kendallknight: An R Package for Efficient Implementation of Kendall's Correlation Coefficient Computation

Corresponding author:

Last updated: 2024-08-31 13:17

Contents

1 Abstract

The kendallknight package introduces an efficient implementation of Kendall's correlation coefficient computation, significantly improving the processing time for large datasets without sacrificing accuracy. The kendallknight package, following Knight (1966) and posterior literature, reduces the computational complexity resulting in drastic reductions in computation time, transforming operations that would take minutes or hours into milliseconds or minutes, while maintaining precision and correctly handling edge cases and errors. The package is particularly advantageous in econometric and statistical contexts where rapid and accurate calculation of Kendall's correlation coefficient is desirable. Benchmarks demonstrate substantial performance gains over the base R implementation, especially for large datasets.

2 Introduction

Kendall's correlation coefficient is a non-parametric measure of association between two variables and it is particularly useful to compute pseudo-R^2 statistics in the context of Poisson regression with fixed effects (Silva and Tenreyro 2006).

The current Kendall's correlation coefficient implementation in R has a computational complexity of $O(n^2)$, which can be slow for large datasets (R Core Team 2024). While R features a highly efficient multi-threaded implementation of the Pearson's correlation coefficient, the Kendall's case that is also multi-threaded can be particularly slow for large datasets (e.g. 10,000 observations or more).

We used C++ in the `kendallknight` package to compute the Kendall's correlation coefficient in a more efficient way, with a computational complexity of $O(n \log(n))$, following Knight (1966), Abrevaya (1999), Christensen (2005) and Emara (2024).

For a dataset with 20,000 observations, a computational complexity $O(n^2)$ involves 400 million operations and a computational complexity $O(n \log(n))$ requires approximately 198,000 operations to obtain the Kendall's correlation coefficient.

Our implementation can reduce computation time by several minutes or hours as we show in the benchmarks, and without sacrificing precision or correct handling of corner cases as we verified with exhaustive testing.

3 Definitions

Kendall's correlation coefficient is a pairwise measure of association and it does not require the data to be normally distributed. For two vectors x and y of length n, it is defined as (Knight 1966):

$$r(x,y) = \frac{c-d}{\sqrt{(c+d+e)(c+d+f)}},$$

where c is the number of concordant pairs, d is the number of discordant pairs, e is the number of ties in x and f is the number of ties in y.

The corresponding definitions for c, d, e and f are:

$$\begin{aligned}
c &= \sum_{i=1}^{n}\sum_{j\neq i}^{n} g_1(x_i, x_j, y_i, y_j),\\
d &= \sum_{i=1}^{n}\sum_{j\neq i}^{n} g_2(x_i, x_j, y_i, y_j),\\
e &= \sum_{i=1}^{n}\sum_{j\neq i}^{n} g_3(x_i, x_j) g_4(y_i, y_j),\\
f &= \sum_{i=1}^{n}\sum_{j\neq i}^{n} g_4(x_i, x_j) g_3(y_j, y_i).
\end{aligned}$$

The functions g_1, g_2, g_3 and g_4 are indicators defined as:

$$
\begin{aligned}
g_1(x_i, x_j, y_i, y_j) &= \begin{cases} 1 & \text{if } (x_i - x_j)(y_i - y_j) > 0, \\ 0 & \text{otherwise,} \end{cases} \\
g_2(x_i, x_j, y_i, y_j) &= \begin{cases} 1 & \text{if } (x_i - x_j)(y_i - y_j) < 0, \\ 0 & \text{otherwise,} \end{cases} \\
g_3(x_i, x_j) &= \begin{cases} 1 & \text{if } x_i = x_j \text{ and } y_i \neq y_j, \\ 0 & \text{otherwise,} \end{cases} \\
g_4(y_i, y_j) &= \begin{cases} 1 & \text{if } x_i \neq x_j \text{ and } y_i = y_j, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

Kendall's correlation correlation is a measure of the proportion of concordant pairs minus the proportion of discordant pairs corrected by the proportion of ties in the data, and it requires to compare $m = n(n-1)/2$ pairs of observations which is why its computational complexity is $O(n^2)$.

Without ties, or duplicates in the data, the Kendall's correlation coefficient simplifies to:

$$
r(x, y) = \frac{c-d}{c+d} = \frac{c-d}{m} = \frac{4c}{n(n-1)} - 1
$$

A naive implementation consisting in comparing all pairs of observations would require $O(n^2)$ operations. However, the Kendall's correlation coefficient can be computed more efficiently by sorting the data and using the number of inversions in the data to compute the correlation in $O(n \log(n))$ operations by using binary trees (Knight 1966).

An array that represents a binary tree has a search operation with a computational complexity of $O(\log(n))$ and an insertion operation with a computational complexity of $O(n)$ (Abrevaya 1999; Christensen 2005). The resulting computational complexity of the search and insert operations in an array is $O(n)$ (Emara 2024). Repeating the search and insert operation for each element in the array results in a computational complexity of $O(n^2)$, resulting in the same computational complexity as the naive implementation.

4 Implementation

Using a merge sort with a binary tree with a depth $1 + \log_2(n)$ results in a search and insert operation with a computational complexity of $O(\log(n))$, resulting in a computational complexity of $O(n \log(n))$ for the Kendall's correlation coefficient (Knight 1966; Emara 2024). An algorithm that conducts the following operations can compute the Kendall's correlation coefficient in an efficient way, with computational complexity of $O(n \log(n))$ instead of $O(n^2)$, as follows:

1. Sort the vector x and keep track of the original indices in a permutation vector.
2. Rearrange the vector y according to x.
3. Compute the total pairs m.
4. Compute the pairs of ties in x as $m_x = t_x(t_x + 1)/2$.
5. Compute the pairs of ties in y as $m_y = t_y(t_y + 1)/2$.
6. Compute the concordant pairs adjusted by the number of swaps in y by using a merge sort as $t = m - t_x - t_y + 2t_p$.
7. Compute the Kendall's correlation coefficient as $r(x, y) = t/(\sqrt{m - m_x}\sqrt{m - m_y})$.

The `kendallknight` package implements these steps in C++ and exports the Kendall's correlation coeeficient as a function that can be used in R by using the `cpp11` headers (Vaughan, Hester, and François 2023). Unlike existing implementations with $O(n \log(n))$ complexity, such as Filzmoser, Fritz, and Kalcher (2023), this implementation also provides dedicated functions to test the statistical significance of the computed correlation, and for which it uses a C++ port of the Gamma function that R already implemented in C (R Core Team 2024).

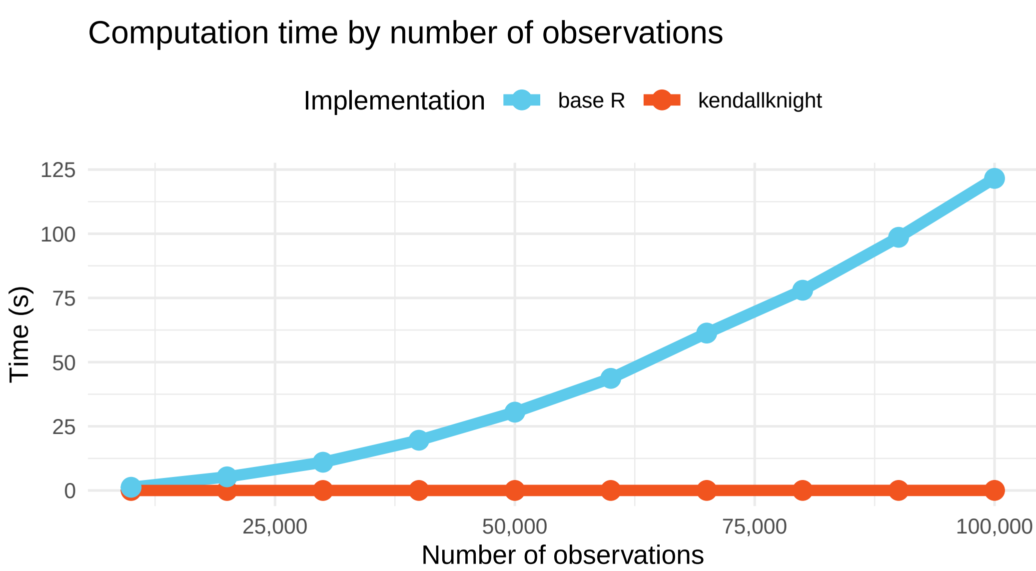
5 Benchmarks

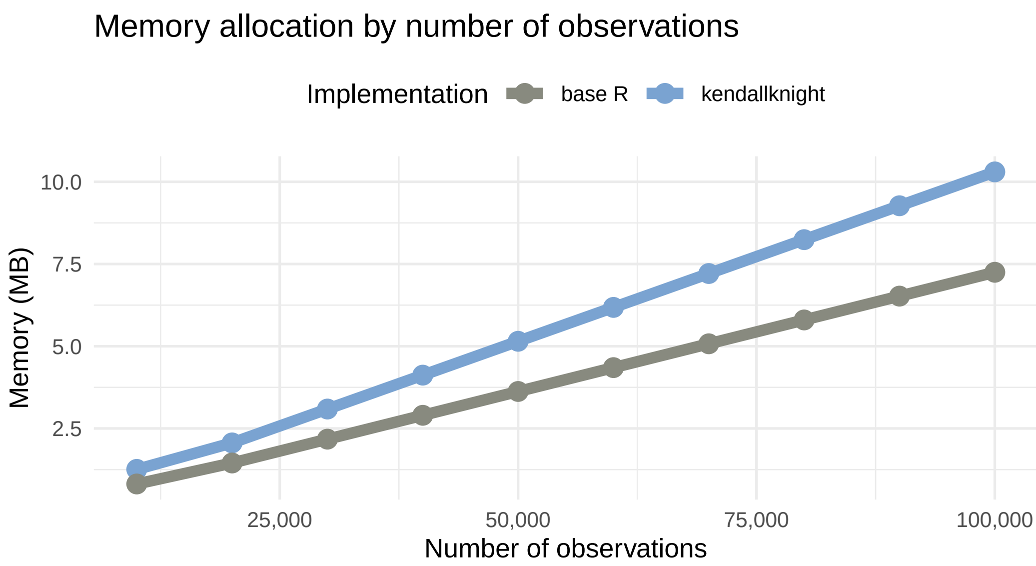
We tested the `kendallknight` package against the base R implementation of the Kendall correlation using the `cor` function with `method = "kendall"` for randomly generated vectors of different lengths. The results are shown in the following tables:

Number of observations	kendallknight median time (s)	base R median time (s)
10,000	0.003	1.251
20,000	0.010	5.313
30,000	0.011	11.002
40,000	0.014	19.578
50,000	0.017	30.509
60,000	0.021	43.670
70,000	0.024	61.310
80,000	0.029	77.993
90,000	0.031	98.614
100,000	0.035	121.552

Number of observations	kendallknight memory allocation (MB)	base R memory allocation (MB)
10,000	1.257	0.812
20,000	2.061	1.450
30,000	3.091	2.175
40,000	4.121	2.900
50,000	5.151	3.625
60,000	6.181	4.350
70,000	7.211	5.074
80,000	8.241	5.799
90,000	9.271	6.524
100,000	10.301	7.249

These results can be complements with the following plots:





As a reference, estimating the coefficients for a Poisson regression using a dataframe with around 28,000 observations, five variables and around 700 exporter-time and importer-time fixed effects takes around 0.4 seconds with the `capybara` package (Vargas Sepulveda 2024). To obtain summary tables for the same model, including clustered standard errors, significance and pseudo-R^2, it takes around 0.2 additional seconds using `kendallknight`. Using base R to compute the Kendall's correlation coefficient for the pseudo-R^2 statistic takes around 7.5 seconds without the rest of statistics.

The benchmarks were conducted on a ThinkPad X1 Carbon Gen 9 with the following specifications:

- Processor: Intel Core i7-1185G7 with eight cores
- Memory: 16 GB LPDDR4Xx-4266
- Operating System: Pop!_OS 22.04 based on Ubuntu 22.04
- R Version: 4.4.1
- BLAS Library: OpenBLAS 0.3.20

6 Testing

The package uses `testthat` for testing (Wickham 2011). The included tests are exhaustive and covered the complete code to check for correctness comparing with the base R implementation, and also checking corner cases and forcing errors by passing unusable input data to the user-visible functions. The current tests cover 100% of the code.

7 Installation and Usage

The `kendallknight` package is available on CRAN and can be installed using the following command:

```r
# CRAN
install.packages("kendallknight")

# GitHub
remotes::install_github("pachadotdev/kendallknight")
```

The package can be used as in the following example:

```r
library(kendallknight)

set.seed(200)
x <- rnorm(100)
y <- rnorm(100)

kendall_cor(x, y)
```

```
[1] 0.1288889
```

```r
kendall_cor_test(x, y, alternative = "less")
```

```
$statistic
[1] 0.1288889

$p_value
[1] 0.971286

$alternative
[1] "alternative hypothesis: true tau is less than 0"
```

8 Conclusion

The `kendallknight` package provides a fast and memory-efficient implementation of the Kendall's correlation coefficient with a computational complexity of $O(n \log(n))$, which is orders of magnitude faster than the base R implementation without sacrificing precision or correct handling of corner cases. For small vectors (e.g., less than 100 observations), the time difference is negligible. However, for larger vectors, the difference can be substantial. This package is particularly useful to solve bottlenecks in the context of econometrics and international trade, but it can also be used in other fields where the Kendall's correlation coefficient is required.

References

Abrevaya, Jason. 1999. "Computation of the Maximum Rank Correlation Estimator." *Economics Letters* 62 (3): 279–85. https://doi.org/10.1016/S0165-1765(98)00255-9.

Christensen, David. 2005. "Fast Algorithms for the Calculation of Kendall's τ." *Computational Statistics* 20 (1): 51–62. https://doi.org/10.1007/BF02736122.

Emara, Salma. 2024. "Khufu: Object-Oriented Programming Using C++." https://learningcpp.org/cover.html.

Filzmoser, Peter, Heinrich Fritz, and Klaudius Kalcher. 2023. *"pcaPP": Robust PCA by Projection Pursuit*. https://CRAN.R-project.org/package=pcaPP.

Knight, William R. 1966. "A Computer Method for Calculating Kendall's Tau with Ungrouped Data." *Journal of the American Statistical Association* 61 (314): 436–39. https://doi.org/10.1080/01621459.1966.10480879.

R Core Team. 2024. *"R": A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Silva, J. M. C. Santos, and Silvana Tenreyro. 2006. "The Log of Gravity." *The Review of Economics and Statistics* 88 (4): 641–58. https://doi.org/10.1162/rest.88.4.641.

Vargas Sepulveda, Mauricio. 2024. *"Capybara": Fast and Memory Efficient Fitting of Linear Models with High-Dimensional Fixed Effects*. https://pacha.dev/capybara/.

Vaughan, Davis, Jim Hester, and Romain François. 2023. *"Cpp11": A C++11 Interface for r's c Interface*. https://CRAN.R-project.org/package=cpp11.

Wickham, Hadley. 2011. "'Testthat': Get Started with Testing." *The R Journal* 3: 5–10. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.