Prime vs Random Sets

Sebastien Plaasch
Maxime Rubio
Lucas Villiere

24 mai 2020

Résumé

Resume de notre projet..

Table des matières

1 Introduction

Ici on fait l'introduction

2 Creation d'ensembles aléatoires suivant la distribution de $\pi(x)$

2.1 Approche analytique

Cette approche pour créer des ensembles aléatoires (désignés par Q) qui partagent la même distribution que celle des nombres premiers, est basée sur un théorème (théorème 1) issu de ... .

Théorème 1. *L'hypothèse de Riemann est équivalente à l'assertion*

$$\forall\ n \geqslant 11,\ \ |p_n - ali(n)| < \frac{1}{\pi}\sqrt{n}\log^{5/2}(n)$$

où p_n représente le n-ième nombre premier.

2.1.1 Méthode de création d'un ensemble aléatoire Q

Les onze premiers éléments d'un ensemble Q sont choisis arbitrairement. Pour $n > 11$, voici la méthode de sélection de l'élément $q_n \in Q$:

- on pose $a = \max\left\{q_{n-1}, \lceil ali(n) - \frac{1}{\pi}\sqrt{n}\log^{5/2}(n)\rceil\right\}$ (où $\lceil x \rceil$ désigne la partie entière supérieure de x) ;
- on pose $b = \lfloor ali(n) + \frac{1}{\pi}\sqrt{n}\log^{5/2}(n)\rfloor$ (où $\lfloor x \rfloor$ désigne la partie entière inférieure de x) ;
- q_n est choisi aléatoirement entre a et b.

En procédant de la sorte, le théorème 1 sera toujours vrai pour tout ensemble aléatoire Q.
Par cette méthode, nous avons créé 200 ensembles, jusqu'à 10^7, dont pour 100 d'entre-eux on a imposé la condition suivante : $\forall\, q_n \in Q$ tel que $n > 11$: q_n est impair.

2.1.2 Définition de la fonction σ

On peut désormais définir $\sigma : [0, \infty[\ \to \mathbb{R},\ x \mapsto \#\{n \in Q : n < x\}$. Nous parlerons systématiquement de la fonction σ alors que cette fonction n'est bien entendu pas unique, elle dépend à chaque fois de l'ensemble aléatoire Q sur lequel on travail. Cependant, nous avons pu remarquer que les différentes fonctions σ sont souvent très proches les unes des autres. À titre d'exemple, pour un grand nombre d'ensembles aléatoires Q, nous avons calculé $\sigma(1000)$. Pour 45% des ensembles, $\sigma(1000) = 148$ et parmi 42% d'entre eux, $\sigma(1000) = 147$.
Afin de visualiser $\sigma(x)$ en la comparant à $\pi(x)$ et $\frac{x}{\log(x)}$, voici leur graphe respectif :

La figure 1 devrait apparaitre ici.

2.1.3 Écart entre $\pi(x)$ et $\sigma(x)$

La fonction σ semble suivre la même allure que $\pi(x)$. Cependant, lors de nos expérimentations, nous avons dessinés des graphes (que vous trouverez en annexe) pour des valeurs de x inférieures à celles de la figure 1. Pour des petites valeurs de x, la courbe de σ était presque confondue avec celle de $\frac{x}{\log(x)}$. Lorsque les valeurs de x sont de plus en plus grandes, $\sigma(x)$ tend vers $\pi(x)$. Pour analyser l'écart entre $\pi(x)$ et $\sigma(x)$, nous avons tracé le graphe de la fonction $\frac{\pi(x)}{\sigma(x)}$:

La figure 2 devrait apparaitre ici. On peut en conclure que $\pi(x) \sim \sigma(x)$.

2.1.4 Le Théorème des Nombres Premiers

L'objectif de cette section est de prouver la fidélité de nos ensembles aléatoires Q à la répartition des nombres premiers. Pour ce faire, nous allons vérifier si le Théorème des Nombres Premiers, cité ci-après, est vrai quand on remplace $\pi(x)$ par $\sigma(x)$.

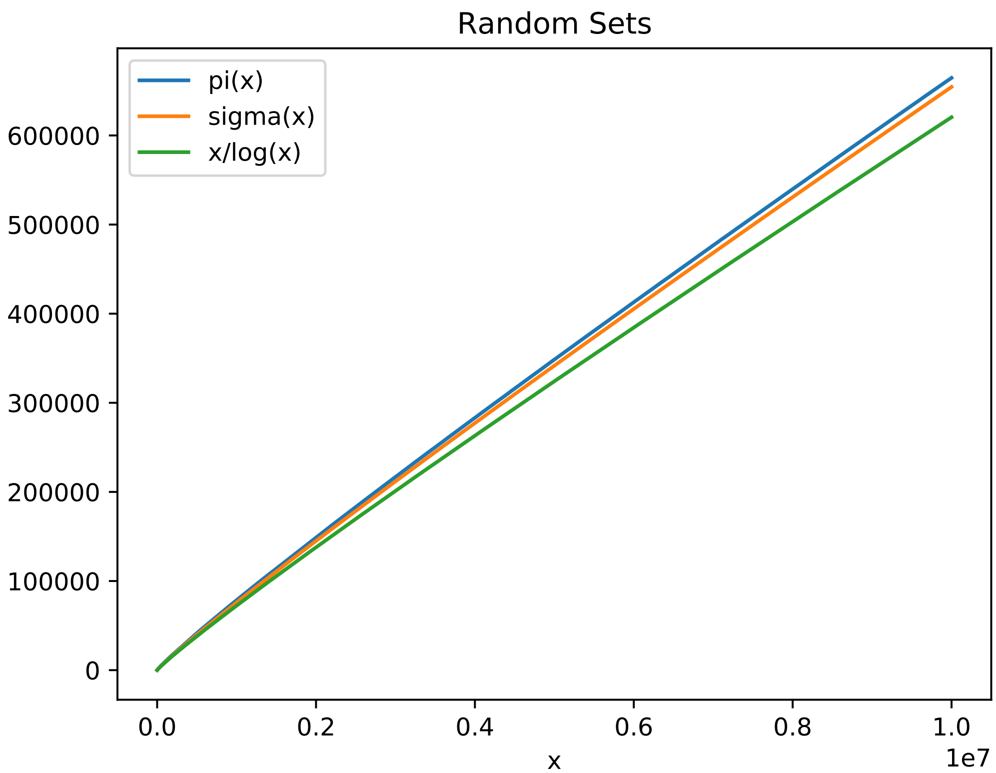


FIGURE 1 – Graphes de $\sigma(x)$, $\pi(x)$ et $\frac{x}{\log(x)}$

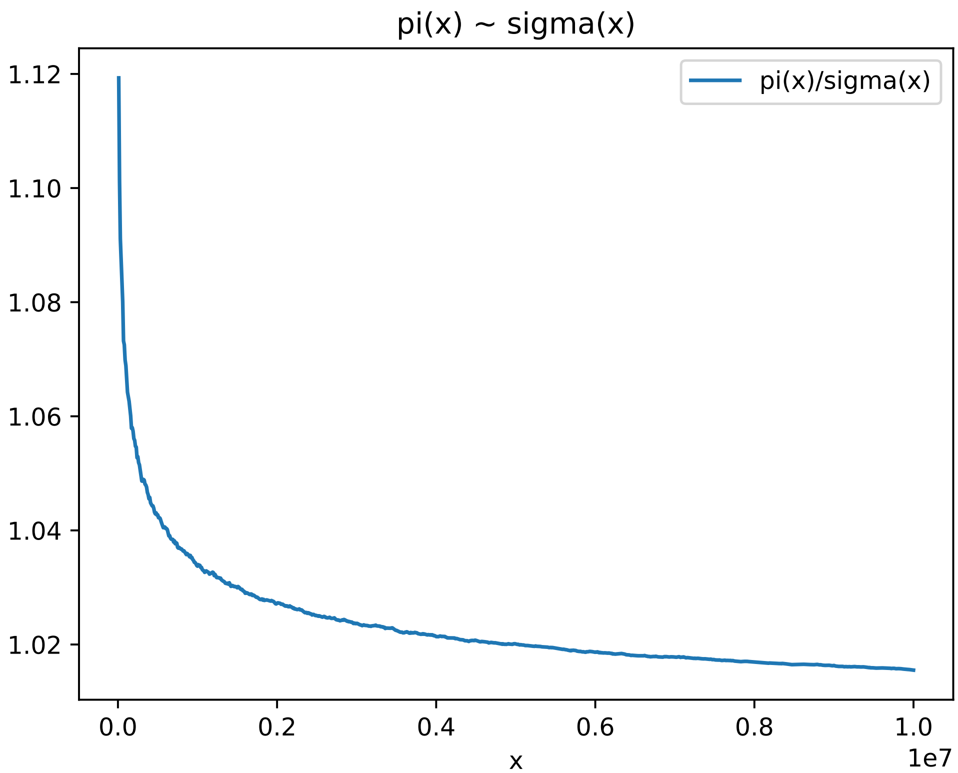


FIGURE 2 – Graphe de $\frac{\pi(x)}{\sigma(x)}$

Théorème 2 (Théorème des Nombres Premiers)**.** *Quand x tend vers l'infini :*

$$\pi(x) \sim \frac{x}{\log(x)}$$

Pour démontrer le Théorème des Nombres Premiers, il a été démontré que, quand x tend vers l'infini, $Li(x) \sim \frac{x}{\log(x)}$. Nous pouvons montrer, graphiquement, que lorsque x tend vers l'infini, $\sigma(x) \sim Li(x)$ (voir figure 3), ce qui implique que $\sigma(x) \sim \frac{x}{\log(x)}$.

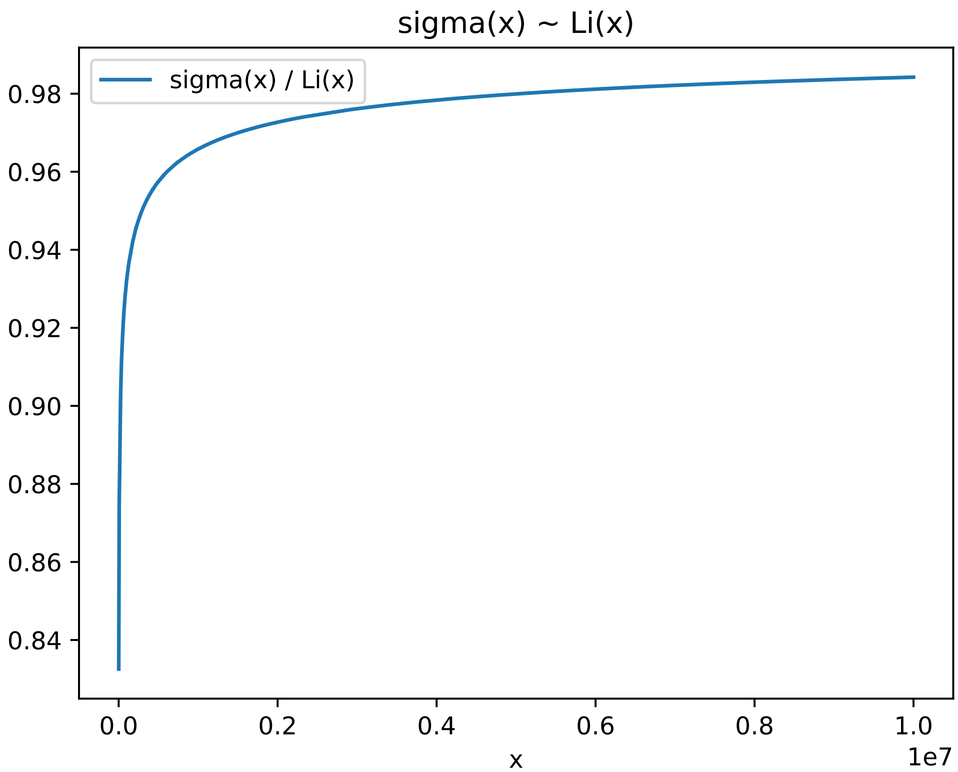


FIGURE 3 – Graphe de $\frac{\sigma(x)}{Li(x)}$

2.2 Approche probabiliste

Soient $E_n := \{k \in N^* \mid k < n\}$ l'ensemble des entiers inférieurs ou égaux à n, $P_n := \{k \in E_n \mid k$ est premier$\}$ l'ensemble des nombres premiers inférieurs ou égaux à n, et la fonction $\pi(n) := \#P_n$, le nombres de premiers inferieurs ou égaux à n.
On a vu que la fonction $\mathrm{Li}(x) = \int_2^x \frac{dt}{\log t}$ donne une bonne approximation de $\pi(n)$.
Cette fonction peut être approximée par la somme de Riemann de pas contant $= 1$:

$$S(\frac{1}{\log x}) = \sum_{k=0}^{n-1} \frac{1}{\log(2+k)} = \frac{1}{\log 2} + \sum_{k=3}^{n-1} \frac{1}{\log k} \tag{1}$$

La fonction $\frac{1}{\log x}$ est une fonction continue, décroissante et positive sur l'interval $[2, \infty[$. L'erreur entre $\mathrm{Li}(n)$ et la fonction en escalier ci-dessus est donc bornée.

$$\left| S(\frac{1}{\log x}) - Li(n) \right| \leq \left| \sum_{k=2}^{n-1} \frac{1}{\log k} - \frac{1}{\log(k+1)} \right| = \frac{1}{\log(2)} - \frac{1}{\log(n)} < \frac{1}{\log 2} < 2$$

2.2.1 Ensembles aléatoires

Nous allons alors générer des ensembles aléatoires $R_k \subset \mathbb{N}, k \in \{1, ..., 100\}$ de sorte que :

$$\forall\ i \in \mathbb{N}, P(i \in R_k) = \begin{cases} 0 & \text{si } i = 1 \\ 1 & \text{si } i = 2 \\ \frac{1}{\log n} & \text{si } i \geq 3 \end{cases}$$

Ces ensembles seront construits jusqu'à $i = 10^7$.
Il est à noter deux cas particuliers :

— Le nombre 1 est exclu. En effet, $\frac{1}{\log 1}$ n'est pas défini. Par définition, 1 n'est pas un nombre premier.

— Le nombre 2 est inclu par défaut. En effet, $P(2 \in R_{k_n}) = \frac{1}{\log 2} > 1$. De plus, le nombre 2 est par définition, un nombre premier.

La fonction $\sigma_{R_k}(n) := \#\{i \in R_k | i < n\}$ mesure donc la taille des ensembles jusqu'à un certain n. Cette fonction mesure la taille de l'ensemble aléatoire. Cette fonction est donc une valeur aléatoire strictement inférieure à n dont l'espérance est donnée par la formule suivante :

$$E[\sigma_{R_k}(n)] = 0 + 1 + \frac{1}{\log 3} + ... + \frac{1}{\log(n-1)} = 1 + \sum_{k=3}^{n-1} \frac{1}{\log k} \tag{2}$$

Notons σ_R (sans indicer R) l'ésperance de cette fonction. On observe alors que l'erreur entre $\sigma_R(n)$ et la somme de Riemann (1) est constante, égale à $\frac{1}{\log 2} - 1 < 1$.
Les ensembles aléatoires générés de cette manière suivront donc une distribution similaire a $\mathrm{Li}(x)$, et donc à $\pi(x)$ (voir figures 6 et 5).

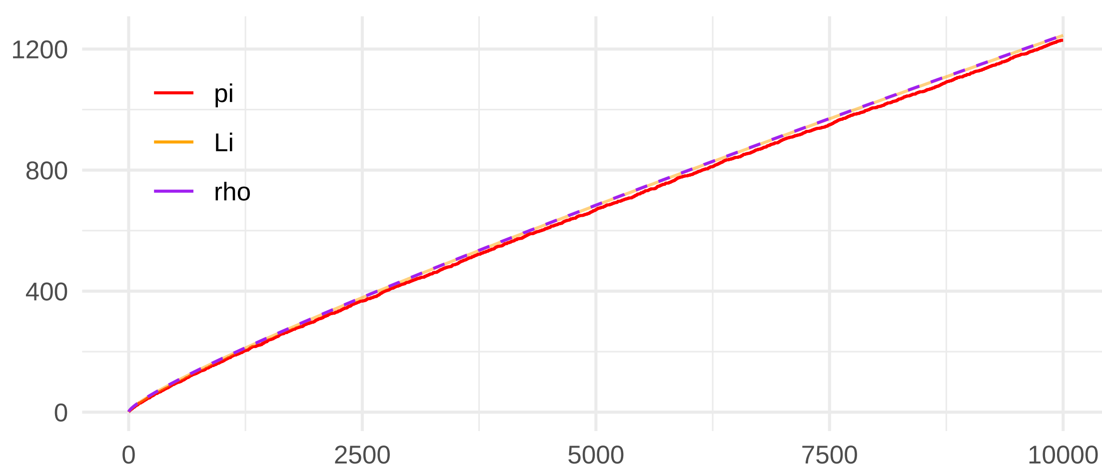


FIGURE 4 – Graphes des fonctions π, Li and σ. Li et σ sont superposées.

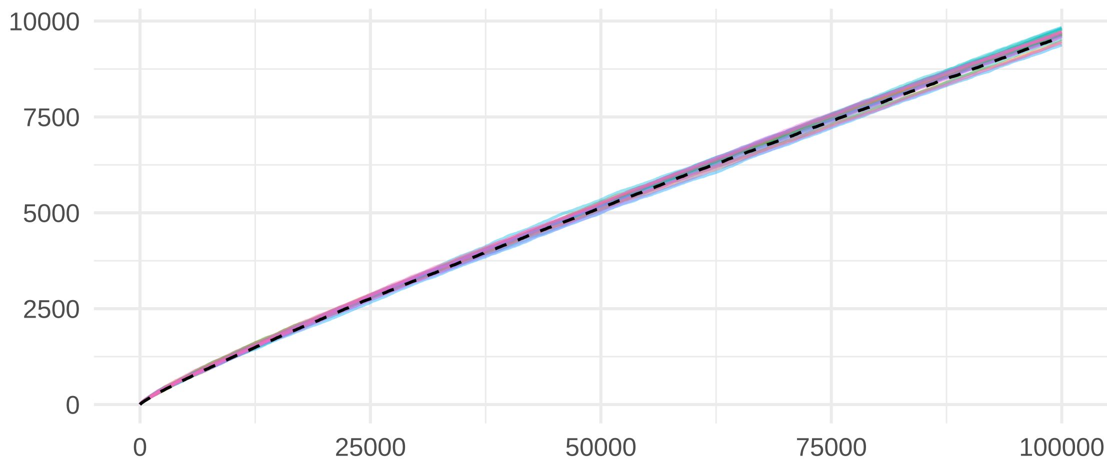


FIGURE 5 – graphes des fonctions σ_{R_k} pour $k \leq 25$ (i.e. les 25 premiers ensembles) et π (en pointillé)

2.2.2 Ensembles probabilistes impairs

Bien que ces ensembles aléatoires suivent la distribution de $\pi(x)$, il manque une propriété importante des nombres premiers : à l'exception de 2, tous les nombres premiers sont impairs. Nous allons alors modifier l'algorithme mentionné précedemment afin de générer des ensembles ayant cette propriété. Soient alors les ensembles $R'_k \subset \{2\mathbb{N} + 1 \cup \{2\}\}$ tels que $\forall\ i \in \{2\mathbb{N} + 1 \cup \{2\}\}$, le probabililité que i soit dans l'ensemble R'_k soit :

$$P(i \in R'_k) = \begin{cases} 0 & \text{si } i = 1 \\ 1 & \text{si } 2 \leq i < 9 \\ \frac{2}{\log n} & \text{si } i \geq 9 \end{cases}$$

Le fait que $P(i \in R'_k) = 1$ pour les entiers impairs inferieurs à 9 découle du au fait que $2/\log(n) > 1$ pour ces entiers.
Ces ensembles débutent tous avec les mêmes éléments : $(2, 3, 5, 7, ...)$, mais contienent ensuite des éléments choisis aléatoirement à partir du 5^e terme. Pour tout $n \in 2\mathbb{N}+1, n \geq 5$, la fonction cette les fonctions $\sigma_{R'_k}$ sont donc des variables aléatoires. Pour simplfier l'écriture dans les sommations ci-dessous, nous posons, pour tout $n \in 2\mathbb{N} + 1, m := \lfloor 2/n \rfloor$, de sorte que $2m - 1$ soit bien le plus grand entier impair inférieur à n. Notons $\sigma_{R'}$ la fonction mesurant l'esperance des fonctions $\sigma_{R'_k}$ en fonction de n.

$$\sigma_{R'}(n) = 4 + \frac{2}{\log 9} + \frac{2}{\log 11} + ... = 4 + \sum_{i=5}^{m} \frac{2}{\log(2i - 1)}$$

L'erreur entre l'esperance de $\sigma_{R'}(n)$ et Li(n) peut aussi être bornée :

$$\begin{aligned}
|\sigma_{R'}(n) - Li(n)| &= \left|4 + (\sum_{k=5}^{m} \frac{2}{\log 2k-1}) - (\int_2^n \frac{dt}{\log t})\right| \\
&= \left|4 + (\sum_{k=5}^{m} \frac{2}{\log 2k-1}) - (Li(9) + \int_9^n \frac{dt}{\log t})\right| \\
&\leq \left|(\sum_{k=5}^{m} \frac{2}{\log 2k-1}) - (\int_9^n \frac{dt}{\log t})\right| + |4 - Li(9)| \\
&= \left|\sum_{k=5}^{m} \frac{2}{\log 2k-1} - \int_{2k-1}^{2k+1} \frac{dt}{\log t}\right| + Li(9) - 4 \\
&< (\sum_{k=5}^{m} \frac{2}{\log 2k-1} - \frac{2}{\log(2(k+1)-1)}) + Li(9) - 4 \\
&= \frac{2}{\log 9} - \frac{2}{\log(2m+1)} + Li(9) - 4 \\
&< \frac{2}{\log 9} + Li(9) - 4 < 2
\end{aligned}$$

La première inégalité résulte de l'inégalité triangulaire et de $Li(9) > 4$. Le second inégalité vient du fait que $1/\log(x)$ est positive et décroissante sur l'interval $[5, \infty[$, et donc que $\frac{2}{\log n} > \int_n^{n+2} \frac{dt}{\log t} > \frac{2}{\log(n+2)}$

Nous obtenons alors des ensembles aléatoires, constitués de nombres impairs (sauf 2), suivant la distribution de Li(x). Les figures suivantes montrent la courbe de la fonction $\sigma_{R'}$ (figure 6), puis la distribution des 25 ensembles aléatoires impairs générés par cet algorithme (7).

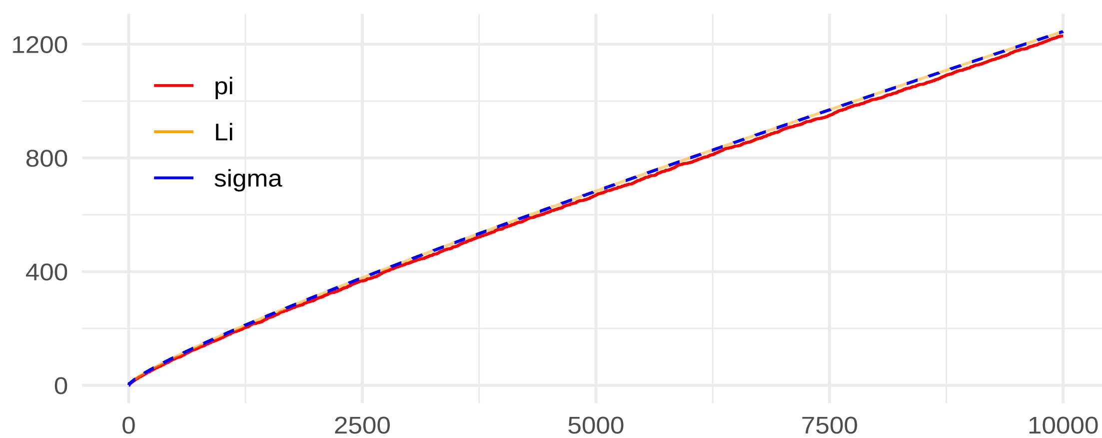


Figure 6 – Graphes des fonctions π et σ.

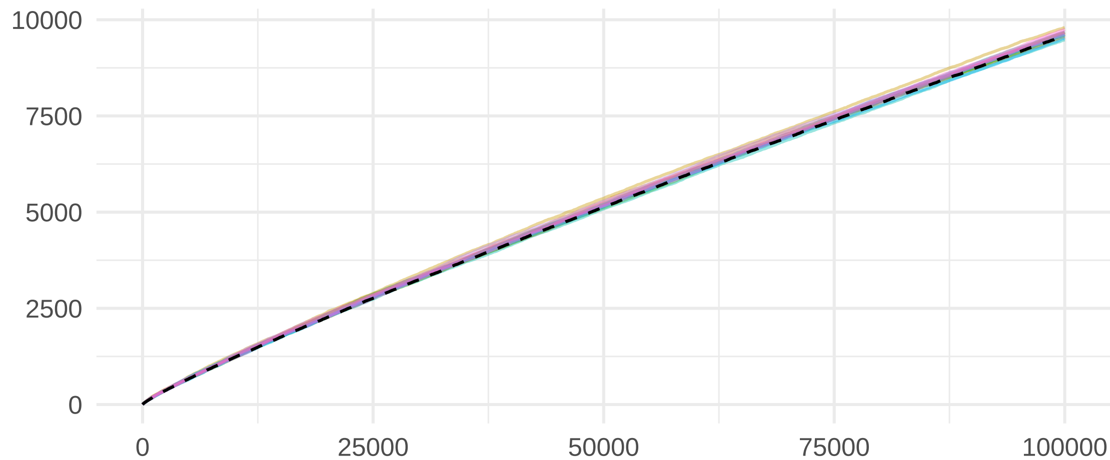


FIGURE 7 – graphes des fonctions σ_k pour $k \leq 25$ (25 premiers ensembles impairs) et π

2.3 Ressemblence et differences entre ensembles aléatoires et nombres premiers

Nous voulons maintenant comparer les groupes entre eux. Pour cela, nous allons mesurer, pour chaque ensemble, le rapport $\frac{\sigma_k(n)}{\pi(n)}$ où $\sigma_k(n)$.

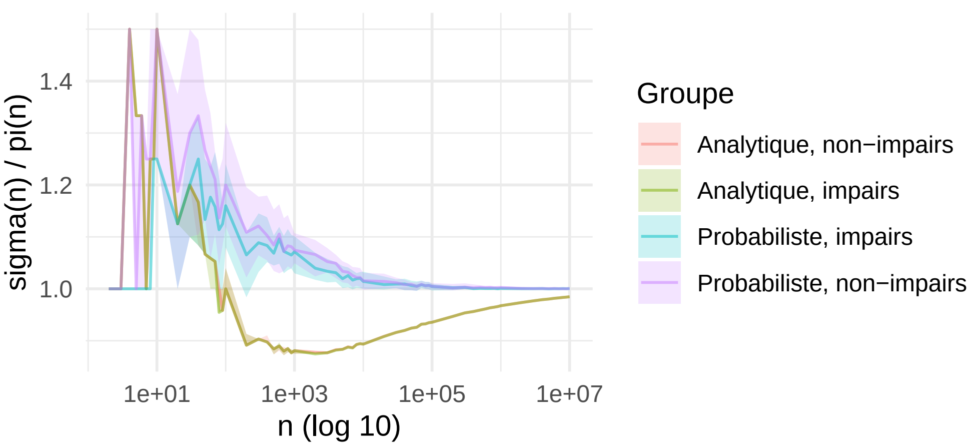


FIGURE 8 – Rapport entre $\sigma(x)$ et $\pi(x)$. La courbe représente la médiane et la surface autour de la courbe les premier et troisième quartiles.

Nous savons maintenant que les ensembles aléatoires suivent assez fidèlement $\pi(x)$. Nous souhaitons alors mesurer à quel point ces ensembles sont différents. Soit R_n l'ensemble dont nous souhaitons mesurer la ressemblance à $P_n := \{p \in \mathbb{N} \mid p \text{ est premier, p} < \text{n}\}$. Nous calculons alors le cardinal de $R_n \cap P_n$ puis divisons ce nombre par le cardinal de R_n afin d'obtenir, en pourcentage, la proportion d'élements premiers de R_n.
La figure 9 montre cette proportion pour chaque groupe d'ensemble.

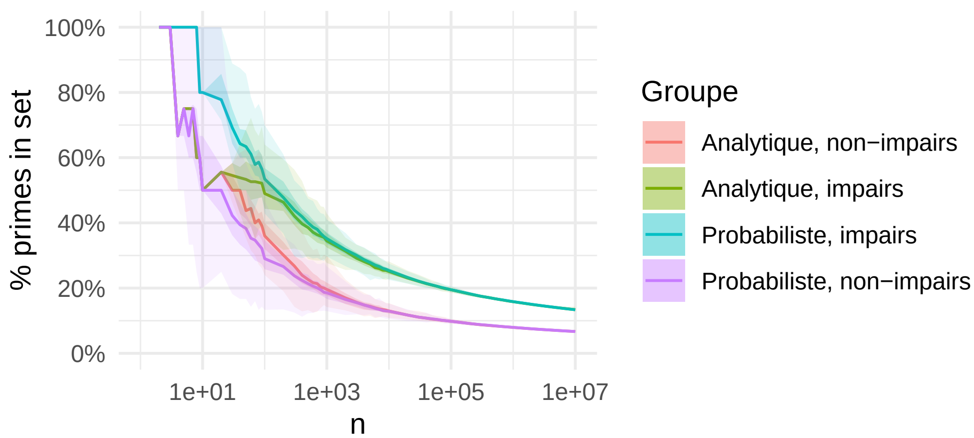


FIGURE 9 – Proportion de nombres premiers dans les ensembles aléatoires en fonction de n, en pourcentage. La courbe représente la médiane, la surface autour de la courbe les premiers et troisièmes quartiles.

À partir de 10^3, moins de la moitié des élements des ensembles sont des nombres premiers. Dès 10^5, les nombres premiers ne représentent plus que 10% des ensembles impairs, et 20% des ensembles non-impairs. De plus, il n'y a pas de différence significative entre deux ensembles d'un même groupe dès lors que n est assez grand.

3 Ensembles aléatoires et conjonctures

Dans cette section, nous allons vérifier si des conjectures sur les nombres premiers peuvent s'appliquer aux ensembles aléatoires créés à la section 2. Ainsi, on sera en mesure d'estimer si la véracité d'une conjecture est susceptible de tenir grâce à la répartition des nombres premiers plutôt qu'à leur propriété d'être premier.

3.1 Les nombres premiers jumeaux

La première conjecture que nous allons analyser, et sans doute la plus célèbre, est la conjecture des nombres premiers jumeaux.

Définition 1. Soient $a, b \in \mathbb{N}$, $a < b$, on dit que a et b sont jumeaux si $a + 2 = b$.

Conjecture 1. *Il y a une infinité de nombres premiers jumeaux.*

Ces dernières années, il y'a eu de grosses avancées dans la démonstration de la conjecture. Ainsi, pour tout $m \geqslant 1$, soit $H_m := \liminf_{n\to\infty}(p_{n+m} - p_n)$, où p_n dénote le n-ième nombre premier. La conjecture des nombres premiers jumeaux est donc équivalente à $H_1 = 2$. En 2013, le mathématicien chinois Zhang Yitang est le premier à trouver une borne supérieure finie pour H_1, il a démontré que $H_1 \leqslant 70\,000\,000$. Suite à la publication de Zhang Yitang, de nombreux mathématiciens se sont mis en quête de réduire la borne supérieure de H_1. En optimisant les résultats de Zhang Yitang et grâce à d'autres méthodes ils ont pu montrer que $H_1 \leqslant 246$.
Afin de tester la conjecture sur les ensembles aléatoires, nous avons tracé un graphe (figure 10), où pour chaque ensembles aléatoires et l'ensemble des nombres premiers (jusqu'à 10^7) on trace une fonction qui compte le nombre de jumeaux. Nous pouvons faire les observations suivantes :

- Pour chaque type d'ensemble (selon l'approche utilisée), on constate, logiquement, que le nombre de jumeaux est plus ou moins le double pour les ensembles impairs.
- Il y'a plus de jumeaux dans les ensembles R que dans les ensembles Q ce qui peut s'expliquer par le fait qu'ils ont plus d'éléments que les ensembles Q.

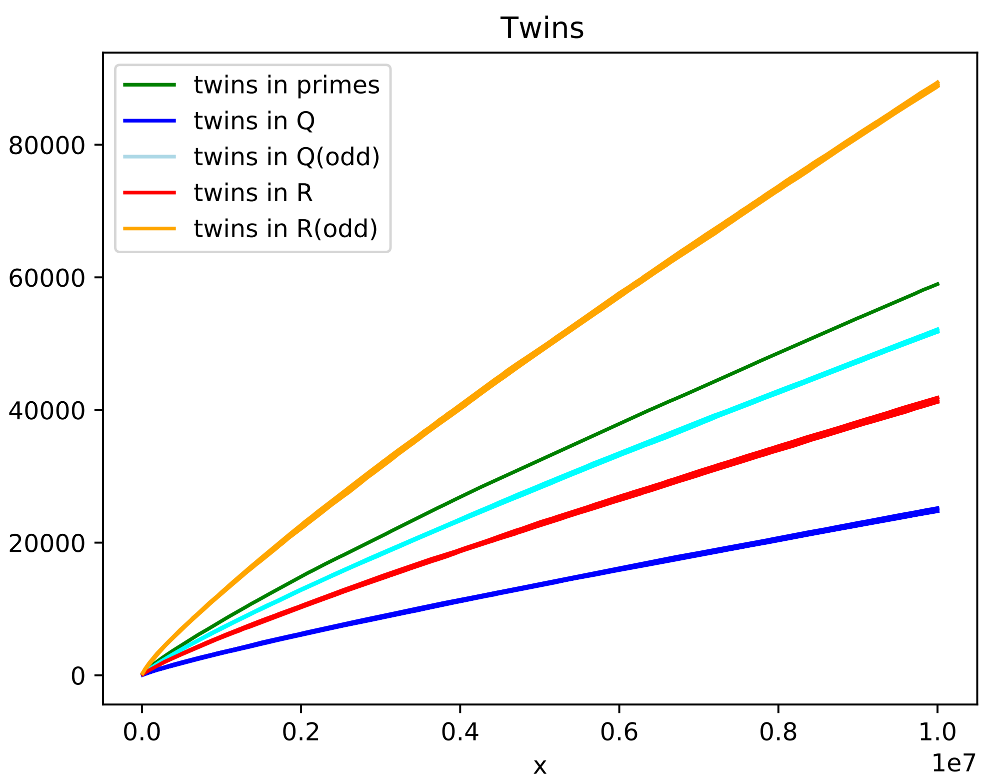


FIGURE 10 – Nombre de jumeaux dans chaque ensemble

— De manière générale, toutes les courbes sont croissantes, ce qui indiquerait, autant pour les nombres premiers que pour les ensembles aléatoires, que le nombre de jumeaux tend vers l'infini.

Par ailleurs, ce graphe éveille une idée intéressante. Si on désigne par f la fonction qui compte le nombre de jumeaux dans l'ensemble des nombres premiers et par g celle qui compte les jumeaux dans un ensemble aléatoire et qu'on parvient à montrer que f et g sont semblables (sous-entendu qu'elles le sont), alors $\lim_{x\to\infty} g(x) = \infty \Rightarrow \lim_{x\to\infty} f(x) = \infty$. Ce qui démontrerait la conjecture des nombres premiers jumeaux. Pour ce faire une idée d'une éventuelle équivalence entre f et g, voici le graphe (figure 11) de $\frac{f(x)}{g(x)}$, où g est appliquée à un ensemble impair Q choisi arbitrairement.

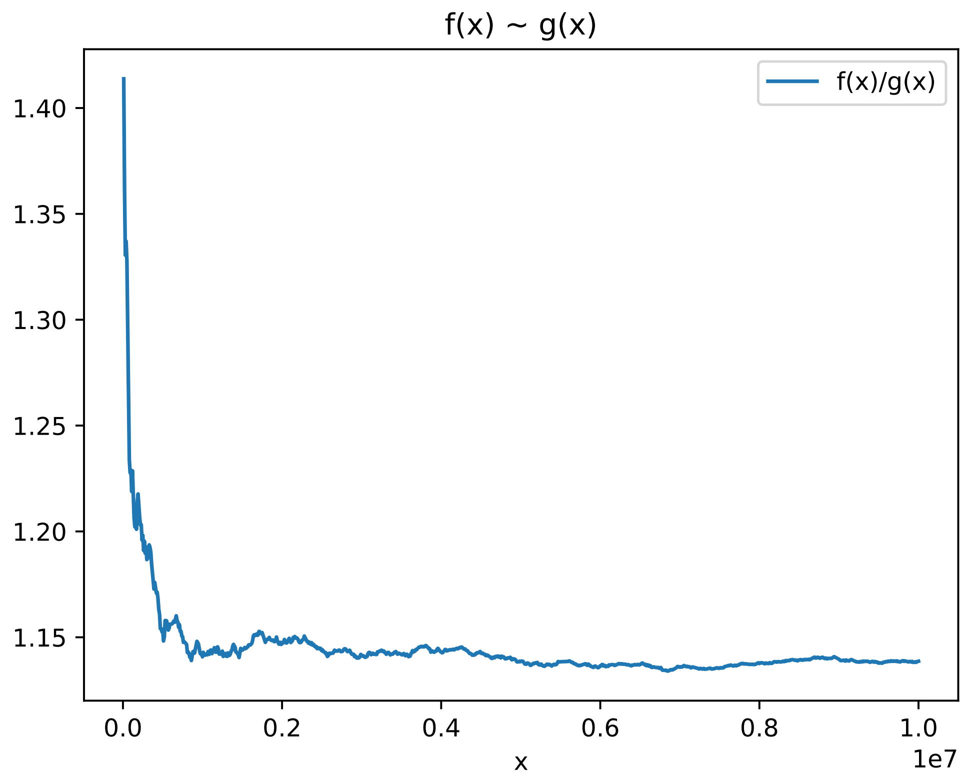


Figure 11 – $\frac{f(x)}{g(x)}$

Pour les ensembles aléatoires R (créés par l'approche probabiliste), on peut avoir une bonne approximation de la fonction g, qui compte le nombre de jumeaux. Soit R un tel ensemble, on définit $g : [0,\infty[\to \mathbb{R}, x \mapsto \sum_{i=2}^{\lfloor x \rfloor} \frac{1}{\log(i)\log(i+2)}$. "Elle donne une bonne approximation car c'est l'espérance mathématique." À titre comparatif, voici le graphe de $g(x)$ (figure 12), d'une fonction qui compte les jumeaux dans l'ensemble des nombres premiers et une qui les compte dans R.

On peut montrer que $\lim_{x\to\infty} g(x) = +\infty$. Donc $f \geqslant g \Rightarrow \lim_{x\to\infty} f(x) = +\infty$.

Pour conclure, la répartition des nombres premiers nous a permis de créer des ensembles aléatoires qui ont une infinité de nombres jumeaux.

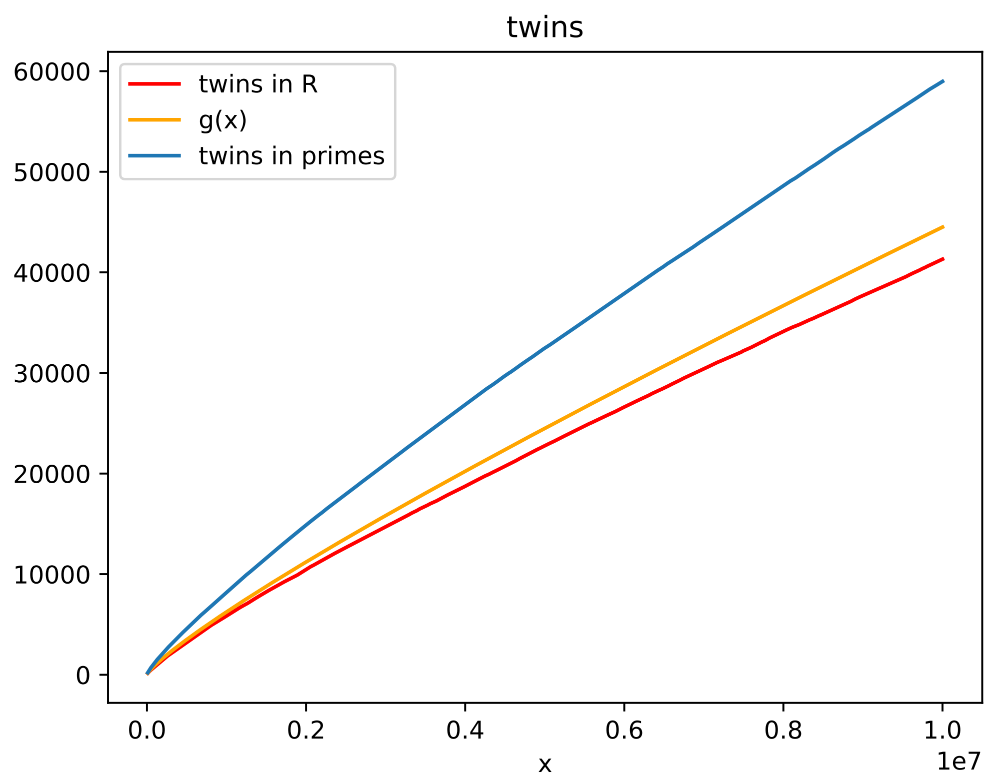


FIGURE 12 – Comparaison de $g(x)$

3.2 Seconde conjecture

3.2.1 Introduction

La conjoncture que nous allons tester ici est la suivante :

Conjecture 2. *Pour tout* $n = 6, 7, ...$*, il existe un nombre premier* p *tel que* $6n - p$ *et* $6n + p$ *sont tous les deux premiers.* [1]

Nous allons d'abord vérifier que cette conjecture tient pour les nombres premiers, puis vérifier si celle-ci tient aussi pour les ensembles aléatoires suivant leur distribution. La procédure pour analyser cette conjecture est donc la suivante : pour chaque $n \in \mathbb{N}, 6 \leq n \leq 10000$, nous vérifierons si l'assertion tient. Nous collecterons alors tout n tel que $\neg P(n)$ dans un tableau de données afin d'analyser, pour chaque ensemble ou chaque groupe d'ensemble, le nombre et la distribution des erreurs.

3.2.2 Analyse

Soit $P(n)$ l'assertion "Il existe $p \in R_{k_n}, p \leq n$ tel que $6n - p \in R_k$ et $6n + p \in R_k$".
En utilisant un algorithme [2], nous avons pu observer que l'assertion est vraie pour tout $n \leq 10^6$. Le même algorithme confirme que la conjecture n'est pas vérifiée, du moins pour tout $n \geq 6$, en ce qui concerne les ensembles aléatoires. Cependant, il semble que certains de ces ensembles possèdent des propriétés similaires si l'on choisit un n plus grand.
Nous nous intéressons au nombre d'erreurs (c'est à dire le nombre d'entiers n pour lesquelle l'assertion n'est pas vérifiée), ainsi que le plus grand entier pour lequel l'assertion est fausse. Cette dernière information est intéressante car si ce plus grand entier est petit, alors la conjoncture est vérifiée pour tout n plus grand.
Le graphique 13 ci-dessous représente en abscisses le nombre total d'erreur pour $n \in \{6, ..., 10^5\}$, et en ordonées le plus grand entier pour laquelle l'assertion est fausse. Chaque point représente un ensemble. Les ensembles ayant les meilleurs "performances" sont alors situés en bas à gauche : ceux-ci ont alors un faible nombre d'erreurs, et vérifie l'assertion pour tout n plus grand.

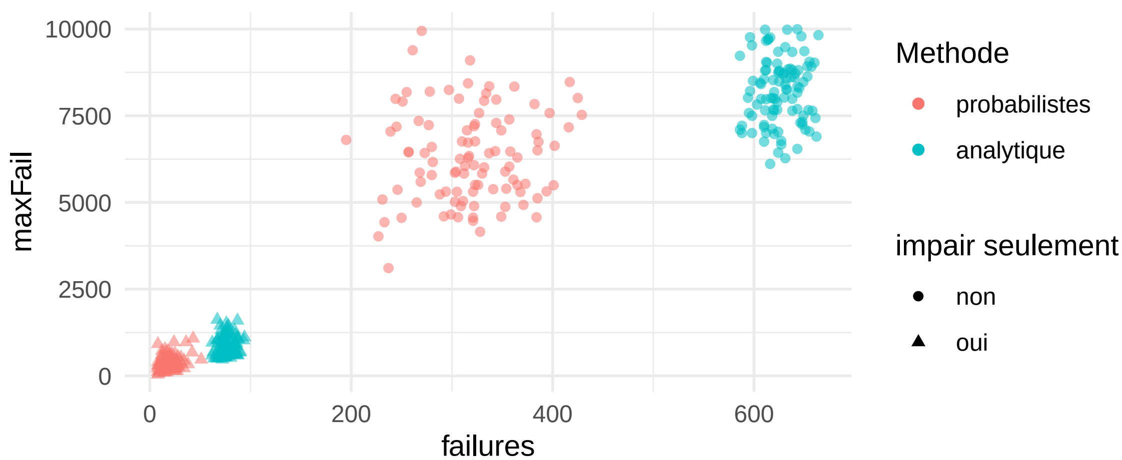
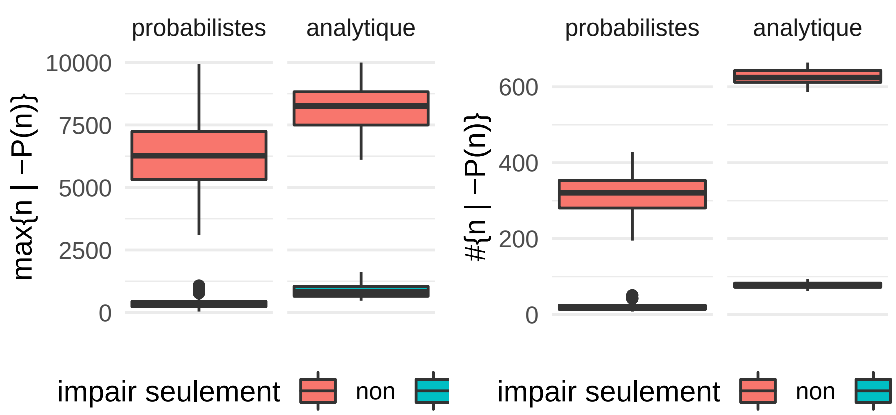


FIGURE 13 –

Nous debuterons par remarquer que l'existence d'un entier p répondant aux critères de la conjecture n'est pas rare. En effet, pour $n \in \{6, ..., 10000\}$, l'assertion est vérifiée par près de 99% des entiers n testés pour chaque ensemble (moins de 650 erreurs).

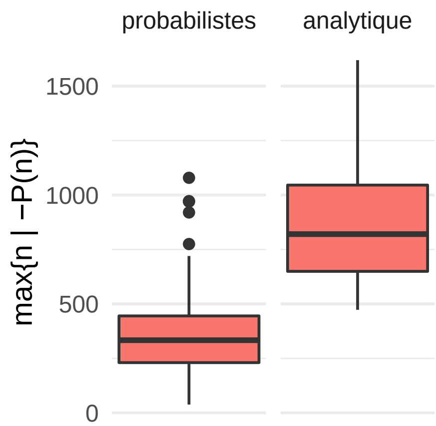
1. Conjecture 2.3 de ...
2. voir appendice : A.2.2 - "py_code/test_conj_2_3.py"

Les ensembles non restreints aux nombres impairs génerent tout de même un nombre assez élevé d'erreurs : plus de 200 erreurs pour la quasi-totalité de ces ensembles. De plus, ces erreurs persistent assez tardivement : pour la majorité de ces ensembles, il existe (au moins) un entier $n > 5000$ pour lequel l'assertion est fausse.
Les ensembles composés de nombres impairs ont cependant de bien meilleurs performances . Ceux-ci ont un faible nombre d'erreurs (moins de 50 pour les ensembles générés par l'algorithme probabilistique, moins de 100 pour les ensembles générés par les algorithmes analytiques, voir figure 15). De plus, le plus grand entier pour lequel l'assertion n'est pas vérifiée est relativement faible : cela signifie que pour tout entier $n > 1500$, l'assertion est vérifiée. Pour plus de trois quarts des ensembles aléatoires générés par l'algorithme probabiliste, on a même l'assertion vérifiée pour tout $n > 500$. Finalement, on remarque aussi que les ensembles crées par l'algorithme probabiliste ont des performances sensiblement meilleures. Cela est surement du au fait qu'ils possèdent sensiblement plus d'élements que les autre ensembles (voir figure 8), ou alors parce que leur distribution est plus proche de celles des nombres premiers.
Les diagrammes à boites 14 et 15 ci-dessus offrent un aperçu de la distribution des erreurs. La seconde figure se concentre sur les ensembles impairs afin de faciliter la lecture du graphique.

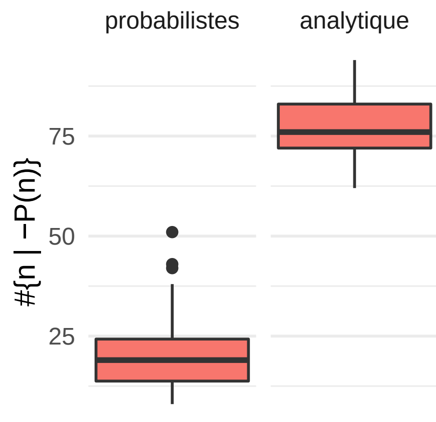


(a) Plus grand n ne vérifiant pas l'assertion (b) Nombre d'erreurs

FIGURE 14 – Diagrammes en boite



(a) Plus grand n ne vérifiant pas l'assertion



(b) Nombre d'erreurs

Figure 15 – Diagrammes en boite - ensembles impairs.

Il existe trois ensembles vérifiant l'assertion pour tout $n, 100 \leq n \leq 10000$:

— l'ensemble probabiliste impair 032 vérifiant l'assertion pour tout $n > 38$

— l'ensemble probabiliste impair 091 vérifiant l'assertion pour tout $n > 74$

— l'ensemble probabiliste impair 033 vérifiant l'assertion pour tout $n > 97$

Pour ces ensembles, l'assertion tient au moins jusqu'à $n = 10^6$.

De plus, au delà de 100, il existe 50 ensembles ayant moins de 5 erreurs. Il apparait alors que la probabilité d'une erreur diminue, et tend vers 0, lorsque n devient grand, comme le font remarquer les graphiques ci-dessous

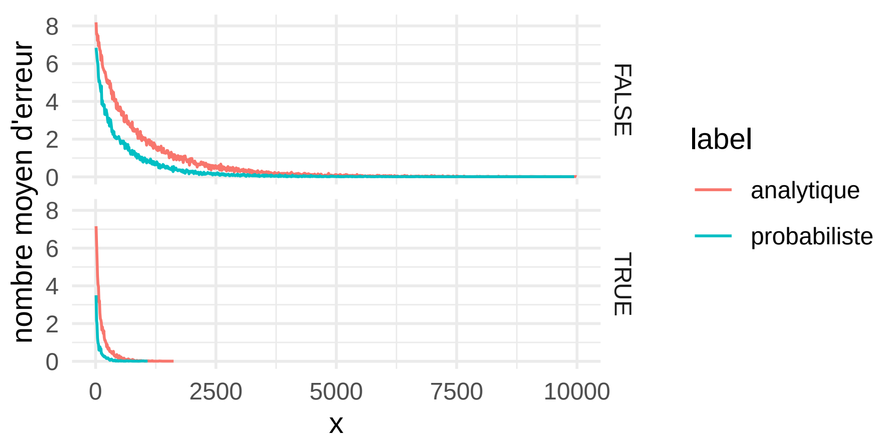


Figure 16 – Diagramme représentant le nombre moyen d'erreur sur l'intervale $]n, n+10], 6|n, 6 \leq n < 10000$

On observe ici que pour les ensembles aléatoires impairs, le nombre moyen d'erreur tends très vite vers 0 jusqu'à qu'aucune erreur n'apparaisse aux alentours de 1200. Cette statistique diminue moins vite pour les autres ensemble, mais tend aussi vers 0. En effet, il apparait que pour un n suffisement grand, la probabilité que l'assertion ne soit pas vérifiée est négligeable.

3.2.3 Conclusion de l'analyse

Certains ensembles aléatoires, notemment les ensembles impairs, semblent vérifier la conjecture pour n plus grand que 6. Pour certain d'entre eux, il semble qu'il existe un entier $m \in \mathbb{N}$ à partir duquel , pour tout $n > m$, il existe toujours un élement $p < n$ dans l'ensemble tel que $p, 6n - p$ et $6n + p$ sont tous dans l'ensemble. Nous nous demandons alors si cette conjecture tient pour les nombres premiers les nombres premiers, comme pour certains ensembles aléatoires démontrant de bonnes "performances", qu'à cause d'une distribution "heureuse" au départ, lorsque n est suffisement petit. Lorsque n est suffisement grand, il apparait que la probabilité d'une erreur est extrement faible, voir négligeable.

4 Conclusion

A Appendice

A.1 Code : Création d’ensembles aléatoires

A.1.1 Création d’ensembles probabiliste

Le code suivant genere 100 ensembles aléatoires non-impairs.

../r_code/generate_samples.R

```
# import base
source("r_code/pack_func.R")

# bound of the samples
bound <- 100000
x <- seq(2,bound)

# Sel generates a random variable between 0 and 1, returns true if variable <= 1/log(x), false otherwise
Sel <- function(x) {runif(1) <= 1/log(x)}

df <- data.frame(x)
# This function generates a single random set. Takes all integers between 2 and x and apply Sel function logic
rsamp <- function(){
  d <- data.frame(x,v= sapply(x, Sel))
  df <<- as.data.frame(cbind(df,R = d$v))
  # df = generate a data frame. It's a bit awkward to work with for the moment...
   d$x[which(d$v == TRUE)]
  # only selected integers are returned in the list.
}

# generates a list of samples
rand_sets <- replicate(100,rsamp())

write.csv(df,file.path("data",paste("prob_samples-",Sys.Date(),".csv",sep="")),row.names = F)
```

A.1.2 Création d'ensembles probabiliste impairs

Le code suivant est celui utilisé pour générer 100 ensembles aléatoires impairs.

../r_code/generate_samples2.R

```r
# import base
source("r_code/pack_func.R")
library(stringr)

# bound of the samples
bound <- 10^7
x <- seq(2,bound)
p <- "data/prob_sets/"
dir.create(p)

# Sel generates a random variable between 0 and 1, returns true if variable <= 1/log(x), false otherwise
Sel <- function(x) {runif(1) <= 1/log(x)}

df <- data.frame(x)
# This function generates a single random set. Takes all integers between 2 and x and apply Sel function logic
k <- 1

rsamp <- function(){
  d <- data.frame(x,v= sapply(x, Sel))
  df <<- as.data.frame(cbind(df,R = d$v))
  sample <- d$x[which(d$v == TRUE)]
  write.table(sample, paste(p,"prob_set_",str_pad(k,3,pad="0"),".txt",sep=""),row.names = F,col.names = F)
  # only selected integers are returned in the list.
  k <<- k + 1
  d
}

# generates a list of samples
rand_sets <- replicate(100,rsamp())
```

A.2 Seconde conjecture

A.2.1 Raport de conjecture

Le code suivant génere un tableau de donnée pour chaque ensemble, contenant le nombre d'entiers vérifiant la conjecture, le nombre d'echec et le plus grand entier ne vérifiant pas la conjecture.

../py_code/conjecture_2_3.py

```
import pandas as pd

primes = set()
for i in range(2, 100000):
  if all(i % p > 0 for p in primes):
      primes.add(i)


def conjecture(n, set):
    """
    Verify conjecture for a given integer n. \n
    Loops through all elements in set se, returns a tupple (n,p) where : \n
    \t - n is the inputed number
    \t - p is the minimum element of the set for which both (6n-p) and (6n+p) are elements of the set
    """
    # n*6-p > 0 => n*6 > p
    s = {p for p in set if p < n}
    for p in s:
        if (n*6 - p) in set and (n*6 + p) in set:
            return (n, p)

def tryForSet(set, bound, setname):
    """
    Try conjecture for a given set up to a certain bound. \n
    setname is a string which will be used for the output report. \n
    Output is a dataframe with setname, number of successes and failures, maximum element for which conjecture was not verified and success rate for the set
    """
    success, failure, maxFail = 0,0,0
    for x in range(6,bound):
        if conjecture(x, set):
            success += 1    # Increment success count if output is non null
        else:
            failure += 1    # Increment failure count if output is nul
            maxFail = x     # then sets maxFail to last unverified element.
    print("Set: ", setname, \
        ", success: ", success, \
        ", failure:", failure, \
        ", maxFail:", maxFail, \
        ", success rate: ", round(success/(failure+success)*100,2),"%", \
        sep = "")
```

```python
    return pd.DataFrame([[setname, success, failure, maxFail]],columns=["set","succes","failures","maxFail"])

import os
bound = 10000

files = [f for f in os.listdir('./data/odd_prob_sets') if f.endswith('.txt')]

def record_output(folder,bound,output):
    """
    Takes a folder containing sets as an input, generates a csv report up to a given bound
    """
    df = pd.DataFrame(columns=["set","succes","failures","maxFail"])
    df = df.append(tryForSet(primes,bound,"primes"))
    files = [f for f in os.listdir(folder) if f.endswith('.txt')]
    for f in files:
        s = {int(line.strip()) for line in open(folder + f)}
        df = df.append(tryForSet(s,bound,f))
        df.to_csv('./data/' + output + ".csv")

record_output('./data/odd_prob_sets/',10000,'conjecture_2_3_odd_10k')
record_output('./data/prob_sets/',10000,'conjecture_2_3_10k_V3')
record_output('./data/odd_analytic_sets/',10000,'conjecture_2_3_odd_analytique_10k')
record_output('./data/analytic_sets/',10000,'conjecture_2_3_analytique_10k')
```

A.2.2 Error mapping

Le code suivant est utilisé pour vérifier la conjecture pour un certain ensemble. Celui-ci ne génere pas de rapport mais s'arrête dès que l'assertion n'est pas vérifiée.

../py_code/test_conj_2_3.py

```python
import pandas as pd
def conjecture(n, set):
    """
    Verify conjecture for a given integer n. \n
    Loops through all elements in set se, returns the first element of the set p for which both 6*n-p and 6*n+p are elements of the set.
    """
    # n*6-p > 0 => n*6 > p
    s = {p for p in set if p < n}
    for p in s:
        if (n*6 - p) in set and (n*6 + p) in set:
            return (p)

minbound = int(input("Please enter lower bound: ")) # initial value for which assertion will by test
maxbound = int(input("Please enter higher bound: ")) # last value for which assertion will by tested

# Select a set here
s =  {int(line.strip()) for line in open('./data/odd_prob_sets/odd_prob_set_032.txt')}

# Loop through all integers between minbound and maxbound
# fails as soon as the assertion is not verified.
for n in range(minbound,maxbound):
    p = conjecture(n,s)
    if p:
            print(n,p)
    else:
        print("Failed at rank", n)
        break
print("Finished")
```