Speeding R up on your computer by parallelized computations — a geostatistical case study

Andreas Papritz

Department of Environmental Systems Science, ETH Zurich

Statistik Stadt Zürich

`papritz@env.ethz.ch`

outline

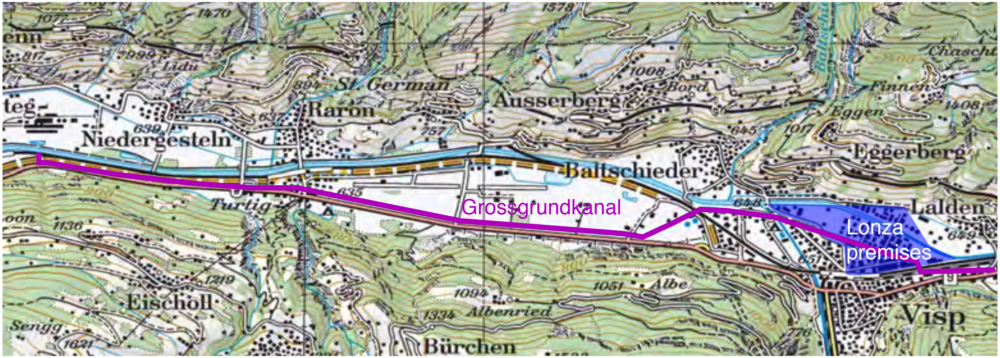
- motivating example: mercury soil contamination in Visp, Valais
- geostatistics in a nutshell
- case study: analysis of mercury topsoil content in Visp West
- “embarrasingly” parallel computations in R
- parallelizing computations for geostatistical analyses

mercury soil contamination in Visp

mercury soil contamination in Visp

- Lonza Ltd operates since 1909 a chemical plant in Visp, Canton of Valais
- mercury (Hg) used for many decades as catalyst in production of chemical compounds
- Hg disposed as sewage into river Rhône by Grossgrundkanal (GGK)

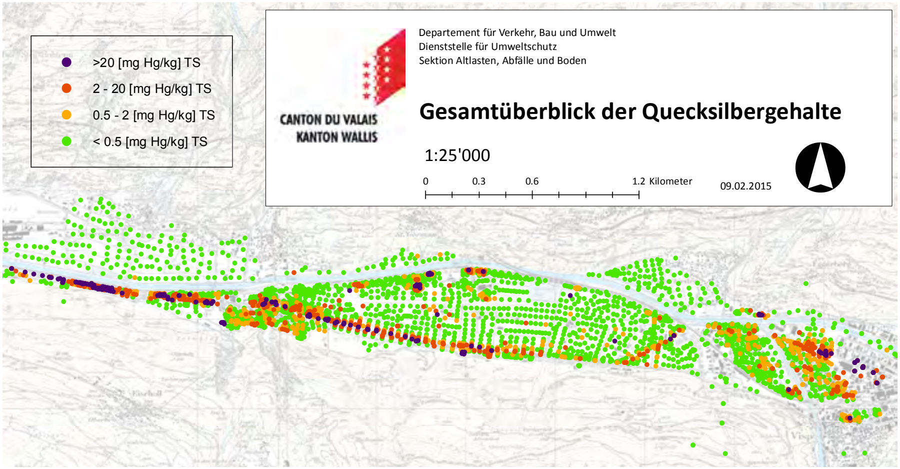
situation


Grossgrundkanal
Lonza premises
Niedergesteln
Raron
St. German
Ausserberg
Baltschieder
Eggerberg
Lalden
Visp
Eischoll
Bürchen
Albenried
Turtig

mercury soil contamination in Visp

- Hg soil pollution "discovered" in course of construction of motorway (2007–2011)
- historical study: dispersal of Hg contaminated sediments of GGK during maintenance until 1988
- since 2011 extensive soil pollution survey

mercury soil contamination in Visp



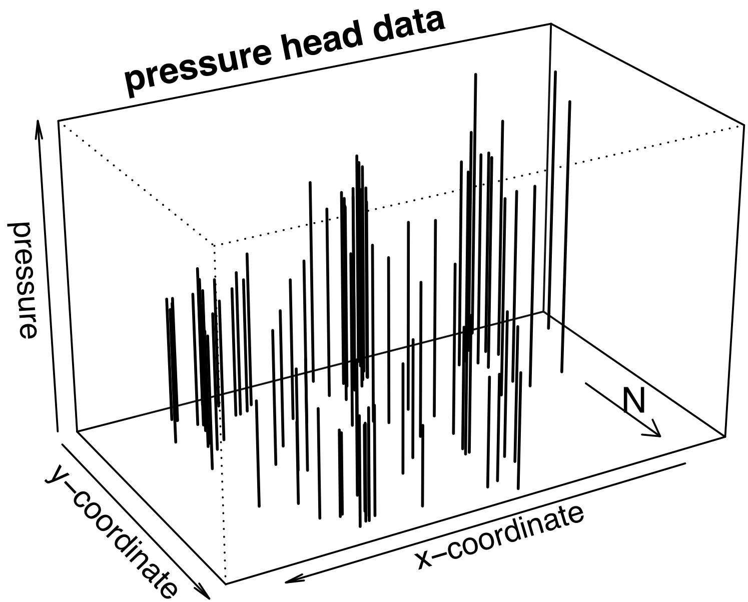
⇒ $>$ 70 parcels in residential areas must be cleaned up

⇒ geostatistical study to map spatial extent of Hg pollution and delimit areas that need further study for decision about clean-up

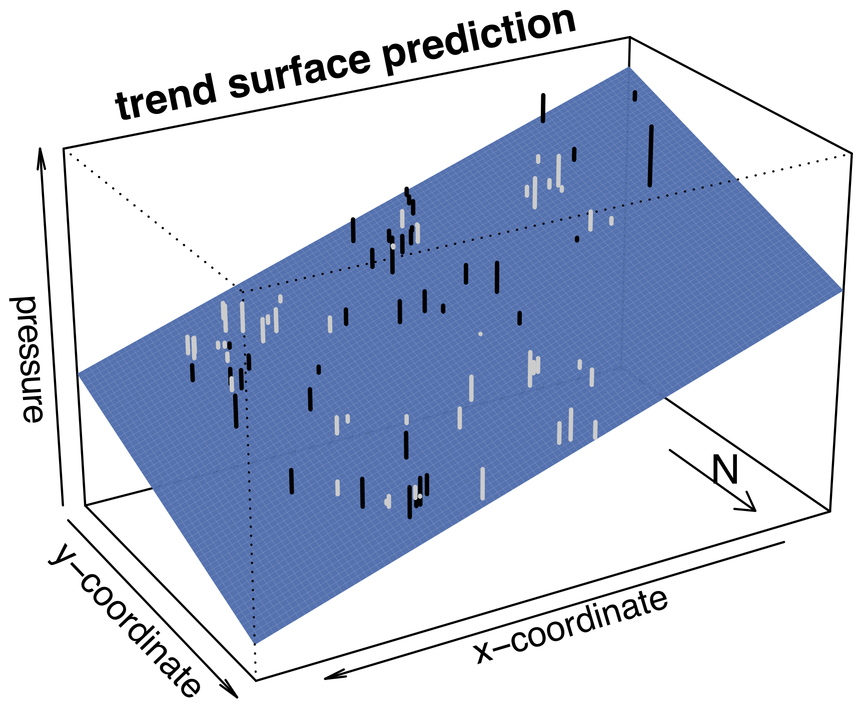
geostatistics in a nutshell

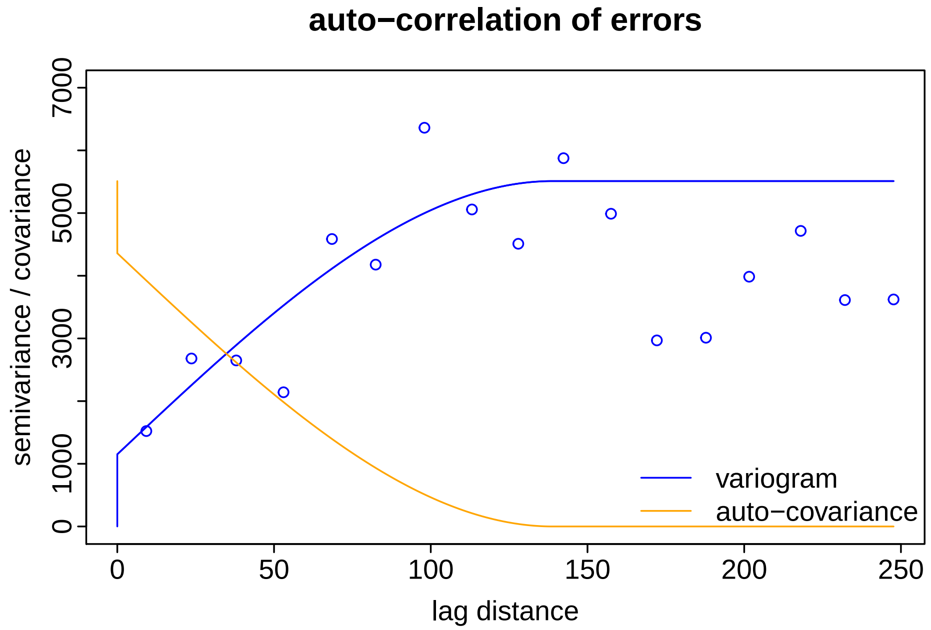
- methology developed in mining
- widely used for spatial interpolation of “point”-support data
- many fields of application: soils, climate, property prices, …

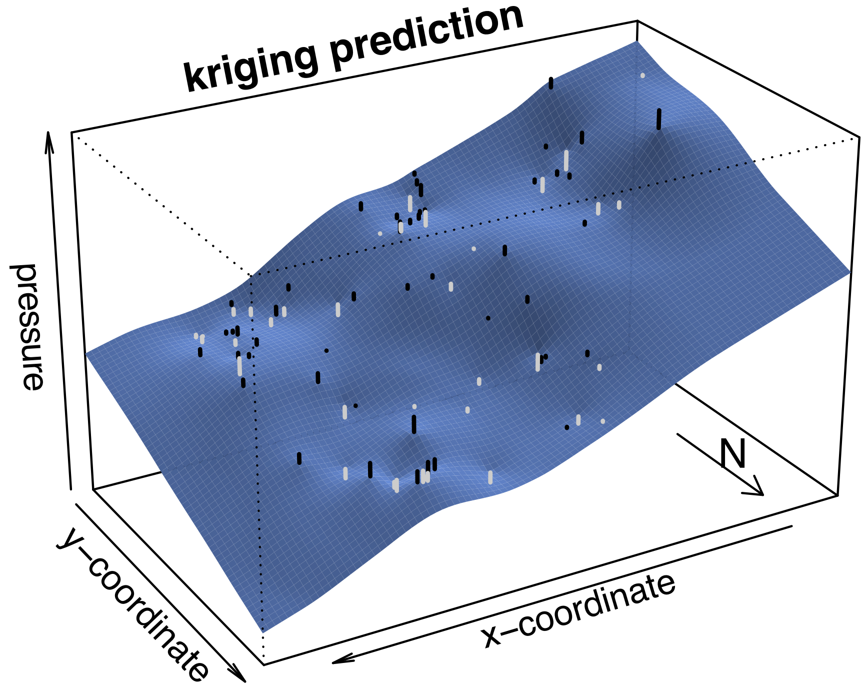
textbook example: Wolfcamp aquifer data



Cressie (1993)


trend surface prediction
pressure
y–coordinate
x–coordinate
N


auto−correlation of errors
semivariance / covariance
0
1000
3000
5000
7000
0
50
100
150
200
250
lag distance
variogram
auto−covariance


kriging prediction
pressure
y–coordinate
x–coordinate
N

geostatistics in a nutshell

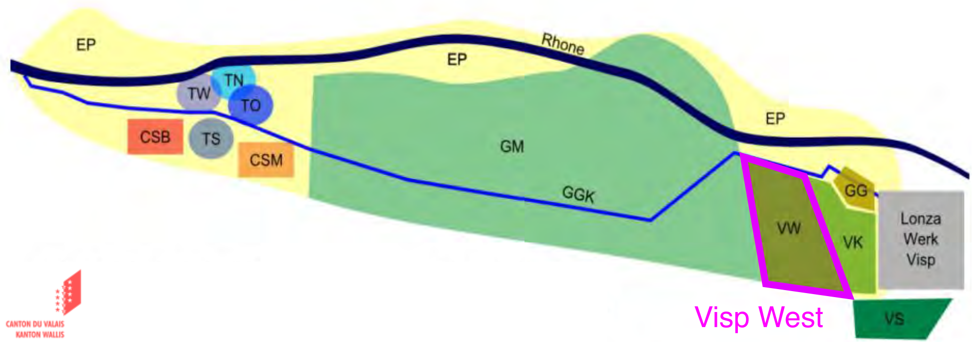
- methology developed in mining
- widely used for spatial interpolation of “point”-data
- many fields of application: soils, climate, property prices, …

$\Rightarrow$ linear mixed modelling approach $\quad \boldsymbol{Y} \sim \mathcal{N}\left(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\right)$

$\Rightarrow$ estimating model parameters by Restricted Maximum Likelihood (REML)

$\Rightarrow$ kriging: empirical (= “plug-in”) best linear unbiased prediction (eBLUP)

case study: topsoil Hg content Visp West



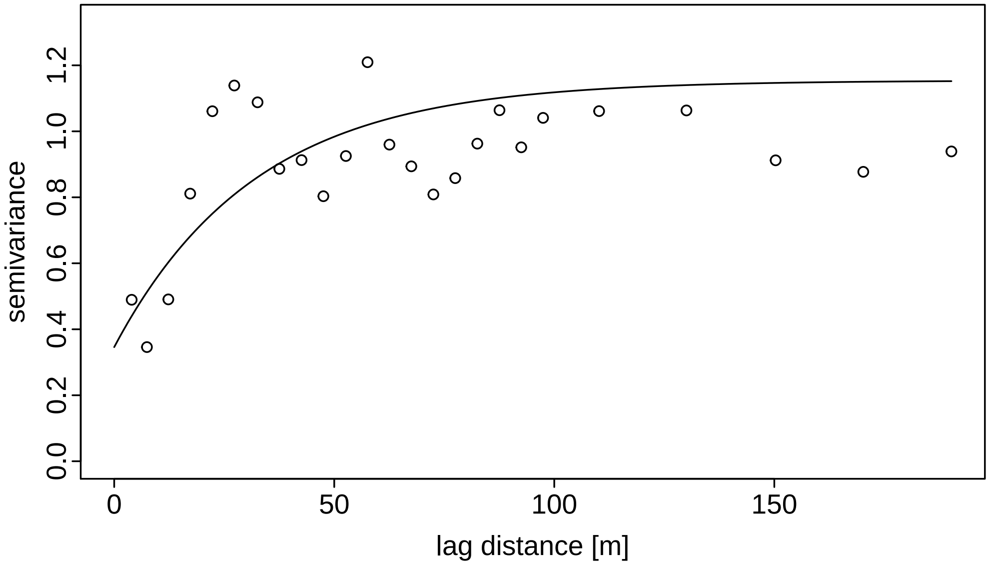
⇒ testing hypotheses about causes for observed spatial patterns

robust REML fit using *georob* package

```
> library(georob)
>
> system.time(fit.0 <- georob(
+   log(hg) ~ logdist.to.ggk + close.to.road + close.to.aqueduct,
+   data = d.vw, locations = ~x+y, variogram.model = "RMexp",
+   param = c(variance = 0.5, nugget = 0.3, scale = 30),
+   tuning.psi = 1))

   user  system elapsed
 11.146   0.927  12.336
```

```
> summary(fit.0)

...

Robust REML estimates

Variogram:  RMexp
               Estimate
variance          0.808
snugget(fixed)    0.000
nugget            0.346
scale            32.140


Fixed effects coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           2.90931    1.15902   2.510 0.012418
logdist.to.ggk       -1.30615    0.34372  -3.800 0.000165
close.to.roadyes      0.33852    0.18578   1.822 0.069092
close.to.aqueductyes  0.02102    0.33463   0.063 0.949947

...
```

```
> plot(fit.0, estimator = "qn", xlab = "lag distance [m]",
+   lag.dist.def= c(seq(0, 100, by = 5), seq(120, 200, by = 20)))
```



testing hypotheses about Hg sources

```
> waldtest(fit.0,
+   . ~ . - close.to.road - close.to.aqueduct,
+   . ~ . - close.to.road - close.to.aqueduct - logdist.to.ggk)
```

```
Wald test

Model 1: log(hg) ~ logdist.to.ggk + close.to.road + close.to.aqueduct
Model 2: log(hg) ~ logdist.to.ggk
Model 3: log(hg) ~ 1
  Res.Df Df       F    Pr(>F)
1    450
2    452 -2  1.6604    0.1912
3    453 -1 19.1807 1.479e-05
```

10-fold cross-validation

```
> fit.1 <- update(fit.0, . ~ . - close.to.road - close.to.aqueduct)
> system.time(cv.1 <- cv(fit.1, sets=d.vw$cv.subset.block.1, lgn=TRUE))

   user  system elapsed
 63.912   7.143  71.764
```

```
> fit.2 <- update(fit.0, . ~ 1)
> cv.2 <- cv(fit.2, sets=d.vw$cv.subset.block.1, lgn=TRUE)
```

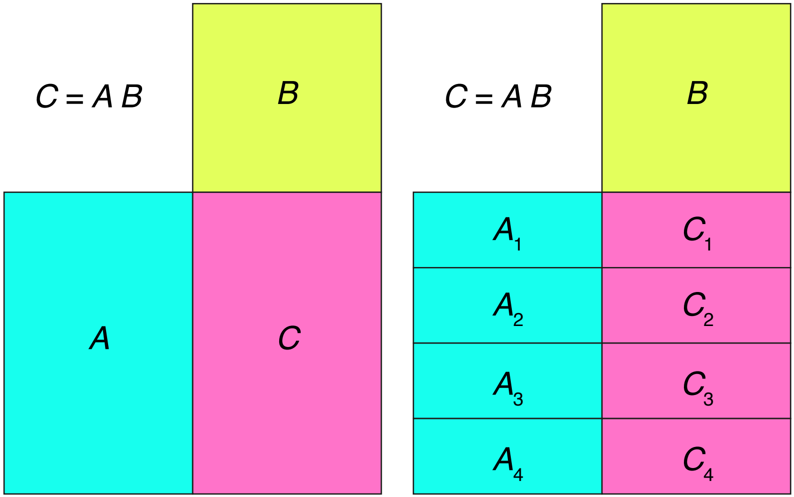
```
> summary(cv.1)

Statistics of back-transformed cross-validation prediction errors
     me     mede     rmse     made
 0.3176  -0.1258   2.2103   0.3053   ...
```

```
> summary(cv.2)

Statistics of back-transformed cross-validation prediction errors
     me     mede     rmse     made
 0.3323  -0.1515   2.2122   0.3368   ...
```

“embarrassingly” parallel tasks

- multiple tasks that do not depend on each other
- examples: cross-valdiation, bootstrapping,…

parallelized matrix multiplication



“embarrassingly” parallel tasks

- multiple tasks that do not depend on each other
- examples: cross-valdiation, bootstrapping,…
- examples: matrix multiplication, …

$\Rightarrow$ parts of a bigger task that can be processed independently from one another

“embarrassingly” parallel computation with R

- useful packages: *parallel*, *snowfall*, …
- starting point: `lapply`-construct

```
> lapply(
+   list(task_1, task_2, ...),
+   function(task_i, x){
+     R-statements_to_solve_task_i
+   },
+   x = common_argument
+ )
```

“embarrassingly” parallel computation with R

- mini-example (Gordon, 2015)

```
> lapply(
+   1:4,
+   function(exponent, x){ x^exponent },
+   x = 2)
[[1]]
[1] 2

[[2]]
[1] 4

[[3]]
[1] 8

[[4]]
[1] 16


> x <- 2
> unlist(lapply(1:4,  function(exponent){ x^exponent }))
[1]  2  4  8 16
```

“embarrassingly” parallel computation with R

- parallelized mini-example using forking (non-Windows OS)

```
> library(parallel)
> unlist(mclapply(1:4, function(exponent){ x^exponent },
+   mc.cores = 2))
[1] 2  4  8 16
```

- parallelized mini-example using parallel socket clusters (all OS)

```
> library(snowfall)
> sfInit(parallel = TRUE, cpus = 2)
> unlist(sfLapply(1:4, function(exponent){ x^exponent }))
ERROR in checkForRemoteErrors(val) :
  2 nodes produced errors; first error: OBJECT 'x' not found

> sfExport("x")    # or: sfExportAll()
> unlist(sfLapply(1:4, function(exponent){ x^exponent }))
[1]  2  4  8 16

> sfStop()
```

parallelized computations in *georob* package

- computing covariance matrices and matrix multiplication
- cross-validation
- computing kriging predictions
- computing likelihood profiles
- stepwise model building

parallelized computations in *georob* package

- model fit using parallelized matrix multiplication

```
> system.time(update(fit.0,
+   control=control.georob(pcmp = control.pcmp(pmm.ncores = 3))))
    user  system elapsed
  24.484   6.172  25.752
  # computation with 1 core:  elapsed 12.336
```

- cross-validation

```
> system.time(cv.1 <- cv(fit.1, sets=d.vw$cv.subset.block.1,
+   lgn=TRUE, ncores = 3))
    user  system elapsed
  57.683   6.008  36.383
  # computation with 1 core:  elapsed 71.764
```

parallelized computations in *georob* package

- computing kriging predictions

```
> system.time(krige.1 <- predict(fit.1, d.grid.vw,
+   control = control.predict.georob(extended.output=TRUE,
      mmax = 1700, ncores = 3)))
   user  system elapsed
  5.812   0.994   3.858
  # computation with 1 core        :  elapsed 5.848
  # computation with 3 cores, PSOCK:  elapsed 7.798
```

- back-transformation and conversion to *SpatialPixelsDataFrame* for display by `spplot()`

```
> krige.1 <- lgnpp(krige.1)
> coordinates(krige.1) <- ~ x+y
> gridded(krige.1) <- TRUE
```

end of story ?

end of story ?

NO!

use R linked to OpenBLAS library

using R with OpenBLAS library on Mac OS X

- install OpenBLAS, e.g. from MacPorts (`https://www.macports.org/`)

- linking OpenBLAS library to R

```
cd /Library/Frameworks/R.framework/Resources/lib
sudo ln -sf /opt/local/lib/libopenblas.dylib libRblas.dylib
```

- enjoy … e.g. Cholesky decomposition of 4000 $\times$ 4000 matrix

```
> y <- rnorm(4000)
> d <- exp(-as.matrix( dist(y)))
> system.time( chol(d) )
   user  system elapsed
  1.466   0.027   0.769
  # using default BLAS library that ships with R:  elapsed 10.327
```

georob computations with R linked to OpenBLAS

- model fit

```
> system.time(update(fit.0))
 user  system elapsed
6.320   1.946   4.318
# standard BLAS:  elapsed 12.336
```

- cross-validation

```
> system.time(cv.1 <- cv(fit.1, sets=d.vw$cv.subset.block.1,
+   lgn=TRUE, ncores = 3))
  user  system elapsed
48.085  21.794  19.556
# standard BLAS & 3 core :  elapsed 36.383
# standard BLAS & 1 core :  elapsed 71.764
```

take-home messages

1. use R along with OpenBLAS, GotoBLAS, Intel's MKL library

   (NB: Microsoft's R Open uses Intels's MKL library, cf. `https://mran.microsoft.com/`)

2. use `parLappy()`, `sfLapply()`, `mclapply()`, etc. instead of `lapply()`

3. DON'T USE `for`-loops for embarrasingly parallel tasks

4. use forking instead of parallel socket clusters on non-Windows OS

References

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley & Sons, New York, revised edition.

Gordon, M. (2015). How to go parallel in R — basics + tips. G-FORGE A blog about orthopaedic surgery, R, research and more. `http://gforge.se/2015/02/how-to-go-parallel-in-r-basics-tips/` (accessed 2018-05-11).

Knaus, J. (2015). *snowfall: Easier cluster computing (based on snow)*.. R package version 1.84-6.1.

Papritz, A. (2018). *georob: Robust Geostatistical Analysis of Spatial Data*. R package version 0.3-6.