The Journal of Open Source Software

phinterval: An R package for representing and manipulating time spans with gaps

Ethan Sansom[1]

1 Department of Statistical Sciences, University of Toronto

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

`phinterval` is an R (R Core Team 2022) package for representing and manipulating time spans that may include gaps. It implements the *phinterval* vector class, designed as a generalization of the `lubridate` (Grolemund and Wickham 2011) package's *Interval* class.

While existing interval classes represent contiguous spans of time defined by a start and an end point, each element of a *phinterval* is a union of zero or more disjoint intervals. This allows empty time spans (e.g., the intersection of two non-overlapping events) and disjoint time spans (e.g., a student's periods of enrollment in school separated by breaks) to be treated as first-class objects and manipulated using well-defined set operations.

Functionality for working with these time spans includes:

- Set Operations: Vectorized union, intersection, difference, and complement.
- Merging of overlapping and adjacent intervals into minimal non-overlapping sets.
- Tests for whether time spans, dates, or times fall within one another or overlap.

The package is designed to work seamlessly with `lubridate`: all `phinterval` functions accept either *Interval* or *phinterval* vectors as input, enabling analysts to safely drop `phinterval` functions into their existing workflows.

Statement of Need

Accurately representing and manipulating dates and times is challenging due to complexities such as time zone adjustments, daylight saving transitions, and leap years. As a result, temporal data analysis is a common source of frustration and error-prone or difficult-to-maintain code (Grolemund and Wickham 2011; Tiwari et al. 2025). Several R packages provide intuitive interfaces that handle these complexities automatically. In particular, `lubridate`, which is widely used for date-time manipulation in R, and `ivs` (Vaughan 2023) simplify the representation and manipulation of time spans, reducing users' cognitive load and the likelihood of mistakes.

However, these packages typically assume that a time span is a single, contiguous block of time. This assumption breaks down when operations produce disjoint or empty spans, such as the intersection of two non-overlapping events, in which `ivs` raises an error, while `lubridate` returns missing values. While support for disjoint spans exists in the `intervals` package (Bourgon 2024), it is limited to numeric endpoints and is implemented as a scalar class, requiring users working with tabular or vectorized data to manually manage lists of scalar interval objects. As a result, analysts lack a vectorized, native date-time representation for disjoint spans and must instead rely on ad hoc workarounds that do not integrate cleanly with tabular workflows.

The `phinterval` package addresses this gap by providing a vectorized representation of disjoint and empty time spans, along with set operations whose results are always valid *phinterval* objects. By closely mirroring the `lubridate` interface and accepting *Interval* vectors as input, `phinterval` reduces adoption costs while enabling safe and intuitive manipulation of arbitrary time spans. The package is intended for analysts and researchers working with temporal data in settings such as event studies and observational data analysis.

Examples

The following examples demonstrate how `phinterval` functions can be used as drop-in replacements for `lubridate` operations, providing correct results for empty or disjoint intervals that would otherwise produce errors or ambiguous output.

```
library(phinterval)

jan_1_to_9 <- interval(as.Date("2000-01-01"), as.Date("2000-01-09"))
jan_2_to_3 <- interval(as.Date("2000-01-02"), as.Date("2000-01-03"))
jan_5_to_9 <- interval(as.Date("2000-01-05"), as.Date("2000-01-09"))
```

In `lubridate`, the intersection of non-overlapping intervals returns an object with missing endpoints, resulting in ambiguity between genuinely empty time spans and missing data. `phinterval` explicitly represents empty time spans as a `<hole>`.

```
lubridate::intersect(jan_2_to_3, jan_5_to_9)
# [1] NA--NA

phint_intersect(jan_2_to_3, jan_5_to_9)
# <phinterval<UTC>[1]>
# [1] <hole>
```

Standard interval classes cannot represent gaps within a single observation, but `phinterval` handles these naturally.

```
try(lubridate::setdiff(jan_1_to_9, jan_2_to_3))
# Error in setdiff.Interval(jan_1_to_9, jan_2_to_3) :
#   Cases 1 result in discontinuous intervals.

phint_setdiff(jan_1_to_9, jan_2_to_3)
# <phinterval<UTC>[1]>
# [1] {2000-01-01--2000-01-02, 2000-01-03--2000-01-09}
```

In addition to standard set operations, `phinterval` provides specialized functions for working with intervals with gaps. For example, `phint_squash()` flattens a vector of intervals into a minimal set of non-overlapping time spans, while `phint_invert()` returns the gaps between disjoint intervals.

These functions are useful for summarizing longitudinal event data, such as periods of employment and intervening unemployment:

```
jobs <- dplyr::tribble(
  ~name,   ~job_title,            ~start,        ~end,
  "Greg",  "Mascot",              "2018-01-01",  "2018-06-03",
  "Greg",  "Chief of Staff",      "2019-03-01",  "2020-11-28",
  "Tom",   "Chairman",            "2019-05-01",  "2020-11-10",
  "Tom",   "CEO",                 "2020-11-10",  "2020-12-31",
  "Shiv",  "Political Consultant", "2017-01-01",  "2019-04-01"
```

```
)

employment <- jobs |>
  dplyr::mutate(span = interval(start, end)) |>
  dplyr::group_by(name) |>
  dplyr::summarize(employed = phint_squash(span))

employment
# # A tibble: 3 × 2
#   name                                          employed
#   <chr>                                      <phint<UTC>>
# 1 Greg  {2018-01-01--2018-06-03, 2019-03-01--2020-11-28}
# 2 Shiv                            {2017-01-01--2019-04-01}
# 3 Tom                             {2019-05-01--2020-12-31}
```

All `phinterval` functions are vectorized, making them well suited for data analysis.

```
unemployment <- employment |>
  dplyr::mutate(
    # When were there gaps in workers' employment?
    unemployed = phint_invert(employed),

    # How long were the gaps in employment?
    n_unemp_days = unemployed / lubridate::ddays(1)
  ) |>
  dplyr::select(name, unemployed, n_unemp_days)

unemployment
# # A tibble: 3 × 3
#   name                unemployed n_unemp_days
#   <chr>            <phint<UTC>>        <dbl>
# 1 Greg  {2018-06-03--2019-03-01}          271
# 2 Shiv                  <hole>            0
# 3 Tom                   <hole>            0
```

References

Bourgon, Richard. 2024. *intervals: Tools for Working with Points and Intervals*. https://doi.org/10.32614/CRAN.package.intervals.

Grolemund, Garrett, and Hadley Wickham. 2011. "Dates and Times Made Easy with lubridate." *Journal of Statistical Software* 40 (3): 1–25. https://doi.org/10.18637/jss.v040.i03.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://doi.org/10.32614/r.manuals.

Tiwari, Shrey, Serena Chen, Alexander Joukov, Peter Vandervelde, Ao Li, and Rohan Padhye. 2025. "It's About Time: An Empirical Study of Date and Time Bugs in Open-Source Python Software." In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, 39–51. https://doi.org/10.1109/MSR66628.2025.00020.

Vaughan, Davis. 2023. *ivs: Interval Vectors*. https://doi.org/10.32614/cran.package.ivs.