The integration of land change modeling framework FUTURES into GRASS GIS 7

Anna Petrasova, Vaclav Petras, Douglas A. Shoemaker, Monica A. Dorning, Ross K. Meentemeyer

NCSU OSGeo Research and Education Lab
Center for Geospatial Analytics
North Carolina State University

July 16, 2015
FUTures Urban-Regional Environment Simulation (FUTURES)

- stochastic, patch-based land change model
- simulates urban growth
- model accounts for location, quantity, and pattern of change
- incorporates positive feedbacks (new development attracts more development)
- allows spatial non-stationarity
FUTURES: Multilevel Simulations of Emerging Urban–Rural Landscape Structure Using a Stochastic Patch-Growing Algorithm

Ross K. Meentemeyer,* Wenwu Tang,* Monica A. Dorning,* John B. Vogler,* Nik J. Cunniffe,† and Douglas A. Shoemaker∗∗

*Department of Geography and Earth Sciences, University of North Carolina at Charlotte
†Department of Plant Sciences, University of Cambridge

We present a multilevel modeling framework for simulating the emergence of landscape spatial structure in urbanizing regions using a combination of field-based and object-based representations of land change. The FUTURE Urban-Regional Environment Simulation (FUTURES) produces regional projections of landscape patterns using coupled submodels that integrate nonstationary drivers of land change: per capita demand, site suitability, and the spatial structure of conversion events. Patches of land change events are simulated as discrete spatial objects using a stochastic region-growing algorithm that aggregates cell-level transitions based on empirical estimation of parameters that control the size, shape, and dispersion of patch growth. At each time step, newly constructed patches reciprocally influence further growth, which agglomerates over time to produce patterns of urban form and landscape fragmentation. Multilevel structure in each submodel allows drivers of land change to vary in space (e.g., by jurisdiction), rather than assuming spatial stationarity across a heterogeneous region. We applied FUTURES to simulate land development dynamics in the rapidly expanding metropolitan region of Charlotte, North Carolina, between 1996 and 2030, and evaluated spatial variation in model outcomes along an urban–rural continuum, including assessments of cell- and patch-based correctness and error. Simulation experiments reveal that changes in per capita land consumption and parameters controlling the distribution of development affect the emergent spatial structure of forests and farmlands with unique and sometimes counterintuitive outcomes. Key Words: fragmentation, land change model, nonstationarity, object-based, region growing algorithm.
The dark side of FUTURES

- no documentation
- know-how limited to certain colleagues
- code not reusable
- different versions of code
- no central storage for code
- extensive manual data preparation
- bad choice of programming language

```c
int getRandomeNumber() {
    return 4; // chosen by fair dice roll. // guaranteed to be random.
}
```

https://xkcd.com/221
Open source approach

We hope FUTURES can make broader impact in land change community...

...but more than just releasing the code online is needed.

We can take example from many open source projects which have established ways and technologies for

- collaboration
- code development and distribution
- providing documentation, user support
Integration

Standalone vs. integrated into open source GIS:

+ all standard GIS tools and algorithms available
+ distribution and installation across all platform solved
+ established ways to provide documentation
+ existing user base with support

- less flexibility in certain aspects
- need to learn to use the particular GIS
GRASS GIS

Why integrate FUTURES into GRASS GIS 7?

- able to process large datasets
- runs on HPC clusters
- efficient I/O libraries
- modular architecture – modules in C/C++ and Python, R
- automatically generated CLI and GUI
- addons hosted on OSGeo servers
  - easy installation: `> g.extension r.futures`
  - daily compiled binaries for Windows (thanks to M. Landa, FCE CTU in Prague)
  - maintained by community and developers (API changes)
FUTURES conceptual diagram

- **INPUT DATA**
  - Environmental, infrastructural, socio-economic predictors
  - Historical development patterns
  - Historical population & projections

- **SUB-MODELS**
  - Development potential
    - Location
  - Patch growing algorithm
    - Spatial structure
  - Land demand
    - Quantity

- **OUTPUTS**
  - Development outcomes

- Connections:
  - Where?
  - Size & shape?
  - How much land?
  - How many people?
  - Update development pressure
FUTURES conceptual diagram

**INPUT DATA**
- ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

**SUB-MODELS**
- DEVELOPMENT POTENTIAL
  - Location
- PATCH GROWING
  - ALGORITHM
    - Spatial Structure
- LAND DEMAND
  - Quantity

**OUTPUTS**
- DEVELOPTMENT OUTCOMES
FUTURES conceptual diagram

**INPUT DATA**
- ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

**SUB-MODELS**
- DEVELOPMENT POTENTIAL
  - [Location]
- PATCH GROWING ALGORITHM
  - [Spatial Structure]
- LAND DEMAND
  - [Quantity]

**OUTPUTS**
- DEVELOPMENT OUTCOMES
- \[ \text{Size & Shape?} \]
- \[ \text{Where?} \]
- \[ \text{How much land?} \]
- \[ \text{How many people?} \]

\[ \text{Update Development Pressure} \]
Patch Growing Algorithm

- stochastic algorithm
- converts land in discrete patches

- implemented in C/C++
- computationally demanding
- inputs are patch characteristics (distribution of patch sizes and compactness) derived from historical data
Patch Growing Algorithm

- stochastic algorithm
- converts land in discrete patches

- implemented in C/C++
- computationally demanding
- inputs are patch characteristics (distribution of patch sizes and compactness) derived from historical data
Patch Growing Algorithm

- stochastic algorithm
- converts land in discrete patches

- implemented in C/C++
- computationally demanding
- inputs are patch characteristics (distribution of patch sizes and compactness) derived from historical data
Patch Growing Algorithm

- stochastic algorithm
- converts land in discrete patches

- implemented in C/C++
- computationally demanding
- inputs are patch characteristics (distribution of patch sizes and compactness) derived from historical data
Patch Growing Algorithm

- stochastic algorithm
- converts land in discrete patches

- implemented in C/C++
- computationally demanding
- inputs are patch characteristics (distribution of patch sizes and compactness) derived from historical data
DEMAND submodel

- estimates the rate of per capita land consumption specific to each subregion
- extrapolates between historical changes in population and land conversion

- inputs are historical landuse, population data, population projection
- ordinary least squares regression (linear or logarithmic relationships)
- informally implemented as R scripts and ArcGIS workflows
FUTURES conceptual diagram

**INPUT DATA**
- ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

**SUB-MODELS**
- DEVELOPMENT POTENTIAL
  - Location
- PATCH GROWING ALGORITHM
  - Spatial Structure
- LAND DEMAND
  - Quantity
- SIZE & SHAPE?
- HOW MUCH LAND?
- WHERE?
- HOW MANY PEOPLE?

**OUTPUTS**
- DEVELOPMENT OUTCOMES
- UPDATE DEVELOPMENT PRESSURE

---

Location

PATCH GROWING ALGORITHM

DEVELOPMENT OUTCOMES

INPUT DATA

SUB-MODELS

OUTPUTS
POTENTIAL submodel

■ multilevel logistic regression for development suitability
■ accounts for variation among subregions (for example policies in different counties)

■ inputs are uncorrelated predictors (distance to roads and development, slope, …)
■ original implementation
  ■ data preparation in ArcGIS
  ■ regression coefficients derived in R
  ■ potential and probability surface recomputed after each step in the main C code
Implementation overview

INPUT DATA
- ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

SUB-MODELS
- DEVELOPMENT POTENTIAL
  [Location]
- PATCH GROWING ALGORITHM
  [Spatial Structure]
- LAND DEMAND
  [Quantity]

OUTPUTS
- DEVELOPMENT OUTCOMES

Flow:
- Where?
- Size & Shape?
- How much land?
- How many people?

Update Development Pressure
Implementation overview

- **INPUT DATA**
  - ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
  - HISTORICAL DEVELOPMENT PATTERNS
  - HISTORICAL POPULATION & PROJECTIONS

- **SUB-MODELS**
  - DEVELOPMENT POTENTIAL
    - Location
  - PATCH GROWING
    - Spatial Structure
  - LAND DEMAND
    - Quantity

- **OUTPUTS**
  - DEVELOPMENT OUTCOMES

Where?
Size & Shape?
How much land?
How many people?

r.futures.pga

Update Development Pressure
r.futures.pga

Originally standalone C++ code converted into GRASS GIS addon
- efficient raster reading and writing
- fixed segfaults
- general cleanup (dead code), revised input options

```
r.futures.pga [-s] developed=name predictors=name[,name,...] 
  demand=name devpot_params=name discount_factor=value 
  ... 
incentive_table=name [constrain_weight=name] [random_seed=value] 
output=name [output_series=basename] [--overwrite] [--help] 
  [--verbose] [--quiet] [--ui]
```
Originally standalone C++ code converted into GRASS GIS addon
- efficient raster reading and writing
- fixed segfaults
- general cleanup (dead code), revised input options
r.futures.pga

Originally standalone C++ code converted into GRASS GIS addon

- efficient raster reading and writing
- fixed segfaults
- general cleanup (dead code), revised input options
r.futures.pga

Originally standalone C++ code converted into GRASS GIS addon

- efficient raster reading and writing
- fixed segfaults
- general cleanup (dead code), revised input options
r.futures.pga

Originally standalone C++ code converted into GRASS GIS addon
- efficient raster reading and writing
- fixed segfaults
- general cleanup (dead code), revised input options

Simulates landuse change using FUTURE Urban-Regional Environment Simulation (FUTURES). Module uses Patch-Growing Algorithm (PGA) to simulate urban-rural landscape structure development.

Additionally, parameter constrain_weight (raster map from 0 to 1) enables to include policies (such as new regulations or fees) which limit the development in certain areas. The probability surface is simply multiplied by the constrain_weight values, which results in decreased site suitability in areas, where the constrain_weight values are lower than 1.

Output
After the simulation ends, raster specified in parameter output is written. If optional parameter output_series is specified, additional output is a series of raster maps for each step. Cells with value 0 represents the initial development, values >= 1 then represent the step in which the cell was developed. Undeveloped cells have value -1.

r.futures.pga developed=<required> predictors=<required> demand=<required> devpot_params=<required> discount_factor=<required> compactness
New addon written in Python for automated calibration of patch characteristics

- provides optimal patch parameters for r.futures.pga by comparing observed land change pattern with the simulated pattern
- runs in multiple processes

Higher compactness

Lower compactness
Implementation overview

INPUT DATA
- ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

SUB-MODELS
- DEVELOPMENT POTENTIAL
  [Location]
- PATCH GROWING
  [Spatial Structure]
- LAND
  [Quantity]
- r.futures.calib
- r.futures.pga
- r.futures.demand

OUTPUTS
- DEVELOPMENT OUTCOMES

Where?
- How much land?
- How many people?

Update Development Pressure
Implementation overview

INPUT DATA
- ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

SUB-MODELS
- r.futures.potential
  - Potential [location]
  - Update Development Pressure
- r.futures.calib
  - Patch Growing [Spatial Structure]
- r.futures.pga
  - Development OUTCOMES
- LAND
  - Demand [Quantity]
  - How many people?
  - How much land?

OUTPUTS
Implementation still needed for

- `r.futures.demand` – Python and Numpy
- `r.futures.potential`
  - R wrapped in Python to provide standard GRASS interface
  - using `dredge` from R package MuMIn\(^1\) for automated model selection (read the `dredge` disclaimer before using)

DEMAND and POTENTIAL statistical models can be implemented in different ways, but we need reference implementation.

\(^1\)http://cran.r-project.org/web/packages/MuMIn/MuMIn.pdf
Implementation overview

INPUT DATA
- ENVIRONMENTAL, INFRASTRUCTURAL, SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

SUB-MODELS
- r.futures.potential
- r.futures.calib
- r.sample.category
- r.futures.pga
- PATCH GROWING ALGORITHM
  - [Spatial Structure]

OUTPUTS
- DEVELOPMENT OUTCOMES
  - LAND
    - r.futures.demand
      - [Quantity]
  - r.futures.potential
  - r.sample.category
  - r.futures.calib

Where?
- How much land?
- How many people?

Update Development Pressure

[Location]
Implementation overview

INPUT DATA
- ENVIRONMENTAL
- INFRASTRUCTURAL
- SOCIO-ECONOMIC
- PREDICTORS

SUB-MODELS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS
- PATTERNS
- PATCH GROWING ALGORITHM

OUTPUTS
- PREDICTORS
- ENVIRONMENTAL,
- INFRASTRUCTURAL,
- SOCIO-ECONOMIC
- PREDICTORS
- PATTERNS
- PATCH GROWING ALGORITHM

DEVELOPMENT OUTCOMES
- DEVEL OPMENT
- OUT CO   MES
- SIZE & SHAPE?
- LAND
- DEMAND
- [QUANTITY]

EXAMPLES:
- r.futures.potential
- r.sample.category
- r.futures.devpressure
- r.futures.pga
- r.futures.calib
- r.futures.demand
- [Spatial Structure]

Other examples:
r.futures.pga
r.sample.category
r.futures.devpressure
r.futures.potential
Data preprocessing for POTENTIAL

Two new Python modules needed for FUTURES data preprocessing, but can be used for other applications:

**r.sample.category**
Create sampling points from each category in a raster map

**r.futures.devpressure**
Moving window computation of distance decay effect

(source: r.sample.category manual page)
Implementation overview

INPUT DATA
- ENVIRONMENTAL
- INFRASTRUCTURAL
- SOCIO-ECONOMIC PREDICTORS
- HISTORICAL DEVELOPMENT PATTERNS
- HISTORICAL POPULATION & PROJECTIONS

SUB-MODELS
- r.futures.potential
- r.futures.devpressure
- r.futures.calib
- r.futures.pga

OUTPUTS
- DEVELOPMENT OUTCOMES
  - LAND: r.futures.demand [Quantity]
  - r.sample.category
  - r.futures.potential

Where?
- Conceptual vs. user view

How much land?
- How many people?

Update Development Pressure

Conceptual vs. user view
Next steps

Finish `r.futures.demans` and `r.futures.potential`
More documentation needed:

- tutorial
- sample dataset – Raleigh for consistency with GRASS GIS
  North Carolina sample dataset
  - use publicly available data (National Land Cover Database)
What do I need to run it?

1. Gather data (historical urban development, population projection, predictors)
2. Use a high-end workstation
3. Installation is simple, works on all platforms:
   > g.extension r.futures

Code available online:
trac.osgeo.org/grass/browser/grass-addons/grass7/raster/r.futures/
Take home message

1. Integrate your work into FOSS
2. Automate tasks
3. Document!

Thank you!

Contact: akratoc@ncsu.edu
FUTURES: Multilevel Simulations of Emerging Urban–Rural Landscape Structure Using a Stochastic Patch-Growing Algorithm

Simulating urbanization scenarios reveals tradeoffs between conservation planning strategies