Spatial stochastic simulation

D G Rossiter

Nanjing Normal University, Geographic Sciences Department
南京师范大学地理学学院
Cornell University, Soil & Crop Sciences Section

November 25, 2018

What is stochastic simulation?

Simulation is the process or result of representing what reality *might* look like, given a **model** of the system.

- studying a system without physically implementing it
- future scenarios; possible realities

Stochastic **random**

Stochastic simulation there is a random component to the simulation model

- each simulation is different
- random components are from assumed **probability distribution**

What are the stochastic components?

Model parameters → sensitivity analysis

- Which parameters most affect the model output?
- How much does the uncertainty in **parameter values** affect model output?

Model inputs uncertain data items

- How much does the uncertainty in **observation values** affect model output?

Spatial position of observations (for spatial models)

- How much does the uncertainty in the observation **location** affect model output?

Time of observations (for temporal models)

- How much does the uncertainty in observation **time** affect model output?

General procedure

1. Assume a model
2. Identify the stochastic components
3. Assume a **statistical distribution** for the stochastic component
4. Assume values of the **parameters** for each distribution
5. Repeat:
    1. **Sample** from the distribution of the stochastic component
    2. Run the model with the sampled values
    3. Collect the results of the model
6. **Summarize** the set of results → quantified uncertainty, alternate realities

Random numbers generators

- Stochastic (= "random") simulation requires **random numbers**, from various **probability distributions**
- Truly random: from apparently random physical processes, e.g., radioactive decay
- **Pseudorandom**: computed deterministically from a starting **seed**, but appear to be random
  - A large number of tests for apparent randomness, e.g., lack of serial correlation
  - See `?Random` for a description of R random number generators, with references for the algorithms
  - `set.seed` function to initialize the random number generator (to reproduce examples)
  - otherwise, an initial seed is created from the current time and the process ID, and then updated as numbers are generated

Random numbers from probability distributions

- R has a set of functions to draw randomly from many probability distributions
- These each have appropriate **parameters**
- Some R functions and their parameters:

`runif` **Uniform** distribution; all values on [0...1] equally likely

`rnorm` **Normal** (Gaussian) distribution: mean μ, standard deviation σ

`rbinom` **Binomial** distribution: probability of success in one trial θ

`rpois` **Poisson** distribution: mean (and variance!) number of occurrences in a time period λ

`rbeta` **Beta** distribution: two shape parameters α and β

Probability distribution density functions

`runif` $f(x \mid a, b) = 1/(b-a))$; special case for $b = 1$, $(a = 0)$: density is 1 everywhere.

`rnorm` $f(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\}$

`rbinom` $f(k, n \mid \theta) = \binom{n}{k}\theta^k(1-\theta)^{n-k}$

`rpois` $p(k \mid \lambda) = e^{-\lambda}\frac{\lambda^k}{k!}$

`rbeta` $f(x \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}$

Generating uniform random numbers on $[0 \ldots 1]$

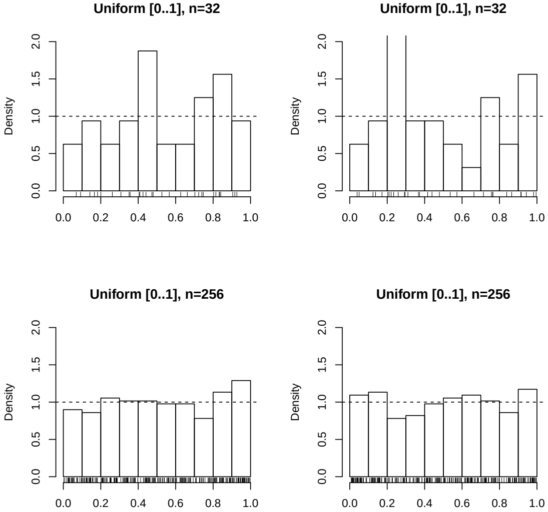
Conceptually (various clever algorithms make this more efficient):

- Generate pseudo-random integers on $[0 \ldots 2^W - 1]$, where W is the computer word length in bits
  - $W = 16 \rightarrow 65535$, $W = 16 \rightarrow 4294967295$, $W = 64 \rightarrow 1.844674 \cdot 10^{19}$
  - various algorithms, e.g., 32-bit Mersenne Twister
- Convert to fractions by dividing by the word length
  - precision even for 16 bits is 0.000015

Examples of R random numbers

```
> runif(10)
 [1] 0.9064575 0.8595720 0.5118016 0.8829810 0.3210650
 [6] 0.2674023 0.6485969 0.9319358 0.3415350 0.6231881
> rnorm(10, mean=10, sd=1)
 [1] 11.254860  9.538351 10.511656  9.759389  9.222882
 [6] 10.747971 11.317742 10.659810 10.538297 11.172101
> rbinom(10, size=24, prob=0.5)
 [1] 15 15 11 13 11 15 15 11  8 13
> rpois(10, lambda=3)
 [1] 9 3 2 3 1 1 6 1 0 0
> rbeta(10, shape1=10, shape2=3)
 [1] 0.7365506 0.6838790 0.7447469 0.5507566 0.4955479
 [6] 0.6605212 0.9126238 0.8364062 0.6262444 0.8169596
```

Random numbers: uniform distribution

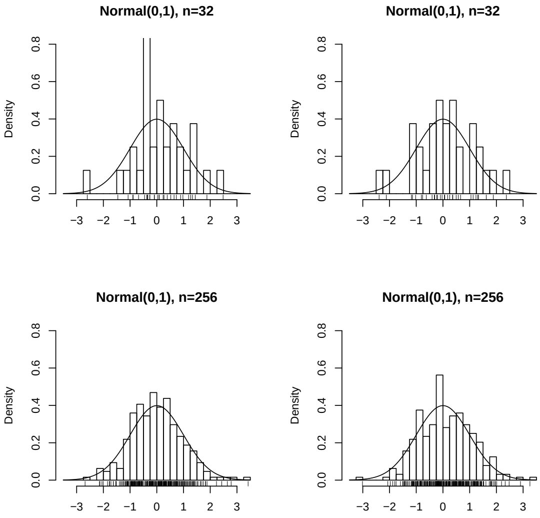


Generating random numbers from probability distributions

Conceptually: (various clever algorithms make this more efficient)

1. Start with Uniformly-distributed variates U
2. Find inverse F^{-1} of of Cumulative Distribution Function (CDF) F
   - e.g.: Normal: $F^{-1} = \mu + \sigma\sqrt{2}\mathrm{erf}^{-1}(2u - 1)$
   - $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,dt$
3. Inverse transform
   - continuous: $X = F^{-1}(U)$
   - discrete: $X = \min\{x : F(x) \geq u\}$

Random numbers: normal distribution



Use of random samples in simulation

- **Assume** a model with stochastic components
- **Assume** a probability distribution for each component
- **Assume** values of the parameters
  - Usually from previous experiments
  - Or, from a hypothesis to test
  - May have correlations between these, i.e., **conditional** distributions
- Make many random draws from these distributions; each is **equally likely**
- Run the model many times, each with different random values

Non-spatial simulation

- Simple example: simulating a **binomial outcome**
  - The number k of "successes" in n independent, exchangeable[1] Bernoulli trials
  - two mutually-exclusive possible outcomes conventionally referred to as 1="successes" and 0="failures"
  - the **process** is **stochastic**: a given probability of success of any one trial
  - One model **parameter**: $\theta \in [0 \ldots 1]$,
  - result follows the Binomial distribution:

$$p(k) = \binom{n}{k} \theta^k (1 - \theta)^{n-k}$$

  - Typical example: a set of flips of a coin (if fair, $\theta = 0.5$), where "heads" is counted as a success.

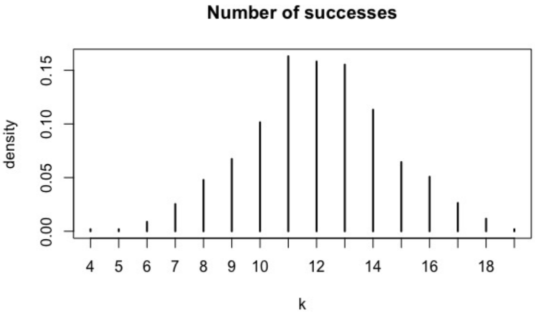
[1] i.e., their order does not matter

Binomial simulation

Simulate 1024 sets of 24 flips of a fair coin:

```
> sample <- rbinom(1024, size=24, prob=0.5); head(sample, n=20)
 [1] 10  8  9 15 14 14 12 12 10 12 10 11  9 10 13 11  9 12 14 14
> (table.k <- table(sample))
  4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  2   2   9  26  49  69 104 167 162 159 116  66  52  27  12   2
> plot(table.k/1024, xlab="k", ylab="density")
```

Note that although 12 of 24 are expected, outcomes from 4 to 19 are possible if we do this 1024 times!
In this simulation 12 is not the mode (most frequent)! It is 11.

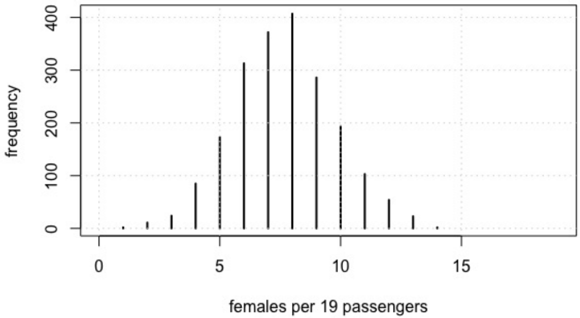


A simple non-spatial simulation

- Risk of an overweight airplane on full 19-seat plane
- Passengers weights assumed to follow a **normal** distribution
  - Estimate mean and standard deviation from measurements from the target population
    - separate distributions for males/females; hierarchical model gender **binomial** → gender-specific **normal**
  - Estimate mean proportion of female passengers (parameter of **binomial**)
- Simulate number of females/males of the 19, from binomial distribution
- Simulate each individual's weight; sum all 19
- Compare to maximum allowable weight; find proportion overweight

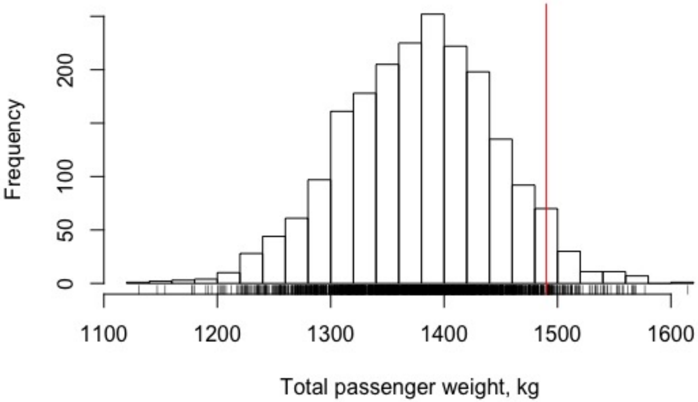
```
# parameters: mean, s.d. of fe/male weights, kg
mu.m <- 80; sd.m <- 14; mu.f <- 65; sd.f <- 12
# parameter: mean proportion of female passengers
prop.f.mu <- 0.35
# Fairchild Metro II: empty 3380 kg, max takeoff 5670kg
load.wt <- (5670-3380); pilots.wt <- 200; fuel.wt <- 600
n <- 19 # number of passengers

nsim <- 2048 # number of simulations
n.females <- vector(mode="integer", length=nsim)
wt.sum <- vector(mode="integer", length=nsim)
for (run in 1:nsim) {
  num.f <- rbinom(n=1, size=n, prob=prop.f.mu)
  num.m <- n - num.f
  wts.f <- rnorm(num.f, mean=mu.f, sd=sd.f)
  wts.m <- rnorm(num.m, mean=mu.m, sd=sd.m)
  n.females[run] <- num.f
  wt.sum[run] <- ceiling(sum(wts.f) + sum(wts.m))
  }
(n.overweight <- sum(wt.sum > (load.wt-pilots.wt-fuel.wt)))
(prob.overweight <- round(n.overweight/nsim,3))
```

2048 simulations; number of females



Per 19 passengers; $\theta = 0.35$.

2048 simulations; proportion overweight 4.5%



Spatial simulation

- The simulation is **spatial** when:
  - The model is explicitly spatial (observations, covariates, predictions); or
  - The model depends on spatial location and/or covariation
    - Spatial correlograms and variograms depend on the spatial separation between observations
    - Kriging depends on the fitted model of spatial correlation, and the positions of the observations
- Provides a **hypothetical map** of a **possible reality . . .**
  - . . . or the results of some process assuming that hypothetical map
  - Example: future land use patterns assuming some process

Simulation of a random spatial sample

- Purpose: simulate a random process over space
- Assume a probability distribution in two coördinates
  - These could be correlated! → anisotropic sample
- For **completely random**: **independent uniform** distributions

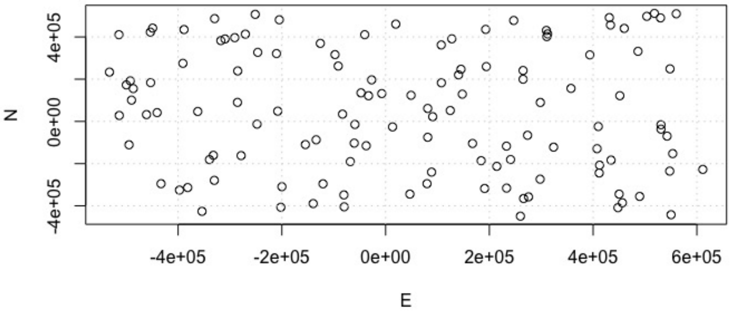
```
> bbox(ne.m)
        min      max
E -536347.6 617037.2
N -454496.7 515513.2
> (e <- runif(10, min=bbox(ne.m)["E","min"],
                  max=bbox(ne.m)["E","max"]))
 [1]  257361.16 -436644.63 -329367.76   66955.62  228298.45
 [6] -292064.55   82650.80  584037.55   97505.06 -379035.29
> (n <- runif(10, min=bbox(ne.m)["N","min"],
                  max=bbox(ne.m)["N","max"]))
 [1]   72812.16 -229296.74  289742.70  -15162.03  134348.71
 [6]  119672.13  167620.98  334487.12 -313401.27  377473.12
```

A simpler approach for a spatial object

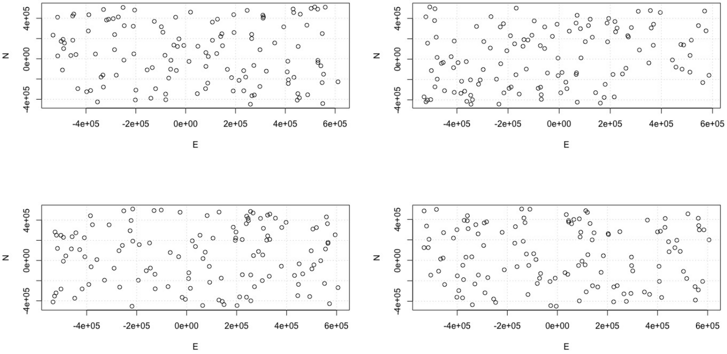
```
> class(ne.m)
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
> (spsample(ne.m, n=10, type="random"))
SpatialPoints:
              E           N
 [1,] -252995.9 -226379.683
 [2,] -269383.8 -245869.032
 [3,] -412918.2 -431693.047
 [4,]  143967.9  326646.091
 [5,] -462451.8  186514.584
 [6,]  523919.4  198377.268
 [7,] -439342.0 -203956.861
 [8,] -254776.4  139335.270
 [9,]  145935.7   -2113.977
[10,] -251060.5  -19041.134
Coordinate Reference System (CRS) arguments: +proj=aea
+lat_0=44.5 +lat_1=42 +lat_2=47 +lon_0=-90 +ellps=WGS84
+units=m
```

Simulating a completely random spatial sample

```
> points <- spsample(ne.m, n=124, type="random")
> plot(coordinates(points)); grid()
```



Comparing simulations



- Equally probable results of the same spatial process
- Same geostatistical properties

Geostatistical simulation

The spatial simulation is **geostatistical** when the model is geostatistical.

- Possible **models of spatial correlation**, given the uncertainty in the observations (positons and/or data values)
- Possible **predictive maps** made by geostatistical methods (e.g., kriging)
- Deeper reason: the **theory of regionalized random variables**

What is geostatistical simulation?

- the construction of a gridded surface corresponding to a **random function**, i.e., model of spatial correlation
- the statistical properties of the surface match those of the sample: spatial mean, spatial variance, semivariogram (model, partial sill, nugget variance, range parameter)
- **Gaussian** simulation assumes that the target field is multivariate Gaussian, with a defined **stationary** spatial mean and covariance structure
- This generates multiple, **equally probable** "realities", i.e., the spatial distribution of the target attribute

Why geostatistical simulation? (1)

- The **theory of regionalized variables** assumes that the values we observe come from some **random process**
  - simplest case: one **expected value** (first-order stationarity) with a **spatially-correlated error** that is the same over the whole area (second-order stationarity).
- The one reality we observe is the results of a random process
- There are **"alternative realities"**; that is, spatial patterns that, by this theory, *could have* occurred in another **realization** of the same **spatial process.**

Why geostatistical simulation? (2)

- Maps made by kriging are *unrealistically smooth*, especially in areas with low sampling density.
  - The nugget variance is *not* reflected in adjacent prediction points, since they are computed from the same observations, with almost the same weights.
- So, any **2D process model** using these maps as an input will not be able to properly account for local noise in the input
  - Example: hydraulic conductivity in soils, if water flows **laterally** as well as **vertically**

When must geostatistical simulation be used?

- Goovaerts: “Smooth interpolated maps should not be used for applications sensitive to the presence of **extreme values and their patterns of continuity.**” (p. 370)
  - Example: ground water travel time depends on sequences of large or small values (“critical paths”), not just on individual values.

Applications of geostatistical simulation

- If the distribution of the target variable(s) over the study area is to be used as input to a **model**, then the uncertainty is represented by a number of simulations.
- Procedure:
  1. Simulate a “large” number of realizations of the spatial field
  2. Run the model on each simulation
  3. Summarize the output of the different model runs
- The statistics of the output give a direct measure of the **uncertainty** of the model in the light of the sample and the model of spatial variability.

Local vs. global uncertainty

- Kriging prediction also provides a **kriging prediction variance** at each prediction location. This is assumed to represent the variance of a normally-distributed target.
- At each prediction location we obtain a **probability distribution** of the prediction, a measure of its **uncertainty.** This is sufficient to evaluate each prediction *individually*.
- It is *not* valid to evaluate the *set* of predictions! Reason: Errors are *by definition* spatially-correlated (as shown by the fitted variogram model), so we can't simulate the error in a field by simulating the error in each point separately.
- **Global uncertainty** is a representation of the error over the **entire field of prediction locations** at the same time.

Conditional geostatistical simulation

- This simulates the field, while **respecting the sample**, i.e., the known observed values.
- The simulated maps resemble the best (kriging) prediction, but usually much more spatially-variable (depending on the magnitude of the nugget).

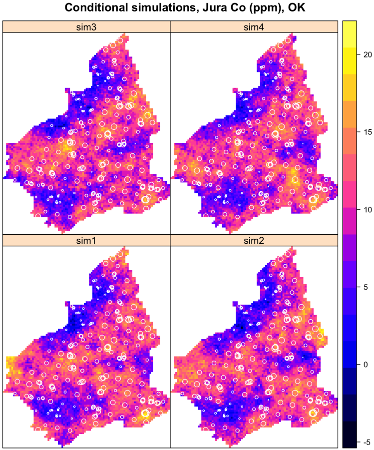
What is preserved in conditional simulation?

1. **Mean** over field
2. **Spatial correlation structure**
3. **Observations** (sample points are predicted exactly)

See figures on the next page.

The OK prediction is then reproduced for comparison.

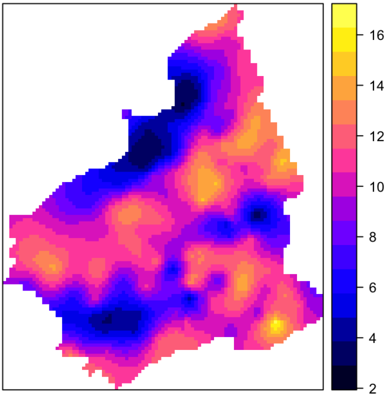
Same model, different realizations



Jura Co concentration; known points over-printed as post-plot

Q: How are the similar? How are they different?

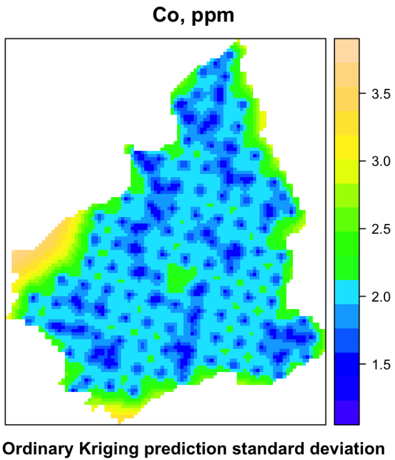
OK prediction – single “best” prediction



Ordinary Kriging prediction

Q: What are the similarities and differences between the conditional simulations and the OK prediction?

OK prediction standard deviation



OK vs. conditional simulation maps

- Simulations are much noisier, OK is smooth
- Near known points the predicted values are similar in OK and all the simulations
- Further than the variogram range from known points: OK predicts the spatial mean, simulation shows a possible reality
- All simulations have a similar spatial pattern, but not the same locations for the pattern

Unconditional geostatistical simulation

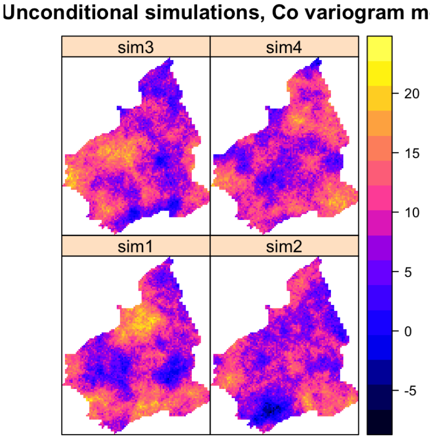
- In **unconditional** simulation, we simulate the field with *no reference to the actual sample*, i.e. the data we have. (It's only one realisation, no more valid than any other.)
- This is used to **visualise a random field** as modelled by a variogram, *not* for prediction.
- Commonly used to investigate sampling plans, assuming a spatial structure of the target variable.
    - Example: how many points are needed for a reliable variogram?

What is preserved in unconditional simulation?

1. **Mean** over field
2. **Covariance structure**

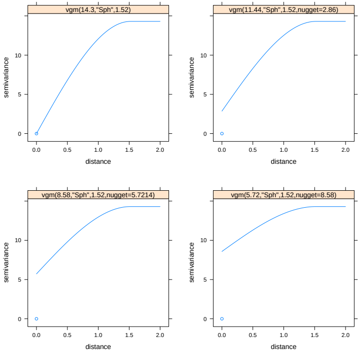
See figure on the next page. Note the similar degree of spatial continuity, but with no regard to the values in the sample.

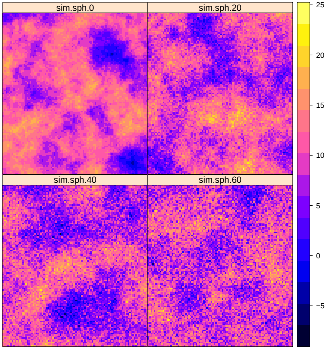
Same model, different realizations



Q: In what respect do the unconditional simulations resemble each other? In what respect do they not? Why?

Unconditional simulation: increasing nugget



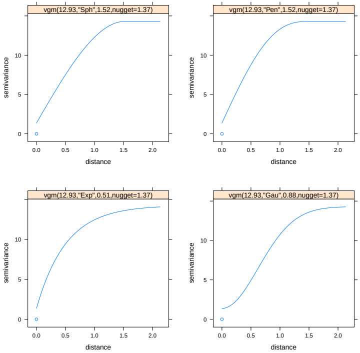


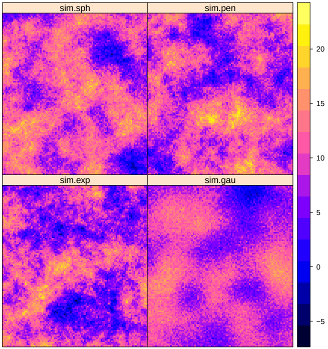
Variogram models

Simulated fields

Q: What is the effect on the simulated random field of increasing the nugget variance?

Unconditional simulation: different models





Variogram models

Simulated fields

Q: What is the effect on the simulated random field of assuming different models of spatial correlation?

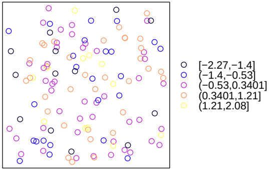
Unconditional simulation to test sampling strategies

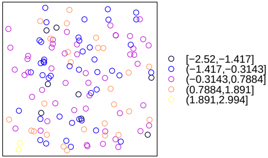
- Simulate a random field with an assumed spatial correlation structure
- Place sample points on the field according to some sampling plan
  - completely random, gridded, clustered . . .
- Extract the simulated data values at the sample points
- Use these to compute some statistic of interest (e.g., mean) or to build a variogram model
- Repeat steps (2)-(4) and summarize the results

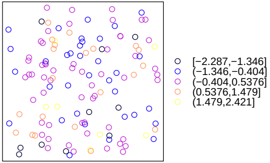
Simulate a completely random sample

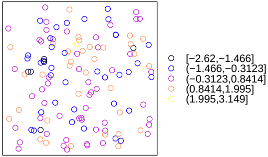
128 points, values obtained from simulated field

Variogram model: spherical, total sill=1, nugget=0, range=10





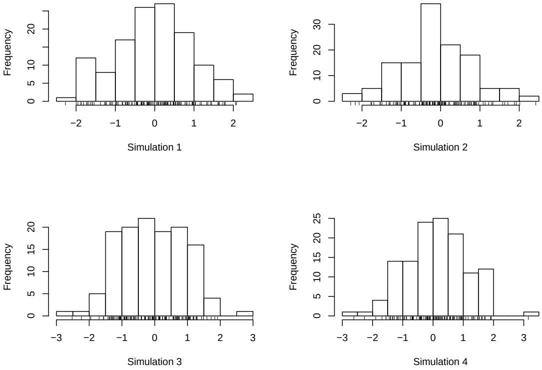




How well does the simulation reproduce the non-spatial statistics?

These should all be $(\mu = 0, \sigma = 1)$

```
[1] -0.04172984  0.97304138
[1] -0.1178978  0.9096889
[1] -0.06316958  0.98974256
[1] 0.1151962 1.0269684
```

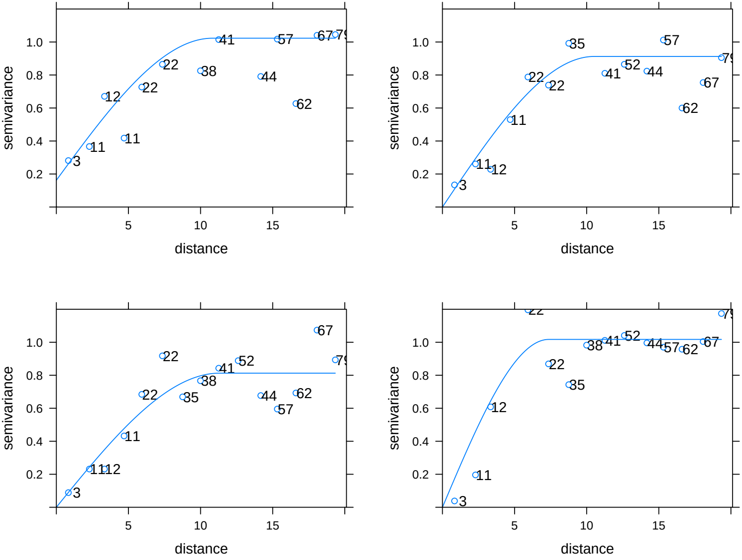


How well does the simulation reproduce spatial covariance structure?

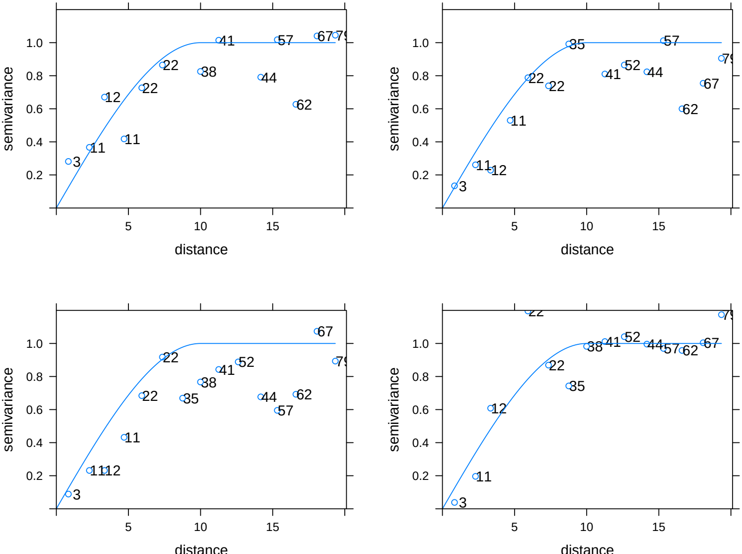
These should all be `psill=(0, 1)` (i.e., no nugget), `range=(0, 10)`

```
  model     psill    range
1   Nug 0.1619946  0.00000
2   Sph 0.8610614 10.82296
  model    psill    range
1   Nug 0.000000  0.00000
2   Sph 0.812842 11.31871
  model     psill    range
1   Nug 0.0000000  0.00000
2   Sph 0.9125189 10.51511
  model    psill    range
1   Nug 0.000000 0.000000
2   Sph 1.017288 7.371042
```

Variogram models fit from sample



Known variogram models vs. empirical variogram from sample



Spatial simulation algorithms

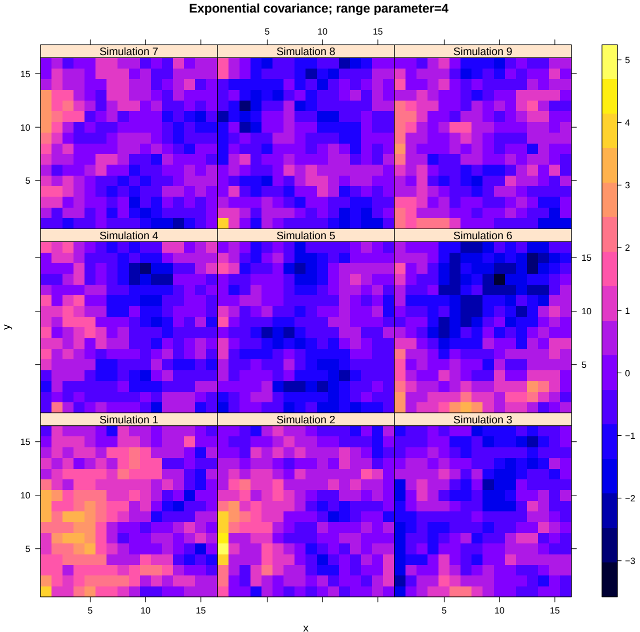
So, how are these simulated random fields calculated? **Conditional sequential simulation** as used in the **gstat** package; in simplified form:

1. Place the data on the prediction grid
2. Pick a random unknown point; make a kriging prediction from the known points, along with its prediction variance
3. Assuming a normally-distributed prediction variance, simulate one value from this; add to the kriging prediction and place this at the previously-unknown point
4. This point is now considered "known"; repeat steps (2)-(3), following a **random path** through the locations, until no more points are left to predict

Unconditional simulation

- The idea here is to simulate the entire field at once, given a **covariance** structure, e.g., exponential with a range constant.
- Algorithm for small, square random fields:
  1. set up a square matrix to represent the field; these are the prediction points
  2. compute the inverse distances between each point, as a *symmetric* square matrix
  3. convert the distances to covariances between points, using the covariance function: matrix C
  4. decompose (Cholesky) into lower triangular and its conjugate: $C = AA^T$
  5. multiply each row of the upper triangle with a vector z of random normal variates with $\sigma^2 = 1$: $y^* = A^T z$
  6. $\text{Var}(y^*) = \text{Var}(Az) = A\text{Var}(z)A^T = C$ because $\text{Var}(z) = 1$
- This preserves the correlation structure! but has a (spatially-correlated) stochastic part.

Some unconditional simulations



R/gstat code for simulation

```
> library(gstat); library(sp)
> data(jura)
> coordinates(jura.cal) <- ~Xloc + Yloc  # known points
> coordinates(jura.grid) <- ~Xloc + Yloc # grid to predict over
> ## empirical variogram
> v <- variogram(Co ~ 1, loc = jura.cal, cutoff = 1.6)
> ## fitted variogram model
> vmf <- fit.variogram(v, vgm(12.5, "Pen", 1.2, 1.5))
> ## conditional simulation
> k.sim.4 <- krige(Co ~ 1, loc = jura.cal, newdata = jura.grid,
                   model = vmf, nsim = 4, nmax = 128)
> ## unconditional simulation
> k.sim.4.u <- krige(z ~ 1, loc = NULL, newdata = jura.grid,
                     model = vmf, nsim = 4, nmax = 128,
                     beta = mean(jura.cal$Co), dummy = T)
```

Note that unconditional simulation requires a known spatial mean beta, as well as the fitted variogram model

References

- Goovaerts, P., 1997. *Geostatistics for natural resources evaluation*. Applied Geostatistics Series. Oxford University Press, New York; Chapter 8.
- Emery, X. (2008). *Statistical tests for validating geostatistical simulation algorithms*. **Computers & Geosciences**, 34(11), 1610-1620. `doi:10.1016/j.cageo.2007.12.012`
- Pebesma, E. J., & Wesseling, C. G. (1998). *Gstat: a program for geostatistical modelling, prediction and simulation*. **Computers & Geosciences**, 24(1), 17-31.

End