Dick J. Brus

Spatial sampling with R

Contents

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.wageningenur.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).

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 $\mathcal{U} = \{u(\mathbf{s}_1), u(\mathbf{s}_2), \ldots, u(\mathbf{s}_N)\}$, 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 field[1]. 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).

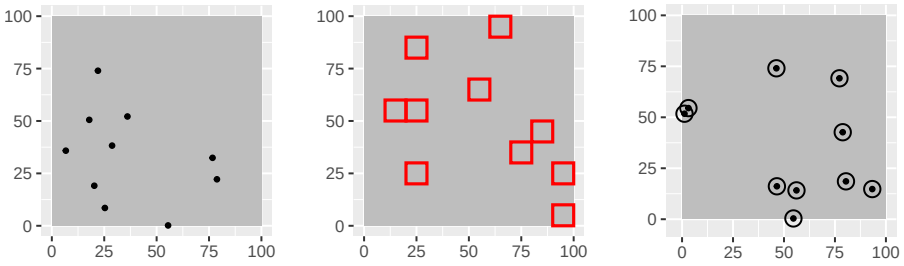

FIGURE 1.1: Three sample supports: points, squares and circles. With disjoint squares the population is finite. With points, and squares or circles that are allowed to overlap as sample support the population is infinite.

With areal elementary sampling units, ideally the selected elementary units are exhaustively observed, so that a measurement of the total or mean of the

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

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.

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

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

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:

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

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:

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

A cumulative distribution function (CDF) is defined as

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.

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

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

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

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 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.

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

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 **??** 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 variable*[2].

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

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

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 $\times$ 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 $\times$ 1 km. The study variable is the aboveground biomass as derived from remotes 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 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.

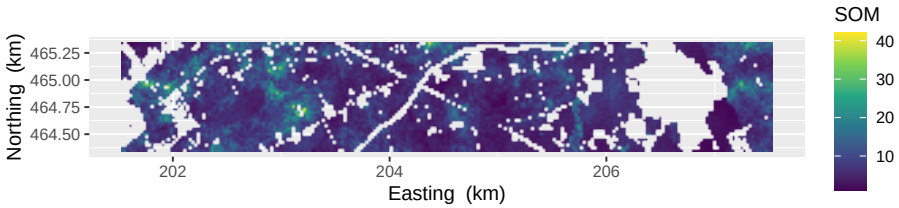

FIGURE 1.3: Simulated soil organic matter concentration (g/kg) in Voorst.

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

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.

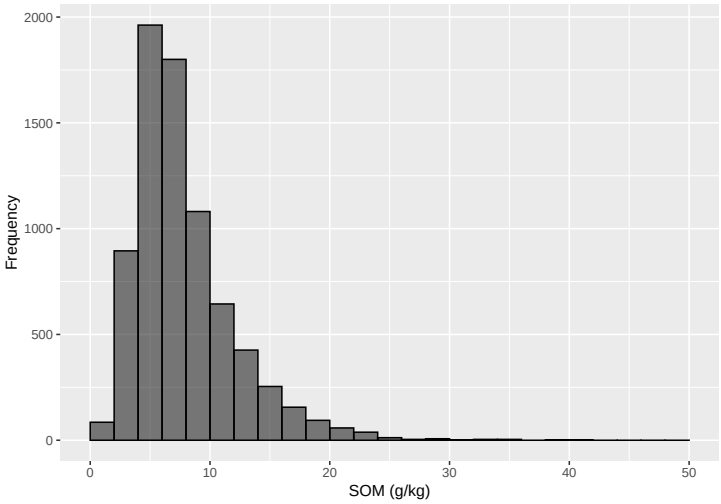

FIGURE 1.4: Histogram of simulated soil organic matter concentration (g/kg) in Voorst.

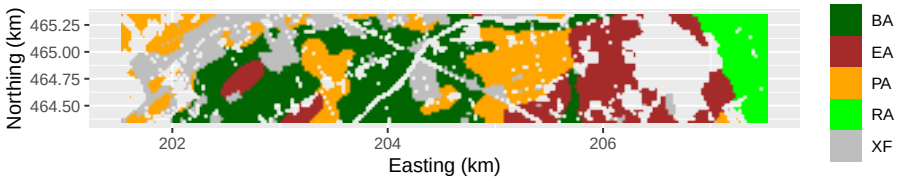

FIGURE 1.5: Soil-land use combinations in study area Voorst (Netherlands).

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 × 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.

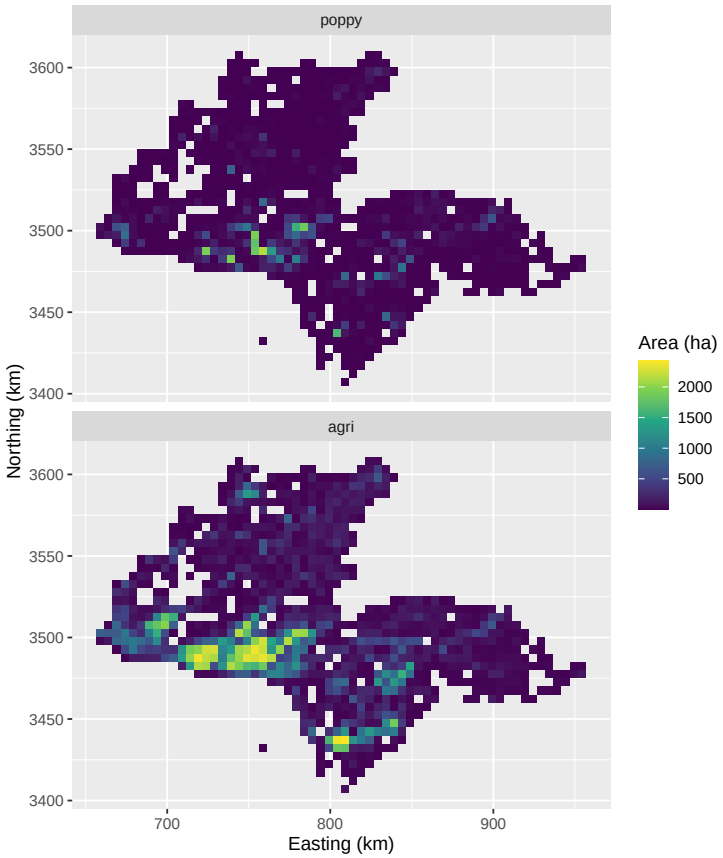

FIGURE 1.6: Agricultural area and simulated area cultivated with opium poppy, in hectares per 5 km by 5 km squares in Kandahar (Afghanistan).

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 × 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 m^2 (Terra_PP), average precipitation in driest month in mm (Prec_dm),

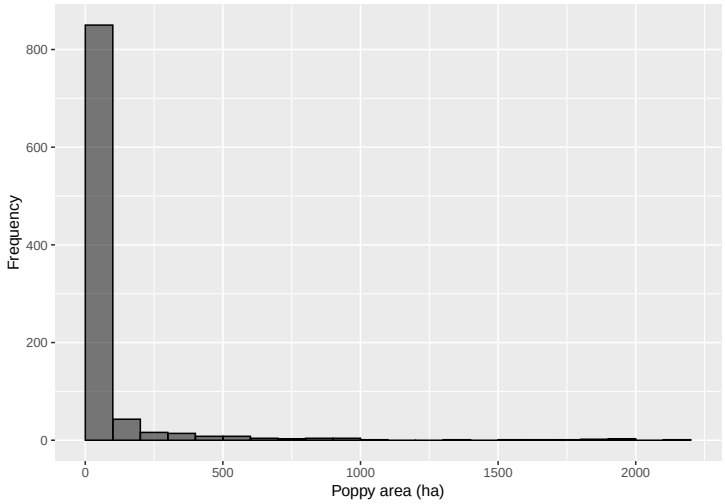

FIGURE 1.7: Histogram of simulated poppy area (ha) per 5 km by 5 km square in Kandahar.

Elevation in m, and Clay content in g per kg soil. All covariates were either 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.

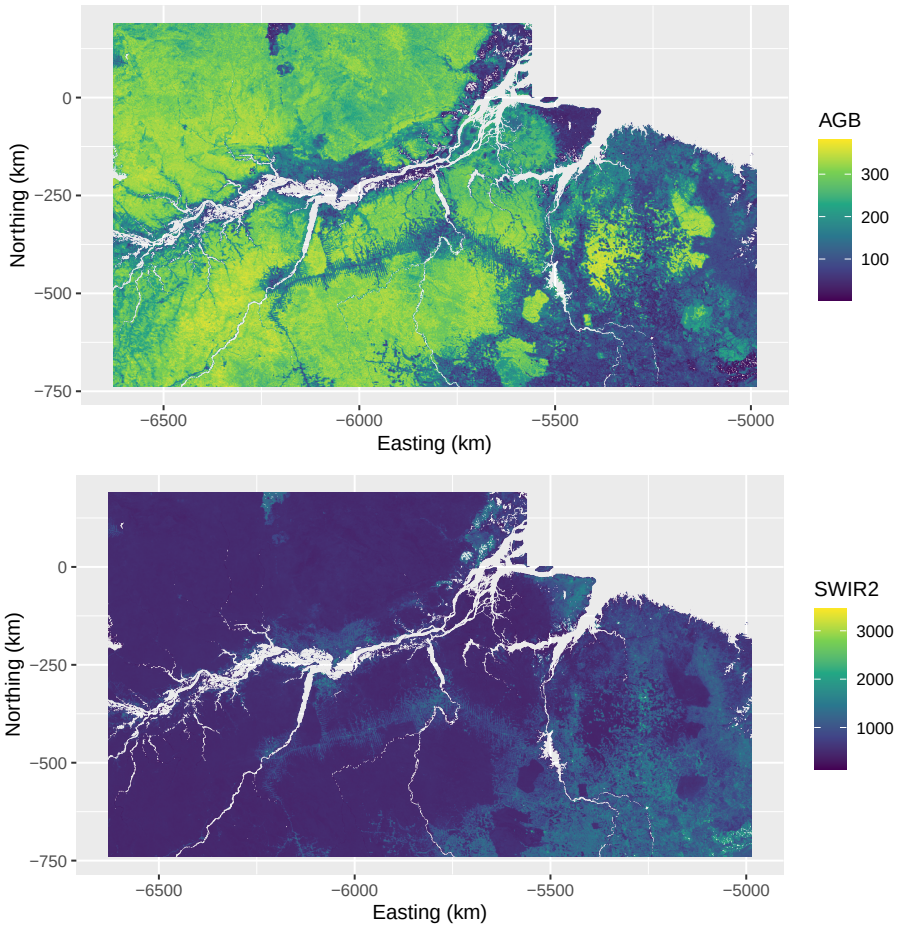

FIGURE 1.8: Aboveground biomas in megatons per ha (AGB), and shortwave infrared radiation (SWIR2) of Eastern Amazonia (Brazil).

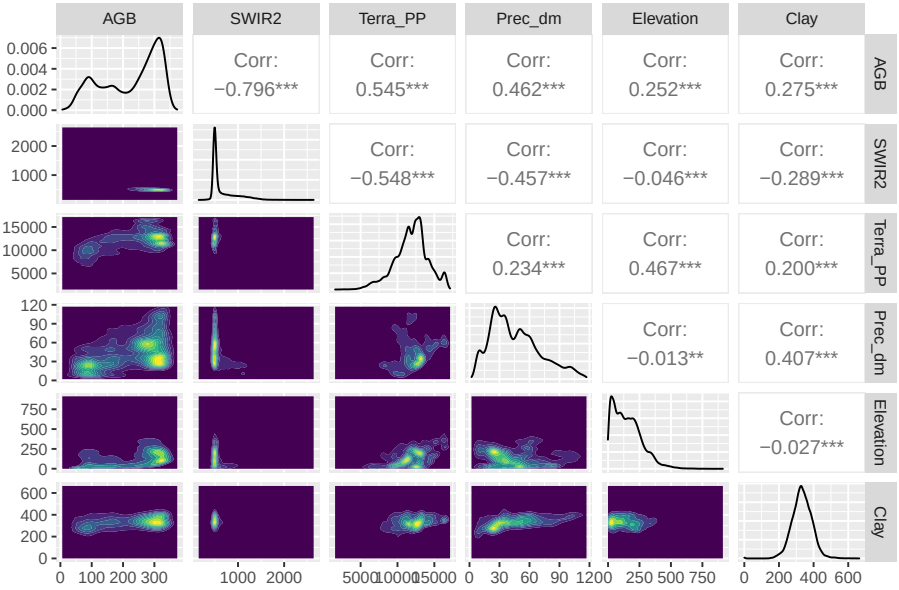

FIGURE 1.9: Matrix of two-dimensional density plots of aboveground biomass (AGB) and five covariates of Eastern Amazonia (Brazil).

Part I

Probability sampling for estimating (sub)population parameters

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:

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

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 nth 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 straightforward[2], as well as the associated estimation of 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 $1 \ldots N$. 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?

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

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

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:

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

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

$$w_k = \frac{1}{\pi_k} \ , \tag{2.3}$$

with π_k the inclusion probability of unit k. The estimator[3] 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 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

[3] An *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.

per ha or per m^2. The inclusion probability density can be interpreted as the sampling intensity[4].

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:

$$\hat{\bar{z}}_\pi = \frac{1}{N} \sum_{k \in \mathcal{S}} \frac{1}{\pi_k} z_k \; . \tag{2.4}$$

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:

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

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 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

[4] The 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.

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

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 $1 - \left(1 - \frac{1}{N}\right)^n$, 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

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

with $t_j(z)$ the total of the cluster selected in the jth 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 **??** 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 **??**.

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

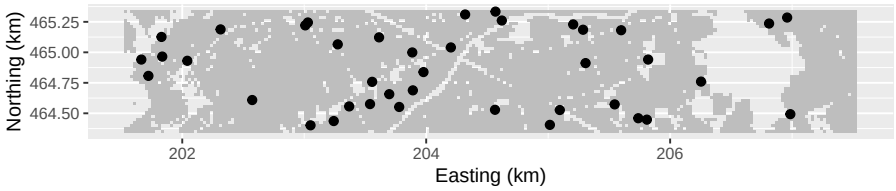

FIGURE 3.1: Simple random sample of size 40 from Voorst.

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)

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

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:

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

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 (**?**)

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

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

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

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 $1 - \left(1 - \frac{1}{N}\right)^n$, 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:

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

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} m^2. To estimate the total of SOM in the soil layer 0-30 cm, first the soil volume in m^3 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 10^6 to obtain the total SOM in megatons (10^9 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).

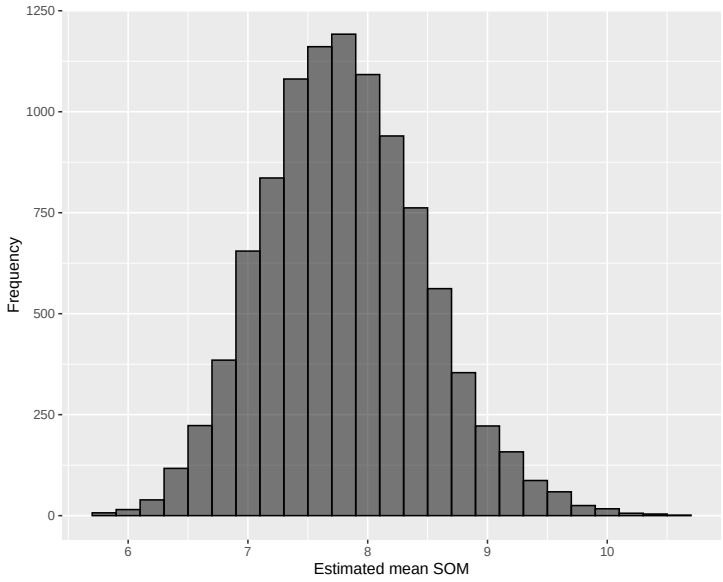

FIGURE 3.2: Sampling distribution of the estimator of the population mean of SOM (g/kg) in Voorst with simple random sampling of size 40.

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* 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 **??** another type of unbiasedness is introduced, model-unbiasedness. The difference between design-unbiasedness and model-unbiasedness is explained in Chapter **??**. 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 individ-

ual 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 1.1.1). For simple random sampling this population proportion can be estimated by the same formula as for the mean (Equation (3.2)):

$$\hat{p} = \frac{1}{n} \sum_{k \in \mathcal{S}} y_k \, . \tag{3.6}$$

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")
```

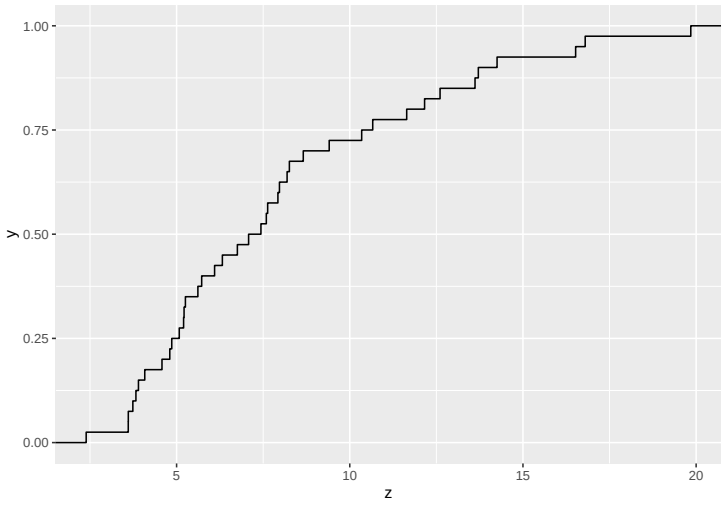

FIGURE 3.3: Cumulative distribution function, estimated from the simple random sample of 40 units from Voorst.

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 kth 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 quan-*

tiles, 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.

```
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.

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

$$V\left(\hat{\bar{z}}\right) = \frac{S^2(z)}{n} \,, \tag{3.7}$$

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

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

and for infinite populations as

$$S^2(z) = \frac{1}{A} \int\limits_{\mathbf{s}\in\mathcal{U}} \left(z(\mathbf{s}) - \bar{z}\right)^2 \mathrm{d}\mathbf{s} \,, \tag{3.9}$$

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

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

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

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

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

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

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

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

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

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

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 `svytotal`, 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 kth circle (sum of observations of all population units in kth 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 (**?**):

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

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.

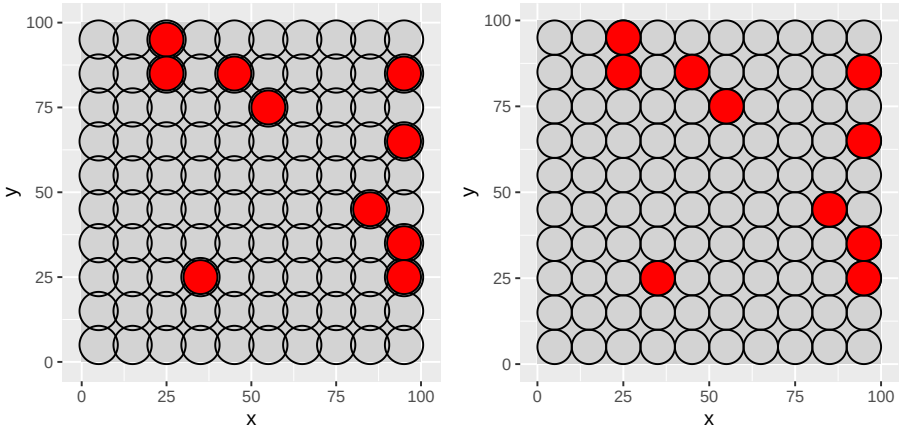

FIGURE 3.4: A simple random sample of ten circular plots from a square area discretised by a finite set of partly overlapping or non-overlapping circular plots.

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 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 be estimated by (**?**)

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

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:

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

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

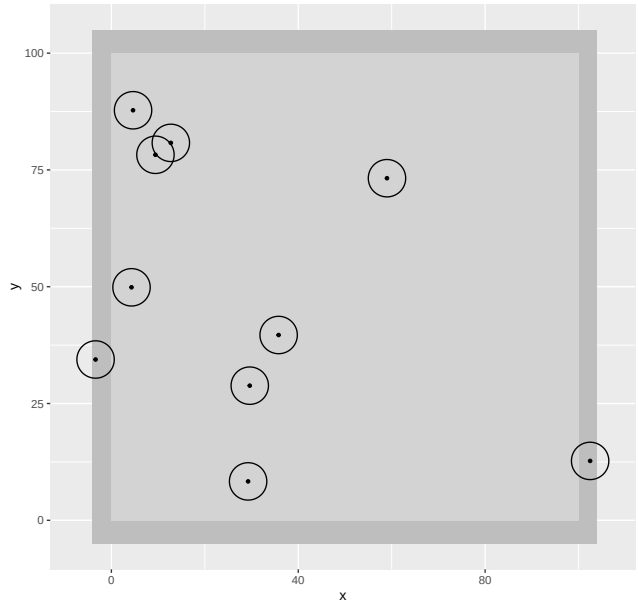

FIGURE 3.5: A simple random sample of ten floating circular plots from a square area.

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

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

$$\hat{\bar{z}} \pm u_{(0.10/2)} \cdot \sqrt{V\left(\hat{\bar{z}}\right)}\,, \tag{3.19}$$

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 $V\left(\hat{\bar{z}}\right)$ 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:

$$\hat{\bar{z}} \pm t^{(n-1)}_{1-\alpha/2} \cdot \sqrt{\widehat{V}\left(\hat{\bar{z}}\right)}\,, \tag{3.20}$$

where $t^{(n-1)}_{1-\alpha/2}$ 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 $t^{(n-1)}_{1-\alpha/2} \cdot \sqrt{\widehat{V}\left(\hat{\bar{z}}\right)}$ 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.

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 rum 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 **?** 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-pearson"))
```

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

The confidence interval is not symmetric around the estimated proportion of

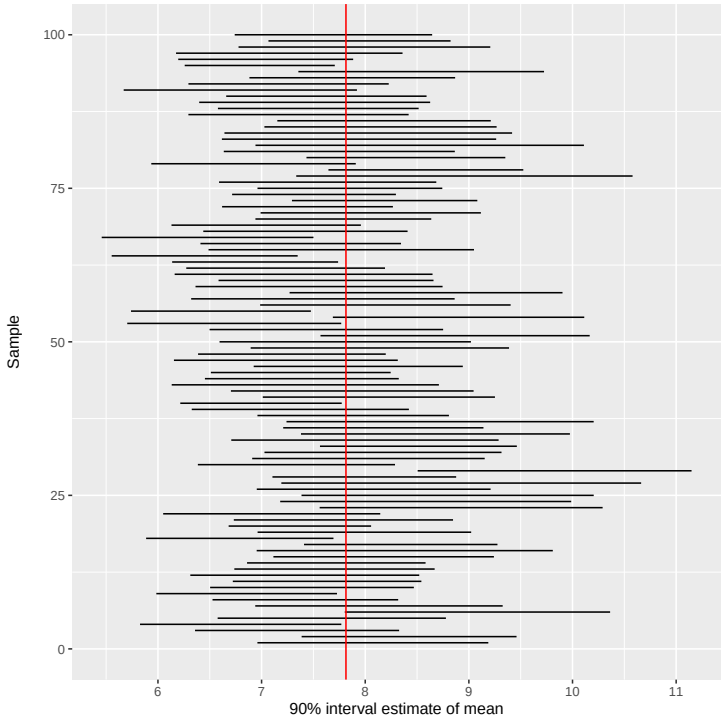

FIGURE 3.6: Estimated confidence intervals of the population mean of SOM (g per kg) in Voorst, estimated from 100 simple random samples of size 40. The vertical red line is at the true population mean.

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 inter-

val 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:

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

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?

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 (STSI). 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.

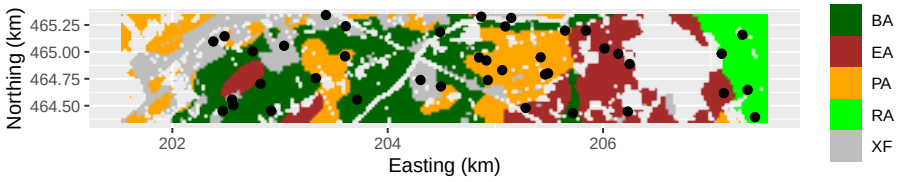

FIGURE 4.1: Stratified simple random sample of size 40 from Voorst. Strata are combinations of soil class and land use.

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:

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

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:

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

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 strata[1] 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

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

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:

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

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

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

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

[1] The 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 (Nh), stratum sample size (nh), 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

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

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.

```
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       1
3712 204305.5 464738.9  6.217245      XF    2159 0.005189017       1
6781 203038.3 465057.0 15.140350      XF    4205 0.005189017       1
7235 202381.1 465096.7  5.766409      XF    4503 0.005189017       1
8484 203610.4 465237.8  9.363492      XF    5336 0.005189017       1
9265 205147.1 465315.5 10.928986      XF    5853 0.005189017       1
```

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).

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

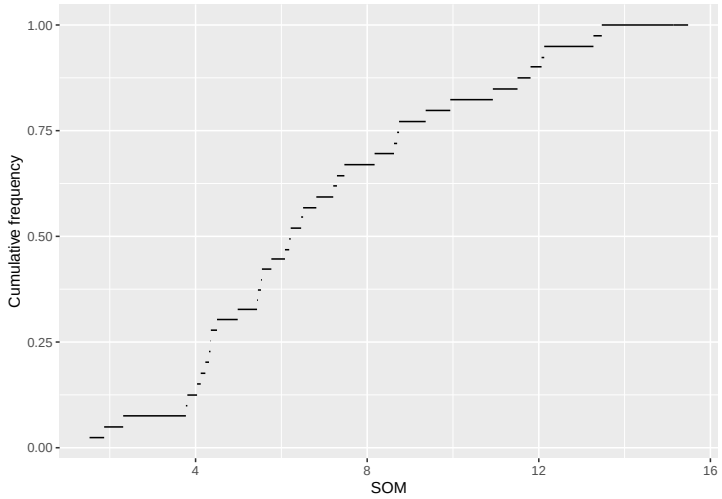

FIGURE 4.2: Cumulative distribution function estimated from the stratified simple random sample of 40 units from Voorst.

```
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.

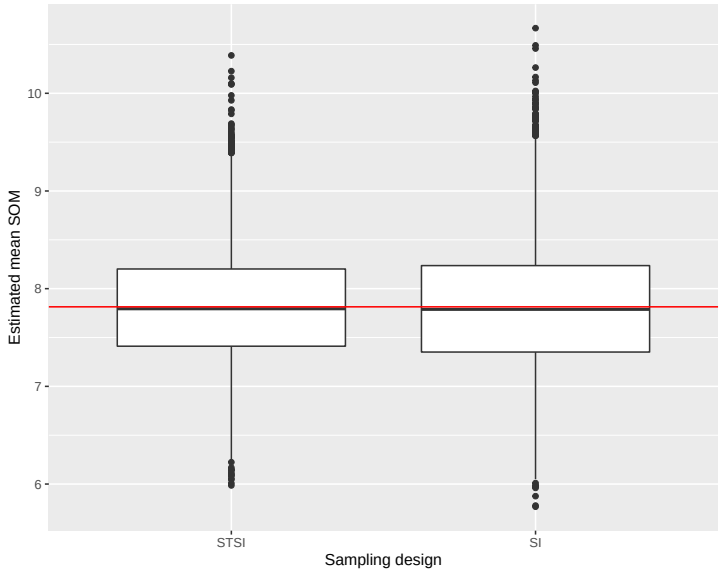

FIGURE 4.3: Sampling distribution of the estimator of the mean of SOM (g/kg) in Voorst for stratified simple random sampling and simple random sampling of size 40.

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 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 (**?**)

$$\widehat{S^2}(z) = \widehat{\overline{z^2}} - \left(\hat{\bar{z}}\right)^2 + \widehat{V}\left(\hat{\bar{z}}\right) \ , \tag{4.7}$$

where $\widehat{\overline{z^2}}$ denotes the estimated population mean of the study variable squared (z^2), obtained in the same way as $\hat{\bar{z}}$ (Equation (4.1), but using squared values, and $\widehat{V}\left(\hat{\bar{z}}\right)$ 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:

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

The population variance can be estimated with function `s2` of package **surveyplanning** (**?**). However, this function is an implementation of an alternative, consistent estimator[2] of the population variance (**?**):

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

```
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 **??**). 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
```

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

```
          z
z 0.9481981
```

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

```
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 (**?**):

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

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

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

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.

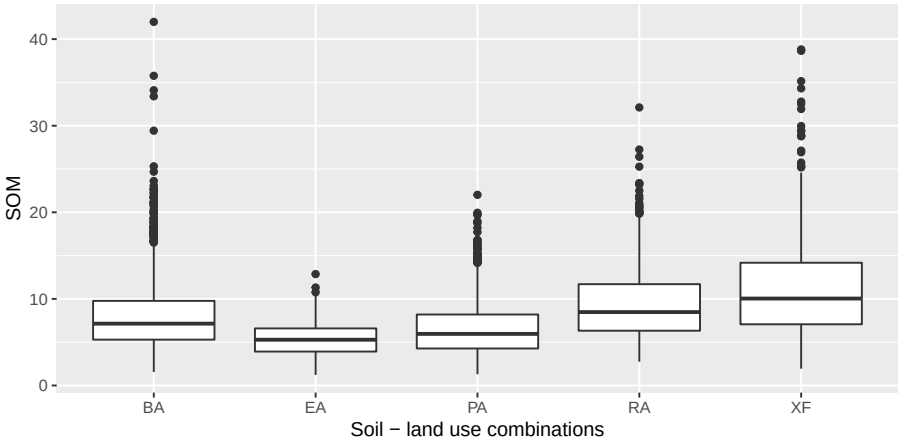

FIGURE 4.4: Boxplots of SOM per landuse-soil combination.

4.2 Confidence interval estimate

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

$$\hat{\bar{z}} \pm t_{\alpha/2,df} \cdot \sqrt{\widehat{V}\left(\hat{\bar{z}}\right)}\,, \tag{4.12}$$

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 Sattherwaites method (**?**):

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

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:

$$n_h = n \cdot \frac{N_h}{\sum N_h} \,, \tag{4.14}$$

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:

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

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:

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

with c_h the costs per sampling unit in stratum h. Optimal means that given the

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

total costs this allocation type leads to minimum sampling variance, assuming a linear costs model:

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

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 `optsize` of package **surveyplanning**.

```
labels <- sort(unique(mysample$stratum))
res <- optsize(labels,n,N_h,S2z_h)
round(res$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 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 **??**, 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.

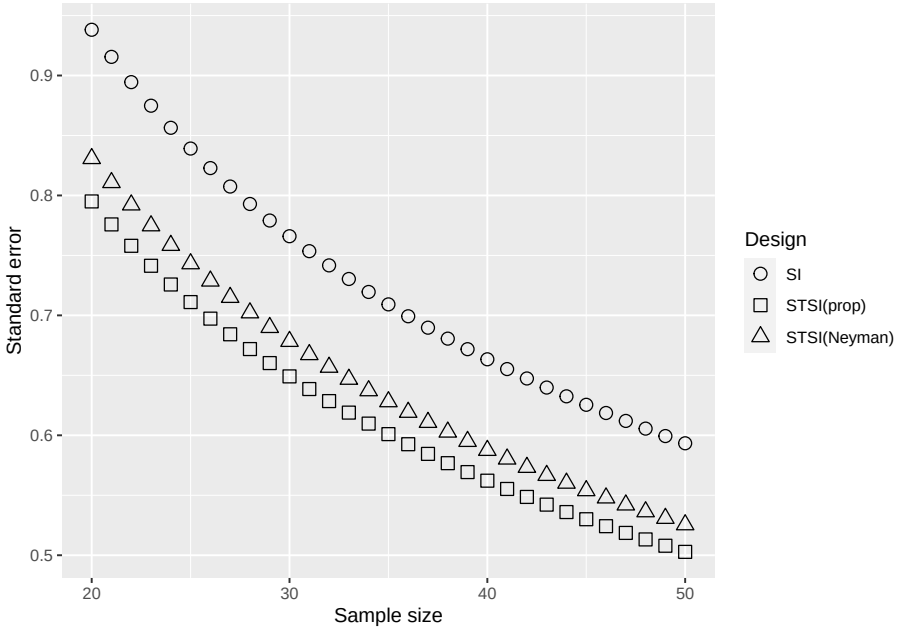

FIGURE 4.5: Standard error of estimated population mean as a function of the total sample size, for simple random sampling and stratified simple random sampling with proportional and Neyman allocation.

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** (**?**).

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 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.

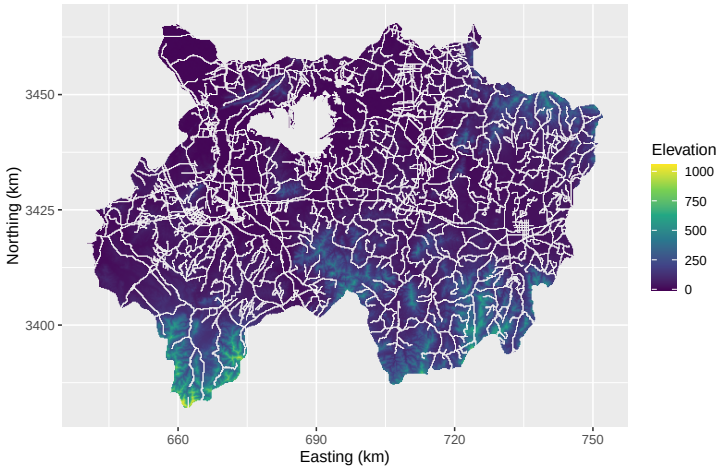

FIGURE 4.6: Elevation used as a stratification variable in cum-root-f stratification.

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.

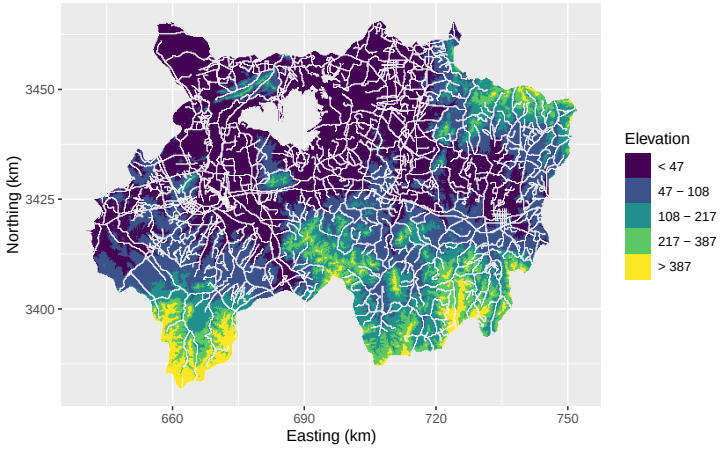

FIGURE 4.7: Strata obtained with cum-root-f method, using elevation as stratification variable.

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 subsampled from the original 1 km × 1 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.

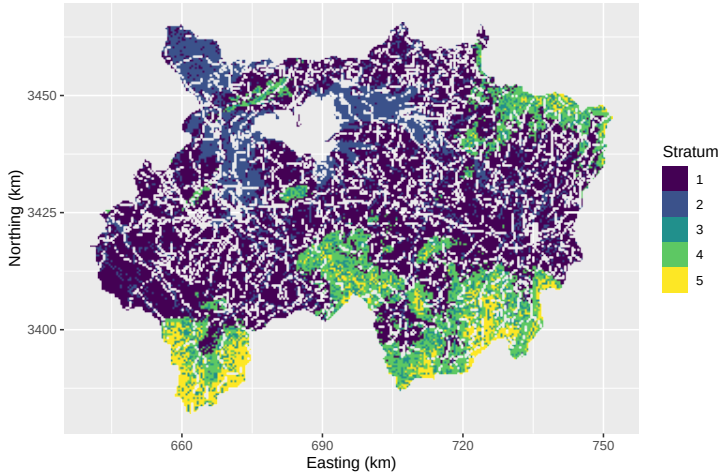

FIGURE 4.8: Five k-means clusters using five scaled covariates in clustering, to be used as strata in random sampling from study area Xuancheng.

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

Stratum	Nh	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

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 **??**.

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 **??** 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 **??**. 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)
```

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.

[3]If 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.

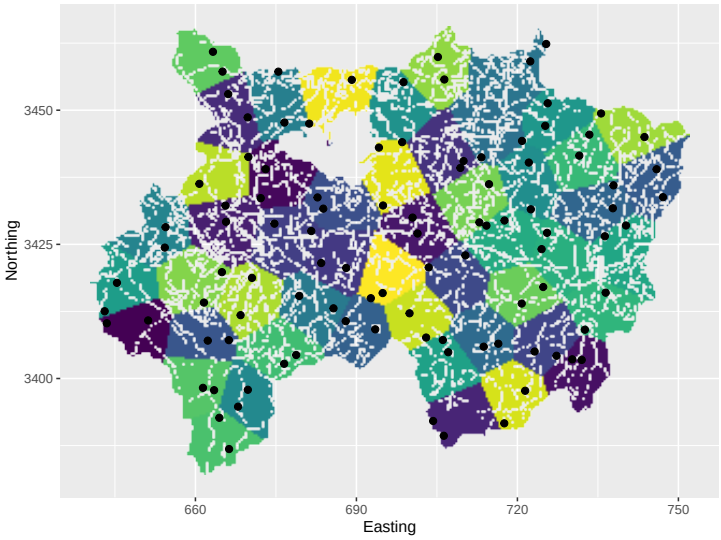

FIGURE 4.9: Compact geostrata of equal size in Xuancheng, and stratified simple random sample of two points per stratum.

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)
```

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 **??** 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.

```
library(SamplingStrata)
df <- data.frame(cd=lcd_kriged$var1.pred,
```

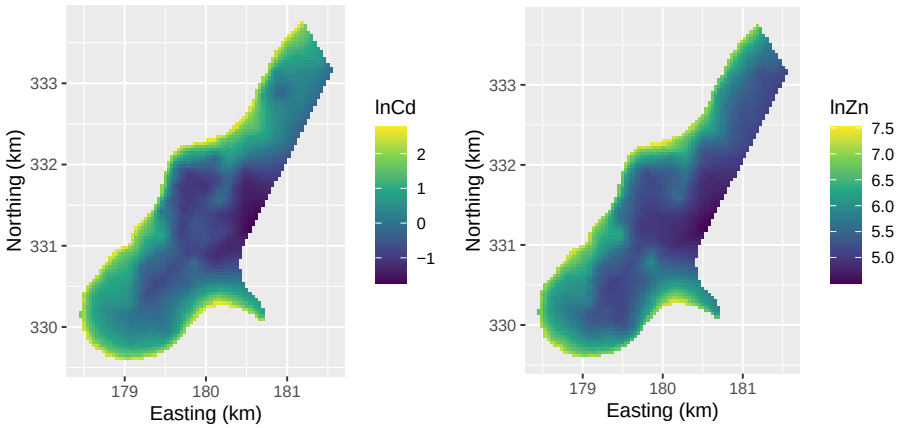

FIGURE 4.10: Kriging predictions of natural logarithms of Cd and Zn concentration in study area Meuse (Netherlands), used as stratification variable in bivariate stratification.

```
                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    5.576
2       2        694          5    1.421    2.090    4.950    6.010
3       3        597          3    2.091    2.630    5.163    6.358
4       4        704          6    2.630    3.476    5.472    6.802
5       5        391          5    3.480    4.781    6.234    7.527
```

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"))
```

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`.

```
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

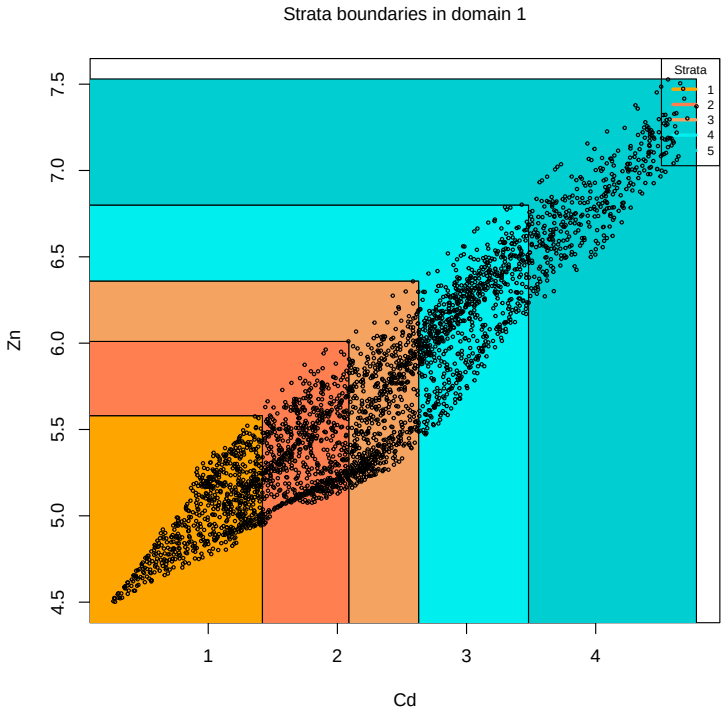

FIGURE 4.11: Optimised bivariate strata for study area Meuse.

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 **??** 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.

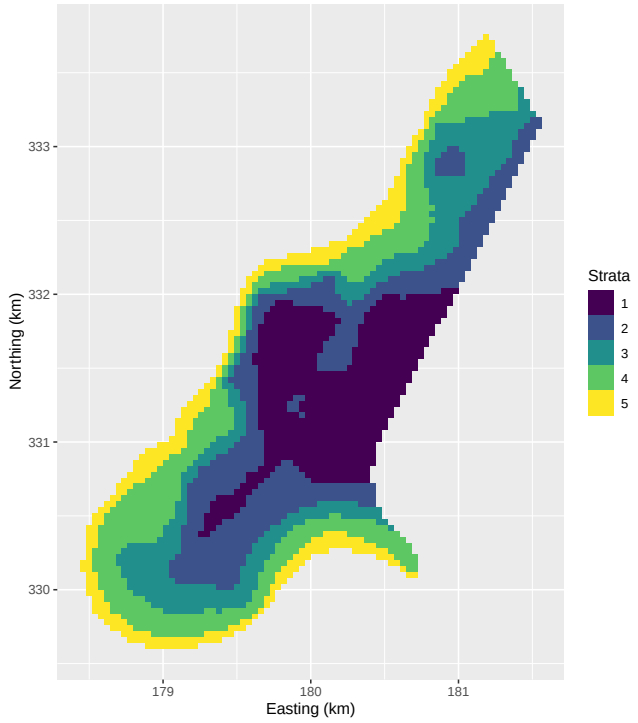

FIGURE 4.12: Optimised bivariate strata for study area Meuse.

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.

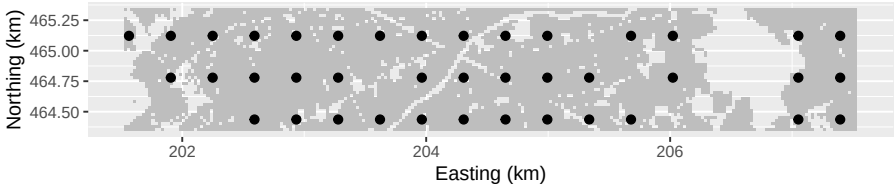

FIGURE 5.1: Systematic random sample (randomly placed square grid) from Voorst.

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 $A/spacing^2$.

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 compared 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.

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.

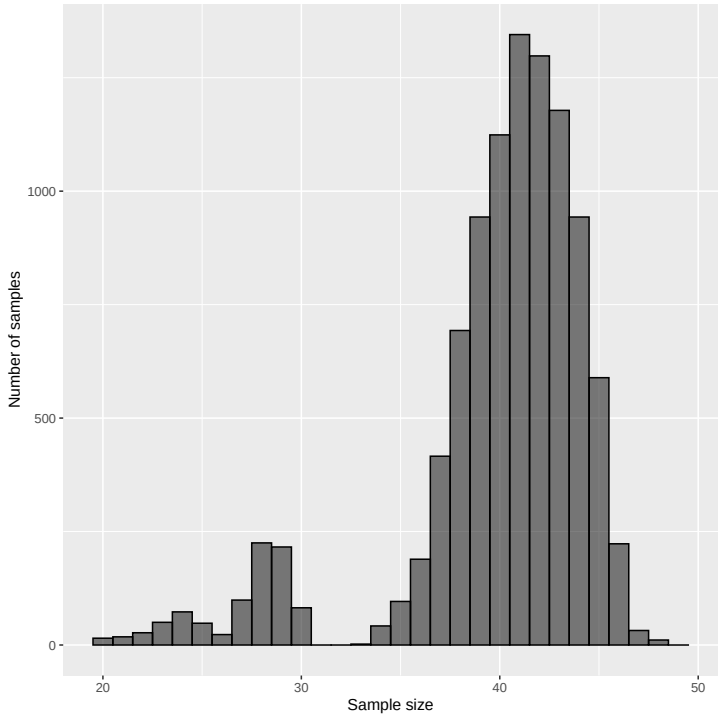

FIGURE 5.2: Sampling distribution of sample size of systematic random sampling.

```
dy <- 1000/3
dx <- A/(n*dy)
mySYsample_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

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

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 $\sqrt{3}/2\ dx$.

```
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)
```

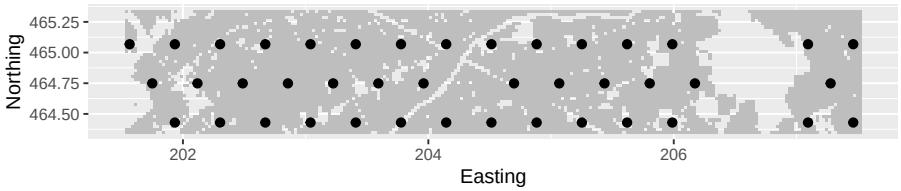

FIGURE 5.3: Systematic random sample (random triangular grid) from Voorst.

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

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

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

$$\hat{\bar{z}} = \sum_{k \in \mathcal{S}} \frac{z_k}{E[n]} \; . \tag{5.2}$$

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:

$$\hat{N} = \sum_{k \in \mathcal{S}} \frac{1}{\pi_k} = n \frac{N}{E[n]} \; . \tag{5.3}$$

This yields the ratio estimator of the population mean:

$$\hat{\bar{z}}_{\text{ratio}} = \frac{1}{n} \sum_{k \in \mathcal{S}} z_k \; . \tag{5.4}$$

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 swopped 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 swopped when the sum of the squared distances of the two units to their corresponding cluster centers is reduced. When the cluster ids are swopped, the centers are recomputed. The two loops are repeated until no swops 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
      }
  }
  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)
```

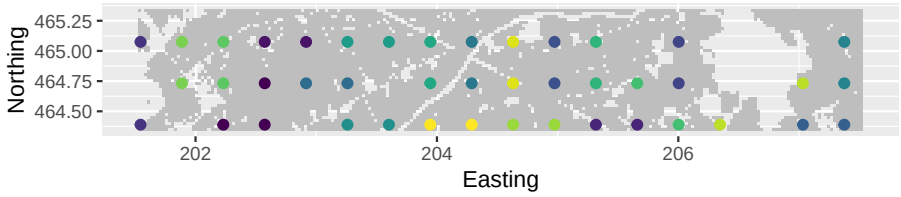

FIGURE 5.4: Clustering of grid points selected from Voorst for approximating the variance of the estimator of the population mean of SOM.

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

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

with $r = 0, 1, \ldots, R$ an index for the column-number and $s = 0, 1, \ldots, S$ 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:

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

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.

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.

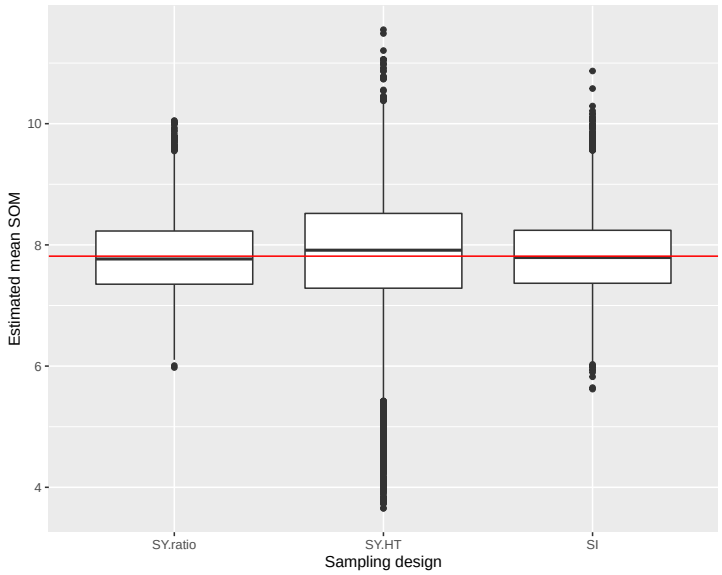

FIGURE 5.5: Sampling distribution of estimators of the population mean of SOM (g per kg) in Voorst, with systematic random sampling (square grid) and simple random sampling and an (expected) sample size of 40. With systematic random sampling both the π estimator (SY.HT) and the ratio estimator (SY.ratio)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

The 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 **??**.

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

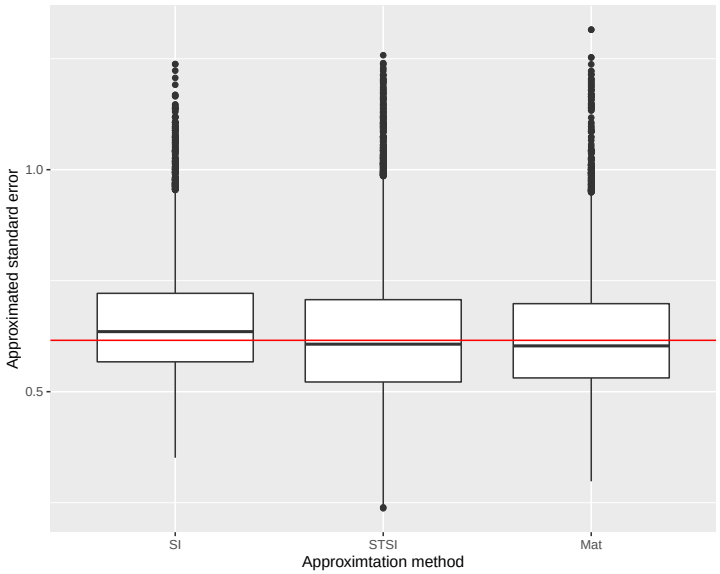

FIGURE 5.6: Sampling distribution of approximated standard error of the ratio estimator of the population mean of SOM (g per kg) in Voorst, with systematic random sampling (square grid) and an expected sample size of 40. Approximations are obtained by treating the systematic sample as a simple random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

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`).

- 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?.

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 $\times$ 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)
```

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
```

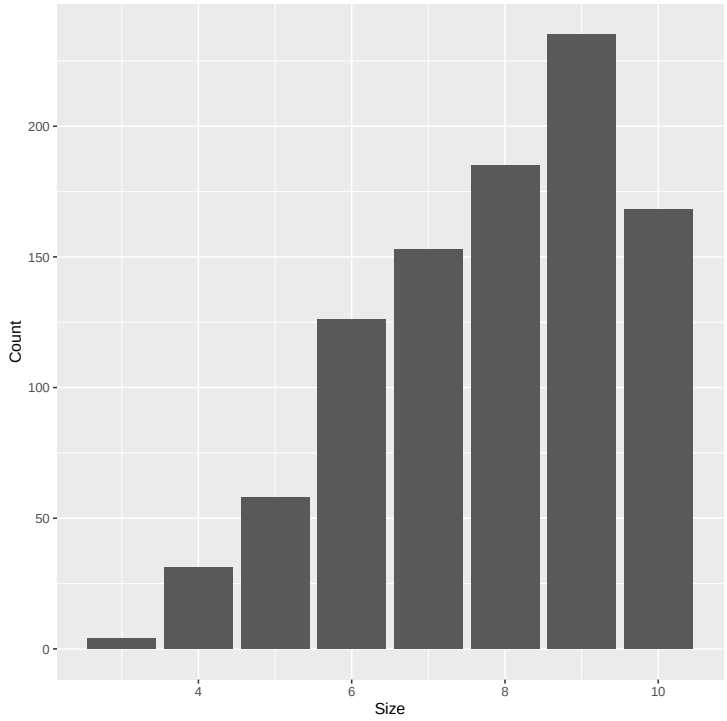

FIGURE 6.1: Bar plot of size (number of grid cells) of clusters: E-W oriented transects within zones, with an inter-point spacing of 100 m.

```
    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 westmost 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.

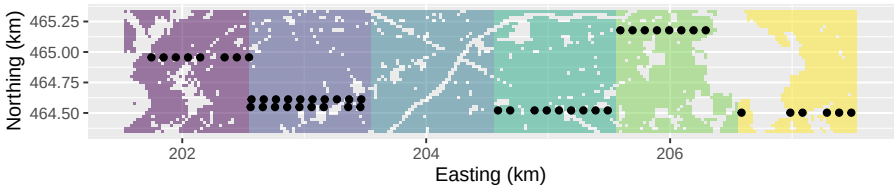

FIGURE 6.2: Cluster random sample from Voorst. Six times a cluster (transect) is selected with probabilities proportional to their size and with replacement

The output data frame of function `cl` 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:

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

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:

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

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

$$\bar{p}_j = \frac{M_j}{M}; \tag{6.3}$$

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

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

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:

$$\hat{\bar{\bar{z}}} = \frac{1}{n} \sum_{j \in \mathcal{S}} \bar{z}_j \,. \tag{6.5}$$

Note the two bars in $\hat{\bar{\bar{z}}}$, 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 **?**)

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

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 **?**)

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

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

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

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).

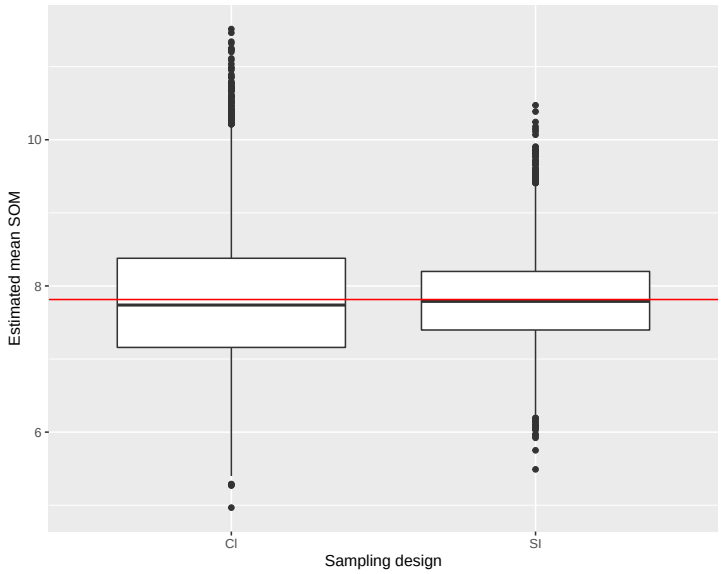

FIGURE 6.3: Sampling distribution of the pwr estimator of the mean of SOM (g/kg) in Voorst with cluster random sampling, and of the π estimator with simple random sampling, both designs with an (expected) sample size of 49 units.

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
```

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

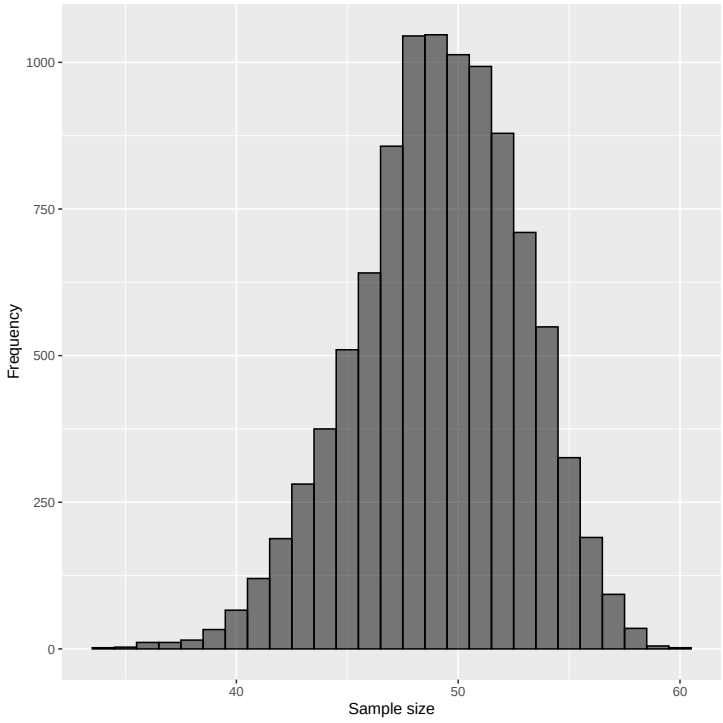

FIGURE 6.4: Sampling distribution of sample size with cluster random sampling.

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 × 1 km zones obtained by joining two neighbouring 1 km × 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 `UPpivotal` 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 <- UPpivotal(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 difficult[1]. 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`.

```
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:

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

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

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

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

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

$$\widehat{M} = \sum_{j \in \mathcal{S}} \frac{M_j}{\pi_j} = \frac{N}{n} \sum_{j \in \mathcal{S}} M_j \; . \tag{6.11}$$

This leads to the ratio estimator of the population mean:

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

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:

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

```
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:

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

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

$$\hat{b} = \frac{\frac{1}{n} \sum_{j \in \mathcal{S}} t_j}{\frac{1}{n} \sum_{j \in \mathcal{S}} M_j} \, . \tag{6.15}$$

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

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

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 km × 1 km zones, obtained by joining two neighbouring 1 km × 1 km zones (Figure 6.2). The clusters are the same as before, i.e. E-W oriented transects within 1 km × 1 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 km × 1 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)
}
```

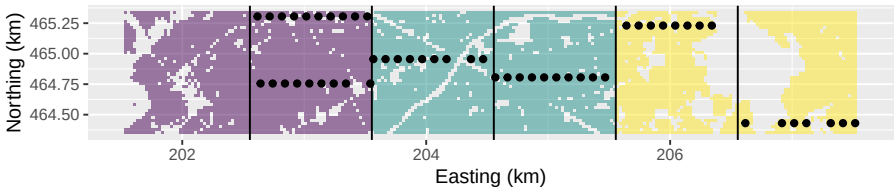

FIGURE 6.5: Stratified cluster random sample from Voorst, with three strata.

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?

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 km × 0.5 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)
  mypsusample <- sframe[units,psu]
```

```
  ssunits <- NULL
  for (psunit in mypsusample) {
    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.

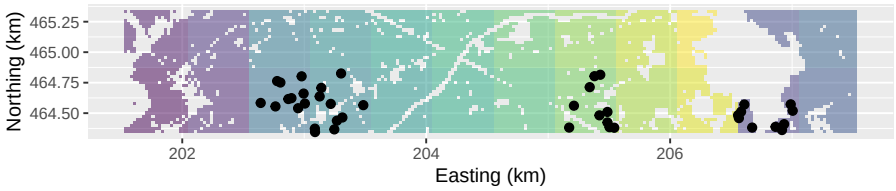

FIGURE 7.1: Two-stage cluster random sample from Voorst. Four times a psu is selected by ppswr. Each time a psu is selected, ten points are selected from that psu.

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:

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

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:

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

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 $\hat{\bar{\bar{z}}}$, 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 $m_j = m, j = 1, \ldots, N$) is equal to (**?**, Equation 11.33)[1]

[1]The equation in Cochran (1977) is the variance estimator for the population total. In

$$V(\hat{\bar{z}}) = \frac{S_{\mathrm{b}}^2}{n} + \frac{S_{\mathrm{w}}^2}{n\,m}\ , \tag{7.3}$$

with

$$S_{\mathrm{b}}^2 = \sum_{j=1}^{N} p_j \left(\bar{z}_j - \bar{z}\right)^2\ , \tag{7.4}$$

and

$$S_{\mathrm{w}}^2 = \sum_{j=1}^{N} p_j S_j^2 \tag{7.5}$$

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:

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

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

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

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

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

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 $\hat{\bar{z}}_j$ the estimated mean of psu j, and $\hat{\bar{\bar{z}}}$ 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 $\hat{\bar{z}}_j$, and the smaller its contribution to the variance.

Let us assume a linear model for the total costs: $C = c_0 + c_1 n + c_2 n\, m$, with c_0 the fixed costs, 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 (**?**)

$$n = \frac{1}{V_{\max}} \left(S_{\mathrm{w}} S_{\mathrm{b}} \sqrt{\frac{c_2}{c_1}} + S_{\mathrm{b}}^2 \right) \tag{7.9}$$

primary units, and

$$m = \frac{S_{\mathrm{w}}}{S_{\mathrm{b}}} \sqrt{\frac{c_1}{c_2}} \tag{7.10}$$

secondary units per primary unit.

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

$$n = \frac{C_{\max} S_{\mathrm{b}}}{S_{\mathrm{w}} \sqrt{c_1 c_2} + S_{\mathrm{b}} c_1} \,, \tag{7.11}$$

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 $M/(d\ m$. 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, deff="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 `clusOpt2` of **R** package **PracTools** (**?**, **?**). This function requires as input various variance measures, which can be computed with 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).

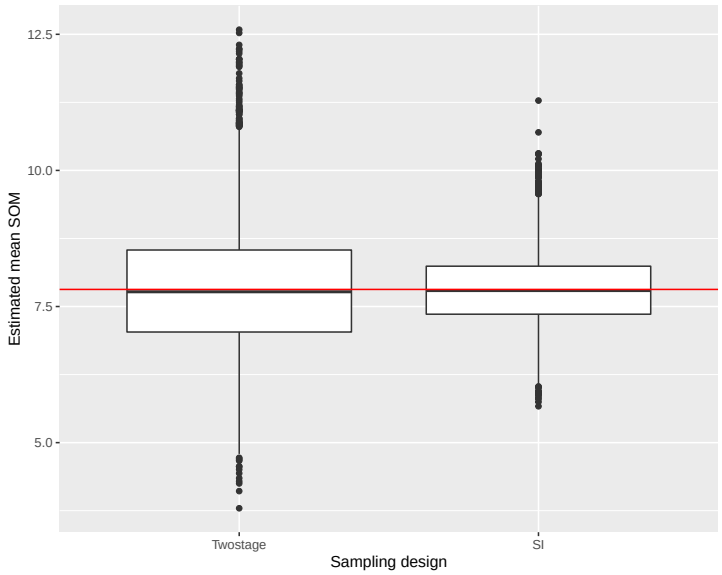

FIGURE 7.2: Sampling distribution of pwr estimator of mean of SOM (g/kg) in Voorst with two-stage random sampling, and of the π estimator with simple random sampling, bot design with a sample size of 40 units.

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 (**?**):

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

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 $m_j = m, j = 1, \ldots, N$, 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 **sampling** (**?**), 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 <- UPpivotal(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 $\pi_j = n/N, j = 1, \ldots, N$, and the population total can be estimated by (compare with Equation (6.9))

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

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:

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

```
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 km × 1 km blocks, as before in stratified cluster random sampling (Figure 6.2). The primary sampling units are 0.5 km × 0.5 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)
```

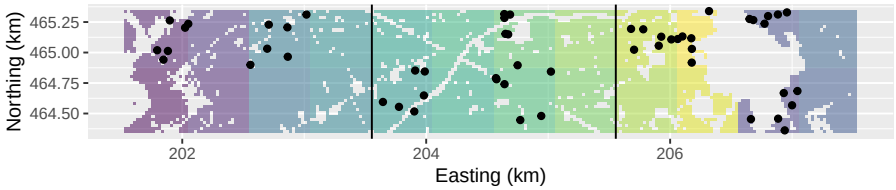

FIGURE 7.3: Stratified twostage random sample from Voorst. Strata are 2 km by 1 km blocks.

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
```

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 $\times$ 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 estimator by the pwr estimator:

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

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

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

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.

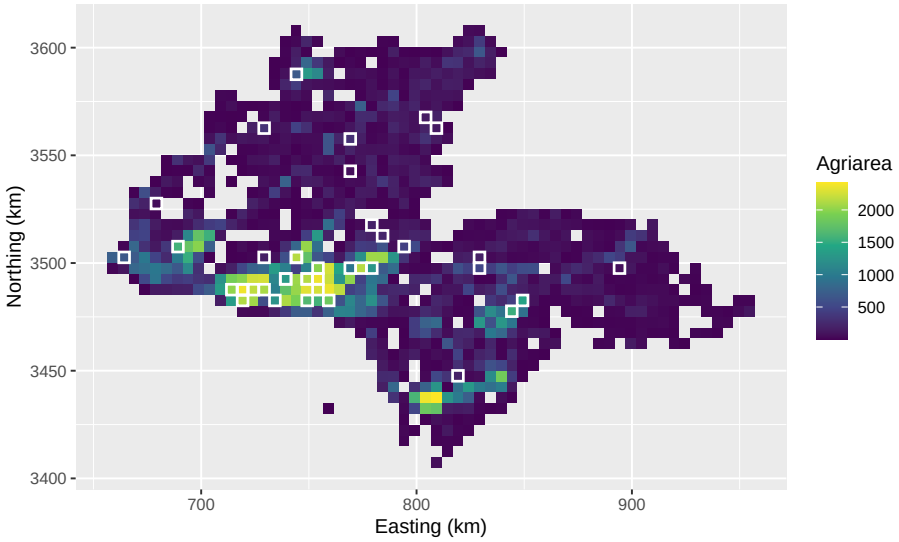

FIGURE 8.1: ppswr sample of size 40 from Kandahar, using agricultural area as a size variable. Four units are selected twice, so that the number of distinct units is 36.

The 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)
```

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)
svytotal(~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 πps sampling, stressing that the *inclusion* probabilities are proportional to a size variable, instead of the draw-by-draw selection probabilities. ppswor sampling starts with assigning 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 $\pi_k = n\, x_k / \sum_{j=1}^{N} x_j, k = 1 \ldots N$.

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
```

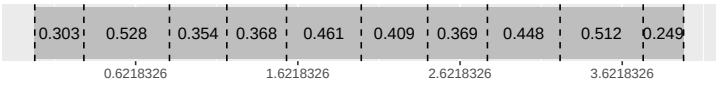

FIGURE 8.2: Systematic random sample along a line with unequal inclusion probabilities.

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 ppswor sampling with fixed order is that no unbiased estimator of the sampling variance exists.

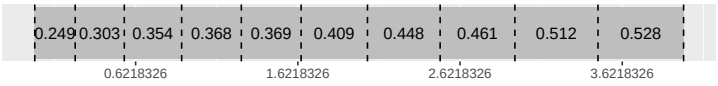

FIGURE 8.3: Systematic random sample along a line with unequal inclusion probabilities. Units are ordered by size.

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 becomes0 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

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

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

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

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:

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

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:

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

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)
svytotal(~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**).

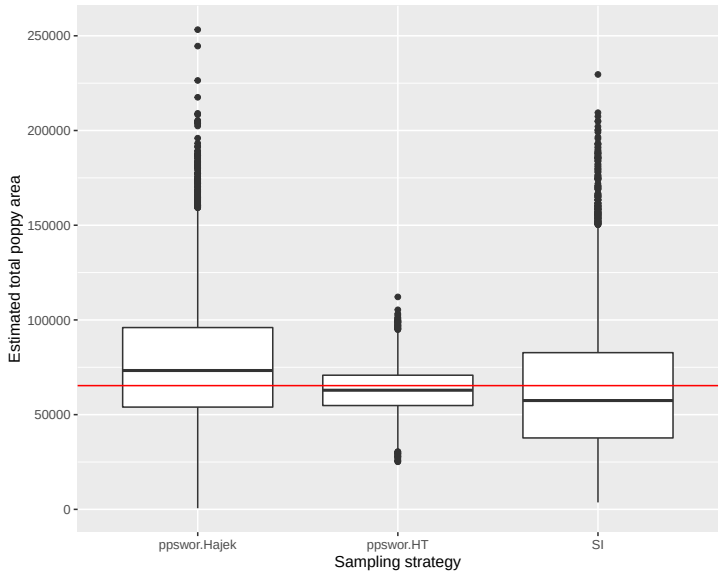

FIGURE 8.4: Sampling distribution of estimators of the total poppy area (ha) in Kandahar with ppswor sampling and simple random sampling without replacement (SI) of 40 units. In ppswor sampling the total poppy area is estimated by the π estimator (ppswor.HT) and Hajek estimator (ppswor.Hajek).

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 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 Kanda-

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

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 balancing 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.

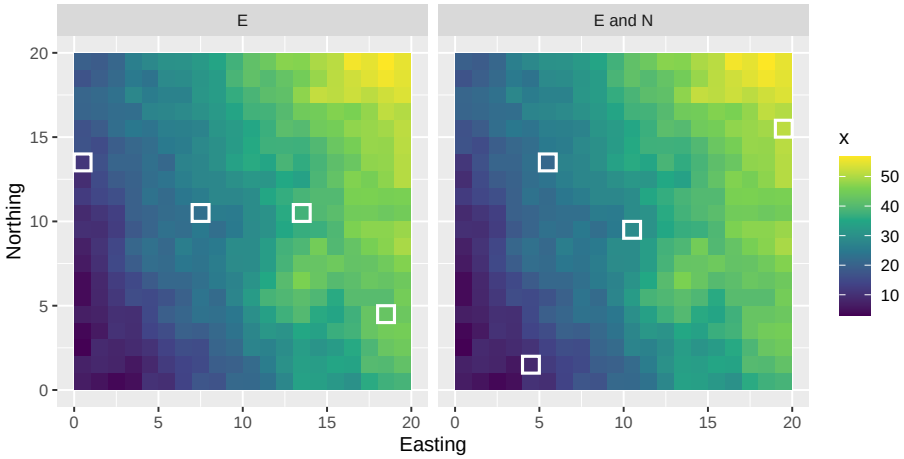

FIGURE 9.1: Sample balanced on Easting (E) and on Easting and Northing (E and N).

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.

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

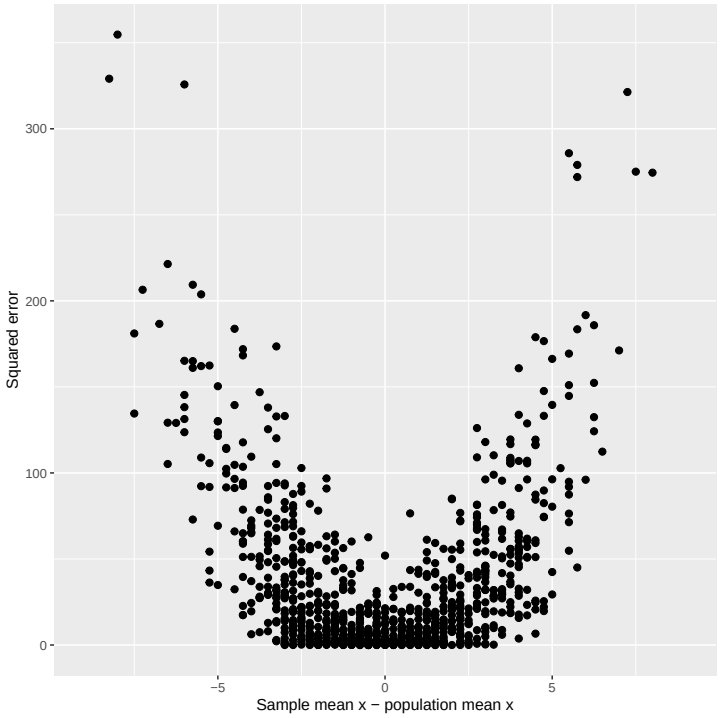

FIGURE 9.2: Squared error in estimated mean of z against difference between population and sample mean of a covariate.

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

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:

$$\sum_{k \in \mathcal{S}} \frac{x_k}{\pi_k} = \sum_{k=1}^{N} x_k \ , \tag{9.1}$$

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 **??**), 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 (**??**))

$$\hat{t}_{\text{regr}}(z) = \hat{t}_{\pi}(z) + \hat{b}\left(t(x) - \hat{t}_{\pi}(x)\right) \ , \tag{9.2}$$

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 **?**. The population total can be estimated by the π estimator:

$$\hat{t}(z) = \sum_{k \in \mathcal{S}} \frac{z_k}{\pi_k}. \tag{9.3}$$

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 (**?**, **?**)

$$\widehat{V}(\hat{\bar{z}}) = \frac{1}{N^2} \frac{n}{n-p} \sum_{k \in \mathcal{S}} c_k \left(\frac{\epsilon_k}{\pi_k}\right)^2 \ , \tag{9.4}$$

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

$$\epsilon_k = z_k - \mathbf{x}_k^{\mathrm{T}} \hat{\mathbf{b}} \ , \tag{9.5}$$

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

$$\hat{\mathbf{b}} = \left(\sum_{k \in \mathcal{S}} c_k \frac{\mathbf{x}_k}{\pi_k} \frac{\mathbf{x}_k}{\pi_k}^{\mathrm{T}} \right)^{-1} \sum_{k \in \mathcal{S}} c_k \frac{\mathbf{x}_k}{\pi_k} \frac{z_k}{\pi_k} \, . \tag{9.6}$$

Working this out for balanced sampling without replacement with equal inclusion probabilities, $\pi_k = n/N, k = 1, \ldots, N$, yields

$$\widehat{V}(\hat{\bar{z}}) = \frac{1}{n(n-p)} \sum_{k \in \mathcal{S}} c_k \epsilon_k^2 \, . \tag{9.7}$$

? 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 (lnSWIR2) 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 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.

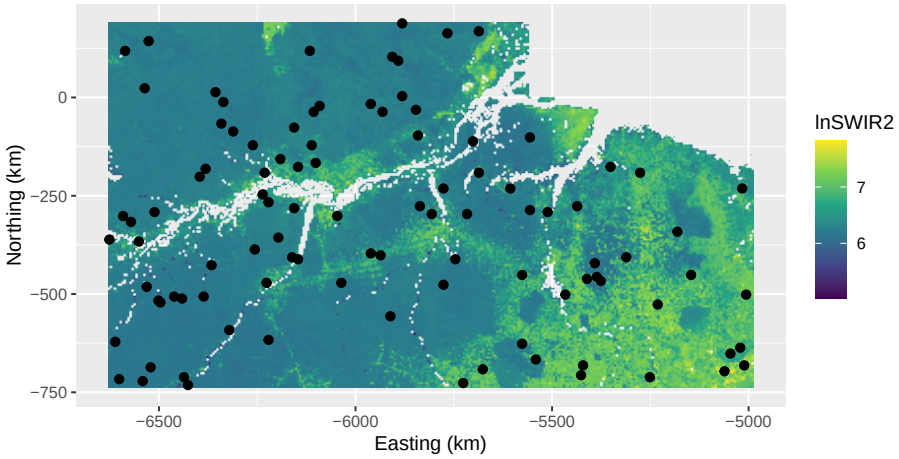

FIGURE 9.3: Balanced sample from Eastern Amazonia (Brazil), balanced on covariate lnSWIR2.

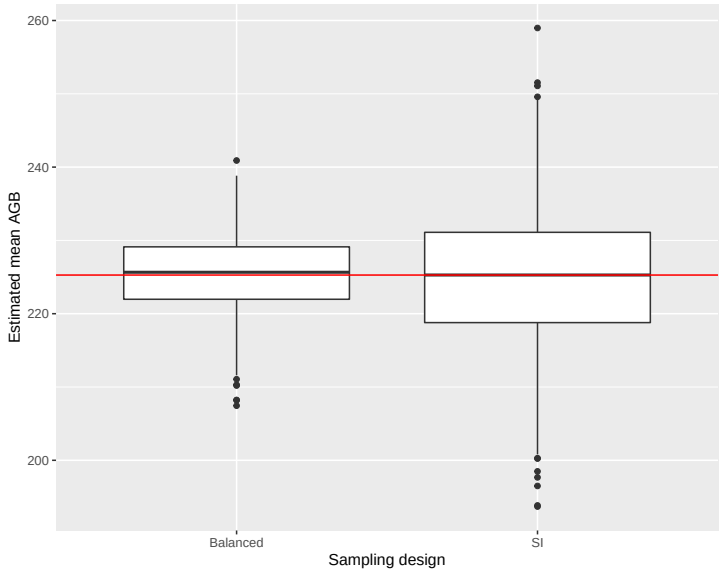

FIGURE 9.4: Sampling distribution of π estimator of the mean aboveground biomass in Eastern Amazonia, with balanced sampling (balanced on lnSWIR2) and simple random sampling, bot designs with a sample size of 100 units.

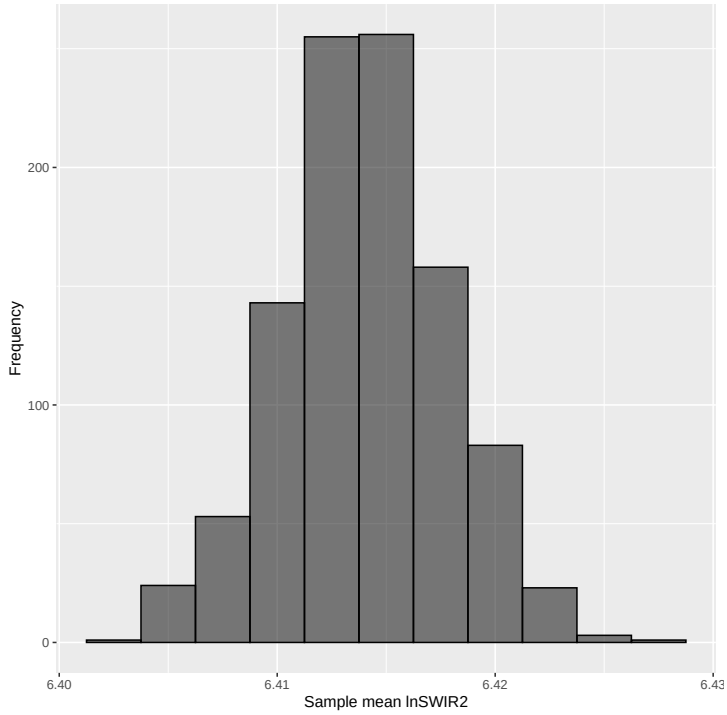

FIGURE 9.5: Sampling distribution of sample mean of balancing variable lnSWIR2.

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:

$$\begin{bmatrix} \pi_{1,1} & 0 & 0 & 0 \\ \pi_{2,1} & 0 & 0 & 0 \\ \pi_{3,1} & 0 & 0 & 0 \\ \pi_{4,1} & 0 & 0 & 0 \\ & \pi_{5,2} & 0 & 0 \\ 0 & \pi_{6,2} & 0 & 0 \\ 0 & 0 & \pi_{7,3} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \pi_{400,4} \end{bmatrix}, \tag{9.8}$$

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 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.

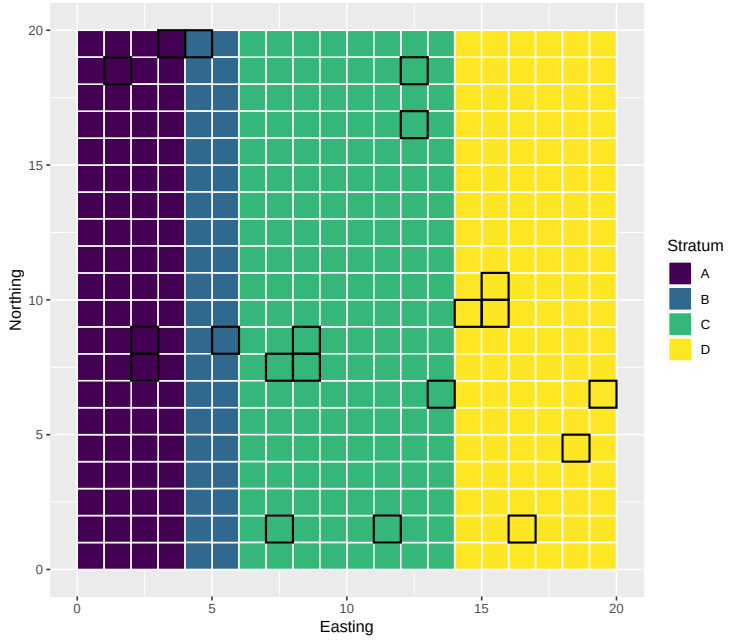

FIGURE 9.6: Sample balanced on a categorical variable with four classes.

As a first step inclusion probabilities are computed by $\pi_{hk} = n_h/N_h, k = 1, \ldots, N_h$, with $n_h = N_h/N$ (proportional allocation).

```
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 perfectly 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 $k = 1, \ldots, N$ 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).

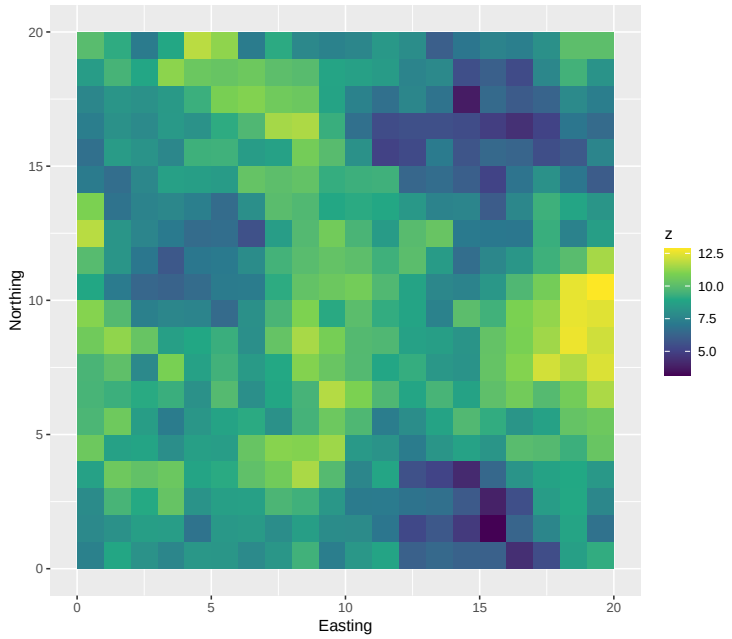

FIGURE 9.7: Simulated population, used for illustration of two-way stratified random sampling.

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 $\mathbf{X}$ 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.

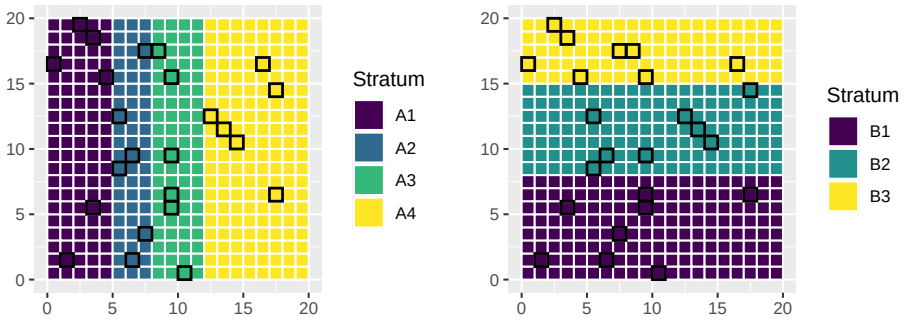

FIGURE 9.8: Two-way stratified sample.

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  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$pih, X=X[units,]/mysample$pih, c=c)
zpred <- X%*%b
e <- mysample$z-zpred[units]
n <- nrow(mysample)
v_tz <- n/(n-ncol(X))*sum(c*(e/mysample$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,]
```

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 (**?**)

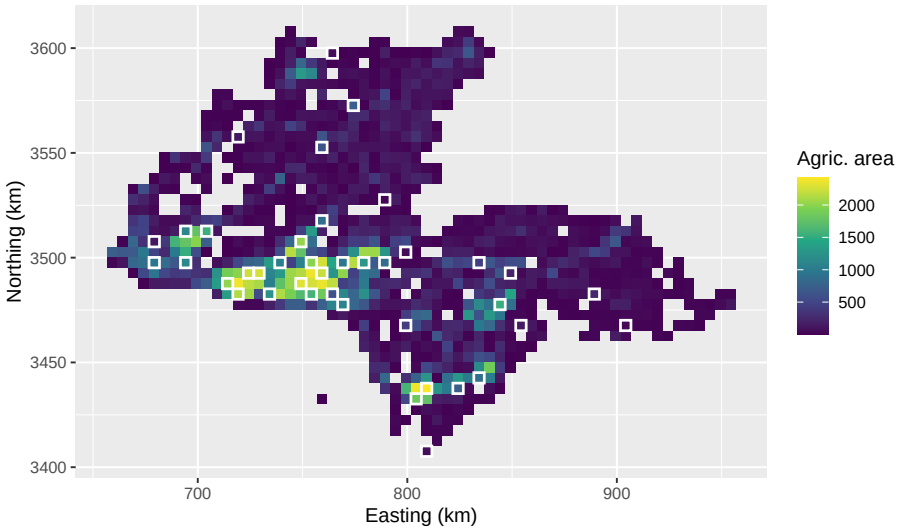

FIGURE 9.9: Spatial ppswor sample selected by local pivotal method, using agricultural area as size variable.

$$\widehat{V}(\hat{t}(z)) = \frac{1}{2}\sum_{k\in\mathcal{S}}\left(\frac{z_k}{\pi_k} - \frac{z_{k_j}}{\pi_{k_j}}\right)^2, \tag{9.9}$$

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 x 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 x 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.

```
set.seed(314)
ord <- sample.int(4,4)
myfinpop_rand <- NULL
```

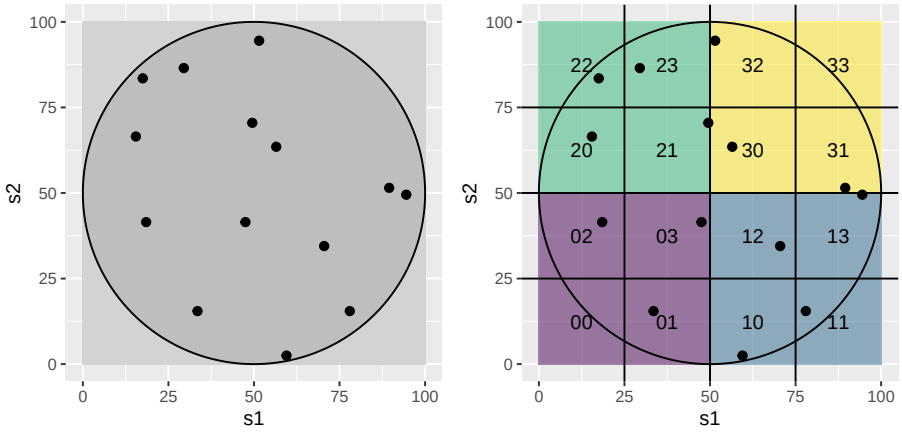

FIGURE 9.10: Numbering of grid cells and subcells for GRTS sampling.

```
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 9.12 shows a systematic random sample along a line with unequal inclusion probabilities. The inclusion probabilities are proportional to a size

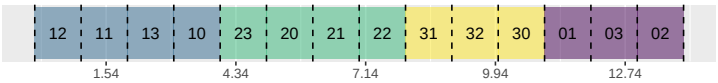

FIGURE 9.11: Systematic random sample along a line with equal inclusion probabilities.

variable, with values 1, 2, 3 or 4. The selected population units are the units in subcells 10, 20, 31, 01 and 02.

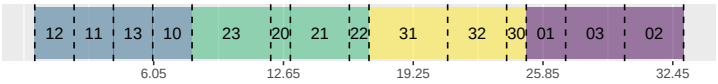

FIGURE 9.12: Systematic random sample along a line with inclusion probabilities proportional to size.

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,]
```

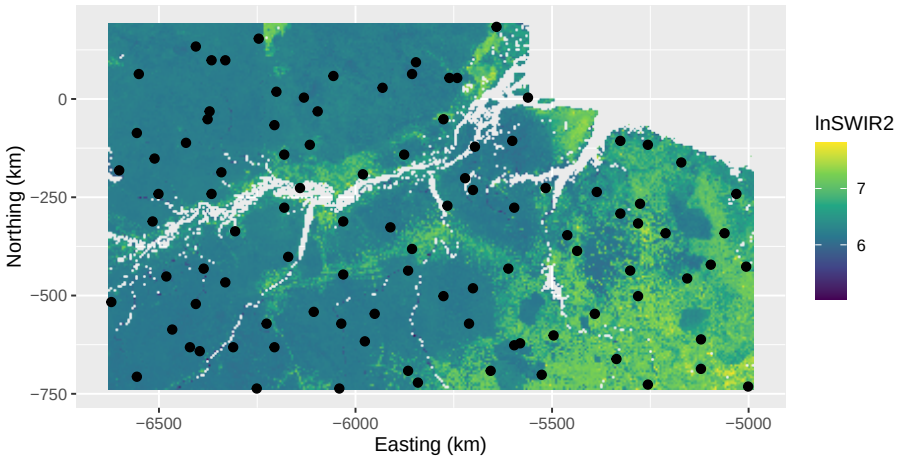

FIGURE 9.13: Balanced sample, balanced on lnSWIR2, with geograhical spreading from Eastern Amazonia (Brazil).

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, **?**)

$$\widehat{V}(\hat{\bar{z}}) = \frac{n}{n-p}\frac{p}{p+1}\sum_{k\in\mathcal{S}}(1-\pi_k)\left(\frac{\epsilon_k}{\pi_k} - \bar{\epsilon}_k\right)^2 , \tag{9.10}$$

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

$$\bar{\epsilon}_k = \frac{\sum_{j=1}^{p+1}(1-\pi_j)\frac{\epsilon_j}{\pi_j}}{\sum_{j=1}^{p+1}(1-\pi_j)} . \tag{9.11}$$

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$AGB/pi, X=Xbal[units,]/pi, c=c)
zpred <- Xbal%*%b
e <- mysample$AGB-zpred[units]
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.

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 **??**. 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 **??**. 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:

$$z_k = \mu(\mathbf{x}_k) + \epsilon_k \,, \tag{10.1}$$

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 $\mathbf{x}_k = (1, x_{1,k}, \ldots, x_{J,k})^{\mathrm{T}}$, and ϵ_k a random variable with zero mean. The model mean $\mu(\mathbf{x}_k)$ can be a lin-