



# Mathématiques - Rapport

APALOO-KINGSLOVE Keziah, AUGIER Lucie

💖 Love Calculator 💖

Lu6A / maths\_2023 · GitLab

GitLab.com

 [https://gitlab.com/Lu6A/maths\\_2023](https://gitlab.com/Lu6A/maths_2023)

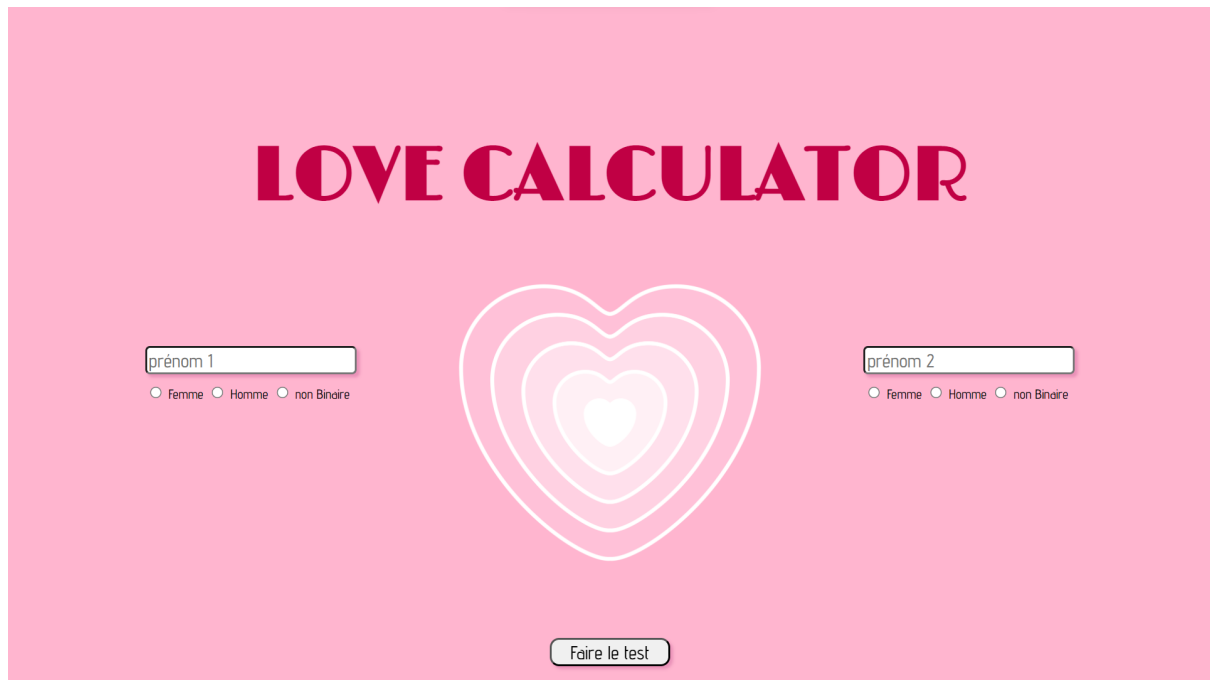


## Présentation du jeu :

Pour le projet de mathématiques, nous avons décidé de coder un "Love Calculator" en html/css/javascript. Le but du jeu est de calculer la compatibilité amoureuse entre deux prénoms.

Chaque prénom entré dans le jeu se voit attribué une fiche d'identité totalement arbitraire basée sur des variables aléatoires. En combinant les deux fiches d'identité créées, une fiche de compatibilité est générée, ainsi qu'un pourcentage global de compatibilité.

La fiche identité est constituée de six variables aléatoires. Elle contient l'âge, le nombre de frères et sœurs, le salaire, le nombre d'animaux, si elle est fumeuse ou non fumeuse, le taux de fidélité et le pourcentage de beauté de la personne. Ces fiches sont cachées et non diffusées à l'utilisateur. Ces fiches identité servent à calculer les variables de la fiche de compatibilité finale. La variance entre les deux variables aléatoires des deux fiches identités est calculée pour créer les nouvelle variable de la fiche de compatibilité avec la formule :  $Variable\_Compatibilité = 100 / (Variance(Variable\_id1, Variable\_id2))$ . Le pourcentage final est calculé à l'aide de tous les pourcentages obtenus dans la fiche de compatibilité grâce à une moyenne. D'un point de vue de l'interface utilisateur, plus le pourcentage est grand plus le cœur situé entre les deux prénoms grossit.



Page d'accueil du site

### Fiche d'identité pour chacun des prénoms (valeurs des caractéristiques attribuées aléatoirement) :

- âge
- nombre de frères et sœurs
- salaire
- nombre d'animaux
- fumeur/non fumeur (si c'est oui on génère un nb aléatoire entre 50 et 100 par ex)
- taux de fidélité
- beauté

### Fiche de compatibilité finale

Les choix des variables d'après lesquelles sont calculées toutes ces données sont totalement arbitraires.

- durée de relation → calculée d'après l'âge
- temps d'apparition de la première dispute → calculé d'après les salaires
- temps de réconciliation entre deux disputes → calculé d'après le nombre de frères et sœurs
- compatibilité sexuelle → calculée d'après le facteur fumeur/non fumeur et la fidélité
- compatibilité physique → calculée d'après beauté
- durée de vie commune (habiter ensemble) → calculé d'après le nombre d'animaux

# LOVE CALCULATOR

Keziah

☒ Femme ☐ Homme ☐ non Binaire



Lucie

☒ Femme ☐ Homme ☐ non Binaire

## DÉTAILS DE COMPATIBILITÉ

Durée de relation estimée :

11.4 an(s)

Durée de vie sous le même toit :

0 an(s)

Apparition de la première dispute :

7 semaine(s)

Temps de réconciliation moyen après une dispute :

5 heure(s)

Compatibilité physique :

53 %

Compatibilité sexuelle :

37 %

Recommencer le test

Keziah et Lucie n'habiteront peut-être pas ensemble, mais au moins elles sont suffisamment compatibles pour avoir codé ensemble un beau "love calculator" à base de nombreuses variables aléatoires.

## Lois :

- **L'âge** : loi uniforme discrète entre 0 et 100 (notés a et b).

La loi uniforme discrète est utilisée pour les variables aléatoires prenant des valeurs entières et équiprobables dans un intervalle donné.

<i>distribution</i>	<i>loi de probabilité</i>	$\mathbb{E}(X)$	$\text{var}(X)$	<i>fonction caract.</i> $\mathbb{E}(e^{itX})$
Uniforme $\mathcal{U}(a, b)$	$\frac{1}{b-a} \mathbb{1}_{[a,b]}(x)$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$	$\frac{e^{ibt} - e^{iat}}{i(b-a)t}$

- **Le nombre de frères et sœurs** : loi de Poisson avec une moyenne de 3.

La loi de Poisson est utilisée pour modéliser des événements rares, avec une moyenne donnée. Comme la plupart des gens ont moins de 15 frères et sœurs, la moyenne de 3 semble raisonnable pour modéliser le nombre de frères et sœurs.

Poisson $\mathcal{P}(\lambda)$	$\mathbb{P}(X = k) = e^{-\lambda} \frac{\lambda^k}{k!}$ $k = 0, 1, \dots$	$\lambda$	$\lambda$	$e^{\lambda(z-1)}$
--------------------------------	--	-----------	-----------	--------------------

- **Le nombre d'animaux** : loi de Poisson avec une moyenne de 1.

La justification de l'utilisation de cette loi est exactement la même que pour les frères et sœurs. Néanmoins ici une moyenne de 2 est plus appropriée.

- **Le salaire** : loi exponentielle avec un paramètre de 0,0001.

La loi exponentielle est utilisée pour modéliser des temps d'attente entre des événements qui se produisent de manière aléatoire. Un paramètre de 0,0001 (noté  $\lambda$ ) est choisi ici pour créer une distribution qui a une queue à droite mais modérée, ce qui est cohérent avec la distribution des salaires dans la plupart des populations, pour éviter d'avoir trop de salaire bas (en dessous de 1000) car ça ne représente pas vraiment la population. On a également encadré les salaires entre 200 et 20 000 pour ne pas avoir de trop gros écarts.

Exponentielle $\mathcal{E}(\lambda)$	$\lambda e^{-\lambda x} \mathbb{1}_{\mathbb{R}^+}(x)$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$	$\frac{\lambda}{\lambda - it}$
--------------------------------------	---	---------------------	-----------------------	--------------------------------

- **Fumeur ou non fumeur** : variable de Bernoulli avec une probabilité de 0,3 de fumer.

La variable de Bernoulli est utilisée pour modéliser une expérience aléatoire avec deux résultats possibles, avec une probabilité donnée pour l'un des résultats. La probabilité 0,3 de fumer est choisie de manière arbitraire, c'est une estimation moyenne de la population.

<i>distribution</i>	<i>loi de probabilité</i>	$\mathbb{E}(X)$	$\text{var}(X)$	<i>fonction génératrice <math>\mathbb{E}(z^X)</math></i>
Bernoulli	$\mathbb{P}(X = 0) = q, \mathbb{P}(X = 1) = p$ $q = 1 - p$	$p$	$pq$	$pz + q$

- **Le taux de fidélité** : loi normale avec une moyenne de 60 et un écart type de 20.

La loi normale est utilisée pour modéliser des variables aléatoires continues avec une distribution symétrique en forme de cloche. Une loi normale avec une moyenne de 70 (notée  $m$ ) et un écart type de 10 (noté  $\sigma$ ) est choisie de manière arbitraire pour créer une distribution qui a une plage suffisamment grande pour inclure des valeurs élevées et basses, mais qui est centrée autour d'une valeur moyenne typique.

Normale $\mathcal{N}(m, \sigma^2)$	$\frac{1}{\sqrt{2\pi} \sigma} \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right)$	$m$	$\sigma^2$	$e^{imt - \frac{1}{2}\sigma^2 t^2}$
------------------------------------	--	-----	------------	-------------------------------------

- **La beauté** : loi gamma avec une forme de 10 et une échelle de 5.

La loi gamma est utilisée pour modéliser des temps d'attente entre des événements qui se produisent de manière aléatoire, avec une distribution de probabilité plus générale que la loi exponentielle. Une loi gamma avec une forme de 10 (notée  $a$ ) et une échelle de 5 (notée  $\lambda$ ) est choisie de manière arbitraire pour créer une distribution qui est toujours positive et a une longue queue à droite. La forme et l'échelle sont choisis pour créer une distribution qui a une variabilité suffisante pour inclure des valeurs élevées et basses, mais qui est centrée autour d'une valeur moyenne typique.

Gamma $\Gamma(a, \lambda)$	$\frac{\lambda^a}{\Gamma(a)} x^{a-1} e^{-\lambda x} \mathbb{1}_{[0, +\infty[}(x)$	$\frac{a}{\lambda}$	$\frac{a}{\lambda^2}$	$\left(\frac{\lambda}{\lambda - it}\right)^a$
----------------------------	---	---------------------	-----------------------	---

### Vocabulaire :

Une queue à droite (ou une queue positive) est une caractéristique d'une distribution de probabilité dans laquelle les valeurs extrêmes supérieures à la moyenne ont une probabilité relativement élevée. Par exemple, si une distribution a une queue à droite, cela signifie qu'il y a une probabilité plus élevée que des valeurs extrêmes élevées apparaissent dans les données. Une distribution avec une queue à droite peut être utile pour modéliser des phénomènes où il y a des événements rares mais importants qui peuvent avoir un impact significatif. En revanche, une distribution avec une queue à gauche (ou une queue négative) est caractérisée par une probabilité plus élevée pour les valeurs extrêmes inférieures à la moyenne.

Une fois qu'on a les 7 valeurs finales de la fiche de compatibilité tirée des deux fiches d'identité on fait la moyenne de toutes ces valeurs avec une certaine pondération pour trouver le pourcentage de compatibilité finale. Plus le pourcentage sera grand plus le cœur grossira.

### Code :

```
// Fonction pour générer une variable aléatoire suivant une loi uniforme discrète entre a et b
function generateUniformDiscrete(a, b) {
  return Math.floor(Math.random() * (b - a + 1)) + a;
}

// Fonction pour générer une variable aléatoire suivant une loi de Poisson avec une moyenne lambda
function generatePoisson(lambda) {
  var L = Math.exp(-lambda);
  var k = 0;
  var p = 1;
  while (p > L) {
    k++;
    p *= Math.random();
  }
  return k - 1;
}

// Fonction pour générer une variable aléatoire suivant une loi exponentielle avec un paramètre lambda
function generateExponential(lambda) {
  return -Math.log(1 - Math.random()) / lambda;
}

// Fonction pour générer une variable aléatoire suivant une variable de Bernoulli avec une probabilité p de succès
```

```

function generateBernoulli(p) {
    return Math.random() < p ? 1 : 0;
}

// Fonction pour générer une variable aléatoire suivant une loi normale avec une moyenne mu et un écart-type sigma
function generateNormal(mu, sigma) {
    var u1 = Math.random();
    var u2 = Math.random();
    var z = Math.sqrt(-2 * Math.log(u1)) * Math.cos(2 * Math.PI * u2);
    return mu + sigma * z;
}

// Fonction pour générer une variable aléatoire suivant une loi gamma avec une forme k et une échelle theta
function generateGamma(k, theta) {
    if (k < 1) {
        return generateGamma(1 + k, theta) * Math.pow(Math.random(), 1 / k);
    } else {
        var d = k - 1 / 3;
        var c = 1 / Math.sqrt(9 * d);
        while (true) {
            var x = generateNormal(0, 1);
            var v = Math.pow(1 + c * x, 3);
            if (v > 0) {
                var u = Math.random();
                if (Math.log(u) < 0.5 * x ** 2 + d - d * v + d * Math.log(v)) {
                    return d * v * theta;
                }
            }
        }
    }
}

```

Explications :

- `generateUniformDiscrete(a, b)` : Cette fonction utilise `Math.random()` pour générer un nombre aléatoire entre 0 (inclus) et 1 (exclus). En multipliant ce nombre par la différence entre `b` et `a` et en ajoutant `a`, on obtient un nombre aléatoire dans l'intervalle `[a, b)`. En utilisant `Math.floor()`, on arrondit ce nombre à l'entier inférieur pour obtenir une valeur entière aléatoire dans l'intervalle `[a, b]`.
- `generatePoisson(lambda)` : Cette fonction génère une variable aléatoire suivant la loi de Poisson en utilisant l'algorithme de la méthode des événements rares. Elle utilise `Math.random()` pour générer des nombres aléatoires entre 0 et 1 successivement jusqu'à ce que la probabilité calculée `p` devienne inférieure à la valeur limite `L = exp(-lambda)`. À chaque itération, on incrémente `k` et multiplie `p` par un nouveau nombre aléatoire entre 0 et 1. Enfin, on retourne `k - 1` pour obtenir la valeur aléatoire suivant la loi de Poisson.
- `generateExponential(lambda)` : Cette fonction génère une variable aléatoire suivant la loi exponentielle en utilisant l'inverse de la fonction de répartition. On génère un nombre aléatoire `u` entre 0 (inclus) et 1 (exclus) à l'aide de `Math.random()`, puis on applique la formule `x = -ln(1 - u) / lambda` pour obtenir la valeur aléatoire suivant la loi exponentielle.
- `generateBernoulli(p)` : Cette fonction génère une variable aléatoire suivant une variable de Bernoulli, qui est une variable aléatoire binaire avec une probabilité de succès `p`. La fonction utilise `Math.random()` pour générer un nombre aléatoire entre 0 et 1. Si ce nombre est inférieur à `p`, la fonction retourne 1 (succès), sinon elle retourne 0 (échec).

- `generateNormal(mu, sigma)` : Cette fonction génère une variable aléatoire suivant une loi normale (ou loi gaussienne) avec une moyenne `mu` et un écart-type `sigma`. Elle utilise la méthode de transformation de Box-Muller pour générer deux nombres aléatoires uniformément distribués entre 0 et 1. En appliquant ces nombres à une formule mathématique, elle obtient un nombre aléatoire qui suit une distribution normale avec la moyenne et l'écart-type spécifiés.
- `generateGamma(k, theta)` : Cette fonction génère une variable aléatoire suivant une loi gamma avec une forme `k` et une échelle `theta`. Elle utilise l'algorithme de Marsaglia et Tsang pour générer une variable aléatoire suivant une distribution gamma en utilisant une combinaison de variables aléatoires suivant une distribution normale standard et une distribution exponentielle. Cette méthode garantit une génération précise de la variable aléatoire gamma pour différentes valeurs de `k` et `theta`.

## Vocabulaire :

**Algorithme de Marsaglia et Tsang** : L'idée principale de l'algorithme est de générer des paires de nombres pseudo-aléatoires, puis de les transformer en un seul nombre réel dans l'intervalle  $[0, 1]$ . Cela est réalisé en utilisant la transformation de Box-Muller pour obtenir des variables aléatoires normalement distribuées, puis en combinant ces variables pour obtenir un nombre uniformément distribué dans  $[0, 1]$ .

L'algorithme de Marsaglia et Tsang a été conçu pour produire des séquences de nombres pseudo-aléatoires avec de bonnes propriétés statistiques, telles qu'une distribution uniforme et une indépendance entre les nombres générés.

**La méthode de transformation de Box-Muller** : Elle permet de générer des variables normalement distribuées à partir de variables uniformément distribuées en utilisant des fonctions mathématiques simples. Cependant, il est important de noter que cette méthode génère des paires de variables normales indépendantes.

**La méthode des événements rares** : La méthode des événements rares est une technique utilisée pour estimer la probabilité d'occurrence d'événements rares, c'est-à-dire des événements qui se produisent avec une faible probabilité dans un système donné. Cette méthode est particulièrement utile lorsque les méthodes analytiques traditionnelles deviennent difficiles ou impossibles à appliquer en raison de la complexité du système ou de l'événement étudié. Cependant, il est important de noter que la méthode des événements rares repose sur des hypothèses et des approximations, et que la précision des résultats dépend de la qualité du modèle et du nombre d'échantillons utilisés.

## Appels des fonctions:

Chaque variable représente un membre de la fiche identité de chaque prénom. Chaque fonction est appelée 2 fois pour générer 2 variables aléatoires pour les deux fiches identité.

exemple :

```
var age1, age2;
age1 = generate_uniformDiscrete(0,99);
age2 = generate_uniformDiscrete(0,99);
```

## Fonctions de la Fiche Compatibilité :

**calculerPourcentage\_final** : Cette fonction calcule le pourcentage final de compatibilité en utilisant différentes pondérations pour chaque aspect de la relation. Elle prend en compte les valeurs retournées par les autres fonctions pour calculer le pourcentage final. Elle limite le résultat entre 0 et 100 et multiplie le pourcentage final par un nombre aléatoire entre 0 et 3.

**calculerDureeRelation** : Cette fonction calcule la durée de la relation en années en prenant en compte la différence d'âge entre les deux partenaires. Elle limite la durée à un maximum de 60 ans. Le résultat final est divisé par 5 pour plus de crédibilité.

**calculerPremiereDispute** : Cette fonction calcule le temps d'apparition de la première dispute en semaines en utilisant les salaires des partenaires. Elle estime l'écart type des salaires et génère un nombre aléatoire entre un minimum et un maximum de semaines, en utilisant la fonction `Math.random()`. Le résultat est arrondi à l'entier inférieur ou égal et divisé par 100.

**calculerTempsReconciliation** : Cette fonction calcule le temps de réconciliation entre deux disputes en heures en utilisant des valeurs relatives à la famille. Elle estime l'écart type des valeurs familiales et génère un nombre aléatoire entre un minimum et un maximum d'heures, en utilisant la fonction `Math.random()`. Le résultat est arrondi à l'entier le plus proche.

**calculerCompatibiliteSexuelle** : Cette fonction calcule la compatibilité sexuelle en pourcentage en utilisant les valeurs de fumeur et de fidélité des partenaires. Elle calcule la moyenne des valeurs de fumeur et de fidélité, puis multiplie chaque moyenne par un poids spécifique. Le résultat est limité entre 0 et 100.

**calculerCompatibilitePhysique** : Cette fonction calcule la compatibilité physique en pourcentage en utilisant les valeurs de beauté des partenaires. Elle calcule la moyenne des valeurs de beauté, puis la multiplie par un poids spécifique. Le résultat est limité entre 0 et 100.

**calculerDureeVieCommune** : Cette fonction calcule la durée de vie commune en années en utilisant le nombre d'animaux de chaque partenaire. Elle calcule la moyenne du nombre d'animaux et le multiplie par la durée de vie moyenne d'un animal. Le résultat est limité entre 0 et 60. Le résultat final est divisé par 5 pour plus de crédibilité.

## L'influence de l'utilisateur :

L'utilisateur est invité à renseigner son sexe après avoir renseigné son prénom. Cette information est ensuite gérée par la fonction `bonusPercentage()` qui se charge de calculer un entier relativement petit à ajouter au pourcentage final renvoyé par `calculerPourcentage_final`.

Si l'utilisateur indique être une femme, le bonus pour le prénom est calculé à partir d'une loi uniforme discrète à valeurs dans  $[0,10]$ .



Si l'utilisateur indique être un homme, le bonus pour le prénom est calculé à partir d'une loi de poisson de moyenne 5.

Si l'utilisateur indique être non-binaire, le bonus pour le prénom est calculé à partir d'une loi de Bernoulli de paramètre 0.7.

Les bonus de chaque prénoms sont ensuite additionnés et ajoutés au pourcentage final.

Le choix de ces lois est très arbitraire. Elles permettent juste de donner la main à l'utilisateur quant au bonus qu'il peut recevoir sur sa note. Ce sont des lois qui permettent d'obtenir des valeurs relativement petites qui n'auront pas une énorme influence sur le score final.

## **Un point sur le site en lui-même :**

Qui dit jeu niais dit direction artistique niaise... Nous n'avons pas dérogé à la règle et avons codé un site de "love test" en bonne et due forme : du rose et des cœurs.

Nous avons fait en sorte que le bouton "faire le test" ne fonctionne que lorsque deux prénoms ont bien été rentrés dans les champs de saisie.

Le cœur entre les deux prénoms change de taille en fonction du pourcentage de compatibilité : plus le pourcentage est élevé, plus le cœur est gros.

Un bouton "recommencer le test" permet de recharger la page.