2026-04-02

Modernising Integer Partitions: Implementing Standard Iterators for the partitions Package

Sam Kamperis
Oxford Brookes University

Robin Hankin
University of Stirling

Executive Summary

Overview

The `partitions` package (Robin K. S. Hankin 2006) is widely used R infrastructure, but its current APIs either materialise all partitions in memory or require non-standard manual iteration. This proposal will add a standard S3 iterator interface (Analytics and Weston 2022), enabling lazy evaluation, lower memory use, and direct interoperability with tools such as `foreach` and `itertools` (Weston and Wickham 2014). The work is low-risk and high-leverage because it modernises the interface layer while preserving the package's established enumeration logic.

Project Scope

Primary deliverable: an S3 `partitions_iter` class conforming to the `iterators` interface.

Secondary deliverables: documentation, worked examples, interoperability tests, and CRAN-ready release preparation.

Budget & Timeline

We request **$8,374.74** to fund **124 hours** of development over six months (July-December 2026), jointly delivered by Oxford Brookes University and the University of Stirling.

Team

The project is led by the grant applicant (9.7% FTE over 6 months) with **Robin Hankin**, the original `partitions` package author, as a collaborator providing technical guidance, code review, and CRAN liaison.

Expected Impact

This modernisation will make `partitions` usable in contemporary iterator and parallel workflows for its 1.3 million-download user base, reducing memory risk and replacing bespoke iteration code with standard, composable workflows.

Signatories

Project team

Sam Kamperis and **Robin Hankin** will deliver the project. Hankin is the original author of `partitions` and will provide technical review and release guidance.

Consulted

Feedback was sought from **James Maunder**, **Fridolin Wild**, and **Inna Skarga-Bandurova**.

The Problem

What Is the Problem?

The `partitions` package (Robin K. S. Hankin 2006) currently offers either bulk-generation functions (`parts()`, `restrictedparts()`, `diffparts()`) that materialise all results in memory, or low-level stepwise functions (`firstpart()`/`nextpart()`) that require manual state management. This is increasingly misaligned with modern R workflows. Partition counts grow explosively, so even moderate inputs can become impractical with bulk generation: for example, $p(50) = 204,226$ and $p(100)$ is approximately 190 million. At the same time, the manual stepping API does not implement the standard `iterators` interface (Analytics and Weston 2022), so users cannot use partition streams directly in `foreach`, `itertools` (Weston and Wickham 2014), or related parallel tooling without custom wrappers.

Who Is Affected?

With **1.3 million total downloads** and **6,000–8,000 monthly users**, the package affects researchers, statisticians, operations researchers, and practitioners who need scalable combinatorial workflows.

Why Is This a Problem?

1. **Ecosystem misalignment:** standard R tooling (`foreach`, `itertools`, `furrr` (Vaughan and Dancho 2022)) expects `nextElem`-style iterators.
2. **Memory constraints:** practical inputs are unsafe with full matrix materialisation.
3. **Concurrency barriers:** non-standard iteration blocks straightforward parallel and distributed use.

What Will Solving This Enable?

Implementing standard iterators will enable lazy evaluation, unlock parallel workflows, improve interoperability with iterator combinators, and let users write familiar R code instead of bespoke control loops.

Existing Work & Precedent

Standard iterator support is already established in the R ecosystem. This project brings `partitions` into that convention while preserving the existing `firstpart()`/`nextpart()` API for backward compatibility.

Competitive Landscape

Related packages such as `RcppAlgos` (Wood 2026b, 2026a) and `arrangements` (Lai 2020) provide strong combinatorial tooling, but use package-specific iteration interfaces. Our approach is complementary: we modernise the `partitions` interface layer so its unique partition semantics integrate directly with standard iterator tooling rather than through bespoke adapters.

The Proposal

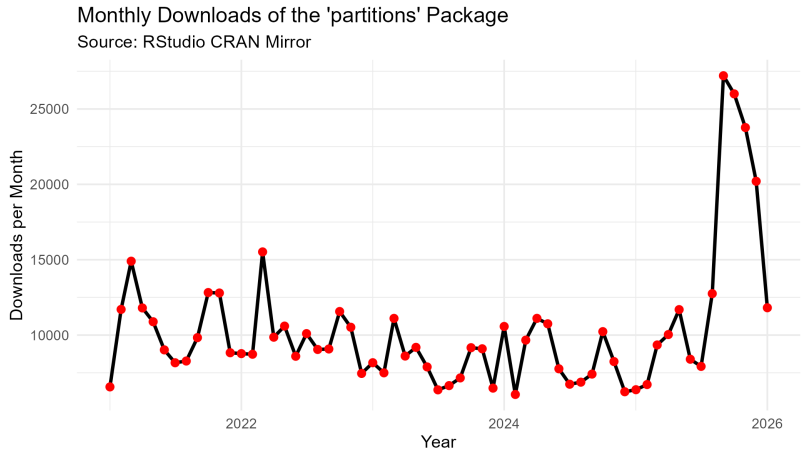
Overview

We will modernise the `partitions` package (Robin K. S. Hankin 2006) by implementing the standard S3 `iterators` protocol (Analytics and Weston 2022). The core deliverable is a new `partitions_iter` S3 class with `nextElem`, allowing partition enumerations to be used directly in `foreach()` loops, `itertools` (Weston and Wickham 2014) pipelines, and parallel workflows. The solution preserves existing APIs and wraps the current stepping logic rather than replacing it, which keeps the implementation low-risk and compatible with existing users.

Delivery

The minimum viable implementation includes a `partitions_iter` class, `nextElem.partitions_iter()`, and factory functions `iparts()`, `irestrictedparts()`, and `idiffparts()`, together with `testthat` (Wickham 2011) coverage demonstrating equivalence with bulk-generation outputs and independent iterator coexistence. We will benchmark bulk versus lazy memory behaviour and confirm interoperability with `foreach` and `itertools`. Detailed architecture, benchmark protocols, and risk controls are provided in the companion plan (Project Plan).

Evidence of Impact



With ~**1.3 million cumulative downloads** and **6,000–8,000 monthly downloads**, the package is a foundational tool in the R ecosystem. This level of usage, comparable to well-established data manipulation and statistical packages, justifies investment in modernising its infrastructure to integrate with contemporary R practices (lazy evaluation, functional programming, parallelisation).

Users across combinatorics research, statistics, operations research, and scientific computing depend on this package. Enabling standard iterator support will unlock new use cases and remove artificial barriers to productivity for researchers working with large combinatorial spaces.

Project Plan

Delivery is structured across six phases over six months (July-December 2026). The main proposal reports milestones and outcomes at a high level; full operational detail appears in Project Plan.

Phase	Duration	Key Deliverables	Effort
Start-up	July	Collaboration setup, scope finalisation, CI baseline	16 hrs
Core implementation	August	`iparts()`, `irestrictedparts()`, `idiffparts()`, `nextElem` methods, core tests	24 hrs
Integration and testing	September	`foreach` and `itertools` integration, expanded tests	20 hrs
Vignette and documentation	October	Quarto vignette, examples, README and docs updates	32 hrs

Phase	Duration	Key Deliverables	Effort
Review and release	November	Final review, issue triage, CRAN submission preparation	20 hrs
Publicity and dissemination	December	CRAN follow-up and public communication	12 hrs
Total	**July-December 2026**	**Modernised iterator interface and release**	**124 hrs**

Budget Summary

- **Total funding requested:** $8,374.74
- **Total duration:** 6 months
- **Total effort:** 124 hours (Sam Kamperis: 80; Robin Hankin: 44)

Availability and Dissemination

All outputs remain under the package's GPL-3 licence. Development and review will be conducted via the official GitHub repository with transparent issue and pull-request tracking. Dissemination includes project updates during delivery and a post-release technical communication to the R community.

Success

Definition of Done

Success is defined by five outcomes: code merged and released to CRAN; `partitions_iter` compliant with the `iterators` S3 interface; tested correctness and interoperability; user documentation with worked examples; and iterator-related issues resolved or explicitly triaged.

Measuring Success

We will track the following headline metrics throughout the project:

Metric	Success Criterion	Measurement Method
Coverage	>= 90% of new iterator code tested	`covr` report
Checks	`devtools::check()` and `R CMD check` pass cleanly	CI and CRAN dry-run
Interop	`foreach()` and `itertools` examples run without error	Tests and vignette examples
Memory	Peak memory of lazy iteration is <= 10% of equivalent bulk generation for large n	Benchmark scripts

For implementation-level architecture, benchmark protocols, detailed monthly milestones, and expanded risk management, see the standalone companion document Project Plan.

Analytics, Revolution, and Steve Weston. 2022. *Iterators: Provides Iterator Construct*. https://doi.org/10.32614/CRAN.package.iterators.

Lai, Randy. 2020. *Arrangements: Fast Generators and Iterators for Permutations, Combinations, Integer Partitions and Compositions*. https://doi.org/10.32614/CRAN.package.arrangements.

Robin K. S. Hankin. 2006. "Additive integer partitions in R." *Journal of Statistical Software, Code Snippets* 16 (May).

Vaughan, Davis, and Matt Dancho. 2022. *Furrr: Apply Mapping Functions in Parallel Using Futures*. https://doi.org/10.32614/CRAN.package.furrr.

Weston, Steve, and Hadley Wickham. 2014. *Itertools: Iterator Tools*. https://doi.org/10.32614/CRAN.package.itertools.

Wickham, Hadley. 2011. "Testthat: Get Started with Testing." *The R Journal* 3: 5–10. https://journal.r-project.org/articles/RJ-2011-002/.

Wood, Joseph. 2026a. *High Performance Benchmarks*. Vignette for R package RcppAlgos. https://cran.r-project.org/web/packages/RcppAlgos/vignettes/HighPerformanceBenchmarks.html.

Wood, Joseph. 2026b. *RcppAlgos: High Performance Tools for Combinatorics and Computational Mathematics*. https://jwood000.github.io/RcppAlgos/.