INFOD203 - Informatique appliquée
Logique et ensembles en R

Matteo Gagliolo
Matteo.Gagliolo@ulb.ac.be

Université libre de Bruxelles

2016-2017

Logique en R

En logique, une **proposition** est une phrase dont on peut se demander si elle est vraie ou fausse.
En mathématique, on a des expressions qui sont vraies ou fausses, genre

$$2 > 3,$$

$$1 + 1 = 2$$

.
Dans logique "classique" que l'on voit ici, et qui est à la base de l'informatique, une proposition est soit vraie, soit fausse (**principe de bivalence**).

Logique en R

En logique, une **proposition** est une phrase dont on peut se demander si elle est vraie ou fausse.
En mathématique, on a des expressions qui sont vraies ou fausses, genre

$$2 > 3,$$

$$1 + 1 = 2$$

.
Dans logique "classique" que l'on voit ici, et qui est à la base de l'informatique, une proposition est soit vraie, soit fausse (**principe de bivalence**).
Il y a différents conventions pour indiquer la *valeur de vérité* d'une proposition.

langage	logique (fr)	logique (en)	algèbre de Boole	R
vrai	*V*	*T*	1	TRUE
faux	*F*	*F*	0	FALSE

Logique en R

En logique, une **proposition** est une phrase dont on peut se demander si elle est vraie ou fausse.
En mathématique, on a des expressions qui sont vraies ou fausses, genre

$$2 > 3,$$

$$1 + 1 = 2$$

.
Dans logique "classique" que l'on voit ici, et qui est à la base de l'informatique, une proposition est soit vraie, soit fausse (**principe de bivalence**).
Il y a différents conventions pour indiquer la *valeur de vérité* d'une proposition.

langage	logique (fr)	logique (en)	algèbre de Boole	R
vrai	*V*	*T*	1	`TRUE`
faux	*F*	*F*	0	`FALSE`

Dans la théorie de l'information, une valeur de vérité (ou **booléenne**) correspond à un **bit**. Dans le langage R, c'est un valeur de classe `"logical"`.

Relations

Ce sont des relations entre nombres, qui peuvent donc être utilisés pour former des propositions (vraies ou fausses). Dans `R`, le résultat de ces expressions est de classe `"logical"` (`TRUE` ou `FALSE`).

Maths	`R`
$x = y$	`x == y`
$x \neq y$	`x != y`
$x > y$	`x > y`
$x < y$	`x < y`
$x \geq y$	`x >= y`
$x \leq y$	`x <= y`

Assignation vs égalité

ATTENTION : Dans les maths on utilise $=$ à la fois comme relation $(1 + 1 = 2)$ et pour assigner un valeur à une variable $(x = 1)$.

Dans la programmation, assignation (= ou <-) et égalité (==) sont deux choses bien distinctes.

Malheureusement `R` ne nous aide pas : c'est a nous de faire attention !

Opérateurs logiques

Ce sont des opérateurs qui combinent une ou deux variables booléennes en donnant comme résultat une variable booléenne. Soient a, b deux propositions, ces opérateurs sont définis :

nom	notation	on lit	résultat	R
Négation	$\neg a$	*non a*	vraie si a fausse, et vice versa	!a

Opérateurs logiques

Ce sont des opérateurs qui combinent une ou deux variables booléennes en donnant comme résultat une variable booléenne. Soient a, b deux propositions, ces opérateurs sont définis :

nom	notation	on lit	résultat	R
Négation	$\neg a$	*non* a	vraie si a fausse, et vice versa	`!a`
Conjonction	$a \wedge b$	a *et* b	vraie si a *et* b sont vraies	`a & b`

Opérateurs logiques

Ce sont des opérateurs qui combinent une ou deux variables booléennes en donnant comme résultat une variable booléenne. Soient a, b deux propositions, ces opérateurs sont définis :

nom	notation	on lit	résultat	R
Négation	$\neg a$	*non* a	vraie si a fausse, et vice versa	`!a`
Conjonction	$a \wedge b$	a *et* b	vraie si a *et* b sont vraies	`a & b`
Disjonction	$a \vee b$	a *ou* b	a est vraie, *ou* b est vraie, ou bien les deux	`a | b`

Ensembles

Tout simplement, un ensemble est une **collection d'objets**, défini uniquement par les objets qui lui appartiennent.
Le but d'un ensemble est de pouvoir considérer les objets qui le forment comme une **unité**.
Pour indiquer qu'un objet x appartient à un ensemble A, on écrit $x \in A$.
Dans le cas contraire, on écrit $x \notin A$.
Dans le discours, on peut lire $x \in A$ ainsi :

- x appartient à A
- x est un élément de A
- x est un membre de A
- x est un A

Ensembles

Tout simplement, un ensemble est une **collection d'objets**, défini uniquement par les objets qui lui appartiennent.
Le but d'un ensemble est de pouvoir considérer les objets qui le forment comme une **unité**.
Pour indiquer qu'un objet x appartient à un ensemble A, on écrit $x \in A$.
Dans le cas contraire, on écrit $x \notin A$.
Dans le discours, on peut lire $x \in A$ ainsi :

- x appartient à A
- x est un élément de A
- x est un membre de A
- x est un A

$x \in A$ est une proposition, donc elle est soit vraie soit fausse.

Ensembles

Tout simplement, un ensemble est une **collection d'objets**, défini uniquement par les objets qui lui appartiennent.
Le but d'un ensemble est de pouvoir considérer les objets qui le forment comme une **unité**.
Pour indiquer qu'un objet x appartient à un ensemble A, on écrit $x \in A$.
Dans le cas contraire, on écrit $x \notin A$.
Dans le discours, on peut lire $x \in A$ ainsi :

- x appartient à A
- x est un élément de A
- x est un membre de A
- x est un A

$x \in A$ est une proposition, donc elle est soit vraie soit fausse.
Sa négation est $x \notin A$:

- Si $x \in A$ est vraie, alors $x \notin A$ est fausse.
- Si $x \in A$ est fausse, alors $x \notin A$ est vraie.

Ensembles en R

On n'a pas une classe de données correspondant aux ensembles en R.
Par contre, on peut considérer un vecteur comme définissant un ensemble, celui de ses éléments (sans ordre ni répétition).
Pour exprimer la relation d'appartenance $\in$ on utilise en R le symbole `%in%`, ou la fonction équivalente `is.element`.
L'expression `x %in% A`

- a valeur `TRUE` si l'élément `x` appartient à l'ensemble des éléments de `A`
- a valeur `FALSE` si l'élément `x` appartient à l'ensemble des éléments de `A`

Il n'y a pas d'opérateur pour représenter la relation $\notin$, qui est donc obtenue en niant l'appartenance.

Exemples : `6 %in% 1:5` vaut `FALSE`
`"un" %in% c("un", "deux", "trois")` vaut `TRUE`

Ensembles en R

On n'a pas une classe de données correspondant aux ensembles en R.
Par contre, on peut considérer un vecteur comme définissant un ensemble, celui de ses éléments (sans ordre ni répétition).
Pour exprimer la relation d'appartenance $\in$ on utilise en R le symbole `%in%`, ou la fonction équivalente `is.element`.
L'expression `x %in% A`

- a valeur `TRUE` si l'élément `x` appartient à l'ensemble des éléments de `A`
- a valeur `FALSE` si l'élément `x` appartient à l'ensemble des éléments de `A`

Il n'y a pas d'opérateur pour représenter la relation $\notin$, qui est donc obtenue en niant l'appartenance. En somme :

notation	R (opérateur)	R (fonction)
$x \in A$	`x %in% A`	`is.element(x, A)`
$x \notin A$, soit $\neg(x \in A)$	`!(x %in% A)`	`!is.element(x, A)`

Exemples : `6 %in% 1:5` vaut `FALSE`
`"un" %in% c("un", "deux", "trois")` vaut `TRUE`

Ensembles en R

Une particularité de la relation d'appartenance `x %in% A` en R est que `x` peut être aussi un vecteur ! Dans ce cas, le résultat est

- un vecteur de valeurs booléens (classe `"logical"`)
- ayant la même taille de `x`
- contenant `TRUE` en correspondance des valeurs `x[i]` tels que `x[i] %in% A`
- `FALSE` en correspondance des valeurs `x[i]` tels que `!(x[i] %in% A)`

Exemples :
`c(2,7) %in% 1:5` vaut `c(TRUE, FALSE)`
`c("j'ai", "un", "chien") %in% c("trois", "deux", "un", "zero")` vaut `c(FALSE, TRUE, FALSE)`

Cardinal

Le **cardinal** d'un ensemble est simplement le nombre d'éléments qui lui appartiennent, donc sa taille.
C'est donc un opérateur unaire, qui donne comme résultat un nombre entier naturel, a partir d'un ensemble comme opérande.
Notation : $\#(A)$. Alternatives : $|A|$ ou $\text{card}(A)$ ou $n(A)$.
En `R` : on n'a pas une fonction, mais on peut d'abord utiliser `unique` pour éliminer toute répétition d'éléments, et `length` pour compter les éléments : `length(unique(A))` nous donne le cardinal de `A`.
Exemple : `length(unique(c(1,2,1,2,1)))` vaut 2.

Prédicats et ensembles : `all` et `any`

Les fonctions `all` et `any` prennent comme argument un vecteur de valeurs booléens et rendent un seul valeur qui est le résultat d'une opération logique entre tous les éléments :

- `all(X)` est `TRUE` si tout élément de `X` est `TRUE`
- `any(X)` est `TRUE` si au moins un élément de `X` est `TRUE`

Donc : `!all(X) == any(!X)`, et `!any(X) == all(!X)`
Si $A = \{x_1, \ldots, x_N\}$ est un ensemble de N éléments, $\mathcal{P}(x)$ est une proposition contenant la variable x, et `P` est la correspondante fonction en `R` :

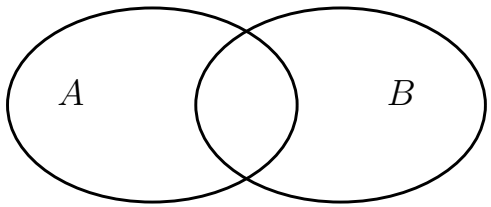
propositions	prédicats	R
$\mathcal{P}(x_1) \wedge \mathcal{P}(x_2) \ldots \wedge \mathcal{P}(x_N)$	$\forall x \in A\ \mathcal{P}(x)$	`all(P(A))`
$\mathcal{P}(x_1) \vee \mathcal{P}(x_2) \ldots \vee \mathcal{P}(x_N)$	$\exists x \in A\ \mathcal{P}(x)$	`any(P(A))`

Exemples : `all(1:5 > 0)` vaut `TRUE`
`any(1:5 < c(1,3,2,4,5))` vaut `TRUE` car $2 < 3$.
Pour ce deux fonctions on peut aussi passer le paramètre `na.rm=TRUE` pour exclure les `NA` de l'input : essayez `all(c(TRUE,NA), na.rm=TRUE)` avec et sans `na.rm`.

Relations entre ensembles

Deux ensembles A et B **se rencontrent** s'ils ont au moins un élément en commun.
On indique ça ainsi :

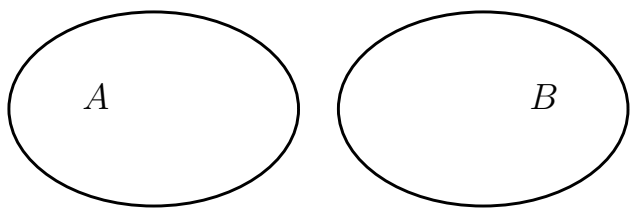
$$A \cap B \neq \varnothing,$$



où $\varnothing$ est l'*ensemble vide*.
En `R` : `any(A %in% B)`.

Deux ensembles A et B sont **disjoints** s'ils ont aucun élément en commun.
On indique ça ainsi :

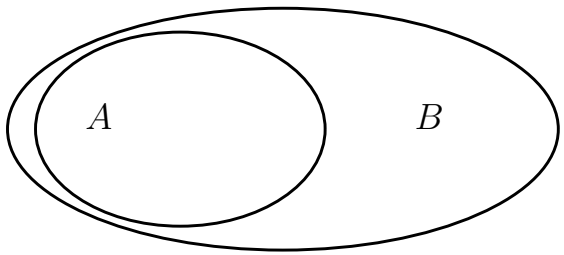
$$A \cap B = \varnothing,$$



où $\varnothing$ est l'*ensemble vide*.
En `R` : `!any(A %in% B)`.

Relations entre ensembles : Inclusion

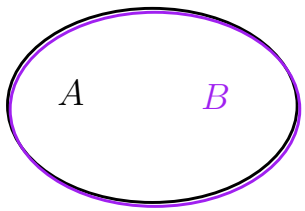
Un ensemble A est **inclus** dans un ensemble B si tout élément de A appartient aussi à B.
On indique ça ainsi : $A \subset B$, ou bien $B \supset A$



En `R` on n'a pas une fonction ad hoc pour représenter l'inclusion, mais on peut simplement la représenter en utilisant `all` et `%in%` :
$A \subset B$ est vraie si tout élément de A est aussi élément de B, ce que on peut écrire ainsi : `all(A %in% B)`.

Relations entre ensembles : Équivalence

Un ensemble A est **équivalent** à un ensemble B si tout élément de A appartient aussi à B **et** tout élément de B appartient à A.
On indique cela ainsi : $A = B$.

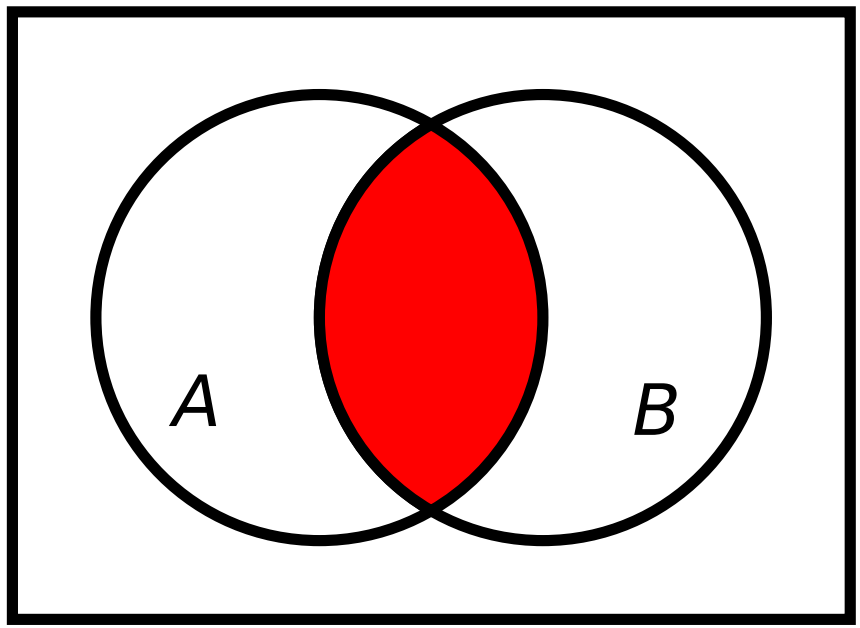


Dans ce cas, on a une fonction à valeur booléenne : on peut écrire $A = B$ ainsi : `setequal(A,B)`.
Remarque : cette fonction considère ses arguments comme des ensembles, donc ni l'ordre ni les répétitions sont pris en compte.
Exemples : `setequal(1:3,c(1,2,3,2,1,2))` vaut `TRUE`
`setequal(c("a", "b"), c("b", "a"))` vaut `TRUE`
`setequal(c("a", "c"), c("a", "b"))` vaut `FALSE`

Opérateur : intersection

Les **opérateurs** suivants nous permettent de définir des nouveaux ensembles à partir des ensembles existants.
L'*intersection* de A et B est l'ensemble qui contient seulement leurs éléments en commune, donc

$$A \cap B = \{x \mid (x \in A) \wedge (x \in B)\}$$



en rouge le résultat de l'opération

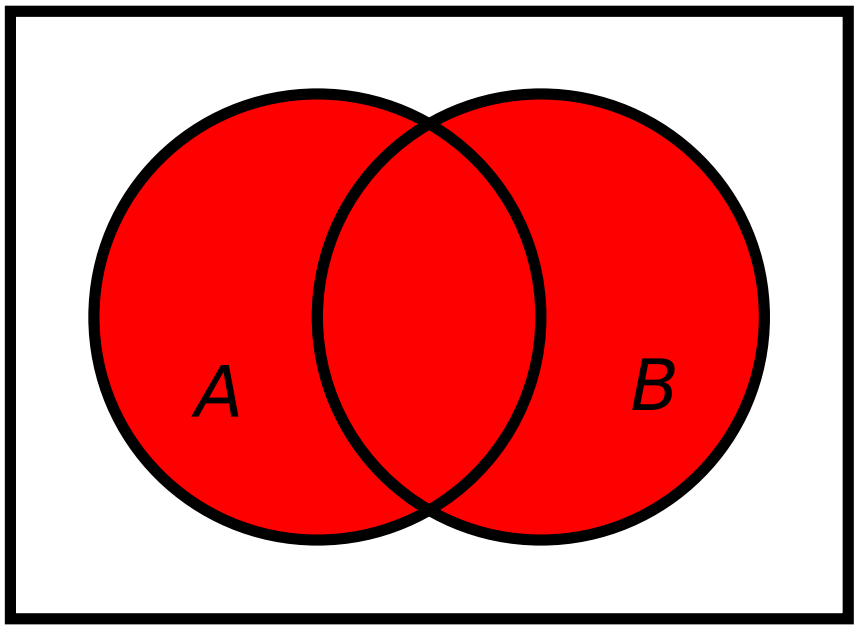
En R : `intersect(A, B)`.
Exemple : `intersect(1:3, 2:5)` vaut `2:3`.

Opérateur : réunion

La *réunion* de A et B est l'ensemble qui contient tous leurs éléments, donc

$$A \cup B = \{x \mid (x \in A) \vee (x \in B)\}$$

Ce n'est donc pas "soit A, soit B", mais "soit A, soit B, soit les deux"



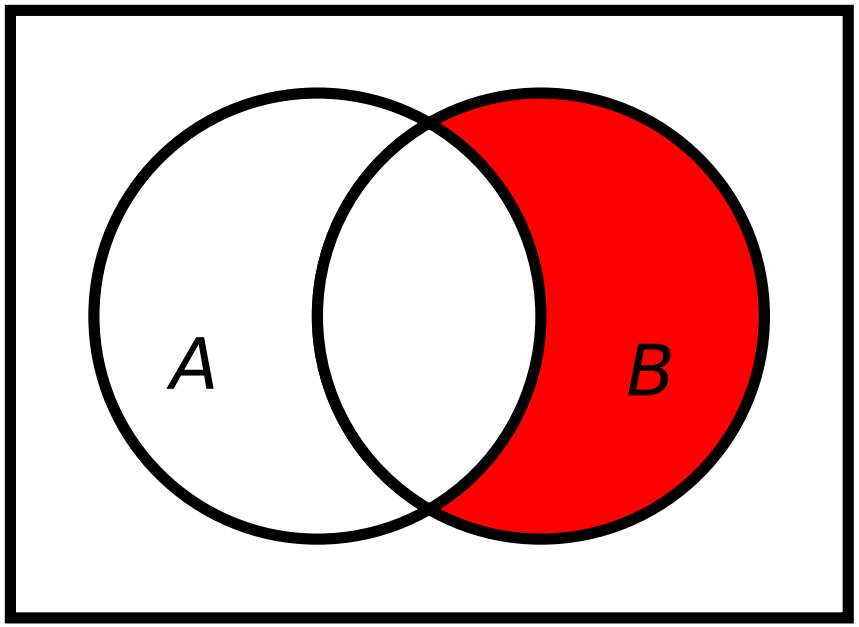
en rouge le résultat de l'opération

En `R` : `union(A, B)`.
Exemple : `union(1:3, 2:5)` vaut `1:5`.

Opérateur : différence

La *difference* entre B et A est l'ensemble des éléments de B qui n'appartiennent pas à A.

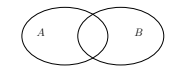
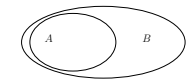
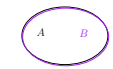
Notation : barre oblique inversée (eng : backslash)
$B \setminus A = \{(x \in B) \mid (x \notin A)\}$
Notation alternative : $B - A$ (ce qui reste à B si on enlève A).
Dans ce cas, l'ordre importe !
$B \setminus A \neq A \setminus B$



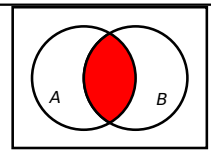
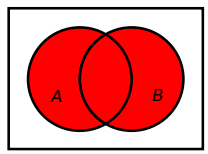
en rouge le résultat de l'opération

En `R` : `setdiff(B, A)`.
Exemple : `setdiff(1:3, 2:5)` vaut `1`, `setdiff(2:5, 1:3)` vaut `4:5`.

Ensembles : résumé relations

	ensembles	logique	R	
relations	$(A \cap B \neq \varnothing)$	$\exists x((x \in A) \wedge (x \in B))$	`any(A %in% B)`	
	$(A \cap B = \varnothing)$	$(x \in A) \Rightarrow (x \notin B)$	`!any(A %in% B)`	
	$(A \subset B)$	$((x \in A) \Rightarrow (x \in B))$	`all(A %in% B)`	
	$(A = B)$	$((x \in A) \Leftrightarrow (x \in B))$	`setequal(A,B)`	

Ensembles : résumé opérateurs

	ensembles	logique	R	
opérateurs	$A \cap B$	$\{x \mid (x \in A) \wedge (x \in B)\}$	`intersect(A, B)`	
	$A \cup B$	$\{x \mid (x \in A) \vee (x \in B)\}$	`union(A, B)`	
	$B \setminus A$	$\{x \mid (x \in B) \wedge (x \notin A)\}$	`setdiff(B, A)`	