# Simulación de modelos evolutivos en tumores con R y C++

Alberto Parramón Castillo

**Universidad Autónoma de Madrid**

***Trabajo de Fin de Grado***

5 de julio de 2016

# Índice de contenidos

# Índice de contenidos

# Índice de contenidos

# Índice de contenidos

# Índice de contenidos

# Índice de contenidos

Un proceso de modelización matemática consiste en:

1. **Identificación de un problema** o situación compleja que necesita ser simulada, optimizada o controlada.

2. **Elección del tipo de modelo que se va a usar**: para ello hay que saber qué tipo de respuesta se quiere obtener y cuáles son los factores de los que depende el modelo.

3. **Formalización del modelo**: detallar cuáles serán los datos de entrada, así como las fórmulas matemáticas en las que se basará. En esta fase posiblemente se introduzcan simplificaciones necesarias para que el problema matemático sea tratable computacionalmente, por ejemplo, mediante la implementación de un algoritmo.

4. **Análisis de los resultados**: análisis del modelo a partir de los resultados y obtención de información de los mismos comparándolos con situaciones o datos reales.

1. **Genotipo**: conjunto de todos los genes que tienen las células.

1. **Genotipo**: conjunto de todos los genes que tienen las células.

2. **Fitness**: es un número que se le asocia al genotipo y denota la capacidad de resistencia y de reproducción que tiene una célula.

1. **Genotipo**: conjunto de todos los genes que tienen las células.

2. **Fitness**: es un número que se le asocia al genotipo y denota la capacidad de resistencia y de reproducción que tiene una célula.

3. **Mutaciones conductoras (driver mutations)**: son cambios genéticos que están muy relacionados con el desarrollo del cáncer, porque producen un aumento del fitness.

1. **Genotipo**: conjunto de todos los genes que tienen las células.

2. **Fitness**: es un número que se le asocia al genotipo y denota la capacidad de resistencia y de reproducción que tiene una célula.

3. **Mutaciones conductoras (driver mutations)**: son cambios genéticos que están muy relacionados con el desarrollo del cáncer, porque producen un aumento del fitness.

4. **Mutaciones pasajeras (passenger mutations)**: son cambios genéticos que no están directamente relacionados con el desarrollo del cáncer ya que no producen aumento del fitness ni ventaja evolutiva.

1. **Genotipo**: conjunto de todos los genes que tienen las células.

2. **Fitness**: es un número que se le asocia al genotipo y denota la capacidad de resistencia y de reproducción que tiene una célula.

3. **Mutaciones conductoras (driver mutations)**: son cambios genéticos que están muy relacionados con el desarrollo del cáncer, porque producen un aumento del fitness.

4. **Mutaciones pasajeras (passenger mutations)**: son cambios genéticos que no están directamente relacionados con el desarrollo del cáncer ya que no producen aumento del fitness ni ventaja evolutiva.

5. **Fijación**: es la expansión de un genotipo en toda la población de células. Es decir, una mutación o genotipo se ha fijado cuando su frecuencia en la población es 1.

1. **Genotipo**: conjunto de todos los genes que tienen las células.
2. **Fitness**: es un número que se le asocia al genotipo y denota la capacidad de resistencia y de reproducción que tiene una célula.
3. **Mutaciones conductoras (driver mutations)**: son cambios genéticos que están muy relacionados con el desarrollo del cáncer, porque producen un aumento del fitness.
4. **Mutaciones pasajeras (passenger mutations)**: son cambios genéticos que no están directamente relacionados con el desarrollo del cáncer ya que no producen aumento del fitness ni ventaja evolutiva.
5. **Fijación**: es la expansión de un genotipo en toda la población de células. Es decir, una mutación o genotipo se ha fijado cuando su frecuencia en la población es 1.
6. **Eventos celulares**: mutación, reproducción, muerte...

1. **Genotipo**: conjunto de todos los genes que tienen las células.
2. **Fitness**: es un número que se le asocia al genotipo y denota la capacidad de resistencia y de reproducción que tiene una célula.
3. **Mutaciones conductoras (driver mutations)**: son cambios genéticos que están muy relacionados con el desarrollo del cáncer, porque producen un aumento del fitness.
4. **Mutaciones pasajeras (passenger mutations)**: son cambios genéticos que no están directamente relacionados con el desarrollo del cáncer ya que no producen aumento del fitness ni ventaja evolutiva.
5. **Fijación**: es la expansión de un genotipo en toda la población de células. Es decir, una mutación o genotipo se ha fijado cuando su frecuencia en la población es 1.
6. **Eventos celulares**: mutación, reproducción, muerte...
7. **Clon**: conjunto de todos los individuos de la población que tienen el mismo genotipo

La base del modelo es predecir la aparición de los eventos celulares, es decir, mutación, reproducción y muerte:

1. ¿**Cuál** se produce?
2. ¿**Cuándo o cuántas veces** se produce dentro de un intervalo de tiempo?

La base del modelo es predecir la aparición de los eventos celulares, es decir, mutación, reproducción y muerte:

1. ¿**Cuál** se produce?

2. ¿**Cuándo o cuántas veces** se produce dentro de un intervalo de tiempo?

Dos tipos de modelos diferentes:

1. **Modelos en tiempo discreto**: cuando se obtienen resultados en intervalos discretos de tiempo.
   - Aplicado a este campo: en cada intervalo de tiempo se producen uno, ninguno, o varios eventos celulares.

2. **Modelos en tiempo continuo**: cuando se pueden obtener los valores de salida en todos los instantes de un intervalo de tiempo. Los modelos continuos suelen estar basados en ecuaciones diferenciales.
   - Aplicado a este campo: calcula el siguiente instante de tiempo en el que se va a producir algún evento celular.

En función de los factores de los que dependen los eventos de mutación, reproducción o muerte:

1. Modelos en que asumen que los eventos celulares son **independientes** en cada célula.
2. Modelos en los que los eventos celulares se influencian de **factores microambientales** y del entorno que les rodea.
3. Modelos **híbridos**.

En función de los factores de los que dependen los eventos de mutación, reproducción o muerte:

1. Modelos en que asumen que los eventos celulares son **independientes** en cada célula.
2. Modelos en los que los eventos celulares se influencian de **factores microambientales** y del entorno que les rodea.
3. Modelos **híbridos**.

En función del espacio dimensional en el que se trabaje:

1. **Modelos adimensionales**.
2. **Modelos espaciales**: Cada individuo, aparte de caracterizarse por su genotipo, tiene asociada una posición única, además suelen tener la posibilidad de moverse

En función de los factores de los que dependen los eventos de mutación, reproducción o muerte:

1. Modelos en que asumen que los eventos celulares son **independientes** en cada célula.
2. Modelos en los que los eventos celulares se influencian de **factores microambientales** y del entorno que les rodea.
3. Modelos **híbridos**.

En función del espacio dimensional en el que se trabaje:

1. **Modelos adimensionales**.
2. **Modelos espaciales**: Cada individuo, aparte de caracterizarse por su genotipo, tiene asociada una posición única, además suelen tener la posibilidad de moverse

En función de que se permita mutación simple o múltiple.

1. **Mutación simple**: una mutación por iteración. Habitual en los modelos en tiempo continuo
2. **Mutación múltiple**: se permite la mutación de varios genes por iteración. Habitual en modelos en tiempo discreto.

En función de cómo se consideran los eventos de reproducción y muerte en el algoritmo.

1. **Wright-Fisher model**: asume que todos los individuos de la población mueren en cada generación y son remplazados por una descendencia de exactamente los mismos individuos. El tamaño de la población se mantiene constante.

En función de cómo se consideran los eventos de reproducción y muerte en el algoritmo.

1. **Wright-Fisher model**: asume que todos los individuos de la población mueren en cada generación y son remplazados por una descendencia de exactamente los mismos individuos. El tamaño de la población se mantiene constante.

2. **Moran model**: se caracteriza porque en cada iteración un solo individuo es elegido al azar para reproducirse, y otro para morir. Por tanto la población no se mantiene constante.

En función de cómo se consideran los eventos de reproducción y muerte en el algoritmo.

1. **Wright-Fisher model**: asume que todos los individuos de la población mueren en cada generación y son remplazados por una descendencia de exactamente los mismos individuos. El tamaño de la población se mantiene constante.

2. **Moran model**: se caracteriza porque en cada iteración un solo individuo es elegido al azar para reproducirse, y otro para morir. Por tanto la población no se mantiene constante.

3. **Branching Process Model**: en cada iteración, cada individuo de la población tiene una probabilidad de reproducirse o de morir.
   - **Branching Process Model (clásico)**: dependiendo del azar, la población puede crecer indefinidamente o incluso decrecer hasta extinguirse.
   - **Logistic Branching Process Model**: añade un parámetro para regular el tamaño de la población, de manera que esta se mantenga cercana a la población inicial.

1. **Desarrollado por Ramón Díaz Uriarte**: Profesor Titular del Dpto. de Bioquímica de la Universidad Autónoma de Madrid.

2. **Pertenece al proyecto Bioconductor**: software para el análisis estadístico de datos de laboratorio en biología molecular.

3. **Escrito en R y C++**.

4. Contiene la implementación de un **modelo en tiempo continuo**.

5. Gran versatilidad y cantidad de opciones diferentes para **especificar cómo se calcula el fitness** asociado a un genotipo.

Dado un genotipo formado por 4 genes: $\{A, B, C\}$, el paquete OncoSimulR nos permite especificar el fitness de distintas maneras:

- **Genes que no interactúan**: Se asocia a cada gen un valor de forma independiente. Este valor se aplica cuando el gen está mutado:

Dado un genotipo formado por 4 genes: $\{A, B, C\}$, el paquete OncoSimulR nos permite especificar el fitness de distintas maneras:

- **Genes que no interactúan**: Se asocia a cada gen un valor de forma independiente. Este valor se aplica cuando el gen está mutado:

```
fe <- allFitnessEffects(noIntGenes = c("A" = -0.03, "B" = 0.05, "C" = 0.08))

evalAllGenotypes(fe, order = FALSE, addwt = TRUE)
##  Genotype  Fitness
##1       WT 1.00000
##2        A 0.97000
##3        B 1.05000
##4        C 1.08000
##5     A, B 1.01850
##6     A, C 1.04760
##7     B, C 1.13400
##8  A, B, C 1.09998
```

Dado un genotipo formado por 4 genes: $\{A, B, C\}$, el paquete OncoSimulR nos permite especificar el fitness de distintas maneras:

- **Genes que no interactúan**: Se asocia a cada gen un valor de forma independiente. Este valor se aplica cuando el gen está mutado:

```
fe <- allFitnessEffects(noIntGenes = c("A" = -0.03, "B" = 0.05, "C" = 0.08))

evalAllGenotypes(fe, order = FALSE, addwt = TRUE)
##  Genotype  Fitness
##1       WT 1.00000
##2        A 0.97000
##3        B 1.05000
##4        C 1.08000
##5     A, B 1.01850
##6     A, C 1.04760
##7     B, C 1.13400
##8  A, B, C 1.09998
```

- **Epistasis**: Las mutaciones de los genes no son independientes entre sí. Por ejemplo: $"C : -A : -B" = 0{,}5, "C : A" = -0{,}5$.

Dado un genotipo formado por 4 genes: $\{A, B, C\}$, el paquete OncoSimulR nos permite especificar el fitness de distintas maneras:

- **Genes que no interactúan**: Se asocia a cada gen un valor de forma independiente. Este valor se aplica cuando el gen está mutado:

```
fe <- allFitnessEffects(noIntGenes = c("A" = -0.03, "B" = 0.05, "C" = 0.08))

evalAllGenotypes(fe, order = FALSE, addwt = TRUE)
##  Genotype  Fitness
##1       WT 1.00000
##2        A 0.97000
##3        B 1.05000
##4        C 1.08000
##5     A, B 1.01850
##6     A, C 1.04760
##7     B, C 1.13400
##8  A, B, C 1.09998
```

- **Epistasis**: Las mutaciones de los genes no son independientes entre sí. Por ejemplo: $"C : -A : -B" = 0{,}5, "C : A" = -0{,}5$.
- **Efectos de orden**: Influye el orden en el que los genes se mutan. Por ejemplo: $"A > B" = 0{,}4, "B > A" = 0{,}1$.

Dado un genotipo formado por 4 genes: $\{A, B, C\}$, el paquete OncoSimulR nos permite especificar el fitness de distintas maneras:

- **Genes que no interactúan**: Se asocia a cada gen un valor de forma independiente. Este valor se aplica cuando el gen está mutado:

```
fe <- allFitnessEffects(noIntGenes = c("A" = -0.03, "B" = 0.05, "C" = 0.08))

evalAllGenotypes(fe, order = FALSE, addwt = TRUE)
##  Genotype  Fitness
##1       WT 1.00000
##2        A 0.97000
##3        B 1.05000
##4        C 1.08000
##5     A, B 1.01850
##6     A, C 1.04760
##7     B, C 1.13400
##8  A, B, C 1.09998
```

- **Epistasis**: Las mutaciones de los genes no son independientes entre sí. Por ejemplo: $"C : -A : -B" = 0{,}5, "C : A" = -0{,}5$.
- **Efectos de orden**: Influye el orden en el que los genes se mutan. Por ejemplo: $"A > B" = 0{,}4, "B > A" = 0{,}1$.
- En forma de **grafo**.

El modelo implementado tiene las siguientes características:

- Es un modelo en **tiempo discreto**.
- Es **adimensional**.
- Esta basado en el **Logistic Branching Process Model**. Los procesos de muerte y reproducción se realizan de la siguiente manera:
  1. Se recorren todos los clones.
  2. La descendencia de cada clon tendrá el mismo genotipo que el clon y dependerá de una distribución de Poisson de parámetro:

$$\lambda_i = C(x_i)n_i \quad \forall i = 1, ..., s$$

- Se permite **mutación múltiple**. Cada gen tiene una probabilidad de mutar. El proceso de mutación se realiza de la siguiente manera:
  1. Se recorren todas las células.
  2. Para cada una de ellas se generan tantos números aleatorios como genes tengan sin mutar. Los números aleatorios siguen una distribución $U(0, 1)$.
  3. Se mutan aquellos genes cuyo número aleatorio generado es igual o más pequeño que su probabilidad de mutación.

- `rFE`: especificación del fitness.
- `mu`: vector de probabilidades de mutación de cada gen en cada iteración.
- `popIni`: número de individuos que tendrá la población al principio del algoritmo.
- `tMax`: número de iteraciones máximas que se realizarán.
- `seed`: este parámetro es necesario para ciertos cálculos probabilísticos.
- `itInfo`: indica cada cuántas iteraciones desea el usuario que se le informe de cómo transcurre la simulación.
- `verbosity`: a mayor valor, más detallada será la información que el usuario reciba sobre el transcurso de la simulación.

Estructura en R con los siguientes campos:

- `TotalPopSize`: población final.
- `Mu`: vector de probabilidades utilizado.
- `Avefitness`: fitness medio de la población final.
- `NumClones`: número de clones en la población final.
- `PopSizeClones`: vector con los tamaños de población de los clones.
- `AbsfitnessClones`: vector con los fitness de cada uno de los clones.
- `GenotypeClones`: vector con los genotipos de los clones.
- `LargestClonePopSize`: tamaño de población del clon con mayor número de individuos.
- `LargestCloneFitness`: fitness del clon con mayor número de individuos.
- `LargestCloneGenotype`: genotipo del clon con mayor número de individuos.

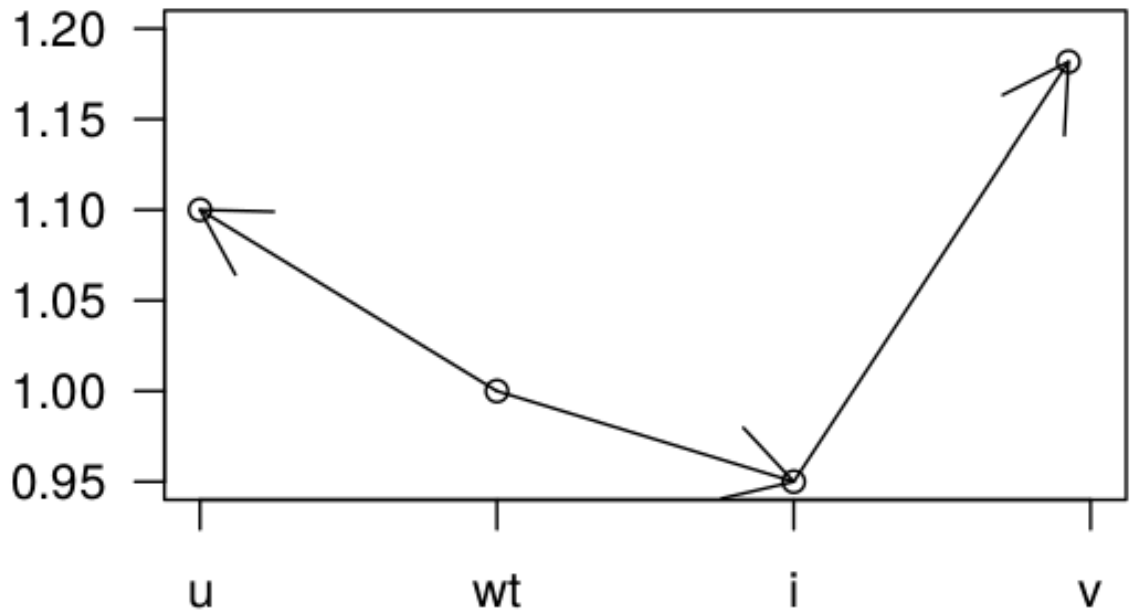
# Integración y pruebas

**Integración:** integrada en el paquete $OncoSimulR$ como una función $discreteModel()$, que se llamaría desde R al instalar la librería.

**Pruebas:** se utiliza el paquete $testthat$. Se prueban:

- La correcta entrada de parámetros: vector $mu$ de probabilidades, población inicial, tiempo de ejecución...
- Correcta salida de información según se cambien los parámetros `itInfo` y `verbosity`.
- Que el modelo obtenga la solución esperable en situaciones evidentes. Por ejemplo: $"A > B" = 0{,}4, "B > A" = 0{,}1$, debe obtener como genotipo dominante $A > B$.
- Que la población se mantenga en torno a su valor inicial de población.

| | Genotipo | Fitness |
|---|---|---|
| 1 | WT | 1 |
| 2 | i | 0.95 |
| 3 | u | 1.1 |
| 4 | i, v | 1.1875 |
| 5,...,8 | resto opciones | 0 |



Ejemplo de salida con `popIni=10000`, `tMax=1000`, `itInfo=100`, `verbosity=3`, **mu<0.0001.**

```
Start --> Population size: 10000---- average fitness: 1 ---- clones: 1
Clon 0 ---- size: 10000 ---- genotype:  ---- fitness: 1
After mutation 100--> Population size: 10033---- average fitness: 1.09092 ---- clones: 3
After mutation 200--> Population size: 9975---- average fitness: 1.1 ---- clones: 1
After mutation 300--> Population size: 9976---- average fitness: 1.09989 ---- clones: 2
After mutation 400--> Population size: 9926---- average fitness: 1.09978 ---- clones: 3
After mutation 500--> Population size: 10027---- average fitness: 1.1 ---- clones: 1
After mutation 600--> Population size: 10065---- average fitness: 1.1 ---- clones: 1
After mutation 700--> Population size: 9960---- average fitness: 1.09989 ---- clones: 2
After mutation 800--> Population size: 9948---- average fitness: 1.09989 ---- clones: 2
After mutation 900--> Population size: 10060---- average fitness: 1.09978 ---- clones: 3
After mutation 1000--> Population size: 10000---- average fitness: 1.1 ---- clones: 1
End: --> Population size: 10000---- average fitness: 1.1 ---- clones: 1
Clon 0 ---- size: 10000 ---- genotype: u ---- fitness: 1.1
```

Ejemplo de salida con `popIni=10000`, `tMax=1000`, `itInfo=100`, `verbosity=3`, **mu<0.001.**

```
Start --> Population size: 10000---- average fitness: 1 ---- clones: 1
Clon 0 ---- size: 10000 ---- genotype:  ---- fitness: 1
After mutation 100--> Population size: 9917---- average fitness: 1.09784 ---- clones: 4
After mutation 200--> Population size: 10128---- average fitness: 1.09859 ---- clones: 3
After mutation 300--> Population size: 9839---- average fitness: 1.09944 ---- clones: 3
After mutation 400--> Population size: 9872---- average fitness: 1.09967 ---- clones: 3
After mutation 500--> Population size: 9938---- average fitness: 1.09911 ---- clones: 3
After mutation 600--> Population size: 9968---- average fitness: 1.0989 ---- clones: 3
After mutation 700--> Population size: 10027---- average fitness: 1.09901 ---- clones: 3
After mutation 800--> Population size: 9933---- average fitness: 1.09922 ---- clones: 3
After mutation 900--> Population size: 10011---- average fitness: 1.09923 ---- clones: 3
After mutation 1000--> Population size: 10022---- average fitness: 1.09923 ---- clones: 2
End: --> Population size: 10022---- average fitness: 1.09923 ---- clones: 2
Clon 0 ---- size: 10015 ---- genotype: u ---- fitness: 1.1
Clon 1 ---- size: 7 ---- genotype: i, u ---- fitness: 0
```

Ejemplo de salida con `popIni=10000`, `tMax=1000`, `itInfo=100`, `verbosity=3`, **mu<0.01**.

```
Start --> Population size: 10000---- average fitness: 1 ---- clones: 1
Clon 0 ---- size: 10000 ---- genotype:  ---- fitness: 1
After mutation 100--> Population size: 9993---- average fitness: 1.1746 ---- clones: 5
After mutation 200--> Population size: 9988---- average fitness: 1.17585 ---- clones: 2
After mutation 300--> Population size: 9884---- average fitness: 1.17717 ---- clones: 2
After mutation 400--> Population size: 10015---- average fitness: 1.17884 ---- clones: 2
After mutation 500--> Population size: 10057---- average fitness: 1.17817 ---- clones: 2
After mutation 600--> Population size: 10140---- average fitness: 1.17661 ---- clones: 2
After mutation 700--> Population size: 9763---- average fitness: 1.17631 ---- clones: 2
After mutation 800--> Population size: 10038---- average fitness: 1.17768 ---- clones: 2
After mutation 900--> Population size: 10123---- average fitness: 1.17565 ---- clones: 2
After mutation 1000--> Population size: 10126---- average fitness: 1.17683 ---- clones: 2
End: --> Population size: 10126---- average fitness: 1.17683 ---- clones: 2
Clon 0 ---- size: 10035 ---- genotype: i, v ---- fitness: 1.1875
Clon 1 ---- size: 91 ---- genotype: i, u, v ---- fitness: 0
```

Resultados de ejecutar 10000 veces la simulación para cada combinación:

| mu | popIni | tMax | WT | u | i,v |
|---|---|---|---|---|---|
| $[0,0{,}01]$ | 100 | 1000 | 83 | 8768 | 1149 |
| $[0,0{,}01]$ | 100 | 10000 | 3 | 8842 | 1155 |
| $[0,0{,}01]$ | 1000 | 1000 | 2 | 5636 | 4362 |
| $[0,0{,}01]$ | 1000 | 10000 | 0 | 5560 | 4440 |
| $[0,0{,}001]$ | 100 | 1000 | 2436 | 7539 | 25 |
| $[0,0{,}001]$ | 100 | 10000 | 280 | 9631 | 89 |
| $[0,0{,}001]$ | 1000 | 1000 | 165 | 9740 | 95 |
| $[0,0{,}001]$ | 1000 | 10000 | 6 | 9828 | 166 |

Observaciones:

1. Cuando la probabilidad de mutación es alta se observan resultados muy semejantes al variar el parámetro `tMax`.
2. Al aumentar el tamaño de la población inicial se observa que el genotipo más ventajoso se impone sobre los demás más veces.

Se comprueba la veracidad de la primera observación mediante los siguientes resultados:

| mu | popIni | tMax | WT | u | i,v |
|---|---|---|---|---|---|
| $[0, 0{,}01]$ | 100 | 1000 | 83 | 8768 | 1149 |
| $[0, 0{,}01]$ | 100 | 900 | 118 | 8718 | 1163 |
| $[0, 0{,}01]$ | 100 | 800 | 139 | 8744 | 1116 |
| $[0, 0{,}01]$ | 100 | 700 | 163 | 8714 | 1121 |
| $[0, 0{,}01]$ | 100 | 600 | 196 | 8769 | 1035 |
| $[0, 0{,}01]$ | 100 | 500 | 239 | 8709 | 1052 |
| $[0, 0{,}01]$ | 100 | 400 | 329 | 8610 | 1060 |
| $[0, 0{,}01]$ | 100 | 300 | 518 | 8482 | 999 |
| $[0, 0{,}01]$ | 100 | 200 | 817 | 8274 | 905 |
| $[0, 0{,}01]$ | 100 | 100 | 1961 | 7372 | 664 |

Se comprueba la veracidad de la segunda observación mediante los siguientes resultados:

| mu | popIni | tMax | WT | u | i,v |
|---|---|---|---|---|---|
| $[0,0{,}01]$ | 10000 | 1000 | 0 | 1302 | 8698 |

Se comprueba también el tiempo de ejecución del modelo. Mediante los siguientes resultados:

| mu | popIni | tMax | tiempo (s) |
|---|---|---|---|
| $[0,0{,}01]$ | 100 | 1000 | 123.48 |
| $[0,0{,}01]$ | 100 | 10000 | 1235.96 |
| $[0,0{,}01]$ | 1000 | 1000 | 961.80 |
| $[0,0{,}01]$ | 1000 | 10000 | 9676.43 |
| $[0,0{,}001]$ | 100 | 1000 | 98.44 |
| $[0,0{,}001]$ | 100 | 10000 | 1141.48 |
| $[0,0{,}001]$ | 1000 | 1000 | 930.12 |
| $[0,0{,}001]$ | 1000 | 10000 | 9511.64 |

Se puede observar que el tiempo de ejecución del algoritmo es linealmente proporcional a:

$$\texttt{popIni} \cdot \texttt{tMax}$$

# Conclusiones

- El análisis de datos a partir del muestreo es una alternativa válida y eficiente al cálculo estadístico a partir de las matemáticas.
- El resultado final de las simulaciones no es tan trivial como suponer que se va a imponer el genotipo más ventajoso.
- No todos los parámetros tienen porque tener un papel determinante a la hora de obtener las probabilidades de fijación. Es necesario estudiar cada escenario con el fin de elaborar un análisis de datos eficiente y de calidad.

# Líneas de trabajo futuro

- Incorporar componente espacial al modelo.
- Ampliar la información de salida que ofrece el modelo.
- Añadir funciones de procesado estadístico de los modelos para obtener las probabilidades de fijación de los genotipos en diferentes condiciones.
- Añadir una interfaz de usuario (por ejemplo con $Shiny$) que mejore visualmente la salida y entrada del programa.

# Líneas de trabajo futuro

- Incorporar componente espacial al modelo.
- Ampliar la información de salida que ofrece el modelo.
- Añadir funciones de procesado estadístico de los modelos para obtener las probabilidades de fijación de los genotipos en diferentes condiciones.
- Añadir una interfaz de usuario (por ejemplo con $Shiny$) que mejore visualmente la salida y entrada del programa.

# Gracias por vuestra atención

Turno de preguntas