



Ateliers R³



MBB
Institut des Sciences de l'Evolution-Montpellier

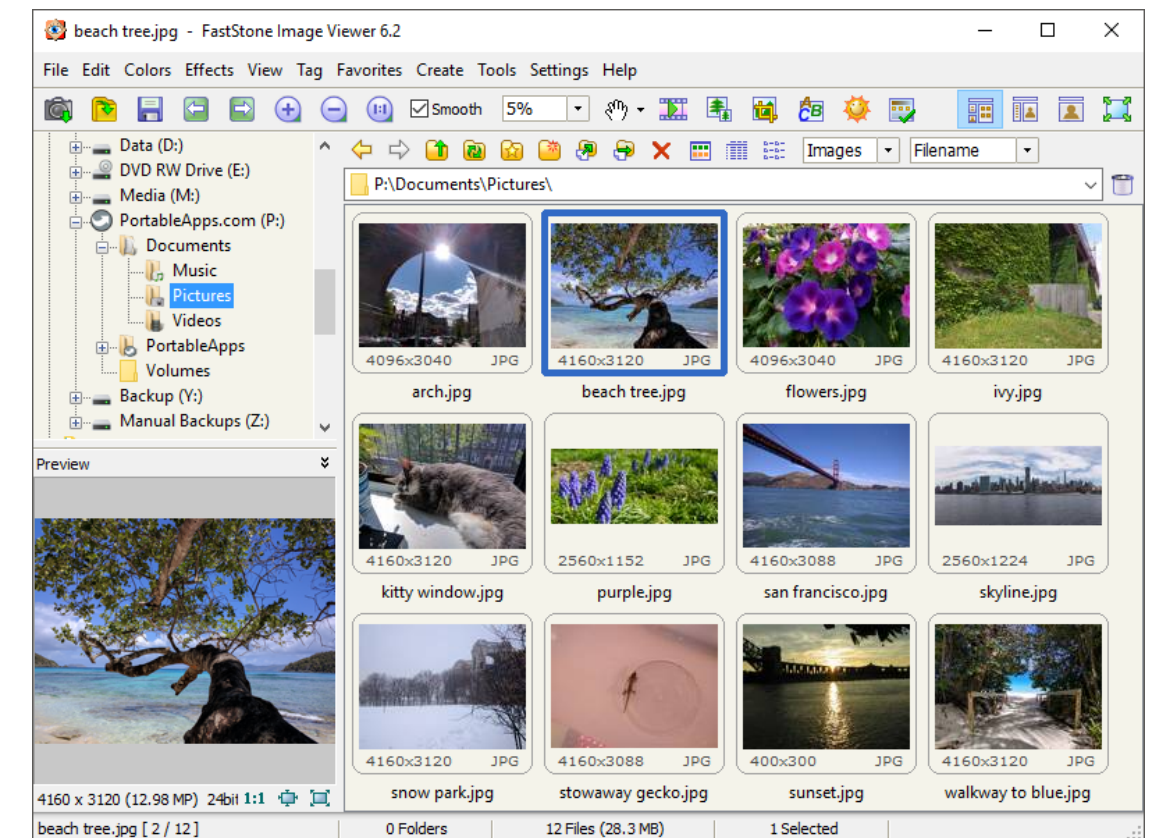
Session 1 - R, les bases

Pourquoi utiliser R ?

Pourquoi utiliser R ?

Vous manipulez des données sur ordinateur en réalisant beaucoup d'opérations répétitives

Vous êtes responsable d'un gros jeu de données et il vous faut un langage pour traiter des données



Pourquoi utiliser R ?

Vous manipulez des données sur ordinateur en réalisant beaucoup d'opérations répétitives

Vous êtes responsable d'un gros jeu de données et il vous faut un langage pour traiter des données

Vous avez besoin de faire des statistiques sur vos données, ou les représenter dans des graphes, documents ou applications parfois complexes

Vous faites des opérations qui impliquent beaucoup d'opérations mathématiques

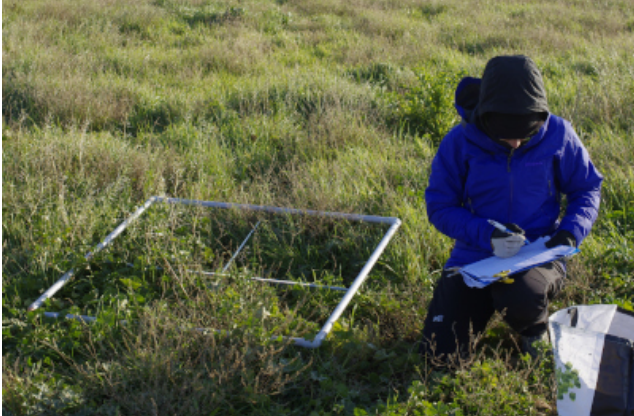
Pourquoi utiliser R ?

Vous travaillez avec des gens qui utilisent R !

Données réelles

8314	14352	46045	51950	76335	85327	113183	114368	114793	135976	140763	159579	162142	164119	164782	173018	173069	174223	174966	176279
A	T	G	G	G	G	G	A	C	C	A	A	A	T	C	C	T	C	C	C
A	T	G	G	G	G	G	A	C	C	A	A	G	T	C	C	T	C	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C	C

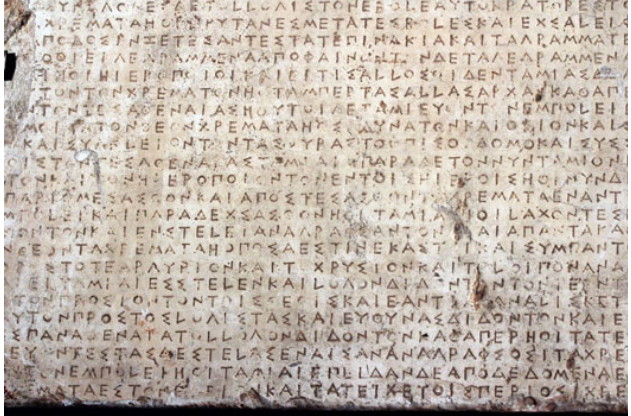
Données génétiques



Des relevés scientifiques



Des photos

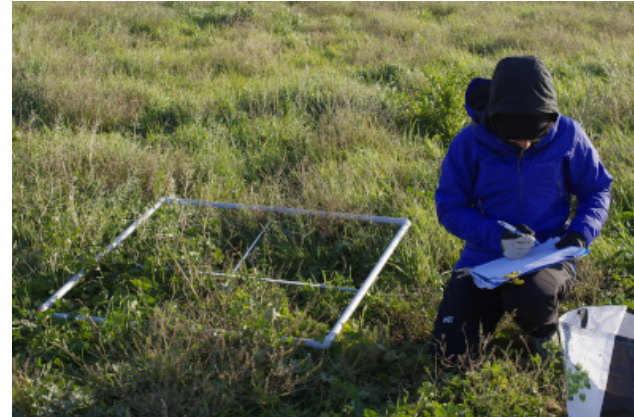


Du texte

Données réelles

8314	14352	46045	51950	76335	85327	113183	114368	114793	135976	140763	159579	162142	184119	184782	173018	173069	174223	174966
A	T	G	G	G	G	G	A	C	C	A	A	A	T	C	C	T	C	C
A	T	G	G	G	G	G	A	C	C	A	A	G	T	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C

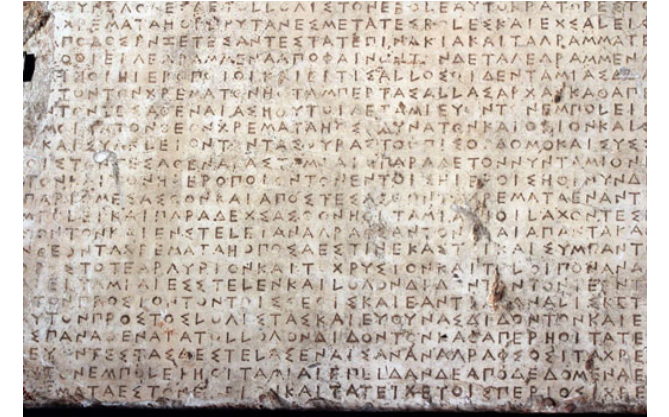
Données génétiques



Des relevés scientifiques



Des photos



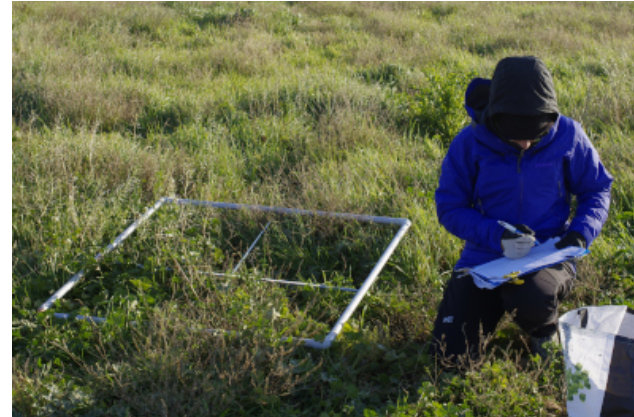
Du texte

Représentées par des
Structures des données

Données réelles

8314	14352	46045	51950	76335	85327	113183	114368	114793	135976	140763	159579	162142	164119	164782	173018	173069	174223	174966
A	T	G	G	G	G	G	A	C	C	A	A	A	T	C	C	T	C	C
A	T	G	G	G	G	G	A	C	C	A	A	G	T	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C

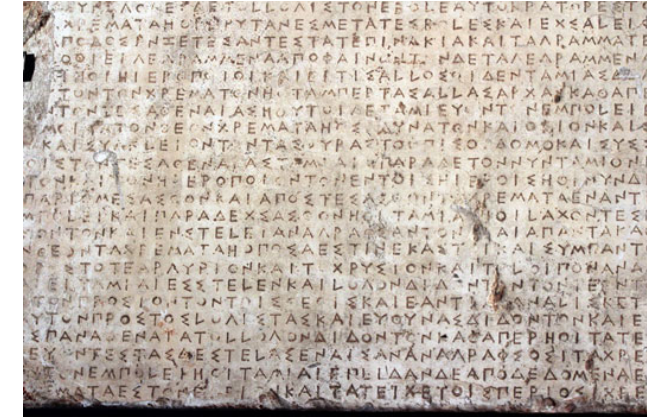
Données génétiques



Des relevés scientifiques



Des photos



Du texte

Représentées par des
Structures des données

Des opérations à effectuer, i.e un
Programme informatique

Données réelles

8314	14352	46045	51950	76335	85327	113183	114368	114793	135976	140763	159579	162142	164119	164782	173018	173069	174223	174966
A	T	G	G	G	G	G	A	C	C	A	A	A	T	C	C	T	C	C
A	T	G	G	G	G	G	A	C	C	A	A	G	T	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C

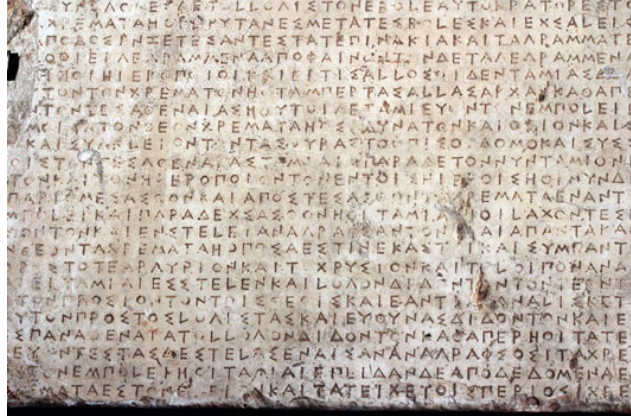
Données génétiques



Des relevés scientifiques



Des photos



Du texte

Représentées par des
Structures des données

Des opérations à effectuer, i.e un
Programme informatique

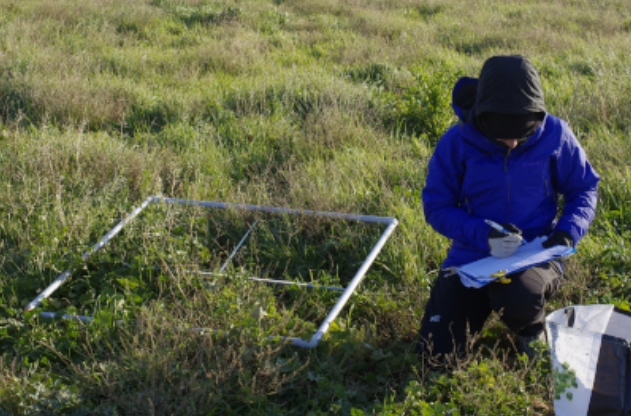


Résultat
Analyse stats
Graphes,
Documents textes
Envoi d'email,
etc...

Données réelles

8314	14352	46045	51950	76335	85327	113183	114368	114793	135976	140763	159579	162142	164119	164782	173018	173069	174223	174966
A	T	G	G	G	G	G	A	C	C	A	A	A	T	C	C	T	C	C
A	T	G	G	G	G	G	A	C	C	A	A	G	T	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C
A	T	A	A	A	G	G	G	A	C	C	A	A	G	C	C	T	C	C

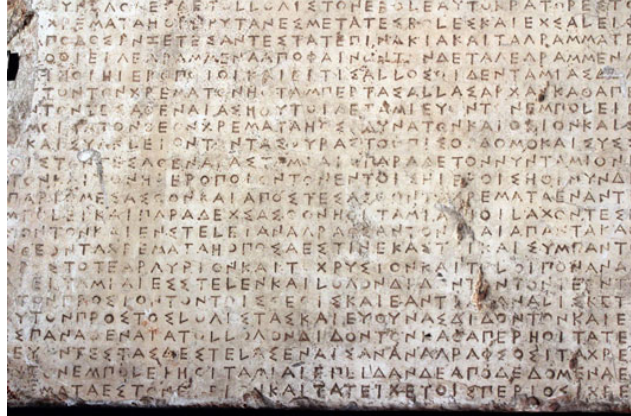
Données génétiques



Des relevés scientifiques



Des photos



Du texte



Des opérations à effectuer, i.e un **Programme informatique**

Représentées par des **Structures des données**

Résultat
Analyse stats
Graphes,
Documents textes
Envoi d'email,
etc...

Représenter des données dans R: Structures de données

Un **type** de données...

Des **nombres**: 1 0 1e-10 1.414214

Des **chaînes de caractères**: "Ça doit être les burgers"
"I <3 ISEM"



Un **type** de données...

Des **nombres**: 1 0 1e-10 1.414214

Des **chaînes de caractères**: "Ça doit être les burgers"
"I <3 ISEM"

Des **booléens**: TRUE FALSE



Un **type** de données...

Des **nombres**: 1 0 1e-10 1.414214

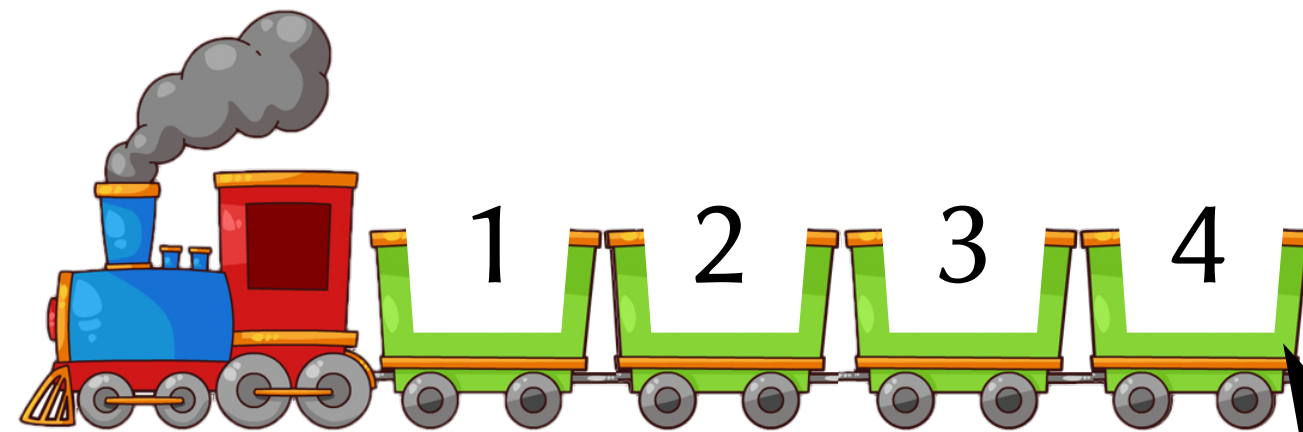
Des **chaînes de caractères**: "Ça doit être les burgers"
"I <3 ISEM"

Des **booléens**: TRUE FALSE

Des **facteurs**: "Janvier" (parmi "Janvier", "Février", ... "Décembre")
"Moyen" (parmi "Gros", "Moyen", "Petit")

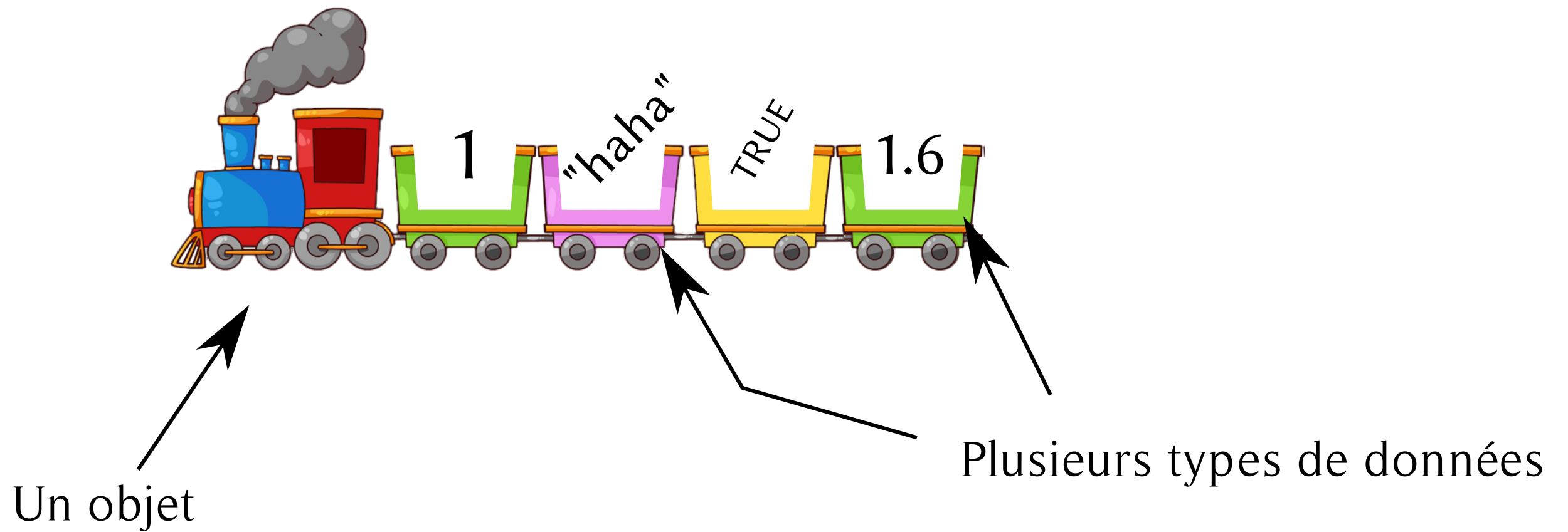
(et bien d'autres)





Un objet

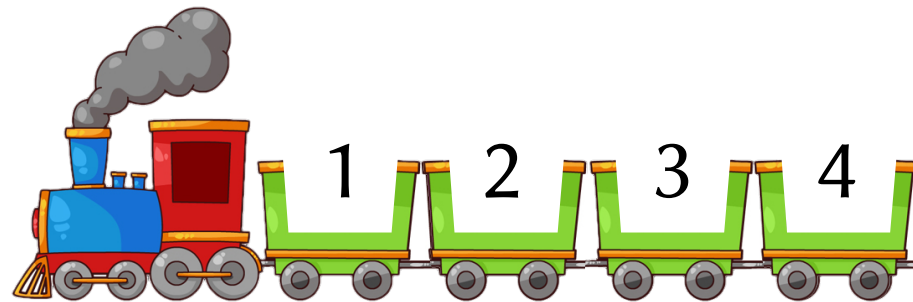
Un type de données



Objets à **une dimension**

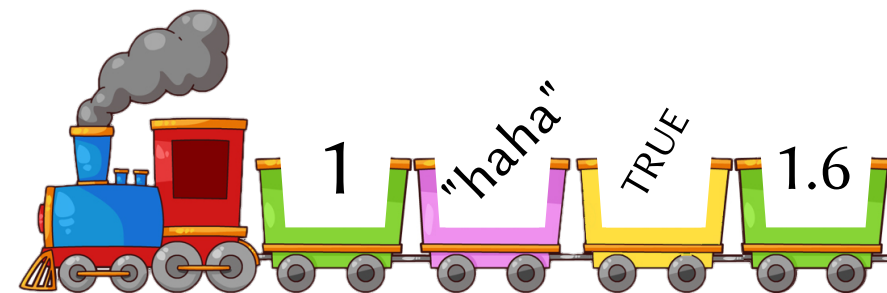
Des données contenues dans des **objets**

Le **vecteur**: un ensemble de valeurs du même type



`c(1, 2, 3, 4)`

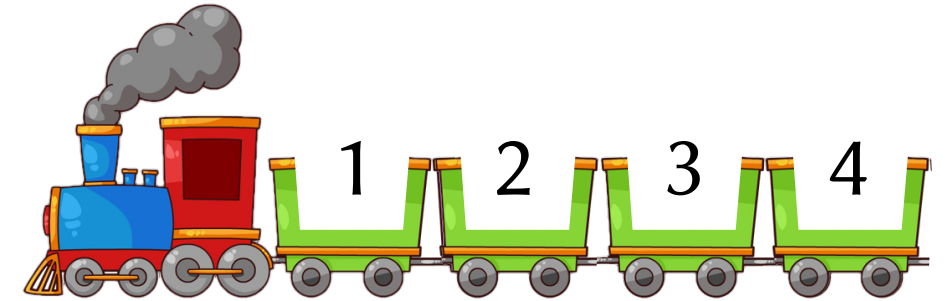
La **liste**: un ensemble de valeurs de types possiblement différents



`list(1, "haha", TRUE, 1.6)`



Le **vecteur**: un ensemble de valeurs du même type



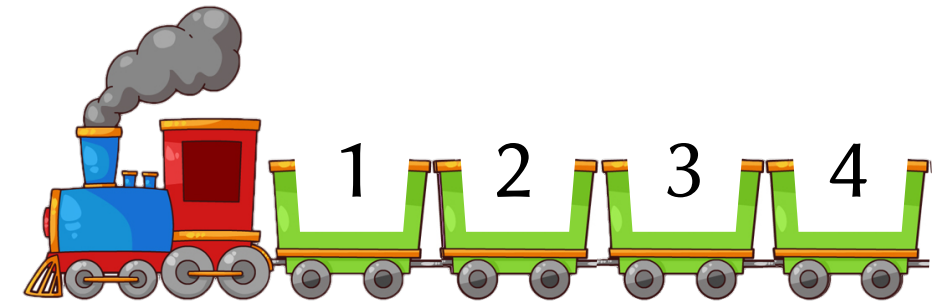
Création: `vec <- c(1, 2, 3, 4)`

Attribution de noms: `names(vec) <- c("a", "b", "c", "d")`

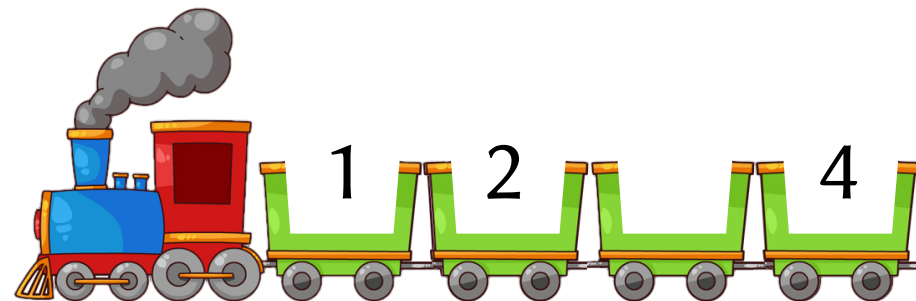
Sélection d'un sous-ensemble: `vec[<...>]`

Remplacement/ajout d'un sous-ensemble: `vec[<...>] <- <...>`

Le **vecteur**: un ensemble de valeurs du même type



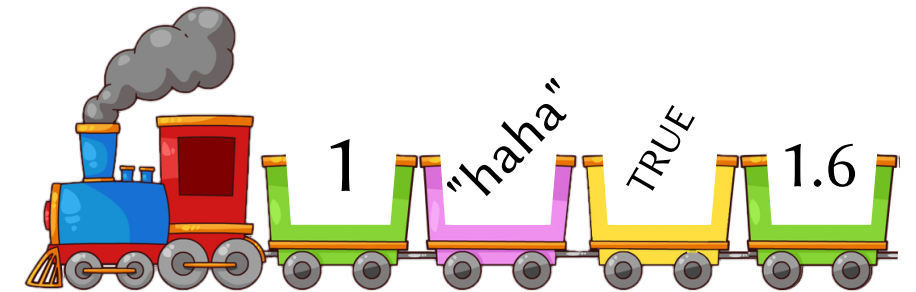
Les **valeurs manquantes**:



`c(1, 2, NA, 4)`



La **liste**: un ensemble de valeurs de types différents



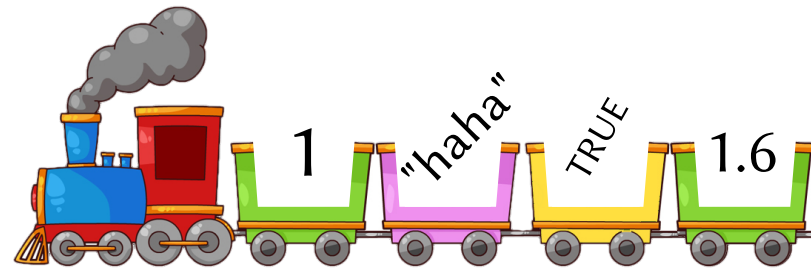
Création: `lis <- list(1, "haha", TRUE, 1.6)`

Attribution de noms: `names(lis) <- c("a", "b", "c", "d")`

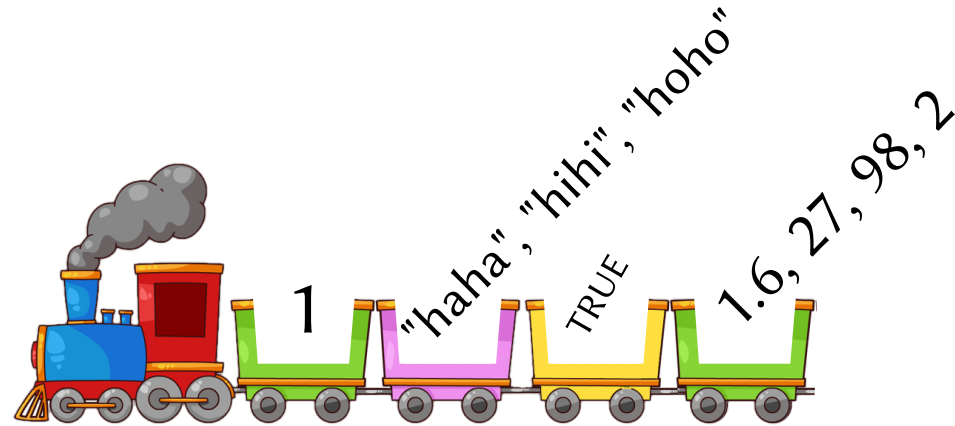
Sélection d'un sous-ensemble: `lis[[<...>]]`

Remplacement/ajout d'un élément: `lis[[<...>]] <- <...>`

La **liste**, c'est très flexible: elle peut contenir des vecteurs (ou d'autres listes)



```
list(1, "haha", TRUE, 1.6)
```



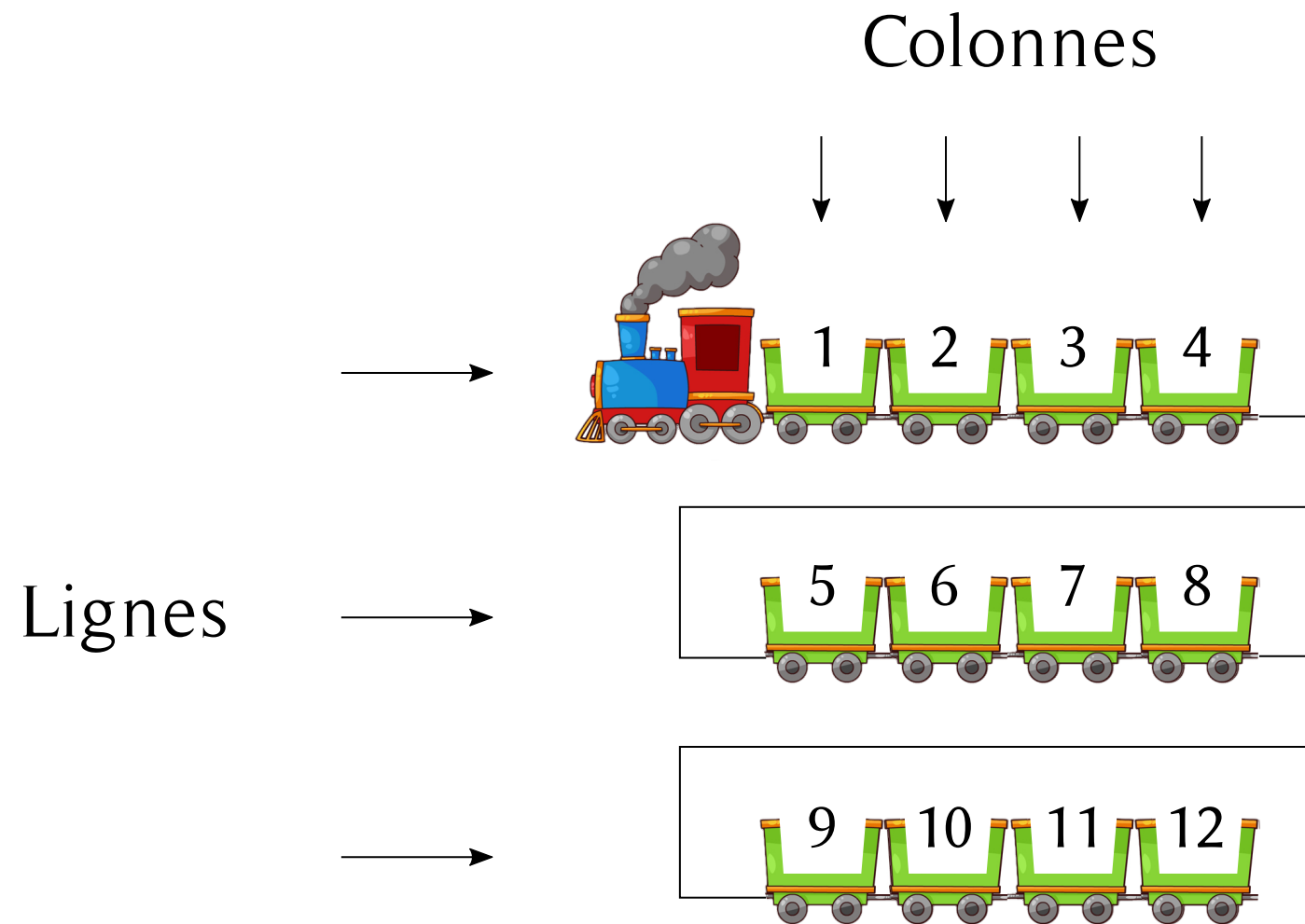
```
list(1,  
      c("haha", "hihi", "hoho"),  
      TRUE,  
      c(1.6, 27, 98, 2))
```

```
list(1,  
      list("a", "b", "d"),  
      TRUE,  
      list(list(1, 2), 4))
```

Des objets à **deux** dimensions

Des données contenues dans des **objets**

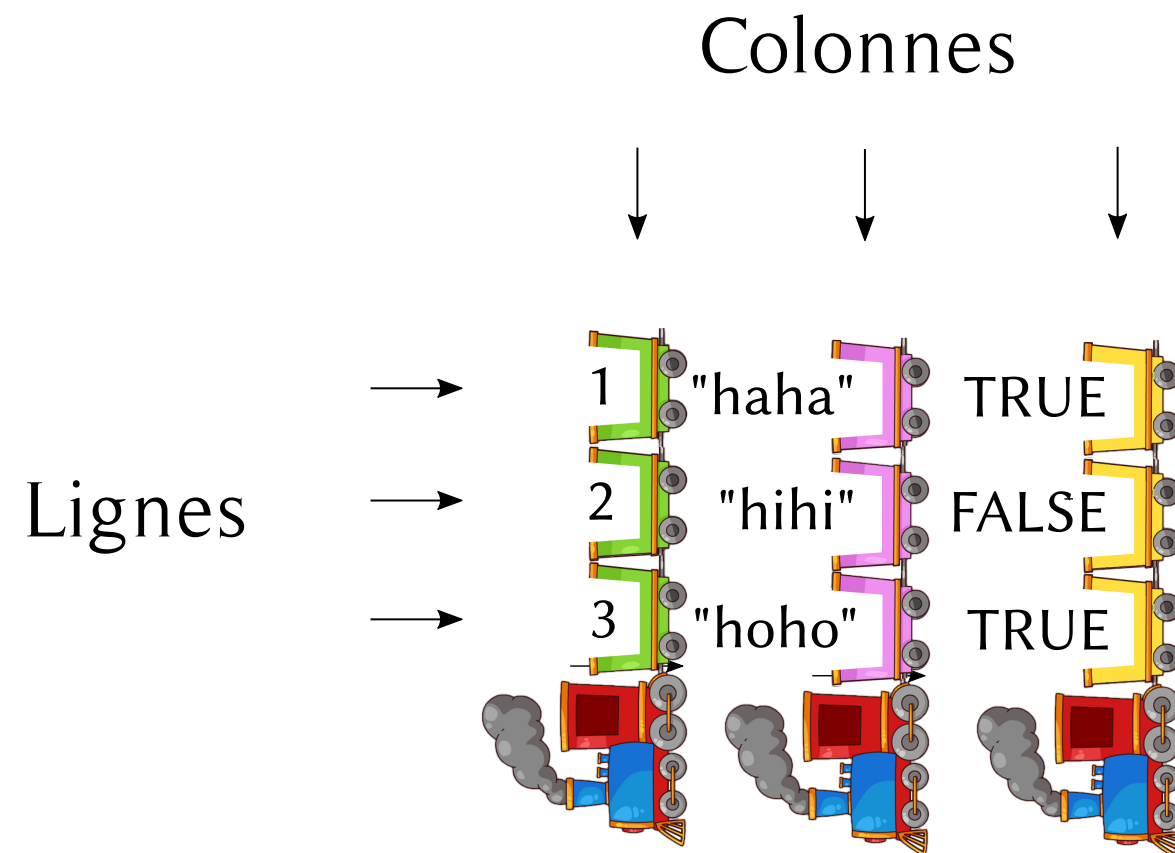
La **matrice**: un ensemble de valeurs à deux dimensions du même type



```
matrix(c(1, 2, 3, 4,  
        5, 6, 7, 8,  
        9, 10, 11, 12),  
       ncol = 4, nrow = 3,  
       byrow = TRUE)
```


Des données contenues dans des **objets**

Le **data.frame**: un ensemble de valeurs à deux dimensions de type différents



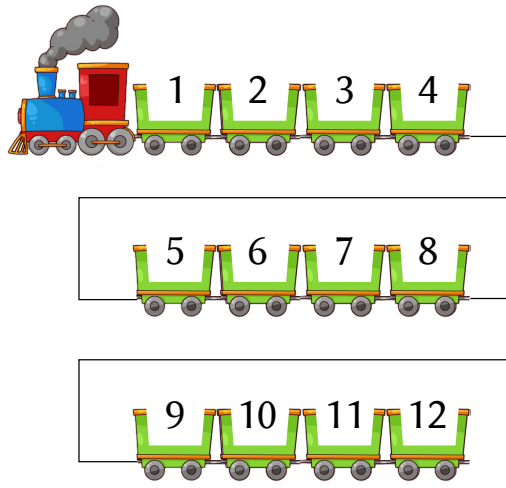
= un assemblage de **vecteurs**

```
data.frame(c(1, 2, 3),  
           c("haha", "hihi", "hoho"),  
           c(TRUE, FALSE, TRUE))
```



Objets 2d

La **matrice**: un ensemble de valeurs à deux dimensions du même type



Création:

```
matrix(c(1, 2, 3, 4,  
        5, 6, 7, 8,  
        9, 10, 11, 12),  
       ncol = 4, nrow = 3,  
       byrow = TRUE)
```

Nommage

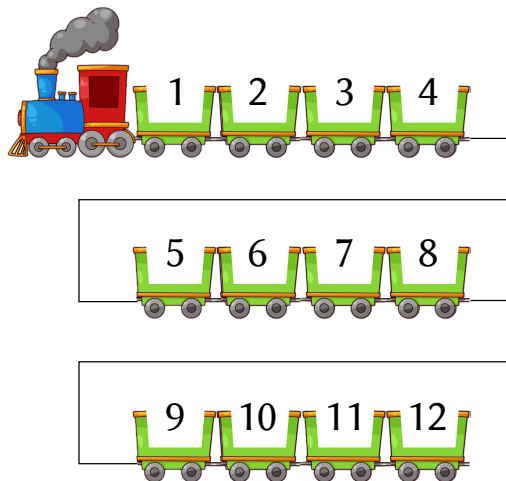
```
lignes: rownames(mat) <- c("a", "b", "c", "d")  
colonnes: colnames(mat) <- c("a", "b", "c", "d")
```

Sélection/remplacement de sous-ensembles:

```
lignes: mat[<...>, ]  
colonnes: mat[<...>, ]  
cellules: mat[<...>, <...>]
```

Objets 2d

La **matrice**: un ensemble de valeurs à deux dimensions du même type



Création:

```
matrix(c(1, 2, 3, 4,  
        5, 6, 7, 8,  
        9, 10, 11, 12),  
       ncol = 4, nrow = 3,  
       byrow = TRUE)
```

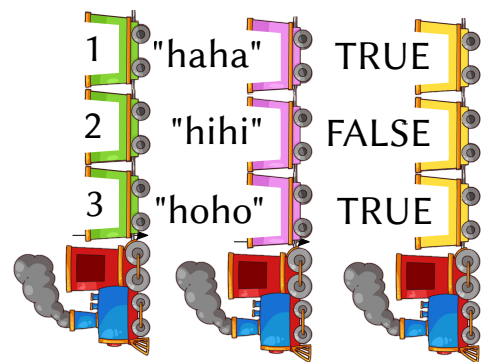
Nommage

```
lignes: rownames(mat) <- c("a", "b", "c", "d")  
colonnes: colnames(mat) <- c("a", "b", "c", "d")
```

Sélection/remplacement de sous-ensembles:

```
lignes: mat[<...>, ]  
colonnes: mat[<...>, ]  
cellules: mat[<...>, <...>]
```

Le **data.frame**: un ensemble de valeurs à deux dimensions de types différents



Création:

```
data.frame(c(1, 2, 3),  
          c("haha", "hihi", "hoho"),  
          c(TRUE, FALSE, TRUE))
```

Nommage

```
lignes: rownames(df) <- c("a", "b", "c", "d")  
colonnes*: colnames(df) <- c("a", "b", "c", "d")
```

Sélection/remplacement de sous-ensembles:

```
lignes: df[<...>, ]  
colonnes: df[<...>, ]  
cellules: df[<...>, <...>]
```

* (ou names(df))

Objets dans R

90% des objets dans R =

	Dimension	
	1D	2D
Même type	Vecteur <code>c(...)</code> <code>names(vec) <- c("a", ...)</code> <code>vec[<...>] <- 1.34</code>	Matrice <code>matrix(c(...), ...)</code> <code>colnames/rownames(mat) <- c("a", ...)</code> <code>mat[<...>,] <- ...</code> <code>mat[,<...>] <- ...</code> <code>mat[<...>,<...>] <- ...</code>
Types différents	Liste <code>list(...)</code> <code>names(lis) <- c("a", ...)</code> <code>lis[[<...>]] <- "element"</code>	Data.frame <code>data.frame(...)</code> <code>colnames/rownames(df) <- c("a", ...)</code> <code>df[<...>,] <- ...</code> <code>df[,<...>] <- ...</code> <code>df[<...>,<...>] <- ...</code>

Inspection des objets



Quel est la structure de l'objet ?

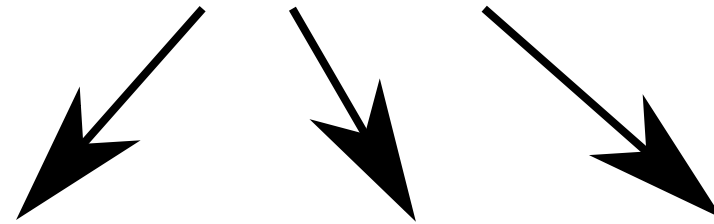
```
str(<...>)
```



Transformer des objets dans \mathbb{R} : Fonctions

La fonction dans R

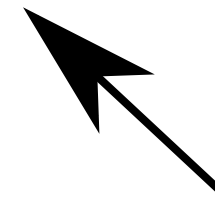
Des arguments



`function(arg1, arg2, arg3, ...)` {

`arg1 + arg2 * arg3`

}



Le corps de la fonction

La fonction dans R

Quels arguments prend la fonction ?

=> Inspecter une fonction: **str()**



La fonction dans R

Quels arguments prend la fonction ?

=> Inspecter une fonction: **str()**

Quels genre d'objets prend la fonction en argument ?

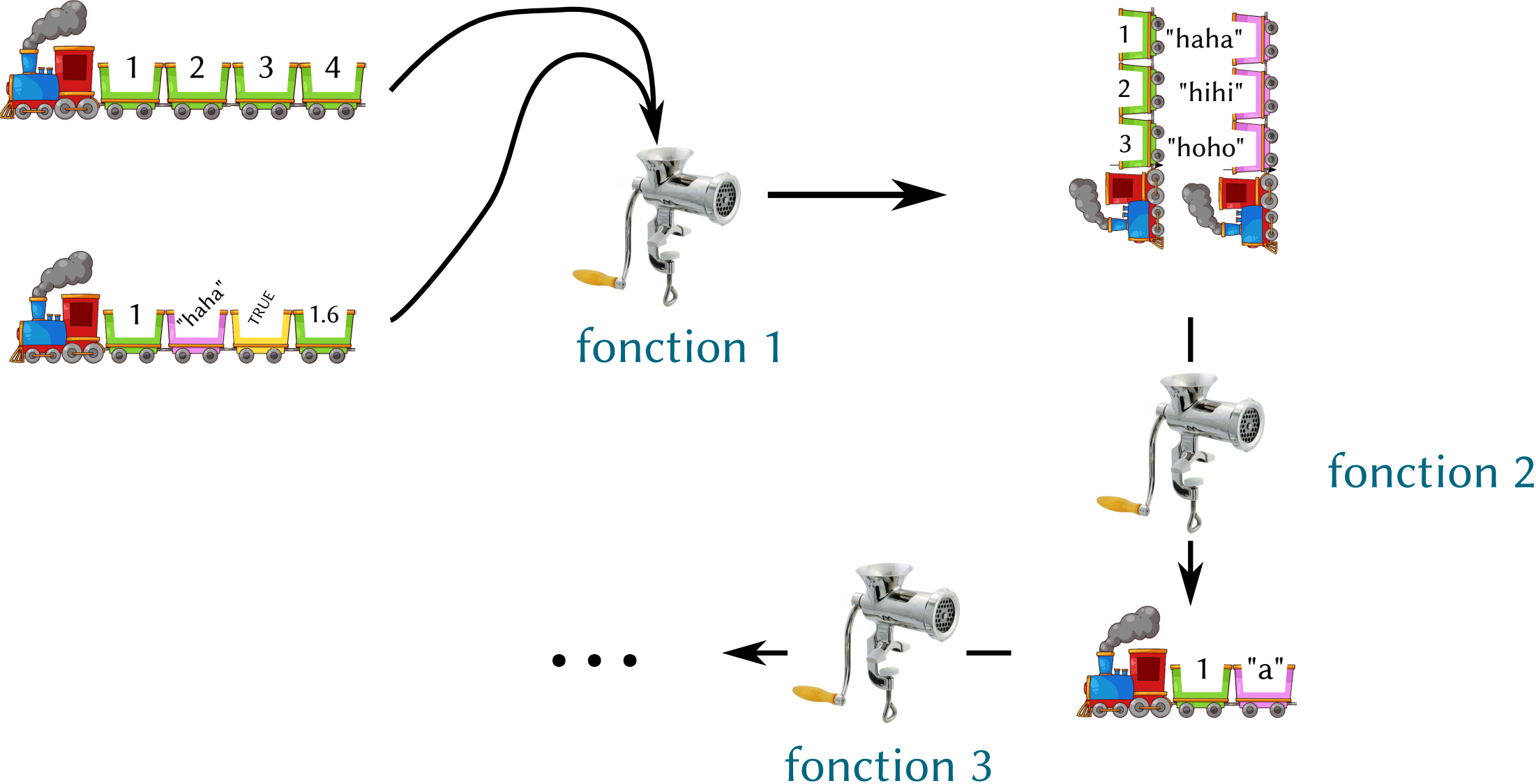
Quel objet renvoie la fonction ?

Pourquoi/quand appliquer la fonction ? quelles références ?

=> L'aide de la fonction: **?function** (par exemple: ?matrix)



Un script R, c'est ça:



En pratique...




Structures de contrôle

La clause if

```
if ( <valeur logique> ) {  
    <quelque chose >  
}
```

= un vecteur de type booléen
de longueur 1



La clause if

```
if ( <valeur logique> ) {  
    < quelque chose >  
}
```

Variantes

```
if ( <valeur logique> ) {  
    < quelque chose >  
} else {  
    < quelque chose d'autre >  
}
```

La clause if

```
if ( <valeur logique> ) {  
    < quelque chose >  
}
```

Variantes

```
if ( <valeur logique> ) {  
    < quelque chose >  
} else {  
    < quelque chose d'autre >  
}
```

```
if ( <valeur logique> ) {  
    < quelque chose >  
} else if ( <valeur logique> ) {  
    < quelque chose d'autre >  
} else {  
    < quelque chose d'autre >  
}
```

La boucle for

```
for ( i in vec ) {
```

```
    < quelque chose répété avec i = vec[1], vec[2], ..., vec[n] >
```

```
}
```

Le corps de la boucle



La boucle for

```
for ( i in vec ) {
```

```
  < quelque chose répété avec i = vec[1], vec[2], ..., vec[n] >
```

```
}
```

Le corps de la boucle



Exemple: additionner de 1 à 4

```
total <- 0  
vec <- c(1, 2, 3, 4)  
for ( i in vec ) {  
  total <- total + i  
}  
print(total)
```

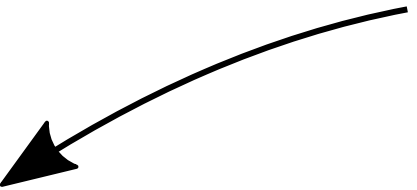
==

```
total <- 0  
vec <- c(1, 2, 3, 4)  
total <- total + vec[1]  
total <- total + vec[2]  
total <- total + vec[3]  
total <- total + vec[4]  
print(total)
```

La boucle while

```
while ( <valeur logique> ) {  
    < quelque chose répété >  
}
```

= un vecteur de type booléen
de longueur 1



Le **corps** de la boucle



La boucle while

```
while ( <valeur logique> ) {  
  < quelque chose répété >  
}
```

← Le corps de la boucle

Exemple: additionner jusqu'à 100

```
total <- 0  
while ( total < 100 ) {  
  total <- total + 1  
}  
print(total)
```

==

```
total <- 0  
total <- total + 1  
total <- total + 1  
...  
total <- total + 1 # total = 99  
total <- total + 1 # total = 100  
print(total)
```

En pratique...



Recap:

- **Types dans R** (numeric, logical, character, factor)
- **Objets dans R** (vecteurs, listes, matrices, data.frames)
- **Fonctions** (utilisation, définition, chercher de l'aide)
- **Structures de contrôle** (if, for, while)

Atelier: jeudi 8 octobre, 14h !

Tous les exercices et infos sur <https://rrr.mbb.cnrs.fr>