Dossier Individuel : Zoom sur le **REGEX**

2025-02-04

ETAT : PAS FINI

Avant-propos

Ce dossier a été réalisé dans le cadre du module OPEN (Outil P E N) dispensé à l'ISARA en 2025 et coordonné par M. PAYET. Ce dossier a pour but de présenter un travail de recherche sous-format R.Mark.Down pour nous familiariser à l'outil et justifier de compétences nécessaires abordées au cours de l'optionnel.

Pour ce faire, nous allons réalisé un support d'information et d'explication sur le concept des *Expressions Régulières* communément appelé *Regex*, qui est utilisé dans plusieurs langages informatiques qui permettre l'identification de structures régulières au sein d'un texte. Puisque le module se fonde essentiellement sur du R, nous aborderons ce concept au travers de ce langage et feront un état des lieux des possibles dans d'autres langages informatique. L'objectif a termes est de fournir un dossier qui permet de permettre l'apprentissage des Regex à toutes personnes en ayant besoin et en fournissant les différents codes adéquats à leur utilisation.

Cet envie d'approfondir mes connaissances sur les expressions régulières provient de mon stage de 4ème année où j'ai été amené à réaliser une base de données à partir de questionnaires. Le réarrangement des données se faisait sur R et je me suis souvent retrouver avec des problèmes de réponses dans les questionnaires. Une des manières de palliers ces problèmes de réponses incohérentes pouvait être le REGEX, puisqu'il permet de trier les informations qu'on souhaite selon un forme spécifique qu'on a pu lui attribuer.

Le REGEX

. . . de l'Homme à la machine

Avant de parler des expressions régulières, nous devons parler d'anatomie. En effet, l'origine des expressions régulières se rapporte à la théorie des automates. Au cours des années 1940, Warren McCULLOCH et Walter PITTS souhaitent décrire le système humain par l'intermédiaire d'automates simples. En 1943, ils arrivent à décrire le neurone formel ou neurone McCullock-Pitts, une représentation mathématique et informatique du neurone biologique. A l'instar du biologique, ce neurone dispose d'une ou plusieurs entrées (dendrites) et une sortie (le cône d'émergence). C'est le début de la théorie des automates. L'ensemble est dirigé selon un modèle mathématique, fonctionnant avec des états binaires (0 ou 1), on peut considérer ce modèle comme un automate fini où chaque neurone est une unité logique activée selon des règles précises. Peu de temps après, en 1945 John von Neuman va formaliser l'architecture des ordinateurs modernes. Il va appliquer la théorie des automates de McCulloch et Pitts pour les adapter aux ordinateurs réels. Dans les années 50, il est devenu nécessaire de décrire formellement les instructions à exécuter, ce qui a abouti aux premiers langages de programmation (Fortran, Algol) qui nécessitaient une grammaire bien définie. C'est alors que fut introduit la Théorie des Langages formels.

Langages formels = grammaires définies + automates pour les reconnaître.

REGEX et type de langage

Avant de parler de REGEX, il faut comprendre pourquoi on les a inventés et pour ce faire nous allons parler de langages. D'après la Hiérarchie de Chomsky, il existerait quatre types de grammaire, qui aboutissent à une famille de langage.

- Type 0 : **langage récursivement énumérable ou général** reconnaissable par une machine de Turing. Ils peuvent exprimer n'importe quel algorithme possible. En soit, l'algorithme peut comprendre toute forme de langage.
- Type 1 : **langage contextuel** : reconnaissable par les automates linéairement bornés. Ils peuvent inclure des règles plus complexes où certaines parties d'une phrase influences d'autres.

Exemple concret d'application : le correcteur grammatical. Un exemple de règle que ces automates peuvent comprendre : pour le mot « ACCESSIBLE » , on sait que le E est sans accent, car la langue française nous interdit de mettre un accent s'il y a présence d'une double consonne (ici le « SS ») juste après.

- Type 2 : **langage algébriques ou hors contexte** : langages reconnaissables par les automates à pile. Ils permettent des structures plus complexes, comme les parenthèses imbriquées dans une phrase.

Exemple concret : : une calculette ou une phrase bien formée en français. Ce type de langage est très bien présenter dans la plupart des langages de programmations. Parmi les règles qui permettent de présenter ce type de langage, cela va être tout ce qui est lié à la formation d'une « phrase ». Ici, on ne va pas s'intéresser au mot spécifiquement, mais à leur succession, leur assemblage dans la phrase. **Exemple typique** : « SUJET + VERBE + COMPLEMENT » ou bien toutes les conditions dans une fonction.

- Type 3 : **langages rationnels ou réguliers** reconnaissables par les automates finis.

Exemple concret : Une machine à tourniquer qui reconnaît un jeton « A » ou « B » ou un détecteur de mots-clés. Ce type de langage permet de vérifier les caractéristiques lexicales d'un mot. Un exemple concret est la vérification de la forme d'un courrier électronique lors d'une identification. En effet, un courrier électronique est composé comme suit : exempleAROBASE.COM

Globalement, tous les langages sont de types 0 et chaque type de langage est un cas particulier du type de langage supérieur. Par exemple, le langage de type 2 est un cas particulier du langage du type 1 au même titre que le type 3 est un cas particulier du type 2. Ainsi, plus on monte dans la hiérarchie, plus langage est puissant, mais aussi, plus il est complexe à analyser et à traiter. À l'inverse, plus on descend, plus le rajout de règles de restrictions des automates va amener à analyser des langages de plus en plus simples. En 1956, Stephen COLE KLEENE, définira les expressions régulières (REGEX) comme appartenant au type 3. C'est le principe du Théorème de Kleene qui affirme qu'un langage est rationnel si et seulement s'il est reconnu par un automate fini.

Il faudra attendre 1959 pour que Michael Rabin et Dana Scott proposent un traitement mathématique de ces concepts, ce qui leur firent accéder au prix Turing en 1976.

De la théorie à la machine

Jusqu'aux années 1960, les expressions régulières étaient une construction mathématique utilisée en logique et en linguistique formelle. En 1968, Ken Thompson, alors chercheur chez Bell Labs, va les faire entrer dans l'univers de l'informatique. En se basant sur les travaux de KLEEN, Thompson va implémenter l'idée des expressions régulières dans l'éditeur de texte ed, pour que ses utilisateurs puissent effectuer des correspondances de modèles avancées dans des fichiers texte. Par la suite, il les intégrera dans l'éditeur grep pour « Global Regular Expression Print » sous Unix. C'est le début de l'utilisation des expressions régulières en informatique. Les expressions régulières sont largement utilisées dans des utilitaires comme Iex et dans les langages de programmation nés sous Unix comme expr, awk, Perl, Tcl voire Python ou PHP. On les retrouve aussi sur R.

Utilisation du REGEX

A l'origine, les expressions régulières permettaient de décrire les langages formels. Aujourd'hui elles sont utilisées dans l'analyuse et la manipulation des langages informatiques. Les compilateurs et les interprètes sont ainsi basés sur celles-ci.

A l'origine créer pour décrire des langages formels, les expres Les expressions régulières sont utilisées dans de nombreux domaines. Les expressions régulières sont utilisés dans de nombreux domaines. On peut notamment les utiliser dans le cadre de la cybersécurité car elles permettent de vérifier si le type de données reçues correspodn au type de données attendues. Dans le domaine du marketing,

SYNTAXE DU REGEX

Imports

Les données utilisées sont issues des données disponibles sur R dansl le packages *dplyr*. Dans un premier temps nous allons donc installer les packages avec puis les importer.

```
#install.packages("dplyr")
#install.packages("ggplot2")

library(ggplot2)
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'

## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag

## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data("starwars")
```

Présentation

Tout d'abord, il est nécessaire de réaliser un premier état des lieux des données. Pour ce faire nous allons utiliser la fonction head()

```
head(starwars)
```

```
## # A tibble: 6 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke Sky~    172    77 blond      fair       blue            19   male  mascu~
```

```
## 2 C-3PO        167    75 <NA>       gold       yellow         112   none  mascu~
## 3 R2-D2         96    32 <NA>       white, bl~ red             33   none  mascu~
## 4 Darth Va~    202   136 none       white      yellow          41.9 male  mascu~
## 5 Leia Org~    150    49 brown      light      brown           19   fema~ femin~
## 6 Owen Lars    178   120 brown, gr~ light      blue            52   male  mascu~
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

On remarque nous avons affaire à une table remplie de nombreuses informations de différents personnages de la saga "Star Wars". Parmi les informations fournies, nous avons : le nom, la taille, le poids, la couleur de cheveux, la couleur de peau, la couleur des yeux, le sexe, le genre, l'origine (homeworld), l'espèce ainsi que les fils où ils sont présent, leurs véhicules et leurs vaisseaux.

Nous avons donc ici 14 caractéristiques de 87 personnages.

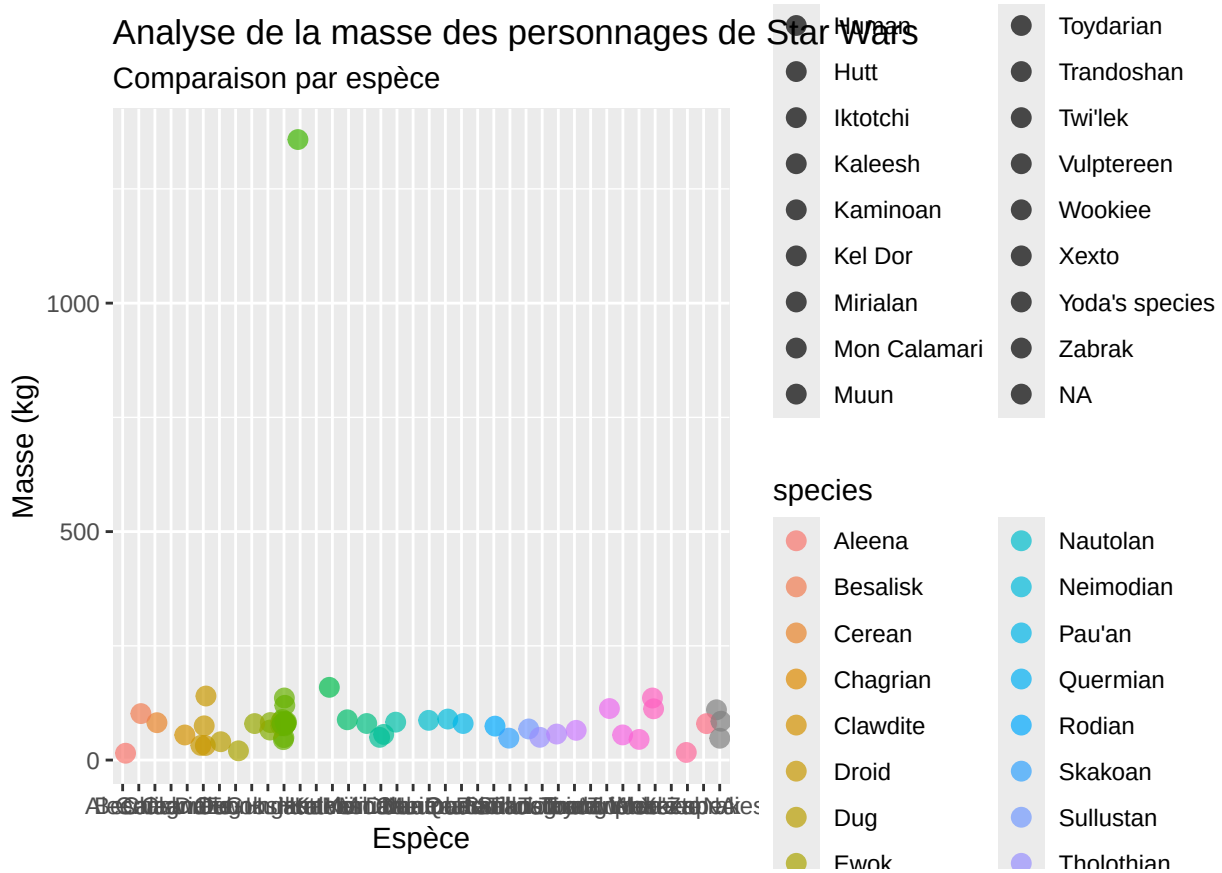
Méthodes et démarches

Le support de notre étude principale

On va une analyse assez simple car l'objectif c'est plus de perdre lecteur que de faire une analyse profonde. Pour ce faire, un simple plot qui met les espèces en abscisses en fonction de leur masse pourrait convenir. On va déjà voir ce que ça donne :

```
ggplot(starwars, aes(x = species, y = mass, fill = species)) +
geom_jitter(aes(color= species), width = 0.2, size = 3, alpha=0.7)+
ggtitle("Analyse de la masse des personnages de Star Wars")+
labs(subtitle= "Comparaison par espèce",
x = "Espèce", y = "Masse (kg)", fill = "Couleurs des Espèce")
```

```
## Warning: Removed 28 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



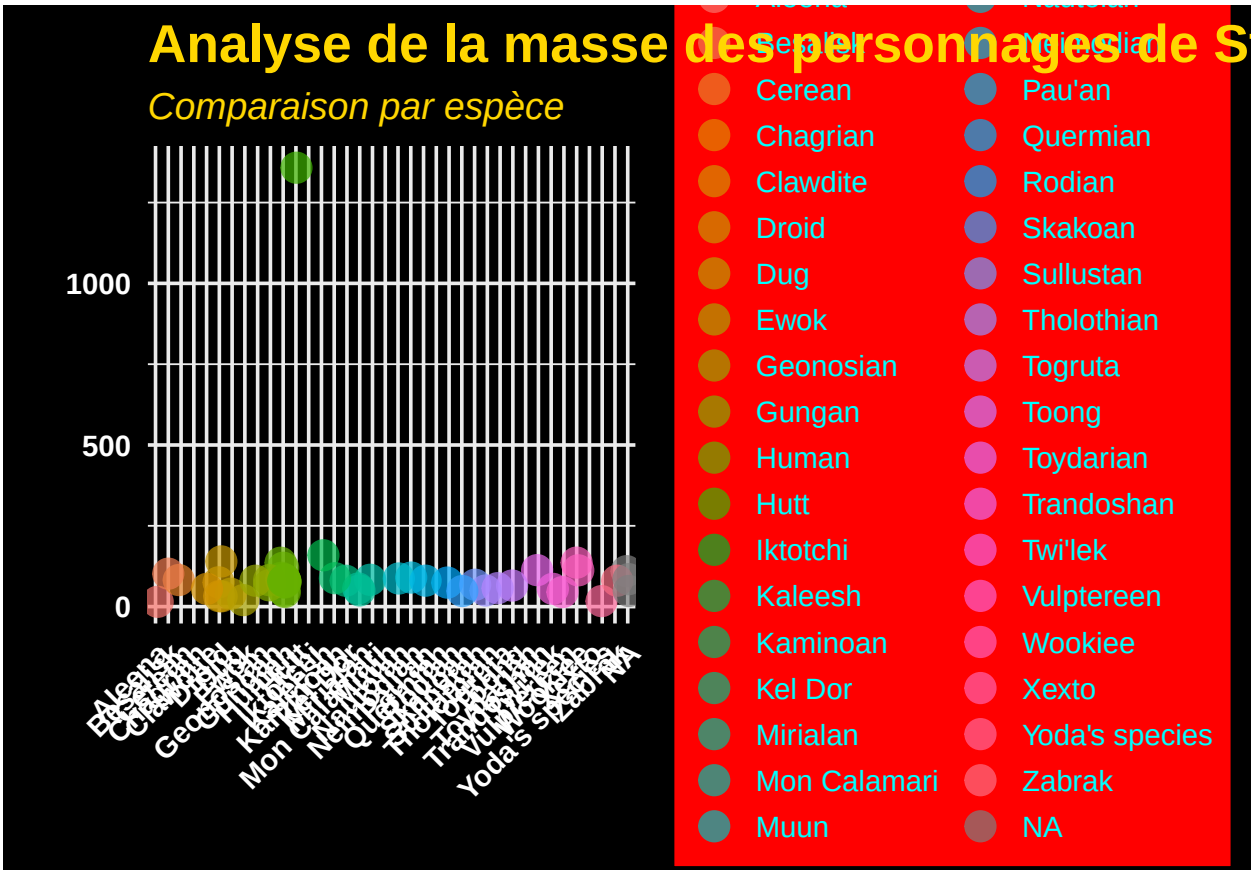
Dégrader le visuel

Maintenant qu'on a déjà une visualition confuse des données, on va renforcée la difficulté de lire à partir de différentes commandes. Les explications de chaque ligne sont présentes dans le code. On va grossir les points car on arrive trop à les distinguer. Dans un second temps on change la police à 14 ensuite on va personnaliser les axes et les affichages. Afin de rester la typologie et les couleurs de star wars, on va mettre le fond en noir et mettrre le titre et le sous-titre en jaune. Pour l'axe X, on met le texte à 45 degrés pour faire chevaucher les différents textes qui contiennent le nom des espèces. On va mettre un fond rouge et une écriture en cyan pour les textes. L'objectif est de rendre illisible, et difficile à regarder le graphique.

```
ggplot(starwars, aes(x = species, y = mass, fill = species)) +
  geom_jitter(aes(color= species), width = 0.2, size = 5, alpha=0.7)+
  ggtitle("Analyse de la masse des personnages de Star Wars")+
  labs(subtitle= "Comparaison par espèce",
  x = "Espèce", y = "Masse (kg)")+
  theme_minimal(base_size = 14)+
  theme(axis.text.x = element_text(angle=45, hjust =1, face="bold",color="white"),
  axis.text.y = element_text(face="bold", color= "white"),
  plot.background = element_rect(fill = "black"),
  panel.background = element_rect(fill="black"),
  legend.background = element_rect(fill="red"),
  legend.text = element_text(color = "cyan"),
  legend.title = element_text(face = "bold", color= "white"),
  plot.title = element_text(face = "bold", size = 20, color = "gold"),
  plot.subtitle = element_text(face = "italic", size = 14, color = "gold"))+
```

```
  theme_minimal(base_size = 14)+
  theme(axis.text.x = element_text(angle=45, hjust =1, face="bold",color="white"),
  axis.text.y = element_text(face="bold", color= "white"),
  plot.background = element_rect(fill = "black"),
  panel.background = element_rect(fill="black"),
  legend.background = element_rect(fill="red"),
  legend.text = element_text(color = "cyan"),
  legend.title = element_text(face = "bold", color= "white"),
  plot.title = element_text(face = "bold", size = 20, color = "gold"),
  plot.subtitle = element_text(face = "italic", size = 14, color = "gold"))
```

```
## Warning: Removed 28 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



Un petit test de biblio

En vitesse et pour montrer la compréhension voici deux petites références biblios selon le thème de des données [Auteur ?] Star Wars (2025). D'ailleurs, le point vert qui est tout en haut est lié à un personnage très charismatique "Jabba" dont sa page wiki de fan est disponible ici. [Auteur ?] *Jabba Desilijic Tiure | Star Wars Wiki | Fandom* ([sans date])

```
#{r logo, echo=FALSE, out.width = '50%', fig.align = "center", fig.cap="Fig. 1: Jabba sur
un échiquier"} #knitr::include_graphics("jabba.png")
```

Bibliographie

Liste des publications :

[Auteur ?] *Jabba Desilijic Tiure / Star Wars Wiki / Fandom*, [sans date]. [en ligne]. [Consulté le 4 février 2025]. Disponible à l'adresse : https://starwars.fandom.com/fr/wiki/Jabba_Desilijic_Tiure

[Auteur ?] Star Wars, 2025. [en ligne]. janvier 2025. [Consulté le 4 février 2025]. Disponible à l'adresse : https://fr.wikipedia.org/w/index.php?title=Star_Wars&oldid=222503327