ESCENAS DEL AZAR

VARIABLES ALEATORIAS

« You can define randomness as something that cannot be compressed at all. It's an object with the property that basically the only way you can describe it to someone is to say “this is it” and show it to them. It has no structure or pattern, there is no concise description...it' s irreducible »

Gregory J. Chaitin

VARIABLES ALEATORIAS

VARIABLES ALEATORIAS

- Distribución de probabilidad

VARIABLES ALEATORIAS

- Distribución de probabilidad
- Función de densidad

VARIABLES ALEATORIAS

- Distribución de probabilidad
- Función de densidad
- Rango y eventos de probabilidad 0

VARIABLES ALEATORIAS

- Distribución de probabilidad
- Función de densidad
- Rango y eventos de probabilidad 0
- Parámetros

DISTRIBUCIÓN UNIFORME

DISTRIBUCIÓN UNIFORME

- Parámetros: extremos

DISTRIBUCIÓN UNIFORME

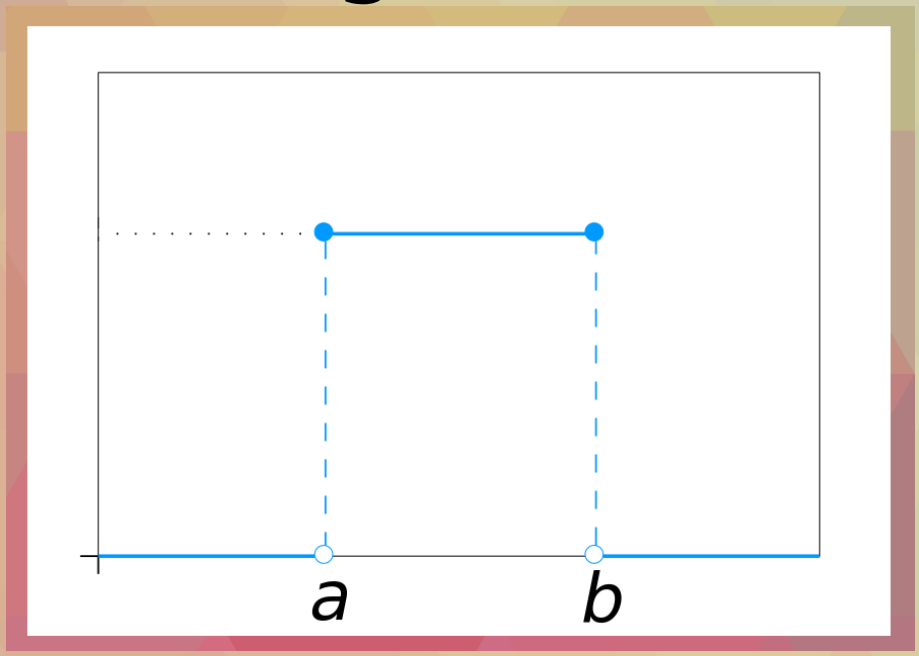
- Parámetros: extremos
- Igual probabilidad dentro del rango

DISTRIBUCIÓN UNIFORME

- Parámetros: extremos
- Igual probabilidad dentro del rango
- En Processing: `random(a, b)`

DISTRIBUCIÓN UNIFORME

- Parámetros: extremos
- Igual probabilidad dentro del rango
- En Processing: `random(a, b)`



DISTRIBUCIÓN UNIFORME

Uniforme escalonada

DISTRIBUCIÓN UNIFORME

Uniforme escalonada

- Partición del rango

DISTRIBUCIÓN UNIFORME

Uniforme escalonada

- Partición del rango
- Asignación de pesos

DISTRIBUCIÓN UNIFORME

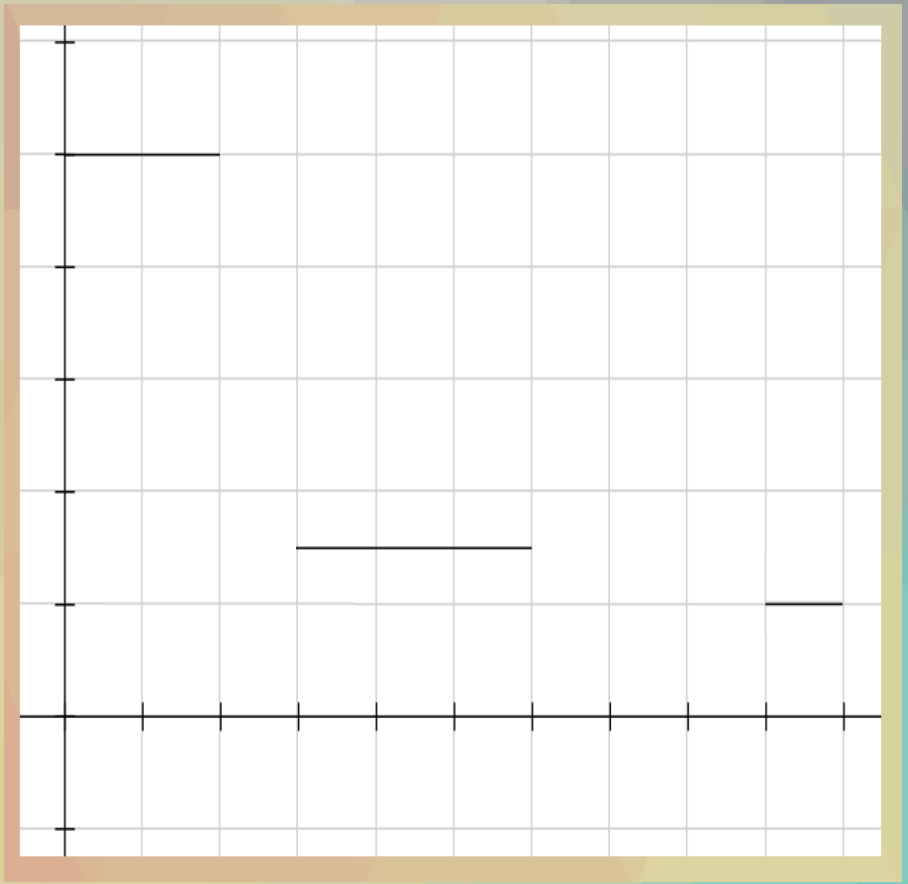
Uniforme escalonada

- Partición del rango
- Asignación de pesos
- La distribución de la variable en a cada región es uniforme

DISTRIBUCIÓN UNIFORME

Uniforme escalonada

- Partición del rango
- Asignación de pesos
- La distribución de la variable en a cada región es uniforme



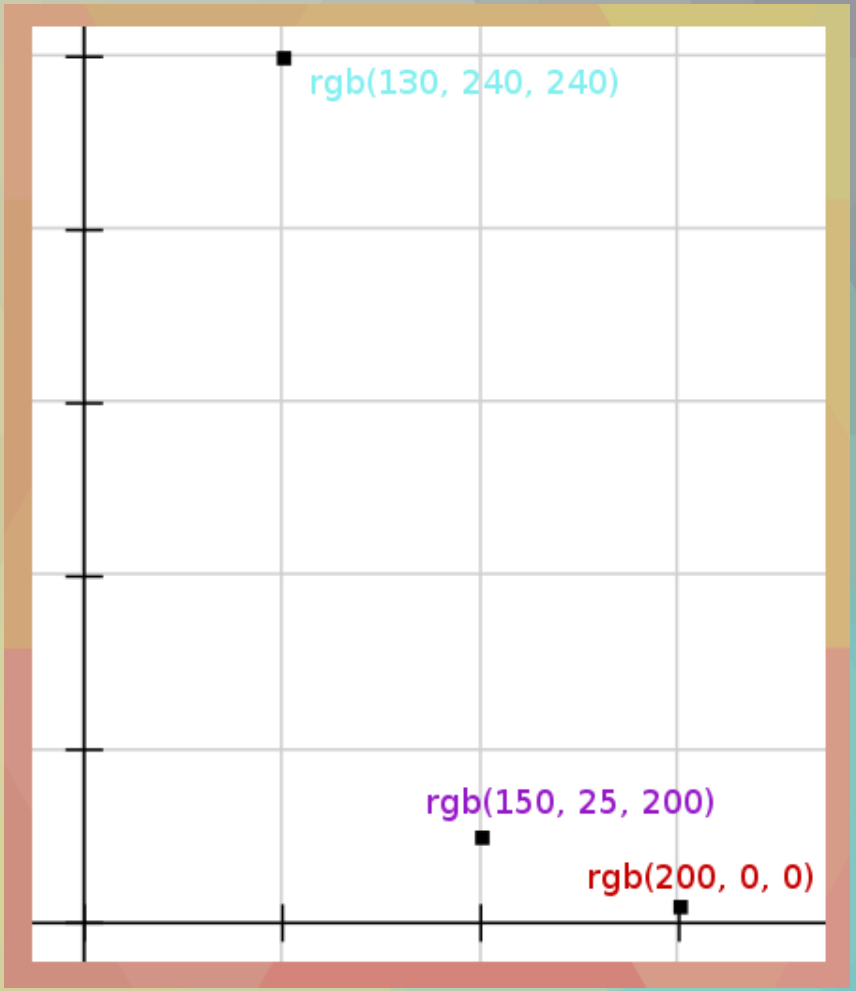
DISTRIBUCIÓN UNIFORME

Uniforme escalonada

```
float randomArea(float[][] intervals, float[] weights) {
  float totalSum = 0;
  for (int i=0; i<weights.length; i++) {
    totalSum += weights[i] * (intervals[i][1] - intervals[i][0]);
  }
  float r = random(totalSum);
  float partialSum = 0;
  for (int i=0; i<weights.length; i++) {
    partialSum += weights[i] * (intervals[i][1] - intervals[i][0]);
    if (r < partialSum) {
      return random(intervals[i][0], intervals[i][1]);
    }
  }
  return random(intervals[intervals.length-1][0], intervals[intervals.length-1][1]);
}
```

DISTRIBUCIÓN UNIFORME

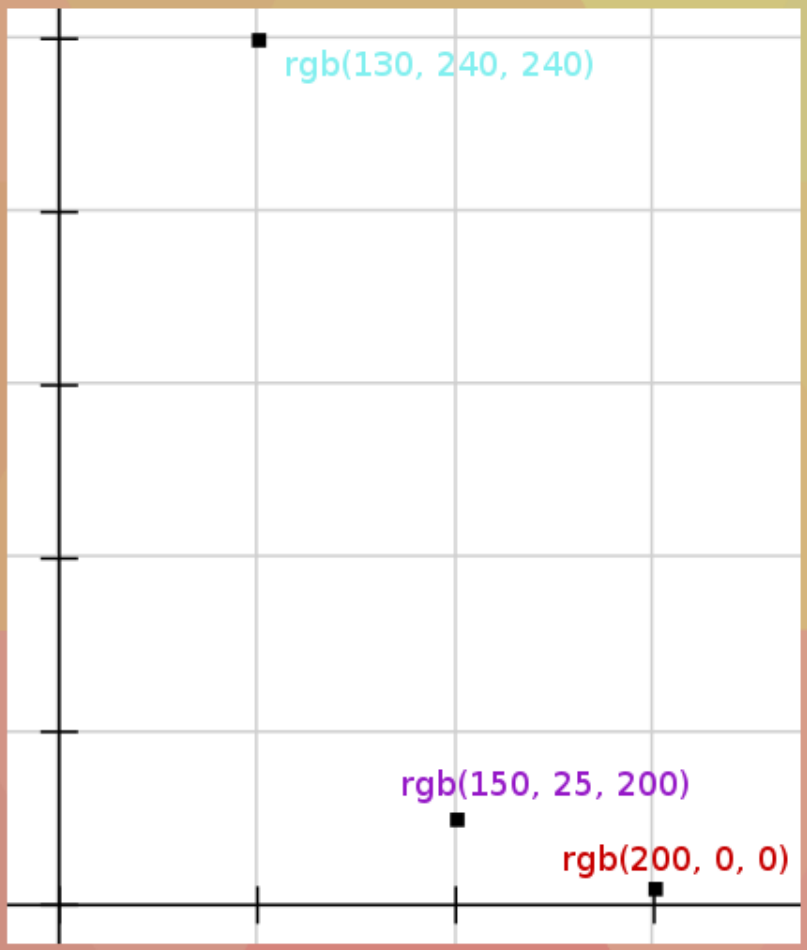
Uniforme escalonada discreta



DISTRIBUCIÓN UNIFORME

Uniforme escalonada discreta

```
color randomColor() {
  float totalSum = 0;
  for (int i=0; i<weights.length; i++) {
    totalSum += weights[i];
  }
  float r = random(totalSum);
  float partialSum = 0;
  for (int i=0; i<weights.length; i++) {
    partialSum += weights[i];
    if (r < partialSum) {
      return colors[i];
    }
  }
  return colors[colors.length-1];
}
```



DISTRIBUCIÓN UNIFORME

Composición de uniformes

DISTRIBUCIÓN UNIFORME

Composición de uniformes

- `random(random(1))` ?

DISTRIBUCIÓN UNIFORME

Composición de uniformes

- `random(random(1))` ?

  Es uniforme ?

DISTRIBUCIÓN UNIFORME

Composición de uniformes

- `random(random(1))` ?

  Es uniforme ?

  Valor más probable ?

DISTRIBUCIÓN UNIFORME

Composición de uniformes

- `random(random(1))` ?

  Es uniforme ?

  Valor más probable ?
  Valor menos probable ?

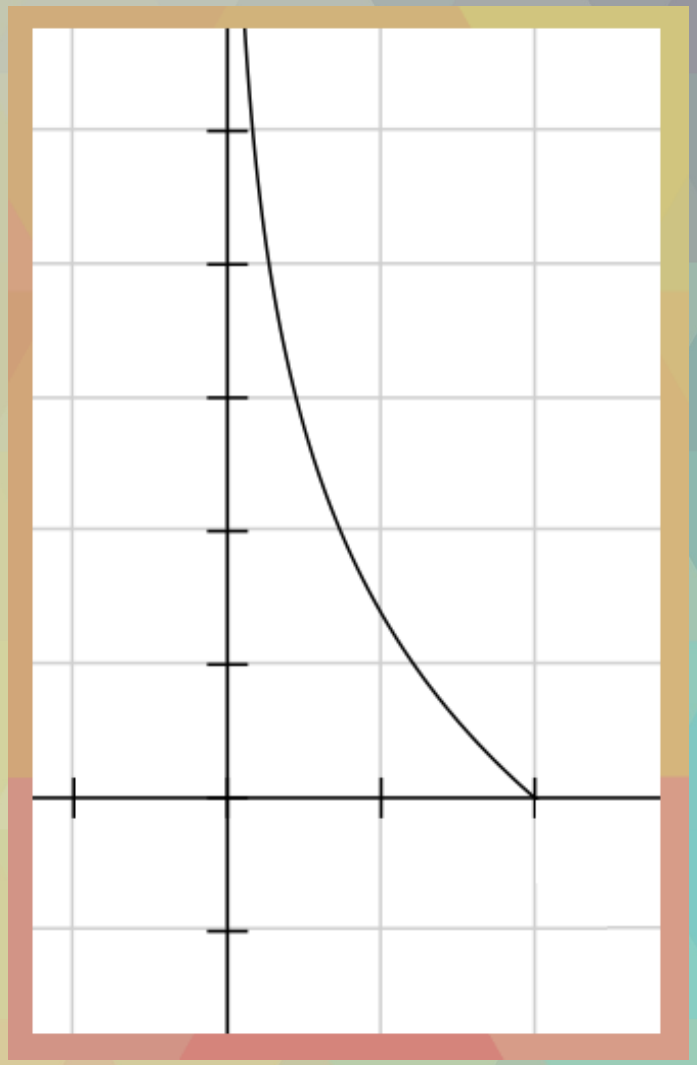
DISTRIBUCIÓN UNIFORME

Composición de uniformes

- `random(random(1))` ?

  Es uniforme ?

  Valor más probable ?
  Valor menos probable ?



DISTRIBUCIÓN UNIFORME

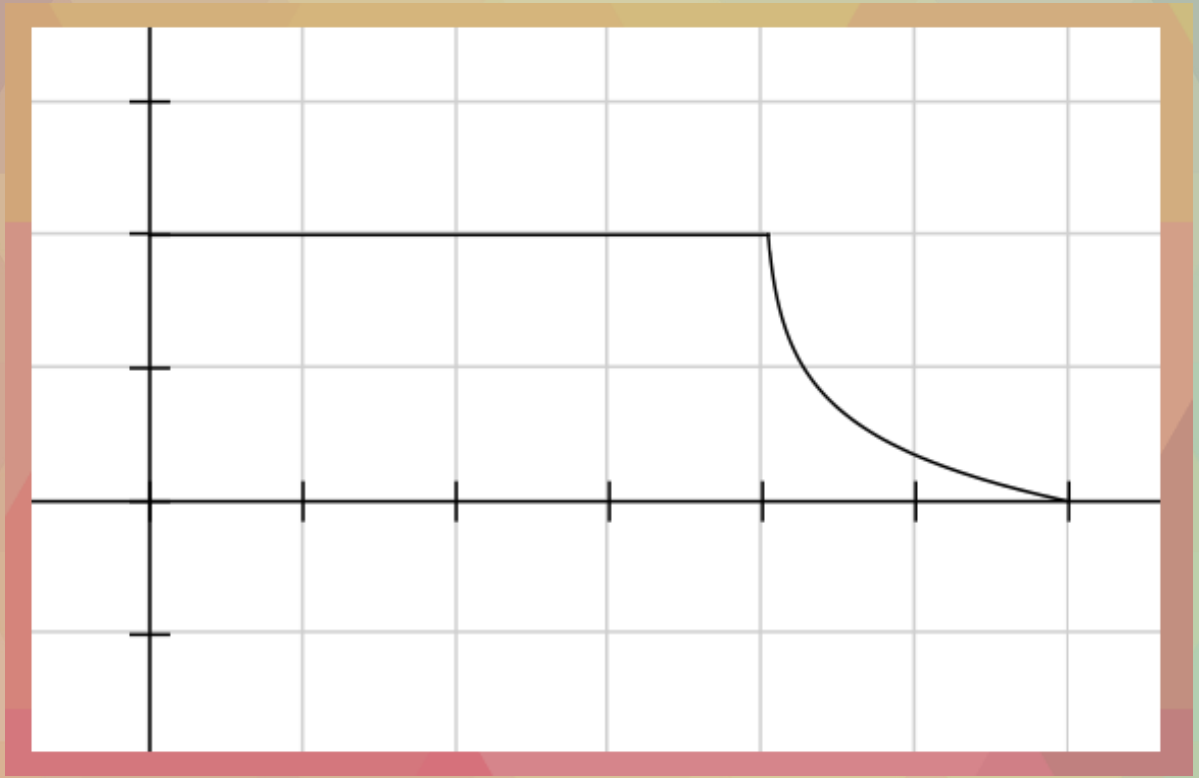
Composición de uniformes

- `random(random(4,6))` ?

DISTRIBUCIÓN UNIFORME

Composición de uniformes

- `random(random(4,6))` ?



DISTRIBUCIÓN UNIFORME

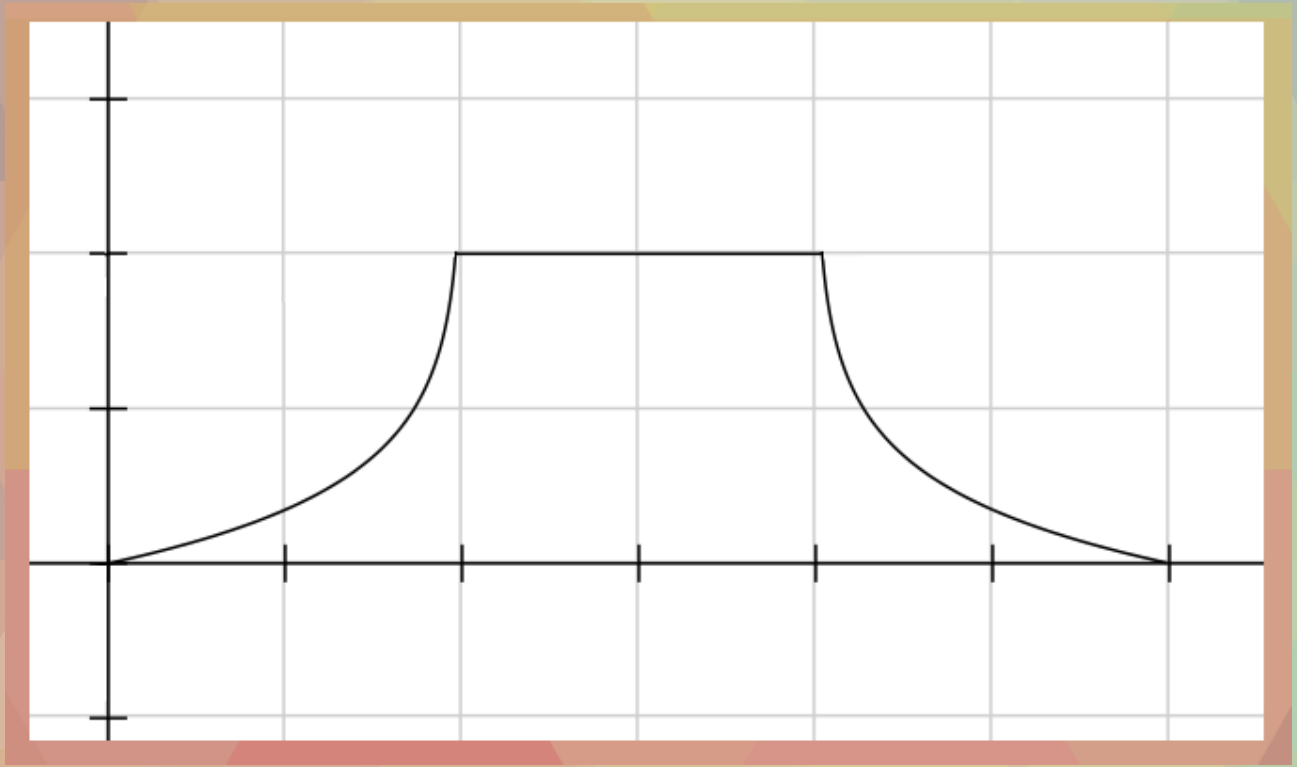
Composición de uniformes

- `random(random(0,2),random(4,6))` ?

DISTRIBUCIÓN UNIFORME

Composición de uniformes

- `random(random(0,2),random(4,6))` ?



DISTRIBUCIÓN NORMAL

DISTRIBUCIÓN NORMAL

- Parámetros: media μ y varianza σ

DISTRIBUCIÓN NORMAL

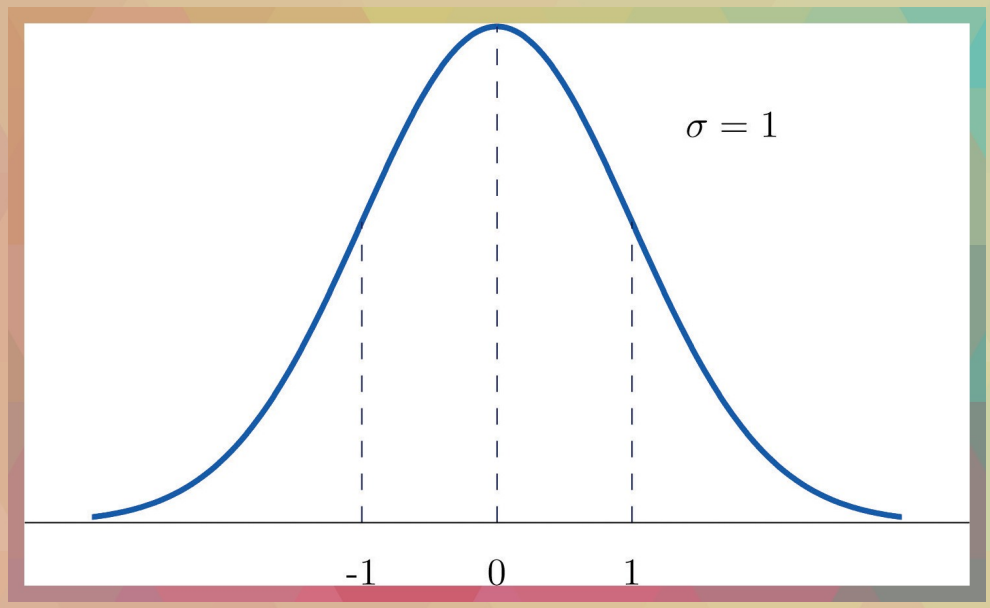
- Parámetros: media μ y varianza σ
- Mayor probabilidad en la media, disminuyendo al alejarse de ésta

DISTRIBUCIÓN NORMAL

- Parámetros: media μ y varianza σ
- Mayor probabilidad en la media, disminuyendo al alejarse de ésta
- En Processing: `randomGuassian()`

DISTRIBUCIÓN NORMAL

- Parámetros: media μ y varianza σ
- Mayor probabilidad en la media, disminuyendo al alejarse de ésta
- En Processing: `randomGuassian()`



DISTRIBUCIÓN NORMAL

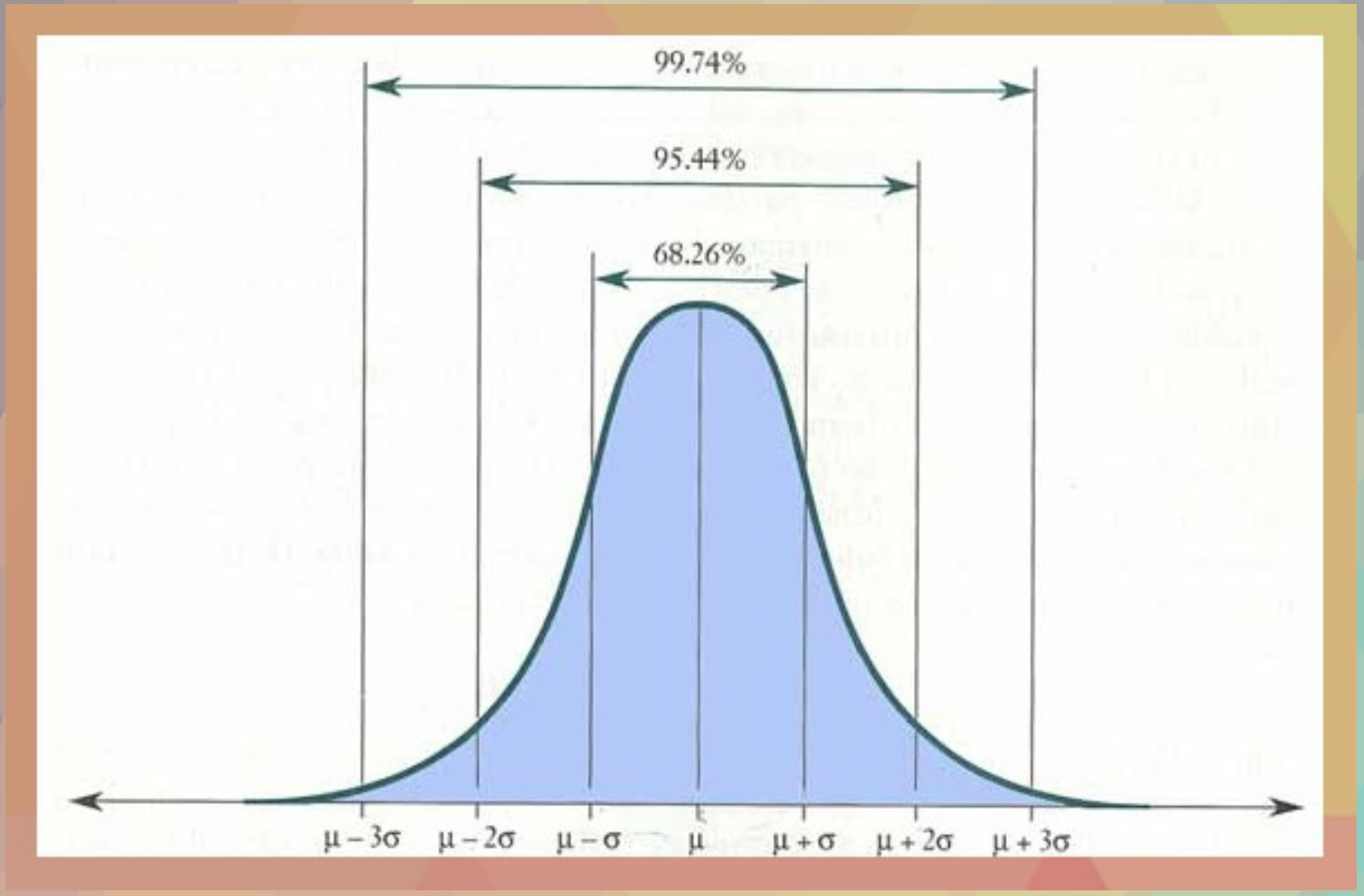
- randomGuassian() es la función con distribución normal estandar, con $\mu = 0$ y $\sigma = 1$

DISTRIBUCIÓN NORMAL

- `randomGuassian()` es la función con distribución normal estandar, con $\mu = 0$ y $\sigma = 1$

- `randomGaussian()*d+m` es la normal con $\mu = m$ y $\sigma = d$

DISTRIBUCIÓN NORMAL



RUIDO (PERLIN)

RUIDO (PERLIN)

- No simula en cada llamado una variable con distribución conocida.

RUIDO (PERLIN)

- No simula en cada llamado una variable con distribución conocida.
- Es una función con variaciones azarosas y sucesivos llamados con los mismos argumentos devuelven los mismos valores

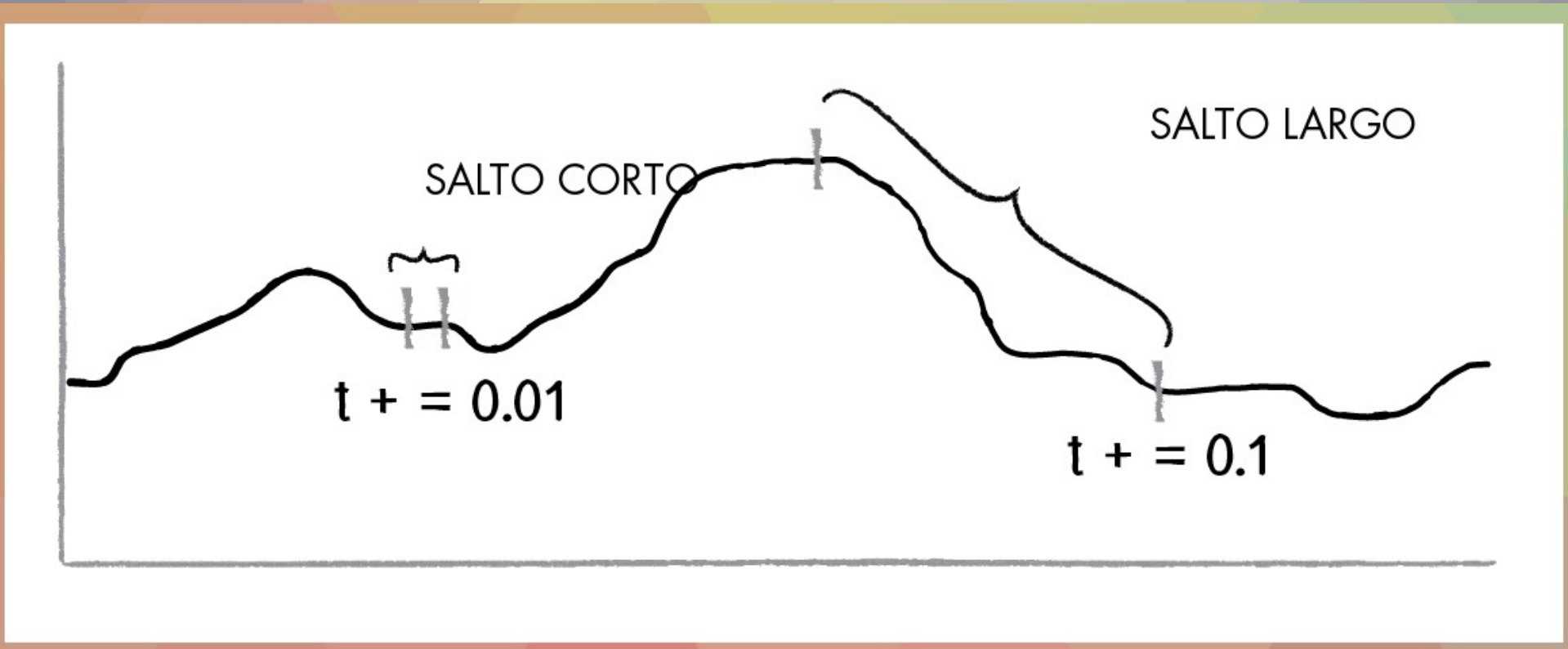
RUIDO (PERLIN)

- No simula en cada llamado una variable con distribución conocida.
- Es una función con variaciones azarosas y sucesivos llamados con los mismos argumentos devuelven los mismos valores
- Permite tener un sentido de *continuidad:* pequeñas variaciones en los argumentos generan pequeñas variaciones en los resultados

RUIDO (PERLIN)

- No simula en cada llamado una variable con distribución conocida.
- Es una función con variaciones azarosas y sucesivos llamados con los mismos argumentos devuelven los mismos valores
- Permite tener un sentido de *continuidad:* pequeñas variaciones en los argumentos generan pequeñas variaciones en los resultados
- En Processing es una función `noise()` que toma hasta 3 argumentos

RUIDO



SEMILLAS

SEMILLAS

- Es un numero usado para inicializar el generador de numeros pseudo-aleatorio. Su especificación es opcional

SEMILLAS

- Es un numero usado para inicializar el generador de numeros pseudo-aleatorio. Su especificación es opcional
- Una vez fija la semilla, la secuencia de numeros devueltos por `random()` y `randomGaussian()` seran los mismos

SEMILLAS

- Es un numero usado para inicializar el generador de numeros pseudo-aleatorio. Su especificación es opcional
- Una vez fija la semilla, la secuencia de numeros devueltos por `random()` y `randomGaussian()` seran los mismos
- En Processing, las semillas se especifican mediante `randomSeed()` y `noiseSeed()`