M1 MIAGE Option RIM
Cours 4 : I/O Haskell, Données

Alexandre Termier

2010-2011 S2

I/O dans un langage pur ?

Rappel

Langage pur : une fonction **n'a pas d'effets de bord**
$\rightarrow$ appelée avec les mêmes paramètres, elle donne *toujours* les mêmes résultats

Fonctions d'I/O

- `getLine` : résultat dépend de ce qu'a tapé l'utilisateur
- ouverture de fichier : dépend de l'existence du fichier (qui peut changer au cours du temps)

$\rightarrow$ **pas pur du tout !**

La solution en Haskell

- Séparer les parties “pures” et “impures” du code
- Notion **d'action I/O**, typée `IO type` ou `IO ()`
- Isolation des parties du programme qui produisent des effets de bord

Example (getLine)

```
Prelude> :t print
print :: (Show a) => a -> IO ()
Prelude> print [1,2,3]
[1,2,3]
```

Un programme Haskell hors de GHCi

Example (getLine compilé)

Fichier cm4_1.hs:

```
main = print "bonjour"
```

```
$ ghc --make cm4_1.hs
[1 of 1] Compiling Main             ( cm4_1.hs, cm4_1.o )
Linking cm4_1 ...
$ ./cm4_1
"bonjour"
```

Enchaîner des actions IO

Notation do

```
do
  expression 1
  ...
  expression N
```

permet d'enchaîner les actions IO. Le type est celui de la dernière expression.

Exemple de do

Example (do)

Fichier cm4_2.hs:

```
main = do
  print $ map (*2) [1..10]
  print "OK"
  putStrLn "Ca marche aussi"
```

```
$ ./cm4_2
[2,4,6,8,10,12,14,16,18,20]
"OK"
Ca marche aussi
```

Récupérer des valeurs venant d'actions IO

La fonction getLine

```
Prelude> :t getLine
getLine :: IO String
```

- Action qui récupère une String donnée par l'utilisateur.
- Comment l'utiliser ?

```
Prelude> valeur <- getLine
cosette
Prelude> :t valeur
valeur :: String
Prelude> putStrLn valeur
cosette
```

Example (programme simple)

Fichier cm4_3.hs :

```
main = do
  putStrLn "Entrez votre nom :"
  nom <- getLine
  putStrLn $ "Bonjour, " ++ nom
```

```
$ ./cm4_3
Entrez votre nom :
Jean Valjean
Bonjour, Jean Valjean
```

Mélange pur / impur

Principe

- Les actions IO servent à communiquer avec l'exterieur
- Le code pur fait les calculs
- On peut utiliser des `let` pour stocker les résultats de calculs pur
- Attention à l'indentation ! (voir exemple)

Exemple de mélange pur/impur

Example (let dans un bloc do)

```
Fichier cm4_4.hs :
import Data.Char
main = do
  putStrLn "Entrez votre nom :"
  nom <- getLine
  let nomUpper = map toUpper nom
      nomLower = map toLower nom
  putStrLn $ "Bonjour, " ++ nomUpper ++
             ". Ou alors préférez vous : " ++ nomLower ++ " ?"
$ ./cm4_4
Entrez votre nom :
Cosette
Bonjour, COSETTE. Ou alors préférez vous : cosette ?
```

Interact

- La fonction `interact` prend une fonction de type `String -> String`
- Elle l'applique à `stdin`
- Elle envoie les résultats sur `stdout`

Exemple d'interact

Example (interact)

Fichier cm4_5.hs :

```
import Data.Char

main = interact (map toUpper)
```

```
./cm4_5 < cm4_5.hs
IMPORT DATA.CHAR

MAIN = INTERACT (MAP TOUPPER)
```

Lire des valeurs

- Rappel
  - Tout type faisant partie de la classe `Read` peut être lu depuis une chaîne de caractères
  - On utilise la fonction `read`

```
Prelude> read "[1,3,4]" :: [Int]
[1,3,4]
Prelude> read "3.5" :: Float
3.5
Prelude> read "True" :: Bool
True
```

Exemple plus intéressant avec interact et read

Example (script sommant des valeurs)

```
$ cat cm4_nums
10
20
5

$ cat cm4_nums | ghc -e 'interact ((++"\n") . show .
                         sum .
                         map read . lines)'
35
```

Source : blog http://jasani.org, article du 2008/02/18

readFile, writeFile

```
Prelude> :t readFile
readFile :: FilePath -> IO String
```

- `FilePath = String` → on donne juste le répertoire+nom du fichier
- `readFile` renverra le fichier comme une seule `String`

```
Prelude> :t writeFile
writeFile :: FilePath -> String -> IO ()
```

- `writeFile` prend le contenu du nouveau fichier comme une seule `String`

Exemple

Example (majuscules v2.0)

```
Fichier cm4_6.hs :
import Data.Char
main = do
  contenu <- readFile "cm4_6.hs"
  writeFile "cm4_6_MAJ" (map toUpper contenu)

$ ./cm4_6
$ cat cm4_6_MAJ
IMPORT DATA.CHAR
MAIN = DO
  CONTENU <- READFILE "CM4_6.HS"
  WRITEFILE "CM4_6_MAJ" (MAP TOUPPER CONTENU)
```

Suite...

LES DONNÉES

Types de données

- Textes bruts
- Images
- Sons
- Vidéos

Dans ce cours

- Ensemble d'**objets** ayant des **attributs**
- Objet = enregistrement, point, tuple, entité, instance...
- Attribut = propriété ou caractéristique d'un objet
  - couleur des yeux d'une personne
  - température
  - ...

Exemple

Num. ID	Nom	Prénom	Sexe	Age	Note	Admis?
887123	Dupont	Pierre	M	19	C-	non
887124	Durand	Marie	F	18	A	oui
887125	Lefevre	Marc	M	18	B+	oui

- Lignes = objets
- Colonnes = attributs

Types d'attributs

- Nominaux
  - valeurs distinctes
  - seule opérations autorisées : $=$, $\neq$
  - ex: Num. ID, Sexe
- Ordinaux
  - Ordonnables entre eux ($<$,$>$)
  - ex: Note
- Intervalles
  - Mesurables, intervalle entre valeurs a une signification (+/-)
  - ex: dates, températures en Celsius/Farenheit
- Ratio
  - Ratio ont du sens (*,/)
  - ex: Age, quantités d'argent,...

Attributs discrets / attributs continus

- Attributs discrets
  - Ensemble de valeurs finies
  - Facilement représentables par des entiers
  - ex: ID, codes postaux, notes, booléens...
- Attributs continus
  - Valeurs = nombres réels
  - ex: température, argent,...

Types de données

- Enregistrements : déjà vu avec les étudiants (ex. précédent)
- Matrices
- Documents
- Transactions
- Graphes
- Sequences

Matrices

- Si nombre d'attributs fixes, et attributs représentables par nombres, on peut faire une matrice à n lignes et m colonnes
- n = nombre d'objets
- m = nombre d'attributs

objet	x	y	z
o_1	30.4	12.6	0.9
o_2	43.8	90.6	3.5
o_3	4.8	2.5	9.0

Documents

- Chaque mot distinct devient un attribut
- La valeur de l'attribut peut être la présence/absence du mot, ou son nombre d'occurrences, entre autres.

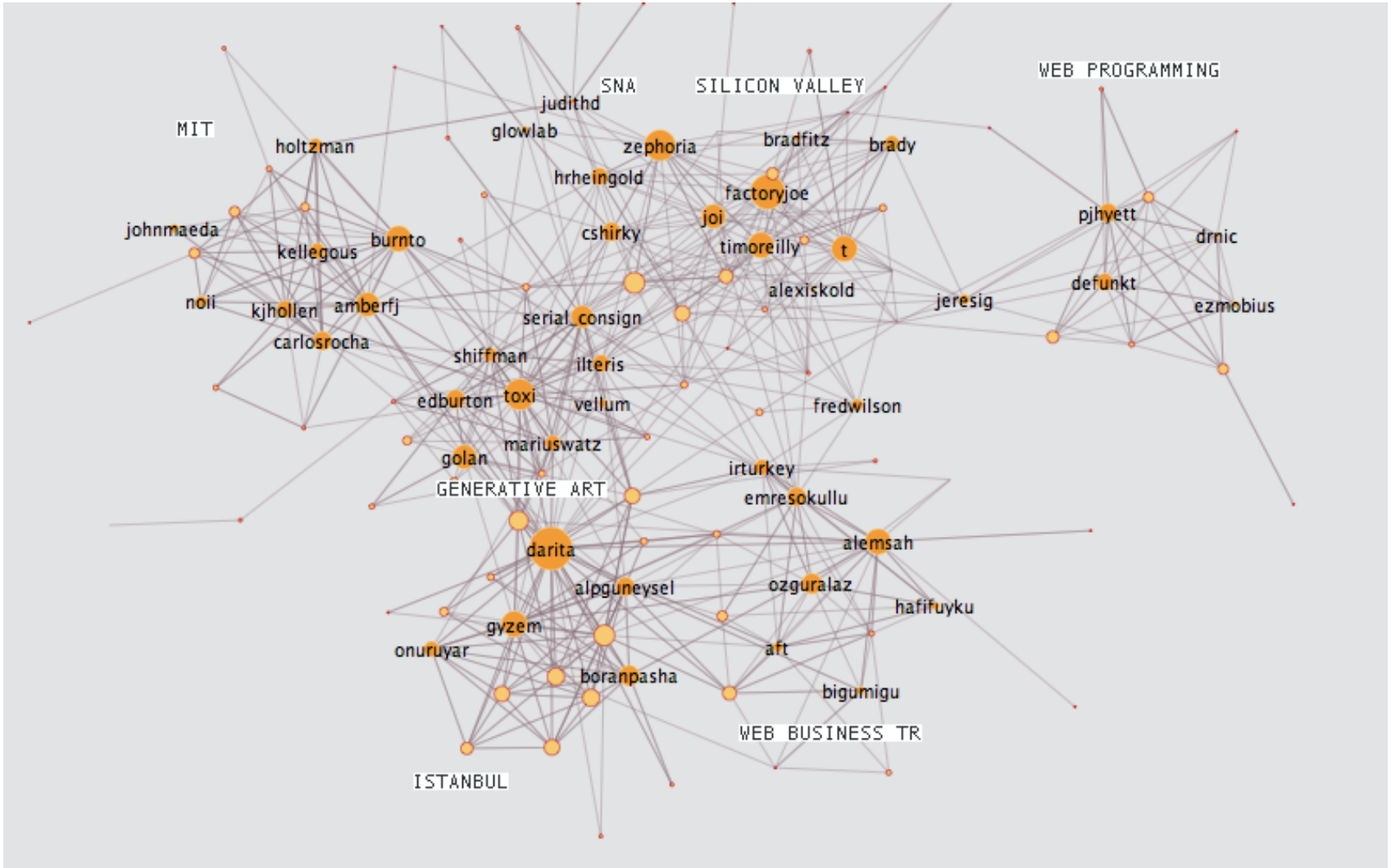
Texte	Jean	Valjean	Cosette	aller	ville	...
Tome 1	207	191	60	52	99	
Tome 2	313	273	265	34	20	

Transactions

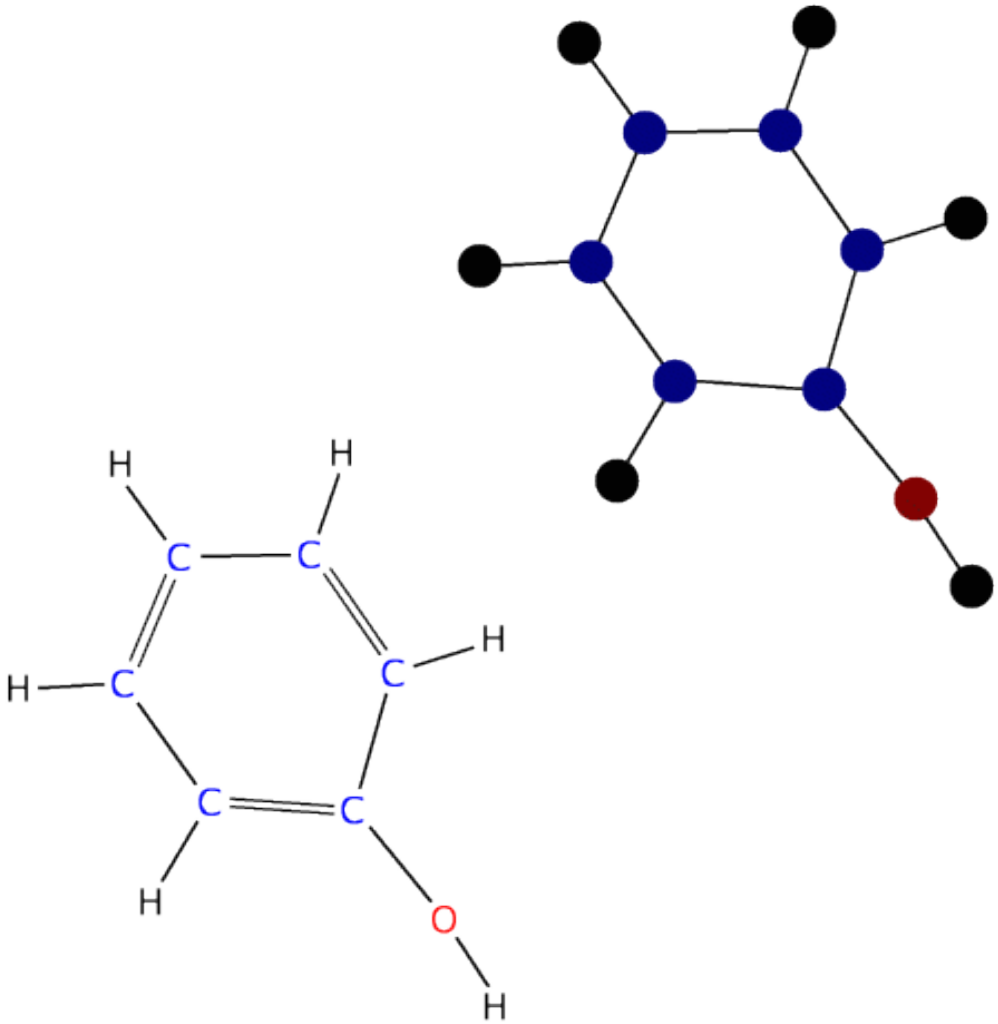
- Chaque objet correspond à une “vente”, et contient un ensemble “d’articles”
- TID = Transaction ID

TID	Articles
1	Pain, Beurre, Chocolat
2	Pain, Crayon, Saucisson, Chocolat
3	Steak, Pain, Bière, Chocolat
4	Vin, Pain, Fromage

Graphes


SNA
SILICON VALLEY
WEB PROGRAMMING
MIT
judithd
glowlab
holtzman
zephoria
bradfitz
brady
hrheingold
factoryjoe
joi
johnmaeda
cshirky
pjhyett
drnic
kellegous
burnto
timoreilly
t
alexiskold
defunkt
noii
kjhollen
amberfj
serial_consign
jeresig
ezmobius
carlosrocha
shiffman
ilteris
edburton
toxi
vellum
fredwilson
mariuswatz
golan
irturkey
GENERATIVE ART
emresokullu
darita
alemsah
alpguneysel
ozguralaz
hafifuyku
gyzem
onuruyar
aft
boranpasha
bigumigu
WEB BUSINESS TR
ISTANBUL

Graphes



Séquences

- Séquences d'évènements

  `(A_login B_login) (C_error) (reboot)`

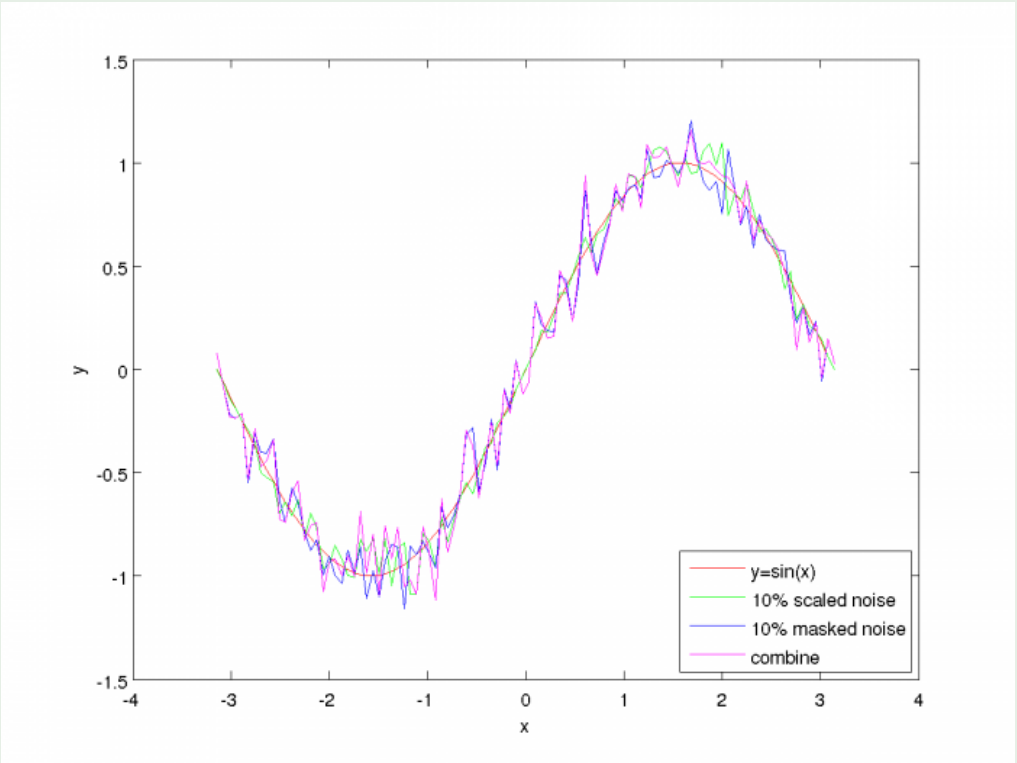
- Séquence d'ADN

  GGTTCCGCCTTCAGCCCCGCGCC...

Bruit

Modification des valeurs originales.

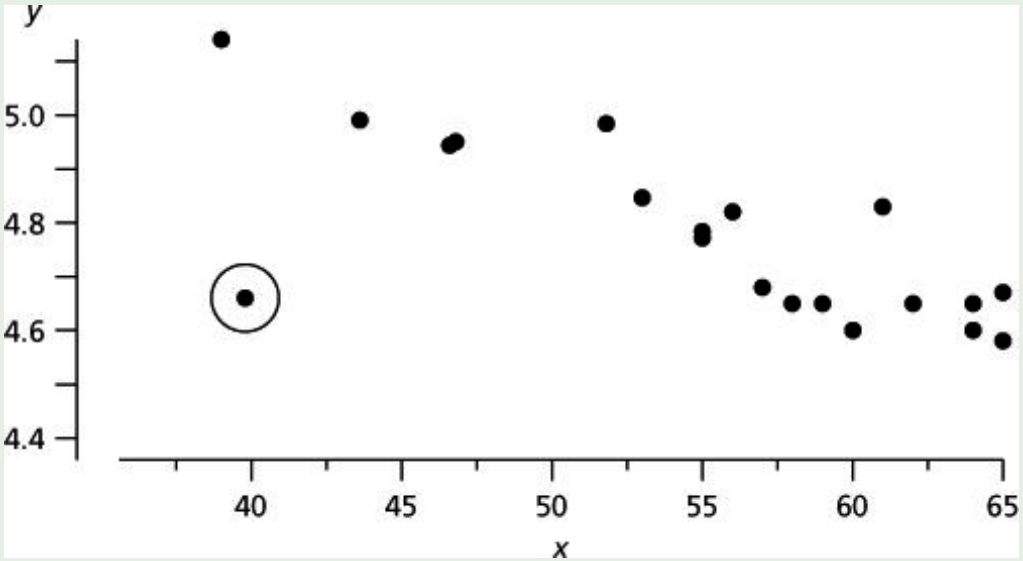
Example (bruit sur sin)



Outliers

Objets dont les caractéristiques sont **très** différentes de la plupart des objets des données.

Example (outliers)



Valeurs manquantes

- Pourquoi ?
  - Informations non récupérées ex: personnes n'ont pas voulu les donner, capteurs defectueux. . .
  - Attributs non utilisables dans tous les cas ex: impôts pour un enfant
- Comment les gérer ?
  - Eliminer les objets correspondants
  - Essayer de deviner les valeurs manquantes
  - Ignorer les valeurs manquantes dans l'algo d'analyse
  - Remplacer par toutes possibilités (+proba)