Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Titulación:
Ingeniería de Organización Industrial

Alumno (nombre y apellidos):
Joan Mas de les Valls Ortiz

Título PFC:
Estudio y análisis de la escasez física y económica de agua dulce en la India en función de la oferta y de la demanda

Directora del PFC:
Mª Núria Salán Ballesteros

Profesora Colaboradora del PFC:
Beatriz Escribano Rodríguez de Robles

Convocatoria de entrega del PFC:
Enero 2011

Contenidos de este volumen:: **-ANEXOS-**

Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ESTUDIO Y ANÁLISIS DE LA ESCASEZ FÍSICA Y ECONÓMICA DE AGUA DULCE EN LA ÍNDIA EN FUNCIÓN DE LA OFERTA Y DE LA DEMANDA

-ANEXOS-

Autor: *Joan Mas de les Valls Ortiz*

Tutora: *Mª Núria Salán Ballesteros*

Profesora colaboradora: *Beatriz Escribano Rodríguez de Robles*

Enero 2011

ÍNDICE

ANEXO A. Glosario

Agua dulce: Agua superficial y subterránea. Según AQUASTAT, las aguas desalinizadas, reutilizadas y salinas utilizadas por diferentes sectores no se consideran agua dulce. Sin embargo, según el Glosario Hidrológico Internacional de la UNESCO, el agua dulce se define como Agua natural con una baja concentración de sales, o generalmente considerada adecuada, previo tratamiento, para producir agua potable.
Fuente: AQUASTAT, 2010; Glosario Hidrológico Internacional, UNESCO, 2010.

Agua extraída consumida: Agua extraída de cursos de agua para su utilización agrícola, industrial o doméstica, que se ha evaporado, transpirado, se ha incorporado a productos y cultivos, ha sido consumida por seres humanos o por el ganado, se ha vertido directamente al mar o a zonas de evaporación (cuenca ciega) o se ha extraído por otros medios de los recursos de agua dulce. Las pérdidas de agua durante el transporte entre el punto de extracción y el punto de uso no se incluyen.
Fuente: AQUASTAT, 2010

Bienestar humano: Es un estado que depende del contexto y de la situación, que comprende aspectos básicos para una buena vida: libertad y capacidad de elección, salud y bienestar corporal, buenas relaciones sociales, seguridad y tranquilidad de espíritu.
Fuente: "Evaluación de Ecosistemas del Milenio", 2001.

Consumo dinámico (CD): Volumen de agua que una vez extraída se evapora, transpira y queda incorporada en el producto final o en los cultivos. Incluye aquella agua que se incorpora a los cuerpos de los seres humanos o del ganado., e incluso aquella que va directamente al mar. Después de su uso no puede volver a utilizarse a corto plazo. Se asocia a perdidas irrecuperables dentro de los consumos internos urbanos e industriales. La unidad de tiempo es el ano y se visualiza a lo largo de un periodo.
Fuente: tesis doctoral "Metodología de análisis en el tiempo para evaluar la escasez de agua dulce en función de la oferta y de la demanda. Caso de estudio: Los países de la región del golfo de Guinea", Beatriz Escribano, 2010.

Demanda: Cantidad de una mercancía que los consumidores desean y pueden comprar a un precio dado en un determinado momento. La demanda, como concepto económico, no se equipara simplemente con el deseo o necesidad que

exista por un bien, sino que requiere además que los consumidores, o demandantes, tengan el deseo y la capacidad efectiva de pagar por dicho bien. La demanda total que existe en una economía se denomina *demanda agregada* y resulta un concepto importante en los análisis macroeconómicos.
Fuente: "Diccionario de Economía y Finanzas", Carlos Sabino, Ed. Panapo, Caracas, 1991.

Demanda de agua dulce: Volumen de agua dulce requerida para satisfacer las posibles necesidades socioeconómicas y ambientales de las actividades humanas en un periodo de tiempo determinado (normalmente un año).
Fuente: "Metodología de análisis en el tiempo para evaluar la escasez de agua dulce en función de la oferta y de la demanda. Caso de estudio: Los países de la región del golfo de Guinea", Beatriz Escribano, 2010

Demanda dinámica (DD): Volumen de agua dulce que se requiere para satisfacer las posibles necesidades socio-económicas para las actividades humanas en un periodo de tiempo determinado. Esta definición cualitativa se traduce cuantitativamente como el volumen de agua dulce renovable extraída de una fuente del medio natural para una actividad humana en una unidad de tiempo -el año- visualizada a lo largo de un periodo.
Fuente: tesis doctoral "Metodología de análisis en el tiempo para evaluar la escasez de agua dulce en función de la oferta y de la demanda. Caso de estudio: Los países de la región del golfo de Guinea", Beatriz Escribano, 2010.

Desarrollo sostenible: Desarrollo que satisface las necesidades de la generación presente sin comprometer la capacidad de las generaciones futuras para satisfacer sus propias necesidades (Comisión Mundial del Medio Ambiente y del Desarrollo, 1988).
El término desarrollo sostenible, perdurable o sustentable se aplica al desarrollo socio-económico y fue formalizado por primera vez en el documento conocido como Informe Brundtland (1987), fruto de los trabajos de la Comisión Mundial de Medio Ambiente y Desarrollo de Naciones Unidas, creada en Asamblea de las Naciones Unidas en 1983. Dicha definición se asume en el Principio 3.º de la Declaración de Río (1992)
El ámbito del desarrollo sostenible puede dividirse conceptualmente en tres partes: ambiental, económica y social. Se considera el aspecto social por la relación entre el bienestar social con el medio ambiente y la bonanza económica. El triple resultado es un conjunto de indicadores de desempeño de una organización en las tres áreas.

Deben satisfacerse las necesidades de la sociedad como alimentación, ropa, vivienda y trabajo, pues si la pobreza es habitual, el mundo estará encaminado a catástrofes de varios tipos, incluidas las ecológicas. Asimismo, el desarrollo y el bienestar social, están limitados por el nivel tecnológico, los recursos del medio ambiente y la capacidad del medio ambiente para absorber los efectos de la actividad humana. Ante esta situación, se plantea la posibilidad de mejorar la tecnología y la organización social de forma que el medio ambiente pueda recuperarse al mismo ritmo que es afectado por la actividad humana
Fuente: Diccionario de Acción Humanitaria y Cooperación al desarrollo, 2010.

Escasez: Es la insuficiencia de recursos y demás fundamentales para satisfacer las necesidades y/o gustos del ser humano, lo cual sostiene que una sociedad no posee los recursos o productividad necesarios para proveer de manera eficiente con sus necesidades a quienes habitan en dicha sociedad. La escasez es causada por varios factores que se clasifican en dos categorías: el incremento de demanda y la disminución o agotamiento de fuentes y/o recursos.
Fuente: "Diccionario de Economía y Finanzas", Carlos Sabino, Ed. Panapo, Caracas, 1991.

Escorrentía: Parte de las precipitaciones, nieve derretida o agua de riego que fluye por la superficie de la tierra y se puede devolver a las corrientes. La escorrentía puede recoger sustancias contaminantes del aire o la tierra y arrastrarlas hasta las aguas en las que se vierte.
Fuente: AQUASTAT, 2010.

Extracción total de agua (suma de sectores): Cantidad de agua extraída cada año para usos agropecuarios, industriales y municipales. Incluye los recursos renovables de agua dulce así como la posible sobre-extracción de aguas subterráneas renovables, aguas subterráneas fósiles, aguas desalinizadas y aguas residuales tratadas. No incluye usos caracterizados por una tasa de consumo neto de agua muy baja, como por ejemplo la hidroelectricidad, la recreación, navegación, pesca de captura (agua dulce), etc.
Fuente: AQUASTAT, 2010.

Extracción de agua agrícola: Cantidad de agua extraída cada año para riego y usos ganaderos. Incluye los recursos renovables de agua dulce así como el potencial de extracción de aguas subterráneas renovables y de extracción de aguas subterráneas fósiles, la utilización de aguas de drenaje para usos agrícolas, aguas desalinizadas y aguas residuales tratadas. Incluye las aguas utilizadas para riego y para abrevar el ganado aunque algunos países incluyen

esta última categoría en la extracción de agua para usos municipales. En lo que respecta a la utilización de agua para riego, el valor es muy superior al consumo del riego, debido a las pérdidas de agua durante su distribución desde la fuente hasta los cultivos. El término "Razón de necesidad de agua" (a veces llamado "eficiencia de riego") se utiliza para hacer referencia a la proporción entre la necesidad neta de agua para riego o la necesidad de agua de los cultivos, que es el volumen de agua necesario para compensar el déficit entre la evapotranspiración potencial y las precipitaciones reales durante el período de cultivo, y la cantidad de agua extraída para riego, incluidas las pérdidas. En el caso concreto del riego del arroz cáscara, hace falta más agua para inundar los campos, facilitar la preparación de la tierra y proteger las plantas. En ese caso, las necesidades de agua para riego son la suma del déficit de precipitaciones y el agua necesaria para inundar los campos. En el ámbito de los perímetros, la proporción de necesidad de agua puede variar entre menos del 20 % y más del 95 %. En lo que respecta al agua para usos ganaderos, la proporción entre el consumo neto y la extracción de agua se estima entre el 60 y el 90 %.
Fuente: AQUASTAT, 2010.

Extracción de agua industrial: Cantidad de agua extraída cada año para usos industriales. Incluye los recursos hídricos renovables así como la posible sobre-extracción de aguas subterráneas renovables, aguas subterráneas fósiles, aguas desalinizadas y aguas residuales tratadas. Normalmente, hace referencia al autosuministro de industrias que no están conectadas a ninguna red de distribución. Se estima que la proporción entre el consumo neto y la extracción es inferior al 5%. Incluye agua utilizada para enfriamiento de plantas termoeléctricas y nucleares, pero no el uso de plantas hidroeléctricas, ni usos con bajo consumo de agua.
Fuente: AQUASTAT, 2010.

Extracción de agua municipal: Cantidad de agua extraída cada año principalmente para su uso directo por parte de la población. Incluye los recursos renovables de agua dulce así como la posible sobre-extracción de aguas subterráneas renovables y aguas subterráneas fósiles y el uso potencial de aguas desalinizadas y aguas residuales tratadas. Normalmente se contabiliza como la cantidad total de agua retirada por la red pública de distribución. Puede incluir la parte que utilizan las industrias conectadas a la red municipal. La proporción entre el consumo neto y la extracción de agua puede variar entre el 5 y el 15 % en las zonas urbanas y entre el 10 y el 50 % en las zonas rurales.
Fuente: AQUASTAT, 2010

Índice de Desarrollo Humano (IDH): Se trata de un resumen de las mediciones del desarrollo humano. Mide el promedio de los logros de un país en tres dimensiones fundamentales del desarrollo humano: 1) vida larga y sana, medida mediante la esperanza de vida al nacer; 2) conocimiento, medido mediante la tasa de alfabetización de la población adulta (ponderación de dos tercios) y la tasa de matriculación bruta en educación primaria, secundaria y terciaria (un tercio); 3) condiciones de vida decentes, medidas mediante el PIB per cápita.
Fuente: UNDP, 2010

Índice de Pobreza Multidimensional (IPM): Se trata de un índice de pobreza estadístico sobre la situación de las personas por países, elaborado desde 2010 que sustituye a indicadores de pobreza anteriores (IPH1 e IPH2). Está elaborado por el Programa de las Naciones Unidas para el Desarrollo (PNUD-ONU) en colaboración con la Oxford Poverty & Human Development Initiative (OPHI) de la Universidad de Oxford y se presenta en el 20 aniversario del Informe Anual Mundial sobre el Desarrollo Humano del PNUD.

Los siguientes 10 indicadores (agrupados en los 3 aspectos básicos) en se usan para calcular el Índice de pobreza multidimensional:

Educación: (Ponderación de los parámetros 1 y 2 de 1/6)

1. Años de escolarización: sin acceso si ningún miembro del hogar ha completado cinco años de escolaridad
2. Niños escolarizados: sin acceso si los niños en edad escolar no asisten a la escuela

Asistencia sanitaria – Salud: (Ponderación de los parámetros 3 y 4 de 1/6)

3. Mortalidad infantil: si un niño ha muerto en la familia
4. Nutrición: sin acceso si un adulto o niño está desnutrido

Calidad de vida - bienestar social: (Ponderación los parámetros 5 al 10 de 1/18)

5. Electricidad: sin acceso si el hogar no tiene electricidad
6. Saneamiento: sin acceso no tienen un baño con condiciones suficientes o si su baño es compartido (según la definición MDG)
7. Agua potable: sin acceso si el hogar no tiene acceso a agua potable o el agua potable está a más de 30 minutos andando desde el hogar (Definición MDG)
8. Suelo: Sin acceso si el piso del hogar tiene suciedad, es de arena, tierra o estiércol

9. Combustible de hogar: sin acceso si se cocina con leña, carbón o estiércol
10. Bienes: sin acceso si el hogar no tiene más de uno de los siguientes bienes: radio, televisión, teléfono, bicicleta o moto

Una persona se considera pobre no tiene acceso en al menos 30% de los indicadores ponderados. La intensidad de la pobreza indica la proporción de los indicadores a los que no se tiene acceso.

Fuente: Oxford Poverty & Human Development Initiative (OPHI).

Oferta: La cantidad de una mercancía o servicio que entra en el mercado a un precio dado en un momento determinado. La oferta es, por lo tanto, una cantidad concreta, bien especificada en cuanto al precio y al período de tiempo que cubre, y no una capacidad potencial de ofrecer bienes y servicios. La *ley de la oferta* establece básicamente que cuanto mayor sea el precio mayor será la cantidad de bienes y servicios que los oferentes están dispuestos a llevar al mercado, y viceversa; cuanto mayor sea el período de tiempo considerado, por otra parte, más serán los productores que tendrán tiempo para ajustar su producción para beneficiarse del precio existente. La *curva de oferta*, esquemáticamente mostrada a continuación, expresa la relación básica que se establece entre ésta y el precio.

Fuente: "Diccionario de Economía y Finanzas", Carlos Sabino, Ed. Panapo, Caracas, 1991.

Oferta de agua dulce: Volumen de agua dulce susceptible de ser utilizado por el ser humano de forma directa con la incorporación de más o menos tecnología en un periodo de tiempo determinado, respetando el medioambiente de la zona estudiada.

Fuente: "Metodología de análisis en el tiempo para evaluar la escasez de agua dulce en función de la oferta y de la demanda. Caso de estudio: Los países de la región del golfo de Guinea", Beatriz Escribano, 2010

Oferta de nivel alto (ONA): La oferta de nivel alto es el 60% del volumen total de recursos hídricos renovables. Este volumen representa los recursos hidricos disponibles para el desarrollo y procedentes de todas las fuentes naturales de cada país. La unidad de este volumen estimado se puede expresar en km3 /ano, o si se divide entre la población del país, en m3/ persona y año, o también se puede expresar en porcentaje si se quiere hacer referencia a lo que representa este volumen en relación con el volumen de su región, de su continente o del mundo.

Fuente: "Metodología de análisis en el tiempo para evaluar la escasez de agua dulce en función de la oferta y de la demanda. Caso de estudio: Los países de la región del golfo de Guinea", Beatriz Escribano, 2010

Oferta de nivel bajo (ONB): La oferta de nivel bajo es la suma del 40% de los recursos totales en aguas superficiales y del 30% de los recursos en aguas subterráneas. La unidad de este volumen estimado se puede expresar en km3/año, o si se divide entre la población del país, en m3/ persona y año, o también se puede expresar en porcentaje si se quiere hacer referencia a lo que representa este volumen en relación con el volumen de su región, de su continente o del mundo.
Fuente: "Metodología de análisis en el tiempo para evaluar la escasez de agua dulce en función de la oferta y de la demanda. Caso de estudio: Los países de la región del golfo de Guinea", Beatriz Escribano, 2010

Población con acceso al agua potable saludable: Porcentaje de la población total que utiliza fuentes mejoradas de agua. Una fuente "mejorada" es la que probablemente suministrará agua potable, como la toma en el hogar, los pozos entubados, etc. La información de que se dispone actualmente aún no permite establecer una relación entre el acceso al agua potable inocua y a fuentes mejoradas, pero la OMS y el UNICEF están estudiando esta relación.
Fuente: WHO/UNICEF Joint Monitoring Programme (JMP) for Water Supply and Sanitation.

Recursos hídricos: Total de los recursos hídricos renovables reales: es la suma de los recursos hídricos renovables internos y los recursos hídricos renovables externos reales. Corresponde a la cantidad máxima teórica de agua disponible realmente cada año para un país en un momento determinado.
Fuente: AQUASTAT, 2010

Satisfacción con la calidad del agua: Porcentaje de entrevistados que responden "sí" a la pregunta de la Encuesta Mundial *Gallup*: "En la ciudad o zona donde vive ¿está satisfecho con la calidad del agua?".
Fuente: Informe sobre Desarrollo Humano 2010 de las Naciones Unidas.

ANEXO B. Lista de abreviaturas y símbolos

AQUASTAT: Sistema de información de la FAO sobre el agua y la agricultura

BaU: Business as usual

CD: Consumo dinámico

CIA: Agencia Central de Inteligencia (EE.UU.)

DD: Demanda dinámica

FAO: Organización de las Naciones Unidas para la Agricultura y Alimentación

FMI: Fondo Monetario Internacional

IDH: Índice de desarrollo humano

IPM: Índice de pobreza multidimensional

IHP: Programa hidrológico internacional

IPCC: Panel Intergubernamental de expertos sobre el Cambio Climático

IWMI: Instituto Internacional para el Manejo del Agua

OCDE: Organización para la Cooperación y el Desarrollo Económico

ODM: Objetivos de desarrollo del milenio

ONA: Oferta de nivel alto de agua dulce

ONB: Oferta de nivel bajo de agua dulce

ONU: Organización de las Naciones Unidas

PIB: Producto interior bruto

PNUD: Programa de las Naciones Unidas para el desarrollo

UNDESA: Departamento de asuntos económicos y sociales de las Naciones Unidas

UNESCO: Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura

UNEP: Programa de las Naciones Unidas para el medio ambiente

UNPD: División de población de las Naciones Unidas

WWAP: Programa mundial de evaluación de los recursos hídricos

ANEXO C. Figuras auxiliares

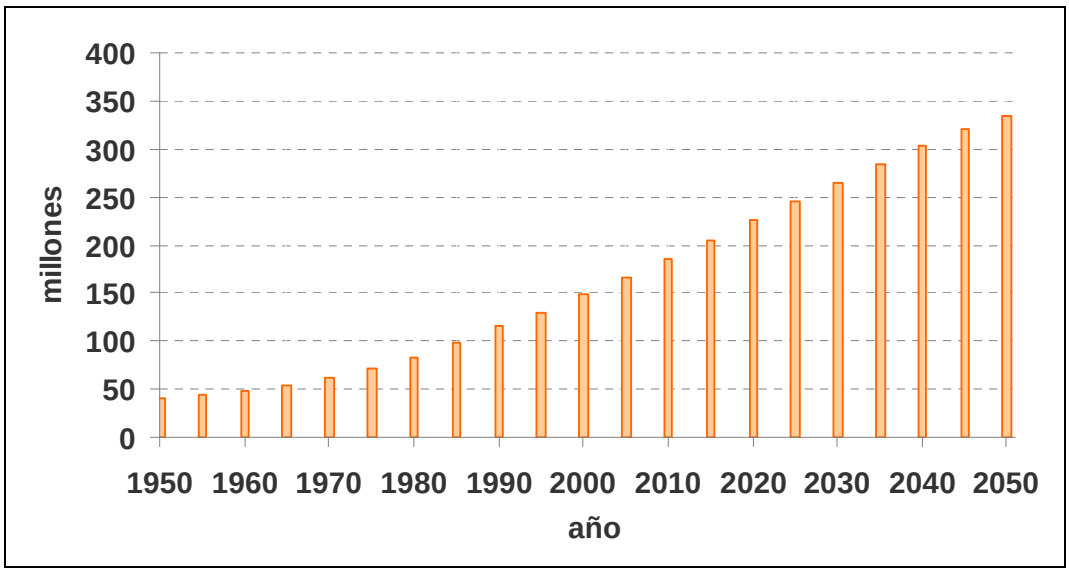

Figura C.1. Estimación de la población de Pakistán (1950-2050)

Fuente: UNITED NATIONS – WORLD POPULATION PROSPECTS, THE 2008 REVISION

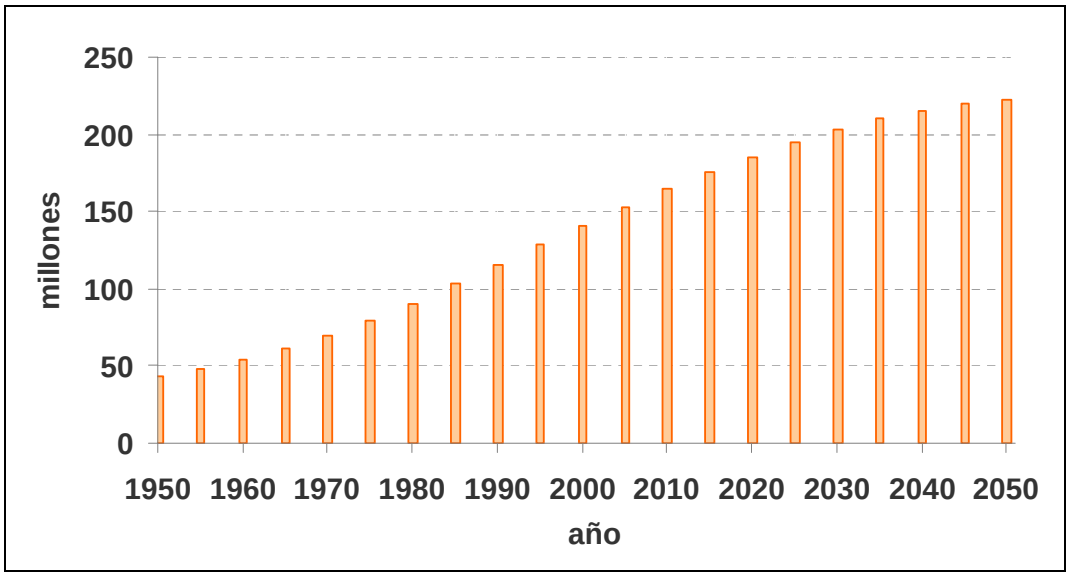

Figura C.2. Estimación de la población de Bangladesh (1950-2050)

Fuente: UNITED NATIONS – WORLD POPULATION PROSPECTS, THE 2008 REVISION

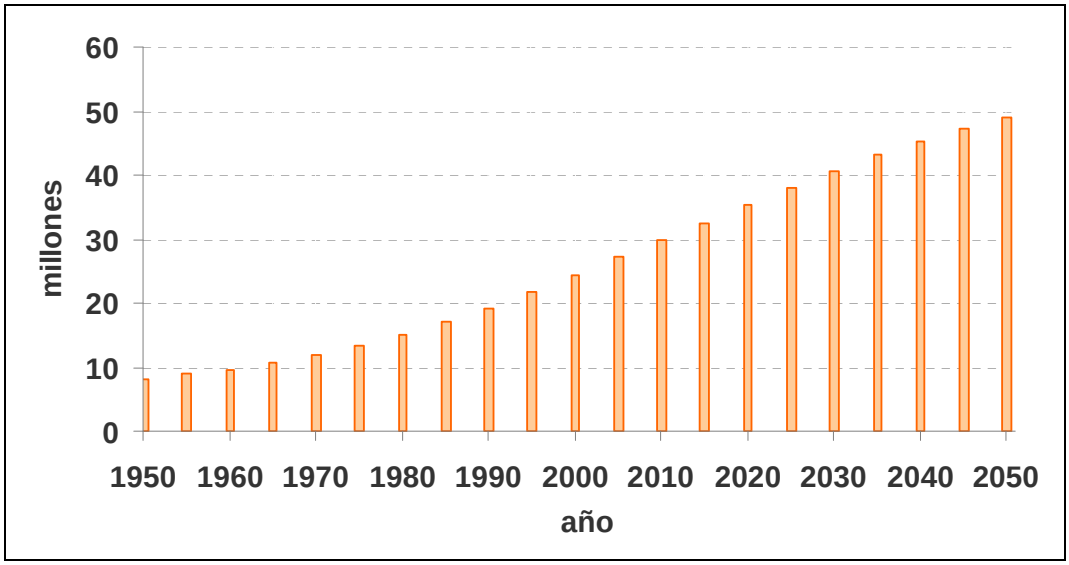

Figura C.3. Estimación de la población de Nepal (1950-2050)

Fuente: UNITED NATIONS – WORLD POPULATION PROSPECTS, THE 2008 REVISION

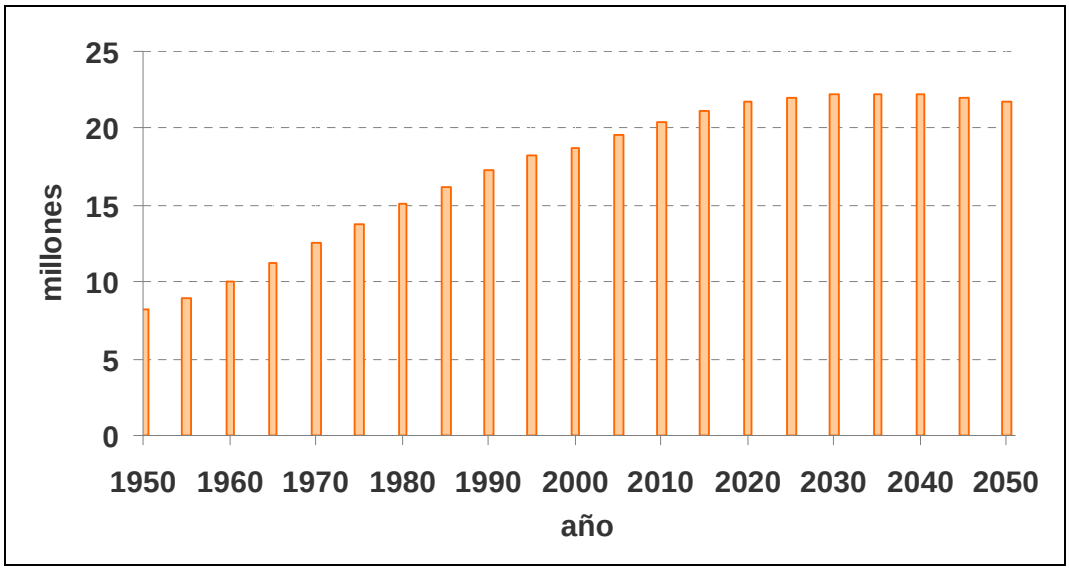

Figura C.4. Estimación de la población de Sri Lanka (1950-2050)

Fuente: UNITED NATIONS – WORLD POPULATION PROSPECTS, THE 2008 REVISION

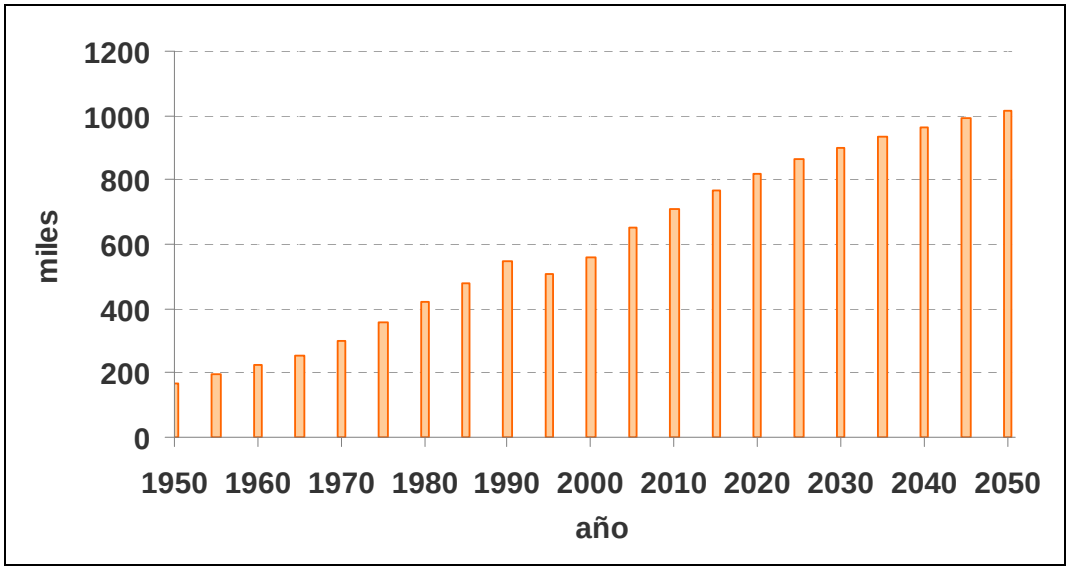

Figura C.5. Estimación de la población de Buhtán (1950-2050)

Fuente: UNITED NATIONS – WORLD POPULATION PROSPECTS, THE 2008 REVISION

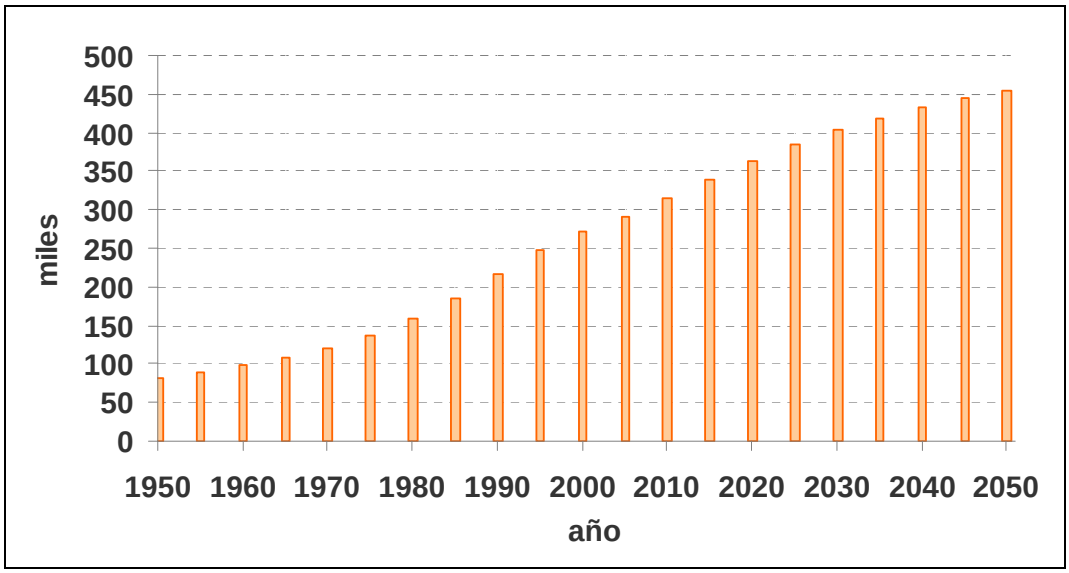

Figura C.6. Estimación de la población de Maldivas (1950-2050)

Fuente: UNITED NATIONS – WORLD POPULATION PROSPECTS, THE 2008 REVISION

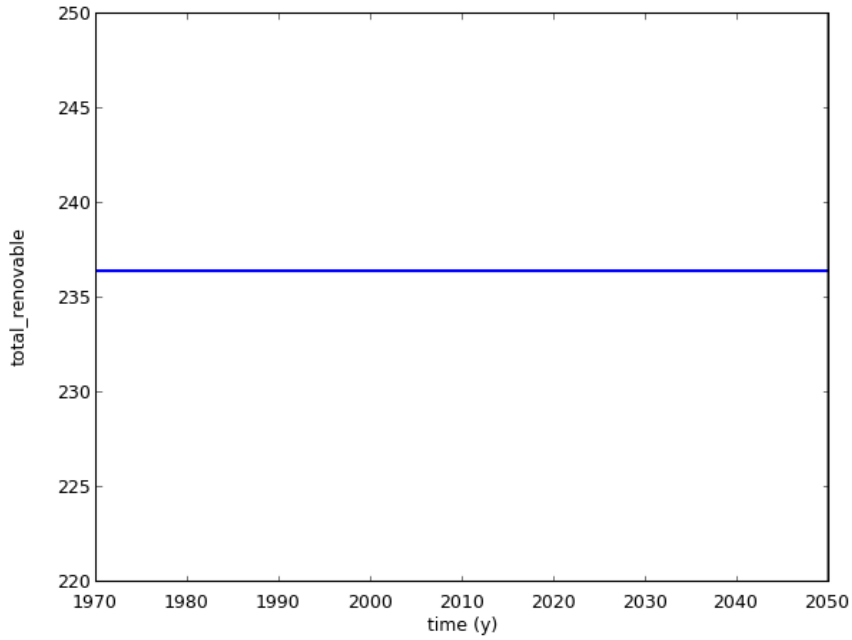

Figura C.7. Escenario business as usual del agua total renovable en Pakistán en el periodo 1970-2050. Fuente: Programa PySight

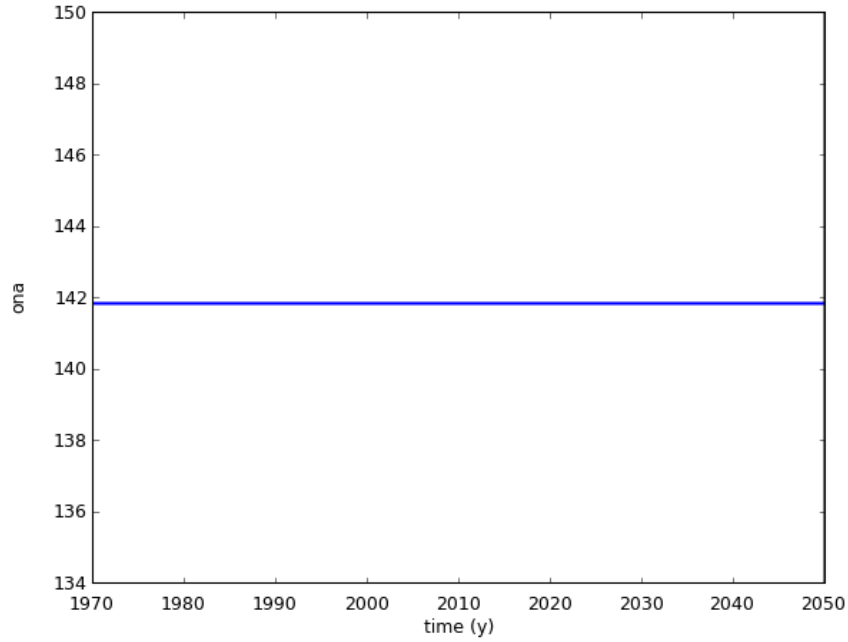

Figura C.8. Escenario business as usual de la Oferta de Nivel Alto (ONA) en Pakistán en el periodo 1970-2050. Fuente: Programa PySight

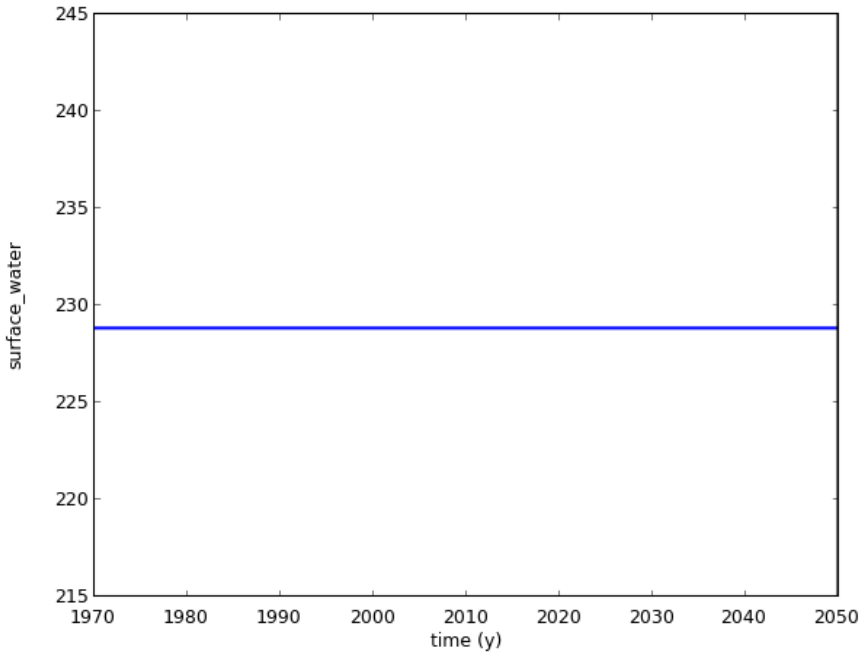

Figura C.9. Escenario business as usual del agua superficial en Pakistán en el periodo 1970-2050. Fuente: Programa PySight

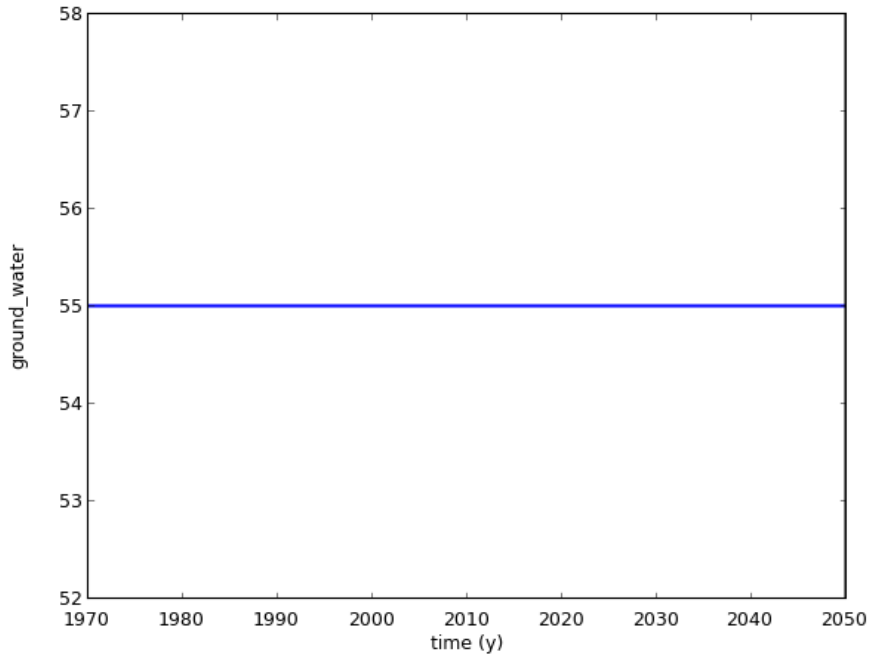

Figura C.10. Escenario business as usual del agua subterránea en Pakistán en el periodo 1970-2050. Fuente: Programa PySight

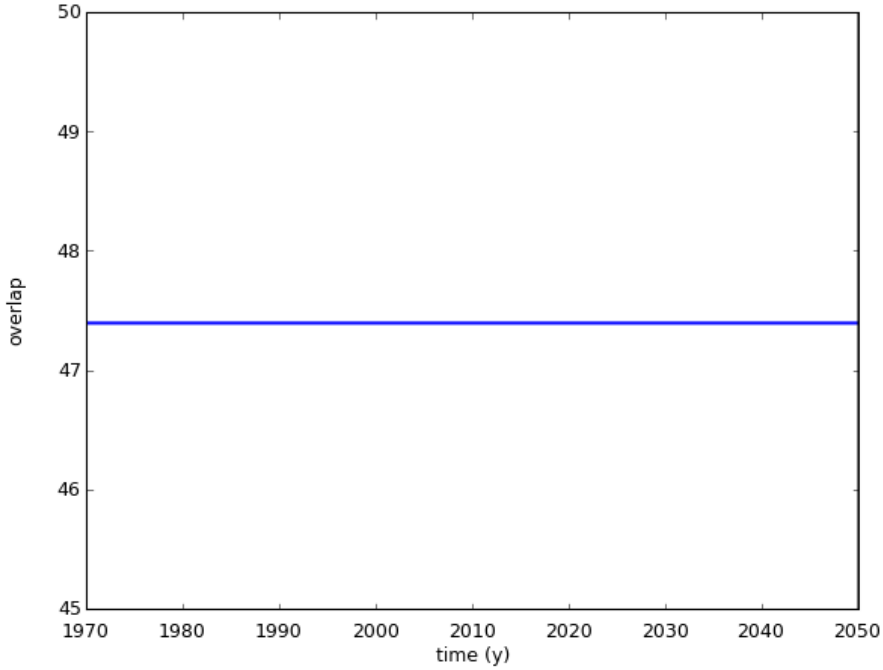

Figura C.11. Escenario business as usual del agua de solapamiento en Pakistán en el periodo 1970-2050. Fuente: Programa PySight

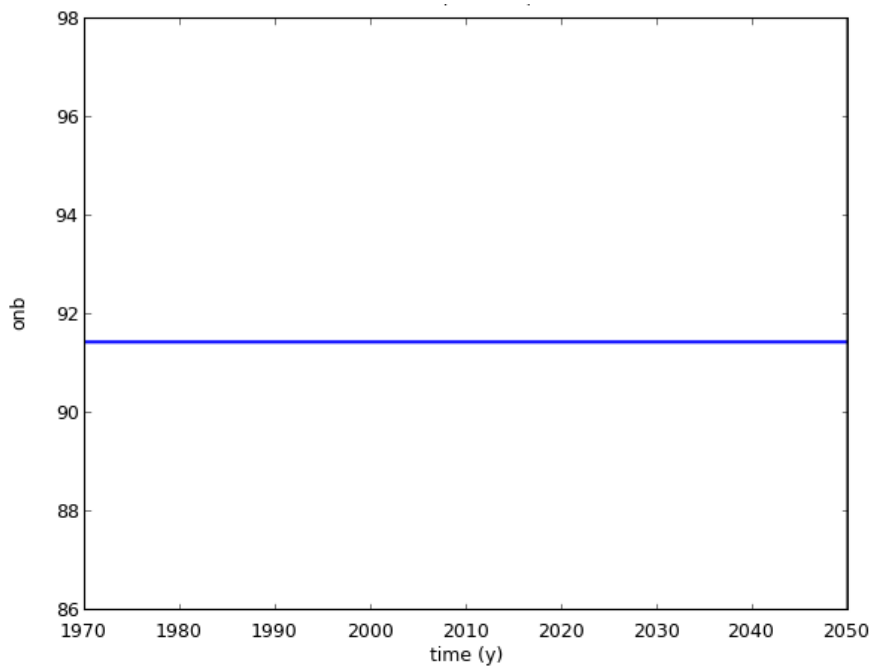

Figura C.12. Escenario business as usual de la Oferta de Nivel Bajo (ONB) en Pakistán en el periodo 1970-2050. Fuente: Programa PySight

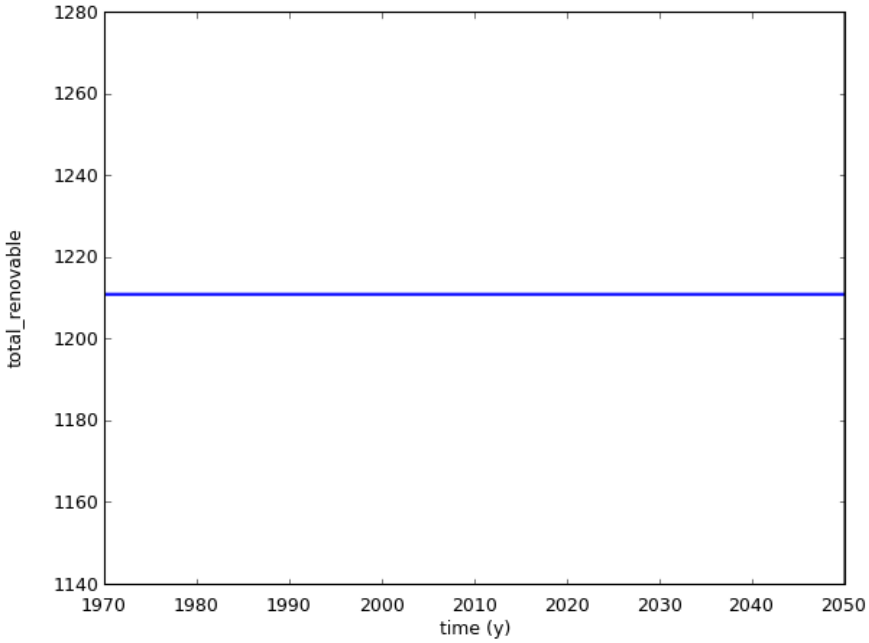

Figura C.13. Escenario business as usual del agua total renovable en Bangladesh en el periodo 1970-2050. Fuente: Programa PySight

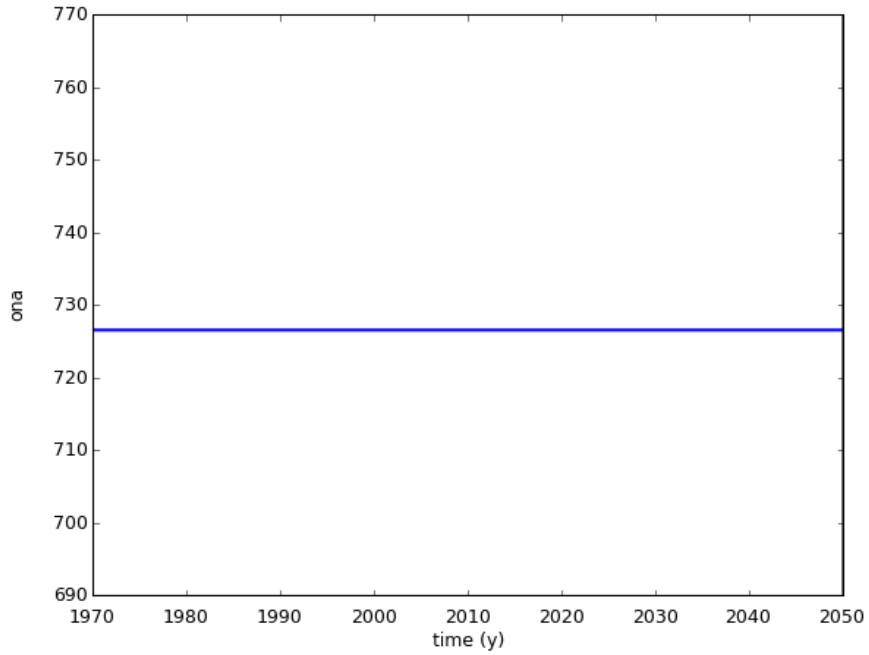

Figura C.14. Escenario business as usual de la Oferta de Nivel Alto (ONA) en Bangladesh en el periodo 1970-2050. Fuente: Programa PySight

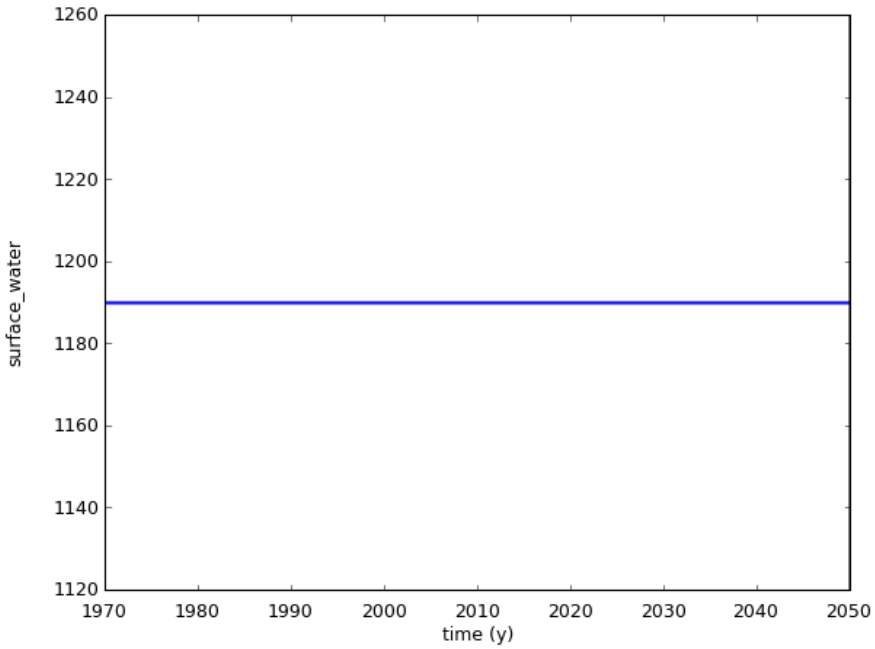

Figura C.15. Escenario business as usual del agua superficial en Bangladesh en el periodo 1970-2050. Fuente: Programa PySight

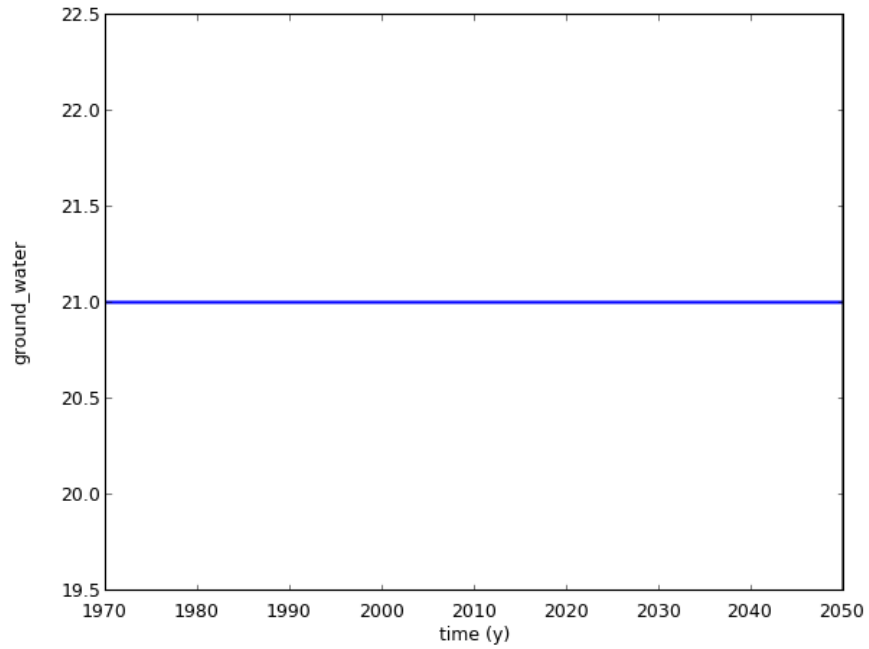

Figura C.16. Escenario business as usual del agua subterránea en Bangladesh en el periodo 1970-2050. Fuente: Programa PySight

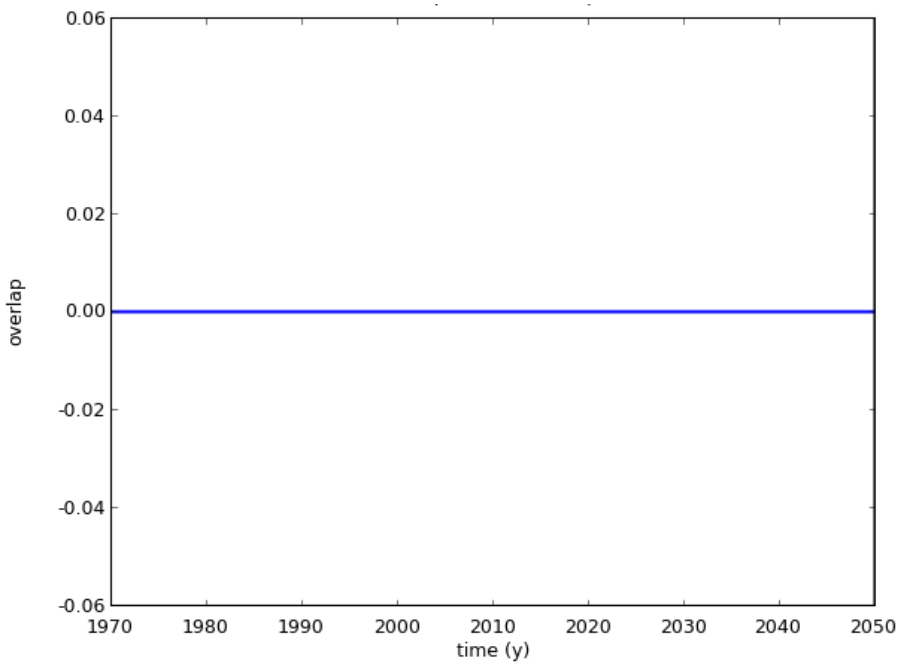

Figura C.17. Escenario business as usual del agua de solapamiento en Bangladesh en el periodo 1970-2050. Fuente: Programa PySight

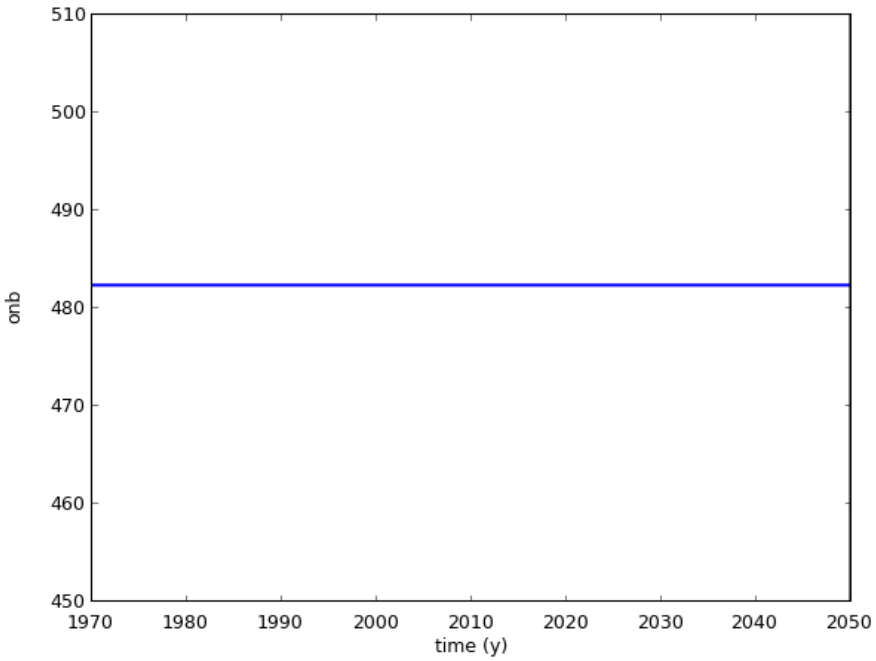

Figura C.18. Escenario business as usual de la Oferta de Nivel Bajo (ONB) en Bangladesh en el periodo 1970-2050. Fuente: Programa PySight

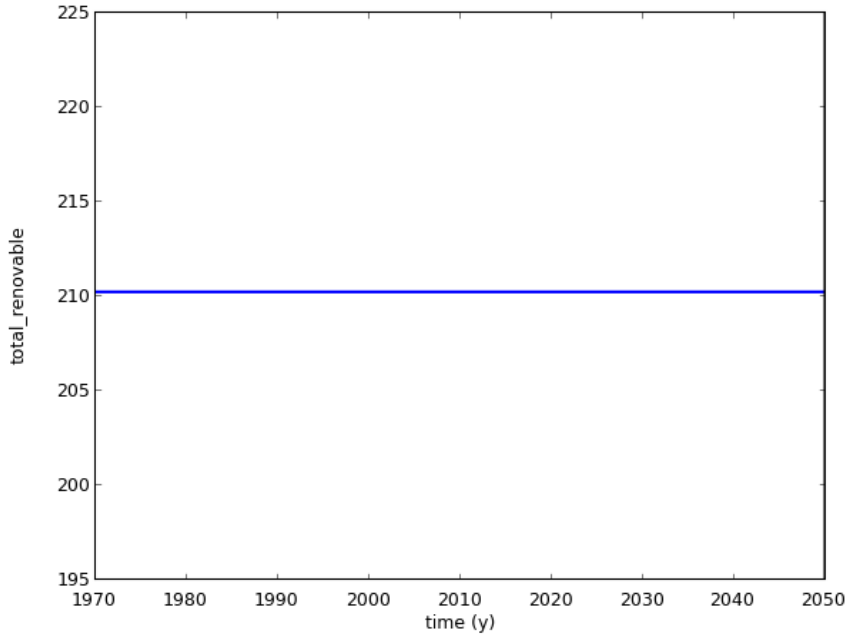

Figura C.19. Escenario business as usual del agua total renovable en Nepal en el periodo 1970-2050. Fuente: Programa PySight

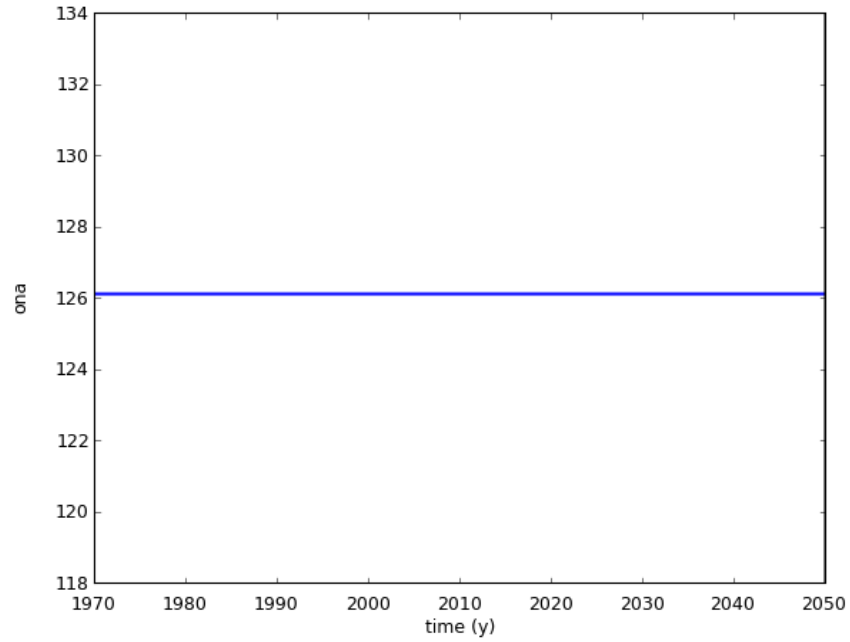

Figura C.20. Escenario business as usual de la Oferta de Nivel Alto (ONA) en Nepal en el periodo 1970-2050. Fuente: Programa PySight

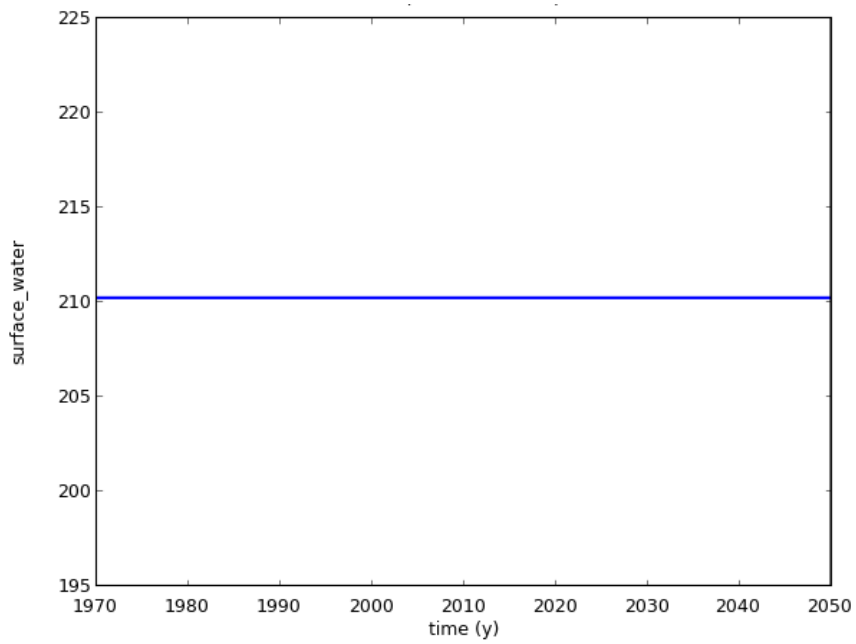

Figura C.21. Escenario business as usual del agua superficial en Nepal en el periodo 1970-2050. Fuente: Programa PySight

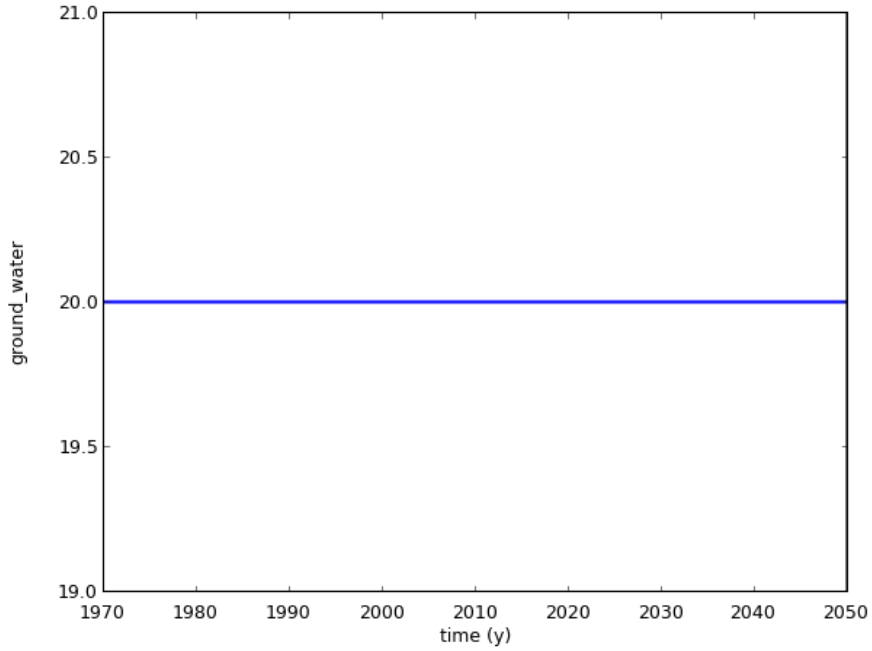

Figura C.22. Escenario business as usual del agua subterránea en Nepal en el periodo 1970-2050. Fuente: Programa PySight

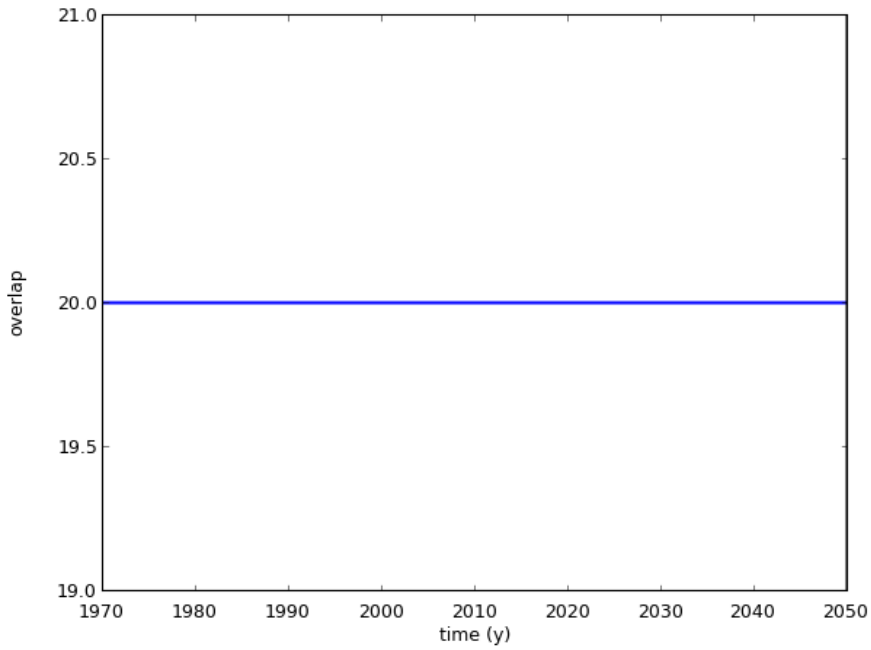

Figura C.23. Escenario business as usual del agua de solapamiento en Nepal en el periodo 1970-2050. Fuente: Programa PySight

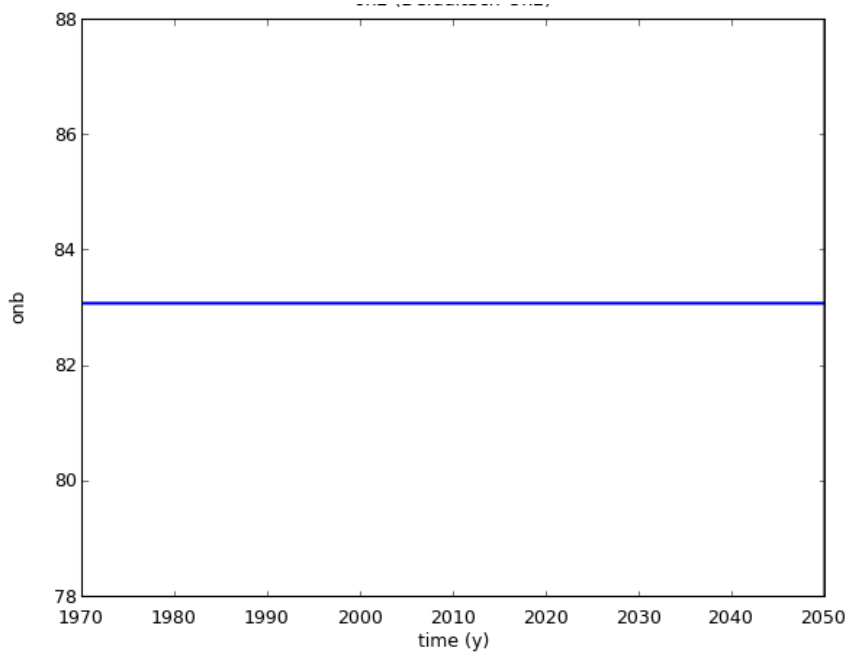

Figura C.24. Escenario business as usual de la Oferta de Nivel Bajo (ONB) en Nepal en el periodo 1970-2050. Fuente: Programa PySight

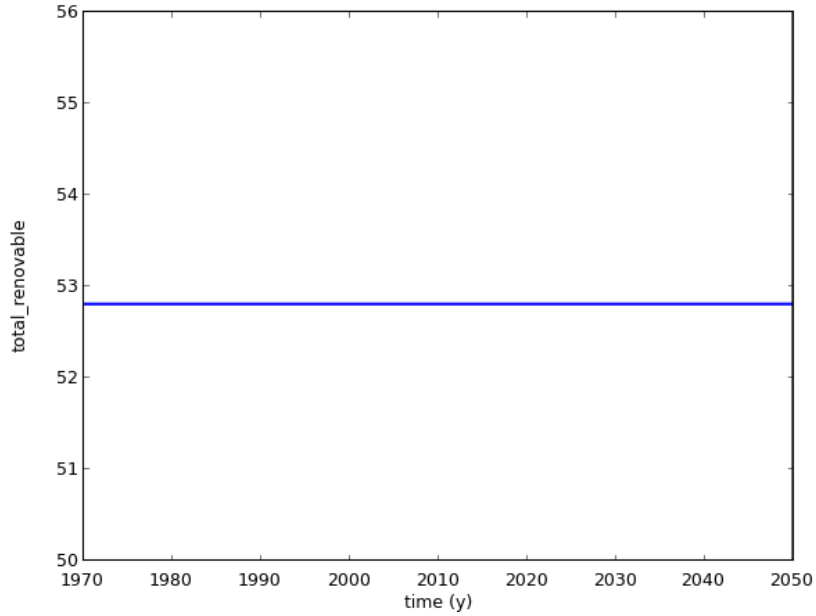

Figura C.25. Escenario business as usual del agua total renovable en Sri Lanka en el periodo 1970-2050. Fuente: Programa PySight

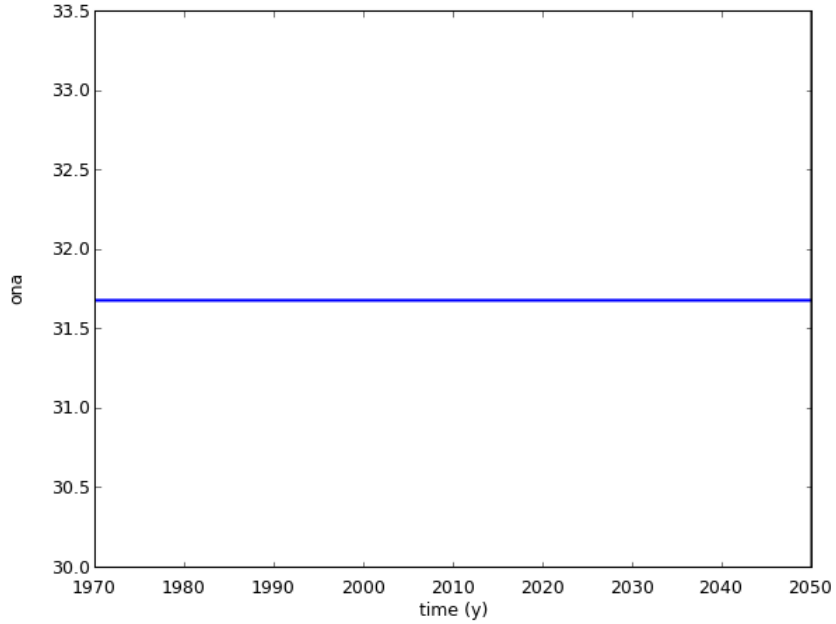

Figura C.26. Escenario business as usual de la Oferta de Nivel Alto (ONA) en Sri Lanka n en el periodo 1970-2050. Fuente: Programa PySight

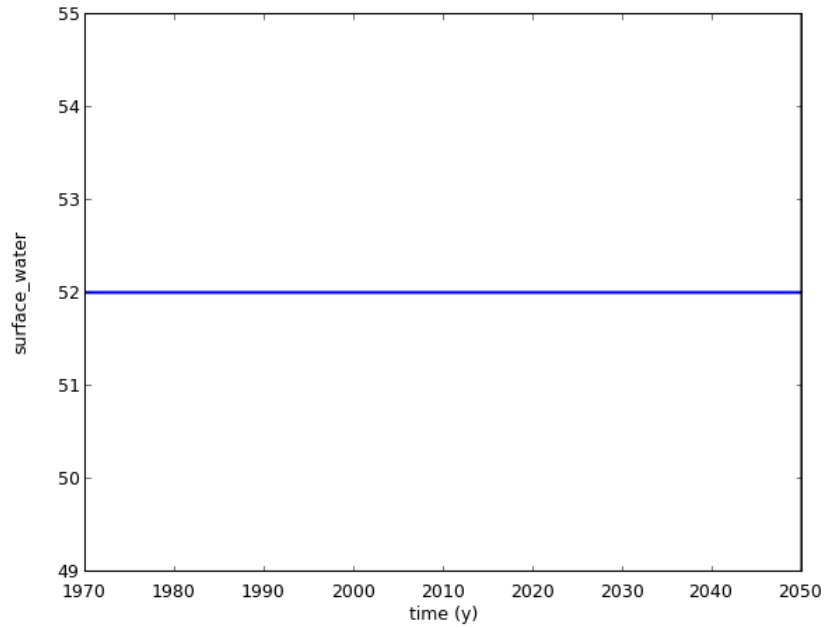

Figura C.27. Escenario business as usual del agua superficial en Sri Lanka en el periodo 1970-2050. Fuente: Programa PySight

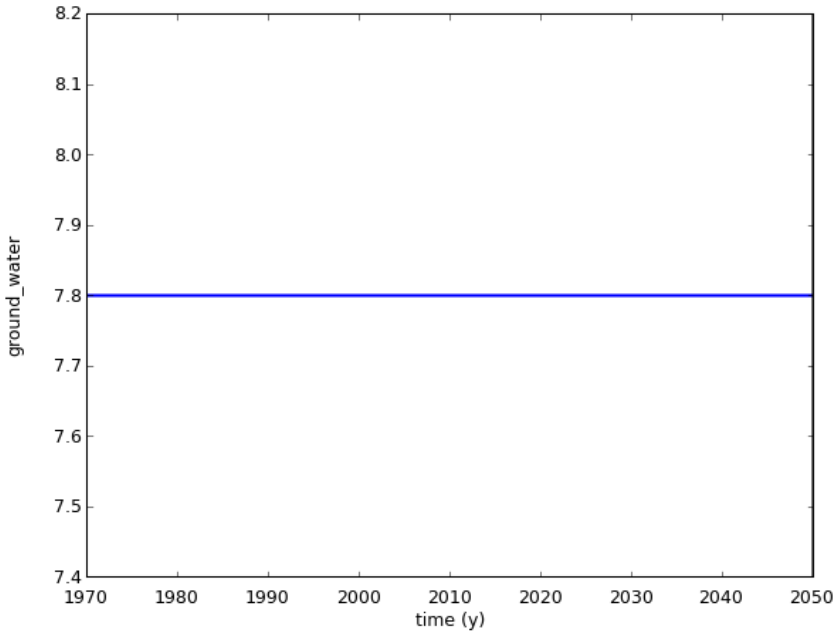

Figura C.28. Escenario business as usual del agua subterránea en Sri Lanka en el periodo 1970-2050. Fuente: Programa PySight

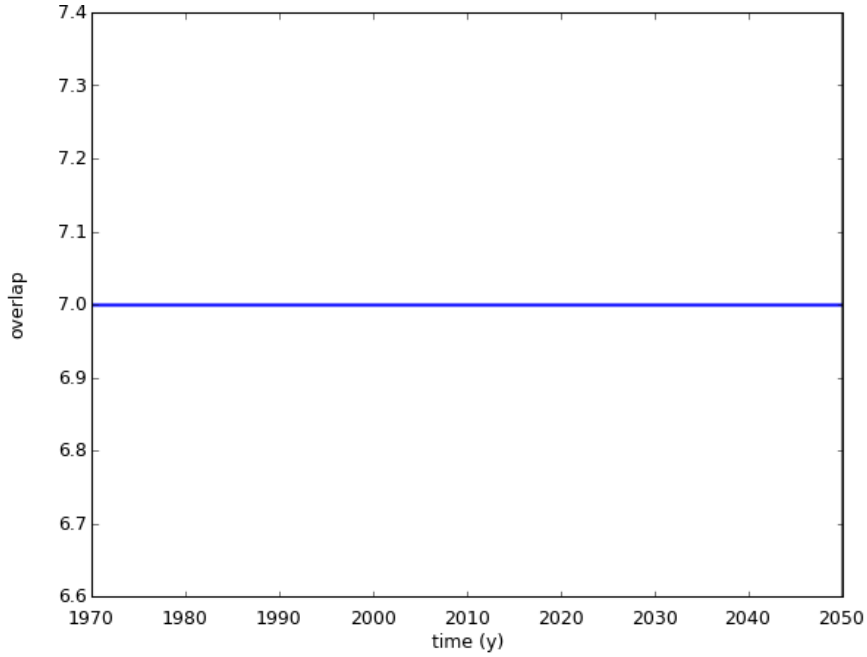

Figura C.29. Escenario business as usual del agua de solapamiento en Sri Lanka en el periodo 1970-2050. Fuente: Programa PySight

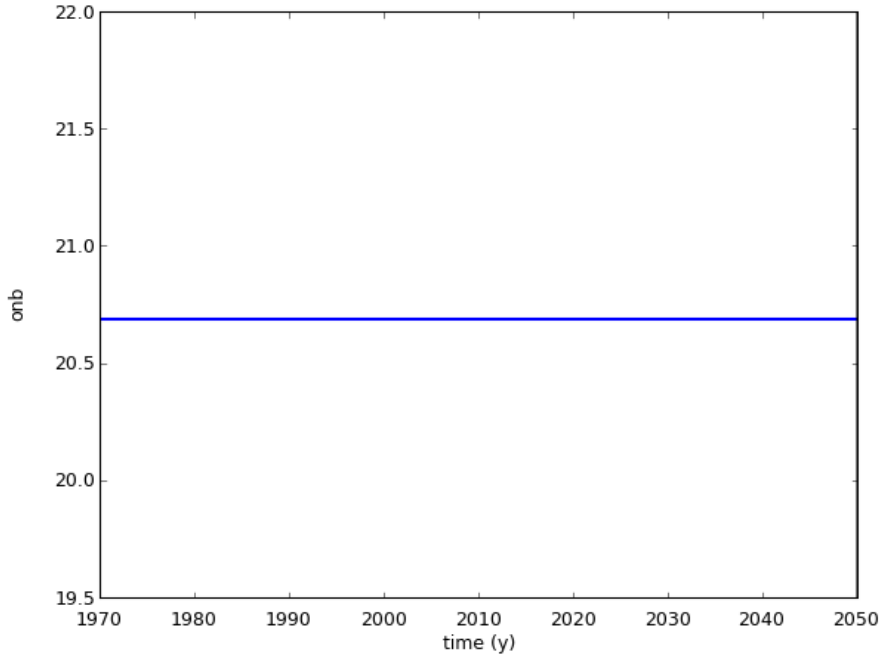

Figura C.30. Escenario business as usual de la Oferta de Nivel Bajo (ONB) en Sri Lanka en el periodo 1970-2050. Fuente: Programa PySight

ANEXO D. El programa PySight

wxPySight.py

```
import imp
import string
import sys
import wx
import wx.grid

from matplotlib import use
use('wxagg')
INK = 'black'
PAPER = 'white'

from matplotlib import rcParams
rcParams['figure.facecolor'] = PAPER
rcParams['figure.edgecolor'] = INK
rcParams['axes.edgecolor'] = INK
rcParams['axes.labelcolor'] = INK
rcParams['xtick.color'] = INK
rcParams['ytick.color'] = INK
rcParams['text.color'] = INK
colorList =
['b','g','r','c','m','y','k','0.75','chartreuse','burlywood']

import matplotlib
from matplotlib.backends.backend_wxagg import Toolbar,
FigureCanvasWxAgg, FigureManager
from matplotlib.figure import Figure
from matplotlib.ticker import FormatStrFormatter
from numpy import *

ID_LB1 = 70
ID_LB2 = 71
ID_OPEN = 101
ID_IMPORT = 104
ID_EXPORT = 105
ID_PRINT = 106
ID_EXIT = 109
ID_DEFAULT = 111
ID_LOAD = 112
ID_SAVE = 113
ID_DEL = 114
ID_DUPE = 115
ID_EDIT_SCN = 116
ID_EDIT_VAR = 121
ID_CALC_VAR = 122
ID_OPT_PLOT = 131
ID_ABOUT = 191
ID_TOGGLE = 201

LINEAL_INTERPOLATION = 1
LINEAL_REGRESSION = 2
EXPONENTIAL_REGRESSION = 3

class EditScnDialog(wx.Dialog):
```

```
def __init__(self, parent, ID, title, name, initial_year,
final_year, pos=wx.DefaultPosition, size=wx.DefaultSize,
style=wx.DEFAULT_DIALOG_STYLE):

wx.Dialog.__init__(self, parent, ID, title, pos, size, style)

self.name = name
self.initial_year = initial_year
self.final_year = final_year
sizer = wx.BoxSizer(wx.VERTICAL)
label = wx.StaticText(self, -1, "Scenario name:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text1 = wx.TextCtrl(self, -1, self.name, size=(80,-1))
sizer.Add(self.text1, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
label = wx.StaticText(self, -1, "First year:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text2 = wx.TextCtrl(self, -1, str(self.initial_year),
size=(80,-1))
sizer.Add(self.text2, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
label = wx.StaticText(self, -1, "Last year:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text3 = wx.TextCtrl(self, -1, str(self.final_year),
size=(80,-1))
sizer.Add(self.text3, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
btnsizer = wx.StdDialogButtonSizer()
btn = wx.Button(self, wx.ID_OK)
btn.SetDefault()
btnsizer.AddButton(btn)
btn = wx.Button(self, wx.ID_CANCEL)
btnsizer.AddButton(btn)
btnsizer.Realize()
sizer.Add(btnsizer, 0, wx.ALIGN_CENTER_VERTICAL|wx.ALL, 5)
self.SetSizer(sizer)
sizer.Fit(self)
wx.EVT_BUTTON(self, wx.ID_OK, self.EvtOK)

def EvtOK(self, evt):
self.name = self.text1.GetValue()

try:
self.initial_year = int(self.text2.GetValue())
self.final_year = int(self.text3.GetValue())

except ValueError :
d = wx.MessageDialog(self, " Years should have integer
values", "ValueError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()

else:
self.EndModal(wx.ID_OK)

class EditVarDialog(wx.Dialog):

def __init__(self, parent, ID, title, initial, tipf,
tabl=LINEAL_INTERPOLATION, pos=wx.DefaultPosition,
size=wx.DefaultSize, style=wx.DEFAULT_DIALOG_STYLE):
wx.Dialog.__init__(self, parent, ID, title, pos, size, style)
self.imax = 50
```

```
if type(initial) is list:
initial_string = ""
initial_table = initial
self.enable_text = False

else:
initial_string = str(initial)
initial_table = []
self.enable_text = True
self.tabl = tabl
sizer = wx.FlexGridSizer(6,1)
label = wx.StaticText(self, -1, "Constant value")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text = wx.TextCtrl(self, -1, initial_string, size=(120,23))
self.text.Enable(self.enable_text)
sizer.Add(self.text, 1, wx.ALIGN_CENTRE|wx.SHAPED, 5)
label = wx.StaticText(self, -1, "Table values")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.grid = wx.grid.Grid(self, -1, size=(265,240))
self.grid.CreateGrid(self.imax,2)

for i in xrange(len(initial_table)):
self.grid.SetCellValue(i,0,str(initial_table[i][0]))
self.grid.SetCellValue(i,1,str(initial_table[i][1]))
self.grid.Enable(not self.enable_text)
sizer.Add(self.grid, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
ipmet = []
self.ipkey = []
lbs = 0

for i in tipf.items():
ipmet.append(i[1][1])
self.ipkey.append(i[0])

if i[0] == self.tabl:
lbs = len(self.ipkey)-1
self.lb = wx.ListBox(self, -1, (20, 40), (90, 70), ipmet,
wx.LB_SINGLE)
self.lb.SetSelection(lbs)
self.lb.Enable(not self.enable_text)
sizer.Add(self.lb, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
btnsizer = wx.StdDialogButtonSizer()
btn = wx.Button(self, wx.ID_OK)
btn.SetDefault()
btnsizer.AddButton(btn)
btn = wx.Button(self, wx.ID_CANCEL)
btnsizer.AddButton(btn)
btnsizer.Realize()
sizer.Add(btnsizer, 0, wx.ALIGN_CENTER_VERTICAL|wx.ALL, 5)
self.SetSizer(sizer)
sizer.Fit(self)
wx.EVT_BUTTON(self, wx.ID_OK, self.EvtOK)
wx.EVT_LEFT_DCLICK(self, self.EvtToggle)

def EvtOK(self, evt):

if self.enable_text:

try:
self.initial = float(self.text.GetValue())

except ValueError :
d = wx.MessageDialog(self, " All numbers should have float
values","ValueError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()
```

```
else:
self.EndModal(wx.ID_OK)

else:
self.initial = []

try:

for i in xrange(self.imax):
xs = self.grid.GetCellValue(i, 0)
ys = self.grid.GetCellValue(i, 1)

if xs != "" and ys != "":
x = float(xs)
y = float(ys)
self.initial.append((x,y))
a = self.lb.GetSelections()
self.tabl = self.ipkey[a[0]]

except ValueError :
d = wx.MessageDialog(self, " All numbers should have float
values","ValueError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()

else:
self.EndModal(wx.ID_OK)

def EvtToggle(self, evt):
self.enable_text = not self.enable_text
self.text.Enable(self.enable_text)
self.grid.Enable(not self.enable_text)
self.lb.Enable(not self.enable_text)

class CalcVarDialog(wx.Dialog):

def __init__(self, parent, ID, title, var, pos=wx.DefaultPosition,
size=wx.DefaultSize, style=wx.DEFAULT_DIALOG_STYLE):
wx.Dialog.__init__(self, parent, ID, title, pos, size, style)
initial_string = ""
self.var = var
sizer = wx.FlexGridSizer(6,1)
label = wx.StaticText(self, -1, "X value")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text_x = wx.TextCtrl(self, -1, initial_string,
size=(120,23))
sizer.Add(self.text_x, 1, wx.ALIGN_CENTRE|wx.SHAPED, 5)
label = wx.StaticText(self, -1, "Y value")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text_y = wx.TextCtrl(self, -1, initial_string,
size=(120,23))
sizer.Add(self.text_y, 1, wx.ALIGN_CENTRE|wx.SHAPED, 5)
btnsizer = wx.StdDialogButtonSizer()
btn = wx.Button(self, wx.ID_OK)
btn.SetDefault()
btnsizer.AddButton(btn)
btn = wx.Button(self, wx.ID_CANCEL)
btnsizer.AddButton(btn)
btnsizer.Realize()
sizer.Add(btnsizer, 0, wx.ALIGN_CENTER_VERTICAL|wx.ALL, 5)
self.SetSizer(sizer)
sizer.Fit(self)
wx.EVT_BUTTON(self, wx.ID_OK, self.EvtOK)

def EvtOK(self, evt):

if self.text_x.IsModified():
```

```
try:
self.x = float(self.text_x.GetValue())

except ValueError :
d = wx.MessageDialog(self, " X should be a
float","ValueError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()

else:
self.text_x.SetModified(False)
itp_method = self.var.tabl

if type(self.var.initial) is list:
self.y = self.var.tipf[itp_method][0](self.x,
self.var.initial)
self.text_y.SetValue(str(self.y))

else:
self.text_y.SetValue("N/A")

elif self.text_y.IsModified():

try:
self.y = float(self.text_y.GetValue())

except ValueError :
d = wx.MessageDialog(self, " Y should be a
float","ValueError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()

else:
self.text_y.SetModified(False)
itp_method = self.var.tabl

if itp_method == LINEAL_REGRESSION:
self.x = (self.y - self.var.params[3])/self.var.params[2]
self.text_x.SetValue(str(self.x))

elif itp_method == EXPONENTIAL_REGRESSION:
self.x = (log(self.y) -
self.var.params[3])/self.var.params[2]
self.text_x.SetValue(str(self.x))

else:
self.text_x.SetValue("N/A")

class EditPlotOptions(wx.Dialog):

def __init__(self, parent, ID, title, pos=wx.DefaultPosition,
size=wx.DefaultSize, style=wx.DEFAULT_DIALOG_STYLE):
wx.Dialog.__init__(self, parent, ID, title, pos, size, style)
sizer = wx.BoxSizer(wx.VERTICAL)
label = wx.StaticText(self, -1, "Configure plot options")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.cb1 =wx.CheckBox(self, -1, "Show table points")
self.cb1.SetValue(frame.show_table_points)
sizer.Add(self.cb1, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.cb2 =wx.CheckBox(self, -1, "Show regression formula")
self.cb2.SetValue(frame.show_reg_formula)
sizer.Add(self.cb2, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.cb3 =wx.CheckBox(self, -1, "Show all scenarios")
self.cb3.SetValue(frame.show_all_scn)
sizer.Add(self.cb3, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
```

```
self.cb4 =wx.CheckBox(self, -1, "Do not show model name")
self.cb4.SetValue(frame.hide_model_name)
sizer.Add(self.cb4, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.cb5 =wx.CheckBox(self, -1, "Use python variable name")
self.cb5.SetValue(frame.use_python_name)
sizer.Add(self.cb5, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.cb6 =wx.CheckBox(self, -1, "Specify variable type")
self.cb6.SetValue(frame.specify_var_type)
sizer.Add(self.cb6, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
label = wx.StaticText(self, -1, "First year:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text_xmin = wx.TextCtrl(self, -1, str(frame.xmin),
size=(80,-1))
sizer.Add(self.text_xmin, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
label = wx.StaticText(self, -1, "Last year:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text_xmax = wx.TextCtrl(self, -1, str(frame.xmax),
size=(80,-1))
sizer.Add(self.text_xmax, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
label = wx.StaticText(self, -1, "Force plot title:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text_title = wx.TextCtrl(self, -1, frame.title, size=(80,-
1))
sizer.Add(self.text_title, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
label = wx.StaticText(self, -1, "Force Y axis title:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text_ytitle = wx.TextCtrl(self, -1, frame.ytitle,
size=(80,-1))
sizer.Add(self.text_ytitle, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
label = wx.StaticText(self, -1, "Force Y axis format:")
sizer.Add(label, 0, wx.ALIGN_CENTRE|wx.ALL, 5)
self.text_yformat = wx.TextCtrl(self, -1, frame.yformat,
size=(80,-1))
sizer.Add(self.text_yformat, 1, wx.ALIGN_CENTRE|wx.ALL, 5)
btnsizer = wx.StdDialogButtonSizer()
btn = wx.Button(self, wx.ID_OK)
btn.SetDefault()
btnsizer.AddButton(btn)
btn = wx.Button(self, wx.ID_CANCEL)
btnsizer.AddButton(btn)
btnsizer.Realize()
sizer.Add(btnsizer, 0, wx.ALIGN_CENTER_VERTICAL|wx.ALL, 5)
self.SetSizer(sizer)
sizer.Fit(self)
wx.EVT_BUTTON(self, wx.ID_OK, self.EvtOK)

def EvtOK(self, evt):
frame.show_table_points = self.cb1.GetValue()
frame.show_reg_formula = self.cb2.GetValue()
frame.show_all_scn = self.cb3.GetValue()
frame.hide_model_name = self.cb4.GetValue()
frame.use_python_name = self.cb5.GetValue()
frame.specify_var_type = self.cb6.GetValue()

try:
frame.xmin = int(self.text_xmin.GetValue())

except ValueError :
frame.xmin = None

try:
frame.xmax = int(self.text_xmax.GetValue())

except ValueError :
frame.xmax = None
frame.title = self.text_title.GetValue()
frame.ytitle = self.text_ytitle.GetValue()
```

```
frame.yformat = self.text_yformat.GetValue()
self.EndModal(wx.ID_OK)

class MainWindow(wx.Frame):

def __init__(self, parent, id, title, modelName, scnName=None):
wx.Frame.__init__(self, parent, id, title, size=(640,480))
self.initModel(modelName, scnName)

def initModel(self, modelName, scnName=None, reload=False):

try:
fp, pathname, description = imp.find_module(modelName)
self.model = imp.load_module(modelName, fp, pathname,
description)

except ImportError:
d = wx.MessageDialog(self, " Model "+modelName+" is not in the
current path", "ImportError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()

else:
self.modelName = modelName
self.simArray = []
self.sampleList1 = []
self.currentScn = 0
self.currentVar = 0
self.simulate()
self.get_variables()

if type(scnName) is str:
self.load_scn_file(scnName+".scn",
self.simArray[self.currentScn])

if reload:
self.destroyComponents()

else:
self.initOptions()
self.makeMenuBar()
self.makeComponents()

if reload:
self.addComponents()
self.sizer.Layout()

else:
self.makeSizer()

def makeGui(self):
self.makeMenuBar()
self.makeComponents()
self.makeSizer()

def initOptions(self):
self.show_table_points = True
self.show_reg_formula = False
self.show_all_scn = False
self.hide_model_name = False
self.use_python_name = False
self.specify_var_type = False
self.xmin = None
self.xmax = None
self.title = ""
self.ytitle = ""
self.yformat = ""
```

```
self.time_units = "time (y)"

def makeMenuBar(self):
self.CreateStatusBar()
filemenu = wx.Menu()
filemenu.Append(ID_OPEN, "&Choose model"," Choose model")
filemenu.AppendSeparator()
filemenu.Append(ID_IMPORT, "&Import all"," Import all")
filemenu.Append(ID_EXPORT, "&Export all"," Export all")
filemenu.AppendSeparator()
filemenu.Append(ID_PRINT, "&Print to file"," Print to file")
filemenu.AppendSeparator()
filemenu.Append(ID_EXIT,"&Quit"," Terminate the program")
scenariomenu = wx.Menu()
scenariomenu.Append(ID_DEFAULT, "&New scenario"," Create new
default scenario")
scenariomenu.AppendSeparator()
scenariomenu.Append(ID_LOAD, "&Load scenario"," Load scenario")
scenariomenu.Append(ID_SAVE, "Sa&ve scenario"," Save scenario")
scenariomenu.AppendSeparator()
scenariomenu.Append(ID_DEL, "Re&move scenario"," Remove current
scenario")
scenariomenu.Append(ID_DUPE, "&Duplicate scenario"," Duplicate
current scenario")
scenariomenu.Append(ID_EDIT_SCN, "Edit scenari&o"," Edit current
scenario")
variablemenu = wx.Menu()
variablemenu.Append(ID_EDIT_VAR, "Edi&t variable"," Edit current
variable")
variablemenu.Append(ID_CALC_VAR, "Ca&lculate"," Calculate")
configuremenu = wx.Menu()
configuremenu.Append(ID_OPT_PLOT, "Config&ure plot"," Configure
plot")
helpmenu = wx.Menu()
helpmenu.Append(ID_ABOUT, "&About"," Information about this
program")
menuBar = wx.MenuBar()
menuBar.Append(filemenu,"&File")
menuBar.Append(scenariomenu,"&Scenario")
menuBar.Append(variablemenu,"Va&riable")
menuBar.Append(configuremenu,"Confi&gure")
menuBar.Append(helpmenu,"&Help")
self.SetMenuBar(menuBar)
wx.EVT_MENU(self, ID_OPEN, self.EvtOpen)
wx.EVT_MENU(self, ID_IMPORT, self.EvtNotImplemented)
wx.EVT_MENU(self, ID_EXPORT, self.EvtNotImplemented)
wx.EVT_MENU(self, ID_PRINT, self.EvtPrint)
wx.EVT_MENU(self, ID_EXIT, self.EvtExit)
wx.EVT_MENU(self, ID_DEFAULT, self.EvtDefault)
wx.EVT_MENU(self, ID_LOAD, self.EvtLoad)
wx.EVT_MENU(self, ID_SAVE, self.EvtSave)
wx.EVT_MENU(self, ID_DEL, self.EvtDel)
wx.EVT_MENU(self, ID_DUPE, self.EvtDupe)
wx.EVT_MENU(self, ID_EDIT_SCN, self.EvtEditScn)
wx.EVT_MENU(self, ID_EDIT_VAR, self.EvtEditVar)
wx.EVT_MENU(self, ID_CALC_VAR, self.EvtCalcVar)
wx.EVT_MENU(self, ID_OPT_PLOT, self.EvtPlotOptions)
wx.EVT_MENU(self, ID_ABOUT, self.EvtAbout)

def makeComponents(self):
self.text1 = wx.StaticText(self, wx.ID_ANY, "Scenario:", (20, 20))
self.text2 = wx.StaticText(self, wx.ID_ANY, "Variable:", (20, 20))
self.lb1 = wx.ListBox(self, ID_LB1, (20, 40), (150, 120),
self.sampleList1, wx.LB_SINGLE)
self.lb2 = wx.ListBox(self, ID_LB2, (20, 40), (150, 120),
self.sampleList2, wx.LB_MULTIPLE)
self.lb1.Bind(wx.EVT_LISTBOX, self.EvtScnBox)
```

```
self.lb1.Bind(wx.EVT_LISTBOX_DCLICK, self.EvtScnBox2)
self.lb2.Bind(wx.EVT_LISTBOX, self.EvtVarBox)
self.lb2.Bind(wx.EVT_LISTBOX_DCLICK, self.EvtVarBox2)
self.lb1.SetSelection(self.currentScn)
self.lb2.SetSelection(self.currentVar)
self.fig = Figure((8.8,6.6), 75)
self.canvas = FigureCanvasWxAgg(self, wx.ID_ANY, self.fig)
self.plot()

def destroyComponents(self):
self.text1.Destroy()
self.text2.Destroy()
self.lb1.Destroy()
self.lb2.Destroy()
self.canvas.Destroy()

def addComponents(self):
self.sizer.Add(self.text1, (1,1), (1,1), wx.TOP | wx.LEFT |
wx.EXPAND)
self.sizer.Add(self.lb1, (2,1), (1,1), wx.TOP | wx.LEFT |
wx.EXPAND)
self.sizer.Add(self.text2, (1,2), (1,1), wx.TOP | wx.LEFT |
wx.EXPAND)
self.sizer.Add(self.lb2, (2,2), (1,1), wx.TOP | wx.LEFT |
wx.EXPAND)
self.sizer.Add(self.canvas, (2,3), (1,1), wx.TOP | wx.LEFT |
wx.EXPAND)

def makeSizer(self):
self.sizer = wx.GridBagSizer(2,3)
self.addComponents()
self.SetSizer(self.sizer)
self.Show(True)
self.Fit()

def plot(self, z = None):
decimals = 5
lletres = 35
fs = 14

if z == None:
n = 1

else:
n = len(z)
fs = fs-2*n

if fs < 6:
fs = 6

for i in xrange(n):

if n>3:
rcParams['xtick.major.pad'] = 3
rcParams['xtick.minor.pad'] = 3

elif n == 3:
rcParams['xtick.major.pad'] = 5
rcParams['xtick.minor.pad'] = 5

else:
rcParams['xtick.major.pad'] = 10
rcParams['xtick.minor.pad'] = 10
sp = n*100+i+11
simCount = -1
lineNumber = len(self.simArray)
```

```
for sim in self.simArray:
simCount = simCount + 1

if sim == self.simArray[self.currentScn] or self.show_all_scn
== True:
time = sim.time

if n == 1 or z == None:
varname = self.sampleList2[self.currentVar]

else:
varname = self.sampleList2[z[i]]
value = getattr(sim, varname).value
initial = getattr(sim, varname).initial
params = getattr(sim, varname).params
svarname = getattr(sim, varname).label

if svarname == "" or self.use_python_name:
svarname = varname

if len(svarname)>lletres:
svarname = svarname[:lletres]

else:
svarname = svarname
a = self.fig.add_subplot(sp)
a.plot(time,value,lw=2,c=colorList[simCount%lineNumber])
xmin, xmax, ymin, ymax = a.axis()

if xmin < self.xmin and self.xmin is not None:
xmin = self.xmin

if xmax > self.xmax and self.xmax is not None:
xmax = self.xmax
a.axis((xmin,xmax,ymin,ymax))

if getattr(sim, varname).__class__ is self.InVar:
s = " (In)"

elif getattr(sim, varname).__class__ is self.OutVar:
s = " (Out)"

elif getattr(sim, varname).__class__ is self.AuxVar:
s = " (Aux)"
else:
s = ""

if self.specify_var_type:
svarname = svarname+s

if type(initial) is list and self.show_table_points:
x = []
y = []

for j in initial:
x.append(j[0])
y.append(j[1])
a.plot(x,y,"go")

if sim == self.simArray[self.currentScn]:

if initial == params[0] and LINEAL_REGRESSION == params[1]

and self.show_reg_formula:
formula = svarname+" = "+str(round(params[2], decimals))+"
time + "+str(round(params[3], decimals))
```

```
a.legend((formula,), 'upper center', shadow=True)
```

```
if initial == params[0] and EXPONENTIAL_REGRESSION ==
params[1] and self.show_reg_formula:
formula = "ln("+svarname+") =
"+str(round(params[2],decimals))+" time +
"+str(round(params[3],decimals))
a.legend((formula,), 'upper center', shadow=True)

if i == 0:

if self.title=="":

if self.hide_model_name:
a.set_title(sim.name, fontsize = fs)

else:
a.set_title(self.modelName+" ("+sim.name+")", fontsize
= fs)

else:
a.set_title(self.title, fontsize = fs)

if i == n-1:
a.set_xlabel(self.time_units, fontsize = fs)

if self.ytitle == "":
a.set_ylabel(svarname+"\n", fontsize = fs)

else:
a.set_ylabel(self.ytitle+"\n", fontsize = fs)

if self.yformat != "":
a.yaxis.set_major_formatter(FormatStrFormatter(self.yformat))
a.xaxis.set_major_formatter(FormatStrFormatter('%d'))

for tick in a.xaxis.get_major_ticks():
tick.label1.set_fontsize(fs)

for tick in a.yaxis.get_major_ticks():
tick.label1.set_fontsize(fs)
multi = a.yaxis.get_offset_text()
multi.set_fontsize(fs)
self.canvas.draw()

def oldplot(self):
decimals = 5
lletres = 4

for sim in self.simArray:

if sim == self.simArray[self.currentScn] or self.show_all_scn ==
True:
time = sim.time
varname = self.sampleList2[self.currentVar]
value = getattr(sim, varname).value
initial = getattr(sim, varname).initial
params = getattr(sim, varname).params

if len(varname)>lletres:
svarname = varname[:lletres]
```

```
else:
svarname = varname
a = self.fig.add_subplot(111)
a.plot(time,value)

if getattr(sim, varname).__class__ is self.InVar:
s = " (In)"

elif getattr(sim, varname).__class__ is self.OutVar:
s = " (Out)"

elif getattr(sim, varname).__class__ is self.AuxVar:
s = " (Aux)"

else:
s = ""

if type(initial) is list and self.show_table_points:
x = []
y = []

for i in initial:
x.append(i[0])
y.append(i[1])
a.plot(x,y,"go")

if sim == self.simArray[self.currentScn]:

if initial == params[0] and LINEAL_REGRESSION == params[1] and
self.show_reg_formula:
formula = svarname+" = "+str(round(params[2], decimals))+"
time + "+str(round(params[3], decimals))
a.legend((formula,), 'upper center', shadow=True)

if initial == params[0] and EXPONENTIAL_REGRESSION ==
params[1] and self.show_reg_formula:
formula = "ln("+svarname+") =
"+str(round(params[2],decimals))+" time +
"+str(round(params[3],decimals))
a.legend((formula,), 'upper center', shadow=True)
a.set_title(self.modelName+" ("+sim.name+")")
a.set_xlabel("time (y)")
a.set_ylabel(varname+s+"\n")
a.xaxis.set_major_formatter(FormatStrFormatter('%d'))
self.canvas.draw()

def simulate(self):
self.sim = self.model.Model(self)
self.sim.calcul()
self.simArray.append(self.sim)
self.sampleList1.append(self.sim.name)
self.currentScn = len(self.simArray) - 1

def get_variables(self):
self.sampleList2 = []
self.in_vars = []
self.out_vars = []
self.aux_vars = []

for i in dir(self.simArray[self.currentScn]):

try:
b = getattr(getattr(self.simArray[self.currentScn], i),
"__class__")

except AttributeError:
```

```
pass

else:

if b is self.InVar:
self.in_vars.append(i)
self.sampleList2.append(i)

if b is self.OutVar:
self.out_vars.append(i)
self.sampleList2.append(i)

if b is self.AuxVar:
self.aux_vars.append(i)
self.sampleList2.append(i)

def load_scn_file(self, fname):

try:
f = open(fname, "r")

except IOError:
d = wx.MessageDialog(self, " Scenario file "+fname+" is not in
the current path.", "IOError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()

else:
self.simulate()
sim = self.simArray[self.currentScn]

for line in f.readlines():
l=line[:-1]
p=l.split(";")

if p[0] == "Model:" and p[1]!=self.modelName:
d = wx.MessageDialog(self, " This scenario was created with
model "+p[1]+"\n but currently you are using model
"+self.modelName+".\n Check that both models define the same
variables.","Warning", wx.OK|wx.ICON_EXCLAMATION)
d.ShowModal()
d.Destroy()

elif p[0] == "Name:":
sim.name = p[1]

elif p[0] == "Time:":
sim.time, sim.n = self.timeInit(int(p[1]), int(p[2]))

elif p[0] == "In:" or p[0] == "Out:" or p[0] == "Aux:":
getattr(sim, p[1]).load_scn(p[2:])
a = getattr(sim, p[1]).initial
b = getattr(sim, p[1]).tabl
s = getattr(sim, p[1]).label
getattr(sim, p[1]).__init__(sim.time, a, b, s)
f.close()
sim.calcul()
self.sampleList1[self.currentScn] = sim.name

def EvtNotImplemented(self, event):
d = wx.MessageDialog(self, " This option is not \n implemented
yet.","Bad luck", wx.OK|wx.ICON_EXCLAMATION)
d.ShowModal()
d.Destroy()
```

```
def EvtOpen(self,e):
d = wx.FileDialog(self, "Choose model", wildcard = "*.py")
d.ShowModal()
fname = d.GetFilename()
d.Destroy()
modelName = fname[:-3]
self.initModel(modelName, reload=True)

def EvtPrint(self,e):
sim = self.simArray[self.currentScn]
if sys.platform == "linux2":
d = wx.FileDialog(self, "Print to file", wildcard = "*.png",
style = wx.FD_SAVE)

else:
d = wx.FileDialog(self, "Print to file", wildcard = "*.png")
d.ShowModal()
fname = d.GetPath() # Tractament multiplataforma extensions
d.Destroy()
self.fig.savefig(fname)

def EvtExit(self,e):
self.Close(True)

def EvtDefault(self, event):
self.simulate()
self.destroyComponents()
self.makeComponents()
self.addComponents()
self.sizer.Layout()

def EvtLoad(self, event):
d = wx.FileDialog(self, "Load scenario", wildcard = "*.scn")
d.ShowModal()
fname = d.GetPath()
d.Destroy()
self.load_scn_file(fname)
self.destroyComponents()
self.makeComponents()
self.addComponents()
self.sizer.Layout()

def EvtSave(self, event):
sim = self.simArray[self.currentScn]

if sys.platform == "linux2":
d = wx.FileDialog(self, "Save scenario", defaultFile = sim.name,
wildcard = "*.scn", style = wx.FD_SAVE)

else:
d = wx.FileDialog(self, "Save scenario", defaultFile = sim.name,
wildcard = "*.scn")
d.ShowModal()
fname = d.GetPath()
d.Destroy()

try:
f = open(fname, "w")

except IOError:
d = wx.MessageDialog(self, " Could not open "+fname+" in write
mode", "IOError", wx.OK|wx.ICON_ERROR)
d.ShowModal()
d.Destroy()

else:
f.write("Model:;"+self.modelName+"\n")
```

```
f.write("Name:;"+sim.name+"\n")
f.write("Time:;"+str(sim.time[0])+";"+str(sim.time[-1])+"\n")

for i in dir(sim):

try:
b = getattr(getattr(sim, i), "__class__")

except AttributeError:

pass

else:

if b is self.InVar:
getattr(sim, i).save_scn(f, "In:;"+i+";")

if b is self.OutVar:
getattr(sim, i).save_scn(f, "Out:;"+i+";")

if b is self.AuxVar:
getattr(sim, i).save_scn(f, "Aux:;"+i+";")
f.close()

def EvtDel(self, event):

if len(self.simArray)>1:
self.simArray =
self.simArray[:self.currentScn]+self.simArray[self.currentScn+1:]
self.sampleList1 =
self.sampleList1[:self.currentScn]+self.sampleList1[self.currentSc
n+1:
]
self.currentScn = self.currentScn-1
self.destroyComponents()
self.makeComponents()
self.addComponents()
self.sizer.Layout()
```

```
def EvtDupe(self, event):
original = self.simArray[self.currentScn]
sample = self.sampleList1[self.currentScn]
self.simulate()
self.sampleList1[self.currentScn] = sample
copia = self.simArray[self.currentScn]
copia.name = original.name
copia.time = original.time
copia.n = original.n

for i in self.sampleList2:
initial = getattr(original, i).initial
tabl = getattr(original, i).tabl
label = getattr(original, i).label
getattr(copia, i).__init__(copia.time, initial, tabl, label)
copia.calcul()
self.destroyComponents()
self.makeComponents()
self.addComponents()
self.sizer.Layout()

def EvtEditScn(self, event):
sim = self.simArray[self.currentScn]
name = sim.name
initial_year = sim.time[0]
final_year = sim.time[-1]
d = EditScnDialog(self, -1, "Edit scenario", name, initial_year,
final_year, size=(350, 200), style = wx.DEFAULT_DIALOG_STYLE)
d.CenterOnScreen()
val = d.ShowModal()

if val == wx.ID_OK:
name = d.name
initial_year = d.initial_year
final_year = d.final_year
sim.name = name
self.sampleList1[self.currentScn] = name
sim.time, sim.n = self.timeInit(initial_year, final_year)

for i in self.sampleList2:
a = getattr(getattr(sim, i), "initial")
b = getattr(getattr(sim, i), "tabl")
s = getattr(getattr(sim, i), "label")
getattr(getattr(sim, i), "__init__")(sim.time, a, b, s)
sim.calcul()
d.Destroy()
self.destroyComponents()
self.makeComponents()
self.addComponents()
self.sizer.Layout()

def EvtEditVar(self, event):
sim = self.simArray[self.currentScn]
var = getattr(sim, self.sampleList2[self.currentVar])
initial = var.initial
tipf = var.tipf
tabl = var.tabl
label = var.label
```

```
if isinstance(var, MainWindow.AuxVar):
d = wx.MessageDialog(self, " This is an auxiliary variable, \n
nothing to modify!","Edit variable", wx.OK|wx.ICON_INFORMATION)
d.ShowModal()
d.Destroy()

else:
d = EditVarDialog(self, -1, "Edit variable", initial, tipf,
tabl, size=(350, 200), style = wx.DEFAULT_DIALOG_STYLE)
d.CenterOnScreen()
val = d.ShowModal()

if val == wx.ID_OK:
var.__init__(sim.time, d.initial, d.tabl, var.label)
sim.calcul()
d.Destroy()
self.fig.clear()
self.plot()

def EvtCalcVar(self, event):
sim = self.simArray[self.currentScn]
var = getattr(sim, self.sampleList2[self.currentVar])

if isinstance(var, MainWindow.AuxVar) or isinstance(var,
MainWindow.OutVar):
d = wx.MessageDialog(self, " This is not an input variable, \n
nothing to calculate!","Calculate", wx.OK|wx.ICON_INFORMATION)
d.ShowModal()
d.Destroy()

else:
d = CalcVarDialog(self, -1, "Calculate", var, size=(350, 200),
style = wx.DEFAULT_DIALOG_STYLE)
d.CenterOnScreen()
val = d.ShowModal()
d.Destroy()

def EvtPlotOptions(self, event):
d = EditPlotOptions(self, -1, "Configure plot", size=(350, 200),
style = wx.DEFAULT_DIALOG_STYLE)
d.CenterOnScreen()
val = d.ShowModal()
d.Destroy()
self.destroyComponents()
self.makeComponents()
self.addComponents()
self.sizer.Layout()

def EvtAbout(self, event):
d = wx.MessageDialog(self, " A simple PySight scenario \n editor
written in wxPython \n by Jordi Sellares","About wxPySight",
wx.OK|wx.ICON_INFORMATION)
d.ShowModal()
d.Destroy()

def EvtScnBox(self, event):
a = self.lb1.GetSelections()

try:
self.currentScn = a[0]

except IndexError:

pass
self.fig.clear()
self.plot()
```

```
def EvtScnBox2(self, event):
a = self.lb1.GetSelections()
self.currentScn = a[0]

def EvtVarBox(self, event):
a = self.lb2.GetSelections()

try:
self.currentVar = a[0]

except IndexError:

pass
self.fig.clear()
self.plot()

def EvtVarBox2(self, event):
a = self.lb2.GetSelections()

try:
self.currentVar = a[0]

except IndexError:

pass
self.fig.clear()
self.plot(a)
############################################################

def timeInit(self, a, b):
c = arange(a, b+1)
n = len(c)-1

return c, n

class PySightVar:

def __init__(self, s = ""):
self.tipf = {LINEAL_INTERPOLATION : (self.intrpl, "Lineal
Interpolation"), LINEAL_REGRESSION : (self.regrsl, "Linear
Regression"), EXPONENTIAL_REGRESSION : (self.regrse, "Exponential
Regression")}
self.params = [None, None]
self.label = s

def intrpl(self, x, b):
self.params = [b, LINEAL_INTERPOLATION]
ftd = True
ftb = True
dpd = 0.0
dpb = 0.0

for i in b:
d = i[0]-x

if d >= 0.0 and (d < dpd or ftd):
xd = i[0]
yd = i[1]
dpd = d
ftd = False

if d <= 0.0 and (d > dpb or ftb):
xb = i[0]
yb = i[1]
dpb = d
ftb = False
```

```
try:
y = yb

if xb != xd:
y = y+(x-xb)*(yd-yb)/(xd-xb)

except UnboundLocalError:

print "WARNING! Nonsense interpolation!"
y = 0.0

return y

def regrsl(self, x, b):

if b <> self.params[0] or LINEAL_REGRESSION <> self.params[1]:
s = 0.0
sx = 0.0
sy = 0.0
sxx = 0.0
sxy = 0.0
syy = 0.0

for i in b:
s = s+1
sx = sx+i[0]
sy = sy+i[1]
sxx = sxx + i[0]*i[0]
sxy = sxy + i[0]*i[1]
syy = syy + i[1]*i[1]
dsc = s*sxx-sx*sx
odn = (sxx*sy-sx*sxy)/dsc
pdt = (s*sxy-sx*sy)/dsc
cor = (s*sxy-sx*sy)/sqrt((s*sxx-sx*sx)*(s*syy-sy*sy))
self.params = [b, LINEAL_REGRESSION, pdt, odn]

print "Pendent", pdt
print "Ordenada a l'origen", odn
print "Correlacio", cor
print

else:
pdt = self.params[2]
odn = self.params[3]
y = pdt*x + odn

return y

def regrse(self, x, b):

if b <> self.params[0] or EXPONENTIAL_REGRESSION <>
self.params[1]:
s = 0.0
sx = 0.0
sy = 0.0
sxx = 0.0
sxy = 0.0
syy = 0.0

for i in b:
j = log(i[1])
s = s+1
sx = sx+i[0]
sy = sy+j
sxx = sxx + i[0]*i[0]
sxy = sxy + i[0]*j
```

```
syy = syy + j*j
dsc = s*sxx-sx*sx
odn = (sxx*sy-sx*sxy)/dsc
pdt = (s*sxy-sx*sy)/dsc
cor = (s*sxy-sx*sy)/sqrt((s*sxx-sx*sx)*(s*syy-sy*sy))
self.params = [b, EXPONENTIAL_REGRESSION, pdt, odn]

print "Pendent", pdt
print "Ordenada a l'origen", odn
print "Correlacio", cor
print

else:
pdt = self.params[2]
odn = self.params[3]
y = odn + pdt*x

return e**y

def save_scn(self, f, prefix):

if type(self.initial) is float:
f.write(prefix)
f.write(str(self.initial))
f.write("\n")

elif type(self.initial) is list:
f.write(prefix)

for i in self.initial:
f.write(str(i[0])+";")
f.write(str(i[1])+";")
f.write(str(self.tabl))
f.write("\n")

else:
f.write(prefix+"None \n")

def load_scn(self, s):
if len(s) == 1:

if s[0] == "None ":
self.initial = None

else:
self.initial = float(s[0])

else:
self.initial = []

for i in xrange(0,len(s)-1,2):
self.initial.append((float(s[i]),float(s[i+1])))
self.tabl=int(s[-1])

class AuxVar(PySightVar):

def __init__(self, a, b = None, c = None, label = ""):
MainWindow.PySightVar.__init__(self, label)
self.tipf = None
b = len(a)
self.value = zeros(b)+0.0
self.initial = None
self.tabl = None

class InVar(PySightVar):
```

```
def __init__(self, a, b, c = LINEAL_INTERPOLATION, label = ""):
MainWindow.PySightVar.__init__(self, label)
ipf = self.tipf[c][0]
d = len(a)

if type(b) is float:
self.value = zeros(d)+b

if type(b) is list:
self.value = zeros(d)+0.0

for i in xrange(len(self.value)):
self.value[i] = ipf(a[i], b)
self.initial = b
self.tabl = c

class OutVar(PySightVar):

def __init__(self, a, b, c = LINEAL_INTERPOLATION, label = ""):
MainWindow.PySightVar.__init__(self, label)
ipf = self.tipf[c][0]
d = len(a)
e = zeros(d)+0.0

if type(b) is float:
e[0] = b

if type(b) is list:
e[0] = ipf(a[0], b)
self.value = zeros(d)+e
self.initial = b
self.tabl = c
############################################################

def get_names(fname):

try:
f = open(fname, "r")
l = f.readline()
l = l[:-1]
f.close()

except IOError:

print "ERROR! Binary or non-existent file!"
sys.exit()
p = l.split(";")

if p[0] == "Model:":
i = p[1]
j = fname[:-4]

else:

print "ERROR! Not a valid scenario file!"
sys.exit()

return i,j

try:
modelName = sys.argv[1]

except IndexError:
modelName = "simple"

try:
```

```
scnName = sys.argv[2]

except IndexError:
scnName = None

if modelName[-4:]==".scn":
modelName, scnName = get_names(modelName)
app = wx.PySimpleApp()
frame=MainWindow(None, wx.ID_ANY, 'wxPySight', modelName, scnName)
app.MainLoop()
```

Generated by GNU enscript 1.6.4.