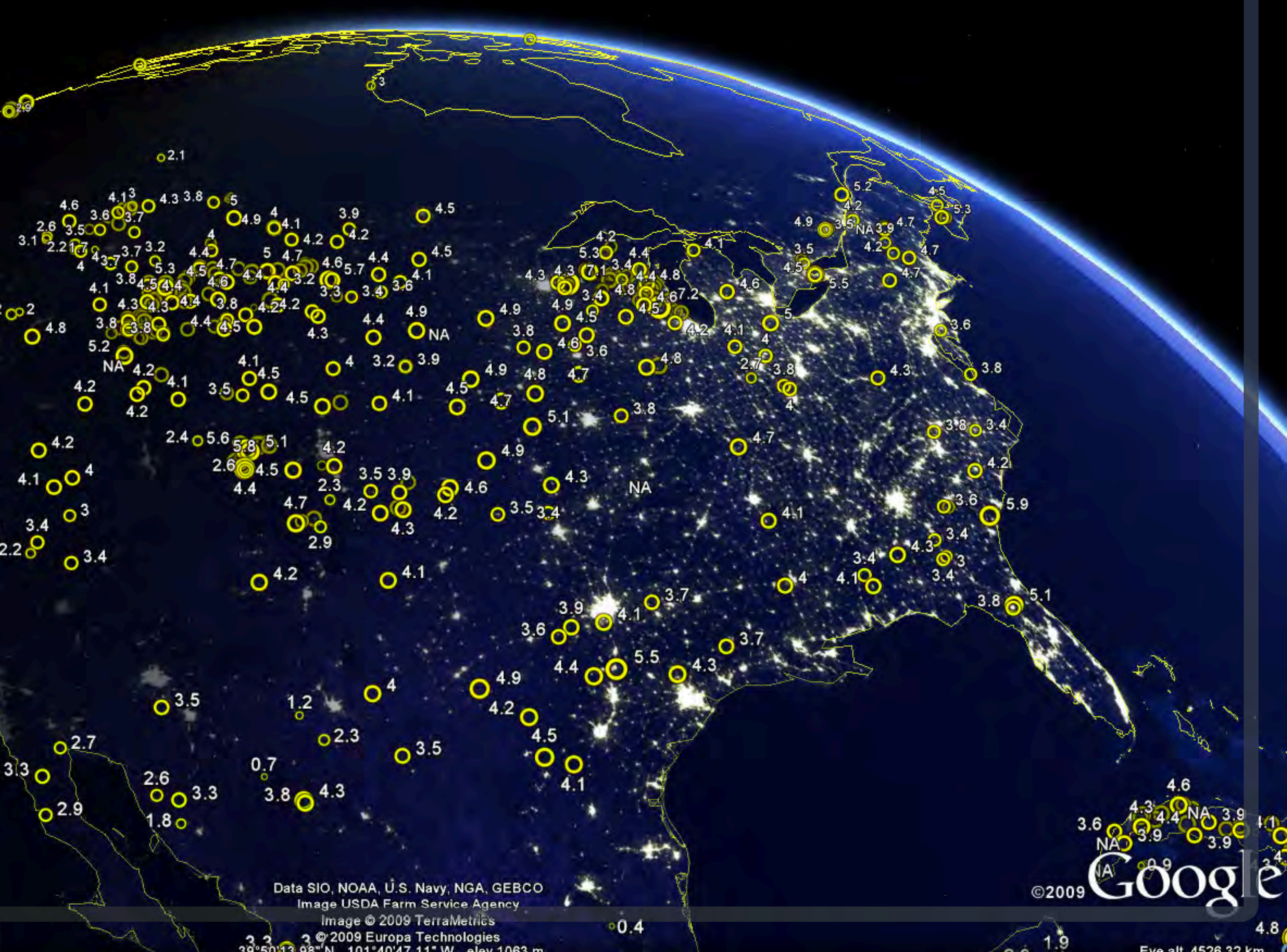
A Practical Guide to Geostatistical Mapping

Tomislav Hengl


Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image USDA Farm Service Agency
Image © 2009 TerraMetrics
© 2009 Europa Technologies
©2009 Google

The methods used in this book were developed in the context of the EcoGRID and LifeWatch projects. EcoGrid (analysis and visualization tools for the Dutch Flora and Fauna database) is a national project managed by the Dutch data authority on Nature (*Gegevensautoriteit Natuur*) and financed by the Dutch ministry of Agriculture (LNV). LifeWatch (e-Science and Technology Infrastructure for Biodiversity Research) is a European Strategy Forum on Research Infrastructures (ESFRI) project, and is partially funded by the European Commission within the 7th Framework Programme under number 211372. The role of the ESFRI is to support a coherent approach to policy-making on research infrastructures in Europe, and to act as an incubator for international negotiations about concrete initiatives.

This is the second, extended edition of the EUR 22904 EN Scientific and Technical Research series report published by Office for Official Publications of the European Communities, Luxembourg (ISBN: 978-92-79-06904-8).

Legal Notice:

Neither the University of Amsterdam nor any person acting on behalf of the University of Amsterdam is responsible for the use which might be made of this publication.

Contact information:

Mailing Address: UvA B2.34, Nieuwe Achtergracht 166, 1018 WV Amsterdam
Tel.: +31- 020-5257379
Fax: +31- 020-5257451
E-mail: T.Hengl@uva.nl
http://home.medewerker.uva.nl/t.hengl/

ISBN 978-90-9024981-0

This document was prepared using the LaTeX 2$_{\varepsilon}$ software.
Printed copies of this book can be ordered via http://www.lulu.com

“I wandered through `http://www.r-project.org`*. To state the good I found there, I’ll also say what else I saw.*

Having abandoned the true way, I fell into a deep sleep and awoke in a deep dark wood. I set out to escape the wood, but my path was blocked by a lion. As I fled to lower ground, a figure appeared before me. ‘Have mercy on me, whatever you are,’ I cried, ‘whether shade or living human’.”

Patrick Burns in *“The R inferno”*

A Practical Guide to Geostatistical Mapping

by Tomislav Hengl

November 2009

Contents

Foreword

This guide evolved from the materials that I have gathered over the years, mainly as lecture notes used for the 5–day training course GEOSTAT. This means that, in order to understand the structure of this book, it is important that you understand how the course evolved and how did the students respond to this process. The GEOSTAT training course was originally designed as a three–week block module with a balanced combination of theoretical lessons, hands on software training and self-study exercises. This obviously cannot work for PhD students and university assistants that have limited budgets, and increasingly limited time. What we offered instead is a *concentrated soup* — three weeks programme in a 5–day block, with one month to prepare. Of course, once the course starts, the soup is still *really really salty*. There are just too many things in too short time, so that many *plates* will typically be left unfinished, and a lot of *food* would have to be thrown away. Because the participants of GEOSTAT typically come from diverse backgrounds, you can never make them all happy. You can at least try to make the most people happy.

Speaking of democracy, when we asked our students whether they would like to see more gentle intros with less topics, or more demos, 57% opted for more demos, so I have on purpose put many demos in this book. Can this course be run in a different way, e.g. maybe via some distance education system? 90% of our students said that they prefer in situ training (both for the professional and social sides), as long as it is short, cheap and efficient. A regular combination of guided training and (creative) self-study is possibly the best approach to learning R+GIS+GE tools. Hence a tip to young researchers would be: every once in a while you should try to follow some short training or refresher course, collect enough ideas and materials, and then take your time and complete the self-study exercises. You should then keep notes and make a list of questions about problems you experience, and subscribe to another workshop, and so on.

If you get interested to run similar courses/workshops in the future (and dedicate yourself to the noble goal of converting the *heretics*) here are some tips. My impression over the years (5) is that the best strategy to give training to beginners with R+GIS+GE is to start with demos (*show them the power of the tools*), then take some baby steps (*show them that command line is not as terrible as it seems*), then get into case studies that look similar to what they do (*show them that it can work for their applications*), and then emphasize the most important concepts (*show them what really happens with their data, and what is the message that they should take home*). I also discovered over the years that some of flexibility in the course programme is always beneficial. Typically, we like to keep 40% of the programme open, so that we can reshape the course at the spot (*ask them what do they want to do and learn, and move away irrelevant lessons*). Try also to remember that it is a good practice if you let the participants control the tempo of learning — if necessary takes some steps back and repeat the analysis (*walk with them, not ahead of them*). In other situations, they can be even more *hungry* than you have anticipated, so make sure you also have some *cake* (bonus exercises) in the fridge. These are the main tips. The rest of success lays in preparation, preparation, preparation… and of course in getting the costs down so that selection is based on quality and not on a budget (*get the best students, not the richest!*). If you want to make money of R (software), I think you are doing a wrong thing. Make money from projects and publicity, give the tools and data your produce for free. Especially if you are already paid from public money.

Almost everybody I know has serious difficulties with switching from some statistical package (or any *point-and-click* interface) to R syntax. It's not only the lack of a GUI or the relatively limited explanation of the functions, it is mainly because R asks for critical thinking about problem-solving (as you will soon find out, very frequently you will need to debug the code yourself, extend the existing functionality or even try to contact the creators), and it does require that you largely change your data analysis philosophy. R is also increasingly extensive, evolving at an increasing speed, and this often represents a problem to less professional users of statistics — it immediately becomes difficult to find which package to use, which method, which parameters to set, and what the results mean. Very little of such information comes with the installation of R. One thing is certain, switching to R without any help and without the right strategy can be very frustrating. From my first contact with R and open source GIS (SAGA) in 2002, the '*first encounter*' is not as terrible any more as it use to be. The methods to run spatio-temporal data analysis (STDA) are now more compact, packages are increasingly compatible, there are increasingly more demos and examples of good practice, there are increasingly more guides. Even if many things (code) in this book frighten you, you should be optimistic about the future. I have no doubts that many of you will one day produce similar guides, many will contribute new packages, start new directions, and continue the legacy. I also have no doubts that in 5–10 years we will be exploring space-time variograms, using voxels and animations to visualize space-time; we will be using real-time data collected through sensor networks with millions of measurements streaming to automated (intelligent?) mapping algorithms. And all this in open and/or free academic software.

This foreword is also a place to acknowledge the work other people have done to help me get this guide out. First, I need to thank the creators of methods and tools, specifically: **Roger Bivand** (NHH), **Edzer Pebesma** (University of Münster), **Olaf Conrad** (University of Hamburg), **Paulo J. Ribeiro Jr.** (Unversidade Federal do Paraná), **Adrian Baddeley** (University of Western Australia), **Markus Neteler** (CEALP), **Frank Warmerdam** (independent developer), and all other colleagues involved in the development of the STDA tools/packages used in this book. If they haven't chosen the path of open source, this book would have not existed. Second, I need to thank the colleagues that have joined me (volunteered) in running the GEOSTAT training courses: **Gerard B. M. Heuvelink** (Wageningen University and Research), **David G. Rossiter** (ITC), **Victor Olaya Ferrero** (University of Plasencia), **Alexander Brenning** (University of Waterloo), and **Carlos H. Grohmann** (University of São Paulo). With many colleagues that I have collaborated over the years I have also become good friend. This is not by accident. There is an enormous enthusiasm around the open source spatial data analysis tools. Many of us share similar philosophy — a similar view on science and education — so that there are always so many interesting topics to discuss over a beer in a pub. Third, I need to thank my colleagues at the University of Amsterdam, most importantly **Willem Bouten**, for supporting my research and for allowing me to dedicate some of my working time to deliver products that often fall far outside our project deliverables. Fourth, I need to thank all participants of the GEOSTAT schools (in total about 130 people) for their interest in this course, and for their tolerance and understanding. Their enthusiasm and intelligence is another strong motive to continue GEOSTAT. Fifth, a number of people have commented on the draft version of this book and have helped me improve its content. Herewith I would especially like to thank **Robert MacMillan** (ISRIC) for reading the whole book in detail, **Jeff Grossman** (USGS) for providing the NGS data set and for commenting on the exercise, **Niels Batjes** (ISRIC) for providing the ISRIC WISE data set and for organizing additional information, **Jim Dalling** (University of Illinois) for providing extra information about the Barro Colorado Island plot, **Nick Hamm** (ITC) and **Lourens Veen** (University of Amsterdam) for reading and correcting large parts of the book, **Chris Lloyd** (Queen's University Belfast), **Pierre Roudier** (Australian Centre for Precision Agriculture), Markus Neteler, **Miguel Gil Biraud** (European Space Agency), **Shaofei Chen** (University of Texas at Dallas), **Thomas E. Adams** (NOAA), **Gabor Grothendieck** (GKX Associates Inc.), **Hanry Walshaw**, **Dylan Beaudette** (U.C. Davis) for providing short but critical comments on various parts of the book. I also need to thank people on the R-sig-geo mailing list for solving many questions that I have further adopted in this book (I think I am now on 50:50% with what I get and what I give to R-sig-geo): Roger Bivand, **Barry Rowlingson** (Lancaster University), Dylan Beaudette and many others.

Finally, however naïve this might seem, I think that all geoscientists should be thankful to the Google company for making GIS popular and accessible to everybody (even my mom now knows how to navigate and find things on maps), and especially for giving away KML to general public. The same way I am thankful to the US environmental data sharing policies and organizations such as **USGS**, **NASA** and **NOAA**, for providing free access to environmental and remotely sensed data of highest quality (that I extensively used in this guide). Europe and other continents have still a lot to learn from the North American neighbors.

I have never meant to produce a Bible of STDA. What I really wanted to achieve with this guide is to bridge the gap between the open source GIS and R, and promote regression-kriging for geostatistical mapping. This is what this guide is really about. It should not be used for teaching geostatistics, but as a supplement. Only by following the literature suggested at the end of each chapter, you will start to develop some geostatistical skills. The first book on your list should definitely be Bivand et al. (2008). The second Diggle and Ribeiro Jr (2007), the third Kutner et al. (2004), the fourth Banerjee et al. (2004) and so on. Much of literature on SAGA can be freely downloaded from web; many similar lecture notes on R are also available. And do not forget to register at the R-sig-geo mailing list and start following the evolution of STDA in real time! Because, this is really the place where most of the STDA evolution is happening today.

This book, both the digital and printed versions, are available only in B/W; exclusive the p.147 that needs to be printed in color. To reproduce full color plots and images, please obtain the original scripts and adjust where needed. For readers requiring more detail about the processing steps it is important to know that complete R scripts, together with plots of outputs and interpretation of processing steps, are available from the contact authors' WIKI project. This WIKI is constantly updated and new working articles are regularly added by the author (that then might appear in the next version of this book). Visit also the book's homepage and submit your comments and suggestions, and this book will become even more useful and more practical.

I sincerely hope to keep this book an open access publication. This was a difficult decision, because open access inevitably carries a risk of lower quality and lower confidence in what is said. On the other hand, I have discovered that many commercial companies have become minimalist in the way they manage scientific books — typically their main interest is to set the price high and sell the book in a bulk package so that all costs of printing and marketing are covered even before the book reaches market. Publishing companies do not want to take any risks, and this would not be so bad if I did not also discover that, increasingly, the real editorial work — page-layouting, reviewing, spell-checking etc. — we need to do ourselves anyway. So why give our work to companies that then sell it at price that is highly selective towards the most developed countries? For somebody who is a dedicated public servant, it is hard to see reasons to give knowledge produced using public money, to companies that were not even involved in the production. Hopefully, you will also see the benefits of open access publishing and help me improve this book by sending comments and suggestions. When preparing this book I followed the example of Paul Bolstad, whose excellent 620 pages *tour de* Geoinformation Science is sold for a symbolic $40 via a small publisher. Speaking of whom, I guess my next mission will be to try to convert Paul also to R+SAGA+GE.

Every effort has been made to trace copyright holders of the materials used in this book. The author apologies for any unintentional omissions and would be pleased to add an acknowledgment in future editions.

Tomislav Hengl

Amsterdam (NL), November 2009

Disclaimer

All software used in this guide is free software and comes with **ABSOLUTELY NO WARRANTY**. The information presented herein is for informative purposes only and not to receive any commercial benefits. Under no circumstances shall the author of this Guide be liable for any loss, damage, liability or expense incurred or suffered which is claimed to resulted from use of this Guide, including without limitation, any fault, error, omission, interruption or delay with respect thereto (reliance at User's own risk).

For readers requiring more detail, the complete R scripts used in this exercise together with the data sets and interpretation of data processing steps are available from the books' homepage[1] (hence, avoid copying the code from this PDF!). The R code might not run on your machine, which could be due to various reasons. Most importantly, **the examples in this book refer to the MS Windows operating systems mainly**. There can be quite some difference between MS Windows, Linux and Mac OS X, although the same functionality should be available on both MS Windows and Linux machines. SAGA GIS is not available for Mac OS X, hence you will need to use a PC with a dual boot system to follow these exercises.

You are welcome to redistribute the programm codes and the complete document provided under certain conditions. For more information, read the GNU general public licence[2]. The author of this book takes no responsibility so ever to accept any of the suggestions by the users registered on the book's website. **This book is a self-published document that presents opinions of the author, and not of the community of users registered on the books' website**.

The main idea of this document is to provide practical instructions to produce quality maps using open-source software. The author of this guide wants to make it clear that maps of limited quality can be produced if low quality inputs are used. Even the most sophisticated geostatistical tools will not be able to save the data sets of poor quality. A quality point data set is the one that fulfills the following requirements:

- *It is large enough* — The data set needs to be large enough to allow statistical testing. Typically, it is recommended to avoid using ≪50 points for reliable variogram modeling and ≪10 points per predictor for reliable regression modeling[3].

- *It is representative* — The data set needs to represent the area of interest, both considering the geographical coverage and the diversity of environmental features. In the case that parts of the area or certain environmental features (land cover/use types, geomorphological strata and similar) are misrepresented or completely ignored, they should be masked out or revisited.

- *It is independent* — The samples need to be collected using a non-preferential sampling i.e. using an objective sampling design. Samples are preferential if special preference is given to locations which are easier to visit, or are influenced by any other type of human bias. Preferably, the point locations

[1] `http://spatial-analyst.net/book/`

[2] `http://www.gnu.org/copyleft/gpl.html`

[3] Reliability of a variogram/regression model decreases exponentially as n approaches small numbers.

should be selected using objective sampling designs such as simple random sampling, regular sampling, stratified random sampling or similar.

- *It is produced using a consistent methodology* — The field sampling and laboratory analysis methodology needs to be consistent, i.e. it needs to be based on standardized methods that are described in detail and therefore reproducible. Likewise, the measurements need to consistently report applicable support size and time reference.

- *Its precision is significantly precise* — Measurements of the environmental variables need to be obtained using field measurements that are significantly more precise than the natural variation.

Geostatistical mapping using inconsistent point samples (either inconsistent sampling methodology, inconsistent support size or inconsistent sampling designs), small data sets, or subjectively selected samples is also possible, but it can lead to many headaches — both during estimation of the spatial prediction models and during interpretation of the final maps. In addition, analysis of such data can lead to unreliable estimates of the model. As a rule of thumb, one should consider repetition of a mapping project if the prediction error of the output maps exceeds the total variance of the target variables in ≥50% of the study area.

Frequently Asked Questions

■ Geostatistics

(1.) **What is an experimental variogram and what does it shows?**

An experimental variogram is a plot showing how one half the squared differences between the sampled values (semivariance) changes with the distance between the point-pairs. We typically expect to see smaller semivariances at shorter distances and then a stable semivariance (equal to global variance) at longer distances. See also §1.3.1 and Fig. 1.9.

(2.) **How to include anisotropy in a variogram?**

By adding two additional parameters — angle of the principal direction (strongest correlation) and the anisotropy ratio. You do not need to fit variograms in different directions. In `gstat`, you only have to indicate that there is anisotropy and the software will fit an appropriate model. See also Fig. 1.11.

(3.) **How do I set an initial variogram?**

One possibility is to use: nugget parameter = measurement error, sill parameter = sampled variance, and range parameter = 10% of the spatial extent of the data (or two times the mean distance to the nearest neighbor). This is only an empirical formula. See also §5.3.2.

(4.) **What is stationarity and should I worry about it?**

Stationarity is an assumed property of a variable. It implies that the target variable has similar statistical properties (mean value, auto-correlation structure) within the whole area of interest. There is the first-order stationarity or the stationarity of the mean value and the second-order stationarity or the covariance stationarity. Mean and covariance stationarity and a normal distribution of values are the requirements for ordinary kriging. In the case of regression-kriging, the target variable does not have to be stationary, but only its residuals. Of course, if the target variable is non-stationary, predictions using ordinary kriging might lead to significant under/over-estimation of values. Read more about stationarity assumptions in section 2.1.1.

(5.) **Is spline interpolation different from kriging?**

In principle, splines and kriging are very similar techniques. Especially regularized splines with tension and universal kriging will yield very similar results. The biggest difference is that the splines require that a user sets the smoothing parameter, while in the case of kriging the smoothing is determined objectively. See also §1.2.3.

(6.) **How do I determine a suitable grid size for output maps?**

The grid size of the output maps needs to match the sampling density and scale at which the processes of interest occur. We can always try to produce maps by using the most detailed grid size that our predictors allow us. Then, we can slowly test how the prediction accuracy changes with coarser grid sizes and finally select a grid size that allows maximum detail, while being computationally effective. See also §10.6.1 and Fig. 10.10.

(7.) **What is logit transformation and why should I use it?**

Logit transformation converts the values bounded by two physical limits (e.g. min=0, max=100%) to $[-\infty, +\infty]$ range. It is a requirement for regression analysis if the values of the target variable are bounded with physical limits (binary values e.g. 0–100%, 0–1 values etc.). See also §5.4.1.

(8.) **Why derive principal components of predictors (maps) instead of using the original predictors?**

Principal Component Analysis is an useful technique to reduce overlap of information in the predictors. If combined with step-wise regression, it will typically help us determine the smallest possible subset of significant predictors. See also §8.2.4 and Fig. 8.6.

(9.) **How can I evaluate the quality of my sampling plan?**

For each existing point sample you can: evaluate clustering of the points by comparing the sampling plan with a random design, evaluate the spreading of points in both geographical and feature space (histogram comparison), evaluate consistency of the sampling intensity. The analysis steps are explained in §5.2.1.

(10.) **How do I test if two prediction methods are significantly different?**

You can derive *RMSE* at validation points for both techniques and then test the difference between the distributions using the two sample t-test (assuming that variable is normally distributed). See also §5.3.3.

(11.) **How should I allocate additional observations to improve the precision of a map produced using geostatistical interpolation?**

You can use the package spatstat and then run weighted point pattern randomization with the map of the normalized prediction variance as the weight map. This will produce a random design with the inspection density proportional to the value of the standardized prediction error. In the next iteration, precision of your map will gradually improve. A more analytical approach is to use Spatial Simulated Annealing as implemented in the intamapInteractive package (see also Fig. 2.13).

- Regression-kriging?

(1.) **What is the difference between regression-kriging, universal kriging and kriging with external drift?**

In theory, all three names describe the same technique. In practice, there are some computational differences: in the case of regression-kriging, the deterministic (regression) and stochastic (kriging) predictions are done separately; in the case of kriging with external drift, both components are predicted simultaneously; the term universal kriging is often reserved for the case when the deterministic part is modeled as a function of coordinates. See also §2.1.4.

(2.) **Can I interpolate categorical variables using regression-kriging?**

A categorical variable can be treated by using logistic regression (i.e. multinomial logistic regression if there are more categories). The residuals can then be interpolated using ordinary kriging and added back to the deterministic component. In the case of fuzzy classes, memberships $\mu \in (0,1)$ can be directly converted to logits and then treated as continuous variables. See also §2.3 and Figs. 5.11 and 9.6.

(3.) **How can I produce geostatistical simulations using a regression-kriging model?**

Both gstat and geoR packages allow users to generate multiple Sequential Gaussian Simulations using a regression-kriging model. However, this can be computationally demanding for large data sets; especially with geoR. See also §2.4 and Fig. 1.4.

(4.) **How can I run regression-kriging on spatio-temporal point/raster data?**

You can extend the 2D space with a time dimension if you simply treat it as the 3rd space–dimension (so called *geometric* space-time model). Then you can also fit 3D variograms and run regression models where observations are available in different time '*positions*'. Usually, the biggest problem of spatio-temporal regression-kriging is to ensure enough ($\gg$10) observations in time-domain. You also need to provide a time-series of predictors (e.g. time-series of remotely sensed images) for the same periods of interest. Spatio-temporal regression-kriging is demonstrated in §11.

(5.) **Can co-kriging be combined with regression-kriging?**

Yes. Additional, more densely sampled covariates, can be used to improve spatial interpolation of the residuals. The interpolated residuals can then be added to the deterministic part of variation. Note that, in order to fit the cross-variograms, covariates need to be also available at sampling locations of the target variable.

(6.) **In which situations might regression-kriging perform poorly?**

Regression-kriging might perform poorly: if the point sample is small and nonrepresentative, if the relation between the target variable and predictors is non-linear, if the points do not represent feature space or represent only the central part of it. See also §2.10.2.

(7.) **Can we really produce quality maps with much less samples than we originally planned (is down-scaling possible with regression-kriging)?**

If correlation with the environmental predictors is strong (e.g. predictors explain >75% of variability), you do not need as many point observations to produce quality maps. In such cases, the issue becomes more how to locate the samples so that the extrapolation in feature space is minimized.

(8.) **Can I automate regression-kriging so that no user-input is needed?**

Automation of regression-kriging is possible in R using the automap package. You can further combine data import, step-wise regression, variogram fitting, spatial prediction (gstat), and completely automate generation of maps. See for example §6.4.

(9.) **How do I deal with heavily skewed data, e.g. a variable with many values close to 0?**

Skewed, non-normal variables can be dealt with using the geoR package, which commonly implements Box-Cox transformation (see §5.5.3). More specific non-Gaussian models (binomial, Poisson) are available in the geoRglm package.

- Software

(1.) **In which software can I run regression-kriging?**

Regression-kriging (implemented using the Kriging with External Drift formulas) can be run in geoR, SAGA and gstat (either within R or using a stand-alone executable file). SAGA has a user-friendly interface to enter the prediction parameters, however, it does not offer possibilities for more extensive statistical analysis (especially variogram modeling is limited). R seems to be the most suitable computing environment for regression-kriging as it permits the largest family of statistical methods and supports data processing automation. See also §3.4.1.

(2.) **Should I use gstat or geoR to run analysis with my data?**

These are not really competitors: you should use both depending on the type of analysis you intend to run — geoR has better functionality for model estimation (variogram fitting), especially if you work with non-Gaussian data, and provides richer output for the model fitting. gstat on the other hand is compatible with sp objects and easier to run and it can process relatively large data sets.

(3.) **Can I run regression-kriging in ArcGIS?**

In principle: No. In ArcGIS, as in ILWIS (see section 3.1.1), it is possible to run separately regression and kriging of residuals and then sum the maps, but neither ArcGIS nor ILWIS support regression-kriging as explained in §2.1. As any other GIS, ArcGIS has limits considering the sophistication of the geostatistical analysis. The statistical functionality of ArcView can be extended using the S-PLUS extension or by using the RPyGeo package (controls the Arc geoprocessor from R).

(4.) **How do I export results of spatial prediction (raster maps) to Google Earth?**

The fastest way to export grids is to use the proj4 module in SAGA GIS — this will automatically estimate the grid cell size and image size in geographic coordinates (see section 5.6.2). Then you can export a raster map as a graphical file (PNG) and generate a Google Earth ground overlay using the maptools package. The alternative is to estimate the grid cell size manually (Eq.3.3.1), then export and resample the values using the akima package or similar (see section 10.6.3).

- Mastering R

(1.) **Why should I invest my time to learn the R language?**

R is, at the moment, the cheapest, the broadest, and the most professional statistical computing environment. In addition, it allows data processing automation, import/export to various platforms, extension of functionality and open exchange of scripts/packages. It now also allows handling and generation of maps. The official motto of an R guru is: *anything is possible with R*!

(2.) **What should I do if I get stuck with R commands?**

Study the R Html help files, study the "*good practice*" examples, browse the R News, purchase books on R, subscribe to the R mailing lists, obtain user-friendly R editors such as Tinn-R or use the package R commander (Rcmdr). The best way to learn R is to look at the existing demos.

(3.) **How do I set the right coordinate system in R?**

By setting the parameters of the CRS argument of a spatial data frame. Visit the European Petroleum Survey Group (EPSG) Geodetic Parameter website, and try to locate the correct CRS parameter by browsing the existing *Coordinate Reference System*. See also p.119.

(4.) **How can I process large data sets ($\gg 10^3$ points, $\gg 10^6$ pixels) in R?**

One option is to split the study area into regular blocks (e.g. 20 blocks) and then produce predictions separately for each block, but using the global model. You can also try installing/using some of the R packages developed to handle large data sets. See also p.94.

(5.) **Where can I get training in R?**

You should definitively consider attending some Use R conference that typically host many half-day tutorial sessions and/or the Bioconductor workshops. Regular courses are organized by, for example, the Aarhus University, The GeoDa Center for Geospatial Analysis at the Arizona State University; in UK a network for Academy for PhD training in statistics often organizes block courses in R. David Rossiter (ITC) runs a distance education course on geostatistics using R that aims at those with relatively limited funds. The author of this book periodically organizes (1–2 times a year) a 5-day workshop for PhD students called **GEOSTAT**. To receive invitations and announcements subscribe to the R-sig-geo mailing list or visit the spatial-analyst.net WIKI (look under "Training in R").

1

Geostatistical mapping

1.1 Basic concepts

Any measurement we take in Earth and environmental sciences, although this is often ignored, has a spatio-temporal reference. A spatio-temporal reference is determined by (at least) four parameters:

(1.) *geographic location* (longitude and latitude or projected X, Y coordinates);

(2.) *height above the ground surface* (elevation);

(3.) *time of measurement* (year, month, day, hour, minute etc.);

(4.) *spatio-temporal support* (size of the blocks of material associated with measurements; time interval of measurement);

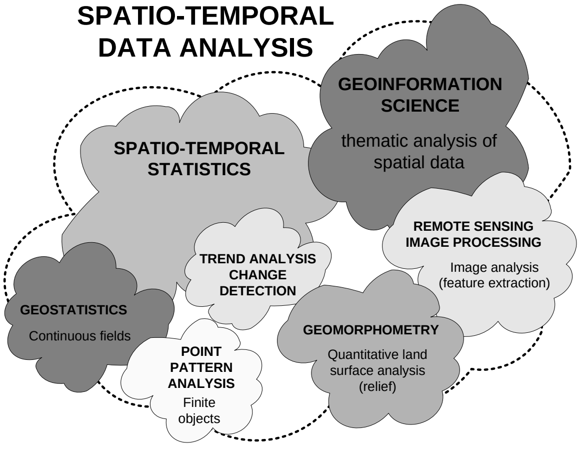

SPATIO-TEMPORAL DATA ANALYSIS
GEOINFORMATION SCIENCE
thematic analysis of spatial data
SPATIO-TEMPORAL STATISTICS
REMOTE SENSING IMAGE PROCESSING
Image analysis (feature extraction)
TREND ANALYSIS CHANGE DETECTION
GEOSTATISTICS
Continuous fields
GEOMORPHOMETRY
Quantitative land surface analysis (relief)
POINT PATTERN ANALYSIS
Finite objects

Fig. 1.1: Spatio-temporal Data Analysis is a group of research fields and sub-fields.

If at least geographical coordinates are assigned to the measurements, then we can analyze and visualize them using a set of specialized techniques. A general name for a group of sciences that provide methodological solutions for the analysis of spatially (and temporally) referenced measurements is **Spatio-temporal Data Analysis** (Fig. 1.1). Image processing techniques are used to analyze remotely sensed data; point pattern analysis is used to analyze discrete point and/or line objects; geostatistics is used to analyze continuous spatial features (fields); geomorphometry is a field of science specialized in the quantitative analysis of topography. We can roughly say that spatio-temporal data analysis (**STDA**) is a combination of two major sciences: *geoinformation science* and *spatio-temporal statistics*; or in mathematical terms: STDA = GIS + statistics. This book focuses mainly on some parts of STDA, although many of the principles we will touch in this guide are common for any type of STDA.

As mentioned previously, **geostatistics** is a subset of statistics specialized in analysis and interpretation of geographically referenced data (Goovaerts, 1997). Cressie (1993) considers geostatistics to be only one of the three scientific fields specialized in the analysis of spatial data — the other two being *point pattern* analysis (focused on point objects; so called *"point-processes"*) and *lattice*[1] statistics (polygon objects) (Fig. 1.2).

[1] The term *lattice* here refers to discrete spatial objects.

For Ripley (2004), spatial statistics is a process of extracting data summaries from spatial data and comparing these to theoretical models that explain how spatial patterns originate and develop. Temporal dimension is starting to play an increasingly important role, so that many principles of spatial statistics (hence geostatistics also) will need to be adjusted.

Because geostatistics evolved in the mining industry, for a long time it meant statistics applied to geology. Since then, geostatistical techniques have successfully found application in numerous fields ranging from soil mapping, meteorology, ecology, oceanography, geochemistry, epidemiology, human geography, geomorphometry and similar. Contemporary geostatistics can therefore best be defined as a **branch of statistics that specializes in the analysis and interpretation of any spatially (and temporally) referenced data, but with a focus on inherently continuous features (spatial fields)**. The analysis of spatio-temporally referenced data is certainly different from what you have studied so far within other fields of statistics, but there are also many direct links as we will see later in §2.1.

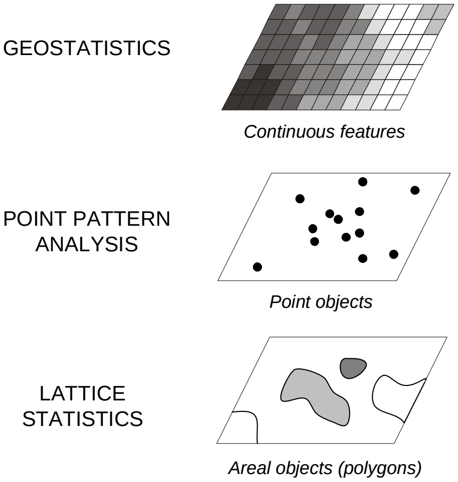


Fig. 1.2: Spatial statistics and its three major subfields after Cressie (1993).

Typical questions of interest to a geostatistician are:

- *How does a variable vary in space-time?*
- *What controls its variation in space-time?*
- *Where to locate samples to describe its spatial variability?*
- *How many samples are needed to represent its spatial variability?*
- *What is a value of a variable at some new location/time?*
- *What is the uncertainty of the estimated values?*

In the most pragmatic terms, geostatistics is an analytical tool for statistical analysis of sampled field data (Bolstad, 2008). Today, geostatistics is not only used to analyze point data, but also increasingly in combination with various GIS data sources: e.g. to explore spatial variation in remotely sensed data, to quantify noise in the images and for their filtering (e.g. filling of the voids/missing pixels), to improve DEM generation and for simulations (Kyriakidis et al., 1999; Hengl et al., 2008), to optimize spatial sampling (Brus and Heuvelink, 2007), selection of spatial resolution for image data and selection of support size for ground data (Atkinson and Quattrochi, 2000).

According to the bibliographic research of Zhou et al. (2007) and Hengl et al. (2009a), the top 10 application fields of geostatistics are: (1) geosciences, (2) water resources, (3) environmental sciences, (4) agriculture and/or soil sciences, (5/6) mathematics and statistics, (7) ecology, (8) civil engineering, (9) petroleum engineering and (10) meteorology. The most influential (highest citation rate) books in the field are: Cressie (1993), Isaaks and Srivastava (1989), Deutsch and Journel (1998), Goovaerts (1997), and more recently Banerjee et al. (2004). These lists could be extended and they differ from country to country of course. The evolution of applications of geostatistics can also be followed through the activities of the following research groups: International Association of Mathematical Geosciences[2] (IAMG), geoENVia[3], pedometrics[4], R-sig-geo[5], spatial accuracy[6] and similar. The largest international conference that gathers geostatisticians is the GEOSTATS conference, and is held every four years; other meetings dominantly focused on the field of geostatistics are GEOENV, STATGIS, and ACCURACY.

[2] http://www.iamg.org
[3] http://geoenvia.org
[4] http://pedometrics.org
[5] http://cran.r-project.org/web/views/Spatial.html
[6] http://spatial-accuracy.org

For Diggle and Ribeiro Jr (2007), there are three scientific objectives of geostatistics:

(1.) **model estimation**, i.e. inference about the model parameters;

(2.) **prediction**, i.e. inference about the unobserved values of the target variable;

(3.) **hypothesis testing**;

Model estimation is the basic analysis step, after which one can focus on prediction and/or hypothesis testing. In most cases all three objectives are interconnected and depend on each other. The difference between hypothesis testing and prediction is that, in the case of hypothesis testing, we typically look for the most reliable statistical technique that provides both a good estimate of the model, and a sound estimate of the associated uncertainty. It is often worth investing extra time to enhance the analysis and get a reliable estimate of probability associated with some important hypothesis, especially if the result affects long-term decision making. The end result of hypothesis testing is commonly a single number (probability) or a binary decision (Accept/Reject). Spatial prediction, on the other hand, is usually computationally intensive, so that sometimes, for pragmatic reasons, naïve approaches are more frequently used to generate outputs; uncertainty associated with spatial predictions is often ignored or overlooked. In other words, in the case of hypothesis testing we are often more interested in the uncertainty associated with some decision or claim; in the case of spatial prediction we are more interested in generating maps (within some feasible time-frame) i.e. exploring spatio-temporal patterns in data. This will become much clearer when we jump from the demo exercise in chapter 5 to a real case study in chapter 6.

Spatial prediction or **spatial interpolation** aims at predicting values of the target variable over the whole area of interest, which typically results in images or maps. Note that there is a small difference between the two because *prediction* can imply both interpolation and extrapolation. We will more commonly use the term *spatial prediction* in this handbook, even though the term *spatial interpolation* has been more widely accepted (Lam, 1983; Mitas and Mitasova, 1999; Dubois and Galmarini, 2004). In geostatistics, e.g. in the case of ordinary kriging, interpolation corresponds to cases where the location being estimated is surrounded by the sampling locations and is within the spatial auto-correlation range. Prediction outside of the practical range (prediction error exceeds the global variance) is then referred to as **extrapolation**. In other words, extrapolation is prediction at locations where we do not have enough statistical evidence to make significant predictions.

An important distinction between geostatistical and conventional mapping of environmental variables is that geostatistical prediction is based on application of quantitative, statistical techniques. Until recently, maps of environmental variables have been primarily been generated by using mental models (expert systems). Unlike the traditional approaches to mapping, which rely on the use of empirical knowledge, in the case of **geostatistical mapping** we completely rely on the actual measurements and semi-automated algorithms. Although this sounds as if the spatial prediction is done purely by a computer program, the analysts have many options to choose whether to use linear or non-linear models, whether to consider spatial position or not, whether to transform or use the original data, whether to consider multicolinearity effects or not. So it is also an expert-based system in a way.

In summary, geostatistical mapping can be defined as **analytical production of maps by using field observations, explanatory information, and a computer program that calculates values at locations of interest** (a study area). It typically comprises:

(1.) design of sampling plans and computational workflow,

(2.) field data collection and laboratory analysis,

(3.) model estimation using the sampled point data (calibration),

(4.) model implementation (prediction),

(5.) model (cross-)evaluation using validation data,

(6.) final production and distribution of the output maps[7].

[7] By this I mainly think of on-line databases, i.e. data distribution portals or Web Map Services and similar.

Today, increasingly, the natural resource inventories need to be regularly updated or improved in detail, which means that after step (6), we often need to consider collection of new samples or additional samples that are then used to update an existing GIS layer. In that sense, it is probably more valid to speak about **geostatistical monitoring**.

1.1.1 Environmental variables

Environmental variables are quantitative or descriptive measures of different environmental features. Environmental variables can belong to different domains, ranging from biology (distribution of species and biodiversity measures), soil science (soil properties and types), vegetation science (plant species and communities, land cover types), climatology (climatic variables at surface and beneath/above), to hydrology (water quantities and conditions) and similar (Table 1.1). They are commonly collected through field sampling (supported by remote sensing); field samples are then used to produce maps showing their distribution in an area. Such accurate and up-to-date maps of environmental features represent a crucial input to spatial planning, decision making, land evaluation and/or land degradation assessment. For example, according to Sanchez et al. (2009), the main challenges of our time that require high quality environmental information are: food security, climate change, environmental degradation, water scarcity and threatened biodiversity.

Because field data collection is often the most expensive part of a survey, survey teams typically visit only a limited number of sampling locations and then, based on the sampled data and statistical and/or mental models, infer conditions for the whole area of interest. As a consequence, maps of environmental variables have often been of limited and inconsistent quality and are usually too subjective. Field sampling is gradually being replaced with **remote sensing systems** and **sensor networks**. For example, elevations marked on topographic maps are commonly collected through land survey i.e. by using geodetic instruments. Today, airborne technologies such as LiDAR are used to map large areas with $\gg$1000 times denser sampling densities. Sensor networks consist of distributed sensors that automatically collect and send measurements to a central service (via GSM, WLAN or radio frequency). Examples of such networks are climatological stations, fire monitoring stations, radiological measurement networks and similar.

From a meta-physical perspective, what we are most often mapping in geostatistics are, in fact, **quantities of molecules of a certain kind or quantities of energy**[8]. For example, a measure of soil or water acidity is the pH factor. By definition, pH is a negative exponent of the concentration of the H^+ ions. It is often important to understand the meaning of an environmental variable: for example, in the case of pH, we should know that the quantities are already on a log-scale so that no further transformation of the variable is anticipated (see further §5.4.1). By mapping pH over the whole area of interest, we will produce a continuous map of values of concentration (continuous fields) of H^+ ions.

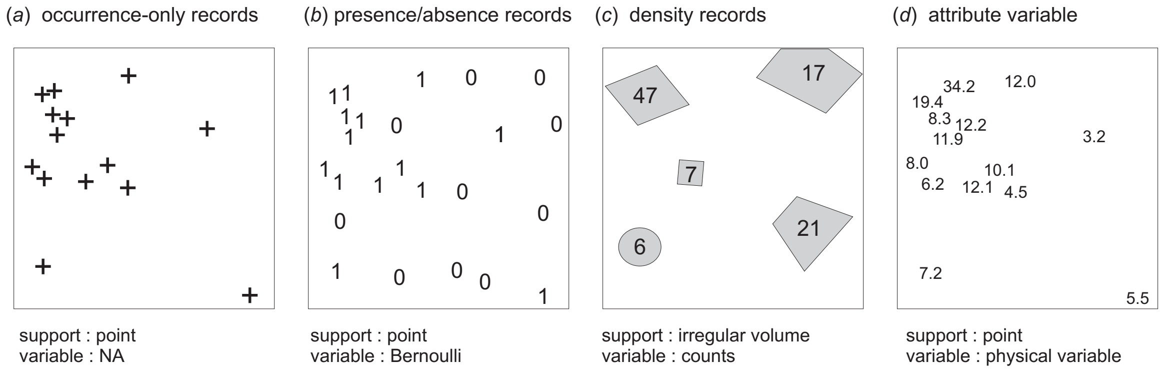


Fig. 1.3: Types of field records in ecology.

In the case of plants and animals inventories, geostatistical mapping is somewhat more complicated. Plants or animals are distinct physical **objects** (individuals), often immeasurable in quantity. In addition, animal

[8] There are few exceptions of course: elevation of land surface, wind speed (kinetic energy) etc.

species change their location dynamically, frequently in unpredictable directions and with unpredictable spatial patterns (non-linear trajectories), which asks for high sampling density in both space and time domains. To account for these problems, spatial modelers rarely aim at mapping distribution of individuals (e.g. represented as points), but instead use compound measures that are suitable for management and decision making purposes. For example, animal species can be represented using density or biomass measures (see e.g. Latimer et al. (2004) and/or Pebesma et al. (2005)).

In vegetation mapping, most commonly field observations of the plant occurrence are recorded in terms of area coverage (from 0 to 100%). In addition to mapping of temporary distribution of species, biologists aim at developing statistical models to define optimal ecological conditions for certain species. This is often referred to as **habitat mapping** or niche modeling (Latimer et al., 2004). Densities, occurrence probability and/or abundance of species or habitat conditions can also be presented as continuous fields, i.e. using raster maps. Field records of plants and animals are more commonly analyzed using point pattern analysis and factor analysis, than by using geostatistics. The type of statistical technique that is applicable to a certain observations data set is mainly controlled by the nature of observations (Fig. 1.3). As we will show later on in §8, with some adjustments, standard geostatistical techniques can also be used to produce maps even from occurrence-only records.

1.1.2 Aspects and sources of spatial variability

Spatial variability of environmental variables is commonly a result of complex processes working at the same time and over long periods of time, rather than an effect of a single realization of a single factor. To explain variation of environmental variables has never been an easy task. Many environmental variables vary not only horizontally but also with depth, not only continuously but also abruptly (Table 1.1). Field observations are, on the other hand, usually very expensive and we are often forced to build 100% complete maps by using a sample of $\ll$1%.

Imagine if we had enough funds to inventory each grid node in a study area, then we would be able to produce a map which would probably look as the map shown in Fig. 1.4[9]. By carefully looking at this map, you can notice several things: (1) there seems to be a spatial pattern of how the values change; (2) values that are closer together are more similar; (3) locally, the values can differ without any systematic rule (randomly); (4) in some parts of the area, the values seem to be in general higher i.e. there is a discrete jump in values.

From the information theory perspective, an environmental variable can be viewed as a *signal process* consisting of three components:

$$Z(\mathbf{s}) = Z^{*}(\mathbf{s}) + \varepsilon'(\mathbf{s}) + \varepsilon'' \qquad (1.1.1)$$

where $Z^{*}(\mathbf{s})$ is the deterministic component, $\varepsilon'(\mathbf{s})$ is the spatially correlated random component and ε'' is the pure noise — partially micro-scale variation, partially the measurement error. This model is, in the literature, often referred to as the **universal model of variation** (see further §2.1). Note that we use a capital letter Z because we assume that the model is probabilistic, i.e. there is a range of equiprobable realizations of the same model $\{Z(\mathbf{s}), \mathbf{s} \in \mathbb{A}\}$; $Z(\mathbf{s})$ indicates that the variable is dependent on the location $\mathbf{s}$.

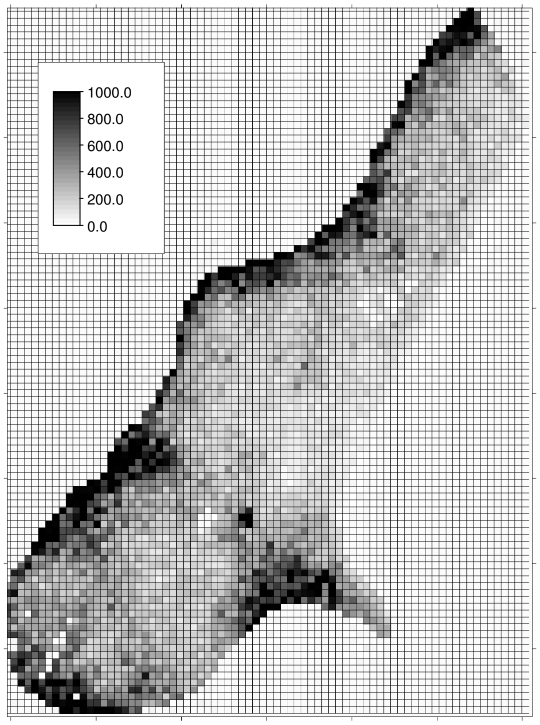


Fig. 1.4: If we were able to sample a variable (e.g. zinc concentration in soil) regularly over the whole area of interest (each grid node), we would probably get an image such as this.

[9] This image was, in fact, produced using geostatistical simulations with a regression-kriging model (see further Fig. 2.1 and Fig. 5.12; §5.5.1).

Table 1.1: Some common environmental variables of interest to decision making and their properties: SRV — short-range variability; TV — temporal variability; VV — vertical variability; SSD — standard sampling density; RSD — remote-sensing detectability. ★ — high, ⋆ — medium, − — low or non-existent. Levels approximated by the author.

Environmental features/topics	**Common variables of interest to decision making**	**SRV**	**TV**	**VV**	**SSD**	**RSD**
Mineral exploration: oil, gas, mineral resources	mineral occurrence and concentrations of minerals; reserves of oil and natural gas; magnetic anomalies;	⋆	−	★	⋆	⋆
Freshwater resources and water quality	O_2, ammonium and phosphorus concentrations in water; concentration of herbicides; trends in concentrations of pollutants; temperature change;	⋆	⋆	⋆	⋆	−
Socio-economic parameters	population density; population growth; GDP per km^2; life expectancy rates; human development index; noise intensity;	⋆	⋆	−	★	★
Health quality data	number of infections; hospital discharge; disease rates per 10,000; mortality rates; health risks;	−	⋆	−	★	−
Land degradation: erosion, landslides, surface runoff	soil loss; erosion risk; quantities of runoff; dissolution rates of various chemicals; landslide susceptibility;	⋆	⋆	−	−	★
Natural hazards: fires, floods, earthquakes, oil spills	burnt areas; fire frequency; water level; earthquake hazard; financial losses; human casualties; wildlife casualties;	★	★	−	⋆	★
Human-induced radioactive contamination	gama doze rates; concentrations of isotopes; PCB levels found in human blood; cancer rates;	⋆	★	−	⋆	★
Soil fertility and productivity	organic matter, nitrogen, phosphorus and potassium in soil; biomass production; (grain) yields; number of cattle per ha; leaf area index;	★	⋆	⋆	⋆	⋆
Soil pollution	concentrations of heavy metals especially: arsenic, cadmium, chromium, copper, mercury, nickel, lead and hexachlorobenzene; soil acidity;	★	⋆	−	★	−
Distribution of animal species (wildlife)	occurrence of species; GPS trajectories (speed); biomass; animal species density; biodiversity indices; habitat conditions;	★	★	−	⋆	−
Distribution of natural vegetation	land cover type; vegetation communities; occurrence of species; biomass; density measures; vegetation indices; species richness; habitat conditions;	⋆	⋆	−	★	★
Meteorological conditions	temperature; rainfall; albedo; cloud fraction; snow cover; radiation fluxes; net radiation; evapotranspiration;	⋆	★	⋆	⋆	★
Climatic conditions and changes	mean, minimum and maximum temperature; monthly rainfall; wind speed and direction; number of clear days; total incoming radiation; trends of changes of climatic variables;	−	★	⋆	⋆	⋆
Global atmospheric conditions	aerosol size; cirrus reflectance; carbon monoxide; total ozone; UV exposure;	⋆	★	★	−	★
Air quality in urban areas	NO_x, SO_2 concentrations; emission of greenhouse gasses; emission of primary and secondary particles; ozone concentrations; Air Quality Index;	★	★	★	★	−
Global and local sea conditions	chlorophyll concentrations; biomass; sea surface temperature; emissions to sea;	⋆	★	⋆	⋆	⋆

In theory, we could decompose a map of a target environmental variable into two grids: (1) the deterministic part (also know as the *trend surface*), and (2) the *error surface*; in practice, we are not able to distinguish the deterministic from the error part of the signal, because both can show similar patterns. In fact, even if we sample every possible part of the study area, we can never be able to reproduce the original signal exactly because of the measurement error. By collecting field measurements at different locations and with different sampling densities, we might be able to infer about the source of variability and estimate probabilistic models of spatial variation. Then we can try to answer how much of the variation is due to the measurement error, how much has been accounted for by the environmental factors, and how much is due to the spatial proximity. Such systematic assessment of the error budget allows us to make realistic interpretations of the results and correctly reason about the variability of the feature of interest.

The first step towards successful geostatistical mapping of environmental variables is to understand the sources of variability in the data. As we have seen previously, the variability is a result of deterministic and stochastic processes plus the pure noise. In other words, the variability in data is a sum of two components: (a) the **natural spatial variation** and (b) the **inherent noise** (ε''), mainly due to the measurement errors (Burrough and McDonnell, 1998). Measurement errors typically occur during positioning in the field, during sampling or laboratory analysis. These errors should ideally be minimized, because they are not of primary concern for a mapper. What the mappers are interested in is the natural spatial variation, which is mainly due to the physical processes that can be explained (up to a certain level) by a mathematical model.

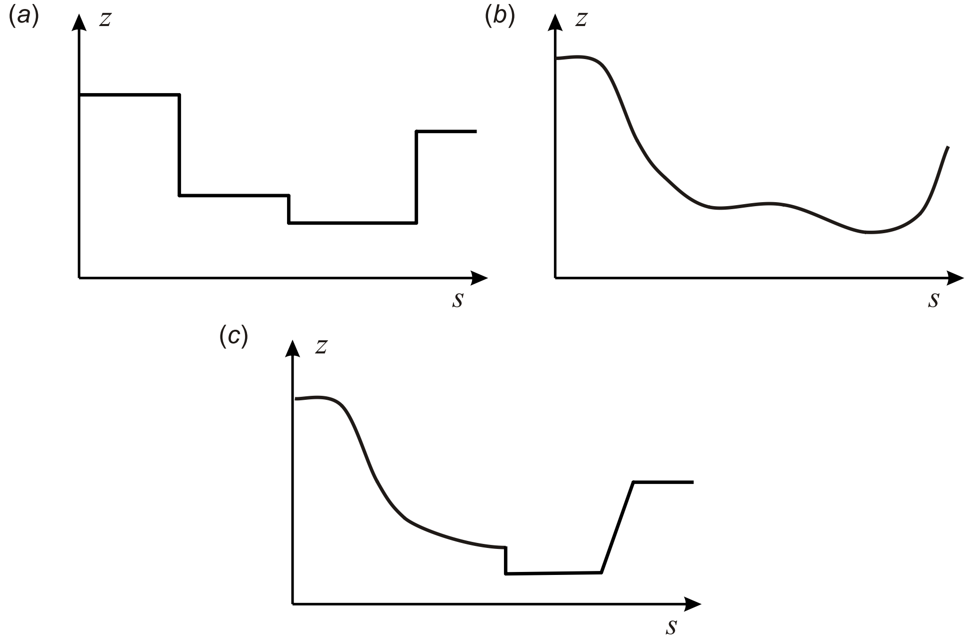


Fig. 1.5: Schematic examples of models of spatial variation: abrupt changes of values can be modeled using a discrete model of spatial variation (a), smooth changes can be modeled using a continuous model of spatial variation (b). In reality, we often need to work with a mixed (or hybrid) model of spatial variation (c).

Physical processes that dominantly control environmental variables differ depending of the type of feature of interest (Table 1.1). In the most general terms, we can say that there are five major factors shaping the status of environment on Earth:

abiotic (global) factors — these include various natural forces that broadly shape the planet. For example, Earth's gravity, rotation cycle, geological composition and tectonic processes etc. Because abiotic factors are relatively constant/systematic and cannot really be controlled, they can be regarded as global fixed conditions.

biotic factors — these include various types of living organism, from microbiological to animal and plant species. Sometimes living organisms can be the major factor shaping environmental conditions, even for wide areas.

anthropogenic factors — these include industrial and agricultural activities, food, water and material consumption, construction of dams, roads and similar. Unfortunately, the human race has irreversibly changed the environment in a short period of time. Extreme examples are the rise in global temperature, loss of biodiversity and deforestation.

transport and diffusion processes — these work upon other abiotic and biotic factors and shape the landscape locally. Unlike global factors, they are often non-linear and highly stochastic.

extra-terrestrial factors — including factors that control climate (e.g. incoming solar radiation, factors that control ice ages etc.), tectonic processes (meteors) and similar.

To illustrate how various factors shape an environmental feature, we can look at land surface (topography) as an example. Land surface is formed, first, as the result of tectonic and volcanic processes. Erosional processes further produce hydrological patterns (river networks, terraces, plains etc.). Living organisms produce soil material and form specific landscapes etc. In some cases extreme events happen such as fall of meteorites, that can suddenly completely change the initial conditions. Again, all these factor work in combination and often with chaotic behavior, so that no simple simulation model of land surface evolution can be constructed. Hence the only way to get an accurate estimate of land surface is to sample.

The second step towards reliable modeling of environmental variables is to consider all aspects of natural variation. Although spatial prediction of environmental variables is primarily concerned with *geographical* variability, there are also other aspects of natural soil variation that are often overlooked by mappers: the *vertical*, *temporal* and *scale* aspects. Below is an overview of the main concepts and problems associated with each of these (see also Table 1.1):

Geographical variation (2D) The results of spatial prediction are either visualised as 2D maps or cross-sections. Some environmental variables, such as thickness of soil horizons, the occurrence of vegetation species or soil types, do not have a third dimension, i.e. they refer to the Earth's surface only. Others, such as temperature, carbon monoxide concentrations etc. can be measured at various altitudes, even below Earth's surface. Geographical part of variation can be modeled using either a **continuous**, **discrete** or **mixed model of spatial variation** (Fig. 1.5).

Vertical variation (3D) Many environmental variables also vary with depth or altitude above the ground surface. In many cases, the measured difference between the values is higher at a depth differing by a few centimeters than at geographical distance of few meters. Consider variables such as temperature or bird density — to explain their vertical distribution can often be more difficult than for the horizontal space. Transition between different soil layers, for example, can also be both gradual and abrupt, which requires a double-mixed model of soil variation for 3D spatial prediction. Some authors suggest the use of cumulative values on volume (areal) basis to simplify mapping of the 3D variables. For example, McKenzie and Ryan (1999) produced maps of total phosphorus and carbon estimated in the upper 1 m of soil and expressed in tons per hectare, which then simplifies production and retrieval. See also further section 7.6.

Temporal variation As mentioned previously, environmental variables connected with distribution of animal and plant species vary not only within a season but also within few moments. Even soil variables such as pH, nutrients, water-saturation levels and water content, can vary over a few years, within a single season or even over a few days (Heuvelink and Webster, 2001). Temporal variability makes geostatistical mapping especially complex and expensive. Maps of environmental variables produced for two different times can differ significantly. Changes can happen abruptly in time. This means that most of the maps are valid for a certain period (or moment) of time only. In many cases the seasonal periodicity of environmental variables is regular, so that we do not necessarily require very dense sampling in time domain (see further §2.5).

Support size Support size is the size or volume associated with measurements, but is also connected with properties such as shape and orientation of areas associated with measurements. Changing the support of a variable creates a different variable which is related to the original, but has different spatial properties (Gotway and Young, 2002). The concept of spatial support should not be confused with the various discretization level of measurements. In the case of spatial predictions, there are two spatial discretization levels: the size of the blocks of land sampled (support size), and grid resolution of the auxiliary maps. Both concepts are closely related with cartographic scale (Hengl, 2006). Field observations are typically collected as point samples. The support size of the auxiliary maps is commonly much larger than the actual blocks of land sampled, e.g. explanatory variables are in general averaged (smoothed), while the environmental variables can describe local (micro) features. As a result, the correlation between the auxiliary maps and measured environmental variables is often low or insignificant (Fig. 1.6). There are two solutions to this problem: (a) to up-scale the auxiliary maps or work with high resolution satellite images, or (b) to average bulk or composite samples within the regular blocks of land (Patil, 2002). The first approach is more attractive for the efficiency of prediction, but at the cost of more processing power and storage. The second solution will only result in a better fit, whereas the efficiency of prediction, validated using point observations, may not change significantly.

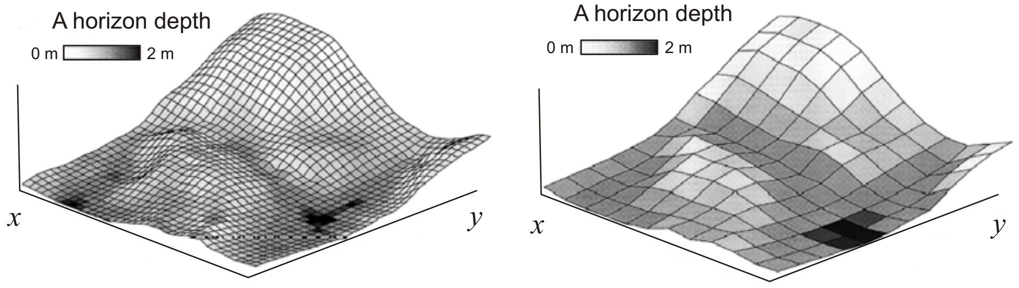


Fig. 1.6: Influence of the support (grid cell) size: predictions of the same variable at coarse grid will often show much less contrast, i.e. it will miss many local hot-spots. Example from Thompson et al. (2001).

In practice, given the space-time domain and feature of interest, one makes measurements by fixing either 2D space, elevation/depth or time. Mixing of lab data from different seasons, depths and with different support sizes in general means lower predictive power and problems in fully interpreting the results. If the focus of prediction modeling is solely the geographical component (2D), then the samples need to be taken under fixed conditions: same season, same depths, same blocks of land. Likewise, if the focus of analysis is generation of spatio-temporal patterns, some minimum of point samples in both space and time domains is needed. Analysts that produce 2D maps often ignore concepts such as temporal variability and support size. To avoid possible misinterpretation, each 2D map of environmental variables generated using geostatistics **should always indicate a time reference (interval), applicable vertical dimension[10], sampling locations, borders of the area of interest, and the size of sampling blocks (support size)**.

1.1.3 Spatial prediction models

In an ideal situation, variability of environmental variables is determined by a finite set of inputs and they exactly follow some known physical law. If the algorithm (formula) is known, the values of the target variables can be predicted exactly. In reality, the relationship between the feature of interest and physical environment is so complex[11] that it cannot be modeled exactly (Heuvelink and Webster, 2001). This is because we either do not exactly know: (a) the final list of inputs into the model, (b) the rules (formulas) required to derive the output from the inputs and (c) the significance of the random component in the system. So the only possibility is that we try to estimate a model by using the actual field measurements of the target variable. This can be referred to as the *indirect* or *non-deterministic* estimation.

Let us first define the problem using mathematical notation. Let a set of observations of a **target variable** (also known as *response* variable) Z be denoted as $z(\mathbf{s}_1)$, $z(\mathbf{s}_2)$,..., $z(\mathbf{s}_n)$, where $\mathbf{s}_i = (x_i, y_i)$ is a location and x_i and y_i are the coordinates (primary locations) in geographical space and n is the number of observations (Fig. 1.7). The geographical domain of interest (area, land surface, object) can be denoted as $\mathbb{A}$. We deal with only one reality (samples $z(\mathbf{s}_n)$), which is a realization of a process ($\mathbf{Z} = \{Z(\mathbf{s}), \forall \mathbf{s} \in \mathbb{A}\}$) that could have produced many realities.

[10] Orthogonal distance from the ground surface.

[11] Because either the factors are unknown, or they are too difficult to measure, or the model itself would be too complex for realistic computations.

Assuming that the samples are *representative*, *non-preferential* and *consistent*, values of the target variable at some new location $\mathbf{s}_0$ can be derived using a **spatial prediction model**. In statistical terms, a spatial prediction model draws realizations — either the most probable or a set of equiprobable realizations — of the feature of interest given a list of inputs:

$$\hat{z}(\mathbf{s}_0) = E\left\{Z|z(\mathbf{s}_i),\ q_k(\mathbf{s}_0),\ \gamma(\mathbf{h}),\ \mathbf{s} \in \mathbb{A}\right\} \tag{1.1.2}$$

where $z(\mathbf{s}_i)$ is the input point data set, $\gamma(\mathbf{h})$ is the covariance model defining the spatial autocorrelation structure (see further Fig. 2.1), and $q_k(\mathbf{s}_0)$ is the list of deterministic predictors, also known as *covariates* or explanatory variables, which need to be available at any location within $\mathbb{A}$. In other words, a spatial prediction model comprises list of procedures to generate predictions of value of interest given the calibration data and spatial domain of interest.

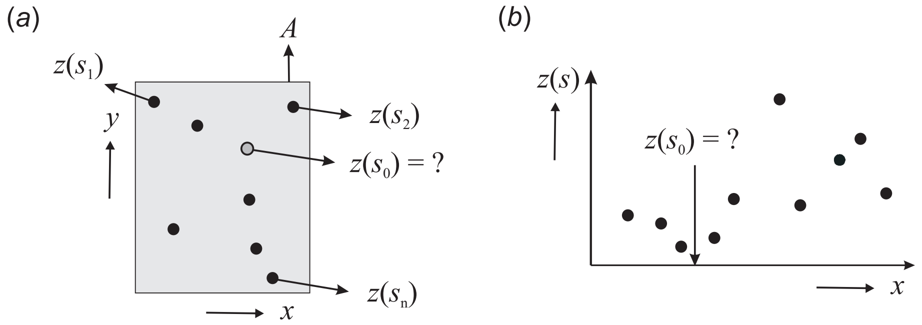


Fig. 1.7: Spatial prediction is a process of estimating the value of (quantitative) properties at unvisited site within the area covered by existing observations: (a) a scheme in horizontal space, (b) values of some target variable in a one-dimensional space.

In raster GIS terms, the geographical domain of interest is a rectangular matrix, i.e. an array with rows×columns number of grid nodes over the domain of interest (Fig. 1.8):

$$\mathbf{z} = \left\{z(\mathbf{s}_j), j = 1, \dots, m\right\}; \quad \mathbf{s}_j \in \mathbb{A} \tag{1.1.3}$$

where $\mathbf{z}$ is the data array, $z(\mathbf{s}_j)$ is the value at the grid node $\mathbf{s}_j$, and m is the total number of grid nodes. Note that there is a difference between predicting values at grid node (punctual) and prediction values of the whole grid cell (block), which has a full topology[12].

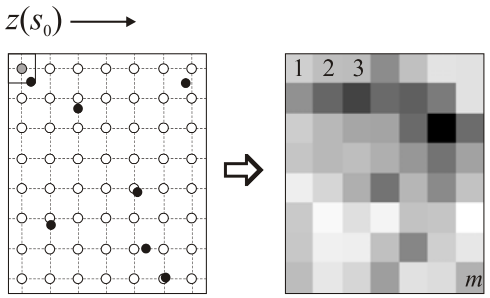


Fig. 1.8: Spatial prediction implies application of a prediction algorithm to an array of grid nodes (*point á point* spatial prediction). The results are then displayed using a raster map.

There seem to be many possibilities to interpolate point samples. At the Spatial Interpolation Comparison 2004 exercise, for example, 31 algorithms competed in predicting values of gamma dose rates at 1008 new locations by using 200 training points (Dubois and Galmarini, 2004; Dubois, 2005). The competitors ranged from splines, to neural networks, to various kriging algorithms. Similarly, the software package Surfer[13] offers dozens of interpolation techniques: Inverse Distance, Kriging, Minimum Curvature, Polynomial Regression, Triangulation, Nearest Neighbor, Shepard's Method, Radial Basis Functions, Natural Neighbor, Moving Average, Local Polynomial, etc. The list of interpolators available in R via its interpolation packages (akima, loess, spatial, gstat, geoR etc.) is even longer.

[12] The sp package in R, for example, makes a distinction between the Spatial Pixel data frame (grid nodes) and a Spatial Grid data frame (grid cells) to distinguish between regular grid with point support and block support.

[13] http://www.ssg-surfer.com

An inexperienced user will often be challenged by the amount of techniques to run spatial interpolation. Li and Heap (2008), for example, list over 40 unique techniques in their extensive review of the spatial prediction methods. Most spatial prediction models are in fact somehow connected. As we will see later on, many standard linear models are in fact just a special case of a more general prediction model. This makes things much less complicated for the non-geostatisticians[14]. It is thus more important to have a clear idea about the connection or hierarchy of predictors, than to be able to list all possible predictors and their variants.

Spatial prediction models (algorithms) can be classified according to the amount of statistical analysis i.e. amount of expert knowledge included in the analysis:

(1.) **MECHANICAL (DETERMINISTIC) MODELS** — These are models where arbitrary or empirical model parameters are used. No estimate of the model error is available and usually no strict assumptions about the variability of a feature exist. The most common techniques that belong to this group are:

- *Thiessen polygons*;
- *Inverse distance interpolation*;
- *Regression on coordinates*;
- *Natural neighbors*;
- *Splines*;
- ...

(2.) **LINEAR STATISTICAL (PROBABILITY) MODELS** — In the case of statistical models, the model parameters are commonly estimated in an objective way, following probability theory. The predictions are accompanied with an estimate of the prediction error. A drawback is that the input data set usually need to satisfy strict statistical assumptions. There are at least four groups of linear statistical models:

- *kriging* (plain geostatistics);
- *environmental correlation* (e.g. regression-based);
- *Bayesian-based models* (e.g. Bayesian Maximum Entropy);
- *hybrid models* (e.g. regression-kriging);
- ...

(3.) **EXPERT-BASED SYSTEMS** — These models can be completely subjective (*ergo* irreproducible) or completely based on data; predictions are typically different for each run. Expert systems can also largely be based on probability theory (especially Bayesian statistics), however, it is good to put them in a different group because they are conceptually different from standard linear statistical techniques. There are at least three groups of expert based systems:

- *mainly knowledge-driven expert system* (e.g. hand-drawn maps);
- *mainly data-driven expert system* (e.g. based on neural networks);
- *machine learning* algorithms (purely data-driven);

Spatial prediction models can also be classified based on the:

Smoothing effect — whether the model smooths predictions at sampling locations or not:

- *Exact* (measured and estimated values coincide);
- *Approximate* (measured and estimated values do not have to coincide);

Transformation of a target variable — whether the target variable is used in its original scale or transformed:

- *Untransformed or Gaussian* (the variable already follows close to a normal distribution);

[14] As we will see later on in §2.1.2, spatial prediction can even be fully automated so that a user needs only to provide quality inputs and the system will select the most suitable technique.

- *Trans-Gaussian* (variable transformed using some link function);

Localization of analysis — whether the model uses all sampling locations or only locations in local proximity:

- *Local* or *moving window* analysis (a local sub-sample; local models applicable);
- *Global* (all samples; the same model for the whole area);

Convexity effect — whether the model makes predictions outside the range of the data:

- *Convex* (all predictions are within the range);
- *Non-convex* (some predictions might be outside the range);

Support size — whether the model predicts at points or for blocks of land:

- *Point-based* or punctual prediction models;
- *Area-based* or block prediction models;

Regularity of support — whether the output data structure is a grid or a polygon map:

- *Regular* (gridded outputs);
- *Irregular* (polygon maps);

Quantity of target variables — whether there is one or multiple variables of interest:

- *Univariate* (model is estimated for one target variable at a time);
- *Multivariate* (model is estimated for multiple variables at the same time);

Another way to look at spatial prediction models is to consider their ability to represent models of spatial variation. Ideally, we wish to use a mixed model of spatial variation (Fig. 1.5c) because it is a generalization of the two models and can be more universally applied. In practice, many spatial prediction models are limited to one of the two models of spatial variation: predicting using polygon maps (§1.3.3) will show discrete changes (Fig. 1.5a) in values; ordinary kriging (§1.3.1) will typically lead to smooth maps (Fig. 1.5b).

1.2 Mechanical spatial prediction models

As mentioned previously, mechanical spatial prediction models can be very flexible and easy to use. They can be considered to be subjective or empirical, because the user him/her-self selects the parameters of the model, often without any deeper analysis, often based only on a visual evaluation — the '*look good*' assessment. Most commonly, a user typically accepts the default parameters suggested by some software, hence the name *mechanical* models. The most widely used mechanical spatial prediction models are Thiessen polygons, inverse distance interpolation, regression on coordinates and various types of splines (Lam, 1983; Myers, 1994; Mitas and Mitasova, 1999). In general, mechanical prediction models are more primitive than statistical models and are often sub-optimal. However, in some situations they can perform as well as statistical models (or better).

1.2.1 Inverse distance interpolation

Probably one of the oldest spatial prediction techniques is **inverse distance interpolation** (Shepard, 1968). As with many other spatial predictors, in the case of inverse distance interpolation, a value of target variable at some new location can be derived as a weighted average:

$$\hat{z}(\mathbf{s}_0) = \sum_{i=1}^{n} \lambda_i(\mathbf{s}_0) \cdot z(\mathbf{s}_i) \tag{1.2.1}$$

where λ_i is the weight for neighbor i. The sum of weights needs to equal one to ensure an unbiased interpolator. Eq.(1.2.1) in matrix form is:

$$\hat{z}(\mathbf{s}_0) = \boldsymbol{\lambda}_\mathbf{0}^{\mathbf{T}} \cdot \mathbf{z} \tag{1.2.2}$$

The simplest approach for determining the weights is to use the **inverse distances** from all points to the new point:

$$\lambda_i(\mathbf{s}_0) = \frac{\frac{1}{d^{\beta}(\mathbf{s}_0,\mathbf{s}_i)}}{\sum\limits_{i=0}^{n} \frac{1}{d^{\beta}(\mathbf{s}_0,\mathbf{s}_i)}}; \qquad \beta > 1 \tag{1.2.3}$$

where $d(\mathbf{s}_0, \mathbf{s}_i)$ is the distance from the new point to a known sampled point and β is a coefficient that is used to adjust the weights. The principle of using inverse distances is largely a reflection of Waldo Tobler's first law in geography which states that "*Everything is related to everything else, but near things are more related than distant things.*" (Tobler, 1970, p.236); hence, points which are close to an output pixel will obtain large weights and that points which are farther away from an output pixel will obtain small weights. The β parameter is used to *emphasize* spatial similarity. If we increase β less importance will be put on distant points. The remaining problem is how to estimate β objectively so that it reflects the true strength of auto-correlation.

Inverse distance interpolation is an exact, convex interpolation method that fits only the continuous model of spatial variation. For large data sets ($\gg 10^3$ points) it can be time-consuming so it is often a good idea to set a threshold distance (search radius) to speed up the calculations.

1.2.2 Regression on coordinates

Assuming that the values of a target variable at some location are a function of coordinates, we can determine its values by finding a function which passes through (or close to) the given set of discrete points. This group of techniques can be termed *regression on coordinates*, although it is primarily known in literature by names **trend surfaces** and/or **moving surface interpolation**, depending on whether the function is fitted for the whole point data set (trend) or for a local (moving) neighbourhood (Hardy, 1971). Regression on coordinates is based on the following model (Webster and Oliver, 2001, p.40–42):

$$Z(\mathbf{s}) = f(x, y) + \varepsilon \tag{1.2.4}$$

and the predictions are made by:

$$\hat{z}(\mathbf{s}_0) = \sum_{r,s \in n} a_{rs} \cdot x^r y^s = \mathbf{a}^{\mathbf{T}} \cdot \mathbf{s_0} \tag{1.2.5}$$

where $r + s < p$ is the number of transformations of coordinates, p is the order of the surface. The model coefficients ($\mathbf{a}$) are determined by maximizing the local fit:

$$\sum_{i=1}^{n} (\hat{z}_i - z_i)^2 \rightarrow \min \tag{1.2.6}$$

which can be achieved by the **Ordinary Least Squares** solution (Kutner et al., 2004):

$$\mathbf{a} = \left(\mathbf{s}^{\mathbf{T}} \cdot \mathbf{s}\right)^{-1} \cdot \left(\mathbf{s}^{\mathbf{T}} \cdot \mathbf{z}\right) \tag{1.2.7}$$

In practice, local fitting of the moving surface is more widely used to generate maps than trend surface interpolation. In the case of a moving surface, for each output grid node, a polynomial surface is fitted to a larger[15] number of points selected by a moving window (circle). The main problem of this technique is that, by introducing higher order polynomials, we can generate many artifacts and cause serious overshooting of the values locally (see further Fig. 1.13). A moving surface will also completely fail to represent discrete changes in space.

[15] The number of points needs to be at least larger than the number of parameters.

Regression on coordinates can be criticized for not relying on empirical knowledge about the variation of a variable (Diggle and Ribeiro Jr, 2007, p.57). As we will see later on in §1.3.2, it is probably advisable to avoid using x, y coordinates and their transforms and instead use **geographic predictors** such as the distance from a coast line, latitude, longitude, distance from water bodies and similar. Similar recommendation also applies to Universal kriging (see p.36) where coordinates are used to explain the deterministic part of variation.

1.2.3 Splines

A special group of interpolation techniques is based on **splines**. A spline is a type of piecewise polynomial, which is preferable to a simple polynomial interpolation because more parameters can be defined including the amount of smoothing. The smoothing spline function also assumes that there is a (measurement) error in the data that needs to be smoothed locally. There are many versions and modifications of spline interpolators. The most widely used techniques are **thin-plate splines** (Hutchinson, 1995) and **regularized spline with tension and smoothing** (Mitasova and Mitas, 1993).

In the case of regularized spline with tension and smoothing (implemented e.g. in GRASS GIS), the predictions are obtained by (Mitasova et al., 2005):

$$\hat{z}(\mathbf{s}_0) = a_1 + \sum_{i=1}^{n} w_i \cdot R(\upsilon_i) \tag{1.2.8}$$

where the a_1 is a constant and $R(\upsilon_i)$ is the radial basis function determined using (Mitasova and Mitas, 1993):

$$\begin{aligned} R(\upsilon_i) &= -\left[E_1(\upsilon_i) + \ln(\upsilon_i) + C_E\right] && (1.2.9)\\ \upsilon_i &= \left[\varphi \cdot \frac{\mathbf{h_0}}{2}\right]^2 && (1.2.10)\end{aligned}$$

where $E_1(\upsilon_i)$ is the exponential integral function, C_E=0.577215 is the Euler constant, φ is the generalized tension parameter and $\mathbf{h_0}$ is the distance between the new and interpolation point. The coefficients a_1 and w_i are obtained by solving the system:

$$\begin{aligned} \sum_{i=1}^{n} w_i &= 0 && (1.2.11)\\ a_1 + \sum_{i=1}^{n} w_i \cdot \left[R(\upsilon_i) + \delta_{ij} \cdot \frac{\varpi_0}{\varpi_i}\right] &= z(\mathbf{s}_i); \qquad j = 1,..,n && (1.2.12)\end{aligned}$$

where ϖ_0/ϖ_i are positive weighting factors representing a smoothing parameter at each given point $\mathbf{s}_i$. The tension parameter φ controls the distance over which the given points influence the resulting surface, while the smoothing parameter controls the vertical deviation of the surface from the points. By using an appropriate combination of tension and smoothing, one can produce a surface which accurately fits the empirical knowledge about the expected variation (Mitasova et al., 2005). Regularized spline with tension and smoothing are, in a way, equivalent to universal kriging (see further §2.1.4) where coordinates are used to explain the deterministic part of variation, and would yield very similar results.

Splines have been widely regarded as highly suitable for interpolation of densely sampled heights and climatic variables (Hutchinson, 1995; Mitas and Mitasova, 1999). However, their biggest criticism is their inability to incorporate larger amounts of auxiliary maps to model the deterministic part of variation. In addition, the smoothing and tension parameters are commonly determined subjectively.

1.3 Statistical spatial prediction models

As mentioned previously, in the case of statistical models, model parameters (coefficients) used to derive outputs are estimated in an objective way following the theory of probability. Unlike mechanical models, in the case of statistical models, we need to follow several statistical data analysis steps before we can generate maps. This makes the whole mapping process more complicated but it eventually helps us: (a) produce more reliable/objective maps, (b) understand the sources of errors in the data and (c) depict problematic areas/points that need to be revisited.

1.3.1 Kriging

Kriging has for many decades been used as a synonym for geostatistical interpolation. It originated in the mining industry in the early 1950's as a means of improving ore reserve estimation. The original idea came from the mining engineers D. G. Krige and the statistician H. S. Sichel. The technique was first[16] published in Krige (1951), but it took almost a decade until a French mathematician G. Matheron derived the formulas and basically established the whole field of linear geostatistics[17] (Cressie, 1990; Webster and Oliver, 2001). Since then, the same technique has been independently discovered many times, and implemented using various approaches (Venables and Ripley, 2002, pp.425–430).

A standard version of kriging is called **ordinary kriging** (**OK**). Here the predictions are based on the model:

$$Z(\mathbf{s}) = \mu + \varepsilon'(\mathbf{s}) \tag{1.3.1}$$

where μ is the constant *stationary* function (global mean) and $\varepsilon'(\mathbf{s})$ is the spatially correlated stochastic part of variation. The predictions are made as in Eq.(1.2.1):

$$\hat{z}_{\texttt{OK}}(\mathbf{s}_0) = \sum_{i=1}^{n} w_i(\mathbf{s}_0) \cdot z(\mathbf{s}_i) = \boldsymbol{\lambda}_\mathbf{0}^{\mathrm{T}} \cdot \mathbf{z} \tag{1.3.2}$$

where $\boldsymbol{\lambda}_\mathbf{0}$ is the vector of kriging weights (w_i), $\mathbf{z}$ is the vector of n observations at primary locations. In a way, kriging can be seen as a sophistication of the inverse distance interpolation. Recall from §1.2.1 that the key problem of inverse distance interpolation is to determine how much importance should be given to each neighbor. Intuitively thinking, there should be a way to estimate the weights in an objective way, so the weights reflect the true spatial autocorrelation structure. The novelty that Matheron (1962) and colleagues introduced to the analysis of point data is the derivation and plotting of the so-called **semivariances** — differences between the neighboring values:

$$\gamma(\mathbf{h}) = \frac{1}{2} E\left[\left(z(\mathbf{s}_i) - z(\mathbf{s}_i + \mathbf{h})\right)^2\right] \tag{1.3.3}$$

where $z(\mathbf{s}_i)$ is the value of a target variable at some sampled location and $z(\mathbf{s}_i + \mathbf{h})$ is the value of the neighbor at distance $\mathbf{s}_i + \mathbf{h}$. Suppose that there are n point observations, this yields $n \cdot (n-1)/2$ pairs for which a semivariance can be calculated. We can then plot all semivariances versus their separation distances, which will produce a variogram cloud as shown in Fig. 1.9b. Such clouds are not easy to describe visually, so the values are commonly averaged for a standard distance called the "*lag*". If we display such averaged data, then we get a standard **experimental or sample variogram** as shown in Fig. 1.9c. What we usually expect to see is that semivariances are smaller at shorter distance and then they stabilize at some distance within the extent of a study area. This can be interpreted as follows: the values of a target variable are more similar at shorter distance, up to a certain distance where the differences between the pairs are more less equal to the global variance[18].

From a meta-physical perspective, spatial auto-correlation in the data can be considered as a result of **diffusion** — a random motion causing a system to decay towards uniform conditions. One can argue that, if there is a physical process behind a feature, one should model it using a deterministic function rather than

[16] A somewhat similar theory was promoted by Gandin (1963) at about the same time.

[17] Matheron (1962) named his theoretical framework the *Theory of Regionalized Variables*. It was basically a theory for modeling stochastic surfaces using spatially sampled variables.

[18] For this reason, many geostatistical packages (e.g. `Isatis`) automatically plot the global variance (horizontal line) directly in a variogram plot.

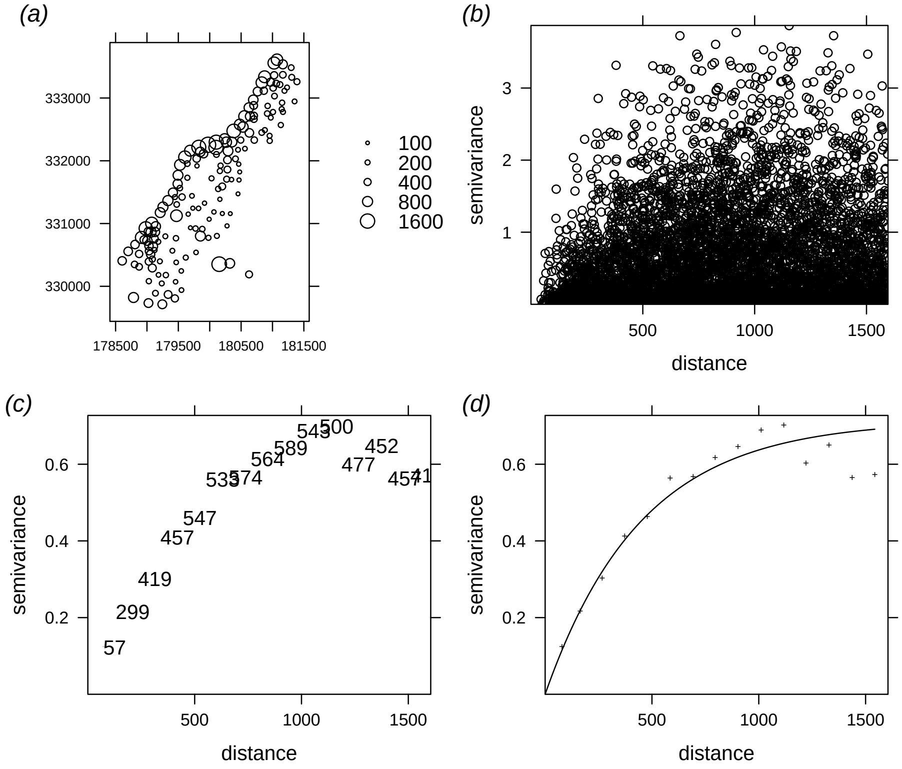


Fig. 1.9: Steps of variogram modeling: (a) sampling locations (155) and measured values of the target variable, (b) variogram cloud showing semivariances for all pairs (log-transformed variable), (c) semivariances aggregated to lags of about 100 m, and (d) the final variogram model fitted using the default settings in gstat. See further p.130.

treating it as a stochastic component. Recall from section 1.1.2, diffusion is a random motion so that there is a meta-statistical argument to treat it as a stochastic component.

Once we calculate an experimental variogram, we can fit it using some of the **authorized variogram models**, such as *linear*, *spherical*, *exponential*, *circular*, *Gaussian*, *Bessel*, *power* and similar (Isaaks and Srivastava, 1989; Goovaerts, 1997). The variograms are commonly fitted by iterative reweighted least squares estimation, where the weights are determined based on the number of point pairs or based on the distance. Most commonly, the weights are determined using $N_j/\mathbf{h}_j^2$, where N_j is the number of pairs at a certain lag, and $\mathbf{h}_j$ is the distance (Fig. 1.9d). This means that the algorithm will give much more importance to semivariances with a large number of point pairs and to shorter distances. Fig. 1.9d shows the result of automated variogram fitting given an experimental variogram (Fig. 1.9c) and using the $N_j/\mathbf{h}_j^2$–weights: in this case, we obtained an exponential model with the nugget parameter = 0, sill parameter = 0.714, and the range parameter = 449 m. Note that this is only a sample variogram — if we would go and collect several point samples, each would lead to a somewhat different variogram plot. It is also important to note that there is a difference between the range factor and the range of spatial dependence, also known as the **practical range**. A practical range is the lag $\mathbf{h}$ for which e.g. $\gamma(\mathbf{h}) \cong 0.95\ \gamma(\infty)$, i.e. that is distance at which the semivariance is close to 95% of the sill (Fig. 1.10b).

The target variable is said to be *stationary* if several sample variograms are '*similar*' (if they do not differ statistically), which is referred to as the **covariance stationarity** or second order stationarity. In summary, three important requirements for ordinary kriging are: (1) the trend function is constant (μ =constant); (2) the variogram is constant in the whole area of interest; (3) the target variable follows (approximately) a normal distribution. In practice, these requirements are often not met, which is a serious limitation of ordinary kriging.

Once we have estimated[19] the variogram model, we can use it to derive semivariances at all locations and

[19] We need to determine the parameters of the variogram model: e.g. the nugget (C_0), sill (C_1) and the range (R) parameter. By knowing these parameters, we can estimate semivariances at any location in the area of interest.

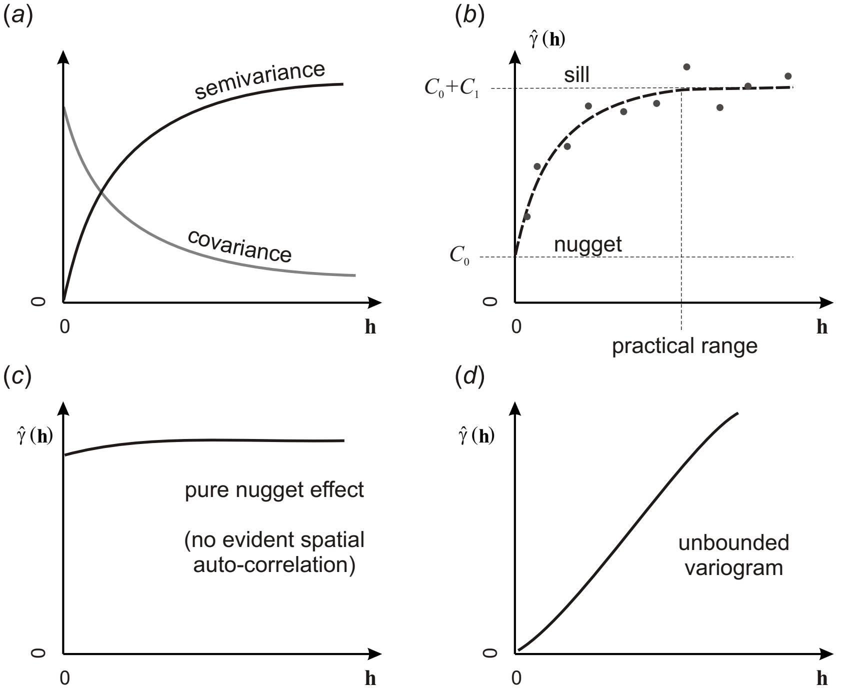


Fig. 1.10: Some basic concepts about variograms: (a) the difference between semivariance and covariance; (b) it is often important in geostatistics to distinguish between the sill variation ($C_0 + C_1$) and the sill parameter (C_1) and between the range parameter (R) and the practical range; (c) a variogram that shows no spatial correlation can be defined by a single parameter (C_0); (d) an unbounded variogram.

solve the kriging weights. The kriging OK weights are solved by multiplying the covariances:

$$\lambda_{\mathbf{0}} = \mathbf{C}^{-1} \cdot \mathbf{c}_{\mathbf{0}}; \qquad C(|\mathbf{h}| = 0) = C_0 + C_1 \tag{1.3.4}$$

where $\mathbf{C}$ is the covariance matrix derived for $n \times n$ observations and $\mathbf{c}_{\mathbf{0}}$ is the vector of covariances at a new location. Note that the $\mathbf{C}$ is in fact $(n+1) \times (n+1)$ matrix if it is used to derive kriging weights. One extra row and column are used to ensure that the sum of weights is equal to one:

$$\begin{bmatrix} C(\mathbf{s}_1,\mathbf{s}_1) & \cdots & C(\mathbf{s}_1,\mathbf{s}_n) & 1 \\ \vdots & & \vdots & \vdots \\ C(\mathbf{s}_n,\mathbf{s}_1) & \cdots & C(\mathbf{s}_n,\mathbf{s}_n) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} C(\mathbf{s}_0,\mathbf{s}_1) \\ \vdots \\ C(\mathbf{s}_0,\mathbf{s}_n) \\ 1 \end{bmatrix} = \begin{bmatrix} w_1(\mathbf{s}_0) \\ \vdots \\ w_n(\mathbf{s}_0) \\ \varphi \end{bmatrix} \tag{1.3.5}$$

where φ is the so-called *Lagrange multiplier*.

In addition to estimation of values at new locations, a statistical spatial prediction technique produces a measure of associated uncertainty of making predictions by using a given model. In geostatistics, this is often referred to as the **prediction variance**, i.e. the estimated variance of the prediction error. OK variance is defined as the weighted average of covariances from the new point ($\mathbf{s}_0$) to all calibration points ($\mathbf{s}_1, .., \mathbf{s}_n$), plus the Lagrange multiplier (Webster and Oliver, 2001, p.183):

$$\hat{\sigma}^2_{\mathtt{OK}}(\mathbf{s}_0) = (C_0 + C_1) - \mathbf{c}_{\mathbf{0}}^{\mathrm{T}} \cdot \lambda_{\mathbf{0}} = C_0 + C_1 - \sum_{i=1}^{n} w_i(\mathbf{s}_0) \cdot C(\mathbf{s}_0, \mathbf{s}_i) + \varphi \tag{1.3.6}$$

where $C(\mathbf{s}_0, \mathbf{s}_i)$ is the covariance between the new location and the sampled point pair, and φ is the Lagrange multiplier, as shown in Eq.(1.3.5).

Outputs from any statistical prediction model are commonly two maps: (1) predictions and (2) prediction variance. The mean of the prediction variance at all locations can be termed the **overall prediction variance**, and can be used as a measure of the overall precision of the final map: if the overall prediction variance gets close to the global variance, then the map is 100% imprecise; if the overall prediction variance tends to zero, then the map is 100% precise[20] (see further Fig. 5.19).

Note that a common practice in geostatistics is to model the variogram using a semivariance function and then, for reasons of computational efficiency, use the **covariances**. In the case of solving the kriging weights, both the matrix of semivariances and covariances give the same results, so you should not really make a difference between the two. The relation between the covariances and semivariances is (Isaaks and Srivastava, 1989, p.289):

$$C(\mathbf{h}) = C_0 + C_1 - \gamma(\mathbf{h}) \tag{1.3.7}$$

where $C(\mathbf{h})$ is the covariance, and $\gamma(\mathbf{h})$ is the semivariance function (Fig. 1.10a). So for example, an exponential model can be written in two ways:

$$\gamma(\mathbf{h}) = \begin{cases} 0 & \text{if} \quad |\mathbf{h}| = 0 \\ C_0 + C_1 \cdot \left[1 - e^{-\left(\frac{\mathbf{h}}{R}\right)}\right] & \text{if} \quad |\mathbf{h}| > 0 \end{cases} \tag{1.3.8}$$

$$C(\mathbf{h}) = \begin{cases} C_0 + C_1 & \text{if} \quad |\mathbf{h}| = 0 \\ C_1 \cdot \left[e^{-\left(\frac{\mathbf{h}}{R}\right)}\right] & \text{if} \quad |\mathbf{h}| > 0 \end{cases} \tag{1.3.9}$$

The covariance at zero distance ($C(0)$) is by definition equal to the mean residual error (Cressie, 1993) — $C(\mathbf{h}_{11})$ also written as $C(\mathbf{s}_1, \mathbf{s}_1)$, and which is equal to $C(0) = C_0 + C_1 = Var\{z(s)\}$.

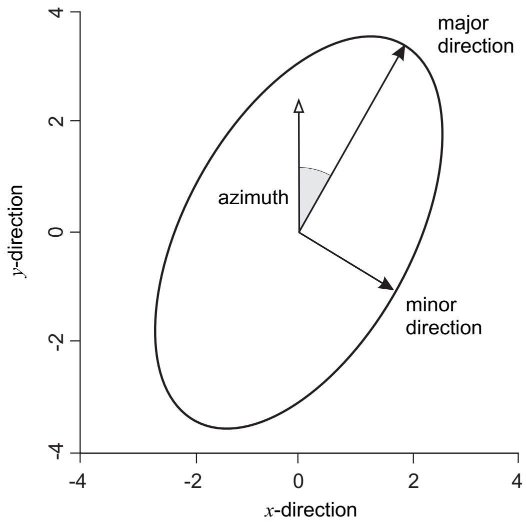


Fig. 1.11: Range ellipse for anisotropic model. After gstat User's manual.

[20] As we will see later on, the precision of mapping is only a measure of how well did we fit the point values. The true quality of map can only be accessed by using validation points, preferably independent from the point data set used to make predictions.

The variogram models can be extended to even larger number of parameters if either (a) **anisotropy** or (b) smoothness are considered in addition to modeling of nugget and sill variation. The 2D geometric anisotropy in `gstat`[21], for example, is modeled by replacing the range parameter with three parameters — range in the major direction (direction of the strongest correlation), angle of the principal direction and the anisotropy ratio, e.g. (Fig. 1.11):

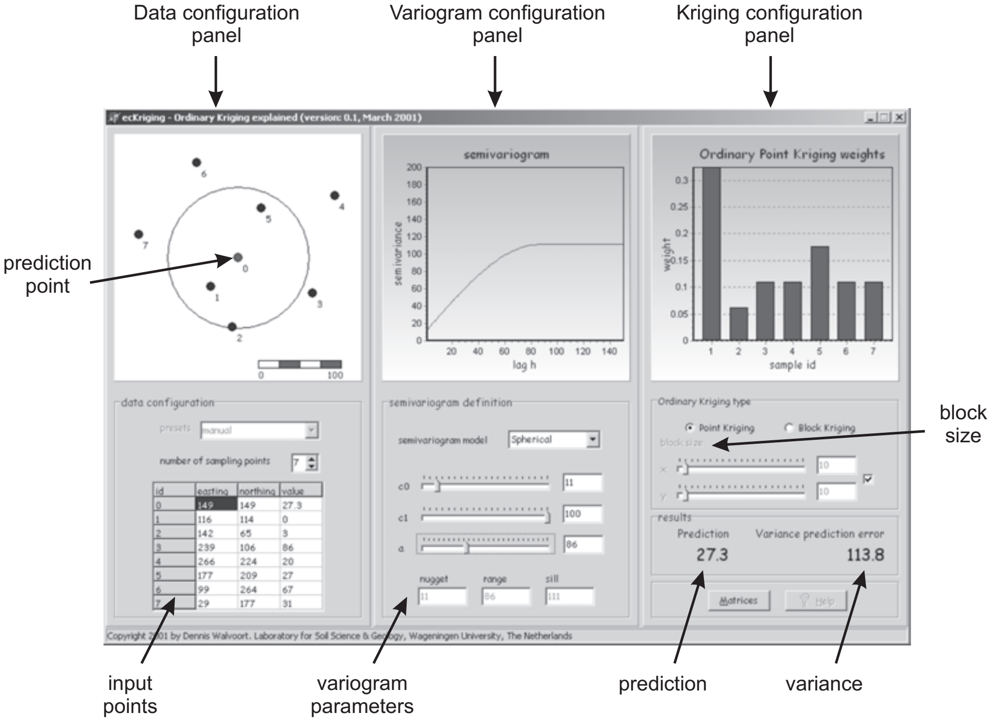
```
> vgm(nugget=1, model="Sph", sill=10, range=2, anis=c(30,0.5))
```

where the value of the angle of major direction is `30` (azimuthal direction measured in degrees clockwise), and the value of the anisotropy ratio is `0.5` (range in minor direction is two times shorter). There is no universal rule whether to use always anisotropic models or to use them only if the variogram shows significant anisotropy. As a rule of thumb, we can say that, if the variogram confidence bands (see further Fig. 5.15) in the two orthogonal directions (major and minor direction) show <50% overlap, than one needs to consider using anisotropic models.

Another sophistication of the standard 3–parameter variograms is the Matérn variogram model, which has an additional parameter to describe the smoothness (Stein, 1999; Minasny and McBratney, 2005):

$$\gamma(\mathbf{h}) = C_0 \cdot \delta(\mathbf{h}) + C_1 \cdot \left[\frac{1}{2^{\nu-1} \cdot \Gamma(\nu)} \cdot \left(\frac{\mathbf{h}}{R} \right)^{\nu} \cdot K_\nu \cdot \left(\frac{\mathbf{h}}{R} \right) \right] \tag{1.3.10}$$

where $\delta(\mathbf{h})$ is the Kronecker delta, K_ν is the modified Bessel function, Γ is the gamma function and ν is the smoothness parameter. The advantage of this model is that it can be used universally to model both short and long distance variation (see further section 10.3.2). In reality, variogram models with more parameters are more difficult to fit automatically because the iterative algorithms might get stuck in local minima (Minasny and McBratney, 2005).



data configuration

presets manual

number of sampling points 7

id	easting	northing	value
0	149	149	27.3
1	116	114	0
2	142	65	3
3	239	106	86
4	266	224	20
5	177	209	27
6	99	264	67
7	29	177	31

semivariogram definition

semivariogram model Spherical

c0 11

c1 100

a 86

nugget 11 range 86 sill 111

Ordinary Kriging type

● Point Kriging ○ Block Kriging

block size

x 10

y 10 ☑

results

Prediction 27.3

Variance prediction error 113.8

Copyright 2001 by Dennis Walvoort. Laboratory for Soil Science & Geology, Wageningen University, The Netherlands

Fig. 1.12: Ordinary kriging explained: E*Z*-Kriging. Courtesy of Dennis J.J. Walvoort, Wageningen University.

The fastest intuitive way to understand the principles of kriging is to use an educational program called **E*Z*-Kriging**, kindly provided by Dennis J.J. Walvoort from the Alterra Research institute. The GUI of E*Z*-

[21] `http://www.gstat.org/manual/node20.html`

Kriging consists of three panels: (1) data configuration panel, (2) variogram panel, and (3) kriging panel (Fig. 1.12). This allows you to zoom into ordinary kriging and explore its main characterizes and behavior: how do weights change for different variogram models, how do data values affect the weights, how does block size affect the kriging results etc. For example, if you study how model shape, nugget, sill and range affect the kriging results, you will notice that, assuming some standard variogram model (zero nugget, sill at global variance and practical range at 10% of the largest distance), the weights will decrease exponentially[22]. This is an important characteristic of kriging because it allows us to limit the search window to speed up the calculation and put more emphasize on fitting the semivariances at shorter distances. Note also that, although it commonly leads to smoothing of the values, kriging is an exact and non-convex interpolator. It is exact because the kriging estimates are equal to input values at sampling locations, and it is non-convex because its predictions can be outside the data range, e.g. we can produce negative concentrations.

Another important aspect of using kriging is the issue of the support size. In geostatistics, one can control the support size of the outputs by averaging multiple (randomized) point predictions over regular blocks of land. This is known as **block prediction** (Heuvelink and Pebesma, 1999). A problem is that we can sample elevations at point locations, and then interpolate them for blocks of e.g. 10×10 m, but we could also take composite samples and interpolate them at point locations. This often confuses GIS users because as well as using point measurements to interpolate values at regular point locations (e.g. by point kriging), and then display them using a raster map (see Fig. 1.8), we can also make spatial predictions for blocks of land (block kriging) and display them using the same raster model (Bishop and McBratney, 2001). For simplicity, in the case of block-kriging, one should always try to use a cell size that corresponds to the support size.

1.3.2 Environmental correlation

If some exhaustively-sampled explanatory variables or **covariates** are available in the area of interest and if they are significantly correlated with our target variable (spatial cross-correlation), and assuming that the point-values are not spatially auto-correlated, predictions can be obtained by focusing only on the deterministic part of variation:

$$Z(\mathbf{s}) = f\left\{q_k(\mathbf{s})\right\} + \varepsilon \tag{1.3.11}$$

where q_k are the auxiliary predictors. This approach to spatial prediction has a strong physical interpretation. Consider Rowe and Barnes (1994) observation that earth surface energy-moisture regimes at all scales/sizes are the dynamic driving variables of functional ecosystems at all scales/sizes. The concept of vegetation/soil-environment relationships has frequently been presented in terms of an equation with six key **environmental factors** as:

$$V \times S[x, y, \tilde{t}] = f \left\{ \begin{array}{l} s[x, y, \tilde{t}]\ c[x, y, \tilde{t}]\ o[x, y, \tilde{t}] \\ r[x, y, \tilde{t}]\ p[x, y, \tilde{t}]\ a[x, y, \tilde{t}] \end{array} \right. \tag{1.3.12}$$

where V stands for vegetation, S for soil, c stands for climate, o for organisms (including humans), r is relief, p is parent material or geology, a is age of the system, x, y are the coordinates and t is time dimension. This means that the predictors which are available over entire areas of interest can be used to predict the value of an environmental variable at unvisited locations — first by modeling the relationship between the target and explanatory environmental predictors at sample locations, and then by applying it to unvisited locations using the known value of the explanatory variables at those locations. Common explanatory environmental predictors used to map environmental variables are land surface parameters, remotely sensed images, and geological, soil and land-use maps (McKenzie and Ryan, 1999). Because many auxiliary predictors (see further section 4) are now also available at low or no cost, this approach to spatial prediction is ever more important (Pebesma, 2006; Hengl et al., 2007a).

Functional relations between environmental variables and factors are in general unknown and the correlation coefficients can differ for different study areas, different seasons and different scales. However, in

[22] In practice, often >95% of weights will be explained by the nearest 30–50 points. Only if the variogram is close to the pure nugget model, the more distant points will receive more importance, but then the technique will produce poor predictions anyhow.

many cases, relations with environmental predictors often reflect causal linkage: deeper and more developed soils occur at places of higher potential accumulation and lower slope; different type of forests can be found at different slope expositions and elevations; soils with more organic matter can be found where the climate is cooler and wetter etc. This makes this technique especially suitable for natural resource inventory teams because it allows them to validate their empirical knowledge about the variation of the target features in the area of interest.

There are (at least) four groups of statistical models that have been used to make spatial predictions with the help of environmental factors (Chambers and Hastie, 1992; McBratney et al., 2003; Bishop and Minasny, 2005):

Classification-based models — Classification models are primarily developed and used when we are dealing with discrete target variables (e.g. land cover or soil types). There is also a difference whether **Boolean** (crisp) or **Fuzzy** (continuous) classification rules are used to create outputs. Outputs from the model fitting process are class boundaries (class centres and standard deviations) or classification rules.

Tree-based models — Tree-based models (classification or regression trees) are often easier to interpret when a mix of continuous and discrete variables are used as predictors (Chambers and Hastie, 1992). They are fitted by successively splitting a data set into increasingly homogeneous groupings. Output from the model fitting process is a **decision tree**, which can then be applied to make predictions of either individual property values or class types for an entire area of interest.

Regression models — Regression analysis employs a family of functions called **Generalized Linear Models** (GLMs), which all assume a linear relationship between the inputs and outputs (Neter et al., 1996). Output from the model fitting process is a set of regression coefficients. Regression models can be also used to represent non-linear relationships with the use of **General Additive Models** (GAMs). The relationship between the predictors and targets can be solved using one-step data-fitting or by using iterative data fitting techniques (neural networks and similar).

Each of the models listed above can be equally applicable for mapping of environmental variables and each can exhibit advantages and disadvantages. For example, some advantages of using tree-based regression are that they: (1) can handle missing values; (2) can use continuous and categorical predictors; (3) are robust to predictor specification; and (4) make very limited assumptions about the form of the regression model (Henderson et al., 2004). Some disadvantages of regression trees, on the other hand, are that they require large data sets and completely ignore spatial position of the input points.

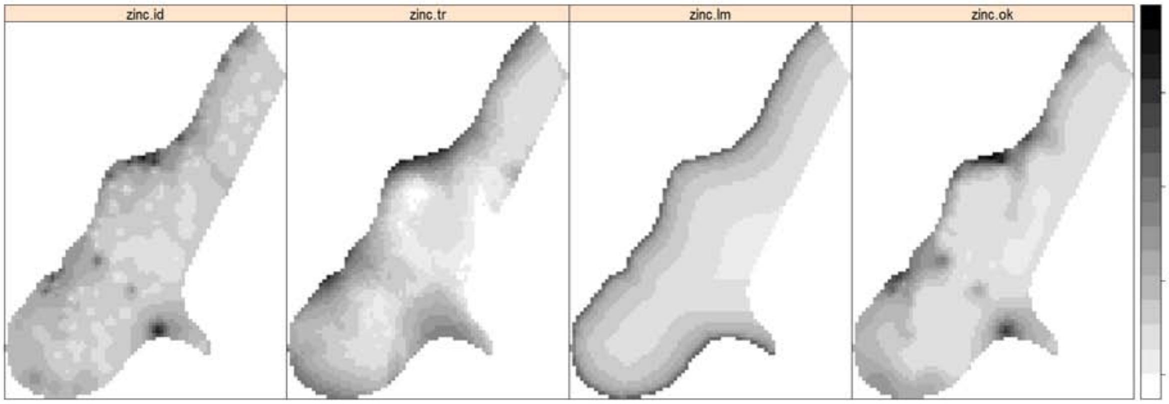


Fig. 1.13: Comparison of spatial prediction techniques for mapping `Zinc` (sampling locations are shown in Fig. 1.9). Note that inverse distance interpolation (`.id`) and kriging (`.ok`) are often quite similar; the moving trend surface (`.tr`; 2nd order polynomial) can lead to artifacts (negative values) — locally where the density of points is poor. The regression-based (`.lm`) predictions were produced using distance from the river as explanatory variable (see further §5).

A common regression-based approach to spatial prediction is **multiple linear regression** (Draper and Smith, 1998; Kutner et al., 2004). Here, the predictions are again obtained by weighted averaging (compare with Eq.(1.3.2)), this time by averaging the predictors:

$$\hat{z}_{\texttt{OLS}}(\mathbf{s}_0) = \hat{b}_0 + \hat{b}_1 \cdot q_1(\mathbf{s}_0) + \ldots + \hat{b}_p \cdot q_p(\mathbf{s}_0) = \sum_{k=0}^{p} \hat{\beta}_k \cdot q_k(\mathbf{s}_0); \qquad q_0(\mathbf{s}_0) \equiv 1 \tag{1.3.13}$$

or in matrix algebra:

$$\hat{z}_{\texttt{OLS}}(\mathbf{s}_0) = \hat{\beta}^{\mathbf{T}} \cdot \mathbf{q} \tag{1.3.14}$$

where $q_k(\mathbf{s}_0)$ are the values of the explanatory variables at the target location, p is the number of predictors or explanatory variables[23], and $\hat{\beta}_k$ are the regression coefficients solved using the **Ordinary Least Squares**:

$$\hat{\beta} = \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} \cdot \mathbf{z} \tag{1.3.15}$$

where $\mathbf{q}$ is the matrix of predictors $(n \times p + 1)$ and $\mathbf{z}$ is the vector of sampled observations. The prediction error of a multiple linear regression model is (Neter et al., 1996, p.210):

$$\hat{\sigma}^2_{\texttt{OLS}}(\mathbf{s}_0) = MSE \cdot \left[1 + \mathbf{q}_0^{\mathrm{T}} \cdot \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}_0\right] \tag{1.3.16}$$

where *MSE* is the mean square (residual) error around the regression line:

$$MSE = \frac{\sum\limits_{i=1}^{n} \left[z(\mathbf{s}_i) - \hat{z}(\mathbf{s}_i)\right]^2}{n-2} \tag{1.3.17}$$

and $\mathbf{q}_0$ is the vector of predictors at new, unvisited location. In the univariate case, the variance of the prediction error can also be derived using:

$$\hat{\sigma}^2(\mathbf{s}_0) = MSE \cdot \left[1 + \frac{1}{n} + \frac{\left[q(\mathbf{s}_0) - \bar{q}\right]^2}{\sum\limits_{i=1}^{n} \left[q(\mathbf{s}_i) - \bar{q}\right]^2}\right] = MSE \cdot \left[1 + v(\mathbf{s}_0)\right] \tag{1.3.18}$$

where v is the curvature of the confidence band around the regression line. This reflects the amount of extrapolation in the feature space (Ott and Longnecker, 2001, p.570). It can be seen from Eq. (1.3.18) that the prediction error, for a given sampling intensity ($n/\mathbb{A}$), depends on three factors:

(1.) Mean square residual error (*MSE*);

(2.) Spreading of points in the feature space $\sum \left[q(\mathbf{s}_i) - \bar{q}\right]^2$;

(3.) '*Distance*' of the new observation from the centre of the feature space $\left[q(\mathbf{s}_0) - \bar{q}\right]$.

So in general, if the model is linear, we can decrease the prediction variance if we increase the spreading of the points in feature space. Understanding this principles allows us to prepare sampling plans that will achieve higher mapping precision and minimize extrapolation in feature space (see further §2.8).

[23] To avoid confusion with geographical coordinates, we use the symbol q, instead of the more common x, to denote a predictor.

The sum of squares of residuals (*SSE*) can be used to determine the **adjusted coefficient of multiple determination** (R_a^2), which describes the goodness of fit:

$$\begin{aligned} R_a^2 &= 1 - \left(\frac{n-1}{n-p}\right) \cdot \frac{SSE}{SSTO} \\ &= 1 - \left(\frac{n-1}{n-p}\right) \cdot \left(1 - R^2\right) \end{aligned} \tag{1.3.19}$$

where *SSTO* is the total sum of squares (Neter et al., 1996), R^2 indicates amount of variance explained by the model, whereas R_a^2 adjusts for the number of variables (p) used. For many environmental mapping projects, a R_a^2 ≥0.85 is already a very satisfactory solution and higher values will typically only mean over-fitting of the data (Park and Vlek, 2002).

The principle of predicting environmental variables using factors of climate, relief, geology and similar, is often referred to as **environmental correlation**. The *environmental correlation approach* to mapping is a true alternative to ordinary kriging (compare differences in generated patterns in Fig. 1.13). This is because both approaches deal with different aspects of spatial variation: regression deals with the deterministic and kriging with the spatially-correlated stochastic part of variation.

The biggest criticism of the pure regression approach to spatial prediction is that the position of points in geographical space is completely ignored, both during model fitting and prediction. Imagine if we are dealing with two point data sets where one data set is heavily clustered, while the other is well-spread over the area of interest — a sophistication of simple non-spatial regression is needed to account for the clustering of the points so that the model derived using the clustered points takes this property into account.

One way to account for this problem is to take the distance between the points into account during the estimation of the regression coefficients. This can be achieved by using the **geographically weighted regression** (Fotheringham et al., 2002). So instead of using the OLS estimation (Eq.1.3.15) we use:

$$\hat{\beta}_{\mathtt{WLS}} = \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{W} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} \cdot \mathbf{W} \cdot \mathbf{z} \tag{1.3.20}$$

where **W** is a matrix of weights, determined using some distance decay function e.g.:

$$w_i(\mathbf{s}_i, \mathbf{s}_j) = \sigma_E^2 \cdot \exp\left[-3 \cdot \frac{d^2(\mathbf{s}_i, \mathbf{s}_j)}{\daleth^2}\right] \tag{1.3.21}$$

where σ_E^2 is the level of variation of the error terms, $d(\mathbf{s}_i, \mathbf{s}_j)$ is the Euclidian distance between a sampled point pair and $\daleth$ is known as the bandwidth, which determines the degree of *locality* — small values of $\daleth$ suggest that correlation only occurs between very close point pairs and large values suggest that such effects exist even on a larger spatial scale. Compare further with Eq.(2.1.3). The problem remains to select a search radius (Eq.1.3.21) using objective criteria. As we will see further (§2.2), geographically weighted regression can be compared with regression-kriging with a moving window where variograms and regression models are estimated locally.

The main benefit of geographically weighted regression (**GWR**) is that this method enables researchers to study local differences in responses to input variables. It therefore more often focuses on coefficients' explanation than on interpolation of the endogenous variables. By setting up the search radius (bandwidth) one can investigate the impact of spatial proximity between the samples on the regression parameters. By fitting the regression models using a moving window algorithm, one can also produce maps of regression coefficients and analyze how much the regression model is dependent on the location. However, the coefficient maps generated by GWR are usually too smooth to be true. According to Wheeler and Tiefelsdorf (2005) and Griffith (2008), the two main problems with GWR are: (1) strong multicollinearity effects among coefficients make the results even totally wrong, and (2) lose degrees of freedom in the regression model. Hence, spatial hierarchical model under bayesian framework (Gelfand et al., 2003) and spatial filtering model (Griffith, 2008) may be better structures for such analyzes than GWR.

1.3.3 Predicting from polygon maps

A special case of environmental correlation is prediction from polygon maps i.e. stratified areas (different land use/cover types, geological units etc). Assuming that the residuals show no spatial auto-correlation, a value at a new location can be predicted by:

$$\hat{z}(\mathbf{s}_0) = \sum_{i=1}^{n} w_i \cdot z(s); \qquad w_i = \begin{cases} 1/n_k & \text{for } x_i \in k \\ 0 & \text{otherwise} \end{cases} \tag{1.3.22}$$

where k is the unit identifier. This means that the weights within some unit will be equal so that the predictions are made by simple averaging per unit (Webster and Oliver, 2001):

$$\hat{z}(\mathbf{s}_0) = \bar{\mu}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} z(\mathbf{s}_i) \tag{1.3.23}$$

Consequently, the output map will show only abrupt changes in the values between the units. The prediction variance of this prediction model is simply the within-unit variance:

$$\hat{\sigma}^2(\mathbf{s}_0) = \frac{\sigma_k^2}{n_k} \tag{1.3.24}$$

From Eq.(1.3.24) it is obvious that the precision of the technique will be maximized if the within-unit variation is infinitely small. Likewise, if the within-unit variation is as high as the global variability, the predictions will be as bad as predicting by taking any value from the normal distribution.

Another approach to make predictions from polygon maps is to use multiple regression. In this case, the predictors (mapping units) are used as indicators:

$$\hat{z}(\mathbf{s}_0) = \hat{b}_1 \cdot MU_1(\mathbf{s}_0) + \ldots + \hat{b}_k \cdot MU_k(\mathbf{s}_0); \qquad MU_k \in [0|1] \tag{1.3.25}$$

and it can be shown that the OLS fitted regression coefficients will equal the mean values within each strata ($b_k = \bar{\mu}(MU_k)$), so that the Eqs.(1.3.25) and (1.3.23) are in fact equivalent.

If, on the other hand, the residuals do show spatial auto-correlation, the predictions can be obtained by **stratified kriging**. This is basically ordinary kriging done separately for each strata and can often be impractical because we need to estimate a variogram for each of the k strata (Boucneau et al., 1998). Note that the strata or sub-areas need to be known *a priori* and they should never be derived from the data used to generate spatial predictions.

1.3.4 Hybrid models

Hybrid spatial prediction models comprise of a combination of the techniques listed previously. For example, a hybrid geostatistical model employs both correlation with auxiliary predictors and spatial autocorrelation simultaneously. There are two main sub-groups of hybrid geostatistical models (McBratney et al., 2003): (a) **co-kriging**-based and (b) **regression-kriging**-based techniques, but the list could be extended.

Note also that, in the case of environmental correlation by linear regression, we assume some basic (additive) model, although the relationship can be much more complex. To account for this, a linear regression model can be extended to a diversity of statistical models ranging from regression trees, to General Additive Models and similar. Consequently, the hybrid models are more generic than pure kriging-based or regression-based techniques and can be used to represent both discrete and continuous changes in the space, both deterministic and stochastic processes.

One can also combine deterministic, statistical and expert-based estimation models. For example, one can use a deterministic model to estimate a value of the variable, then use actual measurements to fit a calibration model, analyze the residuals for spatial correlation and eventually combine the statistical fitting and deterministic modeling (Hengl et al., 2007a). Most often, expert-based models are supplemented with the actual measurements, which are then used to refine the rules, e.g. using neural networks (Kanevski et al., 1997).

1.4 Validation of spatial prediction models

OK or OLS variance (Eqs.1.3.6; and 1.3.18) is the statistical estimate of the model uncertainty. Note that the '*true*' prediction power can only be assessed by using an independent (control) data set. The prediction error is therefore often referred to as the *precision of prediction*. The true quality of a map can be best assessed by comparing estimated values ($\hat{z}(\mathbf{s}_j)$) with actual observations at validation points ($z^*(\mathbf{s}_j)$). Commonly, two measures are most relevant here — (1) the mean prediction error (*ME*):

$$ME = \frac{1}{l} \cdot \sum_{j=1}^{l} \left[\hat{z}(\mathbf{s}_j) - z^*(\mathbf{s}_j) \right]; \qquad E\{ME\} = 0 \tag{1.4.1}$$

and (2) the root mean square prediction error (*RMSE*):

$$RMSE = \sqrt{\frac{1}{l} \cdot \sum_{j=1}^{l} \left[\hat{z}(\mathbf{s}_j) - z^*(\mathbf{s}_j) \right]^2}; \qquad E\{RMSE\} = \sigma(\mathbf{h} = 0) \tag{1.4.2}$$

where l is the number of validation points. We can also standardize the errors based on the prediction variance estimated by the spatial prediction model:

$$RMNSE = \sqrt{\frac{1}{l} \cdot \sum_{j=1}^{l} \left[\frac{\hat{z}(\mathbf{s}_j) - z^*(\mathbf{s}_j)}{\hat{\sigma}_j} \right]^2}; \qquad E\{RMNSE\} = 1 \tag{1.4.3}$$

In order to compare accuracy of prediction between variables of different types, the *RMSE* can also be normalized by the total variation:

$$RMSE_r = \frac{RMSE}{s_z} \tag{1.4.4}$$

which will show how much of the global variation budget has been explained by the model. As a rule of thumb, a value of $RMSE_r$ that is close to 40% means a fairly satisfactory accuracy of prediction (R-square=85%). Otherwise, if $RMSE_r$ >71%, this means that the model accounted for less than 50% of variability at the validation points. Note also that *ME*, *RMSE* and *RMNSE* estimated at validation points are also only a sample from a population of values — if the validation points are poorly sampled, our estimate of the map quality may be equally poor.

Because collecting additional (independent) samples is often impractical and expensive, validation of prediction models is most commonly done by using **cross-validation** i.e. by subsetting the original point set in two data set — calibration and validation — and then repeating the analysis. There are several types of cross-validation methods (Bivand et al., 2008, pp.221–226):

- the k–fold cross-validation — the original sample is split into k equal parts and then each is used for cross-validation;
- *leave-one-out* cross-validation (LOO) — each sampling point is used for cross-validation;
- *Jackknifing* — similar to LOO, but aims at estimating the bias of statistical analysis and not of predictions;

Both k–fold and the *leave-one-out* cross validation are implemented in the `krige.cv` method of `gstat` package. The LOO algorithm works as follows: it visits a data point, predicts the value at that location by kriging *without* using the observed value, and proceeds with the next data point. This way each individual point is assessed versus the whole data set. The results of cross-validation can be visualised to pinpoint the most problematic points, e.g. exceeding three standard deviations of the normalized prediction error, and to derive a summary estimate of the map accuracy. In the case of many outliers and blunders in the input data, the LOO cross-validation might produce strange outputs. Hence many authors recommend 10–fold cross-validation as the most robust approach. Note also that cross-validation is not necessarily independent — points used for cross-validation are subset of the original sampling design, hence if the original design is biased and/or non-representative, then also the cross-validation might not reveal the true accuracy of a technique. However, if the sampling design has been generated using e.g. random sampling, it can be shown that also randomly taken subsets will be unbiased estimators of the true accuracy.

To assess the accuracy of predicting categorical variables we can use the **kappa statistics**, which is a common measure of classification accuracy (Congalton and Green, 1999; Foody, 2004). Kappa statistics measures the difference between the actual agreement between the predictions and ground truth and the agreement that could be expected by chance (see further p.135). In most remote sensing-based mapping projects, a kappa larger than 85% is considered to be a satisfactory result (Foody, 2004). The kappa is only a measure of the overall mapping accuracy. Specific classes can be analyzed by examining the percentage of correctly classified pixels per each class:

$$P_c = \frac{\sum\limits_{j=1}^{m} \left(\hat{C}(s_j) = C(s_j) \right)}{m} \tag{1.4.5}$$

where P_c is the percentage of correctly classified pixels, $\hat{C}(s_j)$ is the estimated class at validation locations (s_j) and m is total number of observations of class c at validation points.

Further reading:

★ Cressie, N.A.C., 1993. **Statistics for Spatial Data**, revised edition. John Wiley & Sons, New York, 416 p.

★ Goovaerts, P., 1997. **Geostatistics for Natural Resources Evaluation** (Applied Geostatistics). Oxford University Press, New York, 496 p.

★ Isaaks, E.H. and Srivastava, R.M. 1989. **An Introduction to Applied Geostatistics**. Oxford University Press, New York, 542 p.

★ Webster, R. and Oliver, M.A., 2001. **Geostatistics for Environmental Scientists**. Statistics in Practice. John Wiley & Sons, Chichester, 265 p.

★ http://www.wiley.co.uk/eoenv/ — The Encyclopedia of Environmetrics.

★ http://geoenvia.org — A research association that promotes use of geostatistical methods for environmental applications.

★ http://www.iamg.org — International Association of Mathematical Geosciences.

2

Regression-kriging

As we saw in the previous chapter, there are many geostatistical techniques that can be used to map environmental variables. In reality, we always try to go for the most flexible, most comprehensive and the most robust technique (preferably implemented in a software with an user-friendly GUI). In fact, many (geo)statisticians believe that there is only one Best Linear Unbiased Prediction (**BLUP**) model for spatial data, from which all other (linear) techniques can be derived (Gotway and Stroup, 1997; Stein, 1999; Christensen, 2001). As we will see in this chapter, one such generic mapping technique is regression-kriging. All other techniques mentioned previously — ordinary kriging, environmental correlation, averaging of values per polygons or inverse distance interpolation — can be seen as special cases of RK.

2.1 The Best Linear Unbiased Predictor of spatial data

Matheron (1969) proposed that a value of a target variable at some location can be modeled as a sum of the deterministic and stochastic components:

$$Z(\mathbf{s}) = m(\mathbf{s}) + \varepsilon'(\mathbf{s}) + \varepsilon'' \tag{2.1.1}$$

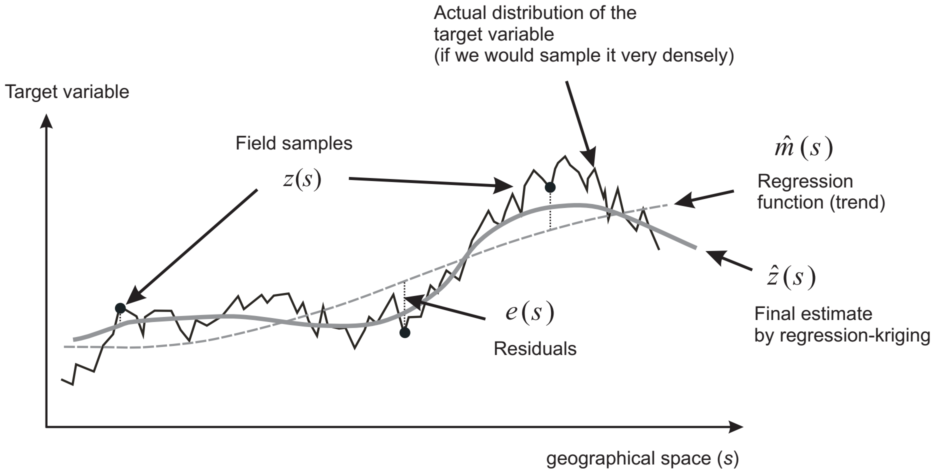


Fig. 2.1: A schematic example of the regression-kriging concept shown using a cross-section.

which he termed the **universal model of spatial variation**. We have seen in the previous sections (§1.3.1 and §1.3.2) that both deterministic and stochastic components of spatial variation can be modeled separately. By combining the two approaches, we obtain:

$$\begin{aligned}\hat{z}(\mathbf{s}_0) &= \hat{m}(\mathbf{s}_0) + \hat{e}(\mathbf{s}_0) \\ &= \sum_{k=0}^{p} \hat{\beta}_k \cdot q_k(\mathbf{s}_0) + \sum_{i=1}^{n} \lambda_i \cdot e(\mathbf{s}_i)\end{aligned} \tag{2.1.2}$$

where $\hat{m}(\mathbf{s}_0)$ is the fitted deterministic part, $\hat{e}(\mathbf{s}_0)$ is the interpolated residual, $\hat{\beta}_k$ are estimated deterministic model coefficients ($\hat{\beta}_0$ is the estimated intercept), λ_i are kriging weights determined by the spatial dependence structure of the residual and where $e(\mathbf{s}_i)$ is the residual at location $\mathbf{s}_i$. The regression coefficients $\hat{\beta}_k$ can be estimated from the sample by some fitting method, e.g. ordinary least squares (OLS) or, optimally, using **Generalized Least Squares** (Cressie, 1993, p.166):

$$\hat{\beta}_{\mathtt{GLS}} = \left(\mathbf{q}^{\mathbf{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathbf{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{z} \tag{2.1.3}$$

where $\hat{\beta}_{\mathtt{GLS}}$ is the vector of estimated regression coefficients, $\mathbf{C}$ is the covariance matrix of the residuals, $\mathbf{q}$ is a matrix of predictors at the sampling locations and $\mathbf{z}$ is the vector of measured values of the target variable. The GLS estimation of regression coefficients is, in fact, a special case of geographically weighted regression (compare with Eq.1.3.20). In this case, the weights are determined objectively to account for the spatial auto-correlation between the residuals.

Once the deterministic part of variation has been estimated, the residual can be interpolated with kriging and added to the estimated trend (Fig. 2.1). Estimation of the residuals and their variogram model is an iterative process: first the deterministic part of variation is estimated using ordinary least squares (OLS), then the covariance function of the residuals is used to obtain the GLS coefficients. Next, these are used to re-compute the residuals, from which an updated covariance function is computed, and so on (Schabenberger and Gotway, 2004, p.286). Although this is recommended as the proper procedure by many geostatisticians, Kitanidis (1994) showed that use of the covariance function derived from the OLS residuals (i.e. a single iteration) is often satisfactory, because it is not different enough from the function derived after several iterations; i.e. it does not affect the final predictions much. Minasny and McBratney (2007) reported similar results: it is often more important to use more useful and higher quality data than to use more sophisticated statistical methods. In some situations[1] however, the model needs to be fitted using the most sophisticated technique to avoid making biased predictions.

In matrix notation, regression-kriging is commonly written as (Christensen, 2001, p.277):

$$\hat{z}_{\mathtt{RK}}(\mathbf{s}_0) = \mathbf{q}_0^{\mathbf{T}} \cdot \hat{\beta}_{\mathtt{GLS}} + \boldsymbol{\lambda}_0^{\mathbf{T}} \cdot (\mathbf{z} - \mathbf{q} \cdot \hat{\beta}_{\mathtt{GLS}}) \tag{2.1.4}$$

where $\hat{z}(\mathbf{s}_0)$ is the predicted value at location $\mathbf{s}_0$, $\mathbf{q}_0$ is the vector of $p+1$ predictors and $\boldsymbol{\lambda}_0$ is the vector of n kriging weights used to interpolate the residuals. The model in Eq.(2.1.4) is considered to be the Best Linear Predictor of spatial data (Christensen, 2001; Schabenberger and Gotway, 2004). It has a prediction variance that reflects the position of new locations (extrapolation effect) in both geographical and feature space:

$$\hat{\sigma}^2_{\mathtt{RK}}(\mathbf{s}_0) = (C_0 + C_1) - \mathbf{c}_0^{\mathbf{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{c}_0 + \left(\mathbf{q}_0 - \mathbf{q}^{\mathbf{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{c}_0\right)^{\mathbf{T}} \cdot \left(\mathbf{q}^{\mathbf{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \left(\mathbf{q}_0 - \mathbf{q}^{\mathbf{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{c}_0\right) \tag{2.1.5}$$

where $C_0 + C_1$ is the sill variation and $\mathbf{c}_0$ is the vector of covariances of residuals at the unvisited location.

[1] For example: if the points are extremely clustered, and/or if the sample is ≪100, and/or if the measurements are noisy or obtained using non-standard techniques.

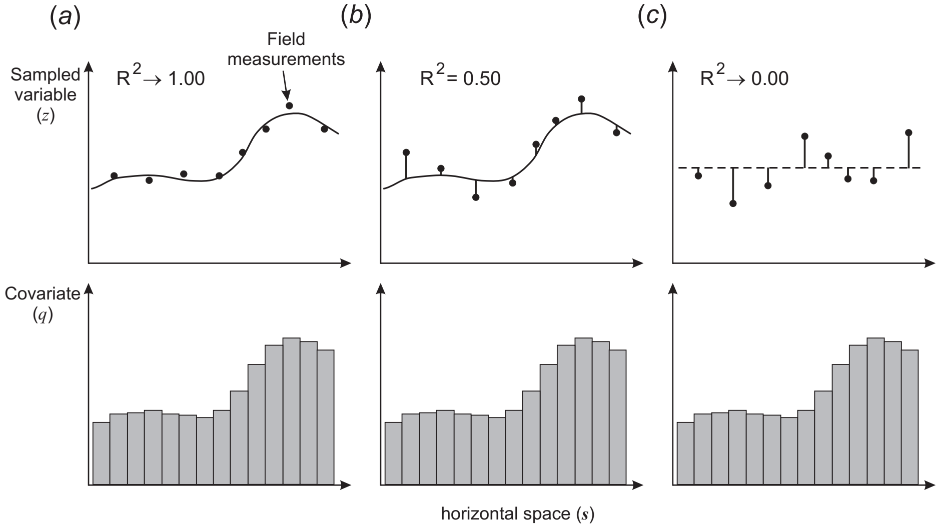


Fig. 2.2: Whether we will use pure regression model, pure kriging or hybrid regression-kriging is basically determined by R-square: (a) if R-square is high, then the residuals will be infinitively small; (c) if R-square is insignificant, then we will probably finish with using ordinary kriging; (b) in most cases, we will use a combination of regression and kriging.

If the residuals show no spatial auto-correlation (pure nugget effect), the regression-kriging (Eq.2.1.4) converges to pure multiple linear regression (Eq.1.3.14) because the covariance matrix (**C**) becomes identity matrix:

$$\mathbf{C} = \begin{bmatrix} C_0 + C_1 & \cdots & 0 \\ \vdots & C_0 + C_1 & 0 \\ 0 & 0 & C_0 + C_1 \end{bmatrix} = (C_0 + C_1) \cdot \mathbf{I} \qquad (2.1.6)$$

so the kriging weights (Eq.1.3.4) at any location predict the mean residual i.e. 0 value. Similarly, the regression-kriging variance (Eq.2.1.5) reduces to the multiple linear regression variance (Eq.1.3.16):

$$\hat{\sigma}^2_{\mathtt{RK}}(\mathbf{s}_0) = (C_0 + C_1) - 0 + \mathbf{q_0^T} \cdot \left(\mathbf{q^T} \cdot \frac{1}{(C_0 + C_1)} \cdot \mathbf{q} \right)^{-1} \cdot \mathbf{q_0}$$

$$\hat{\sigma}^2_{\mathtt{RK}}(\mathbf{s}_0) = (C_0 + C_1) + (C_0 + C_1) \cdot \mathbf{q_0^T} \cdot \left(\mathbf{q^T} \cdot \mathbf{q} \right)^{-1} \cdot \mathbf{q_0}$$

and since $(C_0 + C_1) = C(0) = MSE$, the RK variance reduces to the MLR variance:

$$\hat{\sigma}^2_{\mathtt{RK}}(\mathbf{s}_0) = \hat{\sigma}^2_{\mathtt{OLS}}(\mathbf{s}_0) = MSE \cdot \left[1 + \mathbf{q_0^T} \cdot \left(\mathbf{q^T} \cdot \mathbf{q} \right)^{-1} \cdot \mathbf{q_0} \right] \qquad (2.1.7)$$

Likewise, if the target variable shows no correlation with the auxiliary predictors, the regression-kriging model reduces to ordinary kriging model because the deterministic part equals the (global) mean value (Fig. 2.2c, Eq.1.3.25).

The formulas above show that, depending on the strength of the correlation, RK might turn into pure kriging — if predictors are uncorrelated with the target variable — or pure regression — if there is significant correlation and the residuals show pure nugget variogram (Fig. 2.2). Hence, pure kriging and pure regression should be considered as only special cases of regression-kriging (Hengl et al., 2004a, 2007a).

2.1.1 Mathematical derivation of BLUP

Understanding how a prediction model is derived becomes important once we start getting strange results or poor cross-validation scores. Each model is based on some assumptions that need to be respected and taken into account during the final interpretation of results. A detailed derivation of the BLUP for spatial data can be followed in several standard books on geostatistics (Stein, 1999; Christensen, 2001); one of the first complete derivations is given by Goldberger (1962). Here is a somewhat shorter explanation of how BLUP is derived, and what the implications of various mathematical assumptions are.

All flavors of linear statistical predictors share the same objective of minimizing the estimation error variance $\hat{\sigma}_E^2(\mathbf{s}_0)$ under the constraint of unbiasedness (Goovaerts, 1997). In mathematical terms, the estimation error:

$$\hat{\sigma}^2(\mathbf{s}_0) = E\left\{ \left(\hat{z}(\mathbf{s}_0) - z(\mathbf{s}_0)\right) \cdot \left(\hat{z}(\mathbf{s}_0) - z(\mathbf{s}_0)\right)^T \right\} \tag{2.1.8}$$

is minimized under the (unbiasedness) constraint that:

$$E\left\{\hat{z}(\mathbf{s}_0) - z(\mathbf{s}_0)\right\} = 0 \tag{2.1.9}$$

Assuming the universal model of spatial variation, we can define a generalized linear regression model (Goldberger, 1962):

$$z(\mathbf{s}) = \mathbf{q}^\mathbf{T} \cdot \beta + \varepsilon(\mathbf{s}) \tag{2.1.10}$$

$$E\left\{\varepsilon(\mathbf{s})\right\} = 0 \tag{2.1.11}$$

$$E\left\{\varepsilon \cdot \varepsilon^\mathbf{T}(\mathbf{s})\right\} = \mathbf{C} \tag{2.1.12}$$

where ε is the residual variation, and $\mathbf{C}$ is the $n \times n$ positive-definite variance-covariance matrix of residuals. This model can be read as follows: (1) the information signal is a function of deterministic and residual parts; (2) the best estimate of the residuals is 0; (3) the best estimate of the correlation structure of residuals is the variance-covariance matrix.

Now that we have defined the statistical model and the minimization criteria, we can derive the best linear unbiased prediction of the target variable:

$$\hat{z}(\mathbf{s}_0) = \hat{\delta}_\mathbf{0}^\mathbf{T} \cdot \mathbf{z} \tag{2.1.13}$$

Assuming that we use the model shown in Eq.(2.1.10), and assuming that the objective is to minimize the estimation error $\hat{\sigma}_E^2(\mathbf{s}_0)$, it can be shown[2] that BLUP parameters can be obtained by solving the following system:

$$\begin{bmatrix} \mathbf{C} & \mathbf{q} \\ \mathbf{q}^\mathbf{T} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \delta \\ \phi \end{bmatrix} = \begin{bmatrix} \mathbf{c_0} \\ \mathbf{q_0} \end{bmatrix} \tag{2.1.14}$$

where $\mathbf{c_0}$ is the vector of $n \times 1$ covariances at a new location, $\mathbf{q_0}$ is the vector of $p \times 1$ predictors at a new location, and ϕ is a vector of Lagrange multipliers. It can be further shown that, by solving the Eq.(2.1.14), we get the following:

[2] The actual derivation of formulas is not presented. Readers are advised to obtain the paper by Goldberger (1962).

$$\begin{aligned}\hat{z}(\mathbf{s}_0) &= \mathbf{q}_\mathbf{0}^\mathrm{T} \cdot \hat{\beta} + \hat{\lambda}_\mathbf{0}^\mathrm{T} \cdot (\mathbf{z} - \mathbf{q} \cdot \hat{\beta}) \\ \hat{\beta} &= \left(\mathbf{q}^\mathrm{T} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^\mathrm{T} \cdot \mathbf{C}^{-1} \cdot \mathbf{z} \\ \hat{\lambda}_\mathbf{0} &= \mathbf{C}^{-1} \cdot \mathbf{c_0}\end{aligned} \tag{2.1.15}$$

which is the prediction model explained in the previous section.

Under the assumption of the **first order stationarity** i.e. constant trend:

$$E\left\{z(\mathbf{s})\right\} = \mu \qquad \forall \mathbf{s} \in \mathbb{A} \tag{2.1.16}$$

the Eq.(2.1.15) modifies to (Schabenberger and Gotway, 2004, p.268):

$$\begin{aligned}\hat{z}(\mathbf{s}_0) &= \mu + \hat{\lambda}_\mathbf{0}^\mathrm{T} \cdot (\mathbf{z} - \mu) \\ \hat{\lambda}_\mathbf{0} &= \mathbf{C}^{-1} \cdot \mathbf{c_0}\end{aligned}$$

i.e. to ordinary kriging (§1.3.1). If we assume that the deterministic part of variation is not constant, then we need to consider obtaining a number of covariates ($\mathbf{q}$) that can be used to model the varying mean value.

Another important issue you need to know about the model in Eq.(2.1.15) is that, in order to solve the residual part of variation, we need to know covariances at new locations:

$$C\left(e(\mathbf{s_0}), e(\mathbf{s}_i)\right) = E\left[\left\{e(\mathbf{s_0}) - \mu\right\} \cdot \left\{e(\mathbf{s}_i) - \mu\right\}\right] \tag{2.1.17}$$

which would require that we know the values of the target variable at a new location ($e(\mathbf{s_0})$), which we of course do not know. Instead, we can use the existing sampled values ($e(\mathbf{s}_i) = z(\mathbf{s}_i) - \hat{z}(\mathbf{s}_i)$) to model the covariance structure using a pre-defined mathematical model (e.g. Eq.1.3.8). If we assume that the covariance model is the same (constant) in the whole area of interest, then the covariance is dependent only on the separation vector $\mathbf{h}$:

$$C\left(e(\mathbf{s_0}), e(\mathbf{s}_i)\right) = C(\mathbf{h}) \tag{2.1.18}$$

which is known as the assumption of **second order stationarity**; and which means that we can use the same model to predict values anywhere in the area of interest (global estimation). If this assumption is not correct, we would need to estimate covariance models locally. This is often not so trivial because we need to have a lot of points (see further §2.2), so the assumption of second order stationarity is very popular among geostatisticians. Finally, you need to also be aware that the residuals in Eq.(2.1.10) are expected to be normally distributed around the regression line and homoscedastic[3], as with any linear regression model (Kutner et al., 2004). If this is not the case, then the target variable needs to be transformed until these conditions are met.

The first and second order stationarity, and normality of residuals/target variables are rarely tested in real case studies. In the case of regression-kriging (see further §2.1), the target variable does not have to be stationary but its residuals do, hence we do not have to test this property with the original variable. In the case of regression-kriging in a moving window, we do not have to test neither first nor second order stationarity. Furthermore, if the variable is non-normal, then we can use some sort of GLM to fit the model. If this is successful, the residuals will typically be normal around the regression line in the transformed space, and this will allow us to proceed with kriging. The predicted values can finally be back-transformed to the original scale using the inverse of the link function.

[3] Meaning symmetrically distributed around the feature space and the regression line.

The lesson learned is that each statistical spatial predictor comes with: (a) a conceptual model that explains the general relationship (e.g. Eq.2.1.10); (b) model-associated assumptions (e.g. zero mean estimation error, first or second order stationarity, normality of the residuals); (c) actual prediction formulas (Eq.2.1.15); and (d) a set of proofs that, under given assumptions, a prediction model is the BLUP. Ignoring the important model assumptions can lead to poor predictions, even though the output maps might appear to be visually fine.

2.1.2 Selecting the right spatial prediction technique

Knowing that the most of the linear spatial prediction models are more or less connected, we can start by testing the most generic technique, and then finish by using the most suitable technique for our own case study. Pebesma (2004, p.689), for example, implemented such a nested structure in his design of the gstat package. An user can switch between one and another technique by following a simple decision tree shown in Fig. 2.3.

First, we need to check if the deterministic model is defined already, if it has not been, we can try to correlate the sampled variables with environmental factors. If the environmental factors are significantly correlated, we can fit a multiple linear regression model (Eq.1.3.14) and then analyze the residuals for spatial autocorrelation. If the residuals show no spatial autocorrelation (pure nugget effect), we proceed with OLS estimation of the regression coefficients. Otherwise, if the residuals show spatial auto-correlation, we can run regression-kriging. If the data shows no correlation with environmental factors, then we can still analyze the variogram of the target variable. This time, we might also consider modeling the anisotropy. If we can fit a variogram different from pure nugget effect, then we can run ordinary kriging. Otherwise, if we can only fit a linear variogram, then we might just use some mechanical interpolator such as the inverse distance interpolation.

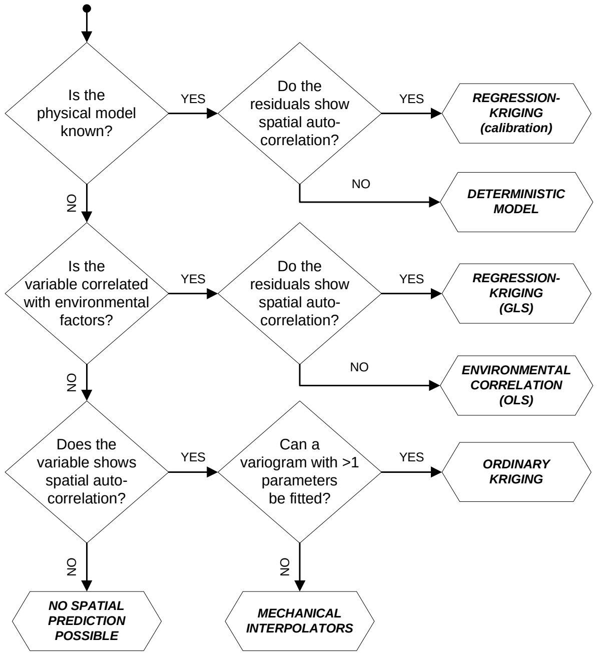

Is the physical model known?
YES
Do the residuals show spatial auto-correlation?
YES
REGRESSION-KRIGING (calibration)
NO
DETERMINISTIC MODEL
NO
Is the variable correlated with environmental factors?
YES
Do the residuals show spatial auto-correlation?
YES
REGRESSION-KRIGING (GLS)
NO
ENVIRONMENTAL CORRELATION (OLS)
NO
Does the variable shows spatial auto-correlation?
YES
Can a variogram with >1 parameters be fitted?
YES
ORDINARY KRIGING
NO
NO SPATIAL PREDICTION POSSIBLE
NO
MECHANICAL INTERPOLATORS

Fig. 2.3: A general decision tree for selecting the suitable spatial prediction model based on the results of model estimation. Similar decision tree is implemented in the gstat package.

If the variogram of the target variable shows no spatial auto-correlation, and no correlation with environmental factors, this practically means that the only statistically valid prediction model is to estimate a global mean for the whole area. Although this might frustrate you because it would lead to a nonsense map where each pixel shows the same value, you should be aware that even this is informative[4].

How does the selection of the spatial prediction model works in practice? In the gstat package, a user can easily switch from one to other prediction model by changing the arguments in the generic krige function in R (Fig. 1.13; see further §3.2). For example, if the name of the input field samples is meuse and the prediction locations (grid) is defined by meuse.grid, we can run the inverse distance interpolation (§1.2.1) by specifying (Pebesma, 2004):

```
> library(gstat)
> data(meuse)
> coordinates(meuse) <- ~ x+y
> data(meuse.grid)
> coordinates(meuse.grid) <- ~ x+y
```

[4] Sometimes an information that we are completely uncertain about a feature is better than a colorful but completely unreliable map.

```
> gridded(meuse.grid) <- TRUE
> zinc.id <- krige(zinc ~ 1, data=meuse, newdata=meuse.grid)
```

```
  [inverse distance weighted interpolation]
```

where zinc is the sampled environmental variable (vector) and zinc.id is the resulting raster map (shown in Fig. 1.13). Instead of using inverse distance interpolation we might also try to fit the values using the coordinates and a 2nd order polynomial model:

```
> zinc.ts <- krige(zinc ~ x+y+x*y+x*x+y*y, data=meuse, newdata=meuse.grid)
```

```
  [ordinary or weighted least squares prediction]
```

which can be converted to the moving surface fitting by adding a search window:

```
> zinc.mv <- krige(zinc ~ x+y+x*y+x*x+y*y, data=meuse, newdata=meuse.grid, nmax=20)
```

```
  [ordinary or weighted least squares prediction]
```

If we add a variogram model, then gstat will instead of running inverse distance interpolation run ordinary kriging (§1.3.1):

```
> zinc.ok <- krige(log1p(zinc) ~ 1, data=meuse, newdata=meuse.grid,
+             model=vgm(psill=0.714, "Exp", range=449, nugget=0))
```

```
  [using ordinary kriging]
```

where vgm(psill=0.714, "Exp", range=449, nugget=0) is the Exponential variogram model with a sill parameter of 0.714, range parameter of 449 m and the nugget parameter of 0 (the target variable was log-transformed). Likewise, if there were environmental factors significantly correlated with the target variable, we could run OLS regression (§1.3.2) by omitting the variogram model:

```
> zinc.ec <- krige(log1p(zinc) ~ dist+ahn, data=meuse, newdata=meuse.grid)
```

```
  [ordinary or weighted least squares prediction]
```

where dist and ahn are the environmental factor used as predictors (raster maps), which are available as separate layers within the spatial layer[5] meuse.grid. If the residuals do show spatial auto-correlation, then we can switch to universal kriging (Eq.2.1.4) by adding the variogram:

```
> zinc.rk <- krige(log1p(zinc) ~ dist+ahn, data=meuse, newdata=meuse.grid,
+             model=vgm(psill=0.151, "Exp", range=374, nugget=0.055))
```

```
  [using universal kriging]
```

If the model between the environmental factors and our target variable is deterministic, then we can use the point samples to calibrate our predictions. The R command would then look something like this:

```
> zinc.rkc <- krige(zinc ~ zinc.df, data=meuse, newdata=meuse.grid,
+              model=vgm(psill=3, "Exp", range=500, nugget=0))
```

```
  [using universal kriging]
```

where zinc.df are the values of the target variable estimated using a deterministic function.

In gstat, a user can also easily switch from estimation to simulations (§2.4) by adding to the command above an additional argument: nsim=1. This will generate Sequential Gaussian Simulations using the same prediction model. Multiple simulations can be generated by increasing the number set for this argument. In addition, a user can switch from block predictions by adding argument block=100; and from global estimation of weights by adding a search radius or maximum number of pairs, e.g. radius=1000 or nmax=60.

By using the automap[6] package one needs to specify even less arguments. For example, the command:

[5] In R a SpatialGridDataframe object.

[6] http://cran.r-project.org/web/packages/automap/

```
> zinc.rk <- autoKrige(log1p(zinc) ~ dist, data=meuse, newdata=meuse.grid)

  [using universal kriging]
```

will do much of the standard geostatistical analysis without any intervention from the user: it will filter the duplicate points where needed, estimate the residuals, then fit the variogram for the residuals, and generate the predictions at new locations. The results can be plotted in a single page in a form of a report. Such generic commands can significantly speed up data processing, and make it easier for a non-geostatistician to generate maps (see further section 2.10.3).

In the intamap package[7], one needs to set even less parameters to generate predictions from a variety of methods:

```
> meuse$value <- log(meuse$zinc)
> output <- interpolate(data=meuse, newdata=meuse.grid)

  R 2009-11-11 17:09:14 interpolating 155 observations, 3103 prediction locations
  [Time models loaded...]
  [1] "estimated time for  copula 133.479866956255"
  Checking object ... OK
```

which gives the (presumably) best interpolation method[8] for the current problem (value column), given the time available set with maximumTime.

A more systematic strategy to select the right spatial prediction technique is to use objective criteria of mapping success (i.e. *a posteriori* criteria). From the application point of view, it can be said that there are (only) five relevant criteria to evaluate various spatial predictors (see also §1.4):

(1.) the **overall mapping accuracy**, e.g. standardized RMSE at control points — the amount of variation explained by the predictor expressed in %;

(2.) the **bias**, e.g. mean error — the accuracy of estimating the central population parameters;

(3.) the **model robustness**, also known as *model sensitivity* — in how many situations would the algorithm completely fail / how much artifacts does it produces?;

(4.) the **model reliability** — how good is the model in estimating the prediction error (how accurate is the prediction variance considering the true mapping accuracy)?;

(5.) the **computational burden** — the time needed to complete predictions;

From this five, you could derive a single composite measure that would then allow you to select '*the optimal*' predictor for any given data set, but this is not trivial! Hsing-Cheng and Chun-Shu (2007) suggest a framework to select the best predictor in an automated way, but this work would need to be much extended. In many cases we simply finish using some **naïve predictor** — that is predictor that we know has a statistically more optimal alternative[9], but this alternative is simply not practical.

The intamap decision tree, shown in Fig. 2.4, is an example of how the selection of the method can be automated to account for (1) anisotropy, (2) specified observation errors, and (3) extreme values. This is a specific application primarily developed to interpolate the radioactivity measurements from the European radiological data exchange platform, a network of around 4000 sensors. Because the radioactivity measurements can often carry local extreme values, robust techniques need to be used to account for such effects. For example, **Copula kriging**[10] methods can generate more accurate maps if extreme values are also present in the observations. The problem of using methods such as Copula kriging, however, is that they can often take even few hours to generate maps even for smaller areas. To minimize the risk of running into endless computing, the authors of the intamap decision tree have decided to select the prediction algorithm based on

[7] http://cran.r-project.org/web/packages/intamap/

[8] intamap automatically chooses between: (1) kriging, (2) copula methods, (3) inverse distance interpolation, projected spatial gaussian process methods in the psgp package, (4) transGaussian kriging or yamamoto interpolation.

[9] For example, instead of using the REML approach to variogram modeling, we could simply fit a variogram using weighted least squares (see §1.3.1), and ignore all consequences (Minasny and McBratney, 2007).

[10] Copula kriging is a sophistication of ordinary kriging; an iterative technique that splits the original data set and then re-estimates the model parameters with maximization of the corresponding likelihood function (Bárdossy and Li, 2008).

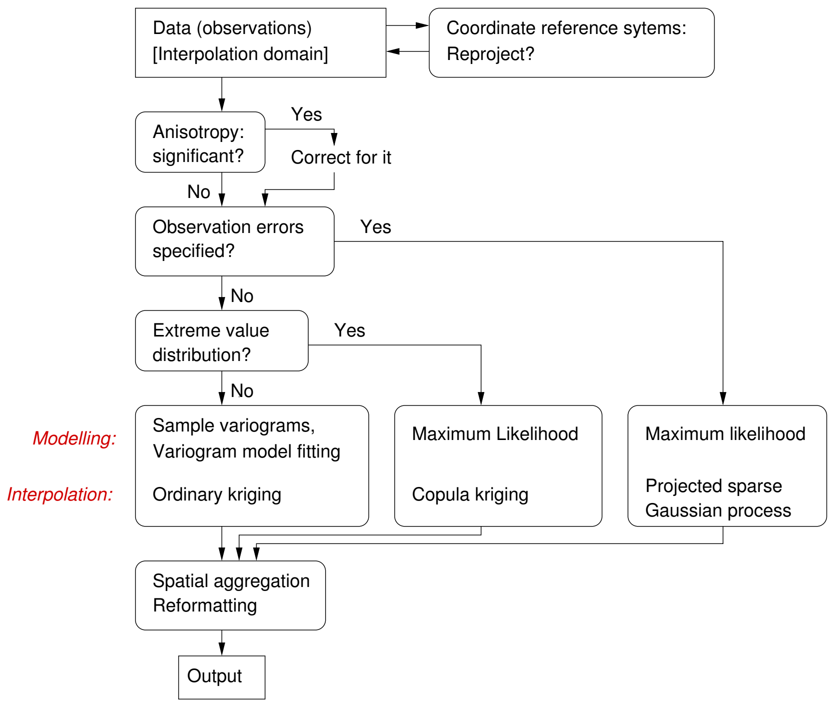


Fig. 2.4: Decision tree used in the intamap interpolation service for automated mapping. After Pebesma et al. (2009).

the computational time. Hence the system first estimates the approximate time needed to run the prediction using the most sophisticated technique; if this is above the threshold time, the system will switch to a more naïve method (Pebesma et al., 2009). As a rule of thumb, the authors of intamap suggest 30 seconds as the threshold time to accept automated generation of a map via a web-service.

2.1.3 The Best Combined Spatial Predictor

Assuming that a series of prediction techniques are mutually independent[11], predictions can be generated as a weighted average from multiple predictions i.e. by generating the Best Combined Spatial Prediction (**BCSP**):

$$\hat{z}_{\mathtt{BCSP}}(\mathbf{s}_0) = \frac{\hat{z}_{\mathtt{SP1}}(\mathbf{s}_0) \cdot \frac{1}{\hat{\sigma}_{\mathtt{SP1}}(\mathbf{s}_0)} + \hat{z}_{\mathtt{SP2}}(\mathbf{s}_0) \cdot \frac{1}{\hat{\sigma}_{\mathtt{SP2}}(\mathbf{s}_0)} + \ldots + \hat{z}_{\mathtt{SPj}}(\mathbf{s}_0) \cdot \frac{1}{\hat{\sigma}_{\mathtt{SPj}}(\mathbf{s}_0)}}{\sum\limits_{j=1}^{p} \frac{1}{\hat{\sigma}_{\mathtt{SPj}}(\mathbf{s}_0)}} \tag{2.1.19}$$

where $\sigma_{\mathtt{SPj}}(\mathbf{s}_0)$ is the prediction error estimated by the model (prediction variance), and p is the number of predictors. For example, we can generate a combined prediction using OK and e.g. GLM-regression and then sum-up the two maps (Fig. 2.5). The predictions will in some parts of the study are look more as OK, in others more as GLM, which actually depicts extrapolation areas of both methods. This map is very similar to predictions produced using regression-kriging (see further Fig. 5.9); in fact, one could probably mathematically prove that under ideal conditions (absolute stationarity of residuals; no spatial clustering; perfect linear relationship), BCSP predictions would equal the regression-kriging predictions. In general, the map in the middle of Fig. 2.5 looks more as the GLM-regression map because this map is about 2–3 times more precise than the OK map. It is important to emphasize that, in order to combine various predictors, we do need to have an estimate of the prediction uncertainty, otherwise we are not able to assign the weights (see further §7.5). In principle, linear combination of statistical techniques using the Eq.(2.1.19) above should be avoided if a theoretical basis exists that incorporates such combination.

[11] If they do not use the same model parameters; if they treat different parts of spatial variation etc.

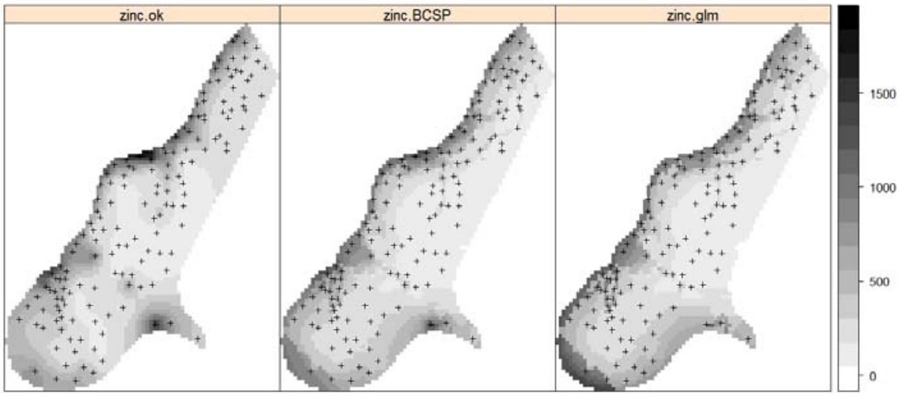


Fig. 2.5: Best Combined Spatial Predictor as weighted average of ordinary kriging (`zinc.ok`) and GLM regression (`zinc.glm`).

In the example above (GLM+OK), we assume that the predictions/prediction errors are independent, and they are probably not. In addition, a statistical theory exists that supports a combination of regression and kriging (see previously §2.1.1), so there is no need to run predictions separately and derive an unrealistic measure of model error. The BCSP can be only interesting for situations where there are indeed several objective predictors possible, where no theory exists that reflects their combination, and/or where fitting of individual models is faster and less troublesome than fitting of a hybrid model. For example, ordinary kriging can be speed-up by limiting the search radius, predictions using GLMs is also relatively inexpensive. External trend kriging using a GLM in geoRglm package might well be the statistically most robust technique you could possibly use, but it can also be beyond the computational power of your PC.

The combined prediction error of a BCSP can be estimated as the smallest prediction error achieved by any of the prediction models:

$$\hat{\sigma}_{\mathtt{BCSP}}(\mathbf{s_0}) = \min\left\{\hat{\sigma}_{\mathtt{SP1}}(\mathbf{s_0}),\ldots,\hat{\sigma}_{\mathtt{SPj}}(\mathbf{s_0})\right\} \tag{2.1.20}$$

which is really an *ad hoc* formula and should be used only to visualize and depict problematic areas (highest prediction error).

2.1.4 Universal kriging, kriging with external drift

The geostatistical literature uses many different terms for what are essentially the same or at least very similar techniques. This confuses the users and distracts them from using the right technique for their mapping projects. In this section, we will show that both universal kriging, kriging with external drift and regression-kriging are basically the same technique. Matheron (1969) originally termed the technique *Le krigeage universel*, however, the technique was intended as a generalized case of kriging where the trend is modeled as a function of coordinates. Thus, many authors (Deutsch and Journel, 1998; Wackernagel, 2003; Papritz and Stein, 1999) reserve the term *Universal Kriging* (UK) for the case when only the coordinates are used as predictors. If the deterministic part of variation (*drift*) is defined externally as a linear function of some explanatory variables, rather than the coordinates, the term *Kriging with External Drift* (KED) is preferred (Wackernagel, 2003; Chiles and Delfiner, 1999). In the case of UK or KED, the predictions are made as with kriging, with the difference that the covariance matrix of residuals is extended with the auxiliary predictors $q_k(\mathbf{s}_i)$'s (Webster and Oliver, 2001, p.183). However, the drift and residuals can also be estimated separately and then summed. This procedure was suggested by Ahmed and de Marsily (1987); Odeh et al. (1995) later named it *Regression-kriging*, while Goovaerts (1997, §5.4) uses the term *Kriging with a trend model* to refer to a family of predictors, and refers to RK as *Simple kriging with varying local means*. Although equivalent, KED and RK differ in the computational steps used.

Let us zoom into the two variants of regression-kriging. In the case of KED, predictions at new locations are made by:

$$\hat{z}_{\mathtt{KED}}(\mathbf{s}_0) = \sum_{i=1}^{n} w_i^{\mathtt{KED}}(\mathbf{s}_0) \cdot z(\mathbf{s}_i) \tag{2.1.21}$$

for

$$\sum_{i=1}^{n} w_i^{\mathtt{KED}}(\mathbf{s}_0) \cdot q_k(\mathbf{s}_i) = q_k(\mathbf{s}_0); \qquad k = 1, ..., p \tag{2.1.22}$$

or in matrix notation:

$$\hat{z}_{\mathtt{KED}}(\mathbf{s}_0) = \delta_{\mathbf{0}}^{\mathrm{T}} \cdot \mathbf{z} \tag{2.1.23}$$

where z is the target variable, q_k's are the predictor variables i.e. values at a new location ($\mathbf{s}_0$), $\delta_{\mathbf{0}}$ is the vector of KED weights ($w_i^{\mathtt{KED}}$), p is the number of predictors and $\mathbf{z}$ is the vector of n observations at primary locations. The KED weights are solved using the extended matrices:

$$\begin{aligned} \lambda_{\mathbf{0}}^{\mathtt{KED}} &= \left\{ w_1^{\mathtt{KED}}(\mathbf{s}_0), ..., w_n^{\mathtt{KED}}(\mathbf{s}_0), \varphi_0(\mathbf{s}_0), ..., \varphi_p(\mathbf{s}_0) \right\}^{\mathrm{T}} \\ &= \mathbf{C}^{\mathtt{KED}-1} \cdot \mathbf{c}_{\mathbf{0}}^{\mathtt{KED}} \end{aligned} \tag{2.1.24}$$

where $\lambda_{\mathbf{0}}^{\mathtt{KED}}$ is the vector of solved weights, φ_p are the Lagrange multipliers, $\mathbf{C}^{\mathtt{KED}}$ is the extended covariance matrix of residuals and $\mathbf{c}_{\mathbf{0}}^{\mathtt{KED}}$ is the extended vector of covariances at a new location.

In the case of KED, the extended covariance matrix of residuals looks like this (Webster and Oliver, 2001, p.183):

$$\mathbf{C}^{\mathtt{KED}} = \begin{bmatrix} C(\mathbf{s}_1,\mathbf{s}_1) & \cdots & C(\mathbf{s}_1,\mathbf{s}_n) & 1 & q_1(\mathbf{s}_1) & \cdots & q_p(\mathbf{s}_1) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ C(\mathbf{s}_n,\mathbf{s}_1) & \cdots & C(\mathbf{s}_n,\mathbf{s}_n) & 1 & q_1(\mathbf{s}_n) & \cdots & q_p(\mathbf{s}_n) \\ 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ q_1(\mathbf{s}_1) & \cdots & q_1(\mathbf{s}_n) & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & 0 & \vdots & & \vdots \\ q_p(\mathbf{s}_1) & \cdots & q_p(\mathbf{s}_n) & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.1.25}$$

and $\mathbf{c}_{\mathbf{0}}^{\mathtt{KED}}$ like this:

$$\mathbf{c}_{\mathbf{0}}^{\mathtt{KED}} = \left\{ C(\mathbf{s}_0,\mathbf{s}_1), ..., C(\mathbf{s}_0,\mathbf{s}_n), q_0(\mathbf{s}_0), q_1(\mathbf{s}_0), ..., q_p(\mathbf{s}_0) \right\}^{\mathrm{T}}; \quad q_0(\mathbf{s}_0) = 1 \tag{2.1.26}$$

Hence, KED looks exactly as ordinary kriging (Eq.1.3.2), except the covariance matrix and vector are extended with values of auxiliary predictors.

In the case of RK, the predictions are made separately for the drift and residuals and then added back together (Eq.2.1.4):

$$\hat{z}_{\mathtt{RK}}(\mathbf{s}_0) = \mathbf{q}_0^{\mathrm{T}} \cdot \hat{\beta}_{\mathtt{GLS}} + \lambda_0^{\mathrm{T}} \cdot \mathbf{e}$$

It can be demonstrated that both KED and RK algorithms give exactly the same results (Stein, 1999; Hengl et al., 2007a). Start from KED where the predictions are made as in ordinary kriging using $\hat{z}_{\mathtt{KED}}(\mathbf{s}_0) = \lambda_{\mathtt{KED}}^{\mathrm{T}} \cdot \mathbf{z}$. The KED kriging weights ($\lambda_{\mathtt{KED}}^{\mathrm{T}}$) are obtained by solving the system (Wackernagel, 2003, p.179):

$$\begin{bmatrix} \mathbf{C} & \mathbf{q} \\ \mathbf{q}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \lambda_{\mathtt{KED}} \\ \phi \end{bmatrix} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{q}_0 \end{bmatrix}$$

where ϕ is a vector of Lagrange multipliers. Writing this out yields:

$$\begin{aligned} \mathbf{C} \cdot \lambda_{\mathtt{KED}} + \mathbf{q} \cdot \phi &= \mathbf{c}_0 \\ \mathbf{q}^{\mathrm{T}} \cdot \lambda_{\mathtt{KED}} &= \mathbf{q}_0 \end{aligned} \tag{2.1.27}$$

from which follows:

$$\mathbf{q}^{\mathrm{T}} \cdot \lambda_{\mathtt{KED}} = \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{c}_0 - \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q} \cdot \phi \tag{2.1.28}$$

and hence:

$$\phi = \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{c}_0 - \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}_0 \tag{2.1.29}$$

where the identity $\mathbf{q}^{\mathrm{T}} \cdot \lambda_{\mathtt{KED}} = \mathbf{q}_0$ has been used. Substituting ϕ back into Eq. (2.1.27) shows that the KED weights equal (Papritz and Stein, 1999, p.94):

$$\begin{aligned} \lambda_{\mathtt{KED}} &= \mathbf{C}^{-1} \cdot \mathbf{c}_0 - \mathbf{C}^{-1} \cdot \mathbf{q} \cdot \left[\left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{c}_0 - \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}_0\right] \\ &= \mathbf{C}^{-1} \cdot \left[\mathbf{c}_0 + \mathbf{q} \cdot \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \mathbf{q} \cdot\right)^{-1} \cdot \left(\mathbf{q}_0 - \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{c}_0\right)\right] \end{aligned} \tag{2.1.30}$$

Let us now turn to RK. Recall from Eq.(2.1.3) that the GLS estimate for the vector of regression coefficients is given by:

$$\hat{\beta}_{\mathtt{GLS}} = \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{z} \tag{2.1.31}$$

and weights for residuals by:

$$\lambda_0^{\mathrm{T}} = \mathbf{c}_0^{\mathrm{T}} \cdot \mathbf{C}^{-1} \tag{2.1.32}$$

and substituting these in RK formula (Eq.2.1.4) gives:

$$\begin{aligned} &= \mathbf{q}_0^{\mathrm{T}} \cdot \hat{\beta}_{\mathtt{GLS}} + \lambda_0^{\mathrm{T}} \cdot (\mathbf{z} - \mathbf{q} \cdot \hat{\beta}_{\mathtt{GLS}}) \\ &= \left[\mathbf{q}_0^{\mathrm{T}} \cdot \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} + \mathbf{c}_0^{\mathrm{T}} \cdot \mathbf{C}^{-1} - \mathbf{c}_0^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q} \cdot \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1}\right] \cdot \mathbf{z} \\ &= \mathbf{C}^{-1} \cdot \left[\mathbf{c}_0^{\mathrm{T}} + \mathbf{q}_0^{\mathrm{T}} \cdot \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}} - \mathbf{c}_0^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q} \cdot \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \mathbf{q}\right)^{-1} \cdot \mathbf{q}^{\mathrm{T}}\right] \cdot \mathbf{z} \\ &= \mathbf{C}^{-1} \cdot \left[\mathbf{c}_0 + \mathbf{q} \cdot \left(\mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{q}\right)^{-1} \cdot \left(\mathbf{q}_0 - \mathbf{q}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \mathbf{c}_0\right)\right] \cdot \mathbf{z} \end{aligned} \tag{2.1.33}$$

The left part of the equation is equal to Eq.(2.1.30), which proves that KED will give the same predictions as RK if same inputs are used. A detailed comparison of RK and KED using the 5–points example in MS Excel is also available as supplementary material[12].

Although the KED seems, at first glance, to be computationally more straightforward than RK, the variogram parameters for KED must also be estimated from regression residuals, thus requiring a separate regression modeling step. This regression should be GLS because of the likely spatial correlation between residuals. Note that many analyst use instead the OLS residuals, which may not be too different from the GLS residuals (Hengl et al., 2007a; Minasny and McBratney, 2007). However, they are not optimal if there is any spatial correlation, and indeed they may be quite different for clustered sample points or if the number of samples is relatively small ($\ll$200).

A limitation of KED is the instability of the extended matrix in the case that the covariate does not vary smoothly in space (Goovaerts, 1997, p.195). RK has the advantage that it explicitly separates trend estimation from spatial prediction of residuals, allowing the use of arbitrarily-complex forms of regression, rather than the simple linear techniques that can be used with KED (Kanevski et al., 1997). In addition, it allows the separate interpretation of the two interpolated components. For these reasons the use of the term *regression-kriging* over *universal kriging* has been advocated by the author (Hengl et al., 2007a). The emphasis on regression is important also because fitting of the deterministic part of variation is often more beneficial for the quality of final maps than fitting of the stochastic part (residuals).

2.1.5 A simple example of regression-kriging

The next section illustrates how regression-kriging computations work and compares it to ordinary kriging using the textbook example from Burrough and McDonnell (1998, p.139-141), in which five measurements are used to predict a value of the target variable (z) at an unvisited location ($\mathbf{s}_0$) (Fig. 2.6a). We extend this example by adding a hypothetical explanatory data source: a raster image of 10×10 pixels (Fig. 2.6b), which has been constructed to show a strong negative correlation with the target variable at the sample points.

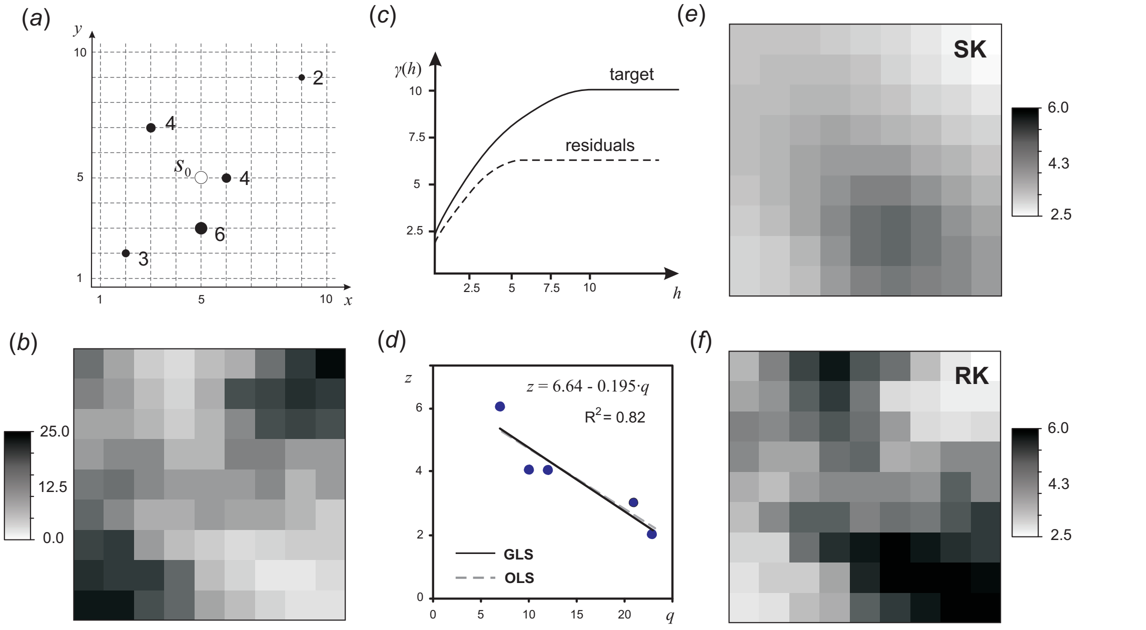


Fig. 2.6: Comparison of ordinary kriging and regression-kriging using a simple example with 5 points (Burrough and McDonnell, 1998, p.139–141): (a) location of the points and unvisited site; (b) values of the covariate q; (c) variogram for target and residuals, (d) OLS and GLS estimates of the regression model and results of prediction for a 10×10 grid using ordinary kriging (e) and regression-kriging (f). Note how the RK maps reflects the pattern of the covariate.

The RK predictions are computed as follows:

[12] http://spatial-analyst.net/book/RK5points

(1.) **Determine a linear model** of the variable as predicted by the auxiliary map q. In this case the correlation is high and negative with OLS coefficients b_0=6.64 and b_1=-0.195 (Fig. 2.6d).

(2.) **Derive the OLS residuals** at all sample locations as:

$$e^*(\mathbf{s}_i) = z(\mathbf{s}_i) - \left[b_0 + b_1 \cdot q(\mathbf{s}_i)\right] \tag{2.1.34}$$

For example, the point at (x=9, y=9) with z=2 has a prediction of $6.64 - 0.195 \cdot 23 = 1.836$, resulting in an OLS residual of $e^* = -0.164$.

(3.) **Model the covariance structure of the OLS residuals**. In this example the number of points is far too small to estimate the autocorrelation function, so we follow the original text in using a hypothetical variogram of the target variable (spherical model, nugget C_0=2.5, sill C_1=7.5 and range R=10) and residuals (spherical model, C_0=2, C_1=4.5, R=5). The residual model is derived from the target variable model of the text by assuming that the residual variogram has approximately the same form and nugget but a somewhat smaller sill and range (Fig. 2.6c), which is often found in practice (Hengl et al., 2004a).

(4.) **Estimate the GLS coefficients** using Eq.(2.1.3). In this case we get just slightly different coefficients b_0=6.68 and b_1=-0.199. The GLS coefficients will not differ much from the OLS coefficients as long there is no significant clustering of the sampling locations (Fig. 2.6d) as in this case.

(5.) **Derive the GLS residuals at all sample locations**:

$$e^{**}(\mathbf{s}_i) = z(\mathbf{s}_i) - \left[b_0 + b_1 \cdot q(\mathbf{s}_i)\right] \tag{2.1.35}$$

Note that the b now refer to the GLS coefficients.

(6.) **Model the covariance structure of the GLS residuals** as a variogram. In practice this will hardly differ from the covariance structure of the OLS residuals.

(7.) **Interpolate the GLS residuals using ordinary kriging** (OK) using the modeled variogram[13]. In this case at the unvisited point location $(5,5)$ the interpolated residual is -0.081.

(8.) **Add the GLS surface to the interpolated GLS residuals** at each prediction point. At the unvisited point location $(5,5)$ the explanatory variable has a value 12, so that the prediction is then:

$$\begin{aligned}\hat{z}(5,5) &= b_0 + b_1 \cdot q_i + \sum_{i=1}^{n} \lambda_i(\mathbf{s}_0) \cdot e(\mathbf{s}_i) \\ &= 6.68 - 0.199 \cdot 12 - 0.081 = 4.21\end{aligned} \tag{2.1.36}$$

which is, in this specific case, a slightly different result than that derived by OK with the hypothetical variogram of the target variable ($\hat{z}$=4.30).

The results of OK (Fig. 2.6e) and RK (Fig. 2.6f) over the entire spatial field are quite different in this case, because of the strong relation between the covariate and the samples. In the case of RK, most of variation in the target variable (82%) has been accounted for by the predictor. Unfortunately, this version of RK has not been implemented in any software package yet[14] (see further §3.4.3). Another interesting issue is that most of the software in use (gstat, SAGA) does not estimate variogram using the GLS estimation of the residuals, but only of the OLS residuals (0 iterations). Again, for most of balanced and well spread sample sets, this will not cause any significant problems (Minasny and McBratney, 2007).

[13] Some authors argue whether one should interpolate residuals using simple kriging with zero expected mean of the residuals (by definition) or by ordinary kriging. In the case of OLS estimation, there is no difference; otherwise one should always use OK to avoid making biased estimates.

[14] Almost all geostatistical packages implement the KED algorithm because it is mathematically more elegant and hence easier to program.

2.2 Local versus localized models

In many geostatistical packages, a user can opt to limit the selection of points to determine the kriging weights by setting up a maximum distance and/or minimum and maximum number of point pairs (e.g. take only the closest 50 points). This way, the calculation of the new map can be significantly speed up. In fact, kriging in global neighborhood where $n \gg 1000$ becomes cumbersome because of computation of $\mathbf{C}^{-1}$ (Eq.1.3.5). Recall from §1.3.1 that the importance of points (in the case of ordinary kriging and assuming a standard initial variogram model) exponentially decreases with their distance from the point of interest. Typically, geostatisticians suggest that already first 30–60 closest points will be good enough to obtain stable predictions.

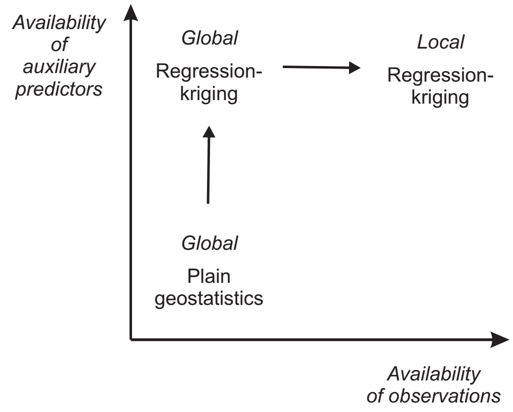
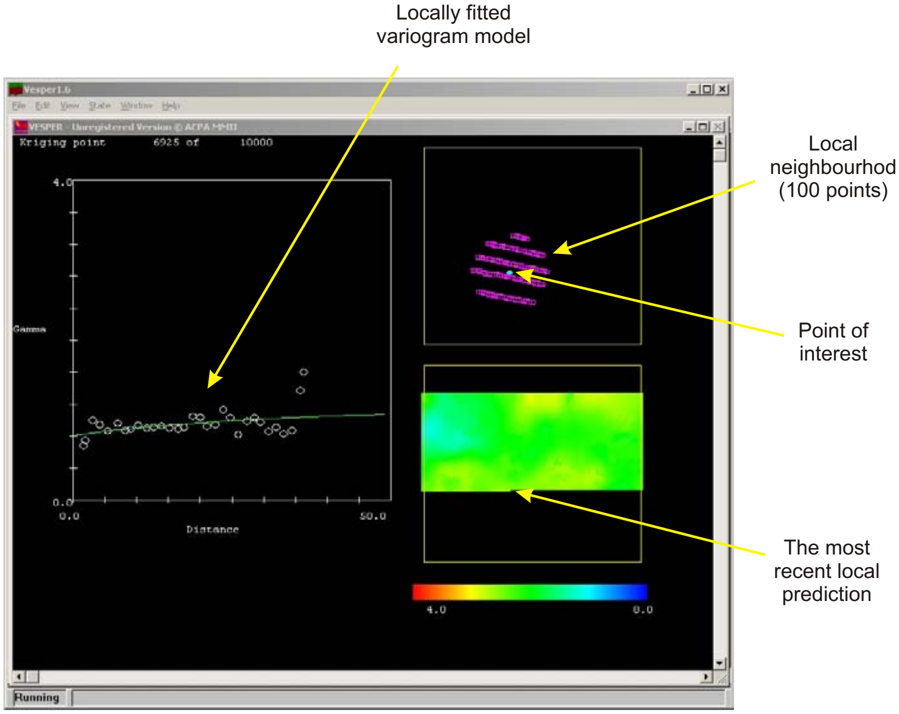


Fig. 2.7: Local regression-kriging is a further sophistication of regression-kriging. It will largely depend on the availability of explanatory and field data.

A prediction model where the search radius for derivation of kriging weights (Eq.1.3.4) is limited to a local neighborhood can be termed **localized prediction model**. There is a significant difference between *localized* and *local* prediction model, which often confuses inexperienced users. For example, if we set a search radius to re-estimate the variogram model, then we speak about a **local prediction model**, also known as the **moving window kriging** or kriging using local variograms (Haas, 1990; Walter et al., 2001; Lloyd, 2009). The local prediction model assumes that the variograms (and regression models) are non-stationary, i.e. that they need to be estimated locally.


Locally fitted variogram model
Vesper1.6
File Edit View State Window Help
VESPER - Unregistered Version © ACPA MMIII
4.0
Gamma
0.0
0.0
50.0
Distance
4.0
8.0
Running
Local neighbourhood (100 points)
Point of interest
The most recent local prediction

```
Kriging point        6925 of        10000
```

Fig. 2.8: Local variogram modeling and local ordinary kriging using a moving window algorithm in Vesper: a user can visually observe how the variograms change locally. Courtesy of Budiman Minasny.

While localized prediction models are usually just a computational trick to speed up the calculations, local prediction models are computationally much more demanding. Typically, they need to allow automated variogram modeling and filtering of improbable models to prevent artifacts in the final outputs. A result of local prediction model (e.g. moving window variogram modeling) are not only maps of predictions, but also spatial distribution of the fitted variogram parameters (Fig. 2.7). This way we can observe how does the nugget variation changes locally, which parts of the area are smooth and which are noisy etc. Typically, local variogram modeling and prediction make sense only when we work with large point data sets (e.g. $\gg$1000 of field observations), which is still not easy to find. In addition, local variogram modeling is not implemented in many packages. In fact, the author is aware of only one: Vesper[15] (Fig. 2.8).

In the case of regression-kriging, we could also run both localized and local models. This way we will not only produce maps of variogram parameters but we would also be able to map the regression coefficients[16]. In the case of kriging with external drift, some users assume that the same variogram model can be used in various parts of the study area and limit the search window to speed up the calculations[17]. This is obviously a simplification, because in the case of KED both regression and kriging part of predictions are solved at the same time. Hence, if we limit the search window, but keep a constant variogram model, we could obtain very different predictions then if we use the global (regression-kriging) model. Only if the variogram of residuals if *absolutely* stationary, then we can limit the search window to fit the KED weights. In practice, either global (constant variogram) or local prediction models (locally estimated regression models and variograms of residuals) should be used for KED model fitting.

2.3 Spatial prediction of categorical variables

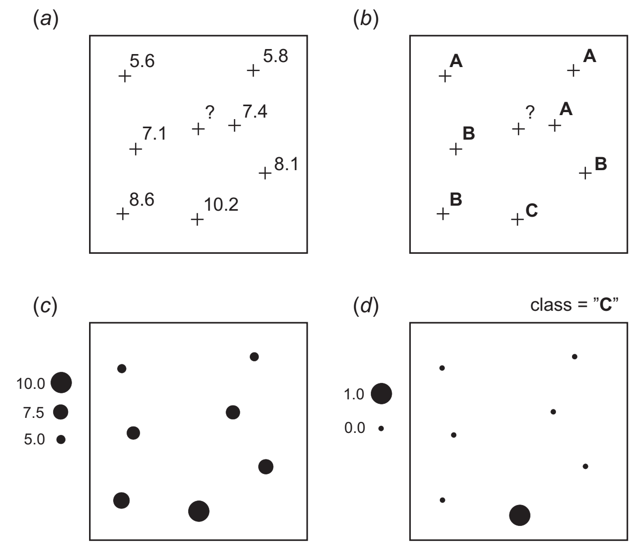


Fig. 2.9: Difficulties of predicting point-class data (b) and (d), as compared to quantitative variables (a) and (c), is that the class-interpolators are typically more complex and computationally more time-consuming.

Although geostatistics is primarily intended for use with continuous environmental variables, it can also be used to predict various types of categorical or class-type variables. Geostatistical analysis of categorical variables is by many referred to as the **indicator geostatistics** (Bierkens and Burrough, 1993). In practice, indicator kriging leads to many computational problems, which probably explains why there are not many operational applications of geostatistical mapping of categorical variables in the world (Hession et al., 2006). For example, it will typically be difficult to fit variogram for less frequent classes that occur at isolated locations (Fig. 2.9d).

Statistical grounds of indicator geostatistics has been recently reviewed by Papritz et al. (2005); Papritz (2009) who recognizes several conceptual difficulties of working with indicator data: (1) inconsistent modeling of indicator variograms, and (2) use of global variogram leads to biased predictions because the residuals are by definition non-stationary. Any attempt to use indicator kriging for data with an apparent trend either explicitly or implicitly by using ordinary indicator kriging within a local neighborhood requires the modeling of non-stationary indicator variograms to preserve the mean square optimality of kriging (Papritz, 2009). Indicator regression-kriging without any transformation has also been criticized because the model (binomial variable) suggests that residuals have mean-dependent variance ($p \cdot (1-p)$), and thus using a single variogram for the full set of residuals is not in accordance with theory.

Let us denote the field observations of a class-type variable as $z_c(\mathbf{s}_1), z_c(\mathbf{s}_2), ..., z_c(\mathbf{s}_n)$, where $c_1, c_2, ..., c_k$ are discrete categories (or states) and k is the total number of classes. A technique that estimates the soil-classes

[15] `http://www.usyd.edu.au/su/agric/acpa/vesper/vesper.html`

[16] Regression coefficients are often mapped with geographically weighted regression (Griffith, 2008).

[17] Software such as gstat and SAGA allow users to limit the search radius; geoR does not allow this flexibility.

at new unvisited location $\hat{z}_c(\mathbf{s}_0)$, given the input point data set $(z_c(\mathbf{s}_1), z_c(\mathbf{s}_2), ..., z_c(\mathbf{s}_n))$, can then be named a class-type interpolator. If spatially exhaustive predictors $q_1, q_2, ..., q_p$ (where p is the number of predictors) are available, they can be used to map each category over the area of interest. So far, there is a limited number of techniques that can achieve this:

Multi-indicator co-kriging — The simple multi-indicator kriging can also be extended to a case where several covariates are used to improve the predictions. This technique is known by the name *indicator* (soft) *co-kriging* (Journel, 1986). Although the mathematical theory is well explained (Bierkens and Burrough, 1993; Goovaerts, 1997; Pardo-Iguzquiza and Dowd, 2005), the application is cumbersome because of the need to fit a very large number of cross-covariance functions.

Multinomial Log-linear regression — This a generalization of logistic regression for situations when there are multiple classes of a target variable (Venables and Ripley, 2002). Each class gets a separate set of regression coefficients ($\beta_{\mathbf{c}}$). Because the observed values equal either 0 or 1, the regression coefficients need to be solved through a maximum likelihood iterative algorithm (Bailey et al., 2003), which makes the whole method somewhat more computationally demanding than simple multiple regression. An example of multinomial regression is given further in section 9.6.

Regression-kriging of indicators — One approach to interpolate soil categorical variables is to first assign memberships to point observations and then to interpolate each membership separately. This approach was first elaborated by de Gruijter et al. (1997) and then applied by Bragato (2004) and Triantafilis et al. (2001). An alternative is to first map cheap, yet descriptive, diagnostic distances and then classify these per pixel in a GIS (Carré and Girard, 2002).

In the case of logistic regression, the odds to observe a class (c) at new locations are computed as:

$$\hat{z}_c^+(\mathbf{s}_0) = \left[1 + \exp\left(-\beta_{\mathbf{c}}{}^{\mathbf{T}} \cdot \mathbf{q_0}\right)\right]^{-1}; \;\; c = 1, 2, .., k \tag{2.3.1}$$

where $\hat{z}_c^+(\mathbf{s}_0)$ are the estimated odds for class (c) at a new location s_0 and k is the number of classes. The multinomial logistic regression can also be extended to regression-kriging (for a complete derivation see Hengl et al. (2007b)). This means that the regression modeling is supplemented with the modeling of variograms for regression residuals, which can then be interpolated and added back to the regression estimate. So the predictions are obtained using:

$$\hat{z}_c^+(\mathbf{s}_0) = \left[1 + \exp\left(-\beta_{\mathbf{c}}{}^{\mathbf{T}} \cdot \mathbf{q_0}\right)\right]^{-1} + \hat{e}_c^+(\mathbf{s}_0) \tag{2.3.2}$$

where $\hat{e}_c^+$ are the interpolated residuals. The extension from multinomial regression to regression-kriging is not as simple as it seems. This is because the estimated values at new locations in Eq.(2.3.2) are constrained within the indicator range, which means that interpolation of residuals might lead to values outside the physical range ($<$0 or $>$1)[18]. One solution to this problem is to predict the trend part in transformed space, then interpolate residuals, sum the trend and residual part and back-transform the values (see §5.4).

Hengl et al. (2007b) show that memberships (μ_c), instead of indicators, are more suitable both for regression and geostatistical modeling, which has been also confirmed by several other authors (McBratney et al., 1992; de Gruijter et al., 1997; Triantafilis et al., 2001). Memberships can be directly linearized using the logit transformation:

$$\mu_c^+ = \ln\left(\frac{\mu_c}{1 - \mu_c}\right); \qquad 0 < \mu_c < 1 \tag{2.3.3}$$

where μ_c are the membership values used as input to interpolation. Then, all fitted values will be within the physical range (0–1). The predictions of memberships for class c at new locations are then obtained using the standard regression-kriging model (Eq.2.1.4):

$$\hat{\mu}_c^+(\mathbf{s}_0) = \mathbf{q}_\mathbf{0}^{\mathbf{T}} \cdot \hat{\beta}_{c,\mathtt{GLS}} + \lambda_{\mathbf{c},\mathbf{0}}^{\mathbf{T}} \cdot \left(\mu_{\mathbf{c}}^+ - \mathbf{q} \cdot \hat{\beta}_{c,\mathtt{GLS}}\right) \tag{2.3.4}$$

[18] The degree to which they will fall outside the 0–1 range is controlled by the variogram and amount of extrapolation in feature space

The interpolated values can then be back-transformed to the membership range using (Neter et al., 1996):

$$\hat{\mu}_c(\mathbf{s}_0) = \frac{e^{\hat{\mu}_c^{+}(\mathbf{s}_0)}}{1+e^{\hat{\mu}_c^{+}(\mathbf{s}_0)}} \tag{2.3.5}$$

In the case of regression-kriging of memberships, both spatial dependence and correlation with the predictors are modeled in a statistically sophisticated way. In addition, regression-kriging of memberships allows fitting of each class separately, which facilitates the understanding of the distribution of soil variables and the identification of problematic classes, i.e. classes which are not correlated with the predictors or do not show any spatial autocorrelation etc.

Spatial prediction of memberships can be excessive in computation time. Another problem is that, if the interpolated classes (odds, memberships) are fitted only by using the sampled data, the predictions of the odds/memberships will commonly not sum to unity at new locations. In this case, one needs to standardize values for each grid node by diving the original values by the sum of odds/memberships to ensure that they sum to unity, which is an *ad-hoc* solution. An algorithm, such as compositional regression-kriging[19] will need to be developed.

A number of alternative hybrid class-interpolators exists, e.g. the Bayesian Maximum Entropy (BME) approach by D'Or and Bogaert (2005). Another option is to use Markov-chain algorithms (Li et al., 2004, 2005a). However, note that although use of the BME and Markov-chain type of algorithms is a promising development, their computational complexity makes it still far from use in operational mapping.

2.4 Geostatistical simulations

Regression-kriging can also be used to generate simulations of a target variable using the same inputs as in the case of spatial prediction system. An equiprobable realization of an environmental variable can be generated by using the sampled values and their variogram model:

$$Z^{(\mathtt{SIM})}(\mathbf{s}_0) = E\left\{Z|z(\mathbf{s}_j),\gamma(\mathbf{h})\right\} \tag{2.4.1}$$

where $Z^{(\mathtt{SIM})}$ is the simulated value at the new location. The most common technique in geostatistics that can be used to generate equiprobable realizations is the **Sequential Gaussian Simulation** (Goovaerts, 1997, p.380-392). It starts by defining a random path for visiting each node of the grid once. At first node, kriging is used to determine the location-specific mean and variance of the conditional cumulative distribution function. A simulated value can then be drawn by using the inverse normal distribution (Box and Muller, 1958; Banks, 1998):

$$z_i^{\mathtt{SIM}} = \hat{z}_i + \hat{\sigma}_i \cdot \sqrt{-2\cdot\ln(1-A)}\cdot\cos(2\cdot\pi\cdot B) \tag{2.4.2}$$

where $z_i^{\mathtt{SIM}}$ is the simulated value of the target variable with induced error, A and B are the independent random numbers within the $0-0.99\ldots$ range, $\hat{z}_i$ is the estimated value at ith location, and $\hat{\sigma}_i$ is the regression-kriging error. The simulated value is then added to the original data set and the procedure is repeated until all nodes have been visited. Geostatistical simulations are used in many different fields to generate multiple realizations of the same feature (Heuvelink, 1998; Kyriakidis et al., 1999), or to generate realistic visualizations of a natural phenomena (Hengl and Toomanian, 2006; Pebesma et al., 2007). Examples of how to generate geostatistical simulations and use them to estimate the propagated error are further shown in section 10.3.2.

2.5 Spatio-temporal regression-kriging

In statistics, temporal processes (time series analysis, longitudinal data analysis) are well-known, but mixed spatio-temporal processes are still rather experimental (Banerjee et al., 2004). The 2D space models can be

[19] Walvoort and de Gruijter (2001), for example, already developed a compositional solution for ordinary kriging that will enforce estimated values to sum to unity at all locations.

extended to the time domain, which leads to **spatio-temporal geostatistics** (Kyriakidis and Journel, 1999). The universal kriging model (Eq.2.1.1) then modifies to:

$$Z(\mathbf{s},t) = m(\mathbf{s},t) + \varepsilon'(\mathbf{s},t) + \varepsilon'' \tag{2.5.1}$$

where $\varepsilon'(\mathbf{s},t)$ is the spatio-temporally autocorrelated residual for every $(\mathbf{s},t) \in S \times T$, while $m(\mathbf{s},t)$, the deterministic component of the model, can be estimated using e.g. (Fassó and Cameletti, 2009):

$$m(\mathbf{s},t) = \mathbf{q}(\mathbf{s},t) \cdot \beta + \mathbf{K}(s) \cdot \mathbf{y}_t + \omega(\mathbf{s},t) \tag{2.5.2}$$

where $\mathbf{q}$ is a matrix of covariates available at all $\mathbf{s}, t$ locations, $\mathbf{y_t}$ is a component of a target variable that is constant in space (global trend), $\mathbf{K}(s)$ is a matrix of coefficients, and $\omega(\mathbf{s},t)$ is the spatial small-scale component (white noise in time) correlated over space.

A possible but tricky simplification of the space-time models is to consider time to be third dimension of space. In that case, spatio-temporal interpolation follows the same interpolation principle as explained in Eq.(1.1.2), except that here the variograms are estimated in three dimensions (two-dimensional position x and y and '*position*' in time). From the mathematical aspect, the extension from the static 2D interpolation to the 3D interpolation is then rather simple. Regression modeling can be simply extended to a space-time model by adding time as a predictor. For example, a spatio-temporal regression model for interpolation of land surface temperature (see further §2.9.2) would look like this:

$$\begin{aligned} LST(\mathbf{s}_0,t_0) = b_0 &+ b_1 \cdot DEM(\mathbf{s}_0) + b_2 \cdot LAT(\mathbf{s}_0) + b_3 \cdot DISTC(\mathbf{s}_0) + b_4 \cdot LSR(\mathbf{s}_0,t_0) \\ &+ b_5 \cdot SOLAR(\mathbf{s}_0,t_0) + b_6 \cdot \cos\left([t_0 - \phi] \cdot \frac{\pi}{180} \right); \qquad \Delta t = 1 \text{ day} \end{aligned} \tag{2.5.3}$$

where *DEM* is the elevation map, *LAT* is the map showing distance from the equator, *DISTC* is the distance from the coast line, *LSR* is the land surface radiation from natural or man-made objects, *SOLAR* is the direct solar insolation for a given cumulative Julian day $t \in (0,+\infty)$, $\cos(t)$ is a generic function to account for seasonal variation of values and ϕ is the phase angle[20]. *DEM*, *LAT*, *DISTC* are temporally-constant predictors, while surface radiation and solar insolation maps need to be provided for each time interval used for data fitting.

The residuals from this regression model can then be analyzed for (spatio-temporal) auto-correlation. In `gstat`, extension from 2D to 3D variograms is possible by extending the variogram parameters: for 3D space-time variograms five values should be given in the form `anis = c(p,q,r,s,t)`, where `p` is the angle for the **principal direction of continuity** (measured in degrees, clockwise from y, in direction of x), `q` is the **dip angle** for the principal direction of continuity (measured in positive degrees up from horizontal), `r` is the third rotation angle to rotate the two minor directions

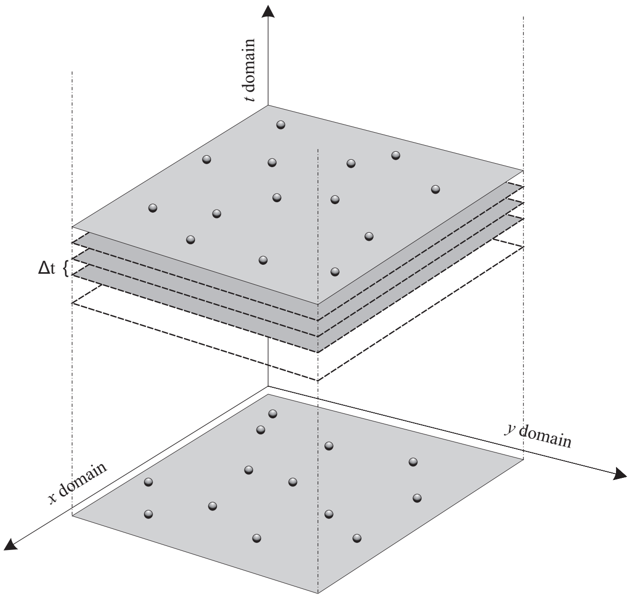


Fig. 2.10: Extension of a 2D prediction model to the space-time domain. Note that in the space-time cube, the amount of pixels needed to store the data exponentially increases as a function of: width × height × number of predictors × number of time intervals.

[20] A time delay from the coldest day.

around the principal direction defined by p and q[21] (see Fig. 1.11). A positive angle acts counter-clockwise while looking in the principal direction.

Once we have fitted the space-time variogram, we can run regression-kriging to estimate the values at 3D locations. In practice, we only wish to produce maps for a given time interval (t_0=constant), i.e. to produce 2D-slices of values in time (Fig. 2.10). Once we have produced a time-series of predictions, we can analyze the successive time periods and run various types of time-series analysis. This will help us detect temporal trends spatially and extract informative images about the dynamics of the feature of interest.

Note that, in order to yield accurate predictions using spatio-temporal techniques, dense sampling in both space and time is required. This means that existing natural resource surveys that have little to no repetition in time ($\ll$10 repetitions in time) cannot be adopted. Not to mention the computational complexity as the maps of predictors now multiply by the amount of time intervals. In addition, estimation of the spatio-temporal variograms will often be a cumbersome because we need to fit space-time models, for which we might not have enough space-time observations. A review of spatio-temporal models, i.e. dynamic linear state-space models, and some practical suggestions how to analyze such data and fit spatially varying coefficients can be followed in Banerjee et al. (2004, §8).

A specific extension of the general model from Eq.(2.5.1) is to estimate the deterministic part of variation by using process-based (simulation) models, which are often based on differential equations. In this case an environmental variable is predicted from a set of environmental predictors incorporated in a dynamic model (Eq.1.3.12):

$$Z(\mathbf{s}, t) = f_{s,c,r,p,a}(t) + \varepsilon'(\mathbf{s}, t) + \varepsilon'' \tag{2.5.4}$$

where s, c, r, p, a are the input (zero-stage) environmental conditions and f is a mathematical deterministic function that can be used to predict the values for a given space-time position. This can be connected with the Einstein's assumption that the Universe is in fact a trivial system that can be modeled and analyzed using "a one–dimensional differential equation — in which everything is a function of time"[22]. Some examples of operational soil-landscape process-based models are given by Minasny and McBratney (2001) and Schoorl et al. (2002). In vegetation science, for example, global modeling has proven to be very efficient for explanation of the actual distribution of vegetation and of global changes (Bonan et al., 2003). Integration of environmental process-based models will soon lead to development of a global dynamic model of environmental systems that would then provide solutions for different multipurpose national or continental systems.

Fassó and Cameletti (2009) recently proposed hierarchical models as a general approach for spatio-temporal problems, including dynamical mapping, and the analysis of the outputs from complex environmental modeling chains. The hierarchical models are a suitable solution to spatio-temporal modeling because they make it possible to define the joint dynamics and the full likelihood; the maximum likelihood estimation can be further simplified by using Expectation-Maximization algorithm. The basis of this approach is the classical two-stage hierarchical state-space model (Fassó and Cameletti, 2009):

$$Z_t = \mathbf{q}_t \cdot \beta + \mathbf{K} \cdot \mathbf{y}_t + \mathbf{e}_t \tag{2.5.5}$$

$$\mathbf{y}_t = \mathbf{G} \cdot \mathbf{y}_{t-1} + \eta_t \tag{2.5.6}$$

where $\mathbf{y}_t$ is modeled as the autoregressive process, $\mathbf{G}$ is the transition matrix and η_t is the innovation error. If all parameters are known, the unobserved temporal process $\mathbf{y}_t$ can be estimated for each time point t using e.g. *Kalman filter* or *Kalman smoother*. Such process-based spatio-temporal models are still experimental and it make take time until their semi-automated software implementations appear in R.

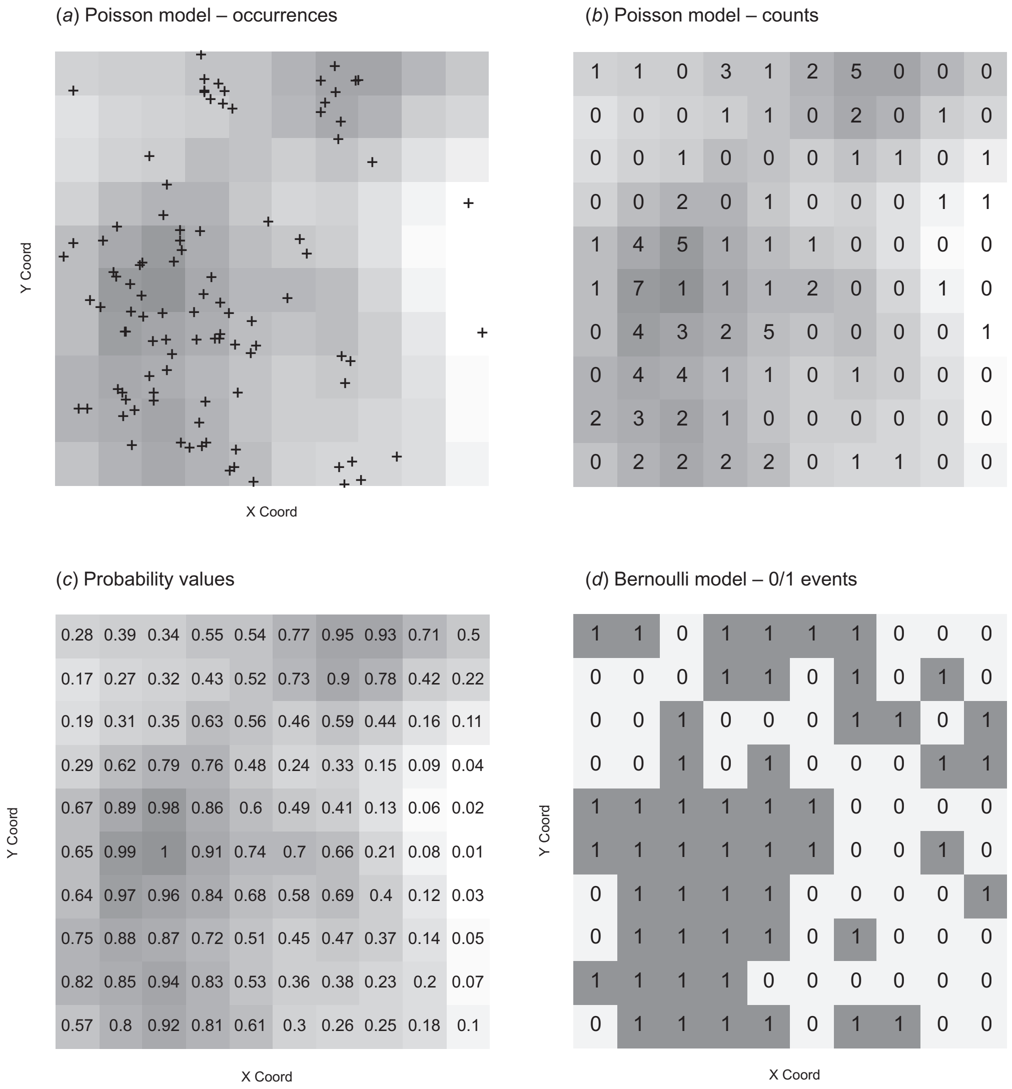
2.6 Species Distribution Modeling using regression-kriging

The key inputs to a Species Distribution Model (SDM) are: the inventory (population) of animals or plants consisting of a total of N individuals (a point pattern $\mathbf{X} = \{\mathbf{s}_i\}_1^N$; where $\mathbf{s}_i$ is a spatial location of individual

[21] `http://www.gstat.org/manual/node20.html`

[22] Quote by James Peebles, Princeton, 1990; published in "God's Equation: Einstein, Relativity, and the Expanding Universe" by Amir D. Aczel.

animal or plant; Fig. 1.3a), covering some area $B_{HR} \subset \mathbb{R}^2$ (where HR stands for home-range and $\mathbb{R}^2$ is the Euclidean space), and a list of environmental covariates/predictors $(q_1, q_2, \ldots q_p)$ that can be used to explain spatial distribution of a target species. In principle, there are two distinct groups of statistical techniques that can be used to map the realized species' distribution: (a) the point pattern analysis techniques, such as kernel smoothing, which aim at predicting density of a point process (Fig. 2.11a); and (b) statistical, GLM-based, techniques that aim at predicting the probability distribution of occurrences (Fig. 2.11c). Both approaches are explained in detail in the following sections.



(*a*) Poisson model – occurrences

(*b*) Poisson model – counts

1	1	0	3	1	2	5	0	0	0
0	0	0	1	1	0	2	0	1	0
0	0	1	0	0	0	1	1	0	1
0	0	2	0	1	0	0	0	1	1
1	4	5	1	1	1	0	0	0	0
1	7	1	1	1	2	0	0	1	0
0	4	3	2	5	0	0	0	0	1
0	4	4	1	1	0	1	0	0	0
2	3	2	1	0	0	0	0	0	0
0	2	2	2	2	0	1	1	0	0

(*c*) Probability values

0.28	0.39	0.34	0.55	0.54	0.77	0.95	0.93	0.71	0.5
0.17	0.27	0.32	0.43	0.52	0.73	0.9	0.78	0.42	0.22
0.19	0.31	0.35	0.63	0.56	0.46	0.59	0.44	0.16	0.11
0.29	0.62	0.79	0.76	0.48	0.24	0.33	0.15	0.09	0.04
0.67	0.89	0.98	0.86	0.6	0.49	0.41	0.13	0.06	0.02
0.65	0.99	1	0.91	0.74	0.7	0.66	0.21	0.08	0.01
0.64	0.97	0.96	0.84	0.68	0.58	0.69	0.4	0.12	0.03
0.75	0.88	0.87	0.72	0.51	0.45	0.47	0.37	0.14	0.05
0.82	0.85	0.94	0.83	0.53	0.36	0.38	0.23	0.2	0.07
0.57	0.8	0.92	0.81	0.61	0.3	0.26	0.25	0.18	0.1

(*d*) Bernoulli model – 0/1 events

1	1	0	1	1	1	1	0	0	0
0	0	0	1	1	0	1	0	1	0
0	0	1	0	0	0	1	1	0	1
0	0	1	0	1	0	0	0	1	1
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	1	0
0	1	1	1	1	0	0	0	0	1
0	1	1	1	1	0	1	0	0	0
1	1	1	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1	0	0

Fig. 2.11: Examples of (simulated) species distribution maps produced using common statistical models.

Species' density estimation using kernel smoothing and covariates

Spatial density (ν; if unscaled, also known as *"spatial intensity"*) of a point pattern (ignoring the time dimension) is estimated as:

$$\mathbb{E}\left[N(\mathbf{X} \cap B)\right] = \int_B \nu(\mathbf{s}) d\mathbf{s} \tag{2.6.1}$$

In practice, it can be estimated using e.g. a kernel estimator (Diggle, 2003; Baddeley, 2008):

$$\nu(\mathbf{s}) = \sum_{i=1}^{n} \kappa \cdot \left(\left\| \mathbf{s} - \mathbf{s}_i \right\| \right) \cdot b(\mathbf{s}) \tag{2.6.2}$$

where $\nu(\mathbf{s})$ is spatial density at location $\mathbf{s}$, $\kappa(\mathbf{s})$ is the kernel (an arbitrary probability density), $\mathbf{s}_i$ is location of an occurrence record, $\left\| \mathbf{s} - \mathbf{s}_i \right\|$ is the distance (norm) between an arbitrary location and observation location, and $b(\mathbf{s})$ is a border correction to account for missing observations that occur when $\mathbf{s}$ is close to the border of the region (Fig. 2.11a). A common (isotropic) kernel estimator is based on a Gaussian function with mean zero and variance 1:

$$\widehat{\nu}(\mathbf{s}) = \frac{1}{H^2} \cdot \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{\left\|\mathbf{s}-\mathbf{s}_i\right\|^2}{2}} \cdot b(\mathbf{s}) \tag{2.6.3}$$

The key parameter for kernel smoothing is the bandwidth (H) i.e. the smoothing parameter, which can be connected with the choice of variogram in geostatistics. The output of kernel smoothing is typically a map (raster image) consisting of M grid nodes, and showing spatial pattern of species' clustering.

Spatial density of a point pattern can also be modeled using a list of spatial covariates q's (in ecology, we call this environmental predictors), which need to be available over the whole area of interest B. For example, using a Poisson model (Baddeley, 2008):

$$\log \nu(\mathbf{s}) = \log \beta_0 + \log q_1(\mathbf{s}) + \ldots + \log q_p(\mathbf{s}) \tag{2.6.4}$$

where log transformation is used to account for the skewed distribution of both density values and covariates; p is the number of covariates. Models with covariates can be fitted to point patterns e.g. in the spatstat package[23]. Such point pattern–covariates analysis is commonly run only to determine i.e. test if the covariates are correlated with the feature of interest, to visualize the predicted trend function, and to inspect the spatial trends in residuals. Although statistically robust, point pattern–covariates models are typically not considered as a technique to improve prediction of species' distribution. Likewise, the model residuals are typically not used for interpolation purposes.

Predicting species' distribution using ENFA and GLM (pseudo-absences)

An alternative approach to spatial prediction of species' distribution using occurrence-only records and environmental covariates is the combination of ENFA and regression modeling. In general terms, predictions are based on fitting a GLM:

$$\mathbb{E}(\mathbf{P}) = \mu = g^{-1}(\mathbf{q} \cdot \beta) \tag{2.6.5}$$

where $\mathrm{E}(\mathbf{P})$ is the expected probability of species occurrence ($P \in [0, 1]$; Fig. 2.11c), $\mathbf{q}\cdot\beta$ is the linear regression model, and g is the link function. A common link function used for SDM with presence observations is the logit link function:

$$g(\mu) = \mu^{+} = \ln\left(\frac{\mu}{1-\mu}\right) \tag{2.6.6}$$

and the Eq.(2.6.5) becomes logistic regression (Kutner et al., 2004).

The problem of running regression analysis with occurrence-only observations is that we work with 1's only, which means that we cannot fit any model to such data. To account for this problem, species distribution modelers (see e.g. Engler et al. (2004); Jiménez-Valverde et al. (2008) and Chefaoui and Lobo (2008)) typically insert the so-called "*pseudo-absences*" — 0's simulated using a plausible models, such as **Environmental**

[23] This actually fits the maximum pseudolikelihood to a point process; for more details see Baddeley (2008).

Niche Factor Analysis (ENFA), MaxEnt or GARP (Guisan and Zimmermann, 2000), to depict areas where a species is not likely to occur. ENFA is a type of factor analysis that uses observed presences of a species to estimate which are the most favorable areas in the feature space, and then uses this information to predict the potential distribution of species for all locations (Hirzel and Guisan, 2002). The difference between ENFA and the Principal Component Analysis is that the ENFA factors have an ecological meaning. ENFA results in a Habitat Suitability Index (HSI$\in [0-100\%]$) — by depicting the areas of low HSI, we can estimate where the species is very unlikely to occur, and then simulate a new point pattern that can be added to the occurrence locations to produce a '*complete*' occurrences+absences data set. Once we have both 0's and 1's, we can fit a GLM as shown in Eq.(2.6.5) and generate predictions (probability of occurrence) using geostatistical techniques as described in e.g. Gotway and Stroup (1997).

Predicting species' density using ENFA and logistic regression-kriging

Point pattern analysis, ENFA and regression-kriging can be successfully combined using the approach explained in Hengl et al. (2009b). First, we will assume that our input point pattern represents only a sample of the whole population ($\mathbf{X}_S = \{\mathbf{s}_i\}_1^n$), so that the density estimation needs to be standardized to avoid biased estimates. Second, we will assume that pseudo-absences can be generated using both information about the potential habitat (HSI) and geographical location of the occurrence-only records. Finally, we focus on mapping the actual count of individuals over the grid nodes (realized distribution), instead of mapping the probability of species' occurrence.

Spatial density values estimated by kernel smoothing are primarily controlled by the bandwidth size (Bivand et al., 2008). The higher the bandwidth, the lower the values in the whole map; likewise, the higher the sampling intensity (n/N), the higher the spatial density, which eventually makes it difficult to physically interpret mapped values. To account for this problem, we propose to use relative density ($\nu_r : B \rightarrow [0,1]$) expressed as the ratio between the local and maximum density at all locations:

$$\nu_r(\mathbf{s}) = \frac{\nu(\mathbf{s})}{\max\{\nu(\mathbf{s})|\mathbf{s} \in B\}_1^M} \tag{2.6.7}$$

An advantage of using the relative density is that the values are in the range $[0,1]$, regardless of the bandwidth and sample size (n/N). Assuming that our sample $\mathbf{X}_S$ is representative and unbiased, it can be shown that $\nu_r(\mathbf{s})$ is an unbiased estimator of the true spatial density (see e.g. Diggle (2003) or Baddeley (2008)). In other words, regardless of the sample size, by using relative intensity we will always be able to produce an unbiased estimator of the spatial pattern of density for the whole population (see further Fig. 8.4).

Furthermore, assuming that we actually know the size of the whole population (N), by using predicted relative density, we can also estimate the actual spatial density (number of individuals per grid node; as shown in Fig. 2.11b):

$$\nu(\mathbf{s}) = \nu_r(\mathbf{s}) \cdot \frac{N}{\sum_{j=1}^{M} \nu_r(\mathbf{s})}; \qquad \sum_{j=1}^{M} \nu(\mathbf{s}) = N \tag{2.6.8}$$

which can be very useful if we wish to aggregate the species' distribution maps over some polygons of interest, e.g. to estimate the actual counts of individuals.

Our second concern is the insertion of pseudo-absences. Here, two questions arise: (1) how many pseudo-absences should we insert? and (b) where should we locate them? Intuitively, it makes sense to generate the same number of pseudo-absence locations as occurrences. This is also supported by the statistical theory of model-based designs, also known as "*D-designs*". For example, assuming a linear relationship between density and some predictor q, the optimal design that will minimize the prediction variance is to put half of observation at one extreme and other at other extreme. All D-designs are in fact symmetrical, and all advocate higher spreading in feature space (for more details about D-designs, see e.g. Montgomery (2005)), so this principle seems logical. After the insertion of the pseudo-absences, the extended observations data set is:

$$\mathbf{X}_\mathrm{f} = \left\{\{\mathbf{s}_i\}_1^n, \{\mathbf{s}^*{}_i\}_1^{n*}\right\}; \qquad n = n^* \tag{2.6.9}$$

where $\mathbf{s}^*_i$ are locations of the simulated pseudo-absences. This is not a point pattern any more because now also quantitative values — either relative densities ($\nu_r(\mathbf{s}_i)$) or indicator values — are attached to locations ($\mu(\mathbf{s}_i) = 1$ and $\mu(\mathbf{s}^*_i) = 0$).

The remaining issue is where and how to allocate the pseudo-absences? Assuming that a spreading of species in an area of interest is a function of the potential habitat and assuming that the occurrence locations on the HSI axis will commonly be skewed toward high values (see further Fig. 8.8 left; see also Chefaoui and Lobo (2008)), we can define the probability distribution (τ) to generate the pseudo-absence locations as e.g.:

$$\tau(\mathbf{s}^*) = \left[100\% - \text{HSI}(\mathbf{s})\right]^2 \tag{2.6.10}$$

where the square term is used to insure that there are progressively more pseudo-absences at the edge of low HSI. This way also the pseudo-absences will approximately follow Poisson distribution. In this paper we propose to extend this idea by considering location of occurrence points in geographical space also (see also an interesting discussion on the importance of geographic extent for generation of pseudo-absences by VanDerWal et al. (2009)). The Eq.(2.6.10) then modifies to:

$$\tau(\mathbf{s}^*) = \left[\frac{d_R(\mathbf{s}) + (100\% - \text{HSI}(\mathbf{s}))}{2}\right]^2 \tag{2.6.11}$$

where d_R is the normalized distance in the range $[0, 100\%]$, i.e. the distance from the observation points ($\mathbf{X}$) divided by the maximum distance. By using Eq.(2.6.11) to simulate the pseudo-absence locations, we will purposively locate them both geographically further away from the occurrence locations and in the areas of low HSI (unsuitable habitat).

After the insertion of pseudo-absences, we can attach to both occurrence-absence locations values of estimated relative density, and then correlate this with environmental predictors. This now becomes a standard geostatistical point data set, representative of the area of interest, and with quantitative values attached to point locations (see further Fig. 8.10d). Recall from Eq.(2.6.7) that we attach relative intensities to observation locations. Because these are bounded in the $[0, 1]$ range, we can use the logistic regression model to make predictions. Thus, the relative density at some new location ($\mathbf{s}_0$) can be estimated using:

$$\widehat{\nu}_r^+(\mathbf{s}_0) = \left[1 + \exp\left(-\beta^{\mathrm{T}} \cdot \mathbf{q_0}\right)\right]^{-1} \tag{2.6.12}$$

where β is a vector of fitted regression coefficients, $\mathbf{q_0}$ is a vector of predictors (maps) at a new location, and $\widehat{\nu}_r^+(\mathbf{s}_0)$ is the predicted logit-transformed value of the relative density. Assuming that the sampled intensities are continuous values in the range $\nu_r \in (0, 1)$, the model in Eq.(2.6.12) is in fact a liner model, which allows us to extend it to a more general linear geostatistical model such as regression-kriging. This means that the regression modeling is supplemented with the modeling of variograms for regression residuals, which can then be interpolated and added back to the regression estimate (Eq.2.1.4):

$$\widehat{\nu}_r^+(\mathbf{s}_0) = \mathbf{q}_\mathbf{0}^{\mathrm{T}} \cdot \hat{\beta}_{\mathtt{GLS}} + \delta_\mathbf{0}^{\mathrm{T}} \cdot \left(\nu_\mathbf{r}^+ - \mathbf{q} \cdot \hat{\beta}_{\mathtt{GLS}}\right) \tag{2.6.13}$$

where $\delta_\mathbf{0}$ is the vector of fitted weights to interpolate the residuals using ordinary kriging. In simple terms, logistic regression-kriging consists of five steps:

(1.) convert the relative intensities to logits using Eq.(2.6.6); if the input values are equal to 0/1, replace with the second smallest/highest value;

(2.) fit a linear regression model using Eq.(2.6.12);

(3.) fit a variogram for the residuals (logits);

(4.) produce predictions by first predicting the regression-part, then interpolate the residuals using ordinary kriging; finally add the two predicted trend-part and residuals together (Eq.2.6.13)

(5.) back-transform interpolated logits to the original $(0, 1)$ scale by:

$$\widehat{v}_r(\mathbf{s}_0) = \frac{e^{\widehat{v}_r^+(\mathbf{s}_0)}}{1+e^{\widehat{v}_r^+(\mathbf{s}_0)}} \tag{2.6.14}$$

After we have mapped relative density over area of interest, we can also estimate the actual counts using the Eq.(2.6.8). This procedure is further elaborated in detail in chapter 8.

2.7 Modeling of topography using regression-kriging

A **Digital Elevation Model** (DEM) is a digital representation of the land surface — the major input to quantitative analysis of topography, also known as Digital Terrain Analysis or Geomorphometry (Wilson and Gallant, 2000; Hengl and Reuter, 2008). Typically, a DEM is a raster map (an image or an elevation array) that, like many other spatial features, can be efficiently modeled using geostatistics. The geostatistical concepts were introduced in geomorphometry by Fisher (1998) and Wood and Fisher (1993), then further elaborated by Kyriakidis et al. (1999), Holmes et al. (2000) and Oksanen (2006). An important focus of using geostatistics to model topography is assessment of the errors in DEMs and analysis of effects that the DEM errors have on the results of spatial modeling. This is the principle of error propagation that commonly works as follows: simulations are generated from point-measured heights to produce multiple equiprobable realizations of a DEM of an area; a spatial model is applied *m* times and output maps then analyzed for mean values and standard deviations per pixel; the results of analysis can be used to quantify DEM accuracy and observe impacts of uncertain information in various parts of the study area (Hunter and Goodchild, 1997; Heuvelink, 1998; Temme et al., 2008).

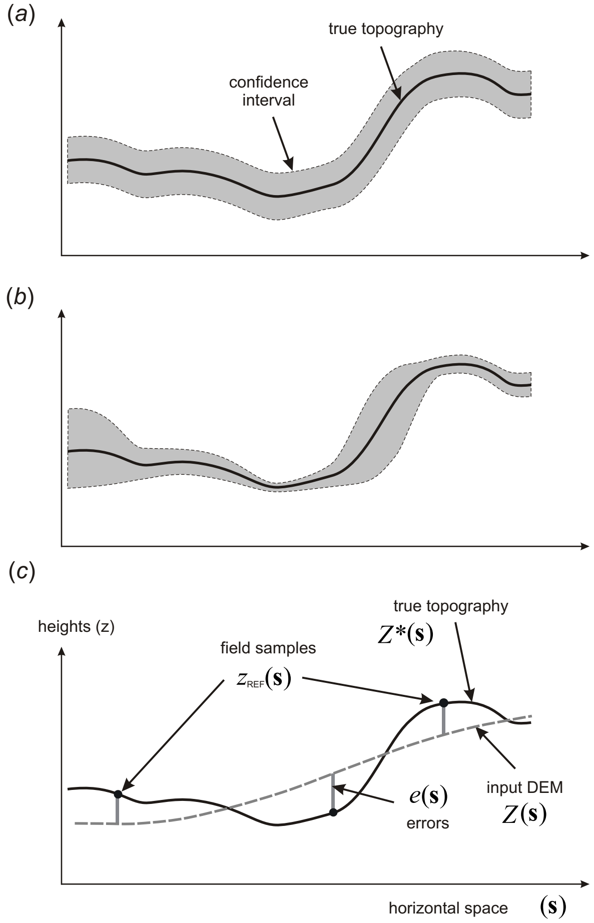


Fig. 2.12: Conceptual aspects of modeling topography using geostatistics. A cross section showing the true topography and the associated uncertainty: (a) constant, global uncertainty model and (b) spatially variable uncertainty; (c) estimation of the DEM errors using precise height measurements.

So far, DEMs have been modeled by using solely point-sampled elevations. For example, ordinary kriging is used to generate DEMs (Mitas and Mitasova, 1999; Lloyd and Atkinson, 2002); conditional geostatistical simulations are used to generate equiprobable realizations of DEMs (Fisher, 1998; Kyriakidis et al., 1999). In most studies, no explanatory information on topography is employed directly in the geostatistical modeling. Compared to the approach of Hutchinson (1989, 1996) where auxiliary maps of streams are often used to produce hydrologically-correct DEMs, the geostatistical approach to modeling of topography has often been limited to analysis of point-sampled elevations.

2.7.1 Some theoretical considerations

DEMs are today increasingly produced using automated (mobile GPS) field sampling of elevations or airborne scanning devices (radar or LiDAR-based systems). In the case elevations are sampled at sparsely-located points, a DEM can be generated using geostatistical techniques such as ordinary kriging (Wood and Fisher, 1993; Mitas and Mitasova, 1999). The elevation at some grid node ($\mathbf{s}_0$) of the output DEM can be interpolated using ordinary kriging (Eq.1.3.2); the same technique can be used to produce simulated DEMs (see section 2.4). Direct simulation of DEMs using the sampled elevations is discussed in detail by Kyriakidis et al. (1999).

The use of kriging in geomorphometry to generate DEMs has been criticized by many (Wood and Fisher, 1993; Mitas and Mitasova, 1999; Li et al., 2005b), mainly because it leads to many artifacts, it oversmooths elevations and it is very sensitive to sampling density and local extreme values. So far, splines have been used in geomorphometry as a preferred technique to generate DEMs or to filter local errors (Mitasova et al., 2005). More recently, Hengl et al. (2008) demonstrated that regression-kriging can be used to employ auxiliary maps, such as maps of drainage patterns, land cover and remote sensing-based indices, directly in the geostatistical modeling of topography. Details are now discussed in the succeeding sections.

If additional, auxiliary maps (drainage network, water bodies, physiographic break-lines) are available, a DEM can be generated from the point-measured elevations using the regression-kriging model (Eq.2.1.4). The biggest advantage of using auxiliary maps is a possibility to more precisely model uncertainty of the sampled elevations and analyze which external factors cause this variability. Whereas, in pure statistical Monte Carlo approach where we work with global, constant parameters (Fig. 2.12a), in the case of geostatistical modeling, the DEM uncertainty can be modeled with a much higher level of detail (Fig. 2.12b).

In the case a DEM is obtained from an airborne or satellite-based scanning mission (radar, LiDAR or stereoscopic images), elevations are already available over the whole area of interest. Geostatistics is then used to analyze inherent errors in the DEM images (Grohmann, 2004), filter local errors caused by physical limitations of the instrument (Lloyd and Atkinson, 2002; Evans and Hudak, 2007), and eventually cluster the area according to their statistical properties (Lloyd and Atkinson, 1998).

Geostatistical simulation of complete elevation data is somewhat more complicated than with point data. At the moment, the simulations of DEM images are most commonly obtained by simulating error surfaces derived from additional field-control samples (Fig. 2.12c). The elevations measured at control points are used to assess the errors. The point map of DEM errors can then be used to generate equiprobable error surfaces, which are then added to the original DEM to produce an equiprobable realization of a DEM (Hunter and Goodchild, 1997; Holmes et al., 2000; Endreny and Wood, 2001; Temme et al., 2008). From a statistical perspective, a DEM produced directly by using scanning devices (SRTM, LiDAR) consists of three components: $Z^*(\mathbf{s})$ the deterministic component, $\varepsilon'(\mathbf{s})$ the spatially correlated random component, and ε'' is the pure noise, usually the result of the measurement error. In raster-GIS terms, we can decompose a DEM into two grids: (1) the deterministic DEM and (2) the error surface. If precise point-samples of topography (e.g. highly precise GPS measurements) are available, they can be used to estimate the errors (Fig. 2.12c):

$$e(\mathbf{s}_i) = z^*_{\mathsf{REF}}(\mathbf{s}_i) - Z(\mathbf{s}_i); \quad E\{e(\mathbf{s})\} = 0 \tag{2.7.1}$$

The measured errors at point locations can also be manipulated using geostatistics to generate the **error surface**:

$$e^{(\mathsf{SIM})}(\mathbf{s}_0) = E\left\{\varepsilon | e(\mathbf{s}_i), \gamma_e(\mathbf{h})\right\} \tag{2.7.2}$$

The simulated error surface can then be added to the deterministic DEM to produce an equiprobable realization of a DEM:

$$z^{(\mathsf{SIM})}(\mathbf{s}_j) = z^*(\mathbf{s}_j) + e^{(\mathsf{SIM})}(\mathbf{s}_j) \tag{2.7.3}$$

An obvious problem with this approach is that the deterministic DEM ($z^*(\mathbf{s}_j)$) is usually not available, so that the input DEM is in fact used to generate simulations, which leads to (see e.g. Holmes et al. (2000); Temme et al. (2008)):

$$\begin{aligned} z^{(\mathsf{SIM})}(\mathbf{s}_j) &= z(\mathbf{s}_j) + e^{(\mathsf{SIM})}(\mathbf{s}_j) \\ &= z^*(\mathbf{s}_j) + \varepsilon'(\mathbf{s}_j) + \varepsilon'' + e^{(\mathsf{SIM})}(\mathbf{s}_j) \end{aligned} \tag{2.7.4}$$

which means that the simulated error surface and the inherent error component, at some locations, will double, and at others will annul each other. However, because the inherent error and the simulated error are in fact independent, the mean of the summed errors will be close to zero (unbiased simulation), but the standard deviation of the error component will be on average 40% larger. Hence a DEM simulated using Eq.(2.7.3) will be much noisier than the original DEM. The solution to this problem is to substitute the deterministic DEM component with a smoother DEM, e.g. a DEM derived from contour lines digitized from a finer-scale topo-map. As an alternative, the deterministic DEM component can be prepared by smoothing the original DEM i.e. filtering it for known noise and systematic errors (see e.g. Selige et al. (2006)).

2.7.2 Choice of auxiliary maps

The spatially correlated error component will also often correlate with the explanatory information (Oksanen, 2006). For example, in traditional cartography, it is known that the error of measuring elevations is primarily determined by the complexity of terrain (the slope factor), land cover (density of objects) and relative visibility (the shadow effect). Especially in the cases where the DEMs are produced through photogrammetric methods, information about the terrain shading can be used to estimate the expected error of measuring heights. Similarly, a SRTM DEM will show systematic errors in areas of higher canopy and smaller precision in areas which are hidden or poorly exposed to the scanning device (Hengl and Reuter, 2008, p.79-80). This opens a possibility to also use the regression-kriging model with auxiliary maps to produce a more realistic error surface (Hengl et al., 2008).

There are three major groups of auxiliary maps of interest to DEM generation:

(1.) Hydrological maps:

- stream line data;
- water stagnation areas (soil-water content images);
- seashore and lakes border lines;

(2.) Land cover maps:

- canopy height;
- Leaf Area Index;
- land cover classes;

(3.) Geomorphological maps:

- surface roughness maps;
- physiographic breaks;
- ridges and terraces;

A lot of topography-connected information can be derived from remote sensing multi- and hyper-spectral images, such as shading-based indices, drainage patterns, ridge-lines, topographic breaks. All these can be derived using automated (pattern recognition) techniques, which can significantly speed up processing for large areas.

Many auxiliary maps will mutually overlap in information and value. Ideally, auxiliary maps used to improve generation of DEMs should be only GIS layers produced independently from the sampled elevations — e.g. remotely sensed images, topographic features, thematic maps etc. Where this is not possible, auxiliary maps can be derived from an existing DEM, provided that this DEM is generated using independent elevation measurements. Care needs to be taken not to employ auxiliary maps which are only indirectly or accidentally connected with the variations in topography. Otherwise unrealistic simulations can be generated, of even poorer quality than if only standard DEM generation techniques are used (Hengl et al., 2008).

2.8 Regression-kriging and sampling optimization algorithms

Understanding the concepts of regression-kriging is not only important to know how to generate maps, but also to know how to prepare a sampling plan and eventually minimize the survey costs. Because the costs of the field survey are usually the biggest part of the survey budget, this issue will become more and more important in the coming years. So far, two main groups of sampling strategies have been commonly utilized for the purpose of environmental mapping (Guttorp, 2003):

- **Regular sampling** — This has the advantage that it systematically covers the area of interest (maximized mean shortest distance), so that the overall prediction variance is usually minimized[24]. The disadvantage of this technique is that it misrepresents distances smaller than the grid size (short range variation).
- **Randomized sampling** — This has the advantage that it represents all distances between the points, which is beneficial for the variogram estimation. The disadvantage is that the spreading of the points in geographic space is lower than in the case of regular sampling, so that the overall precision of the final maps will often be lower.

None of two strategies is universally applicable so that often their combination is recommended: e.g. put half of the points using regular and half using a randomized strategy. Both random and regular sampling strategies belong to the group of design-based sampling. The other big group of sampling designs are the model-based designs. A difference between a design-based sampling (e.g. simple random sampling) and the model-based design is that, in the case of the model-based design, the model is defined and commonly a single optimal design that maximizes/minimizes some criteria can be produced.

In the case of regression-kriging, there are much more possibilities to improve sampling than by using design-based sampling. First, in the case of preparing a sampling design for new survey, the samples can be more objectively located by using some **response surface design** (Hengl et al., 2004b), including the **Latin hypercube sampling** (Minasny and McBratney, 2006). The Latin hypercube sampling will ensure that all points are well-placed in the feature space defined by the environmental factors — these will later be used as predictors — and that the extrapolation in feature space is minimized. Second, once we have collected samples and estimated the regression-kriging model, we can then optimize sampling and derive (1) number of required additional observations and (2) their optimal location in both respective spaces. This leads to a principle of the two-stage[25] model-based sampling (Fig. 2.13).

The **two-stage sampling** is a guarantee of minimization of the survey costs. In the first phase, the surveyors will produce a sampling plan with minimum survey costs — just to have enough points to get a '*rough*' estimate of the regression-kriging model. Once the model is approximated (correlation and variogram model), and depending on the prescribed accuracy (overall prediction variance), the second (additional) sampling plan can be generated. Now we can re-estimate the regression-kriging model and update the predictions so that they fit exactly our prescribed precision requirements. Brus and Heuvelink (2007) tested the use of simulated annealing to produce optimal designs based on the regression-kriging model, and concluded that the resulting sampling plans will lead to hybrid patterns showing spreading in both feature and geographical space. An R package intamap[26] (procedures for automated interpolation) has been recently released that implements such algorithms to run sampling optimization. The interactive version of the intamap package allows users to create either new sampling networks with spatial coverage methods, or to optimally allocate new observations using spatial simulated annealing (see results for the meuse case study in Fig. 2.13).

Smarter allocation of the points in the feature and geographic space often proves that equally precise maps could have been produced with much less points than actually collected. This might surprise you, but it has a strong theoretical background. Especially if the predictors are highly correlated with the target variable and if this correlation is close to linear, there is really no need to collect many samples in the study area. In order to produce precise predictions, it would be enough if we spread them around extremes of the feature space and possibly maximized their spreading in the area of interest (Hengl et al., 2004b). Of course, number of sampling points is mainly dictated by our precision requirements, so that more accurate (low overall precision variance) and detailed (fine cell size) maps of environmental variables will often require denser sampling densities.

[24] If ordinary kriging is used to generate predictions.

[25] Ideally, already one iteration of additional sampling should guarantee map of required accuracy/quality. In practice, also the estimation of model will need to be updated with additional predictors, hence more iterations can be anticipated.

[26] http://intamap.org

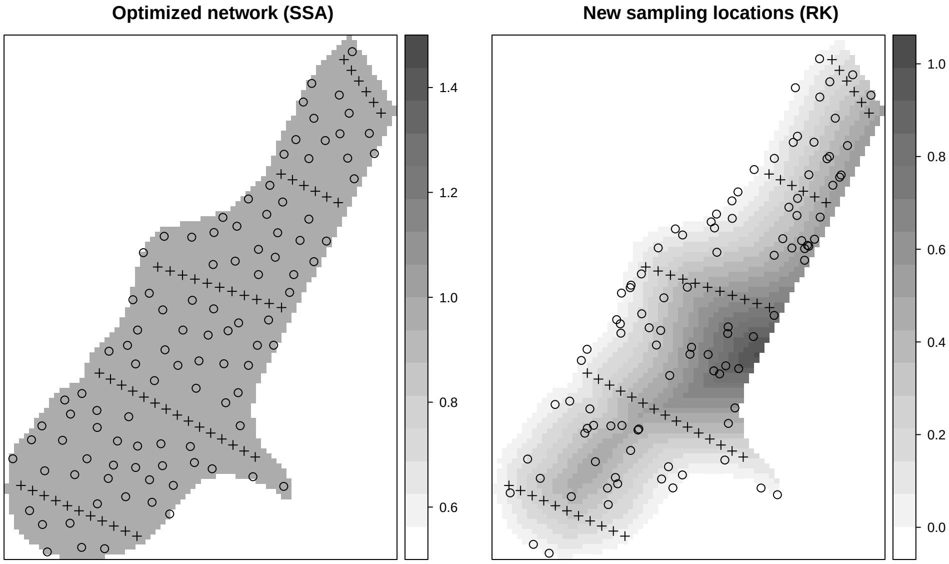


Fig. 2.13: Example of the two-stage model-based sampling: + — 50 first stage samples (transects); o — 100 new samples allocated using the model estimated in the first stage (optimized allocation produced using Spatial Simulated Annealing implemented in the intamapInteractive package). In the case of low correlation with auxiliary maps (left), new sampling design shows have higher spreading in the geographical space; if the correlation with predictors is high (right), then the new sampling design follows the extremes of the features space.

2.9 Fields of application

With the rapid development of remote sensing and geoinformation science, natural resources survey teams are now increasingly creating their products (geoinformation) using ancillary data sources and computer programs — the so-called *direct-to-digital* approach. For example, sampled concentrations of heavy metals can be mapped with higher detail if information about the sources of pollution (distance to industrial areas and traffic or map showing the flooding potential) is used. In the following sections, a short review of the groups of application where regression-kriging has shown its potential is given.

2.9.1 Soil mapping applications

In digital soil mapping, soil variables such as pH, clay content or concentration of a heavy metal, are increasingly mapped using the regression-kriging framework: the deterministic part of variation is dealt with maps of soil forming factors (climatic, relief-based and geological factors) and the residuals are dealt with kriging (McBratney et al., 2003). The same techniques is now used to map categorical variables (Hengl et al., 2007b). A typical soil mapping project based on geostatistics will also be demonstrated in the following chapter of this handbook. This follows the generic framework for spatial prediction set in Hengl et al. (2004a) and applicable also to other environmental and geosciences (Fig. 2.14).

In geomorphometry, auxiliary maps, such as maps of drainage patterns, land cover and remote sensing-based indices, are increasingly used for geostatistical modeling of topography together with point data sets. Auxiliary maps can help explain spatial distribution of errors in DEMs and regression-kriging can be used to generate equiprobable realizations of topography or map the errors in the area of interest (Hengl et al., 2008). Such hybrid geostatistical techniques will be more and more attractive for handling rich LiDAR and radar-based topographic data, both to analyze their inherent geostatistical properties and generate DEMs fit-for-use in various environmental and earth science applications.

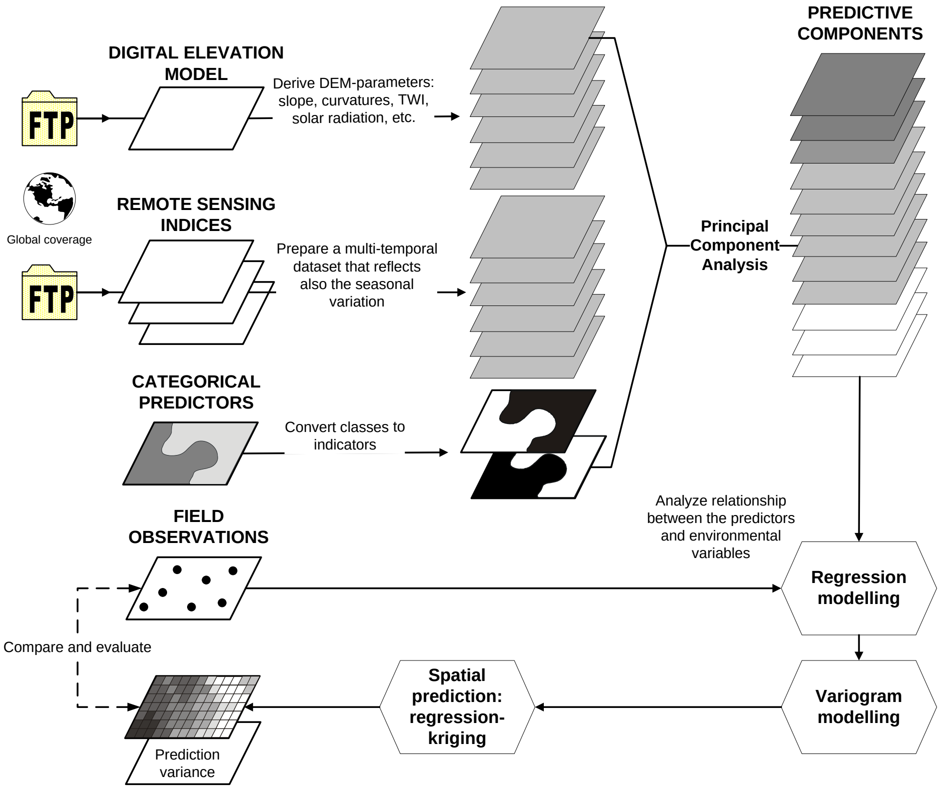

DIGITAL ELEVATION MODEL
FTP
Derive DEM-parameters: slope, curvatures, TWI, solar radiation, etc.
Global coverage
REMOTE SENSING INDICES
FTP
Prepare a multi-temporal dataset that reflects also the seasonal variation
CATEGORICAL PREDICTORS
Convert classes to indicators
Principal Component Analysis
PREDICTIVE COMPONENTS
FIELD OBSERVATIONS
Analyze relationship between the predictors and environmental variables
Regression modelling
Compare and evaluate
Variogram modelling
Spatial prediction: regression-kriging
Prediction variance

Fig. 2.14: A generic framework for digital soil mapping based on regression-kriging. After Hengl et al. (2007b).

2.9.2 Interpolation of climatic and meteorological data

Regression-kriging of climatic variables, especially the ones derived from DEMs, is now favoured in many climatologic applications (Jarvis and Stuart, 2001; Lloyd, 2005). DEMs are most commonly used to adjust measurements at meteorological stations to local topographic conditions. Other auxiliary predictors used range from distance to sea, meteorological images of land surface temperature, water vapor, short-wave radiation flux, surface albedo, snow Cover, fraction of vegetation cover (see also section 4). In many cases, real deterministic models can be used to make predictions, so that regression-kriging is only used to calibrate the values using the real observations (D'Agostino and Zelenka, 1992, see also Fig. 2.3). The exercise in chapter 11 demonstrates the benefits of using the auxiliary predictors to map climatic variables. In this case the predictors explained almost 90% of variation in the land surface temperatures measured at 152 stations. Such high R-square allows us to *extrapolate* the values much further from the original sampling locations, which would be completely inappropriate to do by using ordinary kriging. The increase of the predictive capabilities using the explanatory information and regression-kriging has been also reported by several participants of the Conference on spatial interpolation in climatology and meteorology (Szalai et al., 2007).

Interpolation of climatic and meteorological data is also interesting because the explanatory (meteorological images) data are today increasingly collected in shorter time intervals so that time-series of images are available and can be used to develop spatio-temporal regression-kriging models. Note also that many meteorological prediction models can generate maps of forecasted conditions in the close-future time, which could then again be calibrated using the actual measurements and RK framework.

2.9.3 Species distribution modeling

Geostatistics is considered to be one of the four spatially-implicit group of techniques suited for species distribution modeling — the other three being: autoregressive models, geographically weighted regression and parameter estimation models (Miller et al., 2007). Type of technique suitable for analysis of species (occurrence) records is largely determined by the species' biology. There is a distinct difference between field observation of animal and plant species and measurements of soil or meteorological variables. Especially the observations of animal species asks for high sampling densities in temporal dimension. If the biological species are represented with quantitative composite measures (density, occurrence, biomass, habitat category), such measures are fit for use with standard spatio-temporal geostatistical tools. Some early examples of using geostatistics with the species occurrence records can be found in the work of Legendre and Fortin (1989) and Gotway and Stroup (1997). Kleinschmidt et al. (2005) uses regression-kriging method, based on the Generalized mixed model, to predict the malaria incidence rates in South Africa. Miller (2005) uses a similar principle (predict the regression part, analyze and interpolate residuals, and add them back to predictions) to generate vegetation maps. Miller et al. (2007) further provide a review of predictive vegetation models that incorporate geographical aspect into analysis. Pure interpolation techniques will often outperform niche based models (Bahn and McGill, 2007), although there is no reason not to combine them. Pebesma et al. (2005) demonstrates that geostatistics is fit to be used with spatio-temporal species density records. §8 shows that even occurrence-only records can be successfully analyzed using geostatistics i.e. regression-kriging.

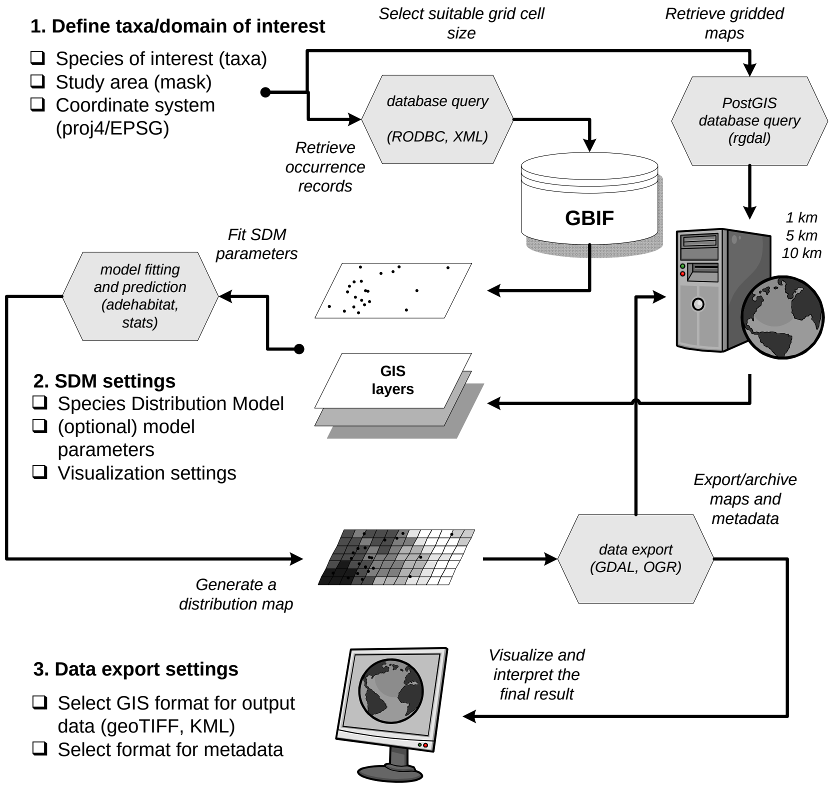

1. Define taxa/domain of interest
❑ Species of interest (taxa)
❑ Study area (mask)
❑ Coordinate system (proj4/EPSG)
Select suitable grid cell size
Retrieve gridded maps
database query (RODBC, XML)
PostGIS database query (rgdal)
Retrieve occurrence records
GBIF
1 km
5 km
10 km
Fit SDM parameters
model fitting and prediction (adehabitat, stats)
GIS layers
2. SDM settings
❑ Species Distribution Model
❑ (optional) model parameters
❑ Visualization settings
Export/archive maps and metadata
data export (GDAL, OGR)
Generate a distribution map
3. Data export settings
❑ Select GIS format for output data (geoTIFF, KML)
❑ Select format for metadata
Visualize and interpret the final result

Fig. 2.15: Schematic example of a geo-processing service to automate extraction of species distribution maps using GBIF occurrence records and gridded predictors. The suitable R packages are indicated in brackets.

Fig. 2.15 shows an example of a generic automated data processing scheme to generate distribution maps and similar biodiversity maps using web-data. The occurrence(-only) records can be retrieved from the Global Biodiversity Information Facility[27] (GBIF) Data Portal, then overlaid over number of gridded predictors (possibly stored in a PostGIS database), a species' prediction model can then be fitted, and results exported to some GIS format / KML. Such automated mapping portals are now increasingly being used to generate up-to-date species' distribution maps.

[27] Established in 2001; today the largest international data sharing network for biodiversity.

2.9.4 Downscaling environmental data

Interpolation becomes down-scaling once the grid resolution in more than 50% of the area is finer than it should be for the given sampling density. For example, in soil mapping, one point sample should cover 160 pixels (Hengl, 2006). If we have 100 samples and the size of the area is 10 km^2, then it is valid to map soil variables at resolutions of 25 m (maximum 10 m) or coarser. Note that down-scaling is only valid if we have some auxiliary data (e.g. digital elevation model) which is of finer resolution than the effective grid resolution, and which is highly correlated with the variable of interest.

If the auxiliary predictors are available at finer resolutions than the sampling intensity, regression-kriging can be used to downscale information. Much of recent research in the field of biogeography, for example, has been focusing on the down-scaling techniques (Araújo et al., 2005). Hengl et al. (2008) shows how auxiliary maps can be used to downscale SRTM DEMs from 90 to 30 m resolution. Pebesma et al. (2007) use various auxiliary maps to improve detail of air pollution predictions. For the success of downscaling procedures using regression-kriging, the main issue is how to locate the samples so that extrapolation in the feature space is minimized.

2.10 Final notes about regression-kriging

At the moment, there are not many contra-arguments not to replace the existing traditional soil, vegetation, climatic, geological and similar maps with the maps produced using analytical techniques. Note that this does not mean that we should abandon the traditional concepts of field survey and that surveyors are becoming obsolete. On the contrary, surveyors continue to be needed to prepare and collect the input data and to assess the results of spatial prediction. On the other hand, they are less and less involved in the actual delineation of features or derivation of predictions, which is increasingly the role of the predictive models.

One such linear prediction techniques that is especially promoted in this handbook is regression-kriging (RK). It can be used to interpolate sampled environmental variables (both continuous and categorical) from large point sets. However, in spite of this and other attractive properties of RK, it is not as widely used in geosciences as might be expected. The barriers to widespread routine use of RK in environmental modeling and mapping are as follows. First, the statistical analysis in the case of RK is more sophisticated than for simple mechanistic or kriging techniques. Second, RK is computationally demanding[28] and often cannot be run on standard PCs. The third problem is that many users are confused by the quantity of spatial prediction options, so that they are never sure which one is the most appropriate. In addition, there is a lack of user-friendly GIS environments to run RK. This is because, for many years GIS technologies and geostatistical techniques have been developing independently. Today, a border line between statistical and geographical computing is fading away, in which you will hopefully be more convinced in the remaining chapters of this guide.

2.10.1 Alternatives to RK

The competitors to RK include completely different methods that may fit certain situations better. If the explanatory data is of different origin and reliability, the Bayesian Maximum Entropy approach might be a better alternative (D'Or, 2003). There are also machine learning techniques that combine neural network algorithms and robust prediction techniques (Kanevski et al., 1997). Henderson et al. (2004) used decision trees to predict various soil parameters from large quantity of soil profile data and with the help of land surface and remote sensing attributes. This technique is flexible, optimizes local fits and can be used within a GIS. However, it is statistically suboptimal because it ignores spatial location of points during the derivation of classification trees. The same authors (Henderson et al., 2004, pp.394–396) further reported that, although there is still some spatial correlation in the residuals, it is not clear how to employ it.

Regression-kriging must also be compared with alternative kriging techniques, such as **collocated co-kriging**, which also makes use of the explanatory information. However, collocated co-kriging is developed for situations in which the explanatory information is not spatially exhaustive (Knotters et al., 1995). CK also requires simultaneous modeling of both direct and cross-variograms, which can be time-consuming for large

[28] Why does RK takes so much time? The most enduring computations are connected with derivation of distances from the new point to all sampled points. This can be speed up by setting up a smaller search radius.

number of covariates[29]. In the case where the covariates are available as complete maps, RK will generally be preferred over CK, although CK may in some circumstances give superior results (D'Agostino and Zelenka, 1992; Goovaerts, 1999; Rossiter, 2007). In the case auxiliary point samples of covariates, in addition to auxiliary raster maps, are available, regression-kriging can be combined with co-kriging: first the deterministic part can be dealt with the regression, then the residuals can be interpolated using co-kriging (auxiliary point samples) and added back to the estimated deterministic part of variation.

2.10.2 Limitations of RK

RK have shown a potential to become the most popular mapping technique used by environmental scientists because it is (a) easy to use, and (b) it outperforms plain geostatistical techniques. However, success of RK largely depends on characteristics of the case study i.e. quality of the input data. These are some main consideration one should have in mind when using RK:

(1.) *Data quality*: RK relies completely on the quality of data. If the data comes from different sources and have been sampled using biased or unrepresentative design, the predictions might be even worse than with simple mechanistic prediction techniques. Even a single bad data point can make any regression arbitrarily bad, which affects the RK prediction over the whole area.

(2.) *Under-sampling*: For regression modeling, the multivariate feature space must be well-represented in all dimensions. For variogram modeling, an adequate number of point-pairs must be available at various spacings. Webster and Oliver (2001, p.85) recommend at least 50 and preferably 300 points for variogram estimation. Neter et al. (1996) recommends at least 10 observations per predictor for multiple regression. We strongly recommend using RK only for data sets with more than 50 total observations and at least 10 observations per predictor to prevent over-fitting.

(3.) *Reliable estimation of the covariance/regression model*: The major dissatisfaction of using KED or RK is that both the regression model parameters and covariance function parameters need to be estimated simultaneously. However, in order to estimate coefficients we need to know covariance function of residuals, which can only be estimated after the coefficients (the chicken-egg problem). Here, we have assumed that a single iteration is a satisfactory solution, although someone might also look for other iterative solutions (Kitanidis, 1994). Lark et al. (2005) recently suggested that an iterative Restricted Maximum Likelihood (REML) approach should be used to provide an unbiased estimate of the variogram and regression coefficients. However, this approach is rather demanding for $\gg 10^3$ point data sets because for each iteration, an $n \times n$ matrix is inverted (Minasny and McBratney, 2007).

(4.) *Extrapolation outside the sampled feature space*: If the points do not represent feature space or represent only the central part of it, this will often lead to poor estimation of the model and poor spatial prediction. For this reason, it is important that the points be well spread at the edges of the feature space and that they be symmetrically spread around the center of the feature space (Hengl et al., 2004b). Assessing the extrapolation in feature space is also interesting to allocate additional point samples that can be used to improve the existing prediction models. This also justifies use of multiple predictors to fit the target variable, instead of using only the most significant predictor or first principal component, which if, for example, advocated by the Isatis development team (Bleines et al., 2004).

(5.) *Predictors with uneven relation to the target variable*: Auxiliary maps should have a constant physical relationship with the target variable in all parts of the study area, otherwise artifacts will be produced. An example is a single NDVI as a predictor of topsoil organic matter. If an agricultural field has just been harvested (low NDVI), the prediction map will (incorrectly) show very low organic matter content within the crop field.

(6.) *Intermediate-scale modeling*: RK has not been adapted to fit data locally, with arbitrary neighborhoods for the regression as can be done with kriging with moving window (Walter et al., 2001). Many practitioners would like to adjust the neighborhood to fit their concepts of the scale of processes that are not truly global (across the whole study area) but not completely local either.

[29] Co-kriging requires estimation of $p+1$ variograms, plus $\left[p \cdot (p+1)\right]/2$ cross-variograms, where the p is the number of predictors (Knotters et al., 1995).

(7.) *Data over-fitting problems*: Care needs to be taken when fitting the statistical models — today, complex models and large quantities of predictors can be used so that the model can fit the data almost 100%. But there is a distinction between the goodness of fit and true success of prediction that cannot really be assessed without independent validation (Rykiel, 1996).

If any of these problems occur, RK can give even worse results than even non-statistical, empirical spatial predictors such as inverse distance interpolation or expert systems. The difficulties listed above might also be considered as challenges for the geostatisticians.

2.10.3 Beyond RK

Although the bibliometric research of Zhou et al. (2007) indicates that the field of geostatistics has already reached its peak in 1996–1998, the development of regression-kriging and similar hybrid techniques is certainly not over and the methods will continue to evolve both from theoretical and practical aspect. Gotway Crawford and Young (2008) recognizes four '*hot*' areas of geostatistics that will receive attention in the near future: (1) geostatistics in non-euclidian space (i.e. space that accounts for barriers, streams, disease transmittion vectors etc.); (2) assessment of spatio-temporal support — spatial prediction methods will be increasingly compared at various spatial/temporal scales; users are increasingly doing predictions from point to area support and vice versa; (3) kriging is increasingly used with discrete data and uncertain data (this emphasized the importance of using Bayesian-based models), and (4) geostatistics as a tool of politics.

What you can certainly anticipate in the near future considering regression-kriging connected methods are the following six developments:

- *More sophisticated prediction models*: Typically, regression-kriging is sensitive to blunders in data, local outliers and small size data sets. To avoid such problems, we will experience an evolution of methods that are more generic and more robust to be used to any type of data set. Recently, several authors suggested ways to make more sophisticated, more universally applicable BLUPs (Lark et al., 2005; Minasny and McBratney, 2007; Bárdossy and Li, 2008). We can anticipate a further development of intelligent, iterative data fitting algorithms that can account for problems of local hot-spots, mixed data and poor sampling strategies. This is now one of the major focuses of the intamap project (Pebesma et al., 2009).

- *Local regression-kriging*: As mentioned previously in §2.2, local regression-kriging algorithms are yet to be developed. Integration of the local prediction algorithms (Haas, 1990; Walter et al., 2001) would open many new data analysis possibilities. For example, with local estimation of the regression coefficients and variogram parameters, a user will be able to analyze which predictors are more dominant in different parts of the study area, and how much these parameters vary in space. The output of the interpolation with not be only a map of predictions, but also the maps of (local) regression coefficients, R-square, variogram parameters and similar. Lloyd (2009) recently compared KED (monthly precipitation in UK) based on local variogram models and discovered that it provides more accurate predictions (as judged by cross-validation statistics) than any other 'global' approach.

- *User-friendly sampling optimisation packages*: Although methodologies both to plan new sampling designs, and to optimize additional sampling designs have already been tested and described (Minasny and McBratney, 2006; Brus and Heuvelink, 2007), techniques such as simulated annealing or Latin hypercube sampling are still not used in operational mapping. The recently released intamapInteractive package now supports simulated annealing and optimization of sampling designs following the regression-kriging modeling. Development of user-friendly sampling design packages will allow mapping teams to generate (*smart*) sampling schemes at the click of button.

- *Automated interpolation of categorical variables*: So far no tool exists that can automatically generate membership maps given a point data with observed categories (e.g. soil types, land degradation types etc.). A compositional RK algorithm is needed that takes into account relationship between all categories in the legend, and then fits regression models and variogram models for all classes (Hengl et al., 2007b).

- *Intelligent data analysis reports generation*: The next generation of geostatistical packages will be intelligent. It will not only generate predictions and prediction variances, but will also provide interpretation of the fitted models and analysis of the intrinsic properties of the input data sets. This will include detection of possible outliers and hot-spots, robust estimation of the non-linear regression model, assessment of the quality of the input data sets and final maps. The R package automap, for example, is pointing to this direction.

- *Multi-temporal, multi-variate prediction models*: At the moment, most of the geostatistical mapping projects in environmental sciences focus on mapping a single variable sampled in a short(er) period of time and for a local area of interest. It will not take too long until we will have a global repository of (multi-temporal) predictors (see further section 4.1) and point data sets that could then be interpolated all at once (to employ all possible relationships and cross-correlations). The future data sets will definitively be multi-temporal and multi-variate, and it will certainly ask for more powerful computers and more sophisticated spatio-temporal 3D mapping tools. Consequently, outputs of the spatial prediction models will be animations and multimedia, rather then simple and static 2D maps.

Although we can observe that with the more sophisticated methods (e.g. REML approach), we are able to produce more realistic models, the quality of the output maps depends much more on the quality of input data (Minasny and McBratney, 2007). Hence, we can also anticipate that evolution of technology such as hyperspectral remote sensing and LiDAR will contribute to the field of geostatistical mapping even more than the development of the more sophisticated algorithms.

Finally, we can conclude that an unavoidable trend in the evolution of spatial prediction models will be a **development and use of fully-automated, robust, intelligent mapping systems** (see further §3.4.3). Systems that will be able to detect possible problems in the data, iteratively estimate the most reasonable model parameters, employ all possible explanatory and empirical data, and assist the user in generating the survey reports. Certainly, in the near future, a prediction model will be able to run more analysis with less interaction with user, and offer more information to decision makers. This might overload the inexperience users, so that practical guides even thicker than this one can be anticipated.

Further reading:

★ Banerjee, S. and Carlin, B. and Gelfand, A. 2004. **Hierarchical Modeling and Analysis for Spatial Data**. Chapman & Hall/CRC, Boca Raton, 472 p.

★ Christensen, R. 2001. Best Linear Unbiased Prediction of Spatial Data: Kriging. In: Cristensen, R. **Linear Models for Multivariate, Time Series, and Spatial Data**, Springer, 420 p.

★ Hengl T., Heuvelink G. B. M., Rossiter D. G., 2007. About regression-kriging: from equations to case studies. Computers & Geosciences, 33(10): 1301–1315.

★ Minasny, B., McBratney, A. B., 2007. Spatial prediction of soil properties using EBLUP with Matérn covariance function. Geoderma 140: 324–336.

★ Pebesma, E. J., 1999. **Gstat user's manual**. Department of Physical Geography, Utrecht University, Utrecht, 96 p.

★ Schabenberger, O., Gotway, C. A., 2004. **Statistical methods for spatial data analysis**. Chapman & Hall/CRC, 524 p.

★ Stein, M. L., 1999. **Interpolation of Spatial Data: Some Theory for Kriging**. Series in Statistics. Springer, New York, 247 p.

3

Software (R+GIS+GE)

This chapter will introduce you to five main packages that we will later on use in various exercises from chapter 5 to 11: R, SAGA, GRASS, ILWIS and Google Earth (GE). All these are available as open source or as freeware and no licenses are needed to use them. By combining the capabilities of the five software packages we can operationalize preparation, processing and the visualization of the generated maps. In this handbook, ILWIS GIS will be primarily used for basic editing and to process and prepare vector and raster maps; SAGA/GRASS GIS will be used to run analysis on DEMs, but also for geostatistical interpolations; R + packages will be used for various types of statistical and geostatistical analysis, but also for data processing automation; Google Earth will be used for visualization and interpretation of results.

In all cases we will use R to control all processes, so that each exercise will culminate in a single R script (‘*R on top*’; Fig. 3.1). In subsequent section, we will refer to the R + Open Source Desktop GIS combo of applications that combine geographical and statistical analysis and visualization as R+GIS+GE.

This chapter is meant to serve as a sort of a mini-manual that should help you to quickly obtain and install software, take first steps, and start doing some initial analysis. However, many details about the installation and processing steps are missing. To find more info about the algorithms and functionality of the software, please refer to the provided URLs and/or documentation listed at the end of the chapter. Note also that the instruction provided in this and following chapters basically refer to Window OS.

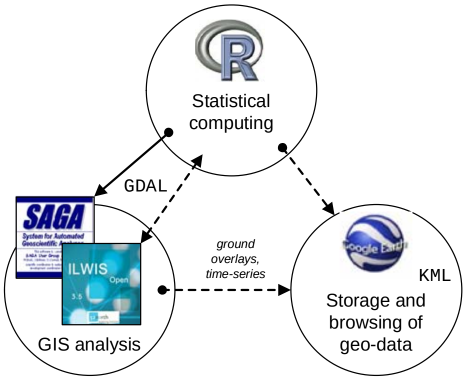
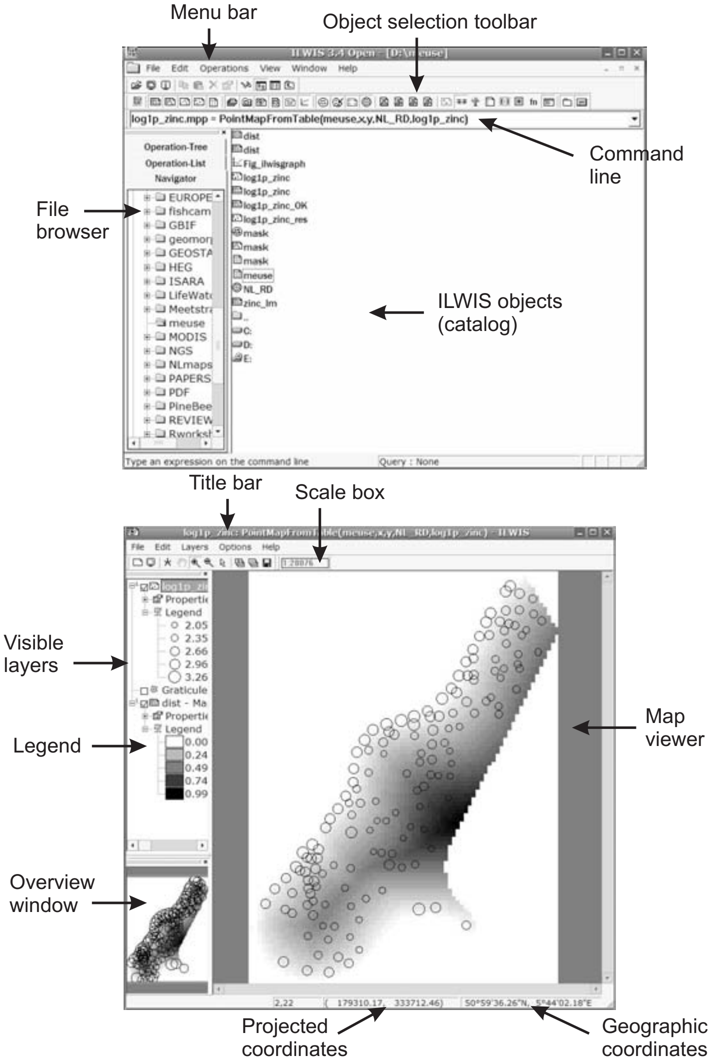

Statistical computing
GDAL
SAGA
ILWIS
ground overlays, time-series
KML
GIS analysis
Storage and browsing of geo-data

Fig. 3.1: The software triangle.

3.1 Geographical analysis: desktop GIS

3.1.1 ILWIS

ILWIS (**Integrated Land and Water Information System**) is a stand-alone integrated GIS package developed at the International Institute of Geo-Information Science and Earth Observations (ITC), Enschede, Netherlands. ILWIS was originally built for educational purposes and low-cost applications in developing countries. Its development started in 1984 and the first version (DOS version 1.0) was released in 1988. ILWIS 2.0 for Windows was released at the end of 1996, and a more compact and stable version 3.0 (WIN 95) was released by mid 2001. From 2004, ILWIS was distributed solely by ITC as shareware at a nominal price, and from July 2007, ILWIS shifted to open source. ILWIS is now freely available (‘as-is’ and free of charge) as open source software (binaries and source code) under the 52°North initiative.



```
log1p_zinc.mpp = PointMapFromTable(meuse,x,y,NL_RD,log1p_zinc)
```

Fig. 3.2: ILWIS main window (above) and map window (below).

The most recent version of ILWIS (3.6) offers a range of image processing, vector, raster, geostatistical, statistical, database and similar operations (Unit Geo Software Development, 2001). In addition, a user can create new scripts, adjust the operation menus and even build Visual Basic, Delphi, or C++ applications that will run on top of ILWIS and use its internal functions. In principle, the biggest advantage of ILWIS is that it is a compact package with a diverse vector and raster-based GIS functionality; the biggest disadvantages are bugs and instabilities and necessity to import data to ILWIS format from other more popular GIS packages.

To install ILWIS, download[1] and run the MS Windows installation. In the installation folder, you will find the main executable for ILWIS. Double click this file to start ILWIS. You will first see the main program window, which can be compared to the ArcGIS catalog (Fig. 3.2). The main program window is, in fact, a file browser which lists all ILWIS operations, objects and supplementary files within a working directory. The ILWIS Main window consists of a Menu bar, a Standard toolbar, an Object selection toolbar, a Command line, a Catalog, a Status bar and an Operations/Navigator pane with an Operation-tree, an Operation-list and a Navigator. The left pane (Operations/Navigator) is used to browse available operations and directories and the right menu shows available spatial objects and supplementary files (Fig. 3.2). GIS layers in different formats will not be visible in the catalog until we define the external file extension.

An advantage of ILWIS is that, every time a user runs an command from the menu bar or operation tree, ILWIS will record the operation in ILWIS command language. For example, you can run ordinary kriging using: from the main menu select *Operations* ↦ *Interpolation* ↦ *Point interpolation* ↦ *kriging*, which will be shown as:

```
ILWIS: log1p_zinc_OK.mpr = MapKrigingOrdinary(log1p_zinc, dist.grf,
+        Exponential(0.000,0.770,449.000), 3000, plane, 0, 20, 60, no)
```

where `log1p_zinc_OK.mpr` is the output map, `MapKrigingOrdinary` is the interpolation function, `log1p_zinc` is the attribute point map with values of the target variable, `dist.grf` is the grid definition (georeference) and `Exponential(0.000,0.770,449.000)` are the variogram parameters (see also section 5.3.2). This means that you can now edit this command and run it directly from the command line, instead of manually selecting the operations from the menu bar. In addition, you can copy such commands into an ILWIS script to enable automation of data analysis. ILWIS script can use up to nine script parameters, which can be either spatial objects, values or textual strings.

The new versions of ILWIS (>3.5) are developed as MS Visual 2008 project. The ILWIS user interface and ILWIS analytical functionality have now been completely separated making it easier to write server side

[1] `https://52north.org/download/Ilwis/`

applications for ILWIS. This allows us to control ILWIS also from R, e.g. by setting the location of ILWIS on your machine:

```
> ILWIS <- "C:\\Progra~1\\N52\\Ilwis35\\IlwisClient.exe -C"
```

To combine ILWIS and R commands in R, we use:

```
> shell(cmd=paste(ILWIS, "open log1p_zinc_OK.mpr -noask"), wait=F)
```

ILWIS has a number of built-in statistical and geostatistical functions. With respect to interpolation possibilities, it can be used to prepare a variogram, to analyze the anisotropy in the data (including the variogram surface), to run ordinary kriging and co-kriging (with one co-variable), to implement universal kriging with coordinates[2] as predictors and to run linear regression. ILWIS has also a number of original geostatistical algorithms. For example, it offers direct kriging from raster data (which can be used e.g. to filter the missing values in a raster map), and also does direct calculation of variograms from raster data. ILWIS is also suitable to run some basic statistical analysis on multiple raster layers (map lists): it offers principal component analysis on rasters, correlation analysis between rasters and multi-layer map statistics (min, max, average and standard deviation).

Although ILWIS cannot be used to run regression-kriging as defined in §2.1.5, it can be used to run a similar type of analysis. For example, a table can be imported and converted to a point map using the *Table to PointMap* operation. The point map can then be overlaid over raster maps to analyze if the two variables are correlated. This can be done by combining the table calculation and the `MapValue` function. In the same table, you can then derive a simple linear regression model e.g. `log1p_zinc = b0 + b1 * dist`. By fitting a least square fit using a polynomial, you will get: `b0=67.985` and `b1=-4.429`. This means that the zinc concentration decreases with an increase of `dist` — distance from the river (Fig. 3.3). Note that, in ILWIS, you cannot derive the Generalized Least Squares (GLS) regression coefficients (Eq.2.1.3) but only the OLS coefficients, which is statistically suboptimal method, because the residuals are possibly auto-correlated (see §2.1). In fact, regression modeling in ILWIS is so limited that the best advice is to always export the table data to R and then run statistical analysis using R packages. After estimating the regression coefficients, you can produce a map of `zinc` content (deterministic part of variation) by running a map calculation:

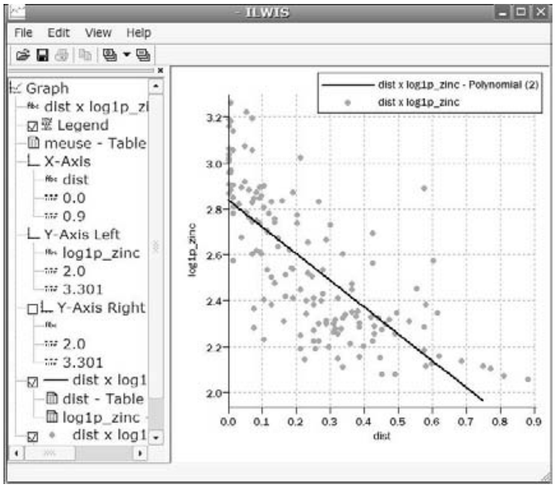


Fig. 3.3: Correlation plot `dist` vs `log1p_zinc`.

```
ILWIS: zinc_lm = 2.838 -1.169 * dist
```

Now you can estimate the residuals at sampled locations using table calculation:

```
ILWIS: log1p_zinc_res = log1p_zinc - MapValue(zinc_lm, coord(X,Y,NL_RD))
```

You can create a point map for residuals and derive a variogram of residuals by using operations *Statistics ↦ Spatial correlation* from the main menu. If you use a lag spacing of 100 m, you will get a variogram that can be fitted[3] with an exponential variogram model (`C0=0.008`, `C1=0.056`, `R=295`). The residuals can now be interpolated using ordinary kriging, which produces a typical kriging pattern. The fitted trend and residuals can then be added back together using:

[2] In ILWIS, the term *Universal kriging* is used exclusively for interpolation of point data using transforms of the coordinates.

[3] ILWIS does not support automated variogram fitting.

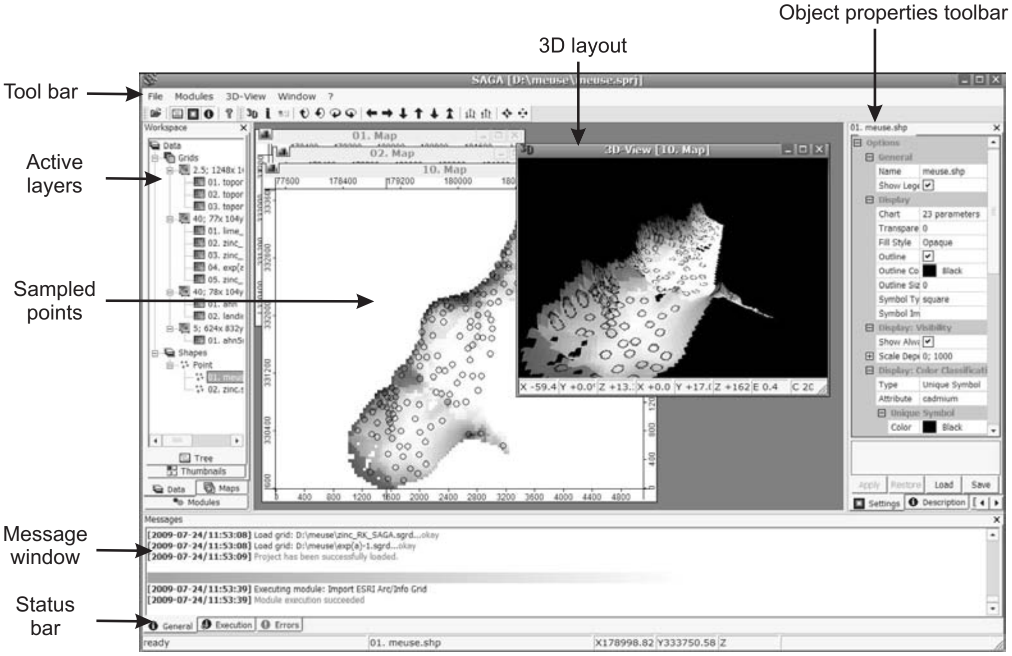
```
ILWIS: zinc_res_OK.mpr = MapKrigingOrdinary(log1p_zinc_res, dist.grf,
+           Exponential(0.008,0.056,295.000), 3000, plane, 0, 20, 60, no)
ILWIS: log1p_zinc_RK = zinc_lm + zinc_res_OK
ILWIS: zinc_rk = pow(10, log1p_zinc_RK)-1
```

which gives regression-kriging predictions. Note that, because a complete RK algorithm with GLS estimation of regression is not implemented in ILWIS (§2.1.5), we are not able to derive a map of the prediction variance (Eq.2.1.5). For these reasons, regression-kriging in ILWIS is not really encouraged and you should consider using more sophisticated geostatistical packages such as gstat and/or geoR.

Finally, raster maps from ILWIS can be exported to other packages. You can always export them to ArcInfo ASCII (.ASC) format. If the georeference in ILWIS has been set as center of the corner pixels, then you might need to manually edit the *.asc header[4]. Otherwise, you will not be able to import such maps to ArcGIS (8 or higher) or e.g. Idrisi. The pending ILWIS v3.7 will be even more compatible with the OGC simple features, WPS query features and similar. At the moment, the fastest and most efficient solution to read/write ILWIS rasters to other supported GDAL formats is FWTools[5].

3.1.2 SAGA

SAGA[6] (**System for Automated Geoscientific Analyzes**) is an open source GIS that has been developed since 2001 at the University of Göttingen[7], Germany, with the aim to simplify the implementation of new algorithms for spatial data analysis (Conrad, 2006, 2007). It is a full-fledged GIS with support for raster and vector data. SAGA includes a large set of geoscientific algorithms, and is especially powerful for the analysis of DEMs. With the release of version 2.0 in 2005, SAGA runs under both Windows and Linux operating systems. SAGA is an open-source package, which makes it especially attractive to users that would like to extend or improve its existing functionality.



01. meuse.shp

Options
General
Name: meuse.shp
Show Lege ☑
Display
Chart: 23 parameters
Transpare: 0
Fill Style: Opaque
Outline ☑
Outline Co: ■ Black
Outline Siz: 0
Symbol Ty: square
Symbol Im
Display: Visibility
Show Alw: ☑
Scale Dep: 0; 1000
Display: Color Classificati
Type: Unique Symbol
Attribute: cadmium
Unique Symbol
Color: ■ Black

```
[2009-07-24/11:53:08] Load grid: D:\meuse\zinc_RK_SAGA.sgrd...okay
[2009-07-24/11:53:08] Load grid: D:\meuse\exp(a)-1.sgrd...okay
[2009-07-24/11:53:09] Project has been successfully loaded.

[2009-07-24/11:53:39] Executing module: Import ESRI Arc/Info Grid
[2009-07-24/11:53:39] Module execution succeeded
```

ready | 01. meuse.shp | X178998.82 Y333750.58 Z

Fig. 3.4: The SAGA GUI elements and displays.

SAGA handles tables, vector and raster data and natively supports at least one file format for each data type. Currently SAGA (2.0.4) provides about 48 free module libraries with >300 modules, most of them

[4] Simply replace in the header of the file xllcenter and yllcenter with xllcorner and yllcorner.

[5] http://fwtools.maptools.org

[6] http://saga-gis.org

[7] The group recently collectively moved to the Institut für Geographie, University of Hamburg.

published under the GPL. The modules cover geo–statistics, geomorphometric analysis, image processing, cartographic projections, and various tools for vector and raster data manipulation. Modules can be executed directly by using their associated parameters window. After you have imported all maps to SAGA, you can also save the whole project so that all associated maps and visualization settings are retained. The most comprehensive modules in SAGA are connected with hydrologic, morphometric and climatic analysis of DEMs.

To install SAGA, download and unzip the compiled binaries[8] to some default local directory. Then run the `saga_gui.exe` and you will get a GUI as shown in Fig. 3.4. Likewise, to install SAGA on Linux machines, you can also download the compiled binaries (e.g. `saga_2.0.4_bin_linux.tar.gz`), then run some basic configuration:

```
> ./configure --enable-unicode --with-gnu-ld=yes
```

Note that there are two possibilities to compile the software under Linux: (a) either a Non-Unicode or (b) a Unicode version of SAGA. Building the Unicode version is straight forward and recommended. Have also in mind that, under Linux, wxWidgets, PROJ 4, GDAL, JASPER, TIFF and GCC packages need to be obtained and configured separately before you can start using SAGA (for the unicode compilation, `wx.` configure does not check for JASPER libraries!). For a correct installation, you should follow the instructions provided via the SAGA Wiki[9]. SAGA is unfortunately still not available for Mac OS X.

In addition to the GUI, a second user front end, the SAGA command line interpreter can be used to execute modules. Library RSAGA[10] provides access to geocomputing and terrain analysis functions of SAGA from within R by running the command line version of SAGA (Brenning, 2008)[11]. RSAGA provides also several R functions for handling and manipulating ASCII grids, including a flexible framework for applying local functions or focal functions to multiple grids (Brenning, 2008). It is important to emphasize that RSAGA package is used mainly to control SAGA operations from R. To be able to run RSAGA, you need to have SAGA installed locally on your machine. SAGA GIS does NOT come in the RSAGA library.

To find what SAGA libraries[12] and modules do and require, you should use the `rsaga.get.modules` and `rsaga.get.usage` commands. For example, to see which parameters are needed to generate a DEM from a shapefile type:

```
> rsaga.env()

 $workspace
 [1] "."

 $cmd
 [1] "saga_cmd.exe"

 $path
 [1] "C:/Progra~1/saga_vc"

 $modules
 [1] "C:/Progra~1/saga_vc/modules"


> rsaga.get.modules("grid_spline")

 $grid_spline
   code                                        name interactive
 1    0                 Thin Plate Spline (Global)       FALSE
 2    1                  Thin Plate Spline (Local)       FALSE
 3    2                    Thin Plate Spline (TIN)       FALSE
 4    3                     B-Spline Approximation       FALSE
 5    4           Multilevel B-Spline Interpolation       FALSE
 6    5 Multilevel B-Spline Interpolation (from Grid)       FALSE
 7   NA                                        <NA>       FALSE
 8   NA                                        <NA>       FALSE
```

[8] http://sourceforge.net/projects/saga-gis/files/

[9] http://sourceforge.net/apps/trac/saga-gis/wiki/CompilingaLinuxUnicodeversion

[10] http://cran.r-project.org/web/packages/RSAGA/

[11] RPyGeo package can be used to control ArcGIS geoprocessor in a similar way.

[12] We also advise you to open SAGA and then first run processing manually (point–and–click) processing. The names of the SAGA libraries can be obtained by browsing the `/modules/` directory.

```
> rsaga.get.usage("grid_spline", 1)

  SAGA CMD 2.0.4
  library path:        C:/Progra~1/saga_vc/modules
  library name:        grid_spline
  module name :        Thin Plate Spline (Local)
  Usage: 1 [-GRID <str>] -SHAPES <str> [-FIELD <num>] [-TARGET <num>] [-REGUL <str>]
   [-RADIUS <str>] [-SELECT <num>] [-MAXPOINTS <num>] [-USER_CELL_SIZE <str>]
   [-USER_FIT_EXTENT] [-USER_X_EXTENT_MIN <str>] [-USER_X_EXTENT_MAX <str>]
   [-USER_Y_EXTENT_MIN <str>] [-USER_Y_EXTENT_MAX <str>] [-SYSTEM_SYSTEM_NX <num>]
   [-SYSTEM_SYSTEM_NY <num>] [-SYSTEM_SYSTEM_X <str>] [-SYSTEM_SYSTEM_Y <str>]
   [-SYSTEM_SYSTEM_D <str>] [-GRID_GRID <str>]
    -GRID:<str>                     Grid
          Data Object (optional output)
    -SHAPES:<str>                   Points
          Shapes (input)
    -FIELD:<num>                    Attribute
          Table field
    -TARGET:<num>                   Target Grid
          Choice
          Available Choices:
          [0] user defined
          [1] grid system
          [2] grid
    -REGUL:<str>                    Regularisation
          Floating point
    -RADIUS:<str>                   Search Radius
          Floating point
    -SELECT:<num>                   Points Selection
          Choice
          Available Choices:
          [0] all points in search radius
          [1] maximum number of points
    -MAXPOINTS:<num>                Maximum Number of Points
          Integer
          Minimum: 1.000000
    -USER_CELL_SIZE:<str>           Grid Size
          Floating point
          Minimum: 0.000000
    -USER_FIT_EXTENT                Fit Extent
          Boolean
    -USER_X_EXTENT_MIN:<str>        X-Extent
          Value range
    -USER_X_EXTENT_MAX:<str>        X-Extent
          Value range
    -USER_Y_EXTENT_MIN:<str>        Y-Extent
          Value range
    -USER_Y_EXTENT_MAX:<str>        Y-Extent
          Value range
    -SYSTEM_SYSTEM_NX:<num>         Grid System
          Grid system
    -SYSTEM_SYSTEM_NY:<num>         Grid System
          Grid system
    -SYSTEM_SYSTEM_X:<str>          Grid System
          Grid system
    -SYSTEM_SYSTEM_Y:<str>          Grid System
          Grid system
    -SYSTEM_SYSTEM_D:<str>          Grid System
          Grid system
    -GRID_GRID:<str>                Grid
          Grid (input)
```

Most SAGA modules — with the exception of a few that can only be executed in interactive mode — can be run from within R using the rsaga.geoprocessor function, RSAGA's low-level workhorse. However, RSAGA also provides R wrapper functions and associated help pages for many SAGA modules. As an example, a slope raster can be calculated from a digital elevation model with SAGA's local morphometry module, which can be accessed with the rsaga.local.morphometry function or more specifically with rsaga.slope (Brenning, 2008).

SAGA can read directly ESRI shapefiles and table data. Grids need to be converted to the native SAGA grid format (*.sgrd). This raster format is now supported by GDAL (starting with GDAL 1.7.0) and can be read directly via the readGDAL method. Alternatively, you can convert some GDAL-supported formats to SAGA grids and back by using:

```
# write to SAGA grid:
> rsaga.esri.to.sgrd(in.grids="meuse_soil.asc", out.sgrd="meuse_soil.sgrd",
+       in.path=getwd())
# read SAGA grid:
> rsaga.sgrd.to.esri(in.sgrd="meuse_soil.sgrd", out.grids="meuse_soil.asc",
+       out.path=getwd())
```

SAGA grid comprises tree files:

(1.) *.sgrd — the header file with name, data format, XLL, YLL, rows columns, cell size, z-factor and no data value;

(2.) *.sdat - the raw data file;

(3.) *.hgrd - the history file;

In some cases, you might consider reading directly the raw data and header data to R, which can be done by using e.g.:

```
# read SAGA grid format:
> sgrd <- matrix((unlist(strsplit(readLines(file("meuse_soil.sgrd")), split="\t= "))),
+        ncol=2, byrow=T)
> sgrd
```

```
      [,1]              [,2]
 [1,] "NAME"            "meuse_soil"
 [2,] "DESCRIPTION"     "UNIT"
 [3,] "DATAFILE_OFFSET" "0"
 [4,] "DATAFORMAT"      "FLOAT"
 [5,] "BYTEORDER_BIG"   "FALSE"
 [6,] "POSITION_XMIN"   "178460.0000000000"
 [7,] "POSITION_YMIN"   "329620.0000000000"
 [8,] "CELLCOUNT_X"     "78"
 [9,] "CELLCOUNT_Y"     "104"
[10,] "CELLSIZE"        "40.0000000000"
[11,] "Z_FACTOR"        "1.000000"
[12,] "NODATA_VALUE"    "-9999.000000"
[13,] "TOPTOBOTTOM"     "FALSE"
```

```
# read the raw data: 4bit, numeric (FLOAT), byte order small;
> sdat <- readBin("meuse_soil.sdat", what="numeric", size=4,
+      n=as.integer(sgrd[8,2])*as.integer(sgrd[9,2]))
> sdat.sp <- as.im(list(x=seq(from=as.integer(sgrd[6,2]),
+      length.out=as.integer(sgrd[8,2]), by=as.integer(sgrd[10,2])),
+      y=seq(from=as.integer(sgrd[7,2]), length.out=as.integer(sgrd[9,2]),
+      by=as.integer(sgrd[10,2])), z=matrix(sdat, nrow=as.integer(sgrd[8,2]),
+      ncol=as.integer(sgrd[9,2]))))
> sdat.sp <- as(sdat.sp, "SpatialGridDataFrame")
# replace the mask value with NA's:
> sdat.sp@data[[1]] <- ifelse(sdat.sp@data[[1]]==as.integer(sgrd[12,2]), NA,
+      sdat.sp@data[[1]])
> spplot(sdat.sp)
```

Another possibility to read SAGA grids directly to R is via the `read.sgrd` wrapper function (this uses `rsaga.sgrd.to.esri` method to write to a `tempfile()`), and then `read.ascii.grid` to import data to R):

```
> gridmaps <- readGDAL("meuse_soil.asc")
> gridmaps$soil <- as.vector(t(read.sgrd("meuse_soil.sgrd", return.header=FALSE)))
```

which reads raw data as a vector to an existing `SpatialGridDataFrame`.

SAGA offers limited capabilities for geostatistical analysis, but in a very user-friendly environment. Note that many commands in SAGA are available only by right-clicking the specific data layers. For example, you can make a correlation plot between two grids by right-clicking a map of interest, then select *Show Scatterplot* and you will receive a module execution window where you can select the second grid (or a shapefile) that you would like to correlate with your grid of interest. This will plot all grid-pairs and display the regression model and its R-square (see further Fig. 10.8). The setting of the Scatterplot options can be modified by selecting *Scatterplot* from the main menu. Here you can adjust the regression formula, obtain the regression details, and adjust the graphical settings of the scatterplot.

Under the module *Geostatistics*, three groups of operations can be found: (a) *Grid* (various operations on grids); (b) *Points* (derivation of semivariances) and (c) *Kriging* (ordinary and universal kriging). Under the group *Grid*, several modules can be run: *Multiple Regression Analysis* (relates point data with rasters), *Radius of Variance* (detects a minimum radius to reach a particular variance value in a given neighborhood), *Representativeness* (derives variance in a local neighborhood), *Residual Analysis* (derives local mean value, local difference from mean, local variance, range and percentile), *Statistics for Grids* (derives mean, range and standard deviation for a list of rasters) and *Zonal Grid Statistics* (derives statistical measures for various zones and based on multiple grids). For the purpose of geostatistical mapping, we are especially interested in correlating points with rasters (see §1.3.2), which can be done via the *Multiple Regression Analysis* module. Initiating this module opens a parameter setting window (Fig. 3.5).

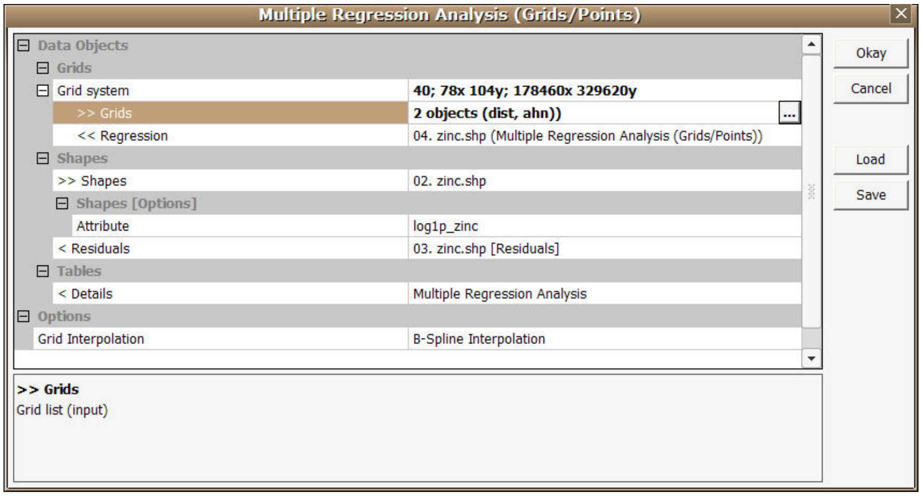
In the *Multiple Regression Analysis* module, SAGA will estimate the values of points at grids, run the regression analysis and predict the values at each location. You will also get a textual output (message window) that will show the regression model, and a list of the predictors according to their importance e.g.:

```
Executing module: Multiple Regression Analysis (Grids/Points)
Parameters
Grid system: 40; 78x 104y; 178460x 329620y
Grids: 2 objects (dist, ahn))
Shapes: zinc.shp
Attribute: log1p_zinc
Details: Multiple Regression Analysis
Residuals: zinc.shp [Residuals]
Regression: zinc.shp (Multiple Regression Analysis (Grids/Points))
Grid Interpolation: B-Spline Interpolation


Regression:
 Y = 6.651112 -2.474725*[dist] -0.116471*[ahn]

Correlation:
1: R2 = 54.695052% [54.695052%] -> dist
2: R2 = 55.268255% [0.573202%] -> ahn
```

in this case the most significant predictor is `dist`; the second predictor explains <1% of the variability in `log1p_zinc` (see further Fig. 5.6). The model explains 55.3% of the total variation.

When selecting the multiple regression analysis options, you can also opt to derive the residuals and fit the variogram of residuals. These will be written as a shapefile that can then be used to derive semivariances. Select *Geostatistics* ↦ *Points* ↦ *Semivariogram* and specify the distance increment (lag) and maximum distance. The variogram can be displayed by again right clicking a table and selecting *Show Scatterplot* option. Presently, the variogram (regression) models in SAGA are limited to linear, exponential and logarithmic models. In general, fitting and use of variograms in SAGA is discouraged[13].

[13] Exceptionally, you should use the logarithmic model which will estimate something close to the exponential variogram model (Eq.1.3.8).



Multiple Regression Analysis (Grids/Points)

Data Objects	
Grids	
Grid system	40; 78x 104y; 178460x 329620y
>> Grids	2 objects (dist, ahn))
<< Regression	04. zinc.shp (Multiple Regression Analysis (Grids/Points))
Shapes	
>> Shapes	02. zinc.shp
Shapes [Options]	
Attribute	log1p_zinc
< Residuals	03. zinc.shp [Residuals]
Tables	
< Details	Multiple Regression Analysis
Options	
Grid Interpolation	B-Spline Interpolation

>> Grids
Grid list (input)

Fig. 3.5: Running predictions by using regression analysis in SAGA GIS: parameter settings window. The "Grid Interpolation" setting indicates the way SAGA will estimate values of grids at calibration points. This should not be confused with other gridding techniques available in SAGA.

Once the regression model and the variogram of the residuals have been estimated, a user can also run regression-kriging, which is available in SAGA under the module *Geostatistics* ↦ *Universal kriging*. Global and local (search radius) version of the Universal kriging are available. Use of local Universal kriging with a small search radius ($\ll$100 points) is not recommended because it over-simplifies the technique, and can lead to artefacts[14]. Note also that, in SAGA, you can select as many predictors as you wish, as long as they are all in the same grid system. The final results can be visualized in both 2D and 3D spaces.

Another advantage of SAGA is the ability to use script files for the automation of complex work-flows, which can then be applied to different data projects. Scripting of SAGA modules is now possible in two ways:

(1.) Using the command line interpreter (`saga_cmd.exe`) with DOS batch scripts. Some instructions on how to generate batch files can be found in Conrad (2006, 2007).

(2.) A much more flexible way of scripting utilizes the Python interface to the SAGA Application Programming Interface (SAGA-API).

In addition to scripting possibilities, SAGA allows you to save SAGA parameter files (`*.sprm`) that contain all inputs and output parameters set using the module execution window. These parameter files can be edited in an ASCII editor, which can be quite useful to automate processing.

In summary, SAGA GIS has many attractive features for both geographical and statistical analysis of spatial data: (1) it has a large library of modules, especially to parameterize geomorphometric features, (2) it can generate maps from points and rasters by using multiple linear regression and regression-kriging, and (3) it is an open source GIS with a popular GUI. Compared to `gstat`, SAGA is not able to run geostatistical simulations, GLS estimation nor stratified or co-kriging. However, it is capable of running regression-kriging in a statistically sound way (unlike ILWIS). The advantage of SAGA over R is that it can load and process relatively large maps (not recommended in R for example) and that it can be used to visualize the input and output maps in 2D and 2.5D (see further section 5.5.2).

3.1.3 GRASS GIS

GRASS[15] (**Geographic Resources Analysis Support System**) is a general-purpose Geographic Information System (GIS) for the management, processing, analysis, modeling and visualization of many types of georeferenced data. It is Open Source software released under GNU General Public License and is available on

[14] Neither local variograms nor local regression models are estimated. See §2.2 for a detailed discussion.

[15] `http://grass.itc.it`

the three major platforms (Microsfot Windows, Mac OS X and Linux). The main component of the development and software maintenance is built on top of highly automated web-based infrastructure sponsored by ITC-irst (Centre for Scientific and Technological Research) in Trento, Italy with numerous worldwide mirror sites. GRASS includes functions to process raster maps, including derivation of descriptive statistics for maps, histograms, but also generation of statistics for time series. There are also several unique interpolation techniques. For example the Regularized Spline with Tension (RST) interpolation, which has been quoted as one of the most sophisticated methods to generate smooth surfaces from point data (Mitasova et al., 2005).

In version 5.0 of GRASS, several basic geostatistical functionalities existed including ordinary kriging and variogram plotting, however, developers of GRASS ultimately concluded that there was no need to build geostatistical functionality from scratch when a complete open source package already existed. The current philosophy (v 6.5) focuses on making GRASS functions also available in R, so that both GIS and statistical operations can be integrated in a single command line. A complete overview of the Geostatistics and spatial data analysis functionality can be found via the GRASS website[16]. Certainly, if you are a Linux user and already familiar with GRASS, you will probably not encounter many problems in installing GRASS and using the syntax.

Unlike SAGA, GRASS requires that you set some initial '*environmental*' parameters, i.e. initial setting that describe your project. There are three initial environmental parameters: `DATABASE` — a directory (folder) on disk to contain all GRASS maps and data; `LOCATION` — the name of a geographic location (defined by a co-ordinate system and a rectangular boundary), and `MAPSET` — a rectangular REGION and a set of maps (Neteler and Mitasova, 2008). Every `LOCATION` contains at least a `MAPSET` called `PERMANENT`, which is readable by all sessions. GRASS locations are actually powerful abstractions that do resemble the way in which workflows were/are set up in larger multi-user projects. The mapsets parameter is used to distinguish users, and `PERMANENT` was privileged with regard to who could change it — often the database/location/mapset tree components can be on different physical file systems. On single-user systems or projects, this construction seems irrelevant, but it isn't when many users work collaborating on the same location.

GRASS can be controlled from R thanks to the spgrass6[17] package (Bivand, 2005; Bivand et al., 2008): `initGRASS` can be used to define the environmental parameters; description of each GRASS module can be obtained by using the `parseGRASS` method. The recommended reference manual for GRASS is the "*GRASS book*" (Neteler and Mitasova, 2008); a complete list of the modules can be found in the GRASS reference manual[18]. Some examples of how to use GRASS via R are shown in §10.6.2. Another powerful combo of applications similar to the one shown in Fig. 3.1 is the QGIS+GRASS+R triangle. In this case, a GUI (QGIS) stands on top of GRASS (which stands *on top* of R), so that this combination is worth checking for users that prefer GUI's.

3.2 Statistical computing: R

R[19] is the open source implementation of the S language for statistical computing (R Development Core Team, 2009). Apparently, the name "R" was selected for two reasons: (1) precedence — "R" is a letter before "S", and (2) coincidence — both of the creators' names start with a letter "R". The S language provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering,. . .) and graphical techniques, and is highly extensible (Chambers and Hastie, 1992; Venables and Ripley, 2002). It has often been the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

Although much of the R code is always under development, a large part of the code is usable, portable and extendible. This makes R one of the most suitable coding environments for academic societies. Although it typically takes a lot of time for non-computer scientists to learn the R syntax, the benefits are worth the time investment.

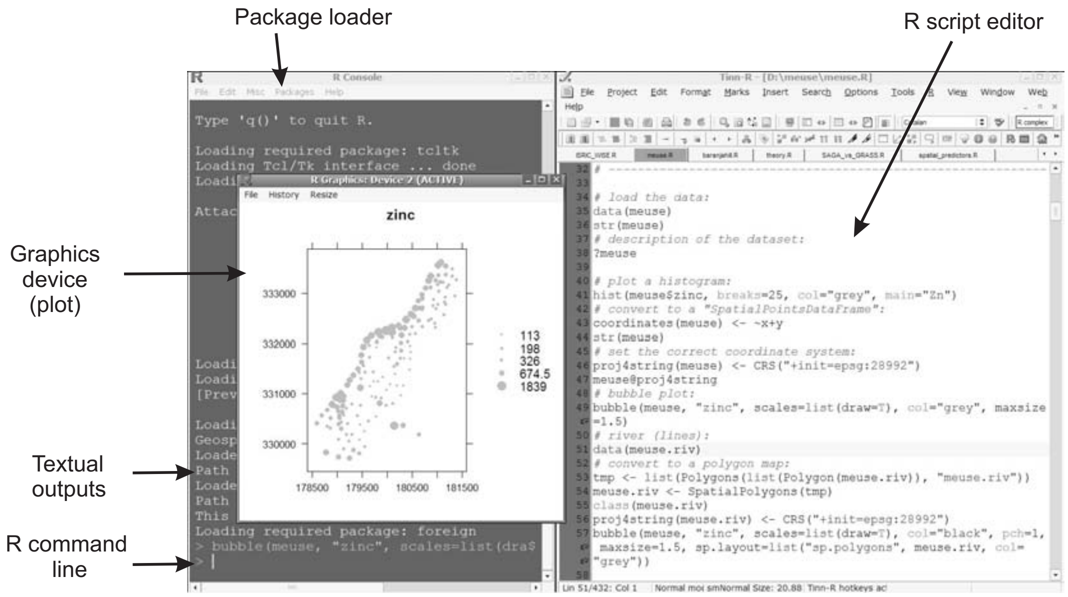
To install R under Windows, download and run an installation executable file from the R-project homepage. This will install R for Windows with a GUI. After starting R, you will first need to set-up the working directory and install additional packages. To run geostatistical analysis in R, you will need to add the following R packages: gstat (gtat in R), rgdal (GDAL import of GIS layers in R), sp (support for spatial objects in R),

[16] `http://grass.itc.it/statsgrass/`

[17] `http://cran.r-project.org/web/packages/spgrass6/`

[18] See your local installation `file:///C:/GRASS/docs/html/full_index.html`.

[19] `http://www.r-project.org`



```
Type 'q()' to quit R.

Loading required package: tcltk
Loading Tcl/Tk interface ... done
Loadi

Attac




Loadi
Loadi
[Prev

Loadi
Geosp
Loade
Path
Loade
Path
This
Loading required package: foreign
> bubble(meuse, "zinc", scales=list(dra$
> |
```

```
# -----------------------------------------------------------

# load the data:
data(meuse)
str(meuse)
# description of the dataset:
?meuse

# plot a histogram:
hist(meuse$zinc, breaks=25, col="grey", main="Zn")
# convert to a "SpatialPointsDataFrame":
coordinates(meuse) <- ~x+y
str(meuse)
# set the correct coordinate system:
proj4string(meuse) <- CRS("+init=epsg:28992")
meuse@proj4string
# bubble plot:
bubble(meuse, "zinc", scales=list(draw=T), col="grey", maxsize
=1.5)
# river (lines):
data(meuse.riv)
# convert to a polygon map:
tmp <- list(Polygons(list(Polygon(meuse.riv)), "meuse.riv"))
meuse.riv <- SpatialPolygons(tmp)
class(meuse.riv)
proj4string(meuse.riv) <- CRS("+init=epsg:28992")
bubble(meuse, "zinc", scales=list(draw=T), col="black", pch=1,
 maxsize=1.5, sp.layout=list("sp.polygons", meuse.riv, col=
"grey"))
```

Fig. 3.6: The basic GUI of R under Windows (controlled using **Tinn-R**) and a typical plot produced using the sp package.

spatstat (spatial statistics in R) and maptools.

To install these packages you should do the following. First start the R GUI, then select the *Packages ↦ Load package* from the main menu. Note that, if you wish to install a package on the fly, you will need to select a suitable CRAN mirror from which it will download and unpack a package. Another quick way to get all packages used in R to do spatial analysis[20] (as explained in Bivand et al. (2008)) is to install the ctv package and then execute the command:

```
> install.packages("ctv")
> library(ctv)
> install.views("Spatial")
```

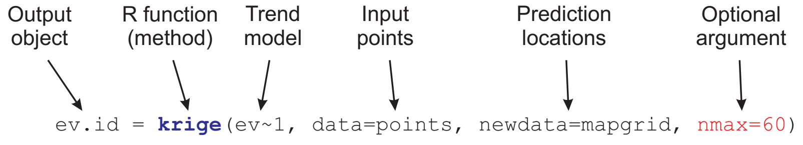
After you install a package, you will still need to load it into your workspace (every time you start R) before you can use its functionality. A package is commonly loaded using e.g.:

```
> library(gstat)
```

Much of information about some package can be found in the package installation directory in sub-directory `html`, or can be called directly from the R session. For example, important information on how to add more rgdal drivers can be found in the attached documentation:

```
> file.show(system.file("README.windows", package="rgdal"))
```

R is an object-based, functional language. Typically, a function in R consists of three items: its arguments, its body and its environment. A function is *invoked* by a name, followed by its arguments. Arguments themselves can be positional or named and can have defaults in which case they can be omitted (see also p.32):



```
ev.id = krige(ev~1, data=points, newdata=mapgrid, nmax=60)
```

[20] `http://cran.r-project.org/web/views/Spatial.html`

Functions typically return their result as their value, not via an argument. In fact, if the body of a function changes an argument it is only changing a local copy of the argument and the calling program does not get the changed result.

R is widely recognized as one of the fastest growing and most comprehensive statistical computing tools[21]. It is estimated that the current number of active R users (Google trends service) is about 430k, but this number is constantly growing. R practically offers statistical analysis and visualization of unlimited sophistication. A user is not restricted to a small set of procedures or options, and because of the contributed packages, users are not limited to one method of accomplishing a given computation or graphical presentation. As we will see later, R became attractive for geostatistical mapping mainly due to the recent integration of the geostatistical tools (gstat, geoR) and tools that allow R computations with spatial data layers (sp, maptools, raster and similar).

Note that in R, the user must type commands to enter data, do analyzes, and plot graphs. This might seem inefficient to users familiar with MS Excel and similar intuitive, point-and-click packages. If a single argument in the command is incorrect, inappropriate or mistyped, you will get an error message indicating where the problem might be. If the error message is not helpful, then try receiving more help about some operation. Many very useful introductory notes and books, including translations of manuals into other languages than English, are available from the documentation section[22]. Another very useful source of information is the R News[23] newsletter, which often offers many practical examples of data processing. Vol. 1/2 of R News, for example, is completely dedicated to spatial statistics in R; see also Pebesma and Bivand (2005) for an overview of classes and methods for spatial data in R. The '*Spatial*' packages can be nicely combined with e.g. the '*Environmentrics*'[24] packages. The interactive graphics[25] in R is also increasingly powerful (Urbanek and Theus, 2008). To really get an idea about the recent developments, and to get support with using spatial packages, you should register with the special interest group R-sig-Geo[26].

Although there are several packages in R to do geostatistical analysis and mapping, many recognize R+gstat/geoR as the only complete and fully-operational packages, especially if you wish to run regression-kriging, multivariate analysis, geostatistical simulations and block predictions (Hengl et al., 2007a; Rossiter, 2007). To allow extension of R functionalities to operations with spatial data, the developer of gstat, with the support of colleagues, has developed the sp[27] package (Pebesma and Bivand, 2005; Bivand et al., 2008). Now, users are able to load GIS layers directly into R, run geostatistical analysis on grid and points and display spatial layers as in a standard GIS package. In addition to sp, two important spatial data protocols have also been recently integrated into R: (1) **GIS data exchange protocols** (GDAL — Geospatial Data Abstraction Library, and OGR[28] — OpenGIS Simple Features Reference Implementation), and (2) **map projection protocols** (PROJ.4[29] — Cartographic Projections Library). These allow R users to import/export raster and vector maps, run raster/vector based operations and combine them with statistical computing functionality of various packages. The development of GIS and graphical functionalities within R has already caused a small revolution and many GIS analysts are seriously thinking about completely shifting to R.

3.2.1 gstat

gstat[30] is a stand-alone package for geostatistical analysis developed by Edzer Pebesma during his PhD studies at the University of Utrecht in the Netherlands in 1997. As of 2003, the gstat functionality is also available as an S extension, either as R package or S-Plus library. Current development focuses mainly on the R/S extensions, although the stand alone version can still be used for many applications. To install gstat (the stand-alone version) under Windows, download the `gstat.exe` and `gstatw.exe` (variogram modeling with GUI) files from the gstat.org website and put them in your system directory[31]. Then, you can always run gstat from the Windows start menu. The `gstat.exe` runs as a DOS application, which means that there is no GUI.

[21] The article in the New Your Times by Vance (2009) has caused much attention.

[22] `http://www.r-project.org/doc/bib/R-books.html`

[23] `http://cran.r-project.org/doc/Rnews/`; now superseded by R Journal.

[24] `http://cran.r-project.org/web/views/Environmetrics.html`

[25] `http://www.interactivegraphics.org`

[26] `https://stat.ethz.ch/mailman/listinfo/r-sig-geo`

[27] `http://r-spatial.sourceforge.net`

[28] `http://www.gdal.org/ogr/` — for Mac OS X users, there is no binary package available from CRAN.

[29] `http://proj.maptools.org`

[30] `http://www.gstat.org`

[31] E.g. `C:\Windows\system32\`

A user controls the processing by editing the command files.

gstat is possibly the most complete, and certainly the most accessible geostatistical package in the World. It can be used to calculate sample variograms, fit valid models, plot variograms, calculate (pseudo) cross variograms, and calculate and fit directional variograms and variogram models (anisotropy coefficients are not fitted automatically). Kriging and (sequential) conditional simulation can be done under (simplifications of) the universal co-kriging model. Any number of variables may be spatially cross-correlated. Each variable may have its own number of trend functions specified (being coordinates, or so-called external drift variables). Simplifications of this model include ordinary and simple kriging, ordinary or simple co-kriging, universal kriging, external drift kriging, Gaussian conditional or unconditional simulation or cosimulation. In addition, variables may share trend coefficients (e.g. for collocated co-kriging). To learn about capabilities of gstat, a user is advised to read the gstat User's manual[32], which is still by far the most complete documentation of the gstat.

A complete overview of gstat functions and examples of R commands are given in Pebesma (2004). A more recent update of the functionality can be found in Bivand et al. (2008, §8); gstat development tree is from 2010 also available from a public SVN[33]. The most widely used gstat functions in R include:

- variogram — calculates sample (experimental) variograms;
- plot.variogram — plots an experimental variogram with automatic detection of lag spacing and maximum distance;
- fit.variogram — iteratively fits an experimental variogram using reweighted least squares estimation;
- krige — a generic function to make predictions by inverse distance interpolation, ordinary kriging, OLS regression, regression-kriging and co-kriging;
- krige.cv — runs krige with cross-validation using the *n*-fold or *leave-one-out* method;

R offers much more flexibility than the stand-alone version of gstat, because users can extend the optional arguments and combine them with outputs or functions derived from other R packages. For example, instead of using a trend model with a constant (intercept), one could use outputs of a linear model fitting, which allows even more compact scripting.

3.2.2 The stand-alone version of gstat

As mentioned previously, gstat can be run as a stand-alone application, or as a R package. In the stand-alone version of the gstat, everything is done via compact scripts or command files. The best approach to prepare the command files is to learn from the list of example command files that can be found in the gstat User's manual[34]. Preparing the command files for gstat is rather simple and fast. For example, to run inverse distance interpolation the command file would look like this:

```
#  Inverse distance interpolation on a mask map

data(zinc): 'meuse.eas', x=1, y=2, v=3;
mask: 'dist.asc'; # the prediction locations
predictions(zinc): 'zinc_idw.asc'; # result map
```

where the first line defines the input point data set (points.eas — an input table in the GeoEAS[35] format), the coordinate columns (x, y) are the first and the second column in this table, and the variable of interest is in the third column; the prediction locations are the grid nodes of the map dist.asc[36] and the results of interpolation will be written to a raster map zinc_idw.asc.

To extend the predictions to regression-kriging, the command file needs to include the auxiliary maps and the variogram model for the residuals:

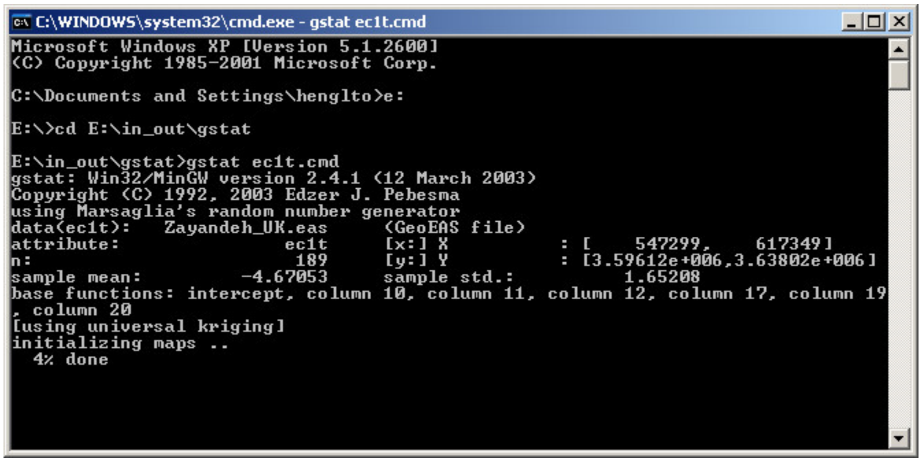
[32] http://gstat.org/manual/
[33] https://52north.org/svn/geostatistics/
[34] http://gstat.org/manual/node30.html
[35] http://www.epa.gov/ada/csmos/models/geoeas.html
[36] Typically ArcInfo ASCII format for raster maps.

```
#  Regression-kriging using two auxiliary maps
#  Target variable is log-transformed

data(ln_zinc): 'meuse.eas', x=1, y=2, v=3, X=4,5, log;
variogram(ln_zinc): 0.055 Nug(0) + 0.156 Exp(374);
mask: 'dist.asc', 'ahn.asc'; # the predictors
predictions(ev): 'ln_zinc_rk.asc'; # result map
```

where `X` defines the auxiliary predictors, `0.055 Nug(0) + 0.156 Exp(374)` is the variogram of residuals and `dist.asc` and `ahn.asc` are the auxiliary predictors. All auxiliary maps need to have the same grid definition and need to be available also in the input table. In addition, the predictors need to be sorted in the same order in both the first and the third line. Note that there are many optional arguments that can be included in the command file: a search radius can be set using "`max=50`"; switching from predictions to simulations can be done using "`method: gs`"; bloc kriging can be initiated using "`blocksize: dx=100`".

To run a command file start DOS prompt by typing: `> cmd`, then move to the active directory by typing: e.g. `> cd c:\gstat`; to run spatial predictions or simulations run the `gstat` program together with a specific `gstat` command file from the Windows cmd console (Fig. 3.7):

```
cmd gstat.exe ec1t.cmd
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\henglto>e:

E:\>cd E:\in_out\gstat

E:\in_out\gstat>gstat ec1t.cmd
gstat: Win32/MinGW version 2.4.1 (12 March 2003)
Copyright (C) 1992, 2003 Edzer J. Pebesma
using Marsaglia's random number generator
data(ec1t):   Zayandeh_UK.eas      (GeoEAS file)
attribute:                ec1t      [x:] X           : [     547299,    617349]
n:                         189      [y:] Y           : [3.59612e+006,3.63802e+006]
sample mean:          -4.67053      sample std.:          1.65208
base functions: intercept, column 10, column 11, column 12, column 17, column 19
, column 20
[using universal kriging]
initializing maps ..
  4% done
```

Fig. 3.7: Running interpolation using the `gstat` stand-alone: the DOS command prompt.

`gstat` can also automatically fit a variogram by using:

```
data(ev): 'points.eas', x=1,y=2,v=3;

# set an initial variogram:
variogram(ev): 1 Nug(0) + 5 Exp(1000);
# fit the variogram using standard weights:
method: semivariogram;
set fit=7;

# write the fitted variogram model and the corresponding gnuplot
set output= 'vgm_ev.cmd';
set plotfile= 'vgm_ev.plt';
```

where set `fit=7` defines the fitting method (weights=$N_j/\mathbf{h}_j^2$), `vgm_ev.cmd` is the text file where the fitted parameters will be written. Once you fitted a variogram, you can then view it using the `wgnuplot`[37]. Note that, for automated modeling of variogram, you will need to define the fitting method and an initial variogram, which is then iteratively fitted against the sampled values. A reasonable initial exponential variogram can be

[37] `http://www.gnuplot.info`

produced by setting nugget parameter = measurement error, sill parameter = sampled variance, and range parameter = 10% of the spatial extent of the data (or two times the mean distance to the nearest neighbor). This can be termed a **standard initial variogram model**. Although the true variogram can be quite different, it is important to have a good idea of what the variogram *should* look like.

There are many advantages to using the stand-alone version of gstat. The biggest ones are that it takes little time to prepare a script and that it can work with large maps (unlike R that often faces memory problems). In addition, the results of interpolation are directly saved in a GIS format and can be loaded to ILWIS or SAGA. However, for regression-kriging, we need to estimate the regression model first, then derive the residuals and estimate their variogram model, which cannot be automated in gstat so we must in any case load the data to some statistical package before we can prepare the command file. Hence, the stand-alone version of gstat, as with SAGA, can be used for geostatistical mapping only after regression modeling and variogram fitting have been completed.

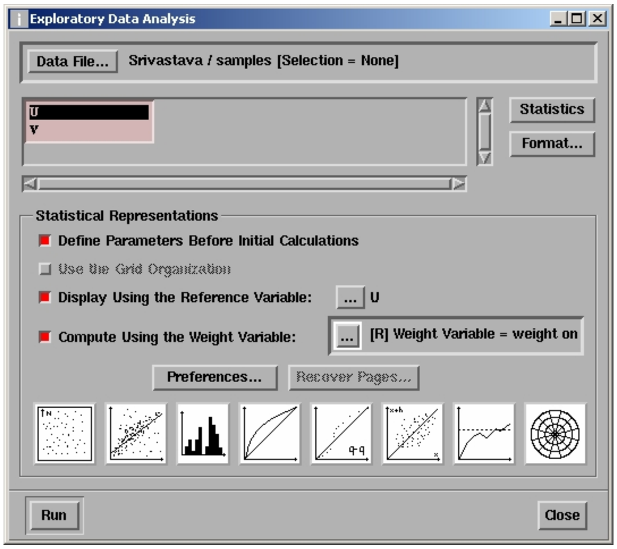
3.2.3 geoR

An equally comprehensive package for geostatistical analysis is geoR, a package extensively described by Diggle and Ribeiro Jr (2007) and Ribeiro Jr et al. (2003); a series of tutorials can be found on the package homepage[38]. In principle, a large part of the functionality of gstat and geoR overlap[39]; on the other hand, geoR has many original methods, including an original format for spatial data (called geodata). geoR can be in general considered to be more suited for variogram model estimation (including interactive visual fitting), modeling of non-normal variables and simulations. A short demo of what geoR can do considering standard geostatistical analysis is given in section 5.5.3.

geoR also allows for Bayesian kriging; its extension — the package geoRglm — can work with binomial and Poisson processes (Ribeiro Jr et al., 2003). In comparison, fitting a Generalized Linear Geostatistical Model (GLGM) can be more conclusive than fitting simple linear models in gstat since we can model the geographical and regression terms more objectively (Diggle and Ribeiro Jr, 2007). This was, for example, the original motivation for the geoRglm and spBayes packages (Ribeiro Jr et al., 2003). However, GLGMs are not yet ready for operational mapping, and R code will need to be adapted.

3.2.4 Isatis

Isatis[40] is probably the most expensive geostatistical package (>10K €) available on the market today. However, it is widely regarded as one of the most professional packages for environmental sciences. Isatis was originally built for Unix, but there are MS Windows and Linux versions also. From the launch of the package in 1993, >1000 licences have been purchased worldwide. Standard Isatis clients are Oil and Gas companies, consultancy teams, mining corporations and environmental agencies. The software will be here shortly presented for informative purposes. We will not use Isatis in the exercises, but it is worth mentioning it.



Data File... Srivastava / samples [Selection = None]

U
V

Statistical Representations

■ Define Parameters Before Initial Calculations
□ Use the Grid Organization
■ Display Using the Reference Variable: ... U
■ Compute Using the Weight Variable: ... [R] Weight Variable = weight on

Fig. 3.8: Exploratory data analysis possibilities in Isatis.

Isatis offers a wide range of geostatistical functions ranging from 2D/3D isotropic and directional variogram modeling, univariate and multivariate kriging, punctual and block estimation, drift estimation, universal kriging, col-

[38] http://www.leg.ufpr.br/geoR/geoRdoc/tutorials.html

[39] Other comparable packages with geostatistical analysis are fields, spatial, sgeostat and RandomFields, but this book for practical reasons focuses only on gstat and geoR.

[40] http://www.geovariances.com — the name "Isatis" is not an abbreviation. Apparently, the creators of Isatis were passionate climbers so they name their package after one climbing site in France.

located co-kriging, kriging with external drift, kriging with inequalities (introduce localized constraints to bound the model), factorial kriging, disjunctive kriging etc. Isatis especially excels with respect to interactivity of exploratory analysis, variogram modeling, detection of local outliers and anisotropy (Fig. 3.8).

Regression-kriging in Isatis can be run by selecting *Interpolation* ↦ *Estimation* ↦ *External Drift (Co)-kriging*. Here you will need to select the target variable (point map), predictors and the variogram model for residuals. You can import the point and raster maps as shapefiles and raster maps as ArcView ASCII grids (importing/exporting options are limited to standard GIS formats). Note that, before you can do any analysis, you first need to define the project name and working directory using the data file manager. After you import the two maps, you can visualize them using the display launcher.

Note that KED in Isatis is limited to only one (three when scripts are used) auxiliary raster map (called *background variable* in Isatis). Isatis justifies limitation of number of auxiliary predictors by computational efficiency. In any case, a user can first run factor analysis on multiple predictors and then select the most significant component, or simply use the regression estimates as the auxiliary map. Isatis offers a variety of options for the automated fitting of variograms. You can also edit the *Model Parameter File* where the characteristics of the model you wish to apply for kriging are stored.

3.3 Geographical visualization: Google Earth (GE)

Google Earth[41], Google's geographical browser, is increasingly popular in the research community. Google Earth was developed by Keyhole, Inc., a company acquired by Google in 2004. The product was renamed Google Earth in 2005 and is currently available for use on personal computers running Microsoft Windows 2000, XP or Vista, Mac OS X and Linux. All displays in Google Earth are controlled by KML files, which are written in the **Keyhole Markup Language**[42] developed by Keyhole, Inc. KML is an XML-based language for managing the display of three-dimensional geospatial data, and is used in several geographical browsers (Google Maps, Google Mobile, ArcGIS Explorer and World Wind). The KML file specifies a set of standard features (placemarks, images, polygons, 3D models, textual descriptions, etc.) for display in Google Earth. Each place always has a longitude and a latitude. Other data can make the view more specific, such as tilt, heading, altitude, which together define a *camera view*. The KML data sets can be edited using an ASCII editor (as with HTML), but they can be edited also directly in Google Earth. KML files are very often distributed as KMZ files, which are zipped KML files with a `.kmz` extension. Google has recently '*given away*' KML to the general public, i.e. it has been registered as an **OGC standard**[43].

To install Google Earth, run the installer that you can obtain from Google's website. To start a KML file, just double-click it and the map will be displayed using the default settings. Other standard background layers, such as roads, borders, places and similar geographic features, can be turned on or off using the Layers panel. There are also commercial Plus and Pro versions of Google Earth, but for purpose of our exercises, the free version has all necessary functionality.

The rapid emergence and uptake of Google Earth may be considered evidence for a trend towards a more visual approach to spatial data handling. Google Earth's sophisticated spatial indexing of very large data sets combined with an open architecture for integrating and customizing new data is having a radical effect on many Geographic Information Systems (Wood, 2008; Craglia et al., 2008). If we extrapolate these trends, we could foresee that in 10 to 20 years time Google Earth will contain near to real-time global satellite imagery of photographic quality, all cars and transportation vehicles (even people) will be GPS-tagged, almost every data will have spatio-temporal reference, and the Virtual Globe will be a digital 4D replica of Earth in the scale 1:1 (regardless of how 'scary' that seems).

One of biggest impacts of Google Earth and Maps is that they have opened up the exploration of spatial data to a much wider community of non-expert users. Google Earth is a ground braking software in at least five categories:

Availability — It is a free browser that is available to everyone. Likewise, users can upload their own geographic data and share it with anybody (or with selected users only).

High quality background maps — The background maps (remotely sensed images, roads, administrative units, topography and other layers) are constantly updated and improved. At the moment, almost

[41] `http://earth.google.com`

[42] `http://earth.google.com/kml/kml_tut.html` / `http://code.google.com/apis/kml/`

[43] `http://www.opengeospatial.org/standards/kml/`

30% of the world is available in high resolution (2 m IKONOS images[44]). All these layers have been georeferenced at relatively high accuracy (horizontal RMSE of 20–30 m or better) and can be used to validate the spatial accuracy of moderate-resolution remote sensing products and similar GIS layers (Potere, 2008). Overlaying GIS layers of unknown accuracy on GE can be revealing. Google has recently agreed to license imagery for their mapping products from the GeoEye-1 satellite. In the near future, Google plans to update 50 cm resolution imagery with near to global coverage on monthly basis.

A single coordinate system — The geographic data in Google Earth is visualized using a 3D model (central projection) rather than a projected 2D system. This practically eliminates all the headaches associated with understanding projection systems and merging maps from different projection systems. However, always have in mind that any printed Google Earth display, although it might appear to be 2D, will always show distortions due to Earth's curvature (or due to relief displacements). At very detailed scales (blocks of the buildings), these distortions can be ignored so that distances on the screen correspond closely to distances on the ground.

Web-based data sharing — Google Earth data is located on internet servers so that the users do not need to download or install any data locally. Rasters are distributed through a tiling system: by zooming in or out of the map, only local tiles at different scales (19 zoom levels) will be downloaded from the server.

Popular interface — Google Earth, as with many other Google products, is completely user-oriented. What makes Google Earth especially popular is the impression of literarily flying over Earth's surface and interactively exploring the content of various spatial layers.

API services — A variety of geo-services are available that can be used via Java programming interface or similar. By using the Google Maps API one can geocode the addresses, downloading various static maps and attached information. Google Maps API service in fact allow mash-ups that often exceed what the creators of software had originally in mind.

There are several competitors to Google Earth (NASA's World Wind[45], ArcGIS Explorer[46], 3D Weather Globe[47], Capaware[48]), although none of them are equivalent to Google Earth in all of the above-listed aspects. On the other hand, Google Earth poses some copyright limitations, so you should first read the *Terms and Conditions*[49] before you decide to use it for your own projects. For example, Google welcomes you to use any of the multimedia produced using Google tools as long as you preserve the copyrights and attributions including the Google logo attribution. However, you cannot sell these to others, provide them as part of a service, or use them in a commercial product such as a book or TV show without first getting a rights clearance from Google.

While at present Google Earth is primarily used as a geo-browser for exploring spatially referenced data, its functionality can be integrated with geostatistical tools, which can stimulate sharing of environmental data between international agencies and research groups. Although Google Earth does not really offer much standard GIS functionality, it can be used also to add content, such as points or lines to existing maps, measure areas and distances, derive UTM coordinates and eventually load GPS data. Still, the main use of Google Earth depends on its visualization capabilities that cannot be compared to any desktop GIS. The base maps in Google Earth are extensive and of high quality, both considering spatial accuracy and content. In that sense, Google Earth is a GIS that exceeds any existing public GIS in the world.

To load your own GIS data to Google Earth, there are several possibilities. First, you need to understand that there is a difference between loading vector and raster maps into Google Earth. Typically, it is relatively easy to load vector data such as points or lines into Google Earth, and somewhat more complicated to do the same with raster maps. Note also that, because Google Earth works exclusively with Latitude/Longitude projection system (WGS84[50] ellipsoid), all vector/raster maps need to be first reprojected before they can be

[44] Microsoft recently released the Bing Maps 3D (`http://www.bing.com/maps/`) service that also has an impressive map/image coverage of the globe.

[45] `http://worldwind.arc.nasa.gov/`

[46] `http://www.esri.com/software/arcgis/explorer/`

[47] `http://www.mackiev.com/3d_globe.html`

[48] `http://www.capaware.org`

[49] `http://www.google.com/permissions/geoguidelines.html`

[50] `http://earth-info.nga.mil/GandG/wgs84/`

exported to KML format. More about importing the data to Google Earth can be found via the Google Earth User Guide[51].

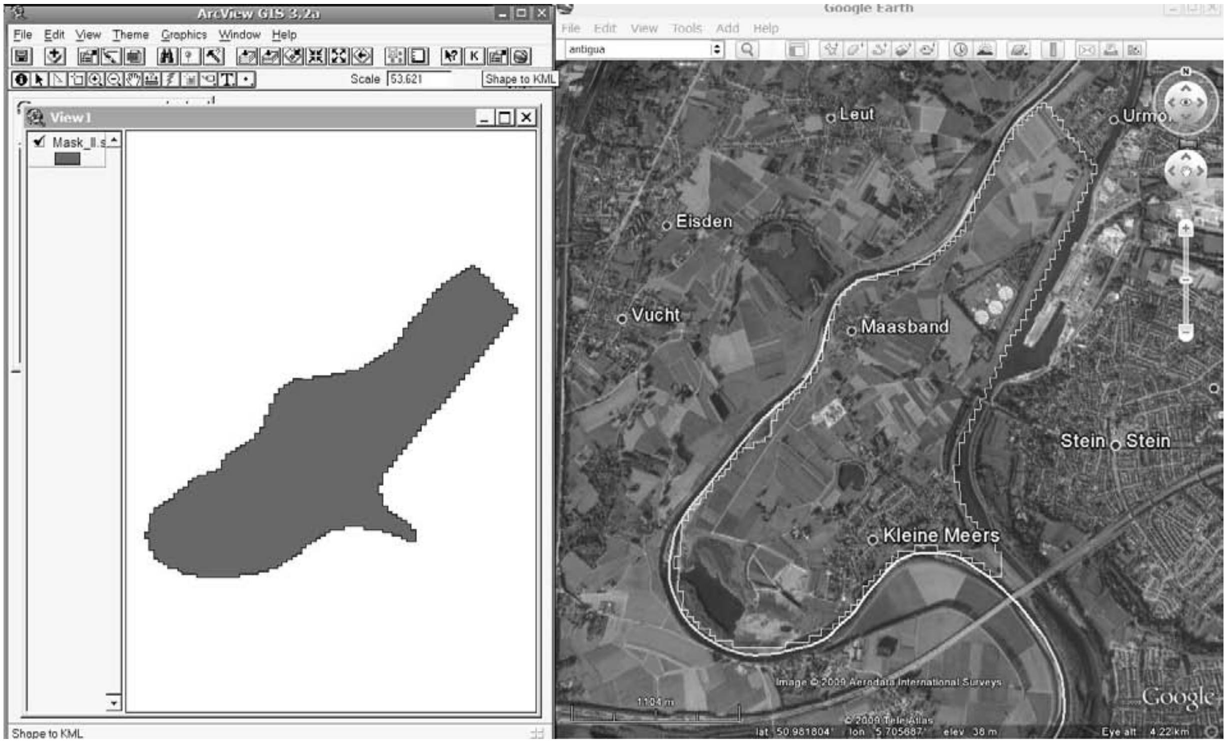


Fig. 3.9: Exporting ESRI shapefiles to KML using the SHAPE 2 KML ESRI script in ArcView 3.2. Note that the vector maps need to be first reprojected to LatLon WGS84 system.

3.3.1 Exporting vector maps to KML

Vector maps can be loaded by using various plugins/scripts in packages such as ArcView, MapWindow and R. Shapefiles can be directly transported to KML format by using ArcView's SHAPE 2 KML[52] script, courtesy of Domenico Ciavarella. To install this script, download it, unzip it and copy the two files to your ArcView 3.2 program directory:

- `..\ARCVIEW\EXT32\shape2KML.avx`
- `..\ARCVIEW\ETC\shp2kmlSource.apr`

This will install an extension that can be easily started from the main program menu (Fig. 3.9). Now you can open a layer that you wish to convert to KML and then click on the button to enter some additional parameters. There is also a commercial plugin for ArcGIS called Arc2Earth[53], which offers various export options. An alternative way to export shapefiles to KML is the Shape2Earth plugin[54] for the open-source GIS MapWindow. Although MapWindow is an open-source GIS, the Shape2Earth plugin is shareware so you might need to purchase it.

To export point or line features to KML in R, you can use the `writeOGR` method available in rgdal package. Export can be achieved in three steps, e.g.:

```
# 1. Load the rgdal package for GIS data exchange:
> require(c("rgdal","gstat","lattice","RASAGA","maptools","akima"))
```

[51] http://earth.google.com/userguide/v4/ug_importdata.html
[52] http://arcscripts.esri.com/details.asp?dbid=14254
[53] http://www.arc2earth.com
[54] http://www.mapwindow.org/download.php?show_details=29

```
# 2. Reproject the original map from local coordinates:
> data(meuse)
> coordinates(meuse) <- ~ x+y
> proj4string(meuse) <- CRS("+init=epsg:28992")
> meuse.ll <- spTransform(meuse, CRS("+proj=longlat +datum=WGS84"))
# 3. Export the point map using the "KML" OGR driver:
> writeOGR(meuse.ll["lead"], "meuse_lead.kml", "lead", "KML")
```

See further p.119 for instructions on how to correctly set-up the coordinate system for the meuse case study. A more sophisticated way to generate a KML is to directly write to a KML file using loops. This way one has a full control of the visualization parameters. For example, to produce a bubble-type of plot (compare with Fig. 5.2) in Google Earth with actual numbers attached as labels to a point map, we can do:

```
> varname <- "lead"  # variable name
> maxvar <- max(meuse.ll[varname]@data)  # maximum value
> filename <- file(paste(varname, "_bubble.kml", sep=""), "w")
> write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>", filename)
> write("<kml xmlns=\"http://earth.google.com/kml/2.2\">", filename, append = TRUE)
> write("<Document>", filename, append = TRUE)
> write(paste("<name>", varname, "</name>", sep=" "), filename, append = TRUE)
> write("<open>1</open>", filename, append = TRUE)
# Write points in a loop:
> for (i in 1:length(meuse.ll@data[[1]])) {
>   write(paste('  <Style id="','pnt', i,'">',sep=""), filename, append = TRUE)
>   write("    <LabelStyle>", filename, append = TRUE)
>   write("     <scale>0.7</scale>", filename, append = TRUE)
>   write("    </LabelStyle>", filename, append = TRUE)
>   write("                <IconStyle>", filename, append = TRUE)
>   write("   <color>ff0000ff</color>", filename, append = TRUE)
>   write(paste("        <scale>", meuse.ll[i,varname]@data[[1]]/maxvar*2+0.3,
+     "</scale>", sep=""), filename, append = TRUE)
>   write("                        <Icon>", filename, append = TRUE)
# Icon type:
>   write("   <href>http://maps.google.com/mapfiles/kml/shapes/donut.png</href>",
+     filename, append = TRUE)
>   write("    </Icon>", filename, append = TRUE)
>   write("          </IconStyle>", filename, append = TRUE)
>   write(" </Style>", filename, append = TRUE)
> }
> write("<Folder>", filename, append = TRUE)
> write(paste("<name>Donut icon for", varname,"</name>"), filename, append = TRUE)
# Write placemark style in a loop:
> for (i in 1:length(meuse.ll@data[[1]])) {
>   write("  <Placemark>", filename, append = TRUE)
>   write(paste("  <name>", meuse.ll[i,varname]@data[[1]],"</name>", sep=""),
+  filename, append = TRUE)
>   write(paste("  <styleUrl>#pnt",i,"</styleUrl>", sep=""), filename, append=TRUE)
>   write("    <Point>", filename, append = TRUE)
>   write(paste("      <coordinates>",coordinates(meuse.ll)[[i,1]],",",
+ coordinates(meuse.ll)[[i,2]],",10</coordinates>", sep=""), filename, append=TRUE)
>   write("    </Point>", filename, append = TRUE)
>   write("    </Placemark>", filename, append = TRUE)
> }
> write("</Folder>", filename, append = TRUE)
> write("</Document>", filename, append = TRUE)
> write("</kml>", filename, append = TRUE)
> close(filename)
# To zip the file use the 7z program:
# system(paste("7za a -tzip ", varname, "_bubble.kmz ", varname, "_bubble.kml", sep=""))
# unlink(paste(varname, "_bubble.kml", sep=""))
```

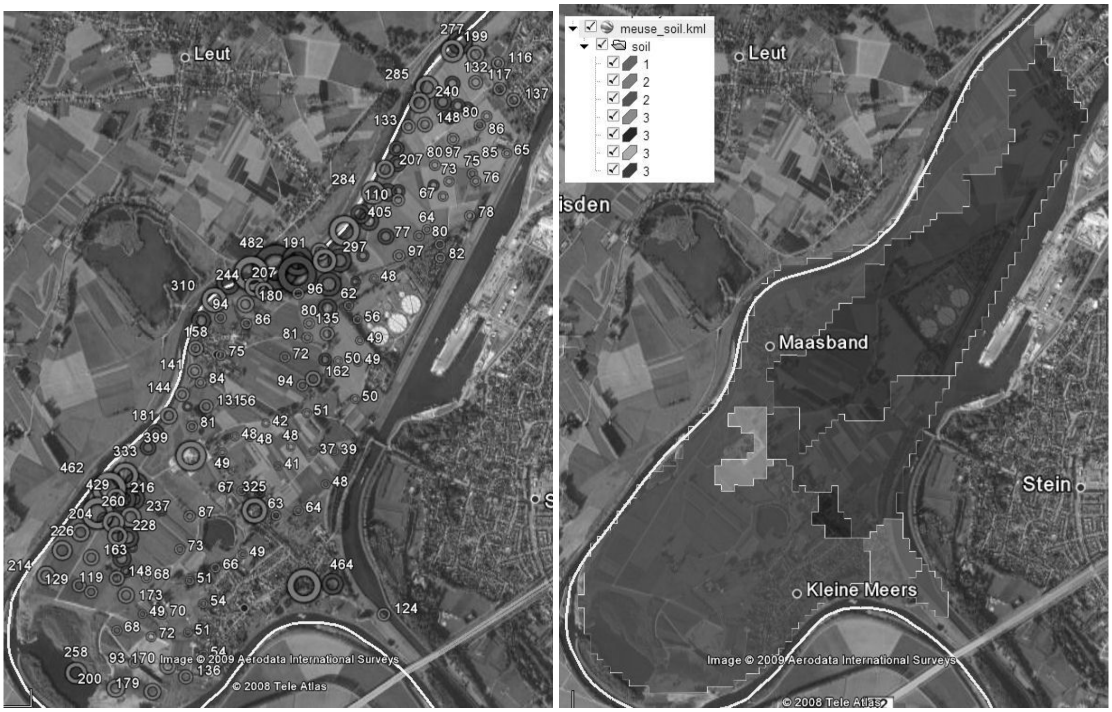


Fig. 3.10: Zinc values visualized using the bubble-type of plot in Google Earth (left). Polygon map (soil types) exported to KML and colored using random colors with transparency (right).

which will produce a plot shown in Fig. 3.10. Note that one can also output a multiline file by using e.g. `cat("ABCDEF", pi, "XYZ", file = "myfile.txt")`, rather than outputting each line separately (see also the `sep=` and `append=` arguments to `cat`).

Polygon maps can also be exported using the `writeOGR` command, as implemented in the package rgdal. In the case of the meuse data set, we first need to prepare a polygon map:

```
> data(meuse.grid)
> coordinates(meuse.grid) <- ~ x+y
> gridded(meuse.grid) <- TRUE
> proj4string(meuse.grid) <- CRS("+init=epsg:28992")
# raster to polygon conversion;
> write.asciigrid(meuse.grid["soil"], "meuse_soil.asc")
> rsaga.esri.to.sgrd(in.grids="meuse_soil.asc", out.sgrd="meuse_soil.sgrd",
+    in.path=getwd())
> rsaga.geoprocessor(lib="shapes_grid", module=6,
+    param=list(GRID="meuse_soil.sgrd", SHAPES="meuse_soil.shp", CLASS_ALL=1))
> soil <- readShapePoly("meuse_soil.shp", proj4string=CRS("+init=epsg:28992"),
+    force_ring=T)
> soil.ll <- spTransform(soil, CRS("+proj=longlat +datum=WGS84"))
> writeOGR(soil.ll["NAME"], "meuse_soil.kml", "soil", "KML")
```

The result can be seen in Fig. 3.10 (right). Also in this case we could have manually written the KML files to achieve the best effect.

3.3.2 Exporting raster maps (images) to KML

Rasters cannot be exported to KML as easily as vector maps. Google Earth does not allow import of GIS raster formats, but only input of images that can then be draped over a terrain (**ground overlay**). The images need to be exclusively in one of the following formats: JPG, BMP, GIF, TIFF, TGA and PNG. Typically, export of raster maps to KML follows these steps:

(1.) Determine the grid system of the map in the LatLonWGS84 system. You need to determine five parameters: southern edge (south), northern edge (north), western edge (west), eastern edge (east) and cellsize in arcdegrees (Fig. 3.11).

(2.) Reproject the original raster map using the new LatLonWGS84 grid.

(3.) Export the raster map using a graphical format (e.g. TIFF), and optionally the corresponding legend.

(4.) Prepare a KML file that includes a JPG of the map (Ground Overlay[55]), legend of the map (Screen Overlay) and description of how the map was produced. The JPG images you can locate on some server and then refer to an URL.

For data sets in geographical coordinates, a cell size correction factor can be estimated as a function of the latitude and spacing at the equator (Hengl and Reuter, 2008, p.34):

$$\Delta x_{\mathsf{metric}} = F \cdot \cos(\varphi) \cdot \Delta x^{0}_{\mathsf{degree}} \tag{3.3.1}$$

where $\Delta x_{\mathsf{metric}}$ is the East/West grid spacing estimated for a given latitude (φ), $\Delta x^{0}_{\mathsf{degree}}$ is the grid spacing in degrees at equator, and F is the empirical constant used to convert from degrees to meters (Fig. 3.11).

Once you have resampled the map, you can then export it as an image and copy to some server. Examples how to generate KML ground overlays in R are further discussed in sections 5.6.2 and 10.6.3. A KML file that can be used to visualize a result of geostatistical mapping has approximately the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1">
<Document>
    <name>Raster map example</name>
    <GroundOverlay>
        <name>Map name</name>
        <description>Description of how was map produced.</description>
        <Icon>
                <href>exported_map.jpg</href>
            </Icon>
            <LatLonBox>
                <north>51.591667</north>
                <south>51.504167</south>
                <east>10.151389</east>
                <west>10.010972</west>
            </LatLonBox>
        </GroundOverlay>
        <ScreenOverlay>
            <name>Legend</name>
            <Icon>
                <href>map_legend.jpg</href>
            </Icon>
            <overlayXY x="0" y="1" xunits="fraction" yunits="fraction"/>
            <screenXY x="0" y="1" xunits="fraction" yunits="fraction"/>
            <rotationXY x="0" y="0" xunits="fraction" yunits="fraction"/>
            <size x="0" y="0" xunits="fraction" yunits="fraction"/>
        </ScreenOverlay>
</Document>
</kml>
```

The output map and the associated legend can be both placed directly on a server. An example of ground overlay can be seen in Fig. 5.20 and Fig. 10.12. Once you open this map in Google Earth, you can edit it, modify the transparency, change the icons used and combine it with other vector layers. Ground Overlays can also be added directly in Google Earth by using commands *Add* ↦ *Image Overlay*, then enter the correct

[55] http://code.google.com/apis/kml/documentation/kml_tut.html

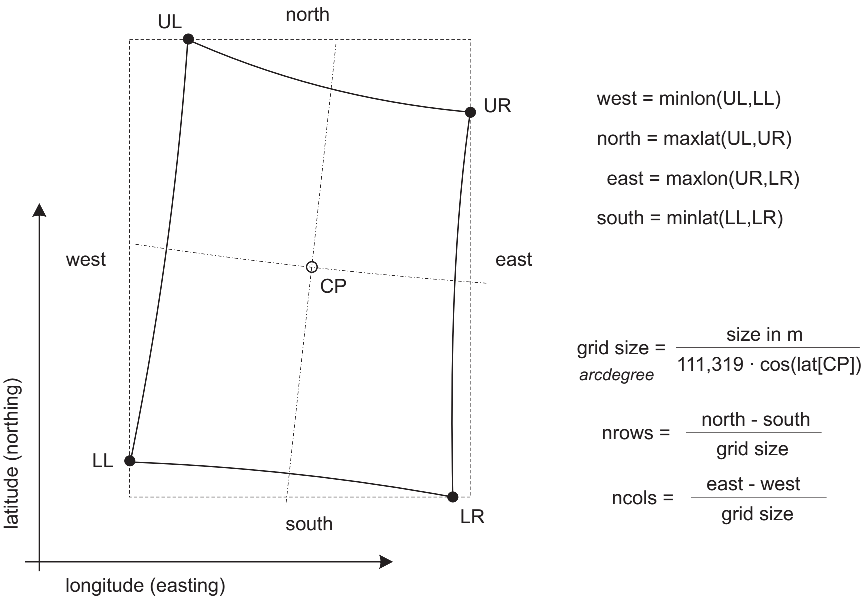
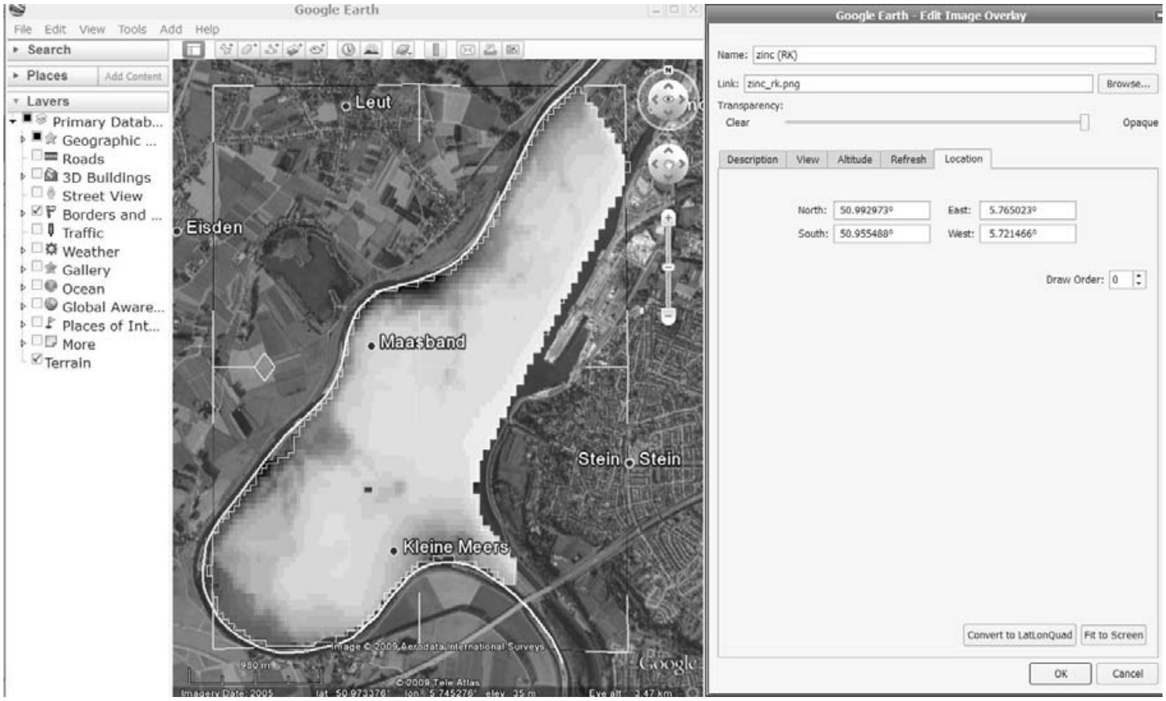


Fig. 3.11: Determination of the bounding coordinates and cell size in the `LatLonWGS84` geographic projection system using an existing Cartesian system. For large areas (continents), it is advisable to visually validate the estimated values.



Name: zinc (RK)

Link: zinc_rk.png

North: 50.992973° East: 5.765023°

South: 50.955488° West: 5.721466°

Draw Order: 0

Fig. 3.12: Preparation of the image ground overlays using the Google Earth menu.

bounding coordinates and location of the image file (Fig. 3.12). Because the image is located on some server, it can also be automatically refreshed and/or linked to a Web Mapping Service (WMS). For a more sophisticated use of Google interfaces see for example the interactive **KML sampler**[56], that will give you some good ideas about what is possible in Google Earth. Another interactive KML creator that plots various (geographical) CIA World Factbook, World Resources Institute EarthTrends and UN Data is the KML FactBook[57].

Another possibility to export the gridded maps to R (without resampling grids) is to use the vector structure of the grid, i.e. to export each grid node as a small squared polygon[58]. First, we can convert the grids to polygons using the maptools package and reproject them to geographic coordinates (Bivand et al., 2008):

```
# generate predictions e.g.:
> zinc.rk <- krige(log1p(zinc) ~ dist+ahn, data=meuse, newdata=meuse.grid,
+           model=vgm(psill=0.151, "Exp", range=374, nugget=0.055))
> meuse.grid$zinc.rk <- expm1(zinc.rk$var1.pred)
# convert grids to pixels (mask missing areas):
> meuse.pix <- as(meuse.grid["zinc.rk"], "SpatialPixelsDataFrame")
# convert grids to polygons:
> grd.poly <- as.SpatialPolygons.SpatialPixels(meuse.pix)
# The function is not suitable for high-resolution grids!!
> proj4string(grd.poly) <- CRS("+init=epsg:28992")
> grd.poly.ll <- spTransform(grd.poly, CRS("+proj=longlat +datum=WGS84"))
> grd.spoly.ll <- SpatialPolygonsDataFrame(grd.poly.ll,
+       data.frame(meuse.pix$zinc.rk), match.ID=FALSE)
```

Next, we need to estimate the Google codes for colors for each polygon. The easiest way to achieve this is to generate an RGB image in R, then reformat the values following the KML tutorial:

```
> tiff(file="zinc_rk.tif", width=meuse.grid@grid@cells.dim[1],
+        height=meuse.grid@grid@cells.dim[2], bg="transparent")
> par(mar=c(0,0,0,0), xaxs="i", yaxs="i", xaxt="n", yaxt="n")
> image(as.image.SpatialGridDataFrame(meuse.grid["zinc.rk"]), col=bpy.colors())
> dev.off()
# read RGB layer back into R:
> myTile <- readGDAL("zinc_rk.tif", silent=TRUE)
> i.colors <- myTile@data[meuse.pix@grid.index,]
> i.colors[1:3,]
```

```
     band1 band2 band3
 69     72     0   255
 146    94     0   255
 147    51     0   255
```

```
> i.colors$B <- round(i.colors$band3/255*100, 0)
> i.colors$G <- round(i.colors$band2/255*100, 0)
> i.colors$R <- round(i.colors$band1/255*100, 0)
# generate Google colors:
> i.colors$FBGR <- paste("9d", ifelse(i.colors$B<10, paste("0", i.colors$B, sep=""),
+     ifelse(i.colors$B==100, "ff", i.colors$B)),
+     ifelse(i.colors$G<10, paste("0", i.colors$G, sep=""),
+     ifelse(i.colors$G==100, "ff", i.colors$G)),
+     ifelse(i.colors$R<10, paste("0", i.colors$R, sep=""),
+     ifelse(i.colors$R==100, "ff", i.colors$R)), sep="")
> i.colors$FBGR[1:3]
```

```
 [1] "9dff0028" "9dff0037" "9dff0020"
```

and we can write Polygons to KML with color attached in R:

[56] http://kml-samples.googlecode.com/svn/trunk/interactive/index.html

[57] http://www.kmlfactbook.org/

[58] This is really recommended only for fairly small grid, e.g. with $\ll 10^6$ grid nodes.

```
> varname <- "zinc_rk"  # variable name
> filename <- file(paste(varname, "_poly.kml", sep=""), "w")
> write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>", filename)
> write("<kml xmlns=\"http://earth.google.com/kml/2.2\">", filename, append = TRUE)
> write("<Document>", filename, append = TRUE)
> write(paste("<name>", varname, "</name>", sep=" "), filename, append = TRUE)
> write("<open>1</open>", filename, append = TRUE)
> for (i in 1:length(grd.spoly.ll@data[[1]])) {
>   write(paste(' <Style id="','poly', i,'">',sep=""), filename, append = TRUE)
>   write("   <LineStyle>", filename, append = TRUE)
>   write("   <width>0.5</width>", filename, append = TRUE)
>   write("   </LineStyle>", filename, append = TRUE)
>   write("                <PolyStyle>", filename, append = TRUE)
>   write(paste('      <color>', i.colors$FBGR[i], '</color>', sep=""),
+         filename, append = TRUE)
>   write("          </PolyStyle>", filename, append = TRUE)
>   write(" </Style>", filename, append = TRUE)
> }
> write("<Folder>", filename, append = TRUE)
> write(paste("<name>Poly ID", varname,"</name>"), filename, append = TRUE)
> for (i in 1:length(grd.spoly.ll@data[[1]])) {
>   write("  <Placemark>", filename, append = TRUE)
>   write(paste("  <name>", grd.spoly.ll@polygons[[i]]@ID, "</name>", sep=""),
+         filename, append = TRUE)
>   write("  <visibility>1</visibility>", filename, append = TRUE)
>   write(paste("  <styleUrl>#poly", i, "</styleUrl>", sep=""), filename, append = TRUE)
>   write("    <Polygon>", filename, append = TRUE)
>   write("    <tessellate>1</tessellate>", filename, append = TRUE)
>   write("      <altitudeMode>extruded</altitudeMode>", filename, append = TRUE)
>   write("      <outerBoundaryIs>", filename, append = TRUE)
>   write("        <LinearRing>", filename, append = TRUE)
>   write("      <coordinates>", filename, append = TRUE)
>   write(paste("          ", grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[1,1], ",",
+        grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[1,2], ",1", sep=""),
+        filename, append = TRUE)
>   write(paste("          ", grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[2,1], ",",
+        grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[2,2], ",1", sep=""),
+        filename, append = TRUE)
>   write(paste("          ", grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[3,1], ",",
+        grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[3,2], ",1", sep=""), ",",
+        filename, append = TRUE)
>   write(paste("          ", grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[4,1], ",",
+        grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[4,2], ",1", sep=""),
+        filename, append = TRUE)
>   write(paste("          ", grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[5,1], ",",
+        grd.spoly.ll@polygons[[i]]@Polygons[[1]]@coords[5,2], ",1", sep=""),
+        filename, append = TRUE)
>   write("      </coordinates>", filename, append = TRUE)
>   write("        </LinearRing>", filename, append = TRUE)
>   write("      </outerBoundaryIs>", filename, append = TRUE)
>   write("    </Polygon>", filename, append = TRUE)
>   write("    </Placemark>", filename, append = TRUE)
> }
> write("</Folder>", filename, append = TRUE)
> write("</Document>", filename, append = TRUE)
> write("</kml>", filename, append = TRUE)
> close(filename)
```

In similar fashion, time-series of maps (range of maps covering the same geographic domain, but referring to a different time period) can be exported and explored in Google Earth using **time-slider**. Note also that a list of remotely sensed image bands or maps produced by geostatistical simulations can be exported as a time-series, and then visualized as animation. The maptools package is not able to generate GE KMLs using space-time data[59], but we can instead write directly a KML file from R. An example is further shown in section 11.5.3.

Users of MatLab can explore possibilities for exporting the results of geostatistical analysis to Google Earth by using the Google Earth Toolbox[60]. This toolbox serves not only to export maps (vectors, ground overlays), but also as a friendly tool to export the associated legends, generate 3D surfaces, contours from isometric maps, wind barbs and 3D vector objects. For example, a gridded map produced using some spatial prediction technique can be converted to a KML format using e.g.:

```
MATLAB: examplemap = ge_groundoverlay(N,E,S,W,... 'imageURL','mapimage.png');
MATLAB: ge_output('examplemap.kml',kmlStr);
```

where N,E,S,W are the bounding coordinates that can be determined automatically or set by the user.

3.3.3 Reading KML files to R

In principle, everything we export from R to KML we can also read back into R. GDAL has an OGR KML driver that should work for vector data also on Windows OS with the standard rgdal binary. The GDAL website[61] indicates that KML reading is only available if GDAL/OGR is built with the Expat XML Parser, otherwise only KML writing will be supported. Supported OGR geometry types are: Point, Linestring, Polygon, MultiPoint, MultiLineString, MultiPolygon and MultiGeometry. Unfortunately, reading of more complex KML files is still a cumbersome. Reading of KML files can not easily be automated because the code often requires editing, for the simple reason that KML does not (yet) have standard spatial data formats.

The most recent version of the package maptools contains methods (e.g. getKMLcoordinates) to read KML files to R and coerce them to the sp formats. KML could carry a lot of non-spatial information, or spatial information that is not supported by most GIS (e.g. viewing angle, transparency, description tags, style definition etc.). In addition, you could have more maps within the same KML file, which makes it difficult to automate import of KML files to a GIS.

To read the point map meuse_lead.kml we exported previously using the writeOGR method, we can always use the XML[62] package:

```
> library(XML)
> meuse_lead.kml <- xmlTreeParse("meuse_lead.kml")
> lengthp <- length(meuse_lead.kml$doc[[1]][[1]][[1]])-1
> lead_sp <- data.frame(Longitude=rep(0,lengthp), Latitude=rep(0,lengthp),
+    Var=rep(0,lengthp))
> for(j in 1:lengthp) {
>   LatLon <- unclass(meuse_lead.kml$doc[[1]][[1]][[1]][j+1][[1]][2][[1]][[1]][[1]])$value
>   Var <- unclass(meuse_lead.kml$doc[[1]][[1]][[1]][j+1][[1]][1][[1]][[1]])$value
>   lead_sp$Longitude[[j]] <- as.numeric(matrix(unlist(strsplit(LatLon,
+      split=",")), ncol=2)[1])
>   lead_sp$Latitude[[j]] <- as.numeric(matrix(unlist(strsplit(LatLon,
+      split=",")), ncol=2)[2])
>   lead_sp$Var[[j]] <- as.numeric(matrix(unlist(strsplit(strsplit(Var,
+      split="<i>")[[1]][2], split="</i>")), ncol=2)[1])
> }
> coordinates(lead_sp) <- ~ Longitude+Latitude
> proj4string(lead_sp) <- CRS("+proj=longlat +ellps=WGS84")
> bubble(lead_sp, "Var")
```

Note that it will take time until you actually locate where in the KML file the coordinates of points and attribute values are located (note long lines of sub-lists). After that it is relatively easy to automate creation of a SpatialPointsDataFrame. This code could be shorten by using the xmlGetAttr(), xmlChildren() and xmlValue() methods. You might also consider using the KML2SHP[63] converter (ArcView script) to read KML files (points, lines, polygons) and generate shapefiles directly from KML files.

[59] The new stpp package (see R-forge) is expected to bridge this gap.

[60] http://www.mathworks.com/matlabcentral/fileexchange/12954

[61] http://www.gdal.org/ogr/drv_kml.html

[62] http://cran.r-project.org/web/packages/XML/

[63] http://arcscripts.esri.com/details.asp?dbid=14988

3.4 Summary points

3.4.1 Strengths and limitations of geostatistical software

Both open source and proprietary software packages have strong and weak points. A comparison of different aspects of geostatistical packages listed at the AI-Geostats website[64] and several well-known GIS packages can be seen in Table 3.1. Although universal kriging (using coordinates) is available in most geostatistical packages, kriging with external drift with multiple auxiliary maps can be run in only a limited number of packages. From all software listed in Table 3.1, only Isatis, SAGA, and gstat/geoR (as stand-alone application or integrated into R) offer the possibility to interpolate a variable using (multiple) auxiliary maps (Hengl et al., 2007a). Note that the comparison of packages is not trivial because many proprietary packages rely on plugins/toolboxes that are either distributed separately or are valid for certain software versions only. For example, ArcGIS has excellent capabilities for writing and reading KML files, but the user needs to obtain the Arc2Earth package, which is sold separately.

Hengl et al. (2007a) tested regression-kriging in variety of packages to discover that RK in Isatis is limited to use of a single (three in script mode) auxiliary maps (Bleines et al., 2004). In gstat, both RK predictions and simulations (predictors as base maps) at both point and block support can be run by defining short scripts, which can help automatize interpolation of large amounts of data. However, gstat implements the algorithm with extended matrix (KED), which means that both the values of predictors and of target variable are used to estimate the values at each new location, which for large data sets can be time-consuming or can lead to computational problems (Leopold et al., 2005). geoR stands out as a package with the most sophisticated approaches to model estimation, but it is not really operational for use with large data sets. It was not designed to work with common spatial classes and data formats and some manual work is needed to get the maps out (see further section 5.5.3).

Setting RK in gstat or SAGA to a smaller window search can lead to termination of the program due to singular matrix problems[65]. In fact, local RK with a global variogram model is not entirely valid because the regression model will differ locally, hence the algorithm should also estimate the variogram model for residuals for each local neighborhood (as mentioned previously in §2.2). The singular matrix problem will happen especially when indicator variables are used as predictors or if the two predictor maps are highly correlated. Another issue is the computational effort. Interpolation of $\gg 10^3$ points over 1M of pixels can last up to several hours on a standard PC. To run simulations in R+gstat with the same settings will take even more time. This proves that, although the KED procedure is mathematically elegant, it might be more effective for real-life applications to fit the trend and residuals separately (the regression-kriging approach). A limitation of gstat.exe is that it is a stand-alone application and the algorithms cannot be adjusted easily. Unlike the gstat package in R that can be extended and then uploaded by anybody. A limitation of R, however, is that it can reach memory use problems if larger rasters or larger quantities of rasters are loaded into R. Visualization and visual exploration of large maps in R is also not recommended.

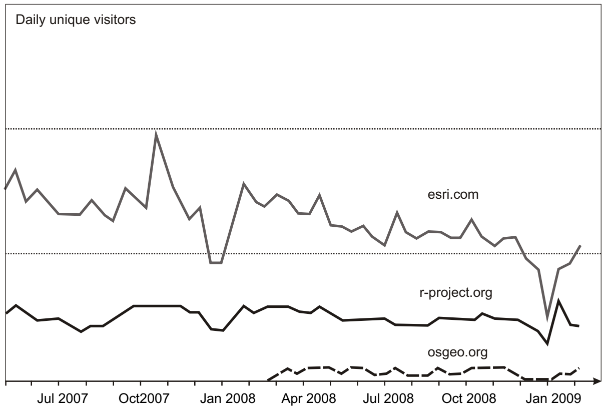


Fig. 3.13: Trends in the web-traffic for esri.com, r-project.org and osgeo.org (following the trends.google.com statistics).

So in summary, if you wish to fit your data in a statistically optimal way and with no limitations on the number of predictors and statistical operations — R and packages gstat and geoR are the most professional choice. If you do not feel confident about using software environment without an interface, then you could try running global Universal kriging in SAGA. However, SAGA does not provide professional variogram modeling capabilities, so a combination R+GIS+GE is probably the best idea. In fact, the computation is a bit faster in SAGA than in R and there are no memory limit problems (see further section 5.5.2). However, in SAGA you

[64] http://ai-geostats.org

[65] You can prevent gstat from breaking predictions by setting the krige option set=list(cn_max=1e10, debug=0).

Table 3.1: Comparison of spatio-temporal data analysis capabilities of some popular statistical and GIS packages (versions in year 2009): ★ — full capability, ⋆ — possible but with many limitations, − — not possible in this package. Commercial price category: I — > 1000 EUR; II — 500-1000 EUR; III — < 500 EUR; IV — open source or freeware. Main application: A — statistical analysis and data mining; B — interpolation of point data; C — processing / preparation of input maps; E — visualization and visual exploration. After Hengl et al. (2007a).

Aspect	S-PLUS	R+gstat	R+geoR	MatLab	SURFER	ISATIS	GEOEas	GSLIB	GRASS	PC Raster	ILWIS	IDRISI	ArcGIS	SAGA
Commercial price category	II	IV	IV	I	III	I	IV	IV	IV	III	IV	II	I	IV
Main application	A, B	A, B	A, B	A, E	B, E	B	B	B	B, C	C	B, C	B, C	B, E	B, C
User-friendly environment to non-expert	★	−	−	★	★	★	−	−	⋆	−	★	★	★	★
Quality of support and description of algorithms	⋆	★	★	★	★	★	★	★	⋆	⋆	★	★	⋆	⋆
Standard GIS capabilities	−	⋆	−	⋆	⋆	−	−	−	★	⋆	★	★	★	★
GDAL support	−	★	⋆	⋆	−	⋆	−	−	★	⋆	⋆	★	★	⋆
Standard descriptive statistical analysis	★	★	★	★	−	★	⋆	⋆	⋆	−	★	★	★	⋆
Image processing tools (orthorectification, filtering, land surface analysis)	−	−	−	⋆	−	−	−	−	★	−	★	★	⋆	★
Comprehensive regression analysis (regression trees, GLM)	★	★	★	★	−	⋆	−	−	−	−	−	⋆	−	−
Interactive (automated) variogram modeling	−	⋆	★	−	−	★	−	⋆	−	−	−	★	★	−
Regression-kriging with auxiliary maps	−	★	⋆	−	−	⋆	−	−	★	⋆	⋆	★	−	★
Dynamic modeling (simulations, spatial iterations, propagation, animations)	−	⋆	⋆	⋆	−	−	−	−	⋆	★	⋆	⋆	⋆	⋆
Processing of large data sets	★	⋆	−	⋆	⋆	★	−	−	★	⋆	⋆	★	★	★
Export of maps to Google Earth (KML)	−	★	⋆	★	−	−	−	−	⋆	−	−	⋆	★	−

will not be able to objectively estimate the variogram of residuals or GLS model for the deterministic part of variation.

The R+SAGA/GRASS+GE combo of applications allows full GIS + statistics integration and can support practically 80% of processing/visualization capabilities available in proprietary packages such as ArcInfo/Map or Idrisi. The advantage of combining R with open source GIS is that you will be able to process and visualize even large data sets, because R is not really suited for processing large data volumes, and it was never meant to be used for visual exploration or editing of spatial data. On the other hand, packages such as ILWIS and SAGA allow you to input, edit and visually explore geographical data, before and after the actual statistical analysis. Note also that ILWIS, SAGA and GRASS extend the image processing functionality (especially image filtering, resampling, geomorphometric analysis and similar) of R that is, at the moment, limited to only few experimental packages (e.g. biOps, rimage; raster[66]). An alternative for GIS+R integration is QGIS[67], which has among its main characteristics a python console, and a very elaborate way of adding Python plugins, which is already in use used for an efficient R plugin (manageR).

In principle, we will only use open source software to run the exercises in this book, and there are several good reasons. Since the 1980's, the GIS research community has been primarily influenced by the (proprietary) software licence practices that limited sharing of ideas and user-controlled development of new functionality (Steiniger and Bocher, 2009). With the initiation of the **Open Source Geospatial Foundation** (OSGeo), a new era began: the development and use of open source GIS has experienced a boom over the last few years; the enthusiasm to share code, experiences, and to collaborate on projects is growing (Bivand, 2006).

Steiniger and Bocher (2009) recognize four indicators of this trend: (1) increasing number of projects run using the open source GIS, (2) increasing financial support by government agencies, (3) increasing download rates, and (4) increasing number of use-cases. By comparing the web-traffic for proprietary and open source GIS (Fig. 3.13) one can notice that OSGeo has indeed an increasing role in the world market of GIS. Young and senior researchers are slowly considering switching from using proprietary software such as ESRI's ArcGIS and/or Mathworks' MatLab to R+SAGA, but experience (Windows *vs* Linux) teaches us that it would be over-optimistic to expect that this shift will go fast and without resistance.

3.4.2 Getting addicted to R

From the previously-discussed software tools, one software needs to be especially emphasized, and that is R (R Development Core Team, 2009). Many R users believe that there is not much in statistics that R cannot do[68] (Zuur et al., 2009). Certainly, the number of packages is increasing everyday, and so is the community. There are at least five good (objective) reasons why you should get deeper into R (Rossiter, 2009):

It is of high quality — It is a non-proprietary product of international collaboration between top statisticians.

It helps you think critically — It stimulates critical thinking about problem-solving rather than a *push the button* mentality.

It is an open source software — Source code is published, so you can see the exact algorithms being used; expert statisticians can make sure the code is correct.

It allows automation — Repetitive procedures can easily be automated by user-written scripts or functions.

It helps you document your work — By scripting in R, anybody is able to reproduce your work (processing metadata). You can record steps taken using history mechanism even without scripting, e.g. by using the `savehistory()` command.

It can handle and generate maps — R now also provides rich facilities for interpolation and statistical analysis of spatial data, including export to GIS packages and Google Earth.

The main problem with R is that each step must be run via a command line, which means that the analyst must really be an R expert. Although one can criticize R for a lack of an user-friendly interface, in fact, most power users in statistics never use a GUI. GUI's are fine for baby-steps and getting started, but not for

[66] `http://r-forge.r-project.org/projects/raster/` — raster is possibly the most active R spatial project at the moment.

[67] `http://qgis.org/`

[68] This is probably somewhat biased statement. For example, R is not (yet) operational for processing of large images (filter analysis, map iterations etc.), and many other standard geographical analysis steps are lacking.

production work. The whole point is that one can develop a script or program that, once it is right, can be re-run and will produce exactly the same results each time (excluding simulations of course).

Here are some useful tips on how to get addicted to R. First, you should note that you can edit the R scripts in user-friendly script editors such as Tinn-R[69] or use the package R commander (Rcmdr[70]), which has an user-friendly graphical interface. Second, you should take small steps before you get into really sophisticated script development. Start with some simple examples and then try to do the same exercises with your own data. The best way to learn R is to look at existing scripts. For example, a French colleague, Romain François, maintains a gallery of R graphics[71] that is dedicated to the noble goal of getting you addicted to R. A similar-purpose website is the **R Graphical Manual**[72]. Robert I. Kabacoff maintains a website called **Quick-R**[73] that gives an overview of R philosophy and functionality. John Verzani maintains a website with **simple examples in R** that will get you going[74]. If you love cookbooks, R has those too[75]. In fact, you might make the following package the first you try. Type the following command into R to (1) download a tutorial package; (2) load the package in R; and (3) tell R to open a webpage with more information about the tutorial (Verzani, 2004):

```
> install.packages("UsingR")
> library(UsingR)
> UsingR("exercises")
```

Third, if your R script does not work, do not despair, try to obtain some specialized literature. Reading specialized literature from the experts is possibly the best way to learn script writing in R. Start with Zuur et al. (2009), then continue with classics such as Chambers and Hastie (1992), Venables and Ripley (2002) and/or Murrell (2006). Other excellent and freely available introductions to R are Kuhnert and Venables (2005), Burns (2009) and Rossiter (2009). If you are interested in running spatial analysis or geostatistics in R, then books by Bivand et al. (2008), Baddeley (2008) Reimann et al. (2008), and/or Diggle and Ribeiro Jr (2007) are a must.

Getting help

Because the R community is large, chances are good that the problems you encounter have already been solved or at least discussed, i.e. it is already *there*. The issue is how to find this information. There are several sources where you should look:

(1.) the help files locally installed on your machine;

(2.) mailing/discussion lists;

(3.) various website and tutorials, and

(4.) books distributed by commercial publishers;

You should really start searching in this order — from your machine to a bookstore — although it is always a good idea to cross-check all possible sources and compare alternatives. Please also bear in mind that (a) not everything that you grab from www is correct and up-to-date; and (b) it is also possible that you have an original problem that nobody else has experienced.

Imagine that you would like to find out which packages on your machine can run interpolation by kriging. You can quickly find out this by running the method `help.search`, which will give something like this:

```
> help.search("kriging")

  Help files with alias or concept or title matching 'kriging' using fuzzy matching:

  image.kriging(geoR)            Image or Perspective Plot with Kriging Results
  krige.bayes(geoR)              Bayesian Analysis for Gaussian Geostatistical Models
```

[69] http://www.sciviews.org/Tinn-R/
[70] http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/
[71] http://addictedtor.free.fr
[72] http://bm2.genes.nig.ac.jp/RGM2/
[73] http://www.statmethods.net
[74] http://www.math.csi.cuny.edu/Statistics/R/simpleR/
[75] http://www.r-cookbook.com

```
krige.conv(geoR)                Spatial Prediction -- Conventional Kriging
krweights(geoR)                 Computes kriging weights
ksline(geoR)                    Spatial Prediction -- Conventional Kriging
legend.krige(geoR)              Add a legend to a image with kriging results
wo(geoR)                        Kriging example data from Webster and Oliver
xvalid(geoR)                    Cross-validation by kriging
krige(gstat)                    Simple, Ordinary or Universal, global or local,
                                Point or Block Kriging, or simulation.
krige.cv(gstat)                 (co)kriging cross validation, n-fold or leave-one-out
ossfim(gstat)                   Kriging standard errors as function of grid spacing
                                and block size
krige(sgeostat)                 Kriging
prmat(spatial)                  Evaluate Kriging Surface over a Grid
semat(spatial)                  Evaluate Kriging Standard Error of Prediction over a Grid

Type 'help(FOO, package = PKG)' to inspect entry 'FOO(PKG) TITLE'.
```

This shows that kriging (and its variants) is implemented in (at least) four packages. We can now display the help for the method "krige" that is available in the package gstat:

```
> help(krige, package=gstat)
```

The archives of the mailing lists are available via the servers in Zürich. They are fairly extensive and the best way to find something useful is to search them. The fastest way to search all R mailing lists is to use the RSiteSearch method. For example, imagine that you are trying to run kriging and then the console gives you the following error message e.g.:

```
"Error : dimensions do not match: locations XXXX and data YYYY"
```

Based on the error message we can list at least 3–5 keywords that will help us search the mailing list, e.g.:

```
> RSiteSearch("krige {dimensions do not match}")
```

This will give over 15 messages[76] with a thread matching exactly your error message. This means that other people also had this problem, so now you only need to locate the right solution. You should sort the messages by date and then start from the most recent message. The answer to your problem will be in one of the replies submitted by the mailing list subscribers. You can quickly check if this is a solution that you need by making a small script and then testing it.

Of course, you can at any time Google the key words of interest. However, you might instead consider using the Rseek.org[77] search engine maintained by Sasha Goodman. The advantage of using Rseek over e.g. general Google is that it focuses only on R publications, mailing lists, vignettes, tutorials etc. The result of the search is sorted in categories, which makes it easier to locate the right source.

If you are eventually not able to find a solution yourself, you can try sending the description of your problem to a mailing list, i.e. asking the R *gurus*. Note that there are MANY R mailing lists[78], so you first have to be sure to find the right one. Sending a right message to a wrong mailing list will still leave you without an answer. Also have in mind that everything you send to a mailing list is public/archived, so better cross-check your message before you send it. When asking for a help from a mailing list, use the existing pre-installed data sets to describe your problem[79].

Do's:

- If you have not done so already, **read the R posting guide**[80]!
- **Use the existing pre-installed data sets** (come together with a certain package) **to describe your problem**. You can list all available data sets on your machine by typing data(). This way you do not have to attach your original data or waste time on trying to explain your case study.

[76] Unfortunately, RSiteSearch() no longer searches R-sig-geo — the full archive of messages is now on Nabble.

[77] http://rseek.org

[78] There are several specific **Special Interest Group** mailing lists; see http://www.r-project.org/mail.html.

[79] Then you only need to communicate the problem and not the specifics of a data set; there is also no need to share your data.

[80] http://www.r-project.org/posting-guide.html

- If your problem is completely specific to your data set, then **upload it an put it on some web-directory** so that somebody can access it and see what really goes on.

- **Link your problem to some previously described problems**; put it in some actual context (to really understand what I mean here, you should consider attending the Use R conferences).

- **Acknowledge the work (time spent) other people do to help you**.

- You can submit not only the problems you discover but also the **information that you think is interesting for the community**.

Don'ts:

- **Do not send poorly formulated questions**. Make sure you give technical description of your data, purpose of your analysis, even the details about your operating system, RAM etc. Try to put yourself in a position of a person that is interested to help — try to provide all needed information as if the person who is ready to help you would feel like sitting at your computer.

- **Do not send too much**. One message, one question (or better to say "*one message, one problem*"). Nobody reading R mailing lists has time to read long articles with multiple discussion points. Your problem should fit half the page; if somebody gets more interested, you can continue the discussion also off the list.

- R **comes with ABSOLUTELY NO WARRANTY**. If you loose data or get strange results, you are welcome to improve the code yourself (or consider obtaining some commercial software). **Complaining to a mailing list about what frustrates you about R makes no sense, because nobody is obliged to take any responsibility**.

- R is a community project (it is based on the solidarity between the users). **Think what you can do for the community and not what the community can do for you**.

Probably the worst thing that can happen to your question is that you do not get any reply (and this does not necessarily mean that nobody wants to help you or that nobody know the solution)! There are several possible reasons why this happened:

- **You have asked too much**! Some people post questions that could take weeks to investigate (maybe this justifies a project proposal?). Instead, you should always limit yourself to 1-2 concrete issues. Broader discussions about various more general topics and statistical theory are sometimes also welcome, but they should be connected with specific packages.

- **You did not introduce your question/topic properly**. If your question is very specific to your field and the subscribers cannot really understand what you are doing, you need to think of ways to introduce your field and describe the specific context. The only way to learn the language used by the R mailing lists is to browse the existing mails (archives).

- **You are requesting that somebody does a work for you that you could do yourself**! R and its packages are all open source, which allows YOU to double check the underlying algorithms and extend them where necessary. If you want other people to do programming for you, then you are at the wrong place (some commercial software companies do accept wish-lists and similar types of requests, but that's what they are paid for anyway).

- **Your question has been answered already few times and it is quite annoying that you did not do your homework to check this**.

Remember: everything you send to mailing lists reach large audiences (for example, R-sig-geo has +1000 subscribers), and it is archived on-line, so you should be more careful about what you post. If you develop a bad reputation of being ignorant and/or too sloppy, then people might start ignoring your questions even if they eventually start getting the right shape.

Tips for successful scripting in R

R is a command line based environment, but users do really write things directly to a command line. It is more common to first write using text editors (e.g. Tinn-R, JGR) and then "*send lines*" of code to R command line. When generating an R script, there are few useful tips that you might consider following (especially if you plan to share this script with a wider community):

- Document your code to explain what you are doing. Comments in R can be inserted after the "#" sign; **You can never put too many comments in your code!**

- **Add some meta-information about the script at the beginning of your script** — its authors, last update, purpose, inputs and outputs, reference where somebody can find more info (R scripts usually come as supplementary materials for project reports or articles) and difficulties one might experience. In many situations it is also advisable to mention the package version used to generate outputs. Because R is dynamic and many re-designs happen, some old scripts might become incompatible with the new packages.

- Once you tested your script and saw that it works, **tidy-up the code** — remove unnecessary lines, improve the code where needed, and test it using extreme cases (Burns (2009) provides some useful tips on how to improve scripting). In R, many equivalent operations can be run via different paths. In fact, the same techniques are commonly implemented in multiple packages. On the other hand, not all methods are equally efficient (speed, robustness), i.e. equally elegant, so that it is often worth investigating what might be the most elegant way to run some analysis. A good practice is to always write a number of smaller functions and then a function that does everything using the smaller functions. Note also that the variable names and list of input maps can be save and dealt with as with lists (see e.g. page166).

- **Place the data sets on-line** (this way you only need to distribute the script) and then call the data by using the `download.file` method in R;

All these things will make a life easier for your colleagues, but also to yourself if you decide to come back to your own script in few years (or few months). Another thing you might consider is to directly write the code and comments in Tinn-R using the Sweave[81] package. Note that you can still run this script from Tinn-R, you only need to specify where the R code begins (<<>>=) and ends (@). This way, you do not only distribute the code, but also all explanation, formulae etc. Building metadata for both the data sets and scripts you produce is becoming increasingly important.

Memory limit problems

Another important aspect to consider is R's ability to handle large data sets. Currently, many pharmaceutical organizations and financial institutions use R as their primary analytical engine. They crunch huge amounts of data, so it works on very large data sets. However loading, displaying and processing large geographic data sets like big raster maps (≫1M pixels) can be problematic with R. Windows 32–bit OS can allocate a maximum of 2 GB of memory to an application[82]. Even if your computer has >2 GB of RAM, you will receive an error message like:

```
Error: cannot allocate vector of size 12.6 Mb
```

which might suggest to you that R has problems with surprisingly small data sets. This message actually means R has already allocated all of the 2 GB of memory available to it. Check the system's memory allocation by typing:

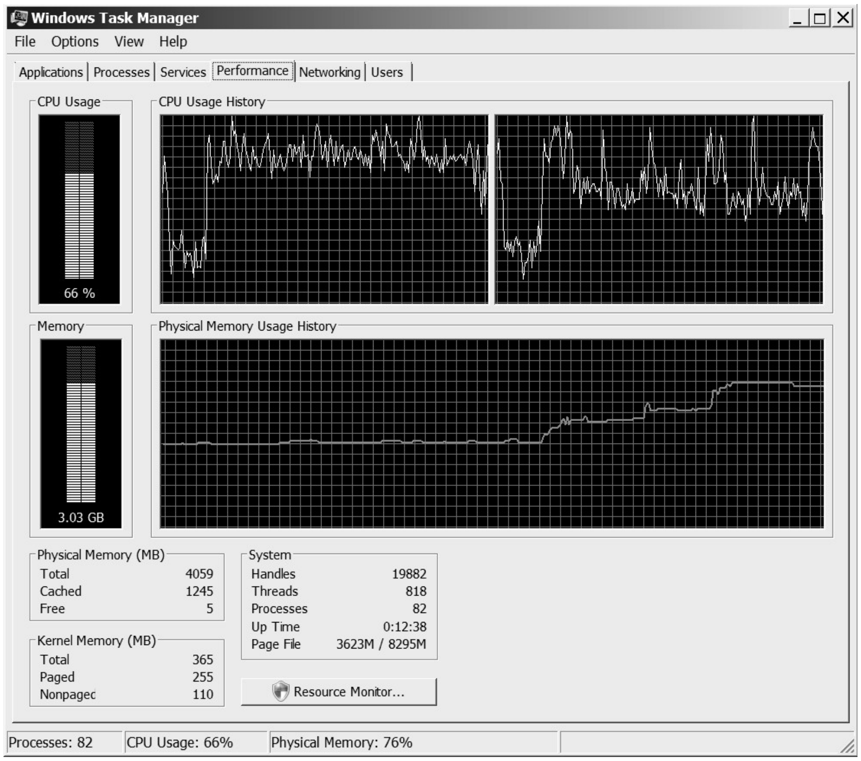
```
> gc()
```

and you will find R is already occupying a lot of RAM. The key issue is that R stores data in a temporary buffer, which can result in problems with memory availability. However, the benefit of this strategy is it allows maximum flexibility in data structures.. For Burns (2009) there are only two solutions to the problem: (1) Improve your code; (2) Get a bigger computer!

To increase your computing capacities, you can also consider doing one of the following:

[81] `http://www.stat.uni-muenchen.de/~leisch/Sweave/`

[82] Read more at: `http://www.microsoft.com/whdc/system/platform/server/PAE/PAEmem.mspx`



CPU Usage: 66 %

Memory: 3.03 GB

Physical Memory (MB)

Total	4059
Cached	1245
Free	5

Kernel Memory (MB)

Total	365
Paged	255
Nonpagec	110

System

Handles	19882
Threads	818
Processes	82
Up Time	0:12:38
Page File	3623M / 8295M

Processes: 82 CPU Usage: 66% Physical Memory: 76%

Fig. 3.14: Windows task manager showing the CPU and memory usage. Once the computing in R comes close to 2 GB of physical memory, Windows will not allow R to use any more memory. The solution to this problem is to use a PC with an 64–bit OS.

- Reduce the grid resolution of your maps. If you reduce the grid cell size by half, the memory usage will be four times smaller.

- Consider splitting your data set into tiles. Load data tile by tile, write the results to physical memory or external database, remove temporary files, repeat the analysis until all tiles are finished. This is the so called "*database*" solution to memory handling (Burns, 2009).

- Obtain a new machine. Install a 64–bit OS with >10GB of RAM. A 64–bit OS will allow you to use more application memory.

- Consider obtaining a personal supercomputer[83] with a completely customizable OS. Supercomputer is about 200 times faster than your PC, mainly because it facilitates multicore operations. The price of a standard personal supercomputer is about 5–10 times that of a standard PC.

During the processing, you might try releasing some free memory by continuously using the `gc()` command. This will remove some temporary files and hence increase some free memory. If you are Windows OS user, you should closely monitor your Windows Task manager (Fig. 3.14), and then, when needed, use garbage collector (`gc()`) and/or remove (`rm()`) commands to increase free space. Another alternative approach is to combine R with other (preferably open-source) GIS packages, i.e. to run all excessive processing externally from R. It is also possible that you could run extensive calculations even with your limited PC. This is because processing is increasingly distributed. For example, colleagues from the Centre for e-Science in Lancaster have been recently developing an R package called MultiR[84] that should be able to significantly speed up R calculations by employing grid computing facilities (Grose et al., 2006).

[83] See e.g. `http://www.nvidia.com/object/personal_supercomputing.html`

[84] `http://cran.r-project.org/web/views/HighPerformanceComputing.html`

3.4.3 Further software developments

There are still many geostatistical operations that we are aware of, but have not been implemented and are not available to broader public (§2.10.3). What programmers might consider for future is the refinement of (local) regression-kriging in a moving window. This will allow users to visualize variation in regression (maps of R-square and regression coefficients) and variogram models (maps of variogram parameters). Note that the regression-kriging with moving window would need to be fully automated, which might not be an easy task considering the computational complexity. Also, unlike OK with a moving window (Walter et al., 2001), regression-kriging has much higher requirements considering the minimum number of observations (at least 10 per predictor, at least 50 to model variogram). In general, our impression is that many of the procedures (regression and variogram modeling) in regression-kriging can be automated and amount of data modeling definitions expanded (local or global modeling, transformations, selection of predictors, type of GLMs etc.), as long as the point data set is large and of high quality. Ideally, users should be able to easily test various combinations of input parameters and then (in real-time) select the one that produces the most satisfactory predictions.

Open-source packages open the door to analyzes of unlimited sophistication. However, they were not designed with a graphical user interfaces (GUI's), or wizards typical for proprietary GIS packages. Because of this, they are not easily used by non-experts. There is thus opportunity both for proprietary GIS to incorporate regression-kriging ideas and for open-source software to become more user-friendly.

3.4.4 Towards a system for automated mapping

Geostatistics provides a set of mathematical tools that have been used now over 50 years to generate maps from point observations and to model the associated uncertainty. It has proven to be an effective tool for a large number of applications ranging from mining and soil and vegetation mapping to environmental monitoring and climatic modeling. Several years ago, geostatistical analysis was considered to be impossible without the intervention of a spatial analyst, who would manually fit variograms, decide on the support size and elaborate on selection of the interpolation technique. Today, the heart of a mapping project can be the computer program that implements proven and widely accepted (geo)statistical prediction methods. This leads to a principle of **automated mapping** where the analyst focuses his work only on preparing the inputs and supervising the data processing[85]. This way, the time and resources required to go from field data to the final GIS product (geoinformation) are used more efficiently.

Automated mapping is still utopia for many mapping agencies. At the moment, environmental monitoring groups worldwide tend to run analyzes separately, often with incorrect techniques, frequently without making the right conclusions, and almost always without considering data and/or results of adjacent mapping groups. On one side, the amount of field and remotely sensed data in the world is rapidly increasing (see section 4); on the other side, we are not able to provide reliable information to decision makers in near real-time. It is increasingly necessary that we automate the production of maps (and models) that depict environmental information. In addition, there is an increasing need to bring international groups together and start "*piecing together a global jigsaw puzzle*"[86] to enable production of a global harmonized GIS of all environmental resources. All this proves that automated mapping is an emerging research field and will receive significant attention in geography and Earth sciences in general (Pebesma et al., 2009).

A group of collaborators, including the author of this book, have begun preliminary work to design, develop, and test a web-based automated mapping system called **auto-map.org**. This web-portal should allow the users to upload their point data and then: (a) produce the best linear predictions depending of the nature/-type of a target variable, (b) interpret the result of analysis through an intelligent report generation system, (c) allow interactive exploration of the uncertainty, and (d) suggest collection of additional samples — all at click of button. The analysis should be possible via a web-interface and through e.g. Google Earth plugin, so that various users can access outputs from various mapping projects. All outputs will be coded using the HTML and Google Earth (KML) language. Registered users will be able to update the existing inputs and re-run analysis or assess the quality of maps (Fig. 3.15). A protocol to convert and synchronize environmental variables coming from various countries/themes will need to be developed in parallel (based on GML/GeoSciML[87]).

[85] See for example outputs of the INTAMAP project; `http://www.intamap.org`.

[86] Ian Jackson of the British Geological Survey; see also the `http://www.onegeology.org` project.

[87] `http://www.cgi-iugs.org`

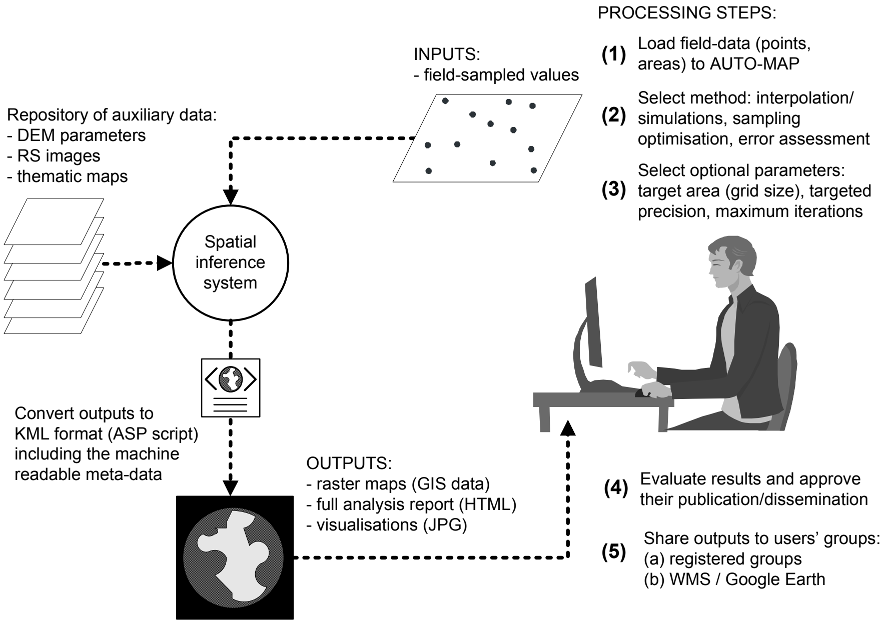

PROCESSING STEPS:
(1) Load field-data (points, areas) to AUTO-MAP
(2) Select method: interpolation/ simulations, sampling optimisation, error assessment
(3) Select optional parameters: target area (grid size), targeted precision, maximum iterations
INPUTS:
- field-sampled values
Repository of auxiliary data:
- DEM parameters
- RS images
- thematic maps
Spatial inference system
Convert outputs to KML format (ASP script) including the machine readable meta-data
OUTPUTS:
- raster maps (GIS data)
- full analysis report (HTML)
- visualisations (JPG)
(4) Evaluate results and approve their publication/dissemination
(5) Share outputs to users' groups:
(a) registered groups
(b) WMS / Google Earth

Fig. 3.15: A proposal for the flow of procedures in auto-map.org: a web-based system for automated predictive mapping using geostatistics. The initial fitting of the models should be completely automated; the user then evaluates the results and makes eventual revisions.

There would be many benefits of having a robust, near-realtime automated mapping tool with a friendly web-interface. Here are some important ones:

- the time spent on data-processing would be seriously reduced; the spatial predictions would be available in near real time;
- through a browsable GIS, such as Google Earth, various thematic groups can learn how to exchange their data and jointly organize sampling and interpolation;
- the cost-effectiveness of the mapping would increase:
  - budget of the new survey projects can be reduced by optimising the sampling designs;
  - a lower amount of samples is needed to achieve equally good predictions;

It is logical to assume that software for automated mapping will need to be *intelligent*. It will not only be able to detect anomalies, but also to communicate this information to users, autonomously make choices on whether to mask out parts of the data sets, use different weights to fit the models or run comparison for various alternatives. This also means that development of such a system will not be possible without a collaboration between geostatisticians, computer scientists and environmental engineers.

Many geostatisticians believe that map production should never be based on a *black-box* system. The author of this guide agrees with these views. Although data processing automation would be beneficial to all, analysts should at any time have the control to adjust the automated mapping system if needed. To do this, they should have full insight into algorithms used and be able to explore input data sets at any moment.

Further reading:

★ Bolstad, P., 2008. **GIS Fundamentals**, 3rd edition. Atlas books, Minnesota, 620 p.

★ Burns, P. 2009. **The R Inferno**. Burns statistics, London, 103 p.

★ Conrad, O. 2007. SAGA — program structure and current state of implementation. In: Böhner, J., Raymond, K., Strobl, J., (eds.) **SAGA — Analysis and modeling applications**, Göttinger Geographische abhandlungen, Göttingen, pp. 39-52.

★ Rossiter, D.G., 2007. **Introduction to the R Project for Statistical Computing for use at ITC**. International Institute for Geo-information Science & Earth Observation (ITC), Enschede, Netherlands, 136 p.

★ Venables, W. N. and Ripley, B. D., 2002. **Modern applied statistics with S**. Statistics and computing. Springer, New York, 481 p.

★ Zuur, A. F., Ieno, E. N., Meesters, E. H. W. G., 2009. **A Beginner's Guide to R**. Springer, Use R series, 228 p.

★ `http://www.52north.org` — 52° North initiative responsible for the distribution of ILWIS GIS.

★ `http://www.saga-gis.org` — homepage of the SAGA GIS project.

★ `http://cran.r-project.org/web/views/Spatial.html` — CRAN Task View: Analysis of Spatial Data maintained by Roger Bivand.

4

Auxiliary data sources

As mention previously in §2.10, geostatistical techniques increasingly rely on the availability of auxiliary data sources, i.e. maps and images of surface and sub-surface features. This chapter reviews some of the widely known sources of remote sensing and digital cartographic data that are of interest for various geostatistical mapping applications. I will first focus on the freely available global data sets and remotely sensed data, and then give an example of how to download and import to GIS various MODIS products using R scripts. A small repository of global maps at 0.1 arcdegree (10 km) resolution is also available from the authors website[1] (see further chapter 7).

4.1 Global data sets

Global maps of our environment (both remote sensing-based and thematic) are nowadays increasingly attractive for environmental modeling at global and continental scales. A variety of publicly available maps can be obtained at no cost at resolutions up to 1 km or better, so that the issue for mapping teams is not any more whether to use this data, but where to obtain it, how to load it to an existing GIS and where to find necessary metadata. The most well known global data sets (sorted thematically) are:

Height/geomorphology data — Global **SRTM Digital Elevation Model** is possibly the most known global environmental data set (Rabus et al., 2003). The area covered is between 60° North and 58° South. It was recorded by X-Band Radar (NASA and MIL, covering 100% of the total global area) and C-Band Radar (DLR and ASI, covering 40%). The non-public DLR-ASI data is available with a resolution of approximately 30 m (1 arcsec). A complete land surface model **ETOPO1 Global Relief Model**[2] (Fig. 4.1; includes bathymetry data) is available at resolution of 1 km and can be obtained from the NOAA's National Geophysical Data Center (Amante and Eakins, 2008). The 90 m SRTM DEMs can be obtained from the CGIAR[3] — Consortium for Spatial Information. An updated 1 km resolution global topography map (SRTM30 PLUS[4]; used by Google Earth) has been prepared by Becker et al. (2009). From June 2009, ASTER-based Global Digital Elevation Model (**GDEM**) at resolution of 30 m has been made publicly available. The GDEM was created by stereo-correlating the 1.3 million-scene ASTER archive of optical images, covering almost 98% of Earth's land surface (Hayakawa et al., 2008). The one-by-one-degree tiles can be downloaded from NASA's EOS data archive and/or Japan's Ground Data System[5].

Administrative data — Administrative data can be used to calculate proximity-based parameters and to orient the users geographically. One such global administrative data database is the **Global Administra-**

[1] `http://spatial-analyst.net/worldmaps/` — this repository is constantly updated.

[2] `http://ngdc.noaa.gov`

[3] `http://srtm.csi.cgiar.org`

[4] `http://topex.ucsd.edu/WWW_html/srtm30_plus.html`

[5] `http://data.gdem.aster.ersdac.or.jp`

tive Areas (GADM[6]) data set. It comprises borders of countries and lower level subdivisions such as provinces and counties (more than 100,000 areas). Another important global data set is the **World Vector Shoreline data set**[7] at scale 1:250,000 (Soluri and Woodson, 1990). This can be, for example, used to derive the global distance from the sea coast. Eight general purpose thematic layers: boundaries, transportation, drainage, population centers, elevation, vegetation, land use and land cover (al at scale 1:1,000,000) can be obtained via the **Global Map Data project**[8].

Socio-economic data — The most important global socio-economic data layers are the population density maps and attached socio-economic variables. The Socioeconomic Data and Applications Center (SEDAC[9]) distributes the **global population density maps** at resolution of 1 km for periods from 1990 up to 2015 (projected density).

Water resources — The most detailed and the most accurate inventory of the global water resources is the **Global Lakes and Wetlands Database** (GLWD[10]), which comprises lakes, reservoirs, rivers, and different wetland types in the form of a global raster map at 30-arcsec resolution (Lehner and Doll, 2004). Shapefiles of the **World basins** and similar vector data can be best obtained via the Remote Sensing and GIS Unit of the International Water Management Institute (IWMI).

Lights at night images — Images of lights at night have shown to be highly correlated with industrial activity and Gross Domestic Product (Doll et al., 2007). A time-series of **annual global night light images** is available via NOAA's National Geophysical Data Center[11]. The lights at night map contains the lights from cities, towns, and other sites with persistent lighting, including gas flares. The filtered annual composites are available from 1992 until 2003.

Land cover maps — Land cover maps are categorical-type maps, commonly derived using semi-automated methods and remotely sensed images as the main input. A Global Land Cover map for the year 2000 (**GLC2000**[12]) at 1 km resolution is distributed by the Joint Research Centre in Italy (Bartholome at al., 2002). A slightly outdated (1998) global map of land cover is the AVHRR **Global Land Cover Classification**[13], provided at resolutions of 1 and 8 km (Hansen et al., 2000). More detailed land cover maps are distributed nationally. Ellis and Ramankutty (2000) prepared the first global map of the anthropogenic biomes (18 classes) showing dense settlements, villages, croplands, rangelands, forested lands and wildlands. The International Water Management Institute also produced the **Global map of Irrigated Areas**[14] (GMIA; 28 classes) and the Global map of Rainfed Cropped Areas (GMRCA), both at 10 km resolution, and based on twenty years of AVHRR images, augmented with higher resolution SPOT and JERS-1 imagery.

Climatic maps — Hijmans et al. (2005) produced **global maps of bioclimatic parameters** (18) derived (thin plate smoothing splines) using >15,000 weather stations. The climatic parameters include: mean, minimum and maximum temperatures, monthly precipitation and bioclimatic variables[15].

Ecoregions / Biogeographic regions — Ecoregions are terrestrial, freshwater and/or marine areas with characteristic combinations of soil and landform that characterize that region. Olson et al. (2001) produced the **Terrestrial Ecoregions**[16] global data set, which shows some 867 distinct eco-units, including the relative richness of terrestrial species by ecoregion. A somewhat more generalized is the **FAO's map of Eco-floristic regions**[17] (e.g. boreal coniferous forest, tropical rainforest, boreal mountain system etc.).

Soil / Geology maps — USGS produced a detailed **Global Soil Regions map**[18] at resolution of 60 arcsec. FAO, IIASA, ISRIC, ISSCAS, JRC have recently produced a 1 km gridded map, merged from various na-

[6] `http://biogeo.berkeley.edu/gadm/`

[7] `http://rimmer.ngdc.noaa.gov/mgg/coast/wvs.html`

[8] `http://www.iscgm.org`

[9] `http://sedac.ciesin.columbia.edu/gpw/`

[10] `http://www.worldwildlife.org/science/data/item1877.html`

[11] `http://ngdc.noaa.gov/dmsp/global_composites_v2.html`

[12] `http://www-tem.jrc.it/glc2000/`

[13] `http://glcf.umiacs.umd.edu/data/landcover/data.shtml`

[14] `http://www.iwmigiam.org/info/gmia/`

[15] The 1 km resolution maps can be obtained via `http://worldclim.org`.

[16] `http://www.worldwildlife.org/science/ecoregions/item1267.html`

[17] `http://cdiac.ornl.gov/ftp/global_carbon/`

[18] `http://soils.usda.gov/use/worldsoils/mapindex/order.html`

tional soil maps, which is also known as the **Harmonized World Soil Database**[19] (v 1.1). The geological maps are now being integrated via the **OneGeology** project. USDA Soil Survey Division also distributes a global map of wetlands (includes: upland, lowland, organic, permafrost and salt affected wetlands). International Soil Reference Information Center (ISRIC) maintains a global soil profile database with over 12,000 profiles and over 50 analytical and descriptive parameters (Batjes, 2009). From NOAA's National Geophysical Data Center one can obtain a point map with all major earth quakes (**Significant Earthquake Database**[20]; with cca 5000 quakes).

Forest / wildlife resources — There are two important global forest/wildlife data sets: (1) The **world map of intact forest landscapes**[21] (hardly touched by mankind) at a scale 1:1,000,000 (includes four classes of intact forests: 1. intact closed forests; 2. intact open forests, 3. woodlands and savannas, closed forests; and 4. open forests, woodlands and savannas) — maintained by the Greenpeace organization (Potapov, P. et al., 2008), and (2) **World Wilderness Areas**[22] at scale 1:1,000,000 — distributed via the UNEP GEO Data Portal (McCloskey and Spalding, 1989).

Biodiversity / human impacts maps — Global maps of biodiversity measures for various groups of taxa (e.g. vascular plants, birds and mammals) can be browsed using the World Atlas of Biodiversity viewer (Groombridge and Jenkins, 2002). Similar type of maps can be browsed via the UNEP's World Conservation Monitoring Centre[23]. Kreft and Jetz (2007) recently produced a **global map of plant species diversity** (number of plant species) by using field records from 1,032 locations. Partners in the GLOBIO consortium created a **World Map of Human Impacts**[24] on the Biosphere. This is basically a map showing a current status of the roads, railways and settlement density. The Carbon Dioxide Information Analysis Center provides access to numerous ecological layers and data of interest to global ecologists. One such product is the **Global Biomass Carbon Map**[25] (Carbon density tones of C ha^{-1}), prepared for year 2000 (Ruesch and Gibbs, 2008a).

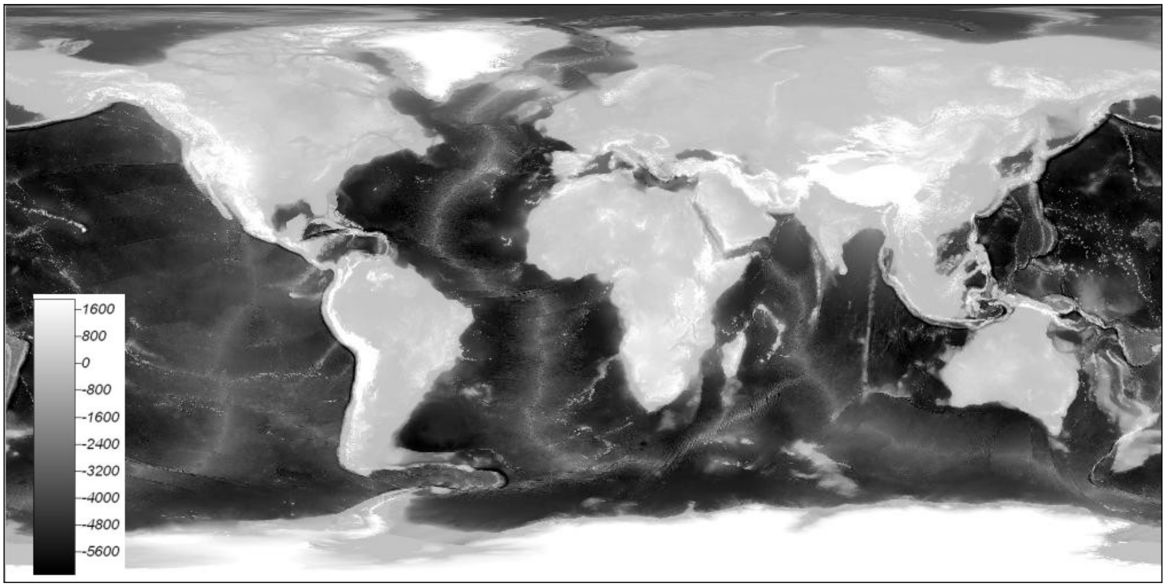


Fig. 4.1: The 1 km resoluton ETOPO1 Global Relief Model (includes bathymetry data). Elevation is probably the most widely used global environmental data set — it is available globally at resolutions from 1 km up to 90 m (SRTM) i.e. 60 m (GDEM). Elevation can be used to extract over 100 unique land surface parameters.

[19] http://www.fao.org/nr/water/news/soil-db.html
[20] http://ngdc.noaa.gov/hazard/earthqk.shtml
[21] http://www.intactforests.org/
[22] http://geodata.grid.unep.ch/
[23] http://www.unep-wcmc.org/
[24] http://www.globio.info/region/world/
[25] http://cdiac.ornl.gov/ftp/

4.1.1 Obtaining data via a geo-service

Much of the geo-data can be accessed directly through R i.e. via some of its packages specialized in getting geo-data via some web-service. Paul Wessel, from the University of Hawai'i, maintains **A Global Self-consistent, Hierarchical, High-resolution Shoreline Database**[26], which is described in detail in Wessel and Smith (1996). These vectors can be imported to R by using the maptools package (see `Rgshhs` method).

Some basic (and slightly out-dated) vector maps are available in the R's package maps. This contains data map of political borders, world cities, **CIA World Data Bank II**[27] data, administrative units from the NUTS III (Tertiary Administrative Units of the European Community) and similar. To obtain these vector maps for your GIS, you can run:

```
> library(maps)
> worldmap <- map2SpatialLines(map("world", fill=TRUE, col="transparent",
+      plot=FALSE), proj4string=CRS("+proj=longlat +ellps=WGS84"))
> worldmap <- SpatialLinesDataFrame(worldmap, data.frame(name=(map("world"))$names),
+      match.ID=F)
> writeOGR(worldmap, "worldmap.shp", "worldmap", "ESRI Shapefile")
```

The original shapefiles of World political borders can be obtained directly via the `thematicmapping.org`[28]. The lower level administrative boundaries (global coverage) can be obtained from the **FAO's GeoNetwork server**[29].

If you wish to obtain similar type of geographic information but only for a specific point location, you should consider using some of the free web-services such as GeoNames (also available via the R package GeoNames). For example, to obtain elevation, name of the closest city and/or actual weather at some point location, we can run:

```
> library(geonames)
> GNfindNearbyPlaceName(lat=47,lng=9)

  name   lat  lng    geonameId   countryCode   countryName   fcl  fcode  distance
  Atzmännig     47.287633   8.988454    6559633   CH   Switzerland  P   PPL  1.6276


> GNgsrtm3(lat=47,lng=9)

    srtm3 lng lat
  1  2834   9  47


> GNweather(north=47,east=8,south=46,west=9)

        clouds weatherCondition
  1 few clouds              n/a
                                                  observation
  1 LSZA 231320Z VRB02KT 9999 FEW045 BKN060 04/M02 Q0991 NOSIG
    ICAO      lng temperature dewPoint windSpeed
  1 LSZA 8.966667           4       -2        02
    humidity stationName            datetime lat
  1       64      Lugano 2009-01-23 14:20:00  46
    hectoPascAltimeter
  1               991
```

Another alternative is Google's maps service, which allows you to obtain similar information. For example, you can use Google's geographic services (see also coverage detail of Google maps[30]) to get geographic coordinates given a street + city + country address. First, register your own Google API key. To geocode an address, you can run in R:

[26] `http://www.soest.hawaii.edu/wessel/gshhs/gshhs.html`

[27] `http://www.evl.uic.edu/pape/data/WDB/`

[28] `http://thematicmapping.org/downloads/world_borders.php`

[29] `http://www.fao.org/geonetwork/srv/en/main.home`

[30] `http://en.wikipedia.org/wiki/Coverage_details_of_Google_Maps`

```
> readLines(url("http://maps.google.com/maps/geo?q=1600+Amphitheatre+Parkway,
+       +Mountain+View,+CA&output=csv&key=abcdefg"), n=1, warn=FALSE)
```

which will give four numbers: 1. HTTP status code, 2. accuracy, 3. latitude, and 4. longitude. In the case from above:

```
  [1] 200.00000 8.00000 37.42197 -122.08414
```

where the status code is 200 (meaning “No errors occurred; the address was successfully parsed and its geocode has been returned”[31]), the geocoding accuracy is 8 (meaning highly accurate; see also the accuracy constants), longitude is 37.42197 and the latitude is -122.08414. Note that the address of a location needs to be provided in the following format:

```
  "StreetNumber+Street,+City,+Country"
```

A large number of maps can be also obtained via some of the many commercial WCS’s[32]. A popular WMS that allows download of the original vector data is **Openstreetmap**[33]. The original data come in the OSM (Open Street Map) format, but can be easily exported and converted to e.g. ESRI shapefiles using the OpenJUMP GIS[34]. Another extensive WMS is NASA’s **OnEarth**[35].

4.1.2 Google Earth/Maps images

You can also consider obtaining color composites of the high resolution imagery (QuickBird, Ikonos) that are used in Google Earth[36]. With RgoogleMaps[37] package you can automate retrieval and mosaicking of images. For example, to obtain a hybrid satellite image of the Netherlands, it is enough to define the bounding box, position of the center and the zoom level (scale):

```
> library(RgoogleMaps)
# Get the Maximum zoom level:
> mzoom <- MaxZoom(latrange=c(50.74995, 53.55488), lonrange=c(3.358871 7.227094),
+       size=c(640, 640))[[1]]
> mzoom

  [1] 7


# Get a satellite image of the Netherlands:
> MyMap <- GetMap.bbox(center=c(52.1551723, 5.3872035), zoom=mzoom,
+       destfile="netherlands.png", maptype="hybrid")

  Read 1 item
  [1] "http://maps.google.com/staticmap?center=52.15517,5.38720&zoom=7&size=640x640
  +   &maptype=hybrid&format=png32&key=****&sensor=true"
  trying URL 'http://maps.google.com/staticmap?center=52.15517,5.38720&zoom=7
  +   &size=640x640&maptype=hybrid&format=png32&key=****=true'
  Content type 'image/png' length 703541 bytes (687 Kb)
  opened URL
  downloaded 687 Kb

  netherlands.png has GDAL driver PNG
  and has 640 rows and 640 columns


> PlotOnStaticMap(MyMap, lat=52.1551723, lon=5.3872035)
```

[31] See also the status code table at: http://code.google.com/apis/maps/documentation/reference.html.

[32] http://www.ogcnetwork.net/servicelist — a list of Open Geospatial Consortium (OGC) WMS’s.

[33] http://www.openstreetmap.org/ — see *export* tab.

[34] http://wiki.openstreetmap.org/index.php/Shapefiles

[35] http://onearth.jpl.nasa.gov/

[36] This has some copyright restrictions; see http://www.google.com/permissions/geoguidelines.html.

[37] http://cran.r-project.org/web/packages/RgoogleMaps/; see also webmaps package.

The tiles obtained using RgoogleMaps are blocks of maximum 640×640 pixels distributed as PNG or JPG (compressed) images, which makes them of limited use compare to the multi-band satellite images we receive from the original distributors. Nevertheless, Google contains high resolution imagery of high spatial quality (Potere, 2008), which can be used to extract additional content for a smaller size GIS e.g. to digitize forest borders, stream lines and water bodies. Before you can extract content, you need to attach coordinates to the Static map i.e. you need to georeference it. To achieve this, we first need to estimate the bounding box coordinates of the image. This is possible with the help of the XY2LatLon.R script:

```
# Get the XY2LatLon.R script from CRAN:
> download.file("http://cran.r-project.org/src/contrib/RgoogleMaps_1.1.6.tar.gz",
+     destfile=paste(getwd(), "/", "RgoogleMaps.tar.gz", sep=""))
> library(R.utils)
> gunzip("RgoogleMaps.tar.gz", overwrite=TRUE, remove=TRUE)
# under windows, you need to use 7z software to unzip *.tar archives:
> download.file("http://downloads.sourceforge.net/sevenzip/7za465.zip",
+     destfile=paste(getwd(), "/", "7za465.zip", sep=""))
> unzip("7za465.zip")
> system("7za e -ttar RgoogleMaps.tar XY2LatLon.R -r -aos")

  7-Zip (A) 4.65  Copyright (c) 1999-2009 Igor Pavlov  2009-02-03

  Processing archive: RgoogleMaps.tar

  Extracting  RgoogleMaps\R\XY2LatLon.R

  Everything is Ok

  Size:       1001
  Compressed: 1228800


> source("XY2LatLon.R")
# Read the Google Static image into R:
> png.map <- readGDAL("netherlands.png")

  netherlands.png has GDAL driver PNG
  and has 640 rows and 640 columns


# Estimate the bounding box coordinates:
> bbox.MyMap <- data.frame(XY2LatLon(MyMap, X=png.map@bbox[1,]-640/2,
+      Y=png.map@bbox[2,]-640/2))
> google.prj <- "+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0
+      +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +wktext +no_defs"
> coordinates(bbox.MyMap) <- ~ lon+lat
> proj4string(bbox.MyMap) <- CRS("+proj=longlat +ellps=WGS84")
# Estimate coordinates in Google Maps projection system:
> bbox.google.prj <- spTransform(bbox.MyMap, CRS(google.prj))
> bbox.google.prj

  SpatialPoints:
            lon     lat
  [1,] 756387.0 7035766
  [2,] 768616.9 7047996
  Coordinate Reference System (CRS) arguments: +proj=merc
  +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0
  +y_0=0 +k=1.0 +units=m +nadgrids=@null +no_defs
```

then attach the right georeference (east, west, north, south coordinates):

```
# copy the bounding box coordinates:
> png.map@bbox[1,1] <- bbox.google.prj@coords[1,1]
> png.map@bbox[2,1] <- bbox.google.prj@coords[1,2]
> png.map@bbox[1,2] <- bbox.google.prj@coords[2,1]
> png.map@bbox[2,2] <- bbox.google.prj@coords[2,2]
# cell size:
> png.map@grid@cellsize <- round(c((png.map@bbox[1,2]-png.map@bbox[1,1])/640,
+       (png.map@bbox[2,2]-png.map@bbox[2,1])/640), 1)
> png.map@coords[1,1] <- png.map@bbox[1,1]+png.map@grid@cellsize[1]/2
> png.map@coords[1,2] <- png.map@bbox[2,1]+png.map@grid@cellsize[2]/2
> png.map@coords[2,1] <- png.map@bbox[1,2]-png.map@grid@cellsize[1]/2
> png.map@coords[2,2] <- png.map@bbox[2,2]-png.map@grid@cellsize[2]/2
# cell offset:
> png.map@grid@cellcentre.offset <- c(png.map@bbox[1,1]+png.map@grid@cellsize[1]/2,
+       png.map@bbox[2,1]+png.map@grid@cellsize[2]/2)
# attach the correct prj:
> proj4string(png.map) <- CRS(google.prj)
# Export map to GIS format:
> write.asciigrid(png.map[1], "netherlands_B1.asc", na.value=-1)
> writeGDAL(netherlands[c(1,2,3)], "netherlands.tif", drivername="GTiff",
+       type="Byte", options="INTERLEAVE=PIXEL")
```

Note that Google Maps currently uses a modification of the Mercator projection system[38] to display tiles. To reproject this grid to some other system consider using the SAGA proj4 functionality, as explained further in §5.6.2.

We can also just quickly estimate how many tiles were used to generate this map by using the maptiler Python script[39] (courtesy of Klokan Petr Pr̂idal):

```
# You need to first install and register Python on your machine!
> download.file("http://www.maptiler.org/google-maps-coordinates-tile-... [TRUNCATED]
+        destfile=paste(getwd(), "/", "globalmaptil ..." ... [TRUNCATED]

  trying URL 'http://www.maptiler.org/google-...-projection/globalmaptiles.py'
  Content type 'text/plain' length 16529 bytes (16 Kb)
  opened URL
  downloaded 16 Kb


> system(paste("python.exe globalmaptiles.py", mzoom, NLprovs.ll@bbox[2,1],
+           NLprovs.ll@bbox[1,1], NLprovs.ll@bbox[2,2], NLprovs.ll@bbox[1,2]))

  Spherical Mercator (ESPG:900913) coordinates for lat/lon:
  (373907.80392051308, 6577181.3183709756)
  Spherical Mercator (ESPG:900913) cooridnate for maxlat/maxlon:
  (804516.38277077128, 7086303.4059718344)
  7/65/85 ( TileMapService: z / x / y )
          Google: 65 42
          Quadkey: 1202021 ( 6281 )

          EPSG:900913 Extent:  (313086.06785608083, 6574807.4249777198,
          626172.13571216539, 6887893.4928338043)
          WGS84 Extent: (50.736455137010637, 2.8124999999999902,
          52.482780222078226, 5.6250000000000133)
          gdalwarp -ts 256 256 -te 313086.067856 6574807.42498 626172.135712
          6887893.49283 <your-raster-file-in-epsg900913.ext> 7_65_85.tif

  7/66/85 ( TileMapService: z / x / y )
          Google: 66 42
          Quadkey: 1202030 ( 6284 )

          EPSG:900913 Extent:  (626172.13571216539, 6574807.4249777198,
```

[38] http://www.spatialreference.org/ref/user/google-projection/

[39] http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/

```
          939258.20356824622, 6887893.4928338043)
          WGS84 Extent: (50.736455137010637, 5.6250000000000133,
          52.482780222078226, 8.4375000000000036)
          gdalwarp -ts 256 256 -te 626172.135712 6574807.42498 939258.203568
          6887893.49283 <your-raster-file-in-epsg900913.ext> 7_66_85.tif

  7/65/86 ( TileMapService: z / x / y )
          Google: 65 41
          Quadkey: 1202003 ( 6275 )

          EPSG:900913 Extent:  (313086.06785608083, 6887893.4928338043,
          626172.13571216539, 7200979.5606898852)
          WGS84 Extent: (52.482780222078226, 2.8124999999999902,
          54.162433968067809, 5.6250000000000133)
          gdalwarp -ts 256 256 -te 313086.067856 6887893.49283 626172.135712
          7200979.56069 <your-raster-file-in-epsg900913.ext> 7_65_86.tif

  7/66/86 ( TileMapService: z / x / y )
          Google: 66 41
          Quadkey: 1202012 ( 6278 )

          EPSG:900913 Extent:  (626172.13571216539, 6887893.4928338043,
          939258.20356824622, 7200979.5606898852)
          WGS84 Extent: (52.482780222078226, 5.6250000000000133,
          54.162433968067809, 8.4375000000000036)
          gdalwarp -ts 256 256 -te 626172.135712 6887893.49283 939258.203568
          7200979.56069 <your-raster-file-in-epsg900913.ext> 7_66_86.tif
```

which shows that four tiles are needed to represent the whole of the Netherlands at zoom level 7. Again, you need to be aware that these maps/images are copyrighted, so you should really use them only for your personal purpose. In addition, access to Google imagery is possible only via the Google maps API, which means that you first need to register your key etc. If you really require detailed remotely sensed images for larger study area, then consider obtaining some of the popular RS products (Landsat, Spot, ASTER, Ikonos) from the official data distributers.

A number of similar Python scripts can be found at GDAL utilities section[40]. For example, if you have a relatively large raster map and you want to export it to KML (*Super Overlays*), you can create a Google Earth-type of hierarchical images by using the `gdal2tiles.py` script (also available through FWtools). This will automatically split the map into tiles, generate a directory with small tiles and metadata following the OSGeo Tile Map Service Specification[41].

4.1.3 Remotely sensed images

Remotely sensed images are increasingly the main source of data for many national and continental scale mapping projects. The amount of field and remotely sensed data in the world is rapidly increasing. To get an idea about how many sensors are currently available, see for example the ITC's database[42]. The most common satellites/images with global coverage that are available at low cost or for free, and which are of interest for global modeling projects, are (Fig. 4.2):

Landsat Landsat collects multispectral satellite imagery at resolutions 15–30 meters. A number of their products is also available at no cost. High resolution (15 m) Landsat images for nearly all of the world (for years 1990 and 2000) can be downloaded from the **NASA's Seamless Server**[43]. European (harmonized) mosaic of Landsat images is distributed by JRC Ispra (see **Image2000**[44]). Another excellent repository of free global imagery is the **GLCF geo-portal**[45] operated by the University of Maryland.

[40] `http://www.gdal.org/gdal_utilities.html`

[41] `http://code.google.com/apis/kml/articles/raster.html`

[42] `http://www.itc.nl/research/products/sensordb/AllSensors.aspx`

[43] `http://seamless.usgs.gov/`

[44] `http://image2000.jrc.ec.europa.eu`

[45] `http://glcf.umiacs.umd.edu/portal/geocover/`

SPOT SPOT is a commercial distributor of high quality resolution satellite imagery (multispectral images at resolution of 2.5–20 m). The most recent satellite SPOT 6 carries a High Resolution Stereoscopy (HRS) sensor that allows production of 3D imagery. **SPOT vegetation**[46] offers relatively coarse vegetation-based 10–day images of the whole Earth collected in the period from 1998 until today. Only two bands are available at the moment: NDVI and radiometry images.

Ikonos The IKONOS sensor (Satellite) is a high-resolution commercial satellite operated by GeoEye company. The images are available at two resolutions: 3.2 m (multispectral, Near-Infrared), and 0.82 m (panchromatic) (Dial et al., 2003). Ikonos images are sold per km^2 (a standard scene is of size 12,5×12,5 km); the user defines an area of interest of any shape but with a minimum surface area. Archived Ikonos images at discounted price can be obtained via various companies.

Meteosat The Meteosat[47] Second Generation (MSG) satellites (from Meteosat-8 onwards) produce SEVIRI (Spinning Enhanced Visible and Infrared Imagery) 15–minutes meteorological images at a resolution of 1 km. The most attractive data set for environmental applications is the **High Rate SEVIRI**, which consists of 12 spectral channels including: visible and near infrared light, water vapor band, carbon dioxide and ozone bands.

ENVISAT The ENVISAT satellite is a platform for several instruments adjusted for monitoring of the environmental resources: ASAR, MERIS, AATSR, MWR and similar. The MEdium Resolution Image Spectrometer (**MERIS**[48]) is used to obtain images of the Earth's surface at a temporal resolution of 3–days. The images comprise 15 bands, all at a resolution of 300 m. To obtain MERIS images (Category 1 use data), one needs to register and wait few days to receive access to the repository[49] (unlike the MODIS images that are available directly via an FTP).

AVHRR Although outdated, AVHRR was the one of the main global environmental monitoring systems in the 80's. Principal components derived from a set of 232 **monthly NDVI images**[50] can be obtained from the author's repository of world maps. The original images are available at a resolution of 300 arcseconds (cca 10 km), and cover the period July, 1981 through September, 2001.

MODIS MODIS contains a number of products ranging from raw multispectral images, to various vegetation and atmospheric indices at a resolution of 250 m (also available at coarser resolutions of 500 m and 1 km), and at very high temporal resolution. If you only wish to use completed MODIS products, then look at NASA's Earth Observation (**NEO**[51]) portal, which distributes global time-series of MODIS-derived parameters such as: snow cover and ice extent, leaf area index, land cover classes, net primary production, chlorophyll concentration in the sea and sea surface temperature, cloud water content, carbon monoxide in the atmosphere and many more. All global maps on NEO are freely available for public use. You can simply download the 0.1 arcdegrees (∼10 km) GeoTiff's and load them to your GIS. Please credit NASA as the source.

From the RS systems listed above, one needs to be emphasized — NASA's MODIS Earth observation system — possibly one of the richest sources of remote-sensing data for monitoring of environmental dynamics (Neteler, 2005; Lunetta et al., 2006; Ozdogana and Gutman, 2008). This is due to the following reasons:

(1.) it has global coverage;

(2.) it has a relatively high temporal resolution/coverage (1–2 days; Fig. 4.2);

(3.) it is open-access (see the MODIS licence specification);

(4.) a significant work has been done to filter the original raw images for clouds and artifacts; a variety of complete MODIS products such as composite 15–day and monthly images is available at three resolutions: 250 m, 500 m and 1 km;

[46] `http://www.spot-vegetation.com`

[47] `http://www.eumetsat.int`

[48] `http://envisat.esa.int/instruments/`

[49] `http://earth.esa.int/dataproducts/accessingeodata/`

[50] Available via the International Water Management Institute at `http://iwmidsp.org/`.

[51] `http://neo.sci.gsfc.nasa.gov/`

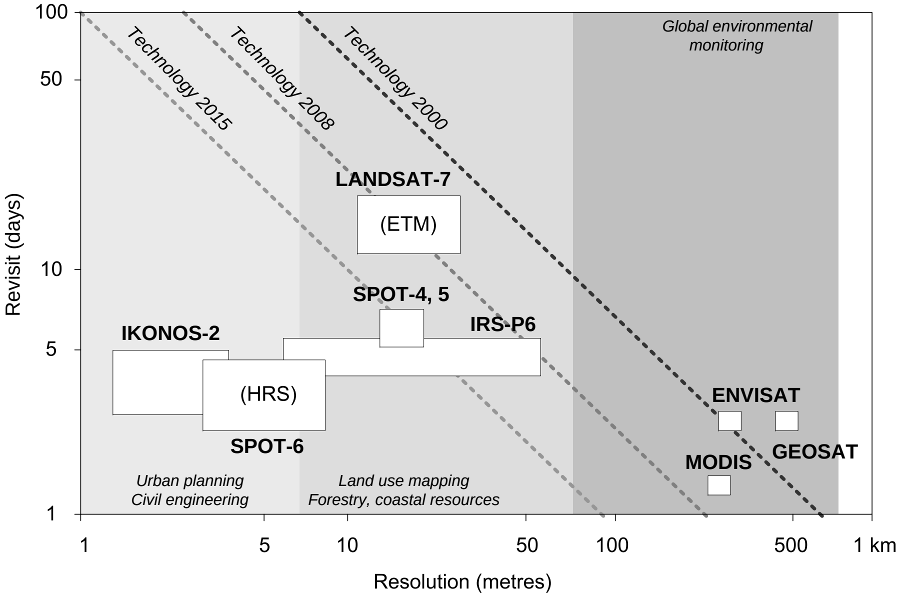


Fig. 4.2: Resolution and revisit time of some common imaging satellites. Modified after Davis et al. (2009).

(5.) efficient tools exist to obtain various MODIS products and import them to various GIS;

One of the best known MODIS products[52] for terrestrial environmental applications is the **Enhanced Vegetation Index** (EVI), which is the improved NDVI (Huete et al., 2002). EVI corrects distortions in the reflected light caused by particles in the air as well as ground cover below the vegetation. The EVI also does not become saturated as easily as the NDVI when viewing rainforests and other areas with large amounts of chlorophyll. EVI can be directly related to the photosynthetic production of plants, and indirectly to the green biomass (Huete et al., 2002). By observing dynamics of EVI for an area of the Earth's surface, we can conclude about the vegetation dynamics within a season, but also detect long-term trends and sudden changes in the biomass (e.g. due to forest fires, deforestation, urban growth and similar).

4.2 Download and preparation of MODIS images

This section explains how to automate download, mosaicking, resampling and import of MODIS product into a GIS. We focus on the Land Surface Temperature (LST) images, which are subsequently used to improve spatio-temporal interpolation of temperatures in chapter 11. Before you can start downloading MODIS images from within R, you need to obtain and install some necessary applications and R libraries:

- RCurl[53] — allows you to list directories on an FTP server;
- wget[54] — this will automate download of images from within R; simply put the `wget.exe` in your Windows system folder[55];
- MRT[56] — MODIS reproject tool can be used to mosaic MODIS images, resample them to other coordinate systems, and export images to more common GIS formats;

After you have finished installing all these software programs, you also need to specify the location of MRT and the directory where you want to output all EVI maps you will produce:

[52] `https://lpdaac.usgs.gov/lpdaac/products/modis_product_table`

[53] `http://www.omegahat.org/RCurl/`

[54] `http://users.ugent.be/~bpuype/wget/`

[55] Note: make sure you disable your antivirus tools such as Norton or McAfee otherwise it might block wget from running. This script has not yet been tested under Mac OS X.

[56] `https://lpdaac.usgs.gov/lpdaac/tools/modis_reprojection_tool`

```
> library(RCurl)
> library(rgdal)
# location of the mosiacing tool:
> MRT <- 'E:\\MODIS\\MRT\\bin\\'
# location of the working directory:
> workd <- 'E:\\MODIS\\HR\\'
# location of the MODIS 1 km blocks:
> MOD11A2 <- "ftp://e4ftl01u.ecs.nasa.gov/MOLT/MOD11A2.005/"
> MOD11A2a <- "ftp://anonymous:test@e4ftl01u.ecs.nasa.gov/MOLT/MOD11A2.005/"
```

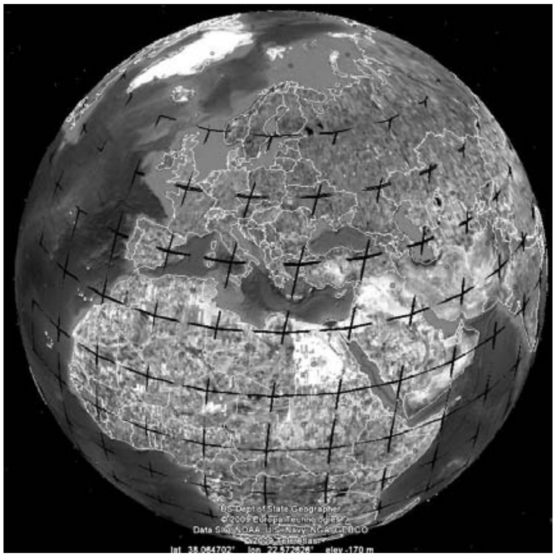

US Dept of State Geographer
© 2009 Europa Technologies
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
© 2009 Tele Atlas
lat 38.064702° lon 22.572826° elev -170 m

Fig. 4.3: MODIS HDF tiles (Sinusoidal grid).

MODIS images are typically distributed as **HDF** (Hierarchical Data Format) 10 by 10 arcdegree-tiles, projected in the sinusoidal projection (Fig. 4.3). The sinusoidal projection has been promoted by geographers as the most suited projection for global image databases (Chang Seong et al., 2002). Unfortunately, both HDF format and sinusoidal projection are yet not supported in many GIS. Therefore, before you can use MODIS images, you will need to run some pre-processing to glue the tiles and convert the data into a more usable format. We wish to obtain the 8–day 1 km resolution MODIS LST images[57], i.e. the average values of clear-sky LSTs during the 8–day periods. We will first download the original tiles, then mosaic them and resample them to the UTM projection system (zone 33), and then export them to a more common GIS format (GeoTiff).

The MOD13A3 tiles can be browsed and downloaded directly via NASA's FTP server. However, for larger areas, you will often need to download tens of tiles, so that is fairly useful to automate the processing. We start by fetching the list of sub-directories (the dates) of interest:

```
# get the list of directories:
> items <- strsplit(getURL(MOD11A2), "\n")[[1]]
> items[2]

  [1] "drwxr-xr-x  2 90 129024 Dec 12  2008 2000.03.05\r"


# you get the folder names but in form of a unix directory listing
# get the last word of any lines that start with 'd':
> folderLines <- items[substr(items, 1, 1)=='d']
# get the directory names and create a new data frame:
> dirs <- unlist(lapply(strsplit(folderLines, " "), function(x){x[length(x)]}))
> dates <- data.frame(dirname=unlist(strsplit(dirs, "\r")))
> str(dates)

  'data.frame':   430 obs. of  1 variable:
   $ dirname: Factor w/ 430 levels "2000.03.05","2000.03.13",..: 1 2 3 4 5 6 ...
```

which gives 430 dates; we are only interested in year 2006 (i.e. 268–313; i.e. 45 dates). Note that there are some differences between MS-Windows *vs* Mac OS machines in the use of `"\r\n"` *vs* `"\r"` (`"\n"` is echoed out in the R terminal with the file names). Next, we need to known the h/v position of the MODIS blocks. For example, we want to generate EVI images for the whole area of Croatia. This area covers two MODIS tiles: `h18v04` and `h19v04` (Fig. 4.3). Each MODIS tile has a unique name. We can do a directory listing and get the full names of the tiles on the FTP by combining the `getURL` and `grep` methods:

[57] This data set is known by the name `MOD11A2`.

```
> getlist <- strsplit(getURL(paste(MOD11A2, dates$dirname[[1]], "/", sep=""),
+      .opts=curlOptions(ftplistonly=TRUE)), "\r\n")[[1]]
> str(getlist)

   chr [1:1268] "BROWSE.MOD11A2.A2006001.h00v08.005.2008098013929.1.jpg" ...
```

This means that the FTP directory of interest[58] contains a total of 1268 files. We wish to obtain only the names of the HDF files (two tiles) for our area of interest. These can be obtained using the grep method:

```
> dates$BLOCK1 <- rep(NA, length(dates$dirname))
> dates$BLOCK2 <- rep(NA, length(dates$dirname))
> BLOCK1 <- getlist[grep(getlist,
+     pattern="MOD11A2.********.h18v04.*************.hdf")[1]]
> BLOCK2 <- getlist[grep(getlist,
+     pattern="MOD11A2.********.h19v04.*************.hdf")[1]]
> BLOCK1

  [1] "MOD11A2.A2006001.h18v04.005.2008098031505.hdf"
```

which means that we have successfully determined names of the tiles we are interested in downloading. We look only for the first (HDF) file; the second file with the same name but `.XML` extension carries the production metadata[59].

Next, we can download each tile using the `download.file` method, with help from the wget package[60]:

```
> download.file(paste(MOD11A2a, dates$dirname[[1]], "/", BLOCK1,sep=""),
+     destfile=paste(getwd(), "/", BLOCK1, sep=""), mode='wb', method='wget')

  --2009-03-01 18:21:37--  ftp://anonymous:*password*@e4ftl01u.ecs.nasa.gov/MOLT/...
             => `D:/MODIS/HR/MOD11A2.A2006001.h18v04.005.2008098031505.hdf'
  Resolving e4ftl01u.ecs.nasa.gov... 152.61.4.83
  Connecting to e4ftl01u.ecs.nasa.gov|152.61.4.83|:21... connected.
  Logging in as anonymous ... Logged in!
  ==> SYST ... done.    ==> PWD ... done.
  ==> TYPE I ... done.  ==> CWD /MOLT/MOD11A2.005/2006.01.01 ... done.
  ==> SIZE MOD11A2.A2006001.h18v04.005.2008098031505.hdf ... 6933421
  ==> PASV ... done.    ==> RETR MOD11A2.A2006001.h18v04.005.2008098031505.hdf ... done.
  Length: 6933421 (6.61M)
```

Once we have downloaded all tiles of interest, we can mosaic them into a single image. To do this, we use the MODIS Resampling Tool, that you should have already installed on your machine. We first generate a parameter file containing a list of tiles that need to be mosaicked:

```
> mosaicname <- file(paste(MRT, "TmpMosaic.prm", sep=""), open="wt")
> write(paste(workd, BLOCK1, sep=""), mosaicname)
> write(paste(workd, BLOCK2, sep=""), mosaicname, append=T)
> close(mosaicname)
```

and then run the MRT mosaicking tool using this parameter file:

```
# Generate a mosaic:
> shell(cmd=paste(MRT, 'mrtmosaic -i ', MRT, 'TmpMosaic.prm -s
+     "1 0 0 0 0 0 0 0 0 0 0 0" -o ', workd, 'TmpMosaic.hdf', sep=""))

  ******************************************************************************

  MODIS Mosaic Tool (v4.0 February 2008)
  Start Time:  Thu Jul 30 14:05:26 2009
```

[58] `ftp://e4ftl01u.ecs.nasa.gov/MOLT/MOD11A2.005/2006.01.01/`

[59] Much about the HDF tile can be read already from the file name; see also the MODIS Naming Conventions.

[60] You need to download the `wget.exe` to your windows system directory, otherwise you will not be able to download the tiles from R.

```
------------------------------------------------------------------

Input filenames (2):
   D:\MODIS\HR\MOD11A2.A2006001.h18v04.005.2008098031505.hdf
   D:\MODIS\HR\MOD11A2.A2006001.h19v04.005.2008098031444.hdf
Output filename: D:\MODIS\HR\TmpMosaic.hdf
Mosaic Array:
   file[ 0]  file[ 1]

Mosaic : processing band LST_Day_1km
% complete (1200 rows): 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Output mosaic image info
------------------------
output image corners (lat/lon):
    UL:  50.000000000003 0.000000000000
    UR:  50.000000000003 31.114476537210
    LL:  39.999999999999 0.000000000000
    LR:  39.999999999999 26.108145786645

output image corners (X-Y projection units):
    UL:  0.000000000000 5559752.598833000287
    UR:  2223901.039532999974 5559752.598833000287
    LL:  0.000000000000 4447802.079065999947
    LR:  2223901.039532999974 4447802.079065999947

     band               type lines smpls pixsiz      min    max   fill
  1) LST_Day_1km      UINT16  1200  2400 926.6254   7500  65535      0

End Time:  Thu Jul 30 14:05:28 2009

Finished mosaicking!

*******************************************************************************
```

where "-s 1 0 0 0 0 0 0 0 0 0 0" is the definition of the spectral selection (1st band is the 8–Day daytime 1 km grid land surface temperature), and 'TmpMosaic.hdf' is the temporary mosaic HDF image. Next, we need to generate a MRT parameter file that can be use to resample the image to the target coordinate system:

```
# resample to UTM 33N:
> filename <- file(paste(MRT, "mrt2006_01_01.prm", sep=""), open="wt")
> write(paste('INPUT_FILENAME = ', workd, 'TmpMosaic.hdf', sep=""), filename)
> write('  ', filename, append=TRUE)
> write('SPECTRAL_SUBSET = ( 1 )', filename, append=TRUE)
> write('  ', filename, append=TRUE)
> write('SPATIAL_SUBSET_TYPE = OUTPUT_PROJ_COORDS', filename, append=TRUE)
> write('  ', filename, append=TRUE)
> write(paste('SPATIAL_SUBSET_UL_CORNER = (', XUL, YUL, ')'), filename,
+    append=TRUE)
> write(paste('SPATIAL_SUBSET_LR_CORNER = (', XLR, YLR, ')'), filename,
+    append=TRUE)
> write('  ', filename, append=TRUE)
> write(paste('OUTPUT_FILENAME = ', workd, 'LST', dirname1, '.tif', sep=""),
+    filename, append=TRUE)
> write('  ', filename, append=TRUE)
> write('RESAMPLING_TYPE = NEAREST_NEIGHBOR', filename, append=TRUE)
> write('  ', filename, append=TRUE)
> write('OUTPUT_PROJECTION_TYPE = UTM', filename, append=TRUE)
> write('  ', filename, append=TRUE)
> write('OUTPUT_PROJECTION_PARAMETERS = ( ', filename, append=TRUE)
> write(paste(lon.c, lat.c, '0.0'), filename, append=TRUE)
> write(' 0 0.0 0.0', filename, append=TRUE)
> write(' 0.0 0.0 0.0', filename, append=TRUE)
> write(' 0.0 0.0 0.0', filename, append=TRUE)
> write(' 0.0 0.0 0.0 )', filename, append=TRUE)
> write('  ', filename, append=TRUE)
> write('UTM_ZONE = 33', filename, append=TRUE)
> write('DATUM = WGS84', filename, append=TRUE)
> write('  ', filename, append=TRUE)
> write('OUTPUT_PIXEL_SIZE = 1000', filename, append=TRUE)
> write('  ', filename, append=TRUE)
> close(filename)
```

and we can again run the MRT to resample the map:

```
> shell(cmd=paste(MRT, 'resample -p ', MRT, 'mrt2006_01_01.prm', sep=""))

 ******************************************************************************

 MODIS Reprojection Tool (v4.0 February 2008)
 Start Time:  Thu Jul 30 14:09:29 2009

 ------------------------------------------------------------------



 Input image and reprojection info
 ---------------------------------
 input_filename:          D:\MODIS\HR\TmpMosaic.hdf
 output_filename:         D:\MODIS\HR\LST2006_01_01.tif
 input_filetype:          HDF-EOS
 output_filetype:         GEOTIFF
 input_projection_type:   SIN
 input_datum:             WGS84
 output_projection_type:  UTM
 output_zone_code:        33
 output_datum:            WGS84
 resampling_type:         NN
 input projection parameters:  6371007.18 0.00 0.00 0.00 0.00 0.00 0.00 0.00
    86400.00 0.00 0.00 0.00 0.00 0.00 0.00
 output projection parameters: 16.33 44.35 0.00 0.00 0.00 0.00 0.00 0.00 0.00
    0.00 0.00 0.00 0.00 0.00 0.00

 input image corners (lat/lon):
     UL:  50.00 0.00
     UR:  50.00 31.11
     LL:  40.00 0.00
     LR:  40.00 26.11

 input image spatial subset corners (lat/lon):
     UL:  46.55 13.18
     UR:  46.47 19.55
     LL:  42.17 13.31
     LR:  42.11 19.23

      band          select   type lines smpls pixsiz      min    max   fill
   1) LST_Day_1km        1 UINT16  1200  2400 926.6254   7500  65535      0


 SINUSOIDAL PROJECTION PARAMETERS:

    Radius of Sphere:     6371007.181000 meters
    Longitude of Center:     0.000000 degrees
    False Easting:      0.000000 meters
    False Northing:     0.000000 meters
 UNIVERSAL TRANSVERSE MERCATOR (UTM) PROJECTION PARAMETERS:

    Zone:      33
    Semi-Major Axis of Ellipsoid:     6378137.000000 meters
    Semi-Minor Axis of Ellipsoid:     6356752.314245 meters
    Scale Factor at C. Meridian:      0.999600
    Longitude of Central Meridian:     15.000000 degrees

 NNResample : processing band LST_Day_1km
 % complete (487 rows): 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

 Output image info
 -----------------
 output image extents (lat/lon):
     UL:  46.548860613555 13.175657895706
     UR:  46.472377855130 19.561116709234
     LL:  42.166491811106 13.306850810151
     LR:  42.100790881433 19.234156734254

 output image extents (X-Y projection units):
     UL:  360141.478220875026 5156649.305139659904
     UR:  850141.478220875026 5156649.305139659904
     LL:  360141.478220875026 4669649.305139659904
     LR:  850141.478220875026 4669649.305139659904

      band               type lines smpls pixsiz      min    max   fill
   1) LST_Day_1km      UINT16   487   490 1000.0000   7500  65535      0

 End Time:  Thu Jul 30 14:09:30 2009

 Finished processing!

 ******************************************************************************
```

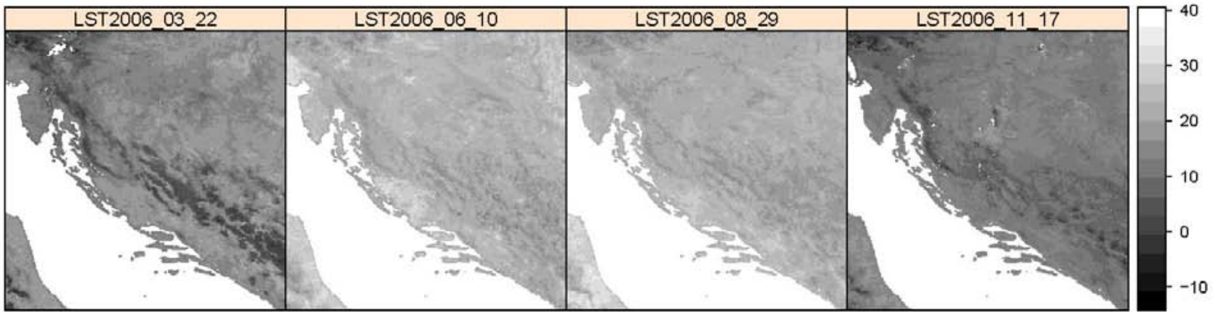


Fig. 4.4: A sample of downloaded and resampled MODIS LST images showing the average values of clear-sky land surface temperature (C°) during an 8–day period; see further chapter 11.

By putting these operations in a loop, one can automate downloading, mosaicking and resampling of MODIS images for large areas. An example of such a script can be obtained from the book's homepage. Just to check that everything is OK with the maps, we can use the `GDALinfo` (a plot of a sample of images is shown in Fig. 4.4):

```
> GDALinfo("LST2006_01_01.LST_Day_1km.tif")

 rows        487
 columns     490
```

```
bands       1
origin.x    360641.5
origin.y    4669149
res.x       1000
res.y       1000
oblique.x   0
oblique.y   0
driver      GTiff
projection  +proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
file        LST2006_01_01.LST_Day_1km.tif
```

4.3 Summary points

The availability of remotely sensed data is everyday increasing. Every few years new generation satellites are launched that produce images at finer and finer detail, at shorter and shorter revisit time, and with richer and richer content (Fig. 4.2). For example, DEMs are now available from a number of sources. Detailed and accurate images of topography can now be ordered from remote sensing systems such as SPOT and ASTER; SPOT5 offers the High Resolution Stereoscopic (HRS) scanner, which can be used to produce DEMs at resolutions of up to 5 m. The cost of data is either free or dropping in price as technology advances. Likewise, access to remotely sensed data and various thematic maps is becoming less and less a technical problem — live image archives and ordering tools are now available. Multi-source layers are also increasingly compatible considering the coverage, scale/resolution, format and metadata description. The intergovernmental Group of Earth Observation is building the cyber-infrastructure called GEOSS[61] needed to enhance merging of various geoinformation layers to a single multi-thematic, multi-purpose data repository. From the time when satellite sensors were used for military purposes only, we now live in an era when anybody can load high resolution images of earth and detailed digital maps to their computer.

In the context of geostatistical mapping, the number of possible covariates is today increasing dramatically, so that there is practically no reason any more to study point-sampled variables in relation to its spatial coordinates only (Pebesma, 2006). External variables should be used wherever possible to improve geostatistical mapping. Majority of global spatial layers are today available at no cost at resolutions 1–10 km (see the world maps repository); MODIS imagery is freely available at resolution of 250 m; global elevation data is available at resolutions of 60–90 m; archive Landsat images (15–30 m) can be obtained for most of the world. All this proves that there is no reason not to use auxiliary covariates (i.e. methods such as regression-kriging) for geostatistical mapping.

In this chapter, I have specifically put an emphasis on MODIS products, mainly because the images are relatively easy to obtain for various parts of the world (or globally), and because MODIS images are valuable for monitoring ecosystem dynamics. From the aspect of ecology and nature conservation, the importance of MODIS images for environmental management and nature conservation is enormous (Lunetta et al., 2006). They can be obtained by anybody at any time and can be used as an independent source of information to quantify degradation of natural systems that is possibly not visible using local sources of information. MODIS EVI images can be used to show changes in land cover, deforestation, damage caused by global warming or even fine long-term succession processes. Because MODIS EVI is available from year the 2000, anybody can compare the current situation with the situation from a decade ago. The script presented can be obtained form the book's homepage and adopted to download any type of MODIS products available from the USGS Land Processes Distributed Active Archive Center.

The next generation remote sensing systems/surveys will certainly be multi-thematic and based on active surface-penetrating sensors. The future of automated mapping lays in using technologies such as LiDAR in combination with other optical, radar and hyperspectral sensors. This type of multi-thematic imagery will enable analysts to represent both surface and sub-surface properties of objects of interest, so that none of the important characteristics are overlooked. Multi-thematic scanners should also make the airborne surveys cheaper and hence suited for local and regional scale environmental studies.

[61] http://www.earthobservations.org

Further reading:

★ Doll, C.N.H., Muller, J.-P., Morley, J.G. 2007. Mapping regional economic activity from night-time light satellite imagery. Ecological Economics, 57(1): 75–92.

★ Hijmans, R.J., Cameron, S.E., Parra, J.L., Jones, P.G., Jarvis, A., 2005. Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology 25: 1965–1978.

★ Pebesma, E.J., 2006. The Role of External Variables and GIS Databases in Geostatistical Analysis. Transactions in GIS, 10(4): 615–632.

★ `http://www.fao.org/geonetwork/srv/en/main.home` — FAO's GeoNetwork server.

★ `http://www.geoportal.org/web/guest/geo_image_gallery` — ESA's GeoPortal gallery.

★ `http://geodata.grid.unep.ch/` — UNEP/GRID GEO DataPortal.

★ `http://neo.sci.gsfc.nasa.gov/` — NASA's Earth Observation (NEO) portal.

★ `http://glcf.umiacs.umd.edu/portal/geocover/` — Global Land Cover Facility (GLCF) geoportal operated by the University of Maryland.

★ `https://lpdaac.usgs.gov/lpdaac/` — Land Processes Distributed Active Archive Center (distributor of MODIS products).

5

First steps (meuse)

5.1 Introduction

This exercise introduces geostatistical tools that can be used to analyze various types of environmental data. It is not intended as a complete analysis of the example data set. Indeed some of the steps here can be questioned, expanded, compared, and improved. The emphasis is on seeing what R and some of its add-in packages can do in combination with an open source GIS such as SAGA GIS. The last section demonstrates how to export produced maps to Google Earth. This whole chapter is, in a way, a prerequisite to other exercises in the book.

We will use the meuse data set, which is a classical geostatistical data set used frequently by the creator of the gstat package to demonstrate various geostatistical analysis steps (Bivand et al., 2008, §8). The data set is documented in detail by Rikken and Van Rijn (1993), and Burrough and McDonnell (1998). It consists of 155 samples of top soil heavy metal concentrations (ppm), along with a number of soil and landscape variables. The samples were collected in a flood plain of the river Meuse, near the village Stein (Lat. 50° 58’ 16", Long. 5° 44’ 39"). Historic metal mining has caused the widespread dispersal of lead, zinc, copper and cadmium in the alluvial soil. The pollutants may constrain the land use in these areas, so detailed maps are required that identify zones with high concentrations. Our specific objective will be to generate a map of a heavy metal (zinc) in soil, and a map of soil liming requirement (binary variable) using point observations, and a range of auxiliary maps.

Upon completion of this exercise, you will be able to plot and fit variograms, examine correlation between various variables, run spatial predictions using the combination of continuous and categorical predictors and visualize results in external GIS packages/browsers (SAGA GIS, Google Earth). If you are new to R syntax, you should consider first studying some of the introductory books (listed in the section 3.4.2).

5.2 Data import and exploration

Download the attached meuse.R script from the book’s homepage and open it in Tinn-R. First, open a new R session and change the working directory to where all your data sets will be located (C:/meuse/). This directory will be empty at the beginning, but you will soon be able to see data sets that you will load, generate and/or export. Now you can run the script line by line. Feel free to experiment with the code and extend it as needed. Make notes if you experience any problems or if you are not able to perform some operation.

Before you start processing the data, you will need to load the following packages:

```
> library(maptools)
> library(gstat)
> library(rgdal)
> library(lattice)
> library(RSAGA)
> library(geoR)
> library(spatstat)
```

You can get a list of methods in each package with the `help` method, e.g.:

```
> help(package="maptools")
```

The `meuse` data set is in fact available in the installation directory of the `gstat` package. You can load the field observations by typing:

```
> data(meuse)
> str(meuse)

  'data.frame':        155 obs. of  14 variables:
   $ x      : num  181072 181025 181165 181298 181307 ...
   $ y      : num  333611 333558 333537 333484 333330 ...
   $ cadmium: num  11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
   $ copper : num  85 81 68 81 48 61 31 29 37 24 ...
   $ lead   : num  299 277 199 116 117 137 132 150 133 80 ...
   $ zinc   : num  1022 1141  640  257  269 ...
   $ elev   : num  7.91 6.98 7.80 7.66 7.48 ...
   $ dist   : num  0.00136 0.01222 0.10303 0.19009 0.27709 ...
   $ om     : num  13.6 14 13 8 8.7 7.8 9.2 9.5 10.6 6.3 ...
   $ ffreq  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
   $ soil   : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
   $ lime   : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
   $ landuse: Factor w/ 15 levels "Aa","Ab","Ag",..: 4 4 4 11 4 11 4 2 2 15 ...
   $ dist.m : num  50 30 150 270 380 470 240 120 240 420 ...
```

which shows a table with 155 observations of 14 variables. To get a complete description of this data set, type:

```
> ?meuse

  Help for 'meuse' is shown in the browser
```

which will open your default web-browser and show the Html help page for this data set. Here you can also find what the abbreviated names for the variables mean. We will focus on mapping the following two variables: `zinc` — topsoil zinc concentration in ppm; and `lime` — the logical variable indicating whether the soil needs liming or not.

Now we can start to visually explore the data set. For example, we can visualize the target variable with a histogram:

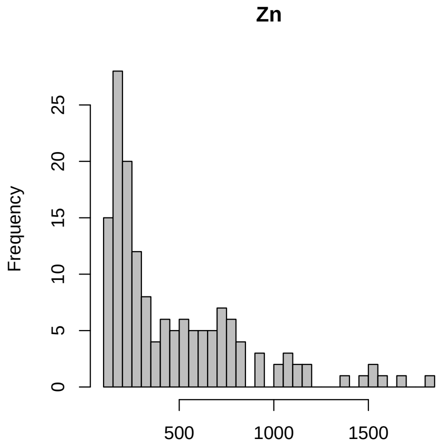


Fig. 5.1: Histogram plot for zinc (meuse data set).

```
> hist(meuse$zinc, breaks=25, col="grey")
```

which shows that the target variable is skewed towards lower values (Fig. 5.1), and it needs to be transformed before we can run any linear interpolation.

To be able to use spatial operations in R e.g. from the `gstat` package, we must convert the imported table into a `SpatialPointDataFrame`, a point map (with attributes), using the `coordinates` method:

```
# 'attach coordinates' - convert table to a point map:
> coordinates(meuse) <- ~ x+y
> str(meuse)

  Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
    ..@ data       :'data.frame':        155 obs. of  12 variables:
    .. ..$ cadmium: num [1:155] 11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
    .. ..$ copper : num [1:155] 85 81 68 81 48 61 31 29 37 24 ...
  .. ..$ lead   : num [1:155] 299 277 199 116 117 137 132 150 133 80 ...
  .. ..$ zinc   : num [1:155] 1022 1141  640  257  269 ...
  .. ..$ elev   : num [1:155] 7.91 6.98 7.80 7.66 7.48 ...
  .. ..$ dist   : num [1:155] 0.00136 0.01222 0.10303 0.19009 0.27709 ...
  .. ..$ om     : num [1:155] 13.6 14 13 8 8.7 7.8 9.2 9.5 10.6 6.3 ...
  .. ..$ ffreq  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ soil   : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
  .. ..$ lime   : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
  .. ..$ landuse: Factor w/ 15 levels "Aa","Ab","Ag",..: 4 4 4 11 4 11 4 2 2 15 ...
  .. ..$ dist.m : num [1:155] 50 30 150 270 380 470 240 120 240 420 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:155, 1:2] 181072 181025 181165 181298 181307 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "x" "y"
  ..@ bbox       : num [1:2, 1:2] 178605 329714 181390 333611
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

Note that the structure is now more complicated, with a nested structure and 5 '*slots*'[1] (Bivand et al., 2008, §2):

(1.) `@data` contains the actual data in a table format (a copy of the original dataframe minus the coordinates);

(2.) `@coords.nrs` has the coordinate dimensions;

(3.) `@coords` contains coordinates of each element (point);

(4.) `@bbox` stands for '*bounding box*' — this was automatically estimated by `sp`;

(5.) `@proj4string` contains the definition of projection system following the proj4[2] format.

The projection and coordinate system are at first unknown (listed as `NA` meaning 'not applicable'). Coordinates are just numbers as far as it is concerned. We know from the data set producers that this map is in the so-called *"Rijksdriehoek"* or RDH (Dutch triangulation), which is extensively documented[3]. This is a:

- stereographic projection (parameter `+proj`);
- on the Bessel ellipsoid (parameter `+ellps`);
- with a fixed origin (parameters `+lat_0` and `+lon_0`);
- scale factor at the tangency point (parameter `+k`);
- the coordinate system has a false origin (parameters `+x_0` and `+y_0`);
- the center of the ellipsoid is displaced with respect to the standard WGS84 ellipsoid (parameter `+towgs84`, with three distances, three angles, and one scale factor)[4];

It is possible to specify all this information with the `CRS` method; however, it can be done more simply if the datum is included in the European Petroleum Survey Group (EPSG) database[5], now maintained by the International Association of Oil & Gas producers (OGP). This database is included as text file (`epsg`) in the `rgdal` package, in the subdirectory `library/rgdal/proj` in the R installation folder. Referring to the EPSG registry[6], we find the following entry:

[1] This is the S4 objects vocabulary. *Slots* are components of more complex objects.

[2] `http://trac.osgeo.org/proj/`

[3] `http://www.rdnap.nl`

[4] The so-called seven datum transformation parameters (translation + rotation + scaling); also known as the *Bursa Wolf* method.

[5] `http://www.epsg-registry.org/`

[6] `http://spatialreference.org/ref/epsg/28992/`

```
  # Amersfoort / RD New <28992> +proj=sterea +lat_0=52.15616055555555
  +lon_0=5.38763888888889 +k=0.999908 +x_0=155000 +y_0=463000 +ellps=bessel
  +towgs84=565.237,50.0087,465.658,-0.406857,0.350733,-1.87035,4.0812
  +units=m +no_defs <>
```

This shows that the Amersfoort / RD New system is EPSG reference 28992. Note that some older installations of GDAL do not carry the seven-transformation parameters that define the geodetic datum! Hence, you will need to add these parameters manually to your library/rgdal/proj/epsg file. Once you have set the correct parameters in the system, you can add the projection information to this data set using the CRS method:

```
> proj4string(meuse) <- CRS("+init=epsg:28992")
> meuse@proj4string

  CRS arguments:
   +init=epsg:28992 +proj=sterea +lat_0=52.15616055555555
   +lon_0=5.38763888888889 +k=0.9999079 +x_0=155000 +y_0=463000 +ellps=bessel
   +towgs84=565.237,50.0087,465.658,-0.406857,0.350733,-1.87035,4.0812
   +units=m +no_defs
```

so now the correct projection information is included in the proj4string slot and we will be able to transform this spatial layer to geographic coordinates, and then export and visualize further in Google Earth.

Once we have converted the table to a point map we can proceed with spatial exploration data analysis, e.g. we can simply plot the target variable in relation to sampling locations. A common plotting scheme used to display the distribution of values is the bubble method. In addition, we can import also a map of the river, and then display it together with the values of zinc (Bivand et al., 2008):

```
# load river (lines):
> data(meuse.riv)
# convert to a polygon map:
> tmp <- list(Polygons(list(Polygon(meuse.riv)), "meuse.riv"))
> meuse.riv <- SpatialPolygons(tmp)
> class(meuse.riv)

  [1] "SpatialPolygons"
  attr(,"package")
  [1] "sp"


> proj4string(meuse.riv) <- CRS("+init=epsg:28992")
# plot together points and river:
> bubble(meuse, "zinc", scales=list(draw=T), col="black", pch=1, maxsize=1.5,
+   sp.layout=list("sp.polygons", meuse.riv, col="grey"))
```

which will produce the plot shown in Fig. 5.2, left[7]. Alternatively, you can also export the meuse data set to ESRI Shapefile format:

```
> writeOGR(meuse, ".", "meuse", "ESRI Shapefile")
```

which will generate four files in your working directory: meuse.shp (geometry), meuse.shx (auxiliary file), meuse.dbf (table with attributes), and meuse.prj (coordinate system). This shapefile you can now open in SAGA GIS and display using the same principle as with the bubble method (Fig. 5.2, right). Next, we import the gridded maps (40 m resolution). We will load them from the web repository[8]:

```
# download the gridded maps:
> setInternet2(use=TRUE)  # you need to login on the book's homepage first!
> download.file("http://spatial-analyst.net/book/system/files/meuse.zip",
+      destfile=paste(getwd(), "meuse.zip", sep="/"))
> grid.list <- c("ahn.asc", "dist.asc", "ffreq.asc", "soil.asc")
```

[7] See also http://r-spatial.sourceforge.net/gallery/ for a gallery of plots using meuse data set.

[8] This has some extra layers compared to the existing meusegrid data set that comes with the sp package.

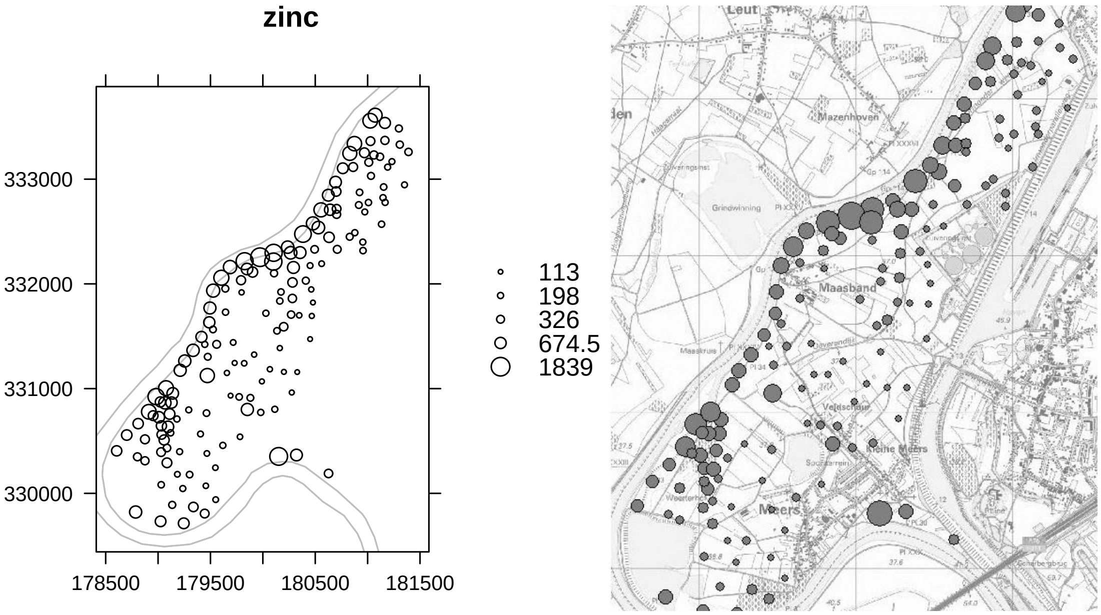


Fig. 5.2: Meuse data set and values of zinc (ppm): visualized in R (left), and in SAGA GIS (right).

```
# unzip the maps in a loop:
> for(j in grid.list){
>   fname <- zip.file.extract(file=j, zipname="meuse.zip")
>   file.copy(fname, paste("./", j, sep=""), overwrite=TRUE)
> }
```

These are the explanatory variables that we will use to improve spatial prediction of the two target variables:

(1.) `ahn` — digital elevation model (in cm) obtained from the LiDAR survey of the Netherlands[9];

(2.) `dist` — distance to river Meuse (in metres).

(3.) `ffreq` — flooding frequency classes: (1) high flooding frequency, (2) medium flooding frequency, (3) no flooding;

(4.) `soil` — map showing distribution of soil types, following the Dutch classification system: (1) Rd10A, (2) Rd90C-VIII, (3) Rd10C (de Fries et al., 2003);

In addition, we can also unzip the 2 m topomap that we can use as the background for displays (Fig. 5.2, right):

```
# the 2 m topomap:
> fname <- zip.file.extract(file="topomap2m.tif", zipname="meuse.zip")
> file.copy(fname, "./topomap2m.tif", overwrite=TRUE)
```

We can load the grids to R, also by using a loop operation:

```
> meuse.grid <- readGDAL(grid.list[1])

  ahn.asc has GDAL driver AAIGrid
  and has 104 rows and 78 columns
```

[9] http://www.ahn.nl

```
# fix the layer name:
> names(meuse.grid)[1] <- sub(".asc", "", grid.list[1])
> for(i in grid.list[-1]) {
>   meuse.grid@data[sub(".asc", "", i[1])] <- readGDAL(paste(i))$band1
> }
```

```
dist.asc has GDAL driver AAIGrid
and has 104 rows and 78 columns
ffreq.asc has GDAL driver AAIGrid
and has 104 rows and 78 columns
soil.asc has GDAL driver AAIGrid
and has 104 rows and 78 columns
```

```
# set the correct coordinate system:
> proj4string(meuse.grid) <- CRS("+init=epsg:28992")
```

Note that two of the four predictors imported (`ffreq` and `soil`) are categorical variables. However they are coded in the ArcInfo ASCII file as integer numbers, which R does not recognize automatically. We need to tell R that these are categories:

```
> meuse.grid$ffreq <- as.factor(meuse.grid$ffreq)
> table(meuse.grid$ffreq)
```

```
   1    2    3
 779 1335  989
```

```
> meuse.grid$soil <- as.factor(meuse.grid$soil)
> table(meuse.grid$soil)
```

```
   1    2    3
1665 1084  354
```

If you examine at the structure of the `meuse.grid` object, you will notice that it basically has a similar structure to a `SpatialPointsDataFrame`, except this is an object with a grid topology:

```
Formal class 'SpatialGridDataFrame' [package "sp"] with 6 slots
  ..@ data       :'data.frame': 8112 obs. of  4 variables:
  .. ..$ ahn  : int [1:8112] NA NA NA NA NA NA NA NA NA NA ...
  .. ..$ dist : num [1:8112] NA NA NA NA NA NA NA NA NA NA ...
  .. ..$ ffreq: Factor w/ 3 levels "1","2","3": NA NA NA NA NA NA NA NA NA NA ...
  .. ..$ soil : Factor w/ 3 levels "1","2","3": NA NA NA NA NA NA NA NA NA NA ...
  ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
  .. .. ..@ cellcentre.offset: Named num [1:2] 178460 329620
  .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
  .. .. ..@ cellsize         : num [1:2] 40 40
  .. .. ..@ cells.dim        : int [1:2] 78 104
  ..@ grid.index : int(0)
  ..@ coords     : num [1:2, 1:2] 178460 181540 329620 333740
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "x" "y"
  ..@ bbox       : num [1:2, 1:2] 178440 329600 181560 333760
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr " +init=epsg:28992 +proj=sterea +lat_0=52.15616055
  +lon_0=5.38763888888889 +k=0.999908 +x_0=155000 +y_0=463000 +ellps=bess"|
   __truncated__
```

Many of the grid nodes are unavailable (`NA` sign), so that it seems that the layers carry no information. To check that everything is ok, we can plot the four gridded maps together (Fig. 5.3):

```
ffreq.plt <- spplot(meuse.grid["ffreq"],
+    col.regions=grey(runif(length(levels(meuse.grid$ffreq)))),
+    main="Flooding frequency classes")
dist.plt <- spplot(meuse.grid["dist"],
+    col.regions=grey(rev(seq(0,1,0.025))),
+    main="Distance to river")
soil.plt <- spplot(meuse.grid["soil"],
+    col.regions=grey(runif(length(levels(meuse.grid$ffreq)))),
+    main="Soil type classes")
ahn.plt <- spplot(meuse.grid["ahn"],
+    col.regions=grey(rev(seq(0,1,0.025))),
+    main="Elevation (cm)")
> print(ffreq.plt, split=c(1, 1, 4, 1), more=TRUE)
> print(dist.plt, split=c(2, 1, 4, 1), more=TRUE)
> print(ahn.plt, split=c(3, 1, 4, 1), more=TRUE)
> print(soil.plt, split=c(4, 1, 4, 1), more=TRUE)
```

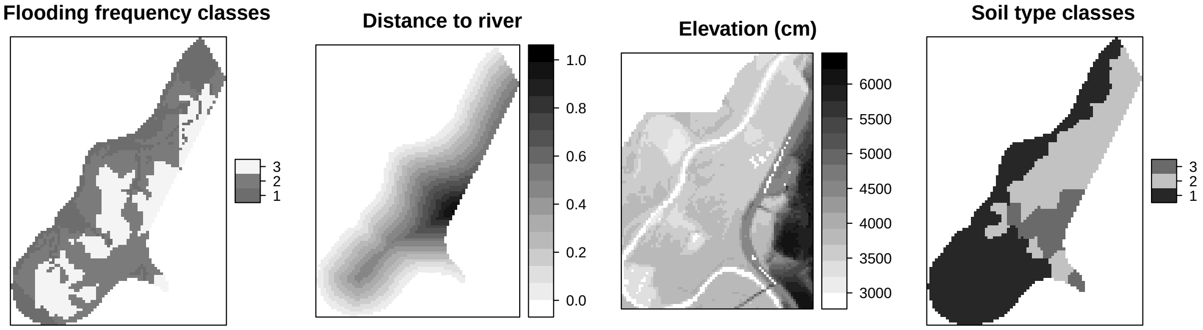


Fig. 5.3: Meuse auxiliary predictors.

5.2.1 Exploratory data analysis: sampling design

As noted in the preface, no geostatistician can promise high quality products without quality input point samples. To assess how representative and consistent the input data are, we can run some basic exploratory analysis to look at the point geometry and how well the environmental features are represented. We can start with point pattern analysis as implemented in the spatstat package, e.g. to determine the average spacing between the points (Baddeley, 2008):

```
# coerce to a ppp object:
> mg_owin <- as.owin(meuse.grid["dist"])
> meuse.ppp <- ppp(x=coordinates(meuse)[,1], y=coordinates(meuse)[,2],
+       marks=meuse$zinc, window=mg_owin)
# plot(meuse.ppp)
> summary(dist.points)

     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    43.93   77.88  107.40  111.70  137.70  353.00
```

which shows that the means shortest distance is 111 m. The following two questions are relevant for further analysis: (1) are the sampling locations distributed independently and uniformly over the area of interest (i.e. is there a significant clustering of locations)? (2) is the environmental feature space well represented? To answer the first question we can test the sampling design for *Complete Spatial Randomness* (CSR). CRS assumes that there are no regions in the study area where events are more likely to occur, and that the presence of a given event does not modify the probability of other events appearing nearby (Bivand et al., 2008).

The compatibility of a sampling design with CSR can be assessed by plotting the empirical function against the theoretical expectation (Bivand et al., 2008, p.160–163):

```
> env.meuse <- envelope(meuse.ppp, fun=Gest)

  Generating 99 simulations of CSR ...
  1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
  ... 91, 92, 93, 94, 95, 96, 97, 98, 99.


> plot(env.meuse, lwd=list(3,1,1,1), main="CSR test (meuse)")
```

which will run 100 simulations using the given point pattern and derive confidence bands for a CSR using the so called *G* function — this measures the distribution of the distances from an arbitrary event to its nearest event (Diggle, 2003). The plot of distributions, actual versus expected CSR (Fig. 5.4), shows that the sampling design is somewhat clustered at shorter distances up to 75 m. Although the line of the observed distribution is in >80% of distance range outside the confidence bands (*envelopes*), we can say that the sampling plan is, in general, representative relative to geographical space.

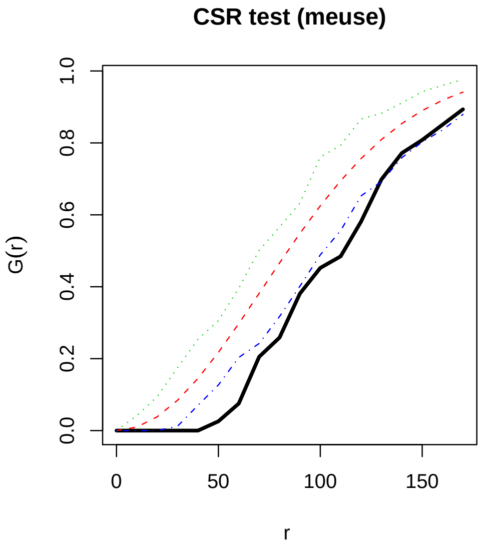


Fig. 5.4: Comparison of the confidence bands for the *G* function (Complete Spatial Randomness) and the actual observed distribution (bold line). Derived using the `envelope` method in spatstat.

Next we look at the feature space coverage. For example, we can check whether there is a significant difference in the distribution of values at sampling locations and in the whole area of interest. To run this type of analysis we need to overlay sampling points and predictors to create an object[10] with just the sample points, values of the target variable and of the feature-space predictors. We use the `overlay` method of the sp package to extract the values from the grid maps:

```
> meuse.ov <- overlay(meuse.grid, meuse)
> meuse.ov@data <- cbind(meuse.ov@data, meuse[c("zinc", "lime")]@data)
> str(meuse.ov@data)

  'data.frame':   155 obs. of  6 variables:
   $ ahn  : int  3214 3402 3277 3563 3406 3355 3428 3476 3522 3525 ...
   $ dist : num  0.00136 0.01222 0.10303 0.19009 0.27709 ...
   $ ffreq: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
   $ soil : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
   $ zinc : num  1022 1141 640 257 269 ...
   $ lime : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
```

Now we can run some explanatory analyzes that focus on the feature space. First, we can visually compare the histograms of the maps with the histograms of values at point locations, e.g. by using a back to back histogram[11]:

```
> library(Hmisc)
> options(digits=1)
> dist.histbb <- histbackback(meuse.ov$dist, meuse.grid$dist, prob=TRUE,
+      xlab=c("sample","map"), main="Distance (m)")
> barplot(-dist.histbb$left, col="dark grey" , horiz=TRUE, space=0, add=TRUE,
+      axes=FALSE)
> barplot(dist.histbb$right, col="grey", horiz=TRUE, space=0, add=TRUE, axes=FALSE)
> ahn.histbb <- histbackback(meuse.ov$ahn, meuse.grid$ahn, prob=TRUE,
+      xlab=c("sample","map"), main="AHN (cm)")
```

[10] Often refer to as the "*regression matrix*".

[11] This requires installation of the package Hmisc.

```
> barplot(-ahn.histbb$left, col="dark grey" , horiz=TRUE, space=0, add=TRUE,
+      axes=FALSE)
> barplot(ahn.histbb$right, col="grey", horiz=TRUE, space=0, add=TRUE, axes=FALSE)
> par(mfrow=c(1,2))
> print(dist.histbb, add=TRUE)
> print(ahn.histbb, add=FALSE)
> dev.off()
> options(digits=3)
```

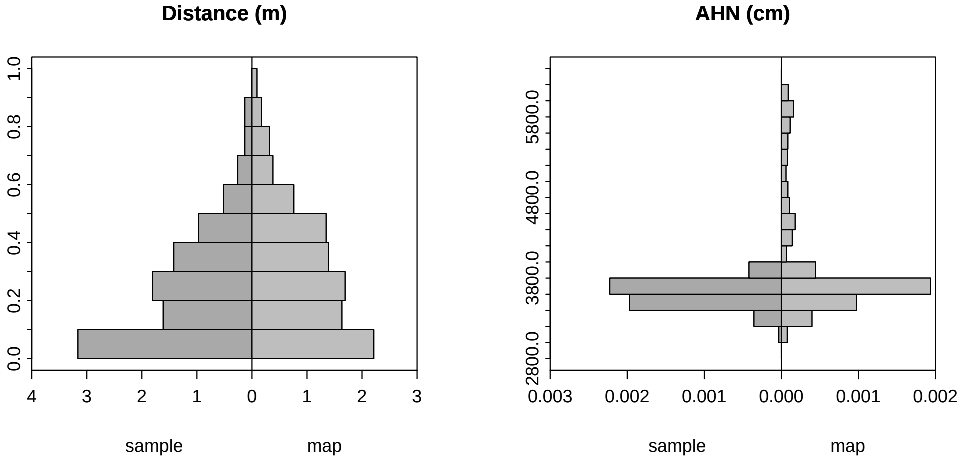


Fig. 5.5: Histogram for sampled values of `dist` and `ahn` (155 locations) versus the histogram of the raster map (all raster nodes). Produced using the `histbackback` method.

This will produce two histograms next to each other so that we can visually compare how well the samples represent the original feature space of the raster maps (Fig. 5.5). In the case of the `points` data set, we can see that the samples are misrepresenting higher elevations, but distances from the river are well represented. We can actually test if the histograms of sampled variables are significantly different from the histograms of original raster maps e.g. by using a non-parametric test such as the Kolmogorov-Smirnov test:

```
> ks.test(dist.histbb$left, dist.histbb$right)

         Two-sample Kolmogorov-Smirnov test

  data:  dist.histbb$left and dist.histbb$right
  D = 0.2, p-value = 0.9945
  alternative hypothesis: two-sided

> ks.test(ahn.histbb$left, ahn.histbb$right)

         Two-sample Kolmogorov-Smirnov test

  data:  ahn.histbb$left and ahn.histbb$right
  D = 0.7, p-value = 0.0001673
  alternative hypothesis: two-sided

  Warning message:
  In ks.test(ahn.histbb$left, ahn.histbb$right) :
    cannot compute correct p-values with ties
```

which shows that the first two histograms (`dist`) do not differ much, but the second two (`ahn`) have significantly different distributions (D=0.7, p-value=0.0001673). Another test that you might do to compare the histograms is to run the correlation test[12]:

[12] Also known as the test of no correlation because it computes t-value for correlation coefficient being equal to zero.

```
> cor.test(ahn.histbb$left, ahn.histbb$right)
```

In the step of geographic analysis of the sampling design we will assess whether the sampling density within different soil mapping units (soil) is consistent. First, we look at how many points fall into each zone:

```
> summary(meuse.ov$soil)

  1  2  3
 97 46 12
```

then we need to derive the observed inspection density using:

```
# observed:
> inspdens.obs <- summary(meuse.ov$soil)[1:length(levels(meuse.ov$soil))]/
+      (summary(meuse.grid$soil)[1:length(levels(meuse.grid$soil))]
+      *meuse.grid@grid@cellsize[[1]]^2)
# expected:
> inspdens.exp <- rep(length(meuse.ov$soil)/
+      (length(meuse.grid$soil[!is.na(meuse.grid$soil)])
+      *meuse.grid@grid@cellsize[[1]]^2), length(levels(meuse.ov$soil)))
# inspection density in no./ha:
> inspdens.obs*10000

     1     2     3
 0.364 0.265 0.212


> inspdens.exp*10000

 [1] 0.312 0.312 0.312
```

which can also be compared by using the Kolmogorov-Smirnov test:

```
> ks.test(inspdens.obs, inspdens.exp)


         Two-sample Kolmogorov-Smirnov test

 data:  inspdens.obs and inspdens.exp
 D = 0.667, p-value = 0.5176
 alternative hypothesis: two-sided

 Warning message:
 In ks.test(inspdens.obs, inspdens.exp) :
   cannot compute correct p-values with ties
```

In this case, we see that inspection density is also significantly inconsistent considering the map of soil, which is not by chance (*p*-value=0.5176). We could also run a similar analysis for land cover types or any other factor-type predictors.

So in summary, we can conclude for the meuse sampling design that:

- the average distance to the nearest neighbor is 111 m and the size of the area is 496 ha;
- the sampling intensity is 3 points per 10 ha, which corresponds to a grid cell size of about 15 m (Hengl, 2006);
- the sampling density varies in geographical space — sampling is significantly clustered for smaller distance (<75 m);
- the sampling is unrepresentative considering the maps of ahn and soil — higher elevations and soil class 3 are significantly under-sampled;

These results do not mean that this data set is unsuitable for generating maps, but they do indicate that it has some limitations considering representativeness, independency and consistency requirements.

5.3 Zinc concentrations

5.3.1 Regression modeling

The main objective of regression-kriging analysis is to build a regression model by using the explanatory gridded maps. We have previously estimated values of explanatory maps and target variables in the same table (overlay operation), so we can start by visually exploring the relation between the target variable and the *continuous* predictors e.g. by using a smoothed `scatterplot` (Fig. 5.6):

```
> par(mfrow = c(1, 2))
> scatter.smooth(meuse.ov$dist, meuse.ov$zinc, span=18/19,
+     col="grey", xlab="Distance to river (log)", ylab="Zinc (ppm)")
> scatter.smooth(meuse.ov$ahn, meuse.ov$zinc, span=18/19,
+     col="grey", xlab="Elevation (cm)", ylab="Zinc (ppm)")
```

which shows that the values of zinc decrease as the distance from (water) streams and elevation increases. This supports our knowledge about the area — the majority of heavy metals has been originated from fresh water deposition. The relation seems to be particulary clear, but it appears to be non-linear as the fitted lines are curved.

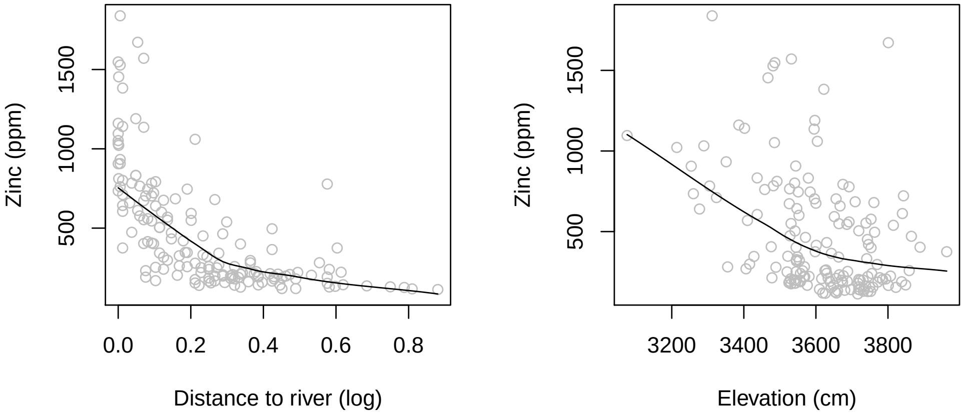


Fig. 5.6: Scatterplots showing the relation between zinc and distance from river, and elevation.

Another useful analysis relevant for the success of regression modeling is to look at the multicolinearity of predictors. Some predictors show the same feature, i.e. they are not independent. For example, `dist.asc` and `ahn.asc` maps are correlated:

```
> pairs(zinc ~ ahn+dist, meuse.ov)
> cor(meuse.grid$ahn, meuse.grid$dist, use="complete.obs")

  [1] 0.294
```

To visualize the relationship between the target variable and the *classified* predictors we used a grouped `boxplot`; this also allows us to count the samples in each class (Fig. 5.7):

```
> par(mfrow=c(1,2))
> boxplot(log1p(meuse.ov$zinc) ~ meuse.ov$soil,
+      col=grey(runif(length(levels(meuse.ov$soil)))),
+      xlab="Soil type classes", ylab="Zinc (ppm)")
> boxplot(log1p(meuse.ov$zinc) ~ meuse.ov$ffreq,
+      col=grey(runif(length(levels(meuse.ov$soil)))),
+      xlab="Flooding frequency classes", ylab="Zinc (ppm)")
> dev.off()
> boxplot(log1p(meuse.ov$zinc) ~ meuse.ov$soil, plot=FALSE)$n
```

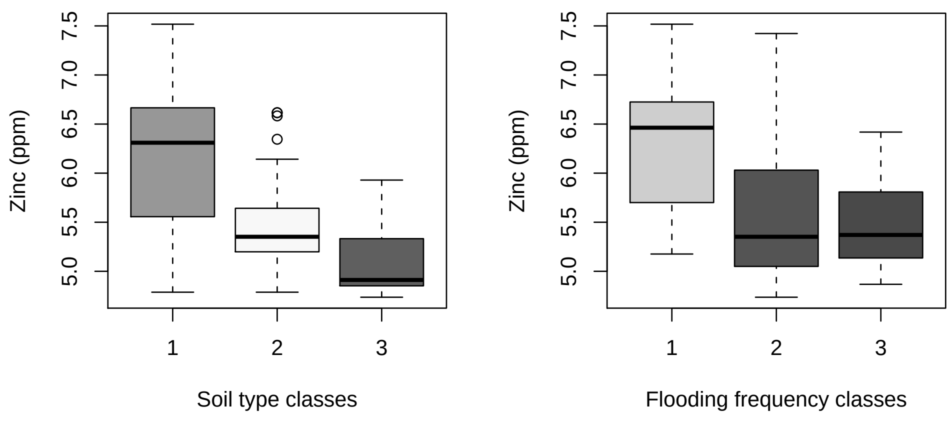


Fig. 5.7: Boxplots showing differences in zinc values (log-scale) between various soil and flooding frequency mapping units.

```
[1] 97 46 12
```

```
> boxplot(log1p(meuse.ov$zinc) ~ meuse.ov$ffreq, plot=FALSE)$n
```

```
[1] 73 53 29
```

which indicates that soil class "1" carries significantly higher zinc content than the remaining two classes. Note that there are only 12 field samples in the soil class "3", but still enough[13] to fit a regression model.

Now that we have some idea of the qualitative relation between the predictors and target variable, we proceed with fitting a regression model. We will first try to explain variation in the target variable by using all possible physical predictors — continuous and categorical. Because the target variable is heavily skewed towards lower values, we will use its transform to fit a linear model:

```
> lm.zinc <- lm(log1p(zinc) ~ ahn+dist+ffreq+soil, meuse.ov)
> summary(lm.zinc)
```

```
Call:
lm(formula = log1p(zinc) ~ ahn + dist + ffreq + soil, data = meuse.ov)

Residuals:
    Min      1Q  Median      3Q     Max
-0.8421 -0.2794  0.0036  0.2469  1.3669

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.114955   1.121854    7.23  2.3e-11 ***
ahn         -0.000402   0.000320   -1.26  0.21069
dist        -1.796855   0.257795   -6.97  9.7e-11 ***
ffreq2      -0.434477   0.092897   -4.68  6.5e-06 ***
ffreq3      -0.415166   0.121071   -3.43  0.00078 ***
soil2       -0.368315   0.094768   -3.89  0.00015 ***
soil3       -0.097237   0.163533   -0.59  0.55302
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.43 on 148 degrees of freedom
Multiple R-squared: 0.658,      Adjusted R-squared: 0.644
F-statistic: 47.4 on 6 and 148 DF,  p-value: <2e-16
```

[13] By a rule of thumb, we should have at least 5 observations per mapping unit to be able to fit a reliable model.

The lm method has automatically converted factor-variables into indicator (*dummy*) variables. The summary statistics show that our predictors are significant in explaining the variation in log1p(zinc). However, not all of them are equally significant; some could probably be left out. We have previously demonstrated that some predictors are cross-correlated (e.g. dist and ahn). To account for these problems, we will do the following: first, we will generate indicator maps to represent all classes of interest:

```
> meuse.grid$soil1 <- ifelse(meuse.grid$soil=="1", 1, 0)
> meuse.grid$soil2 <- ifelse(meuse.grid$soil=="2", 1, 0)
> meuse.grid$soil3 <- ifelse(meuse.grid$soil=="3", 1, 0)
> meuse.grid$ffreq1 <- ifelse(meuse.grid$ffreq=="1", 1, 0)
> meuse.grid$ffreq2 <- ifelse(meuse.grid$ffreq=="2", 1, 0)
> meuse.grid$ffreq3 <- ifelse(meuse.grid$ffreq=="3", 1, 0)
```

so that we can convert all grids to principal components to reduce their multi-dimensionality[14]:

```
> pc.predmaps <- prcomp( ~ ahn+dist+soil1+soil2+soil3+ffreq1+ffreq2+ffreq3,
+     scale=TRUE, meuse.grid)
> biplot(pc.predmaps, xlabs=rep(".", length(pc.predmaps$x[,1])), arrow.len=0.1,
+     xlab="First component", ylab="Second component")
```

After the principal component analysis, we need to convert the derived PCs (10) to grids, since they have lost their spatial reference. This will take few steps:

```
> pc.comps <- as.data.frame(pc.predmaps$x)
# insert grid index:
> meuse.grid$nrs <- seq(1, length(meuse.grid@data[[1]]))
> meuse.grid.pnt <- as(meuse.grid["nrs"], "SpatialPointsDataFrame")
# mask NA grid nodes:
> maskpoints <- as.numeric(attr(pc.predmaps$x, "dimnames")[[1]])
# attach coordinates:
> pc.comps$X <- meuse.grid.pnt@coords[maskpoints, 1]
> pc.comps$Y <- meuse.grid.pnt@coords[maskpoints, 2]
> coordinates(pc.comps) <- ~ X + Y
# convert to a grid:
> gridded(pc.comps) <- TRUE
> pc.comps <- as(pc.comps, "SpatialGridDataFrame")
> proj4string(pc.comps) <- meuse.grid@proj4string
> names(pc.comps)

 [1] "PC1" "PC2" "PC3" "PC4" "PC5" "PC6" "PC7" "PC8"
```

overlay the points and PCs again, and re-fit a regression model:

```
> meuse.ov2 <- overlay(pc.comps, meuse)
> meuse.ov@data <- cbind(meuse.ov@data, meuse.ov2@data)
```

Because all predictors should now be independent, we can reduce their number by using step-wise regression:

```
> lm.zinc <- lm(log1p(zinc) ~ PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8, meuse.ov)
> step.zinc <- step(lm.zinc)
> summary(step.zinc)

 Call:
 lm(formula = log1p(zinc) ~ PC1 + PC2 + PC3 + PC4 + PC6, data=meuse.ov)

 Residuals:
       Min        1Q    Median        3Q       Max
 -0.833465 -0.282418  0.000112  0.261798  1.415499
```

[14] An important assumption of linear regression is that the predictors are mutually independent (Kutner et al., 2004).

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.6398     0.0384  146.94  < 2e-16 ***
PC1          -0.3535     0.0242  -14.59  < 2e-16 ***
PC2          -0.0645     0.0269   -2.40  0.01756 *
PC3          -0.0830     0.0312   -2.66  0.00869 **
PC4           0.0582     0.0387    1.50  0.13499
PC6          -0.2407     0.0617   -3.90  0.00014 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.426 on 149 degrees of freedom
Multiple R-squared: 0.661,        Adjusted R-squared: 0.65
F-statistic: 58.2 on 5 and 149 DF,  p-value: <2e-16
```

The resulting models shows that there are only two predictors that are highly significant, and four that are marginally significant, while four predictors can be removed from the list. You should also check the diagnostic plots for this regression model to see if the assumptions[15] of linear regression are met.

5.3.2 Variogram modeling

We proceed with modeling of the variogram, which will be later used to make predictions using universal kriging in gstat. Let us first compute the sample (experimental) variogram with the variogram method of the gstat package:

```
> zinc.svar <- variogram(log1p(zinc) ~ 1, meuse)
> plot(zinc.svar)
```

This shows that the semivariance reaches a definite sill at a distance of about 1000 m (Fig. 5.8, left). We can use the automatic fitting method[16] in gstat to fit a suitable variogram model. This method requires some initial parameters. We can set them using the following rule of thumb:

- Nugget is zero;
- Sill is the total (nonspatial) variance of the data set;
- Range is one-quarter of the diagonal of the bounding box.

In R, we can code this by using:

```
> zinc.vgm <- fit.variogram(zinc.svar, model=zinc.ivgm)
> zinc.vgm

  model psill range
1   Nug 0.000     0
2   Exp 0.714   449


> zinc.vgm.plt <- plot(zinc.svar, zinc.vgm, pch="+", pl=TRUE,
+     col="black", main="log1p(zinc)")
```

The idea behind using default values for initial variogram is that the process can be automated, without need to visually examine each variogram; although, for some variograms the automated fit may not converge to a reasonable solution (if at all). In this example, the fitting runs without a problem and you should get something like Fig. 5.8.

In order to fit the regression-kriging model, we actually need to fit the variogram for the residuals:

[15] Normally distributed, symmetric residuals around the regression line; no heteroscedascity, outliers or similar unwanted effects.

[16] The fit.variogram method uses weighted least-squares.

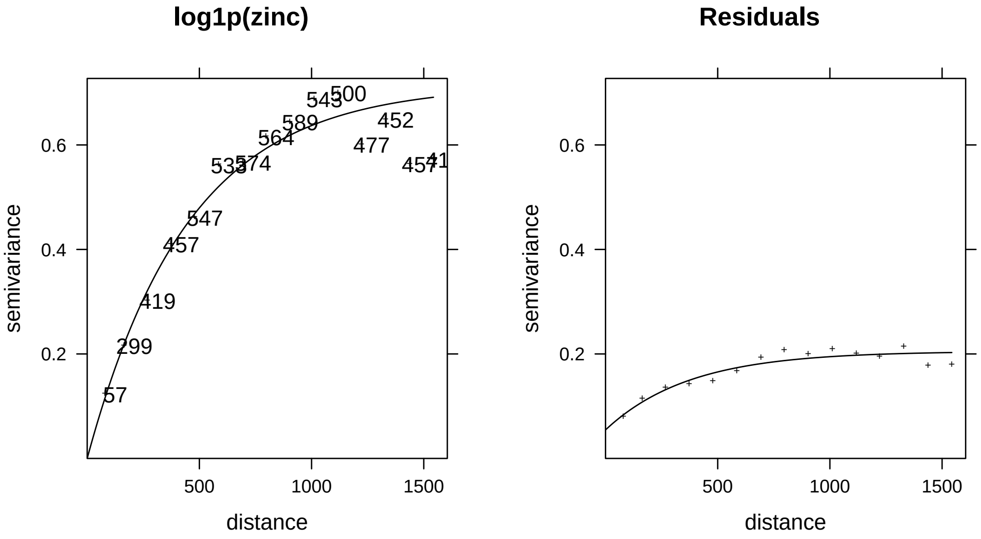


Fig. 5.8: Variogram for original variable, and regression residuals.

```
> zinc.rsvar <- variogram(residuals(step.zinc) ~ 1, meuse.ov)
> zinc.ivgm <- vgm(nugget=0, model="Exp",
+     range=sqrt(diff(meuse@bbox["x",])^2 + diff(meuse@bbox["y",])^2)/4,
+     psill=var(residuals(step.zinc)))
> zinc.rvgm <- fit.variogram(zinc.rsvar, model=zinc.ivgm)
> zinc.rvgm
```

```
  model  psill range
1   Nug 0.0546     0
2   Exp 0.1505   374
```

```
> zinc.rvgm.plt <- plot(zinc.rsvar, zinc.rvgm, pc="+", pl=FALSE,
+    col="black", main="Residuals")
# synchronize the two plots:
> zinc.rvgm.plt$x.limits <- zinc.vgm.plt$x.limits
> zinc.rvgm.plt$y.limits <- zinc.vgm.plt$y.limits
> print(zinc.vgm.plt, split=c(1,1,2,1), more=TRUE)
> print(zinc.rvgm.plt, split=c(2,1,2,1), more=FALSE)
```

which shows a somewhat different picture than in the case of the original variable (`zinc.vgm`): the sill parameter is now much smaller, as you can notice from the plot (Fig. 5.8, right). This is expected because that the regression model (§5.3.1) has already explained 65% of the variation in the target variable.

5.3.3 Spatial prediction of Zinc

Once we have fitted both the regression model (deterministic part of variation) and the variogram for residuals (stochastic, spatially-autocorrelated part of variation), we can proceed with regression-kriging. This method is implemented in the gstat's generic spatial prediction method called `krige`:

```
> zinc.rk <- krige(step.zinc$call$formula, meuse.ov, pc.comps, zinc.rvgm)
```

```
[using universal kriging]
```

```
# back-transform the values:
> zinc.rk$var1.rk <- expm1(zinc.rk$var1.pred)
```

where `step.zinc$call$formula` is the regression model estimated in the previous section:

```
> step.zinc$call$formula

 log1p(zinc) ~ PC1 + PC2 + PC3 + PC4 + PC6
```

where `zinc.rvgm` is the fitted residual variogram, and `expm1` is back-transformation function. For a comparison, we can make predictions at all locations of interest also using ordinary kriging:

```
> zinc.ok <- krige(log1p(zinc) ~ 1, meuse, meuse.grid["soil"], zinc.vgm)

 [using ordinary kriging]


> zinc.ok$var1.rk <- expm1(zinc.ok$var1.pred)
```

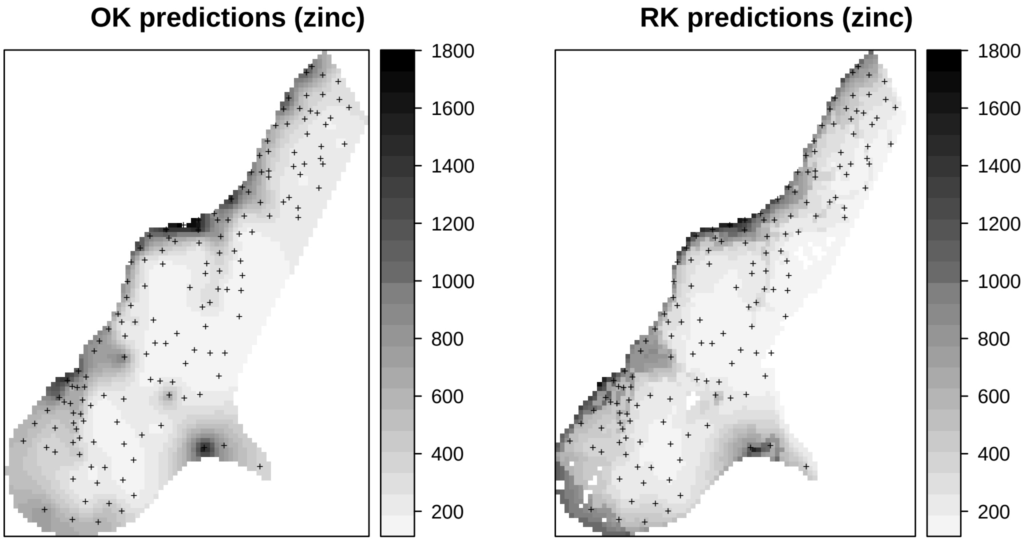


Fig. 5.9: Ordinary kriging vs regression-kriging: spatial prediction of zinc.

and compare the two maps side-by-side:

```
# display limits:
> at.zinc <- seq(min(meuse$zinc),max(meuse$zinc),sd(meuse$zinc)/5)
> zinc.ok.plt <- spplot(zinc.ok["var1.rk"],
+    col.regions=grey(rev(seq(0,0.97,1/length(at.zinc)))), at=at.zinc,
+    main="OK predictions (zinc)", sp.layout=list("sp.points",pch="+",
+    col="black", meuse))
> zinc.rk.plt <- spplot(zinc.rk["var1.rk"],
+    col.regions=grey(rev(seq(0,0.97,1/length(at.zinc)))), at=at.zinc,
+    main="RK predictions (zinc)", sp.layout=list("sp.points", pch="+",
+    col="black", meuse))
> print(zinc.ok.plt, split=c(1,1,2,1), more=T)
> print(zinc.rk.plt, split=c(2,1,2,1), more=F)
```

Visually, there are some clear differences between the two maps. The regression-kriging map shows much more local detail (the maps showing distance from river and soil and flooding frequency classes are reflected in the output map); the locations of hot-spots on both maps are, on the other hand, very similar. But visual comparison is not enough. Hence, we also would like to see which technique is more accurate. To achieve this, we use use the *leave-one-out cross validation method*, as implemented in the `krige.cv` method of `gstat` package (Pebesma, 2004). To run cross-validations, we simply use the built-in `krive.cv` method:

```
> cross.zinc.ok <- krige.cv(log1p(zinc) ~ 1, meuse.ov,
+     zinc.vgm, verbose=FALSE) # show no output
> cross.zinc.rk <- krige.cv(step.zinc$call$formula, meuse.ov,
+     zinc.rvgm, verbose=FALSE)
```

You will notice that the kriging system is solved once for each input data point. To evaluate the cross-validation, we can compare RMSE summaries (§1.4), and in particular the standard deviations of the errors (field `residual` of the cross-validation object). To estimate how much of variation has been explained by the two models, we can run:

```
# amount of variation explained by the models:
> 1-var(cross.zinc.ok$residual, na.rm=T)/var(log1p(meuse$zinc))

  [1] 0.701

> 1-var(cross.zinc.rk$residual, na.rm=T)/var(log1p(meuse$zinc))

  [1] 0.773
```

which shows that OK is not much worse than RK — RK is a *better* predictor, but the difference is only 7%. This is possibly because variables `dist` and `soil` are also spatially continuous, and because the samples are equally spread in geographic space. Indeed, if you look at Fig. 5.9 again, you will notice that the two maps do not differ much. Note also that amount of variation explained by RK geostatistical model is about 80%, which is satisfactory.

5.4 Liming requirements

5.4.1 Fitting a GLM

In the second part of this exercise, we will try to interpolate a categorical variable using the same regression-kriging model. This variable is not as simple as zinc, since it ranges from 0 to 1 i.e. it is a binomial variable. We need to respect that property of the data, and try to fit it using a GLM (Kutner et al., 2004):

$$\mathbb{E}(\mathbf{P_c}) = \mu = g^{-1}(\mathbf{q}\cdot\beta) \tag{5.4.1}$$

where E($\mathbf{P}$) is the expected probability of class c ($P_c \in [0,1]$), $\mathbf{q}\cdot\beta$ is the linear regression model, and g is the link function. Because the target variable is a binomial variable, we need to use the *logit* link function:

$$g(\mu) = \mu^{+} = \ln\left(\frac{\mu}{1-\mu}\right) \tag{5.4.2}$$

so the Eq.(5.4.1) becomes logistic regression (Kutner et al., 2004). How does this works in R? Instead of fitting a simple linear model (`lm`), we can use a more generic `glm` method with the `logit` link function (Fig. 5.10, left):

```
> glm.lime <- glm(lime ~ PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8, meuse.ov,
+      family=binomial(link="logit"))
> step.lime <- step(glm.lime)
# check if the predictions are within 0-1 range:
> summary(round(step.lime$fitted.values, 2))

     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.000   0.010   0.090   0.284   0.555   0.920
```

What you do not see from your R session is that the GLM model is fitted iteratively, i.e. using a more sophisticated approach than if we would simply fit a `lm` (e.g. using ordinary least square — no iterations). To learn more about the GLMs and how are they fitted and how to interpret the results see Kutner et al. (2004).

Next, we can predict the values[17] at all grid nodes using this model:

[17] Important: note that we focus on values in the transformed scale, i.e. logits.

```
> p.glm <- predict(glm.lime, newdata=pc.comps, type="link", se.fit=T)
> str(p.glm)
```

```
 List of 3
  $ fit           : Named num [1:3024] 2.85 2.30 2.25 1.83 2.77 ...
   ..- attr(*, "names")= chr [1:3024] "68" "144" "145" "146" ...
  $ se.fit        : Named num [1:3024] 1.071 0.813 0.834 0.729 1.028 ...
   ..- attr(*, "names")= chr [1:3024] "68" "144" "145" "146" ...
  $ residual.scale: num 1
```

which shows that the spatial structure of the object was lost. Obviously, we will not be able to display the results as a map until we convert it to a gridded data frame. This takes few steps:

```
# convert to a gridded layer:
> lime.glm <- as(pc.comps, "SpatialPointsDataFrame")
> lime.glm$lime <- p.glm$fit
> gridded(lime.glm) <- TRUE
> lime.glm <- as(lime.glm, "SpatialGridDataFrame")
> proj4string(lime.glm) <- meuse.grid@proj4string
```

5.4.2 Variogram fitting and final predictions

The remaining residuals we can interpolate using ordinary kriging. This is assuming that the residuals follow an approximately normal distribution. If the GLM we use is appropriate, this should indeed be the case. First, we estimate the variogram model:

```
> hist(residuals(step.lime), breaks=25, col="grey")
# residuals are normal;
> lime.ivgm <- vgm(nugget=0, model="Exp",
+    range=sqrt(diff(meuse@bbox["x",])^2 + diff(meuse@bbox["y", ])^2)/4,
+    psill=var(residuals(step.lime)))
> lime.rvgm <- fit.variogram(variogram(residuals(step.lime) ∼ 1, meuse.ov),
+    model=lime.ivgm)
```

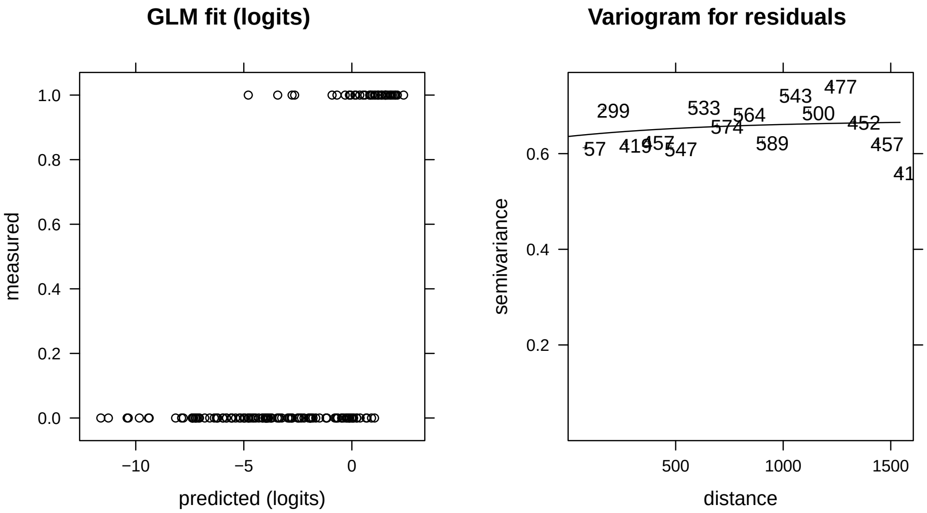


Fig. 5.10: Measured and predicted (GLM with binomial function) values for lime variable (left); variogram for the GLM residuals (right).

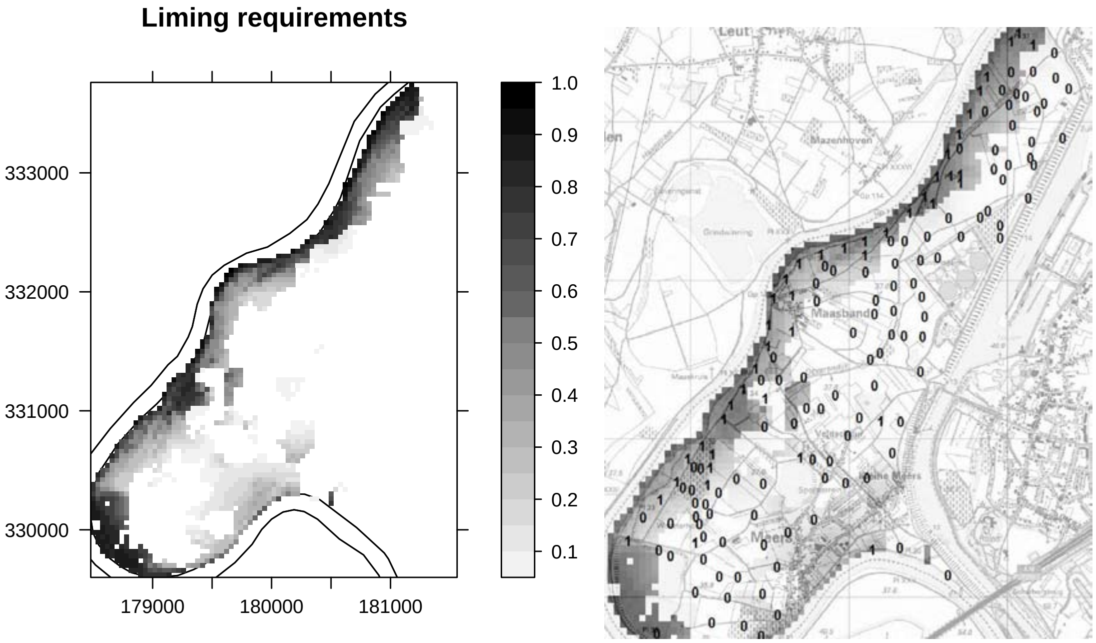


Fig. 5.11: Liming requirements predicted using regression-kriging; as shown in R (left) and in SAGA GIS (right).

which shows that the variogram is close to pure nugget effect (Fig. 5.10, right)[18]. We can still interpolate the residuals using ordinary kriging:

```
> lime.rk <- krige(residuals(step.lime) ~ 1, meuse.ov, pc.comps, lime.rvgm)

  [using ordinary kriging]
```

and then add back to the predicted regression part of the model:

```
> lime.rk$var1.rk <- lime.glm$lime + lime.rk$var1.pred
> lime.rk$var1.rko <- exp(lime.rk$var1.rk)/(1 + exp(lime.rk$var1.rk))
# write to a GIS format:
> write.asciigrid(lime.rk["var1.rko"], "lime_rk.asc", na.value=-1)
> lime.plt <- spplot(lime.rk["var1.rko"], scales=list(draw=T),
+     at=seq(0.05, 1, 0.05), col.regions=grey(rev(seq(0, 0.95, 0.05))),
+     main="Liming requirements", sp.layout=list("sp.polygons",
+     col="black", meuse.riv))
```

After you export the resulting map to SAGA GIS, a first step is to visually explore the maps to see how well the predicted values match the field observations (Fig. 5.11). Note that the map has problems predicting the right class at several isolated locations. To estimate the accuracy of predicting the right class, we can use:

```
> lime.ov <- overlay(lime.rk, meuse)
> lime.ov@data <- cbind(lime.ov@data, meuse["lime"]@data)
> library(mda)
> library(vcd) # kappa statistics
> Kappa(confusion(lime.ov$lime, as.factor(round(lime.ov$var1.rko, 0))))

              value    ASE
  Unweighted 0.678 0.0671
  Weighted   0.678 0.0840
```

which shows that in 68% of cases the predicted liming requirement class matches the field records.

[18] The higher the nugget, the more the algorithm will smooth the residuals. In the case of pure nugget effect, it does not make any difference if we use only results of regression, or if we add interpolated residuals to the regression predictions.

5.5 Advanced exercises

5.5.1 Geostatistical simulations

A problem with kriging is that it over-smooths reality; especially processes that exhibits a nugget effect in the variogram model. The kriging predictor is the "best linear unbiased predictor" (BLUP) at each point, but the resulting field is commonly smoother than in reality (recall Fig. 1.4). This causes problems when running distributed models, e.g. erosion and runoff, and also gives a distorted view of nature to the decision-maker.

A more realistic visualization of reality is achieved by the use of *conditional geostatistical simulations*: the sample points are taken as known, but the interpolated points reproduce the variogram model including the local noise introduced by the nugget effect. The same `krige` method in gstat can be used to generate simulations, by specifying the optional `nsim` ("*number of simulations*") argument. It's interesting to create several 'alternate realities', each of which is equally-probable. We can re-set R's random number generator with the `set.seed` method to ensure that the simulations will be generated with the same random number seed[19], and then generate four realizations:

```
> set.seed(25)
> zinc.rksim <- krige(step.zinc$call$formula, meuse.ov, pc.comps,
+     zinc.rvgm, nsim=4, nmax=100)
```

```
  drawing 4 GLS realisations of beta...
  [using conditional Gaussian simulation]
```

Now back-transform the predicted values, and plot all four simulations together:

```
# back-transform the values:
> for(i in 1:length(zinc.rksim@data)){
>   zinc.rksim@data[,i] <- expm1(zinc.rksim@data[,i])
> }
> spplot(zinc.rksim, col.regions=grey(c(rev(seq(0,0.97,1/length(at.zinc))),0)),
+       at=at.zinc, main="RK simulations of log(zinc)")
```

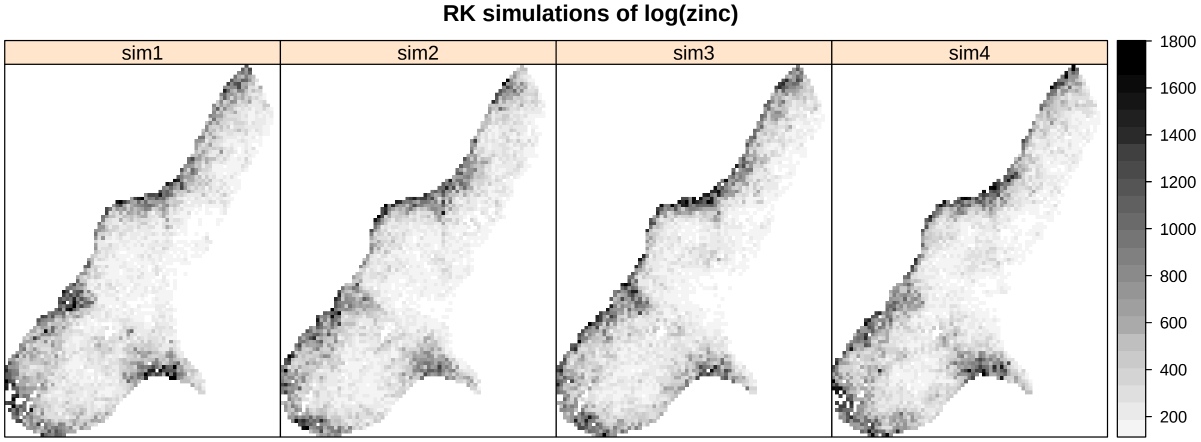


Fig. 5.12: Four simulations of zinc using the fitted regression kriging model. Back-transformed to original scale (ppm). Compare with Fig. 5.9.

which shows that the general pattern of the zinc distribution repeats in each simulation (Fig. 5.12). However, we can also notice that some small features are not as clear as they look in Fig. 5.9. For example, it is relatively hard to notice the borders of the soil units, which in this case change from simulation to simulation. This confirms that the best predictor of zinc is the distance to the river (`dist.asc` map).

To produce simulations of liming requirements, we can run (Bivand et al., 2008, p.230):

[19] Hence differences between the simulated realizations are due only to the different values of the model parameters.

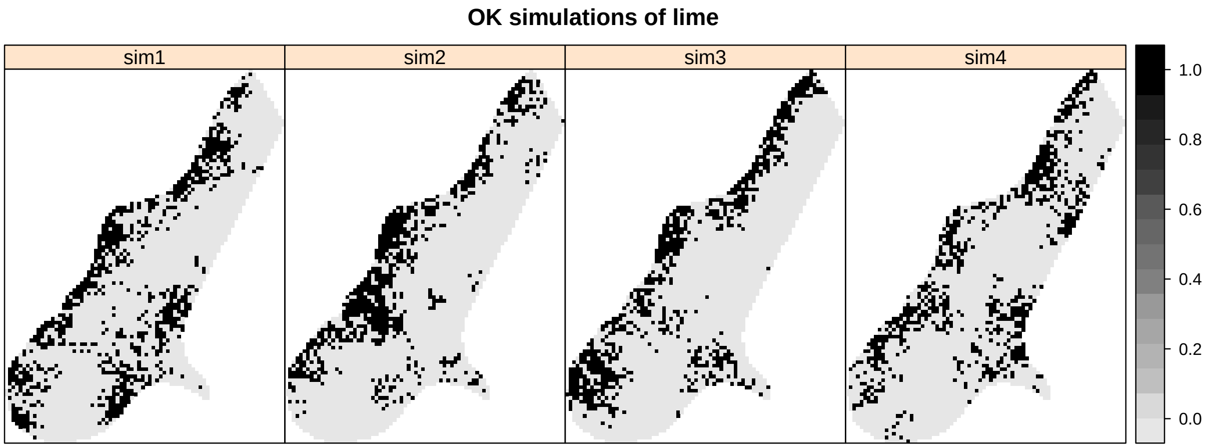


Fig. 5.13: Four simulations of liming requirements (indicator variable) using ordinary kriging. Compare with Fig. 5.11.

```
# fit a variogram:
> lime.ovgm <- fit.variogram(variogram(I(lime == 1) ~ 1, meuse), vgm(1, "Sph", 800, 1))
> lime.sim <- krige(I(lime == 1) ~ 1, meuse, meuse.grid, lime.ovgm,
+          nsim=4, indicators=TRUE, nmax=40)

  drawing 4 GLS realisations of beta...
  [using conditional indicator simulation]


> spplot(lime.sim, col.regions=grey(c(rev(seq(0,0.9,0.05)), 0))
+         main="OK simulations of lime")
```

the result is shown in Fig. 5.13. Note that in the case of liming requirements, there is a distinct difference between the variogram of the original variable and of the residuals: most of spatially-correlated variation in the lime requirements can be explained with auxiliary predictors, so that the variogram of residuals shows pure nugget effect (Fig. 5.10, right).

5.5.2 Spatial prediction using SAGA GIS

Although SAGA has relatively limited geostatistical functionality, it contains a number of modules that are of interest for geostatistical mapping: multiple linear regression (points and grids), ordinary kriging, and regression-kriging. To start, we can examine if the algorithms implemented in gstat and SAGA are the same. We can use the same model parameters estimated in section 5.3 to produce predictions of `log1p(zinc)`. First, we need to prepare vector and gridded maps in SAGA format:

```
# export the point data (transformed!):
> meuse.ov$log1p_zinc <- log1p(meuse.ov$zinc)
> writeOGR(meuse.ov["log1p_zinc"], ".", "zinc", "ESRI Shapefile")
# export the grids to SAGA format:
> PCs.list <- names(step.zinc$model)[-1]
> for(i in PCs.list){
>   write.asciigrid(pc.comps[i], paste(i, ".asc", sep=""), na.value=-9999)
> }
> rsaga.esri.to.sgrd(in.grids=set.file.extension(PCs.list, ".asc"),
+     out.sgrds=set.file.extension(PCs.list, ".sgrd"), in.path=getwd())
```

and then make the predictions by using the Universal kriging module in SAGA (Fig. 5.14):

```
# regression-kriging using the same parameters fitted previously:
> gridcell <- pc.comps@grid@cellsize[1]
> rsaga.geoprocessor(lib="geostatistics_kriging", module=8,
+     param=list(GRID="zinc_rk_SAGA.sgrd", SHAPES="zinc.shp",
+     GRIDS=paste(set.file.extension(PCs.list, ".sgrd"), collapse=";", sep=""),
+     BVARIANCE=F, BLOCK=F, FIELD=1, BLOG=F, MODEL=1, TARGET=0,
+     USER_CELL_SIZE=gridcell, NUGGET=zinc.rvgm$psill[1], SILL=zinc.rvgm$psill[2],
+     RANGE=zinc.rvgm$range[2], INTERPOL=0,
+     USER_X_EXTENT_MIN=pc.comps@bbox[1,1]+gridcell/2,
+     USER_X_EXTENT_MAX=pc.comps@bbox[1,2]-gridcell/2,
+     USER_Y_EXTENT_MIN=pc.comps@bbox[2,1]+gridcell/2,
+     USER_Y_EXTENT_MAX=pc.comps@bbox[2,2]-gridcell/2))
```

```
  SAGA CMD 2.0.4
  library path:   C:/Progra~1/saga_vc/modules
  library name:   geostatistics_kriging
  module name :   Universal Kriging (Global)
  author      :   (c) 2003 by O.Conrad

  Load shapes: zinc.shp...
  ready

  Load grid: PC1.sgrd...
  ready

  ...

  Load grid: PC6.sgrd...
  ready

  Parameters

  Grid: [not set]
  Variance: [not set]
  Points: zinc.shp
  Attribute: log1p_zinc
  Create Variance Grid: no
  Target Grid: user defined
  Variogram Model: Exponential Model
  Block Kriging: no
  Block Size: 100.000000
  Logarithmic Transformation: no
  Nugget: 0.054588
  Sill: 0.150518
  Range: 374.198454
  Linear Regression: 1.000000
  Exponential Regression: 0.100000
  Power Function - A: 1.000000
  Power Function - B: 0.500000
  Grids: 5 objects (PC1.sgrd, PC2.sgrd, PC3.sgrd, PC4.sgrd, PC6.sgrd))
  Grid Interpolation: Nearest Neighbor

  Save grid: zinc_rk_SAGA.sgrd...
```

Visually (Fig. 5.14), the results look as if there is no difference between the two pieces of software. We can then load back the predictions into R to compare if the results obtained with gstat and SAGA match exactly:

```
> rsaga.sgrd.to.esri(in.sgrds="zinc_rk_SAGA",
+     out.grids="zinc_rk_SAGA.asc", out.path=getwd())
> zinc.rk$SAGA <- readGDAL("zinc_rk_SAGA.asc")$band1
> plot(zinc.rk$SAGA, zinc.rk$var1.pred, pch=19, xlab="SAGA", ylab="gstat")
> lines(3:8, 3:8, col="grey", lwd=4)
```

which shows that both software programs implement the same algorithm, but there are some small differences between the predicted values that are possibly due to rounding effect.

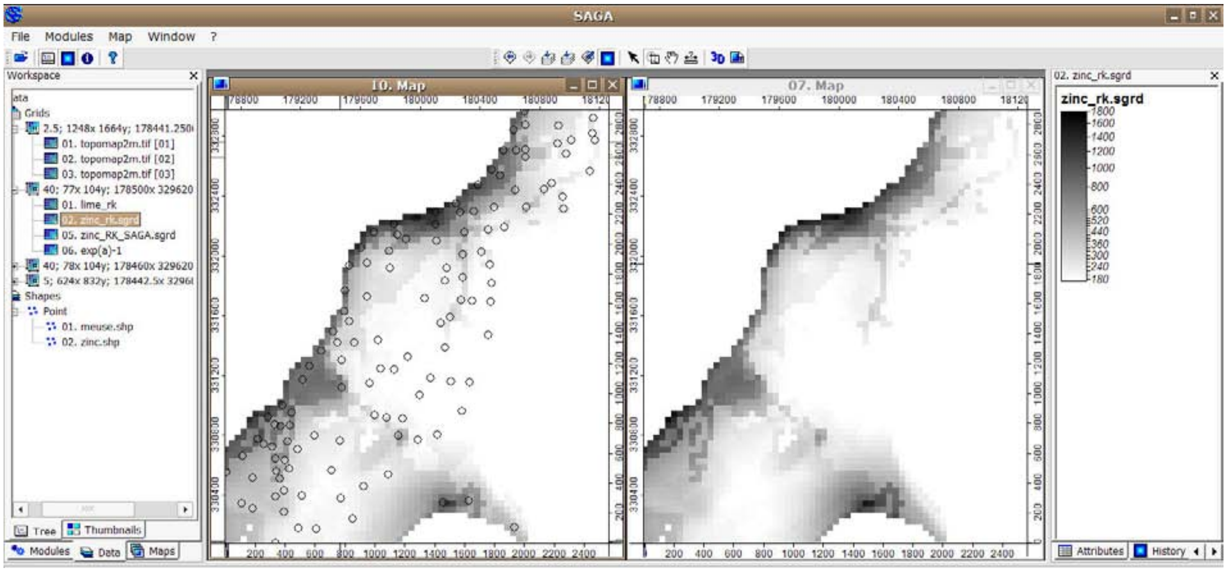


Fig. 5.14: Comparing results from SAGA (left map) and gstat (right map): regression-kriging using the same model parameters estimated in section 5.3.

Next, we want to compare the computational efficiency of gstat and SAGA, i.e. the processing time. To emphasize the difference in computation time, we will use a somewhat larger grid (2 m), and then re-run ordinary kriging in both software packages:

```
> meuse.grid2m <- readGDAL("topomap2m.tif")

  topomap2m.tif has GDAL driver GTiff
  and has 1664 rows and 1248 columns


> proj4string(meuse.grid2m) <- meuse.grid@proj4string
```

Processing speed can be measured from R by using the `system.time` method, which measures the elapsed seconds:

```
> system.time(krige(log1p(zinc) ~ 1, meuse, meuse.grid2m, zinc.vgm))

  [using ordinary kriging]
     user  system elapsed
   319.14    7.96  353.44
```

and now the same operation in SAGA:

```
> cellsize2 <- meuse.grid2m@grid@cellsize[1]
> system.time(rsaga.geoprocessor(lib="geostatistics_kriging", module=6,
+     param=list(GRID="zinc_ok_SAGA.sgrd", SHAPES="zinc.shp", BVARIANCE=F, BLOCK=F,
+     FIELD=1, BLOG=F, MODEL=1, TARGET=0, USER_CELL_SIZE=cellsize2,
+     NUGGET=zinc.vgm$psill[1], SILL=zinc.vgm$psill[2], RANGE=zinc.rvgm$range[2],
+     USER_X_EXTENT_MIN=meuse.grid2m@bbox[1,1]+cellsize2/2,
+     USER_X_EXTENT_MAX=meuse.grid2m@bbox[1,2]-cellsize2/2,
+     USER_Y_EXTENT_MIN=meuse.grid2m@bbox[2,1]+cellsize2/2,
+     USER_Y_EXTENT_MAX=meuse.grid2m@bbox[2,2]-cellsize2/2)))

     user  system elapsed
     0.03    0.71  125.69
```

We can see that SAGA will be faster for processing large data sets. This difference will become even larger if we would use large point data sets. Recall that the most '*expensive*' operation for any geostatistical mapping is the derivation of distances between points. Thus, by limiting the search radius one can always increase the processing speed. The problem is that a software needs to initially estimate which points fall within the search radius, hence it always has to take into account location of all points. Various *quadtree* and similar algorithms then exist to speed up the neighborhood search algorithm (partially available in gstat also), but their implementation can differ between various programming languages.

Note also that it is not really a smart idea to try to visualize large maps in R. R graphics plots grids as vectors; each grid cell is plotted as a separate polygon, which takes a huge amount of RAM for large grids, and can last up to few minutes. SAGA on other hand can handle and display grids ≫10 million pixels on a standard PC without any delays (Fig. 5.14). When you move to other exercises you will notice that we will typically use R to fit models, SAGA to run predictions and visualize results, and Google Earth to visualize and explore final products.

5.5.3 Geostatistical analysis in geoR

We start by testing the variogram fitting functionality of geoR. However, before we can do any analysis, we need to convert our point map (sp) to geoR geodata format:

```
> zinc.geo <- as.geodata(meuse.ov["zinc"])
> str(zinc.geo)
```

```
 List of 2
  $ x , y    : num [1:155, 1:2] 181072 181025 181165 181298 181307 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:155] "300" "455" "459" "540" ...
   .. ..$ : chr [1:2] "x" "y"
  $ data       : num [1:155] 1022 1141 640 257 269 ...
  - attr(*, "class")= chr "geodata"
```

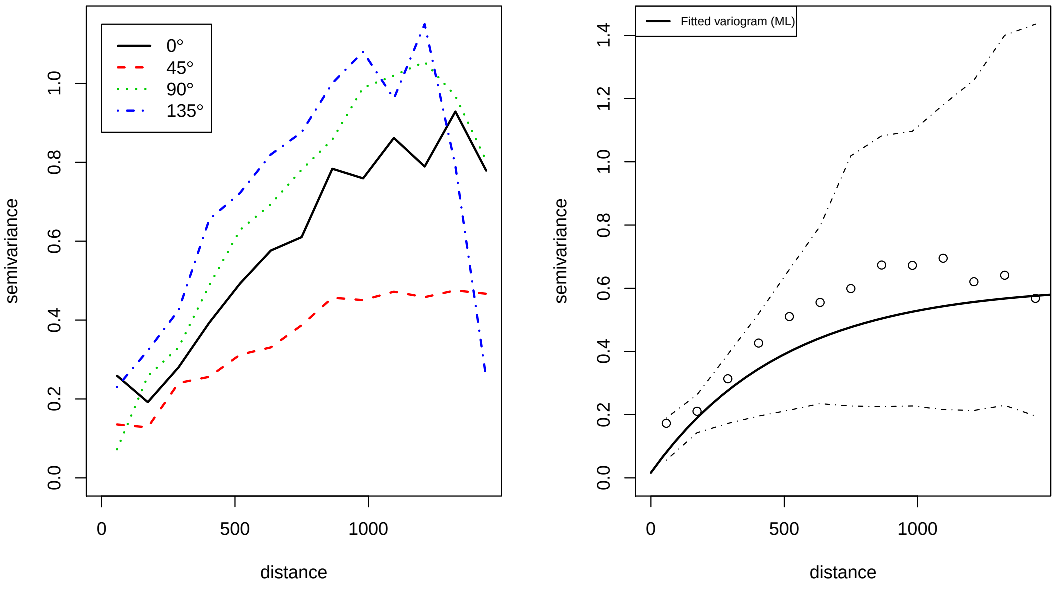


Fig. 5.15: Anisotropy (left) and variogram model fitted using the Maximum Likelihood (ML) method (right). The confidence bands (*envelopes*) show the variability of the sample variogram estimated using simulations from a given set of model parameters.

which shows much simpler structure than a SpatialPointsDataFrame. A geodata-type object contains only: a matrix with coordinates of sampling locations (coords), values of target variables (data), matrix with coordinates of the polygon defining the mask map (borders), vector or data frame with covariates (covariate). To produce the two standard variogram plots (Fig. 5.15), we will run:

```
> par(mfrow=c(1,2))
# anisotropy ("lambda=0" indicates log-transformation):
> plot(variog4(zinc.geo, lambda=0, max.dist=1500, messages=FALSE), lwd=2)
# fit variogram using likfit:
> zinc.svar2 <- variog(zinc.geo, lambda=0, max.dist=1500, messages=FALSE)
> zinc.vgm2 <- likfit(zinc.geo, lambda=0, messages=FALSE,
+       ini=c(var(log1p(zinc.geo$data)),500), cov.model="exponential")
> zinc.vgm2

  likfit: estimated model parameters:
        beta      tausq    sigmasq        phi
  "  6.1553" "  0.0164" "  0.5928" "500.0001"
  Practical Range with cor=0.05 for asymptotic range: 1498

  likfit: maximised log-likelihood = -1014


# generate confidence bands for the variogram:
> env.model <- variog.model.env(zinc.geo, obj.var=zinc.svar2, model=zinc.vgm2)

  variog.env: generating 99 simulations (with 155 points each) using grf
  variog.env: adding the mean or trend
  variog.env: computing the empirical variogram for the 99 simulations
  variog.env: computing the envelops


> plot(zinc.svar2, envelope=env.model); lines(zinc.vgm2, lwd=2);
> legend("topleft", legend=c("Fitted variogram (ML)"), lty=c(1), lwd=c(2), cex=0.7)
> dev.off()
```

where variog4 is a method that generates semivariances in four directions, lambda=0 is used to indicate the type of transformation[20], likfit is the generic variogram fitting method, ini is the given initial variogram, and variog.model.env calculates confidence limits for the fitted variogram model. Parameters tausq and sigmasq corresponds to nugget and sill parameters; phi is the range parameter.

In general, geoR offers much richer possibilities for variogram modeling than gstat. From Fig. 5.15(right) we can see that the variogram fitted using this method does not really go through all points (compare with Fig. 5.8). This is because the ML method discounts the potentially wayward influence of sample variogram at large inter-point distances (Diggle and Ribeiro Jr, 2007). Note also that the confidence bands (*envelopes*) also confirm that the variability of the empirical variogram increases with larger distances.

Now that we have fitted the variogram model, we can produce predictions using the ordinary kriging model. Because geoR does not work with sp objects, we need to prepare the prediction locations:

```
> locs <- pred_grid(c(pc.comps@bbox[1,1]+gridcell/2,
+     pc.comps@bbox[1,2]-gridcell/2), c(pc.comps@bbox[2,1]+gridcell/2,
+     pc.comps@bbox[2,2]-gridcell/2), by=gridcell)
# match the same grid as pc.comps;
```

and the mask map i.e. a polygon showing the borders of the area of interest:

```
> meuse.grid$mask <- ifelse(!is.na(meuse.grid$dist), 1, NA)
> write.asciigrid(meuse.grid["mask"], "mask.asc", na.value=-1)
# raster to polygon conversion;
> rsaga.esri.to.sgrd(in.grids="mask.asc", out.sgrd="mask.sgrd", in.path=getwd())
> rsaga.geoprocessor(lib="shapes_grid", module=6, param=list(GRID="mask.sgrd",
```

[20] geoR implements the Box–Cox transformation (Diggle and Ribeiro Jr, 2007, p.61), which is somewhat more generic than simple log() transformation.

```
+      SHAPES="mask.shp", CLASS_ALL=1))
> mask <- readShapePoly("mask.shp", proj4string=CRS("+init=epsg:28992"),
+      force_ring=T)
# coordinates of polygon defining the area of interest:
> mask.bor <- mask@polygons[[1]]@Polygons[[1]]@coords
> str(mask.bor)

   num [1:267, 1:2] 178880 178880 178760 178760 178720 ...
```

Ordinary kriging can be run by using the generic method for linear Gaussian models `krige.conv`[21]:

```
> zinc.ok2 <- krige.conv(zinc.geo, locations=locs,
+      krige=krige.control(obj.m=zinc.vgm2), borders=mask.bor)


  krige.conv: results will be returned only for locations inside the borders
  krige.conv: model with constant mean
  krige.conv: performing the Box-Cox data transformation
  krige.conv: back-transforming the predicted mean and variance
  krige.conv: Kriging performed using global neighborhood


# Note: geoR will automatically back-transform the values!
> str(zinc.ok2)

  List of 6
   $ predict     : num [1:3296] 789 773 756 740 727 ...
   $ krige.var   : num [1:3296] 219877 197718 176588 159553 148751 ...
   $ beta.est    : Named num 6.16
    ..- attr(*, "names")= chr "beta"
   $ distribution: chr "normal"
   $ message     : chr "krige.conv: Kriging performed using global neighbourhood"
   $ call        : language krige.conv(geodata = zinc.geo, locations = locs,
             borders = mask.bor,      krige = krige.control(obj.m = zinc.vgm2))
   - attr(*, "sp.dim")= chr "2d"
   - attr(*, "prediction.locations")= symbol locs
   - attr(*, "parent.env")=<environment: R_GlobalEnv>
   - attr(*, "data.locations")= language zinc.geo$coords
   - attr(*, "borders")= symbol mask.bor
   - attr(*, "class")= chr "kriging"
```

To produce plots shown in Fig. 5.16, we use:

```
> par(mfrow=c(1,2))
> image(zinc.ok2, loc=locs, col=gray(seq(1,0.1,l=30)), xlab="Coord X",
+     ylab="Coord Y")
> title(main="Ordinary kriging predictions")
> contour(zinc.ok2, add=TRUE, nlev=8)
> image(zinc.ok2, loc=locs, value=sqrt(zinc.ok2$krige.var),
+     col=gray(seq(1,0.1,l=30)), xlab="Coord X", ylab="Coord Y")
> title(main="Prediction error")
> krige.var.vals <- round(c(quantile(sqrt(zinc.ok2$krige.var),0.05),
+     sd(zinc.geo$data), quantile(sqrt(zinc.ok2$krige.var), 0.99)), 0)
> legend.krige(x.leg=c(178500,178800), y.leg=c(332500,333500),
+     values=sqrt(zinc.ok2$krige.var), vert=TRUE, col=gray(seq(1,0.1,l=30)),
+     scale.vals=krige.var.vals)
> points(zinc.geo[[1]], pch="+", cex=.7)
```

To run regression-kriging (in geoR "*external trend kriging*") we first need to add values of covariates to the original `geodata` object:

[21] Meaning "*kriging conventional*" i.e. linear kriging.

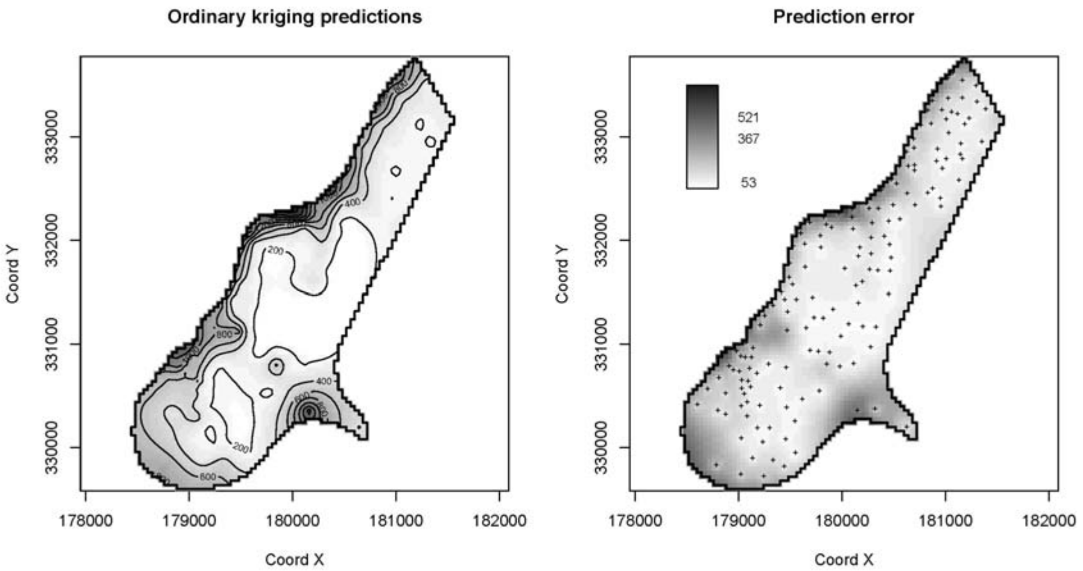


Fig. 5.16: Zinc predicted using ordinary kriging in geoR. The map on the left is considered to be below critical accuracy level in the areas where the prediction error (right map) exceeds the global variance (the middle value in legend). Compare with Fig. 5.9.

```
> zinc.geo$covariate <- meuse.ov@data[,PCs.list]
```

which now allows us to incorporate the trend argument in the variogram model:

```
# trend model:
> step.zinc$call$formula[c(1,3)]

  ~ PC1 + PC2 + PC3 + PC4 + PC6


> zinc.rvgm2 <- likfit(zinc.geo, lambda=0, trend=step.zinc$call$formula[c(1,3)],
+   messages=FALSE, ini=c(var(residuals(step.zinc)),500), cov.model="exponential")
> zinc.rvgm2

  likfit: estimated model parameters:
       beta0      beta1      beta2      beta3      beta4      beta5
  "  5.6919" " -0.4028" " -0.1203" " -0.0176" "  0.0090" " -0.3199"
       tausq    sigmasq        phi
  "  0.0526" "  0.1894" "499.9983"
  Practical Range with cor=0.05 for asymptotic range: 1498

  likfit: maximised log-likelihood = -975
```

Note that geoR reports also the regression coefficients for the five predictors (beta0 is the intercept). In the case of gstat this information will be hidden: gstat will typically fit a regression model only to derive the residuals (regression coefficients can be printed by setting the debugging options). Neither gstat nor geoR report on the goodness of fit and similar regression diagnostics.

Before we can make predictions, we also need to prepare the covariates at all locations. Unfortunately, geoR is not compatible with sp grids, so we need to prepare the covariate values so they exactly match prediction locations:

```
# get values of covariates at new locations:
> locs.sp <- locs
> coordinates(locs.sp) <- ~ Var1+Var2
> PCs.gr <- overlay(pc.comps, locs.sp)
# fix NAs:
> for(i in PCs.list){
>   PCs.gr@data[,i] <- ifelse(is.na(PCs.gr@data[,i]), 0, PCs.gr@data[,i])
> }
```

which allows us to run predictions using the same trend model as used in section 5.3.1:

```
# define the geostatistical model:
> KC <- krige.control(trend.d = step.zinc$call$formula[c(1,3)],
+     trend.l = ~ PCs.gr$PC1+PCs.gr$PC2+PCs.gr$PC3+PCs.gr$PC4+PCs.gr$PC6,
+     obj.m = zinc.rvgm2)
# run predictions (external trend kriging):
> zinc.rk2 <- krige.conv(zinc.geo, locations=locs, krige=KC, borders=mask.bor)

  krige.conv: results will be returned only for prediction inside the borders
  krige.conv: model with mean defined by covariates provided by the user
  krige.conv: performing the Box-Cox data transformation
  krige.conv: back-transforming the predicted mean and variance
  krige.conv: Kriging performed using global neighbourhood
```

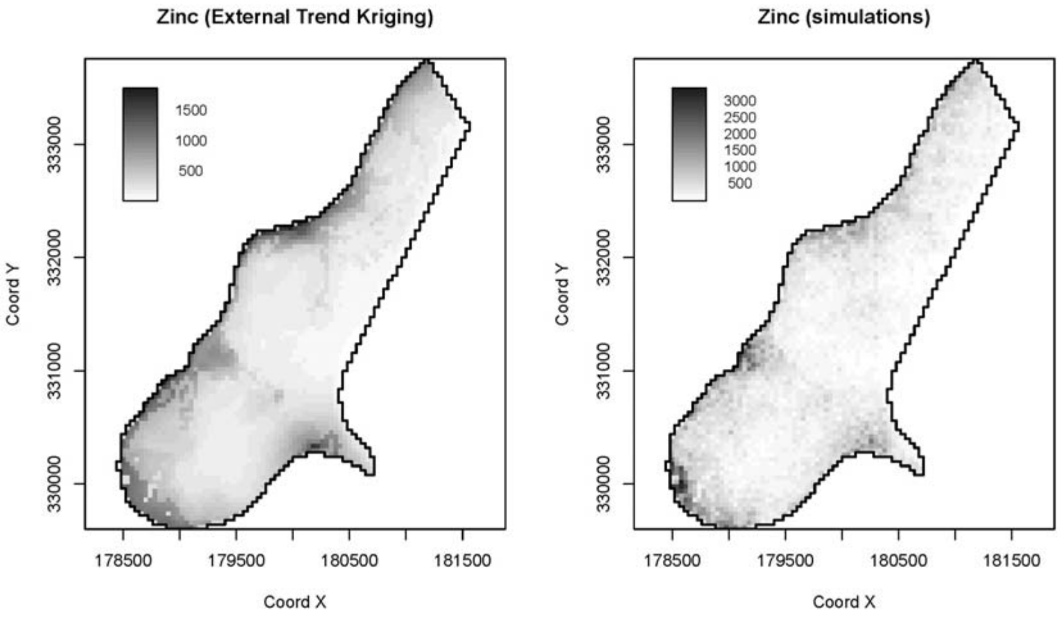


Fig. 5.17: Zinc predicted using external trend kriging in geoR (left); simulations using the same model (right). Compare with Figs. 5.9 and 5.12.

The result is shown in Fig. 5.17. geoR also allows generation of simulations using the same external trend model by setting the output.control parameter (the resulting map shown in Fig. 5.17; right):

```
> zinc.rk2 <- krige.conv(zinc.geo, locations=locs, krige=KC, borders=mask.bor,
+     output=output.control(n.predictive=1))

  krige.conv: results will be returned only for prediction inside the borders
  krige.conv: model with mean defined by covariates provided by the user
  krige.conv: performing the Box-Cox data transformation
  krige.conv: sampling from the predictive distribution (conditional simulations)
  krige.conv: back-transforming the simulated values
  krige.conv: back-transforming the predicted mean and variance
  krige.conv: Kriging performed using global neighborhood
```

which shows a somewhat higher range of values than the simulation using a simple linear model (Fig. 5.12). In this case geoR seems to do better in accounting for the skewed distribution of values than gstat. However such simulations in geoR are extremely computationally intensive, and are not recommended for large data sets. In fact, many default methods implemented in geoR (Maximum Likelihood fitting for variograms, Bayesian methods and conditional simulations) are definitively not recommended with data sets with ≫1000 sampling points and/or over ≫100,000 new locations. Creators of geoR seem to have selected a path of running only global neighborhood analysis on the point data. Although the author of this guide supports that decision (see also section 2.2), some solution needs to be found to process larger point data sets because computing time exponentially increases with the size of the data set.

Finally, the results of predictions can be exported[22] to some GIS format by copying the values to an sp frame:

```
> mask.ov <- overlay(mask, locs.sp)
> mask.sel <- !is.na(mask.ov$MASK.SGRD)
> locs.geo <- data.frame(X=locs.sp@coords[mask.sel,1],
+    Y=locs.sp@coords[mask.sel,2], zinc.rk2=zinc.rk2[[1]],
+    zinc.rkvar2=zinc.rk2[[2]])
> coordinates(locs.geo) <- ~ X+Y
> gridded(locs.geo) <- TRUE
> write.asciigrid(locs.geo[1], "zinc_rk2.asc", na.value=-1)
```

5.6 Visualization of generated maps

5.6.1 Visualization of uncertainty

The following paragraphs explain how to visualize results of geostatistical mapping to explore uncertainty in maps. We will focus on the technique called **whitening**, which is a simple but efficient technique to visualize mapping error (Hengl and Toomanian, 2006). It is based on the Hue-Saturation-Intensity (HSI) color model (Fig. 5.18a) and calculations with colors using the color mixture (CM) concept. The HSI is a psychologically appealing color model — hue is used to visualize values or taxonomic space and whiteness (paleness) is used to visualize the uncertainty (Dooley and Lavin, 2007). For this purpose, a 2D legend was designed to accompany the visualizations. Unlike standard legends for continuous variables, this legend has two axis (Fig. 5.18b): (1) vertical axis (hues) is used to visualize the predicted values and (2) horizontal axis (whiteness) is used to visualize the prediction error. Fig. 5.19 shows an example of visualization using whitening for the meuse data set.

Visualization of uncertainty in maps using whitening can be achieved using one of the two software programs: ILWIS and R. In ILWIS, you can use the VIS_error script that can be obtained from the author's homepage. To visualize the uncertainty for your own case study using this technique, you should follow these steps (Hengl and Toomanian, 2006):

(1.) Download the ILWIS script (VIS_error[23]) for visualization of prediction error and unzip it to the default directory (C:\Program Files\ILWIS\Scripts\).

(2.) Derive the predictions and prediction variance for some target variable. Import both maps to ILWIS. The prediction variance needs to be then converted to normalized prediction variance by using Eq.(1.4.4), so you will also need to determine the global variance of your target variable.

(3.) Start ILWIS and run the script from the left menu (operations list) or from the main menu ↦ *Operations* ↦ *Scripts* ↦ VIS_error. Use the help button to find more information about the algorithm.

(4.) To prepare final layouts, you will need to use the legend2D.tif legend file[24].

A more interesting option is to visualize maps using whitening in R[25]. You will need to load the following additional package:

[22] Note that the results of prediction in geoR is simply a list of values without any spatial reference.

[23] http://spatial-analyst.net/scripts/

[24] http://spatial-analyst.net/scripts/legend2D.tif; This legend is a Hue-whitening legend: in the vertical direction only Hue values change, while in the horizontal direction amount of white color is linearly increased from 0.5 up to 1.0.

[25] http://spatial-analyst.net/scripts/whitening.R

```
> library(colorspace)
```

The example with the meuse data set:

```
# ordinary kriging:
> m <- vgm(.59, "Sph", 874, .04)
> vismaps <- krige(log(zinc) ~ 1, meuse, meuse.grid, model=m)
```

As a result of ordinary kriging, we have produced two maps: predictions and the prediction variance. Now, we can visualize both maps together using whitening. We start by setting up threshold levels (lower and upper limits), and stretching the values within that range:

```
> z1 <- min(log(meuse$zinc), na.rm=TRUE)
> z2 <- max(log(meuse$zinc), na.rm=TRUE)
> e1 <- 0.4
> e2 <- 0.7
> vismaps$er <- sqrt(vismaps$e)/sqrt(var(log(meuse$zinc)))
> vismaps$tmpz <- (vismaps$z-z1)/(z2-z1)
# Mask the values outside the 0-1 range:
> vismaps$tmpzc <- ifelse(vismaps$tmpz<=0, 0, ifelse(vismaps$tmpz>1, 1, vismaps$tmpz))
```

The Hue-Saturation-Value (HSV) bands we can generate using:

```
# The hues should lie between between 0 and 360, and the saturations
# and values should lie between 0 and 1.
> vismaps$tmpf1 <- -90-vismaps$tmpzc*300
> vismaps$tmpf2 <- ifelse(vismaps$tmpf1<=-360, vismaps$tmpf1+360, vismaps$tmpf1)
> vismaps$H <- ifelse(vismaps$tmpf2>=0, vismaps$tmpf2, (vismaps$tmpf2+360))
# Strech the error values (e) to the inspection range:
# Mask the values out of the 0-1 range:
> vismaps$tmpe <- (vismaps$er-e1)/(e2-e1)
> vismaps$tmpec <- ifelse(vismaps$tmpe<=0, 0, ifelse(vismaps$tmpe>1, 1, vismaps$tmpe))
# Derive the saturation and intensity images:
> vismaps$S <- 1-vismaps$tmpec
> vismaps$V <- 0.5*(1+vismaps$tmpec)
```

The HSV values can be converted to RGB bands using:

```
> RGBimg <- as(HSV(vismaps$H, vismaps$S, vismaps$V), "RGB")
> summary(RGBimg@coords)
> vismaps$red <- as.integer(ifelse(is.na(vismaps@data[1]), 255, RGBimg@coords[,1]*255))
> vismaps$green <- as.integer(ifelse(is.na(vismaps@data[1]), 255, RGBimg@coords[,2]*255))
> vismaps$blue <- as.integer(ifelse(is.na(vismaps@data[1]), 255, RGBimg@coords[,3]*255))
> summary(vismaps[c("red", "green", "blue")])
```

```
Object of class SpatialGridDataFrame
Coordinates:
     min    max
x 178440 181560
y 329600 333760
Is projected: NA
proj4string : [NA]
Number of points: 2
Grid attributes:
  cellcentre.offset cellsize cells.dim
x            178460       40        78
y            329620       40       104
Data attributes:
      red             green             blue      
 Min.   :  0.0   Min.   :  0.0   Min.   :  0.0  
 1st Qu.:153.0   1st Qu.:183.0   1st Qu.:194.0  
 Median :255.0   Median :255.0   Median :255.0  
 Mean   :206.2   Mean   :220.5   Mean   :219.2  
 3rd Qu.:255.0   3rd Qu.:255.0   3rd Qu.:255.0  
 Max.   :255.0   Max.   :255.0   Max.   :255.0  
```

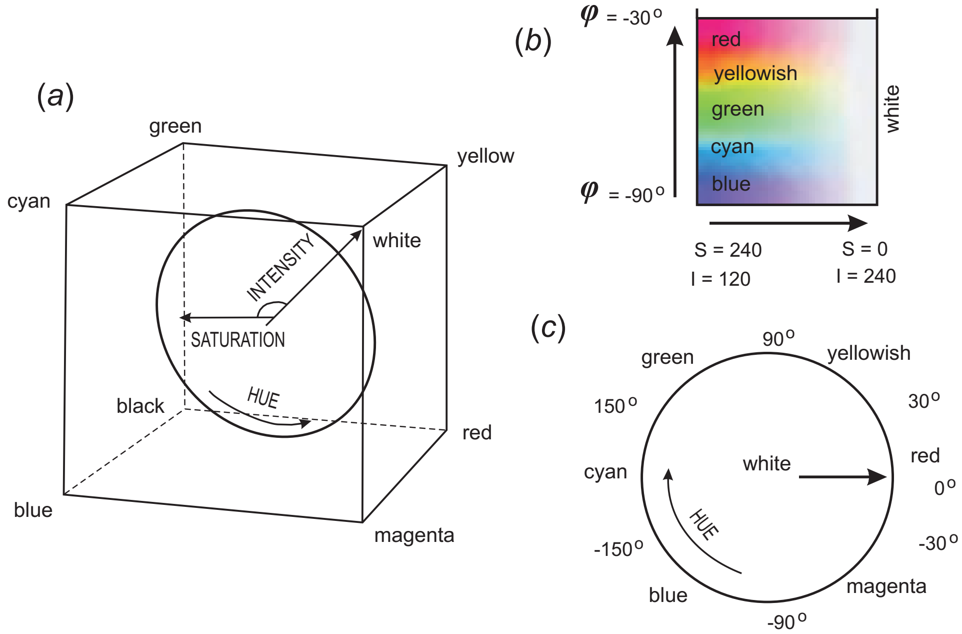


Fig. 5.18: Design of the special 2D legend used to visualize the prediction variance using whitening: (a) the HSI color model, (b) the 2D legend and (c) the common types of Hues. After Hengl et al. (2004a).

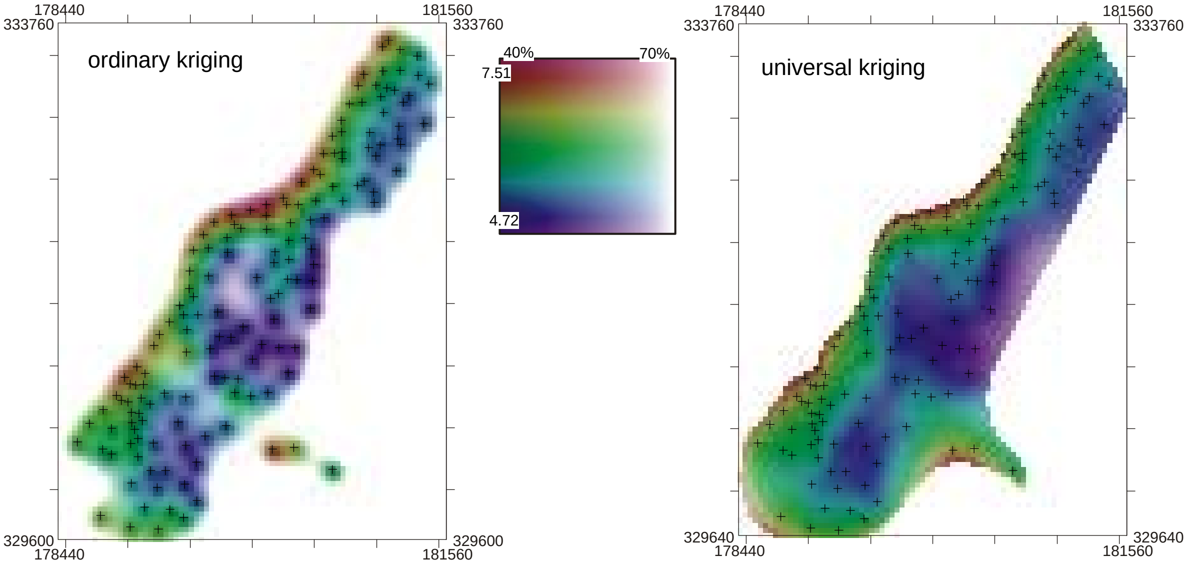


Fig. 5.19: Mapping uncertainty for zinc visualized using whitening: ordinary kriging (left) and universal kriging (right). Predicted values in log-scale. See cover of this book for a color version of this figure.

which is now a spatial object with three RGB bands. To display a true RGB image in R, use the SGDF2PCT method[26]:

```
> RGBimg <- SGDF2PCT(vismaps[c("red", "green", "blue")], ncolors=256, adjust.bands=FALSE)
> vismaps$idx <- RGBimg$idx
> image(vismaps, "idx", col=RGBimg$ct)
> plot(meuse, pch="+", add=TRUE)
```

In the last step (optional), we can set the right georeference and export the map to e.g. GeoTIFF format:

```
> proj4string(vismaps) <- CRS("+init=epsg:28992")
# Export as geoTIFF / or any other format:
> writeGDAL(vismaps[c("red", "green", "blue")], "vismap.tif", drivername="GTiff",
+    type="Byte", options="INTERLEAVE=PIXEL")
```

A comparison of uncertainty for maps produced using ordinary kriging and universal kriging in gstat can be seen in Fig. 5.19. In this case, the universal kriging map is distinctly more precise. You can manually change the lower and upper values for both prediction and error maps depending on your mapping requirements. By default, thresholds of 0.4 and 0.8 (max 1.0) are used for the normalized prediction error values. This assumes that a satisfactory prediction is when the model explains more than 85% of the total variation (normalized error = 40%; see p.23). Otherwise, if the value of the normalized error get above 80%, the model accounts for less than 50% of variability at calibration points and the prediction is probably unsatisfactory.

To prepare the 2D legend shown in Fig. 5.19 (100×100 pixels), we use:

```
> legend.2D <- expand.grid(x=seq(.01,1,.01),y=seq(.01,1,.01))
# Hues
> legend.2D$tmpf1 <- -90-legend.2D$y*300
> legend.2D$tmpf2 <- ifelse(legend.2D$tmpf1<=-360, legend.2D$tmpf1+360,
+      legend.2D$tmpf1)
> legend.2D$H <- ifelse(legend.2D$tmpf2>=0, legend.2D$tmpf2, (legend.2D$tmpf2+360))
# Saturation:
> legend.2D$S <- 1-legend.2D$x
# Intensity:
> legend.2D$V <- 0.5+legend.2D$x/2
> gridded(legend.2D) <- ~ x+y
> legend.2D <- as(legend.2D, "SpatialGridDataFrame")
> legendimg <- as(HSV(legend.2D$H, legend.2D$S, legend.2D$V), "RGB")
> legend.2D$red <- as.integer(legendimg@coords[,1]*255)
> legend.2D$green <- as.integer(legendimg@coords[,2]*255)
> legend.2D$blue <- as.integer(legendimg@coords[,3]*255)
# Write as a RGB image:
> legend.2Dimg <- SGDF2PCT(legend.2D[c("red", "green", "blue")], ncolors=256,
+      adjust.bands=FALSE)
> legend.2D$idx <- legend.2Dimg$idx
> writeGDAL(legend.2D[c("red", "green", "blue")], "legend2D.tif",
+      drivername="GTiff", type="Byte", options="INTERLEAVE=PIXEL")
```

Another sophisticated option to visualize the results of (spatio-temporal) geostatistical mapping is the stand-alone visualization software called Aquila[27] (Pebesma et al., 2007). Aquila facilitates interactive exploration of the spatio-temporal **Cumulative Distribution Functions** (CDFs) and allows decision makers to explore uncertainty associated with attaching different threshold or its spatial distribution in the area of interest. It is actually rather simple to use — one only needs to prepare a sample (e.g. 12 slices) of quantile estimates, which are then locally interpolated to produce CDFs.

5.6.2 Export of maps to Google Earth

To export maps we have produced to Google Earth, we first need to reproject the maps to the WGS84 coordinate system (the native system for Google Earth). We can first reproject the map of sample points, using the

[26] Note that the results might differ slightly between ILWIS and R, which is mainly due to somewhat different HSI–RGB conversion algorithms. For example, the SGDF2PCT method is limited to 256 colors only!

[27] http://pcraster.geo.uu.nl/projects/aguila/

spTransform method of the sp package, and then export them using writeOGR:

```
> meuse.ll <- spTransform(meuse, CRS("+proj=longlat +datum=WGS84"))
> writeOGR(meuse.ll, "meuse.kml", "meuse", driver="KML")
```

You can examine these in Google Earth by opening the KML file meuse.kml which you just wrote. Next, we want to export the predictions of zinc, which means that we first need to reproject the interpolated values onto geographic coordinates. The most efficient way to achieve this is by using the SAGA proj4 module[28]:

```
> rsaga.geoprocessor(lib="pj_proj4", 2, param=list(SOURCE_PROJ=paste('"',
+     proj4string(meuse.grid), '"', sep=""), TARGET_PROJ="\"+proj=longlat
+     +datum=WGS84\"", SOURCE="zinc_rk.sgrd", TARGET="zinc_rk_ll.sgrd",
+     TARGET_TYPE=0, INTERPOLATION=1))
```

```
 SAGA CMD 2.0.4
 library path:   C:/Progra~1/saga_vc/modules
 library name:   pj_proj4
 module name :   Proj.4 (Command Line Arguments, Grid)
 author      :   O. Conrad (c) 2004-8

 Load grid: zinc_rk.sgrd...
 ready

 Parameters

 Inverse: no
 Source Projection Parameters:  +init=epsg:28992 +proj=sterea
  +lat_0=52.15616055555555 +lon_0=5.38763888888889 +k=0.999908 +x_0=155000
  +y_0=463000 +ellps=bessel +towgs84=565.237,50.0087,465.658,-0.406857,
  0.350733,-1.87035,4.0812 +units=m +no_defs
 Target Projection Parameters: +proj=longlat +datum=WGS84
 Grid system: 40; 77x 104y; 178500x 329620y
 Source: zinc_rk.sgrd
 Target: [not set]
 Shapes: [not set]
 X Coordinates: [not set]
 Y Coordinates: [not set]
 Create X/Y Grids: no
 Target: user defined
 Interpolation: Bilinear Interpolation

 Source:  +init=epsg:28992 +proj=sterea +lat_0=52.15616055555555
 +lon_0=5.38763888888889 +k=0.999908 +x_0=155000 +y_0=463000
 +ellps=bessel +towgs84=565.237,50.0087,465.658,-0.406857,0.350733,
 -1.87035,4.0812 +units=m +no_defs

 Target: +proj=longlat +datum=WGS84

 ready
 Save grid: zinc_rk_ll.sgrd...
```

Once we have created this gridded result, we can plot the maps and export the plots to Google Earth. First we need to set up metadata in the form of a SpatialGrid object for defining the size and placing of a PNG image overlay in Google Earth; this is the job of the GE_SpatialGrid method of the maptools package:

```
# read back into R:
> rsaga.sgrd.to.esri(in.sgrds="zinc_rk_ll.sgrd", out.grids="zinc_rk_ll.asc",
+        out.path=getwd())
> zinc_rk.ll <- readGDAL("zinc_rk_ll.asc")
```

[28] SAGA will automatically estimate both the grid cell size and the bounding box in geographical coordinates. Compare with section 10.6.3.

```
  zinc_rk_ll.asc has GDAL driver AAIGrid
  and has 105 rows and 122 columns
```

```
> proj4string(zinc_rk.ll) <- CRS("+proj=longlat +datum=WGS84")
> zinc_rk.kml <- GE_SpatialGrid(zinc_rk.ll)
```

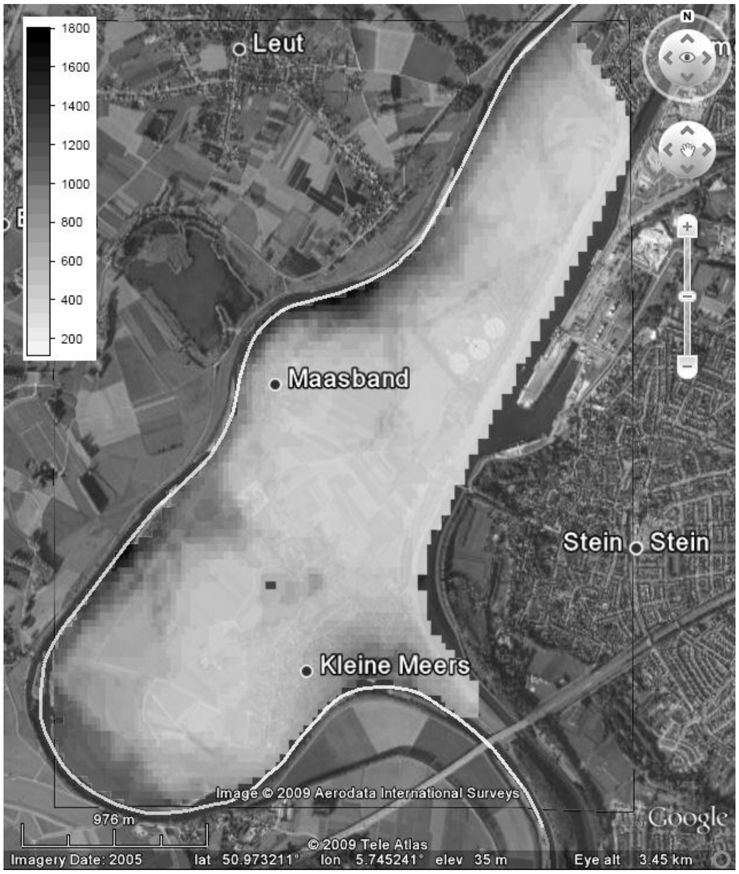


Fig. 5.20: RK predictions of zinc for Meuse area — as visualized in Google Earth.

where zinc_rk.kml is the name of R object, which carries only a definition of the ground overlay frame and not the data to be exported. Next we create a PNG (Portable Network Graphics) file (the format recognized as an overlay by Google Earth) using the png graphics device:

```
> png(file="zinc_rk.png", width=zinc_rk.kml$width, height=zinc_rk.kml$height, bg="transparent")
> par(mar=c(0,0,0,0), xaxs="i", yaxs="i")
> image(as.image.SpatialGridDataFrame(zinc_rk.ll[1]),
+     col=grey(rev(seq(0,0.95,1/length(at.zinc)))),
+     xlim=zinc_rk.kml$xlim, ylim=zinc_rk.kml$ylim)
```

which will plot the map over the whole area of the plotting space, so that border coordinates exactly match the borders of the plot. We then create the overlay itself, from the PNG file, with the kmlOverlay method, specifying the SpatialGrid object that orients the map in Google Earth:

```
> kmlOverlay(zinc_rk.kml, kmlfile="zinc_rk.kml", imagefile="zinc_rk.png", name="zinc")
```

```
  [1] "<?xml version='1.0' encoding='UTF-8'?>"
  [2] "<kml xmlns='http://earth.google.com/kml/2.0'>"
  [3] "<GroundOverlay>"
```

```
[4] "<name>zinc (RK)</name>"
[5] "<Icon><href>zinc_rk.png</href><viewBoundScale>0.75</viewBoundScale></Icon>"
[6] "<LatLonBox><north>50.992973</north><south>50.955488</south>
+     <east>5.76502299201062</east><west>5.721466</west></LatLonBox>"
[7] "</GroundOverlay></kml>"
```

```
> dev.off()
```

When you open the resulting KML in Google Earth, you will see a display shown in Fig. 5.20. This allows you to orient yourself and make an interpretation of the produced maps. Open the final map in Google Earth and visually explore how many areas next to the populated areas show high concentrations of zinc.

Self-study exercises:

(1.) How many pixels in the `meuse.grid` are available for spatial prediction? (HINT: Number of pixels that do not have missing values for any of the variables.)

(2.) What is the correlation coefficient between maps `dist` and `ahn`? (HINT: use the `cor` method.) Is the default Pearson's parametric correlation appropriate? (HINT: Make a scatterplot of the two maps using the `plot` method. Compute also a non-parametric Spearman correlation.)

(3.) How much of the variation in Zinc does the RK model explains, and which are the most significant predictors? (HINT: Look at the R-square and the `Pr(>|t|)`.)

(4.) Up to which distance from the points are the predictions improved by the ordinary kriging model (rather than just by using an average of all the observations)? (HINT: look at the original variance, and then find at which distance does the semivariance exceeds the original variance.)

(5.) Up to which distance is zinc spatially auto-correlated based on this model? Provide R code to support your answer.

(6.) Is there significant difference in the accuracy of the predictions between OK and RK?

(7.) Up to which distance from the points are the predictions improved by the model? (HINT: At which distance does the regression-kriging prediction variance exceed the global variance?)

(8.) Generate geostatistical simulations of zinc by using only ordinary kriging model and compare your results with Fig 5.12. Which simulation produces higher variability of values (HINT: derive standard deviation and range) — with RK or OK model?

(9.) Randomly split the meuse points in two data sets. Then repeat OK and RK using the same procedure explained in section 5.3.1 and see if the difference in accuracy at validation points is still the same?

(10.) If you had more funds available to locate additional 50 points, and then sample soil again, where would you put them? (HINT: use the sampling optimization algorithm implemented in the intamapInteractive package.)

Further reading:

★ Bivand, R., Pebesma, E., Rubio, V., 2008. **Applied Spatial Data Analysis with R**. Use R Series. Springer, Heidelberg.

★ Ribeiro Jr, P. J., Christensen, O. F. and Diggle, P. J., 2003. geoR and geoRglm: Software for Model-Based Geostatistics. In: Hornik, K. and Leisch, F. and Zeileis, A. (eds) **Proceedings of the 3rd International Workshop on Distributed Statistical Computing** (DSC 2003), Technical University Vienna, pp. 517–524.

★ Kutner, M. H., Nachtsheim, C. J., Neter, J., Li, W. (Eds.), 2004. **Applied Linear Statistical Models**, 5th Edition. McGraw-Hill.

★ Pebesma, E. J., 2004. Multivariable geostatistics in S: the gstat package. Computers & Geosciences 30(7), 683–691.

★ `http://leg.ufpr.br/geoR/` — The geoR package project.

★ `http://www.gstat.org` — The gstat package project.

6

Heavy metal concentrations (NGS)

6.1 Introduction

Now that you have become familiar with basic geostatistical operations in gstat and SAGA, we can proceed with running a mapping exercises with a more complex data set, i.e. a case study that is much closer to real applications. In this exercise we will produce maps of heavy metal concentrations (**HMCs**) for a large area (almost an entire continent), by using an extensive point data set, and a large quantity of auxiliary maps.

Heavy metals occur naturally in rocks and soils, but increasingly higher quantities of them are being released into the environment by anthropogenic activities. Before any solution to the problem of soil heavy metal pollution can be suggested, a distinction needs to be made between natural anomalies and those resulting from human activities. Auxiliary maps such as the ones used in this exercise can be used to show that HMCs are due to industrial activities, toxic releases, or due to the background geology. Such investigations permit one to do an in-depth analysis of the processes that cause the distribution of HMCs, so that also appropriate remediation policies can be selected.

We use the US National Geochemical Survey database (**NGS**), which contains 74,408 samples of 53(+286) attributes sampled over the period from (1979[1]) 2003 to 2008. The original goal of the NGS project was to analyze at least one stream-sediment sample in every 289 km^2 area by a single set of analytical methods across the entire USA. This is one of the most complete and largest geochemical databases in the World. Nevertheless, the most recent version of NGS (v.5) still contains some gaps[2], mostly in western states, although the overall coverage is already >80% of the country. The original data is provided as point maps or as county-level averages. The database is explained in detail in Grossman et al. (2008) and is publicly accessible[3]. Our objective is to map the areas of overall pollution for eight critical heavy metals: arsenic, cadmium, chromium, copper, mercury, nickel, lead and zinc. For practical reasons, we will focus only on the contiguous 48-state area.

The National Geochemical Survey Team has not yet analyzed these data using geostatistical techniques; so far, only maps of individual heavy metal parameters (interpolated using inverse distance interpolation) can be obtained from the NGS website. The maps presented in this exercise were created for demonstration purposes only. The map shown in Fig. 6.11 should be taken with a caution. Its accuracy needs to be assessed using objective criteria.

The advantages of NGS, as compared to e.g. similar European geochemical surveys[4] is that it is: (1) produced using consistent survey techniques; (2) it is spatially representative for all parts of USA; (3) it is extensive; and (4) it is of high quality. The biggest disadvantage of the NGS is that the samples from different media (water, soil and stream sediments) are not equally spread around the whole country — in fact in most cases the two sampling projects do not overlap spatially. Soil samples are only available for about 30% of the whole territory; stream/lake sediment samples have a slightly better coverage. It is unfortunate that different

[1] The NURE samples were collected around 1980.

[2] Mainly Indian reservations and military-controlled areas (see further Fig. 6.2).

[3] `http://tin.er.usgs.gov/geochem/`

[4] European Geological Surveys Geochemical database contains only 1588 georeferenced samples for 26 European countries.

media are not equally represented in different parts of the USA. Mixing of media in this analysis is certainly a serious problem, which we will ignore in this exercise. For more info about NGS data, please contact the USGS National Geochemical Survey Team.

This exercise is largely based on papers previously published in Geoderma (Rodriguez Lado et al., 2009), and in Computers and Geosciences journals (Romić et al., 2007). A more comprehensive introduction to geochemical mapping can be followed in Reimann et al. (2008). A more in-depth discussion about the statistical aspects of dealing with hot spots, skewed distributions and measurement errors can be followed in the work of Moyeed and Papritz (2002) and Papritz et al. (2005). Note also that this is an exercise with intensive computations using a large data set. It is not recommended to run this exercises using a PC without at least 2 GB of RAM and at least 1 GB of free space. Also make sure you regularly save your R script and the R workspace, using the `save.image` method so you do not lose any work.

6.2 Download and preliminary exploration of data

6.2.1 Point-sampled values of HMCs

Open the R script (`NGS.R`) and run line by line. The NGS shapefile can be directly obtained from:

```
> download.file("http://tin.er.usgs.gov/geochem/ngs.zip",
+       destfile=paste(getwd(),"ngs.zip",sep="/"))
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("ngs", j, sep=""), zipname="ngs.zip")
>   file.copy(fname, paste("./ngs", j, sep=""), overwrite=TRUE)
> }
```

To get some info about this map, we can use the `ogrInfo` method:

```
> ogrInfo("ngs.shp", "ngs")
```

```
 Driver: ESRI Shapefile number of rows 74408
 Feature type: wkbPoint with 2 dimensions
 Number of fields: 53
         name type length typeName
 1      LABNO    4     10   String
 2   CATEGORY    4     11   String
 3    DATASET    4     19   String
 4   TYPEDESC    4     12   String
 5      COUNT    0      6  Integer
 6  ICP40_JOB    4      9   String
 7   AL_ICP40    2     14     Real
 8   CA_ICP40    2     14     Real
 ...
 52     SE_AA    2     12     Real
 53     HG_AA    2     14     Real
```

which shows the number of points in the map, and the content of the attribute table: `AL_ICP40` are the measurements of aluminum using the ICP40 method (Inductively Coupled Plasma-atomic emission spectrometry[5]) with values in wt%; `HG_AA` are values of mercury estimated using the AA method (Activation Analysis) in ppm and so on. The file is rather large, which will pose some limitations on the type of analysis we can do. We can now import the shapefile to R:

```
> ngs <- readOGR("ngs.shp", "ngs")
```

The samples that compose the NGS come from five main media (`CATEGORY`):

```
> summary(ngs$CATEGORY)
```

[5] http://tin.er.usgs.gov/geochem/doc/analysis.htm

```
      NURE NURE SIEVED       PLUTO
     45611        4754        1453
      RASS       STATE   SW-ALASKA
      1755       20126         709
```

The majority of points belongs to the National Uranium Resource Evaluation (`NURE`) and the `STATE` program, both were collected in the period (1979) 2003–2008. Following the NGS documentation[6], these points are in geographical coordinates with the NAD27 datum and Clarke 1866 ellipsoid, which we can set in R as:

```
> proj4string(ngs) <- CRS("+proj=longlat +ellps=clrk66 +datum=NAD27 +no_defs")
```

We are interested in mapping the following nine[7] variables:

```
> HMC.list <- c("AS_ICP40", "CD_ICP40", "CR_ICP40", "CU_ICP40",
+     "NI_ICP40", "ZN_ICP40", "AS_AA", "HG_AA", "PB_ICP40")
# short names:
> HM.list <- c("As","Cd","Cr","Cu","Ni","Zn","As2","Hg","Pb")
```

Let us first take a look at the properties of the data, i.e. what the range of values is, and how skewed the variables are. We can look at the first variable on the list:

```
> summary(ngs@data[,HMC.list[1]])
```

```
     Min. 1st Qu.  Median    Mean 3rd Qu.  Max.    NA's
   -40.00  -10.00  -10.00   -1.82   10.00  5870.00 3164.00
```

which shows that there are also negative values in the table — obviously artifacts. If we go back to the original documentation[8], we can notice that negative values mean "*measured, but below detection limit*". Masking the negative values with `NA` will not be correct because these are really zero or close to zero measurements. This would make a huge difference for further analysis because the proportion of such measurements can be significant. The solution is to automatically replace all negative (and values below the detection limit) values with half the detection limit (Reimann et al., 2008, p.23–24):

```
# "AS_ICP40"
> ngs@data[,HMC.list[1]] <- ifelse(ngs@data[,HMC.list[1]]<2,
+     abs(ngs@data[,HMC.list[1]])/2, ngs@data[,HMC.list[1]])
# "CD_ICP40"
> ngs@data[,HMC.list[2]] <- ifelse(ngs@data[,HMC.list[2]]<1, 0.2,
+     ngs@data[,HMC.list[2]])
> for(hmc in HMC.list[-c(1,2)]){
>   ngs@data[,hmc] <- ifelse(ngs@data[,hmc]<0, abs(ngs@data[,hmc])/2, ngs@data[,hmc])
> }
# check the summary statistics now:
> summary(ngs@data[,HMC.list[1]])
```

```
      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
     2.500     5.000     5.000     8.843    10.000  5870.000  3164.000
```

Now that we have filtered negative values, we can take a look at the histograms and cross-correlations between HMCs. We assume that all distributions are skewed and hence use in further analysis the log-transformed values (Fig. 6.1):

```
> HMC.formula <- as.formula(paste(" ~ ", paste("log1p(", HMC.list, ")",
+      collapse="+"), sep=""))
> HMC.formula ~ log1p(AS_ICP40) + log1p(CD_ICP40) + log1p(CR_ICP40) +
+     log1p(CU_ICP40) + log1p(NI_ICP40) + log1p(ZN_ICP40) + log1p(AS_AA) +
+     log1p(HG_AA) + log1p(PB_ICP40)
> pc.HMC <- prcomp(HMC.formula, scale=TRUE, ngs@data)
> biplot(pc.HMC, arrow.len=0.1, xlabs=rep(".", length(pc.HMC$x[,1])),
+      main="PCA biplot", xlim=c(-0.04,0.02), ylim=c(-0.03,0.05), ylabs=HM.list)
```

[6] See the complete metadata available in the attached document: `ofr-2004-1001.met`.

[7] We will make a total of eight maps in fact. Metal Arsenic is measured by using two laboratory methods.

[8] see also `http://tin.er.usgs.gov/geochem/faq.shtml`

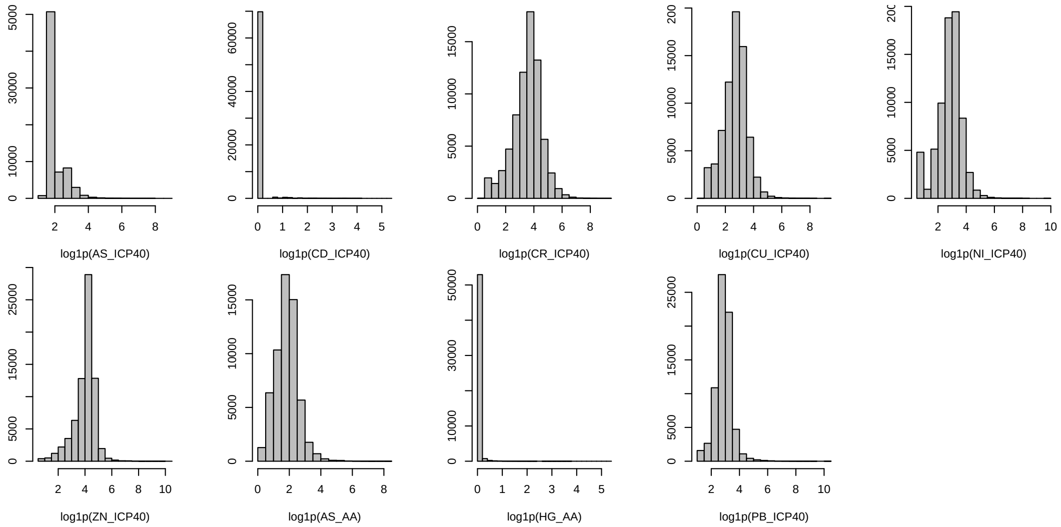


Fig. 6.1: Histograms for log-transformed target variables (HMCs). Note that variables Cd and Hg show skewness even after the transformation.

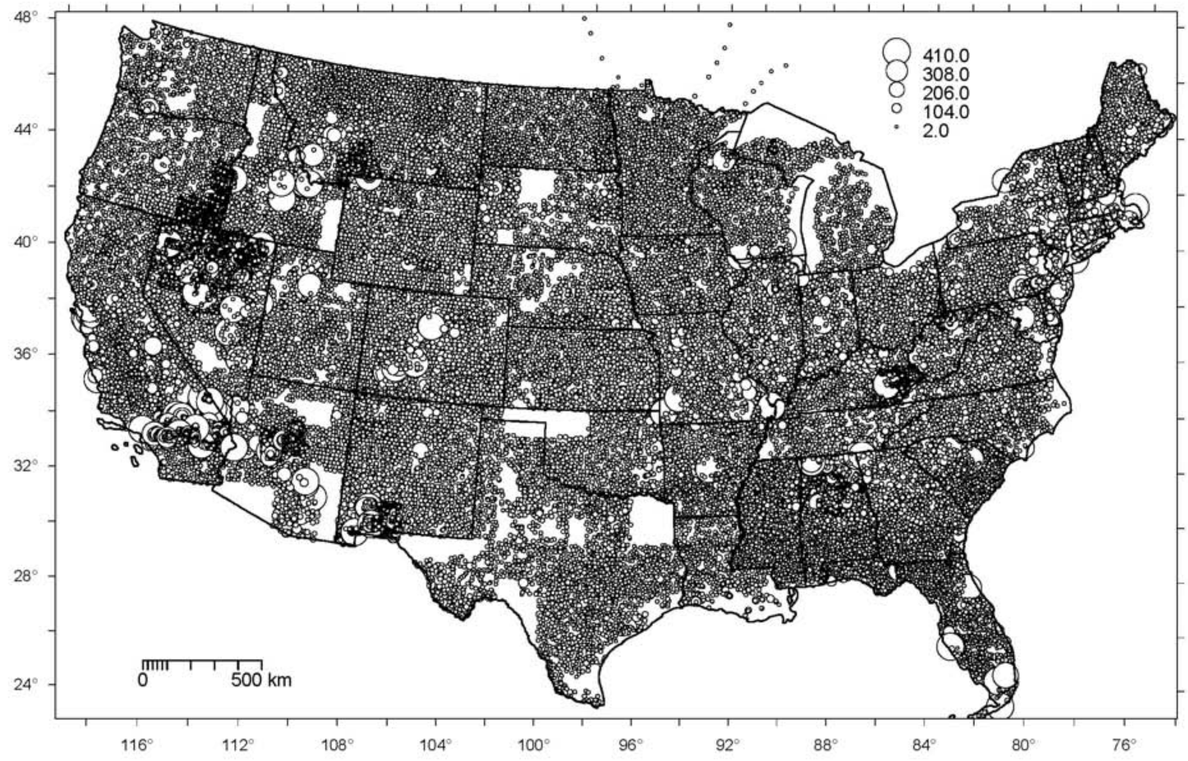


Fig. 6.2: Sampling locations and values of Pb (ppm) based on the NGS data set. Notice the large areas completely unsampled (Indian reservations and military controlled areas), while on the other hand, many states and/or regions have been really densely sampled. Visualized in ILWIS GIS.

The biplot (Fig. 6.3) shows two interesting things: (1) it appears that there are two main clusters of values — Zn, Cu, Ni, Cr, and As, Pb, Hg, Cd; (2) HMCs in both clusters are positively correlated, which means that if e.g. values of Zn increase, so will the values of Cu, Ni, Cr. Similar properties can be noticed with the HMCs in Europe (Rodriguez Lado et al., 2009).

If we look at individual correlations we can see that the most correlated heavy metals are: Cu, Ni and Zn (r=0.76), Cr and Zn (r=0.62), As and Zn (r=0.56). Note also that, at this stage, because the density of points is so high, and because distributions of the target variables are skewed, it is hard to see any spatial pattern in the HMC values (see Fig. 6.2).

6.2.2 Gridded predictors

A sound approach to geostatistical mapping is to first consider all factors that can possibly control the spatial distribution of the feature of interest, and then try to prepare maps that can represent that expert knowledge (Koptsik et al., 2003; Pebesma, 2006). For example, we can conceptualize that the distribution of HMCs is controlled by the a number of environmental and anthropogenic factors: (a) geology, (b) continuous industrial activities — traffic, heating plants, chemical industry and airports, (c) historic land use and mining activities, and (d) external factors and random events — spills and accidents, transported and wind-blown materials.

Because for USA a large quantity of GIS layers[9] is available from the USGS[10] or the National Atlas[11], we have at our disposal a wide variety of predictors:

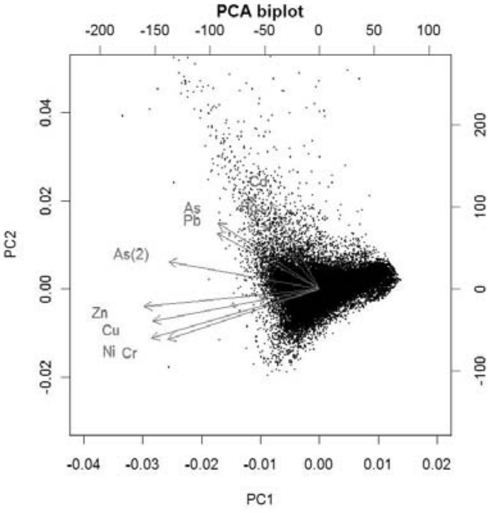


Fig. 6.3: HMCs shown using a PCA biplot.

Urbanization level — Urbanization level can be represented by using images of lights at night (Fig. 6.4; `nlights03.asc`). These are typically highly correlated with industrial activity. The assumption is that HMCs will increase in areas of high urbanization.

Density of traffic — Maps of main roads and railroads can be used to derive density[12] of transportation infrastructure (`sdroads.asc`). In addition, we can use the kernel density of airport traffic (`dairp.asc`), derived using the total enplanements (ranging from few hundreds to >30 million) as weights.

Density of mineral operations — The National Atlas contains an inventory of all major mineral operations (ferrous and nonferrous metal mines). These layers can be merged to produce a kernel density map (`dmino.asc`) showing overall intensity of mineral exploration. In addition, we can also consider the type of mineral exploration (`minotype.asc`; 67 classes), which should help us explain the local hot spots for different heavy metals.

Density of Earthquakes — The magnitude of significant United States Earthquakes (1568–2004) can be used to derive the overall intensity of earthquakes (`dquksig.asc`). We are not certain if this feature controls the distribution of HMCs, but it quantifies geological activities, *ergo* it could also help us explain background concentrations.

Industrial pollutants — The pan-American Environmental Atlas of pollutants (35,000 industrial facilities in North America that reported on releases or transfers of pollutants in 2005) can be used to derive the density of toxic releases (`dTRI.asc`, Fig. 6.4). This feature should likewise be able to explain local hot-spots for heavy metals.

Geological stratification — Geological map at scale 1:1,000,000 (`geomap.asc`; 39 classes) is available from USGS (Fig. 6.4). We can assume that, within some geological units, concentrations will be more homogenous.

[9] See also section 7.2.1.

[10] `http://www.usgs.gov/pubprod/data.html`

[11] `http://nationalatlas.gov/`

[12] See function `Segment density` in ILWIS GIS.

Geomorphological characteristics — From the global Digital Elevation Model (globedem.asc), a number of DEM parameters of interest can be derived: Topographic Wetness Index (twi.asc), visible sky (vsky.asc) and wind effect index (winde.asc). This can help us model HMCs carried by wind or water.

Green biomass — Green biomass can be represented with the Global Biomass Carbon Map that shows carbon density in tons of C ha^{-1} (gcarb.asc, Fig. 6.4). We assume that areas of high biomass amortize pollution by HMCs, and are inversely correlated with HMCs.

Wetlands areas — Because the HMCs have been sampled in various mediums (streams, rivers, lakes, soils, rocks etc.), we also need to use the map of lakes and wetlands (glwd31.asc) to distinguish eventual differences.

The maps listed above can be directly obtain from the data repository, and then extracted to a local directory:

```
> download.file("http://spatial-analyst.net/book/system/files/usgrids5km.zip",
+     destfile=paste(getwd(), "usgrids5km.zip", sep="/"))
> grid.list <- c("dairp.asc", "dmino.asc", "dquksig.asc", "dTRI.asc", "gcarb.asc",
+     "geomap.asc", "globedem.asc", "minotype.asc", "nlights03.asc", "sdroads.asc",
+     "twi.asc", "vsky.asc", "winde.asc", "glwd31.asc")
> for(j in grid.list){
>   fname <- zip.file.extract(file=j, zipname="usgrids5km.zip")
>   file.copy(fname, paste("./", j, sep=""), overwrite=TRUE)
> }
```

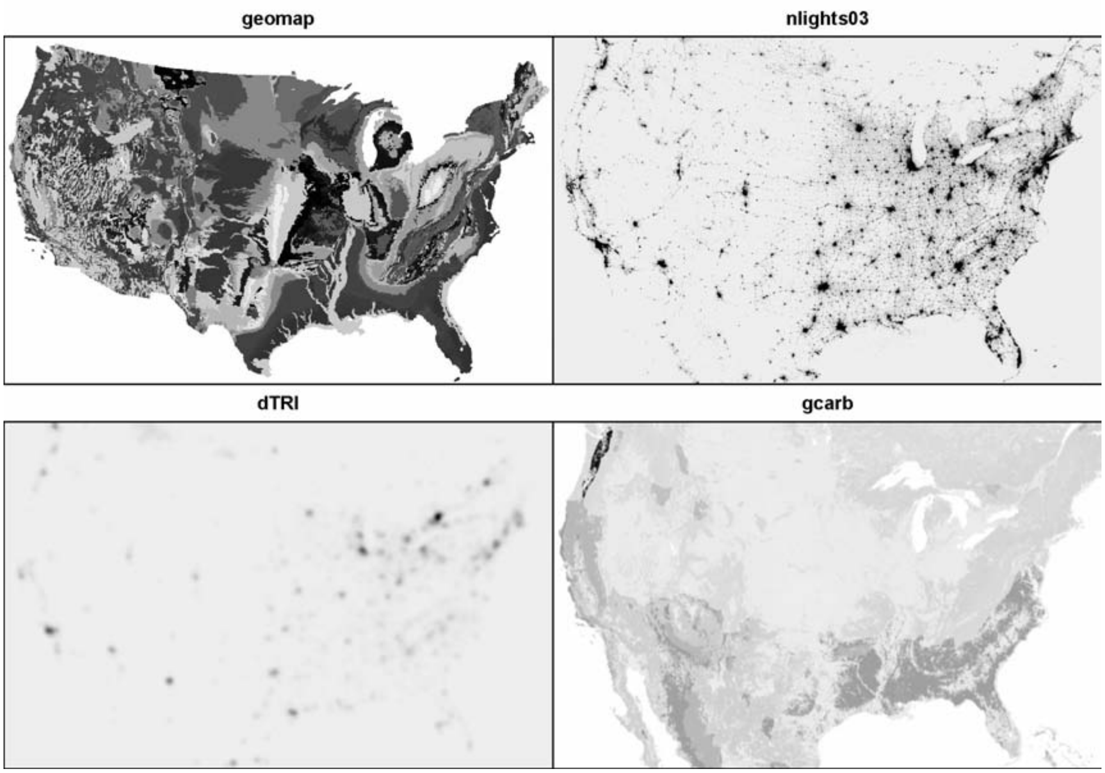


Fig. 6.4: Examples of environmental predictors used to interpolate HMCs: geomap — geological map of US; nlights03 — lights and night image for year 2003; dTRI — kernel density of reported toxic releases; gcarb — biomass carbon map. Visualized using the image method of the adehabitat package.

In total, we have at our disposal 14 gridded maps, possibly extendable to 130 grids if one also includes the indicators (geomap.asc, minotype.asc, and glwd31.asc maps are categories). Note also that, although all layers listed above are available in finer resolutions (1 km or better), for practical reasons we will work with the 5 km resolution maps.

The gridded maps are projected in the Albers equal-area projection system:

```
# read grids into R:
> gridmaps <- readGDAL(grid.list[1])
> names(gridmaps)[1] <- sub(".asc", "", grid.list[1])
> for(i in grid.list[-1]) {
>   gridmaps@data[sub(".asc", "", i[1])] <- readGDAL(paste(i))$band1
> }
> AEA <- "+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96 +x_0=0 +y_0=0
+         +ellps=GRS80 +datum=NAD83 +units=m +no_defs"
> proj4string(gridmaps) <- CRS(AEA)
```

which is often used to display the whole North American continent.

In the same zip file (`usgrids5km.zip`), you will also find a number of ASCII files with the extension `*.rdc`[13]. The `*.rdc` file carries the complete layer metadata, which allows easy access and editing. This is an example of a description file for the Global Lakes and Wetlands map (categorical variable):

```
file format : Arc/Info ASCII Grid
file title  : glwd31.asc
last update : 12.07.2009
producer    : T. Hengl
lineage     : The GLWD31 1 km grid was resampled to 5 km grid.
data type   : byte
file type   : ASCII
columns     : 940
rows        : 592
meas. scale : categorical
description : Global Lakes and Wetlands
proj4string : +proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96
+x_0=0 +y_0=0 +ellps=GRS80 +datum=NAD83 +units=m +no_defs
ref. system : projected
ref. units  : meters
unit dist.  : 1
min. X      : -2405000
max. X      : 2295000
min. Y      : 260000
max. Y      : 3220000
pos'n error : 1000
resolution  : 5000
min. value  : 1
max. value  : 12
display min : 1
display max : 12
value units : factor
value error : unspecified
flag value  : -1
flag def'n  : unavailable data
src. English: Global Lakes and Wetlands v3.1
src. URL    : http://www.worldwildlife.org/science/data/item1877.html
src. scale  : 1000 m
src. refs   : Lehner, B., Doll, P., 2004. Development and validation of a global
 database of lakes, reservoirs and wetlands. Journal of Hydrology 296(1-4): 1-22.
src. date   : 2003
src. owner  : WWF; GLWD is available for non-commercial scientific, conservation,
 and educational purposes.
legend cats : 13
category   0: other classes
```

[13] Idrisi GIS format raster (image) documentation file.

```
category   1: Lake
category   2: Reservoir
category   3: River
category   4: Freshwater Marsh, Floodplain
category   5: Swamp Forest, Flooded Forest
category   6: Coastal Wetland
category   7: Pan, Brackish/Saline Wetland
category   8: Bog, Fen, Mire (Peatland)
category   9: Intermittent Wetland/Lake
category  10: 50-100% Wetland
category  11: 25-50% Wetland
category  12: Wetland Complex (0-25% Wetland)
```

such metadata will become important once we start doing interpretation of the results of model fitting — we might need to check the original document describing how was the map produced, what each legend category means etc.

6.3 Model fitting

6.3.1 Exploratory analysis

Before we run any geostatistical predictions, we can simply generate a raster map showing the general spatial pattern of the target variable by using a mechanical interpolator. Because this is a data set with a fairly dense sampling density, the easiest way to generate a complete map from points is to use the "*Close gap*" operation in SAGA. First, we need to convert point data to a raster map:

```
# prepare the mask map:
> rsaga.esri.to.sgrd(in.grids="geomap.asc",
+      out.sgrds="geomap.sgrd", in.path=getwd())
# reproject to the local coordinate system:
> ngs.aea <- spTransform(ngs, CRS(AEA))
# write each element and convert to a raster map:
> for(hmc in HMC.list){
>   writeOGR(subset(ngs.aea, !is.na(ngs.aea@data[,hmc]))[hmc],
+        paste(hmc, ".shp", sep=""), paste(hmc), "ESRI Shapefile")
>   rsaga.geoprocessor(lib="grid_gridding", module=0, param=list(GRID=paste(hmc,
+       ".sgrd",sep=""), INPUT=paste(hmc, ".shp",sep=""), FIELD=0, LINE_TYPE=0,
+       USER_CELL_SIZE=cell.size, USER_X_EXTENT_MIN=gridmaps@bbox[1,1]+cell.size/2,
+       USER_X_EXTENT_MAX=gridmaps@bbox[1,2]-cell.size/2,
+       USER_Y_EXTENT_MIN=gridmaps@bbox[2,1]+cell.size/2,
+       USER_Y_EXTENT_MAX=gridmaps@bbox[2,2]-cell.size/2))
# close gaps (linear interpolation):
>   rsaga.geoprocessor(lib="grid_tools", module=7, param=list(INPUT=paste(hmc,
+       ".sgrd", sep=""), MASK="geomap.sgrd", RESULT=paste(hmc, ".sgrd", sep="")))
> }
```

which will produce the maps shown below (Fig. 6.5). Although these maps seem to be rather complete, they can also be very misleading because we have produced them by completely ignoring landscape features, geology, anthropogenic sources, heterogeneity in the sampling density, spatial auto-correlation effects and similar.

6.3.2 Regression modeling using GLM

Before we can correlate HMCs with environmental predictors, we need to obtain values of predictor grids at sampling locations:

```
> ngs.ov <- overlay(gridmaps, ngs.aea)
> ngs.ov@data <- cbind(ngs.ov@data, ngs.aea@data)
```

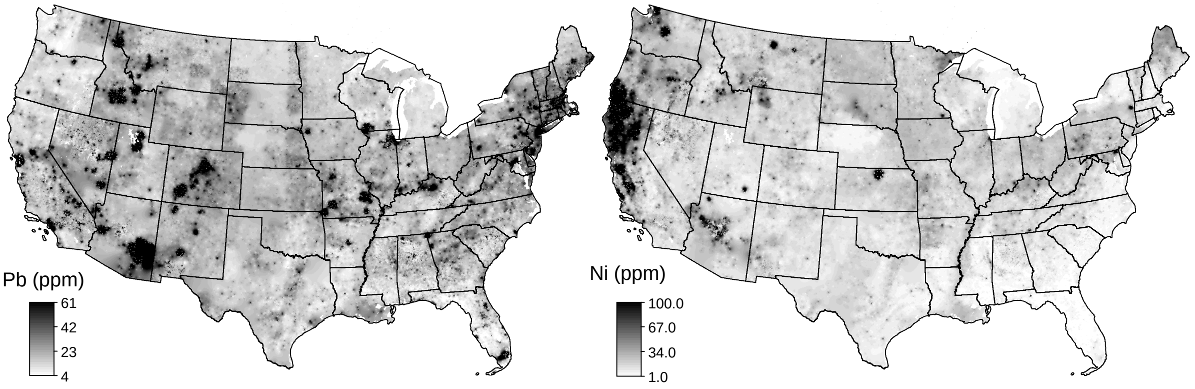


Fig. 6.5: General spatial pattern in values of Pb and Ni generated using interpolation from gridded data (close gap operation in SAGA). Compare further with the maps generated using regression-kriging (as shown in Fig. 6.9).

which creates a matrix with 9+14 variables. This allows us to do some preliminary exploration, e.g. to see for example how much the geological mapping units differ for some HMCs. Let us focus on Pb:

```
> boxplot(log1p(ngs.ov@data[,HMC.list[9]]) ~ ngs.ov$geomap.c,
+       col=grey(runif(levels(ngs.ov$geomap.c))), ylim=c(1,8))
```

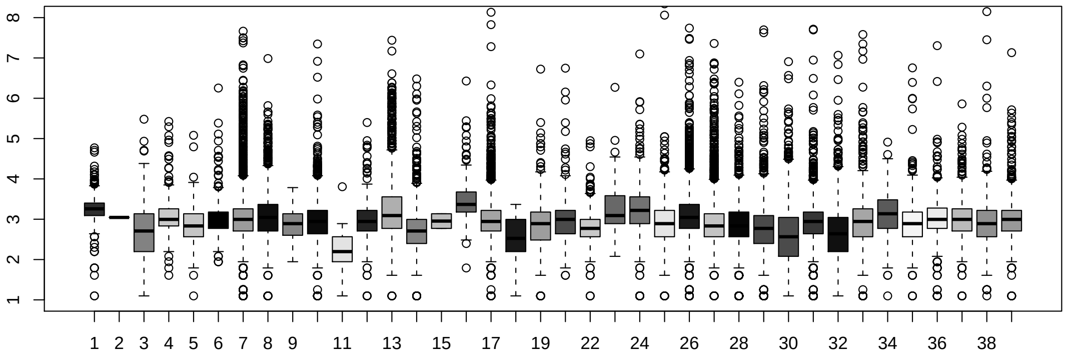


Fig. 6.6: Boxplot showing differences in values of Pb for different geological mapping units (39).

which shows that only a few units (e.g. "1", "11", "16") have distinctly different concentrations of Pb. Some units do not have enough points for statistical analysis, which poses a problem:

```
# how many units need to be masked:
> summary(summary(ngs.ov$geomap.c)<5)

    Mode   FALSE    TRUE    NA's
 logical      37       2       0
```

We can run the same check for the other two categorical maps:

```
> summary(ngs.ov$minotype.c)

     0      1      2      3      4      5     65     66  NA's
 67245      4      0      6     12     13      9      0  6749


> summary(summary(ngs.ov$minotype.c)<5)
```

```
   Mode   FALSE    TRUE    NA's
logical      27      36       0
```

which shows that there are many units that do not have enough observations for statistical analysis and need to be masked out[14]. We need to do that also with the original grids, because it is important that the regression matrix and the prediction locations contain the same range of classes. We can replace the classes without enough points with dominant classes in the map by using:

```
# determine inappropriate classes (geomap):
> geomap.c.fix <- as.numeric(attr(sort(summary(ngs.ov$geomap.c))
+       [1:sum(summary(ngs.ov$geomap.c)<5)], "names"))
> geomap.c.fix

 [1]  2 15

> geomap.c.dom <- as.numeric(attr(sort(summary(gridmaps$geomap.c
+       [!is.na(gridmaps$geomap.c)]), decreasing=TRUE)[1], "names"))
# replace the values using the dominant class:
> for(j in geomap.c.fix){
>   gridmaps$geomap <- ifelse(gridmaps$geomap==j, geomap.c.dom, gridmaps$geomap)
> }
> gridmaps$geomap.c <- as.factor(gridmaps$geomap)
# repeat the same for minotype and glwd31...
```

and we can check that the classes with insufficient observations have been replaced:

```
# update the regression matrix:
> ngs.ov <- overlay(gridmaps, ngs.aea)
> ngs.ov@data <- cbind(ngs.ov@data, ngs.aea@data)
> summary(summary(ngs.ov$geomap.c)<5)

   Mode   FALSE    NA's
logical      37       0
```

Next we can fit a regression model for our sample variable and generate predictions at all grid nodes. Because the values of HMCs are skewed, we will be better off if we fit a GLM model to this data, i.e. using a *Poisson* family with a log link function:

```
> Pb.formula <- as.formula(paste(HMC.list[9], "~", paste(sub(".asc", "",
+      grid.list[-c(6,8,14)]), collapse="+"), "+geomap.c+glwd31.c"))
> Pb.formula

 PB_ICP40 ~ dairp + dmino + dquksig + dTRI + gcarb + globedem +
     nlights03 + sdroads + twi + vsky + winde + geomap.c + glwd31.c

# fit the model using GLM:
> Pb.lm <- glm(Pb.formula, ngs.ov@data, family=poisson(link="log"))
# predict values at new locations:
> Pb.trend <- predict(Pb.lm, newdata=gridmaps, type="link", na.action=na.omit, se.fit=TRUE)
```

note that the result of prediction is just a data frame with two columns: (1) predicted values in the transformed scale[15] (controlled with `type="link"`, (2) model prediction error (set with `se.fit=TRUE`):

```
> str(Pb.trend)

 List of 3
  $ fit           : Named num [1:314719] 3.55 3.75 3.84 3.71 3.62 ...
   ..- attr(*, "names")= chr [1:314719] "10429" "10430" "10431" "10432" ...
  $ se.fit        : Named num [1:314719] 0.0065 0.00557 0.00529 0.00509 0.00473 ...
   ..- attr(*, "names")= chr [1:314719] "10429" "10430" "10431" "10432" ...
  $ residual.scale: num 1
```

[14] Recall that, by a rule of thumb, we should have at least 5 observations per mapping unit.

[15] We use the transformed scale because we will further sum the interpolated residuals, which are also in the transformed scale.

which means that the coordinates of the grids nodes are not attached any more, and we cannot really visualize or export this data to a GIS. We can reconstruct a gridded map because the grid node names are still contained in the attribute field. This will take several processing steps:

```
# get the coordinates of the original grid:
> pointmaps <- as(gridmaps["globedem"], "SpatialPointsDataFrame")
> sel2 <- as.integer(attr(Pb.trend$fit, "names"))
> rk.Pb <- data.frame(X=pointmaps@coords[sel2,1],
+      Y=pointmaps@coords[as.integer(attr(Pb.trend$fit, "names")),2],
+      HMC=Pb.trend$fit, HMC.var=Pb.trend$se)
> coordinates(rk.Pb) <- ~ X+Y
> gridded(rk.Pb) <- TRUE
# resample to the original grid:
> write.asciigrid(rk.Pb["HMC"], "tmp.asc", na.value=-999)
> rsaga.esri.to.sgrd(in.grids="tmp.asc", out.sgrds="tmp.sgrd", in.path=getwd())
# create an empty grid:
> rsaga.geoprocessor(lib="grid_tools", module=23,
+     param=list(GRID="tmp2.sgrd", M_EXTENT=0,
+     XMIN=gridmaps@bbox[1,1]+cell.size/2, YMIN=gridmaps@bbox[2,1]+cell.size/2,
+     NX=gridmaps@grid@cells.dim[1], NY=gridmaps@grid@cells.dim[2],
+     CELLSIZE=cell.size))
# add decimal places:
> rsaga.geoprocessor("grid_calculus", module=1,
+     param=list(INPUT="tmp2.sgrd", RESULT="Pb_trend_GLM.sgrd", FORMUL="a/100"))
# resample the target grid:
> rsaga.geoprocessor(lib="grid_tools", module=0, param=list(INPUT="tmp.sgrd",
+     GRID="Pb_trend_GLM.sgrd", KEEP_TYPE=FALSE, METHOD=2, SCALE_DOWN_METHOD=0,
+     GRID_GRID="Pb_trend_GLM.sgrd"))
> rsaga.sgrd.to.esri(in.sgrds="Pb_trend_GLM.sgrd", out.grids="Pb_trend_GLM.asc",
+     out.path=getwd())
```

We can quickly check whether the prediction model is efficient in reflecting the original distribution of sampled Pb values:

```
# compare the distributions (95% range of values):
> round(quantile(expm1(Pb.trend$fit), c(.05,.95), na.rm=TRUE), 0)
```

```
 5% 95%
 12  46
```

```
# samples:
> quantile(ngs.ov@data[,HMC.list[9]], c(.05,.95), na.rm=TRUE)
```

```
 5% 95%
  6  41
```

```
# precision:
> sd(residuals(Pb.lm))
```

```
[1] 6.65979
```

If we zoom into the original data, we can notice that there are very few of the original point data that are extremely high (over 5000 times higher than the mean value), most of the values are in the range 6–41 ppm. If we plot the predicted and measured values next to each other, we can notice that the model will have serious problems in predicting both high and low values. There is noticable scatter around the regression line, which also means that the residuals will be significant.

At this stage, it also useful to explore some individual plots between the target variable and predictors. In the case of mapping Pb, it seems that only `dTRI.asc` shows a clear relationship with the target variable, all other correlation plots are less distinct (Fig. 6.7). The good news is that majority of correlations reflect our expectations in qualitative terms: higher concentrations of Pb are connected with higher density of toxic releases and higher industrial activity.

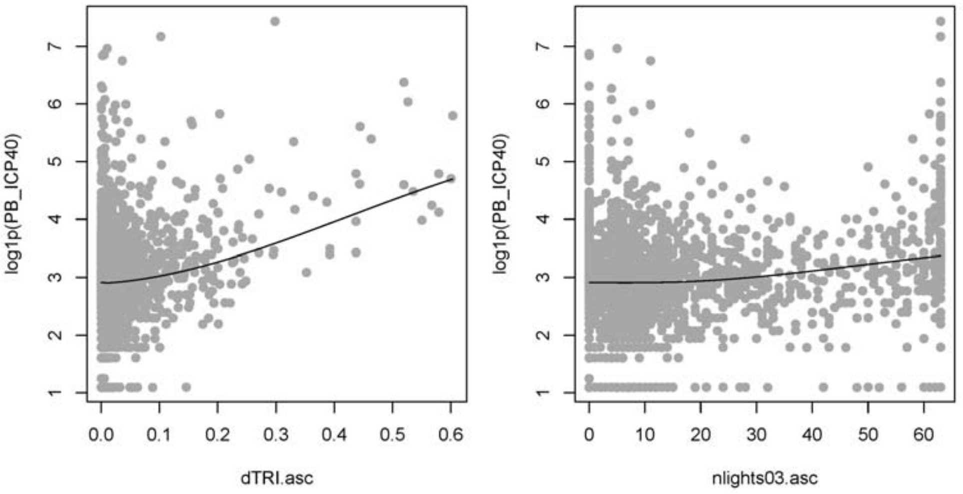


Fig. 6.7: Correlation plots Pb versus some significant predictors: density of toxic release accidents (`dTRI.asc`), and lights at night image (`nlights03.asc`).

6.3.3 Variogram modeling and kriging

Now that we have predicted the trend part of the model, we can proceed with interpolating the residuals. We will also interpolate the residuals in the transformed scale, and not in the response scale, which means that we need to derive them:

```
> residuals.Pb <- log1p(Pb.lm$model[,HMC.list[j]])-log1p(fitted.values(Pb.lm))
```

An important check we need to make is to see that the residuals are normally distributed:

```
> hist(residuals.Pb, breaks=25, col="grey")
# residuals are normally distributed!
```

which is a requirement to interpolate this variable using ordinary kriging.

Fitting a variogram model with >50,000 points in gstat is computationally intensive and would take significant time, especially if we would like to do it using global search radius. Instead, we can speed up the processing by: (a) limiting the search radius, (b) sub-setting the points randomly[16]. To estimate the mean shortest distance between points we can use spatstat package:

```
> library(spatstat)
> ngs.ppp <- as(ngs.aea[1], "ppp")
> boxplot(dist.ngs <- nndist(ngs.ppp), plot=F)$stats

          [,1]
 [1,]     0.000
 [2,]  1020.381
 [3,]  3021.972
 [4,]  7319.961
 [5,] 16766.662

> search.rad <- 2*boxplot(dist.ngs <- nndist(ngs.ppp), plot=F)$stats[5]
```

which shows that the mean shortest distance to the nearest point is about 3 km, none of the points is >17 km away from the first neighbor. To be on the safe side, we can limit the search radius to two times the highest `nndist` i.e. 34 km in this case.

Next, we can prepare a point map with residuals, and randomly sub-sample the data set to 20% of its original size:

[16] Assuming that a large part of variation has already been explained by the GLM model, we can be less accurate about fitting the variogram.

```
> sel <- as.integer(attr(Pb.lm$model, "na.action"))
> res.Pb <- data.frame(X=coordinates(ngs.ov[-sel,])[,1],
+        Y=coordinates(ngs.ov[-sel,])[,2], res=residuals.Pb)
# mask out NA values:
> res.Pb <- subset(res.Pb, !is.na(res.Pb$res))
> coordinates(res.Pb) <- ~ X+Y
> proj4string(res.Pb) <- CRS(AEA)
# sub-sample to 20%!
> res.Pb.s <- res.Pb[runif(length(res.Pb@data[[1]]))<0.2,]
```

so that fitting of the variogram will go much faster:

```
> var.Pb <- variogram(res ~ 1, data=res.Pb.s, cutoff=34000)
> rvgm.Pb <- fit.variogram(var.Pb, vgm(nugget=var(res.Pb$res, na.rm=TRUE)/5,
+      model="Exp", range=34000, psill=var(res.Pb$res, na.rm=TRUE)))
> plot(var.Pb, rvgm.Pb, plot.nu=F, pch="+", cex=2,
+      col="black", main="Vgm for Pb residuals")
```

The variogram shows that the feature is correlated up to the distance of about 10 km; about 50% of sill variation (nugget) we are not able to explain. Use of GLM *Poisson* model is beneficial for further geostatistical modeling — the residuals have a symmetrical distribution; the final predictions will also follow a similar distribution, i.e. they will maintain hot-spot locations, which might have been otherwise smoothed-out if a simple linear regression was used.

To speed up the interpolation[17], we use the SAGA geostatistics module:

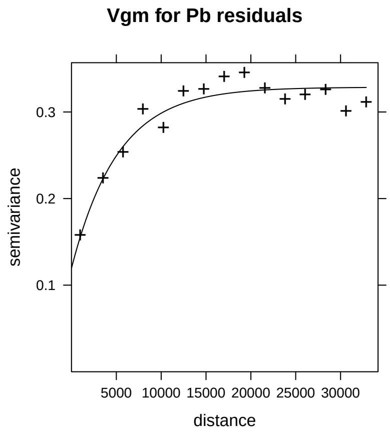


Fig. 6.8: Results of variogram fitting for the Pb GLM-residuals (log-transformed).

```
# export to a shapefile:
> writeOGR(res.Pb, "Pb_res.shp", "Pb_res",
+        "ESRI Shapefile")
# Ordinary kriging in SAGA:
> rsaga.geoprocessor(lib="geostatistics_kriging",
+   module=5, param=list(GRID="Pb_res_OK.sgrd",
+   SHAPES="Pb_res.shp", BVARIANCE=F, BLOCK=F,
+   FIELD=1, BLOG=F, MODEL=1, TARGET=0,
+   NPOINTS_MIN=10, NPOINTS_MAX=60,
+   NUGGET=rvgm.Pb$psill[1], SILL=rvgm.Pb$psill[2],
+   RANGE=rvgm.Pb$range[2],
+   MAXRADIUS=3*search.rad, USER_CELL_SIZE=cell.size,
+   USER_X_EXTENT_MIN=gridmaps@bbox[1,1]+cell.size/2,
+   USER_X_EXTENT_MAX=gridmaps@bbox[1,2]-cell.size/2,
+   USER_Y_EXTENT_MIN=gridmaps@bbox[2,1]+cell.size/2,
+   USER_Y_EXTENT_MAX=gridmaps@bbox[2,2]-cell.size/2))
```

Finally, we can combine the two maps (predicted trend and interpolated residuals) to produce the best estimate of the Pb values (Fig. 6.9):

```
# sum the regression and residual part:
> rsaga.sgrd.to.esri(in.sgrds="Pb_rk.sgrd", out.grids="Pb_rk.asc", out.path=getwd())
> gridmaps@data[,"Pb_rk"] <- exp(readGDAL("Pb_rk.asc")$band1)
> spplot(gridmaps["Pb_rk"], col.regions=grey(rev((1:59)^2/60^2)), at=seq(4,250,5))
```

[17] The data set consists of >50,000 points! Even if we are using a small search radius, this data set will always take a significant amount of time to generate predictions.

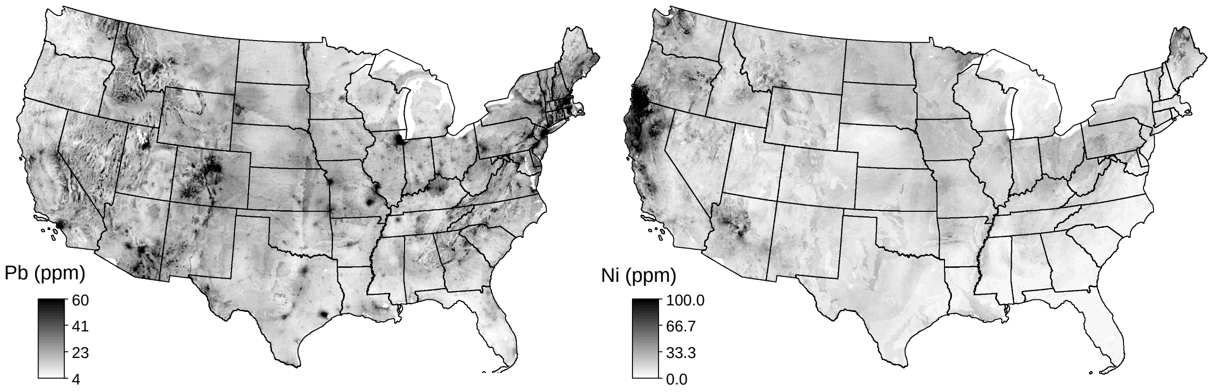


Fig. 6.9: Distribution of Pb and Ni predicted using regression-kriging. Note that many local hot-spots from Fig. 6.5 have been now smoothed out by the kriging algorithm.

6.4 Automated generation of HMC maps

Now that we have become familiar with the geostatistical steps, i.e. now that we have tested different methods and tidy up the R code, we can pack all the steps together. The results of fitting we will save as lists, so that we can review them later on; all other temporary files we can recycle[18]:

```
# generate empty lists:
> formula.list <- as.list(rep(NA, length(HMC.list)))
> vgm.list <- as.list(rep(NA, length(HMC.list)))
> vgmplot.list <- as.list(rep(NA, length(HMC.list)))
> for(j in 1:length(HMC.list)){
# fit a GLM:
>   formula.list[[j]] <- as.formula(paste(HMC.list[j], "~",
+      paste(sub(".asc", "", grid.list[-c(6,8,14)]), collapse="+"), "+geomap.c+glwd31.c"))
>   glm.HMC <- glm(formula.list[[j]], ngs.ov@data, family=poisson(link="log"))
...
# sum the regression and residual part:
>   rsaga.geoprocessor("grid_calculus", module=1,
+     param=list(INPUT=paste(HM.list[j], "_trend_GLM.sgrd", ";", HM.list[j],
+     "_res_OK.sgrd", sep=""), RESULT=paste(HM.list[j], "_rk.sgrd", sep=""), FORMUL="a+b"))
>   rsaga.sgrd.to.esri(in.sgrds=paste(HM.list[j], "_rk.sgrd", sep=""),
+     out.grids=paste(HM.list[j], "_rk.asc", sep=""), out.path=getwd())
>   gridmaps@data[,paste(HM.list[j], "_rk", sep="")] <- exp(readGDAL(paste(HM.list[j],
+     "_rk.asc", sep=""))$band1)
>   write.asciigrid(gridmaps[paste(HM.list[j], "_rk", sep="")],
+     paste(HM.list[j], "_rk.asc", sep=""), na.value=-1)
> }
```

In summary, the script follows previously described steps, namely:

(1.) Fit the GLM using the regression matrix. Derive the residuals (log-scale) and export them to a shapefile.

(2.) Predict values using the fitted GLM. Convert the predictions to the same grid as the predictor maps.

(3.) Fit the variogram model for residuals. Save the fitted variogram parameters and the variogram plot.

(4.) Interpolate the residuals using ordinary kriging in SAGA.

(5.) Sum the predicted trend (GLM) and residuals (OK) and import the maps back into R.

[18] This will take a lot of your memory, hence consider using `gc()` to release some memory from time to time. It is especially important to recycle the results of GLM modeling. The resulting GLM object will often take a lot of memory because it makes copies of the original data set, masked observations and observations used to build the model.

(6.) Back-transform the values to the original scale; export the map to a GIS format.

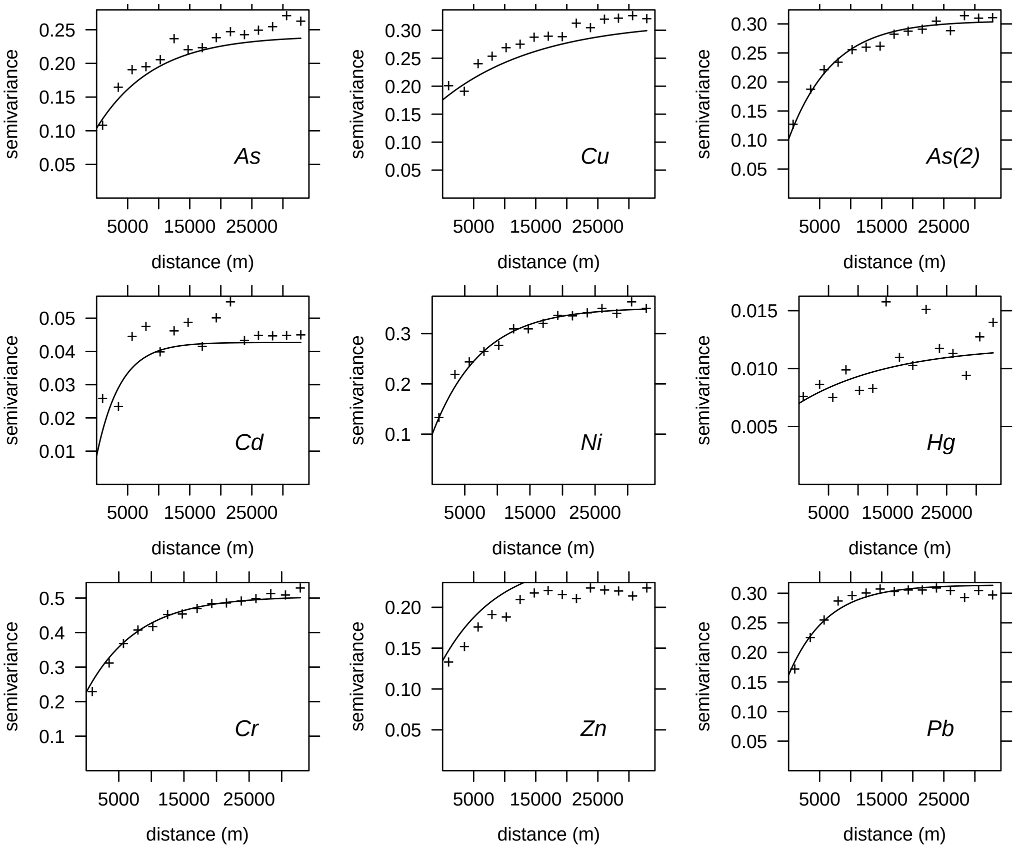


Fig. 6.10: Variograms fitted for GLM residuals.

To review the results of model fitting we can best look at the fitted variograms (Fig. 6.10). If the variograms are stable and fitted correctly, and if they do not exceed the physical range of values, we can be confident that the predictions will be meaningful. In this case, we can see that all variograms have a standard shape, except for Hg, which seems to show close to pure nugget effect. We can repeat the variogram fitting *by-eye* for this HMC, and then re-interpolate the data, at least to minimize artifacts in the final map. Note also that the nugget variation is significant for all variables.

The final predictions for various HMCs can be used to extract the principal components, i.e. reduce eight maps to two maps. Recall from Fig. 6.3 that there are basically two big groups of HMCS: Zn, Cu, Ni, Cr; and As, Pb, Hg, Cd. The first component derived using these maps is shown in Fig. 6.11. This map can be considered to show the overall pollution by HMCs with '*industrial*' origin (As, Pb, Hg and Cd) for the whole of USA.

Based on the results of analysis, we can conclude the following. First, auxiliary maps such as density of toxic releases, urbanization intensity, geology and similar, can be used to improve interpolation of various heavy metals. For example, distribution of Pb can be largely explained by density of toxic releases and night light images, several heavy metals can be explained by geological soil mapping units. Second, selected heavy metals are positively correlated — principal component plots for NGS are similar to the results of the European case study (Rodriguez Lado et al., 2009). Third, most of HMCs have distributions skewed towards low values. This proves that HMCs can in general be considered to follow a *Poisson*-type distribution.

This results also confirm that some local hot-spots shown in Fig. 6.5 are not really probable, and therefore have been smoothed out (compare with Fig. 6.9). Interpolation of some HMCs is not trivial. Mercury is, for example, a difficult element for which to obtain accurate analyzes (Grossman et al., 2008). Samples can easily be contaminated with Hg during handling, storage, and preparation for analysis.

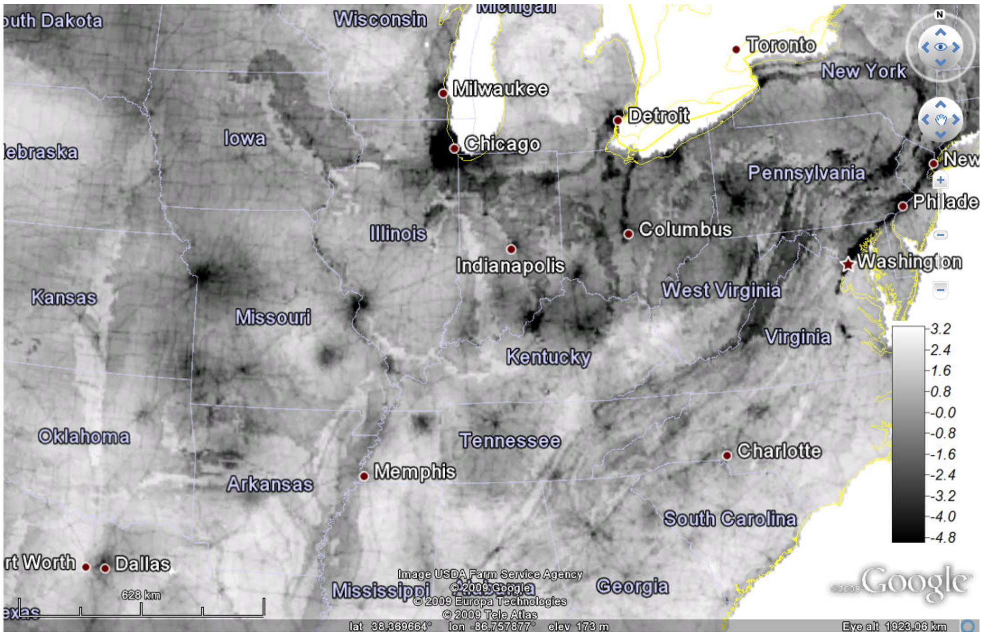


Fig. 6.11: First principal component derived using a stack of predicted maps of eight heavy metals. This PC basically represents mapped overall concentration of As, Pb and Cd (compare with Fig. 6.3); shown as a ground overlay in Google Earth.

6.5 Comparison of ordinary and regression-kriging

Finally we can also run a comparison between OK and RK methods to analyze the benefits of using auxiliary predictors (or are there benefits at all)? The recommended software to run such analysis is geoR, because it is more suited to analyzing skewed variables, and because it provides more insight into the results of model fitting. To speed up processing, we can focus on two US states (Illinois and Indiana) and only the most significant predictors. We can subset (using the bounding box coordinates) the auxiliary maps using:

```
# subset the original predictors:
> grid.list.s <- c("dairp.asc", "dTRI.asc", "nlights03.asc", "sdroads.asc")
> rsaga.esri.to.sgrd(in.grids=grid.list.s, out.sgrds=set.file.extension(grid.list.s,
+     ".sgrd"), in.path=getwd())
> for(i in 1:length(grid.list.s)) {
# first, create a new grid:
>   rsaga.geoprocessor(lib="grid_tools", module=23, param=list(GRID="tmp2.sgrd",
+        M_EXTENT=1, XMIN=360000, YMIN=1555000, XMAX=985000, YMAX=2210000, CELLSIZE=5000))
# 0.01 decimal places:
>   rsaga.geoprocessor("grid_calculus", module=1, param=list(INPUT="tmp2.sgrd",
+        RESULT=paste("m_", set.file.extension(grid.list.s[i], ".sgrd"), sep=""),
+        FORMUL="a/100")) # 0.01 decimal places
# now, resample all grids:
>   rsaga.geoprocessor(lib="grid_tools", module=0,
+        param=list(INPUT=set.file.extension(grid.list.s[i], ".sgrd"),
+        GRID=paste("m_", set.file.extension(grid.list.s[i], ".sgrd"), sep=""),
+        GRID_GRID=paste("m_", set.file.extension(grid.list.s[i], ".sgrd"), sep=""),
+        METHOD=2, KEEP_TYPE=FALSE, SCALE_DOWN_METHOD=0))
> }
> rsaga.sgrd.to.esri(in.sgrds=paste("m_", set.file.extension(grid.list.s, ".sgrd"),
+        sep=""), out.grids=paste("m_", set.file.extension(grid.list.s, ".asc"),
+        sep=""), out.path=getwd(), pre=3)
# read maps into R:
> gridmaps.s <- readGDAL(paste("m_", set.file.extension(grid.list.s[1], ".asc"), sep=""))
> for(i in 2:length(grid.list.s)) {
```

```
>   gridmaps.s@data[i] <- readGDAL(paste("m_", set.file.extension(grid.list.s[i],
+         ".asc"), sep=""))$band1
> }
> names(gridmaps.s) <- sub(".asc", "", grid.list.s)
> str(gridmaps.s@data)
```

```
 'data.frame':   16632 obs. of  4 variables:
  $ dairp    : num  0.031 0.03 0.031 0.032 0.033 ...
  $ dTRI     : num  0.007 0.007 0.007 0.008 0.008 ...
  $ nlights03: num  6 3 6 2 0 4 5 16 5 5 ...
  $ sdroads  : num  0 0 7497 0 0 ...
```

which has resampled the original grids to a 125×131 pixels block. We also need to subset the point data (we focus on Pb) using the same window (X_{min}=360000, X_{max}=985000, Y_{min}=1555000, Y_{max}=2210000) with the help of SAGA module `shapes_tools`:

```
# subset the point data:
> rsaga.geoprocessor(lib="shapes_tools", module=14,
+     param=list(SHAPES="PB_ICP40.shp", CUT="m_PB_ICP40.shp", METHOD=0, TARGET=0,
+     CUT_AX=360000, CUT_BX=985000, CUT_AY=1555000, CUT_BY=2210000))
> m_PB <- readOGR("m_PB_ICP40.shp", "m_PB_ICP40")
```

```
 OGR data source with driver: ESRI Shapefile
 Source: "m_PB_ICP40.shp", layer: "m_PB_ICP40"
 with  2787  rows and  1  columns
 Feature type: wkbPoint with 2 dimensions
```

which limits the analysis to only 2787 points within the area of interest. We convert the data to the native geoR format:

```
> Pb.geo <- as.geodata(m_PB["PB_ICP40"])
```

```
 as.geodata: 622 redundant locations found
 WARNING: there are data at coincident or very closed locations, some of the geoR's
 functions may not work. Use function dup.coords to locate duplicated coordinates.
```

which shows that there might be some problems for further analysis because there are many duplicate points and the calculation might fail due to singular matrix problems. Even though the data set is much smaller than the original NGS data set, geoR might still have problems running any analysis. Hence, a good idea is to (1) remove duplicates, and (2) randomly subset point data:

```
> m_PB <- remove.duplicates(m_PB)
> str(Pb.geo[[2]])
```

```
  num [1:2165] 9 10 10 9 16 14 8 15 11 9 ...
```

```
> m_PB.ov <- overlay(gridmaps.s, m_PB)
# subset to speed up:
> sel <- runif(length(m_PB@data[[1]]))<0.5
> Pb.geo1 <- as.geodata(m_PB[sel, "PB_ICP40"])
> str(Pb.geo1[[2]])
```

```
  num [1:1120] 9 10 14 11 11 18 14 13 10 8 ...
```

```
# copy values of covariates:
> Pb.geo1$covariate <- m_PB.ov@data[sel, sub(".asc", "", grid.list.s)]
```

We can now proceed with variogram modeling. First, we estimate the variogram for the original variable:

```
> Pb.vgm <- likfit(Pb.geo1, lambda=0, messages=FALSE, ini=c(var(log1p(Pb.geo$data)),
+      50000), cov.model="exponential")
> Pb.vgm
```

```
  likfit: estimated model parameters:
        beta       tausq     sigmasq        phi 
  "2.889e+00" "2.952e-01" "2.170e-01" "5.000e+04" 
  Practical Range with cor=0.05 for asymptotic range: 149786.6

  likfit: maximised log-likelihood = -1736
```

then for the residuals[19]:

```
> Pb.rvgm <- likfit(Pb.geo1, lambda=0, trend= ~ dairp+dTRI+nlights03+sdroads,
+   messages=FALSE, ini=c(var(log1p(Pb.geo$data))/5, 25000), cov.model="exponential")
> Pb.rvgm

  likfit: estimated model parameters:
        beta0        beta1        beta2        beta3        beta4 
  "    2.7999" "   -0.4811" "    2.4424" "    0.0022" "    0.0000" 
        tausq      sigmasq          phi 
  "    0.2735" "    0.1737" "24999.9999" 
  Practical Range with cor=0.05 for asymptotic range: 74893.3

  likfit: maximised log-likelihood = -2763
```

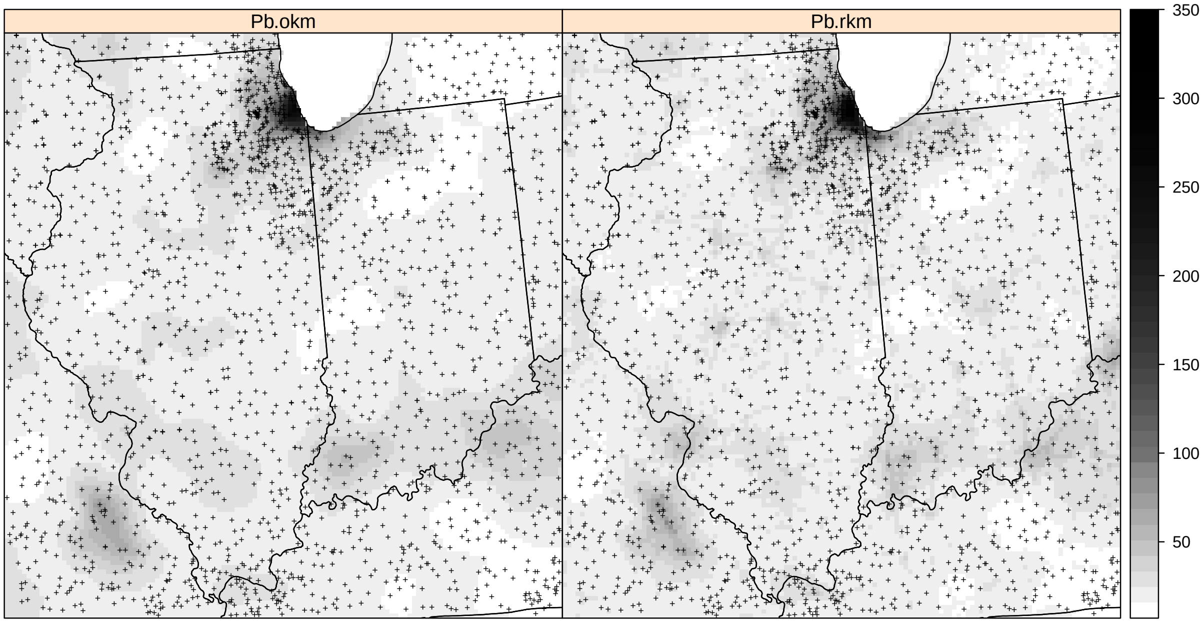


Fig. 6.12: Comparison of results of predicting values of Pb (ppm) using ordinary and regression-kriging (subset of 1120 points) for two US states (Illinois and Indiana). See text for more details.

Now that we have fitted the geostatistical model, we can prepare the prediction locations and run both ordinary and regression-kriging:

```
# prepare the covariates:
> locs.sp <- locs
> coordinates(locs.sp) <- ~ Var1+Var2
> gridmaps.gr <- overlay(gridmaps.s, locs.sp)
# Ordinary kriging:
> Pb.ok <- krige.conv(Pb.geo1, locations=locs, krige=krige.control(obj.m=Pb.vgm))
```

[19] These are residuals fitted using linear modeling, but after the Box-Cox transformation.

```
  krige.conv: model with constant mean
  krige.conv: performing the Box-Cox data transformation
  krige.conv: back-transforming the predicted mean and variance
  krige.conv: Kriging performed using global neighbourhood
```

```
# Regression-kriging:
> KC <- krige.control(trend.d = ~ dairp+dTRI+nlights03+sdroads, trend.l =
+    ~ gridmaps.gr$dairp+gridmaps.gr$dTRI+gridmaps.gr$nlights03+gridmaps.gr$sdroads,
+    obj.m = Pb.rvgm)
> Pb.rk <- krige.conv(Pb.geo1, locations=locs, krige=KC)
```

```
  krige.conv: model with mean defined by covariates provided by the user
  krige.conv: performing the Box-Cox data transformation
  krige.conv: back-transforming the predicted mean and variance
  krige.conv: Kriging performed using global neighbourhood
```

This time we did not use any mask (border coordinates), so that we need to mask the water bodies after converted the data to sp class (compare with §5.5.3):

```
# sp plot:
> locs.geo <- data.frame(X=locs.sp@coords[,1], Y=locs.sp@coords[,2],
+     Pb.rk=Pb.rk[[1]], Pb.ok=Pb.ok[[1]], Pb.rkvar=Pb.rk[[2]], Pb.okvar=Pb.ok[[2]])
> coordinates(locs.geo) <- ~ X+Y
> gridded(locs.geo) <- TRUE
# mask out water bodies:
> mask.s <- as.vector(t(as.im(gridmaps.s["geomap"])$v))  # flip pixels up-side down
> locs.geo$Pb.ok <- ifelse(is.na(mask.s), NA, locs.geo$Pb.ok)
> locs.geo$Pb.rk <- ifelse(is.na(mask.s), NA, locs.geo$Pb.rk)
> spplot(locs.geo[c("Pb.ok", "Pb.rk")], col.regions=grey(rev(seq(0,1,0.025)^2)),
+       at=seq(5,350,l=40), sp.layout=list(list("sp.points", m_PB, pch="+", col="black"),
+       list("sp.lines", USA.borders, col="black")))
> summary(locs.geo$Pb.okvar)
```

```
      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
     25.91   145.80   198.40   336.10   271.00 26860.00
```

```
> summary(locs.geo$Pb.rkvar)
```

```
     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    22.13   144.10   190.40   306.50   258.60 42200.00
```

The results are illustrated in Fig. 6.12. RK does seem to be more efficient in reflecting the spatial pattern of industrial activities and pollution sources. The range of values in the map predicted using RK is somewhat higher, which is due the fact that we have now used geoR package that deals very well with skewed distributions so that many peaks, invisible in the OK predictions map, have been emphasized with RK. The prediction error of RK is, as expected, somewhat smaller than for OK, but the difference is rather small (378 vs 391). This is also because the prediction error of the RK is proportionally higher in areas with high values, so that the overall average error is higher.

Self-study exercises:

(1.) At which locations are the maps shown in Fig. 6.5 and 6.9 the most different? (HINT: derive and plot a difference map)

(2.) Which predictors are most highly correlated with each other? Plot first and second component derived using all maps — what do they reflect?

(3.) Which HMC is the most difficult to interpolate? (HINT: look at the residuals of the regression model, nugget parameter etc.)

(4.) Split the original NGS point data set based on source (stream sediments, soils etc.) and then repeat the analysis for at least three HMCs. Is there a difference between the regression models? (HINT: plot correlation lines for various media in the same graph.)

(5.) How much does the accuracy in the HMC maps decrease if we use only 10% of samples (randomly selected) versus the complete data set? (HINT: use standard measures described in §1.4 to run a comparison.)

(6.) Which US state has highest concentration of Pb on average?

(7.) Run the cross-validation following the exercise in §6.5 and see if the predictions made using RK are significantly better than with OK.

(8.) Compare the accuracy of predictions for element Pb using ordinary kriging on untransformed data and using the Box–Cox transformation — are there significant differences? (HINT: randomly split the NGS data set to two equal size data sets; then use one for validation only.)

Further reading:

★ Grossman, J. N. and Grosz, A. E. and Schweitzer, P. N. and Schruben, P. G., 2008. The National Geochemical Survey — Database and Documentation, Version 5. U.S. Geological Survey, Reston, VA.

★ Papritz, A. and Reichard, P. U., 2009. Modeling the risk of Pb and PAH intervention value exceedance in allotment soils by robust logistic regression. Environmental Pollution, 157(7): 2019–2022.

★ Reimann, C., Filzmoser, P., Garrett, R., Dutter, R., 2008. **Statistical Data Analysis Explained Applied Environmental Statistics with R**. Wiley, Chichester, 337 p.

★ Rodriguez Lado, L. and Hengl, T. and Reuter, H. I., 2009. Heavy metals in European soils: a geostatistical analysis of the FOREGS Geochemical database. Geoderma, 148(2): 189–199.

★ `http://tin.er.usgs.gov/geochem/` — The National Geochemical Survey website.

★ `http://www.gtk.fi/publ/foregsatlas/` — The Geochemical atlas of Europe.

7

Soil Organic Carbon (WISE_SOC)

7.1 Introduction

ISRIC WISE is an international soil profile data set; a selection of globally distributed soil profiles, prepared by the International Soil Reference and Information Centre (ISRIC) located in Wageningen (Batjes, 2008, 2009). The most recent version (3.1) of the soil profile database contains 10,253 profiles[1]. The database consists of several tables, the most important are: WISE3_SITE (information about the soil profile site) and WISE3_HORIZON (laboratory data per horizon). This chapter demonstrates how to estimate the global **Soil Organic Carbon** (SOC) stock using regression-kriging and a large repository of publicly accessible global environmental maps (about 10 km resolution) described in section 4.1. The results contribute to the GlobalSoilMap.net initiative that aims at producing high resolution images of key soil properties and functions (Sanchez et al., 2009).

The maps presented in this exercise were created for demonstration purposes only. The true accuracy/-consistency of these maps has not been evaluated and is heavily controlled by the representativeness of the sampling pattern and accuracy of individual measurements (refer to the ISRIC WISE general disclaimer[2]). Positional accuracy of profiles in WISE varies depending on the source materials from which they were derived — this may range from the nearest second Lat/Lon up to few meters. Most of the available legacy data considered in WISE date from the pre-GPS era. In addition, the list of predictors we use in this exercise could be much more extensive; many of the maps are also available at finer resolutions (∼1 km). Compare also the maps produced in this chapter with the global soil property maps distributed by ISRIC[3], and/or the Global Biomass Carbon Map produced by the CDIAC[4] (Ruesch and Gibbs, 2008b). Note that this is also a relatively large data set and computations can become time-consuming. It is not recommended to run this exercise using a PC without at least 2 GB of RAM and at least 1 GB of hard disk memory.

7.2 Loading the data

To run this script, you first need to register and obtain the MS Access file from the ISRIC website[5]. Before you start using the ISRIC WISE database, please also read about its limitations in Batjes (2008). Next, load the necessary packages:

```
> library(RODBC)
> library(gstat)
> library(rgdal)
> library(RSAGA)
> library(spatstat)
```

[1] Not all profiles are complete.

[2] http://www.isric.org/isric/webdocs/Docs/ISRIC_Report_2008_02.pdf

[3] http://www.isric.org/UK/About+Soils/Soil+data/Geographic+data/Global/WISE5by5minutes.htm

[4] http://cdiac.ornl.gov/epubs/ndp/global_carbon/carbon_documentation.html

[5] http://www.isric.org/isric/CheckRegistration.aspx?dataset=9

For more info on how to set-up SAGA GIS and run the commands from R see section 3.1.2.

7.2.1 Download of the world maps

Next, download and unzip all relevant predictors from the web-repository[6]:

```
# location of maps:
> URL <- "http://spatial-analyst.net/worldmaps/"
# list of maps:
>  map.list <- c("biocl01", "biocl02", "biocl04", "biocl05", "biocl06", "biocl12",
+     "biocl15", "countries", "dcoast", "globedem", "landcov", "landmask", "gcarb",
+     "nlights", "pcndvi1", "pcndvi2", "pcndvi3", "pcpopd1", "himpact", "glwd31",
+     "wildness", "hwsd", "quakein", "iflworld", "treecov")
# download the zipped maps one by one:
> for(i in 1:length(map.list)) {
>   download.file(paste(URL, map.list[i], ".zip", sep=""),
+        destfile=paste(getwd(), "/", map.list[i], ".zip", sep=""))
>   unzip(paste(getwd(), "/", map.list[i], ".zip", sep=""))
>   unlink(paste(map.list[i], ".zip", sep=""))
# Delete temporary file:
>   unlink(paste(map.list[i], ".zip", sep=""))
> }
```

```
  trying URL 'http://spatial-analyst.net/worldmaps/biocl01.zip'
  Content type 'application/zip' length 1362739 bytes (1.3 Mb)
  opened URL
  downloaded 1.3 Mb
  ...
```

where biocl1-15 are long-term bioclimatic variables; dcoast is the distance from coastline; globedem is the ETOPO1 Global Relief Model; landcov is the Global Land Cover map of the world; landmask is the land mask; gcarb is the carbon (biomass) density in tones of C/ha; nlights is the long-term annual image of lights at night; pcndvi1/2 are first and second principal component derived from 20 years of AVHRR NDVI monthly images; pcpopd1 is PC1 of the Gridded Population of the World, version 3 (GPWv3), himpact is the world map of human impacts-free areas estimated by the GLOBIO initiative of the United Nations Environment Programme; glwd31 is the indicator map showing location of wetlands based on the Global Lakes and Wetlands Database (GWLD3.1) database; wildness is a map of the World wilderness areas; hwsd is soil class map based on the FAO Harmonized World Soil Database v 1.1 (37 classes); quakein is the Earthquake intensity (magnitude) based on the NOAA's Significant Earthquake Database; iflworld is the world map of intact forest landscapes; treecov is the Vegetation percent tree cover (see also §4.1). If the download was successful, you will notice that the ArcInfo ASCII grids are now available in your working directory. A detailed description of each layer is available via the raster description (*.rdc) file. See p.159 for an example.

We can load some maps, that we will need later on, into R using rgdal:

```
> worldmaps <- readGDAL("landmask.asc")
```

```
  landmask.asc has GDAL driver AAIGrid
  and has 1300 rows and 3600 columns
```

```
> names(worldmaps) <- "landmask"
> worldmaps$landcov <- as.factor(readGDAL("landcov.asc")$band1)
> worldmaps$glwd31 <- as.factor(readGDAL("glwd31.asc")$band1)
> worldmaps$hwsd <- as.factor(readGDAL("hwsd.asc")$band1)
> proj4string(worldmaps) <- CRS("+proj=longlat +ellps=WGS84")
```

[6] http://spatial-analyst.net/worldmaps/

7.2.2 Reading the ISRIC WISE into R

If you have obtained the ISRIC-WISE_ver3.mdb file from ISRIC, you can connect to it by using the RODBC[7] package:

```
> cGSPD <- odbcConnectAccess("ISRIC-WISE_ver3.mdb")
# Look at available tables:
> sqlTables(cGSPD)$TABLE_NAME
```

```
   [1] "MSysAccessObjects"
   [2] "MSysAccessXML"
   [3] "MSysACEs"
   [4] "MSysObjects"
   [5] "MSysQueries"
   [6] "MSysRelationships"
   [7] "WISE3__ReadMeFirst"
   [8] "WISE3_coding_conventions"
   [9] "WISE3_HORIZON"
  [10] "WISE3_LABcodes_Description"
  [11] "WISE3_LABname"
  [12] "WISE3_LABname_codes"
  [13] "WISE3_SITE"
  [14] "WISE3_SOURCE"
```

Now that we have connected to the database, we query it to obtain values from the tables like with any other SQL database. We need to obtain the following five variables:

- ORGC = Organic carbon content in *promille* (or g C kg^{-1});
- TOPDEP, BOTDEP = Thickness of soil horizons in cm;
- LON, LAT = Point coordinates;

We first fetch measured values for organic content and the depths of each horizon from WISE3_HORIZON table:

```
> GSPD.HOR <- sqlQuery(cGSPD, query="SELECT WISE3_ID, HONU, ORGC, TOPDEP,
+       BOTDEP FROM WISE3_HORIZON")
> str(GSPD.HOR)
```

```
  'data.frame':   47833 obs. of  5 variables:
   $ WISE3_ID: Factor w/ 10253 levels "AF0001","AF0002",..: 1 1 1 2 2 2 2 3 3 3 ...
   $ HONU    : int  1 2 3 1 2 3 4 1 2 3 ...
   $ ORGC    : num  7.6 2.3 0.9 12.8 6 3.9 2.7 5.9 2.4 NA ...
   $ TOPDEP  : int  0 15 60 0 20 60 110 0 20 50 ...
   $ BOTDEP  : int  15 60 150 20 60 110 170 20 50 110 ...
```

```
# Horizon thickness:
> GSPD.HOR$HOTH <- GSPD.HOR$BOTDEP-GSPD.HOR$TOPDEP
# unique ID:
> GSPD.HOR$ID <- as.factor(paste(as.character(GSPD.HOR$WISE3_ID),
+        GSPD.HOR$HONU, sep="_"))
```

where HONU is the horizon number (from the soil surface) and TOPDEP and BOTDEP are the upper and lower horizon depths. This shows that there are over 45 thousand measurements of the four variables of interest. The number of soil profiles is in fact much smaller — as you can see from the WISE3_ID column (unique profile ID), there are 10,253 profiles in total.

We know from literature that total soil organic carbon (SOC) depends on bulk density of soil and coarse fragments (Batjes, 1996; Batjes et al., 2007). There is certainly a difference in how ORGC relates to SOC in

[7] http://cran.r-project.org/web/packages/RODBC/

volcanic soils, wetland soils, organic soils and well drained mineral soils. To correctly estimate total Soil Organic Carbon in kg C m^{-2}, we can use the following formula[8]:

$$\mathtt{SOC}\ [\mathrm{kg\ m^{-2}}] = \frac{\mathtt{ORGC}}{1000}\ [\mathrm{kg\ kg^{-1}}] \cdot \frac{\mathtt{HOTH}}{100}\ [\mathrm{m}] \cdot \mathtt{BULKDENS} \cdot 1000\ [\mathrm{kg\ m^{-3}}] \cdot \frac{100 - \mathtt{GRAVEL}\ [\%]}{100} \qquad (7.2.1)$$

where BULKDENS is the soil bulk density[9] in g cm^{-3} and GRAVEL is the gravel content in profile expressed in %.

Because we are interested in total organic carbon content for soil profile, we will first estimate SOC values for all horizons, then aggregate these values per whole profile. Alternatively, one could try to predict organic carbon for various depths separately, then aggregate number of maps. Spatial analysis of soil layers makes sense because one can observe both shallow soils with high and low SOC content and vice versa, and these can both be formed under different environmental conditions. For the purpose of this exercise, we will focus only on the aggregate value i.e. on the total estimated soil organic carbon per profile location.

We can load an additional table[10] with Bulk density / gravel content estimated at fixed depth intervals (0–20, 20–40, 40–60, 60–80, 80–100, 100–150, 150–200 cm):

```
> load(url("http://spatial-analyst.net/book/system/files/GSPD_BDG.RData"))
> str(GSPD.BDG)
```

```
 'data.frame':   47111 obs. of  4 variables:
  $ WISE3_ID: Factor w/ 8189 levels "AF0001","AF0002",..: 1 1 1 1 1 1 2 2 2 2 ...
  $ BULKDENS: num  1.55 1.58 1.58 1.6 1.55 ...
  $ GRAVEL  : int  16 5 6 5 4 3 2 4 4 2 ...
  $ DEPTH   : num  10 30 50 70 90 125 10 30 50 70 ...
```

where BULKDENS is expressed in g cm^{-3}, GRAVEL is expressed in %, and DEPTH in cm. We first need to re-estimate the BULKDENS and GRAVEL at original depths for which we have ORGC measurements. We can do this by using e.g. linear interpolation:

```
# re-estimate values of BULKDENS and GRAVEL for original depths:
> GSPD.BDGa <- merge(x=GSPD.HOR[,c("WISE3_ID", "ID", "HOTH")],
+         y=GSPD.BDG, by=c("WISE3_ID"))
# estimate inverse distance weights:
> GSPD.BDGa$w <- 1/(GSPD.BDGa$HOTH-GSPD.BDGa$DEPTH)^2
> GSPD.BDGa$w <- ifelse(is.infinite(GSPD.BDGa$w), 0, GSPD.BDGa$w)
> GSPD.BDGa$BULKDENSa <- GSPD.BDGa$BULKDENS*GSPD.BDGa$w
> GSPD.BDGa$GRAVELa <- GSPD.BDGa$GRAVEL*GSPD.BDGa$w
# aggregate per each horizon:
> GSPD.BDG_ID <- aggregate(GSPD.BDGa[c("BULKDENSa", "GRAVELa", "w")],
+         by=list(GSPD.BDGa$ID), FUN=sum)
> GSPD.BDG_ID$BULKDENS <- GSPD.BDG_ID$BULKDENSa/GSPD.BDG_ID$w
> GSPD.BDG_ID$GRAVEL <- GSPD.BDG_ID$GRAVELa/GSPD.BDG_ID$w
> names(GSPD.BDG_ID)[1] <- "ID"
```

To combine the two tables we use:

```
> GSPD.HORa <- merge(x=GSPD.HOR[c("WISE3_ID", "HONU", "ID", "ORGC", "HOTH")],
+          y=GSPD.BDG_ID[,c("ID", "BULKDENS", "GRAVEL")], by=c("ID"))
```

and now we can estimate ORGC.d (kg C m^{-2}) using Eq.(7.2.1):

```
> GSPD.HORa$ORGC.d <- GSPD.HORa$ORGC/1000 * GSPD.HORa$HOTH/100 * GSPD.HORa$BULKDENS*1000
+             * (100-GSPD.HORa$GRAVEL)/100
> options(list(scipen=3,digits=3))
> round(summary(GSPD.HORa$ORGC.d), 1)  # total organic carbon in kg m^-2
```

[8] http://www.eoearth.org/article/soil_organic_carbon

[9] The average soil density is about 1682 kg m^{-3}. Different mean values for bulk density will apply for e.g. organic soils, Andosols, Arenosols, low activity (LAC) and high activity clays (HAC) soils.

[10] Prepared by Niels Batjes by using taxo-transfer procedures described in Batjes et al. (2007).

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    0.0     0.7     1.5     2.9     3.1   298.0  4851.0
```

where HOTH is the total thickness of the profile and ORGC.d is the estimated soil organic carbon for each horizon. We can now estimate the total soil organic carbon (SOC) in kg m^{-2} for the whole profile:

```
# select only horizons with ORGC!
> GSPD.orgc <- subset(GSPD.HORa, !is.na(GSPD.HORa$ORGC.d)&GSPD.HORa$ORGC.d>0,
+         c("WISE3_ID", "ORGC.d"))
# aggregate OGRC values per profiles (in kg / m^2):
> GSPD.orgc <- aggregate(GSPD.orgc["ORGC.d"], by=list(GSPD.orgc$WISE3_ID), FUN=sum)
# thickness of soil with biological activity:
> GSPD.hoth <- subset(GSPD.HORa, !is.na(GSPD.HORa$ORGC.d)&GSPD.HORa$ORGC.d>0,
+         c("WISE3_ID", "HOTH"))
# aggregate HOTH values to get the thickness of soil:
> GSPD.orgc$HOTH <- aggregate(GSPD.hoth["HOTH"],
+       by=list(GSPD.hoth$WISE3_ID), FUN=sum)$HOTH
```

which gives the following result:

```
> GSPD.orgc[1:10,]

   Group.1 ORGC.d HOTH
 1   AF0001   4.26  150
 2   AF0002  12.07  170
 3   AF0003   2.70   50
 4   AF0004   4.63   35
 5   AF0005   3.69  190
 6   AL0001  10.66   94
 7   AL0002   6.31   87
 8   AL0003   8.73   85
 9   AL0004  22.34  120
 10  AL0005  11.89  170
```

This shows that, for example, the profile AF0001 has 4.3 kg C m^{-2}, and organic carbon was observed up to the depth of 150 cm. Next, we want to obtain the coordinates of profiles:

```
# coordinates of points
> GSPD.latlon <- sqlQuery(cGSPD, query="SELECT WISE3_id, LATIT, LATDEG, LATMIN,
+     LATSEC, LONGI, LONDEG, LONMIN, LONSEC FROM WISE3_SITE")
> GSPD.latlon[1,]

  WISE3_id LATIT LATDEG LATMIN LATSEC LONGI LONDEG LONMIN LONSEC
 1   AL0030     N     40     39     40     E     20     48     58
```

These need to be converted to arcdegrees i.e. merged in single column. First, we remove the missing coordinates and then convert the multiple columns to a single column:

```
# make coordinates in arcdegrees:
> GSPD.latlon <- subset(GSPD.latlon, !is.na(GSPD.latlon$LATDEG)&
+    !is.na(GSPD.latlon$LONDEG)&!is.na(GSPD.latlon$LATMIN)&
+    !is.na(GSPD.latlon$LONMIN))
> GSPD.latlon$LATSEC <- ifelse(is.na(GSPD.latlon$LATSEC), 0, GSPD.latlon$LATSEC)
> GSPD.latlon$LONSEC <- ifelse(is.na(GSPD.latlon$LONSEC), 0, GSPD.latlon$LONSEC)
# define a new function to merge the degree, min, sec columns:
> cols2dms <- function(x,y,z,e)
+    {as(char2dms(paste(x, "d", y, "'", z, "\"", e, sep="")), "numeric")}
> GSPD.latlon$LAT <- cols2dms(GSPD.latlon$LATDEG, GSPD.latlon$LATMIN,
+    GSPD.latlon$LATSEC, GSPD.latlon$LATIT)
> GSPD.latlon$LON <- cols2dms(GSPD.latlon$LONDEG, GSPD.latlon$LONMIN,
+    GSPD.latlon$LONSEC, GSPD.latlon$LONGI)
```

The two tables (horizon properties and profile locations) can be merged by using:

```
> GSPD <- merge(x=data.frame(locid=GSPD.latlon$WISE3_id, LAT=GSPD.latlon$LAT,
+     LON=GSPD.latlon$LON), y=data.frame(locid=GSPD.orgc$Group.1, HOTH=GSPD.orgc$HOTH,
+     SOC=GSPD.orgc$ORGC.d), all.y=F, all.x=T, sort=F, by.x="locid")
> str(GSPD)

  'data.frame':   8065 obs. of  5 variables:
   $ locid: Factor w/ 10253 levels "AF0001","AF0002",..: 1 2 3 4 5 6 7 8 9 10 ...
   $ LAT  : num  34.5 34.5 34.5 34.3 32.4 ...
   $ LON  : num  69.2 69.2 69.2 61.4 62.1 ...
   $ HOTH : int  150 170 50 35 190 94 87 85 120 170 ...
   $ SOC  : num  4.26 12.07 2.7 4.63 3.69 ...
```

which can be converted to a point map (Fig. 7.1), and exported to a shapefile:

```
> coordinates(GSPD) <- ~ LON+LAT
> proj4string(GSPD) <- CRS("+proj=longlat +ellps=WGS84")
# export to a shapefile:
> writeOGR(GSPD, "SOC.shp", "SOC", "ESRI Shapefile")
# plot the world distribution:
> load(url("http://spatial-analyst.net/book/system/files/worldborders.RData"))
> bubble(subset(GSPD, !is.na(GSPD$SOC))["SOC"], col="black",
+      sp.layout=list("sp.lines", worldborders, col="light grey"))
```

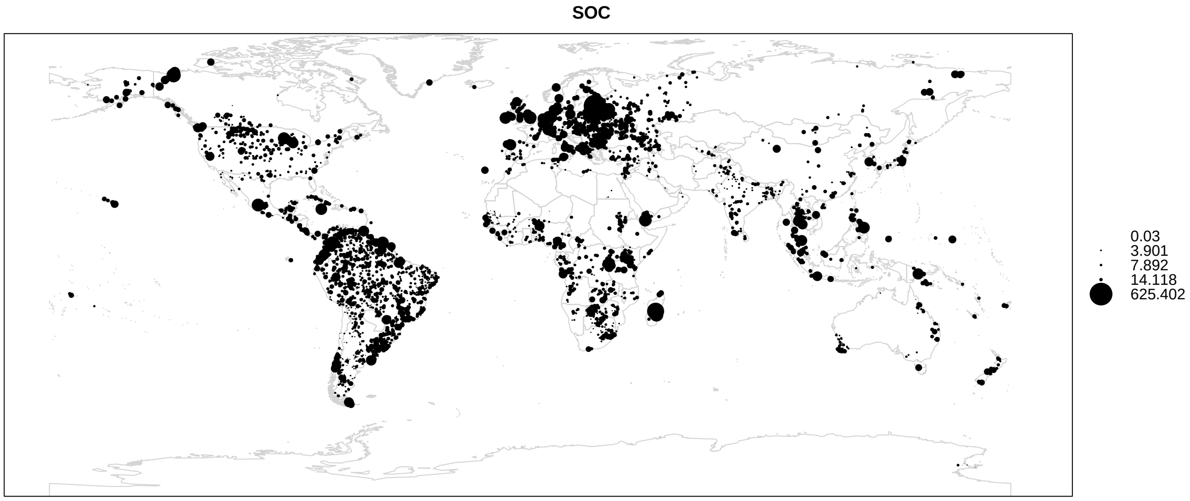


Fig. 7.1: Global distribution of soil profiles in the ISRIC WISE v3 database and values of total soil organic carbon (SOC in kg C m^{-2}). Note that the distribution of points is highly non-uniform — many large areas are not represented. See Batjes (2008) for more info.

Because we will use SAGA GIS to run the interpolation, you will also need to convert the downloaded worldmaps to the SAGA grid format:

```
> rsaga.esri.to.sgrd(in.grids=set.file.extension(map.list, ".asc"),
+      out.sgrds=set.file.extension(map.list, ".sgrd"), in.path=getwd())
```

Have in mind that these are relatively large grids (3600×1200 pixels), so the conversion process can take few minutes. To check that conversion was successful, you can open the maps in SAGA.

7.3 Regression modeling

Now that we have prepared a point map showing values of aggregated target variables, we can overlay the points over predictors (worldmaps) and prepare a regression matrix. Because there are many maps and they are relatively large, we run this operation instead using SAGA GIS[11]:

```
> rsaga.geoprocessor(lib="shapes_grid", module=0, param=list(SHAPES="SOC.shp",
+     GRIDS=paste(set.file.extension(map.list, ".sgrd"), collapse=";"),
+     RESULT="SOC_ov.shp", INTERPOL=0))  # simple nearest neighbor overlay

  SAGA CMD 2.0.4

  library path:   C:/PROGRA~2/R/R-29~1.2/library/RSAGA/saga_vc/modules
  library name:   shapes_grid
  module name :   Add Grid Values to Points
  author      :   (c) 2003 by O.Conrad

  Load shapes: SOC.shp...
  ready

  Load grid: biocl01.sgrd...
  ready

  ...

  Points: GSPD
  Grids: 25 objects (biocl01, biocl02, biocl04, biocl05, biocl06, biocl12, biocl15,
   countries, dcoast, globedem, landcov, landmask, nlights, pcndvi1, pcndvi2,
   pcndvi3, pcpopd1, himpact, glwd31, wildness, gcarb, quakein, iflworld, treecov))
  Result: Result
  Interpolation: Nearest Neighbor

  ready
  Save shapes: SOC_ov.shp...

  ready
  Save table: SOC_ov.dbf...
```

This will produce a point shapefile, which we can then read back into R:

```
> SOC.ov <- readShapePoints("SOC_ov.shp", CRS("+proj=longlat +ellps=WGS84"))
# fix the names:
> names(SOC.ov@data)[4:length(SOC.ov@data)] <- map.list
# note that SAGA can not generate NA values but inserts instead "0" values!!
> SOC.ov <- subset(SOC.ov, SOC.ov$landmask==1&SOC.ov$HOTH>0)
# some points fall outside the landmask!
> str(SOC.ov@data)

  'data.frame':   7681 obs. of  30 variables:
   $ LOCID    : Factor w/ 8065 levels "AF0001","AF0002",..: 1 2 3 4 5 6 7 8 9 10 ...
   $ HOTH     : int  150 170 50 35 190 94 87 85 120 170 ...
   $ SOC      : num  4.26 12.07 2.7 4.63 3.69 ...
   $ biocl01  : num  119 119 119 172 199 104 138 110 160 156 ...
   $ biocl02  : num  149 149 149 156 174 ...
   $ biocl04  : num  8678 8678 8678 8549 9028 ...
   $ biocl05  : num  321 321 321 375 421 268 294 270 291 291 ...
   $ biocl06  : num  -82 -82 -82 -13 -9 -28 16 -30 48 40 ...
   $ biocl12  : num  340 340 340 222 79 ...
   $ biocl15  : num  98 98 98 102 107 32 61 26 46 44 ...
   $ countries: num  1 1 1 1 1 2 85 2 2 2 ...
```

[11] Loading such a large quantity of maps to R would be very inefficient and is not recommended for OS with <4GB RAM.

```
 $ dcoast   : num  1091 1091 1091 803 782 ...
 $ globedem : num  1790 1790 1790 776 780 ...
 $ landcov  : num  9 9 9 9 9 4 1 11 11 11 ...
 $ landmask : num  1 1 1 1 1 1 1 1 1 1 ...
 $ nlights  : num  3 3 3 0 0 6 0 0 4 5 ...
 $ pcndvi1  : num  2295 2295 2295 2140 2238 ...
 $ pcndvi2  : num  -77 -77 -77 -178 -159 -64 -244 13 -170 -153 ...
 $ pcndvi3  : num  123 123 123 118 122 108 127 73 152 128 ...
 $ pcpopd1  : num  30076.6 30076.6 30076.6 24.5 81.9 ...
 $ himpact  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ glwd31   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ wildness : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hwsd     : num  10 10 10 10 7 21 20 26 4 26 ...
 $ gcarb    : num  6.5 6.5 6.5 4.2 2.5 ...
 $ quakein  : num  7.9 7.9 7.9 3.2 0 ...
 $ iflworld : num  0 0 0 0 0 0 0 0 0 0 ...
 $ treecov  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ coords.x1: num  69.2 69.2 69.2 61.4 62.1 ...
 $ coords.x2: num  34.5 34.5 34.5 34.3 32.4 ...
```

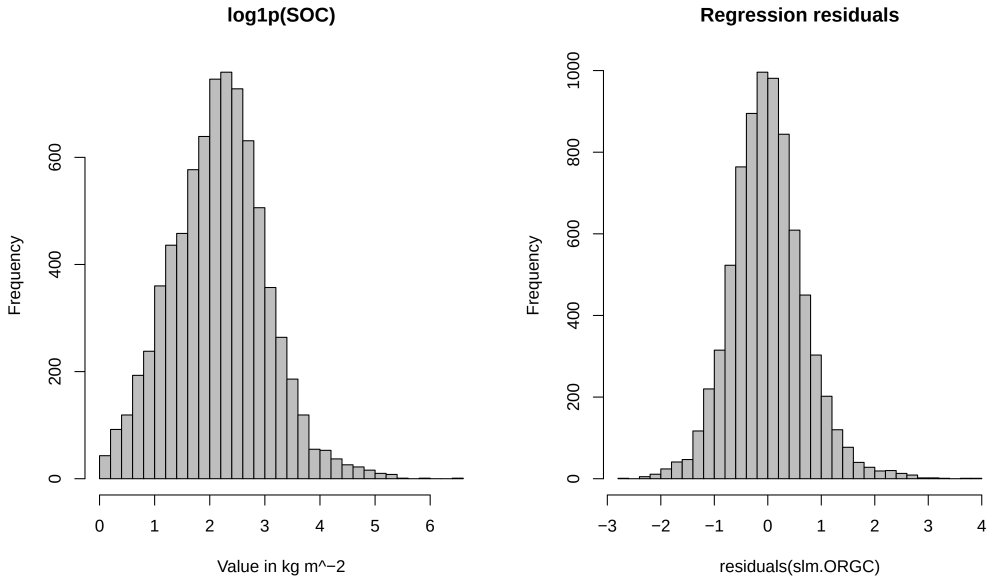


Fig. 7.2: Target variable (soil organic carbon) after the necessary transformation and histogram for the regression residuals.

Before we proceed with regression analysis, it is a good idea to visualize histograms for the target variable, in order to see if they need to be transformed before model fitting[12]. You will soon notice that SOC needs to be transformed before regression modeling (Fig. 7.2):

```
> hist(log1p(SOC.ov$SOC), col="grey")
```

The transformed variable shows close to normal distribution, so that we can now fit a regression model:

```
> orgc.formula <- as.formula(paste("log1p(SOC)~", paste(sub(".asc", "",
+    map.list[!(map.list %in% c("landmask", "countries", "wwfeco"))]), collapse="+")))
# some maps we do not need!
> orgc.formula
```

[12] Close-to-normal distribution is a prerequisite for regression modeling.

```
  log1p(SOC) ~ biocl01 + biocl02 + biocl04 + biocl05 + biocl06 +
      biocl12 + biocl15 + dcoast + globedem + landcov + nlights +
      pcndvi1 + pcndvi2 + pcndvi3 + pcpopd1 + himpact + glwd31 +
      wildness + hwsd + gcarb + quakein + iflworld + treecov
```

```
> lm.ORGC <- lm(orgc.formula, SOC.ov@data)
> slm.ORGC <- step(lm.ORGC, trace=-1) # step-wise regression
> summary(slm.ORGC)$adj.r.squared
```

```
  [1] 0.363
```

This shows that the predictors explain 36% of variability in the SOC values (cumulative density of organic carbon in the soil). For Digital Soil Mapping projects (Lagacherie et al., 2006), this is a promising value.

For practical reasons (computational intensity), we will further focus on using only the top 20 most significant predictors to generate predictions. These can be selected by using:

```
> pr.rank <- rank(summary(slm.ORGC)$coefficients[,4])<20
> SOC.predictors <- attr(summary(slm.ORGC)$coefficients[pr.rank,1], "names")[-1]
> SOC.predictors
```

```
   [1] "biocl01"   "biocl02"   "biocl04"   "biocl12"
   [5] "globedem"  "landcov9"  "landcov12" "nlights"
   [9] "glwd312"   "glwd314"   "glwd317"   "hwsd4"
  [13] "hwsd6"     "hwsd17"    "hwsd19"    "hwsd25"
  [17] "hwsd26"    "quakein"
```

After we have determined the top 20 most significant predictors, we can make predictions by using the SAGA **multiple linear regression module**. However, before we can produce predictions in SAGA, we need to prepare the indicator maps and a shapefile with transformed target variable. For example, to prepare indicators for different classes of land cover, we can use:

```
> for(j in c("9","12")){
>   worldmaps$tmp <- ifelse(worldmaps$landcov==j, 1, 0)
>   write.asciigrid(worldmaps["tmp"], paste("landcov", j, ".asc", sep=""), na.value=-1)
> }
...
# list all indicators and convert to SAGA grids:
> indicator.grids <- c(list.files(getwd(), pattern="hwsd[[:digit:]]*.asc",
+         recursive=F, full=F),
+         list.files(getwd(), pattern="glwd31[[:digit:]]*.asc", recursive=F, full=F),
+         list.files(getwd(), pattern="landcov[[:digit:]]*.asc", recursive=F, full=F))
> rsaga.esri.to.sgrd(in.grids=indicator.grids,
+         out.sgrds=set.file.extension(indicator.grids, ".sgrd"), in.path=getwd())
```

We also need to prepare the point map with transformed target variables:

```
> SOC.ov$SOC.T <- log1p(SOC.ov$SOC)
> SOC.ov$HOTH.T <- sqrt(SOC.ov$HOTH)
> writeOGR(SOC.ov[c("SOC.T","HOTH.T")], "SOC_ov.shp", "SOC_ov", "ESRI Shapefile")
```

which now allows us to use SAGA GIS to make predictions using multiple linear regression:

```
> rsaga.geoprocessor(lib="geostatistics_grid", module=4,
+    param=list(GRIDS=paste(set.file.extension(SOC.predictors, ".sgrd"), collapse=";"),
+    SHAPES="SOC_ov.shp", ATTRIBUTE=0, TABLE="regout.dbf", RESIDUAL="res_SOC.shp",
+    REGRESSION="SOC_reg.sgrd", INTERPOL=0))
```

```
  ...
  1: RÂš = 15.508441% [15.508441%] -> biocl02

  2: RÂš = 21.699108% [6.190666%] -> globedem
```

```
3: RÂš = 24.552229% [2.853121%] -> hwsd4

4: RÂš = 26.552463% [2.000235%] -> biocl12

5: RÂš = 30.908089% [4.355626%] -> biocl01

6: RÂš = 31.498327% [0.590238%] -> hwsd26

7: RÂš = 31.993559% [0.495233%] -> hwsd6

8: RÂš = 32.407088% [0.413529%] -> hwsd17

9: RÂš = 32.738160% [0.331072%] -> landcov12

10: RÂš = 33.136920% [0.398761%] -> landcov9

11: RÂš = 33.434208% [0.297288%] -> hwsd19

12: RÂš = 33.700079% [0.265871%] -> biocl04

...
```

which shows that the best predictors are Mean Diurnal Range (biocl02), elevation (globedem), various soil types (hwsd), annual temperature (biocl01), temperature seasonality (biocl04), annual precipitation (biocl12), and land cover classes. Most of variation in SOC values can be explained by using only bioclimatic maps.

The resulting map (Fig. 7.3) shows that high organic carbon concentration in soil can be mainly observed in the wet and cooler areas (mountain chains); deserts and areas of low biomass have distinctly lower soil organic carbon. Surprisingly, the model predicts high SOC concentration also in arctic zones (Greenland) and Himalayas, which is an obvious artifact. Recall that the sampling locations have not been chosen to represent all possible environmental conditions, so the model is probably extrapolating in these areas (see also p.59).

To speed up further analysis we will focus on estimating SOC for South American continent only. This is the continent with best (most consistent) spatial coverage, as visible from Fig. 7.3 (below). For further analysis, we do not really need all maps, but just the estimate of the trend (SOC_reg). We can reproject the maps using (see also §6.5):

```
> SA.aea <- "+proj=aea +lat_1=-5 +lat_2=-42 +lat_0=-32 +lon_0=-60 +x_0=0 +y_0=0
+      +ellps=aust_SA +units=m +no_defs"
> rsaga.geoprocessor(lib="pj_proj4", 2,
+      param=list(SOURCE_PROJ="\"+proj=longlat +datum=WGS84\"",
+      TARGET_PROJ=paste('"', SA.aea ,'"', sep=""), SOURCE="SOC_reg.sgrd",
+      TARGET="m_SOC_reg.sgrd", TARGET_TYPE=2, INTERPOLATION=1,
+      GET_SYSTEM_SYSTEM_NX=586, GET_SYSTEM_SYSTEM_NY=770, GET_SYSTEM_SYSTEM_X=-2927000,
+      GET_SYSTEM_SYSTEM_Y=-2597000, GET_SYSTEM_SYSTEM_D=10000))
```

```
SAGA CMD 2.0.4

library path:   C:/PROGRA~2/R/R-29~1.2/library/RSAGA/saga_vc/modules
library name:   pj_proj4
module name :   Proj.4 (Command Line Arguments, Grid)
author      :   O. Conrad (c) 2004-8

Load grid: SOC_reg.sgrd...
ready

Parameters

Inverse: no
Source Projection Parameters: +proj=longlat +datum=WGS84
Target Projection Parameters: +proj=aea +lat_1=-5 +lat_2=-42 +lat_0=-32
+lon_0=-60 +units=m +no_defs +x_0=0 +y_0=0 +ellps=aust_SA
```

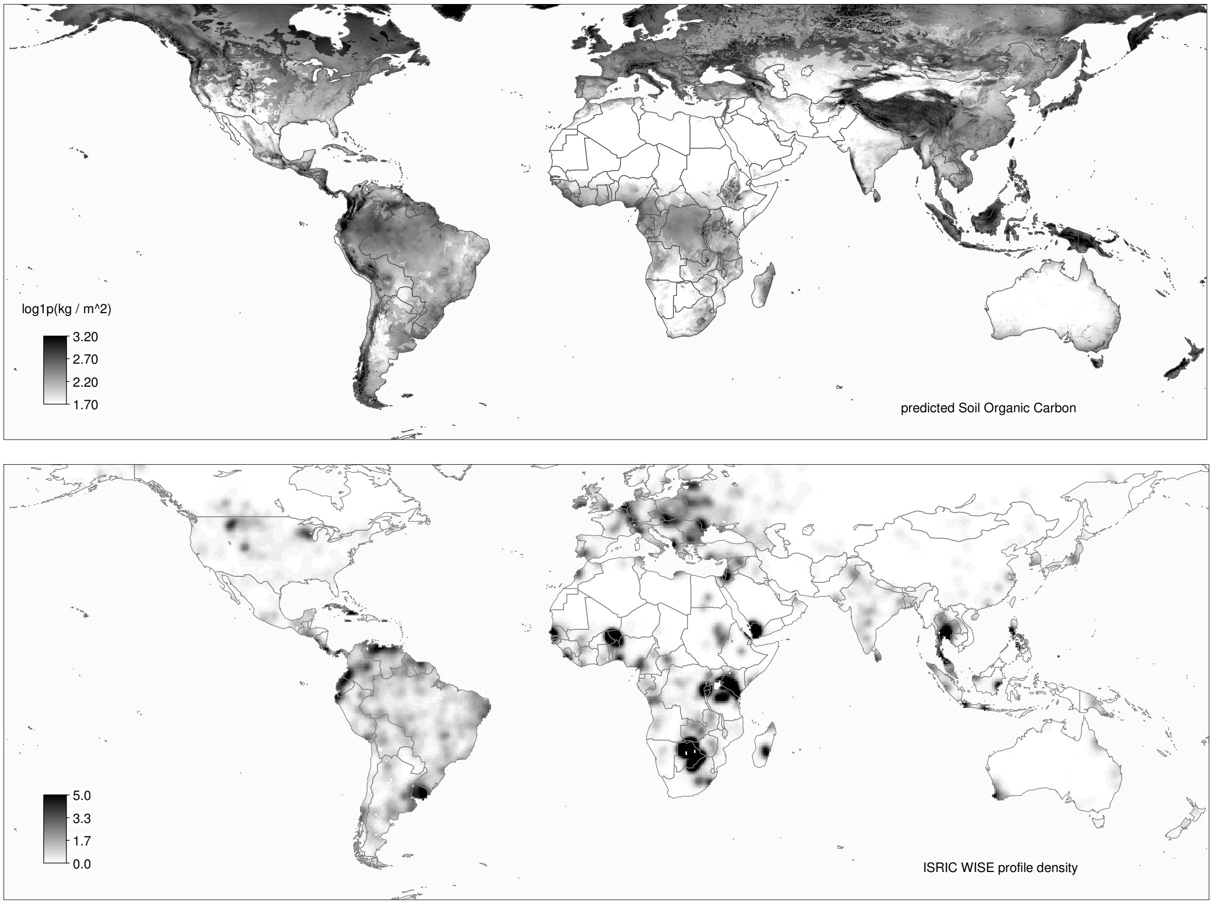


Fig. 7.3: Predicted values of the target variable (`log1p(SOC)`) using the 20 most significant predictors and multiple linear regression module in SAGA GIS (above). ISRIC WISE coverage map — sampling density on 0.5 arcdegree grid derived using kernel smoothing (below).

```
  Grid system: 0.1; 3600x 1300y; -179.95x -64.95y
  Source: SOC_reg
  Target: [not set]
  Shapes: [not set]
  X Coordinates: [not set]
  Y Coordinates: [not set]
  Create X/Y Grids: no
  Target: grid system
  Interpolation: Bilinear Interpolation

  Source: +proj=longlat +datum=WGS84

  Target: +proj=aea +lat_1=-5 +lat_2=-42 +lat_0=-32 +lon_0=-60 +x_0=0
  +y_0=0 +ellps=aust_SA +units=m +no_defs

  Save grid: m_SOC_reg.sgrd...
  ready
```

```
> gridsSA <- readGDAL("m_SOC_reg.asc")

  m_SOC_reg.asc has GDAL driver AAIGrid
  and has 770 rows and 586 columns
```

```
> names(gridsSA) <- "SOC_reg"
> proj4string(gridsSA) <- CRS(SA.aea)
> SA.bbox <- gridsSA@bbox
> SA.bbox

        min     max
 x -2932000 2928000
 y -2602000 5098000
```

which will reproject and resample the predicted `log1p(SOC)` map from geographic coordinates to the Albers Equal-Area Conic projection system[13], commonly used to represent the whole South American continent.

7.4 Modeling spatial auto-correlation

We have explained some 36% of variation in the SOC values using worldmaps. Next we can look at the variograms i.e. try to improve interpolations using kriging. Because we focus only on the South American continent, we also need to subset the point map of profiles:

```
# reproject the profile data:
> GSPD.aea <- spTransform(GSPD, CRS(SA.aea))
> writeOGR(GSPD.aea, "GSPD_aea.shp", ".", "ESRI Shapefile")
# subset the points:
> rsaga.geoprocessor(lib="shapes_tools", module=14,
+      param=list(SHAPES="GSPD_aea.shp", CUT="m_GSPD_aea.shp",
+      METHOD=0, TARGET=0, CUT_AX=SA.bbox[1,1], CUT_BX=SA.bbox[1,2],
+      CUT_AY=SA.bbox[2,1], CUT_BY=SA.bbox[2,2]))
```

which will subset the input point map to 1729 points. These can be now analyzed for spatial auto-correlation:

```
> m_SOC <- readShapePoints("m_GSPD_aea.shp", CRS(SA.aea))
> m_SOC.ov <- overlay(gridsSA, m_SOC)
> m_SOC.ov$SOC <- m_SOC$SOC
> m_SOC.ov <- remove.duplicates(m_SOC.ov)  # many duplicate points!
> sel <- !is.na(m_SOC.ov$SOC)&!is.na(m_SOC.ov$SOC_reg)
> res_SOC.svar <- variogram(log1p(SOC) ~ SOC_reg, m_SOC.ov[sel,])
> SOC.rvgm <- fit.variogram(res_SOC.svar, vgm(nugget=var(SOC.ov$SOC, na.rm=T)/2,
+     model="Exp", range=80000, sill=var(SOC.ov$SOC, na.rm=T)/2))
> SOC.rvgm

   model     psill    range
 1   Nug 0.3879486      0.0
 2   Exp 0.1151655 823650.5
```

which shows that the residuals are correlated up to the distance of >1000 km. This number seems unrealistic. In practice, we know that soils form mainly at watershed level or even at short distances, so chances that two profile locations that are so far away still make influence on each other are low. On the other hand, from statistical perspective, there is no reason not to utilize this auto-correlation to improve the existing predictions.

Variograms for the original variable and regression residuals can be seen in Fig. 7.4. Note also that the variance of the residuals is about 70% of the original variance, which corresponds to the R-square estimated by the regression model. Compare also this plot to some previous exercises, e.g. Fig. 5.8. We can in general say that the nugget variation of SOC is relatively high, which indicates that our estimate of global SOC will be of limited accuracy.

7.5 Adjusting final predictions using empirical maps

Once we have estimated the variogram for residuals, we can proceed with regression-kriging[14]:

[13] `http://spatialreference.org/ref/esri/102033/`

[14] In this case implemented as kriging with external drift, and with a single predictor — regression estimate (see §2.1.4).

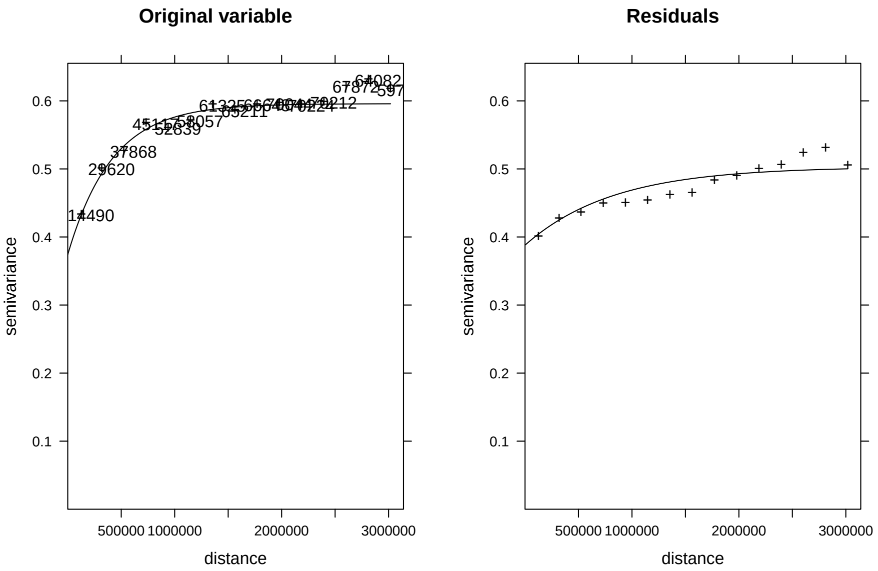


Fig. 7.4: Variograms for SOC fitted in gstat using standard settings.

```
# block regression-kriging:
> m_SOC.rk <- krige(log1p(SOC) ~ SOC_reg, m_SOC.ov[sel,], gridsSA, SOC.rvgm,
+      nmin=30, nmax=40, block=c(10e3, 10e3))

  [using universal kriging]
  Warning message:
  In points2grid(points, tolerance, round, fuzz.tol) :
    grid has empty column/rows in dimension 2

# back-transform values:
> m_SOC.rk$SOC_rk <- expm1(m_SOC.rk$var1.pred)
```

in this case, gstat reported about empty pixels in the map that have been removed. The final regression-kriging 1 map of SOC for South American continent can be seen in Fig. 7.5. The RK model predicts even in the areas 2 where there are almost no soil profiles, hence the map is possibly of poor quality in some regions. To improve 3 this map, we can use the USDA–produced Soil Organic Carbon Map[15], which is shown in Fig. 7.5 (2): 4

```
# reproject and import the USDA map:
> download.file("http://spatial-analyst.net/worldmaps/SOC.zip",
+     destfile=paste(getwd(), "/SOC.zip", sep=""))
> unzip("SOC.zip")
# resample to the same grid:
> rsaga.esri.to.sgrd(in.grids="SOC.asc", out.sgrd="SOC.sgrd", in.path=getwd())
> rsaga.geoprocessor(lib="pj_proj4", 2,
+     param=list(SOURCE_PROJ="\"+proj=longlat +datum=WGS84\"",
+     TARGET_PROJ=paste('"', SA.aea ,'"', sep=""),
+     SOURCE="SOC.sgrd", TARGET="m_SOC_USDA.sgrd", TARGET_TYPE=2, INTERPOLATION=1,
+     GET_SYSTEM_SYSTEM_NX=m_SOC.rk@grid@cells.dim[[1]],
+     GET_SYSTEM_SYSTEM_NY=m_SOC.rk@grid@cells.dim[[2]],
+     GET_SYSTEM_SYSTEM_X=m_SOC.rk@grid@cellcentre.offset[[1]],
+     GET_SYSTEM_SYSTEM_Y=m_SOC.rk@grid@cellcentre.offset[[2]],
+     GET_SYSTEM_SYSTEM_D=10000))
> rsaga.sgrd.to.esri(in.sgrds="m_SOC_USDA.sgrd", out.grids="m_SOC_USDA.asc",
+     out.path=getwd(), prec=3)
> m_SOC.rk$SOC_USDA <- readGDAL("m_SOC_USDA.asc")$band1
```

[15] http://soils.usda.gov/use/worldsoils/mapindex/

Recall from §2.1.3 that, if we know the uncertainty of both maps, we can derive a weighted average and create a combined prediction. In this case, we do not have any estimate of the uncertainty of the USDA SOC map; we only have an estimate of the uncertainty of RK SOC map. Because there are only two maps, the weights for the RK map (Fig. 7.5 (3)) can be derived using the inverse of the relative prediction variance (see p.25); the remaining weights for the USDA map can be derived as $1 - w$. Or in R syntax:

```
# merge the two maps (BCSP formula):
> w <- sqrt(m_SOC.rk$var1.var)/sqrt(var(log1p(m_SOC.ov$SOC), na.rm=T))
> m_SOC.rk$w <- 1-w/max(w, na.rm=TRUE)
> m_SOC.rk$SOC.f <- m_SOC.rk$w * m_SOC.rk$SOC_rk + (1-m_SOC.rk$w) * m_SOC.rk$SOC_USDA
```

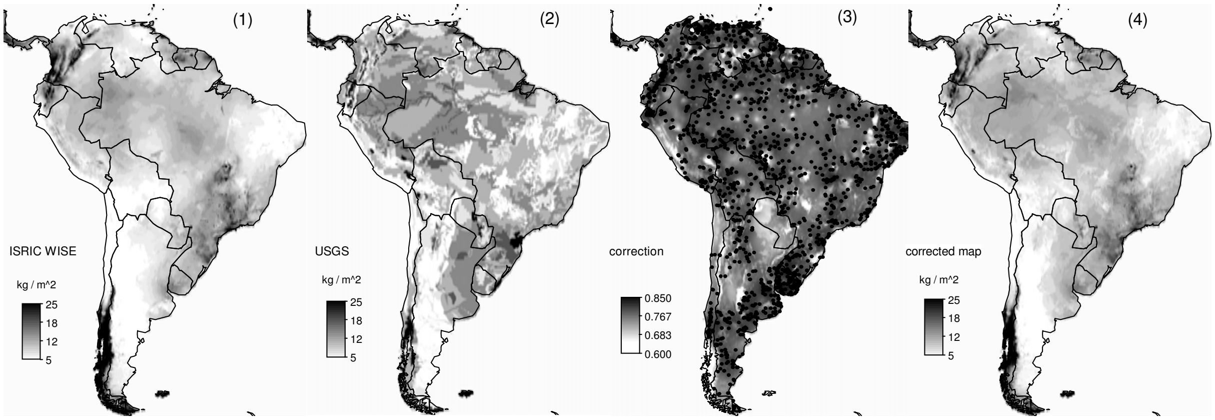


Fig. 7.5: Soil Organic Carbon stock (kg C m^{-2}) for South America: (1) predicted using regression-kriging, (2) the USDA SOC map produced using soil regions; (3) sampling locations and map of weights derived as the inverse relative prediction error; (4) the final corrected map of SOC derived as a weighted average between the maps (1) and (2).

The final corrected map of SOC is shown in Fig. 7.5 (4). In this case, the USDA map is assume to be more spatially 'consistent' about the actual SOC stock. The weighted average between the two maps is possibly the best estimate of the Soil Organic Carbon given the limited data. To validate this map, one would need to collect block estimates of SOC with a support size of 10 km (Heuvelink and Pebesma, 1999).

7.6 Summary points

Estimation of organic carbon stock using ISRIC WISE profiles and geostatistical techniques is possible, but the final map is of limited quality: (a) soil samples are fairly clustered (Fig. 7.3, below), for many regions there are still no measured soil data; (b) predictors used are rather coarse (cca. 10 km), which limits the regression modeling; (c) the residuals for SOC consequently show high nugget. The map presented in Fig. 7.5 (1) can be considered to be especially poor where the density of point samples is low. The question remains whether the models would improve if one would consider fitting variogram models locally (moving window), or by using finer-grain predictors (<10 km) that could potentially be able to explain short-range variation. On the other hand, we know that there is inherent uncertainty in the geo-locations of WISE profiles, so that not even finer-grain predictors would help us improve the predictions.

If you repeat a similar analysis with other soil variables of interest, you will notice that the gridded predictors explain only between 10–40% of the observed variability in the values (e.g. 36% for SOC, 11% for HOTH, 22% for SAND, 28% for SILT, 15% for CLAY), which means that these maps are of limited accuracy. The variograms also show relatively high nugget, which also means that about 30–60% of variability in these target variables cannot be explained by regression-kriging. This is particulary problematic for large regions that are completely under represented — most of the former Russian federation, Australia and Canada (see the map in Fig. 7.1). Nevertheless, the main patterns of soil parameters produced using ISRIC WISE will often correspond to our empirical knowledge: high soil organic carbon mainly reflects the cold and wet temperatures (Batjes, 1996); deep soils are predicted in the tropical regions and areas of high biomass (Eswaran et al., 1993); texture classes are connected with the land cover, relief and soil mapping units etc.

The advantage of automating the operations, on the other hand, is that these maps can be easily updated once the ISRIC WISE becomes more representative and of higher quality. Due to the data processing automation, many other important soil parameters from the ISRIC WISE database could easily be revised once updates with better geographical coverage are released .

Self-study exercises:

(1.) Estimate nugget variation for `SOC` for the five largest countries in the world. Plot the variograms one over the other.

(2.) Compare the Global Biomass Carbon Map distributed by The Carbon Dioxide Information Analysis Center and the total soil carbon map shown in Fig. 7.5(4). Are the two maps correlated and how much? Where is the difference highest and why?

(3.) Repeat the spatial prediction of soil carbon by focusing on the North American continent (HINT: resample the maps following the previous exercise in §6.5.)

(4.) Which country in the world has highest reserves of organic carbon in absolute terms (total soil carbon in tones), and which one in relative terms (average density of carbon)?

(5.) Compare spatial prediction of `SOC` for South America and Africa (regression-kriging variance). Why are soil profile data in South America more suited for geostatistical mapping?

(6.) Interpolate soil textures (`SAND`, `SILT`, `CLAY`) using the same procedure explained in the text and produce global maps.

(7.) Focus on Australia and compare the soil organic carbon map available from the Australian soil atlas with the map shown in Fig. 7.3. Plot the two maps next to each other using the same grid settings.

Further reading:

★ Batjes, N.H., 2009. Harmonized soil profile data for applications at global and continental scales: updates to the WISE database. Soil Use and Management 25, 124-127.

★ Lagacherie, P., McBratney, A.B., Voltz, M., (eds) 2006. **Digital Soil Mapping: An Introductory Perspective**. Developments in Soil Science, Volume 31. Elsevier, Amsterdam, 350 p.

★ Ruesch, A., Gibbs, H.K., 2008. New IPCC Tier-1 Global Biomass Carbon Map For the Year 2000. Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 45 p.

★ `http://www.isric.org` — ISRIC World Soil Information center;

★ `http://www.pedometrics.org` — The international research group on pedometrics;

★ `http://www.globalsoilmap.net` — International consortium that aims to make a new digital soil map of the world using state-of-the-art and emerging technologies for soil mapping and predicting soil properties at fine resolution;

8

Species’ occurrence records (bei)

8.1 Introduction

A special group of Species Distribution Models (SDMs) focuses on the so-called **occurrence-only records** — pure records of locations where a species occurred (Tsoar et al., 2007). Although such point data set can be considered of interest to geostatistics, standard geostatistical techniques cannot be used to generate (realized) species’ distributions using occurrence-only data, mainly for two reasons: (1) absence locations are missing (‘1’s only), so that it is not possible to analyze the data using e.g. indicator geostatistics; and (2) the sampling is purposive and points are often clustered in both geographical and feature spaces, which typically causes difficulties during the model estimation.

Spatial statisticians (e.g. Diggle (2003); Bivand et al. (2008)) generally believe that geostatistical techniques are suited only for modeling of features that are inherently continuous (spatial fields); discrete objects (points, lines, polygons) should be analyzed using point pattern analysis and similar methods. This exercises tries to bridge this gap. It demonstrates how to combine geostatistical techniques with conceptually different techniques — point pattern analysis and Niche analysis — to allow prediction of species’ distributions using regression-kriging. For more info about the theoretical basis for this approach see section 2.6. This chapter is based on an article published in Ecological Modeling journal (Hengl et al., 2009b).

We will use the data set `bei`, distributed together with the spatstat[1] package, and used in school books on point pattern analysis by Baddeley (2008) and many other authors. This data set consists of a point map showing observed locations of trees of the species *Beilschmiedia pendula Lauraceae*. This is only a small part of the Barro Colorado Island 50 Hectare Plot[2] Data set (Panama) that contains about 2 million records — complete inventory of all plant species and numerous plant and environmental attributes for six time periods (Leigh et al., 2004).

8.1.1 Preparation of maps

To run the `bei.R` script, you will first need to load the following packages:

```
> library(spatstat)
> library(adehabitat)
> library(gstat)
> library(splancs)
> library(rgdal)
> library(RSAGA)
```

We load the `bei` data set directly from R:

[1] `http://spatstat.org`

[2] `http://www.stri.org/english/research/facilities/terrestrial/barro_colorado/`

```
> data(bei)
> str(bei)
```

```
List of 5
 $ window    :List of 5
  ..$ type  : chr "rectangle"
  ..$ xrange: num [1:2] 0 1000
  ..$ yrange: num [1:2] 0 500
  ..$ units :List of 3
  .. ..$ singular  : chr "metre"
  .. ..$ plural    : chr "metres"
  .. ..$ multiplier: num 1
  .. ..- attr(*, "class")= chr "units"
  ..$ area  : num 5e+05
  ..- attr(*, "class")= chr "owin"
 $ n         : int 3604
 $ x         : num [1:3604]  11.7 998.9 980.1 986.5 944.1 ...
 $ y         : num [1:3604] 151 430 434 426 415 ...
 $ markformat: chr "none"
 - attr(*, "class")= chr "ppp"
```

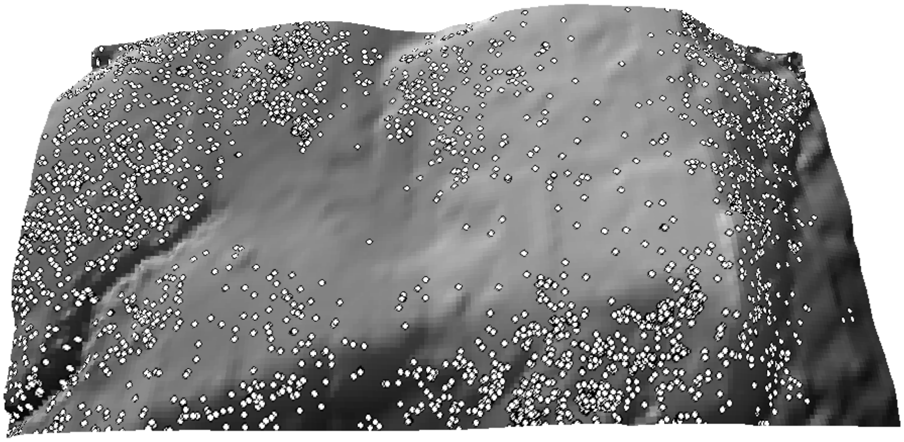


Fig. 8.1: The bei data set — locations of *Beilschmiedia pendula Lauraceae* trees, and a 5 m DEM used as environmental predictor. Viewed from south to north.

This shows locations of individual trees (a total of 3604 trees) observed in a 1000 m by 500 m rectangular window[3]. The beginning of the rectangle is at Lat. 9.15125°, Long. -79.8553°. The bei object is of type ppp, which is the native spatstat format (point pattern data set in the two-dimensional plane). If you would try to analyze this data set using some geostatistical package, you would soon find out that not much can be done because this is not even a binary variable.

8.1.2 Auxiliary maps

In addition to the point map a map of elevation and slope gradient is provided in the bea.extra data set:

```
> str(bei.extra, max.level=1)
```

```
List of 4
 $ elev :List of 11
  ..- attr(*, "class")= chr "im"
 $ grad :List of 11
  ..- attr(*, "class")= chr "im"
```

[3] What makes this data set especially suitable for this exercise is the fact that the complete population of the trees has been mapped for the area of interest.

We can extend the initial list of covariates and attach two more maps that we can derive in SAGA and then import back into R — the wetness index and vertical distance from the channel network (Fig. 8.2):

```
> grids <- as(bei.extra[[1]], "SpatialGridDataFrame")
> names(grids)[1] <- "elev"
> grids$grad <- as(bei.extra[[2]], "SpatialGridDataFrame")$v
> write.asciigrid(grids["elev"], "dem.asc")
> write.asciigrid(grids["grad"], "grad.asc")
# Generate the wetness index and vertical distance from the channel network:
> rsaga.esri.to.sgrd(in.grids="dem.asc", out.sgrds="dem.sgrd", in.path=getwd())
# Filter the spurious sinks:
> rsaga.geoprocessor(lib="ta_preprocessor", module=2,
+   param=list(DEM="dem.sgrd", RESULT="dem_f.sgrd"))
> rsaga.geoprocessor(lib="ta_hydrology", module=15, param=list(DEM="dem_f.sgrd",
+   C="catharea.sgrd", GN="catchslope.sgrd", CS="modcatharea.sgrd", SB="twi.sgrd", T=10))
> rsaga.geoprocessor(lib="ta_channels", module=0, param=list(ELEVATION="dem.sgrd",
+   CHNLNTWRK="chnlntwrk.sgrd", CHNLROUTE="channel_route.sgrd", SHAPES="channels.shp",
+   INIT_GRID="dem_f.sgrd", DIV_CELLS=10, MINLEN=30))
> rsaga.geoprocessor(lib="ta_channels", module=3, param=list(ELEVATION="dem.sgrd",
+   CHANNELS="chnlntwrk.sgrd", ALTITUDE="achan.sgrd", THRESHOLD=0.1, NOUNDERGROUND=TRUE))
> rsaga.sgrd.to.esri(in.sgrds=c("twi.sgrd","achan.sgrd"),
+   out.grids=c("twi.asc","achan.asc"), prec=1, out.path=getwd())
> grids$achan <- readGDAL("achan.asc")$band1
> grids$twi <- readGDAL("twi.asc")$band1
```

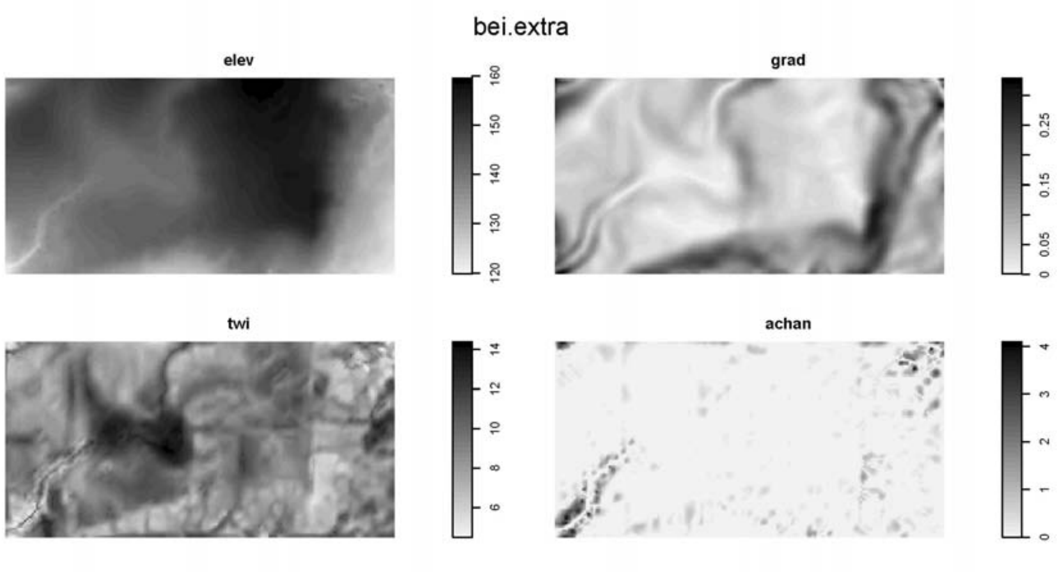


Fig. 8.2: Auxiliary (environmental) maps used to explain species' distribution: Elevation (`elev`), Slope in % (`grad`), Topographic Wetness Index (`twi`) and Altitude above channel network in m (`achan`) derived in SAGA GIS.

where `twi.sgrd` is the Topographic Wetness Index, and `achan.sgrd` is the Altitude above channel network. For more info about the SAGA syntax, see section 3.1.2 or refer to Conrad (2007).

We will now implement all steps described in §2.6 to predict the spatial density of trees over the area of interest (M=20301 grid nodes). To test our algorithm we will use a 20% sub-sample of the original population and then validate the accuracy of our technique versus the whole population.

8.2 Species distribution modeling

8.2.1 Kernel density estimation

We start by estimating a suitable bandwidth size for kernel density estimation (Eq.2.6.3). For this we use the method of Berman and Diggle (1989), as described in Bivand et al. (2008, p.166–167), that looks for the smallest Mean Square Error (MSE) of a kernel estimator:

```
> gridbbox <- as.points(list(x=c(grids@bbox[1,1], grids@bbox[1,2], grids@bbox[1,2],
+   grids@bbox[1,1]), y=c(grids@bbox[2,1], grids@bbox[2,1], grids@bbox[2,2], grids@bbox[2,2])))
> mserw <- mse2d(as.points(coordinates(bei.pnt)), gridbbox, 100, 10*bei.pixsize)
> bw <- mserw$h[which.min(mserw$mse)]
> plot(mserw$h,mserw$mse, type="l")
```

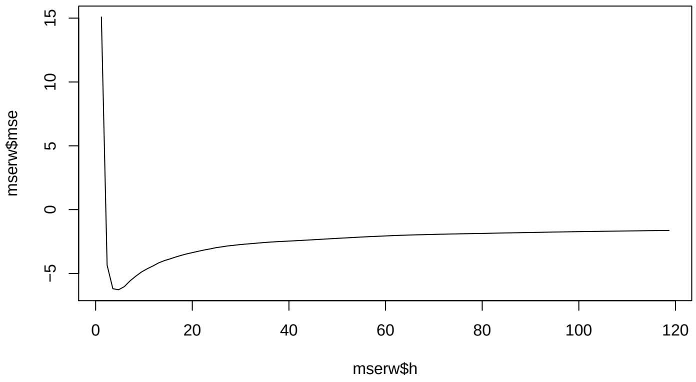


Fig. 8.3: Selection of the optimal bandwidth using the method of Berman and Diggle (1989).

This shows that the optimal bandwidth size is about 4 m. But since our grid cell size is 5 m this bandwidth is not really suited for this scale of work. Based on the plot from above we only know that we should not use a bandwidth finer/smaller than 5 m; coarser bandwidths are all plausible. We can also consider the least squares cross validation method to select the bandwidth size using the method of Worton (1995), as implemented in the adehabitat package:

```
> bei.pnt <- data.frame(x=bei$x, y=bei$y, no=rep(1, length(bei$x)))
> coordinates(bei.pnt) <- ~ x+y
> dem.asc <- import.asc("dem.asc")
> bei.kdens <- kernelUD(xy=as.data.frame(bei.pnt@coords), id=NULL, h="LSCV", grid=dem.asc)

  Warning message:
  In kernelUD(xy = as.data.frame(bei.pnt@coords), id = NULL, h = "LSCV",  :
    The algorithm did not converge within the specified range of hlim: try to increase it
```

This does not converge either[4], hence we need to set the bandwidth size using some *ad hoc* method. As a rule of thumb we can start by estimating the smallest suitable range as the average size of block:

$$p = \sqrt{area(B_{HR})/N} \tag{8.2.1}$$

and then set the bandwidth size at two times this value. There are 3605 trees (N) in the area of size 507,525 m^2, which means that we could use a bandwidth of 24 m (H):

[4] This is unfortunately a very common problem with many real point patterns.

```
> bei.pixsize <- sqrt(areaSpatialGrid(grids)/length(bei$x))
> bei.pixsize

  [1] 11.86687
```

We next derive a relative kernel density map using the standard methods in the spatstat package (Eq.2.6.7). The resulting density map is shown in Fig. 8.4:

```
> bei.ppp <- ppp(coordinates(bei.pnt)[,1], coordinates(bei.pnt)[,2],
+      marks=bei.pnt$no, window=as(grids[1], "owin"))
# reformat the point pattern so it fits the grids:
> summary(bei.ppp)

  Marked planar point pattern: 3604 points
  Average intensity 0.0071 points per square unit
  marks are  numeric,  of type 'double'
  Summary:
     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
        1       1       1       1       1       1

  Window: binary image mask
  101 x 201 pixel array (ny, nx)
  pixel size: 5 by 5 units
  enclosing rectangle: [-2.5, 1002.5] x [-2.5, 502.5] units
  Window area =  507525 square units


> bei.dens <- density(bei.ppp, sigma=2*bei.pixsize)
> grids$dens <- as(bei.dens, "SpatialGridDataFrame")$v
# relative density:
> grids$densr <- grids$dens/(max(grids$dens, na.rm=T))
> bei.pnt.plot <- list("sp.points", bei.pnt, pch="+", cex=.7, col="black")
> plt.dens1 <- spplot(grids["densr"], scales=list(draw=F), at=seq(0,1,0.025),
+      col.regions=grey(rev(seq(0,1,0.025))), sp.layout=bei.pnt.plot)
```

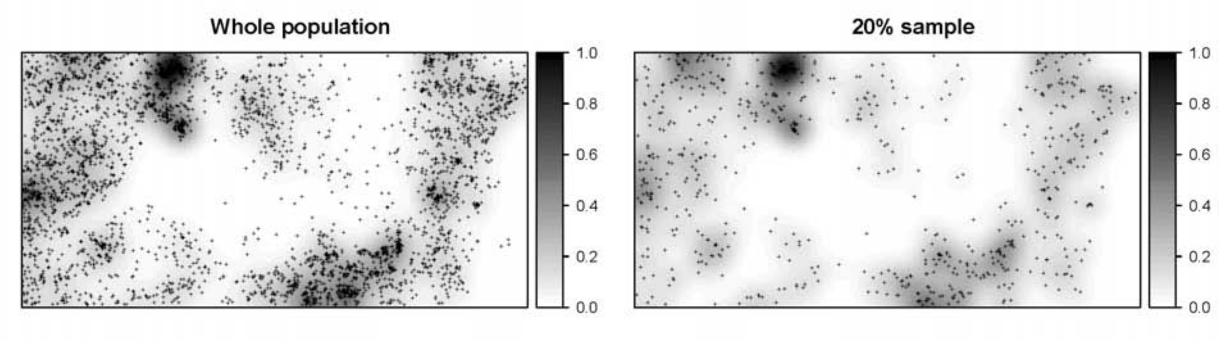


Fig. 8.4: Relative intensity estimated for the original bei data set (left), and its 20% sub-sample (right). In both cases the same bandwidth was used: H=24 m.

If we randomly subset the original occurrence locations and then re-calculate the relative densities, we notice that the spatial pattern of the two maps does not differ significantly, and neither do their histograms:

```
# Randomly subset points:
> sel <- runif(length(bei.pnt$no))<0.2
> bei.sub.pnt <- bei.pnt[sel,]
> bei.sub <- ppp(coordinates(bei.sub.pnt)[,1], coordinates(bei.sub.pnt)[,2],
+       marks=bei.sub.pnt$no, window=as(grids[1], "owin"))
# plot(bei.sub)
# Derive kernel density:
```

```
> bei.sub.dens <- density(bei.sub, sigma=2*bei.pixsize)
> grids$sub.dens <- as(bei.sub.dens, "SpatialGridDataFrame")$v
> grids$sub.densr <- grids$sub.dens/(max(grids$sub.dens, na.rm=T))
# Plot the second map:
> bei.sub.pnt.plot <- list("sp.points", bei.sub.pnt, pch="+", cex=.7, col="black")
> plt.dens2 <- spplot(grids["sub.densr"], scales=list(draw=F), at=seq(0,1,0.025),
+       col.regions=grey(rev(seq(0,1,0.025))), sp.layout=bei.sub.pnt.plot)
> print(plt.dens1, split=c(1,1,1,2), more=T)
> print(plt.dens2, split=c(1,2,1,2), more=T)
```

then check if the two maps follow the same distribution:

```
> t.test(grids$sub.densr, grids$densr)
> t.test(grids$sub.densr, grids$densr)

         Welch Two Sample t-test

  data:  grids$sub.densr and grids$densr
  t = -6.5351, df = 40567.5, p-value
  = 6.432e-11
  alternative hypothesis: true difference in means is not equal to 0
  95 percent confidence interval:
   -0.010186940 -0.005486209
  sample estimates:
  mean of x mean of y
  0.1125212 0.1203578
```

which confirms that the two maps basically follow the same distribution. This supports our assumption that the relative density map (Eq.2.6.7) can be indeed reproduced also from a representative sample (n=721).

We proceed with preparing the environmental predictors and testing their correlation with the density values. We can extend the original single auxiliary map (DEM) by adding some hydrological parameters: slope, topographic wetness index and altitude above channel network. We want to attach the `SpatialGridDataFrame` class maps to the original `im` class `bei.extra`, which means that we need to 'coerce' the objects to format of interest:

```
> bei.extra$twi <- as.im(as.image.SpatialGridDataFrame(grids["twi"]))
> bei.extra$achan <- as.im(as.image.SpatialGridDataFrame(grids["achan"]))
> plot.im(bei.extra, col=grey(rev(seq(0,1,0.025))))
```

which will produce the map shown in Fig. 8.2. The result of fitting a non-stationary point process with a log-linear density using the ppm method of spatstat shows that density is negatively correlated with wetness index, and positively correlated with all other predictors (Baddeley, 2008):

```
> bei.sub.ppm <- ppm(bei.sub, ~ elev+grad+twi+achan, covariates=list(elev=bei.extra$elev,
+     grad=bei.extra$grad, twi=bei.extra$twi, achan=bei.extra$achan))
> summary(bei.sub.ppm)

  Point process model
  fitted by maximum pseudolikelihood (Berman-Turner approximation)
  Call:
  ppm(Q = bei.sub, trend = ~ elev + grad + twi + achan,
  covariates = list(elev = bei.extra$elev,...
  Edge correction: "border"
  ----------------------------------------------------
  FITTED MODEL:

  Nonstationary Poisson process

   ---- Intensity: ----

  Trend formula: ~ elev + grad + twi + achan
  Model involves external covariates
```

```
Fitted coefficients for trend formula:
(Intercept)        elev        grad         twi       achan
-9.00309481  0.01675857  4.24494406 -0.07240637  0.03017035
```

We can actually use this model to predict the species density by using the generic `predict` method:

```
> bei.ppm.trend <- predict(bei.sub.ppm, type="trend", ngrid=c(grids@grid@cells.dim[2],
+       grids@grid@cells.dim[1]), window=as(grids[1], "owin"))
> plot(bei.ppm.trend)
> plot(bei.sub, add=T, pch=".")
```

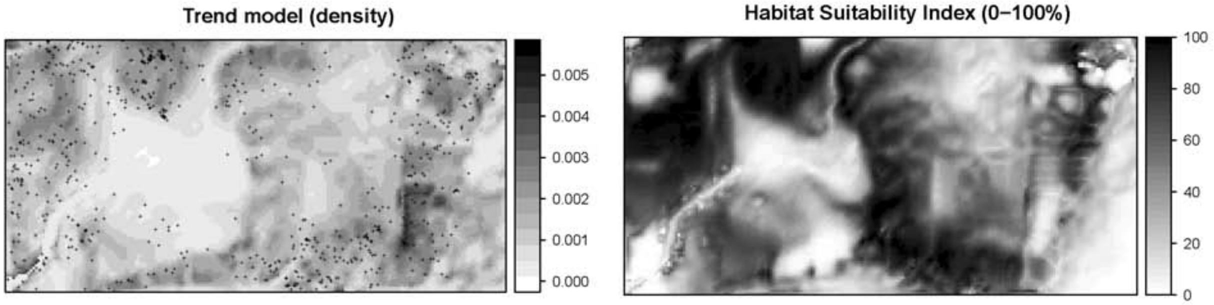


Fig. 8.5: Trend model "ppm" predicted using elevation, slope, topographic wetness index and altitude above channel network as environmental covariates (left); Habitat Suitability Index (0-100%) derived in the adehabitat package using the same list of covariates (right).

Visually (Fig. 8.5), we can see that the predicted trend seriously misses some hot-spots i.e. clusters of points. It seems that using point pattern analysis techniques to map (realized) species' distribution with covariates is of limited use. A comparison between the *Akaike Information Criterion* (AIC) for a model without predictors and with predictors shows that there is a slight gain in using the covariates to predict the spatial density:

```
> fitnull <- ppm(bei.sub, ~ 1)
> AIC(fitnull)
```

```
[1] 10496.56
```

```
> AIC(bei.sub.ppm)  # this is a slightly better model
```

```
[1] 10289.79
```

Just for a comparison, we can also fit a GLM model[5] using all grid nodes:

```
> bei.ppp.lm <- glm(sub.densr ~ grad+elev+twi+achan, grids@data, family=poisson())
```

```
There were 50 or more warnings (use warnings() to see the first 50)
```

```
> summary(bei.ppp.lm)
```

This shows a similar picture: the best predictor of the density is `TWI`.

[5] We need to use the log as the link function because the density values are heavily skewed.

8.2.2 Environmental Niche analysis

We proceed with the Environmental Niche Factor Analysis (Hirzel and Guisan, 2002), as implemented in the adehabitat[6] package (Calenge, 2007). Our objective is to derive the Habitat Suitability Index (0–100%) which basically shows potential spreading of a species in the feature space (environmental preference). We need to prepare the maps in the adehabitat native format:

```
> beidata <- data2enfa(as.kasc(list(dem=import.asc("dem.asc"), grad=import.asc("grad.asc"),
+     twi=import.asc("twi.asc"), achan=import.asc("achan.asc"))), bei.sub.pnt@coords)
# run ENFA and make predictions of habitat suitability index:
> enfa.bei <- enfa(dudi.pca(beidata$tab, scannf=FALSE), beidata$pr, scannf=FALSE, nf=2)
```

```
 Warning message:
 In predict.enfa(enfa.bei, beidata$index, beidata$attr) :
   the enfa is not mathematically optimal for prediction:
  please consider the madifa instead
```

```
> bei.dist <- predict(enfa.bei, beidata$index, beidata$attr)
> grids$bei.dist <- as.SpatialGridDataFrame.im(asc2im(bei.dist))$v
# Convert to 0-100 scale:
> sum.dist <- summary(grids$bei.dist)
> grids$rankv <- rank(grids$bei.dist, ties.method="first")
> grids$hsit <- ifelse(grids$bei.dist<sum.dist[["Median"]],
+     (1-grids$rankv/max(grids$rankv))*100, (1-grids$rankv/max(grids$rankv))*100)
> grids$hsi <- 100*round((grids$hsit-min(grids$hsit, na.rm=T))/(max(grids$hsit,
+     na.rm=T)-min(grids$hsit, na.rm=T)), 3)
> plt.HSI <- spplot(grids["hsi"], at=seq(0,100,2.5), col.regions=bpy.colors())
```

The resulting HSI map shows that this species generally avoids the areas of high wetness index, i.e. it prefers ridges/dry positions (Fig. 8.5, right). This spatial pattern is now more distinct (compare with the trend model in Fig. 8.5, left). This demonstrates the power of ENFA, which is in this case more suited for analysis of the occurrence-only locations than regression analysis and/or the point pattern analysis.

8.2.3 Simulation of pseudo-absences

In the next section we will use the results of Niche analysis to generate pseudo-absences. Recall (§2.6) that insertion of pseudo-absences is an important step because it will allow us to fit regression models. We use two maps as weight maps to randomize allocation of the pseudo-absences: the geographical distance from the occurrence locations and habitat suitability index. We first derive the buffer distance to the points in SAGA GIS:

```
# first the buffer distance:
> rsaga.geoprocessor(lib="grid_gridding", module=3, param=list(GRID="bei_buffer.sgrd",
+    INPUT="bei_sub.shp", FIELD=0, LINE_TYPE=0, USER_CELL_SIZE=grids@grid@cellsize[[1]],
+    USER_X_EXTENT_MIN=grids@bbox[1,1]+grids@grid@cellsize[[1]]/2,
+    USER_X_EXTENT_MAX=grids@bbox[1,2]-grids@grid@cellsize[[1]]/2,
+    USER_Y_EXTENT_MIN=grids@bbox[2,1]+grids@grid@cellsize[[1]]/2,
+    USER_Y_EXTENT_MAX=grids@bbox[2,2]-grids@grid@cellsize[[1]]/2))
# now extract a buffer distance map and load it back into R:
# (the parameters DIST and IVAL are estimated based on the grid properties)
> rsaga.geoprocessor(lib="grid_tools", module=10, param=list(SOURCE="bei_buffer.sgrd",
+   DISTANCE="bei_dist.sgrd", ALLOC="bei_alloc.sgrd", BUFFER="bei_bdist.sgrd",
+   DIST=sqrt(areaSpatialGrid(grids))/3, IVAL=grids@grid@cellsize[[1]]))
> rsaga.sgrd.to.esri(in.sgrds="bei_dist.sgrd", out.grids="bei_dist.asc",
+   out.path=getwd(), prec=1)
> grids$buffer <- readGDAL("bei_dist.asc")$band1
# relative distance:
> grids$bufferr <- grids$buffer/max(grids$buffer, na.rm=T)
```

[6] http://cran.r-project.org/web/packages/adehabitat/

By combining HSI and buffer map around the occurrence locations (Eq.2.6.11) we can generate the same amount of pseudo-absence locations:

```
# weight map:
> grids$weight <- ((100*grids$bufferr+(100-grids$hsi))/2)^2
> dens.weight <- as.im(as.image.SpatialGridDataFrame(grids["weight"]))
> bei.absences <- rpoint(length(bei.sub.pnt$no), f=dens.weight)
> bei.absences <- data.frame(x=bei.absences$x, y=bei.absences$y,
+    no=rep(0, length(bei.sub.pnt$no)))
> coordinates(bei.absences) <- ~ x+y
# combine the occurences and absences:
> bei.all <- rbind(bei.sub.pnt["no"], bei.absences["no"])
```

One realization of the simulated pseudo-absences (overlaid over the weight map) can be seen in Fig. 8.10a. Correct insertion of pseudo-absences is crucial for the success of regression analysis (see Chefaoui and Lobo (2008) for discussion), hence it also a good idea to visually validate each simulated pseudo-absence and remove ‘suspicious’ candidates.

8.2.4 Regression analysis and variogram modeling

Before we run any regression analysis we will convert the original four predictors to principal components (in order to reduce their dimensions and the multicolinearity effect):

```
# derive PCs from the original predictors:
> pc.grids <- prcomp( ~ grad+elev+twi+achan, scale=TRUE, grids)
> pc.comps <- as.data.frame(pc.grids$x)
> pointgrids <- as(grids, "SpatialPointsDataFrame")
> pc.comps$X <- coordinates(pointgrids)[,1]
> pc.comps$Y <- coordinates(pointgrids)[,2]
> coordinates(pc.comps) <- ~ X+Y
> gridded(pc.comps) <- TRUE
> pc.comps <- as(pc.comps, "SpatialGridDataFrame")
> spplot(pc.comps, c("PC1","PC2","PC3","PC4"),
+    col.regions=grey(c(rep(0,10),seq(0,1,0.05),rep(1,10))), cuts=40)
```

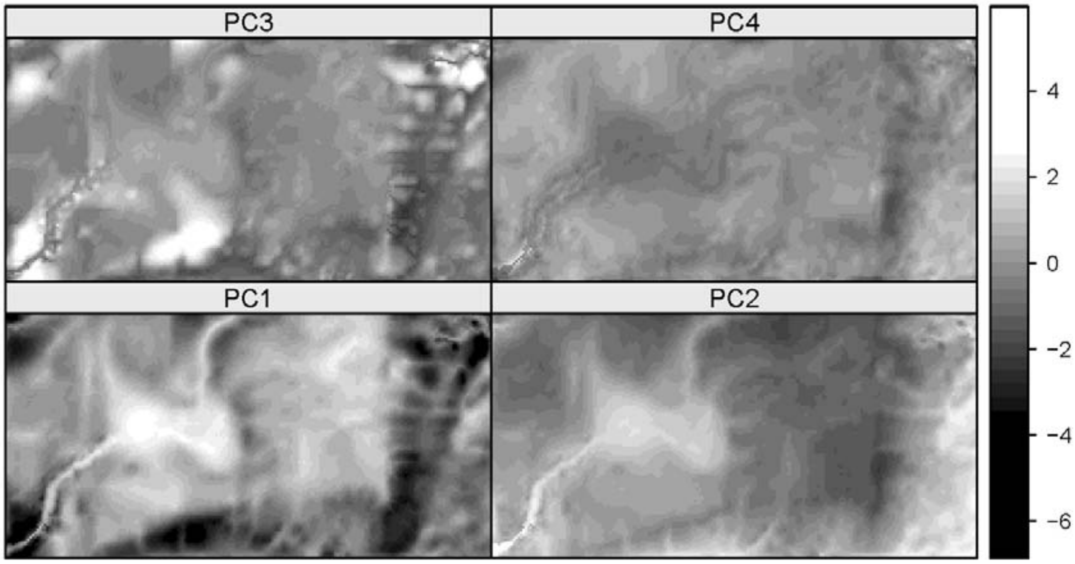


Fig. 8.6: Principal Components generated using the original predictors.

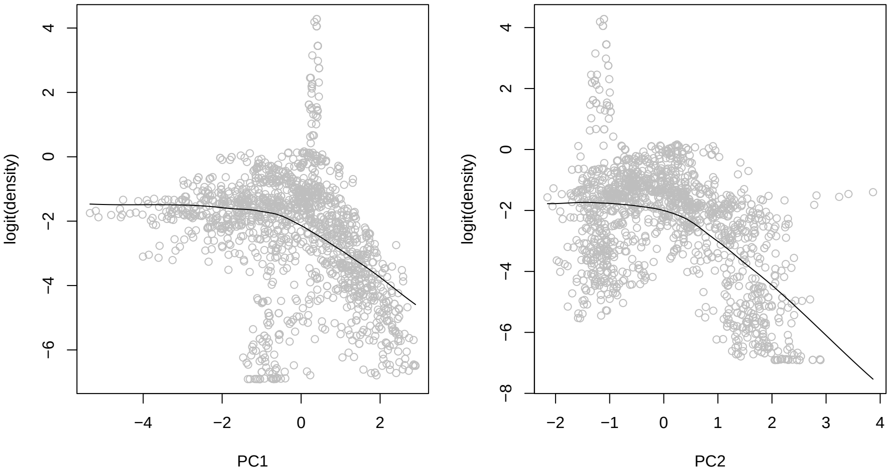


Fig. 8.7: Correlation plot PC1/PC2 versus the *logit*-transformed density.

which shows that the first component reflects pattern in the `twi` and `grad`; the third component reflects pattern in `achan` (Fig. 8.6). Next, we can overlay the predictors and the estimated intensities at occurrence and absence locations:

```
> bei.ov2 <- overlay(pc.comps, bei.all)
> bei.ov2$no <- bei.all$no
# convert the original values to logits:
> bei.ov2$log.densr <- log((bei.ov2$densr+0.001)/(1-(bei.ov2$densr-0.001)))
> summary(bei.ov2$log.densr)

    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -6.908  -3.378  -1.882  -2.297  -1.197   4.253


> var(bei.ov2$log.densr, na.rm=T)

  [1] 3.666165
```

The benefit of converting the target variable to logits and the predictors to PCs is that the relationship between the two is now *close* to linear (Fig. 8.7):

```
> scatter.smooth(bei.ov2$PC1, bei.ov2$log.densr, span=19/20, col="grey",
+    xlab="PC1", ylab="logit(density)")
> lm3.bei <- lm(log.densr ~ PC1+PC2+PC3+PC4, bei.ov2@data)
> summary(lm3.bei)

  Call:
  lm(formula = log.densr ~ PC1 + PC2 + PC3 + PC4, data = bei.ov2@data)

  Residuals:
       Min       1Q   Median       3Q      Max
  -4.09463 -0.98535  0.04709  0.99504  5.52267

  Coefficients:
              Estimate Std. Error t value
  (Intercept) -2.35896    0.04247 -55.546
  PC1         -0.40174    0.02864 -14.025
  PC2         -0.79147    0.03684 -21.482
  PC3         -0.34820    0.04629  -7.522
  PC4          0.34360    0.07423   4.629
```

```
            Pr(>|t|)
(Intercept)  < 2e-16 ***
PC1          < 2e-16 ***
PC2          < 2e-16 ***
PC3         9.65e-14 ***
PC4         4.02e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.559 on 1379 degrees of freedom
Multiple R-squared: 0.3441,     Adjusted R-squared: 0.3422
F-statistic: 180.9 on 4 and 1379 DF,  p-value: < 2.2e-16
# the model explains  34% of variation
```

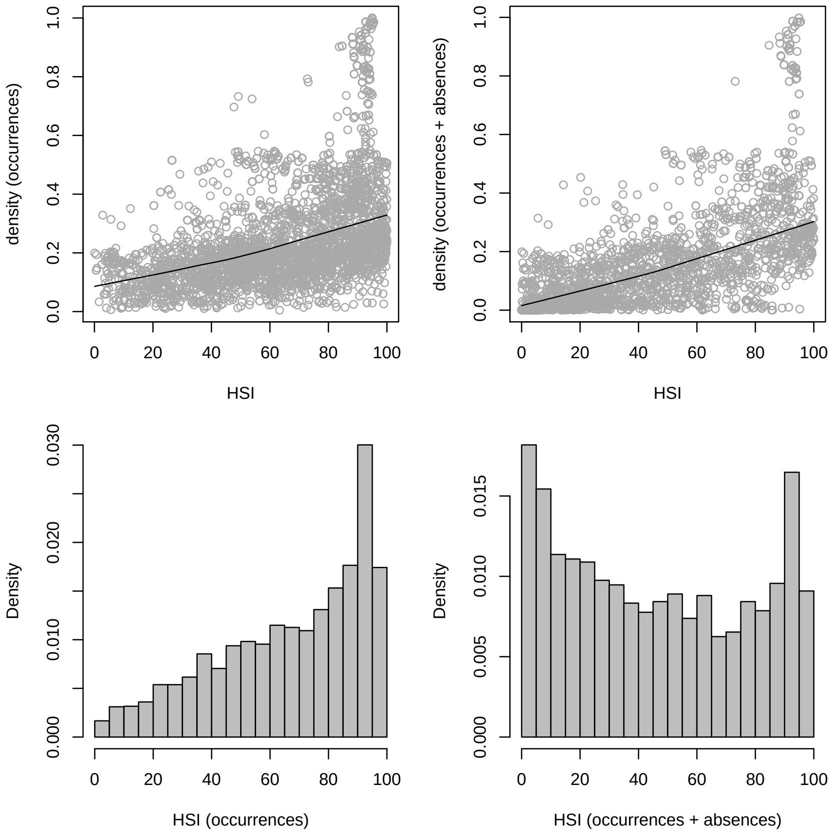


Fig. 8.8: Correlation plot HSI vs relative density with occurrence-only locations (left) and after the insertion of the pseudo-absence locations (right). Note that the pseudo-absences ensure equal spreading around the feature space (below).

The correlation between the HSI and density is now clearer and the spreading of the points around the HSI feature space is symmetric:

```
# Glue occurrences and pseudo-absences:
> bei.all <- rbind(bei.sub.pnt["no"], bei.absences["no"])
> scatter.smooth(bei.ov2$hsi[bei.ov2$no>0], bei.ov2$densr[bei.ov2$no>0],  span=19/20,
+   col="darkgrey", pch=21, cex=.5, xlab="HSI", ylab="density (occurrences-only)")
> scatter.smooth(bei.ov2$hsi, bei.ov2$densr, span=19/20, col="darkgrey", pch=21,
+   cex=.5, xlab="HSI", ylab="density (+ absences)")
> hist(bei.ov2$hsi[bei.ov2$no>0], col="grey", freq=F, breaks=30,
+   xlab="HSI (occurrences)", ylab="frequency", main="")
> hist(bei.ov2$hsi, col="grey", freq=F, breaks=30,
+   xlab="HSI (occurrences + absences)", ylab="frequency", main="")
```

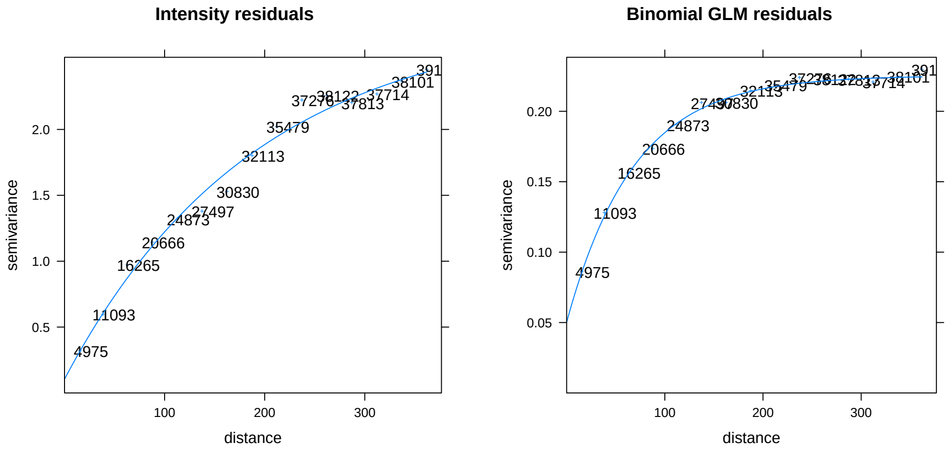


Fig. 8.9: Variogram models for residuals fitted in gstat using occurrence-absence locations: (left) density values (logits), and (right) probability values.

which produces the plot shown in Fig. 8.8 (right). Consequently, the model fitting is more successful: the adjusted R-square fitted using the four environmental predictors jumped from 0.07 to 0.34. This demonstrates the benefits of inserting the pseudo-absence locations using a feature-space design model. It can be further demonstrated that, if we would randomly insert the pseudo-absences, the model would not improve or would become even noisier.

We proceed with analyzing the point data set indicated in Fig. 8.10b using standard geostatistical tools. We can fit a variogram for the residuals and then predict relative densities using regression-kriging as implemented in the gstat package. For a comparison, we also fit a variogram for the occurrence-absence data but using the residuals of the GLM modeling with the logit link function, i.e. using the 0/1 values (Fig. 8.9):

```
> glm.bei <- glm(no ~ PC1+PC2+PC3+PC4, bei.ov2@data, family=binomial())
```

We can now deal with the residuals by using some standard geostatistical techniques. To fit a variogram for the residuals, we run:

```
> res.var <- variogram(residuals(lm3.bei) ~ 1, bei.ov2)
> res.vgm <- fit.variogram(res.var, vgm(nugget=0, model="Exp",
+   range=sqrt(areaSpatialGrid(grids))/3, psill=var(residuals(lm3.bei), na.rm=T)))
> plot(res.var, res.vgm, plot.nu=T, pch="+", main="Intensity residuals")
```

As with any indicator variable, the variogram of the binomial GLM will show higher nugget and less distinct auto-correlation than the variogram for the (continuous) density values (Fig. 8.9). This is also because the residuals of the density values will still reflect kernel smoothing, especially if the predictors explain only a small part of variation in the density values.

8.3 Final predictions: regression-kriging

We can finally generate predictions using a regression-kriging model[7]:

```
# regression part:
> vt.gt <- gstat(id=c("dens"), formula=log.densr ~ PC1+PC2+PC3+PC4, data=bei.ov2)
# residuals:
> vt.gr <- gstat(id=c("dens"), formula=residuals(lm3.bei) ~ 1,
+      data=bei.ov2, model=res.vgm)
> vt.reg <- predict.gstat(vt.gt, pc.comps) # regression part
> vt.ok <- predict.gstat(vt.gr, pc.comps, nmax=80, beta=1, BLUE=FALSE,
+      debug.level=-1)
```

[7] Residuals and the deterministic part are predicted separately.

```
# Back-transform:
> vt.reg$dens <- exp(vt.reg$dens.pred+vt.ok$dens.pred)/
+       (1+exp(vt.reg$dens.pred+vt.ok$dens.pred))
> vt.reg$densA <- vt.reg$dens * length(bei.pnt$no)/sum(vt.reg$dens, na.rm=F)
> sum(vt.reg$densA) # check if the sum of counts equals the population!

  [1] 3604
```

```
> bei$n

  [1] 3604
```

To predict the probability of species' occurrence we run similar steps:

```
> bin.reg <- predict(glm.bei, newdata=pc.comps, type="response", na.action=na.omit)
> bin.gr <- gstat(id=c("no"), formula=glm.bei$residual ~ 1, data=bei.ov2, model=res.bvgm)
> bin.ok <- predict.gstat(bin.gr, pc.comps, nmax=80, beta=1, BLUE=FALSE, debug.level=-1)
> bin.ok$rk.bin <- bin.reg$fit + bin.ok$no.pred
> bin.ok$rk.binf <- ifelse(bin.ok$rk.bin<0, 0, ifelse(bin.ok$rk.bin>1, 1, bin.ok$rk.bin))
```

The resulting map of density predicted using regression-kriging (shown in Fig. 8.10c) is indeed a hybrid map that reflects kernel smoothing (hot spots) and environmental patterns, thus it is a map richer in content than the pure density map estimated using kernel smoothing only (Figs. 8.4), or the Habitat Suitability Index (Fig. 8.5, right). Note also that, although the GLM-kriging with a binomial link function (Fig. 8.10d) is statistically a more straight forward procedure (it is completely independent from point pattern analysis), its output is limited in content because it also fails to represent some hot-spots. GLM-kriging of 0/1 values in fact only shows the areas where a species' is likely to occur, without any estimation of how dense will the population be. Another advantage of using the occurrences+absences with attached density values is that we are able not only to generate predictions, but also to generate geostatistical simulations, map the model uncertainty, and run all other common geostatistical analysis steps.

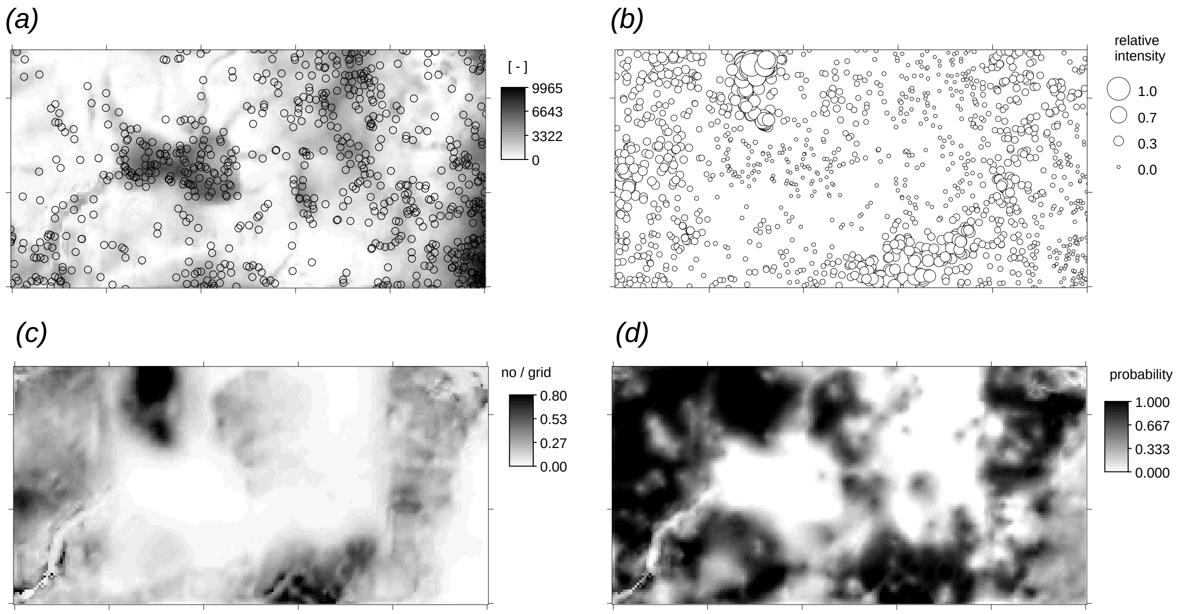


Fig. 8.10: Spatial prediction of species distribution using the bei data set (20% sub-sample): (a) the weight map and the randomly generated pseudo-absences using the Eq.(2.6.11); (b) input point map of relative intensities (includes the simulated pseudo-absences); (c) the final predictions of the overall density produced using regression-kriging (showing number of individuals per grid cell as estimated using Eq.(2.6.8); and (d) predictions using a binomial GLM.

In the last step of this exercises we want to validate the model performance using the ten-fold cross validation (as implemented in the gstat package):

```
> vt.cross <- krige.cv(log.densr ~ PC1+PC2+PC3+PC4, bei.ov2, res.vgm, nfold=10)
# 10-fold cross-validation
# MPE, ideally 0 (unbiased estimate)
> mean(vt.cross$residual)

  [1] -0.004776853

# MSPE, ideally small
> var(vt.cross$residual, na.rm=T)

  [1] 0.01412567

# portion of variation explained by the model:
> 1-var(vt.cross$residual, na.rm=T)/var(bei.ov2$log.densr, na.rm=T)

  [1] 0.9961782
```

which shows that the model is highly precise — it explains over 99% of the variance in the training samples. Further comparison between the map shown in Fig. 8.10c and Fig. 8.4 shows that with 20% of samples and four environmental predictors we are able to explain 96% of the pattern in the original density map (R-square=0.96). Fig. 8.11 indeed confirms that this estimator is unbiased. These are rather high values[8] and you will possibly not expect to achieve such high accuracy with your own data.

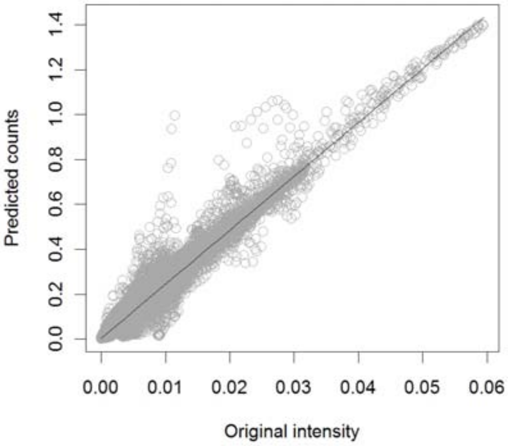


Fig. 8.11: Evaluation of the mapping accuracy for the map shown in Fig. 8.10c versus the original mapped density using 100% of samples (Fig. 8.4).

One last point: although it seems from this exercise that we are recycling auxiliary maps and some analysis techniques (we use auxiliary maps both to generate the pseudo-absences and make predictions), we in fact use the HSI map to generate the pseudo-absences, and the original predictors to run predictions, which do not necessarily need to reflect the same features. Relative densities do not have to be directly correlated with the HSI, although a significant correlation will typically be anticipated. Likewise, we use kernel smoother to estimate the intensities, but we then fit a variogram, which is controlled by the amount of smoothing, i.e. value of the bandwidth, hence the variogram will often show artificially smooth shape (as shown in Fig. 8.9). The only way to avoid this problem is to estimate the bandwidth using some objective technique (which we failed to achieve in this example), or to scale the variogram fitted for the indicator variable (Fig. 8.9; right) to the density values scale.

8.4 Niche analysis using MaxEnt

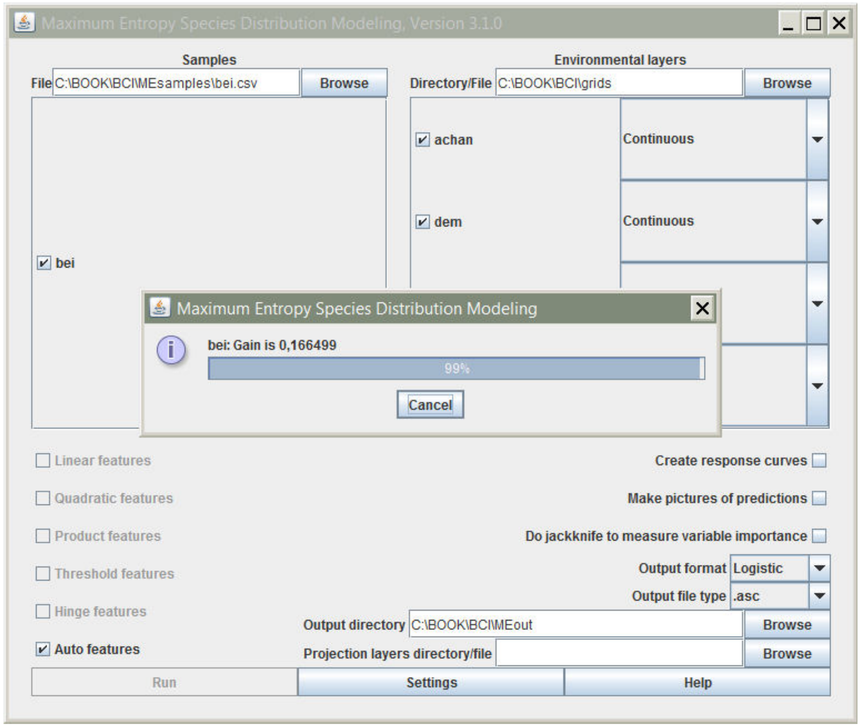
In section §8.2.2 we have derived Habitat Suitability Index map using the adehabitat package. An alternative approach to estimate species ecological preference is by using the MaxEnt package (Phillips and Dudík, 2008), which is by many considered to be the most robust approach to species distribution modeling (see §2.6). For example, MaxEnt can work with both continuous and categorical predictors and has very extensive and flexible possibilities for analysis of biodiversity data. With MaxEnt, we can request cross-validation and investigate which predictors are most significant, which is not as simple with adehabitat.

MaxEnt is not available as an R package, therefore you will first need to request and download it from the MaxEnt homepage[9]. The complete algorithm is contained in a single `maxent.jar` (**J**ava **AR**chive) file, which is basically a zipped Java (class file) code[10]. Then, define location of MaxEnt in the R session:

[8] This is an ideal data set where all individual trees have been mapped and the species' distribution is systematic.

[9] `www.cs.princeton.edu/~schapire/maxent/`

[10] Obviously, before you can run MaxEnt, you will need to install **Java** software on your machine.



Maximum Entropy Species Distribution Modeling, Version 3.1.0

Samples

File C:\BOOK\BCI\MEsamples\bei.csv Browse

☑ bei

Environmental layers

Directory/File C:\BOOK\BCI\grids Browse

☑ achan Continuous

☑ dem Continuous

Maximum Entropy Species Distribution Modeling

bei: Gain is 0,166499

99%

Cancel

☐ Linear features

☐ Quadratic features

☐ Product features

☐ Threshold features

☐ Hinge features

☑ Auto features

Create response curves ☐

Make pictures of predictions ☐

Do jackknife to measure variable importance ☐

Output format Logistic

Output file type .asc

Output directory C:\BOOK\BCI\MEout Browse

Projection layers directory/file Browse

Fig. 8.12: Extraction of Habitat Suitability Index using MaxEnt.

```
# Location of MaxEnt and directories:
> Sys.chmod(getwd(), mode="7777")  # write permissions
> MaxEnt <- "C:\\MaxEnt\\maxent.jar"
> dir.create(path="MEout"); dir.create(path="MEsamples")
> MaxEnt.layers <- paste(gsub("/", "\\\\", getwd()), "\\grids", sep="")
> MaxEnt.out <- paste(gsub("/", "\\\\", getwd()), "\\MEout", sep="")
> MaxEnt.samples <- paste(gsub("/", "\\\\", getwd()), "\\MEsamples", sep="")
```

where `MEout` is the directory where MaxEnt will write the results of analysis (plots, grids and table data), and `MEsamples` is a directory containing the input samples. Next, copy the grids of interest to some working directory e.g. `/grids`:

```
> dir.create(path="grids")
> for(j in c("dem.asc", "grad.asc", "twi.asc", "achan.asc")) {
>   file.copy(j, paste("grids/", j, sep=""), overwrite=TRUE)
> }
> asc.list <- list.files(paste(getwd(), "/grids", sep=""),
+      pattern="\\.asc$", recursive=TRUE, full=FALSE)
> asc.list

  [1] "achan.asc" "dem.asc"   "twi.asc"
```

Before we can run MaxEnt, we still need to prepare the occurrence records in the required format (`.csv`):

```
# Write records to a csv file (species, longitude, latitude):
> bei.csv <- data.frame(sp=rep("bei", bei$n), gx=bei$x, gy=bei$y)
> write.csv(bei.csv[,c("sp","gx","gy")], "MEsamples/bei.csv", quote=FALSE, row.names=FALSE)
```

We can now run MaxEnt using the `system` command in R:

```
# Run a batch process (opens a process window):
> system(command=paste("java -mx1000m -jar ", MaxEnt, " environmentallayers=",
+    MaxEnt.layers, " samplesfile=", MaxEnt.samples, "\\bei.csv", " outputdirectory=",
+    MaxEnt.out, " randomtestpoints=25 maximumiterations=100 redoifexists
+    autorun nowarnings notooltips", sep=""))
```

where randomtestpoints=25 will randomly take 25% points for cross-validation, redoifexists will replace the existing files, autorun, nowarnings and notooltips will force MaxEnt to run without waiting. For more information about MaxEnt batch mode flags, look in the Help file.

After the analysis, open the /MEout directory and browse the generated files. We are most interested in two files: bei.asc — the Habitat Suitability Index map (0–1), and bei.html — the complete report from the analysis. Optionally, you can also read the result of analysis directly to R by using the XML package:

```
> library(XML)
> bei.html <- htmlTreeParse("MEout/bei.html")
# summary result:
> bei.html[[3]][[1]][[2]][[65]]; bei.html[[3]][[1]][[2]][[52]];
+     bei.html[[3]][[1]][[2]][[56]]; bei.html[[3]][[1]][[2]][[67]]
```

```
12703 points used to determine the Maxent distribution
(background points and presence points).
Regularized training gain is 0.167, training AUC is 0.712,
 unregularized training gain is 0.193.
Test AUC is 0.704, standard deviation is 0.008
Environmental layers used (all continuous): achan dem grad twi
```

which shows that AUC (Area Under Curve) statistics for cross-validation locations is 0.704 (maximum AUC is 1.0). Note that the HSI map derived using ENFA and MaxEnt are in fact very similar. MaxEnt also shows the most important predictors that can be used to explain distribution of bei trees are twi (40.6%) and dem (32.5%).

Self-study exercises:

(1.) Delineate all areas with >50 trees per ha. How big is this area in ha?

(2.) Randomly allocate 300 points and derive an HSI map using these points. Is there a significant difference between the map you derived and the map shown in Fig. 8.5 (right)?

(3.) Derive additional four DEM parameters in SAGA: two based on the lightning module and two based on the residual analysis (see p.230) and repeat the regression modeling explained in §8.2.4 using the extended list of predictors. Which predictor is now the most significant?

(4.) Produce a factor type map showing: (1) actual spreading (predicted probability exceeds 0.5), (2) potential spreading (HSI> 50%), and (3) no spreading using the original data set.

(5.) Run 4–6 geostatistical simulations using the model fitted in §8.2.4. Where is the distribution of the species most variable? Relate to both geographical and state space.

(6.) Try to allocate the pseudo-absences (section 8.2.3) using even higher spreading at the edges of the feature space. Determine the optimal spreading based on the improvement in the R-square.

(7.) Compare HSI maps derived in adehabitat and MaxEnt (using the same inputs) and derive correlation coefficient between the two maps.

Further reading:

★ Baddeley, A., 2008. **Analysing spatial point patterns in R**. CSIRO, Canberra, Australia.

★ Calenge, C., 2007. Exploring Habitat Selection by Wildlife with adehabitat. Journal of Statistical Software 22(6), 2–19.

★ Diggle, P. J., 2003. **Statistical Analysis of Spatial Point Patterns**, 2nd Edition. Arnold Publishers.

★ Hirzel A. H., Hausser J., Chessel D., Perrin N., 2002. Ecological-niche factor analysis: How to compute habitat-suitability maps without absence data? Ecology, 83, 2027–2036.

★ Pebesma, E. J., Duin, R. N. M., Burrough, P.A., 2005. Mapping sea bird densities over the North Sea: spatially aggregated estimates and temporal changes. Environmetrics 16(6), 573–587.

★ Phillips, S. J., Dudík, M., 2008. Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. Ecography, 31(2): 161-175.

★ Tsoar, A., Allouche, O., Steinitz, O., Rotem, D., Kadmon, R., 2007. A comparative evaluation of presence-only methods for modeling species distribution. Diversity & Distributions 13(9), 397–405.

9

Geomorphological units (fishcamp)

9.1 Introduction

The purpose of this exercise is to: (1) generate and filter a DEM from point data, and use it to derive various DEM parameters; (2) extract landform classes using an objective procedure (fuzzy *k*-means algorithm); and (3) improve the accuracy of soil mapping units using an existing map. We will also use geostatistical tools to assess variogram parameters for various landform classes to see if there are significant differences between them. For geographical analysis and visualization, we will exclusively use SAGA GIS (Brenning, 2008); an equally good alternative to run a similar analysis is GRASS GIS[1] (Neteler and Mitasova, 2008).

We will use three standard elevation data sets common for contemporary geomorphometry applications: point-sampled elevations (LiDAR), contours lines digitized from a topo map, and a raster of elevations sampled using a remote sensing system. All three elevation sources (`lidar.shp`, `contours.shp` and `DEMSRTM1.asc`) refer to the same geographical area — a 1×2 km case study `fishcamp` located in the eastern part of California (Fig. 9.4). This area is largely covered with forests; the elevations range from 1400 to 1800 meters. The data set was obtained from the USGS National Map seamless server[2]. The map of soil mapping units was obtained from the Natural Resources Conservation Service (NRCS) Soil Data Mart[3]. There are six soil mapping units: (1) Holland family, 35 to 65% slopes; (2) Chaix-chawanakee family-rock outcrop complex; (3) Chaix family, deep, 5 to 25% slopes; (4) Chaix family, deep, 15 to 45% slopes, (5) Holland family, 5 to 65% slopes, valleys; (6) Chaix-chawanakee families-rock outcrop complex, hilltops. The complete data set shown in this chapter is available via the geomorphometry.org website[4]; the scripts used to predict soil mapping units and extract landforms are available via the book's homepage.

There are basically two inputs to a supervised extraction of landforms (shown in Fig. 9.5): (1) raw elevation measurements (either points or un-filtered rasters); (2) existing polygon map i.e. the expert knowledge. The raw elevations are used to generate the initial DEM, which typically needs to be filtered for artifacts. An expert then also needs to define a set of suitable **Land Surface Parameters** (LSPs) that can be used to parameterize the features of interest. In practice, this is not trivial. On one hand, classes from the geomorphological or soil map legend are often determined by their morphology; hence we can easily derive DEM parameters that describe shape (curvatures, wetness index), hydrologic context (distance from the streams, height above the drainage network) or climatic conditions (incoming solar radiation). On the other hand, many classes are defined by land surface and sub-surface (geological) parameters that are difficult to obtain and often not at our disposal. Hence, the results of mapping soil and landform units will often be of limited success, if based only on the DEM and its derivatives. Please keep that in mind when running similar types of analysis with your own data.

This chapter is largely based on the most recent book chapter for the Geomorphological Mapping handbook by Seijmonsbergen et al. (2010). An introduction to some theoretical considerations connected with the

[1] `http://grass.itc.it`

[2] `http://seamless.usgs.gov`

[3] `http://soildatamart.nrcs.usda.gov`

[4] `http://geomorphometry.org/content/fishcamp`

geostatistical modeling of land surface topography is given in section 2.7.

9.2 Data download and exploration

Open the R script (`fishcamp.R`) and start preparing the data. First, download the `fischamp.zip` complete data set and layers of interest:

```
> download.file("http://geomorphometry.org/system/files/fishcamp.zip",
+     destfile=paste(getwd(), "fishcamp.zip", sep="/"))
# LiDAR points:
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("lidar", j, sep=""),
+       zipname="fishcamp.zip")
>   file.copy(fname, paste("./lidar", j, sep=""), overwrite=TRUE)
> }
# contour lines:
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("contours", j, sep=""),
+       zipname="fishcamp.zip")
>   file.copy(fname, paste("./contours", j, sep=""), overwrite=TRUE)
> }
# streams:
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("streams", j, sep=""),
+       zipname="fishcamp.zip")
>   file.copy(fname, paste("./streams", j, sep=""), overwrite=TRUE)
> }
# SRTM DEM:
> fname <- zip.file.extract(file="DEMSRTM1.asc", zipname="fishcamp.zip")
> file.copy(fname, "./DEMSRTM1.asc", overwrite=TRUE)
# soil map:
> fname <- zip.file.extract(file="soilmu.asc", zipname="fishcamp.zip")
> file.copy(fname, "./soilmu.asc", overwrite=TRUE)
```

where `lidar.shp` is a point map (LiDAR ground reflections), `contours.shp` is a map of contours (lines) digitized from a topo map, and `DEMSRTM1.asc` is the 1 arcsec (25 m) Shuttle Radar Topography Mission (SRTM) DEM.

To load LiDAR point data set, SRTM DEM, and the soil map to R, we use functionality of the rgdal package:

```
> lidar <- readOGR("lidar.shp", "lidar")

 OGR data source with driver: ESRI Shapefile
 Source: "lidar.shp", layer: "lidar"
 with  273028  rows and  1  columns
 Feature type: wkbPoint with 2 dimensions
```

```
> grids25m <- readGDAL("DEMSRTM1.asc")

 DEMSRTM1.asc has GDAL driver AAIGrid
 and has 40 rows and 80 columns
```

```
> grids5m <- readGDAL("soilmu.asc")

 soilmu.asc has GDAL driver AAIGrid
 and has 200 rows and 400 columns
```

this shows that `lidar.shp` contains 273,028 densely sampled points. The original LiDAR data set consist of, in fact, over 5 million of points; these points were subsampled for the purpose of this exercise, i.e. to speed up the processing. Next, we can estimate the approximate grid cell size[5] based on the density of points in the area of interest:

[5] We can take a rule of thumb that there should be at least 2 points per grid cell.

```
> pixelsize <- round(2*sqrt(areaSpatialGrid(grids25m)/length(lidar$Z)),0)
> pixelsize

  [1] 5
```

and then also attach the correct coordinate system to each spatial object:

```
> proj4string(lidar) <- CRS("+init=epsg:26911")
> proj4string(lidar)

  [1] " +init=epsg:26911 +proj=utm +zone=11 +ellps=GRS80
  +      +datum=NAD83 +units=m +no_defs +towgs84=0,0,0"
```

which is the UTM coordinate system with North American Datum 83 and GRS80 ellipsoid[6]. This allows us to obtain the geographical coordinates of the study area:

```
> proj4string(grids25m) <- CRS("+init=epsg:26911")
# coordinates of the center:
> grids25m.ll <- spTransform(grids25m, CRS("+proj=longlat +ellps=WGS84"))
> grids25m.ll@bbox

          min        max
  x -119.6232 -119.60060
  y   37.4589   37.46817

> clon <- mean(grids25m.ll@bbox[1,])
> clat <- mean(grids25m.ll@bbox[2,])
```

9.3 DEM generation

9.3.1 Variogram modeling

The first data analysis step is generation of a DEM from the LiDAR point data. Here geostatistics can provide a lot of information. First, we can determine how smoothly elevation varies in space, are the measurements noisy, is the feature of interest anisotropic? This type of analysis can be run by using e.g. the gstat package. However, fitting a variogram model with such a large point data set will take a long time on standard desktop PC. Instead, an equally reliable variogram can be produced by taking a random sub-sample of the measurements:

```
> lidar.sample <- lidar[runif(length(lidar$Z))<0.05,]
> varmap.plt <- plot(variogram(Z ~ 1, lidar.sample, map=TRUE, cutoff=50*pixelsize,
+       width=pixelsize), col.regions=grey(rev(seq(0,1,0.025))))
> Z.svar <- variogram(Z ~ 1, lidar.sample, alpha=c(45,135)) # cutoff=50*dem.pixelsize
> Z.vgm <- fit.variogram(Z.svar, vgm(psill=var(lidar.sample$Z), "Gau",
+       sqrt(areaSpatialGrid(grids25m))/4, nugget=0, anis=c(p=135, s=0.6)))
> vgm.plt <- plot(Z.svar, Z.vgm, plot.nu=F, cex=2, pch="+", col="black")
# plot the two variograms next to each other:
> print(varmap.plt, split=c(1,1,2,1), more=T)
> print(vgm.plt, split=c(2,1,2,1), more=F)
```

which results in Fig. 9.1. This variogram map provides a clear picture of how semivariances change in every compass direction. This allows one to more easily find the appropriate principal axis for defining the anisotropic variogram model. In this case, elevations are more spatially '*continuous*' in the direction NW–SE.

```
> Z.vgm

    model       psill    range ang1 anis1
  1   Nug    64.34422   0.0000    0   1.0
  2   Gau 11309.80343 963.0828  135   0.6
```

The resulting variogram shows that the feature of interest varies smoothly in the area of interest (Fig. 9.1), which is typical for elevation data. Nugget variation is insignificant, but $\neq 0$. This information can help us determine the amount of filtering needed to reduce man-made objects and artifacts in the LiDAR DEM.

[6] http://spatialreference.org/ref/epsg/26911/

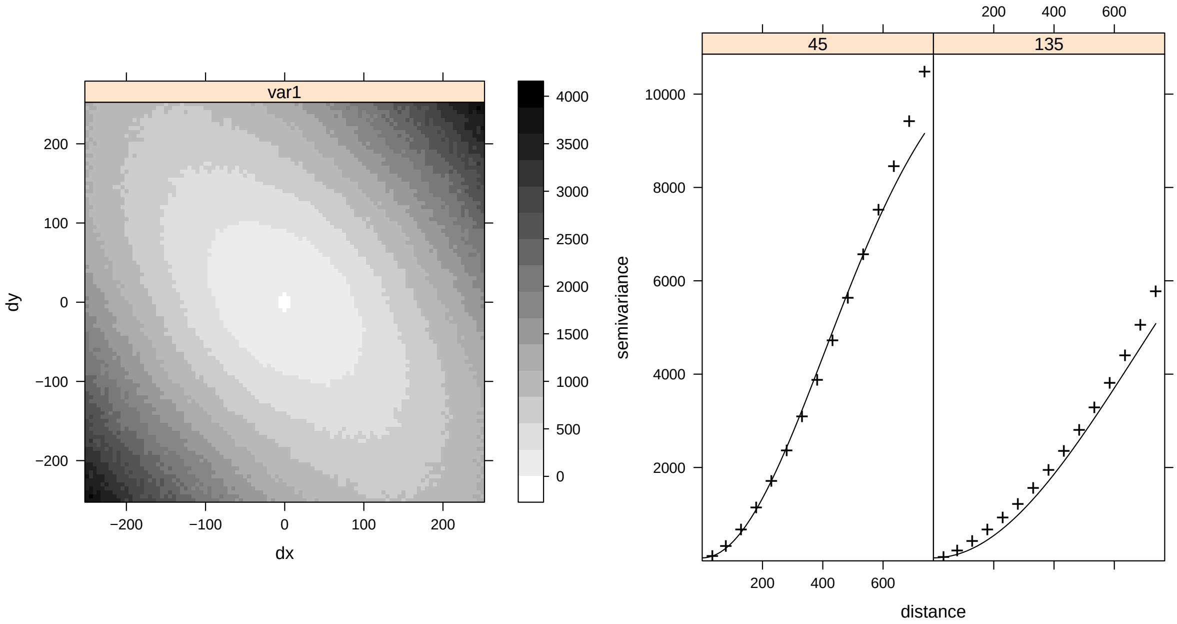


Fig. 9.1: Variogram map (left) and fitted anisotropic variogram model (right) for LiDAR-based elevations.

9.3.2 DEM filtering

Although the original LiDAR product is suppose to contain only ground reflection measurements, we can easily notice that there are still many artificial spikes and isolated pixels, with much higher elevation values than the neighboring pixels. For example, we can quickly generate a DEM in SAGA by converting the point map to a raster map:

```
> rsaga.geoprocessor(lib="grid_gridding", module=0,
+     param=list(GRID="DEM5LIDAR.sgrd", INPUT="lidar.shp", FIELD=0, LINE_TYPE=0,
+     USER_CELL_SIZE=pixelsize, USER_X_EXTENT_MIN=grids5m@bbox[1,1]+pixelsize/2,
+     USER_X_EXTENT_MAX=grids5m@bbox[1,2]-pixelsize/2,
+     USER_Y_EXTENT_MIN=grids5m@bbox[2,1]+pixelsize/2,
+     USER_Y_EXTENT_MAX=grids5m@bbox[2,2]-pixelsize/2))
```

which will contain many missing pixels (Fig. 9.2a). In addition, this DEM will show many small pixels with 10–20 m higher elevations from the neighbors. Spikes, roads and similar artifacts are not really connected with the geomorphology and need to be filtered before we can use the DEM for geomorphological mapping. Spikes[7] can be detected using, for example, difference from the mean value, given a search radius (see '*residual analysis*' in SAGA):

```
> rsaga.geoprocessor(lib="geostatistics_grid", 0,
+       param=list(INPUT="DEM5LIDAR.sgrd", MEAN="tmp.sgrd", STDDEV="tmp.sgrd",
+       RANGE="tmp.sgrd", DEVMEAN="tmp.sgrd", PERCENTILE="tmp.sgrd", RADIUS=5,
+       DIFF="dif_lidar.sgrd"))
# read back into R and mask out all areas:
> rsaga.sgrd.to.esri(in.sgrd=c("dif_lidar.sgrd", "DEM5LIDAR.sgrd"),
+       out.grids=c("dif_lidar.asc", "DEM5LIDAR.asc"), out.path=getwd(), prec=1)
> grids5m$DEM5LIDAR <- readGDAL("DEM5LIDAR.asc")$band1
> grids5m$dif <- readGDAL("dif_lidar.asc")$band1
> lim.dif <- quantile(grids5m$dif, c(0.025,0.975), na.rm=TRUE)
> lim.dif
```

[7] These individual pixels are most probably dense patches of forest, which are very difficult for LiDAR to penetrate.

```
   2.5% 97.5%
   -3.9   3.4

> grids5m$DEM5LIDARf <- ifelse(grids5m$dif<=lim.dif[[1]]|grids5m$dif>=lim.dif[[2]],
+       NA, grids5m$DEM5LIDAR)
> summary(grids5m$DEM5LIDARf)[7]/length(grids5m@data[[1]])
# 15% pixels have been masked out
```

which will remove about 15% of '*suspicious*' pixels. The remaining missing pixels can be filtered/re-interpolated[8] 1
from the neighboring pixels (see '*close gaps*' method in SAGA; the resulting map is shown in Fig. 9.2b): 2

```
rsaga.geoprocessor(lib="grid_tools", module=7, param=list(INPUT="DEM5LIDARf.sgrd",
+      RESULT="DEM5LIDARf.sgrd")) # we write to the same file!
```

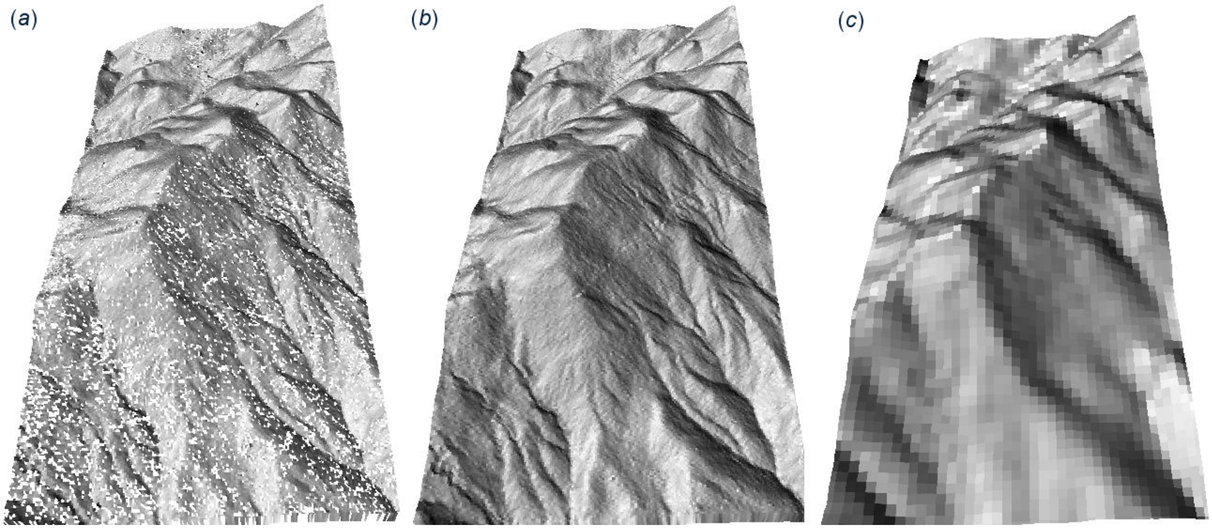


Fig. 9.2: Initial 5 m DEM (a) generated directly from the LiDAR points, and after filtering (b). In comparison with the 25 m DEM (c) derived from the contour lines. Seen from the western side.

9.3.3 DEM generation from contour data 3

We can also generate DEM surfaces from digitized contour lines (`contours.shp`) using a spline interpolation, 4
which is often recommended as the most suited DEM gridding technique for contour data (Conrad, 2007; 5
Neteler and Mitasova, 2008). In SAGA: 6

```
> rsaga.geoprocessor(lib="grid_spline", module=1, param=list(GRID="DEM25TPS.sgrd",
+      SHAPES="contours.shp", TARGET=0, SELECT=1, MAXPOINTS=10,
+      USER_CELL_SIZE=25, USER_FIT_EXTENT=T))
```

This looks for the nearest 10 points in a local search radius and fits the Thin Plate Spline[9] over a 25 grid. 7
This initial DEM can be *hydrologically* adjusted using the deepen drainage route: 8

```
> rsaga.geoprocessor(lib="ta_preprocessor", module=1,
+       param=list(DEM="DEM25TPS.sgrd", DEM_PREPROC="DEM25TPSf.sgrd", METHOD=0))
```

The resulting DEM surface can be seen in Fig. 9.2(c). We can compare the LiDAR-based and topo-map 9
based DEMs and estimate the accuracy[10] of the DEM derived from the contour lines. First, we need to aggregate 10
the 5 m resolution `DEM5LIDAR` to 25 m resolution: 11

[8] This can then be considered a void filling type of DEM filtering (Hengl and Reuter, 2008, p.104–106).

[9] SAGA implements the algorithm of Donato and Belongie (2003).

[10] Assuming that the LiDAR DEM is the '*true*' DEM.

```
# create empty grid:
> rsaga.geoprocessor(lib="grid_tools", module=23,
+     param=list(GRID="DEM25LIDAR.sgrd", M_EXTENT=0,
+     XMIN=grids5m@bbox[1,1]+pixelsize/2, YMIN=grids5m@bbox[2,1]+pixelsize/2,
+     NX=grids25m@grid@cells.dim[1], NY=grids25m@grid@cells.dim[2], CELLSIZE=25))
# resample to 25 m:
> rsaga.geoprocessor(lib="grid_tools", module=0,
+     param=list(INPUT="DEM5LIDARf.sgrd", GRID="DEM25LIDAR.sgrd",
+     GRID_GRID="DEM25LIDAR.sgrd", METHOD=2, KEEP_TYPE=FALSE, SCALE_UP_METHOD=5))
```

The difference between the two DEMs is then:

```
> rsaga.sgrd.to.esri(in.sgrd=c("DEM25LIDAR.sgrd", "DEM25TPS.sgrd"),
+     out.grids=c("DEM25LIDAR.asc", "DEM25TPS.asc"), out.path=getwd(), prec=1)
> grids25m$DEM25LIDAR <- readGDAL("DEM25LIDAR.asc")$band1
> grids25m$DEM25TPS <- readGDAL("DEM25TPS.asc")$band1
> sqrt(sum((grids25m$DEM25LIDAR-grids25m$DEM25TPS)^2)/length(grids25m$DEM25LIDAR))

 [1] 5.398652
```

which means that the average error of the DEM derived using contour lines (topo map) is about 5 m, which is well within the accuracy standards for this scale.

9.4 Extraction of Land Surface Parameters

We proceed with the extraction of LSPs that will be used to explain the distribution of soil mapping units. SAGA can derive over 100 LSPs given an input DEM. There is no need of course to use all of them; instead we need to try to list the LSPs that are relevant to the mapping objectives, study area characteristics and scale of application. For example, because the area is of high relief we can derive some representative DEM parameters that can explain hydrological, climatic and morphological properties of a terrain. For example: (1) SAGA Topographic Wetness Index (TWI), (2) Valley depth (VDEPTH), (3) Solar Insolation (INSOLAT), and (4) Convergence index (CONVI):

```
# Topographic Wetness Index:
> rsaga.geoprocessor(lib="ta_hydrology", module=15,
+    param=list(DEM="DEM5LIDARf.sgrd", C="catharea.sgrd", GN="catchslope.sgrd",
+    CS="modcatharea.sgrd", SB="TWI.sgrd", T=10))
# valley depth:
> rsaga.geoprocessor(lib="ta_morphometry", module=14,
+    param=list(DEM="DEM5LIDARf.sgrd", HO="tmp.sgrd", HU="VDEPTH.sgrd",
+    NH="tmp.sgrd", SH="tmp.sgrd", MS="tmp.sgrd", W=12, T=120, E=4))
# incoming solar radiation:
> rsaga.geoprocessor(lib="ta_lighting", module=2,
+    param=list(ELEVATION="DEM5LIDARf.sgrd", INSOLAT="INSOLAT.sgrd",
+    DURATION="durat.sgrd", LATITUDE=clat, HOUR_STEP=2, TIMESPAN=2, DAY_STEP=5))
# convergence index:
> rsaga.geoprocessor(lib="ta_morphometry", module=2,
+    param=list(ELEVATION="DEM5LIDARf.sgrd", RESULT="CONVI.sgrd", RADIUS=3,
+    METHOD=0, SLOPE=TRUE))
```

Note that, because the features of interest (soil mapping units) are geographically continuous and smooth, we should use a wide search radius to derive the LSPs. In this case we use arbitrary parameters to derive specific LSPs — these are not as easy to determine objectively[11].

Now that we have prepared a list of DEM parameters that can be used to describe geomorphology of the terrain, we can proceed with the extraction of land-surface objects. Before we can proceed, we need to read the maps into R:

[11] Note also that the resulting LSPs can also differ largely for different combination of parameters

```
> LSP.list <- c("TWI.asc", "VDEPTH.asc", "INSOLAT.asc", "CONVI.asc")
> rsaga.sgrd.to.esri(in.sgrds=set.file.extension(LSP.list, ".sgrd"),
+      out.grids=LSP.list, prec=1, out.path=getwd())
> for(i in 1:length(LSP.list)){
>   grids5m@data[strsplit(LSP.list[i], ".asc")[[1]]] <- readGDAL(LSP.list[i])$band1
> }

  TWI.asc has GDAL driver AAIGrid
  and has 200 rows and 400 columns
  ...
  CONVI.asc has GDAL driver AAIGrid
  and has 200 rows and 400 columns
```

9.5 Unsupervised extraction of landforms

9.5.1 Fuzzy *k*-means clustering

Geomorphological classes can be optimally extracted using, for example, the **fuzzy *k*-means clustering** approach as implemented in that stats package (Venables and Ripley, 2002). This will optimally assign each individual pixel to an abstract class; the class centers will be selected in such way that the within groups sum of squares is minimized. In statistical terms, this is the cluster analysis approach to extraction of features.

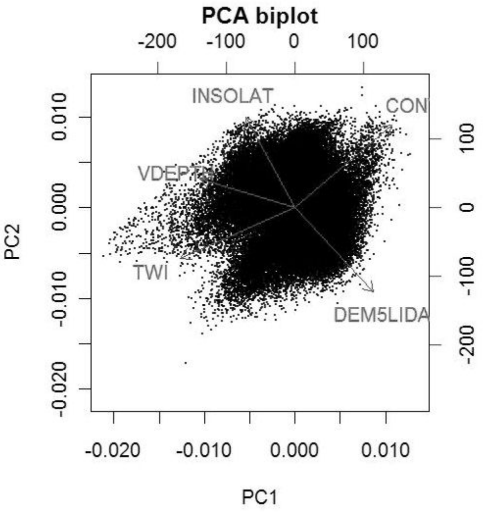


Fig. 9.3: The PCA biplot showing first two components derived using five LSPs.

We can start by converting the LSPs to independent components by using Principal Component analysis (Fig. 9.3):

```
> pc.dem <- prcomp( ~ DEM5LIDARf+TWI+VDEPTH+
+      INSOLAT+CONVI, scale=TRUE, grids5m@data)
> biplot(pc.dem, arrow.len=0.1,
+      xlabs=rep(".", length(pc.dem$x[,1])),
+      main="PCA biplot")
```

which shows that the LSPs are relatively independent. To be statistically correct, we will proceed with clustering the Principal Components instead of using the original predictors. Next we can try to obtain the optimal number of classes for fuzzy *k*-means clustering by using (Venables and Ripley, 2002)[12]:

```
> demdata <- as.data.frame(pc.dem$x)
> wss <- (nrow(demdata)-1)*sum(apply(demdata,2,var))
> for (i in 2:20) {wss[i] <- sum(kmeans(demdata, centers=i)$withinss)}

  Warning messages:
  1: did not converge in 10 iterations
```

which unfortunately did not converge[13]. For practical reasons, we will assume that 12 classes are sufficient:

```
> kmeans.dem <- kmeans(demdata, 12)
> grids5m$kmeans.dem <- kmeans.dem$cluster
> grids5m$landform <- as.factor(kmeans.dem$cluster)
> summary(grids5m$landform)
```

[12] An alternative to *k*-means clustering is to use the Partitioning Around Medoids (PAM) method, which is generally more robust to '*messy*' data, and will always return the same clusters.

[13] Which also means that increasing the number of classes above 20 will still result in smaller within groups sum of squares.

```
   1     2     3     4     5     6
2996  3718  6785  7014  4578  7895
   7     8     9    10    11    12
7367 13232  2795  6032  7924  9664
```

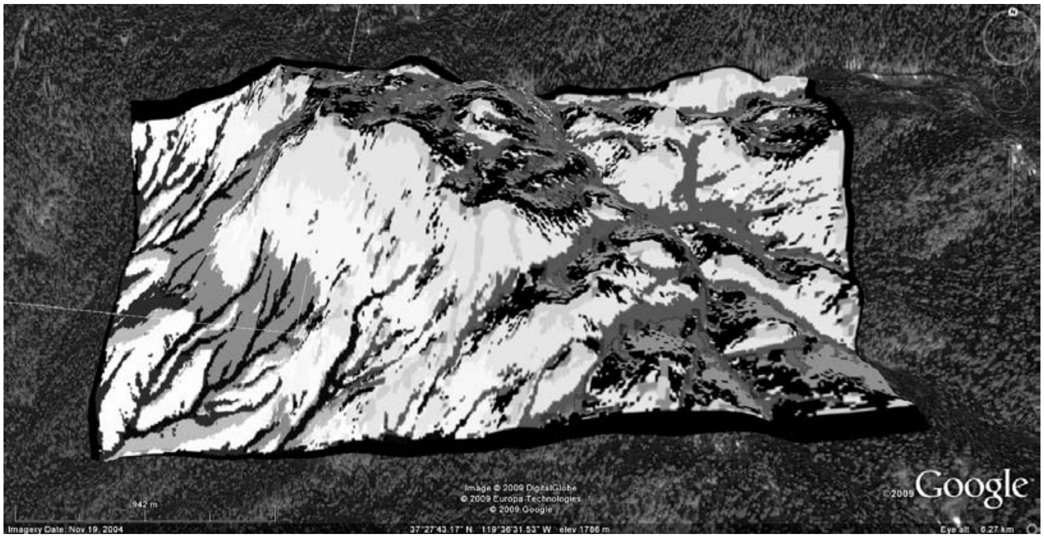

942 m
Image © 2009 DigitalGlobe
© 2009 Europa Technologies
© 2009 Google
©2009 Google
Imagery Date: Nov 19, 2004
37°27'43.17" N 119°36'31.53" W elev 1786 m
Eye alt 6.27 km

Fig. 9.4: Results of unsupervised classification (12 classes) visualized in Google Earth.

The map of predicted classes can be seen in Fig. 9.4. The size of polygons is well distributed and the polygons are spatially continuous. The remaining issue is what do these classes really mean? Are these really geomorphological units and could different classes be combined? Note also that there is a number of object segmentation algorithms that could be combined with extraction of (homogenous) landform units (Seijmonsbergen et al., 2010).

9.5.2 Fitting variograms for different landform classes

Now that we have extracted landform classes, we can see if there are differences between the variograms for different landform units (Lloyd and Atkinson, 1998). To be efficient, we can automate variogram fitting by running a loop. The best way to achieve this is to make an empty data frame and then fill it in with the results of fitting:

```
> lidar.sample.ov <- overlay(grids5m["landform"], lidar.sample)
> lidar.sample.ov$Z <- lidar.sample$Z
# number of classes:
> landform.no <- length(levels(lidar.sample.ov$landform))
# empty dataframes:
> landform.vgm <- as.list(rep(NA, landform.no))
> landform.par <- data.frame(landform=as.factor(levels(lidar.sample.ov$landform)),
+      Nug=rep(NA, landform.no), Sill=rep(NA, landform.no),
+      range=rep(NA, landform.no))
# fit the variograms:
> for(i in 1:length(levels(lidar.sample.ov$landform))) {
>   tmp <- subset(lidar.sample.ov,
+      lidar.sample.ov$landform==levels(lidar.sample.ov$landform)[i])
>   landform.vgm[[i]] <- fit.variogram(variogram(Z ~ 1, tmp, cutoff=50*pixelsize),
+      vgm(psill=var(tmp$Z), "Gau", sqrt(areaSpatialGrid(grids25m))/4, nugget=0))
>   landform.par$Nug[i] <- round(landform.vgm[[i]]$psill[1], 1)
>   landform.par$Sill[i] <- round(landform.vgm[[i]]$psill[2], 1)
>   landform.par$range[i] <- round(landform.vgm[[i]]$range[2], 1)
> }
```

and we can print the results of variogram fitting in a table:

```
> landform.par
```

```
   landform Nug   Sill range
1         1 4.0 5932.4 412.6
2         2 0.5  871.1 195.1
3         3 0.1 6713.6 708.2
4         4 8.6  678.7 102.0
5         5 0.0 6867.0 556.8
6         6 7.6 2913.6 245.5
7         7 3.3 1874.2 270.9
8         8 5.8  945.6 174.1
9         9 6.4  845.6 156.0
10       10 5.0 3175.5 281.2
11       11 5.1  961.6 149.5
12       12 7.4 1149.9 194.4
```

which shows that there are distinct differences in variograms between different landform classes (see also Fig. 9.4). This can be interpreted as follows: the variograms differ mainly because there are differences in the surface roughness between various terrains, which is also due to different tree coverage.

Consider also that there are possibly still many artificial spikes/trees that have not been filtered. Also, many landforms are '*patchy*' i.e. represented by isolated pixels, which might lead to large differences in the way the variograms are fitted. It would be interesting to try to fit local variograms[14], i.e. variograms for each grid cell and then see if there are real discrete jumps in the variogram parameters.

9.6 Spatial prediction of soil mapping units

9.6.1 Multinomial logistic regression

Next, we will use the extracted LSPs to try to improve the spatial detail of an existing *traditional*[15] soil map. A suitable technique for this type of analysis is the multinomial logistic regression algorithm, as implemented in the multinom method of the nnet package (Venables and Ripley, 2002, p.203). This method iteratively fits logistic models for a number of classes given a set of training pixels. The output predictions can then be evaluated versus the complete geomorphological map to see how well the two maps match and where the most problematic areas are. We will follow the iterative computational framework shown in Fig. 9.5. In principle, the best results can be obtained if the selection of LSPs and parameters used to derive LSPs are iteratively adjusted until maximum mapping accuracy is achieved.

9.6.2 Selection of training pixels

Because the objective here is to refine the existing soil map, we use a selection of pixels from the map to fit the model. A simple approach would be to randomly sample points from the existing maps and then use them to train the model, but this has a disadvantage of (wrongly) assuming that the map is absolutely the same quality in all parts of the area. Instead, we can place the training pixels along the medial axes for polygons of interest. The medials axes can be derived in SAGA, but we need to convert the gridded map first to a polygon map, then extract lines, and then derive the buffer distance map:

```
# convert the raster map to polygon map:
> rsaga.esri.to.sgrd(in.grids="soilmu.asc", out.sgrd="soilmu.sgrd",
+    in.path=getwd())
> rsaga.geoprocessor(lib="shapes_grid", module=6, param=list(GRID="soilmu.sgrd",
+    SHAPES="soilmu.shp", CLASS_ALL=1))
# convert the polygon to line map:
> rsaga.geoprocessor(lib="shapes_lines", module=0,
+    param=list(POLYGONS="soilmu.shp", LINES="soilmu_l.shp"))
```

[14] Local variograms for altitude data can be derived in the Digeman software provided by Bishop et al. (2006).

[15] Mapping units drawn manually, by doing photo-interpretation or following some similar procedure.

```
# derive the buffer map using the shapefile:
> rsaga.geoprocessor(lib="grid_gridding", module=0,
+    param=list(GRID="soilmu_r.sgrd", INPUT="soilmu_l.shp", FIELD=0, LINE_TYPE=0,
+    TARGET_TYPE=0, USER_CELL_SIZE=pixelsize,
+    USER_X_EXTENT_MIN=grids5m@bbox[1,1]+pixelsize/2,
+    USER_X_EXTENT_MAX=grids5m@bbox[1,2]-pixelsize/2,
+    USER_Y_EXTENT_MIN=grids5m@bbox[2,1]+pixelsize/2,
+    USER_Y_EXTENT_MAX=grids5m@bbox[2,2]-pixelsize/2))
# buffer distance:
> rsaga.geoprocessor(lib="grid_tools", module=10,
+    param=list(SOURCE="soilmu_r.sgrd", DISTANCE="soilmu_dist.sgrd",
+    ALLOC="tmp.sgrd", BUFFER="tmp.sgrd", DIST=sqrt(areaSpatialGrid(grids25m))/3,
+    IVAL=pixelsize))
# surface specific points (medial axes!):
> rsaga.geoprocessor(lib="ta_morphometry", module=3,
+    param=list(ELEVATION="soilmu_dist.sgrd", RESULT="soilmu_medial.sgrd",
+    METHOD=1))
```

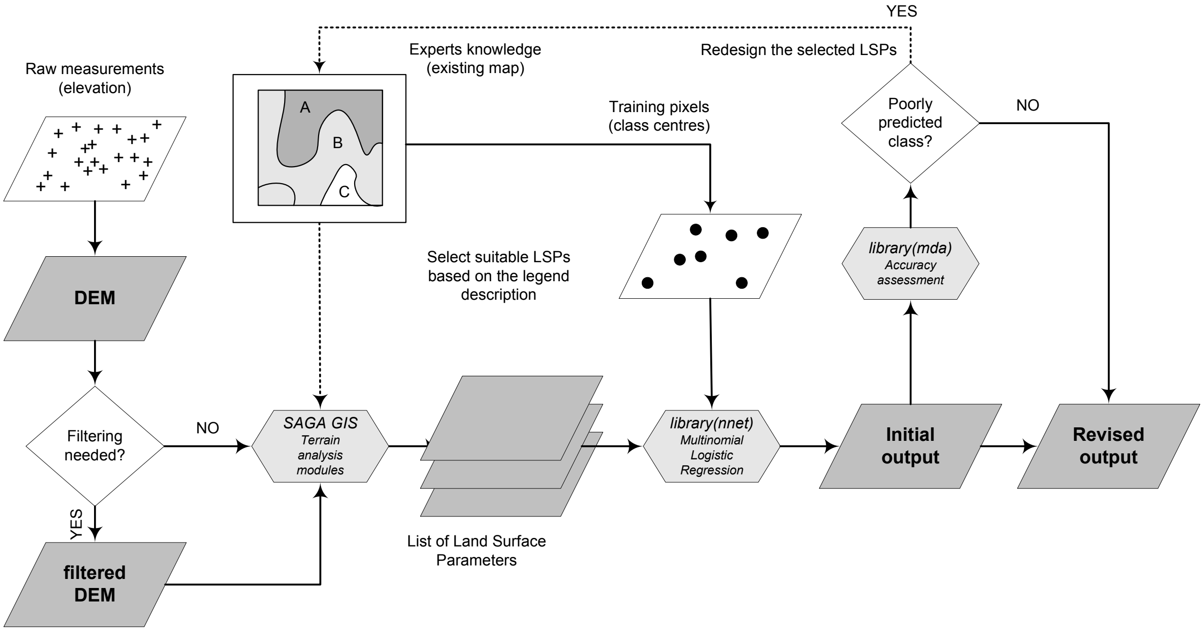

YES
Raw measurements (elevation)
Experts knowledge (existing map)
Redesign the selected LSPs
Poorly predicted class?
NO
A
B
C
Training pixels (class centres)
DEM
Select suitable LSPs based on the legend description
library(mda)
Accuracy assessment
Filtering needed?
NO
SAGA GIS
Terrain analysis modules
library(nnet)
Multinomial Logistic Regression
Initial output
Revised output
YES
filtered DEM
List of Land Surface Parameters

Fig. 9.5: Data analysis scheme and connected R packages: supervised extraction of geomorphological classes using the existing geomorphological map — a hybrid expert/statistical based approach.

The map showing medial axes can then be used as a weight map to randomize the sampling (see further Fig. 9.6a). The sampling design can be generated using the rpoint method[16] of the spatstat package:

```
# read into R:
> rsaga.sgrd.to.esri(in.sgrds="soilmu_medial.sgrd",
+    out.grids="soilmu_medial.asc", prec=0, out.path=getwd())
> grids5m$soilmu_medial <- readGDAL("soilmu_medial.asc")$band1
# generate the training pixels:
> grids5m$weight <- abs(ifelse(grids5m$soilmu_medial>=0, 0, grids5m$soilmu_medial))
> dens.weight <- as.im(as.image.SpatialGridDataFrame(grids5m["weight"]))
# image(dens.weight)
```

[16] This will generate a point pattern given a prior probability i.e. a mask map.

```
> training.pix <- rpoint(length(grids5m$weight)/10, f=dens.weight)
# plot(training.pix)
> training.pix <- data.frame(x=training.pix$x, y=training.pix$y,
+    no=1:length(training.pix$x))
> coordinates(training.pix) <- ~ x+y
```

This reflects the idea of sampling the class centers, at least in the geographical sense. The advantage of using the medial axes is that also relatively small polygons will be represented in the training pixels set (or in other words — large polygons will be under-represented, which is beneficial for the regression modeling). The most important is that the algorithm will minimize selection of transitional pixels that might well be in either of the two neighboring classes.

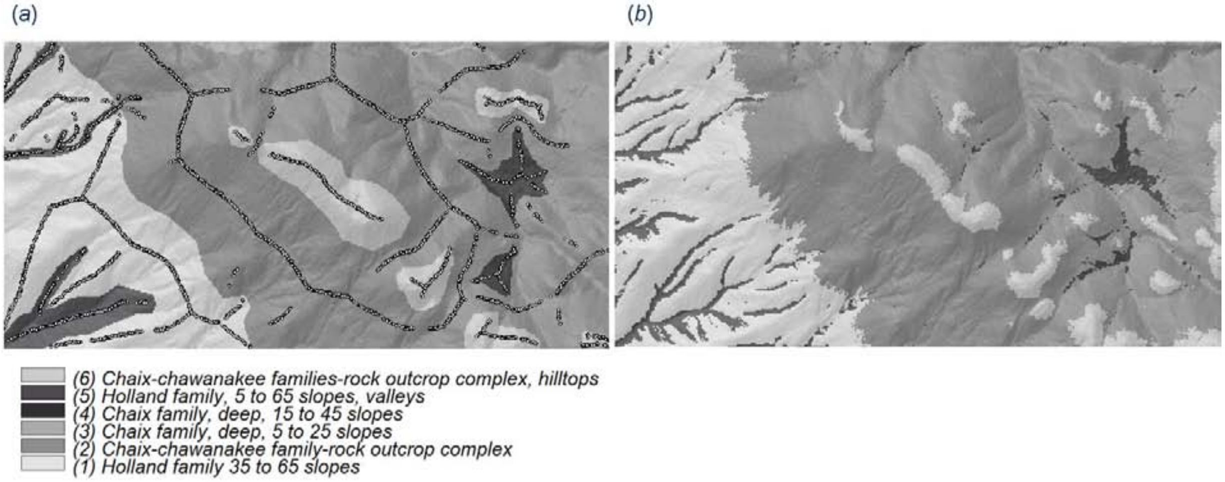

(a)
(b)
(6) Chaix-chawanakee families-rock outcrop complex, hilltops
(5) Holland family, 5 to 65 slopes, valleys
(4) Chaix family, deep, 15 to 45 slopes
(3) Chaix family, deep, 5 to 25 slopes
(2) Chaix-chawanakee family-rock outcrop complex
(1) Holland family 35 to 65 slopes

Fig. 9.6: Results of predicting soil mapping units using DEM-derived LSPs: (a) original soil mapping units and training pixels among medial axes, (b) soil mapping units predicted using multinomial logistic regression.

Once we have allocated the training pixels, we can fit a logistic regression model using the nnet package, and then predict the mapping units for the whole area of interest:

```
# overlay the training points and grids:
> training.pix.ov <- overlay(grids5m, training.pix)
> library(nnet)
> mlr.soilmu <- multinom(soilmu.c ~ DEM5LIDARf+TWI+VDEPTH+INSOLAT+CONVI, training.pix.ov)

 # weights:  42 (30 variable)
 initial  value 14334.075754
 iter  10 value 8914.610698
 iter  20 value 8092.253630
 ...
 iter 100 value 3030.721321
 final  value 3030.721321
 stopped after 100 iterations


# make predictions:
> grids5m$soilmu.mlr <- predict(mlr.soilmu, newdata=grids5m)
```

Finally, we can compare the map generated using multinomial logistic regression versus the existing map (Fig. 9.6). To compare the overall fit between the two maps we can use the mda package:

```
> library(mda) # kappa statistics
> sel <- !is.na(grids5m$soilmu.c)
> Kappa(confusion(grids5m$soilmu.c[sel], grids5m$soilmu.mlr[sel]))
```

```
                value         ASE
  Unweighted 0.6740377 0.002504416
  Weighted   0.5115962 0.003207276
```

which shows that the matching between the two maps is 51–67%. A relatively low kappa is typical for soil and/or geomorphological mapping applications[17]. We have also ignored that these map units represent suites of soil types, stratified by prominence of rock outcroppings and by slope classes and NOT uniform soil bodies. Nevertheless, the advantage of using a statistical procedure is that it reflects the experts knowledge more objectively. The results will typically show more spatial detail (small patches) than the hand drawn maps. Note also that the multinom method implemented in the nnet package is a fairly robust technique in the sense that it generates few artifacts. Further refinement of existing statistical models (regression-trees and/or machine-learning algorithms) could also improve the mapping of landform categories.

9.7 Extraction of memberships

We can also extend the analysis and extract memberships for the given soil mapping units, following the fuzzy *k*-means algorithm described in Hengl et al. (2004c). For this purpose we can use the same training pixels, but then associate the pixels to classes just by standardizing the distances in feature space. This is a more trivial approach than the multinomial logistic regression approach used in the previous exercise.

The advantage of using membership, on the other hand, is that one can observe how crisp certain classes are, and where the confusion of classes is the highest. This way, the analyst has an opportunity to focus on mapping a single geomorphological unit, adjust training pixels where needed and increase the quality of the final maps (Fisher et al., 2005). A supervised fuzzy *k*-means algorithm is not implemented in any R package (yet), so we will derive memberships step-by-step.

First, we need to estimate the class centers (mean and standard deviation) for each class of interest:

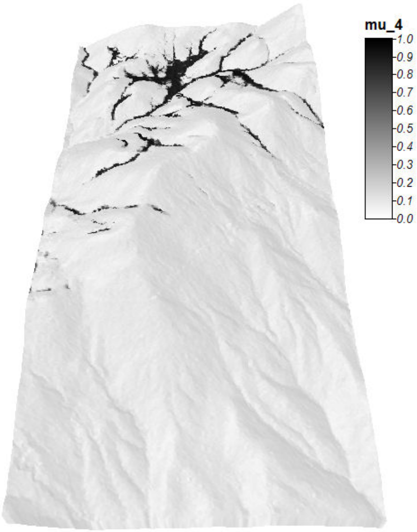


Fig. 9.7: Membership values for soil mapping unit: Chaix family, deep, 15 to 45% slopes.

```
# mask-out classes with <5 points:
mask.c <- as.integer(attr(summary(training.pix.ov$soilmu.c
+      [summary(training.pix.ov$soilmu.c)<5]), "names"))
# fuzzy exponent:
> fuzzy.e <- 1.2
# extract the class centroids:
> class.c <- aggregate(training.pix.ov@data[c("DEM5LIDARf", "TWI", "VDEPTH",
+      "INSOLAT", "CONVI")], by=list(training.pix.ov$soilmu.c), FUN="mean")
> class.sd <- aggregate(training.pix.ov@data[c("DEM5LIDARf", "TWI", "VDEPTH",
+      "INSOLAT", "CONVI")], by=list(training.pix.ov$soilmu.c), FUN="sd")
```

which allows us to derive diagonal/standardized distances between the class centers and all individual pixels:

```
# derive distances in feature space:
> distmaps <- as.list(levels(grids5m$soilmu.c)[mask.c])
> tmp <- rep(NA, length(grids5m@data[[1]]))
> for(c in (1:length(levels(grids5m$soilmu.c)))[mask.c]){
>   distmaps[[c]] <- data.frame(DEM5LIDARf=tmp, TWI=tmp, VDEPTH=tmp,
+        INSOLAT=tmp, CONVI=tmp)
>   for(j in list("DEM5LIDARf", "TWI", "VDEPTH", "INSOLAT", "CONVI")){
>     distmaps[[c]][j] <- ((grids5m@data[j]-class.c[c,j])/class.sd[c,j])^2
```

[17] An in-depth discussion can be followed in Kempen et al. (2009).

```
>   }
> }
# sum up distances per class:
> distsum <- data.frame(tmp)
> for(c in (1:length(levels(grids5m$soilmu.c)))[mask.c]){
>   distsum[paste(c)] <- sqrt(rowSums(distmaps[[c]], na.rm=T, dims=1))
> }
> str(distsum)

  'data.frame':   80000 obs. of  6 variables:
   $ 1: num  1.53 1.56 1.75 2.38 3.32 ...
   $ 2: num  4.4 4.41 4.53 4.88 5.48 ...
   $ 3: num  37.5 37.5 37.5 37.5 37.6 ...
   $ 4: num  10 10 10.1 10.2 10.5 ...
   $ 5: num  2.64 2.54 3.18 4.16 5.35 ...
   $ 6: num  8.23 8.32 8.28 8.06 7.58 ...


# total sum of distances for all pixels:
> totsum <- rowSums(distsum^(-2/(fuzzy.e-1)), na.rm=T, dims=1)
```

Once we have estimated the standardized distances, we can then derive memberships using formula (Sokal and Sneath, 1963):

$$\mu_c(i) = \frac{\left[d_c^2(i)\right]^{-\frac{1}{(q-1)}}}{\sum\limits_{c=1}^{k}\left[d_c^2(i)\right]^{-\frac{1}{(q-1)}}} \qquad c = 1, 2, ...k \qquad i = 1, 2, ...n \tag{9.7.1}$$

$$\mu_c(i) \in [0, 1] \tag{9.7.2}$$

where $\mu_c(i)$ is a fuzzy membership value of the ith object in the cth cluster, d is the similarity (*diagonal*) distance, k is the number of clusters and q is the fuzzy exponent determining the amount of fuzziness. Or in R syntax:

```
> for(c in (1:length(levels(grids5m$soilmu.c)))[mask.c]){
>   grids5m@data[paste("mu_", c, sep="")] <-
+         (distsum[paste(c)]^(-2/(fuzzy.e-1))/totsum)[,1]
> }
```

The resulting map of memberships for class "*Chaix family, deep, 15 to 45% slopes*" is shown in Fig. 9.7. Compare also with Fig. 9.6.

Self-study exercises:

(1.) How much is elevation correlated with the TWI map? (derive correlation coefficient between the two maps)

(2.) Which soil mapping unit in Fig. 9.6 is the most correlated to the original map? (HINT: convert to indicators and then correlate the maps.)

(3.) Derive the variogram for the filtered LIDAR DEM and compare it to the variogram for elevations derived using the (LiDAR) point measurements. (HINT: compare nugget, sill, range parameter and anisotropy parameters.)

(4.) Extract landform classes using the `pam` algorithm as implemented in the `cluster` package and compare it with the results of `kmeans`. Which polygons/classes are the same in >50% of the area? (HINT: overlay the two maps and derive summary statistics.)

(5.) Extract the landform classes using unsupervised classification and the coarse SRTM DEM and identify if there are significant differences between the LiDAR based and coarse DEM. (HINT: compare the average area of landform unit; derive a correlation coefficient between the two maps.)

(6.) Try to add five more LSPs and then re-run multinomial logistic regression. Did the fitting improve and how much? (HINT: Compare the resulting AIC for fitted models.)

(7.) Extract membership maps (see section 9.7) for all classes and derive the confusion index. Where is the confusion index the highest? Is it correlated with any input LSP?

Further reading:

★ Brenning, A., 2008. Statistical Geocomputing combining R and SAGA: The Example of Landslide susceptibility Analysis with generalized additive Models. In: J. Böhner, T. Blaschke & L. Montanarella (eds.), **SAGA — Seconds Out** (Hamburger Beiträge zur Physischen Geographie und Landschaftsökologie, 19), 23–32.

★ Burrough, P. A., van Gaans, P. F. M., MacMillan, R. A., 2000. High-resolution landform classification using fuzzy k-means. Fuzzy Sets and Systems 113, 37–52.

★ Conrad, O., 2007. **SAGA — Entwurf, Funktionsumfang und Anwendung eines Systems für Automatisierte Geowissenschaftliche Analysen**. Ph.D. thesis, University of Göttingen, Göttingen.

★ Fisher, P. F., Wood, J., Cheng, T., 2005. Fuzziness and ambiguity in multi-scale analysis of landscape morphometry. In: Petry, F. E., Robinson, V. B., Cobb, M. A. (Eds.), Fuzzy Modeling with Spatial Information for Geographic Problems. Springer-Verlag, Berlin, pp. 209–232.

★ Smith, M. J., Paron, P. and Griffiths, J. (eds) 2010. **Geomorphological Mapping: a professional handbook of techniques and applications**. Developments in Earth Surface Processes, Elsevier, in preparation.

10

Stream networks (baranjahill)

10.1 Introduction

The purpose of this exercise is to generate a map of stream networks using **error propagation**[1] techniques (Heuvelink, 2002), i.e. to generate multiple DEMs using conditional simulations, then derive a stream network for each realization, and finally evaluate how the propagated uncertainty correlates to various topographic parameters. Upon completion of this exercise, you will be able to generate loops, run SAGA via the R command line, and import and export raster maps from R to SAGA GIS and Google Earth. The last section in this exercise focuses on GRASS GIS and how to run a similar type of analysis in this open source GIS.

The "Baranja hill" study area, located in eastern Croatia, has been mapped extensively over the years and several GIS layers are available at various scales (Hengl and Reuter, 2008). The study area corresponds approximately to the size of a single 1:20,000 aerial photo. Its main geomorphic features include hill summits and shoulders, eroded slopes of small valleys, valley bottoms, a large abandoned river channel, and river terraces. Elevation of the area ranges from 80 to 240 m with an average of 157.6 m and a standard deviation of 44.3 m. The complete "Baranja hill" data set is available from the geomorphometry website[2].

In principle, the only input for this exercise is a point map showing field-measured elevations (ESRI Shapefile). This map will be used to generate multiple realizations of a Digital Elevation Model, and then extract drainage networks, as implemented in the SAGA GIS package. This exercise is based on previous articles published in Computers and Geosciences (Hengl et al., 2008). A similar exercise can be found in Temme et al. (2008). A detailed description of the RSAGA package can be can be found in Brenning (2008).

10.2 Data download and import

First, open a new R session and change the working directory to where all the data sets are located. Download the R script (`baranjahill.R`) needed to complete this exercise, and open it in some script editor. Before you start any processing, you will need to load the following packages:

```
> library(maptools)
> library(gstat)
> library(geoR)
> library(rgdal)
> library(lattice)
> library(RSAGA)
```

Check `rsaga.env` to make sure that RSAGA can find your local installation of SAGA. As indicated in §3.1.2, SAGA is not an R package, but an external application and needs to be installed separately.

We also need to set-up the correct coordinate system for this study area, which is the Gauss-Krueger-based coordinate system zone 6, with a 7–parameter datum definition:

[1] A practical guide to error propagation is available via `http://spatial-accuracy.org/workshopSUP`.

[2] `http://geomorphometry.org/content/baranja-hill`

```
# set the correct coordinate system:
> gk_6 <- "+proj=tmerc +lat_0=0 +lon_0=18 +k=0.9999 +x_0=6500000 +y_0=0
+           +ellps=bessel +units=m
+           +towgs84=550.499,164.116,475.142,5.80967,2.07902,-11.62386,0.99999445824
```

Next, download the shapefile (elevations.shp) and extract it to the local folder[3]:

```
> download.file("http://spatial-analyst.net/book/system/files/elevations.zip",
+     destfile=paste(getwd(),"elevations.zip",sep="/"))

  trying URL 'http://spatial-analyst.net/book/system/files/elevations.zip'
  Content type 'application/x-zip-compressed' length 165060 bytes (161 Kb)
  opened URL
  downloaded 161 Kb
```

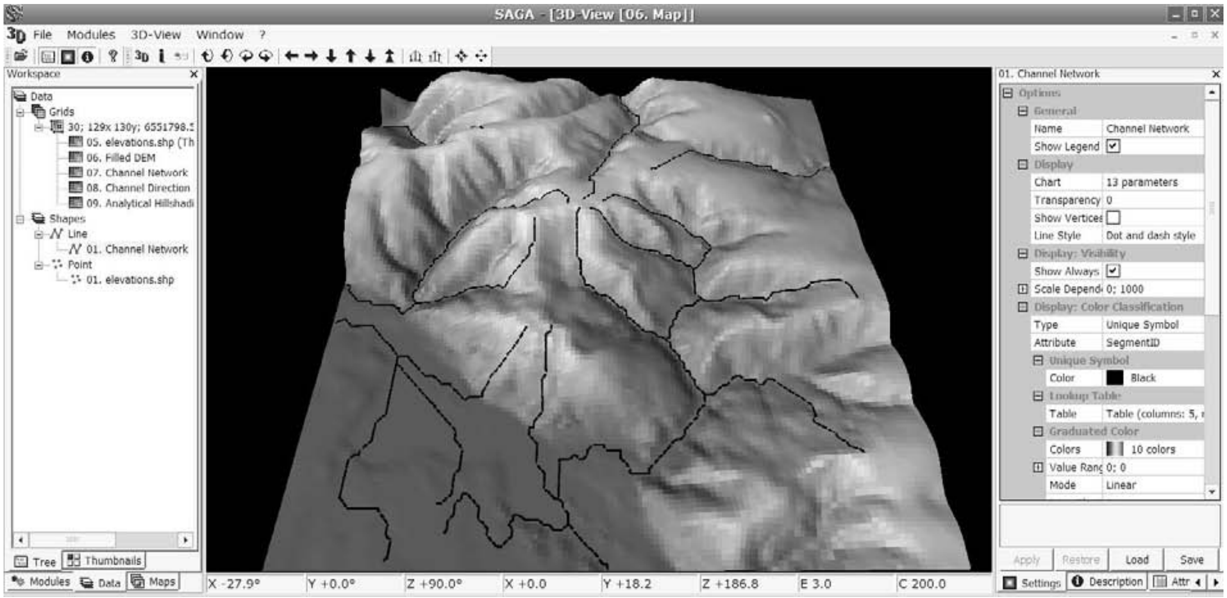
```
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("elevations", j, sep=""),
+        zipname="elevations.zip")
>   file.copy(fname, paste("./elevations", j, sep=""), overwrite=TRUE)
> }
> unlink("elevations.zip")
> list.files(getwd(), recursive=T, full=F)

  [1] "baranjahill.R"  "elevations.dbf"
  [3] "elevations.shp" "elevations.shx"
```

We can import the sampled elevations from elevations.shp to R using:

```
> elevations <- readShapePoints("elevations.shp", proj4string=CRS(gk_6))
> str(elevations)

  Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
    ..@ data       :'data.frame': 6367 obs. of  1 variable:
    .. ..$ VALUE: num [1:6367] 206 208 207 204 203 ...
    .. ..- attr(*, "data_types")= chr "N"
    ..@ coords.nrs : num(0)
    ..@ coords     : num [1:6367, 1:2] 6551880 6552027 6551949 6552134 6551846 ...
    .. ..- attr(*, "dimnames")=List of 2
    .. .. ..$ : chr [1:6367] "0" "1" "2" "3" ...
    .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
    ..@ bbox       : num [1:2, 1:2] 6551799 5070471 6555640 5074356
    .. ..- attr(*, "dimnames")=List of 2
    .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
    .. .. ..$ : chr [1:2] "min" "max"
    ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
    .. .. ..@ projargs: chr " +proj=tmerc +lat_0=0 +lon_0=18 +k=0.9999 +x_0=6500000
    +y_0=0 +ellps=bessel +units=m +towgs84=550.499,164.116,475.142,5.80967,2"|
    __truncated__
```

```
> names(elevations@data) <- "Z"
```

This shows that the data set consists of 6367 points of field measured heights. The heights can be used to generate a Digital Elevation Model (DEM), that can then be used to extract a stream network. For example, you can open the point layer in SAGA GIS, then use the module *Grid* ↦ *Gridding* ↦ *Spline interpolation* ↦ *Thin Plate Splines (local)* and generate a smooth DEM[4]. Then, you can preprocess the DEM to remove spurious sinks using the method of Planchon and Darboux (2001). Select *Terrain Analysis Preprocessing Fill sinks*, and then set the minimum slope parameter to 0.1. Now that we have prepared a DEM, we can derive stream networks using the *Channel Network* function which is available in SAGA under *Terrain Analysis* ↦ *Channels*. You can use e.g. 40 (pixels) as the minimum length of streams. This would produce a map (vector layer) as



⊟ Options

⊟ General

Name	Channel Network
Show Legend	☑

⊟ Display

Chart	13 parameters
Transparency	0
Show Vertices	☐
Line Style	Dot and dash style

⊟ Display: Visibility

Show Always	☑
⊞ Scale Depend	0; 1000

⊟ Display: Color Classification

Type	Unique Symbol
Attribute	SegmentID

⊟ Unique Symbol

Color	■ Black

⊟ Lookup Table

Table	Table (columns: 5, r

⊟ Graduated Color

Colors	10 colors
⊞ Value Rang	0; 0
Mode	Linear

Fig. 10.1: Stream network generated in SAGA GIS. Viewed from the West side.

shown in Fig. 10.1. Assuming that the DEM and the stream extraction model are absolutely accurate, i.e. that they perfectly fit the reality, this would then be the end product of the analysis.

However, in reality, we know that errors exist — they are inherent both in measurements of elevations and in the stream extraction algorithm — and that they possibly have a significant influence on the final product (stream network). But how important is the influence of errors, and how are the errors connected with the geomorphometric properties of terrain? This is exactly what we will try to answer in this exercise.

10.3 Geostatistical analysis of elevations

10.3.1 Variogram modelling

In the previous exercise we generated a smooth DEM by using spline interpolation and by setting some parameters '*by hand*'. We would now like to produce a surface model using a more objective approach. We can take a step back and do some preliminary analysis in geoR to estimate possible anisotropy and evaluate how smooth is the elevation surface. First, let us look at the distribution of values:

```
> range(elevations$Z)

  [1]  85.0 244.2
```

```
# sub-sample -- geoR cannot deal with large data sets!
> sel <- runif(length(elevations@data[[1]]))<0.2
> Z.geo <- as.geodata(elevations[sel,"Z"])
# histogram:
> plot(Z.geo, qt.col=grey(runif(4)))
```

which shows that the values of Z are approximately normally distributed, with some clustering around low values (Fig. 10.2, bottom right). You can notice from Fig. 10.1 that the clustered low values are elevations measured in the floodplain.

To get some idea about the smoothness of the target variable and possible anisotropy, we can plot the two standard variograms:

[3] This exercise starts with a single input data set, but then results in a long list of maps! Make sure you have enough space on your computer.

[4] You can set 500 m as the search radius and grid resolution of 30 m.

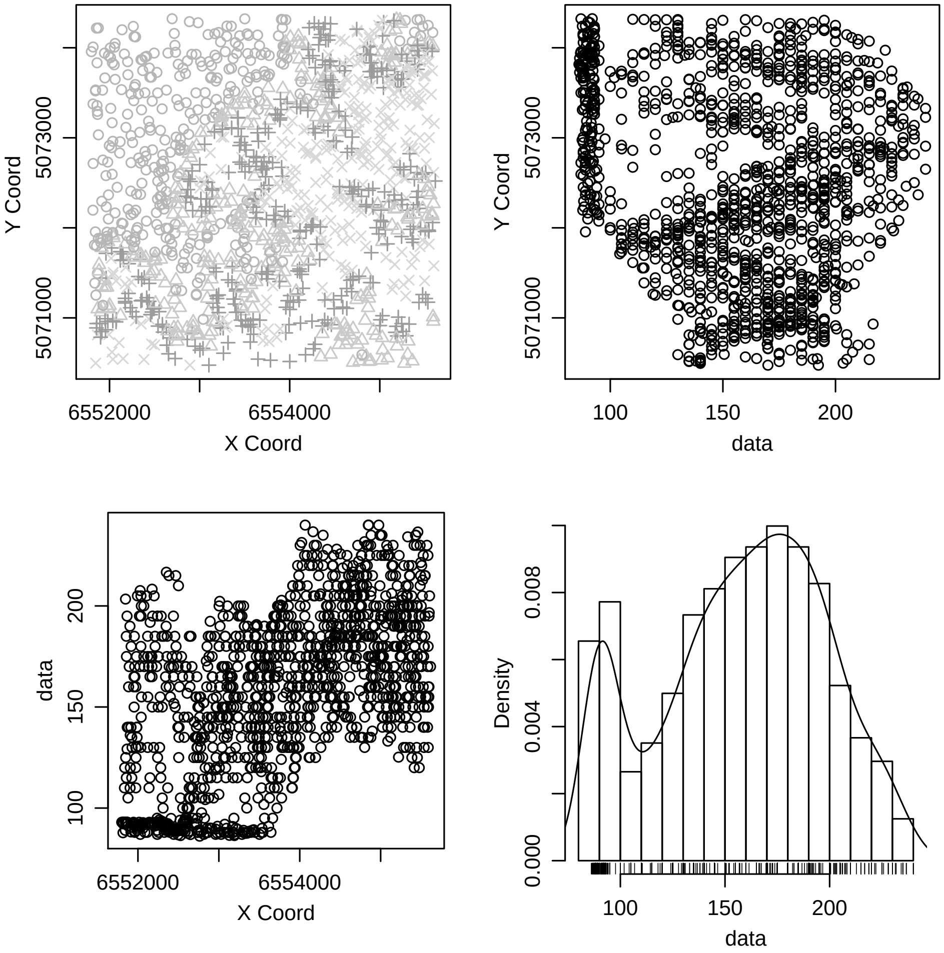


Fig. 10.2: Distribution of the target variable in geographical and feature space. A standard plot (plot.geodata) in geoR using a 20% sub-sample of the original data set.

```
> par(mfrow=c(1,2))
# anisotropy:
> plot(variog4(Z.geo, max.dist=1000, messages=FALSE), lwd=2)
# fit variogram using likfit:
> Z.svar <- variog(Z.geo, max.dist=1000, messages=FALSE)
# WLS fitting:
> Z.vgm <- variofit(Z.svar, ini=c(var(Z.geo$data), 1000), fix.nugget=T, nugget=0)
> Z.vgm

  variofit: model parameters estimated by WLS (weighted least squares):
  covariance model is: matern with fixed kappa = 0.5 (exponential)
  fixed value for tausq =  0
  parameter estimates:
    sigmasq       phi
  1352.3685  650.9268
  Practical Range with cor=0.05 for asymptotic range: 1950.002

  variofit: minimised weighted sum of squares = 31777661

> env.model <- variog.model.env(Z.geo, obj.var=Z.svar, model=Z.vgm)
> plot(Z.svar, envelope=env.model); lines(Z.vgm, lwd=2);
> dev.off()
```

which shows that the target variable (Z) varies equally in all directions, i.e. it can be modeled using isotropic models (Fig. 10.3). It is also a relatively smooth variable — there is no nugget variation and spatial autocorrelation is valid (practical range) up to a distance of 2 km. This is in fact a typical variogram for elevation data i.e. representation of a land surface. Note also that the confidence bands (envelopes) are now much narrower than in Fig. 5.15, possibly because there are more points and because the feature is more smooth.

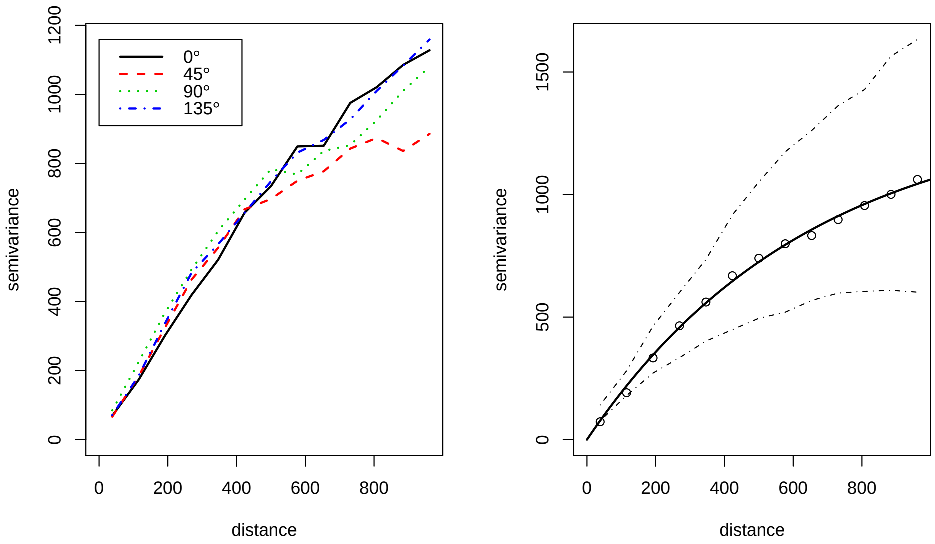


Fig. 10.3: Standard variograms fitted for elevations: (left) anisotropy in four directions; (right) isotropic variogram model fitted using the weighted least squares (WLS) and its confidence bands.

10.3.2 Geostatistical simulations

We can now use the variogram model to generate multiple realizations of the target variable. We first need to create a new empty grid for which a DEM can be derived. An empty grid with full topology can be generated in sp package:

```
> demgrid <- spsample(elevations, type="regular", cellsize=c(30,30))
> gridded(demgrid) <- TRUE
> fullgrid(demgrid) <- TRUE
> demgrid@grid
```

```
                       x1      x2
cellcentre.offset 6551822 5070484
cellsize               30      30
cells.dim             128     130
```

Now that we have fitted the variogram model and prepared a grid of interest, we can simulate *N* DEMs. We will use the **Stochastic Conditional Gaussian Simulations** algorithm as implemented in the gstat[5] package. The number of realizations *N* must be sufficiently large to obtain stable results, but exactly how large *N* should be depends on how accurate the results of the uncertainty analysis should be. The accuracy of the Monte-Carlo method is proportional to the square root of the number of runs *N*. Therefore, to double the accuracy one must quadruple the number of runs (Temme et al., 2008). This means that although many runs may be needed to reach stable and accurate results, any degree of precision can be reached by taking a large enough sample *N*. Consequently, the Monte-Carlo method is computationally demanding, particularly when the GIS operation

[5] Conditional simulations can also be generated in geoR, but gstat is much less time-consuming.

takes significant computing time (Heuvelink, 2002). As a rule of thumb, we will take 100 simulations as large enough.

Because we further use gstat, we need to copy the fitted variogram values to a vgm object:

```
# copy the values fitted in geoR:
> Z.ovgm <- vgm(psill=Z.vgm$cov.pars[1], model="Mat",
+      range=Z.vgm$cov.pars[2], nugget=Z.vgm$nugget, kappa=1.2)
> Z.ovgm

   model    psill    range kappa
 1   Nug    0.000   0.0000     0
 2   Mat 1352.368 650.9268   1.2
```

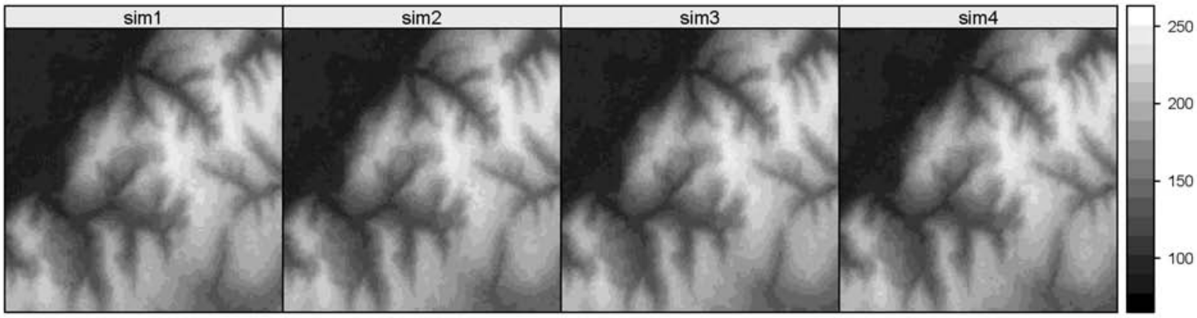


Fig. 10.4: Four realizations of the DEM following conditional geostatistical simulations.

Note that we have purposively set the kappa[6] parameter at 1.2 (it was originally 0.5). Following our knowledge about the feature of interest, we know that a land surface is inherently smooth — due to the erosional processes and permanent leveling of topography — so we wish to generate realizations of DEMs that fit our knowledge of the area.

The conditional simulations in gstat can be run by simply adding the nsim argument to the generic krige method:

```
> N.sim <- 100
> DEM.sim <- krige(Z ~ 1, elevations, demgrid, Z.ovgm, nmax=30, nsim=N.sim)

  drawing 100 GLS realisations of beta...


# this can take few minutes!!
> fullgrid(DEM.sim) <- TRUE
> spplot(DEM.sim[1:4], col.regions=grey(seq(0,1,0.025)))
```

which shows that we have accomplished our objective: we have managed to simulate 100 equiprobable DEMs that are equally smooth as the DEM shown in Fig. 10.1. To visualize the differences between realizations, we can make a cross section and plot all simulated surfaces on top of each other (Fig. 10.5):

```
# Cross-section at y=5,073,012:
> cross.s <- data.frame(X=seq(demgrid@bbox[1,1]+gridcell/2,
+      demgrid@bbox[1,2]-gridcell/2, gridcell), Y=rep(5073012, demgrid@grid@cells.dim[1]))
> coordinates(cross.s) <- ~ X+Y
# proj4string(cross.s) <- elevations@proj4string
> cross.ov <- overlay(DEM.sim, cross.s)
> plot(cross.ov@coords[,1], cross.ov@data[[1]], type="l", xlab="X",
+      ylab="Z", col="grey")
> for(i in 2:N.sim-1){
>   lines(cross.ov@coords[,1], cross.ov@data[[i]], col="grey")
> }
> lines(cross.ov@coords[,1], cross.ov@data[[N.sim]], lwd=2)
```

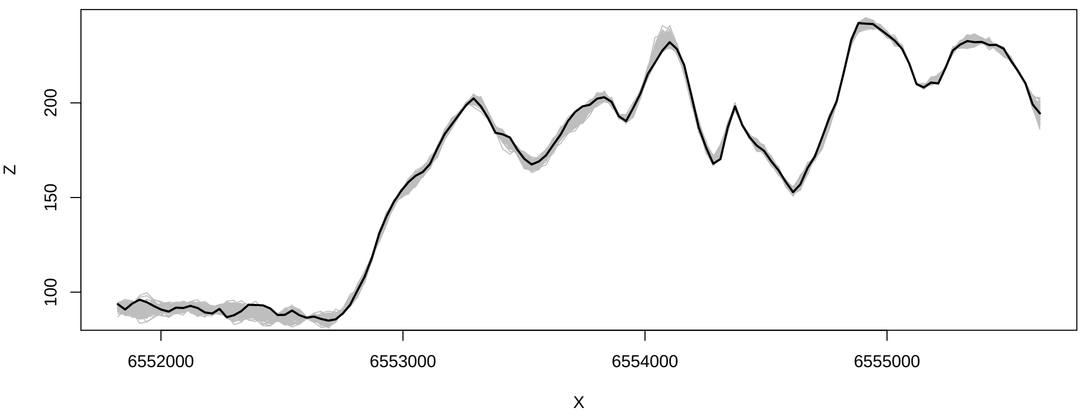


Fig. 10.5: 100 simulations of DEM showing using a cross-section from West to East (cross-section at X=5,073,012). Compare with Temme et al. (2008, p.130).

You will notice that the confidence band is relatively wide (few meters). The confidence band is controlled with the density of sampling locations — the further you get from sampling locations, the higher will be the error. This property we will further explore in §10.5.

Why is a high kappa parameter necessary? If you run DEM simulations with e.g. an exponential model, you will see that the realizations will be much noisier than what we would expect (Hengl et al., 2008). This will happen even if you set the nugget parameters at zero (smooth feature). There are several explanations for this. Having a non-zero grid resolution implies that the correlation between adjacent grid cells is not equal to 1, so that grids may still appear to have noise (Temme et al., 2008). A noisy DEM leads to completely different drainage networks — the streams will be shorter and more random — which we know does not fit knowledge about the area. The Matérn variogram model (Eq.1.3.10), on the other hand, allow us to produce smoother DEMs, while still using objectively estimated nugget, sill and range parameters. This makes it especially suitable for modeling of land surface.

10.4 Generation of stream networks

Now that we have simulated N DEMs, we can derive stream networks using the "*Channel Network*" function, which is available also via the command line i.e. via the `ta_channels` SAGA library. To get complete info about this module you can use:

```
> rsaga.get.usage("ta_channels", 0)
```

First, convert the maps to SAGA format:

```
# write simulated DEMs to SAGA format:
> for(i in 1:N.sim){
>   write.asciigrid(DEM.sim[i], paste("DEM", i, ".asc", sep=""), na.value=-1)
> }
# get a list of files:
> dem.list <- list.files(getwd(), pattern="DEM[[:digit:]]*.asc")
> rsaga.esri.to.sgrd(in.grids=dem.list, out.sgrd=set.file.extension(dem.list,
+     ".sgrd"), in.path=getwd(), show.output.on.console=FALSE)
> unlink(dem.list)
```

Recall that, before we derive a stream network, we also want to remove[7] spurious sinks. For this we can use e.g. the method of Planchon and Darboux (2001):

[6] See Diggle and Ribeiro Jr (2007, p.51–53).

[7] This is also based on the empirical knowledge: water typically erodes small obstacles and creates continuous paths.

```
> stream.list <- list(rep(NA, N.sim))
> for (i in 1:N.sim) {
# First, filter the spurious sinks:
>  rsaga.geoprocessor(lib="ta_preprocessor", module=2,
+      param=list(DEM=paste("DEM", i, ".sgrd", sep=""),
+      RESULT="DEMflt.sgrd", MINSLOPE=0.05), show.output.on.console=FALSE)
# Second, extract the channel network:
>  rsaga.geoprocessor(lib="ta_channels", module=0, param=list(ELEVATION="DEMflt.sgrd",
+      CHNLNTWRK=paste("channels", i, ".sgrd", sep=""), CHNLROUTE="channel_route.sgrd",
+      SHAPES="channels.shp", INIT_GRID="DEMflt.sgrd", DIV_CELLS=3, MINLEN=40),
+      show.output.on.console=FALSE)
# read vector maps into R:
>  stream.list[[i]] <- readOGR("channels.shp", "channels")
>  proj4string(stream.list[[i]]) <- elevations@proj4string
> }
```

Here we use arbitrary input parameters for the minimum length of streams (40) and initial grid, but this is not relevant for this exercise. Note that we do not really need all maps, but only the vector map showing the position of streams. Therefore, we can recycle temporary maps in each loop.

Once the processing is finished, we can visualize all derived streams on top of each other:

```
# plot all derived streams on top of each other:
> stream.plot <- as.list(rep(NA, N.sim))
> for(i in 1:N.sim){
>   stream.plot[[i]] <- list("sp.lines", stream.list[[i]])
> }
> lines.plt <- spplot(DEM.sim[1], col.regions=grey(seq(0.5,1,0.025)),
+      scales=list(draw=T), sp.layout=stream.plot, main="100 streams")
```

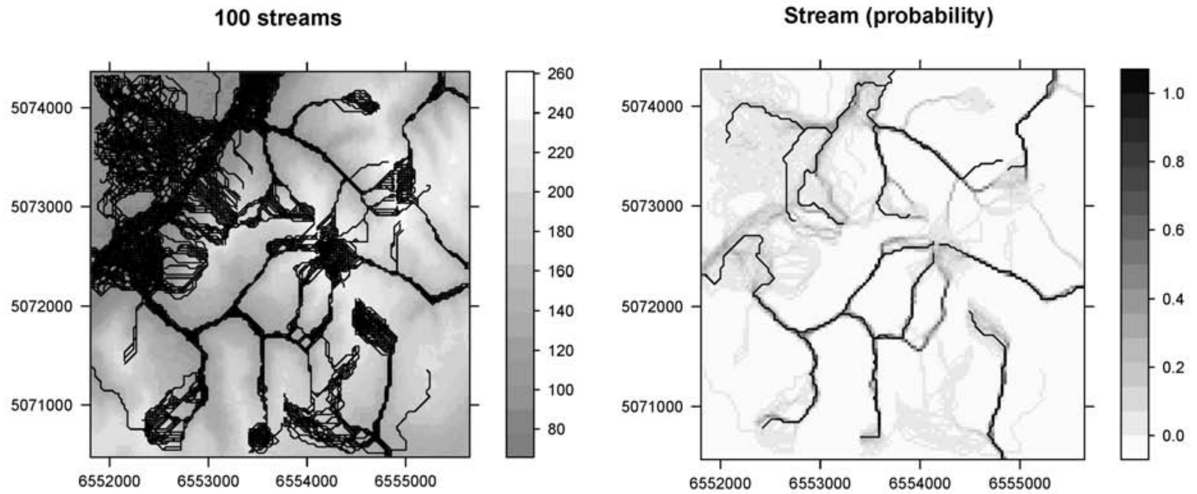


Fig. 10.6: 100 realizations of stream network overlaid on top of each other (left); probability of stream network, overlaid with one realization (right).

which is shown in Fig. 10.6 (left). This visualization of density of streams illustrates the concept of propagated uncertainty. If you zoom in into this map, you will notice that the streams follow the gridded-structure of the DEMs, which explains some artificial breaks in the lines.

To actually derive a probability of mapping a stream, we need to import all gridded maps of streams, then count how many times the model estimated a stream over a certain grid node:

```
# get the list of maps:
> streamgrid.list <- list.files(getwd(), pattern="channels[[:digit:]]*.sgrd")
> rsaga.sgrd.to.esri(in.sgrds=streamgrid.list,
+    out.grids=set.file.extension(streamgrid.list, ".asc"),
+    out.path=getwd(), prec=0, show.output.on.console=FALSE)
# read all grids into R:
> streamgrid <- readGDAL(set.file.extension(streamgrid.list[[1]], ".asc"))
> streamgrid@data[[1]] <- ifelse(streamgrid$band1<0, 0, 1)
> for(i in 2:length(streamgrid.list)){
>   tmp <- readGDAL(set.file.extension(streamgrid.list[[i]], ".asc"))
# convert to a binary map:
>   streamgrid@data[[i]] <- ifelse(tmp$band1<0, 0, 1)
> }
> names(streamgrid) <- set.file.extension(streamgrid.list, ".asc")
> proj4string(streamgrid) <- elevations@proj4string
```

Now we have a pack of grids (`streamgrid`), with 0/1 values depending on the stream occurrence (yes/no). These can be summed using the `rowSums` method:

```
> streamgrid$pr <- rowSums(streamgrid@data, na.rm=TRUE, dims=1)/length(streamgrid@data)
```

and the probability of detecting a stream is simply the average value of stream from multiple simulations. This map is shown in Fig. 10.6 (right):

```
> stream.plt <- spplot(streamgrid["pr"], col.regions=grey(rev((1:59)/60)),
+      scales=list(draw=T), sp.layout=list("sp.lines", stream.list[[1]]),
+      main="Stream (probability)")
> print(lines.plt, split=c(1,1,2,1), more=TRUE)
> print(stream.plt, split=c(2,1,2,1), more=FALSE)
```

Next, we would like to derive the propagated uncertainty of mapping a stream. Theoretically speaking, stream is a discrete feature that follows a Bernoulli distribution which takes value 1 with success probability p and value 0 with failure probability $q{=}1-p$. Thus the error of mapping a stream can be derived as: $-q\ln(q)-p\ln(p)$, or in R syntax:

```
> streamgrid$pr.var <- -streamgrid$pr*log2(streamgrid$pr)
+              -(1-streamgrid$pr)*log2(1-streamgrid$pr)
```

which means that the highest uncertainty of mapping a stream is when p approaches 0.5 (equal probability of stream and not-stream). As anticipated, the propagated variability of detecting a stream is much higher (in cumulative terms) in the terrace region of the study area (Fig. 10.5). The remaining issue is whether we could explain this variability using some topographic, land surface parameters.

10.5 Evaluation of the propagated uncertainty

Now that we have estimated the propagated uncertainty of extracting channel networks (streams) from DEMs, we can try to understand how this uncertainty relates to the geomorphology of terrain. We will derive and run a comparison by using only a few land surface parameters; you might consider extending the list. First, we can derive mean value (the '*most probable*' DEM) and standard deviation i.e. the propagated uncertainty of mapping elevation:

```
> rsaga.geoprocessor(lib="geostatistics_grid", module=5,
+     param=list(GRIDS=paste(set.file.extension(dem.list,".sgrd"), collapse=";"),
+     MEAN="DEM_avg.sgrd", STDDEV="DEM_std.sgrd"), show.output.on.console=FALSE)
> rsaga.sgrd.to.esri(in.sgrds=c("DEM_avg.sgrd","DEM_std.sgrd"),
+     out.grids=c("DEM_avg.asc","DEM_std.asc"), out.path=getwd(), prec=2)
```

Next, it is interesting to derive a map of slope, as it largely controls the hydrological properties, and the difference from the mean value in 5×5 search radius [8]:

[8] Note that these are wrapper function, which means that they combine several operations together — in this case derivation of slope and conversion of maps to ArcInfo ASCII format.

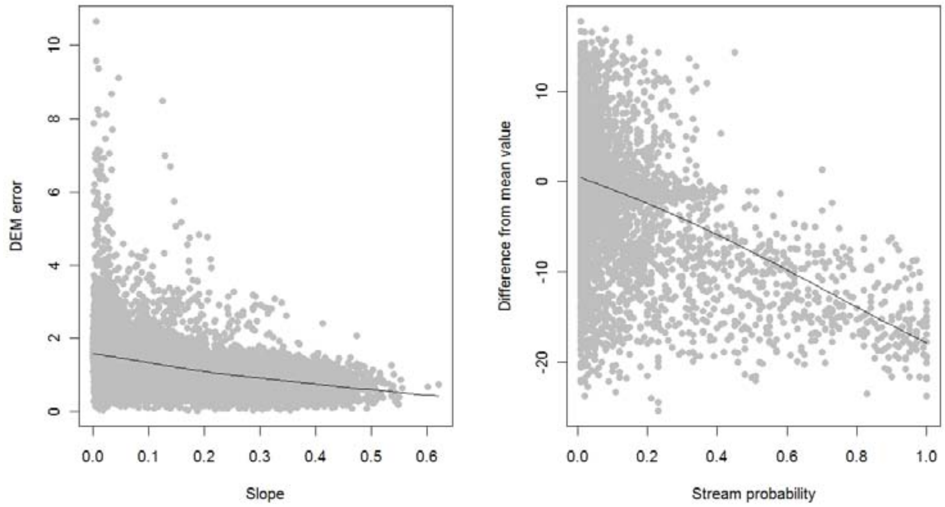


Fig. 10.7: Relationship between the standard error of interpolating elevation and local slope; and probability of deriving streams and difference from the mean elevation.

```
# slope:
> rsaga.esri.wrapper(rsaga.slope, method="poly2zevenbergen", in.dem="DEM_avg.sgrd",
+      out.slope="SLOPE.sgrd", prec=3, clean.up=F)
# residual analysis:
> rsaga.geoprocessor(lib="geostatistics_grid", 0, param=list(INPUT="DEM_avg.sgrd",
+      MEAN="tmp.sgrd", STDDEV="tmp.sgrd", RANGE="tmp.sgrd", DEVMEAN="tmp.sgrd",
+      PERCENTILE="tmp.sgrd", RADIUS=5, DIFF="DIFMEAN.sgrd"))
```

and then read back the results to R:

```
> rsaga.sgrd.to.esri(in.sgrds=c("DEM_avg.sgrd","DEM_std.sgrd", "DIFMEAN.sgrd"),
+    out.grids=c("DEM_avg.asc","DEM_std.asc", "DIFMEAN.asc"), out.path=getwd(), prec=2)
# read back into R:
> gridmaps <- readGDAL("DEM_avg.asc")
> names(gridmaps) <- "DEM"
> gridmaps$std <- readGDAL("DEM_std.asc")$band1
> gridmaps$SLOPE <- readGDAL("SLOPE.asc")$band1
> gridmaps$DIFMEAN <- readGDAL("DIFMEAN.asc")$band1
```

Which allows us to plot the two DEM parameters versus the probability of finding stream and propagated DEM error:

```
> par(mfrow=c(1,2))
> scatter.smooth(gridmaps$SLOPE, gridmaps$std, span=18/19, col="grey",
+     xlab="Slope", ylab="DEM error", pch=19)
> scatter.smooth(streamgrid$pr[streamgrid$pr>0], gridmaps$DIFMEAN[streamgrid$pr>0],
+     span=18/19, col="grey", xlab="Stream probability", ylab="Dif. from mean", pch=19)
> dev.off()
```

Fig. 10.7 shows two interesting things: (1) the errors in elevations are largely controlled by the slope; (2) streams are especially difficult to map in areas where the difference from the mean value is high, i.e. close to zero or positive (meaning areas with low local relief or close to concave shapes). This largely reflects our expectation, but it is rewarding to be able to prove these assumptions using hard data.

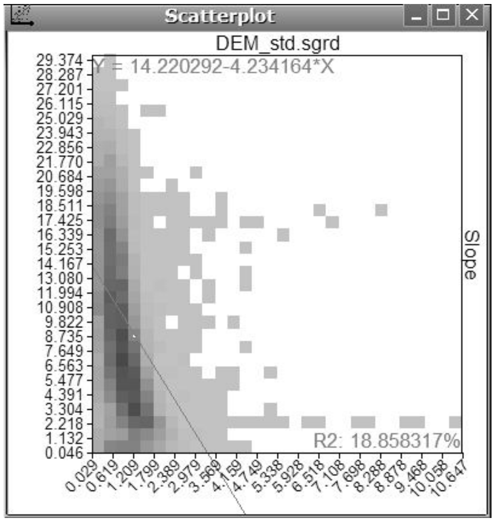


Fig. 10.8: Scatter plot in SAGA GIS. This is the same plot as shown in Fig. 10.7.

Optional: open all derived maps in SAGA and visualize correlations between the probability of streams and DEM error versus various DEM parameters using the scatter plot option. To produce a plot shown in Fig. 10.8 right click on the raster map of interest (`DEM_std`) and then select "*Show scatter-plot*" ↦ select grid. You can adjust the size of grid cells and default representation in this plot by editing properties (from the main menu).

10.6 Advanced exercises

10.6.1 Objective selection of the grid cell size

In the previous exercise we set the grid cell size at 30 m, without any real justification. Now we can consider statistically sound approach to select a grid cell size based on the accuracy of the derived stream network. This follows the idea of Hutchinson (1996), who use an iterative DEM cell-size optimization algorithm as implemented in the ANUDEM package. By plotting the error of mapping streams versus the grid spacing index, one can select the grid cell size that shows the maximum information content in the final map. The optimal grid cell size is the one where further refinement does not change the accuracy of derived streams.

We can implement this principle using our case study, i.e. we can derive drainage networks using several grid cell sizes and then see if the spatial location of streams differs significantly from the one derived using the finest resolution (see Fig. 10.9). We can start with generating DEMs from point data (e.g. using splines) using a range of grid cell sizes:

```
> pixel.range <- c(20, 30, 40, 50, 60, 80, 100)
# generate DEMs using splines:
> for(i in 1:length(pixel.range)){
>   rsaga.geoprocessor(lib="grid_spline", module=1,
+      param=list(GRID=paste("DEMpix", pixel.range[i], ".sgrd", sep=""),
+      SHAPES="elevations.shp", FIELD=1, RADIUS=sqrt(areaSpatialGrid(demgrid))/3,
+      SELECT=1, MAXPOINTS=10, TARGET=0, USER_CELL_SIZE=pixel.range[i],
+      USER_X_EXTENT_MIN=demgrid@bbox[1,1]+pixel.range[i]/2,
+      USER_X_EXTENT_MAX=demgrid@bbox[1,2]-pixel.range[i]/2,
+      USER_Y_EXTENT_MIN=demgrid@bbox[2,1]+pixel.range[i]/2,
+      USER_Y_EXTENT_MAX=demgrid@bbox[2,2]-pixel.range[i]/2))
> }
```

Next, we can derive stream networks for each DEM, and then buffer distance to the stream network using a loop. We set the minimum length of stream (`min.len`) based on the cell size of DEM:

```
# estimate drainage map for each DEM:
> for(i in 1:length(pixel.range)){
# filter the spurious sinks:
>   rsaga.geoprocessor(lib="ta_preprocessor", module=2,
+      param=list(DEM=paste("DEMpix", pixel.range[i], ".sgrd", sep=""),
+      RESULT="DEMflt.sgrd", MINSLOPE=0.05), show.output.on.console=FALSE)
# minimum length:
>   min.len <- round(sqrt(areaSpatialGrid(demgrid))/(pixel.range[i]*3.5), 0)
# extract the channel network:
>   rsaga.geoprocessor(lib="ta_channels", module=0,
+      param=list(ELEVATION="DEMflt.sgrd", CHNLNTWRK=paste("chnlntwrk_pix",
+      pixel.range[i], ".sgrd", sep=""), CHNLROUTE="tmp.sgrd",
+      SHAPES=paste("channels_pix", i, ".shp", sep=""), INIT_GRID="DEMflt.sgrd",
```

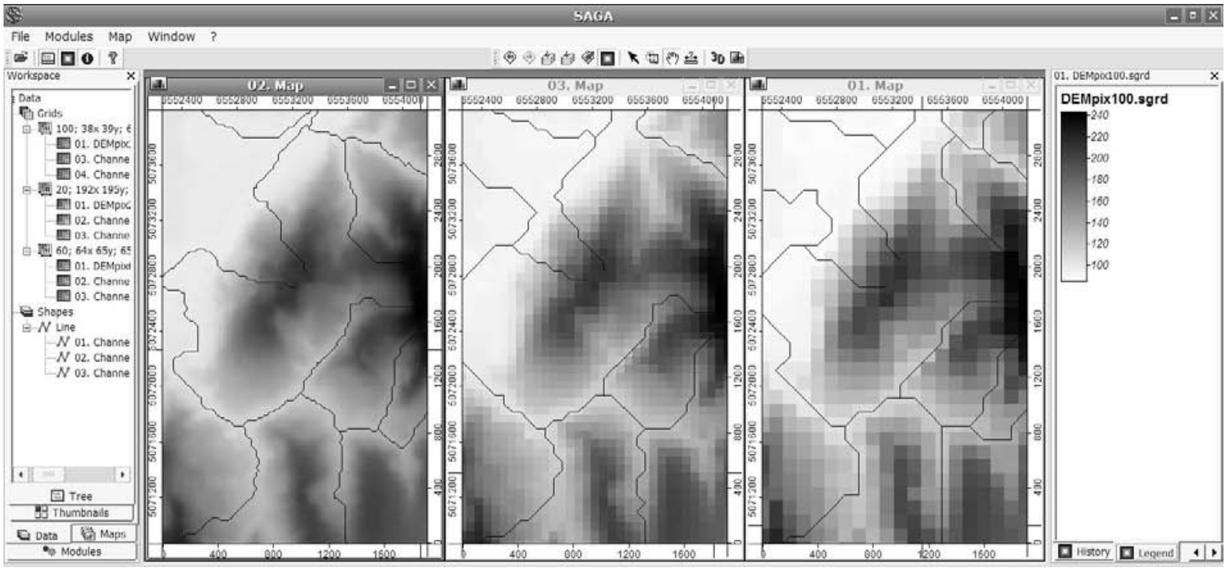


Fig. 10.9: Drainage network derived using different grid cell sizes.

```
+      DIV_CELLS=3, MINLEN=min.len), show.output.on.console=FALSE)
# buffer distance to actual streams (use the finest grid cell size):
>   rsaga.geoprocessor(lib="grid_gridding", module=0,
+      param=list(GRID="stream_pix.sgrd",
+      INPUT=paste("channels_pix", i, ".shp", sep=""), FIELD=1, LINE_TYPE=0,
+      USER_CELL_SIZE=pixel.range[1],
+      USER_X_EXTENT_MIN=demgrid@bbox[1,1]+pixel.range[1]/2,
+      USER_X_EXTENT_MAX=demgrid@bbox[1,2]-pixel.range[1]/2,
+      USER_Y_EXTENT_MIN=demgrid@bbox[2,1]+pixel.range[1]/2,
+      USER_Y_EXTENT_MAX=demgrid@bbox[2,2]-pixel.range[1]/2),
+      show.output.on.console=FALSE)
# extract a buffer distance map:
>   rsaga.geoprocessor(lib="grid_tools", module=10,
+      param=list(SOURCE="stream_pix.sgrd", DISTANCE="tmp.sgrd",
+      ALLOC="tmp.sgrd", BUFFER=paste("buffer_pix", pixel.range[i], ".sgrd", sep=""),
+      DIST=2000, IVAL=pixel.range[1]), show.output.on.console=FALSE)
> }
```

and then read maps into R:

```
> rsaga.sgrd.to.esri(in.sgrds="chnlntwrk_pix20.sgrd",
+     out.grids="chnlntwrk_pix20.asc", out.path=getwd(), prec=1)
> griddrain <- readGDAL("chnlntwrk_pix20.asc")
> names(griddrain) <- "chnlntwrk"
> for(i in 2:length(pixel.range)){
>   rsaga.sgrd.to.esri(in.sgrds=paste("buffer_pix", pixel.range[i], ".sgrd", sep=""),
+       out.grids=paste("buffer_pix", pixel.range[i], ".asc", sep=""),
+       out.path=getwd(), prec=1)
>   griddrain@data[paste("buffer_pix", pixel.range[i], sep="")] <-
+       readGDAL(paste("buffer_pix", pixel.range[i], ".asc", sep=""))$band1
> }
> str(griddrain@data)
```

```
 'data.frame':   37440 obs. of  7 variables:
  $ chnlntwrk    : num  NA NA NA NA NA NA NA NA NA NA ...
  $ buffer_pix30 : num  500 500 480 460 460 440 420 420 400 400 ...
  $ buffer_pix40 : num  500 480 460 460 440 420 420 400 400 380 ...
  $ buffer_pix50 : num  520 500 500 480 460 460 440 440 420 420 ...
  $ buffer_pix60 : num  540 540 520 500 500 480 460 440 440 420 ...
  $ buffer_pix80 : num  500 480 460 440 440 420 400 400 380 380 ...
  $ buffer_pix100: num  500 480 460 460 440 420 420 4
```

which shows distance from the channel network derived using the finest resolution and increasingly coarser resolutions (30, 40, ...100 m). This allows us to compare how much the stream networks deviate from the '*true*' stream:

```
# summary statistics:
> stream.dist <- as.list(rep(NA, length(pixel.range)))
> mean.dist <- c(0, rep(NA, length(pixel.range)-1))
> for(i in 2:length(pixel.range)){
>   stream.dist[[i]] <- summary(griddrain@data[!is.na(griddrain$chnlntwrk),i])
>   mean.dist[i] <- stream.dist[[i]][4]
> }
# final plot:
> plot(pixel.range, mean.dist, xlab="Grid cell size", ylab="Error", pch=19)
> lines(pixel.range, mean.dist)
```

which will produce the plot shown in Fig. 10.10. Surprisingly, resolution of 50 m is even better than the 30 m resolution; with coarser resolutions (>50 m) the spatial accuracy of mapping streams progressively decreases (average error already >70 m). Note also that these results indicate that there there are '*jumps*' in the accuracy: the stream extraction algorithm generates similar results for resolutions 30–50 m, then it again stabilizes at 80–100 m resolutions.

It appears that we could save a lot of processing time if we would use a resolution of 50 m (instead of 30 m) for this type of modeling. Note that we estimated the error using a single realization of the DEM. One would again need to run such analysis using simulated DEMs at different resolutions, to prove that this difference in accuracy for different grid cell sizes is not an accident.

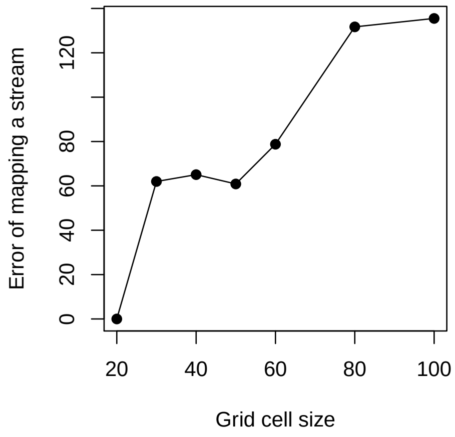


Fig. 10.10: Mean location error of stream network for varying grid cell size (values of both coordinates are in meters).

10.6.2 Stream extraction in GRASS

An equally good alternative to SAGA for processing of elevation data and extraction of DEM parameters and hydrological features is GRASS GIS (Neteler and Mitasova, 2008). We will now run, just for a comparison, extraction of streams in GRASS using the same data set[9]. First, you need to obtain and install GRASS GIS to your computer. After you have finished installing GRASS, switch to your R session and start the spgrass6[10] library that will allow us to control GRASS from R:

```
> library(spgrass6) # version => 0.6-1

  Loading required package: XML
  GRASS GIS interface loaded with GRASS version: (GRASS not running)
```

Because we do not want to worry where GRASS saves temporary files, we can simply assign the environmental parameters to some temporary directory:

```
# Location of your GRASS installation:
> loc <- initGRASS("C:/GRASS", home=tempdir())
> loc
```

[9] The following examples are based on the Windows XP OS. There can be large differences between different OS and different versions of GRASS/spgrass6!

[10] http://cran.r-project.org/web/packages/spgrass6/

```
  gisdbase    c:/WINNT/profiles/software/LOCALS~1/Temp/RtmpdeK5s9
  location    file678418be
  mapset      file3d6c4ae1
  rows        1
  columns     1
  north       1
  south       0
  west        0
  east        1
  nsres       1
  ewres       1
  projection  NA
```

Note that these settings are in fact nonsense. We will soon replace these with the actual parameters once we import an actual raster map. What is important is that we have established a link with GRASS and set the working directory. We can proceed with importing the previously derived DEM into GRASS:

```
> parseGRASS("r.in.gdal") # commmand description

  Command: r.in.gdal
  Description: Import GDAL supported raster file into a binary raster map layer.
  Keywords: raster, import
  Parameters:
    name: input, type: string, required: no, multiple: no
   [Raster file to be imported]
   ...


# Import the ArcInfo ASCII file to GRASS:
> execGRASS("r.in.gdal", flags="o", parameters=list(input="DEM_avg.asc", output="DEM"))

  WARNING: Over-riding projection check

  RINGDA~1 complete. Raster map <DEM> created.
```

We can use the parameters of the imported map to set up the geographic region:

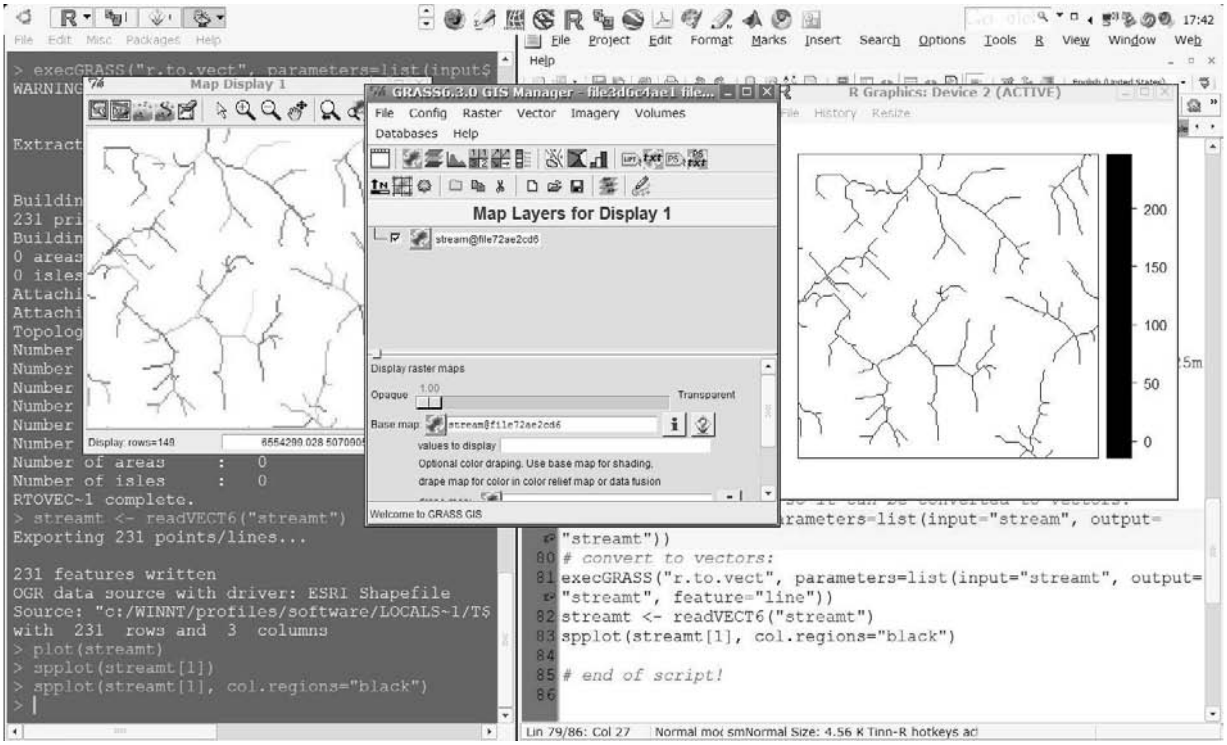
```
> execGRASS("g.region", parameters=list(rast="DEM"))
> gmeta6()

  gisdbase    c:/WINNT/profiles/software/LOCALS~1/Temp/RtmpdeK5s9
  location    file678418be
  mapset      file3d6c4ae1
  rows        130
  columns     128
  north       5074369
  south       5070469
  west        6551807
  east        6555647
  nsres       30
  ewres       30
  projection  NA
```

which is the averaged DEM from multiple realizations derived in §10.5. Note that Windows system generates the temporary name of the mapset and location. Again, we do not worry too much about this because we use GRASS as an external application to run geographical analysis; temporary files will be recycled, at the end we will read only the final results back into R.

We proceed with the extraction of the drainage network:

```
# extract the drainage network:
> execGRASS("r.watershed", flags=c("m", "overwrite"),
+     parameters=list(elevation="DEM", stream="stream", threshold=as.integer(50)))
```

```
  SECTION 1 beginning: Initiating Variables. 5 sections total.
  SECTION 1b (of 5): Determining Offmap Flow.

  SECTION 2: A * Search.

  SECTION 3: Accumulating Surface Flow.

  SECTION 4: Watershed determination.

  SECTION 5: Closing Maps.
```



```
> execGRASS("r.to.vect", parameters=list(input$
WARNING

Extract


Buildin
231 pri
Buildin
0 areas
0 isles
Attachi
Attachi
Topolog
Number
Number
Number
Number
Number
Number
Number of areas      :    0
Number of isles      :    0
RTOVEC~1 complete.
> streamt <- readVECT6("streamt")
Exporting 231 points/lines...

231 features written
OGR data source with driver: ESRI Shapefile
Source: "c:/WINNT/profiles/software/LOCALS~1/T$
with  231  rows and  3  columns
> plot(streamt)
> spplot(streamt[1])
> spplot(streamt[1], col.regions="black")
> |
```

```
rameters=list(input="stream", output=
"streamt"))
# convert to vectors:
execGRASS("r.to.vect", parameters=list(input="streamt", output=
"streamt", feature="line"))
streamt <- readVECT6("streamt")
spplot(streamt[1], col.regions="black")

# end of script!

```

Lin 79/86: Col 27 Normal mo(smNormal Size: 4.56 K Tinn-R hotkeys ac

Map Layers for Display 1

stream@file72ae2cd6

Display raster maps

Opaque 1.00 Transparent

Base map: stream@file72ae2cd6

values to display

Optional color draping. Use base map for shading,
drape map for color in color relief map or data fusion

Welcome to GRASS GIS

Display: rows=148 6554299.028 507090

File Edit Misc Packages Help
Map Display 1
GRASS6.3.0 GIS Manager - file3d6c4ae1 file...
File Config Raster Vector Imagery Volumes
Databases Help
File Project Edit Format Marks Insert Search Options Tools R View Window Web
Help
R Graphics: Device 2 (ACTIVE)
File History Resize
200
150
100
50
0
17:42
25m

Fig. 10.11: Example of a screen shot — drainage extraction steps using the Baranja hill data set in GRASS GIS.

which will generate a raster map showing the position of streams (Fig. 10.11). Note that GRASS typically generates a rich output with many technical details of interest to a specialist. Before we can convert the derived map to a vector layer, we need to thin it:

```
> execGRASS("r.thin", parameters=list(input="stream", output="streamt"))

  File stream -- 130 rows X 128 columns
  Bounding box: l = 2, r = 129, t = 2, b = 131
  Pass number 1
  Deleted 55 pixels
  Pass number 2
  Deleted 0 pixels
  Thinning completed successfully.
  Output file 130 rows X 128 columns
  Window 130 rows X 128 columns


# convert to vectors:
> execGRASS("r.to.vect", parameters=list(input="streamt",
+             output="streamt", feature="line"))
```

```
WARNING: Default driver / database set to:
         driver: dbf
         database: $GISDBASE/$LOCATION_NAME/$MAPSET/dbf/
Extracting lines...

Building topology for vector map <streamt>...
Registering primitives...

165 primitives registered
652 vertices registered
Building areas...

0 areas built
0 isles built
Attaching islands...
Attaching centroids...

Number of nodes: 176
Number of primitives: 165
Number of points: 0
Number of lines: 165
Number of boundaries: 0
Number of centroids: 0
Number of areas: 0
Number of isles: 0
RTOVEC ~1 complete.
```

Processing is now complete, so we can read the produced map from R. For this, we use the generic spgrass6 command that reads any type of GRASS vector:

```
# read the generated stream network map into R:
> streamt <- readVECT6("streamt")

  Exporting 165 points/lines...

  165 features written
  OGR data source with driver: ESRI Shapefile
  Source: "c:/WINNT/profiles/software/LOCALS~1/Temp/RtmpdeK5s9/file3d6c4ae1/.tmp",
  layer: "streamt"
  with  165  rows and  3  columns
  Feature type: wkbLineString with 2 dimensions


> plot(streamt)
```

which shows a similar result as shown in Fig. 10.1. You can now open all maps you have generated and visualize them in GRASS. Fig. 10.11 will give you some idea about the GRASS interface. In summary, there are noticable differences in the ways the things are run with SAGA and GRASS. GRASS seem to be more demanding considering the control of the package. On the other hand, it is a much larger and more international project than SAGA. It is really a question of taste if one prefers to use one or the other, but there are also no obstacles to combine them.

10.6.3 Export of maps to GE

In the final step we will export the stream probability map from R to Google Earth, this time without using SAGA GIS[11]. We can start by re-projecting the derived grid to the latitude longitude system:

```
> streamgrid.ll <- spTransform(streamgrid["pr"], CRS("+proj=longlat +datum=WGS84"))
> streamgrid.ll@bbox
```

[11] This is somewhat more complicated. Compare with §5.6.2.

```
        min      max
  x 18.66122 18.71065
  y 45.77646 45.81158
```

which will create a point map (not a grid!), which means that we need to create the grid topology in the `longlat` coordinate system ourselves. To do this, we first need to estimate the grid cell size in arcdegrees, e.g. by using the Eq.(3.3.1) explained in §3.3.1. The width correction factor[12] based on the latitude of the center of the study area, can be estimated as:

```
> corrf <- (1 + cos((streamgrid.ll@bbox[1, "max"] +
+          streamgrid.ll@bbox[2, "min"])/2 * pi/180))/2
```

and then the grid cell size in arcdegrees is approximately:

```
> geogrd.cell <- corrf*(streamgrid.ll@bbox[1,"max"] -
+          streamgrid.ll@bbox[1, "min"]) / streamgrid@grid@cells.dim[1]
> geogrd.cell

  [1] 0.0003564123
```

which means that a width of a 30 m pixel at this latitude corresponds to about 1.3 arcseconds.

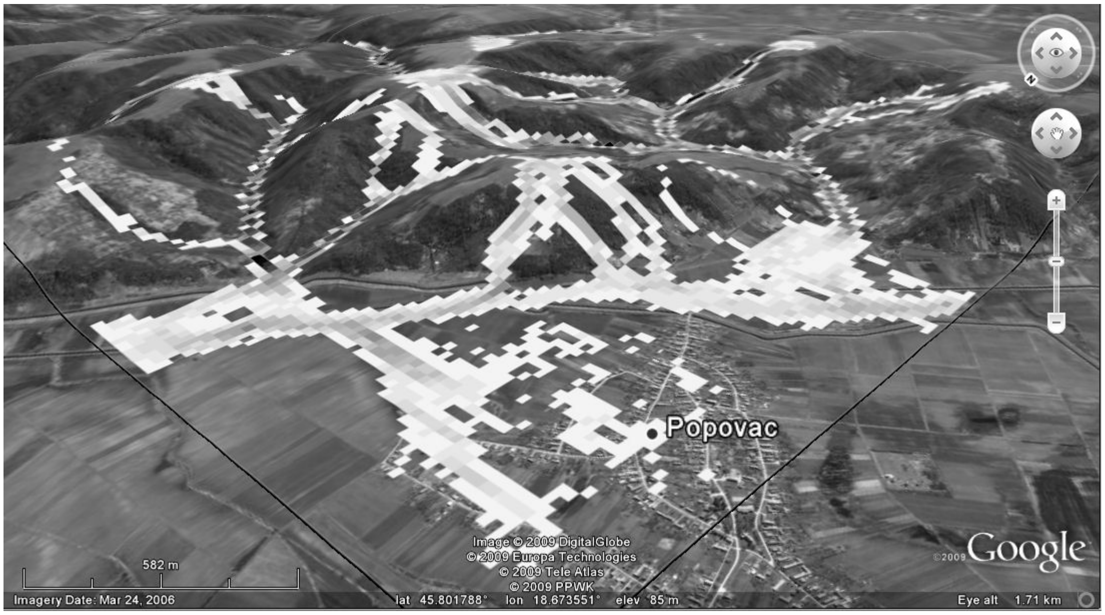

Popovac
582 m
Imagery Date: Mar 24, 2006
Image © 2009 DigitalGlobe
© 2009 Europa Technologies
© 2009 Tele Atlas
© 2009 PPWK
lat 45.801788° lon 18.673551° elev 85 m
©2009 Google
Eye alt 1.71 km

Fig. 10.12: Baranja hill and derived stream probability, visualized in Google Earth.

Once we have estimated the grid cell size in geographical coordinates, we can also generate the new grid system:

```
> geoarc <- spsample(streamgrid.ll, type ="regular", cellsize=c(geogrd.cell, geogrd.cell))
> gridded(geoarc) <- TRUE
> gridparameters(geoarc)

     cellcentre.offset     cellsize
  x1          18.66127 0.0003564123
  x2          45.77662 0.0003564123
     cells.dim
  x1       139
  x2        99
```

[12] For data sets in geographical coordinates, a cell size correction factor can be estimated as a function of the latitude and spacing at the equator.

```
> geoarc.grid <- SpatialGridDataFrame(geoarc@grid,
+       data=data.frame(rep(1, length(geoarc@grid.index))),
+       proj4string=streamgrid.ll@proj4string)
```

Now we need to estimate values of our target variable at the new grid nodes. We use the interp method as implemented in the akima package, which leads to a simple bilinear resampling:

```
> library(akima)
> streamgrid.llgrd <- interp(x=streamgrid.ll@coords[,1], y=streamgrid.ll@coords[, 2],
+    z=streamgrid.ll$pr, xo=seq(geoarc.grid@bbox[1, "min"], geoarc.grid@bbox[1, "max"],
+    length=geoarc.grid@grid@cells.dim[[1]]), yo=seq(geoarc.grid@bbox[2, "min"],
+    geoarc.grid@bbox[2, "max"], length=geoarc.grid@grid@cells.dim[[2]]),
+    linear=TRUE, extrap=FALSE)
# convert to sp class:
> streamgrid.llgrd <- as(as.im(streamgrid.llgrd), "SpatialGridDataFrame")
> proj4string(streamgrid.llgrd) <- CRS("+proj=longlat +datum=WGS84")
# mask the "0" values:
> streamgrid.llgrd$pr <- ifelse(streamgrid.llgrd$v < 0.05, NA, streamgrid.llgrd$v)
```

which allows us to finally generate a KML ground overlay for Google Earth (Fig. 10.12):

```
> streamgrid.kml <- GE_SpatialGrid(streamgrid.llgrd)
> png(file="stream.png", width=streamgrid.kml$width,
+     height=streamgrid.kml$height, bg="transparent")
> par(mar=c(0, 0, 0, 0), xaxs="i", yaxs="i")
> image(as.image.SpatialGridDataFrame(streamgrid.llgrd["pr"]),
+     col=rev(), xlim=streamgrid.kml$xlim,
+     ylim=streamgrid.kml$ylim)
> kmlOverlay(streamgrid.kml, "stream.kml", "stream.png",
+     name="Stream probability")

 [1] "<?xml version='1.0' encoding='UTF-8'?>"
 [2] "<kml xmlns='http://earth.google.com/kml/2.0'>"
 [3] "<GroundOverlay>"
 [4] "<name>Stream probability</name>"
 [5] "<Icon><href>stream.png</href><viewBoundScale>0.75</viewBoundScale></Icon>"
 [6] "<LatLonBox><north>45.8119041798664</north><south>45.7762593102449</south>
     <east>18.7108375328584</east><west>18.6609075254189</west></LatLonBox>"
 [7] "</GroundOverlay></kml>"
```

```
> dev.off()

 windows
       2
```

Visualization of generated maps in Google Earth is important for several reasons: (1) we can check if the coordinate system definition is correct; (2) we can evaluate and interpret the results of mapping using high resolution satellite imagery; (3) Google Earth allows 3D exploration of data, which is ideal for this type of exercise (read more in §3.3.1).

Before closing the R session, it is also advisable to clean up all the temporary files:

```
> save.image(".RData")
> unlink("*.hgrd")
> unlink("*.sgrd")
> unlink("*.sdat")
> unlink("DEM**.***")
> unlink("channels**.***")
```

Self-study exercises:

(1.) What is the sampling density of the `elevations` map in no./ha? (HINT: divide number of points per size of area.)

(2.) How precise is the interpolation of elevations overall following the ordinary kriging model? (HINT: run ordinary kriging and derive mean value of the kriging variance for the whole area.)

(3.) What is the inter-quantile range of derived stream probability? (HINT: derive summary statistics and then take the first and third quantile.)

(4.) How does the precision of generating DEM changes considering the edge of area? (HINT: plot the standard deviation of generated DEMs versus the edge contamination map that you can derive in SAGA GIS.)

(5.) How does the probability of mapping streams change with PLANC? (HINT: derive a correlation coefficient between propagated error and mapped values; plot a scatter plot.)

(6.) What is the percentage of area with stream probability >0.5?

(7.) How much is 50 m in arcdegrees for this study area? And in arcsecond?

Further reading:

★ Hengl, T., Bajat, B., Reuter, H. I., Blagojević, D., 2008. Geostatistical modelling of topography using auxiliary maps. Computers & Geosciences, 34: 1886–1899.

★ Hengl, T., Reuter, H. (Eds.), 2008. **Geomorphometry: Concepts, Software, Applications**. Vol. 33 of Developments in Soil Science. Elsevier, Amsterdam, p. 772.

★ Heuvelink, G. B. M. 2002. Analysing uncertainty propagation in GIS: why is it not that simple?. In: Foody, G. M., Atkinson, P. M. (Eds.), Uncertainty in Remote Sensing and GIS, Wiley, Chichester, pp. 155–165.

★ Temme, A. J. A. M., Heuvelink, G. B. M., Schoorl, J. M., Claessens, L., 2008. Geostatistical simulation and error propagation in geomorphometry. In: Hengl, T., Reuter, H. I. (Eds.), Geomorphometry: Concepts, Software, Applications, volume 33: Developments in Soil Science. Elsevier, Amsterdam, 121–140.

★ `http://geomorphometry.org` — The Geomorphometry research group.

11

Land surface temperature (HRtemp)

11.1 Introduction

In this exercise we use one year of measurements of daily mean temperature in Croatia; kindly provided by Melita Perčec–Tadić from the Croatian Meteorological and Hydrological Service[1]. Croatia is a relatively small country but it comprises several different climate regions, which is result from its specific position on the Adriatic sea and fairly diverse topography ranging from plains on the east, through a hilly central part to the mountains separating the continental from the maritime part of the country.

Weather systems originating or crossing over Croatian territory are strongly influenced by this topography, thus the influence they have on weather and climate is highly dependent on the region. The temperature measurements are automatically collected at 123 meteorological stations (Fig. 11.1). The spatial distribution of the stations is not ideal (Zaninović et al., 2008): there is a certain under-sampling at higher elevations and in areas with lower population density (for practical reasons, areas of higher population density have been given a priority). Hence, one could expect that mapping accuracy will be lower at higher elevations and in highlands. In addition, some well-known smaller scale effects cannot be represented successfully, e.g. the Zagreb urban heat island that is, according to measurement, 0.5–1.5°C warmer from the surrounding countryside.

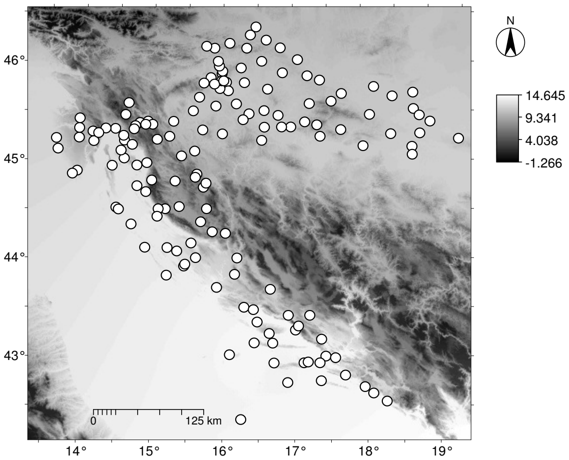


Fig. 11.1: Location of climatic stations in Croatia and long-term monthly temperature for April from the Climatic Atlas of Croatia (Zaninović et al., 2008).

We will model temperature as a function of elevation, distance from the sea, latitude, longitude, time of the year and MODIS daily LST images:

$$\begin{aligned} LST(\mathbf{s}_0, t_0) = {} & b_0 + b_1 \cdot DEM(\mathbf{s}_0) + b_2 \cdot LAT(\mathbf{s}_0) + b_3 \cdot LON(\mathbf{s}_0) + b_4 \cdot DISTC(\mathbf{s}_0) \\ & + b_5 \cdot \cos\left([t_0 - \phi] \cdot \frac{\pi}{180} \right) + b_6 \cdot LST_{\mathtt{MODIS}}(\mathbf{s}_0, t_0); \qquad \Delta t = 1 \text{ day} \end{aligned} \tag{11.1.1}$$

where *DEM* is the elevation map, *LAT* is the map showing distance from the equator, *LON* is the longitude, *DISTC* is the distance from the coast line, $\cos(t)$ is a generic function to account for seasonal variation of values, ϕ is the phase angle[2], and $LST_{\mathtt{MODIS}}$ is the land surface temperature estimated by the MODIS satellite.

[1] http://meteo.hr

[2] A time delay from the coldest day.

DEM, *LAT*, *DISTC* are temporally-constant predictors; $LST_{\mathtt{MODIS}}$ are temporally variable predictors i.e. time-series of remotely sensed images. More details about how were the MODIS LST images were obtained can be found in section 4.2.

The residuals from such a spatio-temporal regression model can also be analyzed for (spatio-temporal) auto-correlation and used to run 3D interpolation (see §2.5). Once we have fitted the space-time variogram, we can then run **spatio-temporal regression-kriging**[3] to estimate the values at 3D locations. In practice, we only wish to produce maps for a given time interval (t_0=constant), i.e. to produce 2D-slices of values in time (see Fig. 2.10; §2.5). For a more gentle introduction to spatio-temporal interpolation see some classical papers by e.g. Huerta et al. (2004), Jost et al. (2005) and Pebesma et al. (2007). A geostatistical exercises with stochastic simulation of rainfall data using remote sensing images can be followed in Teo and Grimes (2007). Schuurmans et al. (2007) propose an automated framework, similar to the one described in this chapter, for prediction of the rainfall fields using spatio-temporal data and Kriging with External Drift.

11.2 Data download and preprocessing

The time-series of remotely sensed images, data from meteorological stations and auxiliary maps have been previously prepared by author (§4.2). First, open a new R session and change the working directory to where all the data sets are located (e.g. `C:/croatia/`). Open the R script (`HRtemp.R`) and load the necessary libraries:

```
> library(maptools)
> library(gstat)
> library(rgdal)
> library(lattice)
```

The ground measurements of temperatures can be obtained from the book's homepage. There are two zip files: (1) `HRtemp2006.zip` — contains a digital elevation model, distance from the coast line and temperature measurements from the meteorological stations, (2) `LST2006HR.zip` — contains 92 geotiff's of reprojected MODIS LST images. We need to download and unzip them locally:

```
# Download MODIS LST images:
> download.file("http://spatial-analyst.net/book/system/files/LST2006HR.zip",
+      destfile=paste(getwd(), "LST2006HR.zip", sep="/"))

  trying URL 'http://spatial-analyst.net/book/system/files/LST2006HR.zip'
  Content type 'application/zip' length 20655622 bytes (19.7 Mb)
  opened URL
  downloaded 19.7 Mb


> unzip(zipfile="LST2006HR.zip", exdir=getwd())
> unlink("LST2006HR.zip")
# Download auxiliary maps and LST measurements:
> download.file("http://spatial-analyst.net/book/system/files/HRtemp2006.zip",
+      destfile=paste(getwd(), "HRtemp2006.zip", sep="/"))

  trying URL 'http://spatial-analyst.net/book/system/files/HRtemp2006.zip'
  Content type 'application/zip' length 682970 bytes (666 Kb)
  opened URL
  downloaded 666 Kb


> unzip(zipfile="HRtemp2006.zip", exdir=getwd())
```

and you will find the following data sets:

- `HRdem.asc` — Digital Elevation Model projected in the UTM (zone 33) system;
- `HRdsea.asc` — buffer to coast line in km;

[3] This is called a "*Space-time metric model*", because time dimension is modeled as space dimension (Huerta et al., 2004).

- IDSTA.shp — location of meteorological stations in geographical coordinates;
- HRtemp2006.txt — mean daily temperatures measured at 123 locations for 365 days (the whole year 2006);
- LST2006_**_**.LST_Day_1km.tif — 8-day estimates of daily LST;
- LST2006_**_**.LST_Night_1km.tif — 8-day estimates of night time LST;

We start by importing the temperatures from the HRtemp2006.txt file:

```
> HRtemp2006 <- read.delim("HRtemp2006.txt")
> str(HRtemp2006) # Mean daily temperatures;

  'data.frame':   44895 obs. of  3 variables:
   $ IDT_AK: Factor w/ 123 levels "GL001","GL002",..: 1 1 1 1 1 1 1 1 1 1 ...
   $ DATE  : Factor w/ 365 levels "2006-1-1","2006-1-10",..: 1 12 23 26 27 28 29...
   $ MDTEMP: num  1.6 0.7 1.5 0.3 -0.1 1 0.3 -1.9 -5.4 -3.6 ...
```

This shows that there are 44,895 measurements of mean daily temperature in total. These are only daily mean values, meteorologists typically work with even finer support size (e.g. hourly values). We need to format the imported dates, from string format to the date-time class and then to a numerical format, so that we can use them in further quantitative analysis:

```
> HRtemp2006$cday <- floor(unclass(as.POSIXct(HRtemp2006$DATE))/86400)
```

where POSIXct is the date-time class. Now the days are expressed as cumulative days since 1970-01-01, i.e. as numeric values. For example, a date 2006-01-30, corresponds to:

```
> floor(unclass(as.POSIXct("2006-01-30"))/86400)[[1]]

  [1] 13177


# inverse transformation:
# as.POSIXct(13177*86400, origin="1970-01-01")
```

Next, we can import the latitude/longitude coordinates of the 152 meteorological stations and convert them to the target coordinate system[4]:

```
> IDSTA <- readShapePoints("IDSTA.shp", proj4string=CRS("+proj=longlat +datum=WGS84"))
> IDSTA.utm <- spTransform(IDSTA, CRS("+proj=utm +zone=33 +ellps=WGS84
+         +datum=WGS84 +units=m +no_defs"))
> locs <- as.data.frame(IDSTA.utm)
> names(locs) <- c("IDT_AK", "X", "Y")
> str(locs)

  'data.frame':   152 obs. of  3 variables:
   $ IDT_AK: Factor w/ 152 levels "GL001","GL002",..: 1 2 3 4 5 6 7 8 9 10 ...
   $ X     : num  670760 643073 673778 752344 767729 ...
   $ Y     : num  5083464 5086417 5052001 4726567 4717878 ...


# stations without measurements:
dif.IDSTA <- merge(locs["IDT_AK"], data.frame(IDT_AK=levels(HRtemp2006$IDT_AK),
+     sel=rep(1, length(levels(HRtemp2006$IDT_AK)))), by.x="IDT_AK", all.x=TRUE)
```

and then the raster maps:

[4] Note that we import coordinates of the stations separately because the stations are fixed, so that we only need to know the ID of a station. Otherwise would be inefficient to attach coordinates to each space-time measurements.

```
# Import grids:
> grids <- readGDAL("HRdem.asc")
> names(grids@data)[1] <- "HRdem"
> grids$HRdsea <- readGDAL("HRdsea.asc")$band1
> proj4string(grids) <- IDSTA.utm@proj4string
# create dummy grids (Lat/Lon):
> grids.ll <- spTransform(grids[1], CRS("+proj=longlat +datum=WGS84"))
> grids$Lat <- grids.ll@coords[,2]
> grids$Lon <- grids.ll@coords[,1]
> str(grids@data)

 'data.frame':   238630 obs. of  4 variables:
  $ HRdem : int  1599 1426 1440 1764 1917 1912 1707 1550 1518 1516 ...
  $ HRdsea: num  93 89.6 89.8 93.6 95 ...
  $ Lat   : num  46.5 46.5 46.5 46.5 46.5 ...
  $ Lon   : num  13.2 13.2 13.2 13.2 13.2 ...
```

We will import both nighttime and daytime values, and then derive the average daily values of LST as an average between the two[5]. From the data set description for the MOD11A2 MODIS product we can notice that the original values are in degree Kelvin, which we need to transform to degree Celsius; all values <7500 are NA values; the scaling ratio is 0.02. So in summary, we run:

```
> LST.listday <- dir(pattern=glob2rx("LST2006_**_**.LST_Day_1km.tif"))
> LST.listnight <- dir(pattern=glob2rx("LST2006_**_**.LST_Night_1km.tif"))
> for(i in 1:length(LST.listday)){
>   LSTname <- strsplit(LST.listday[i], ".LST_")[[1]][1]
>   tmp1 <- readGDAL(LST.listday[i])$band1
>   tmp2 <- readGDAL(LST.listnight[i])$band1
# convert to Celsius:
>   tmp1 <- ifelse(tmp1<=7500, NA, tmp1*0.02-273.15)
>   tmp2 <- ifelse(tmp2<=7500, NA, tmp2*0.02-273.15)
>   grids@data[,LSTname] <- (tmp1+tmp2)/2
# simple average -- this ignores that day/night ratio is variable!
> }
```

If you visualize some LST images for various dates (use SAGA GIS), you will notice that there are many NA pixels (especially for winter months). In average, there will always be 10–30% missing pixels in the MODIS images, which is a serious limitation. Also notice that the images can be fairly noisy and with many strange patterns — line or polygon features, jumps in values — which are obviously artifacts. You need to know that these images have been created by patching together images from a period of ±4 days, this way the amount of clouds can be reduced to minimum. Depending on the local meteorological conditions, amount of clouds in an 8–day LST image can still be high (up to 100%). On the other hand, the advantage of using MODIS LST images is that they account for small differences in temperature that are due to different land cover, moisture content, and human-connected activities. Such features cannot be modeled with the constant physical parameters such as elevation, latitude, longitude and distance from the coast line.

11.3 Regression modeling

We first need to prepare the regression matrix by overlaying the meteorological stations and imported grids:

```
> IDSTA.ov <- overlay(grids, IDSTA.utm)
> locs <- cbind(IDSTA.ov@data[c("HRdem", "HRdsea", "Lat", "Lon")], locs)
> str(locs)

 'data.frame':   152 obs. of  7 variables:
  $ HRdem : int  161 134 202 31 205 563 80 96 116 228 ...
  $ HRdsea: num  198.5 181.7 192.9 0 1.5 ...
  $ Lat   : num  45.9 45.9 45.6 42.7 42.6 ...
```

[5] Here we will ignore that the length of daytime and nighttime differ for different days.

```
 $ Lon   : num  17.2 16.8 17.2 18.1 18.3 ...
 $ IDT_AK: Factor w/ 152 levels "GL001","GL002",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ X     : num  670760 643073 673778 752344 767729 ...
 $ Y     : num  5083464 5086417 5052001 4726567 4717878 ...
```

which is the initial regression matrix with temporally constant predictors. Temperatures measured at the meteorological stations are missing. We also need to copy the coordinates of stations to the original table. The two tables can be merged by using:

```
> HRtemp2006locs <- merge(HRtemp2006, locs, by.x="IDT_AK")
```

which will basically copy values of constant predictors for all dates. Next we also need to copy the values of MODIS estimated LST at meteorological stations. This is not as trivial because MODIS LST images are not available for all dates. They also need to be sorted in a way that is suitable for analysis. First, let us see which days of the year are available as images:

```
> LSTdate <- rep(NA, length(LST.listday))
> for(i in 1:length(LST.listday)){
>   LSTdate[i] <- gsub("_", "-", strsplit(strsplit(LST.listday[i],
+              ".LST_")[[1]][1], "LST")[[1]][2])
> }
# cumulative days since 2006-01-01:
> LSTcdate <- round((unclass(as.POSIXct(LSTdate)) -
+           unclass(as.POSIXct("2006-01-01")))/86400, 0)
# add one extra day:
> LSTcdate <- c(LSTcdate, 365)
> LSTcdate[1:5]

[1]  0  8 16 24 32
```

next, we need to sort the values in a data frame of the same size as the `HRtemp2006locs`:

```
# create an empty data frame:
> MODIStemp <- expand.grid(IDT_AK=levels(HRtemp2006$IDT_AK),
+      DATE=levels(HRtemp2006$DATE), stringsAsFactors=TRUE)
> MODIStemp$MODIS.LST <- rep(NA, length(MODIStemp[1]))
# copy MODIS LST values per date:
> MODIStemp$MODIS.LST[1:(123*4)] <- rep(IDSTA.ov@data[!is.na(dif.IDSTA$sel),
+      strsplit(LST.listday[i], ".LST_")[[1]][1]], 4)
# all other days:
> for(i in 2:length(LST.listday)){
>   LSTname <- strsplit(LST.listday[i], ".LST_")[[1]][1]
# position/date:
>   d.days <- round((LSTcdate[i+1]-LSTcdate[i-1])/2, 0)
>   d.begin <- round((LSTcdate[i]-d.days/2)*123+1, 0)
>   d.end <- round((LSTcdate[i]+d.days/2)*123+1, 0)
# copy the values:
>   MODIStemp$MODIS.LST[d.begin:d.end] <- rep(IDSTA.ov@data[!is.na(dif.IDSTA$sel),
+      LSTname], d.days)
> }
# the last days:
> MODIStemp$MODIS.LST[(d.end+1):length(MODIStemp$MODIS.LST)] <-
+      rep(IDSTA.ov@data[!is.na(dif.IDSTA$sel),
+      strsplit(LST.listday[i], ".LST_")[[1]][1]], 2)
```

so that we can finally copy all MODIS LST values:

```
> HRtemp2006locs$MODIS.LST <- MODIStemp$MODIS.LST[order(MODIStemp$IDT_AK)]
> str(HRtemp2006locs)
```

```
'data.frame':   44895 obs. of  11 variables:
 $ IDT_AK   : Factor w/ 123 levels "GL001","GL002",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ DATE     : Factor w/ 365 levels "2006-1-1","2006-1-10",..: 1 12 23 26 27 28...
 $ MDTEMP   : num  1.6 0.7 1.5 0.3 -0.1 1 0.3 -1.9 -5.4 -3.6 ...
 $ cday     : num  13148 13149 13150 13151 13152 ...
 $ HRdem    : int  161 161 161 161 161 161 161 161 161 161 ...
 $ HRdsea   : num  198 198 198 198 198 ...
 $ Lat      : num  45.9 45.9 45.9 45.9 45.9 ...
 $ Lon      : num  17.2 17.2 17.2 17.2 17.2 ...
 $ X        : num  670760 670760 670760 670760 670760 ...
 $ Y        : num  5083464 5083464 5083464 5083464 5083464 ...
 $ MODIS.LST: num  -1.17 -1.17 -1.17 -1.17 -5.92 ...
```

note that the values of MODIS LST are now available as a single column in the original regression matrix. Before we can create a 3D point data set, it is a useful thing to scale *t*-coordinate, so it has approximately similar range[6] as the XY coordinates:

```
# scale the values:
> tscale <- (((grids@bbox[1,"max"]-grids@bbox[1,"min"])+(grids@bbox[2,"max"]
+    -grids@bbox[2,"min"]))/2)/(max(HRtemp2006locs$cday)-min(HRtemp2006locs$cday))
> HRtemp2006locs$cdays <- tscale * HRtemp2006locs$cday
# 3D points:
> coordinates(HRtemp2006locs) <- c("X", "Y", "cdays")
> proj4string(HRtemp2006locs) <- CRS(proj4string(grids))
# copy values:
> HRtemp2006locs$cdays <- HRtemp2006locs@coords[,"cdays"]
```

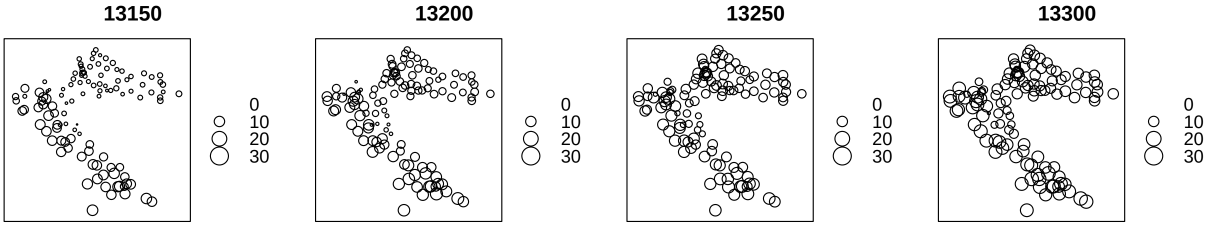


Fig. 11.2: Spatial pattern of measured mean-daily temperatures for 2006-01-02 (13150), 2006-02-21 (13200), 2006-04-12 (13250), 2006-06-01 (13300).

Now that we have attached coordinates to the temperature measurements and created a 3D point object, we can visualize them by using the `bubble` method available in the `sp` package (Fig. 11.2):

```
> bubble(subset(HRtemp2006locs, HRtemp2006locs$cday==13150&!is.na(HRtemp2006locs$MDTEMP),
+     select="MDTEMP"), fill=F, col="black", maxsize=2,
+     key.entries=c(0,10,20,30), main="13150")
```

We can also make subset of the data and observe how the values change through time at a specific station:

```
# pick three meteorological stations:
> GL001 <- subset(HRtemp2006locs@data, IDT_AK=="GL001", select=c("MDTEMP", "cday"))
> KL003 <- subset(HRtemp2006locs@data, IDT_AK=="KL003", select=c("MDTEMP", "cday"))
> KL094 <- subset(HRtemp2006locs@data, IDT_AK=="KL094", select=c("MDTEMP", "cday"))
> par(mfrow=c(1,3))
> scatter.smooth(GL001$cday, GL001$MDTEMP, xlab="Cumulative days",
+     ylab="Mean daily temperature (\260C)", ylim=c(-12,28), col="grey")
> scatter.smooth(KL003$cday, KL003$MDTEMP, xlab="Cumulative days",
+     ylab="Mean daily temperature (\260C)", ylim=c(-12,28), col="grey")
> scatter.smooth(KL094$cday, KL094$MDTEMP, xlab="Cumulative days",
+     ylab="Mean daily temperature (\260C)", ylim=c(-12,28), col="grey")
```

[6] This is not really a requirement for the analysis, but it makes visualization of 3D space and variograms easier.

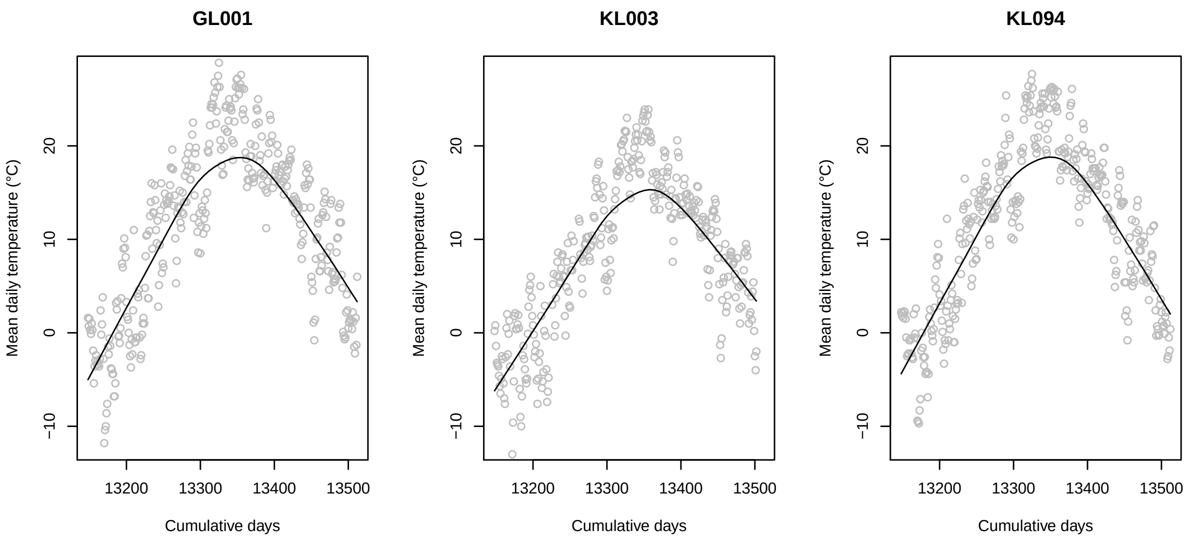


Fig. 11.3: Temporal dynamics of mean-daily temperatures at selected meteorological stations.

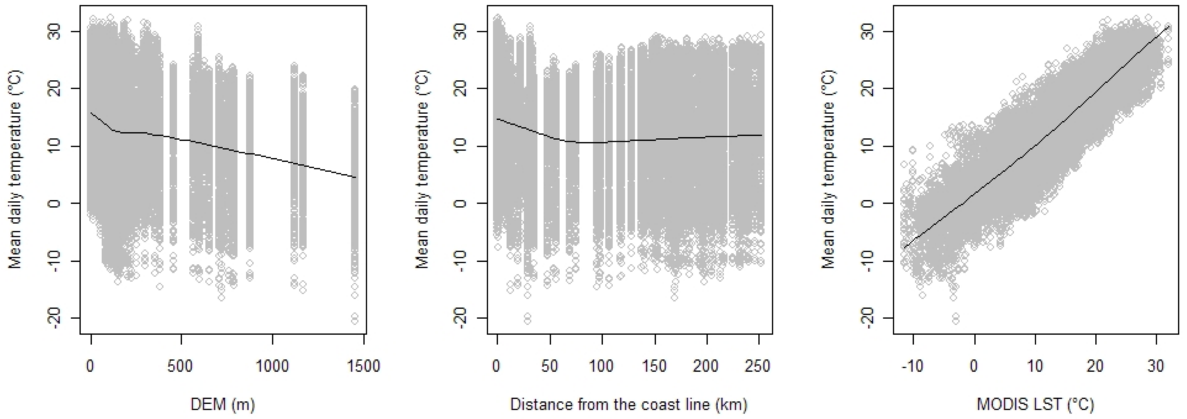


Fig. 11.4: Scatter plots showing the general relationship between daily temperature (MDTEMP), elevation, distance from the coast line and MODIS LST images.

which shows that the values change more systematically in time domain, than in the space domain. It will also be interesting to observe individual relationships between MDTEMP, HRdem, HRdsea and MODIS.LST (Fig. 11.4). This shows, as expected, that the temperature drops with elevation, and with distance from the coast line[7]. Note also that MODIS LST seem to be a fairly accurate (unbiased) predictor of the actual measured temperature: the relationship is highly linear and the scatter around the regression line is constant. Nevertheless, Fig. 11.4 also shows that there is a significant scatter around the regression lines, which means that also the residuals will be significant.

We can now fit a linear model using the Eq.(11.1.1):

```
> theta <- min(HRtemp2006locs$cday)
> lm.HRtemp <- lm(MDTEMP ~ HRdem+HRdsea+Lat+Lon+cos((cday-theta)*pi/180)+MODIS.LST,
+       data=HRtemp2006locs)
```

[7] Based on this data, it seems that the influence of the sea climate is up to the distance of about 80 km.

```
> summary(lm.HRtemp)$adj.r.squared

  [1] 0.8423278


# plot(lm.HRtemp)
```

which shows that all predictors are highly significant; the model explains 84% of variability in the MDTEMP values; the derived residuals are normally distributed around the 0 value, with only 3 influential values (outliers).

11.4 Space-time variogram estimation

Now we can also try to fit the 3D variogram for residuals. Note that gstat supports 3D interpolation, but it does not actually support interactive fitting of 3D variograms. In fact, to my knowledge, interactive modeling of space-time autocorrelation is still very limited in R, but also in commercial packages such as Isatis or ArcGIS.

We can start with plotting the points in a 3D space, which is possible by using the cloud method available in the lattice package. This will produce a plot shown in Fig. 11.5, which gives us a good idea about the sampling design specific to this data set. The data set consist of large number of space-time points. Furthermore, for the purpose of this exercise, we can first randomly subset the original data set to 10% of its size to speed up plotting and fitting of the variograms:

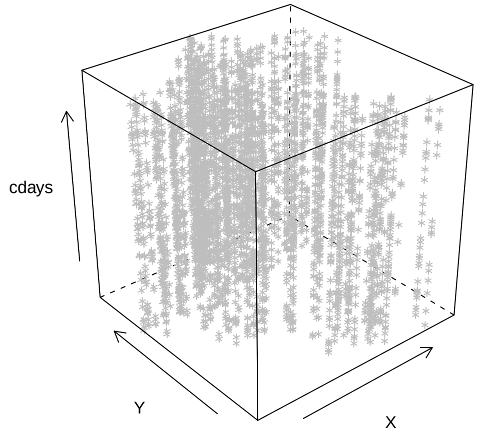


Fig. 11.5: Cloud plot showing location of meteorological stations in the space-time cube.

```
# remove missing values:
> HRtemp2006.f <- HRtemp2006locs[-lm.HRtemp$na.action,]
# copy the residuals:
> HRtemp2006.f$rMDTEMP2006 <- lm.HRtemp$residuals
# sub-sample:
> HRtemp2006.sel <- HRtemp2006.f[
+    runif(length(HRtemp2006.sel$rMDTEMP2006))<0.1,]
# str(HRtemp2006.sel)
# plot the 3D points:
> coords <- as.data.frame(HRtemp2006.sel@coords)
> cloud(cdays ~ X*Y, coords, col="grey")
```

In addition, assuming that the variogram will be anisotropic, a good idea is to plot a variogram map (Fig. 11.6, left):

```
> varmap.plt <- plot(variogram(rMDTEMP2006 ~ 1, HRtemp2006.sel, map=TRUE,
+    cutoff=sqrt(areaSpatialGrid(grids))/2, width=30*grids@grid@cellsize[1]),
+    col.regions=grey(rev(seq(0,1,0.025))))
> rv.MDTEMP2006 <- variogram(rMDTEMP2006 ~ 1, HRtemp2006.sel, alpha=c(45,135))
> rvgm.MDTEMP2006 <- fit.variogram(rv.MDTEMP2006,
+    vgm(psill=var(HRtemp2006.sel$rMDTEMP2006),
+    "Exp", sqrt(areaSpatialGrid(grids))/4, nugget=0, anis=c(p=45,s=0.5)))
> vgm.plt <- plot(rv.MDTEMP2006, rvgm.MDTEMP2006, plot.nu=FALSE, cex=2, pch="+",
+    col="black")
> print(varmap.plt, split=c(1,1,2,1), more=T)
> print(vgm.plt, split=c(2,1,2,1), more=F)
> rvgm.MDTEMP2006
```

```
  model    psill    range ang1 anis1
1   Nug 6.094418     0.00    0   1.0
2   Exp 4.639634 15392.17   45   0.5
```

which shows that a strong anisotropy exists (the azimuth of the longer axis is somewhere at 135°), although it is not so distinct. Recall that the main mountain chain on the Balkans (see Fig. 11.1) spreads approximately in the same direction.

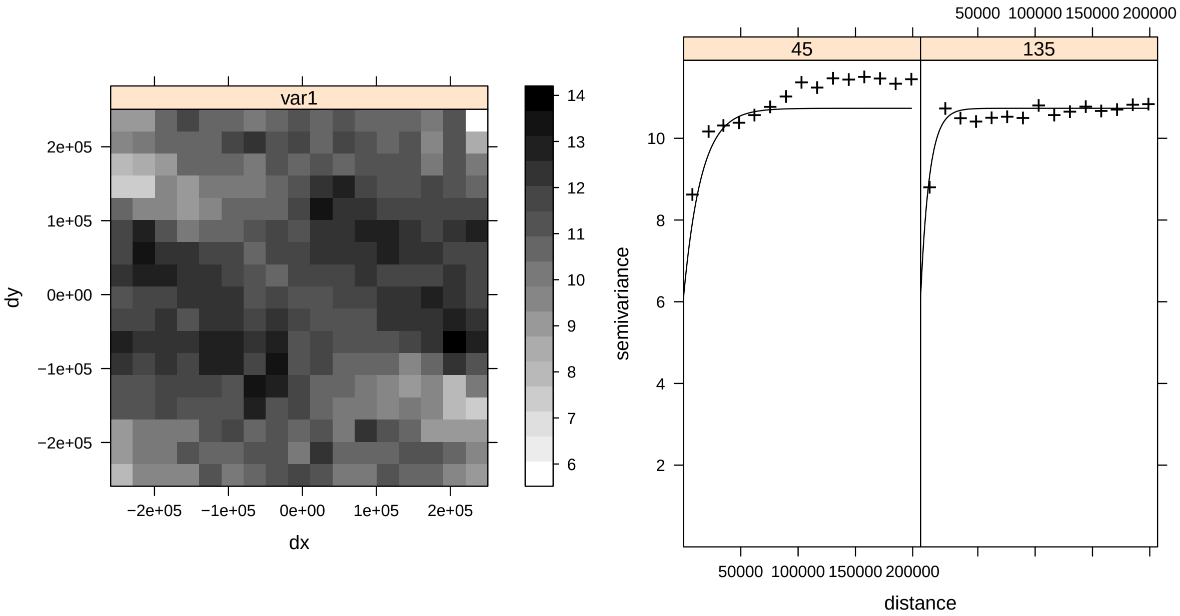


Fig. 11.6: Variogram map (left) and fitted anisotropic variogram model (right) for `MDTEMP` residuals.

The plotted 3D variograms look the same as in Fig. 9.1, although we know that this is not a 2D but a 3D data set. Visual exploration of space-time (3D or 4D) variogram models in R is at the moment limited to 2D spaces, although the situation might change with the new space-time (stpp[8]) package. Note also from Fig. 11.3 that is obvious that nugget variation in time direction will be much higher greater in the space domain.

11.5 Spatio-temporal interpolation

Making prediction in the space-time domain (3D) is not as easy as making 2D predictions (the previous exercises). It will take some time to prepare the prediction locations, get the values of all constant and temporal predictors and then visualize the final results. As mentioned previously, we do not intend to make predictions for the whole space-time cube, but only for fixed time-intervals, i.e. time slices (see also Fig. 2.10). Detailed steps are now explained down-below.

11.5.1 A single 3D location

We can start by defining the geostatistical model (for residuals):

```
> g.MDTEMP <- gstat(id=c("rMDTEMP2006"), formula=rMDTEMP2006 ~ 1,
+       data=HRtemp2006.f, nmax=40, model=rvgm.MDTEMP2006)
```

We can now first test the algorithm by using a single 3D point. For example, we ask ourselves what is the mean daily temperature for 1st of August 2006, at location X=575671 E, Y=5083528 N? To prepare the new point location we can use:

[8] http://stpp.r-forge.r-project.org

```
> newloc.t <- "2006-08-01"
> newloc.ct <- floor(unclass(as.POSIXct(newloc.t))/86400)[[1]]
> newloc.x <- 575671
> newloc.y <- 5083528
> newloc.xyt <- as.data.frame(matrix(c(newloc.x, newloc.y, tscale*newloc.ct),
+       ncol=3, dimnames = list(c("p1"), c("X","Y","cdays"))))
> coordinates(newloc.xyt) <- c("X","Y","cdays")
> proj4string(newloc.xyt) <- CRS(proj4string(grids))
# 3D prediction location:
> newloc.xyt
```

```
  SpatialPoints:
            X       Y     cdays
  [1,] 575671 5083528 17929560
  Coordinate Reference System (CRS)
  arguments: +proj=utm +zone=33
  +ellps=WGS84 +datum=WGS84 +units=m
  +no_defs +towgs84=0,0,0
```

Next, we need to get the values of the auxiliary predictors at this location. We can do this by overlaying the new point with the imported gridded maps:

```
> newloc.xy <- as.data.frame(newloc.xyt)
> coordinates(newloc.xy) <- c("X","Y")
> proj4string(newloc.xy) <- CRS(proj4string(grids))
> ov.newloc.xy <- overlay(grids, newloc.xy)
# add the "time"-location:
> ov.newloc.xy$cday <- newloc.ct
> str(ov.newloc.xy@data)
```

```
  'data.frame':   1 obs. of  52 variables:
   $ HRdem        : int 805
   $ HRdsea       : num 165
   $ Lat          : num 45.9
   $ Lon          : num 16
   $ LST2006_01_01: num NA
   $ LST2006_01_09: num -5.23
  ...
   $ LST2006_12_27: num 0.37
   $ cday         : num 13360
```

Notice that the value of MODIS.LST is missing. The overlay operation works only in 2D, hence it does not know what the value of MODIS.LST is for the given date. We need to first estimate what would be the closest MODIS image for 2006-08-01, and then copy those values:

```
> cdate <- round((unclass(as.POSIXct(newloc.t)) -
+          unclass(as.POSIXct("2006-01-01")))/86400, 0)[1]
> cdate
```

```
  [1] 212
```

```
> LSTname <- strsplit(LST.listday[which.min(abs(cdate-LSTcdate))], ".LST_")[[1]][1]
> LSTname
```

```
  [1] "LST2006_07_28"
```

```
> ov.newloc.xy$MODIS.LST <- ov.newloc.xy@data[,LSTname]
> ov.newloc.xy$MODIS.LST
```

```
  [1] 21.29
```

which shows that the '*closest*' MODIS image is from `2006-07-28` and MODIS estimated temperature for that date is `21.29`°C. Now the new location is complete, we can make predict the mean daily temperature at this location, and using the model estimated in the previous section:

```
> locMDTEMP.reg <- predict(lm.HRtemp, ov.newloc.xy)
# the trend part:
> locMDTEMP.reg
```

```
   35497
18.76291
```

```
# OK of residuals;
> locMDTEMP <- predict.gstat(g.MDTEMP, newloc.xyt, beta=1, BLUE=FALSE)
```

```
[using ordinary kriging]
```

```
> locMDTEMP.reg + locMDTEMP$rMDTEMP2006.pred
```

```
   35497
18.98239
```

which is somewhat lower than we expected for this time of year. Just to check how close the result is to the temperature actually measured at the closest location:

```
# locate the closest measured temperature:
> closest.pnt <- which.min(dist(rbind(newloc.xy@coords,
+      IDSTA.utm@coords))[1:length(IDSTA.utm@coords[,1])])
> closest.IDSTA <- as.character(IDSTA.utm$IDSTA[closest.pnt])
> closest.IDSTA
```

```
[1] "GL023"
```

```
> subset(HRtemp2006locs, HRtemp2006locs$cday==newloc.ct&
+   HRtemp2006locs$IDT_AK==closest.IDSTA, select="MDTEMP")
```

```
                       coordinates MDTEMP
8243 (575118, 5084260, 17929600)   17.2
```

which shows that the predicted temperature is somewhat higher than the one measured at the same day, at the closest station. The values are higher mainly because the LST image shows higher values for that period. Surprisingly, the residuals are positive, even though measured values are above predicted and even though the prediction point is fairly close to the measurement location (distance is only 913 m). Take into account that station `GL023` is at the top of a mountain, so that it is realistic to always expect a lower temperature even at so short distance.

11.5.2 Time-slices

We can now generate predictions for a list of new locations. Because there is still no support for 3D grids in R, we can instead make 3D regular points and then predict at those locations. In fact, we will define only slices of the space-time cube for which we will make predictions. Another important issue is that we will put operations in a loop to speed up the processing. Imagine, there are 365 days, and if we would want to interpolate the values for `MDTEMP` — this would take a lot of scripting. To avoid this problem, we create a R loop that will interpolate as many maps as we like. For the purpose of this exercise, we derive only time-slices for which we also have MODIS images available:

```
# available MODIS images:
> slices <- LSTdate
# new locations:
> grids.xy <- as(grids[c("HRdem", "HRdsea", "Lat", "Lon")], "SpatialPixelsDataFrame")
> for(i in 1:length(slices)) {
>   newlocs.xyt <- grids.xy@data
>   newlocs.xyt$X <- grids.xy@coords[,"x"]
>   newlocs.xyt$Y <- grids.xy@coords[,"y"]
>   slice <- floor(unclass(as.POSIXct(slices[i]))/86400)[[1]]
>   newlocs.xyt$cday <- rep(slice, length(newlocs.xyt[1]))
>   newlocs.xyt$cdays <- tscale * newlocs.xyt$cday
>   LSTname <- strsplit(LST.listday[i], ".LST_")[[1]][1]
>   newlocs.xyt$MODIS.LST <- grids@data[grids.xy@grid.index,LSTname]
>   coordinates(newlocs.xyt) <- c("X","Y","cdays")
>   proj4string(newlocs.xyt) <- CRS(proj4string(grids))
>   MDTEMP.ok <- predict.gstat(g.MDTEMP, newlocs.xyt, beta=1, BLUE=FALSE)
>   MDTEMP.reg <- predict(lm.HRtemp, newlocs.xyt)
>   grids@data[,paste(LSTname,".RK",sep="")] <- MDTEMP.reg+MDTEMP.ok$rMDTEMP2006.pred
> }
```

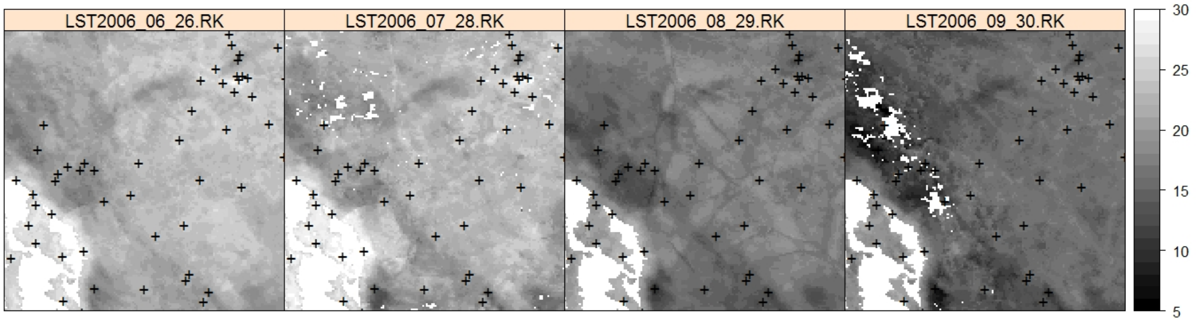


Fig. 11.7: Mean daily temperatures (°C) predicted using spatio-temporal regression-kriging (see titles for dates). Missing pixels are due to clouds in the MODIS LST images.

This operation is relatively time and memory consuming[9] so try also to limit the number of slices to <20. After the process is finished, a useful thing to do is to visualize the predicted maps together with the measured point values by using e.g. (Fig. 11.7):

```
> pr.list <- c("LST2006_06_26.RK", "LST2006_07_28.RK", "LST2006_08_29.RK", "LST2006_09_30.RK")
> spplot(grids[pr.list], col.regions=grey(seq(0,1,0.025)), at=seq(5,30,1),
+      xlim=c(450000,600000), ylim=c(4950000,5100000), sp.layout=list("sp.points",
+      IDSTA.utm, pch="+", cex=1.5, col="black"))
```

You will soon discover that these predictions are mainly controlled by the MODIS LST images (also clearly visible from Fig. 11.4); a large portion of NA areas is visible in the output maps. These could be fixed by iteratively filtering[10] the original MODIS images before these are used as predictors.

11.5.3 Export to KML: dynamic maps

We have produced a series of maps of daily temperatures. We want now to export these maps to Google Earth and visualize them as a time series. Note that Google Earth support spatio-temporal data that can be browsed by using a *Timeline* bar (Fig. 11.8). In principle, transformation of 2D data to spatio-temporal data is rather

[9] A way to speed up the processing would be to limit the search radius (see p.94), but this would also lead to artifacts in the maps.
[10] See for example Addink and Stein (1999).

simple — we only need to add a field called <TimeSpan> and then define the begin and end time to which a map refers to. If Google Earth sees that this field has been defined, it will automatically browse it with a time slider. We first need to resample all maps to geographic grid (see also section 5.6.2 for more details):

```
# resampling of maps to geographic coordinates:
> for(i in seq(5,35,2)) {
>   LSTname <- strsplit(LST.listday[i], ".LST_")[[1]][1]
>   write.asciigrid(grids[paste(LSTname, ".RK", sep="")],
+      paste(LSTname, "_RK.asc", sep=""), na.value=-999)
>   rsaga.esri.to.sgrd(in.grids=paste(LSTname, "_RK.asc", sep=""),
+      out.sgrd=paste(LSTname, "_RK.sgrd", sep=""), in.path=getwd())
 # bilinear resample:
>   rsaga.geoprocessor(lib="pj_proj4", 2, param=list(SOURCE_PROJ=paste('"',
+      proj4string(grids), '"', sep=""), TARGET_PROJ="\"+proj=longlat
+      +datum=WGS84\"", SOURCE=paste(LSTname, "_RK.sgrd", sep=""),
+      TARGET=paste(LSTname, "_RK_ll.sgrd", sep=""), TARGET_TYPE=0,
+      INTERPOLATION=1))
 # write back to ASCII:
>   rsaga.sgrd.to.esri(in.sgrds=paste(LSTname, "_RK_ll.sgrd", sep=""),
+      out.grids=paste(LSTname, "_RK.asc", sep=""), prec=1, out.path=getwd())
> }
```

and then read the maps back into R:

```
> MDTEMP.list <- dir(pattern=glob2rx("LST2006_**_**_RK.asc"))
> grids.geo <- readGDAL(MDTEMP.list[1])
> proj4string(grids.geo) <- CRS("+proj=longlat +datum=WGS84")
> names(grids.geo) <- strsplit(MDTEMP.list[1], ".asc")[[1]][1]
> for(i in 2:length(MDTEMP.list)){
>    LSTname <- strsplit(MDTEMP.list[i], ".asc")[[1]][1]
>    grids.geo@data[,LSTname] <- readGDAL(MDTEMP.list[i])$band1
> }
```

We can now prepare a GE_SpatialGrid object using the original sp SpatialGridDataFrame:

```
> grids.kml <- GE_SpatialGrid(grids.geo)
```

We want to use the same legend for all time-slices and add it as a screen overlay, which means that we need to determine suitable limits for the legend, e.g. 98% range of values:

```
> MDTEMPxlim <- quantile(HRtemp2006.f$MDTEMP, probs=c(0.01,0.99))
> MDTEMPxlim

   1%  99%
 -6.0 28.4
```

We export all maps as PNGs using a fixed legend:

```
# export all maps as PNG:
> for(i in 1:length(MDTEMP.list)) {
>   LSTname <- strsplit(MDTEMP.list[i], ".asc")[[1]][1]
>   png(file=paste(LSTname, ".png", sep=""), width=grids.kml$width,
+     height=grids.kml$height, bg="transparent")
>   par(mar=c(0,0,0,0), xaxs="i", yaxs="i")
>   image(as.image.SpatialGridDataFrame(grids.geo[LSTname]), col=bpy.colors(),
+     zlim=(MDTEMPxlim), xlim=grids.kml$xlim, ylim=grids.kml$ylim)
>   dev.off()
> }
```

To export the legend PNG, we use:

```
# prepare the legend:
> png(file="legend.png", width=grids.kml$width/4, height=grids.kml$height/3, bg="white")
> par(mar=c(0,0.1,0,0.1), yaxs="i")
> image(grids, names(grids[5]), col="white")
> source("http://spatial-analyst.net/scripts/legend_image.R")
> legend_image(c(grids@bbox[1,1],
+     grids@bbox[1,2]-(grids@bbox[1,2]-grids@bbox[1,1])/4), c(grids@bbox[2,1],
+     grids@bbox[2,2]), seq(MDTEMPxlim[[1]],MDTEMPxlim[[2]],1), vertical=TRUE,
+     col=bpy.colors(round(MDTEMPxlim[[2]]-MDTEMPxlim[[1]]/1,0)), offset.leg=.2,
+     cex=1.5)
> dev.off()
```

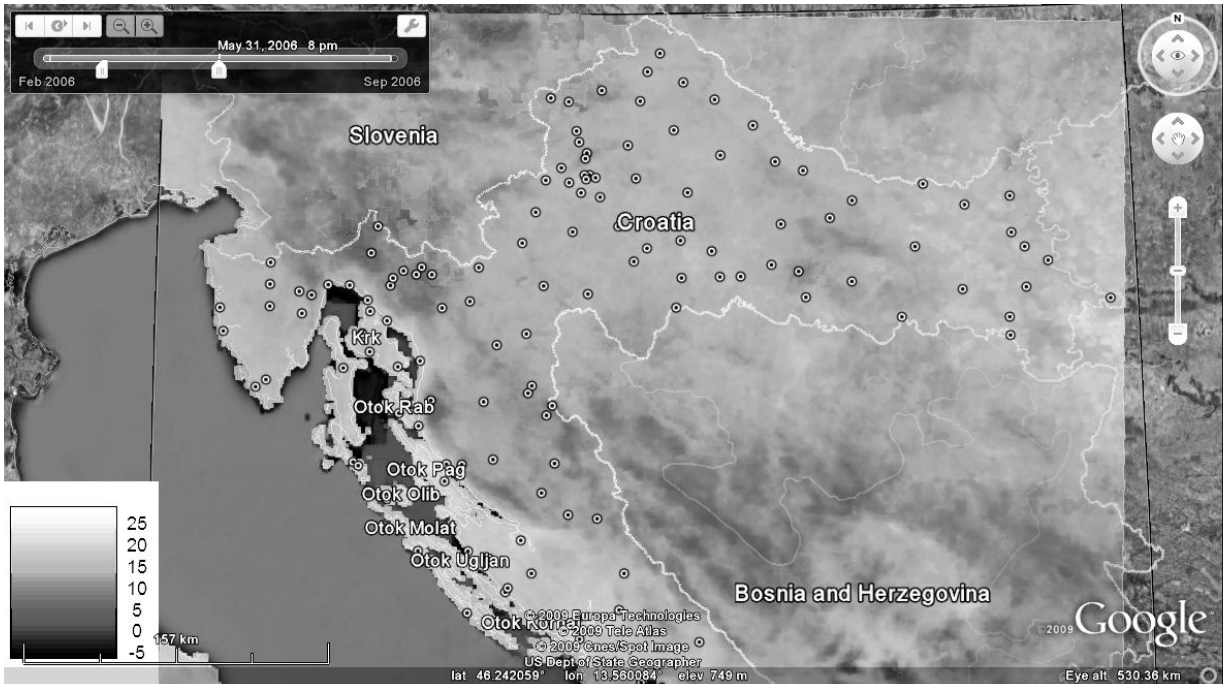


Fig. 11.8: Interpolation of temperatures visualized in Google Earth as time-series of maps. Unlike many standard GIS, Google Earth allows visual exploration of spatio-temporal data. Note from the time-bar that you can also edit the temporal support and produce *smoothed* images in time dimension.

The package maptools does not support export of time-series of maps to Google Earth. This means that we need to write the KML file ourselves. This is more simple than you anticipate, because we can again use loops (see below). In principle, we only need to respect some common headers and structure used in KML, everything else we can easily control. Note also that the KML file structure is easy to read and can be easily edited, even manually:

```
> filename <- file("MDTEMP2006.kml")
> write('<?xml version="1.0" encoding="UTF-8"?>', filename)
> write('<kml xmlns="http://earth.google.com/kml/2.2">', filename, append=TRUE)
> write('<Folder>', filename, append=TRUE)
> write('        <name>Mean Daily TEMP</name>', filename, append=TRUE)
> write('        <open>1</open>', filename, append=TRUE)
> for(i in 1:length(MDTEMP.list)) {
>   LSTname <- strsplit(MDTEMP.list[i], ".asc")[[1]][1]
# KML is compatible with POSIXct formats:
>   slice <- as.POSIXct(gsub("_", "-", strsplit(strsplit(MDTEMP.list[i],
+           ".asc")[[1]][1], "LST")[[1]][2]))
>   write('     <GroundOverlay>', filename, append=TRUE)
>   write(paste('   <name>', LSTname, '</name>',sep=""), filename, append=TRUE)
>   write('    <TimeSpan>', filename, append=TRUE)
>   write(paste('    <begin>',slice,'</begin>',sep=""), filename,
+           append=TRUE)
>   write(paste('      <end>',slice+1,'</end>',sep=""), filename, append=TRUE)
>   write('      </TimeSpan>', filename, append=TRUE)
>   write('     <color>99ffffff</color>', filename, append=TRUE)
>   write('      <Icon>', filename, append=TRUE)
>   write(paste('     <href>', getwd(), '/', LSTname, '.png</href>',sep=""),
+          filename, append=TRUE)
>   write('     <viewBoundScale>0.75</viewBoundScale>', filename,
+          append=TRUE)
>   write('     </Icon>', filename, append=TRUE)
>   write('      <altitude>50</altitude>', filename, append=TRUE)
>   write('      <altitudeMode>relativeToGround</altitudeMode>', filename, append=TRUE)
>   write('     <LatLonBox>', filename, append=TRUE)
>   write(paste('   <north>',grids.kml$ylim[[2]],'</north>',sep=""),
+          filename, append=TRUE)
>   write(paste('   <south>',grids.kml$ylim[[1]],'</south>',sep=""),
+         filename, append=TRUE)
>   write(paste('      <east>',grids.kml$xlim[[2]],'</east>',sep=""),
+          filename, append=TRUE)
>   write(paste('      <west>',grids.kml$xlim[[1]],'</west>',sep=""),
+          filename, append=TRUE)
>   write('    </LatLonBox>', filename, append=TRUE)
>   write('   </GroundOverlay>', filename, append=TRUE)
> }
> write('<ScreenOverlay>', filename, append=TRUE)
> write(paste('   <name>from',MDTEMPxlim[[1]],' to
+          ',MDTEMPxlim[[2]],'</name>',sep=""), filename, append=TRUE)
> write('   <Icon>', filename, append=TRUE)
> write(paste('     <href>',getwd(),'/legend.png</href>',sep=""), filename, append=TRUE)
> write('    </Icon>', filename, append=TRUE)
> write('    <overlayXY x="0" y="0" xunits="fraction" yunits="fraction"/>',
+       filename, append=TRUE)
> write('    <screenXY x="0" y="0" xunits="fraction" yunits="fraction"/>',
+       filename, append=TRUE)
> write('    <rotationXY x="0" y="0" xunits="fraction" yunits="fraction"/>',
+       filename, append=TRUE)
> write('    <size x="0" y="0" xunits="fraction" yunits="fraction"/>',
+       filename, append=TRUE)
> write('  </ScreenOverlay>', filename, append=TRUE)
> write('</Folder>', filename, append=TRUE)
> write('</kml>', filename, append=TRUE)
> close(filename)
```

The final output of data export is shown in Fig. 11.8. You can try to modify the original script and produce even more time-slices. In theory, we would normally want to interpolate temperatures for all 365 days and then export all these as images to Google Earth, but this is not maybe a good idea to do considering the size of the maps and the computational effort.

11.6 Summary points

The results of this case study demonstrate that regression-kriging models can be used also with spatio-temporal records, both of predictors and of target variables. The output pattern of the interpolated temperatures for this data set is mainly controlled with the MODIS LST images. Surprisingly, much of the variation in values can be explained by using just date. On the other hand, local variability of temperatures in the time dimension is more noisy (hence the significant nugget in Fig. 11.6) than in the geographical space. Although elevation and distance from the sea are less significant predictors of temperature than e.g. dates, it is even more important to include information on landscape to model changes in temperatures because this model is based on a physical law.

This exercises also shows that spatio-temporal prediction models are both more complex and more computationally intensive than plain spatial techniques. One significant issue not really addressed in this exercise is the problem of space-time anisotropy and space-time interactions. One should always address the issue that time units principally differ from space units, and separately quantify how fast autocorrelation decreases in time, and in space — the differences will often be large (Pebesma et al., 2007). Think of rainfall that might occur abruptly over short temporal periods, but then continuously over wide geographical areas. In this exercise we have only considered modeling the geometric anisotropy; we have completely ignored products of space-time and different scale in time dimension. An alternative would be to model the **zonal anisotropy**, where (also) the variance (sill) depends on direction.

Estimation of spatio-temporal variograms will often be cumbersome because we need to fit space-time models, for which we might not have enough space-time observations (Jost et al., 2005). Not to mention that many authors do not believe that temporal variograms are the same in both directions (past vs future). In fact, many argue whether there is any logical explanation to use observations from today to predict values from yesterday — conceptually speaking, the future does not influence the past! Nevertheless, if you compare this approach with the plain 2D techniques used to interpolate daily temperatures (Jarvis and Stuart, 2001; Schuurmans et al., 2007), you will notice that the space-time model (Eq.11.1.1) will be able to explain more variation in the temperature measurements than e.g. simple ordinary kriging using only temporally fixed measurements. Hence it is an investment worth the computational effort.

Self-study exercises:

(1.) What is the 95% range of values for `MDTEMP`? (HINT: use the `quantile` method and probabilities at 0.025 and 0.975.)

(2.) Are the differences between daily values of temperature also very different at other stations? (HINT: try looking at least two more stations.)

(3.) Try to fit a linear model with temperature only. Is the model much different? What would happen if we would try to model temperature as a function of time only?

(4.) How would you decrease the nugget variation in Fig. 11.6? (Consider at least two strategies and then try to prove them.)

(5.) Does it make sense to make predictions of temperature using measurements from a day or more days after (the future measurements)? How would you limit the search algorithm to one direction in time only?

(6.) What is the mean daily temperature for 1st of June 2006, at location `lon`=15.5 E, `lat`=45.0 N? (HINT: you will need to convert the time to numeric variable and then prepare a one-point spatial data layer. See section 11.5.1.)

(7.) At which locations is standard deviation of daily temperatures over the year the highest? (HINT: predict `MDTEMP` values for at least 20 slices over the whole year; then derive the mean value of temperature at each grid node and the standard deviation.)

(8.) Are the maps produced using the method explained in section 11.5.2 valid for the entire mapped region or only in the vicinity of the points? (how much are we allowed to extrapolate in geographical space? HINT: look at the variogram of residuals in Fig. 11.6.)

(9.) Obtain temperature measurements for a similar area, download the MODIS LST images (following the instructions in §4.2), and repeat this exercise with your own case study.

Further reading:

★ Jarvis, C. H., Stuart, N., 2001. A comparison among strategies for interpolating maximum and minimum daily air temperatures. Part II: The interaction between number of guiding variables and the type of interpolation method. Journal of Applied Meteorology 40: 1075–1084.

★ Jost, G., Heuvelink, G. B. M., Papritz, A. 2005. Analysing the space-time distributions of soil water storage of a forest ecosystem using spatio-temporal kriging. Geoderma 128 (3), 258–273.

★ Huerta, G., Sansó, B., Stroud, J.R. 2004. A spatiotemporal model for Mexico City ozone levels. Journal Of The Royal Statistical Society Series C, 53 (2): 231–248.

★ Pebesma, E. J., de Jong, K., Briggs, D. J., 2007. Visualising uncertain spatial and spatio-temporal data under different scenarios: an air quality example. International Journal of Geographical Information Science 21 (5): 515–527.

★ Schuurmans, J., Bierkens, M., Pebesma, E., Uijlenhoet, R., 2007. Automatic prediction of high-resolution daily rainfall fields for multiple extents: The potential of operational radar. Journal of Hydrometeorology 8: 1204–1224.

Bibliography

Addink, E. A., Stein, A., 1999. A comparison of conventional and geostatistical methods to replace clouded pixels in NOAA-AVHRR images. International Journal of Remote Sensing 20 (5): 961–977.

Ahmed, S., de Marsily, G., 1987. Comparison of geostatistical methods for estimating transmissivity using data on transmissivity and specific capacity. Water Resources Research 23 (9): 1717–1737.

Amante, C., Eakins, B. W., 2008. ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. National Geophysical Data Center, NESDIS, NOAA, U.S. Department of Commerce, Boulder, CO, p. 54.

Araújo, M. B., Thuiller, W., Williams, P. H., Reginster, I., 2005. Downscaling european species atlas distributions to a finer resolution: implications for conservation planning. Global Ecology and Biogeography 14 (1): 17–30.

Atkinson, P., Quattrochi, D. A., 2000. Special issue on geostatistics and geospatial techniques in remote sensing. Computers & Geosciences 26 (4): 359.

Baddeley, A., 2008. Analysing spatial point patterns in R. CSIRO, Canberra, Australia, p. 171.

Bahn, V., McGill, B. J., 2007. Can niche-based distribution models outperform spatial interpolation? Global Ecology and Biogeography 16 (6): 733–742.

Bailey, N., Clements, T., Lee, J. T., Thompson, S., 2003. Modelling soil series data to facilitate targeted habitat restoration: a polytomous logistic regression approach. Journal of Environmental Management 67 (4): 395–407.

Banerjee, S., Carlin, C. P., Gelfand, A. E. (Eds.), 2004. Hierarchical Modeling and Analysis for Spatial Data. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, p. 472.

Banks, J. (Ed.), 1998. Handbook of Simulation — Principles, Methodology, Advances, Applications, and Practice. Wiley, New York, p. 864.

Bárdossy, A., Li, J., 2008. Geostatistical interpolation using copulas. Water Resources Research 44: W07412.

Bartholome at al., 2002. GLC 2000 Global Land Cover mapping for the year 2000. EUR 20524 EN. European Commission, DG Joint Research Centre, Luxemburg, p. 62.

Batjes, N., 1996. Total carbon and nitrogen in the soils of the world. European Journal of Soil Science 47: 151–163.

Batjes, N., 2008. ISRIC-WISE Harmonized Global Soil Profile Dataset (Ver. 3.1). Report 2008/02 (with dataset). ISRIC — World Soil Information, Wageningen, p. 59.

Batjes, N., 2009. Harmonized soil profile data for applications at global and continental scales: updates to the WISE database. Soil Use and Management 25: 124–127.

Batjes, N., Al-Adamat, R., Bhattacharyya, T., Bernoux, M., Cerri, C., Gicheru, P., Kamoni, P., Milne, E., Pal, D., Rawajfi, Z., 2007. Preparation of consistent soil data sets for modelling purposes: Secondary SOTER data for four case study areas. Agriculture, Ecosystems & Environment 122: 26–34.

Becker, J. J., Sandwell, D. T., Smith, W. H. F., Braud, J., Binder, B., Depner, J., Fabre, D., Factor, J., Ingalls, S., Kim, S.-H., Ladner, R., Marks, K., Nelson, S., Pharaoh, A., Sharman, G., Trimmer, R., vonRosenburg, J., Wallace, G., Weatherall, P., 2009. Global Bathymetry and Elevation Data at 30 Arc Seconds Resolution: SRTM30_PLUS. Marine Geodesy in press: 18.

Berman, M., Diggle, P. J., 1989. Estimating weighted integrals of the second-order intensity of a spatial point process. Journal of the Royal Statistical Society B 51: 81–92.

Bierkens, M. F. P., Burrough, P. A., 1993. The indicator approach to categorical soil data I: Theory. Journal of Soil Science 44: 361–368.

Bishop, T., Minasny, B., McBratney, A., 2006. Uncertainty analysis for soil-terrain models. International Journal of Geographical Information Science 20 (1-2): 117–134.

Bishop, T. F. A., McBratney, A. B., 2001. A comparison of prediction methods for the creation of field-extent soil property maps. Geoderma 103 (1-2): 149–160.

Bishop, T. F. A., Minasny, B., 2005. Digital Soil-Terrain Modelling: The Predictive Potential and Uncertainty. In: Grunwald, S. (Ed.), Environmental Soil-Landscape Modeling: Geographic Information Technologies and Pedometrics. CRC Press, Boca Raton, Florida, pp. 185–213.

Bivand, R., 2006. Implementing Spatial Data Analysis Software Tools in R. Geographical Analysis 38: 23–40.

Bivand, R., Pebesma, E., Rubio, V., 2008. Applied Spatial Data Analysis with R. Use R Series. Springer, Heidelberg, p. 400.

Bivand, R. S., 2005. Interfacing GRASS 6 and R. Status and development directions. GRASS Newsletter 3: 11–16.

Bleines, C., Perseval, S., Rambert, F., Renard, D., Touffait, Y., 2004. ISATIS. Isatis software manual, 5th Edition. Geovariances & Ecole Des Mines De, Paris, p. 710.

Bolstad, P. (Ed.), 2008. GIS Fundamentals, 3rd Edition. Atlas Books, Minnesota, p. 650.

Bonan, G. B., Levis, S., Sitch, S., Vertenstein, M., Oleson, K. W., 2003. A dynamic global vegetation model for use with climate models: concepts and description of simulated vegetation dynamics. Global Change Biology 9 (11): 1543–1566.

Boucneau, G., van Meirvenne, M., Thas, O., Hofman, G., 1998. Integrating properties of soil map delineations into ordinary kriging. European Journal of Soil Science 49 (2): 213–229.

Box, G. E. P., Muller, M. E., 1958. A note on the generation of random normal deviates. The Annals of Mathematical Statistics 29 (2): 610–611.

Bragato, G., 2004. Fuzzy continuous classification and spatial interpolation in conventional soil survey for soil mapping of the lower Piave plain. Geoderma 118 (1-2): 1–16.

Brenning, A., 2008. Statistical geocomputing combining r and saga: The example of landslide susceptibility analysis with generalized additive models. In: Böhner, J., Blaschke, T., Montanarella, L. (Eds.), SAGA — Seconds Out. Vol. 19. Hamburger Beiträge zur Physischen Geographie und Landschaftsökologie, pp. 23–32.

Brus, D. J., Heuvelink, G. B. M., 2007. Optimization of sample patterns for universal kriging of environmental variables. Geoderma 138 (1-2): 86–95.

Burns, P., 2009. The R Inferno. Burns Statistics, London, p. 103.

Burrough, P. A., McDonnell, R. A., 1998. Principles of Geographical Information Systems. Oxford University Press Inc., New York, p. 333.

Calenge, C., 2007. Exploring Habitat Selection by Wildlife with adehabitat. Journal of Statistical Software 22 (6): 2–19.

Carré, F., Girard, M. C., 2002. Quantitative mapping of soil types based on regression kriging of taxonomic distances with landform and land cover attributes. Geoderma 110 (3-4): 241–263.

Chambers, J. M., Hastie, T. J., 1992. Statistical Models in S. Wadsworth & Brooks/Cole, Pacific Grove, California, p. 595.

Chang Seong, J., Mulcahy, K., Usery, E., 2002. The Sinusoidal Projection: A New Importance in Relation to Global Image Data. The Professional Geographer 54 (2): 218–225.

Chefaoui, R. M., Lobo, J. M., 2008. Assessing the effects of pseudo-absences on predictive distribution model performance. Ecological Modelling 210: 478–486.

Chiles, J. P., Delfiner, P., 1999. Geostatistics: modeling spatial uncertainty. John Wiley & Sons, New York, p. 720.

Christensen, R., 2001. Linear Models for Multivariate, Time Series, and Spatial Data, 2nd Edition. Springer Verlag, New York, p. 393.

Congalton, R. G., Green, K., 1999. Assessing the accuracy of remotely sensed data: principles and practices. Lewis, Boca Raton, FL, p. 137.

Conrad, O., 2006. SAGA — Program Structure and Current State of Implementation. In: Böhner, J., McCloy, K. R., Strobl, J. (Eds.), SAGA — Analysis and Modelling Applications. Vol. 115. Verlag Erich Goltze GmbH, pp. 39–52.

Conrad, O., 2007. SAGA — Entwurf, Funktionsumfang und Anwendung eines Systems für Automatisierte Geowissenschaftliche Analysen. Ph.D. thesis, University of Göttingen, Göttingen.

Craglia, M., Goodchild, M., Annoni, A., Camara, G., Gould, M., Kuhn, W., Mark, D., Masser, I., Maguire, D., Liang, S., Parsons, E., 2008. Next-Generation Digital Earth: A position paper from the Vespucci Initiative for the Advancement of Geographic Information Science. International Journal of Spatial Data Infrastructures Research 3 (6): 146–167.

Cressie, N. A. C., 1990. The origins of kriging. Mathematical Geology 22 (3): 239–252.

Cressie, N. A. C., 1993. Statistics for Spatial Data, revised edition. John Wiley & Sons, New York, p. 416.

D'Agostino, V., Zelenka, A., 1992. Supplementing solar radiation network data by co-Kriging with satellite images. International Journal of Climatology 12 (7): 749–761.

Davis, C., Fonseca, F., Câmara, G., 2009. Beyond SDI: Integrating Science and Communities to Create Environmental Policies for the Sustainability of the Amazon. International Journal of Spatial Data Infrastructures Research 4: 156–174.

de Fries, F., Groot, W., Hoogland, T., Denneboom, J., 2003. De Bodemkaart van Nederland digitaal. Alterra Rapport 811. Alterra, Wageningen, p. 45.

de Gruijter, J. J., Walvoort, D. J. J., van Gaans, P. F. M., 1997. Continuous soil maps — a fuzzy set approach to bridge the gap between aggregation levels of process and distribution models. Geoderma 77 (2-4): 169–195.

Deutsch, C. V., Journel, A. G., 1998. GSLIB: Geostatistical Software and User's Guide, 2nd Edition. Oxford University Press, New York, p. 384.

Dial, G., Bowen, H., Gerlach, F., Grodecki, J., Oleszczuk, R., 2003. Ikonos satellite, imagery, and products. Remote Sensing of Environment 88: 23–36.

Diggle, P. J., 2003. Statistical Analysis of Spatial Point Patterns, 2nd Edition. Arnold Publishers, p. 288.

Diggle, P. J., Ribeiro Jr, P. J., 2007. Model-based Geostatistics. Springer Series in Statistics. Springer, p. 288.

Doll, C., Muller, J.-P., Morley, J., 2007. Mapping regional economic activity from night-time light satellite imagery. Ecological Economics 57 (1): 75–92.

Donato, G., Belongie, S., 2003. Approximation Methods for Thin Plate Spline Mappings and Principal Warps. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (Eds.), Proceedings, Part III, Lecture Notes in Computer Science. Computer Vision — ECCV 2002: 7th European Conference on Computer Vision. Copenhagen, Denmark, pp. 21–31.

Dooley, M. A., Lavin, S. J., 2007. Visualizing method-produced uncertainty in isometric mapping. Cartographic Perspectives 56: 17–36.

D'Or, D., 2003. Spatial prediction of soil properties, the Bayesian Maximum Entropy approach. Phd, Université Catholique de Louvain.

D'Or, D., Bogaert, P., 2005. Spatial prediction of categorical variables with the Bayesian Maximum Entropy approach: the Ooypolder case study. European Journal of Soil Science 55 (December): 763–775.

Draper, N. R., Smith, H., 1998. Applied Regression Analysis, 3rd Edition. John Wiley, New York, p. 697.

Dubois, G. (Ed.), 2005. Automatic mapping algorithms for routine and emergency monitoring data. Report on the Spatial Interpolation Comparison (SIC2004) exercise. EUR 21595 EN. Office for Official Publications of the European Communities, Luxembourg, p. 150.

Dubois, G., Galmarini, S., 2004. Introduction to the Spatial Interpolation Comparison (SIC). Applied GIS 1 (2): 9–11.

Ellis, E., Ramankutty, N., 2000. Putting people in the map: anthropogenic biomes of the world. Frontiers in Ecology and the Environment 6 (8): 439–447.

Endreny, T. A., Wood, E. F., 2001. Representing elevation uncertainty in runoff modelling and flowpath mapping. Hydrological Processes 15: 2223–2236.

Engler, R., Guisan, A., Rechsteiner, L., 2004. An improved approach for predicting the distribution of rare and endangered species from occurrence and pseudo-absence data. Journal of Applied Ecology 41 (2): 263–274.

Eswaran, H., van den Berg, E., Reich, P., 1993. Organic carbon in soils of the world. Soil Science Society of America journal 57: 192–194.

Evans, J. S., Hudak, A. T., 2007. A multiscale curvature filter for identifying ground returns from discrete return lidar in forested environments. IEEE Transactions on Geoscience and Remote Sensing 45 (4): 1029–1038.

Fassó, A., Cameletti, M., 2009. A Unified Statistical Approach for Simulation, Modeling, Analysis and Mapping of Environmental Data. SIMULATION in press: 1–16.

Fisher, P., 1998. Improved Modeling of Elevation Error with Geostatistics. GeoInformatica 2 (3): 215–233.

Fisher, P. F., Wood, J., Cheng, T., 2005. Fuzziness and Ambiguity in Multi-Scale Analysis of Landscape Morphometry. In: Petry, F. E., Robinson, V. B., Cobb, M. A. (Eds.), Fuzzy Modeling with Spatial Information for Geographic Problems. Springer-Verlag, Berlin, pp. 209–232.

Foody, G. M., 2004. Thematic map comparison: evaluating the statistical significance of differences. Photogrammetric Engineering and Remote Sensing 70: 627–633.

Fotheringham, A. S., Brunsdon, C., Charlton, M., 2002. Geographically Weighted Regression: The Analysis of Spatially Varying Relationships. GIS & Remote Sensing. Wiley, p. 282.

Gandin, L. S., 1963. Objective Analysis of Meteorological Fields. translated from Russian in 1965 by Israel Program for Scientific Translations, Jerusalem. Gidrometeorologicheskoe Izdatel'stvo (GIMIZ), Leningrad, p. 242.

Gelfand, A. E., Kim, H.-J., Sirmans, C. F., , Banerjee, S., 2003. Spatial Modeling With Spatially Varying Coefficient Processes. Journal of the American Statistical Association 98 (462): 387–396.

Goldberger, A., 1962. Best Linear Unbiased Prediction in the Generalized Linear Regression Model. Journal of the American Statistical Association 57: 369–375.

Goovaerts, P., 1997. Geostatistics for Natural Resources Evaluation (Applied Geostatistics). Oxford University Press, New York, p. 496.

Goovaerts, P., 1999. Geostatistics in soil science: State-of-the-art and perspectives. Geoderma 89 (1-2): 1–45.

Gotway, C., Young, L., 2002. Combining Incompatible Spatial Data. Journal of the American Statistical Association 97: 632–648.

Gotway, C. A., Stroup, W. W., 1997. A Generalized Linear Model approach to spatial data analysis and prediction. Journal of Agricultural, Biological, and Environmental Statistics 2 (2): 157–198.

Gotway Crawford, C. A., Young, L. J., 2008. Geostatistics: What's Hot, What's Not, and Other Food for Thought. In: Wan, Y. et al. (Ed.), Proceeding of the 8th international symposium on spatial accuracy assessment in natural resources and environmental sciences (Accuracy 2008). World Academic Union (Press), Shanghai, pp. 8–16.

Griffith, D., 2008. Spatial-filtering-based contributions to a critique of geographically weighted regression (GWR). Environment and Planning A40: 2751–2769.

Grohmann, C. H., 2004. Morphometric analysis in Geographic Information Systems: applications of free software GRASS and R. Computers & Geosciences 30: 1055–1067.

Groombridge, B., Jenkins, M., 2002. World Atlas of Biodiversity: Earth's Living Resources in the 21st Century. University of California Press, p. 340.

Grose, D., Crouchley, R., van Ark, T., Allan, R., Kewley, J., Braimah, A., Hayes, M., 2006. sabreR: Grid-Enabling the Analysis of MultiProcess Random Effect Response Data in R. In: Halfpenny, P. (Ed.), Second International Conference on e-Social Science. Vol. 3c. National Centre for e-Social Science, Manchester, UK, p. 12.

Grossman, J., Grosz, A., Schweitzer, P., Schruben, P. (Eds.), 2008. The National Geochemical Survey — Database and Documentation, Version 5. Open-File Report 2004-1001. U.S. Geological Survey, Reston, VA, p. 45.

Guisan, A., Zimmermann, N. E., 2000. Predictive habitat distribution models in ecology. Ecological Modelling 135 (2-3): 147–186.

Guttorp, P., 2003. Environmental Statistics — A Personal View. International Statistical Review 71: 169–179.

Haas, T. C., 1990. Kriging and automated semivariogram modelling within a moving window. Atmospheric Environment 24A: 1759–1769.

Hansen, M., DeFries, R., Townshend, J., Sohlberg, R., 2000. Global land cover classification at 1km resolution using a decision tree classifier. International Journal of Remote Sensing 21: 1331–1365.

Hardy, R. L., 1971. Multiquadratic equations of topography and other irregular surfaces. Journal of Geophysical Research 76: 1905–1915.

Hayakawa, Y., Oguchi, T., Lin, Z., 2008. Comparison of new and existing global digital elevation models: ASTER G-DEM and SRTM-3. Geophys. Res. Lett. 35: L17404.

Henderson, B. L., Bui, E. N., Moran, C. J., Simon, D. A. P., 2004. Australia-wide predictions of soil properties using decision trees. Geoderma 124 (3-4): 383–398.

Hengl, T., 2006. Finding the right pixel size. Computers & Geosciences 32 (9): 1283–1298.

Hengl, T., Bajat, B., Reuter, H., Blagojević, D., 2008. Geostatistical modelling of topography using auxiliary maps. Computers & Geosciences 34: 1886–1899.

Hengl, T., Heuvelink, G. B. M., Rossiter, D. G., 2007a. About regression-kriging: from theory to interpretation of results. Computers & Geosciences 33 (10): 1301–1315.

Hengl, T., Heuvelink, G. M. B., Stein, A., 2004a. A generic framework for spatial prediction of soil variables based on regression-kriging. Geoderma 122 (1-2): 75–93.

Hengl, T., Minasny, B., Gould, M., 2009a. A geostatistical analysis of geostatistics. Scientometrics 80: 491–514.

Hengl, T., Reuter, H. (Eds.), 2008. Geomorphometry: Concepts, Software, Applications. Vol. 33 of Developments in Soil Science. Elsevier, Amsterdam, p. 772.

Hengl, T., Rossiter, D. G., Stein, A., 2004b. Soil sampling strategies for spatial prediction by correlation with auxiliary maps. Australian Journal of Soil Research 41 (8): 1403–1422.

Hengl, T., Sierdsema, H., Radović, A., Dilo, A., 2009b. Spatial prediction of species' distributions from occurrence-only records: combining point pattern analysis, ENFA and regression-kriging. Ecological Modelling 220: 3499–3511.

Hengl, T., Toomanian, N., 2006. Maps are not what they seem: representing uncertainty in soil-property maps. In: Caetano, M., Painho, M. (Eds.), Proceedings of the 7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences (Accuracy 2006). Instituto Geográphico Português, Lisbon, Portugal, pp. 805–813.

Hengl, T., Toomanian, N., Reuter, H. I., Malakouti, M. J., 2007b. Methods to interpolate soil categorical variables from profile observations: lessons from Iran. Geoderma 140 (4): 417–427.

Hengl, T., Walvoort, D. J. J., Brown, A., Rossiter, D. G., 2004c. A double continuous approach to visualisation and analysis of categorical maps. International Journal of Geographical Information Science 18 (2): 183–202.

Hession, S. L., Shortridge, A. M., Torbick, M. N., 2006. Categorical models for spatial data uncertainty. In: Caetano, M., Painho, M. (Eds.), Proceedings of the 7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences (Accuracy 2006). Instituto Geográphico Português, Lisbon, pp. 386–395.

Heuvelink, G., 1998. Error propagation in environmental modelling with GIS. Taylor & Francis, London, UK, p. 144.

Heuvelink, G., 2002. Analysing uncertainty propagation in GIS: why is it not that simple? In: Foody, G., Atkinson, P. (Eds.), Uncertainty in Remote Sensing and GIS. Wiley, Chichester, pp. 155–165.

Heuvelink, G. B. M., Pebesma, E. J., 1999. Spatial aggregation and soil process modelling. Geoderma 89 (1-2): 47–65.

Heuvelink, G. B. M., Webster, R., 2001. Modelling soil variation: past, present, and future. Geoderma 100 (3-4): 269–301.

Hijmans, R., Cameron, S., Parra, J., Jones, P., Jarvis, A., 2005. Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology 25: 1965–1978.

Hirzel, A. H., Guisan, A., 2002. Which is the optimal sampling strategy for habitat suitability modelling. Ecological Modelling 157 (2-3): 331–341.

Holmes, K. W., Chadwick, O. A., Kyriakidis, P. C., 2000. Error in a USGS 30m digital elevation model and its impact on digital terrain modeling. Journal of Hydrology 233: 154–173.

Hsing-Cheng, H., Chun-Shu, C., 2007. Optimal Geostatistical Model Selection. Journal of the American Statistical Association 102: 1009–1024.

Huerta, G., Sansó, B., Stroud, J., 2004. A spatiotemporal model for Mexico City ozone levels. Journal Of The Royal Statistical Society Series C 53 (2): 231–248.

Huete, A., Didan, K., Miura, T., Rodriguez, E., Gao, X., Ferreira, L., 2002. Overview of the radiometric and biophysical performance of the MODIS vegetation indices. Remote Sensing of Environment 83: 195–213.

Hunter, G. J., Goodchild, M. F., 1997. Modeling the Uncertainty of Slope and Aspect Estimates Derived from Spatial Databases. Geographical Analysis 29 (1): 35–49.

Hutchinson, M. F., 1989. A new procedure for gridding elevation and stream line data with automatic removal of spurious pits. Journal of Hydrology 106: 211–232.

Hutchinson, M. F., 1995. Interpolating mean rainfall using thin plate smoothing splines. International Journal of Geographical Information Systems 9: 385–403.

Hutchinson, M. F., 1996. A locally adaptive approach to the interpolation of digital elevation models. In: Proceedings of the Third International Conference/Workshop on Integrating GIS and Environmental Modeling. National Center for Geographic Information and Analysis, Santa Barbara, CA, p. 6.

Isaaks, E. H., Srivastava, R. M., 1989. Applied Geostatistics. Oxford University Press, New York, p. 542.

Jarvis, C. H., Stuart, N., 2001. A comparison among strategies for interpolating maximum and minimum daily air temperatures. Part II: The interaction between number of guiding variables and the type of interpolation method. Journal of Applied Meteorology 40: 1075–1084.

Jiménez-Valverde, A., Gómez, J., Lobo, J., Baselga, A., Hortal, J., 2008. Challenging species distribution models: the case of Maculinea nausithous in the Iberian Peninsula. Annales Zoologici Fennici 45: 200–210.

Jost, G., Heuvelink, G. B. M., Papritz, A., 2005. Analysing the space-time distributions of soil water storage of a forest ecosystem using spatio-temporal kriging. Geoderma 128 (3): 258–273.

Journel, A. G., 1986. Constrained interpolation and qualitative information. Mathematical Geology 18 (3): 269–286.

Kanevski, M., Maignan, M., Demyanov, V., Maignan, M., 1997. How neural network 2-D interpolations can improve spatial data analysis: neural network residual kriging (NNRK). In: Hohn, M. (Ed.), Proceedings of the Third Annual Conference of the IAMG. International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, pp. 549–554.

Kempen, B., Brus, D., Heuvelink, G., Stoorvogel, J., 2009. Updating the 1:50,000 Dutch soil map using legacy soil data: A multinomial logistic regression approach. Geoderma 151 (3-4): 311–326.

Kitanidis, P. K., 1994. Generalized covariance functions in estimation. Mathematical Geology 25: 525–540.

Kleinschmidt, I. Sharp, B. L., Clarke, G. P. Y., Curtis, B., Fraser, C., 2005. Use of Generalized Linear Mixed Models in the Spatial Analysis of Small-Area Malaria Incidence Rates in KwaZulu Natal, South Africa. American Journal of Epidemiology 153 (12): 1213–1221.

Knotters, M., Brus, D. J., Voshaar, J. H. O., 1995. A comparison of kriging, co-kriging and kriging combined with regression for spatial interpolation of horizon depth with censored observations. Geoderma 67 (3-4): 227–246.

Koptsik, S., Koptsik, G., Livantsova, S., Eruslankina, L., Zhmelkova, T., Vologdina, Z., 2003. Heavy metals in soils near the nickel smelter: Chemistry, spatial variation, and impacts on plant diversity. Journal of Environmental Monitoring 5 (3): 441–50.

Kreft, H., Jetz, W., 2007. Global Patterns and Determinants of Vascular Plant Diversity. Proceedings National Academy of Science 104: 5925–5930.

Krige, D. G., 1951. A statistical approach to some basic mine valuation problems on the Witwatersrand. Journal of the Chemical, Metallurgical and Mining Society 52: 119–139.

Kuhnert, P., Venables, W. N., 2005. An Introduction to R: Software for Statistical Modelling & Computing. CSIRO, Canberra, Australia, p. 362.

Kutner, M. H., Nachtsheim, C. J., Neter, J., Li, W. (Eds.), 2004. Applied Linear Statistical Models, 5th Edition. McGraw-Hill, p. 1396.

Kyriakidis, P. C., Journel, A. G., 1999. Geostatistical Space–Time Models: A Review. Mathematical Geology 31 (6): 651–684.

Kyriakidis, P. C., Shortridge, A. M., Goodchild, M. F., 1999. Geostatistics for conflation and accuracy assessment of Digital Elevation Models. International Journal of Geographical Information Science 13 (7): 677–708.

Lagacherie, P., McBratney, A., Voltz, M. (Eds.), 2006. Digital Soil Mapping: An Introductory Perspective. Developments in Soil Science. Elsevier, Amsterdam, p. 350.

Lam, N. S.-N., 1983. Spatial interpolation methods: a review. The American Cartographer 10: 129–149.

Lark, R. M., Cullis, B., Welham, S. J., 2005. On Spatial Prediction of Soil Properties in the Presence of a Spatial Trend: The Empirical Best Linear Unbiased Predictor (E-BLUP) with REML. European Journal of Soil Science 57: 787–799.

Latimer, A. M., Wu, S., Gelfand, A. E., Silander Jr., J. A., 2004. Building statistical models to analyze species distributions. Ecological Applications 16 (1): 33–50.

Legendre, P., Fortin, M. J., 1989. Spatial pattern and ecological analysis. Plant Ecology 80 (2): 107–138.

Lehner, B., Doll, P., 2004. Development and validation of a global database of lakes, reservoirs and wetlands. Journal of Hydrology 296 (1-4): 1–22.

Leigh, E. G. J., de Lao, S. L., Condit, R. G., Hubbell, S. P., Foster, R. B., Perez, R., 2004. Barro Colorado Island Forest Dynamics Plot, Panama. In: Losos, E. C., Leigh, E. G. J. (Eds.), Tropical forest diversity and dynamism: Findings from a large-scale plot network. University of Chicago Press, Chicago, pp. 451–463.

Leopold, U., Heuvelink, G. B. M., Tiktak, A., Finke, P. A., Schoumans, O., 2005. Accounting for change of support in spatial accuracy assessment of modelled soil mineral phosphorous concentration. Geoderma 130 (3-4): 368–386.

Li, J., Heap, A., 2008. A review of spatial interpolation methods for environmental scientists. Record 2008/23. Geoscience Australia, Canberra, p. 137.

Li, W., Zhang, C., Burt, J., Zhu, A., 2005a. A markov chain-based probability vector approach for modelling spatial uncertainties of soil classes. Soil Science Society of America Journal 69: 1931–1942.

Li, W., Zhang, C., Burt, J., Zhu, A., Feyen, J., 2004. Two-dimensional markov chain simulation of soil type spatial distribution. Soil Science Society of America Journal 68: 1479–1490.

Li, Z., Zhu, Q., Gold, C., 2005b. Digital Terrain Modeling: Principles and Methodology. CRC Press, Boca Raton, Florida, p. 319.

Lloyd, C., 2009. Nonstationary models for exploring and mapping monthly precipitation in the United Kingdom. International Journal of Climatology in press.

Lloyd, C. D., 2005. Assessing the effect of integrating elevation data into the estimation of monthly precipitation in Great Britain. Journal of Hydrology 308 (1-4): 128–150.

Lloyd, C. D., Atkinson, P. M., 1998. Scale and the spatial structure of landform: optimizing sampling strategies with geostatistics. In: Proceedings of the 3rd International Conference on GeoComputation, University of Bristol, United Kingdom, 17-19 September 1998. University of Bristol, Bristol, UK, p. 16.

Lloyd, C. D., Atkinson, P. M., 2002. Deriving DSMs from LiDAR data with kriging. International Journal of Remote Sensing 23 (12): 2519–2524.

Lunetta, R., Knight, J., Ediriwickrema, J., Lyon, J., Dorsey Worthy, L., 2006. Land-cover change detection using multi-temporal MODIS NDVI data. Remote Sensing of Environment 105 (2): 142–154.

Matheron, G., 1962. Traité de géostatistique appliquée. Vol. 14 of Mémoires du Bureau de Recherches Géologiques et Minières. Editions Technip, Paris, p. NA.

Matheron, G., 1969. Le krigeage universel. Vol. 1. Cahiers du Centre de Morphologie Mathématique, École des Mines de Paris, Fontainebleau, p. NA.

McBratney, A. B., de Gruijter, J. J., Brus, D. J., 1992. Spatial prediction and mapping of continuous soil classes. Geoderma 54 (1-4): 39–64.

McBratney, A. B., Mendoça Santos, M. L., Minasny, B., 2003. On digital soil mapping. Geoderma 117 (1-2): 3–52.

McCloskey, J., Spalding, H., 1989. A reconnaissance level inventory of the amount of wilderness remaining in the world. Ambio 18 (4): 221–227.

McKenzie, N. J., Ryan, P. J., 1999. Spatial prediction of soil properties using environmental correlation. Geoderma 89 (1-2): 67–94.

Miller, J., 2005. Incorporating Spatial Dependence in Predictive Vegetation Models: Residual Interpolation Methods. The Professional Geographer 57 (2): 169–184.

Miller, J., Franklin, J., Aspinall, R., 2007. Incorporating spatial dependence in predictive vegetation models. Ecological Modelling 202: 225–242.

Minasny, B., McBratney, A. B., 2001. A rudimentary mechanistic model for soil formation and landscape development II. A two-dimensional model incorporating chemical weathering. Geoderma 103: 161–179.

Minasny, B., McBratney, A. B., 2005. The Matérn function as a general model for soil variograms. Geoderma 128 (3-4): 192–207.

Minasny, B., McBratney, A. B., 2006. A conditioned Latin hypercube method for sampling in the presence of ancillary information. Computers & Geosciences 32 (9): 1378–1388.

Minasny, B., McBratney, A. B., 2007. Spatial prediction of soil properties using EBLUP with Matérn covariance function. Geoderma 140: 324–336.

Mitas, L., Mitasova, H., 1999. Spatial interpolation. In: Longley, P., Goodchild, M. F., Maguire, D. J., Rhind, D. W. (Eds.), Geographical Information Systems: Principles, Techniques, Management and Applications. Vol. 1. Wiley, pp. 481–492.

Mitasova, H., Mitas, L., 1993. Interpolation by regularized spline with tension, I Theory and implementation. Mathematical Geology 25: 641–655.

Mitasova, H., Mitas, L., Harmon, R., 2005. Simultaneous Spline Approximation and Topographic Analysis for Lidar Elevation Data in Open-Source GIS. IEEE Geoscience and Remote Sensing Letters 2: 375–379.

Montgomery, D. C., 2005. Design and Analysis of Experiments, 6th Edition. Wiley, New York, p. 660.

Moyeed, R., Papritz, A., 2002. An Empirical Comparison of Kriging Methods for Nonlinear Spatial Point Prediction. Mathematical Geology 34 (4): 365–386.

Murrell, P., 2006. R Graphics. Computer Science and Data Analysis Series. Chapman & Hall/CRC, Boca Raton, FL, p. 328.

Myers, D. E., 1994. Spatial interpolation: an overview. Geoderma 62: 17–28.

Neteler, M., 2005. Time series processing of MODIS satellite data for landscape epidemiological applications. International Journal of Geoinformatics 1 (1): 133–138.

Neteler, M., Mitasova, H., 2008. Open Source GIS: A GRASS GIS Approach, 3rd Edition. Springer, New York, p. 406.

Neter, J., Kutner, M. H., Nachtsheim, C. J., Wasserman, W. (Eds.), 1996. Applied Linear Statistical Models, 4th Edition. McGraw-Hill, p. 1391.

Odeh, I. O. A., McBratney, A. B., Chittleborough, D. J., 1995. Further results on prediction of soil properties from terrain attributes: heterotopic cokriging and regression-kriging. Geoderma 67 (3-4): 215–226.

Oksanen, J., 2006. Uncovering the statistical and spatial characteristics of fine toposcale DEM error. International Journal of Geographical Information Science 20 (4): 345–369.

Olson et al., 2001. Terrestrial Ecoregions of the World: A New Map of Life on Earth. BioScience 51: 933–938.

Ott, R. L., Longnecker, M. (Eds.), 2001. An Introduction to Statistical Methods and Data Analysis, 5th Edition. Duxbury press, p. 1152.

Ozdogana, M., Gutman, G., 2008. A new methodology to map irrigated areas using multi-temporal MODIS and ancillary data: An application example in the continental US. Remote Sensing of Environment 112 (9): 3520–3537.

Papritz, A., 2009. Limitations of Indicator Kriging for Predicting Data with Trend. In: Cornford, D. et al. (Ed.), StatGIS Conference Proceedings. Milos, Greece, pp. 1–6.

Papritz, A., Herzig, C., Borer, F., Bono, R., 2005. Modelling the spatial distribution of copper in the soils around a metal smelter in northwestern Switzerland. In: Renard, P., Demougeot-Renard, H., Froidevaux, R. (Eds.), Geostatistics for environmental Applications: Proceedings of the fifth European conference on geostatistics for environmental applications. Springer, Berlin Heidelberg New York, pp. 343–354.

Papritz, A., Stein, A., 1999. Spatial prediction by linear kriging. In: Stein, A., van der Meer, F., Gorte, B. (Eds.), Spatial statistics for remote sensing. Kluwer Academic publishers, Dodrecht, pp. 83–113.

Pardo-Iguzquiza, E., Dowd, P. A., 2005. Multiple indicator cokriging with application to optimal sampling for environmental monitoring. Computers & Geosciences 31 (1): 1–13.

Park, S. J., Vlek, P. L. G., 2002. Environmental correlation of three-dimensional soil spatial variability: a comparison of three adaptive techniques. Geoderma 109 (1-2): 117–140.

Patil, G. P., 2002. Composite sampling. In: El-Shaarawi, A. H., Piegorsch, W. W. (Eds.), Encyclopedia of Environmetrics. Vol. 1. John Wiley & Sons, Chichester, UK, pp. 387–391.

Pebesma, E., 2006. The Role of External Variables and GIS Databases in Geostatistical Analysis. Transactions in GIS 10 (4): 615–632.

Pebesma, E., Cornford, D., Dubois, G., Heuvelink, G., Hristopoulos, D., Pilz, J., Stoehlker, U., Skoien, J., 2009. INTAMAP: an interoperable automated interpolation web service. In: Cornford, D. et al. (Ed.), StatGIS Conference Proceedings. Milos, Greece, pp. 1–6.

Pebesma, E. J., 2004. Multivariable geostatistics in S: the gstat package. Computers & Geosciences 30 (7): 683–691.

Pebesma, E. J., Bivand, R. S., 2005. Classes and methods for spatial data in R. R News 5 (2): 9–13.

Pebesma, E. J., de Jong, K., Briggs, D. J., 2007. Visualising uncertain spatial and spatio-temporal data under different scenarios: an air quality example. International Journal of Geographical Information Science 21 (5): 515–527.

Pebesma, E. J., Duin, R. N. M., Burrough, P. A., 2005. Mapping sea bird densities over the North Sea: spatially aggregated estimates and temporal changes. Environmetrics 16 (6): 573–587.

Phillips, S. J., Dudík, M., 2008. Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. Ecography 31: 161–175.

Planchon, O., Darboux, F., 2001. A fast, simple and versatile algorithm to fill the depressions of digital elevation models. Catena 46: 159–176.

Potapov, P. et al., 2008. Mapping the world's intact forest landscapes by remote sensing. Ecology and Society 13 (2): 51.

Potere, D., 2008. Horizontal Positional Accuracy of Google Earth's High-Resolution Imagery Archive. Sensors 8: 7973–7981.

R Development Core Team, 2009. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, p. 409, ISBN 3-900051-07-0.

Rabus, B., Eineder, M., Roth, A., Bamler, R., 2003. The shuttle radar topography mission — a new class of digital elevation models acquired by spaceborne radar. Photogrammetric Engineering and Remote Sensing 57 (4): 241–262.

Reimann, C., Filzmoser, P., Garrett, R., Dutter, R., 2008. Statistical Data Analysis Explained Applied Environmental Statistics with R. Wiley, Chichester, p. 337.

Ribeiro Jr, P. J., Christensen, O. F., Diggle, P. J., 2003. geoR and geoRglm: Software for Model-Based Geostatistics. In: Hornik, K., Leisch, F., Zeileis, A. (Eds.), Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003). Technical University Vienna, Vienna, pp. 517–524.

Rikken, M., Van Rijn, R., 1993. Soil pollution with heavy metals — an inquiry into spatial variation, cost of mapping and the risk evaluation of copper, cadmium, lead and zinc in the oodplains of the meuse west of Stein. Technical Report. Department of Physical Geography, Utrecht University, Utrecht, p. NA.

Ripley, B. D., 2004. Spatial statistics, 4th Edition. Wiley-IEEE, London, p. 252.

Rodriguez Lado, L., Hengl, T., Reuter, H., 2009. Heavy metals in European soils: a geostatistical analysis of the FOREGS Geochemical database. Geoderma 148: 189–199.

Romić, M., Hengl, T., Romić, D., 2007. Representing soil pollution by heavy metals using continuous limitation scores. Computers & Geosciences 33: 1316–1326.

Rossiter, D. G., 2007. Technical Note: Co-kriging with the gstat package of the R environment for statistical computing, 2nd Edition. International Institute for Geo-information Science & Earth Observation (ITC), Enschede, Netherlands, p. 81.

Rossiter, D. G., 2009. Introduction to the R Project for Statistical Computing for use at ITC, 3rd Edition. International Institute for Geo-information Science & Earth Observation (ITC), Enschede, Netherlands, p. 128.

Rowe, J. S., Barnes, B. V., 1994. Geo-ecosystems and bio-ecosystems. Bulletin of the Ecological Society of America 75 (1): 40–41.

Ruesch, A., Gibbs, H., 2008a. New IPCC Tier-1 Global Biomass Carbon Map For the Year 2000. Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, Oak Ridge, Tennessee, p. NA.

Ruesch, A., Gibbs, H., 2008b. New IPCC Tier-1 Global Biomass Carbon Map For the Year 2000. Carbon Dioxide Information Analysis Center, Oak Ridge, Tennessee, p. 68.

Rykiel, E. J., 1996. Testing ecological models: the meaning of validation. Ecological Modelling 90: 229–244.

Sanchez et al., 2009. Digital Soil Map of the World. Science 325: 680–681.

Schabenberger, O., Gotway, C., 2004. Statistical methods for spatial data analysis. Chapman & Hall/CRC, Boca Raton, FL, p. 524.

Schoorl, J. M., Veldkamp, A., Bouma, J., 2002. Modelling water and soil redistribution in a dynamic landscape context. Soil Science Society of America Journal 66 (5): 1610–1619.

Schuurmans, J., Bierkens, M., Pebesma, E., Uijlenhoet, R., 2007. Automatic prediction of high-resolution daily rainfall fields for multiple extents: The potential of operational radar . Journal of Hydrometeorology 8: 1204–1224.

Seijmonsbergen, A., Hengl, T., Anders, N., 2010. Semi-automated identification and extraction of geomorphological features using digital elevation data. In: Smith, M., Paron, P., Griffiths, J. (Eds.), Geomorphological Mapping: a professional handbook of techniques and applications. Developments in Earth Surface Processes. Elsevier, Amsterdam, p. in progress.

Selige, T., Böhner, J., Ringeler, A., 2006. Processing of SRTM X-SAR Data to Correct Interferometric Elevation Models for Land Surface Applications. In: Böhner, J., McCloy, K. R., Strobl, J. (Eds.), SAGA — Analyses and Modelling Applications. Vol. 115. Verlag Erich Goltze GmbH, pp. 97–104.

Shepard, D., 1968. A two-dimensional interpolation function for irregularly-spaced data. In: Blue, R. B. S., Rosenberg, A. M. (Eds.), Proceedings of the 1968 ACM National Conference. ACM Press, New York, pp. 517–524.

Sokal, R. R., Sneath, P. H. A., 1963. Principles of Numerical Taxonomy. W. H. Freeman and Company, San Francisco, p. 359.

Soluri, E., Woodson, V., 1990. World Vector Shoreline. International Hydrographic Review LXVII (1): NA.

Stein, M. L., 1999. Interpolation of Spatial Data: Some Theory for Kriging. Series in Statistics. Springer, New York, p. 247.

Steiniger, S., Bocher, E., 2009. An Overview on Current Free and Open Source Desktop GIS Developments. International Journal of Geographical Information Science 23: 1345–1370.

Szalai, S., Bihari, Z., Szentimrey, T., Lakatos, M. (Eds.), 2007. COST Action 719 — The Use of Geographic Information Systems in Climatology and Meteorology. Proceedings from the Conference on on spatial interpolation in climatology and meteorology. Office for Official Publications of the European Communities, Luxemburg, p. 264.

Temme, A. J. A. M., Heuvelink, G. B. M., Schoorl, J. M., Claessens, L., 2008. Geostatistical simulation and error propagation in geomorphometry. In: Hengl, T., Reuter, H. I. (Eds.), Geomorphometry: concepts, software, applications. Developments in Soil Science. Elsevier, pp. 121–140.

Teo, C.-K., Grimes, D., 2007. Stochastic Modelling of rainfall from satellite data. Journal of Hydrology 346 (1-2): 33–50.

Thompson, J. A., Bell, J. C., Butler, C. A., 2001. Digital elevation model resolution: effects on terrain attribute calculation and quantitative soil-landscape modeling. Geoderma 100: 67–89.

Tobler, W. R., 1970. A computer model simulation of urban growth in the detroit region. Economic Geography 46 (2): 234–240.

Triantafilis, J., Ward, W. T., Odeh, I. O. A., McBratney, A. B., 2001. Creation and Interpolation of Continuous Soil Layer Classes in the Lower Namoi Valley. Soil Science Society of America Journal 65: 403–413.

Tsoar, A., Allouche, O., Steinitz, O., Rotem, D., Kadmon, R., 2007. A comparative evaluation of presence-only methods for modelling species distribution. Diversity & Distributions 13 (9): 397–405.

Unit Geo Software Development, 2001. ILWIS 3.0 Academic user's guide. ITC, Enschede, p. 520.

Urbanek, S., Theus, M., 2008. Interactive Graphics for Data Analysis: Principles and Examples. Chapman & Hall/CRC, p. 290.

Vance, A., January 7 2009. Data Analysts Captivated by R's Power. The New York Times .

VanDerWal, J., Shooa, L. P., Grahamb, C., Williams, S. E., 2009. Selecting pseudo-absence data for presence-only distribution modeling: How far should you stray from what you know? Ecological Modelling 220: 589–594.

Venables, W. N., Ripley, B. D., 2002. Modern applied statistics with S, 4th Edition. Springer-Verlag, New York, p. 481.

Verzani, J., 2004. Using R for Introductory Statistics. Chapman & Hall, p. 432.

Wackernagel, H., 2003. Multivariate geostatistics: an introduction with applications, 2nd Edition. Springer-Verlag, p. 381.

Walter, C., McBratney, A. B., Donuaoui, A., Minasny, B., 2001. Spatial prediction of topsoil salinity in the Chelif valley, Algeria, using local ordinary kriging with local variograms versus whole-area variogram. Australian Journal of Soil Research 39: 259–272.

Walvoort, D. J. J., de Gruijter, J. J., 2001. Compositional Kriging: A Spatial Interpolation Method for Compositional Data. Mathematical Geology 33 (8): 951–966.

Webster, R., Oliver, M. A., 2001. Geostatistics for Environmental Scientists. Statistics in Practice. Wiley, Chichester, p. 265.

Wessel, P., Smith, W. H. F., 1996. A Global Self-consistent, Hierarchical, High-resolution Shoreline Database. Journal of Geophysical Research 101: 8741–8743.

Wheeler, D., Tiefelsdorf, M., 2005. Multicollinearity and correlation among local regression coefficients in geographically weighted regression. Journal of Geographical Systems 7: 161–187.

Wilson, J. P., Gallant, J. C. (Eds.), 2000. Terrain Analysis: Principles and Applications. Wiley, New York, p. 303.

Wood, J., 2008. Overview of software packages used in geomorphometry. In: Hengl, T., Reuter, H. I. (Eds.), Geomorphometry: concepts, software, applications. Developments in Soil Science. Elsevier, pp. 257–267.

Wood, J., Fisher, P. F., 1993. Assessing interpolation accuracy in elevation models. IEEE Computer Graphics and Applications 13 (2): 48–56.

Worton, B. J., 1995. Using Monte Carlo simulation to evaluate kernel-based home range estimators. Journal of Wildlife Management 4: 794–800.

Zaninović, K., Gajić-Čapka, M., Perčec-Tadić, M., et al., 2008. Klimatski atlas Hrvatske / Climate altas of Croatia 1961–1990., 1971–2000. Meteorological and Hydrological Service Republic of Croatia, Zagreb, p. 157.

Zhou, F., Huai-Cheng, G., Yun-Shan, H., Chao-Zhong, W., 2007. Scientometric analysis of geostatistics using multivariate methods. Scientometrics 73: 265–279.

Zuur, A., Ieno, E., Meesters, E., 2009. A Beginner's Guide to R. Use R. Springer, p. 228.

Index

Title: A Practical Guide to Geostatistical Mapping
Author: Tomislav Hengl
Employer: University of Amsterdam
2009 — 290 p. — 8.5 × 11" or 21.6 × 27.9 cm

URL: `http://spatial-analyst.net/book/`
Estimated manufacturing cost (standard paper, BW prints, perfect bound): $12.5
ISBN: 978-90-9024981-0

Abstract

Geostatistical mapping can be defined as analytical production of maps by using field observations, auxiliary information and a computer program that calculates values at locations of interest. The purpose of this guide is to assist you in producing quality maps by using fully-operational open source software packages. It will first introduce you to the basic principles of geostatistical mapping and regression-kriging, as the key prediction technique, then it will guide you through software tools — R+gstat/geoR, SAGA GIS and Google Earth — which will be used to prepare the data, run analysis and make final layouts. Geostatistical mapping is further illustrated using seven diverse case studies: interpolation of soil parameters, heavy metal concentrations, global soil organic carbon, species density distribution, distribution of landforms, density of DEM-derived streams, and spatio-temporal interpolation of land surface temperatures. Unlike similar books from the "use R" series, or purely GIS user manuals, this book specifically aims to bridge the gap between statistical and geographical computing. Materials presented in this book have been used for the five–day advanced training course "GEOSTAT: spatio-temporal data analysis with R+SAGA+Google Earth", that is periodically organized by the author and collaborators. Visit the book's homepage to obtain a copy of the data sets and scripts used in the exercises.

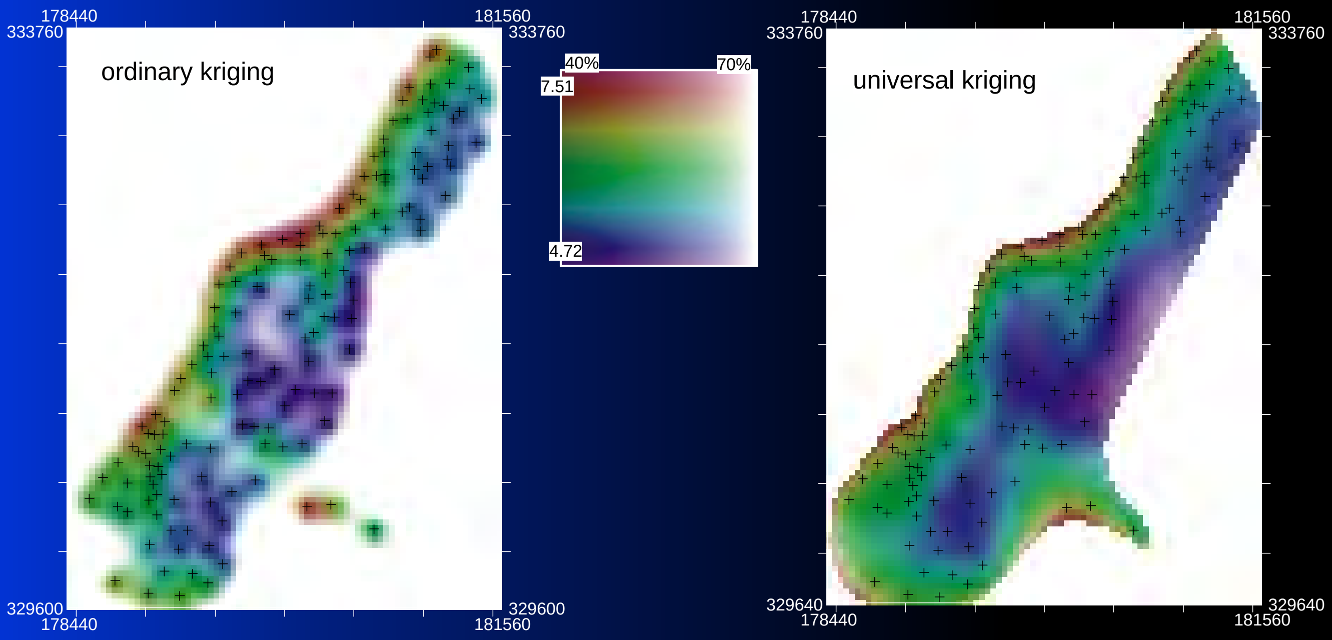


Fig. 5.19. Mapping uncertainty for zinc visualized using whitening: ordinary kriging (left) and universal kriging (right). Predicted values in log-scale.

Geostatistical mapping can be defined as analytical production of maps by using field observations, auxiliary information and a computer program that calculates values at locations of interest. The purpose of this guide is to assist you in producing quality maps by using fully-operational open source software packages. It will first introduce you to the basic principles of geostatistical mapping and regression-kriging, as the key prediction technique, then it will guide you through software tools – R+gstat/geoR, SAGA GIS and Google Earth – which will be used to prepare the data, run analysis and make final layouts. Geostatistical mapping is further illustrated using seven diverse case studies: interpolation of soil parameters, heavy metal concentrations, global soil organic carbon, species density distribution, distribution of landforms, density of DEM-derived streams, and spatio-temporal interpolation of land surface temperatures. Unlike other books from the "use R" series, or purely GIS user manuals, this book specifically aims at bridging the gaps between statistical and geographical computing.

Materials presented in this book have been used for the five-day advanced training course "GEOSTAT: spatio-temporal data analysis with R+SAGA+Google Earth", that is periodically organized by the author and collaborators.

Visit the book's homepage to obtain a copy of the data sets and scripts used in the exercises:

`http://spatial-analyst.net/book/`

Get involved: join the R-sig-geo mailing list!

Printed copies of this book can be ordered via

www.lulu.com

ISBN 978-90-9024981--0

9 789090 249810 >