Earth Syst. Dynam. Discuss.,
https://doi.org/10.5194/esd-2019-62-AC3, 2019

Earth System Dynamics Discussions

Interactive comment on "Earth system data cubes unravel global multivariate dynamics" *by* Miguel D. Mahecha et al.

Miguel D. Mahecha et al.

mmahecha@bgc-jena.mpg.de

Received and published: 20 December 2019

Comments by Appel and Pebesma are pasted here in bold font; *our answers are given in italics.*

We congratulate the authors of this paper with a very nice contribution describing the ESDC and its application, an activity that has involved a substantial conceptual development as well as implementation work, that has triggered a number of scientific papers already, and that now helps furthering the discussion what spatiotemporal data cubes are. We are mostly interested in having this latter discussion. *We are very pleased about this contribution to the discussion from two leading experts in the field of data cubes. The points raised here will carefully be*

considered in the revision of the manuscript wherever possible.

1 Pre-grid all the data, or do this on-the-fly?

The ESDC pre-grids all the data, and (l456) "One of the most commonly expressed practical concerns is the choice of a unique data grid". Given that an ESDC is defined on a single grid, but integrates a lot of different datasets, each of these datasets have to be forged into the target grid. This involves resampling, statistical downscaling, interpolation and/or aggregation of data both in space and time, as well as handling coordinate reference systems and possibly different calendars (Gregorian, 365day/noleap, 360 day) into one. It is unclear whether the ESDC can do all this, and also if it can give an idea about the errors introduced by doing so, or whether it can give users otherwise recommendations what a good grid is? Other systems, e.g. Google Earth Engine or Sentinel Hub work from the raw data (which can be in many different coordinate reference systems, e.g. Sentinel-2 distributed over 120 UTM zones), and create user-defined data cubes on the fly. This has the advantage that low-resolution computations can be done relatively fast and interactively, which helps data exploration and model development, and that the effect of different target resolutions can be easily evaluated. With the ESDC, once data are cubed, users no longer have the possibility to go back to the original data. We think this issue is important, and missed a discussion on this matter in the paper. Given that large, operational systems exist that do not store precubed datasets but that work with data cube views (Pebesma et al., 2019) we believe that this may be at least a viable option. *At first glance, the concern raised here seems to be of rather fundamental nature. But after some discussion we think it is probably not that critical for most aspects of paper. First of all, the question of regridding is not at all questioning the fundamental principles as outlined in section 2 (the concept we promote here) and rather concerns the concrete implementation described in section 3.2. With respect to the implementation we would like to emphasize that we too see a lot of potential strategies that need to* *be explored. There is clearly much room for improvement. Some thoughts on the specific question of pre-gridding versus on the fly pre-processing: Systems that allow for on-the-fly regridding would probably need to cache access to data that come in a resolution that is lower than requested. Otherwise, such a system would have to re-read all the data for every computation. We therefore like to think that the pre-curation of a cube in different resolutions can be conceptually seen as an explicit "cache". A cache that has been created with some fundamental considerations on how it is preprocess (e.g. guaranteeing mass balances when later summing up). Yet another point is that we don't really understand how fast computations can be achieved when aggregating on the fly. But in essence we admit that we have also a historical legacy: When we started implementing the first versions of the ESDL a few years back, we had the choice to either to regrid the data and quickly come to an usable cube, or to invest much more time in the computational developments. Our practical decision was to go for the regridding, and focussing all our energy on the conceptual multidimensional aspects of the implementation as requested by the scientific community. In fact we had several user consultation meetings along these lines. Still, and for us this is important to communicate here, we are not at all against considering different solutions to the problem and considering user-defined on-the-fly regridding. In fact, we had already several discussions among co-authors regarding this aspect. What we are missing so far for such an approach is a suitable storage backend library that can deal with this and would allow us to offer this approach.*

2 Is latitude the same as time?

In Line 598, the paper states that "The ESDL is probably the most radical data cubing approach", and refers to some grey literature that also claims that data cubes should treat all dimensions identically, irrespective their semantics. We believe however that space and time are inherently different, and need different treatment. The example (e.g. fig 3) shows that all longitudes are aggregated to give profiles per latitude and time, but we feel that this is a rather contrived ex-

ample; in general, any transect in space could be a good candidate for reducing space to one dimension, and this would require mapping onto that transect; it is not so likely that you would want to do that on an arbitrary transect in the two-dimensional space defined by e.g. (longitude, time). In general, a large number of functions applied to space operate on both spatial dimensions (e.g., polygonal crop), and time series models are of a very different kind and are rarely applied to single spatial dimensions. The fact that you can do this is nice, but we do not believe it more of a big selling point than an opportunity for users to shoot themselves in the foot. *Thank you for this important comment which has many aspects to discuss. Let's start with the least critical point: We agree that one should not really refer to gray literature, but in this very case the Data Cube manifesto did come out in parallel to our developments and we thought that it would be appropriate to cite outcomes from projects that have thought along similar lines, even if their implementation philosophy is very different than ours. The question whether space and time have to be regarded as inherently different, however, is not that obvious to us. We agree that one cannot compare these aspects in physical and philosophical terms, but, and this is important for us, to the computer they can look identical. This makes our lives easier, as we can write any UDF without thinking about the characteristic data-properties of the different dimensions. We also don't agree with the perception that Fig. 3 is a contrived example. There are many examples where such statistics are used, e.g. when averaging primary production by latitude and then repeating this for different periods of the year. Another example are the famous "flying carpets" (e.g. CO2 concentrations as a function of time and latitude) e.g. at `http://www.esa-ghg-cci.org/?q=node/115`. In other terms, many standard applications in the Earth system sciences do make it necessary to have equal access to space and time. Another prominent example is certainly the Hovmöller diagram type. And yes, why not applying a PCA on this to retain the underlying orthogonal components only? Dimensionality reduction applications can well require either considering space and time or time and variable or space and variables in a single framework. Furthermore, we would argue that many relevant operations* *can be reduced to convolutions (in space or time) and formulated for the 1D, 2D or 3D case. It is very important for us to emphasize these points, because we want to firmly reject the argument that we are looking for ' selling points". Of course there are models that only make sense in a temporal context, e.g. causal time series models. But we trust that any scientist applying a model of this kind is sufficiently knowledgeable to also understand what dimensions can be addressed with it.*

3 Is a lat/long grid the only way we can cube the Earth?

No, it isn't, and many datasets use other global grids, discrete global grids, or collections of grids (Equi7, or UTM zones). All this has reasons, and the Earth will never be flat so the problem will remain. A discussion on generalizing the ESDC to other grids, or collections of these, would be welcome. *Indeed it isn't. We have a bias of thinking in lat-long-worlds because we are coming from a branch of science where basically all widely used datasets are distributed on a such a longitude-latitude grid. But we totally agree that other grids are likewise highly relevant. We will discuss this accordingly in the revised manuscript.*

4 Vector data cubes are missing

In the representation of the ESDC, the implicit assumption is being made that data cubes correspond to spatial raster data. This is not the case: space can be a one-dimensional set of feature geometries (e.g. points, or polygons). One of the most requested feature of data cubes is to retrieve all the cube information at a given set of points, or aggregated over a given set of polygons. This leads naturally to vector data cubes. Can ESDC answer such queries, or do the authors consider this to not be data cubes? *This question touches a common request. In order to show that we can likewise deal with other vector data cubes we are elaborating a fourth use case that we will include either in the revised version of the paper or the appendix. In short, our answer we consider these data cubes and we can deal with them.*

5 Where is support?

Spatial grids may refer to a collection of points on a regular grid, or to a collection of grid cells as if they were small square polygons. In the latter case, properties can be either continuous over a grid cell (e.g., land use, soil type, geology) or may be aggregated values over the grid cell (e.g., the total amount of carbon in the grid cell, or its maximum elevation). Similarly can time be conceived as a set of time instances or intervals, with the two interval interpretations. Does the ESDC take care of some of these options, or is this all assumed to be remembered by the users? *In the current form, we do not treat such grid cells differently. There are, however, developments that will try to get this solved in the near future cf. https://github.com/JuliaGeo/DimensionalArrayTraits.jl.*

6 Code and reproducibility

We are happy to learn that the software is open source, and can be used by others. Have the authors heard success stories of others installing and using the software? A few small code examples in the paper to get a taste for how simple queries to the ESDC look like would have worked well, and could be encourage those hungry for trying it out. We did not try out the Julia script but hope that other reviewers will report whether they did, and whether they were successful in reproducing the results shown in the paper. *Yes, we have finalized an experiment within an "Early Adopter Call" funded by ESA where students from the B.A. to the PhD level had a few weeks of time to work with the ESDL system. The results showed a very broad way to utilize it. For instance, one project adapted the ESDL to ingest high resolution Sentinel 1 data. This shows us that it seems feasible to work with the software. Regarding the question for "small code examples" we have in fact the opposite request from reviewer 2 - removing the implementation part. So we think the current solution of having real code examples on a git is better as it also has more flexibility compared to having them in a paper.*

7 Dropping dimensions

If a dimension has only one value, it is dropped; we can see this is useful, but it also drops the information of the dimension (e.g. the species name, or the elevation value). We see a lot of 4-dimensional NetCDF files in the wild having 1 dimension with one values, to specify the value for that dimension, which seems all but useless. You need to be able to drop it, but can the ESDC also not drop it, e.g. so that sub-cubes can be meaningfully combined along the otherwise dropped dimension? *A comparable comment has been raised by reviewer 2. To reiterate our answer: We have decided for dropping dimensions "as default" for keeping data cubes that result from some operation at their minimal dimensionality. We are aware that this approach also means losing information. For the example you give: you have e.g. a cube of lat, lon, species presence. If you would subset the cube and only retain the presence-absence of one species in a single cube we would actually expect the user to name the cube accordingly. That aside: the main point we have to make is that there is no problem in outputting a dimension of length 1. This can be easily integrated in both our notation and implementation if desired. For the latter, the `mapCube` function can do exactly this if the output dimension would be given with length 1.*

8 The data model

Having a data model that maps from dimensions to IR and NA is great; a similar approach was adopted in Appel and Pebesma (2019). For end users it also means there is no way to properly handle logical (TRUE/FALSE or NA), categorical, or e.g. time variables. Of course 8-byte doubles can encode anything, but everyone who has tried to use them for encoding categories knows the nightmare. Do end users need that? Some sort of a discussion on this issue would be welcome. *Any number of logical type can be represented here and we just tried to propose a notation that is "close" to what we suspect is the expectation of the majority of potential users.*

9 Other issues

1. Can the ESDC cope with irregular dimensions, e.g. irregularly distributed time steps? *In principle, i.e. if the irregularity is consistent across all other dimensions, yes. We simply need a dimension e.g. an irregular time step that repeats across variables or model runs or whatever property the other dimensions encode. If we would have different time-steps in different variable we have the same issue as discussed above on dealing with different lat-long grids and we are back to the question if we should enable on-the-fly regridding.*

2. lines 260-270: array databases by default store the data in a database, not in HDF5 or NetCDF, although some can be made to do so; theymay import data from HDF5 or NetCDF though. We believe the "Earth system scientists" mentioned in line 269 will soon be the old school if data cube access principles get more widely used beyond GEE and ESDC. No data scientist will want to go back to the individual files underlying a properly implemented data cube. *Well, but it is important for us that one can easily understand where the data are... But yes, people may not do that if there would be "properly implemented data cubes" indeed.*

3. line 180: we think that spectral decomposition does not map from (time) to (time,freq), but to (freq). Eq (12) and (13) have the same problem. *You would be right, if we had analyzed the power spectrum. But, as we write in the text this is about "discrete" decompositions via FFT. We will elaborate the text better to make this crystal clear.*

4. Eq (15) para should be par? *We used `para` in all points of the paper where we talked about parameters. `par` would be interpreted by many people as "Photosynthetically Active Radiation".*

5. line 500 ff: we believe that UDFs are quite widely spread and are implemented in SciDB, rasdaman commercial, openEO (GeoPySpark/GeoTrellis, Grass GIS), R package stars, and Python module xarray. Seeing an example of an ESDC

UDF in the paper would be nice! *All of these examples allow definition of UDFs, but they are in our mind not yet first-class citizens of the ecosystem. For us a UDF should be applicable on every combination of input axes and be equally efficient as applying "built-in" functions. For example, when applying a user-defined function in* `xarray+dask` *(*`xarray.apply_ufunc`*) with time as a core dimension. In this case the user is still limited to functions that are efficiently operating on many spatial data points at once, i.e. which can be expressed using vectorized numpy operations. Otherwise, if the user wants to apply a function which is defined to work on single time series only over a spatial grid, then this is possible (setting the argument vectorize=True), but will have significant performance overhead. Here we think that our Julia implementation is quite unique, because the user can formulate the algorithm in the most natural way (writing a function that operates on a single time series) and can apply this to cubes of any dimensionality with negligible overhead. One of the best examples is the definition of the sufficient_dimensions function, which can be found here:* `https://bit.ly/2ZgUltT`*. Here we apply an intermediately complex method on multivariate time series for each spatial location on a data cube which does not fit into the computer's memory. The whole operation is defined and applied with only a few lines of code using multiple processes and runs reasonably fast.*

Finally, we would like to thank Dr. Marius Appel and Prof. Edzer Pebesma again for their contribution to this discussion. We will acknowledge their great comments also in the revised version of the paper.