Projet de Modélisation et Calcul Scientifique : AB Testing

ANDRÉS WANG Luis
CLERE HAMELIN Maxime
DE MALET Clément

20 avril 2025

Université Paris-Saclay
L3 Mathématiques - MAN

Table des matières

1 Introduction

L'AB Testing est une stratégie de marketing essentielle dans tous les business en ligne utilisée pour déterminer si un innovation vaut le coup d'être implémentée définitivement, ou si telle ou telle version d'un produit est préférée à une autre par les clients.
Son principe est simple : on propose les deux versions du produit à deux échantillons d'individus différents, qui ne sont a priori par indépendants. L'un se voit proposer une première version, l'autre la seconde. Dans chacun des deux groupes, chaque individu a deux choix, soit il conserve le produit, soit il ne le retient pas. Par cette méthode, on souhaite obtenir une estimation de la préférence pour une version par rapport à l'autre par la différence des proportions de conservation du produit dans les deux groupes : si la proportion de personnes ayant conservé la version A du produit est "assez grande" dans le groupe concerné en comparaison de la proportion de personnes ayant conservé la version B dans le second groupe, on décidera que la version A est meilleure.
Deux questions importantes se posent alors. Premièrement, il faut obtenir des estimations assez précises des deux proportions évoquées plus haut, sans quoi toute comparaison serait biaisée et n'aurait aucun sens, c'est tout le travail des premières sections du projet.
Deuxièmement, une fois que nous avons des estimations jugées assez précises des deux proportions, il faut déterminer une règle de décision efficace de comparaison entre ces dernières afin de décider de la version la meilleure, c'est l'objet de la seconde partie du projet.
Ce projet est accompagnée d'une application à un cas concret, disponible et détaillé sous forme de notebook en annexe A.2.

2 Loi de la proportion observée d'utilisateurs

Dans cette première section, on commence par modéliser, pour un échantillon donné d'individus auquel on a proposé une version d'un produit, le comportement de ces individus. L'objectif est d'obtenir un estimateur de la proportion réelle d'individus qui souhaiteraient conserver la version en question.

Pour cela, on suppose que l'utilisateur adopte un comportement modélisé par une variable aléatoire de Bernoulli. On considère un échantillon de n utilisateurs, dont les choix sont modélisés par les variables aléatoires X_i pour i entre 1 et n, qui suivent une loi de Bernoulli de paramètre p. Pour tout i entre 1 et n, la variable X_i prend la valeur 1 si l'individu i décide de conserver le produit qui lui est proposé, 0 dans le cas contraire. Ensuite, on modélise la proportion d'individus qui souhaitent conserver le produit par la moyenne des X_i, donnée par :

$$\hat{p}_n = \frac{1}{n}\sum_{i=1}^{n} X_i. \tag{1}$$

Ceci est un estimateur de la proportion des utilisateurs qui aiment le produit en question.
On a $n\hat{p}_n = \sum_{i=1}^{n} X_i \sim \mathcal{B}(n,p)$, où p est la proportion réelle de personnes qui voudraient conserver la version du produit,
d'où :

$$\mathbb{P}(n\hat{p}_n = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Soit :

$$\mathbb{P}(\hat{p}_n = \frac{k}{n}) = \binom{n}{k} p^k (1-p)^{n-k}$$

Ainsi, comme $\hat{p}_n$ est une variable aléatoire d'espérance p et de variance $\frac{1}{n^2}\mathbb{V}[n\hat{p}_n] = \frac{p(1-p)}{n}$, on peut appliquer l'inégalité de Bienaymé-Tchébychev :

$$\forall \epsilon > 0, \mathbb{P}(|\hat{p}_n - p| > \epsilon) \leq \frac{p(1-p)}{n}\frac{1}{\epsilon^2}$$

et donc en passant à la limite :

$$\forall \epsilon > 0, \lim_{n\to+\infty} \mathbb{P}(|\hat{p}_n - p| > \epsilon) \leq \lim_{n\to+\infty} \frac{p(1-p)}{n}\frac{1}{\epsilon^2} = 0$$

Soit enfin :

$$\forall \epsilon > 0, \lim_{n\to+\infty} \mathbb{P}(|\hat{p}_n - p| > \epsilon) = 0$$

ie.

$$\hat{p}_n \xrightarrow{\mathbb{P}} p \quad \text{lorsque} \quad n \to +\infty$$

Ainsi, notre estimateur $\hat{p}_n$ est bien convergent en probabilité. On va maintenant chercher à obtenir d'autres propriétés intéressantes sur cet estimateur, c'est l'objet de la prochaine section.

Avec les simulations numériques, on constate les résultats de loi des grands nombres et du théorème central limite sur un échantillon de X_i.

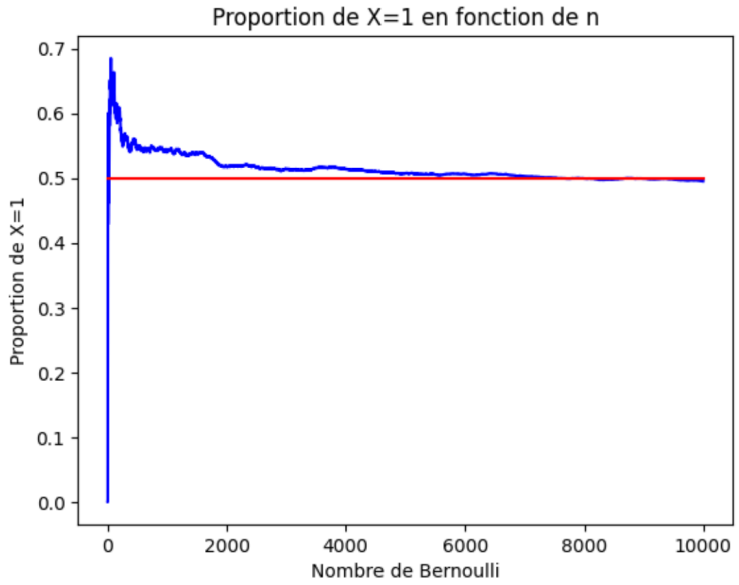


FIGURE 1 – Loi des grands nombres pour p = 0.5

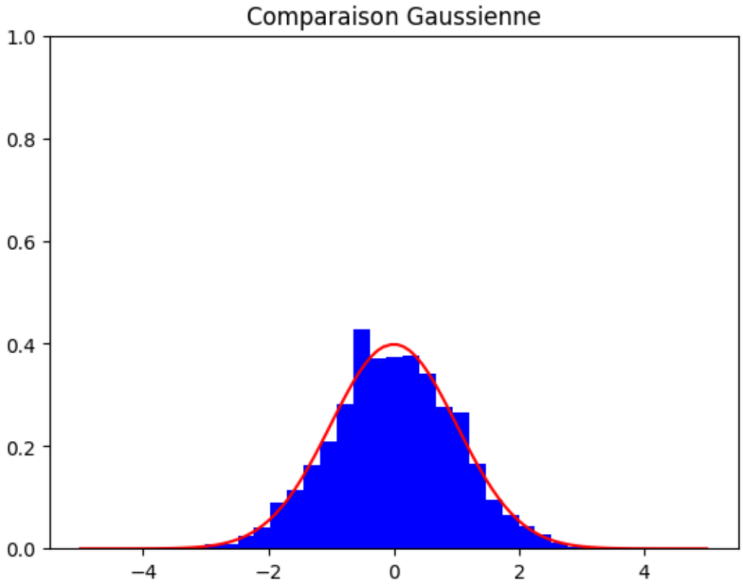


FIGURE 2 – Théorème central limite

2.1 Limite en loi de l'estimateur $\hat{p}_n$

A présent, on cherche à demontrer un théorème qui va nous permettre d'obtenir des inégalités par la suite, qui elles-mêmes nous permettront de trouver des intervalles de confiance pour notre proportion réelle. Le théorème est le suivant :

$$\forall t \geq 0, \text{lorsque} \quad n \to +\infty$$

on a les deux choses suivantes :

$$\mathbb{P}(\hat{p}_n \geq p + \frac{t}{\sqrt{n}}) \to \mathbb{P}(N \geq \frac{t}{\sqrt{p(1-p)}}) \tag{2}$$

$$\mathbb{P}(|\hat{p}_n - p| \geq \frac{t}{\sqrt{n}}) \to \mathbb{P}(|N| \geq \frac{t}{\sqrt{p(1-p)}}) \tag{3}$$

où N est une variable gaussienne centrée réduite.
Nous démontrons à présent ce théorème.

On considère toujours le même échantillon de n individus suivant une loi de Bernoulli de paramètre p. L'estimateur $\hat{p}_n$ est d'espérance p et de variance $\frac{p(1-p)}{n}$, donc par le Théorème central limite, on a :

$$\frac{\hat{p}_n - p}{\sqrt{\frac{p(1-p)}{n}}} \xrightarrow[n\to\infty]{\mathcal{L}} N \sim \mathcal{N}(0,1)$$

Par définition de la convergence en loi, la fonction de répartition de la loi normale étant continue sur $\mathbb{R}$, on a :

$$\forall t \geq 0, \mathbb{P}(\frac{\sqrt{n}(\hat{p}_n - p)}{\sqrt{p(1-p)}} \leq \frac{t}{\sqrt{p(1-p)}}) \xrightarrow[n\to+\infty]{\mathcal{TCL}} \mathbb{P}(N \leq \frac{t}{\sqrt{p(1-p)}})$$

Or, par le Théorème du porte-manteau, étant donné que la fonction de répartition de la loi normale est telle que :

$$\mathbb{P}(N = p + \frac{t}{\sqrt{n}}) = 0$$

On a :

$$\lim_{n\to\infty} \mathbb{P}(\frac{\hat{p}_n - p}{\sqrt{\frac{p(1-p)}{n}}} \geq \frac{t}{\sqrt{p(1-p)}}) = \mathbb{P}(N \geq \frac{t}{\sqrt{p(1-p)}})$$

Ce qui se réécrit directement :

$$\mathbb{P}(\hat{p}_n \geq p + \frac{t}{\sqrt{n}}) \xrightarrow[n\to+\infty]{} \mathbb{P}(N \geq \frac{t}{\sqrt{p(1-p)}})$$

Pour la seconde équation à montrer, on a la chose suivante :

$$\mathbb{P}(|\hat{p}_n - p| \geq \frac{t}{\sqrt{n}}) = \mathbb{P}(\hat{p}_n - p \geq \frac{t}{\sqrt{n}}) + \mathbb{P}(\hat{p}_n - p \leq -\frac{t}{\sqrt{n}})$$

Or, par le résultat que l'on a obtenu précédemment, on a :

$$\mathbb{P}(\hat{p}_n - p \geq \frac{t}{\sqrt{n}}) \xrightarrow[n\to+\infty]{} \mathbb{P}(N \geq \frac{t}{\sqrt{p(1-p)}})$$

et pour les mêmes raisons que précédemment, par le Théorème central limite :

$$\mathbb{P}(\hat{p}_n - p \leq -\frac{t}{\sqrt{n}}) \xrightarrow[n\to+\infty]{} \mathbb{P}(N \leq -\frac{t}{\sqrt{p(1-p)}})$$

Ce qui nous donne bien finalement :

$$\mathbb{P}(|\hat{p}_n - p| \geq \frac{t}{\sqrt{n}}) \xrightarrow[n\to+\infty]{} \mathbb{P}(|N| \geq \frac{t}{\sqrt{p(1-p)}})$$

Ces deux limites en poche, on cherche maintenant des bornes explicites pour ces dernières. On cherche à montrer que :

$$\lim_{n\to+\infty} \mathbb{P}(\hat{p} \geq p + \frac{t}{\sqrt{n}}) \leq \frac{1}{2} e^{-\frac{t^2}{2p(1-p)^2}}$$

et :

$$\lim_{n\to+\infty} \mathbb{P}(|\hat{p} - p| \geq \frac{t}{\sqrt{n}}) \leq e^{-\frac{t^2}{2p(1-p)^2}}$$

Pour cela, on commence par montrer que :

$$\mathbb{P}(N \geq t) \leq \frac{1}{2} e^{-\frac{t^2}{2}}$$

ainsi que :

$$\mathbb{P}(|N| \geq t) \leq e^{-\frac{t^2}{2}}$$

Pour ceci, on pose :

$$f(t) = \mathbb{P}(N \geq t) - \frac{1}{2} e^{\frac{-t^2}{2}}$$

Ou encore :

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_t^{\infty} e^{\frac{-x^2}{2}} dx - \frac{1}{2} e^{\frac{-t^2}{2}}$$

On remarque que f est $\mathcal{C}^\infty$ sur $\mathbb{R}^+$ par composition de fonctions $\mathcal{C}^\infty$ sur ce même ensemble, on a :

$$f'(t) = -\frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} + \frac{1}{2} t e^{\frac{-t^2}{2}}$$

On résoud $f'(t) \geq 0$:

$$f'(t) = e^{\frac{-t^2}{2}} (\frac{1}{2} t - \frac{1}{\sqrt{2\pi}}) \geq 0$$

D'où :

$$\frac{1}{2} t - \frac{1}{\sqrt{2\pi}} \geq 0$$

Donc :

$$t \geq \sqrt{\frac{2}{\pi}}$$

On en déduit que f est décroissante sur $[0, \sqrt{\frac{2}{\pi}}]$ et croissante sur $[\sqrt{\frac{2}{\pi}}, +\infty]$

De plus, $\lim_{t \to +\infty} f(t) = 0$ et $f(0) = 0$, En effet :

$$f(0) = \frac{1}{\sqrt{2\pi}} \int_0^{\infty} e^{\frac{-x^2}{2}} dx - \frac{1}{2} = \frac{1}{\sqrt{2\pi}} \frac{1}{2} \int_{-\infty}^{+\infty} e^{\frac{-x^2}{2}} dx - \frac{1}{2} = \frac{1}{\sqrt{2\pi}} \frac{1}{2} \sqrt{2\pi} - \frac{1}{2} = 0$$

On a utilisé ici le résultat : $\int_{-\infty}^{+\infty} e^{\frac{-x^2}{2}} dx = \sqrt{2\pi}$

Donc f négative est sur $\mathbb{R}^+$.

On en déduit l'inégalité souhaitée ci-dessus. On a ainsi :

$$\mathbb{P}(|N| \geq t) = \mathbb{P}((N \leq -t) \sqcup (N \geq t)) = 2\mathbb{P}(N \geq t) \leq e^{\frac{-t^2}{2}}, \text{ par symétrie et inégalité précédente.}$$

On obtient également grâce à ces résultats :

$$\lim_{n \to +\infty} \mathbb{P}(\hat{p} \geq p + \frac{t}{\sqrt{n}}) = \mathbb{P}(N \geq \frac{t}{\sqrt{p(1-p)}}) \leq \frac{1}{2} e^{-\frac{t^2}{2p(1-p)^2}}, \text{ par (2).}$$

et :

$$\lim_{n \to +\infty} \mathbb{P}(|\hat{p} - p| \geq \frac{t}{\sqrt{n}}) = \mathbb{P}(|N| \geq \frac{t}{\sqrt{p(1-p)}}) \leq e^{-\frac{t^2}{2p(1-p)^2}}, \text{ par (3).}$$

On a donc bien les résultats souhaités.

2.2 Inégalités classiques

Toujours dans le but de trouver des intervalles de confiance pour nos proportions, on établit à nouveau des inégalités pour nos probabilités, en partant de trois inégalités classiques : Bienaymé-Tchebytchev, Hoeffding, Bernstein. On rappelle ici les trois inégalités utilisées par la suite.

Théorème (Bienaymé-Tchebychev)**.** *Soit X une variable aléatoire d'espérance $\mu = \mathbb{E}[X]$ et de variance $\sigma^2 = \mathrm{Var}(X)$. Alors, pour tout $\varepsilon > 0$,*

$$\mathbb{P}\left(|X - \mu| \geq \varepsilon\right) \leq \frac{\sigma^2}{\varepsilon^2}.$$

Théorème (Hoeffding). *Soient $Z_1, \dots, Z_n$ des variables aléatoires indépendantes telles que, pour tout i,*

$$Z_i \in [a_i, b_i],$$

alors,

$$\mathbb{P}\left(\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Théorème (Bernstein). *On suppose que* $\mathrm{Var}(Z_i) \leq \sigma_i^2$. *Alors :*

$$\mathbb{P}\left(\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \geq t\right) \leq \exp\left(-\frac{t^2}{2\sum_{i=1}^{n}\sigma_i^2 + \frac{2t}{3}B}\right)$$

où

$$B = \max_i \max(b_i - \mathbb{E}[Z_i], \mathbb{E}[Z_i] - a_i) \leq \max_i(b_i - a_i).$$

Nous commençons avec Bienaymé-Tchebytchev : $\hat{p}_n$ étant une variable aléatoire d'espérance p et de variance $\frac{p(1-p)}{n}$, on peut appliquer l'inégalité de Bienaymé-Tchebytchev et ainsi obtenir :

$$\mathbb{P}(|\hat{p} - p| \geq \frac{t}{\sqrt{n}}) \overset{BT}{\leq} \frac{1}{n}p(1-p)\frac{n}{t^2} = \frac{p(1-p)}{t^2}$$

En pratique, ceci n'est pas utilisable car on ne peut pas calculer explicitement le membre de droite dans l'inégalité, il nous faut trouver une borne qui ne dépend pas de p, qui est inconnu. C'est pouquoi on va utiliser Hoeffding et Bernstein.
Commmençons avec Hoeffding, en utilisant le théorème énoncé précédemment, et en considérant toujours l'échantillon des X_i pour i allant de 1 à n, représentant les choix des n individus comme décrit dans l'introduction. On a $\forall i \in \{1, \dots, n\},\ X_i \in [0, 1]$ et par le théorème de Hoeffding :

$$\begin{aligned}\mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X_i] \geq \frac{t}{\sqrt{n}}) &= \mathbb{P}(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) \geq \frac{t}{\sqrt{n}}) \\ &= \mathbb{P}(\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) \geq t\sqrt{n}) \\ &\overset{\text{Hoeffding}}{\leq} \exp(\frac{-2nt^2}{\sum_{i=1}^{n}(1-0)^2}) = e^{-2t^2}\end{aligned} \tag{4}$$

Or, on a d'autre part :

$$\mathbb{P}(|\hat{p}_n - p| \geq \frac{t}{\sqrt{n}}) = \mathbb{P}(\hat{p}_n - p \leq -\frac{t}{\sqrt{n}}) + \mathbb{P}(\hat{p}_n - p \geq \frac{t}{\sqrt{n}}) \tag{5}$$

car $]-\infty, -\frac{t}{\sqrt{n}}] \cap [\frac{t}{\sqrt{n}}, +\infty[= \varnothing$
Or, on a d'une part :

$$\mathbb{P}(\hat{p}_n - p \geq \frac{t}{\sqrt{n}}) \leq e^{-2t^2}$$

en utilisant (4).
Et d'autre part :

$$\mathbb{P}(\hat{p}_n - p \leq -\frac{t}{\sqrt{n}}) = \mathbb{P}(p - \hat{p}_n \geq \frac{t}{\sqrt{n}}) = \mathbb{P}(\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[X_i] - X_i \geq \frac{t}{\sqrt{n}}) = \mathbb{P}(\sum_{i=1}^{n}(-X_i) - \mathbb{E}[-X_i] \geq t\sqrt{n})$$

On utilise alors à nouveau l'inégalité de Hoeffding avec les $-X_i$ et non plus les X_i. On a $\mathbb{E}[-X_i] = -p$ et $-X_i \in [-1, 0]$. On obtient à nouveau :

$$\mathbb{P}(\hat{p}_n - p \leq -\frac{t}{\sqrt{n}}) \leq e^{-2t^2}$$

Finalement, à partir de la somme de l'équation (5) :

$$\mathbb{P}(|\hat{p} - p| \geq \frac{t}{\sqrt{n}}) \leq 2e^{-2t^2}$$

Enfin, on obtient une dernière inégalité à l'aide du théorème de Bernstein :

$$\begin{aligned}\mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X_i] \geq \frac{t}{\sqrt{n}}) &= \mathbb{P}(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) \geq \frac{t}{\sqrt{n}}) \\ &= \mathbb{P}(\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) \geq t\sqrt{n}) \\ &\overset{\text{Bernstein}}{\leq} \exp(\frac{-nt^2}{2n + \frac{2}{3}t\sqrt{n}})\end{aligned}$$

Et donc finalement :

$$\mathbb{P}(\hat{p} \geq p + \frac{t}{\sqrt{n}}) \leq \exp(\frac{-nt^2}{2n + \frac{2}{3}t\sqrt{n}})$$

En utilisant le fait que $\forall i \in \{1, \ldots, n\}, \sigma_i = p(1-p) \leq 1$ et $B \leq 1$. D'autre part, comme précédemment avec l'inégalité de Hoeffding, on applique l'inégalité de Bernstein à l'échantillon des $-X_i$. La variance reste identique, l'espérance change seulement de signe, on obtient :

$$\mathbb{P}(\sum_{i=1}^{n}(-X_i) - \mathbb{E}[-X_i] \geq t\sqrt{n}) \leq \exp(\frac{-nt^2}{2n + \frac{2}{3}t\sqrt{n}})$$

Ceci nous donner finalement pour Bernstein, en raisonnant de la même façon qu'on l'a fait avec Hoeffding :

$$\mathbb{P}(|\hat{p} - p| \geq \frac{t}{\sqrt{n}}) \leq 2\exp(\frac{-nt^2}{2n + \frac{2}{3}t\sqrt{n}})$$

On a obtenu deux inégalités qui ne dépendent pas du paramètre inconnu p, ce qui est le cas pour toutes les inégalités obtenues précédemment, dans la première section puis avec l'inégalité de Bienaymé-Tchebychev. Ces inégalités nous informent quant à la précision de l'estimation de p par $\hat{p}$. En effet, les trois inégalités obtenues dans cette deuxième section nous donnent les intervalles suivants :

Pour Bienaymé-Tchebychev : $\mathbb{P}(\hat{p}_n \in [p - \frac{t}{\sqrt{n}}, p + \frac{t}{\sqrt{n}}] \geq 1 - \frac{p(1-p)}{t^2}$
Pour Hoeffding : $\mathbb{P}(\hat{p}_n \in [p - \frac{t}{\sqrt{n}}, p + \frac{t}{\sqrt{n}}] \geq 1 - 2e^{-2t^2}$
Pour Bernstein : $\mathbb{P}(\hat{p}_n \in [p - \frac{t}{\sqrt{n}}, p + \frac{t}{\sqrt{n}}] \geq 1 - 2\exp(\frac{-nt^2}{2n + \frac{2}{3}t\sqrt{n}})$

Deux "types" de précision sont à prendre en compte. Tout d'abord, pour une longueur d'intervalle donnée, laquelle de ces bornes nous assure la plus grande probabilité d'avoir un estimateur proche de la valeur réelle du paramètre. Deuxièmement, pour une même valeur de la borne, quel est l'intervalle qui est de plus petite taille. La taille de l'intervalle dépend à la fois de la valeur de t et de celle de n, donc du nombre d'observations. La valeur de n ne peut pas tendre vers l'infini en pratique, ce qui est un défaut des bornes obtenues dans la première section du projet, qui étaient des bornes asymptotiques. Dans le cas des bornes de la seconde section, nos intervalles ne sont pas asymptotiques, ce qui permet d'avoir des résultats plus précis a priori, mais nous n'avons pas traité cela.
Pour Bienaymé-Tchebytchev, la borne n'est pas calculable, ce qui nous pousse à la rejeter directement.
Hoeffding quant à elle ne dépend pas du nombre d'observations, donc augmenter le nombre d'individus dans l'échantillon réduit la taille de l'intervalle pour une borne de probabilité donnée, mais n'augmente pas simultanément à cela la borne de probabilité.
Enfin, la borne obtenue avec Bernstein dépend en revanche aussi de n, mais elle est équivalente à la borne de Hoeffding pour n assez grand, reste à voir ce qu'il se passe selon les valeurs de n. Nous avons fait des simulations, et Hoeffding reste une meilleure borne, comme nous allons le voir, pour des valeurs de n raisonnables pour ce genre d'expériences. Passons à l'interprétation des figures à présent.

Comme dit précédemment, deux types de précision sont importants, il nous faut trouver le meilleur arbitrage entre taille de l'intervalle $[p - \frac{t}{\sqrt{n}}, p + \frac{t}{\sqrt{n}}]$ et le niveau de certitude quant à la présence de $\hat{p}_n$ dans cet intervalle. Pour cela, nous avons essayé pour plusieurs valeurs de n, de comparer les bornes calculées, et la proportion observée d'estimateurs effectivement dans l'intervalle $[p - \frac{t}{\sqrt{n}}, p + \frac{t}{\sqrt{n}}]$ pour $p = \frac{1}{2}$, en fonction de la valeur de t, qui on le rappelle est positif. Lorsque t augmente, les bornes augmentent, mai il est de même pour la taille de l'intervalle : on est plus certain d'être dans l'intervalle puisque ce dernier est plus large. L'augmentation de n réduit la taille de l'intervalle sans faire bouger les bornes pour n assez grand.
On souhaite un niveau de précision élevé, que nous avons fixé à 95%. La borne qui est la plus proche de la proportion effectivement observée de $\hat{p}$ dans l'intervalle (en rouge sur la figure 4) est celle de Hoeffding, pour un niveau de confiance de 95% (ligne horizontale orange sur la figure 4). C'est celle qui nous permet d'avoir l'intervalle de plus petit possible pour ce niveau de confiance minimal de 95 pourcent, puisque en utilisant les autres bornes, on devrait agmenter la valeur de t donc la taille de l'intervalle pour obtenir un tel niveau de confiance. On peut alors calculer une valeur de t optimale, qui correspond à l'intersection entre la borne de Hoeffding et la droit $y = 0.95$.

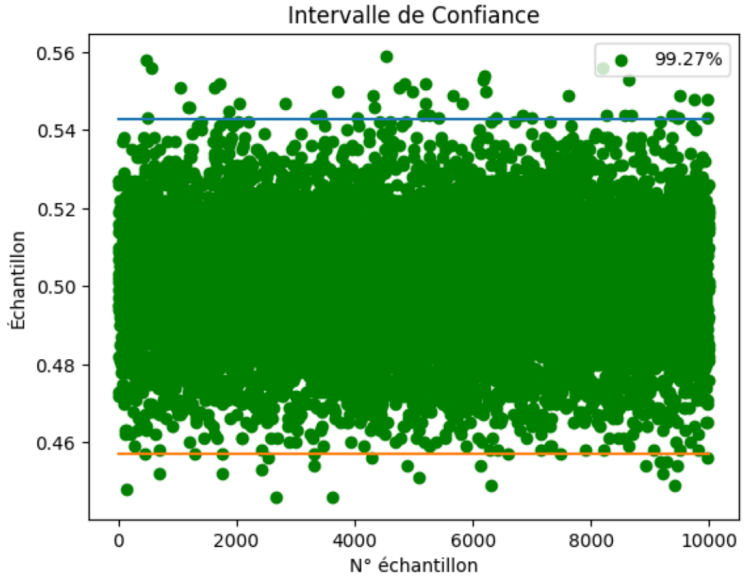


FIGURE 3 – Proportion de $\hat{p}$ dans l'intervalle de Hoeffding

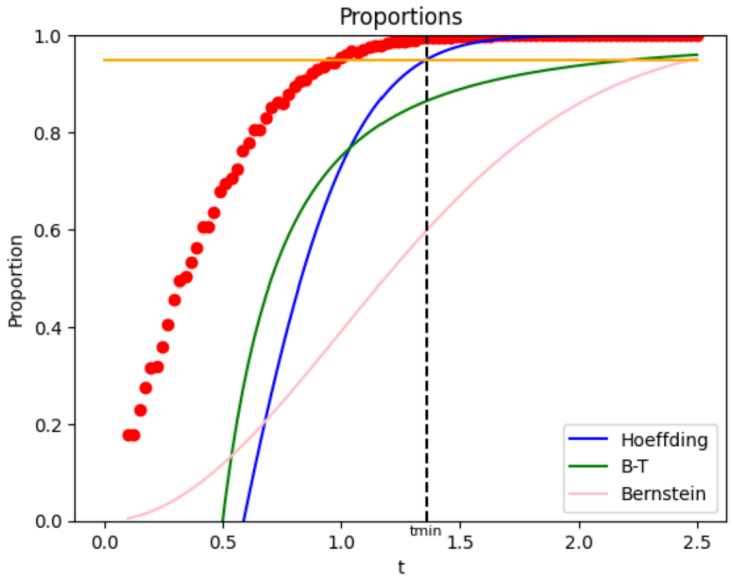


FIGURE 4 – Graphiques des trois bornes et des proportion de $\hat{p}$ dans l'IDC selon t

3 Comparaison entre deux proportions

Nous cherchons à décider entre deux options A et B en fonction des proportions observées $\hat{p}_A$ et $\hat{p}_B$. Nous analysons la règle de décision basée sur :

$$\hat{p}_B \leq \hat{p}_A + \delta + \sqrt{\frac{-\log\gamma}{2}}\left(\frac{1}{\sqrt{n_A}} + \frac{1}{\sqrt{n_B}}\right). \tag{6}$$

Pour $\delta > 0$ et $\gamma \in]0,1[$.
On considère un échantillon $(X_1, X_2, ..., X_n)$ de loi de Bernoulli de paramètre $p \in [0,1]$.

De plus, on a pour tout $i \in \{1, 2, \ldots, n\}, X_i \in [0,1]$. Par conséquent, par le théorème de Hoeffding utilisé précédemment, on a :

$$\begin{aligned}
\forall t > 0, \mathbb{P}(\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) \geq t\sqrt{n}) &\leq e^{-2t^2} \\
\Rightarrow \mathbb{P}(\sum_{i=1}^{n} X_i - np \geq t\sqrt{n}) &\leq e^{-2t^2} \\
\Rightarrow \mathbb{P}(\sum_{i=1}^{n} X_i \geq p + \frac{t}{\sqrt{n}}) &\leq e^{-2t^2} \\
\Rightarrow \mathbb{P}(\hat{p}_n \geq p + \frac{t}{\sqrt{n}}) &\leq e^{-2t^2}
\end{aligned}$$

On pose les deux évènements suivants :

$$\begin{aligned}
E_B &: \hat{p}_b \leq p_B + \frac{t}{\sqrt{n}} \\
E_A &: \hat{p}_b \leq p_A + \frac{t}{\sqrt{n}}
\end{aligned}$$

Ainsi que l'évènement B : "choisir l'option B". L'évènement B est équivalent à l'évènement :

$$\hat{p}_B > \hat{p}_A + \delta + \sqrt{-\frac{\ln(\gamma)}{2}}(\frac{1}{\sqrt{n_A}} + \frac{1}{\sqrt{n_B}}), \text{ où } \gamma = e^{-2t^2}.$$

De plus, on a :

$$\begin{aligned}
E_A \cap E_B \Rightarrow \hat{p}_B - \hat{p}_A &\leq p_B - p_A + t(\frac{1}{\sqrt{n_B}} - \frac{1}{\sqrt{n_A}}) \\
\Rightarrow \hat{p}_B - \hat{p}_A &\leq \gamma + t(\frac{1}{\sqrt{n_B}} - \frac{1}{\sqrt{n_A}}), \text{ car } p_B - p_A \leq \gamma \\
\Rightarrow \hat{p}_B - \hat{p}_A &\leq \gamma + t(\frac{1}{\sqrt{n_B}} + \frac{1}{\sqrt{n_A}}) \\
\Rightarrow \hat{p}_B &\leq \gamma + \hat{p}_A + t(\frac{1}{\sqrt{n_B}} + \frac{1}{\sqrt{n_A}})
\end{aligned}$$

Or,

$$\begin{aligned}
\mathbb{P}(E_A \cap E_B) = 1 - \mathbb{P}((E_A \cap E_B)^{\complement}) &= 1 - \mathbb{P}((E_A)^{\complement} \cup (E_B)^{\complement}) \\
&= 1 - \mathbb{P}((E_A)^{\complement}) - \mathbb{P}((E_B)^{\complement}) + \mathbb{P}((E_A)^{\complement} \cap E_B)^{\complement}) \\
&\geq 1 - \mathbb{P}((E_A)^{\complement}) - \mathbb{P}((E_B)^{\complement}) \\
&\geq 1 - 2e^{-2t^2}
\end{aligned}$$

Par suite, on a :

$$\begin{aligned}\mathbb{P}((E_A \cap E_B)^{\complement}) &= 1 - \mathbb{P}(E_A \cap E_B) \\ &\leq 1 - (1 - 2e^{-2t^2} \\ &= 2e^{-2t^2} \\ &= 2\gamma \text{ , avec } \gamma = e^{-2t^2}\end{aligned}$$

Or, on a aussi :

$$\mathbb{P}(B) \leq \mathbb{P}((E_A \cap E_B)^{\complement})$$

Car :
Supposons que l'évènement B se réalise, et que l'on a pas l'évènement $(E_A \cap E_B)^{\complement}$.
Dans ce cas, on a $E_A \cap E_B$, donc :

$$\hat{p}_B \leq p_B + \frac{t}{\sqrt{n_B}}, \text{ et } \hat{p}_A \leq p_A + \frac{t}{\sqrt{n_A}}$$

Ce qui, par ce qui précède donne :

$$\hat{p}_B \leq \hat{p}_A + \delta + \sqrt{\frac{-\log\gamma}{2}}\left(\frac{1}{\sqrt{n_A}} + \frac{1}{\sqrt{n_B}}\right).$$

Ce qui implique que l'on ne choisit pas B, ce qui est absurde.
Par conséquent, $B \subset (E_A \cap E_B)^{\complement}$, d'où :

$$\begin{aligned}\mathbb{P}(B) &\leq \mathbb{P}((E_A \cap E_B)^{\complement}) \\ &\leq 2e^{-2t^2} \\ &= 2\gamma\end{aligned}$$

On observe numériquement que la probabilité de choisir B sachant qu'on devait choisir A est de moins de 1% (environ 1 chance sur 100000 sur les simulations), ce résultat est disponible en fin d'annexe A.1.

4 Conclusion

Au terme de ce projet, nous avons conçu et analysé une méthode de décision pour l'A/B Testing reposant sur la modélisation des taux de succès par des variables aléatoires de type Bernoulli. En comparant les performances des deux groupes à l'aide de l'estimateur de proportion et de leur différence, nous avons formulé une règle de décision basée sur un seuil δ, permettant de trancher en faveur d'un groupe ou de conclure à une égalité.

Nous avons ensuite évalué la fiabilité de cette méthode à l'aide d'inégalités de concentration, notamment celles de Bienaymé-Tchebychev, Hoeffding et Bernstein, qui nous ont permis d'encadrer la probabilité d'erreur de manière non asymptotique. Cela nous a permis de justifier mathématiquement que, pour un choix judicieux de δ, cette probabilité d'erreur pouvait être rendue arbitrairement faible en augmentant la taille des échantillons.

Enfin, une application numérique présente en annexe A.2 illustre la mise en œuvre concrète de notre méthode et valide nos résultats théoriques : nous avons montré qu'il était possible de prendre une décision fiable à partir de données simulées, avec un taux d'erreur effectivement contrôlé.

A Annexes

A.1 Notebook des graphiques et autres résultats numériques

Ce notebook contient toute la partie numérique du sujet, donc les graphiques, vérifications empiriques et autres illustrations accompagnés de leurs codes en Python.

AB_Testing

April 14, 2025

[11]:
```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sc
```

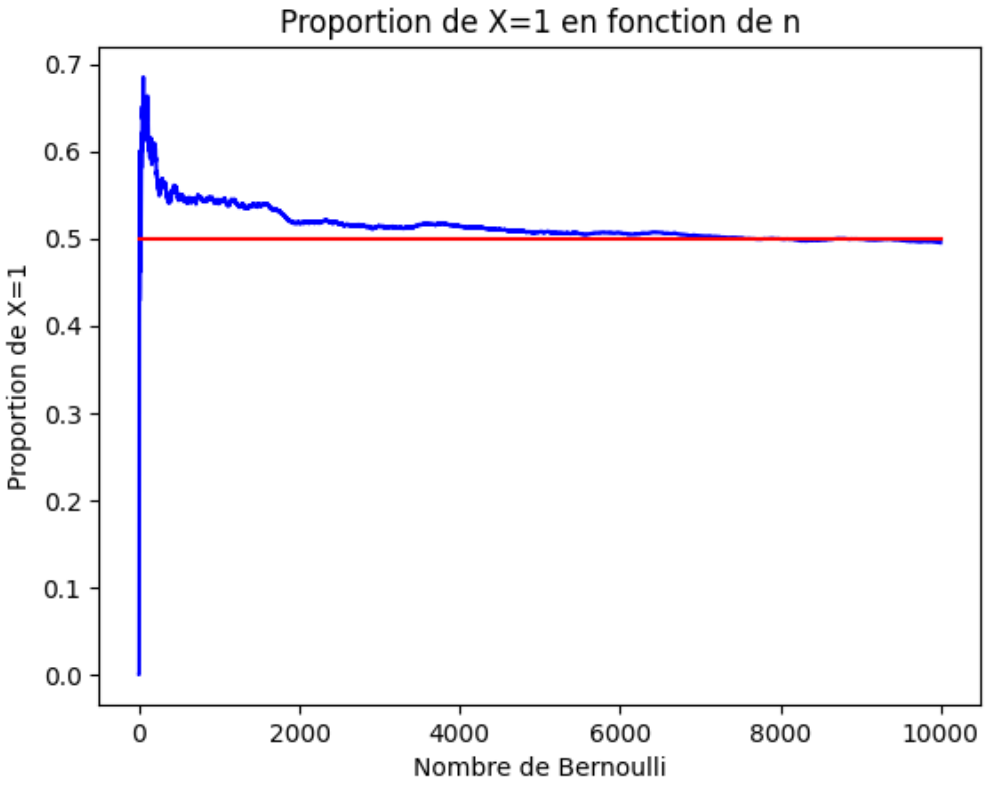
-> Fonction permettant de générer les lois binomiales

[12]:
```
def Gen_M_Binomiale(m,n,p):
    """
    Gen_M_Binomiale(m,n,p)
    Génère m v.a X~B(n,p)

    Paramètres :

    m (int): le nombre de v.a à générer

    p (float 0<p<1): le paramètre de la loi

    n (int): le nombre de variables de Bernouilli

    Retourne :
    np.array: les valeurs des m v.a
    """

    return np.random.binomial(n,p,m)
```

-> Fonction illustrant la loi des grands nombres

[13]:
```
def TracerCV(n,p):
    """
    TracerCV(m,p)
    Trace un graph des 1/n*sum(Xi), l'objectif est d'illuster la LGN pour une␣
 ↪B(n,p).

    Paramètres :
    p (float 0<p<1): le paramètre de la loi

    n (int): le nombre de variables de Bernouilli

    Retourne :
```

```
    None
    """

    X = np.random.binomial(1,p,n)
    somme = np.cumsum(X)
    Y = somme / (np.arange(1,n+1))
    plt.figure()
    plt.plot(np.linspace(1,n,n), Y, color='blue')
    plt.plot([0,n],[p,p],color='red')
    plt.title("Proportion de X=1 en fonction de n")
    plt.xlabel("Nombre de Bernoulli")
    plt.ylabel("Proportion de X=1")
    plt.show()

TracerCV(10000,0.5)
```



-> Fonction générant une moyenne empirique d'une v.a des $(X_1,...,X_n)$

[14]:
```
def Gen_Proportion(m,n,p):
```

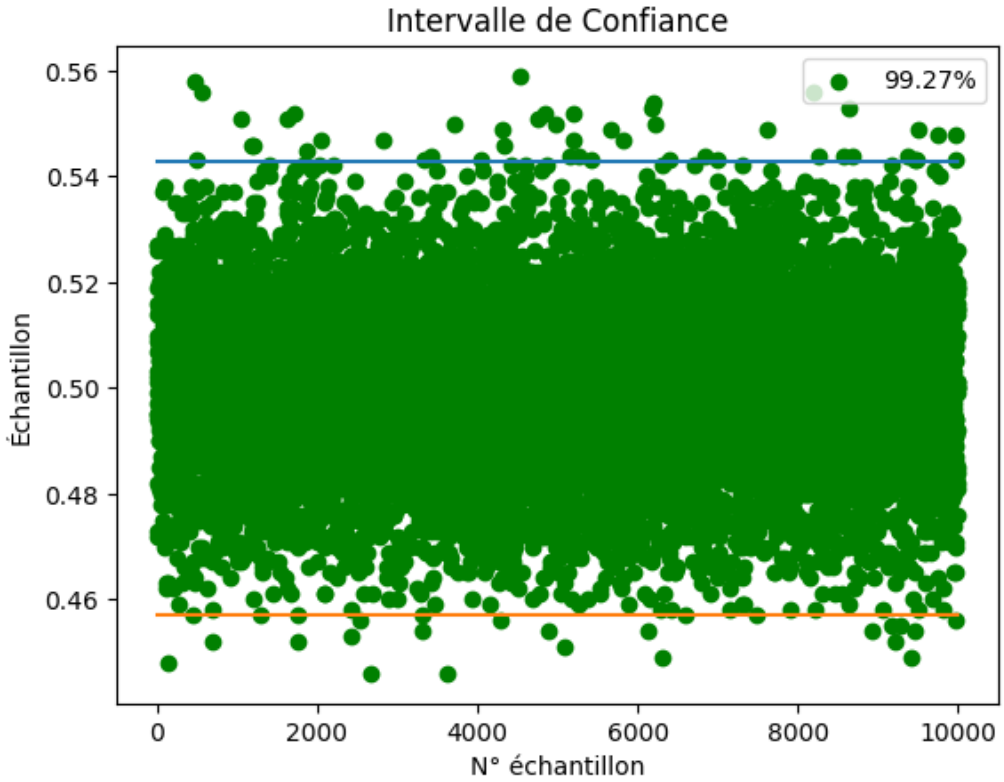
```
    """
    Gen_Proportion(m,n,p)
    Génère m valeur de Binomiale(n,p)/n

    Paramètres :
    p (float 0<p<1): le paramètre de la loi

    n (int): le nombre de variables de Bernouilli

    m (int): le nombre de Norm_Binomiale à générer

    Retourne :
    np.array : les m valeurs de Binomiale/n
    """

    return (1/n)*np.random.binomial(n,p,m)
```

-> Fonction générant une proportion de v.a de loi binomiale étant dans l'intervalle construit

[15]:
```
def ProportionInIC(echp,n,p,t):
    """
    ProportionInIc(echp,n,p,t)
    Compte la proportion de points dans l'intervalles de confiance

    Paramètres :

    echp (np.array): Un array de v.a Binomiale/n

    n (int): le nombre de variables de Bernouilli

    p (float 0<p<1): le paramètre de la loi

    t (float t>0): le t choisi pour l'intervalle de confiance


    Retourne :
    (float) la proportion de points dans l'intervalle de confiance
    """

    m = len(echp)
    res1 = np.where(echp>=p-(t/np.sqrt(n)),True,False)
    res2 = np.where(echp<=p+(t/np.sqrt(n)),True,False)
    res = np.logical_and(res1, res2)
    somme = np.count_nonzero(res)
    return (1/m)*somme
```

-> Fonction affichant les proportions et l'intervalle

[16]:
```
def TracerIntervalleconf(echp,n,p,t):
    """
    TracerIntervalleconf(echp,n,p,t,m)
    ///

    Paramètres :

    echp (np.array): Un array de v.a Binomiale/n

    n (int): le nombre de variables de Bernouilli

    p (float 0<p<1): le paramètre de la loi

    t (float t>0): le t choisi pour l'intervalle de confiance

    Retourne :
    None
    """

    m = len(echp)
    plt.figure()
    #plt.ylim((0.4,0.6))
    prop = ProportionInIC(exprop,1000,0.5,t)
    plt.plot([0,m],[p+(t/np.sqrt(n)),p+(t/np.sqrt(n))])
    plt.plot([0,m],[p-(t/np.sqrt(n)),p-(t/np.sqrt(n))])
    plt.scatter(np.linspace(1,m,m),echp, color='green',label = f'{prop*100}%')
    plt.title("Intervalle de Confiance")
    plt.xlabel("N° échantillon")
    plt.ylabel("Échantillon")
    plt.legend()
    plt.show()

exprop = Gen_Proportion(10000,1000,0.5)
TracerIntervalleconf(exprop,1000,0.5,1.358102)
```



-> Hoeffding, Bienaimé-Tchebytchev, Bernstein

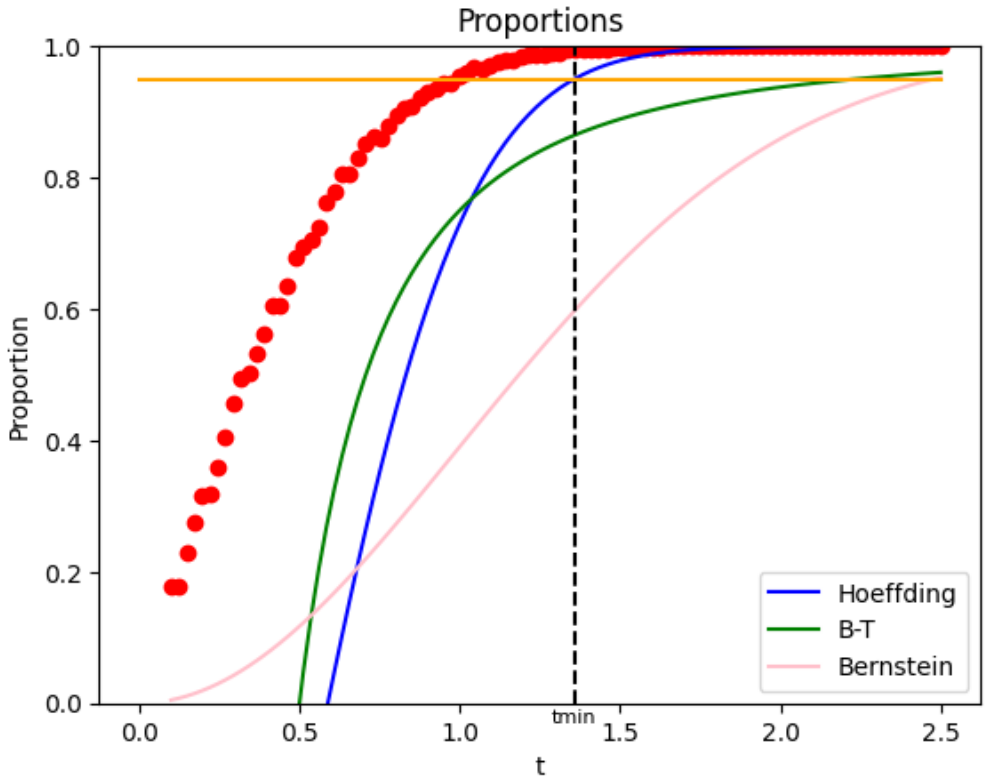
[17]:
```python
def Hoeff(x):
    """
    Hoeff(x)
    Fonction de x -> 1-2*np.exp(-2*(x**2))

    Paramètres :

    (float) x


    Retourne :
    1-2*np.exp(-2*(x**2))
    """

    return 1-2*np.exp(-2*(x**2))


def BT(x,p):
```

```
    """
    BT(x,p)
    Fonction de x -> 1-(p*(1-p))/(x**2)

    Paramètres :

    (float) x

    p (float 0<p<1): le paramètre de la loi


    Retourne :
    1-(p*(1-p))/(x**2)
    """

    return 1-(p*(1-p))/(x**2)


def Bernstein(x,n):
    """
    Bernstein(x,n)
    Fonction de x -> 1-np.exp(-n*(x**2)/((2*n)+(2/3)*np.sqrt(n)*x))

    Paramètres :

    (float) x

    n (int n>0): le nombre de Bernoulli dans chaque Binomiale


    Retourne :
    1-np.exp(-n*(x**2)/((2*n)+(2/3)*np.sqrt(n)*x))
    """

    return 1-np.exp(-n*(x**2)/((2*n)+(2/3)*np.sqrt(n)*x))
```

-> Graph des trois courbes et t_0

[18]:
```
def TraceProp(n,p,m,l):
    """
    TraceProp(n,p,m)
    ///

    Paramètres :

    n (int n>0): le nombre de Bernoulli dans chaque Binomiale

    p (float 0<p<1): le paramètre de la loi
```

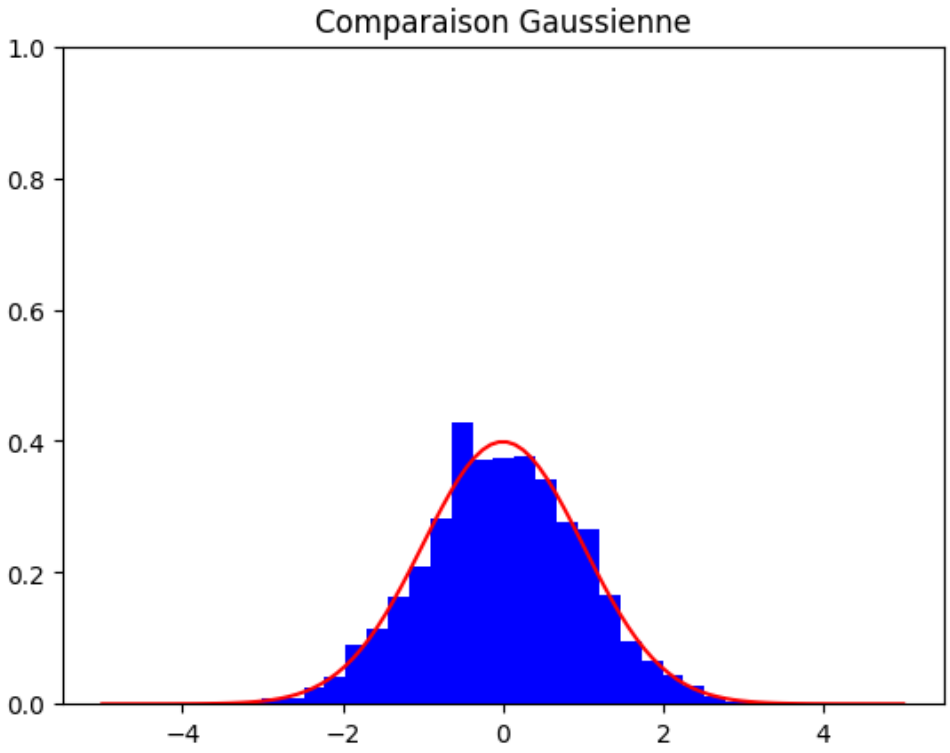
```
    m (int): le nombre de proportion à générer

    l (int): le nombre de proportion à vérifier


    Retourne :
    None
    """

    T = np.linspace(0.1,2.5,l)
    echp = [Gen_Proportion(m,n,p) for i in range(l)]
    echprop = [ProportionInIC(echp[i],n,p,T[i]) for i in range(l)]
    plt.figure()
    plt.scatter(T,echprop,color='red')
    plt.plot(T,[Hoeff(t) for t in T],color='blue',label="Hoeffding")
    plt.plot(T,[BT(t,p) for t in T],color='green',label="B-T")
    plt.plot(T,[Bernstein(t,n) for t in T],color='pink',label="Bernstein")
    plt.plot([0,2.5],[0.95,0.95],color="orange")
    plt.axvline(x=(1/np.sqrt(2))*np.sqrt(np.log(2)-np.log(0.05)),␣
↪color='black', linestyle='--')
    plt.text((1/np.sqrt(2))*np.sqrt(np.log(2)-np.log(0.05)), -0.03, 'tmin',␣
↪color='black', fontsize=8, ha='center')
    plt.plot()
    plt.ylim(0,1)
    plt.title("Proportions")
    plt.xlabel("t")
    plt.ylabel("Proportion")
    plt.legend()
    plt.show()

TraceProp(1000,0.5,10000,100)
```



-> Fonction montrant que les proportions suivent une gaussienne pour n grand

[19]:
```
def TraceCR(n,p,m,l):
    """
    TraceCR(n,p,m,l)
    ///

    Paramètres :

    n (int n>0): le nombre de Bernoulli dans chaque Binomiale

    p (float 0<p<1): le paramètre de la loi

    m (int): le nombre de proportion à générer

    l (int): le nombre de proportion à vérifier


    Retourne :
    None
```

```
    """

    X = np.linspace(-5,5,1)
    echp = Gen_Proportion(m,n,p)
    plt.figure()
    plt.hist(2*np.sqrt(n)*(echp-p),30,density =True,color='blue')
    plt.plot(X, [sc.norm.pdf(x,0,1) for x in X], color='red')
    plt.ylim(0,1)
    plt.title("Comparaison Gaussienne")
    plt.show()

TraceCR(1000,0.5,10000,100)
```



-> Fonction calculant la proportion de $pB > pA + delta$

[20]:
```
def verif_Proba_B(pA, pB, na, nb, m, t=1.358102, delta=0):
    """
    verif_Proba_B(pA, pB, na, nb, m, t, delta)
    ///

    Paramètres :
```

```
    na (int n>0): le nombre d'utilisateurs ayant évalué A

    nb (int n>0): le nombre d'utilisateurs ayant évalué B

    pA (float 0<p<1): paramètre de la loi pour A

    pB (float 0<p<pA + delta): paramètre de la loi pour B

    m (int): le nombre de proportion à générer

    t (float): tmin déterminer précédemment

    delta (float): Cran de sureté donné


    Retourne :
    Le taux de pB_hat > pA_hat + delta
    """

    if pA + delta < pB:
        return 'Input pB > pA + delta'

    A_conserv_emp = np.zeros(m)
    A_conserv_emp = np.array(A_conserv_emp, dtype='bool')
    ech_pA = (1/na)*np.random.binomial(na,pA,m)
    ech_pB = (1/nb)*np.random.binomial(nb,pB,m)

    for i in range(m):
        A_conserv_emp[i] = ech_pB[i] <= ech_pA[i]*1.02 + delta + t*(1/np.
↪sqrt(na) + 1/np.sqrt(nb))
        #On prend delta = 2% de pA

    ProbaB = m - A_conserv_emp.sum()
    return ProbaB/m

print(verif_Proba_B(0.51, 0.5, 1000, 1000, 1000000))
```

```
1e-06
```

A.2 Notebook de l'application à une campagne de pub

Cette application est basée sur le dataset de Kaggle 'A/B Testing DataSet' qui est un bon exemple d'application pour notre sujet, il s'agit du 1er cas d'utilisation de cette méthode de AB testing présenté en introduction (i.e : déterminer si un innovation vaut le coup d'être implémentée définitivement). Toutes les informations concernant ce Dataset sont dans le pdf ci-après.

AB_Testing_Analysis

April 14, 2025

[28]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import scipy.stats as stats
```

1 Contexte du test et du dataset :

Un exemple d'application du AB Testing est d'aider à trouver une meilleure approche pour trouver des clients, commercialiser des produits, obtenir une portée plus élevée, ou toute autre chose qui aide une entreprise à convertir la majorité de ses clients cibles en clients réels.

Voici toutes les features du jeu de données :

- **Campaign Name** : Le nom de la campagne
- **Date** : Date de l'enregistrement
- **Spend** : Montant dépensé pour la campagne en dollars
- **Impressions** : Nombre d'impressions que l'annonce a obtenues pendant la campagne
- **Reach** : Nombre d'impressions uniques reçues par l'annonce
- **Website Clicks** : Nombre de clics sur le site web reçus par les annonces
- **Searches** : Nombre d'utilisateurs ayant effectué des recherches sur le site
- **View Content** : Nombre d'utilisateurs ayant vu du contenu et des produits sur le site
- **Add to Cart** : Nombre d'utilisateurs ayant ajouté des produits au panier
- **Purchase** : Nombre d'achats

Deux campagnes ont été menées par l'entreprise :

- **Control Campaign** : Campagne de contrôle, avec les paramètres habituels
- **Test Campaign** : Campagne de test, avec les nouveaux paramètres

1.0.1 Importation et préparation des données

[29]:
```
control_df = pd.read_csv('control_group.csv', delimiter=';')
test_df = pd.read_csv('test_group.csv', delimiter=';')

full_df = pd.concat((control_df, test_df))
```

[30]:
```
full_df.sample(n=5)
```

[30]:
```
        Campaign Name        Date  Spend [USD]  # of Impressions    Reach  \
10  Control Campaign  11.08.2019         2490          115247.0  95843.0
27     Test Campaign  28.08.2019         2247           54627.0  41267.0
22     Test Campaign  23.08.2019         2407           60286.0  49329.0
4   Control Campaign   5.08.2019         1835               NaN      NaN
21  Control Campaign  22.08.2019         2939          105705.0  86218.0

    # of Website Clicks  # of Searches  # of View Content  # of Add to Cart  \
10               8137.0         2941.0             2486.0            1887.0
27               8144.0         2432.0             1281.0            1009.0
22               5077.0         2592.0             2004.0             632.0
4                   NaN            NaN                NaN               NaN
21               6843.0         3102.0             2988.0             819.0

    # of Purchase
10          475.0
27          721.0
22          473.0
4             NaN
21          387.0
```

[31]:
```
full_df = full_df.rename(
    columns=lambda x: x.replace("# of ", "") if x.startswith("# of ") else x)
full_df = full_df.rename(columns={"Campaign Name": "Campaign"})

full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60 entries, 0 to 29
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Campaign        60 non-null     object
 1   Date            60 non-null     object
 2   Spend [USD]     60 non-null     int64
 3   Impressions     59 non-null     float64
 4   Reach           59 non-null     float64
 5   Website Clicks  59 non-null     float64
 6   Searches        59 non-null     float64
 7   View Content    59 non-null     float64
 8   Add to Cart     59 non-null     float64
 9   Purchase        59 non-null     float64
dtypes: float64(7), int64(1), object(2)
memory usage: 5.2+ KB
```

[32]:
```
full_df[full_df.isnull().any(axis=1)]
```

[32]:
```
            Campaign       Date  Spend [USD]  Impressions  Reach  \
4  Control Campaign  5.08.2019         1835          NaN    NaN

   Website Clicks  Searches  View Content  Add to Cart  Purchase
4             NaN       NaN           NaN          NaN       NaN
```

[33]:
```
full_df = full_df.dropna()

full_df['Campaign'].value_counts()
```

[33]:
```
Campaign
Test Campaign       30
Control Campaign    29
Name: count, dtype: int64
```

[34]:
```
full_df.dtypes
```

[34]:
```
Campaign           object
Date               object
Spend [USD]         int64
Impressions       float64
Reach             float64
Website Clicks    float64
Searches          float64
View Content      float64
Add to Cart       float64
Purchase          float64
dtype: object
```

[35]:
```
full_df['Date'] = pd.to_datetime(full_df['Date'], format="%d.%m.%Y")
full_df[full_df.select_dtypes(include=['float64']).columns] = full_df.
 ↪select_dtypes(include=['float64']).astype('int64')

full_df.dtypes
```

[35]:
```
Campaign                  object
Date              datetime64[ns]
Spend [USD]                int64
Impressions                int64
Reach                      int64
Website Clicks             int64
Searches                   int64
View Content               int64
Add to Cart                int64
Purchase                   int64
dtype: object
```

[36]:
```
full_df['Date'].dt.year.value_counts(), full_df['Date'].dt.month.value_counts()
```

[36]:
```
(Date
 2019    59
 Name: count, dtype: int64,
 Date
 8    59
 Name: count, dtype: int64)
```

[37]:
```
#Les données ont été récupérées sur le mois d'août 2019, on garde donc␣
 ↪seulement le jour.
full_df["Date"] = full_df["Date"].dt.day
full_df.rename(columns={"Date": "Day"}, inplace=True)
full_df.head()
```

[37]:

	Campaign	Day	Spend [USD]	Impressions	Reach	Website Clicks	\
0	Control Campaign	1	2280	82702	56930	7016	
1	Control Campaign	2	1757	121040	102513	8110	
2	Control Campaign	3	2343	131711	110862	6508	
3	Control Campaign	4	1940	72878	61235	3065	
5	Control Campaign	6	3083	109076	87998	4028	

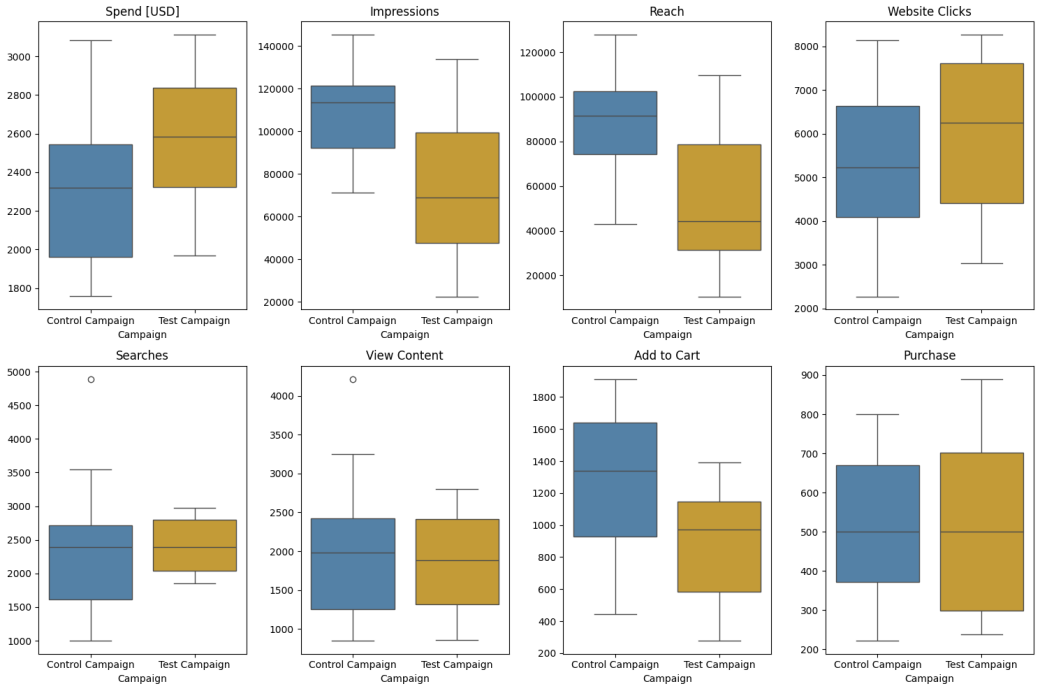
	Searches	View Content	Add to Cart	Purchase
0	2290	2159	1819	618
1	2033	1841	1219	511
2	1737	1549	1134	372
3	1042	982	1183	340
5	1709	1249	784	764

1.0.2 Traitement des outliers

[38]:
```
def any_outlier(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    limit_low = Q1 - 1.5*IQR
    limit_high = Q3 + 1.5*IQR

    not_outlier = df[col].between(limit_low, limit_high, inclusive='both').sum()

    outlier_percentage = (len(df[col]) - not_outlier)*100/len(df[col])

    return outlier_percentage

def remove_outlier(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
```

```
    limit_low = Q1 - 1.5*IQR
    limit_high = Q3 + 1.5*IQR

    return df[df[col].between(limit_low, limit_high, inclusive='both')]

def plot_numeric(func):
    fig, axes = plt.subplots(2, 4, figsize=(15, 10))
    axes = axes.flatten()

    numeric_cols = full_df.select_dtypes(include=['int']).columns.to_list()
    numeric_cols.remove('Day')

    for i, col in enumerate(numeric_cols):
        func(data=full_df, x='Campaign', y=col, ax=axes[i], hue='Campaign',␣
 ↪palette={'Control Campaign': 'steelblue', 'Test Campaign': '#DAA520'},␣
 ↪legend=False)
        axes[i].set_title(col)
        axes[i].set_ylabel('')

    plt.tight_layout()
    plt.show()
```

[39]:
```
plot_numeric(sns.boxplot)
```

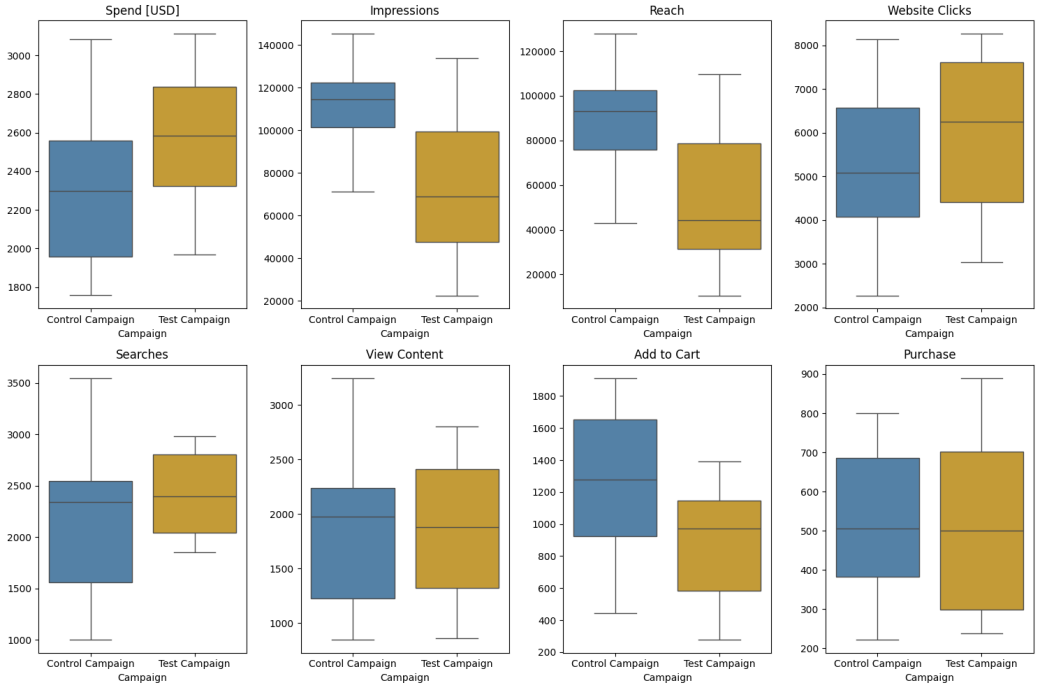


[40]:
```
numeric_cols = full_df.select_dtypes(include=["int"]).columns.to_list()
numeric_cols.remove("Day")

outlier_percentages = {}
for col in numeric_cols:
    outlier_percentage = any_outlier(full_df, col)
    outlier_percentages[col] = outlier_percentage

outlier_df = pd.DataFrame(outlier_percentages.items(), columns=["Feature",␣
 ↪"Outlier Percentage"])
outlier_df
```

[40]:

	Feature	Outlier Percentage
0	Spend [USD]	0.000000
1	Impressions	0.000000
2	Reach	0.000000
3	Website Clicks	0.000000
4	Searches	1.694915
5	View Content	1.694915
6	Add to Cart	0.000000
7	Purchase	0.000000

[41]:
```
removable_outliers = outlier_df[outlier_df['Outlier Percentage'].between(0,3,␣
 ↪inclusive='neither')]

for feature in removable_outliers.Feature :
    full_df = remove_outlier(full_df, feature)

plot_numeric(sns.boxplot)
```



1.0.3 Résumé des colonnes

[42]:
```
pd.set_option("display.max_columns", None)
full_df.drop(columns=["Day"]).groupby("Campaign").describe()
```

[42]:

Campaign	Spend [USD] count	Spend [USD] mean	Spend [USD] std	Spend [USD] min	Spend [USD] 25%	Spend [USD] 50%	Spend [USD] 75%	Spend [USD] max	Impressions count	Impressions mean	Impressions std	Impressions min	Impressions 25%	Impressions 50%	Impressions 75%	Impressions max	Reach count	Reach
Control Campaign	28.0	2301.535714	369.944989	1757.0	1956.5	2299.5	2557.00	3083.0	28.0	110185.857143	21818.408023	71274.0	101243.25	114338.5	122223.25	145248.0	28.0	
Test Campaign	30.0	2563.066667	348.687681	1968.0	2324.5	2584.0	2836.25	3112.0	30.0	74584.800000	32121.377422	22521.0	47541.25	68853.5	99500.00	133771.0	30.0	

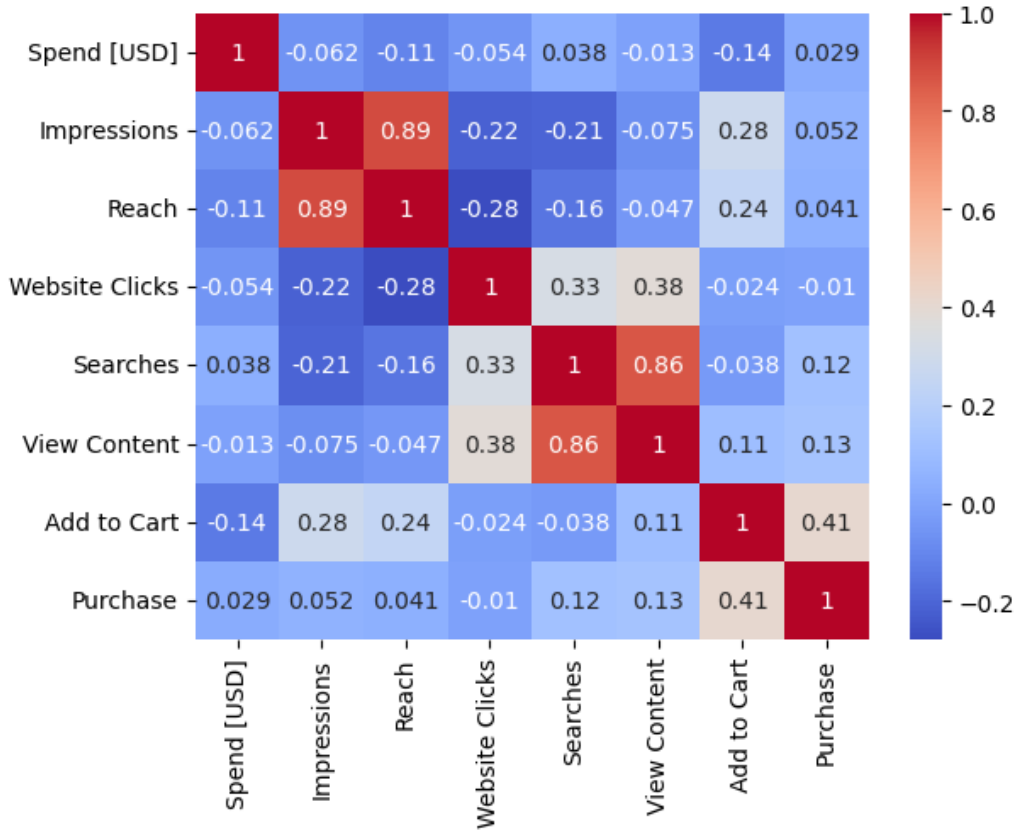
```
                                                                        \
                         mean           std      min       25%      50%
Campaign
Control Campaign  89368.250000  22046.974382  42859.0  76022.25  93237.5
Test Campaign     53491.566667  28795.775752  10598.0  31516.25  44219.5

                                  Website Clicks                \
                       75%       max          count         mean
Campaign
Control Campaign  102487.50  127852.0           28.0  5220.571429
Test Campaign      78778.75  109834.0           30.0  6032.333333

                                                                \
                         std     min      25%     50%      75%     max
Campaign
Control Campaign  1703.130482  2277.0  4070.75  5082.5  6572.50  8137.0
Test Campaign     1708.567263  3038.0  4407.00  6242.5  7604.75  8264.0

                Searches                                              \
                   count         mean         std     min     25%     50%
Campaign
Control Campaign    28.0  2125.964286  710.276810  1001.0  1560.5  2340.0
Test Campaign       30.0  2418.966667  388.742312  1854.0  2043.0  2395.5

                                  View Content                        \
                     75%     max        count         mean         std
Campaign
Control Campaign  2543.00  3549.0         28.0  1862.535714  654.520208
Test Campaign     2801.25  2978.0         30.0  1858.000000  597.654669

                                                    Add to Cart  \
                    min     25%     50%     75%     max       count
Campaign
Control Campaign  848.0  1224.0  1979.5  2239.5  3249.0        28.0
Test Campaign     858.0  1320.0  1881.0  2412.0  2801.0        30.0

                                                                \
                        mean         std    min    25%     50%     75%
Campaign
Control Campaign  1293.357143  413.332636  442.0  924.5  1279.0  1654.5
Test Campaign      881.533333  347.584248  278.0  582.5   974.0  1148.5

                       Purchase                                       \
                     max    count        mean         std    min    25%
Campaign
Control Campaign  1913.0     28.0  529.535714  184.760201  222.0  382.5
Test Campaign     1391.0     30.0  521.233333  211.047745  238.0  298.0
```

	50%	75%	max
Campaign			
Control Campaign	506.0	686.0	800.0
Test Campaign	500.0	701.0	890.0

1.0.4 Analyse des corrélations

[43]:
```
mat_corr = full_df.drop(columns=["Campaign", "Day"]).corr()
sns.heatmap(mat_corr, annot=True, cmap="coolwarm")
```

[43]: <Axes: >



1.0.5 Interprétation des corrélations

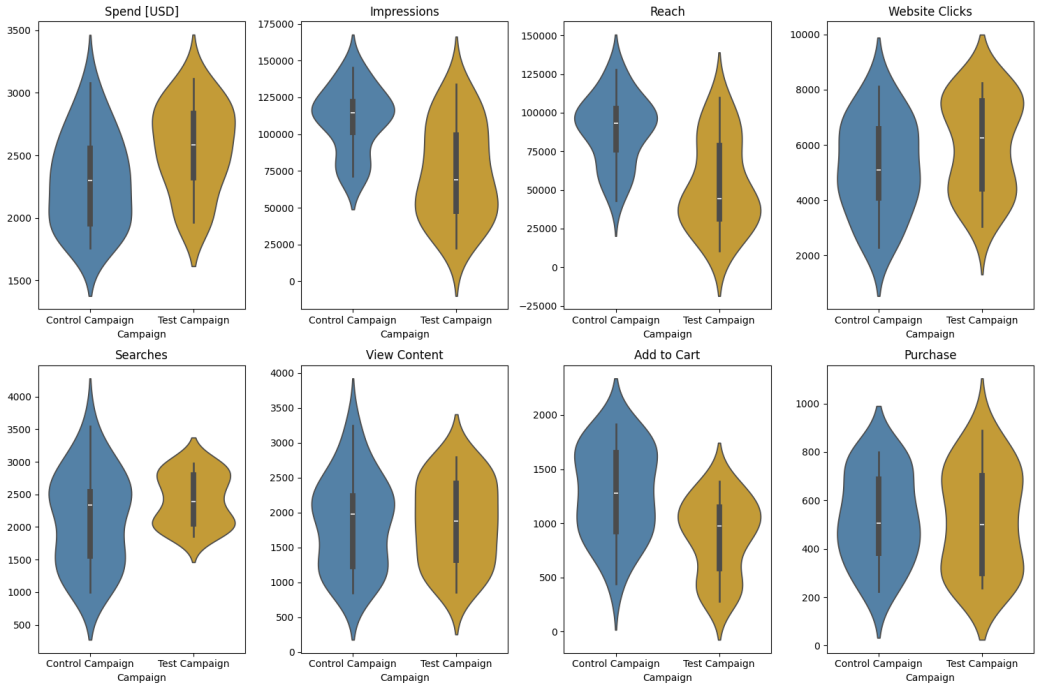
- On peut extraire deux couples de variables fortement corrélées :
  - Reach & Impressions
  - Searches & View Content

Cependant, Reach et Impressions désignent respectivement le nombre de personnes (unique) atteintes et le nombre d'impressions.

- Corrélation très faible entre Spend et Purchase :

Ceci semble indiquer que dépenser plus pour une campagne n'augmenterait pas nécessairement le nombre d'achats.

[44]:
```
plot_numeric(sns.violinplot)
```



1.0.6 Interprétation des diagrammes en violon

Spend [USD]:

- Les deux distributions sont bien différentes, avec une médiane plus élevée pour la campagne de test.
- La dispersion semble élevée dans les deux groupes, ce qui indique une large gamme de dépense.

Impressions:

- La médiane est plus élevée pour la campagne de contrôle.
- La dispersion de la campagne de test est grande, et est plus dense dans les valeurs plus faibles.

Reach:

- Similaire à 'Impressions' ce qui conforte l'idée énoncée précédemment.

Website Clicks:

- La campagne de test à l'air de mieux convertir en taux de clicks.

Searches :

- Médianes équivalentes et dispersion plus faible pour la campagne de test.

Add to Cart:

- Malgré un taux de 'Website clicks' plus élevé, on observe une légère baisse sur ce critère.

Purchase (Intérêt ici):

- On remarque que malgré la baisse de 'Reach', les valeurs de 'Purchase' restent globalement similaires.
- Premier quartile un légèrement plus faible.

1.0.7 Conclusion de l'interprétation avant Test statistique

On peut imaginer que la campagne de test se base sur des dépenses plus élévées ('Spend [USD]'), mais cela ne semble pas affecter positivement les centres d'intérêt ici, à savoir les impressions uniques ('Impressions') et le nombre de vente ('Purchase').

1.0.8 Test statistique n°1 (Hoeffding):

Test n°1 : Implémentation du Test créé lors du projet

[45]:
```
def HoeffdingT(cols, delta, df=full_df, t=1.358102):
    c_df = df[df['Campaign'] == 'Control Campaign'].copy()
    t_df = df[df['Campaign'] == 'Test Campaign'].copy()

    nA = c_df['Reach'].sum() #Nombre de personnes ayant vu la campagne de␣
↪contrôle
    nB = t_df['Reach'].sum() #Nombre de personnes ayant vu la campagne de test

    res = []

    for i, col in enumerate(cols):
        hatPA = c_df[col].sum()/nA
        hatPB = t_df[col].sum()/nB

        threshold = hatPA + delta[i] + t * (1 / np.sqrt(nA) + 1 / np.sqrt(nB))

        decision = "On choisit la campagne de contrôle" if hatPB <= threshold␣
↪else "On choisit la campagne de test"

        res.append([col, hatPA, hatPB, threshold, decision])

    return res
```

[46]:
```
res_obt = HoeffdingT(['Purchase', 'Website Clicks', 'Add to Cart'], [0.0003, 0.
↪003, 0.00075])
#On prend ici delta = 5% du pA observé

for var in res_obt:
    print(f'Variable : {var[0]} || Décision : {var[4]}')
```

```
Variable : Purchase || Décision : On choisit la campagne de test
Variable : Website Clicks || Décision : On choisit la campagne de test
Variable : Add to Cart || Décision : On choisit la campagne de contrôle
```

Conclusion (Test n°1):

- *Purchase* -> Le test indique qu'il vaut mieux choisir la campagne de test.
- *Website Clicks* -> Le test indique qu'il vaut mieux choisir la campagne de test.
- *Add to Cart* -> Le test indique qu'il vaut mieux choisir la campagne de contrôle.

Test n°2 : Comparaison des résultats obtenus avec le Test t : Pour la suite, on va utiliser le Test t (resp. le test Mann-Whitney U) si la variable à tester est gaussienne (resp. non gaussienne). On utilise aussi le test de Levene pour déterminer si les variances sont homogènes ou non.

Implémentation du Test t

Hypothèses du test t : -> Purchase * H (hypothèse nulle) : Il n'y a pas de différence significative entre les taux de conversion des deux campagnes.

- H (hypothèse alternative) : Les taux de conversion de la campagne de test est plus grande que celle de contrôle.

-> Website Clicks * H (hypothèse nulle) : Il n'y a pas de différence significative entre les taux de clicks des deux campagnes.

- H (hypothèse alternative) : Les taux de conversion des deux campagnes sont différentes.

-> Add to Cart * H (hypothèse nulle) : Il n'y a pas de différence significative entre les taux de mise en panier des deux campagnes.

- H (hypothèse alternative) : Le taux d'ajout au panier de la campagne de contrôle est plus élevé.

[47]:
```
def check_normality(col, alpha=0.05):
    stat, p = stats.shapiro(col) #On utilise shapiro car nous n'avons pas␣
↪beacoup de données (n<50)

    if p < alpha:
        return "Not Normal"
    return "Normal"
```

[48]:
```
def test_T(cols, df=full_df, alt='two-sided'):
    c_df = df[df['Campaign'] == 'Control Campaign'].copy()
    t_df = df[df['Campaign'] == 'Test Campaign'].copy()

    res = []

    for i, col in enumerate(cols):

        if check_normality(c_df[col]) == "Normal" and␣
 ↪check_normality(t_df[col]) == "Normal":
            stat, p_lev = stats.levene(c_df[col], t_df[col])
            equal_var = p_lev > 0.05

            stat, p_value = stats.ttest_ind(c_df[col], t_df[col],␣
 ↪equal_var=equal_var, alternative=alt[i])
        else:
            stat, p_value = stats.mannwhitneyu(c_df[col], t_df[col],␣
 ↪alternative=alt[i])

        decision = "Rejetter H0" if p_value < 0.05 else "Ne pas rejetter H0"

        res.append([col, stat, p_value, decision])



    return res
```

[49]:
```
res_comp = test_T(['Purchase', 'Website Clicks', 'Add to Cart'], alt=['less',␣
 ↪'two-sided', 'greater'])

for var in res_comp:
    print(f'Variable : {var[0]} || Décision : {var[3]}')
```

```
Variable : Purchase || Décision : Ne pas rejetter H0
Variable : Website Clicks || Décision : Ne pas rejetter H0
Variable : Add to Cart || Décision : Rejetter H0
```

Conclusion (Test n°2):

- *Purchase* -> Le test indique qu'il n'y a pas de différence significative entre la campagne de contrôle et celle de test. Donc on choisit celle de contrôle.
- *Website Clicks* -> Le test indique qu'il n'y a pas de différence significative entre la campagne de contrôle et celle de test. Donc on choisit celle de contrôle.
- *Add to Cart* -> Le taux d'ajout au panier de la campagne de contrôle est plus élevée. Donc on choisit celle de contrôle.

1.1 Résumé des Tests

On conclut finalement que notre test est plus adapté si une des deux campagnes n'est pas encore mise en place. En effet, on ne remarque pas ici de différence significative avec les tests classiques, ce qui explique que changer toute une campagne ne serait pas très avantageux pour si peu de gain. À l'inverse, si aucune campagne n'est déjà mise en place, il peut être intéressant de relevé les petites différences, d'où l'intérêt de notre test.

Voici un tableau récapitulatif de cette application :

Critère	Test maison (Hoeffding-like)	Test classique (t-test / Mann-Whitney)
Objectif	Décision pragmatique : *choisir entre deux options avec un seuil*	Détection d'une différence significative
Contrôle sur l'incertitude	*delta* explicite et interprétable	Seuil alpha implicite basé sur la p-valeur
Interprétation business	Très claire ("choisir A ou B selon le seuil")	Moins intuitive
Hypothèses	Aucune ou très peu	Normalité, égalité des variances, etc.