

Spatial sampling with R

Dick J. Brus

2021-08-20

Contents

1	Introduction	1
1.1	Basic sampling concepts	2
1.1.1	Population parameters	6
1.1.2	Descriptive statistics versus inference about a population	7
1.1.3	Random sampling versus probability sampling	7
1.2	Design-based versus model-based approach	8
1.3	Populations used in sampling experiments	9
1.3.1	Soil organic matter in Voorst (Netherlands)	10
1.3.2	Poppy fields in Kandahar (Afghanistan)	12
1.3.3	Aboveground biomass in Eastern Amazonia	12
I	Probability sampling for estimating (sub)population parameters	17
2	Introduction to probability sampling	19
2.1	Horvitz-Thompson estimator	21
2.2	Hansen-Hurwitz estimator	23
2.3	Using models in design-based approach	24
3	Simple random sampling	27
3.1	Estimation of population parameters	30
3.1.1	Population proportion	34
3.1.2	Cumulative distribution function and quantiles	35
3.2	Sampling variance of estimator of population mean, total and proportion	37

3.3	Simple random sampling of circular plots	41
3.3.1	Sampling from a finite set of fixed circles	41
3.3.2	Sampling from an infinite set of floating circles	42
3.4	Confidence interval estimates	45
3.4.1	Confidence interval for proportion	48
4	Stratified simple random sampling	51
4.1	Estimation of population parameters	53
4.1.1	Estimation of population proportion, cumulative distribution function and quantiles	57
4.1.2	Why should we stratify?	59
4.2	Confidence interval estimate	63
4.3	Allocation of sample size to strata	64
4.4	<i>Cum-root-f</i> stratification	68
4.5	Stratification with multiple covariates	71
4.6	Geographical stratification	73
4.7	Multi-way stratification	78
4.8	Multivariate stratification	78
5	Systematic random sampling	85
5.1	Estimation of population parameters	90
5.1.1	Approximating the sampling variance of the estimator of the mean	91
6	Cluster random sampling	101
6.1	Estimation of population parameters	106
6.2	Clusters selected with probabilities proportional to size, without replacement	112
6.3	Simple random sampling of clusters	113
6.4	Stratified cluster random sampling	117
7	Two-stage cluster random sampling	121
7.1	Estimation of population parameters	124
7.2	Primary sampling units selected with probabilities proportional to size, without replacement	130
7.3	Simple random sampling of primary sampling units	132
7.4	Stratified two-stage cluster random sampling	134
8	Sampling with probabilities proportional to size	137

8.1	Probability-proportional-to-size sampling with replacement	139
8.2	Probability-proportional-to-size sampling without replacement	142
8.2.1	Systematic pps sampling without replacement	143
8.2.2	The pivotal method	145
9	Balanced and well-spread sampling	153
9.1	Balanced sampling	154
9.1.1	Balanced sample versus balanced sampling design	155
9.1.2	Unequal inclusion probabilities	156
9.1.3	Stratified random sampling	160
9.1.4	Multi-way stratification	164
9.2	Well-spread sampling	168
9.2.1	Local pivotal method	169
9.2.2	Generalised random-tessellation stratified sampling	172
9.3	Balanced sampling with spreading	176
10	Model-assisted estimation	179
10.1	Generalized regression estimator	181
10.1.1	Simple and multiple regression estimators	183
10.1.2	Penalised least squares estimation	192
10.1.3	Regression estimator with stratified simple random sampling	194
10.2	Ratio estimator	202
10.2.1	Ratio estimators with stratified simple random sampling .	206
10.2.2	Poststratified estimator	208
10.3	Model-assisted estimation using machine learning techniques . .	212
10.3.1	Predicting with a regression tree	215
10.3.2	Predicting with a random forest	219
10.4	Big data and volunteer data	223
11	Two-phase random sampling	225
11.1	Two-phase random sampling for stratification	227
11.2	Two-phase random sampling for regression	231
12	Computing the required sample size	237
12.1	Standard error of estimator	238
12.2	Length of confidence interval	239
12.2.1	Length of confidence interval for a proportion	241
12.3	Statistical testing of hypothesis	242
12.3.1	Sample size for testing a proportion	243

12.4 Accounting for design effect	245
12.5 Bayesian sample size determination	246
12.5.1 Bayesian criteria for sample size computation	248
12.5.2 Mixed Bayesian-likelihood approach	250
12.5.3 Estimation of population mean	251
12.5.4 Estimation of a population proportion	254
13 Model-based optimisation of probability sampling designs	259
13.1 Model-based optimisation of sampling design type and sample size	260
13.1.1 Analytical approach	262
13.1.2 Geostatistical simulation approach	268
13.1.3 Bayesian approach	275
13.2 Model-based optimisation of spatial strata	283
14 Sampling for estimating parameters of (small) domains	295
14.1 Direct estimator for large domains	296
14.2 Model-assisted estimators for small domains	299
14.2.1 Regression estimator	300
14.2.2 Synthetic estimator	305
14.3 Model-based prediction	307
14.3.1 Random intercept model	308
14.3.2 Geostatistical model	312
14.4 Supplemental probability sampling of small domains	319
15 Repeated sample surveys for monitoring population parameters	323
15.1 Space-time designs	323
15.2 Space-time population parameters	325
II Sampling for mapping	327
16 Introduction to sampling for mapping	329
16.1 When is probability sampling not required?	329
16.2 Sampling for simultaneously mapping and estimating means	331
16.2.1 Balanced stratified random sampling	334
16.3 Broad overview of sampling designs for mapping	336
17 Sampling on a regular grid	339
18 Spatial coverage sampling	343

18.1 Spatial infill sampling	348
19 Covariate space coverage sampling	351
19.1 Covariate space infill sampling	355
19.2 Performance of covariate space coverage sampling in random forest prediction	357
20 Conditioned Latin hypercube sampling	361
20.1 Conditioned Latin hypercube infill sampling	367
20.2 Performance of conditioned Latin hypercube sampling in random forest prediction	367
21 Spatial response surface sampling	373
21.1 Increasing the sample size	382
21.2 Stratified spatial response surface sampling	385
22 Introduction to kriging	391
22.1 Block-kriging	398
22.2 Kriging with an external drift	400
22.3 Estimating the semivariogram	405
22.3.1 Method-of-moments	405
22.3.2 Maximum likelihood	408
22.4 Estimating the residual semivariogram	409
22.4.1 Iterative method-of-moments	410
22.4.2 Restricted maximum likelihood	412
23 Model-based optimisation of grid spacing	415
23.1 Optimal grid spacing for ordinary kriging	416
23.2 Controlling the mean or a quantile of the ordinary kriging variance	418
23.3 Optimal grid spacing for block-kriging	421
23.4 Optimal grid spacing for kriging with an external drift	423
23.5 Bayesian approach	427
24 Model-based optimisation of sampling pattern	433
24.1 Spatial simulated annealing	434
24.2 Optimising the sample pattern for ordinary kriging	435
24.3 Optimising the sample pattern for kriging with an external drift	441
24.4 Model-based infill sampling	445
24.4.1 Model-based infill sampling for kriging with an external drift	446

25 Sampling for estimating the semivariogram	451
25.1 Nested sampling	452
25.2 Independent sampling of pairs of points	459
25.3 Model-based optimisation of sample pattern for semivariogram estimation	464
25.3.1 Uncertainty about semivariogram parameters	465
25.3.2 Uncertainty about the kriging variance	469
25.4 Model-based optimisation of a single sample pattern for both semivariogram estimation and prediction	474
25.5 A practical solution	482
26 Sampling for validation of maps	485
26.1 Map quality indices	486
26.1.1 Estimation of map quality indices	489
26.2 Real-world case study	491
26.2.1 Estimation of the population Mean Error and Mean Squared Error	491
26.2.2 Estimation of the standard error of the estimated population ME and MSE	494
26.2.3 Estimation of MEC	496
26.2.4 Statistical testing of hypothesis about population ME and MSE	496
27 Design-based, model-based and model-assisted approach for sampling and inference	499
27.1 Two sources of randomness	500
27.2 “Identically and independently distributed” (i.i.d.)	502
27.3 Bias and variance	507
27.4 Effective sample size	509
27.5 Exploiting spatial structure in design-based approach	515
27.6 Model-assisted versus model-dependent	516
Appendix	523
A Answers to exercises	525

Preface

Since the start of the R Series of Chapman and Hall/CRC in 2011 numerous books have been published on the statistical analysis and modelling of data using **R**. Until now no book was published in this series on how these data can be best collected. From my viewpoint this was an omission, as scientific research often starts with data collection. If the data collection is part of the project, it might be a good idea to start thinking right at the start of the project instead of after the data have been collected, to make a well-founded decision on how many data are needed and on the type of sampling design.

In the past decades numerous wall-to-wall data sets are collected by remote sensing devices such as satellites and drones. These remote sensing images are valuable sources of information of the natural environment and resources. The question may arise how useful it still is in this big data era to collect data in the field at a restricted number of sampling locations. Do we really need these data to estimate a population mean or total, for instance of the aboveground biomass or carbon stocks in the soil, or to map these study variables? In many cases the answer is that it is indeed still useful to collect sample data on the study variable, because the remote sensing images provide only proxys of the study variable. The variables derived from the remote sensing images can be related to the study variable, but we still need groundtruth data of the study variable to model this relation. By combining the wall-to-wall data of covariates and the sample data of the groundtruth we can increase the accuracy of the survey result compared to using only one of these data sources.

The handbook Sampling for Natural Resource Monitoring (SNRM) (?) presents an overview of sampling strategies for the survey of natural resources at a given point in time, as well as for how these resources can be monitored through repeated surveys. This new book can be seen as a follow-up of SNRM. In SNRM

spatial sampling designs for survey and space–time designs for monitoring are described and illustrated with notional and real world examples. Estimators for global and local quantities in space and in space–time, and for the variance of these estimators are presented. However, neither computer code for how a sample with a given design can be selected, nor code for how the estimates can be computed is presented in SNRM. This new book fills this gap.

This book describes and illustrates classical, basic sampling designs for spatial survey, as well as more recently developed, advanced sampling designs and estimators. Part I of the book is about random sampling designs for estimating a mean, total or proportion of a population or of several subpopulations. Part II focuses on sampling designs for mapping.

The computer code is written in the popular programming language **R** (?). There are several good reasons for choosing **R** as a language. First of all it is open source, giving users the right to view the source code and modify it to their needs. Second, as a result of this open source, numerous add-on packages have been developed, and this number is still increasing. Happily enough also quite a few add-on packages have been published for sampling design and analysis. All these add-on packages make the writing of computer code much more simple. Even very advanced statistical methods for sampling design and statistical analysis are now in the reach of many scientists: only a few lines of **R** code are needed to do the work. A risk is that the appliers of the packages do not fully understand the implemented statistical method. This understanding is not needed to obtain a result. For this reason I decided not to jump to the add-on packages right after the theory, but to follow a more gradual approach. First I show in as simple as possible **R** code how a sample can be selected with a given sampling design and how the (sub)population parameters can be estimated. After that I show how the same result can be obtained with an add-on package.

The target group of this book are practitioners of sample surveys, as well as students in environmental, ecological, agricultural science or any other science in which knowledge about a population of interest is collected through spatial sampling. I have added exercises to quite a few chapters, making this book suitable as a textbook for students. The answers to the exercises can be found in the appendix of this book. Large parts of the book are self-contained, requiring no prior knowledge of statistics. For the chapters in Part I on more advanced sampling designs, such as balanced sampling, and advanced estimators of (sub)population parameters (model-assisted estimators), knowledge of

matrix algebra and regression analysis is required. For the final chapters of Part II basic knowledge of geostatistics is required. This knowledge is also needed for some chapters in Part I. For that reason I have added a chapter introducing the basics of geostatistics (Chapter @ref{Introkriging}).

A gitbook version of this book (html files) is available at <https://git.wageningen.nl/brus003/spatialsamplingwithr/tree/master>. This is also the place where you can report errata and comment on text and/or **R** code.

Acknowledgements

In 2006 our handbook Sampling for Natural Resource Monitoring (SNRM) was published (?). Soon after this milestone Jaap retired from Wageningen University and Research (WUR). Now I arrived in the aftermath of my career at WUR. Since a couple of years I was thinking of a revision of our handbook, to repair errors and to include new developments in sampling design. Then I realized that to increase the impact of our book, it might be a better idea to write a new book, showing with computer code how the sampling designs can be implemented, and how the sample data can be used in statistical inference.

A nice side effect of the publication of SNRM was that I was asked to give sampling courses at many places in the world: China, Ethiopia, Uzbekistan, Australia and various countries in the European Union. I have very good memories of these courses, they made my life as a scientist very joyful. For these courses I wrote numerous scripts with computer code, using the popular programming language **R** (?). My naive idea was that all I had to do is to bundle these **R** scripts into an Rmarkdown document (?), and add some text explaining the theory and the **R** code. As usual, it appeared to be much more work than expected, but I am very happy that I was able to finish the job just before my retirement. My experience as a statistical consultant is that many researchers pay little attention to the method for data collection. Too many researchers start thinking when the data are there. Often I had to conclude that the way the data were collected was suboptimal, or even unsuitable for their aim. I hope that this new book may help researchers, practitioners and students to implement proper sampling designs, tailored at their problems at hand, so that valuable data are collected that can be used to answer the research questions.

I could not have written this book without the help of many fellow researchers. First, I am very grateful for the support I received from the authors of various

packages used in this book: Thomas Lumley for his support with package **survey**, Yves Tillé and Alina Gabriela Matei with package **sampling**, Anton Grafström with package **Balancedsampling**, Giulio Barcaroli and Marco Ballin with package **SamplingStrata**, Andreas Hill and Alex Massey with package **Forestinventory**, and Martins Liberts with package **surveyplanning**. No need to say that I am responsible for all shortcomings of the **R** code.

Second, I would like to thank the following researchers for their valuable comments on (parts) of the book: Gerard Heuvelink (Wageningen University and ISRIC World Soil Information, Netherlands), David Rossiter (Cornell University, USA and ISRIC World Soil Information, Netherlands), Yuha Heikkinen (Luke, Natural Resources Institute, Finland), Steve Stehman (SUNY College of Environmental Science and Forestry, USA), Anton Grafström (Swedish University of Agricultural Sciences), Dennis Walvoort (Wageningen University and Research, Netherlands) and Ben Marchant (British Geological Survey, United Kingdom). Dennis Walvoort also was very supportive with the writing of various **R** scripts.

Finally, I would like to thank Alexandre Wadoux (University of Sydney) for preparing the data set of aboveground biomass and numerous environmental and climatological covariates of Eastern Amazonia (Brazil); Coen Bussink (UN Organization on Drugs and Crime) for giving permission to use data on the occurrence of opium poppy fields in Kandahar (Afghanistan), Akmal Akramkhanov for providing the data set with measurements of the salinity of soil at a farm which is part of a regional Cotton Research Station in Khorezm (Uzbekistan). These data were collected in the ZEF/UNESCO Landscape Restructuring project in Khorezm province, with financial support by the German Ministry for Education and Research (BMBF; project number 0339970A); Lin Yang (Nanjing University) for giving permission to use the data on soil organic matter concentration in Xuancheng (China) collected in a project supported by the National Natural Science Foundation of China (Project No 41471178, 41431177); Hailu Shiferaw (Ethiopian Agricultural Transformation Agency) to allow me to use the soil organic matter data in three woredas in Ethiopia; Siegfried Hofman (Flemish Institute for Technological Research) for giving permission to use the nitrate-N data of several agricultural fields in Flanders (Belgium); and Budiman Minasny (University of Sydney) for giving permission to use the raster maps with terrain attributes in Hunter Valley (Australia).

Chapter 1

Introduction

This book is about sampling for spatial *surveys*. A survey is an inventory of an object of study about which statements will be made, referred to as the population of interest or target population, by collecting data on the population. Examples are a survey of the organic carbon stored in the soil of a country, the water quality of a lake, the wood volume in a forest, the total annual yield of rice in a country, etc. So this book is about *observational research*, not about experiments. In experiments observations are done under controlled circumstances, think of an experiment on crop yields as a function of application rates of fertilizer. Several levels of fertilizer application rate are chosen and randomly assigned to experimental plots. In observational research factors that influence the study variable are not controlled. This implies that in observational research no conclusions can be drawn on causal relations.

If the whole population is observed this is referred to as a *census*. In general we cannot afford such a census. Only some parts of the population are selected and properties of the study variable are observed (measured) on these selected parts only. Such a survey is referred to as a *sample survey*. The observations are subsequently used to derive characteristics of the whole population. For instance, to estimate the wood volume in a forest, we cannot afford to measure the wood volume of every tree in the forest. Instead, some trees are selected, the wood volume of these trees is measured, and based on these measurements the total wood volume in the forest is estimated.

1.1 Basic sampling concepts

In this book the populations of interest have a spatial dimension. In selecting parts of such populations for observation we may account for the spatial coordinates of the parts, but this is not strictly needed. Examples of spatial sampling designs are designs selecting sampling units that are spread out throughout the study area, often leading to more precise estimates of the population mean or total.

Two types of populations can be distinguished: discrete and continuous populations. *Discrete populations* consist of discrete, natural objects, think of trees, agricultural fields, lakes etc. These objects are referred to as *population units*. The total number of population units in a discrete population is finite. A finite spatial population of discrete units can be denoted by $\mathcal{U} = \{u(\mathbf{s}_1), u(\mathbf{s}_2), \dots, u(\mathbf{s}_N)\}$, with $u(\mathbf{s}_k)$ the unit located at \mathbf{s}_k , where \mathbf{s} is a vector with spatial coordinates. The population units naturally serve as the *elementary sampling units*. In this book the spatial populations are two-dimensional, so a vector \mathbf{s} has two coordinates, Easting and Northing.

Other populations may, for the purpose of sampling, be considered as a physical continuum, e.g. the soil in a region, the water in a lake, the crop on a field if interest lies in crop properties per areal unit of the field¹. Such continuous, spatial populations can be denoted by $\mathcal{U} = \{u(\mathbf{s}), \mathbf{s} \in \mathcal{A}\}$, with \mathcal{A} the study area. Discrete objects that can serve as elementary sampling unit do not exist in such *continuous populations*. We must define these elementary sampling units. The elementary sampling units can be areal units, e.g. 10 m by 10 m squares or circular plots with a radius of 5 m, or “points”, i.e. units that have an area that is so small compared to the area of the population that the area can be ignored.

In this book a population unit and an elementary sampling unit can be an individual object of a discrete population, as well as a point or areal sampling unit of a continuous population.

The size and geometry of the elementary units used in sampling a continuous population is referred to as the sample support. The total number of elementary sampling units in a continuous population can be finite, e.g. all 25 m by 25 m (disjoint) raster cells in an area (raster cells in Figure 1.1), or infinite, e.g. all points in an area, or all squares or circular plots with a given radius that are allowed to overlap in an area (circles in Figure 1.1).

¹If interest lies in properties per plant, the population is discrete.

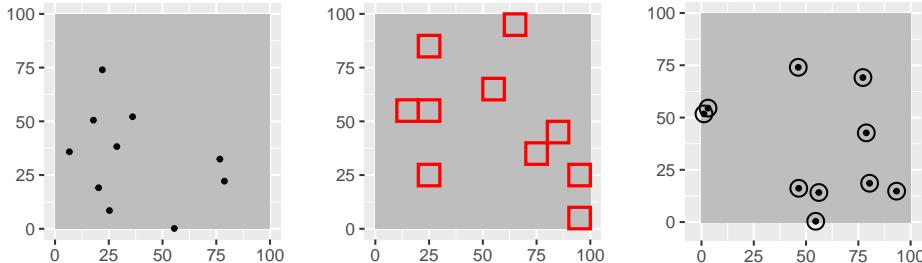


Figure 1.1: Three sample supports: points, squares and circles. With disjoint squares the population is finite. With points, and squares or circles that are allowed to overlap as sample support the population is infinite.

With areal elementary sampling units, ideally the selected elementary units are exhaustively observed, so that a measurement of the total or mean of the study variable within an areal unit is obtained, think for instance of the total aboveground biomass. In some cases this is not feasible, think for instance of measuring the mean of some soil property in squares of 25 m x 25 m. In this case from each selected square a sample of points is selected, and the measurement is done on the selected points. These measurements at points are used to estimate the mean of the squares. ? introduced the concept of a response design as “the protocol used to determine the reference condition of an element of the population”. So in this case the response design is the sampling design and estimator for the mean of the areal units.

Ideally the sample support is constant, but in some situations a varying sample support cannot be avoided. Think, for instance, of square sampling units in an irregularly shaped study area. Near the border of the study area there are squares that cross the border. The part of a square that falls outside the study area is not observed. So the support of the observation on these sampling units near the border is smaller than for the squares in the interior of the study area. See also Section 3.3.

To sample a finite spatial population, the population units are listed in a data frame. This data frame contains the spatial coordinates of the population units, and other information needed for selecting sampling units according to a specific design. Think, for instance, of the labels of more or less homogeneous subpopulations (used as strata in stratified random sampling, see Chapter 4), and the labels of clusters of population units, for instance, all units in a polygon

of a map (used in cluster random sampling, see Chapter 6). Besides, if we have information about covariates possibly related to the study variable, which we would like to use in selecting the population units, these covariates are added to the list. The list used for selecting sampling units is referred to as the *sampling frame*.

In this book also continuous populations are sampled using a list as a sampling frame. The infinite population is discretised by the nodes of a fine square grid. The grid nodes are listed in the sampling frame. So the infinite population is represented by a finite list of points that are the centers of square grid cells. The advantage of this is that existing **R** packages for sampling of finite populations can also be used for sampling infinite populations.

If the disjoint square grid cells are the elementary sampling units (sample support is a square) the population is finite, and the grid cells can be selected through selection of their centers that are listed in the sampling frame. The grid cells can be selected without or with replacement.

If the elementary sampling units are points (sample support is a point), the population is infinite. In this case sampling of points can be implemented by a two-step approach. In the first step grid cells are selected without or with replacement, and in the second step one or more points are selected within the selected grid cells. Figure 1.2 is an illustration of this two-step approach for simple random sampling of points from a discretised infinite population. Ten grid cells are selected by simple random sampling with replacement. Every time a grid cell is selected one point is randomly selected from that grid cell. Note that a grid cell can be selected more than once, so that more than one point will be selected from that grid cell. Note also that we may select a point that falls outside the boundary of the study area. This is actually the case with one grid cell in Figure 1.2. These points outside the study area are discarded and replaced by a randomly selected new point inside the study area. Finally, note that near the boundary there are small areas that are not covered by a grid cell, so that no points can be selected in these areas. It is important that the discretisation grid is fine enough to keep the discretisation error so small that it can be ignored. The alternative is to extend the discretisation grid beyond the boundaries of the study area so that the full study area is covered by grid cells.

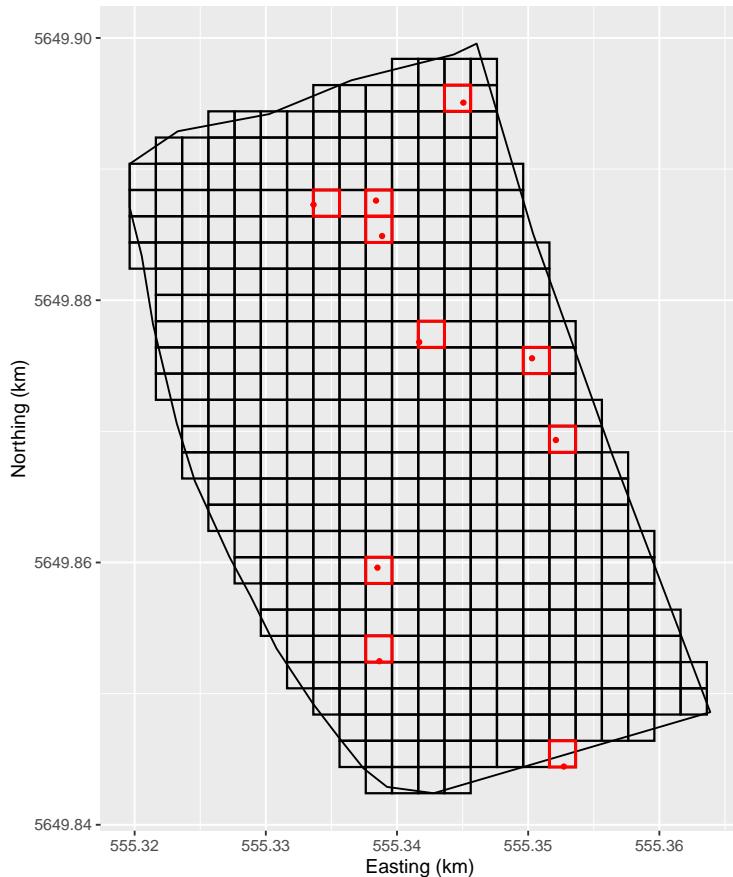


Figure 1.2: Sampling of points from discretised infinite population. The grid cells are randomly selected with replacement. Each time a grid cell is selected a random point is selected random from that grid cell.

1.1.1 Population parameters

The sample data are used to estimate characteristics of the whole population, e.g. the population mean or total, some quantile e.g. the median or 90th percentile, or even the entire cumulative frequency distribution.

A finite population total is defined as

$$t(z) = \sum_{k \in \mathcal{U}} z_k = \sum_{k=1}^N z_k , \quad (1.1)$$

with N the number of population units, and z_k the study variable for population unit i . A finite population mean is defined as a finite population total divided by N . An infinite population total is defined as an integral of the study variable over the study area:

$$t(z) = \int_{\mathbf{s} \in \mathcal{U}} z(\mathbf{s}) \, d\mathbf{s} . \quad (1.2)$$

An infinite population mean is defined a finite population total divided by the area, A , covered by the population.

A finite population proportion is defined as the population mean of an 0/1 indicator y with value 1 if the condition is satisfied, and 0 otherwise:

$$p = \frac{\sum_{k \in \mathcal{U}} y_k}{N} . \quad (1.3)$$

A cumulative distribution function (CDF) is defined as

$$F(z) = \sum_{x \leq z} f(x) , \quad (1.4)$$

with $f(x)$ the proportion of population units whose value for the study variable z equals x . A population quantile, for instance the population median or the population 90th percentile is defined as

$$q_p = F^{-1}(p) , \quad (1.5)$$

where p is a number between 0 and 1 (e.g 0.5 for the median, 0.9 for the 90th percentile), and $F^{-1}(p)$ is the smallest value of the study variable z satisfying $F(z) \geq p$.

In surveys of spatial populations the aim can also be to make a map of the population.

1.1.2 Descriptive statistics versus inference about a population

As we have observed only a (small) part of the population, we are uncertain about the population parameter estimates and map. By using statistical methods we can quantify how uncertain we are about these results. In decision making it can be important to take this uncertainty into account. An example is a survey of water quality. In Europe the concentration levels of nutrients are regulated in the European Water Framework Directive. To test whether the mean concentration of a nutrient complies with its standard, it is important to account for the uncertainty in the estimated mean. When the estimated mean is just below the standard, there is still a large probability that the population mean exceeds the standard. This example shows that it is important to distinguish computing descriptive statistics from characterizing the population using the sample data. For instance, we can compute the sample mean (average of the sample data) without error, but if we use this sample mean as an *estimate* of the population mean, there is certainly an error in this estimate.

1.1.3 Random sampling versus probability sampling

Many sampling methods are available. At the highest level one may distinguish random from non-random sampling methods. In random sampling a subset of population units is randomly selected from the population, using a random number generator. In non-random sampling no such (pseudo) random number generator is used. Examples of non-random sampling are convenience sampling e.g. along roads, arbitrary sampling i.e. sampling without a specific purpose in mind, and targeted sampling e.g. at sites suspected of soil pollution.

In the literature the term random sampling is often used for arbitrary sampling, i.e. sampling without a specific purpose in mind. To avoid confusion the term *probability sampling* is used for random sampling using a (pseudo) random number generator, so that for any unit in the population the probability of selecting that unit is known. More precisely, a probability sample is a sample from a

Table 1.1: Statistical approaches for sampling and inference.

Approach	Sampling	Inference
Design-based	Probability sampling	Based on sampling distribution (no model-used)
Model-based	Probability sampling not required	Based on statistical model

population such that every unit of the population has a positive probability of being included in the sample. Besides, these *inclusion probabilities* must be known, at least for the selected units, as they are needed in estimation. This is explained in following chapters.

1.2 Design-based versus model-based approach

The choice between probability or non-probability sampling is closely connected with the choice between a design-based or model-based approach for sampling and statistical inference (estimation, hypothesis testing). The difference between these two approaches is a rather technical subject, and therefore not to discourage you already in this very first chapter, I keep it short. In Chapter 27 I elaborate on the fundamental difference of these two approaches and a third approach, the model-assisted approach, which can be seen as a compromise of the design-based and model-based approach.

In the design-based approach units are selected by probability sampling (Table 1.1). Estimates are based on the inclusion probabilities of the sampling units as determined by the sampling design (design-based inference). No model is used in estimation. On the contrary, in a model-based approach a statistical model is used in prediction, i.e. a model with a random error term, for instance a regression model. As the model already contains a random error term, probability sampling is not required in this approach.

Which statistical approach is best largely depends on the aim of the survey, see ? and ?. Broadly speaking the following aims can be distinguished:

1. To *estimate parameters* (mean, total, proportion, percentile) for the population.
2. To *estimate parameters* (mean, total, proportion, percentile) for several

subpopulations.

3. To map the study variable².

When the aim is to map the study variable, a model-based approach is the most natural option. This implies that for this aim probability sampling is not required. For estimating (sub)population parameters in principle both approaches are suitable. The more subpopulations are distinguished, the more attractive a model-based approach becomes. If the units are selected by probability sampling, then estimates of the population parameters can be obtained by design-based or model-based inference. This flexibility can be attractive, for instance when the sample size is rather small for model building. When the sampling units are not selected by probability sampling, model-free, design-based estimation is impossible, and model-based estimation is the only option.

1.3 Populations used in sampling experiments

In this book various data sets are used to illustrate the sampling designs. Three data sets, Voorst, Kandahar and Eastern Amazonia, are exhaustive, i.e. for all population units data of the study variable and ancillary data are available. Two exhaustive data sets, Voorst in the Netherlands and Kandahar in Afghanistan, are obtained through simulation, i.e. by drawing numbers from a probability distribution. Sample data from these two study areas are used to calibrate a statistical model. This model is subsequently used to simulate values of the study variable for all population units. Voorst actually is an infinite population of points. However, this study area is discretised by a fine grid, and the study variable, the soil organic matter concentration, is simulated for all nodes of this discretisation grid. Kandahar is a finite population consisting of 965 squares of size 5 km × 5 km. The study variable is the area cultivated with poppy. Eastern Amazonia is a map in raster format, with a resolution of 1 km × 1 km. The study variable is the aboveground biomass as derived from remote sensing images. The aboveground biomass value of a raster cell is treated as the average biomass of that raster cell.

The exhaustive data sets are used in the first part of this book on probability sampling for estimating population means and totals. By taking the population as the reality, we know the population mean and total. Also, for any randomly

²A map of the study variable is obtained by predicting the study variable at the points of a very fine grid discretising the study area.

selected sample from this population, the study variable values for the selected sampling units are known, so that we can *estimate* the population mean or total from this sample. The estimated mean (total) can then be compared with the population mean (total). The difference between these two is the *sampling error* in the estimated mean (total). This opens up the possibility of repeating the selection of random samples with a given sampling design a large number of times, estimating the population mean (total) for every sample, so that a frequency distribution of the estimated population mean is obtained. Ideally, the mean of this frequency distribution, referred at as the *sampling distribution*, is equal to the population mean (mean sampling error equals zero), and the variance of the estimated means is small. Another advantage is that sampling designs can be compared on the basis of the sampling distribution, for instance the sampling distributions of stratified random sampling and simple random sampling, to evaluate whether the stratification leads to more accurate estimates of the population mean.

Besides, various data sets are used with data for a sample of population units only. These data sets are described at places where they are first used.

1.3.1 Soil organic matter in Voorst (Netherlands)

The study area of Voorst is located in the eastern part of the Netherlands. The size of the study area is 6 km by 1 km. At 132 points samples of the topsoil were collected by graduate students of Wageningen University, which were analyzed in the laboratory on soil organic matter (SOM) concentrations (in g per kg). The map is created by conditional geostatistical simulation of natural logarithms of SOM on a 25 m by 25 m grid, followed by backtransformation, using a linear mixed model with spatially correlated residuals and combinations of soil type and land use as a qualitative predictor (factor). Figure 1.3 shows the simulated map of SOM.

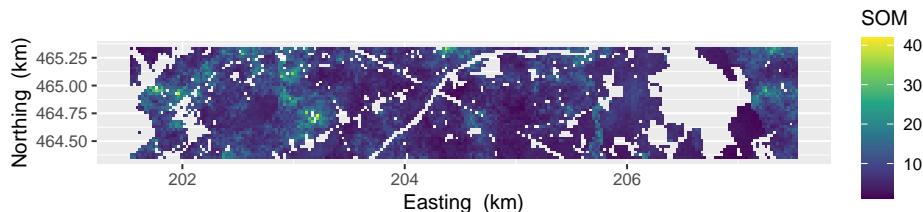


Figure 1.3: Simulated soil organic matter concentration (g/kg) in Voorst.

The histogram of the simulated values at all 7528 grid cells shows that SOM is skewed to the right (Figure 1.4).

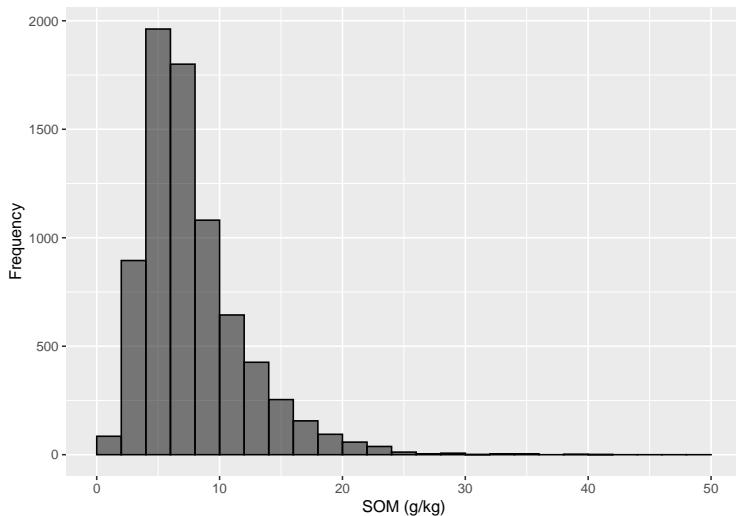


Figure 1.4: Histogram of simulated soil organic matter concentration (g/kg) in Voorst.

Summary statistics are:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.216	4.982	6.812	7.814	9.571	41.991

The ancillary information consist of a map of soil classes and a land use map, which are combined to five soil-land use combinations (Figure 1.5). The first letter in the labels for the combinations stands for the soil type: B for beekeerdgrond (sandy wetland soil with gleyic properties), E for enkeerdgrond (sandy soil with thick anthropogenic humic topsoil), P for podzols (sandy soil with eluviated horizon below the topsoil), R for river clay soil, and X for sandy soils. The second letter is for land use: A for agriculture (grassland, arable land), and F for forest.



Figure 1.5: Soil-land use combinations in study area Voorst (Netherlands).

1.3.2 Poppy fields in Kandahar (Afghanistan)

Cultivation of poppy for opium production is a serious problem in Afghanistan. The United Nations Organization on Drugs and Crime (UNODC) monitors the area cultivated with poppy through detailed analysis of areal photographs and satellite images. This is laborious, and for that reason this analysis is restricted to a probability sample of 5 km by 5 km squares. These sample data are then used to estimate the total poppy area (?).

In 2014 the poppy area of 83 squares in the province of Kandahar (Afghanistan) was determined, as well as the agricultural area of all 965 squares in this province. These data were used to simulate a map of poppy area per 5 km by 5 km square. The map is simulated with an ordinary kriging model for the logit transform of the proportion of the agricultural area within a 5 km by 5 km square cultivated with poppy. For privacy reasons the field was simulated *unconditionally* on these sample data. Figure 1.6 shows the map with the agricultural area in hectares per 5 km \times 5 km square, and the simulated poppy area in hectares, per square. The histogram of the simulated poppy area per square shows very strong positive skew (Figure 1.7). For 375 squares the simulated poppy area was smaller than 1 ha.

1.3.3 Aboveground biomass in Eastern Amazonia

This data set consists of data on the aboveground live woody biomass (AGB) in megatons per ha (?). A rectangular area of 1642 km by 928 km in Eastern Amazonia (Brazil) was selected from this data set, and aggregated to a map with a resolution of 1 km \times 1 km. Besides, a stack of five ecologically relevant covariates of the same spatial extent was prepared, being MODIS long term mean of short-wave infrared radiation (SWIR2), Primary Production in kg C per m² (Terra_PP), average precipitation in driest month in mm (Prec_dm), Elevation in m, and Clay content in g per kg soil. All covariates were either



Figure 1.6: Agricultural area and simulated area cultivated with opium poppy, in hectares per 5 km by 5 km squares in Kandahar (Afghanistan).

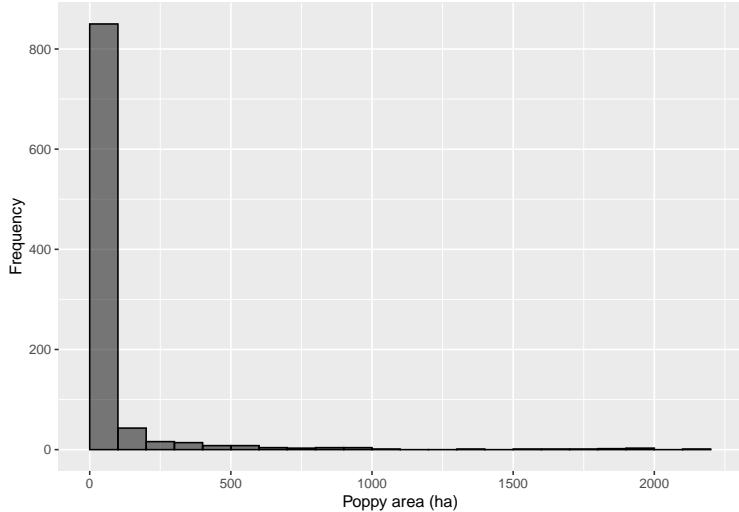


Figure 1.7: Histogram of simulated poppy area (ha) per 5 km by 5 km square in Kandahar.

resampled using bilinear interpolation, or aggregated to conform with the grid of the above-ground biomass map. Figure 1.8 shows a map of AGB and SWIR2.

Figure 1.9 shows a matrix of two-dimensional density plots of aboveground biomass and the five covariates, made with function `ggpairs` of **R** package **GGally** (?). The covariate with the strongest correlation with AGB is SWIR2. The Pearson correlation coefficient with AGB is -0.80. The relation does not look linear. The correlation of AGB with the covariates Terra_PP and Prec_dm is weakly positive. All correlations are significant, but this is not meaningful because of the very large number of data used in computing the correlation coefficients.

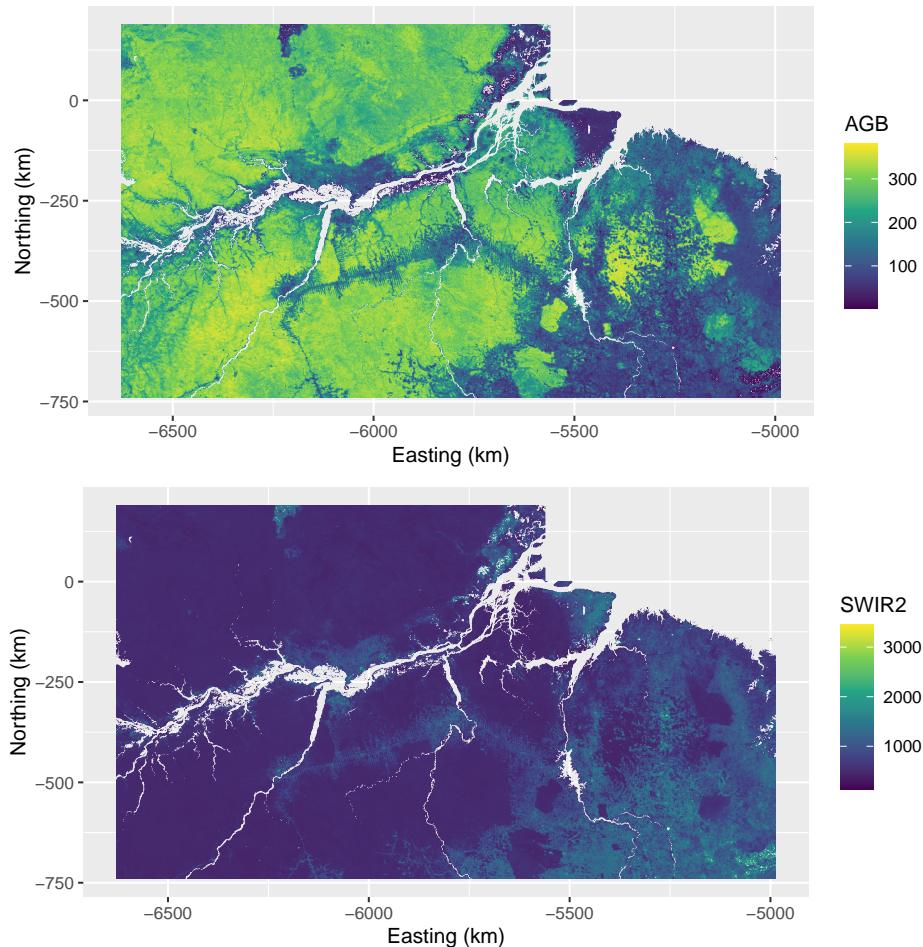


Figure 1.8: Aboveground biomass in megatons per ha (AGB), and short-wave infrared radiation (SWIR2) of Eastern Amazonia (Brazil).

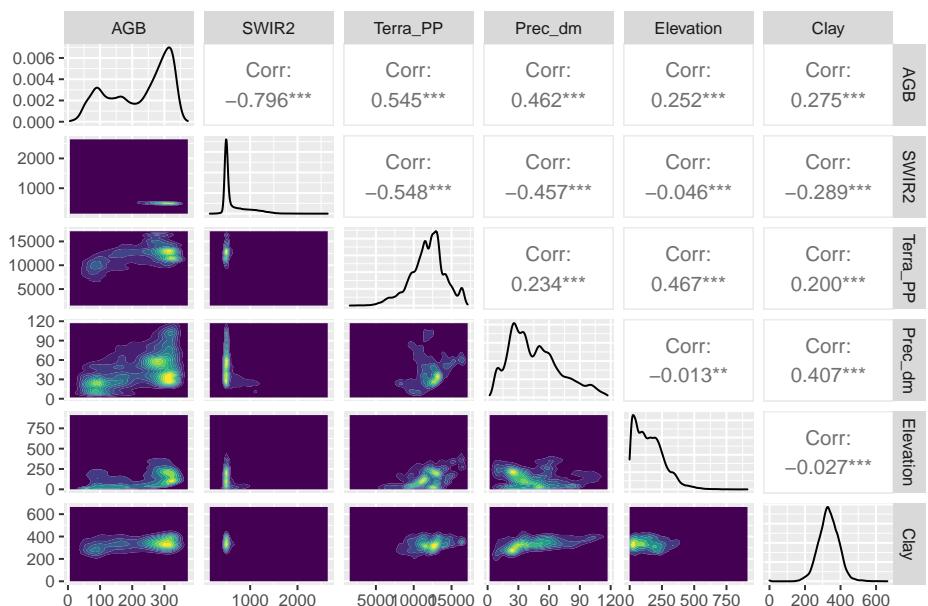


Figure 1.9: Matrix of two-dimensional density plots of aboveground biomass (AGB) and five covariates of Eastern Amazonia (Brazil).

Part I

Probability sampling for estimating (sub)population parameters

Chapter 2

Introduction to probability sampling

To estimate population parameters like the mean or the total, *probability sampling* is most appropriate. Probability sampling is random sampling using a random number generator such that all population units have a probability larger than zero of being selected, and that these probabilities are known for at least the selected units.

The probability that a unit is included in the sample, in short the inclusion probability of that unit, can be calculated as the sum of the selection probabilities over all samples that can be selected with a given sampling design and that contain this unit. In formula:

$$\pi_k = \sum_{\mathcal{S} \ni k} p(\mathcal{S}) , \quad (2.1)$$

where $\mathcal{S} \ni k$ indicates that the sum is over all samples that contain unit k , and $p(\mathcal{S})$ is the selection probability of sample \mathcal{S} . $p(\cdot)$ is called the *sampling design*. It is a function that assigns a probability to every possible sample (subset of population units) that can be selected with a given sample selection scheme (sampling algorithm). For instance, consider the following sample selection scheme from a finite population of N units:

-
1. Select with equal probability $1/N$ a first unit.
 2. Select with equal probability $1/(N - 1)$ a second unit from the remaining $N - 1$ units.
 3. Repeat this until an n th unit is selected with equal probability from the $N - (n - 1)$ units.

This is a selection scheme for simple random sampling without replacement. With this scheme the selection probability of any sample of n units is $1/\binom{N}{n}$, and zero for all other samples. There are $\binom{N-1}{n-1}$ samples of size n . The inclusion probability of each unit k therefore is $\binom{N-1}{n-1}/\binom{N}{n} = \frac{n}{N}$. The sampling design plays a key role in the design-based approach as it determines the sampling distribution of random quantities computed from a sample such as the estimator of the population mean, see Section 2.1. The number of selected population units is referred to as the *sample size*¹.

A common misunderstanding is that with probability sampling the inclusion probabilities must be equal. Sampling with unequal inclusion probabilities can be more efficient than with equal probabilities. Unequal probability sampling is no problem as long as these inclusion probabilities are known and proper formulas are used for estimation, see Section 2.1.

There are many schemes for selecting a probability sample. The following sampling designs are described and illustrated in this book:

1. Simple random sampling.
2. Stratified random sampling.
3. Systematic random sampling.
4. Cluster random sampling.
5. Two-stage cluster random sampling.
6. Sampling with probabilities proportional to size.
7. Balanced and well-spread sampling.
8. Two-phase random sampling.

The first five sampling designs are basic sampling designs. Implementation of these designs is rather straightforward², as well as the associated estimation of

¹In sampling with replacement the number of unique populations units in the sample can be smaller than the sample size.

²Although a proper implementation of cluster random sampling can be quite difficult, see Chapter 6.

the population mean, total or proportion, and their sampling variance. The final three sampling designs are more advanced, and require more knowledge of sampling theory and statistics, such as linear regression.

Exercises

1. Suppose a researcher selects a sample of points from a study area by throwing darts on a map depicting the study area. Is the resulting sample a probability sample? If not, why not?
2. Suppose we have a population of N units, numbered $1 \dots N$. Can you think of a simple but proper way of selecting a probability sample of n units? What is the inclusion probability of any given unit?

2.1 Horvitz-Thompson estimator

For any probability sampling design the population total can be estimated as a weighted sum of the observations (measurements) of the study variable on the selected population units:

$$\hat{t}_\pi(z) = \sum_{k \in \mathcal{S}} w_k z_k , \quad (2.2)$$

with \mathcal{S} the sample, z_k the observed study variable for unit k , and w_k the design weight attached to unit k :

$$w_k = \frac{1}{\pi_k} , \quad (2.3)$$

with π_k the inclusion probability of unit k . The estimator³ of Equation (2.2) is referred to as the Horvitz-Thompson estimator or π estimator. The z_k/π_k values are referred to as the π -expanded values. The z value of unit k in the sample is multiplied by the reciprocal of the inclusion probability of that unit, and the sample sum of these π -expanded values is used as an estimator of the population

³An *estimator* is not the same as an *estimate*. An estimate is a single value calculated from the data of a selected sample, whereas an estimator is a *random variable* that is a function of the sample data, and has a probability distribution.

total. The inclusion probabilities are determined by the type of sampling design and the sample size.

For infinite populations, think of points in a continuous population, the same estimator for the population total can be used, but special attention must then be paid to the inclusion probabilities. Suppose the infinite population is discretised by a fine grid of N nodes, and a simple random sample of n nodes is selected. The inclusion probabilities of the grid nodes is then n/N . However, constraining the sampling points to the nodes of the discretisation grid is not needed, and even undesirable. To account for the infinite number of points in the population we may adopt a two-step approach, see Figure 1.2. In the first step n grid nodes are selected by simple random sampling *with replacement*. In the second step one point is selected fully randomly from the grid cells with the grid nodes selected in the first step at their centers. If a grid cell is selected more than once, more points are selected in that grid cell. With this selection procedure the inclusion probability density is n/A , with A the area of the study area. This inclusion probability density equals the expected number of sampling points per unit area, e.g. the expected number of points per ha or per m^2 . The inclusion probability density can be interpreted as the sampling intensity⁴.

The π estimator for the *mean* of a finite population, \bar{z} , is simply the π estimator for the total, divided by the total number of units in the population, N :

$$\hat{\bar{z}}_\pi = \frac{1}{N} \sum_{k \in S} \frac{1}{\pi_k} z_k . \quad (2.4)$$

For infinite populations discretised by a finite set of points the same estimator can be used.

For infinite populations the population total can be estimated by multiplying the estimated population mean by the area of the population A :

$$\hat{t}_\pi(z) = A \hat{\bar{z}}_\pi . \quad (2.5)$$

The π estimator can be worked out for the different types of sampling design listed above, by inserting the inclusion probabilities as determined by the sampling design. For simple random sampling this leads to the unweighted sample

⁴The integral of the inclusion probability density over the study area equals the (expected) sample size n , not one, and for that reason, to avoid confusion with a probability density, I prefer the term sampling intensity.

mean (see Chapter 3), and for stratified simple random sampling the π estimator is equal to the weighted sum of the sample means per stratum, with weights equal to the relative size of the strata (see Chapter 4).

2.2 Hansen-Hurwitz estimator

In sampling finite populations, units can be selected with or without replacement. In sampling with replacement after each draw the selected unit is replaced. As a consequence a unit can be selected more than once. Sampling with replacement is less efficient than sampling without replacement. If a population unit is selected in a given draw, there is no additional information in this unit if it is selected again. One reason that sampling with replacement is still used is that it is more easy to implement.

The most common estimator used for sampling with replacement is the Hansen-Hurwitz estimator, referred to as the pwr estimator by ?. With direct unit sampling, i.e. sampling of individual population units, the pwr estimator is

$$\hat{t}_{\text{pwr}}(z) = \frac{1}{n} \sum_{k \in \mathcal{S}} \frac{z_k}{p_k}, \quad (2.6)$$

with p_k the *draw-by-draw selection probability* of population unit k . The acronym pwr stands for p -expanded with replacement. In the pwr estimator the observations on the selected units are expanded by the draw-by-draw selection probability. For instance, in simple random sampling with replacement the draw-by-draw selection probability p of each unit is $1/N$. If we select only one unit k , the population total can be estimated by the observation on that unit divided by p , $\hat{t}(z) = z_k/p_k = Nz_k$. If we repeat this n times, this results in n estimated population totals. The pwr estimator is the average of these n elementary estimates. If a unit occurs multiple times in the sample \mathcal{S} , this unit provides multiple elementary estimates of the population total.

A sample obtained by sampling with replacement is referred to as an *ordered sample* (?). Selecting the distinct units from this ordered sample results in the *set-sample*. Instead of using the ordered sample in the pwr estimator, we may use the set-sample in the π estimator. This requires computation of the inclusion probabilities for with replacement sampling. For instance, for simple random sampling with replacement the inclusion probability of each unit equals $1 - (1 - \frac{1}{N})^n$, with n the number of draws. This probability is smaller than

n/N , the inclusion probability for simple random sampling without replacement. There is no general rule which estimator is most accurate (?). In this book I only use the pwr estimator for sampling with replacement.

Sampling with replacement can also be applied at the level of clusters of population units as in cluster random sampling and two-stage cluster random sampling. If the clusters are selected with probabilities proportional to their size and with replacement, estimation of the population mean, proportion or total is rather simple. This is a second reason why sampling with replacement can be attractive. With cluster sampling the pwr estimator is

$$\hat{t}_{\text{pwr}}(z) = \frac{1}{n} \sum_{j \in \mathcal{S}} \frac{t_j(z)}{p_j}, \quad (2.7)$$

with $t_j(z)$ the total of the cluster selected in the j th draw. If not all population units of a selected cluster are observed, but a sample of population units from a cluster only as in two-stage cluster random sampling, the cluster totals $t_j(z)$ are replaced by the estimated cluster totals $\hat{t}_j(z)$.

2.3 Using models in design-based approach

Design-based estimates of population parameters such as the population mean, population total or population proportion (areal fraction) are model-free: no use is made of a model for the spatial variation of the study variable. However, such a model can be used to optimise the probability sampling design. In Chapter 13 I describe how a model can be used to compare alternative sampling designs at equal costs or equal precision to evaluate which sampling design performs best, to optimise the sample size(s) given a requirement on the precision of the estimated population parameter, or to optimize the spatial strata for stratified random sampling.

A model of the spatial variation can also be used at a later stage, after the data are collected, in estimating the population parameter of interest. If one or more ancillary variables that are related to the study variable are available, these variables can be used in estimation to increase the accuracy. This leads to alternative estimators, such as the simple and multiple regression estimator, ratio estimator and poststratified estimator (Chapter 10). These estimators together are referred to as model-assisted estimators. In model-assisted estimation the

inclusion probabilities as determined by the random sampling design, play a key role, but besides, modelling assumptions about how the population might have been generated are used to work out an efficient estimator. The role of the model in the model-assisted approach is fundamentally different from its role in the model-based approach. This is explained in Chapter 27.

For novices in geostatistics Chapters 10 and 13 can be quite challenging, and I recommend to skip these Chapters first, to return to them after reading the introductory Chapter on geostatistics (Chapter 22).

Chapter 3

Simple random sampling

Simple random sampling is the most basic form of probability sampling. There are two subtypes:

1. Simple random sampling with replacement (SIR).
2. Simple random sampling without replacement (SI).

This distinction is irrelevant for infinite populations. In with replacement sampling a population unit may be selected more than once.

In **R** a simple random sample can be selected with or without replacement by the function `sample.int` from the `base` package. For instance, a simple random sample without replacement of 10 units from a population of 100 units labeled as 1,2, ... ,100, can be selected by

```
sample.int(100, size=10, replace=FALSE)
```

```
[1] 83 12 99 45 21 23 25 73 86 87
```

The number of units in the sample is referred to as the sample size ($n = 10$ in the code chunk above). Use argument `replace = TRUE` to select a simple random sample with replacement.

When the spatial population is continuous and infinite, as in sampling points

from an area, the infinite population is discretised by a very fine grid. Discretisation is not strictly needed (we could also sample points directly), but it is used in this book for reasons explained in Chapter 1. The nodes of this grid are then listed in a data frame, which serves as the sampling frame (Chapter 1). In the next code chunk a simple random sample without replacement of size 40 is selected from `Voorst`. The infinite population is represented by the nodes of a square grid with a spacing of 25 m. These nodes are listed in the `data.frame` `grdVoorst`.

```
load("data/Voorst.RData")
n <- 40
N <- nrow(grdVoorst)
set.seed(314)
units <- sample.int(N, size=n, replace=FALSE)
mysample <- grdVoorst[units,]
head(mysample)
```

	stratum	s1	s2	z
1659	EA	206992	464505.5	2.389717
2442	XF	202567	464605.5	8.657280
1823	XF	205092	464530.5	4.578232
1994	EA	203367	464555.5	7.434995
8083	PA	205592	465180.5	14.252331
5773	XF	201842	464955.5	19.845524

The result of the function `sample.int` is a vector with the selected nodes of the discretisation grid. The order of the elements of the vector is the order in which these are selected. Restricting the sampling points to the nodes of a discretisation grid can be avoided as follows. The columns `s1` and `s2` in the `data.frame` `grdVoorst` are the spatial coordinates of the centers of grid cells of 25 m by 25 m. Now a simple random sample is selected in two stages. First n times a grid cell is selected by simple random sampling *with replacement*. Second, every time a grid cell is selected, one point is selected fully randomly within this grid cell. This selection procedure accounts for the infinite number of points in the population. In the code chunk below the second step of this selection procedure is implemented with function `jitter`. It adds random noise to the spatial coordinates of the centers of the selected grid cells, by drawing from a continuous uniform distribution $unif(-c, c)$, with c half the side length of the square grid cells. With this selection procedure we respect that the population

actually is infinite.

```
set.seed(314)
units <- sample.int(N, size=n, replace=TRUE)
mysample <- grdVoorst[units,]
cellsize <- 25
mysample$s1 <- jitter(mysample$s1, amount=cellsize/2)
mysample$s2 <- jitter(mysample$s2, amount=cellsize/2)
head(mysample)
```

	stratum	s1	s2	z
1659	EA	206985.5	464493.0	2.389717
2442	XF	202573.6	464608.9	8.657280
1823	XF	205095.4	464527.1	4.578232
1994	EA	203368.8	464556.4	7.434995
8083	PA	205598.3	465181.2	14.252331
5773	XF	201836.1	464965.1	19.845524

The variable `stratum` is not used in this chapter. The result is shown in Figure 3.1. Note that before using the function `sample.int` I set a seed for the random number generator. This is to be able to reproduce the result: every time the same seed is used, the same sample of units is selected. Without setting a seed, we do not control the seed (R will choose one), and different samples will be selected.

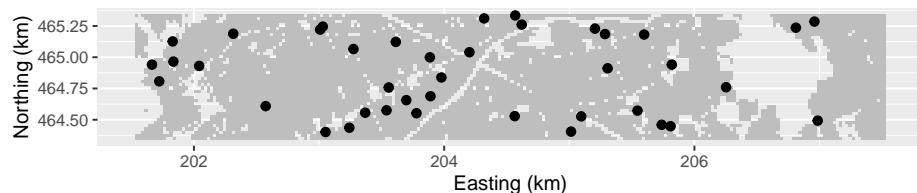


Figure 3.1: Simple random sample of size 40 from Voorst.

Drop outs

What to do with selected units that do not belong to the target population, or cannot be observed for whatever reason (e.g. no permission)? In practice it may happen that inspection in the field shows that a selected sampling unit does

not belong to the target population. For instance, in a soil survey the sampling unit may happen to fall on a road or in a built-up area. Shifting this unit to a nearby unit may lead to a biased estimator of the population mean, i.e. a systematic error in the estimated population mean. Besides, knowledge of the inclusion probabilities is lost. This can be avoided by discarding these units and to replace them by sampling units from a back-up list, selected in the same way, i.e. by the same type of sampling design. The order of sampling units in this list must be the order in which they are selected. In summary, do not replace a deleted sampling unit by the nearest sampling unit from the back-up list, but by the first unit, not yet selected, from the back-up list.

Arbitrary (haphazard) sampling versus probability sampling

In publications it is commonly stated that the sampling units were selected (more or less) at random (within strata), without further specification of how the sampling units were precisely selected. In statistical inference, the sampling units are subsequently treated as if they were selected by (stratified) simple random sampling. With probability sampling all units in the population have a positive probability of being selected, and the inclusion probabilities are known for all units. It is highly questionable whether this also holds for arbitrary and haphazard sampling. In arbitrary and haphazard sampling the sampling units are not selected by a probability mechanism. So the selection probabilities of the sampling units and of combinations of sampling units are unknown. This makes design-based estimation impossible, as this is based on the inclusion probabilities as determined by the sampling design. The only option for statistical analysis using arbitrarily or haphazardly selected samples is model-based inference, i.e. a model of the spatial variation must be assumed.

3.1 Estimation of population parameters

In simple random sampling without replacement of a finite population every possible sample of n units has an equal probability of being selected. There are $\binom{N}{n}$ samples of size n , and $\binom{N-1}{n-1}$ samples that contain unit k . From this it follows that the probability that unit k is included in the sample is $\binom{N-1}{n-1}/\binom{N}{n} = \frac{n}{N}$ (?). Substituting this in the general π -estimator for the total (Equation (2.2)) gives for simple random sampling without replacement (from finite populations)

$$\hat{t}(z) = \frac{N}{n} \sum_{k \in \mathcal{S}} z_k = N \bar{z}_{\mathcal{S}}, \quad (3.1)$$

with $\bar{z}_{\mathcal{S}}$ the (unweighted) *sample mean*. So for simple random sampling without replacement the π estimator of the population mean is the *unweighted* sample mean:

$$\hat{\bar{z}} = \bar{z}_{\mathcal{S}} = \frac{1}{n} \sum_{k \in \mathcal{S}} z_k. \quad (3.2)$$

In simple random sampling with replacement of finite populations a unit may occur multiple times in the sample \mathcal{S} . In this case the population total can be estimated by the pwr estimator (?)

$$\hat{t}(z) = \frac{1}{n} \sum_{k \in \mathcal{S}} \frac{z_k}{p_k}, \quad (3.3)$$

where n is the number of draws, and p_k is the draw-by-draw selection probability of unit k . With simple random sampling $p_k = 1/N, k = 1, \dots, N$. Inserting this in the pwr estimator yields

$$\hat{t}(z) = \frac{N}{d} \sum_{k \in \mathcal{S}} z_k. \quad (3.4)$$

Alternatively, the population total can be estimated by the π estimator. With simple random sampling with replacement the inclusion probability of each unit k equals $1 - (1 - \frac{1}{N})^n$, which is smaller than the inclusion probability with simple random sampling without replacement of size n (?). Inserting these inclusion probabilities in the general π estimator of the population total (Equation (2.2)) where the sample \mathcal{S} is reduced to the unique units in the sample, yields the π estimator of the total for simple random sampling with replacement.

With simple random sampling of *infinite* populations the π estimator of the population mean equals the sample mean. Multiplying this estimator with the area of the region of interest A yields the π estimator of the population total:

$$\hat{t}(z) = \frac{A}{n} \sum_{k \in \mathcal{S}} z_k . \quad (3.5)$$

The simple random sample of size 40 selected above is used to estimate the population total of SOM. First the population mean is estimated.

```
mz <- mean(mysample$z)
```

The estimated population mean is 8.07. Simply multiplying the estimated population mean by the area A to obtain an estimate of the population total is not very useful, as the dimension of the total then is g kg⁻¹ m². To estimate the total of SOM in the soil layer 0-30 cm, first the soil volume in m³ is computed by the total number of grid cells, N , multiplied by the size of the grid cells and by the thickness of the soil layer. The total is then estimated by the product of this volume, the bulk density of soil in g cm⁻³ and the estimated population mean. This is divided by 10⁶ to obtain the total SOM in megatons (10⁹ kg).

```
Vol <- N*25^2*0.3
bd <- 1.5
tz <- Vol*bd*mz*10^-6
```

The estimated total is 17.1 megaton. Note that that a constant bulk density is used. Ideally, this bulk density is also measured at the sampling points, by collecting soil aliquots of a constant volume. The measured SOM concentration and bulk density can then be used to compute the volumetric SOM content in kg m⁻³ at the sampling points. The estimated population mean of this volumetric SOM content can then be multiplied by the total volume of soil in the study area, to get an estimate of the total mass of SOM in the study area.

The simulated population is now sampled 10,000 times to see how sampling affects the estimates. For each sample the population mean is estimated by the sample mean. Figure 3.2 shows a histogram of the 10,000 estimated population means (sample means).

If we would repeat the sampling an infinite number of times and make the width of the bins in the histogram infinitely small, then we obtain, after scaling so that the sum of the area under the curve equals 1, the *sampling distribution*

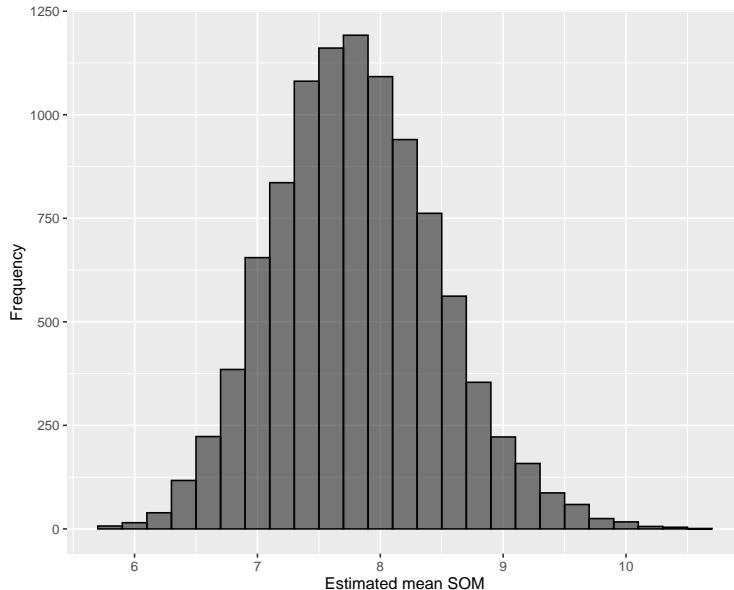


Figure 3.2: Sampling distribution of the estimator of the population mean of SOM (g/kg) in Voorst with simple random sampling of size 40.

of the estimator of the population mean. Important summary statistics of this sampling distribution are:

1. Expectation (mean).
2. Variance, referred to as the *sampling variance*.

When the expectation equals the population mean, there is no systematic error. The estimator is then said to be *design-unbiased* or *p-unbiased*. In Chapter 22 another type of unbiasedness is introduced, model-unbiasedness. The difference between design-unbiasedness and model-unbiasedness is explained in Chapter 27. In following chapters of Part I unbiased means design-unbiased. Actually, it is not the estimator which is unbiased, but the combination of a sampling design and an estimator. For instance, with unequal probability sampling designs, the sample mean is not an unbiased estimator of the population mean, whereas it is in combination with an equal probability sampling design.

The sampling variance is a measure of the random error. Ideally this variance is as small as possible, so that there is a large probability that for an individual estimate the estimation error is small. The variance is a measure of the *precision* of an estimator. An estimator with a small variance but a strong bias is not a good estimator. To assess the quality of estimator we should look at both. The variance and the bias are often combined in the *mean squared error* (MSE), which is the sum of the variance and the *squared* bias. An estimator with a small MSE is an *accurate* estimator. So contrary to precision, accuracy also accounts for the bias.

Do not confuse the *population* variance and the *sampling* variance. The population variance (spatial variance) is a *population characteristic*, whereas the sampling variance is a *characteristic of a sampling strategy*, index{Sampling strategy} i.e. a combination of a sampling design and an estimator. The sampling variance quantifies our *uncertainty* about the population mean. The sampling variance can be manipulated by changing the sample size n , the type of sampling design, and the estimator. This has no effect on the population variance. The average of the 10,000 estimated population means equals 7.81, so the difference with the true population means equals -0.004. The variance of the 10,000 estimated population means equals 0.45. The square root of this variance, referred to as the *standard error*, equals 0.67. Note that the standard error has the same dimension as the study variable, g/kg, whereas the dimension of the variance is the squared dimension of the study variable.

3.1.1 Population proportion

In some cases one is interested in the proportion of the population (study area) satisfying a given condition. Think for instance of the proportion of trees in a forest infected by some disease, the proportion of an area (areal fraction) in which a soil pollutant exceeds some critical threshold, or the proportion of an area where habitat conditions are suitable for some endangered species. Recall that a population proportion is defined as the population mean of an 0/1 indicator y with value 1 if the condition is satisfied, and 0 otherwise (Section ??PopulationParameters)). For simple random sampling this population proportion can be estimated by the same formula as for the mean (Equation (3.2)):

$$\hat{p} = \frac{1}{n} \sum_{k \in S} y_k . \quad (3.6)$$

3.1.2 Cumulative distribution function and quantiles

The population CDF is defined in Equation (1.4). A population CDF can be estimated by repeated application of the indicator technique described in the previous section on estimating a population proportion. A series of threshold values is chosen. Each threshold results in n indicator values having value 1 if the observed study variable z of unit k is smaller than or equal to the threshold, and 0 otherwise. These indicator values are then used to estimate the proportion of the population with a z value smaller than or equal to that threshold. For simple random sampling these proportions can be estimated with Equation (3.6). Commonly the unique z values in the sample are used as threshold values, leading to as many estimated population proportions as there are unique values in the sample.

Figure 3.3 shows the estimated CDF, estimated from the simple random sample of 40 units from Voorst. The steps are at the unique values of SOM in the sample.

```
ggplot(mysample, mapping=aes(z)) +
  stat_ecdf(geom="step")
```

The estimated population proportions can be used to estimate a population quantile for any population proportion (cumulative frequency, probability), for instance the median, first quartile and third quartile, corresponding with a population proportion of 0.5, 0.25 and 0.75, respectively. A simple estimator is the smallest k th order statistic with an estimated proportion larger than or equal to the desired cumulative frequency (?).

The estimated CDF shows jumps of size $1/n$, so that the estimated population proportion can be larger than the desired proportion. The estimated population proportions therefore are often interpolated, for instance by linear interpolation. Function `quantile` of the `stats` package can be used to estimate a quantile. With argument `type=4` linear interpolation is used to estimate the quantiles. Note that this function `quantile` actually computed *sample quantiles*, i.e. it assumes that the population units are selected with equal inclusion probabilities (as in simple random sampling), so that the estimators of the population proportions obtained with Equation (3.6) are unbiased. With unequal inclusion probabilities these probabilities must be accounted for in estimating the population proportions, see following chapters.

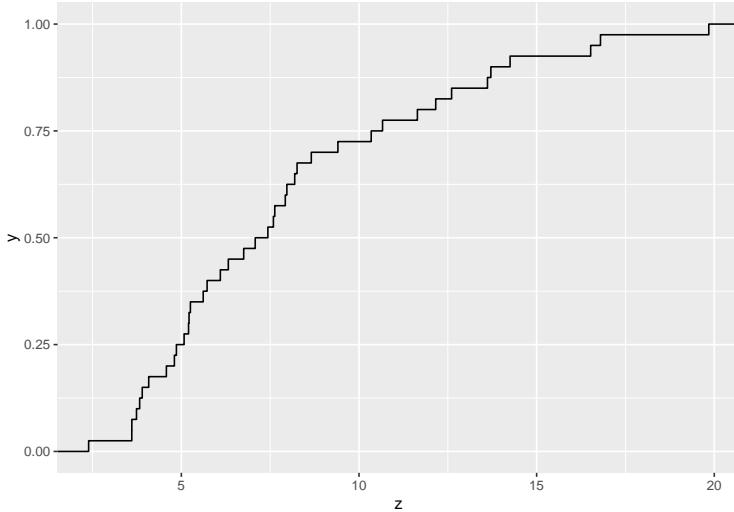


Figure 3.3: Cumulative distribution function, estimated from the simple random sample of 40 units from Voorst.

```
quantile(mysample$z, probs=c(0.25,0.5,0.75), type=4) %>%
  round(.,3)
```

```
25%      50%      75%
4.860  7.079 10.344
```

Package **QuantileNPCI** (?) can be used to compute a non-parametric confidence interval estimate of a quantile, using fractional order statistics (?). Parameter **q** specifies the proportion.

```
library(QuantileNPCI)
res <- quantCI(mysample$z, q=0.5, alpha=0.05, method="exact")
```

The estimated median equals 7.257, the lower bound of the 95% confidence interval equals 5.376, and the upper bound 8.234.

372. Sampling variance of estimator of population mean, total and proportion

Exercises

1. Compare the histogram of the estimated population means with the histogram of the 7528 simulated values in the population (Figure 1.4). Explain the differences.
2. What happens with the spread in the histogram (variance of estimated population means) when the sample size n is increased?
3. Suppose we would repeat the sampling 10^{12} number of times, what would happen with the difference between the average of the estimated population means and the population mean?

3.2 Sampling variance of estimator of population mean, total and proportion

For simple random sampling of an infinite population and simple random sampling with replacement of a finite population the sampling variance of the estimator of the population mean equals

$$V(\hat{z}) = \frac{S^2(z)}{n} , \quad (3.7)$$

with $S^2(z)$ the *population variance*, also referred to as the spatial variance. For finite populations this population variance is defined as (?)

$$S^2(z) = \frac{1}{N-1} \sum_{k=1}^N (z_k - \bar{z})^2 , \quad (3.8)$$

and for infinite populations as

$$S^2(z) = \frac{1}{A} \int_{\mathbf{s} \in \mathcal{U}} (z(\mathbf{s}) - \bar{z})^2 d\mathbf{s} , \quad (3.9)$$

with $z(\mathbf{s})$ the value of the study variable z at a point with two-dimensional coordinates $\mathbf{s} = (s_1, s_2)$, A the area of the study area, and \mathcal{U} the universe of

interest (study area). In practice we select only one sample, i.e. we do not repeat the sampling many times. Still it is possible to *estimate* the variance of the estimator of the population means if we would repeat the sampling. In other words, we can estimate the sampling variance of the estimator of the population mean from a single sample. We do so by estimating the population variance from the sample, and this estimate can then be used to estimate the *sampling* variance of the estimator of the population mean. For simple random sampling *with replacement* from finite populations the sampling variance of the estimator of the population mean can be estimated by

$$\widehat{V}(\widehat{\bar{z}}) = \frac{\widehat{S^2}(z)}{n} = \frac{1}{n(n-1)} \sum_{k \in \mathcal{S}} (z_k - \bar{z}_{\mathcal{S}})^2 , \quad (3.10)$$

with $\widehat{S^2}(z)$ the *estimated* population variance. With simple random sampling the *sample* variance, i.e. the variance of the sample data, is an unbiased estimator of the population variance. This estimator can also be used for *infinite* populations. For simple random sampling *without replacement* from finite populations the sampling variance of the estimator of the population mean can be estimated by

$$\widehat{V}(\widehat{\bar{z}}) = \left(1 - \frac{n}{N}\right) \frac{\widehat{S^2}(z)}{n} . \quad (3.11)$$

The term $1 - \frac{n}{N}$ is referred to as the finite population correction (fpc).

In the sampling experiment described above, the average of the 10,000 *estimated* sampling variances equals 0.442. The true sampling variance equals 0.438. So the difference is very small, indicating that the estimator of the sampling variance, Equation (3.11), is design-unbiased.

The sampling variance of an estimated total of a finite population can be estimated by multiplying the estimated variance of the estimator of the population mean by N^2 . For simple random sampling without replacement this estimator thus equals

$$\widehat{V}(\widehat{t}(z)) = N^2 \left(1 - \frac{n}{N}\right) \frac{\widehat{S^2}(z)}{n} . \quad (3.12)$$

For simple random sampling of infinite populations the sampling variance of the estimator of the total can be estimated by

392. Sampling variance of estimator of population mean, total and proportion

$$\widehat{V}(\hat{t}(z)) = A^2 \frac{\widehat{S^2}(z)}{n} . \quad (3.13)$$

The sampling variance of an estimated proportion \hat{p} for simple random sampling without replacement of a finite population can be estimated by

$$\widehat{V}(\hat{p}) = \left(1 - \frac{n}{N}\right) \frac{\hat{p}(1 - \hat{p})}{n - 1} . \quad (3.14)$$

The numerator in this estimator is an estimate of the population variance of the indicator. Note that this estimated population variance is divided by $n - 1$, and not by n as in the estimator of the mean (?).

Estimation of the standard error of the estimated population mean in **R** is very straightforward. To estimate the standard error of the estimated total in megatons the standard error of the estimated population mean must be multiplied by a constant.

```
se_mz <- sqrt(var(mysample$z)/n)
se_tz <- se_mz*Vol*bd*10^-6
```

The estimated standard error of the estimated total equals 1.4. This standard error does not account for spatial variation of bulk density.

Although there is no advantage in using package **survey** (?) to compute the π estimator and its standard error for this simple sampling design, I illustrate how this works. For more complex designs and alternative estimators, estimation of the population mean and its standard error with functions defined in this package is very convenient, as will be shown in the following chapters.

First the sampling design that is used to select the sampling units is specified with function **svydesign**. The first argument specifies the sampling units. In this case the nodes of discretisation grid are used as sampling units, which is indicated by the formula **id=~1**. In Chapter 6 clusters of population units are used as sampling units, and in Chapter 7 both clusters and individual units are used as sampling units. The argument **probs** specifies the inclusion probabilities of the sampling units. Alternatively, we may specify the weights with argument **weights**, which are in this case equal to the inverse of the inclusion probabilities.

The variable `pi` is a column in the data frame `mysample`, which is indicated with the tilde in `probs=~pi`.

The population mean is then estimated with function `svymean`. The first argument is a formula specifying the study variable. The argument `design` specifies the sampling design.

```
library(survey)
mysample$pi <- n/N
design_si <- svydesign(id=~1, probs=~pi, data=mysample)
svymean(~z, design=design_si)

mean      SE
z 8.0736 0.6611
```

For simple random sampling of finite populations without replacement, argument `fpc` is used to correct the standard error.

```
mysample$N <- N
design_si <- svydesign(id=~1, probs=~pi, fpc=~N, data=mysample)
svymean(~z, design_si)

mean      SE
z 8.0736 0.6594
```

The estimated standard error is smaller now due to the finite population correction, see Equation (3.11).

Population totals can be estimated with function `svytotals`, quantiles with function `svyquantile`, and ratios of population totals with `svyratio`, to mention a few functions that will be used in following chapters.

```
svyquantile(~z, design_si, quantile=c(0.5,0.9))

$z
  quantile   ci.2.5   ci.97.5       se
0.5 7.07889  5.254256  8.256144 0.7420532
0.9 13.71163 12.159431 19.845524 1.8999673
```

```
attr(,"hasci")
[1] TRUE
attr(,"class")
[1] "newsvyquantile"
```

Exercises

4. Is the sampling variance for simple random sampling without replacement larger or smaller than for simple random sampling with replacement, given the sample size n ? Explain your answer.
5. What is the effect of the population size N on this difference?
6. In Section 3.2 I computed the true sampling variance, i.e. the variance of the estimator of the population means if we would repeat the sampling an infinite number of times. How can this true sampling variance be computed?
7. In reality we cannot compute the true sampling variance. Why not?

3.3 Simple random sampling of circular plots

In forest inventory, vegetation surveys and agricultural surveys circular sampling plots are quite common. Using circular plots as sampling units is not entirely straightforward because the study area cannot be partitioned into a finite number of circles that fully cover the study area. The use of circular plots as sampling units can be implemented in two ways (?):

1. Sampling from a finite set of fixed circles.
2. Sampling from an infinite set of floating circles.

3.3.1 Sampling from a finite set of fixed circles

Sampling from a finite set of fixed circles is most simple, but as we will see requires an assumption about the distribution of the study variable in the population. In this implementation the sampling units consist of a finite set of slightly overlapping or non-overlapping fixed circular plots (Figure 3.4). The circles can be constructed as follows. A grid with squares is superimposed on

the study area, so that it fully covers the study area. These squares are then substituted by circles with an area equal to the area of the squares, or by non-overlapping tangent circles inscribed in the squares. The radius of the partly overlapping circles equals $\sqrt{a/\pi}$, with a the area of the squares, the radius of the non-overlapping circles equals $\sqrt{a}/2$. In both implementations the infinite population is replaced by a finite population of circles that does not fully tessellate the study area. When using the partly overlapping circles as sampling units we may avoid overlap by selecting a systematic sample (Chapter 5) of circular plots. The population total can then be estimated by Equation (3.12), substituting A/a for N , and where z_k is the total of the k th circle (sum of observations of all population units in k th circle). However, no unbiased estimator of the sampling variance of the estimator of the population total or mean is available for this sampling design, see Chapter 5.

With simple random sampling of non-overlapping circular plots the finite population total can be estimated by Equation (3.1), and its sampling variance by Equation (3.12). However, the circular plots do not cover the full study area, and as a consequence the total of the infinite population is underestimated. A corrected estimate can be obtained by estimating the mean of the finite population and multiplying this estimated population mean by A/a (?):

$$\hat{t}(z) = \frac{A}{a} \hat{\bar{z}}, \quad (3.15)$$

with $\hat{\bar{z}}$ the estimated mean of the finite population. The variance can be estimated by the variance of the mean of the finite population multiplied by the square of A/a . However, we still need to assume that the mean of the finite population is equal to the mean of the infinite population. This assumption can be avoided by sampling from an infinite set of floating circles.

3.3.2 Sampling from an infinite set of floating circles

Simple random sampling of floating circular plots can be done by selecting the centers of the plots by simple random sampling, and then determining the border of the circular plots. The circular plots overlap if two selected points are separated by a distance smaller than the diameter of the circular plots. Besides, when a plot is selected near the border of the study area, a part of the plot is outside the study area. This part is ignored in estimating the population mean or total. To select the centers the study area must be extended by a zone with

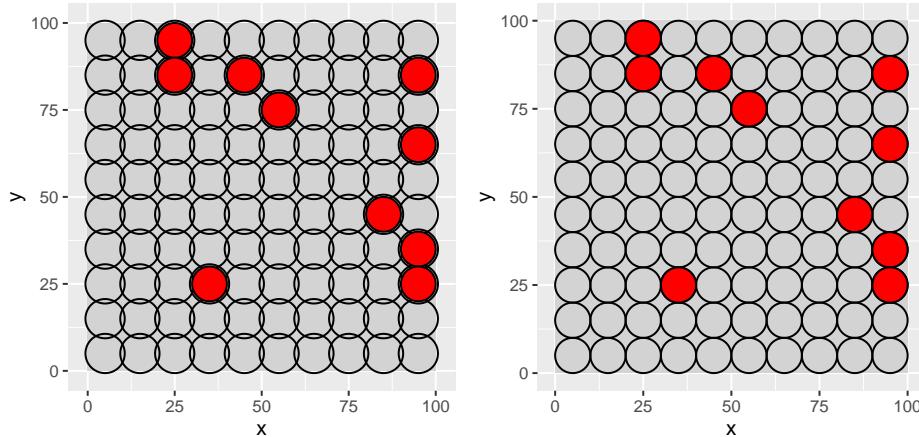


Figure 3.4: A simple random sample of ten circular plots from a square area discretised by a finite set of partly overlapping or non-overlapping circular plots.

a width equal to the radius of the circular plots. This is illustrated in Figure 3.5, showing a square study area of 100 m x 100 m. To select ten circular plots with a radius of 5 m from this square, ten points are selected by simple random sampling, using function `runif`, with -5 as lower limit and 105 as upper limit of the uniform distribution.

```
set.seed(129)
s1 <- runif(10, min=-5, max=105)
s2 <- runif(10, min=-5, max=105)
```

Two points are selected outside the study area, in the extended zone. For both points a small part of the circular plot is inside the square. To determine the study variable for these two sampling units, only the part of the plot inside the square is observed. In other words, these two observations have a smaller support than the observations on the other eight plots, see Chapter 1.

In the upper left corner two sampling units are selected that largely overlap. The intersection of the two circular plots is used twice, to determine the study variable of both sampling units.

Given the observations for the selected circular plots, the population total can

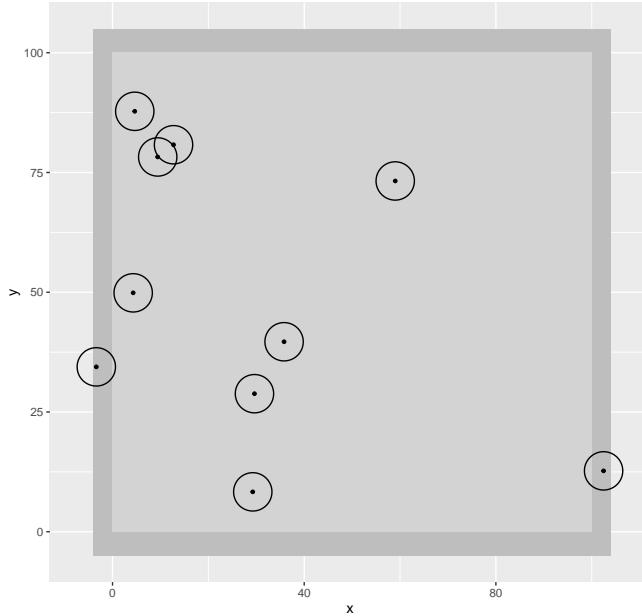


Figure 3.5: A simple random sample of ten floating circular plots from a square area.

be estimated by (?)

$$\hat{t}(z) = \frac{A}{a} \frac{1}{n} \sum_{k \in S} z_k , \quad (3.16)$$

with a the area of the circle and z_k the observed total of sampling unit k (circle). The same estimate of the total is obtained if we divide the observations by a to obtain a mean per sampling unit:

$$\hat{t}(z) = A \frac{1}{n} \sum_{k \in S} \frac{z_k}{a} . \quad (3.17)$$

The sampling variance of the estimator of the total can be estimated by

$$\widehat{V}(\widehat{t}(z)) = \left(\frac{A}{a}\right)^2 \frac{\widehat{S^2}(z)}{n}, \quad (3.18)$$

with $\widehat{S^2}(z)$ the estimated population variance of the totals per population unit (circle).

3.4 Confidence interval estimates

A second way of expressing our uncertainty about the estimated total, mean or proportion is to present not merely a single number, but an interval. The wider the interval, the more uncertain we are, and vice versa, the narrower the interval, the more confident we are about the estimate. To learn how to compute a confidence interval, I return to the sampling distribution of the estimator of the mean soil organic matter concentration. Suppose we would like to compute the bounds of an interval $[a, b]$ such that 5% of the estimated population means is smaller than a , and 5% is larger than b . To compute the lower bound a and upper bound b of this 90%-interval, we must specify the distribution function. When the distribution of the study variable z is normal and we know the variance of z in the population, then the sampling distribution of the estimator of the population mean is also normal, regardless of the sample size. The larger the sample size, the smaller the effect of the distribution of z on the sampling distribution of the estimator of the population mean. For instance, even when the distribution of z is far from symmetric, then still the sampling distribution of the estimator of the population mean is approximately normal if the sample size is large, say $n > 100$. This is the essence of the central limit theorem. Above we already noticed that the sampling distribution is much less asymmetric than the histogram of the simulated values, and looks much more like a normal distribution. Assuming a normal distribution, the bounds of the 90%-interval are given by

$$\widehat{\bar{z}} \pm u_{(0.10/2)} \cdot \sqrt{V(\widehat{\bar{z}})}, \quad (3.19)$$

where $u_{(0.10/2)}$ is the 0.95 quantile of the standard normal distribution, i.e. the value of u having a tail area of 0.05 to its right. Note that in this equation the sampling variance of the estimator of the population mean $V(\widehat{\bar{z}})$ is used. In practice this variance is unknown, because the population variance is unknown, and must be estimated from the sample (Equations (3.10) and (3.11)). To

account for the unknown sampling variance, the standard normal distribution is replaced by the Student's t distribution, which has thicker tails than the standard normal distribution. This leads to the following bounds of the $100(1 - \alpha)\%$ confidence interval estimate of the mean:

$$\hat{z} \pm t_{1-\alpha/2}^{(n-1)} \cdot \sqrt{\hat{V}(\hat{z})}, \quad (3.20)$$

where $t_{1-\alpha/2}^{(n-1)}$ is the $(1 - \alpha/2)$ quantile of the Student's t distribution with $(n - 1)$ degrees of freedom. The quantity $(1 - \alpha)$ is referred to as the confidence level. The larger the number of degrees of freedom $(n - 1)$, the closer the Student's t distribution is to the standard normal distribution. The quantity $t_{1-\alpha/2}^{(n-1)} \cdot \sqrt{\hat{V}(\hat{z})}$ is referred to as the margin of error.

The function `qt` computes a quantile of a Student's t distribution, given the degrees of freedom and the cumulative probability. The bounds of the confidence interval can then be computed as follows.

```
alpha <- 0.05
margin <- qt(1-alpha/2, n-1, lower.tail=TRUE)*se_mz
lower <- mz - margin
upper <- mz + margin
```

More easily we can use method `confint` of package `survey` to compute the confidence interval.

```
confint(svymean(~z, design_si), df=degf(design_si), level=0.95)

2.5 % 97.5 %
z 6.739875 9.407342
```

The interpretation of a confidence interval is not straightforward. A common misinterpretation is that if the 90% confidence interval estimate of the mean equals $[a, b]$, then the probability that the population mean is in this interval equals 90%. In classical sampling theory this cannot be a correct interpretation, because the population mean is not a random variable, and consequently the probability that the population mean is in an interval does not exist. However, the estimated bounds of the confidence interval are random variables, because

the estimated population mean and also the estimated sampling variance varies between samples drawn with the sampling design, so it does make sense to attach a probability to this interval. Figure 3.6 shows the 90% confidence interval estimates of the mean for the first 100 simple random samples drawn above. Note that both the location and the length of the intervals differ between samples. For each sample I determined whether this interval covers the population mean.

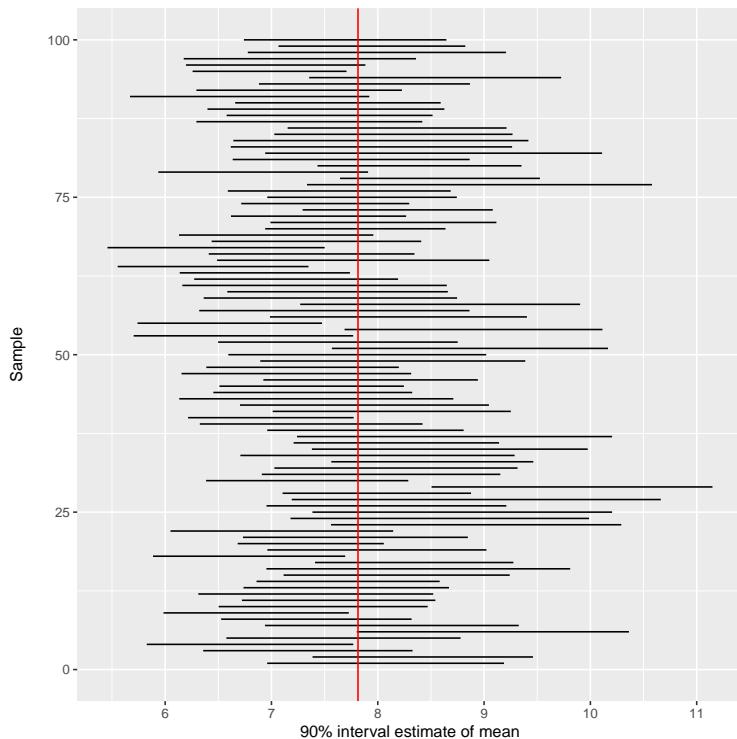


Figure 3.6: Estimated confidence intervals of the population mean of SOM (g per kg) in Voorst, estimated from 100 simple random samples of size 40. The vertical red line is at the true population mean.

Out of the 10,000 samples, 1153 samples do not cover the population mean, i.e. close to the specified 10%. So, a 90% confidence interval is a random interval that contains in the long run the population mean 90% of the time.

3.4.1 Confidence interval for proportion

Ideally a confidence interval for a population proportion is based on the binomial distribution of the number of sampling units satisfying a condition (the number of successes). The binomial distribution is a discrete distribution. There are various methods for computing coverage probabilities of confidence intervals for a binomial proportion, see `?BinomCI` for a discussion. A common method for computing the confidence interval of a proportion is the Clopper-Pearson method. Function `BinomCI` of package **DescTools** can be used to compute confidence intervals for proportions (`?`).

```
library(DescTools)
n <- 50
k <- 5
print(p.est <- BinomCI(k, n, conf.level=0.95,
                        method="clopper-peerson"))

      est      lwr.ci      upr.ci
[1,] 0.1 0.03327509 0.2181354
```

The confidence interval is not symmetric around the estimated proportion of 0.1. As can be seen below the upper bound is the proportion at which the probability of 5 or less successes is 0.025,

```
pbinom(q=k, size=n, prob=p.est[3])

[1] 0.025
```

and the lower bound of the confidence interval is the proportion at which the probability of 5 or more successes is also equal to 0.025. Note that to compute the upper tail probability we must assign $k - 1 = 4$ to argument `q`, because with argument `lower.tail=FALSE` function `pbinom` computes the probability of $X > x$, not of $X \geq x$.

```
pbinom(q=k-1, size=n, prob=p.est[2], lower.tail=FALSE)

[1] 0.025
```

For large sample sizes and for proportions close to 0.5 the confidence interval can be computed with a normal distribution as an approximation to the binomial distribution, using Equation (3.14) for the variance estimator of a proportion:

$$\hat{p} \pm u_{\alpha/2} \sqrt{\frac{\hat{p}(1 - \hat{p})}{n - 1}}. \quad (3.21)$$

This interval is referred to as the Wald interval. It is a fact that unless n is very large, the actual coverage probability of the Wald interval is poor for p near 0 or 1. A rule of thumb is that the Wald interval should be used only when $n \cdot \min\{p, (1 - p)\}$ is at least 5 or 10. For small n ? recommend the Wilson interval, and the Agresti-Coull interval for larger n . These intervals can be computed with function `BinomCI` of package **DescTools**.

Exercises

8. Write an **R** script to select a simple random sample of size 100 from Voorst (data are in `data/Voorst.RData`).
- Use the selected sample to estimate the population mean of SOM and its standard error (SOM is in the column z of the data frame).
 - Compute the lower- and upper bound of the 90% confidence interval using the Student's t distribution, and check whether the population mean SOM is covered by the interval.
 - Compare the length of the 90% confidence interval with the length of the 95% interval. Explain the difference in width.
 - Use the selected sample to estimate the total mass of soil organic matter in the topsoil (0 - 20 cm) of Voorst. Use as a bulk density 1.4 g/kg. The size of the pixels is 25 m by 25 m.
 - Estimate the standard error of the estimated total.
 - Do you think this standard error is a realistic estimate of the uncertainty about the estimated total?

Chapter 4

Stratified simple random sampling

In stratified random sampling the population is divided into subpopulations, for instance soil mapping units, areas with the same land use or land cover, administrative units, etc. The subareas are mutually exclusive, i.e. they do not overlap, and are jointly exhaustive, i.e. their union equals the entire population (study area). Within each subpopulation, referred to as a stratum, a probability sample is selected by some sampling design. If these probability samples are selected by simple random sampling, as described in the previous chapter, the design is stratified *simple* random sampling (STS). If sampling units were selected by cluster random sampling, then the design is stratified *cluster* random sampling. This chapter is about stratified simple random sampling.

Stratified simple random sampling is illustrated with Voorst (Figure 4.1). In the data frame with simulated data there is a column `stratum`. These are combinations of soil classes and land use, obtained by overlaying a soil map and a land use map. To select a stratified simple random sample, we set the total sample size n and the sampling units must be apportioned to the strata. I chose to apportion the units proportionally to the size (area, number of pixels) of the strata (see for details Section 4.3 hereafter). The larger a stratum, the more units are selected from this stratum. The stratum sizes (total number of pixels) are computed with function `tapply`.

```

library(sampling)
load("data/Voorst.RData")
N_h <- tapply(grdVoorst$stratum, INDEX=grdVoorst$stratum,
               FUN=length)
w_h <- N_h/sum(N_h)
n <- 40
print(n_h <- round(n * w_h))

```

BA EA PA RA XF
13 8 9 4 7

The sum of the stratum sample sizes is 41, we want 40, so we reduce the largest stratum sample size by 1.

```
n_h[1] <- n_h[1] - 1
```

The stratified simple random sample is selected with the function `strata` of package `sampling` (?). The name of the package is added to the function (`sampling::strata`), as `strata` is also a function in another package. Not adding the name of the package may result in an error message. The argument `size` specifies the stratum sample sizes. These stratum sample sizes must be in the order in which the strata are encountered in the data frame `grdVoorst`, which is determined first with function `unique`. Within the strata the grid cells are selected by simple random sampling *with replacement* (`method="srswr"`), so that in principle more than one point can be selected within a grid cell, see Chapter 3 for a motivation of this. The function `getdata` extracts the observations for the selected units from the sampling frame, as well as the spatial coordinates and the stratum of these units. The coordinates of the centers of the selected grid cells are jittered by an amount equal to half the side of the grid cells.

```

ord <- unique(grdVoorst$stratum)
set.seed(314)
units <- sampling::strata(
  grdVoorst, stratanames="stratum", size=n_h[ord], method="srswr")
mysample <- getdata(grdVoorst, units)
cellsize <- 25

```

```
mysample$s1 <- jitter(mysample$s1, amount=cellsize/2)
mysample$s2 <- jitter(mysample$s2, amount=cellsize/2)
```

Figure 4.1 shows the selected sample.



Figure 4.1: Stratified simple random sample of size 40 from Voorst. Strata are combinations of soil class and land use.

4.1 Estimation of population parameters

With simple random sampling within strata, the estimator of the mean for simple random sampling (Equation (3.2)) is applied at the level of the strata. The estimated stratum means are then averaged, using the relative sizes (relative areas) of the strata as weights:

$$\hat{\bar{z}} = \sum_{h=1}^H w_h \hat{\bar{z}}_h , \quad (4.1)$$

where H is the number of strata, w_h are the relative sizes (areas) of the strata (stratum weights): $w_h = N_h/N$, and $\hat{\bar{z}}_h$ is the estimated mean of stratum h estimated by the sample mean for stratum h :

$$\hat{\bar{z}}_h = \frac{1}{n_h} \sum_{k \in \mathcal{S}_h} z_k , \quad (4.2)$$

with \mathcal{S}_h the sample selected from stratum h .

The same estimator is found when the π estimator is worked out for stratified simple random sampling. With stratified simple random sampling without

replacement and different sampling fractions for the strata¹ the inclusion probabilities differ among the strata and equal $\pi_{hk} = n_h/N_h$ for all k in stratum h , with n_h the sample size of stratum h and N_h the size of stratum h . Inserting this in the π estimator of the population mean (Equation (2.4)) gives

$$\hat{\bar{z}} = \frac{1}{N} \sum_{h=1}^H \sum_{k \in \mathcal{S}_h} \frac{z_{hk}}{\pi_{hk}} = \frac{1}{N} \sum_{h=1}^H \frac{N_h}{n_h} \sum_{k \in \mathcal{S}_h} z_{hk} = \sum_{h=1}^H w_h \hat{\bar{z}}_h . \quad (4.3)$$

The sampling variance of the estimator of the population mean is estimated by first estimating the sampling variances of the estimated stratum means, and then pooling. Note that for the sampling variance we must square the stratum weights:

$$\widehat{V}(\hat{\bar{z}}) = \sum_{h=1}^H w_h^2 \widehat{V}(\hat{\bar{z}}_h) , \quad (4.4)$$

where $\widehat{V}(\hat{\bar{z}}_h)$ is the estimated sampling variance of $\hat{\bar{z}}_h$:

$$\widehat{V}(\hat{\bar{z}}_h) = (1 - \frac{n_h}{N_h}) \frac{\widehat{S^2}_h(z)}{n_h} , \quad (4.5)$$

with $\widehat{S^2}_h(z)$ the estimated variance of z within stratum h :

$$\widehat{S^2}_h(z) = \frac{1}{n_h - 1} \sum_{k \in \mathcal{S}_h} (z_{hk} - \hat{\bar{z}}_h)^2 . \quad (4.6)$$

For stratified simple random sampling with replacement of finite populations and stratified simple random sampling of infinite populations the $1 - (n_h/N_h)$ can be dropped.

¹The sampling fractions are usually slightly different, even with proportional allocation (Section 4.3) because n_h/N_h cannot be made exactly equal for all strata.

Table 4.1: Stratum size (N_h), stratum sample size (n_h), estimated stratum mean (Mean), estimated variance (Variance) and estimated variance of estimated stratum mean (Variance of mean).

	Nh	nh	Mean	Variance	Variance of mean
BA	2371	12	8.22	10.20	0.850
EA	1442	8	5.55	4.35	0.544
PA	1710	9	5.97	14.42	1.603
RA	659	4	7.64	9.24	2.310
XF	1346	7	8.73	11.48	1.640

```
mz_h <- tapply(mysample$z, INDEX=mysample$stratum, FUN=mean)
mz <- sum(w_h*mz_h)
S2z_h <- tapply(mysample$z, INDEX=mysample$stratum, FUN=var)
v_mz_h <- S2z_h/n_h
se_mz <- sqrt(sum(w_h^2*v_mz_h))
```

Table 4.1 shows per stratum the estimated means, estimated variances and estimated sampling variances of the estimated means. We can see large differences in the within-stratum variances. For the stratified sample of Figure 4.1 the estimated population mean equals 7.238 and the estimated standard error of this estimator equals 0.507.

The population mean can also be estimated directly using the basic π estimator (Equation (3.2)). The inclusion probabilities are included in the data frame `mysample`, in the column `Prob`.

```
head(mysample)
```

s1	s2	z	stratum	ID_unit	Prob	Stratum
1962	202554.8	464556.7	6.460569	XF	1135	0.005189017
3712	204305.5	464738.9	6.217245	XF	2159	0.005189017
6781	203038.3	465057.0	15.140350	XF	4205	0.005189017
7235	202381.1	465096.7	5.766409	XF	4503	0.005189017
8484	203610.4	465237.8	9.363492	XF	5336	0.005189017
9265	205147.1	465315.5	10.928986	XF	5853	0.005189017

The population total is estimated first, and by dividing this estimated total by the total number of population units N an estimate of the population mean is obtained.

```
tz <- sum(mysample$z/mysample$Prob)
print(mz <- tz/sum(N_h))

[1] 7.254982
```

The two estimates of the population mean are not exactly equal. This is due to rounding errors in the inclusion probabilities. This can be shown by computing the sum of the inclusion probabilities over all population units. This sum should be equal to the sample size $n = 40$, but as we can see below, this sum is slightly smaller.

```
pi_h <- tapply(mysample$Prob, INDEX=mysample$stratum, FUN=unique)
print(sum(pi_h*N_h))

[1] 39.90711
```

Now suppose that we ignore that the sample data come from a stratified sampling design, and we use the (unweighted) sample mean as an estimate of the population mean.

```
print(mean(mysample$z))

[1] 7.211431
```

The sample mean slightly differs from the proper estimate of the population mean. The sample mean is a *biased* estimator, but the bias is only small. The reason for the small bias is that the stratum sample sizes are about proportional to the sizes of the strata, so that the inclusion probabilities (sampling intensities) are about equal for all strata: 0.0050494, 0.0055344, 0.0052509, 0.006056, 0.005189. The probabilities are not exactly equal because the stratum sample sizes are necessarily rounded to integers and because we reduced the largest sample size by one unit. The bias would have been substantially larger if an equal number of units would have been selected from each stratum, leading to much larger differences in the inclusion probabilities among the strata. Sampling

intensity in stratum BA, for instance, then would be much smaller compared to the other strata, and so are the inclusion probabilities of the units in this stratum as compared to the other strata. Stratum is underrepresented in the sample. This is not a problem as long as we account for the difference in inclusion probabilities of the units in estimation of the population mean. If we do not account for these differences in inclusion probabilities, the estimator of the mean will be seriously biased.

The next code chunk shows how the population mean and its standard error can be estimated with package **survey** (?). Note that the stratum weights N_h/n_h must be given to function **svydesign** in argument **weight**. These are first attached to the data frame **mysample** by creating a look-up table **lut**, which is then merged with function **merge** to the data frame **mysample**.

```
library(survey)
labels <- sort(unique(mysample$stratum))
lut <- data.frame(stratum=labels, weight=N_h/n_h)
mysample <- merge(x=mysample, y=lut)
design_stsi <- svydesign(
  id=~1, strata=~stratum, weight=~weight, data=mysample)
svymean(~z, design_stsi)

      mean      SE
z 7.2382 0.5071
```

4.1.1 Estimation of population proportion, cumulative distribution function and quantiles

The proportion of a population satisfying some condition can be estimated by Equations (4.1) and (4.2), substituting for the study variable z_k an indicator y_k with value 1 if for unit k the condition is satisfied, and 0 otherwise (Section 3.1.1). In general with stratified simple random sampling the inclusion probabilities are not exactly equal, so that the estimated population proportion is not equal to the sample proportion.

These unequal inclusion probabilities must also be accounted for when estimating the cumulative distribution function (CDF) and quantiles (Section 3.1.2), as shown in the next code chunk for the CDF.

```
thresholds <- sort(unique(mysample$z))
cumfreq <- numeric(length=length(thresholds))
for (i in 1:length(thresholds)) {
  ind <- mysample$z <= thresholds[i]
  mh_ind <- tapply(ind, INDEX=mysample$stratum, FUN=mean)
  cumfreq[i] <- sum(w_h*mh_ind)
}
df <- data.frame(x=thresholds, y=cumfreq)
```

Figure 4.2 shows the estimated CDF, estimated from the stratified simple random sample of 40 units from Voorst (Figure 4.1).

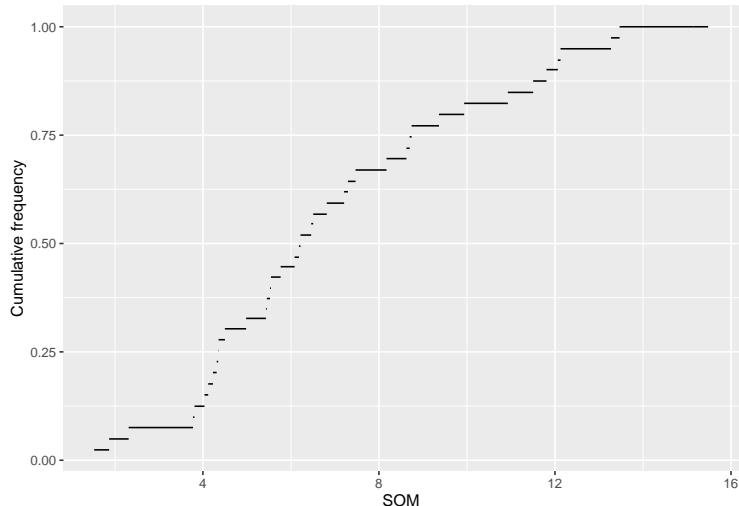


Figure 4.2: Cumulative distribution function estimated from the stratified simple random sample of 40 units from Voorst.

The estimated proportions (cumulative frequencies) are used to estimate a quantile. These estimates are easily obtained with function `svyquantile` of package `survey`.

```

svyquantile(~z, design_stsi, quantile=c(0.5,0.9))

$z
  quantile   ci.2.5   ci.97.5      se
0.5  6.460569  5.524261 8.625079 0.7637077
0.9 12.064708 10.928986       NaN       NaN

attr("hasci")
[1] TRUE
attr("class")
[1] "newsvyquantile"

```

4.1.2 Why should we stratify?

There can be two reasons for stratifying the population:

1. We are interested in the means (totals) per stratum.
2. We want to increase the precision of the estimated mean (total) for the entire population.

Figure 4.3 shows the sampling distributions of the estimator of the population mean for stratified simple random sampling and simple random sampling, both of size 40, obtained by repeating the random sampling with each design and estimation 10,000 times.

The sampling distributions of the estimators of the population mean with the two designs are not very different. With stratified random sampling the spread of the estimated means is somewhat smaller. The horizontal red line is the population mean. The gain in precision due to the stratification, referred to as the stratification effect, can be quantified by ratio of the variance with simple random sampling and the variance with stratified simple random sampling. So when this variance ratio is larger than 1, stratified simple random sampling is more precise than simple random sampling. For Voorst the stratification effect with proportional allocation (Section 4.3) equals 1.263. This means that with simple random sampling we need 1.263 more sampling units than stratified simple random sampling to obtain an estimate of the same precision.

The stratification effect is computed from the population variance $S^2(z)$ (Equation (3.7)) and the variances within the strata $S_h^2(z)$. In the sampling experiment

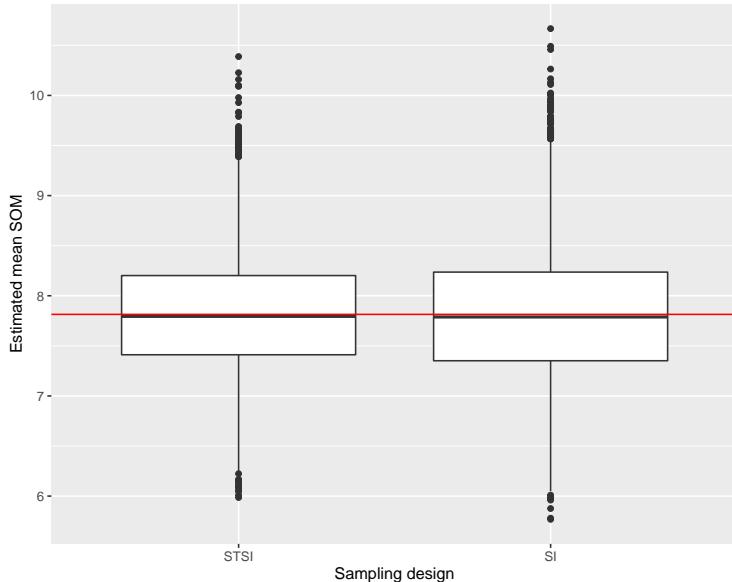


Figure 4.3: Sampling distribution of the estimator of the mean of SOM (g/kg) in Voorst for stratified simple random sampling and simple random sampling of size 40.

these variances are known without error because we know the z -values for all units in the population. In practice we only know the z -values for the sampled units. However, a design-unbiased estimator of the population variance is (?)

$$\widehat{S^2}(z) = \widehat{\bar{z}^2} - (\widehat{\bar{z}})^2 + \widehat{V}(\widehat{\bar{z}}) , \quad (4.7)$$

where $\widehat{\bar{z}^2}$ denotes the estimated population mean of the study variable squared (\bar{z}^2), obtained in the same way as $\widehat{\bar{z}}$ (Equation (4.1), but using squared values, and $\widehat{V}(\widehat{\bar{z}})$ the estimated variance of the estimator of the population mean (Equation (4.4)).

The estimated population variance is then divided by the sum of the stratum sample sizes to get an estimate of the sampling variance of the estimator of the mean with simple random sampling of an equal number of units:

$$\widehat{V}(\widehat{\bar{z}}_{SI}) = \frac{\widehat{S^2}(z)}{\sum_{h=1}^H n_h}. \quad (4.8)$$

The population variance can be estimated with function `s2` of package **surveyplanning** (?). However, this function is an implementation of an alternative, consistent estimator² of the population variance (?):

$$\widehat{S^2}(z) = \frac{N-1}{N} \frac{n}{n-1} \frac{1}{N-1} \sum_{k \in S} \frac{(z_k - \widehat{\bar{z}}_\pi)^2}{\pi_k}. \quad (4.9)$$

```
library(surveyplanning)
S2z <- s2(mysample$z, w=mysample$weight)
```

The design effect is defined as the ratio of the variance of the sampling design under study and the variance of the estimator of the mean with simple random sampling of an equal number of units (Chapter 12.4). So the design effect of stratified random sampling is the reciprocal of the stratification effect. For the stratified simple random sample of Figure 4.1 the design effect can then be estimated as follows. The function `SE` extracts the standard error of the estimated mean from the output of function `svymean`. The extracted standard error is then squared to obtain an estimate of the sampling variance of the estimator of the population with stratified simple random sampling. Finally, this variance is divided by the variance with simple random sampling of an equal number of units.

```
v_mz_SI <- S2z/n
res <- svymean(~z, design_stsi)
SE(res)^2/v_mz_SI
```

```
z
z 0.9481981
```

The same value is obtained with argument `deff` of function `svymean`.

²An estimator is consistent if the estimate becomes exactly equal to the population value when $n = N$ (?).

```

design_stsi <- svydesign(
  id=~1, strata=~stratum, weight=~weight, data=mysample)
svymean(~z, design_stsi, deff="replace")

      mean      SE    DEff
z 7.23820 0.50707 0.9482

```

So, when using package **survey** estimation of the population variance is not needed to estimate the design effect. I only added this to make clear how the design effect is computed with functions in package **survey**. In following chapters I will skip the estimation of the population variance.

The estimated design effect as estimated from the stratified sample is a bit smaller than 1, showing that stratified simple random sampling is slightly more efficient than simple random sampling. The reciprocal of the estimated design effect is considerably smaller than the stratification effect as computed in the sampling experiment, but this is an estimate of the design effect from one stratified sample only. The estimated population variance varies among stratified samples, and so does the estimated design effect.

Stratified simple random sampling with proportional allocation (Section 4.3) is more precise than simple random sampling when the sum of squares of the stratum means is larger than the sum of squares within strata (?):

$$SSB > \sum_{h=1}^H \left(1 - \frac{N_h}{N}\right) S_h^2, \quad (4.10)$$

with SSB the weighted sum-of-squares between the stratum means:

$$SSB = \sum_{h=1}^H N_h (\bar{z}_h - \bar{z})^2. \quad (4.11)$$

In other words, the smaller the differences in the stratum means and the larger the variances within the strata, the smaller the stratification effect will be. Figure 4.4 shows boxplots of SOM per stratum (soil-land use combination). The stratum means are equal to 7.94, 5.26, 6.56, 9.47, 11.12. The stratum variances are 14.5, 3.6, 9.2, 17.4, 27.6. The rather small differences in stratum means, in

combination with the large stratum variances explain the modest gain in precision realised by stratified simple random sampling compared to simple random sampling in this case.

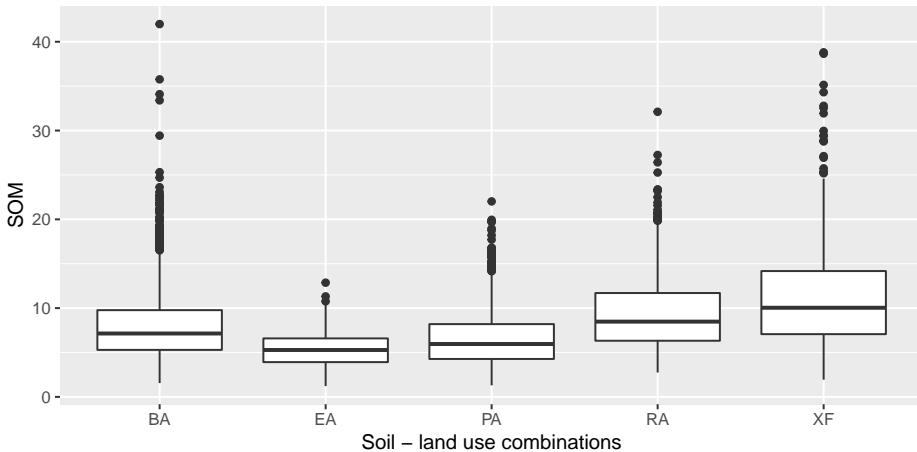


Figure 4.4: Boxplots of SOM per landuse-soil combination.

4.2 Confidence interval estimate

The $100(1 - \alpha)\%$ confidence interval for \bar{z} is given by

$$\hat{z} \pm t_{\alpha/2, df} \cdot \sqrt{\hat{V}(\hat{z})}, \quad (4.12)$$

where $t_{\alpha/2, df}$ is the value of t of a Student's t distribution with df degrees of freedom having a tail area of $\alpha/2$ to its right. In other words this is the $(1 - (\alpha/2))$ quantile of the Student's t distribution with df degrees of freedom. The degrees of freedom df can be approximated by $n - H$, as proposed by ?. This is the number of the degrees of freedom if the variances within the strata are equal. With unequal variances within strata df can be approximated by Satterwaite's method (?):

$$df \approx \frac{\left(\sum_{h=1}^H w_h^2 \frac{\widehat{S}_{h(z)}^2}{n_h} \right)^2}{\sum_{h=1}^H w_h^4 \left(\frac{\widehat{S}_{h(z)}^2}{n_h} \right)^2 \frac{1}{n_h - 1}}. \quad (4.13)$$

A confidence interval estimate of the population mean can be extracted with method `confint` of package `survey`. It uses $n - H$ degrees of freedom.

```
res <- svymean(~z, design_stsi)
df_stsi <- degf(design_stsi)
confint(res, df=df_stsi, level=0.95)

2.5 %   97.5 %
z 6.208797 8.267613
```

4.3 Allocation of sample size to strata

After we have decided on the total sample size n , we must decide how to apportion the units to the strata. It is reasonable to allocate more sampling units to large strata, and fewer to small strata. The simplest way to achieve this is proportional allocation:

$$n_h = n \cdot \frac{N_h}{\sum N_h}, \quad (4.14)$$

with N_h the total number of population units (size) of stratum h . With infinite populations N_h is replaced by the area A_h . The sample sizes computed with this equation are rounded to the nearest integers.

If we have prior information on the variance of the study variable within the strata, then it makes sense to account for differences in variance. Heterogeneous strata should receive more sampling units than homogeneous strata, leading to Neyman allocation:

$$n_h = n \cdot \frac{N_h S_h(z)}{\sum_{h=1}^H N_h S_h(z)}. \quad (4.15)$$

with $S_h(z)$ the standard deviation (square root of variance) of the study variable z in stratum h .

Finally, costs of sampling may differ between strata. It can be relatively expensive to sample nearly inaccessible strata, and we do not want to sample many units there. This leads to optimal allocation:

$$n_h = n \cdot \frac{\frac{N_h S_h(z)}{\sqrt{c_h}}}{\sum_{h=1}^H \frac{N_h S_h(z)}{\sqrt{c_h}}} , \quad (4.16)$$

with c_h the costs per sampling unit in stratum h . Optimal means that given the total costs this allocation type leads to minimum sampling variance, assuming a linear costs model:

$$C = c_0 + \sum_{h=1}^H n_h c_h . \quad (4.17)$$

with c_0 overhead costs. So the more variable a stratum and the lower the costs, the more units will be selected from this stratum.

```
S2z_h <- tapply(X=grdVoorst$z, INDEX=grdVoorst$stratum, FUN=var)
n_h_Neyman <- round(n*N_h*sqrt(S2z_h)/sum(N_h*sqrt(S2z_h)))
```

These optimal sample sizes can be computed with function `optsizes` of package `surveyplanning`.

```
labels <- sort(unique(mysample$stratum))
res <- optsizes(labels,n,N_h,S2z_h)
round(res$nh,0)
```

```
[1] 13 4 8 4 11
```

Table 4.2 shows the proportional and optimal sample sizes for the five strata of the study area Voorst, for a total sample size of 40. Stratum XF is the one-but-smallest stratum and therefore receives only seven sampling units. However, the standard deviation in this stratum is the largest, and as a consequence with

Table 4.2: Proportional and Neyman sample sizes in stratified simple random sampling of Voorst with a total sample size of 40. Nh: stratum size; Sh: stratum standard deviation

Stratum	Nh	Sh	nhprop	nhNeyman
BA	2371	3.81	12	13
EA	1442	1.89	8	4
PA	1710	3.04	9	8
RA	659	4.18	4	4
XF	1346	5.25	7	11

optimal allocation the sample size in this stratum is increased by four points, at the cost of stratum EA which is relatively homogeneous.

Figure 4.5 shows the standard error of the estimated population mean as a function of the total sample size for simple random sampling, and stratified simple random sampling with proportional and Neyman allocation for study area Voorst. A small extra gain in precision can be achieved using Neyman allocation instead of proportional allocation. However, in practice often Neyman allocation is not achievable because we do not know the standard deviations of the study variable within the strata. If a quantitative covariate x is used for stratification (see Sections 4.4 and 13.2, hereafter), the standard deviations $S_h(z)$ are approximated by $S_h(x)$, resulting in approximately optimal stratum sample sizes. The gain in precision compared to proportional allocation is then partly or entirely lost.

Optimal allocation and Neyman allocation assume univariate stratification, i.e. the stratified simple random sample is used to estimate the mean of a single study variable. If we have multiple study variables, optimal allocation becomes more complicated. In Bethel allocation the total sampling costs, assuming a linear costs model (Equation (4.17)), are minimised given a constraint on the precision of the estimated mean for each study variable (?), see Section 4.8. Bethel allocation can be computed with function `bethel` of package `SamplingStrata` (?).

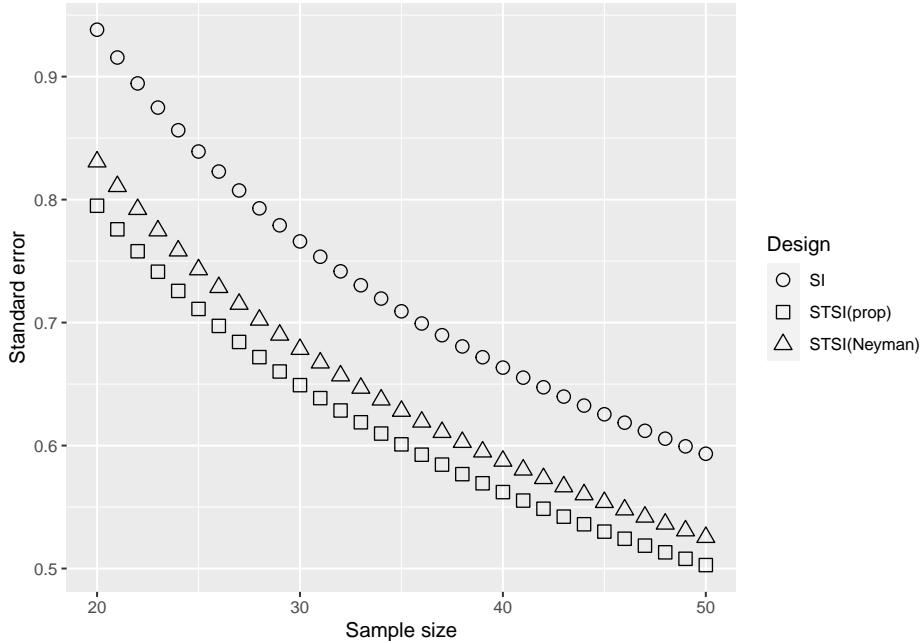


Figure 4.5: Standard error of estimated population mean as a function of the total sample size, for simple random sampling and stratified simple random sampling with proportional and Neyman allocation.

Exercises

1. Looking at Figure 4.4, which strata do you expect can be merged without losing much precision of the estimated population mean?
2. Load the data of Voorst, and use function `fct_collapse` of package `forcats` (?) to merge the strata EA and PA.
 - Compute the true sampling variance of the estimator of the mean for this new stratification, a total sample size of 40 and proportional allocation. Check that the sum of stratum sample sizes is 40. (Hint: compute the population variances of SOM per stratum, and divide these by the stratum sample sizes).

- Compare this true sampling variance with the true sampling variance using the original five strata (same sample size, proportional allocation). What is your conclusion about the new stratification?
3. Proof that the sum of the inclusion probabilities over all populations units with stratified simple random sampling equals the sample size n .

4.4 *Cum-root-f* stratification

When we have a quantitative covariate x related to the study variable z and that is known for all units in the population, strata can be constructed with the *cum-root-f* method using this covariate as a stratification variable, see ? and ?. Population units with similar values for the covariate (stratification variable) are grouped into a stratum. Strata are computed as follows:

1. Compute a histogram of the stratification variable using a large number of bins.
2. Compute the square root of the histogram frequencies.
3. Cumulate the square root of the frequencies, i.e. compute $\sqrt{f_1}$, $\sqrt{f_1} + \sqrt{f_2}$, $\sqrt{f_1} + \sqrt{f_2} + \sqrt{f_3}$, etc.
4. Divide the cumulative sum of the last bin by the number of strata, multiply this value by 1, 2, ..., $H-1$, with H the number of strata, and select the boundaries of the histogram bins closest to these values.

In *cum-root-f* stratification it is assumed that (after linear transformation) the covariate values are nearly perfect predictions of the study variable, so that the prediction errors do not affect the stratification. Under this assumption the stratification is optimal.

Cum-root-f stratification is illustrated with the data of Xuancheng (Anhui province, China). We wish to estimate the mean organic matter concentration in the topsoil (SOM, g/kg) of this area. Various covariates are available that are correlated with SOM, such as elevation, yearly average temperature, slope and various other terrain attributes. Elevation (the name of this variable in the data frame is dem) is used as a single stratification variable, see Figure 4.6. The correlation coefficient of SOM and elevation in a sample of

183 observations is 0.59. The positive correlation can be explained as follows. Temperature is decreasing with elevation, leading to smaller a decomposition rate of organic matter in the soil.

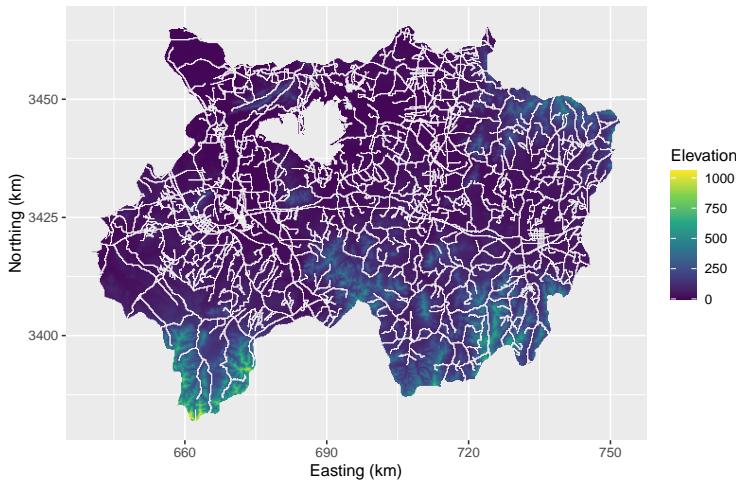


Figure 4.6: Elevation used as a stratification variable in cum-root-f stratification.

The strata can be constructed with the package **stratification** (?). Care should be taken that the data are sorted in ascending order by the columns used for stratification, see help of function **strata.cumrootf**. The argument **n** of this function is the total sample size, but this value has no effect on the stratification. The argument **Ls** is the number of strata. I arbitrarily chose to construct five strata. The argument **nclass** is the number of bins of the histogram. The output object of the function **strata.cumrootf** is a list containing amongst others a numeric vector with the stratum breaks (**bh**) and a factor with the stratum levels of the grid cells (**stratumID**). Finally, note that the values of the stratification variable must be positive. The minimum elevation is -5 m, so we added the absolute value of this minimum to elevation.

```
library(stratification)
grd <- grd[order(grd$dem),]
dem_new <- grd$dem+abs(min(grd$dem))
```

```

crfstrata <- strata.cumrootf(x=dem_new, n=100, Ls=5, nclass=500)
bh <- crfstrata$bh
grd$crfstrata <- crfstrata$stratumID

```

Stratum breaks are threshold values of the stratification variable elevation; these stratum breaks are equal to 46.8, 108.4, 216.9, 386.9. Note that the number of stratum breaks is one less than the number of strata. The resulting stratification is shown in Figure 4.7. Note that most strata are not single polygons, but are made up of many smaller ones. This may be even more so if the stratification variable shows a noisy spatial pattern. This is not a problem at all, a stratum is just a collection of population units (raster cells), and need not be spatially contiguous.

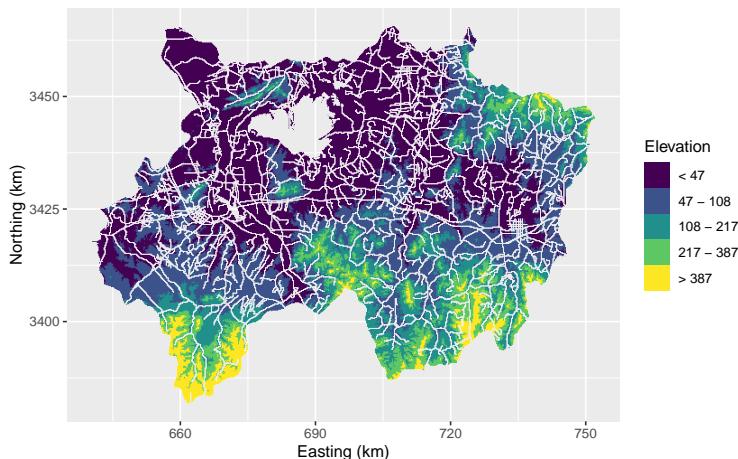


Figure 4.7: Strata obtained with cum-root-f method, using elevation as stratification variable.

Exercises

4. Write an **R** script to compute five *cum-root-f* strata for Eastern Amazonia to estimate the population mean of aboveground biomass (AGB), using log-transformed short-wave infrared (SWIR2) as stratification variable. To speed up the computations use the 5 km × 5 km subgrid sub-

sampled from the original $1 \text{ km} \times 1 \text{ km}$ grid. This subgrid is in file `data/Amazonia_5km.RData`.

- Compute ten *cum-root-f* strata, using function `strata` of package `sampling`. Sort the units first in ascending order on `lnSWIR2`. Use the stratum sample sizes as computed by the function `strata.cumrootf`. What allocation is used for computing the stratum sample sizes?
- Select a stratified simple random sample of 100 units. First compute the stratum sample sizes for proportional allocation.
- Estimate the population mean of AGB and its sampling variance.
- Compute the true sampling variance of the estimator of the mean for this sampling design (see Exercise 2 for a hint).
- Compute the stratification effect (gain in precision) (Hint: compute the sampling variance for simple random sampling by computing the population variance of AGB, and divide this by the total sample size).

4.5 Stratification with multiple covariates

If we have multiple variables that are possibly related to the study variable, we may want to use them all or a subset of them as stratification variables. Using the quantitative variables one-by-one in *cum-root-f* stratification, followed by overlaying the maps with univariate strata, may lead to numerous cross-classification strata.

A simple solution is to construct homogeneous groups, referred to as clusters, of population units (raster cells). The units within a cluster are more similar to each other than to the units in other clusters. Various clustering techniques are available. Here I use hard k -means.

This is illustrated again with the Xuancheng case study. Five quantitative covariates are used for constructing the strata. Besides elevation which was used as a single stratification variable in the previous section, now also temperature, slope, topographic wetness index (twi) and profile curvature are used to construct clusters that are used as strata in stratified simple random sampling. To speed up the computations a subgrid with a spacing of 400 m is selected, using

function `spsample` of package `sp`, see Chapter 5 (?).

```
library(sp)
gridded(grd) <- c("x1", "x2")
subgrd <- spsample(
  grd, type="regular", cellsize=0.4, offset=c(0.5,0.5))
subgrd <- data.frame(coordinates(subgrd), over(subgrd,grd))
```

Five clusters are computed with k -means using as clustering variables elevation (dem), temperature, slope, profile curvature and topographic wetness index (twi). The scale of the five covariates is largely different, and for that reason they must be scaled before being used in clustering. The k -means algorithm is a deterministic algorithm, i.e. the same initial clustering will end in the same final, optimised clustering. This final clustering can be suboptimal, and therefore it is recommended to repeat the clustering as many times as feasible, with different initial clusterings. Argument `nstart` is the number of initial clusterings. The best clustering, i.e. the one with the smallest within-cluster sum-of-squares, is kept.

```
x <- c("dem", "temperature", "slope", "profile.curvature", "twi")
set.seed(314)
myClusters <- kmeans(
  scale(subgrd[,x]), centers=5, iter.max=1000, nstart=100)
subgrd$cluster <- myClusters$cluster
```

Figure 4.8 shows the five strata obtained by the k -means clustering of the raster cells.

The size of the clusters (strata) is largely different among the strata (Table 4.3). This table also shows means of the unscaled covariates used in clustering.

In the situation that we already have some data of the study variable, an alternative solution is to calibrate a model for the study variable, for instance a multiple linear regression model, using the covariates as predictors, and to use the predictions of the study variable as a single stratification variable in *cum-root-f* stratification or in optimal spatial stratification, see Section 13.2.

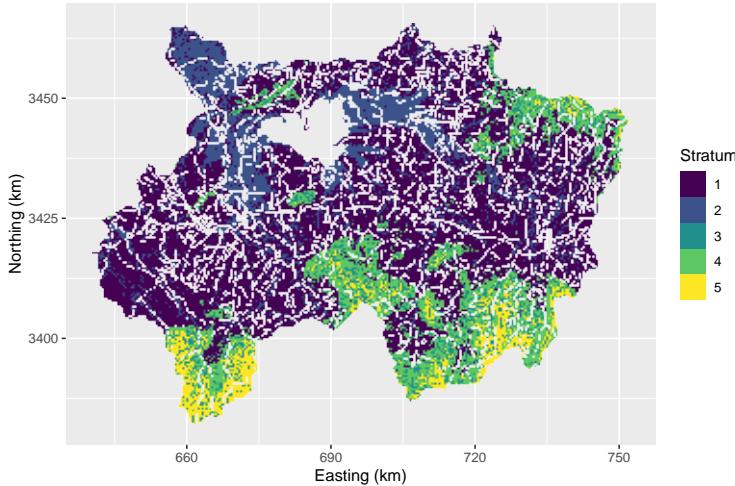


Figure 4.8: Five k -means clusters using five scaled covariates in clustering, to be used as strata in random sampling from study area Xuancheng.

Table 4.3: Total number of raster cells (N_h) and means of clustering variables of the five strata in Xuancheng obtained with k -means clustering of raster cells.

Stratum	N_h	Elevation	Temperature	Slope	Profilecurv	Twi
1	16032	53	15.44	2.09	0.00001	9.25
2	5312	19	15.60	0.47	0.00006	17.16
3	1650	308	14.43	12.78	-0.00143	6.29
4	4184	176	14.66	10.67	0.00066	7.78
5	1710	401	13.74	20.90	0.00019	6.54

4.6 Geographical stratification

When no covariate is available, we may still decide to apply a *geographical stratification*. For instance, a square study area can be divided into 4×4 equally sized subsquares that are used as strata. When we select one or two points per subsquare, we avoid strong spatial clustering of the sampling points. Geographical stratification improves the *spatial coverage*. When the study variable is spatially structured, think for instance of a spatial trend, then geographical stratification

will lead to more precisely estimated means (smaller sampling variances).

A simple method for constructing geographical strata is k -means clustering (?). See Chapter 18 for a simple illustrative example of how geographical strata are computed with k -means clustering. In this approach the study area is discretised by a large number of pixels (raster cells). These pixels are the objects that are clustered. The clustering variables are simply the s1-coordinate (Easting) and s2-coordinate (Northing) of the centers of the pixels. This method leads to compact geographical strata, shortly referred to as geostrata. Geostrata can be computed, as in Section 4.5, with function **kmeans**. The two clustering variables have the same scale, so they should not be scaled because this would lead to an arbitrary distortion of geographical distances. The geostrata generally will not have the same area (number of pixels). Geostrata of equal area can be attractive, as then the sample becomes selfweighting, i.e. the sample mean is an unbiased estimator of the population mean.

Geostrata of the same area can be computed with function **stratify** of the package **spcosa** (?), with argument **equalArea=TRUE**³. ? describe the k -means algorithms implemented in this package in detail. The argument **object** of function **stratify** specifies a spatial object of the population units. In the R code below the data frame **grdVoorst** is changed into a **SpatialPixelsDataFrame** with function **gridded** of the package **sp**. The spatial object can also be of class **SpatialPolygons**. In that case either argument **nGridCells** or argument **cellSize** must be set, so that the vector map in **object** can be discretised by a finite number of grid cells. Argument **nTry** specifies the number of initial stratifications in k -means clustering, and so is comparable with argument **nstart** of function **kmeans**. For more details on spatial stratification using k -means clustering, see Chapter 18. The k -means algorithm used with **equalArea=TRUE** takes much more computing time than the one used with **equalArea=FALSE**.

```
library(spcosa)
library(sp)
set.seed(314)
gridded(subgrd) <- ~x1+x2
mygeostrata <- stratify(
  object=subgrd, nStrata=50, nTry=1, equalArea=TRUE)
```

³If the total number of pixels divided by the number of strata is an integer, the stratum sizes are exactly equal, otherwise the difference is 1 pixel.

Function **spsample** of package **spcosa** is used to select from each geostratum a simple random sample of two points.

```
set.seed(314)
mysample <- spcosa::spsample(mygeostrata, n=2)
mysample <- as(mysample, "data.frame")
mygeostrata <- as(mygeostrata, "data.frame")
```

The operator **%>%** of package **magrittr** (?) can be used to merge the two lines in the code chunk above. In this way we can save memory, as we do not need an object of the class **SamplingPatternRandomSamplingUnits** obtained with function **spsample**.

```
library(magrittr)
mysample <- spcosa::spsample(mygeostrata, n=2) %>% as(., "data.frame")
```

Figure 4.9 shows 20 compact geostrata of equal area of Voorst with the selected sampling points. Note that the sampling points are reasonably well spread throughout the study area.

Once the observations are done, the population mean can be estimated with function **estimate**. For Xuancheng I simulated data from a normal distribution, to illustrate estimation with function **estimate**. Various statistics can be estimated, among which the population mean (spatial mean), the standard error, and the cumulative distribution function (CDF). The CDF is estimated by transforming the data to indicators (Section 3.1.2).

```
library(spcosa)
load(file="results/geostrata_Xuancheng.RData")
mysample <- spcosa::spsample(mygeostrata, n=2)
mydata <- data.frame(z=rnorm(100, mean=10, sd=2))
mean <- estimate(
  "spatial mean", mygeostrata, mysample, data=mydata)
se <- estimate(
  "standard error", mygeostrata, mysample, data=mydata)
cdf <- estimate(
  "scdf", mygeostrata, mysample, data=mydata)
```

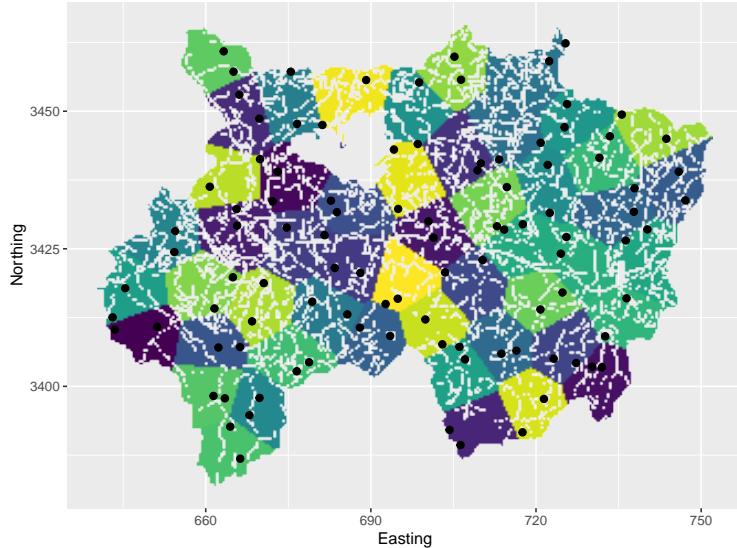


Figure 4.9: Compact geostrata of equal size in Xuancheng, and stratified simple random sample of two points per stratum.

The estimated population mean equals 9.82, with an estimated standard error of 0.192.

Exercises

5. Why is it attractive to select at least two points per geostratum?
6. The alternative to 20 geostrata and two points per geostratum is 40 geostrata and one point per geostratum. Which sampling strategy will be more precise?
7. The geostrata in the figure above have equal area, which can be enforced by argument `equalArea=TRUE`. Why are equal areas attractive? Work out the estimator of the population mean for strata of equal size.
8. Write an **R** script to construct 20 compact geographical strata of equal area for agricultural field Leest. Read the shapefile `Leest5` using function

`readOGR` of the package **rgdal**. Remove the projection attributes with `proj4string(shpField) <- NA_character_`. Select two points per geostratum, using function `spsample` of package **spcosa**. Repeat this with 40 strata of equal area, and randomly select one point per stratum.

- If only one point per stratum is selected, the sampling variance can be approximated by the collapsed strata estimator. In this method pairs of strata are formed, and the two strata of a pair are joined. In each new stratum we now have two points. With an odd number of strata there will be one group of three strata and three points. The sample is then analyzed as if it were a random sample from the new collapsed strata. Suppose we group the strata on the basis of the measurements of the study variable. Do you think this is a proper way of grouping?
 - In case you think this is not a proper way of grouping the strata, how would you group the strata?
 - Will the estimated sampling variance estimator be unbiased? If not, will it be overestimated or underestimated?
9. Laboratory costs for measuring the study variable can be saved by bulking the soil aliquots (composite sampling). There are two options: bulking all soil aliquots from the same stratum (bulking within strata) or bulking by selecting one aliquot from each stratum (bulking across strata). In **spcosa** bulking across strata is implemented. Write an **R** script to construct 20 compact geographical strata for study area Voorst. Use the argument `equalArea = TRUE`. Select four points per stratum using argument `type="composite"`, and change class of resulting object in **SpatialPoints**. Extract the z-values in `grdVoorst` at the selected sampling points using function `over`. Add a column to the resulting data frame indicating the composite (points 1 to 4 are from the first stratum, points 5 to 8 from the second stratum, etc.), and estimate the means for the four composites using function `tapply`. Estimate the population mean and its standard error.
 - Can the sampling variance of the estimator of the mean be estimated for bulking within the strata?
 - The alternative to analyzing the concentration of four composite samples obtained by bulking across strata is to analyze all 20×4

aliquots separately. The strata have equal area, so the inclusion probabilities are equal. As a consequence the sample mean is an unbiased estimator of the population mean. Is the precision of this estimated population mean equal to the estimated population mean with composite sampling? If not, is it smaller or larger, and why?

- If you use argument `equalArea = FALSE` in combination with argument `type="composite"`, you get an error message. Why does this not work?

4.7 Multi-way stratification

In Section 4.5 multiple continuous covariates are used to construct clusters of raster cells using k-means. These clusters are then used as strata. This section considers the case where we have multiple categorical and/or continuous variables that we would like to use as stratification variables. The continuous stratification variables are first used to compute strata based on that stratification variable, e.g. using the *cumroot-f* method. What could be done then is to compute the cross-classification of each unit, and use these cross-classifications as strata in random sampling. However, this may lead to numerous strata, maybe even more than the intended sample size. To reduce the total number of strata, we may aggregate cross-classification strata with similar means of the study variable, based on our prior knowledge.

An alternative to aggregation of cross-classification strata is to use the separate strata, i.e. the strata based on an individual stratification variable, as *marginal* strata in random sampling, instead of using the cross-classifications as strata. How this works is explained in Section 9.1.4.

4.8 Multivariate stratification

Another situation is where we have multiple study variables, and we would like to optimise the stratification and allocation for estimating the population means of all study variables. Optimal stratification for multiple study variables is only relevant if we would like to use different stratification variables for the study variables. In many cases we do not have reliable prior information about the different study variables justifying the use of multiple stratification variables. We are already happy to have one stratification variable that may serve to

increase the precision of the estimated means of all study variables.

However in case we do have multiple stratification variables, tailored at different study variables, the aim is to partition the population in strata, so that for a given allocation, the total sampling costs, assuming a linear costs model (Equation (4.17)), is minimised given a constraint on the precision of the estimated mean for each study variable.

Package **SamplingStrata** (?) can be used to optimise multivariate strata. ? gives details about the objective function and the algorithm used for optimising the strata. Sampling units are allocated to the strata by Bethel allocation (?). The required precision is specified in terms of a coefficient of variation, one per per study variable.

Multivariate stratification is illustrated with the Meuse data set of package **gstat** (?). The prior data of heavy metal concentrations of Cd and Zn, are used in spatial prediction, to create maps of these two study variables. These predictions of the study variables are used as stratification variables in designing a new sample for design-based estimation of the population means of Cd and Zn.

The maps of natural logarithms of the two metal concentrations are created by kriging with an external drift, using the square root of the distance to the Meuse river as a predictor for the mean, see Section 22.2 for how this spatial prediction method works.

Figure 4.10 shows the map with the predicted log Cd and log Zn concentration.

The predicted log concentrations of the two heavy metal concentrations are used as stratification variables. For the log of Cd there are negative predicted concentrations (Figure 4.10). This leads to an error when running function **optimStrata**. The minimum predicted log Cd concentration is -1.7, so I added 2 to the predictions. A variable indicating the domains of interest is added to the data frame. The value of this variable is 1 for all grid cells, so that a sample is designed for estimating the mean of the entire population. As a first step function **buildFrameDF** is used to create a data frame that can be handled by function **optimStrata**. Argument X specifies the stratification variables, and argument Y the study variables. In our case the stratification variables and the study variables are the same. This is typical for the situation where the stratification variables are obtained by mapping the study variables.

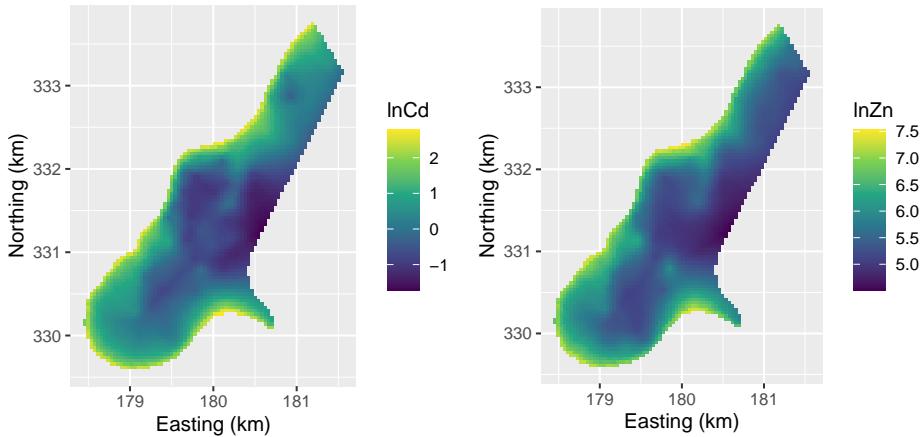


Figure 4.10: Kriging predictions of natural logarithms of Cd and Zn concentration in study area Meuse (Netherlands), used as stratification variable in bivariate stratification.

```
library(SamplingStrata)
df <- data.frame(cd=lcd_kriged$var1.pred,
                  zn=lzn_kriged$var1.pred)
df$cd <- df$cd+2
df$dom <- rep(1,nrow(df))
df$id <- c(1:nrow(df))
frame <- buildFrameDF(
  df=df, id="id",
  X=c("cd", "zn"), Y=c("cd", "zn"),
  domainvalue="dom")
```

Next, a data frame with the precision requirements for the estimated means is created. The precision requirement is given as a coefficient of variation, i.e. the standard error of the estimated population mean, divided by the estimated mean. The study variables as specified in `Y` are used to compute estimated means and the standard errors for a given stratification and allocation.

```
cv <- as.data.frame(
  list(DOM="DOM1", CV1=0.02, CV2=0.02, domainvalue=1))
```

Finally, the multivariate stratification is optimised, by optimising the stratum bounds, using a genetic algorithm (?).

```
set.seed(314)
res <- optimStrata(
  method="continuous", errors=cv, framesamp=frame, nStrata=5,
  iter=50, pops=20, showPlot=FALSE)
```

A summary of the strata can be obtained with function `summaryStrata`.

```
smrstrata <- summaryStrata(
  res$framenew, res$aggr_strata, progress=FALSE)
```

	Stratum	Population	Allocation	Lower_X1	Upper_X1	Lower_X2	Upper_X2
1	1	717		7	0.266	1.421	4.502
2	2	694		5	1.421	2.090	4.950
3	3	597		3	2.091	2.630	5.163
4	4	704		6	2.630	3.476	5.472
5	5	391		5	3.480	4.781	6.234

The column `Population` is the size of the strata (number of pixels). The total sample size equals 26. The sample sizes per stratum are computed with Bethel allocation, see Section 4.3. The last four columns contain the lower and upper bounds of the orthogonal intervals.

Figure 4.11 shows a 2D-plot of the bivariate strata. The strata can be plotted as a series of nested rectangles. All population units in the smallest rectangle belong to stratum 1; all units in the one-but-smallest rectangle that are not in the smallest rectangle belong to stratum 2, etc. If we have more than two stratification variables the strata form a series of nested hyperrectangles or boxes. The strata are obtained as the Cartesian product of orthogonal intervals.

```
plt <- plotStrata2d(
  res$framenew, res$aggr_strata, domain=1,
  vars=c("X1", "X2"), labels=c("Cd", "Zn"))
```

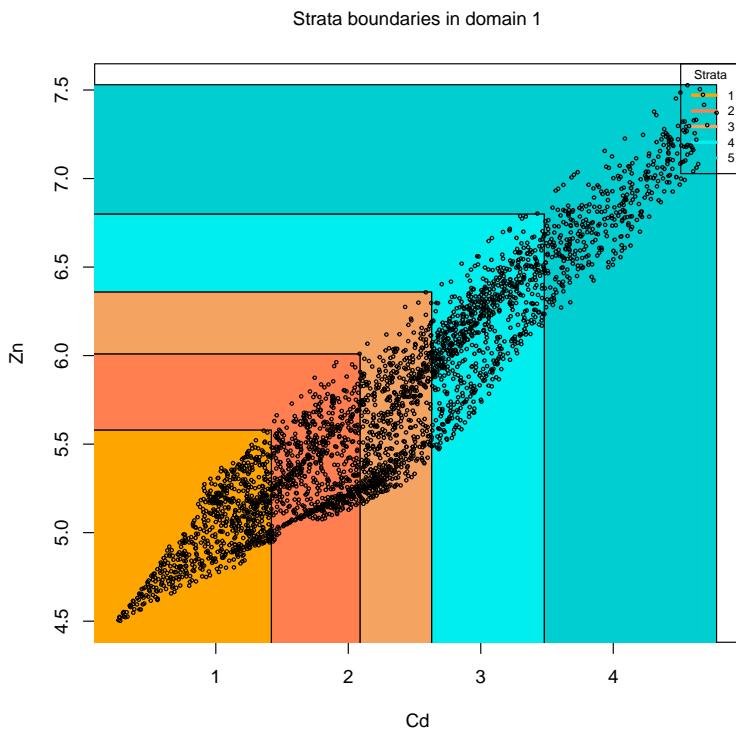


Figure 4.11: Optimised bivariate strata for study area Meuse.

It may happen that during the optimisation of the stratum bounds in some resulting strata no units are contained. If the solution with a smaller number of strata requires fewer sampling units, then this is retained as the optimal stratification (personal communication Giulio Barcaroli).

Figure 4.12 shows a map of the optimised strata.

The expected coefficient of variation can be extracted with function `expected_CV`.

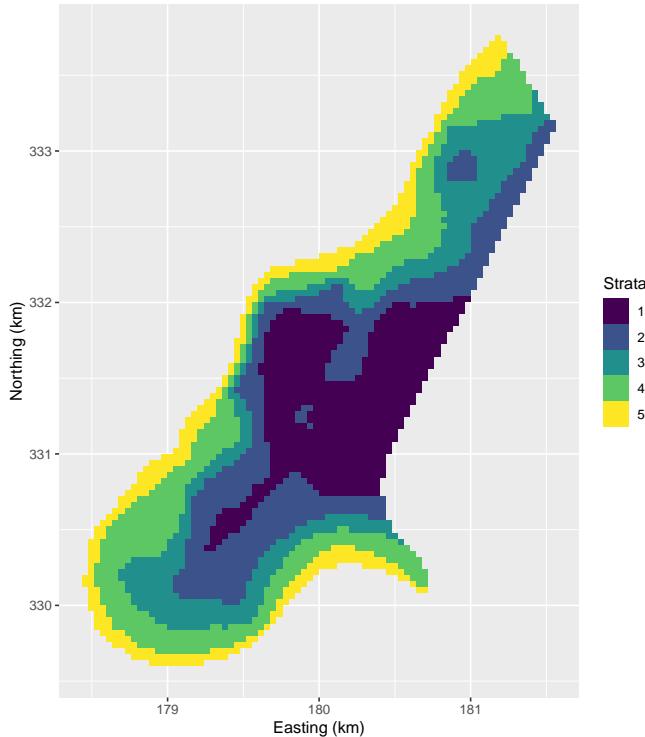


Figure 4.12: Optimised bivariate strata for study area Meuse.

```
expected_CV(res$aggr_strata)
```

```
cv(Y1) cv(Y2)
DOM1 0.02 0.009
```

The coefficient of variation of Cd is indeed equal to the desired level of 0.02, for Zn it is smaller. So in this case Cd is the study variable that determines the total sample size of 26 units.

Note that these coefficients of variation are computed from the stratification variables, which are predictions of the study variable. Errors in these predictions are not accounted for. It is well-known that kriging is a smoother, so that the

variance of the predicted values within a stratum is smaller than the variance of the true values. As a consequence the coefficient of variation underestimates the coefficient of variation of the study variable. See Section 13.2 for how prediction errors and spatial correlation of prediction errors can be accounted for in optimal stratification. An additional problem is that I added a value of 2 to the log Cd concentrations. This does not affect the standard error of the estimated mean, but does effect the estimated mean, so that also for this reason the coefficient of variation is underestimated.

Chapter 5

Systematic random sampling

A simple way of drawing probability samples whose units are spread uniformly over the study area, is systematic random sampling (SY). Systematic random sampling from a two-dimensional spatial population entails sampling on a regular grid. A systematic sample can be selected with function `spsample` of package `sp` with argument `type = "regular"` (?). The argument `offset` is not used, so that the grid is randomly placed on the study area. This is illustrated, as in the previous chapters, with Voorst.

```
load("data/Voorst.RData")
set.seed(314)
#change class of grdVoorst to SpatialPixelsDataFrame
gridded(grdVoorst) <- ~s1+s2
n <- 40
mySYsample <- spsample(x=grdVoorst, n=n, type="regular") %>%
  as(., "data.frame")
```

Figure 5.1 shows the randomly selected systematic sample. The shape of the grid square, and the orientation is E-W, N-S. There is no strict need for random selection of the orientation of the grid. Random placement of the grid on the

study area suffices for design-based estimation.

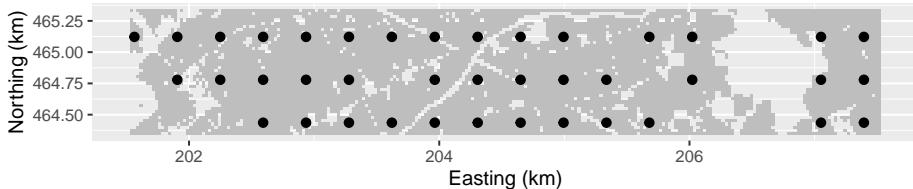


Figure 5.1: Systematic random sample (randomly placed square grid) from Voorst.

The argument `n` in function `spsample` is used to set the sample size. Note that this is the *expected* sample size, i.e. on average, over repeated sampling the sample size is 40. In Figure 5.1 the number of selected sampling points equals 40, but this is a lucky shot only. Given the *expected* sample size, the spacing of the square grid can be computed with $\sqrt{A/n}$, with A the area of the study area. This area A can be computed by the total number of pixels multiplied by the pixel area. Function `getGridTopology` is used to retrieve the cell size of the pixels of the `SpatialPixelsDataFrame`. Note that this area is smaller than the number of pixels in the horizontal direction, multiplied by the number of pixels in the vertical direction, multiplied by the pixel area, as we have non-availables (built-up areas, roads, etc.).

```
gridtop <- as(getGridTopology(grdVoorst), "data.frame")
A <- nrow(grdVoorst)*gridtop$cellsize[1]*gridtop$cellsize[2]
(spacing <- sqrt(A/n))
```

```
[1] 342.965
```

Instead of argument `n` we may use argument `cellsize` to select a grid with a specified spacing. The expected sample size of a square grid can then be computed with $A/\text{spacing}^2$.

The spatial coverage with random grid sampling is better than with a stratified random sample using compact geographical strata (Section 4.6), even with one sampling unit per geostratum. Consequently, in general systematic random sampling results in more precise estimates of the mean or total.

However, there are also two disadvantages of systematic random sampling com-

pared to geographically stratified random sampling. First, for systematic random sampling no design-unbiased estimator of the sampling variance exists. Second, the number of sampling units with random grid sampling is not fixed, but varies among randomly drawn samples. We may choose the grid spacing such that *on average* the number of sampling units equals the required (allowed) number of sampling units, but for the actually drawn sample, this number can be smaller or larger. In Voorst the variation of the sample size is quite large. The histogram shows a bimodal distribution (Figure 5.2). The smaller sample sizes are of square grids with only two East-West oriented rows of points instead of three rows.

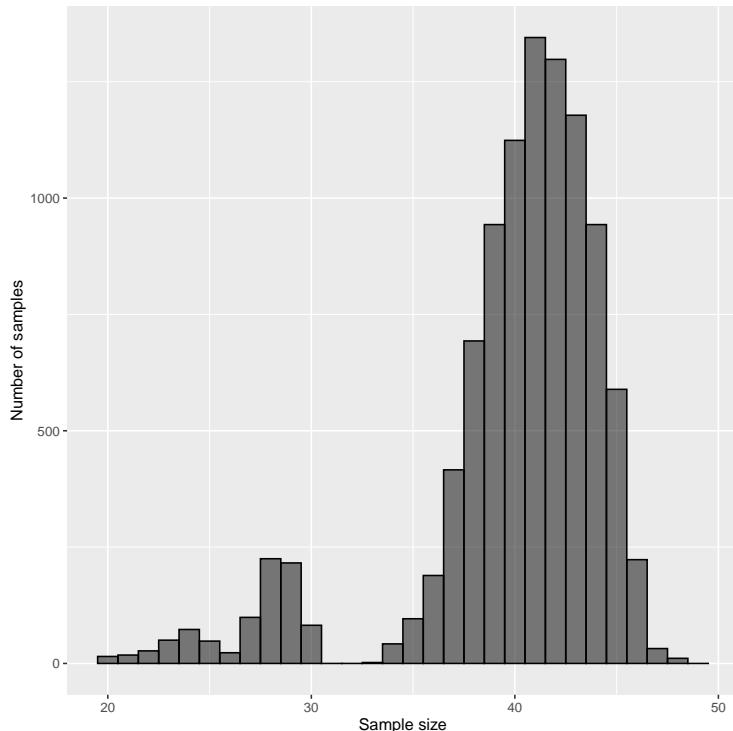


Figure 5.2: Sampling distribution of sample size of systematic random sampling.

A large variation in sample size, if the sampling with the sampling design under

study would be repeated, is undesirable and should be avoided when possible. In this case a simple solution is to select a rectangular grid instead of a square grid, with a spacing in the North-South direction that results in a fixed number of East-West oriented rows of sampling points over repeated selection of grids. This is achieved with a North-South spacing equal to the dimension of the study area in North-South direction divided by an integer. The spacing in East-West direction is then adapted so that on average a given number of sampling points is selected. The North-South dimension of the study area is 1,000 m. A North-South spacing of $1,000/3$ m is chosen, so that the number of East-West oriented rows of sampling points in the systematic sample equals three.

```
dy <- 1000/3
dx <- A/(n*dy)
mySYSsample_rect <- spsample(
  x=grdVoorst, cellsize=c(dx,dy), type="regular")
```

The East-West spacing is somewhat larger than the North-South spacing: 352.875 m. The variation in sample size with the random rectangular grid is much smaller than that of the square grid.

```
summary(sampleSizes)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
33.00	38.00	40.00	39.99	42.00	46.00

An alternative shape for the grid is triangular. Triangular grids can be selected with the argument `type = "hexagonal"`¹. The triangular grid was shown to be yield most precise estimates of the population mean given the expected sample size (?). Given the spacing of a triangular grid, the expected sample size can be computed by the area A of the study area divided by the area of hexagonal grid cells with the sampling points at their centers. The area of a hexagon equals $6\sqrt{3}/4 r^2$, with r the radius of the circle circumscribing the hexagon (distance from center to a corner of the hexagon). So by choosing a radius of $\sqrt{A/(6\sqrt{3}/4)} n$ the expected sample equals n . The distance between neighbouring points of the triangular grid in the East-West direction then equals $r\sqrt{3}$. The North-South distance equals $\sqrt{3}/2 dx$.

¹The centers of hexagonal grid cells form a triangular grid.

```
cnst <- 6*sqrt(3)/4
r <- sqrt(A/(cnst*n))
dx <- r*sqrt(3)
dy <- sqrt(3)/2*dx
```

Function `spsample` does not work properly in combination with argument `type="hexagonal"`. Over repeated sampling the average sample size is not equal to the chosen sample size specified with argument `n`. The same problem remains when using argument `cellsize`.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
18.00	23.00	26.00	28.39	35.00	41.00

The following code can be used for random selection of triangular grids.

```
SY_triangular <- function(dx, grd) {
  dy <- sqrt(3)/2*dx
  # randomly select offset
  offset_x <- runif(1, min=0, max=dx)
  offset_y <- runif(1, min=0, max=dy)
  #compute x-coordinates of 1 row and y-coordinates of 1 column
  bbox <- bbox(grd)
  nx <- ceiling((bbox[1,2]-bbox[1,1])/dx)
  ny <- ceiling((bbox[2,2]-bbox[2,1])/dy)
  x <- (-1:nx)*dx+offset_x
  y <- (0:ny)*dy+offset_y
  #compute coordinates of rectangular grid
  xy <- expand.grid(x, y)
  names(xy) <- c("x", "y")
  #shift points of even rows in horizontal direction
  units <- which(xy$y %in% y[seq(from=2, to=ny, by=2)])
  xy$x[units] <- xy$x[units] + dx/2
  #add coordinates of origin
  xy$x <- xy$x+bbox[1,1]
  xy$y <- xy$y+bbox[2,1]
  #overlay with grid
  coordinates(xy) <- ~x+y
  mysample <- data.frame(coordinates(xy), over(xy, grd))
```

```

#delete points with NA
mysample <- mysample[!is.na(mysample[,3]),]
}
set.seed(314)
mySYsample_tri <- SY_triangular(dx=dx, grd=grdVoorst)

```

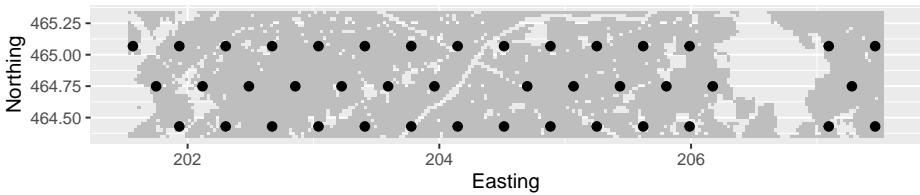


Figure 5.3: Systematic random sample (random triangular grid) from Voorst.

5.1 Estimation of population parameters

With systematic random sampling all units have an equal inclusion probability, equal to $E[n]/N$, with $E[n]$ the expected sample size. Consequently, the population total can be estimated by

$$\hat{t}(z) = \sum_{k \in S} \frac{z_k}{\pi_k} = N \sum_{k \in S} \frac{z_k}{E[n]} . \quad (5.1)$$

The population mean can be estimated by dividing this π estimator of the population total by the population size:

$$\hat{\bar{z}} = \sum_{k \in S} \frac{z_k}{E[n]} . \quad (5.2)$$

In this π estimator of the population mean the sample sum of the observations is not divided by the number of selected units, but by the expected number of units.

An alternative estimator is obtained by dividing the π estimator of the population total by the π estimator of the population size:

$$\hat{N} = \sum_{k \in S} \frac{1}{\pi_k} = n \frac{N}{E[n]} . \quad (5.3)$$

This yields the ratio estimator of the population mean:

$$\hat{\bar{z}}_{\text{ratio}} = \frac{1}{n} \sum_{k \in S} z_k . \quad (5.4)$$

So the ratio estimator of the population total is equal to the unweighted sample mean. The variance of this ratio estimator is in general smaller than that of the π estimator. On the other side the π estimator is design-unbiased, whereas the ratio estimator is not, although this bias can be negligibly small. Only in the very special case where the sample size with systematic random sampling is fixed, the two estimators are equivalent.

Recall that for Voorst we have exhaustive knowledge of the study variable z : values of SOM were simulated for all pixels. To determine the z -values at the selected sampling points first an overlay of the systematic random sample and the `SpatialPixelsDataFrame` is made, using function `over` of package `sp`.

```
set.seed(1956)
mySYsample <- spsample(x=grdVoorst, n=n, type="regular")
res <- over(mySYsample, grdVoorst)
mySYsample <- as(mySYsample, "data.frame")
mySYsample$z <- res$z
mz_HT <- sum(mySYsample$z)/n
mz_ratio <- mean(mySYsample$z)
```

The π estimated population mean equals 8.197, the ratio estimate equals 7.626. The ratio estimate is slightly smaller because the size of the selected sample is one unit larger than the expected sample size.

5.1.1 Approximating the sampling variance of the estimator of the mean

An unbiased estimator of the sampling variance of the estimator of the mean is not available. A simple, often applied procedure is to calculate the sampling

variance as if the sample were a simple random sample (Equation (3.2)). In general this procedure overestimates the sampling variance, so that we are on the safe side.

```
av_SI_mz <- var(mySYsample$z)/nrow(mySYsample)
```

The approximated variance equals 0.441.

Alternatively, the sampling variance can be estimated by treating the systematic random sample as if it were a stratified simple random sample (Equation (4.1)). The sampling units are clustered on the basis of their spatial coordinates into $H = n/2$ clusters (n even) or $H = (n - 1)/2$ clusters (n odd). In the next code chunk a simple k -means function is defined to cluster the sampling units of the grid into equal sized clusters. Arguments $s1$ and $s2$ are the spatial coordinates of the sampling units, k is the number of clusters. As a first step in the function the ids of equal-sized clusters are randomly assigned to the sampling units on the nodes of the grid (initial clustering), and the centers of the clusters are computed, i.e. the means of the spatial coordinates of the clusters are computed (initial cluster centers). There are two for-loops. In the inner-loop it is determined whether the cluster id of the unit selected in the outer-loop should be swapped with that of the next unit. If both units have the same cluster id the next unit is selected, until a unit of a different cluster is found. The cluster ids of the two units are swapped when the sum of the squared distances of the two units to their corresponding cluster centers is reduced. When the cluster ids are swapped, the centers are recomputed. The two loops are repeated until no swaps are made anymore.

```
.kmeans_equal_size <- function(s1, s2, k) {
  n <- length(s1)
  cluster_id <- rep(1:k, times=ceiling(n/k))
  cluster_id <- cluster_id[1:n]
  cluster_id <- cluster_id[sample.int(n, size=n)]
  s1_c <- tapply(s1, INDEX=cluster_id, FUN=mean)
  s2_c <- tapply(s2, INDEX=cluster_id, FUN=mean)
  repeat {
    n_swop <- 0
    for (i in 1:(n-1)) {
      ci <- cluster_id[i]
```

```

for (j in (i+1):n) {
  cj <- cluster_id[j]
  if(ci==cj) {next}
  d1 <- (s1[i] - s1_c[ci])^2 + (s2[i] - s2_c[ci])^2 +
    (s1[j] - s1_c[cj])^2 + (s2[j] - s2_c[cj])^2
  d2 <- (s1[i] - s1_c[cj])^2 + (s2[i] - s2_c[cj])^2 +
    (s1[j] - s1_c[ci])^2 + (s2[j] - s2_c[ci])^2
  if (d1 > d2) {
    #swop cluster ids and recompute cluster centers
    cluster_id[i] <- cj; cluster_id[j] <- ci
    s1_c <- tapply(s1, cluster_id, mean)
    s2_c <- tapply(s2, cluster_id, mean)
    n_swop <- n_swop + 1
    break
  }
}
}
if(n_swop==0) {break}
}

D <- fields::rdist(x1=cbind(s1_c,s2_c), x2=cbind(s1,s2))
dmin <- apply(D, MARGIN=2, FUN=min)
MSSD <- mean(dmin^2)
list(clusters=cluster_id, MSSD=MSSD)
}

```

The clustering is repeated 100 times (`ntry=100`). The clustering with the smallest sum of the squared distances of the sampling units to their cluster centers (MSSD) is selected.

```

kmeans_equal_size <- function(s1, s2, k, ntry) {
  res_opt <- NULL
  MSSD_min <- Inf
  for (i in 1:ntry) {
    res <- .kmeans_equal_size(s1, s2, k)
    if (res$MSSD < MSSD_min) {
      MSSD_min <- res$MSSD
      res_opt <- res
    }
  }
  return(res_opt)
}

```

```

        }
    }
    res_opt
}
n <- nrow(mySYsample); k <- floor(n/2)
set.seed(314)
res <- kmeans_equal_size(
  s1=mySYsample$x1/1000, s2=mySYsample$x2/1000,
  k=k, ntry=100)
mySYsample$cluster <- res$clusters

```

Figure 5.4 shows the clustering of the systematic random sample of Figure 5.1. The two (or three) sampling units of a cluster are then treated as a simple random sample from a stratum, and the variance estimator for stratified random sampling is used. With n even the stratum weights are $1/H$ for all strata, with n odd the weights are computed as $w_h = n_h/n$. For more details on variance estimation with stratified simple random sampling, I refer to Section 4.1.

```

S2z_h <- tapply(mySYsample$z, INDEX=mySYsample$cluster, FUN=var)
nh <- tapply(mySYsample$z, INDEX=mySYsample$cluster, FUN=length)
v_mz_h <- S2z_h/nh
w_h <- nh/sum(nh)
av_STSI_mz <- sum(w_h^2*v_mz_h)

```

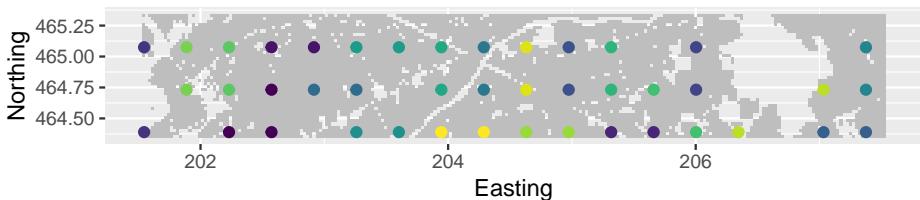


Figure 5.4: Clustering of grid points selected from Voorst for approximating the variance of the estimator of the population mean of SOM.

This method yields an approximated variance of 0.462, which is considerably smaller than the simple random sample approximation.

A similar approach for approximating the variance is proposed a long time ago by Matérn (?). In this approach the variance is approximated by computing the squared difference of two local means. A local mean is computed by linear interpolation of the observations at the two nodes on the diagonal of a square grid cell. The four corners of a square grid cell serve as a group. Every grid node belongs to four groups and so the observation at a grid node is used four times in computing a local mean. Near the edges of the study area we have incomplete groups: one, two or even three observations are missing. To compute a squared difference these missing values are replaced by the sample mean. This results in as many squared differences as we have groups. Note that the number of groups is larger than the sample size. The squared differences are computed by

$$\begin{aligned} d_{r,s}^2 &= \left(\frac{z_{r,s} + z_{r+1,s+1}}{2} - \frac{z_{r+1,s} + z_{r,s+1}}{2} \right)^2 \\ &= \frac{(z_{r,s} - z_{r+1,s} - z_{r,s+1} + z_{r+1,s+1})^2}{4}, \end{aligned} \quad (5.5)$$

with $r = 0, 1, \dots, R$ an index for the column-number and $s = 0, 1, \dots, S$ an index for the row-number of the extended grid. The variance of the estimator of the mean (sample mean) is then approximated by the sum of the squared differences divided by the squared sample size:

$$\widehat{V}(\bar{z}_S) = \frac{\sum_{g=1}^G d_g^2}{n^2}, \quad (5.6)$$

with d_g^2 the squared difference of group unit g , and G the total number of groups.

To approximate the variance with Matérn's method a function is defined.

Before using this function the data frame with the sample data must be extended with two variables: an index i for the column number, and index j for the row number of the square grid.

```
mySYsample <- mySYsample %>%
  mutate(i=round((x1-min(x1))/spacing),
        j=round((x2-min(x2))/spacing))
matern(mySYsample)
```

```
[1] 0.5007008
```

Figure 5.5 shows the sampling distributions of the estimator of the population mean for systematic random sampling, using a randomly placed square grid with fixed orientation and an expected sample size of 40, and simple random sampling, obtained by repeating the random sampling with each design and estimation 10,000 times. To estimate the population mean from the systematic random samples both the π estimator and the ratio estimator are used.

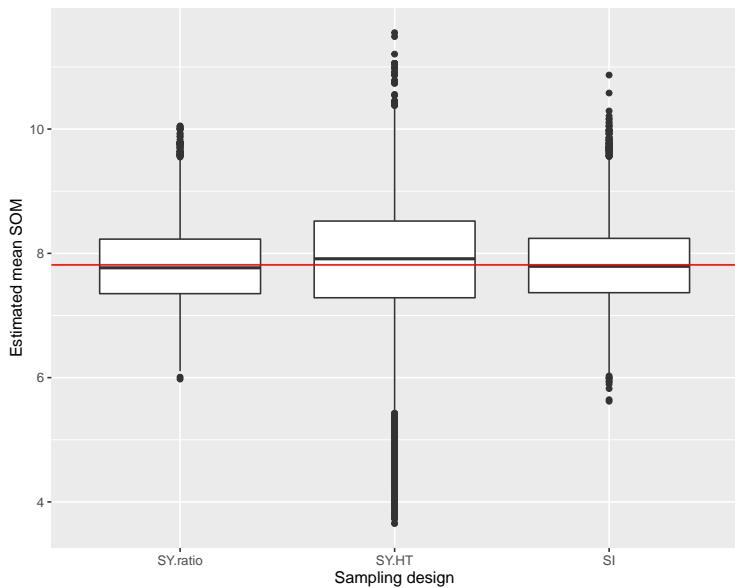


Figure 5.5: Sampling distribution of estimators of the population mean of SOM (g per kg) in Voorst, with systematic random sampling (square grid) and simple random sampling and an (expected) sample size of 40. With systematic random sampling both the π estimator (SY.HT) and the ratio estimator (SY.ratio) are used in estimation.

The boxplots of the estimated means indicate that systematic random sampling in combination with the ratio estimator is more precise than simple random sampling. The variance of the 10,000 ratio estimates equals 0.379, whereas for simple random sampling this variance equals 0.440. Systematic random sampling in combination with the π estimator performs very poor: the variance

equals 1.292. This can be explained by the strong variation in sample size (Figure 5.2), which is not accounted for in the π estimator.

The mean of the 10,000 ratio estimates is 7.81, which is about equal to the population mean 7.814, showing that in this case the design-bias of the ratio estimator is negligibly small indeed.

The average of the 10,000 approximated variances treating the systematic sample as a simple random sample equals 0.443. This is larger than the variance of the ratio estimator (0.379). The stratified simple random sample approximation of the variance is somewhat better: the mean of this variance approximation equals 0.415. The average of the 10,000 variances approximated with Matérn's method equals 0.41. Figure 5.6 shows boxplots of the approximated standard error of the estimated population mean. The horizontal red line is at the standard deviation of the 10,000 ratio estimates of the population mean. Differences between the three approximation methods are small in this case.

The variance of the 10,000 ratio estimates of the population mean with the triangular grid and an expected sample size of 40 equals 0.31. Treating the triangular grid as a simple random sample strongly overestimates the variance: the average approximate variances equals 0.66. Matérn's method cannot be used to approximate the variance with a triangular grid. Function `kmeans_equal_size` can be used to cluster the points of the triangular grid into clusters of equal size.

The approximated variance for this clustering equals 0.39.

? compared various variance approximations for systematic random sampling, among which model-based prediction of the variance, using a semivariogram that is estimated from the systematic sample, see Chapter 13.

Exercises

1. One solution to the problem of variance estimation with systematic random sampling is to select multiple systematic random samples independently from each other. So, for instance, instead of one systematic random sample of with an expected sample size of 40, we may select two systematic random samples with an expected size of 20.
 - Write an **R** script to select two systematic random samples (random square grids) both with an expected size of 20 from `Voorst` (data are in `data/Voorst.RData`).

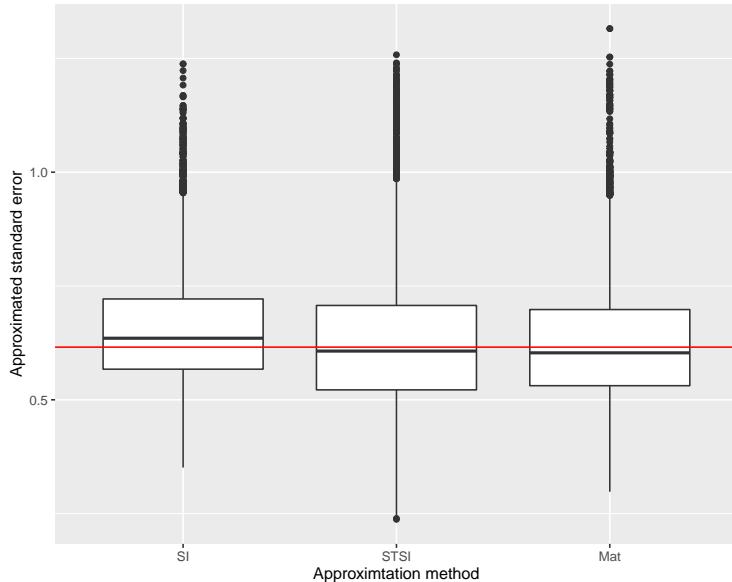


Figure 5.6: Sampling distribution of approximated standard error of the ratio estimator of the population mean of SOM (g per kg) in Voorst, with systematic random sampling (square grid) and an expected sample size of 40. Approximations are obtained by treating the systematic sample as a simple random sample (SI), stratified simple random sample (STSI), and with Matern's method (Mat).

- Use each sample to estimate the population mean, so that you obtain two estimated means. Overlay the points of each sample with `grdVoorst`, using function `over` and extract the z -values.
- Use the two estimated means to estimate the sampling variance of the estimator of the mean for systematic random sampling *with an expected sample size of 20*.
- Use the two estimated means to compute a single, final estimate of the population mean, as estimated from *two systematic random samples, each with an expected sample size of 20*.

- Estimate the sampling variance of the final estimate of the population mean.
2. Do you like this solution? What about the variance of the estimator of the mean, obtained by selecting two systematic random samples of half the expected size, as compared with the variance of the estimator of the mean obtained with a single systematic random sample? Hint: plot the two random square grids. What do you think of the spatial coverage of the two samples?.

Chapter 6

Cluster random sampling

With stratified random sampling with geographical strata and systematic random sampling the sampling units are well spread throughout the study area. In general this leads to an increase of the precision of the estimated mean (total). This is because many spatial populations show spatial structure, so that the values of the study variable at two close points are more similar than those at two distant points. With large study areas the price to be paid for this is long travel times, so that fewer sampling units can be observed in a given survey time. In this situation it can be more efficient to select *spatial clusters* of population units. In cluster random sampling, once a cluster is selected, *all* units in this cluster are observed. For this reason this design is also referred to as *single-stage* cluster random sampling. The clusters are not subsampled, as in two-stage cluster random sampling (see Chapter 7).

In spatial sampling a popular cluster shape is a transect. This is because the individual sampling units of a transect can easily be located in the field, which was in particular an advantage in the pre-GPS era.

The implementation of cluster random sampling is not straightforward. I have seen many examples in the literature of an improper implementation of this sampling design. A proper selection technique is as follows (?). In the first step a starting point (unit) is selected, for instance by simple random sampling. Then the remaining units of the cluster to which the starting point belongs are identified by making use of the definition of the cluster. For instance, with

clusters defined as E-W oriented transects, with a cluster spacing of 100 m, all points east and west of the starting point at a distance of 100 m, 200 m etc. that fall inside the study area are selected. These two steps are repeated until the required number of *clusters* (not the number of points) is selected.

A requirement of a valid selection method is that the same cluster is selected, regardless of which of its units is used as a starting point. In the example above this is the case: regardless of which of the points on the transect is selected first, the final set of points selected is the same because, as stated above, all points E and W of the starting point are selected.

An example of an improper implementation of this sampling design is the following. A cluster is defined as an E-W oriented transect of four points with a mutual spacing of 100 m. A cluster is selected by randomly selecting a starting point. The remaining three points of the cluster are selected E of this starting point. Points outside the study area are ignored. With this selection method the set of selected points is *not* independent of the starting point, and therefore is invalid.

Note that the size of the clusters, i.e. the number of elementary units (points) of a cluster, need not be constant. With the proper selection method described above the selection probability of a cluster is proportional to its size. With irregularly shaped study areas the size of the cluster can vary strongly. The size of the clusters can be controlled by subdividing the study area into blocks, for instance stripes perpendicular to the direction of the transects, or square blocks in case the clusters are grids. In this case, the remaining units are identified by extending the transect or grid until the boundary of the block. With irregularly shaped areas blocking will not eliminate entirely the variation in cluster sizes.

Cluster random sampling is illustrated with the selection of E-W oriented transects in Voorst. In order to delimit the length of the transects the study area is split into six 1 km × 1 km zones. In this case the zones have an equal size, but this is not needed. Note that these zones do not serve as strata. When used as strata, from each zone one or more clusters would be selected, see Section 6.4.

In the code chunk below the zones are constructed by first computing a vector with the s1-coordinates of the boundaries of the zones. Half the size of the cells of the discretisation grid (12.5 m) is added to `s1bnd` so that the boundaries are halfway discretisation nodes. The function `findInterval` of the **base** package (?) is then used to determine for all discretisation nodes in which zone they fall.

```
library(sp)
load("data/Voorst.RData")
gridded(grdVoorst) <- ~s1+s2
gridtop <- as(getGridTopology(grdVoorst), "data.frame")
cellsize <- gridtop$cellsize[1]
grdVoorst <- as(grdVoorst, "data.frame")
w <- 1000 #width of zones
s1bnd <- seq(from=min(grdVoorst$s1)+w, to=min(grdVoorst$s1)+5*w,
              by=w)+cellsize/2
grdVoorst$zone <- findInterval(grdVoorst$s1, s1bnd)
```

As a first step in the **R** code below all clusters in the finite representation of the population are constructed. This is done by computing the interaction of three factors:

1. the modulus of the s1-coordinate and the spacing of units within a transect (cluster) (computed with the operator %%).
2. the s2-coordinates of the grid cells.
3. the zones of the grid cells.

Factor 1 has four levels, as the modulus of the s1-coordinates and a spacing of 100 has four possible values: 0, 25, 50 and 75. The cluster-id is added to the sampling frame. Each point belongs exactly to one cluster.

```
#compute local coordinates
s1local <- grdVoorst$s1-min(grdVoorst$s1)
s2local <- grdVoorst$s2-min(grdVoorst$s2)
spacing <- 100
mods1 <- s1local%%spacing
#construct clusters (E-W oriented transects within zones)
grdVoorst$cluster <- interaction(
  as.factor(mods1),
  as.factor(s2local),
  as.factor(grdVoorst$zone)) %>% as.character()
M_cl <- tapply(
  grdVoorst$z, INDEX=grdVoorst$cluster, FUN=length)
```

```
grdVoorst$unit <- 1:nrow(grdVoorst)
```

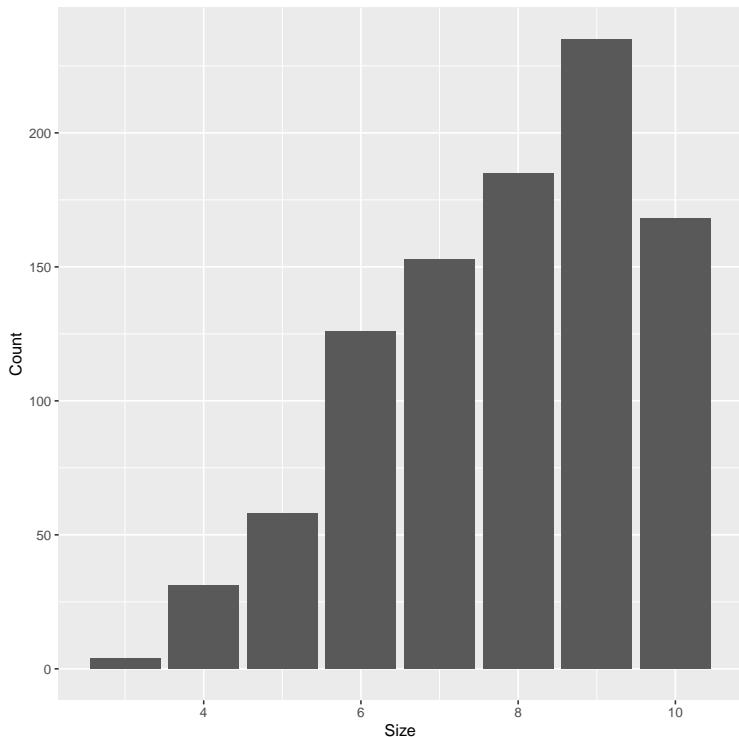


Figure 6.1: Bar plot of size (number of grid cells) of clusters: E-W oriented transects within zones, with an inter-point spacing of 100 m.

In total there are 960 clusters in the population. Figure 6.1 shows the distribution of the size (number of grid cells) of the clusters.

Clusters are selected with probabilities proportional to their size and with replacement (ppswr). So the sizes of all clusters must be known, which explains that all clusters must be enumerated. Selection of clusters by ppswr can be done by simple random sampling with replacement of elementary units (grid cells), and identifying the clusters to which these units belong. Finally, all units of the selected clusters are included in the sample. In the code chunk below a function

is defined for selecting clusters by ppswr. Note the variable `cldraw`, that has value 1 for all units selected in the first draw, value 2 for all units selected in the second draw, etc. This variable is needed in estimating the population mean, see hereafter.

```
cl_ppswr <- function(sframe, n) {
  units <- sample.int(nrow(sframe), size=n, replace=TRUE)
  units_cl <- sframe$cluster[units]
  mysamples <- NULL
  for (i in 1:length(units_cl)) {
    mysample <- sframe[sframe$cluster %in% units_cl[i],]
    mysample$start <- 0
    mysample$start[mysample$unit %in% units[i]] <- 1
    mysample$cldraw <- rep(i, nrow(mysample))
    mysamples <- rbind(mysamples, mysample)
  }
  mysamples
}
```

The function `cl_ppswr` is now used to select six times a cluster by ppswr.

```
n <- 6
set.seed(314)
mysample <- cl_ppswr(sframe=grdVoorst, n=n)
```

As our population actually is infinite, the selected sampling points (nodes of the discretisation grid) are jittered to a random point within the selected grid cells. Note that the same noise is added to all points of a cluster.

```
for (i in 1:n) {
  units <- which(mysample$cldraw==i)
  mysample$s1[units] <- mysample$s1[units] +
    runif(1,min=-12.5,max=12.5)
  mysample$s2[units] <- mysample$s2[units] +
    runif(1,min=-12.5,max=12.5)
}
```

Figure 6.2 shows the selected sample. Note that in this case the second west-most zone has two transects (clusters) whereas three zones have none, showing that the zones are not used as strata. The total number of selected points equals 50. Similar to systematic random sampling, with cluster random sampling the total sample size is random, so that we do not have perfect control of the total sample size. This is because in this case the sizes (number of points) of the clusters is not constant but varies.

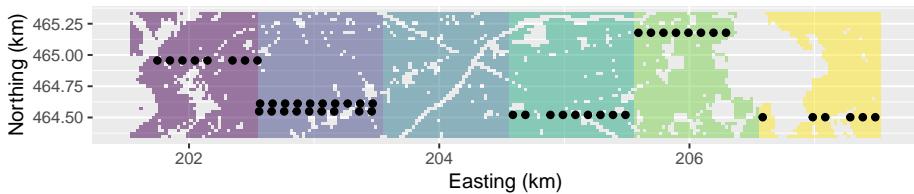


Figure 6.2: Cluster random sample from Voorst. Six times a cluster (transect) is selected with probabilities proportional to their size and with replacement

The output data frame of function `c1` has a column named `start`. This is an indicator with value 1 if this point of the cluster is selected first, and 0 otherwise. When in the field it appears that the first selected point of a cluster does not belong to the target population, all other points of that cluster are also discarded. This is to keep the selection probabilities of the clusters exactly proportional to their size. The column `cldraw` is needed in estimation because clusters are selected with replacement. In case a cluster is selected more than once, multiple means of that cluster are used in estimation, see next section.

6.1 Estimation of population parameters

With pps with replacement (ppswr) sampling of clusters, the population total can be estimated by the pwr estimator:

$$\hat{t}(z) = \frac{1}{n} \sum_{j \in \mathcal{S}} \frac{t_j(z)}{p_j} , \quad (6.1)$$

with n the number of cluster draws, p_j the draw-by-draw selection probability of cluster j and $t_j(z)$ the total of cluster j :

$$t_j(z) = \sum_{k=1}^{M_j} z_{kj}, \quad (6.2)$$

with M_j the size (number of units) of cluster j and z_{kj} the study variable value of unit k in cluster j .

The draw-by-draw selection probability of a cluster equals

$$\bar{p}_j = \frac{M_j}{M}; \quad (6.3)$$

with M the total number of population units (for Voorst $M = 7528$). Inserting this in Equation (6.1) yields

$$\hat{t}(z) = \frac{M}{n} \sum_{j \in S} \frac{t_j(z)}{M_j} = \frac{M}{n} \sum_{j \in S} \bar{z}_j, \quad (6.4)$$

with \bar{z}_j the mean of cluster j . Note that if a cluster is selected more than once, multiple means of that cluster are in the estimator.

Dividing this estimator by the total number of population units M , gives the estimator of the population mean:

$$\hat{\bar{z}} = \frac{1}{n} \sum_{j \in S} \bar{z}_j. \quad (6.5)$$

Note the two bars in $\hat{\bar{z}}$, indicating that the observations are averaged twice.

The sampling variance of the estimator of the mean with cluster random sampling (clusters selected with probabilities proportional to size with replacement, ppswr) is equal to (Equation 9A.6 in ?)

$$V(\hat{\bar{z}}) = \frac{1}{d} \sum_{j=1}^N \frac{M_j}{M} (\bar{z}_j - \bar{z})^2, \quad (6.6)$$

with N total number of clusters (for Voorst, $N = 960$), \bar{z}_j the mean of cluster j , and \bar{z} the population mean. Note that M_j/M is the selection probability of cluster j .

This sampling variance can be estimated by (Equation 9A.22 in ?)

$$\widehat{V}\left(\widehat{\bar{z}}\right) = \frac{\widehat{S}^2(\bar{z})}{n}, \quad (6.7)$$

where $\widehat{S}^2(\bar{z})$ is the estimated variance of cluster means (between cluster variance):

$$\widehat{S}^2(\bar{z}) = \frac{1}{n-1} \sum_{j \in S} (\bar{z}_j - \widehat{\bar{z}})^2. \quad (6.8)$$

In R the population mean and the sampling variance of the estimator of the population means can be estimated as follows.

```
mz_cl <- tapply(
  mysample$z, INDEX=mysample$cldraw, FUN=mean)
mz <- mean(mz_cl)
se_mz <- sqrt(var(mz_cl)/n)
```

The estimated mean equals 8.549 and the estimated standard error equals 1.473. Note that the size of the clusters (number of units) does not appear in these formulas. This simplicity is due to the fact that the clusters are selected with probabilities proportional to size. The effect of the cluster size on the variance is implicitly accounted for. To understand this, consider that larger clusters result in smaller variance among their means.

The same estimates are obtained with functions `svydesign` and `svymean` of package `survey` (?). Argument `weights` specifies the weights of the sampled clusters equal to $M/(M_j d)$ (Equation (6.4)).

```
library(survey)
M <- nrow(grdVoorst)
mysample$weights <- M/(M_cl[mysample$cluster]*n)
design_cluster <- svydesign(
  id=~cldraw, weights=~weights, data=mysample)
svymean(~z, design_cluster, deff="replace")
```

```
mean      SE     DEff
z 8.5489 1.4729 3.8145
```

The design effect as estimated from the selected cluster sample is considerably larger than 1. About 3.8 times more sampling points are needed with cluster random sampling compared to simple random sampling to estimate the population mean with the same precision.

A confidence interval estimate of the population mean can be computed with method `confint`. The number of degrees of freedom equals the number of cluster draws minus 1.

```
confint(svymean(
  ~z, design_cluster, df=degf(design_cluster), level=0.95))

  2.5 % 97.5 %
z 5.662115 11.43574
```

Figure 6.3 shows the sampling distributions of the pwr estimator of the mean with cluster random sampling and of the π estimator with simple random sampling, obtained by repeating the random sampling with each design and estimation 10,000 times. The size of the simple random samples is equal to the expected sample size of the cluster random sampling design (rounded to nearest integer).

The variance of the 10,000 estimated population means with cluster random sampling equals 0.822. This is considerably larger than with simple random sampling: 0.364. The large variance is caused by the strong spatial clustering of points. This may save travel time in large study areas, but in Voorst the saved travel time will be very limited, and therefore cluster random sampling in Voorst is not a good idea. The average of the estimated variances with cluster random sampling equals 0.811. The difference with the variance of the 10,000 estimated means is small because the estimator of the variance, Equation (6.7), is unbiased. Figure 6.4 shows the sampling distribution of the sample size. The expected sample size can be computed as follows:

```
p <- M_cl/sum(M_cl)
print(m_n <- n*sum(p*M_cl))

[1] 49.16844
```

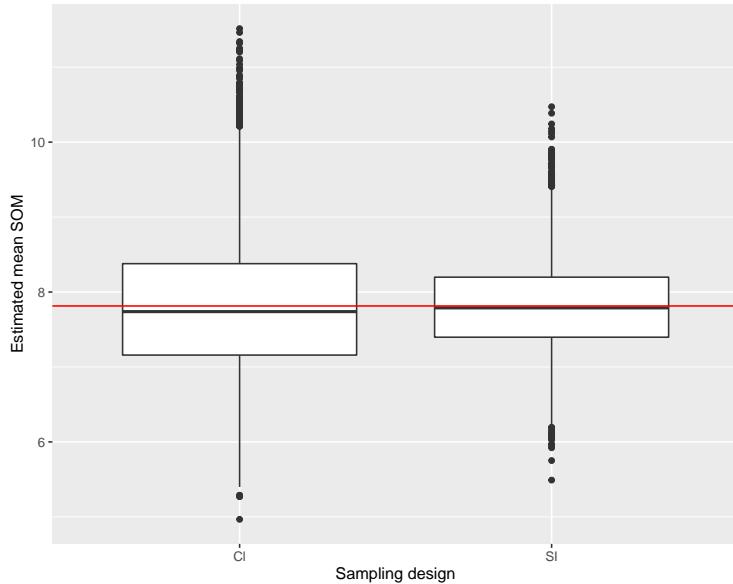


Figure 6.3: Sampling distribution of the pwr estimator of the mean of SOM (g/kg) in Voorst with cluster random sampling, and of the π estimator with simple random sampling, both designs with an (expected) sample size of 49 units.

So the unequal draw-by-draw selection probabilities of the clusters are accounted for in computing the expected sample size.

Exercises

1. Write an **R** script to compute the true sampling variance of the estimator of the population mean of SOM in Voorst, for cluster random sampling, clusters selected with probabilities proportional to their size and with replacement, $n = 6$, see Equation (6.6). Compare the sampling variance for cluster random sampling with the sampling variance for simple random sampling with a sample size equal to the expected sample size of cluster random sampling.
2. As an alternative we may select three times a transect, using three 2 km

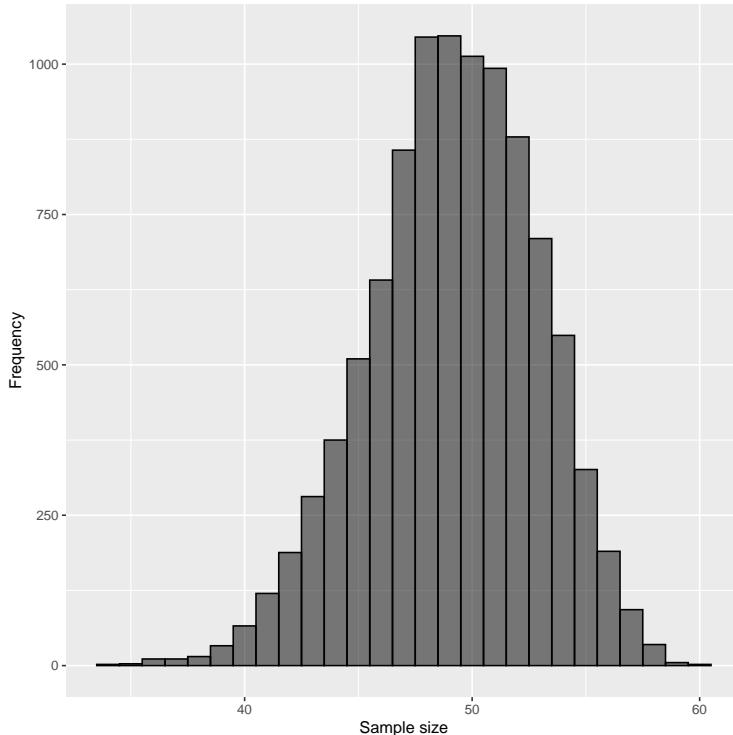


Figure 6.4: Sampling distribution of sample size with cluster random sampling.

$\times 1$ km zones obtained by joining two neighbouring 1 km $\times 1$ km zones of Figure 6.2. Do you expect that the sampling variance of the estimator of the mean is equal, larger or smaller than that of the sampling design with six transects of “half the length”?

6.2 Clusters selected with probabilities proportional to size, without replacement

In the previous section the clusters were selected with probabilities proportional to size and with replacement (ppswr). The advantage of with replacement sampling is that this keeps the statistical inference simple, more specifically the

estimation of the standard error of the estimated population mean. However, in sampling from finite populations, cluster sampling with replacement is less efficient than cluster sampling without replacement sampling, especially with large sampling fractions of clusters, i.e. if $1 - n/N$ is small. If a cluster is selected more than once, there is less information about the population mean in this sample than in a sample with all clusters different. Selection of clusters with probabilities proportional to size without replacement (ppswor) is not straightforward. The problem is the computation of the inclusion probabilities of the clusters. After we have selected a first cluster, we must adapt the sum of the sizes of the $N - 1$ remaining clusters, and recompute the selection probabilities of the remaining clusters in the second draw, etc. Section 6.4 of ? nicely describes how the inclusion probabilities of the N clusters in a cluster random sample of size two, selected by ppswor can be computed. Many algorithms have been developed for ppswor sampling, see ? for an overview, and many of them are implemented in package **sampling** (?). In the next code chunk function **UPpivot** is used to select a cluster random sample with ppswor. For an explanation of this algorithm, see Section 8.2.2.

```
library(sampling)
n <- 6
pi <- n*M_cl/M
set.seed(314)
eps <- 1e-6
sampleind <- UPpivot(pik=pi, eps=eps)
clusters <- sort(unique(grdVoorst$cluster))
sampledclusters <- clusters[sampleind==1]
mysample <- grdVoorst[grdVoorst$cluster %in% sampledclusters,]
```

The population mean can be estimated with function **svymean** of package **survey** (?). Estimation of the sampling variance in pps sampling of clusters without replacement is difficult¹. A simple solution is to treat the cluster sample as a ppswr sample, and to estimate the variance with Equation (6.7). With small sampling fractions this variance approximation is fine: the overestimation of the variance is negligible. For larger sampling fractions various alternative variance approximations are developed, see ? for details. One of the methods is Brewer's method, which is implemented in function **svydesign**.

¹The problem is the computation of the joint inclusion probabilities of pairs of points.

```

mysample$pi <- n*M_cl[mysample$cluster]/M
design_clppswor <- svydesign(
  id=~cluster, data=mysample, pps="brewer", fpc=~pi)
svymean(~z,design_clppswor)

  mean      SE
z 9.6777 1.2931

```

Another variance estimator implemented in function `svydesign` is the Hartley-Rao estimator. The two estimated standard errors are nearly equal.

```

p2sum<-sum((n*M_cl[mysample$cluster]/M)^2)/n
design_hr <- svydesign(
  id=~cluster, data=mysample, pps=HR(p2sum), fpc=~pi)
svymean(~z, design_hr)

  mean      SE
z 9.6777 1.2921

```

6.3 Simple random sampling of clusters

Suppose the clusters have unequal size, but we do not know the size of the clusters, so that we cannot select the clusters with probabilities proportional to their size, or for some other reason we selected the clusters by simple random sampling without replacement. The inclusion probability of a cluster equals n/N with n the number of selected clusters and N the total number of clusters in the population. This yields the following π estimator of the population total:

$$\hat{t}(z) = \frac{N}{n} \sum_{j \in \mathcal{S}} t_j(z) , \quad (6.9)$$

The population mean can be estimated by dividing this estimator of the population total by the total number of units in the population M :

$$\hat{\bar{z}}_\pi(z) = \frac{\hat{t}(z)}{M} . \quad (6.10)$$

Alternatively, we may estimate the population mean by dividing the estimate of the population total by the *estimated* population size:

$$\widehat{M} = \sum_{j \in \mathcal{S}} \frac{M_j}{\pi_j} = \frac{N}{n} \sum_{j \in \mathcal{S}} M_j . \quad (6.11)$$

This leads to the ratio estimator of the population mean:

$$\hat{\bar{z}}_{\text{ratio}}(z) = \frac{\hat{t}(z)}{\widehat{M}} . \quad (6.12)$$

The π estimator and ratio estimator are equal when the clusters are selected with probabilities proportional to size. This is because the estimated population size is equal to the true population size.

```
print(M_HT <- sum(1/mysample$pi))

[1] 7528
```

However, when clusters of different size are selected with equal probabilities, the two estimators are different. This is shown below. Six clusters are selected by simple random sampling without replacement.

```
set.seed(314)
clusters <- sort(unique(grdVoorst$cluster))
units_cl <- sample.int(length(clusters), size=n, replace=FALSE)
sampledclusters <- clusters[units_cl]
mysample <- grdVoorst[grdVoorst$cluster %in% sampledclusters,]
```

The π estimator and ratio estimator of the population mean are computed for the selected sample.

```
N <- length(clusters)
mysample$pi <- n/N
tz_HT <- sum(mysample$z/mysample$pi)
mz_HT <- tz_HT/M
```

```
M_HT <- sum(1/mysample$pi)
mz_ratio <- tz_HT/M_HT
```

The π estimate equals 6.449, and the ratio estimate equals 6.596. The π estimator of the population mean can also be computed by first computing totals of clusters, see Equation (6.9).

```
tz_cluster <- tapply(mysample$z, INDEX=mysample$cluster, FUN=sum)
pi_cluster <- n/N
tz_HT <- sum(tz_cluster/pi_cluster)
print(mz_HT <- tz_HT/M)
```

```
[1] 6.448566
```

The variance of the π estimator of the population mean can be estimated by first estimating the variance of the estimator of the total, and dividing this variance by the squared number of population units:

$$\begin{aligned}\widehat{V}(\hat{t}(z)) &= N^2 \left(1 - \frac{n}{N}\right) \frac{\widehat{S^2}(t(z))}{n} \\ \widehat{V}(\bar{\bar{z}}) &= \frac{1}{M^2} \widehat{V}(\hat{t}(z)) .\end{aligned}\quad (6.13)$$

```
fpc <- 1-n/N
v_tz <- N^2*fpc*var(tz_cluster)/n
se_mz_HT <- sqrt(v_tz/M^2)
```

The estimated standard error equals 0.82. To compute the variance of the ratio estimator of the population mean we first compute residuals of cluster totals:

$$e_j = t_j(z) - \hat{b}M_j , \quad (6.14)$$

with \hat{b} the ratio of the estimated population mean of the cluster totals to the estimated population mean of the cluster sizes:

$$\hat{b} = \frac{\frac{1}{n} \sum_{j \in \mathcal{S}} t_j}{\frac{1}{n} \sum_{j \in \mathcal{S}} M_j}. \quad (6.15)$$

The variance of the ratio estimator of the population mean can be estimated by

$$\hat{V}(\hat{\bar{z}}_{\text{ratio}}) = \left(1 - \frac{n}{N}\right) \frac{1}{(\frac{1}{n} \sum_{j \in \mathcal{S}} M_j)^2} \frac{\widehat{S^2}_e}{n}, \quad (6.16)$$

with $\widehat{S^2}_e$ the estimated variance of the residuals.

```
m_M_cl <- mean(M_cl[unique(mysample$cluster)])
b <- mean(tz_cluster)/m_M_cl
e_cl <- tz_cluster -
      b*M_cl[sort(unique(mysample$cluster))]
S2e <- var(e_cl)
print(se_mz_ratio <- sqrt(fpc*1/m_M_cl^2*S2e/n))

[1] 0.9801064
```

The ratio estimate can also be computed with function `svymean` of package `survey`, which also provides an estimate of the standard error of the estimated mean.

```
design_SIC <- svydesign(
  id=~cluster, probs=~pi, fpc=~pi, data=mysample)
svymean(~z, design_SIC)

  mean      SE
z 6.5958  0.9801
```

6.4 Stratified cluster random sampling

The basic sampling designs stratified random sampling (Chapter 4) and cluster random sampling can be combined into stratified cluster random sampling. So instead of selecting simple random samples from the strata, within each stratum

clusters are randomly selected. Figure 6.5 shows a stratified cluster random sample from Voorst. The strata consist of three $2 \text{ km} \times 1 \text{ km}$ zones, obtained by joining two neighbouring $1 \text{ km} \times 1 \text{ km}$ zones (Figure 6.2). The clusters are the same as before, i.e. E-W oriented transects within $1 \text{ km} \times 1 \text{ km}$ zones, with a inter-point spacing of 100 m. Within each stratum two times a cluster is selected by ppswr. The stratification avoids the clustering of the selected transects in one part of the study area. Compared to (unstratified) cluster random sampling, the geographical spreading of the clusters is improved, which may lead to an increase of the precision of the estimated population mean. In Figure 6.5 in the most western stratum the two selected transects are in the same $1 \text{ km} \times 1 \text{ km}$ zone. The alternative would be to use the six zones as strata, leading to an improved spreading of the clusters, but there is also a downside with this design, see Exercise 3.

```
grdVoorst$zonestratum <- as.factor(grdVoorst$zone)
levels(grdVoorst$zonestratum) <- rep(c("a", "b", "c"), each=2)
n_h <- c(2, 2, 2)
set.seed(324)
stratumlabels <- unique(grdVoorst$zonestratum)
mysample <- NULL
for (i in 1:3) {
  grd_h <- grdVoorst[grdVoorst$zonestratum==stratumlabels[i],]
  mysample_h <- cl_ppswr(sframe=grd_h, n=n_h[i])
  mysample <- rbind(mysample, mysample_h)
}

```

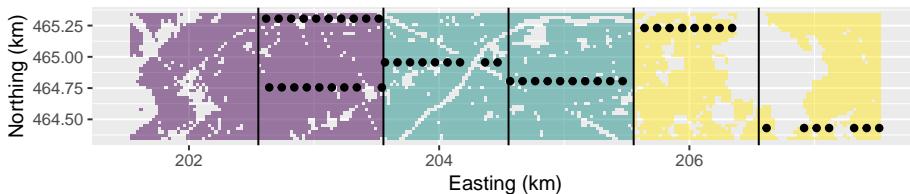


Figure 6.5: Stratified cluster random sample from Voorst, with three strata.

The population mean is estimated by first estimating the stratum means using Equation (6.5), followed by computing the weighted average of the estimated stratum means using Equation (4.3). The variance of the estimator of the

population mean is estimated in the same way, by first estimating the variance of the estimator of the stratum means using Equation (6.7), followed by computing the weighted average of the estimated variances of the estimated stratum means (Equation (4.4)).

```
mz_h <- v_mz_h <- numeric(length=3)
for (i in 1:3) {
  units <- which(mysample$zonestratum==letters[i])
  mysample_h <- mysample[units,]
  mz_cl <- tapply(mysample_h$z,
    INDEX=mysample_h$cldraw, FUN=mean)
  mz_h[i] <- mean(mz_cl)
  v_mz_h[i] <- var(mz_cl)/n_h[i]
}
M_h <- tapply(
  grdVoorst$z, INDEX=grdVoorst$zonestratum, FUN=length)
w_h <- M_h/M
mz <- sum(w_h*mz_h)
se_mz <- sqrt(sum(w_h^2*v_mz_h))
```

The estimated mean equals 7.708, and the estimated standard error equals 0.66. The same estimates are obtained with function `svymean`. Weights for the clusters are computed as before, but now at the level of the strata. Note the argument `nest=TRUE`, which means that the clusters are nested within the strata.

```
mysample$weights <- M_h[mysample$zonestratum]/
  (M_cl[mysample$cluster]*n_h[mysample$zonestratum])
design_strcluster <- svydesign(
  id=~cldraw, strata=~zonestratum, weights=~weights,
  data=mysample, nest=TRUE)
svymean(~z,design_strcluster)
```

mean	SE
z 7.7075	0.6597

Exercises

3. Why is it attractive in stratified random cluster sampling to select at least two clusters per stratum?

Chapter 7

Two-stage cluster random sampling

As opposed to cluster random sampling in which all population units of cluster are observed (Chapter 6), in two-stage cluster random sampling not all sampling units of the selected clusters are observed, but only some of them. In two-stage cluster random sampling the clusters will generally be contiguous groups of units, for instance all points in a map polygon (the polygons on the map are the clusters), whereas in single-stage cluster random sampling the clusters generally are non-contiguous. The sampling units to be observed are selected by random subsampling of the randomly selected clusters. In two-stage cluster sampling the clusters are commonly referred to as primary sampling units (psu's) or shortly primary units (pu's), and the units selected in the second stage as the secondary sampling units (ssu's) or secondary units (su's).

As with cluster random sampling, two-stage cluster random sampling may lead to a strong spatial clustering of the selected sampling units (ssu's) in the study area. This may save considerable time for fieldwork, and more population units can be observed for the same budget. However, due to the spatial clustering the estimates will generally be less precise compared to samples of the same size selected by a design that leads to a much better spreading of the sampling units throughout the study area, such as systematic random sampling.

In two-stage cluster random sampling in principle any type of sampling design

can be used at the two stages, leading to numerous combinations. An example is (SI,SI), in which both psu's and ssu's are selected by simple random sampling.

Commonly the psu's have unequal size, i.e. the number of ssu's (finite population) or the area (infinite population) are not equal for all psu's. Think for instance of the agricultural fields, forest stands, lakes, river sections etc., in an area. If the psu's are of unequal size, then psu's can best be selected with probabilities proportional to their size (pps). Recall that in (one-stage) cluster random sampling I also recommended to select the clusters with probabilities proportional to their size, see Chapter 6. If the total of the study variable of a psu is proportional to its size, then pps sampling leads to more precise estimates compared to simple random sampling of psu's. Also, with pps sampling of psu's the estimation of means or totals and of their sampling variances is much simpler compared to selection with equal probabilities. Implementation of selection with probabilities proportional to size is easiest when units are replaced (pps with replacement, ppswr). This implies that a psu might be selected more than once, especially if the total number of psu's in the population is small compared to the number of psu draws (large sampling fraction in first stage).

Using a list as a sampling frame, the following algorithm can be used to select n times a psu by ppswr from a total of N psu's in the population:

1. Select randomly one ssu from the list with $M = \sum_{j=1}^N M_j$ ssu's (M_j is number of ssu's of psu j), and determine the psu of the selected ssu.
2. Repeat step 1 until n selections have been made.

In the first stage a ssu is selected in order to select a psu. This may seem unnecessary complicated. The reason for this is that this procedure automatically adjusts for the size of the psu's (number of ssu's within a psu), i.e. a psu is selected with probability proportional to its size. In the second stage, a *pre-determined* number of secondary sampling units, m_j , is selected every time psu j is selected. Predetermined means that it is not allowed to decide on the secondary sample sizes (number of ssu's) per selected psu after the selection of the psu's. Note that the ssu selected in the first step of the two algorithms primarily serve to identify the psu, but these ssu's can also be used as selected ssu's.

The selection of a two-stage cluster random sample is illustrated again with Voorst. Twenty-four blocks of $0.5 \text{ km} \times 0.5 \text{ km}$ are constructed that serve as psu's. Note that due to built-up areas, roads etc., these psu's have unequal size,

i.e. the number of secondary units (pixels) within the psu's varies among the psu's.

```
load("data/Voorst.RData")
w <- 500 #width of psu's
s1bnd <- seq(from=min(grdVoorst$s1)+w, to=min(grdVoorst$s1)+(11*w), by=w)+12.5
s1f <- findInterval(grdVoorst$s1,s1bnd)
s2bnd <- min(grdVoorst$s2)+w+12.5
s2f <- findInterval(grdVoorst$s2, s2bnd)
grdVoorst$psu <- as.character(interaction(s1f, s2f))
```

To select a two-stage cluster random sample a function is defined.

```
twostage <- function(sframe, psu, n, m) {
  units <- sample.int(nrow(sframe), size=n, replace=TRUE)
  mysusample <- sframe[units,psu]
  ssunits <- NULL
  for (psunit in mysusample) {
    ssunit <- sample(
      x = which(sframe[,psu]==psunit), size=m, replace=TRUE)
    ssunits <- c(ssunits, ssunit)
  }
  psudraw <- rep(c(1:n), each=m)
  mysample <- data.frame(ssunits,sframe[ssunits,], psudraw)
  mysample
}
```

Note that both the primary and secondary sampling units are selected with replacement. The secondary units are selected by simple random sampling with replacement because the actual population is infinite. The infinite population is discretised by a finite number of grid nodes (centers of grid cells). If a grid node is selected, one point is selected fully randomly from the associated grid cell. This is done with function `jitter`. In every grid cell there is an infinite number of points, so we must select the grid cells with replacement. If a grid node is selected more than once, more than one point is selected from that grid cell. The column `psudraw` in the output data frame of function `twostage` is needed in estimation because psu's are selected with replacement. In case a psu is selected more than once, multiple estimates of the mean of that psu are used

in estimation, see next section.

The function `twostage` is used to select four times a psu ($n = 4$), with probabilities proportional to size and with replacement (ppswr). The second stage sample size m_i equals 10 for all psu's. These secondary sampling units (ssu's) are selected by simple random sampling.

```
n <- 4
m <- 10
set.seed(314)
mysample <- twostage(sframe=grdVoorst, psu="psu", n=n, m=m)
cellsize <- 25
mysample$s1 <- jitter(mysample$s1, amount=cellsize/2)
mysample$s2 <- jitter(mysample$s2, amount=cellsize/2)
```

Figure 7.1 shows the selected sample.

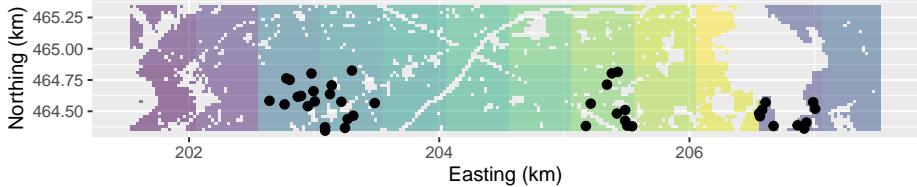


Figure 7.1: Two-stage cluster random sample from Voorst. Four times a psu is selected by ppswr. Each time a psu is selected, ten points are selected from that psu.

7.1 Estimation of population parameters

The population total can be estimated by substituting the estimated cluster (primary unit) totals in Equation (6.4). This yields the following estimator for the population total:

$$\hat{t}(z) = \frac{M}{n} \sum_{j \in \mathcal{S}} \frac{\hat{t}_j(z)}{M_j} = \frac{M}{n} \sum_{j \in \mathcal{S}} \hat{\bar{z}}_j , \quad (7.1)$$

where n is the number of psu selections and M_j is total number of ssu's in psu j . This shows that the mean of cluster j , \bar{z}_j , is replaced by the estimated mean of psu j , $\hat{\bar{z}}_j$. Dividing this estimator by the total number of population units M gives the pwr estimator of the population mean:

$$\hat{\bar{z}} = \frac{1}{n} \sum_{j \in \mathcal{S}} \hat{\bar{z}}_j , \quad (7.2)$$

with $\hat{\bar{z}}_j$ the estimated mean of the psu j . With simple random sampling of ssu's this mean can be estimated by the sample mean of this psu. Note the two bars $\hat{\bar{z}}$, indicating that the population mean is estimated as the mean of estimated primary unit means. When m_i is equal for all psu's the sampling design is self-weighting, i.e. the average of z over all selected secondary units is an unbiased estimator of the population mean.

The sampling variance of the estimator of the mean with two-stage cluster random sampling (primary units selected with probabilities proportional to size with replacement, secondary units by simple random sampling (with replacement in case of finite populations) and $m_j = m, j = 1, \dots, N$) is equal to (?, Equation 11.33)¹

$$V(\hat{\bar{z}}) = \frac{S_b^2}{n} + \frac{S_w^2}{n m} , \quad (7.3)$$

with

$$S_b^2 = \sum_{j=1}^N p_j (\bar{z}_j - \bar{z})^2 , \quad (7.4)$$

and

$$S_w^2 = \sum_{j=1}^N p_j S_j^2 \quad (7.5)$$

¹The equation in Cochran (1977) is the variance estimator for the population total. In Exercise 5 you are asked to derive the estimator of the variance of the estimator of the population mean from the estimator of the variance of the estimator of the population total.

with N the total number of psu's in the population, $p_j = M_j/M$ the draw-by-draw selection probability of psu j , \bar{z}_j the mean of psu j , \bar{z} the population mean of z , and S_j^2 the variance of z within primary unit j :

$$S_j^2 = \frac{1}{M_j} \sum_{k=1}^{M_j} (z_{kj} - \bar{z}_j)^2. \quad (7.6)$$

Note that the first term of Equation (7.3) is equal to the variance of Equation (6.6). This variance component accounts for the variance of the true primary unit means within the population. The second variance component quantifies our additional uncertainty about the population mean, as we do not observe all secondary units of the selected primary units, but only a subset (sample) of these units.

The sampling variance of the estimator of the population mean can simply be estimated by

$$\widehat{V}\left(\widehat{\bar{z}}\right) = \frac{\widehat{S}^2(\widehat{\bar{z}})}{n}, \quad (7.7)$$

with $\widehat{S}^2(\widehat{\bar{z}})$ the estimated variance of the *estimated* primary unit means:

$$\widehat{S}^2(\widehat{\bar{z}}) = \frac{1}{n-1} \sum_{j \in S} (\widehat{\bar{z}}_j - \widehat{\bar{z}})^2, \quad (7.8)$$

with $\widehat{\bar{z}}_j$ the estimated mean of psu j , and $\widehat{\bar{z}}$ the estimated population mean (Equation (7.2)). Note that neither the sizes of the psu's, M_j , nor the secondary sample sizes m_j occur in these formulas. This simplicity is due to the fact that the psu's are selected with replacement and with probabilities proportional to size. The effect of the secondary sample sizes on the variance is implicitly accounted for. To understand this, note that the larger m_j , the less variable $\widehat{\bar{z}}_j$, and the smaller its contribution to the variance.

Let us assume a linear model for the total costs: $C = c_1 d + c_2 n m$, with c_1 the cost per primary unit and c_2 the cost per secondary unit. We want to minimise the total costs, under the constraint that the variance of the estimator of the

population mean may not exceed V_{\max} . The total costs can then be minimised by selecting (?)

$$n = \frac{1}{V_{\max}} \left(S_w S_b \sqrt{\frac{c_2}{c_1}} + S_b^2 \right) \quad (7.9)$$

primary units, and

$$m = \frac{S_w}{S_b} \sqrt{\frac{c_1}{c_2}} \quad (7.10)$$

secondary units per primary unit.

Conversely, given a budget C_{\max} , the optimal number of primary unit selections can be computed with (?)

$$n = \frac{C_{\max} S_b}{S_w \sqrt{c_1 c_2 + S_b c_1}} , \quad (7.11)$$

and m as above.

In R the population mean and the sampling variance of the estimator of the mean can be estimated as follows.

```
mz_psu <- tapply(mysample$z, INDEX=mysample$psudraw, FUN=mean)
mz <- mean(mz_psu)
se_mz <- sqrt(var(mz_psu)/n)
```

The estimated mean equals 7.094 and the estimated standard error equals 1.428. The sampling design is self-weighting, and so the estimated mean is equal to the sample mean.

```
print(mean(mysample$z))
```

```
[1] 7.093781
```

The same estimate is obtained with functions **svydesign** and **svymean** of package **survey** (?). The estimator of the population total can be written as a

weighted sum of the observations with all weights equal to $M/(d \cdot m)$. These weights are assigned to argument `weight`.

```
library(survey)
M <- nrow(grdVoorst)
mysample$weights <- M/(n*m)
design_2stage <- svydesign(
  id=~psudraw+ssunits, weight=~weights, data=mysample)
svymean(~z, design_2stage, def="replace")

  mean      SE    DEff
z 7.0938 1.4281 3.1686
```

Similar to (one-stage) cluster random sampling the estimated design effect is much larger than 1.

A confidence interval estimate of the population mean can be computed with method `confint`. The number of degrees of freedom equals the number of psu draws minus one.

```
confint(svymean(
  ~z, design_2stage, df=degf(design_2stage), level=0.95))

  2.5 %  97.5 %
z 4.294756 9.892806
```

Figure 7.2 shows the sampling distributions of the pwr estimator of the population mean with two-stage cluster random sampling and of the π estimator with simple random sampling from Voorst, obtained by repeating the random sampling with each design and estimation 10,000 times. For simple random sampling the sample size is equal to $n \times m$.

The variance of the 10,000 means with two-stage cluster random sampling equals 1.238. This is considerably larger than with simple random sampling: 0.429. The average of the estimated variances equals 1.218.

Optimal sample sizes for two-stage cluster random sampling (ppswr in first stage, simple random sampling without replacement in second stage) can be computed with function `clus0pt2` of R package **PracTools** (?). This function requires as input various variance measures, which can be computed with

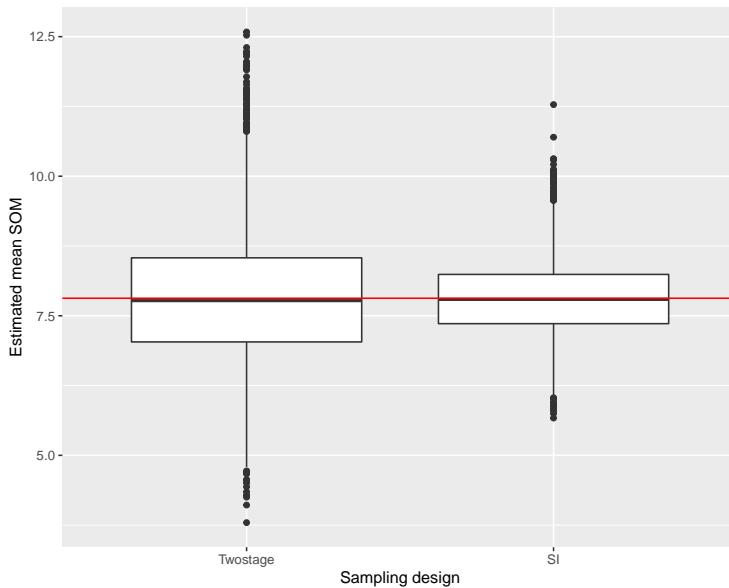


Figure 7.2: Sampling distribution of pwr estimator of mean of SOM (g/kg) in Voorst with two-stage random sampling, and of the π estimator with simple random sampling, bot design with a sample size of 40 units.

function `BW2stagePPS` in case the study variable is known for the whole population, or estimated from a sample with function `BW2stagePPSe`. This is left as an exercise (Exercise 5).

Exercises

1. Write an **R** script to compute the true sampling variance of the estimator of the population mean for $n = 4$ and $m = 10$, see Equation (7.3).
2. Do you expect that the standard error of the estimated population mean with ten psu draws ($n = 10$) and four ssu's per psu draw ($m = 4$) are larger or smaller than with four psu draws ($n = 4$) and ten ssu's per psu draw ($m = 10$)?
3. Compute the optimal sample sizes n and m for a maximum variance of

the estimator of the population mean of 1, $c_1 = 2$ and $c_2 = 1$ monetary unit, see Equations (7.9) and (7.10).

4. Compute the optimal sample sizes n and m for a budget of 100 monetary units, $c_1 = 2$ and $c_2 = 1$ monetary units, see Equations (7.11) and (7.10).
5. Use function `clusOpt2` of **R** package **PracTools** to compute optimal sample sizes given the precision requirement for the estimated population mean of Exercise 3 and given the budget of Exercise 4. First use function `BW2stagePPS` to compute the variance measures needed as input for function `optClus2`. Note that the precision requirement of function `clusOpt2` is the coefficient of variation of the estimated population total, i.e. the standard deviation of the estimated population total divided by the population total. Compute this coefficient of variation from the maximum variance of the estimator of the population mean used in Exercise 3.
6. Derive the variance estimator for the estimated population mean, Equation (7.3), from the variance estimator of the estimated population total (?):

$$V(\hat{t}(z)) = \frac{1}{n} \sum_{j=1}^N p_j \left(\frac{t_j(z)}{p_j} - t(z) \right)^2 + \frac{1}{n} \sum_{j=1}^N \frac{M_j^2(1-f_{2j})S_j^2}{m_j p_j}, \quad (7.12)$$

with $\hat{t}(z)$ and $t(z)$ the estimated and true population total of z , respectively, $t_j(z)$ the total of psu j , and $p_j = M_j/M$. Use $m_j = m, j = 1, \dots, N$, and $f_{2j} = 0$, i.e. sampling from infinite population, or sampling of ssu's within psu's by simple random sampling *with replacement* from finite population.

7.2 Primary sampling units selected with probabilities proportional to size, without replacement

Similar to cluster random sampling, we may prefer to select the primary sampling units without replacement. This leads to less strong spatial clustering of the sampling points, especially with large sampling fractions of primary sampling units. The psu's are selected with function `UPpivotal` of package **sam-**

pling (?), see Section 8.2.2. The second stage sample of secondary sampling units is selected with function **strata** of the same package, using the psu's as strata.

```
library(sampling)
M_psu <- tapply(grdVoorst$z, INDEX=grdVoorst$psu, FUN=length)
n <- 6
pi <- n*M_psu/M
set.seed(314)
sampleind <- UPpivot(pik=pi, eps=1e-6)
psus <- sort(unique(grdVoorst$psu))
sampledpsus <- psus[sampleind==1]
mysample_stage1 <- grdVoorst[grdVoorst$psu %in% sampledpsus,]
units <- sampling::strata(
  mysample_stage1, stratanames="psu",
  size=rep(m,n), method="srswor")
mysample <- getdata(mysample_stage1, units)
mysample$ssunits <- units$ID_unit
mysample$pi <- n*m/M
print(mean_HT <- sum(mysample$z/mysample$pi)/M)

[1] 7.306838
```

The population mean can be estimated with function **svymean** of package **survey** (?). A simple solution is to treat the two-stage cluster random sample as a pps sample with replacement, and to estimate the variance with Equation (7.7). With small sampling fractions of psu's the overestimation of the variance is negligible. As in cluster random sampling without replacement, the variance is approximated with Brewer's method, see ? (option 2).

```
mysample$fpc1 <- n*M_psu[mysample$psu]/M
mysample$fpc2 <- m/M_psu[mysample$psu]
design_2stageppswor <- svydesign(
  id=~psu+ssunits, data=mysample, pps="brewer", fpc=fpc1+fpc2)
svymean(~z,design_2stageppswor)

  mean      SE
z 7.3068 1.3998
```

7.3 Simple random sampling of primary sampling units

Suppose the primary sampling units are for some reason not selected with probabilities proportional to their size, but by simple random sampling without replacement. The inclusion probability of the psu's then equal $\pi_j = n/N, j = 1, \dots, N$, and the population total can be estimated by (compare with Equation (6.9))

$$\hat{t}(z) = \sum_{j=1}^n \frac{\hat{t}_j(z)}{\pi_j} = \frac{N}{n} \sum_{j=1}^n \hat{t}_j(z), \quad (7.13)$$

with $\hat{t}_j(z)$ an estimator of the total of psu j . The population mean can be estimated by dividing this estimator by the population size M .

Alternatively, we may estimate the population mean by dividing the estimate of the population total by the *estimated* population size. The π estimator of the population size for two-stage cluster sampling is equal to that for cluster random sampling, see Equation (6.11). The π estimator and ratio estimator are equal when the psu's are selected with probabilities proportional to size and with replacement, but not so when the psu's of different size are selected with equal probabilities. This is shown below. First a sample is selected by selecting both psu's and ssu's by simple random sampling without replacement.

```
library(sampling)
set.seed(314)
psus <- sort(unique(grdVoorst$psu))
ids_psu <- sample.int(length(psus), size=n, replace=FALSE)
sampledpsus <- psus[ids_psu]
mysample_stage1 <- grdVoorst[grdVoorst$psu %in% sampledpsus,]
units <- sampling::strata(
  mysample_stage1, stratanames="psu",
  size=rep(m,n), method="srswor")
mysample <- getdata(mysample_stage1, units)
mysample$ssunits <- units$ID_unit
```

The population mean is estimated by the π estimator and the ratio estimator.

```

N <- length(unique(grdVoorst$psu))
M_psu <- tapply(grdVoorst$z, INDEX=grdVoorst$psu, FUN=length)
pi_psu <- n/N
pi_ssu <- m/M_psu[mysample$psu]
mysample$pi <- pi_psu*pi_ssu
z_piexpanded <- with(mysample,z/pi)
tz_HT <- sum(z_piexpanded)
mz_HT <- tz_HT/M
M_HT <- sum(1/mysample$pi)
mz_ratio <- tz_HT/M_HT

```

The π estimate equals 7.333 and the ratio estimate equals 7.412. The π estimator of the population mean can also be computed by first estimating totals of psu's, see Equation (7.13).

```

tz_psu <- tapply(mysample$z/pi_ssu, INDEX=mysample$psu, FUN=sum)
tz_HT <- sum(tz_psu/pi_psu)
(mz_HT <- tz_HT/M)

```

```
[1] 7.333272
```

The variance of the π estimator of the population mean can be estimated by first estimating the variance of the estimator of the psu totals, and dividing this variance by the squared number of population units:

$$\begin{aligned}\widehat{V}(\widehat{t}(z)) &= N^2 \left(1 - \frac{n}{N}\right) \frac{\widehat{S}^2(\widehat{t}_i(z))}{n} \\ \widehat{V}(\bar{z}) &= \frac{1}{M^2} \widehat{V}(\widehat{t}(z)) .\end{aligned}\tag{7.14}$$

```

fpc <- 1-n/N
v_tz <- N^2*fpc*var(tz_psu)/n
(se_mz_HT <- sqrt(v_tz/M^2))

```

```
[1] 0.6783605
```

The ratio estimator of the population mean and its standard error can be computed with function `svymean` of package `survey`.

```
mysample$fpc1 <- N
mysample$fpc2 <- M_psu[mysample$psu]
design_2stage <- svydesign(
  id=~psu+ssunits, fpc=~fpc1+fpc2, data=mysample)
svymean(~z, design_2stage)

      mean      SE
z 7.412 0.6547
```

The estimated standard error of the ratio estimator is slightly smaller than the standard error of the π estimator.

7.4 Stratified two-stage cluster random sampling

The basic sampling designs stratified random sampling (Chapter 4) and two-stage cluster random sampling can be combined into stratified two-stage cluster random sampling. Figure 7.3 shows a stratified two-stage cluster random sample from Voorst. The strata are $2\text{ km} \times 1\text{ km}$ blocks, as before in stratified cluster random sampling (Figure 6.2). The primary sampling units are $0.5\text{ km} \times 0.5\text{ km}$ blocks, as before in (unstratified) two-stage cluster random sampling (Figure 7.1). Within each stratum two times a psu is selected by `ppswr`, and every time a psu is selected, six ssu's (points) are selected by simple random sampling. The stratification avoids the clustering of the selected psu's in one part of the study area. Compared to (unstratified) two-stage cluster random sampling, the geographical spreading of the psu's is somewhat improved, which may lead to an increase of the precision of the estimated population mean.

```
n_h <- c(2,3,4)
m <- 6
set.seed(314)
stratumlabels <- unique(grdVoorst$zonestratum)
mysample <- NULL
for (i in 1:3) {
```

```

grd_h <- grdVoorst[grdVoorst$zonestratum==stratumlabels[i],]
mysample_h <- twostage(sframe=grd_h, psu="psu", n=n_h[i], m=m)
mysample <- rbind(mysample, mysample_h)
}
mysample$s1 <- jitter(mysample$s1, amount=cellsize/2)
mysample$s2 <- jitter(mysample$s2, amount=cellsize/2)

```

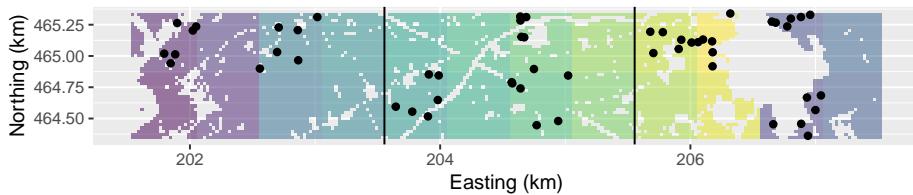


Figure 7.3: Stratified twostage random sample from Voorst. Strata are 2 km by 1 km blocks.

The population mean can be estimated in much in the same way as with stratified cluster random sampling. With function `svymean` this is an easy task.

```

N_h <- tapply(
  grdVoorst$psu, INDEX=grdVoorst$zonestratum,
  FUN=function(x) {length(unique(x))})
M_h <- tapply(
  grdVoorst$z, INDEX=grdVoorst$zonestratum, FUN=length)
mysample$w1 <- N_h[mysample$zonestratum]
mysample$w2 <- M_h[mysample$zonestratum]
design_str2stage <- svydesign(
  id=~psudraw+ssunits, strata=~zonestratum,
  weights=~w1+w2, data=mysample, nest=TRUE)
svymean(~z, design_str2stage)

      mean      SE
z 7.559 0.5259

```


Chapter 8

Sampling with probabilities proportional to size

In simple random sampling the inclusion probabilities are equal for all population units. The advantage of this is simple and straightforward statistical inference. With equal inclusion probabilities the unweighted sample mean is an unbiased estimator of the spatial mean, i.e. the sampling design is *self-weighting*. However, in some situations equal probability sampling is not very efficient, i.e. given the sample size the precision of the estimated mean or total will be relatively low. An example is the following. In order to estimate the total area of a given crop in a country, a raster of square cells of, for instance, 10 km x 10 km is constructed and projected on the country. The square cells are the population units, and these units serve as the sampling units. Note that near the country border cells cross the border. Some of them may contain only a few hectares of the target population, the country under study. We do not want to select many of these squares with only a few hectares of the study area, as intuitively it is clear that this will result in a low precision of the estimated crop area. In such situation it can be more efficient to select sampling units with probabilities proportional to the area of the target population within the squares, so that small sampling units near the border have smaller probability of being selected than interior sampling units. Actually, the sampling units are not the square cells, but the pieces of land obtained by overlaying the cells and the GIS map of the country under study. As a consequence the sampling units have unequal size,

i.e. the support is different (Chapter 1). The sampling units of unequal size are selected by probabilities proportional to their size (pps). In the previous Chapters 6 and 7 pps sampling was already used to select clusters (primary sampling units) of population units. In this chapter the *individual* population units (elementary sampling units) are selected with probabilities proportional to size.

If we have a GIS map of land use categories such as agriculture, built-up areas, water bodies, forests, etc., we may use this file to further adapt the selection probabilities. The crop will be grown in agricultural areas only, so we expect small crop areas in cells largely covered by non-agricultural land. As a size measure in computing the selection probabilities we may use the agricultural area (as represented in the GIS map) in the country under study within the cells. Note that size now has a different meaning. It does not refer to the area of the sampling units anymore, but to an ancillary variable that we expect to be related to the study variable, i.e. the crop area. When the crop area per cell is proportional to the agricultural area per cell, then the precision of the estimated total area of the crop can be increased by selecting the cells with probabilities proportional to the agricultural area.

In this example the sampling units have an area. However, sampling with probabilities proportional to size is not restricted to areal sampling units, but can also be used for selecting points. If we have a map of an ancillary variable that is expected to be (linearly) related to the study variable, this ancillary variable can be used as a size measure. For instance, in areas where soil organic carbon shows a positive (linear) relation with (relative) elevation, it can be efficient to select sampling points with a selection probability proportional to this environmental variable. The ancillary variable must be strictly positive for all points.

Sampling units can be selected with probabilities proportional to their size *with* or *without* replacement. This distinction is immaterial for infinite populations, as in sampling points from an area. pps sampling with replacement (ppswr) is much easier to implement than pps sampling without replacement (ppswor). The problem with ppswor is that after each draw the selected unit is removed from the sampling frame, so that the sum of the size variable over all remaining units changes, and as a result the draw-by-draw selection probabilities of the units.

pps sampling is illustrated with the simulated map of poppy area per 5 km × 5 km square in the province of Kandahar (Figure 1.6).

```
load("data/Kandahar.RData")
head(grdKandahar)

      x      y     agri     poppy
113 809.2319 3407.627 65.74340  0.90462202
114 814.2319 3412.627 15.64782  0.00452904
115 794.2319 3417.627 17.61077  11.26092257
116 809.2319 3417.627 14.02964  0.10979369
117 814.2319 3417.627 22.16968  0.03437396
118 819.2319 3417.627 13.34585  0.14278906
```

8.1 Probability-proportional-to-size sampling with replacement

In the first draw a sampling unit is selected with probability $p_k = x_k/t(x)$, with x_k the size variable for unit k and $t(x) = \sum_{k=1}^N x_k$ the population total of the size variable. The selected unit is then replaced, and these two steps are repeated n times. Note that with this sampling design population units can be selected more than once, especially with large sampling fractions n/N .

The population total can be estimated by the pwr estimator:

$$\hat{t}(z) = \frac{1}{n} \sum_{k \in S} \frac{z_k}{p_k}, \quad (8.1)$$

where n is the sample size (number of draws). The population mean can be estimated by the estimated population total divided by the population size N . With independent draws the sampling variance of the estimator of the population total can be estimated by

$$\widehat{V}(\hat{t}(z)) = \frac{1}{n(n-1)} \sum_{k \in S} \left(\frac{z_k}{p_k} - \hat{t}(z) \right)^2. \quad (8.2)$$

The sampling variance of the estimator of the mean can be estimated by the variance of the estimator of the total divided by N^2 . ppswr samples can be selected with function `sample.int`. As a first step I check whether the size

variable is strictly positive. The minimum equals 0.307 m^2 , so this is the case. If there are values equal to or smaller than 0 these must be replaced by a small number, so that these units also have a positive probability of being selected. Then the draw-by-draw selection probabilities are computed.

```
grdKandahar$p <- grdKandahar$agri/sum(grdKandahar$agri)
N <- nrow(grdKandahar)
n <- 40
set.seed(314)
units <- sample.int(N, size=n, replace=TRUE, prob=grdKandahar$p)
mysample <- grdKandahar[units,]
```

To select the units, computing the selection probabilities is not strictly needed. Exactly the same units are selected when the agricultural area within the units (column `agri` in the data frame) are used in argument `prob` of `sample.int`. Four units are selected twice:

```
table_frq <- table(units) %>% data.frame(.)
print(table_frq[table_frq$Freq>1,])
```

	units	Freq
9	278	2
13	334	2
14	336	2
24	439	2

Figure 8.1 shows the selected sampling units, plotted on a map of the agricultural area within the units, used as a size variable.

The next code chunk shows how the population total of the poppy area can be estimated, using Equation (8.1), as well as its standard error (Equation (8.2), followed by taking the square root). As a first step the observations are inflated (expanded) through division of the observations by the selection probabilities of the corresponding units.

```
z_pexpanded <- mysample$poppy/mysample$p
tz <- mean(z_pexpanded)
se_tz <- sqrt(var(z_pexpanded)/n)
```

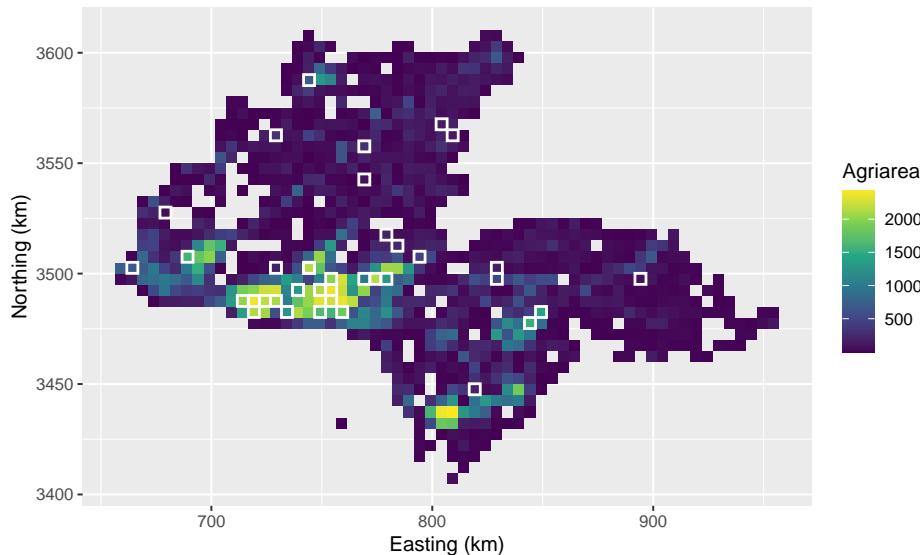


Figure 8.1: ppswr sample of size 40 from Kandahar, using agricultural area as a size variable. Four units are selected twice, so that the number of distinct units is 36.

The estimated total equals 65735 ha, with a standard error of 12944. The same estimates are obtained with package **survey** (?).

```
library(survey)
mysample$weight <- 1/(mysample$p*n)
design_ppswr <- svydesign(id=~1, data=mysample, weights=~weight)
svytotals(~poppy, design_ppswr)

      total      SE
poppy 65735 12944
```

In pps sampling with replacement a sampling unit can be selected more than once, especially with large sampling fractions n/N . This may decrease the sampling efficiency. With large sampling fractions the alternative is pps sampling without replacement, see next section. For infinite populations the probability that a unit is selected more than once is 0, so that there is no reason not to use

estimator of Equation (8.1). If the spatial population is discretised by a finite set of grid points, one must make the discretisation grid fine enough.

Exercises

1. Write an **R** script to select a pps with replacement sample from Eastern Amazonia to estimate the population mean of aboveground biomass (AGB), using log-transformed short-wave infrared (SWIR2) as a size variable. Use the 5 km x 5 km subgrid to reduce computing time (data are in `data/Amazonia_5km.RData`).
 - The correlation of AGB and lnSWIR2 is negative. The first step is to compute an appropriate size variable, so that the larger the size variable the larger the selection probability is. Multiply the lnSWIR2 values by -1. Then add a small value, so that the size variable becomes strictly positive.
 - Select in a for-loop 10,000 times a ppswr sample of size 100 ($n = 100$), and estimate from each sample the population mean of AGB with the pwr estimator (Hansen-Hurwitz estimator), and its sampling variance. Compute the variance of the 10,000 estimated population means, and the mean of the 10,000 estimated variances. Make a histogram of the 10,000 estimated means.
 - Compute the true sampling variance of the π estimator with simple random sampling with replacement and the same sample size.
 - Compute the gain in precision by the ratio of the variance of the estimator of the mean with simple random sampling to the variance with ppswr.

8.2 Probability-proportional-to-size sampling without replacement

The alternative to pps sampling with replacement (ppswr) is pps sampling without replacement (ppswor), i.e. sampling with inclusion probabilities proportional to a size variable. ppswor sampling is also referred to as π_{ps} sampling, stressing that the *inclusion* probabilities are proportional to a size variable, instead of the draw-by-draw selection probabilities. ppswor sampling starts with assign-

ing target inclusion probabilities to all units in the population. With inclusion probabilities proportional to a size variable x the target inclusion probabilities are computed by $\pi_k = n x_k / \sum_{j=1}^N x_j, k = 1 \dots N$.

8.2.1 Systematic pps sampling without replacement

Many algorithms are available for ppswor sampling, see ? for an overview. A simple, straightforward method is systematic ppswor sampling. The sampling frame is a list of the population units. Two subtypes can be distinguished, systematic ppswor sampling with fixed frame order and systematic ppswor sampling with random frame order (?). Given some order of the units, the cumulative sum of the inclusion probabilities is computed. Each population unit is then associated with an interval of cumulative inclusion probabilities. The larger the inclusion probability of a unit, the wider the interval. Then a random number from the uniform distribution is drawn, which serves as the start of a 1-dimensional systematic sample of size n with an interval of 1. Finally, the units are determined for which the systematic random values are in the interval of cumulative inclusion probabilities, see Figure 8.2 for ten population units and a sample size of four. The units selected are 2, 5, 7 and 9.

```
library(sampling)
set.seed(314)
N <- 10
x <- rnorm(N, mean=20, sd=5)
n <- 4
pi <- inclusionprobabilities(x, n)
print(df <- data.frame(id=seq(1:10), x, pi))
```

	id	x	pi
1	1	13.55882	0.3027383
2	2	23.63731	0.5277684
3	3	15.83538	0.3535687
4	4	16.48162	0.3679978
5	5	20.63624	0.4607613
6	6	18.32529	0.4091630
7	7	16.50655	0.3685545
8	8	20.06336	0.4479702
9	9	22.94495	0.5123095

```

10 10 11.15957 0.2491684

cumsumpi <- c(0, cumsum(pi))
start <- runif(1, min=0, max=1)
sys <- 0:(n-1)+start
print(units <- findInterval(sys, cumsumpi))

[1] 2 5 7 9

```

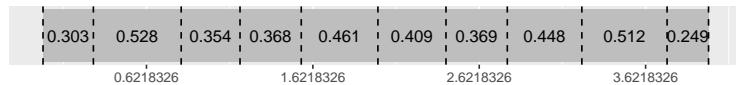


Figure 8.2: Systematic random sample along a line with unequal inclusion probabilities.

In Figure 8.2 the population units are in random order. Sampling efficiency can be increased by ordering the units in the frame, for instance in a way leading to an improved geographical spreading (see Section 9.2.2), or by the size variable. In Figure 8.3 the ten units are ordered by size. With this design the third, fourth, fifth and second unit in the original frame are selected, with sizes 15.8, 16.5, 20.6 and 23.6, respectively. Ordering the units by size leads to a large within-sample variance of the size variable, and a small between-sample variance. If the study variable is proportional to the size variable, this results in a smaller sampling variance of the estimator of the mean of the study variable. A drawback of systematic ppsworr sampling with fixed order is that no unbiased estimator of the sampling variance exists.

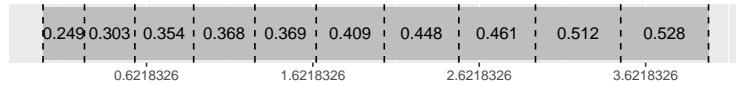


Figure 8.3: Systematic random sample along a line with unequal inclusion probabilities. Units are ordered by size.

A small simulation study is done next to see how much gain in precision can be achieved by ordering the units by size. A size variable x and a study variable z are simulated by drawing 1,000 values from a bivariate normal distribution with a correlation coefficient of 0.8. Function `mvtnorm` of package **MASS** (?) is used for the simulation.

```

library(MASS)
rho <- 0.8
mu1 <- 10; sd1 <- 2
mu2 <- 15; sd2 <- 4
mu <- c(mu1, mu2)
sigma <- matrix(data=c(sd1^2, sd1*sd2*rho, sd1*sd2*rho, sd2^2),
                 nrow=2, ncol=2)
N <- 1000
set.seed(314)
dat <- as.data.frame(mvrnorm(N, mu=mu, Sigma=sigma))
names(dat) <- c("z", "x")
head(dat)

      z         x
1 9.462930 9.149784
2 12.605847 17.306046
3 7.892686 11.979986
4 7.945021 12.567608
5 11.004325 15.165744
6 10.369943 13.258177

```

Twenty units are selected by systematic ppswor sampling with random order and ordered by size. This is repeated 10,000 times.

The standard deviation of the 10,000 estimated means with systematic ppswor sampling with random order is 0.336, and when ordered by size 0.321. So a small gain in precision is achieved through ordering the units by size. For comparison I also computed the standard error for simple random sampling without replacement (SI) of the same size. The standard error with this basic design is 0.424.

8.2.2 The pivotal method

Another interesting algorithm for ppswor sampling is the pivotal method (?). A nice adaptation of this algorithm, the local pivotal method, leading to samples with improved geographical spreading, is described in Section 9.2. In the pivotal method the N -vector with inclusion probabilities is successively updated to a vector with indicators. If the indicator value for sampling unit k becomes one, then this sampling unit is selected, if it becomes 0 then it is not selected. The

updating algorithm can be described as follows:

1. select randomly two units k and l with $0 < \pi_k < 1$ and $0 < \pi_l < 1$
2. If $\pi_k + \pi_l < 1$ then update the probabilities by

$$(\pi'_k, \pi'_l) = \begin{cases} (0, \pi_k + \pi_l) & \text{with probability } \frac{\pi_l}{\pi_k + \pi_l} \\ (\pi_k + \pi_l, 0) & \text{with probability } \frac{\pi_k}{\pi_k + \pi_l} \end{cases} \quad (8.3)$$

and if $\pi_k + \pi_l \geq 1$ then update the probabilities by

$$(\pi'_k, \pi'_l) = \begin{cases} (1, \pi_k + \pi_l - 1) & \text{with probability } \frac{1 - \pi_l}{2 - (\pi_k + \pi_l)} \\ (\pi_k + \pi_l - 1, 1) & \text{with probability } \frac{1 - \pi_k}{2 - (\pi_k + \pi_l)} \end{cases} \quad (8.4)$$

3. Replace (π_k, π_l) by (π'_k, π'_l) , and repeat the first two steps until each population unit is either selected (inclusion probability equals 1) or not selected (inclusion probability equals 0).

In words, when the sum of the inclusion probabilities is smaller than one, the updated inclusion probability of one of the units will become 0, which means that this unit will not be sampled. The inclusion probability of the other unit will become the sum of the two inclusion probabilities, which means that the probability increases that this unit will be selected in one of the subsequent iterations. The probability of a unit of being excluded from the sample is proportional to the inclusion probability of the other unit, so that the larger the inclusion probability of the other unit, the larger the probability that it will not be selected.

When the sum of the inclusion probabilities of the two units is larger than or equal to one, then one of the units is selected (updated inclusion probability is one), while the inclusion probability of the other is lowered by one minus the inclusion probability of the selected unit. The probability of being selected is proportional to the complement of the inclusion probability of the other unit. After the inclusion probability of a unit is updated to either 0 or 1, this unit cannot be selected anymore in the next iteration.

With this ppswor design the population total can be estimated by the π estimator:

$$\hat{t}(z) = \sum_{k \in \mathcal{S}} w_k z_k , \quad (8.5)$$

where $w_i = 1/\pi_i$. Note that the inclusion probabilities π_i are not the final probabilities obtained with the local pivotal method, which are either 0 or 1, but the initial inclusion probabilities. The π estimator of the mean is obtained simply by dividing the estimator for the total by the population size N .

An alternative estimator of the population mean is the Hajek estimator:

$$\hat{\bar{z}}_{\text{Hajek}}(z) = \frac{\sum_{k \in \mathcal{S}} w_k z_k}{\sum_{k \in \mathcal{S}} w_k} , \quad (8.6)$$

with $w_k = 1/\pi_k$. The denominator is an estimator of the population size N . The Hajek estimator of the total is obtained by multiplying the Hajek estimator of the mean with the population size N .

Various functions in package **sampling** (?) can be used to select a ppswor sample. In the code chunk below I use the function **UPRandompivotal**. With this function the order of the population units is randomized before function **UPpivotal** is used. In function **UPpivotal** in each iteration the first two units are selected. The argument **pi** is a numeric with the inclusion probabilities. These are computed with the function **inclusionprobabilities**. Recall that $\pi_k = n x_k / t(x)$. The sum of the inclusion probabilities should be equal to the sample size n . Function **UPpivotal** returns a numeric of length N with elements 1 and 0, 1 if the unit is selected, 0 if it is not selected. **eps** is a small number; the default value is 10^{-6} .

```
library(sampling)
n <- 40
size=ifelse(grdKandahar$agri<1E-12, 0.1, grdKandahar$agri)
pi <- inclusionprobabilities(size, n)
set.seed(314)
eps <- 1e-6
sampleind <- UPRandompivotal(pik=pi, eps=eps)
mysample <- data.frame(
  grdKandahar[sampleind==1,], pi=pi[sampleind==1])
nrow(mysample)
```

```
[1] 39
```

As can be seen not 40 but only 39 units are selected. If we replace `sampleind==1` by `sampleind>1-eps`, 40 units are selected.

```
mysample <- data.frame(grdKandahar[sampleind>1-eps,],
                        pi=pi[sampleind>1-eps])
nrow(mysample)
```

```
[1] 40
```

The total poppy area can be estimated from the ppswor sample by

```
tz_HT <- sum(mysample$poppy/mysample$pi)
tz_Hajek <- N*sum(mysample$poppy/mysample$pi)/
  sum(1/mysample$pi)
```

The total poppy area as estimated with the π estimator equals 88501 ha. The Hajek estimator results in a much smaller estimated total: 62169 ha.

The π estimate can also be computed with function `svytotal` of package `survey`, which also provides an approximate estimate of the standard error. Various methods are implemented in function `svydesign` for approximating the standard error. These methods differ in the way the pairwise inclusion probabilities are approximated from the unit-wise inclusion probabilities. These approximated pairwise inclusion probabilities are then used in the π variance estimator, or the Yates-Grundy variance estimator. In the next code chunks Brewer's method is used, see option 2 of Brewer's method in `?`, as well as Hartley-Rao's method for approximating the variance.

```
library(survey)
design_ppsworbrewer <- svydesign(
  id=~1, data=mysample, pps="brewer", fpc=~pi)
svytotal(~poppy, design_ppsworbrewer)

      total      SE
poppy 88501 14046
```

```
p2sum<-sum(mysample$pi^2)/n
design_ppsworhr <- svydesign(
  id=~1, data=mysample, pps=HR(p2sum), fpc=~pi)
svytot(~poppy, design_ppsworhr)
```

	total	SE
poppy	88501	14900

In package **samplingVarEst** (?) also various functions are available for approximating the variance: **VE.Hajek.Total.NHT**, **VE.HT.Total.NHT**, and **VE.SYG.Total.NHT**. The first variance approximation is the Hajek-Rosen variance estimator (see Equation 4.3 in ?). The latter two functions require the pairwise inclusion probabilities, which can be estimated by function **Pkl.Hajek.s**.

```
library(samplingVarEst)
se_tz_Hajek <- sqrt(VE.Hajek.Total.NHT(
  mysample$poppy, mysample$pi))
pikl <- Pkl.Hajek.s(mysample$pi)
se_tz_HT <- sqrt(VE.HT.Total.NHT(
  mysample$poppy, mysample$pi, pikl))
se_tz_SYG <- sqrt(VE.SYG.Total.NHT(
  mysample$poppy, mysample$pi, pikl))
```

The three standard errors equal 14045, 14068, and 14017. The differences in the approximated standard errors are small when related to the estimated total.

Figure 8.4 shows the sampling distributions of estimators of the total poppy area with ppswor sampling and simple random sampling without replacement of size 40, obtained by repeating the random sampling with each design and estimation 10,000 times. With the ppswor samples the total poppy area is estimated by the π estimator and the Hajek estimator. For each ppswor sample the variance of the π estimator is approximated by the Hajek-Rosen variance estimator (using function **VE.Hajek.Total.NHT** of package **samplingVarEst**).

Sampling design ppswor in combination with the π estimator is clearly much more precise than simple random sampling. The standard deviation of the 10,000 π estimates of the total poppy area with ppswor equals 11684. The

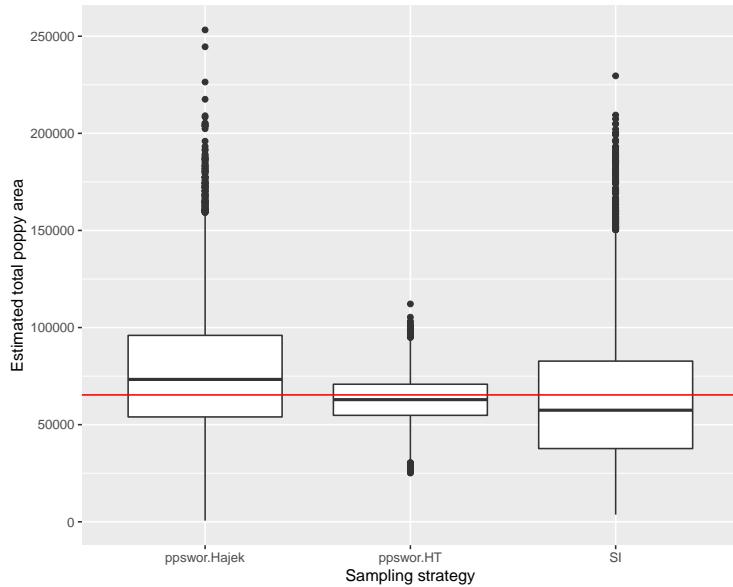


Figure 8.4: Sampling distribution of estimators of the total poppy area (ha) in Kandahar with ppswor sampling and simple random sampling without replacement (SI) of 40 units. In ppswor sampling the total poppy area is estimated by the π estimator (ppswor.HT) and Hajek estimator (ppswor.Hajek).

average of the square root of the Hajek-Rosen approximate variances equals 12332.

Interestingly, with ppswor sampling the variance of the 10,000 Hajek estimates is much larger than that of the π estimates. The standard deviation of the 10,000 Hajek estimates with ppswor sampling is about equal to that of the π estimates with simple random sampling: 3.1234×10^4 and 3.3178×10^4 , respectively.

Exercises

2. A field with poppy was found outside Kandahar in a selected sampling unit crossing the boundary. Should this field be included in the sum of the poppy area of that sampling unit?
3. In another sampling unit a poppy field was encountered in Kandahar but

in the area represented as non-agriculture in the GIS map. Should this field be included in the sum of that sampling unit?

Chapter 9

Balanced and well-spread sampling

In this chapter two sampling designs are described and illustrated that are related but also fundamentally different. The similarities and difference are shortly described, but will become more clear in following sections. Roughly speaking, a balanced sample is a sample of which the sample mean of one or more covariates is equal to the population means of these covariates. When the covariates are linearly related to the study variable this may yield a more accurate estimate of the population mean or total.

A well-spread sample is a sample with a large range of values for the covariates, from small to large values, but also intermediate values. In more technical terms: the sampling units are well-spread along the axes spanned by the covariates. If the spatial coordinates are used as covariates (spreading variables), this results in samples that are well-spread in geographical space. Such samples are commonly referred to as spatially balanced samples, which is somewhat confusing, as the geographical spreading is not implemented through balancing on the geographical coordinates. On the other hand, the averages of the spatial coordinates of a sample well-spread in geographical space will be close to the population means of the coordinates, and therefore will be approximately balanced on the spatial coordinates (?). The reverse is not true: with balanced sampling the spreading of the sampling units in the space spanned by the bal-

ancing variables can be poor. A sample with all values of a covariate used in balancing near the population mean of that variable has a poor spreading along the covariate axis, but can still be perfectly balanced.

9.1 Balanced sampling

Balanced sampling is a sampling method that exploits one or more quantitative covariates that are related to the study variable. The idea behind balanced sampling is that, if we know the mean of the covariates, then the sampling efficiency can be increased by selecting a sample whose averages of the covariates must be equal to the population means of the covariates.

The simulated population shown in Figure 9.1 shows a linear trend from West to East, and also a trend from South to North, but we will ignore this South to North trend for the moment. In other words, the simulated study variable z is correlated with the covariate Easting. To estimate the population mean of the simulated study variable, intuitively it is attractive to select a sample with an average of the Easting coordinate that is equal to the population mean of Easting (which is 10). Figure 9.1(a) shows such a sample; we say that the sample is ‘balanced’ on the covariate Easting.

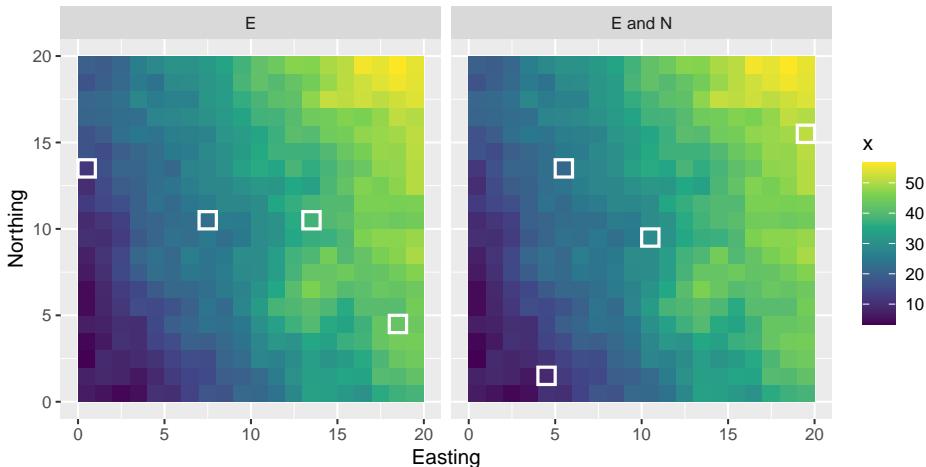


Figure 9.1: Sample balanced on Easting (E) and on Easting and Northing (E and N).

9.1.1 Balanced sample versus balanced sampling design

We must distinguish a balanced *sample* from a balanced sampling *design*. A sampling design is balanced on a covariate x when *all possible* samples that can be generated by the design are balanced on x . So, simple random sampling is not a balanced sampling design, because for many simple random samples the sample mean of x is not equal to the population mean of x . Only the *expectation* of the sample mean of x , i.e. the mean of the sample mean over an infinite number of simple random samples, equals the population mean of x .

Figure 9.2 shows for one thousand simple random samples the squared error of the estimated population mean of the study variable z against the difference between the sample mean of x and the population mean of x .

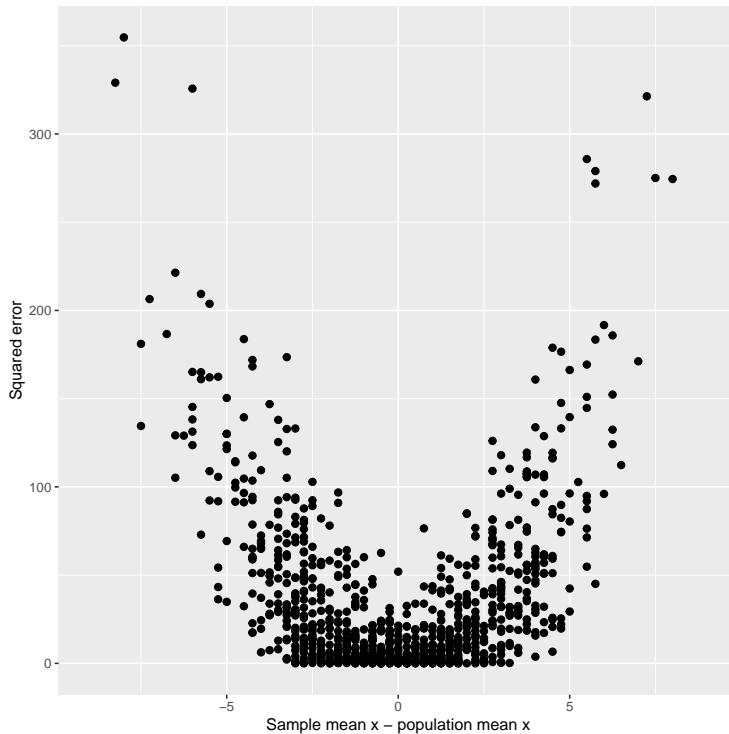


Figure 9.2: Squared error in estimated mean of z against difference between population and sample mean of a covariate.

Table 9.1: Sampling variances of estimated mean for simple random sampling and balanced sampling of four units.

Sampling design	Balancing variables	Sampling variance
SI	-	39.70
Balanced	Easting	14.40
Balanced	Easting+Northing	9.77

Clearly, the larger the absolute value of the difference, the larger on average the squared error. So to obtain an accurate estimate of the population mean of z , we better select samples with a difference close to 0.

Sampling designs can also be balanced on multiple covariates. Figure 9.1(b) shows a sample balanced on both Easting and Northing. Using Easting as a balancing variable reduced the sampling variance of the estimator of the mean substantially, see Table 9.1. Using Northing as a second balancing variable further reduced the sampling variance.

9.1.2 Unequal inclusion probabilities

Until now we assumed that the inclusion probabilities of the population units are equal, but this is not a requirement for balanced sampling designs. A more general definition is: a sampling design is balanced on variable x when for all samples generated by the design the π estimator of the population total of x equals the population total of x :

$$\sum_{k \in S} \frac{x_k}{\pi_k} = \sum_{k=1}^N x_k , \quad (9.1)$$

with π_k the inclusion probability of unit k , x_k the covariate of unit k , and N the total number of units in the population.

Similar to the regression estimator (Section 10.1.1), balanced sampling exploits the linear relation between the study variable and one or more covariates. In the regression estimator this is done at the estimation stage. Balanced sampling does so at the sampling stage. For a single covariate the regression estimator of the population total equals (see Equation (10.10))

$$\hat{t}_{\text{regr}}(z) = \hat{t}_\pi(z) + \hat{b} (t(x) - \hat{t}_\pi(x)) , \quad (9.2)$$

with $\hat{t}_\pi(z)$ and $\hat{t}_\pi(x)$ the π estimators of the population total of the study variable z and the covariate x , respectively, $t(x)$ the population total of the covariate, and \hat{b} the estimated slope parameter (see hereafter). With a perfectly balanced sample the adjustment term in the regression estimator (the second term) equals zero.

Balanced samples can be selected with the cube algorithm of ?, see also ? for a detailed description of this algorithm. The population total can be estimated by the π estimator:

$$\hat{t}(z) = \sum_{k \in \mathcal{S}} \frac{z_k}{\pi_k} . \quad (9.3)$$

So with equal inclusion probabilities, equal to n/N , the population total is estimated by the sample mean of the study variable multiplied by the number of population units N . The population mean is estimated by dividing the estimated population total by N .

The approximate variance of the π estimator of the population mean can be estimated by (?, ?)

$$\widehat{V}(\hat{z}) = \frac{1}{N^2} \frac{n}{n-p} \sum_{k \in \mathcal{S}} c_k \left(\frac{\epsilon_k}{\pi_k} \right)^2 , \quad (9.4)$$

with p the number of balancing variables, c_k a weight for unit k (see hereafter), and ϵ_k the residual of unit k given by

$$\epsilon_k = z_k - \mathbf{x}_k^T \hat{\mathbf{b}} , \quad (9.5)$$

with \mathbf{x}_k a vector of length p with the balancing variables for unit k , and $\hat{\mathbf{b}}$ the estimated population regression coefficients, given by

$$\hat{\mathbf{b}} = \left(\sum_{k \in \mathcal{S}} c_k \frac{\mathbf{x}_k (\mathbf{x}_k)^T}{\pi_k \pi_k} \right)^{-1} \sum_{k \in \mathcal{S}} c_k \frac{\mathbf{x}_k z_k}{\pi_k \pi_k} . \quad (9.6)$$

Working this out for balanced sampling without replacement with equal inclusion probabilities, $\pi_k = n/N, k = 1, \dots, N$, yields

$$\widehat{V}(\widehat{z}) = \frac{1}{n(n-p)} \sum_{k \in \mathcal{S}} c_k \epsilon_k^2 . \quad (9.7)$$

? give several formulas for computing the weights c_k , one of which is $c_k = (1 - \pi_k)$.

Balanced sampling is now illustrated with the aboveground biomass (AGB) data of Eastern Amazonia, see Figure 1.8. Log-transformed short-wave infrared ($\ln\text{SWIR2}$) is used as a balancing variable. The `samplecube` function of the `sampling` package (?) implements the cube algorithm. Argument `X` of this function specifies the matrix of ancillary variables on which the sample must be balanced. The first column of this matrix must be filled with ones, so that the sample size is fixed. Equal inclusion probabilities are used, i.e. for all population units the inclusion probability equals n/N .

```
load("data/Amazonia_5km.RData")
gridAmazonia$lnSWIR2 <- log(gridAmazonia$SWIR2)
library(sampling)
N <- nrow(gridAmazonia)
n <- 100
X <- cbind(rep(1,times=N), gridAmazonia$lnSWIR2)
pi <- rep(n/N,times=N)
sample_ind <- samplecube(X=X, pik=pi, comment=FALSE, method=1)
eps <- 1e-6
units <- which(sample_ind>(1-eps))
mysample <- gridAmazonia[units,]
```

The population mean can be estimated by the sample mean.

```
mz_sample <- mean(mysample$AGB)
```

To estimate the variance a function is defined for estimating the population regression coefficients.

```

estimate_b <- function(z,X,c) {
  cXX <- matrix(nrow=ncol(X),ncol=ncol(X),data=0)
  cXz <- matrix(nrow=1,ncol=ncol(X),data=0)
  for (i in 1:length(z)) {
    x <- X[i,]
    cXX_i <- c[i]*(x %*% t(x))
    cXX <- cXX+cXX_i
    cXz_i <- c[i]*t(x)*z[i]
    cXz <- cXz+cXz_i
  }
  b <- solve(cXX,t(cXz))
  b
}

```

The next code chunk shows how the variance of the π estimator of the population mean can be estimated.

```

pi <- rep(n/N,n)
c <- (1-pi)
b <- estimate_b(z=mysample$AGB/pi, X=X[units,]/pi, c=c)
zpred <- X%*%b
e <- mysample$AGB-zpred[units]
v_tz <- n/(n-ncol(X))*sum(c*(e/pi)^2)
v_mz <- v_tz/N^2

```

Figure 9.3 shows the result. The sample mean of AGB equals 224.5. The population mean of AGB equals 225.3. Note the spatial clustering of some units. The standard error of the estimated mean equals 6.1.

Figure 9.4 shows the sampling distributions of the π estimator of the mean of AGB with balanced sampling and simple random sampling, obtained by repeating the random sampling with both designs and estimation 1,000 times.

The variance of the 1000 estimates of the population mean of the study variable AGB equals 28.8. The gain in precision compared to simple random sampling, equals 3, so with simple random sampling three times more sampling units are needed to estimate the population mean with the same precision. The mean of the 1,000 estimated variances equals 26.4, indicating that the approximate

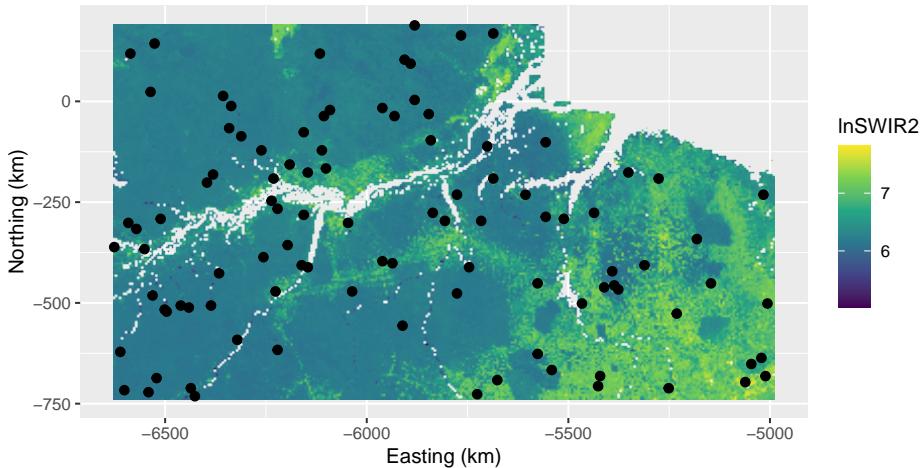


Figure 9.3: Balanced sample from Eastern Amazonia (Brazil), balanced on covariate lnSWIR2.

variance estimator somewhat underestimates the true variance in this case. The population mean of the balancing variable lnSWIR2 equals 6.414. The sample mean of lnSWIR2 varies a bit among the samples (Figure 9.5). In other words, many samples are not perfectly balanced on lnSWIR2. This is not exceptional, in most cases perfect balance is impossible.

9.1.3 Stratified random sampling

In the previous section a continuous variable was used to balance the sample. However, also a categorical variable can be used for this. A sample balanced on a categorical variable actually is a stratified random sample. Figure 9.6 shows four strata. These four strata can be used in balanced sampling by constructing the following design matrix \mathbf{X} with as many columns as there are classes:

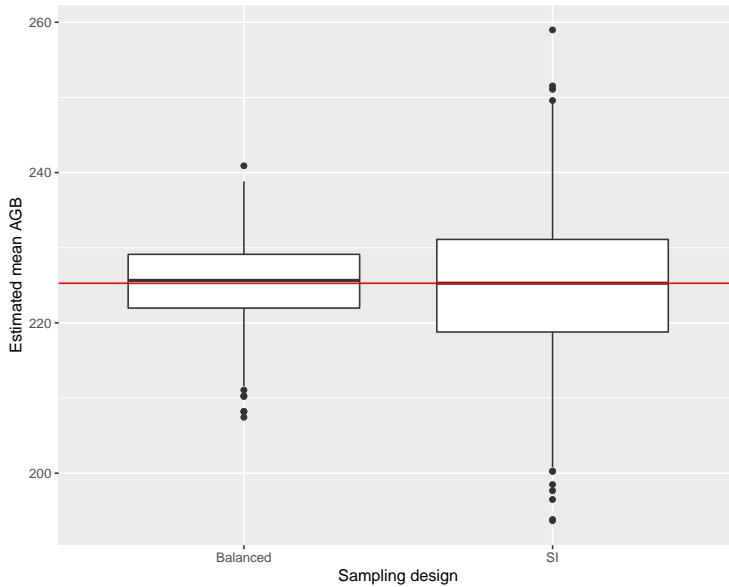


Figure 9.4: Sampling distribution of π estimator of the mean aboveground biomass in Eastern Amazonia, with balanced sampling (balanced on lnSWIR2) and simple random sampling, bot designs with a sample size of 100 units.

$$\begin{bmatrix} \pi_{1,1} & 0 & 0 & 0 \\ \pi_{2,1} & 0 & 0 & 0 \\ \pi_{3,1} & 0 & 0 & 0 \\ \pi_{4,1} & 0 & 0 & 0 \\ & \pi_{5,2} & 0 & 0 \\ 0 & \pi_{6,2} & 0 & 0 \\ 0 & 0 & \pi_{7,3} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \pi_{400,4} \end{bmatrix}, \quad (9.8)$$

The first four rows refer to the four leftmost bottom row population units in Figure 9.6. These units belong to class A, which explains that the first column for these units contain non-zeroes. These non-zeroes are the inclusion probabilities of the units in stratum A. The other three columns for these rows contain

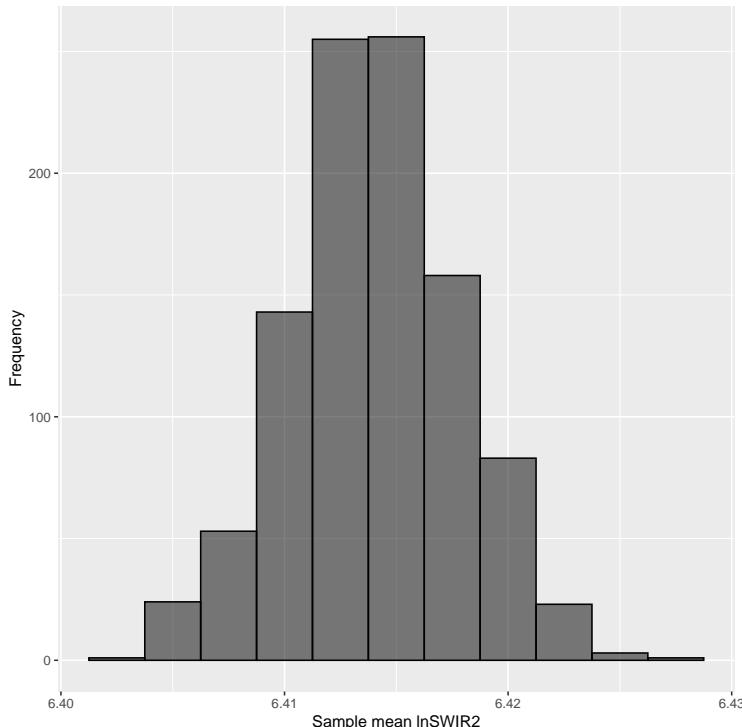


Figure 9.5: Sampling distribution of sample mean of balancing variable lnSWIR2.

all zeroes. The fifth and sixth unit belong to stratum B, so that the second column for these rows contain the inclusion probabilities for stratum B, and so on. The final row is the upper-right sampling unit in stratum D, so the first three columns contain zeroes, and the fourth column is filled with the inclusion probability of this stratum. The sum of the inclusion probabilities in the first column is the sample size of stratum A. Or reversely, if, for instance, we want to select n_h units from stratum h with equal probability for all units in this stratum, then the inclusion probabilities should equal n_h/N_h , with N_h the total number of units in this stratum.

As a first step inclusion probabilities are computed by $\pi_{hk} = n_h/N_h, k = 1, \dots, N_h$, with $n_h = N_h/N$ (proportional allocation).

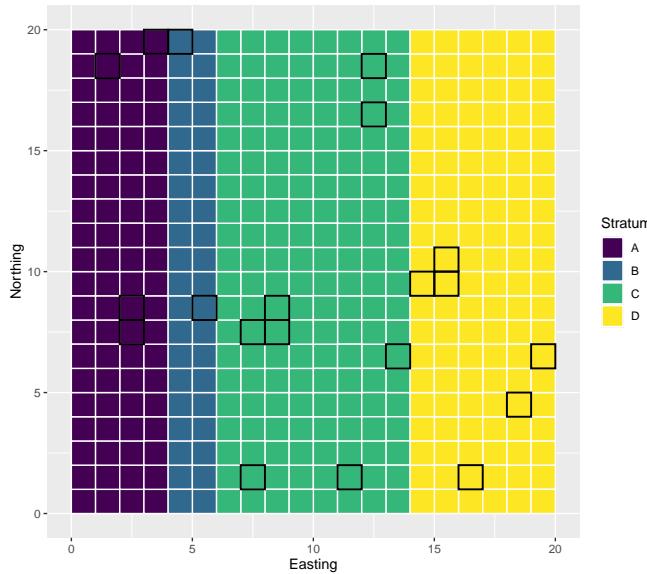


Figure 9.6: Sample balanced on a categorical variable with four classes.

```

N_h <- tapply(mypop$s1, INDEX=mypop$stratum, FUN=length)
n <- 20
n_h <- n*N_h/sum(N_h)
labels <- sort(unique(mypop$stratum))
lut <- data.frame(stratum=labels, pi=n_h/N_h)
mypop<- merge(x=mypop, y=lut, by="stratum")

```

The design matrix \mathbf{X} is computed with function `model.matrix`, expanding the factor `stratum` to a set of dummy variables. By adding `-1` to the formula, we avoid that the first column in the design matrix has all ones. The design matrix has four columns with dummy variables (indicators), indicating to which stratum a unit belongs.

The columns in the design matrix with dummy variables are multiplied by the vector with inclusion probabilities, using function `sweep`. This is not strictly needed. Using the design matrix with dummy variables implies that the population totals equal the number of population units in the strata, N_h . For a per-

fectedly balanced sample, the sample sums of the balancing variables, the dummy variables, divided by the inclusion probability are also equal to N_h . Multiplication of the dummy variables with the vector with inclusion probabilities implies that the population totals equal the targeted sample sizes per stratum. For a perfectly balanced sample, the sample sums of the balancing variable (having value π_{hk} or 0), divided by the inclusion probability are also equal to n_h .

```
X <- model.matrix(~ stratum-1, data=mypop)
X <- sweep(X, MARGIN= 1, mypop$pi, `*`)
set.seed(314)
sample_ind <- samplecube(
  X=X, pik=mypop$pi, comment=FALSE, method=1)
mysample <- mypop[sample_ind>(1-eps),]
```

In this case all units in a stratum have the same inclusion probability, yielding a stratified simple random sample. We may also use variable inclusion probabilities, for instance proportional to a size measure of the units, yielding a stratified pps random sample.

The advantage of selecting a stratified random sample by balancing the sample on a categorical variable becomes clear in case we have multiple classifications that we would like to use in stratification, and we cannot afford to use all cross-classifications as strata. This is the topic of the next section.

9.1.4 Multi-way stratification

? describe how a multi-way stratified sample can be selected as a balanced sample. Multi-way stratification is of interest when one has multiple stratification variables, each stratification variable leading to several strata, so that the total number of cross-classification strata becomes so large that the stratum sample sizes are strongly disproportional to their size, or even exceed the total sample size.

Let M be the sum of the number of map units over all maps used for stratification. For instance, if we have three maps with $4 + 3 + 6$ map units, M equals 13. Instead of using all cross-classification map units as strata, the M map units are used as strata. The sample sizes of the marginal strata can be controlled by using a design matrix with as many columns as there are strata. The units of an individual map used for stratification are referred to as marginal strata.

Each row $k = 1, \dots, N$ in the design matrix \mathbf{X} has as many non-zero values as we have maps, in entries corresponding with the cross-classification map unit population unit k belongs to, and zeroes in the remaining entries. The non-zero value is the inclusion probability of that unit. The inclusion probability of a unit is independent of the map used for stratification, so the non-zero values in a row are all equal. Each column of the design matrix has non-zero values at entries corresponding with the population units in that marginal stratum, and zeroes at all other entries.

Two-way stratified random sampling is illustrated with a simulated population of 400 units (Figure 9.7).

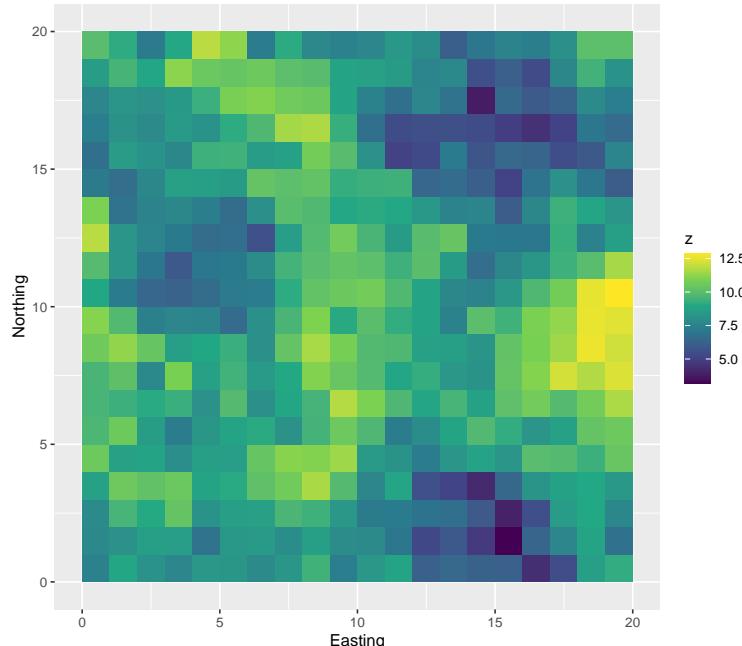


Figure 9.7: Simulated population, used for illustration of two-way stratified random sampling.

Figure 9.8 shows two classifications of the population units. Classification A consists of four classes (map units), classification B of three classes. Instead of using $4 \times 3 = 12$ cross-classifications as strata in random sampling, only $4+3 = 7$

marginal strata are used in two-way stratified random sampling.

As a first step the inclusion probabilities are added to the data frame `mypop` with the spatial coordinates and simulated values. To keep it simple I computed inclusion probabilities equal to two divided by the number of population units in a cross-classification stratum. Note that this does not imply that a sample is selected with two units per cross-stratum. As we will see later it is possible that in some cross-classification stratum no units are selected at all, while in other cross-classification strata more than two units are selected. In multi-way stratified sampling the marginal stratum sample sizes are controlled. The inclusion probabilities should result in six selected units for all four units of map A, and eight selected units for all three units of map B.

```
mypop <- mypop %>%
  group_by(A, B) %>%
  summarise(N_h=n(), .groups="drop") %>%
  mutate(pih=rep(2,12)/N_h) %>%
  right_join(mypop, by=c("A", "B"))
```

The next step is to create the design matrix. Two submatrices are computed, one per stratification. The two submatrices are joined column-wise, using function `cbind`. The columns are multiplied by the vector with inclusion probabilities.

```
XA <- model.matrix(~A-1, mypop)
XB <- model.matrix(~B-1, mypop)
X <- cbind(XA,XB)
X <- sweep(X, MARGIN= 1, mypop$pih, `*`)
```

Matrix `X` can be reduced by one column if in the first column the inclusion probabilities of *all* population units are inserted. This first column contains no zeroes. Balancing on this variable implies that the total sample size is controlled. Now there is no need anymore to control the sample sizes of all marginal strata. It is sufficient to control the sample sizes of three marginal strata of map A and two marginal strata of map B.

```
X <- model.matrix(~A+B, mypop)
X <- sweep(X, MARGIN= 1, mypop$pih, `*`)
```

This reduced design matrix is not strictly needed for selecting a multi-way stratified sample, but must be used in estimation. If in estimation as many balancing variables are used as we have marginal strata, the matrix with the sum of squares of balancing variables in Equation (9.6)) cannot be inverted (matrix is singular), and as a consequence the population regression coefficients cannot be estimated.

Finally, the two-way stratified random sample is selected with function `samplecube` of package `sampling` (?).

```
sample_ind <- samplecube(
  X=X, pik=mypop$pih, method=1, comment=FALSE)
eps <- 1e-6
units <- which(sample_ind>(1-eps))
mysample <- mypop[units,]
```

Figure 9.8 shows the selected sample.

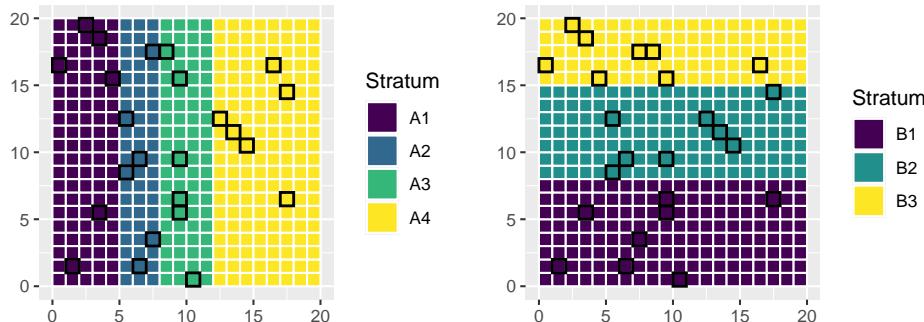


Figure 9.8: Two-way stratified sample.

All marginal sample sizes of map A are six, and all marginal sample sizes of map B are eight, as expected. The sample sizes of the cross-classification strata vary from zero to four.

```
addmargins(table(mysample$A, mysample$B))
```

	B1	B2	B3	Sum
A1	1	1	1	3
A2	1	1	1	3
A3	1	1	1	3
A4	1	1	1	3

A1	2	0	4	6
A2	2	3	1	6
A3	3	1	2	6
A4	1	4	1	6
Sum	8	8	8	24

The population mean can be estimated by the π estimator.

```
N <- nrow(mypop)
print(mean <- sum(mysample$z/mysample$pih)/N)
```

```
[1] 8.688435
```

The variance is estimated as before (Equation (9.4)).

```
c <- (1-mysample$pih)
b <- estimate_b(
  z=mysample$z/mysample$pih, X=X[,]/mysample$pih, c=c)
zpred <- X%*%b
e <- mysample$z-zpred[,]
n <- nrow(mysample)
v_tz <- n/(n-ncol(X))*sum(c*(e/mysample$pih)^2)
print(v_mz <- v_tz/N^2)
```

```
[1] 0.1723688
```

Exercises

1. Spatial clustering of sampling units is not avoided in balanced sampling. What effect do you expect of this spatial clustering on the precision of the estimated mean? Can you think of a situation where this effect does not occur?

9.2 Well-spread sampling

With balanced sampling the spreading of the sampling units in the space spanned by the balancing variables can be poor. For instance, in Figure 9.1(a) the Easting coordinates of all units of a sample balanced on Easting can be

equal or close to the population mean of 10. So, in this example balancing does not guarantee a good geographical spreading. A balanced sample can be selected that shows strong clustering in the space spanned by the balancing variables. This clustering may inflate the standard error of the estimated population total and mean. The clustering in geographical or covariate space can be avoided by the local pivotal method (?), and the spatially correlated Poisson sampling method (?).

For spreading in *geographical* space various other designs are available. A simple design is stratified random sampling from compact geographical strata, see Section 4.6. Alternative designs are generalised random-tessellation stratified sampling (?), and balanced acceptance sampling (?).

9.2.1 Local pivotal method

The local pivotal method (LPM) is a modification of the pivotal method explained in Section 8.2. The only difference with the pivotal method is the selection of the pairs of units. In the pivotal method at each step two units are selected, for instance, the first two units in the vector with inclusion probabilities after randomizing the order of the units. In the local pivotal method the first unit is selected fully randomly and the nearest neighbour of this unit is used as its counterpart. Recall that when one unit of a pair is included in the sample, the inclusion probability of its counterpart is decreased. This leads to a better spreading of the sampling units in the space spanned by the spreading variables.

LPM can be used for arbitrary inclusion probabilities. The inclusion probabilities can be equal, but as in the pivotal method these probabilities may also differ among the population units.

Selecting samples with LPM can be done with functions `lpm`, `lpm1` or `lpm2` of package **BalancedSampling** (?). The functions `lpm1` and `lpm2` only differ in the selection of neighbours that are allowed to compete, for details see ?. For most populations the two algorithms perform similar (personal communication Anton Grafström). The algorithm implemented in the function `lpm` is only recommended when the population size is too large for `lpm1` or `lpm2`. It only uses a subset of the population in search for nearest neighbours, and is thus not as good. Another function `lpm2_kdtree` of package **SamplingBigData** (?) is developed for big data sets.

Inclusion probabilities are computed with function `inclusionprobabilities` of

package **sampling**. A matrix **X** must be defined with the values of the spreading variables of the population units. Figure 9.9 shows a sample of 40 units selected from the sampling frame of Kandahar, using the spatial coordinates of the population units as spreading variables. Inclusion probabilities are proportional to the agricultural area within the population units. The geographical spreading is improved compared with the sample shown in Figure 8.1.

```
library(BalancedSampling)
library(sampling)
load("data/Kandahar.RData")
n <- 40
pi <- inclusionprobabilities(grdKandahar$agri, n)
X <- cbind(grdKandahar$x, grdKandahar$y)
set.seed(314)
units <- lpm1(pi, X)
myLPMsample <- grdKandahar[units,]
```

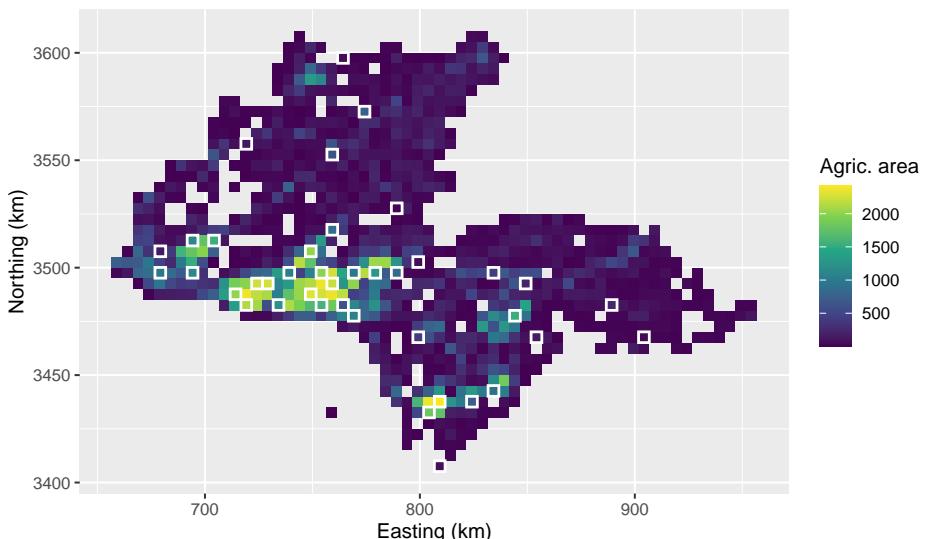


Figure 9.9: Spatial ppswor sample selected by local pivotal method, using agricultural area as size variable.

The total poppy area can be estimated with the π estimator (Equation (8.5)).

```
myLPMsample$pi <- pi[units]
tz_HT <- sum(myLPMsample$poppy/myLPMsample$pi)
```

The estimated total poppy area equals 56420 ha. The sampling variance of the estimator of the population total with the local pivotal method can be estimated by (?)

$$\widehat{V}(\hat{t}(z)) = \frac{1}{2} \sum_{k \in S} \left(\frac{z_k}{\pi_k} - \frac{z_{k_j}}{\pi_{k_j}} \right)^2, \quad (9.9)$$

with k_j the nearest neighbour of unit k in the sample. This variance estimator is for the case where we have only one nearest neighbour.

Function `vsb` of package **BalancedSampling** is an implementation of a more general variance estimator that accounts for more than one nearest neighbour (equation 6 in ?). We expect a somewhat smaller variance compared to pps sampling, so we may use the variance of the pwr estimator (Equation (8.2)) as a conservative variance estimator.

```
Xsample <- X[units,]
se_tz_HT <- sqrt(vsb(pi[units], myLPMsample$poppy, Xsample))
pk <- myLPMsample$pi/n
se_tz_pwr <- sqrt(var(myLPMsample$poppy/pk)/n)
```

The standard error obtained with function `vsb` equals 14222, the standard error of the Hansen-Hurwitz estimator equals 13468. So in this case the Hansen-Hurwitz variance estimator is smaller than the other variance estimator, but on average it will be larger.

As explained above, the LPM design can also be used to select a probability sample well-spread in the space spanned by one or more quantitative covariates. Matrix **X** then should contain the values of the *scaled* (standardised) covariates instead of the spatial coordinates.

Exercises

2. Geographical spreading of the sampling units can also be achieved by random sampling from compact geographical strata (Section 4.6). Can you think of one or more advantages of LPM sampling over random sampling from geostrata?

9.2.2 Generalised random-tessellation stratified sampling

Generalised random-tessellation stratified sampling (GRTS) is designed for sampling discrete objects scattered throughout space, think for instance of the lakes in Finland, segments of hedgerows in England etc. Each object is represented by a point in 2D-space. It is a complicated design, and for sampling points from a continuous universe, or raster cells from a finite population, I recommend more simple designs such as the local pivotal method (Section 9.2.1), balanced sampling with geographical spreading (Section 9.3), or sampling from compact geographical strata (Section 4.6). Let me try to explain the GRTS design with a simple example of a finite population of point objects in a circular study area (Figure 9.10). For a more detailed description of this design I refer to ?. As a first step a square bounding box of the study area is constructed. This bounding box is recursively partitioned into square grid cells. First 2×2 grid cells are constructed. These grid cells are numbered in a predefined order. In Figure 9.10(b) this numbering is from lower left, lower right, upper left to upper right. Each grid cell is then subdivided into four subcells; the subcells are numbered using the same order. This is repeated until at most one population unit occurs in each subcell. For our population only two iterations were needed, leading to 4×4 subcells. Note that in some subcells no population unit occurs. Each address of a subcell consists of two digits, the first digit is for the grid cell, the second digit for the subcell.

The next step is to place the sixteen subcells on a line in a random order. The randomisation is done hierarchically. First the four grid cells at the highest level are randomized. In our example the randomized order is 1, 2, 3, 0 (Figure 9.11). Then within each grid cell the order of the subcells is randomized. This is done independently for the grid cells. In our example for grid cell 1 the randomized order of the subcells is 2, 1, 3, 0 (Figure 9.11). Note that the empty subcells, subcells (0,0) and (3,3) are removed from the line.

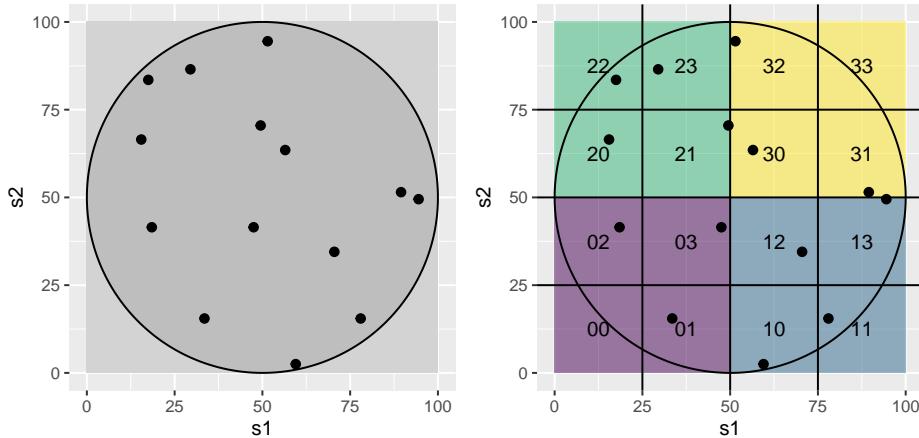


Figure 9.10: Numbering of grid cells and subcells for GRTS sampling.

```
set.seed(314)
ord <- sample.int(4,4)
myfinpop_rand <- NULL
for (i in ord) {
  units <- which(myfinpop$partit1==i)
  units_rand <- sample(units, size=length(units))
  myfinpop_rand <- rbind(myfinpop_rand, myfinpop[units_rand,])
}
```

After the subcells are placed on a line, a one-dimensional systematic random sample is selected (Figure 9.11), see also Section 8.2.1. This can be done either with equal or unequal inclusion probabilities. With equal inclusion probabilities the length of the lines representing the population units is constant. With unequal inclusion probabilities the length of the lines is proportional to a size variable. For a sample size of n , the total line is divided into n segments of equal length. A random point is selected in the first segment, and the other points of the systematic sample are determined. Finally, the population units corresponding with the selected systematic sample are identified. With equal probabilities the five selected units are the units in subcells 11, 23, 22, 32 and 03 (Figure 9.11).

```

size <- rep(1,N)
n <- 5
interval <- sum(size)/n
start <- round(runif(1)*interval,2)
mysys <- c(start,1:(n-1)*interval+start)

```

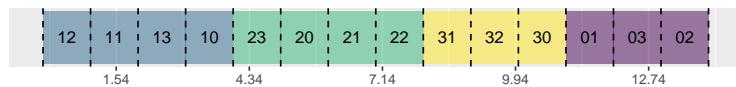


Figure 9.11: Systematic random sample along a line with equal inclusion probabilities.

Figure 9.12 shows a systematic random sample along a line with unequal inclusion probabilities. The inclusion probabilities are proportional to a size variable, with values 1, 2, 3 or 4. The selected population units are the units in subcells 10, 20, 31, 01 and 02.

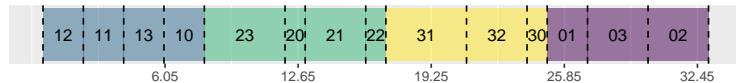


Figure 9.12: Systematic random sample along a line with inclusion probabilities proportional to size.

GRTS samples can be selected with function `grts` of package `spsurvey` (?). The next code chunk shows the selection of a GRTS sample of 40 units from Kandahar. First a data frame is created representing the sampling frame. With unequal inclusion probabilities this data frame must include a variable for the inclusion probabilities. Next, a named list specifying the sampling design is created. The first element specifies how many units must be selected. In case of stratified random sampling these sample sizes must be set per stratum. Also, more than one sample can be selected (per stratum), referred to as panels. Per sampling round only one panel is observed. After multiple rounds the sample data can be used for estimating the temporal change of the spatial mean or total. The element `seltype` in the design list must be set to “Continuous” for sampling with probabilities proportional to an ancillary variable specified with argument `mdcaty`. If the argument `shift.grd` is set to TRUE (the default value), the hierarchical grid is shifted to a random position.

```

library(spsurvey)
n <- 40
pi <- inclusionprobabilities(grdKandahar$agri, n)
N <- nrow(grdKandahar)
samplingframe <- data.frame(
  x=grdKandahar$x, y=grdKandahar$y, mdcaty=pi, ids=1:N)
design <- list(
  None=list(panel=c(PanelOne=n), seltype="Continuous"))
set.seed(314)
res <- grts(
  design, type.frame="finite", src.frame="att.frame",
  att.frame=samplingframe, xcoord="x", ycoord="y",
  mdcaty="mdcaty", do.sample=TRUE, shapefile=FALSE)

```

```

Stratum: None
Current number of levels: 3
Current number of levels: 5
Current number of levels: 6
Final number of levels: 6

```

```

units <- res$ids
myGRTSsample <- grdKandahar[units,]

```

The total poppy area is estimated by the π estimator.

```

tz_GRTS <- sum(myGRTSsample$poppy/pi[units])

```

The estimated total is 56979. Function `vsb` of package **BalancedSampling** can be used to estimate the standard error of the estimated total poppy area.

```

X <- cbind(grdKandahar$x, grdKandahar$y)
Xsample <- X[units,]
sqrt(vsb(pi[units], myGRTSsample$poppy, Xsample)) %>% round(.,0)

```

```
[1] 12887
```

9.3 Balanced sampling with spreading

As mentioned in the introduction to this chapter a sample balanced on a covariate still may have a poor spreading along the axis spanned by the covariate. ? presented a method for selecting balanced samples that are also well-spread in the space spanned by the covariates, which they refer to as doubly-balanced sampling. If we take one or more covariates as balancing variables, and besides Easting and Northing as spreading variables, this leads to balanced samples with good *geographical* spreading. When the residuals of the regression model show spatial structure (are spatially correlated), the estimated population mean of the study variable becomes more precise thanks to the improved geographical spreading. Balanced samples with spreading can be selected with function `lcube` of package **BalancedSampling** (?). This is illustrated with Eastern Amazonia, using as before `lnSWIR2` for balancing the sample.

```
library(BalancedSampling)
N <- nrow(gridAmazonia)
n <- 100
Xbal <- cbind(rep(1,times=N), gridAmazonia$lnSWIR2)
Xspread <- cbind(gridAmazonia$x1, gridAmazonia$x2)
pi <- rep(n/N, times=N)
set.seed(314)
units <- lcube(Xbal=Xbal, Xspread=Xspread, prob=pi)
mysample <- gridAmazonia[units,]
```

Comparing this sample with the balanced sample in Figure 9.3 shows that the geographical spreading of the sample is improved, although there still are some close points. The π estimate of the mean is 225.61.

The variance of the estimator of the mean can be estimated by (equation 7, ?)

$$\widehat{V}(\widehat{z}) = \frac{n}{n-p} \frac{p}{p+1} \sum_{k \in S} (1 - \pi_k) \left(\frac{\epsilon_k}{\pi_k} - \bar{\epsilon}_k \right)^2, \quad (9.10)$$

with p the number of balancing variables, ϵ_k the regression model residual of unit k (Equation (9.5)), and $\bar{\epsilon}_k$ the local mean of the residuals of this unit, computed by

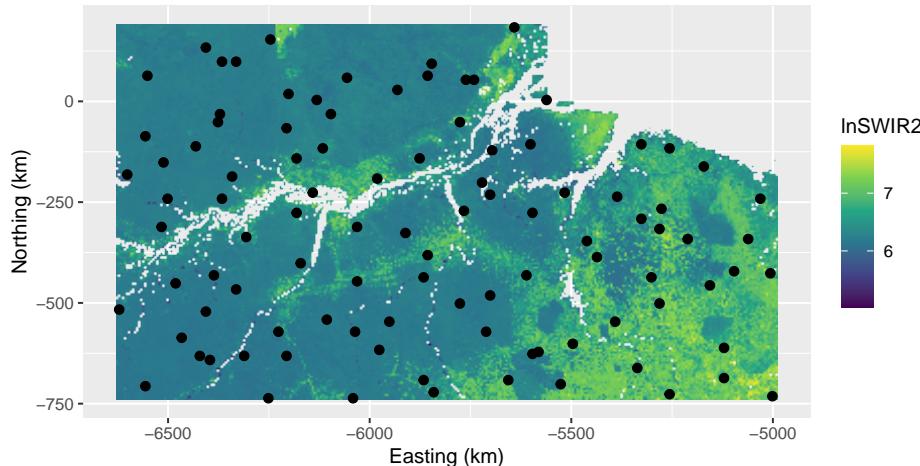


Figure 9.13: Balanced sample, balanced on lnSWIR2, with geographical spreading from Eastern Amazonia (Brazil).

$$\bar{\epsilon}_k = \frac{\sum_{j=1}^{p+1} (1 - \pi_j) \frac{\epsilon_j}{\pi_j}}{\sum_{j=1}^{p+1} (1 - \pi_j)} . \quad (9.11)$$

This variance estimator is easy to compute with functions `localmean.weight` and `localmean.var` of package `spsurvey` (?).

```
library(spsurvey)
pi <- rep(n/N,n)
c <- (1-pi)
b <- estimate_b(z=mysample$AGB/pi, X=Xbal[,]/pi, c=c)
zpred <- Xbal%*%b
e <- mysample$AGB-zpred[,]
weights <- localmean.weight(
  x=mysample$x1, y=mysample$x2, prb=rep(pi,n), nbh=3)
v_mz <- localmean.var(z=e/pi, weight.lst=weights)/N^2
```

The estimated standard error is 2.8, which is considerably smaller than the estimated standard error of the balanced sample without geographical spreading.

Chapter 10

Model-assisted estimation

In many cases ancillary information is available that could be useful to increase the accuracy of the estimated mean or total of the study variable. The ancillary variable(s) can be qualitative (i.e. classifications) or quantitative. As we have seen before, both types of ancillary variable can be used at the design stage, i.e. in selecting the sampling units, to improve the performance of the sampling strategy, for instance by stratification (Chapter 4), selecting sampling units with probabilities proportional to size (Chapter 8), or through balancing and/or spreading the sample on the covariates (Chapter 9). In this chapter I explain how these covariates can be used at the stage of *estimation*, once the data are collected.

In the design-based approach for sampling various estimators are developed that exploit one or more covariates. These estimators are derived from different superpopulation model of the study variable. A superpopulation model is a statistical model that can be used to generate an infinite number of populations, a superpopulation, through simulation. An example is the simulation of spatial populations using a geostatistical model, through sequential Gaussian simulation, see Chapter 13. A superpopulation is a construct, it does not exist in reality. We assume that the population of interest is one of the populations that can be generated with the chosen model. The combination of probability sampling and estimators that are build on a superpopulation model, is referred to as the model-assisted approach. Also in the model-based approach a superpopulation model is used, however, its role is fundamentally different from that

in the model-assisted approach, see Chapter 27. To stress the different use of the superpopulation model in the model-assisted approach this model is referred to as the “working model”, i.e. the superpopulation model that is used to derive a model-assisted estimator.

? present an overview of model-assisted estimators derived from a general working model:

$$z_k = \mu(\mathbf{x}_k) + \epsilon_k, \quad (10.1)$$

with $\mu(\mathbf{x}_k)$ the model mean for population unit k which is a function of the covariate values of that unit collected in vector $\mathbf{x}_k = (1, x_{1,k}, \dots, x_{J,k})^T$, and ϵ_k a random variable with zero mean. The model mean $\mu(\mathbf{x}_k)$ can be a linear or a non-linear combination of the covariates. If the study variable and the covariate values were observed for all population units, all these data can be used to compute a so-called hypothetical population fit of the model parameters. These model parameters can then be used to compute *estimates* of the model means $\mu(\mathbf{x}_k)$, denoted by $m(\mathbf{x}_k)$, for all population units. For instance, with a multiple regression model $m(\mathbf{x}_k) = \mathbf{x}_k^T \mathbf{b}$, with \mathbf{b} the vector with regression coefficients estimated from observations of the study variable z and the covariates on *all* population units. In practice we have a sample only, which is used to estimate $m(\mathbf{x}_k)$ by $\hat{m}(\mathbf{x}_k)$. For the multiple regression model $\hat{m}(\mathbf{x}_k) = \mathbf{x}_k^T \hat{\mathbf{b}}$, with $\hat{\mathbf{b}}$ the vector with regression coefficients estimated from the sample data. This leads to the generalised difference estimator (?):

$$\hat{\bar{z}}_{\text{dif}} = \frac{1}{N} \sum_{k=1}^N \hat{m}(\mathbf{x}_k) + \frac{1}{N} \sum_{k \in S} \frac{z_k - \hat{m}(\mathbf{x}_k)}{\pi_k}, \quad (10.2)$$

with π_k the inclusion probability of unit k . The first term is the population mean of model predictions of the study variable, the second term is the π estimator of the population mean of the residuals.

A wide variety of model-assisted estimators have been developed and tested in the past decades. They differ in the working model used to obtain the estimates $\hat{m}(\mathbf{x}_k)$ in Equation (10.2). The best known class of model-assisted estimators is the generalised regression estimator that uses a linear model in prediction (?). Alternative model-assisted estimators are the estimators using machine learning techniques for prediction. In the era of big data with a vastly increasing number

of exhaustive data sets, and a rapid development of machine learning techniques, these estimators have great potentials for spatial sample survey.

10.1 Generalized regression estimator

The working model of the generalised regression estimator is the heteroscedastic multiple linear regression model:

$$Z_k = \mathbf{x}_k^T \boldsymbol{\beta} + \epsilon_k , \quad (10.3)$$

with ϵ_k uncorrelated residuals, with zero mean and variance $\sigma^2(\epsilon_k)$. Note that I use uppercase Z to distinguish the random variable Z_k of unit k from one realization of this random variable for unit k in the population of interest, z_k . Further note that the variance of the residuals $\sigma^2(\epsilon_k)$ need not be constant but may differ among the population units. If $\{z_k, x_{1,k}, \dots, x_{J,k}\}$ were observed for all units $k = 1, \dots, N$ in the population, the regression coefficients $\boldsymbol{\beta}$ would be estimated by

$$\mathbf{b} = \left(\sum_{k=1}^N \frac{\mathbf{x}_k \mathbf{x}_k^T}{\sigma^2(\epsilon_k)} \right)^{-1} \sum_{k=1}^N \frac{\mathbf{x}_k z_k}{\sigma^2(\epsilon_k)} , \quad (10.4)$$

with \mathbf{x}_k the vector $(1, x_{1,k}, \dots, x_{J,k})^T$, and $\sigma^2(\epsilon_k)$ the variance of the residual of unit k . So, similar to the distinction between model mean and population mean (see Chapter 27), here the model regression coefficients $\boldsymbol{\beta}$ are distinguished from the population regression coefficients \mathbf{b} . The means $m(\mathbf{x}_k)$ would then be computed by

$$m(\mathbf{x}_k) = \mathbf{x}_k^T \mathbf{b} . \quad (10.5)$$

If we have a probability sample from the population of interest, \mathbf{b} is estimated by replacing the population totals in Equation (10.4) by their π estimators:

$$\hat{\mathbf{b}} = \left(\sum_{k \in S} \frac{\mathbf{x}_k \mathbf{x}_k^T}{\sigma^2(\epsilon_k) \pi_k} \right)^{-1} \sum_{k \in S} \frac{\mathbf{x}_k z_k}{\sigma^2(\epsilon_k) \pi_k} , \quad (10.6)$$

Note that with unequal inclusion probabilities, the design-based estimators of the population regression coefficients differ from the usual ordinary least squares (OLS) estimators of the regression coefficients defined as model parameters. The values \hat{b}_j are estimates of the *population parameters* b_j .

The mean values $m(\mathbf{x}_k)$ are now estimated by

$$\hat{m}(\mathbf{x}_k) = \mathbf{x}_k^T \hat{\mathbf{b}} . \quad (10.7)$$

Plugging Equation (10.7) into the generalised difference estimator, Equation (10.2), leads to the generalised regression estimator for the population mean:

$$\hat{\bar{z}}_{\text{regr}} = \frac{1}{N} \sum_{k=1}^N \mathbf{x}_k^T \hat{\mathbf{b}} + \frac{1}{N} \sum_{k \in \mathcal{S}} \frac{z_k - \mathbf{x}_k^T \hat{\mathbf{b}}}{\pi_k} . \quad (10.8)$$

This estimator can also be written as

$$\hat{\bar{z}}_{\text{regr}} = \hat{\bar{z}}_\pi + \sum_{j=1}^J \hat{b}_j (\bar{x}_j - \hat{\bar{x}}_{j,\pi}) , \quad (10.9)$$

with $\hat{\bar{z}}_\pi$ and $\hat{\bar{x}}_{j,\pi}$ the π estimator of the study variable z and the j th covariate, respectively, \bar{x}_j the population mean of the j th covariate, and \hat{b}_j the estimated slope coefficient associated with the j th covariate. So the generalised regression estimate is equal to the π estimate when the estimated means of the covariates are equal to the population means. This is the rationale of balanced sampling (Chapter 9).

The alternative formulation of the regression estimator (Equation (10.9)) shows that we do not need to know the covariate values for all population units. Knowledge of the population means of the covariates is sufficient. This is because a linear relation is assumed between the study variable and the covariates. On the contrary, for non-linear working models such as a random forest model, exhaustive knowledge of the covariates is needed so that the estimated mean $\hat{m}(\mathbf{x}_k)$ in Equation (10.2) can be computed for every unit in the population.

? worked out the generalised regression estimator for various superpopulation models, such as the simple and multiple linear regression model, the ratio model and the ANOVA model.

10.1.1 Simple and multiple regression estimators

The working model of the simple and multiple regression estimator is the homoscedastic linear regression model. The only difference with the heteroscedastic model (Equation (10.3)) is that the variance of the residuals is assumed constant: $\sigma^2(\epsilon_k) = \sigma^2(\epsilon), k = 1, \dots, N$.

In the simple linear regression model the mean is a linear combination of a single covariate, $\mu(x_k) = \alpha + \beta x_k$. The simple linear regression model leads to the simple regression estimator. With simple random sampling this estimator for the population mean is

$$\hat{\bar{z}}_{\text{regr}} = \bar{z}_{\mathcal{S}} + \hat{b} (\bar{x} - \bar{x}_{\mathcal{S}}) , \quad (10.10)$$

where $\bar{z}_{\mathcal{S}}$ and $\bar{x}_{\mathcal{S}}$ are the sample means of the study variable and the covariate, respectively, \bar{x} is the population mean of the covariate, and \hat{b} is the estimated slope coefficient:

$$\hat{b} = \frac{\sum_{k \in \mathcal{S}} (x_k - \bar{x}_{\mathcal{S}})(z_k - \bar{z}_{\mathcal{S}})}{\sum_{k \in \mathcal{S}} (x_k - \bar{x}_{\mathcal{S}})^2} . \quad (10.11)$$

The rationale of the regression estimator is that when the estimated mean of the covariate is, for instance, smaller than the population mean of the covariate, then with a positive correlation between the study variable and covariate, also the estimated mean of the study variable is expected to be smaller than the population mean of the study variable. The difference between the population mean and estimated mean of the covariate can be used to improve the π estimate of the mean of z (which is for simple random sampling equal to the sample mean $\bar{z}_{\mathcal{S}}$), by adding a term proportional to the difference between the estimated mean and population mean of the covariate. As a scaling factor the estimated slope of the fitted regression line is used.

The sampling variance of this regression estimator can be estimated by computing first the regression residuals $e_k = z_k - \hat{z}_k, i = 1 \dots n$ at the sampling points. To compute these residuals we also need an estimate of the intercept. With simple random sampling this intercept can be estimated by

$$\hat{a} = \bar{z}_{\mathcal{S}} - \hat{b} \bar{x}_{\mathcal{S}} . \quad (10.12)$$

The sampling variance of the regression estimator is *approximately* equal to the sampling variance of the estimator of the mean of these residuals:

$$\widehat{V}\left(\widehat{\bar{z}}_{\text{regr}}\right) = \frac{\widehat{S^2}(e)}{n}, \quad (10.13)$$

with $\widehat{S^2}(e)$ the estimated population variance of the regression residuals

$$\widehat{S^2}(e) = \frac{1}{n-1} \sum_{k \in \mathcal{S}} e_k^2. \quad (10.14)$$

The variance estimator is an approximation because the regression coefficients are also estimated from the sample, which makes the regression estimator non-linear. The approximation of the variance is based on a Taylor linearisation of the regression estimator (? , p. 235). For simple random sampling without replacement from finite populations, the variance estimator must be multiplied by the finite population correction factor $1 - n/N$, see Chapter 3.

In the multiple linear regression model the mean is a linear function of multiple covariates. This model leads to the multiple regression estimator . With simple random sampling the population regression coefficients of this estimator can be estimated by

$$\hat{\mathbf{b}} = \left(\sum_{k \in \mathcal{S}} \mathbf{x}_k \mathbf{x}_k^T \right)^{-1} \sum_{k \in \mathcal{S}} \mathbf{x}_k z_k. \quad (10.15)$$

Comparison with the general regression estimator of the population regression coefficients (Equation (10.6)) shows that both the variance of the residuals, $\sigma^2(\epsilon_k)$, and the inclusion probabilities π_k are missing, as they are (assumed) constant.

The simple regression estimator is illustrated with Eastern Amazonia, see Section 1.3. The population mean of the aboveground biomass (AGB) is estimated by the simple regression estimator, using natural logarithms of MODIS short-wave infrared radiation (SWIR2) as a covariate.

The correlation coefficient equals -0.827. The slope of the fitted line equals -228.1. Now a simple random sample without replacement of 100 units is se-

lected, and the two population regression coefficients are estimated with Equation (10.15).

```
N <- nrow(gridAmazonia)
n <- 100
set.seed(321)
units <- sample.int(nrow(gridAmazonia), size=n, replace=FALSE)
mysample <- gridAmazonia[units,c("AGB","lnSWIR2")]
X <- matrix(nrow=n,ncol=2,data=1)
X[,2] <- mysample$lnSWIR2
XX <- t(X) %*% X
XX_inv <- chol2inv(chol(XX))
Xz <- t(X) %*% mysample$AGB
print(ab <- t(XX_inv %*% Xz))

[,1]      [,2]
[1,] 1751.636 -237.1379
```

The same estimates are obtained by ordinary least squares (OLS) fitting of the model with function `lm`.

```
lm_sample <- lm(AGB~lnSWIR2, data=mysample)
print(ab.mb <- coef(lm_sample))

(Intercept)      lnSWIR2
1751.6363     -237.1379
```

But care must be taken: the design-based estimates of the population regression coefficients are only equal to these model-based OLS estimates of the regression coefficients for equal probability sampling designs. Also be aware that the variance of the design-based estimates of the population regression coefficients is not equal to the model-based variance of the model regression coefficients. See Section 11.2.2.1 in ? for how to estimate the variance of the design-based estimates of the population regression coefficients.

Figure 10.1 shows the scatter plot for the sample and the fitted simple linear regression model.

The simple random sample is used to estimate the population mean of the study variable AGB by the simple regression estimator, and to approximate the

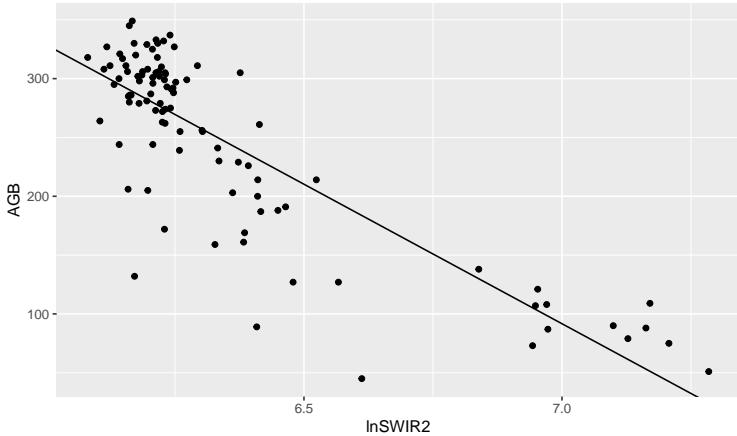


Figure 10.1: Scatterplot and fitted simple linear regression model for above-ground biomass (AGB), using lnSWIR2 as predictor, for simple random sample.

sampling variance of the regression estimator. The residuals of the fitted model can be extracted with function `residuals`, because in this case the model-based estimates of the regression coefficients are equal to the design-based estimates.

```
mx_pop <- mean(gridAmazonia$lnSWIR2)
mx_sam <- mean(mysample$lnSWIR2)
mz_sam <- mean(mysample$AGB)
mz_regr <- mz_sam+ab[2]*
  (mx_pop-mx_sam)
e <- residuals(lm_sample)
S2e <- var(e)
N <- nrow(gridAmazonia)
se_mz_regr <- sqrt((1-n/N)*S2e/n)
```

The difference $\delta(x)$ between the population mean of the covariate lnSWIR2 (6.415) and its estimated mean (6.347) equals 0.068. We may expect the difference between the unknown population mean of the study variable AGB and its sample mean (246.51) to be equal to $\delta(x)$, multiplied by the estimated slope of the line, which equals -237.1. The result (-16.1039) is added to the simple random sample estimate.

The estimated approximate standard error of the regression estimator equals 4.458. The approximated variance is a simplification of a more complicated approximation derived from writing the regression estimator of the population total as a weighted sum of the observations divided by the inclusion probabilities, see Equation (6.5.9) in ?. The alternative expression for the regression estimator of the population mean is

$$\hat{\bar{z}}_{\text{regr}} = \frac{1}{N} \sum_{k \in \mathcal{S}} g_k \frac{z_k}{\pi_k}, \quad (10.16)$$

with g_k the weight attached to the observation on sampling unit k . For simple random sampling the weights are equal to (Equation 6.5.12 in ?)

$$g_k = 1 + \frac{(\bar{x} - \bar{x}_{\mathcal{S}})(x_k - \bar{x}_{\mathcal{S}})}{\widehat{S^2}(x)}. \quad (10.17)$$

```
S2x <- sum((mysample$lnSWIR2-mean(mysample$lnSWIR2))^2)/n
g <- 1+((mx_pop-mx_sam)*(mysample$lnSWIR2-mx_sam))/S2x
```

The sample mean of the weights equals one,

```
mean(g)
```

```
[1] 1
```

and the sample mean of the product of the weights and the covariate x equals the population mean of the covariate.

```
all.equal(mean(g*mysample$lnSWIR2),mean(gridAmazonia$lnSWIR2))
```

```
[1] TRUE
```

In other words, the sample is calibrated on the known population means. The variance of the regression estimator of the population mean can then be approximated by (Section 6.6 in ?)

$$\widehat{V}(\hat{\bar{z}}_{\text{regr}}) = \left(1 - \frac{n}{N}\right) \frac{\sum_{k \in S} g_k^2 e_k^2}{n(n-1)}. \quad (10.18)$$

Comparing this with Equation (10.13) shows that in the first approximation we assumed that all weights are equal to one.

```
S2ge <- sum(g^2*e^2)/(n-1)
(se_mz_regr <- sqrt((1-n/N)*S2ge/n))
```

```
[1] 4.546553
```

The regression estimator and its standard error can be computed with package **survey**. After specifying the sampling design with function **svydesign**, function **calibrate** is used to calibrate the sample on the known population totals N and $t(x) = \sum_{k=1}^N x_k$, with x_k the value of covariate lnSWIR2 for unit k .

```
library(survey)
mysample$fpc <- N
design_si <- svydesign(id=~1, data=mysample, fpc=~fpc)
populationtotals <- c(N,sum(gridAmazonia$lnSWIR2))
mysample_cal <- calibrate(
  design_si, formula=~lnSWIR2, population=populationtotals,
  calfun="linear")
```

The calibrated weights can be extracted with function **weights**. These weights are divided by the inclusion probabilities $\pi = n/N$, so that the sample sum of the weights equals N , and not the sample size n (as in the code chunk above),

```
g <- weights(mysample_cal)
all.equal(sum(g),N)
```

```
[1] TRUE
```

and the sample sum of the product of the weights and the covariate equals the population total of the covariate:

```
all.equal(sum(g*mysample$lnSWIR2),sum(gridAmazonia$lnSWIR2))
```

```
[1] TRUE
```

Finally, the population mean can be estimated with function `svymean`. This is simply the sample sum of the product of the weights and the study variable AGB, divided by N .

```
svymean(~AGB, mysample_cal)
```

mean	SE
AGB 230.41	4.5466

Figure 10.2 shows the sampling distribution of the simple regression estimator along with the distribution of the π estimator, obtained by repeating simple random sampling of 100 units and estimation 10,000 times.

The average of the 10,000 regression estimates equals 224.702. The population mean of the study variable AGB equals 225.048, so the estimated bias of the regression estimator equals -0.347. The variance of the 10,000 regression estimates equals 26.936, and the average of the 10,000 estimated approximate variances equals 27.105. The gain in precision due to the regression estimator, quantified by the ratio of the variance of the π estimator to the variance of the regression estimator equals 3.163.

Using multiple covariates in the regression estimator is straightforward with the function `calibrate`. As a first step the best model is selected with function `regsubsets` of package `leaps` (?).

```
library(leaps)
n <- 100
set.seed(321)
units <- sample.int(nrow(gridAmazonia), size=n, replace=FALSE)
covars <- c("AGB","lnSWIR2","Terra_PP","Prec_dm",
           "Elevation","Clay")
mysample <- gridAmazonia[units,covars]
models <- regsubsets(AGB~., data=mysample, nvmax=4)
```

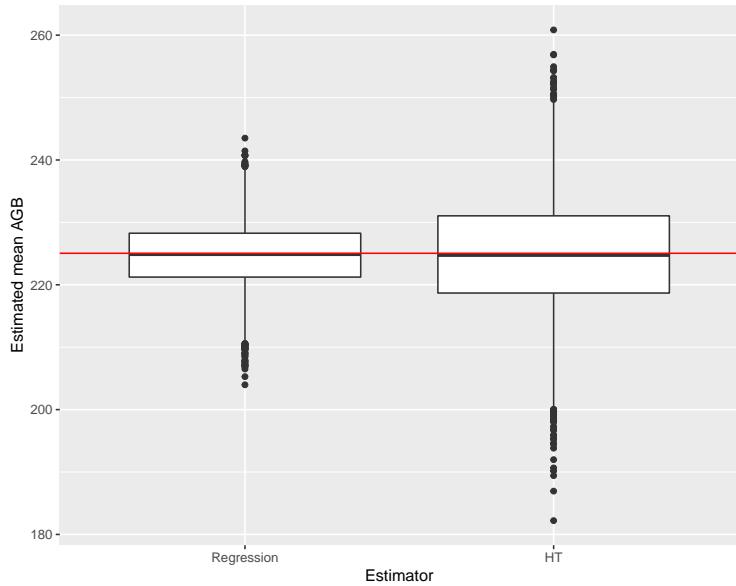


Figure 10.2: Sampling distribution of simple regression estimator and π estimator of mean aboveground biomass (AGB) in Eastern Amazonia, for simple random sampling without replacement of size 100.

```
res.sum <- summary(models)
res.sum$outmat
```

	lnSWIR2	Terra_PP	Prec_dm	Elevation	Clay
1	(1)	"*"	" "	" "	" "
2	(1)	"*"	"*"	" "	" "
3	(1)	"*"	"*"	" "	"*"
4	(1)	"*"	"*"	"*"	" "

The best model with one predictor is the model with lnSWIR2, the best model with two predictors is the one with lnSWIR2 and Terra_PP, etc. Of these models the third model, i.e. the model with lnSWIR2, Terra_PP and Elevation is the best when using adjusted R^2 as a selection criterion.

```
which.max(res.sum$adjr2)
```

```
[1] 3
```

The standard error of the estimated mean of AGB is somewhat reduced by adding the covariates Terra_PP and Elevation to the regression estimator.

```
mysample$fpc <- nrow(gridAmazonia)
design_si <- svydesign(id=~1, data=mysample, fpc=~fpc)
totals <- c(nrow(gridAmazonia),
            sum(gridAmazonia$lnSWIR2),
            sum(gridAmazonia$Terra_PP),
            sum(gridAmazonia$Elevation))
mysample_cal <- calibrate(
  design_si, formula=~lnSWIR2+Terra_PP+Elevation,
  population=totals, calfun="linear")
svymean(~AGB, mysample_cal)

      mean      SE
AGB 230.54 4.2224
```

Another interesting package for model-assisted estimation is package **mase** (?). The regression estimate can be computed with function **greg**.

```
library(mase)
covars <- c("lnSWIR2", "Terra_PP", "Elevation")
res <- greg(
  y=mysample$AGB, xsample=mysample[covars],
  xpop=gridAmazonia[covars], pi=rep(n/N,n),
  var_est=TRUE, var_method="LinHTSRS", model="linear")
res$pop_mean
```

```
[1] 230.5407
```

The multiple regression estimate is equal to the estimate obtained with function **calibrate** of package **survey**. The estimated standard error equals

```
sqrt(res$pop_mean_var)
```

```
[,1]
[1,] 4.207809
```

which is slightly smaller than the standard error computed with package **survey**. This standard error is computed by ignoring the g-weights (?).

```
mlr <- lm(AGB~lnSWIR2+Terra_PP+Elevation, data=mysample)
e <- residuals(mlr)
S2e <- var(e)
print(se <- sqrt((1-n/N)*S2e/n))
```

10.1.2 Penalised least squares estimation

In the previous section I first selected a best subset of covariates before using these covariates in estimating the population regression coefficients. The alternative is to skip the selection of the best model, and to estimate the population regression coefficients of *all* covariates by penalised least squares estimation (PLS). In PLS a penalty equal to the sum of the absolute or squared values of the population regression coefficients is added to the minimisation criterion, see ? for details. PLS is implemented in function **gregElasticNet** of package **mase**.

```
covars <- c("lnSWIR2", "Terra_PP", "Prec_dm", "Elevation", "Clay")
res <- gregElasticNet(
  y=mysample$AGB, xsample=mysample[covars],
  xpop=gridAmazonia[covars], pi=rep(n/N,n),
  var_est=TRUE, var_method="LinHTSRS", model="linear",
  lambda="lambda.min", cvfolds=100)
signif(res$coefficients,4)

(Intercept)      lnSWIR2      Terra_PP      Prec_dm      Elevation      Clay
-71.07000     -2.83800      0.02204      0.52710     -0.03507      0.13630
```

All five covariates are used in prediction, but the coefficients associated with these predictors are small except for lnSWIR2.

The estimated standard error is considerably larger than the standard error obtained with lnSWIR2, Terra_PP and Elevation as predictors, so in this case the elastic net regression estimator works not as well as the multiple regression estimator using the best subset of the covariates.

```
sqrt(res$pop_mean_var)
```

```
s1  
s1 6.207637
```

Exercises

1. The data for this Exercise are in `data/Amazonia_1km.RData`. Write an **R** script to
 - Compute the sampling variance of the simple regression estimator of the mean of AGB, using lnSWIR2 as a covariate, for simple random sampling and sample sizes of 10, 25 and 100, assuming that the population regression coefficients are perfectly known. Hint: fit a simple linear regression model on all data, and compute the population variance of the residuals.
 - Select 10,000 times a simple random sample with replacement of size 10 (use a for-loop). Use each sample to estimate the population mean of AGB with the simple regression estimator (using sample estimates of the population regression coefficients), and estimate the approximate variance of the estimator of the mean. Compute the variance of the 10,000 regression estimates and the average of the 10,000 approximate variance estimates. Repeat this for sample sizes 25, 100.
 - Compute for each sample size the difference between the experimental variance (variance of the 10,000 regression estimates) and the variance obtained with the population fit of the regression model as a proportion of the experimental variance. Explain what you see.
 - Compute for each sample size the difference between the average of the 10,000 approximated variances and the experimental variance, as a proportion of the experimental variance. Explain what you see.

10.1.3 Regression estimator with stratified simple random sampling

With stratified simple random sampling there are two regression estimators, the *separate* and the *combined* regression estimator. In the first estimator the regression estimator for simple random sampling is applied at the level of the strata. This implies that for each stratum separately a vector with population regression coefficients \mathbf{b}_h is estimated. The regression estimates of the stratum means are then combined by computing the weighted average, using the relative sizes of the strata as weights:

$$\hat{\bar{z}}_{\text{sregr}} = \sum_{h=1}^H w_h \hat{\bar{z}}_{\text{regr},h} , \quad (10.19)$$

with, for the simple linear estimator

$$\hat{\bar{z}}_{\text{regr},h} = \bar{z}_{sh} + \hat{b}_h (\bar{x}_h - \bar{x}_{sh}) , \quad (10.20)$$

with \bar{z}_{sh} and \bar{x}_{sh} the stratum sample means of the study variable and covariate, respectively, \bar{x}_h the mean of the covariate in stratum h , and \hat{b}_h the estimated slope coefficient for stratum h .

The variance of this separate regression estimator of the population mean can be estimated by first estimating the variances of the regression estimators of the stratum means using Equation (10.13), and then combining these variances using Equation (4.4).

The separate regression estimator is illustrated with Eastern Amazonia. Biomes are used as strata. There are four biomes, the levels of which are given short names using function `levels`.

```
library(sampling)
gridAmazonia$Biome <- as.factor(gridAmazonia$Biome)
levels(gridAmazonia$Biome)

[1] "Mangroves"
[2] "Tropical & Subtropical Dry Broadleaf Forests"
[3] "Tropical & Subtropical Grasslands, Savannas & Shrublands"
```

```
[4] "Tropical & Subtropical Moist Broadleaf Forests"
```

```
biomes <- c("Mangrove", "Forest.dry", "Grassland", "Forest.moist")
levels(gridAmazonia$Biome) <- biomes
```

Moist forest is by far the largest stratum, it covers 92% of the area. Mangrove, dry forest and grassland cover 0.4%, 2.3% and 5.5% of the area, respectively. A stratified simple random sample of size 100 is selected using function **strata** of package **sampling** (?), see Chapter 4. I chose five units as a minimum sample size. Note that the stratum sample sizes are not proportional to their size.

```
ord <- unique(gridAmazonia$Biome)
N_h <- table(gridAmazonia$Biome)
n_h <- c(5, 5, 5, 85)
set.seed(314)
units <- sampling::strata(
  gridAmazonia, stratanames="Biome", size=n_h[ord],
  method="srswor")
mysample <- getdata(gridAmazonia, units)
```

As a first step in estimation, for each stratum the mean of the covariate over all units in a stratum (population mean per stratum) and the sample means of the study variable and covariate are computed.

```
mx_h_pop <- tapply(
  gridAmazonia$lnSWIR2, INDEX=gridAmazonia$Biome, FUN=mean)
mzh_sam <- tapply(
  mysample$AGB, INDEX=mysample$Biome, FUN=mean)
mx_h_sam <- tapply(
  mysample$lnSWIR2, INDEX=mysample$Biome, FUN=mean)
```

The next step is to estimate the regression coefficients per stratum. This is done in a for-loop. The estimated slope coefficient is used to compute the regression estimator per stratum. The residuals are extracted to approximate the variance of the regression estimator per stratum.

```

b_h <- mz_h_regr <- v_mz_h_regr <- numeric(length=4)
for (i in 1:4) {
  subsam <- subset(
    mysample, mysample$Biome==levels(gridAmazonia$Biome)[i])
  lm_sample <- lm(AGB~lnSWIR2, data=subsam)
  b_h[i] <- coef(lm_sample)[2]
  mz_h_regr[i] <- mzh_sam[i]+b_h[i]*
    (mx_h_pop[i]-mx_h_sam[i])
  e <- residuals(lm_sample)
  S2e_h <- var(e)
  v_mz_h_regr[i] <- (1-n_h[i]/N_h[i])*S2e_h/n_h[i]
}

```

Finally, the separate regression estimate is computed as a weighted average of the regression estimates per stratum,

```
w_h <- N_h/sum(N_h)
print(mz_sepreg <- sum(w_h*mz_h_regr))
```

```
[1] 223.9426
```

and its standard error by the square root of the pooled variances of the regression estimator per stratum, using the squared relative size of the strata as weights.

```
sum(w_h^2*v_mz_h_regr) %>% sqrt(.)
```

```
[1] 5.077558
```

The separate regression estimator can be computed with package **survey**. The computation of the population totals merits special attention. For the regression estimator with *simple* random sampling these totals are the total number of populations units and the population total of the covariate lnSWIR2. These are the population totals associated with the columns of the design matrix that is constructed with function **lm** to estimate the regression coefficients. The column with ones results in an estimated intercept, the column with lnSWIR2 values in an estimated slope.

The model that is fitted now is an ANCOVA model with the factor Biome and

covariate lnSWIR2.

```
ancova <- lm(AGB~Biome*lnSWIR2, data=mysample)
```

R uses the so-called cornerstone representation of the ANCOVA model. The reference level is stratum Mangrove. The question is what population totals must be assigned to the function `calibrate` with this ANCOVA model. To make this clear, let us have a look at the columns of the design matrix, used to fit the ANCOVA model. Only the first five rows are printed.

```
designmat <- model.matrix(ancova, mysample)
```

	(Intercept)	BiomeForest.dry	BiomeGrassland	BiomeForest.moist	lnSWIR2
[1,]	1	0	0	0	6.307024
[2,]	1	0	0	0	6.236278
[3,]	1	0	0	0	6.232421
[4,]	1	0	0	0	6.249708
[5,]	1	0	0	0	6.224987
	BiomeForest.dry:lnSWIR2	BiomeGrassland:lnSWIR2	BiomeForest.moist:lnSWIR2		
[1,]	0	0	0	6.307024	
[2,]	0	0	0	6.236278	
[3,]	0	0	0	6.232421	
[4,]	0	0	0	6.249708	
[5,]	0	0	0	6.224987	

With this model formulation the first population total is the total number of population units. The second, third and fourth population totals are the number of population units in stratum levels 2, 3 and 4. The fifth population total is the population total of covariate lnSWIR2, and the sixth, seventh and eighth population totals are the totals of covariate lnSWIR2 in stratum levels 2, 3 and 4. Note that the line `names(totals) <- names(coef(ancova))` is not strictly needed. This is just to suppress a warning that the names of the numeric with the population totals does not match the names of the columns of the design matrix. As a consequence we do not need to fit the ANCOVA model, as shown in the code chunk above, either.

```

N_h <- as.numeric(N_h)
lut <- data.frame(Biome=biomes,N_h)
mysample <- merge(x=mysample, y=lut)
design_stsi <- svydesign(id=~1, strata=~factor(Biome),
                         data=mysample, fpc=~N_h)
tx_pop <- sum(gridAmazonia$lnSWIR2)
tx_h_pop <- N_h*mx_h_pop
totals <- c(sum(N_h),N_h[c(2,3,4)],tx_pop,tx_h_pop[c(2,3,4)])
names(totals) <- names(coef(ancova))
mysample_cal <- calibrate(
  design_stsi, formula=~Biome*lnSWIR2, population=totals,
  calfun="linear")
svymean(~AGB, mysample_cal)

      mean      SE
AGB 223.94 5.8686

```

Alternatively, we may use the following formula in function `lm`.

```

ancova2 <- lm(AGB~0+Biome/lnSWIR2, data=mysample)
designmat <- model.matrix(ancova, mysample)

      (Intercept) BiomeForest.dry BiomeGrassland BiomeForest.moist  lnSWIR2
[1,]          1              0              0          1 6.307024
[2,]          1              0              0          1 6.236278
[3,]          1              0              0          1 6.232421
[4,]          1              0              0          1 6.249708
[5,]          1              0              0          1 6.224987
      BiomeForest.dry:lnSWIR2 BiomeGrassland:lnSWIR2 BiomeForest.moist:lnSWIR2
[1,]              0              0          6.307024
[2,]              0              0          6.236278
[3,]              0              0          6.232421
[4,]              0              0          6.249708
[5,]              0              0          6.224987

```

With this formula the population totals are the number of population units in stratum levels 1, 2, 3 and 4, and the population totals of covariate `lnSWIR2` per stratum.

```

totals <- c(N_h,tx_h_pop)
names(totals) <- names(coef(ancova2))
mysample_cal <- calibrate(
  design_stsi, formula=~0+Biome/lnSWIR2, population=totals,
  calfun="linear")
svymean(~AGB, mysample_cal)

      mean      SE
AGB 223.94 5.8686

```

Recall the alternative formulation of the regression estimator, Equation (10.16), and that the sample sum of the z -expanded values multiplied by the calibrated weights g_k are equal to the population totals. This explains why the population totals should match the columns of the design matrix.

10.1.3.1 Combined regression estimator

The alternative to the separate regression estimator is the combined regression estimator

$$\hat{z}_{\text{cregr}} = \hat{z}_\pi + \hat{b} (\bar{x} - \hat{x}_\pi) , \quad (10.21)$$

with \hat{b} the estimated slope coefficient, estimated by Equation (10.6), discarding the variance of the residuals $\sigma^2(\epsilon_k)$ as they are assumed constant, and using the appropriate inclusion probabilities which differ among the strata, and \hat{z}_π and \hat{x}_π the π estimators of the population mean of the study variable and the covariate with stratified simple random sampling, respectively.

In the combined regression estimator only one regression coefficient b is estimated, the slope coefficient for the entire population. This combined regression estimator is recommended when the stratum sample sizes are small as in our case, so that the estimated regression coefficients per stratum, \hat{b}_h , become unreliable. The estimators above are for infinite populations and for stratified simple random sampling with replacement of finite populations. For sampling without replacement from finite populations, finite population corrections $1 - n_h/N_h$ must be added to the numerator and denominator of \hat{b} .

The approximate variance of the combined regression estimator can be estimated as follows:

1. Compute residuals: $e_k = z_k - (\hat{a}_\pi + \hat{b}x_k)$.
2. Estimate for each stratum the variance of the estimator of the mean of the residuals: $\widehat{V}(\hat{e}_h) = \widehat{S^2}_h(e)/n_h$, with $\widehat{S^2}_h(e)$ the estimated variance of the residuals in stratum h .
3. Combine the estimated variances per stratum: $\widehat{V}(\hat{z}_{\text{cregr}}) = \sum_{h=1}^H w_h^2 \widehat{V}(\hat{e}_h)$.

The next code chunk shows the estimation procedure. First the population means of the study variable AGB and of the covariate lnSWIR2 are estimated by the π estimator, see Chapter 4.

```
mz_h_HT <- tapply(mysample$AGB, INDEX=mysample$Biome, FUN=mean)
mx_h_HT <- tapply(mysample$lnSWIR2, INDEX=mysample$Biome, FUN=mean)
mz_HT <- sum(w_h*mz_h_HT)
mx_HT <- sum(w_h*mx_h_HT)
```

The next step is to estimate the population regression coefficients, using Equation (10.6) in which the variances $\sigma^2(\epsilon_k)$ can be dropped, as these are assumed constant. The inclusion probabilities are in the column `Prob` of `mysample`.

```
W <- diag(x=1/mysample$Prob, nrow=n, ncol=n)
X <- matrix(nrow=n, ncol=2, data=1)
X[,2] <- mysample$lnSWIR2
XW <- crossprod(X, W)
XWX <- XW %*% X
XWX_inv <- chol2inv(chol(XWX))
XWz <- XW %*% mysample$AGB
print(ab <- t(XWX_inv %*% XWz))

[,1]      [,2]
[1,] 1678.268 -226.6772
```

Note that the same estimates are obtained by model-based estimation, using weighted least squares, based on the assumption of that the variances $\sigma^2(\epsilon_k)$ are proportional to the inclusion probabilities (which is a weird assumption).

```
lm_wls <- lm(AGB~lnSWIR2, weights=1/Prob, data=mysample)
coef(lm_wls)
```

(Intercept)	lnSWIR2
1678.2684	-226.6772

So in model-based estimation the weights differ among the units because of assumed differences in the variance of the residuals, whereas in design-based estimation we assign different weights because the units have different inclusion probabilities (?).

Finally the combined regression estimate is computed.

```
print(mz_combreg <- mz_HT+ab[2]*(mx_pop-mx_HT))
```

```
[1] 224.1433
```

To approximate the variance of the combined regression estimator first the residuals are computed. Then these residuals are used to estimate the spatial variance of the residuals within the strata, $\widehat{S^2}_h(e)$, and the variance of the estimator of the mean of the residuals, $\widehat{V}(\widehat{\bar{e}}_h)$.

```
mysample$e <- mysample$AGB-(ab[1]+ab[2]*mysample$lnSWIR2)
v_me_h <- numeric(length=4)
for (i in 1:4) {
  subsam <- subset(
    mysample, mysample$Biome==levels(gridAmazonia$Biome)[i])
  S2e_h <- var(subsam$e)
  v_me_h[i] <- (1-n_h[i]/N_h[i])*S2e_h/n_h[i]
}
print(sqrt(sum(w_h^2*v_me_h)))
```

```
[1] 5.122518
```

Computing the combined regression estimator with package **survey** goes as follows.

```

design_stsi <- svydesign(
  id=~1, strata=~factor(Biome), data=mysample, fpc=~N_h)
totals <- c(nrow(gridAmazonia),sum(gridAmazonia$lnSWIR2))
mysample_cal <- calibrate(
  design_stsi, formula=~lnSWIR2, population=totals,
  calfun="linear")
svymean(~AGB, mysample_cal)

      mean      SE
AGB 224.14 5.8707

```

Function `calibrate` computes the regression estimate and its standard error with the calibrated weights g_k (Equation 6.5.12 in ?). This explains the difference between the two standard errors. In the next code chunk the standard error of the regression estimator is computed with the calibrated weights g_k .

```

lut <- data.frame(Biome=biomes,pi=n_h/N_h)
mysample <- merge(x=mysample, y=lut)
S2x <- sum(((mysample$lnSWIR2-mx_HT)^2)/mysample$pi) /
  sum(mysample$pi^-1)
g <- 1+(mx_pop-mx_HT)*(mysample$lnSWIR2-mx_HT)/S2x
lm_sample <- lm(AGB~lnSWIR2, weights=1/Prob, data=mysample)
e <- residuals(lm_sample)
ge <- g*e
S2geh <- tapply(ge, INDEX=mysample$Biome,FUN=var)
print(sqrt(sum(w_h^2*(1-n_h/N_h)*S2geh/n_h)))

```

```
[1] 5.870737
```

10.2 Ratio estimator

In some cases it is reasonable to assume that the fitted line passes through the origin. An example is the case study on poppy area in Kandahar (Chapter 8). The covariate is the agricultural area within the $5 \text{ km} \times 5 \text{ km}$ squares that serve as sampling units. It is reasonable to assume that when the covariate equals zero, also the poppy area is zero. So, if we have an estimate of the ratio of the total poppy area in the population to the total agricultural area in the population,

and besides know the total agricultural area in the population, the total poppy area in the population can be estimated by multiplying the estimated ratio with the known population total agricultural area:

$$\hat{t}_{\text{ratio}}(z) = \frac{\hat{t}_\pi(z)}{\hat{t}_\pi(x)} t(x) = \hat{b} t(x), \quad (10.22)$$

with $\hat{t}_\pi(z)$ and $\hat{t}_\pi(x)$ the π estimators of the total of the study variable (poppy area) and ancillary variable (agricultural area), respectively, and $t(x)$ the total of the ancillary variable, which must be known.

The working model of the ratio estimator is a heteroscedastic model, i.e. a model with non-constant variance (see Exercise 2, hereafter):

$$\begin{aligned} Z(x_k) &= \beta x_k + \epsilon_k \\ \sigma^2(\epsilon_k) &= \sigma^2 x_k, \end{aligned} \quad (10.23)$$

with β the slope of the line, and σ^2 a constant (variance of residual for $x_k = 1$).

This is a general estimator that can be used for any sampling design, not only for simple random sampling. For simple random sampling the population means of z and x are estimated by the sample means.

For simple random sampling the sampling variance of the ratio estimator of the population total can be approximated by

$$\widehat{V}(\hat{t}_{\text{ratio}}(z)) = N^2 \frac{\widehat{S^2}(e)}{n}, \quad (10.24)$$

with $\widehat{S^2}(e)$ the estimated variance of the residuals $e_k = z_k - \hat{b}x_k$:

$$\widehat{S^2}(e) = \frac{1}{n-1} \sum_{k \in S} e_k^2. \quad (10.25)$$

For simple random sampling without replacement from finite populations Equation (10.24) must be multiplied by $(1 - \frac{n}{N})$.

The ratio estimator for the total poppy area and the estimator of its variance for a simple random sample without replacement can be estimated as follows.

```

load("data/Kandahar.RData")
n <- 50
N <- nrow(grdKandahar)
units <- sample.int(N, size=n, replace=FALSE)
mysample <- grdKandahar[units,]
b <- mean(mysample$poppy)/mean(mysample$agri)
tx_pop <- sum(grdKandahar$agri)
print(tz_ratio <- b*tx_pop)

[1] 55009.69

e <- mysample$poppy-b*mysample$agri
print(se_tz_ratio <- sqrt(N^2*(1-(n/N))*var(e)/n))

[1] 18847.31

```

A better variance approximation is obtained with Equation (10.18). For the ratio model and simple random sampling the calibrated weights are equal to (? , p. 248)

$$g = \frac{t(x)}{\hat{t}_\pi(x)}, \quad (10.26)$$

with $t(x)$ the population total of the covariate, and $\hat{t}_\pi(x)$ the π estimator of the population total of the covariate.

```

pi <- n/N
tx_HT <- sum(mysample$agri/pi)
g <- tx_pop/tx_HT
S2ge <- sum(g^2*e^2)/(n-1)
print(se_tz_ratio <- sqrt(N^2*(1-n/N)*S2ge/n))

[1] 17149.62

```

Note that to compute S^2_{ge} $n - 1$ is used, as we have now one population regression coefficient only. The ratio estimate and its standard error can be computed with package **survey** as follows.

```
mysample$N <- N
design_si <- svydesign(id=~1, data=mysample, fpc=~N)
b <- svyratio(~poppy, ~agri, design=design_si)
predict(b, total=tx_pop)

$total
      agri
poppy 55009.69

$se
      agri
poppy 17149.62
```

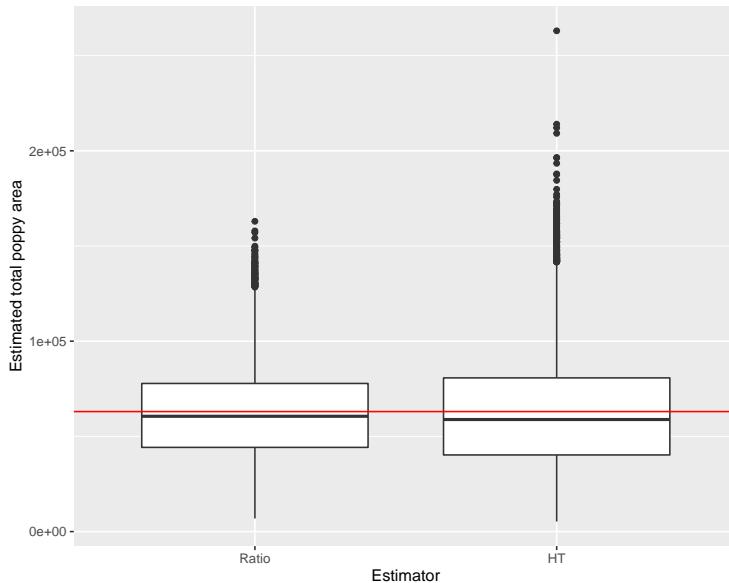


Figure 10.3: Sampling distribution of ratio estimator and π estimator of total poppy area (ha) in Kandahar with simple random sampling of size 50.

Figure 10.3 shows the sampling distribution of the ratio estimator and π estimator, obtained by repeating simple random sampling of size 50 and estimation 10,000 times. The average of the 10,000 ratio estimates of the total poppy area equals 62512. The population total of poppy equals 63038, so the estimated bias of the ratio estimator equals -526. The boxplots in Figure 10.3 show that the ratio estimator has less extreme outliers. The standard deviation of the 10,000 ratio estimates equals 24177. The gain in precision due to the ratio estimator, quantified by the ratio of the variance of the π estimator to the variance of the ratio estimator equals 1.5.

Exercises

2. Write an **R** script to compute the ratio of the population total poppy area and population total agricultural area ($t(z)/t(x)$). Then use all data to fit a linear model without intercept for the poppy area, using the agricultural area as a covariate, assuming that the variance of the residuals is proportional to the agricultural area (heteroscedastic model). Hint: use function `lm` with argument `formula = poppy ~ sim agri-1` and argument `weights=1/agri`. Also fit a model without intercept, assuming a constant variance of the residuals (homoscedastic model). Compare the estimated slopes of the two models with the ratio of the total poppy area and total agricultural area.

10.2.1 Ratio estimators with stratified simple random sampling

With stratified simple random sampling, there are, similar to the regression estimator, two options: either estimate the ratios separately for the strata, or estimate a combined ratio. The separate ratio estimator of the population total is

$$\hat{t}_{\text{sratio}}(z) = \sum_{h=1}^H \hat{t}_{\text{ratio},h}(z) , \quad (10.27)$$

with

$$\hat{t}_{\text{ratio},h}(z) = \frac{\hat{t}_{\pi,h}(z)}{\hat{t}_{\pi,h}(x)} t_h(x) , \quad (10.28)$$

in which $\hat{t}_{\pi,h}(z)$ and $\hat{t}_{\pi,h}(x)$ are the π estimators of the population total of the study variable and the covariate for stratum h , respectively.

The combined ratio estimator is

$$\hat{t}_{\text{cratio}}(z) = \frac{\sum_{h=1}^H \hat{t}_{\pi,h}(z)}{\sum_{h=1}^H \hat{t}_{\pi,h}(x)} t(x). \quad (10.29)$$

The code chunk below shows how the combined and separate regression estimator can be computed with package **survey**. First two equal-sized strata are computed using the median of the covariate **agri** as a stratum break. Stratum sample sizes are computed, and a stratified simple random sample without replacement is selected.

```
breaks <- quantile(grdKandahar$agri, probs=0.5)
grdKandahar$stratum <- findInterval(grdKandahar$agri, breaks)+1
N_h <- table(grdKandahar$stratum)
n_h <- round(n*N_h/sum(N_h))
set.seed(314)
units <- sampling::strata(
  grdKandahar, stratanames="stratum", size=n_h, method="srswor")
mysample <- getdata(grdKandahar, units)
```

The stratum sizes **N_h** are added to **mysample**, the function **svydesign** is to specify the sampling design, function **svyratio** is used to estimate the population ratio and its variance, and finally function **predict** is used to estimate the population total.

```
lut <- data.frame(stratum=c(1,2), N_h)
mysample <- merge(x=mysample, y=lut)
design_stsi <- svydesign(
  id=~1, strata=~stratum, data=mysample, fpc=~Freq)
common <- svyratio(~poppy, ~agri, design_stsi, separate=FALSE)
predict(common, total=sum(grdKandahar$agri))

$total
    agri
```

```
poppy 28389.02
```

```
$se
      agri
poppy 8845.847
```

The same estimate is obtained with function `calibrate`.

```
mysample_cal <- calibrate(
  design_stsi, ~agri-1, population=tx_pop, variance=1)
svytotal(~poppy, mysample_cal)
```

```
      total      SE
poppy 28389 8845.8
```

Computing the separate ratio estimator goes along the same lines. The output object of function `svyratio` contains the estimated ratio and its variance for each stratum separately. To predict the population total the stratum totals of the covariate must be assigned to argument `total` of function `predict`.

```
separate <- svyratio(~poppy, ~agri, design_stsi, separate=TRUE)
tx_h_pop <- tapply(
  grdKandahar$agri, INDEX=grdKandahar$stratum, FUN=sum)
predict(separate, total=tx_h_pop)
```

```
$total
      agri
poppy 28331.32
```

```
$se
      agri
poppy 8882.492
```

10.2.2 Poststratified estimator

In stratified random sampling (Chapter 4) the population is divided into several disjoint subpopulations, and from each subpopulation a probability sample is selected. The subpopulations then serve as strata. The larger the difference in

the stratum means and the smaller the variance within the strata, the larger the gain in precision compared to simple random sampling, see Section 4.1.2.

The alternative to using the subpopulations as strata when selecting the population units is to use them as poststrata in estimating the population mean. For instance, if we have selected a simple random sample from a spatial population, and we have a map of subpopulations possibly related to the study variable, then these subpopulations still can be used in the poststratified estimator. What only needs to be done is to classify the selected units. The subpopulations that serve as poststrata are referred to as groups hereafter.

For any probability sampling design the population mean can be estimated by

$$\hat{\bar{z}}_{\text{pos}} = \sum_{g=1}^G w_g \frac{\hat{t}_g(z)}{\hat{N}_g} = \sum_{g=1}^G w_g \frac{\sum_{k \in \mathcal{S}_g} \frac{z_k}{\pi_k}}{\sum_{k \in \mathcal{S}_g} \frac{1}{\pi_k}}, \quad (10.30)$$

where \mathcal{S}_g is the sample from group g , $w_g = N_g/N$ is the relative size of group g , $\hat{t}_g(z)$ is the estimated total of the study variable for group g , \hat{N}_g is the estimated size of N_g , n_g is the number of sampling units in group g and π_k is the inclusion probability of unit k . The estimated group means are weighted by their relative sizes w_g , which are assumed to be known. In spite of this, the group means are estimated by dividing the estimated group totals by their *estimated* size, \hat{N}_g , because this ratio estimator is more precise than the group sample mean.

This poststratified estimator is the natural estimator for the one-way ANOVA model:

$$\begin{aligned} Z_k &= \mu_g + \epsilon_k \\ \sigma_k^2 &= \sigma_g^2, \end{aligned} \quad (10.31)$$

with μ_g the mean for group (subpopulation) $g = 1, \dots, G$, and σ_g^2 the variance of the study variable of group g .

For simple random sampling, the poststratified estimator reduces to

$$\hat{\bar{z}}_{\text{pos}} = \sum_{g=1}^G w_g \bar{z}_{\mathcal{S}_g}, \quad (10.32)$$

where $\bar{z}_{\mathcal{S}_g}$ is the sample mean of group g . If for all groups we have at least two sampling units, $n_g \geq 2$, the variance of this poststratified estimator of the mean can be estimated by

$$\widehat{V}(\widehat{\bar{z}}_{\text{pos}} | \mathbf{n}_g) = \sum_{g=1}^G w_g^2 \frac{\widehat{S}_g^2}{n_g}, \quad (10.33)$$

where \widehat{S}_g^2 is the estimated spatial variance of z in group g , which for simple random sampling can be estimated by

$$\widehat{S}_g^2 = \frac{1}{n_g - 1} \sum_{k \in \mathcal{S}_g} (z_k - \bar{z}_{\mathcal{S}_g})^2. \quad (10.34)$$

This is an estimator of the *conditional* sampling variance, i.e. the variance of the poststratified estimator over all simple random samples with group sample sizes, collected in the vector \mathbf{n}_g , equal to the group sample sizes in the sample actually selected. The poststratified estimator requires that the sizes (area) of the strata are known. See Section 11.1 for a sampling strategy that does not require known stratum sizes.

The poststratified estimator is illustrated with study area Voorst. We consider the situation that we do not have the map with the five combinations of soil type and land use that served as strata in Chapter 4. The soil type - land use classes (groups) used in the poststratified estimator are only observed at the selected sampling units. Only three poststrata are distinguished: the original strata BA, EA and PA are merged into one stratum SA with function `fct_collapse` of package **forcats** (?). The sizes of these poststrata must be known.

```
load("data/Voorst.RData")
library(forcats)
grdVoorst$poststratum <-
  fct_collapse(grdVoorst$stratum, SA=c("BA", "EA", "PA"))
print(N_g <- tapply(
  grdVoorst$z, INDEX=grdVoorst$poststratum, FUN=length))

  SA      RA      XF
5523   659  1346
```

One hundred points are selected by simple random sampling with replacement. The expected sample sizes per group are proportional to the size of the groups, $E(n_g/n) = N_g/N$, but for a single sample the sample proportions may deviate considerably from the population proportions.

```
n <- 100
N <- nrow(grdVoorst)
set.seed(314)
units <- sample.int(N, size=n, replace=TRUE)
mysample <- grdVoorst[units,]
n_g <- tapply(mysample$z, INDEX=mysample$poststratum, FUN=length)
print(n_g)

SA RA XF
71 6 23
```

The population mean is estimated by first computing the sample means per group, followed by computing the weighted average of the sample means, using the relative sizes of the groups as weights.

```
mz_g <- tapply(mysample$z, INDEX=mysample$poststratum, FUN=mean)
w_g <- N_g/N
print(mean.poststrat <- sum(w_g*mz_g))

[1] 8.049562
```

The variance of the estimator of the mean is estimated by computing the sample variances per group, dividing these by the sample sizes per group, and computing the weighted average, using as weights the squared relative group sizes. This estimated sampling variance is the variance of the estimator of the mean over all simple random samples with 71 units of group SA, 6 units of group RA and 23 units of XF.

```
S2z_g <- tapply(mysample$z, INDEX=mysample$poststratum, FUN=var)
v_mz_g <- S2z_g/as.numeric(n_g)
print(condse_mz <- sqrt(sum(w_g^2*v_mz_g)))

[1] 0.3540979
```

Note that this variance estimator can only be computed with at least two units per group. For that reason, I recommend to use a limited number of groups, especially for small sample sizes.

Function `postStratify` of package `survey` can be used to compute the post-stratified estimator, and its standard error.

```
mysample$weights <- N/n
design_si <- svydesign(id=~1, weights=~weights, data=mysample)
pop <- data.frame(poststratum=c("SA","RA","XF"), Freq=N_g)
mysample_pst <- postStratify(
  design_si, strata=~poststratum, population=pop)
svymean(~z, mysample_pst)

      mean      SE
z 8.0496 0.3492
```

? warns for data snooping. By defining groups after analyzing the data, arbitrarily small sampling variances of the estimated mean can be obtained.

10.3 Model-assisted estimation using machine learning techniques

? review model-assisted estimators based on machine learning techniques. Of special interest is the general approach proposed by ? for incorporating non-linear predictions in the model-assisted estimator. They show how non-linear predictions of the study variable, for instance obtained by a regression tree or random forest, can be used in the model-calibration estimator:

$$\hat{\bar{z}}_{MC} = \hat{\bar{z}}_\pi + \hat{a} \left(1 - \frac{1}{N} \sum_{k \in \mathcal{S}} \frac{1}{\pi_k} \right) + \hat{b} \left(\frac{1}{N} \sum_{k=1}^N \hat{m}(\mathbf{x}_k) - \frac{1}{N} \sum_{k \in \mathcal{S}} \frac{\hat{m}(\mathbf{x}_k)}{\pi_k} \right), \quad (10.35)$$

with \hat{b} a slope coefficient estimated by

$$\hat{b} = \frac{\sum_{k \in \mathcal{S}} 1/\pi_k \{\hat{m}(\mathbf{x}_k) - \hat{\bar{m}}_\pi\} \{z_k - \hat{\bar{z}}_\pi\}}{\sum_{k \in \mathcal{S}} 1/\pi_k \{\hat{m}(\mathbf{x}_k) - \hat{\bar{m}}_\pi\}^2}, \quad (10.36)$$

with \hat{z}_π the π estimator of the population mean of the study variable, \hat{m}_π the π estimator of the population mean of the predicted values, and \hat{a} an intercept estimated by

$$\hat{a} = (1 - \hat{b}) \left(\frac{1}{N} \sum_{k \in \mathcal{S}} \frac{z_k}{\pi_k} \right). \quad (10.37)$$

The second term in Equation (10.35) cancels for all sampling designs for which the sum of the design weights, i.e. the sum of the reciprocal of the inclusion probabilities, equals the population size: $\sum_{k \in \mathcal{S}} 1/\pi_k = N$. Only for some unequal probability sampling designs this may not be the case.

The alternative is to plug the fitted values $\hat{m}(\mathbf{x}_k)$ into the generalised difference estimator, Equation (10.2). If we drop the second term, the model-calibration estimator can be rewritten as

$$\hat{z}_{MC} = \frac{1}{N} \sum_{k=1}^N \hat{b} \hat{m}(\mathbf{x}_k) + \frac{1}{N} \sum_{k \in \mathcal{S}} \frac{z_k - \hat{b} \hat{m}(\mathbf{x}_k)}{\pi_k}. \quad (10.38)$$

Comparison with the generalised difference estimator, Equation (10.2), shows that these two estimators are equivalent when $\hat{b} = 1$. For non-linear working models, generally $\hat{b} \neq 1$, so that these two estimators are not the same. ⁷ shows that the calibration estimator has a general optimality property.

In case you are confused by all these model-assisted estimators, let me clarify. The most general estimator is the model-calibration estimator. If we take for \hat{b} the value 1, this estimator is equivalent to the generalised difference estimator (Equation (10.2)). The predictions $\hat{m}(\mathbf{x}_k)$ in these estimators can be computed either by a linear model or a non-linear model. If a linear model is used in the generalised difference estimator, this estimator is equal to the generalised regression estimator. With linear models $\hat{b} = 1$, so that all three estimators are equal.

For simple random sampling the inclusion probabilities of the units are the same for all points: $\pi_k = n/N$, reducing Equations (10.35) and (10.36) to

$$\hat{z}_{MC} = \frac{1}{n} \sum_{k \in \mathcal{S}} z_k + \hat{b}_{SI} \left(\frac{1}{N} \sum_{k=1}^N \hat{m}(\mathbf{x}_k) - \frac{1}{n} \sum_{j \in \mathcal{S}} \hat{m}(\mathbf{x}_j) \right), \quad (10.39)$$

with \hat{b}_{SI} equal to

$$\hat{b}_{\text{SI}} = \frac{\sum_{k \in \mathcal{S}} \{\hat{m}(\mathbf{x}_k) - \bar{m}(\mathbf{x}_k)\} \{z_k - \bar{z}_{\mathcal{S}}\}}{\sum_{k \in \mathcal{S}} \{\hat{m}(\mathbf{x}_k) - \bar{m}(\mathbf{x}_k)\}^2}. \quad (10.40)$$

The variance of the model-assisted calibration estimator equals

$$V(\hat{\bar{z}}_{\text{MC}}) = V(\hat{\bar{\epsilon}}_{\pi}), \quad (10.41)$$

with $\hat{\bar{\epsilon}}_{\pi}$ the π estimator of the population mean of the residuals ϵ . For sampling designs with fixed sample size these residuals are equal to $\epsilon_k = z_k - b m(\mathbf{x}_k)$, with $m(\mathbf{x}_k)$ the fitted values based on all population units, and b the population fit of the (superpopulation) regression coefficient:

$$b = \frac{\sum_{k=1}^N \hat{m}(\mathbf{x}_k) z_k}{\sum_{k=1}^N \hat{m}(\mathbf{x}_k)^2}. \quad (10.42)$$

An estimator of the variance is obtained by replacing the population fits $m(\mathbf{x}_k)$ by their sample estimates $\hat{m}(\mathbf{x}_k)$, and b by its estimator, Equation (10.36). For simple random sampling with replacement from finite populations and simple random sampling from infinite populations the variance equals

$$V(\hat{\bar{z}}_{\text{MC}}) = \frac{S^2(\epsilon)}{n}, \quad (10.43)$$

with $S^2(\epsilon)$ the population variance of the residuals ϵ .

The variance of the generalised difference estimator equals

$$V(\hat{\bar{z}}_{\text{dif}}) = V(\hat{\bar{d}}_{\pi}), \quad (10.44)$$

with $\hat{\bar{d}}_{\pi}$ the π estimator of the population mean of the differences $d_k = z_k - m(\mathbf{x}_k)$. An estimator is obtained by substituting $\hat{m}(\mathbf{x}_k)$ for $m(\mathbf{x}_k)$ to compute the differences.

The data of Eastern Amazonia are used to illustrate model-assisted estimation of AGB, using five environmental covariates in predicting AGB. First a regression

tree is used as a machine learning technique for prediction, after that a random forest. For an introduction to regression trees and random forest modelling, see this blog¹². In this blog the study variable is a categorical variable, whereas in our example the study variable is quantitative (continuous). This is not essential. The only difference is the measure for quantifying how good a split is. With a quantitative study variable this is quantified by the following sum of squares:

$$SS = \sum_{k=1}^2 \sum_{i=1}^{N_k} (z_{ki} - \bar{z}_k)^2, \quad (10.45)$$

with \bar{z}_k the sample mean of group k .

10.3.1 Predicting with a regression tree

A simple random sample without replacement of size 100 is selected.

```
N <- nrow(gridAmazonia)
n <- 100
set.seed(314)
units <- sample.int(N, size=n, replace=FALSE)
covs <- c("SWIR2", "Terra_PP", "Prec_dm", "Elevation", "Clay")
mysample <- gridAmazonia[units, c("AGB", covs)]
```

Package **rpms** (?) is used to build a regression tree for AGB, using all five covariates as predictors. Note that I now use the original untransformed SWIR2 as a predictor. Transformation of predictors so that the relation with the study variable becomes linear is not needed when fitting a non-linear model such as a regression tree. Using the default value of 0.05 for argument **pval** leads to the following tree.

```
library(rpms)
tree <- rpms(
  AGB~SWIR2+Terra_PP+Prec_dm+Elevation+Clay, data=mysample,
```

¹<https://victorzhou.com/blog/intro-to-random-forests/>

²<https://victorzhou.com/blog/intro-to-random-forests/>

```
pval=0.05)
print(tree)

RPMS Recursive Partitioning Equation
AGB ~ SWIR2 + Terra_PP + Prec_dm + Elevation + Clay

Estimating Equation
AGB ~ 1

[1] "Simple Random Sample"
[1] "R-squared of model: 0.891634330420172"

===== Tree Model =====

Splits
sp   SWIR2 <= 569.983734130859
sp   SWIR2 <= 569.983734130859 & SWIR2 <= 522.027557373047
sp   SWIR2 <= 569.983734130859 & SWIR2 <= 522.027557373047 & Elevation <= 71.5
sp   SWIR2 <= 569.983734130859 & SWIR2 <= 522.027557373047 & Elevation > 71.5
sp   SWIR2 > 569.983734130859 & SWIR2 <= 835.015594482422

coefficients
node      1
 8  275.33333
18  316.65217
19  287.30769
5   253.61538
6   154.00000
7   77.59091
```

The regression tree is used to predict AGB for all population units.

```
AGBpred <- predict(tree, newdata=gridAmazonia)
```

The population mean is then estimated by the generalised difference estimator.

```
d <- gridAmazonia$AGB[units]-AGBpred[units]
mean(AGBpred) + mean(d)
```

```
[1] 226.7433
```

Its standard error is estimated by the square root of the variance of the estimator of the mean differences.

```
S2d <- var(d)
sqrt((1-n/N)*S2d/n)
```

```
[1] 3.212222
```

This estimation procedure is implemented in function `gregTree` of package **mase** (?).

```
library(mase)
pi <- rep(n/N,n)
res <- gregTree(
  mysample$AGB, xsample=mysample[,cobs],
  xpop=gridAmazonia[,cobs], pi=pi,
  var_est=TRUE, var_method="LinHTSRS")
res$pop_mean
```

```
[1] 226.7433
```

```
sqrt(res$pop_mean_var)
```

```
[1] 3.212222
```

The variance of the estimator of the mean can also be estimated by bootstrapping the sample (? , Section 9.3.3).

```
res <- gregTree(
  mysample$AGB, xsample=mysample[,cobs],
```

```

xpop=gridAmazonia[,cobs], pi=pi,
var_est=TRUE, var_method="bootstrapSRS", B=100)

sqrt(res$pop_mean_var)

[1,] 4.405042

```

The standard error obtained by the bootstrap is considerably larger than the previous standard error based on a Taylor linearisation of the estimator of the mean. As we will see hereafter, the Taylor linearisation seriously underestimates the true standard error.

The simple random sampling of 100 units and the model-assisted estimation is repeated 500 times, using a regression tree for prediction. The variance is estimated by Taylor linearisation (`var_method=LinHTSRS`) and by bootstrapping (`var_method=bootstrapSRS`) using 100 bootstrap samples.

The variance of the 500 estimated population means of AGB is 20.8. Estimation of the variance through Taylor linearisation strongly underestimates the variance: the average of the 500 estimated variances equals 14.5. On the contrary, the bootstrap variance estimator overestimates the variance: the average of the 500 estimated variances equals 23.4. I prefer to overestimate my uncertainty about the mean instead of being overoptimistic, and so I would recommend to report the bootstrap variance.

10.3.2 Predicting with a random forest

The package `ranger` (?) is used to fit a random forest model for AGB using the five environmental covariates as predictors and the simple random sample of size 100 selected in the previous section. Function `importance` shows how often the covariates are used in a binary splitting. All five covariates are used, SWIR2 by far most often.

```

library(ranger)
set.seed(314)
forest.sample <- ranger(

```

```
AGB~., data=mysample, num.trees=1000, importance="impurity")
importance(forest.sample)
```

```
SWIR2 Terra_PP Prec_dm Elevation Clay
466318.39 214716.35 136505.65 62429.66 34612.68
```

Out-of-bag predictions for the selected units are saved in the element `predictions` of the output-object of the function `ranger`. The fitted model is also used to predict AGB at all 1306296 units (raster cells), using function `predict`.

```
AGBpred_OOB <- forest.sample$predictions
res <- predict(
  forest.sample, data=gridAmazonia, type="response")
AGBpred <- res$predictions
```

Finally, the model-calibration estimate and the generalised difference estimate are computed. Both estimators and their variances are computed in two ways. They differ in how the study variable AGB is predicted for the sampling units:

1. Using all trees of the forest (the predictions obtained with function `predict`).
2. Using only the trees calibrated on bootstrap samples that do not include the sampling unit used as a prediction unit. These out-of-bag predictions are stored in the element `predictions` of the output object of function `ranger`.

The next code chunk shows how the model-calibration estimate can be computed with the AGB data of the simple random sample and the random forest predictions of AGB. First use all trees.

```
b <- sum((AGBpred[units]-mean(AGBpred[units]))*
  (mysample$AGB-mean(mysample$AGB)))/
  sum((AGBpred[units]-mean(AGBpred[units]))^2)
mz_MC <- mean(mysample$AGB) +
  b*(mean(AGBpred)-mean(AGBpred[units]))
```

```
u <- mysample$AGB-AGBpred[units]*b
v_mz_MC <- (1-n/N)*var(u)/n
```

Next, use the out-of-bag predictions.

```
b_OOB <- sum((AGBpred_OOB-mean(AGBpred_OOB))*  
               (mysample$AGB-mean(mysample$AGB)))/  
               sum((AGBpred_OOB-mean(AGBpred_OOB))^2)  
mz_MC_OOB <- mean(mysample$AGB)+  
               b_OOB*(mean(AGBpred)-mean(AGBpred_OOB))  
u_OOB <- mysample$AGB-AGBpred_OOB*b_OOB  
v_mz_MC_OOB <- (1-n/N)*var(u_OOB)/n
```

The two calibration estimates are about equal: 226.8 using sample predictions obtained with function `predict`, and 227.3 with the out-of-bag sample predictions. However, their estimated variances are largely different: 2.35 and 12.46, respectively.

In the next code chunk the generalised difference estimate (Equation (10.2)) is computed. Similar to the model-calibration estimate, this difference estimate is computed from predictions based on all trees, and from the out-of-bag predictions.

```
d <- mysample$AGB-AGBpred[units]
mz_MD <- mean(AGBpred)+mean(d)
v_mz_MD <- (1-n/N)*var(d)/n
#using out-of-bag predictions
d_OOB <- mysample$AGB-AGBpred_OOB
mz_MD_OOB <- mean(AGBpred)+mean(d_OOB)
v_mz_MD_OOB <- (1-n/N)*var(d_OOB)/n
```

For the difference estimator the results are very similar. The two difference estimates are 226.6 and 227, and their estimated variances are 2.7 and 12.8. The model-calibration estimate and the generalised difference estimate are nearly equal.

The sampling and estimation is repeated 1,000 times: 1,000 times a simple random sample of size 100 is selected. Each sample is used to calibrate a random

forest, each forest consisting of 1,000 trees. This resulted in $2 \times 1,000$ model-calibration estimates and their estimated variances, and $2 \times 1,000$ difference estimates and their estimated variances. To limit the computing time a 5 km \times 5 km subgrid is used for selecting the simple random samples and for predicting AGB with the random forest.

For each estimator the 1,000 estimated means are used to compute the relative bias:

$$bias = \frac{\frac{1}{1000} \sum_{i=1}^{1000} \hat{\bar{z}}_i - \bar{z}}{\bar{z}} . \quad (10.46)$$

Besides, for each estimator the variance of the 1,000 estimates is computed, which can be compared with the mean of the 1,000 estimated variances. The mean of the estimated variances is used to compute the variance to mean squared error ratio:

$$R = \frac{\frac{1}{1000} \sum_{i=1}^{1000} \hat{V}(\hat{\bar{z}}_i)}{MSE} , \quad (10.47)$$

with

$$MSE = \frac{1}{1000} \sum_{i=1}^{1000} (\hat{\bar{z}}_i - \bar{z})^2 . \quad (10.48)$$

Ideally this ratio equals 1. If it is smaller than 1, the variance estimator underestimates the mean squared error.

Finally, the relative efficiency is computed as the ratio of the MSE of the π estimator and the MSE of a model-assisted estimator. The π estimator is unbiased, so the MSE equals the variance, which can be computed without error by the population variance divided by the sample size. If the relative efficiency is larger than 1, the model-assisted estimator is more accurate than the π estimator.

The variance of the 1,000 model-calibration estimates, using all trees to predict AGB for the sample units, equals 15.0281. This is 5.6 times larger than the average of the 1,000 estimated variances, which equals 2.6612. When using the out-of-bag sample predictions, the experimental variance is about equal to the average of the 1,000 variance estimates: 15.5255 versus 14.975.

Table 10.1: Summary statistics of 1,000 repeated calibration estimates of mean of aboveground biomass in Eastern Amazonia and their variance, using random forest model with five covariates as a working model and out-of-bag sample predictions, for simple random sampling without replacement.

Sample size:	50	100	250
Relative bias	-0.0014	-0.0006	-0.0012
Experimental variance	33.9075	15.5255	5.6111
Mean variance estimates	36.1512	14.9750	5.0467
Variance to MSE ratio	1.0642	0.9642	0.8882
Relative efficiency	5.0066	5.4755	5.9866

The reason of underestimation of the variance when predicting AGB at sample units with function `predict` is that all 1,000 trees are used in prediction, including the trees calibrated on bootstrap samples that contain the unit in the sample to be predicted. On the contrary, the out-of-bag predicted values are computed as the average of predictions from the trees calibrated on bootstrap samples that do not contain the sample unit to be predicted. The default sample fraction is 0.632, see argument `sample.fraction` of function `ranger`, so with 1,000 trees these predictions are the average of, on average, 368 tree predictions. This explains why the out-of-bag prediction errors are larger than the prediction errors obtained with function `predict`. In other words, the variance of the out-of-bag differences d and of the out-of-bag residuals u are larger than those obtained with predicting using all trees.

Hereafter I report the results obtained with the out-of-bag samples only.

The relative bias is negligibly small for all sample sizes (Table 10.1). Besides, for $n = 100$ and 250 the average of the 1,000 variance estimates is close to the variance of the 1,000 estimates; for $n = 50$ the experimental variance is overestimated by the variance estimator. The variance to MSE ratio is smaller than 1 for $n = 100$ and 250 , but larger than 1 for $n = 50$. The model-calibration estimator is much more accurate than the π estimator for all three sample sizes. The relative efficiency increases with the sample size.

The performance statistics of the generalised difference estimator are very similar to those of the model-calibration estimator (Table 10.2).

Table 10.2: Summary statistics of 1,000 repeated generalised difference estimates of aboveground biomass in Eastern Amazonia and their variance, using random forest model with five covariates as a working model and out-of-bag sample predictions, for simple random sampling without replacement.

Sample size:	50	100	250
Relative bias	-0.0008	-0.0006	-0.0013
Experimental variance	34.5188	16.2339	5.6891
Mean variance estimates	38.4629	15.4568	5.1125
Variance to MSE ratio	1.1144	0.9520	0.8867
Relative efficiency	4.9275	5.2379	5.8997

10.4 Big data and volunteer data

In the past decades numerous large data sets became available, and this number will further increase in the future, think of data sets collected by the numerous satellites. These data sets may contain valuable information about the study variables, so that they can be used in model-assisted estimation of global (this chapter) or local means and totals (Chapter 14).

Another interesting source of information are the geographic data collected by volunteers. These data typically are from non-probability samples. Despite this, there can be valuable information in these data about the global or local means of the study variable.

When the volunteer data are supplemented by a probability sample with observations on the study variable, the volunteer data can be used at the design stage and/or at the estimation stage. As an example of the first approach, the volunteer data are used to predict the study variable at a fine discretisation grid. These predictions are then used to construct strata for supplemental sampling, for instance by the *cum-root-f* method (Section 4.4), or using the approach described in Section 13.2.

At the estimation stage the volunteer data are used to predict the study variable at points of the supplemental probability sample and at the nodes of a discretisation grid. These predictions are then used in model-assisted estimation, using the generalised difference or regression estimator, as explained in this chapter and in ?.

? compared this model-assisted estimation approach with a certainty stratum approach for estimating the area covered by land cover classes and the accuracy of land cover maps. The volunteer data are treated as the data of a stratum of which all units are observed. A probability sample is selected from the remaining units not observed by the volunteers. The total (area of a land cover class) of the certainty stratum is added to the estimated total of the subpopulation not observed by the volunteers.

The model-assisted approach requires a supplemental sample with observations of the study variable. ? described an approach in which the big data sample is combined with a probability sample with observations of one or more ancillary variables.

? also describe an alternative approach that does not require a probability sample at all. In this approach the big data sample is subsampled to correct the selection bias in the big data sample. The subsample is selected by inverse sampling, using data on an ancillary variable x , either from a census or a probability sample. The subsample is selected with conditional inclusion probabilities equal to the subsample size multiplied by an importance weight (Equation 4, ?).

Chapter 11

Two-phase random sampling

The regression and ratio estimators of Chapter 10 require that the means of the ancillary variables are known. If these are unknown, but the ancillary variable can be measured cheaply, one may decide to estimate the population means of the ancillary variables from a large sample. The study variable is measured in a random subsample of this large sample only. This technique is known in the sampling literature as two-phase random sampling or double sampling. Another application of two-phase sampling is two-phase sampling for stratification. Stratified random sampling (Chapter 4) requires a map with the strata. The poststratified estimator of Section 10.2.2 requires that the sizes of the strata are known. With two-phase sampling for stratification neither a map of the strata, nor knowledge of the stratum sizes is required. Note that the term ‘phase’ does not refer to a period of time; all data can be collected in one sampling campaign. Let me also explain the difference with two-stage cluster sampling (Chapter 7). In two-stage cluster random sampling we have two types of sampling units, clusters of population units and individual population units. In two-phase sampling we have one type of sampling units only, the population units.

In two-phase sampling for regression and two-phase sampling for stratification the two phases have the same aim, i.e. to estimate the population mean of the study variable. The observations of the covariate(s) and/or strata in the

first phase are merely done to increase the precision of the estimated mean of the study variable. Another application of two-phase sampling is subsampling an existing probability sample designed for a different aim. So in this case the observations in the first phase sample may not be related to the study variable observed in the second-phase sample. An example is LUCAS-Topsoil (?). LUCAS-Topsoil is a subsample of approximately 22,000 units sampled from a much larger sample, the LUCAS sample, designed for estimating totals of land cover and land use classes across the European Union. It was not feasible to observe the soil properties at all sites of the LUCAS sample, and for that reason a subsample was selected. Regrettably, this subsample is not a probability sample from the LUCAS sample: the inclusion probabilities are either zero or unknown, both are not allowed. Design-based or model-assisted estimation of means of soil properties for domains of interest is not feasible. The only option is model-based prediction.

In case the subsample is a probability subsample from the first-phase sample, and no variable observed in the first-phase sample is of use for estimating the total or mean of the study variable observed in the subsample, the population total can be estimated by the π estimator:

$$\hat{t}(z) = \sum_{k \in \mathcal{S}_2} \frac{z_k}{\pi_{1k} \pi_{k|\mathcal{S}_1}} = \sum_{k \in \mathcal{S}_2} \frac{z_k}{\pi_k^*}, \quad (11.1)$$

with n_2 the size of the second-phase sample (subsample size), π_{1k} the probability that unit k is selected in the first phase, and $\pi_{k|\mathcal{S}_1}$ the probability that unit k is selected in the second phase, given the first-phase sample \mathcal{S}_1 . This general π estimator for two-phase sampling, referred to as the π^* -estimator by ?, can be used for any combination of probability sampling designs in the first and second phase.

To derive the variance it is convenient to write the total estimation error as the sum of two errors:

$$\hat{t}(z) - t(z) = \left(\sum_{k \in \mathcal{S}_1} \frac{z_k}{\pi_{1k}} - t(z) \right) + \left(\sum_{k \in \mathcal{S}_2} \frac{z_k}{\pi_k^*} - \sum_{k \in \mathcal{S}_1} \frac{z_k}{\pi_{1k}} \right) = \epsilon_1 + \epsilon_2. \quad (11.2)$$

The first error ϵ_1 is the error in the estimated population total, as estimated by the usual π estimator using the study variable values for the units in the

first-phase sample. This estimator cannot be computed in practice as the study variable values are only known for a subset of the units in the first-phase sample. The second error ϵ_2 is the difference between the π^* -estimator using the study variable values for the units in the subsample only, and the π estimator using the study variable values for all units in the first-phase sample.

The variance of the π^* -estimator can be decomposed into the variance of these two errors as follows:

$$V_{p_1,p_2}(\hat{t}) = V_{p_1}E_{p_2}(\hat{t}|\mathcal{S}_1) + E_{p_1}V_{p_2}(\hat{t}|\mathcal{S}_1) = V_{p_1}(\epsilon_1) + E_{p_1}V_{p_2}(\epsilon_2|\mathcal{S}_1), \quad (11.3)$$

with V_{p_1} and E_{p_1} the variance of the estimator for the population total over repeated sampling with the design of the first phase, and V_{p_2} and E_{p_2} the variance and expectation of the estimator for the population total over repeated sampling with the design of the second phase. The population mean can be estimated by the estimated total divided by the population size N (or A for infinite populations).

11.1 Two-phase random sampling for stratification

In two-phase sampling for stratification a large sample is taken, and the selected sampling units are all classified. The classes thus formed are then used as strata in the second sampling phase. A stratified subsample is selected, and the study variable is observed on the units in the subsample only.

This sampling design is applied, for instance, to monitor land use and land cover in the European Union. In the first phase a systematic random sample of points is selected consisting of a $2 \text{ km} \times 2 \text{ km}$ grid. Land use and land cover (LULC) is then determined at the selected grid nodes, using orthophotographs, satellite imagery and fieldwork. The idea is that this procedure results in a more accurate classification of LULC at the selected points than by overlaying the points with an existing LULC map such as the Corine Land Cover map. The site-specific determinations of LULC classes are then used to select a stratified random subsample.

Two-phase sampling for stratification is illustrated with study area Voorst. A map with five combinations of soil type and land use is available of this study

area. These combinations were used as strata in Chapter 4, and in the poststratified estimator of Section 10.2.2 the strata sizes were used. Here we consider the situation that we do not have this map, and that we do not know the sizes of these strata either. In the first phase a simple random sample of size 100 is selected. In the field the soil type - land use class is determined for the selected units, see Figure 11.1. Here we assume that the field determinations are equal to the classes as shown on the map.

```
load("data/Voorst.RData")
n1 <- 100
set.seed(123)
N <- nrow(grdVoorst)
units <- sample.int(N, size=n1, replace=FALSE)
mysample <- grdVoorst[units,]
```

The simple random sample is subsampled by stratified simple random sampling, using the soil type - land use classes as strata. The total sample size of the second phase is set to 40. The number of units in the simple random sample per stratum is determined. Then the subsample size per stratum is computed for proportional allocation, and it is checked whether the sum of the stratum sample sizes equals 40. Finally function **strata** of package **sampling** (?) is used to select a stratified simple random sample without replacement, see Chapter 4 for details.

```
library(sampling)
n2 <- 40
n1_h <- tapply(mysample$z, INDEX=mysample$stratum, FUN=length)
n2_h <- round(n1_h/n1*n2, 0)
n2_h_ord <- n2_h[unique(mysample$stratum)]
units <- sampling::strata(mysample, stratanames="stratum",
                           size=n2_h_ord, method="srswor")
mysubsample <- getdata(mysample, units)
table(mysubsample$stratum)

BA EA PA RA XF
15 8 6 4 7
```

With simple random sampling in the first phase and stratified simple random

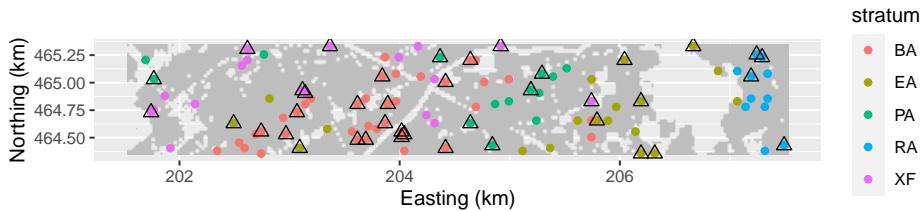


Figure 11.1: Two-phase sample for stratification from Voorst. Coloured dots: simple random sample of 100 points classified into five soil type - land use classes (first-phase sample). Triangles: stratified simple random sample of 40 points with measurements of SOM (second-phase sample).

sampling in the second phase, the population mean can be estimated by

$$\hat{\bar{z}} = \sum_{h=1}^{H_{S_1}} \frac{n_{1h}}{n_1} \bar{z}_{S_{2h}}, \quad (11.4)$$

where H_{S_1} is the number of strata used for stratification of the first-phase sample, n_{1h} is the number of units in the first-phase sample that form stratum h in the second phase, n_1 is the total number of units of the first phase sample and $\bar{z}_{S_{2h}}$ is the mean of the subsample from stratum h .

```
mz_h_subsam <- tapply(
  mysubsample$z, INDEX=mysubsample$stratum, FUN=mean)
mz <- sum(n1_h/n1*mz_h_subsam, na.rm=TRUE)
```

The estimated population mean equals 8.68. The sampling variance over repeated sampling with both designs can be approximated¹ by (see Equation at bottom of p. 353 in ?)

¹In the approximation it is assumed that N is much larger than n_1 , and $(n_{1h} - 1)/(n_1 - 1)$ is replaced by n_{1h}/n_1 .

$$\begin{aligned}\widehat{V}(\widehat{z}) &= \sum_{h=1}^{H_{S_1}} \left(\frac{n_{1h}}{n_1} \right)^2 \frac{\widehat{S}_{S_{2h}}^2}{n_{2h}} \\ &+ \frac{1}{n_1} \sum_{h=1}^{H_{S_1}} \frac{n_{1h}}{n_1} \left(\bar{z}_{S_{2h}} - \widehat{z} \right)^2.\end{aligned}\quad (11.5)$$

```
S2z_h_subsam <- tapply(
  mysubsample$z, INDEX=mysubsample$stratum, FUN=var)
w1_h <- n1_h/n1
v_mz_1 <- sum(w1_h^2*S2z_h_subsam/n2_h)
v_mz_2 <- 1/n1*sum(w1_h*(mz_h_subsam-mz)^2)
se_mz <- sqrt(v_mz_1+v_mz_2)
```

The estimated standard error equals 0.701.

The mean and its standard error can be estimated with functions `twophase` and `svymean` of package `survey` (?). A `data.frame` with the first phase sample is assigned to the argument `data` of function `twophase`. A column in this `data.frame`, assigned to the argument `subset`, is an indicator with value `TRUE` if this unit is selected in the second phase and `FALSE` otherwise.

```
library(survey)
mysample$ind <- FALSE
mysample$ind[units$ID_unit] <- TRUE
mysample$fpc1 <- N
labels <- sort(unique(mysample$stratum))
fpc2 <- n1_h
lut <- data.frame(stratum=labels, fpc2)
mysample <- merge(x=mysample, y=lut)
mysample$stratum <- as.factor(mysample$stratum)
design_2phase <- survey::twophase(
  id=list(~1,~1), strata=list(NULL,~stratum),
  data=mysample, subset=~ind, fpc=list(~fpc1,~fpc2))
svymean(~z, design_2phase)
```

```
mean      SE
z 8.6783 0.6869
```

As shown in the next code chunk, the standard error is computed with the original variance estimator, without approximation (see Equation (9.4.14) in ?).

```
v_mz_1 <- 1/N^2*N*(N-1)*
  sum(((n1_h-1)/(n1-1))-((n2_h-1)/(N-1)))*w1_h*S2z_h_subsam/n2_h
v_mz_2 <- 1/N^2*(N*(N-n1))/(n1-1)*sum(w1_h*(mz_h_subsam-mz)^2)
sqrt(v_mz_1+v_mz_2)

[1] 0.6868616
```

11.2 Two-phase random sampling for regression

The simple regression estimator of Equation (10.10) requires that the population mean of the ancillary variable x is known. This section is about applying the regression estimator in situations where this mean of x is unknown. A possible application is estimating the soil carbon stock (until a given depth) in an area. To estimate this carbon stock soil samples are collected and analyzed in a laboratory. These laboratory measurements can be very accurate, but also expensive. Proximal sensors can be used to derive soil carbon concentrations from the spectra. Compared to laboratory measurements of soil these proximal sensor determinations are much cheaper, but also less accurate. If there is a relation between the laboratory and proximal sensing determinations of SOC, then we expect that the regression estimator of the carbon stock will be more accurate than the π estimator which does not exploit the proximal sensing measurements. However, the population mean of the proximal sensing determinations is unknown. What we can do is to estimate this mean from a large sample. For a subsample of this large sample, SOC concentration is also measured in the laboratory. This is another example of two-phase sampling. Intuitively, with two-phase sampling the variance of the regression estimator of the total carbon stock will be larger than when the population mean of the proximal sensing determinations would be known. We are more uncertain about the total carbon stock, because we are uncertain about the population mean of the proximal sensing determinations.

Figure 11.2 shows a two-phase sample from Eastern Amazonia (Brazil). In

the first phase 250 points (the dots in the plot) are selected by simple random sampling. In the second phase a subsample of 100 points (the triangles in the plot) are selected from the 250 points by simple random sampling without replacement. At all 250 points of the first-phase sample the covariate lnSWIR2 is measured, whereas the study variable (AGB) is measured at the 100 subsample points only. This sampling design does not require a full-coverage map of the covariate. In this case we do have a full coverage map of lnSWIR2, so a two-phase sample is not needed. I chose the same case study to show the effect of ignorance of the population mean of the covariate on the variance of the regression estimator.

```
load("data/Amazonia_1km.RData")
n1 <- 250; n2 <- 100
set.seed(314)
units_1 <- sample.int(nrow(gridAmazonia), size=n1, replace=FALSE)
mysample <- gridAmazonia[units_1,]
units_2 <- sample.int(n1, size=n2, replace=FALSE)
mysubsample <- mysample[units_2,]
```

Estimation of the population mean or total by the regression estimator from a two-phase sample is very similar to estimation when the covariate mean is known, as described in Section 10.1.1 (Equation (10.10)). The observations on the *subsample* can be used to estimate the regression coefficient b . The true population mean of the ancillary variable, (\bar{x} in Equation (10.10)), is unknown now. This true mean is replaced by the mean as estimated from the relatively large first-phase sample, \bar{x}_{S_1} . The estimated mean of the covariate, \hat{x}_S in Equation (10.10), is estimated from the subsample, \bar{x}_{S_2} . This leads to the following estimator:

$$\hat{z} = \bar{z}_{S_2} + \hat{b} (\bar{x}_{S_1} - \bar{x}_{S_2}) , \quad (11.6)$$

where \bar{z}_{S_2} is the subsample mean of the study variable, and \bar{x}_{S_1} and \bar{x}_{S_2} are the means of the covariate in the first-phase sample and subsample (second-phase sample), respectively.

The sampling variance is larger than that of the regression estimator with known mean of x . The variance can be decomposed into a component equal to the sampling variance of the estimator of the mean of z with the sampling design

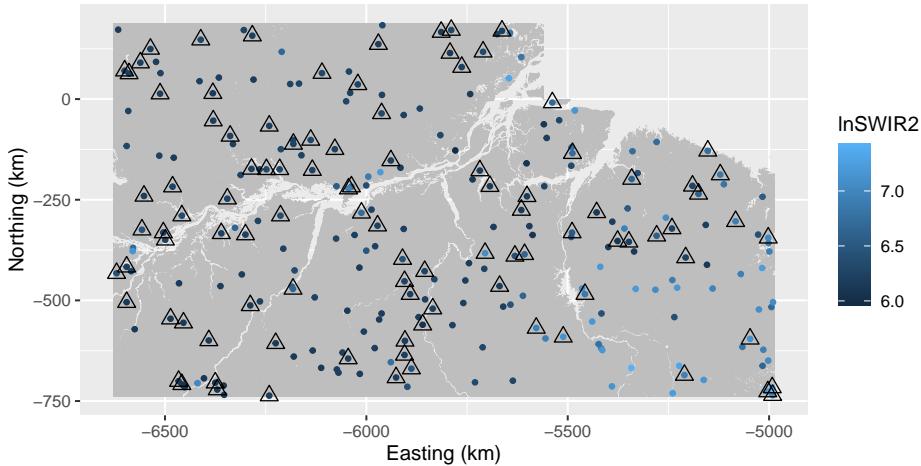


Figure 11.2: Two-phase sample for the regression estimator of the population mean of aboveground biomass (AGB) in Eastern Amazonia. Coloured dots: simple random sample of 250 units with measurements of covariate lnSWIR2 (first-phase sample). Triangles: simple random subsample of 100 units with measurements of AGB (second-phase sample).

of the first phase (in this case simple random sampling without replacement), supposing that the study variable would be observed on all units of the first-phase sample, and a component equal to the sampling variance of the regression estimator of the sample mean of z in the first-phase sample with the design of the second phase sample (again simple random sampling without replacement in this case):

$$\widehat{V}(\widehat{z}) = \left(1 - \frac{n_1}{N}\right) \frac{\widehat{S}^2(z)}{n_1} + \left(1 - \frac{n_2}{n_1}\right) \frac{\widehat{S}^2(\epsilon)}{n_2}, \quad (11.7)$$

with $\widehat{S}^2(\epsilon)$ the variance of the regression residuals as estimated from the subsample:

$$\widehat{S}^2(\epsilon) = \frac{1}{(n_2 - 1)} \sum_{k \in \mathcal{S}} \epsilon_k^2. \quad (11.8)$$

Note the finite population corrections (fpc), $(1 - n_1/N)$ and $(1 - n_2/n_1)$, in the variance estimator. These fpc's account for the reduced variance due to sampling the finite population and subsampling the first-phase sample without replacement.

```
lm_subsample <- lm(AGB~lnSWIR2, data=mysubsample)
ab <- coef(lm_subsample)
mx_sam <- mean(mysample$lnSWIR2)
mx_subsam <- mean(mysubsample$lnSWIR2)
mz_subsam <- mean(mysubsample$AGB)
mz_reg2ph <- mz_subsam+ab[2]*(mx_sam-mx_subsam)
```

The estimated population mean equals 228.1. The standard error can be approximated as follows.

```
e <- residuals(lm_subsample)
S2e <- sum(e^2)/(n2-1)
S2z <- var(mysubsample$AGB)
N <- nrow(gridAmazonia)
se_mz_reg2ph <- sqrt((1-n1/N)*S2z/n1 + (1-n2/n1)*S2e/n2)
```

The estimated standard error equals 6.35.

The regression estimator for two-phase sampling and its standard error can also be computed with package **survey**, as shown below. The standard error slightly differs from the standard error computed above because it is computed with the g-weights, see Section 10.1.1. Note the argument **fpc=list(~N,NULL)**. There is no need to add the first phase sample size as a second element of the list, because this sample size is simply the number of rows of the data frame. Setting the second element of the list to **NULL** does not mean that the standard error is computed for with replacement sampling in the second phase. Function **twophase** assumes that the second phase units are always selected without replacement.

```
mysample$id <- 1:n1
mysample$ind <- ifelse(mysample$id %in% units_2, TRUE, FALSE)
mysample$N <- rep(N, n1)
```

```

design_2phase <- survey::twophase(
  id=list(~1,~1), data=mysample, subset=~ind, fpc=list(~N,NULL))
mysample_cal <- calibrate(
  design_2phase, formula=~lnSWIR2, calfun="linear", phase=2)
svymean(~AGB, mysample_cal)

      mean      SE
AGB 228.07 7.2638

```

Exercises

1. Write an **R** script to select a simple random sample without replacement of 250 units from Eastern Amazonia (`data/Amazonia_1km.RData`) and a subsample of 100 units by simple random sampling without replacement. Repeat this 1,000 times in a for-loop.
 - Use each one-phase sample (sample of 250 units) to estimate the population mean of AGB by the regression estimator (Equation (10.10) in Chapter 10), using `lnSWIR2` as a covariate.
 - Use each two-phase sample to estimate the regression estimator for two-phase sampling (Equation (11.6)). Estimate the population mean of `lnSWIR2` from the first-phase sample of 250 points. The study variable `AGB` is observed at the subsample only. Approximate the variance of the regression estimator for two-phase sampling (Equation (11.7)).
 - Compute the variance of the 10,000 regression estimates of the mean of `AGB`.
 - Compute the variance of the regression estimator for this two-phase sampling design.
 - Compare the two variances, and explain the difference.
 - Compute the average of the 10,000 approximate variances, and compare with the variance of the 10,000 estimated means, as estimated by the regression estimator for two-phase sampling.

Chapter 12

Computing the required sample size

An important decision in designing sampling schemes is the number of units to select. In other words, what should the sample size be? If a certain budget is available for sampling, we can determine the affordable sample size from this budget. A cost model is then needed. For simple random sampling the affordable sample size can then simply be computed by the available budget divided by the average cost per unit.

The alternative to deriving the affordable sample size from the budget is to start from a requirement on the quality of the survey result obtained by statistical inference. Two types of inference are distinguished, estimation and testing. The required sample size depends on the type of sampling design. With stratified random sampling we expect that we need fewer sampling units compared to simple random sampling to estimate the population mean with the same precision, whereas with cluster random sampling and two-stage cluster random sampling in general we need more sampling units. To compute the sample size given some quality requirement we may start with computing this required sample size for simple random sampling, and then correct this sample size to account for the design effect. Therefore I start with presenting formulas for computing the required sample size for simple random sampling. Section 12.4 describes how required sample sizes for other types of sampling designs can be derived.

Hereafter formulas for computing the required sample size are presented for simple random sampling *with replacement* (SIR) of finite populations and simple random sampling of infinite populations. For SI *without replacement* (SI) of finite populations these sample sizes can be corrected by (?)

$$n_{\text{SI}} = \frac{n}{1 + \frac{n_{\text{SIR}}}{N}}, \quad (12.1)$$

with n the required sample size for simple random sampling with replacement.

12.1 Standard error of estimator

A first option is to set a limit on the variance of the estimated mean, see Equation (3.7), or on the square root of this variance, the standard error of the estimated mean. Given a chosen limit for the standard error se_{\max} , the required sample size for simple random sampling with replacement can be computed by

$$n = \left(\frac{S^*(z)}{se_{\max}} \right)^2, \quad (12.2)$$

with $S^*(z)$ a prior estimate of the population standard deviation. The required sample size n should be rounded to the nearest integer greater than the right-hand side of Equation (12.2). This also applies to the following equations.

For the population proportion (areal fraction) as the parameter of interest, the required sample size can be computed by (see Equation (3.14))

$$n = \left(\frac{\sqrt{p^*(1-p^*)}}{se_{\max}} \right)^2 + 1, \quad (12.3)$$

with p^* a prior estimate of the population proportion. Note that in this case we need a prior estimate of the population parameter of interest itself, whereas for the population mean a prior estimate is needed of the population standard deviations and therefore differs from the parameter of interest. The parameter of which a prior estimate is needed is referred to as the design parameter.

Alternatively, we may require that with a large probability $1 - \alpha$ the *relative* standard error, i.e. the standard error divided by the population mean¹, may not exceed a given limit rse_{\max} . In this case the required sample size can be computed by

$$n = \left(\frac{S^*(z)}{rse_{\max} \bar{z}} \right)^2 = \left(\frac{cv^*}{rse_{\max}} \right)^2, \quad (12.4)$$

with cv^* a prior estimate of the coefficient of variation $S(z)/\bar{z}$. For a requirement on the relative standard error of the population proportion estimator we obtain

$$n = \left(\frac{p^*(1-p^*)}{rse_{\max} p^*} \right)^2 + 1 = \left(\frac{1-p^*}{rse_{\max}} \right)^2 + 1. \quad (12.5)$$

12.2 Length of confidence interval

Another option is to require that the length of the confidence interval of the mean may not exceed a given limit l_{\max} :

$$2 t_{\alpha/2, n-1} \frac{S(z)}{\sqrt{n}} \leq l_{\max}, \quad (12.6)$$

with $t_{\alpha/2, n-1}$ the $(1 - (\alpha/2))$ quantile of the t distribution with $n - 1$ degrees of freedom, $S(z)$ the population standard deviation of the study variable, and n the sample size. The problem is that we do not know the degrees of freedom (we want to determine the sample size n). Therefore $t_{\alpha/2, n-1}$ is replaced by $u_{\alpha/2}$, the $(1 - (\alpha/2))$ quantile of the standard normal distribution. Rearranging gives

$$n = \left(u_{\alpha/2} \frac{S^*(z)}{l_{\max}/2} \right)^2. \quad (12.7)$$

The requirement can also be formulated as

¹The relative standard error is also referred to as the coefficient of variation. Note that this coefficient of variation is also used for the standard deviation of the study variable in the population divided by the population mean.

$$P(|\hat{\bar{z}} - \bar{z}| \leq d_{\max}) \leq 1 - \alpha , \quad (12.8)$$

with d_{\max} the margin of error: $d_{\max} = l_{\max}/2$.

An alternative is to require that with a large probability $(1 - \alpha)$ the absolute value of the *relative* error of the estimated mean may not exceed a given limit r_{\max} . In formula:

$$P\left(\frac{|\hat{\bar{z}} - \bar{z}|}{\bar{z}} \leq r_{\max}\right) \leq 1 - \alpha . \quad (12.9)$$

Noting that the absolute error equals $r_{\max} \cdot \bar{z}$, and inserting this in Equation (12.7) gives

$$n = \left(u_{\alpha/2} \frac{cv^*}{r_{\max}}\right)^2 . \quad (12.10)$$

As an example the required sample size is computed for estimating the population mean of the soil organic matter concentration in Voorst. The requirement is that with a probability of 95% the absolute value of the *relative* error does not exceed 10%. A prior estimate of 0.5 for the population coefficient of variation is used.

```
cv <- 0.5
rmax <- 0.1
u <- qnorm(p=1-0.05/2, mean=0, sd=1)
n <- ceiling((u*cv/rmax)^2)
```

The same result is obtained with function `nContMoe` of package **PracTools** (?).

```
library(PracTools)
print(nContMoe(moe.sw=2, e=rmax, alpha=0.05, CVpop=cv))
```

[1] 96.03647

12.2.1 Length of confidence interval for a proportion

Each of the methods for computing a confidence interval of a proportion described in Section 3.4.1 can be used to compute the required sample size given a limit for the length of the confidence interval of a proportion. The most simple option is to base the required sample size on the Wald interval (Equation (3.21)), so that the required sample size can be computed with

$$n = \left(u_{\alpha/2} \frac{\sqrt{p^*(1-p^*)}}{l_{\max}/2} \right)^2 + 1 . \quad (12.11)$$

The Wald interval approximates the discrete binomial distribution by a normal distribution. See the rule of thumb in Section 3.4.1 for when this approximation is reasonable.

Package **binomSamSize** (?) has quite a few functions for computing the required sample size. The function **ciss.wald** uses the normal approximation. Argument **d** in the functions below is *half* the length of the confidence interval. Required sample sizes are computed for a prior estimate of the population proportion p^* of 0.2.

```
library(binomSamSize)
p_prior <- 0.2
n_prop_wald <- ciss.wald(p0=p_prior, d=0.1, alpha=0.05)
n_prop_agrcll <- ciss.agresticoull(p0=p_prior, d=0.1, alpha=0.05)
n_prop_wilson <- ciss.wilson(p0=p_prior, d=0.1, alpha=0.05)
```

The required sample sizes are 62, 58 and 60, for the Wald, Agresti-Coull and Wilson approximation of the binomial proportion confidence interval, respectively. The required sample size with function **ciss.wald** is one unit smaller than as computed with Equation (12.11), as shown in the code chunk below.

```
ceiling((qnorm(0.975)*sqrt(p_prior*(1-p_prior))/0.1)^2+1)
```

```
[1] 63
```

12.3 Statistical testing of hypothesis

The required sample size for testing a population mean with a two-sided alternative hypothesis can be computed with (?)

$$n = \frac{S^2(z)}{\Delta^2} (u_{\alpha/2} + u_{\beta})^2 \quad (12.12)$$

with Δ the smallest relevant difference of the population mean from the test value, α the tolerable probability of a type I error, i.e. the probability of rejecting the null hypothesis when the population mean is equal to the test value, β the tolerable probability of a type II error, i.e. the probability of not rejecting the null hypothesis when the population mean is not equal to the test value, $u_{\alpha/2}$ as before, and u_{β} the $(1 - \beta)$ quantile of the standard normal distribution. The quantity $1 - \beta$ is the power of a test: the probability of correctly rejecting the null hypothesis. For a one-sided test, $u_{\alpha/2}$ must be replaced by u_{α} .

In the next code chunk the sample size required for a given target power is computed with the standard normal distribution (Equation (12.12)), as well as with the t distribution using function `pwr.t.test` of package **pwr** (?)². This requires some iterative algorithm, as the degrees of freedom of the t distribution are a function of the sample size(s). The required sample size is computed for a one-sample test and a one-sided alternative hypothesis.

```
library(pwr)
sd <- 4; delta <- 1; alpha <- 0.05; beta <- 0.2
n_norm <- (sd/delta)^2*(qnorm(1-alpha)+qnorm(1-beta))^2
n_t <- pwr.t.test(
  d=delta/sd, sig.level=alpha, power=(1-beta), type="one.sample",
  alternative="greater")
```

In this example the required sample size computed with the t distribution is two units larger than that obtained with the standard normal distribution: 101 versus 99. Package **pwr** has various functions for computing the power of a test given the sample size, or reversely, the sample size for a given power, such as for the two independent samples t test, binomial test (for one proportion), test for two proportions, etc.

²The same result is obtained with function `power.t.test` of the **stats** package (?).

12.3.1 Sample size for testing a proportion

For testing a proportion a graph is computed with the power of a binomial test against the sample size. This is illustrated with a one-sided alternative $H_a : p > 0.20$ and a smallest relevant difference of 0.10.

```
p_test <- 0.20; alpha <- 0.10; delta <- 0.10
n <- 1:150
power <- k_min <- numeric(length=length(n))
for (i in 1:length(n)) {
  k_min[i] <- qbinom(p=1-alpha, size=n[i], prob=p_test)
  power[i] <- pbiniom(
    q=k_min[i], size=n[i], prob=p_test+delta, lower.tail=FALSE)
}
```

As can be seen in the **R** code, as a first step for each total sample size the smallest number of successes k_{\min} is computed at which the null-hypothesis is rejected. Then the binomial probability is computed of $k_{\min} + 1$ or more successes for a probability of success equal to $p_{\text{test}} + \Delta$. Note that there is no need to add 1 to k_{\min} as with argument `lower.tail=FALSE` the value specified with argument `q` is not included.

Figure 12.1 shows that the power does not increase monotonically with the sample size. The graph shows a saw-toothed behaviour. This is caused by the step-wise increase of the critical number of successes (k_{\min}) with the total sample size.

The required sample size can be computed in two ways. The first option is to compute the smallest sample size for which the power is larger than or equal to the required power $1 - \beta$. The alternative is to compute the smallest sample size for which the power is larger than or equal to $1 - \beta$ for all sample sizes larger than this.

```
n1 <- min(n[which(power > 1-beta)])
ind <- (power > 1-beta)
for(i in 1:length(n)) {
  if(ind[length(n)-i]==FALSE)
    break
}
```

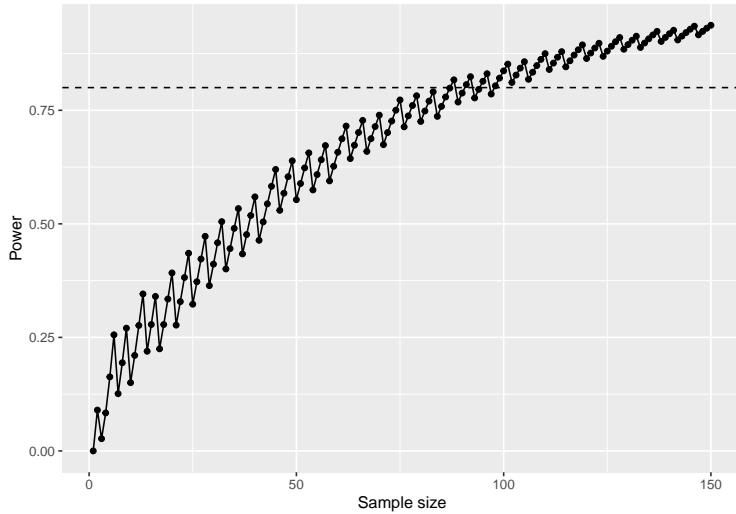


Figure 12.1: Power of right-tail binomial test (test-proportion: 0.2; significance level: 0.10).

```
n2 <- n[length(n)-i+1]
```

The smallest sample size at which the desired level of 0.8 is reached is 88. However, as can be seen in Figure 12.1 for sample sizes 89, 90, 93, 94 and 97 the power drops below the desired level of 0.80. The smallest sample size at which the power stays above the level of 0.8 is 98.

Alternatively, we may use function `pwr.p.test` of package `pwr`. This is an approximation, using an arcsine transformation of proportions. The first step is to compute Cohens's h , which is a measure of the distance between two proportions: $h = 2 \arcsin(\sqrt{p_1}) - 2 \arcsin(\sqrt{p_2})$. This can be done with function `ES.h`. The value of h must be positive, which is achieved when the proportion assigned to argument `p1` is larger than the proportion assigned to argument `p2`.

```
h <- ES.h(p1=0.30, p2=0.20)
n_approx <- pwr.p.test(
```

```
h, power=(1-beta), sig.level=alpha, alternative="greater")
```

The approximated sample size equals 84, which is somewhat smaller than the required sample sizes computed above.

Exercises

1. Write an **R** script to compute the required sample size, given a requirement in terms of half the length of the confidence interval of a proportion. Use a normal approximation for computing the confidence interval. Use a range of values for half the length of the interval: $d = (0.05, 0.1, \dots, 0.45)$. Use a prior (anticipated) proportion of 0.2, and a significance level of 0.95. Plot the required sample size against d . Explain what you see. Why is it needed to provide a value for the prior proportion?
2. Do the same for a single value for half the width of the interval of 0.2, and a range of values for the prior proportion $p^* = (0.05, 0.10, \dots, 0.50)$. Explain what you see. Why is it not needed to compute the required sample size for prior proportions > 0.5 ?

12.4 Accounting for design effect

The required sample sizes computed in the previous sections are all for simple random sampling in combination with the π estimator of the population mean. But what is the required sample size for other types of sampling design, such as stratified (simple) random sampling, systematic random sampling, two-stage cluster random sampling and cluster random sampling? Broadly speaking we expect that with stratified random sampling and systematic random sampling the sampling variance of the estimator of the mean will be smaller than with simple random sampling of the same number of units, whereas with two-stage cluster random sampling and cluster random sampling we expect larger sampling variances. Therefore, reversely, for the first two types of sampling design we expect that the sample size required to achieve the same level of accuracy or confidence will be smaller than with simple random sampling, and for the latter two design types this sample size will be larger. The design effect is commonly

quantified by the ratio of two sampling variances of the estimated mean (?)³:

$$de(p, \hat{\bar{z}}) = \frac{V_p(\hat{\bar{z}})}{V_{SI}(\hat{\bar{z}}_\pi)} = \frac{V_p(\hat{\bar{z}})}{S^2(z)/n}, \quad (12.13)$$

with $V_p(\hat{\bar{z}})$ the sampling variance of an estimator (π estimator, regression estimator) of the population mean with sampling design p and $V_{SI}(\hat{\bar{z}}_\pi)$ the sampling variance of the π estimator of the population mean with simple random sampling. Given an estimate of this design effect, the required sample size for a more complex sampling strategy (combination of sampling design and estimator), given a constraint on the standard error or the half-length of a confidence interval, can be computed by

$$n_{req}(p, \hat{\bar{z}}) = \sqrt{de(p, \hat{\bar{z}})} n_{req}(SI, HT). \quad (12.14)$$

12.5 Bayesian sample size determination

A serious drawback of the classical frequentist approach of computing the required sample size explained in the previous sections is that the required sample sizes are sensitive to the design parameters S^* , p^* and cv^* . We are rather uncertain about these parameters, and therefore it is attractive to replace a single value for these parameters by a probability distribution. This leads to a different statistical approach of computing the required sample size, the Bayesian approach. This Bayesian approach also offers the possibility of accommodating existing information about the population mean or proportion. In this section I show how this approach can be used to compute the required sample size for estimating a population mean or proportion.

But before going into details, let me explain the basics of the Bayesian approach of statistical inference. In the previous sections the statistical inference was from the frequentist perspective. How does the frequency distribution of the estimator of the population mean look like if we repeat the selection of a sample with a given sampling design over and over? Is the mean of this frequency

³It can also be quantified by the ratio of two standard errors. Then there is no need to take the square root of the design effect, as done in Equation (12.14), to compute the required sample size for a more complex design, given a constraint on the standard error or the half-length of a confidence interval.

distribution, referred to as the sampling distribution, equal to the population mean, and what is the variance of this sampling distribution?

The Bayesian approach is fundamentally different. The frequency distribution of the frequentist approach is replaced by a probability distribution of the population mean reflecting our *belief* about the population mean. Note that expressing our belief in terms of a probability distribution implies that in the Bayesian approach, contrary to the frequentist approach, the population mean is a random variable. Whereas in the frequentist approach it is incorrect to say that the probability that the population mean is inside the 95% confidence interval equals 95% (see Section 3.4), this is perfectly fine in the Bayesian approach. The term confidence interval is replaced by the term *credible interval* to underline the fundamental different meaning of the interval.

The first step in the Bayesian approach of statistical inference is to postulate a *prior distribution* for the population parameter of interest. This prior distribution expresses our belief and uncertainty about the parameter before the sample data are taken into account.

The next step is to formalize a theory about the data. This boils down to making an assumption about the type of distribution function of the data. Can we safely assume that the data follow a normal or a binomial distribution? Once the type of distribution has been specified, we can write an equation for the probability of the data *as a function of the parameter*. This function is referred to as the *likelihood function*.

The final step is to revise our prior belief about the population parameter of interest, using the data and our theory about the data as expressed in the likelihood function. This results in the *posterior distribution* of the parameter. Our revised or updated belief is computed with Bayes' rule:

$$f(\theta|\mathbf{z}) = \frac{f(\theta)f(\mathbf{z}|\theta)}{f(\mathbf{z})}, \quad (12.15)$$

with $f(\theta|\mathbf{z})$ the posterior distribution, i.e. the probability density⁴ of the parameter given the sample data, $f(\theta)$ our prior belief in the parameters specified by a probability distribution (prior distribution), $f(\mathbf{z}|\theta)$ the likelihood of the data, and $f(\mathbf{z})$ the probability distribution of the data.

⁴I assume here that the parameter of interest θ is a continuous random variable.

12.5.1 Bayesian criteria for sample size computation

Equation (12.15) shows that the posterior distribution of the population parameter of interest depends on the probability distribution of the new data $f(\mathbf{z})$. The problem is that these new data are not yet known. We are designing a sample, and the data yet are to be collected, so at first glance this might seem an unsolvable problem. However, what we could do is to simulate with the prior probability density function a large number of possible vectors with n data. Each data vector is then used to update the prior to the posterior, using Bayes' rule (Equation (12.15)). For each posterior either the length of the highest posterior density (HPD) interval with a coverage probability of $1 - \alpha$ is computed, or reversely, the coverage probability of the HPD interval of length l_{\max} . Finally the average of the lengths of the HPD intervals, or the average of the coverage probabilities is computed, and these averages are compared with our precision requirement. If the average length is larger than l_{\max} , or the coverage probability of intervals of length l_{\max} is smaller than $1 - \alpha$, then we must increase n and repeat the whole procedure until our precision requirement is met. Simulation is one option to compute the sample size, (partly) analytical approaches are also available.

More formally, the procedure is as follows. The prior probability density function on the population parameter(s) θ is used to compute for a given sample size n the *predictive* distribution of the data:

$$f(\mathbf{z}|n) = \int_{\Theta} f(\mathbf{z}|\theta, n) f(\theta) d\theta \quad (12.16)$$

with Θ the parameter space for θ containing all possible values of θ . This predictive distribution is also named the *preposterior* distribution, stressing that the data are not yet accounted for in the distribution.

Even if θ would be fixed, we do not have only one vector \mathbf{z} with n data values but a probability distribution, from which we can simulate possible data vectors, referred to as the data space \mathcal{Z} . In case of a binomial probability and sample size n , the data space \mathcal{Z} (in the form of the number of observed successes given sample size n) can be written as the set $\{0, 1, \dots, n\}$, i.e. one vector of length n with all failures, n vectors of length n with one success, $\binom{n}{2}$ vectors with two successes, etc. Each data vector is associated with a probability density (for continuous data) or probability mass (for discrete data). As a consequence, we do not have only one posterior distribution function $f(\theta|\mathbf{z})$, but as many as we

have data vectors in the data space. For each posterior distribution function the coverage of the HPD interval of a given length can be computed, or reversely, the length of the HPD interval for a given coverage. This leads to various criteria for computing the required sample size, among which are the average length criterion, the average coverage criterion and the worst outcome criterion (?, ?).

12.5.1.1 Average length criterion

For a fixed posterior HPD interval coverage of $100(1 - \alpha)\%$ the smallest sample size n is determined such that

$$\int_{\mathcal{Z}} l(\mathbf{z}, n) f(\mathbf{z}|n) d\mathbf{z} \leq l_{\max} , \quad (12.17)$$

where $f(\mathbf{z}|n)$ is the predictive distribution of the data (Equation (12.16)), and $l(\mathbf{z}, n)$ is the length of the $100(1 - \alpha)\%$ HPD interval for data \mathbf{z} and sample size n , obtained by solving

$$\int_v^{v+l(\mathbf{z}, n)} f(\theta|\mathbf{z}, n) d\theta = 1 - \alpha , \quad (12.18)$$

for $l(\mathbf{z}, n)$, for each possible data set $\mathbf{z} \in \mathcal{Z}$. $f(\theta|\mathbf{z}, n)$ is the posterior density of the population parameter of interest given the data \mathbf{z} and sample size n . ALC ensures that the average length of $100(1 - \alpha)\%$ posterior HPD intervals, weighted by $f(\mathbf{z}|n)$, is at most l_{\max} .

12.5.1.2 Average coverage criterion

For a fixed posterior HPD interval of length l_{\max} the smallest sample size n is determined such that

$$\int_{\mathcal{Z}} \left\{ \int_v^{v+l_{\max}} f(\theta|\mathbf{z}, n) d\theta \right\} f(\mathbf{z}|n) d\mathbf{z} \geq 1 - \alpha . \quad (12.19)$$

The average coverage criterion (ACC) ensures that the average coverage of HPD intervals of length l_{\max} is at least $1-\alpha$. The integral inside the curly brackets is the integral of the posterior density of the population parameter of interest over the HPD interval $(v, v + l_{\max})$, given a data vector \mathbf{z} of size n . The mean

of this integrated posterior density of the parameter of interest θ is obtained by multiplying the integrated density with the predictive probability of the data, and integrating over all possible data sets in \mathcal{Z} .

12.5.1.3 Worst outcome criterion

Neither ALC nor ACC guarantee that for a particular data set \mathbf{z} the criterion is met, as these two criteria are defined as averages over all possible data sets in \mathcal{Z} . A more conservative sample size can be computed by requiring that for all data sets \mathcal{Z} both criteria are met. ? modified this criterion by restricting the data sets to a subset \mathcal{W} of most likely data sets. The criterion thus obtained is referred to as the modified worst outcome criterion, but I refer to it shortly as the worst outcome criterion (WOC). So the criterion is

$$\inf_{\mathbf{z} \in \mathcal{W}} \left\{ \int_v^{v+l(\mathbf{z}, n)} f(\theta | \mathbf{z}, n) d\theta \right\} \geq 1 - \alpha . \quad (12.20)$$

The smallest sample size satisfying this condition is used as the sample size. For instance, if the 95% most likely data sets are chosen as subspace \mathcal{W} , WOC guarantees that there is 95% assurance that the length of the 100(1 - α)% posterior HPD intervals will be at most l_{\max} . The fraction of most likely data sets in subspace \mathcal{W} is referred to as the worst level.

12.5.2 Mixed Bayesian-likelihood approach

Besides the fully Bayesian approach, ? describe a mixed Bayesian-likelihood approach for determining the sample size. In the mixed Bayesian-likelihood approach of sample size determination the prior distribution of the parameter(s) is only used to derive the predictive distribution of the data (Equation (12.16)), not for deriving the posterior distributions of the parameter of interest for each data vector. For analysis of the posterior distribution, an uninformative prior is therefore used.

This mixed approach is of interest when, after the data have been collected, we prefer to estimate the population mean from these data only, using the frequentist approach described in the previous sections. For instance, consider the situation where we have legacy data, but we would like to collect more data so that we will be more confident about the (current) population mean once these new are collected. The legacy data are to construct a prior distribution.

If we have doubts about the quality of the legacy data, this mixed Bayesian-likelihood approach can be a good option – we are willing to use these legacy data to plan the sampling, but not to make statements about the population.

No closed formula for computing the required sample size exists for this approach because the posterior density function $f(\theta|z, n)$ is not a well-defined distribution as before. However, the required sample size still can be approximated by simulation.

12.5.3 Estimation of population mean

The three criteria (ALC, ACC and WOC) described above are now used to compute the required sample size for estimating the population mean, assuming that the data come from a normal distribution. As we are uncertain about the population standard deviation σ in Equation (12.7), a prior distribution is assigned to this parameter. It is convenient to assign a gamma distribution as a prior distribution to the *reciprocal* of the population variance, referred to as the precision parameter $\lambda = 1/\sigma^2$. More precisely, a prior *bivariate* normal-gamma distribution is assigned to the population mean and the precision parameter⁵. With this prior distribution, the *posterior* distribution of the population mean is fully defined, i.e. both the type of distribution and its parameters are known. The prior distribution is so-called *conjugate* with the normal distribution.

The gamma distribution has two parameters a and b . Figure 12.2 shows the gamma distribution for $a = 5$ and $b = 100$.

```
a <- 5; b <- 100
x <- seq(from=0, to=0.5, length=1000)
dg <- dgamma(x=x, shape=a, scale=1/b)
plot(x=x, y=dg, type="l", ylab="Density", xlab="Precision")
```

The mean of the precision parameter λ is given by a/b , and its standard deviation by $\sqrt{a/b^2}$

The normal-gamma prior is used to compute the predictive distribution for the data. For ACC the required sample size can then be computed with (?)

⁵This is equal to a normal-inverse gamma distribution to the population mean and population variance.

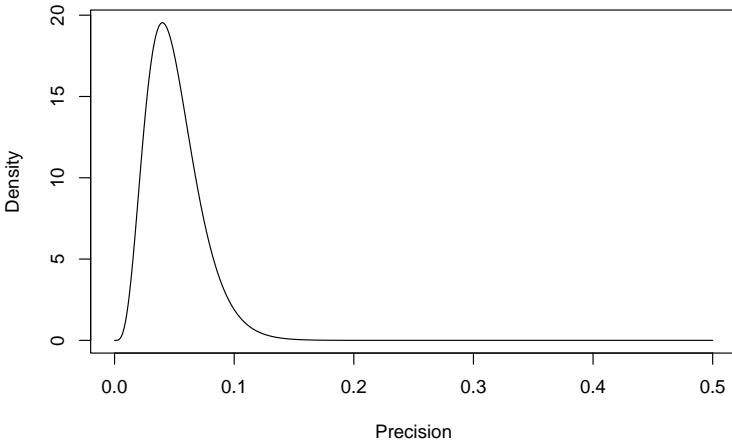


Figure 12.2: Prior gamma distribution for the precision parameter for a shape parameter $a = 5$ and a scale parameter $1/b = 1/100$.

$$n = \frac{4b}{a l_{\max}^2} t_{2a;1-\alpha/2}^2 - n_0 , \quad (12.21)$$

with t_{2a}^2 the squared $(1 - \alpha/2)$ quantile of the (usual, i.e. neither shifted, nor scaled) t distribution with $2a$ degrees of freedom, and n_0 the number of prior points. The prior sample size n_0 is only relevant if we have prior information about the population mean and an informative prior is used for this population mean. If we have no information about the population mean a non-informative prior is used, and n_0 equals 0. Note that as a/b is the prior mean of the inverse of the population variance, Equation (12.21) is similar to Equation (12.7). The only difference is that a quantile from the standard normal distribution is replaced by a quantile from a t distribution with $2a$ degrees of freedom.

No closed-form formula exists for computing the smallest n satisfying ALC, but the solution can easily be found by a bisectional search algorithm (?).

Package **SampleSizeMeans** (?) is used to compute Bayesian required sample sizes, using both criteria (ACC and ALC), for the fully Bayesian and the mixed Bayesian-likelihood approach. The gamma distribution plotted in Figure 12.2 is used as a prior distribution for the precision parameter λ . As a reference, also the frequentist required sample size is computed.

Table 12.1: Required sample sizes for estimating a normal mean computed with three criteria for fully Bayesian and mixed Bayesian-likelihood (MBL) approach. Freq is required sample size computed with frequentist approach.

Freq	ALC	ALC-MBL	ACC	ACC-MBL	WOC	WOC-MBL
77	92	93	100	102	194	201

```
library(SampleSizeMeans)
lmax <- 2
n_freq <- mu.freq(len=lmax, lambda=a/b, level=0.95)
n_alc <- mu.alc(len=lmax, alpha=a, beta=b, n0=0, level=0.95)
n_alcmb <- mu.mblalc(len=lmax, alpha=a, beta=b, level=0.95)
n_acc <- mu.acc(len=lmax, alpha=a, beta=b, n0=0, level=0.95)
n_accmb <- mu.mblacc(len=lmax, alpha=a, beta=b, level=0.95)
n_woc <- mu.modwoc(
  len=lmax, alpha=a, beta=b, n0=0, level=0.95, worst.level=0.95)
n_wocmb <- mu.mblmodwoc(
  len=lmax, alpha=a, beta=b, level=0.95, worst.level=0.95)
```

Table 12.1 shows that all six required sample sizes are larger than the frequentist required sample size. This makes sense as the frequentist approach does not account for uncertainty in the population variance parameter. The mixed approach leads to slightly larger required sample sizes than the fully Bayesian approach. This is because in the mixed approach the prior distribution of the precision parameter is not used. Apparently, we do not lose much information by ignoring this prior. With WOC the required sample sizes are about twice the sample sizes obtained with the other two criteria, but this depends of course on the size of the subspace \mathcal{W} . If, for instance the 80% most likely data sets are chosen as subspace \mathcal{W} , the required sample sizes are much smaller.

```
n_woc80 <- mu.modwoc(
  len=lmax, alpha=a, beta=b, n0=0, level=0.95, worst.level=0.80)
n_wocmb180 <- mu.mblmodwoc(
  len=lmax, alpha=a, beta=b, level=0.95, worst.level=0.80)
```

The required sample sizes with this criterion are 124 and 128 using the fully

Bayesian and mixed Bayesian-likelihood approach, respectively.

12.5.4 Estimation of a population proportion

The same criteria can be used to estimate the proportion of a population, or in case of an infinite population the areal fraction, satisfying some condition (?). With simple random sampling this boils down to estimating the probability-of-success parameter p of a binomial distribution. In this case the space of possible outcomes \mathcal{Z} is the number of successes, which is discrete: $\mathcal{Z} = \{0, 1, \dots, n\}$ with n the sample size.

The conjugate prior for the binomial likelihood is the beta distribution.

$$p \sim \frac{1}{B(c, d)} \pi^{c-1} (1 - \pi)^{d-1}, \quad (12.22)$$

where $B(c, d)$ is the beta function. The two parameters c and d correspond to the number of “successes” and “failures” in the problem context. The larger these numbers, the more the prior information, and the more sharply defined the probability distribution. The plot below shows this distribution for $c = 0.6$ and $d = 2.4$.

```
c <- 0.6; d <- 2.4
x <- seq(from=0, to=1, length=1000)
dbt <- dbeta(x=x, shape1=c, shape2=d)
plot(x=x, y=dbt, type="l", ylab="Density", xlab="Proportion")
```

The mean of the binomial proportion equals $c/(c+d)$, and its standard deviation $\sqrt{cd/[(c+d+1)(c+d)^2]}$.

The preposterior marginal distribution of the data is the beta-binomial distribution

$$f(z|n) = \binom{n}{z} \frac{B(z+c, n-z+d)}{B(c, d)}, \quad (12.23)$$

and for a given number of successes z out of n trials the posterior distribution of p equals

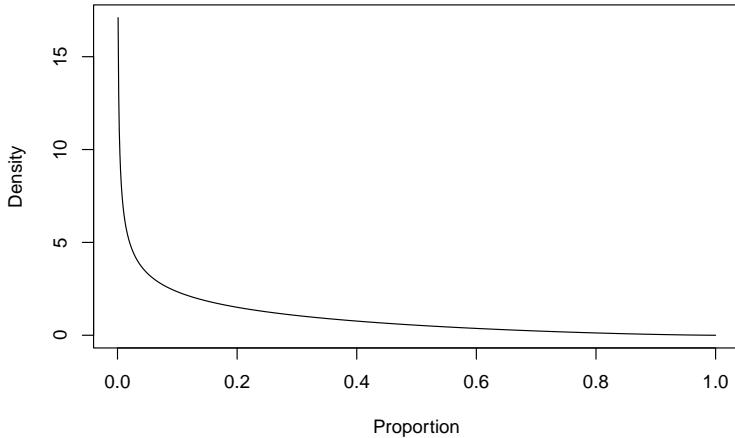


Figure 12.3: Prior beta distribution for the binomial proportion, for a beta function $B(0.6, 2.4)$.

$$f(p|z, n, c, d) = \frac{1}{B(z + c, n - z + d)} p^{z+c-1} (1-p)^{n-z+d-1}. \quad (12.24)$$

For the binomial parameter, criterion ALC (Equation (12.17)) can be written as

$$\sum_{z=0}^n l(z, n) f(z, n) \leq l_{\max}. \quad (12.25)$$

To compute the smallest n satisfying this condition, for each value of z and each n , $l(z, n)$ must be computed so that

$$\int_v^{v+l(z,n)} f(p|z, n, c, d) dp = 1 - \alpha. \quad (12.26)$$

with v the lower bound of the HPD credible set given the sample size and observed number of successes z .

For the binomial parameter, criterion ACC (Equation (12.19)) can be written as

$$\sum_{z=0}^n \Pr\{p \in (v, v + l_{\max})\} f(z, n) \geq 1 - \alpha , \quad (12.27)$$

with

$$\Pr\{p \in (v, v + l_{\max})\} \propto \int_v^{v+l_{\max}} p^z (1-p)^{n-z} f(p) dp , \quad (12.28)$$

with $f(p)$ the prior density of the binomial parameter.

For more details about ACC and ALC, and about how the required sample size can be computed with WOC in case of the binomial parameter p , I refer to ?.

Required sample sizes for the average length criterion (ALC), the average coverage criterion (ACC) and worst outcome criterion (WOC) described in the previous section, using the fully Bayesian approach or the mixed Bayesian-likelihood approach can be computed with package **SampleSizeBinomial**, available at <http://www.medicine.mcgill.ca/epidemiology/Joseph/software/Bayesian-Sample-Size.html>. This package is used to compute the required sample sizes using the beta distribution shown in Figure 12.3 as a prior for the population proportion. Note that argument **len** of the various functions of package **SampleSizeBinomial** specify the total length of the confidence interval, not *half* the length as used in function **ciss.wald**.

```
library(SampleSizeBinomial)
n_alc <- prop.alc(
  len=0.2, alpha=c, beta=d, level=0.95, exact=TRUE)$n
n_alcmbl <- prop.mblalc(
  len=0.2, alpha=c, beta=d, level=0.95, exact=TRUE)$n
n_acc <- prop.acc(
  len=0.2, alpha=c, beta=d, level=0.95, exact=TRUE)$n
n_accmbl <- prop.mblacc(
  len=0.2, alpha=c, beta=d, level=0.95, exact=TRUE)$n
n_woc <- prop.modwoc(
  len=0.2, alpha=c, beta=d, level=0.95, exact=TRUE,
  worst.level=0.80)$n
n_wocmbl <- prop.mblmodwoc(
```

Table 12.2: Required sample sizes for estimating a binomial proportion, computed with three criteria for fully Bayesian and mixed Bayesian-likelihood (MBL) approach. Freq is the required sample size computed with frequentist approach.

Freq	ALC	ALC-MBL	ACC	ACC-MBL	WOC	WOC-MBL
62	33	38	50	53	80	81

```

len=0.2, alpha=c, beta=d, level=0.95, exact=TRUE,
worst.level=0.80)$n
library(binomSamSize)
n_freq <- ciss.wald(p0=c/(c+d), d=0.1, alpha=0.05)

```


Chapter 13

Model-based optimisation of probability sampling designs

In Chapter 10 on model-assisted estimation I explained how a linear regression model or a non-linear model obtained with a machine learning algorithm, can be used to increase the precision of design-based estimates of the population mean or total using the data collected by a given probability sampling design. In this chapter I explain how a model of the study variable can be used at an earlier stage, to optimise probability sampling designs. I show how a model can be used to choose between alternative sampling design types, for instance between systematic random sampling and two-stage cluster random sampling, as well as how a model can be used to optimise the sample size of a given sampling design type, for instance, the number of primary and secondary units with two-stage cluster random sampling. In the final section of this chapter I explain how a model can be used to optimise spatial strata for stratified simple random sampling.

The models used in this chapter are all geostatistical models of the spatial variation. Chapter 22 is an introduction to geostatistical modelling. Several geostatistical concepts explained in that chapter are needed here to predict the sampling variance. For an explanation of these concepts, I refer to that chapter.

A general geostatistical model of the spatial variation is

$$\begin{aligned} Z(\mathbf{s}) &= \mu(\mathbf{s}) + \epsilon(\mathbf{s}) \\ \epsilon(\mathbf{s}) &\sim \mathcal{N}(0, \sigma^2) \\ \text{Cov}(\epsilon(\mathbf{s}), \epsilon(\mathbf{s}')) &= C(\mathbf{h}), \end{aligned} \tag{13.1}$$

with $Z(\mathbf{s})$ the study variable at point \mathbf{s} , $\mu(\mathbf{s})$ the mean at point \mathbf{s} , $\epsilon(\mathbf{s})$ the residual at point \mathbf{s} , and $C(\mathbf{h})$ the covariance of the residuals at two points separated by vector $\mathbf{h} = \mathbf{s} - \mathbf{s}'$. Note that contrary to the mean μ the variance of the residuals σ^2 is assumed to be constant.

The model of the spatial variation has several parameters. In case of a model in which the mean is a linear combination of covariates, these are the regression coefficients associated with the covariates, and the parameters of a semivariogram describing the spatial dependence of the residuals. A semivariogram is a plot of half the expectation of the squared difference of the study variable (in this case the residuals), referred to as the semivariance, against the distance between two points (Chapter 22). Use of the model for prediction of the sampling variance of a design-based estimator of the population mean requires knowledge of these parameters. When data are available collected from the study area of interest, these data can be used to estimate these parameters. If no such data are available, we must make a best guess, based on data collected in other areas. In all cases I recommend to keep the model as simple as possible.

13.1 Model-based optimisation of sampling design type and sample size

In Chapter 12 I presented methods and formulas for computing the required sample size given various constraints on the quality of the result. These required sample sizes are for simple random sampling. For other types of sampling design, the required sample size can be approximated by multiplying the required sample size for simple random sampling with the design effect, see Section 12.4. An alternative is to use a model of the spatial variation to predict the sampling variance of the estimator of the mean for the type of sampling design under study and a range of sample sizes, plotting the predicted variance (or standard error) against the sample size, and using this plot inversely to derive the required sample size given a constraint on the sampling variance (standard error).

The computed sample size applies to a given sampling design type. So, for instance, for stratified random sampling the sample size is computed for a given stratification and sample size allocation scheme, for cluster random sampling for given clusters and number of cluster draws, and for two-stage cluster random sampling for given primary sampling units (psu's), number of psu draws and number of secondary sampling units (ssu's) selected per psu draw. However, the model can also be used to optimise these sampling design parameters. For stratified random sampling the optimal allocation can be computed by predicting the population variances within strata and using the predicted population variances per stratum in Equation (4.16), and even the stratification can be optimised (Section 13.2)). If we have a cost model, for cluster random sampling the size and shape of the clusters, and the number of cluster draws can be optimised, and for two-stage cluster random sampling the size and shape of the primary sampling units, the number of psu draws and number of ssu's per psu draw can be optimised.

Model-based prediction of the sampling variance can also be useful to compare alternative types of sampling design at equal total costs or equal variances of the estimated population mean or total. For instance, to compare systematic random sampling, leading to good spatial coverage, and two-stage cluster random sampling, resulting in spatial clusters of observations.

Three approaches for model-based prediction of the sampling variance of a design-based estimator of the population mean (or total) are described, the analytical approach (Section 13.1.1), the geostatistical simulation approach (Section 13.1.2) and the Bayesian approach (Section 13.1.3). In the analytical approach we assume that the mean, $\mu(\mathbf{s})$ in Equation (13.2), is everywhere the same, so we assume no spatial trend. This assumption is relaxed in the geostatistical simulation approach. This approach can also be used to predict the sampling variance using a model in which the mean is a linear combination of covariates, and to predict the sampling variance of the estimator of the mean of trans-Gaussian variables, i.e. random variables that can be transformed to a Gaussian variable.

The predicted sampling variances of the estimated population mean obtained with the analytical and geostatistical simulation approach are conditional on the model of the spatial variation. Uncertainty about this model is not accounted for. On the contrary, in the Bayesian approach we do account for our uncertainty about the assumed model, and we analyze how this uncertainty propagates to the sampling variance of the estimator of the mean.

13.1.1 Analytical approach

In the analytical approach the sampling variance of the estimator of the mean is derived from mean semivariances within the study area and mean semivariances within the sample. These mean semivariances are a function of the separation distance between the points, see Chapter 22 for a definition of the semivariance of two random variables.

The sampling variance of a design-based estimator of the population mean can be predicted by (?, ?)

$$E_\xi\{V_p(\hat{z})\} = \bar{\gamma} - E_p(\lambda' \Gamma_S \lambda), \quad (13.2)$$

where $E_\xi(\cdot)$ is the statistical expectation over realisations from the model ξ , $E_p(\cdot)$ is the statistical expectation over repeated sampling with sampling design p , $V_p(\hat{z})$ is the variance of the estimator of the mean over repeated sampling with sampling design p , $\bar{\gamma}$ is the mean semivariance of the random variable at two randomly selected points in the area, λ is the vector of design-based weights of the units of a sample selected with design p , and Γ_S is the matrix of semivariances between the units of a sample S selected with design p .

The mean semivariance $\bar{\gamma}$ is a model-based prediction of the population variance (spatial variance):

$$\bar{\gamma} = E_\xi\{\sigma^2(z)\}. \quad (13.3)$$

The mean semivariance $\bar{\gamma}$ can be calculated by discretising the area by a fine square grid, and computing the matrix with geographical distances between the discretisation nodes, transforming this into a semivariance matrix, and computing the average of all elements of the semivariance matrix. The second term $E_p(\lambda' \Gamma_S \lambda)$ can be evaluated by Monte-Carlo simulation, repeatedly selecting a sample according to design p , calculating $\lambda' \Gamma_S \lambda$, and averaging.

The semivariance at zero distance (same point) is 0, so on the diagonal of a semivariance matrix we have zeroes. If a semivariogram with nugget is assumed, these zeroes on the diagonal must be replaced by the nugget to compute $\bar{\gamma}$. The same holds for the diagonal zeroes in Γ_S .

This generic procedure is still computationally demanding, but it is the only option for complex spatial sampling designs. For basic sampling designs the

general formula can be worked out. For simple random sampling, the sampling variance can be predicted by

$$E_\xi\{V_{SI}(\hat{\bar{z}})\} = \bar{\gamma}/n , \quad (13.4)$$

and for stratified simple random sampling by

$$E_\xi\{V_{STS1}(\hat{\bar{z}})\} = \sum_{h=1}^H w_h^2 \bar{\gamma}_h / n_h , \quad (13.5)$$

with H the number of strata, w_h the stratum weight (relative size), $\bar{\gamma}_h$ is the mean semivariance of stratum h , and n_h the number of sampling points of stratum h . For systematic random sampling (sampling on a randomly placed grid), the variance can be predicted by

$$E_\xi\{V_{SY}(\hat{\bar{z}})\} = \bar{\gamma} - E_{SY}(\bar{\gamma}_{SY}) , \quad (13.6)$$

with $E_{SY}(\bar{\gamma}_{SY})$ the expectation over repeated systematic sampling of the mean semivariance within the systematic sample (grid). With systematic random sampling the number of grid points within the study area can vary among samples, as well as the spatial configuration of the points (Chapter 5). Therefore multiple systematic random samples must be selected, and the average of the mean semivariance within the systematic sample must be computed.

The analytical approach is illustrated with the data of agricultural field Leest (?). Nitrate-N (NO_3-N) in kg/ha in the layer 0 - 90 cm below surface, using a standard soil density of 1500 kg/m³, is measured at 30 points. These data are used to compute a sample semivariogram using the method-of-moments, see Chapter 22. A spherical model without nugget is fitted to the sample semivariogram using function `fit.variogram`. The numbers in this plot are the numbers of pairs of points used to compute the semivariances.

```
library(gstat)
mydata <- read.csv("data/Leest05.csv")
coordinates(mydata) <- ~Easting+Northing
vg <- variogram(N~1, data=mydata)
vglm_MoM <- fit.variogram(
```

```
vg, model=vgm(model="Sph", psill=2000, range=20)
plot(vg, vgm_MoM, plot.numbers=TRUE)
```

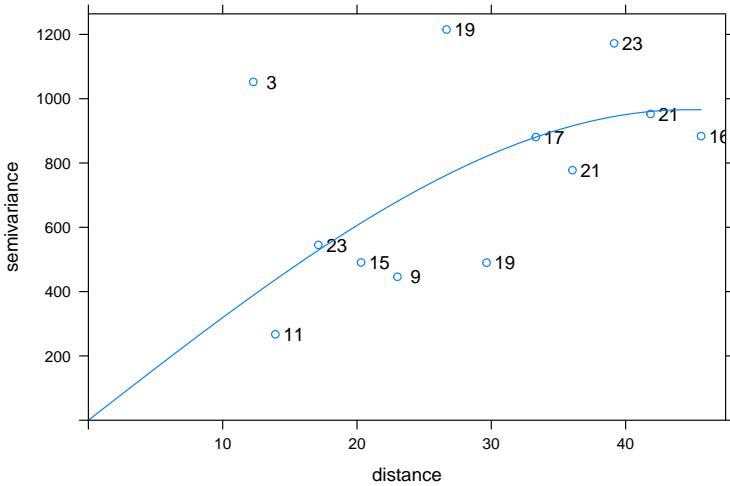


Figure 13.1: Sample semivariogram and fitted spherical model for NO₃-N in agricultural field Leest. Numbers are numbers of pairs used in computing semivariances.

The few data makes that the sample semivariogram is very noisy. For the moment I ignore my uncertainty about the semivariogram parameters. In Section 13.1.3 I show how we can account for our uncertainty about the semivariogram parameters in model-based prediction of the sampling variance. A spherical semivariogram model without nugget is fitted to the sample semivariogram, i.e. the intercept is 0. The fitted range of the model is 45 m, and the fitted sill equals 966. The fitted semivariogram is used to predict the sampling variance for three sampling designs: simple random sampling, stratified simple random sampling, and systematic random sampling. The sample size is 25 points. For systematic random sampling the number of points varies among the samples. For this sampling design the *expected* sample size is 25 points.

For simple random sampling we must compute the mean semivariance within the field (Equation (13.4)). This mean semivariance is approximated by discretising the field by a square grid of 2000 points, computing the 2000 × 2000

matrix with distances between all pairs of discretisation nodes, transforming this distance matrix into a semivariance matrix using function `variogramLine` of package `gstat` (?) , and finally averaging the semivariances. Note that in this case we do not need to replace the zeroes on the diagonal of the semivariance matrix by the nugget, as a model without nugget is fitted. The shape file is read with function `readOGR` of package `rgdal` (?), resulting in an object of class `SpatialPolygonsDataFrame`. The projection attributes of this object are removed with function `proj4string` to avoid an error message when running function `spsample`.

```
library(rgdal)
shpField <- readOGR(dsn="data", layer="Leest5", verbose=FALSE)
proj4string(shpField) <- NA_character_
mygrid <- spsample(
  shpField, type="regular", n=2000, offset=c(0.5, 0.5)) %>%
  as(., "data.frame")
H <- as.matrix(dist(mygrid))
G <- variogramLine(vgm_MoM, dist_vector=H)
m_semivar_field <- mean(G)
n <- 25
Exi_V_SI <- m_semivar_field/n
```

The model-based prediction of the sampling variance of the estimator of the mean with this design equals 35.

The strata of the stratified simple random sampling design are compact geographical strata of equal size (Section 4.6). The number of geostrata is equal to the sample size, 25 points, so that we have one point per stratum. To predict the sampling variance we must compute the mean semivariances within the geostrata, see Equation (13.5). Note that the stratum weights are constant as the strata have equal size: $w_h = 1/25$.

```
library(spcosa)
gridded(mygrid) <- ~x1+x2
mygeostrata <- stratify(
  mygrid, nStrata=n, equalArea=TRUE, nTry=10) %>%
  as(., "data.frame")
m_semivar_geostrata <- numeric(length=n)
```

```

for (i in 1:n) {
  ids <- which(mygeostrata$stratumId==(i-1))
  mysubgrd <- mygeostrata[ids,]
  H_geostratum <- as.matrix(dist(mysubgrd[,c(2,3)]))
  G_geostratum <- variogramLine(vgm_MoM, dist_vector=H_geostratum)
  m_semivar_geostrata[i] <- mean(G_geostratum)
}
Exi_V_STSI <- sum(m_semivar_geostrata)/n^2

```

The model-based prediction of the sampling variance of the estimator of the mean with this design equals 13.6, which is much smaller than with simple random sampling. The large stratification effect can be explained by the assumed strong spatial structure of NO₃-N in the agricultural field and the improved geographical spreading of the sampling points.

To predict the sampling variance for systematic random sampling with an expected sample size of 25 points, we must compute the expectation of the mean semivariance within the systematic sample (Equation (13.6)). This expectation is approximated by selecting a large number of systematic random samples, computing the mean semivariance for each sample, and averaging. This average of mean semivariances within a systematic sample is subtracted from the mean semivariance within the field.

```

set.seed(314)
m_semivar_SY <- numeric(length=100)
for (i in 1:100) {
  mySYsample <- spsample(x=mygrid, n=n, type="regular") %>%
    as(., "data.frame")
  H_SY <- as.matrix(dist(mySYsample))
  G_SY <- variogramLine(vgm_MoM, dist_vector=H_SY)
  m_semivar_SY[i] <- mean(G_SY)
}
Exi_V_SY <- m_semivar_field-mean(m_semivar_SY)

```

The model-based prediction of the sampling variance of the estimator of the mean with this design equals 8, which is smaller than that of stratified simple random sampling, which can be explained by the improved geographical spreading of the sampling points with systematic random sampling as compared to

stratified simple random sampling with compact geographical strata.

13.1.1.1 Bulking soil aliquots into a composite sample

If the soil aliquots collected at the points of the stratified random sample are bulked into a composite, as is usually done in soil testing of agricultural fields, the procedure for predicting the variance of the estimator of the mean is slightly different. Only the composite sample is analyzed in a laboratory on NO₃-N, not the individual soil aliquots. This implies that the contribution of the measurement error to the total uncertainty about the population mean is larger. To predict the sampling variance in this situation, we need the semivariogram of errorless measurements of NO₃-N, i.e. of the true NO₃-N contents of soil aliquots collected at points. The sill of this semivariogram will be smaller than the sill of the semivariogram of measured NO₃-N data. A simple option is to subtract an estimate of the measurement error variance from the semivariogram of measured NO₃-N data that contain a measurement error. So the measurement error variance is subtracted from the nugget. This may lead to negative nuggets, which is not allowed (a variance cannot be negative). The preferable alternative is to fit the model with maximum likelihood, and adding the measurement error variance to the diagonal of the covariance matrix of the data, see function 11 in Section 13.1.3.

Exercises

1. Write an **R** script to predict the sampling variance of the estimator of the mean of NO₃-N of agricultural field Leest, estimated by simple random random sampling and a sample size of 25 points. Use in prediction a spherical semivariogram with a nugget of 483, a partial sill of 483 and a range of 44.6 m. The sum of the nugget and partial sill (966) is equal to the sill of the semivariogram used above in predicting sampling variances. Compare the predicted sampling variance with the predicted sampling variance for the same sampling design, obtained with the semivariogram without nugget. Explain the difference.
2. Write an **R** script to compute the required sample size for simple random random sampling of agricultural field Leest, for a maximum length of a 95% confidence interval of 20. Use the semivariogram without nugget in predicting the sampling variance. See Section 12.2 (Equation (12.7)) for how to compute this required sample size given a prior

estimate of the standard deviation of the study variable in the population.

3. Do the same for systematic random sampling. Note that for this sampling design no such formula is available. Predict for a series of *expected* sample sizes, $n = 5, 6, \dots, 40$, the sampling variance of the estimator of the mean, using Equation (13.6). Approximate $E_{SY}(\bar{\gamma}_{SY})$ from ten repeated selections. Compute the length of the confidence interval from the predicted sampling variances, and plot the interval length against the sample size. Finally determine the required sample size for a maximum length of 20. What is the design effect for an expected sample size of 34 points (the required sample size for simple random sampling), see Equation (12.13)? Also compute the design effect for expected sample sizes of $5, 6, \dots, 40$. Explain why the design effect is not constant.

13.1.2 Geostatistical simulation approach

In the analytical approach no values of the study variable are simulated, neither at the nodes of a discretisation grid to predict the population variance, nor at the points of random samples to predict the variance within samples. Mean semivariances, $\bar{\gamma}$, $\bar{\gamma}_h$, $\bar{\gamma}_{SY}$, are computed from matrices with distances between pairs of points that are transformed into matrices with semivariances. It is true that for systematic random sampling in cases where the sample size is and/or the spatial pattern of the sampling points is random, simulation is needed. A large number of systematic samples is selected to approximate the design-expectation of the mean semivariance within a systematic sample (Equation (13.6)). However, only the spatial coordinates of these samples are used. No values of the study variable at the sampling points are simulated.

The alternative is to use a geostatistical simulation approach. It is computationally more demanding, but an advantage of this approach is its flexibility. It can also be used to predict the sampling variance of the estimator of the mean using a geostatistical model with a non-constant mean. And besides, this approach can also handle trans-Gaussian variables, i.e. of variables whose distribution can be transformed into a normal distribution. In Section 13.1.3 on the Bayesian approach geostatistical simulation is used to predict the variance of the estimator of the mean of a lognormal variable.

The geostatistical simulation approach for predicting the sampling variance of a design-based estimator of the population mean involves the following steps:

1. Select a large number S of random samples with sampling design p .
2. Use the model to simulate values of the study variable for all sampling points.
3. Estimate for each sample the population mean, using the design-based estimator of the population mean for sampling design p . This results in S estimated population means.
4. Compute the variance of the S estimated means.
5. Repeat steps 1 to 4 R times, and compute the mean of the R variances.

This approach is illustrated with the three administrative regions (woredas) in Ethiopia where a large sample is available with organic matter data in the topsoil (SOM) in mass percentages (wt%) of dry soil. The soil samples are collected along roads (see Figure 18.4). It is a convenience sample, not a probability sample, so these sample data cannot be used in design-based or model-assisted estimation of the mean or total soil carbon stock in the study area. However, the data can be used to model the spatial variation of SOM, and this geostatistical model can then be used to design a probability sample for design-based estimation of the total SOM stock. Apart from the point data of SOM, maps of covariates are available, such as a digital elevation model and remote sensing reflectance data. I selected four covariates to model the mean of SOM: elevation (dem), average near infrared reflectance (rfl-NIR), average red reflectance (rfl-red) and average land surface temperature (lst). I assume a normal distribution for the residuals of the linear model. The model parameters are estimated by restricted maximum likelihood (REML), using package **geoR** (?), see Section 22.4.2 for details on REML estimation of a geostatistical model.

```
library(geoR)
load(file="data/ThreeWoredasEthiopia.RData")
priordata <- as(priordataEthiopia, "data.frame")
dGeoR <- as.geodata(
  obj=priordata, header=TRUE,
  coords.col=13:14, data.col=1, covar.col=c(3,9,10,11))
vgm_REML <- likfit(
  geodata=dGeoR,
```

```
trend=~dem+rfl_NIR+rfl_red+lst,
cov.model="spherical", ini.cov.pars=c(1,5),nugget=0.2,
lik.method="REML", messages=FALSE)
```

The estimated parameters of the residual semivariogram of SOM are shown in Table 23.2. The estimated regression coefficients are 56.5 for the intercept, 0.157 for elevation (dem), 9.54 for NIR reflectance, -8.98 for red reflectance and -0.178 for land surface temperature.

Package **gstat** (?) is used for geostatistical simulation, and therefore first the REML estimates of the semivariogram parameters are assigned to the arguments **nugget**, **psill** and **range** of the function **vgm** of this package.

```
vgm_REML_gstat <- vgm(
  model="Sph",
  nugget=vgm_REML$tausq,
  psill=vgm_REML$sigmasq,
  range=vgm_REML$phi)
```

The fitted model of the spatial variation of SOM is used to compare systematic random sampling and two-stage cluster random sampling at equal variances of the estimated mean. First the sampling variance with systematic random sampling is predicted. One hundred systematic random samples ($S = 100$) with an expected sample size of 50 points ($E[n] = 50$) are selected. The four covariates at the selected sampling points are extracted by overlaying the **SpatialPointsDataFrame** **mySYsamples** and the **SpatialPixelsDataFrame** **grd** with function **over** of package **sp** (?). Values at the sampling points are simulated by sequential Gaussian simulation (?), using function **krige** with argument **nsim = 1** of package **gstat**. The argument **dummy** is set to **TRUE** to enforce unconditional simulation. The alternative is conditional simulation, using the data of the convenience sample as a conditioning data. Conditional simulation is only recommended if the quality of these legacy data is sufficient, and we may trust that the study variable at the legacy points is not changed since these legacy data are collected.

Note that by first drawing 100 samples, followed by simulating values of z at the selected sampling points, instead of first simulating values of z at the nodes of a discretisation grid, followed by selecting samples and overlaying with the

simulated field, the simulated values of points in the same discretisation cell differ, so that we account for the infinite number of points in the population.

With systematic random sampling the sample mean is an approximately unbiased estimator of the population mean (Chapter 5). Therefore, of each sample the mean of the simulated values is computed, using the function `tapply`. Finally, the variance of the 100 sample means is computed. This is a conditional variance, conditional on the simulated values. The whole procedure is repeated 100 times ($R = 100$), leading to 100 conditional variances of sample means.

```
load(file="data/CovariatesThreeWoredasEthiopia.RData")
grd <- grdEthiopia
gridded(grd) <- ~s1+s2
S <- R <- 100
v_mzsim_SY <- numeric(length=R)
set.seed(314)
for (i in 1:R) {
  mySYsamples <- NULL
  for (j in 1:S) {
    xy <- spsample(x=grd, n=50, type="regular")
    mySY <- data.frame(
      s1=xy$x1, s2=xy$x2, sample=rep(j, length(xy)))
    mySYsamples <- rbind(mySYsamples, mySY)
  }
  coordinates(mySYsamples) <- ~s1+s2
  res <- over(mySYsamples, grd)
  mySYs <- data.frame(mySYsamples, res[,c(1,3,4,5)])
  coordinates(mySYs) <- ~s1+s2
  zsim <- krige(
    dummy~dem+rfl_NIR+rfl_red+lst,
    locations=mySYs, newdata=mySYs,
    model=vgm_REML_gstat, beta=vgm_REML$beta,
    nmax=20, nsim=1,
    dummy=TRUE,
    debug.level=0) %>% as(., "data.frame")
  m_zsim <- tapply(zsim$sim1, INDEX=mySYs$sample, FUN=mean)
  v_mzsim_SY[i] <- var(m_zsim)
}
```

The mean of the 100 conditional variances equals 0.018. This is a Monte Carlo approximation of the model-based prediction of the sampling variance of the ratio estimator of the mean for systematic random sampling with an expected sample size of 50.

Due to the geographical spreading of the sampling points with systematic random sampling, the accuracy of the estimated mean is expected to be high compared to that of other sampling designs of the same size. However, with large areas the time needed for travelling to the sampling points can become substantial, lowering the sampling efficiency. With large areas, sampling designs leading to spatial clusters of sampling points can be an attractive alternative. One option then is two-stage cluster random sampling, see Chapter 7. The question is whether this alternative design is more efficient than systematic random sampling.

For the three woredas in Ethiopia 100 compact geostrata (see Section 4.6) are computed. Here these geostrata are not used as strata in stratified random sampling, but as primary sampling units (psu's) in two-stage cluster random sampling. The difference is that in stratified random sampling from each geostratum at least one sampling unit is selected, whereas in two-stage cluster random sampling only a randomly selected subset of the geostrata is sampled. The compact geostrata, used as psu's, are computed with function **kmeans**, and as a consequence the psu's do not have equal size. This is not needed in two-stage cluster random sampling, see Chapter 7. If psu's of equal size are preferred, then these can be computed with function **stratify** of package **spcosa** with argument **equalArea=TRUE**, see Section 4.6.

```
load(file="data/CovariatesThreeWoredasEthiopia.RData")
grd <- grdEthiopia
set.seed(314)
res <- kmeans(
  grd[,c("s1","s2")], iter.max=1000, centers=100, nstart=100)
mypsus <- res$cluster
psusize <- as.numeric(table(mypsus))
summary(psusize)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	79.0	103.8	109.0	108.4	113.0	131.0

To keep the estimation of the population mean simple, the psu's are selected with

probabilities proportional to their size and with replacement (ppswr sampling), see Chapter 7.

13.1.2.1 Optimisation of sample sizes for two-stage cluster random sampling

In Section 7.1 formulas are presented for computing the optimal number of psu draws and ssu draws per psu draw. The optimal sample sizes are a function of the pooled variance of primary unit means, S_b^2 , and the pooled variance of secondary units (points) within the primary units, S_b^2 . In this section these variance components are predicted with the geostatistical model.

As a first step a large number of fields is simulated.

```
grd$psu <- mypsus
coordinates(grd) <- ~s1+s2
set.seed(314)
zsim <- krige(
  dummy~dem+rfl_NIR+rfl_red+lst,
  locations=grd, newdata=grd,
  model=vgm_REML_gstat, beta=vgm_REML$beta,
  nmax=20, nsim=1000,
  dummy=TRUE,
  debug.level=0) %>% as(., "data.frame")
zsim <- zsim[,-c(1,2)]
```

For each simulated field the means of the psu's and the variances within the psu's are computed using function `tapply` in function `apply`.

```
m_zsim_psu <- apply(zsim, MARGIN=2, FUN=function(x)
  tapply(x, INDEX=grd$psu, FUN=mean))
v_zsim_psu <- apply(zsim, MARGIN=2, FUN=function(x)
  tapply(x, INDEX=grd$psu, FUN=var))
```

Next for each simulated field the pooled variance of psu means and pooled variance within psu's is computed, and finally these pooled variances are averaged over all simulated fields. These averages are approximations of the model-expectations of the pooled between unit and within unit variances, $E_\xi[S_b^2]$ and

$$E_{\xi}[S_w^2].$$

```
p_psu <- psusize/sum(psusize)
S2b <- apply(m_zsim_psu, MARGIN=2, FUN=function(x)
  sum(p_psu*(x-sum(p_psu*x))^2))
S2w <- apply(v_zsim_psu, MARGIN=2, FUN=function(x)
  sum(p_psu*x))
Exi_S2b <- mean(S2b)
Exi_S2w <- mean(S2w)
```

The optimal sample sizes are computed for a simple linear costs model: $C = c_1 n + c_2 m$, with c_1 the access costs per primary unit, including the access costs of the ssu's (points) with a given psu, and c_2 the observation costs per ssu. Note that for the optimal sample sizes only the ratio of c_1 and c_2 is important, not their absolute values. Given values for c_1 and c_2 , the optimal number of psu draws n , and optimal number of ssu draws per psu draw m is computed, required for a sampling variance of the estimator of the mean equal to the sampling variance with systematic random sampling of 50 points, see Equations (7.9) and (7.10).

```
c1 <- 2; c2 <- 1
nopt <- 1/Exi_vrz_SY*(sqrt(Exi_S2w*Exi_S2b)*sqrt(c2/c1)+Exi_S2b)
mopt <- sqrt(Exi_S2w/Exi_S2b)*sqrt(c1*c2)
```

The optimised number of psu draws is 32, and the optimal number of points per psu draw equals 3. The total number of sampling points is $32 \times 3 = 96$. This is much larger than the sample size of 50 obtained with systematic random sampling. The total observation costs therefore are substantially larger. However, the access time can be substantially smaller due to the spatial clustering of sampling points. To answer the question whether the costs saved by this reduced access time outweighs the extra costs of observation, the model for the access costs and observation costs must be further developed.

13.1.3 Bayesian approach

The model-based prediction of the variance of the design-based estimate of the population mean for a given sampling design is conditional on the model. If we change the model type or the model parameters, the predicted sampling

variance also changes. In most situations we are quite uncertain about the model, even in situations where we have data that can be used to estimate the model parameters, as in the Ethiopia case study. Instead of using the best estimated model to predict the sampling variance as done in the previous sections, we may prefer to account for the uncertainty about the model parameters. This can be done through a Bayesian approach, in which the legacy data are used to update a prior distribution of the model parameters to a posterior distribution. For details about a Bayesian approach for estimating model parameters, see Section 23.5. A sample from the posterior distribution of model parameters is used one-by-one to predict the sampling variance. This can be done either analytically, as described in Section 13.1.1, or through geostatistical simulation. Both approaches result in a *distribution* of sampling variances, reflecting our uncertainty about the sampling variance of the estimator of the population mean due to uncertainty about the semivariogram. The mean or median of the distribution of sampling variances can be used as the predicted sampling variance.

The Bayesian approach is illustrated with a case study on predicting the sampling variance of NO₃-N in an agricultural field in Belgium (?). Data of NO₃-N are available at 30 points, forming approximately a square grid with a spacing of about 4.5 m. As a first step, I check whether we can safely assume that the data come from a normal distribution.

```
mydata <- read.csv("data/Melle17.csv", header=TRUE)
ggplot(mydata, aes(sample=N)) +
  geom_qq() +
  geom_qq_line()
```

```
pvalue <- shapiro.test(mydata$N)$p.value
```

The Q-Qplot shows that a normal distribution is not very likely: there are too many large values, the distribution is skewed to the right. Also the *p*-value of the Shapiro-Wilk test shows that we should reject the null-hypothesis of a normal distribution for the data: $p=0.0028$. I therefore proceed with the natural logarithm of NO₃-N, in short lnN.

As a first step the semivariogram of lnN is estimated by maximum likelihood (Section 22.3.2). An exponential semivariogram model is assumed. Note that

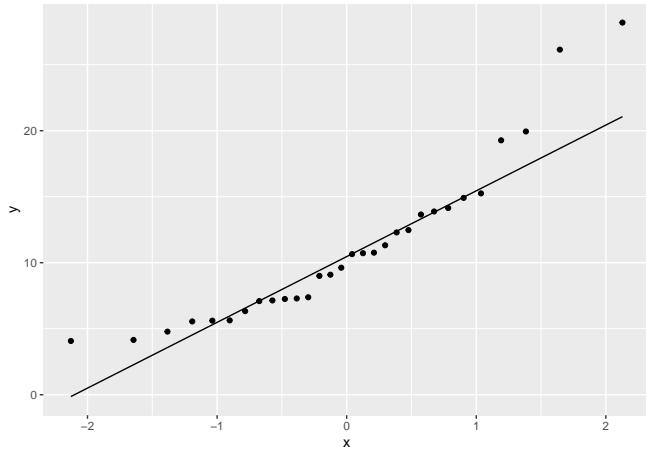


Figure 13.2: Q-Q plot of Nitrate-N of field Melle.

the parameters that are estimated are the reciprocal of the sill λ , the ratio of spatial dependence ξ , defined as the partial sill divided by the sill, and the distance parameter ϕ . This parameterisation of the semivariogram is chosen because hereafter in the Bayesian approach prior distributions are chosen for these parameters.

The likelihood function is defined, using a somewhat unusual parameterisation, tailored at the Markov chain Monte Carlo (MCMC) sampling from the posterior distribution of the semivariogram parameters. In MCMC a Markov chain of sampling units (vectors with semivariogram parameters) is generated by using the previous sampling unit to randomly generate the next sampling unit (? , chapter 11]. In MCMC sampling the probability of accepting a proposed sampling unit θ^* is a function of the ratio of the posterior density of the proposed sampling unit and that of the current sampling unit, $f(\theta^*|\mathbf{z})/f(\theta_{t-1}|\mathbf{z})$, so that the normalizing constant, the denominator of Equation (12.15), cancels. For a nice introduction to MCMC I refer to these lecture notes¹.

```
library(mvtnorm)
ll <-function(thetas) {
```

¹<http://nitro.biosci.arizona.edu/courses/EEB596/handouts/Gibbs.pdf>

```
sill <- 1/thetas[1]
psill <- thetas[2]*sill
nugget <- sill-psill
vgmodel <- vgm(
  model=model, psill=psill, range=thetas[3], nugget=nugget)
C <- variogramLine(vgmodel, dist_vector=D, covariance=TRUE)
Cinv <- chol2inv(chol(C))
XC <- crossprod(X, Cinv)
XCz <- XC%*%z
XCX <- XC%*%X
beta <- solve(XCX, XCz)
mu <- as.numeric(X%*%beta)
logLik <- dmvnorm(x=z, mean=mu, sigma=C, log=TRUE)
logLik
}
```

Next, initial estimates of the semivariogram parameters are estimated by maximising the likelihood, using function `optim`.

```
lambda.ini <- 1/var(mydata$lnN)
xi.ini <- 0.5
phi.ini <- 20
pars <- c(lambda.ini, xi.ini, phi.ini)
D <- as.matrix(dist(mydata[,c(1,2)]))
X <- matrix(1, nrow(mydata), 1)
z <- mydata$lnN
model <- "Exp"
vgML <- optim(
  pars, ll, control=list(fnscale = -1),
  lower=c(1e-6,0,1e-6), upper=c(1000,1,150),
  method="L-BFGS-B")
```

The maximum likelihood (ML) estimates of the semivariogram parameters are used as initial values in MCMC sampling. In the Bayesian approach, I use a uniform prior for the inverse of the sill parameter, $\lambda = 1/\sigma^2$, with a lower bound of 10^{-6} and an upper bound of 1. For the distance parameter ϕ of the exponential semivariogram a uniform prior is assumed, with a lower bound of

10^{-6} m. and an upper bound of 150 m. For the relative nugget, τ^2/σ^2 , a uniform prior is assumed with a lower bound of 0 and an upper bound of 1.

These priors can be defined by function `createUniformPrior` of package **BayesianTools** (?). The function `createBayesianSetup` is then used to define the setup of the MCMC sampling, specifying the likelihood function, the prior, and the vector with best prior estimates of the model parameters, specified with argument `best`. Argument `sampler` of function `runMCMC` specifies the type of MCMC sampler. I used the differential evolution algorithm of ?. Argument `start` of function `getSample` specifies the burn-in period, i.e. the number of first samples that are discarded to diminish the influence of the initial semivariogram parameter values. Argument `numSamples` specifies the sample size, i.e. the number of saved vectors with semivariogram parameter values, drawn from the posterior distribution.

```
library(BayesianTools)
priors <- createUniformPrior(lower=c(1e-6, 0, 1e-6),
                               upper=c(1000, 1, 150))
bestML <- c(vgML$par[1], vgML$par[2], vgML$par[3])
setup <- createBayesianSetup(
  likelihood=ll, prior=priors,
  best=bestML, names=c("lambda", "xi", "phi"))
set.seed(314)
res <- runMCMC(setup, sampler="DEzs")
MCMCsample <- getSample(
  res, start=1000, numSamples=1000) %>% data.frame(.)
```

Figure 13.3 shows several semivariograms, sampled by MCMC from the posterior distribution of the estimated semivariogram parameters.

The evaluated sampling design is the same as used in Section 13.1.1 above for field Leest: stratified simple random sampling, using compact geographical strata of equal size, a total sample size of 25 points, and one point per stratum.

The next step is to simulate with each of the sampled semivariograms a large number of maps of lnN. This is done by sequential Gaussian simulation, conditional on the available data. The simulated values are backtransformed. Each simulated map is then used to compute the variance of the simulated values within the geostrata S_h^2 . These stratum variances are used to compute the sampling variance of the estimator of the mean:

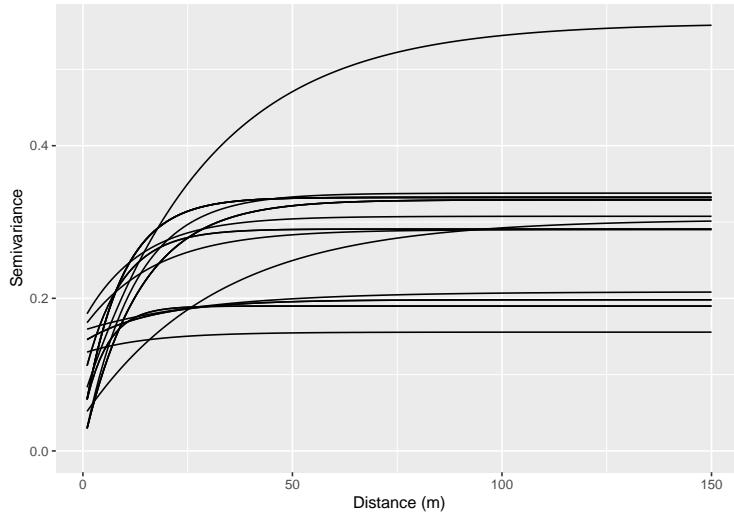


Figure 13.3: Semivariograms of $\ln N$, obtained by MCMC sampling from posterior distribution of the estimated semivariogram parameters for field Melle.

$$V(\hat{z}) = \frac{1}{H^2} \sum_{h=1}^H S_h^2. \quad (13.7)$$

This variance estimator follows from Equation (4.4). Plugging $w_h = 1/H$ (all strata have equal size) and $n_h = 1$ into this variance estimator yields Equation (13.7).

I used the first 100 sampled semivariograms, and with each semivariogram I simulated 100 maps.

```
V <- matrix(data=NA, nrow=100, ncol=100)
coordinates(mydata) <- ~Easting+Northing
set.seed(314)
for (i in 1:100) {
  sill <- 1/MCMCsamp$lambda[i]
  psill <- MCMCsamp$xi[i]*sill
  nug <- sill - psill
```

```

range <- MCMCsamp$phi[i]
vgmdl <- vgm(model="Exp", nugget=nug, psill=psill, range=range)
ysim <- krige(
  lnN~1, locations=mydata, newdata=mygrid,
  model=vgmdl,
  nmax=20, nsim=100,
  debug.level=0) %>% as(., "data.frame")
zsim <- exp(ysim[,-c(1,2)])
S2h <- apply(zsim, MARGIN=2, FUN=function(x)
  tapply(x, INDEX=as.factor(mygeostrata$stratumId), FUN=var))
V[i,] <- 1/n^2*apply(S2h, MARGIN=2, FUN=sum)
}

```

Figure 13.4 shows sixteen simulated maps, simulated with the first four semivariograms. The four maps in a row (a to d) are simulated with the same semivariogram. All maps show that the simulated data have positive skew, which is in agreement with the prior data. The data obtained by simulating from a lognormal distribution are always strictly positive. This is not guaranteed when simulating from a normal distribution.

The sampling variances of the estimated mean of NO₃-N obtained with these sixteen maps are shown below.

	a	b	c	d
1	1.322	0.816	0.869	0.842
2	1.407	1.170	0.959	1.143
3	0.589	0.587	0.563	0.552
4	0.873	1.778	0.904	0.679

The sampling variance shows quite strong variation among the maps. The histogram shows the uncertainty distribution of the sampling variance, due to uncertainty about the semivariogram, as well as due to uncertainty about the spatial distribution of NO₃-N within the agricultural field given the semivariogram and the available data from that field.

As a model-based prediction of the sampling variance we can take the average or the median of the sampling variances over all 100×100 simulated maps, which are equal to 0.724 and 0.663, respectively. If we want to be more safe, we can take a high quantile, e.g. the P90 of this distribution as the predicted sampling variance, which is equal to 1.099

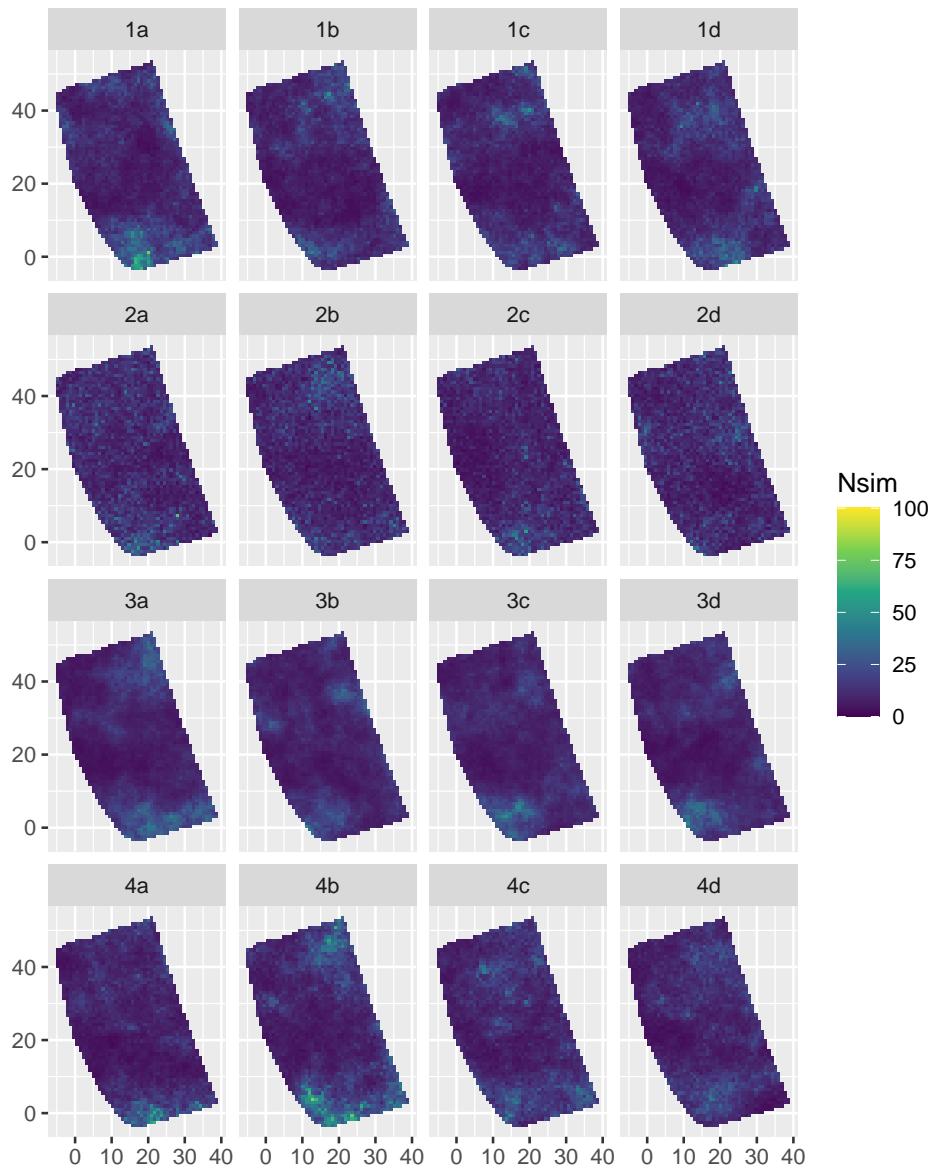


Figure 13.4: Maps of $\text{NO}_3\text{-N}$ simulated with four semivariograms (rows). Each semivariogram is used to simulate four maps (columns a-d).

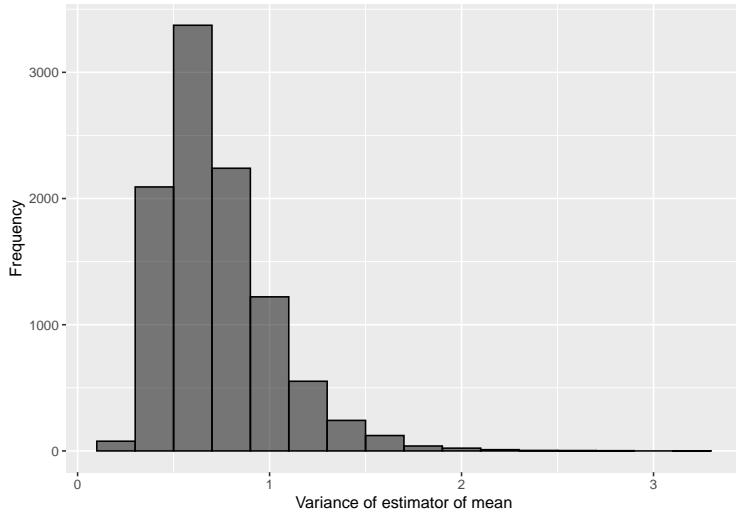


Figure 13.5: Histogram of simulated sampling variances of the estimator of the mean of NO3-N for stratified simple random sampling of field Melle.

Above I used the 30 data as conditioning data in geostatistical simulation, Unconditional simulation is recommended if we cannot rely on the quality of the legacy data, for instance due to a temporal change in lnN since the time the legacy data were observed. For NO3-N this might well be the case. I believe that, although the effect of 30 observations on the simulated fields and on the uncertainty distribution of the sampling variance will be very small, one still may prefer unconditional simulation. With unconditional simulation we must assign the model mean μ to the argument `beta` of function `krige`. The estimated model mean depends on the semivariogram parameters. The next code chunk shows how this model mean can be estimated(see also function `11` above).

```
C <- variogramLine(vgmodel, dist_vector=D, covariance=TRUE)
Cinv <- chol2inv(chol(C))
XC <- crossprod(X, Cinv)
XCz <- XC%*%z
XCX <- XC%*%X
beta <- solve(XCX, XCz)
```

13.2 Model-based optimisation of spatial strata

? described a novel spatial stratification method that uses model predictions of the study variable as a stratification variable while accounting for errors in the predictions, as well as spatial correlation of the prediction errors.

The **Julia** package **Ospats** is an implementation of the stratification method. In **Ospats** the stratification is optimised through iterative reallocation of the raster cells to the strata. Recently, this stratification method was implemented in the package **SamplingStrata** (? , ?). However, the algorithm used to optimise the strata differs from that in **Ospats**. In **SamplingStrata** the stratification is optimised by optimising the breaks (splitting points) on the stratification variable with a genetic algorithm. Optimisation of the strata through optimisation of the breaks on the stratification variable necessarily leads to non-overlapping strata, while with iterative reallocation the strata may overlap, i.e. when the strata are sorted on the mean of the stratification variable, the upper bound of a stratum can be larger than the lower bound of the next stratum. As argued by ? optimisation of strata through optimisation of the stratum breaks can be suboptimal. On the other hand, optimisation through optimisation of the breaks needs fewer computations, and therefore is quicker.

The situation considered in this section is that prior data are available, either from the study area itself or from another similar area, that can be used to fit a statistical model for the study variable, using one or more quantitative covariates and/or factors as predictors. We wish to collect (more) data by stratified simple random sampling, to be used in design-based estimation of the population mean or total of the study variable. The central research question then is how to construct these strata.

Recall the formula for the variance of the estimator of the mean for stratified simple random sampling (see Equations (4.4) and (4.5)):

$$V(\hat{z}) = \sum_{h=1}^H w_h^2 \frac{S_h^2(z)}{n_h} . \quad (13.8)$$

Plugging the stratum sample sizes under optimal allocation (Equation (4.16)) into Equation (13.8), gives for the variance of the estimator of the mean:

$$V(\hat{z}) = \frac{1}{n} \left(\sum_{h=1}^H w_h S_h(z) \sqrt{c_h} \sum_{h=1}^H \frac{w_h S_h(z)}{\sqrt{c_h}} \right). \quad (13.9)$$

So given the total sample size n the variance of the estimator of the mean is minimal when the criterion

$$O = \sum_{h=1}^H w_h S_h(z) \sqrt{c_h} \sum_{h=1}^H \frac{w_h S_h(z)}{\sqrt{c_h}} \quad (13.10)$$

is minimised.

? assume that the costs are equal for all population units, so that the mean costs are the same for all strata. In this case the minimisation criterion reduces to

$$O = \left(\sum_{h=1}^H w_h S_h(z) \right)^2. \quad (13.11)$$

In practice we do not know the values of the study variable z . ? consider the situation where we have predictions of the study variable from a statistical model: $\hat{z} = z + \epsilon$, with ϵ the prediction error. So this implies that we do not know the stratum standard deviations $S_h(z)$ of Equation (13.9). What we do have are the stratum standard deviations of the predictions of z : $S_h(\hat{z})$. With many statistical models, such as regression and kriging models, the standard deviation of the predictions are smaller than that of the study variable: $S_h(\hat{z}) < S_h(z)$. This is known as the smoothing or leveling effect.

? replaced the stratum standard deviations in the minimisation criterion by model-expectations of these stratum standard deviations, i.e. by model-based predictions of the stratum standard deviations), $E[S_h(z)]$. This leads to the following minimisation criterion:

$$E_\xi[O] = \left(\sum_{h=1}^H w_h E_\xi[S_h(z)] \right)^2. \quad (13.12)$$

? predicted the stratum variances by

$$E_\xi[S_h^2(z)] = \frac{1}{N_h^2} \sum_{i=1}^{N_h-1} \sum_{j=i+1}^{N_h} E_\xi[d_{ij}^2], \quad (13.13)$$

with $d_{ij}^2 = (z_i - z_j)^2$ the squared difference of the study variable values at two nodes of a discretisation grid. The model-expectation of the squared differences are equal to

$$E_\xi[d_{ij}^2] = (\hat{z}_i - \hat{z}_j)^2 + S^2(\epsilon_i) + S^2(\epsilon_j) - 2S^2(\epsilon_i, \epsilon_j), \quad (13.14)$$

with $S^2(\epsilon_i)$ the variance of the prediction error at node i , and $S^2(\epsilon_i, \epsilon_j)$ the covariance of the prediction errors at nodes i and j . The authors then argue that for smoothers, such as kriging and regression, the first term must be divided by the squared correlation coefficient R^2 :

$$E_\xi[d_{ij}^2] = \frac{(\hat{z}_i - \hat{z}_j)^2}{R^2} + S^2(\epsilon_i) + S^2(\epsilon_j) - 2S^2(\epsilon_i, \epsilon_j). \quad (13.15)$$

The predicted stratum standard deviations are approximated by the square root of Equation (13.15). Plugging these model-based predictions of the stratum standard deviations into the minimisation criterion, Equation (13.11), yields

$$E_\xi[O] = \frac{1}{N} \sum_{h=1}^H \left(\sum_{i=1}^{N_h-1} \sum_{j=i+1}^{N_h} \frac{(\hat{z}_i - \hat{z}_j)^2}{R^2} + S^2(\epsilon_i) + S^2(\epsilon_j) - 2S^2(\epsilon_i, \epsilon_j) \right)^{1/2}. \quad (13.16)$$

Optimal spatial stratification with package **SamplingStrata** is illustrated with a survey of the soil organic matter (SOM) concentration (g/kg) in the topsoil (A horizon) of Xuancheng (China). Three samples are available. These three samples are merged. The total number of sampling points is 183. This sample is used to fit a simple linear regression model for SOM, using the elevation of the surface (dem) as a predictor. The function `lm` of the **stats** package is used to fit the simple linear regression model.

```

sample_grid <- read.csv("data/Xuancheng_gridsample.csv")
sample_iPSM <- read.csv("data/Xuancheng_iPSMsample.csv")
sample_test <- read.csv("data/Xuancheng_Stratifiedrandomsample.csv")
mysample <- rbind(sample_grid, sample_iPSM,
                    sample_test[,-c(13,14,15)])
lm_SOM <- lm(SOM_A_hori~dem, data=mysample)

```

In fitting a linear regression model we assume that:

1. the relation is linear.
2. the residual variance is constant (independent of the fitted value).
3. the residuals have a normal distribution.

These assumptions are checked with a scatter plot of the residuals against the fitted value and a Q-Q plot, respectively.

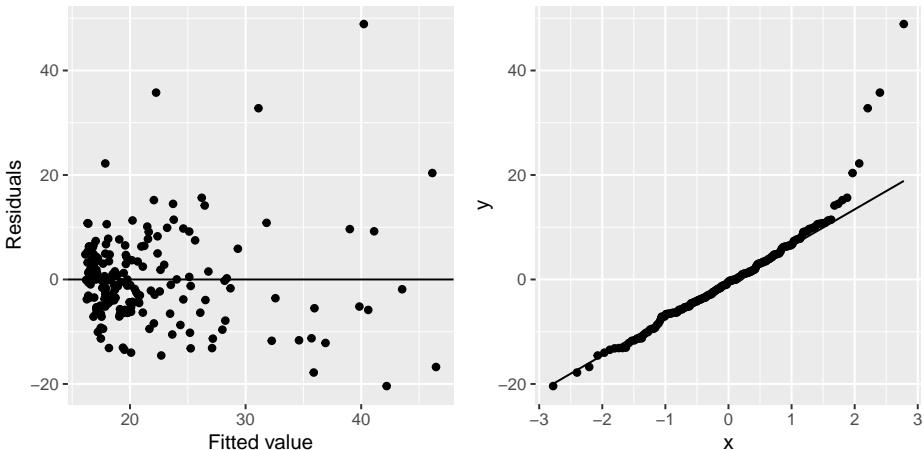


Figure 13.6: Scatter plot of residuals against fitted value, and Q-Q plot of residuals, for a simple linear regression model of soil organic matter concentration in Xuancheng, with elevation as a predictor.

The scatter plot shows that the first assumption is realistic. No pattern can be seen, at all fitted values the residuals are scattered around the horizontal line.

However, the second and third assumptions are questionable: the residual variance clearly increases with the fitted value, and the distribution of the residuals has positive skew, i.e. it has a long upper tail. There clearly is some evidence that these two assumptions are violated. Possibly these problems can be solved by fitting a model for the natural logarithm of SOM.

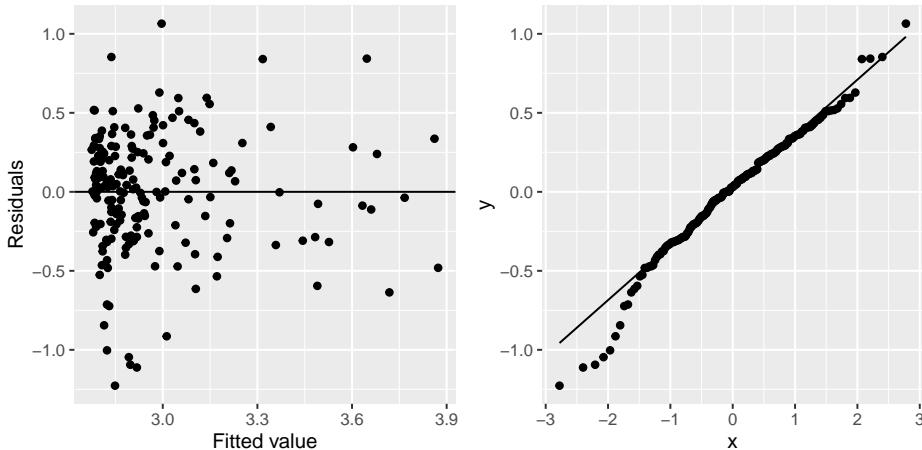


Figure 13.7: Scatter plot of residuals against fitted value, and Q-Q plot of residuals, for a simple linear regression model of natural logarithm of soil organic matter concentration in Xuancheng, with elevation as a predictor.

The variance of the residuals is more constant, and the Q-Q plot is improved, although we now have too many strong negative residuals for a normal distribution. I proceed with the model for natural log-transformed SOM ($\ln\text{SOM}$).

Another assumption in linear regression is that the residuals are independent. This assumption can be checked by computing a semivariogram. If the residual semivariance is not constant with the distance but increases, the assumption is violated. For details I refer to Chapter 22. Figure 13.8 shows the semivariogram of the residuals computed with function `variogram` of package `gstat` (?).

The first two points in the semivariogram have somewhat smaller values. This indicates that the residuals at two close points, say $< \pm 5$ km are not independent, whereas if the distance between the two points $> \pm 5$ km, they are independent. This spatial dependency of the residuals can be modelled, e.g. by an exponential function:

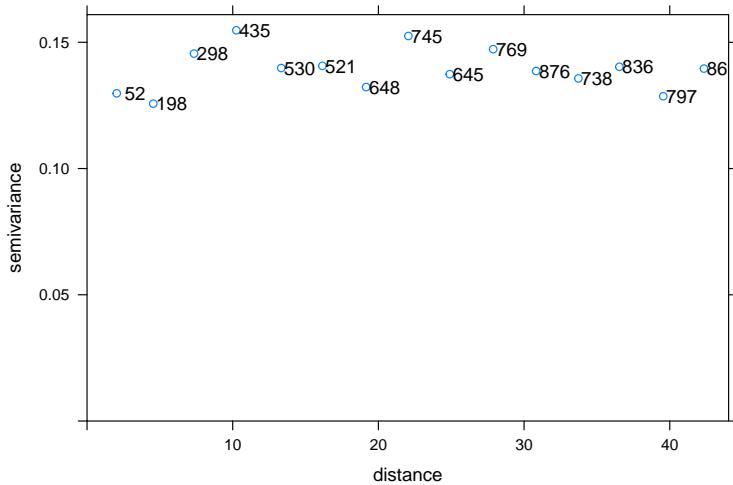


Figure 13.8: Sample semivariogram of residuals of simple linear regression model for natural logarithm of soil organic matter concentration in Xuancheng.

$$\gamma(h) = \begin{cases} 0 & \text{if } h = 0 \\ c_0 + c_1 \exp(-h/\phi) & \text{if } h > 0 \end{cases} \quad (13.17)$$

This exponential semivariogram has three parameters, the nugget variance c_0 , the partial sill variance c_1 and the distance parameter ϕ . The total number of model parameters now is five: two regression coefficients (intercept and slope for elevation), and three semivariogram parameters. All five parameters can best be estimated by restricted maximum likelihood, see Section 22.4.2. Table 13.1 shows the estimated regression coefficients and semivariogram parameters. Up to a distance of about three times the estimated distance parameter ϕ , which is about 8 km, the residuals are spatially correlated, beyond that distance, they are hardly correlated anymore.

The fitted model is used to predict lnSOM at the nodes of a 200 m \times 200 m discretisation grid.

```
load("data/Xuancheng.RData")
res <- predict(lm_lnSOM, newdata=as(grd, "data.frame"), se.fit=TRUE)
```

Table 13.1: Estimated regression coefficients (intercept and slope for dem), and parameters of exponential semivariogram for natural logarithm of soil organic matter concentration in Xuancheng (China).

Int	dem	Nugget	Partial sill	Distance parameter (km)
2.771	0.00222	0.085	0.061	2.588

```
grd <- within(as(grd,"data.frame"),
  {lnSOMpred <- res$fit; varpred <- res$se.fit^2})
```

The predictions and their standard errors are shown in Figure 13.9.

The discretisation grid with predicted lnSOM consists of 115,526 nodes. These are too many for function `optimStrata`. The grid is therefore thinned to a grid with a spacing of 800 m × 800 m, resulting in 7257 nodes.

The first step in optimisation of spatial strata with package `SamplingStrata` is to build the sampling frame with function `buildFrameSpatial`. The argument `X` specifies the stratification variables, and argument `Y` specifies the study variables. In our case we have only one stratification variable and one study variable, and these are the same variable. Argument `var` specifies the variance of the prediction error of the study variable. The variable `dom` is an identifier of the domain of interest of which we want to estimate the mean or total. I assigned the value 1 to all population units, which implies that the stratification is optimised for the entire population. If we have multiple domains of interest, the stratification is optimised for each domain separately.

Finally, as a preparatory step we must specify how precise the estimated mean should be. This precision must be specified in terms of the coefficient of variation (cv), i.e. the standard error of the estimated mean divided by the mean. In case of multiple domains of interest, and multiple study variables a cv must be specified per domain and per study variable. This precision requirement is used to compute the sample size for Neyman allocation (Equation (4.15))². The optimal stratification is independent of the precision requirement, although for a large cv, the optimal number of strata can be smaller than the value assigned

²For multivariate stratification, i.e. stratification with multiple study variables, Bethel allocation is used to compute the required sample size.

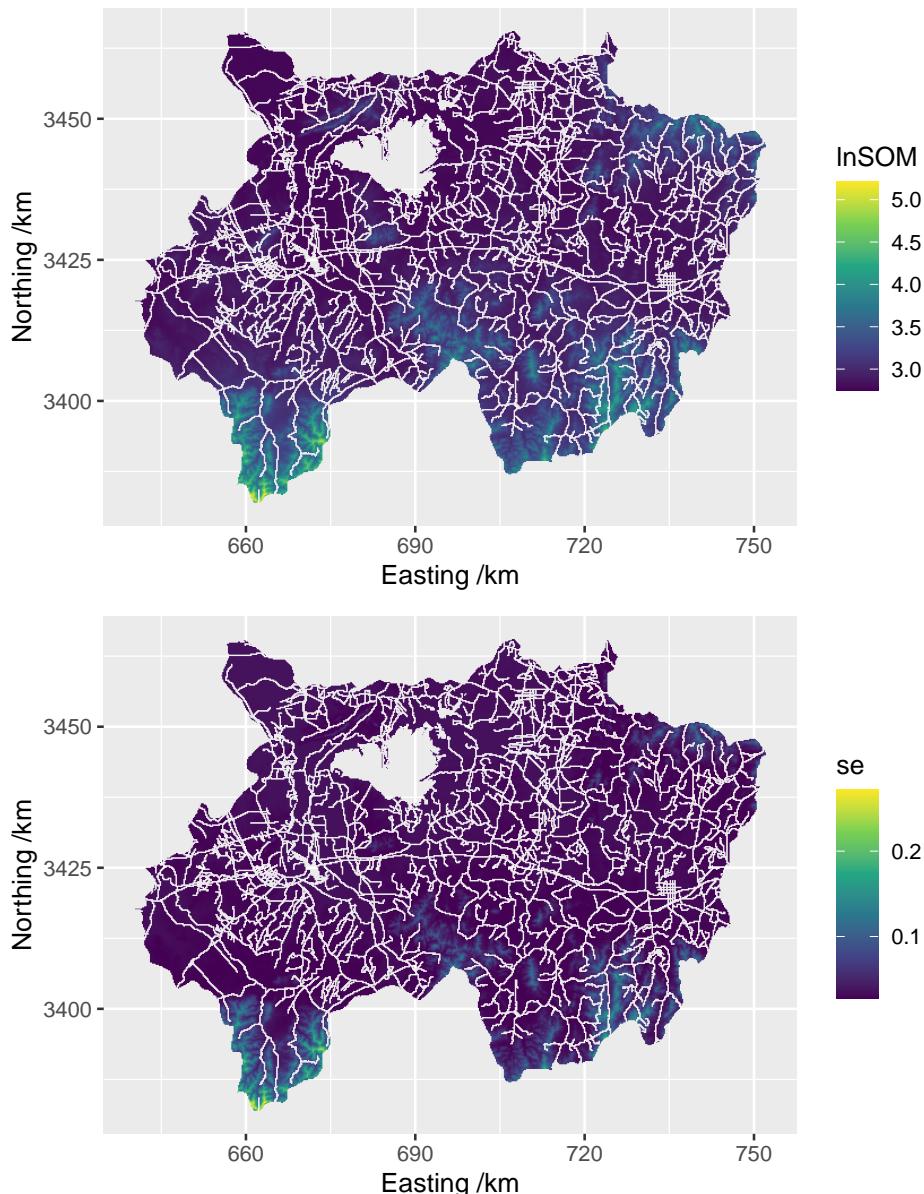


Figure 13.9: Predicted natural logarithm of soil organic matter concentration (g/kg) in the topsoil, and its standard error in Xuancheng, obtained with a linear regression model with elevation as predictor.

to argument `nStrata` of function `optimStrata`, see hereafter.

```
library(SamplingStrata)
subgrd$id <- c(1:nrow(subgrd))
subgrd$dom <- rep(1,nrow(subgrd))
frame <- buildFrameSpatial(
  df=subgrd, id="id",
  X=c("lnSOMpred"), Y=c("lnSOMpred"),
  variance=c("varpred"),
  lon="x1", lat="x2",
  domainvalue="dom")
cv <- as.data.frame(list(DOM="DOM1", CV1=0.005, domainvalue=1))
```

The optimal spatial stratification can be computed with function `optimStrata`, with argument `method="spatial"`. The R^2 value of the linear regression model, used in the minimisation criterion (Equation (13.16)), can be assigned to argument `fitting`. I used an R^2 value of one, because I believe the smoothing effect is already accounted for by the variance and covariance of the prediction errors (Equation (13.14)). Arguments `range` and `kappa` are parameters of an exponential semivariogram, needed for computing the covariance of the prediction errors. Function `optimStrata` uses an extra parameter in the exponential semivariogram (Equation (13.17)), $c_0 + c_1 \exp(-\kappa h/\phi)$, so for the usual exponential semivariogram `kappa` equals 1.

```
res <- optimStrata(
  framesamp=frame,
  method="spatial",
  errors=cv,
  nStrata=5,
  fitting=1,
  range=c(vgm_REML$phi),
  kappa=1,
  showPlot=FALSE)
```

A summary of the optimised strata can be obtained with function `summaryStrata`.

```
print(smr_strata <- summaryStrata(
  res$framenev, res$aggr_strata, progress=FALSE))

  Domain Stratum Population Allocation SamplingRate Lower_X1 Upper_X1
1       1       1      3090          12   0.004035 2.767950 2.862539
2       1       2      1931          10   0.005087 2.864846 2.991733
3       1       3      1133          10   0.008555 2.994040 3.215515
4       1       4       769          11   0.014398 3.217822 3.600789
5       1       5       334          11   0.033443 3.605403 5.098052
```

In the next code chunk it is checked whether the coefficient of variation is indeed equal to the desired value.

```
strata <- res$aggr_strata
framenev <- res$framenev
N_h <- strata$N
w_h <- N_h/sum(N_h)
se <- sqrt(sum(w_h^2*strata$S1^2/strata$SOLUZ))
se/mean(framenev$Y1)

[1] 0.005033349
```

The coefficient of variation can also be computed with function `expected_CV`.

```
expected_CV(strata)

cv(Y1)
DOM1 0.005
```

Figure 13.10 shows the optimised strata. I used the stratum bounds in the data frame `smr_strata`, to compute the stratum for all raster cells of the original 200 m × 200 m grid.

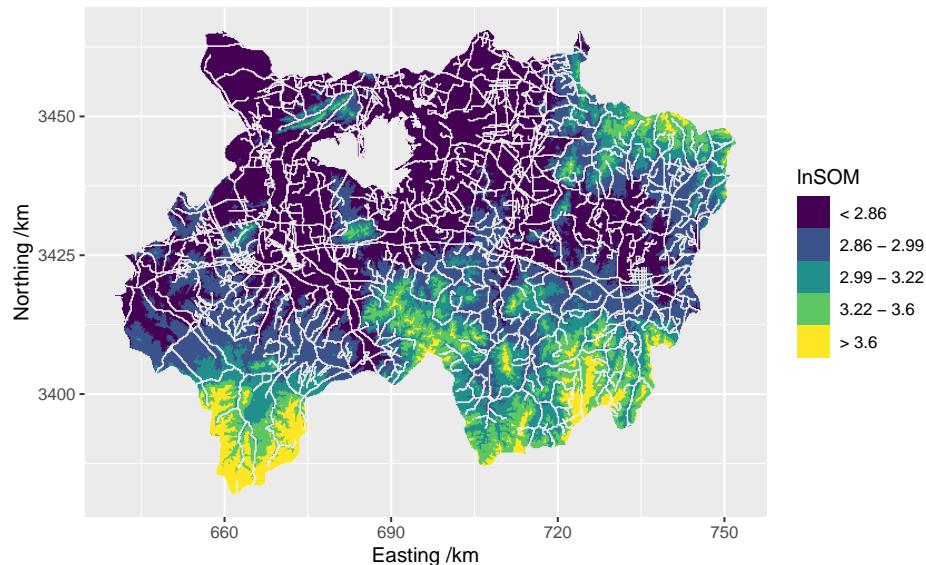


Figure 13.10: Model-based optimal strata for estimating the population mean of natural logarithm of soil organic matter (lnSOM) concentration (g/kg).

Chapter 14

Sampling for estimating parameters of (small) domains

This chapter is about probability sampling and estimation of means or totals of subpopulations (subareas, subregions). In the sampling literature these subpopulations are referred to as domains of interest, or shortly domains. Ideally at the stage of designing a sample these domains are known, and of every population unit we know to which domain it belongs. In that situation it is most convenient to use these domains as strata in random sampling, so that we can control the sample size in each domain (Chapter 4).

If we have multiple maps with domains, think for instance of a soil class map, a map with land cover classes, and a map with countries, we can make an overlay of these maps to construct the cross-tabulation strata. However, this may result in numerous strata, in some cases even more than the sample size. In this situation an attractive solution is multi-way stratification (Section 9.1.4). With this design the domains of interest are used as strata, not their cross-classification, and the sample sizes of these marginal strata are controlled.

Even with a multi-way stratified sample, resulting in controlled sample sizes for each domain, the sample size of a domain can be too small for a reliable estimate

of the total or mean when using the data of that domain only. In that case we may use model-assisted estimators. In Chapter 10 model-assisted estimation of the population mean or total is explained, using all data collected from the population are used. In this chapter (Section 14.2) also data outside the domain are used, more specifically to estimate the population regression coefficients (? , ?, ?).

We may also wish to estimate the mean or total of domains that are not used as (marginal) strata. The sample size in these domains is then not controlled, and varies among samples selected with the sampling design. As before with multi-way stratified sampling, the mean can either be estimated with the direct estimator, using the data from the domain only (Section 14.1), or a model-assisted estimator, also using data from outside the domain for estimating regression coefficients (Section 14.2).

14.1 Direct estimator for large domains

If the sample size of a domain d is considered large enough to obtain a reliable estimate of the mean, and besides the size of the domain is known, the mean of that domain can be estimated by the direct estimator:

$$\hat{z}_d = \frac{1}{N_d} \sum_{k \in \mathcal{S}_d} \frac{z_{dk}}{\pi_{dk}}, \quad (14.1)$$

where N_d is the size of the domain, z_{dk} is the value for unit k of domain d , and π_{dk} is the inclusion probability of this point.

When the domain is not used as a (marginal) stratum, so that the sample size of the domain is random, the mean of the domain can best be estimated by

$$\hat{z}_{\text{ratio},d} = \frac{\hat{t}_d(z)}{\widehat{N}_d} = \frac{\sum_{k \in \mathcal{S}_d} \frac{z_{dk}}{\pi_{dk}}}{\sum_{k \in \mathcal{S}_d} \frac{1}{\pi_{dk}}}. \quad (14.2)$$

with \widehat{N}_d the estimated size of the domain:

$$\widehat{N}_d = \sum_{k \in \mathcal{S}_d} \frac{1}{\pi_i}. \quad (14.3)$$

This ratio estimator can also be used when the size of the domain is unknown. An example of this is estimating the mean of soil classes *as observed in the field*, and not *as depicted on a soil map*. A soil map is impure, i.e. the map units contain patches with other soil classes than as indicated on the map. The area of a given true soil class is not known.

For simple random sampling $\pi_{dk} = n/N$. Inserting this in Equation (14.2) gives

$$\hat{\bar{z}}_{\text{ratio},d} = \frac{1}{n_d} \sum_{k \in \mathcal{S}_d} z_{dk} . \quad (14.4)$$

The mean of the domain is simply estimated by the mean of the z -values observed in the domain, i.e. the sample mean in domain d . The variance of this estimator can be estimated by

$$\widehat{V}(\hat{\bar{z}}_{\text{ratio},d}) = \frac{1}{\hat{a}_d^2} \cdot \frac{1}{n(n-1)} \sum_{k \in \mathcal{S}_d} (z_{dk} - \bar{z}_{\mathcal{S}_d})^2 , \quad (14.5)$$

where $\bar{z}_{\mathcal{S}_d}$ is the sample mean in domain d , and \hat{a}_d is the estimated relative size of domain d :

$$\hat{a}_d = \frac{n_d}{n} . \quad (14.6)$$

I refer to Section 8.2.2 in ? for the ratio estimator and its standard error with stratified simple random sampling in case the domains cut across the strata, and other sampling designs.

The ratio estimator and its standard error can be computed with function `svyby` of package **survey** (?). This is illustrated with Eastern Amazonia. We wish to estimate the mean aboveground biomass (AGB) of the sixteen ecoregions from a simple random sample of 200 units.

```
library(survey)
n <- 200
set.seed(314)
units <- sample.int(nrow(gridAmazonia), size=n, replace=FALSE)
mysample <- gridAmazonia[units,c("AGB","Ecoregion")]
```

Table 14.1: Ratio estimates of mean aboveground biomass (AGB) and their standard errors of ecoregions in Eastern Amazonia. The ecoregions with an estimated standard error of 0.0 have one sampling unit only, and actually are non-availables.

Ecoregion	AGB	se
Cerrado	99.7	15.6
Guianan highlands moist forests	296.0	0.0
Guianan lowland moist forests	263.0	0.0
Gurupa varzea	80.0	0.0
Madeira-Tapajos moist forests	286.0	14.0
Marajo varzea	116.6	27.2
Maranhao Babassu forests	90.0	9.6
Mato Grosso tropical dry forests	177.0	90.7
Monte Alegre varzea	189.5	89.0
Purus-Madeira moist forests	145.5	47.1
Tapajos-Xingu moist forests	288.6	8.9
Tocantins/Pindare moist forests	176.3	17.9
Uatuma-Trombetas moist forest	274.7	6.7
Xingu-Tocantins-Araguaia moist forests	223.1	16.6

```
mysample$N <- nrow(gridAmazonia)
design_si <- svydesign(id=~1, data=mysample, fpc=~N)
res <- svyby(~AGB, by=~Ecoregion, design=design_si, FUN=svymean)
```

The ratio estimates are shown in Table 14.1.

The sampling is repeated 1,000 times, and every sample is used to estimate the mean AGB of the ecoregions both with the π estimator and the ratio estimator. As can be seen below the standard error of the ratio estimator is much smaller than the standard deviation of the π estimator. The reason is that the number of sampling units in a ecoregion varies among samples, i.e. the sample size of an ecoregion is random. When many units are selected from an ecoregion, the estimated total of that ecoregion is large. The estimated mean as obtained with the π estimator then is large too, because the estimated total is divided by the fixed size (total number of population units, N_d) of the ecoregion. However, in

the ratio estimator the size of an ecoregion is estimated from the same sample, despite that we know its size, see Equation (14.2). With many units selected from an ecoregion, the estimated size of that ecoregion, \widehat{N}_d is also large. By dividing the large estimated total by the large estimated size, a more stable estimate of the mean of the domain is obtained. For quite a few ecoregions the standard errors are very large, especially of the π estimator. These are the ecoregions with very small average sample sizes. With simple random sampling the expected sample size can simply be computed by $E[n] = n N_d/N$. In the following section alternative estimators are described for these ecoregions with small expected sample sizes. To speed up the computations I used a $5 \text{ km} \times 5 \text{ km}$ subgrid in this sampling experiment.

No covariates are used in the ratio estimator. If we wish to exploit covariates, the mean of a domain can best be estimated by the ratio of the regression estimate of the domain total (Chapter 10) and the estimated size of the domain:

$$\hat{z}_{\text{ratio},d} = \frac{\hat{t}_{\text{regr},d}(z)}{\widehat{N}_d}. \quad (14.7)$$

14.2 Model-assisted estimators for small domains

When the domains are not well represented in the sample, the direct estimators from the previous section lead to large standard errors. In this situation we may try to increase the precision by also using observations from outside the domain. If we have covariates related to the study variable, we may exploit this ancillary information by fitting a regression model relating the study variable to the covariates, and using the fitted model to predict the study variable for all population units (nodes of discretisation grid), see Chapter 10. However, for small domains we may have too few sampled units in that domain to fit a separate regression model. The alternative then is to use the entire sample to estimate the regression coefficients and to use this global regression model to estimate the means of the domains. This introduces a systematic error, a design-bias, in the estimator. However, this extra error is potentially outweighed by the reduction of the random error due to the use of the globally estimated regression coefficients. If one or more units are selected from a domain, the observations of the study variable on these units can be used to correct for the bias. This leads

to the regression estimator for small domains. In the absence of such data, the mean of the domain can still be estimated by the so-called synthetic estimator.

There are quite a few packages for model-assisted estimation of means of small areas, the **maSAE** package (?), the **JoSAE** package (?), the **rsae** package (?), and the **forestinventory** package (?). I use package **forestinventory** for model-assisted estimation, and package **JoSAE** for model-based prediction of the means of small areas.

14.2.1 Regression estimator

In the regression estimator the potential bias due to the globally estimated regression coefficients, can be eliminated by adding the π estimator of the mean of the regression residuals to the mean of the predictions in the domain (compare with Equation (10.8)) (? , ?):

$$\hat{z}_{\text{regr},d} = \frac{1}{N_d} \sum_{k=1}^{N_d} \mathbf{x}_{dk}^T \hat{\mathbf{b}} + \frac{1}{N_d} \sum_{k \in \mathcal{S}_d} \frac{\epsilon_{dk}}{\pi_{dk}} = \bar{\mathbf{x}}_d^T \hat{\mathbf{b}} + \frac{1}{N_d} \sum_{k \in \mathcal{S}_d} \frac{\epsilon_{dk}}{\pi_{dk}}, \quad (14.8)$$

with \mathbf{x}_{dk} the vector with covariate values for unit k in domain d , $\hat{\mathbf{b}}$ the vector with globally estimated regression coefficients, ϵ_{dk} the residual for unit k in domain d , π_{dk} the inclusion probability of that unit, and $\bar{\mathbf{x}}_d$ the mean of the covariates in domain d . Alternatively the mean of the residuals in a domain is estimated by the ratio estimator:

$$\hat{z}_{\text{regr},d} = \bar{\mathbf{x}}_d^T \hat{\mathbf{b}} + \frac{1}{\hat{N}_d} \sum_{k \in \mathcal{S}_d} \frac{\epsilon_{dk}}{\pi_{dk}}, \quad (14.9)$$

with \hat{N}_d the estimated size of domain d , see Equation (14.2). The regression coefficients can be estimated by Equation (10.15). With simple random sampling the second term in this estimator is equal to the sample mean of the residuals, so that the estimator reduces to

$$\hat{z}_{\text{regr},d} = \bar{\mathbf{x}}_d^T \hat{\mathbf{b}} + \bar{\epsilon}_{\mathcal{S}_d}, \quad (14.10)$$

with $\bar{\epsilon}_{\mathcal{S}_d}$ the sample mean of the residuals in domain d .

A regression estimate can only be computed if we have at least one observation of the study variable in the domain d . The variance of the regression estimator of the mean for a small domain can be estimated by (?)

$$\widehat{V}(\hat{z}_{\text{regr},d}) = \bar{\mathbf{x}}_d^T \widehat{\mathbf{C}}(\hat{\mathbf{b}}) \bar{\mathbf{x}}_d + \widehat{V}(\hat{\epsilon}_d), \quad (14.11)$$

with $\widehat{\mathbf{C}}(\hat{\mathbf{b}})$ the matrix with estimated sampling variances and sampling covariances of the regression coefficients. The first variance component is the contribution due to uncertainty about the regression coefficients, the second component accounts for the uncertainty about the mean of the residuals in the domain. For simple random sampling the sampling variance of the π estimator of the mean of the residuals in a domain can be estimated by the sample variance of the residuals in that domain divided by the sample size n_d . This variance estimator is presented in ?. If the domain is not used as a stratum, and the domain mean of the residuals is estimated by the ratio estimator, the second variance component can be estimated by

$$\widehat{V}(\hat{e}_{\text{ratio},d}) = \frac{1}{\hat{a}_d^2} \cdot \frac{1}{n(n-1)} \sum_{k \in \mathcal{S}_d} (\epsilon_{dk} - \bar{\epsilon}_{\mathcal{S}_d})^2, \quad (14.12)$$

where \hat{a}_d is the estimated relative size of domain d :

$$\hat{a}_d = \frac{n_d}{n}. \quad (14.13)$$

With simple random sampling the sampling variances and covariances of the estimated regression coefficients can be estimated by (see Equation 2 in ?)

$$\widehat{\mathbf{C}}(\hat{\mathbf{b}}) = \frac{1}{n} \left(\sum_{k \in \mathcal{S}} \mathbf{x}_k \mathbf{x}_k^T \right)^{-1} \left(\frac{1}{n^2} \sum_{k \in \mathcal{S}} \epsilon_k^2 \mathbf{x}_k \mathbf{x}_k^T \right) \frac{1}{n} \left(\sum_{k \in \mathcal{S}} \mathbf{x}_k \mathbf{x}_k^T \right)^{-1}. \quad (14.14)$$

Note that these sampling variances and covariances are not equal to the model variances and covariances of the estimated regression coefficients as obtained with multiple linear regression, using functions `lm` and `vcov`, see Chapter 27.

Function `twophase` of package **forestinventory** (?) can be used to compute the regression estimator for small domains and its standard error. The name

‘twophase’ is somewhat confusing. It suggests that we have a large sample which is subsampled in a second phase, as described in Chapter 11. This is not the case here. However, `? twophase` considers infinite populations, and they treat the grid that discretises this infinite population as the first phase sample. The sampling error introduced by this discretisation grid can then be accounted for. I ignore this sampling error, it will be very small anyway, because the number of grid cells is very large. This can be done by assigning the domain means of the covariates to argument `exhaustive` of function `twophase`. Function `twophase` assumes simple random sampling (unless optional argument `cluster` is used). Note that for the unobserved population units (not selected units) the AGB values are changed into non-availables. In package `survey` also a function `twophase` is defined, for that reason the name of the package is made explicit by `forestinventory::twophase`. With arguments `psmall=TRUE` and element `unbiased=TRUE` in the list `small_area` the regression estimate is computed.

```
library(forestinventory)
n <- 200
set.seed(314)
units <- sample.int(nrow(gridAmazonia), size=n, replace=FALSE)
gridAmazonia$ind <- rep(0,nrow(gridAmazonia))
gridAmazonia$ind[units] <- 1
gridAmazonia$AGB[gridAmazonia$ind == 0] <- NA
mx_eco_pop <- tapply(
  gridAmazonia$lnSWIR2, INDEX=gridAmazonia$Ecoregion, FUN=mean)
mX_eco_pop <- data.frame(
  Intercept=rep(1,length(mx_eco_pop)), lnSWIR2=mx_eco_pop)
ecos_in_sam <- unique(mysample$Ecoregion)
res <- forestinventory::twophase(
  AGB~lnSWIR2,
  data=gridAmazonia,
  phase_id=list(
    phase.col="ind",
    terrgrid.id=1),
  small_area=list(
    sa.col="Ecoregion",
    areas=sort(ecos_in_sam),
    unbiased=TRUE),
  psmall=TRUE,
```

Table 14.2: Regression estimates of mean aboveground biomass (AGB) of ecoregions in Eastern Amazonia, using lnSWIR2 as a predictor. For explanation of variances of regression estimator, see text. In the final column the number of sampling units per ecoregion is listed.

Ecoregion	AGB	ext_var	g_var	n2G
Cerrado	105.5	51.6	82.6	12
Guianan highlands moist forests	281.8	NA	NaN	1
Guianan lowland moist forests	280.3	NA	NaN	1
Gurupa varzea	57.8	NA	NaN	1
Madeira-Tapajos moist forests	296.1	91.2	107.3	19
Marajo varzea	163.1	535.2	546.0	10
Maranhao Babassu forests	114.2	157.1	178.1	7
Mato Grosso tropical dry forests	144.6	959.6	981.5	2
Monte Alegre varzea	209.1	4105.7	4117.3	2
Purus-Madeira moist forests	225.7	2428.9	2441.1	2
Tapajos-Xingu moist forests	277.8	23.4	36.9	38
Tocantins/Pindare moist forests	157.8	75.8	87.7	30
Uatuma-Trombetas moist forest	270.6	33.3	47.5	46
Xingu-Tocantins-Araguaia moist forests	223.6	51.1	61.6	29

```
exhaustive=mX_eco_pop)
regr <- res$estimation
```

The alternative is to save the selected units (the sample) in a `data.frame`, assigned to argument `data`. The results are identical because the true means of the covariate x assigned to argument `exhaustive` contains all required information at the population level.

For two ecoregions no estimate of the mean AGB is obtained (Table 14.2). No units are selected from these domains. The variance of the estimated domain mean is in the column `g_variance`. The column `ext_variance` ignores the first variance component of Equation (14.11). Note that for the ecoregions with a sample size of one unit (the sample sizes per domain are in column `n2G`) no estimate of the variance is available, because the variance of the estimated mean of the residuals cannot be estimated from one unit.

Figure 14.1 shows the regression estimates plotted against the ratio estimates. The variation of the regression estimates is smaller than those of the ratio estimates. The intercept of the line, fitted with ordinary least squares (OLS), is larger than 0, and the slope is smaller than 1. Using the regression model predictions in the estimation of the means leads to some smoothing.

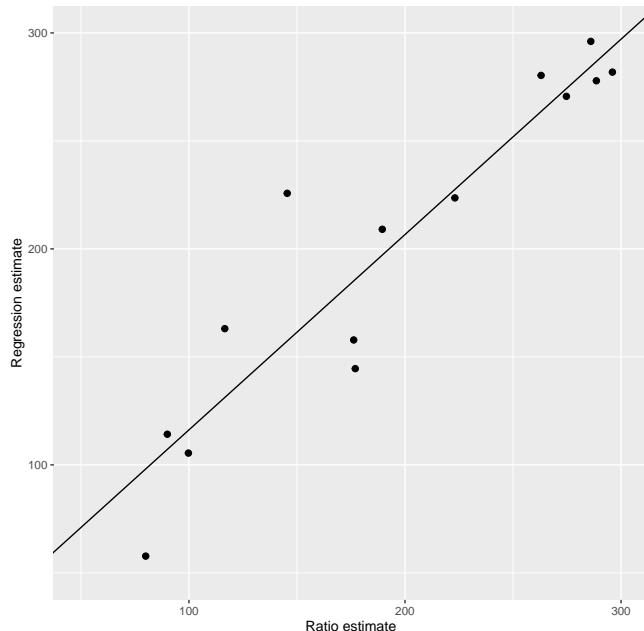


Figure 14.1: Scatterplot of ratio estimate and regression estimate of mean aboveground biomass (AGB) of ecoregions in Eastern Amazonia. In the regression estimate $\ln(\text{SWIR}2)$ is used as a predictor. The line is fitted by ordinary least squares.

I quantified the gain in the precision due to the use of the regression model by the variance of the ratio estimator divided by the variance of the regression estimator (Table 14.3). For ratios larger than 1 there is a gain in precision. Both variances are estimated from the 1,000 repeated ratio and regression estimates. For all but two small ecoregions there is a gain. For quite a few ecoregions the gain is quite large. These are the ecoregions where the globally fitted regression model explains a large part of the spatial variation of AGB.

Table 14.3: Gain in precision of estimated mean of AGB of ecoregions in Eastern Amazonia, as quantified by the estimated variance of the ratio estimator (no covariate used) divided by the estimated variance of the regression estimator (using lnSWIR2 as a predictor).

Ecoregions	Gain
Amazon-Orinoco-Southern Caribbean mangroves	0.57
Cerrado	1.63
Guianan highlands moist forests	1.01
Guianan lowland moist forests	1.26
Guianan savanna	6.70
Gurupa varzea	0.65
Madeira-Tapajos moist forests	1.44
Marajo varzea	1.87
Maranhao Babassu forests	1.70
Mato Grosso tropical dry forests	2.81
Monte Alegre varzea	1.21
Purus-Madeira moist forests	1.74
Tapajos-Xingu moist forests	2.73
Tocantins/Pindare moist forests	2.56
Uatuma-Trombetas moist forest	2.10
Xingu-Tocantins-Araguaia moist forests	4.06

14.2.2 Synthetic estimator

For small domains from which no units are selected, the mean can still be estimated by the synthetic estimator, also referred to as the synthetic regression estimator:

$$\hat{\bar{z}}_{\text{syn},d} = \bar{\mathbf{x}}_d^T \hat{\mathbf{b}} . \quad (14.15)$$

So the second term in Equation (14.8) is dropped. The variance can be estimated by

$$\widehat{V}\left(\hat{\bar{z}}_{\text{syn},d}\right) = \bar{\mathbf{x}}_d^T \widehat{\mathbf{C}}(\hat{\mathbf{b}}) \bar{\mathbf{x}}_d . \quad (14.16)$$

This is equal to the first variance component of Equation (14.11). The synthetic estimate can be computed with function `twophase`, with arguments `psmall=FALSE` and element `unbiased=FALSE` in the list `small_area`.

```
res <- forestinventory::twophase(
  AGB~lnSWIR2,
  data=gridAmazonia,
  phase_id=list(
    phase.col="ind",
    terrgrid.id=1),
  small_area=list(
    sa.col="Ecoregion",
    areas=ecoregions,
    unbiased=FALSE),
  psmall=FALSE,
  exhaustive=mX_eco_pop)
synt <- res$estimation
```

For all ecoregions, also the unsampled ones, a synthetic estimate of the mean AGB is obtained (Table 14.4). For the sampled ecoregions the synthetic estimate differs from the regression estimate. This difference can be quite large for ecoregions with a small sample size. Averaged over all sampled ecoregions the difference, computed as synthetic estimate minus regression estimate, equals 14.9. The variance of the regression estimator is always much larger than the variance of the synthetic estimator. The difference is the variance of the estimated domain mean of the residuals. However, recall that the regression estimator is design-unbiased, whereas the synthetic estimator is not. A more fair comparison is on the basis of the root mean squared error (Table 14.5).

In the synthetic estimator and the regression estimator both quantitative covariates and categorical variables can be used. If one or more categorical variables are included in the estimator, the variable names in the data frame with the true means of the ancillary variables per domain, assigned to argument `exhaustive`, must correspond with the column names of the design matrix that is generated with function `lm`, see Section 10.1.3.

Table 14.4: Synthetic estimates of mean aboveground biomass (AGB) of ecoregions in Eastern Amazonia, using lnSWIR2 as a predictor. In the final column the number of sampling units per ecoregion is listed.

Ecoregion	AGB	g_var	n2G
Amazon-Orinoco-Southern Caribbean mangroves	225.3	10.6	0
Cerrado	81.6	31.0	12
Guianan highlands moist forests	278.6	16.0	1
Guianan lowland moist forests	277.3	15.8	1
Guianan savanna	171.8	12.2	0
Gurupa varzea	218.2	10.4	1
Madeira-Tapajos moist forests	279.2	16.1	19
Marajo varzea	230.6	10.8	10
Maranhao Babassu forests	118.3	20.9	7
Mato Grosso tropical dry forests	114.0	21.9	2
Monte Alegre varzea	242.6	11.6	2
Purus-Madeira moist forests	250.4	12.3	2
Tapajos-Xingu moist forests	261.1	13.4	38
Tocantins/Pindare moist forests	175.7	11.9	30
Uatuma-Trombetas moist forest	267.2	14.2	46
Xingu-Tocantins-Araguaia moist forests	221.9	10.5	29

14.3 Model-based prediction

The alternative for design-based and model-assisted estimation of the means or totals of small domains is model-based prediction. The model describes the spatial variation of the study variable. An example of such a model is a linear regression model. Such model is used in the model-assisted estimator of the previous sections. A more advanced model is a linear mixed model. In a linear mixed model the mean of the study variable is modelled as a linear combination of covariates, similar to a linear regression model. The difference with a linear regression model is that also the variance of the residuals is modelled.

Table 14.5: Standard error (se), bias, and root mean squared error (RMSE) of the regression estimator (reg) and synthetic estimator (syn). The regression estimator is design-unbiased, so the RMSE of the regression estimator is equal to its standard error.

Ecoregion	se.reg	se.syn	bias.syn	RMSE.syn
Amazon-Orinoco Carib. mangroves	48.8	3.7	96.2	96.3
Cerrado	14.2	7.1	-17.3	18.7
Guianan highlands moist forests	29.4	4.9	-1.2	5.0
Guianan lowland moist forests	15.9	4.8	-7.2	8.7
Guianan savanna	25.9	4.1	12.3	13.0
Gurupa varzea	71.8	3.7	119.4	119.5
Madeira-Tapajos moist forests	11.8	4.9	-2.2	5.4
Marajo varzea	18.1	3.8	74.4	74.5
Maranhao Babassu forests	19.7	5.7	3.9	6.9
Mato Grosso tropical dry forests	24.5	5.8	3.4	6.7
Monte Alegre varzea	57.5	3.9	50.6	50.8
Purus-Madeira moist forests	42.1	4.2	32.9	33.1
Tapajos-Xingu moist forests	7.3	4.4	-18.5	19.0
Tocantins/Pindare moist forests	9.5	4.0	14.7	15.3
Uatuma-Trombetas moist forest	5.6	4.5	-14.2	14.9
Xingu-Toc.-Arag. moist forests	8.5	3.7	-2.5	4.4

14.3.1 Random intercept model

A basic linear mixed model for model-based prediction of means of small domains is the random intercept model:

$$\begin{aligned}
 Z_{dk} &= \mathbf{x}_{dk}^T \beta + v_d + \epsilon_{dk} \\
 v_d &\sim \mathcal{N}(0, \sigma_v^2) \\
 \epsilon_{dk} &\sim \mathcal{N}(0, \sigma_\epsilon^2).
 \end{aligned} \tag{14.17}$$

Two random variables are now involved, both with a normal distribution with mean zero: v_d a random intercept at the domain level with variance σ_v^2 , and the

residuals ϵ_{dk} at the unit level with variance σ_ϵ^2 . The variance σ_v^2 can be interpreted as a measure of the heterogeneity among the domains after accounting for the fixed effect (?). With this model the mean of a domain can be predicted by

$$\hat{\bar{z}}_{d,\text{mb}} = \bar{\mathbf{x}}_d^T \hat{\mathbf{b}} + \hat{v}_d , \quad (14.18)$$

with $\hat{\beta}$ the best linear unbiased estimates (BLUE) of the regression coefficients, and \hat{v}_d the best linear unbiased prediction (BLUP) of the intercept for domain d , v_d . The model-based predictor can also be written as

$$\hat{\bar{z}}_{d,\text{mb}} = \bar{\mathbf{x}}_d^T \hat{\mathbf{b}} + \lambda_d \left(\frac{1}{n_d} \sum_{k \in \mathcal{S}_d} \epsilon_{dk} \right) , \quad (14.19)$$

with λ_d a weight for the second term that corrects for the bias of the synthetic estimator. This weight is computed by

$$\lambda_d = \frac{\hat{\sigma}_v^2}{\hat{\sigma}_v^2 + \hat{\sigma}_\epsilon^2 / n_d} . \quad (14.20)$$

This equation shows that the larger the estimated residual variance $\hat{\sigma}_\epsilon^2$, the smaller the weight for the bias correction factor, and the larger the sample size n_d , the larger the weight. Comparing Equations (14.18) and (14.19) shows that the random intercept of a domain is predicted by the sample mean of the residuals of that domain, multiplied by a weight factor computed by Equation (14.20).

The means of the small domains can be computed with function `eblup.mse.f.wrap` of package **JosAE** (?). It requires as input a linear mixed model generated with function `lme` of package **nlme** (?). The simple random sample of size 200 is used to fit the linear mixed model, with `lnSWIR2` as a fixed effect, i.e. the effect of SWIR2 on the mean of AGB. The random effect is added by assigning another formula to the argument `random`. The formula, `~1|Ecoregions` means that the intercept is treated as a random variable, and that it varies among the Ecoregions. This linear mixed model is referred to as a random intercept model: the intercepts are allowed to differ among the small domains, whereas the effects of the covariates, `lnSWIR2` in our case, is equal for all domains.

```
library(nlme)
library(JoSAE)
lmm_AGB <- lme(fixed=AGB~lnSWIR2, data=mysample, random=~1|Ecoregion)
```

The fixed effects can be extracted with function `fixed.effects`.

```
fixed_lmm <- fixed.effects(lmm_AGB)
```

The fixed effects differ somewhat from the fixed effects in the simple linear regression model:

```
fixed.lm fixed.lmm
(Intercept) 1778.1959 1667.9759
lnSWIR2      -241.1567 -225.6561
```

The random effect can be extracted with function `random.effect`.

```
random.effects(lmm_AGB)
```

	(Intercept)
Cerrado	21.439891
Guianan highlands moist forests	6.397816
Guianan lowland moist forests	5.547995
Gurupa varzea	-52.985839
Madeira-Tapajos moist forests	27.479921
Marajo varzea	-50.587786
Maranhao Babassu forests	-1.702322
Mato Grosso tropical dry forests	18.812207
Monte Alegre varzea	-12.201148
Purus-Madeira moist forests	-9.320683
Tapajos-Xingu moist forests	28.760508
Tocantins/Pindare moist forests	-8.940962
Uatumá-Trombetas moist forest	16.165017
Xingu-Tocantins-Araguaia moist forests	11.135385

The random intercepts are added to the fixed intercept; the coefficient of lnSWIR2 is the same for all ecoregions:

```
coef(lmm_AGB)

(Intercept)    lnSWIR2
Cerrado          1689.416 -225.6561
Guianan highlands moist forests      1674.374 -225.6561
Guianan lowland moist forests        1673.524 -225.6561
Gurupa varzea           1614.990 -225.6561
Madeira-Tapajos moist forests       1695.456 -225.6561
Marajo varzea            1617.388 -225.6561
Maranhao Babassu forests          1666.274 -225.6561
Mato Grosso tropical dry forests   1686.788 -225.6561
Monte Alegre varzea             1655.775 -225.6561
Purus-Madeira moist forests        1658.655 -225.6561
Tapajos-Xingu moist forests        1696.736 -225.6561
Tocantins/Pindare moist forests    1659.035 -225.6561
Uatumá-Trombetas moist forest     1684.141 -225.6561
Xingu-Tocantins-Araguaia moist forests 1679.111 -225.6561
```

The fitted model can now be used to predict the means of the ecoregions as follows. As a first step a data frame must be defined with the sizes of the domains and the population means of the covariate lnSWIR2 per domain. This data frame is assigned to argument `domain.data` of function `eblup.mse.f.wrap`. This function computes the model-based prediction as well as the regression estimator (Equation (14.8)) and the synthetic estimator (Equation (14.15)) and their variances. The model-based predictor is the variable `EBLUP` in the output data frame. For the model-based predictor two standard errors are computed, see `? eblup.mse.f.wrap` for details.

```
N_eco <- tapply(gridAmazonia$AGB, INDEX=gridAmazonia$Ecoregion,
                  FUN=length)
df_eco <- data.frame(Ecoregion=ecoregions, N=N_eco,
                      lnSWIR2=mx_eco_pop)
res <- eblup.mse.f.wrap(
  domain.data=df_eco, lme.obj=lmm_AGB)
df <- data.frame(
  Ecoregion=res$domain.ID, mb=res$EBLUP,
  se.1=res$EBLUP.se.1, se.2=res$EBLUP.se.2)
```

Table 14.6: Model-based predictions of mean aboveground biomass (AGB) of ecoregions in Eastern Amazonia, obtained with random intercept model and lnSWIR2 as a predictor.

Ecoregion	AGB	se.1	se.2
Cerrado	101.9	11.4	11.5
Guianan highlands moist forests	271.1	25.8	26.4
Guianan lowland moist forests	269.1	25.8	26.4
Gurupa varzea	155.3	36.0	31.8
Madeira-Tapajos moist forests	292.8	9.3	9.3
Marajo varzea	169.3	13.6	13.2
Maranhao Babassu forests	113.1	14.1	14.4
Mato Grosso tropical dry forests	129.6	22.9	23.1
Monte Alegre varzea	218.9	22.2	22.7
Purus-Madeira moist forests	229.0	22.2	22.7
Tapajos-Xingu moist forests	277.2	6.7	6.7
Tocantins/Pindare moist forests	159.6	7.4	7.5
Uatuma-Trombetas moist forest	270.2	6.0	6.0
Xingu-Tocantins-Araguaia moist forests	222.9	7.5	7.6

Note that with this model no predictions are obtained for the unsampled ecoregions. For the unsampled ecoregions no prediction is obtained of the random intercept v_d , see Equation (14.19).

14.3.2 Geostatistical model

In the random intercept model (Equation (14.18)) it is assumed that the residuals ϵ_{ij} are independent. In a geostatistical model this assumption is relaxed, and the spatial correlation of the residuals is modelled. For details about geostatistical modelling, I refer to Chapter 22.

A simple random sample of size 200 is used to compute the sample semivariogram of the residuals of the simple linear regression model, using lnSWIR2 as a single predictor. Figure 14.2 shows that the larger the distance, the larger half the expected squared difference. This shows that the residuals of units with a separation distance smaller than about 300 km are correlated. A spherical model with nugget is fitted to model the spatial correlation of the residuals e .

The residual semivariogram is computed with function `variogram` of package `gstat` (?). A spherical model is fitted with function `fit.variogram`. The coordinates are shifted to a random point within a 5 km \times 5 km grid cell. This is only done to avoid an error message when the geostatistical model is used to predict AGB at the nodes of the grid hereafter.

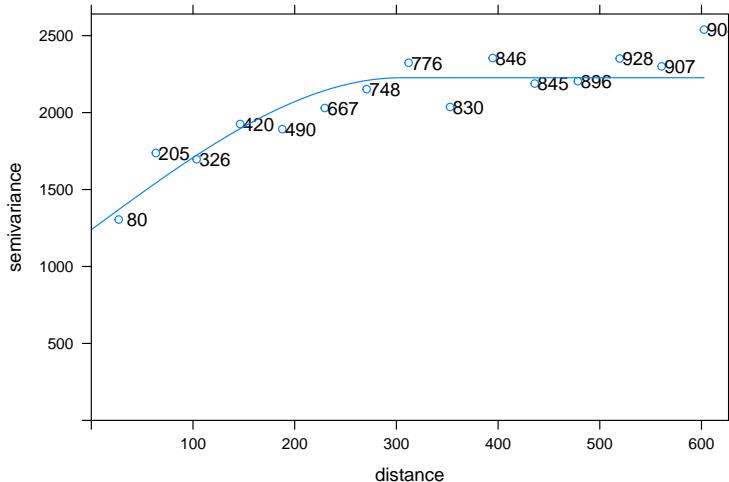


Figure 14.2: Semivariogram of residuals of simple linear regression model for AGB, using lnSWIR2 as a predictor.

The semivariogram parameters and the regression coefficients can best be estimated by restricted maximum likelihood (REML), see Section 22.4.2. This estimation procedure is also used in function `lme` to fit the random intercept model. Here I use function `likfit` of package `geoR` (?). First a `geoR` object must be generated with function `as.geodata`.

```
library(geoR)
mysample <- as(mysample, "data.frame")
dGeoR <- as.geodata(
  mysample, header=TRUE,
  coords.col=1:2, data.col=3, covar.col=4)
```

The model parameters can then be estimated with function `likfit`.

```
vgm_REML <- likfit(
  geodata=dGeoR,
  trend=~lnSWIR2,
  cov.model="spherical",
  ini.cov.pars=c(1000,300), nugget=1500,
  lik.method="REML", messages=FALSE)
```

The estimated semivariogram parameters are 1622 for the nugget (intercept of the semivariogram), 700 for the partial sill (maximum semivariance minus the nugget), and 650.6 for the range (distance at which the semivariance reaches its maximum). The estimated regression coefficients and parameters of the semivariogram can then be used to predict AGB for all units in the population.

```
coordinates(mysample) <- ~x1+x2
vgm_REML_gstat <- vgm(
  nugget=vgm_REML$nugget, psill=vgm_REML$sigmasq,
  range=vgm_REML$phi, model="Sph")
coordinates(mysample) <- ~x1+x2
coordinates(gridAmazonia) <- ~x1+x2
predictions <- krige(
  formula=AGB~lnSWIR2,
  locations=mysample,
  newdata=gridAmazonia,
  model=vgm_REML_gstat,
  debug.level=0) %>%
as(.,"data.frame")
```

The first six rows of `predictions` are shown below.

	x1	x2	var1.pred	var1.var
1	-6628.193	188.4642	295.7996	1987.310
2	-6627.193	188.4642	294.7392	1986.003
3	-6626.193	188.4642	287.2151	1984.183
4	-6625.193	188.4642	280.0769	1982.710
5	-6624.193	188.4642	290.3131	1982.188
6	-6623.193	188.4642	299.4357	1982.078

Besides a prediction (column `var1.pred`), for every population unit the variance

of the prediction error is computed (`var1.var`). The unit-wise predictions can be averaged across all units of an ecoregion to obtain a model-based prediction of the mean of that ecoregion.

```
AGBpred_unit <- predictions$var1.pred
gridAmazonia <- as(gridAmazonia, "data.frame")
mz_eco_mb <- tapply(AGBpred_unit, INDEX=gridAmazonia$Ecoregion,
                     FUN=mean) %>% round(.,1)
```

A difficulty is the computation of the standard error of these model-based predictions of the ecoregion mean. We cannot simply sum the unit-wise variances and divide the sum by the squared number of units because the prediction errors of units with a mutual distance smaller than about 450 km (the estimated range of the semivariogram) are correlated. A straightforward approach to obtain the standard error of the predicted mean is geostatistical simulation. A large number of maps is simulated, conditional on the selected sample. For an infinite number of maps, the “average map”, i.e. the map obtained by averaging for each unit all simulated values of that unit, is equal to the map with predicted AGB. For each simulated map, the average of the simulated values across all units of an ecoregion is computed. This results in as many averages as we have simulated maps. The variance of the averages of an ecoregion is an estimate of the variance of the predicted mean of that ecoregion. To reduce computing time the 5 km × 5 km subgrid is used in geostatistical simulation

```
load(file="data/Amazonia_5km.RData")
nsim <- 1000
coordinates(gridAmazonia) <- ~x1+x2
simulations <- krige(
  formula=AGB~lnSWIR2,
  locations=mysample,
  newdata=gridAmazonia,
  model=vgm_REML_gstat,
  nmax=100, nsim=nsim,
  debug.level=0) %>%
  as(., "data.frame")
gridAmazonia <- as(gridAmazonia, "data.frame")
AGBsim_eco <- matrix(nrow=length(ecoregions), ncol=nsim)
```

Table 14.7: Model-based predictions of mean aboveground biomass (AGB) of ecoregions in Eastern Amazonia, obtained with geostatistical model and lnSWIR2 as a predictor for the mean. se: standard error of predicted mean.

Ecoregion	AGB	se
Amazon-Orinoco-Southern Caribbean mangroves	191.1	22.4
Cerrado	96.8	12.1
Guianan highlands moist forests	299.0	28.1
Guianan lowland moist forests	279.3	16.4
Guianan savanna	166.0	10.2
Gurupa varzea	201.3	15.6
Madeira-Tapajos moist forests	287.0	8.6
Marajo varzea	194.6	11.3
Maranhao Babassu forests	121.5	12.0
Mato Grosso tropical dry forests	121.4	11.7
Monte Alegre varzea	241.3	10.8
Purus-Madeira moist forests	228.9	24.3
Tapajos-Xingu moist forests	273.8	8.2
Tocantins/Pindare moist forests	163.0	6.8
Uatumá-Trombetas moist forest	265.9	6.4
Xingu-Tocantins-Araguaia moist forests	220.0	6.2

```
for (i in 1:nSIM) {
  AGBsim_eco[,i] <- tapply(
    simulations[,i+2], INDEX=gridAmazonia$Ecoregion, FUN=mean)
}
```

Similar to the synthetic estimator, for all ecoregions an estimate of the mean AGB is obtained, also for the unsampled ecoregions (Table 14.7). The model-based prediction is strongly correlated with the synthetic estimate (Figure 14.3).

The most striking difference is the standard error. The standard errors of the synthetic estimator range from 3.2 to 4.3, whereas the standard errors of the geostatistical predictions range from 6.2 to 28.1. However, these two standard errors are fundamentally different, and should not be compared. The standard

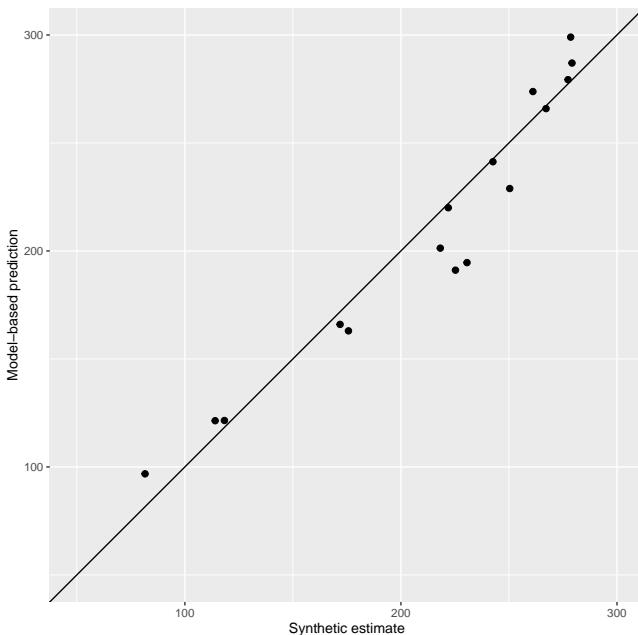


Figure 14.3: Scatterplot of model-based prediction and synthetic estimate of the mean AGB of ecoregions in Eastern Amazonia. The solid line is the 1:1 line.

error of the synthetic estimator is a *sampling* standard error, i.e. it quantifies the variation of the estimated mean of an ecoregion over repeated random sampling with the sampling design, in this case simple random sampling of 200 units. The model-based standard error is not a sampling standard error but a model standard error, which expresses our uncertainty about the means of the domains due to our imperfect knowledge of the spatial variation of AGB. Given the observations of AGB at the selected sample, the map with the covariate lnSWIR2, and the estimated semivariogram model parameters, we are uncertain about the exact value of AGB at unsampled units. No other samples are considered than the one actually selected. For the fundamental difference between the design-based, model-assisted, and model-based estimates of means, I refer to Section 1.2 and Chapter 27.

It makes more sense to compare the two model-based predictions, the random intercept model predictions and the geostatistical predictions, and their stan-

dard errors. Figure 14.4 shows that the two model-based predictions are very similar.

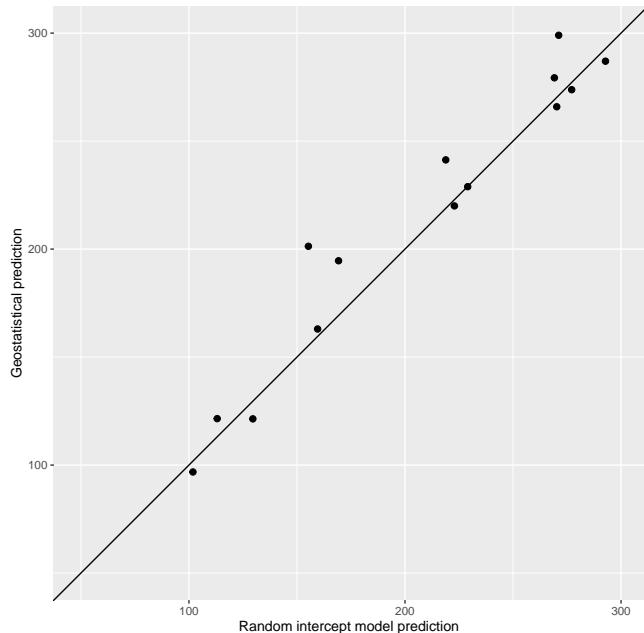


Figure 14.4: Scatterplot of model-based predictions of the mean AGB of ecoregions in Eastern Amazonia, obtained with the random intercept model and the geostatistical model. The solid lines is the 1:1 line.

For four ecoregions the standard errors of the geostatistical model predictions are much smaller than those of the random intercept model predictions (Figure 14.5). These are ecoregions with small sample sizes.

Note that if a different semivariogram model were used, both the predicted means per ecoregion and the standard errors would be different. Especially the variance is sensitive to the semivariogram. For that reason the model-based predictions are also referred to as model-dependent predictions.

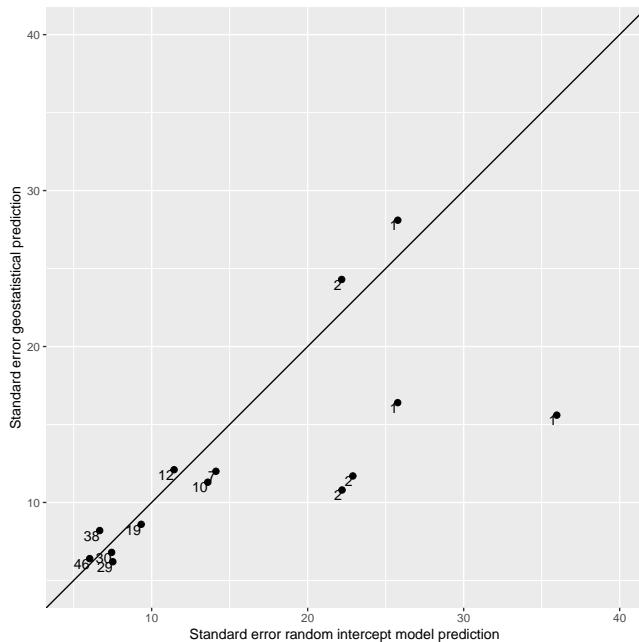


Figure 14.5: Scatterplot of standard error of model-based predictions of the mean AGB of ecoregions in Eastern Amazonia, obtained with the random intercept model and the geostatistical model. The numbers refer to the number of sampling units in an ecoregion. The solid line is the 1:1 line.

14.4 Supplemental probability sampling of small domains

The sample size in small domains of interest can be so small that no reliable statistical estimate of the mean or total of these domains can be obtained. In that case we may decide to collect a supplemental sample from these domains. It is convenient to use these domains as strata in supplemental probability sampling, so that we can control the sample sizes in the strata. If we can safely assume that the study variable at the units of the first sample are not changed, there is no need to revisit these units, otherwise we must revisit them to observe the current values.

There are two approaches for using the two samples to estimate the population mean or total of a small domain (?). In the first approach the two samples are combined, and then the merged sample is used to estimate the population mean or total. In the second approach not the samples are combined, but the two estimates from the separate samples. In this section only the first approach is illustrated with a simple situation in which the two samples are easily combined. I refer to ? for a more general approach of how multiple probability samples can be combined.

Suppose that the original sample is a simple random sample from the entire study area. A supplemental sample is selected from small domains, i.e. domains that have few selected units only. For a given small domain, the first sample is supplemented by selecting a simple random sample from the units not yet selected in the first sample. The size of the supplemental sample of a domain depends on the number of units of that domain in the first sample. The first sample is supplemented so that the total sample size of that domain is fixed. In this case the combined sample of a domain is a simple random sample from that domain, so that the usual estimators for simple random sampling can be used to estimate the domain mean or total, and its standard error.

This sampling strategy is illustrated with Eastern Amazonia. A simple random sample without replacement of 400 units is selected.

```
load("data/Amazonia_1km.RData")
gridAmazonia$Biome <- as.factor(gridAmazonia$Biome)
biomes <- c("Mangrove", "Forest.dry", "Grassland", "Forest.moist")
levels(gridAmazonia$Biome) <- biomes
n1 <- 400
set.seed(123)
units_1 <- sample.int(nrow(gridAmazonia), size=n1, replace=FALSE)
mysample_1 <- gridAmazonia[units_1, c("AGB", "Biome")]
print(n1.biome <- table(mysample_1$Biome))
```

Mangrove	Forest.dry	Grassland	Forest.moist
2	9	26	363

The selected units are removed from the sampling frame. For each of the three small biomes, Mangrove, Forest.dry and Grassland, the size of the supplemental sample is computed so that the total sample size becomes 40. The supplemental

sample is selected by stratified simple random sampling without replacement, using the small biomes as strata.

```
units_notselected <- gridAmazonia[-units_1,]
Biomes_NFM <-
  units_notselected[units_notselected$Biome!="Forest.moist",]
n_biome <- 40
n2_biome <- rep(n_biome,3) - n1_biome[-4]
ord <- unique(Biomes_NFM$Biome)
units_2 <- sampling::strata(
  Biomes_NFM, stratanames="Biome",
  size=n2_biome[ord], method="srswor")
mysample_2 <- getdata(Biomes_NFM, units_2)
mysample_2 <- mysample_2[c("AGB","Biome")]
```

The two samples are merged, and the means of the domains are estimated by the sample means.

```
mysample <- rbind(mysample_1, mysample_2)
print(mz_biome <- tapply(
  mysample$AGB, INDEX=mysample$Biome, FUN=mean))
```

	Mangrove	Forest.dry	Grassland	Forest.moist
	112.8000	122.1750	123.6250	233.6749

Finally, the standard error is estimated, accounting for sampling without replacement from a finite population (Equation (3.11)).

```
N_biome <- table(gridAmazonia$Biome)
fpc <- (1-n_biome/N_biome)
S2z_biome <- tapply(mysample$AGB, INDEX=mysample$Biome, FUN=var)
print(se_mz_biome <- sqrt(fpc*(S2z_biome/n_biome)))
```

	Mangrove	Forest.dry	Grassland	Forest.moist
	5.227434	8.213407	11.709257	13.937597

This sampling approach and estimation is repeated 10,000 times, i.e. 10,000 times a simple random sample of size 400 is selected from Eastern Amazonia,

Table 14.8: Summary statistics of 10,000 estimated means of small domains (biomes) in Eastern Amazonia, estimated by combining a simple random sample of size 400 from Eastern Amazonia, and a supplemental sample from the domains. The total sample size per small domain is 40.

	Mangrove	Dry forest	Grassland
Average of estimated means	122.476	113.983	114.462
True means	122.494	113.931	114.390
Standard deviation of estimated means	6.186	7.563	10.986
Average of estimated standard errors	6.169	7.450	10.946
Coverage rate 95%	0.950	0.937	0.945
Coverage rate 90%	0.903	0.888	0.896
Coverage rate 80%	0.801	0.792	0.799

and the samples from the three small domains are supplemented so that the total sample sizes in these domains become 40. In two out of the 10,000 samples the size of the first sample in one of the domains exceeded 40 units. These two samples are discarded. Ideally, these samples are not discarded, but their sizes in the small domains are reduced to 40 units, which are then used to estimate the means of the domains.

For all three small domains the average of the 10,000 estimated means is about equal to the true mean (Table 14.8). Also the mean of the 10,000 estimated standard errors is very close to the standard deviation of the 10,000 estimated means. The coverage rates of 95%, 90% and 80% confidence intervals are about equal to the nominal coverage rates.

This simple approach is feasible because at the domain level the two merged samples are a simple random sample. This approach is also applicable when at the first sample is a stratified simple random sample from the entire population and the supplemental sample is a stratified simple random sample from a small domain, using as strata the intersections of the strata used in the first phase and that domain.

Chapter 15

Repeated sample surveys for monitoring population parameters

The previous chapters are all about sampling to estimate population parameters *at a given time*. The survey is done in a relatively short period of time, so that we can safely assume that the study variable did not change during that period. This chapter is about repeating the sample survey two or more times, to estimate a temporal change in a population parameter. Sampling locations are selected by probability sampling, by any design type. Sampling times are not selected randomly, but purposively. For instance, to monitor the carbon stock in the soil of a country, we may decide to repeat the survey after five years, in the same season of the year as the first sampling round.

15.1 Space-time designs

? present an overview of space-time designs. Four of these designs are schematically shown in Figure 15.1. With repeated sampling in two-dimensional space there are three dimensions: two spatial and one time dimension. In Figure 15.1 the sampling locations in 2D are plotted in one dimension, along the horizontal axis. A selected unit along this axis actually represents a sampling location in

2D. Twenty sampling locations are selected by simple random sampling.

In the static-synchronous design, also referred to as a pure panel, all sampling locations selected in the first survey are revisited in the subsequent surveys. On the contrary, in an independent synchronous design, in each survey a probability sample is selected independent from the samples selected in the previous surveys. The serially alternating design is a compromise between a static-synchronous and an independent synchronous design. The sample selected in the first survey is revisited in the third survey. The sample of the second survey is selected independently from the sample of the first survey, and these locations are revisited in the fourth survey. In this case the period of revisits is two, i.e. two sampling intervals between successive surveys, but this can also be increased. For instance with a period of three, three samples are selected independently from each other for the first three surveys. The sample of the first survey is revisited in the fourth survey, et cetera. Another compromise design is a supplemented panel. In this space-time design only a subset of the sampling locations of the first survey is revisited in the subsequent surveys. The samples of the subsequent surveys are supplemented by samples that are selected independently from the samples in the previous surveys. In Figure 15.1, one-half of the sampling locations (ten locations) are fixed, i.e. revisited in all surveys, but the proportion of fixed sampling locations can be smaller or larger and, if prior information on the variation in space and time is available, even optimised.

In Figure 15.1 the shape and colour of the symbols represent a panel. A panel is a group of sampling locations that is observed in the same surveys. In the static-synchronous design there is only one panel. All locations are observed in all surveys, so all locations are in the same panel. In the independent synchronous design there are as many panels as there are surveys. In the serially alternating design with a period of two the number of panels equals the number of surveys divided by two. In these three space-time designs (SS, IS and SA) all sampling locations of a given survey are in the same panel. This is not the case in the supplemented panel design. In each survey two panels are observed, one panel of fixed sampling locations (pure panel part of sample), and another panel of swarming sampling locations. The concept of panels is needed hereafter in estimating space-time population parameters.

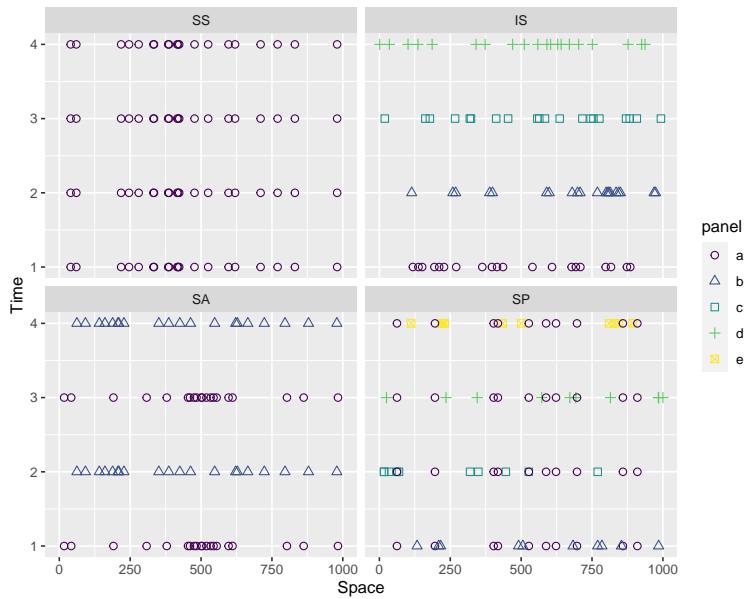


Figure 15.1: Space-time designs for monitoring population parameters. Twenty sampling locations are selected by simple random sampling. SS: static-synchronous design; IS: independent synchronous design; SA: serially alternating design; SP: supplemented panel design.

15.2 Space-time population parameters

The data of repeated surveys can be used to estimate various parameters. In this chapter I show how to estimate the current population parameter, i.e. the population parameter in the latest survey, the change of the population parameter between two surveys, and the temporal trend of the population parameter. The current population parameter need not be defined here as only one survey (one sampling time) is involved in this parameter, so that the definitions in Section 1.1.1 are also relevant here. The change of the population parameter is defined as the population parameter at a given survey minus this parameter at an earlier survey. For instance, the change of an infinite population mean is defined as

$$\bar{d}_{2,1} = \frac{1}{A} \left(\int_{\mathbf{s} \in \mathcal{U}} z(\mathbf{s}, t_2) d\mathbf{s} - \int_{\mathbf{s} \in \mathcal{U}} z(\mathbf{s}, t_1) d\mathbf{s} \right), \quad (15.1)$$

With more than two surveys an interesting population parameter is the *average change* per time unit of the population parameter, referred to as the temporal trend. It is defined as a linear combination of the population parameters at the sampling times. For instance, the temporal trend of the population mean is defined as (?)

$$b = \sum_{j=1}^r w_j \bar{z}_j, \quad (15.2)$$

with r the number of sampling times and \bar{z}_j the population mean at time t_j , and weights w_j equal to

$$w_j = \frac{t_j - \bar{t}}{\sum_{j=1}^r (t_j - \bar{t})^2}, \quad (15.3)$$

with \bar{t} the mean of the sampling times. Note that the temporal trend is defined as a parameter of a space-time population, not as a parameter of a time-series model.

Part II

Sampling for mapping

Chapter 16

Introduction to sampling for mapping

16.1 When is probability sampling not required?

This second part of the book deals with sampling for mapping, i.e. for predicting the study variable at the nodes of a fine discretisation grid. For mapping a model-based sampling approach is the most natural option. When a statistical model, i.e. a model containing an error term modelled by a probability distribution, is used to map the study variable from the sample data, selection of the sampling units by probability sampling is not strictly needed anymore in order to make statistical statements about the population, i.e. statements with quantified uncertainty, see Section 1.2. In a model-based approach we assume the model is correct, possibly after we have managed to verify the model assumptions. As a consequence there is room for optimising the sampling units, i.e. searching for those units that lead to the most accurate map in some operational sense related to map quality, for instance, the smallest squared prediction error averaged over all locations in the mapped study area, see Chapter 26.

As an illustration, consider the following statistical model to be used for mapping: a linear regression model between a single predictor and the target variable to be mapped:

$$Z_k = \beta_0 + \beta_1 x_k + \epsilon_k , \quad (16.1)$$

with Z_k the study variable of unit k , β_0 and β_1 regression coefficients and ϵ_k the error (residual) at unit k , normally distributed with mean zero and a constant variance σ^2 . The errors are assumed independent, so that $Cov(\epsilon_k, \epsilon_j) = 0$ for all $k \neq j$. Figure 16.1 shows a simple random sample without replacement and the sample optimised for mapping with a simple linear regression model. Both samples are plotted on a map of the single covariate (predictor).

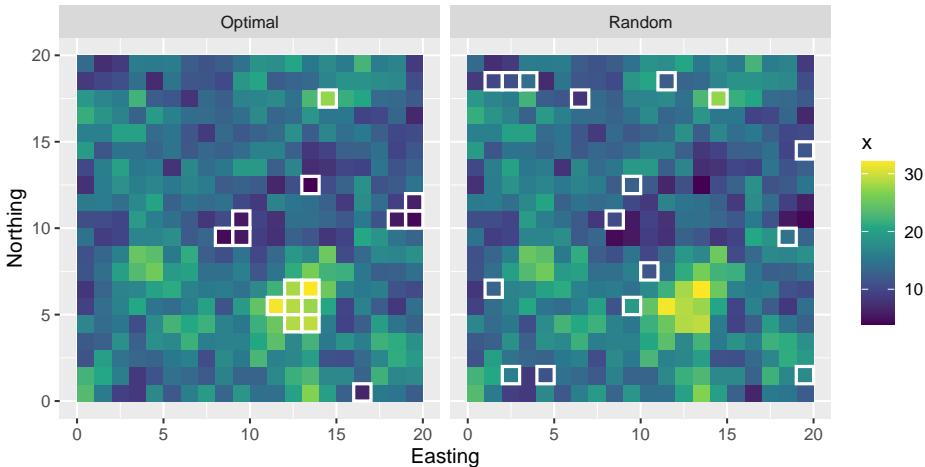


Figure 16.1: Simple random sample (a), and optimised sample for mapping with simple linear regression model (b), plotted in a map of the covariate.

The optimised sample for mapping with a simple linear regression model contains the units with the smallest or the largest values of the covariate x . The optimised sample shows strong spatial clustering. Spatial clustering is not avoided because in a simple linear regression model we assume that the residuals are not spatially correlated. The standard errors of both regression coefficients are considerably smaller for the optimised sample (Table 16.1). The joint uncertainty about the two regression coefficients, quantified by the determinant of the variance-covariance matrix of the estimated regression coefficients, equals 0.00388 for the simple random sample and 0.000999 for the optimised sample. When we are less uncertain about the regression coefficients, we are also less

Table 16.1: Standard errors and determinant of variance-covariance matrix of estimated regression coefficients for simple random sample and optimised sample.

Sampling design	se intercept	se slope	Determinant
SI	1.70	0.118	0.003880
Optimised	1.08	0.051	0.000999

uncertain about the regression model predictions of the study variable z for at points where we have observations of the covariate x only. So, we conclude that for mapping with a simple linear regression model, simple random sampling is not a good option if we are willing to rely on the assumption that there is a linear relation between z and x .

Of course, this simple example would only be applicable if we have evidence of the linear relation between covariate and target variable and independence of the residuals.

16.2 Sampling for simultaneously mapping and estimating means

Although probability sampling is not strictly needed for mapping with a statistical model, in some situations, when feasible, it can still be advantageous to select a probability sample. If the aim of the survey is to map the study variable, as well as to estimate the mean or total for the entire study area or for several subareas, probability sampling can be a good option. Think, for instance, of sampling for the dual objectives of mapping and at the same time estimating soil carbon stocks. Although the statistical model used for mapping can also be used for model-based prediction of the total carbon stocks in the study area and subareas (Section 14.3), we may prefer to estimate these totals by design-based (Section 14.1) or model-assisted inference (14.2). The advantage of design-based and model-assisted estimates of these totals is their validity. Validity means that an objective assessment of the uncertainty of the estimated mean or total is warranted, and that the coverage of confidence intervals is (almost) correct, provided that the sample is large enough to assume an approximately normal distribution of the estimator and design-unbiasedness of the variance estimator, see Chapter 27. In design-based estimation no model of the spatial variation is

used. Discussions about how realistic modelling assumptions are, therefore are avoided. In model-assisted estimation these discussions are irrelevant as well, because we do not rely on these assumptions. A poor model results in large variances of the estimated mean or total, and as a consequence a wide confidence interval, so that the coverage of the confidence interval is in agreement with the nominal coverage, see Section 27.6 for more details.

The question then is: what is a suitable probability sampling design for both aims? First, I would recommend a sampling design with equal inclusion probabilities. This is because in standard model-based inference unequal inclusion probabilities are not accounted for, which may lead to systematic prediction errors when small or large values are overrepresented in the sample (Section 27.3).

In case we have subareas of which we would like to estimate the mean or total (domains of interest), using these subareas as strata in stratified random sampling makes sense, unless there are too many. This requires that of all population units (nodes of discretisation grid) we must know to which subarea it belongs, so that this information can be added to the sampling frame.

A sampling design spreading out the sampling units in geographical space is attractive as well, for instance through compact geographical stratification (Section 4.6), or sampling with the local pivotal method (Section 9.2.1). We may profit from this geographical spreading if some version of kriging is used for mapping (Chapter 22). In addition, the geographical spreading may enhance the coverage of the space spanned by covariates related to the study variable. The alternative is to exploit these covariates explicitly in the sampling design, for instance through stratified random sampling from strata constructed by k-means, using the covariates and the spatial coordinates in clustering the population units (Section 4.5), or through balanced sampling with (geographical) spreading (Section 9.3).

As an illustration I selected a single sample of 500 units from Amazonia with the dual aim of mapping aboveground biomass (AGB) as well as estimating the means of AGB for four biomes. The biomes are used as strata in stratified random sampling. First the stratum sample sizes are computed for proportional allocation, so that the inclusion probabilities are approximately equal for all population units.

```

load("data/Amazonia_5km.RData")
gridAmazonia$Biome <- as.factor(gridAmazonia$Biome)
biomes <- c("Mangrove", "Forest.dry", "Grassland", "Forest.moist")
levels(gridAmazonia$Biome) <- biomes
N_h <- table(gridAmazonia$Biome)
n <- 500
n_h <- round(n*N_h/sum(N_h))
n_h[3] <- n_h[3]+1
print(n_h)

```

Mangrove	Forest.dry	Grassland	Forest.moist
2	11	28	459

Biome Forest.moist is by far the largest biome with a sample size of 459 points. The points from this biome are selected by balanced sampling with spreading in geographical space, using the covariate from remote sensing lnSWIR2 as a balancing variable.

```

Biome_FM <- gridAmazonia[gridAmazonia$Biome=="Forest.moist",]
library(BalancedSampling)
Xbal <- cbind(rep(1, times=N_h[4]), Biome_FM$lnSWIR2)
Xspread <- cbind(Biome_FM$x1, Biome_FM$x2)
pi <- rep(n_h[4]/N_h[4], times=N_h[4])
set.seed(314)
units <- lcube(Xbal=Xbal, Xspread=Xspread, prob=pi)
mysample_FM <- Biome_FM[units,]

```

The sample sizes of the other three biomes are rather small, so I decided to keep it simple by selecting simple random samples from these three biomes.

```

library(sampling)
Biomes_NFM <- gridAmazonia[gridAmazonia$Biome!="Forest.moist",]
ord <- unique(Biomes_NFM$Biome)
units <- sampling::strata(
  Biomes_NFM, stratanames="Biome", size=n_h[ord], method="srswor")
mysample_NFM <- getdata(Biomes_NFM, units)

```

Figure 16.2 shows the selected sample.

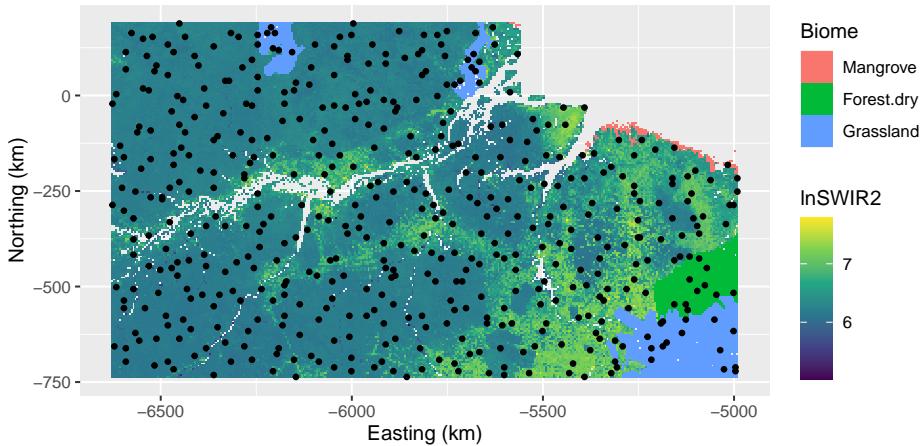


Figure 16.2: Balanced sample with geographical spreading from biome moist tropical forest, balanced on lnSWIR, and stratified simple random sample from other three biomes, Eastern Amazonia (Brazil).

16.2.1 Balanced stratified random sampling

A somewhat more advanced sampling design is balanced sampling, using both the categorical variable biome and the continuous variable lnSWIR2 as balancing variables (Section 9.1.3).

```

pi <- n_h/N_h
stratalabels <- levels(gridAmazonia$Biome)
lut <- data.frame(Biome=stratalabels, pi=as.numeric(pi))
gridAmazonia <- merge(x=gridAmazonia, y=lut)
Xbal <- model.matrix(~Biome-1, data=gridAmazonia)
Xbal <- cbind(Xbal, gridAmazonia$lnSWIR2)
Xspread <- cbind(gridAmazonia$x1, gridAmazonia$x2)
set.seed(314)
units <- lcube(Xbal=Xbal, Xspread=Xspread, prob=gridAmazonia$pi)
mysample <- gridAmazonia[units,]

```

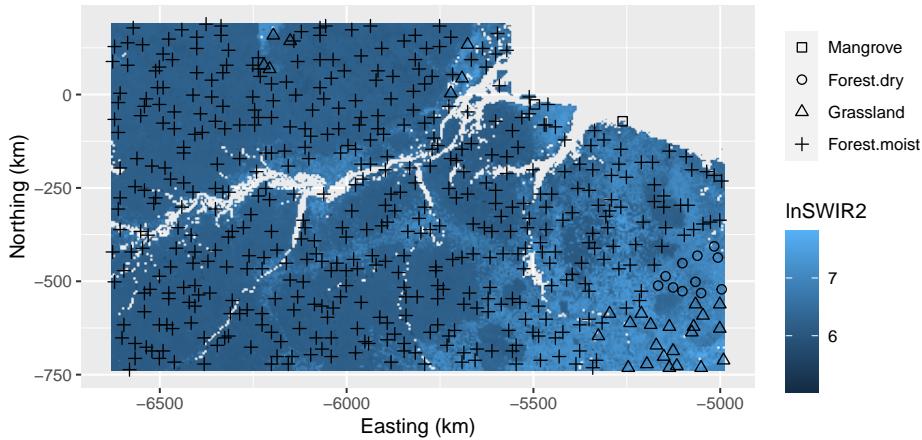


Figure 16.3: Balanced stratified sample with geographical spreading, balanced on lnSWIR2 and stratum sample sizes (biomes used as strata), Eastern Amazonia (Brazil).

I think these are suitable samples, both for mapping AGB across the entire study area, for instance by kriging with an external drift (Section 22.2), and for estimating the mean AGB of the four biomes. For biome Forest.moist this population mean can be estimated from the data of this biome only, using the π estimator, as the sample size of this biome is very large (Section 9.3). For the other three biomes we may prefer model-assisted estimation for small domains as described in Section 14.2.

In this example I used one quantitative covariate, lnSWIR2, for balancing the sample. If we have a legacy sample that can be used to fit a linear or nonlinear model, for instance a random forest using multiple covariates and factors as predictors (Chapter 10), then this model can be used to predict the study variable for all populations units, so that we can use this variable with predictions of the study variable as a balancing variable, see Section 10.3.

16.3 Broad overview of sampling designs for mapping

The non-probability sampling designs for mapping described in the following chapters can be grouped into three categories:

1. Geometric designs (Chapters 17, 18 and 19)
2. Adapted experimental designs (Chapters 20 and 21)
3. Model-based designs (Chapters 23, 24)

Square and triangular grids are examples of geometric sampling designs; the sampling units show a regular, geometric spatial pattern. In other geometric sampling designs the spatial pattern is not perfectly regular. Yet these are classified as geometric sampling designs when the samples are obtained by minimising some geometric criterion, i.e. a criterion defined in terms of distances between the sampling units and the nodes of a fine prediction grid discretising the study area (Chapters 18 and 19).

In model-based sampling designs the samples are obtained by minimising a criterion that is defined in terms of variances of prediction errors. An example is the mean kriging variance criterion, i.e. the average of the kriging variances over all nodes of the prediction grid. Model-based sampling therefore requires prior knowledge of the model of spatial variation. Such a model must be specified and justified. Once this model is given the sample can be optimised. In Chapter 23 I show how a spatial model can be used to optimise the spacing of a square grid given a requirement on the accuracy of the map. The grid spacing determines the number of sampling units, so this optimisation boils down to determining the required sample size. In Chapter 24 I show how a sample of a given size can be further optimised through optimisation of the spatial coordinates of the sampling units.

In Chapter 1 I introduced the design-based and model-based approaches for sampling and statistical inference. Note that a model-based approach does not necessarily imply model-based sampling. The adjective model-based refers to the model-based inference, not to the selection of the units. In a model-based approach sampling units can be, but need not be, selected by model-based sampling. If they are, then both in selecting the units and in mapping a statistical model is used. In most cases the two models differ: once the sample

data are collected, these are used to update the postulated model used for sampling design. This updated model is then used for mapping.

Besides geometric and model-based sampling designs for spatial survey a third category can be distinguished: sampling designs that are adaptations of experimental designs. An adaptation is necessary because in contrast to experiments, in observational studies one is not free to choose combinations of levels of different factors. For instance, when two covariates are strongly positively correlated, it may happen that there are no units with a relatively large value for one covariate and a relatively small value for the other covariate.

In a full factorial design all combinations of factor levels are observed. For instance, suppose we have only two covariates, e.g., application rates for N and P in an agricultural experiment, and four levels for each covariate. To account for possible non-linear effects, a good option is to have multiple plots for all 4×4 combinations. This is referred to as a full factorial design. With k factors and l levels per factor the total number of observations is l^k . With numerous factors and/or numerous levels per factor this becomes unfeasible in practice. Alternative designs have been developed that need fewer observations but still provide detailed information about how the study variable responds to changes in the factor levels. Examples are Latin hypercube samples and response surface designs. The survey sampling analogues of these experimental designs are described in Chapters 20 and 21, respectively.

Chapter 17

Sampling on a regular grid

Sampling on a regular grid is an attractive option for mapping because of its simplicity. The data collected on the grid-points are not used for design-based estimation of the population mean or total, and for that reason the grid need not be placed randomly on the study area as in systematic random sampling (Chapter 5). The grid can be located such that the grid nodes optimally cover the study area, in the sense of the average distance of the nodes of a fine discretisation grid to the nearest node of the sampling grid. Commonly used grid configuration are square, triangular and hexagonal. If the grid data are used in kriging (Chapter 22), the optimal configuration depends, among others, on the semivariogram model. If the study variable shows moderate to strong spatial autocorrelation, triangular grids give the best result in comparison to square grids.

Besides the shape of the grid cells, we must decide on the grid spacing. The grid spacing determines the number of sampling units in the study area, i.e. the sample size. There are two options to decide on this spacing, either starting from the available budget or from a requirement on the quality of the map. The latter is explained in Chapter 23, as this requires a model of the spatial variation, and as a consequence this is an example of model-based sampling. Starting from the available budget and an estimate of the costs per point, we first compute the affordable sample size. Then we may derive from this number the grid spacing. For square grids, the grid spacing in meters is calculated as $\sqrt{A/n}$, where A is the area in m^2 , and n is the number of sampling units (sample size).

Grids can be selected with function `spsample` of package `sp` (?). The argument `offset` is used to select non-randomly a grid. A sample size can be specified, using argument `n`; alternatively a grid spacing can be specified using argument `cellsize`.

```
library(sp)
load("data/Voorst.RData")
gridded(grdVoorst) <- ~s1+s2
mysample <- spsample(
  x=grdVoorst, type="regular", cellsize=c(200,200),
  offset=c(0.5,0.5)) %>% as(., "data.frame")
```

Figure 17.1 shows the selected square grid.

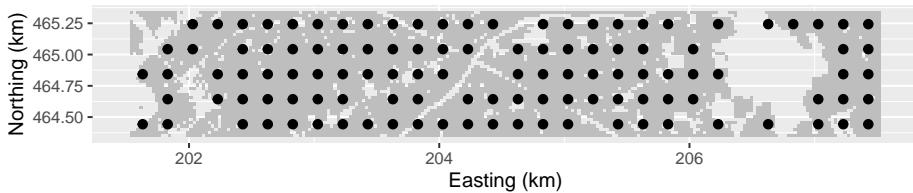


Figure 17.1: Non-random square grid sample from Voorst.

The number of grid points in this example equals 115. Nodes of the square grid in parts of the area not belonging to the population of interest, such as built-up areas and roads, are discarded by `spsample`, because these areas are not included in the sampling frame file `grdVoorst`. As a consequence, there are some undersampled areas, for instance in the middle of the study area where two roads cross. If we use the square grid in spatial interpolation, e.g., by ordinary kriging, we are more uncertain about the predictions in these undersampled areas than in areas where the grid is complete. In the next chapter I show how this local undersampling can be avoided.

Exercises

1. Write an **R** script to select a square grid of size 100 from the three woredas in Ethiopia. The data.frame `grdEthiopia` is stored in `data/CovariatesThreeWoredasEthiopia.RData`. Use argument

`offset=c(0.5,0.5)` so that the starting point of the grid is not selected randomly.

- Compute the number of selected grid points. How comes that it is not exactly equal to 100?
- Select a square grid with a spacing of 10.2 distance units (km), and compute the sample size.
- Write a for-loop to select 200 times a square grid of on average 100 points with random starting point. Set a seed so that results can be reproduced. Determine for each randomly selected grid the number of selected grid points, and save this in a numeric. Compute summary statistics of the sample size and plot a histogram.
- Select a square grid of exactly 100 points.

Chapter 18

Spatial coverage sampling

Local undersampling with regular grids can be avoided by relaxing the constraint that the sampling units are restricted to the nodes of a regular grid. This is what is done in *spatial coverage sampling* or, in case of a sample that is added to an existing sample, in *spatial infill sampling*. Spatial coverage and infill samples cover the area or fill in the empty space as uniformly as possible. The sampling units are obtained by minimising a criterion that is defined in terms of the geographic distances between the nodes of a fine discretisation grid and the sampling units. ? proposed to minimise the mean of the squared distances of the grid nodes to their nearest sampling unit (mean squared shortest distance, MSSD):

$$MSSD = \frac{1}{N} \sum_{k=1}^N \min_j (D_{kj}^2) , \quad (18.1)$$

where N is the total number of nodes of the discretisation grid, and D_{kj} is the distance between the k th grid node and the j th sampling point. This distance measure can be minimised by the k -means algorithm, which is a numerical, iterative procedure. Figure 18.1 illustrates the selection of a spatial coverage sample of four points from a square. In this simple example the optimal sample is known, being the centers of four subsquares of equal size. A simple random sample of four points serves as an initial solution. Each raster cell is then assigned to the closest sampling point. This is the initial clustering. In the

next iteration the centers of these initial clusters are computed. Next, the raster cells are re-assigned to the closest new centers. This continues until there is no change anymore. In this case only nine iterations are needed, where an iteration consists of computing the clusters by assigning the raster cells to the nearest center (sampling unit), followed by computing the centers of these clusters. Figure 18.1 shows the first, second and ninth iteration. Note that the compact geostrata have equal size, but this is an artefact due to the shape of the area and the number of clusters. Geostrata of equal size are not guaranteed with the k -means algorithm described here.

The same algorithm was used in Chapter 4 to construct compact geographical strata (shortly referred to as geostrata) for stratified random sampling. In stratified random sampling, one or more sampling units are selected randomly from each geostratum. However, for mapping purposes probability sampling is not required, so the random selection of a unit within each stratum is not needed. With random selection the coverage can be optimal. To select a sample with optimal spatial coverage for each compact geostratum the means of the spatial coordinates of the nodes of the discretisation grid of that geostratum are calculated. These centers are used as sampling points. This improves the spatial coverage compared to stratified *random* sampling.

In probability sampling we may want to have strata of equal area (clusters with equal numbers of units), so that the sampling design becomes self-weighting. For mapping this constraint is not recommended as it may lead to samples with suboptimal spatial coverage. Note that in Figure 18.1 the geostrata (clusters) are of equal area, but selecting four points from a square is a very special case. In other cases the geostrata might well have unequal area. If geostrata of equal area are required for stratified simple random sampling, a different k -means algorithm must be used, using swops.

Spatial coverage samples can be computed with package **spcosa** (?), using functions **stratify** and **spsample**, see code chunk below. Note that function **spsample** without optional argument **n** selects non-randomly one point, the center, in each cluster. Figure 18.2 shows a spatial coverage sample of the same size as the regular grid in study area Voorst (Figure 17.1). Note that the undersampled area in the center of the study area is now covered by a sampling point.

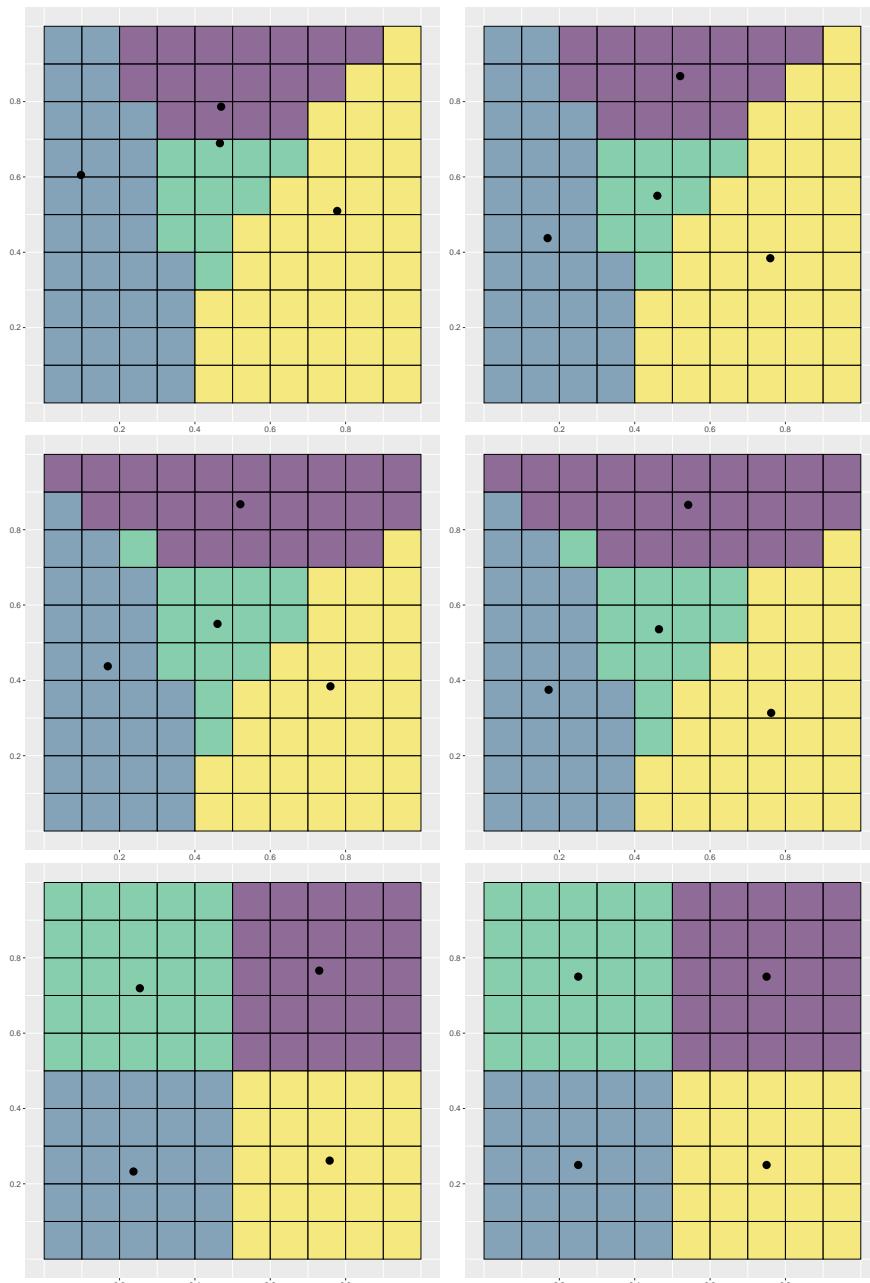


Figure 18.1: First, second and ninth iteration of k-means algorithm to select a spatial coverage sample of four points from a square. Iterations are rowwise from top to bottom. In the left column of subfigures clusters are computed by assigning the raster cells to the nearest center. In the right column of subfigures centers of the clusters are computed.

```

load("data/Voorst.RData")
library(spcosa)
n <- 115
set.seed(314)
gridded(grdVoorst) <- ~s1+s2
mystrata<-stratify(grdVoorst, nStrata=n, equalArea=FALSE, nTry=10)
mysample <- spsample(mystrata)
plot(mystrata,mysample)

```

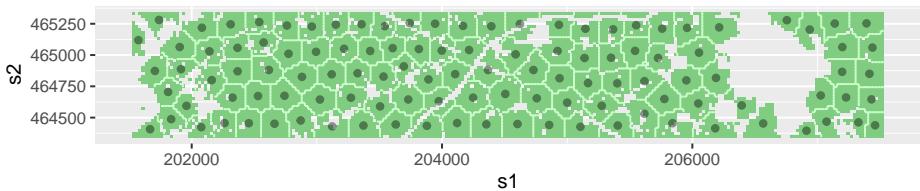


Figure 18.2: Spatial coverage sample from Voorst.

```
mysample <- as(mysample, "data.frame")
```

If the clusters need not be of equal size, we may also use function `kmeans` of the `stats` package, using the spatial coordinates as clustering variables. This requires less computing time, especially with large data sets.

```

grdVoorst <- as(grdVoorst, "data.frame")
mystrata_kmeans <- kmeans(
  grdVoorst[,c("s1","s2")], centers=n, iter.max=10000, nstart=10)
mysample_kmeans <- mystrata_kmeans$centers %>% data.frame()

```

When function `kmeans` is used to compute the spatial coverage sample, there is no guarantee that the computed centers of the clusters used as sampling points are inside the study area. In Figure 18.3 there are eight such centers.

This problem can easily be solved by selecting points inside the study area closest to the centers that are outside the study area. Function `rdist` of package `fields` is used to compute a matrix with distances between the centers outside

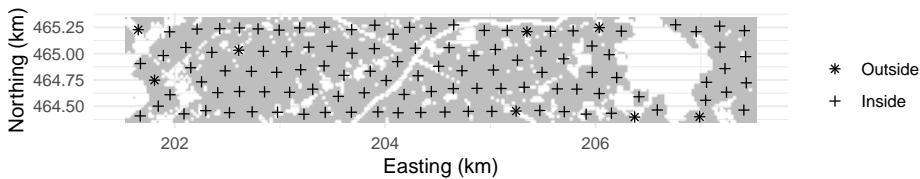


Figure 18.3: Centers of spatial clusters computed with kmeans.

the study area and the nodes of the discretisation grid. Then function `apply` is used with argument `FUN=which.min` to compute the discretisation nodes closest to the centers outside the study area. This procedure is implemented in function `spsample` of package `spcosa` without argument `n`.

```
library(fields)
units_out <- which(inside==FALSE)
D <- fields::rdist(x1=mysample_kmeans[units_out,],
                     x2=grdVoorst[,c("s1","s2")])
units_close <- apply(D, MARGIN=1, FUN=which.min)
mysample_kmeans[units_out,] <- grdVoorst[units_close,c("s1","s2")]
```

Exercises

1. In forestry and vegetation surveys square or circular plots are often used as sampling units, for instance squares of $2 \text{ m} \times 2 \text{ m}$, or circles with a diameter of 2 m. To study the relation between the vegetation and the soil, soil samples must be collected from the vegetation plots. Suppose we want to collect four soil samples from a square plot. Where would you locate the four sampling points, so that they optimally cover the plot?
2. Suppose we are also interested in the accuracy of the estimated plot means of the soil properties, not just the means. In that case the soil samples should not be bulked into a composite sample, but analyzed separately. How would you select the sampling points in this case?
3. For circular vegetation plots it is less clear where the sampling points with smallest MSSD are. Write an **R** script to compute a spatial coverage sample of five points from a circular plot discretised by the nodes of a fine

square grid. Use argument `equalArea=FALSE`. Check the size (number of grid nodes) of the strata. Repeat this for six sampling points.

4. Consider the case of six strata. The strata are not of equal size. If the soil samples are bulked into a composite sample, the measurement on this single sample is a biased estimator of the plot mean. How can this bias be avoided?

18.1 Spatial infill sampling

If georeferenced data are available that can be used for mapping the study variable, but we need more data for mapping, it is attractive to account for these existing sampling units when selecting the additional sampling units. The aim now is to fill in the empty spaces, i.e. the parts of the study area not covered by the existing sampling units. This is referred to as *spatial infill sampling*. Existing sampling units can easily be accommodated in the k -means algorithm, by using them as fixed cluster centers.

Figure 18.4 shows a spatial infill sample for three woredas (districts) in Ethiopia. A large set of legacy data on soil organic matter (SOM) in wt% is available, but these data come from strongly spatially clustered units along roads. This is a nice example of a convenience sample. The legacy data are not ideal for mapping SOM throughout the three woredas. Clearly, it is desirable to collect additional data in the off-road parts of the woredas, with the exception of the northeastern part where we have already quite a few data not near the main roads. The legacy data are specified as a *SpatialPoints* object to the `priorPoints` optional argument of `spcosa`. This argument fixes these points as cluster centers. A spatial infill sample of 100 points is selected, taking into account these fixed points.

```
load("data/CovariatesThreeWoredasEthiopia.RData")
load("data/ThreeWoredasEthiopia.RData")
gridded(grdEthiopia) <- ~s1+s2
n <- 100
ntot <- n+length(priordataEthiopia)
priordata <- as(priordataEthiopia, "SpatialPoints")
proj4string(priordata) <- NA_character_
set.seed(314)
```

```
mystrata <- stratify(
  grdEthiopia, nStrata=ntot, priorPoints=priordata, nTry=10)
mysample <- spsample(mystrata)
plot(mystrata, mysample)
```

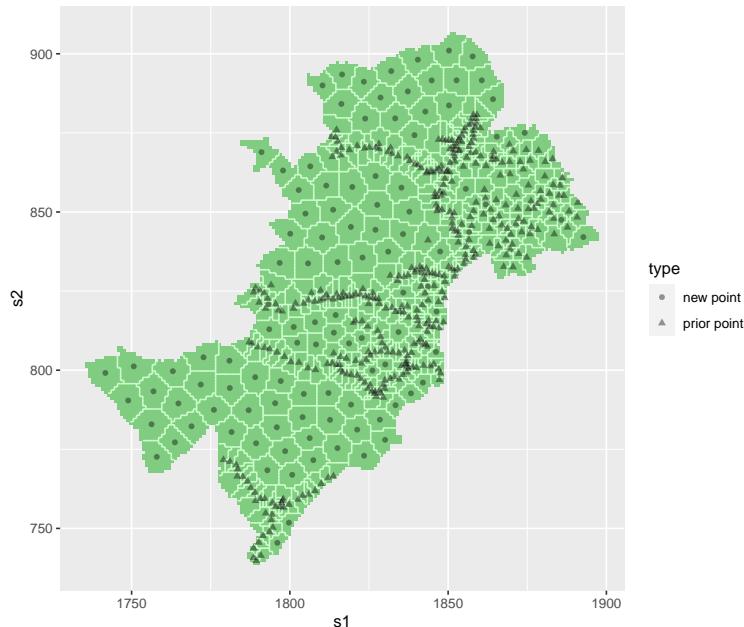


Figure 18.4: Spatial infill sample in three woredas of Ethiopia.

In the output object of `spsample` both the prior and the new sampling points are included. The new points can be obtained as follows:

```
units <- which(mysample@isPriorPoint==FALSE)
mysample <- as(mysample, "data.frame")
mysample_new <- mysample[units,]
```

Exercises

5. Write an **R** script to select a spatial infill sample of size 100 from study area Xuancheng in China. Existing data on soil organic matter (SOM, g/kg) in the topsoil at 60 sampling points are in the data.frame `data/Xuancheng_iPSMsample.csv`. To map SOM we want to measure SOM at 100 more sampling points. Use the raster file `data/Xuancheng_elevation.tif` as a discretisation of the study area. This file can be read with function `raster` of package `raster`.
 - In the example of this section there are far too many raster cells that could be used as new points in spatial infill sampling. That many cells are not needed. Subsample the raster file by selecting a square grid with a spacing of 900 m × 900 m. First change the class of the raster file to a `SpatialPixelsDataFrame`. Then use function `spsample` with argument `type=regular`.
 - Select a spatial infill sample using functions `stratify` and `sample` of package `spcosa`.

Chapter 19

Covariate space coverage sampling

Regular grid sampling and spatial coverage sampling are pure spatial sampling designs. Covariates possibly related to the study variable are not accounted for in selecting sampling units. This can be suboptimal when the study variable is related to covariates of which maps are available, think for instance of remote sensing imagery or digital elevation models related to soil properties. Maps of these covariate can be used in mapping the study variable by, for instance, a multiple linear regression model or a random forest. This chapter describes a simple, straightforward method for selecting sampling units on the basis of the covariate values of the grid cells. This is a pure feature (attribute) space design.

The simplest option for covariate space coverage sampling (CSC sampling) is to cluster the grid cells by the k -means clustering algorithm in multivariate covariate space. Similar to spatial coverage sampling (Chapter 18) the mean squared shortest distance (MSSD) is minimised, but now the distance is not measured in geographical space but in a p -dimensional space spanned by the p covariates (think of it as a multidimensional scatter plot with the covariates along the axes). The covariates are centered and scaled, so that their means become 0 and standard deviations become 1. This is needed because, contrary to the spatial coordinates used as clustering variables in spatial coverage sampling, the dimensions of the covariates used as clustering variables generally differ,

and the ranges of the covariates in the population can differ greatly. In the clustering of the grid cells the mean squared shortest *scaled* distance (MSSSD)¹ is minimised.

In the next code chunk a CSC sample of size 20 is selected from Eastern Amazonia. To speed up the computations a subgrid with a spacing of 5000 m is selected, using function `spsample` of package `sp`. All five quantitative covariates, SWIR2, Terra_PP, Prec_dm, Elevation and Clay, are used as covariates. To select twenty points, twenty clusters are constructed using function `kmeans` of the `stats` package (?). The number of clusters is specified with argument `centers`. Note that the number of clusters is not based, as usual in cluster analysis, on the expected number of subregions with a high density of points in the multivariate distribution, but rather on the number of observation points to be sampled. The k -means clustering algorithm is a deterministic algorithm, i.e. the final optimised clustering is fully determined by the initial clustering. This final clustering can be suboptimal, i.e. the minimised MSSSD value is somewhat larger than the global minimum. Therefore the clustering should be repeated many times, every time starting with a different random initial clustering. The number of repeats is specified with argument `nstart`.

```
load("data/Amazonia_5km.RData")
covs <- names(gridAmazonia)[c(4,5,6,7,8)]
n <- 20
set.seed(314)
myclusters <- kmeans(
  scale(gridAmazonia[,covs]), centers=n,
  iter.max=10000, nstart=100)
gridAmazonia$cluster <- myclusters$cluster
```

Grid cells with the shortest scaled Euclidean distance in covariate-space to the centers of the clusters are selected as the sampling points. To this end first a matrix with the distances of all the grid cells to the cluster centers is computed with function `rdist` of package `fields` (?). The grid cells closest to the centers are computed with function `apply`, using argument `FUN=which.min`.

¹The name ‘scaled distance’ can be confusing. Not the geographic distances are scaled, rather, the distances are computed in a space spanned by the scaled covariates

```
library(fields)
D <- rdist(x1=myclusters$centers, x2=scale(gridAmazonia[,covs]))
units <- apply(D, MARGIN=1, FUN=which.min)
myCSCsample <- gridAmazonia[units,]
```

Figure 19.1 shows the clustering of the grid cells and the grid cells closest in covariate space to the centers, used as the selected sample. In Figure 19.2 the selected sample is plotted in scatter diagrams of some pairs of covariates. In the scatter diagrams some sampling points are clearly clustered. However, this is misleading, as actually we must look in five-dimensional space to see whether the points are clustered. Two points with a large separation distance in a five-dimensional space can look quite close when these two points are projected on a two-dimensional plane, see also Exercise 2 of this chapter.

The next code chunk shows how the MSSSD of the selected sample can be computed.

```
popmeans <- apply(gridAmazonia[,covs], MARGIN=2, FUN=mean)
popsds <- apply(gridAmazonia[,covs], MARGIN=2, FUN=sd)
D <- rdist(
  x1=scale(myCSCsample[,covs], center=popmeans, scale=popsds),
  x2=scale(gridAmazonia[,covs]))
dmin <- apply(D, MARGIN=2, min)
MSSSD <- mean(dmin^2)
```

Note that to center and scale the covariate values in the CSC sample, the population means and population standard deviations are used, as specified with arguments `center` and `scale` of function `scale`. If these means and standard deviations are unspecified, the *sample* means and *sample* standard deviations are used, resulting in an incorrect value of the minimised MSSSD value. The MSSSD of the selected sample equals 1.004.

Instead of function `kmeans` we may use function `kmeanspp` of package **LICORS** (?). This function is an implementation of the *k-means++* algorithm (?). This algorithm consists of two parts, namely the selection of an optimised initial sample, followed by the standard *k*-means. The algorithm is as follows:

1. Select one unit (raster cell) at random.

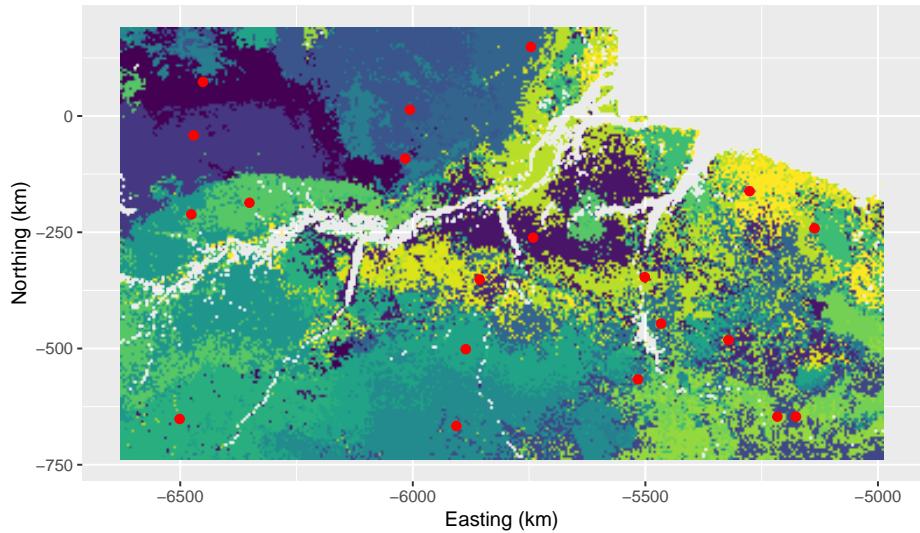


Figure 19.1: Covariate space coverage sample of twenty points obtained with k -means, from Eastern Amazonia using five covariates in clustering. The map shows the spatial distribution of the twenty clusters.

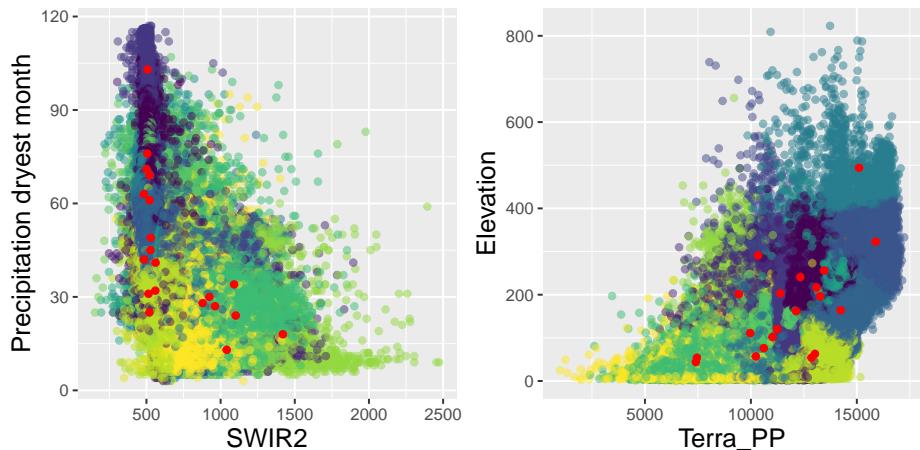


Figure 19.2: Covariate space coverage sample, obtained with k -means, plotted in scatter diagrams of pairs of covariates, coloured by cluster.

-
2. For each unit j , compute d_{ij} , the distance in standardised covariate-space between j and the nearest unit i that has already been selected.
 3. Choose one new raster cell at random as a new sampling unit with probabilities proportional to d_{ij}^2 , and add the selected raster cell to the set of selected cells.
 4. Repeat Steps 2 and 3 until n centers have been selected.
 5. Now that the initial centers have been selected, proceed using standard k -means.

```
library(LICORS)
myclusters <- kmeanspp(
  scale(gridAmazonia[,covs]), k=n, iter.max=10000, nstart=30)
```

Due to the improved initial centers, the risk of ending in a local minimum is reduced. The k -means++ algorithm is of most interest for small sample sizes. For large sample sizes the extra time needed for computing the initial centers can become substantial and may not outweigh the larger number of starts that can be afforded with the usual k -means algorithm for the same computing time.

19.1 Covariate space infill sampling

If we have legacy data that can be used to fit a model for mapping, it is more efficient to select an infill sample, similar to spatial infill sampling explained in Section 18.1. The only difference with spatial infill sampling is that the legacy data are now plotted in the space spanned by the covariates, and the empty regions we would like to fill in are now the undersampled regions in this covariate space. The legacy sample units serve as fixed cluster centers, they cannot move through the covariate space during the optimisation of the infill sample. In the next code chunk a function is defined for covariate space infill sampling.

```
CSIS <- function (fixed,nsup,nstarts,mygrd) {
  nfix <- nrow(fixed)
  p <- ncol(mygrd)
```

```

units <- fixed$units
mygrd_minfx <- mygrd[-units,]
MSSSD_cur <- NA
for (s in 1:nstarts) {
  units <- sample.int(nrow(mygrd_minfx), nsup)
  centers_sup <- mygrd_minfx[units,]
  centers <- rbind(fixed[, names(mygrd)], centers_sup)
  repeat {
    D <- rdist(x1=centers, x2=mygrd)
    clusters <- apply(X=D, MARGIN=2, FUN=which.min) %>%
      as.factor(.)
    centers_cur <- centers
    for (i in 1:p) {
      centers[, i] <- tapply(mygrd[, i], INDEX=clusters, FUN=mean)
    }
    #restore fixed centers
    centers[1:nfix,] <- centers_cur[1:nfix,]
    #check convergence
    sumd <- diag(rdist(x1=centers, x2=centers_cur)) %>% sum(.)
    if (sumd < 1E-12) {
      D <- rdist(x1=centers, x2=mygrd)
      Dmin <- apply(X=D, MARGIN=2, FUN=min)
      MSSSD <- mean(Dmin^2)
      if (s==1 | MSSSD < MSSSD_cur) {
        centers_best <- centers
        clusters_best <- clusters
        MSSSD_cur <- MSSSD
      }
      break
    }
  }
}
list(centers=centers_best, clusters=clusters_best)
}

```

The function is used to select an infill sample of fifteen units from Eastern Amazonia. A legacy sample of five units is randomly selected.

```
set.seed(314)
units <- sample.int(nrow(gridAmazonia), 5)
fixed <- data.frame(units, scale(gridAmazonia[,covs])[units,])
mygrd <- data.frame(scale(gridAmazonia[,covs]))
res <- CSIS(fixed=fixed, nsup=15, nstarts=10, mygrd=mygrd)
```

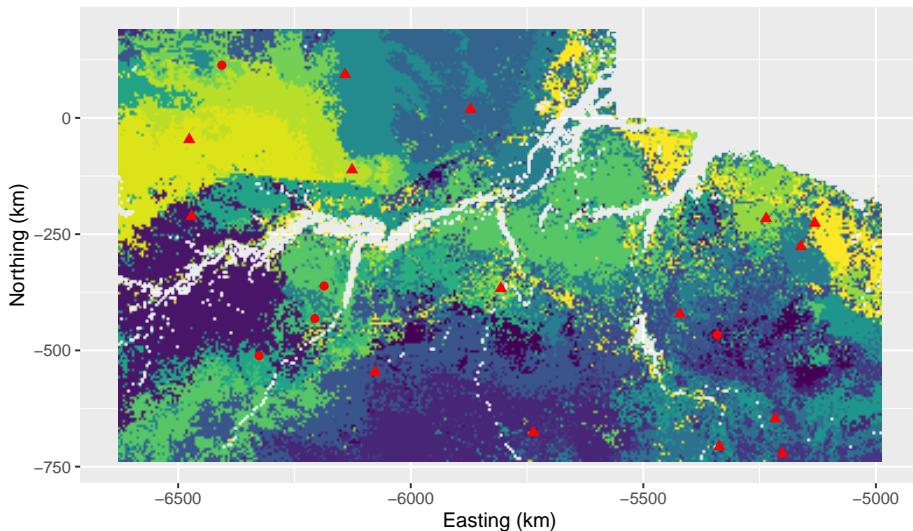


Figure 19.3: Covariate space infill sample, obtained with modified k -means from Eastern Amazonia in a map of SWIR2. The dots represent the fixed centers (legacy sample, five units), the triangles the infill sample (fifteen units).

19.2 Performance of covariate space coverage sampling in random forest prediction

Covariate space coverage sampling can be a good candidate sampling design if we have multiple maps of covariates and we do not want to rely on a linear relation between the study variable and the covariates. In this situation we may consider mapping with machine learning algorithms, such as neural networks and random forests.

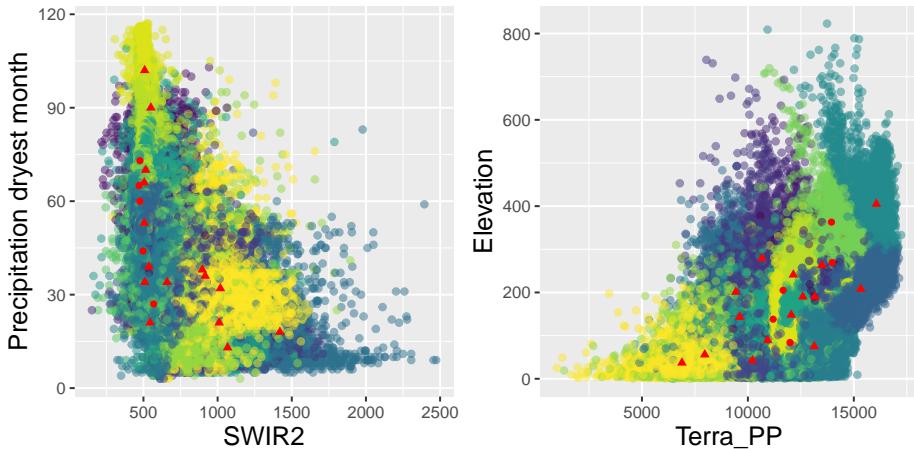


Figure 19.4: Covariate space infill sample, obtained with modified k -means, plotted in scatter diagrams of pairs of covariates. The dots represent the fixed centers (legacy sample), the triangles the infill sample.

I used the Eastern Amazonia data set to evaluate CSC sampling for mapping the aboveground biomass (AGB). The five covariates are used as predictors in random forest modelling. The calibrated models are used to predict AGB at the units of a validation sample of size 25000, selected by simple random sampling without replacement from the $1 \text{ km} \times 1 \text{ km}$ grid, excluding the cells of the $10 \text{ km} \times 10 \text{ km}$ grid from which the calibration samples are selected. The predicted AGB values at the validation units are compared with the true AGB values, and the prediction errors computed. The sample mean of the (squared) prediction error is a design-unbiased estimator of the population mean (squared) error, i.e. the mean of the (squared) errors at all population units (excluding the units of the $10 \text{ km} \times 10 \text{ km}$ grid), see Chapter 26. Besides, the Nash-Sutcliffe model efficiency coefficient (MEC) is estimated.

Three sample sizes are used, $n = 25, 50, 100$. Of each sample size 500 CSC samples are selected using the k -means algorithm, leading to 1500 CSC samples in total. The number of starts are 500, 350 and 200 for $n = 25, 50, 100$, respectively. With these number of starts the computing time was about equal to conditioned Latin hypercube sampling, see next chapter. Each sample is used to calibrate a random forest model. Simple random sampling (SI) is used as a reference strategy. The results are described in detail in the next chapter. In

short: for $n = 25$ and 50 CSC performs on average somewhat better than SI, for $n = 100$ they perform about equal. Most striking is the smaller spread in the map quality indices with CSC as compared to SI.

In Figure 19.5 the RMSE is plotted against the minimised MSSSD, both for the 3×500 CSC samples, and for the 3×500 simple random samples. The vertical strings of points in the scatter plots are the points for the CSC samples. For all three sample sizes the minimised MSSSD values of the CSC samples is substantially smaller than those of the SI samples. For $n = 25$ and 50 the slope of the fitted line is positive: the smaller the MSSSD the smaller the expected RMSE. For $n = 100$ the estimated slope parameter is slightly negative. So for this sample size we do not profit from spreading the sampling units in multivariate covariate space through k -means clustering.

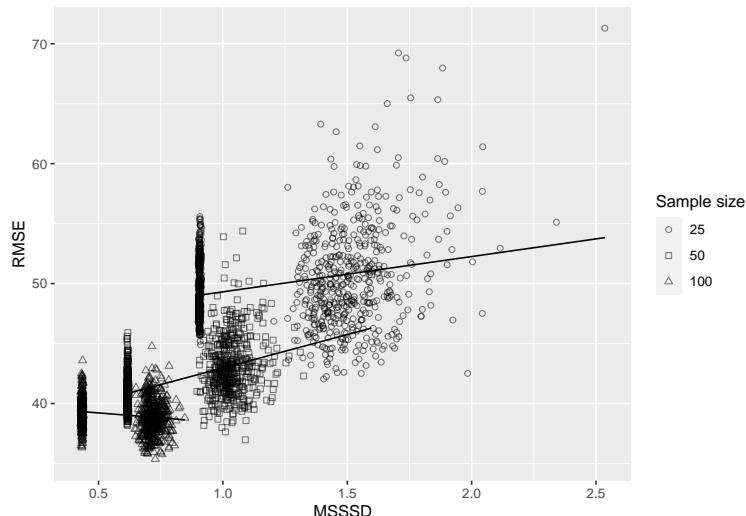


Figure 19.5: Relation between minimisation criterion MSSSD and the root mean squared error (RMSE) of random forest predictions of aboveground biomass in Eastern Amazonia for three sample sizes. Solid lines are the fitted simple linear regression models for RMSE with MSSSD as a predictor.

Exercises

1. Write an **R** script to select a covariate space coverage (CSC) sample of size 20, using the covariates cti and ndvi in k -means clustering of the grid cells. Plot the clusters and the sample in a scatter diagram of cti against ndvi.
2. Repeat this, but now with covariates cti, ndvi *and* elevation_m as covariates. Plot the clusters and the sample again in a scatter diagram of cti against ndvi. Explain that in this case a sampling location (cluster center) can be quite close to another sampling location, and that a cluster in a 2D-scatter diagram may also contain units of a different cluster.

Chapter 20

Conditioned Latin hypercube sampling

This chapter and the next on response surface sampling are about experimental designs that have been adapted for spatial surveys. Adaptation is necessary because, in contrast to experiments, in observational studies one is not free to choose any possible combination of levels of different factors. When two covariates are strongly correlated it may happen that there are no units with a relatively large value for one covariate and a relatively small value for the other covariate. By contrast, in experimental research it is possible to select combinations of levels of factors so that the factors are independent.

In a full factorial design all combinations of factor levels are observed. With k factors and l levels per factor the total number of observations is l^k . With numerous factors and/or numerous levels per factor observing l^k experimental units becomes unfeasible in practice. Alternative experimental designs have been developed that need fewer observations but still provide detailed information about how the study variable responds to changes in the factor levels. In this chapter I describe and illustrate the survey sampling analogue of Latin hypercube sampling. Response surface sampling will follow in the next chapter.

Latin hypercube sampling (LHS) is used in designing industrial process, agricultural and computer experiments, with numerous covariates and/or factors of which we want to study the effect on the output (?). With numerous covariates

and/or levels per covariate a full factorial design becomes unfeasible. A much cheaper alternative then is an experiment with, for all covariates, exactly one observation per level. So in the agricultural experiment described in Chapter 16 with two factors, being application rates of N and of P, and four levels for each factor, this would entail four observations only, distributed in a square in such way that we have in all rows and in all columns one observation. This is referred to as a Latin square. The generalisation of a Latin square to a higher number of dimensions is a Latin hypercube (LH).

? adapted LH sampling for observational studies; this adaptation is referred to as conditioned Latin hypercube sampling (cLH sampling). For each covariate a series of intervals (marginal strata) is defined. The breaks of the marginal strata are chosen such that the numbers of grid cells in these marginal strata are equal. This can be done by using the quantiles corresponding with evenly spaced cumulative probabilities as stratum breaks. For instance, for five marginal strata we use the quantiles corresponding with the cumulative probabilities 0.2, 0.4, 0.6 and 0.8.

The minimisation criterion proposed by ? is a weighted sum of three components:

1. O1: the sum over all marginal strata of the absolute deviations of the marginal stratum sample size from the targeted sample size (equal to 1).
2. O2: the sum over all classes of categorical covariates of the absolute deviations of the sample proportion of a given class from the population proportion of that class.
3. O3: the sum over all entries of the correlation matrix of the absolute deviation of the correlation in the sample from the correlation in the population.

With cLH sampling the marginal distributions of the covariates in the sample are close to these distributions in the population. This can be advantageous for mapping methods that do not rely on linear relations, for instance in machine learning techniques like classification and regression trees (CART), and random forests. In addition, criterion O3 ensures that the correlations between predictors are respected in the sample set.

cLH samples can be selected with package **clhs** (?). With this package the criterion is minimised by simulated annealing, see Section 24.1 for an explanation

of this optimisation method.

cLH sampling is illustrated with the five covariates of Eastern Amazonia that were used before in covariate space coverage sampling (Chapter 19).

```
library(clhs)
load("data/Amazonia_5km.RData")
covs <- names(gridAmazonia)[c(4,5,6,7,8)]
set.seed(314)
res <- clhs(
  gridAmazonia[,covs], size=20, tdecrease=0.95,
  iter=10000, progress=FALSE, simple=FALSE)
index <- res$index_samples
myCLHsample <- gridAmazonia[index, ]
```

Figure 20.1 shows the selected sample in a map of SWIR2.

Figure 20.2 shows the sample plotted in a scatter diagram of Precipitation Dryest Month against SWIR2. Each black dot represents one grid cell in the population. The horizontal and vertical lines in this scatter diagram are at the boundaries of the marginal strata of SWIR2 and Precipitation Dryest Month, respectively. The number of black dots within each rectangle formed by the horizontal and vertical bars are equal (i.e. marginal strata have equal size). Thus the intervals are the narrowest (rectangles have the least area) where the density of black dots in the plot is highest. Ideally in each column and each row there is exactly one sampling point but of course this is not possible if the number of points is smaller than the number of boxes in hyperspace. For example, we can see in Figure 20.2 that in the first marginal stratum of SWIR2 there are no sampling units, whereas in the 13th marginal two units are selected.

Figure 20.3 shows the sample sizes for all 100 marginal strata.

For all marginal strata with one sampling unit the contribution to component O1 of the criterion is zero. For marginal strata with zero or two sampling units, the contribution is 1, for marginal strata with three sampling units the contribution equals 2.

Figure 20.4 shows the trace of the objective function, i.e. the values of the minimisation criterion during the optimisation.

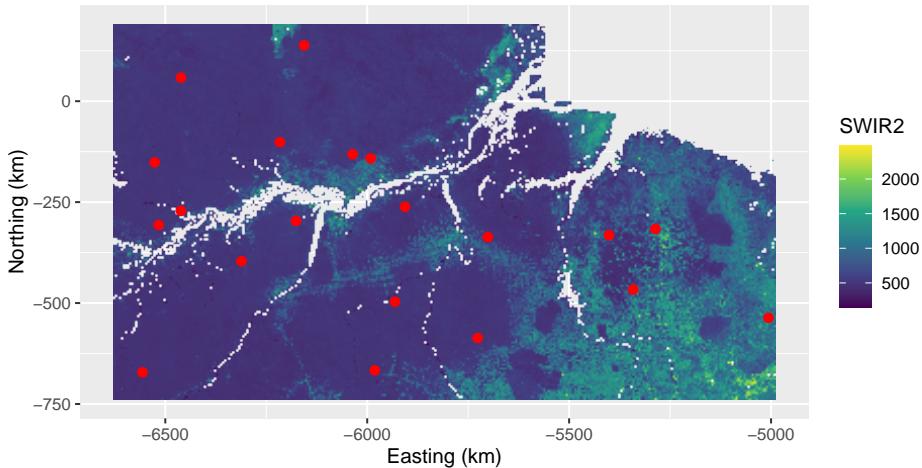


Figure 20.1: Conditioned Latin hypercube sample from Eastern Amazonia in a map of SWIR2.

```
trace <- res$obj
```

cLH samples can also be selected with function `optimCLHS` of package `spsann` (?). With this package the criterion is minimised by a spatial version of simulated annealing. The function `scheduleSPANN` is used to define the annealing schedule. It has some interesting arguments so that we have better control of the annealing compared to the package `clhs`, such as the arguments `initial.acceptance` and `stopping`, see Chapter 24 for details. On the other hand, my experience is that package `clhs` is much quicker than package `spsann`, and can handle larger data sets.

Exercises

1. Load the simulated data of Figure 16.1 (`data/SimulatedSquare.RData`), and select a cLH sample of size 16, using the covariate x and the spatial coordinates as stratification variables. Plot the selected sample in the square with simulated covariate values.
 - What do you think of the geographical spreading of the sampling units (spatial coverage)? Is it optimal?

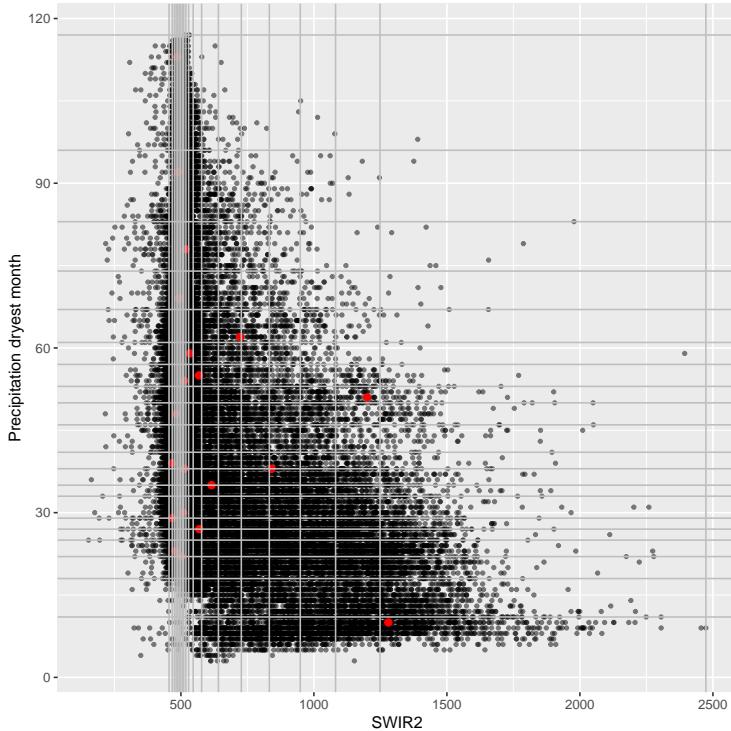


Figure 20.2: Conditioned Latin hypercube sample plotted in a scatter diagram of SWIR2 against precipitation in dryest month. The horizontal and vertical lines are at the boundaries of the marginal strata of the covariates SWIR2 and precipitation dryest month, respectively.

- Compute the number of sampling points in the marginal strata of s_1 , s_2 and the covariate x . First compute the breaks of these marginal strata. Are all marginal strata of s_1 and s_2 sampled? Suppose that all marginal strata of s_1 and s_2 are sampled (contain one sampling point), does this guarantee good spatial coverage?
- Plot the trace of the minimisation criterion, and retrieve the minimised value. Is this minimised value in agreement with the marginal

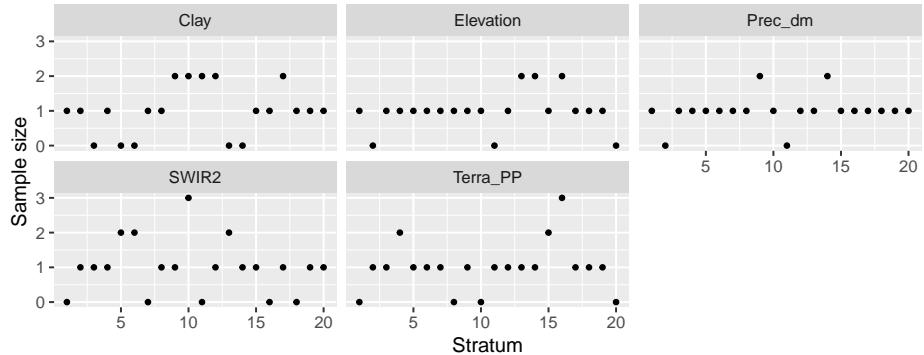


Figure 20.3: Sample sizes of marginal strata for the conditioned Latin hypercube sample of size twenty from Eastern Amazonia.

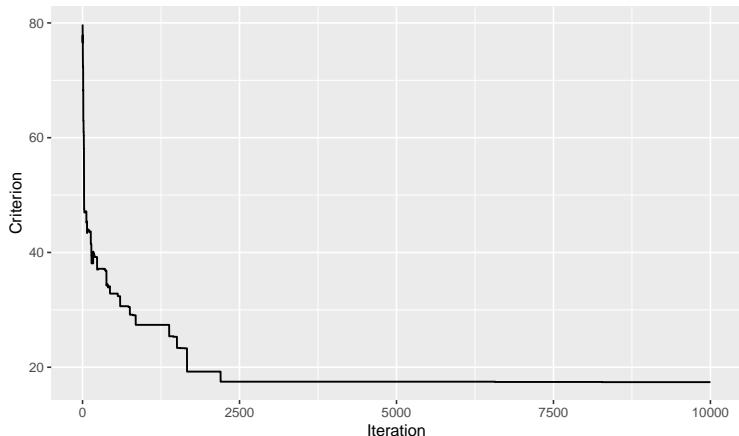


Figure 20.4: Trace of minimisation criterion during optimisation of conditioned Latin hypercube sampling from Eastern Amazonia.

stratum sample sizes?

20.1 Conditioned Latin hypercube infill sampling

Package **clhs** can also be used for selecting a conditioned Latin hypercube sample in addition to existing sampling points, as in spatial infill sampling (Chapter 18). For this the argument **include** can be used. The argument size must then be set to the total sample size, i.e. the number of mandatory points plus the number of additional infill points.

To illustrate conditioned Latin hypercube *infill* sampling (cLHIS), I selected randomly ten points from Eastern Amazonia to serve as existing data (legacy sample). Twenty new points are selected by cLHIS. The ten mandatory points (i.e. already sampled and thus must be in the sample set computed by cLHIS) are at the end of the vector with the index of the selected raster cells.

```
set.seed(314)
units <- sample.int(nrow(gridAmazonia), 10, replace=FALSE)
res <- clhs(
  gridAmazonia[,covs], size=30, include=units,
  tdecrease=0.95, iter=10000,
  progress=FALSE, simple=FALSE)
index <- res$index_samples
myCLHIsample <- gridAmazonia[index, ]
myCLHIsample$free <- as.factor(rep(c(1,0), c(20,10)))
```

Figure 20.5 shows the selected Latin hypercube infill sample in a map of SWIR2, and in Figure 20.6 the sample is plotted in a scatter diagram of SWIR2 against Prec_dm. The marginal strata already covered by the legacy sample are mostly avoided by the additional sample.

20.2 Performance of conditioned Latin hypercube sampling in random forest prediction

The performance of conditioned Latin hypercube sampling is studied in the same experiment as covariate space coverage sampling of the previous chapter.

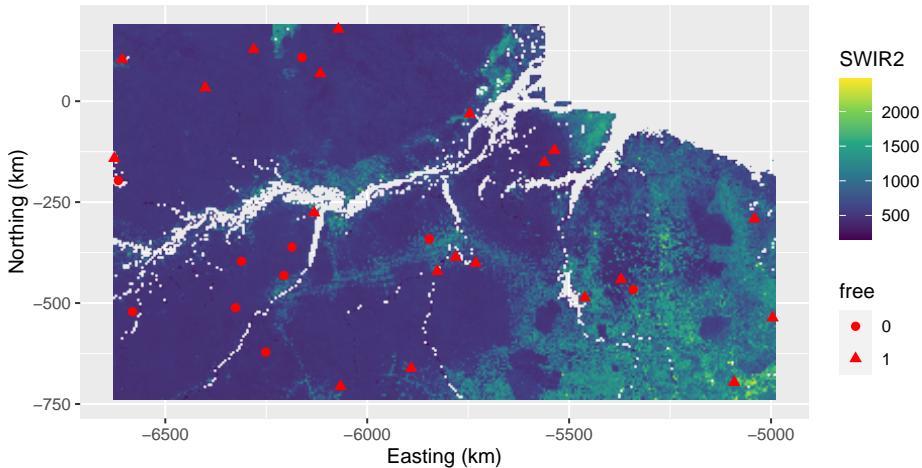


Figure 20.5: Conditioned Latin hypercube infill sample from Eastern Amazonia in a map of SWIR2. Legacy points have free-value 0, infill points have free-value 1.

For this, 500 cLH samples of size 25, and an equal number of samples of size 50 and 100 are selected. Each sample is used to calibrate a random forest model for aboveground biomass (AGB), using five covariates as predictors. The calibrated models are used to predict AGB at the 25,000 validation units, selected by simple random sampling without replacement. The same procedure is used for a simple random sample (SI), as a prediction model that ignores covariates. The prediction errors are used to estimate three map quality indices, the population mean error (ME), population mean squared error (MSE) and the population Nash-Sutcliffe model efficiency coefficient (MEC), see Chapter 26. Figure 20.7 shows the results as boxplots, each based on 500 estimates.

Figure 20.7 shows that for $n = 25$ and 100 cLH sampling performs best, whereas for $n = 50$ CSC sampling performs best. For $n = 25$ and 50 the boxplots of cLH and simple random sampling show quite a few outliers with large values of RMSE, resulting in small values of MEC. For CSC these map quality indices are more stable. The poor performance of SI shows that, if covariates are well-related with the target variable, SI is far from optimal in covering the covariate space, so that the resulting random forest models are quite poor.

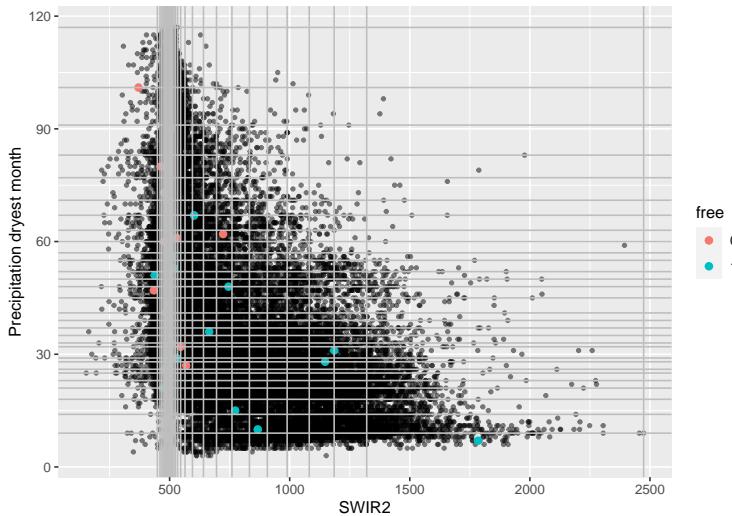


Figure 20.6: Conditioned Latin hypercube infill sample plotted in scatter diagram of SWIR2 against precipitation in dryest month. Legacy points have free-value 0, infill points have free-value 1.

In Figure 20.8 the RMSE is plotted against the minimised $O_1 + O_3$ criterion for the cLH and the simple random samples. For all three sample sizes the slope of the fitted line is positive, so the smaller $O_1 + O_3$, the smaller the expected RMSE. The slope decreases with the sample size. Especially for small and moderate sample sizes selecting sampling units by minimising the $O_1 + O_3$ criterion has a positive effect on the quality of the estimates (smaller RMSE).

These results are somewhat different from the results of ? and ?. In these case studies cLH sampling appeared to be an inefficient design for selecting a calibration sample that is subsequently used for mapping. ? compared cLH sampling, CSC sampling, spatial coverage sampling (SCS) (Chapter 18), and simple random sampling for mapping soil organic carbon in France with a random forest model. The latter two sampling designs do not exploit the covariates in selecting the calibration units. Sample sizes were 100, 200, 500 and 1000. cLH sampling of calibration units performed worse (larger RMSE) than CSC sampling, and not significantly better than SI sampling for all sample sizes. For sample sizes 500 and 1000, SCS performed the best.

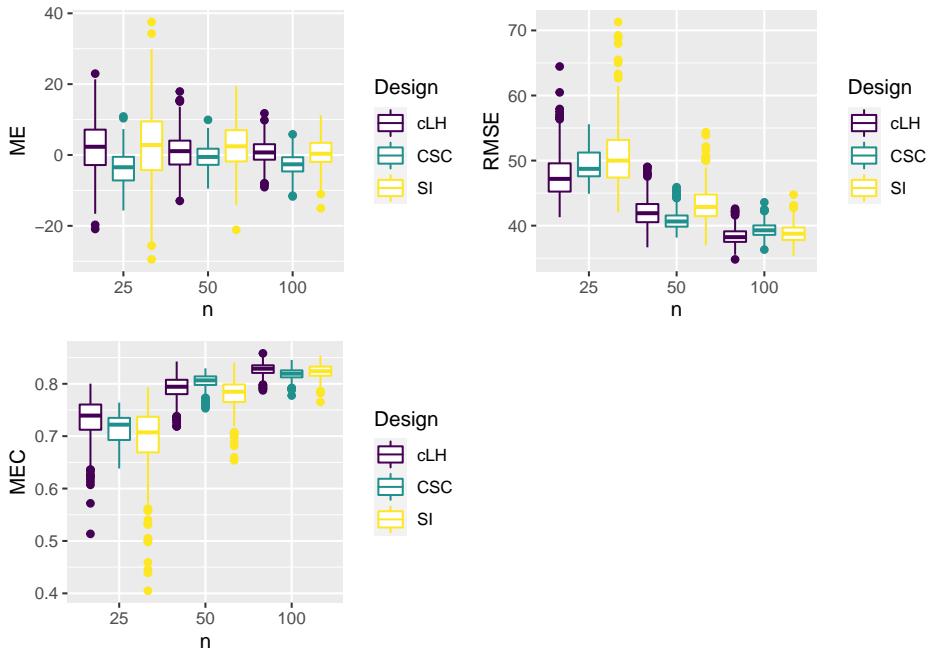


Figure 20.7: Boxplots of ME, RMSE and MEC of predictions with random forest models calibrated on conditioned Latin hypercube (cLH), covariate space coverage (CSC) and simple random (SI) samples of Eastern Amazonia, for sample sizes of 25, 50 and 100 units.

? compared cLH, CSC and simple random sampling for mapping soil classes by various models, among which a random forest model, in a study area in Germany. Sample sizes were 20, 30, 40, 50, 75, and 100 points. They found no relation between the minimisation criterion of cLH sampling and the overall accuracy of the map with predicted soil classes. Models calibrated on CSC samples performed better on average, i.e. on average the overall accuracy of the maps obtained by calibrating the models on these CSC samples had higher overall accuracy. cLH sampling was hardly better than simple random sampling.

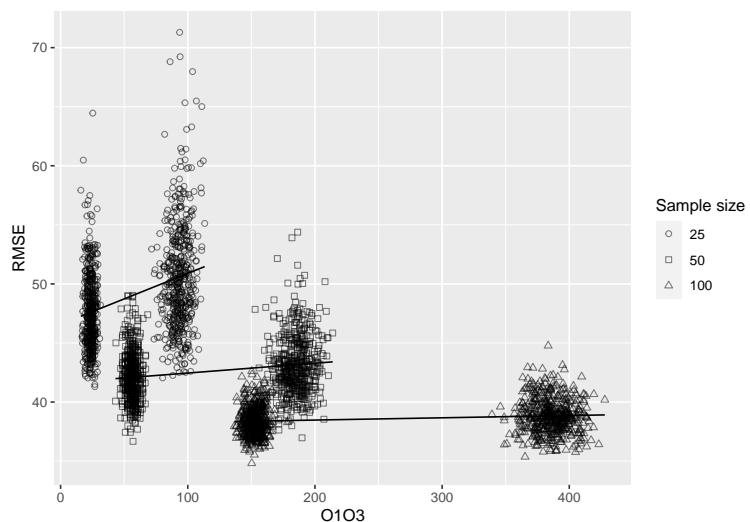


Figure 20.8: Relation between minimisation criterion $O_1 + O_3$ and the root mean squared error (RMSE) of random forest predictions of aboveground biomass in Eastern Amazonia for three sample sizes. Solid lines are the fitted simple linear regression models for RMSE with $O_1 + O_3$ as a predictor.

Chapter 21

Spatial response surface sampling

As with conditioned Latin hypercube sampling (cLHS), spatial response surface sampling is an experimental design adapted for spatial surveys. Experimental response surface designs aim at finding an optimum combination of the response within specified ranges of the factors. There are many types of response surface designs, see ?. With response surface sampling one assumes that some type of low order (linear or quadratic) regression model can be used to accurately approximate the relationship between the study variable and the covariates. A commonly used design is the central composite design. The data produced as a result of this design are used to fit a curved, quadratic surface, i.e. a multiple linear regression model with quadratic terms.

The response surface sampling approach is an example of a model-based sampling design. From that viewpoint I should have described this sampling design in the later part of this book dealing with model-based sampling. Sampling units are selected to implicitly optimise the estimation of the linear or quadratic regression model. However, this optimisation is done under one or more spatial constraints. Unconstrained optimisation of the sampling design under the linear regression model will not prevent the units from spatial clustering, see Figure 16.1(b). The assumption of independent data might be violated when the sampling units are spatially clustered. For that reason the response surface

sampling design selects samples with good spatial coverage, so that the design becomes robust against violation of the independence assumption.

? adapted the response surface methodology so that it can be applied in observational studies. Several problems needed to be tackled. First, when multiple covariates are used, the covariates must be decorrelated. Second, when sampling units are spatially clustered, the assumption in linear regression modelling of spatially uncorrelated model residuals can be violated. To address these two problems ? proposed the following procedure; see also ?:

1. Transform the covariate matrix into a scaled, centered, de-correlated matrix by principal components analysis (PCA).
2. Choose the response surface design type.
3. Select candidate sampling units based on the distance from the design points in PC-space and then select multiple units per design point.
4. Select the combination of candidate sampling units with the highest value for a criterion that quantifies how uniform the sample is spread across the study area.

This design has been applied, among others, for mapping soil salinity (ECE), using electromagnetic induction (EM) measurements and surface array conductivity measurements as predictors in multiple linear regression models. For applications, see ?, ?, ?, ? and ?.

Spatial response surface sampling is illustrated with the electromagnetic induction (EM) measurements (mS/m) of the apparent electrical conductivity on the 80 ha Cotton Research Farm in Uzbekistan (Section 1.3). The EM measurements in vertical dipole mode with transmitter at 1 m and 0.5 m from the receiver, are on transects covering the Cotton Research Farm (Figure 21.1). As a first step the natural logarithm of the two EM measurements are interpolated by ordinary kriging to a fine grid (Figure 21.2). These ordinary kriging predictions of $\ln\text{EM}$ are used as covariates in response surface sampling (Figure 21.1). The two covariates are strongly correlated, $r = 0.73$, as would be expected since they are of the same variable at two depths.

Function `prcomp` of the **stats** package (?) is used to compute the principal component scores for all units in the population (grid cells). The two covariates are centered and scaled, i.e. we compute standardised principal components.

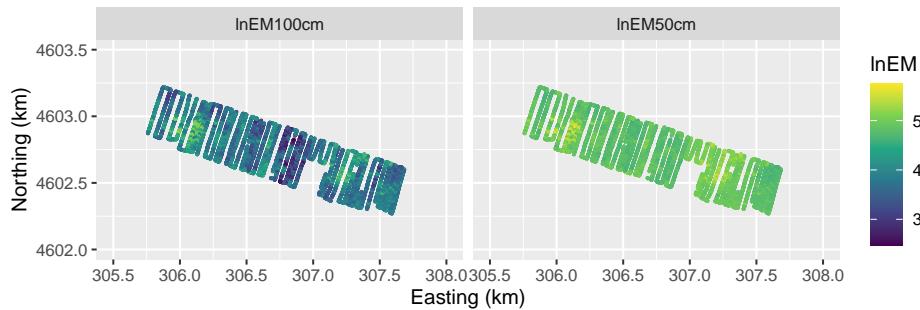


Figure 21.1: Natural logarithm of EM38 measurements on the Cotton Research Farm in Uzbekistan (with transmitter at 1 m and 0.5 m from receiver).

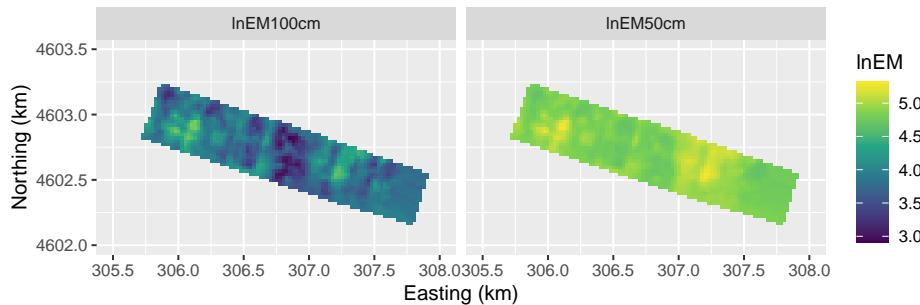


Figure 21.2: Interpolated surfaces of natural logarithm of EM38 measurements on the Cotton Research Farm in Uzbekistan, used as covariates in spatial response surface sampling.

```
grdsub <- subset(EM_CRF, select=c(lnEM100cm,lnEM50cm))
pc <- prcomp(grdsub, center=TRUE, scale=TRUE)
```

The means of the two principal component scores are 0, their standard deviations are 1.316 and 0.518. The principal component scores are then scaled so that their standard deviations become 1, i.e. they will have the same weight in the following steps.

```
EM_CRF$PC1 <- pc$x[,1]*1/pc$sdev[1]
EM_CRF$PC2 <- pc$x[,2]*1/pc$sdev[2]
```

Figure 21.4 shows a scatter plot of the two standardised principal component scores of all 1552 grid cells.

Function `ccd` of package `rsm` (?) is now used to generate a central composite response surface design (CCRSD). Argument `basis` specifies the number of factors, which is two in our case; argument `n0` is the number of center points, and argument `alpha` determines the position of the “star points” (explained hereafter).

```
library(rsm)
set.seed(314)
print(ccdesign <- ccd(basis=2, n0=1, alpha="rotatable"))
```

	run.order	std.order	x1.as.is	x2.as.is	Block
1	1	4	1.000000	1.000000	1
2	2	5	0.000000	0.000000	1
3	3	2	1.000000	-1.000000	1
4	4	1	-1.000000	-1.000000	1
5	5	3	-1.000000	1.000000	1
6	1	5	0.000000	0.000000	2
7	2	3	0.000000	-1.414214	2
8	3	1	-1.414214	0.000000	2
9	4	4	0.000000	1.414214	2
10	5	2	1.414214	0.000000	2

```
Data are stored in coded form using these coding formulas ...
x1 ~ x1.as.is
x2 ~ x2.as.is
```

The experiment consists of two blocks, each of five experimental units. Block 1, the so-called cube block, consists of one center point and four cube points. In the experimental unit represented by the center point both factors have levels in the center of the experimental range. In the experimental units represented by the cube points the levels of both factors is either -1 or +1 unit in the design space. Block 2, referred to as the star block, consists of one center point and

four star points. With `alpha = rotatable` the start points are on the circle circumscribing the square (Figure 21.3).

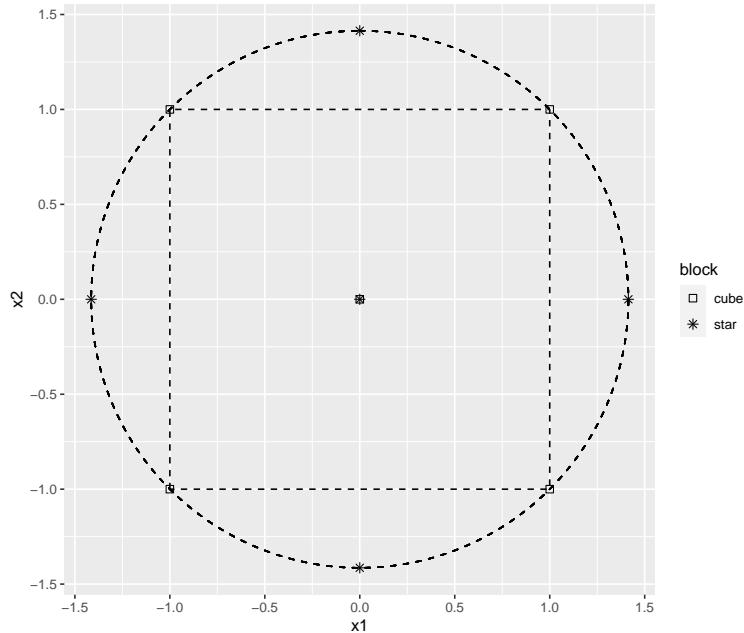


Figure 21.3: Rotatable central composite response surface design for two factors.

To adapt this design for an observational study, we drop one of the center points $(0,0)$.

```
ccd_df <- data.frame(x1=ccdesign$x1, x2=ccdesign$x2)
ccd_df <- ccd_df[-6,]
```

The points in the CCRSD design are adjusted so that a large proportion p of the bivariate standardised principal component scores of the population units are within a circle of a given radius. This radius is computed as a sample quantile of the empirical distribution of the distances of the points in the scatter to the center. For p I chose 0.7.

```

d <- sqrt(EM_CRF$PC1^2+EM_CRF$PC2^2)
radius <- quantile(d, p=0.7)
print(radius)

70%
1.508801

ccd_df$x1 <- ccd_df$x1*radius
ccd_df$x2 <- ccd_df$x2*radius

```

The next step is to select for each design point several candidate sampling units. For each of the nine design points 0.5% of the grid cells are selected that are closest to that design point. This results in 9×8 candidate sampling points.

```

EM_CRF$unit <- 1:nrow(EM_CRF)
candi_all <- NULL
for (i in 1:nrow(ccd_df)) {
  d2dpnt <- sqrt((EM_CRF$PC1-ccd_df$x1[i])^2+
    (EM_CRF$PC2-ccd_df$x2[i])^2)
  q <- quantile(d2dpnt, p=0.005)
  units_close <- which(d2dpnt < q)
  candi <- EM_CRF[units_close,c("unit","x1","x2","PC1","PC2")]
  candi$dpnt <- i
  candi_all <- rbind(candi_all,candi)
}

```

Figure 21.4 shows the nine clusters of candidate sampling points around the design points. Note that the location of the candidate points associated with the design points with coordinates $(-1.51, -1.51)$, $(0, -2.13)$, $(1.51, -1.51)$ and $(2.13, 0)$ are all far inside the circle that passes through the design points. So for the optimised sample the value of one of the principal component scores will be less extreme than the ideal values according to the CCRSD design.

Figure 21.5 shows that in geographical space for most design points there are multiple clusters of candidate units. For instance, for design point 9 there are three clusters of candidate sampling units. Due to this there is scope to optimise the sample computationally.

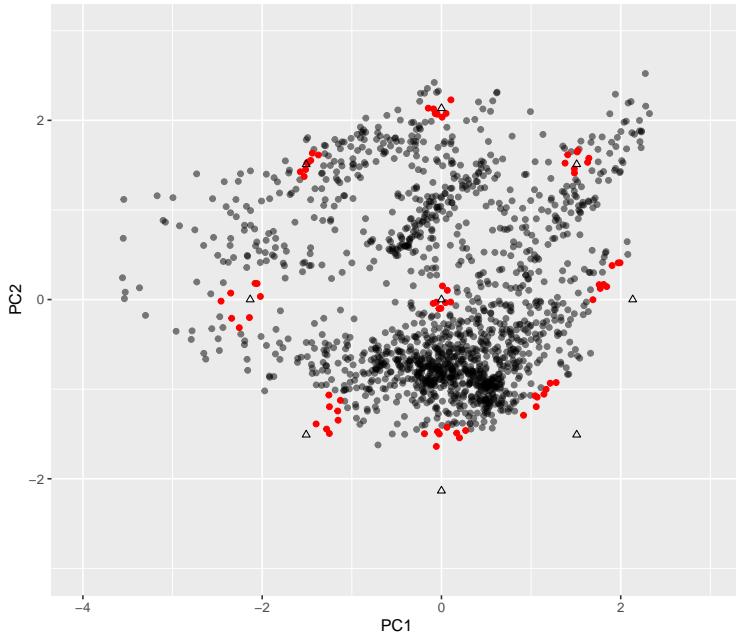


Figure 21.4: Clusters of units (red points) around the design points (triangles) of a central composite design (two covariates), serving as candidate sampling points.

As a first step an initial subsample from the candidate sampling units is selected by stratified simple random sampling, using the factor `dpnt` as strata. Function `strata` of package **sampling** is used for stratified random sampling (?).

```
library(sampling)
set.seed(314)
units_stsi <- sampling::strata(
  candi_all, stratanames="dpnt", size=rep(1,9))
mysample0 <- getdata(candi_all, units_stsi)
mysample0 <- mysample0[,c("unit","x1","x2","PC1","PC2","dpnt")]
```

The locations of the nine sampling points are now optimised by minimising a criterion. Two minimisation criteria are implemented, a geometric criterion and

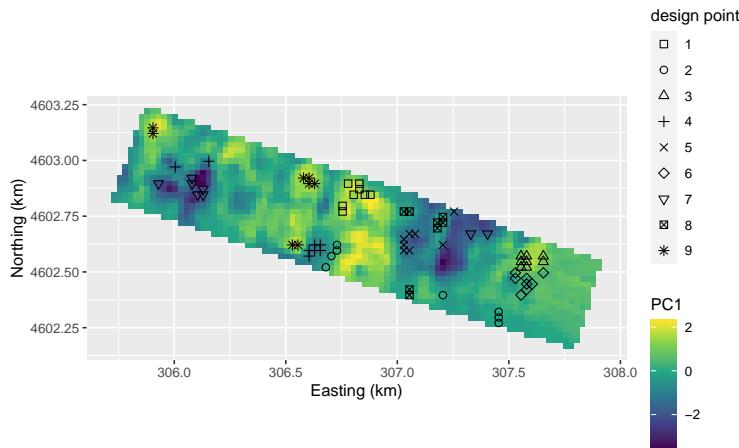


Figure 21.5: Candidate sampling points plotted in geographical space on the first standardised principal component.

a model-based criterion.

In the geometric criterion (as proposed by ?) for each sampling point the shortest distance to the other points is computed, and the logarithm of this shortest distance is computed. The minimisation criterion is the negative of the sample mean of the logarithm of shortest distance.

The model-based minimisation criterion is the average correlation of the sampling points. This criterion requires as input the parameters of a residual correlogram (see Section 22.2). I assume an exponential correlogram without nugget, so that the only parameter to be chosen is the distance parameter ϕ . Three times ϕ is referred to as the effective range of the exponential covariance function. The correlation of two random variables at this distance is 0.05. A penalty term is added to these criteria, equal to the average distance, in the space spanned by the principal component scores, of the sampling points to the associated design points, multiplied by a weight. With weights > 0 sampling points close to the design points are preferred over more distant points.

A function is defined for computing the minimisation criterion. Given a chosen value for ϕ , the 9×9 distance matrix of the sampling points can be converted into a correlation matrix, using function `variogramLine` of package `gstat` (?). Argument `weight` is an optional argument with default value 0.

```
getCriterion <- function(mysample, dpnt, weight=0, phi=NULL){
  D2dpnt <- sqrt((mysample$PC1-dpnt$x1)^2+
    (mysample$PC2-dpnt$x2)^2)
  D <- as.matrix(dist(mysample[,c("x1","x2")]))
  if (!is.null(phi)) {
    vgmodel <- vgm(model="Exp", psill=1, range=phi)
    C <- variogramLine(vgmodel, dist_vector=D, covariance=TRUE)
    criterion_cur <- mean(C)+mean(D2dpnt)*weight
  } else {
    diag(D) <- NA
    logdmin <- apply(D, MARGIN=1, FUN=min, na.rm=TRUE) %>% log(.)
    criterion_cur <- mean(-logdmin)+mean(D2dpnt)*weight
  }
}
```

The function `getCriterion` is used to compute the geometric criterion for the initial sample.

```
criterion_geo <- getCriterion(mysample=mysample0, dpnt=ccd_df)
```

The initial value of the geometric criterion is -5.185. The initial value for the model-based criterion is computed with an effective range of 50 m. Note that it does not make sense to make this range smaller than the size of the grid cells, which is 25 m in our case. For smaller ranges the correlation matrix is for any sample a matrix with zeroes. If the effective range is smaller than the smallest distance between two points in a cluster, the mean correlation is equal for all samples.

```
phi <- 50
criterion_mb <- getCriterion(
  mysample=mysample0, dpnt=ccd_df, phi=phi)
```

The initial value of the model-based criterion is 0.118.

The objective function defining the minimisation criterion is minimised with simulated annealing (? , ?). One sampling point is randomly selected, and replaced by another candidate sampling point from the same cluster. If the

criterion of the new sample is smaller than that of the current sample, the new sample is accepted, if it is larger it is accepted with a probability that is a function of the change in the criterion (the larger the increase, the smaller the acceptance probability) and of an annealing parameter named the temperature (the higher the temperature, the larger the probability of accepting a new, poorer sample, given an increase of the criterion). See Section 24.1 for a more detailed introduction to simulated annealing. The annealing functions for spatial response surface sampling are defined in R script AnnealingFunctions4SpatialResponseSurfaceSampling.R¹².

```
source('Rscripts/AnnealingFunctions4SpatialResponseSurfaceSampling.R')
set.seed(314)
mySRSsample <- anneal(
  mysample=mysample0, candidates=candi_all, dpnt=ccd_df,
  phi=50, T_ini=1, coolingRate=0.9,
  maxPermuted=25*nrow(mysample0), maxNoChange=20,
  verbose=TRUE)
```

Figure 21.6 shows the optimised CCRSD samples, obtained with the geometric and model-based criterion, plotted together with the design points. The two optimised samples are very similar.

Figure 21.7 shows the two optimised CCRSD samples plotted in geographical space on the first standardised principal component scores.

21.1 Increasing the sample size

Nine points are rather few for fitting a polynomial regression model, especially for a second-order polynomial with interaction. Therefore, in experiments often multiple observations are done for each design point. Increasing the sample size of a response surface sample in observational studies is not straightforward. The challenge is to avoid spatial clustering of sampling points. A simple solution is to select multiple points from each subset of candidate sampling units. The success of this solution depends on the strength of the spatial clustering of the

¹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/AnnealingFunctions4SpatialResponseSurfaceSampling.R>

²<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/AnnealingFunctions4SpatialResponseSurfaceSampling.R>

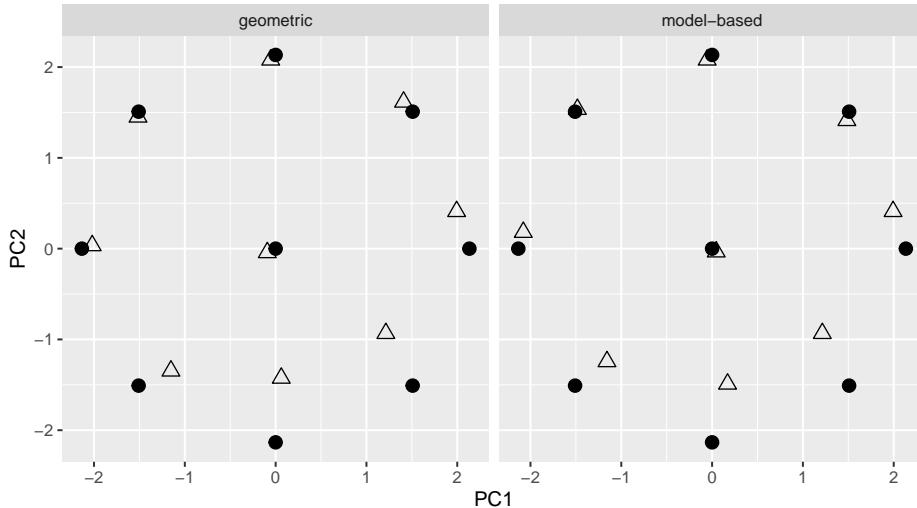


Figure 21.6: Principal component scores of CCRSD sample (triangles), optimised with the geometric and model-based criterion. Dots: design points.

candidate sampling units per design point. For the Cotton Research Farm for most design points the candidate sampling units are not in one spatial cluster, so in this case this solution may work properly. I increased the number of candidate sampling units per design point (argument p in function `quantile` is now 0.01), so that there is a larger choice in the optimisation of the sample pattern.

```
candi_all <- NULL
for (i in 1:nrow(ccd_df)) {
  d2dpnt <- sqrt((EM_CRF$PC1-ccd_df$x1[i])^2+
    (EM_CRF$PC2-ccd_df$x2[i])^2)
  q <- quantile(d2dpnt, p=0.01)
  units_close <- which(d2dpnt < q)
  candi <- EM_CRF[units_close,c("unit","x1","x2","PC1","PC2")]
  candi$dptn <- i
  candi_all <- rbind(candi_all,candi)
}
```

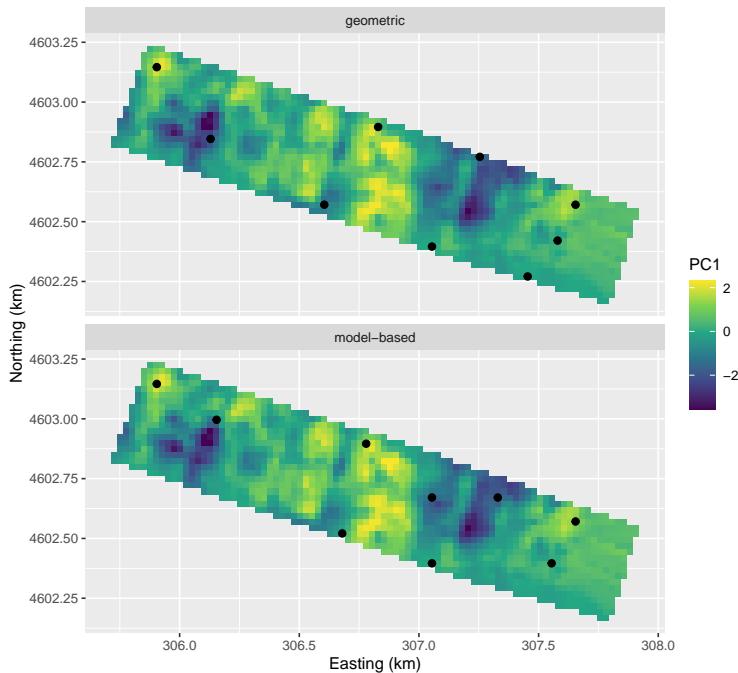


Figure 21.7: CCRSD sample from the Cotton Research Farm in Uzbekistan, optimized with the geometric and model-based criterion.

A stratified random subsample of two points per stratum is selected, which serves as an initial sample.

```
set.seed(314)
units_stsi <- sampling::strata(
  candi_all, stratanames="dpnt", size=rep(2,9))
mysample0 <- getdata(candi_all, units_stsi)
mysample0 <- mysample0[,c("unit", "x1", "x2", "PC1", "PC2", "dpnt")]
```

The `data.frame` with the design points must be doubled. Note that the order of the design points must be equal to the order in the stratified subsample.

```

tmp <- data.frame(ccd_df, dpnt=1:9)
dpnt <- rbind(tmp, tmp)
ccd_df2 <- dpnt[order(dpnt$dpnt),]

```

Figures 21.8 and 21.9 show the optimised CCRSD sample of eighteen points in geographical and principal component space, respectively, obtained with the model-based criterion, an effective range of 50 m, and zero weight for the penalty term. Sampling points are not spatially clustered, so I do not expect violation of the assumption of independent residuals. In principal component space all points are pretty close to the design points, except for the four design points in the lower right corner, where no candidate units near these design points are available.

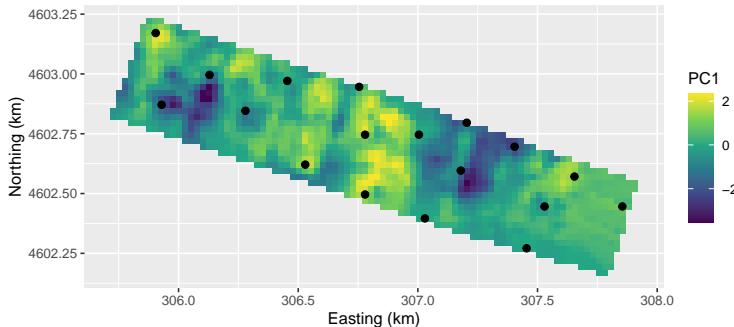


Figure 21.8: CCRSD sample with two points per design point, from the Cotton Research Farm in Uzbekistan.

21.2 Stratified spatial response surface sampling

The sample size can also be increased by stratified spatial response surface sampling. The strata are subareas of the study area. When the subsets of candidate sampling units for some design points are strongly spatially clustered, the final optimised sample obtained with the method of the previous section may also show strong spatial clustering. An alternative is then to split the study area into two or more subareas (strata), and to select from each stratum candidate sampling units. This guarantees that for each design point we have at least as many spatial clusters of candidate units as we have strata. The spatial

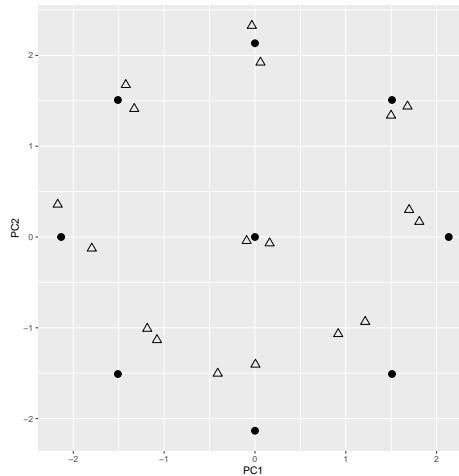


Figure 21.9: CCRSD sample (triangles) with two points per design point (dots), optimised with model-based criterion.

strata are not used for fitting separate regression models; all data are used to fit one (second-order) polynomial regression model.

Figure 21.10 shows two subareas used as strata in stratified response surface sampling of the Cotton Research Farm.

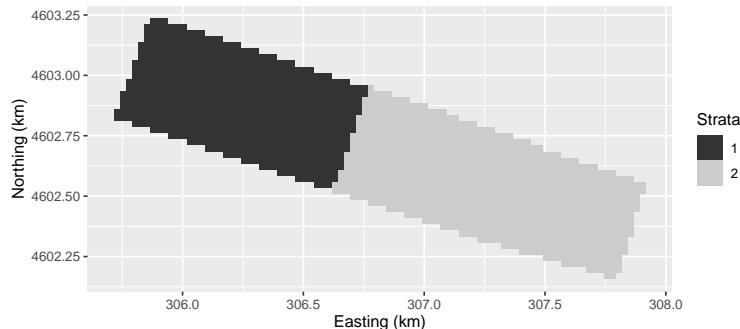


Figure 21.10: Two subareas of the Cotton Research Farm in Uzbekistan used as strata in stratified CCRSD sampling.

The candidate sampling units are selected in a double for-loop. The outer loop is over the strata, the inner loop over the design points. Note that the variable `dpnt` continues to increase by 1 after the inner-loop over the nine design points in subarea 1 is completed, so that the variable `dpnt` (used as a stratum in subsampling the sample of candidate sampling points) now has values 1, 2, ..., 18. Stratum 2 is larger than stratum 1. To select an equal number of candidate sampling points per design point in both strata (eight points), I adapted argument `p` of function ‘`quantile`’.

```
candi_all <- NULL
p <- c(0.0105,0.009)
for (h in c(1,2)) {
  units <- which(EM_CRF$subarea==h)
  data_stratum <- EM_CRF[units,]
  candi_stratum <- NULL
  for (i in 1:nrow(ccd_df)) {
    d2dpnt <- sqrt((data_stratum$PC1-ccd_df$x1[i])^2+
      (data_stratum$PC2-ccd_df$x2[i])^2)
    q <- quantile(d2dpnt, p=p[h])
    units_close <- which(d2dpnt < q)
    candi <- data_stratum[units_close,
      c("unit","x1","x2","PC1","PC2")]
    candi$dpnt <- i + (h-1)*nrow(ccd_df)
    candi_stratum <- rbind(candi_stratum, candi)
  }
  candi_all <- rbind(candi_all, candi_stratum)
}
```

As before, `dpnt` is used as a stratum identifier to subsample the candidate sampling units. Finally, the number of rows in the `data.frame` `ccd_df` with the design points is doubled.

```
set.seed(314)
units_stsi <- sampling::strata(
  candi_all, stratanames="dpnt", size=rep(1,18))
mysample0 <- getdata(candi_all, units_stsi)
mysample0 <- mysample0[,c("unit","x1","x2","PC1","PC2","dpnt")]
```

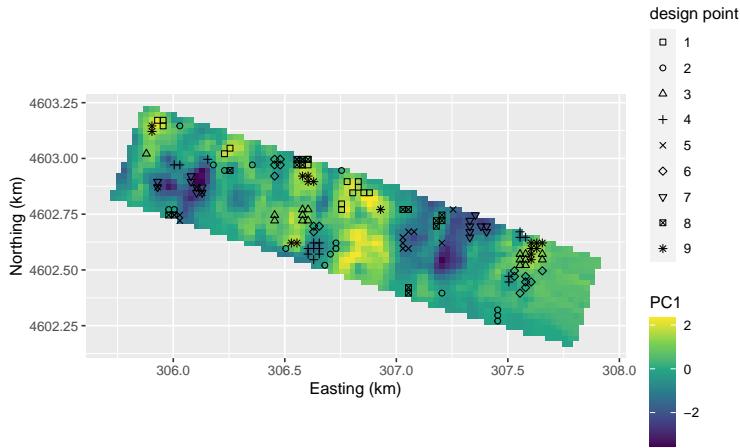


Figure 21.11: Candidate sampling points for stratified CCRSD sampling, plotted on first principal component (PC1)

```
ccd_df2 <- rbind(ccd_df, ccd_df)
```

Figures 21.12 and 21.13 show the optimised sample of eighteen points in geographical and principal component space, obtained with the model-based criterion with an effective range of 50 m. The pattern in the principal component space is worse compared to the pattern in Figure 21.9. In stratum 1 the distance to the star point at the top and the upper left cube point is very large. In this stratum no population units are present that are close to these two design points. The distance could be decreased a bit by adding a penalty term to the minimisation criterion that is proportional to this distance. However, the optimised sample obtained with `weight=5` shows much stronger spatial clustering. In addition, there are some pairs of points with a separation distance smaller than the effective range used in optimisation, so that violation of the independence assumption becomes more likely.

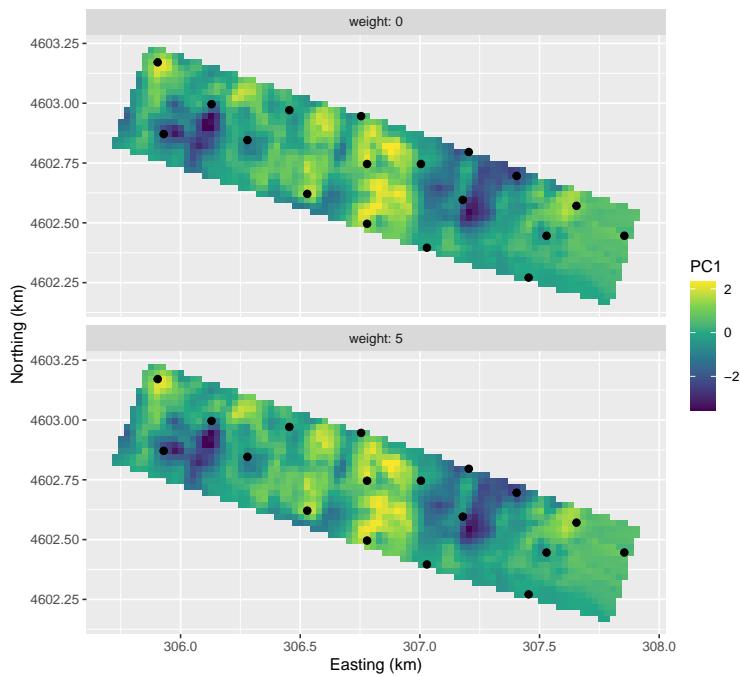


Figure 21.12: Stratified CCRSD samples from the Cotton Research Farm in Uzbekistan, optimised with model-based criterion, obtained without ($\text{weight} = 0$) and with penalty ($\text{weight} = 5$) for large average distance to design points.

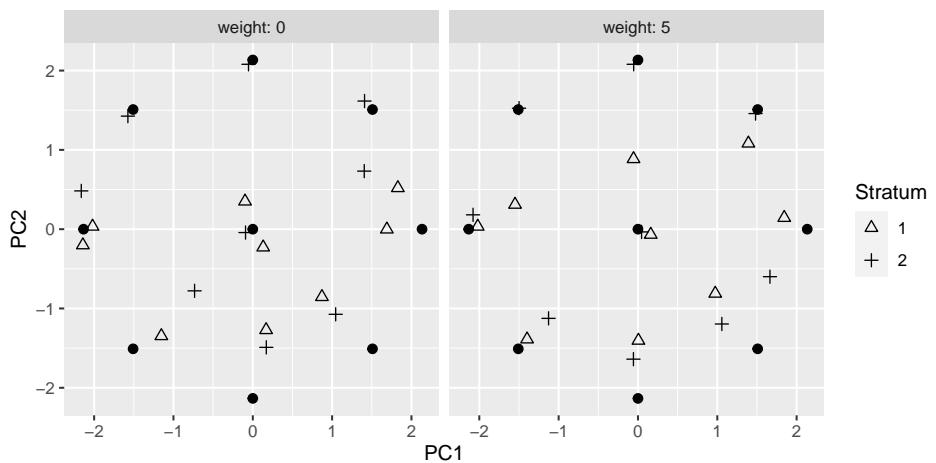


Figure 21.13: Principal component scores of the stratified CCRSD sample, optimised with model-based criterion, obtained without ($\text{weight} = 0$) and with penalty ($\text{weight} = 5$) for a large average distance to design points (dots).

Chapter 22

Introduction to kriging

In the following chapters a geostatistical model, i.e. a statistical model of the spatial variation of the study variable, is used to optimise the sample size and/or spatial pattern of the sampling units. This is the core of model-based sampling. This chapter is a short introduction to geostatistical modelling.

In Chapter 13 we have already seen how a geostatistical model can be used to optimise probability sampling designs for estimating the population mean or total. In the following chapters the focus is on mapping. A map of the study variable is obtained by predicting the study variable at the nodes of a fine discretisation grid. Spatial prediction using a geostatistical model is referred to as kriging (?).

With this prediction method besides a map of the kriging predictions, a map of the variance of the prediction error is obtained. I show hereafter that this prediction error variance is not influenced by the values of the study variable at the sampling units. For that reason it is possible to search, before the start of the survey, for the sampling locations that lead to the minimum prediction error variance averaged over all nodes of a fine prediction grid, provided that the semivariogram is known.

The kriging predictions and prediction error variances are derived from a statistical model of the spatial variation of the study variable. There are several versions of kriging, building on different models of the spatial variation. In *ordinary kriging* (OK) it is assumed that the mean of the study variable is constant,

i.e. the same everywhere (?):

$$\begin{aligned} Z(\mathbf{s}) &= \mu + \epsilon(\mathbf{s}) \\ \epsilon(\mathbf{s}) &\sim \mathcal{N}(0, \sigma^2) \\ \text{Cov}(\epsilon(\mathbf{s}), \epsilon(\mathbf{s}')) &= C(\mathbf{h}), \end{aligned} \tag{22.1}$$

with $Z(\mathbf{s})$ the study variable at location \mathbf{s} , μ the constant mean, independent of the location \mathbf{s} , $\epsilon(\mathbf{s})$ the error or residual (difference between study variable z and mean μ) at location \mathbf{s} , and $C(\mathbf{h})$ the covariance of ϵ at two points separated by vector $\mathbf{h} = \mathbf{s} - \mathbf{s}'$. $C(\cdot)$ is the covariance function, also referred to as the covariogram, and a scaled version of it obtained by dividing $C(\cdot)$ by the variance $C(0)$, is the correlation function or correlogram. Stated otherwise, in ordinary kriging we assume *stationarity in the mean* over the area to be mapped. The residuals are not independent, but spatially correlated. So if we have a large positive residual at some location \mathbf{s} , then in many cases we also have a positive residual at a nearby location \mathbf{s}' .

If the data set is of substantial size, it is possible to specify that not all sample data be used to predict the study variable at a prediction location, but only the sample data in some predefined neighbourhood. This implies that the stationarity assumption is relaxed to the often more realistic assumption of a constant mean within neighbourhoods.

In ordinary kriging, the value of the study variable at a prediction location \mathbf{s}_0 , $\widehat{Z}(\mathbf{s}_0)$, is predicted as a weighted average of the observations at the sampling locations within the neighbourhood:

$$\widehat{Z}_{\text{OK}}(\mathbf{s}_0) = \sum_{i=1}^n \lambda_i Z(\mathbf{s}_i), \tag{22.2}$$

where $Z(\mathbf{s}_i)$ is the study variable at the i^{th} sampling location, and λ_i is the weight attached to this location. The weights should be related to the strength of correlation of the study variable at the sampling location and the prediction location. Note that as the mean is assumed constant (Equation (22.2)), the correlation of the study variable Z is equal to the correlation of the residual ϵ . Roughly speaking, the stronger this *autocorrelation* (auto refers to the fact that the same variable is considered at both locations), the larger the weight must be.

So if we have a model for this autocorrelation, then we can use this model to find the optimal weights. Further, if two sampling locations are very close, the weight attached to these two locations should not be twice the weight attached to a single, isolated sampling location at the same distance of the interpolation node. This explains that in computing the kriging weights, besides the covariances of the n pairs of interpolation node and sampling location, also the covariances of the $n \cdot (n - 1)/2$ pairs that can be formed with the n sampling units are used. For OK, the optimal weights, i.e. the weights that lead to the model-unbiased¹ predictor with minimum error variance (best linear unbiased predictor) can be found by solving the following $(n + 1)$ equations:

$$\begin{aligned} \sum_{j=1}^n \lambda_j C(\mathbf{s}_1, \mathbf{s}_j) + \nu &= C(\mathbf{s}_1, \mathbf{s}_0) \\ \sum_{j=1}^n \lambda_j C(\mathbf{s}_2, \mathbf{s}_j) + \nu &= C(\mathbf{s}_2, \mathbf{s}_0) \\ &\vdots && , \\ \sum_{j=1}^n \lambda_j C(\mathbf{s}_n, \mathbf{s}_j) + \nu &= C(\mathbf{s}_n, \mathbf{s}_0) \\ \sum_{j=1}^n \lambda_j &= 1 \end{aligned} \quad (22.3)$$

where $C(\mathbf{s}_i, \mathbf{s}_j)$ is the covariance of the i th and j th sampling location, $C(\mathbf{s}_i, \mathbf{s}_0)$ the covariance of the i th sampling location and the interpolation node s_0 , and ν an extra parameter to be estimated, referred to as the Lagrange multiplier. This Lagrange multiplier must be included in the set of equations because the error variance is minimised under the constraint that the kriging weights sum to one, see the final line in Equation (22.3). This constraint ensures that the OK-predictor is model-unbiased.

It is convenient to write this system of equations in matrix form:

$$\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} & 1 \\ C_{21} & C_{22} & \dots & C_{2n} & 1 \\ \vdots & \vdots & \dots & \vdots & 1 \\ C_{n1} & C_{n2} & \dots & C_{nn} & 1 \\ 1 & 1 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \nu \end{bmatrix} = \begin{bmatrix} C_{10} \\ C_{20} \\ \vdots \\ C_{n0} \\ 1 \end{bmatrix}. \quad (22.4)$$

¹Model-unbiasedness is explained in Chapter 27.

Replacing submatrices by single symbols results in the shorthand matrix equation:

$$\begin{bmatrix} \mathbf{C} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} = \begin{bmatrix} \mathbf{c}_0 \\ 1 \end{bmatrix}. \quad (22.5)$$

The kriging weights λ and the Lagrange multiplier ν can then be computed by premultiplying both sides of Equation (22.5) with the inverse of the first matrix of this equation:

$$\begin{bmatrix} \lambda \\ \nu \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{c}_0 \\ 1 \end{bmatrix}. \quad (22.6)$$

The variance of the prediction error (ordinary kriging variance) at a prediction location equals

$$V_{OK}(\hat{Z}(\mathbf{s}_0)) = \sigma^2 - \lambda^T \mathbf{c}_0 - \nu, \quad (22.7)$$

with σ^2 the sill of the semivariogram (a priori variance). This equation shows that the ordinary kriging variance is not a function of the data at the sampling locations. Given a covariance function it is fully determined by the coordinates of the sampling locations and of the prediction location. It is this property of kriging that makes it possible to optimise the grid-spacing, and as we will see in Chapter 24 to optimise the spatial pattern of the sampling units, given a requirement on the kriging variance. If the kriging variance were a function of the data at the sampling locations, optimisation would be much more complicated.

Commonly the kriging variance is expressed in terms of the *semivariances* between the sampling points and a prediction point (hereafter the relation between a covariance function and a semivariogram is explained):

$$V_{OK}(\hat{Z}(\mathbf{s}_0)) = \lambda^T \gamma_0 + \nu, \quad (22.8)$$

with γ_0 the vector with semivariances between the sampling points and a prediction point.

Computing the kriging predictor requires a model for the covariance as a function of the vector separating two locations. Often the covariance is modelled as a function of the length of the separation vector only, so as a function of the Euclidian distance between two locations. We then assume isotropy. In general practice, the covariance function is not used in kriging, rather, a semivariogram. A semivariogram is a model of the *dissimilarity* of the study variable at two locations. The dissimilarity is quantified by half the squared difference of the study variable values at two points. A covariance function and semivariogram are related by, see Figure 22.1:

$$\gamma(\mathbf{h}) = \sigma^2 - C(\mathbf{h}) . \quad (22.9)$$

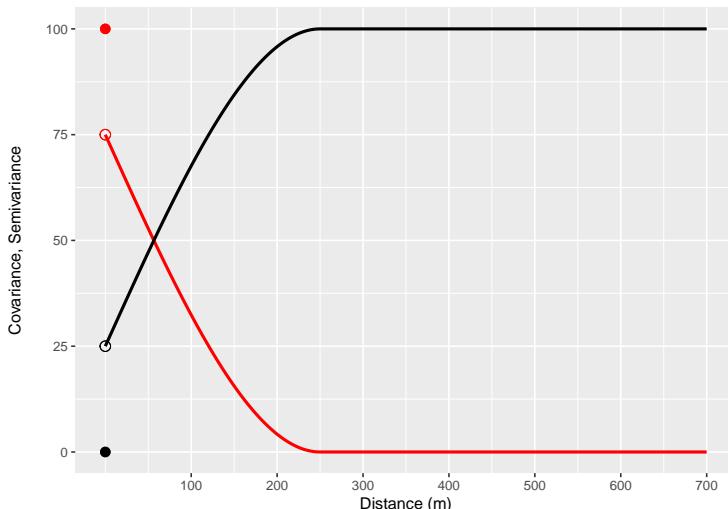


Figure 22.1: Spherical covariance function (red line and dot) and semivariogram (black line and dot).

Only authorized functions are allowed, ensuring that the variance of any linear combination of random variables, like the kriging predictor, is positive. Commonly used functions are an exponential, spherical and Matérn model.

The spherical semivariogram model has three parameters:

1. Nugget (c_0): this is where the semivariogram crosses the y-axis (in Figure

22.1: 25).

2. Partial sill (c_1): the difference between the maximum semivariance and the nugget (in Figure 22.1: 75).
3. Range (ϕ): this is the distance at which the semivariance reaches its maximum (in Figure 22.1: 500 m).

The formula for the spherical semivariogram is:

$$\gamma(h) = \begin{cases} 0 & \text{if } h = 0 \\ c_0 + c_1 \left[1 - \frac{3}{2} \left(\frac{h}{\phi} \right) + \frac{1}{2} \left(\frac{h}{\phi} \right)^3 \right] & \text{if } 0 < h \leq \phi \\ c_0 + c_1 & \text{if } h > \phi \end{cases} \quad (22.10)$$

The sum of the nugget and the partial sill is referred to as the sill (or sill variance, or a priori variance). An exponential semivariogram model also has three parameters. Its formula is

$$\gamma(h) = \begin{cases} 0 & \text{if } h = 0 \\ c_0 + c_1 \exp(-h/\phi) & \text{if } h > 0 \end{cases} \quad (22.11)$$

In an exponential semivariogram the semivariance goes asymptotically to a maximum; it never reaches it. In an exponential semivariogram the range parameter is replaced by the distance parameter. In an exponential semivariogram without nugget the semivariance at three times the distance parameter is at 95% of the sill. Three times the distance parameter is referred to as the *effective* or *practical* range.

In following chapters I also use a correlogram, which is a scaled covariance function, such that the sill of the correlogram equals one:

$$\rho(\mathbf{h}) = \frac{C(\mathbf{h})}{\sigma^2}. \quad (22.12)$$

To illustrate that the ordinary kriging variance is independent of the values of the study variable at the sampling locations, I simulated a spatial population of 50×50 units. At each unit a value of the study variable is simulated, using the semivariogram of Figure 22.1. This is repeated ten times, resulting in ten

simulated populations of 2500 units. Figure 22.2 shows two of the ten simulated populations. Note that the two simulations clearly show spatial structure (patches of similar values).

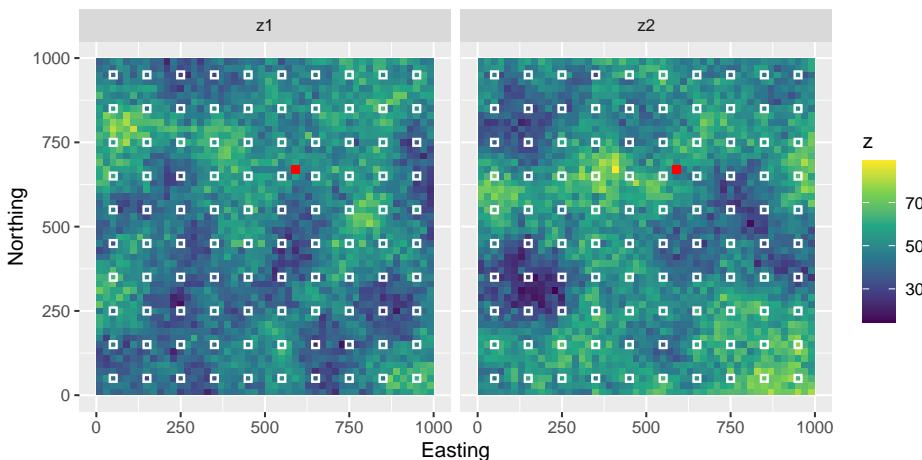


Figure 22.2: Two maps simulated with the spherical semivariogram of the figure above, centered square grid of sampling units, and prediction unit (red cell with coordinates (590,670)).

The simulated fields are sampled on a centered square grid with a spacing of 100 distance units, resulting in a sample of 100 units. Each sample is used one-by-one to predict the study variable at one prediction location (see Figure 22.2), using again the semivariogram of Figure 22.1. The semivariogram is specified by function `vgm` of package `gstat` (?). Usually this semivariogram is estimated from a sample, see Chapter 25, but here we assume that it is known. Function `krige` of package `gstat` is used for kriging. The formula argument specifies the dependent (study variable) and independent variables (covariates). The formula `z ~ 1` means that we do not have covariates, and that predictions are done by ordinary kriging (or simple kriging, see Section 22.2). Argument `locations` is a file of class `SpatialPointsDataFrame` with the spatial coordinates and observations. Argument `newdata` is a file of class `SpatialPoints` with the locations where we want to predict. Argument `nmax` can be used to specify the number of nearest observations that should be used in kriging.

```

library(sp)
library(gstat)
vgmodel <- vgm(model="Sph", nugget=25, psill=75, range=500)
gridded(mypop) <- ~s1+s2
mysample <- spsample(
  x=mypop, type="regular", cellsize=c(100,100),
  offset=c(0.5,0.5))
zsim_sample <- over(mysample, mypop)
coordinates(s_0) <- ~s1+s2
zpred_OK <- v_zpred_OK <- NULL
for (i in 1:ncol(Z)) {
  mysample$z <- zsim_sample[,i]
  predictions <- krige(
    formula=z~1,
    locations=mysample,
    newdata=s_0,
    model=vgmodel,
    debug.level=0)
  zpred_OK[i] <- predictions$var1.pred
  v_zpred_OK[i] <- predictions$var1.var
}

```

As can be seen below, unlike the predicted value, the ordinary kriging variance produced from the different simulations is constant.

22.1 Block-kriging

In the previous section the support of the prediction units is equal to that of the sampling units. So if the observations are done at points (point support), the support of the predictions are also points. There is no change of support. In some cases we may prefer predictions at a larger support than that of the observations. For instance, we may prefer predictions of the average concentration of some soil pollutant of blocks of $5 \text{ m} \times 5 \text{ m}$, instead of predictions at points, simply because of practical relevance. If the observations are at points, there is a change of support, from points to blocks. Kriging with a prediction support that is larger than the support of the sample data is referred to as block-kriging. Kriging without change of support, the sample support and prediction

Table 22.1: Ordinary kriging predictions and kriging variance at a fixed prediction location for ten data sets with simulated values at a square grid.

Kriging prediction	Kriging variance
52.97348	42.0167
55.59440	42.0167
52.09590	42.0167
51.12708	42.0167
62.91966	42.0167
43.35531	42.0167
52.66519	42.0167
48.27031	42.0167
47.20410	42.0167
45.96422	42.0167

support are equal, is referred to as point-kriging. Note that point-kriging does not necessarily imply that the support is a point, it can be, for instance, a small block.

In block-kriging the mean of a prediction block \mathcal{B}_0 is predicted as a weighted average of the observations at the sampling units. The kriging weights are derived much in the same way as in point-kriging (Equations (22.3) to (22.6)). In block-kriging the covariance between a sampling point i and a prediction point, $C(\mathbf{s}_i, \mathbf{s}_0)$ is replaced by the *mean* covariance between the sampling point and a prediction block $\bar{C}(\mathbf{s}_i, \mathcal{B}_0)$ (Equation (22.3)). This mean covariance can be approximated by discretising the prediction block by a fine grid, computing the covariance between a sampling point i and each of the discretisation points, and averaging.

The variance of the prediction error of the block-mean (block-kriging variance) equals

$$V_{\text{OBK}}(\widehat{\bar{Z}}(\mathcal{B}_0)) = \lambda^T \bar{\gamma}(\mathcal{B}_0) + \nu - \bar{\gamma}(\mathcal{B}_0, \mathcal{B}_0), \quad (22.13)$$

with $\bar{\gamma}(\mathcal{B}_0)$ the vector with mean semivariances between the sampling points and a prediction block, and $\bar{\gamma}(\mathcal{B}_0, \mathcal{B}_0)$ the mean semivariance within the prediction block. Comparing this with the variance with point-kriging (Equation (22.8))

shows that the block-kriging variance is smaller than the point-kriging variance by an amount approximately equal to the mean semivariance within a prediction block. Recall from Chapter 13 that the mean semivariance within a block is a model-based prediction of the variance within a block (Equation (13.3)).

22.2 Kriging with an external drift

In kriging with an external drift (KED) the spatial variation of the study variable is modelled as the sum of a linear combination of covariates and a spatially correlated residual:

$$\begin{aligned} Z(\mathbf{s}) &= \sum_{k=0}^p \beta_k x_k(\mathbf{s}) + \epsilon(\mathbf{s}) \\ \epsilon(\mathbf{s}) &\sim \mathcal{N}(0, \sigma^2) \\ \text{Cov}(\epsilon(\mathbf{s}), \epsilon(\mathbf{s}')) &= C(\mathbf{h}), \end{aligned} \tag{22.14}$$

with $x_k(\mathbf{s})$ the value of the k th covariate at location \mathbf{s} ($x_0 = 1$ for all locations), p the number of covariates, and $C(\mathbf{h})$ the covariance of the residuals at two points separated by vector $\mathbf{h} = \mathbf{s} - \mathbf{s}'$. The constant mean μ in Equation (22.2) is replaced by a linear combination of covariates, and as a consequence the mean is not constant anymore, but varies in space.

With KED the study variable at a prediction location \mathbf{s}_0 is predicted by

$$\hat{Z}_{\text{KED}}(\mathbf{s}_0) = \sum_{k=0}^p \hat{\beta}_k x_k(\mathbf{s}_0) + \sum_{i=1}^n \lambda_i \left[Z(\mathbf{s}_i) - \sum_{k=0}^p \hat{\beta}_k x_k(\mathbf{s}_i) \right], \tag{22.15}$$

with $\hat{\beta}_k$ the estimated regression coefficient. The first component of this predictor is the estimated expectation at the new location using the covariate values at this location and the estimated regression coefficients, and the second component is a weighted sum of the residuals at the sampling locations.

The optimal kriging weights $\lambda_i, i = 1 \dots n$ are obtained in a similar way as in ordinary kriging. The difference is that additional constraints on the weights are needed, to ensure unbiased predictions. Not only the weights must sum to one, but also for all p covariates the weighted sum of the covariate values

at the sampling locations must equal the covariate value at the target location ($\sum_{i=1}^n \lambda_i x_k(\mathbf{s}_i) = x_k(\mathbf{s}_0)$ for all $k = 1 \dots p$). This leads to a system of $n + p + 1$ simultaneous equations that must be solved. In matrix notation this system is

$$\begin{bmatrix} \mathbf{C} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{x}_0 \end{bmatrix}, \quad (22.16)$$

with

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \dots & & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}. \quad (22.17)$$

The submatrix \mathbf{O} is a $((p+1) \times (p+1))$ matrix with zeroes, ν a $(p+1)$ vector with Lagrange multipliers, and \mathbf{x}_0 a $(p+1)$ vector with covariate values at the prediction location (including a 1 as the first entry).

The kriging variance with KED equals

$$V_{\text{KED}}(\widehat{Z}(\mathbf{s}_0)) = \sigma^2 - \lambda^T \mathbf{c}_0 - \nu^T \mathbf{x}_0. \quad (22.18)$$

The prediction error variance with KED can also be written as the sum of the variance of the predictor of the mean and the variance of the error in the interpolated residuals (?):

$$V(\widehat{Z}(\mathbf{s}_0)) = \sigma^2 - \mathbf{c}_0^T \mathbf{C}^{-1} \mathbf{c}_0 + (\mathbf{x}_0 - \mathbf{X}^T \mathbf{C}^{-1} \mathbf{c}_0)^T (\mathbf{X}^T \mathbf{C}^{-1} \mathbf{X})^{-1} (\mathbf{x}_0 - \mathbf{X}^T \mathbf{C}^{-1} \mathbf{c}_0). \quad (22.19)$$

The first two terms constitute the interpolation error variance, the third term the variance of the predictor of the mean. Note that the computation is as Equation (22.18), with the λ and ν from the solution of the KED system; Equation (22.19) shows this in terms of linear modelling theory.

To illustrate that the kriging variance with KED depends on the values of the covariate at the sampling locations and prediction location, values of a covariate

x and of a correlated study variable z are simulated for the 50×50 units of a spatial population (Figure 22.3). First a field with covariate values is simulated. Then a field with residuals is simulated. The field of the study variable is then obtained by multiplying the simulated field with covariate values by two, adding a constant of 10, and finally adding the simulated field with residuals.

```
#simulate covariate values
vgmdl_x <- vgm(model="Sph", psill=10, range=200, nugget=0)
C <- variogramLine(vgmdl_x, dist_vector=H, covariance=TRUE)
Upper <- chol(C)
set.seed(314)
N <- rnorm(n=nrow(mypop), 0, 1)
mypop$x <- crossprod(Upper, N)+10
#simulate values for residuals
vgmdl_resi <- vgm(model="Sph", psill=5, range=100, nugget=0)
C <- variogramLine(vgmdl_resi, dist_vector=H, covariance=TRUE)
Upper <- chol(C)
set.seed(314)
N <- rnorm(n=nrow(mypop), 0, 1)
e <- crossprod(Upper, N)
#compute mean of study variable
betas <- c(10, 2)
mu <- betas[1] + betas[2]*mypop$x
#compute study variable z
mypop$z <- mu + e
```

As before, a centred square grid with a spacing of 100 distance units is selected. The simulated values of the study variable z and covariate x are used to predict z at a prediction location s_0 by kriging with an external drift (red cell in Figure 22.3). Although at the prediction location we have only one simulated value of covariate x , a series of covariate values is used to predict z at that location: $x_0 = 0, 2, 4, \dots, 20$. In practice we have of course only one value of the covariate at a fixed location, but this is for illustration purposes only. Note that we have only one data set with ‘observations’ of x and z at the sampling locations (square grid).

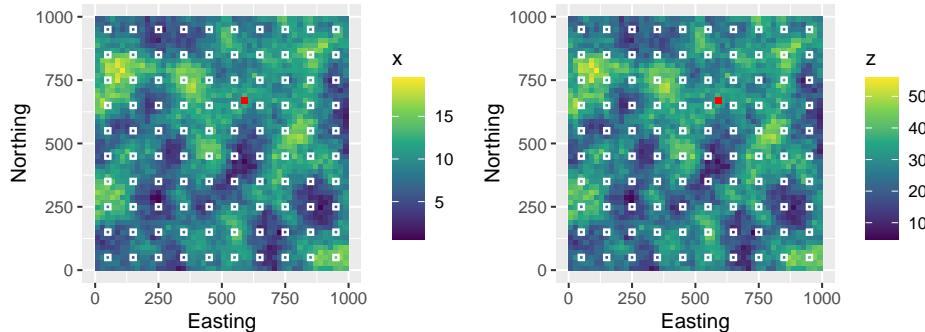


Figure 22.3: Maps with simulated values of covariate x and study variable z , centered square grid of sampling units, and prediction unit (red cell with coordinates (590,670)).

```

zxsim_sample <- over(mysample, mypop)
mysample$z <- zxsim_sample$z
mysample$x <- zxsim_sample$x
x0 <- seq(from=0, to=20, by=2)
v_zpred_KED <- NULL
for (i in 1:length(x0)) {
  s_0$x <- x0[i]
  predictions <- krige(
    formula=z~x,
    locations=mysample,
    newdata=s_0,
    model=vgmdl_resi,
    debug.level=0)
  v_zpred_KED[i] <- predictions$var1.var
}

```

Note the formula $z \sim x$ in the code chunk above, indicating that there is now an independent variable (covariate). The covariate values are attached to the file with the prediction location one-by-one in a for-loop. Also note that for KED we need the semivariogram of the residuals, not of the study variable itself. The residual semivariogram used in prediction is the same as the one used in simulating the fields: a spherical model without nugget, a sill of 5 and a range

of 100 distance units.

To assess the contribution of the uncertainty about the mean, I also predict the values assuming that the mean is known. In other words we assume that the two regression coefficients β_0 (intercept) and β_1 (slope) are known. This type of kriging is referred to as simple kriging (SK). With SK the constraints explained above are removed, so that there are no Lagrange multipliers involved. Argument `beta` is used to specify the known regression coefficients. I used the same values as in simulation.

```
v_zpred_SK <- NULL
for (i in 1:length(x0)) {
  s_0$x <- x0[i]
  prediction <- krige(
    formula=z~x,
    locations=mysample,
    newdata=s_0,
    model=vgmdl_resi,
    beta=betas,
    debug.level=0)
  v_zpred_SK[i] <- prediction$var1.var
}
```

Figure 22.4 shows that, contrary to the ordinary kriging variance, the kriging variance with KED is not constant, but depends on the covariate value at the prediction location. It is the smallest near the mean of the covariate values at the sampling sites, which is 10.0. The more extreme the covariate value at the prediction location, the larger the kriging variance with KED. This is analogous to the variance of predictions with a linear regression model.

The variance with simple kriging (SK) is constant. This is because with SK only the error in the interpolation of the residuals contribute to the variance (first two terms in Equation (22.19)). This interpolation error is independent of the value of x at the prediction location. In Figure 22.4 the difference between the variance with KED and with SK is the variance of the predictor of the mean, due to uncertainty about the regression coefficients. These regression coefficients must be *estimated* from the sample data.

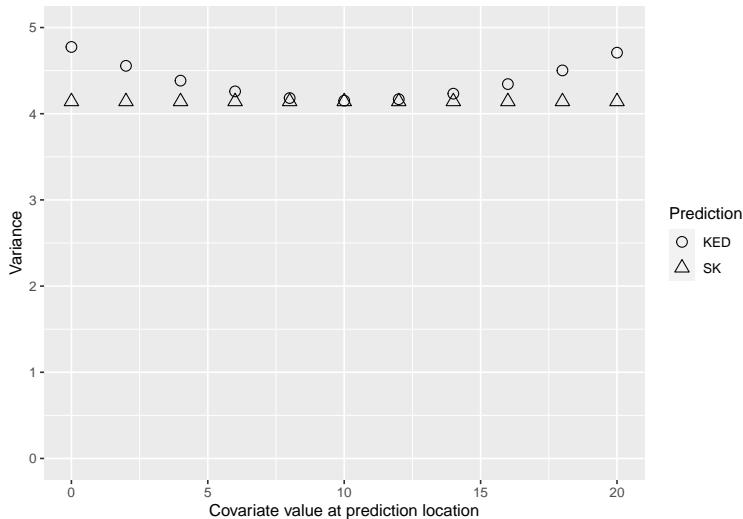


Figure 22.4: Variance of prediction error with kriging with an external drift (KED) and simple kriging (SK) as a function of the covariate value at a fixed prediction location.

22.3 Estimating the semivariogram

Kriging requires a semivariogram or covariance function as input for computing the covariance matrix of the study variable at the sampling locations and the vector of covariance of the study variable at the sampling locations and the prediction location. In most cases the semivariogram model is unknown and must be estimated from sample data; however it may be justified to use a semivariogram model developed in a similar study. Once the parameters of the semivariogram model are estimated, these are plugged in the kriging equations. There are two fundamentally different approaches for estimating the semivariogram parameters from sample data: the *method-of-moments* and *maximum likelihood estimation* (?).

22.3.1 Method-of-moments

With the method-of-moments approach the semivariogram is estimated in two steps. In the first step a sample semivariogram, also referred to as an experimen-

tal semivariogram, is estimated. This is done by choosing a series of distance intervals (bins)². For each distance interval all pairs of points with a separation distance in that interval are identified. For each pair half the squared difference of the study variable is computed, and the average of these squared differences over all point-pairs of that interval is computed. This average is the estimated semivariance of that distance interval. The estimated semivariances for all distance intervals are plotted against the average separation distances per distance interval. In the second step a permissible model is fitted to the sample semivariogram, using some form of weighted least squares. For details, see ?.

The next code chunk shows how this can be done in **R**. A simple random sample of 150 points is selected from the first simulated field shown in Figure 22.2. Function `variogram` of package `gstat` is then used to compute the sample semivariogram. Note that no distance intervals need be specified to function `variogram`. This can be done with argument `width` or argument `boundaries`. In their absence, the default value for the argument `width` is equal to the maximum separation distance divided by fifteen, so that there are fifteen points in the sample semivariogram. The maximum separation distance can be set with the argument `cutoff`. The default value for this argument is equal to one-third of the longest diagonal of the bounding box of the point set. The output is a data frame, with the number of point-pairs, average separation distance and estimated semivariance in the first three columns.

```
set.seed(123)
units <- sample.int(n=nrow(mypop), size=150)
mysample <- mypop[units,]
coordinates(mysample) <- ~s1+s2
vg <- variogram(z~1, data=mysample)
head(vg[,c(1,2,3)])
```

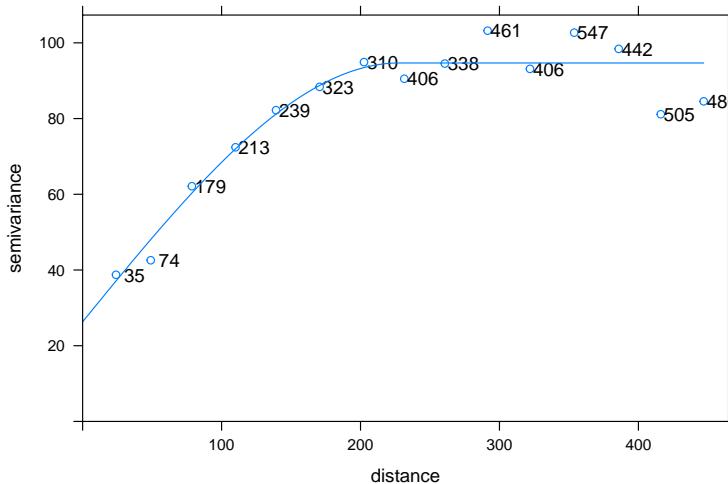
	np	dist	gamma
1	35	24.02379	38.73456
2	74	49.02384	42.57199
3	179	78.62401	62.13025
4	213	110.05088	72.41387
5	239	139.20163	82.21432

²If a semivariogram in different directions is required we must also choose direction intervals

```
6 323 170.55712 88.38168
```

The next step is to fit a model. This can be done with function `fit.variogram` of the `gstat` package. Many models can be fitted with this function (type `vgm()` to see all models). I chose a spherical model. Function `fit.variogram` requires initial values of the semivariogram parameters. From the sample semivariogram my eyeball estimates are 25 for the nugget, 250 for the range and 75 for the partial sill. These are specified to the `vgm` function, and used as the `model` argument to `fit.variogram`. The sample semivariogram and the fitted model can simply be plotted with function `plot`.

```
vgm_MoM <- fit.variogram(
  vg, model=vgm(model="Sph", psill=75, range=250, nugget=25))
plot(vg, vgm_MoM, plot.numbers=TRUE)
```



```
print(vgm_MoM)
```

model	psill	range
1 Nug	26.32048	0.0000
2 Sph	68.36364	227.7999

Note that `fit.variogram` has several options for weighted least squares optimisation, see `?fit.variogram` for details. Also, note that this non-linear fit may

not converge to a solution, especially if the starting values specified to `vgm` are not near their optimal values.

Further, this method depends on the choice of cutoff and distance intervals. We hope that changing these does not change the fitted model too much, but this is not always the case, especially with smaller data sets.

22.3.2 Maximum likelihood

In contrast to the method-of-moments, with the maximum likelihood method the data are not paired into couples and binned into a sample semivariogram. Instead the semivariogram model is estimated in one step. To apply this method one typically assumes that (possibly after transformation) the n sample data come from a multivariate normal distribution. If we have one observation from a normal distribution, the probability density of that observation is given by

$$f(z|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2\right\}, \quad (22.20)$$

with μ the mean and σ^2 the variance. With multiple independent observations, each of them coming from a normal distribution, the joint probability density is given by the product of the probability densities per observation. However, if the data are not independent we must account for the covariances, and the joint probability density can be computed with

$$f(\mathbf{z}|\mu, \theta) = (2\pi)^{-\frac{n}{2}} |\mathbf{C}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{z} - \mu)^T \mathbf{C}^{-1} (\mathbf{z} - \mu)\right\}, \quad (22.21)$$

where \mathbf{z} is the vector with the n sample data, μ is the vector with means, θ is the vector with parameters of the covariance function, and \mathbf{C} is the $n \times n$ matrix with variances and covariances of the sample data. If the probability density of Equation (22.21) is regarded as a function of μ and θ with the data \mathbf{z} fixed, this equation defines the likelihood.

Maximum likelihood estimates of the semivariogram can be obtained with function `likfit` of package **geoR** (?). First a `geoR` object must be made, specifying in which columns of the data frame the spatial coordinates and the study variable are.

Table 22.2: Estimates of semivariogram parameters obtained with method-of-moments and maximum likelihood.

Parameter	MoM	ML
nugget	26.3	19.5
partial sill	68.4	83.8
range	227.8	217.8

```
library(geoR)
mysample <- as(mysample, "data.frame")
dGeoR <- as.geodata(
  obj=mysample, header=TRUE,
  coords.col=1:2, data.col=3)
```

The model parameters can then be estimated with function `likfit`. The argument `trend="cte"` means that we assume that the mean is constant throughout the study area.

```
vgm_ML <- likfit(
  geodata=dGeoR, trend="cte",
  cov.model="spherical", ini.cov.pars=c(80,200),
  nugget=20, lik.method="ML", messages=FALSE)
```

Table 22.2 shows the maximum likelihood estimates together with the method-of-moments estimates.

As can be seen, the estimates are substantially different, especially the division of overall variance into partial sill and nugget.

22.4 Estimating the residual semivariogram

For KED estimates of the regression coefficients and of the parameters of the residual semivariogram are needed. Estimation of these model parameters is not a trivial problem, as the estimated regression coefficients and estimated residual semivariogram parameters are not independent. The residuals depend on the estimated regression coefficients, and as a consequence also the parameters of

the residual semivariogram depend on the estimated coefficients. Vice versa, the estimated regression coefficients depend on the spatial correlation of the residuals, and so on the estimated residual semivariogram parameters. This a classic “which came first, the chicken or the egg?” recursive problem.

Estimation of the model parameters is illustrated with the simulated field of Figure 22.3. A simple random sample of size 150 is selected to estimate the model parameters.

22.4.1 Iterative method-of-moments

A simple option is iterative estimation of the regression coefficients followed by method-of-moments estimation of the sample semivariogram and fitting of a semivariogram model. In the first iteration the regression coefficients are estimated by ordinary least squares (OLS). This implies that the data are assumed independent, i.e. we assume a pure nugget residual semivariogram. The sample semivariogram of the OLS residuals is then computed by the method-of-moments, followed by fitting a model to this sample semivariogram. With package **gstat** this can be done with one line of **R** code, using in function **variogram** as an argument a formula specifying the study variable and the predictors.

```
set.seed(314)
units <- sample.int(n=nrow(mypop), size=150)
mysample <- cbind(mypop[units,])
vg_resi <- variogram(z~x, data=mysample)
vgmdl <- vgm(model="Sph", psill=10, range=150, nugget=0)
vgmresi_MoM <- fit.variogram(vg_resi, model=vgmdl)
```

Given these estimates of the semivariogram parameters, the regression coefficients are re-estimated by accounting for spatial dependency of the residuals. This can be done by generalised least squares (GLS). Function **spDists** of package **sp** is used to compute the matrix with distances between the sampling locations, **variogramLine** of package **gstat** to transform the distance matrix into a covariance matrix. The operator **%*%** is for multiplying two matrices. Type **?<name of function>** for help about the functions **chol2inv**, **crossprod** and **solve**.

```
X <- matrix(data=1, nrow=nrow(as(mysample, "data.frame")), ncol=2)
X[,2] <- mysample$x
z <- mysample$z
D <- spDists(mysample)
C <- variogramLine(vgmresi_MoM, dist_vector=D, covariance=TRUE)
Cinv <- chol2inv(chol(C))
XC <- crossprod(X, Cinv)
XCz <- XC%*%z
XCX <- XC%*%X
betaGLS <- solve(XCX, XCz)
```

The GLS estimates of the regression coefficients can then be used to re-compute the residuals of the mean, and so on, until the changes in the model parameters are negligible.

```
repeat {
  betaGLS.cur <- betaGLS
  mu <- X%*%betaGLS
  mysample$e <- z-mu
  vg_resi <- variogram(e~1, data=mysample)
  vgmresi_MoM <- fit.variogram(vg_resi, model=vgmdl)
  C <- variogramLine(vgmresi_MoM, dist_vector=D, covariance=TRUE)
  Cinv <- chol2inv(chol(C))
  XC <- crossprod(X, Cinv)
  XCz <- XC%*%z
  XCX <- XC%*%X
  betaGLS <- solve(XCX, XCz)
  if (sum(abs(betaGLS-betaGLS.cur))<0.0001) {
    break
  }
}
```

Table 22.3 shows the estimates of the residual semivariogram parameters and regression coefficients, together with the estimates obtained by restricted maximum likelihood, which is explained in the next section.

22.4.2 Restricted maximum likelihood

The estimates of the residual semivariogram parameters obtained by the iterative method-of-moments procedure are not unbiased. When the mean is not constant but a linear combination of one or more covariates, also ML estimation results in biased estimates of the residual semivariogram parameters. Unbiased estimates of the regression coefficients and residual semivariogram parameters can be obtained by restricted maximum likelihood (also referred to as residual maximum likelihood), REML. In REML the vector with the data are premultiplied by a so-called projection matrix \mathbf{P} (?). This projection matrix has the property that a vector with zeroes is obtained when the matrix \mathbf{X} with the covariate values at the sampling locations (and ones in first column) is pre-multiplied with \mathbf{P} :

$$\mathbf{P}\mathbf{X} = \mathbf{0} . \quad (22.22)$$

Pre-multiplying both sides of the KED model (Equation (22.15)) with \mathbf{P} gives (?)

$$\mathbf{P}\mathbf{z}(\mathbf{s}) = \mathbf{y}(\mathbf{s}) = \mathbf{P}\mathbf{X}\beta + \mathbf{P}\epsilon(\mathbf{s}) = \mathbf{P}\epsilon(\mathbf{s}) . \quad (22.23)$$

In words, by pre-multiplying the variable \mathbf{z} with matrix \mathbf{P} a new variable \mathbf{y} is obtained that has a constant mean. So the trend is filtered out, whatever the regression coefficients are. The semivariogram parameters of this new variable can be estimated by ML. The projection matrix \mathbf{P} can be computed with

$$\mathbf{P} = \mathbf{I} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T , \quad (22.24)$$

with \mathbf{I} the $n \times n$ identity matrix (matrix with ones on the diagonal, zeroes in all off-diagonal elements). The natural logarithm of the residual likelihood can be computed with (?)

$$\ell(\theta|\mathbf{z}) = \text{constant} - 0.5(\ln|\mathbf{C}| + |\mathbf{X}^T\mathbf{C}^{-1}\mathbf{X}| + \mathbf{y}^T\mathbf{C}^{-1}(\mathbf{I} - \mathbf{Q})\mathbf{z})) , \quad (22.25)$$

with $\mathbf{Q} = \mathbf{X}(\mathbf{X}^T\mathbf{C}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{C}^{-1}$

Table 22.3: Estimates of residual spherical semivariogram parameters and regression coefficients obtained with iterative method-of-moments (iMoM) and restricted maximum likelihood (REML).

Parameter	iMoM	REML
nugget	0.000	0.000
partial sill	2.137	1.909
range	129.147	141.491
intercept	3.255	3.083
x	2.712	2.732

Restricted maximum likelihood estimates of the semivariogram can be obtained with function `likfit` of package `geoR`, used above in maximum likelihood estimation of the variogram (?).

```
vgm_REML <- likfit(
  geodata=dGeoR, trend=~x,
  cov.model="spherical", ini.cov.pars=c(2,100),
  nugget=0, lik.method="REML", messages=FALSE)
```

Table 22.3 shows that REML results in a smaller estimated (partial) sill and larger estimated range than iterative method-of-moments. Of the two regression coefficients especially the estimated intercept differs considerably among the two estimation methods.

I realise that this is a rather short introduction to kriging. I refer to ? for an introduction to geostatistics, to ? for an exposé of the many versions of kriging, and to ? for an elaborate explanation of kriging. A nice educational tool for getting a feeling for ordinary kriging is E{Z}-Kriging³⁴.

³https://wiki.52north.org/AI_GEOSTATS/SWEZKriging

⁴https://wiki.52north.org/AI_GEOSTATS/SWEZKriging

Chapter 23

Model-based optimisation of grid spacing

This is the first chapter on model-based sampling¹. In Chapters 18 and 19 a geometric criterion was minimised, i.e. a criterion defined in terms of distances, either in geographic space (Chapter 18), or in covariate space (Chapter 19). In model-based sampling the minimisation criterion is a function of the variance of the prediction errors.

This first chapter on model-based sampling is about optimisation of the spacing of a regular grid, i.e. the distance between neighbouring points in the grid. In case the budget for the survey may not exceed a given amount, and the total costs can be approximated by the costs per point multiplied by the sample size, this spacing can be derived from the affordable sample size (Chapter 17). The larger the affordable sample size, the smaller the grid spacing. The alternative is to derive the grid spacing from a requirement on the accuracy of the map. Here and in following chapters I assume that the map is constructed by kriging, see Chapter 22 for an introduction.

¹Spatial response surface sampling can also be considered as model-based sampling, especially when a model-based criterion is used, see Chapter @ref(SpatialResponseSurface).

23.1 Optimal grid spacing for ordinary kriging

Suppose that we require the mean kriging variance not to exceed a given threshold. The question then is what is the tolerable or maximum possible grid spacing given this requirement. For finding this tolerable grid spacing one must have prior knowledge of the spatial variation. I first consider the situation in which it is reasonable to assume that the mean of the study variable is constant throughout the study area, but unknown. With the mean unknown, this implies that ordinary kriging is used for mapping. Further, we need a semivariogram of the study variable. In practice we often do not have a reliable estimate of the semivariogram. In the best case scenario we have some existing data, of sufficient quantity and suitable spatial distribution, that can be used to estimate the semivariogram. In other cases such data are lacking, and a best guess of the semivariogram must be made, for instance using data for the same study variable from other, similar areas.

There is no simple equation that relates the grid spacing to the kriging variance. What can be done is to calculate the mean kriging variance for a range of grid spacings, plot the mean kriging variances against the grid spacing, and use this plot inversely to determine the tolerable grid spacing, given a constraint on the mean ordinary kriging variance.

In the next code chunks this procedure is used to compute the tolerable spacing of a square grid for mapping soil organic matter (SOM) in the three woredas of Ethiopia. The legacy data used before to design a spatial infill sample (Chapter 18) are used here to estimate a semivariogram. A sample semivariogram is estimated by the method-of-moments, and a spherical model is fitted using functions of package `gstat` (?). Note that the values for the partial sill, range and nugget in the `model` argument of the function `fit.variogram` are guesses from an eyeball examination of the sample semivariogram obtained with the function `variogram`, see Figure 23.1.

```
library(gstat)
load(file="data/CovariatesThreeWoredasEthiopia.RData")
load(file="data/ThreeWoredasEthiopia.RData")
vg <- variogram(SOM~1, data=priordataEthiopia)
vgmdl <- vgm(model="Sph", psill=0.6, range=40, nugget=0.6)
vgm_MoM <- fit.variogram(vg, model=vgmdl)
```

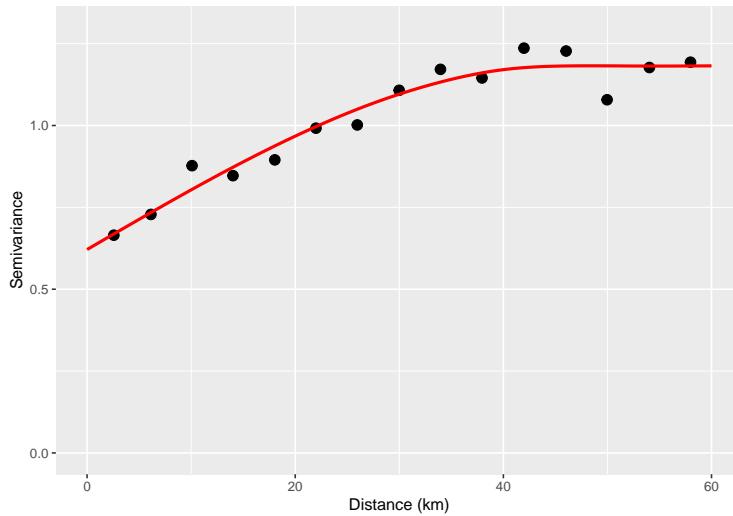


Figure 23.1: Sample semivariogram and fitted spherical model of SOM estimated from legacy data in three woredas of Ethiopia.

The semivariogram of SOM can also be estimated by maximum likelihood using function `likfit` of package `geoR` (?).

```
library(geoR)
priordata <- as(priordataEthiopia, "data.frame")
dGeoR <- as.geodata(
  obj=priordata, header=TRUE,
  coords.col=13:14, data.col=1)
vgm_ML <- likfit(
  geodata=dGeoR, trend="cte",
  cov.model="spherical", ini.cov.pars=c(0.4,40),
  nugget=0.6, lik.method="ML", messages=FALSE)
```

Table 23.1 shows the maximum likelihood (ML) estimates of the parameters of the spherical semivariogram, together with the method-of-moments estimates. Either could be used in the following steps.

Table 23.1: Method-of-moments and ML estimates of parameters of spherical semivariogram of SOM in Ethiopia.

Parameter	MoM	ML
nugget	0.62	0.56
partial sill	0.56	0.68
range	45.40	36.90

23.2 Controlling the mean or a quantile of the ordinary kriging variance

To decide on the grid spacing we may require that the population mean of the kriging variance (MKV) not exceeds a given threshold. Instead of the population mean, we may use the population median or any other quantile, for instance the 0.90 quantile (P90) of the kriging variance as a quality criterion. Hereafter the ML semivariogram is used to optimise the grid spacing given a requirement on the mean, median and P90 of the kriging variance.

As a first step a series of spacings of the square grid with observations are specified. Only spacings are considered which would result in expected sample sizes that are reasonable for kriging. With a spacing of 5 km the expected sample size is 434 points, with a spacing of 12 km these are 75 points.

`spacing <- 5:12`

The next step is to select a simple random sample of evaluation points. It is important to select a large sample, so that the precision of the estimated population mean or quantile of the kriging variance will be high. The standard error of the estimated MKV can be estimated, see Chapter 3, substituting the kriging variance at the sampling points for the study variable values.

```
load(file="data/CovariatesThreeWoredasEthiopia.RData")
set.seed(314)
units <- sample.int(nrow(grdEthiopia), size=5000, replace=TRUE)
mysample <- grdEthiopia[units,]
mysample$s1 <- jitter(mysample$s1, amount=0.5)
```

```
mysample$s2 <- jitter(mysample$s2, amount=0.5)
```

The **R** code below shows the next steps. Given a spacing, a square grid with a fixed starting point is selected with function **spsample**, using argument **offset**. A dummy variable is added to the data frame, having value 1 at all grid points. So the predicted value will everywhere be 1, since an unbiased predictor, being a weighted sum (as in OK) of any single value, must be that value. However, we are not interested in the predicted value, but in the kriging variance only, and we have seen in Chapter 22 that the kriging variance is independent of the observations. The maximum likelihood estimates of the semivariogram are used in function **vgm** of package **gstat** to define a semivariogram model of class **variogramModel** that can be handled by function **krige**. For each grid spacing the population mean, median and P90 of the kriging variance is estimated from the evaluation sample. The median and P90 can be computed with function **quantile** of the **stats** package.

```
coordinates(mysample) <- ~s1+s2
gridded(grdEthiopia) <- ~s1+s2
MKV_OK <- P50KV_OK <- P90KV_OK <- samplesize <-
  numeric(length=length(spacing))
vgm_ML_gstat <- vgm(
  model="Sph", nugget=vgm_ML$nugget, psill=vgm_ML$sigmasq,
  range=vgm_ML$phi)
for (i in 1:length(spacing)) {
  mygridxy <- spsample(
    x=grdEthiopia, cellsize=spacing[i],
    type="regular", offset=c(0.5,0.5))
  mygrid<-data.frame(s1=mygridxy$x1, s2=mygridxy$x2, dummy=1)
  samplesize[i] <- nrow(mygrid)
  coordinates(mygrid) <- ~s1+s2
  predictions <- krige(
    formula=dummy~1,
    locations=mygrid,
    newdata=mysample,
    model=vgm_ML_gstat,
    nmax=100,
    debug.level=0)
```

```

MKV_OK[i] <- mean(predictions$var1.var)
P50KV_OK[i] <- quantile(predictions$var1.var, probs=0.5)
P90KV_OK[i] <- quantile(predictions$var1.var, probs=0.9)
}
dfKV_OK <- data.frame(
  spacing, samplesize, MKV_OK, P50KV_OK, P90KV_OK)

```

The estimated mean and quantiles of the kriging variance are plotted against the grid spacing.

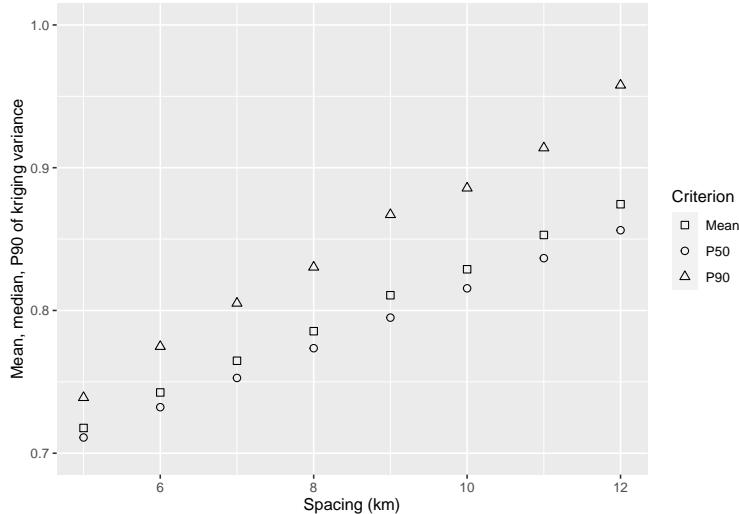


Figure 23.2: Mean, median (P50) and 0.90 quantile (P90) of the ordinary kriging variance of predictions of SOM in three woredas of Ethiopia, as a function of the spacing of a square grid.

The tolerable grid spacing for the three quality indices can be computed with function `approx` of the `base` package, as shown below for the median kriging variance.

```

spacing_tol_P50 <- approx(
  x=dfKV_OK$P50, y=dfKV_OK$spacing, xout=0.8)$y

```

For a mean kriging variance of 0.8 the tolerable grid spacing is 8.6 km, for the median kriging variance this is 9.2, which is somewhat larger, leading to a smaller sample size. The smaller grid spacing for the mean can be explained by the right-skewed distribution of the kriging variance, so that the mean is larger than the median. For the 0.90 quantile of the kriging variance the tolerable grid spacing is much smaller: 6.8, leading to a much larger sample size.

Exercises

1. Write an **R** script to determine the tolerable grid spacing so that the 0.50, 0.80 and 0.95 quantiles of the variance of ordinary kriging predictions of SOM in the three woredas of Ethiopia do not exceed 0.8. Estimate the semivariogram by the method-of-moments.
2. In practice we are uncertain about the semivariogram. For that reason it can be wise to explore the sensitivity of the tolerable grid spacing for the semivariogram parameters. Increase the nugget parameter of the method-of-moments semivariogram by 5%, and change the partial sill parameter so that the sill (nugget + partial sill) is unchanged. Compute the mean kriging variance for grid spacings of 5, 6, ..., 12 km.
 - Do the same by reducing the range of the method-of-moments semivariogram by 5%. Reset the nugget and partial sill to their original values.
 - Plot the mean kriging variance with the original semivariogram, semivariogram with increased nugget and semivariogram with smaller range against the grid spacing. Explain the difference in tolerable grid spacings obtained with the three semivariograms.
 - Compute the tolerable grid spacings for the three semivariograms for a mean kriging variance of 0.85, and the corresponding expected sample sizes.

23.3 Optimal grid spacing for block-kriging

In the previous section the tolerable grid spacing is derived from a constraint on the mean, P50 or P90 of the error variances of predictions at points. The alternative is to put a constraint on the mean (P50, P90) of the error variances

of the predicted means of blocks. These means can be predicted with block-kriging (Section 22.1). Block-kriging predictions can be obtained with function `krige` of package `gstat`, using argument `block`. In the code chunk below the means of 100 m × 100 m blocks are predicted by ordinary block-kriging.

```
MKV_OBK <- P50KV_OBK <- P90KV_OBK <-
  numeric(length=length(spacing))
for (i in 1:length(spacing)) {
  mygridxy <- spsample(
    x=grdEthiopia, cellsize=spacing[i],
    type="regular", offset=c(0.5,0.5))
  mygrid<-data.frame(s1=mygridxy$x1, s2=mygridxy$x2, dummy=1)
  samplesize[i] <- nrow(mygrid)
  coordinates(mygrid) <- ~s1+s2
  predictions <- krige(
    formula=dummy~1,
    locations=mygrid,
    newdata=mysample,
    model=vgm_ML_gstat,
    block=c(0.1,0.1),
    nmax=100,
    debug.level=0)
  MKV_OBK[i] <- mean(predictions$var1.var)
  P50KV_OBK[i] <- quantile(predictions$var1.var,probs=0.5)
  P90KV_OBK[i] <- quantile(predictions$var1.var,probs=0.9)
}
dfKV_OBK <- data.frame(spacing, MKV_OBK, P50KV_OBK, P90KV_OBK)
```

Figure 23.3 shows that the mean, P50 and P90 of the block-kriging predictions are substantially smaller than those of the point-kriging predictions (Figure 23.2). This can be explained by the large nugget of the semivariogram (Table 23.1). The side length of a prediction block (100 m) is much smaller than the range of the variogram (36.9 km), so that in this case the mean semivariance within a prediction block is about equal to the nugget. Roughly speaking, given a grid spacing the mean point-kriging variance is reduced by an amount about equal to this means semivariance, to yield the mean block-kriging variance for this spacing (Section 22.1). Recall that the mean semivariance within a block is a model-based prediction of the variance within a block (Equation (13.3)).

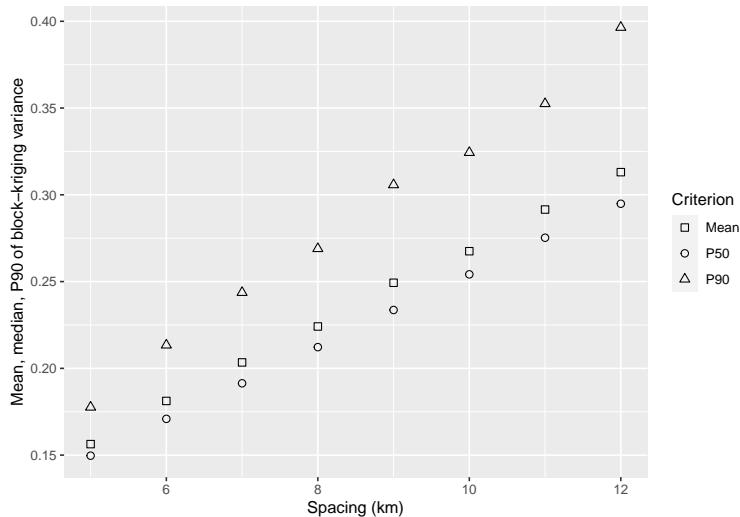


Figure 23.3: Mean, median (P50) and 0.90 quantile (P90) of the ordinary block-kriging variance of predictions of the mean SOM of blocks of 100 m by 100 m, in three woredas of Ethiopia, as a function of the spacing of a square grid.

23.4 Optimal grid spacing for kriging with an external drift

In the previous section I assumed a constant mean for the study variable. I now consider the case where covariates that are related to the study variable are available. A model is calibrated that is the sum of a linear combination of the covariates (spatial trend) and a spatially structured residual, see Equation (22.15). Predictions at the nodes of a fine grid are obtained by kriging with an external drift (KED).

The SOM data of the three woredas of Ethiopia are used to estimate the parameters (regression coefficients and residual semivariogram parameters) of the model by restricted maximum likelihood (REML), see Section 22.4.2.

Table 23.2: Maximum likelihood estimates of parameters of spherical semivariogram of SOM, and restricted maximum likelihood estimates of residual semivariogram of SOM in Ethiopia.

Parameter	ML	REML
nugget	0.56	0.36
partial sill	0.68	0.44
range	36.90	5.20

```
library(geoR)
dGeoR <- as.geodata(
  obj = priodata, header=TRUE,
  coords.col=13:14, data.col=1,
  covar.col=c(3,9,10,11))
vgm_REML <- likfit(
  geodata=dGeoR, trend=~dem+rfl_NIR+rfl_red+lst,
  cov.model="spherical", ini.cov.pars=c(1,5),
  nugget=0.2, lik.method="REML", messages=FALSE)
```

The total sill (partial sill + nugget) of the residual semivariogram equals 0.8, which is considerably smaller than that of the ML semivariogram of SOM (Table 23.2). A considerable part of the variance of SOM is explained by the covariates. Note the much smaller range of the residual semivariogram. So the spatial structure of SOM is largely captured by the covariates. The residuals of the mean, which is a linear combination of the covariates, do not show much spatial structure anymore.

The mean kriging variance as obtained with KED is used as the evaluation criterion. With KED the kriging variance is also a function of the values of the covariates at the sampling locations and prediction location (Section 22.2). Compared to the procedure above for ordinary kriging, in the code chunk below a slightly different procedure is used. The square grid of a given spacing is randomly placed on the area (option `offset` in function `spsample` is not used), and this is repeated several times.

```

r <- 10
MKV_KED <- matrix(nrow=length(spacing), ncol=r)
vgm_REML_gstat <- vgm(
  model="Sph", nugget=vgm_REML$nugget,
  psill=vgm_REML$sigmasq, range=vgm_REML$phi)
set.seed(314)
for (i in 1:length(spacing)) {
  for (j in 1:r) {
    mygridxy <- spsample(x=grdEthiopia, cellsize=spacing[i],
                           type="regular")
    mygrid <- data.frame(s1=mygridxy$x1, s2=mygridxy$x2, dummy=1)
    coordinates(mygrid) <- ~s1+s2
    mygrd <- data.frame(over(mygrid, grdEthiopia), mygrid)
    coordinates(mygrd) <- ~s1+s2
    predictions <- krige(
      formula=dummy~dem+rfl_NIR+rfl_red+lst,
      locations=mygrd,
      newdata=mysample,
      model=vgm_REML_gstat,
      nmax=100,
      debug.level=0)
    MKV_KED[i,j] <- mean(predictions$var1.var)
  }
}
dfKV_KED <- data.frame(spacing,MKV_KED)

```

Figure 23.4 shows the mean kriging variances obtained with kriging with an external drift and with ordinary kriging, as a function of the grid spacing.

Interestingly for grid spacings smaller than about nine km, the mean kriging variance with KED is larger than with OK. So in this case only for larger grid spacings KED outperforms OK in terms of the mean kriging variance, and only for a mean kriging variances larger than about 0.8, we can afford with KED a larger grid spacing (smaller sample size) than with OK.

```

MMKV_KED <- apply(dfKV_KED[,-1], MARGIN=1, FUN=mean)
spacing_tol_KED <- approx(x=MMKV_KED, y=dfKV_KED$spacing, xout=0.8)$y

```

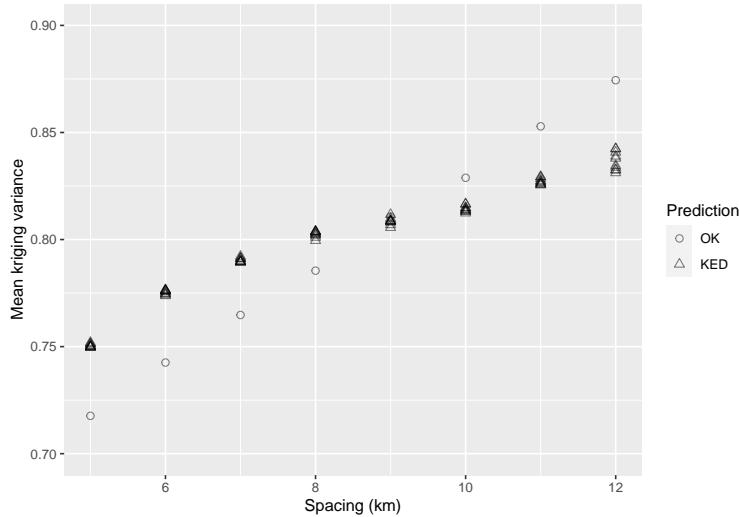


Figure 23.4: Mean kriging variance of SOM predictions with KED and OK, in three woredas of Ethiopia, as a function of the spacing of a square grid. With KED for each spacing ten MKV values are shown corresponding with ten randomly placed grids.

The tolerable grid spacing for a mean kriging variance of 0.8, using KED, equals 7.9 km.

Exercises

3. Given a grid spacing the mean kriging variance varies among randomly selected grids, especially for large spacings. Explain why.
4. Write an **R** script to compute the tolerable grid spacing for kriging with an external drift of natural logarithms of the electrical conductivity of the soil (ECe150) in the Cotton Research Farm of Uzbekistan, using natural logarithms of EMv100cm measurements as a covariate. Use the estimated parameters of an exponential semivariogram of the residuals, as shown in Table 24.1. Select a simple random sample of size 1,000 of evaluation points from the discretisation grid with interpolated lnEMv1m values (available in file `data/CottonResearchFarm.RData`) to compute the mean

kriging variance. Do this by selecting 100 grid cells by simple random sampling with replacement, and jittering the centers of the selected grid cells by an amount equal to half the size of the grid cell. Use as grid spacings 70, 75, ..., 100 m. With a spacing of 100 m the number of grid points is about 100 (the farm has an area of about 97 ha). What is the tolerable grid spacing for a mean kriging variance of lnECE150 of 0.1605?

23.5 Bayesian approach

In practice we do not know the semivariogram. In the best case we have prior data that can be used to estimate the semivariogram. However, even in this case we are uncertain about the semivariogram type (spherical, exponential, etc.) and the semivariogram parameters. ? showed how in a Bayesian approach we can account for uncertainty about the semivariogram parameters when we must decide on the grid spacing. In this approach a prior distribution of the semivariogram parameters is updated with the sample data to a posterior distribution (?):

$$f(\theta|\mathbf{z}) = \frac{f(\theta)f(\mathbf{z}|\theta)}{f(\mathbf{z})}, \quad (23.1)$$

with $f(\theta|\mathbf{z})$ the posterior distribution function, i.e. the probability density function of the semivariogram parameters given the sample data, $f(\theta)$ our prior belief in the parameters specified by a probability density function, $f(\mathbf{z}|\theta)$ the likelihood of the data, and $f(\mathbf{z})$ the probability density function of the data. This probability density function $f(\mathbf{z})$ is hard to obtain.

Problems with analytical derivation of the posterior distribution are avoided by selecting a large sample of units (vectors with semivariogram parameters) from the posterior distribution through Markov chain Monte Carlo (MCMC) sampling, see Section 13.1.3.

In a Bayesian approach we must define the likelihood function of the data, see Section 22.3.2. I assume that the SOM data in the Ethiopia case study have a multivariate normal distribution, and that the spatial covariance of the data can be modelled by a spherical function, see Section 22.3.2. The likelihood function is already defined in Section 13.1.3. The likelihood is a function of the semivariogram parameters. Given a vector of semivariogram parameters, the variance-covariance matrix of the data is computed from the matrix with

geographic distances between the sampling points. Function `spDists` of package `sp` is used to compute this distance matrix.

```
D <- spDists(priordataEthiopia)
```

Besides the likelihood function, in a Bayesian approach we must define prior distributions for the semivariogram parameters. Here we combine the partial sill and nugget into *ratio of spatial dependence*, i.e. the proportion of the total sill attributable to the partial sill. This terminology is because the proportion attributable to the nugget has no spatial structure. For the ratio of spatial dependence ξ and the distance parameter ϕ I chose uniform distributions as priors, with lower bounds equal to 0 and 10^{-6} , respectively, and upper bounds of one for the ratio of spatial dependence and 100 km for the range. A uniform distribution for the sill is not recommended (?). Instead, I assume a uniform distribution for the *inverse* of the sill, with a lower bound of 10^{-6} wt%², and an upper bound of 2 wt%².

These priors can be defined by function `createUniformPrior` of package `BayesianTools` (?). There are also functions to define a beta density function (commonly used as a prior for proportions) and a truncated normal distribution as a prior. The function `createBayesianSetup` is then used to define the setup of the MCMC sampling, specifying the likelihood function, the prior, and the vector with best prior estimates of the model parameters, specified with argument `best`. The maximum likelihood estimates computed in Section 23.1 are used as starting values for the inverse of the sill parameter, the ratio of spatial dependence, and the range.

```
library(BayesianTools)
priors <- createUniformPrior(
  lower = c(1E-6, 0, 1E-6), upper = c(2, 1, 100))
sill_ML <- vgm_ML$psill[1]+vgm_ML$psill[2]
thetas_ML <- c(1/sill_ML, vgm_ML$psill[2]/sill_ML, vgm_ML$range[2])
model <- "Sph"
setup <- createBayesianSetup(
  likelihood=ll, prior=priors,
  best=thetas_ML, names=c("lambda", "xi", "range"))
```

A sample from the posterior distribution of the semivariogram parameters is

Table 23.3: First ten units of MCMC sample from posterior distribution of parameters of spherical semivariogram for SOM.

Inverse of sill	Ratio of spatial dependence	Range
0.624	0.655	55.0
0.828	0.588	33.3
0.769	0.480	51.9
0.624	0.655	55.0
0.872	0.538	47.2
0.559	0.643	60.6
0.605	0.690	66.8
0.764	0.585	42.7
0.793	0.608	44.8
0.480	0.741	87.1

then obtained with function `runMCMC`. Various sampling algorithms are implemented in package **BayesianTools**. The algorithm can be specified with argument `sampler`. I used the default sampler `DEzs`, which is based on differential evolution Markov chain (?). It is common not to use all sampled units, but to discard the units of the burn-in period that are possibly influenced by the initial arbitrary settings, and to thin the series of units after this period. The extraction of the ultimate sample is done with function `getSample`. Argument `start` specifies the unit where the extraction starts, and argument `numSamples` specifies how many units are selected through systematic sampling of the full MCMC sample. The alternative is to use argument `thin` which defines the thinning interval.

```
set.seed(314)
res <- runMCMC(setup, sampler="DEzs")
mcmcsample <- getSample(res, start=1000, numSamples=1000) %>%
  as.data.frame()
```

Table 23.3 shows the first ten units of the MCMC sample from the posterior distribution of the semivariogram parameters.

The units of the MCMC sample (vectors with semivariogram parameters) are used one-by-one to compute the average of the kriging variances at the simple

random sample of evaluation points.

For each unit in the MCMC sample the tolerable grid spacing is computed for a target MKV of 0.8. Figure 23.5 shows that the grid spacing with the largest number of MCMC samples equals 8 km, which corresponds with the tolerable grid spacing derived above for ordinary kriging. For 165 units in the MCMC sample the tolerable grid spacing exceeds 12 km. However this grid spacing leads to a sample size that is too small for estimating the semivariogram and kriging.

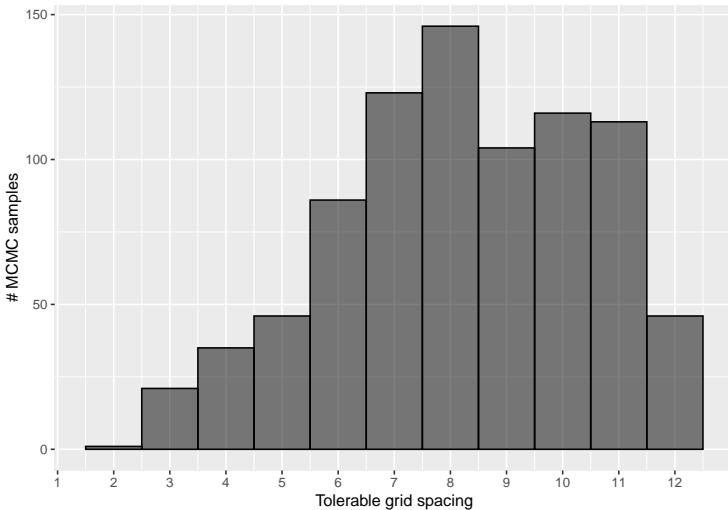


Figure 23.5: Histogram of tolerable grid spacings for a targeted MKV of 0.8.

Finally, for each grid spacing, the proportion of MCMC samples with a MKV smaller or equal to the target MKV of 0.8 is computed. Figure 23.6 shows, for instance, that if we require a probability of 80% that the MKV does not exceed the target MKV of 0.8, the tolerable grid spacing is about 6.25 km. With a grid spacing of 8 km as determined before, the probability that the MKV exceeds 0.8 is about 57%.

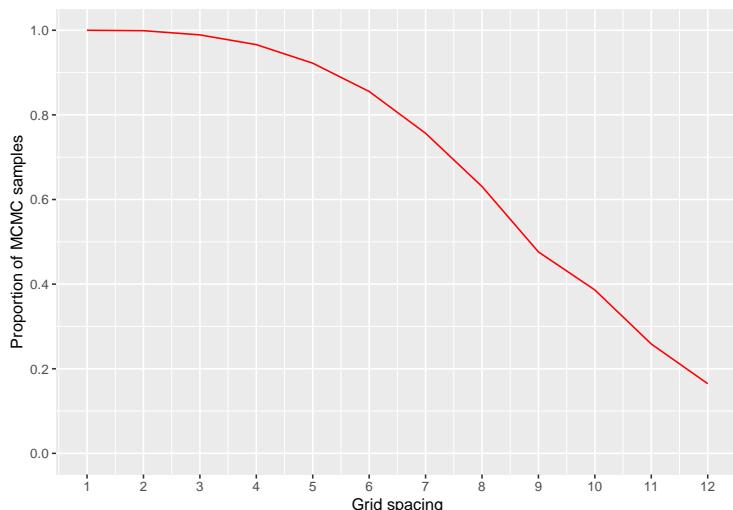


Figure 23.6: Proportion of MCMC samples with a MKV smaller than or equal to a target MKV of 0.8.

Chapter 24

Model-based optimisation of sampling pattern

In Chapter 23 a model of the spatial variation was used to optimise the spacing of a regular grid. The grid spacing determines the number of grid points within the study area, so optimisation of the grid spacing is equivalent to optimisation of the sample size.

This chapter is about optimisation of the spatial coordinates of the sampling units *given the sample size*. So we are searching for the optimal locations of a fixed number of sampling units. The constraint of sampling on a regular grid is dropped. In general, the optimal spatial pattern is irregular. Similar to spatial coverage sampling (Chapter 18), we search for the optimal sampling locations through minimisation of an explicit criterion. In spatial coverage sampling the minimisation criterion is the mean squared shortest distance (MSSD), which was minimised by k -means. In this chapter the minimisation criterion is the mean kriging variance (MKV). k -means cannot be used for minimising MKV as it uses (standardised) distances between cluster centers (the sampling locations) and the nodes of a discretisation grid, and the kriging variance is not a simple linear function of these distances. The kriging system also takes into account the separation between sampling points. A different optimisation algorithm is needed. Here spatial simulated annealing (SSA) is used. This is an optimisation approach that mimics the gradual cooling of metal alloys, resulting in an

optimum or near-optimum structure of the atoms in the alloy.

24.1 Spatial simulated annealing

Inspired by the potentials of optimisation through simulated annealing (?), ? proposed to optimise the sampling locations by spatial simulated annealing (SSA), see also ? and ?. This is an iterative, random search procedure, in which a sequence of samples is generated. A new sample (proposed sample) is obtained by slightly modifying the current sample. One sampling location of the current sample is randomly selected, and this location is shifted to a random location within the neighbourhood of the selected location.

The minimisation criterion is computed for the proposed sample and compared wth that of the current sample. If the criterion of the proposed sample is smaller, it is accepted. If the criterion is larger, the proposed sample is accepted with a probability equal to

$$P = e^{-\frac{\Delta}{T}} , \quad (24.1)$$

with Δ the increase of the criterion, and T the “temperature” which is one of the annealing schedule parameters¹. The larger the value of T , the larger the probability that a proposed sample with a given increase of the criterion is accepted (Figure 24.1). The temperature T is stepwise decreased during the optimisation: $T_{k+1} = \alpha T_k$. In Figure 24.1 α equals 0.9. The effect of decreasing the temperature is that the acceptance probability of worse samples decreases during the optimisation and approaches 0 towards the end of the optimisation. Note that the temperature remains constant during a number of iterations, referred to as the chain length. In Figure 24.1 this chain length equals 100 iterations. Finally, a stopping criterion is required. Various stopping criteria are possible; one option is to set the maximum numbers of chains with no improvement.

¹The name of this parameter shows the link with annealing in metallurgy. Annealing is a heat treatment of a material above its recrystallisation temperature.

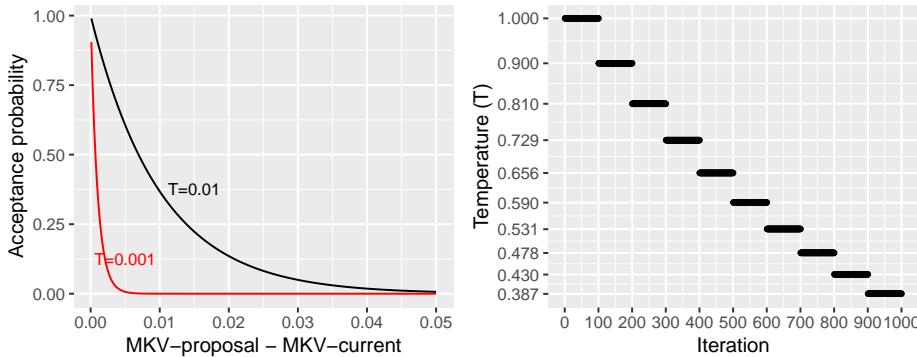


Figure 24.1: Acceptance probability as a function of the change in the mean kriging variance (MKV) which is used as a minimisation criterion, and cooling schedule in spatial simulated annealing. For negative changes (MKV of proposed sample smaller than of current sample) the acceptance probability equals 1.

24.2 Optimising the sample pattern for ordinary kriging

In ordinary kriging we assume a constant mean. No covariates are available that are related to the study variable. Optimisation of the sampling locations for ordinary kriging is illustrated with the Cotton Research Farm in Uzbekistan, which was used before to illustrate spatial response surface sampling (Chapter 21). The spatial coordinates of 50 sampling locations are optimised for mapping of the electrical conductivity (ECe) of the soil by ordinary kriging. In this section the coordinates of the sampling points are optimised for ordinary kriging. In Section 24.3 hereafter this is done for kriging with an external drift, and a map of electromagnetic induction (EM) is used to further optimise the coordinates of the sampling points.

Model-based optimisation of the sampling locations for ordinary kriging requires as input a semivariogram of the study variable. For the Cotton Research Farm I used the ECe data collected in eight surveys in the years 2008 - 2011 at 142 points to estimate this semivariogram. The ECe data are natural log transformed. The sample semivariogram is shown in Figure 24.2. The R code below shows how I fitted the semivariogram model with function `nls` (“non-linear least squares”) of the `stat` package. I did not use function `fit.variogram` of the `gstat` package,

because this function requires the output of the function `variogram` as input, whereas the sample semivariogram is here computed in a different way². The semivariogram parameters as estimated by `nls` are then used to define a semivariogram model of class `variogramModel` of package `gstat`, using function `vgm`. This is done because the function `optimMKV` requires a semivariogram model of this class, see hereafter. As already mentioned before in Chapter 23, in practice we often do not have legacy data from which we can estimate the semivariogram, and a best guess of the semivariogram then must be made.

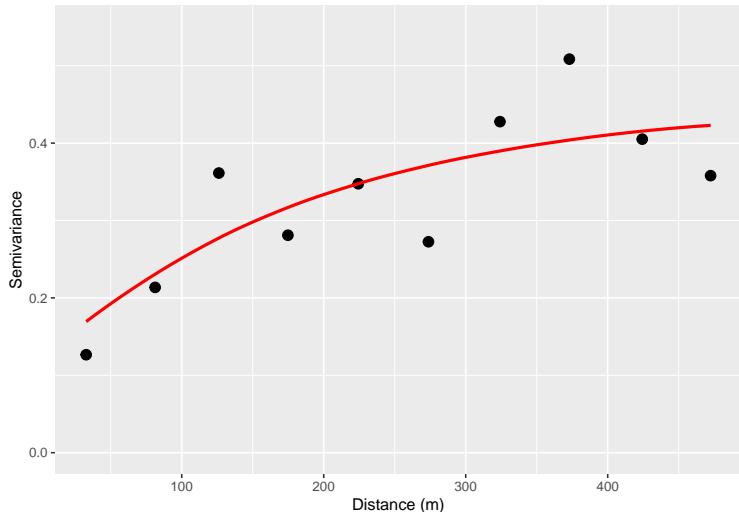


Figure 24.2: Sample semivariogram and fitted exponential model of natural logarithms of soil salinity (ECe) for the Cotton Research Farm in Uzbekistan.

```
library(gstat)
res <- nls(
  semivar~nugget+psill*(1-exp(-h/range)),
  start=list(nugget=0.1, psill=0.4, range=200),
  weights=somnp)
vgm_lnECe <- vgm(
```

²The sample semivariogram is computed by first estimating sample semivariograms for each of the eight surveys separately, followed by computing weighted averages of semivariances and distances per lag, using the numbers of pairs as weights.

```
model="Exp", nugget=coef(res)[1],
psill=coef(res)[2], range=coef(res)[3])
```

The estimated semivariogram parameters are shown in Table 24.1. The nugget-to-sill ratio is about 1/4, and the effective range is about 575 m (three times the distance parameter of an exponential model).

The coordinates of the sampling points are optimised with function `optimMKV` of package `spsann` (?). First, the candidate sampling points are specified by the nodes of a grid discretising the population. As explained hereafter, this does not necessarily imply that the population is treated as a finite population. Next, the parameters of the annealing schedule are set. Note that both the initial acceptance rate and the initial temperature are set, which may seem weird as the acceptance rate is a function of the initial temperature, see Equation (24.1). The initial acceptance rate is used as a threshold value. If an initial temperature is chosen that leads to an acceptance rate smaller than the chosen value for the initial acceptance rate, then the optimisation stops. In this case a larger value for the initial temperature must be chosen. The arguments `chain.length` and `stopping` of function `scheduleSPSANN` are multipliers. So for a chain length of five, the number of iterations equals $5 n$, with n the sample size. During the optimisation a sample is perturbed by replacing one randomly selected point of the current sample by a new point. This is done by two-stage cluster sampling. In the first stage one node of the selected sampling grid (specified with argument `candi`) is randomly selected. Only the nodes within a neighbourhood defined by `x.min`, `x.max`, `y.min` and `y.max` can be selected. The nodes within this neighbourhood have equal probability of being selected. In the second stage one point is selected within a grid cell with the selected node at its center, and a side length specified by argument `cellsize`. So it is natural to set `cellsize` to the spacing of the sampling grid. Only with `cellsize=0` the population is finite, and the sampling points are restricted to the nodes of the sampling grid.

```
library(spsann)
load(file="data/CottonResearchFarm.RData")
candi <- EM_CRF[,c(1,2)]
names(candi) <- c("x", "y")
schedule <- scheduleSPSANN(
  initial.acceptance=0.8,
```

```
initial.temperature=0.002, temperature.decrease=0.95,
chains=500, chain.length=5, stopping=20,
x.min=0, y.min=0, cellsize=25)
```

The R code for optimising the coordinates of the sampling points is as follows.

```
set.seed(314)
rslt <- optimMKV(
  points=50, candi=candi,
  vgm=vgm_lnECe, eqn=z~1,
  schedule=schedule,
  plotit=FALSE, track=TRUE)
mysample <- candi[rslt$points$id,]
trace <- rslt$objective$energy
```

The spatial pattern of the sample in Figure 24.3 and the trace of the MKV in Figure 24.4 suggest that we are close to the global optimum.

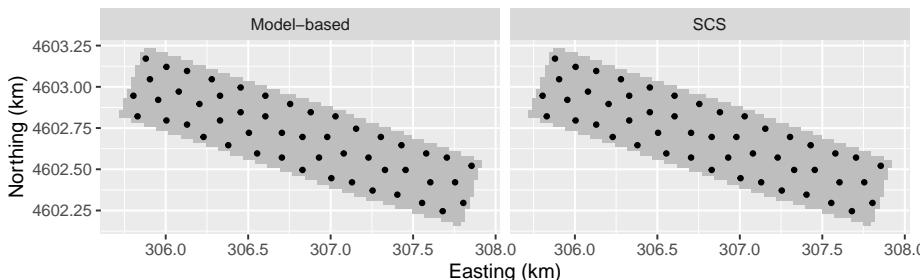


Figure 24.3: Sample optimised with the mean variance of ordinary kriging predictions of lnECe (Model-based), and spatial coverage sample (SCS) from the Cotton Research Farm in Uzbekistan.

For comparison I also computed a spatial coverage sample of the same size. The spatial patterns of the two samples are quite similar (Figure 24.3). The MKV of the spatial coverage sample equals 0.2633, whereas for the model-based sample this MKV equals 0.2626. So a small gain in precision is achieved only by the model-based optimisation of the sampling locations compared to spatial coverage sampling. This result is in agreement with the results reported by ?.

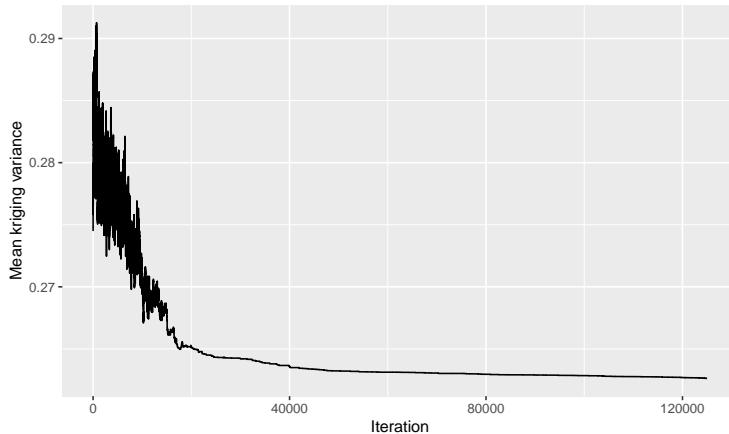


Figure 24.4: Trace of mean ordinary kriging variance.

Instead of the mean of the ordinary kriging variance (MOKV), we may prefer to use some quantile of the frequency distribution of the ordinary kriging variance as a minimisation criterion. For instance, if we use the 0.90 quantile as criterion, we are searching for the sampling locations so that the 90th percentile (P90) of the ordinary kriging variance is minimal. This can be done with function `optimUSER` of package `spsann` (?). This function has an argument `fun` that can be used to specify the objective function to be minimised. In this case the objective function is as follows.

```
QOKV <- function(points, esample, model, nmax, prob) {
  points <- as.data.frame(points)
  coordinates(points) <- ~x+y
  points$dum=1
  res <- krige(
    formula=dum~1,
    locations=points,
    newdata=esample,
    model=model,
    nmax=nmax,
    debug.level=0)
  quantile(res$var1.var, probs=prob)
```

}

The next code chunk shows how this objective function can be minimised.

```
myevalsample <- candi
coordinates(myevalsample) <- ~x+y
set.seed(314)
rslt <- optimUSER(
  points=50, candi=candi,
  fun=QOKV,
  esample=myevalsample,
  model=vgm_lnECe,
  nmax=20, prob=0.9,
  schedule=schedule)
```

Argument `esample` specifies a `SpatialPoints` object with the evaluation points, i.e. the points at which the kriging variance is computed. Here I used all candidate sampling points as evaluation points. Computing time can be reduced by selecting a coarser square grid with evaluation points, see Chapter 17. Argument `nmax` specifies the number of points used in kriging, and `prob` is the cumulative probability of the kriging variance quantile. As we will see in the next exercise, the nugget of the residual semivariogram has a strong effect on the optimised sample pattern, stressing the importance of a reliable prior estimate this semivariogram parameter.

Exercises

1. Write an **R** script to optimise the coordinates of sixteen points in a square for ordinary kriging. First create a discretisation grid of 20×20 nodes. Use an exponential semivariogram without nugget, with a sill of 2 and a distance parameter of four times the spacing of the discretisation grid. Optimise the locations with spatial simulated annealing (using functions `scheduleSPSANN` and `optimMKV` of package `spsann`).
 - Check whether the optimisation has converged by plotting the trace of the optimisation criterion MKV.
 - Based on the coordinates of the sampling points, do you think this is the global optimum, i.e. the sample with the smallest possible MKV?

2. Write an **R** script to optimise the coordinates of 50 points, using the P90 of the variance of ordinary kriging predictions of lnECe on the Cotton Research Farm as a minimisation criterion. Use the semivariogram parameters of Table 24.1.
 - Compare the optimised sample with the sample optimised with the mean ordinary kriging variance (shown in Figure 24.3).

24.3 Optimising the sample pattern for kriging with an external drift

If we have one or more covariates that are linearly related to the study variable, the study variable can be mapped by kriging with an external drift (KED). A requirement is that we have maps of the covariate so that, once we have estimated the parameters of the model for KED from the data collected at the optimised sample, these covariate maps can be used to map the study variable.

Optimisation of the sampling locations for KED requires as input the semivariogram of the residuals. Besides, we must decide on the covariates for the mean. Note that we do not need as input estimates of the regression coefficients associated with the covariates, just which combination of covariates we want to use for modelling the mean of the study variable, and as said before, the values of these covariates at the prediction locations.

Optimisation of the sampling locations for KED is illustrated with the Cotton Research Farm. The interpolated natural logarithm of the EM data is used as a covariate, see Figure 21.1. The data for fitting the model are in data file `clbdat`. The parameters of the residual semivariogram are estimated by REML, see Section 22.4.2.

At several points multiple observations have been made; these data have exactly the same spatial coordinates. This leads to problems with REML estimation (covariance matrix not positive definite so that it cannot be inverted). To solve this problem I jittered the coordinates of the sampling points by a very small amount.

```
library(geoR)
clbdat$lnEM100 <- log(clbdat$EMv1m)
```

Table 24.1: Fitted parameters of exponential semivariogram for natural logarithm of ECe (estimated by MoM) and of residuals (estimated by REML).

Variable	Nugget	Partial sill	Distance parameter (m)
lnECe	0.116	0.336	192
residuals	0.126	0.083	230

```

clbdat$x <- jitter(clbdat$x, amount=0.001)
clbdat$y <- jitter(clbdat$y, amount=0.001)
dGeoR <- as.geodata(
  obj=clbdat, header=TRUE,
  coords.col=17:18, data.col=16, covar.col=19)
vgm_REML <- likfit(
  geodata=dGeoR, trend=~lnEM100,
  cov.model="exponential", ini.cov.pars=c(0.1,200),
  nugget=0.1, lik.method="REML", messages=FALSE)

```

The REML estimates of the parameters of the residual semivariogram are shown in Table 24.1. The estimated sill of the residual semivariogram is substantially smaller than that of lnECe and the range is somewhat shorter, showing that the linear model explains a considerable part of the spatial variation.

To optimise the locations for KED, using the mean KED variance as a minimisation criterion, a data frame with the covariates at the candidate sampling locations must be assigned to the argument `covars`. The argument `eqn` specifies the KED model as a formula.

```

covars <- EM_CRF
set.seed(314)
rslt <- optimMKV(
  points=50, candi=candi, covars=covars,
  vgm=vgm_REML_gstat, eqn=z~lnEM100cm,
  schedule=schedule,
  plotit=FALSE, track=FALSE)

```

Figure 24.5 shows the optimised locations of a sample of 50 points. This clearly

shows the irregular spatial pattern of the sampling points induced by the covariate.

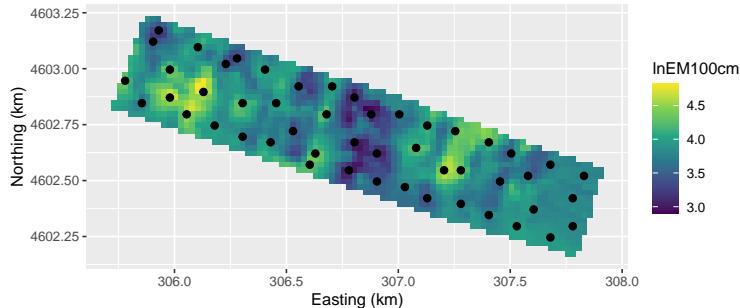


Figure 24.5: Optimised sampling locations for kriging with an external drift of lnEM at the Cotton Research Farm in Uzbekistan, using lnEM as a covariate.

Comparing the population and sample histograms of the covariate clearly shows that locations with either small or large values for the covariate are preferentially selected (Figure 24.6). The optimised sample pattern is a compromise between spreading in geographic space and feature space, see also ? and ?. More precisely, locations are selected by spreading them out throughout the study area, while accounting for the values of the covariates at the selected locations, in a way that locations with covariate values near the minimum and maximum are preferred. This can be explained by noting that the variance of the KED prediction error can be decomposed into two components: the variance of the interpolated residuals and the variance of the predictor of the mean, see Section 22.2. The contribution of the first variance component is minimised through geographical spreading, that of the second component by selecting locations with covariate values near the minimum and maximum.

When one or more covariates are used in sample optimisation, but not used in KED once the data are collected, the sample is suboptimal for the model used in prediction. Inversely, ignoring a covariate in sample optimisation while using this covariate as a predictor also leads to suboptimal samples.

Further, note that a sample with covariate values close to the minimum and maximum only is not desirable if we do not want to rely on the assumption of a linear relation between the study variable and the covariates. To identify a non-linear relation, locations with intermediate covariate values are needed.

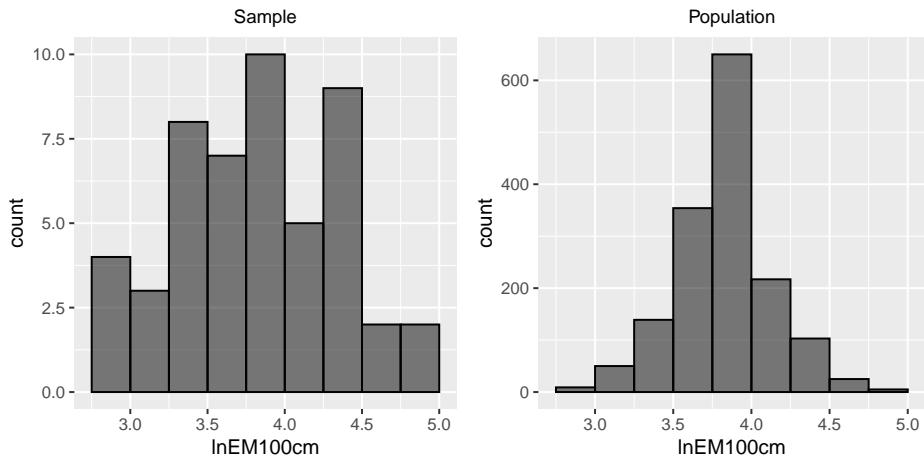


Figure 24.6: Sample and population histograms of $\ln EM$, used as covariate in model-based optimisation of sampling locations for mapping with KED.

Optimisation using a semivariogram with clear spatial structure leads to geographical spreading of the sampling units, so that most likely also locations with intermediate covariate values will be selected.

Exercises

3. Write an **R** script to optimise the coordinates of sixteen points in a square for kriging with an external drift. Use the x -coordinate as a covariate. First create a discretisation grid of 20×20 nodes. Use an exponential residual semivariogram without nugget, with a sill of 2 and a distance parameter of four times the spacing of the discretisation grid. Optimise the locations with spatial simulated annealing (using functions `scheduleSPSANN` and `optimMKV` of package `spsann`)
 - What do you think of the spatial coverage of the optimised sample? Compare the sample with the optimised sample for ordinary kriging, see exercise of Section 24.2
 - Repeat the optimisation using a residual semivariogram, with a nugget of 1.5 and a partial sill of 0.5. Note that the sill is again 2, as before

-
- Compare the optimised sample with the previous sample. What is the most striking difference?
 - How will the optimised sample look like with a pure nugget semivariogram? Check your assumption by using such semivariogram in spatial simulated annealing.

24.4 Model-based infill sampling

Similar to spatial infill sampling using MSSD as a minimisation criterion (Section 18.1), we may design a model-based infill sample. Package **spsann** can be used for this, using argument **points** of function **optimMKV**.

The legacy data are used to estimate the parameters of a spherical semivariogram by maximum likelihood, see Section 22.3.2.

```
library(geoR)
load(file="data/CovariatesThreeWoredasEthiopia.RData")
load(file="data/ThreeWoredasEthiopia.RData")
priodata <- as(priodataEthiopia, "data.frame")
dGeoR <- as.geodata(
  obj=priodata, header=TRUE,
  coords.col=13:14, data.col=1)
vgm_ML <- likfit(
  geodata=dGeoR, trend="cte", cov.model="spherical",
  ini.cov.pars=c(0.4,40), nugget=0.6,
  lik.method="ML", messages=FALSE)
```

In the next code chunk a list is created, containing a data frame (or matrix) with the coordinates of the fixed points (specified with sub-argument **fixed**), and an integer of the number of additional points to be selected (specified with sub-argument **free**). The list is assigned to argument **points** of function **optimMKV**. For kriging I slightly reduced the number of legacy points by keeping one point only per grid cell of $1 \text{ km} \times 1 \text{ km}$. This is done with function **remove.duplicates** of package **sp**.

```

library(sp)
load(file="data/ThreeWoredasEthiopia.RData")
legacy <- remove.duplicates(
  priordataEthiopia, zero=1, remove.second=TRUE)
pnts <- list(fixed=coordinates(legacy), free=100)
candi <- grdEthiopia[,c(1,2)]
names(candi) <- c("x", "y")

```

With numerous legacy points computing time can be reduced with argument `nmax`, which is passed to `gstat` function `krige`.

```

set.seed(314)
vgm_ML_gstat <- vgm(
  model="Sph", psill=vgm_ML$sigmasq,
  range=vgm_ML$phi, nugget=vgm_ML$nugget)
rslt <- optimMKV(
  points=pnts, candi=candi,
  vgm=vgm_ML_gstat, eqn=z~1,
  nmax=20, schedule=schedule, track=FALSE)
ids <- which(rslt$points$free==1)
infillSample <- rslt$points[ids,]

```

Figure 24.7 shows a model-based infill sample of 100 points for ordinary kriging of SOM across the three woredas in Ethiopia. Comparison of the model-based infill sample with the spatial infill sample of Figure 18.4 shows that in a wider zone on both sides of the roads no new sampling points are selected. This can be explained by the large range of 36.9 km of the semivariogram.

24.4.1 Model-based infill sampling for kriging with an external drift

For the study area in Ethiopia maps of covariates are available that can be used in kriging with an external drift (KED), see Section 23.4. The prediction error variance with KED is partly determined by the covariate values, and therefore when filling in the undersampled areas, locations with extreme values for the covariates are preferably selected. The legacy data are used to estimate the residual semivariogram by restricted maximum likelihood (REML), see Section

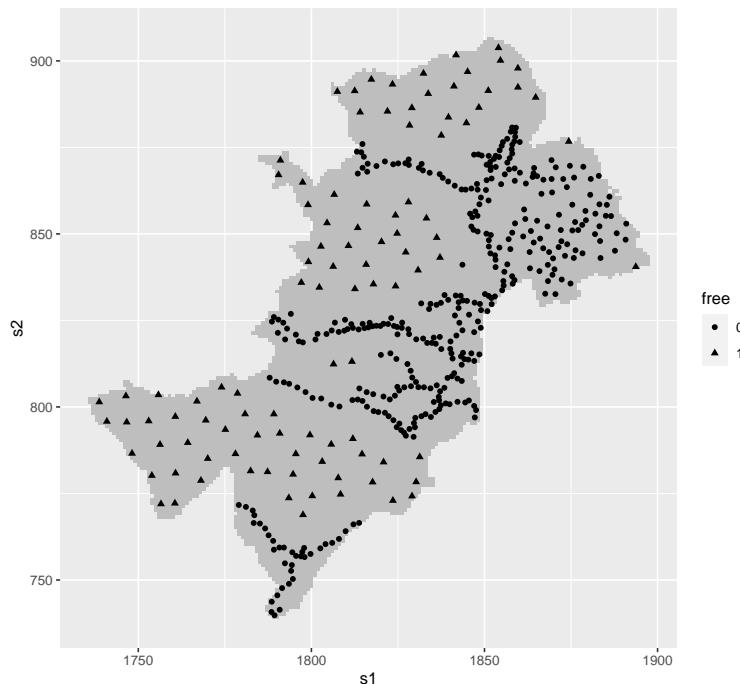


Figure 24.7: Model-based infill sample for ordinary kriging of SOM across three woredas of Ethiopia.

22.4.2.

```
library(geoR)
dGeoR <- as.geodata(
  obj=priodata, header=TRUE,
  coords.col=13:14, data.col=1, covar.col=c(3,9,10,11)
)
vgm_REML <- likfit(
  geodata=dGeoR, trend=~dem+rfl_NIR+rfl_red+lst,
  cov.model="spherical", ini.cov.pars=c(1,5), nugget=0.2,
  lik.method="REML", messages=FALSE)
```

```
covars <- grdEthiopia[,c("dem","rfl_NIR","rfl_red","lst")]
vgm_REML_gstat <- vgm(
  model="Sph", psill=vgm_REML$sigmasq,
  range=vgm_REML$phi, nugget=vgm_REML$nugget)
set.seed(314)
rslt <- optimMKV(
  points=pnts, candi=candi, covars=covars,
  vgm=vgm_REML_gstat,
  eqn=z~dem+rfl_NIR+rfl_red+lst,
  nmax=20, schedule=schedule, track=TRUE)
```

Figure 24.8 shows the optimised sample for KED using as covariates elevation, NIR-reflectance, red-reflectance and land surface temperature. Again the legacy points are avoided, but the infill sampling of the under-sampled areas is less uniform compared to Figure 24.7. Spreading in geographical space is less important than with ordinary kriging because the residual semivariogram has a much smaller range (Table 23.2). Spreading in covariate space does not play any role with ordinary kriging, whereas with KED selecting locations with extreme values for the covariates is important to minimise the uncertainty about the estimated mean.

The mean kriging variance of the optimised sample equals 0.878. This is considerably smaller than the mean kriging variance obtained when the model-based infill sample for ordinary kriging (Figure 24.7) is used in kriging with an external drift: 0.177.

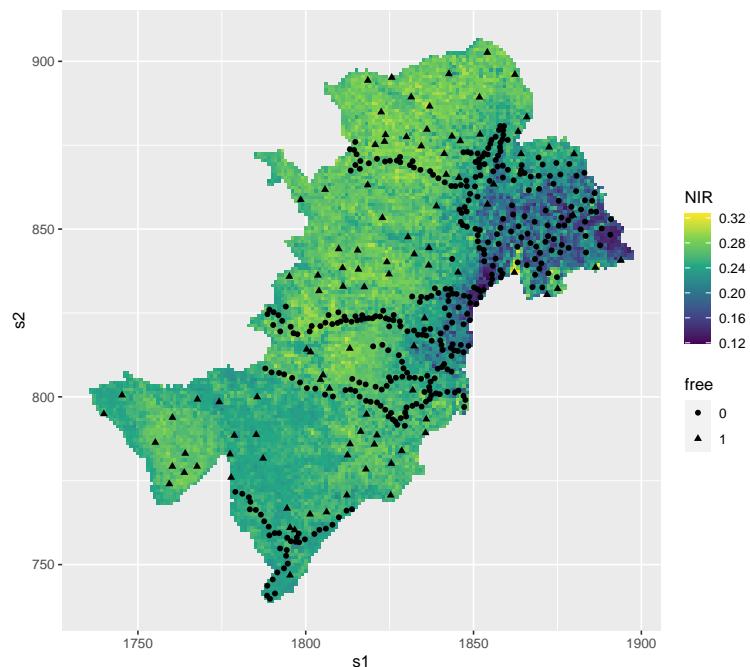


Figure 24.8: Model-based infill sample for kriging with an external drift of SOM across three woredas of Ethiopia, plotted on a map of one of the covariates.

Chapter 25

Sampling for estimating the semivariogram

For model-based sampling as described in Chapters 23 and 24 we must specify the (residual) semivariogram. In case we do not have a reasonable prior estimate from a similar study, we may decide to collect first data with the specific aim of estimating the semivariogram. This semivariogram is subsequently used to design a model-based sample for mapping. This chapter is about how to design a reconnaissance sample survey for estimating the semivariogram.

The first question is how many observations we need for this. ? gave as a rule of thumb that 150 to 225 points are needed to obtain a reliable semivariogram when estimated by the method-of-moments. ? showed that with maximum likelihood (ML) estimation two-third to only half of the observations are needed to achieve equal precision of the estimated semivariogram parameters. Once we have decided on the sample size, we must select the locations of the sampling units. Two random sampling designs for semivariogram estimation are described in this chapter, nested sampling and independent sampling of pairs of points. A following section is devoted to model-based optimisation of the sample pattern for semivariogram estimation. The final section is about how to design a single sample that can be used both for estimating the semivariogram and prediction (mapping).

25.1 Nested sampling

Nested sampling can be used to estimate the semivariance at several chosen separation distances, see `?` and `?`. We must first decide on these separation distances. We need point-pairs at various separations, and especially for small separations so that we get reliable estimates of this part of the semivariogram, which has a strong effect on the kriging weights. Usually separation distances are chosen in a geometric progression, for instance 2, 8, 32, 128 and 512 m. The multiplier, which is four in this example, should be not too small; as a rule of thumb use three or larger.

There are two versions of nested sampling. In the first version, in the first stage several main stations are selected in a way that they cover the study area well, for instance by spatial coverage sampling. In the second stage each of the main stations is used as a starting point to select one point at a distance equal to the largest chosen separation distance (512 m in the example), in a random direction from the main station. This doubles the sample size. In the third stage all points selected in the previous stages (main stations of stage 1 plus the points of stage 2) are used as starting points to select one point at a distance equal to the second largest separation distance, and so on. All points selected in the various stages are included in the nested sample. The code chunk below shows the function for random selection of one point at distance h from a starting point. Note the while loop which continues until a point is found that is inside the area. this is checked with function `over` of package `sp`.

```
SelectPoint <- function(start, h, area){
  dxy <- numeric(length=2)
  inArea <- NA
  while(is.na(inArea)) {
    angle <- runif(n=1, min=0, max=2*pi)
    dxy[1] <- h*sin(angle); dxy[2] <- h*cos(angle)
    xypnt <- start+dxy
    coordinates(xypnt) <- ~s1+s2
    inArea <- as.numeric(over(x=xypnt, y=area))[1]
  }
  xypoint <- as.data.frame(xypnt)
  xypoint
}
```

The first stage of the second version is equal to that of the first version. In the second stage each of the main stations serves as a starting point for randomly selecting a *pair of points* with a separation distance equal to the largest chosen separation distance, with the main station halfway. In the third stage each of the substations is used to select in the same way a pair of points separated by the second largest chosen distance, and so on. Only the points selected in the final stage are used as sampling points. The **R** code below shows the function for random selection of two points separated by h distance units, with a starting point halfway the pair of points. Note that the while loop continues until both points of a pair are inside the area.

```
SelectPair <- function(start, h, area){
  dxy <- numeric(length=2)
  xypoints <- NULL
  inArea1 <- inArea2 <- NA
  while(is.na(inArea1) | is.na(inArea2)) {
    angle <- runif(n=1, min=0, max=2*pi)
    dxy[1] <- h*sin(angle)/2; dxy[2] <- h*cos(angle)/2
    xypnt1 <- start+dxy
    coordinates(xypnt1) <- ~s1+s2
    inArea1 <- as.numeric(over(x=xypnt1, y=area))[1]
    dxy[1] <- -dxy[1]; dxy[2] <- -dxy[2]
    xypnt2 <- start+dxy
    coordinates(xypnt2) <- ~s1+s2
    inArea2 <- as.numeric(over(x=xypnt2, y=area))[1]
  }
  xypoints <- rbind(as.data.frame(xypnt1), as.data.frame(xypnt2))
  xypoints
}
```

The **R** code below shows the selection of a nested sample from the Hunter Valley using both versions. Only one main station is selected. In total sixteen points are selected in four stages. The separation distances are 2000, 1000, 500 and 250 m. Note that the separation distances are in descending order. The largest separation distance should not be chosen too large, because then, when the main station is somewhere in the middle of the study area, it may happen that using the first version no pair can be found with that separation distance. A similar problem may occur with the second version when in subsequent stages a

station is selected near the border of the study area. A copy of the data frame `grd` is made because both the original data frame is needed as well as a gridded version of this data frame.

```
library(sp)
load(file="data/HunterValley.RData")
grd <- grdHunterValley
names(grd)[c(1,2)] <- c("s1","s2")
#make a copy of grd (class of grd is changed)
grid <- grd
gridded(grd) <- ~s1+s2
lags <- c(2000,1000,500,250)
```

The next code chunk is an implementation of the first version of nested sampling.

```
set.seed(614)
id <- sample.int(nrow(grid),1)
mainstation <- grid[id,c(1,2)]
newpnt <- SelectPoint(start=mainstation, h=lags[1], area=grd)
mynestedsample <- rbind(mainstation, newpnt)
for (j in 2:length(lags)) {
  newpnts <- NULL
  for (i in 1:nrow(mynestedsample)) {
    pnts <- SelectPoint(
      start=mynestedsample[i,], h=lags[j], area=grd)
    newpnts <- rbind(newpnts, pnts)
  }
  mynestedsample <- rbind(mynestedsample, newpnts)
}
```

R code for the second version is in the next code chunk.

```
id <- sample.int(nrow(grid), 1)
mainstation <- grid[id,c(1,2)]
pnt <- SelectPoint(start=mainstation, h=lags[1], area=grd)
stations <- rbind(mainstation, pnt)
allstations <- rbind(mainstation, pnt)
```

```

for (j in 2:length(lags)) {
  newstations <- NULL
  for (i in 1:nrow(stations)) {
    pnts <- SelectPair(start=stations[i,], h=lags[j], area=grd)
    newstations <- rbind(newstations, pnts)
    allstations <- rbind(allstations, pnts)
  }
  stations <- newstations
}
mynestedsample_2 <- stations

```

Figure 25.1 shows the two selected nested samples. For the sample selected with the second version also the stations that served as starting point for the selection of the pairs of points are plotted.

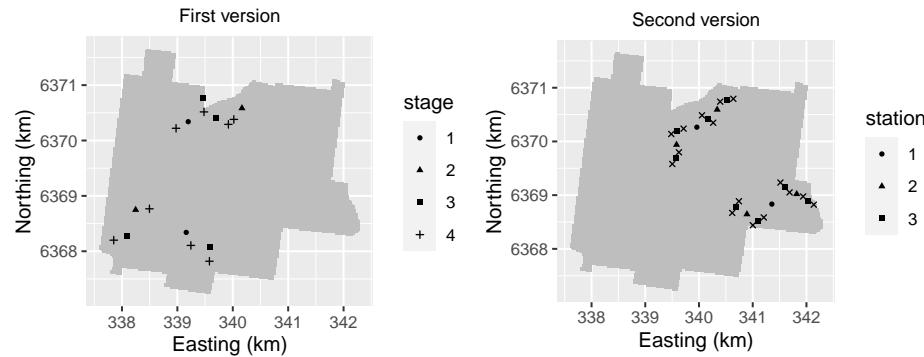


Figure 25.1: Balanced nested samples from Hunter Valley, selected with the two versions. In the subfigure of the second version the selected sampling points (symbol x) are plotted together with the selected stations (halfway the two points of a pair).

The sample of Figure 25.1 is an example of a *balanced* nested sample: in all stages all stations selected in the previous stage are used to select a pair of points. If in the first version of nested sampling all points selected in all previous stages are used to select a new point, then this also results in a balanced nested sample. The number of pairs of points separated by a given distance doubles with every stage. As a consequence, the estimated semivariances for the smallest

separation distance are much more precise than for the largest distance. We are most uncertain about the estimated semivariances for the largest separation distances. If in the first stage only one pair of points is selected separated by the largest distance, then we have only one degree of freedom for estimating the variance component associated with this stage. It is more efficient to select more than one main station, say about ten, and to select fewer points in the final stages. For instance, with the second version we may decide to select a pair of points at only half the number of stations selected in the one-but-last stage. The nested sample then becomes unbalanced.

The model for nested sampling with four stages is a hierarchical ANOVA model with random effects:

$$Z_{ijk} = \mu + A_i + B_{ij} + C_{ijk} + \epsilon_{ijkl} . \quad (25.1)$$

with μ the mean, A_i the effect of the i th first stage station, B_{ij} the effect of the j th second stage station within the i th first stage station, and so on. A_i , B_{ij} , C_{ijk} and ϵ_{ijkl} are random quantities (random effects), all with zero mean, and variances σ_1^2 , σ_2^2 , σ_3^2 and σ_4^2 respectively.

For balanced designs the variance components can be estimated by the method-of-moments from a hierarchical ANOVA. The first step is to assign factors to the sampling points that indicate the grouping of the sampling points in the various stages. The number of factors needed is the number of stages minus one. All factors have two levels. Figure 25.2 shows the levels of the three factors. The levels of the first factor show the strongest spatial clustering, those of the second factor the one-but strongest, and so on.

The **R** code below shows the construction of the three factors for the second version of nested sampling.

```
mynestedsample_2$factor1 <- rep(1:2, each=8)
mynestedsample_2$factor2 <- rep(1:2, each=4)
mynestedsample_2$factor3 <- rep(1:2, each=2)
```

For unbalanced nested designs the variance components can be estimated by restricted maximum likelihood (REML) (?). REML estimation is also recommended if in Equation (25.1) instead of a constant mean μ the mean is a linear combination of one or more covariates (fixed effects). The semivariances at the

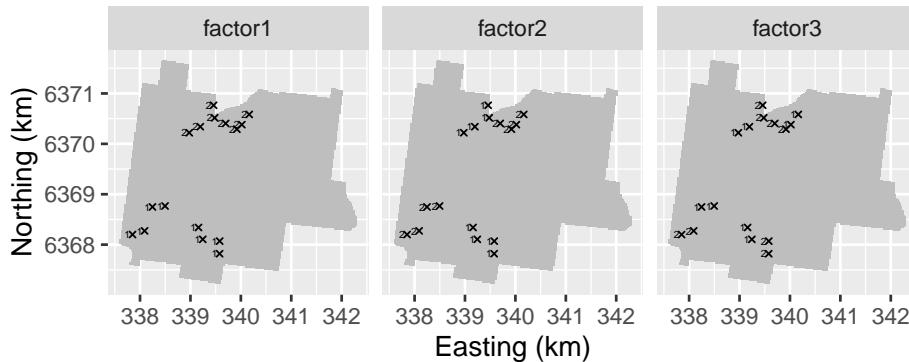


Figure 25.2: The levels of the three factors assigned to the sampling points of the balanced nested sample selected with the first version.

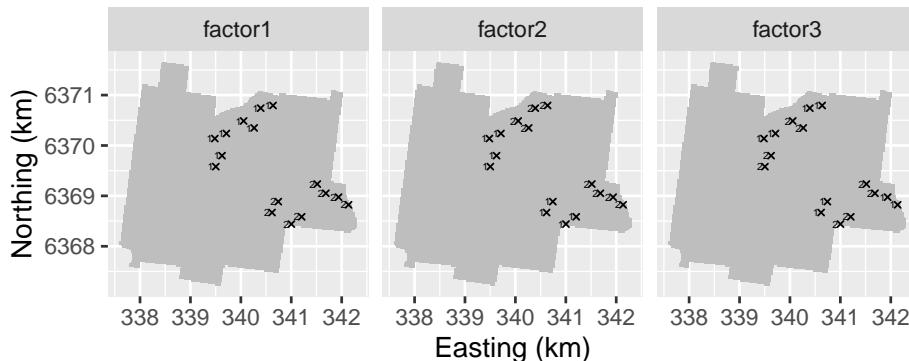


Figure 25.3: The levels of the three factors assigned to the sampling points of the balanced nested sample selected with the second version.

chosen separation distances are obtained by cumulating the estimated variance components.

The **R** code below shows how the variance components and semivariances can be estimated, once the data are collected and added to the data frame, with the function **lme** of the package **nlme** (?). This function fits linear mixed-effects models and allows for nested random effects. It can be used both for balanced and unbalanced nested samples, and for a constant mean or a mean that is a

linear combination of covariates. The argument `fixed` is a formula describing the fixed effects, with the response variable on the left of the `~` operator, and the covariates for the mean on the right. If a constant mean is assumed, as in our example, this is indicated by a `1` on the right of the `~` operator. The argument `random` is a one-sided formula (no response variable is on the left of the `~` operator). On the right of the `|` separator, the nested structure of the data is specified using the factors of Figure 25.2. The `1` on the left of the `|` separator means that we assume that all regression coefficients associated with the covariates are fixed (non-random) quantities.

```
library(nlme)
lmodel <- lme(fixed=z~1, data=mynestedsample,
               random=~1|factor1/factor2/factor3)
res <- as.matrix(VarCorr(lmodel))
sigmas <- as.numeric(res[c(2,4,6,7),1])
sigma <- rev(sigmas)
semivar <- cumsum(sigmas)
```

Random sampling of the points is not strictly needed because a model-based approach is followed here. The model of Equation (25.1) is a superpopulation model, i.e. we assume that the population is generated by this model. ?, for instance, selected the points (using the second version) non-randomly to improve the control of the nested subareas and the average separation distances.

? describes a method for optimisation of a nested design, given the total number of points and the chosen separation distances.

Exercises

1. Write an **R** script to select with the first version a balanced nested sample from the Hunter Valley. Use as separation distances 1,000, 500, 200, 100 and 50 m.
 - Add the factors that are needed for estimating the variance components to the `data.frame` with the selected sampling points.
 - Overlay the sampling points with the `SpatialPixelsDataFrame`, and estimate the semivariances for the attribute compound topographic index (`cti`).

25.2 Independent sampling of pairs of points

With the nested design the estimated semivariances for the different separation distances are not independent. Independent estimated semivariances can be obtained by independent random selection of pairs of points (IPP sampling). Independence here means design-independence, see Section @ref(i.i.d.). Similar to a regression model, a semivariogram can be defined as a superpopulation model or as a population model. In this section a semivariogram is defined at the population level. Such a semivariogram is referred to as a non-ergodic semivariogram or local semivariogram (?). For simple random sampling of point-pairs this method is very straightforward. For each separation distance a point-pair is selected by first selecting fully randomly one point from the study area. Then the second point is randomly selected from the circle with the first point at its center and a radius equal to the chosen separation distance. If this second point is outside the study area, both points are ignored. This is repeated until we have the required point-pairs for this separation distance.

```
SIpairs <- function(h, n, area) {
  topo <- as(getGridTopology(area), "data.frame")
  cellsize <- topo$cellsize[1]
  xy <- coordinates(area)
  dxy <- numeric(length=2)
  xypnts1 <- xypnts2 <- NULL
  i <- 1
  while (i <= n) {
    id1 <- sample.int(n=length(area), size=1)
    xypnt1 <- xy[id1,]
    xypnt1[1] <- jitter(xypnt1[1], amount=cellsize/2)
    xypnt1[2] <- jitter(xypnt1[2], amount=cellsize/2)
    angle <- runif(n=1, min=0, max=2*pi)
    dxy[1] <- h*sin(angle); dxy[2] <- h*cos(angle)
    xypnt2 <- as.data.frame(t(xypnt1+dxy))
    coordinates(xypnt2) <- ~s1+s2
    inArea <- as.numeric(over(x=xypnt2, y=area))[1]
    if (!is.na(inArea)) {
      xypnts1 <- rbind(xypnts1, xypnt1)
      xypnts2 <- rbind(xypnts2, as.data.frame(xypnt2))
      i <- i+1
    }
  }
}
```

```

    }
    rm(xypnt1, xypnt2)
}
cbind(xypnts1, xypnts2)
}

```

IPP sampling is illustrated below with the compound topographic index (cti) data of the Hunter Valley. Five separation distances are chosen, collected in numeric `h`, and for each distance $n = 100$ point-pairs are selected by simple random sampling.

```

library(sp)
h <- c(50,100,200,500,1000)
n <- 100
set.seed(123)
allpairs <- NULL
for (i in 1:length(h)){
  pairs <- SIpairs(h=h[i], n=n, area=grd)
  allpairs <- rbind(allpairs, pairs, make.row.names=FALSE)
}

```

The data frame file `allpairs` has four columns: the spatial coordinates of the first and of the second point of a pair. An overlay is made of the selected points with the `SpatialPixelsDataFrame` and the `cti` values are extracted.

```

pnt1 <- allpairs[,c(1,2)]
coordinates(pnt1) <- ~s1+s2
z1 <- over(x=pnt1, y=grd)[4]
pnt2 <- allpairs[,c(3,4)]
coordinates(pnt2) <- ~s1+s2
z2 <- over(x=pnt2, y=grd)[4]
mysample <- data.frame(h=rep(h, each=n), z1, z2)
names(mysample)[c(2,3)] <- c("z1", "z2")

```

The semivariances for the chosen separation distances are estimated, as well as the variance of these estimated semivariances.

```

gammah <- vgammah <- numeric(length=length(h))
for (i in 1:length(h)){
  units <- which(mysample$h==h[i])
  pairsh <- mysample[units,]
  gammah[i] <- mean((pairsh$z1-pairsh$z2)^2, na.rm=TRUE)/2
  vgammah[i] <- var((pairsh$z1-pairsh$z2)^2, na.rm=TRUE)/(n*4)
}

```

A spherical model with nugget is fitted to the sample semivariogram, using function `nls`, with weights equal to the reciprocal of the estimated variances of the estimated semivariances.

```

samplevariogram <- data.frame(h, gammah, vgammah)
SphNug <- function(h, range, psill, nugget) {
  h <- h/range
  nugget+psill*ifelse(h<1, (1.5*h-0.5*h^3), 1)
}
fit.var <- nls(
  gammah~SphNug(h, range, psill, nugget),
  data=samplevariogram,
  start=list(psill=4, range=200, nugget=1),
  weights=1/vgammah, algorithm="port", lower=c(0,0,0))
print(pars <- signif(coef(fit.var),3))

psill    range   nugget
 3.29  188.00   1.21

```

Figure 25.4 shows the sample semivariogram and the fitted model.

The covariances of the estimated semivariances at different separation distances are 0, as the point-pairs are selected independently. This keeps estimation of the variances and covariances of the estimated semivariogram parameters simple. In the next code chunk this is done by bootstrapping. In bootstrapping for each separation distance a simple random sample *with replacement* of point-pairs is drawn. The number of draws is equal to the total number of pairs per separation distance. So a pair can be selected more than once. Every run of the bootstrap results in as many bootstrap samples as there are separation distances. These bootstrap samples are used to fit a semivariogram model. This whole procedure

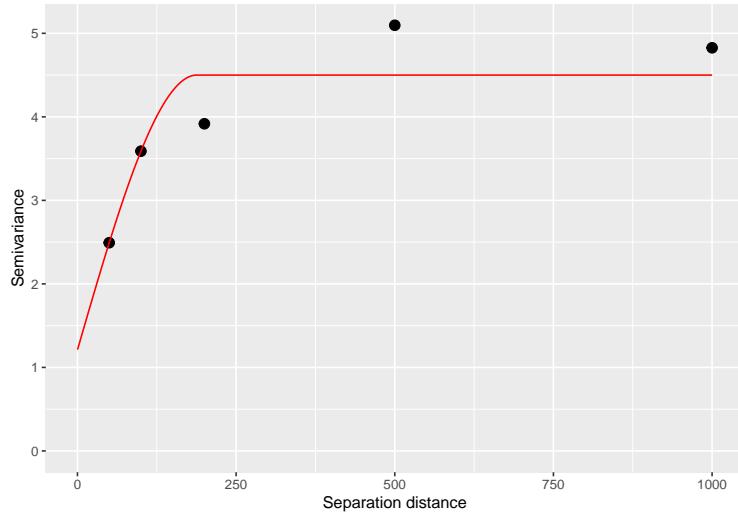


Figure 25.4: Sample semivariogram obtained by independent sampling of pairs of points, and fitted spherical model of cti in Hunter Valley.

is repeated a large number of times, say R times, resulting in R vectors with model parameters, from which we can estimate the variances and covariances of the semivariogram parameters.

```

allpars <- NULL
R <- 500
for (j in 1:R) {
  gammah <- vgammah <- numeric(length=length(h))
  for (i in 1:length(h)) {
    units <- which(mysample$h==h[i])
    pairs <- mysample[units,]
    ids <- sample.int(n, size=n, replace=TRUE)
    mybtps <- pairs[ids,]
    gammah[i] <- mean((mybtps$z1-mybtps$z2)^2, na.rm=TRUE)/2
    vgammah[i] <- var((mybtps$z1-mybtps$z2)^2, na.rm=TRUE)/(n*4)
  }
  samplevariogram <- data.frame(h, gammah, vgammah)
}

```

```

tryCatch({fittedvariogram <- nls(
  gammah~SphNug(h, range, psill, nugget),
  data=samplevariogram,
  start=list(psill=4, range=200, nugget=1),
  weights=1/vgammah, algorithm="port", lower=c(0,0,0))
pars <- coef(fittedvariogram)
allpars <- rbind(allpars,pars)}, error=function(e){}
}
#compute variance-covariance matrix
signif(var(allpars),3)

      psill    range   nugget
psill    1.160   -44.6   -0.803
range   -44.600  66700.0  172.000
nugget   -0.803    172.0    1.070

```

Note the large variance for the range parameter (standard deviation is 215 m), and the negative covariance of the nugget and partial sill parameter (Pearson correlation coefficient is -0.62). Histograms of the three estimated semivariogram parameters are shown in Figure 25.5.

? show how a probability sample of *points* (instead of pairs of points) can be used in design-based estimation of the semivariogram. From the n randomly selected points $n(n - 1)/2$ point-pairs are constructed, and the *second-order inclusion probabilities* of these point-pairs are used in estimating the mean semivariance for separation distance classes. This sampling strategy makes better use of the data, and therefore is potentially more efficient than IPP sampling.

Exercises

2. Write an **R** script to select simple random samples of pairs of points for estimating the semivariogram of cti in Hunter Valley. Use as separation distances 25, 50, 100, 200 and 400 m. Note that these separation distances are smaller than used above. Select 100 pairs per separation distance.
 - Compute the sample semivariogram and estimate a spherical model with nugget using function **nls**.
 - Compare the estimated semivariogram parameters with the estimates obtained with the larger separation distances.

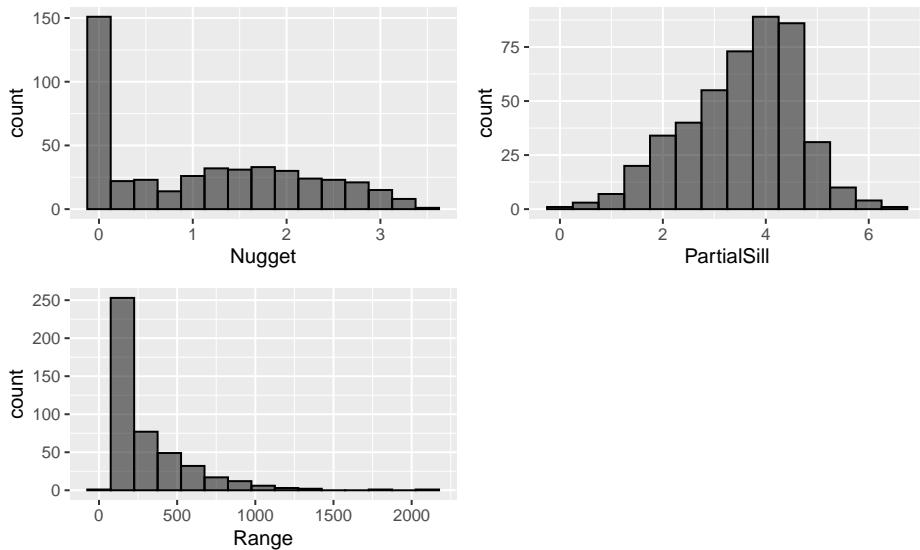


Figure 25.5: Sampling distribution of the estimators of the parameters of a spherical semivariogram with nugget of cti in Hunter Valley.

- Estimate the variance-covariance matrix of the estimated semivariogram parameters by bootstrapping.
- Compare the variances of the estimated semivariogram parameters with the variances obtained with the larger separation distances. Which variance is changed most?

25.3 Model-based optimisation of sample pattern for semivariogram estimation

There is rich literature on model-based optimisation of the sampling locations for semivariogram estimation. Several design criteria (minimisation criteria) have been proposed for optimising the sampling locations. Various authors have proposed a measure of the uncertainty about the semivariogram parameters as a minimisation criterion. This is because the kriging variance, which could be used

as an optimisation criterion, is sensitive to errors in the estimated semivariogram. ? proposed to use a measure of the uncertainty about the kriging variance as a minimisation criterion.

25.3.1 Uncertainty about semivariogram parameters

? and ? proposed the determinant of the variance-covariance matrix of semivariogram parameters, estimated by generalised least squares to fit the experimental method-of-moments semivariogram. For instance, if we have two semivariogram parameters, θ_1 and θ_2 , the determinant of the 2×2 variance-covariance matrix equals $\text{var}(\theta_1)\text{var}(\theta_2) - (\text{covar}(\theta_1, \theta_2))^2$. So, if the two estimated parameters are correlated, the determinant of the matrix is smaller than if they are uncorrelated and the covariance term is 0. The determinant is a measure of our *joint* uncertainty about the semivariogram parameters.

? proposed as a minimisation criterion the logarithm of the determinant of the inverse Fisher information matrix in maximum likelihood (ML) estimation of the semivariogram, hereafter shortly denoted by logdet. The Fisher information about a semivariogram parameter is a function of the likelihood of the semivariogram parameter; the likelihood of a semivariogram parameter is the probability of the data as a function of the semivariogram parameter. The logarithm of this likelihood can be plotted against values of the parameter. The flatter the log-likelihood surface, the less information is in the data about the parameter. The flatness of the surface can be measured by the first derivative of the log-likelihood to the semivariogram parameter. Strong negative or positive derivative values indicate a steep surface. The Fisher information for a model parameter is defined as the expectation of the *square* of the first derivative of the log-likelihood to that semivariogram parameter, see ? for a nice tutorial on this subject. The more information, the less uncertain we are about the parameter, which explains why the inverse of the Fisher information can be used as a measure of uncertainty. The inverse Fisher information matrix contains the variances and covariances of the estimated semivariogram parameters.

The code chunks hereafter show how logdet can be computed. It makes use of the result of ? who showed that each element of the Fisher information matrix $\mathbf{I}(\theta)$ can be obtained with (see also ?)

$$[\mathbf{I}(\theta)]_{ij} = \frac{1}{2} \text{Tr} \left[\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \theta_i} \mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \theta_j} \right], \quad (25.2)$$

with \mathbf{A} the correlation matrix of the sampling points, $\frac{\partial \mathbf{A}}{\partial \theta_i}$ the partial derivative of the correlation matrix to the i th semivariogram parameter, and $\text{Tr}[\cdot]$ the trace of a matrix.

As an illustration I selected a simple random sample of 100 points from Hunter valley. A matrix with distances between the points of a sample is computed. Preliminary values for the semivariogram parameters ξ (ratio of spatial dependence) and ϕ (distance parameter) are specified by visual inspection of the sample semivariogram, and the **gstat** function **variogramLine** is used to compute the correlation matrix.

```
library(sp)
library(gstat)
n <- 100
set.seed(314)
units <- sample.int(nrow(grd), n)
mysample0 <- grd[units,]
coordinates(mysample0) <- ~Easting+Northing
D <- spDists(mysample0)
xi <- 0.8; phi <- 200
thetas <- c(xi,phi)
vgmodel <- vgm(
  model="Exp", psill=thetas[1],
  range=thetas[2], nugget=1-thetas[1])
A <- variogramLine(vgmodel, dist_vector=D, covariance=TRUE)
```

In the next step the semivariogram parameters are slightly changed one-by-one. The changes, referred to as perturbation, are a small fraction of the preliminary semivariogram parameter values. The perturbed semivariogram parameters are used to compute the perturbed correlation matrices (**pA**) and the partial derivatives of the correlation matrix (**dA**) for each perturbation.

```
perturbation <- 0.01
pA <- dA <- list()
for (i in 1:length(thetas)) {
  thetas_pert <- thetas
  thetas_pert[i] <- (1+perturbation)*thetas[i]
  vgmodel_pert <- vgm(
```

```

model="Exp", psill=thetas_pert[1],
range=thetas_pert[2], nugget=1-thetas_pert[1])
pA[[i]] <- variogramLine(
  vgmmodel_pert, dist_vector=D, covariance=TRUE)
dA[[i]] <- (pA[[i]]-A)/(thetas[i]*perturbation)
}

```

Finally the Fisher information matrix is computed, using Equation (25.2), the matrix is inverted, and the determinant is computed.

```

library(matrixcalc)
invA <- chol2inv(chol(A))
I <- matrix(0, length(thetas), length(thetas))
for (i in 1:length(thetas)){
  for (j in i:length(thetas)){
    I[i,j] <- I[j,i] <-
      0.5*matrix.trace(invA%*%dA[[i]]%*%invA%*%dA[[j]])
  }
}
(invI <- chol2inv(chol(I)))

[,1]      [,2]
[1,]  0.1291458 -29.63607
[2,] -29.6360672 10957.87394

logdet0 <- determinant(invI, logarithm=TRUE)$modulus

```

Note the large variance of the distance parameter, equal to 10958. The standard error of the ratio of spatial dependence parameter ξ equals 0.36 (the square root of the value in the first column and first row of the inverse Fisher information matrix), and for the distance parameter ϕ it is 105 m (the square root of the value in the second row and second column). The joint uncertainty about the semivariogram parameters as quantified by the logarithm of the determinant of the matrix equals 6.286. Hereafter we will see how much this joint uncertainty can be reduced by optimising the locations of the sample used for semivariogram estimation, compared to the simple random sample used in the above calculation.

Function `optimUSER` of package **spsann** can be used to search for the locations with the minimum value of `logdet`. This function has been used before in Section 24.2. This package cannot deal with the 22124 candidate grid nodes of Hunter Valley, these are too many. I therefore selected a subgrid of 50 m × 50 m. Note that the argument `cellsize` is also set to this number. The objective function `logdet` is defined in R script `ObjectiveFunctions4MBSamplingVariogram.R`¹². Argument `points` specifies the size of the sample for variogram estimation. A simple random sample of size 100 is used as an initial sample by function `optimUSER`. Argument `model` specifies the model *type*, using the characters of `gstat`. Argument `thetas` specifies the preliminary semivariogram parameter values. Argument `perturbation` specifies how much the variogram parameters are changed to compute the perturbed correlation matrices (`pA`) and the partial derivatives of the correlation matrix (`dA`).

```
source ("Rscripts/ObjectiveFunctions4MBSamplingVariogram.R")
gridded(grd) <- ~Easting+Northing
candi <- spsample(grd, type="regular", cellsize=c(50,50))
candi <- as.data.frame(candi)
names(candi) <- c("x", "y")
schedule <- scheduleSPSANN(
  initial.acceptance=0.8,
  initial.temperature=0.03, temperature.decrease=0.8,
  chains=500, chain.length=2, stopping=5,
  x.min=0, y.min=0, cellsize=50)
set.seed(314)
rslt <- optimUSER(
  points=100, candi=candi,
  fun=logdet,
  model="Exp", thetas=thetas, perturbation=0.01,
  schedule=schedule, track=TRUE)
```

Figure 25.6 shows the optimised sample of 100 points. The `logdet` of the optimised sample equals 2.346 which is 37% of the value of the simple random sample used above to illustrate the computations. The optimised sample shows

¹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/ObjectiveFunctions4MBSamplingVariogram.R>

²<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/ObjectiveFunctions4MBSamplingVariogram.R>

strong spatial clustering. Connecting neighbouring points of the sample by lines results in a sort of mycelium.

25.3.2 Uncertainty about the kriging variance

? proposed as a minimisation criterion the estimation variance of the kriging variance due to uncertainty in the ML estimates of the semivariogram parameters (hereafter denoted by VKV). This variance is approximated by a first order Taylor series, requiring the partial derivatives of the kriging variance with respect to the semivariogram parameters:

$$VKV(\mathbf{s}_0) = \sum_{i=1}^p \sum_{j=1}^p \text{Cov}(\theta_i, \theta_j) \frac{\partial V_{OK}(\mathbf{s}_0)}{\partial \theta_i} \frac{\partial V_{OK}(\mathbf{s}_0)}{\partial \theta_j}, \quad (25.3)$$

with p the number of semivariogram parameters, $\text{Cov}(\theta_i, \theta_j)$ the covariances of the semivariogram parameters θ_i and θ_j (elements of the inverse of the Fisher information matrix $\mathbf{I}^{-1}(\theta)$) (Equation (25.2)), and $\frac{\partial V_{OK}(\mathbf{s}_0)}{\partial \theta_i}$ the partial derivative of the kriging variance to the i th semivariogram parameter at prediction location \mathbf{s}_0 .

The first step is to select a sample for the second sampling round in which data are collected for mapping. For this a spatial coverage sample is selected, using function `stratify` of package `spcosa`, see Chapter 18. The observations on this sample are used for prediction by kriging. The population mean of VKV (MVKV) is used as a minimisation criterion. This population mean is estimated from a centred square grid of 101 points, the evaluation sample. (In the code chunk below `n=100` is used in function `spsample`, but actually 101 locations are selected.)

```
library(spcosa)
grd <- grdHunterValley
names(grd)[c(1,2)] <- c("s1", "s2")
gridded(grd) <- ~s1+s2
n <- 100
set.seed(314)
myStrata <- stratify(grd, nStrata=n, equalArea=FALSE, nTry=10)
mySCsample <- as(spsample(myStrata), "SpatialPoints")
```

```
myevalsample <- spsample(
  x=grd, n=100, type="regular", offset=c(0.5,0.5))
```

The following code chunks show how the estimation variance of the kriging variance at the evaluation point is computed. First the correlation matrix of the spatial coverage sample (A), and the correlation matrix of the spatial coverage sample and the evaluation points (A_0) is computed. Correlation matrix A is extended with a column and row with ones, see Equation (22.4).

```
D <- spDists(mySCsample)
vgmodel <- vgm(model="Exp", psill=thetas[1], range=thetas[2],
                 nugget=1-thetas[1])
A <- variogramLine(vgmodel, dist_vector=D, covariance=TRUE)
nobs <- length(mySCsample)
B <- matrix(data=0, nrow=nobs+1, ncol=nobs+1)
B[1:nobs,1:nobs] <- A
B[1:nobs,nobs+1] <- 1
B[nobs+1,1:nobs] <- 1
D0 <- spDists(x=myevalsample, y=mySCsample)
A0 <- variogramLine(vgmodel, dist_vector=D0, covariance=TRUE)
```

Next the semivariogram parameters are perturbed one-by-one, and the perturbed correlation matrices pA and pA_0 are computed.

```
pA <- pA0 <- list()
for (i in 1:length(thetas)) {
  thetas_pert <- thetas
  thetas_pert[i] <- (1+perturbation)*thetas[i]
  vgmodel_pert <- vgm(
    model="Exp", psill=thetas_pert[1],
    range=thetas_pert[2], nugget=1-thetas_pert[1])
  pA[[i]] <- variogramLine(
    vgmodel_pert, dist_vector=D, covariance=TRUE)
  pA0[[i]] <- variogramLine(
    vgmodel_pert, dist_vector=D0, covariance=TRUE)
}
```

```
pB <- pb <- list()
for (i in 1:length(thetas)) {
  pB[[i]] <- B
  pB[[i]][1:nobs,1:nobs] <- pA[[i]]
  pb[[i]] <- cbind(pA0[[i]],1)
}
```

Next the kriging variance and perturbed kriging variances are computed, and the partial derivatives of the kriging variance with respect to the semivariogram parameters are approximated. See Equations (22.6) and (22.7) for how the kriging weights \mathbf{l} and the kriging variance var are computed.

```
var <- numeric(length=length(myevalsample))
pvar <- matrix(nrow=length(myevalsample), ncol=length(thetas))
for (i in 1:length(myevalsample)) {
  b <- c(A0[i,],1)
  #compute kriging weights and Lagrange multiplier
  l <- solve(B,b)
  var[i] <- 1-l[1:nobs]*%*%A0[i,]-l[nobs+1]
  for (j in 1:length(thetas)){
    pl <- solve(pB[[j]], pb[[j]][i,])
    pvar[i,j] <- 1-pl[1:nobs]*%*%pA0[[j]][i,]-pl[nobs+1]
  }
}
dvar <- list()
for (i in 1:length(thetas)) {
  dvar[[i]] <- (pvar[,i]-var)/(thetas[i]*perturbation)
}
```

Finally the partial derivatives of the kriging variance are used to approximate the estimation variance of the kriging variance. For this the variances and covariances of the estimated semivariogram parameters are needed, estimated by the inverse of the Fisher information matrix (`invI`), see Equation (25.3). Matrix `invI` computed in the previous Section 25.3.1 for the simple random sample of 100 points is used for this. Note that this variance-covariance matrix is computed from the sample for variogram estimation, not from the spatial coverage sample of the second sampling round used for prediction.

```

VKV <- numeric(length=length(var))
for (i in 1:length(thetas)){
  for (j in 1:length(thetas)){
    VKVij <- invI[i,j]*dvar[[i]]*dvar[[j]]
    VKV <- VKV+VKVij
  }
}
MVKV0 <- mean(VKV)

```

For the simple random sample of 100 points the square root of the estimation variance of the kriging variance at the evaluation point equals 0.166. The mean kriging variance at these points equals 0.768, so the uncertainty about the kriging variance is substantial. Hereafter we will see how much this estimation variance can be reduced by optimising the sample with spatial simulated annealing.

As for logdet the sample with minimum value for MVKV can be searched for using **spsann** function **optimUSER**. The objective function **varkrigvar** is defined in **R** script **ObjectiveFunctions4MBSamplingVariogram.R**³⁴. Argument **points** specifies the size of the sample for variogram estimation. A simple random sample of size 100 is used as an initial sample. The argument **psample** is used to specify the sample used for prediction at the evaluation points (after the second round of sampling). The argument **esample** is to specify the sample with evaluation points, used to estimate the population mean of VKV.

```

source ("Rscripts/ObjectiveFunctions4MBSamplingVariogram.R")
schedule <- scheduleSPSANN(
  initial.acceptance=0.8,
  initial.temperature=0.0004, temperature.decrease=0.8,
  chains=500, chain.length=2, stopping=5,
  x.min=0, y.min=0, cellsize=50)
set.seed(314)
rslt <- optimUSER(

```

³<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/ObjectiveFunctions4MBSamplingVariogram.R>

⁴<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/ObjectiveFunctions4MBSamplingVariogram.R>

```
points=100, candi=candi,
fun=varkrigvar,
psample=mySCsample, esample=myevalsample,
model= "Exp", thetas=thetas,
perturbation=0.01, schedule=schedule, track=TRUE)
```

Figure 25.6 shows the optimised sample. The minimised value of MVKV is 27 % of the value of the simple random sample used to illustrate the computations. As for logdet the optimised sample shows strong spatial clustering. However, the shape is quite different. The sample minimised for MVKV is more a spatial coverage sample of an ellipse.

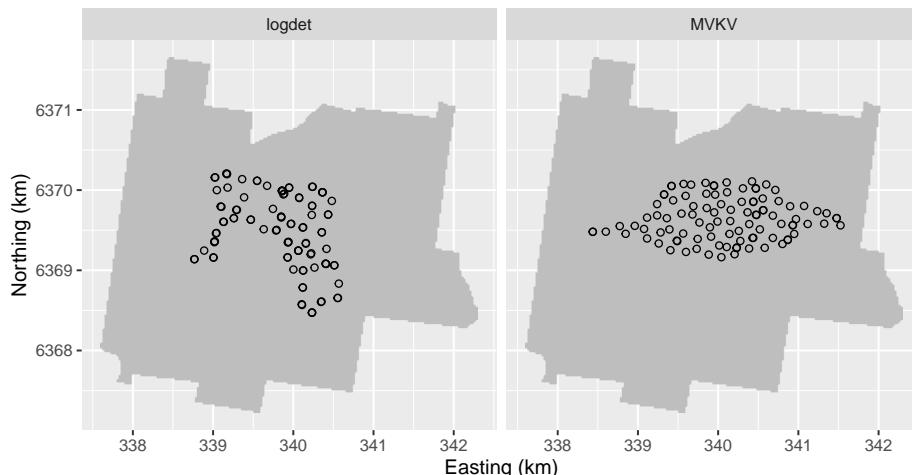


Figure 25.6: Optimised pattern of 100 sampling points for semivariogram estimation, using the logarithm of the determinant of the inverse Fisher information matrix (logdet) or the mean estimation variance of the kriging variance (MVKV) as minimisation criterion.

Both minimisation criteria, logdet and MVKV, are a function of the semivariogram parameters θ , showing that the problem is circular. Using a preliminary estimate of the semivariogram parameters, $\hat{\theta}$, leads to a locally optimal design at $\hat{\theta}$. For that reason ? and ? proposed a Bayesian approach in which a multivariate prior distribution for the semivariogram parameters is postulated, and the

expected value over this distribution of the criterion is minimised. ? computed the average of the estimation variance of the kriging variance over a number of semivariograms.

Both methods for sample optimisation rely on, amongst others, the assumption that the mean and variance are constant throughout the area. Under this assumption it is no problem that the sampling units are spatially clustered. So we assume that the semivariogram estimated from the data collected in a small portion of the study area is representative for the whole study area. If we do not feel comfortable with this assumption, spreading out the sampling units by the sampling methods described in the next two sections can be a good option.

Exercises

3. Write an **R** script to design a model-based sample of 100 points for the Hunter Valley, to estimate the semivariogram for a study variable. Use logdet as a minimisation criterion. Optimisation with this criterion requires a prior estimate of the semivariogram. Use for this an exponential semivariogram with a distance parameter of 200 m and a ratio of spatial dependence of 0.5. Compare the sample with the optimised sample in Figure 25.6, which was obtained with a spatial dependence ratio of 0.8.
4. Repeat this for MVKV as a minimisation criterion.

25.4 Model-based optimisation of a single sample pattern for both semivariogram estimation and prediction

In practice, often a reconnaissance survey for semivariogram estimation is not feasible, and a single sample must be designed that is suitable both for estimating the model parameters and prediction with the estimated model parameters. Another reason is that in a reconnaissance survey we seldom can afford a sample size large enough to obtain reliable estimates of the model parameters. ? found that for a sample size of 192 points the estimated variance components with balanced and unbalanced nested designs were highly uncertain. For this reason it is attractive to use also the sampling points designed for spatial prediction (mapping) for estimating the semivariogram. From this it follows that designing two samples, one for estimating the semivariogram and one for spatial prediction, is suboptimal. Designing one sample that can be used both for estimation

of the model parameters and for prediction potentially is more efficient.

Finally, with nested sampling and sampling of independent pairs of points we aim at estimating the semivariogram of the residuals of a constant mean (see Equation (25.1)). In other words, with these designs we aim at estimating the parameters of a semivariogram model used in ordinary kriging. In situations where we have covariates that can partly explain the spatial variation of the study variable, kriging with an external drift is more appropriate. In these situations the reconnaissance survey should be tailored at estimating both the regression coefficients associated with the covariates and the parameters of the residual semivariogram.

Model-based methods for designing a single sample for estimating the model parameters and for prediction with the estimated model parameters are proposed, amongst others, by ?, ?, ? and ?. The methods use a different minimisation criterion. ? proposed to minimise the kriging variance (at the center of a square grid) that is augmented by an amount that accounts for the additional uncertainty in the kriging predictions due to uncertainty in the semivariogram parameters, hereafter referred to as the *augmented kriging variance* (AKV):

$$AKV(\mathbf{s}_0) = V_{OK}(\mathbf{s}_0) + E[\tau^2(\mathbf{s}_0)] , \quad (25.4)$$

with $V_{OK}(\mathbf{s}_0)$ the ordinary kriging variance, see Equation (22.7), and $E[\tau^2(\mathbf{s}_0)]$ the expectation of the additional variance component due to uncertainty about the semivariogram parameters estimated by ML. The additional variance component is approximated by a first order Taylor series):

$$E[\tau^2(\mathbf{s}_0)] = \sum_{i=1}^p \sum_{j=1}^p \text{Cov}(\theta_i, \theta_j) \frac{\partial \lambda^T}{\partial \theta_i} \mathbf{A} \frac{\partial \lambda}{\partial \theta_j} , \quad (25.5)$$

with $\frac{\partial \lambda}{\partial \theta_j}$ the vector of partial derivatives of the kriging weights with respect to the j th semivariogram parameter. As before in Chapter 24 I use the mean of the AKV over the nodes of a prediction grid (evaluation grid) as a minimisation criterion (MAKV). The same criterion can also be used in situations where we have maps of covariates that we want to use in prediction. In that case the aim is to design a single sample that is used both for estimating the *residual* semivariogram and for prediction by kriging with an external drift. The ordinary

kriging variance $V_{OK}(\mathbf{s}_0)$ in Equation (25.4) is then replaced by the prediction error variance with kriging with an external drift $V_{KED}(\mathbf{s}_0)$, see Equation (22.18).

? proposed as a minimisation criterion a linear combination of the augmented kriging variance (Equation (25.4) and the estimation variance of the kriging variance (Equation (25.3), the *estimation adjusted criterion* (EAC):

$$EAC(\mathbf{s}_0) = AKV(\mathbf{s}_0) + \frac{1}{2V_{OK}(\mathbf{s}_0)} V KV(\mathbf{s}_0)) . \quad (25.6)$$

Again, the mean of the EAC values (MEAC) over the nodes of a prediction grid (evaluation) is used as a minimisation criterion.

Computing time for optimisation of the coordinates of a large sample, say > 50 points, can become prohibitively long. To reduce computing time ? proposed a two-step approach. In the first step, for a fixed proportion $p \in (0, 1)$ the locations of $(1-p) \cdot n$ points are optimised for prediction with given parameters, for instance by minimising MKV. This ‘prediction sample’ is supplemented with $p \cdot n$ points, so that the two combined samples of size n minimise logdet or MVKV. This is repeated for different values of p . In the second step MEAC is computed for the combined samples of size n , and the proportion and associated sample with minimum MEAC is selected.

A simplification of this two-step approach is to select in the first step a square grid or a spatial coverage sample (Chapter 18), and to supplement this sample by a fixed number of points whose coordinates are optimised by spatial simulated annealing (SSA), using either MAKV or MEAC computed from both samples (grid sample or spatial coverage sample plus supplemental sample) as a minimisation criterion. In SSA the grid or spatial coverage sample is fixed, i.e. the locations are not further optimised. ? recommended as a rule of thumb to add about 10% of the fixed sample as short distance points.

The following code chunks show how the AKV and EAC can be computed. First a spatial coverage sample of 90 points is selected using function **stratify** of package **spcosa**, see Chapter 18. In addition, a simple random sample of ten points is selected. This sample is the initial supplemental sample, whose locations are optimised. A square grid of approximately 100 points is selected for evaluation of the minimisation criterion.

```
library(spcosa)
gridded(grd) <- ~s1+s2
n <- 90
set.seed(314)
myStrata <- stratify(grd, nStrata=n, equalArea=FALSE, nTry=10)
mySCsample <- as(spsample(myStrata), "SpatialPoints")
nsup <- 10
units <- sample.int(nrow(grd), nsup)
mysupsample0 <- as(grd[units,], "SpatialPoints")
myevalsample <- spsample(
  x=grd, n=100, type="regular", offset=c(0.5,0.5))
```

The next step is to compute the inverse of the Fisher information matrix, given a preliminary semivariogram model, which is used as the variance-covariance matrix of the estimated semivariogram parameters. Contrary to Section 25.3 now *all* sampling locations are used to compute this matrix. The locations of the spatial coverage sample and the supplemental sample are merged into one `SpatialPoints` object.

```
mysample <- rbind(mysupsample0, mySCsample)
```

For how the inverse of the Fisher information matrix is computed, I refer to the code chunks in Section 25.3.

In the next code chunk for each evaluation point the kriging weights (`L`), the kriging variance (`var`), the perturbed kriging weights (`pL`) and perturbed kriging variances (`pvar`) are computed. In the final lines the partial derivatives of the kriging weights (`dL`) and of the kriging variances (`dvar`) with respect to the semivariogram parameters are computed. The partial derivatives of the kriging variances with respect to the semivariogram parameters are needed for computing the estimation variance of the kriging variance, see Equation (25.3), which is needed for computing criterion EAC, see Equation (25.6).

```
L <- matrix(nrow=length(myevalsample), ncol=nobs)
pL <- array(
  dim=c(length(myevalsample), length(mysample), length(thetas)))
var <- numeric(length=length(myevalsample))
```

```

pvar <- matrix(nrow=length(myevalsample), ncol=length(thetas))
for (i in 1:length(myevalsample)) {
  b <- c(A0[i,],1)
  l <- solve(B,b)
  L[i,] <- l[1:nobs]
  var[i] <- 1-l[1:nobs]*%*%A0[i,]-l[-(1:nobs)]
  for (j in 1:length(thetas)){
    pl <- solve(pB[[j]], pb[[j]][i,])
    pL[i,,j] <- pl[1:nobs]
    pvar[i,j] <- 1-pl[1:nobs]*%*%pA0[[j]][i,]-pl[-(1:nobs)]
  }
}
dL <- dvar <- list()
for (i in 1:length(thetas)) {
  dL[[i]] <- (pL[,,i]-L)/(thetas[i]*perturbation)
  dvar[[i]] <- (pvar[,i]-var)/(thetas[i]*perturbation)
}

```

In the next code the expected variance due to uncertainty about the semivariogram parameters (Equation (25.5)) is computed.

```

tausq <- numeric(length=length(myevalsample))
tausqk <- 0
for (k in 1:length(myevalsample)) {
  for (i in 1:length(dL)){
    for (j in 1:length(dL)){
      tausqijk <- invI[i,j]*t(dL[[i]][k,])%*%A%*%dL[[j]][k,]
      tausqk <- tausqk+tausqijk
    }
  }
  tausq[k] <- tausqk
  tausqk <- 0
}

```

The AKVs are computed by adding the kriging variances and the extra variances due to semivariogram uncertainty (Equation (25.4)). The estimation variances of the kriging variances are computed (VKV), and the EAC values. Both the

AKV and EAC differ among the evaluation points. As a summary, the mean of the two variables is computed.

```
augmentedvar <- var+tausq
MAKVO <- mean(augmentedvar)
VKV <- numeric(length=length(var))
for (i in 1:length(dvar)){
  for (j in 1:length(dvar)){
    VKVij <- invI[i,j]*dvar[[i]]*dvar[[j]]
    VKV <- VKV+VKVij
  }
}
EAC <- augmentedvar+(VKV/(2*var))
MEACO <- mean(EAC)
```

For the spatial coverage sample of 90 points supplemented by a simple random sample of ten points MAKV equals 0.846, MEAC equals 0.896.

The sample can be optimised with **spsann** function **optimUSER**. The argument **points** is a list containing a data frame (or matrix) with the coordinates of the fixed points (specified with sub-argument **fixed**), and an integer of the number of supplemental points of which the locations are optimised (specified with sub-argument **free**). Recall that an important difference with Section 25.3 is that the free and fixed sample are merged, and are used together, both for estimating the semivariogram and for prediction. The objective functions **augvar** and **EAC** are defined in **R** script **ObjectiveFunctions4MBSamplingVariogram.R**⁵⁶.

```
source ("Rscripts/ObjectiveFunctions4MBSamplingVariogram.R")
schedule <- scheduleSPSANN(
  initial.acceptance=0.8,
  initial.temperature=0.008, temperature.decrease=0.8,
  chains=500, chain.length=20, stopping=10,
  x.min=0, y.min=0, cellsize=50)
fixed <- coordinates(mySCsample)
```

⁵<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/ObjectiveFunctions4MBSamplingVariogram.R>

⁶<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Rscripts/ObjectiveFunctions4MBSamplingVariogram.R>

```

names(fixed) <- c("x", "y")
pnts <- list(fixed=fixed, free=10)
set.seed(314)
rslt <- optimUSER(
  points=pnts, candi=candi,
  fun=augvar,
  esample=myevalsample,
  model="Exp", thetas=thetas, perturbation=0.01,
  schedule=schedule, track=TRUE)

```

Figure 25.7 shows for the Hunter Valley spatial coverage samples of 90 points, supplemented by ten points optimised by SSA, using MAKV and MEAC as a minimisation criterion.

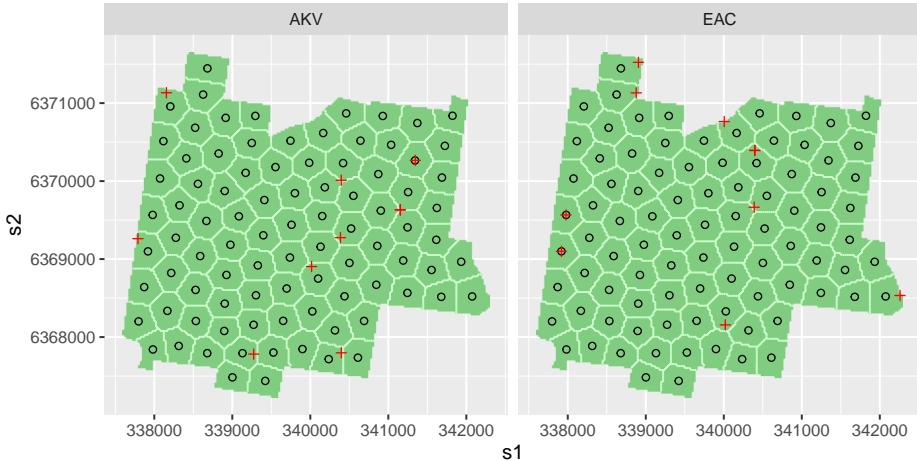


Figure 25.7: Optimised sample of ten points supplemented to spatial coverage sample, for semivariogram estimation and prediction, using the mean augmented kriging variance (MAKV) or the mean estimation adjusted criterion (MEAC) as minimisation criterion.

Figure 25.7 shows that with MAKV as a minimisation criterion there is one point that is very close to a spatial coverage sample point (distance < 2 m.). A histogram of the shortest distance to the spatial coverage sample is shown in Figure 25.8. With MEAC two points are at very short distance of the spatial

coverage sample (< 3 m). The average distance between neighbouring spatial coverage sampling points equals 381.

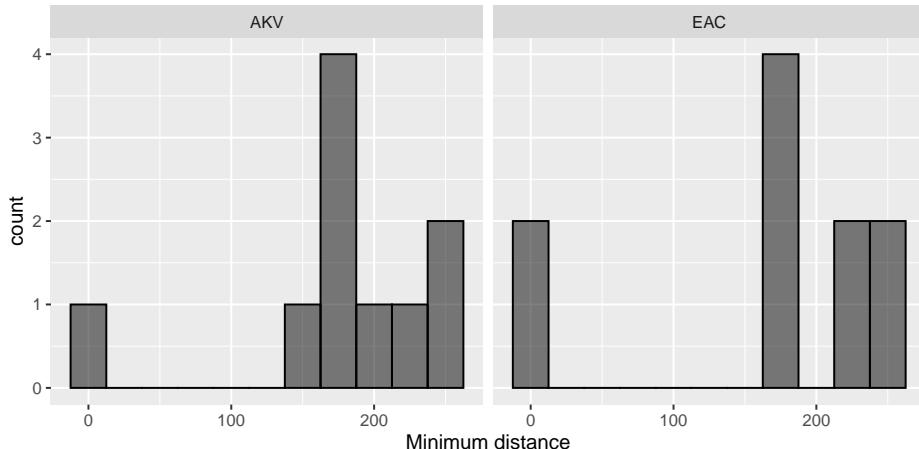


Figure 25.8: Shortest distance of supplemental sample, optimised with mean augmented kriging variance (AKV) and mean estimation adjusted criterion (EAC), to spatial coverage sample.

The MAKV of the optimised sample equals 0.775 which is 92% of the MAKV of the initial sample. The MEAC of the optimised sample equals 0.788 which is 88% of the MEAC of the initial sample. The reduction of these two criteria through the optimisation is much smaller than for logdet and MVKV in Section 25.3. This can be explained by the small number of sampling units that is optimised: only the locations of ten units are optimised, 90 are fixed. In Section 25.3 all 100 locations were optimised.

Exercises

5. Write an **R** script to select from the Hunter Valley a spatial coverage sample of 90 points supplemented by 10 points. Use MEAC as a minimisation criterion, an exponential semivariogram with a distance parameter of 200 m and a ratio of spatial dependence of 0.5. Compare the sample with the optimised sample in Figure 25.7, which was obtained with a spatial dependence ratio of 0.8.

25.5 A practical solution

Based on the optimised samples shown above, a very straightforward, simple sampling design for estimating the model parameters and for prediction is a spatial coverage sample supplemented with randomly selected points in between the points of the spatial coverage sample at some chosen, fixed distances. Figure 25.9 shows an example. A subsample of ten points is selected from the 90 points of the spatial coverage sample, using simple random sampling without replacement. These points are used as a starting point to select a point at a distance of 20 m in a random direction.

```
h <- 20
m <- 10
set.seed(314)
units <- sample.int(nrow(mySCsample), m, replace=FALSE)
mySCsubsample <- mySCsample[units,]
dxy <- matrix(nrow=m, ncol=2)
angle <- runif(n=m, min=0, max=2*pi)
dxy[,1]=h*sin(angle); dxy[,2]=h*cos(angle)
mysupsample <- mySCsubsample+dxy
```

The MAKV of this practical sample equals 0.801, and the MEAC equals 0.809. For MAKV 36% of the maximal reduction is realised by this practical solution; for MEAC this is 20%.

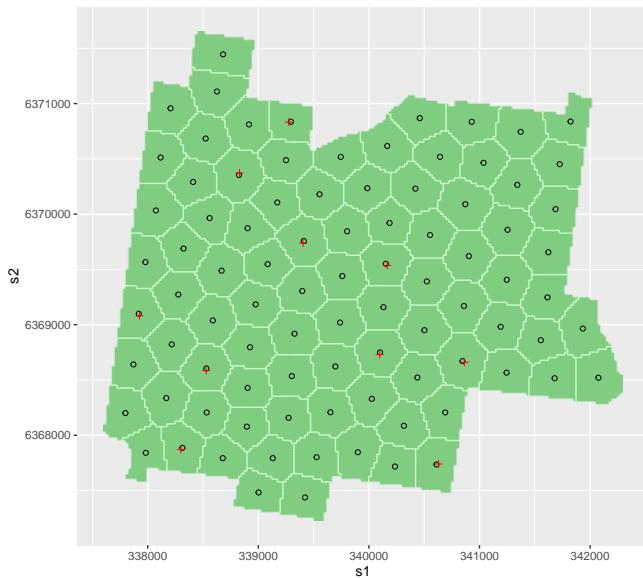


Figure 25.9: Spatial coverage sample of 90 points supplemented by ten points at short distance (20 m) from randomly selected spatial coverage point.

Chapter 26

Sampling for validation of maps

In the previous chapters of Part II various methods are described for selecting sampling units with the aim to map the study variable. Once the map has been made, we would like to know how good it is. It should come as no surprise that the value of the study variable at a randomly selected location as shown on the map differs from the value at that location in reality. This difference is a prediction error. The question is how large this error is on average, and how variable it is. In this chapter I describe and illustrate with a real world case study how to select sampling units at which we will confront the predictions with the true values, and how to estimate map quality indices from the prediction errors at these sampling locations.

If the map has been made with a statistical model, then the predictors are typically model-unbiased and the variance of the prediction errors can be computed from the model. Think for instance of kriging, which also results in a map of the kriging variance. In Chapters 23 and 24 I showed how this kriging variance can be used to optimise the grid spacing (sample size) and the spatial coordinates of the sampling units for mapping. So, if we have a map of these variances, why do we still need to collect new data for estimating the map quality?

The problem is that these kriging variances rely on the validity of the assumptions made in modelling the spatial variation of the study variable. Do we as-

sume a constant mean, or a mean that is linear combination of some covariates? Which covariates? Or should we model the mean with a non-linear function, as in Random Forests? How good is our estimate of the semivariogram model form and parameters, either for the target variable or the residuals from some model of the covariates? If one or more of these modelling assumptions are violated, the variances of the prediction errors may become biased. For that reason, the quality of the map is preferably determined through independent validation, i.e. by comparing predictions with observations not used in mapping, and followed by *model-free* design-based estimation of the map quality indices. This process is often referred to as validation, perhaps better statistical validation, a subset of the more comprehensive term map quality evaluation, which includes the concept of fitness-for-use.

Statistical validation of maps is often done through data-splitting or cross-validation. In data-splitting the data are split into two subsets, one for calibrating the model and mapping, one for validation. In cross-validation the data set is split into a number of disjoint subsets of equal size. Each subset is used one-by-one for calibration and prediction. The remaining subsets are used for validation. Leave-one-out-crossvalidation (LOOCV) is a special case of this, in which each sampling unit is left out one-by-one, and all other units are used for calibration and prediction of the study variable of the unit that is left out. The problem with data-splitting and cross-validation is that the data used for mapping typically are from non-probability samples. This makes model-free estimation of the map quality indices unfeasible (?). Designing a sampling scheme starts with a comprehensive description of the aim of the sampling project (?). Mapping and validation are different aims, which ask for different sampling approaches. For validation probability sampling is the best option, because then a statistical model of the spatial variation of the prediction errors is not needed, and map quality indices defined as population parameters (see next section) can be estimated model-free by design-based inference (see also Section 1.2).

26.1 Map quality indices

First I would like to emphasize that in validation we want to assess the accuracy of the map as a whole. We are not interested in the accuracy at a sample of population units only. For instance, we would like to know the prediction error averaged over all population units, and not merely the average prediction error at a sample of units. Map quality indices are therefore defined as population

means. Because we cannot afford to determine the prediction error for each unit of the mapping area to calculate these population means (if we could do that there would be no need for a mapping model), we have to take a sample of units at a limited number of locations in the mapped area. This sample is then used to *estimate* the population means and our uncertainty about these estimated means, as quantified by their sampling standard errors.

For quantitative maps, i.e. maps depicting a quantitative study variable popular map quality indices are the population mean error (ME), the population mean absolute error (MAE) and the population mean squared error, defined as

$$ME = \frac{1}{N} \sum_{k=1}^N (\hat{z}_k - z_k) \quad (26.1)$$

$$MAE = \frac{1}{N} \sum_{k=1}^N (|\hat{z}_k - z_k|) \quad (26.2)$$

$$MSE = \frac{1}{N} \sum_{k=1}^N (\hat{z}_k - z_k)^2 , \quad (26.3)$$

with N the total number of units (e.g. pixels) in the population, \hat{z}_k the predicted value for unit k , z_k the true value of that unit, and $|\cdot|$ the absolute value operator. For infinite populations the sum must be replaced by an integral. The ME quantifies the systematic error, and ideally equals 0. It can be positive (in case of overprediction) and negative (in case of underprediction). Positive and negative errors cancel out, and as a consequence the ME does not quantify the magnitude of the prediction errors. The MAE and RMSE do quantify the magnitude of the errors, they are nonnegative. Often the square root of MSE is taken, denoted by RMSE, so that the units are the same as of the study variable. The RMSE is strongly effected by outliers (large prediction errors), due to the squaring of the errors, and for that reason it is recommended to estimate both MAE and RMSE.

Two other important map quality indices are the population coefficient of determination and the Nash-Sutcliffe model efficiency coefficient (MEC). The population coefficient of determination R^2 is defined as the square of the Pearson correlation coefficient r of the study variable and the predictions of the study variable, given by

$$r = \frac{\sum_{k=1}^N (z_k - \bar{z})(\hat{z}_k - \bar{\hat{z}})}{\sqrt{\sum_{k=1}^N (z_k - \bar{z})^2} \sqrt{(\hat{z}_k - \bar{\hat{z}})^2}} = \frac{S^2(z, \hat{z})}{S(z)S(\hat{z})}, \quad (26.4)$$

with \bar{z} the population mean of the study variable, $\bar{\hat{z}}$ the population mean of the predictions, $S^2(z, \hat{z})$ the population covariance of the study variable and the predictions of z , $S(z)$ the population standard deviation of the study variable, and $S(\hat{z})$ the population standard deviation of the predictions. Note that R^2 is unaffected by bias, and therefore should not be used in isolation, but always accompanied by ME.

MEC is defined as (?)

$$MEC = 1 - \frac{\sum_{k=1}^N (\hat{z}_k - z_k)^2}{\sum_{k=1}^N (z_k - \bar{z})^2} = 1 - \frac{MSE}{S^2(z)}, \quad (26.5)$$

with $S^2(z)$ the population variance of the study variable. MEC quantifies the improvement made by the model over using the mean of the observations as prediction. A value of one ($MEC = 1$) indicates a perfect match between the measured and predicted values of the study variable, whereas a value of 0 indicates that the mean of the measured values is as good a predictor as the model. A negative value occurs when the mean of the measured values is a better predictor than the model, i.e. when the residual variance is larger than the variance of the measurements.

For categorical maps a commonly used map quality index is the overall purity, which is defined as the proportion of units (finite population) or fraction of the area (infinite populations) that is correctly classified (mapped):

$$P = \frac{1}{N} \sum_{k=1}^N y_k, \quad (26.6)$$

with y_k an indicator for unit k having value 1 if the predicted class equals the true class, and 0 otherwise:

$$y_k = \begin{cases} 1 & \text{if } \hat{c}_k = c_k \\ 0 & \text{otherwise,} \end{cases} \quad (26.7)$$

with c_k and \hat{c}_k the true and predicted class of unit k , respectively.

The population ME, MSE, R^2 , MEC and purity can also be defined for subpopulations. For categorical maps natural subpopulations are the classes depicted in the map, the map units. In that case the purity of map unit u is defined as the fraction of the area of map unit u that is correctly mapped as u .

A different subpopulation is the part of the population that is *in reality* class u (but possibly not mapped as u). We are interested in the fraction of the area covered by this subpopulation that is correctly mapped as u . This is referred to as the class representation of class u , for which I use hereafter the symbol R_u .

26.1.1 Estimation of map quality indices

The map quality indices are defined as population or subpopulation means. To estimate these (sub)population means a design-based sampling approach is most appropriate. Sampling units are selected by probability sampling, and the map quality indices are estimated by model-free, design-based inference. So the ME of a finite population can be estimated by the π estimator (see Equation (2.4)):

$$\widehat{ME}_\pi = \frac{1}{N} \sum_{k \in S} \frac{1}{\pi_k} e_k , \quad (26.8)$$

with $e_k = \hat{z}_k - z_k$ the prediction error for unit k . By taking the absolute value of the prediction errors, or by squaring the prediction errors e_k in Equation (26.8), the π estimator for the MAE and MSE is obtained. By replacing e_k by the indicator y_k of Equation (26.7), the π estimator for the overall purity is obtained.

With simple random sampling the square of the sample correlation coefficient, i.e. the correlation of the study variable and the predictions of the study variable in the sample, is an unbiased estimator of R^2 . See ? (p.486-491) for how to estimate R^2 for other sampling designs.

The population MEC can be estimated by

$$\widehat{MEC} = 1 - \frac{\widehat{MSE}}{\widehat{S^2}(z)} , \quad (26.9)$$

For simple random sampling the sample variance, i.e. the variance of the observations of z in the sample, is an unbiased estimator of the population variance $S^2(z)$. For other sampling designs this population variance can be estimated by Equation (4.9).

Estimation of the class representations is slightly more difficult because the sizes of the classes (number of pixels or areas where in reality class u is present) are unknown, and therefore must also be estimated from the sample. This leads to the estimator of a ratio (?):

$$\hat{R}_{u,\text{ratio}} = \frac{\sum_{k \in \mathcal{S}} \frac{y_k}{\pi_k}}{\sum_{k \in \mathcal{S}} \frac{x_k}{\pi_k}}, \quad (26.10)$$

where $y_{u,k}$ denotes an indicator defined as

$$y_{u,k} = \begin{cases} 1 & \text{if } \hat{c}_k = c_k = u \\ 0 & \text{otherwise,} \end{cases} \quad (26.11)$$

and x_k denotes an indicator defined as

$$x_k = \begin{cases} 1 & \text{if } c_k = u \\ 0 & \text{otherwise.} \end{cases} \quad (26.12)$$

This estimator is also recommended in situations where the sample size is not fixed but varies among samples selected with the sampling design. This is the case, for instance, when estimating the mean error or purity of a given map unit from a simple random sample. The number of selected sampling units within the map unit is uncontrolled and varies between simple random samples. In this case we can either estimate the mean error or purity of a map unit u by using the *known* size (area, number of pixels) of map unit u in the denominator, or the *estimated* size. Interestingly, by dividing by the *estimated* size of the map unit instead of its known area, the estimator generally becomes more precise (?). See also Section 14.1.

In principle all probability sampling designs described in Part I are appropriate for validation. ? and ? evaluated five basic probability sampling designs, and concluded that in general stratified random sampling is a good choice. For validation of categorical maps natural strata are the map units, i.e. the groups of polygons or grid cells assigned to each class. Systematic random sampling is

less suitable as no unbiased estimator of the sampling variance of the estimator of the population mean exists for this design (see Chapter 5). For validation of maps of extensive areas, think of whole continents, travel time between sampling locations can become substantial. In this case sampling designs that lead to spatial clustering of validation locations can become efficient, for instance two-stage cluster random sampling (Chapter 7) or cluster random sampling (Chapter 6).

26.2 Real-world case study

As an illustration two soil maps of the three northern counties of Xuancheng City (Anhui province, China), both depicting soil organic matter (SOM) concentration (g/kg) in the topsoil, are evaluated. In Section 13.2 the data of three samples, including the stratified random sample, were merged to estimate the parameters of a spatial model for the natural log of SOM. Here only the data of the two non-random samples are used to map SOM. The stratified simple random sample is used for validation.

Two methods are used in mapping, kriging with an external drift (KED) and a machine learning method, random forest (RF). For mapping with RF seven covariates are used: planar curvature, profile curvature, slope, temperature, precipitation, topographic wetness index and elevation. For mapping with KED only the two most important covariates in the RF model are used: precipitation and elevation. The two maps are evaluated by statistical validation with a stratified simple random sample of 62 units (points). The strata are the eight units of a geological map (Figure 26.1).

The two maps that are to be validated are shown in Figure 26.2. Note that non-soil areas (built-up, water, roads) are not predicted. The maps are quite similar. The most striking difference between the maps is the highest and lowest predictions by KED that are not seen in the RF map.

26.2.1 Estimation of the population Mean Error and Mean Squared Error

To estimate the population MSE of the two maps, first the squared prediction errors are computed. The name of the measured study variable at the validation sample in the data frame `mysample` is `SOM_A_hori`. Four new columns are added to `mysample`, by computing the prediction errors for KED and RF, and squaring

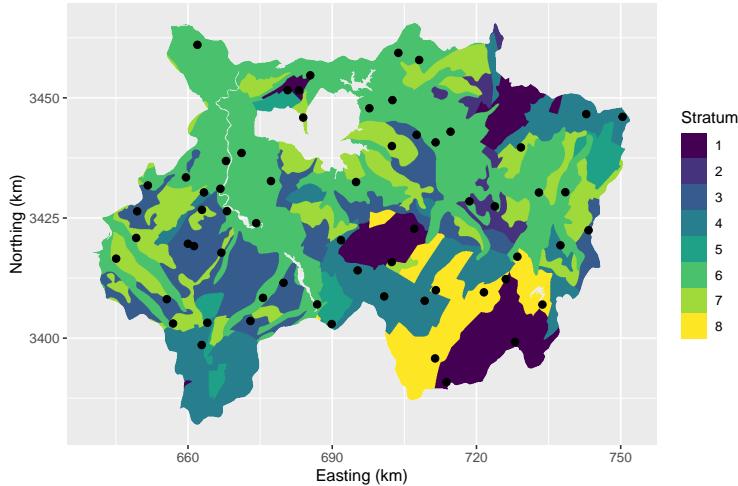


Figure 26.1: Stratified simple random sample for validation of the map of soil organic map concentrations in topsoil of Xuancheng, China.

these errors. The next code chunks show the computations for KED only.

```
mysample <- read.csv(
  file="data/Xuancheng_Stratifiedrandomsample.csv", header=TRUE)
mysample$eKED <- mysample$SOM_A_hori-mysample$KED
mysample$e2KED <- (mysample$eKED)^2
```

These four new variables now are our study variables, of which we would like to estimate the population means. These population means can be estimated as explained in Chapter 4. First the stratum weights are computed, that is the relative number of raster cells.

```
strata <- read.csv(file="data/Xuancheng_StrataSize.csv")
w_h <- strata$Nh/sum(strata$Nh)
```

Now the stratum means of the prediction errors, obtained with KED and RF, are estimated by the sample means, and the populations mean of the errors are estimated by the weighted mean of the estimated stratum means.

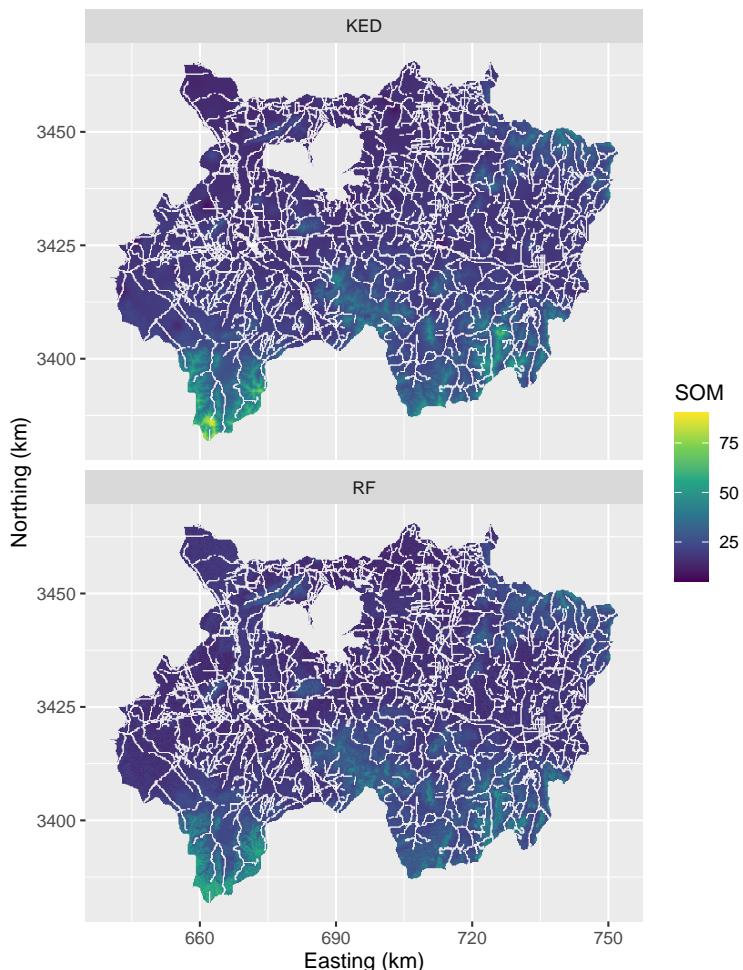


Figure 26.2: Map of soil organic matter concentration (g/kg) in the topsoil of Xuancheng, obtained by kriging with an external drift (KED) and random forest (RF).

```
me_KED_h <- tapply(mysample$eKED, INDEX=mysample$stratum, FUN=mean)
me_KED <- sum(w_h*me_KED_h)
```

This is repeated for the squared prediction errors.

```
mse_KED_h <- tapply(mysample$e2KED, INDEX=mysample$stratum,
                      FUN=mean)
mse_KED <- sum(w_h*mse_KED_h)
```

Exercises

1. The estimated MSE of the KED map equals 89.4, that of the RF map 95.9. Are you certain that the population MSE of the KED map is smaller than the population MSE of the RF map?

26.2.2 Estimation of the standard error of the estimated population ME and MSE

We are uncertain about both population MSE's, as we measured the squared errors at 62 sampling points only. So we would like to know how uncertain we are. This uncertainty is quantified by the standard error of the estimated population MSE. A problem is that in the second stratum we have only one sampling point. So for this stratum we cannot compute the variance of the squared errors. To compute the variance we need at least two sampling points.

```
table(mysample$stratum)
```

1	2	3	4	5	6	7	8
5	1	8	10	2	23	9	4

A solution is to merge stratum 2 with stratum 1, which is a similar geological map unit (we know this from the domain expert). This is referred to as collapsing the strata. A look-up table is constructed to add the collapsed strata identifiers to the `data.frame` `mysample` and `strata`, using function `merge` of the `base` R library.

Table 26.1: Estimated population mean error (ME) and mean squared error (MSE) of KED and RF map, and their standard errors.

	KED	seKED	RF	seRF
ME	0.81	1.2	0.55	1.31
MSE	89.40	25.5	95.90	26.30

```

levels <- sort(unique(mysample$stratum))
collapsedstrata <- c(1,1,2,3,4,5,6,7)
# look-up table
lut <- data.frame(stratum=levels, collapsedstrata)
mysample <- merge(x=mysample, y=lut)
strata <- merge(x=strata, y=lut)

```

Now the collapsed strata can be used to estimate the standard errors of the estimated population MSE's. As a first step the stratum weights and the sample sizes of the collapsed strata are computed.

```

N_hc <- tapply(strata$Nh, INDEX=strata$collapsedstrata, FUN=sum)
w_hc <- N_hc/sum(N_hc)
n_hc <- table(mysample$collapsedstrata)

```

The sampling variance of the estimator of the mean of the (squared) prediction error can be estimated by Equation (4.4). In the next code chunk this is shown for the RF predictions only.

```

s2e_KED_hc <- tapply(
  mysample$eKED, INDEX=mysample$collapsedstrata, FUN=var)
se_me_KED <- sqrt(sum(w_hc^2*s2e_KED_hc/n_hc))
s2e2_KED_hc <- tapply(
  mysample$e2KED, INDEX=mysample$collapsedstrata, FUN=var)
se_mse_KED <- sqrt(sum(w_hc^2*s2e2_KED_hc/n_hc))

```

Exercises

2. Do you think there is a systematic error in the KED and RF predictions?
3. Do you think the difference between the two estimated population MSE's is statistically significant?

26.2.3 Estimation of MEC

To estimate MEC, we must first estimate the population variance of the study variable from the stratified simple random sample (the denominator in Equation (26.9)). This population variance is estimated with function `s2` of package **surveyplanning** (Section 4.1.2).

```
library(surveyplanning)
lut <- data.frame(collapsedstrata=1:7, weight=N_hc/n_hc)
mysample <- merge(x=mysample, y=lut)
s2z <- s2(mysample$SOM_A_hori, w=mysample$weight)
```

Now the MEC's for KED and RF can be estimated.

```
mec_KED <- 1-(mse_KED/s2z)
mec_RF <- 1-(mse_RF/s2z)
```

The estimated MEC for KED equals 0.024, and for RF -0.046, showing that the two models used in mapping are no better than the estimated mean of SOM used as a predictor. This is quite a disappointing result.

26.2.4 Statistical testing of hypothesis about population ME and MSE

The hypothesis that the population ME equals 0 can be tested by a one-sample *t*-test. The alternative hypothesis is that ME is unequal to 0 (two-sided alternative). The number of degrees of freedom of the *t* distribution is approximated by the total sample size minus the number of strata (Section 4.2). Note that we have a two-sided alternative hypothesis, so we must compute a two-sided *p*-value.

```
t_KED <- me_KED/se_me_KED
df <- nrow(mysample)-length(unique(mysample$collapsedstrata))
lowertail <- (t_KED<0)
p_KED <- 2*pt(t_KED, df=df, lower.tail=lowertail)
```

The outcomes of the test statistics are 0.676 and 0.418 for KED and RF, respectively, with p -values 0.502 and 0.678. So we clearly have not enough evidence for systematic errors, neither with KED, nor with RF mapping.

Now we test whether the two population MSE's differ significantly. This can be done by a paired t -test. The first step in a paired t -test is to compute pairwise differences of squared predictions errors, and then we can proceed as in a one-sample t -test.

```
mysample$dife2 <- mysample$e2KED-mysample$e2RF
mdife2_h <- tapply(
  mysample$dife2, INDEX=mysample$stratum, FUN=mean)
mdife2 <- sum(w_h*mdife2_h)
vardife2_h <- tapply(
  mysample$dife2, INDEX=mysample$collapsedstrata, FUN=var)
sdmdife2 <- sqrt(sum(w_hc^2*vardife2_h/n_hc))
t <- mdife2/sdmdife2
lowertail <- (t<0)
p <- 2*pt(t, df=df, lower.tail=lowertail)
```

The outcome of the test statistic is -0.632, with a p -value of 0.53, so we clearly do not have enough evidence that the population MSE's obtained with the two mapping methods are different.

Chapter 27

Design-based, model-based and model-assisted approach for sampling and inference

In Section 1.2 I already mentioned the design-based and model-based approach for sampling and statistical inference. In this chapter the fundamental differences between these two approaches are explained in more detail. Several misconceptions about the design-based approach for sampling and statistical inference, based on classical sampling theory, seem to be quite persistent. These misconceptions are the result of confusion about basic statistical concepts such as independence, expectation, and bias and variance of estimators or predictors. These concepts have a different meaning in the design-based and model-based approach. Besides, a population mean is still often confused with a model mean, and a population variance with a model variance, leading to invalid formulas for the variance of an estimator of the population mean. In this chapter the fundamental differences between these two approaches are illustrated with simulations, so that hopefully a better understanding of this subject is obtained. This Chapter has been published as part of a journal paper, see ?.

27.1 Two sources of randomness

In my classes about spatial sampling I ask the participants the following question. Suppose we have measurements of a soil property, for instance soil organic carbon content, at two locations separated by 20 cm. Do you think these two measurements are correlated? I ask them to vote for one of three answers:

1. Yes, they are ($> 80\%$ confident)
2. No, they are not ($> 80\%$ confident)
3. I do not know

Most students vote for answer 1, the other students vote for answer 3, nearly no one votes for answer 2. Then I explain that you cannot say, simply because for correlation we need two series of data, not just two numbers. The question then is how to generate these two series of data. We need some random process for this. This random process differs between the design-based and model-based approach.

In the design-based approach the random process is the random selection of sampling units, whereas in the model-based approach randomness is introduced via the statistical model of the spatial variation (Table 1.1). So, the design-based approach requires probability sampling, i.e. random sampling, using a random number generator, in such way that all population units have a positive probability of being included in the sample, and that these inclusion probabilities are known for at least the selected population units(?). A probability sampling design can be used to generate an infinite number of samples in theory, although in practical applications only one is selected for sampling.

The spatial variation model used in the model-based approach contains two terms, one for the mean (deterministic part), and an error, with a specified probability distribution. For instance, Equation (22.2) in Chapter 22 describes the model used in ordinary kriging. This model can be used to simulate an infinite number of spatial populations. All these populations together are referred to as a superpopulation (?). Depending on the model of spatial variation, the simulated populations may show spatial structure, because the mean is a function of covariates, as in kriging with an external drift, and/or when the errors are spatially autocorrelated. A superpopulation is a construct, the populations do not exist in real world. The populations are similar, but not identical. For instance, the mean differs among the populations. The expectation of the popu-

lation mean, i.e. the average over all possible simulated populations, equals the superpopulation mean, commonly referred to as the model mean, parameter μ in Equation (22.2). The variance also differs among the populations. Contrary to the mean, the average of the population variance over all populations generally is not equal to the model variance, parameter σ^2 in Equation (22.2), but smaller. I will come to this later. The differences between the simulated spatial populations (see bottom row of Figure 27.1, illustrate our uncertainty about the spatial variation of the study variable in the population that is sampled or will be sampled.

In the design-based approach only one population is considered, the one sampled, but all samples that can be generated by a probability sampling design are considered. The top row of Figure 27.1 shows five simple random samples of size ten. The population is the same in all plots. Design-based proponents do not like to consider other populations than the one sampled. Their challenge is to characterize this one population from a probability sample.

On the contrary, in the model-based approach only one sample is considered, but all populations that can be generated with the spatial variation model. Model-based proponents do not like to consider other samples than the one selected. Their challenge is to get most out of the sample that is selected. The bottom row of Figure 27.1 shows a spatial coverage sample, superimposed on five different populations simulated with an ordinary kriging model, using a spherical semivariogram with a nugget of 0.1, partial sill of 0.6 and a range of 75 m. Note that in the model-based approach there is no need to select a probability sample (see Table 1.1), there are no requirements on how the units are selected.

As stressed by ? and ? both approaches have their strengths and weaknesses. Broadly speaking, the design-based approach is most appropriate if interest is in the population mean (toal proportion) or the population means (totals, proportions) of a restricted number of subpopulations (subareas). The model-based approach is most appropriate if our aim is to map the study variable. Further, the strength of the design-based approach is the strict validity of the estimates. Validity means that an objective assessment of the uncertainty of the estimator is warranted, and that the coverage of confidence intervals is (almost) correct, provided that the sample is large enough to assume an approximately normal distribution of the estimator and design-unbiasedness of the variance estimator (?). The strength of the model-based approach is efficiency, i.e. more precise estimates of the (sub)population mean given the sample size, provided

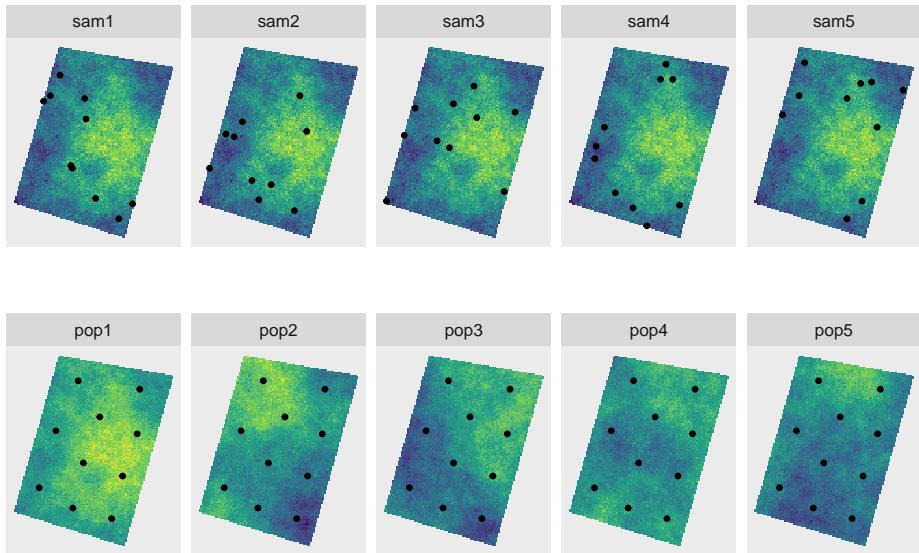


Figure 27.1: Random process considered in the design-based (top row) and model-based approach (bottom row). In the design-based approach only the sampled population is considered, but all samples that can be generated by the sampling design. In the model-based approach all populations that can be generated by the model are considered, but only the sample that is selected.

that a reasonably good model is used. So, if validity is more important than efficiency, the design-based approach is the best choice; in the reverse case, the model-based approach is preferable. For further reading I recommend ? and ?.

27.2 “Identically and independently distributed” (i.i.d.)

In a recent review paper on spatial sampling by ? there is a section with the caption ‘Sampling of i.i.d populations’. Here i.i.d. stands for “identically and independently distributed”. In this section of ? we can read: “In SRS (simple random sampling) it is assumed that the population is independent and identically distributed”. This is one of the old misconceptions revitalized by this review paper. I will make clear that in statistics i.i.d is not a characteristic of

populations, so the concept of i.i.d. populations does not make sense. The same misconception can be found in ?: “There is considerable literature on sample size estimation, much of which is discussed by Cochran (1977, Chapter 4). This literature, however, is valid for samples of independent data but may not retain its validity for spatial data”. Also according to ? the classical formula for the variance of the estimator of the mean with simple random sampling, $V = \sigma^2/n$, only holds when data are independent. They say: “However in the case of spatial data, although members of the sample are independent by construction, data values that are near to one another in space, are unlikely to be independent because of a fundamental property of attributes in space, which is that they show spatial structure or continuity (spatial autocorrelation)”. According to ? the variance should be approximated by

$$V(\hat{z}) = \frac{\sigma^2 - \overline{\text{Cov}(z_i, z_j)}}{n}, \quad (27.1)$$

with $V(\hat{z})$ the variance of the estimator of the regional mean (mean of spatial population), σ^2 the population variance, n the sample size, and $\overline{\text{Cov}(z_i, z_j)}$ the average autocovariance between all pairs of individuals (i, j) in the population (sampled and unsampled). So according to this formula, ignoring the mean covariance within the population leads to an over-estimation of the variance of the estimator of the mean. In Section 27.4 I will make clear that this formula is incorrect, and that the classical formula is still valid, also for populations showing spatial structure or continuity.

Remarkably, in other publications we can read that the classical formula for the variance of the estimator of the population mean with simple random sampling *underestimates* the true variance for populations showing spatial structure, see for instance ? and ?. The reasoning is that due to the spatial structure, there is less information in the sample data about the population mean. In Section 27.4 I explain that this is also a misconception. Do not get confused by these publications, and stick to the classical formulas which you can find in standard textbooks on sampling theory such as ? and ?, and in Chapter 3.

The concept of independence of random variables is illustrated with a simulation. The top row of Figure 27.2 shows five simple random samples of size two. The two points are repeatedly selected from the same population (showing clear spatial structure), so this top row represents the design-based approach. The bottom row shows two points, not selected randomly and independently, but at

a fixed distance of 10 m. These two points are placed on different populations generated by the model described above, so the bottom row represents the model-based approach.

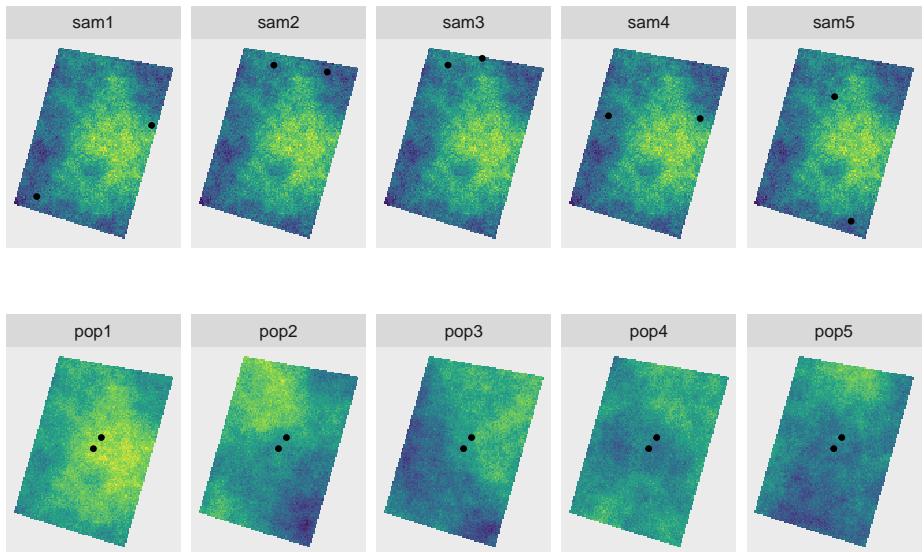


Figure 27.2: Illustration of independence in design-based and model-based approach. The top row shows five samples of two points selected randomly and independently from each other from one population (design-based approach). The bottom row shows two points not selected randomly, at a distance of 10 m from each other, from five model realisations (model-based approach).

The values measured at the two points are plotted against each other in a scatter plot, but now not for just five simple random samples or five populations, but for 1,000 samples and populations (Figure 27.3). As we can see there is no correlation between the two variables generated by the repeated random selection of the two points (design-based), whereas the two variables generated by the repeated simulation of populations (model-based) are correlated.

Instead of two points, we may select two series of probability samples independently from each other, for instance two series of simple random samples (SI) of size 10, or two series of systematic random samples with random origin (SY) with an average size of 10, see Figure 27.4.

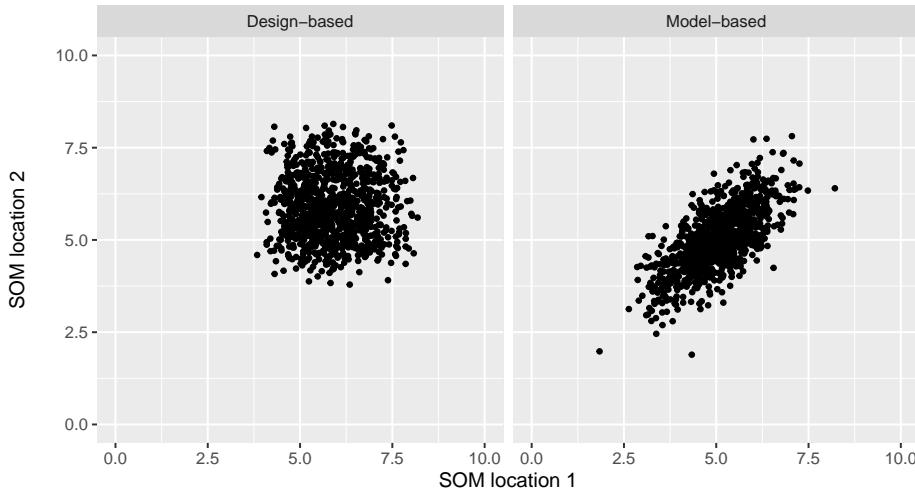


Figure 27.3: Scatter plot of the values at two randomly and independently selected points, 1000 times selected from one population (design-based approach), and at two fixed points with a separation distance of 10 m, selected non-randomly from 1000 model realisations (model-based approach).

Again, if we plot the sample means of pairs of simple random samples and pairs of systematic random samples against each other, we see that the two averages are not correlated (Figure 27.5). Note that the variation of the averages of the systematic random samples is considerably smaller than that of the simple random samples. The sampled population shows spatial structure, and by spreading the sampling units out over the spatial population, the precision of the estimated population mean is increased, see Chapter 5.

This sampling experiment shows that independence is not a characteristic of a population, as stated by ?, but of random variables (in the experiment the values at points, or the sample means) generated by a random process. As the random process differs between the design-based and model-based approach, independence has a different meaning in these two approaches. For that reason, it is imperative to be more specific when using the term independence, by saying that data are *design-independent* or that you *assume* that the data are *model-independent*.

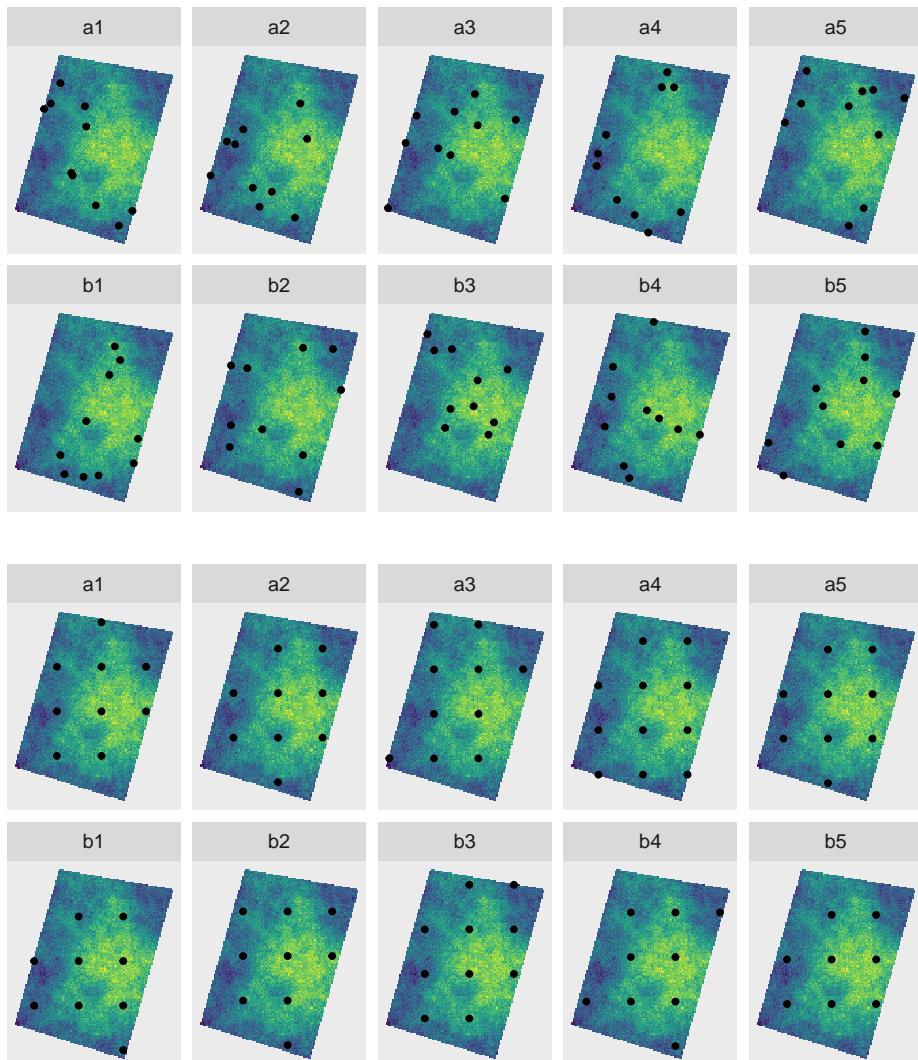


Figure 27.4: Two series (a and b) of simple random samples of ten points (top), and two series of systematic random samples of, on average, ten points (bottom). The samples of series a and b are selected independently from each other.

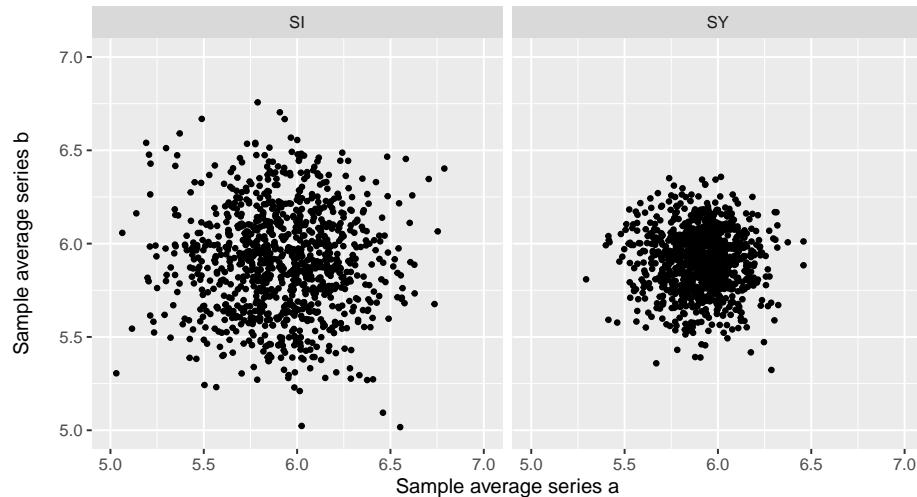


Figure 27.5: Scatterplot of averages of 1000 pairs of simple random samples of ten points, and of averages of 1000 pairs of systematic random samples of ten points on average.

27.3 Bias and variance

Bias and variance are commonly used statistics to quantify the quality of an estimator. Bias quantifies the systematic error, variance the random error of the estimator. Both are defined as expectations. But expectations over the realisations of which random process? Over realisations of a probability sampling design (samples), or realisations of statistical model (populations)? Like independence, it is important to distinguish *design-bias* from *model-bias*, and *design-variance* (commonly referred to as sampling variance) from *model-variance*.

The concept of model-unbiasedness deserves more attention. Figure 27.6 shows a preferential sample from a population simulated by sequential Gaussian simulation with a constant mean of 10 and an exponential semivariogram without nugget, a sill of 5 and a distance parameter of 20. The points are selected by sampling with draw-by-draw selection probabilities proportional to size (pps sampling, Chapter 8), using the square of the simulated values as a size variable. We may have a similar sample that is collected for delineating soil contamination or detecting hot spots of soil bacteria, etc. Many samples are selected at

locations with a large value, few points at locations with a small value. The sample data are used in ordinary kriging (Figure 27.6). The prediction errors are computed by subtracting the kriged map from the simulated population.

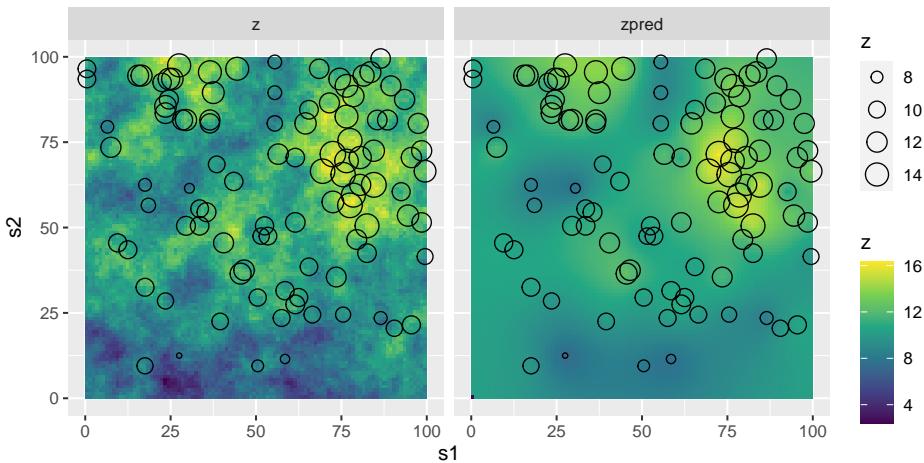


Figure 27.6: Preferential sample (size of open dots is proportional to value of study variable) and ordinary kriging predictions.

Figure 27.7 shows a histogram of the prediction errors. The population mean error equals 0.483, not 0. You may have expected a positive systematic error because of the overrepresentation of locations with large values, but on the other hand, kriging predictions are best linear unbiased predictions (BLUP), so from that point of view, this systematic error might be unexpected. BLUP means that at individual locations the ordinary kriging predictions are unbiased. However, apparently this does not guarantee that the average of the prediction errors, averaged over all population units, equals 0. The reason is that unbiasedness is defined here over all realisations (populations) of the statistical model of spatial variation. So, the U in BLUP stands for model-unbiasedness. For other model realisations, sampled at the same points, we may have much smaller values, leading to a negative mean error of that population. On average, over all populations, the error at any point will be 0, and consequently also the average over all populations of the mean error.

This experiment shows that model-unbiasedness does not protect us against selection bias, i.e. bias due to preferential sampling.

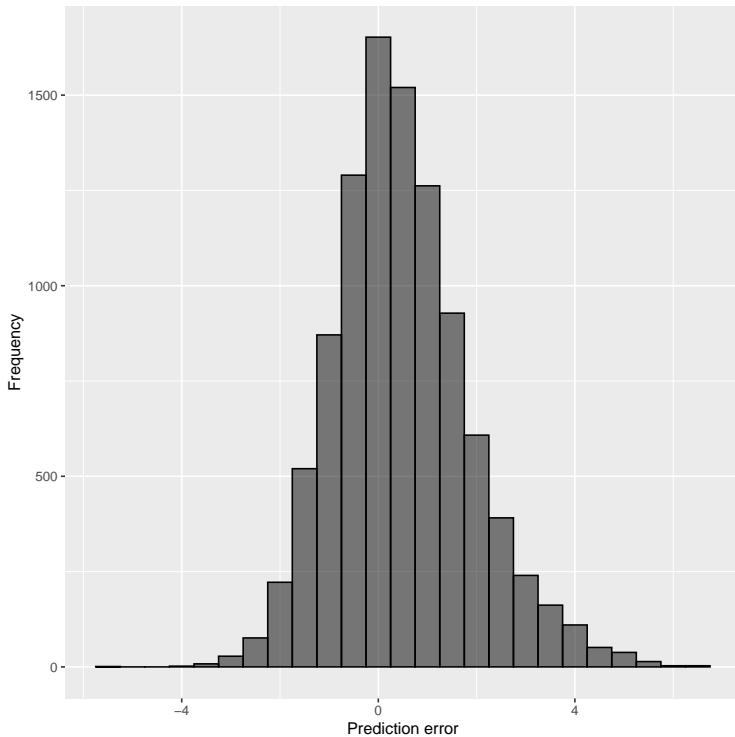


Figure 27.7: Histogram of errors of ordinary kriging predictions from a preferential sample.

27.4 Effective sample size

Another persistent misconception is that when estimating the variance of the estimator of the mean of a spatial population or the correlation of two variables of a population we must account for autocorrelation of the sample data. This misconception occurs, for instance, in [?](#) and in various sections (for instance, Sections 3.5, 10.1 and 11.2) of [?](#). The reasoning is that, due to the spatial autocorrelation in the sample data, there is less information in the data about the parameter of interest, and so the effective sample size is smaller than the actual sample size. An early example of this misconception is Barnes' publication on the required sample size for estimating nonparametric tolerance intervals ([?](#)). [?](#)

showed that a basic probability sampling design like simple random sampling requires fewer sampling points than the model-based sampling design proposed by Barnes.

The misconception is caused by confusing population parameters with model parameters. Recall that the population mean and the model mean are not the same; the model mean μ of Equation (22.2) is the expectation of the population means over all populations that can be simulated with the model. The same holds for the variance of a variable, and the covariance and Pearson correlation coefficient of two variables. All these parameters can be defined as a parameter of a (finite or infinite) population or of random variables generated by a super-population model. Using an effective sample size to quantify the variance of an estimator is perfectly correct for model parameters, but not so for population parameters. For instance, when the correlation coefficient is defined as a population parameter and sampling units are selected by simple random sampling, there is no need to apply the method proposed by ? to correct the p -value in a significance test for the presence of spatial autocorrelation.

I elaborate on this for the mean as the parameter of interest. Suppose a sample is selected in some way (need not be random), and the sample mean is used as an estimator of the model mean. Note that for a model with a constant mean as in Equation (22.2), the sample mean is a model-unbiased estimator of the model mean, but in general not the best linear unbiased estimator (BLUE) of the model mean. If the random variables are model-independent, the variance of the sample mean, used as an estimator of the model mean, can be computed by

$$V(\hat{\mu}) = \frac{\sigma^2}{n} , \quad (27.2)$$

with σ^2 the model-variance of the random variable (see Equation (22.2)). The variance presented in Equation (27.2) necessarily is a model-variance as it quantifies our uncertainty about the model mean, which only exists in the model-based approach. If the random variables are not model-independent, the model-variance of the sample mean can be computed by (?)

$$V(\hat{\mu}) = \frac{\sigma^2}{n} \{1 + (n - 1)\bar{\rho}\} , \quad (27.3)$$

with $\bar{\rho}$ the mean correlation within the sample (the average of the correlation of all pairs of sampling points). The term inside the curly brackets is larger than one, unless $\bar{\rho}$ equals 0. So the variance of the estimator of the model mean with dependent data is larger than when data are independent. The number of independent observations that is equivalent to a spatially autocorrelated data set's sample size n , referred to as the effective sample size, can be computed with (?)

$$n_{\text{eff}} = \frac{n}{\{1 + (n - 1)\bar{\rho}\}} . \quad (27.4)$$

So, if we substitute n_{eff} for n in Equation (27.2), we obtain the variance presented in Equation (27.3). Equation (27.4) is equivalent to Equation 2 in ?. Figure 27.8 shows that the effective sample size decreases sharply with the mean correlation. With a mean correlation of 0 the effective sample size equals the actual sample size, with a mean correlation of one the effective sample size equals one.

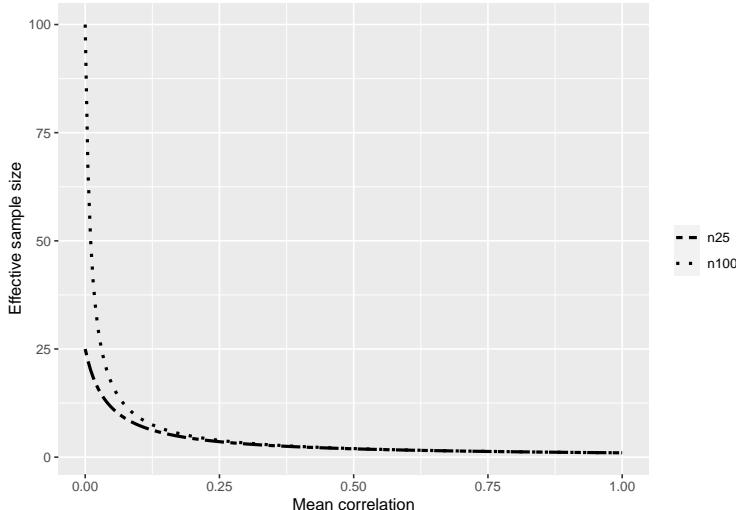


Figure 27.8: Effective sample sizes for samples of size 25 and 100, as a function of the mean correlation within the sample.

To illustrate the difference between the model-variance and design-variance of

a sample mean, I simulated a finite population of 100 units, located at the nodes of a square grid, with a model mean of 10, an exponential semivariogram without nugget, an effective range of three times the distance between adjacent population units, and a sill of one (Figure 27.9). The model-variance of the average of a simple random sample *without replacement* of size n is computed using Equation (27.3), as well as the design-variance of the sample mean, used as an estimate of the population mean, computed by (see Equation (3.11))

$$V(\hat{z}) = \left(1 - \frac{n}{N}\right) \frac{S^2}{n}, \quad (27.5)$$

with N the total number of population units ($N = 100$). This is done for a range of sample sizes: $n = 10, 11, \dots, 100$. Note that for $n < 100$ the model-variance of the sample mean for a given n , differs between samples. For samples showing strong spatial clustering, the mean correlation is relatively large, and consequently the model-variance is relatively large (see Equation (27.3)). There is less information in these samples about the model mean than in samples without spatial clustering of the points. Therefore, to estimate the expectation of the model-variance over repeated simple random sampling for a given n , I selected 200 simple random samples of that size n , and I averaged the 200 model-variances. Figure 27.10 shows the result. Both the model-variance and the design-variance of the sample mean decrease with the sample size. For all sample sizes the model-variance is larger than the design-variance. The design-variance goes to 0, for $n = 100$ (see Equation (27.5)), whereas the model-variance for $n = 100$ equals 0.0509. This can be explained as follows. Although with $n = 100$ we know the population mean without error, this population mean is only an estimate of the model mean. Recall that the model mean is the expectation of the population mean over all realisations of the model.

In Figure 27.11 we can see that the population mean shows considerable variation. The variance of 10,000 simulated population means equals 0.0513, which is nearly equal to the value of 0.0509 for the model-variance computed with Equation (27.3).

In observational research I cannot think of situations in which interest is in estimation of the mean of a superpopulation model. This in contrast to experimental research. In experimental research we are interested in the effects of treatments, think for instance of the effects of different types of soil tillage on the soil carbon stock. These treatment effects are quantified by different model means. Also, in time-series analysis of data collected in observational studies we

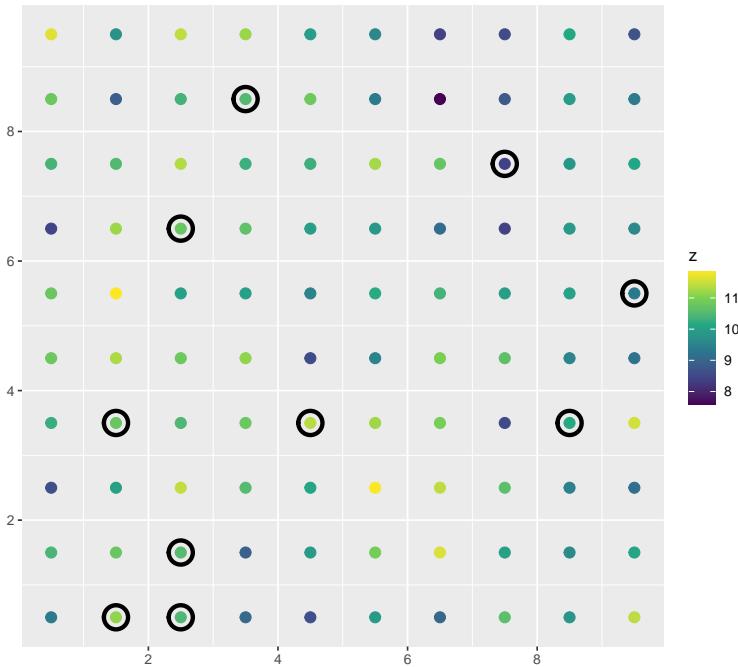


Figure 27.9: Simple random sample without replacement of ten points from a finite population simulated with a model with a model mean of 10, model-variance of 1 and an exponential semivariogram (without nugget) with a distance parameter equal to the distance between neighbours (effective range is three times this distance). The mean correlation within the sample equals 0.135, and the model-variance of the estimator of the model mean equals 0.222.

might be more interested in the model mean than in the mean over a bounded period of time.

Now let us return to Equation (27.1). What is wrong with this variance estimator? Where \bar{z} confused the population mean and the model mean, $\hat{\sigma}^2$ confused the population variance with the sill (a priori variance) of the random process that has generated the population (\bar{z}). The parameter σ^2 in their formula is defined as the population variance, and in doing so the variance estimator is clearly wrong. However, if we define σ^2 in this formula as the sill, the formula makes more sense, but even then, the equation is not fully correct. The

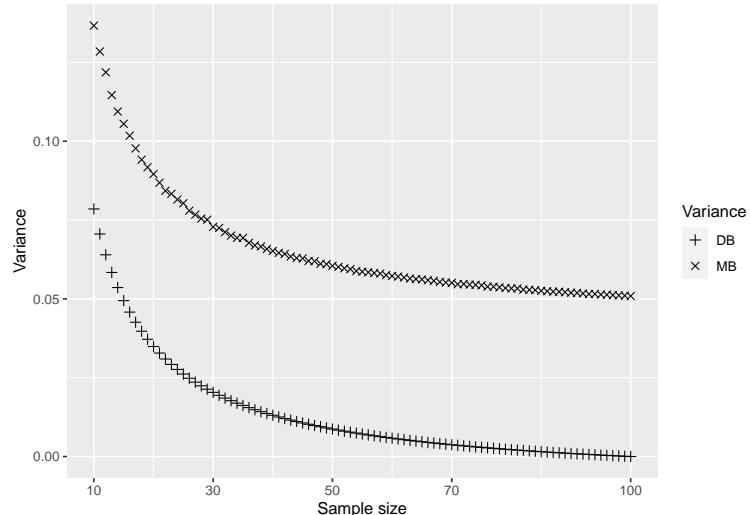


Figure 27.10: Model variance (MB) and design-variance (DB) of the average of simple random sample without replacement as a function of the sample size.

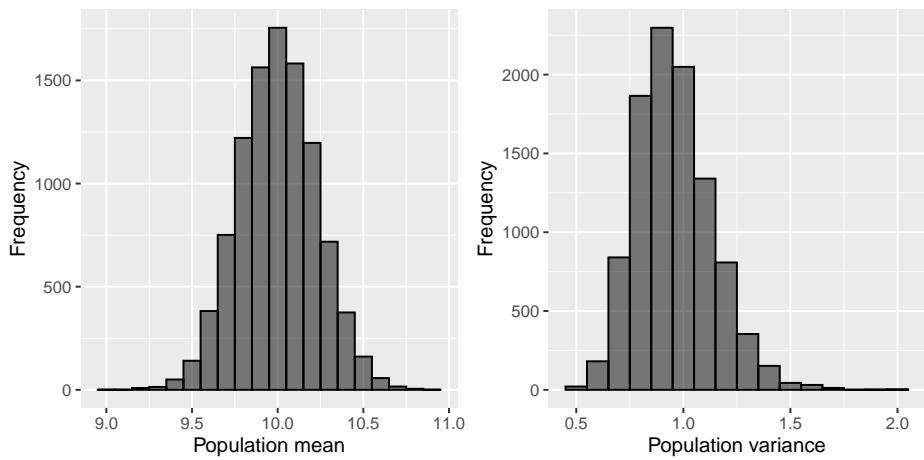


Figure 27.11: Histograms of means and variances of 10,000 simulated populations.

variance computed with this equation is not the design-variance of the average of a simple random sample selected from the sampled population, but the *expectation* of this design-variance over all realisations of the model. So, it is a model-based prediction of the design-variance of the estimator of the population mean, estimated from a simple random sample, see Chapter 13. For the population actually sampled, the design-variance is either smaller or larger than this expectation. Figure 27.11 shows that there is considerable variation in the population variance among the 10,000 populations simulated with the model. Consequently, for an individual population the variance of the estimator of the population mean, estimated from a simple random sample, can largely differ from the model-expectation of this variance. Do not use Equation (27.1) for estimating the design-variance of the estimator of the population mean, but simply use Equation (27.5) (for simple random sampling with replacement and simple random sampling of infinite populations the term $(1 - n/N)$ can be dropped). Equation (27.1) is only relevant for comparing simple random sampling under a variety of models of spatial variation (? , ?).

27.5 Exploiting spatial structure in design-based approach

Another misconception is that in the design-based approach the possibilities of exploiting our knowledge about the spatial structure of the study variable are limited, because the sampling units are selected randomly. This would indeed be a very serious drawback, but happily enough, this is not true. There are various ways of utilizing this knowledge. Our knowledge about the spatial structure can be used either at the stage of designing the sample, and/or at the stage of the statistical inference once the data are collected (Table 27.1).

I distinguish the situation in which maps of covariates are available from the situation in which such maps are lacking. In the first situation, the covariate maps can be used, for instance, to stratify the population (Chapter 4). With a quantitative covariate, optimal stratification methods are available. Other options are, for instance, pps sampling (Chapter 8), and balanced sampling and well-spread sampling in covariate space with the local pivotal method (Chapter 9). At the inference stage the covariate maps can be used in a model-assisted approach, using, for instance, a linear regression model to increase the precision of the design-based estimator (Chapter 10, Section 27.6).

Table 27.1: Exploiting knowledge about the spatial structure of the study variable in the design-based approach. LPM: local pivotal method.

Stage	Covariates available	No covariates
Sampling	Stratified random sampling	Systematic random sampling
	pps sampling	Compact geographical stratification
	Balanced sampling	Geographical spreading with LPM
	Covariate-space spreading with LPM	GRTS sampling
Inference	Model-assisted: regression model	Model-assisted: spatial spline

If no covariate maps are available, we may anticipate the presence of spatial structure by spreading out the sampling units throughout the study area. This spreading can be done in many ways, for instance by systematic random sampling (Chapter 5), compact geographical stratification (Section 4.6), well-spread sampling in geographical space with the local pivotal method (Section 9.2.1), and generalized random tessellation stratified sampling (Section 9.2.2). At the inference stage, again a model-assisted approach can be advantageous, using the spatial coordinates in a regression model.

27.6 Model-assisted versus model-dependent

In this section the difference between the model-assisted approach and model-based approach is explained. The model-assisted approach is a hybrid approach in between the design-based and model-based approach. It tries to build the strength of the model-based approach, a potential increase of the accuracy of estimates, into the design-based approach. As in the design-based approach, sampling units are selected by probability sampling, and consequently bias and variance are defined as design-bias and design-variance (Table 27.2). As in the model-based approach, a superpopulation model is used. However, the role of this model in the two approaches is fundamentally different. In both approaches we assume that the population of interest is a realisation of the superpopulation model. However, as explained above, in the model-based approach, the statistical properties of the estimators (predictors), such as bias and variance are defined over all possible realisations of the model (Table 27.2). So unbiasedness and minimum variance of an estimator (predictor) means *model*-unbiasedness and minimum *model*-variance. On the contrary, in the model-assisted approach the model is used to derive an efficient estimator (Chapter 10). To stress its different role in the model-assisted approach, the model is referred to as a working

Table 27.2: Three statistical approaches for sampling and inference.

Approach	Sampling	Inference	Regression coefficients	Quality criteria
Design-based	Prob. sampling	Design-based	No model	Design-bias, design-variance
Model-assisted	Prob. sampling	Model-assisted	Population par.	Design-bias, design-variance
Model-based	No requirement	Model-depend.	Superpop. par.	Model-bias, model-variance

model.

An important property of model-assisted estimators is that, if a poor working model is used (our assumptions about how our population is generated are incorrect), then for moderate sample sizes the results are still valid, i.e. the empirical coverage rate of a model-assisted estimate of the confidence interval of the population mean still is approximately equal to the nominal coverage rate. This is because the mismatch of the superpopulation model and the applied model-assisted estimator results in a large design-variance of the estimator of the population mean. This is illustrated with a simulation study, in which I compare the effect of using a correct vs. an incorrect model in estimation.

A population is simulated with a simple linear regression model with an intercept of 15 ($\beta_0 = 15$), a slope coefficient of 0.5 ($\beta_1 = 0.5$), and a constant residual standard deviation of 2 ($\sigma_\epsilon = 2$). This is done by first simulating a population with covariate values with a model mean of 20 ($\mu(x) = 20$), using an exponential semivariogram without nugget, a sill variance of 25, and a distance parameter of 20 distance units. This field with covariate values is then linearly transformed using the above mentioned regression coefficients. Finally, ‘white noise’ is added by drawing independently for each population unit a random number from a normal distribution with zero mean and a standard deviation of 2 (Figure 27.12).

The population mean of the study variable equals 25.052, which is pretty close to the known model mean $\mu(z)$: $\beta_0 + \beta_1\mu(x) = 15 + 0.5 \cdot 20$.

Figure 27.13 shows a scatterplot for all population units. The Pearson correlation coefficient equals 0.745. Two models are fitted to the exhaustive scatter plot, a simple linear regression model and a ratio model. The ratio model assumes that the intercept β_0 equals 0, and that the residual variance is proportional to the covariate values: $\sigma_\epsilon^2 \propto x$. The population fit of the coefficients of the simple linear regression model are 14.9989 and 0.4997, which are very close to the model regression coefficients. The fitted ratio model is clearly very poor. The residual standard deviation of the population fit of the ratio model equals

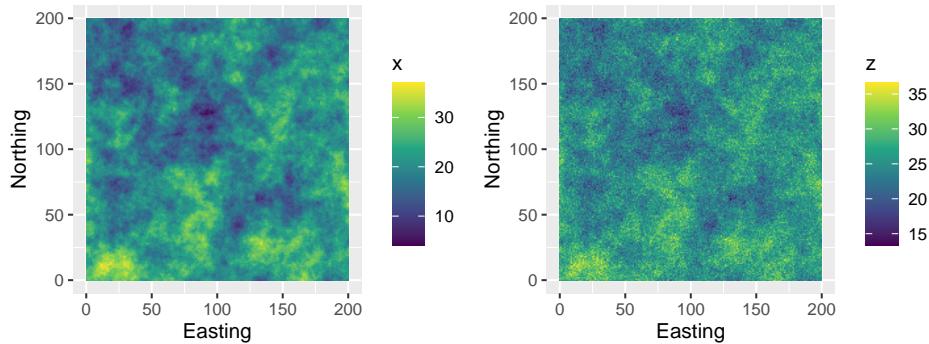


Figure 27.12: Realisation of simple linear regression model.

3.887, which is much larger than 2.001 of the simple linear regression model.

The population mean of the study variable is estimated by selecting 5,000 times a simple random sample of 25 units. Each sample is used to estimate the population mean by two model-assisted estimators, the simple regression estimator and the ratio estimator (Chapter 10). The first estimator correctly assumes that the population is a realisation of a simple linear regression model, whereas the latter incorrectly assumes that it is a realisation of a ratio model. For each sample the standard error of the two estimators are estimated as well, which is used to compute a 95% confidence interval of the population mean. Then the empirical coverage rate is computed, i.e. the proportion of samples for which the population mean is inside the 95% confidence interval. Ideally this empirical coverage rate is equal to the nominal coverage rate of 0.95.

The coverage rates of the simple regression estimator and the ratio estimator equal 0.931 and 0.948, respectively. Both coverage rates are very close to the nominal coverage rate of 0.95. So despite that the ratio estimator assumes an improper superpopulation model, the estimated confidence interval is still valid. The price we pay for the invalid model assumption is not an overestimated coverage rate of a confidence interval, but an increased standard error of the estimated population mean. The average over the 5,000 samples of the estimated standard error of the regression estimator equals 0.387, whereas that of the ratio estimator equals 0.772. The larger standard error of the ratio estimator leads to wider confidence intervals, which explains that the coverage rate is still correct.

This sampling experiment is now repeated for samples sizes $n = 10, 25, 50, 100$

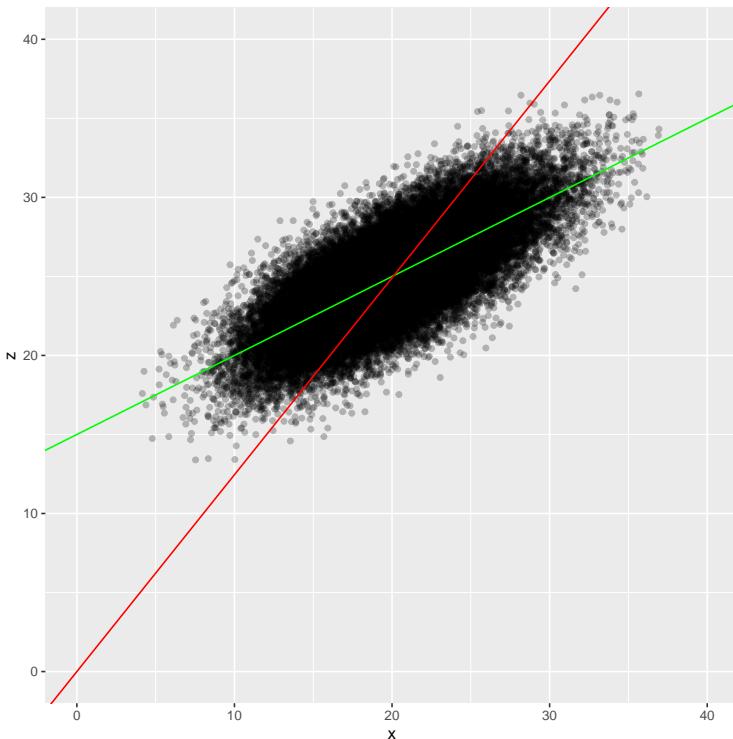


Figure 27.13: Exhaustive scatter plot of simulated population, with population fit of simple linear regression model (green line), and ratio model (red line) fitted with weights inversely proportional to the covariate.

and for confidence levels $1 - \alpha = 0.01, 0.02, \dots, 0.99$.

Figures 27.14 and 27.15 show that the empirical coverage rates are close to the nominal coverage rate, for all four sample sizes, both estimators, and all confidence levels. For the regression estimator and $n = 10$ the empirical coverage rate is somewhat too small. This is because the standard error of the regression estimator is slightly underestimated at this sample size. The average of the estimated standard errors (square root of estimated variance of regression estimator) equals 0.609, which is somewhat smaller than the standard deviation of the 5,000 regression estimates of 0.678 (Table 27.3). The relative bias, computed as

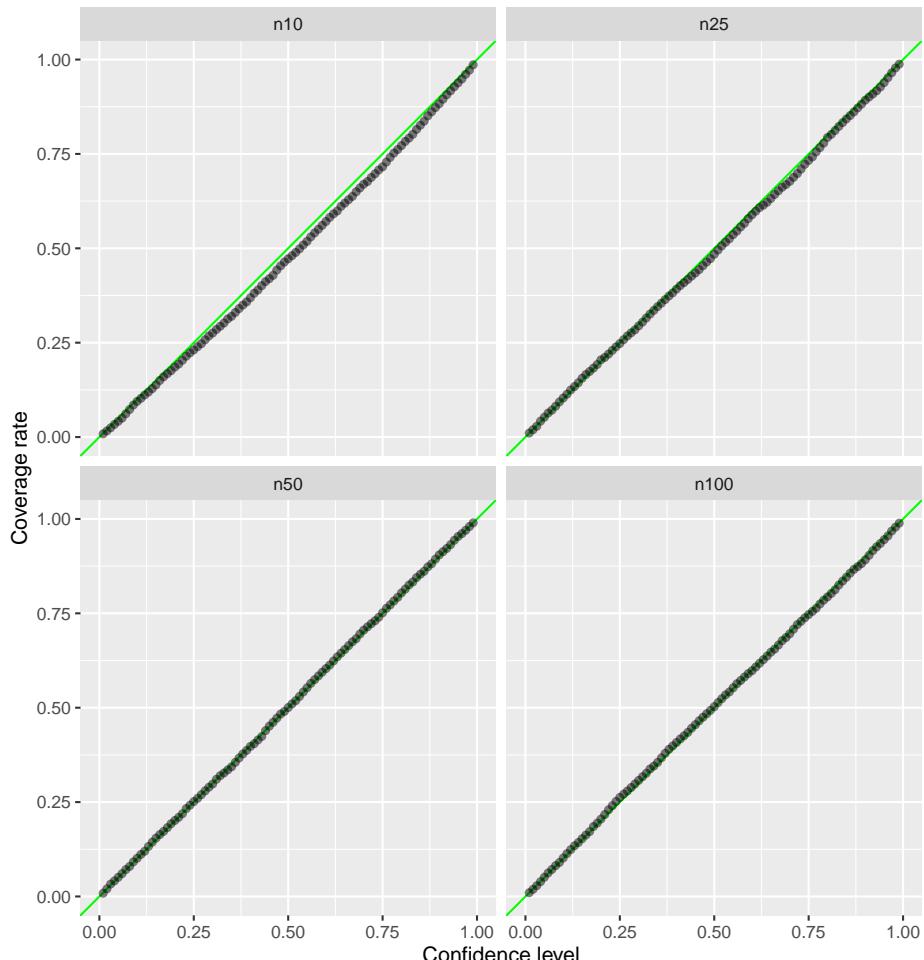


Figure 27.14: Empirical versus nominal coverage rates of confidence intervals for the population mean, estimated by the simple regression estimator, for sample sizes 10, 25, 50 and 100.

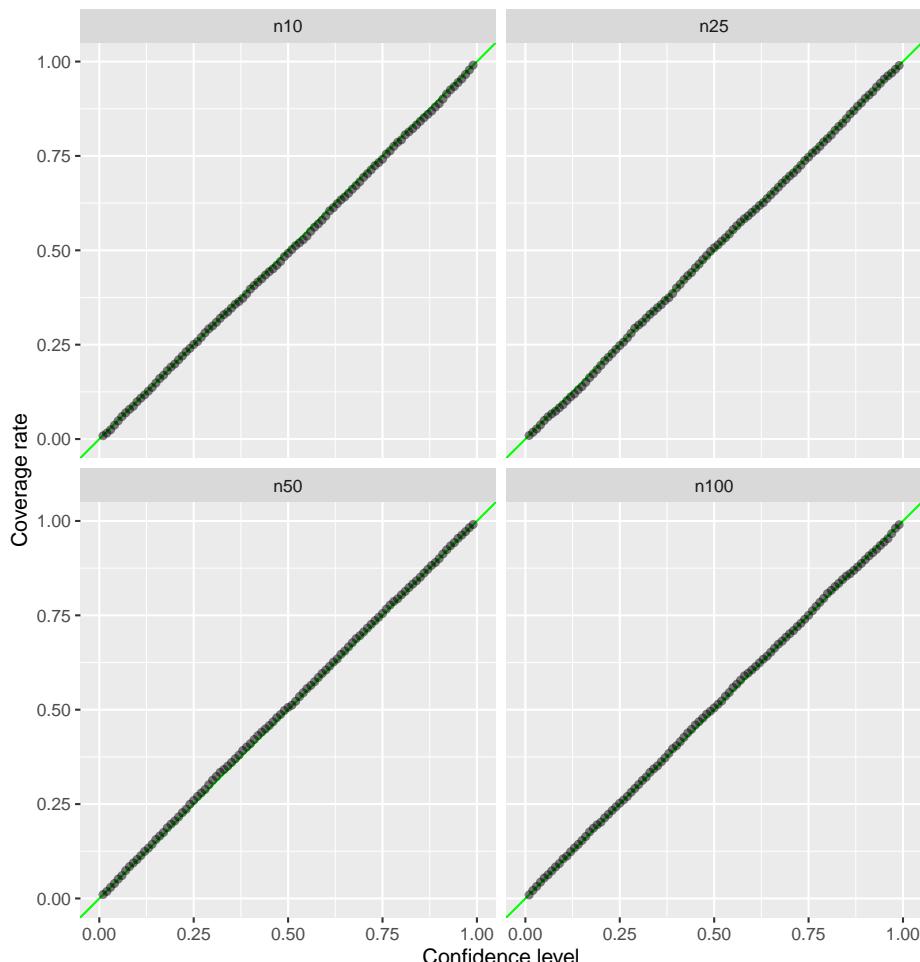


Figure 27.15: Empirical versus nominal coverage rates of confidence intervals for the population mean, estimated by the ratio estimator, for sample sizes 10, 25, 50 and 100.

Table 27.3: Estimated relative bias of regression estimator, standard deviation of 5000 regression estimates, and average of 5000 estimated standard errors of regression estimator.

Sample size	Bias	Standard deviation	Average standard error
10	0	0.678	0.609
25	0	0.411	0.397
50	0	0.282	0.282
100	0	0.202	0.200

Table 27.4: Estimated relative bias of ratio estimator, standard deviation of 5000 ratio estimates, and average of 5000 estimated standard errors of ratio estimator.

Sample size	Bias	Standard deviation	Average standard error
10	-0.003	1.273	1.206
25	-0.001	0.776	0.769
50	-0.001	0.548	0.549
100	0.000	0.390	0.388

$$bias = \frac{\frac{1}{5000} \sum_{s=1}^{5000} (\hat{z}_s - \bar{z})}{\bar{z}}, \quad (27.6)$$

is about 0 for both estimators and all four sample sizes.

Contrarily, if in the model-based approach a poor superpopulation model is used, then the predictions and the prediction error variances still are model-unbiased, but for the sampled population we may have serious systematic error in the estimated population mean, and the variance of local predictions may be seriously over- or underestimated. For that reason, model-based inference is also referred to as *model-dependent* inference, stressing that we fully rely on the model, and that the validity of the estimates and predictions depends on the quality of the model (?).

Appendix A

Answers to exercises

R scripts of the answers to the exercises are available at the Exercises folder of the github repository of this book.

Introduction to probability sampling

1. No this is not a probability sample because with this implementation the probabilities of selection of the units are unknown.
2. A simple way would be to generate with a random number generator n integers between 1 and N . This can be done in **R** with function `sample.int`. For instance, `sample.int(1000, 10)` selects fully randomly with equal probability 10 integers from the discrete interval $\{1, 2, \dots, 1000\}$.

Simple random sampling

1. The most remarkable difference is the much smaller range of values in the sampling distribution of the estimator of the population mean. This can be explained by the smaller variance of the average of n randomly selected values compared to the variance of an individual randomly selected value. A second difference is that the sampling distribution is more symmetric,

less skewed to the right. This is an illustration of the central limit theorem.

2. The variance (and so the standard deviation) becomes smaller.
3. Then this difference will be very close to 0, showing that the estimator is unbiased.
4. For simple random sampling without replacement (from a finite population) the sampling variance will be smaller. When units are selected with replacement, a unit can be selected more than once. This is inefficient as there is no extra information in the unit that has been selected before.
5. The larger the population size N , the smaller this effect (given the sample size n).
6. The true sampling variance of the estimator of the mean for simple random sampling from an infinite population can be computed with the population variance divided by the sample size: $V(\hat{z}) = S^2(z)/n$.
7. This is because in reality we do not know the values of z for all units in the population, so that we do not know the population variance $S^2(z)$.
8. See SI.R¹. The 90% confidence interval is less wide than the 95% interval because a larger proportion of samples is allowed not to cover the population mean. The estimated standard error of the estimated total underestimates the true standard error because a constant bulk density is used. In reality this bulk density also varies.

Stratified random sampling

1. Strata EA and PA can be merged: their means are about equal.
2. See STSI.R.
3. The proof is as follows: $\sum_N \pi_i = \sum_H \sum_{N_h} \pi_{hi} = \sum_H \sum_{N_h} n_h/N_h = \sum_H n_h = n$.

¹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/SI.R>

4. See STSIcumrootf.R².
5. With at least two points per geostratum, the variance of the estimator of the stratum mean can be estimated without bias by the estimated stratum variance divided by the number of points in that stratum.
6. On average the sampling variance of the estimator of the mean with 40×1 is smaller than with 20×2 points because the geographical spreading will be somewhat better (less spatial clustering).
7. With geostrata of equal size and equal number of sampling points per geostratum, the sampling intensity is equal for all strata, so that the sample mean is an unbiased estimator of the population mean. In formula:
$$\hat{z} = \sum_{h=1}^H w_h \bar{z}_{sh} = \frac{1}{H} \sum_{h=1}^H \bar{z}_{sh} = \bar{z}_s$$
 with \bar{z}_{sh} the average of the sample from stratum h and \bar{z}_s the average of all sampling points.
8. See STSIgeostrata.R³.
 - Collapsing the geostrata on the basis of the measurements of the study variable is not a proper way, as it will lead to a biased estimator of the sampling variance of the estimator of the mean. The estimated stratum variances $\widehat{S}^2(z)$ will be small, and so the estimated sampling variance will underestimate the true sampling variance.
 - I propose to group neighbouring geostrata, i.e. geostrata that are close to each other.
 - The sampling variance is slightly overestimated because we assume that the two (or three) points within a collapsed stratum are selected by simple random sampling, whereas they are selected by stratified random sampling (a collapsed stratum consists of two or three geostrata), and so there is less spatial clustering compared to simple random sampling.

²<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/STSIcumrootf.R>

³<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/STSIgeostrata.R>

9. See STSIgeostrata_composite.R.

- No, with bulking within strata the sampling variance cannot be estimated, because then we cannot estimate the sampling variances of the estimated stratum means, which are needed for estimating the sampling variance of the estimator of the population mean.
- If all aliquots are analyzed separately the estimated population mean is more precise (variance of the estimator of the mean is smaller) because the contribution of the measurement error to the total variance of the estimator of the mean is smaller.
- This does not work because with geostrata of unequal area the mean of a composite sample is a biased estimator of the population mean. All aliquots bulked into a composite get equal weight, but they should get different weights because they do not represent equal fractions of the population.

Systematic random sampling

1. See SY.R⁴.
2. As can be seen on the plot, the spatial coverage of the study area by the two systematic random samples can be quite poor. So I expect that the variance of the estimator of the mean using the data of two systematic random samples of half the expected size, is larger than the variance of the estimator of the mean based on the data of a single systematic random sample.

⁴<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/SY.R>

Cluster random sampling

1. See Cluster.R⁵.
2. I expect that the sampling variance is larger, as the sampling points are more spatially clustered.
3. With two independently selected clusters per stratum the sampling variance of the estimator of the mean can be estimated without bias, as the variance of cluster means within the strata can be estimated from the two cluster means.

Two-stage random sampling

1. See TwoStage.R⁶.
2. With ten psu draws and four ssu's per psu draw (10×4) the expected standard error of the estimated population is smaller than with 4×10 because spatial clustering of the sampling points is less strong.
3. See TwoStage.R⁷.
4. See TwoStage.R⁸.
5. See TwoStage.R⁹.

⁵<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/Cluster.R>

⁶<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/TwoStage.R>

⁷<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/TwoStage.R>

⁸<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/TwoStage.R>

⁹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/TwoStage.R>

-
6. For the first variance component:

$$\begin{aligned} \frac{1}{n} \sum_{j=1}^N p_j \left(\frac{t_j(z)}{p_j} - t(z) \right)^2 &= \frac{1}{n} \sum_{j=1}^N p_j \left(M \frac{t_j(z)}{M_j} - M \bar{z} \right)^2 = \\ \frac{1}{n} \sum_{j=1}^N p_j (M(\bar{z}_j - \bar{z}))^2 \frac{M^2}{n} \sum_{j=1}^N p_j (\bar{z}_j - \bar{z})^2. \end{aligned} \quad (\text{A.1})$$

For the second variance component:

$$\frac{1}{n} \sum_{j=1}^N \frac{M_j^2 S_j^2}{m_j p_j} = \frac{1}{nm} \sum_{j=1}^N \frac{M_j^2 S_j^2}{p_j} = \frac{1}{nm} \sum_{j=1}^N M M_j S_j^2 = \quad (\text{A.2})$$

$$\frac{1}{nm} \sum_{j=1}^N M^2 \frac{M_j}{M} S_j^2 = \frac{M^2}{nm} \sum_{j=1}^N p_j S_j^2. \quad (\text{A.3})$$

Division of both variance components by M^2 yields the variance estimator of the estimated population mean, Equation (7.3).

Sampling with probabilities proportional to size

1. See pps.R¹⁰.
2. No this field should not be included in the poppy area of that sampling unit because this field is outside the target area.
3. Yes, this field must be included in the poppy area of that sampling unit as it is located inside the target area. The target area is the territory of Kandahar, regardless of how an area inside this territory is depicted on the map, as agricultural land or otherwise.

Balanced and well-spread sampling

1. Spatial clustering may lead to a less precise estimate of the population mean. This will happen when the residuals of the regression model are

¹⁰<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/pps.R>

spatially correlated (show spatial structure). This will occur when the spatial variation of the study variable is also determined by covariates or factors that are not used in balancing the sample. If the residuals are not spatially correlated (white noise), spatial clustering does no harm.

2. One advantage is that unequal inclusion probabilities can be used in the LPM design. If the sampling units have unequal size (as in the poppy survey of Kandahar) or if a covariate is available that is linearly related to the study variable (as in the aboveground biomass survey of Eastern Amazonia), sampling efficiency can be increased by sampling with (inclusion) probabilities proportional to size. The only option for random sampling from geostrata is then to select the unit(s) *within geostrata* by pps sampling.

Ratio and regression estimator

1. See RegressionEstimator.R¹¹. In reality we do not have a population fit of the regression coefficients, but these coefficients must be estimated from a sample. The estimated coefficients vary among the samples, which explains that the experimental variance, i.e. the variance of the 10,000 regression estimates obtained by estimating the coefficients from the sample (Sample in Figure A.1) is larger than the variance as computed with the population fit of the regression coefficients (Exhaust in Figure A.1).

The difference between the variance obtained with the population fit and the experimental variance (variance of regression estimator with sample fit of coefficients), as a proportion of the experimental variance, decreases with the sample size. The same holds for the difference between the approximated variance and the experimental variance. Both findings can be explained by the smaller contribution of the variance of the estimated regression coefficients to the variance of the regression estimator with the large sample size. The approximated variance does not account for the uncertainty about the regression coefficients, so that for all three sample sizes this approximated variance is about equal to the variance of the regression estimator as computed with the population fit of the regression coefficients.

¹¹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/RegressionEstimator.R>

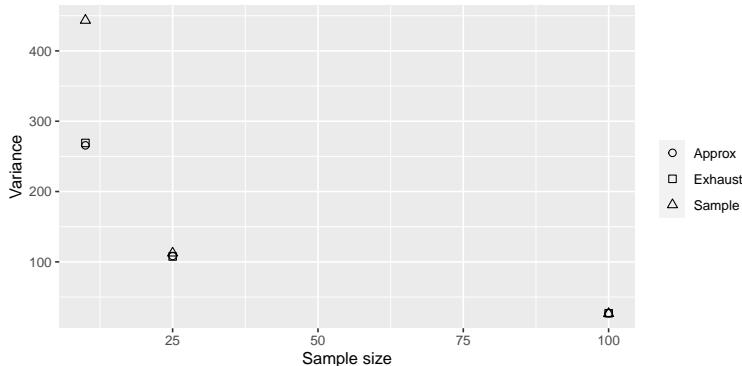


Figure A.1: Variance of regression estimator of mean AGB with population fit of regression coefficients (Exhaust), with sample fit of regression coefficients (Sample), and approximated variance of regression estimator (Approx)

2. See RatioEstimator.R¹². The population fit of the slope coefficient of the homoscedastic model differs from the ratio of the population total of poppy area and population total of agricultural area. For the heteroscedastic model these are equal.

Two-phase random sampling

1. See RegressionEstimator_Twophase.R¹³. The variance of the regression estimator for two-phase sampling is considerably larger than the variance of the usual regression estimator (Figure A.2). The usual regression estimator exploits our knowledge of the population mean of the covariate SWIR2, whereas in two-phase sampling this population mean must be estimated from the first phase sample, introducing additional uncertainty.

The average of the 10,000 approximated variances equals 40.5, which is considerably smaller than the variance of the 10,000 regression estimates for two-phase sampling, which is equal to 51.2.

¹²<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/RatioEstimator.R>

¹³https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/RegressionEstimator_Twophase.R

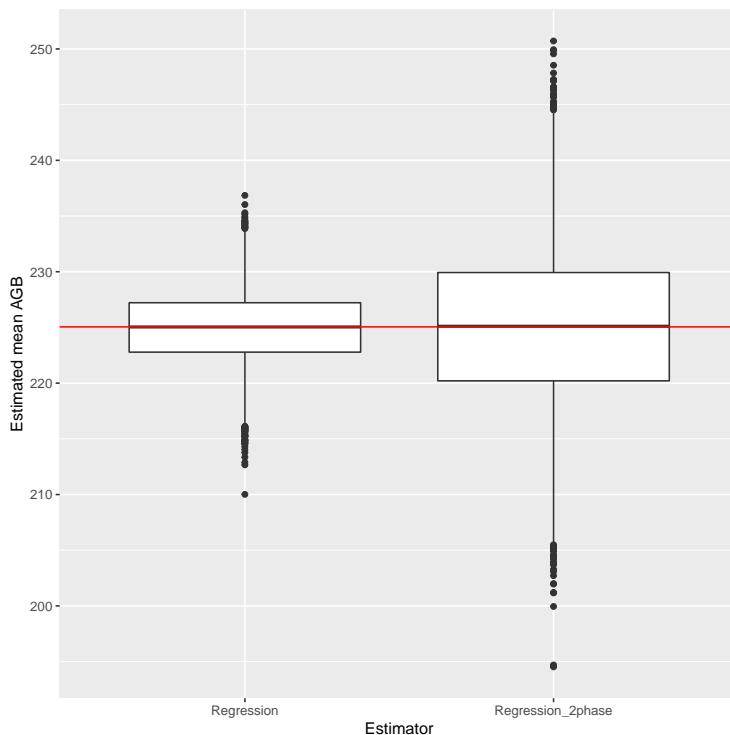


Figure A.2: Sampling distribution of simple regression estimator and simple regression estimator for two-phase sampling.

Required sample size

1. See RequiredSampleSize_CIprop.R¹⁴. The plots show that the required sample size decreases sharply with the length of the confidence interval, and increases with the anticipated proportion.

A prior for the proportion is needed because the standard error of the estimated proportion is a function of the estimated proportion \hat{p} itself: $se(\hat{p}) = \frac{\sqrt{\hat{p}(1-\hat{p})}}{\sqrt{n}}$, so that the width of the confidence interval, computed with the normal approx-

¹⁴https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/RequiredSampleSize_CIprop.R

imation, is also a function of \hat{p} , see Equation (12.11).

For a prior proportion p^* of 0.5 the standard deviation $\sqrt{p^*(1-p^*)}$ is maximum. The closer the prior proportion to zero or one, the smaller the standard error of the estimated proportion, the smaller the required sample size.

2. See RequiredSampleSize_CIprop.R¹⁵. There is no need to compute the required sample size for prior proportions > 0.5 , as this required sample size is symmetric. For instance, the required sample size for $p^* = 0.7$ is equal to the required sample size for $p^* = 0.3$.

Model-based prediction of sampling variance of estimated mean

1. See MBSamplingVarSI_VariogramwithNugget.R¹⁶. The predicted sampling variance is slightly larger compared to the predicted sampling variance obtained with the semivariogram without nugget (and the same sill and range), because 50% of the spatial variation is not spatially structured, so that in sampling we can profit less from spatial correlation.
2. See first part of MBRequiredSampleSize_SIandSY.R¹⁷.
3. See second part of MBRequiredSampleSize_SIandSY.R¹⁸. The model-based prediction of the required sample size for simple random sampling is 34, and for systematic random sampling 13. The design effect at a sample size of 34 equals 0.185. The design effect decreases with the sample size, i.e. the ratio of the variance with systematic random sampling to the variance with simple random sampling becomes smaller. This is because the larger the sample size, the more we profit from the spatial correlation.

¹⁵https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/RequiredSampleSize_CIprop.R

¹⁶https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBSamplingVarSI_VariogramwithNugget.R

¹⁷https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBRequiredSampleSize_SIandSY.R

¹⁸https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBRequiredSampleSize_SIandSY.R

Regular grid sampling

1. See SquareGrid.R¹⁹.

Spatial coverage sampling

1. The optimal spatial coverage sample (optimal in terms of MSSD) consists of the four points in the center of the four subsquares of equal size.
2. If we are also interested in the accuracy of the estimated plot mean, the sampling units should be selected randomly from the subsquares (strata). Preferably at least two points should then be selected from the strata, see Section 4.6.
3. See SpatialCoverageCircularPlot.R²⁰.



Figure A.3: Spatial coverage samples of five and six points in a circular plot used as sampling unit.

4. Bias can be avoided by constructing strata of equal size. Note that in this case we cannot use function `spsample` to select the centers of these geostrata. These centers must be computed by hand.

¹⁹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/SquareGrid.R>

²⁰<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/SpatialCoverageCircularPlot.R>

5. See `SpatialInfill.R`²¹.

Covariate space coverage sampling

1. See `CovariateSpaceCoverageSample.R`²².
2. See second part of `CovariateSpaceCoverageSample.R`²³.

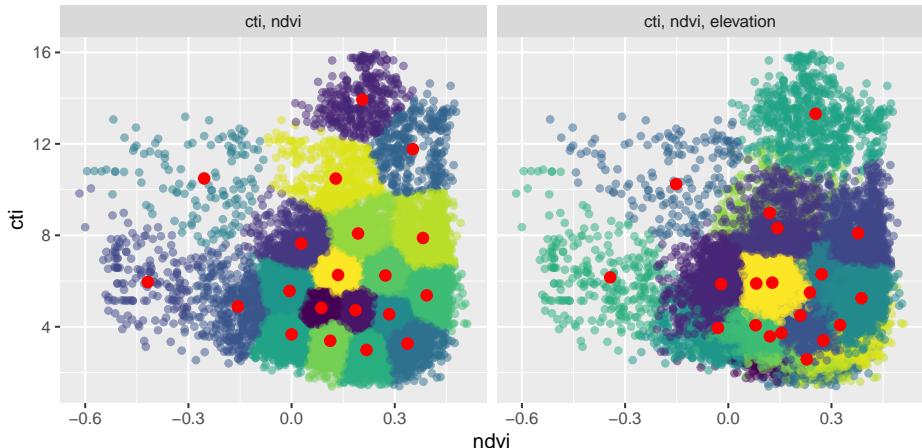


Figure A.4: Covariate space coverage samples, plotted in a scatter diagram of `cti` against `ndvi`. Two samples are selected, one using `cti` and `ndvi` as covariates in clustering, and one using `cti`, `ndvi` and `elevation` as clustering variables.

When using only two covariates (`cti` and `ndvi`), in a two-dimensional scatter diagram the sampling points are at the center of clearly visible clusters (Figure A.4). Plotting the clusters in a two-dimensional scatter diagram yields pure clusters: a cluster does not contain any unit of a different cluster. However when more than two covariates are used in k -means clustering, in a two-dimensional scatter diagram the clusters are less clear because the multidimensional clusters

²¹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/SpatialInfill.R>

²²<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/CovariateSpaceCoverage.R>

²³<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/CovariateSpaceCoverage.R>

are projected on a two-dimensional plane. In the case of the three covariates cti, ndvi, and elevation-m actually there is a third axis perpendicular to plane spanned by the covariates cti and ndvi. This also explains that with more than two covariates a sampling location (cluster center) can be quite close to another sampling location, after projection on a two-dimensional plane.

Conditioned Latin hypercube sampling

1. See cLHS.R²⁴.
 - Spatial coverage is improved by using the spatial coordinates as covariates, but it is not optimal in terms of mean squared shortest distance (MSSD).
 - It may happen that not all marginal strata of s_1 and s_2 are sampled. Even when all these marginal strata are sampled, this does not guarantee a perfect spatial coverage.
 - There are two unsampled marginal strata, and two marginal strata with two sampling locations. So the sum of the absolute values of the marginal stratum sample sizes minus 1 equals four. The minimised value (4.28) is slightly larger due to the contribution of O3 to the criterion.

Model-based optimisation of the grid spacing

1. See MBGridspacing_QOKV.Rmd²⁵. For P95 the tolerable grid spacing is < 6 km, for P80 about 8.5 km and for P95 about 9 km (Figure A.5).
2. See MBGridspacing_Sensitivity.Rmd²⁶. Increasing the nugget by 5% and decreasing the range by 5% yields a tolerable grid spacing that is smaller than with the original semivariogram (Figure A.6). The tolerable grid spacings for a mean kriging variance of 0.85 are 10.6, 8.9 and 7.4 km for the original semivariogram, the semivariogram with increased nugget,

²⁴<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/cLHS.R>

²⁵https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBGridspacing_QOKV.Rmd

²⁶https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBGridspacing_MaxOKV.Rmd

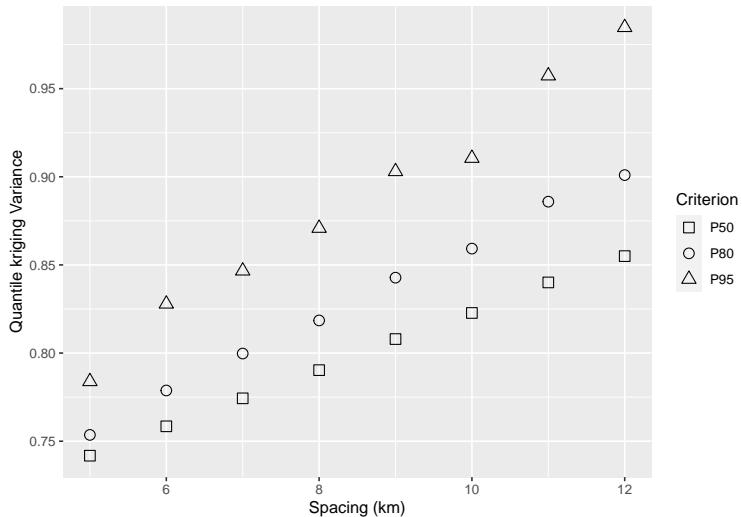


Figure A.5: Three quantiles of the ordinary kriging variance of predicted SOM in three woredas in Ethiopia, as a function of the grid spacing.

and the semivariogram with the smaller range, respectively, leading to a required expected sample size of 97, 137 and 200 points.

3. This variation can be explained by the random sample size (for a given spacing the number of points of a randomly selected grid inside the study area is not fixed but varies), and partly because the covariate values at the grid points vary, so that also the variance of the estimator of the mean differs among grid samples.
4. See MBGridspacing_MKEDV.Rmd²⁷. The tolerable grid spacing for a mean kriging variance is 82 m.

²⁷https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBGridspacing_MKEDV.Rmd

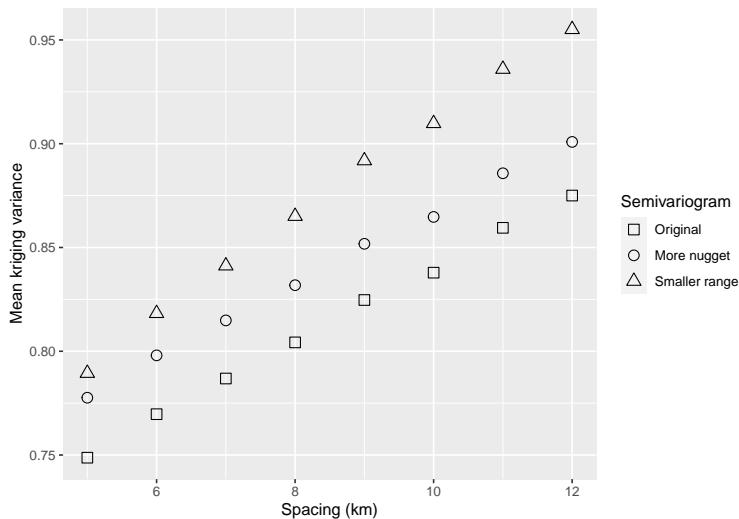


Figure A.6: Mean ordinary kriging variance predicted SOM in three woredas of Ethiopia, as a function of grid spacing for three semivariograms.

Model-based optimisation of sampling pattern

1. See MBSSampleSquare_OK.Rmd²⁸. The optimised sample (Figure A.7) is most likely not the global optimum. The spatial pattern is somewhat irregular. I expect the optimal sampling locations to be close to the centers of the subsquares.
2. See MBSSample_QOKV.Rmd²⁹.

Figure A.8 shows the optimised sample pattern. Compared with the optimised sample pattern using the *mean* ordinary kriging variance as a minimisation criterion (Figure 24.3), the sampling locations are pushed more to the border of the study area. This is because with, for instance, a spatial coverage sample, near the border the kriging variances are the largest. By pushing sampling locations towards the border, the kriging variances in this border zone are strongly

²⁸https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBSSampleSquare_OK.Rmd

²⁹<https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBSSampleQOKV.Rmd>

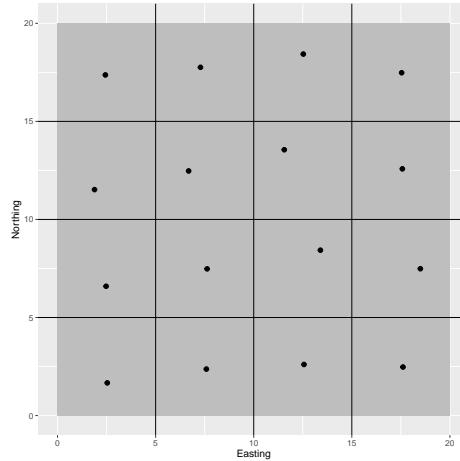


Figure A.7: Optimised coordinates of sixteen sampling points for ordinary kriging.

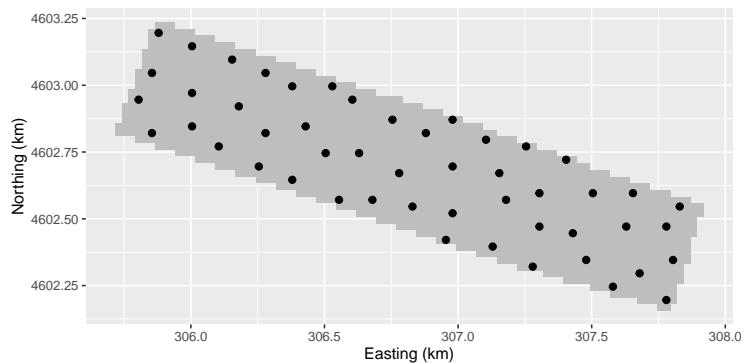


Figure A.8: Optimized sample pattern of 50 points on the Cotton Research Farm, using the P90 of ordinary kriging predictions of lnECE as a minimisation criterion.

reduced.

3. See MBSampleSquare_KED.Rmd³⁰. Figure A.9 shows the optimised sample patterns with the three semivariograms.
 - With zero nugget and a (partial) sill of 2 the sampling points are well spread throughout the area.
 - With a nugget of 1.5 and a partial sill of 0.5 the sampling points are pushed toward the left and right side of the square. With this residual semivariogram the contribution of the variance of the predictor of the mean (as a proportion) to the total kriging variance is larger than with the previous semivariogram. By shifting the sampling points towards the left and right side of the square this contribution becomes smaller. At the same time the variance of the interpolation error increases as the spatial coverage becomes worse. The optimised sample is the right balance of these two variance components.
 - With a pure nugget semivariogram all sampling points are at the left and right side of the square. This is because with a pure nugget semivariogram the variance of the interpolation error is independent of the locations (the variance equals the nugget variance everywhere), while the variance of the predictor of the mean is minimal for this sample.

Sampling for estimating the semivariogram

1. See NestedSampling_v1.R³¹.
2. See SI_PointPairs.R³². With the seed I used (314) the variance of the estimator of the range parameter is much smaller compared to that obtained with the larger separation distances.

³⁰https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBSampleSquare_KED.Rmd

³¹https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/NestedSampling_v1.R

³²https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/SI_PointPairs.R

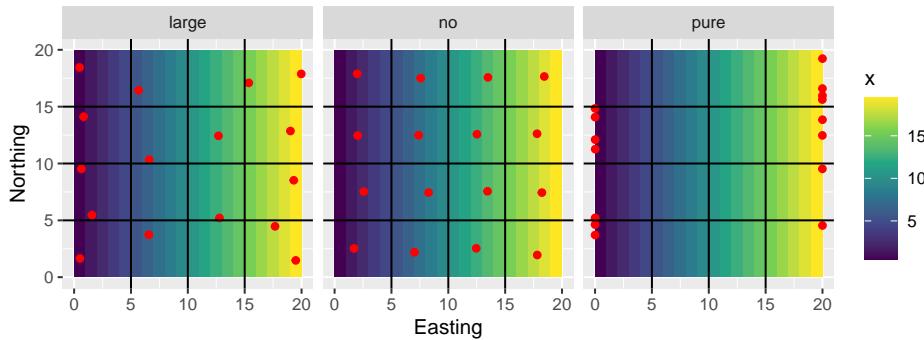


Figure A.9: Effect of size of nugget on optimised coordinates of sixteen points for kriging with an external drift.

3. See MBSample_SSA_logdet.R³³. Figure A.10 shows the optimised sample pattern.
4. See MBSample_SSA_varkrigvar.R³⁴. Figure A.11 shows the optimised sample pattern.

Sampling for validation

1. I am not certain about that, because the computed MSE's are estimates of the population MSE's only, and I am uncertain about both population MSE's.
2. The standard errors of the estimated ME's are large when related to the estimated ME's, so my guess is that we do not have enough evidence against the hypothesis that there is no systematic error.
3. Both standard errors are large compared to the difference in MSE's, so maybe there is no significant difference. However we must be careful, because the variance of the difference in MSE's cannot be computed as the sum of the variances of estimated MSE's. This is because the two

³³https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBSample_SSA_logdet.R

³⁴https://git.wageningenur.nl/brus003/spatialsamplingwithr/tree/master/Exercises/MBSample_SSA_varkrigvar.R

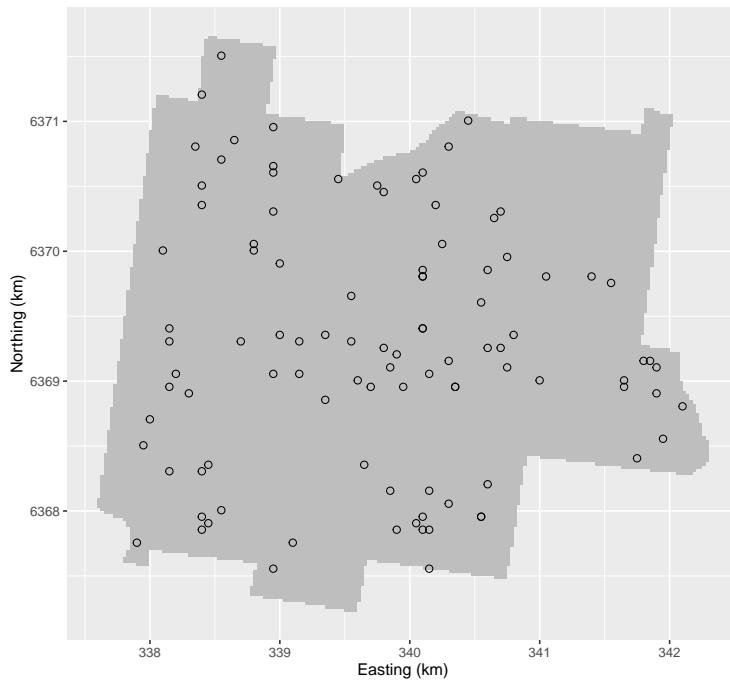


Figure A.10: Model-based sample for estimating the semivariogram, using logdet as a minimisation criterion. The prior semivariogram used in optimising the sample pattern is an exponential semivariogram with a range of 200 m and a ratio of spatial dependence of 0.5.

predictions errors at the same location are correlated, so the covariance must be subtracted from the sum of the variances to obtain the variance of the estimator of the difference in MSE's.

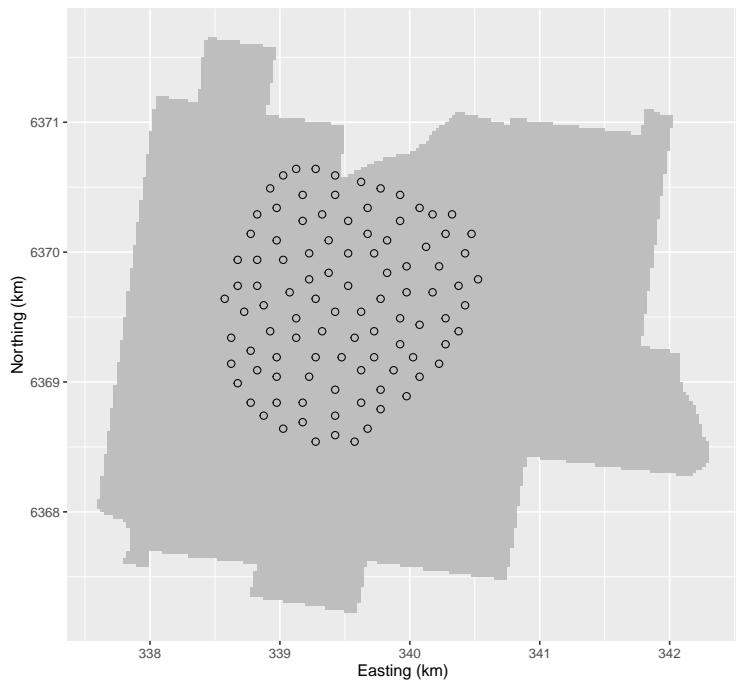


Figure A.11: Model-based sample for estimating the semivariogram, using the variance of the kriging variance (VKV) as a minimisation criterion. The prior semivariogram used in optimising the sample pattern is an exponential semivariogram with a range of 200 m and a ratio of spatial dependence of 0.5.

Bibliography

- A. Baccini and S. J. Goetz and W. S. Walker and N. T. Laporte and M. Sun and D. Sulla-Menashe and J. Hackler and P. S. A. Beck and R. Dubayah and M. A. Friedl and S. Samanta and R. A. Houghton (2012). Estimated carbon dioxide emissions from tropical deforestation improved by carbon-density maps. *Nature Climate Change*, 2(3):182–185.
- Aarts, E. and Korst, J. (1987). *Simulated Annealing and Boltzmann Machines*. Wiley.
- Adcock, C. J. (1988). A Bayesian approach to calculating sample sizes. *Statistician*, 37:433–439.
- Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 07-09-January-2007:1027–1035.
- Bache, S. M. and Wickham, H. (2014). *magrittr: A Forward-Pipe Operator for R*. R package version 1.5.
- Baillargeon, S. and Rivest, L.-P. (2011). The construction of stratified designs in R with the package stratification. *Survey Methodology*, 37:53–65.
- Ballabio, C., Lugato, E., Fernández-Ugalde, O., Orgiazzi, A., Jones, A., Borrelli, P., Montanarella, L., and Panagos, P. (2019). Mapping LUCAS topsoil chemical properties at European scale using Gaussian process regression. *Geoderma*, 355:113912.
- Barcaroli, G. (2014). SamplingStrata: An R Package for the optimisation of Stratified Sampling. *Journal of Statistical Software*, 61(4).
- Barcaroli, G., Ballin, M., Odendaal, H., Pagliuca, D., Willighagen, E., and

- Zardetto, D. (2020). *SamplingStrata: Optimal Stratification of Sampling Frames for Multipurpose Sampling Surveys*. R package version 1.5-1.
- Barnes, R. J. (1988). Bounding the required sample size for geologic site characterization. *Mathematical Geology*, 20:477–490.
- based on Fortran code by Alan Miller, T. L. (2020). *leaps: Regression Subset Selection*. R package version 3.1.
- Berger, Y. G. (2004). A simple variance estimator for unequal probability sampling without replacement. *Journal of Applied Statistics*, 31(3):305–315.
- Bethel, J. (1989). Sample allocation in multiple surveys. *Survey Methodology*, 15(1):47–57.
- Bivand, R., Keitt, T., and Rowlingson, B. (2020). *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*. R package version 1.5-17.
- Bivand, R. S., Pebesma, E., and Gomez-Rubio, V. (2013). *Applied spatial data analysis with R, Second edition*. Springer, NY.
- Bogaert, P. and Russo, D. (1999). Optimal spatial sampling design for the estimation of the variogram based on a least squares approach. *Water Resources Research*, 35:1275–1289.
- Breidaks, J., Liberts, M., and Jukams, J. (2020). *surveyplanning: Survey planning tools*. Riga, Latvia. R package version 4.0.
- Breidenbach, J. (2018). *JoSAE: Unit-Level and Area-Level Small Area Estimation*. R package version 0.3.0.
- Breidenbach, J. and Astrup, R. (2012). Small area estimation of forest attributes in the Norwegian National Forest Inventory. *European Journal of Forest Research*, 131:1255–1267.
- Breidt, F. J. and Fuller, W. A. (1999). Design of supplemented panel surveys with application to the National Resources Inventory. *Journal of Agricultural, Biological and Environmental Statistics*, 4:391–402.
- Breidt, F. J. and Opsomer, J. D. (2017). Model-assisted survey estimation with modern prediction techniques. *Statistical Science*, 32(2):190–205.
- Brown, L. D., Cai, T. T., and DasGupta, A. (2001). Interval estimation for a binomial proportion - Comment - Rejoinder. *Statistical Science*, 16(2):101–133.

- Brus, D. J. (2015). Balanced sampling: A versatile sampling approach for statistical soil surveys. *Geoderma*, 253-254:111–121.
- Brus, D. J. (2021). Statistical approaches for spatial sample survey: Persistent misconceptions and new developments. *European Journal of Soil Science*.
- Brus, D. J. and de Gruijter, J. J. (1994). Estimation of nonergodic variograms and their sampling variance by design-based sampling strategies. *Mathematical Geology*, 26:437–454.
- Brus, D. J. and de Gruijter, J. J. (1997). Random sampling or geostatistical modelling? Choosing between design-based and model-based sampling strategies for soil (with Discussion). *Geoderma*, 80:1–59.
- Brus, D. J. and de Gruijter, J. J. (2003). A method to combine non-probability sample data with probability sample data in estimating spatial means of environmental variables. *Environmental Monitoring and Assessment*, 83:303–317.
- Brus, D. J., de Gruijter, J. J., and van Groenigen, J. W. (2007). Designing spatial coverage samples using the k-means clustering algorithm. In Lagacherie, P., McBratney, A. B., and Voltz, M., editors, *Digital Soil Mapping. An introductory perspective*, pages 183–192. Elsevier.
- Brus, D. J. and Heuvelink, G. B. M. (2007). Optimisation of sample patterns for universal kriging of environmental variables. *Geoderma*, 138:86–95.
- Brus, D. J., Kempen, B., and Heuvelink, G. B. M. (2011). Sampling for validation of digital soil maps. *European Journal of Soil Science*, 62(3):394–407.
- Brus, D. J. and Saby, N. P. A. (2016). Approximating the variance of estimated means for systematic random sampling, illustrated with data of the French Soil Monitoring Network. *Geoderma*, 279:77–86.
- Brus, D. J., Spätkens, L. E. E. M., and de Gruijter, J. J. (1999). A sampling scheme for estimating the mean extractable phosphorus concentration of fields for environmental regulation. *Geoderma*, 89:129–148.
- Cassel, C., Särndal, C., and Wretman, J. (1977). *Foundations in Inference in Survey Sampling*. Wiley, New York.
- Champely, S. (2020). *pwr: Basic Functions for Power Analysis*. R package version 1.3-0.
- Chauduri, A. (1994). Small domain statistics: a review. *Statistica Neerlandica*, 48:215–236.

- Christensen, R. (1991). *Linear Models for Multivariate, Time Series and Spatial Data*. Springer, New York.
- Clifford, P., Richardson, S., and Hemon, D. (1989). Assessing the significance of the correlation between two spatial processes. *Biometrics*, 45:123–134.
- Cochran, W. G. (1977). *Sampling Techniques*. Wiley, New York.
- Corwin, D. L. and Lesch, S. M. (2005). Characterizing soil spatial variability with apparent soil electrical conductivity: Part II. case study. *Computers and Electronics in Agriculture*, 46(1–3):135–152.
- Corwin, D. L., Lesch, S. M., Segal, E., Skaggs, T. H., and Bradford, S. A. (2010). Comparison of sampling strategies for characterizing spatial variability with apparent soil electrical conductivity directed soil sampling. *Journal of Environmental and Engineering Geophysics*, 15(3):147–162.
- Cullman, A. D. (2020). *maSAE: Mandallaz' Model-Assisted Small Area Estimators*. R package version 2.0.2.
- Dalenius, T. and Hodges, J. L. (1959). Minimum variance stratification. *Journal of the American Statistical Association*, 54(285):88–101.
- de Gruijter, J. J., Brus, D. J., Bierkens, M. F. P., and Knotters, M. (2006). *Sampling for Natural Resource Monitoring*. Springer, Berlin.
- de Gruijter, J. J., Minasny, B., and McBratney, A. B. (2015). optimising stratification and allocation for design-based estimation of spatial means using predictions with error. *Journal of Survey Statistics and Methodology*, 3:19–42.
- de Gruijter, J. J. and ter Braak, C. J. F. (1990). Model-free estimation from spatial samples: a reappraisal of classical sampling theory. *Mathematical Geology*, 22:407–415.
- de Gruijter, J. J. and ter Braak, C. J. F. (1992). Design-based versus model-based sampling strategies: comment on R.J. Barnes' ‘Bounding the required sample size for geologic site characterization’. *Mathematical Geology*, 24:859–864.
- De Vries, P. G. (1986). *Sampling Theory for Forest Inventory: A Teach-Yourself Course*. Springer Verlag, New York.
- Deville, J. C. and Tillé, Y. (1998). Unequal probability sampling without replacement through a splitting method. *Biometrika*, 85(1):89–101.

- Deville, J. C. and Tillé, Y. (2004). Efficient balanced sampling: The cube method. *Biometrika*, 91:893–912.
- Deville, J. C. and Tillé, Y. (2005). Variance approximation under balanced sampling. *Journal of Statistical Planning and Inference*, 128:569–591.
- Domburg, P., de Gruijter, J. J., and Brus, D. J. (1994). A structured approach to designing soil survey schemes with prediction of sampling error from variograms. *Geoderma*, 62:151–164.
- Douglas Nychka, Reinhard Furrer, John Paige, and Stephan Sain (2017). *fields: Tools for spatial data*. University Corporation for Atmospheric Research, Boulder, CO, USA. R package version 11.5.
- Escobar, E. L., Zamudio, E. B., and Munoz Rosas, J. F. (2019). *samplingVarEst: Sampling Variance Estimation*. R package version 1.4.
- Falorsi, P. and Righi, P. (2008). A balanced sampling approach for multi-way stratification designs for small area estimation. *Survey Methodology*, 34:223–234.
- Fitzgerald, G. J. (2010). *Response Surface Sampling of Remotely Sensed Imagery for Precision Agriculture*, chapter 10, pages 121–129. Springer Netherlands, Sydney, Australia.
- Fitzgerald, G. J., Lesch, S. M., Barnes, E. M., and Luckett, W. E. (2006). Directed sampling using remote sensing with a response surface sampling design for site-specific agriculture. *Computers and Electronics in Agriculture*, 53(2):98–112.
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. CRC Press, third edition.
- Gershenfeld, N. (1999). *The Nature of Mathematical Modeling*. Cambridge University Press, Cambridge.
- Goerg, G. M. (2013). *LICORS: Light Cone Reconstruction of States - Predictive State Estimation From Spatio-Temporal Data*. R package version 0.2.0.
- Goovaerts, P. (1997). *Geostatistics for Natural Resources Evaluation*. Oxford University Press, New York.
- Grafström, A. (2012). Spatially correlated Poisson sampling. *Journal of Statistical Planning and Inference*, 142(1):139–147.

- Grafström, A., Ekström, M., Jonsson, B., Esseen, P.-A., and Stahl, G. (2019). On combining independent probability samples. *Survey Methodology*, 45(2):349–364.
- Grafström, A. and Lisić, J. (2016). *BalancedSampling: Balanced and Spatially Balanced Sampling*. R package version 1.5.1.
- Grafström, A., Lundström, N. L. P., and Schelin, L. (2012). Spatially balanced sampling through the pivotal method. *Biometrics*, 68:514–520.
- Grafström, A. and Schelin, L. (2014). How to select representative samples. *Scandinavian Journal of Statistics*, 41(2):277–290.
- Grafström, A. and Tillé, Y. (2013). Doubly balanced spatial sampling with spreading and restitution of auxiliary totals. *Environmetrics*, 24(2):120–131.
- Griffith, D. A. (2005). Effective geographic sample size in the presence of spatial autocorrelation. *Annals of the Association of American Geographers*, 95(4):740–760.
- Hankin, D. G., Mohr, M. S., and Newman, K. B. (2019). *Sampling Theory: For the Ecological and Natural Resource Sciences*. Oxford University Press.
- Hansen, M. H., Madow, W. G., and Tepping, B. J. (1983). An evaluation of model-dependent and probability sampling inferences in sample-surveys. *Journal of the American Statistical Association*, 78:805–807.
- Hartig, F., Minunno, F., and Paul, S. (2018). *BayesianTools: General-Purpose MCMC and SMC Samplers and Tools for Bayesian Statistics*. R package version 0.1.5.
- Heuvelink, G. B. M., Brus, D. J., and de Gruijter, J. J. (2007). Optimisation of sample configurations for digital mapping of soil properties. In Lagacherie, P., McBratney, A. B., and Voltz, M., editors, *Digital Soil Mapping. An introductory perspective*, pages 1020–1030. Elsevier.
- Hill, A., Massey, A., and Mandallaz, D. (2021). The R package forestinventory: Design-based global and small area estimations for multiphase forest inventories. *Journal of Statistical Software*, 97(4).
- Hofman, S. C. K. and Brus, D. J. (2021). How many sampling points are needed to estimate the mean nitrateconcentration of agricultural fields? A geostatistical simulation approach withuncertain variograms. *Geoderma*.

- Höhle, M. (2017). *binomSamSize: Confidence Intervals and Sample Size Determination for a Binomial Proportion under Simple Random Sampling and Pooled Sampling*. R package version 0.1-5.
- Hutson, A. D. (1999). Calculating nonparametric confidence intervals for quantiles using fractional order statistics. *Journal of Applied Statistics*, 26:343 – 353.
- Hutson, N., Hutson, A., and Yan, L. (2019). *QuantileNPCI: Nonparametric Confidence Intervals for Quantiles*. R package version 0.9.0.
- Hyndman, R. and Fan, Y. (1996). Sample quantiles in statistical packages. *American Statistician*, 50:361 – 365.
- Isaaks, E. H. and Srivastava, R. H. (1989). *An Introduction to Applied Geostatistics*. Oxford University Press, New York.
- Janssen, P. H. M. and Heuburger, P. S. C. (1995). Calibration of process-oriented models. *Ecological Modelling*, 83(1-2):55–66.
- Joseph, L. and Bélisle, P. (1997). Bayesian sample size determination for normal means and differences between normal means. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 46(2):209–226.
- Joseph, L. and Bélisle, P. (2012). *SampleSizeMeans: Sample size calculations for normal means*. R package version 1.1.
- Joseph, L., Wolfson, D. B., and Du Berger, R. (1995). Sample size calculations for binomial proportions via highest posterior density intervals. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 44(2):143–154.
- Kim, J. and Wang, Z. (2019). Sampling techniques for big data analysis. *International Statistical Review*, 87(S1):S177–S191.
- Kincaid, T. M., Olsen, A. R., and Weber, M. H. (2019). *spsurvey: Spatial Survey Design and Analysis*. R package version 4.1.0.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). optimisation by simulated annealing. *Science*, 220(4598):671–680.
- Kitanidis, P. (1983). Statistical estimation of polynomial generalized covariance functions and hydrological applications. *Water Resources Research*, 19:909–921.

- Kitanidis, P. K. (1987). Parametric estimation of covariances of regionalised variables. *Water Resources Bulletin*, 23:557–567.
- Lark, R. M. (2000). Estimating variograms of soil properties by the method-of-moments and maximum likelihood. *European Journal of Soil Science*, 51:717–728.
- Lark, R. M. (2002). Optimised spatial sampling of soil for estimation of the variogram by maximum likelihood. *Geoderma*, 105:49–80.
- Lark, R. M. (2011). Spatially nested sampling schemes for spatial variance components: Scope for their optimisation. *Computers & Geosciences*, 37(10):1633–1641.
- Lark, R. M., Hamilton, E. M., Kanninga, B., Maseka, K. K., Mutondo, M., Sakala, G. M., and Watts, M. J. (2017). Planning spatial sampling of the soil from an uncertain reconnaissance variogram. *SOIL*.
- Lark, R. M. and Marchant, B. (2018). How should a spatial-coverage sample design for a geostatistical soil survey be supplemented to support estimation of spatial covariance parameters? *Geoderma*, 319:89–99.
- Lark, R. M. and Webster, R. (2006). Geostatistical mapping of geomorphic variables in the presence of trend. *Earth Surface Processes and Landforms*, 31:862–874.
- Lenth, R. V. (2009). Response-Surface Methods in R, Using rsm. *Journal of Statistical Software*, 32(7):1–17.
- Lesch, S. M. (2005). Sensor-directed response surface sampling designs for characterizing spatial variation in soil properties. *Computers and Electronics in Agriculture*, 46(1–3):153–179.
- Lesch, S. M., Strauss, D. J., and Rhoades, J. D. (1995). Spatial prediction of soil salinity using electromagnetic induction techniques 2. An efficient spatial sampling algorithm suitable for multiple linear regression model identification and estimation. *Water Resources Research*, 31:387–398.
- Lisic, J. and Grafström, A. (2018). *SamplingBigData: Sampling Methods for Big Data*. R package version 1.0.0.
- Lohr, S. L. (1999). *Sampling: Design and Analysis*. Duxbury Press, Pacific Grove, USA.

- Lumley, T. (2020). *survey: analysis of complex survey samples*. R package version 4.0.
- Ly, A., Marsman, M., Verhagen, J., Grasman, R. P. P. P., and Wagenmakers, E. J. (2017). A tutorial on fisher information.
- Ma, T., Brus, D. J., Zhu, A.-X., Zhang, L., and Scholten, T. (2020). Comparison of conditioned Latin hypercube and feature space coveragesampling for predicting soil classes using simulation from soil maps. *Geoderma*, 370.
- Mandallaz, D. (2007). *Sampling Techniques for Forest Inventories*. Chapman and Hall/CRC.
- Mandallaz, D., Breschan, J., and Hill, A. (2013). New regression estimators in forest inventories with two-phase sampling and partially exhaustive information: A design-based Monte Carlo approach with applications to small-area estimation. *Canadian Journal of Forest Research*, 43(11):1023–1031.
- Marcelli, A., Corona, P., and Fattorini, L. (2019). Design-based estimation of mark variograms in forest ecosystem surveys. *Spatial Statistic*, 30:27–38.
- Marchant, B. P. and Lark, R. M. (2007). optimised sample schemes for geostatistical surveys. *Mathematical Geology*, 39(1):113–134.
- Matérn, B. (1947). Methods of estimating the accuracy of line and sample plot surveys. *Meddelanden från Statens Skogsforstningsinstitut*, 36:138 –.
- Matérn, B. (1986). *Spatial Variation*, volume 36 of *Lecture Notes in Statistics*. Springer-Verlag, New York, second edition.
- McConville, K. (2018). A tutorial on model-assisted estimation with application to forest inventory. *Forests*, 11(244).
- McConville, K., Tang, B., Zhu, G., Cheung, S., and Li, S. (2018). *mase: Model-Assisted Survey Estimation*.
- McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.
- Minasny, B. and McBratney, A. B. (2006). A conditioned Latin hypercube method for sampling in the presence of ancillary information. *Computers & Geosciences*, 32:1378–1388.

- Müller, W. G. and Zimmerman, D. L. (1999). Optimal designs for variogram estimation. *Environmetrics*, 10:23–27.
- Myers, R. H., Montgomery, D. C., and Anderson-Cook, C. M. (2002). *Response Surface Methodology: Process and Product optimisation Using Designed Experiments. Third edition*. Wiley, New York.
- Nanthakumar, A. and Selvavel, K. (2004). Estimation of proportion of success from a stratified population: a comparative study. *Communications in Statistics*, 33:2245–2257.
- Oliver, M. A. and Webster, R. (1986). Combining nested and linear sampling for determining the scale and form of spatial variation of regionalized variables. *Geographical Analysis*, 18:227–242.
- Ott, R. L. and Longnecker, M. (2015). *An Introduction to Statistical Methods and Data Analysis, Seventh edition*. Cengage Learning.
- Papritz, A., Duemig, A., Zimmermann, C., Gerke, H. H., Felderer, B., Koegel-Knabner, I., Schaaf, W., and Schulin, R. (2011). Uncertainty of variance component estimates in nested sampling: a case study on the field-scale spatial variability of a restored soil. *European Journal of Soil Science*, 62(3):479–495.
- Pebesma, E. J. (2004). Multivariable geostatistics in S: the gstat package. *Computers & Geosciences*, 30:683–691.
- Pebesma, E. J. and Bivand, R. S. (2005). Classes and methods for spatial data in R. *R News*, 5(2):9–13.
- Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D., and R Core Team (2014). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-117.
- Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D., and R Core Team (2020). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-149.
- Plant, R. E. (2012). *Spatial data analysis in ecology and agriculture using R*. CRC Press.
- R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Rao, J. (2003). *Small Area Estimation*. Wiley.
- Ribeiro Jr, P. J., Diggle, P. J., Schlather, M., Bivand, R., and Ripley, B. (2020). *geoR: Analysis of Geostatistical Data*. R package version 1.8-1.

- Ripley, B. D. (1981). *Spatial statistics*. John Wiley Sons, New York.
- Robertson, B. L., Brown, J. A., McDonald, T., and Jaksons, P. (2013). BAS: Balanced Acceptance Sampling of Natural Resources. *Biometrics*.
- Rosén, B. (1997). On sampling with probability proportional to size. *Journal of Statistical Planning and Inference*, 62:159–191.
- Roudier, P. (2011). *clhs: a R package for conditioned Latin hypercube sampling*.
- Samuel-Rosa, A. (2016). *spsann: optimisation of Sample Configurations using Spatial Simulated Annealing*. R package version 2.0-0.
- Särndal, C. E., Swensson, B., and Wretman, J. (1992). *Model Assisted Survey Sampling*. Springer, New York.
- Schloerke, B., Cook, D., Larmarange, J., Briatte, F., Marbach, M., Thoen, E., Elberg, A., and Crowley, J. (2021). *GGally: Extension to 'ggplot2'*. R package version 2.1.2.
- Schoch, T. (2014). *rsae: Robust Small Area Estimation*. R package version 0.1-5.
- Signorell, A. (2020). *DescTools: Tools for Descriptive Statistics*. R package version 0.99.38.
- Stehman, S. V. (1999). Basic probability sampling designs for thematic map accuracy assessment. *International Journal of Remote Sensing*, 20:2423–2441.
- Stehman, S. V., Fonte, C. C., Foody, G. M., and See, L. (2018). Using volunteered geographic information (VGI) in design-based statistical inference for area estimation and accuracy assessment of land cover. *Remote Sensing of Environment*, 212:47–59.
- Stevens, D. L. and Olson, A. R. (2004). Spatially balanced sampling of natural resources. *Journal of American Statistical Association*, 99:262–278.
- ter Braak, C. J. F. and Vrugt, J. A. (2008). Differential evolution markov chain with snooker updater and fewer chains. *Statistics and Computing*, 18:435–446.
- Tillé, Y. (2006). *Sampling algorithms*. Springer.
- Tillé, Y. and Matei, A. (2016). *sampling: Survey Sampling*. R package version 2.8.

- Toth, D. (2019). *rpms: Recursive Partitioning for Modeling Survey Data*. R package version 0.4.0.
- United Nations on Drugs and Crime, Islamic Republic of Afghanistan, Ministry of Counter Narcotics (2014). Afghanistan opium survey 2014. Technical report.
- Vaillant, R., Dever, J. A., and Kreuter, F. (2018). *Practical Tools for Designing and Weighting Survey Samples*. Springer, New York, second edition.
- Valliant, R., Dever, J. A., and Kreuter, F. (2020). *PracTools: Tools for Designing and Weighting Survey Samples*. R package version 1.2.2.
- van Groenigen, J. W., Pieters, G., and Stein, A. (2000). optimising spatial sampling for multivariate contamination in urban areas. *Environmetrics*, 11:227–244.
- van Groenigen, J. W., Siderius, W., and Stein, A. (1999). Constrained optimisation of soil sampling for minimisation of the kriging variance. *Geoderma*, 87:239–259.
- van Groenigen, J. W. and Stein, A. (1998). Constrained optimisation of spatial sampling using continuous simulated annealing. *Journal of Environmental Quality*, 27:1078–1086.
- Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition.
- Wadoux, A. M. J.-C., Brus, D. J., and Heuvelink, G. B. M. (2019). Sampling design optimisation for soil mapping with random forest. *Geoderma*, 355.
- Walvoort, D. J. J., Brus, D. J., and de Gruijter, J. J. (2010). An R package for spatial coverage sampling and random sampling from compact geographical strata by k-means. *Computers and Geosciences*, 36:1261–1267.
- Wang, J., Haining, R., and Cao, Z. (2010). Sample surveying to estimate the mean of a heterogeneous surface: reducing the error variance through zoning. *International Journal of Geographical Information Science*, 24(4):523–543.
- Wang, J., Stein, A., Gao, B., and Ge, Y. (2012). A review of spatial sampling. *Spatial Statistics*, 2(4):1–14.
- Webster, R. and Oliver, M. A. (1992). Sample adequately to estimate variograms of soil properties. *Journal of Soil Science*, 43:177–192.

- Webster, R. and Oliver, M. A. (2007). *Geostatistics for Environmental Scientists, Second Edition*. Wiley, Chichester.
- Webster, R., Welham, S. J., Potts, J. M., and Oliver, M. A. (2006). Estimating the spatial scales of regionalized variables by nested sampling, hierarchical analysis of variance and residual maximum likelihood. *Computers and Geosciences*, 32:1320–1333.
- Wickham, H. (2020). *forcats: Tools for Working with Categorical Variables (Factors)*. R package version 0.5.0.
- Wright, M. N. and Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77(1).
- Wu, C. (2003). Optimal calibration estimators in survey sampling. *Biometrika*, 90(4):937–951.
- Wu, C. and Sitter, R. R. (2001). A model-calibration approach to using complete auxiliary information from survey data. *Journal of the American Statistical Association*, 96(453):185–193.
- Xie, Y., Dervieux, C., and Riederer, E. (2020). *R Markdown Cookbook*. Chapman & Hall/CRC, first edition.
- Zhu, Z. and Stein, M. L. (2005). Spatial sampling design for parameter estimation of the covariance function. *Journal of Statistical Planning and Inference*, 134:583–603.
- Zhu, Z. and Stein, M. L. (2006). Spatial sampling design for prediction with estimated parameters. *Journal of Agricultural Biological and Environmental Statistics*, 11:24–44.
- Zhu, Z. and Zhang, H. (2006). Spatial sampling under the infill asymptotic framework. *Environmetrics*, 17:323–337.
- Zimmerman, D. L. (2006). Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction. *Environmetrics*, 17:635–652.

Index

- π estimator, 21
- π -expanded value, 21
- π^* -estimator, 226
- Cum-root-f* stratification, 68
- k*-means clustering, 72, 92
- k*-means++ algorithm, 355
- k*th order statistic, 35
- p*-value of a test, 275
 - Two-sided *p*-value, 496
- A priori variance, 394, 396
- Accuracy, 34
- Adapted experimental design, 336
- Allocation
 - Bethel allocation, 66
 - Neyman allocation, 64
 - optimal allocation, 65
 - proportional allocation, 64
- Allocation: Bethel allocation, 79
- ANCOVA model, 196
- Annealing schedule, 364
- Arbitrary sampling, 7, 30
- Areal sampling unit, 2, 138
- Augmented kriging variance, 475
- Autocorrelation, 392, 509
- Average coverage criterion, 249
- Average length criterion, 249
- Back-up list of sampling units, 30
- Balanced acceptance sampling, 169
- Balanced sampling, 153
- Bayes' rule, 247
- Bayesian approach
 - to grid spacing determination, 427
 - to prediction of sampling variance, 275
 - to sample size determination, 246
- Best linear unbiased estimator, 510
- Best linear unbiased predictor, 393, 508
- Beta distribution, 254
- Bias, 34
- Big data, 224
- Binomial distribution, 48
- Binomial proportion, 48
- Binomial test, 243
- Bivariate normal-gamma distribution, 251
- Block-kriging variance, 399
- Bootstrap sample, 219, 461
- Brewer's variance estimator, 113, 131, 148
- Burn-in period, 278
- Calibrated weights, 188, 202
- Categorical map, 488
- Census, 1
- Central composite design, 373
- Central composite response surface design, 376

- Central limit theorem, 45
Certainty stratum approach, 224
Chain length, 434
Change of population mean (total), 325
Change of support, 398
Circular sampling plot, 41
Class representation, 489
Classical sampling theory, 46, 499
Clopper-Pearson method, 48
Cluster random sampling, 101
Cluster size, 102
Coefficient of variation, 79
Cohens's h , 244
Collapsed strata, 77, 495
Compact geographical strata, 74, 265
Composite sampling, 77, 267
Conditioned Latin hypercube sampling, 362
Confidence interval, 45
Confidence level, 46
Conjugate prior, 254
Conservative variance estimator, 171
Consistent estimator, 61
Convenience sample, 269
Correlation function, 392
Correlogram, 392, 396
Covariance function, 392
Covariate space coverage sampling, 351
Covariogram, 392
Coverage probability, 248
Credible interval, 247
Cross-classification strata, 78, 164
Cross-validation, 486
 leave-one-out-crossvalidation, 486
Cube algorithm for balanced sampling, 157
Cube point, 376
Cumulative distribution function, 6
Data snooping, 212
Data-splitting, 486
Degrees of freedom, 46
Design effect, 61, 109, 128, 245
Design matrix, 160
Design parameter, 238, 246
Design point, 378
Design weight, 21
Design-based approach, 8, 499
Design-bias, 507
Design-independent, 507
Design-unbiased, 33
Design-variance, 507
Determinant of variance-covariance matrix, 330, 465
Differential evolution algorithm, 278
Direct estimator, 296
Discretisation error, 4
Distance parameter, 288, 466
Domains of interest, 295
Double sampling, 225
Doubly-balanced sampling, 176
Draw-by-draw selection probability
 of a cluster, 107
 of a population unit, 23
 of a primary sampling unit, 126
Drop outs, 29
Effective range, 380, 437
Effective sample size, 510
Elementary sampling unit, 2
Empirical coverage rate, 517
Estimation adjusted criterion, 476
Estimator, 21
Evaluation point, 440
Exhaustive data set, 9
Expectation of estimator, 33