Computational Statistics - Assignment 2

Cay Rahn, David Pomerenke, Alexander Prochnow

Our code can be found at https://github.com/Zianor/computational-statistics-2.

I. Complete enumeration bootstrap

We use `combinations_with_replacement` from Python's `itertools` library for producing the complete enumeration. For calculating the associated probabilities (see [1]), we use the `multinomial` density function from *scipy.stats*. The method produces 77558760 possible enumerations, and runs for about an hour on a laptop for the calculation of the statistics, and another hour for the calculation of the probabilities. With parallelization via *joblib* we can cut this runtime down proportionally to the number of jobs. The resulting distribution is shown in Figure 1 and compared to the Monte Carlo bootstrapped distribution.

II. Gray codes

The original gray codes are a way to encode integers in such a way that a change to the next number is only reflected by the change of a single bit. In a similar way graycodes can be created for compositions. For $n = 3$ the 10 produced combinations are encoded in the following order:

$$300, 210, 120, 030, 021, 111, 201, 102, 012, 003 \qquad (1)$$

Each of this combinations is a representation of a sample. This order ensures that from one combination to another only one element is changed. This can simplify the calculations of several sample statistics and can therefore result in a significant speedup when computing the complete enumeration. This will be explained in more detail in section III.

The generation of the gray codes is done by using an iterative algorithm described by Paul Klingsberg in [2].

III. Speedup from Gray codes

A. Theoretical considerations

Our analysis is inspired by [3, sec. 3.3].

The speedup can be calculated by comparing the runtime (here defined as the absolute number of atomic calculations, *not* in asymptotic terms) of calculating the metric of interest a) from scratch based on a sample, vs b) based on results (and intermediate results) of the iteration for the previous sample, and the differenes between the previous sample and the current sample.

Diaconis and Holmes [3] explain that the mean – and statistics that are based on the mean – are especially attractive for runtime reductions, because previous results can be reused effectively. They argue that calculating the mean from scratch takes $n - 1$ operations, while calculating it based on the previous mean just takes 2 operations, so there is a speedup of $n - 3$ operations in absolute terms. We note the mean can also be computed recursively in $\frac{n}{2}\lceil \log \frac{n}{2} \rceil = \frac{n}{2}\lceil \log n \rceil - 1$, which is parallelizable and only takes $\lceil \log n \rceil - 1$ operations on parallel hardware, so the speedup is $\lceil \log n \rceil - 3$.

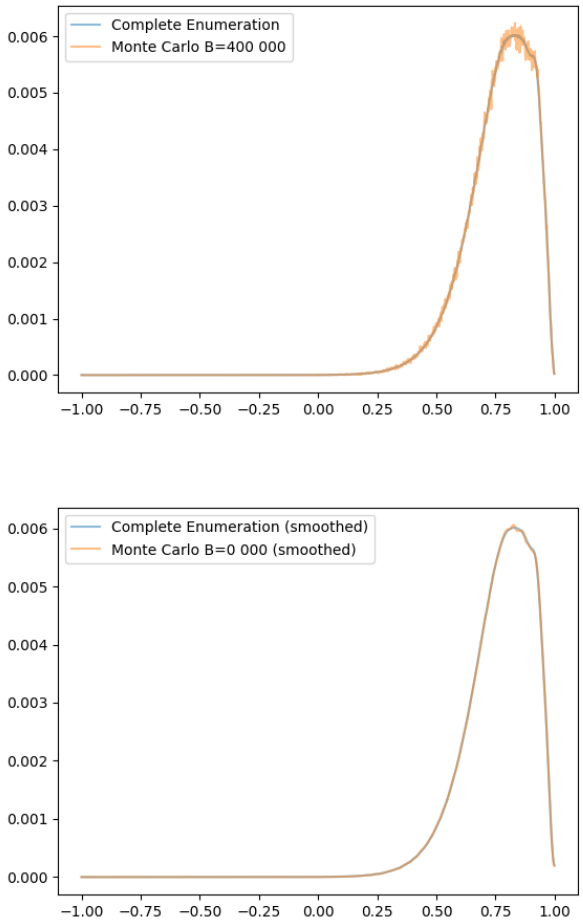

Fig. 1. Complete enumeration and Monte Carlo bootstrap distributions for the Pearson correlation coefficient. Top: Raw histograms with 1000 bins. Bottom: Same histograms with 1000 bins, smoothed with a Gaussian filter with $\sigma = 3$.

It is more interesting to look at the relative speedup rather than the absolute number of operations that can be omitted, and for that purpose we need to take into account the runtime for calculating the statistic of interest, which is here the Pearson correlation coefficient.

First we consider the sequential case. We count n operations

for subtracting $\bar{X}$ from each X_i and another n operations for subtracting $\bar{Y}$ from each Y_i, to obtain values that are centered around 0 for both X and Y. Then there are n multiplications and $n-1$ additions each for calculating $Cov(X,Y)$, $Var(X)$, and $Var(Y)$, respectively, and a final multiplication and division for obtaining the correlation coefficient from these values. Overall, these are $6n+3(n-1)+2 = 9n-1$ operations. We add the operations for calculating the two means and arrive at $(9n-1)+2(n-1) = 11n-3$. As explained above, we save $n-3$ operations, so that is a speedup of $\frac{n-3}{11n-3}$, which is $\approx \frac{3}{11} \approx 9\%$ for large n and even less for small n, so it is not really impressive.

Next we consider the parallelized case. Centering X and Y can now happen in a single operation, if we have already calculated the mean. Calculating the covariance and the variances only takes another single operation, and lastly we still have to multiply and divide, so we arrive at 4 operations, plus the $\lceil \log n \rceil - 1$ operations for calculating the mean that we have explained above, so $\lceil \log n \rceil + 3$ operations overall. The relative speedup is thus $\frac{\lceil \log n \rceil - 3}{\lceil \log n \rceil + 3}$, which approximates 100% for large n – very nice! –, but is smaller for small n and goes down to -100% for $n = 1$, that is a slowdown by factor 2.

Considering our law school examination data, we have $n = 15$. That is a speedup of 7.4% in the sequential case and of $\frac{3-3}{3+3} = 0$ – meaning no speedup but also no slowdown – in the parallel case.

B. Implementation with speedup

For each sample the pearson's correlation coefficient is calculated with the formula $r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}}$. Although a speedup in the mean computation is possible theoretically, there is overhead generated when checking which values must be removed and added. Therefore this was not further pursued in the scope of this assignment. Instead, the pearson's formula was rearranged to the following form: $r_{xy} = \frac{n*\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n*\sum y_i^2 - (\sum y_i)^2}}$. In this form there are several parts that can be computed one time globally instead of computing them in every iteration. The first part consists of the cross products between x_i and y_i which can be saved for each i. Additionally x_i^2 and y_i^2 can computed once and saved for each i as well.

For a full enumeration we have $\binom{2n-1}{n-1}$ samples. For a calculation without speedup there are $11n-3$ operations per sample as explained in section III-A. When using the above explained speedup this is reduced to $3*n$ initial operations and 12 for every sample. This results in $\binom{2n-1}{n-1}11$ operations compared to $3*n+12\cdot\binom{2n-1}{n-1}*2$ operations. For $n=6$ this results in the traditional implementation being 5.2 times slower than the one with the graycodes.

$$\text{Number of samples} : \binom{11}{5} = 462$$

$$\text{Traditional} : \binom{11}{5}\cdot 11\cdot 6 - 3 = 29106 \text{ Operations}$$

$$\text{With Graycode} : 3*n+2+\binom{11}{5}*12 = 5564 \text{ Operations}$$

In practice the runtime varied significantly between different runs but in general it corresponded to the theoretical speedup (e.g. 8047 iterations per second for graycodes vs. 1879 iterations per second for the traditional implementation).

IV. Removals

The experiments for this task were carried out in the removals.ipynb notebook.

In order to create a sample where the results of the Monte Carlo and the complete enumerations simulation look more similar, we decided on first trying to removing the outliers and observe if this has the intended effect. For our data, these are the observations 0 and 10, which lie the furthest away from a imagined line going through the center of the data (see Figure 2).

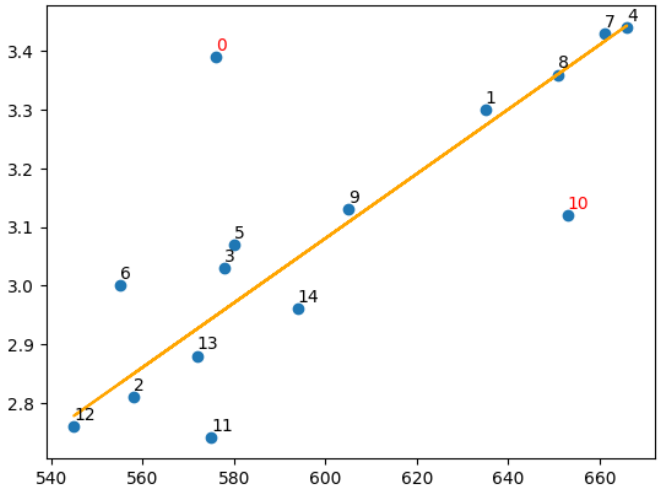

Fig. 2. Dataset with index labels. In orange a linear regression line was plotted that was fit using all samples except the two outliers (labeled in red).

With the outliers identified, we investigated three experimental conditions:
(A) Only removing observation 0,
(B) only removing observation 10 or
(C) removing both observation 0 and 10.
We can then assess which removal results in the most similar correlation coefficient histograms: Figure 3 shows the resulting histograms, with the x-axis limited to the bulk of the distribution $(0.5, 1)$ in order to see the effects more clearly. Compared to the baseline (i.e. no removals) we see that in each condition

the fluctuations in the Monte Carlo distribution decrease and the distribution becomes smoother as well as closer to the complete enumeration distribution. This effect is maximized in condition C, where we removed both outliers.

This may be due to the fact that removing outliers causes the (linear) correlation in the data to be higher, resulting in less variance of the mean correlations. A higher correlation causes the histogram to be more skewed towards the right, and lower variance in the sampled mean correlations cause the histogram to be narrower as well as smoother, closer to the complete enumeration distribution.

V. Monte Carlo vs complete enumeration bootstrap

It is claimed in [3] that the complete enumeration bootstrap is qualitatively superior to the Monte Carlo bootstrap, citing as evidence a ”corner” that is only visible in the histogram of the complete enumeration bootstrap but not the Monte Carlo bootstrap (see Figure 4). That is quite natural, because for their histogram for the Monte Carlo bootstrap is rather fuzzy. We find that after smooting the histogram with a Gaussian filter to remove the noise, the ”corner” is also visible in Monte Carlo bootstrap samples (see Figure 1, Figure 5).

The difference between Monte Carlo and complete enumeration bootstrap is that the first is a random sample of the distribution that is perfectly represented by the second. The first approaches the second in the limit (this can be easily shown from the law of large numbers as an exercise). In a sense, the complete enumeration bootstrap is therefore superior: It is not noisy but beautifully smooth, such that one can inspect its curvature even without smoothing. But this apparent superiority and perfection is deceptive; for the complete enumeration bootstrap is still just based on a sample and does not perfectly represent the population, as perfect as it may appear.

In [3] outlier detection is presented as an exemplary application of the complete enumeration bootstrap, and this idea is reflected in the present assignment. We have two concerns with this. First, such detection can also be carried out with the Monte Carlo bootstrap in combination with smoothed visualizations. Second, the method is not very reproducible, since it depends on the visual detection of corners in a histogram. We would rather suggest to use the impact of a data point on the numerical value of the correlation coefficient as an outlier detection procedure.

References

[1] Nicholas I Fisher and Peter Hall. “Bootstrap algorithms for small samples”. In: *Journal of statistical planning and inference* 27.2 (1991), pp. 157–169.

[2] Paul Klingsberg. “A gray code for compositions”. In: *Journal of Algorithms* 3.1 (1982), pp. 41–44. DOI: 10.1016/0196-6774(82)90006-2.

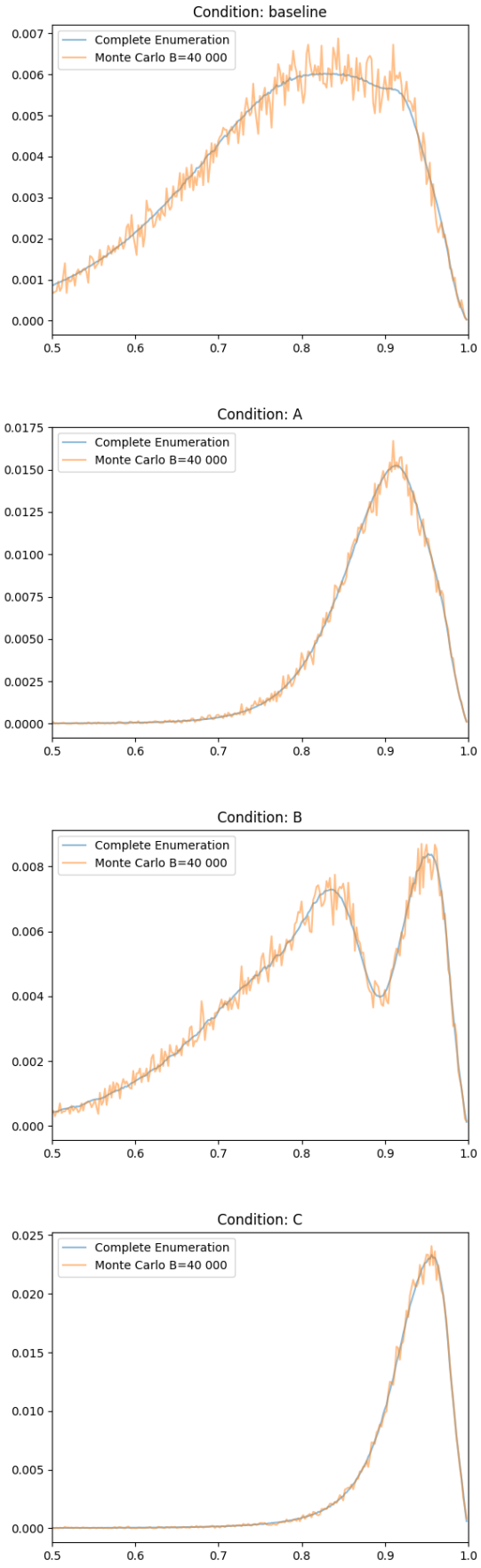

Fig. 3. Correlation coefficient histograms for both complete enumeration and Monte Carlo bootstrap when removing different outliers. X-axis focused on the bulk of the distribution $(0.5, 1)$.

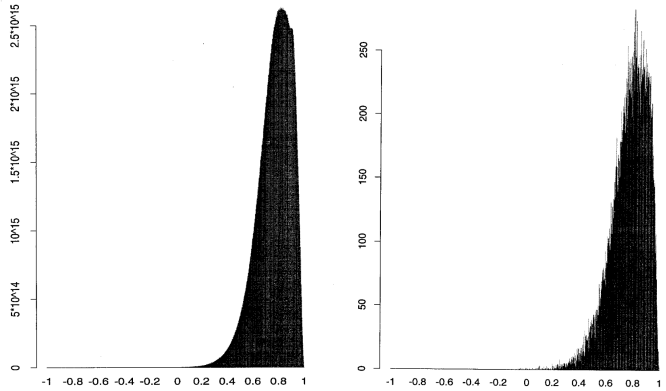

Fig. 4. Left: Histogram of the complete enumeration bootstrap for the law school examination data. Right: Same histogram for the Monte Carlo bootstrap with $B = 40000$ samples. Reproduced from [3, p. 288].

[3] Persi Diaconis and Susan Holmes. “Gray codes for randomization procedures”. In: *Statistics and Computing* 4 (1994), pp. 287–302.

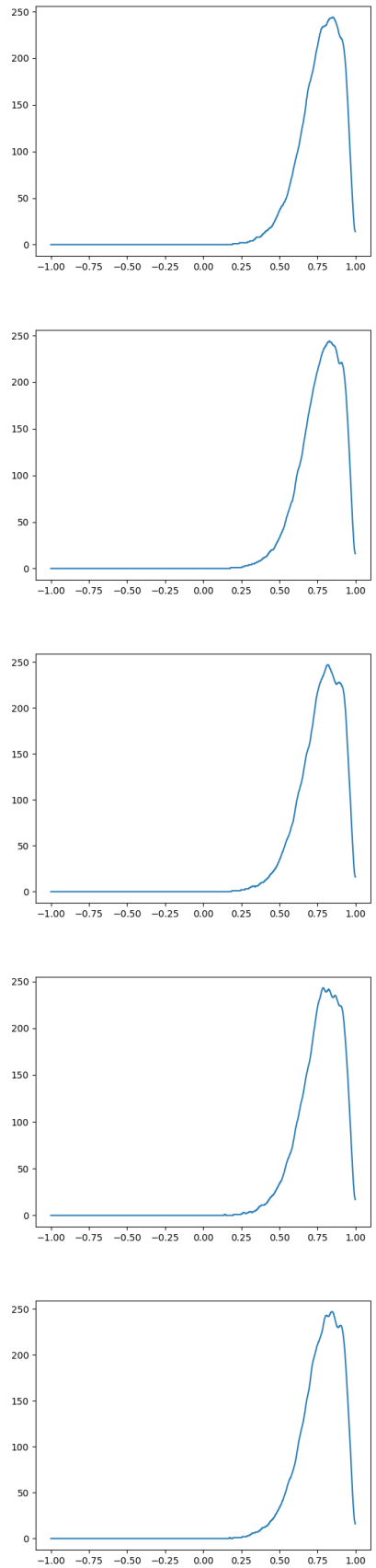

Fig. 5. Histograms for the Monte Carlo bootstrap on the law school examination data for $B = 40000$ samples, for 5 different random seeds, smoothed with a Gaussian filter. The ”corner” that is visible in the complete enumeration bootstrap is also visible in most of the histograms here.