WWU
MÜNSTER

Dynamic modelling with PCRaster Python

Judith A. Verstegen

living.knowledge

ifgi
Institut für Geoinformatik
Universität Münster

Schedule

- 14:00 – 14:30: Intro lecture + demo
- 14:30 – 15:30: Tutorial Part I
- 15:30 – 15:45: Discuss answers Part I
- (15:45 – 16:30: Part II)

Coffee break: go as you like

Outline intro lecture

1. PCRaster – concepts
2. Implementation
3. Example: Land use change
4. Tutorial: Fire spread modelling

PCRaster – concepts

Entities in PCRaster

Map

- main variable in a model, **typically every variable is a map**
- six data types exist
- PCRaster .map format (binary file)

Table

- used to assign new values as a function of input maps or store output statistics
- ascii data file

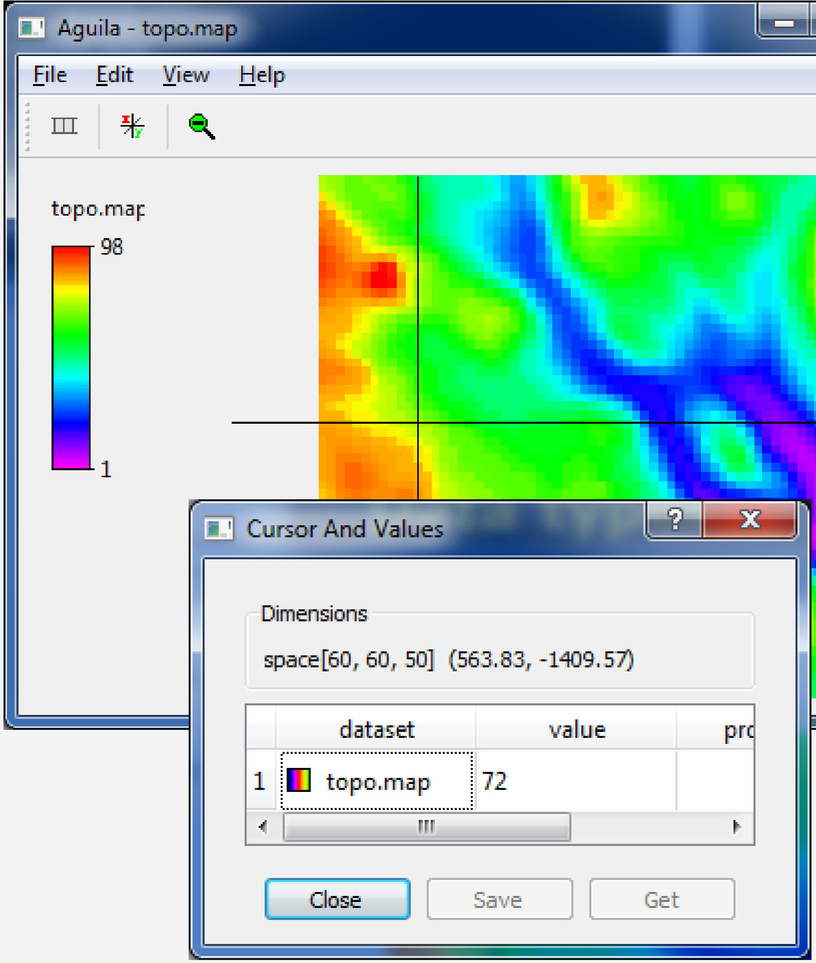
Time series

- used in dynamic models to have inputs per time step
- ascii data file

The 6 data types in PCRaster

data type	**description attributes**	**domain**	**example**
Boolean	boolean	0 (false), 1 (true)	suitable/unsuitable, visible/non visible
nominal	classified, no order	0...255, whole values	soil classes, administrative regions
ordinal	classified, order	0...255, whole values	succession stages, income groups
scalar	continuous, linear	- 10exp(37)...10exp(37), real values	elevation, temperature
directional	continuous, directional	0 to 2 pi (radians), or to 360 (degrees), and -1 (no direction), real values	aspect
ldd	local drain direction to neighbour cell	1...9 (codes of drain directions)	drainage networks, wind directions

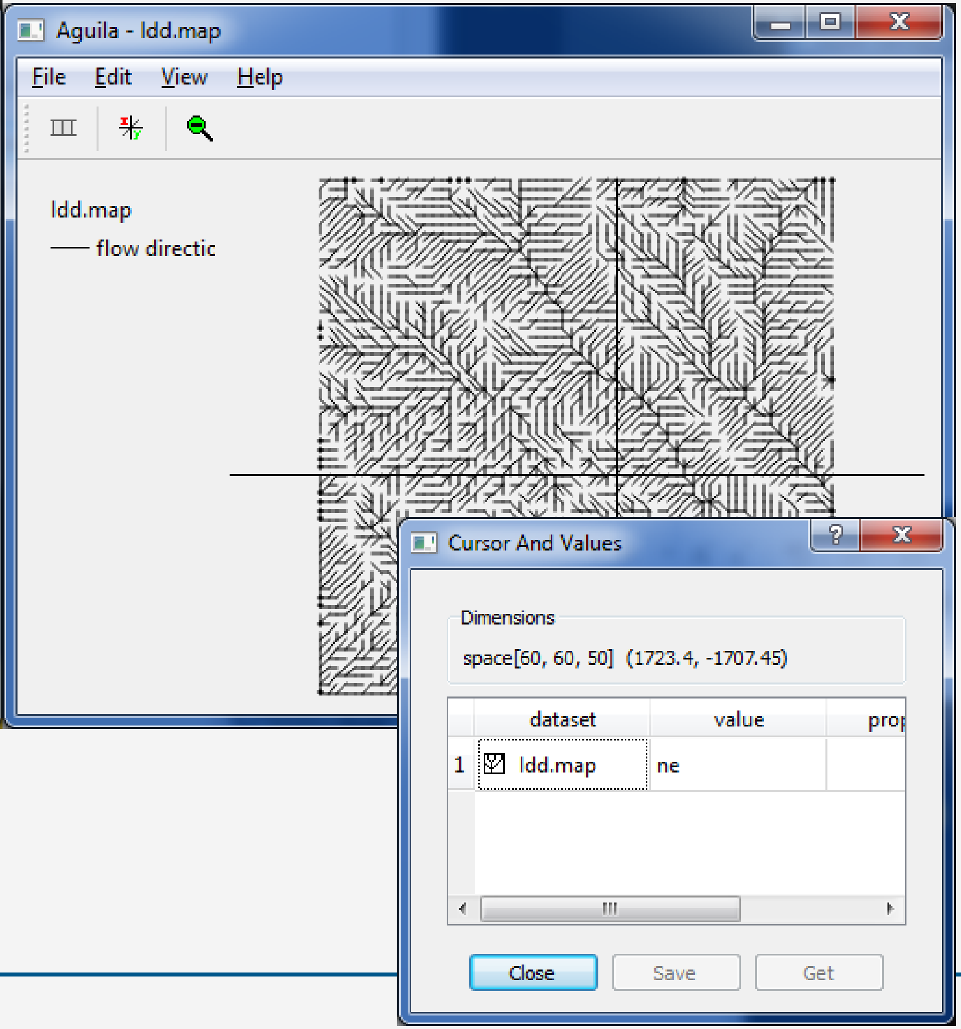
Data types: example



Dimensions

space[60, 60, 50] (563.83, -1409.57)

	dataset	value	pro
1	topo.map	72	



Dimensions

space[60, 60, 50] (1723.4, -1707.45)

	dataset	value	prop
1	☒ ldd.map	ne	

The PCRaster python framework

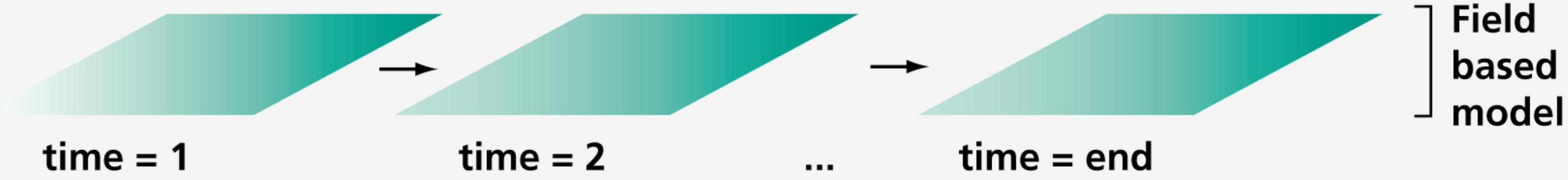
What is the PCRaster Python framework?

- A set of python classes that you can use for model construction
- Spatial operations can be used from the PCRaster library
- It takes care of the model initialization and the time steps for you
- It includes routines for Monte Carlo simulation and data assimilation

Field-based Modelling

Dynmaic model:

$z(1..m)(t+1) = \mathrm{f}(z(1..m)(\mathrm{t}), i(1..n)(\mathrm{t}))$



time = 1

time = 2

...

time = end

Field based model

With:

🟩 Field containing multiple attributes

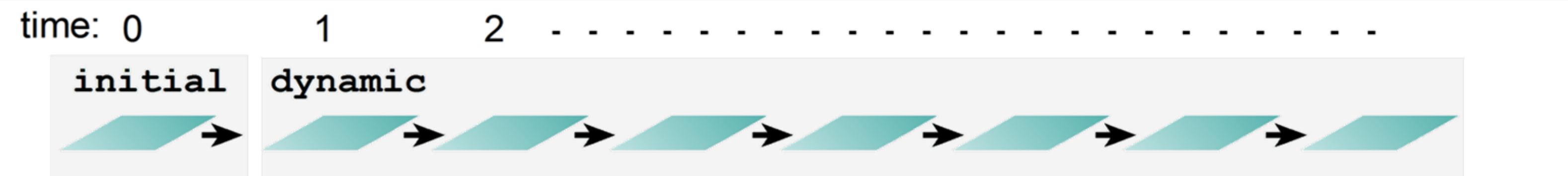
→ Functions representing change over a time step

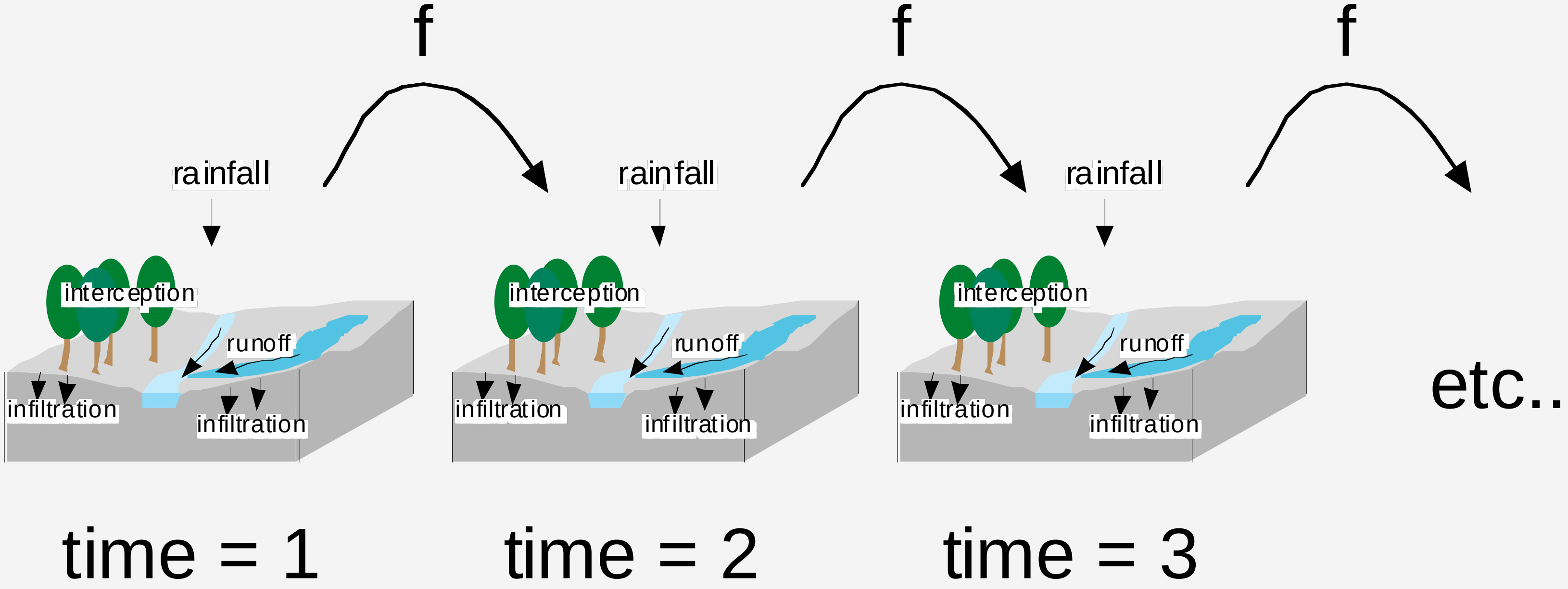
$i(1..n)$: inputs (maps/single values)

$z(1..m)$: outputs (model variables) changing over time (t)

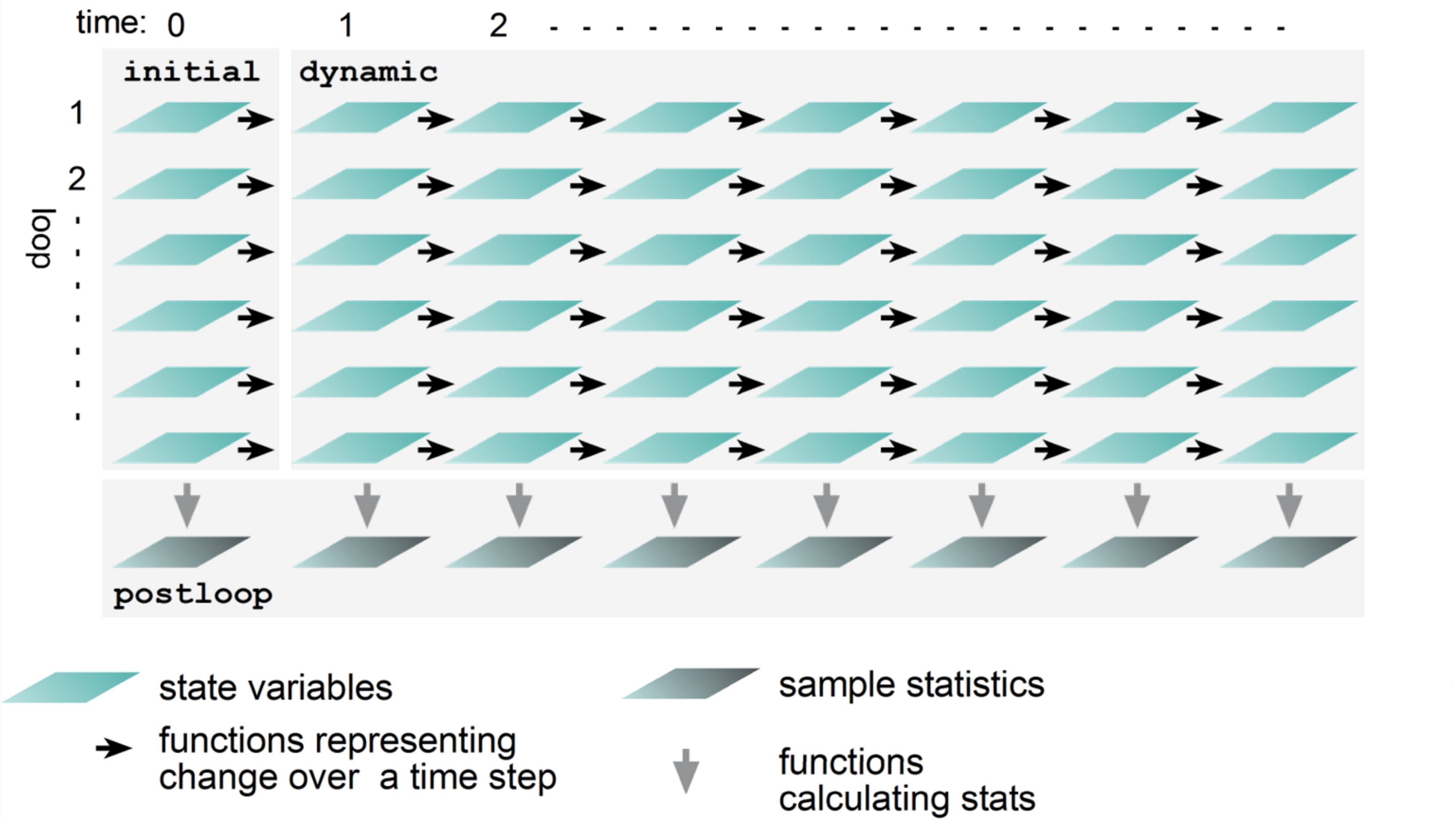
f: the dynamic model (transition function)

PCRaster python framework: dynamic

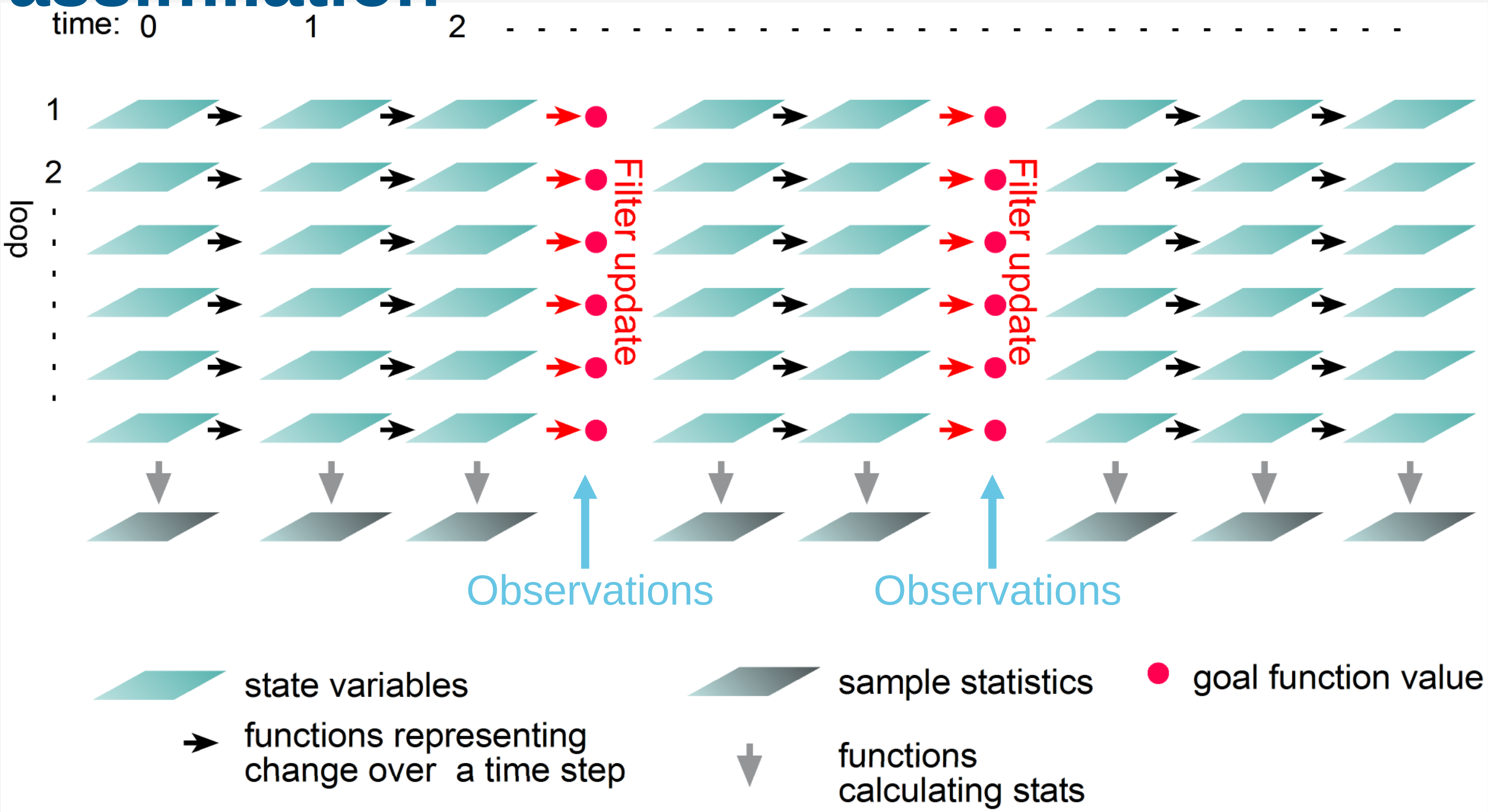


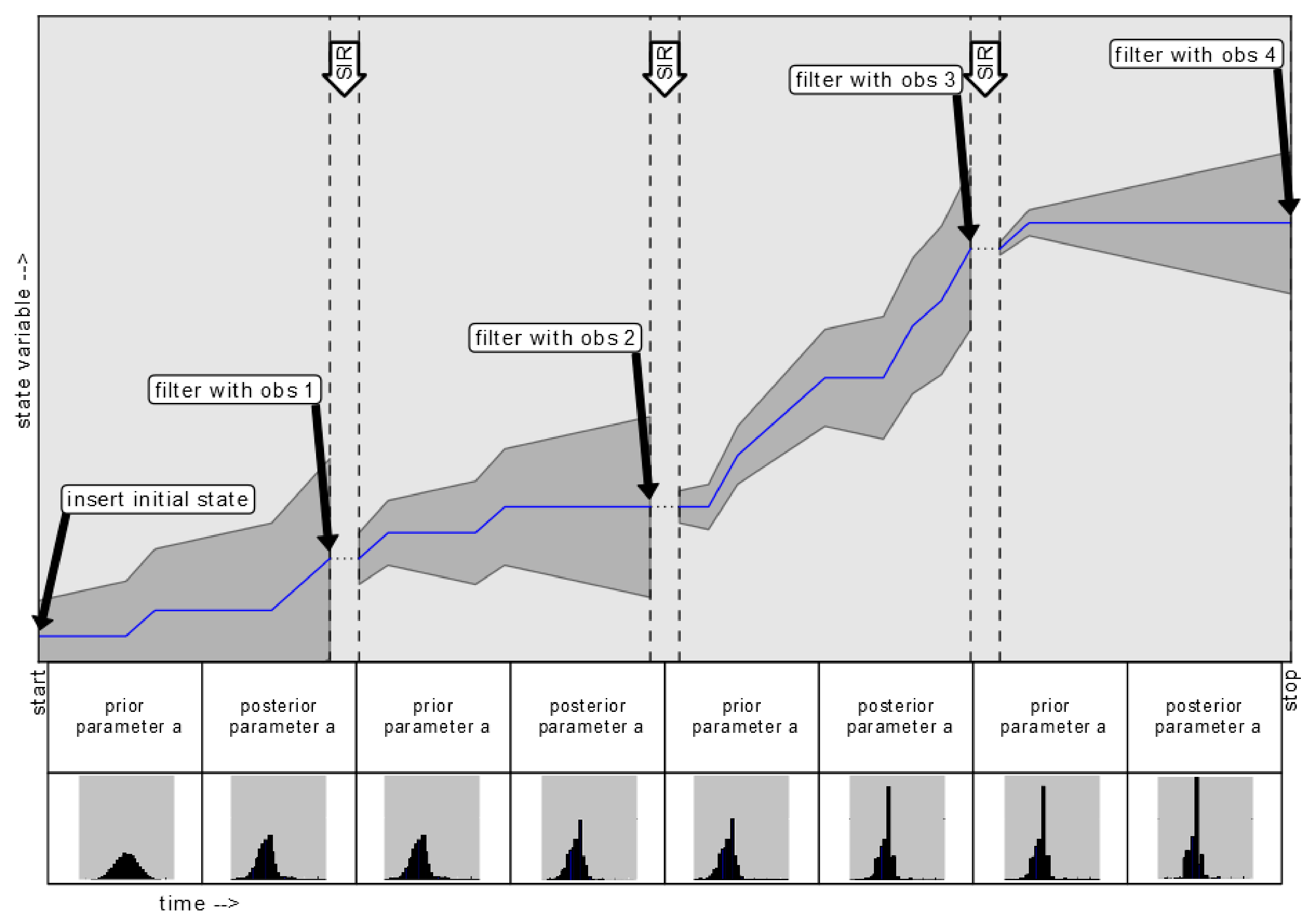


PCRaster python framework: stochastic



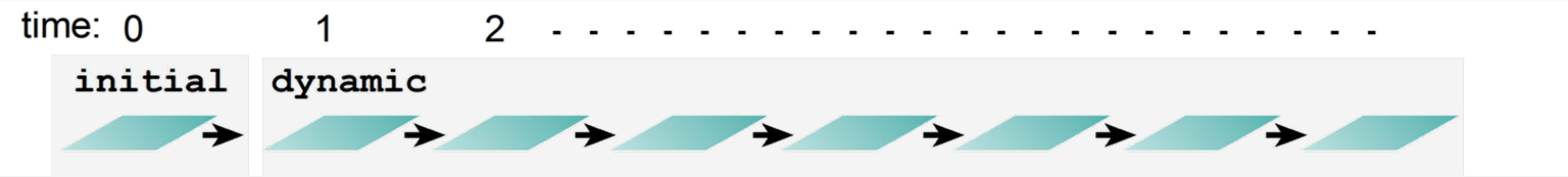
PCRaster python framework: data assimilation





[Verstegen et al., 2014, Environmental Modelling & Software]

PCRaster python framework: dynamic



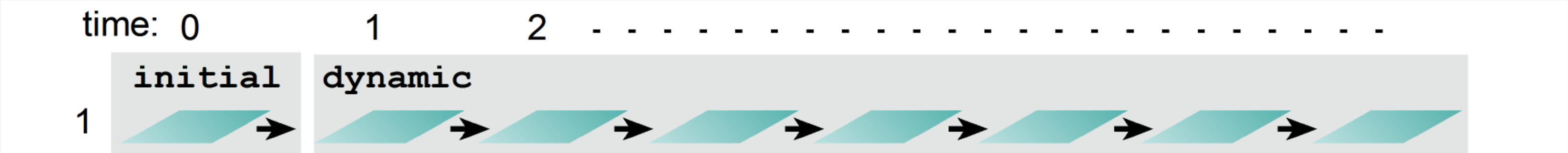
initial():

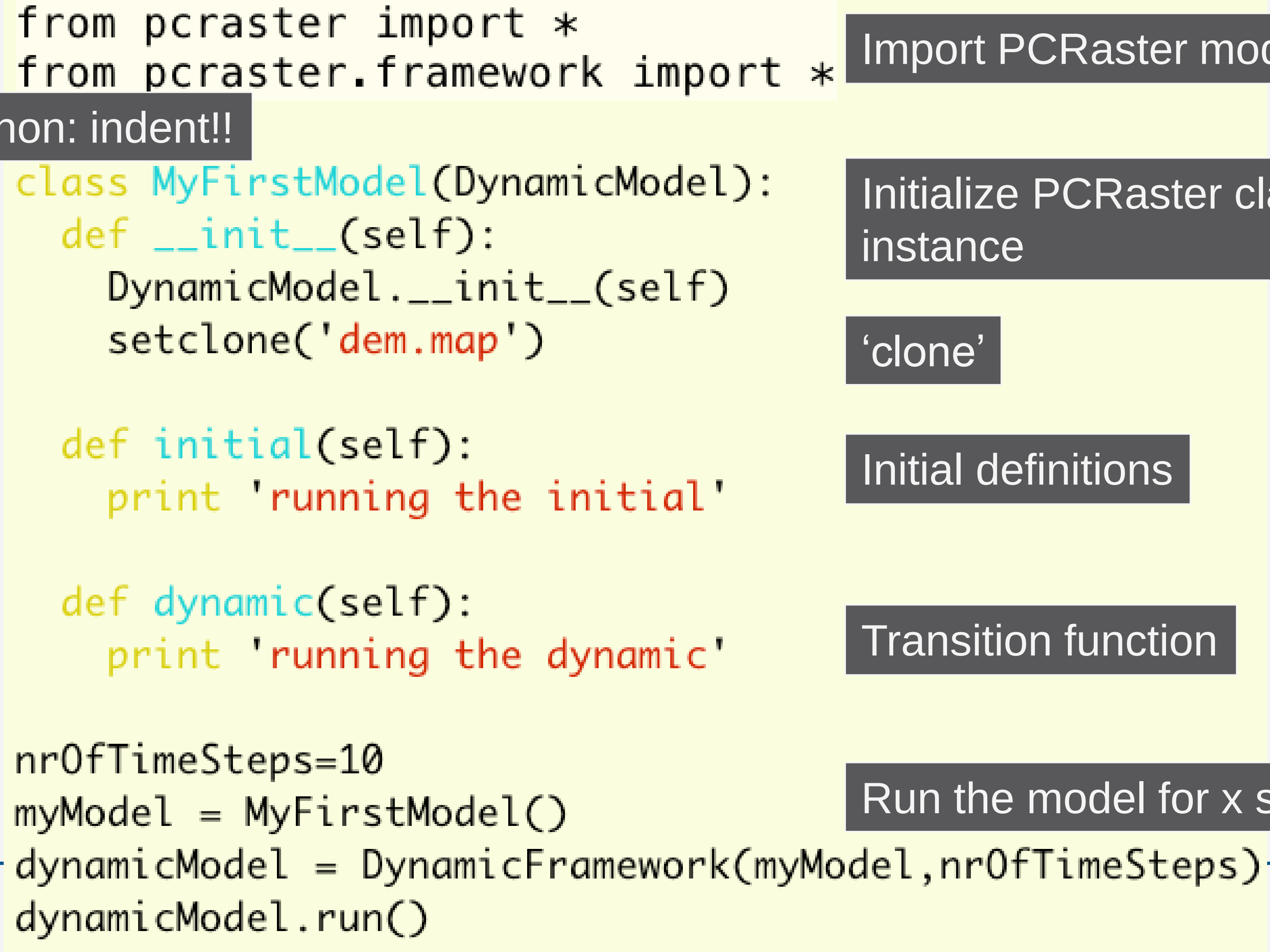
- The initial system state definition

dynamic():

- The transition function that is run in each time step

Implementation





```
from pcraster import *
from pcraster.framework import *

class MyFirstModel(DynamicModel):
  def __init__(self):
    DynamicModel.__init__(self)
    setclone('dem.map')

  def initial(self):
    print 'running the initial'

  def dynamic(self):
    print 'running the dynamic'

nrOfTimeSteps=10
myModel = MyFirstModel()
dynamicModel = DynamicFramework(myModel,nrOfTimeSteps)
dynamicModel.run()
```

Import PCRaster module

Python: indent!!

Initialize PCRaster class instance

‘clone’

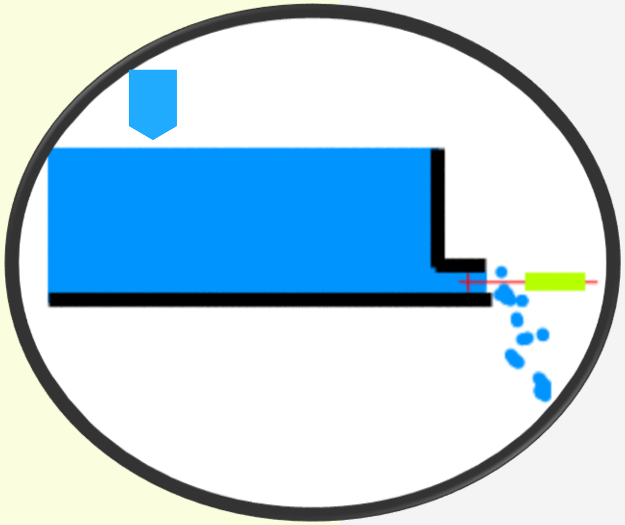
Initial definitions

Transition function

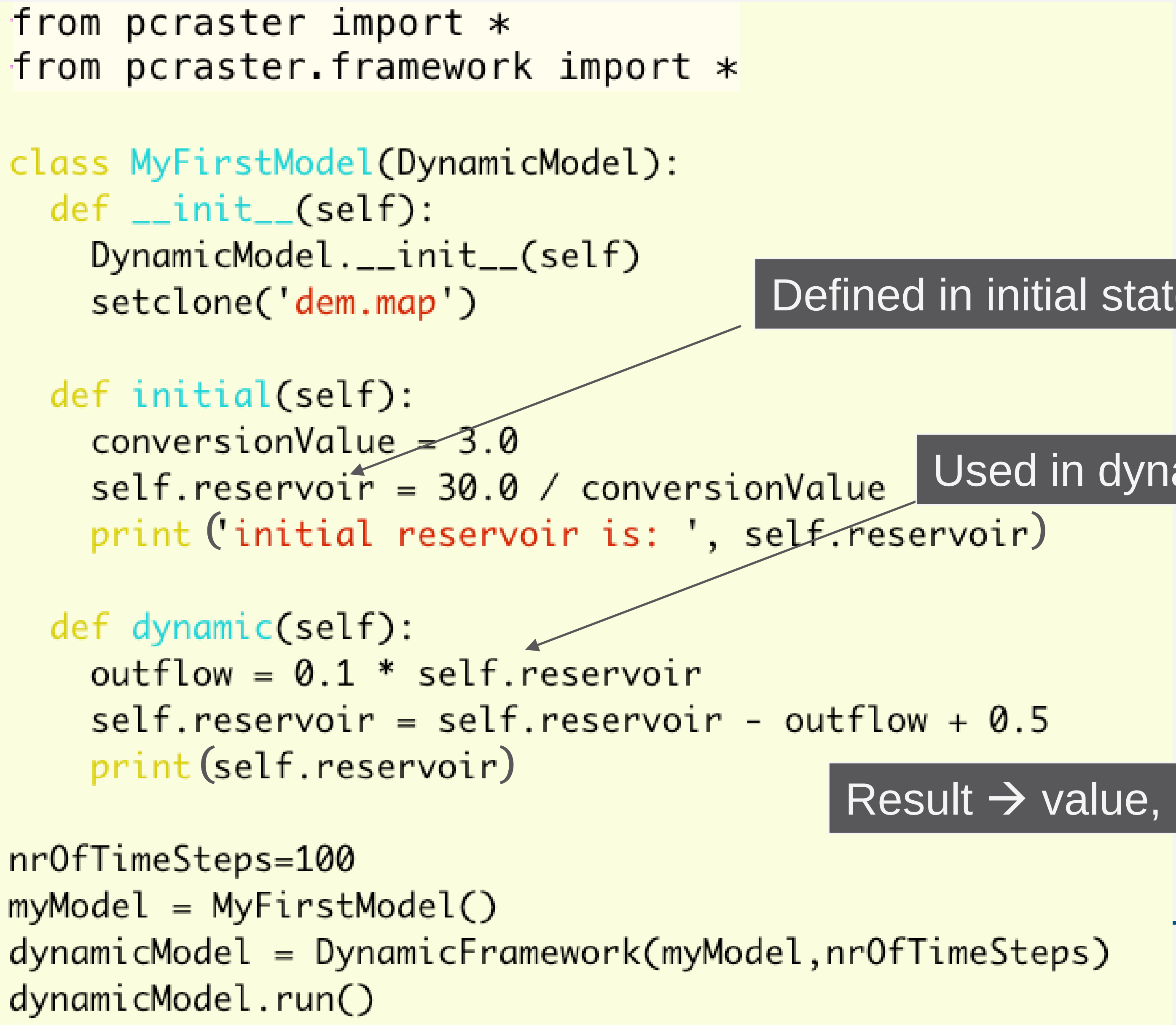
Run the model for x steps

Non-spatial model

```
from pcraster import *
from pcraster.framework import *

class MyFirstModel(DynamicModel):
  def __init__(self):
    DynamicModel.__init__(self)
    setclone('dem.map')

  def initial(self):
    conversionValue = 3.0
    self.reservoir = 30.0 / conversionValue
    print 'initial reservoir is: ', self.reservoir

  def dynamic(self):
    outflow = 0.1 * self.reservoir
    self.reservoir = self.reservoir - outflow + 0.5
    print self.reservoir

nrOfTimeSteps=100
myModel = MyFirstModel()
dynamicModel = DynamicFramework(myModel,nrOfTimeSteps)
dynamicModel.run()
```



Non-spatial model



```
from pcraster import *
from pcraster.framework import *

class MyFirstModel(DynamicModel):
  def __init__(self):
    DynamicModel.__init__(self)
    setclone('dem.map')

  def initial(self):
    conversionValue = 3.0
    self.reservoir = 30.0 / conversionValue
    print ('initial reservoir is: ', self.reservoir)

  def dynamic(self):
    outflow = 0.1 * self.reservoir
    self.reservoir = self.reservoir - outflow + 0.5
    print(self.reservoir)

nrOfTimeSteps=100
myModel = MyFirstModel()
dynamicModel = DynamicFramework(myModel,nrOfTimeSteps)
dynamicModel.run()
```

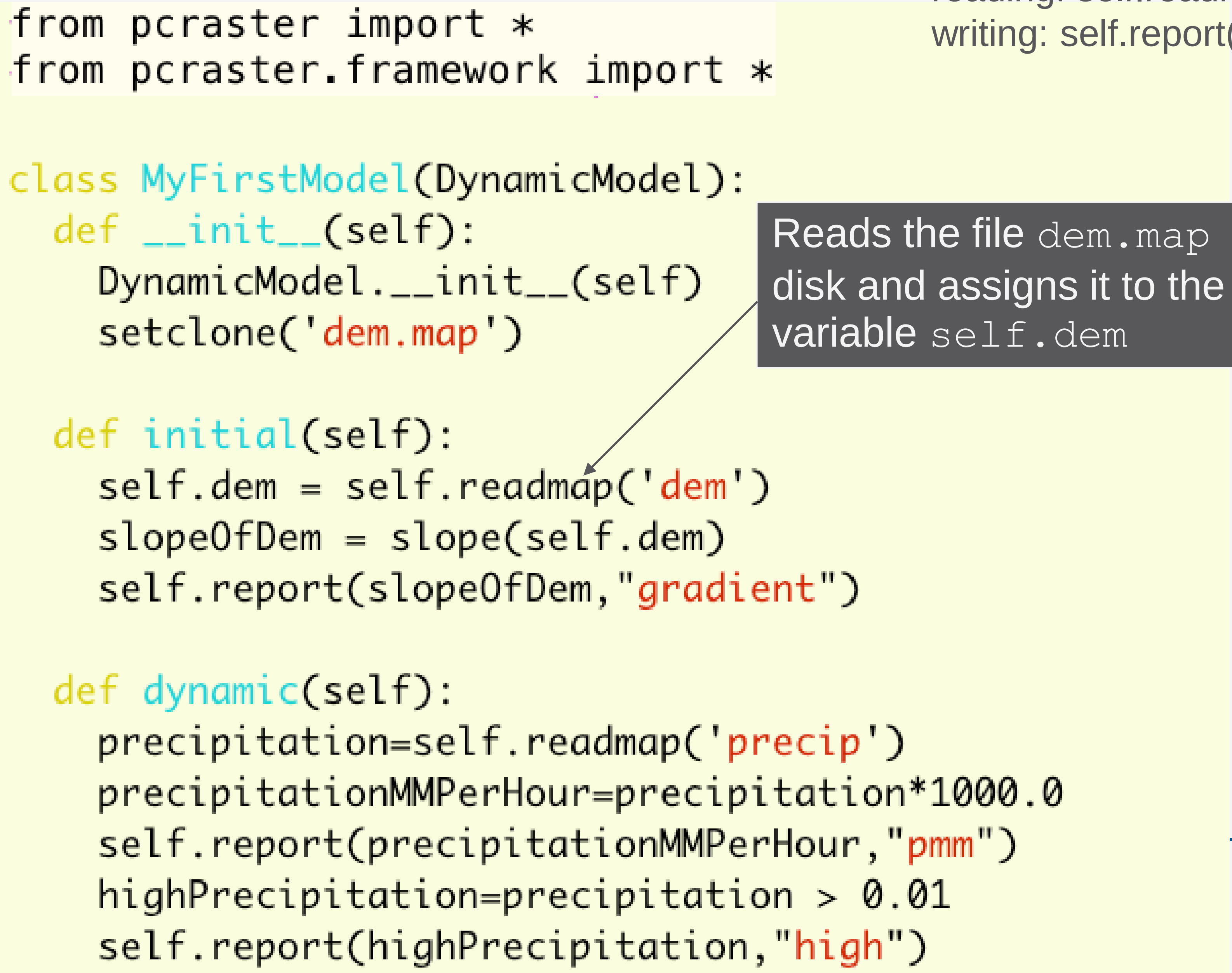
Defined in initial state

Used in dynamic → class

Result → value, not a map

Spatial model

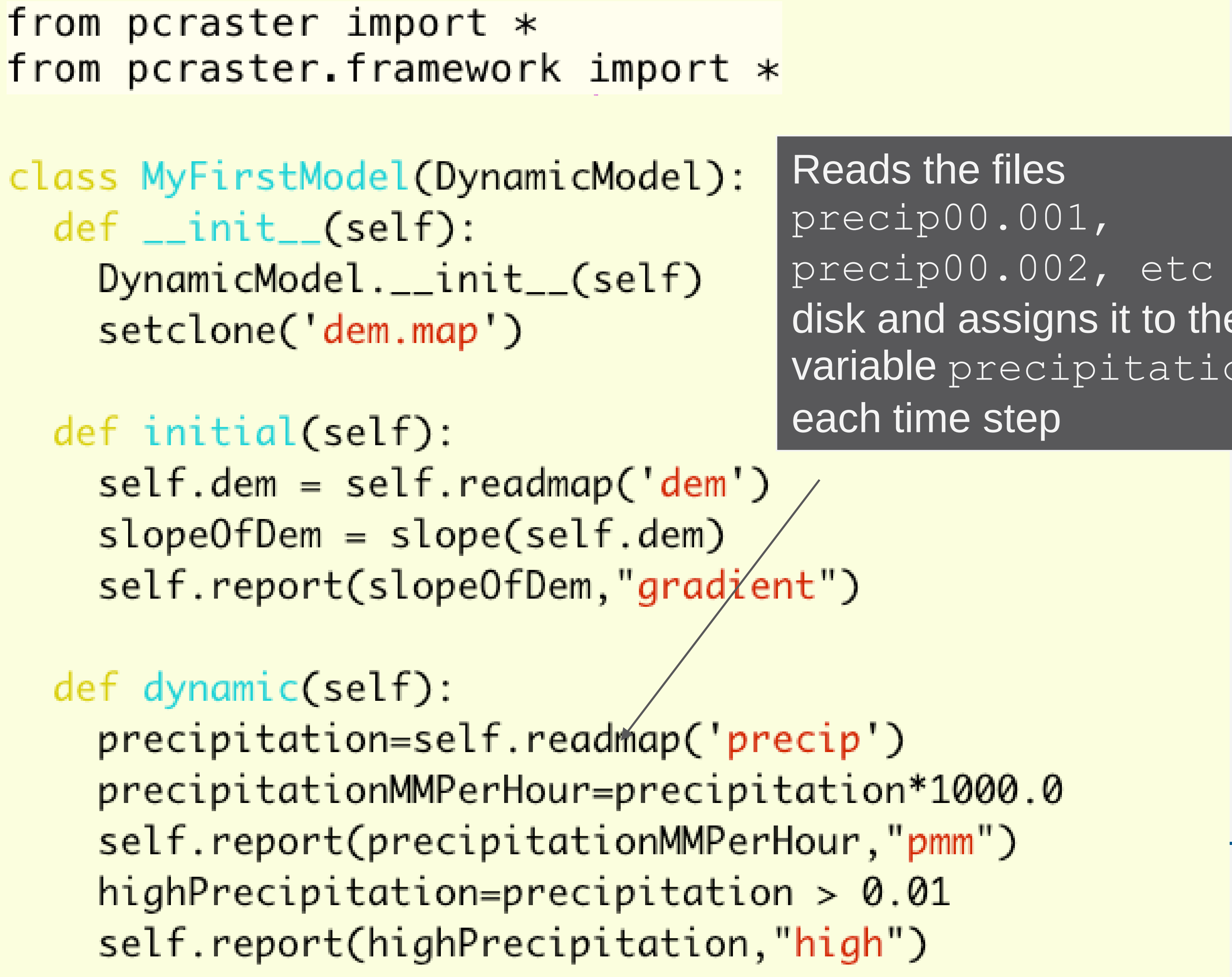
reading: self.readmap()
writing: self.report()



```
from pcraster import *
from pcraster.framework import *

class MyFirstModel(DynamicModel):
  def __init__(self):
    DynamicModel.__init__(self)
    setclone('dem.map')

  def initial(self):
    self.dem = self.readmap('dem')
    slopeOfDem = slope(self.dem)
    self.report(slopeOfDem,"gradient")

  def dynamic(self):
    precipitation=self.readmap('precip')
    precipitationMMPerHour=precipitation*1000.0
    self.report(precipitationMMPerHour,"pmm")
    highPrecipitation=precipitation > 0.01
    self.report(highPrecipitation,"high")
```

Reads the file `dem.map` from disk and assigns it to the variable `self.dem`

Spatial model

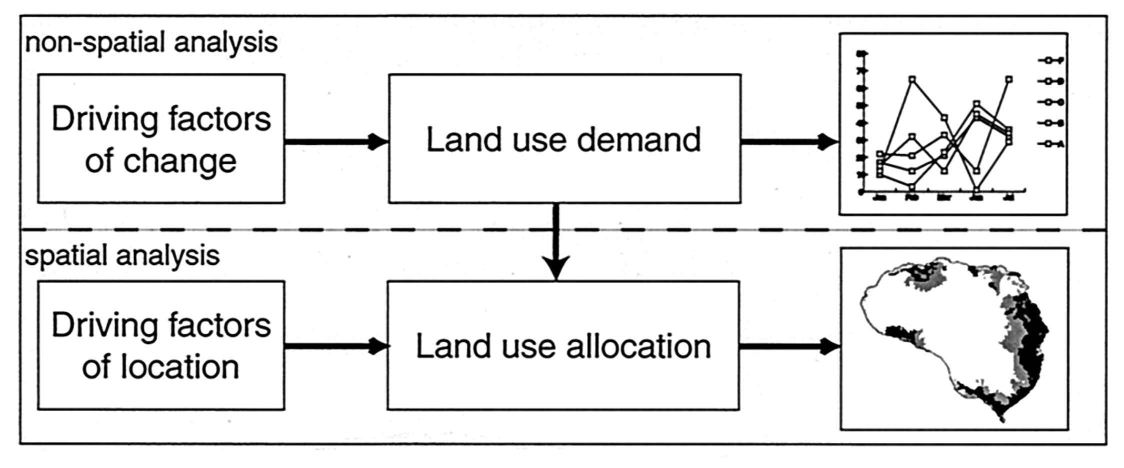


```python
from pcraster import *
from pcraster.framework import *

class MyFirstModel(DynamicModel):
  def __init__(self):
    DynamicModel.__init__(self)
    setclone('dem.map')

  def initial(self):
    self.dem = self.readmap('dem')
    slopeOfDem = slope(self.dem)
    self.report(slopeOfDem,"gradient")

  def dynamic(self):
    precipitation=self.readmap('precip')
    precipitationMMPerHour=precipitation*1000.0
    self.report(precipitationMMPerHour,"pmm")
    highPrecipitation=precipitation > 0.01
    self.report(highPrecipitation,"high")
```

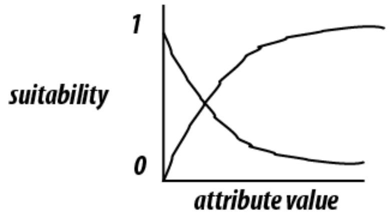
Reads the files `precip00.001`, `precip00.002`, `etc` from disk and assigns it to the variable `precipitation` for each time step

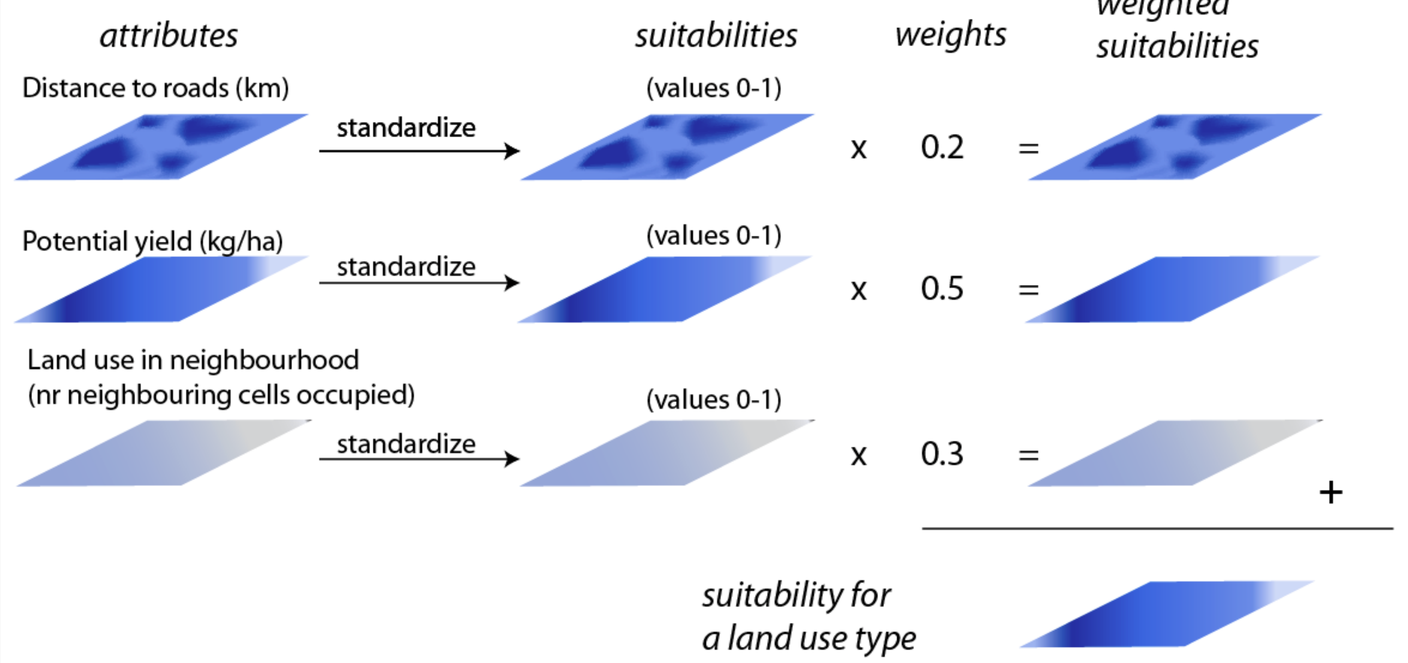
Example: land use change

Conceptual



Verburg et al. 2002

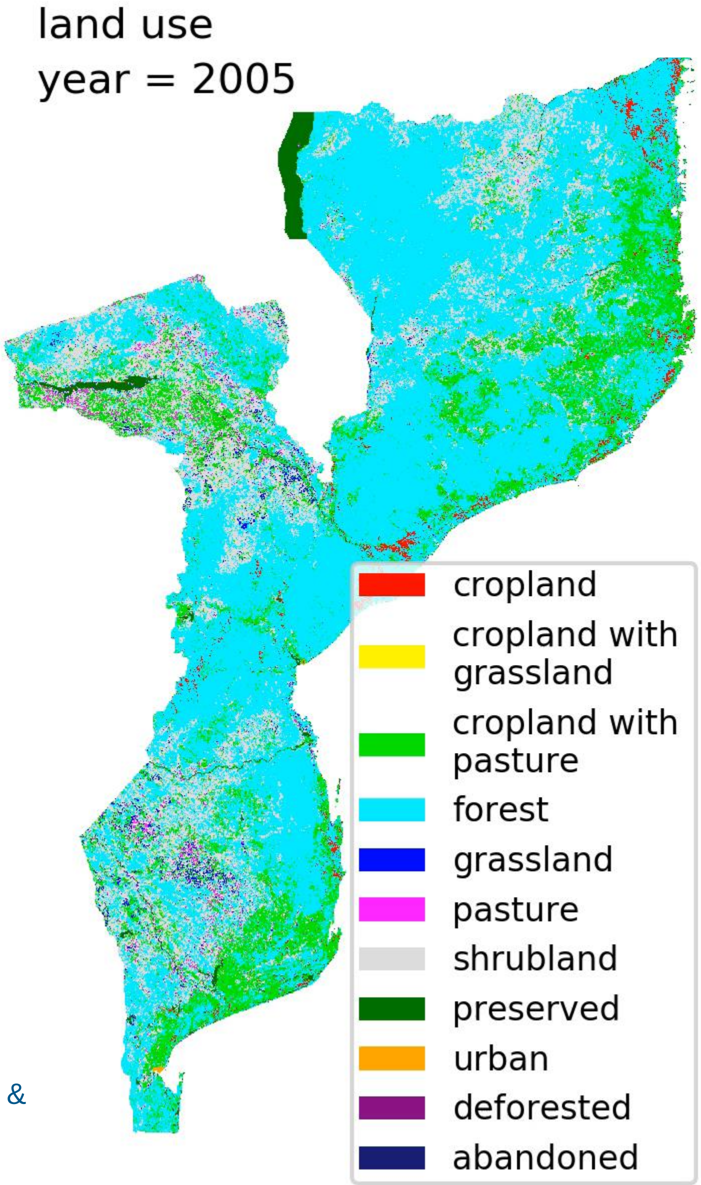




What goes in?

- initial()
- dynamic()

Result



[Verstegen et al., 2012, Computers, Env. & Urban Systems]

Demo

Questions?

Tutorial: fire spread modelling

https://github.com/JudithVerstegen/PCRaster_Python_tutorial

In this tutorial, you will build a fire spread model.

The tutorial is not written as a recipe, in the hope to encourage you to think, explore and learn.

Note the lists of required functions in sections 5 and 6.

I’ll be here to help you!