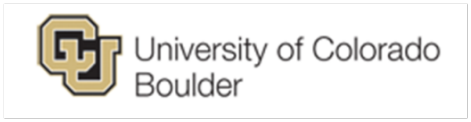
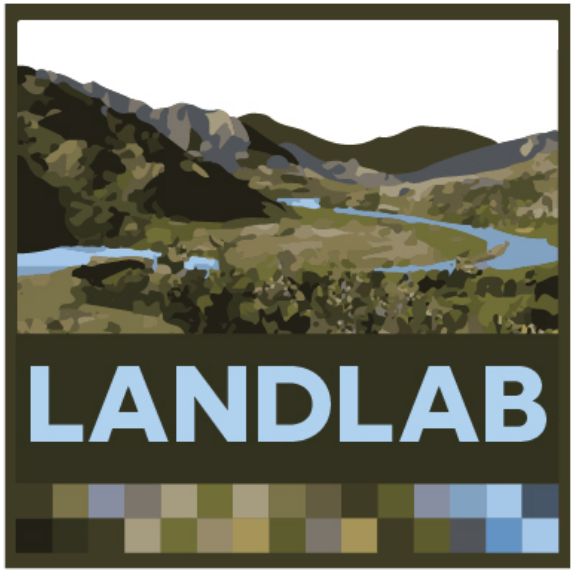




Modeling Earth-Surface Dynamics with Landlab 1.0

University of Colorado & CSDMS:

Greg Tucker, Eric Hutton, Jennifer Knuth , Katy Barnhardt

Tulane University:

Nicole Gasparini, Jordan Adams, Margaux Mouchene, Nathan Lyons

University of Washington:

Sai S. Nudurupati, Erkan Istanbulluoglu, Christina Bandaragoda and Ronda Strauch

Cardiff University:

Daniel Hobley

Geosci. Model Dev., 9, 823–839, 2016
www.geosci-model-dev.net/9/823/2016/
doi:10.5194/gmd-9-823-2016

Geoscientific Model Development
Open Access
EGU

CellLab-CTS 2015: continuous-time stochastic cellular automaton modeling using Landlab

Gregory E. Tucker[1,2], Daniel E. J. Hobley[1,2], Eric Hutton[3], Nicole M. Gasparini[4], Erkan Istanbulluoglu[5], Jordan M. Adams[4], and Sai Siddartha Nudurupati[5]

Earth Surf. Dynam., 5, 21–46, 2017
www.earth-surf-dynam.net/5/21/2017/
doi:10.5194/esurf-5-21-2017

Earth Surface Dynamics
Open Access
EGU

Creative computing with Landlab: an open-source toolkit for building, coupling, and exploring two-dimensional numerical models of Earth-surface dynamics

Daniel E. J. Hobley[1,2,3], Jordan M. Adams[4], Sai Siddhartha Nudurupati[5], Eric W. H. Hutton[6], Nicole M. Gasparini[4], Erkan Istanbulluoglu[5], and Gregory E. Tucker[1,2]

Geosci. Model Dev., 10, 1645–1663, 2017
www.geosci-model-dev.net/10/1645/2017/
doi:10.5194/gmd-10-1645-2017

Geoscientific Model Development
Open Access
EGU

The Landlab v1.0 OverlandFlow component: a Python tool for computing shallow-water flow across watersheds

Jordan M. Adams[1], Nicole M. Gasparini[1], Daniel E. J. Hobley[2], Gregory E. Tucker[3,4], Eric W. H. Hutton[5], Sai S. Nudurupati[6], and Erkan Istanbulluoglu[6]

Overview

- **Introduction to Landlab, http://landlab.github.io/#/**
  - **Other important websites coming.**

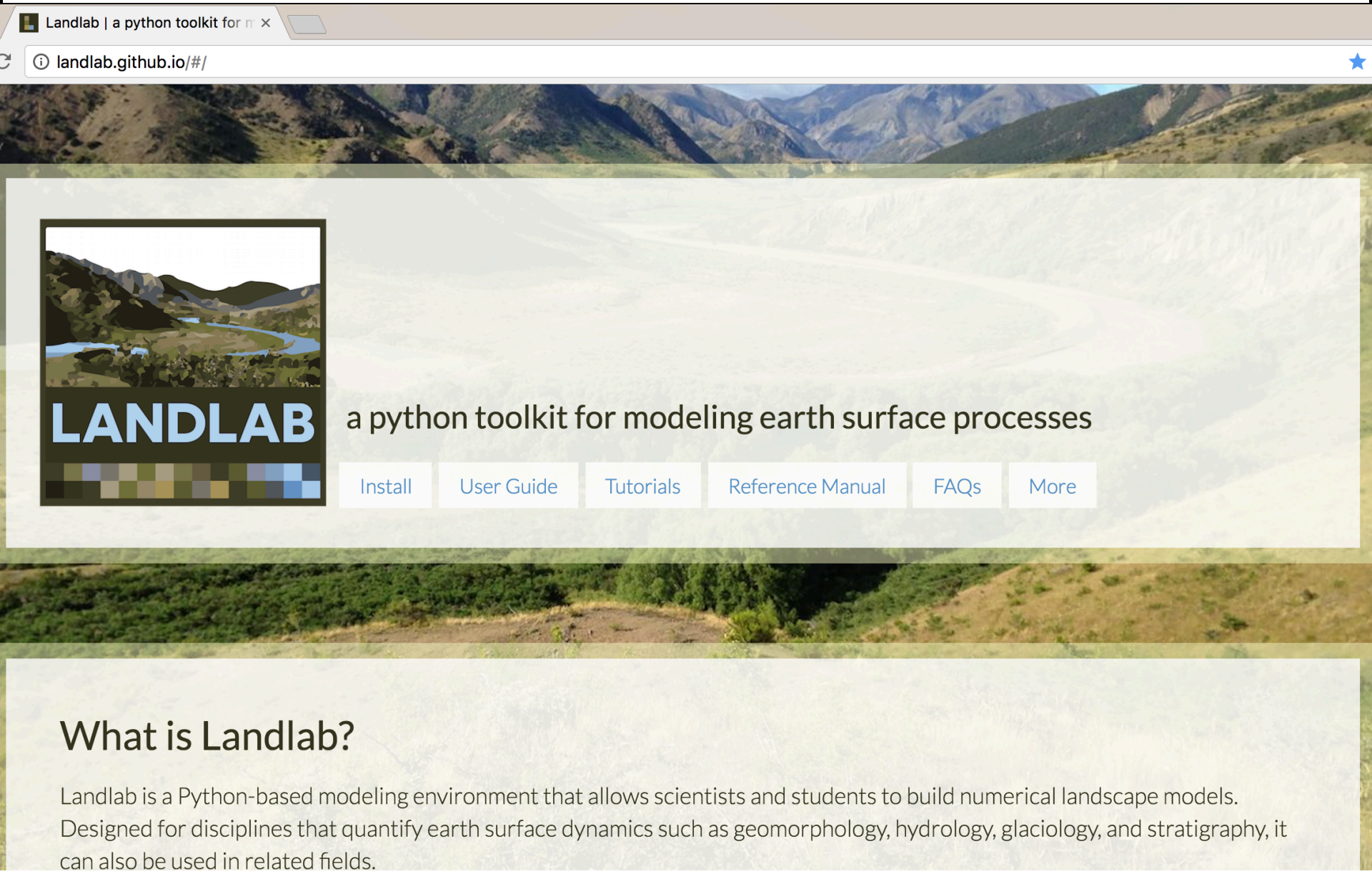
- **Running Landlab on Hydroshare, https://www.hydroshare.org/**

  - **"Traditional" landscape evolution model**

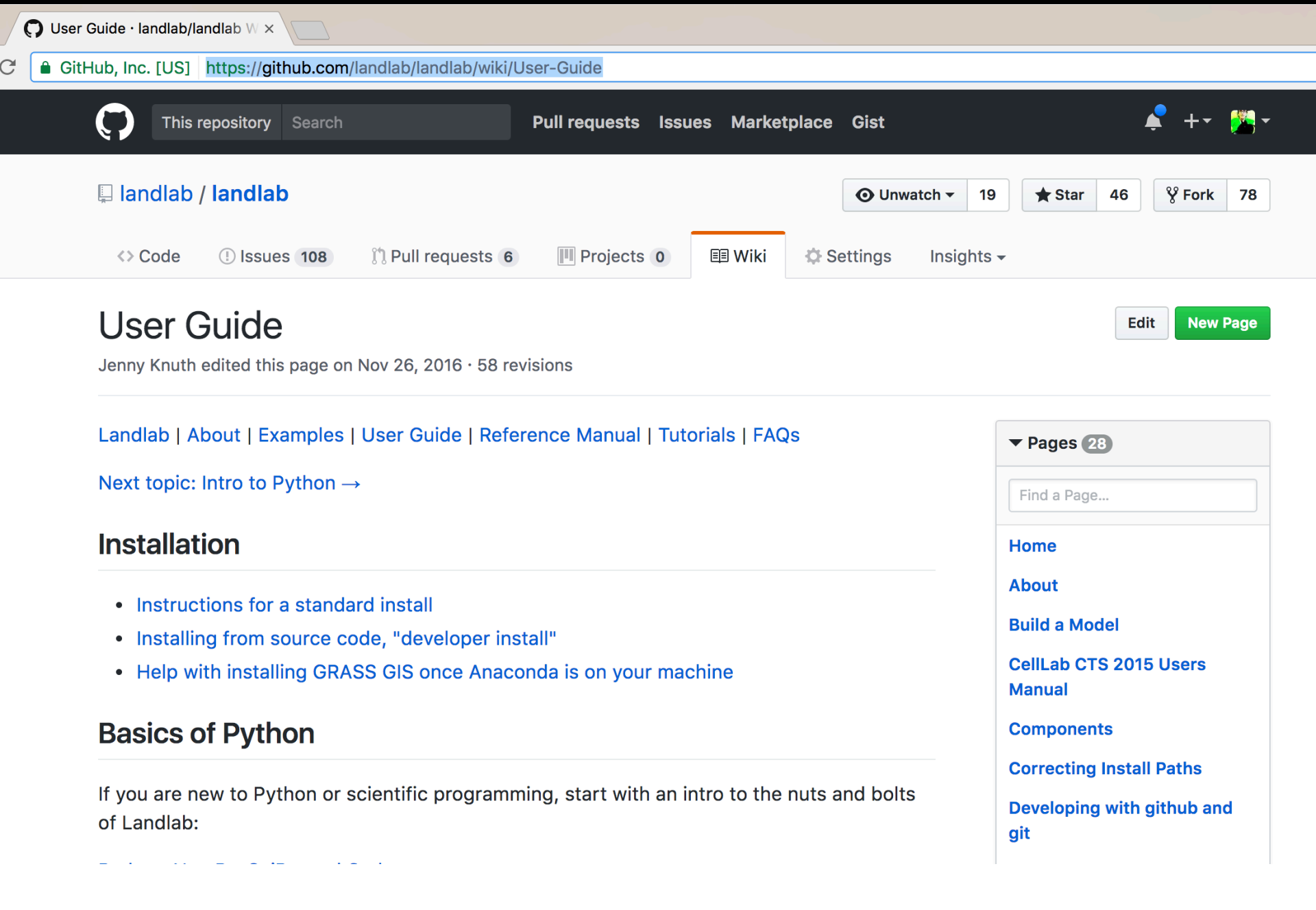
  - **2D flood routing and erosion model**

  - **Ecohydrology tutorial and model(s)**

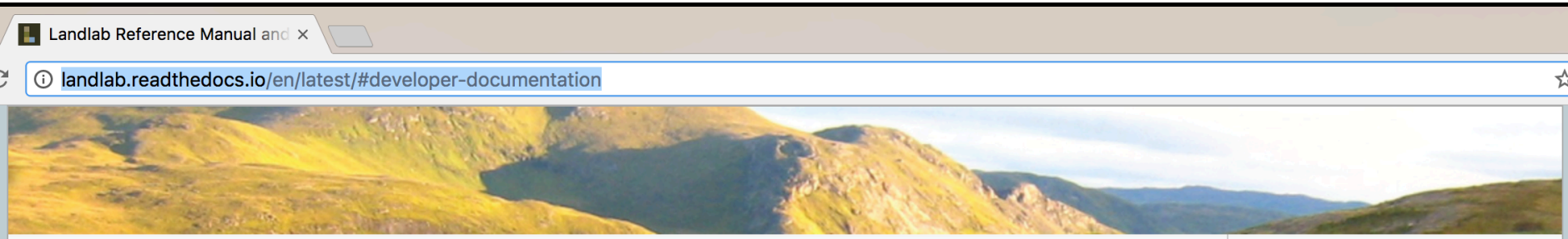
landlab.github.io



https://github.com/landlab/landlab/wiki/User-Guide



http://landlab.readthedocs.io/en/latest/#developer-documentation



Find Landlab's User Guide on the Landlab Wiki

Landlab Reference Manual and API Documentation

A guide to Landlab's classes and code.

Grids

Grid types

As of Landlab version 0.2, there are four types of Landlab grid:

- Raster
- Voronoi-Delaunay
- Hex
- Radial

The base class is *ModelGrid* with subclasses *RasterModelGrid* and *VoronoiDelaunayGrid*.

VoronoiDelaunayGrid has two further specialized subclasses: *HexModelGrid* and *RadialModelGrid*.

Methods and properties common to all grids

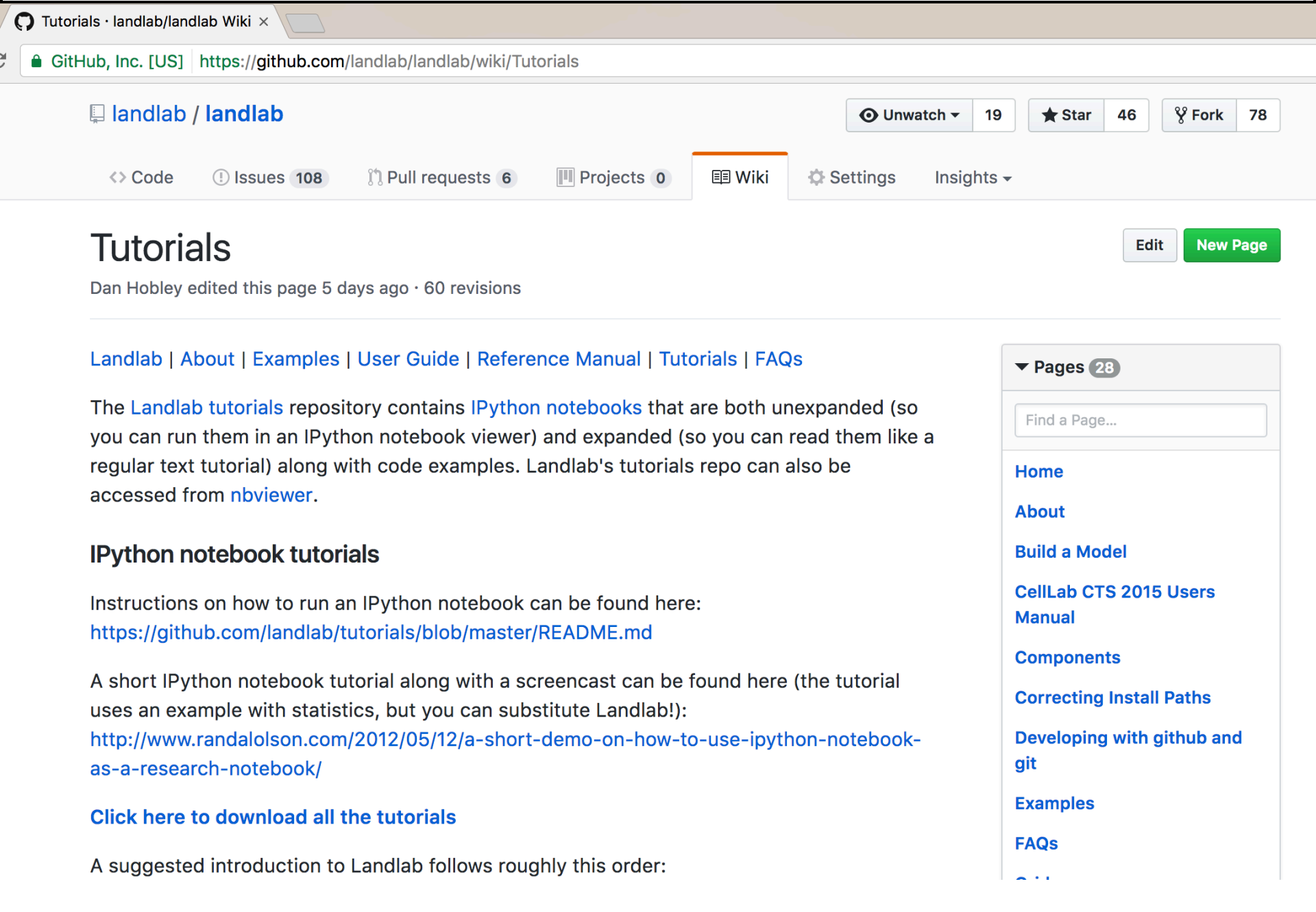
- General class methods and attributes of the *landlab.grid.base* module
- Getting Information about a Grid
  - Information about the grid as a whole
  - Information about nodes
  - Information about links
  - Information about cells

INDEX

Search

Table Of Contents

https://github.com/landlab/landlab/wiki/Tutorials



The first example uses set_watershed_boundary_condition which finds the outlet for the user.

- First import what we need.

In [ ]:
```
from landlab import RasterModelGrid
import numpy as np
from landlab.plot.imshow import imshow_grid
%matplotlib inline
```

- Now we create a 5 by 5 grid with a spacing (dx and dy) of 1.
- We also create an elevation field with value of 1. everywhere, except at the outlet, where the elevation is 0. In this case the outlet is in the middle of the bottom row, or at at location (0,2) and has a node id of 2.

In [ ]:
```
mg1 = RasterModelGrid((5,5), 1.)
z1 = mg1.add_ones('node','topographic__elevation')
mg1.at_node['topographic__elevation'][2] = 0.
mg1.at_node['topographic__elevation']
```

- The set_watershed_boundary_condition in RasterModelGrid will find the outlet of the watershed.
- This method takes the node data, in this case z, and, optionally the no_data value.
- This method sets all nodes that have no_data values to closed boundaries.
- This example does not have any no_data values, which is fine.
- In this case, the code will set all of the perimeter nodes as CLOSED_BOUNDARY (boundary status 4) in order to create this boundary around the core nodes.
- The exception on the perimeter is node 2 (with elevation of 0). Although it is on the perimeter, it has a value and it has the lowest value. So in this case node 2 will be set as FIXED_VALUE_BOUNDARY (boundary status 1).
- The rest of the nodes are set as a CORE_NODE (boundary status 0)

In [ ]:
```
mg1.set_watershed_boundary_condition(z1)
```

- Check to see that node status were set correctly.
- imshow_grid will default to not plot the value of CLOSED_BOUNDARY nodes, which is why we override this below with the option color_for_closed

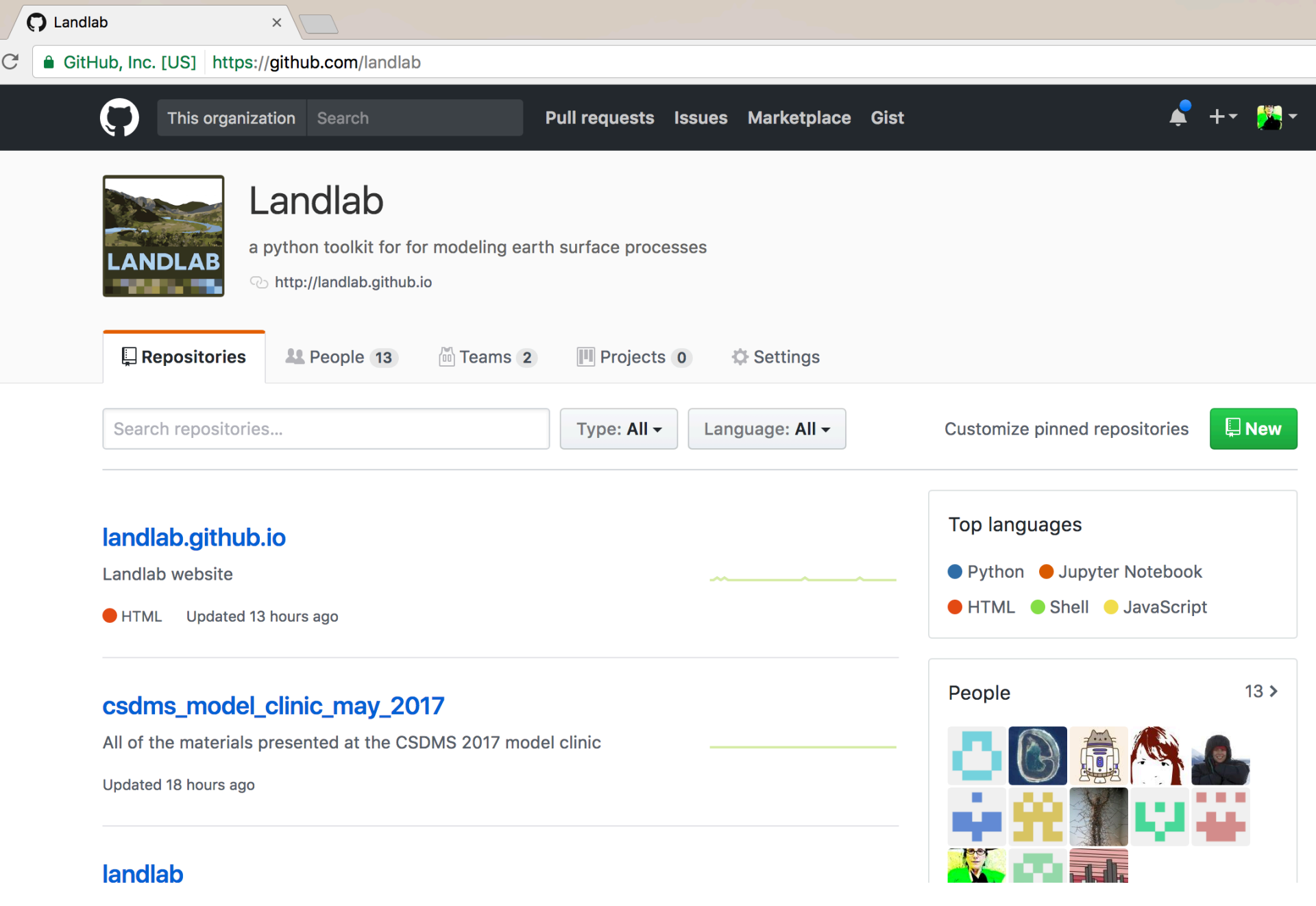
In [ ]:
```
imshow_grid(mg1, mg1.status_at_node, color_for_closed='blue')
```

The second example uses set_watershed_boundary_condition_outlet_coords

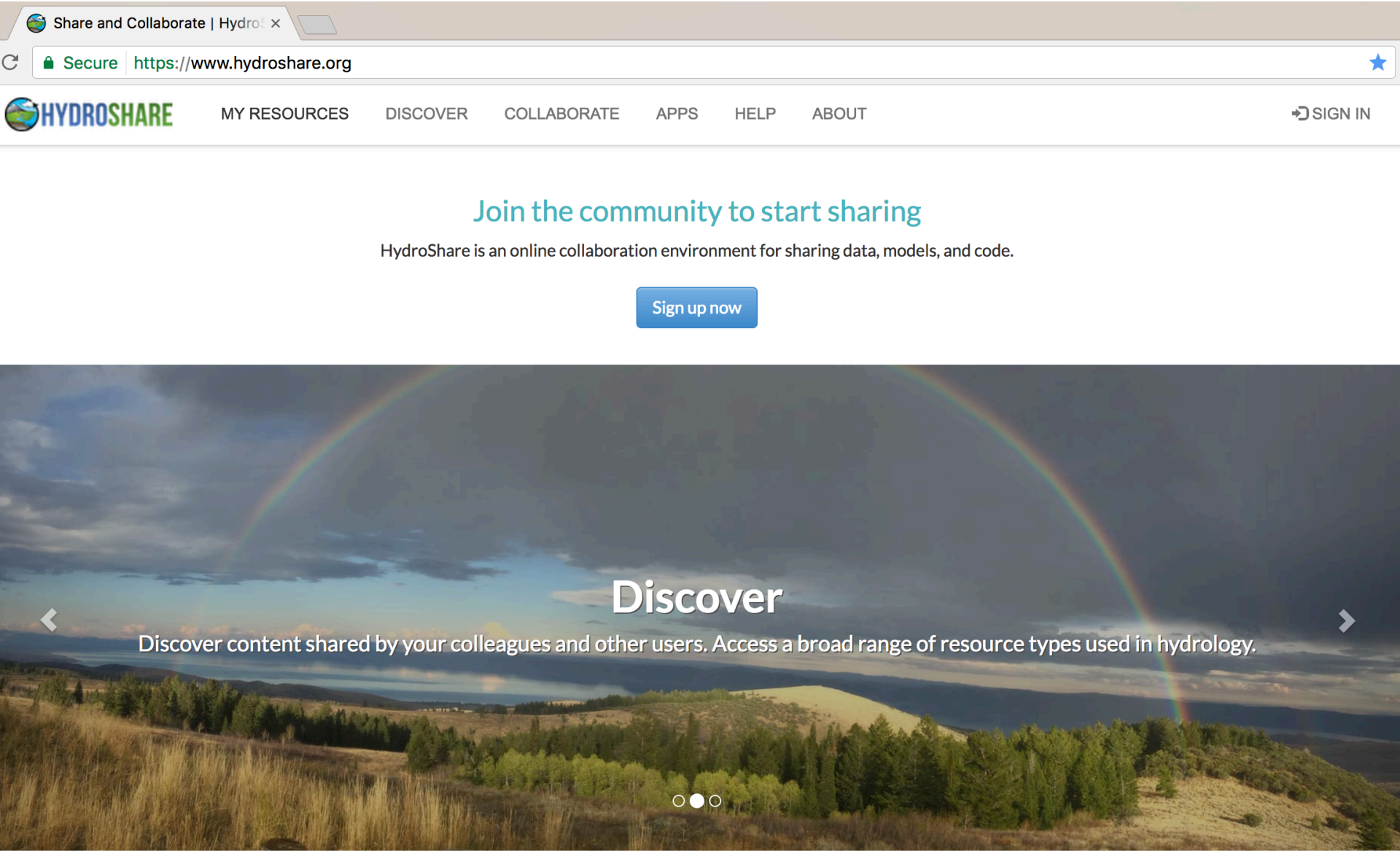
- In this case the user knows the coordinates of the outlet node.
- First instantiate a new grid, with new data values.

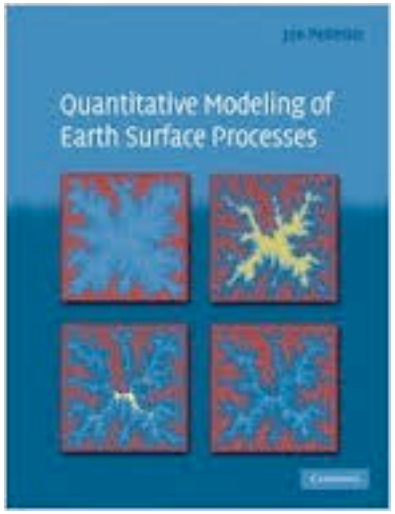
In [ ]:
```
mg2 = RasterModelGrid((5,5), 10.)
z2 = mg2.add_ones('node','topographic__elevation')
mg2.at_node['topographic__elevation'][1] = 0.
mg2.at_node['topographic__elevation']
```

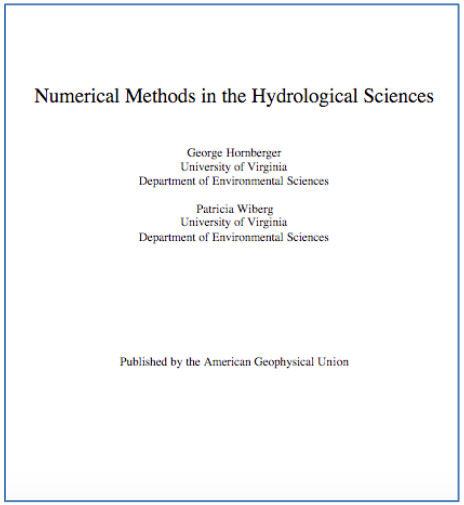
https://github.com/landlab



https://www.hydroshare.org/






Numerical Methods in the Hydrological Sciences

George Hornberger
University of Virginia
Department of Environmental Sciences

Patricia Wiberg
University of Virginia
Department of Environmental Sciences

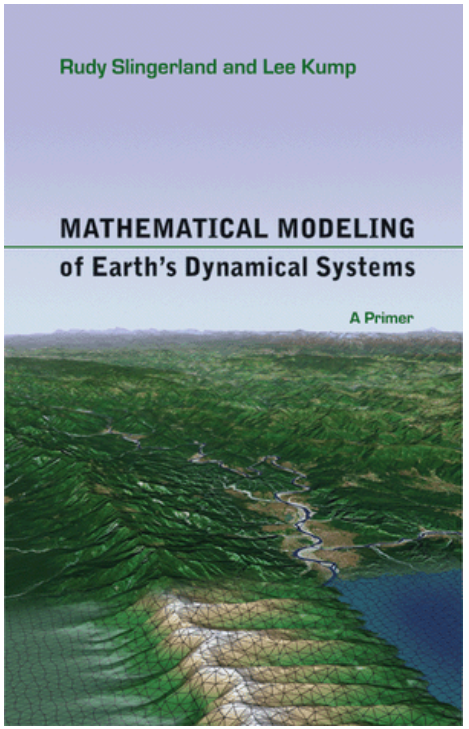
Published by the American Geophysical Union

SOFTWARE

NUMERICAL ALGORITHM

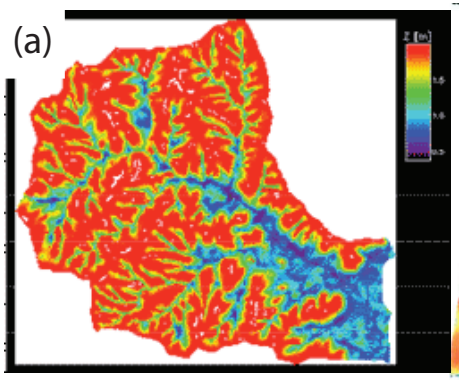
DYNAMICAL MODEL

NATURE

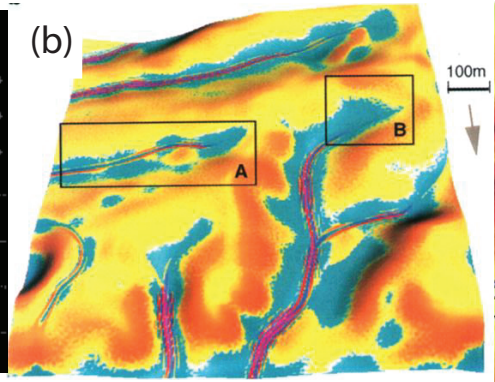


2D models of earth-surface processes

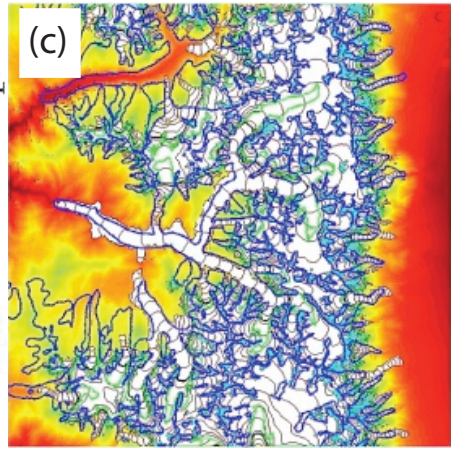
CATCHMENT HYDROLOGY
(Ivanov et al., 2004)

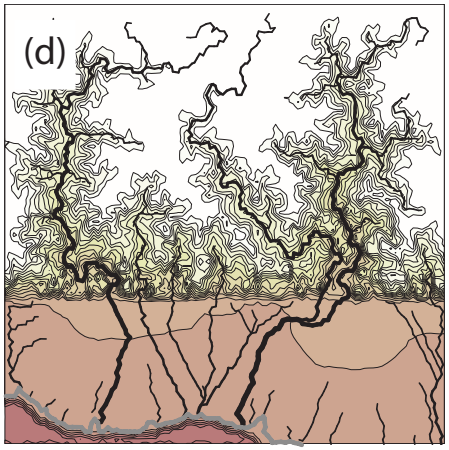


SOIL EROSION
(Mitas and Mitasova, 1998)

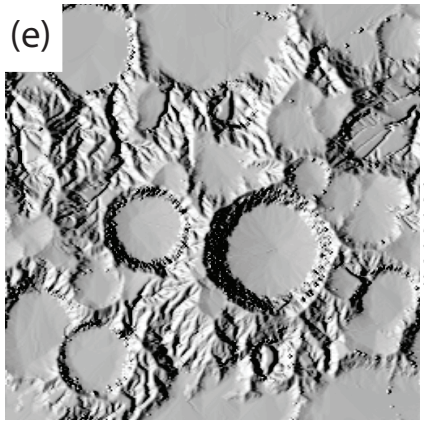


GLACIER DYNAMICS
(Kessler et al., 2006)

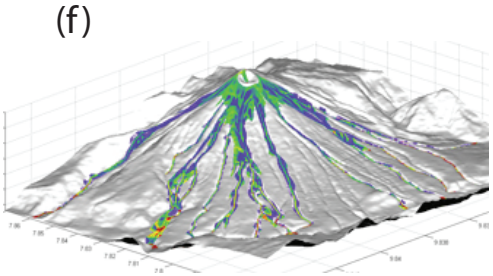




LANDSCAPE EVOLUTION
(Tucker and Hancock, 2010)



IMPACT CRATERING AND DEGRADATION
(Howard, 2007)

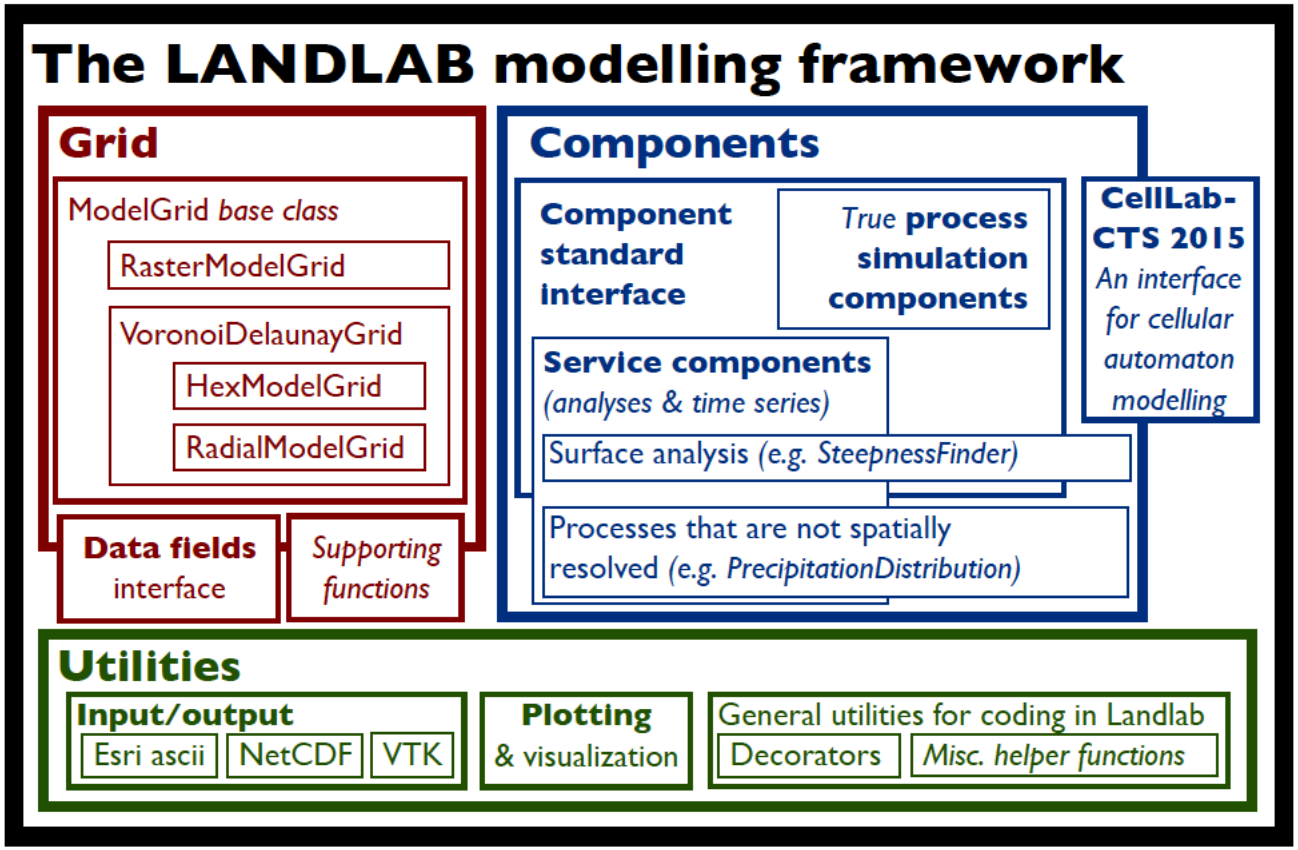


LAVA FLOWS
(Kelfoun et al., 2009)

What is Landlab?

- An **open source Python-language programming library**
- Supports efficient **creation and/or coupling of 2D numerical models**
- Geared toward (but not limited to) earth-surface dynamics

What is Landlab?



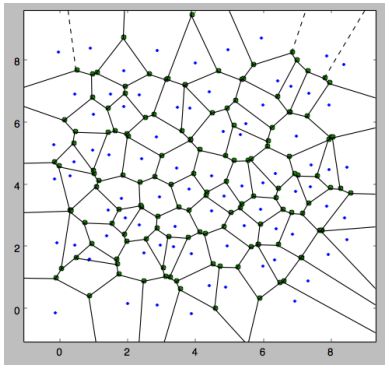
(Hobley et al., 2017, ESURF)

What Landlab provides

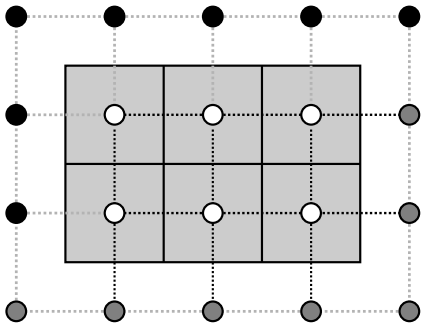
1. Grid creation and management

- **Create** a structured or unstructured **grid** in one or a few lines of code
- **Attach data** to grid elements
  - Facilitates staggered-grid schemes
  - Passing the grid = passing the data

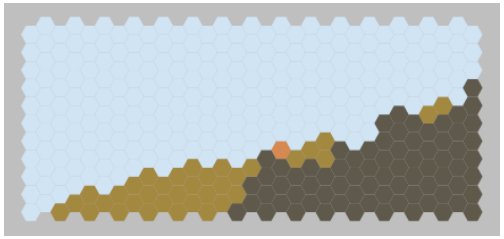
VORONOI / DELAUNAY



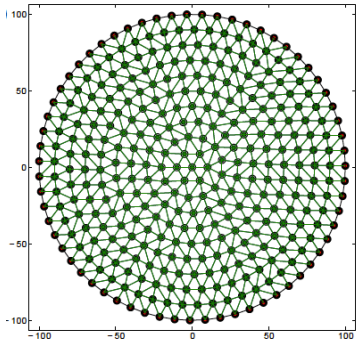
RASTER



HEXAGONAL



RADIAL



What Landlab provides

2. **Components and coupling framework for components**

- A ***component*** models a **single process** (e.g., lithosphere flexure, incident solar radiation, flow routing across terrain)
- Components have a **standard interface** and can be combined by writing a **short Python script** (model, driver)
- **Initialized** using input grid and parameters
- **Update** relevant **'fields'** on the grid
- Components also include analytical tools for analyzing landscapes (e.g. channel steepness, hillslope length, …)

What Landlab provides

(Hobley et al., 2017, ESURF)

Table 5. Components available in Landlab v.1.0.

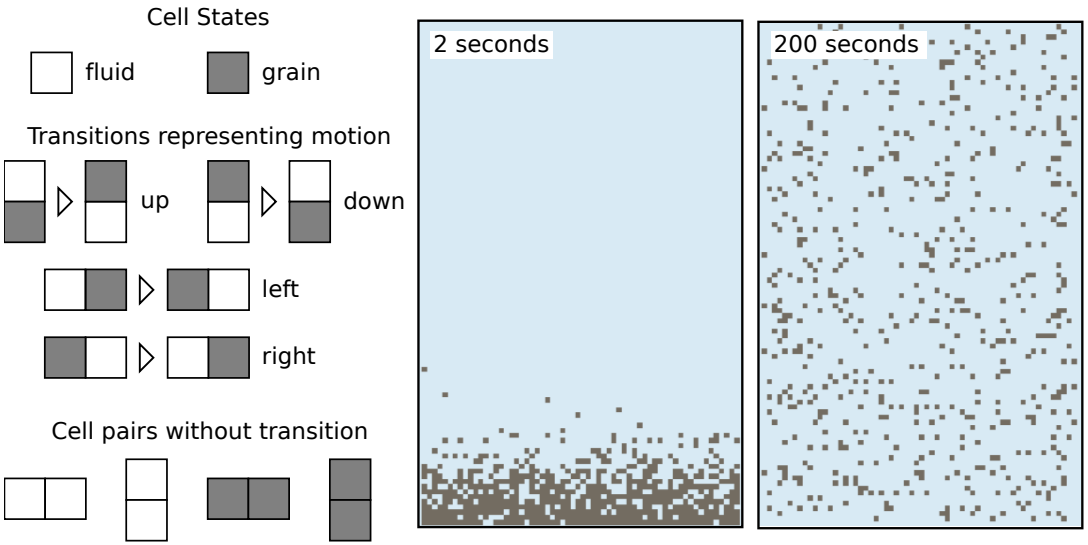
Component name	Process simulated/analysis performed	Key reference
LinearDiffuser	Linear diffusion of topography	Culling (1963)
PerronNLDiffuse	Nonlinear hillslope diffusion	Perron (2011)
Flexure	Simple lithospheric flexure under loading	Lambeck (1988), Hutton and Syvitski (2008)
gFlex	A more complex flexure model, utilizing gFlex	Wickert (2016)
FlowRouter	A convergent flow router, following the Fastscape algorithms	Braun and Willett (2013)
DepressionFinderAndRouter	A lake filler that can route flow across depressions	Tucker et al. (2001a)
SinkFiller	An algorithm to fill depressions in a surface	Tucker et al. (2001b)
OverlandFlow	A shallow overland flow approximation	de Almeida et al. (2012), Adams et al. (2016)
KinematicWaveRengers	A solution to the depth varying Manning equation for surface flow	Julien et al. (1995), Rengers et al. (2016)
SoilInfiltrationGreenAmpt	Infiltrate surface water into a soil following the Green–Ampt method	Julien et al. (1995), Rengers et al. (2016)
SoilMoisture	Compute local inter-storm water balance and root-zone soil moisture saturation fraction	Laio et al. (2001)
PotentialEvapotranspiration	Calculate potential evapotranspiration across a surface	ASCE-EWRI (2005), Zhou et al. (2013)
Radiation	Calculate total incident shortwave solar radiation	Bras (1990)
Vegetation	Calculate above-ground live and dead biomass, and leaf area index	Istanbulluoglu et al. (2012), Zhou et al. (2013)
VegCA	Cellular automata algorithm to simulate spatial organization of PFTs	Zhou et al. (2013)
PrecipitationDistribution	Generate a storm sequence of intervals and intensities	Eagleson (1978)
FireGenerator	Produces intervals between fire events, following a Weibull distribution	Polakow and Dunne (1999)
StreamPowerEroder	Implements fluvial erosion according to stream power, using the Fastscape algorithms	Braun and Willett (2013)
FastscapeEroder	An alternative implementation of the Fastscape stream power algorithms	Braun and Willett (2013)
DetachmentLtdErosion	An implementation of stream power erosion *not* based on Fastscape	Howard (1994)
SedDepEroder	Sediment-flux-dependent shear stress based fluvial incision	Hobley et al. (2011)
SteepnessFinder	Calculates steepness indices for a channel network	Wobus et al. (2006)
ChiFinder	Calculates the chi index along a channel network	Perron and Royden (2012)

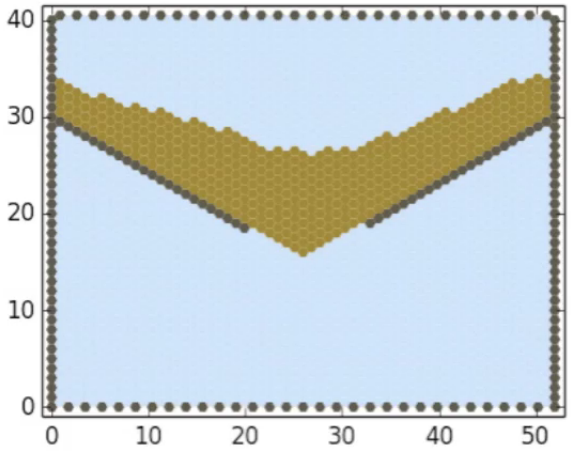
What Landlab provides

3. Input, output, visualization
   - Read model parameters from a formatted text file (optional)
   - **Read in** digital terrain data (e.g., **DEM**s) → grid
   - **Write gridded output** to files (netCDF format)
   - **Plot** data using Matplotlib graphics library

What Landlab provides

4. Support for cellular-automaton modeling
   - CellLab-CTS: Continuous-time stochastic CA model “engine”





(Tucker et al., 2016 Geoscientific Model Development)

How do you create a Landlab model (driver)?

- Your **unique** creation.
- ***Usually*** a model will:
  - Operate on a **grid**;
  - **Create data**;
  - Use components to **update data values through time**;
  - **Visualize data**.
- but it **does not have to do *all* of these things**.

ModelGrid Class

TERRAIN REPRESENTATION

Regular / Irregular GRID or TIN

Node properties from DEMs and Observations, or created based on input parameters or models

DRIVER

- Import data
- Instantiate grid
- Initialize components
- Initialize variables

Loop

- Operate components
- Couple components
- Components output variables
- Visualize data
- Export data

DATA STRUCTURES

Stored within the grid at cells, nodes or links

Stored as arrays or fields

COMPONENTS

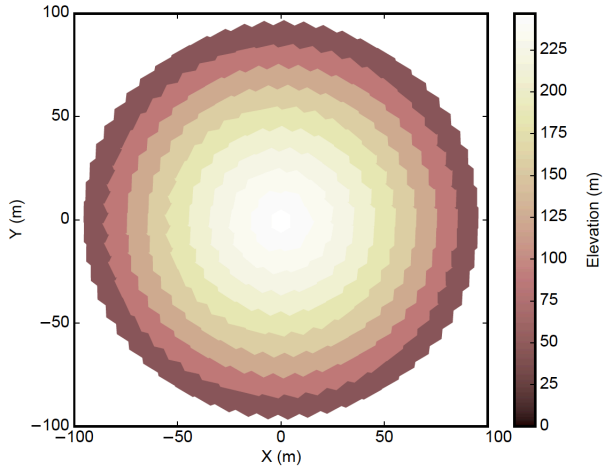
For example...

- Weather generator
- Runoff & discharge
- River incision
- Hillslope diffusion
- Rock fracture
- Soil moisture
- Vegetation Dynamics
- Cratering
- 2D Flexure

Single component diffusion model

Code to implement a simple diffusion model on a radial Landlab grid, using Landlab components:

```
>>> from landlab import RadialModelGrid, imshow_grid
>>> from landlab.components import LinearDiffuser
>>> from matplotlib.pyplot import show
>>> mg = RadialModelGrid(num_shells=10, dr=10.)
>>> z = mg.add_zeros('node', 'topographic__elevation')
>>> dt = 2000.  # no longer the stable timestep
>>> ld = LinearDiffuser(mg, linear_diffusivity=1.e-2)
>>> for i in range(500):
...     z[mg.core_nodes] += 0.001*dt
...     ld.run_one_step(dt)
>>> imshow_grid(mg, z, grid_units=('m', 'm'), var_name='Elevation (m)')
>>> show()
```



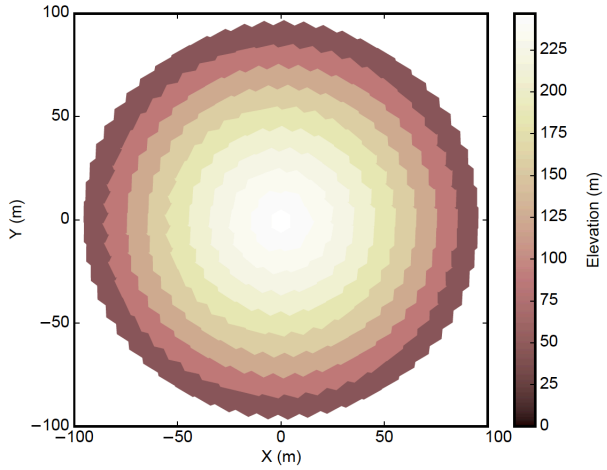
(Hobley et al., 2017, ESURF)

Single component diffusion model

Code to implement a simple diffusion model on a radial Landlab grid, using Landlab components:

Import Statements

```
1. >>> from landlab import RadialModelGrid, imshow_grid
2. >>> from landlab.components import LinearDiffuser
3. >>> from matplotlib.pyplot import show
4. >>> mg = RadialModelGrid(num_shells=10, dr=10.)
5. >>> z = mg.add_zeros('node', 'topographic__elevation')
6. >>> dt = 2000.  # no longer the stable timestep
7. >>> ld = LinearDiffuser(mg, linear_diffusivity=1.e-2)
8. >>> for i in range(500):
9. ...     z[mg.core_nodes] += 0.001*dt
10. ...     ld.run_one_step(dt)
11. >>> imshow_grid(mg, z, grid_units=('m', 'm'), var_name='Elevation (m)')
12. >>> show()
```



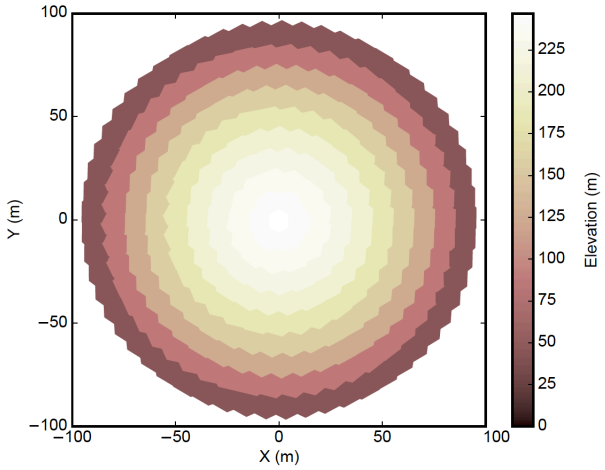
(Hobley et al., 2017, ESURF)

Single component diffusion model

Code to implement a simple diffusion model on a radial Landlab grid, using Landlab components:

Instantiate model grid; Add data/create field

```
>>> from landlab import RadialModelGrid, imshow_grid
>>> from landlab.components import LinearDiffuser
>>> mg = RadialModelGrid(num_shells=10, dr=10.)
>>> z = mg.add_zeros('node', 'topographic__elevation')
>>> dt = 2000.  # no longer the stable timestep
>>> ld = LinearDiffuser(mg, linear_diffusivity=1.e-2)
>>> for i in range(500):
...     z[mg.core_nodes] += 0.001*dt
...     ld.run_one_step(dt)
>>> imshow_grid(mg, z, grid_units=('m', 'm'), var_name='Elevation (m)')
>>> show()
```



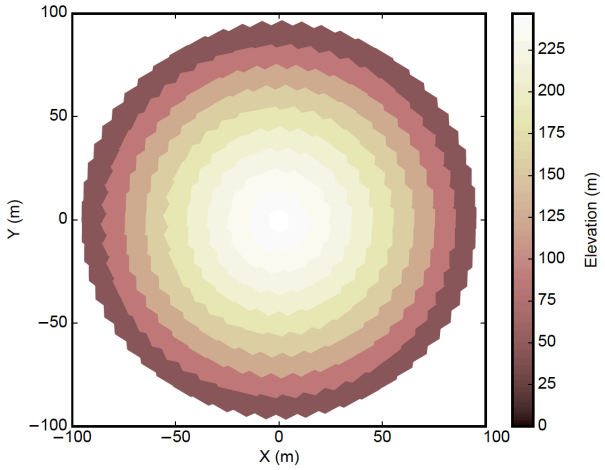
(Hobley et al., 2017, ESURF)

Single component diffusion model

Code to implement a simple diffusion model on a radial Landlab grid, using Landlab components:

Initialize parameter; Instantiate diffusion object

```
>>> from landlab import RadialModelGrid, imshow_grid
>>> from landlab.components import LinearDiffuser
>>> from matplotlib.pyplot import show
>>> mg = RadialModelGrid(num_shells=10, dr=10.)
>>> dt = 2000.  # no longer the stable timestep
>>> ld = LinearDiffuser(mg, linear_diffusivity=1.e-2)
>>> for i in range(500):
...     z[mg.core_nodes] += 0.001*dt
...     ld.run_one_step(dt)
>>> imshow_grid(mg, z, grid_units=('m', 'm'), var_name='Elevation (m)')
>>> show()
```



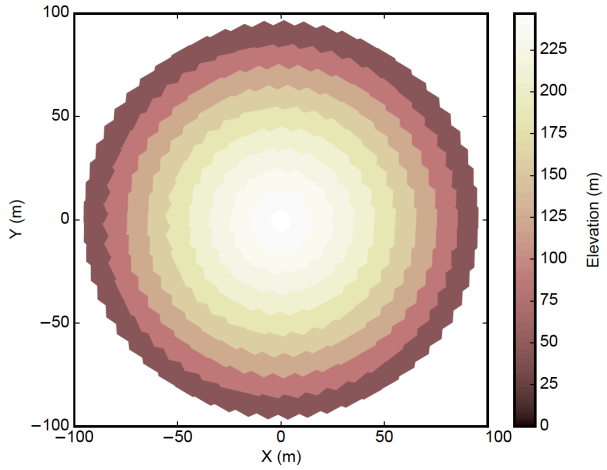
(Hobley et al., 2017, ESURF)

Single component diffusion model

Code to implement a simple diffusion model on a radial Landlab grid, using Landlab components:

```
1. >>> from landlab import RadialModelGrid, imshow_grid
2. >>> from landlab.components import LinearDiffuser
3. >>> from matplotlib.pyplot import show
4. >>> mg = RadialModelGrid(num_shells=10, dr=10.)
5. >>> z = mg.add_zeros('node', 'topographic__elevation')
6. >>> dt = 2000.  # no longer the stable timestep
8. >>> for i in range(500):
9. ...     z[mg.core_nodes] += 0.001*dt
10. ...     ld.run_one_step(dt)
12. >>> show()
```

Time loop; Do the calculations!



(Hobley et al., 2017, ESURF)

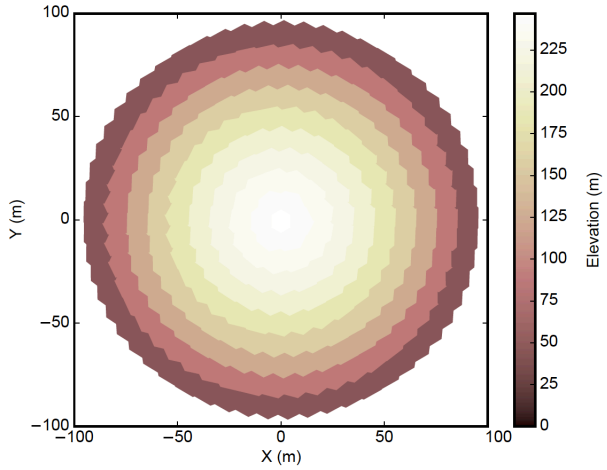
Single component diffusion model

Code to implement a simple diffusion model on a radial Landlab grid, using Landlab components:

```
>>> from landlab import RadialModelGrid, imshow_grid
>>> from landlab.components import LinearDiffuser
>>> from matplotlib.pyplot import show
>>> mg = RadialModelGrid(num_shells=10, dr=10.)
>>> z = mg.add_zeros('node', 'topographic__elevation')
>>> dt = 2000.  # no longer the stable timestep
>>> ld = LinearDiffuser(mg, linear_diffusivity=1.e-2)
>>> for i in range(500):
...     z[mg.core_nodes] += 0.001*dt
```

Visualize data

```
>>> imshow_grid(mg, z, grid_units=('m', 'm'), var_name='Elevation (m)')
>>> show()
```



(Hobley et al., 2017, ESURF)

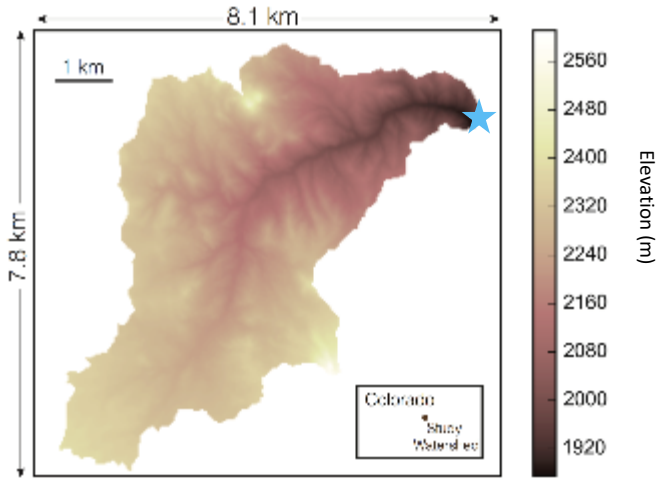
Every model is some variant of the previous example

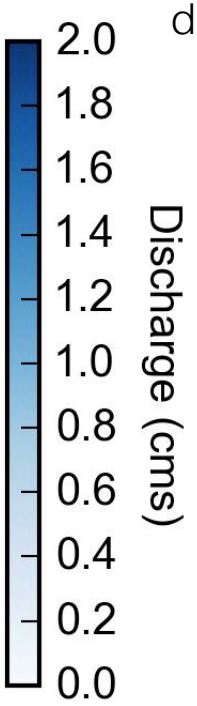
- Can be on the **command line** (as last example).
- Can be a **Python (.py) file** that is run from the command line.
- Can be an **iPython notebook**, which is great for teaching and learning.
- Can run **on your computer** and/or on **Hydroshare**.
  - If you run *on your own computer*, you need to *install Landlab* locally. (http://landlab.github.io)
  - If you run *on Hydroshare*, you need join Hydroshare, but you *do not need Landlab installed locally*. (https://www.hydroshare.org/)

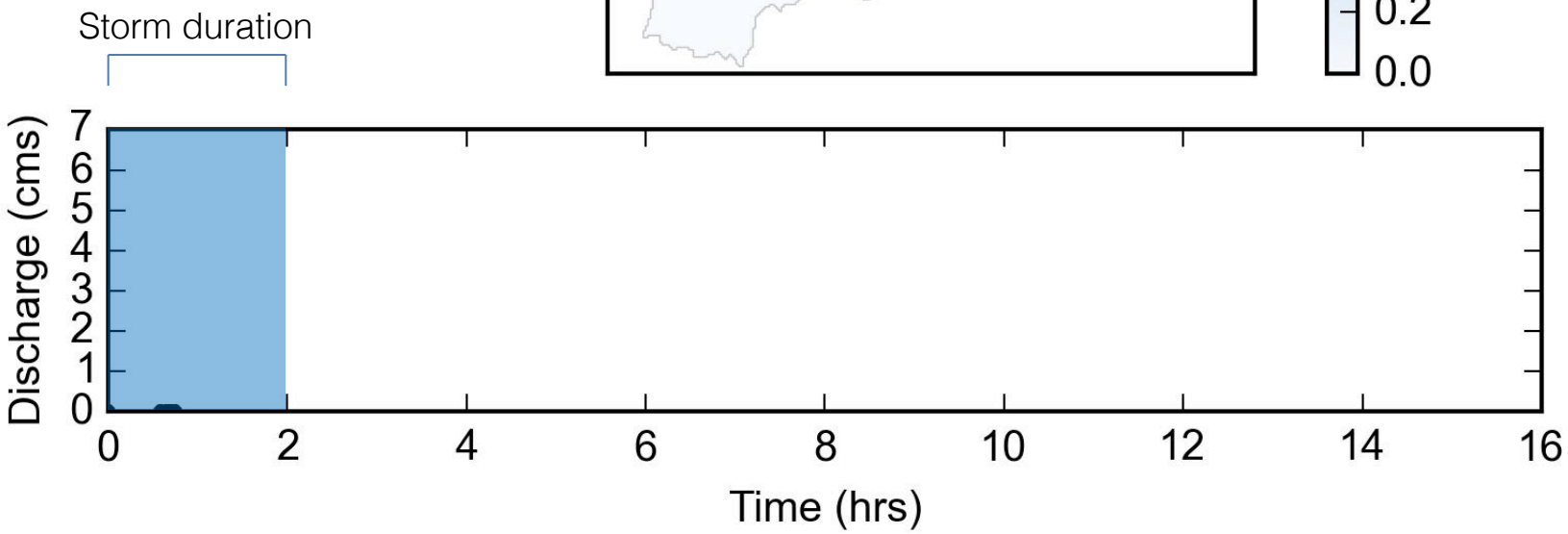
More examples of build Landlab models

Application in a real world setting: Spring Creek, CO.

*Note scale differences



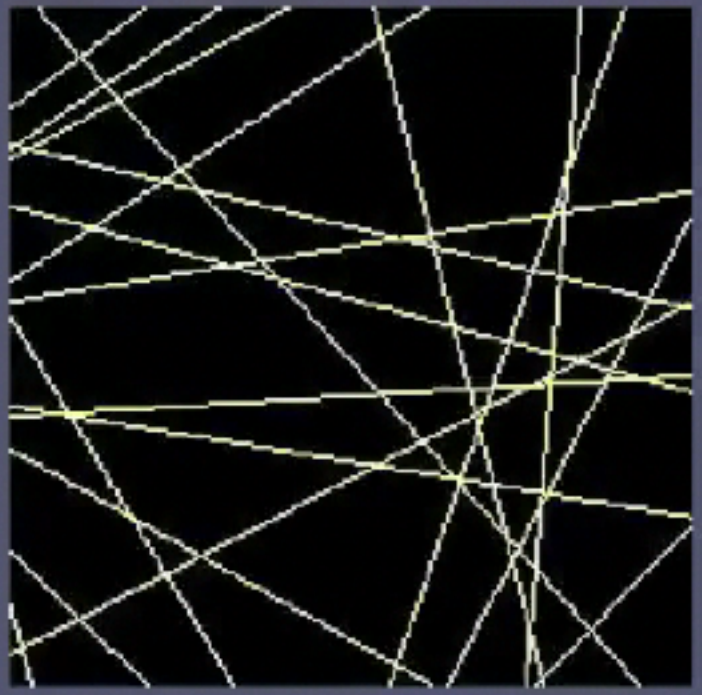




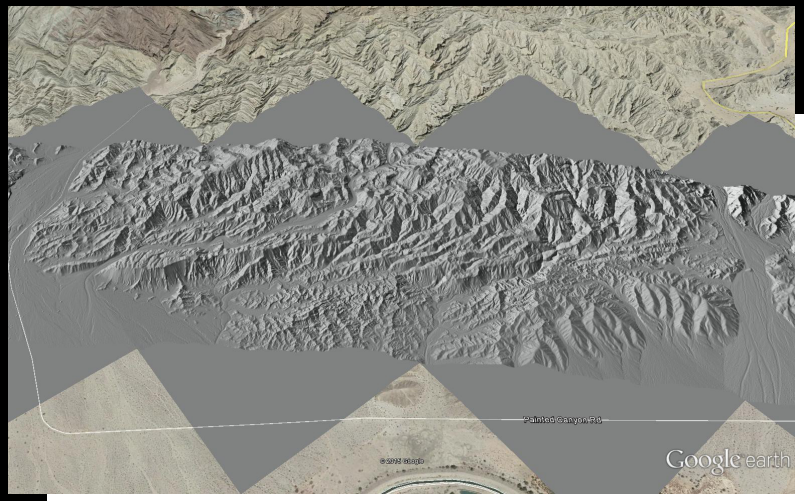
(source: Jordan Adams, Tulane University)

Cellular automaton model of weathering along fractures

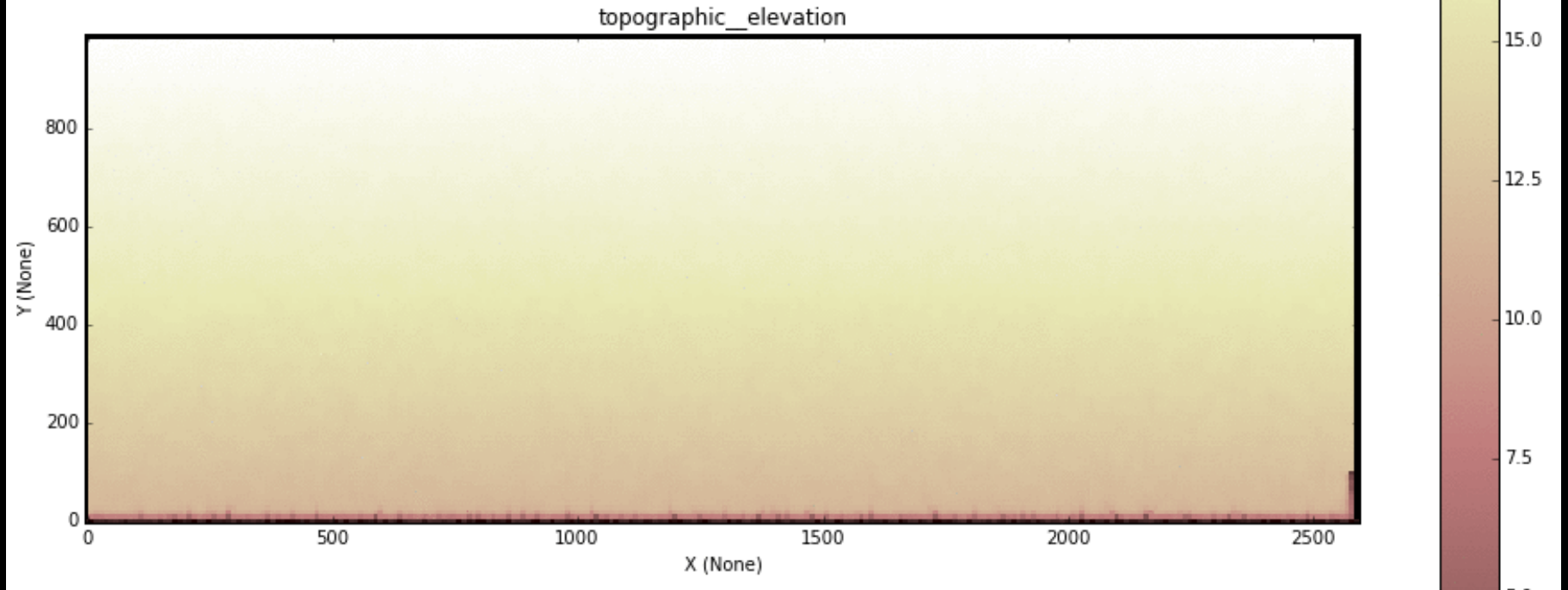
(Source: Greg Tucker, CU-Boulder)



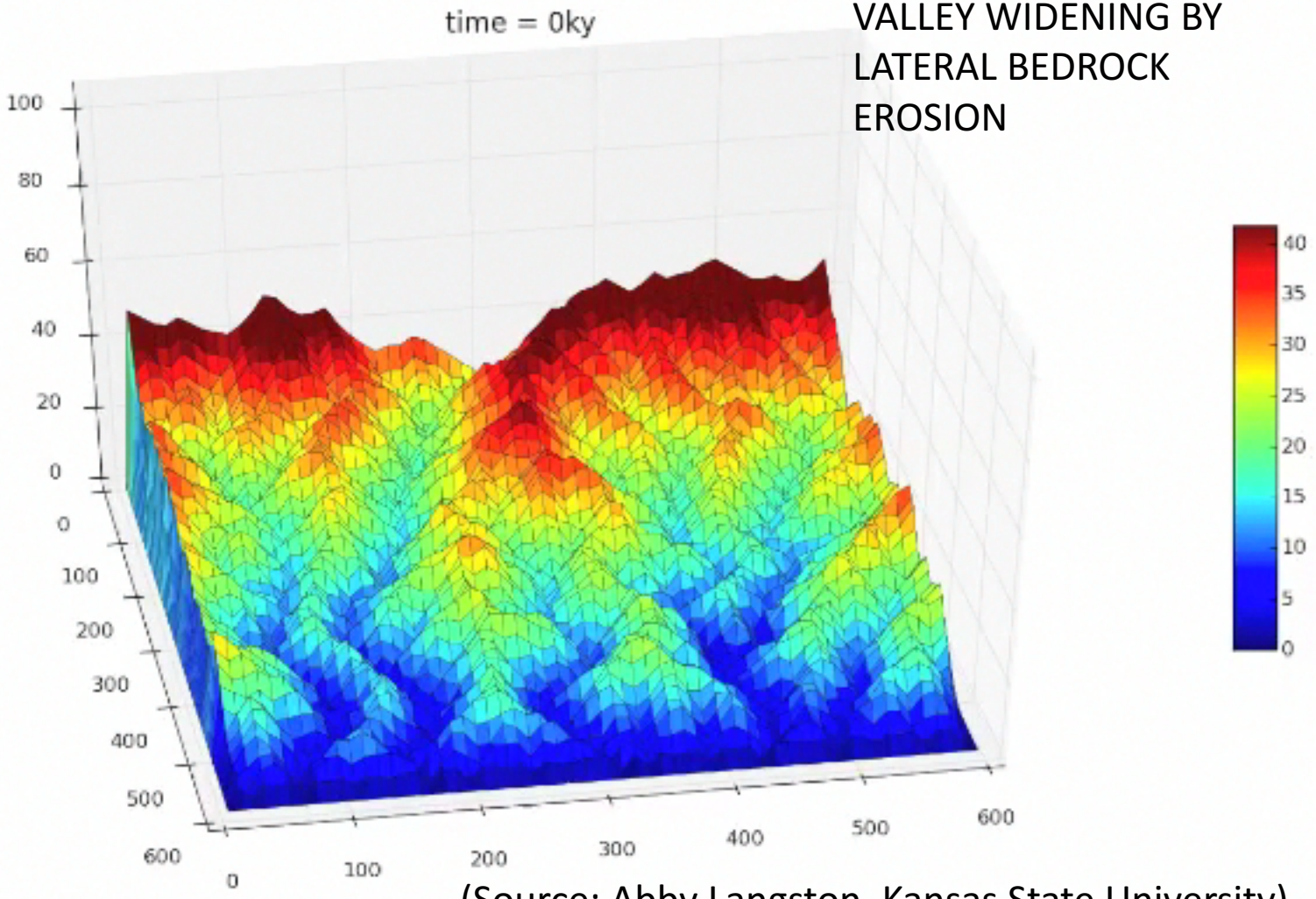
Why do strike-slip faults sometimes show distributed shear, and sometimes not?



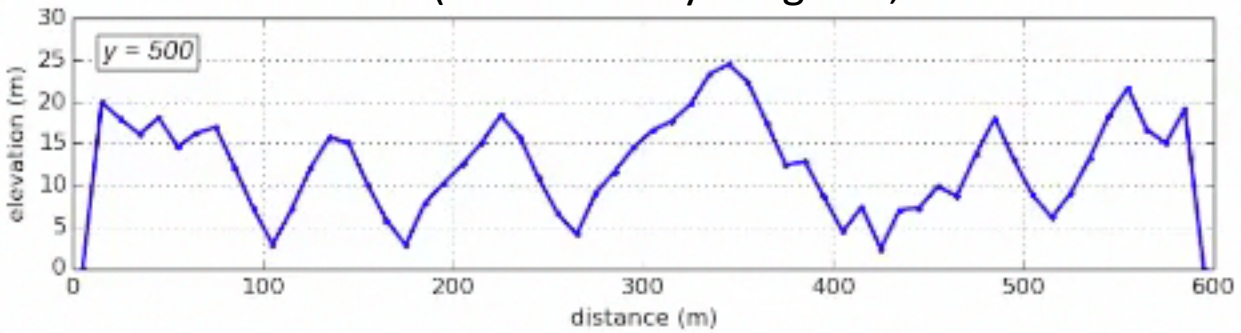
(Source: Harrison Gray, CU-Boulder)



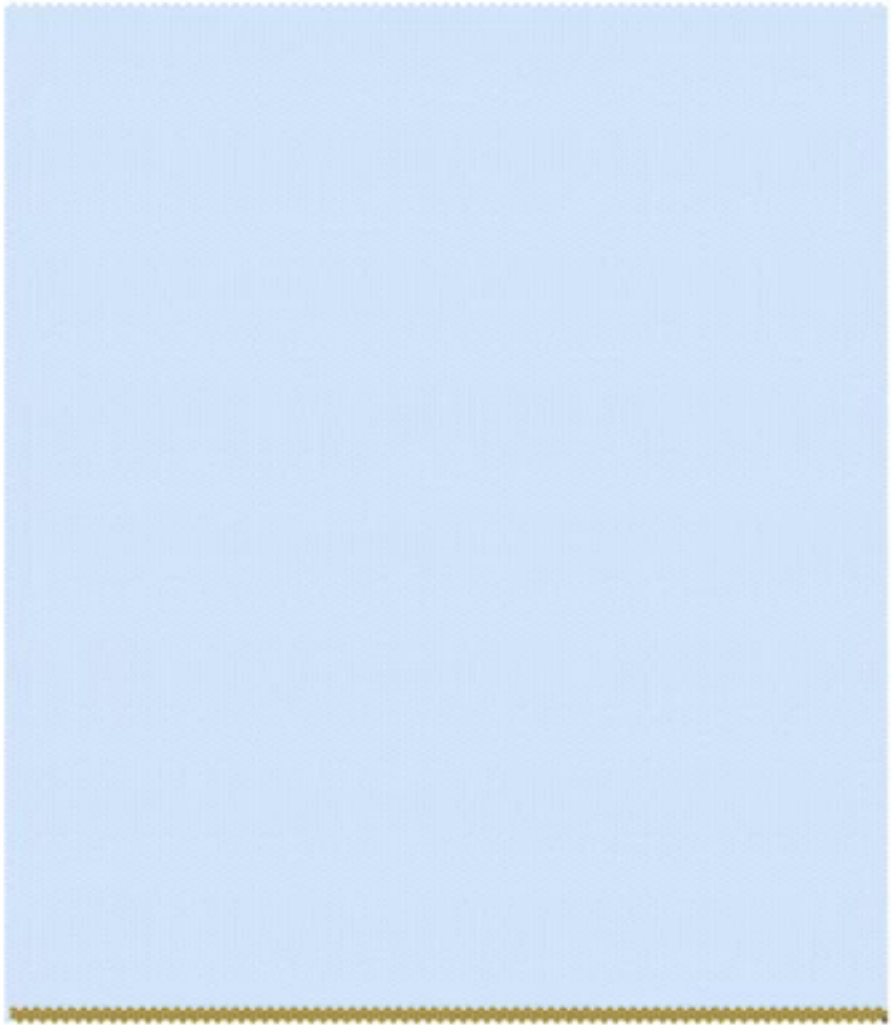
VALLEY WIDENING BY LATERAL BEDROCK EROSION



(Source: Abby Langston, Kansas State University)



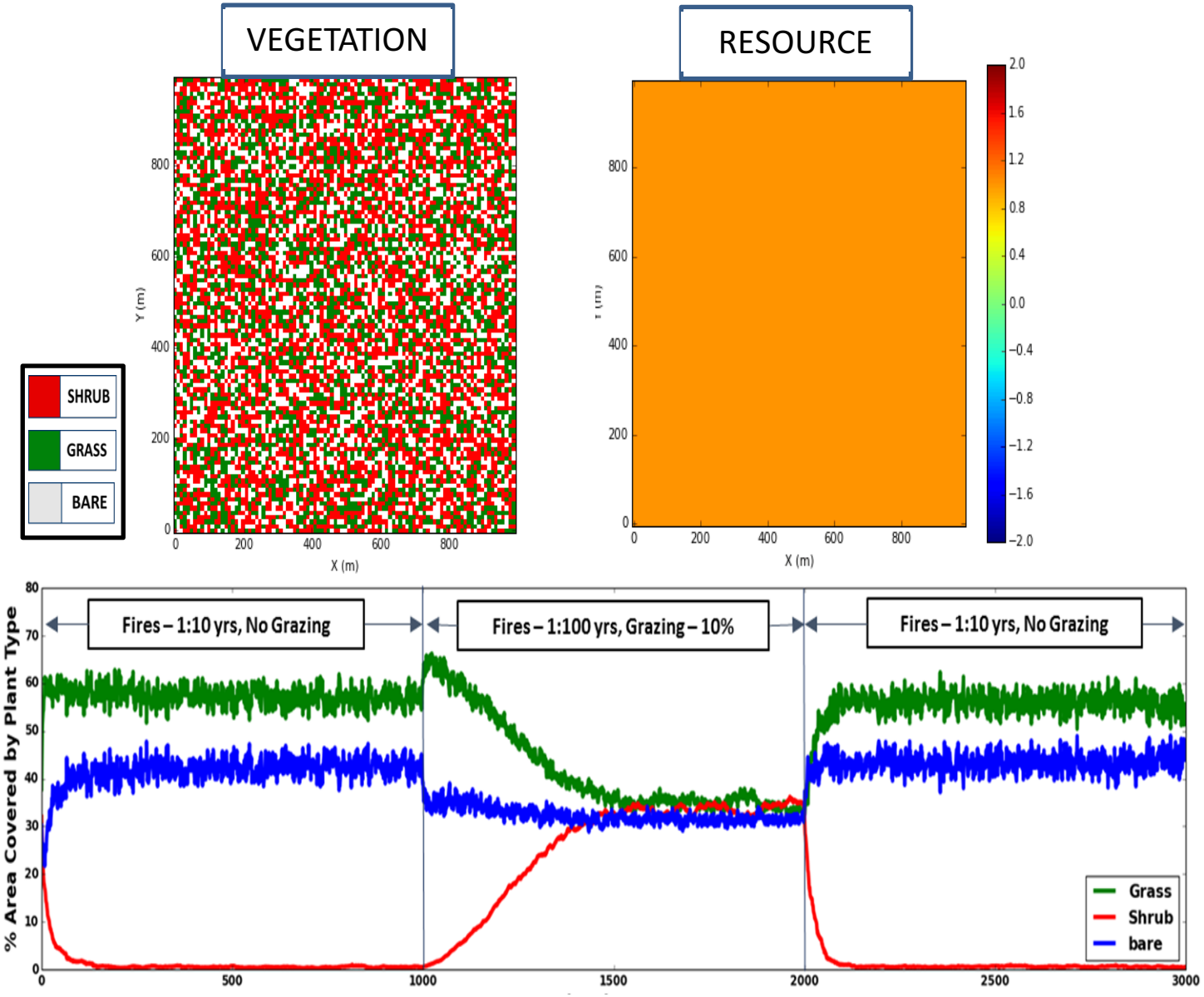
Weathering & disturbance similar to slip rate



$W' = D' = 1$

(Source: Greg Tucker, CU-Boulder)

Climate Change Experiments #1

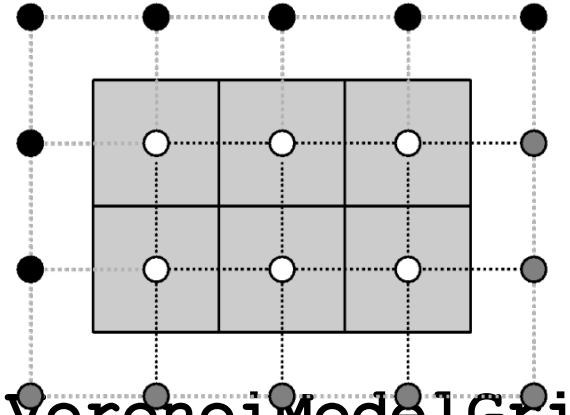


(Source: Sai Nudurupati, U. Washington)

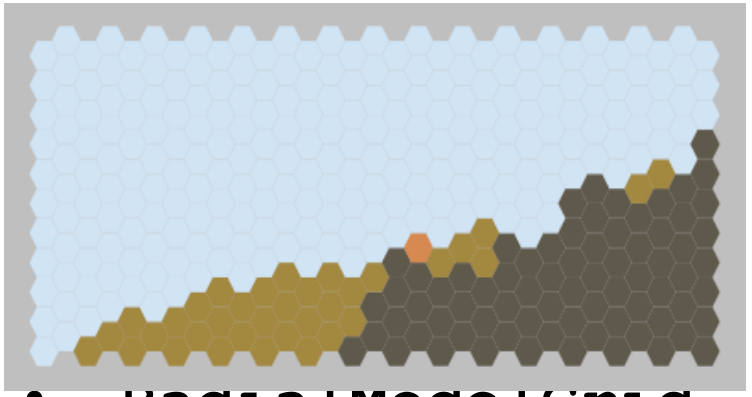
A few more basics before we start modeling

Spatial representation of computational domains

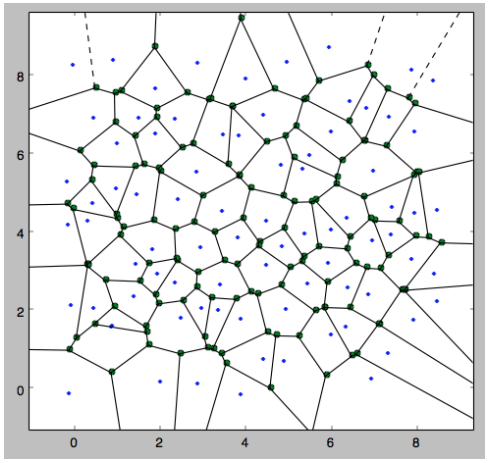
- `RasterModelGrid`



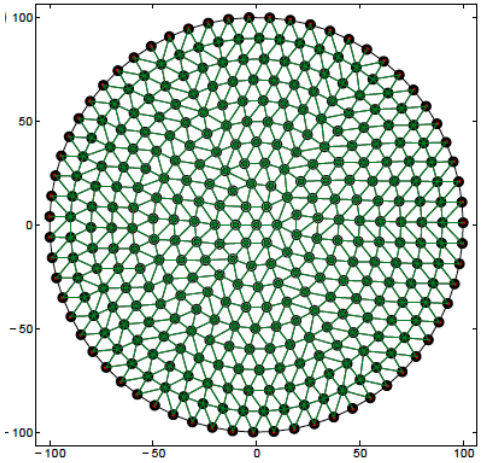
- `HexModelGrid`



- `VoronoiModelGrid`



- `RadialModelGrid`

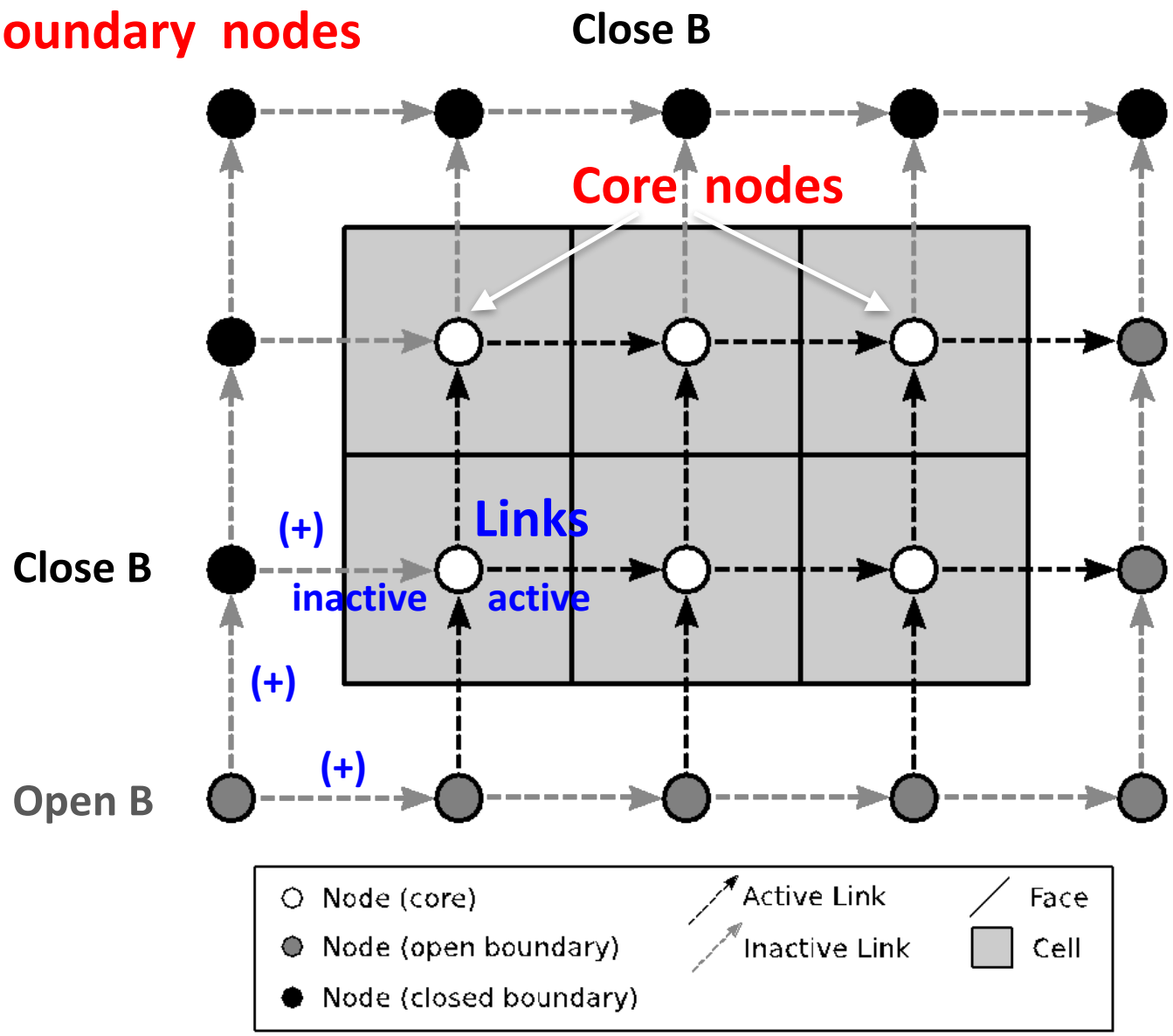


Model Grid Documentation:

http://landlab.readthedocs.io/en/latest/#developer-documentation

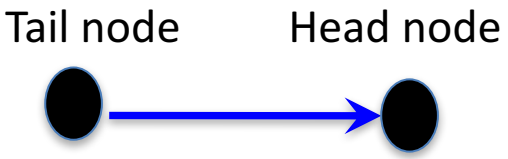
RasterModelGrid

Boundary nodes



Link **= directed line segment connecting two adjacent nodes**

Link direction is toward upper right half-space by default



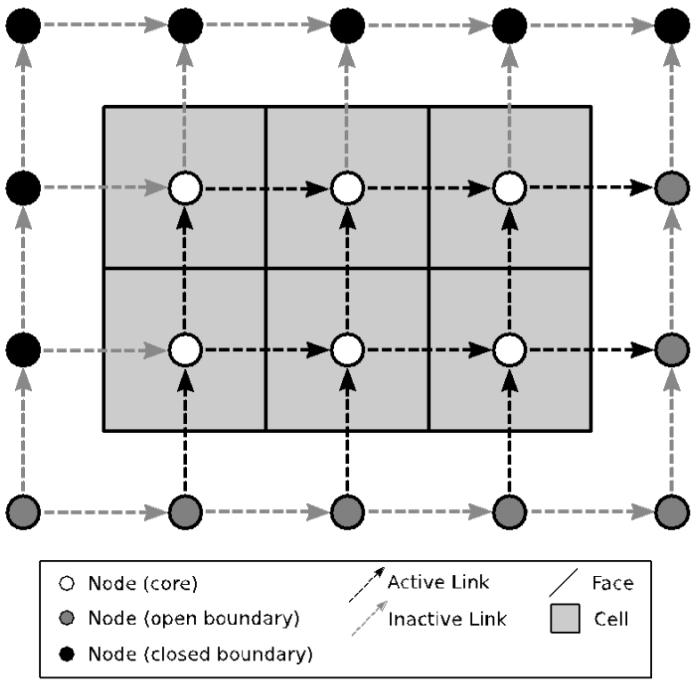
CORE_NODE: 0; FIXED_VALUE_BOUNDARY: 1 (e.g. outlet)
TRACKS_CELL_BOUNDARY: 3 (looped boundary); CLOSED_BOUNDARY: 4

Fields: Attaching data to the grid

- A **field** is a NumPy array containing data that are associated with a particular type of grid element (typically nodes or links)
- Fields are **1D arrays**
- Values correspond to the element with the same ID. Example: value 5 of a node field belongs to node #5.
- Fields are "**attached**" to the grid (the grid object includes dictionaries listing all the fields)
- Fields have names (as strings)
- Create fields with grid functions `add_zeros`, `add_ones`, or `add_empty`

RasterModelGrid example: Create a 4 by 5 grid

```
>> import landlab
>> from landlab import RasterModelGrid
>> rg = RasterModelGrid((4,5),10.0)
```




```
>> rg.number_of_nodes
Out: 20
>> rg.number_of_links
Out: 31
>> rg.number_of_node_rows
Out: 4
>> rg.number_of_node_columns
Out: 5
>> rg.number_of_core_nodes
Out: 6
>> rg.number_of_cells
Out: 6
```

Example: attaching data to the grid

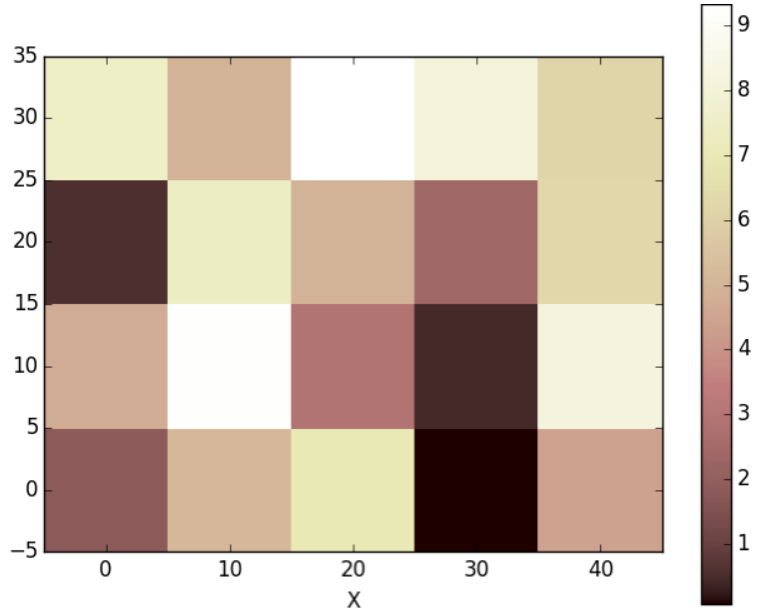
```
>> import landlab
>> from landlab import RasterModelGrid
>> rg = RasterModelGrid((4,5),10.0)
```

Create a random field and “attach” it to the grid as ‘elevation’

```
>> import numpy as np
>> z=10*np.random.rand(rg.number_of_nodes)
>> rg.add_field('node','elevation', z)
```

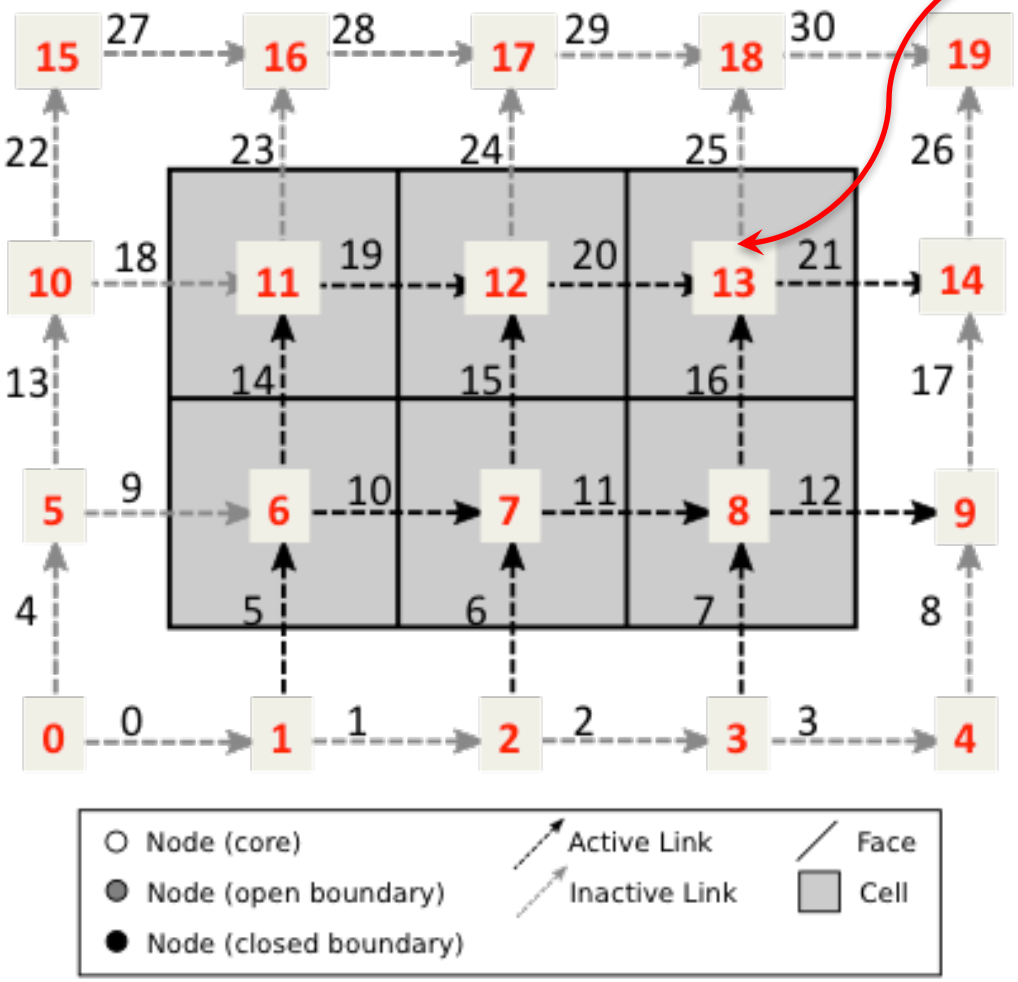
Plot the elevation field

```
>> from landlab.plot import imshow_grid
>> import matplotlib.pyplot as plt
>> imshow_grid(rg, 'elevation')
>> plt.show()
```



Explore node and link structure and “attached” data

Starting from the Southwest corner nodes are numbered to the East in each row.
Links are numbered using the number of their tail nodes.




```
>> rg.status_at_node[13]
Out: 0
>> rg.links_at_node[13]
Out: array([21, 25, 20, 16])
>> rg.link_dirs_at_node[13]
Out: array([-1, -1,  1,  1])
>> rg.node_at_link_tail[20]
Out: 12
>> rg.node_at_link_head[20]
Out: 13
```

```
>> rg.at_node['elevation'][13]
Out: 2.36
```

Now set ‘elevation’ to 100

```
>> rg.at_node['elevation’][13]=100
```

OR: z[13]=100

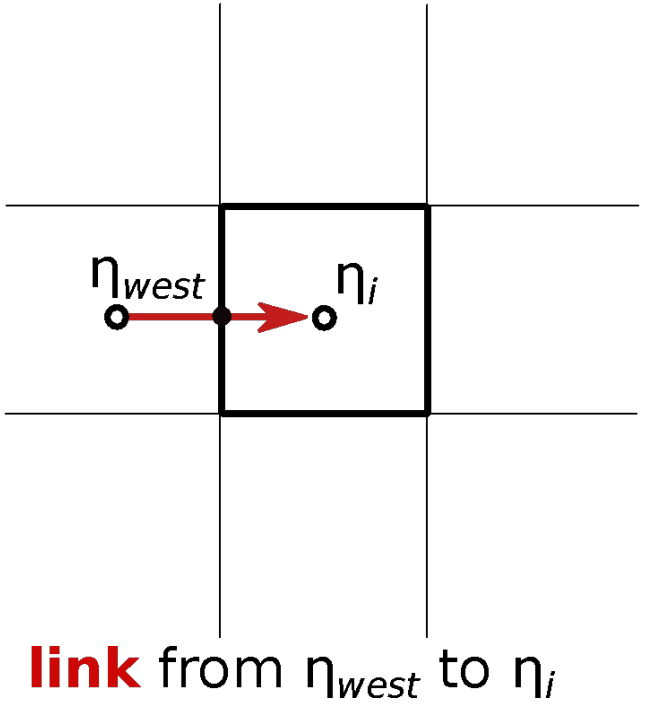
```
>> rg.at_node['elevation'][13]
Out: 100
```

CORE_NODE: 0; **FIXED_VALUE_BOUNDARY: 1 (e.g. outlet)**
TRACKS_CELL_BOUNDARY: 3 (looped boundary); CLOSED_BOUNDARY: 4

Examples of single-line functions, here a method in RasterModelGrid

```
deta_dx = rg.calc_grad_at_link(eta)
```

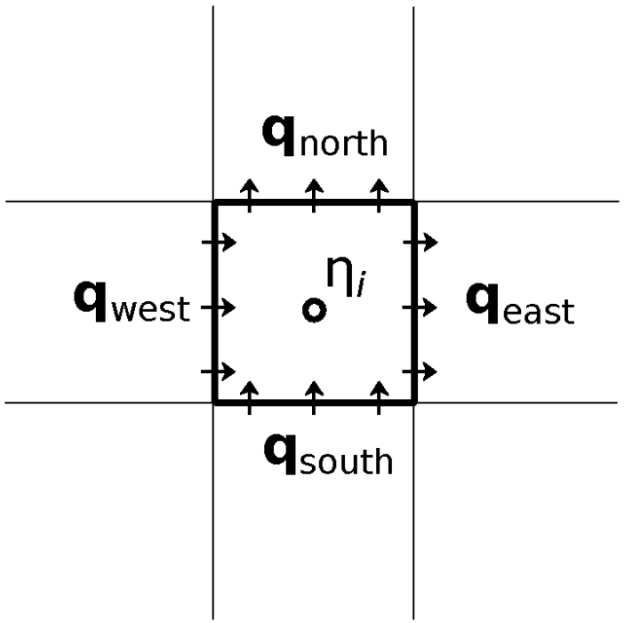
- *eta* is a scalar defined at nodes
- One value of *deta_dx* for every link
- Positive when *eta* increases in the link direction
- Negative when *eta* decreases in the link direction



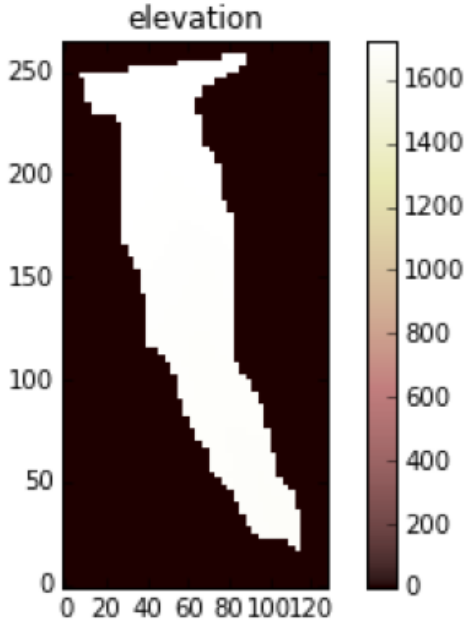
Calculating the divergence of a gradient field

```
>>> q = -D * deta_dx
>>> dqdx = rg.calc_flux_div_at_node(q)
```

- *q* is a vector defined at links
- One value of *dqdx* for every node
- Positive when net flux is outwards



Reading raster digital terrain data



Landlab's read_esri_ascii function:

- Reads data from ESRI ASCII raster file
- Creates a RasterModelGrid and a data field
- Also: read/write netCDF files
- Example:

```
>> from landlab.io import read_esri_ascii
>> (mg, z) = read_esri_ascii('file_name.asc',
                            name='elevation')
```

ModelGrid Class

TERRAIN REPRESENTATION

Regular / Irregular GRID or TIN

Node properties from DEMs and Observations, or created based on input parameters or models

DRIVER

- Import data
- Instantiate grid
- Initialize components
- Initialize variables

Loop

- Operate components
- Couple components
- Components output variables
- Visualize data
- Export data

DATA STRUCTURES

Stored within the grid at cells, nodes or links

Stored as arrays or fields

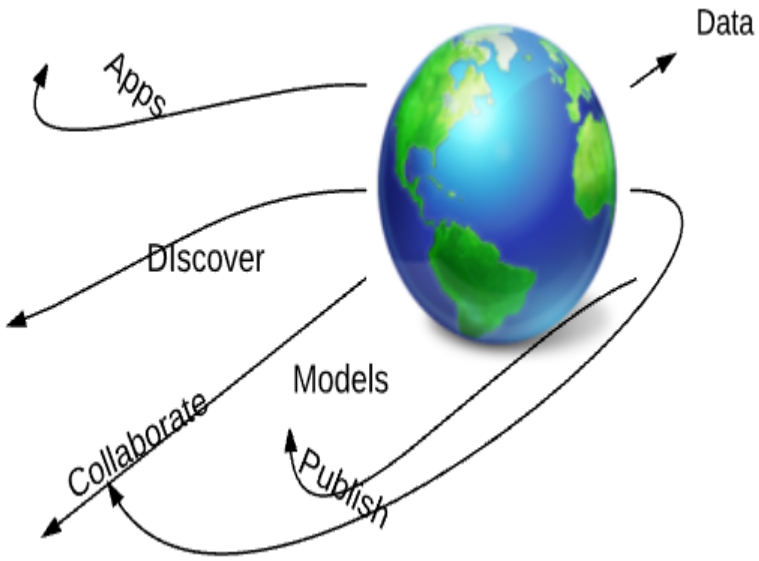
COMPONENTS

For example...

- Weather generator
- Runoff & discharge
- River incision
- Hillslope diffusion
- Rock fracture
- Soil moisture
- Vegetation Dynamics
- Cratering
- 2D Flexure

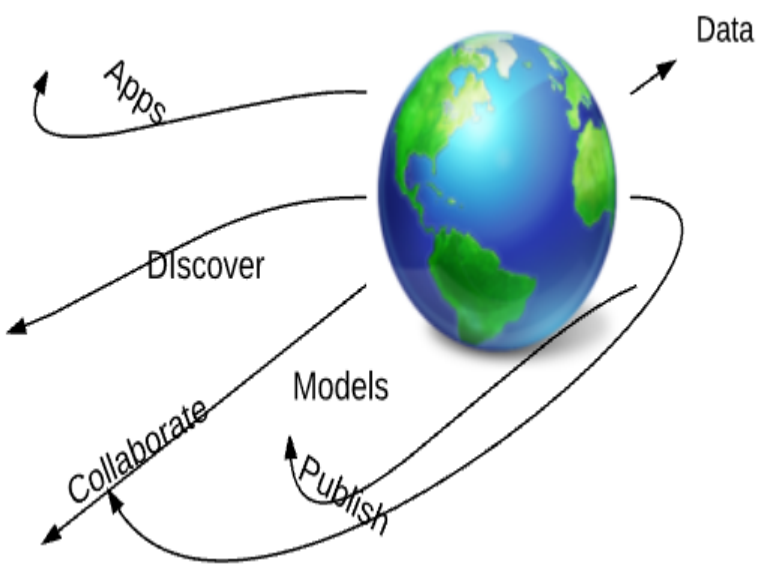
Using Cyber-Infrastructure to do science

Big problems in your head



Using Cyber-Infrastructure to do science

Big water problems in your head



Big water problems in HydroShare

The Landlab HydroShare User Story

Landlab is a Python based exploratory environment for anyone to build any model – so we chose to **Open With JupyterHub** link to USU for development and NSCA for running. Thank you **Tony Castronova** (USU) & **Dandong Yin** (U of I)!

Thank you **Alva Couch**! Cyberhugs to our favorite software architect!

UW CEE Watershed Dynamics group starts the Landlab **Group** in HydroShare.
Landlab project collaborators from Tulane, UC Boulder, and CSDMS become users, join group, help test development server.

Christina, Ronda, Sai, Erkan
Join HydroShare as Users
Create **Resources**
Erkan tackles the challenge of remembering his password.

Users publish tutorials for Landlab model components (Overland Flow, ET, DEM processing, landlides… also available on Github. Shout out to Data Driven Education!!

Landlab tutorials and models are published on HydroShare, linked to **Help HydroShare** and **Collections** for CUAHSI meeting

Now let's get modeling!

- Log in to https://www.hydroshare.org/
- Hit COLLABORATE tab at top.
- Enter the Landlab group (if you haven't already joined, join now!)
- Hit Resources tab.
- Enter the "CSDMS Landlab Models Clinic 2017"
- Get modeling…
- If you have Landlab installed on your laptop and want to work locally, go to https://github.com/landlab/csdms_model_clinic_may_2017 and download. Navigate to "notebooks" in the download:
  - >>> jupyter notebook

Running on Hydroshare…

- Suggested order of models to run is:
  1. Simple Landscape Evolution Model
  2. Overland Flow and Erosion Example
  3. Landlab Ecohydrologic Mapping Tutorial
  4. Spatio-temporal Ecohydrologic Mapping Tutorial
- BUT, there are no rules. Follow your interests, except that 3 before 4 will be easier for understanding.

- Two step process in Hydroshare, open an initial jupyter notebook, then the actual model.
- Shift-enter will run a block of code.
- Code blocks can be edited and re-run.

How to download and run more tutorials locally

- Go to: https://github.com/landlab/landlab/wiki/Tutorials
- Click:

  Click here to download all the tutorials

- Save ZIP
- Double-click to unpack
- In terminal or command window, **navigate to new folder**
- >>> `jupyter notebook`
- Shift-Enter to move through each cell