Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

cubble: An R Package for Structuring Spatio-temporal Data

H. Sherry Zhang
Monash University

Dianne Cook
Monash University

Ursula Laa
University of Natural Resources and Life Sciences

Nicolas Langrené
CSIRO Data61

Patricia Menéndez
Monash University

Abstract

The abstract of the article.

Keywords: spatio-temporal data, R.

1. Introduction

Spatio-temporal data + examples

Spatio-temporal data record changes of variables in spatially separated regions across time. In this article, we consider spatio-temporal vector data, which are recorded in a fixed interval. Examples of this type of data include the house price of a city or county, climate measures from weather stations in a country, and river level data from electronic gauges more examples here.

point to common feature: spatial level variables + time level variables relates to Table 13 in Tidy data paper

Usually, this type of spatio-temporal data don't come in as a single table. Tidy data principle (. . .) prescribes each type of observational unit to form a table. This would suggest two tables to store the data, one for spatial-level and one for temporal-level. The Table 13 in the tidy data paper present a structure like this and the author argues the lack of tools to work with relational data.

My proposal

Recent software development in R has proposed several relational data structure: `tidygraph`(Pedersen 2020) for graph manipulation, `dm` (Schieferdecker, Müller, and Bergant 2021) for relational data model, while spatio-temporal data could benefit from having its own relational data structure. In this paper, we propose a tidy data structure for vector spaio-temporal data.

Section division

The rest of the paper will be divided as follows: Section 2 reviews the existing data structure for spatio, temporal, and spatio-temporal data. Section 3 presents a new data structure for spatio-temporal data: cubble. Then the paper introduces the workflow of data manipulation and visualisation with the cubble structure in Section 4. Section 5 gives some examples on how common spatial and temporal manipulations are performed with cubble and how static and interactive visualisation help to understand climate and [. . .] data.

2. Existing data structure for spatio and temporal data

Existing packages

Many data structures have been proposed for spatial and temporal data, but not many for spatio-temporal data. One of the reason could be the inherent different levels of information make it inefficient to store in the same table. `spacetime`(Pebesma 2012) proposed four space-time layouts: Full grid (STF), sparse grid(STS), irregular (STI), and trajectory (STT) based on underlying spatial structure `sp` (Pebesma and Bivand 2005) and temporal structure `xts`(Ryan and Ulrich 2020). `spatstat` (Baddeley and Turner 2005) implements a `ppp` class for point pattern data. More recent package `stars` (Pebesma 2021) uses a spatiotemporal array to store the data and the array structure has its influence from `cubelyr` (Wickham 2020), a dplyr data cube backend.

Criteria of a data structure we want

With recent development in the R community, `sf` (Pebesma 2018) and `tsibble`(Wang, Cook, and Hyndman 2020) have replaced `sp` and `xts` to be the convention structure for spatial and temporal data. One reason for their popularity is its integration with tidyverse ecosystem, making them intuitive and easy to adopt. For spatio-temporal data, we hope to build a data structure has the following features:

1) A data structure that handles spatial and temporal dimension in a relational structure. This derives from the 3rd tidy data principal.
2) An intuitive and easy to use interface that fits into the tidyverse ecosystem, and
3) Compatibility with the latest spatial and temporal data structure. This would give users the flexibility to use work from existing packages.

3. Workflow/ Data pipeline

Now we introduce cubble in a data pipeline structure (review the concept of data pipeline? at least articulate its importance - automation, predictable of output)

3.1. Spatio-temporal data in the wild

Format of st in the wild
Analysing this type of data requires less considerations on the geographical geometry type and map projection but more on how measures in these fixed locations changes across the time domain and whether these changes are related for adjacent locations. For example, when nearby areas show patterns that are regular enough, visualising spatio-temporal data can 1) discover regional time series features, i.e. trend and seasonality, 2) find the Waldo sites from the crowd, and 3) see how correlation of nearby sites changes across time.

The main difficulty and challenge
The main difficulty in visualising this type of data is to show information in both space and time dimension with the proper level of details without information overflow. This would sometimes require aggregating the time dimension into the proper level or slicing the data into a reasonable number of subset for display. In this sense, a data structure that regulates the manipulation spatio-temporal data will benefit the analysis workflow. While many implementations focus on manipulating and visualising pure spatial or temporal data, there are not sufficient tools to deal with spatio-temporal data. The purpose of this paper is to introduce a spatio-temporal vector data structure for data analysis in R.

3.2. Cubble

We form a cubble by defining abc... - Nested and long form
When manipulating the spatial dimension it uses a nest form that:

- defines each group in a row,
- displays the group-related variables in columns, and
- nests all the time-related variables into a column called `ts`.

When manipulating the temporal dimension, it uses the long form that:

- each combination of group and timestamp occupies a row
- time-related variables are displayed, and
- group-related variables are not explicitly displayed but can be accessed through the `meta` attribute.

stretch

By default build a nested cubble, stretch to long, create spatial attributes

tamp

Tamp to nested, use the spatial attributes

Support on hierarchical structure

3.3. Tidyverse compatibility

list supported tidyverse functions

3.4. Others

4. Examples

4.1. Australia precipitation pattern in 2020

Forming a cubble + basic tidyverse verbs - Vig 2 Aggregation - Vig 4

4.2. Matching precipitation and river level in Victria water gauges

Matching - Vig 3

5. Conclusion

6. Old stuff

Many spatial and spatio-temporal data structures have been developed by the R-spatial team for both raster and vector spatial data. For vector spatial data, which is the focus of this paper, `sf` (?) represents spatial vector information with simple features: points, lines, polygons and their multiples. Various `st_` function are designed to manipulate these features based on their geometric relationships. For spatio-temporal data, `stars` (Pebesma 2021) can represent both raster and vector data using multi-dimensional array. However, the underlying array structure can be difficult to operate for data analysts who are more familiar with a flat 2D data frame structure used by the tidyverse ecosystem.

In the temporal aspect, the `tsibble` (?) structure and its tidyverts ecosystem have provided a [...] workflow to work with temporal data. In a tsibble structure, temporal data is characterised by `index` and `key` where `index` is the temporal identifier and `key` is the identifier for multiple series, which could be used as a spatio identifier. However, a tsibble object, by construction, always requires the `index` in its structure. This makes it less appealing for spatio-temporal data since the output of calculated spatio-specific variables (i.e. features of each series) don't have the time dimension. Analysts will either need to have an additional step to join this output to the original tsibble or operate with variables stored in two separate objects. In addition, the long form structure of a tsibble object means spatio variables (i.e. longitude, latitude, and features of each series if joined back to the tsibble) of each spatio identifier will be repetitively recorded at each timestamp. This repetition is unnecessary and would inflate the object size for long series.

7. A new data structure for spatio-temporal data

Spatio-temporal data don't usually come to the analysts as a whole piece. A way to look at these data is to divide it into spatial and temporal dimension with an ID that links between the two. The first row in Figure 1 illustrates this representation where in the spatial dimension, the data is characterised by `id`, `lat`, `long`. V_s in the last column represents all the other site-wise variables, for example, elevation and full name etc. The temporal dimension, on the other hand, can be characterised by `id` and `t` with V_t representing all the time-wise variables. In climate data, this could include precipitation, maximum or minimum temperature, and wind speed etc.

To work with spatio-temporal data, analysts can choose to either work separately on each dimension or join the two sets together, however, each approach has its own problem: While is is natural to work separately on each sheet (since spatial and temporal operations usually don't overlap), analysts will need to manually keep the other data frame up to date. For example, the following pseudo code illustrates the scenario where once the spatial dataset is filtered for those within Victoria, the temporal dataset needs to be manually updated to reflect this spatial filter.

```
R> spatial_new <- spatial %>% filter(SITES_IN_VICTORIA)
R> temporal_new <- temporal %>% filter(id %in% spatial_new$id)
```

If analysts choose to join the spatial and temporal data together, the joined dataset could be too large since each spatial variable will be repeated at each time stamp for each site. Also,

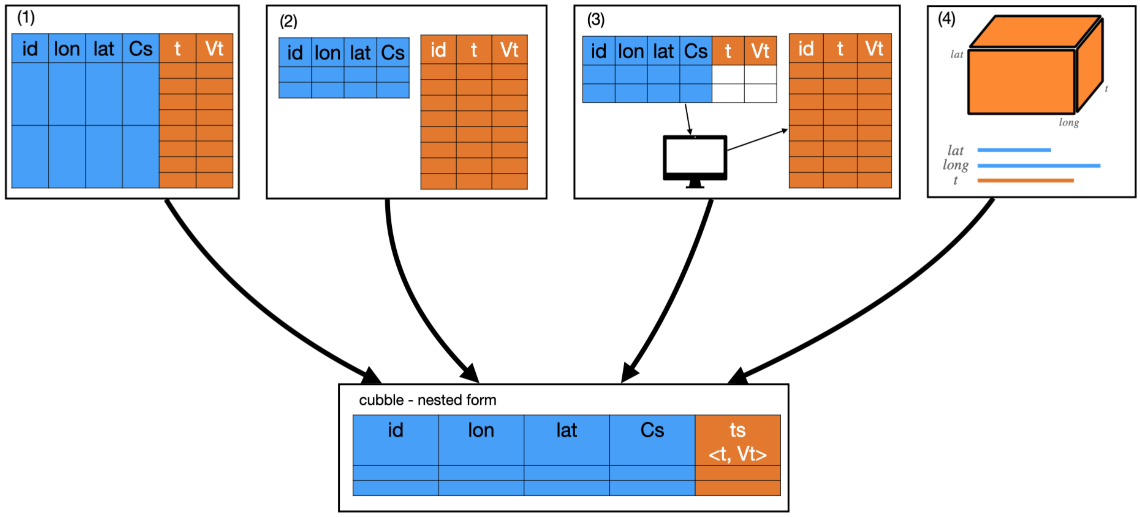

Figure 1: Cubble diagram

recordings of the site ID from different data sources can be slightly different from each other, causing a painful checking and cleaning of site IDs before the join.

A cubble, in essence, wires both dimensions in the spatio-temporal data into one object while provide two forms for manipulation the spatial and temporal dimension separately.

When manipulating the spatial dimension it uses a nest form that:

- defines each group in a row,
- displays the group-related variables in columns, and
- nests all the time-related variables into a column called `ts`.

When manipulating the temporal dimension, it uses the long form that:

- each combination of group and timestamp occupies a row
- time-related variables are displayed, and
- group-related variables are not explicitly displayed but can be accessed through the `meta` attribute.

8. Create a cubble

The creation of a cubble requires the site identifier (`key`), as well as the spatial (`coords`) and temporal (`index`) identifier. `climate_flat` is already a tibble and it uses `id` to identify each station, `date` as the time identifier, and `c(long, lat)` as the spatial identifier. To create a cubble for this data, use:

```
R> climate_flat %>% as_cubble(key = id, index = date, coords = c(long, lat))
```

```
# cubble:   id [5]: nested form
# bbox:     [115.97, -32.94, 133.55, -12.42]- check gap on long and lat
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id            lat  long  elev name           wmo_id ts
  <chr>       <dbl> <dbl> <dbl> <chr>           <dbl> <list>
1 ASN00009021 -31.9  116.  15.4 perth airport   94610 <tibble [366 x 4]>
2 ASN00010311 -31.9  117. 179   york            94623 <tibble [366 x 4]>
3 ASN00010614 -32.9  117. 338   narrogin        94627 <tibble [366 x 4]>
4 ASN00014015 -12.4  131.  30.4 darwin airport  94120 <tibble [366 x 4]>
5 ASN00015131 -17.6  134. 220   elliott         94236 <tibble [366 x 4]>
```

Most of the time, spatio-temporal data doesn't come into this form and analysts need to query the climate variables based on station metadata. This is also a problem illustrated in Section 3.5 in @tidydata. Here we provide a structured way to query this data based on the row-wise operator and nested list. For this type of task, one can structure a metadata into a tibble and use row-wise operator to query the climate variables into a nested list. As an example here we demonstrate the workflow to find the 5 closest stations to Melbourne. We first create a station data frame with the 5 target stations.

```
# A tibble: 5 x 8
  id            lat  long  elev name                 wmo_id  dist city
  <chr>       <dbl> <dbl> <dbl> <chr>                 <dbl> <dbl> <chr>
1 ASN00086038 -37.7  145.  78.4 essendon airport      95866  10.8 melbourne
2 ASN00086282 -37.7  145. 113.  melbourne airport     94866  20.1 melbourne
3 ASN00086077 -38.0  145.  12.1 moorabbin airport     94870  21.9 melbourne
4 ASN00088162 -37.4  145. 528.  wallan (kilmore gap)  94860  48.1 melbourne
5 ASN00087113 -38.0  144.  10.6 avalon airport        94854  48.8 melbourne
```

We can query the climate information into a nested list named `ts` for each station with the `rowwise()` operator. To create a cubble, supply the same identifiers as with the first example.

```
R> sydmel_climate <- stations %>%
+   rowwise() %>%
+   mutate(ts = list(meteo_pull_monitors(id,
+                                         date_min = "2020-01-01",
+                                         date_max = "2020-12-31",
+                                         var = c("PRCP", "TMAX", "TMIN")) %>%
+                      select(-id))) %>%
+   as_cubble(key = id, index = date, coords = c(long, lat))
```

```
# cubble:   id [5]: nested form
# bbox:     [144.47, -38.03, 145.1, -37.38]
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id            lat  long  elev name                   wmo_id  dist city   ts
  <chr>       <dbl> <dbl> <dbl> <chr>                   <dbl> <dbl> <chr>  <list>
1 ASN00086038 -37.7  145.  78.4 essendon airport        95866  10.8 melbo~ <tibbl~
2 ASN00086282 -37.7  145. 113.  melbourne airport       94866  20.1 melbo~ <tibbl~
3 ASN00086077 -38.0  145.  12.1 moorabbin airport       94870  21.9 melbo~ <tibbl~
4 ASN00088162 -37.4  145. 528.  wallan (kilmore gap)  94860  48.1 melbo~ <tibbl~
5 ASN00087113 -38.0  144.  10.6 avalon airport          94854  48.8 melbo~ <tibbl~
```

Below are the how the nested and long form look like for Australia climate data, which records daily precipitation, maximum and minimum temperature for 55 stations across Australia from 2015- 2020. Notice that each station forms a group in both forms and specifically, the nested and long form have a underlying `rowwise_df` and `grouped_df` respectively.

With a cubic framework on mind, different types of manipulation with cubble can be thought of as slicing the cube in various way. The table below shows how some `dplyr` verbs are mapped into the operation in a cubble. With the existing grouping on the station, additional groupping can be added with `group_by` and removed with `ungrouped`. [talk about why it is useful]

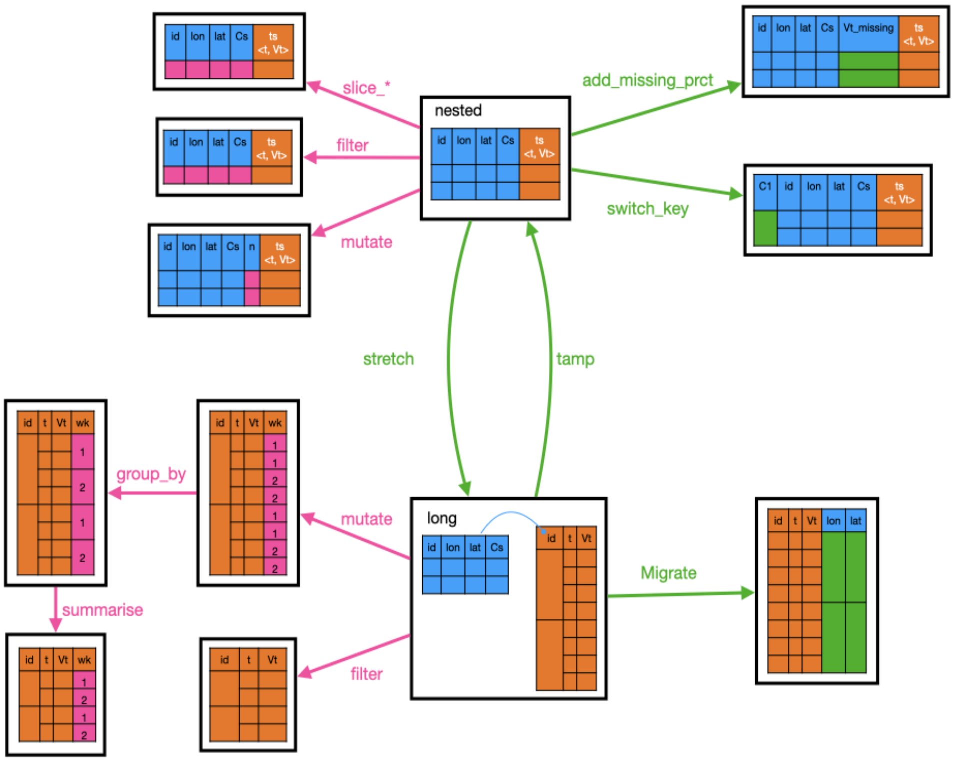

Figure 2: Cubble operations

8.1. Cubble operations

Basics

- `stretch`: nest to long form
- `tamp`: long to nest form
- `migrate`: move selected spatial variables to the long form.
- `add_dscrb_prct`: summary stats for missingness

dplyr compatibility:

- mutate, filter, summarise, select, arrange
- group and ungroup: group_by, ungroup
- slice family

Combine two cubbles

- match river and weather gauges data

- involve combining two cubbles
- join operations combine the two together by appending more rows but what we really want is to bind rows.
- bind rows also doesn't work since we want to bind only when there' s a matching????
- introduce bind_join

Hierarchical structure in cubble

- hierarchical is common.
- Given examples.
- Essence: switch between different levels
- introduce `switch_key`

9. Examples

Daily climate data (prcp, tmax, and tmin) from RNOAA - lots of stations across Australia

An exploratory data analysis questions: What's the climate profile look like in Australia

- General features: Any general trend/ fluctuation in prcp, tmax, and tmin?
- Local features: Any station stands out from the crowd?

References

Baddeley A, Turner R (2005). "Spatstat: An R Package for Analyzing Spatial Point Patterns." *Journal of Statistical Software*, **12**(6), 1–42. URL `https://doi.org/10.18637/jss.v012.i06`.

Pebesma E (2012). "spacetime: Spatio-Temporal Data in R." *Journal of Statistical Software*, **51**(7), 1–30. URL `https://doi.org/10.18637/jss.v051.i07`.

Pebesma E (2021). *stars: Spatiotemporal Arrays, Raster and Vector Data Cubes.* R package version 0.5-2, URL `https://CRAN.R-project.org/package=stars`.

Pebesma E, Bivand RS (2005). "S classes and methods for spatial data: the sp package." *R news*, **5**(2), 9–13.

Pebesma EJ (2018). "Simple features for R: standardized support for spatial vector data." *R Journal*, **10**(1), 439.

Pedersen TL (2020). *tidygraph: A Tidy API for Graph Manipulation.* R package version 1.2.0, URL `https://CRAN.R-project.org/package=tidygraph`.

Ryan JA, Ulrich JM (2020). *xts: eXtensible Time Series.* R package version 0.12.1, URL `https://CRAN.R-project.org/package=xts`.

Schieferdecker T, Müller K, Bergant D (2021). *dm: Relational Data Models*. R package version 0.2.5, URL https://CRAN.R-project.org/package=dm.

Wang E, Cook D, Hyndman RJ (2020). "A new tidy data structure to support exploration and modeling of temporal data." *Journal of Computational and Graphical Statistics*, **29**(3), 466–478. doi:10.1080/10618600.2019.1695624. URL https://doi.org/10.1080/10618600.2019.1695624.

Wickham H (2020). *cubelyr: A Data Cube 'dplyr' Backend*. R package version 1.0.1, URL https://CRAN.R-project.org/package=cubelyr.

Affiliation:

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd