spup – an R package for uncertainty propagation analysis in spatial environmental modelling

Kasia Sawicka and Gerard B.M. Heuvelink

(with contributions from Dennis Walvoort, Stefan van Dam and Damiano Luzzi)

Underlying methodology

Functionality

Examples

Planned applications

Acknowledgements

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 607000.

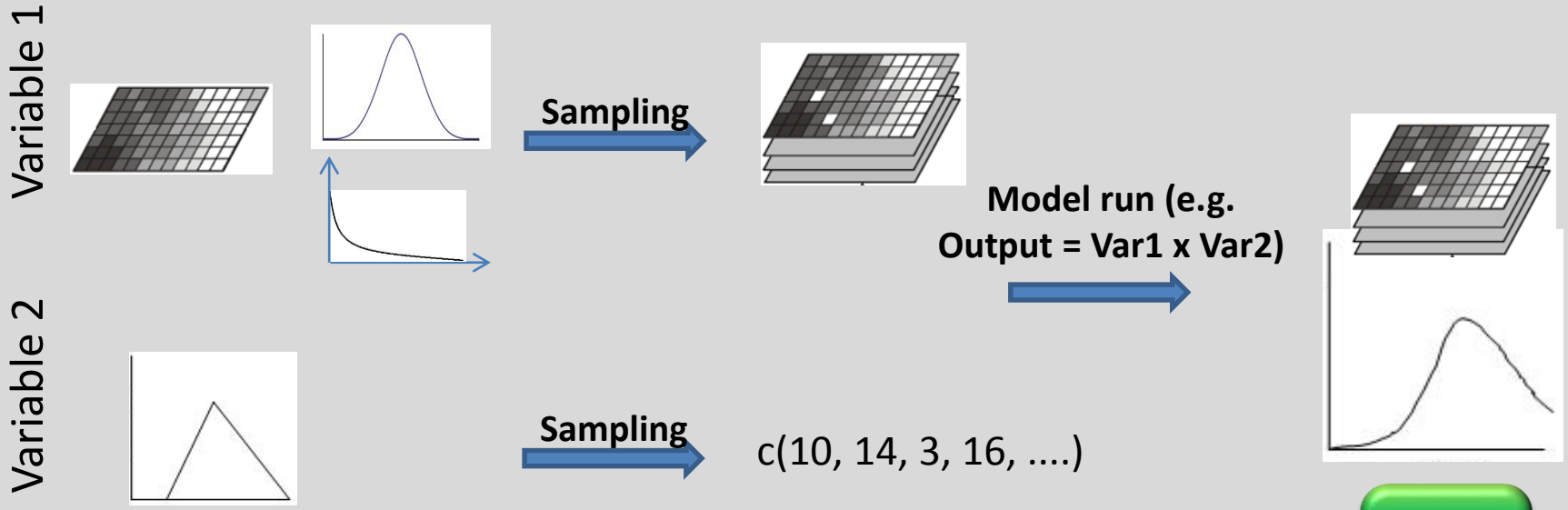
Oval shapes are clickable

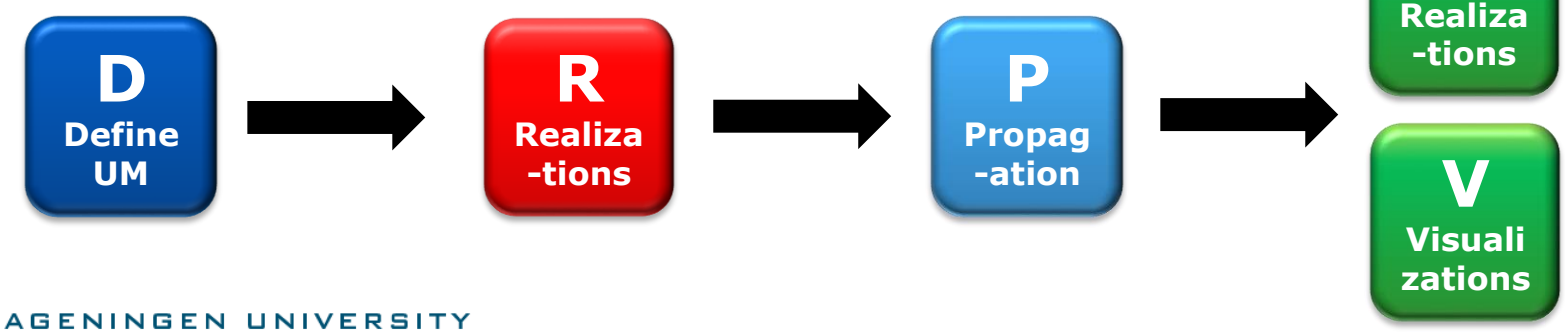
WAGENINGEN UNIVERSITY
WAGENINGEN UR

QUICS

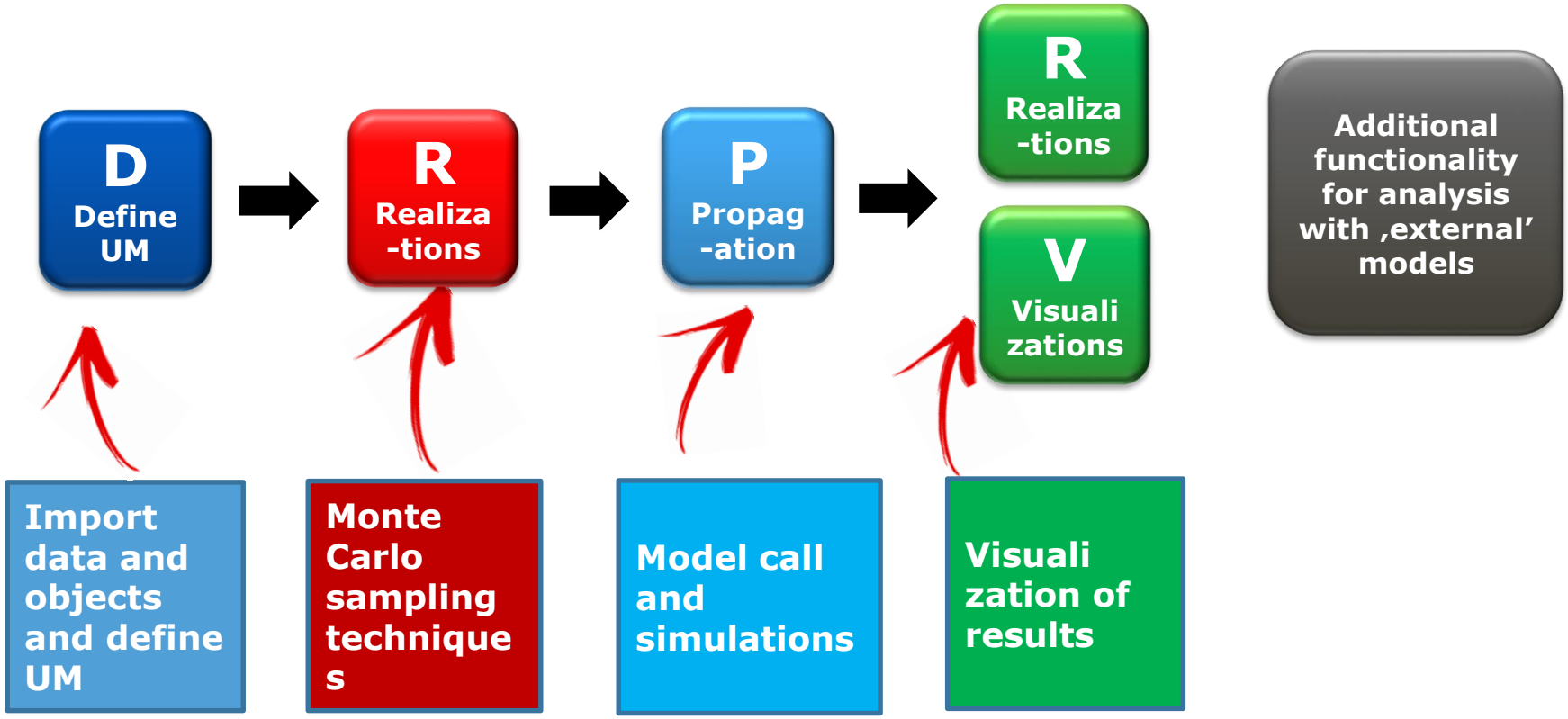
Underlying methodology

Monte Carlo approach principle



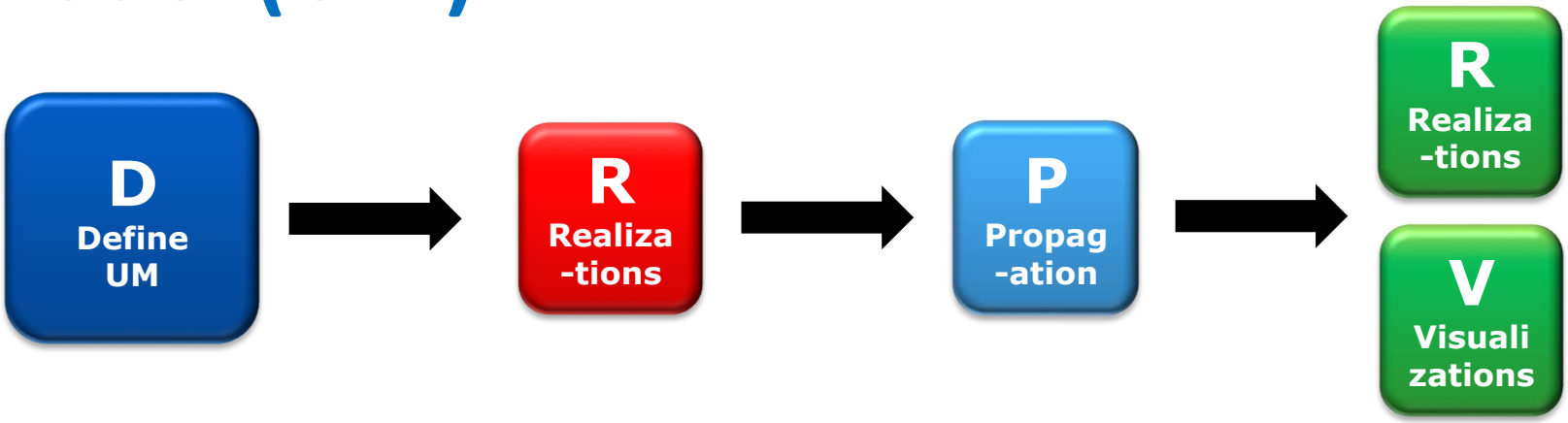


spup functionality



- Imports: gstat, magrittr, mvtnorm, purrr, raster, whisker
- Available on CRAN and GitHub

Defining uncertainty model (UM)



$$Z(x) = \mu(x) + \sigma(x) \cdot \varepsilon(x)$$

- μ is the (deterministic) mean of the variable of interest Z
- σ is a spatially variable standard deviation associated with the prediction μ
- $\varepsilon(x)$ is the (stochastic) error about it (typically zero mean, but non-zero variance and spatially correlated)

defineUM {spup}

Define an uncertainty model for a single variable

Description

Function that allows to define marginal uncertainty distributions for model inputs and subsequ

Usage

```
defineUM(uncertain = TRUE, distribution = NULL, distr_param = NULL,
  crm = NULL, categories = NULL, cat_prob = NULL, id = NULL, ...)
```

Arguments

uncertain	"TRUE" or "FALSE", determines if specification of Uncertainty Model (UM furture implementation of contributions analysis.
distribution	a string that specifies which distribution to sample from. Only in use for c

defineMUM {spup}

Define Mulivariate Uncertainty Model

Description

Function that uses output of defineUM() to define joint probability distrit

Usage

```
defineMUM(UMlist, cormatrix, ...)
```

Arguments

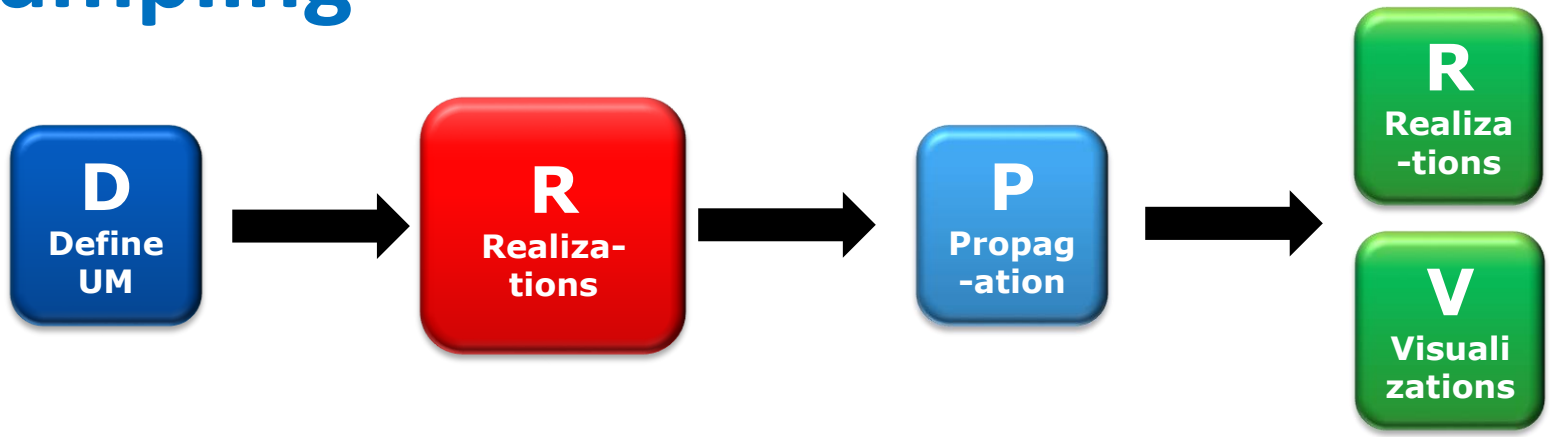
UMlist	a list of uncertain objects creaded in defineUM().
cormatrix	matrix of cross-correlations.
...	additional parameters.

More details

Defining uncertainty model (UM)

Function	Main arguments	Description	Output
defineUM()	For continuous or discrete numerical variables: type of the distribution and corresponding parameters, correlogram model; for categorical variables: categories and corresponding probabilities.	Allows to define marginal uncertainty distributions for model inputs/parameters for subsequent Monte Carlo analysis. Output class depends on the arguments provided.	If provided arguments are: type of the distribution and corresponding parameters, and corresponding parameters are spatial objects - an object of class "MarginalNumericSpatial". If provided arguments are: type of the distribution and corresponding parameters, and corresponding parameters are non-spatial objects - an object of class "MarginalScalar". If provided arguments are: categories and probabilities, and probabilities are saved in spatial object - an object of class "MarginalCategoricalSpatial". If provided arguments are: categories and probabilities, and probabilities are saved in non-spatial object - an object of class "MarginalCategoricalDataFrame".
defineMUM()	A list of outputs of defineUM(), cross-correlation matrix.	Allows the user to define joint uncertainty distributions for continuous numerical model inputs/parameters for subsequent Monte	If output of defineUM() is spatial - an object of class "JointNumericSpatial". If output of defineUM() is non-spatial - an object of class "JointScalar".

Monte Carlo sampling



S3 methods for MC sampling from uncertain inputs

More details

genSample {spup} R Documentation

Methods for generating Monte Carlo realizations from uncertain inputs.

Description

Methods for classes: "MarginalNumericSpatial", "MarginalScalar", "MarginalCategoricalSpatial", "JointNumericSpatial", "JointScalar". Function that runs Monte Carlo simulations depending on the type of uncertain object. Facilitates unconditional Gaussian simulation of errors for spatially auto-correlated residuals, as well as random and stratified random sampling if no spatial auto-correlation is included.

Usage

```
genSample(UMobject, n, samplemethod, p = 0, asList = TRUE,
  debug.level = 1, ...)
```

Arguments

UMobject	an uncertain object to sample from, output of defineUM() or defineMUM().
n	integer, number of Monte Carlo realizations.
samplemethod	a string, "ugs", "randomSampling", "stratifiedSampling", "lhs" ("lhs" currently not in use).
p	A vector of quantiles. Optional. Only required if sample method is "stratifiedSampling" or "lhs".
asList	logical. If asList = TRUE returns list of all samples as a list. If asList = FALSE returns samples in a format of distribution parameters in UMobject.

Monte Carlo sampling

Function	Main arguments	Description	Output
genSample()	Output of defineUM() or defineMUM(), number of Monte Carlo runs, sampling method, logical parameter asList.	Methods for generating Monte Carlo sample from uncertain variables.	A Monte Carlo sample of uncertain object. If logical argument asList is set to TRUE - an object of class "list", where each element is a single MC realization. If logical argument asList is set to FALSE - an object of the same class as distribution parameters (numerical vars) or probabilities (categorical vars) to sample from.

S3 methods for MC sampling from uncertain inputs

More details

Methods for generating Monte Carlo realizations from uncertain inputs.

Description

Methods for classes: "MarginalNumericSpatial", "MarginalScalar", "MarginalCategoricalSpatial", "JointNumericSpatial", "JointScalar". Function that runs Monte Carlo simulations depending on the type of uncertain object. Facilitates unconditional Gaussian simulation of errors for spatially auto-correlated residuals, as well as random and stratified random sampling if no spatial auto-correlation is included.

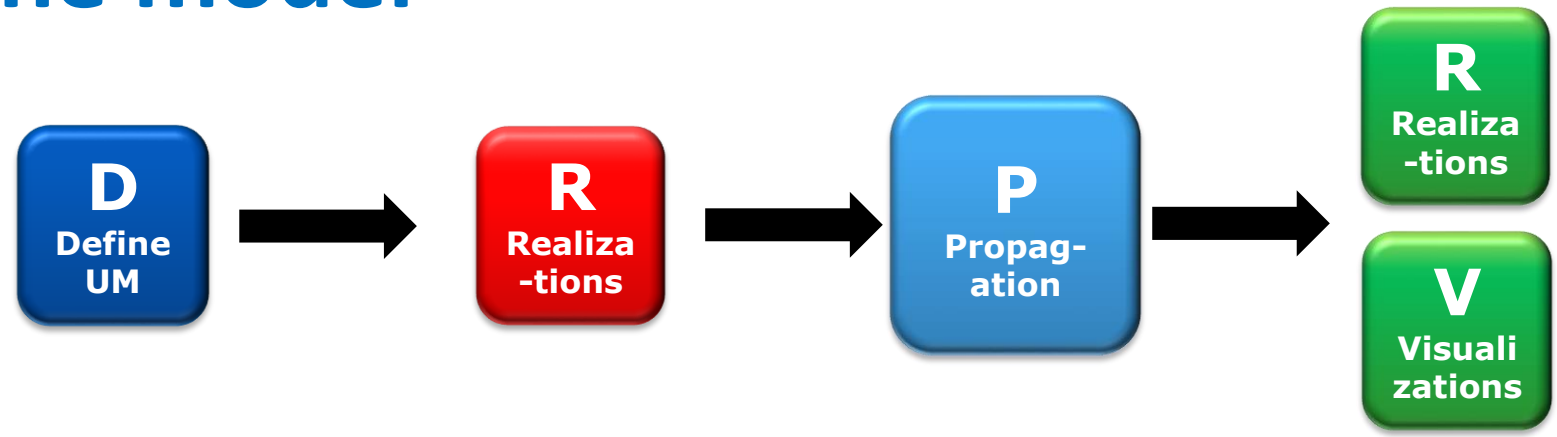
Usage

```
genSample(UMobject, n, samplemethod, p = 0, asList = TRUE,
  debug.level = 1, ...)
```

Arguments

UMobject	an uncertain object to sample from, output of defineUM() or defineMUM().
n	integer, number of Monte Carlo realizations.
samplemethod	a string, "ugs", "randomSampling", "stratifiedSampling", "lhs" ("lhs" currently not in use).
p	A vector of quantiles. Optional. Only required if sample method is "stratifiedSampling" or "lhs".
asList	logical. If asList = TRUE returns list of all samples as a list. If asList = FALSE returns samples in a format of distribution parameters in UMobject.

Propagation through the model



Run any model written as R function with a MC sample of uncertain input and additional parameters/inputs

More details

propagate {spup} R Documentation

Propagation function

Description

A function that runs a model repeatedly with Monte Carlo samples of uncertain inputs.

Usage

```
propagate(realizations, model, n, ...)
```

Arguments

`realizations`	a list where each element is a single Monte Carlo realizations if only one parameter/variable is considered uncertain; a list of such lists if more than one parameter/variable is considered uncertain.
`model`	model that is written as a function in R.
`n`	number of Monte Carlo Runs.
`...`	any further arguments that the model takes.

Propagation through the model

Function	Main arguments	Description	Output
propagate()	Monte Carlo sample or a list of Monte Carlo samples - output of genSample(), a model - wrapped in a R function, number of Monte Carlo runs.	Runs the model repeatedly with Monte Carlo realizations of the uncertain input/parameters.	A Monte Carlo sample of a model output. An object of class "list", where each element is a single MC run.

Run any model written as R function with a MC sample of uncertain input and additional parameters/inputs

More details

Propagation function

Description

A function that runs a model repeatedly with Monte Carlo samples of uncertain inputs.

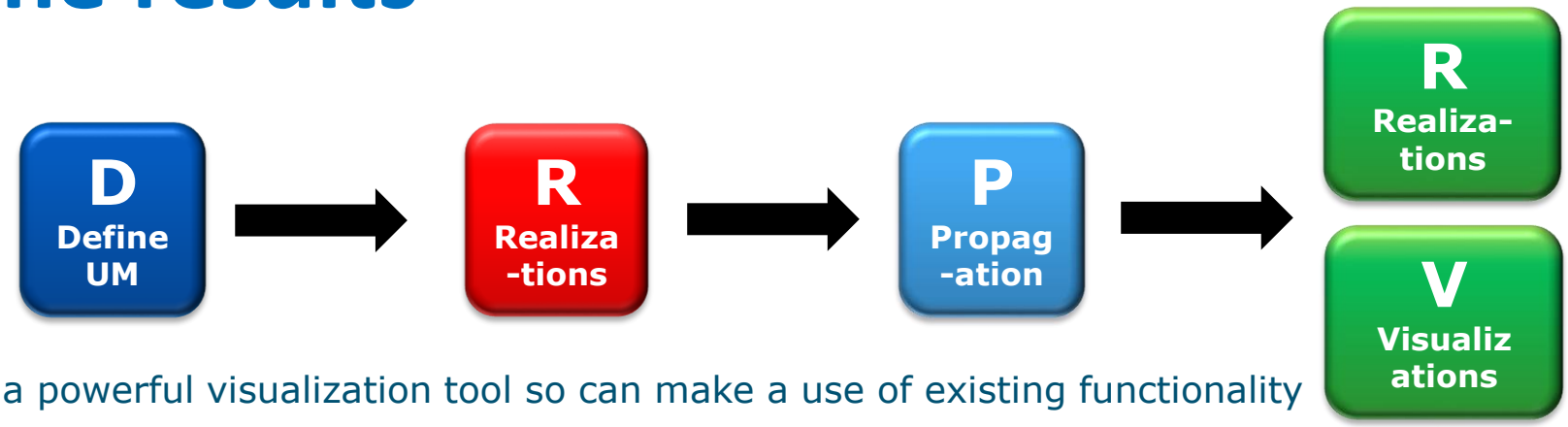
Usage

```
propagate(realizations, model, n, ...)
```

Arguments

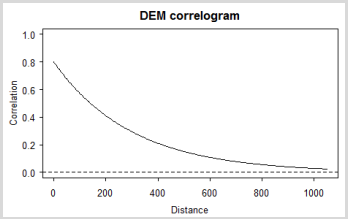
realizations	a list where each element is a single Monte Carlo realizations if only one parameter/variable is considered uncertain; a list of such lists if more than one parameter/variable is considered uncertain.
model	model that is written as a function in R.
n	number of Monte Carlo Runs.
...	any further arguments that the model takes.

Visualization of the results

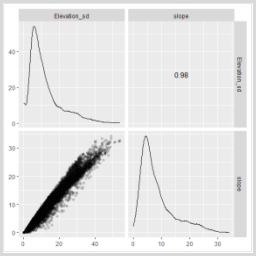


R is a powerful visualization tool so can make a use of existing functionality

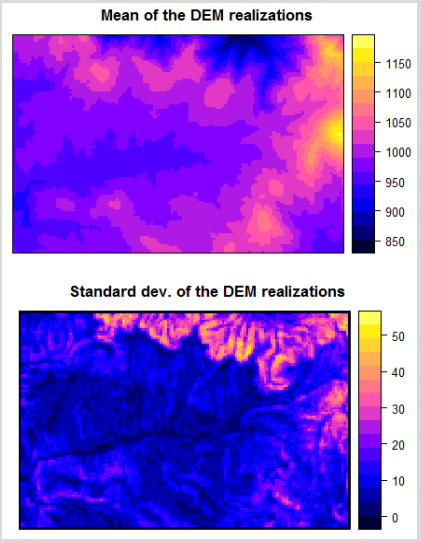
Spatial correlogram



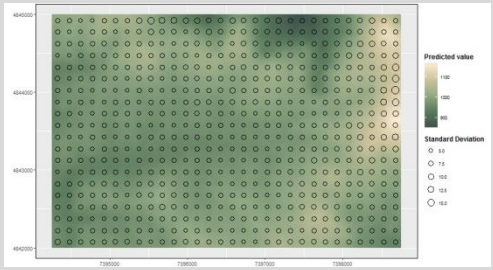
Correlations



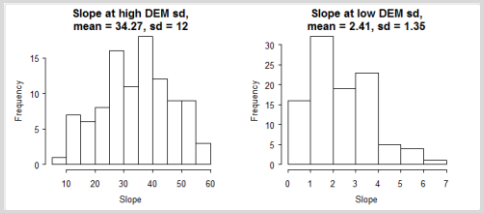
Adjacent maps



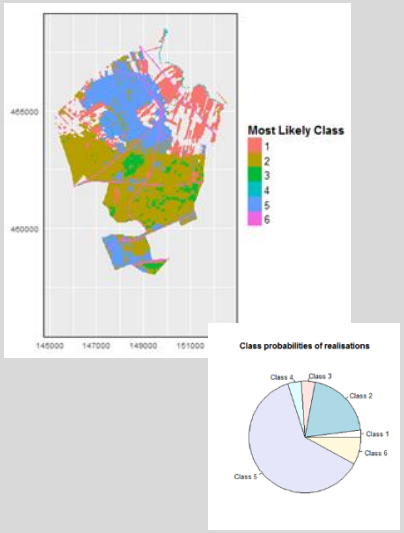
Glyphs



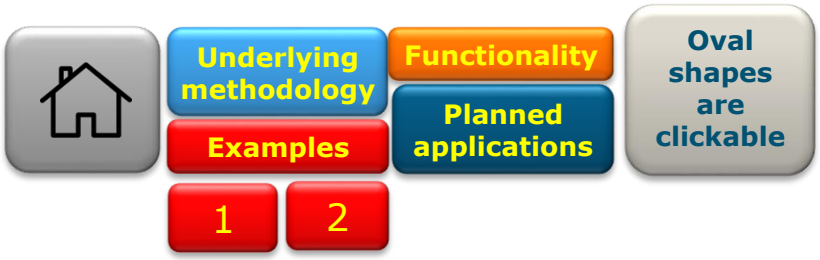
Histograms



Categorical data



Examples



Example 1

Uncertainty propagation analysis with cross-correlated numerical variables – predicting soil C/N ratio from soil organic carbon and total nitrogen content

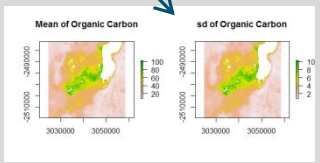
Example 2

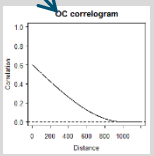
Uncertainty propagation analysis with a model writen in C – simple linear regression model with uncertain scalar parameters

Examples (1) – uncertainty propagation with cross-correlated vars.

Predicting soil C/N ratio from soil organic carbon and total nitrogen content in south region of lake Alaotra in Madagascar

1.
```
# define uncertainty model for the OC and TN
OC_UM <- defineUM(TRUE, distribution = "norm",
distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
```

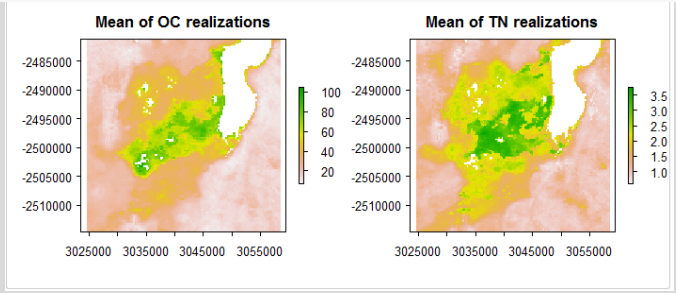




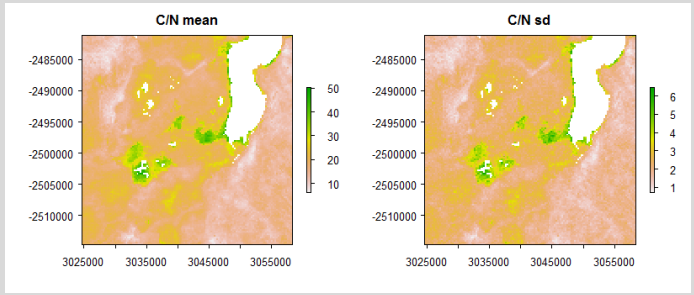
- repeat the same for TN

```
# define multivariate uncertainty model
mySpatialMUM <- defineMUM(UMlist = list(OC_UM, TN_UM),
cormatrix = matrix(c(1,0.7,0.7,1), nrow=2, ncol=2))
```

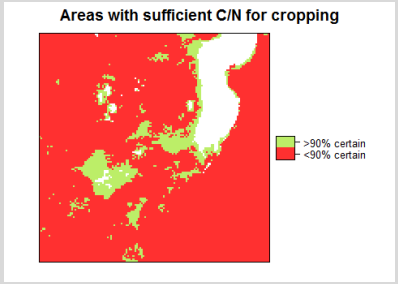
2.
```
# create possible realizations from the joint distribution of OC and TN
MC <- 100
OCTN_sample <- genSample(UMobject = mySpatialMUM, n = MC, samplemethod = "ugs", nmax = 20)
```



3.
```
# run uncertainty propagation
CN_sample <- propagate(realizations = OCTN_sample,
                       model = C_N_model_raster, n = 100)
```



4. Identify areas that are suitable for crop production with 90% certainty



Data source: ISRIC Soil Grid database (www.soilgrids.org) (Hengl et al., 2017)

Examples (2) – uncertainty propagation with ‚external' models

Simple linear regression model written in C with uncertain scalar parameters

Spatial (or other) inputs to the models are often stored in ASCII files. In that case, when using external models in R we need code to:

1. **Modify ASCII input file.**
2. Run the external model.

Modifying ASCII files - rendering

Rendering is the process of replacing the tags in moustaches by text.

For rendering ASCII input files, the mustache templating framework is implemented (https://mustache.github.io). In R this is implemented in the package `whisker`.

Function `template()` allow user to define a 'container' class to store all templates with model inputs.

A template is simply a model input file with:

1. The additional extension `.template`.
2. Input that needs to be modified is replaced by mustache-style tags.

For example, suppose we have a model that needs the input file: `input.txt`. This input file contains two parameters "b0" and "b1". The contents of the original file may look like:

```
read_lines("examples/input.txt")
```

```
[1] "-0.788907241483209 0.0155277014710009"
```

The corresponding template file should be named as `input.txt.template`. It contains:

```
read_lines("examples/input.txt.template")
```

```
[1] "{{b0}} {{b1}}"
```

A template stored as a file will always be rendered on disk.

```
my_template <- template("examples/input.txt.template")
```

with contents:

```
my_template %>%
    read_lines
```

```
[1] "{{b0}} {{b1}}"
```

Rendering will create a new file, called `input.txt`.

```
my_template %>%
    render(b0 = 3, b1 = 4)
```

```
[1] "examples/input.txt"
```

```
my_template %>%
    render(b0 = 3, b1 = 4) %>%
    read_lines
```

```
[1] "3 4"
```

Examples (2) – uncertainty propagation with ‚external' models

Simple linear regression model written in C with uncertain scalar parameters

Spatial (or other) inputs to the models are often stored in ASCII files. In that case, when using external models in R we need code to:

1. Modify ASCII input file.
2. **Run the external model.**

Running external models

An external model can be called from R by means of the `system` or `system2` function. To facilitate this, *spup* includes the wrapper function `executable()`.

```
dummy_model <- executable("examples/dummy_model.exe")
```

Running the rendering procedure allows to pass any values for b0 ad b1 and the model gives:

```
# create template
my_template <- template("examples/input.txt.template")

# render the template
render(my_template, b0 = 3.1, b1 = 4.2)

# run external model
dummy_model()

# read output (output file of dummy_model is "output.txt")
scan(file = "examples/output.txt", quiet = TRUE)
```

```
[1]  7.3 11.5 15.7 19.9 24.1 28.3 32.5 36.7 40.9
```

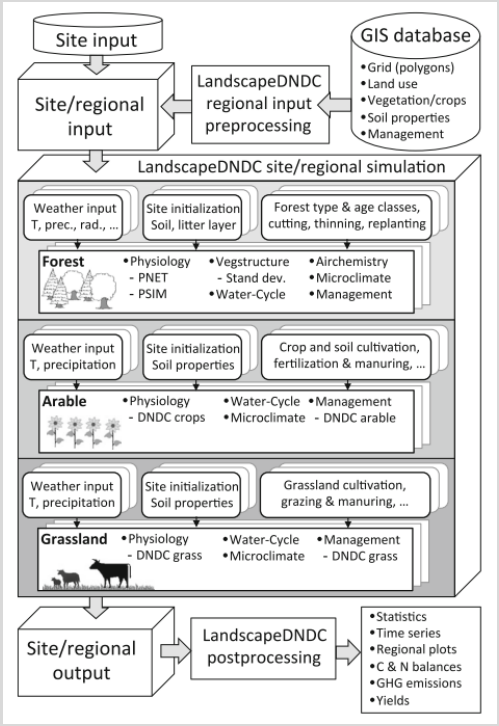
To perform the uncertainty propagation analysis we need to derive multiple realizations of the model output in steps as follows:
1. Render the template, 2. Run the model,
3. Read the results. 4. Process the results.
For example:

```
# number of Monte Carlo runs
n_realizations <- 100

n_realizations %>%
    purrr::rerun({
        # render template
        render(my_template, b0 = rnorm(n = 1), b1 = runif(n = 1))

        # run model
        dummy_model()

        # read output
        scan("examples/output.txt", quiet = TRUE)
    }) %>%
    set_names(paste0("r", 1:n_realizations)) %>%
    as_data_frame %>%
    apply(MARGIN = 1, FUN = quantile)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]
0%	-2.3500	-2.1900	-2.0300	-1.870	-1.7100	-1.6700	-1.6500	-1.6300	-1.6100
25%	-0.3550	0.0525	0.3375	0.720	1.0325	1.2325	1.3925	1.6300	1.9425
50%	0.4050	0.8400	1.2550	1.640	2.0750	2.5100	2.9850	3.5400	4.0750
75%	1.1075	1.8525	2.5825	3.245	3.9500	4.5750	5.4450	6.3075	7.0675
100%	2.9200	3.2500	4.1600	5.080	6.0000	6.9200	7.8400	8.7600	9.7100

Planned applications

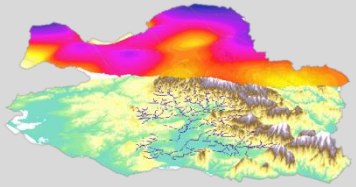
Uncertainty propagation analysis with process-based model LandscapeDNDC



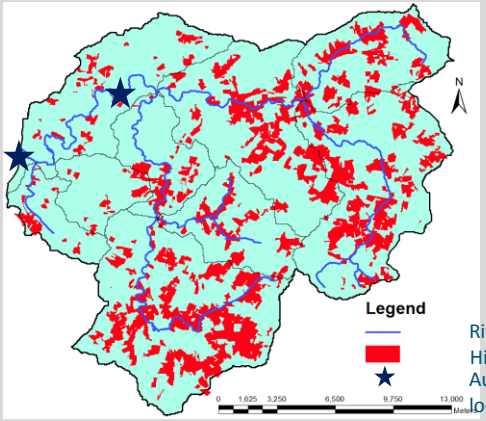
Haas et. al. (2013)

Uncertainty propagation analysis with Metaldehyde Prediction Model

Radar rainfall data



Risk map



Acknowledments

Damiano Luzzi
Stefan van Dam
Sytze de Bruin
Dennis Walvoort
QUICS fellows and partners
EU funding