Tiko'n Documentation

Versión 3.0.0

Julien Malard

08 de septiembre de 2022

General


Tiko'n

CAPÍTULO 1

¿Qué es Tiko'n?

Tiko'n es un programa para desarrollar modelos de redes tróficas agroecológicas de manera sencilla, reproducible y divertida.

CAPÍTULO 2

¿Por qué se llama Tiko'n?

Tiko'n quiere decir *cultivo* o *agricultura* en Kaqchikel.

CAPÍTULO 3

¿Qué es un modelo de redes tróficas?

Un modelo de redes tróficas es un modelo que representa las interacciones entre organismos vivos en un campo agrícola (en general insectos y otros artrópodos). Predice el nivel de control biológico natural, las presiones futuras de plagas dado factores como el clima y la presencia de insectos benéficos, y hacia permite predecir cuáles serían los impacto de varias estrategias de manejo de plagas.

CAPÍTULO 4

¿Por qué utilizar Tiko'n?

Redes tróficas con cosas muy difíciles a modelizar. Aun las más sencillas (3 insectos) presentan problemas como

1. Multitud de parámetros (15+ por insecto)
2. Inestabilidad matemática (dificultando predicciones fiables)
3. Multitud de posibilidades de ecuaciones en la literatura para cada fase de vida (reproducción, edad, transiciones, muertes, depredación)
4. Muy alta incertidumbre en los valores verdaderos de los parámetros
5. Pocos datos para calibración y validación
6. Requerimiento de conocimientos informáticos avanzados para conectar con modelos de cultivos externos o con predicciones de cambios climáticos

Así que si quieres hacer modelos tróficos pero te desaniman estos retos, ¡Tiko'n es para ti! Tiko'n manejará todos los puntos arriba para ti, siempre dándote el nivel de control que quieres sobre los detalles de tu modelo, nada menos, nada más.

1. Colección interna de la mayoría de ecuaciones para redes tróficas disponibles en la literatura, con selección automática de ecuaciones para principiantes
2. Posibilidad de agregar tus propias ecuaciones si quieres
3. Conexión automática con bases de datos y con predicciones y observaciones climáticas
4. Manejo automático de parametros y de cálculos de poblaciones y de depredación
5. Funcionalidades integradas para calibración y validación de modelos
6. Formato estandardizado para guardar y compartir modelos calibrados
7. Y, por supuesto, gráficos bonitos

CAPÍTULO 5

¿Qué no es Tiko'n?

Tiko'n **no** es, y no intenta(rá) ser:

- Un modelo basado en agentes. Estos modelos toman demasiado poder computacional a las escalas geográficas y temporales y con el nivel de incertidumbre paramétrico y estocástico para los cuales se diseñó Tiko'n.
- Un modelo con resolución intraparcela. Si quieres más resolución espacial, debes crear parcelas más pequeñas.

CAPÍTULO 6

Contenido

6.1 Instalación

Es muy fácil instalar Tiko'n. Primero necesitarás la versión más recién de Python. Después, puedes instalar Tiko'n en la terminal con:

```
pip install tikon
```

Si quieres la versión más recién (en desarrollo), puedes obtenerla de GitHub directamente con:

```
pip install git+git://github.com/julienmalard/tikon.git@master
```

Nota: Si tienes Windows, es posible que tengas que instalar el `C++ redistributable` de aquí. Toma la versión terminando en `...x86.exe` si tienes Python de 32 bits y en `...x64.exe` si tienes **Python** (no Windows) de 64 bits. Después, instálalo. Por razones obscuras, `SciPy`, un paquete requerido por Tiko'n, no funciona en Windows sin éste.

6.2 Estructura general

La modelización agroecológica es un tema un poco más complicado que otros tipos de modelos por causa de su complejidad y comportamiento dinámico. La estructura de Tiko'n toma en cuenta estos desafíos por su inclusión de repeticiones estocásticas, incertidumbre paramétrica, y filosofía modular y flexible.

Aquí sigue una breve introducción a los términos específicos a Tiko'n.

6.2.1 Simulador

Un `Simulador` es la unidad fundamental de un modelo en Tiko'n. Contiene varios módulos (`Módulo`) que interactuan entre sí mismos. Todas simulaciones y calibraciones se efectuan a través de un `Simulador`.

6.2.2 Módulos

Cada `Módulo` en Tiko'n representa una parte del agroecosistema. Por ejemplo, existen módulos para la red agroecológica, para el clima, para el cultivo y para el manejo humano. Los módulos pueden intercambiar valores de variables en el transcurso de una simulación.

6.2.3 Experimentos

Un `Exper` representa un experimento, o sea, una combinación de decisiones de observaciones (reales o hipotéticas) para una simulación. Aun simulaciones sin datos observados implementan un experimento vacío automáticamente.

6.2.4 Simulaciones

Se efectuan simulaciones por llamar `simular()` con especificaciones de escala temporal, repeticiones paramétricas y estocásticas, y experimento. Adentro de cada simulación, el modelo se va a `iniciar()`, `correr()`, y finalmente `cerrar()`.

6.2.5 Resultados

Los resultados de simulación tienen su propia clase (`ResultadosSimul`), la cual incluye los resultados (`ResultadosMódulo`) de cada módulo del simulador, los cuales en torno contienen los resultados (`Resultado`) de cada variable del módulo. Resultados se pueden `validar()` y también `graficar()`.

6.2.6 Parámetros

Por supuesto, todo modelo necesita parámetros. En Tiko'n, los parámetros se implementan por `Parám`, y cada parámetro puede tener varias calibraciones conteniendo distintas distribuciones de valores (`Dist`).

6.2.7 Ecuaciones

El módulo `RedAE` implementa ecuaciones (`Ecuación`) para representar cada fase del ciclo de vida de los insectos en la red. Las ecuaciones se pueden por supuesto modificar, agregar, o desactivar según sus necesidades.

6.3 Publicaciones

En cuento se publiquen artículos sobre Tiko'n, se incluirán aquí.

6.4 Agradecimientos

Tiko'n es un proyecto de fuente abierta. Las personas e instituciones siguientes hicieron posible su desarrollo.

6.4.1 Autores del código

- Julien Malard (Sitio internet)
- Marcela Rojas (Sitio internet)

6.4.2 Traductores

- Julien Malard (Sitio internet)

6.4.3 Instituciones afiliadas

Tiko'n se desarrolla en colaboración con las instituciones siguientes:

6.4.4 Financiamiento

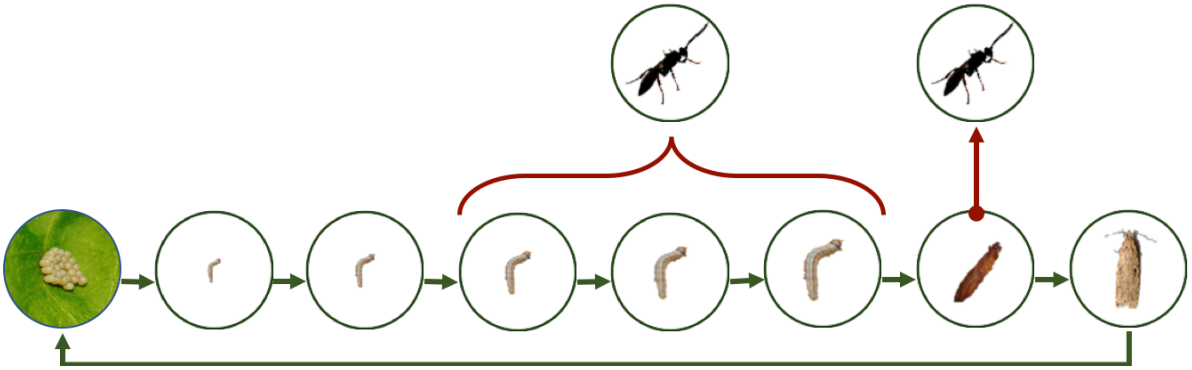
Los organismos siguientes contribuyeron apoyo financiero a uno o más de los autores del código mientras trabajaban en el desarrollo de Tiko'n:

- Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG)
- Centre de recherches pour le développement international du Canada (CRDI)
- Fonds de recherche Nature et technologies du Québec (Canada) (FQRNT)

6.5 Introducción

En este ejemplo vamos a construir nuestra propria red trófica de un sistema de coco en Sri Lanka. Para guardar las cosas sencillas no vamos a incluir el cultivo. De todo modo, no tengo modelo de cultivo para árboles de coco.

Aquí está un imagen general de los insectos involucrados (3 en total). Tenemos solamente una plaga, y dos parasitoides se atacan a varias etapas de su ciclo de vida.



6.5.1 Especificar insectos

Primero especificaremos nuestros insectos presentes, una oruga y dos parasitoides (ver *Insectos* para la lista completa de clases disponibles).

```
from tikon.rae.orgs.insectos import MetamCompleta, Parasitoide
from tikon.rae.red_ae import RedAE

# Mariposas tienen metamórfosis completa
Oarenosella = MetamCompleta('O. arenosella', njuvenil=5)

# 2 tipos de parasitoides
Paras_larvas = Parasitoide('Parasitoide larvas', pupa=True)
Paras_pupa = Parasitoide('Parasitoide pupa')

# Juntamos todo en una red
with RedAE([Oarenosella, Paras_larvas, Paras_pupa]) as red:
    # El parasitoide de larvas parasita las fases 3, 4, y 5 de O. arenosela y emergen␣
↪después de la quinta
    Paras_larvas.parasita(Oarenosella, ['juvenil_3', 'juvenil_4', 'juvenil_5'], etp_
↪emerg='juvenil_5')

    # El parasitoide de pupas parasita y emerge de la pupa
    Paras_pupa.parasita(Oarenosella, 'pupa', etp_emerg='pupa')
```

6.5.2 A prioris

Un *a priori* es una distribución probable del valor de un parámetro que especificas antes de correr una calibración. Ayuda el algoritmo de calibración a encontrar el mejor valor del parámetro.

Aquí vamos a cargar las distribuciones a prioris para los parámetros de la red. Los tienes que especificar manualmente para cada experimento.

> **Advertencia:** Es **muy importante** establecer buenos *a prioris* para calibraciones. Las limitaciones de algoritmos existentes de calibración de estos tipos de modelos hacen primordial la especificación de *a prioris* razonables y bastante precisos para todos los parámetros del modelo.
>
> Nota para estudiantes: Mejorar estos algoritmos podría ser buena idea de tesis. :)

En este caso ya especificamos nuestros a prioris en un documento separado así que los vamos a cargar directamente.

```
from tikon.ejemplos.opisina_arenosella.a_prioris import a_prioris
red.espec_aprioris(a_prioris)
```

6.5.3 El experimento

Aquí se conecta la red con observaciones de campo a través de un experimento (`Exper`).

```
from tikon.central import Parcela
from tikon.ejemplos.datos import obt_datos, obt_ref
from tikon.exper.exper import Exper
from tikon.rae.red_ae.obs import ObsPobs

# Datos de observaciones
datos = obt_datos('Perera et al 1988/Oarenosella_A.csv')

# También se puede visualisar la referencia para los datos
print(obt_ref('Perera et al 1988/Oarenosella_A.csv'))

# Se trata de observaciones de poblaciones (y no de otro variable, como depredación).
pobs = ObsPobs.de_csv(
    datos,
    col_tiempo='Día',
    corresp={
        'Estado 1': Oarenosella['juvenil_1'],
        'Estado 2': Oarenosella['juvenil_2'],
        'Estado 3': Oarenosella['juvenil_3'],
        'Estado 4': Oarenosella['juvenil_4'],
        'Estado 5': Oarenosella['juvenil_5'],
        'Pupa': Oarenosella['pupa'],
        'Para_larva_abs': Paras_larvas['juvenil'],
        'Para_pupa_abs': Paras_pupa['juvenil']
    },
    factor=655757.1429 / 500  # para convertir a individuos por ha
)
exper_A = Exper('Sitio A', Parcela('Sitio A', geom=GeomParcela((7.297, 79.865))))
exper_A.datos.agregar_obs(pobs)
```

6.5.4 Calibración

Ahora vamos a calibrar nuestro modelo. Primero creamos un `Simulador` para poder correr simulaciones y calibraciones. En nuestro ejemplo sencillo el simulador solamente tiene un módulo (la red), pero se podrían incluir clima, manejo, o cultivos también.

```
from tikon.central. import Modelo

modelo = Modelo(red)

modelo.calibrar('Sitio A', exper=exper_A)
```

6.5.5 Validación

En este ejemplo vamos a hacer trampa y validar con los mismos datos de calibración. Primero hacemos una simulación normal, y despues vamos a `validar()` los resultados. También los podremos `graficar()`.

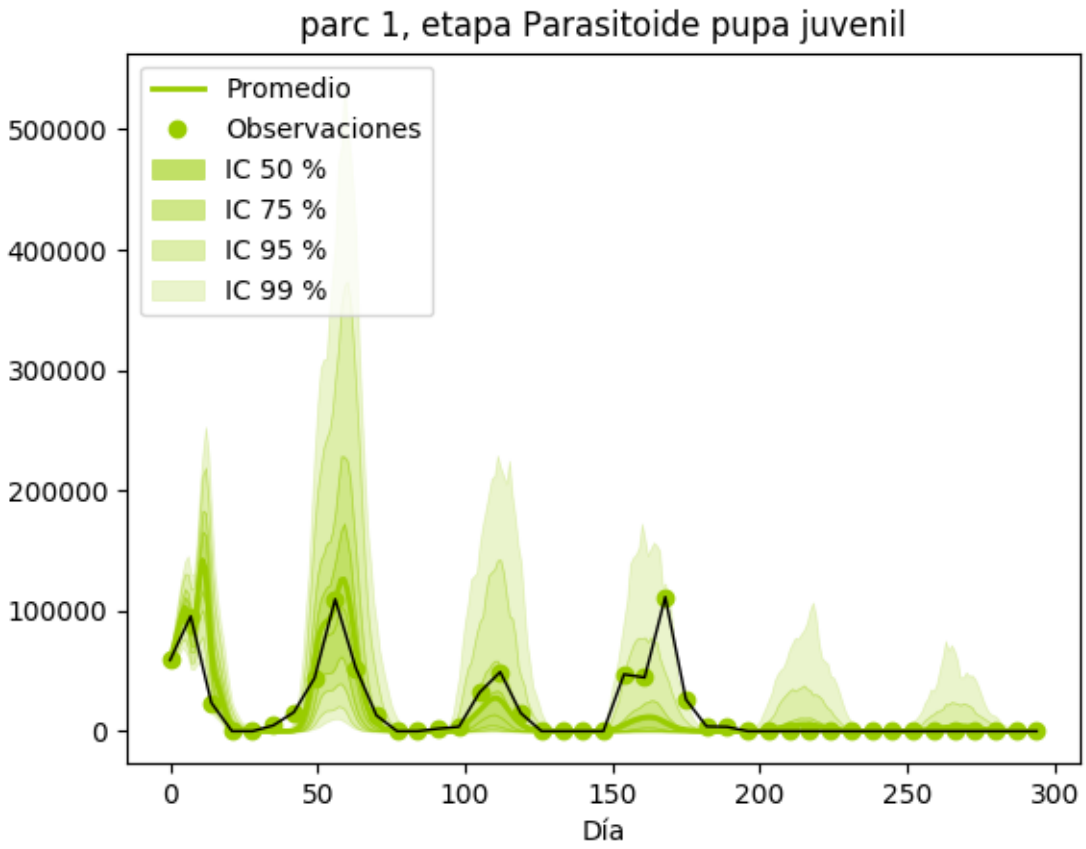
Las observaciones especificadas arriba quedaron vinculadas en los resultados y por eso se tomarán en cuenta en la validación y en los gráficos.

```
res = simul.simular(exper=exper_A)

from pprint import pprint
pprint(res.validar())

res.graficar('gráficos Sitio A')
```

Tiko'n generará un gráfico para cada insecto de la red, con su población predicha, los márgenes de incertidumbre y las observaciones, si hay.



6.5.6 Guardar y cargar

Vamos a guardar los resultados de la calibración para ahorar tiempo en el futuro. Tiko'n calibra automáticamente las poblaciones iniciales para etapas cuyas poblaciones no se observaron en el experimento, así que guardaremos la calibración del experimento también.

```
simul.guardar_calib('calibs Sitio A')
exper_A.guardar_calib('calibs Sitio A')
```

Se pueden después cargar las calibraciones para más trabajo. Igualmente se pueden compartir entre usuarias de Tiko'n. Por ejemplo, en otra sesión de Python:

```
red.cargar_calib('calibs Sitio A')
exper_A.cargar_calib('calibs Sitio A')

red.simular(exper=exper_A)
```

Igualmente puedes guardar tu calibración al directorio de Tiko'n. Será después disponible para todas los usuarios de tu instalación de Tiko'n. Si quieres, también lo puedes compartir en GitHub con el resto de la comunidad de Tiko'n.

```
from tikon.ejemplos.calibs import guardar_calib

guardar_calib(
    [red, exper_A],
```

(continué en la próxima página)

(proviene de la página anterior)

```
    'Opisina arenosella, Perera et al. 1988',
    autor='Yo :)'
    correo='julien.malard@mail.mcgill.ca',
    detalles='Calibración con Sitio A'
)
```

Después se podrá acceder con:

```
from tikon.ejemplos.calibs import obt_calib, obt_ref

dir_ = 'Opisina arenosella, Perera et al. 1988'
red.cargar_calib(obt_calib(dir_))
exper_A.cargar_calib(obt_calib(dir_))

# Visualizar la información de la calibración
print(obt_ref(dir_))
```

6.6 Manejo

En Tiko'n, *manejo* se refiere a acciones que toman humanos para modificar el sistema agrícola.

Aquí vamos a predecir el impacto de biocontrol con el modelo de *Opisina arenosella* que ya desarrollamos.

6.6.1 Cómo funciona

Una **regla** (`Regla`) tiene dos componentes: una **condición** (`Condición`) y una o más **acciones** (`Acción`). Cuando se cumple la condición, Tiko'n ejecutará el acción. ¡Sencillo!

Las condiciones pueden ser según el tiempo (`CondTiempo` y `CondCada`) o según el valor de un variable particular (`CondVariable`). Igualmente puedes combinar varias condiciones con `CondY` y `CondO`.

Nota: Las condiciones basades en valores de variables se evaluan **separadamente** para cada repetición estocástica y paramétrica del modelo.

6.6.2 Biocontrol

Primero cargamos el modelo y sus calibraciones.

```
from tikon.ejemplos.calibs import obt_calib
from tikon.ejemplos.proyectos.opisina_arenosella import modelo as mod
from tikon.central.simulador import Simulador

dir_calibs = 'Opisina arenosella, Perera et al. 1988'

mod.red.cargar_calib(obt_calib(dir_calibs))
mod.exper_A.cargar_calib(obt_calib(dir_calibs))
```

Ahora vamos a soltar parasitoides de *O. arenosella* cada 30 días.

```
from tikon.manejo.acciones import AgregarPob
from tikon.manejo.conds import CondCada
from tikon.manejo.manejo import Manejo, Regla

acción = AgregarPob(mod.Paras_pupa['adulto'], 200000)
cond = CondCada(30)
manejo_tiempo = Manejo(Regla(cond, acción))

simul = Simulador([mod.red, manejo_tiempo])
res_tiempo = simul.simular(400, exper=mod.exper_A, n_rep_estoc=5)

res_tiempo.graficar('mis resultados aquí/temporales')
```

También podemos hacer control biológico más inteligente, y solamente soltar parasitoides cuando la población de *O. arenosella* sube demasiado. En este ejemplo soltamos 200000 parasitoides adultos por hectárea cada vez que tenemos una población de pupas de *O. arenosella* superior a 200000. Igualmente esperamos 30 días después de una aplicación de parasitoides antes de poder considerar una nueva aplicación.

```
from tikon.manejo.conds import CondPoblación, SuperiorOIgual

acción = AgregarPob(mod.Paras_pupa['adulto'], 200000)
cond = CondPoblación(mod.Oarenosella['pupa'], SuperiorOIgual(200000), espera=30)
manejo_dinámico = Manejo(Regla(cond, acción))

simul = Simulador([mod.red, manejo_dinámico])
res_dinámicos = simul.simular(400, exper=mod.exper_A, n_rep_estoc=5)

res_dinámicos.graficar('mis resultados aquí/dinámicos')
```

6.7 Insectos

Tiko'n implementa una gran variedad de ciclos de vida de insectos a través de subclases especiales de `Insecto`.

Nota: Tiko'n toma una vista *ecológica* de lo que es un insecto. Es decir, si come como un insecto y se come como un insecto, entonces *es* un insecto por lo que le importa a Tiko'n.

Así que no te sorprendes al ver arañas y gusanos en las listas de insecto. Sé que entomólogicamente es una herecía, pero programáticamente es la mejor solución.

Cada insecto viene con sus etapas (huevo, larva, etc.) y las ecuaciones correspondientes ya especificadas.

6.7.1 Sencillo

El tipo de insecto más sencillo posible. Por lo tanto es también generalmente inútil, pruebas teoréticas a parte. Solamente lleva una forma adulta, y se implementa con la clase `Sencillo`.

6.7.2 Metamórfosis completa

Tiko'n lleva la clase `MetamCompleta` para representar a insectos con ciclos de vida completos (de huevo a adulto, pasando por una pupa).

6.7.3 Metamórfosis incompleta

Insectos con ciclos de vida incompletos (sin pupa) se pueden representar con la clase `MetamIncompleta`.

Todos insectos de tipos `Sencillo`, `MetamCompleta` o `MetamIncompleta` pueden ser depredadores:

6.7.4 Parasitoides

Parasitoides, aunque técnicamente por su mayor parte insectos con metamórfosis completa, se representan por su propia clase (`Parasitoide`) porque se debe tomar en cuenta el hecho de que su fase juvenil se desarrolla adentro de su huésped.

6.7.5 Esfécidos

Esfécidos son avispas similares a parasitoides pero que paralizan e inactivan su presa al momento del parasitismo, lo cual puede se interior o exterior. Se deben representar de manera distinta (`Esfécido`) a paridoides convencionales, porque la presa se quita del ecosistema al momento del acto de parasitismo y no al momento de la emergencia de la avispa adulta.

6.7.6 Cambiar ecuaciones

Puedes modificar las ecuaciones empleadas para un insecto en particular.

6.8 Cultivos

Nota: Las funcionalidades de cultivos quedan en desarrollo.

Regresa pronto o ayúdanos.

6.9 A prioris

Distribuciones *a priori* especifican rangos probables para los parámetros de un modelo. Se pueden especificar por el usuario y después se tomarán en consideración para corridas de simulaciones y calibraciones.

Existen dos maneras de especificar una distribución a priori: por densidad y por forma analítica.

> **Advertencia:** Distribuciones *a priori* solamente se toman en cuenta automáticamente para corridas de calibraciones. Ver *Calibraciones y simulaciones* para más detalles.

6.9.1 Densidad

Se puede especificar *a prioris* por un rango de dos valores y el nivel de confianza que el valor verdadero se encuentre efectivamente en el rango. Tiko'n generará automáticamente una distribución que corresponde a la especificación, y eso, tomando también en cuenta los límites teoréticos del parámetro.

6.9.2 Analíticas

También se puede especificar un *a priori* directamente por su forma analítica.

6.9.3 Uso en modelos

Interacciones

En el caso de parámetros con interacciones, los a prioris especificados se aplicarán a la raíz del árbol de interacciones del parámetro. Igualmente puedes especificar un nivel de interacción al cual aplicar el *a priori*.

Nota: Ver *Ecuaciones* para una explicación de niveles en árboles de ecuaciones, y *Heredar interacciones* para una descripción de cómo desactivar la propagación de *a prioris* a través los niveles de interacción.

6.10 Ecuaciones

6.10.1 Árbol de ecuaciones

6.10.2 Especificar ecuaciones

6.11 Con clima

6.12 Calibraciones y simulaciones

En Tiko'n, una *calibración* se refiere a un conjunto de valores para los parámetros de un modelo, todos calibrados en conjunto. Una *simulación* se refiere a cualquier ejecución del modelo. Por ejemplo, calibrar un modelo generalmente requiere la ejecución de una cantidad importante de simulaciones.

Por lo tanto, cada *simulación* normal se corre con una especificación de cuales *calibraciones* tomar en cuenta para establecer los valores de sus parámetros.

6.12.1 Comportamiento automático

Si no se especifican calibraciones para una corrida, Tiko'n tomará una decisión razonable para ti.

- Si es una corrida de **simulación**, se tomará cada calibración disponible, y se ignorarán *a prioris* especificados
- Si es una corrida de **calibración**, se tomarán primero los *a prioris* especificados, y, para los parámetros sin aprioris, se generarán distribuciones a base de las calibraciones ya existentes.

6.12.2 Calibraciones específicas

Si quieres cambiar el comportamiento automático, puedes especificar cuáles calibraciones existantes quieres incluir en la corrida.

6.12.3 Opciones avanzadas

Puedes cambiar el comportamiento automático por pasar un objeto `EspecCalibsCorrida` a la función de simulación `simular()` del `Simulador` (e igualmente a las que la llaman indirectamente, como `calibrar()` y `sensib()`). `EspecCalibsCorrida` toma opciones para especificar el uso de *a prioris*, de correspondencia entre parámetros, y de herencia de interacciones.

A prioris

Puedes especificar si se deberían emplear distribuciones *a prioris* para los parámetros donde han sido especificados. En este caso, se ignorarán todas otras calibraciones disponibles para los parámetros con *a prioris* disponibles.

Correspondencia

Cuando se calibra un modelo con varios parámetros, no solamente importan los valores estimados de los parámetros sino también las interacciones entre los parámetros sí mismos. Es decir, en la calibración no se busca el valor óptimo de cada parámetro individualmente sino un conjunto de valores para todos los parámetros que da buenos resultados. Por eso es importante, tanto como sea posible, guardar las correlaciones entre los valores de los parámetros calibrados y Tiko'n hará su posible para únicamente tomar valores de calibraciones que están disponibles para todos los parámetros necesarios a la simulación.

Si quieres desactivar esta funcionalidad, puedes indicarlo así:

Heredar interacciones

Parámetros en Tiko'n pueden tener interacciones con otros objetos, por ejemplo, en el caso de un parámetro de eficacidad de depredación que también se ve influido por la identidad de la presa. La opción automática es que cada parámetro, si le faltan calibraciones, puede heredar las calibraciones o *a prioris* del nivel de interacción subyacente.

Nota: Ver *Ecuaciones* para una explicación de niveles en árboles de ecuaciones.

6.13 Modificar la Red

Hay dos maneras de agregar funcionalidad a la Red en Tiko'n: se pueden agregar nuevas opciones de organismos, y también se pueden agregar nuevas ecuaciones para modelizar sus ciclos de vida.

6.14 Nuevos cultivos

6.15 Traducciones

Tiko'n es un proyecto abierto y beneficia de ser lo más accesible posible. Puedes ayudar a traducir los varios componentes de Tiko'n en tu idioma aquí. Necesitarás crear una cuenta gratis en Transifex primero.

1. Documentación (lo que estás leyendo ahora)
2. Programa (mensajes de error, etc.)
3. Interfaz (para un día futuro)

Nota: Si no aparece tu idioma favorito en la lista, por favor haz una demanda de activación de idioma en los enlaces arriba.

6.16 Interfaz de programación

6.16.1 Insectos

6.16.2 Clima

6.16.3 Cultivo

6.16.4 Manejo

Reglas

Condiciones

Temporales

Variables

Combinadas

Acciones

6.16.5 General

6.16.6 Resultados

class tikon.datos.obs.**Obs**(*datos*)

Índice

O