Looking Beyond the Local Neighbourhood: dynamic kernel topographic indices (KTPI)

Presentation for **FOSS4G 2016**
302A - Wednesday, May 4, 2016 - 17:00 to 17:15 |tesera.com

Introductions

Presenting: Michael Kieser

Key collaborators (stuck at home working):

Matt Zandstra, Boris Cosic, Yves Richard

Agenda

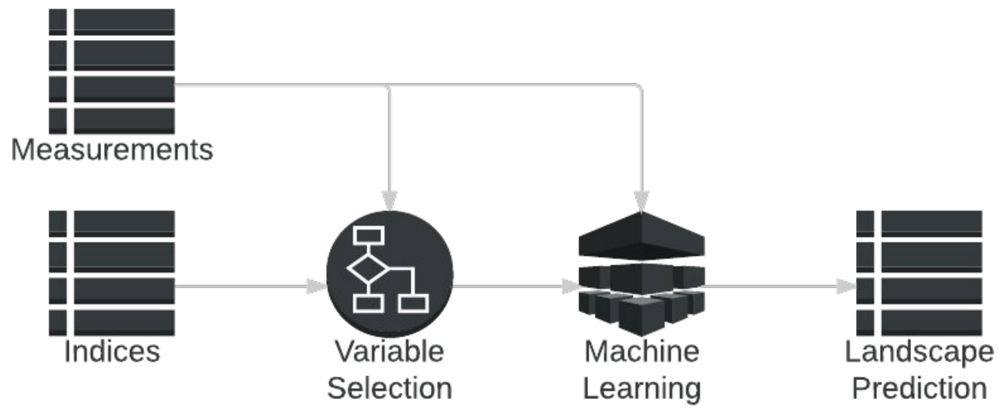
Why KTPI for prediction

What Capabilities

How Cloud based task distribution

TESERA

Why KTPI



- **Indices** are used as inputs for landscape prediction / imputation processes for Forestry and Insurance clients.

- **Multi-scale** kernels improve the predictive power for each measurement (target).

- Process at multiple-scales and let the models decide which scale is best for each measurement (target)

- Processing is **CPU Intensive**

What

https://github.com/tesera/ktpi

Kernel Topographic Indices (KTPI) is a R based utility to classify topography using raster focal and zonal analysis based to derive elevation statistics, terrain indices, topographic position indices, and kernel aspect indices from DEM (digital elevation model) data.

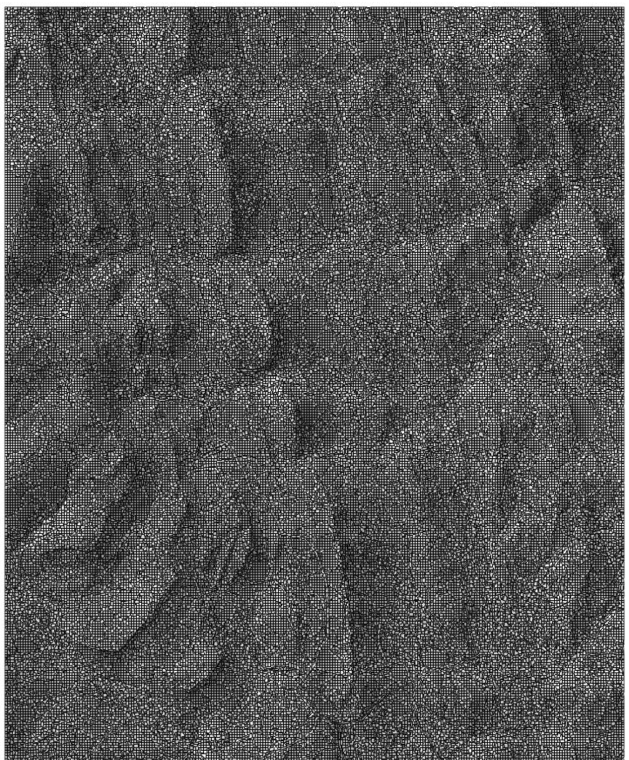
Hijmans, R. + Weiss, A. + Stage, A.

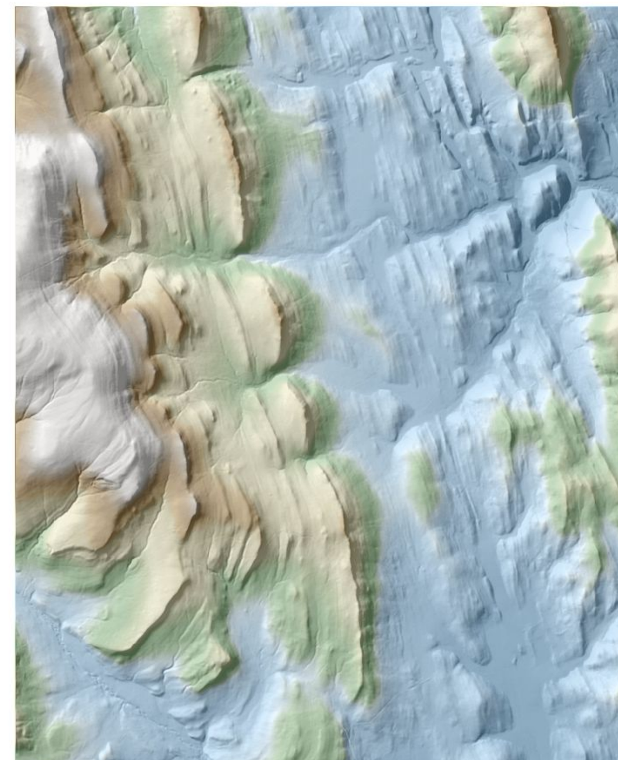
What

INPUT

feature data (unique id): rasterized point, line, polygon

digital elevation model

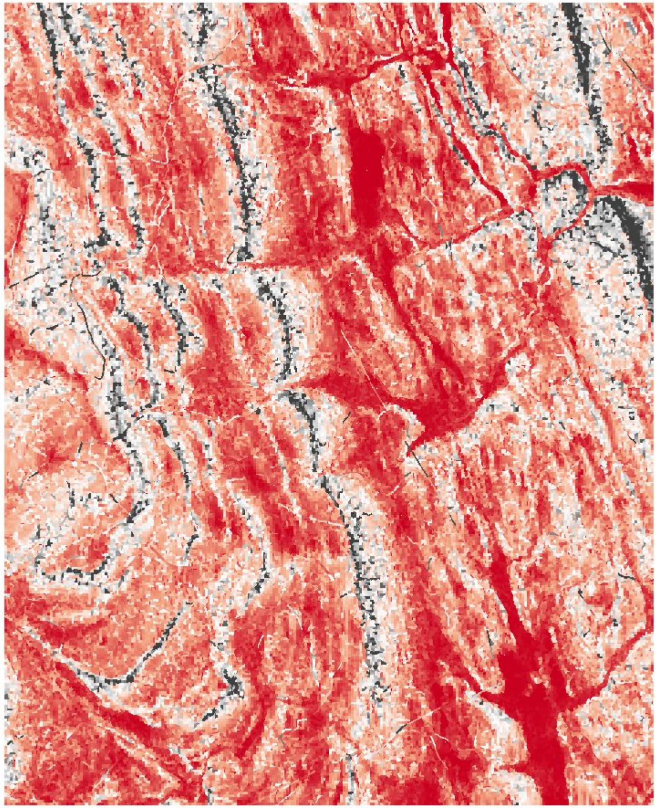


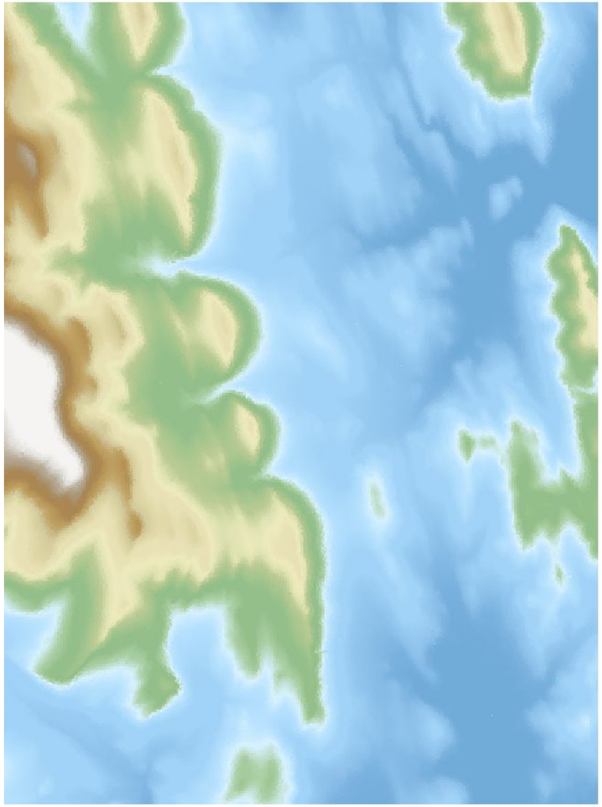


What

ktpi.R STATISTIC

zonal analysis of elevation value statistics for each feature

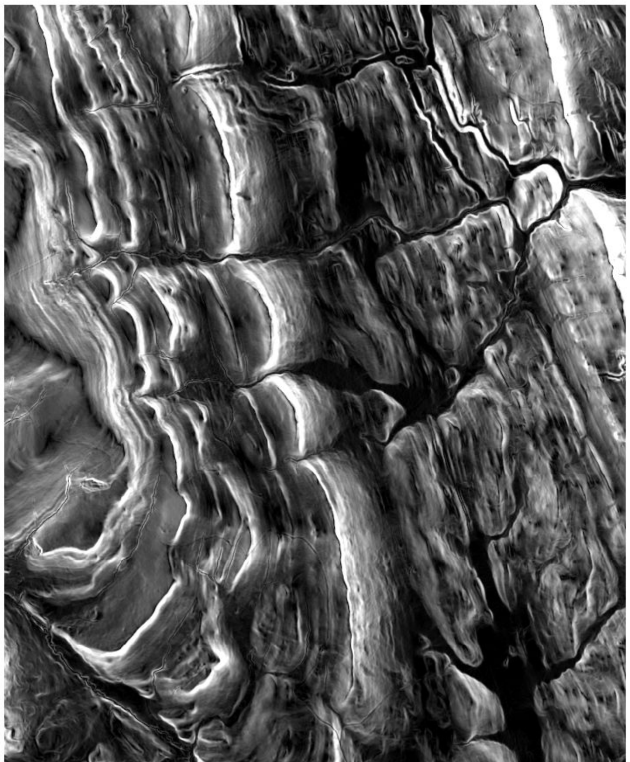


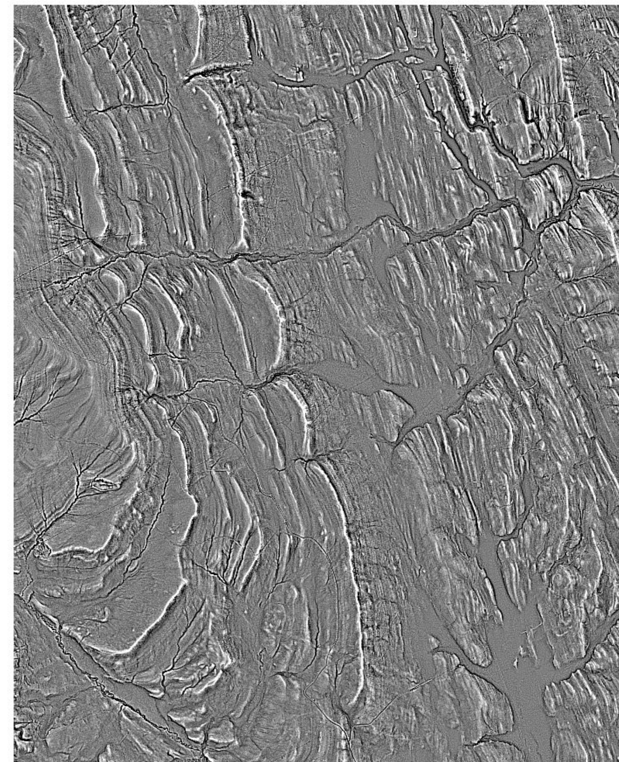


What

ktpi.R TERRAIN

zonal analysis of “terrain” function (Robert J. Hijmans **R raster** library) outputs for each feature

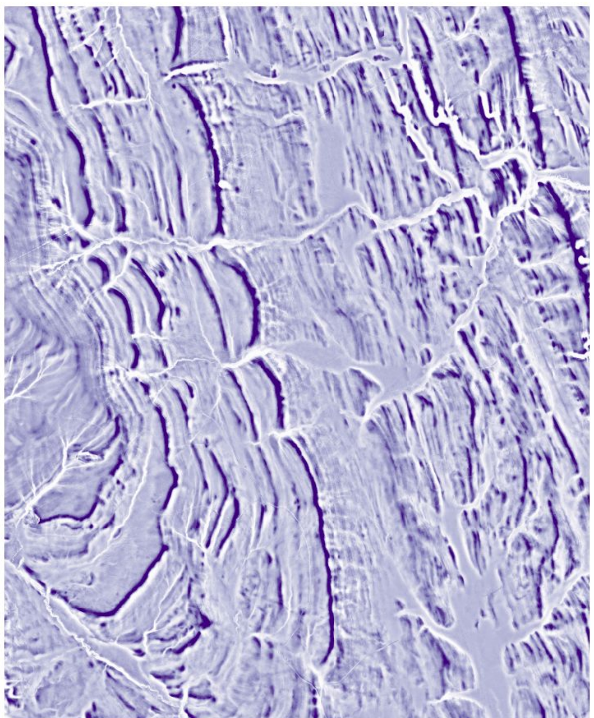


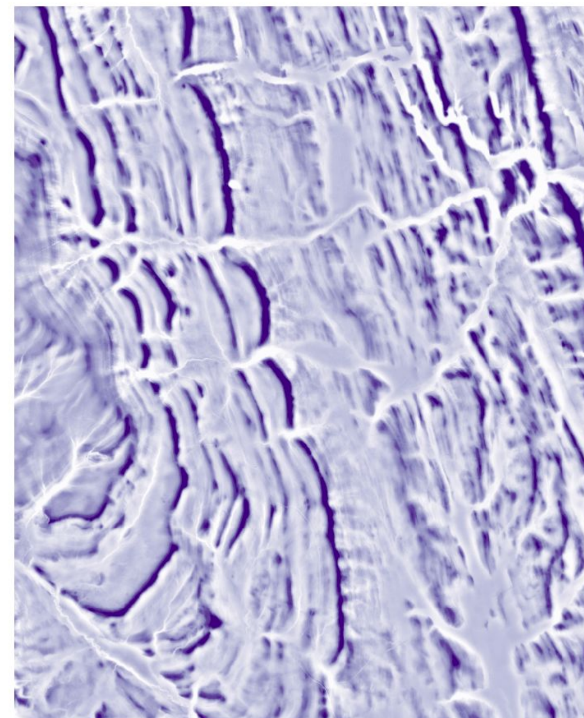


What

ktpi.R KTPI

zonal analysis of, a focal analysis of difference from mean and standard deviation within a given kernel (Andrew D. Weiss, 2001), for each feature

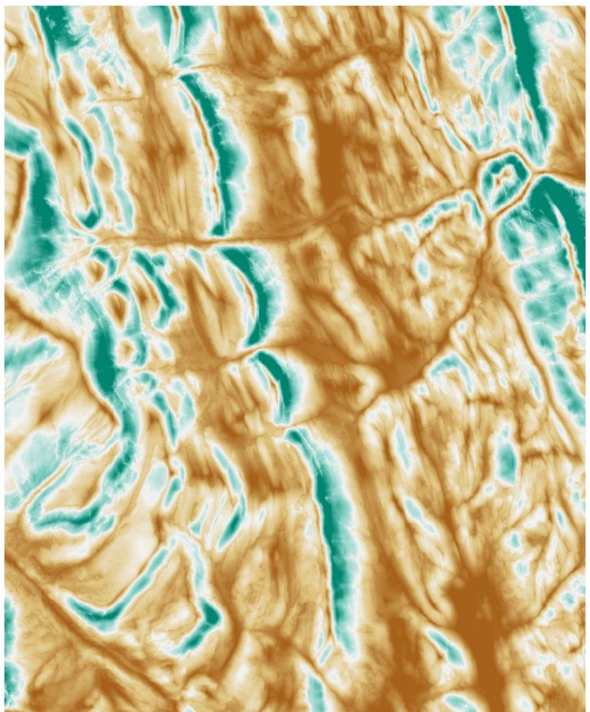


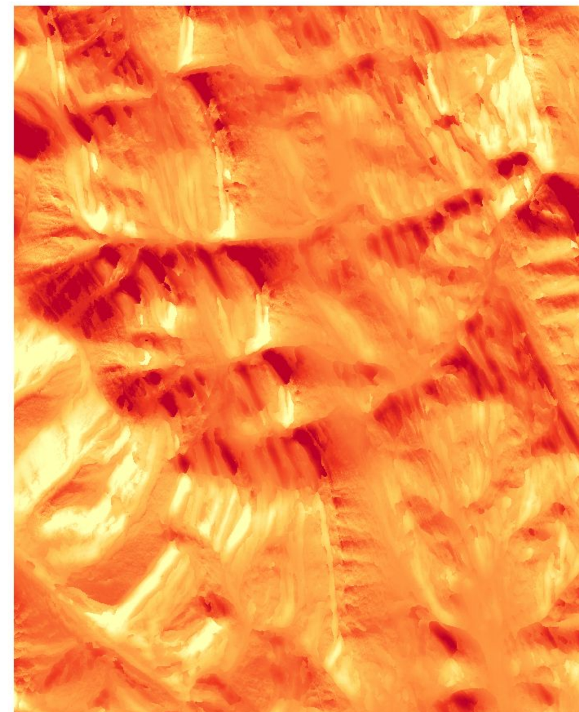


What

ktpi.R KASP

zonal analysis of, a focal analysis of the interactions of direction and slope of aspect and elevation within a given kernel (Albert R. Stage, 2007), for each feature





How

Option 1: Run a single CLI command:

```
ktpi.R statistic /input/feature/6/7.tif /input/dem /output --dem-cell-size 10
```

Option 2: Run batch on a single machine:

a. Use CLI Generator to generate commands:

```
./ktpi.R ktpi-cli --ktpi-function statistic --ktpi-function terrain ../input/features ..
/input/dems ../output --tile-col-min 0 --tile-col-max 10 --tile-row-min 0 --tile-row-max 10
--raster-cells 500 --raster-cell-size 5 -d 5 -l none
```

b. Run CLI commands generated:

```
ktpi.R statistic /input/feature/6/7.tif /input/dem /output --dem-cell-size 10
ktpi.R terrain /input/feature/6/7.tif /input/dem /output --dem-cell-size 10
ktpi.R ktpi /input/feature/6/7.tif /input/dem /output --dem-cell-size 10 --kernel-size 400
...
```

How

Option 3: Prepare tasks for AWS ECS Cluster

a. Use SQS Generator to generate SQS messages:

```
./ktpi.R ktpi-sqs --ktpi-feature feature --ktpi-function statistic --ktpi-function terrain --tile-col-min 0 --tile-col-max 10 --tile-row-min 0 --tile-row-max 10 --raster-cells 500 --raster-cell-size 5 -d 5 -l none
```

b. Push messages to SQS

c. Push data to S3

d. Configure AWS Elastic Compute Service (ECS): tools/run-instances.sh

e. Start ECS tasks: tools/start-task.sh + Dockerfile (r-spatial-raster + ktpi.R)

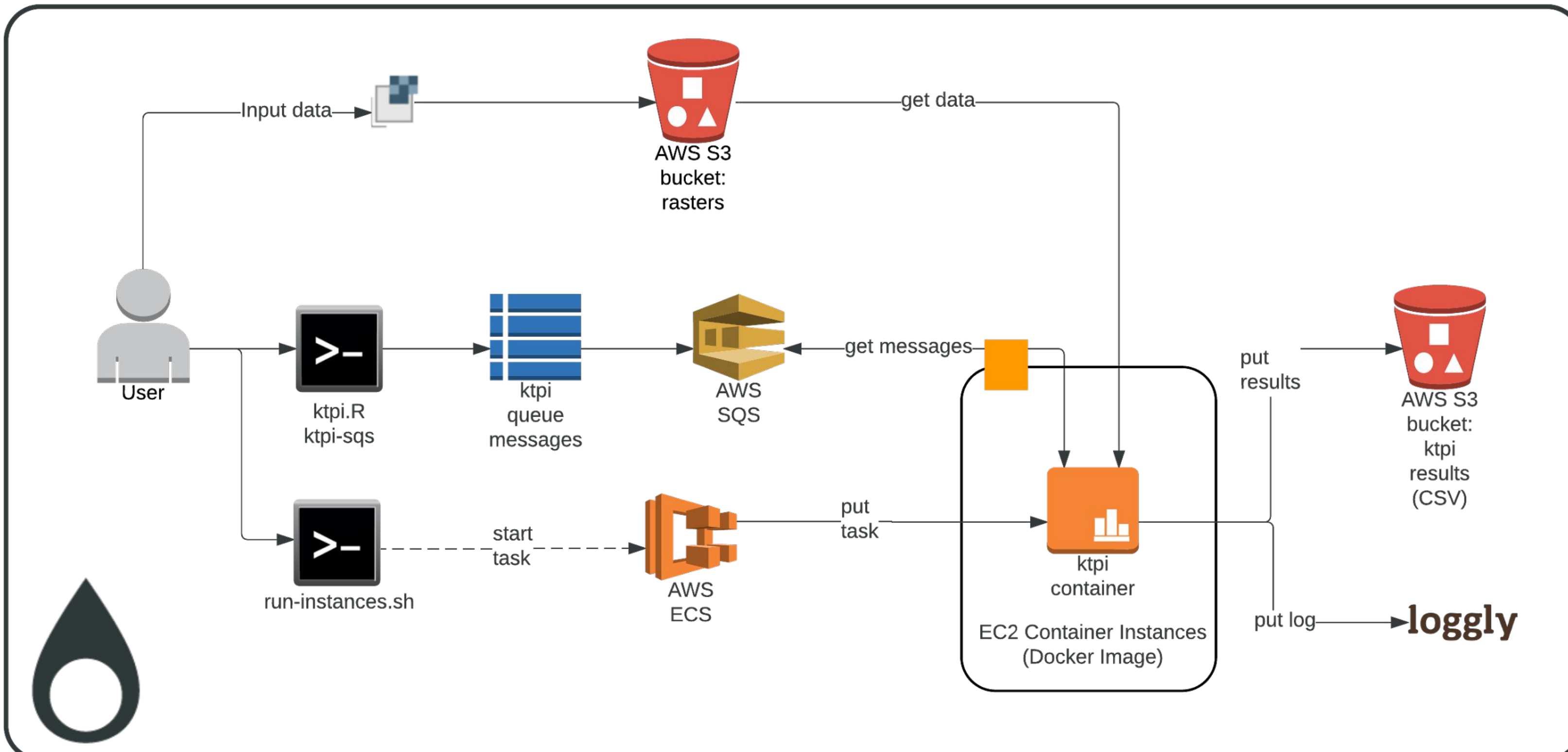
f. ECS instance retrieves messages: tools/client.js

g. ECS instance generates commands:tools/runner.sh

h. ECS instance runs ktpi.R command

i. ECS checks output & writes to S3: tools/runner.sh

Distributed in Amazon Web Services



Results

- **ECS / Docker** used to overcome challenges with deploying geospatial dependencies (R, gdal, geos, sp, raster, rgdal, rgeos, docopt)

- **SQS** as a task repository with built in failure handling

- **Isolated self contained tasks** enables graceful degradation and retry.

- **User friendly** frontend under development.

Over and out

Thank you for your time :)

tesera

Michael Kieser MSc PEng

SENIOR SYSTEMS ENGINEER - PARTNER

E michael.kieser@tesera.com **T** 1 866 698 8789
D 1 403 973 3277

tesera.com