

# Extensión del modelo Sugarscape: Documentación

Jonathan Arturo Preciado Reyes

September 12, 2024

## 1 Sobre el modelo

El modelo original de Sugarscape fue propuesto por los economistas J. Epstein y R. Axtell en el capítulo 2 de su libro *Growing Artificial Societies: Social Science from the Bottom Up*, publicado en 1996. Provee una simulación hecha desde cero de desigualdad. En ella, los patches o casillas tienen una cantidad variable de azúcar que representa la riqueza y está dispuesta para el consumo de los agentes. En este modelo se puede observar que solo una pequeña minoría de la población tiene una riqueza por encima del promedio, mientras que la mayoría de los agentes tienen una riqueza cercana a la provista en la inicialización.

### 1.1 Descripción general

Cada casilla tiene una cantidad arbitraria de azúcar en un rango de 0 a la cantidad máxima dada por el usuario (en NetLogo, ésta cantidad es establecida por un mapa donde también se especifica la distribución de azúcar en cada patch). Con cada iteración, cada casilla aumenta su azúcar en una unidad hasta llegar a la cantidad máxima. Visualmente, esta riqueza se puede observar por el color de la casilla, entre más oscura, más azúcar tiene. En la inicialización, los agentes se encuentran dispersos de forma aleatoria por la rejilla, todos ellos con un rango de visión entre 1 y 6 casillas de distancia en horizontal y vertical. Éstos también contarán con una edad máxima de entre 60 y 100. Si un agente rebasa esta edad, morirá. Para cada unidad de tiempo (tick), cada agente se moverá a la casilla más cercana sin ocupar con mayor cantidad de azúcar de acuerdo a su rango de visión y recolectará toda la azúcar del patch. El agente se mantendrá en la misma casilla si esta sigue teniendo más azúcar que el resto de casillas a la vista sin ocupar. El agente usará y perderá una cierta cantidad de azúcar por cada tick según su rango de metabolismo. Si un agente se queda sin azúcar, éste morirá. Para mantener la población constante, se generará un nuevo agente de acuerdo a la inicialización antes descrita que reemplazará al agente muerto.

## 2 Parámetros y funciones de las casillas

### 2.1 patches-own

Establece las variables de los patches. Las casillas tendrán dos variables: *psugar* y *max-psugar*, que reflejan la cantidad actual de azúcar en el patch y la cantidad máxima, respectivamente.

### 2.2 setup-patches

Función que inicializa los patches. Esta se encarga de cargar un mapa con la distribución del azúcar en la latiz. Dicho mapa es un archivo de texto plano llamado *sugar-map.txt*. Con él, cada patch leerá su cantidad de azúcar, que funge como la cantidad inicial y a su vez como la cantidad máxima que el patch puede almacenar. Finalmente, el patch obtiene un color de acuerdo con una función *patch-recolor* que depende de la cantidad de azúcar.

## 2.3 patch-recolor

Función que establece un tono de amarillo a este patch. A mayor cantidad de azúcar almacenada, más oscuro será el tono de la casilla.

## 2.4 patch-growback

Función que aumenta en uno la cantidad de azúcar de este patch

# 3 Parámetros y funciones de los agentes.

## 3.1 turtles-own

Establece las variables de los agentes, las cuales son:

Variable	Descripción
sugar	Variable que almacena la cantidad actual de este agente.
metabolism	La cantidad de azúcar que el agente pierde con cada tick.
vision	La distancia que este agente puede ver en horizontal y vertical.
vision-points	Los puntos que este agente puede ver en relación con su posición actual (depende de vision).
age	La edad actual en ticks de este agente.
max-age	La edad en la que este agente morirá por causas naturales.
decile	El segmento de la población al que este agente pertenece de acuerdo a su cantidad de azúcar.
changed	Indica si el agente sufrió un cambio de decil
past-sugar	Una variable que guarda la cantidad de azúcar que tuvo el agente en el tick pasado
diff-sugar	la diferencia entre la cantidad de azúcar actual y la del tick pasado

Table 1: Tabla de Variables del Agente

## 3.2 turtle-setup

Inicializa los valores por defecto de un agente. Se establece el color rojo, forma circular, posición en la rejilla aleatoria pero sin ocupar el espacio de otro agente, un valor de metabolismo aleatorio uniforme entre 1 y 4, un valor de edad máxima aleatorio entre 60 y 100, una edad aleatoria normal con media 30 y desviación estándar de 10, un parámetro de visión aleatorio entre 1 y 6 y puntos de visión de acuerdo al parámetro vision.

## 3.3 turtle-move

El agente decide moverse a la casilla más cercana (o mantenerse en la actual) según su disponibilidad y riqueza en azúcar. Solo puede considerar aquellas casillas que se encuentren dentro de su rango de visión (vision points). Un agente podrá tomar una decisión aleatoria de movimiento con un 20% de probabilidad para simular cierto nivel de irracionalidad.

### 3.4 turtle-eat

Colecta todo el azúcar de esta casilla y metaboliza (pierde azúcar) de acuerdo con su parámetro metabolism.

### 3.5 my-current-decile

Retorna el decil actual de este agente. Considerando los rangos de los diez deciles de la variable global "deciles", el agente determina a cuál grupo pertenece en este tick.

## 4 Parámetros globales, métricas y visualización.

### 4.1 Parámetros globales

Variable	Descripción
gini-index-reserve	Suma acumulada de las diferencias entre la proporción de la población y el acumulado de la riqueza
lorenz-points	Variable utilizada para construir la curva de Lorenz
deaths	Muertes totales
starvation	Muertes por falta de azúcar
starvation-per-tick	La cantidad de agentes muertos por falta de azúcar en este tick
avg-gini	El índice Gini promedio en los últimos 100 ticks
ginis	Una lista con los valores del índice Gini en los últimos 100 ticks (usado para calcular avg-gini)
gini	El valor actual del índice Gini
deciles	Una lista con las cotas superiores de cada decil
total-wealth	La suma total del azúcar de los agentes
productivity	El azúcar per cápita del sistema. Se define como "total-wealth" dividido por la cantidad de agentes
productivities	Una lista con los últimos 100 valores de "productivity"
avg-productivity	El grado promedio de azúcar per cápita en las últimas 100 iteraciones
treasury	La cantidad de azúcar recaudada con impuestos (si aplica) en esta iteración
dist-by-decile	Una lista con el porcentaje total de azúcar que tiene cada decil.

Table 2: Tabla de Variables globales

### 4.2 Métricas

#### 4.2.1 Riqueza

**Distribución de riqueza.** Un histograma que muestra la distribución de la riqueza por decil. La cantidad de agentes se encuentra en el eje de las ordenadas, la cantidad de azúcar en el eje de las abscisas. Los deciles se calculan con cada iteración con la función "update-deciles"

**Riqueza per cápita.** Una gráfica del valor "productivity", que se calcula como la riqueza total de los agentes dividida por la cantidad de agentes.

**Curva de Lorenz.** Una métrica del porcentaje de riqueza según la población. Entre más se acerque la curva a la línea de la función identidad, menos desigualdad habrá. La curva de Lorenz y el índice Gini acumulado se calcula con la función "update-lorenz-and-gini" por cada tick

**Índice Gini frente al tiempo.** Una gráfica de la variación del índice Gini con el paso de los ticks .El índice Gini es una medida de entre 0 y 1 que mide la desigualdad. Entre más cercano a 1 sea su valor, más desigualdad habrá. Se calcula con cada iteración con la variable "gini-index-reserve"

**Gini.** Un monitor del promedio de los últimos 100 índices Gini. Se calcula con la función "update-avg-gini".

**Impuestos recaudados.** Una gráfica del parámetro treasury.

#### 4.2.2 Edad y muertes

**Edad promedio.** Un monitor de la edad promedio de la población.

**Edades.** Un histograma de la distribución de las edades de la población. La cantidad de agentes se encuentra en el eje de las ordenadas.

**Muertes por hambruna.** Un monitor que muestra la cantidad de muertes de los agentes por falta de azúcar hasta ahora.

**Muertes totales.** Un monitor que muestra la cantidad de muertes hasta ahora.

### 4.3 Visualización

Es posible visualizar a los agentes según determinados valores:

Valor	Descripción
no-visualization	Sin visualización. Todos los agentes se muestran rojos.
color-agents-by-vision	Colorea a los agentes según su capacidad de visión, a mejor visión, más oscuro será un agente.
color-agents-by-metabolism	Colorea a los agentes según su metabolismo. A mayor nivel de metabolismo, más claro será un agente.
color-agents-by-age	Colorea a los agentes según su edad. Los agentes más jóvenes tendrán tonos cercanos al azul, los más longevos tendrán tonos cercanos al rojo.
color-agents-by-decile	Colorea a los agentes según el decil de la población al que pertenecen. Los deciles más altos tendrán tonos cercanos al rojo, los más bajos al azul.

Table 3: Opciones de Visualización de Agentes

## 5 Inicialización y ejecución del modelo

### 5.1 Parámetros de control.

Es posible cambiar la cantidad de población con el deslizador "initial-population". La dotación inicial de azúcar varía según el rango mínimo y máximo controlables por los deslizadores "minimum-sugar-endowment" y "maximum-sugar-endowment" respectivamente.

## 5.2 Inicialización general

La rejilla está compuesta por  $50 * 50$  patches con fronteras periódicas. La inicialización se da con la función set-up y se encarga de comprobar que el rango de dotación inicial de azúcar sea válido, para posteriormente inicializar la población, los patches y las variables globales.

## 5.3 Ejecución

La ejecución del modelo está dada por la constante ejecución de la función go, la cual se encarga de comprobar que el rango de dotación inicial sea válido y que aún haya agentes en la rejilla como condiciones de paro. Respecto a las variables globales, la función go se encarga de reiniciar la variable starvation-per-tick para borrar registros previos de muertes por falta de azúcar y calcular los nuevos. Referente a los patches, con cada iteración, el azúcar vuelve a crecer en una unidad y el color del patch cambia según su cantidad de azúcar con la llamada a las funciones "patch-growback" y "patch-recolor" respectivamente. En cuanto a los agentes, se les mueve y se les hace metabolizar, para posteriormente aumentar en una unidad su edad y recalcular el decil al que pertenecen. Si los agentes llegan a su máxima edad o se quedan sin azúcar, estos mueren y actualizan las variables deaths y starvation / starvation-per-tick según sea el caso. Finalmente, se actualizan los valores de la curva de Lorenz, el índice Gini, la riqueza acumulada actual y los deciles para obtener las métricas de interés.

# 6 Implementación de políticas de redistribución de la riqueza

Dependiendo de los seleccionadores "taxation" y "redistribution", se recolectarán impuestos y se redistribuirá riqueza de maneras distintas. Por defecto, el sistema se encontrará en "no-collection" y "no-redistribution", en donde no se ejecutan políticas de recaudación ni redistribución de riqueza.

## 6.1 linear-collection

Método que recolecta riqueza de forma lineal. Se toma el decil de los agentes y se multiplica por 0.03, de manera que los agentes con decil 1 pagan solo un 3% de impuestos, mientras que los agentes con decil 10 pagan el 30%.

## 6.2 dynamic-collection

Método que recolecta impuestos a los agentes según su decil de forma dinámica dependiendo de la lista "dist-by-decile", de forma que recauda tanto porcentaje de riqueza como porcentaje de concentración de la riqueza haya en cada decil.

## 6.3 uniform-collection

Método que recolecta de manera uniforme el 10% de la riqueza individual de los agentes.

## 6.4 UBI

Método que redistribuye la riqueza recaudada de forma uniforme para todos los agentes.

## 6.5 poorest

Método que redistribuye la riqueza recaudada únicamente a agentes con decil I y II, dirigiendo un 60% al decil I y un 40% al decil II.

## 6.6 linear

Método que redistribuye la riqueza recaudada de forma lineal decreciente, de manera que el decil I recibe el 20%, el decil II el 17% y así sucesivamente siguiendo la fórmula

$$\frac{10 - decil}{45}$$

## 6.7 dynamic

Método que redistribuye la riqueza recaudada de forma inversamente proporcional a lo recaudado en el método de recolección dinámico.