Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Master's diploma thesis

in the field of study Mathematics

and specialisation Statistics and Data Analysis

OPTIMAL ALLOCATION IN STRATIFIED SAMPLING SCHEMES

Wojciech Wójciak

student record book number 199163

thesis supervisor

prof. dr hab. inż. Jacek Wesołowski

WARSAW 2019

....

supervisor's signature author's signature

Abstract

OPTIMAL ALLOCATION IN STRATIFIED SAMPLING SCHEMES

This dissertation aims at establishing necessary and sufficient conditions for classical problem of optimum sample allocation in stratified sampling scheme with simple random sampling without replacement design in each stratum. In the context of this thesis, an optimal allocation is in the classical Tschuprow-Neyman's (Tschuprow 1923; Neyman 1934) sense, and it satisfies additional lower and upper bounds restrictions imposed on sample sizes in strata. We formulate optimal allocation problem in the language of mathematical optimization, and then use Karush-Kuhn-Tucker conditions to derive necessary and sufficient conditions for the problem in subject. We also provide corresponding numerical algorithms, derived from established optimality conditions, including implementation in R programming language. The approach to the optimal allocation problem presented in this work is unique.

Keywords: survey sampling, stratified sampling, optimal allocation, Karush-Kuhn-Tucker conditions, Neyman optimal allocation, optimal allocation sufficient conditions.

Streszczenie

ALOKACJA OPTYMALNA W WARSTWOWYCH SCHEMATACH PRÓBKOWANIA

Celem pracy jest skonstruowanie warunków koniecznych i wystarczających dla rozwiązania optymalnego w problemie alokacji próby w warstwowych schematach próbkowania z losowaniem prostym bez zwracania w każdej z wartsw. W ramach tej pracy, alokacja optymalna rozumiana jest jako alokacja w sensie Tschuprowa-Neymana (Tschuprow 1923; Neyman 1934) i taka, dla której spełnione są dodatkowe ograniczenia nakładane na rozmiar próby na poziomie warstwy. Problem zostanie sformułowany w terminach optymalizacji matematycznej, a następnie, z wykorzystaniem warunków Karusha-Kuhna-Tuckera, zostaną wyprowadzone docelowe warunki optymalności. W pracy przedstawiono również algorytmy numeryczne, wraz z implementacją w języku programowania R, realizujące rozwiązanie problemu optymalnej alokacji, oparte na wyprowadzonych warunkach dostatecznych optymalności. Prezentowane w pracy podejście i metodologia są unikatowe w przedstawionym zastosowaniu.

Słowa kluczowe: metoda reprezentacyjna, losowanie warstwowe, optymalna alokacja, warunki Karusha-Kuhna-Tuckera, optymalna alokacja Neymana, warunki dostateczne alokacji optymalnej.

Warsaw, July 2, 2019

Declaration

I hereby declare that the thesis entitled „OPTIMAL ALLOCATION IN STRATIFIED SAMPLING SCHEMES”, submitted for the Master degree, supervised by prof. dr hab. inż. Jacek Wesołowski, is entirely my original work apart from the recognized reference.

....

Acknowledgments

I would like to thank my supervisor prof. dr hab. inż. Jacek Wesołowski for all his enthusiasm, help and support given whilst completing this disertation.

Contents

1. Introduction

1.1. Context and Notation

We start with defining the context of the optimal allocation problem, including basic notions and general terms used throughout this thesis.

Population

Consider a finite population, $\mathcal{U}$, consisting of N distinct and measurable units, where the i^{th} unit is represented by the label i, such that $\mathcal{U} = \{1, \ldots, i, \ldots, N\}$. Population $\mathcal{U}$ is partitioned into H subpopulations, called strata and denoted by $\{\mathcal{U}_h\}_{h \in J}$, where $J = \{1, \ldots, H\}$ is a set of strata indices. Index h will also be used as a synonymous of stratum $\mathcal{U}_h$, $h \in J$, whenever it follows from the context. A stratum $\mathcal{U}_h$ has size N_h, $h \in J$, and $\sum_{h \in J} N_h = N$. For a variable under study $\mathcal{Y}$ defined on $\mathcal{U}$, we write y_i to denote its value associated with population element $i \in \mathcal{U}$. The parameter of interest is the total of a given study variable $\mathcal{Y}$ in $\mathcal{U}$, i.e. $t = \sum_{i \in \mathcal{U}} y_i$. The standard deviation of variable $\mathcal{Y}$ in stratum $\mathcal{U}_h$ is denoted by S_h, and it assumes the form

$$S_h^2 = \frac{1}{N_h - 1} \sum_{i \in \mathcal{U}_h} (y_i - \overline{y}_h)^2, \qquad N_h \geq 2,$$

where $\overline{y}_h = \frac{1}{N_h} \sum_{i \in \mathcal{U}_h} y_i$, $N_h \geq 1$, $h \in J$.

Sample

We assume that the values y_i, $i \in \mathcal{U}$, and therefore t are unknown at the outset. To estimate unknown population total t of study variable $\mathcal{Y}$, we observe y_i for a subset of $\mathcal{U}$ called sample $\mathcal{S}$. More explicitly, in stratified sampling, a probability sample $\mathcal{S}_h$ of size n_h is selected from each stratum $\mathcal{U}_h$, $h \in J$ according to a chosen sampling design. In the context of this thesis, it is the *simple random sampling without replacement* design (abbreviated SI in [9, Chapter 3.3, p. 66]). Under this design, every sample of the fixed size receives the same probability of being selected. Selections between strata are independent. The resulting total sample $\mathcal{S} = \bigcup_{h \in J} \mathcal{S}_h$, and its size $n = \sum_{h \in J} n_h$. Of course, we require at least $0 < n_h \leq N_h$ for all $h \in J$. Nevertheless, additional constraints can be added to n_h, e.g. $0 < l_h \leq n_h \leq u_h \leq N_h$ for all $h \in J$.

Stratified π-estimator of the total t takes the form

$$\hat{t}_{st} = \sum_{h \in J} \frac{N_h}{n_h} \sum_{i \in \mathcal{S}_h} y_i.$$

It is unbiased and its variance is

$$D^2_{\hat{t}_{st}}(n_1, \ldots, n_H) = \sum_{h \in J} \frac{N_h^2 S_h^2}{n_h} - \sum_{h \in J} N_h S_h^2 \geq 0. \tag{1.1}$$

More details are given in [9, Result 3.7.2, p. 103]. The optimum sample allocation problem in stratified sampling scheme with simple random sampling without replacement design is formulated as the determination of vector $(n_1^*, \ldots n_H^*)$, that minimizes variance (1.1) under given constraints (*optimal allocation problem* in the sequel). The values of H, N_h, S_h, as well as n, l_h, u_h, $h \in J$ are assumed to be known throughout the thesis.

The following three abbreviations allow for compact formulation of numerous expressions used throughout the text, particularly in sections dedicated to algorithms

$$\begin{aligned} d_h &:= N_h S_h, \\ d_h^L &:= \frac{d_h}{l_h}, \\ d_h^U &:= \frac{d_h}{u_h}, \qquad h \in J. \end{aligned}$$

As indicated above, l_h and u_h denote lower and upper bound respectively, optionally imposed on the size n_h of sample $\mathcal{S}_h$, $h \in J$.

1.2. Background

Classical Tschuprow-Neyman optimal allocation [11], [8], minimizes variance (1.1) and it satisfies simple restriction imposed on overall sample size (i.e. $\sum_{h=1}^{H} n_h = n$), guaranteeing that the sample size $n_h > 0$ for all $h \in J$. It does not however ensure, that the allocated samples do not exceed strata sizes (i.e. $n_h \leq N_h$, $\forall h \in J$). Moreover, there might be a need to require a minimum sample size per stratum (i.e. $n_h \geq l_h > 0$, $\forall h \in J$). In many applications such additional constraints have to be considered. This topic got an attention in domain literature, and in consequence some algorithms have been proposed.

Recursive version of the Neyman allocation (*Recursive Neyman* in the sequel) is a standard operational approach to the problem with additional upper bounds equal to strata sizes (i.e. $n_h \leq u_h = N_h$, $\forall h \in J$). It repeatedly applies the Neyman's optimal allocation to step-wise reduced set of strata, as described in [9, Remark 12.7.1, p. 466]. Yet, despite the intuitive idea behind, there is no formal proof of its optimality in the domain literature. Stenger and Gabler, in their work from 2005, form necessary but not sufficient conditions for the optimal solution for the problem with upper bounds equal to strata sizes [10, Lemma 1, Appendix, p. 149]. The proof authors offered is based on properties of the gradient of the function under consideration. Interestingly, an algorithm finding optimal solution to the optimal allocation problem, can be derived directly from the course of that proof. Another algorithm - the *noptcond* proposed by Gabler, Ganninger and Munnich in 2012 [5, Section 3 Programme code, p. 158], addresses the optimal allocation problem with any feasible lower and upper bounds (i.e. $0 < l_h \leq n_h \leq u_h \leq N_h, \forall h \in J$). Apart from what is stated in that paper, the *noptcond* algorithm does not always yield an optimal solution, as it will be shown in Chapter 3.

It seems natural to formulate the optimal allocation problem in the language of mathematical optimization[1] and then use methods pertained to this domain to derive optimality conditions for an optimal solution. Optimality conditions, which are often given as closed-form expressions, are fundamental to the analysis of an optimization problem. They constitute trustworthy underlay for the development and the analysis of effective algorithms. Namely, algorithms recognize solutions by checking whether they satisfy various optimality conditions and terminate when such conditions hold. This elegant strategy has evident advantages over existing approaches, yet it does not seem to have received an attention in survey sampling literature it deserves.
The method of Lagrange multipliers is the major workhorse in mathematical optimization. It is however limited only to the problems with equality constraints. The Karush-Kuhn-Tucker approach to constrained optimization generalizes the method of Lagrange multipliers by allowing for inequality constraints. Furthermore, the problem being a subject of this thesis belongs to special types of problems termed *convex problem.* Convex problems have well established treatment in mathematical optimization due to many desirable properties they possess. For instance, for convex problem (with some minor regularity conditions) the Karush-Kuhn-Tucker conditions are necessary and sufficient.

[1]Appendix A

1.3. Thesis Outline

The structure of this thesis arises from constraints imposed on a sample size n_h in stratum $\mathcal{U}_h$, $h \in J$, i.e. (1) only upper bounds ($0 < n_h \leq u_h \leq N_h$, $\forall h \in J$) are considered in Section 2, and (2) lower and upper bounds ($0 < l_h \leq n_h \leq u_h \leq N_h$, $\forall h \in J$) are considered in Section 3. For each of these two scenarios: (a) necessary and sufficient optimality conditions will be formulated, (b) corresponding algorithms will be presented, including new algorithms as well as modifications of those already existing. In particular, in Section 2.3.4 we provide precise description of the *Recursive Neyman* algorithm with generalization of upper bounds $u_h \leq N_h$ for all $h \in J$, together with the formal proof of its optimality. In Section 3.3, we show that the *noptcond* algorithm proposed by Gabler, Ganninger and Munnich in [5, Section 3 Programme code, p. 158], is not optimal in general. We modify this algorithm so that it meets not only necessary but also sufficient requirements for an optimal solution. Selected algorithms proposed in this thesis were implemented in R programming language and attached to the thesis.

2. Finite Upper Bounds and no Lower Bounds

2.1. Problem Formulation

Consider an optimum allocation problem where sample of total size n is allocated among strata, in such a way that from each stratum a sample of size n_h is drawn, and the size of the sample n_h must not exceed imposed upper bound u_h for all $h \in J$. This problem is formally written as follows.

Problem 2.1.

$$\min_{(n_1,\dots n_H)\in\mathbf{\Omega}} \quad D^2_{\hat{t}_{st}}(n_1,\dots,n_H), \quad \mathbf{\Omega} = \mathbb{R}^H_+,$$

subject to

$$\sum_{h=1}^{H} n_h - n = 0, \tag{2.1}$$

$$n_h - u_h \le 0, \qquad h = 1,\dots,H, \tag{2.2}$$

where function $D^2_{\hat{t}_{st}}$ is given by (1.1), and n, u_h, $h \in J$ are known constants. Following the sampling design, we shall make throughout, a natural assumptions about known constants, i.e. $n \le \sum_{i\in J} u_i$, and $0 < u_h \le N_h$ for all $h \in J$. Without the first assumption the problem might be infeasible. The set constraint $\mathbf{\Omega}$ was not placed on the list of functional constraints, assuming that the solution is in the interior of $\mathbf{\Omega}$. The proof of Theorem 2.4 justifies this assumption.

Remark 2.2. Optimization Problem 2.1 is a convex optimization problem[1] since the objective function $D^2_{\hat{t}_{st}}$ and inequality constraint functions $n_h - u_h$ of n_h, for all $h \in J$ are convex, equality constraint function $\sum_{h=1}^{H} n_h - n$ of $(n_1,\dots,n_H)$ is affine.

Proposition 2.3. The optimal solution to Problem 2.1 exists and it is globally unique.

Proof. Problem 2.1 is feasible under assumptions we made, i.e. $n \le \sum_{i\in J} u_i$, and $0 < u_h \le N_h$

[1] Appendix A.1, A.2

for all $h \in J$. Then, the existence of the optimal solution is guaranteed by the Extreme Value Theorem[2], as the feasible region is compact subset of $\mathbf{\Omega}$ (equivalent to closed and bounded for Euclidean space, following Heine–Borel Theorem) and objective function $D^2_{\hat{t}_{st}}$ is continuous on $\mathbf{\Omega}$. Moreover, since objective function $D^2_{\hat{t}_{st}}$ is strictly convex, the optimal solution is unique and global optimum solution. $\square$

2.2. Optimality Conditions

In this section, in Theorem 2.4 below, we formulate necessary and sufficient conditions for an optimal solution to Problem 2.1, that are the basis for several algorithms. Next, we draw additional optimality properties (Proposition 2.8), that could possibly further simplify the algorithms.

Theorem 2.4. $(n_1^*, \dots, n_H^*)$ is the optimal solution to Problem 2.1 for $n < \sum_{i \in J} u_i$, if and only if there exists set $\emptyset \neq J_* \subseteq J$, and number $\lambda^* \in \mathbb{R}^+$ such that

$$\forall h \in J_* : \quad n_h^* = \frac{n - \sum\limits_{i \in J_*^c} u_i}{\sum\limits_{i \in J_*} d_i} d_h < u_h, \tag{2.3}$$

$$\forall h \in J_*^c : \quad n_h^* = u_h, \tag{2.4}$$

$$\forall h \in J_*,\ \forall i \in J_*^c : \quad \frac{d_h}{n_h^*} = \sqrt{\lambda^*} \leq \frac{d_i}{u_i}, \tag{2.5}$$

where $J_*^c = J \setminus J_*$.

The proof of Theorem 2.4 we offer below is based on Karush-Kuhn-Tucker (KKT) conditions. More details on KKT conditions are given in Appendix B.

Proof. Let the partial derivative of function (1.1) with respect to the variable n_h be denoted by φ_h, and its inverse function by φ_h^{-1}, i.e.

$$\begin{aligned} \varphi_h : \ \mathbb{R}_+ \to \mathbb{R}_- \quad & \varphi_h(n_h) = \frac{\partial}{\partial n_h} D^2_{\hat{t}_{st}}(n_1, \dots, n_H) = -\frac{d_h^2}{n_h^2}, \\ \varphi_h^{-1} : \ \mathbb{R}_- \to \mathbb{R}_+ \quad & \varphi_h^{-1}(x) = \frac{d_h}{\sqrt{-x}}, \quad \forall h \in J. \end{aligned} \tag{2.6}$$

Problem 2.1 is a convex optimization problem (Remark 2.2) with objective and constraint functions of class C^1 on $\mathbf{\Omega}$. Therefore, the Karush-Kuhn-Tucker conditions are necessary and sufficient, and they take the following form for Problem 2.1

[2]Appendix A.3 and A.4

$\exists \lambda^* \in \mathbb{R}$, $\exists (\mu_1^*, \dots, \mu_H^*) \in \mathbb{R}^H$ such that for all $h \in J$

$$\begin{aligned}
\varphi_h(n_h^*) + \lambda^* + \mu_h^* &= 0, && (\textit{stationarity}) \\
\sum_{i \in J} n_i^* - n &= 0, && (\textit{primal feasibility}) \\
n_h^* - u_h &\leq 0, && (\textit{primal feasibility}) \\
\mu_h^*(n_h^* - u_h) &= 0, && (\textit{comp. slackness}) \\
\mu_h^* &\geq 0. && (\textit{dual feasibility})
\end{aligned} \tag{2.7}$$

The *primal feasibility* inequality condition can be divided into two distinct cases, leading to the following optimality conditions equivalent to (2.7):
$\exists J_* \subseteq J$, $\exists \lambda^* \in \mathbb{R}^+$ such that

$$n_h^* < u_h \quad \text{and} \quad \varphi_h(n_h^*) = -\lambda^*, \quad \forall h \in J_*, \tag{2.8}$$

$$n_h^* = u_h \quad \text{and} \quad \varphi_h(u_h) \leq -\lambda^*, \quad \forall h \in J_*^c, \tag{2.9}$$

$$\sum_{i \in J_*} n_i^* + \sum_{i \in J_*^c} u_i - n = 0, \tag{2.10}$$

where $J_*^c = J \setminus J_*$.
Conditions (2.8) - (2.10) can be further shortened when $J_* \neq \emptyset$. This condition is equivalent to $n < \sum_{i \in J} u_i$. From now on, we narrow the domain of possible solutions to $n_h^* > 0$, $\forall h \in J$. This restriction does not invalidate the proof, since functions φ_h are even functions on their natural domains $\mathbb{R} \setminus \{0\}$. It follows from (2.8) that for $n_h^* > 0$, we get $n_h^* = \varphi_h^{-1}(-\lambda^*)$ for all $h \in J_*$, and therefore

$$\sum_{i \in J_*} n_i^* = \sum_{i \in J_*} \varphi_i^{-1}(-\lambda^*) = \sum_{i \in J_*} \frac{d_i}{\sqrt{\lambda^*}} = \frac{1}{\sqrt{-\varphi_h(n_h^*)}} \sum_{i \in J_*} d_i = \frac{1}{\sqrt{\frac{d_h^2}{(n_h^*)^2}}} \sum_{i \in J_*} d_i = \frac{n_h^*}{d_h} \sum_{i \in J_*} d_i,$$

$$n_h^* = \frac{\sum_{i \in J_*} n_i^*}{\sum_{i \in J_*} d_i} d_h, \quad \forall h \in J_*.$$

Given (2.10)

$$n_h^* = \frac{n - \sum_{i \in J_*^c} u_i}{\sum_{i \in J_*} d_i} d_h, \quad \forall h \in J_*. \tag{2.11}$$

Furthermore, (2.9) and (2.8) read
$\forall (h, i) \in J_* \times J_*^c$, $\exists \lambda^* \in \mathbb{R}^+$ such that

$$\varphi_i(u_i) \leq -\lambda^* = \varphi_h(n_h^*),$$

i.e.

$$\frac{d_h}{n_h^*} = \sqrt{\lambda^*} \leq \frac{d_i}{u_i}. \tag{2.12}$$

Due to (2.11) and (2.12), optimality conditions (2.8) - (2.10) can be formulated as in Theorem 2.4.

To complete the proof, it should be noted that optimality conditions (2.3) - (2.4), and $n_h^* = \varphi_h^{-1}(-\lambda^*) > 0$ for all $h \in J_*$ guarantee that the optimal solution $(n_1^*, \ldots, n_H^*)$ is in the interior of $\boldsymbol{\Omega}$. □

Remark 2.5. (to the proof of Theorem 2.4)
Optimality conditions (2.3) - (2.5) were derived from KKT conditions preserving equivalence if and only if $J_* \neq \emptyset$. This requirement comes from the fact that the optimality condition (2.10) was enclosed into (2.3). Hence, if $J_* = \emptyset$, then (2.3) decays and condition (2.10) may possibly be violated resulting in non-optimal solution.

Proposition 2.6. Consider Theorem 2.4. In case of $n = \sum_{i \in J} u_i$, set $J_* = \emptyset$, and therefore $J_*^c = J$. Optimality conditions (2.3) - (2.5) are replaced with

$$\begin{aligned} \forall h \in J: \quad & n_h^* = u_h, \\ & \sum_{i \in J} n_h^* - n = 0, \end{aligned}$$

in order to preserve sufficiency. In other words

$$(n_1^*, \ldots, n_H^*) = (u_1, \ldots, u_H) \;\Leftrightarrow\; n = \sum_{i=1}^{H} u_i,$$

which is rather an intuitive result without the formalism of the KKT conditions.

Proof. The proof is an immediate conclusion of Remark 2.5. In order to preserve sufficiency of the optimality conditions stated in Theorem 2.4, condition (2.10) should explicitly be evaluated in place of condition (2.3) when $J_* = \emptyset$. Moreover, in case of $J_* = \emptyset$, condition (2.5) assumes the form $\sqrt{\lambda^*} \leq \frac{d_i}{u_i}$ and it therefore vanishes. □

Proposition 2.7. In Theorem 2.4, in case of overall sample size $n < \sum_{i \in J} u_i$, set $J_* \neq \emptyset$, and number $\lambda^* \in \mathbb{R}^+$ exist and are unique, i.e. there is one and only one non-empty subset J_* of set J, and number λ^* corresponding to the optimal solution. In the remaining case, i.e. $n = \sum_{i \in J} u_i$, set $J_* = \emptyset$, $J_*^c = J$

Proof. The proof is an immediate conclusion from Proposition 2.3 and distinctness of the *primal feasibility* condition (2.7) split, made in the course of the proof of Theorem 2.4. The extraction of subset J_* from J is equivalent of that split. □

Theorem 2.4 splits set of strata indices J into two subsets, Proposition 2.8 below, reveals the property bound up with these subsets, that might be used by the algorithms to Problem 2.1.

Proposition 2.8. Let J_*, J_*^c be the sets as in Theorem 2.4. The following property holds

$$\frac{d_h}{u_h} < \frac{d_i}{u_i}, \quad \forall (h,i) \in J_* \times J_*^c.$$

Proof. Let function φ_h be given by (2.6). Since φ_h is strictly increasing, (2.8) yields

$$\varphi_h(u_h) > -\lambda^*, \quad \forall h \in J_*, \tag{2.13}$$

and due to (2.9)

$$\varphi_i(u_i) \leq -\lambda^*, \quad \forall i \in J_*^c. \tag{2.14}$$

Inequalities (2.13) and (2.14) imply $\varphi_h(u_h) > \varphi_i(u_i)$, i.e.

$$-\frac{d_h^2}{u_h^2} > -\frac{d_i^2}{u_i^2} \equiv \frac{d_h}{u_h} < \frac{d_i}{u_i}, \quad \forall (h,i) \in J_* \times J_*^c.$$

□

Proposition 2.8 reduces number of candidates to J_* from naive $\sum_{k=0}^{H} \binom{H}{k} = 2^H$ (all subsets of J) to H. More explicitly, kth candidate is equal $\{j(k), \dots, j(H)\} \subseteq J,\ k = 1, \dots, H$, where j is permutation of $J = \{1, \dots, H\}$ such that $\frac{d_{j(1)}}{u_{j(1)}} \geq \frac{d_{j(2)}}{u_{j(2)}} \geq \cdots \geq \frac{d_{j(H)}}{u_{j(H)}}$.

2.3. Algorithms

Two algorithms solving Problem 2.1 will be presented in this section. The first algorithm *Sequential Allocation* is a direct consequence of Theorem 2.4 and Proposition 2.8 and it comes with three different versions. They differ between themselves in the way they find set of indices J_*. The third version of the *Sequential Allocation* algorithm is probably the most interesting. It forms a sequence of numbers on the basis of optimality condition (2.3) and examines its monotonicity in order to find J_*. The second algorithm, *Recursive Neyman*, has been known among practitioners for years, and its principles are outlined e.g. in [9, Remark 12.7.1, p. 466]. We first provide precise description of this algorithm for the case when upper bounds $u_h \leq N_h$, for all $h \in J$, and afterwards we use Theorem 2.4 to demonstrate its correctness. Analogies of *Recursive Neyman* allocation to the *Sequential Allocation* algorithm will be discussed as well.

The *Sequential Allocation* algorithm makes use of Proposition 2.8. It is therefore convenient to work with relabelled[3] strata indices, i.e. $j(k) \mapsto k$ for $k = 1, \dots, H$, and denote the set of relabelled strata indices by $\widetilde{J}$. Here, j is permutation of $J = \{1, \dots, H\}$ such that $d^U_{j(1)} \geq d^U_{j(2)} \geq \dots \geq d^U_{j(H)}$. Corresponding sets J_*, J^c_*, introduced in Theorem 2.4 will be relabelled accordingly, and denoted by $\widetilde{J}_*$, $\widetilde{J}^c_*$.

2.3.1. Sequential Allocation (version 0)

Theorem 2.4 provides closed-form expression for an optimal solution to Problem 2.1. Whereas construction of set J_* is not directly discussed in that theorem, the optimality conditions (2.3) - (2.5), and Proposition 2.8, can be used to identify it. Proposition 2.7 ensures such identification will be unique. It is beneficial for subsequent derivations to introduce a sequence, which is defined on the basis of optimality condition (2.3), and its kth element depends on $J_k = \{k, \dots, H\} \subseteq \widetilde{J}$. This allows for convenient examination of candidates J_k. Such sequence is defined below by (2.15).

Definition 2.9. Let $\widetilde{J} = \{1, \dots, H\}$ $(d^U_1 \geq d^U_2 \geq \dots \geq d^U_H)$, $J_k = \{k, \dots, H\} \subseteq \widetilde{J}$, $J^c_k = \widetilde{J} \setminus J_k$, $k \in \widetilde{J}$.

$$\xi_k = \frac{n - \sum\limits_{i \in J^c_k} u_i}{\sum\limits_{i \in J_k} d_i}, \quad k = 1, \dots, H. \tag{2.15}$$

In this version of the *Sequential Allocation* algorithm, candidate sets J_k are evaluated sequentially[4] for $k = 1, \dots, k_* \leq H$ against $\xi_k d_k \geq u_k$ condition until it is not met. Hence, $k_* = \min\{k \in \widetilde{J} \mid \xi_k d^U_k < 1\}$ and $\widetilde{J}_* = J_{k_*}$.

A formal statement of the algorithm is given below. The *order(x)* function used throughout, returns a permutation which rearranges the input series x into ascending (default) or descending order, e.g. $order([3, 1, 2, 4]) = [2, 3, 1, 4]$. Comments are preceded by the sign ▷.

Algorithm 1 Sequential Allocation

```
Require: $0 < n \leq \sum_{i \in J} u_i$                      ▷ problem must be feasible
1: procedure SeqAlloc($d[1, \dots, H]$, $u[1, \dots, H]$, $n$)
2:     $n^* \leftarrow u$
3:     if ($n < \sum_i u[i]$) then                                ▷ $J_* \neq \emptyset$
4:         $j \leftarrow order([\frac{d[1]}{u[1]}, \dots, \frac{d[H]}{u[H]}], decreasing)$     ▷ $d^U_{j[1]} \geq \dots \geq d^U_{j[H]}$
```

[3]Except formal statements of the algorithms.

[4]It is worth to observe that $J_{k+1} = J_k \setminus \{k\}$, i.e. strata are deleted as iterations progress.

```
5:          $ds \leftarrow \sum_i d[i]$
6:          for $k \leftarrow 1$ to $H$ do
7:              if $\frac{n}{ds} \cdot \frac{d[j[k]]}{u[j[k]]} \geq 1$ then          ▷ if $\xi_k d^U_{j[k]} \geq 1$, then stratum $j[k]$ goes to $J^c_*$
8:                  $n \leftarrow n - u[j[k]]$
9:                  $ds \leftarrow ds - d[j[k]]$
10:             else          ▷ executed at least once as $J_* \neq \emptyset$ ensures $\xi_H d^U_{j[H]} < 1$
11:                 $k_* \leftarrow k$
12:                 break
13:             end if
14:         end for
15:         $n^*[j[k_*], \ldots, j[H]] \leftarrow [\frac{n}{ds} \cdot d[j[k_*]], \ldots, \frac{n}{ds} \cdot d[j[H]]]$
16:     end if
17:     return $n^*$
18: end procedure
```

The diagram below illustrates the state of objects upon algorithm termination. A single dot symbol used in subscripts stands for each and every element from J^c_* (left column) or J_* (right column).

$J = \{1, \ldots, H\}$					
$j(1)$	$\ldots$	$j(k_* - 1)$	$j(k_*)$	$\ldots$	$j(H)$
$d^U_{j(1)} \geq$	$\ldots$	$\geq d^U_{j(k_*-1)}$	$> d^U_{j(k_*)} \geq$	$\ldots$	$\geq d^U_{j(H)}$
$\xi_1 d^U_{j(1)} \geq 1$	$\ldots$	$\xi_{k_*-1} d^U_{j(k_*-1)} \geq 1$	$\xi_{k_*} d^U_{\cdot} < 1$		
J^c_*			J_*		
$n^*_{\cdot} \equiv u_{\cdot}$			$n^*_{\cdot} = \xi_{k_*} d_{\cdot} < u_{\cdot}$		

Proposition 2.10. Let (ξ_k) be a sequence defined by (2.15) and $k_* = \min\{k \in \widetilde{J} \mid \xi_k d^U_k < 1\}$. Then, for $k_* \in \{1, \ldots, H-1\}$, the following inequalities hold

$$0 < \xi_1 \leq \xi_2 \leq \ldots \leq \xi_{k_*} > \xi_{k_*+1}. \tag{2.16}$$

Proof. Inequality $\xi_1 > 0$ is evident, given natural assumptions $n > 0,\ N_h > 0,\ S_h > 0$ are met for all $h \in J$. Below equivalences hold for $k = 1, \ldots, H-1$.

$$\begin{aligned}
\xi_{k+1} \geq \xi_k \Leftrightarrow \quad & \frac{n - \sum\limits_{i \in J_k^c} u_i - u_k}{\sum\limits_{i \in J_k} d_i - d_k} \geq \frac{n - \sum\limits_{i \in J_k^c} u_i}{\sum\limits_{i \in J_k} d_i} \qquad \Big| \, (\sum_{i \in J_k} d_i - d_k) \sum_{i \in J_k} d_i > 0 \\
\Leftrightarrow \quad & \cancel{(n - \sum_{i \in J_k^c} u_i) \sum_{i \in J_k} d_i} - u_k \sum_{i \in J_k} d_i \geq \cancel{(n - \sum_{i \in J_k^c} u_i) \sum_{i \in J_k} d_i} - (n - \sum_{i \in J_k^c} u_i) d_k \\
\Leftrightarrow \quad & (n - \sum_{i \in J_k^c} u_i) d_k \geq u_k \sum_{i \in J_k} d_i \\
\Leftrightarrow \quad & \xi_k d_k^U \geq 1.
\end{aligned}$$

Last equivalence, together with the following fact

$$\begin{aligned}
\xi_k d_k^U &\geq 1, \quad \forall k \in \{1, \ldots, k_* - 1\} = \widetilde{J}_*^c, \\
\xi_{k_*} d_{k_*}^U &< 1,
\end{aligned}$$

completes the proof. □

Theorem 2.11. Algorithm 1 determines the unique optimal solution to Problem 2.1.

Proof. From Proposition 2.3 we know that the optimal solution exists when the problem is feasible. Algorithm 1 runs sequentially through all feasible splits of J (i.e. as given by Proposition 2.8) in order to find set J_*. This set exists and it is unique, following Proposition 2.7. Hence, algorithm terminates in at most H iterations. Upon termination, the solution corresponding to J_*, clearly satisfies optimality conditions (2.3) - (2.4). To prove that condition (2.5) is met too, it should be noted that

$$\xi_k d_k^U \geq 1, \quad \forall k \in \{1, \ldots, k_* - 1\} = \widetilde{J}_*^c.$$

Given this fact, and since (ξ_k) is non-decreasing with $k = 1, \ldots, k_*$ (Proposition 2.10), it follows that

$$\xi_{k_*} d_k^U \geq 1, \quad \forall k \in \widetilde{J}_*^c.$$

This last inequality is equivalent to

$$\exists \lambda^* \in \mathbb{R}^+ : \ \frac{d_k}{u_k} \geq \frac{1}{\frac{n - \sum_{i \in J_*^c} u_i}{\sum_{i \in J_*} d_i}} \frac{d_h}{d_h} = \frac{d_h}{n_h^*} = \sqrt{\lambda^*}, \quad \forall (h, k) \in \widetilde{J}_* \times \widetilde{J}_*^c.$$

□

Example 2.12. Optimal allocation (column n^*.) for two different sample total sizes n for an example population with 4 strata. Column $\widetilde{J}_*^c$ indicates whether stratum is assigned to $\widetilde{J}_*^c$ set. All floating numbers in below table are rounded to 2 decimal places.

$n = 190$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.d^U_.$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.43	0.97	0	67.86
2	4000	90	44.44		0	54.29
3	3000	100	30.00		0	40.71
4	2000	80	25.00		0	27.14

$\widetilde{J}_* = J_1 = \{1, 2, 3, 4\}$, $\widetilde{J}^c_* = \emptyset$.

$n = 230$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.d^U_.$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.43	1.17	1	70.00
2	4000	90	44.44	0.79	0	71.11
3	3000	100	30.00		0	53.33
4	2000	80	25.00		0	35.56

$\widetilde{J}_* = J_2 = \{2, 3, 4\}$, $\widetilde{J}^c_* = \{1\}$.

2.3.2. Sequential Allocation (version 1)

The algorithm described above does not make use of certain property that sequence (ξ_k) possesses. This property could further improve the algorithm. It turns out that it might not always be necessary to calculate ξ_k for every iteration k of the *for* loop (lines 6 - 14) in order find set J_*. Corollary 2.13 below, is essential to justify this simplification.

Corollary 2.13. Let (ξ_k) be a sequence defined by (2.15). The following implication holds for all $k \in \{1, \dots, k_* - 1\} = \widetilde{J}^c_*$

$$(\exists l \in \{1, \dots, k-1\} \mid \xi_{k-l} d^U_k \geq 1) \implies \xi_k d^U_k \geq 1. \tag{2.17}$$

Proof. The proof is an immediate conclusion of Proposition 2.10. □

It is worth noticing, that converse implication does not necessary hold as it may happen that

$$1 < \xi_k d^U_k < 1 + x d^U_k, \qquad \text{where } x = \xi_k - \xi_{k-l} > 0,$$

resulting in

$$\begin{aligned} \xi_k d^U_k &< 1 + (\xi_k - \xi_{k-l}) d^U_k, \\ \xi_{k-l} d^U_k &< 1. \end{aligned}$$

At a single iteration k, Algorithm 1 continues to next iteration when condition $\xi_k d^U_k \geq 1$ is met (line 7). In such case, stratum k is classified to $\widetilde{J}^c_*$. Corollary 2.13 assures that as long as it is true that $\xi_{k-l} d^U_k \geq 1$ for some $l \in \{1, \dots, k-1\}$, inequality $\xi_k d^U_k \geq 1$ holds as well, and therefore stratum k can be classified to $\widetilde{J}^c_*$ through inequality $\xi_{k-l} d^U_k \geq 1$. It is apparent that $\xi_{k-l} d^U_k < 1$ (premise of the implication (2.17) is false) is inconclusive, that is $\xi_k d^U_k < 1$ or $\xi_k d^U_k \geq 1$. The formal statement of this enhancement is described below. It is the *repeat* loop solely responsible for J_* construction. It replaces *for* loop from the Algorithm 1 (lines 6 - 14). The *repeat* loop type is more convenient here than *for*, as none of the indices k, g is necessary altered in every single iteration of the loop.

```
 1: $k \leftarrow 1$
 2: $g \leftarrow 1$                                          ▷ Here, $g := k - l$ with regard to notation of Corollary 2.13
 3: **repeat**
 4:     **if** $\frac{n}{ds} \cdot \frac{d[j[k]]}{u[j[k]]} \geq 1$ **then**          ▷ if $\xi_g d^U_{j[k]} \geq 1$, stratum $j[k]$ goes to $J^c_*$
 5:         $k \leftarrow k + 1$
 6:     **else if** $g < k$ **then**                        ▷ $\xi_g d^U_{j[k]} < 1$ inconclusive for $g < k$ ...
 7:         $n \leftarrow n - \sum_{i=g}^{k-1} u[j[i]]$
 8:         $ds \leftarrow ds - \sum_{i=g}^{k-1} d[j[i]]$
 9:         $g \leftarrow k$                                ▷ ... hence, $\xi_k d^U_{j[k]} \geq 1$ checked at next iter.
10:     **else**                                            ▷ $\xi_k d^U_{j[k]} < 1$, then $J_* = \{j[k], \ldots, j[H]\}$
11:         $k_* \leftarrow k$
12:         **break**
13:     **end if**
14: **end repeat**
```

Example 2.14. Optimal allocation (column $n^*.$) for two different sample total sizes n for an example population with 4 strata. For $n = 320$, $\xi_1 d^U_2 = 0.0229 \cdot 44.4444 = 1.0159 > 1$, hence stratum 2 is classified to $\widetilde{J}^c_*$ together with stratum 1 through ξ_1. Column $\widetilde{J}^c_*$ indicates whether stratum is assigned to $\widetilde{J}^c_*$ set. All floating numbers in below table are rounded to 3 decimal places.

$n = 210$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.$	$\xi.d^U_.$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.429	0.015	1.071	1	70.000
2	4000	90	44.444	0.016	0.691	0	62.222
3	3000	100	30.000			0	46.667
4	2000	80	25.000			0	31.111

$n = 320$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.$	$\xi.d^U_.$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.429	0.023	1.633	1	70
2	4000	90	44.444		1.016	1	90
3	3000	100	30.000	0.032	0.960	0	96
4	2000	80	25.000			0	64

2.3.3. Sequential Allocation (version 2)

Finally, Proposition 2.10 induces yet another version of the *Sequential Allocation* algorithm. In this approach, monotonicity of sequence (ξ_k) is inspected in order to find J_*. Formal statement of the algorithm routine, solely responsible for J_* construction, is given below. It replaces *for* loop from the Algorithm 1 (lines 6 - 14).

```
1: $ksi \leftarrow \frac{n}{ds}$
2: $ksi1 \leftarrow 0$
3: $k \leftarrow 1$
4: while $k \leq H - 1$ do
5:     $n \leftarrow n - u[j[k]]$
6:     $ds \leftarrow ds - d[j[k]]$
7:     $ksi1 \leftarrow \frac{n}{ds}$
8:     if $ksi > ksi1$ then                          ▷ if change of monotonicity found
9:         break
10:    else
11:        $ksi \leftarrow ksi1$
12:        $k \leftarrow k + 1$
13:    end if
14: end while
15: if $(k == H - 1)$ & $(ksi \leq ksi1)$ then         ▷ if change of monotonicity not found
16:    $k_* \leftarrow H$                    ▷ then it must be that $J_* = \{j(H)\}$ as $J_* \neq \emptyset$
17: else
18:    $k_* \leftarrow k$
19: end if
```

Example 2.15. Optimal allocation (column n^*.) for four different sample total sizes n for an example population with 4 strata. Value in column ξ. $\downarrow$ and row $(k+1)$ indicates if $\xi_k > \xi_{k+1}$ inequality is met, $k = 1, \ldots, 3$. Column $\widetilde{J}_*^c$ indicate whether stratum is assigned to $\widetilde{J}_*^c$ set. All floating numbers in below table are rounded to 4 decimal places.

$n = 190$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.$	$\xi. \downarrow$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.4286	0.0136		0	67.86
2	4000	90	44.4444	0.0133	1	0	54.29
3	3000	100	30.0000			0	40.71
4	2000	80	25.0000			0	27.14

$n = 270$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.$	$\xi. \downarrow$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.4286	0.0193		1	70.00
2	4000	90	44.4444	0.0222	0	0	88.89
3	3000	100	30.0000	0.0220	1	0	66.67
4	2000	80	25.0000			0	44.44

$n = 300$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.$	$\xi. \downarrow$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.4286	0.0214		1	70
2	4000	90	44.4444	0.0256	0	1	90
3	3000	100	30.0000	0.0280	0	0	84
4	2000	80	25.0000	0.0200	1	0	56

$n = 330$

$\widetilde{J}$	$d.$	$u.$	$d^U_.$	$\xi.$	$\xi. \downarrow$	$\widetilde{J}^c_*$	$n^*_.$
1	5000	70	71.4286	0.0236		1	70
2	4000	90	44.4444	0.0289	0	1	90
3	3000	100	30.0000	0.0340	0	1	100
4	2000	80	25.0000	0.0350	0	0	70

Sequential Allocation algorithm is based on Proposition 2.8. As it was mentioned, this proposition reduces the number of candidates to J_* from 2^H to H. Yet, there is the price to pay for this facilitation, the set of strata indices J must be properly ordered up-front.

2.3.4. Recursive Neyman Allocation

Recursive Neyman allocation approach to Problem 2.1, here termed *Recursive Neyman* algorithm, is historically not connected to Theorem (2.4). We start with describing the principles of this approach for the case when upper bounds $u_h \leq N_h$ for all $h \in J$. Consider Problem 2.1 without inequality constraints (2.2), referred here as Relaxed Problem 2.1. The classical Neyman optimal allocation [9, Chapter 3.7.4.i, p. 106] provides an optimal solution

$$n^*_h = \frac{n}{\sum\limits_{i \in J} d_i} d_h, \quad \forall h \in J, \tag{2.18}$$

to Relaxed Problem 2.1.

It turns out that the Neyman allocation (2.18), when applied in a recursive way to step-wise reduced set of strata [9, Remark 12.7.1, p. 466][5], it can be used to determine the optional solution to problems with inequality constrains on lower or (exclusive or) upper bound of $(n_1, \ldots, n_H)$, such as Problem 2.1. Namely, if the solution to such additionally constrained problem, given by allocation (2.18), violates some of the inequality constraints, then constraints limiting values should be taken as a solution for strata exceeding these restrictions. For allocation in remaining

[5]Costs are not considered here in this work, by assuming fixed cost $c_h \equiv 1$, $\forall h \in J$.

strata, allocation (2.18) is applied again on reduced set of strata. The whole operation is iteratively repeated until the allocation meets all of the inequality constraints. Below is the formal statement of this concept for Problem 2.1. Function *length(x)* used in below pseudo-code, returns length of the vector x, e.g. $length([3, 1, 2, 5]) = 4$.

Algorithm 2 Recursive Neyman allocation - upper finite bounds only

```
Require: $0 < n \le \sum_{i \in J} u_i$                                  ▷ problem must be feasible
 1: procedure RNEYMAN($d[1, \ldots, H],\ u[1, \ldots, H],\ n$)
 2:     $n^* \leftarrow u$
 3:     if $(n < \sum_i u[i])$ then                                     ▷ $J_* \neq \emptyset$
 4:         $ds \leftarrow \sum_i d[i]$
 5:         $j \leftarrow [1, \ldots, H]$
 6:         repeat
 7:             $jl \leftarrow length(j)$
 8:             $nopt \leftarrow [\frac{n}{ds} \cdot d[j[1]], \ldots, \frac{n}{ds} \cdot d[j[jl]]]$
 9:             for $i \leftarrow 1$ to $jl$ do
10:                 if $nopt[i] \ge u[j[i]]$ then          ▷ if true, then strata $j[i]$ goes to $J_*^c$
11:                     $n \leftarrow n - u[j[i]]$
12:                     $ds \leftarrow ds - d[j[i]]$
13:                     $j \leftarrow j[1, \ldots, i-1, i+1, \ldots, jl]$
14:                 end if
15:             end for
16:             if $jl == length(j)$ then    ▷ if no bounds exceeded in a given iteration of repeat
17:                 break                                              ▷ stop, $j$ is equal $J_*$
18:             end if
19:         end repeat
20:         $n^*[j[1], \ldots, j[jl]] \leftarrow nopt$
21:     end if
22:     return $n^*$
23: end procedure
```

It would be convenient for further derivations to introduce the following sequence, in a way similar to sequence (ξ_k) defined by (2.15).

Definition 2.16. Let $J_r^R \subseteq J$ denote set of strata indices that left for allocation at iteration

$r = 1, \dots, r_*$ of the *Recursive Neyman* algorithm.

$$\xi_r^R = \frac{n - \sum\limits_{i \in J \setminus J_r^R} u_i}{\sum\limits_{i \in J_r^R} d_i}, \quad r = 1, \dots, r_*. \tag{2.19}$$

Proposition 2.17. Let $J_*^c = \{h \in J \mid n_h^* = u_h\}$, where allocation n_h^*, $h \in J$ is computed by the *Recursive Neyman* algorithm. At every iteration of the algorithm (except the last iteration), strata indices h with highest d_h^U among all strata remaining for allocation at that iteration, are moved to J_*^c.

Proof. Let $J_r^R \subseteq J$ denote set of strata indices that left for allocation at iteration $r = 1, \dots, r_*$ of the algorithm, and ξ_r^R be defined by (2.19). At every iteration $r = 1, \dots, r_*$, all strata indices $h \in J_r^R$ are evaluated against condition

$$\xi_r^R d_h^U \geq 1.$$

Since $\xi_r^R > 0$, $d_h^U > 0$, for $r = 1, \dots, r_*$, $\forall h \in J$; it is evident that if there are any strata indices $h \in J_r^R$ that meet this above condition at iteration r, these are the strata with highest d_h^U. $\square$

Theorem 2.18. *Recursive Neyman* algorithm (Algorithm 2), determines the optimal solution to Problem 2.1.

Proof. Let $J_{*R} = \{h \in J \mid n_h^* < u_h\}$, and $J_{*R}^c = \{h \in J \mid n_h^* = u_h\}$, where allocation n_h^*, $h \in J$ is computed by the *Recursive Neyman* algorithm, and ξ_r^R is defined by (2.19). Upon termination, the algorithm leads to the following allocation

$$\begin{aligned} \forall h \in J_{*R}: \quad & n_h = \frac{n - \sum\limits_{i \in J_{*R}^c} u_i}{\sum\limits_{i \in J_{*R}} d_i} d_h \in (0, u_h), \\ \forall h \in J_{*R}^c: \quad & n_h = u_h, \end{aligned} \tag{2.20}$$

which clearly satisfies optimality conditions (2.3) - (2.4) stated in Theorem 2.4 for an optimal solution. In order to prove that optimality condition (2.5) is met too, the following two facts should be noticed:

(1) $\exists \lambda^* \in \mathbb{R}^+$, i.e. $\lambda^* = \frac{d_h}{n_h^*} = (\xi_{r_*}^R)^{-1}$, $\forall h \in J_{*R}$.

(2) It is a direct consequence of Proposition 2.17 that sequence $(\xi_r^R)_{r=1,\dots,r_*}$ is a subsequence of the sequence $(\xi_k)_{k=1,\dots,k_*}$ defined by (2.15). Thus, sequence $(\xi_r^R)_{r=1,\dots,r_*}$ is non-decreasing, following Proposition 2.10. Then, since $\forall r \in \{1, \dots, r_* - 1\}$ $\forall i \in J_r^R$, we have

$$\xi_r^R d_i^U \geq 1, \tag{2.21}$$

the following inequality holds

$$\xi^R_{r_*} d^U_i \geq 1. \tag{2.22}$$

Facts (1) and (2) read, $\lambda^* = \frac{d_h}{n^*_h} = \frac{1}{\xi^R_{r_*}} \leq d^U_i$, $\forall h \in J_{*R}$, $\forall i \in J^c_{*R}$.
To complete the proof, it should be noted that by the uniqueness of the optimal solution (Proposition 2.3) and uniqueness of corresponding J_* (Proposition 2.7), it follows that $J_{*R} = J_*$, and $J^c_{*R} = J^c_*$. □

Closer look at this algorithm reveals its analogy to *Sequential Allocation* (version 1) algorithm introduced in the previous part of this chapter. Unlike *Sequential Allocation* algorithm, the *Recursive Neyman* does not order strata indices up-front. Yet, in every iteration (except the last one), it classifies to J^c_* strata indices with highest d^U (see Proposition 2.17). That in fact imposes automatically the order on J.
Recursive Neyman algorithm, calculates allocations for all strata remaining for allocation at a given iteration, and then searches for strata exceeding inequality constraints among all strata allocated in that iteration. In contrast, *Sequential Allocation* algorithm, calculates allocations in sequence for strata in ordered set of strata indices left for allocation at a given iteration, until it finds the first stratum not exceeding inequality constraint.
In other words, assuming strata set of indices J were ordered up-front with regard to non-increasing d^U_h, it would be sufficient for *Recursive Neyman* algorithm to calculate allocations at given iteration, only for those first strata left for allocation, for which inequality constraints are exceeded. This is what in fact the *Sequential Allocation* algorithm in version 1 does.

3. Finite Lower and Finite Upper Bounds

3.1. Problem Formulation

Problem 2.1 will be generalized by adding lower bound constraints l_h of n_h for all $h \in J$.

Problem 3.1.

$$\min_{(n_1,\dots n_H)\in\mathbf{\Omega}} \quad D^2_{\hat{t}_{st}}(n_1,\dots,n_H), \quad \mathbf{\Omega} = \mathbb{R}^H_+,$$

subject to

$$\sum_{h=1}^{H} n_h - n = 0, \tag{3.1}$$

$$l_h - n_h \le 0, \qquad h = 1,\dots,H, \tag{3.2}$$

$$n_h - u_h \le 0, \qquad h = 1,\dots,H. \tag{3.3}$$

where function $D^2_{\hat{t}_{st}}$ is given by (1.1), and n, l_h, u_h, $h \in J$ are known constants. Following the sampling design, we shall make throughout a natural assumptions about known constants, i.e. $\sum_{i\in J} l_i \le n \le \sum_{i\in J} u_i$, and $0 \le l_h \le u_h \le N_h$ for all $h \in J$. Without the first assumption Problem 3.1 might be infeasible. We will additionally require $l_h \neq 0$. Case $l_h = 0$ simplifies Problem 3.1 to Problem 2.1. Clearly, the optimal solution is in the interior of $\mathbf{\Omega}$ due to presence of inequality constraints (3.2), (3.3), and assumptions about known constants. Hence the set constraint $(n_1,\dots n_H) \in \mathbf{\Omega}$ is not placed on the list of functional constraints.

Remark 3.2. Optimization Problem 3.1 is a convex optimization problem[1] since the objective function $D^2_{\hat{t}_{st}}$ and inequality constraint functions $l_h - n_h$, $n_h - u_h$ of n_h for all $h \in J$ are convex, equality constraint function $\sum_{h=1}^{H} n_h - n$ of $(n_1,\dots,n_H)$ is affine.

Proposition 3.3. The optimal solution to Problem 3.1 exists and it is globally unique.

Proof. The proof is analogous to the proof of Proposition 2.3 □

[1] Appendix A.1

3.2. Optimality Conditions

Similarly to Chapter 2, optimality conditions can be formulated for an optimal solution to Problem 3.1. Gabler, Ganninger and Munnich formulate necessary (not sufficient) conditions in [5, Theorem 1, p. 152]. Theorem 3.4 below, provides necessary and sufficient conditions for an optimal solution without assumption of particular ordering[2] of set of strata indices J.

Theorem 3.4. $(n_1^*, \dots, n_H^*)$ is an optimal solution to Problem 3.1, if and only if there exist disjoint sets J_*, J_*^L, J_*^U, $J_* \cup J_*^L \cup J_*^U = J$, and number $\lambda^* \in \mathbb{R}^+$ such that:
(i) for $J_* \neq \emptyset$

$$\forall h \in J_*: \quad l_h < n_h^* = \frac{n - \sum\limits_{i \in J_*^L} l_i - \sum\limits_{i \in J_*^U} u_i}{\sum\limits_{i \in J_*} d_i} d_h < u_h, \tag{3.4}$$

$$\forall h \in J_*^L: \quad n_h^* = l_h, \tag{3.5}$$

$$\forall h \in J_*^U: \quad n_h^* = u_h, \tag{3.6}$$

$$\forall h \in J_*,\ \forall i \in J_*^L,\ \forall j \in J_*^U: \quad \frac{d_i}{l_i} \le \frac{d_h}{n_h^*} = \sqrt{\lambda^*} \le \frac{d_j}{u_j}; \tag{3.7}$$

(ii) for $J_* = \emptyset$

$$\forall h \in J_*^L: \quad n_h^* = l_h, \tag{3.8}$$

$$\forall h \in J_*^U: \quad n_h^* = u_h, \tag{3.9}$$

$$\forall i \in J_*^L,\ \forall j \in J_*^U: \quad \frac{d_i}{l_i} \le \frac{d_j}{u_j}, \tag{3.10}$$

$$\sum_{i \in J_*^L} l_i + \sum_{i \in J_*^U} u_i - n = 0. \tag{3.11}$$

Proof. Problem 3.1 is a convex optimization problem (Remark 3.2) with objective and constraint functions of class C^1 on $\mathbf{\Omega}$. Therefore, the Karush-Kuhn-Tucker conditions are necessary and sufficient, and they take the following form for Problem 3.1

[2]Authors of [5] assume that the set $J = \{1, 2, \dots, H\}$ is ordered such that $\frac{d_1}{u_1} \le \dots \le \frac{d_H}{u_H}$. This additional assumption is obsolete.

$\exists\lambda^* \in \mathbb{R}$, $\exists(\mu_1^{L*},\dots,\mu_H^{L*}) \in \mathbb{R}^H$, $\exists(\mu_1^{U*},\dots,\mu_H^{U*}) \in \mathbb{R}^H$ such that for all $h \in J$

$$\begin{aligned}
\varphi_h(n_h^*) + \lambda^* - \mu_h^{L*} + \mu_h^{U*} &= 0, && (\textit{stationarity})\\
\sum_{i\in J} n_i^* - n &= 0, && (\textit{primal feasibility})\\
n_h^* - u_h &\le 0, && (\textit{primal feasibility})\\
l_h - n_h^* &\le 0, && (\textit{primal feasibility})\\
\mu_h^{L*}(l_h - n_h^*) &= 0, && (\textit{comp. slackness})\\
\mu_h^{U*}(n_h^* - u_h) &= 0, && (\textit{comp. slackness})\\
\mu_h^{L*} &\ge 0, && (\textit{dual feasibility})\\
\mu_h^{U*} &\ge 0, && (\textit{dual feasibility})
\end{aligned} \tag{3.12}$$

where partial derivative function φ_h and its inverse function φ_h^{-1} are given by (2.6).

Each of the *primal feasibility* inequality conditions can be divided into two distinct cases, leading to the following optimality conditions equivalent to (3.12):
There exist disjoint sets J_*, J_*^L, J_*^U, $J_* \cup J_*^L \cup J_*^U = J$, and $\exists\lambda^* \in \mathbb{R}^+$ such that

$$l_h < n_h^* < u_h \quad \text{and} \quad \varphi_h(n_h^*) = -\lambda^*, \qquad \forall h \in J_*, \tag{3.13}$$

$$n_h^* = l_h \quad \text{and} \quad \varphi_h(l_h) \ge -\lambda^*, \qquad \forall h \in J_*^L, \tag{3.14}$$

$$n_h^* = u_h \quad \text{and} \quad \varphi_h(u_h) \le -\lambda^*, \qquad \forall h \in J_*^U, \tag{3.15}$$

$$\sum_{i\in J_*} n_i^* + \sum_{i\in J_*^L} l_i + \sum_{i\in J_*^U} u_i - n = 0. \tag{3.16}$$

Conditions (3.13) - (3.16) can be further shortened when $J_* \neq \emptyset$. From now on, we narrow the domain of possible solutions to $n_h^* > 0$, $\forall h \in J$. This restriction does not invalidate the proof, since functions φ_h are even functions on their natural domains $\mathbb{R}\setminus\{0\}$. It follows from (2.8) that for $n_h^* > 0$, we get $n_h^* = \varphi_h^{-1}(-\lambda^*)$ for all $h \in J_*$, and therefore

$$\sum_{i\in J_*} n_i^* = \sum_{i\in J_*} \varphi_i^{-1}(-\lambda^*) = \sum_{i\in J_*} \frac{d_i}{\sqrt{\lambda^*}} = \frac{1}{\sqrt{-\varphi_h(n_h^*)}} \sum_{i\in J_*} d_i = \frac{1}{\sqrt{\frac{d_h^2}{(n_h^*)^2}}} \sum_{i\in J_*} d_i = \frac{n_h^*}{d_h} \sum_{i\in J_*} d_i,$$

$$n_h^* = \frac{\sum_{i\in J_*} n_i^*}{\sum_{i\in J_*} d_i} d_h, \quad \forall h \in J_*.$$

Given (3.16)

$$n_h^* = \frac{n - \sum_{i\in J_*^L} l_i - \sum_{i\in J_*^U} u_i}{\sum_{i\in J_*} d_i} d_h, \quad \forall h \in J_*. \tag{3.17}$$

Furthermore, (3.13) - (3.15) read
$\forall(h,i,j) \in J_* \times J_*^L \times J_*^U$, $\exists\lambda^* \in \mathbb{R}^+$ such that

$$\varphi_j(u_j) \le \varphi_h(n_h^*) = -\lambda^* \le \varphi_i(l_i),$$

i.e.

$$\frac{d_i}{l_i} \leq \frac{d_h}{n_h^*} = \sqrt{\lambda^*} \leq \frac{d_j}{u_j}. \tag{3.18}$$

Due to (3.17) and (3.18), optimality conditions (3.13) - (3.16) can be formulated as in Theorem 3.4.

To complete the proof, it should be noted that optimality conditions (3.4) - (3.6) guarantee that the optimal solution $(n_1^*, \ldots, n_H^*)$ is in the interior of $\mathbf{\Omega}$. □

Remark 3.5. (to the proof of Theorem 3.4)
Optimality condition (3.16) was enclosed (and replaced) into (3.17) in the course of the proof assuming $J_* \neq \emptyset$. If $J_* = \emptyset$, condition (3.17) and therefore (3.16) decay, resulting in possibly non-optimal solution. Hence, when $J_* = \emptyset$, condition (3.16) must be evaluated explicitly in order to preserve sufficiency.

Remark 3.6. In Theorem 3.4, the split of J into distinct sets J_*, J_*^L, J_*^U is unique if and only if $l_h \neq u_h$, for all $h \in J$.

Proof. The proof is an immediate conclusion from Proposition 3.3 and distinctness of the split of *primal feasibility* conditions (3.12) made in the course of the proof of Theorem 3.4. The division of set J into J_*, J_*^L, J_*^U is equivalent of that split. □

Remark 3.7. It is an immediate conclusion from Theorem 3.4 that the following expressions hold for an optimal solution $(n_1^*, \ldots, n_H^*)$

$$(n_1^*, \ldots, n_H^*) = (l_1, \ldots, l_H) \;\Leftrightarrow\; n = \sum_{i=1}^{H} l_i,$$
$$(n_1^*, \ldots, n_H^*) = (u_1, \ldots, u_H) \Leftrightarrow\; n = \sum_{i=1}^{H} u_i,$$

which is rather an intuitive result without the formalism of the KKT conditions.

Theorem 3.4 introduces three distinct subsets J_*, J_*^L, J_*^U of J. Below Proposition 3.8 reveals the property related to these sets which allows for simplification of algorithms to Problem 3.1.

Proposition 3.8. Let J_*, J_*^L, J_*^U be as in Theorem 3.4. The following properties hold

$$\frac{d_i}{l_i} < \frac{d_h}{l_h}, \quad \forall (i,h) \in J_*^L \times J_*,$$

$$\frac{d_h}{u_h} < \frac{d_j}{u_j}, \quad \forall (h,j) \in J_* \times J_*^U.$$

Proof. Let the function φ_h be given by (2.6). Since φ_h is strictly increasing, (3.13) yields

$$\varphi_h(l_h) < -\lambda^*, \tag{3.19}$$

$$\varphi_h(u_h) > -\lambda^*, \quad \forall h \in J_*. \tag{3.20}$$

Due to (3.14)

$$\varphi_i(l_i) \geq -\lambda^*, \quad \forall i \in J_*^L, \tag{3.21}$$

and (3.15)

$$\varphi_j(u_j) \leq -\lambda^*, \quad \forall j \in J_*^U. \tag{3.22}$$

Inequalities (3.19) and (3.21) imply $\varphi_h(l_h) < \varphi_i(l_i)$, i.e.

$$-\frac{d_h^2}{l_h^2} < -\frac{d_i^2}{l_i^2} \equiv \frac{d_i}{l_i} < \frac{d_h}{l_h}, \quad \forall (i,h) \in J_*^L \times J_*.$$

Inequalities (3.20) and (3.22) imply $\varphi_h(u_h) > \varphi_j(u_j)$, i.e.

$$-\frac{d_h^2}{u_h^2} > -\frac{d_j^2}{u_j^2} \equiv \frac{d_h}{u_h} < \frac{d_j}{u_j}, \quad \forall (h,j) \in J_* \times J_*^U.$$

□

3.3. Algorithms

3.3.1. The noptcond by Gabler, Ganniger and Munnich (2012)

Gabler, Ganninger and Munnich propose the algorithm called *noptcond* [5, Section 3 Programme code, p. 158][3] providing feasible, but - as it will be revealed further below - not necessary optimal solution to Problem 3.1. This algorithm tests each feasible (i.e. as indicated by Proposition 3.8) split of J into J^0, J^L, J^U, against optimality condition (3.4) in order to determine a feasible solution. If the condition is violated for a given split, algorithm proceeds to next candidate. The table below illustrates feasible splits (one candidate per one row) in the order the algorithm evaluates them. The procedure stops at the candidate for which optimality condition (3.4) is met. Using the notation adopted in this thesis, the *noptcond* orders strata so that $\frac{d_1}{l_1} \leq \cdots \leq \frac{d_H}{l_H}$, and i describes permutation of strata indices such that $\frac{d_{i(1)}}{u_{i(1)}} \leq \cdots \leq \frac{d_{i(H)}}{u_{i(H)}}$. The symbol R denotes total number of strata to allocate to $J^L \cup J^U$ at given iteration, and it is consistent with variable R in the R program code [5, Section 3 Programme code, p. 158]. Only distinct candidates

[3]The R code of this algorithm can be found in Appendix C.1 of this thesis.

(i.e. $J^L \cap J^U = \emptyset$) are considered, others, non-distinct are simply skipped. The algorithm terminates in at most $1 + (H-1)H$ iterations. It is worth noticing that $R = H$ (equivalent to $J^0 = \emptyset$) even if allowed by the algorithm, is never reached. This is caused by accepting non-strict inequalities when assessing optimality condition (3.4). Such approach does not invalidate the results obtained, yet it causes sets J_*, J^L_*, J^U_* found by the algorithm, even if they correspond to unique optimal solution, they may not be consistent with Theorem 3.4. It also leads to one less iteration.

R	J^L	J^U
0	$\emptyset$	$\emptyset$
1	$\{1\}$	$\emptyset$
1	$\emptyset$	$\{i(H)\}$
2	$\{1,2\}$	$\emptyset$
2	$\{1\}$	$\{i(H)\}$
2	$\emptyset$	$\{i(H-1), i(H)\}$
	$\dots$	
H-1	$\{1,\dots,H-1\}$	$\emptyset$
	$\dots$	
H-1	$\emptyset$	$\{i(2),\dots,i(H)\}$

It is important to remark that that the feasible solution provided by the algorithm may not be optimal. The fact that the solution is feasible is evident from the construction of the algorithm. The optimality is not guaranteed as the procedure does not ensure that the optimality condition (3.7) is met. The impact of possible violation of the condition (3.7), can readily be spotted with the following example. Let the algorithm stops with $R = 1$ and $J^L = \{1\}$, $J^U = \emptyset$, meaning that the corresponding solution is feasible. It is however possible that the optimal solution is for $J^L_* = \emptyset$, $J^U_* = \{i(H)\}$. That however won't be discovered by *noptcond*, since the algorithm stops before moving to this candidate. Such scenario is possible, for instance for $J = \{1,2\}$,
$\exists n \in \mathbb{R}^+$, $\exists \kappa = \frac{d_2}{d_1}$, such that

$$\begin{aligned}
l_1 + l_2 < n < u_1 + u_2, &\quad \text{| problem must be feasible} \\
\frac{n}{d_1 + d_2} > \frac{u_1}{d_1} \equiv n > u_1(\kappa + 1), &\quad \text{| e.g. uper bound exceeded} \\
\frac{d_1}{l_1} < \frac{d_2}{l_2} \equiv \kappa > \frac{l_2}{l_1}, &\quad \text{| stratum 1 goes to } J^L \\
l_2 < n - l_1 < u_2 \equiv l_1 + l_2 < n < l_1 + u_2, &\quad \text{| no bounds exceeded} \\
\frac{d_1}{l_1} > \frac{d_2}{n - l_1} \equiv n > l_1(\kappa + 1). &\quad \text{| but opt. condition (3.7) violated}
\end{aligned}$$

First restriction (feasibility due to n), together with $l_1 < u_1$, $l_2 < u_2$, simplify above inequalities to

$$\max(l_1 + l_2,\ u_1(\kappa + 1)) < n < l_1 + u_2, \tag{3.23}$$

$$\kappa > \frac{l_2}{l_1}. \tag{3.24}$$

It is evident these three inequalities can be met for some n, and κ, when choosing u_2 big enough. Numerical example below illustrates the scenario considered.

Example 3.9. Non-optimal allocation by *noptcond* alogorithm and optimal allocation for an example of population with 2 strata with $\sum_{h \in J} l_h = 70$, $\sum_{h \in J} u_h = 250$, and $\kappa = 1.5$. Hence, $\max(l_1 + l_2,\ u_1(\kappa + 1)) = 125 < n = 160 < 230 = l_1 + u_2$, as well as $\kappa = 1.5 > 1.33 = \frac{l_2}{l_1}$.

J	$N_.$	$S_.$	$l_.$	$u_.$	$n_.^{noptcond}$	$n_.^*$
1	100	20	30	50	30	50
2	300	10	40	200	130	110

Optimal solution yields $J_*^L = \emptyset$, $J_* = \{2\}$, $J_*^U = \{1\}$, while *noptcond* terminates with $J_*^L = \{1\}$, $J_* = \{2\}$, $J_*^U = \emptyset$. Corresponding variances (rounded to the nearest integer): $D^2_{\hat{t}_{st}}(n_1^{noptcond}, n_2^{noptcond}) = 132564 > 91818 = D^2_{\hat{t}_{st}}(n_1^*, n_2^*)$.

3.3.2. Sufficient noptcond

A simple adjustment can be made to *noptcond* algorithm so that it provides the optimal solution to Problem 3.1. Specifically, a feasible candidate found, should additionally be checked against optimality condition (3.7). This causes that all necessary and sufficient optimality conditions (3.4) - (3.7) are evaluated for every feasible candidate J^0, J^L, J^U. Re-implemented algorithm in R, including proposed enhancement can be found in Appendix C.2.

4. Concluding Remarks

Although results presented in this work are complete, there are possible further development paths. Among them, the most important are as follows.

1. Algorithms presented in Chapter 2.3 can be further analyzed against computational complexity and memory usage. For instance, most of available sorting algorithms (sorting operation is invoked in all of the algorithms presented in Chapter 2.3, except *Recursive Neyman* algorithm), immediately yields $\mathcal{O}(n^2)$ upper bound[1] on the worst-case running time. There are however sorting algorithms with $\mathcal{O}(nlog(n))$ worst-case running time. Furthermore, if some information about the probabilistic distribution of the algorithm input is available, the average-case running time could be computed. It may turn out that for some different population/sampling characteristics some algorithms perform better (in terms of average-case running time) and some other worst.

2. Algorithm *noptcond* discussed in Chapter 3.3, at each iteration (except last one) moves only one stratum to $J^L \cup J^U$. It would be desirable to move more than just one stratum to $J^L \cup J^U$ at single iteration whenever possible, in the way similar to *Recursive Neyman* allocation in Problem 2.1. It evident that recursive Neyman approach does not yield optimal solution for Problem 3.1. This trivial example below clearly illustrates it.

   Example 4.1. Results of classical Neyman allocation for sample total size $n = 55$ for an example of population with 2 strata. Both restrictions are exceeded.

J	$d.$	$l.$	$u.$	$n.^{neyman}$	$n.^{neyman} < l.$	$n.^{neyman} > u.$
1	90	18	25	13.56	TRUE	FALSE
2	275	30	40	41.44	FALSE	TRUE

[1]$\mathcal{O}(g(n)) = \{f(n) \mid \exists c > 0,\ \exists n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n)\ \forall n \geq n_0\}$. $\mathcal{O}(\cdot)$-notation is used to give an upper bound on a function, up to a constant factor.

3. It is evidently desired that strata sample sizes are integers. Problems 2.1 and 3.1 were formulated in the way they do not require integer solution. Consequently, the optimality conditions obtained, and the algorithms proposed do not assure the solution will be integer. Furthermore, it may happen that after rounding of non-integer optimal solution: (1) minimum of the objective function is not guaranteed anymore, (2) sample size (i.e. overall sample size, or sample sizes within strata) violates imposed constraints [12]. These drawbacks received an attention in e.g. [12] and [4], where authors propose algorithms yielding integer solution to optimal allocation problem under given sample size constraints. In the context of this work, it seems interesting to investigate whether optimal allocation problem formulated in the language of mathematical optimization could be adjusted so that it yields an optimal integer solution.

   Apparently, algorithms giving integer optimal solutions are more time consuming, whilst the gain in optimality is typically negligible. Therefore, at least from the applications point of view, it is rather desirable to have an efficient algorithm for the optimal, possibly non-integer solution, and then round such a solution properly to an integer (which will be negligibly sub-optimal).

Appendices

A. Mathematical Optimization Background

Mathematical optimization is concerned with problems called *optimization problems.* Below is the most common formulation of the optimization problem

$$\begin{aligned} \text{optimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h_i(\mathbf{x}) = 0, \quad i = 1, \dots, l, \\ & g_j(\mathbf{x}) \leq 0, \quad j = 1, \dots, p, \\ & \mathbf{x} \in \boldsymbol{\Omega}, \end{aligned}$$

where $l, p \geq 0$ and 'optimize' means minimize or (exclusive or) maximize.
If $l = 0$ and $p = 0$ the problem is an *unconstrained* optimization problem. Point $\mathbf{x}$ is the *optimization variable* of the problem, the function $f(\cdot)$ is the *objective function*, and $h_i(\cdot), g_j(\cdot)$ are *constraint functions.* The constraints $h_i(\mathbf{x}) = 0$, $g_j(\mathbf{x}) \leq 0$ are referred to as *functional constraints*; more specifically, $h_i(\mathbf{x}) = 0$ is termed as *equality constraint* and $g_j(\mathbf{x}) \leq 0$ as *inequality constraint.* The constraint $\mathbf{x} \in \boldsymbol{\Omega}$ is a *set constraint.* Set constraints are quite often de-emphasized, assuming in most cases that either is the whole Euclidean space E^n or that the solution to the problem is in the interior of $\boldsymbol{\Omega}$. A point $\mathbf{x} \in \boldsymbol{\Omega}$ that satisfies all the functional constraints is said to be *feasible.* A set of all feasible points is called *feasible region* or just *feasible set.* A feasible point $\mathbf{x}^*$ is called optimal, or a solution of the problem, if it has the smallest (in case of minimization) or largest (in case of maximization) objective value among all feasible points.
Many books cover this topic, including [6], and [2].

Definition A.1. (Convex Problem)
A convex optimization problem is an optimization problem in which the objective function is a convex function and the feasible region is a convex set.

Proposition A.2. Sufficient conditions for an optimization problem to be a convex optimization problems are:

- The objective function is convex,

- Inequality constraint functions are convex.
- Equality constraint functions are affine,

Proof. Presence of the first condition is evident. Since the objective function is convex, its domain set is convex (1). Sublevels $\{x \mid g(\mathbf{x}) \leq 0\}$ ($g(\mathbf{x})$ - inequality constraint function) of convex sets are convex (2), and affine sets are convex (3). The intersection of convex sets (1), (2), (3) is convex. More details on convex optimization problems can be found in [3, Chapter 4.2 Convex optimization, p. 150] □

Theorem A.3. (The Extreme Value Theorem)
If $f : \mathbb{X} \rightarrow \mathbb{R}$ is real valued function from a compact space to the real numbers, then f attains a largest value, that is there is an $x \in \mathbb{X}$ such that $f(x) \geq f(y)\ \forall y \in \mathbb{X}$.

Theorem A.4. (Heine-Borel Theorem)
A subset of $\mathbb{R}^n$ is compact if and only if it is closed and bounded.

Proofs to A.3 and A.4 can be found in [7].

B. Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker (KKT) approach to constrained nonlinear programming generalizes the method of Lagrange multipliers (which allows for equality constraints only).

Theorem B.1. (Karush-Kuhn-Tucker Necessary Conditions) Suppose $f : \mathbb{R}^H \to \mathbb{R}$, $f \in C^1$; $h_i : \mathbb{R}^H \to \mathbb{R}$, $h_i \in C^1(i = 1, \ldots, l)$; $g_j : \mathbb{R}^H \to \mathbb{R}$, $g_j \in C^1(j = 1, \ldots, p)$. Let $\mathbf{x} = \mathbf{x}^*$ be a local optimum to the problem

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, l, \\ &&& g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, p, \end{aligned} \tag{B.1}$$

where $l, p \geq 1$, $\mathbf{x} = (x_1, \cdots, x_H)$, and constraint functions $h_i(\cdot)$, $g_j(\cdot)$ satisfy some regularity conditions. Then

$$\begin{aligned} &\exists \lambda^* \in \mathbb{R}^l, \exists \mu^* \in \mathbb{R}^p \text{ such that} \\ &\textbf{Stationarity} \\ &\frac{\partial}{\partial x_h} f(\mathbf{x}^*) + \sum_{i=1}^{l} \lambda_i^* \frac{\partial}{\partial x_h} h_i(\mathbf{x}^*) + \sum_{j=1}^{p} \mu_j^* \frac{\partial}{\partial x_h} g_j(\mathbf{x}^*) = 0, \quad h = 1, \ldots, H, \\ &\textbf{Primal feasibility} \\ &h_i(\mathbf{x}^*) = 0, \quad i = 1, \ldots, l, \\ &g_j(\mathbf{x}^*) \leq 0, \quad j = 1, \ldots, p, \\ &\textbf{Dual feasibility} \\ &\mu_j^* \geq 0, \quad j = 1, \ldots, p, \\ &\textbf{Complementary slackness} \\ &\mu_j^* g_j(\mathbf{x}^*) = 0, \quad j = 1, \ldots, p. \end{aligned} \tag{B.2}$$

In case $h_i(\cdot)$ and $g_j(\cdot)$ are affine functions, then no further regularity conditions are needed. Furthermore, for convex optimization problem (see Definition A.1) with $f(\cdot) \in C^1$, $h_i(\cdot) \in C^1$ and $g_j(\cdot) \in C^1$, necessary conditions (B.2) are also sufficient for optimality [3, Chapter 5.5.3 KKT optimality conditions, p. 243]. In such case, it can be stated that, $\mathbf{x}^*$ is a local optimum to the problem (B.1) if and only if $\mathbf{x}^*$ satisfies conditions (B.2).

Objective function $f(\cdot)$ and constraint functions $h_i(\cdot)$, $g_j(\cdot)$ take the following form for Problem 2.1

$$\begin{aligned} f(n_1, \ldots, n_H) &= \sum_{h \in J} \left(\frac{N_h^2 S_h^2}{n_h} - N_h S_h^2 \right), \\ h(n_1, \ldots, n_H) &= \sum_{h \in J} n_h - n = 0, \\ g_h(n_h) &= n_h - u_h \leq 0, \quad \forall h \in J. \end{aligned} \tag{B.3}$$

More detailed discussion on the KKT conditions can be found in [6, Part III, Chapter 11.8, p. 341], or [1].

C. Algorithms - R code

C.1. The noptcond

This is the *noptcond* algorithm in R, as originally proposed in [5, Section 3 Programme code, p. 158] by Gabler, Ganninger and Munnich. Comments originated by the author of this thesis are enclosed in square brackets [ ].

```r
noptcond <- function(dh ,mh ,Mh ,n) {
  H <- length(dh) # Number of strata
  m <- sum(mh) # Total of lower bounds
  M <- sum(Mh) # Total of upper bounds
  U1 <- order(dh/mh ,decreasing=FALSE) # Ordered set equivalent to U(L_1)
  U3 <- order(dh/Mh ,decreasing=TRUE) # Ordered set equivalent to U(L_2)
  nopt <- n*dh/sum(dh) # Naiive allocation
  Hk <- 1:H # Index 1 to H
  R <- 0
  while(R<=H){
    i <- 0
    while(i<=R){
      s1 <- as.integer (0)
      s3 <- as.integer(H+1)
      if(i<R){s1 <- as.integer(U1[1:(R-i)])}
      if(i >0){ s3 <- as.integer(U3[1:i])}
      if(!any(s1%in%s3)){
        noptU1 <- numeric(length(s1))
        noptU3 <- numeric(length(s3))
        if(i<R){ noptU1 <- mh[s1]}
        if(i >0){ noptU3 <- Mh[s3]}
        # Omit solutions which violate the requirement
        # sum(noptU1 ,noptU3 )<=n
        if(sum(noptU1 ,noptU3 )<=n){
          if(i==0 & R==0){ s2 <- Hk}
          if(i==0 & R>0) {s2 <- Hk[-s1]} # [it starts (i.e. i=0) with moving all
              strata to JL]
          if(i>0 & R==i){s2 <- Hk[-s3]} # [it ends (i.e. i=R) with moving all strata JU]
          if(i>0 & R>i) {s2 <- Hk[-c(s1 ,s3)]} # [in the midle (i.e. 0 < i < R), strata are moving to JL and JU]
          noptU2 <- (n-sum(noptU1)-sum(noptU3 ))*dh[s2]/sum(dh[s2]) # [this is allocation after moving strata to JL, JU]
          # If all conditions are met , stop and return solution
          if(sum(noptU2 <mh[s2 ])==0 & sum(noptU2 >Mh[s2 ])==0){ #[note: all conditions are met at non-strict inequalities]
            if(i==0 & R==0){ nopt1 <- cbind(s2 ,noptU2 )}
            if(i==0 & R >0){ nopt1 <- cbind(c(s1 ,s2), c(noptU1 ,noptU2 ))}
            if(i>0 & R==i){ nopt1 <- cbind(c(s2 ,s3), c(noptU2 ,noptU3 ))}
            if(i>0 & R>i){ nopt1 <- cbind(c(s1 ,s2 ,s3), c(noptU1 ,noptU2 ,noptU3 ))}
            return(nopt1[order(nopt1 [ ,1]) ,2])
          }
        }
      }
      i <- i+1
    }
    R <- R+1
  }
}
```

C.2. Sufficient noptcond

This is the modified version of the *noptcond* algorithm in R, assuring solution returned is optimal.

```r
#' Optimal sample allocation in stratified random sampling scheme.
#' Maximize: \deqn{ \sum_{h \in J} \frac{N_h^2 S_h^2}{n_h} - \sum_{h \in J} N_h S_h^2 }
#' Subject to: \deqn{ \sum_{h \in J} = n } and \deqn{ l_h \leq n_h \leq u_h \forall h \in J }
#'
#' @param d numeric vector, d = N*S, where N - strata sizes, S - strata standard deviations.
#' @param l numeric vector, lower bounds on sample sizes in strata.
#' @param u numeric vector, upper bounds on sample sizes in strata.
#' @param n numeric scalar, total sample size.
#'
```

```r
#' @return numeric vector with optimal allocation or error if the problem is infeasible
#'
#' @examples
#'
#' noptcond_sufficient(d = c(2000, 3000),
#'                     l = c(30, 40),
#'                     u = c(50, 200),
#'                     n = 160)
#'
noptcond_sufficient <- function(d, l, u, n) {

  `&` <- function(x, y) {
    if (!all(x)) return(FALSE)
    if (!all(y)) return(FALSE)
    return(TRUE)
  }

  if(n > sum(u) | n < sum(l))
    stop("n is not feasible")

  dL <- d/l
  dU <- d/u

  J_dL <- order(dL)
  J_dU <- order(dU, decreasing = TRUE)
  H <- length(d)
  J <- seq_along(d)

  for( R in 0:(H-1) ) { # if problem is feasible, solution must be found in at most H-1 iterations

    for( i in 0:R ) {

      JL <- J_dL[0:(R-i)]
      JU <- J_dU[0:i]

      if( !any(JL %in% JU) & (n > sum(l[JL], u[JU])) ) {

        J0 <- if(R != 0) J[-c(JL, JU)] else J
        noptJ0 <- (n - sum(l[JL], u[JU])) * ( d[J0] / sum(d[J0]) )

        if( ( l[J0] <= noptJ0 & noptJ0 <= u[J0] ) &
            ( d[J0] / noptJ0 >= suppressWarnings(max(dL[JL])) &
              d[J0] / noptJ0 <= suppressWarnings(min(dU[JU])) ) )

          # form solution
          return(c(l[JL], noptJ0, u[JU])[order(c(JL, J0, JU))])

      }
    }
  }
}

# Note: max() and min() functions are wrapped with suppressWarnings()
# to suppress a warning, thrown when the argument is of length 0,
# which is a behavior of base::max() and base::min().
```

Bibliography

[1] Avriel, M. *Nonlinear Programming. Analysis and Methods*, 1976 ed. Dover Publications, Inc., 2003.

[2] Bertsekas, D. P. *Nonlinear Programming*, 3 ed. Athena Scientific, 2016.

[3] Boyd, S., and Vandenberghe, L. *Convex Optimization*, 2004 ed. Cambridge University Press, 2004. Seventh printing with corrections, 2009.

[4] Friedrich, U., Münnich, R., de Vries, S., and Wagner, M. Fast integer-valued algorithms for optimal allocations under constraints in stratified sampling. *Computational Statistics & Data Analysis 92* (December 2015), 1–12.

[5] Gabler, S., Ganninger, M., and Münnich, R. Optimal allocation of the sample size to strata under box constraints. *Metrika 75* (2012), 151–161.

[6] Luenberger, D. G., and Ye, Y. *Linear and Nonlinear Programming*, 3 ed. Springer, 2008.

[7] Murphy, J. Topological Proofs of the Extreme and Intermediate Value Theorems. `http://www.math.uchicago.edu/~may/VIGRE/VIGRE2008/REUPapers/Murphy.pdf`. Accessed: 2019-06-15.

[8] Neyman, J. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society 97*, 4 (1934), 558–606.

[9] Sarndal, C.-E., Swensson, B., and Wretman, J. *Model Assisted Survey Sampling.* Springer, 1992. First softcover printing, 2003.

[10] Stenger, H., and Gabler, S. Combining random sampling and census strategies - justification of inclusion probabilities equal to 1. *Metrika 61*, 2 (2005), 137–156.

[11] Tschuprow, A. A. On the mathematical expectation of the moments of frequency distributions in the case of correlated observations. *Metron 2* (1923), 461–493, 646–683.

[12] Wright, T. Exact optimal sample allocation: More efficient than Neyman. *Statistics & Probability Letters 129* (October 2017), 50–57.