Response to Reviewers - Article 4780

cubble: An R Package for Organizing and Wrangling Multivariate Spatio-temporal Data

H. Sherry Zhang, Dianne Cook, Ursula Laa, Nicolas Langrené, Patricia Menéndez

2023-06-26

We provide an overall summary of the changes made to the paper and the package and then we make a point-by-point explanation of how we addressed the reviewers' comments.

Summary of the changes

Paper

Section 2 has been reworked to combine the original Sections 2 and 3. It now focuses more on the inner workings of the package: the cubble class and its attributes, object creation and coercion, functions and methods implemented, compatibility with `tsibble` and `sf`, and comparison with other existing spatio-temporal packages (`stars` and `sftime`).

In the comparison section (Section 2.5), we compare `cubble` with `stars` and `sftime`. With `stars`, we believe spatio-temporal analysts who are more comfortable processing data with tidy tools operating on `tibble`s from tidyverse will find `cubble` useful, especially since `stars` is built from arrays. With `sftime`, we argue that `cubble` is more efficient in memory since it doesn't repeat spatial variables at each time stamp. We give an example to compare the object size of the same data stored as an `sftime` object and a `cubble` object (118MB vs. 9MB for 639 climate weather stations observed daily in 2020).

Other sections of the paper have been changed for conciseness and to better match the numerous package vignettes.

Software

The `cubble` class has been re-factored as suggested by reviewer 1. Now the `cubble` class has two sub-classes, instead of forms:

- nested/ spatial cubble: `c("spatial_cubble_df", "cubble_df")`, and
- long/ temporal cubble: `c("temporal_cubble_df", "cubble_df")`.

All the S3 methods have been modified accordingly. Methods that behave consistently in the nested and long cubble are implemented in the `cubble_df` class, or if appropriate, separately in the `spatial_cubble_df` and `temporal_cubble_df` classes.

Other substantial changes made to the package include:

- The syntax for creating a `cubble` from separate spatial and temporal components has been changed from `as_cubble(list(spatial = ..., temporal = ...), ...)` to `make_cubble(spatial = ..., temporal = ..., ...)`.
- The key checking in creating `cubble` from separate spatial and temporal components has been moved into a separate function: `check_key()`.
- The newly added toy data for demonstration: `stations` and `stations_sf` (the spatial component of three airport weather stations and it as an `sf` object), `meteo` and `meteo_ts` (the temporal component of the three weather stations and it as a `tsibble` object).

- A new version of the package (v0.3.0) is now available on CRAN.

Reviewers' comments are in black and **our responses are in red**.

Response to B-review_1_1

Paper

The paper describes a new data structure cubble and associated workflows. In a nutshell cubble subsumes two data structures:

- nested which is indexed by spatial location and keeps the time-series data in a list column long which is indexed by time and keeps the spatial data in a hidden "spatial" attribute.

The functionality of the package consists of

- data transformation functions: face_temporal, face_spatial and unfold
- data matching (merging) functions: match_sites, match_spatial and match_temporal

The package and the proposed data structures could certainly constitute a valuable base for the spatio-temporal statistics tooling. Unfortunately the paper does not clearly describe the data structure, nor the associated functionality. I had to read the package vignettes, run the attached code and even look into the package's code in order to fully understand the data structure.

We have reworked Section 2 to more completely describe the cubble class and its associated functionality. Additionally, the text in the paper has been aligned with the package vignette.

Given the small size of the package, the paper feels a bit on the heavy side. Most of the code in the paper is not self-contained which makes it rather difficult to follow. Examples at the end of the paper don't elucidate the inner workings of the package but rather advertise the functionality of other, mostly visualization, packages.

The data structure proposed in the paper is an infrastructure development to make the the analysis of spatio-temporal data simpler and to better construct visualisation. The examples illustrate how the infrastructure can be used as part of routine analyses. We added a paragraph at the beginning of Section 4 to explain this purpose.

More fundamentally, the design of the package has an immediately apparent drawback. The cubble class is used to represent two incompatible data types - long and nested. Internally, the two data types are distinguished by the internal attribute "form". In other words, the authors use an attribute of an object to mimic the functionality of a sub-class. A more natural implementation would be start with two distinct data structures, both sub-classes of the abstract cubble_df class, c("spatial_cubble_df", "cubble_df") and c("spatial_temporal_df", "cubble_df"). Such a generic implementation would allow for natural extension of the functionality through dedicated methods. Currently an ever-present check on "form" attribute is necessary throughout the code base. To make matters worse, as_cubble currently produces yet another data structure - a list of paired matches, as can be seen from example 5.1 in the paper.

Your suggestion has been adopted, see the Software section in the summary of the changes for details. The code in the example (now Section 4.1) is simplified.

Relatedly, the rationale for the parallel naming convention long/nested and temporal/spatial is not entirely clear. It seems to me that the semantically unambiguous temporal/spatial could be used throughout without ambiguity, thus resolving the terminological redundancy. **We have replaced nested/long cubble with spatial/temporal cubble.**

As cubble is a new package and no reverse dependencies yet exist, I would suggest rewriting the class dispatch mechanism before pursuing with the paper publication. My recommendation would be a combination of the following:

- Rework the definitional parts (sections 2 and 3) by following more closely the “design” vignette. **Done.**
- Describe at a glance the core functions which operate on the class and their motivation. **Done in Section 2.3.**
- Rework section 4 by making it more concise and provide links to sections in supplementary material or dedicated vignettes. **Done.**
- Rework examples section 5 into “Applications”. Briefly describe the applications and provide and refer to dedicated and self-contained vignettes. **Done.**

Software

The package is generally badly documented. The meaning of the arguments is almost never clear from the documentation alone. For example the documentation of as_cubble states:

```
  data: the object to be created or tested as cubble
   key: the spatial identifier
 index: the time identifier
coords: the coordinates that characterise the spatial dimension
   ...: a list object to create new cubble
```

- What objects are supported as input data?
- What is the assumption of the “nestedness” of the input data.
- What is the accepted data type of key, index and coords?

The user would need to visit the documentation of tsibble in order to understand the meaning of the key and index. But the relationship to tsibble is not even mentioned in the docs. The documentation of the key and index is not perfect in tsibble either, but sheds some more light on their role:

```
   key: Variable(s) that uniquely determine time indices. ‘NULL’ for
        empty key, and ‘c()’ for multiple variables. It works with
        tidy selector (e.g. ‘dplyr::starts_with()’).
 index: A variable to specify the time index variable.
```

The doc states:

```
The constructor for the cubble class
```

If it’s the constructor then the convention is to name it cubble(). as_xyz is the converter which translates between different data types. For example, if the proper sub-classing would be used, as_spatial_cubble and as_temporal_cubble would be more standard names rather than the face_spatial and face_temporal respectively.

Documentation of the package has been substantially improved with a focus on clarifying the input argument types. For the specific example of as_cubble(), the documentation now states:

- **data: an object to be converted into a cubble object. Currently support objects of classes tibble, ncdf4, stars, and sftime.**
- **key: a character (or symbol), the spatial identifier.**
- **index: a character (or symbol), the temporal identifier. Currently support base R classes Date, POSIXlt, POSIXct, and tsibble’s yearmonth, yearweek, and yearquarter class.**
- **coords: a vector of character (or symbol) of length two, in the order of longitude first and then latitude, the argument can be omitted if created from an sf and its subclasses. In case the sf geometry column is not POINT, coords will be the centroid coordinates.**

Further suggestions

Consider not throwing an error when face_spatial is applied to the spatial object. This would allow for generic code where the "form" of the input the data is not known or does not matter.

Done. Now when applying face_spatial to a spatial cubble, the cubble will be printed as it is with a message signalling it is already in the nested form. See the reprex below. Same for face_temporal on a temporal cubble object.

```
library(cubble)
class(climate_mel)
#> [1] "spatial_cubble_df" "cubble_df"         "tbl_df"
#> [4] "tbl"               "data.frame"
invisible(face_spatial(climate_mel))
#> The cubble is already in the nested form
```

Created on 2023-06-18 with reprex v2.0.2

It would be useful to print a few rows of the spatial attribute of the "long" object. **When is is subclass spatial_cubble_df these rows are printed, but it could be misleading to do so if it is a subclass temporal_cubble_df. Users can extract the rows of the spatial variables with the spatial() function, as explained in Section 2.1.**

Section 3.6 is missing `tidyr` package which is especially relevant here given its nesting and unnesting operations https://tidyr.tidyverse.org/articles/nest.html **The idea of tidy data is underlies the cubble class (as explained in Section 2.1) but tidyr functions are not applicable to a cubble. This is the reason that tidyr was not included in the original section 3.6. The section has been replaced by Tables 1 and 2.**

More comments inlined in the pdf. **Done with the Section restructuring.**

Response to B-review_2_1

Paper

The manuscript is in general clearly structured. However, an important reference to the R-package sftime is missing that also deals with the representation of spatiotemporal data. It might be beneficial to provide illustrative examples in the manuscript that clearly compare between a cubble representation and representations of existing packages. Which data sets/structures cannot (or with a greater effort) be represented with existing R-packages such as e.g. stars and sftime?

A new section, Section 2.5, has been dedicated to comparing cubble with existing spatio-temporal classes, specifically, stars and sftime. See the Paper section in the summary of the changes.

For spatial data, the coordinate reference system is essential metadata information. In one example of the manuscript, the input data does have information on the CRS but it is not discussed in the manuscript how that is handled within cubble. How would coordinate transformations be handled with cubble?

The coordinate reference system is handled, under the hood, by the sf::st_tranform(). In the case where a cubble object is created from an sf object, the sf class will be retained. When it is not, users can promote the cubble object to include the sf class with make_spatial_sf()

The use-case of temporal matching based on features of the time series appears an interesting, but also a very specific one. How could this be generalized to a more generic approach?

In spatial matching, several distance calculations are now made available under sf::st_distance. Temporal matching now accepts an argument temporal_match_fn to allow user-specified functions for matching.

Additionally, several typos and language issues arise and limit the readability of the manuscript. Some examples are:

- doubled/missing words/none correct sentences:
  - "*components* spatio-temporal *components*" **Fixed.**
  - "... fits works ..." **Fixed.**
  - "... be activate rows ..." **Fixed.**
  - "... highlighted *in* the ..." **Fixed.**
  - ". . . using 2020 measurements using match_sites() function." **Fixed.**
  - "An example of this using is included in the Appendix" **Fixed.**
  - ". . . the data in -a- multiple -of- ways on-the-fly" **Fixed.**
- Surprising references: in Section 4.4 it says ". . . Glyph maps (Section 3.4)", but Section 3.4 says ". . . glyph maps will be explained in Section 4.4". **Fixed.**
- Should ". . . it's temperatures are more *consistent*" rather be ". . . it's temperatures are more *constant*"? **Changed.**
- Typos:
  - ". . . spatial and *tmeporal* information are available." **Fixed.**
  - The polar vortex, signalled by the high *speicfic* humidity, splits into two on 2002-09-26 and *further-s- split_s_* into four on 2002-10-04. **Fixed.**
  - the data in *-a- multiple -of-* ways on-the-fly. **Fixed.**

Software

I have been a bit puzzled by the "print" of a nested cubble in its temporal face. The row # temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl] appears at first sight (also in comparison with the print of the spatial face), as if the temporal domain is given by all those variables. To me, a notion such as

```
# temporal: date [date]
```

```
# variables: prcp [dbl], tmax [dbl], tmin [dbl]
```

would have been more intuitive. Possibly also including the temporal range (as for the spatial domain its bbox).

For temporal cubble, the header now prints the temporal meta data: start date, end date, interval, and whether there are gaps in the data, e.g. "temporal: 2020-01-01 – 2020-01-10 [1D], no gaps"

In the manuscript and in the reproducible-script.R, a data set `historical_tmax` is introduced. In the package, the corresponding data set seems to be `tmax_hist`.

The data historical_tmax used in the paper now replaces the tmax_hist data in the package.

The manuscript cannot be reproduced, as the reproducible-script.R (part of the submission to JSS) only uses a subset, but does not provide a separate result file that would allow to compare the reproduced outputs with the desired output for the subset. **A result file in HTML is submitted along with the reproducible-script.R file.**