A template for the *arXiv* style

A Preprint

H.Sherry Zhang
Department of Econometrics and Business Statistics
Monash University
Melbourne, Australia
huize.zhang@monash.edu

Dianne Cook
Department of Econometrics and Business Statistics
Monash University
Melbourne, Australia
dicook@monash.edu

Ursula Laa
Institute of Statistics
University of Natural Resources and Life Sciences
Vienna, Austria
ursula.laa@boku.ac.at

Nicolas Langrené
34 Village Street, Docklands VIC 3008 Australia
CSIRO Data61
Melbourne, Australia
nicolas.langrene@csiro.au

Patricia Menéndez
Department of Econometrics and Business Statistics
Monash University
Melbourne, Australia
patricia.menendez@monash.edu

August 18, 2021

Abstract

Enter the text of your abstract here.

***K*eywords** blah · blee · bloo · these are optional and can be removed

1 Introduction

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 and are point based, characterised by longitude and latitude, in the spatial aspect. 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.

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 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.

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

Below we review some structure for spatial, temporal, and spatio-temporal data.

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` (E. J. Pebesma 2018) 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` (E. 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` (Wang, Cook, and Hyndman 2020) 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.

3 A new data structure for spatio-temporal data

Spatio-temporal data can be thought of as characterised by three dimensions: group, time, and variable, which allows us to use a cube to represent the data. While group and time are can be seen as identifiers of the record, variables can be further divided into those that are group-related and time-related. Group-related variables are invariant to the time and have only one value per group while time-related variables change in time and requires both group and time to identify a record. Hence the cubic representation can be further simplified for those group-related variables as in the sketch 1, where group-related variables are squashed in the time dimension.

A cubble simplifies the workflow in spatio-temporal data through managing group and time-related variables separately in two forms: list-column and long form. The nested form:

- 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`.

This form focuses on the time-invariant variable by squashing the time dimension

In the long form,

- 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

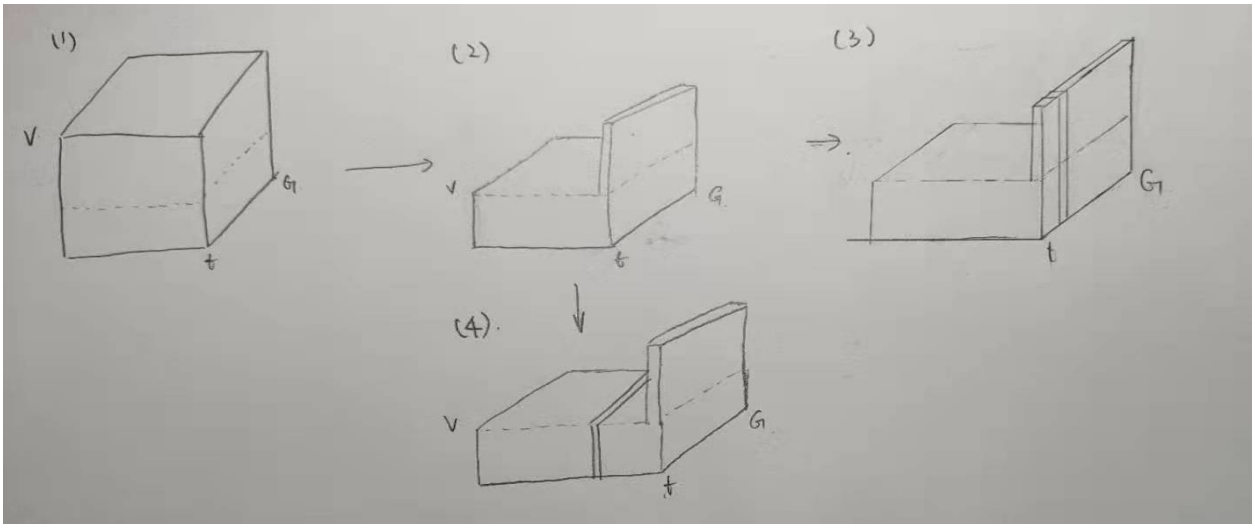

Figure 1: Cubic representation of spatio-temporal data. (1) shows the full cube, (2) simplifies teh cube by squashing the time dimension of time-invariant variables. With (2), a slice along the group dimension will affect both time-variant and invariant variables as in (3) while, a slice along the time will only affect the tim-variant variables as in (4).

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.

```
## # Cubble: station-wise: nested form
## # Group:  station [55]
##    station       lat  long elevation name                ts
##    <fct>       <dbl> <dbl>     <dbl> <fct>               <list>
##  1 ASN00001019 -14.3  127.      23   kalumburu           <tbl_ts [2,147 x 4]>
##  2 ASN00002012 -18.2  128.     422   halls creek airport <tbl_ts [2,191 x 4]>
##  3 ASN00003003 -17.9  122.       7.4 broome airport      <tbl_ts [2,192 x 4]>
##  4 ASN00006011 -24.9  114.       4   carnarvon airport   <tbl_ts [2,192 x 4]>
##  5 ASN00009021 -31.9  116.      15.4 perth airport       <tbl_ts [2,192 x 4]>
##  6 ASN00009193 -32.0  116.      43.1 rottnest island     <tbl_ts [2,192 x 4]>
##  7 ASN00009518 -34.4  115.      13   cape leeuwin        <tbl_ts [2,184 x 4]>
##  8 ASN00009789 -33.8  122.      25   esperance           <tbl_ts [2,187 x 4]>
##  9 ASN00010286 -31.6  117.     217.  cunderdin airfield  <tbl_ts [2,190 x 4]>
## 10 ASN00010917 -32.7  117.     275   wandering           <tbl_ts [2,192 x 4]>
## # ... with 45 more rows
```

```
## # Cubble: time-wise: long form [tsibble]
## # Group:  station [55]
##    station     date        prcp  tmax  tmin
##    <fct>       <date>     <dbl> <dbl> <dbl>
##  1 ASN00001019 2015-01-01   164  31.5  25
##  2 ASN00001019 2015-01-02   124  31.3  25
##  3 ASN00001019 2015-01-03    50  29.2  24.7
##  4 ASN00001019 2015-01-04   204  32.1  24.3
##  5 ASN00001019 2015-01-05   412  31.4  24.6
##  6 ASN00001019 2015-01-06   200  28.7  25.1
##  7 ASN00001019 2015-01-07   822  30.2  24.5
##  8 ASN00001019 2015-01-08    22  31    26.6
##  9 ASN00001019 2015-01-09    62  32.7  26.2
## 10 ASN00001019 2015-01-10   274  32.9  22.6
```

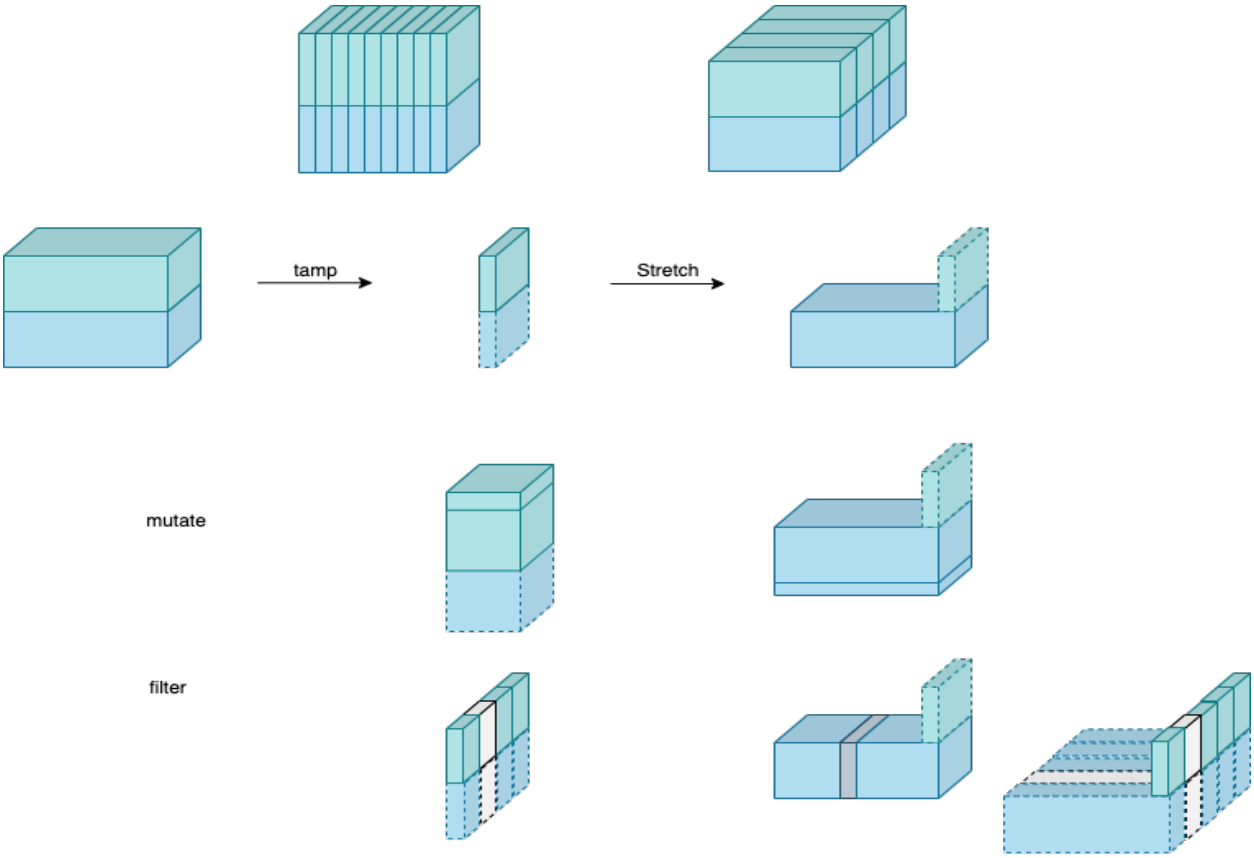

Figure 2: sdfsdf

```
## # ... with 120,311 more rows
```

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]

3.1 Cubble verbs

Mention different types of manipulation with cubble:

- `dplyr` support for cubble:
  - basic 5s: mutate, filter, summarise, select, arrange
  - group and ungroup: group_by, ungroup
  - slice family
- summarise missing stats

4 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?

4.1 Manipulation

4.1.1 Mutate and filter

In the first example, we want to only keep the stations that have `tmax` recorded in 2020. This requires first narrow down the records to those in 2020, determine if `tmax` is missing for each station, and then retain those stations that have `tmax` recorded. The year filtering is an operation on the time axis, so we start with the long form. Whether each station has `tmax` recorded is a result of each station, rather than of each time point, hence we need to switch to the nested form with `tamp()`. To calculate whether `tmax` is recorded, we mutate a column `tmax_missing` that takes `TRUE` if all the `tmax` in the nested list column `ts` are `NA` and `FALSE` otherwise. To get the stations that we want, we need another filter on `tmax_missing`.

```
climate_small %>%
  stretch() %>%
  filter(year(date) == 2020) %>%
  tamp() %>%
  mutate(tmax_missing = ifelse(all(is.na(ts$tmax)), TRUE, FALSE)) %>%
  filter(!tmax_missing)
```

```
## # Cubble: station-wise: nested form
## # Group:  station [55]
##    station      lat  long elevation name             ts             tmax_missing
##    <fct>      <dbl> <dbl>     <dbl> <fct>            <list>         <lgl>
##  1 ASN000010~ -14.3  127.      23   kalumburu        <tbl_ts [366 ~ FALSE
##  2 ASN000020~ -18.2  128.     422   halls creek air~ <tbl_ts [366 ~ FALSE
##  3 ASN000030~ -17.9  122.       7.4 broome airport   <tbl_ts [366 ~ FALSE
##  4 ASN000060~ -24.9  114.       4   carnarvon airpo~ <tbl_ts [366 ~ FALSE
##  5 ASN000090~ -31.9  116.      15.4 perth airport    <tbl_ts [366 ~ FALSE
##  6 ASN000091~ -32.0  116.      43.1 rottnest island  <tbl_ts [366 ~ FALSE
##  7 ASN000095~ -34.4  115.      13   cape leeuwin     <tbl_ts [366 ~ FALSE
##  8 ASN000097~ -33.8  122.      25   esperance        <tbl_ts [361 ~ FALSE
##  9 ASN000102~ -31.6  117.     217.  cunderdin airfi~ <tbl_ts [366 ~ FALSE
## 10 ASN000109~ -32.7  117.     275   wandering        <tbl_ts [366 ~ FALSE
## # ... with 45 more rows
```

4.1.2 Join

Now we want to select the stations that have been registered with world meteorological organisation (WMO) and the dataset `station` has a column `wmo_id` that records this information. To do this task, we first need to join the `station` dataset with our climate dataset and then filter out those stations that don't have the WMO id. Since the join is by station rather than by time, we start with the nested form and write the exact same syntax of join and filter as with tidyverse.

```
# join wmo_id for each station
to_join <- station %>% select(id, wmo_id)
out <- climate_small %>%
  left_join(to_join, by = c("station" = "id")) %>%
  filter(!is.na(wmo_id))
out
```

```
## # Cubble: station-wise: nested form
```

```
## # Group:  station [51]
##    station       lat  long elevation name              ts                 wmo_id
##    <fct>       <dbl> <dbl>     <dbl> <fct>             <list>              <dbl>
##  1 ASN00001019 -14.3  127.      23   kalumburu         <tbl_ts [2,147 x ~  94100
##  2 ASN00002012 -18.2  128.     422   halls creek airp~ <tbl_ts [2,191 x ~  94212
##  3 ASN00003003 -17.9  122.       7.4 broome airport    <tbl_ts [2,192 x ~  94203
##  4 ASN00006011 -24.9  114.       4   carnarvon airport <tbl_ts [2,192 x ~  94300
##  5 ASN00009021 -31.9  116.      15.4 perth airport     <tbl_ts [2,192 x ~  94610
##  6 ASN00009193 -32.0  116.      43.1 rottnest island   <tbl_ts [2,192 x ~  94602
##  7 ASN00009518 -34.4  115.      13   cape leeuwin      <tbl_ts [2,184 x ~  94601
##  8 ASN00009789 -33.8  122.      25   esperance         <tbl_ts [2,187 x ~  94638
##  9 ASN00010286 -31.6  117.     217.  cunderdin airfie~ <tbl_ts [2,190 x ~  95625
## 10 ASN00010917 -32.7  117.     275   wandering         <tbl_ts [2,192 x ~  95640
## # ... with 41 more rows
```

Sometimes, we would like to have station-wise and time-wise variables in the same form (i.e. when plotting glyph maps). This can also be seen as a joining task, on the long form, with the dataset to join being the metadata. `migrate()` is a verb introduced as the shortcut for `left_join()` with a cubble's metadata and below is the comparison of the two syntaxes.

```
out %>%
  stretch() %>%
  migrate(station, lat, long)
```

```
## # Cubble: time-wise: long form
## # Group:  station [55]
##    station     date        prcp  tmax  tmin   lat  long
##    <fct>       <date>     <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 ASN00001019 2015-01-01   164  31.5  25   -14.3  127.
##  2 ASN00001019 2015-01-02   124  31.3  25   -14.3  127.
##  3 ASN00001019 2015-01-03    50  29.2  24.7 -14.3  127.
##  4 ASN00001019 2015-01-04   204  32.1  24.3 -14.3  127.
##  5 ASN00001019 2015-01-05   412  31.4  24.6 -14.3  127.
##  6 ASN00001019 2015-01-06   200  28.7  25.1 -14.3  127.
##  7 ASN00001019 2015-01-07   822  30.2  24.5 -14.3  127.
##  8 ASN00001019 2015-01-08    22  31    26.6 -14.3  127.
##  9 ASN00001019 2015-01-09    62  32.7  26.2 -14.3  127.
## 10 ASN00001019 2015-01-10   274  32.9  22.6 -14.3  127.
## # ... with 111,562 more rows
```

- data quality check: filter out stations have variables not properly recorded
- data summary:
  - daily -> monthly/ weekly,
  - summarise by mean for tmax/ tmin, sum for prcp
-

4.2 Graphics

Static + interactive -> tooltip to show additional information upon hovering

- Where are those stations on the map?
  - Mention mostly aero, airport, and lighthouse

Summary

Pebesma, Edzer. 2021. *Stars: Spatiotemporal Arrays, Raster and Vector Data Cubes.* `https://CRAN.R-project.org/package=stars`.

Pebesma, Edzer J. 2018. "Simple Features for r: Standardized Support for Spatial Vector Data." *R J.* 10 (1): 439.

Wang, Earo, Dianne Cook, and Rob J Hyndman. 2020. "A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data." *Journal of Computational and Graphical Statistics* 29 (3): 466–78. https://doi.org/10.1080/10618600.2019.1695624.