`allocation`: Exact Optimal Allocation Algorithms for Stratified Sampling

Andrew M. Raim

When sampling from a finite population, the N units are often partitioned into strata based on predetermined criteria. The survey designer must determine the number of units to sample from each strata. The `allocation` package implements several algorithms from Wright (2012) and Wright (2017) which reconsider Neyman's classic method of allocating a given sample size n among such strata (Neyman 1934). These algorithms provide optimal integer-valued solutions to minimize the variance of an estimator for the population total.

Table of contents

Disclaimer and Acknowledgments **1**

1 Introduction **2**

2 Overview of the Package **2**

3 Examples **4**
3.1 Allocation for Fixed Overall Sample Size 4
3.2 Allocation for a Desired Precision 6

References **7**

Disclaimer and Acknowledgments

This document is released to inform interested parties of ongoing research and to encourage discussion of work in progress. Any views expressed are those of the author and not those of the U.S. Census Bureau.

Thanks to Tommy Wright (U.S. Census Bureau) for discussions which prompted to the development of this package, and for providing a review of the materials.

Although there are no guarantees of correctness of the `allocation` package, reasonable efforts will be made to address shortcomings. Comments, questions, corrections, and possible improvements can be communicated through the Github repository for the package (https://github.com/andrewraim/allocation).

Document was compiled 2025-08-22 04:54:12 EDT and corresponds to `allocation` version 0.1.0. **Contact:** andrew.raim@gmail.com, Center for Statistical Research & Methodology, U.S. Census Bureau, Washington, DC, 20233, U.S.A.

1 Introduction

Suppose there are N units in a population which is partitioned into H strata of known sizes $N_1, \dots, N_H$. A sample consisting of $n_1, \dots, n_H$ units will be taken from the corresponding strata with $n_h \geq 1$ for all h. Therefore, the overall sample size will be $n = \sum_{h=1}^H n_h$. Denote the population mean and variance for the hth stratum as

$$\bar{Y}_h = \sum_{j=1}^{N_h} Y_{hj} \quad \text{and} \quad S_h^2 = \frac{1}{N_h - 1} \sum_{j=1}^{N_h} (Y_{hj} - \bar{Y}_h)^2,$$

where Y_{hj} is the value of the variable of interest for the jth unit. To estimate the population total of $T_Y = \sum_{h=1}^H N_h \bar{Y}_h$ from the sampled units, consider the estimator

$$\hat{T}_Y = \sum_{h=1}^H N_h \bar{y}_h,$$

where $\bar{y}_h$ is the sample mean from the hth strata. The variance of $\hat{T}_Y$ is

$$\text{Var}(\hat{T}_Y) = \sum_{h=1}^H \frac{N_h}{n_h} (N_h - n_h) S_h^2, \tag{1}$$

Neyman's allocation method minimizes (1) with respect to $n_1, \dots, n_H$ as the optimization variables, subject to the constraint $n = \sum_{h=1}^H n_h$ for a given n. Regarding the variables $n_1, \dots, n_H$ as real numbers, Lagrange's method can be used to obtain the solution

$$n_h = n \frac{N_h S_h}{\sum_{\ell=1}^H N_\ell S_\ell}, \quad h = 1, \dots, H.$$

Rounding is then used to obtain integer-valued $n_1, \dots, n_H$ which are needed in practice; however, rounding may not yield an optimal integer solution. The allocation methods in Wright (2012) and Wright (2017) address this by directly obtaining integer solutions. Exploiting the structure of (1), units are iteratively placed into strata to yield an optimal integer solution. Additionally, these methods support nonnegative real-valued bounds a_h, b_h such that $a_h \leq n_h \leq b_h$, $a_h > 0$ and $b_h \leq N_h$. We consider two methods in particular. Algorithm III of Wright (2017) assumes a target sample size of n_0 and finds an optimal allocation such that $\sum_{h=1}^H n_h = n_0$. Algorithm IV of Wright (2017) assumes a target variance assumes a target variance V_0 and finds an optimal allocation with the smallest overall sample size $\sum_{h=1}^H n_h$ such that (1) is no larger than V_0. Algorithms III and IV of Wright (2017) are summarized as Algorithms 1 and 2 here, respectively. See Wright (2017) for further details.

The remainder of the vignette proceeds as follows. Section 2 gives an overview of the `allocation` package and its interface. Section 3 demonstrates use of the package on several examples from Wright (2017).

2 Overview of the Package

The `allocation` package makes use of the `Rmpfr` package to handle very large numbers while avoiding loss of precision. Furthermore, users may consider encode such values with `Rmpfr` rather than standard floating point numbers; especially for numbers such as target variances which may be very large.

```r
library(Rmpfr)
```

The following functions implement Neyman's allocation method, Algorithm 1, and Algorithm 2, respectively.

Algorithm 1 Optimal allocation for a fixed overall sample size.

inputs
  n_0: desired overall sample size.
  $N_1, \dots, N_H$: population sizes.
  $S_1, \dots, S_H$: standard deviations.
  $a_1, \dots, a_H$: lower bounds.
  $b_1, \dots, b_H$: upper bounds.
end inputs
Let $n_h = a_h$ for $h = 1, \dots, H$.
while $\sum_{h=1}^{H} n_h < n_0$ **do**
  $P_h \leftarrow N_h S_h / \sqrt{n_h(n_h+1)}$ if $n_h + 1 \leq b_h$, $P_h \leftarrow 0$ otherwise, for $h = 1, \dots, H$.
  $h \leftarrow \text{argmax}(P_1, \dots, P_H)$.
  $n_h \leftarrow n_h + 1$.
end while
return $n_1, \dots, n_H$.

Algorithm 2 Optimal allocation for a desired precision.

inputs
  V_0: desired variance target.
  $N_1, \dots, N_H$: population sizes.
  $S_1, \dots, S_H$: standard deviations.
  $a_1, \dots, a_H$: lower bounds.
  $b_1, \dots, b_H$: upper bounds.
end inputs
Let $n_h = a_h$ for $h = 1, \dots, H$.
Let $V = \sum_{h=1}^{H} N_h (N_h - n_h) S_h^2 / n_h$
while $V > V_0$ and $\sum_{h=1}^{H} n_h < \sum_{h=1}^{H} N_h$ **do**
  $P_h \leftarrow N_h S_h / \sqrt{n_h(n_h+1)}$ if $n_h + 1 \leq b_h$, $P_h \leftarrow 0$ otherwise, for $h = 1, \dots, H$.
  $h \leftarrow \text{argmax}(P_1, \dots, P_H)$.
  $n_h \leftarrow n_h + 1$.
  $V \leftarrow \sum_{h=1}^{H} N_h (N_h - n_h) S_h^2 / n_h$
end while
return $n_1, \dots, n_H$.

```
allocate_neyman =
function (n0, N, S, control = allocation_control())
```

```
allocate_fixn =
function (n0, N, S, lo = NULL, hi = NULL, control = allocation_control())
```

```
allocate_prec =
function (v0, N, S, lo = NULL, hi = NULL, control = allocation_control())
```

The arguments are as follows:

- `n0`: target sample size n_0,
- `v0`: target variance V_0,
- `N`: the vector $(N_1, \dots, N_H)$,
- `S`: the vector $(S_1, \dots, S_H)$,
- `lo`: the vector $(a_1, \dots, a_H)$,
- `hi`: the vector $(b_1, \dots, b_H)$.

The argument `control` contains additional arguments and can be created with the following function. See its manual page for further information.

```
print_interface(allocation_control)
```

```
allocation_control =
function (verbose = FALSE, bits = 256, tol = 1e-10, digits = 4)
```

Several accessors are provided to operate on results from the allocation methods.

```
out = allocate_fixn(n0, N, S)
allocation(out)  ## Extract allocation (n[1], ..., n[H]).
print(out)       ## Print table with allocation and other information.
```

3 Examples

3.1 Allocation for Fixed Overall Sample Size

Here we demonstrate Algorithm 1 using an example in Wright (2017).

```
N = c(47, 61, 41)
S = sqrt(c(100, 36, 16))
lo = c(1,2,3)
hi = c(5,6,4)
n0 = 10

out1 = allocate_fixn(n0, N, S, lo, hi)
print(out1)
```

```
  lo hi n
1  1  5 4
2  2  6 3
3  3  4 3
--
Made 4 selections
Target n: 10
Achieved v: 101,290.3333
```

Note that rows labels are the stratum indices $1, \dots, H$. The columns `lo`, `hi`, and `n` correspond to the vectors $a_1, \dots, a_H$, $b_1, \dots, b_H$, and $n_1, \dots, n_H$, respectively. To see details justifying each selection, run `allocate_fixn` with the `verbose` option enabled.

```
out1 = allocate_fixn(v0, N, S, lo, hi, control = allocation_control(verbose = TRUE))
```

Let us compare the above results to Neyman allocation.

```
out2 = allocate_neyman(n0, N, S)
print(out2)
```

```
   N      S      n
1 47 10.000 4.7000
2 61 6.0000 3.6600
3 41 4.0000 1.6400
--
v: 92,448.0000
```

The number of decimal points in the output can be changed using the control object.

```
print(out2, control = allocation_control(digits = 2))
```

```
   N    S    n
1 47 10.0 4.70
2 61 6.00 3.66
3 41 4.00 1.64
--
v: 92,448.00
```

Extract the allocation as a numeric vector using the `allocation` accessor function.

```
allocation(out1)  ## allocate_fixn result
```

```
[1] 4 3 3
```

```
allocation(out2)  ## allocate_neyman result
```

```
[1] 4.70 3.66 1.64
```

3.2 Allocation for a Desired Precision

Run Algorithm 2 using an example in Wright (2017). Since our target variance `v0` is a very large number, we pass it as an `mpfr` object to avoid loss of precision.

```
H = 10
v0 = mpfr(388910760, 256)^2
N = c(819, 672, 358, 196, 135, 83, 53, 40, 35, 13)
lo = c(3, 3, 3, 3, 3, 3, 3, 3, 3, 13)
S = c(330000, 518000, 488000, 634000, 1126000, 2244000, 2468000, 5869000,
    29334000, 1233311000)

print(data.frame(N, S, lo))
```

```
     N          S lo
1  819     330000  3
2  672     518000  3
3  358     488000  3
4  196     634000  3
5  135    1126000  3
6   83    2244000  3
7   53    2468000  3
8   40    5869000  3
9   35   29334000  3
10  13 1233311000 13
```

```
out1 = allocate_prec(v0, N, S, lo)
print(out1)
```

```
   lo  hi  n
1   3 819  4
2   3 672  5
3   3 358  3
4   3 196  3
5   3 135  3
6   3  83  3
7   3  53  3
8   3  40  3
9   3  35 13
10 13  13 13
--
Target v0: 151,251,579,243,777,600.0000
Achieved v: 149,400,057,961,841,025.6410
```

To see details justifying each selection, we can run `allocate_prec` with the `verbose` option enabled.

```
out1 = allocate_prec(v0, N, S, lo, control = allocation_control(verbose = TRUE))
```

Compare the above results to Neyman allocation. Here, we first need to compute a target sample size. This is done with a given cv and revenue data; see Wright (2017) for details. We also exclude the 10th stratum from the allocation procedure, as it is a certainty stratum; its allocation is considered fixed at 13.

```r
cv = 0.042
rev = mpfr(9259780000, 256)
n = sum(N[-10] * S[-10])^2 / ((cv * rev)^2 + sum(N[-10] * S[-10]^2))
out2 = allocate_neyman(n, N[-10], S[-10])
print(out2)
```

```
    N              S       n
1 819    330,000.0000  3.8874
2 672    518,000.0000  5.0068
3 358    488,000.0000  2.5128
4 196    634,000.0000  1.7873
5 135  1,126,000.0000  2.1864
6  83  2,244,000.0000  2.6789
7  53  2,468,000.0000  1.8814
8  40  5,869,000.0000  3.3766
9  35 29,334,000.0000 14.7672
--
v: 151,251,579,243,777,625.5132
```

Extract the final allocations.

```r
allocation(out1)  ## allocate_prec result
```

```
 [1]  4  5  3  3  3  3  3  3 13 13
```

```r
allocation(out2)  ## allocate_neyman result
```

```
[1]  3.887378  5.006774  2.512822  1.787328  2.186408  2.678921  1.881395
[8]  3.376627 14.767205
```

References

Neyman, Jerzy. 1934. "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): 558–625. http://www.jstor.org/stable/2342192.

Wright, Tommy. 2012. "The Equivalence of Neyman Optimum Allocation for Sampling and Equal Proportions for Apportioning the U.S. House of Representatives." *The American Statistician* 66 (4): 217–24. https://doi.org/10.1080/00031305.2012.733679.

———. 2017. "Exact Optimal Sample Allocation: More Efficient Than Neyman." *Statistics & Probability Letters* 129: 50–57. https://doi.org/10.1016/j.spl.2017.04.026.