périodes où la série Réelle est > 0,95, Vert sinon.</i>
Inférieure	<	
Inférieure ou égal	<=	
Supérieure	>	
Supérieure ou égal	>=	
ET logique	&	Une expression logique peut contenir plusieurs conditions en utilisant l'opérateur OU (" "), ou l'opérateur ET ("&"). <i>Exemple : if([(series("\Réelle")>0.95) (series("\Réelle")<neg(0.95))], RED, GREEN) renvoie Rouge pour les périodes où la série Réelle n'est pas comprise entre -0,95 et 0,95, vert sinon.</i>
OU logique		



Précédence (priorité) des opérations

Pour imposer l'ordre des opérations il faut utiliser les parenthèses et les crochets : () []
 Sans parenthèses ou crochets, DataDrill effectue les opérations dans l'ordre suivant :

- % Modulo
- ^ Exposant
- * et / Multiplication et division
- + et - Addition et soustraction

4.2 Les constantes

Les constantes numériques :

nombre	<p>Toutes valeurs numérique, entière ou réelle.</p> <p>Remarques:</p> <ul style="list-style-type: none"> • utilisez la fonction neg(exp) pour les valeurs négatives (Exemple : neg(3) pour -3) • utilisez le point comme séparateur décimal (Exemple : 5.3 pour 5 virgule 3). • Les réels sont affichés avec une précision de 4 chiffres après la virgule. Exemples <ul style="list-style-type: none"> ○ 1/3 renvoie 0,3333 ○ 5.9999 renvoie 5,9999 ○ 5.99999 renvoie 6 ○ 5.99994 renvoie 5,9999 • Mais les calculs sont effectués avec davantage de précision (celle d'un double). Ainsi 5.99999*100000 renvoie 599999.
PI	Renvoie la valeur de π (3,1416)

Les constantes de type couleur :

Elles sont utilisables avec les équations liées aux séries de type « Alarme ».

Black	Noir
Blue	Bleu
Green	Vert
Orange	Orange
Red	Rouge
White	Blanc
Yellow	Jaune

Constante particulière :

Elle est utilisable aussi bien avec les séries de type « donnée » que « alarme »

IsMissing	Renvoie la valeur « manquante » (« non définie »)
-----------	---

4.3 Règles et astuces

- Les valeurs décimales doivent utiliser le point "." comme séparateur décimal. Dans les expressions DataDrill, la virgule est utilisée comme séparateur des arguments des fonctions, il n'est donc pas possible de l'utiliser comme séparateur décimal (par exemple écrire 5.3 et non 5,3 pour le nombre 5 virgule 3).
- N'utilisez pas de séparateur de milliers (par exemple écrire « 1234 » et non « 1 234 »).
- Les nombres négatifs doivent être écrits en utilisant la fonction "neg" (par exemple, utilisez « neg (10) » et non « -10 » qui ne fonctionnera pas toujours).
- Lorsque vous utilisez une expression logique (par exemple avec la fonction if), vous devez l'entourer avec des parenthèses ou crochets. Par exemple, vous devez écrire "if ((1> 5),...)" et non "if(1> 5,...)".
- Depuis la version 2 de DataDrill, les retours chariot ne rendent plus fou l'éditeur d'équations, vous pouvez donc utiliser le retour à la ligne pour améliorer la lisibilité de vos équations. De même, l'éditeur d'équations n'est plus sensible à la casse (majuscule/minuscule) des fonctions et/ou noms de séries.
- Certaines fonctions acceptent un paramètre optionnel qui permet de spécifier une valeur à utiliser si la valeur calculée est « missing » (manquante). Par exemple, la fonction `series("Réelle", 0)` retourne 0 si la valeur de la série *Réelle* est « missing ».
- Désignation d'une série :

Pour les graphes d'unité :

Dans le même graphe	<code>\<ref_série></code>
Dans un autre graphe de la même unité et du même item	<code>\<ref_graphe>\<ref_série></code>
Dans un graphe d'une autre unité et/ou autre item	<code>\<ref_unité>\<ref_item>\<ref_graphe>\<ref_série></code>

Où

Ref_série	Peut être le nom de la série, son id ou les 2: <code>[<Id_série>]<nom_série></code>
-----------	--

Ref_graphe	Peut être le nom du graphe, son id ou les 2 : [<Id_graphe>] <nom_graphe>
Ref_item	Peut être le nom de l'item, son id ou les 2 : [<Id_item>] <nom_item>
Ref_unité	Peut être le chemin complet (répertoire + unité) de l'unité, son id ou les 2 : [<Id_unité>] <Chemin>

Pour les graphes de la bibliothèque :

Dans le même graphe	\<nom_série>
Dans un autre graphe	\<nom_graphe>\<nom_série>

Remarques :

- L'id a priorité sur le nom, c'est-à-dire que lorsqu'un objet est identifié à la fois par son id et son nom, c'est l'id qui est utilisé et le nom n'est là qu'à titre indicatif.
- Attention, en cas de renommage du nom d'une série, d'un graphe, ou d'un répertoire, les équations y faisant référence par leur nom, ne seront pas mise à jour et retourneront une erreur lors de leurs évaluations.
- A contrario, si une série, un graphe, ou un répertoire a été effacé, puis recréé avec le même nom, il ne possédera plus le même identifiant, et les équations faisant référence à l'ancien id seront également en erreur lors de leurs évaluations.
- La grande majorité des équations utilisées font généralement référence à d'autres séries du même graphe. Il est alors facile en cas de renommage de se souvenir de mettre à jour les séries basées sur une équation. Cependant, compte tenu des 2 remarques précédentes, il est préférable de ne pas abuser des références hors du graphe et encore moins hors de l'unité, sous peine de ne plus savoir qui utilise quoi.
- Quelques équations (comme *graphsum*) peuvent faire référence à un (ou des) graphe(s) (et non des séries) les règles ci-dessus s'appliquent également en faisant abstraction de la partie concernant la série. Ainsi un chemin complet pour un graphe sera :
\<ref_unité>\<ref_item>\<ref_graphe>.
- Où trouver l'id des objets :

- Pour le graphe : sur la page des propriétés du graphe

Titre:	Cost- Budget At Completion	(2016)
--------	----------------------------	--------

- Pour une série : sur la page des propriétés de la série

Titre:	ACWP	7368
--------	------	------

- Pour une unité : sur la page des propriétés de l'unité

Unit Definition		
Titre:	Project 1	(185)

- Pour un item : sur la page des propriétés de l'item

Définition de l'item		
Titre:	Project01.xls	(241)



Exemples

- `series("\ACWP")`
désigne la série ACWP du graphe courant.
- `series("\[7368]")`
désigne la série portant l'id 7368 du graphe courant.
- `series("\[7368] ACWP")`
désigne la série d'id 7369 et de nom ACWP du graphe courant.
- `series("\Cost- Cost Performance\ACWP")`
désigne la série ACWP du graphe « Cost- Cost Performance »
- `series("\[2017] Cost- Cost Performance\[7368] ACWP ")`
désigne la série d'id 7368 et de nom ACWP du graphe d'id 2017 et de nom « Cost- Cost Performance »
- `series("\Projects\Project 2\Project02.xls\[2161] Cost- Budget At Completion\ACWP")`
désigne la série ACWP du graphe d'id 2161 et de nom « Cost- Budget At Completion » du répertoire « \Projects\ » de l'unité « Project 2 », de litem « Project02.xls »

4.4 Les fonctions principales

4.4.1 Fonctions spécifiques aux alarmes (Alarm)

Les fonctions suivantes sont généralement employées dans un graphe de contrôle (avec des fonctions SPC) pour indiquer sa stabilité/instabilité. Elles feront l'objet d'un autre document dédié à SPC.

Fonction	Exemples et remarques
alarm1G3s (Série, sérieLCL, sérieCL, sérieUCL)	Cette fonction renvoie rouge si un point est en dehors de ± 3 sigma, sinon la fonction renvoie vert.
alarm2o3G2s (Série, sérieLCL, sérieCL, sérieUCL)	Cette fonction renvoie rouge si deux de trois points consécutifs sont supérieurs à 2 sigma, sinon la fonction renvoie vert.
alarm4o5G1s (Série, sérieLCL, sérieCL, sérieUCL)	Cette fonction renvoie rouge si quatre de cinq points consécutifs sont supérieurs à 1 sigma, sinon la fonction renvoie vert.
Alarm8cl (Série, sérieLCL, sérieCL, sérieUCL)	Cette fonction renvoie rouge si huit points consécutifs sont du même côté de la ligne centrale, autrement, la fonction renvoie vert.
alarmAll4 (Série, sérieLCL, sérieCL, sérieUCL)	Cette fonction renvoie rouge si une quelconque des conditions suivantes est vraie : alarm1G3s, alarm2o3G2s, alarm4o5G1s or alarm8cl

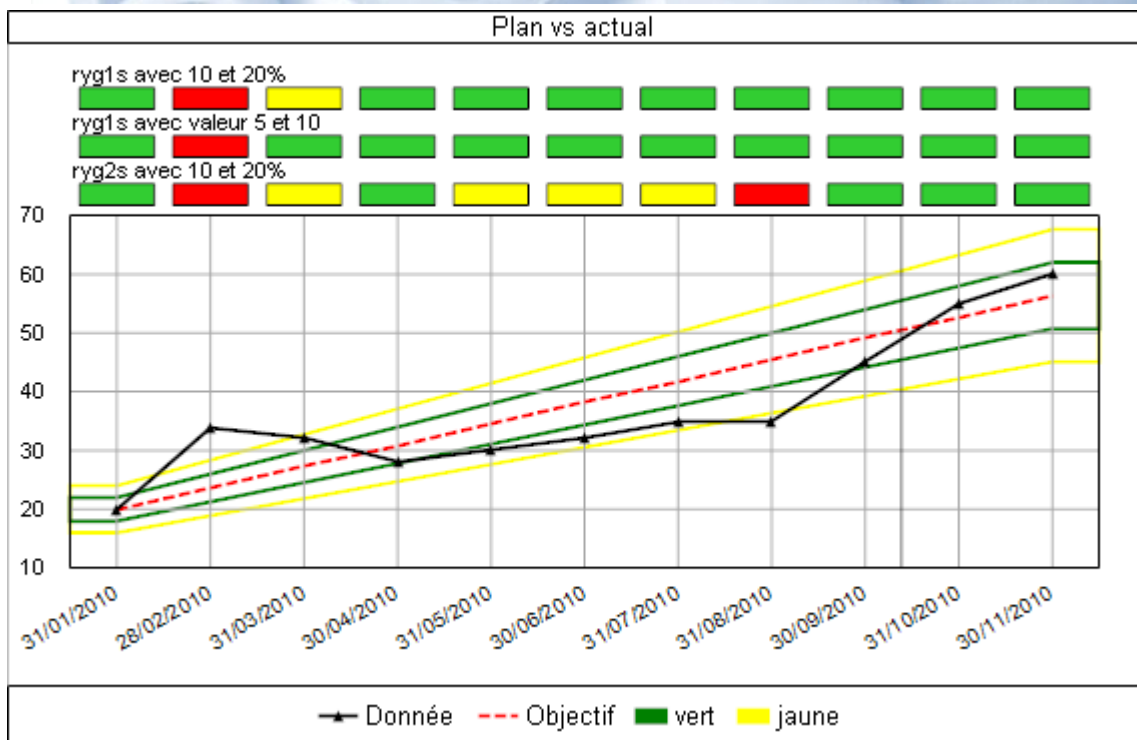
Les fonctions suivantes permettent la comparaison d'une série « objectif » avec une série « réelle » avec la prise en charge d'une tolérance autour de l'objectif (commandes ajoutées en version 3.8.0):

Fonction	Exemples et remarques
ryg1s (valeur(0) ou %(1), série_réelle, série_estimée, inner, outer)	Compare la série "réelle" et "estimée" et renvoie : vert si : réelle < estimée + inner Jaune si : estimée + inner < réelle < estimée + outer Rouge si : réelle > estimée + outer Inner et outer étant exprimés en valeur (0) ou en pourcentage (1) de la série « estimée » en fonction du premier paramètre. <i>Exemple : ryg1s(0, "Donnée", "Objectif", 10, 20)</i>
ryg2s (valeur(0) ou %(1), série_réelle, série_estimée, inner, outer)	Compare la série "réelle" et "estimée" et renvoie : vert si : estimée - inner < réelle < estimée + inner Jaune si : estimée - outer < réelle < estimée + outer Rouge : sinon Inner et outer étant exprimés en valeur (0) ou en pourcentage (1) de la série « estimée » en fonction du premier paramètre. <i>Exemple : ryg2s(0, "Donnée", "Objectif", 10, 20)</i>

Voici en exemple en image

Pour simplifier la lecture du graphe, nous avons également ajouté sur ce graphe, une région verte qui entoure l'objectif +10% et jaune pour l'objectif+20%. Voici les séries utilisées dans cet exemple :

Objectif (données)	<code>line(20,60)</code>
Données (Données)	<code>Manuelles: 20,34,32,28,30,32,35,35,45,55,60,70</code>
ryg1s avec 10 et 20% (alarme)	<code>ryg1s(0, "Donnée", "Objectif", 10, 20)</code>
ryg1s avec valeur 5 et 10 (alarme)	<code>ryg1s(1, "Donnée", "Objectif", 5, 10)</code>
ryg2s avec 10 et 20% (alarme)	<code>ryg2s(0, "Donnée", "Objectif", 10, 20)</code>
Vert (région)	<code>series("\Objectif")*1.1</code> <code>series("\Objectif")*0.9</code>
Jaune (région)	<code>series("\Objectif")*1.2</code> <code>series("\Objectif")*0.8</code>



4.4.2 Fonctions spécifiques aux prévisions (Forecast)

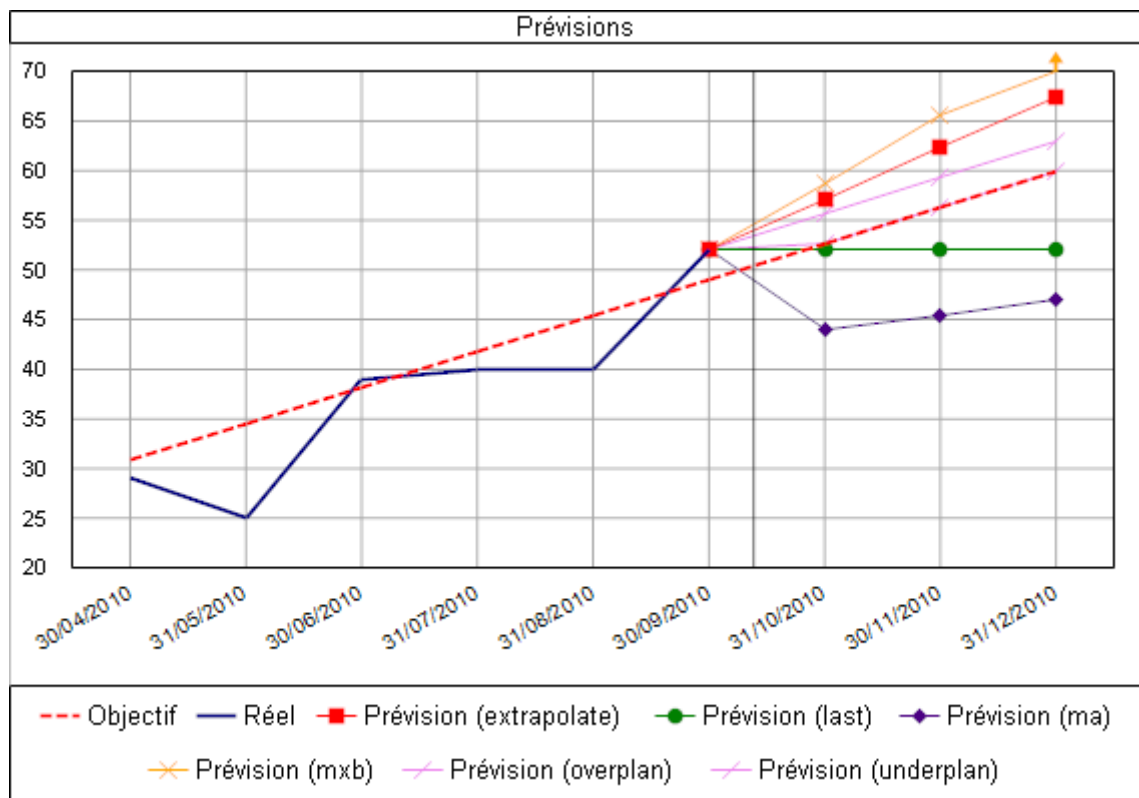
Voici un ensemble d'équations spécifiques à la valorisation des séries de données de nature « Prévision » (forecast).

Fonction	Exemples et remarques
extrapolate (série)	Extrapolation en construisant la droite passant par le premier et dernier point valorisée de la série spécifiée (tangente <premier point>/<dernier point >). <i>Exemple : extrapolate("\réel")</i>
last (Série)	Capture la dernière valeur de la série spécifiée et l'étend dans le futur. <i>Exemple : last("\réel")</i>
ma (série,n)	Calcule la moyenne mobile sur les n dernières périodes de la série spécifiée. <i>Exemple : ma("\réel",3)</i>
mxb (série,n)	Extrapolation en construisant la droite passant par le dernier et le n ^{ème} point valorisée de la série spécifiée (tangente <dernier point>/<dernière point -n +1>). <i>Exemple : mxb("\réel",5)</i>
overplan (série_réelle, série_estimée)	Si pour la dernière période valorisée de série_estimée, série_réelle > série_estimée, renvoie la valeur de série_estimée rehaussé de la différence entre séries_réelle et série_estimée sinon renvoie série_estimée <i>Exemple : overplan("\réel", "\Objectif")</i>
underplan (série_réelle,	Si pour la dernière période valorisée de série_estimée série_réelle < série_estimée, renvoie la valeur de série_estimée rabaisé de la

série_estimée)	différence entre séries_réelle et série_estimée sinon renvoie série_estimée <i>Exemple : <code>underplan("\réel","\Objectif")</code></i>
----------------	--

Voici en exemple en image, avec les séries utilisées dans cet exemple :

Prévision (extrapolate)	<code>extrapolate("\réel")</code>
Prévision (last)	<code>last("\réel")</code>
Prévision (ma)	<code>ma("\réel",3)</code>
Prévision (mxb)	<code>mxb("\réel",5)</code>
Prévision (overplan)	<code>overplan("\réel","\Objectif")</code>
Prévision (underplan)	<code>underplan("\réel","\Objectif")</code>



Remarque: Certaines autres équations peuvent également être utilisées sur les séries de nature « Prévision » (forecast), par exemple, `seriesmin`, `seriesmax`,...

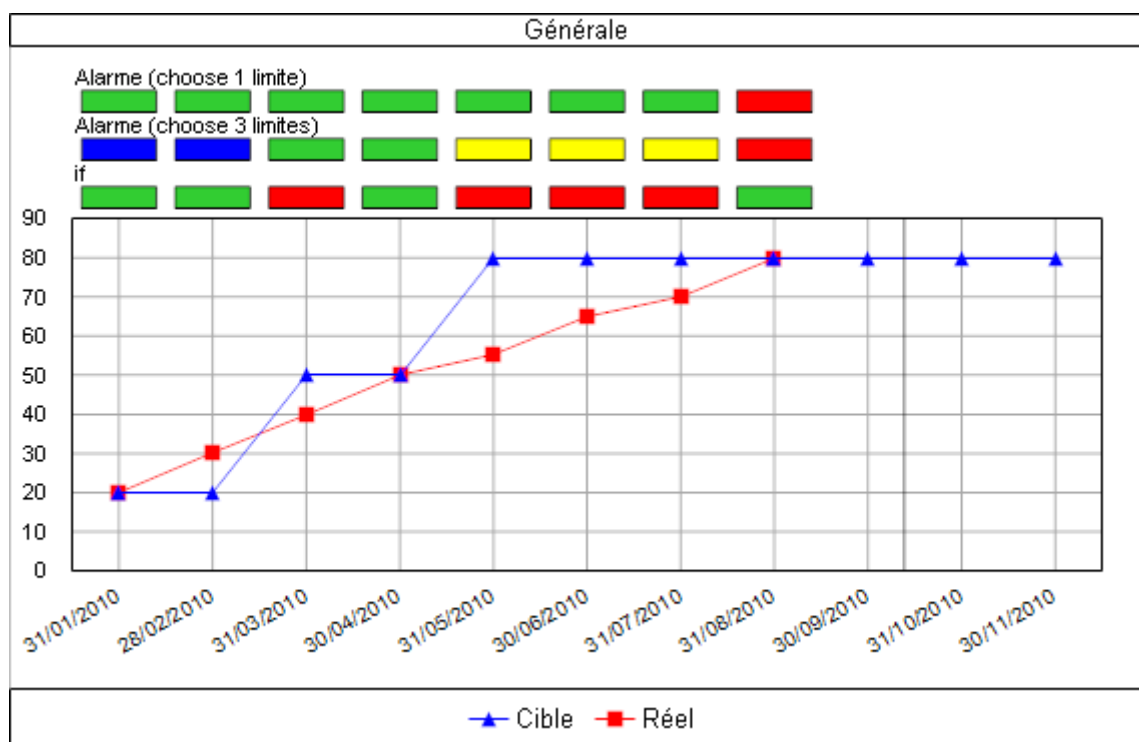
4.4.3 Fonctions générales (General)

Fonction	Exemples et remarques
attr (nom_attribut)	Renvoie la valeur de l'attribut (de type entier ou réel) valorisé sur la série courante, le graphe courant ou l'unité courante (dans cet ordre de priorité). <i>Exemple : Attr("Valeue Cible")</i>
choose (Série, Couleur1, Limite1, Couleur2 [,Limite2, Couleur3 [,Limite3, Couleur4 [,Limite4, Couleur5]]])	Pour chaque période, renvoie Couleur1 si la « Série » est entre 0 et Limite1 (inclusive), Couleur2 entre Limite1 et Limite2 (inclusive) Couleur3 entre Limite2 et Limite3 (inclusive) Couleur4 entre Limite3 et Limite4 (inclusive) Couleur 5 au delà de Limite4 (inclusive) Cette fonction peut avoir de 1 à 4 limites <i>Exemples : choose("\Réal", GREEN, 75, RED) choose("\Réal", BLUE, 30, GREEN, 50, YELLOW, 75, RED)</i>
data (nombre)	Renvoie le <i>nombre</i> pour toutes les périodes. Identique à la saisie du nombre directement ! <i>Exemple : data(10.25) renvoie 10,25</i>
data (val1,date1,val2 [,date2,val3 [,date3,val4 [,date4,val5]]])	Pour chaque période, renvoie val1 avant date1, val2 entre date1 et date2 val3 entre date2 et date3 val4 entre date3 et date4 val5 après date 4 Cette fonction peut avoir de 1 à 4 dates. Le format des dates est le format défini par votre culture ou la forme universelle (ISO) que nous vous conseillons AAAA/MM/JJ <i>Exemples : data(80, "2010/3/1", 50, "2010/5/1", 20)</i>
datapoints (liste de valeur)	Pour chaque période renvoie une valeur de la liste dans l'ordre de celle-ci. Les valeurs décimales sont autorisées et doivent utiliser le séparateur lié à votre culture et non le point ! Les facteurs « k » et « m » peuvent être utilisés respectivement pour Kilo et million <i>Exemple : datapoints("900;1,2k;1,4k;1,65k;1,0m") Renvoie 900, 1200, 1400, 1650, 1000000</i>
if (expression, si_vrai, si_faux)	Pour chaque période, si l'expression est vraie (ou différente de 0), renvoie l'argument 2, sinon renvoie l'argument 3. <i>Exemple : if((series("\Réelle")>0.95), RED, GREEN)</i>
ifin (valeur, min, max,	Pour chaque période, si la valeur est comprise entre min et max, renvoie l'argument 4, sinon renvoie l'argument 5. <i>Exemple : ifin(series("\Réelle"), 0, 100, RED, GREEN)</i>

si_vrai, si_faux)	
ifout (valeur, min, max, si_vrai, si_faux)	Pour chaque période, si la valeur est en dehors de l'intervalle [min...max], renvoie l'argument 4, sinon renvoie l'argument 5. <i>Exemple : ifout(series("\Réelle"),0,100,RED,GREEN)</i>
interp (regle, Periode1,Valeur1 ,Periode2,Valeur2 [,Periode3,Valeur3 [,Periode4,Valeur4]])	Permet à partir de 2 à 4 couples (période, valeur) d'interpoler chaque période en reliant chaque point spécifié par une ligne droite et en fonction de la règle donnée en premier paramètre : 0 : garde la 1 ^{ère} et dernière valeur 1 : garde la 1 ^{ère} valeur et interpole la dernière 2 : interpole la 1 ^{ère} valeur et garde la dernière 3 : interpole la 1 ^{ère} et la dernière valeur <i>Exemple : interp(0, 3, 12, 10, 33).</i> <i>Cf exemple en image</i>
region (série)	Renvoie la couleur de la région traversée par la série. <i>Exemple : region("\Réelle")</i>

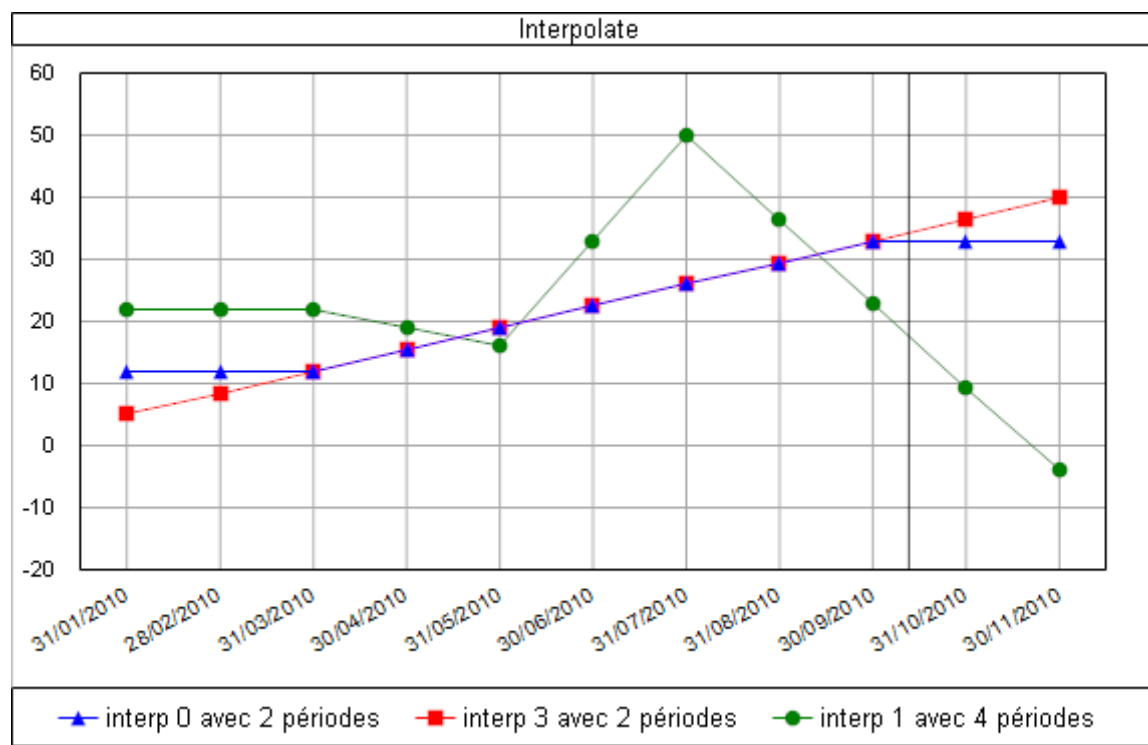
Voici un exemple en image de quelques fonctions, avec les séries utilisées dans cet exemple :

<i>Cible</i>	<i>data(20, "2010/03/01", 50, "2010/05/01", 80)</i>
<i>Réel</i>	<i>datapoints("10;15;20;25;30;35;40;45;50")</i>
<i>Alarme (choose 1 limite)</i>	<i>choose("\Réal", GREEN, 75, RED)</i>
<i>Alarme (choose 3 limites)</i>	<i>choose("\Réal", BLUE, 30, GREEN, 50, YELLOW, 75, RED)</i>
<i>if</i>	<i>if([series("\Réal") < series("\Cible")], RED, GREEN)</i>



Voici un exemple en image de la fonction *interp*, avec les séries utilisées dans cet exemple :

<i>interp 0 avec 2 périodes</i>	<i>interp(0, 2,12, 8,33)</i>
<i>interp 3 avec 2 périodes</i>	<i>interp(3, 2,12, 8,33)</i>
<i>interp 1 avec 4 périodes</i>	<i>interp(1, 2,22, 4,16, 6,50, 8,23)</i>



4.4.4 Fonctions logarithmiques (Log)

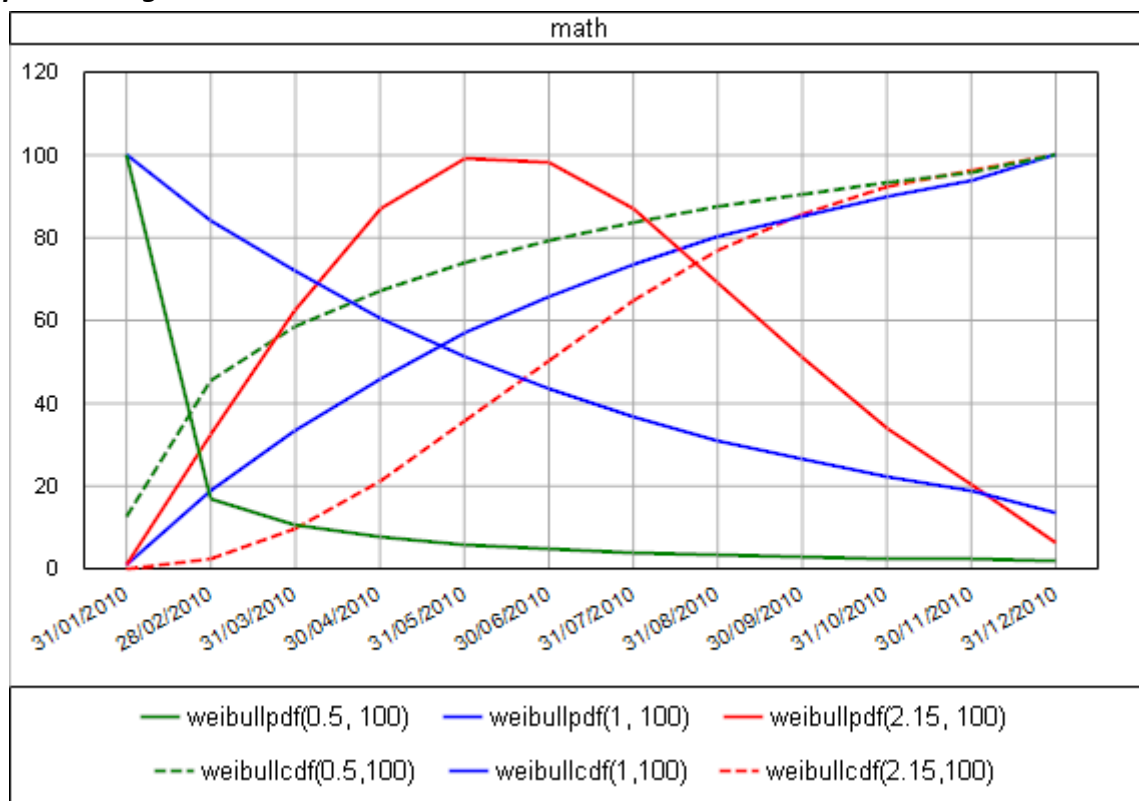
Fonction	Exemples et remarques
log (nombre)	logarithme (base 10) du nombre spécifié
logn (nombre, base)	logarithme base n du nombre spécifié
ln (nombre)	logarithme népérien (ou naturel) du nombre spécifié

4.4.5 Fonctions mathématiques (Math)

Fonction	Exemples et remarques
abs (nombre)	Valeur absolue du nombre spécifié <i>Exemple : abs(series("\r\u00e9elle"))</i>
exp (nombre)	Exponentiel du nombre spécifié <i>Exemple : exp(series("\r\u00e9elle"))</i>
int	Partie enti\u00e8re.

<p>(nombre) fix (nombre)</p>	<p>La seule différence entre les 2 fonctions concerne le traitement des nombres négatifs (cf exemple). <i>Exemple : <code>int(3.6)</code> et <code>fix(3.6)</code> retourne 3 <code>int(-3.6)</code> retourne -4 et <code>fix(3.6)</code> retourne -3</i></p>
<p>line (y1,y2)</p>	<p>Trace une ligne droite de (PP,y1) à (DP,y2). Ou PP= première période du calendrier du graphe, DP = dernière période du calendrier du graphe. (nous parlons bien ici des périodes du calendrier du graphe et non des périodes visualisées, qui peuvent être différentes) <i>Exemple : <code>line(0,250)</code> trace une ligne de 0 à 150</i></p>
<p>min (valeur1, valeur2) max (valeur1, valeur2)</p>	<p>renvoie le minimum ou maximum des 2 valeurs données en arguments. <i>Exemple : <code>min(100,10)</code> renvoie 10</i></p>
<p>neg (nombre)</p>	<p>Valeur négative du nombre <i>Exemple : <code>neg(5)</code> retourne -5</i></p>
<p>normale (réel)</p>	<p>Répartition selon loi normale. La répartition est faite sur l'ensemble des périodes du graphe, le pic est centré sur le milieu du graphe et son maximum donné en paramètre. <i>Exemple : <code>normale(100)</code></i></p>
<p>pow (x,y)</p>	<p>Retourne x à la puissance y <i>Exemple : <code>pow(series("\réelle"),10)</code></i></p>
<p>rayleigh ou rayleighcdf (forme)</p>	<p>Répartition selon la fonction de répartition de la loi de Rayleigh. La répartition est faite sur l'ensemble des périodes du graphe. <i>Exemple : <code>Rayleighcdf(2.15)</code></i></p>
<p>sqrt (nombre)</p>	<p>Retourne la racine carrée du nombre spécifié <i>exemple : <code>sqrt(series("\réelle"))</code></i></p>
<p>weibullcdf (forme, echelle)</p>	<p>Répartition selon la fonction de répartition (Cumulative Distribution function) de la loi de weibull. La répartition est faite sur l'ensemble des périodes du graphe. <i>Exemple : <code>weibullcdf(1.5,75)</code></i></p>
<p>weibullpdf (forme, echelle)</p>	<p>Répartition selon la densité de probabilité (Probability Density Function) de la loi de Weibull La répartition est faite sur l'ensemble des périodes du graphe. <i>Exemple : <code>weibullpdf(2.15, 100)</code></i></p>

Exemple en image de l'utilisation de la loi de Weibull :



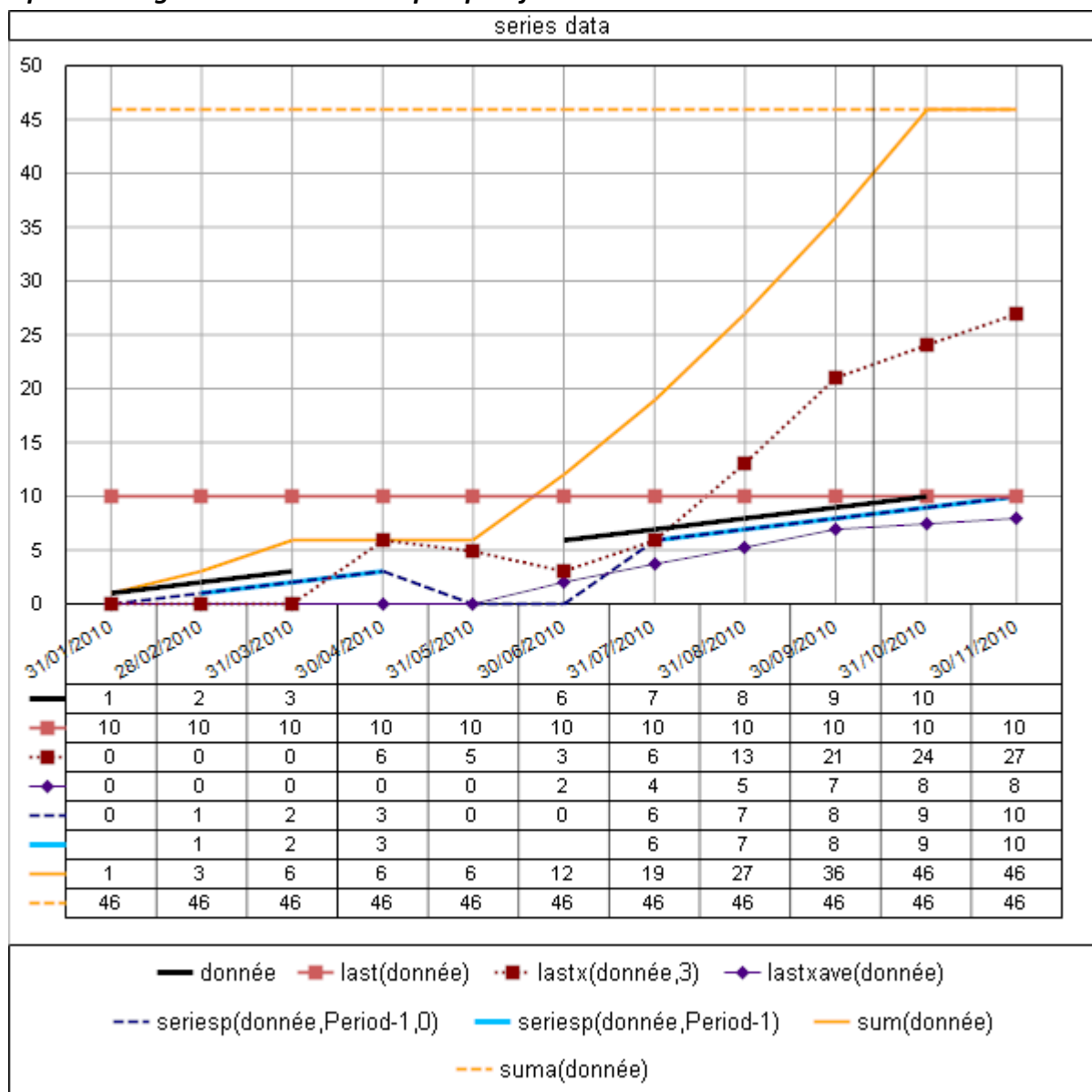
4.4.6 Fonctions liées aux séries de données (Series data)

Fonction	Exemples et remarques
first (Série)	L'ensemble des périodes est valorisé avec la première valeur de la série spécifiée. <i>Exemple : first("\Réelle")</i>
graphsum (Graphe)	Pour chaque période, somme de toutes les séries de nature « Réelle » du graphe spécifié. <i>Exemple : graphsum(""), renvoie la somme de toutes les séries Réelle du graphe courant (c-a-d du graph auquel appartient la série éditée). graphsum("\MonGraph ") renvoie la somme de toutes les séries Réelle du graphe « MonGrappe »</i>
last (Série)	L'ensemble des périodes est valorisé avec la dernière valeur définie (c-a-d non « missing ») de la série spécifiée. <i>Exemple : last("\Réelle")</i>
lastx (Série, n)	Pour chaque période, somme des n périodes précédentes de la série spécifiée. (Une valeur « missing » compte pour 0 dans la somme). <i>Exemple : lastx("\Réelle", 2)</i>

lastxave (Série, n)	Pour chaque période, la moyenne des n périodes précédentes de la série spécifiée. La moyenne ne prend en compte que les points « non-missing » parmi les n. <i>Exemple : lastxave("\Réelle", 3)</i>
lastxcount (Série, n)	Pour chaque période, nombre de point non « missing » dans les n périodes précédentes de la série spécifiée. <i>Exemple : lastxcount("\Réelle", 5)</i>
missing (Série)	Pour chaque période, renvoie 1 si la valeur de la série spécifiée est « IsMissing », 0 sinon. <i>Exemple : missing("\Réelle")</i>
period(0)	Pour chaque période, le numéro séquentiel de la période démarrant à 0. L'argument est toujours 0. <i>Exemple : period(0)</i>
points (Série)	Pour chaque période, le nombre de points de données (hors « missing ») de la série spécifiée à la période courante. <i>Exemple : points("\Réelle")</i>
pointsA (Série)	Nombre total de points de données de la série spécifiée (hors « missing ») <i>Exemple : pointsa("\Réelle")</i>
series (Série)	Pour chaque période, valeur de la série spécifiée. <i>Exemple : series("\Réelle")</i>
series (Série, VAL)	Pour chaque période, retourne la valeur de la série spécifiée ou « Val » si cette valeur est IsMissing <i>Exemple : series("\Réelle",0) qui renvera 0 lorsque Réelle est « IsMissing », à la difference de series("\Réelle") qui renvoie IsMissing lorsque « Réelle » est « IsMissing »</i>
seriesp (Série,period)	Pour chaque période, la valeur de la série spécifiée à une période donnée <i>Exemple : seriesp("\Réelle", (Period(0)-1))</i>
seriesp (Série,period,VAL)	Pour chaque période, la valeur de la série spécifiée à une période donnée ou « Val » si cette valeur est IsMissing <i>Exemple : seriesp("\Réelle", (Period(0)-1),0)</i>
sum (Série)	Pour chaque période, le cumul (somme jusqu'à la période courante) de la série spécifiée <i>Exemple : sum("\Réelle")</i>
suma (Série)	Somme sur toutes les périodes de la série spécifiée (hors missing) <i>Exemple : suma("\Réelle")</i>
sumtomissing (Série)	Cumul (comme sum) mais uniquement pour les périodes valorisées (hors IsMissing) et jusqu'au premier IsMissing. <i>Exemple : sumtomissing("\Réelle")</i>

<p>todate (Série)</p>	<p>Pour toutes les périodes, la somme des données de la série spécifiée pour les périodes incluses de celle-ci. Ceci n'a d'intérêt que lorsque l'on fait référence à une série d'un graphe utilisant un calendrier d'une fréquence inférieure. Par exemple pour utiliser des données mensuelles dans un graphe trimestriel.</p> <p><i>Exemple : <code>todate("\Graphe\Réelle")</code></i></p> <p>(ajoutée en version 3.8.0)</p>
<p>todate (Série,DateSpec)</p>	<p>Pour toutes les périodes, la somme des données de la série spécifiée pour les périodes incluses de celle-ci. Ceci n'a d'intérêt que lorsque l'on fait référence à une série d'un graphe utilisant un calendrier d'une fréquence inférieure et non strictement superposable. Par exemple pour utiliser des données semestrielles dans un graphe mensuel.</p> <p>1 utilisation de la date de début 2 utilisation de la date de fin</p> <p><i>Exemple : <code>todate("\Graphe\Réelle",1)</code></i></p> <p>(ajoutée en version 3.8.0)</p>

Exemple en image de l'utilisation de quelques fonctions :



- `last("\donnée")` retourne une droite d'ordonnée 10 qui est la dernière valeur de données
- `lastx("\donnée", 3)` effectue une somme (mobile) des 3 périodes précédentes de données
- `lastxave("\donnée", 5)` effectue la moyenne mobile des 5 périodes précédentes de données
- `seriesp("\donnée", (Period(0) - 1))` renvoie la série donnée décalée d'une période
- `seriesp("\donnée", (Period(0) - 1), 0)` renvoie la série donnée décalée d'une période et en remplaçant les missing par 0.
- `sum("\donnée")` renvoie la somme de proche en proche (période par période)
- `sums("\donnée")` renvoie le cumul de toutes les périodes

4.4.7 Fonctions liées aux séries min-max (Series min-max)

Fonction	Exemples et remarques
seriesmin (Série)	Pour chaque période, valeur minimale de la série spécifiée jusqu'à la période courante <i>Exemple : seriesmin("\Réelle")</i>
seriesmax (Série)	Pour chaque période, valeur maximale de la série spécifiée jusqu'à la période courante <i>Exemple : seriesmax("\Réelle")</i>
seriesave (Série)	Pour chaque période, moyenne de la série spécifiée jusqu'à la période courante (hors missing) <i>Exemple : seriesave("\Réelle")</i>
seriesmina (Série)	Valeur minimale sur toute la série spécifiée <i>Exemple : seriesmina("\Réelle")</i>
seriesmaxa (Série)	Valeur maximale sur toute la série spécifiée <i>Exemple : seriesmaxa("\Réelle")</i>
seriesavea (Série)	Moyenne sur toutes les valeurs de la série spécifiée (hors missing) <i>Exemple : seriesavea("\Réelle")</i>

Remarques : toutes ces fonctions acceptent un second paramètre, permettant de définir la valeur à renvoyer à la place de « IsMissing ». Par exemple Exemple : *seriesmin("\Réelle", 0)*, renvoie pour chaque période, la valeur minimale de la série spécifiée jusqu'à la période courante et 0 si le minimum est IsMissing.

4.4.8 Fonctions Statistiques (Spc)

Fonction	Exemples et remarques
mean (Série)	Moyenne sur toutes les valeurs de la série spécifiée (hors missing) (identique à la fonction seriesavea) <i>Exemple : mean("\Réelle")</i>
range (Série)	Écart absolu maximum de toutes les valeurs de la série spécifiée (identique à SeriesMaxA – SeriesMinA). <i>Exemple : range("\Réelle")</i>
sigma (Série)	Ecart type de toutes les valeurs de la série spécifiée <i>Exemple : sigma("\Réelle")</i>
variance (Série)	Variance de toutes les valeurs de la série spécifiée (=carré de l'écart type) <i>Exemple : variance("\Réelle")</i>

La rubrique SPC contient également les équations applicables à la maîtrise statistique des procédés (en Anglais : Statistical Process Control). Cette technique fera l'objet d'un autre document.

4.4.9 Fonctions trigonométriques (Trig)

Fonction	Exemples et remarques
cos (angle)	Cosinus de l'angle spécifié
sin (angle)	Sinus de l'angle spécifié
tan (angle)	Tangente de l'angle spécifié
cosh (réel)	Cosinus hyperbolique du réel spécifié
sinh (réel)	Sinus hyperbolique du réel spécifié
tanh (réel)	Tangente hyperbolique du réel spécifié
acos (réel)	Arc cosinus du réel spécifié
asin (réel)	Arc sinus du réel spécifié
atan (réel)	Arc tangente du réel spécifié

Remarque : les fonctions Arc Cosinus, sinus et tangente ne sont pas dans le wizard, mais peuvent être utilisées.

5 Traitement des références entre graphes avec calendrier différent

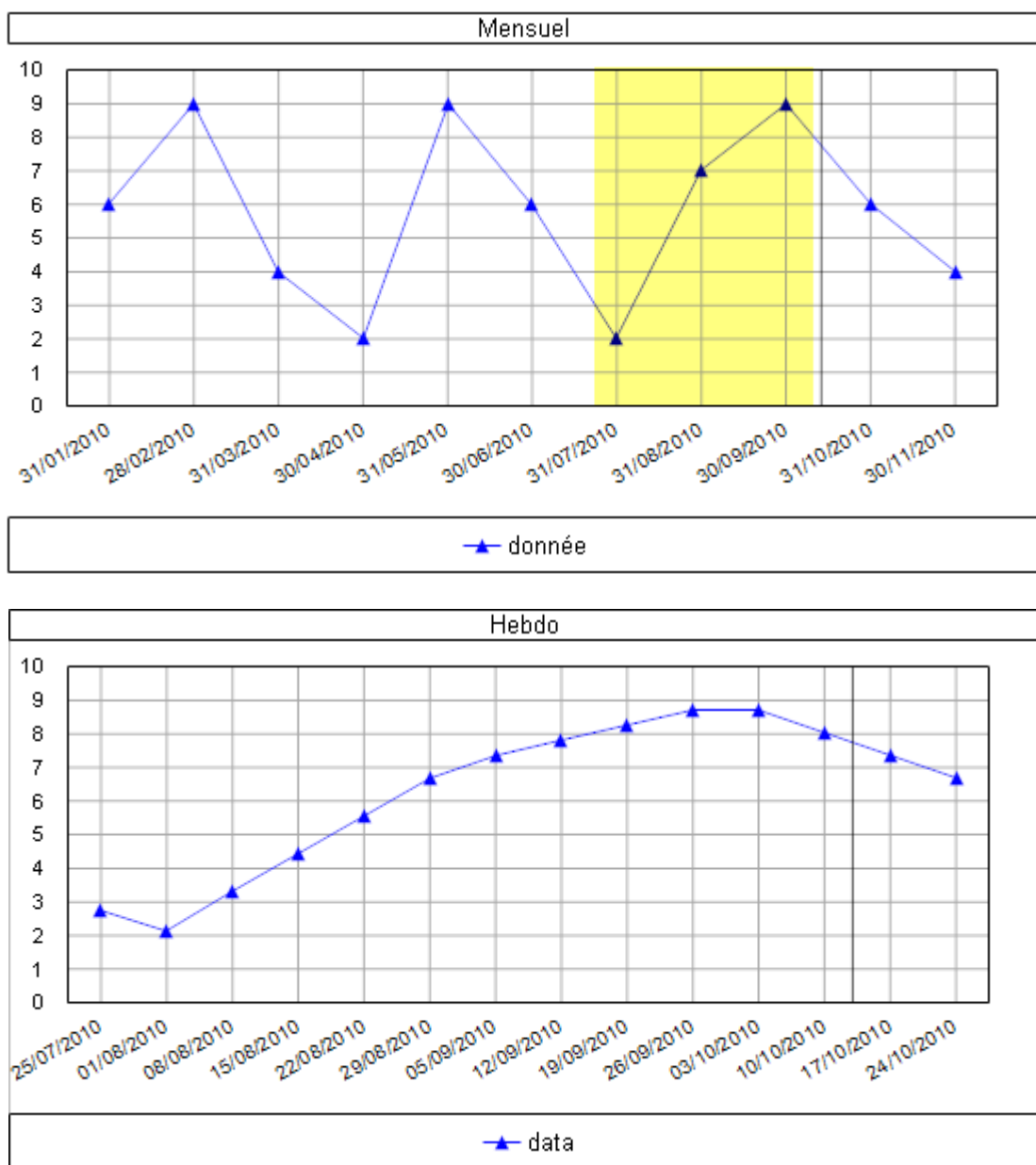
Dans le chapitre précédent, nous avons vu qu'il était possible de référencer dans un graphe une série venant d'un autre graphe. Mais, qu'advient-il lorsque les calendriers des 2 graphes ne sont pas identiques ?

Il convient de différencier 3, et même 4 cas :

1. Utiliser une série provenant d'un graphe qui a une fréquence plus élevée. Par exemple, un graphe mensuel contenant des séries que je souhaite utiliser dans un graphe hebdomadaire.
2. Utiliser une série provenant d'un graphe qui a une fréquence identique mais décalée. Par exemple, 2 graphes utilisant 2 calendriers mensuels différents l'un avec des périodes débutant au 1^{er} de chaque mois, l'autre avec des périodes débutant au 15 de chaque mois.
3. Utiliser une série provenant d'un graphe qui a une fréquence moins élevée. Ici, il convient de considérer 2 sous-cas :
 - a. Les fréquences synchronisées. Par exemple, un graphe mensuel contenant des séries à utiliser dans un graphe trimestriel.
 - b. Les fréquences ne sont pas synchronisées. Par exemple un graphe hebdomadaire contenant des séries à utiliser dans un graphe mensuel.

Cas 1 : utilisation d'une série mensuel dans un graphe hebdo

Voici le graphe mensuel avec en jaune la zone que nous allons visualiser sur le second graphe hebdomadaire.

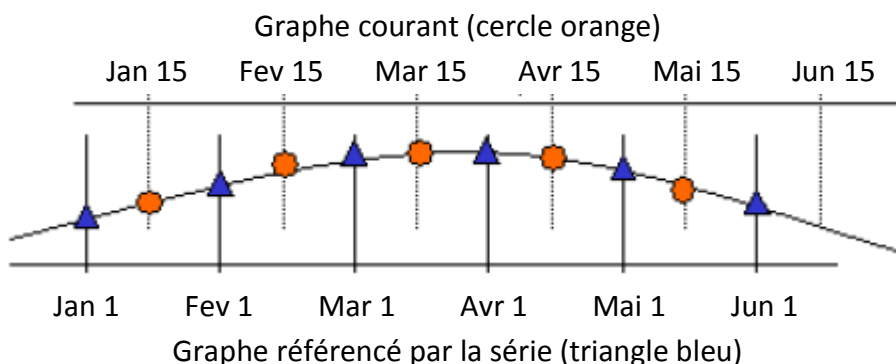


Nous voyons ici que DataDrill effectue une interpolation automatiquement. En fait, le mécanisme implémenté par DataDrill est le suivant : DataDrill évalue chaque équation pour chaque période du graphe. Il vérifie si la série référencée a une valeur pour la date exacte considérée. Si oui, il prend la valeur sans transformation. Sinon, et à condition d'avoir une valeur pour les dates avant et après, alors une interpolation est effectuée.

Cas 2 : utilisation d'une série mensuel d'un graphe mensuel décalé

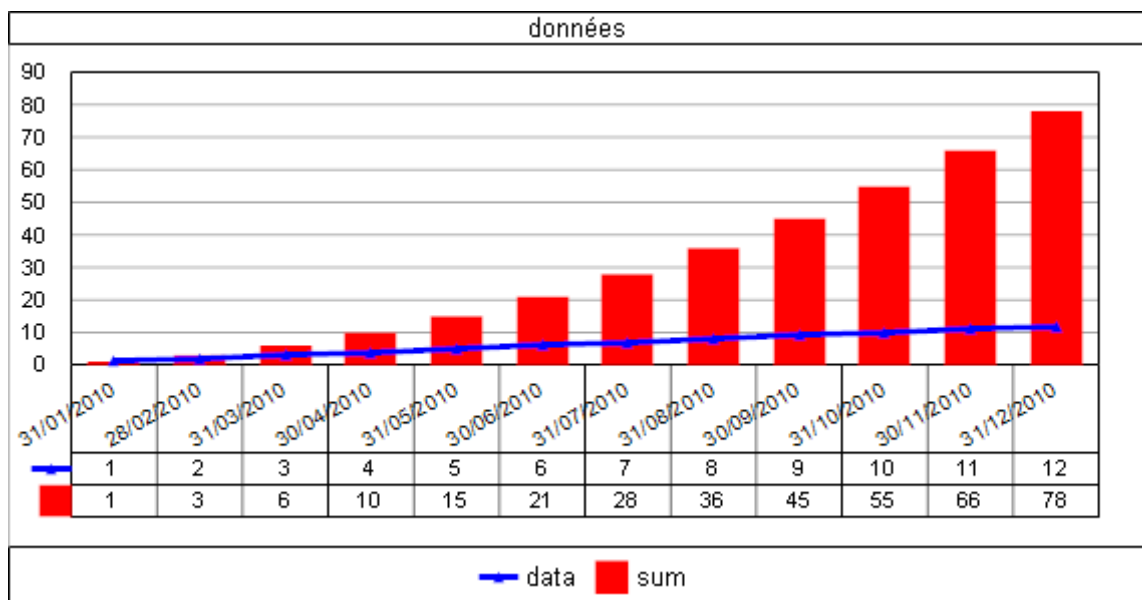
Dans cet exemple, nous avons un graphe utilisant un calendrier mensuel, basé sur le 15 de chaque mois, et qui référence une série d'un second graphe qui, lui, utilise un calendrier mensuel plus traditionnel puisque basé sur le 1^{er} de chaque mois.

Nous nous retrouvons dans un cas semblable à l'exemple précédent, et DataDrill effectuera une interpolation :

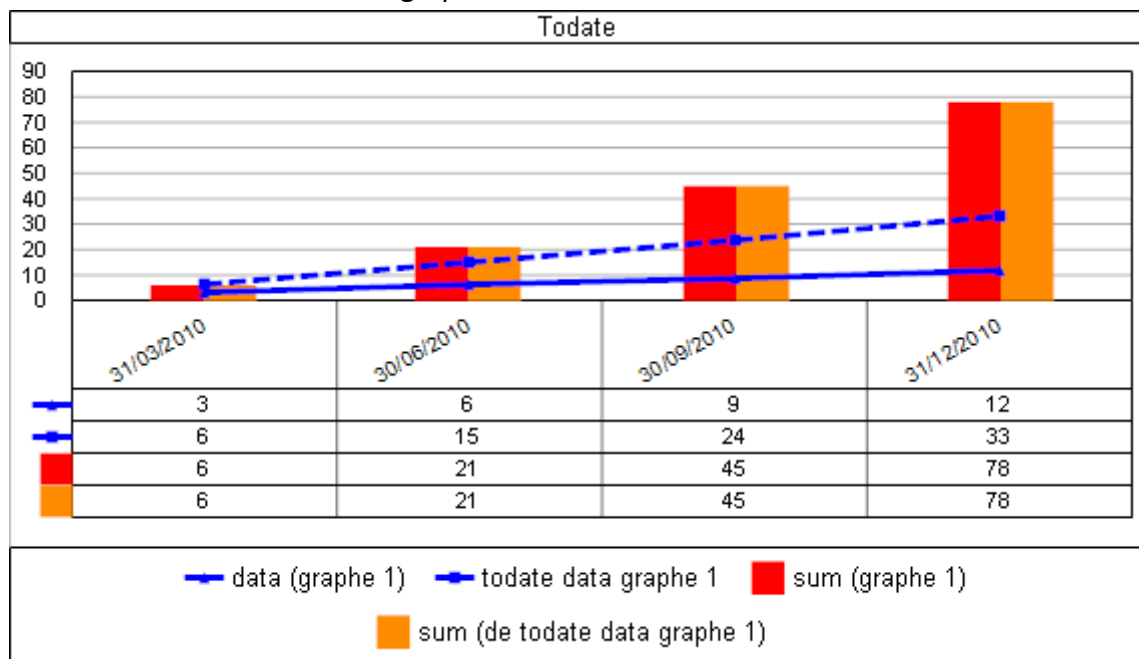


Cas 3a : Exemple de l'adaptation de l'échelle temps et de l'utilisation de la fonction Totate :

Voici un graphe que nous allons utiliser comme référence. Il contient une série de données nommée « data » (ligne bleu) et son cumul nommé « sum » (barre rouge), le tout sur un calendrier mensuel :



Considérons maintenant un second graphe où nous utiliserons un calendrier trimestriel.

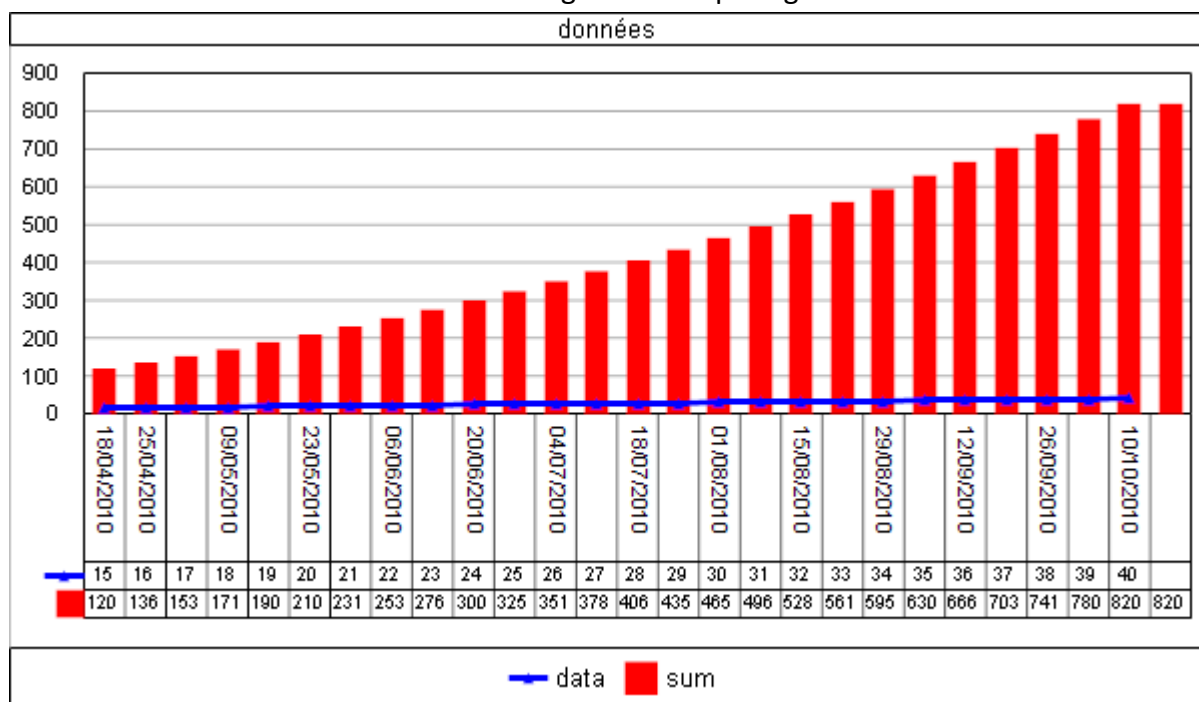


- data (graphe 1) (ligne bleu est l'équation `series("[2296] test\data")`). Nous observons que la valeur pour chaque fin de période est bien égale à la fin de période correspondante sur le graphe 1. Par exemple 6 pour les 20/06/2010 sur les 2 graphes.
- todate data graphe 1 (ligne pointillé bleu est l'équation `todate("[2296] test\data")`). L'équation additionne les valeurs de la série data pour toutes les périodes du graphe 1 incluse dans le période du graphe 2. Par exemple, pour obtenir la valeur 15 du second trimestre (au 20/06/2010), todate a calculé 4+5+6 ce qui correspond aux périodes d'avril, mai et juin.
- sum (graphe 1) (barre rouge est l'équation `series("[2296] test\sum")`). Correspond au report du cumul du graphe 1.
- sum (de todate data graphe 1) (barre orange est l'équation `sum("\todate data graphe 1")`). Correspond au cumul de la seconde série du graphe 2 (nommé « todate data graphe 1 »). Remarquons que les 2 barres sont identiques.

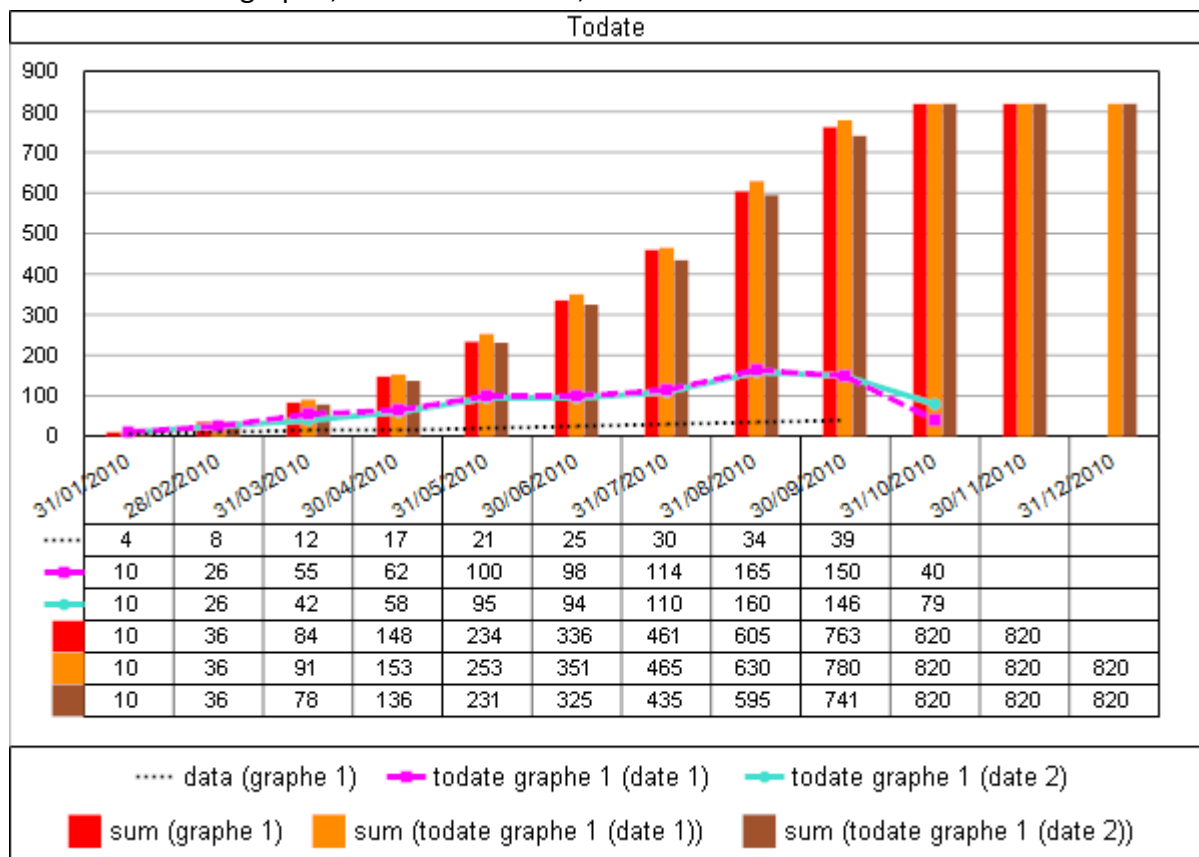
« A quoi cela sert-il ? » pouvez-vous vous demander ! Et bien, imaginons que les données du graphe 1 soient un nombre de défauts par exemple. Dans ce cas, si dans un second graphe de fréquence inférieure on souhaite réutiliser la même information, il convient de l'adapter à la bonne fréquence. Ainsi, le nombre de défauts d'un trimestre est bien la somme des défauts des 3 mois du trimestre.

Cas 3b : Autre Exemple avec des calendriers semestriels et mensuels:

Voici le graphe de référence, comme dans l'exemple précédent, mais avec un calendrier hebdomadaire. Nous avons réduit l'affichage à 6 mois pour garder une certaine visibilité.



Et voici le second graphe, utilisant cette fois, un calendrier mensuel :



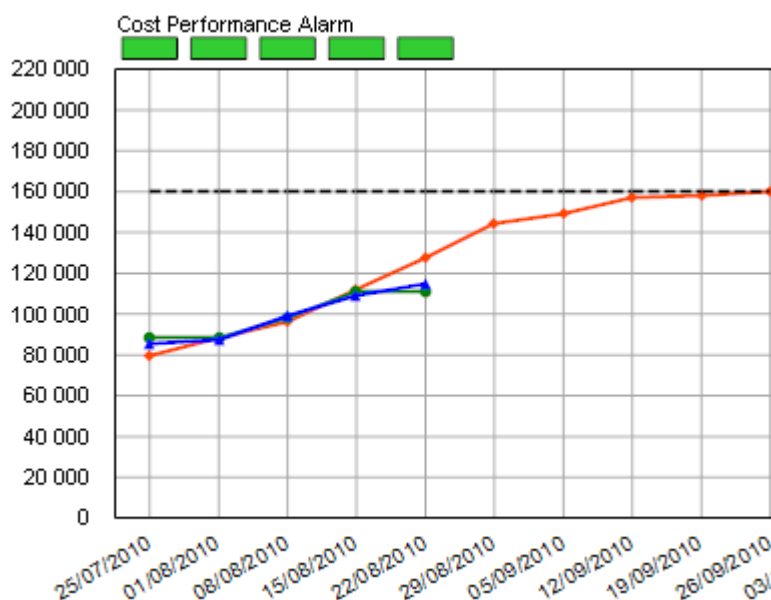
Dans le premier exemple, les mois étaient inclus entièrement dans les trimestres. L'inclusion du calendrier « mois » dans le calendrier « trimestre » était donc simple. Ici, nous avons un nouveau problème car il n'est pas possible d'inclure des semaines entières dans des mois. Nous devons donc utiliser une règle nous permettant de savoir quels mois sont à considérer comme faisant partie du mois ou pas. La fonction `today` possède un second argument facultatif permettant de gérer cela. Ce second argument peut prendre la valeur 1 ou 2. 1 pour considérer la date de début et 2 pour considérer la date de fin. Appliqué à notre exemple, cela signifie 1 pour considérer toutes les semaines commençant dans le mois comme en faisant partie, 2 pour considérer toutes les semaines finissant dans le mois comme en faisant partie.

6 Traitement des données manquantes (missing)

DataDrill autorise qu'une donnée soit « manquante » (pour une ou plusieurs périodes d'une série). C'est d'ailleurs la valeur par défaut : avant que les périodes d'une série soit valorisées, les valeurs sont considérées comme manquantes (appelées « missing » ou « `IsMissing` » dans `datadrill`).

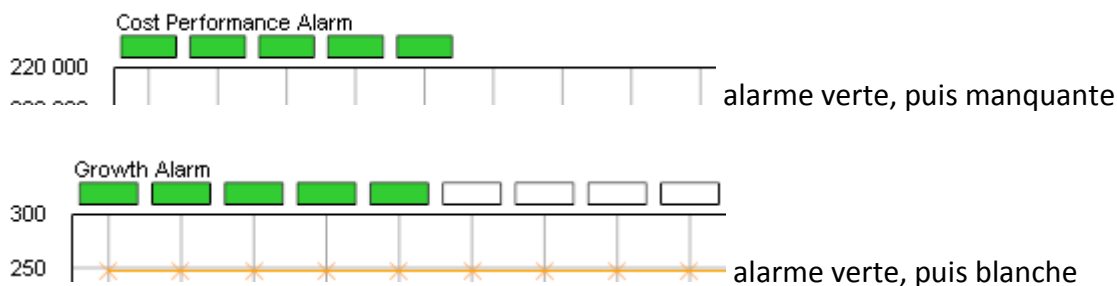
Il est important de comprendre le comportement des équations DataDrill face aux données manquantes, qui par nature sont différentes de toutes autres valeurs. Précisons aussi que nous ne parlons pas là de la valeur 0, qui est une valeur comme une autre, mais bien de l'absence de toute valeur. C'est assimilable au « `NULL` » des bases de données.

Sur l'exemple ci-contre, les séries de données rouges et noires sont valorisées sur toutes les périodes, mais la bleu et la rouge ne le sont que jusqu'au 22/08/2010. Au-delà, la valeur est « `IsMissing` » et donc non dessiné sur le graphe. Idem pour l'alarme qui est valorisée jusqu'à la même date, puis manquante.



Puisque les périodes « missing » n'ont pas de valeur, le comportement des opérations arithmétiques, mais aussi logiques, sera différent. En effet, comment comparer une valeur numérique à une valeur manquante ? Ou comment additionner un nombre et une valeur manquante ? Donc dans tous ces cas, le résultat sera lui-même manquant et cela pour tous les types de séries (Alarme, données, région, texte).

Une question revient parfois : comment distinguer une alarme Missing d'une alarme blanche ? La réponse en image si- dessous :



Les combinaisons (équation) incluant une donnée manquante sont elles-mêmes considérées comme manquantes :

valeur <opérateur> *MISSING* renvoie *IsMissing*

MISSING <opérateur> *valeur* renvoie *IsMissing*

Cela quelque soit l'opérateur arithmétique (+,*,/%^) ou logique (= <> & |).

Enfin, revenons sur le traitement des valeurs missing :

Comment pourrions-nous écrire une équation conditionnelle au fait qu'une série ACWP soit valorisée ou non ? Une équation du type `if((series("\ACWP")=IsMissing), 0, series("\ACWP"))` ne fonctionnera pas, puisque nous avons appris ci-dessus que toute combinaison avec une valeur missing renvoie missing, donc le résultat de cette équation sera toujours missing.

Par contre nous pouvons utiliser la fonction `missing(série)` :

`if((Missing("\ACWP")=1), 0, series("\ACWP"))` qui renvoie 0 si ACWP est missing, la valeur de la série ACWP sinon.

Il y a même une écriture encore plus simple : `series("\ACWP", 0)`. En effet la fonction `series()` accepte un second paramètre optionnel, permettant de préciser la valeur à utiliser en cas de missing.



7 Spirula en bref

Depuis près de 10 ans, Spirula propose des solutions pour mieux estimer et piloter les projets de développement de logiciels et systèmes.

Leader sur son marché, l'offre Spirula – expertise, outils, formation – permet de mieux Comprendre le passé, Piloter le présent et Prévoir l'avenir des projets d'ingénierie logicielle et systèmes.

Nous aidons nos clients à définir les processus de développement les plus efficaces, implémenter des tableaux de bords pour le suivi des projets et augmenter la fiabilité des estimations des coûts, effort et délais des projets.

Nos consultants sont experts dans le pilotage de projet et les estimations et conduisent l'implémentation des bonnes pratiques, comme le CMMI, dont Spirula est un des co-auteurs.

Parmi nos clients, nous comptons des PME/PMI ayant une forte activité de développement de logiciels et de systèmes ainsi que des grands comptes internationaux tel qu'Alstom, BAe, Continental, Philips, Renault, Thales, ...