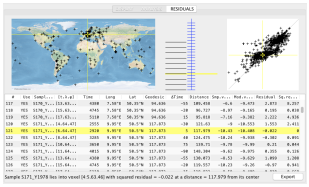
Software

TIMESCAPE GLOBAL

SPACE-TIME INTERPOLATION TOOL: Projected Coordinates Java Standalone Application

Marco Ciolfi,[a] Francesca Chiocchini,[a] Michele Mattioni,[a] Marco Lauteri[a]



Many ecological variables vary both in space and time and the datasets are often distributed far form the ideal statistical sampling: the Timescape algorithm attempts to cope with environmental sciences datasets with a scattered spatial and temporal distribution. TimescapeGlobal deals with continental to worldwide datasets, based on angular (longitude and latitude) coordinates. The evaluation of the distances with angular coordinates is arithmetically intensive and topologically tricky: TimescapeGlobal uses a simplified spherical great circle distance evaluation, which takes into account the cyclic (longitude) and singular (latitude) coordinates needs. The software offers a variety of variogram options and other descriptive statistical tools for the analysis of data distribution. The interpolation can be done with some preloaded algorithms or with any user-defined function, since the application includes a fairly extensive syntax parser. A typical run lasts some hours, depending on the complexity of the algorithm and the number of output cells.

Keywords: Software, Java, projected coordinates datasets.

1 Introduction

Timescape Global is a geostatistical interpolation tool. It is aimed at scientists and researchers who routinely work with datasets which include time as well as space evolution patterns. Standard, "flat", geostatistical algorithms[I] are implemented in most GIS software suites, but they do not include time as one of the driving factors of change.

Timescape Global is tuned for the investigation of continental-scale evolution phenomena (including the Earth as a whole),[II] it uses geographical (longitude-latitude) coordinates and the surface distances are evaluated as geodesic arc lengths.

The time is converted into a sort of third dimension, with some peculiarities which distinguish it form an ordinary real number. In particular, there is a *forward-only constraint* that guides the evaluation procedure.

The software produces three-dimensional discrete *voxel* models.[III] These models can be explored in both a time- or space-oriented way, producing time series as well as Grid layers to be used for further analysis in users' GIS an statistical workflow

The models can be composed of a few thousand elements, as a first scratch, but can also be made of billions of elements: this calls for the usage of an external data storage. The users can choose the database management system they like most. MySQL schema creation files are provided with the installation package; other RDBMS flavors require some customization.

The software is written in Java, so it is portable, OS-independent and it is based on standard framework components. The source code is available as part of the package (see section 13 for code details and section 2 for license).

The interpolation algorithm is an extension of the Inverse Distance Weighting so it produces, in a broad sense, *deterministic* models. In fact, users can expand the basic IDW algorithm in many ways: they can control the role of time in the evaluation of distances up to the finer details, users can also define their own interpolation functions, also including ancillary data in the evaluation of values.

1.1 The Application

The main object of **Timescape Global** is the *model*: it is a collection of values located at regular space-time lattice sties, which are evaluated interpolating the values of a *sample point dataset*. Only one dataset can be used, shared by all the models. **Timescape Global** has four distinct panels: a general manager; a **Setup panel** for defining the characteristics of each model; a **Evaluation panel** for the actual interpolation and an **Exploration panel** for exploring finished models and exporting parts of them.

[a] *Istituto di Biologia Agroambientale e Forestale (IABF), Via Guglielmo Marconi N. 2, Porano (TR), Italia.*

I For a thorough introduction to the field of geostatistics see for example [1]. The more recent [2] is devoted to space-time statistics. Users should be aware, in particular, of the problem of *stability* which is central in geostatistics. This topic is never mentioned directly throughout this manual, but just note that the addition of time variability does not diminish the importance of such an issue.

II Its companion tool, **Timescape Local**, focuses on local areas, using local projected coordinates. It is also available on Sourceforge [3,4].

III A *voxel* (volume element) is the three-dimensional counterpart of a *pixel*, a two-dimensional picture element.

The algorithm is described in detail in section 3, it is a rather technical exposition that can be skipped at a first reading. The zipped version of the algorithm is:

- every model is based on a subset of the sample dataset.
- The time is *spatialized*, i.e. converted in a third spatial dimension. The conversion factor can be a constant or a function of position and time itself.
- The surface (geodesic) distance is corrected taking into account the difference of times between the sample points and the voxel being evaluated.
- A *causality constraint* is checked to select the relevant source points for any voxel. This can be pictured as an upside-down *cone* emerging form a source point: the broader the cone, the looser the constraint. Users can modulate this *cone* from a laser-thin aim to the entire Earth surface, according to the patterns of change which are typical of their particular field of research.[IV]
- Each voxel's value is evaluated weighting the values the nearest points. How to weight such values is chosen by the user, via the definition of suitable functions. Harmonic (periodic) function can be used, as well as ancillary data associated to the sample points.

1.2 What Timescape Global is for

- It is suited for interpolation whenever time and space variability of sampled data are of the same order of magnitude and of the same importance for the modeled phenomenon.
- The research fields of interest include Earth sciences, biology and environmental sciences.
- it can be run on any respectable computer, even a laptop. Database room and running times scale with the number of voxels.[V]
- It fits well in a "old times" workflow, since it outputs .csv text files of values other than Grid layers. Many exports are easily loaded into R data-frames. Images and .gif animations can be exported as well.

1.3 What Timescape Global is not for

- It is not a complete GIS package itself. It is intended to be an addition to the geostatistical toolbox in an established GIS workflow.
- It is not aimed at meteorology, since there is no room for the third space dimension, height, which in fact would produce four-dimensional models.
- It is not for the casual user, since the initial configuration takes some time and the learning curve has a steep start.
- It is not for high precision geodetic measurements, since the Earth shape is approximated with a simple sphere.

1.4 Manual overview

The software installation and configuration is discussed in detail in sections 4 and 5. A configuration tool (sec. 5.5) can be of some help.

Users can try the sample datasets included in the software distribution (sec. 15) to check whether it all works, before moving to their actual projects.

Section 7 describes in detail the model definition tool (blank models are created and managed by mean of a dedicated tool, see sec. 6).

The evaluation phase (section 8) is the simplest one, by an user's point of view, but it is the most stressful one for both the running machine and the database server.

Finished models can bee explored, drilled, sliced and all this can be exported, see section 9. Section 10 describes in detail the structure of all exported files.

2 Version and License

Timescape Global is free software, it is released according to the GNU GPLv3 License, please refer to [5] for details. The source code is part of the standard distribution.

Users are allowed and encouraged to modify any part of the software according to their particular needs. Users are also encouraged to publish their contributions following the same licensing policy. The logo on the main window may be changed, see section 14 for details.

Users cannot include parts of this software into non-open derivative products.

Timescape Global is provided as-is. It is aimed at scientific research institutions working on Earth sciences, biology and environmental issues, possibly as part of a broader GIS infrastructure. The software can be quite demanding in terms of processor usage. Data storage is on a separate Database.

IV Choosing different conversion factors and causal constraints produces completely different models. See section 3.3 for details.

V The hardware architecture is easily scaled: for example one can use one database server per dataset and one or more machines for models' evaluation. The actual resources allocated depend on the user's Java Virtual Machine settings.

Timescape Global is a branch of the fork of TimescapeZero: the original, unpublished, space-time interpolation software project. TimescapeZero has been split into **Timescape Local**, also released under GPL3 License [3], and **Timescape Global**. The first uses local (projected) coordinates, while this one is based on geographical coordinates, so it is more suited for wide-area modeling, from a regional/continental scale to the whole Earth.

This is the official release, TimescapeGlobal1.1β, with minor changes from the previous 1.0β

Please report any issue or bug to the author: `marco.ciolfi@ibaf.cnr.it`

3 Timescape Global Algorithm

3.1 A mathematical premise

Global spacetime models are built according to the appropriate $\mathbb{S}^2 \times \mathbb{R}_0^+$ topology, where the spatial coordinates [VI] $(\lambda, \varphi) \in \mathbb{S}^2$the two-dimensional sphere, and the time $t \in \mathbb{R}_o^+$, the set of non-negative real numbers. The models are equally spaced according to angular coordinates, so they are not evenly spaced according to a regular projected lattice. In particular, large $|\varphi|$ areas (the polar regions) are oversampled with respect to small $|\varphi|$ areas (tropical regions): this is a common issue of regular lattices in spherical coordinates which translates, on practical grounds, in a harmless evaluation time loss.

A single model is a lattice of longitude/latitude equally spaced cells, replicated for a certain number of time sheets. The thickness of such sheets should be chosen according to the time sales typical of the phenomenon under investigation, in particular if one has to model periodic changes of the values.[VII] On the other hand, the width of the space cells is not so critical and can be chosen according to one's needs; sometimes it can be useful to subdivide a model in a union of an equatorial plus two polar submodels, with different cellsizes, especially if the model will be resampled as part of GIS project.

A *model M* is defined as a discrete collection of volume elements (*voxels*) $\{m(t_k; \lambda_i, \varphi_j)\}$ where t_k labels the time sheet and (λ_i, φ_j) the space cell:

- t_k is the center of the k-th time sheet, $k \in \{1 \dots N_T\}$, t has a lower bound (conventionally $t_0 = 0$) and as a matter of principle it would not have an upper bound. Although for practical reasons the values of t much greater than the maximum value of the samples' times could be meaningless. The number[VIII] N_T of sheets of thickness w_s is $N_T = \left\lceil \frac{T_M - T_m}{s_s} \right\rceil$ where $t \in [T_m, T_M]$, m and M indicating the minimum and maximum values, respectively.
- λ_i is the i-th longitude cell, $i \in \{1 \dots N_\Lambda\}$, λ is a cyclic coordinate $(\lambda + 2n\pi = \lambda)$. The number N_Λ of cells of size w_c is $N_\Lambda = \left\lceil \left| \frac{\Lambda_M - \Lambda_m}{w_c} \right| \right\rceil$, where $\lambda \in [\Lambda_m, \Lambda_M]$. A meaningful model can have $\Lambda_m, > \Lambda_M$, provided that $\Lambda_M > \Lambda_m + 2\pi$.
- φ_j is the j-th latitude cell, $j \in \{1 \dots N_\Phi\}$. φ is an angular, but not cyclic coordinate. The poles (North $\varphi = +\frac{\pi}{2}$ and South $\varphi = -\frac{\pi}{2}$) are singular points: they coincide for all λ values. The number N_Φ of cells of size w_c is $N_\Phi = \left\lceil \frac{\Phi_M - \Phi_m}{w_c} \right\rceil$, where $\varphi \in [\Phi_m, \Phi_M]$.

The total number N of voxels building up the model is thus $N = N_T N_\Lambda N_\Phi$. This can be quite a large number, given approximatively by

$$N = \left\lceil \frac{|\Lambda_M - \Lambda_m|\,(\Phi_M - \Phi_m)(T_M - T_m)}{w_c^2 w_s} \right\rceil$$

A number of cells about e.g. some thousands in space and time gives models composed of several billions of voxels. Since the typical size of geostatistical raster models is between hundreds and thousands of cells for both spatial coordinates, it is the number of time sheets that ultimately controls "how bulky" the model actually is. Since any single time sheet is independent from the others, one can subdivide a big model in a number of smaller, more manageable, ones.

3.2 Distance in space and time

Since the weighting of the samples values of the model is dependent on the distance, we must define a *metric* in $\mathbb{S}^2 \times \mathbb{R}_0^+$: A metric (i.e. a distance function) $d : X \times X \to \mathbb{R}_0^+$ should satisfy:

- non-negativity: $d(x,y) \geq 0 \ \forall x, y \in X$
- symmetry: $d(x,y) = d(y,x) \ \forall x, y \in X$
- coincidence: $d(x,y) = 0$ iff $x = y$
- subadditivity: $d(x,z) \leq d(x,y) + d(y,z) \ \forall x, y, z \in X$ (triangle inequality)

We define separately a metric on $\mathbb{S}^2$ for the spatial part and on $\mathbb{R}_0^+$ for the temporal one. The Earth surface is approximately spherical: at a global scale we can use a relatively simple geodesic approximation of the distance between two points P_0 and P of

VI There is some confusion in literature about geographical coordinates labelling. Here and henceforth we will stick to the convention that λ represents the *longitude*, from $-\pi$ (180° W) to $+\pi$ (180° E), while φ is the *latitude*, from $-\frac{\pi}{2}$ (90° S) to $+\frac{\pi}{2}$ (90° N). All the calculations are performed in radians but the user interface is in decimal degrees.

VII To represent periodic phenomena it is advisable to subdivide the period into a whole number of sheets, e.g. one sheet per month to model yearly periodic variations.

VIII The symbol $\lceil \dots \rceil$ represents the *ceiling* function, i.e. the largest integer approximating the argument.

coordinates, respectively, (λ_0, φ_0) and (λ, φ): the distance $D_{\lambda\varphi}$ is taken as the Earth radius R times the aperture $\widehat{P_0OP}$ of the $arc\widehat{P_0P}$ (the geodesic line lying on the surface of the sphere connecting points P_0 and P):

$$D_{\lambda\varphi}(P_0, P) = R \arccos\left(\sin\varphi \sin\varphi_0 + \cos\varphi \cos\varphi_0 \cos(\lambda - \lambda_0)\right)$$

The distance in time needs some care to be defined. Let's call an *event* $E = (t; \lambda, \varphi)$ any point belonging to $\mathbb{R}_0^+ \times \mathbb{S}^2$, so that a voxel model is any discretization of a finite subset of $\mathbb{R}_0^+ \times \mathbb{S}^2$, each event possibly carrying a value:[IX] $M \subset \mathbb{R}_0^+ \times \mathbb{S}^2 \times \left(null \cup \mathbb{R}\right)$.

The first step consists in transforming the time coordinate in a space coordinate (spatialization): this can be achieved through the multiplication by a constant factor c or by a function[X] $c(E_0, E)$ of the events coordinates. Both c and $c(E_0, E)$ must have L T^{-1} dimensions (like a velocity) so that a time multiplied by c gives a space. So we define the *time distance* D_t between any two events E_0 and E as:

$$D_t(E_0.E) = c(E_0, E)\,|t - t_0|$$

Note that $D_t(E_0, E) = D_t(E, E_0)$ only if $c(E_0, E) = c(E, E_0)$, as it is the case, obviously, whenever c is a constant. The value of c is linked to the typical velocity of diffusion of the values in the model lattice which, in turn, is loosely linked to the actual physical processes underlying the modeled phenomenon.

Then we have to combine space and time. Let's use the geodesic distance between points on the surface of a sphere as the spatial part of the distance between events: $D_{\lambda\varphi}(E_0, E) \equiv D_{\lambda\varphi}(P_0, P)$. Then apply the Pythagorean theorem to the distance components, defining the *distance* $D(E_0, E)$ *between the events* E_0 *and* E:

$$D(E_0, E) = \sqrt{D_t^2 + D_{\lambda\varphi}^2}$$

which is

$$D(E_0, E) = R\sqrt{\frac{c^2(E_0, E)(t - t_0)^2}{R^2} + \arccos^2\left(\sin\varphi \sin\varphi_0 + \cos\varphi \cos\varphi_0 \cos(\lambda - \lambda_0)\right)}$$

that reduces to

$$D(E_0, E) = R\sqrt{\frac{c^2}{R^2}(t - t_0)^2 + \arccos^2\left(\sin\varphi \sin\varphi_0 + \cos\varphi \cos\varphi_0 \cos(\lambda - \lambda_0)\right)}$$

if the conversion factor c is a constant. The c/R ratio of the latter highlights the importance of the c parameter in modulating the influence of time in the model: since the distances on the surface of the Earth typically range up to thousands of kilometers, the value of c times the model time range should be of such an order of magnitude, in order to influence the distance sensibly. Moreover, a c factor that is locally dependent on the events introduces anisotropies which need to be accounted for by the patterns of change of the modeled phenomenon.

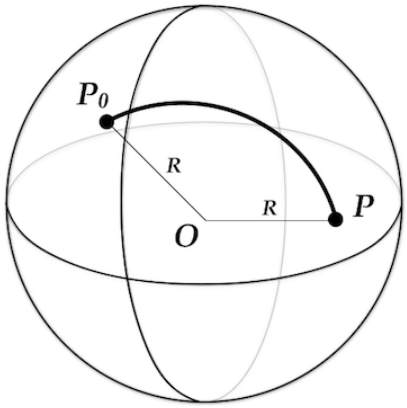


Fig. 1

3.3 Causality

The inclusion of a causal constraint is a key feature of Timescape software: the basic idea is that it take time to move things around, i.e. the area influenced by event E grows up with time, so that only a certain area $A_E(t)$ can be affected at time t by what happens at event E. An model value at event E can depend on event E_0 only if $E \in A_{E_0}(t)$. plus, a forward-only constraint is introduced: $A_E(t)$ is defined only for the future of E.

The figure 2 illustrates the concept of *causal cone*: an event E is causally connected to the event E_0 because it lies within the cone, while event F is not causally connected to E_0 since it is outside the cone. The width (apex angle) of the causal cone $K(E_0)$ modulates the causally accessible subset of $\mathbb{S}^2 \times \mathbb{R}_0^+$ to any event E_0. $K(E_0)$ is define d as the union of all the sets $A_{E_0}(t)$ evolving from E_0:

$$k(E_0) = \bigcup_{t \in \mathbb{R}_0^+} A_{E_0}(t)$$

TThe term *cone* is somewhat improper, since the actual shape of $A_{E_0}(t)$ can in principle be any two-ball of $\mathbb{S}^2$ centered on E_0 . The extreme cases of a point-like $(A_{E_0}(t) = \{E_0\} \forall t)$ and infinitely-broad cone $(A_{E_0}(t) = \mathbb{S}^2 \forall t)$ are also included.

For any event E, the set of possible causes $C(E)$ is formed by the events that can be considered as such, i.e. the events E_k which causal cone $K(E_k) \ni E$:

$$c(E) = \{E_k \in \mathscr{E} \mid E \in K(E_k)\}$$

IX The term *event* comes from relativistic physics, meaning a point in spacetime.

X Care should be taken in the definition of the conversion function for the risk of violation of the constraints of the metric function, in particular the triangle inequality.

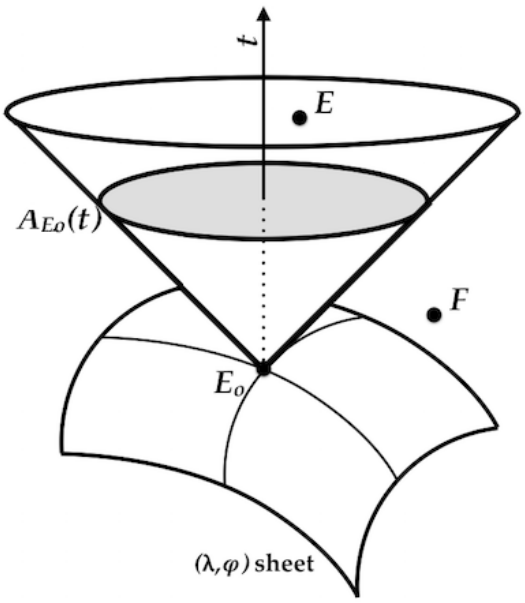


Fig. 2

where $\mathscr{E}$ is the set of all sample points used in the model. If $C(E) =$ the value of E is null. Naïvely speaking, the causal cone is just the area of influence of the model sample points, which generally broadens with the time.

Operatively, we define an apex angle, or better a maximum allowed $D_{\lambda\varphi}/D_t$ ratio k, which in general is a function $k(E_0,E)$ of the sample event E_0 and the model voxel event E such that

$$E \in K(E_0) \text{ whenever } \frac{D_{\lambda\varphi}}{D_t} \leq k(E_0,E)$$

or, more practically, $D_{\lambda\varphi} \leq k(E_0,E)D_t$, as if $k(E_0,E)D_t$ were the maximum spatial distance that the phenomenon we're investigating can "travel" over a time t.

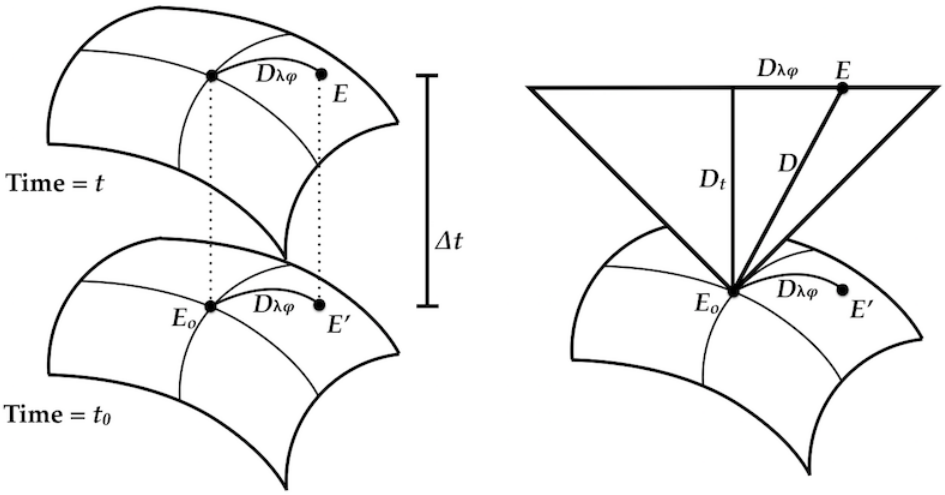


Fig. 3

The causal constraint is checked this way:

- First, the space distance $D_{\lambda\varphi}$ from E_0 to E' (a fictional event sharing E_0 space coordinates and E time) is evaluated and translated over time for $\Delta t = |t - t_0|$
- the time distance $D_t = c(E_0,E)|t - t_0|$ is evaluated, from the latter we can compute the radius of $A_{E_0}(t)$ which is $c(E_0,E)k(E_0,E)|t - t_0|$
- if $E \in K(E_0)$, that is if $D_{\lambda\varphi} \leq k(E_0,E)D_t$, the distance D is $\sqrt{D_{\lambda\varphi}^2 + D_t^2}$
- otherwise E_0 is thought to have no influence on E.

As said, the causal constraint can be relaxed up to the inclusion of all possible sample points as possible causes ($C(E) = (E)\ \forall E$). It is also possible to release the time-forward-only constraint replicating an upside-down version of the infinitely broad cone: this

way the time becomes fully a third spatial dimension (set aside the c factor transformation).

On the opposite side of the causality scale, we can define a collapsed cone, which contains only the future events place at exactly the same site; this models a completely stationary pattern (admittedly, there's nothing particularly interesting to model in this case).

3.4 Timescape modeling algorithm

A Timescape model M consists of a finite set of voxels (E, v)

$$M = \left\{ (E_{kij}, v_{kij}) \mid E_{kij} \in \mathbb{S}^2 \times \mathbb{R}_0^+, v_{kij} \in \{\text{null}\} \cup \mathbb{R}, k \leq N_T, i \leq N_\Phi \right\}$$

of values v_{kij} defined on a finite *support* $\{E_{kij}\}$ of spacetime events in $\mathbb{S}^2 \times \mathbb{R}_0^+$. the values can be real or null, whenever the predictive power is too scarce.

In order to build a model we must start from a finite set of *sample points* S which represent the actual observations (i.e. the physical measurements) of the investigated phenomenon:

$$S = \left\{ (E_n^0, v_n) \mid E_n^0 \in \mathscr{E}, v_n \in \mathbb{R},\ n = 1 \ldots N \right\}$$

We than apply a set of deterministic inference rules to obtain M from S.[XI] Models are computed according to a broad generalization of IDW (Inverse Distance Weighted) algorithm. Any model is characterized by a set of parameters:

- space and time bounds $[\Lambda_m, \Lambda_M] \times [\Phi_m, \Phi_M]$ and $[T_m, T_M]$
- space cell size w_c and time sheet thickness w_s
- maximum number N_{max} of S elements for interpolation (can be unbound)
- probability $\mathscr{P}$ of picking an element for interpolation (usually 1)[XII]
- maximum geodesic distance $D_{\lambda\varphi}^{max}$ for a source E_n^0 to be considered causally relevant
- time-space conversion factor c or function $c(E_0, E)$
- causal cone aperture factor k or function $k(E_0, E)$
- weight function $w(E_0, E)$ and value function $v(E_0, E)$

In its most basic implementation the IDW algorithm calculates output values as a weighted, inverse distance-based, mean of the input values:

$$\text{value(out)} = \left[\sum_{in} \frac{\text{value(in)}}{\text{dist(in,out)}} \right] / \left[\sum_{in} \frac{1}{\text{dist(in,out)}} \right]$$

A bit rough, this modeling procedure often outputs characteristic bulls-eyed surfaces, but it has some interesting features that can be expanded in several ways. This is what the Timescape does: it keeps the basic assumption of a weighted mean, but allows the definition of whatever function for the value and the weight, plus a few other fine tuning capabilities.

Before going to the actual voxel values calculation, for the sake of efficiency, the software calculates the lattice grid coordinates of the voxels, placing them in the vectors $\underline{t}$, $\underline{\lambda}$ and $\underline{\varphi}$.

Algorithm 1 Preliminary steps

procedure COORDINATES($w_c, w_s, T_m, T_M, \Lambda_m, \Lambda_M, \Phi_m, \Phi_M$)
  $\delta t \leftarrow w_s$
  $t_0 \leftarrow T_m + \delta t/2$
  while $t_n \leq T_M$ **do** ▷ voxel sheets times $\underline{t}$
    $t_{n+1} \leftarrow t_n + \delta t$
  $\delta\lambda \leftarrow w_c$
  $\lambda_0 \leftarrow \Lambda_m + \delta\lambda/2$
  while $\lambda_n \leq \Lambda_M$ **do** ▷ voxel longitudes $\underline{\lambda}$
    $\lambda_{n+1} \leftarrow \lambda_n + \delta\lambda$
  $\delta\varphi \leftarrow w_c$
  $\varphi_0 \leftarrow \Phi_m + \delta\varphi/2$
  while $\varphi_n \leq \Phi_M$ **do** ▷ voxel latitudes $\underline{\varphi}$
    $\varphi_{n+1} \leftarrow \varphi_n + \delta\varphi$
return $\{\underline{t}, \underline{\lambda}, \underline{\varphi}\}$

Then the model is evaluated, one voxel at a time:[XIII]

XI It is possible to add some random spicing to the computation, picking the sample points somewhat at random, but the modeling is basically deterministic.

XII This is the only random component of the algorithm, see the previous note.

XIII Operatively, all voxels are inserted empty in the database before their actual evaluation. This is done in order to minimize the database file growing / shrinking during the run. Voxels records will just be updated with the appropriate values after evaluation: this does not change the record size.

Algorithm 2 General model loop

procedure MODEL($\underline{t}, \underline{\lambda}, \underline{\varphi}, \mathscr{E}, N_{max}, \mathscr{P}, D^{max}_{\lambda\phi}, c, k, w, v$)
  for $t_k \in \underline{t}$ **do**
    for $\lambda_i \in \underline{\lambda}$ **do**
      for $\varphi_j \in \underline{\varphi}$ **do**
        $E_{kij} \leftarrow (t_k; \lambda_i, \varphi_j)$
        $\mathbf{E} \leftarrow$ CAUSAL($E_{kij}, \mathscr{E}, N_{max}, \mathscr{P}, D^{max}_{\lambda\phi}, c, k$) ▷ causally connected $\{E^0_n\} \subseteq \mathscr{E}$
        if $\mathbf{E} \neq \varnothing$ **then**
          $v_{kij} \leftarrow null$ ▷ an empty voxel
        else
          $V \leftarrow 0$
          $W \leftarrow 0$
          for $E^0_n \in E$ **do**
            $V \leftarrow V + w(E^0_n, E_{kij})\, v(E^0_n, E_{kij})$
            $W \leftarrow W + w(E^0_n, E_{kij})$
          $v_{kij} \leftarrow V/W$ ▷ a valid voxel
  $M \leftarrow \{(E_{kij}, v_{kij})\}$ ▷ the finished model
  return M

The innermost loop of the latter is the IDW implementation step. It is quite straightforward if not for the presence of the weight $w(E_0, E)$ and value $v(E_0, E)$ functions: these unctions can deeply change the behavior of the IDW interpolation.[XIV]

The most involved part of the evaluation algorithm is the selection of the causally connected sample points to be used in the IDW-like part.

Algorithm 3 Causally connected samples evaluation - I

procedure CAUSAL($E_{kij}, \mathscr{E}, N_{max}, \mathscr{P}, D^{max}_{\lambda\phi}, c, k$)
  $\mathbf{E} \leftarrow \varnothing$
  for $E^0_n \in \mathscr{E}$ **do**
    if RAND $\leq \mathscr{P}$ **then** ▷ random spicing, always true if $\mathscr{P} = 1$
      $K \leftarrow k(E^0_n, E_{kij})$
      $D_t \leftarrow c(E^0_n, E_{kij})\,|t - t_0|$
      $D_{\lambda\varphi} \leftarrow R \arccos\left(\sin\varphi \sin\varphi_0 + \cos\varphi \cos\varphi_0 \cos(\lambda - \lambda_0)\right)$
      if $D_{\lambda\varphi} \leq K D_t \wedge D_{\lambda\varphi} \leq D^{max}_{\lambda\varphi}$ **then**
        $d_n \leftarrow \sqrt{D^2_{\lambda\varphi} + D^2_t}$
        $E \leftarrow E \cup \{(E^0_n, d_n)\}$ ▷ one more causal event
  if $\mathbf{E} \neq \varnothing$ **then**
    $\mathbf{E} \leftarrow$ SORT($\mathbf{E}, d$) ▷ order by increasing distance
    $\mathbf{E} \leftarrow$ TRIM($\mathbf{E}, N_{max}$) ▷ keep only the closest N_{max} events
  return $\mathbf{E}$

Operatively, the $\mathbf{E}$ is recycled from voxel to voxel, in order to minimize the database accesses: $\mathbf{E}$ is initialized once at the beginning of the procedure, containing all the candidate source events, and a flag f_n is switched on and off for any event at any execution of the loop:

XIV A suitable choice of $v(E_0, E)$ can even produce interpolated values outside of the range of values of the sample points $\mathscr{E}$, which is impossible with ordinary IDW. Care should be taken in the choice of such functions, in order to avoid error conditions that would produce empty voxels.

Algorithm 4 Causally connected samples evaluation - II

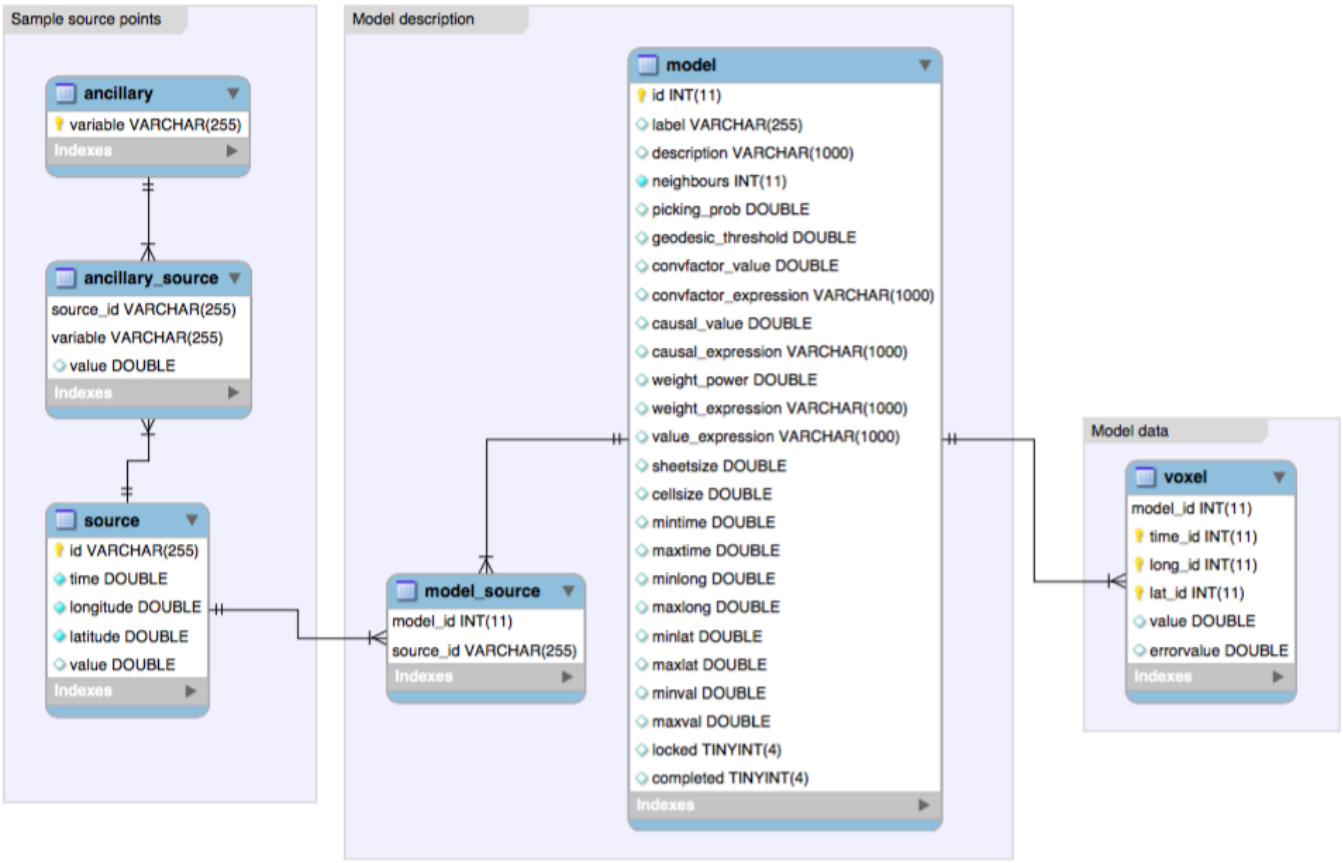
procedure CAUSAL($E_{kij}, \mathscr{E}, N_{max}, \mathscr{P}, D^{max}_{\lambda\phi}, c, k$)
  for $E^0_n \in \mathscr{E}$ **do**
    if RAND $\leq \mathscr{P}$ **then** ▷ random spicing, always true if $\mathscr{P} = 1$
      $K \leftarrow k(E^0_n, E_{kij})$
      $D_t \leftarrow c(E^0_n, E_{kij})\,|t - t_0|$
      $D_{\lambda\varphi} \leftarrow R \arccos\left(\sin\varphi \sin\varphi_0 + \cos\varphi\cos\varphi_0\cos(\lambda - \lambda_0)\right)$
      refresh element $(E^0_n, d_n = \sqrt{D^2_{\lambda\varphi} + D^2_t}, f_n = 1)$ ▷ update event distance
      if $D_{\lambda\varphi} > \min(KD_t, D^{max}_{\lambda\varphi})$ **then**
        $d_n \leftarrow null$
        $f_n \leftarrow 0$
  $\mathbf{E} \leftarrow$ SORT($\mathbf{E}, d$) ▷ order by increasing distance
  for $E^0_n \in \mathbf{E}$ **do**
    if $n > N_{max}$ **then**
      $f_n \leftarrow 0$ ▷ use only the first N_{max} events
  return $\mathbf{E}$

As said, any voxel is evaluated independently from the others, so any model can be easily subdivided according to a union of submodels, provided some care is given to the limits of the submodels and that cell- and sheet sizes are kept unchanged.

A null value of d_n is associated with those events that are outside the causal cone. Null-d elements are sorted down to the tail of **E**.

4 Database structure

The database structure is pretty simple. Three main blocks can be found; a Source block, a Model description and a Model data block:



- Source: Three tables contain the source data values. The main table is **source**, which stores the spacetime coordinates and the values of the sample points.[XV] Two other tables, **ancillary** and **ancillary_source** are used only when ancillary data are collected.

XV Values cannot be null.

- Model: a single **model** table stores all the relevant parameters to each model. Fields include all the descriptors of the model, plus the minimum and maximum values of the evaluated voxels and a couple of flags used to record whether the model is being evaluated or already completed. The **model_source** table is a cross-reference object that store which source point are actually used in each model.[XVI]
- Data: a single **voxel** table is used to store the model voxels. To keep the record size to a minimum, the actual coordinates are not stored in the table. Only the coordinate indices are stored. The *errorvalue* field is as yet unused.

The database is accessed by the application via an *hibernate* midlayer, so to let the user choose his/her preferred database management system. The amount of data calls for an external, robust, data storage system, such ad that provided by a consolidated RDBMS. The ER model shown is the MySQL implementation of the abstract hibernate model. The database creation script is also provided in MySQL flavor.

Although the application is devoted to geographical data, it is not important to have spatially indexed fields. Care is given to keep the number database queries to a minimum and a gain in efficiency is advisable only for the exploration of completed models. During the evaluation of the models, empty records are inserted prior to the actual calculation to avoid the continuous grow of **voxel** table and to be sure that the evaluation will be not aborted due to lack of database space.

The user can change the database flavor and location[XVII] to taste. Connection speed is of the utmost importance, mostly during the phases of voxels insertion and of model exploration (the latter, consisting in select-only accesses, is far less critical then the insertion phase).

4.1 Object details

The detailed structure of the tables follows (see the ER model and sql script for constraints, indices and relations).

Table **ancillary**

INDEX	FIELD	JAVA TYPE	MYSQL TYPE	DESCRIPTION
♢	variable	String	varchar(255)	the ancillary variable name[XVIII]

18 These names are converted to lowercase and preceded by an underscore (_) for user defined expressions.

Table **source**

INDEX	FIELD	JAVA TYPE	MYSQL TYPE	DESCRIPTION
♢	id	String	varchar(255)	the sample name
	time	double	double	sample time, any unit
	longitude	double	double	sample longitude[XIX], d.ddd
	latitude	double	double	sample latitude, d.ddd
	value	double	double	the sample value (cannot be null)

19 Angular coordinates are expressed in decimal degrees.

Table **ancillary_source**

INDEX	FIELD	JAVA TYPE	MYSQL TYPE	DESCRIPTION
♢	source_id	String	varchar(255)	sample id → **source**
♢	variable	String	varchar(255)	variable → **ancillary**
	value	double	double	the ancillary variable value (can be null)

XVI There is not a global on/off switch on source points.

XVII The example above shows a typical development configuration: The database is hosted by the same development unit (*localhost*), it is accessed by the root user without a password protection. Of course this is not an optimal solution.

Table **model**				
INDEX	FIELD	JAVA TYPE	MYSQL TYPE	DESCRIPTION
♢	id	int	int(11)	self-incremented model id
	label	String	varchar(255)	user-defined model label
	description	String	varchar(1000)	user-defined model description
	neighbours	int	int(11)	number of neighboring points[XX]
	picking_prob	double	double	point picking probability[XXI]
	geodesic_threshold	double	double	maximum allowed geodesic distance[XXII]
	convfactor_value	double	double	time to space conversion factor[XXIII]
	convfactor_expression	String	varchar(1000)	time to space conversion expression
	causal_value	double	double	causal cone ratio[XXIV]
	causal_expression	String	varchar(1000)	causal cone ratio expression
	weight_power	double	double	interpolation weight function power[XXV]
	weight_expression	String	varchar(1000)	interpolation weight function expression
	value_expression	String	varchar(1000)	interpolation value function expression[XXVI]
	sheetsize	double	double	time sheet interval, any unit
	cellsize	double	double	space cell size, d.ddd
	mintime	double	double	minimum time, any unit
	maxtime	double	double	maximum time, any unit
	minlong	double	double	minimum longitude, d.ddd
	maxlong	double	double	maximum longitude, d.ddd
	minlat	double	double	minimum latitude, d.ddd
	maxlat	double	double	maximum latitude, d.ddd
	minval	double	double	minimum value, any unit
	maxval	double	double	maximum value, any unit
	locked	boolean	tinyint(4)	model locked flag
	completed	boolean	tinyint(4)	model complete flag

Table **model_source**				
INDEX	FIELD	JAVA TYPE	MYSQL TYPE	DESCRIPTION
♢	model_id	int	int(11)	model id → **model**
♢	source_id	String	varchar(255)	sample id → **source**

model contains all the relevant parameters of the models; the setup phase just updates these values. **model_source** is a cross-join table containing the association model-source point.

Table **voxel**				
INDEX	FIELD	JAVA TYPE	MYSQL TYPE	DESCRIPTION
♢	model_id	int	int(11)	model id → **model**
♢	time_id	int	int(11)	time sheet id
♢	long_id	int	int(11)	longitude cell id
♢	lat_id	int	int(11)	latitude cell id
	value_id	double	double	voxel value, can be null[XXVII]
	errorvalue_id	double	double	voxel value, can be null, unused[XXVIII]

The **voxel** table contains the values and the ids of the coordinates (not the actual coordinates values, which need to be calculated). The `errorvalue` field is unused.

XX neighbours = 0 → use all points

XXI picking_prob = 1 → certainty

XXII geodesic_threshold = -1 → no threshold

XXIII convfactor_value = -1 → parse custom convfactor_expression

XXIV causal_value = -1 → use custom causal_expression, -2 → use all forward points, -3 → also backwards

XXV weight_power = -1 → parse custom weight_expression

XXVI value_expression ≠ *null* → parse this value_expression, otherwise use default _val

XXVII A *null* value indicates an empty voxel

XXVIII This field is intended for storing the estimate error associated to the voxel. As yet unused.

4.2 Preparation of the samples dataset

The samples have to be formatted in a comma separated values file according to the following record structure (ancillaries are optional):

```
ID,TIME,LONG,LAT,VALUE[,ANCILLARY_1,ANCILLARY_2...,ANCILLARY_N]
```

UniqueSampleID	Year	Latitude	Longitude	d18O_permil	dD_permil	?O_anom_pe	anom_pe	Tmp_C	mm_per_
S164-M-1-Y1977	1977	53.87	8.72	-8.1	-57.8	-1.1	-8.7	7.7	68.2
S164-M-1-Y1978	1978	53.87	8.72	-8.3	-60.6	-1.3	-11.5	7.3	55.2
S164-M-1-Y1979	1979	53.87	8.72	-6.7	-50.8	0.2	-1.7	8.1	49.3
S164-M-1-Y1980	1980	53.87	8.72	-7.0	-49.3	0.0	-0.2	7.9	63.5
S164-M-1-Y1981	1981	53.87	8.72	-6.5	-45.9	0.5	3.2	8.6	71.9
S164-M-1-Y1982	1982	53.87	8.72	-7.2	-52.9	-0.2	-3.8	9.5	65.1
S164-M-1-Y1983	1983	53.87	8.72	-7.5	-52.1	-0.5	-3.0	9.1	69.5
S164-M-1-Y1984	1984	53.87	8.72	-6.5	-46.4	0.4	2.7	8.3	56.5

ID	TIME	LONG	LAT	VALUE	D2H	D18OA	D2HA	TEMP	PREC
S164-Y1977	1977	8.72	53.87	-8.1	-57.8	-1.1	-8.7	7.7	68.2
S164-Y1978	1978	8.72	53.87	-8.3	-60.6	-1.3	-11.5	7.3	55.2
S164-Y1979	1979	8.72	53.87	-6.7	-50.8	0.2	-1.7	8.1	49.3
S164-Y1980	1980	8.72	53.87	-7.0	-49.3	0.0	-0.2	7.9	63.5
S164-Y1981	1981	8.72	53.87	-6.5	-45.9	0.5	3.2	8.6	71.9
S164-Y1982	1982	8.72	53.87	-7.2	-52.9	-0.2	-3.8	9.5	65.1
S164-Y1983	1983	8.72	53.87	-7.5	-52.1	-0.5	-3.0	9.1	69.5
S164-Y1984	1984	8.72	53.87	-6.5	-46.4	0.4	2.7	8.3	56.5

ANCILLARIES

Fig. 4

Many spreadsheets allow the preprocessing of the dataset almost to the finished csv file. It is wise to check it thoroughly before injecting it into the application.

A sample file of stable isotope composition in precipitation is included in the software distribution as working examples.[XXIX] See section 5.6 for further details on setup and section 15 for a brief description of the dataset.

5 Software installation and configuration

Timescape Global software distribution consists in an executable, pre-packed jar file, an Eclipse project, a MySQL file for the creation of the required tables, and some sample data. The first step to perform is the creation of the database (a dedicated database for each sample points dataset), than comes the configuration of the application. The jar as it comes out-of-the-box needs to be configured in order to be able to access the database.

The installation package consists in:

- the TIMESCAPEGLOBAL.JAR executable jar file
- a SQL folder containing the MySQL tables creation script and some sample data
- a SOURCE folder containing the source code, the ant build file, etc
- a configuration tool (TIMESCAPECONFIGURATOR.JAR, see 5.5) to assist the user during the application setup

5.1 Prerequisites

This application needs a separate database management system for data storage. The installation package comes only with the MySQL version of the scripts, this is so because of robustness, availability and support that characterize it. All the development and testing has been conducted using MySQL databases. The SQL script is as plain and standard as possible, so to allow an easy translation into other db flavors, should the user be interested in doing so.

The computer running the application must be equipped with a Java virtual machine, version 1.7 or above. There is not a minimum memory requirement, however, it is advisable to have a minimum of one gigabyte dedicated to the virtual machine.[XXX]

Three files need to be customized:

- HIBERNATE.CFG.XML database configuration. This is the only mandatory step

XXIX From the Global Network of Isotopes in Precipitation, see [6].

XXX The main panel of the application monitors the virtual machine status, while running.

- LOG4J.PROPERTIES logging configuration. Tell the application where to put the log file
- TIMESCAPE.PROPERTIES software fine-tuning, can be left unchanged

Two ways are at the user's disposal to configure the application: modify the precompiled jar or modify the source code and prepare him/herself a new jar. The simplest option is the configuration of the JAR, it is basically a ZIP archive that can be unzipped, modified, zipped again and renamed to .JAR.

5.2 Preparing the database

This is the critical step: there is a single table (VOXEL) that can host really a great number of records. Just a rough estimate: one thousand by one thousand cells, replicated for one thousand time sheets, gives one billion records. It is advisable to check the storage capacity of the database before embarking in the calculation of really bulky models, and to evaluate a few downscaled models first. In an ideal world, the database machine should be dedicated one, but a *localhost* implementation can do de job, too.

The user accessing the database needs to have *write* permissions on all of the tables. At setup time all tables are empty. Follow these steps to prepare for storage:

- create a database user for the application,[XXXI]
- create an empty database, or one for each dataset if more than one is involved
- create the tables executing the `timescape_global.sql` script
- assign/check the appropriate permissions to the user

5.3 Configuring Hibernate

Once the database is ready, it is time to configure the application in order to point to it. The data part of TimescapeGlobal is based on the *hibernate* framework: just the first few lines of the configuration file should be changed, indicating database flavor and access details. Edit the `hibernate.cfg.xml` file:

```
    ...
<property name="connection.url">URL</property>
<property name="connection.username">USERNAME</property>
<property name="connection.password">PASSWORD</property>
<property name="connection.driver_class">DRIVER</property>
<property name="dialect">DIALECT</property>
    ...
```

configuring the appropriate parameters[XXXII]:

- `URL`: the URL of the database resource, generally of the form `jdbc:flavor://host/dbname`
- `USERNAME` and `PASSWORD`: the database user login credentials
- `DRIVER` and `DIALECT`: the appropriate *hibernate* drivers configuration strings[XXXIII]

For example, the connection parameters for a MySQL database running on the same machine of the application (localhost), on a database named `mydatabase` with user `myuser` and password `mypass` will read:

```
    ...
<property name="connection.url">jdbc:mysql://localhost/mydatabase </property>
<property name="connection.username">myuser</property>
<property name="connection.password">mypass</property>
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    ...
```

As a general rule, it could be a good idea to have another database user, who can only read from the tables, for further analysis of the data.

5.4 Setting up application properties

A few more settings are in order before running TimescapeGlobal.

First, the logger of the application is to be set: edit the LOG4J.PROPERTIES file accordingly:

```
#logging directory
logdir=/configure/my/logging/directory
```

XXXI Root user on localhost can do the job, if one has no security concerns.
XXXII It can also be done via the configuration tool, see 5.5.
XXXIII See the hibernate documentation at *http://hibernate.org*.

```
#log file name
logfile=TimescapeGlobal.log

#configuration
log4j.rootLogger=INFO
log4j.logger.timescape=,timescape
log4j.appender.timescape=org.apache.log4j.RollingFileAppender
log4j.appender.timescape.File=${logdir}/${logfile}
log4j.appender.timescape.MaxFileSize=100KB
log4j.appender.timescape.MaxBackupIndex=10
log4j.appender.timescape.layout=org.apache.log4j.PatternLayout
log4j.appender.timescape.layout.ConversionPattern=%d{yyyy/MM/dd-HH:mm:ss.SSS} %-5p %m%n
```

The only parameter that should be set is `logdir`, pointing to any suitable place where the software can place its log file. Those with a thorough knowledge of the*log4j* logging framework can also modify other parameters according to their taste. As it comes out-of-the-box the application logs on a file named `TimescapeGlobal.log`, rolling of up to ten files of 100KB.

Other parameters can be configured to fine-tune the application. They are all contained in a single file: TIMESCAPE.PROPERTIES:[XXXIV]

```
window.title=TS Global
author.mailto=marco.ciolfi@ibaf.cnr.it

#earth radius (km) - long/lat in d.ddd
earth.radius=6367.4491

format.rounding=1000.
format.rounding.deg=1000.
stats.histogram.bins=12
stats.histogram.separator=_
stats.histogram.empty=..
variogram.histogram.bins=24
grid.nullvalue=-9999
r2.significance.star=.333
r2.significance.starstar=.667

model.default.neighbours=0
model.default.sheetsize=365
model.default.cellsize=1
model.default.convfact=1
model.default.causal=1
model.default.sourceprob=1
model.default.timestretch=0.5
model.default.weightpower=1

evaluator.threadkill.millis=1000

scriptengine.language=JavaScript

logger.logonfile=yes
```

These parameters have been carefully chosen to suit most application needs, but the expert user can change them accordingly to his/her needs. The casual user can skip the following list and simply keep the default values. For those interested:

- `window.title` is the title of the main window. It is a good idea to change it accordingly to the data used for interpolation, especially if more than one instance of the software are used at the same time.
- `author.mailto` the author's email. As a matter of principle, license terms do not allow changing this parameter, however if the user modifies the code and creates a new version of the application, it is advisable to change this parameter value accordingly.
- `earth.radius` is (you guessed) the Earth radius, in kilometers. Change it according to the planet of choice. The Earth shape is assumed to be spherical.
- `format.rounding` and `format.rounding.deg` control the number of digits after the decimal dot (1000 = three digits). They have only a cosmetic role, not affecting the number precision (double), for both database storage and export. The second parameter controls angular values in degrees.
- `stats.histogram.bins`, `stats.histogram.separator` and `stats.histogram.empty`: the number of bins of histograms, and the characters used to separate bins (default _) and to mark an empty bin (default `..`).

XXXIV It can also be done via the configuration tool, see 5.5, as is the case with the logging configuration.

- `variogram.histogram.bins`: the number of bins of the variograms, this parameter affects the lag size, but it is only a starting value, that the user can modify at will at runtime from 1 to 100. The default value of 24 is a good compromise between readability and completeness of information.
- `grid.nullvalue` is the no-data marker for ESRI grid exported files. Default value is -9999, change it only if the latter is an allowed value for the interpolated data. Some GIS applications do not allow a different value.
- `r2.significance.star` and `r2.significance.starstar` are the threshold values for showing an asterisk (*) or two (**) close to the R^2 value in some regression estimates. Default values are 1/3 (.333) and 2/3 (.667).
- `model.default` variables are the default values of the new models:

  - `neighbours`: number of nearest neighbours, 0 stands for no limit
  - `sheetsize`: time sheet size, default 365 (days, but any unit can be used)
  - `cellsize`: longitude/latitude cell size, default 1°
  - `convfact`: time to space conversion factor, default 1
  - `causal`: causal cone apex width expressed as space/time ratio, default 1 (45°)
  - `sourceprob`: sample point picking probability, default 1 (certainty)
  - `timestretch`: forecast duration with respect to time span of samples, default 0.5
  - `weightpower`: (negative) exponent of the weight function, default 1

- `evaluator.threadkill.millis` is the number of milliseconds to wait before destroying the evaluation threads, in case of abort. Default 1000 (one second). Do not change unless it is absolutely necessary to do so, in case of a serious bottleneck in database access times.
- `scriptengine.language` is the script language for all user-defined functions, default JavaScript. Be wise before considering changing this parameter.
- `logger.logonfile` is a yes/no flag controlling whether the application logs on file (yes) or on stdout (no), which is almost like no logging at all. default is yes.

5.5 Configuration tool

Timescape Global comes with a little configuration tool, the **Timescape Configurator**, to aid the setup whithout having to crawl into zip and jar files. This tool opens up the application jar, finds the three configuration files and shows them in three panels for easy editing.

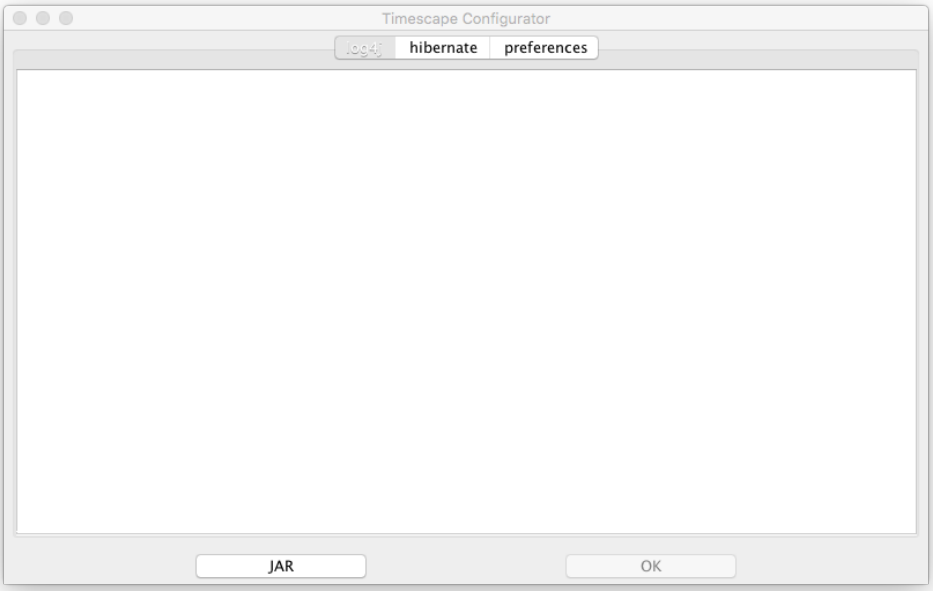


Fig. 5

Just select (pressing the `JAR` button) the source jar, then edit the configuration files as they appear in the editors. Warning: the configurator does not perform any test on the actual content of the modified configuration files. It is exactly like changing their content by hand, faster and easier, but not safer at all.

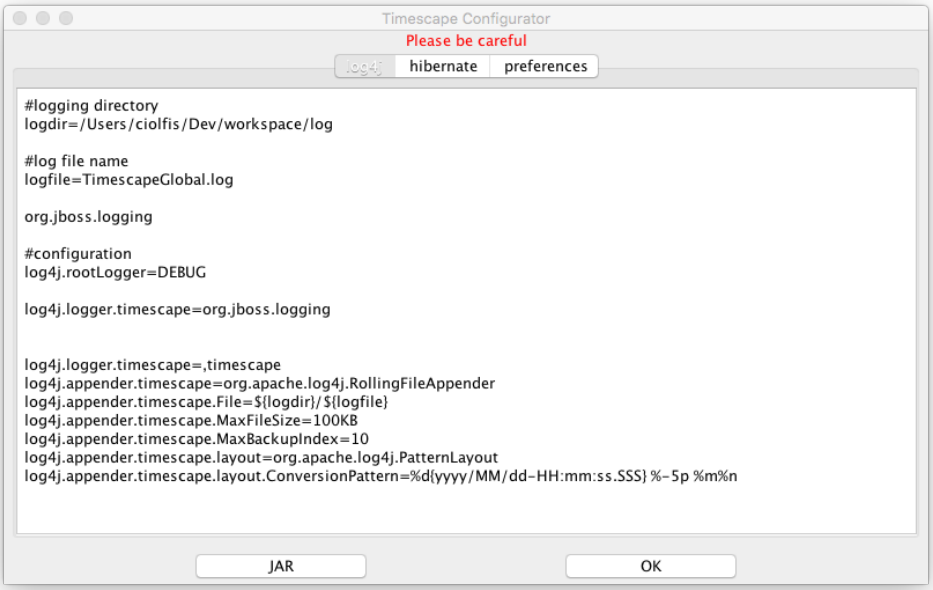


Fig. 6

Press `OK` to select a destination `jar` file (not the same as the original one). The configurator carbon-copies Timescape Global to the new executable `jar`. That's all.

5.6 Running Timescape Global for the first time

On the first run, upon finding an empty database, **Timescape Global** will open the following dialog window, asking for a formatted dataset.

It is possible to paste a formatted text or to let the application grab the text from a file, than clicking the `PARSE & INSERT` button the text is parsed and the data inserted. the text shown at the opening of the window is a short reminder of the format of the text.

The first fields (`ID, TIME, LONG, LAT, VALUE`) are mandatory and cannot be null. Then, the user can indicate a certain number of ancillary fields, the latter can also be null. The formatting rules are:

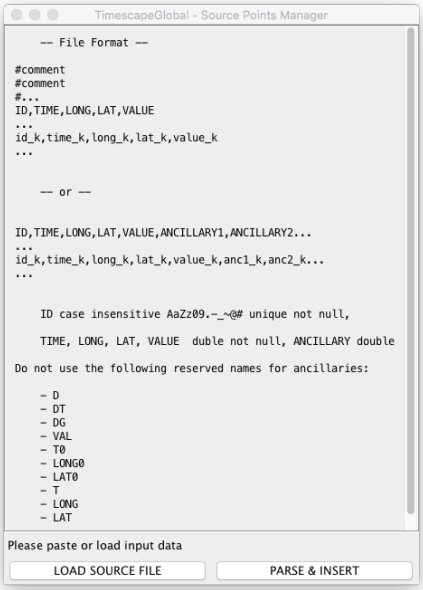


Fig. 7

- the file can start with any number of comment lines, starting with an `#`.
- the first line must be `ID,TIME,LONG,LAT,VALUE` if there are not ancillary fields
- if there are ancillary fields, their names should be indicated int the starting line, immediately following `VALUE`, also separated by commas:
  `ID,TIME,LONG,LAT,VALUE,ANCILLARY_FIELD_1, ... ,ANCILLARY_FIELD_N`
  the characters allowed for fields names are `azAZ09-_` no spaces, no repetitions.[XXXV]
- now data lines follow. fields are comma-separated and must obey the following rules:
  - the `ID` field is the unique, not-null identifier of the sample point. Allowed characters are `azAZ09-_.~@#`, no spaces. It is better to be more conservative and limit the characters to `azAZ09_-`
  - the `TIME` field contains the time variable value (double, dot as decimal separator, no thousand separator, scientific notation allowed, no null value allowed)

XXXV The following strings cannot be used as ancillary names `D, DT, DG, VAL, T0, LONG0, LAT0, T, LONG, LAT`: they are reserved names used in custom functions calculation.

  - the `LONG` field contains the longitude in decimal degrees, from `-180` ($180°W$) to `+180` ($180°E$), same format as `TIME`, no null value allowed
  - the `LAT` field contains the latitude in decimal degrees, from `-90` ($90°S$) to `+90` ($90°N$), same format as `TIME`, no null value allowed
  - the `VALUE` field contains the value of the sample measure, same format as `TIME`, no null value allowed
  - `ANCILLARY_k` fields contain the values of the ancillary variables, in the same order of the starting line, of course, null values allowed (just leave the field empty)
- nothing else is allowed in the text to be parsed

During the parsing procedure, the text is substituted with the results of parsing.

The database population window can be accessed anytime by clicking four times on the lower right icon of the main window of the application.

Warning: parsing a new sample dataset deletes any model, data and all! There is no way to recover anything erased afterward.

The tables interested by this procedure are SOURCE, which contains ids, coordinates and values, and is always populated, and the ANCILLARY and ANCILLARY_SOURCE tables, that contain ancillary field names and values.

Upon closing this database population window, the software shifts into the main window that, from now on, will be the crossroad of all user interaction.

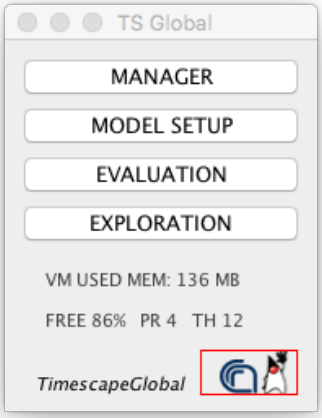


Fig. 8

6 Timescape Global Application

The structure of **Timescape Global** is centered on the concept of *model*: the user creates his/her models starting from the same sample points.[XXXVI] The samples' dataset structure is described in detail in the installation section of this manual. a model is a collection of pseudo-volume[XXXVII] elements (*voxels*) interpolated from the samples dataset according to the set of rules of inference that define the model itself.[XXXVIII] Basically, the user defines a model's boundaries, both in space and in time, and the rules that the interpolator must follow to calculate the values of the voxels. The interpolation follows the footsteps if the IDW (Inverse Distance Weighted [mean]) in its general procedure, adding an ample set of customization parameters which can give a real distinctive flavor to each model. It is also possible, through the definition of custom functions, to overcome a few known limitations if the IDW approach to geostatistical modeling.

There is no "interaction" among voxels: this means that every element is evaluated from the sample dataset, given the interpolation rules, regardless of its neighborhood. This is in a broad sense reductive, but has its strong sides: the computing power needed is kept to a minimum, so that any respectful PC or laptop can do the job and, above all, a bulky model can be partitioned in a set of smaller, more easily manageable ones.

As said, the computing power is not a particularly critical issue, but the data storage is. Typical geostatistical models scale with the square of the order of magnitude of their components (i.e. a 1000×1000 image weights one million pixels), while three-dimensional models scale with the cube of dimension (repeating 1000 times the quoted image we would have a billion voxels model). Care should be given to avoid an unnecessary growth of the models size. Although the voxels storage is on a separate database, consider that the visualization operations can get painfully slow with bigger models (especially with many space cells) and that the exporting billions of records from a database can be critical too.[XXXIX] Database limitations in terms of number of records and file size should be kept in mind before attempting to build a really bulky model. On the other hand, one can always build a coarse-grained model, spanning large space and time intervals, then focus to a few interesting spots, making fine-grained, localized models. A specific model cloning function is available for this purpose.

The software-user interaction gravitates around the **main panel** which gives access to **the manager panel**, the **setup panel**, the **evaluation panel** and the **exploration panel**. As their names suggest, each panel is devoted to a single task of the model workflow:

- A model has to be created from scratch in the **manager panel**. A fresh model is assigned some parameters default values. It is also possible to clone an existing model.
- A model is created (or cloned) without voxels. The first step is to adjust the model parameters through the **setup panel**; this panel eases the user's task with the aim of graphical and statistical tools. Statistics can be saved for future reference.

XXXVI One sample points dataset per single installation of TimescapeGlobal. All the models must make use of the same dataset, possibly excluding some of the points.

XXXVII It is not, strictly speaking, a volume because we are mixing two space and one time dimensions; on practical grounds, however, there is not much of a difference.

XXXVIII See the algorithm section for mathematical subtleties.

XXXIX The software allows the exporting of georeferenced images, grid layers and time series, plus a variety of standard statistical indicators. The user can benefit of direct access to the database for his/her specific needs.

- After configuration, the model is ready for the evaluation[XL] that takes place in the **evaluation panel**. The actual evaluation is preceded by the insertion of empty voxels for database efficiency.
- A finished model can be examined in the **exploration panel**, that offers comprehensive descriptive statistics, model sheet slicing and core digging, and extensive exporting features through images, GIS layers and csv files.

This section describes in detail the main and manager panels only. The other panels are described in detail in dedicated sections.

6.1 Main panel

The main panel is opened when launching the application (if the database is not empty, see the installation section for details). This panel consists just of a set of buttons ordered top-down according to the workflow sequence. At the first run the setup, evaluation and exploration buttons are disabled because there is no model defined yet. Under the buttons, two text lines show the status of the Java VM: this information[XLI] is provided (to the Java *connoisseurs*) to monitor the health of the Virtual Machine.

Two more buttons are hidden beneath the bottom icons, they are accessed by clicking them four times (like a double-double-click). These functions should not be used in normal conditions, and are provided as an escape lane just in case something goes wrong. Four-clicking the *TimescapeGlobal* label opens up the **lock-/unlock panel**, while four-clicking on the rightmost icons gives again access to the **database panel**. The latter is the same panel used in the configuration of the application. Warning! **loading a new sample dataset deletes everything stored in the database**, so use this function only if this is what you want. For usage instructions follow the directions of section 5.6.

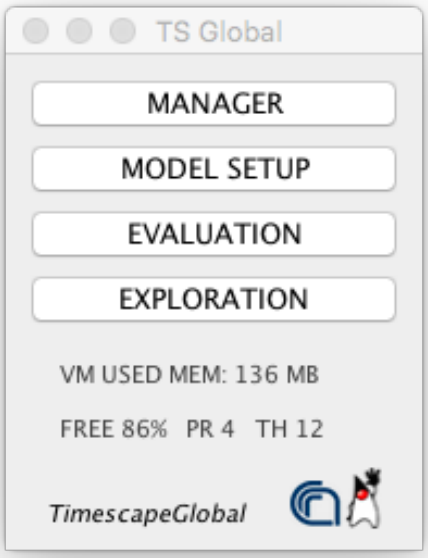


Fig. 9

6.1.1 Lock/unlock hidden panel

– Skip this until you are not in trouble –

Every model is equipped with a pair of *status flags*, indicating whether a model is being evaluated or is completed. The user is not directly aware of these flags, used internally to protect the models from unwanted or improper access. The flag policy follows this directions:

- A fresh model has both flags off. Its parameters can be modified.
- A model that is being evaluated is locked (lock flag on). It cannot be modified.
- A model that has been evaluated is complete (lock and complete flags on). It cannot be modified nor evaluated again.

The hidden panel allows one to change the status of such flags anytime. As a matter of principle, such panel should be completely useless, but sometimes something that can go wrong goes even worst. In such a case, this panel allows one not to lose all the configurations made during the setup.

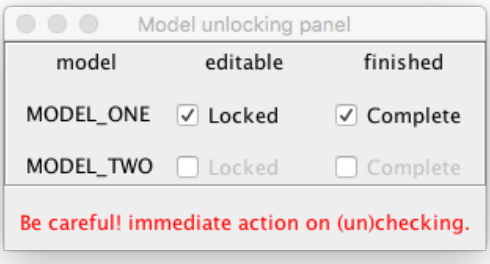


Fig. 10

XL A model is already evaluable as it is created/cloned, according its default parameters.

XLI The parameters shown are: the Virtual Machine memory size VM USED MEM, the free memory in percent FREE, the number of processors seen by the virtual machine PR and the number of threads TH. It is a good practice to monitor the application via this panel, together with other monitoring tools, especially with regard to disk usage and database transactions.

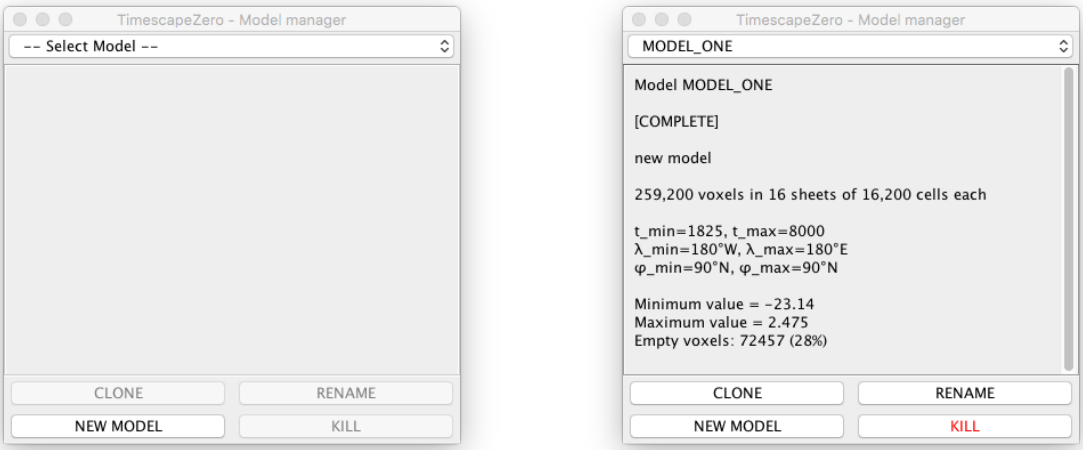
Acting on the checkboxes, one controls directly the status of the flags:

- **Checking Locked** locks a model: It cannot be modified.
- **Unchecking Locked** unlocks a model: It can be modified. NEVER do so while a model is being evaluated.
- **Checking Complete** makes the model unevaluable, even if it has not been evaluated beforehand.
- **Unchecking Complete** makes the model evaluable, regardless of having been already evaluated. Warning: it also DELETES ALL THE VOXELS of the model.

Any action performed on these checkboxes takes place immediately. It is advisable to resort to this panel as a last hope, to recover from a crash of the system or some other equally exceptional event. The status of the flags controls in turn the status of the buttons of the main panel: the evaluate button, for example, is enabled only if there are evaluable models; the explore button is enabled only when we have complete models.

6.2 Manager panel

This panel allows the user to create, rename,[XLII] clone and delete models.



(a) **(b)**

Fig. 11

When **Timescape Global** is run for the first time, the manager panel looks like the one on the left: all one can do is creating a new model. Just click on the NEW MODEL and give it a name.[XLIII] The name will be converted to all-uppercase, the spaces are changed in underscores (_), spaces before and after the name are trimmed. Name conventions allow for letters, numbers, spaces and a few special characters (that the wise user will never type).

On top of the manager panel there is a list of selectable models, it lists all the models in the database, regardless of their status. Upon selection of an item, the main text window of the panel fills up with some details about the status of the model, and the buttons on the bottom of the panel become active. The user can:

- **Rename** a model: it pops-up a tiny window to type the new name. Should the name be already used for another model, the field turns red and one cannot change the name until a new one is chosen.
- **Clone** a model: this generates a fresh, identical copy of the model, with the same name followed by _CLONE (and a progressive number if needed). The model is unlocked and not complete, ready to enter the workflow.
- **Kill** a model: it does exactly what it is supposed to do, it deletes the model (for the sake of safety, a confirm request appears, twice if the model is complete).

There can be only one manager window per instance of the software. The user can open or close it at his/her own will: the topmost button of the main window allows to open it again.

7 Model setup

A newly created model parameters (see 6) are chosen according to the space-time distribution of the sample points, in particular we have:

- all sample points are associated to the model

XLII Renaming a model is harmless, since the actual ID is a numerical one, invisible to the user. Anyway, two models cannot share the name.
XLIII A small window pops-up to type the name.

- the space bounds $[\Lambda_m, \Lambda_M] \times [\Phi_m, \Phi_M]$ contain exactly all the sample points
- the time bounds $[T_m, T_M]$ contain all the sample points, plus a forecast region spreading one half of the samples time distribution: $T_m = \min_t\{t_k\}$, while $T_M = \max_k\{t_k\} + \frac{1}{2}\left(\max_k\{t_k\} - \min_k\{t_k\}\right)$
- the cellsize w_c is set to 1°
- the time sheets' separation w_s is set to the value chosen in the preferences (see 5.4)[XLIV]
- all the interpolation parameters are set to default IDW

The SETUP panel opens up with a drop down list of available models. Pick one.

7.1 Model setup panel

As a matter of principle, a model could be evaluated as it is comes out of the box just after creation, but certainly the user will change many parameter according to his/her needs. This is accomplished through the MODEL SETUP tab of the SETUP panel. The other tabs (SAMPLES TREND and VARIOGRAM) display helpful statistical informations to help the user tune the model.

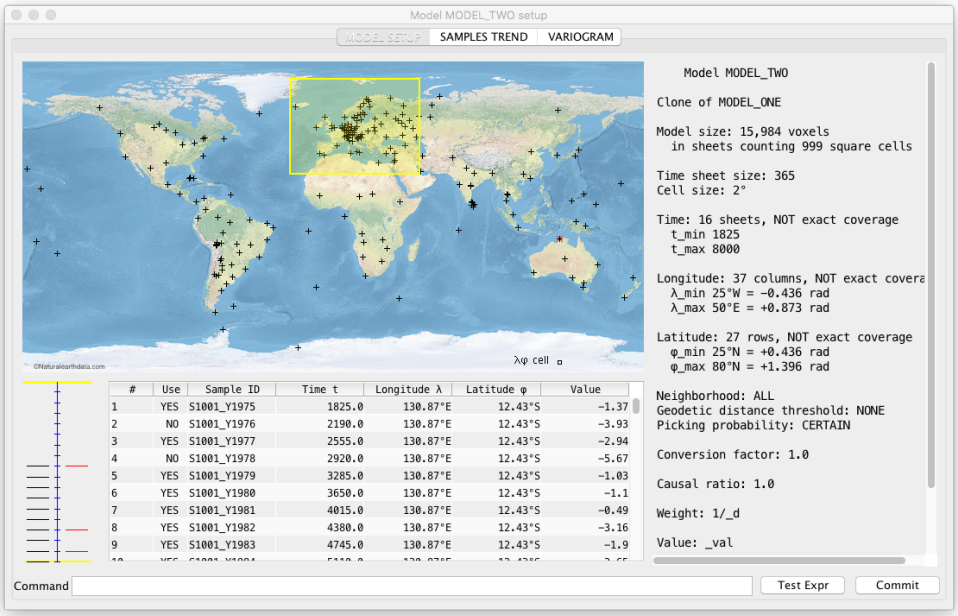


Fig. 12

The MODEL SETUP tab has a few areas for the user interaction: tho modify a parameter value the user types a command in the **command area**. The command is parsed and the appropriate changes made.

XLIV Default value is 365, supposing the time is measured in days from a given date.

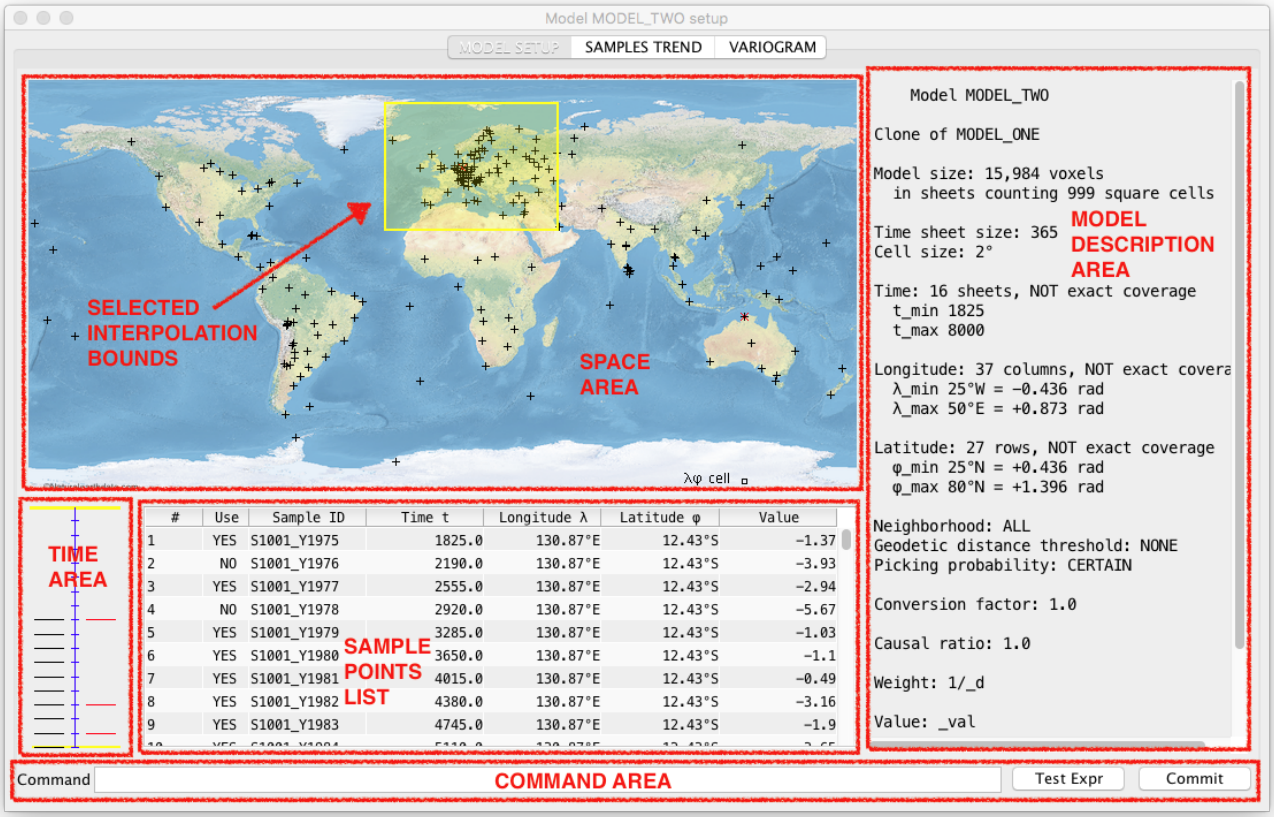


Fig. 13

If the parsing fails or if the parameter value chosen is not acceptable, the command is not applied and the typed text turns red, to alert the user. Other important areas are the **model description**, **space** and **time** areas, and the **samples list** area. These describe all the information needed in order to setup the model.

All the user's changes happen in a "sandbox" environment, so that nothing is applied to the actual model until the `Commit` button is pressed. The user can quit at any time the setup procedure discarding the changes performed (a confirmation dialog is shown).

The **space area** depicts a world map[XLV] with all the samples placed according to their coordinates. The $[\Lambda_m, \Lambda_M] \times [\Phi_m, \Phi_M]$ area chosen by the user is also shown. The cellsize w_c is displayed in the bottom right corner. If a sample is used in the model it is plot with a black $+$ mark, otherwise a red $\times$ marks unused samples. The map size is 720×360 pixels, corresponding to a 2×2 square per degree cell.

The **time area** displays the time distribution of the samples; the central blue bar shows the time sheets that compose the model, on its left side there are the used (black) samples, on its right side the unused (red) ones. Two thick yellow lines mark the $[T_m, T_M]$ interval.

The **sample points list** area consists in a table of all the allowable samples. A model can be evaluated from any number of these points. A `YES` - `NO` flag close to the sample name tells the user whether such sample is used or not.

The **model description** area shows all the relevant parameters of the model while the user is changing them; these consist in both geometrical information (space and time bounds, displayed first), algorithm details in the middle of the description, and allowable variables at the bottom of the area.

7.1.1 Command syntax

All the commands are typed into the **command area**. They are parsed immediately on pressing the `ENTER` key. A successful command is applied to the model (in the sandbox), in case of parsing failure, the command is displayed in red, to catch

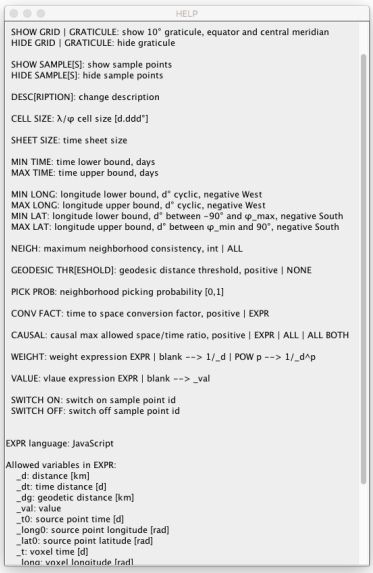


Fig. 14

XLV The base image comes from *http://www.naturalearthdata.com*.

the user's attention. All commands are case-insensitive. The first and most useful command is `HELP`. It pops-up the `HELP PANEL`, that shows a list of all the allowed commands.

7.1.2 General commands

- `DESC` or `DESCRIPTION` followed by a descriptive text: changes the description of the model. This field is just a reference for the user.[XLVI]
- `SHOW SAMPLE` or `SHOW SAMPLES`: shows the samples sites over the world map. The time area is not affected.
- `HIDE SAMPLE` or `HIDE SAMPLES`: hides the samples.
- `SHOW GRATICULE` or `SHOW GRID`: shows a $10° \times 10°$ grid above the world map. The equator and the central meridian are thicker.
- `HIDE GRATICULE` or `HIDE GRID`: hides the grid.

7.1.3 Geometry commands

These commands define the geometrical boundaries of the model:

- `CELL SIZE <value>` changes the value of the cell size w_c. The value is expressed in decimal degrees $d.ddd$.
- `SHEET SIZE <value>` changes the value of the time sheet interval w_s. The value is expressed in arbitrary units.
- `MIN LONG <value` changes the value of Λ_m (minimum longitude). The value is expressed in decimal degrees $d.ddd$.
- `MAX LONG <value` changes the value of Λ_M (maximum longitude). The value is expressed in decimal degrees $d.ddd$.
- `MIN LAT <value` changes the value of Φ_m (minimum latitude). The value is expressed in decimal degrees $d.ddd$.
- `MAX LAT <value` changes the value of Φ_M (maximum latitude). The value is expressed in decimal degrees $d.ddd$.
- `MIN TIME <value` changes the value of T_m (minimum time). The value is expressed in arbitrary units.
- `MAX TIME <value` changes the value of T_M (maximum time). The value is expressed in arbitrary units.

A note about geographical coordinates: the latitude value must be between -90 (90°S, the South pole) and +90 (90°N, the North pole); southern latitudes are negative. As the following figure demonstrates, the minimum longitude Λ_m can exceed the maximum Λ_M: in this case the area covered by the interpolation is wrapped around the $\lambda = \pm 180°$ meridian.

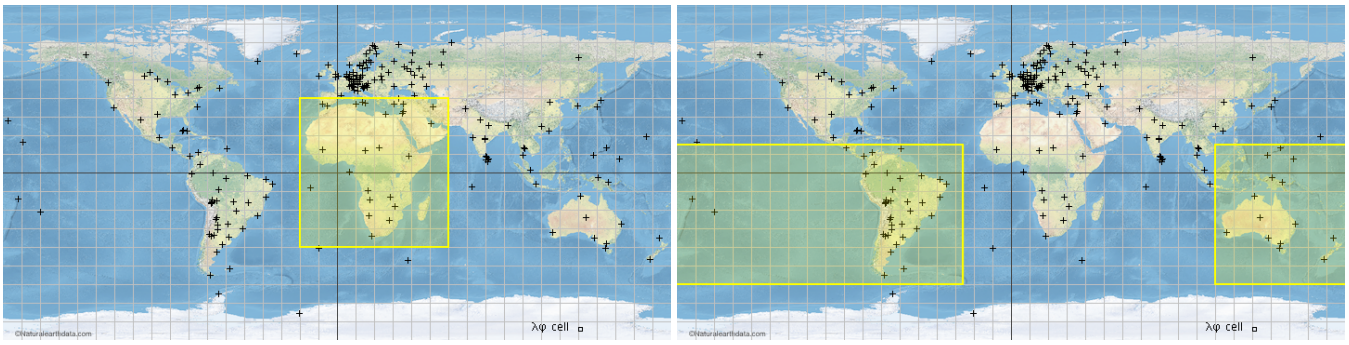


Fig. 15

Note also that the effective coverage of the set of voxels is always contained within the above limits, but it can actually be a slightly smaller subset. This is because the w_c and w_s constraint are kept fixed so that the effective upper limits of the intervals are recalculated in order to have a whole number of cells and sheets. The user is notified whether the coverage is exact or not in the model description.

The number of cells per sheet, the number of sheets and the total voxel count is shown in the first lines of the model description.

7.1.4 Sample points selection commands

The set of samples is a property of the entire application. Models. however, do not have to share all the sample points among themselves. The user can choose at will which points associate with any model,[XLVII] leaving the others unused.

The commands are:

- `SWITCH ON <sample id>` uses such sample.
- `SWITCH OFF <sample id>` excludes such sample in the interpolation.

There is also a shortcut, if many points need to be excluded from interpolation: just double-click the `YES NO` flag on the left of the sample id (the **Use** field of the list) to swap its status.

XLVI The name of the model can be changed only in the `MANAGER PANEL`, see 6.

XLVII This allows some *jackknifing* to be done on a bunch of otherwise identical models, to check for bias and stability of the interpolation.

7.1.5 Algorithm commands

This is the preeminent and most complex section of the model setup. Here the user defines all the relevant parameters and functions for the evaluation of the voxel values. The newly created model interpolation technique is set to plain IDW. Sure one wants some more.

The customization of the algorithm involves the tuning of some parameters (number of prime nears, geodesic threshold, picking probability), the definition of time's role and causality (conversion factor and causal cone) and finally the definition of the weight and value functions. See section 3 for details. The applicable commands are:

- `NEIGH <number>` changes the number of prime nears (the maximum number of points used in interpolation) to `<number>`.
- `NEIGH ALL` removes the neighborhood check using all selected sample points. This is also the default option.
- `GEODESIC THRESHOLD <number>` or `GEODESIC THR <number>` defines a maximum distance (as calculated on the Earth surface) for using a point in the interpolation.
- `GEODESIC THRESHOLD NONE` or `GEODESIC THR NONE` eliminates the threshold, all points allowed (default option).
- `PICK PROB <value>` sets the value of the probability of a point to be picked for the interpolation.[XLVIII] Default value is 1 (certainty).
- `CONV FACT <value>` sets the (fixed) value of the time to space conversion factor, the default value is 1.
- `CONV FACT <expression>` defines the expression that gives the value of the time to space conversion factor. This expression is a function space and time coordinates of the voxels and the samples (see 7.2 for details).
- `CAUSAL <value>` sets the value of the aperture of the causal cone (see 3.3) as the allowed space / time distance. Default value is 1. The combined effects of this parameter, together with `CONV FACT`, give each model a distinctive causal character.
- `CAUSAL ALL` sets an infinitely-broad causal cone, i.e. a given source point, in principle, can influence any voxel in its future. Use this option as a first guess if you have no idea of a causal pattern in the phenomenon under scrutiny.
- `CAUSAL ALL BOTH` releases the time-forward-only interpolation constraint, so that a backwards action is possible. Setting this value makes the time a third space dimension in all respects.
- `CAUSAL <expression>` defines the expression that gives he value of the causal cone aperture (space to time ratio) as a function of space and time coordinates (see 7.2).

7.1.6 Weight and Value commands

Once we have selected the relevant source points and assigned them a distance and a causal role, it's time to use these points to interpolate the values of our voxel. This is done with a generalization of the IDW algorithm, so we have to define a weight function ad also a value function which can be the simple value of the source point or a more complex function that includes ancillary data too.

The commands are:

- `WEIGHT` sets the weight function as simply $w = \frac{1}{d}$ (default option).
- `WEIGHT POW <number>` sets the weight function as $w = \frac{1}{d^n}$ where n is the user's `<number>`.
- `WEIGHT <expression>` sets the weight function accordingly to the user input (see the following section for details).

Given these functions, the value v of each voxel is evaluated as:[XLIX]

$$v = \sum_k \bar{w}(s_k, v) f(s_k, v) \quad \text{where} \quad \bar{w}(s_k, v) = \frac{w(s_k, v)}{\sum_k w(s_k, v)}$$

Where $\{s_k\}$ is the set of source points causally connected with the voxel under evaluation, $w(\cdot)$ is the weight function and $f(\cdot)$ is the value function defined above. This is basically a weighted sum, but don't be misled by the supposedly simple expression: $w(\cdot)$ and $f(\cdot)$ can be **any function** of time, space and source values, including ancillary fields, if included in the sample dataset. The user can include, for example, an harmonic time component simply weighting the average with a sine or cosine tuned with appropriate frequency and phase.

For example, the user code[L] `(1 + Math.sin(2 * _dt) / (_dg * _dg + 4)` becomes[LI]

$$\frac{1 + \sin(2c\Delta t)}{D^2_{\lambda\varphi} + 4}$$

7.2 User-defined functions

Timescape Global users can define their own functions for interpolation. There is a constraint, however: the software performs a weighted sum of somehow chosen nearest points' values. This said, the user is free to define his/her function for both the weight

XLVIII This is the only random phase of the otherwise deterministic algorithm (see section 3 for details. It can be used to see how the randomization of the samples affects the results.

XLIX $\bar{w}(s_k, v)$ are the normalized weights.

L Note that all variables are preceded by an underscore (_) mark.

LI Δt is the time from source to voxel, $D_{\lambda\varphi}$ is the geodesic distance.

an the values being summed. It is also possible to alter the causal constraints which shape the causal cone (see section 3), but it requires some care.

The functions must follow the syntax of Javascript language[LII] with the convention that variable names begin with an underscore (_). User can use the following variables, related to the space-time distribution (GEOmetry) or to the values of the sample points (VALue):

- `_dt` [GEO] is the time distance $D_t = |c(t-t_0)|$ between voxel and sample point
- `_dg` [GEO] is the geodesic distance $D_{\lambda\varphi}$ between voxel and sample point
- `_d` [GEO] is the distance D between voxel and sample point $\sqrt{D_t^2 + D_{\lambda\varphi}^2}$
- `_t0`, `_long0` and `_lat0` [GEO] are the sample point coordinates $(t_0; \lambda_0, \varphi_0)$, longitude and latitude are expressed in radians
- `_t`, `_long` and `_lat` [GEO] are the voxel center coordinates $(t; \lambda, \varphi)$, longitude and latitude are expressed in radians
- `_val` [VAL] is the value of the sample (cannot be *null*)
- `_xxxx`, `_yyyy`... [VAL] a sample can have any number of ancillary variables associated. Ancillary values can be *null*; in this case the user function can fail to be evaluated

Most of the useful Javascript functions can be accessed by the Math object, including:

- mathematical constants, like `Math.E`, `Math.SQRT2` or `Math.PI`
- trigonometrical functions: `Math.sin`, `Math.cos`, `Math.tan`
- inverse trigonometrical functions: `Math.asin`, `Math.acos`, `Math.atan`, `Math.atan2`[LIII]
- power and exponential functions: `Math.pow`, `Math.exp`, `Math.log`
- `Math.min` and `Math.max` which calculate the extremal values of any number of arguments

Spaces can be used to improve readability when typing the functions expressions, but they are removed before parsing, and they are not stored into the database, so no spaces appear in the description of the model.

Other than `Math`, Javascript offers the powerful ternary operator `?:` that is very useful for branching the function flow, so giving some *intelligence* to the user expressions.

The syntax of the ternary operator is:

```
<test> ?  <true_result> :  <false_result>
```

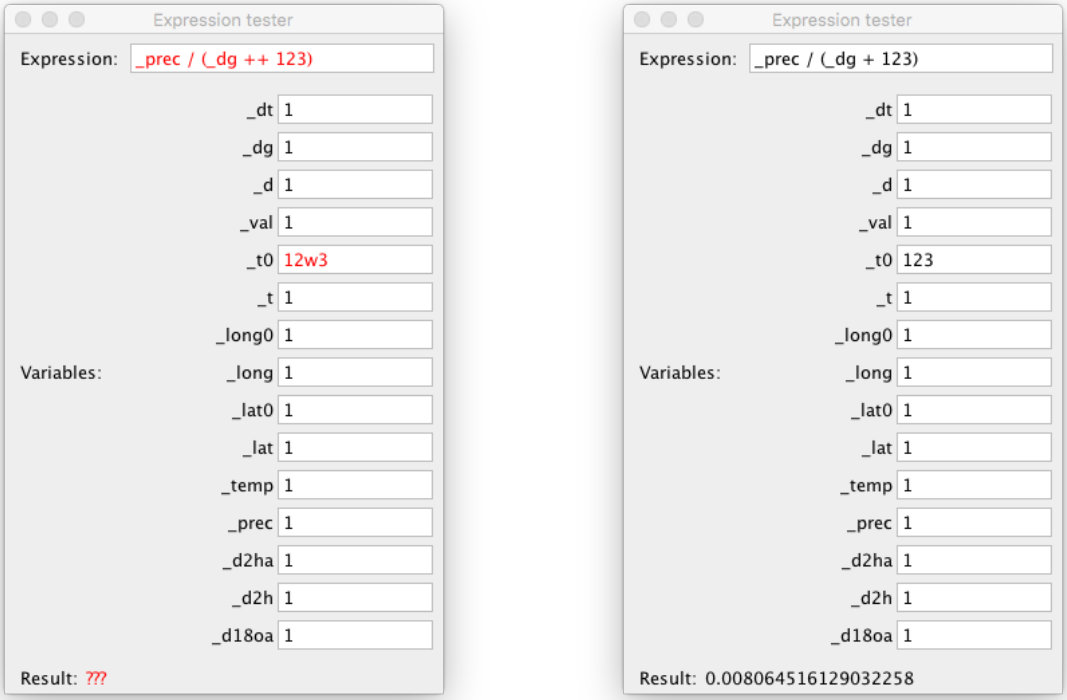
e.g. `(_val<0)?12.34:56.78` gives 12.34 if the value is negative or 56.78 otherwise. The ternary operator can be used to mimic the Heaviside step function:

$$\theta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

simply typing `(_xxxx<0)?0:1`, where `_xxxx` is the argument of the θ.

The command parser will not accept malformed functions, but no checking of the results is performed until the evaluation of the model.[LIV] There is an expression testing tool to check the outcomes of the user defined functions against some known values. This tool is accessed by pressing the `Test Expr` button on the lower right corner of the setup panel, close to the `Commit` button.

LII More correctly it is *ECMAScript*, which is interpreted in Java by the javax.script.ScriptEngine class.
LIII `Math.atan` is a single-argument function, while `Math.atan2` takes two arguments and calculates the inverse tangent of their ratio.
LIV All evaluation errors are shown to the user during the evaluation, they are also logged in detail.



(a) **(b)**

Fig. 16

The testing tool presents the user with an expression field and all the available variables. To check an expression, simply type it in the `Expression field` and assign suitable values to the variables. All the ancillary fields can be used, too.

An expression with a wrong syntax is highlighted in red (as it is the case for all the variables, which must be real numbers). If the expression can be evaluated, the result is shown in the `Result` field. Change the variables values to see if the function behaves as expected.

A note of caution: user defined functions are interpreted at run-time, so they are much slower than ordinary compiled Java code. The software is capable of using compiled code for the default options and for the simplest cases of inverse-power weight functions, but it switches to the Javascript interpreter in all other cases.

7.2.1 Functions for weight and value

This makes the actual difference from plain IDW to a complex pattern description. It is up to the user to find a suitable couple of functions for the value and the weight, to be able to replicate the patterns of evolution that characterize the modeled phenomenon. This is also the phase when ancillary data come into play.

Recall how a voxel value v is evaluated:[LV]

$$v = \sum_k \bar{w}(s_k, v) f(s_k, v) \quad \text{where} \quad \bar{w}(s_k, v) = \frac{w(s_k, v)}{\sum_k w(s_k, v)}$$

Suppose, for example, that one wants to correct the value function adding a contribution taken from the ancillary variable `_xxx` (say half of the absolute value), but only near the equator (within ± 0.2 radians):
`_val+((_lat>.2)?0:1)*((_lat<-.2)?0:1)*Math.log(_xxx)/2:`

$$\text{value} = v + \frac{\theta(\lambda - 0.2)\,\theta(\lambda + 0.2)\,\log(\text{xxx})}{2}$$

Note the use of two ternary operators to pick only the points close to the equator. It is not easy to read such functions, but some practice will help. Do not hesitate to put extra parentheses in the expressions in case of doubt about the order of execution of the operations.

If one needs to incorporate a periodic behavior into the weight function, putting at the same time a lower bound of, say, 10 on

LV See section 7.1.6. See also section 3 for details on the algorithm.

the values, one possibility is[LVI]

$$\text{weight} = \sin(\Delta t) \longrightarrow \texttt{Math.sin(\_dt)}$$

$$\text{value} = \max(v, 10) \longrightarrow \texttt{Math.max(10,\_val)}$$

7.2.2 Shaping the causal cone

The concept of *causal cone* plays a central role in Timescape Global. It sorts the causally connected sample points to be used to interpolate any voxel value. The user can choose how wide such a cone actually is (see section 3.3): The parameters that control the width of the cone are c (the time-space conversion factor) and k (the aperture of the cone expressed as height/radius ratio.[LVII] From one extreme case (infinitely broad cone) to the other (cone contracted to a its axis) there is a full range of choices. The most common one consists in choosing a fixed value for k, but it is possible to modify this behavior including ad hoc functions. For example, the user may not like an indefinite widening of the cone over time, so he/she wants to impose a maximum allowed value, a threshold T:

$$k = \min\left(T, \frac{D_{\lambda\varphi}}{D_t}\right)$$

that translates to[LVIII] `Math.min(T,_dg/_dt)`. Or, the user might need to slow down the growth of the cone with a logarithm:

$$k = \log\left(1 + \frac{D_{\lambda\varphi}}{D_t}\right)$$

that is implemented as `Math.log(1+_dg/_dt)`. One more example: the cone is deformed to a cylinder, with radius R:

$$k = \begin{cases} \frac{D_{\lambda\varphi}}{D_t} & \text{if } D_{\lambda\varphi} \le R \\ 0 & \text{otherwise} \end{cases}$$

i.e. `(_dg>R)?0:(_dg/_dt)` so that confronting $D_{\lambda\varphi}$ with $D_t \times \frac{D_{\lambda\varphi}}{D_t}$ gives $D_{\lambda\varphi} \le D_{\lambda\varphi}$, which is trivially true unless $k = 0$. A less elegant but equally working solution could be `(_dg>R)?0:1000000` or any other large number (larger than any reasonable distance that can be found between model's elements); this is also faster to execute.

Also the conversion factor c can be modified according to the user needs. It can be made time- or space-dependent (or both).[LIX] As an example, consider a conversion factor that depends on the latitude of the sample point φ_0 and the time of the voxel t, like

$$c = \sqrt{|\varphi_0|} + \log(1+t)$$

that translates to `Math.sqrt(Math.abs(_lat0))+Math.log(1+_t)`. This gives a value of D_t

$$D_t = \left[\sqrt{|\varphi_0|} + \log(1+t)\right](t - t_0)$$

so that the causal condition reads

$$D_{\lambda\varphi} \le k(t_0, \lambda_0, \varphi_0; t, \lambda, \varphi)\left[\sqrt{|\varphi_0|} + \log(1+t)\right]|t - t_0|$$

which shows all the possible dependencies of k (and of c).

7.3 Samples trend panel

This panel shows some trend statistics about the source points values. This information has nothing to do with the model per se, but it can be very helpful, suggesting a trend of the values against space (and time) and also with respect to the ancillary variables, if any.

LVI The interested reader can see [7] as a script language resource.

LVII The apex angle α is given by $\tan\frac{\alpha}{2} = \frac{\_dg}{\_dt}$, where `_dg` is the geodesic distance and `_dt` is the *time distance*.

LVIII T must be the actual value, no variables are allowed other then those already listed.

LIX Remember that c and k can be dependent on the geometry of the model, but not on the ancillary variables.

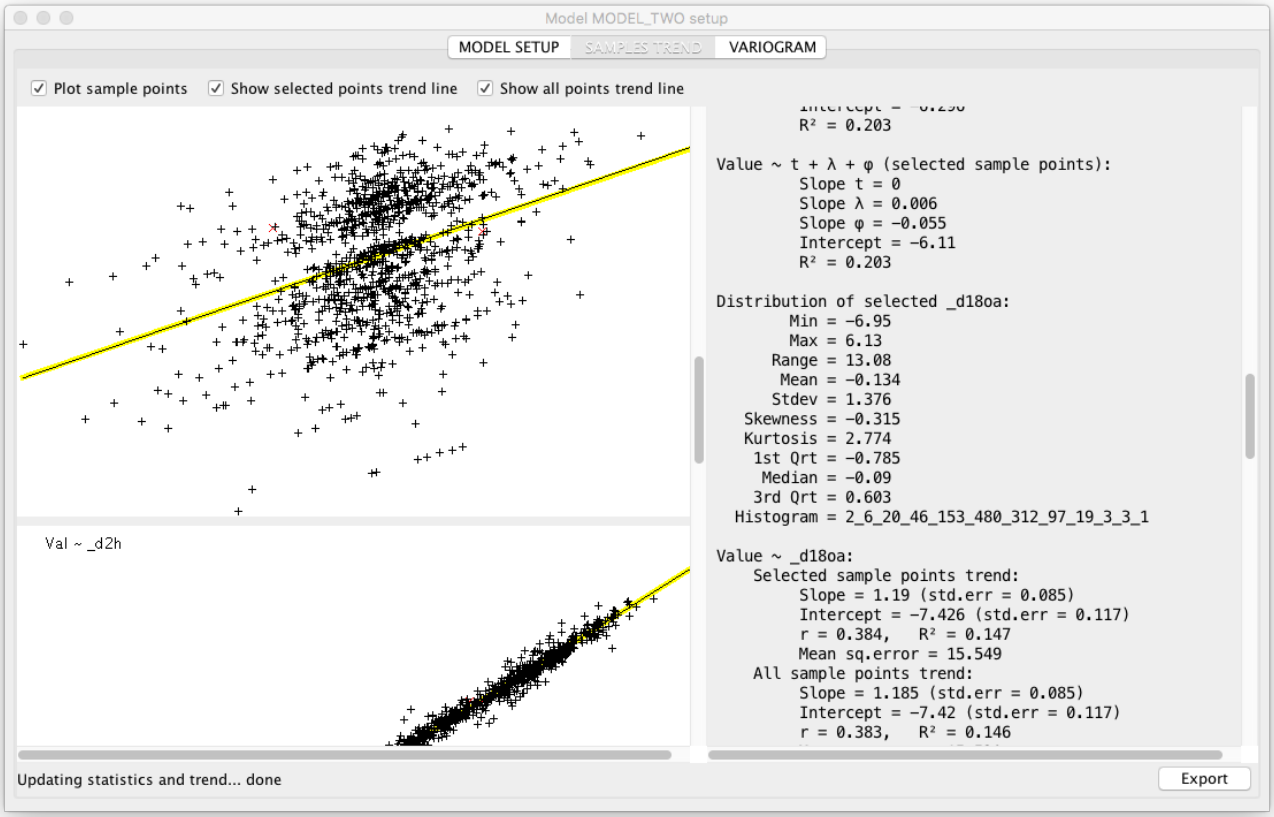


Fig. 17

There's no much user interaction here. On The left the panel shows a series of graphs, plotting samples values vs longitude, latitude, time[LX] and the ancillary variables. On the right there are the corresponding linear or multilinear regressions. Statistics include:

- minimum, maximum and mean values
- standard deviation, skewness and kurtosis
- 25, 50 (median) and 75 percentiles
- distribution histogram[LXI]
- slope, intercept, r and R^2 of the linear regressions

The regression parameters are evaluated using only the selected points and also with all the sample points, for reference. Any time the user switches on or off a sample point in the setup panel, graphs and statistics are refreshed. The graphs show the points distribution and the linear regression curves.[LXII]

7.3.1 Trend export

The trend statistics and the corresponding graphs can be exported. pressing the `Export` button a dialog appears to let the user choose the export folder. The output is a folder containing a text file (a copy of the statistics) and a `.png` image for each trend plot. The default name of the folder is `<model name>_TREND`; all the files it contains are labelled accordingly, using `<model name>_TREND` as a prefix followed by the variable name. See section 10 for details.

7.4 Variogram panel

The variogram is one of the most valuable gadgets in the geostatistical toolbox. Incorporating time, however poses some problems of practical and theoretical order. Using the whole surface of the Earth also contributes to make the situation worse.

LX Other than linear regressions for longitude, latitude and time, there are also multilinear regressions for longitude plus latitude, and for these combined with time, too. Only linear singe-variable regressions are plotted.

LXI the number of bins is configurable, see section 5.4.

LXII The user can choose the elements to visualize.

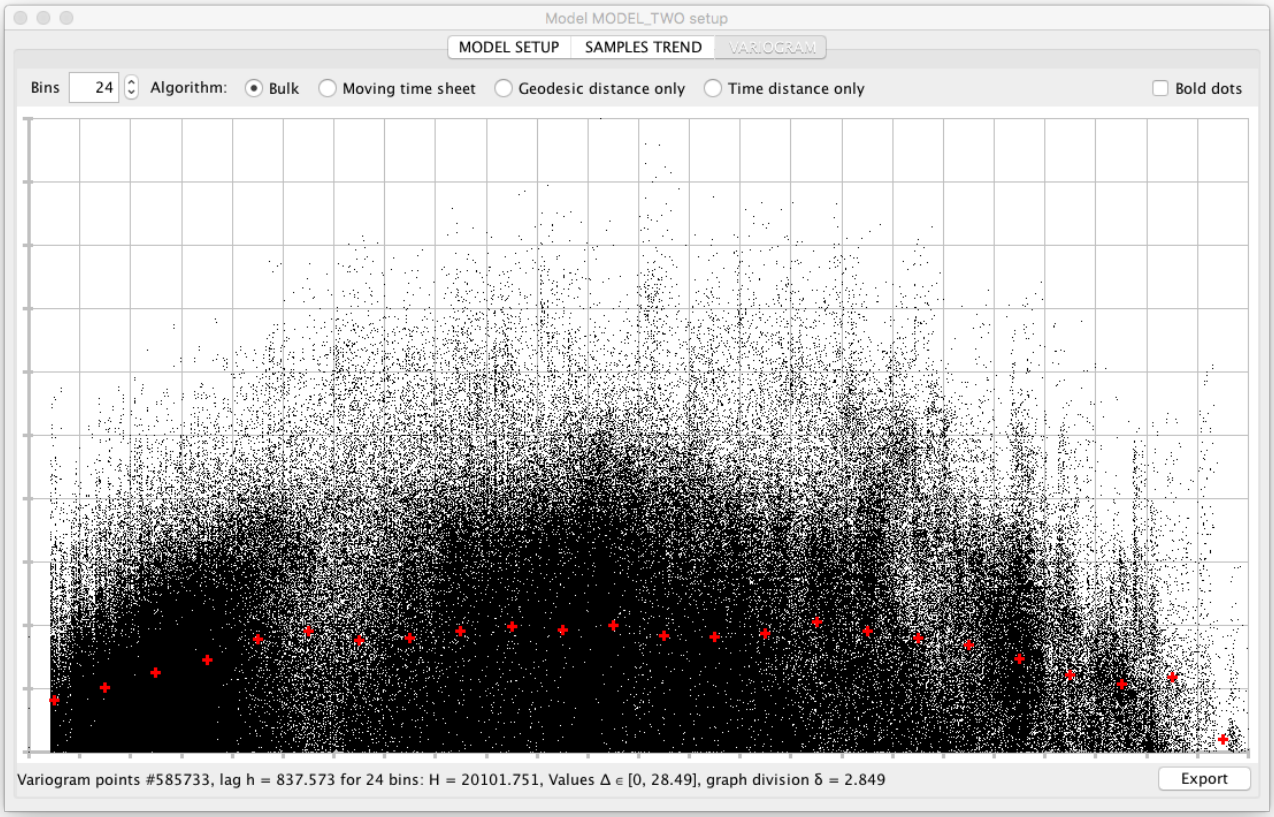


Fig. 18

On the practical side, computational problems grow proportionally with the number of source points (more precisely, they grow with the square of their number). This is not a concern per se, but sometimes it slows down the computation and the visualization processes (the variogram plot is updated whenever the user changes the c or k parameters or if he/she changes the status of a sample point in the **setup** panel).

On the theoretical side, there are some issues that the user should take into account. First of all, if the points are spread all over the surface of the Earth, or at least if they span from pole to pole, it is possible that points that are located very far away bear the same values. In this case the shape of the variogram or, better said, the trend[LXIII] as a function of the distance h (look at the red $+$s in the previous figure): a standard, *respectable*, variogram shows an initial increase as a function of distance which leads to a plateau (the *sill*) that is asymptotically stable. An initial *nugget*[LXIV] is also commonly found.

It is not uncommon, whenever large latitude spans are involved, to find a variogram decrease for very large distances: this decrease, however, should not be interpreted as a spooky long-distance correlation, but rather as a concidence of values because similar phenomena happen approaching the poles. It is also possible that points far away (say at te equator, but on the opposite points of a diameter) share the same value or nearly so. These effects cannot be compensated with the removal of a linear trend, as it is commonly done with projected coordinates, due to the peculiar topology of the sphere.

Most of the problems with the variogram arise because of inclusion of time: many interesting phenomena show a typical seasonal behavior which consists of periodical (more or less regular) variations of the values with some random fluctuiations superimposed. A typical case of study can be the search for a trend over time which is hidden beneath seasonal variations that are much bigger in magnitude. It is important to be aware of the possibility of periodic and multi-periodic patterns that can influence the shape of the variogram.

Timescape Global provides the user with a variogram calculated from the parameters of the model in this way: All the sample points are checked pairwise, if one falls within the causal cone of the other, a point is added to the variogram (a tiny black dot in the plot). Then the points are gropued in a certain number ib bins[LXV] and the averages are plotted with red + marks. These marks show the actual variogram trend (the so-called $\gamma(h)$ function. In the extremely unlikey case of a variogram composed of very few points, the user can check the **bold dots** checkbox to enlarge them. Normally the points of the variogram look like a cloud, maybe clustered around a few values, but there are too many of them to spot a pattern at first sight.

LXIII See [1] for Variograms theory.

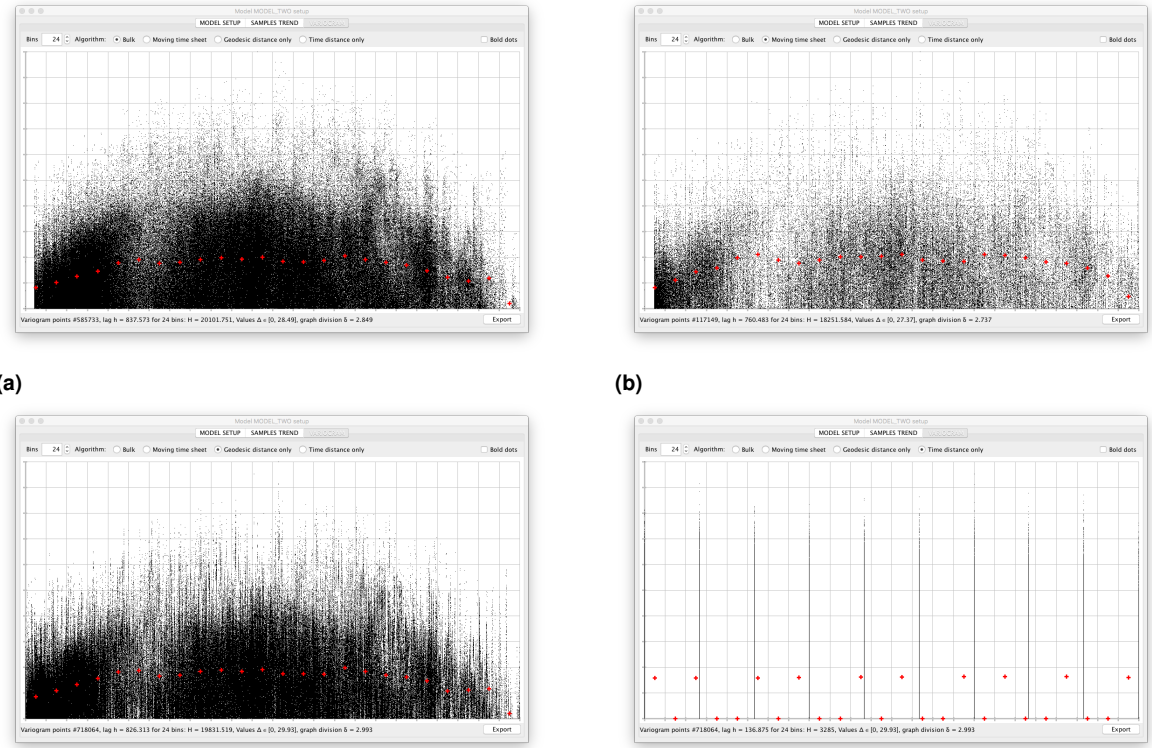
LXIV The *nugget* is associate with the short-distance variability of the sample.

LXV The number of bins can be selected by the user. Default value is 24. Changing the number of bins does not cause the variogram to be recalculated.

7.4.1 Various variogram variations

The addition of the time variable in the evaluation of the distance h poses a series of issues, as sketched previously, but it also offer the possibility of playing a bit with some "variations" to the variogram calculation. In particular, one can take into account or not the time component D_t of the distance (see section 3.3). The way in which time is included also gives the user one more degree of freedom in the variogra evaluation. It is also possible to ignore the distribution of the samples over time or their geographical position on the Earth.

Timescape Global offers four different algorithms for the evaluation of the variogram that the user can choose by selecting the appropriate option in the upper row of the panel:[LXVI]



(a) **(b)**

(c) **(d)**

Fig. 19

- **Geodesic distance only** takes into account only the spatial part of the distance, i.e. $D = D_{\lambda\varphi}$, the difference in time plays no role in the calculations.
- **Time distance only** takes into account only the temporal part of the distance, i.e. $D = D_t$, the difference of position plays no role in the calculations. This is somehow the opposite of the **Geodesic** option.
- **Bulk** (the default option) evaluates D as of the model parameters; this is how the distances are actually calculated during the evauation of the model.
- **Moving time sheet** is a variation of **Bulk** which adds the constraint that each pair of source points should lie be within a time interval of size w_s.[LXVII] This is useful if the variogram cloud is too cluttered to see any pattern at all. This algorithm (the most computationally complex) provileges short-lived effects with respect to the samples' scale.

The user's choice of the variogram algorithm does not affect the evaluation of the model in any way: the distances are avaluated as the model parameters indicate. The variogram is nore a suggestion to the user to help choosing the best parameters for his/her needs.

On the bottom line the user can find a few information for the interpretation of the plot:[LXVIII]

- **Variogram points**: the number of causally connected pairs showed as tiny black dots in the plot.
- **lag** h: the width of one distance bin, measured in length units (kilometers). The user can modify this value indirectly, changing the number of bins.
- **maximum distance** H: the greatest distance between two causally connected source points; $H = nh$, where n is the number of bins.

LXVI Changing the algorithm causes the variogram to be recalculated. It can take some time, depending on the square of the number of sample points.
LXVII Changing w_s causes the variogram to be recalculated.
LXVIII This line shows a running percentage during the evaluation.

- **range** Δ: the range of the absolute value of the difference of values, generally between zero, or nearly so, and $\max\{v_0\} - \min\{v_0\}$, the range of samples' values.
- **division** δ: one tenth of $\max\Delta - \min\Delta$; just to ease the reading of the graph.

7.4.2 Variogram export

The variogram can be exported. pressing the `Export` button a dialog appears to let the user choose the export folder. The output is a folder containing a text file with all the causally connected samples pairs and the `.png` image of the variogram.[LXIX] The default name of the folder is `<model name>_VARIO`, containing `<model name>_VARIO.txt` and `<model name>_VARIO.png`. See section 10 for details.

8 Model evaluation

The evaluation phase is pretty simple (for the user). It is split into two sub-steps: First comes the insertion of all the voxels of the model (empty), so that the database grows of the exact amount needed to host the model. Then comes the actual evaluation phase. This strategy is safer for both the database and the application.

Clicking the **EVALUATION** button on the main panel opens up the evaluation window. Choosing a model to evaluate from the available ones fills the panel with the relevant configuration of the model chosen.[LXX] The coordinates of the centers of the voxels are shown.

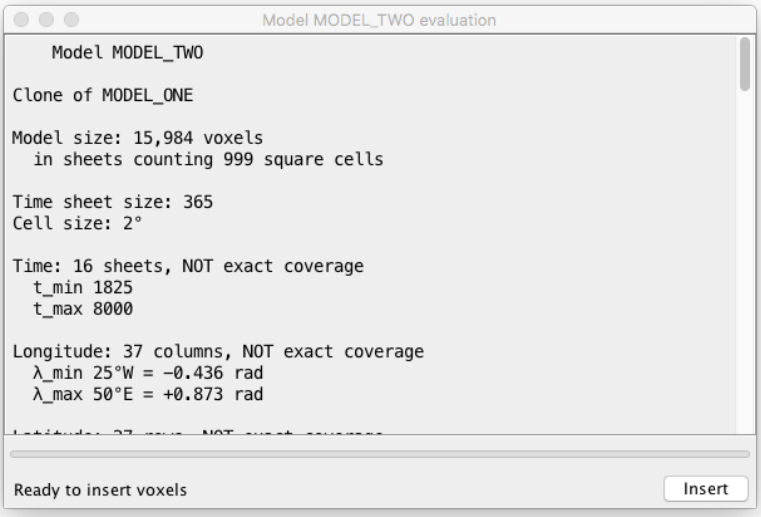


Fig. 20

The user has nothing to do but pressing the **Insert** button. The empty voxels are inserted sheet-by-sheet. The user is informed while the process goes on through a scrollbar and a rolling log. The insertion process can be stopped at any time by closing the window.[LXXI]

LXIX Only the currently displayed variogram is saved. If the user wants to save the variograms resulting from other algorithms or other choices of the number of bins, he/she has to do it by hand.

LXX Choosing a model also locks it to prevent modifications from other panels.

LXXI Insertion abortion releases the model lock and deletes the voxels inserted thus far.

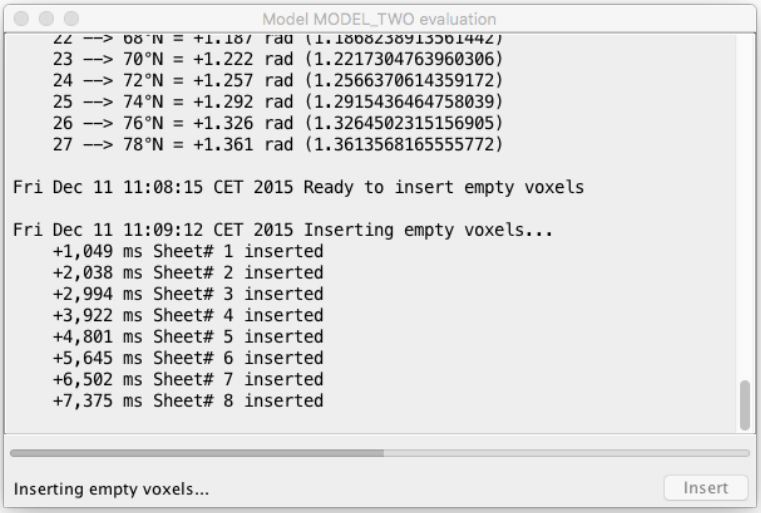


Fig. 21

When all the voxels have been inserted, The evaluation phase can start. Just press the **Evaluate** button and wait:

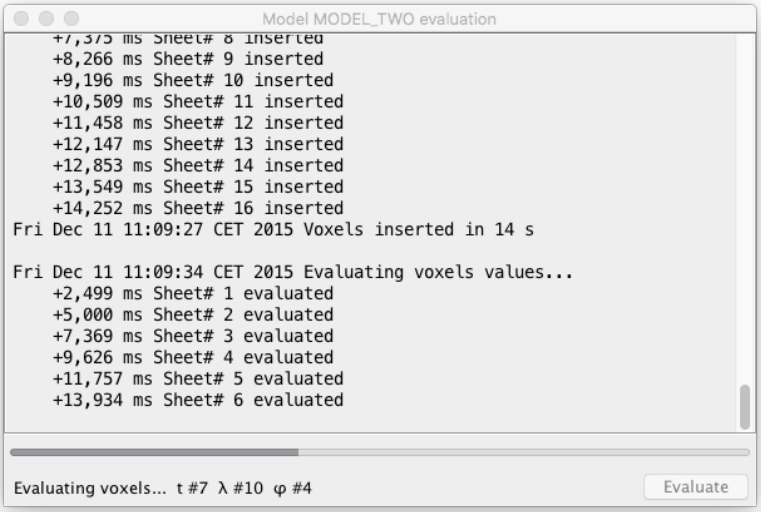


Fig. 22

The model voxels are evaluated one after the other. the rolling log keeps track of the entire process showing the intermediate times after every sheet completion and, together with the scrollbar, the voxel under evaluation is shown at the bottom of the panel. Also this phase can be aborted closing the window: a confirmation dialog appears, the database is cleaned and the model lock released.

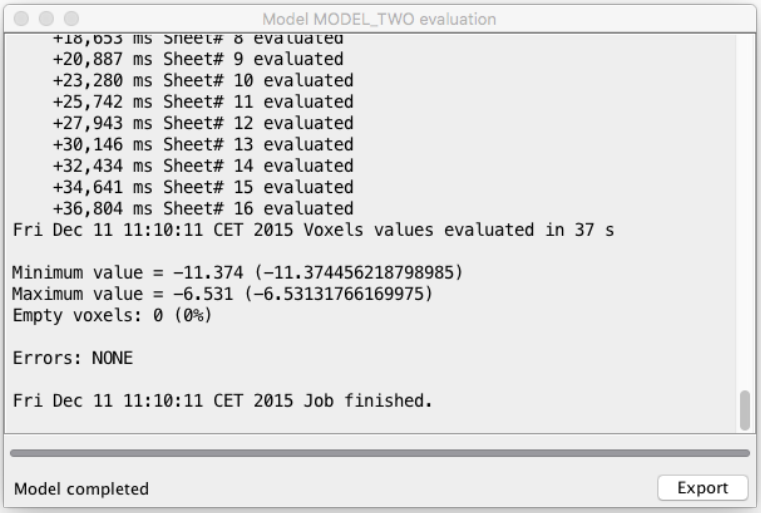


Fig. 23

The evaluation phase can be as fast as the insertion phase or painfully slow, depending on the complexity of the interpolating functions.[LXXII] The log now shows the minimum and maximum interpolated values ant the number of empty voxels. The model stays locked and it is also complete, ready for the next exploration step. That's all.

Before closing the panel, the evaluation log can be exported pressing the **Export** button.[LXXIII]

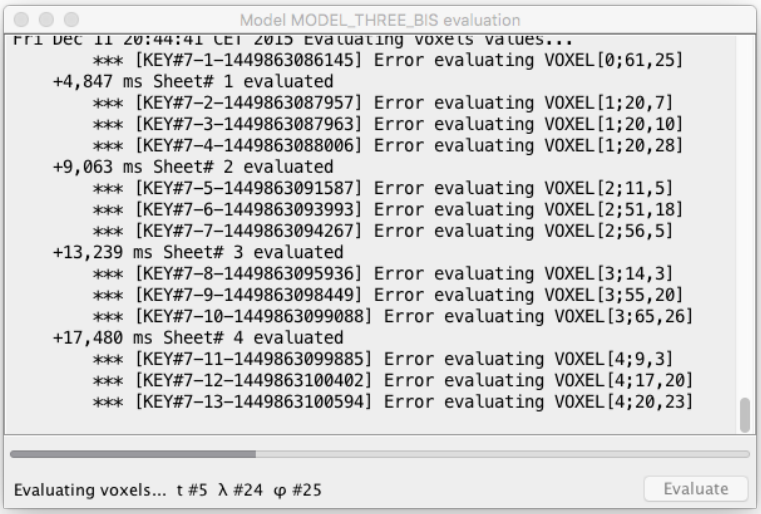


Fig. 24

Sometimes something can go wrong. When it happens during the evaluation of a voxel value, its values remains *null* and the rolling log shows a warning line, like

```
*** [KEY#<id>-<prog>-<timestamp>] Error evaluating VOXEL[<idt>;<idlong>,<idlat>]
```

The corresponding error details are logged in the application log (see section 12) and can be found through the corresponding `KEY#`.[LXXIV] A fragment of the log looks like this:

LXXII The bottleneck for the insertion of the voxels is the database connection, but the evaluation time depends on how many user custom functions have to be parsed at runtime.

LXXIII The log is saved anyway in the application log, see section 12 for details.

LXXIV The key is composed with the model id, a progressive number and the *timestamp*.

```
2015/12/11-20:44:53.994 ERROR timescape.gui.eval.EvaluationFrame.showException.329:
  [KEY#7-6-1449863093993] got EvalException evaluating VOXEL[2;51,18] for
  Model[id:7; label:MODEL_THREE_BIS; description:new model; neighbors:ALL;
    convfact:1.0; causal:1.0; weigh:1/_d; value:_val; scriptEngine:JavaScript;
    pickingProb:1.0; geodesicThreshold-1.0; sheetsize:365.0; cellsize:5.0;
    minTime:1825.0; maxTime:6752.5; minLongitude:-177.37; maxLongitude:173.28;
    minLatitude:-75.5833; maxLatitude:69.7667; minValue:-20.700000000000003;
    maxValue:0.76; locked:Y; completed:N]
timescape.eval.EvalException:
   at timescape.gui.eval.EvaluationFrame.interpolate(EvaluationFrame.java:297)
   at timescape.gui.eval.EvaluationFrame.access$20(EvaluationFrame.java:295)
   at timescape.gui.eval.EvaluationFrame$6.run(EvaluationFrame.java:255)
2015/12/11-20:44:54.103 INFO  timescape.util.KillableThread.kill.10:
   killing Thread[Thread-9,6,main]
```

A fair amount of Java programming skills is advisable in order to read such a kind of information, but also the casual user can get some clues about what has gone wrong during the evaluation: not all errors are real ones.

The error log always begins with the pattern

```
[KEY#...] got EvalException evaluating VOXEL[...] for Model[...]
   timescape.eval.EvalException: ...
```

Here `EvalException` introduces the cause of the error, sometimes it is relatively easy to realize what has happened and why. Most times the error condition is due to a parsing error of some user defined function. This is not a particularly bad situation since it is possible that such functions are not defined for all the possible values of the arguments. If this happens for a relatively negligible number of voxels, we can simply ignore it. The setup panel (see section 7.2) provides an expression testing tool that is of great help in writing the custom functions.

Sometimes an error condition occurs also using the most basic settings (e.g. the default ones) without any custom function. In this case further investigation is advisable: the best strategy consists in finding the location of the bad voxel[LXXV] and running a very fine-grained model in its neighborhood, such clarifying whether the error is due to a really unlucky combination of coordinates values or not. A serious problem is never limited to a single voxel but spreads easily, provided the test model is sufficiently fine-grained.

As a rule of thumb, a small number of errors (say, a few units out of millions) can be negligible, but it is better to have at least an idea of what gone wrong, when and why. This is what the log is for.

8.1 A note on values

Timescape Global is written in Java: in this language values like *Infinity* and *NaN*[LXXVI] make perfectly sense. Most databases, however, cannot record such exotic values as doubles, so all not-so-well behaved values are treated as *null* and as such inserted into the database table. The user is alerted in the evaluation log of any ill-valued voxel. On the other hand, regular *null* voxels are located out of all the causal cones emanating from the sample points and are not associated to any error condition or anomalous value.

9 Model exploration

A **Timescape Global** model is basically a set of voxels. These are recorded as a set of values associated to their ids: $v(t_k;\lambda_i,\varphi_j)$.[LXXVII] A conventional geostatistical two-dimensional model is normally examined in a GIS environment as a raster/grid dataset while a Timescape model contains a lot more information, linking time series to geostatistics, but it comes at a price: data must be *extracted* somehow from the model to be analyzed.

Data can be accessed directly from the database table `voxel` of course, thus using TimescapeGlobal as a step of an articulated workflow, but something interesting can be done within the software itself, that can output time series, GIS layers and some statistics out of the model voxels. Three main panels constitute the **Exploration** window, accessed by the `Exploration` button of the main panel:

- A **Display panel**, the core of the model explorer, where the values are plotted against space and time, the model can be examined at the finest detail, also through animations. The user can also export GIS layers, images and time series.
- An **Analysis panel** that portrays descriptive statistics, trend and autocorrelation analysis, sheet-by-sheet. Statistics can be exported in text format for reference and for further in-deep analysis.
- A **Residual panel**, in which the source values are confronted with the model-interpolated ones. This is the place for checking the accuracy of the calculations. Also residuals can be exported as text file.

LXXV For storing efficiency, the voxels do not contain the actual coordinates, but integer pointers to them. To retrieve the coordinates (time, longitude and latitude) one must see the rolling log of the evaluation panel, where the latter are shown before the voxel insertion phase. The ids are shown in this order: `VOXEL[idt;idlong,idlat]` and are shifted of one unit with respect to the database ids.

LXXVI *NaN* stands for "Not a Number", the outcome of an undefined (i.e. $0/0$) or meaningless expression.

LXXVII The actual coordinates are not recorded in the `voxel` table, see section 4.1

As said, voxels do not bear the time an coordinates information, the exploration window converts for the user the coordinates ids into the actual values. All the values are displayed with a certain number of significant digits[LXXVIII] but are exported with the full available precision.

9.1 Display panel

This panel shows at the same time a section at at constant time and a time series "dug" at a point (by default, the display panel starts with the time series of the central space cell and the space layer of the middle of the available times). the information is presented both graphically and numerically and can be exported at any time.

From left to right, top to bottom, the panel presents: a GIS-like map of the Earth with a semi-transparent model layer superimposed; a time series core,[LXXIX] some numerical details, export buttons and a 1:1 pixel map. These sub-panels are synchronized so that clicking on a space site sets a new space location and clicking on the time core sets the appropriate time sheet. All the values and images change accordingly (sometimes this can be very slow, depending on the size of the model and the speed of database connection).

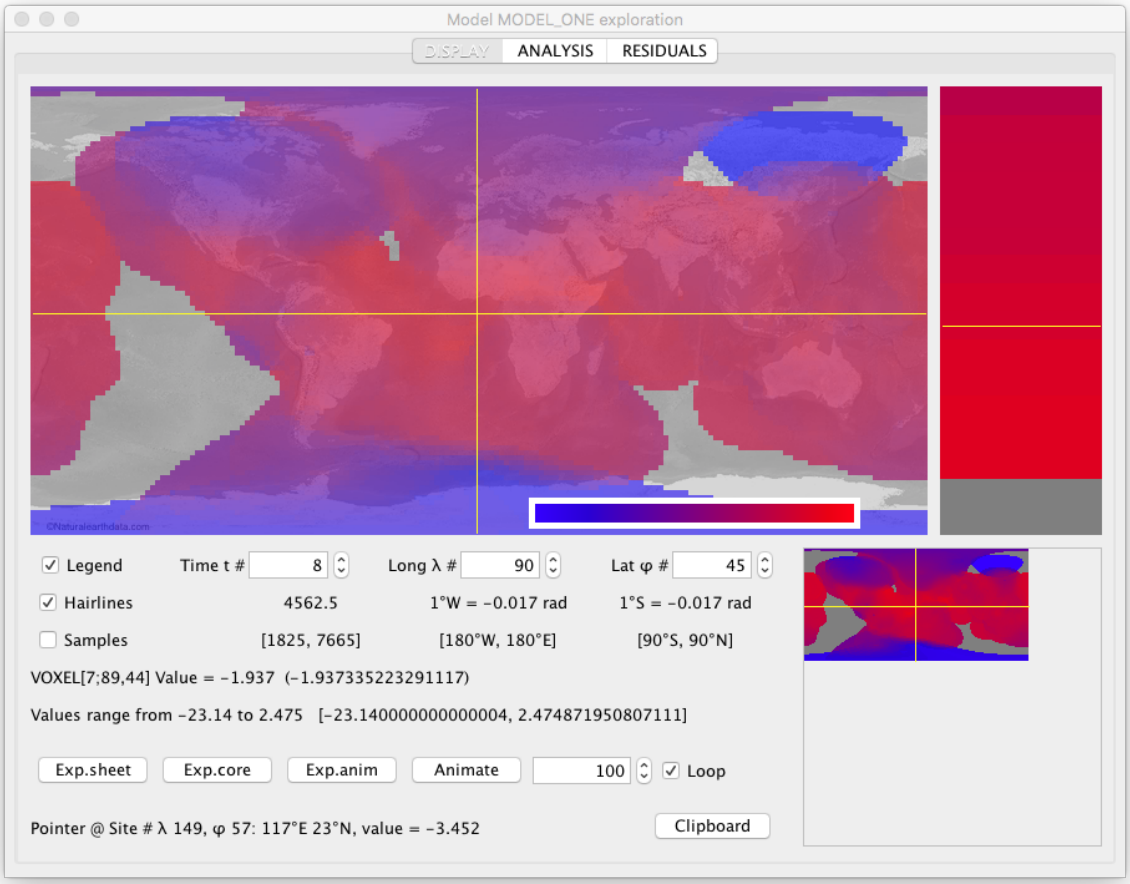


Fig. 25

Hovering the mouse pointer on the map or the core makes the `Pointer` line (lower left corner of the panel) show the voxel coordinates and value. Other than clicking on the images, The voxels can be selected by choosing their ids (*t*#, *λ*#, *φ*# fields): the actual time and coordinates appear just below.[LXXX] One more line shows the minimum and maximum values for the coordinates. The `VOXEL` line shows the voxel as it is in the database (note the shifted value of the ids), its value and the values range, in the following line.

The `Clipboard` button copies the content of the `VOXEL` line into the system clipboard, to be pasted in other applications. Three checkboxes affect the display of the model:

- `Legend` shows/hides the color legend (blue = minimum to red = maximum value) on the lower right corner of the world map.

LXXVIII This is configurable, see section 5.4.

LXXIX The crosshairs indicate the space cell of the time series (the core) and the yellow bar on the core indicate the time of the displayed sheet.

LXXX Longitude and latitude are expressed both in decimal degrees and radians, time is expressed in the appropriate units.

- `Hairline` shows/hides the yellow hairline on the world map, the time series core and the 1:1 map. Hiding hairlines does not affect the synchronization on mouse click.
- `Samples` shows/hides the samples position in the world map and time core. The samples are represented as +s on the map and bars on the core.[LXXXI]

The **animation** function allows the user to have a fast view of the behavior of the model. Just press the `Animate` button to start the animation; the parameter on the right of the button controls the frame rate (in milliseconds) and the `Loop` checkbox sets the loop on and off. While the animation is going, the `Animate` button turns into `Stop`: press it to stop the animation.

The most useful functions of the **Display panel** are those for exporting parts of the model.[LXXXII] In any case the user is presented a dialog window for selecting the appropriate file or directory:

- export **animation**. By clicking the `Exp.anim` button the application creates a .gif image of the sheets, one time after the other, with the frame rate selected by the user. The result of the export is a single file called `<model name>.gif`.
- export **sheet**. By pressing `Exp.sheet` the application creates a directory containing:
  - a text file containing all the sheet's voxels details;
  - the world map as .png image;
  - the 1:1 pixel map as .png image and
  - the associated .pgw file for georeferencing;
  - an ESRI Grid .asc georeferenced file to be used in GIS.

  the directory default name is `<model name>_SHEET_TIME<time id>`, the files inside are labelled accordingly. A note about georeferencing: when the minimum longitude exceeds the maximum longitude ($\Lambda_m > \Lambda_M$: i.e. the space boundaries are wrapped around the Pacific) the sheet is exported regularly, but the longitude coordinates exceed the value of +180. Most GIS need to be set in order to reproject such kind of data, or they will unwrap the section exceeding $\lambda = 180°$.
- export **core**. Pressing `Exp.core` exports a directory with the image of the core and a text file containing all the relevant values. The default directory name is `<model name>_CORE_LONG<long id>_LAT<lat id>`, the files inside are labelled accordingly.

More details on exported media can be found in section 10.

9.2 Analysis panel

This panel shows some statistics on a sheet-by-sheet baisis, these include general informations about the model, descriptive statistics of the sheets, trend analysis, histograms and autocorrelation estimation.

The upper left panel contains all the relevant model informations, including the time and geographical coordinates of the voxels centers.

The upper right panel shows, for any sheet, the distribution of the values, the linear trend against longitude and latitude, and the multilinear trend with respect to longitude and latitude combined, with the Pearson's r and/or R^2 values.

LXXXI More precisely, used samples are represented with +s and bars on the left half of the core, while unused samples are represented as ×s on the map and bars on right half of the core.

LXXXII There is not an "export model as a whole" function because this would take too long and produce too big files. The interested user should access the database directly.

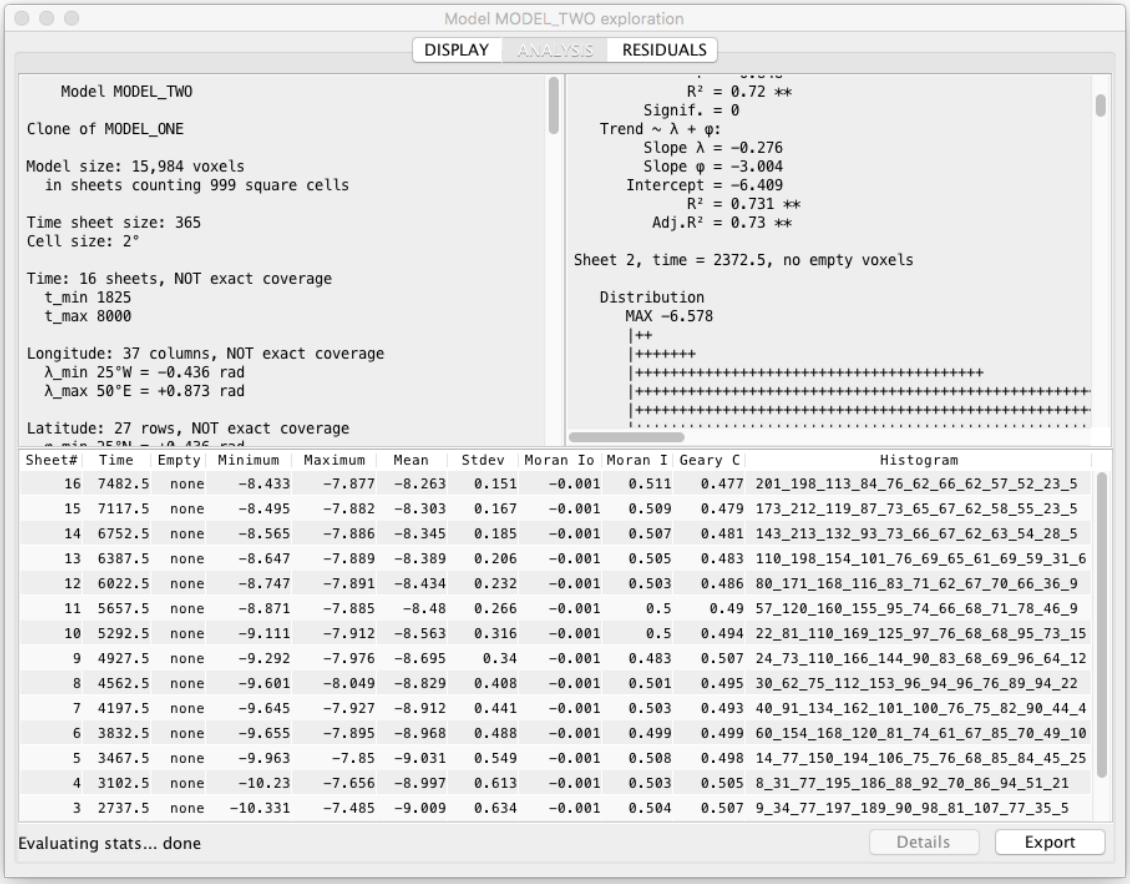


Fig. 26

The lower panel contains a list of statistical parameters for any time sheet, in reverse order. The computation time can be quite long, so the bottom line shows what's going on:

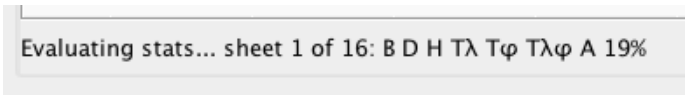


Fig. 27

The meaning of the line is: B: evaluation of the boundaries is going on; D: descriptive statistics going on; H: histogram; $T\lambda$: linear trend values $\sim$ longitude; $T\varphi$: trend $\sim$ latitude; $T\lambda\varphi$: multilinear trend $\sim$ longitude + latitude; A: autocorrelation functions (this is really slow). The upper right panel is filled only after the completion of these stats. Field by field, the informations provided include:

- **Sheet#**: the sheet number
- **Time**: the time value of the center of the voxels in user's time units
- **Empty**: the number of empty voxels
- **Minimum**: the voxels minimum value
- **Maximum**: the voxels maximum value
- **Mean**: the voxels average value
- **Stdev**: the values standard deviation
- **Moran** I_0: the Moran null huypotesis value, i.e.

$$I_0 = \frac{-1}{N-1}$$

  where N is the non-empty voxels number, I_0 is almost zero in most cases
- **Moran** I: the Moran autocorrelation I function:

$$I = \frac{N}{\sum_{ij} w_{ij}} \frac{\sum_{ij} w_{ij} (v_i - \bar{v}) (v_j - \bar{v})}{\sum_i (v_i - \bar{v})^2}, \quad w_{ij} = \frac{D_{ij}^{-1}}{\sum_{kl} D_{kl}^{-1}}$$

where D_{ij} is the distance between i^{th} and j^{th} non-empty voxels, which values are v_i and v_j resepectively, and $\bar{v}$ is the sheet's mean value. During the evaluation this field shows the completion percentage

- **Geary** C: the Geary autocorrelation function:[LXXXIII]

$$C = \frac{1}{2}\frac{N-1}{\sum_{ij} w_{ij}}\frac{\sum_{ij} w_{ij}(v_i - v_j)^2}{\sum_i (v_i - \bar{v})^2}$$

- **Histogram**: the histogram of values. The number of bins can be configured in the properties file of the application, see section 5.4

As soon as the list is complete, the user can click on a row to see more details about the sheet's stats: skewness and kurtosis, 25, 50 (the median) and 75 percentiles; the values are shown with all the available digits. These stats are intended as a companion to the graphical objects exported from the **Display panel**.

All these statistics can be exported in a single text file by pressing the `Export` button. The user can choose the name of the file (default filename is `<model name>_ANALYSIS.txt`). See section 10 for details.

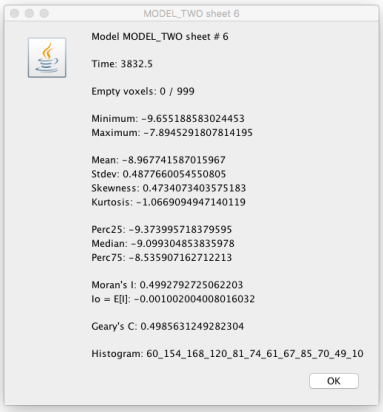


Fig. 28

9.3 Residuals panel

This panel shows both graphically and analytically the values of the model's residuals. The residuals ore calculated as follows: for any source point are calculated the time and space coordinates of the containing voxel, if this voxel is not-empty, a residual is evaluated as the difference $v_0 - v$ (sample minus model, the order matters!)

The residuals are shown graphically on a world map, a time map and a model vs sample plot. The user can select a row clicking on it, thus hilighting the residual's source point in the said panels.

As usual, an used sample point is plot with a black $+$ mark on the Earth panel and a black bar in the left half of thr time panel, while an unused point is plot with a red $\times$ mark on the world map and a red bar on the right half of the time panel. The blue "fishbone" in the middle of the time panel shows the time sheets.

In detail, the informations provided for any source point are:

- **Use**: used/unused flag, as set in the model
- **Sample ID**: id of the sample
- **ids**: time, longitude and latitude ids of the voxel containing the sample[LXXXIV]
- **Time**: time of the sample, in user's units
- **Long**: longitude of the sample, in decimal degrees
- **Lat**: latitude of the sample, in decimal degrees
- **Geodesic**: geodesic distance between the sample and the center of the voxel
- **ΔTime**: difference between sample and voxel center times
- **Distance**: distance between sample and voxel center
- **Smp.value**: value of the sample
- **Mod.value**: value of the voxel (can be null)
- **Residual**: residual (can be null)
- **Sq.residual**: squared residual (can be null)

The residuals table can be exported along with the plot by clicking the `Export` button. The user has to choose a directory name (default name is `<model nam>_RESID`). **TimescapeGlobal** saves an enlarged version of the residuals plot in .png format[LXXXV] and a .txt copy of the table. More details in section 10.

LXXXIII See [1] for details about the measures of spatial autocorrelation.

LXXXIV These ids range from one to the number of sheets/cells. In the details line below te table the ids are the actual database ones, shifted by -1 unit. The model's unique numerical id is shown too, separated by a colon. This is provided as a reference to the interested user for finding the right record in the database.

LXXXV Indep (horizontal) axis bears the sample values, while the model values are on the dep (vertical) axis.

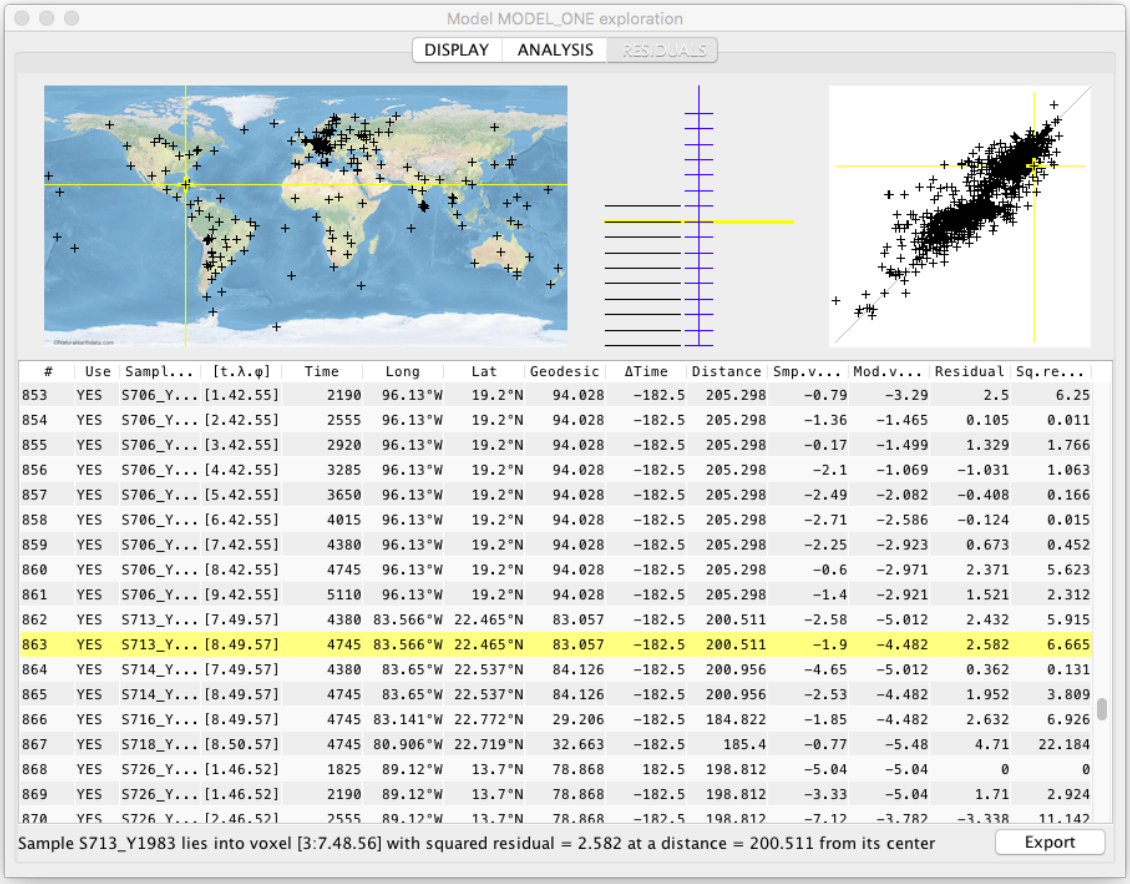


Fig. 29

10 Exported Timescape objects

As said in the previous sections, Timescape can produce many files to be used as input data for other elaborations. At every stage of a model's workflow there is something to export: the sample points statistics and variogram, the evaluation log and many images, gis layers and statistics for the finished model.

Hint: if one has forgotten to save the sample point statistics or the variogram, during the setup phase, it is possible to clone a finished model, so accessing its setup statistics again.

The figure below shows a sample of all the possible exports of **Timescape Global**. A purple dot marks the setup files, a red dot is referred to the evaluation and a green dot marks the finished model exploration exports.

As a general rule, the software assigns a tentative label to the export. This label is composed by the model name followed by the kind of information being saved, one or more indices if appropriate, and a suitable suffix. The user can of course rename the file(s) as he/she likes.

Sometimes the export consists in a single file, but more often it will be a directory containing some files, in this case the files share the same name with different suffixes according to their content.

All the images are stored in .png format, with the exception of the animated .gif (.png does not support animation) and the Grid GIS layer (which is a GIS *raster* grid bot not, strictly speaking, an image.

All text files are recorded in plain ascii format, with .txt suffix. Generally the first few lines of the text files contain ♯-commented rows that contain a timestamp, the file name and some notes. Then data follows, in a .cvs (comma separated values) fashion.

In the following the user can find a detailed description of the content of the exported files.

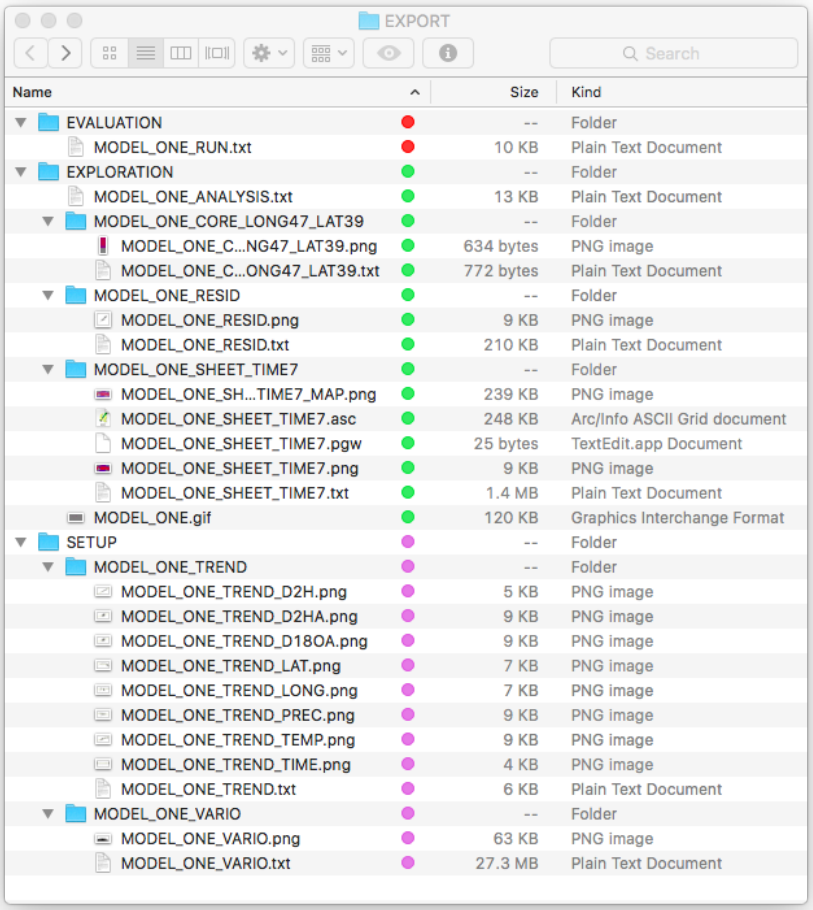


Fig. 30

10.1 Setup

Both trend statistics and variogram can be exported.

The variogram consists in an image (the copy of what can be seen on the **VAriogram panel**) and a text file. The lattor contains all the numerical information to complemet the image, and two .csv lists, one relative to the red crosses (the general trend) and one relative to every single point (the black cloud). The fields corresponding to the trend are `bin` (the bin ♯, `dist` (the h value of the center of the bin) and `delta` (the mean difference of the variogram points that fell into the bin.

The complete variogram has more fields:

- `from`: the id of the source point/event emanating the causal cone
- `to`: the id of the sample point/event that sits inside the caudal cone of the latter
- `dist`: the space-time distance separating the two events
- `delta`: the absolute value of the difference of the values.

Note that the variogram export file can be millions rows long, the order of magnitude of the square of the number of sample points.

The trend statistics and images can also be exported. The images are the exact .png copies of left-hand panel images array; the images are labelled following the name of the independent variable: `_TREND_TIME`, `_TREND_LONG` and `_TREND_LAT` fot time, longitude and latitude.

If ancillary data are present (as in the example above), there will be one more image per variable, labelled accordingly.

The text file of the trend analysis contanis le right-hand side statisics.[LXXXVI]

10.2 Evaluation

The evaluation export consists simply in a copy of the running log shown in the main text area of the **Evaluation panel**. The suggested suffix of the file is `_RUN`.

LXXXVI To be ascii compliant, the greek letters λ and φ are substituted by `long` and `lat`.

If the user does not save the evaluation running log, it can be found in the application log (see section 12). However, logs are not forwver, this information will be eventually lost as new lines are added to the log.

10.3 Exploration

The **Analysis panel** export consists in a .txt file, the suggested prefix is `_ANALYSIS`. The content of the file consists in the statistics contained in the upper left panel, followed by a (large) table that joins the content of the upper right panel with the lower table.[LXXXVII]

The fields are:

- `sheet`: time sheet♯
- `time`: time value of the center of the sheet
- `empty`: number of empty voxels
- `minimum`: minimum value
- `maximum`: maximum value
- `mean`: mean value
- `stdev`: standard deviation
- `skewn`: skewness
- `kurt`: kurtosis
- `perc25`: distribution 25th percentile
- `perc50`: distribution median
- `perc75`: distribution 75th percentile
- `moran_I0`: Moran's null hypotesis value I_0
- `moran_I`: Moran's autocorrelation function I
- `geary_C`: Geary's autocorrelation function C
- `hist_bin1...hist_binN` number of elements in the histogram bins (the number of bins can be set in the application preferences, see section 5.4
- `long_slope`: slope of the value $\sim$ longitude linear regression
- `long_intcp`: intercept of the value $\sim$ longitude regression
- `long_r`: Pearson's r of the value $\sim$ longitude regression
- `long_r2`: R^2 of the value $\sim$ longitude regression
- `long_sgnf`: significance of the value $\sim$ longitude regression
- `lat_slope`: slope of the value $\sim$ latitude linear regression
- `lat_intcp`: intercept of the value $\sim$ latitude regression
- `lat_r`: Pearson's r of the value $\sim$ latitude regression
- `lat_r2`: R^2 of the value $\sim$ latitude regression
- `lat_sgnf`: significance of the value $\sim$ latitude regression
- `longlat_slopelong`: longitude slope of the value $\sim$ longitude + latitude multilinear regression
- `longlat_slopelat`: latitude slope of the value $\sim$ longitude + latitude regression
- `longlat_intcp`: intercept of the value $\sim$ longitude + latitude regression
- `longlat_r2`: R^2 slope of the value $\sim$ longitude + latitude regression
- `longlat_adjr2`: adjusted R^2 slope of the value $\sim$ longitude + latitude regression

The **Residuals panel** export consists in an image (the inflated version of the model vs sample values plot) and a .txt file with a copy af the residuals table. The directory name propesed is `_RESID`, the files contained therein are labelled accordingly.

The fields are:

- `num`: progressive number
- `status`: `good` if the sample falls into a regular voxel or `empty` if the sample falls into an ampty voxel
- `used`: used flag, indicates whether the sample is used in the model or not
- `sample_id`: unique sample id
- `sample_time`: sample time
- `sample_long_deg`: sample longitude (degrees)
- `sample_long_rad`: sample longitude (radians)
- `sample_lat_deg`: sample latitude (degrees)
- `sample_lat_rad`: sample latitude (radians)
- `voxel_time_id`: voxel time id
- `voxel_time`: voxel time
- `voxel_long_id`: voxel longitude id
- `voxel_long_rad`: voxel longitude (radians)

LXXXVII All statistics are evaluated via the Apache `commons.math3.stat` package.

- `voxel_lat_id`: voxel latitude id
- `voxel_lat_rad`: voxel latitude (radians)
- `deltat`: time difference from the sample point to the center of the voxel
- `geodesic`: geodesic distance from the sample point to the center of the voxel
- `distance`: distance from the sample point to the center of the voxel
- `sample_value`: value of the sample
- `voxel_value`: value of the voxel (can be empty)
- `residual`: residual (sample - voxel, can be empty)
- `squared_residual`: squared residual, can be empty

The **Display panel** exports are the real interesting stuff. These include time series and GIS layers, as well as ascii. txt files. The labelling convention goes as follow: the model name id followed by `_ShHEET` and the time id for a time sheet, or `_CORE_LONG` + longitude id + `_LAT` + latitude id for a time series.

A time series directory contains a .png image (the copy of the upper image panel of the **Display palel**) and a .txt file with all the relevant information (coordintaes of the core) and the time series as a list of values.

The sheet directory contains five files:

- `_MAP`: a copy of the World map in .png format.
- the 1:1 .png image (bottom right corner of the **Display panel**) with the accompainying .pgw file, for georeferencing.
- a Grid .asc file, ready for GIS usage
- an ascii .txt file containing the following fields:
  - `num`: a progressive number
  - `lattice_id`: the lattice coordinates of the voxel (time, longitude and latitude ids in square brackets, separated by _)
  - `long_id`: longitude id
  - `longitude_deg`: longitude (degrees)
  - `longitude_rad`: longitude (radians)
  - `lat_id`: latitude id
  - `latitude_deg`: latitude (degrees)
  - `latitude_rad`: latitude (radians)
  - `value`: value, can be empty

The time-ralated information is contained in the header rows of the file.

Last but not least, an animated .gif can be exported from the **Display panel**. This is not per se a substitute of conventional statistics, but it can be a valuable help in spotting an evolution pattern out of the data. The image is an ordinary .gif.[LXXXVIII]

11 GIS interaction

Timescape Global was developed with a GIS workflow in mind. Its models can be used to produce time series (digging cores) and, above all, GIS layers (slicing equal-time sheets). This is done basically in the **Display panel** of the **Exploration** window.

The export of a sheet produces three files, a world map which is just the copy of the screen image (not georeferenced), a georeferenced .png image,[LXXXIX] colored according to a blue-red gradient and, most important, an ESRI Grid raster layer. Grids are somehow old-times fashioned ascii files, but have some advantages over binary (say, geotiff) images. First of all, being ascii files it is easy to dig into them, conversion is easily accomplished in any GIS package ([8] in the image above).

Empty voxels translate into *null* grid cells, which are recorded with a conventional value (by default it is -9999).[XC] It is a good idea to export all the equal-time sheets of the model, but the practical feasibility of such an operation depends on the actual size of the model: really bulky models are better stored in a database: check whether the database is protected by a regular backup service or consider an ascii export of the relevant data.

LXXXVIII The encoding class was developed by Kevin Weiner, see section 13 for details.

LXXXIX The georeferencing is contained in the accompanying .pgw file.

XC Should the value -9999 be within the range of acceptable model values, the user must change it, setting a suitable value in the `timescape.properties` preferences file (see section 5.4). Some rough GIS applications cannot understand other values for *null*, change it wisely.

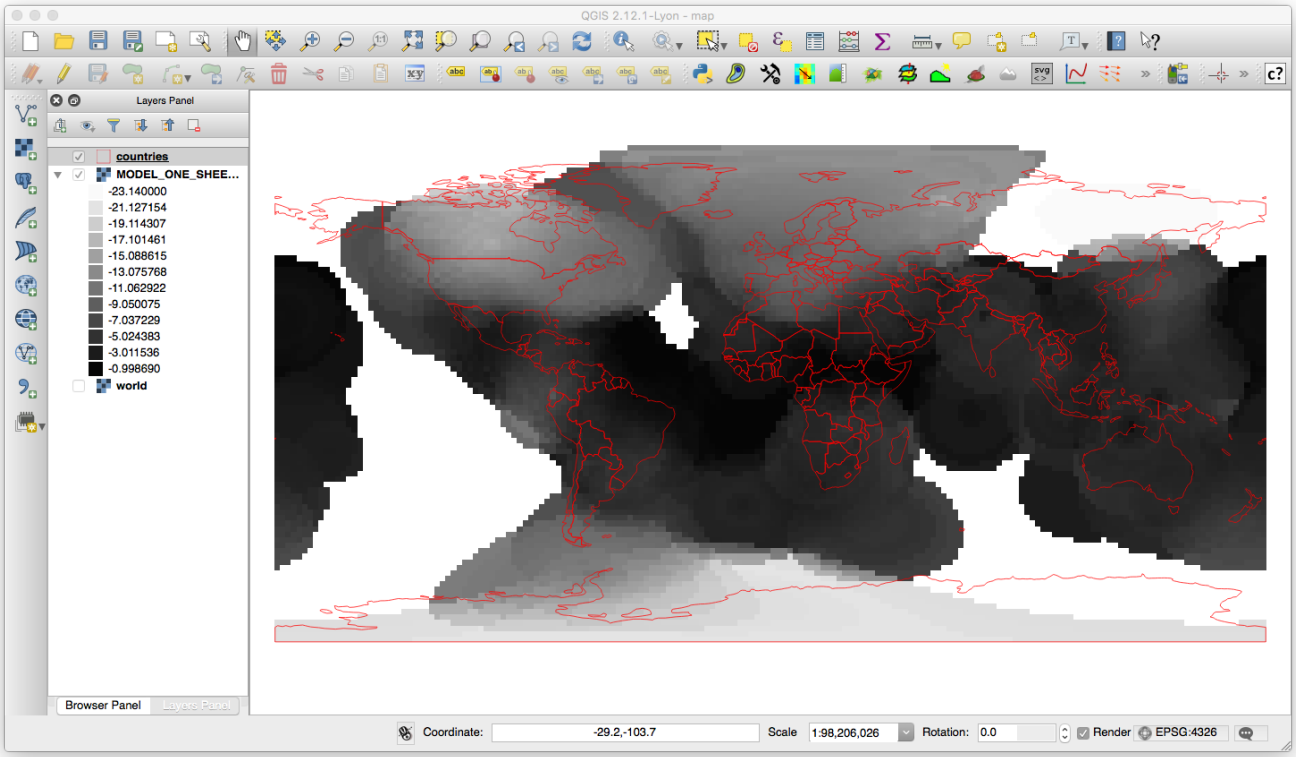


Fig. 31

It is also easy to import the data into an R *DataFrame* object [9], or into any other statistics package of choice. The figure below shows an RStudio [10] data import session.

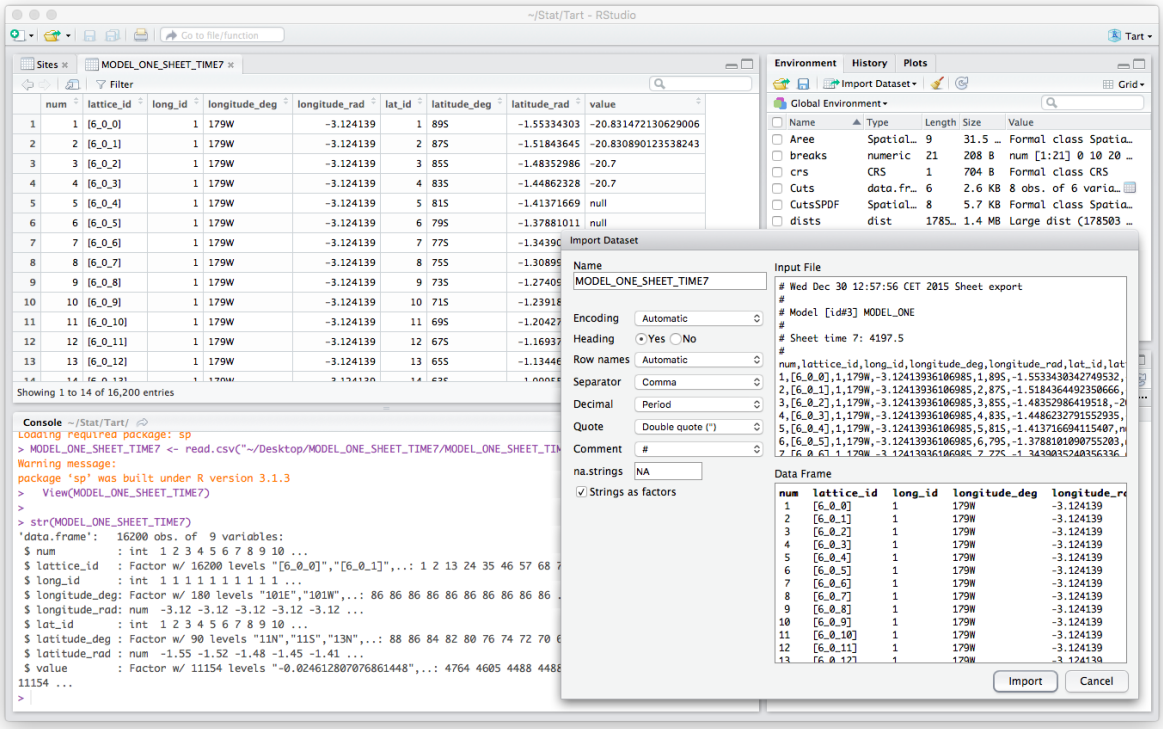


Fig. 32

R is also the statistical environment of choice for playing with time series. The interested user can see e.g. [11]. Note that in order to have time series with an appreciable number of observations (say ~ 1000) the model has to be big. If we are interested

only in the time evolution of a small area, it can be wise to evaluate a “column” model, with a few base pixels and a high time resolution.

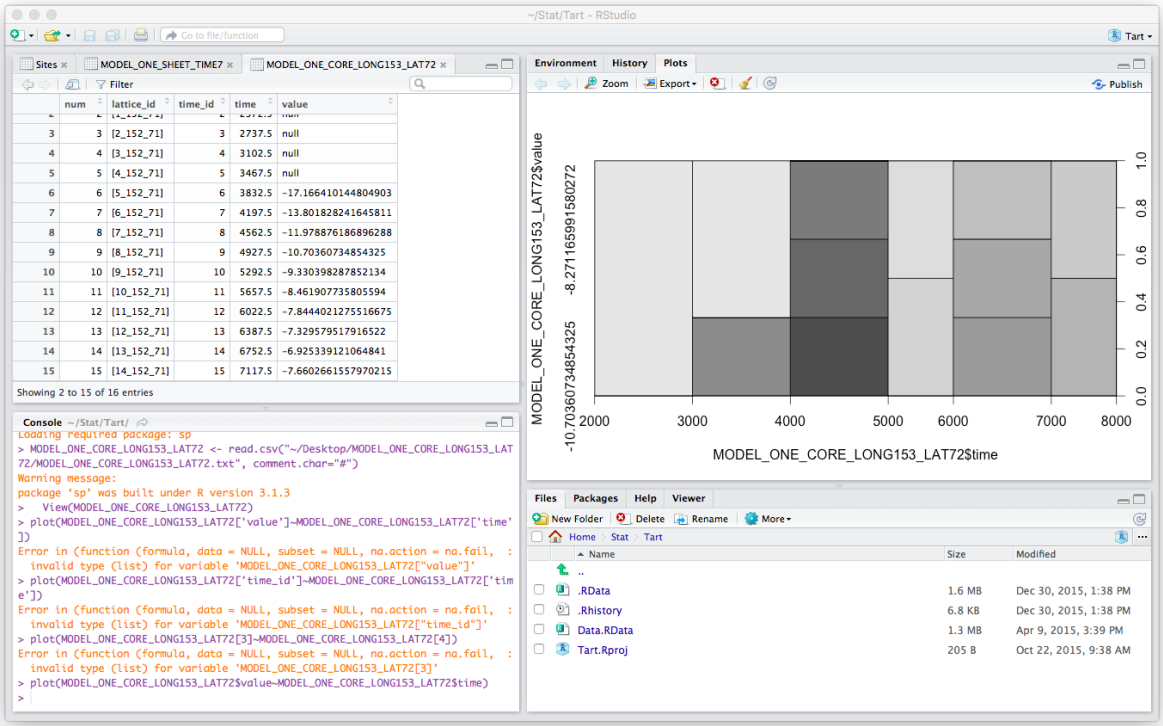


Fig. 33

12 Logging

Timescape Global logging system is based on the Apache log4j logging framework (see [12]). In an ideal world, the user should have no need to read the application log, but sometimes it helps finding the cause of evaluation errors.

The logging directory must be configured (see section 5.4). the user who wants to configure the other logging parameters can do so, according to the Apache log4j standards.

For the unexperienced user, a few hints about the logging pattern: see for example the line[XCI]

```
2016/01/02-12:21:56.837 INFO  timescape.TimescapeGlobal.main.20: TimescapeGlobal started
```

The pattern of each logged line is:

- `2016/01/02-12:21:56.837` is the timestamp, up to milliseconds. The lines are recorder in order of time (older first), when a log file is full, it is renamed with a `.1` (`.2` `.3` etc) suffix, up to `.10`.
- `INFO` is the logging level. Levels include `INFO` and `ERROR`, that often hilights the most useful informations.
- `timescape.TimescapeGlobal.main.20` is the Java class that is actually logging, including the code line (`20`). This tells the user of the actual instruction of the code[XCII] that is logging the line. That is of no use to the causal, non Java-aware, user.
- `TimescapeGlobal started` the message to the user.

The message can be as short as few words (`TimescapeGlobal started`), or a verbose, multi-line explanation of what’s going on, especially in case of error, when it is not uncommon to find a long list of nested Java *Exceptions*.

When a model evaluation is finished, all the relevant informations are also logged. Sure the user has already saved them on a file (see section 8) but in case of need the log keeps them stored, or nearly so: logs are not forever, a rolling system keeps only the “freshest” information, discarding older lines as files grow up.

As described in detail in section 8, every voxel calculation error during the model evaluation is logged with a reference key: Care is given to make this information understandable to all users, but sometimes the verbose Java syntax can be a bit confusing to the uninitiated.

XCI This is logged anytime **TimescapeGlobal** starts.
XCII See section 13.

13 Timescape Global code

The language of **Timescape Global** is Java. It is not a common choice in the GIS word and it has its pros and cons, the main reason for this choice is the stability and reliability that the Java language offers, especially when a contimuous database dialog is central to the application (which is not so common in most GIS workflows). Java is also easily portable, just installing the Java Virtual Machine[XCIII] (see [13]).

The code is available to anyone as part of the distribution, as an Eclipse project. Classes can be easily imported into any other software development environment of choice.

13.1 Code structure

The code layout is pretty simple, a few packages are structured functionally: data model representation, GUI (Graphical User Interface) and evaluation. Various levels of cleanliness characterize these packages' code, reflecting the complexity of the tasks involved. The data model is fairly standard, Hibernate-based, so that the user can easily recycle the data model classes. The `eval` package is surprisingly light, in fact it contains the classes used for the evaluation, but the algorithm is implemented in a GUI window.[XCIV]

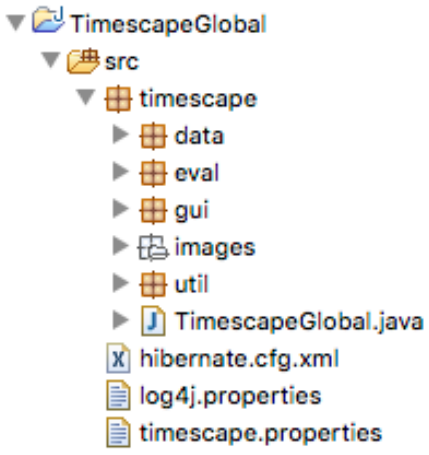


Fig. 34

The GUI is also subdivided functionally, according to the setup–evaluation–exploration structure of the workflow.

The main class, invoked by the Virtual Machine at startup, is `TimescapeGlobal`. This class establishes a database connection and opens up the main window (or the database population window if the database is found to be empty). Note also the location of the preferences files, at root level; the preferences settings are described in detail in section 7. The user can opt for the manual editing of the preferences or the use of the **Timescape Configurator** utility (see section 5.5).

The `util` package contains (you guessed) some utility classes for formatting, logging and exporting files. Also in this package is found the class `KillableThread`, an extension of `Thread` used in many sections of the GUI. The class `AnimatedGifEncoder`[XCV] is used to encode the gif animations of the models.

The `images` package contains all the image used by Timescape's GUI.

13.2 Data model

The `data` package contains all the data structures used in the application, it is composed basically form the `hibernate` package which mirrors the database model, plus some other classes used for the evaluation of the model (`Event`), the variogram (`Vario`) and the evaluation of the residuals (`Residual`).

The `hibernate` package might be the most interesting one for the developer who wants to modify or integrate **Timescape Global**. Here is the mapping of the ER model (see section 4). In particular we have:

- a `Connector` class.
- a `BLO` class, containing some business logic.
- three sub-packages which map the actual data structure. Each sub-package contains one or more table-mapping objects, a `Manager` class and an .xlm hibernate configuration class; these configuration files need not to be configured for the normal usage of the software, only a modification of the table structure calls for their editing. In detail, the `hibernate` packages are:
  - `model`: it contains the mapping of the `model` and `model_source` tables; this mapping is the one most likely to be changed in case of a modification of the software.
  - `source`: it contains the mapping of `ancillary`, `source` and `ancillary_source` tables; here is all what refer to the sample points (remember, one sample dataset per instance of the software). `Bounds` is a utility class for calculating the time- and space bounds of the sample points.
  - `voxel`: it contains the mapping of the `voxel` table.

Also into the `voxel` package is Hue, a service class which supplies the colors for painting the voxels in the **Exploration panel**.

The other `data` classes (`Event`, `Vario` and `Residual`) have nothing to do with the database. They are used to store into RAM as much data as possible, during the evaluation, to avoid database calls as much as possible. A big list of `Vario` is instanced when calculating the variogram in the **Setup panel**, This can have a lot of elements (at most the square of the dataset points number, maybe millions). The lists of `Residual` of the **Exploration panel** has as many elements as the samples dataset, as is the case of the

XCIII **TimescapeGlobal** requires Java version 1.7 or later.

XCIV This is, aesthetically speaking, the worst *mixing* that occurs among **TimescapeGlobal** packages.

XCV The AnimatedGifEncoder class was written by Kevin Weiner (kweiner@fmsware.com) and imported as such in Timescape application. It is used only once.

list of `Event` used during the evaluation. In fact, an `Event` wraps a `Source` object[XCVI] and is designed for efficiency: During the evaluation the list of `Event` objects is instanced once and for all at the beginning, then for any voxel the distances are calculated, the list is rearranged according to distance and the sample points that are not causally connected are temporarily switched-off (but remain in the list). See section 3 for details.

13.3 Application GUI

Here are all the classes for user interaction, subdivided functionally and named accordingly. The `MainFrame`[XCVII] is invoked at startup by the `TimescapeGlobal` class. It carries the button array that pop-up the other windows.

The `manager` package contains the manager classes: model manager, database manager and hidden window, see section 6 for usage instructions. On the code side, it is worth noting that `MainFrame` has some spaghetti-like interaction with `ManagerFrame`, `HiddenFrame`, `SetupFrame`, `EvaluationFrame` and `SeekFrame`. This is because one has to keep track of what can be done for each model: something is hard-coded in the database (a locking flag mechanism) and the list of available actions need to be refreshed any time a model is modified (the `updateButtons` methods takes care of it). The database manager class is `SourceFrame`.

The `setup` package scope is the editing of a record in the `model` table. Put this way is seems diminutive, but this is all it does.[XCVIII] The main class is `SetupPanel`, contained in `SetupFrame`: in this class there are the command parser (the `parse` method) and an inner class for samples plotting. The GUI building blocks are standard swing objects.

The `TrendPanel` and `VariogramPanel` show, respectively, the sample points trend analysis and the variogram. These have nothing to do with the actual model setting, they are just an analytical help for the user to choose the right parameters.

The `HelpFrame` is just a collection of commands and the `ExpressionFrame` is the expression parser for checking the user custom functions.

The `seek` package corresponds to the `Exploration panel` of the GUI. The main class is `SeekFrame`, which is just a frame for the three panels `SeekPanel` (display end export of a model's voxels), `AnalysisPanel` (sheet-by-sheet statistics) and `ResidualPanel` (model's residuals). So also for the GUI's `Exploration panel` there is a function-class symmetry.

The `EvaluationFrame` class "breaks the symmetry" in that it is part of the GUI[XCIX] (see section 8) but in fact, functionally speaking, it is the evaluator of the model. This is not the kind of admixture a developer should be proud of, but it works. The reason for this functional mixing is that this way the evaluating thread can easily communicate with the GUI for showing a progressbar and other informations: the deal is to trade aesthetics for efficiency. The class is described in detail in the next section.

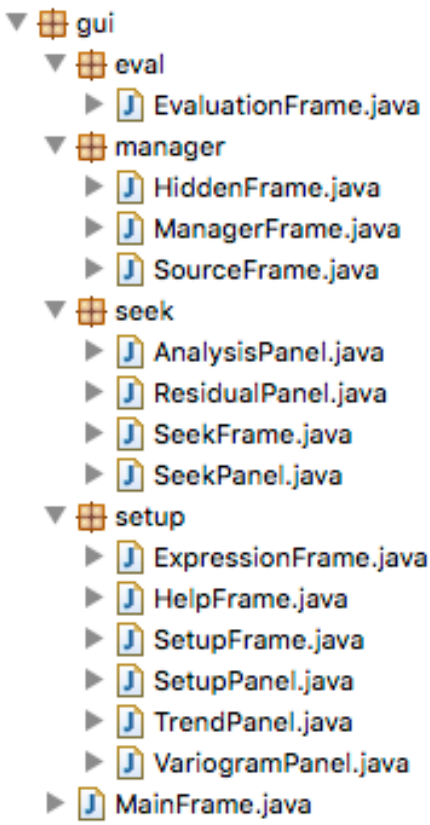


Fig. 36

13.4 Evaluation

The evaluation of a model involves more or less all the data Objects at the same time. Care is given, however, to keep separated the database input/output and the evaluation, as far as it is possible. A few more objects need to be considered:

- the `Distance` class: a collection of static methods for the evaluation of the functions related to the Timescape algorithm (see section 3).
- the `EvalException` an `Exception` extension; this is used to manage the error conditions that occurr during a model's voxels evaluation.
- the `EventPair` class is an `Event` wrapper, it is the backbone of the evaluation process.

Referring to the evaluation steps as described in section 8, the `EvaluationFrame` class has a method for each phase: pre-processing, empty voxels insertion and evaluation. The code is not particularly involved, it is mostly procedural (inside the methods), reflecting strictly the algorithm steps.

XCVI Or better the radians coordinate version of it.

XCVII All `*Frame` classes are extensions of `JFrame` and all `*Panel` classes are extensions of `JPanel`.

XCVIII As a matter of principle, one can write the model parameters values on the database directly. The GUI performs checks on all values preventing the insertion of inconsistent values.

XCIX The only class in the `evaluation` package.

13.4.1 Preprocessing

Just upon choosing e model for evaluation, the `prepare` method is called. The model is locked to prevent unwanted actions by other application panels;[C] the lattice points, i.e. the voxels centers, are calculated and stored in suitable variables. All the results of these calculations are shown to the user in the main text area of the `EvaluationFrame` window. The action button text becomes `Insert`, showing that the application is ready to insert the model's voxels.

Up to now, nothing has been inserted into the database. Quitting now the evaluation process just releases the lock on the model.

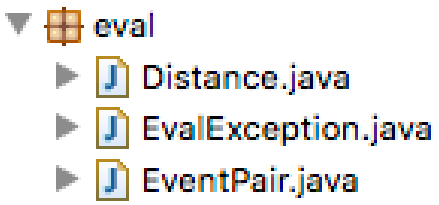


Fig. 37

13.4.2 Empty voxels insertion

Pressing the `Insert` button calls the `insert` method of the `EvaluationFrame` class: now the database is stressed.

All the voxels are inserted, one time sheet after the other, without any actual value calculation. During this phase the database inflates very fast, so it is advisable to insert voxels one model at a time only. Should the database run off of space, the procedure stops with an error condition, and it is not possible to go further. Quitting the procedure deletes the inserted voxels and releases the lock.

If the insertion completes correctly, the action button becomes `Evaluate`. During the insertion a progressbar runs and, after any time sheet insertion, a message appears in the main text area.

13.4.3 Voxels evaluation

Pressing the `Evaluate` button calls the `evaluate` method. Now the real business starts: first a list of `EventPair` is defined, the same for all voxel.

Next, inside a nested loop, voxels values are eveluated one at a time and the correstponding records of the database are updated. There is al lot of database transactional activity now too, but the tables do not inflate further, because the record length is fixed and no new records are needed. The `interpolate` method is called for each voxel:

```
 1 private Voxel interpolate(Voxel voxel, Event event) throws EvalException{
 2 try{
 3    int num = 0;
 4    double weight;
 5    double weightSum = 0.;
 6    double valueSum = 0.;
 7    voxel.setValue(null);
 8    for(EventPair pair : pairs) pair.setTo(event);
 9       Collections.sort(pairs);
10       for(EventPair pair : pairs){
11          if(pair.getDist() == null) break;
12          if(model.getNeighbours() != Model.NEIGHBOURS_ALL && num == model.getNeighbours()) break;
13          boolean test = model.getPickingProb() == 1. || Math.random() <= model.getPickingProb();
14          test &= model.getGeodesicThreshold() == Model.THRESHOLD_NONE ||
                    pair.getGeodesic() <= model.getGeodesicThreshold();
15          if(test){
16             weight = pair.getWeight();
17             valueSum += weight * pair.getValue();
18             weightSum += weight;
19             ++num;
20          }
21       }
22       if(num != 0) voxel.setValue(valueSum / weightSum);
23       if(voxel.getValue() != null && (voxel.getValue().isNaN() || voxel.getValue().isInfinite()))
              throw new EvalException("value " + voxel.getValue() + " out of range");
24       return voxel;
25    }
26    catch(Throwable ex){ throw new EvalException(ex); }
27 }
```

The most important instruction is the seemingly innocent `Collections.sort(pairs)`, line 11: this actually picks the sample points which are causally connected to the voxel under evaluation and puts them in order of space-time distance (closer points come first). In fact, dependong on the user's choice, only the first N_{max} poinst are actually used in the interpolation.

This procedure is processor-baking, in particular the instruncion at line 8 forces the calculation of the distances from all the sample points to the given voxel: it is about a few tens floating point operations for each sample point for each voxel. That said, nothing happens with the database until the end of the procedure.

C If the model were opened for setup in another application window, the changes will not be recorded.

Line 12 checks the N_{max} threshold; lines 13 to 15 check whether the voxel lies within one or more causal cones emanating from its nearest[CI] source points. Lines 16 to 19 accumulate the value and weight funcion and line 22 sets the voxel value.

If everything is good line 24 returns the voxel value (it can be null), otherwise line 26 throws an `EvalException`; this in turn is captured and processed, giving rise to an error condition that is reported to the user.

A note on variables names: an `Event` is a point in spacetime. This can be both a sample point (`Source`) or a `Voxel`. Anytime the term *event* is found in the code, it is referred to some kind of spacetime coordinate triple.

14 Exotic usage of Timescape

Timescape Global has been designed with the Earth in mind, nothing prevents, however, its usage with the other lballs floating out there. The basic assumption is that the object is approximately spherical, so that the distances can be measured as arcs of circles. The only parameter that needs to be set is the radius (see section 5.4).

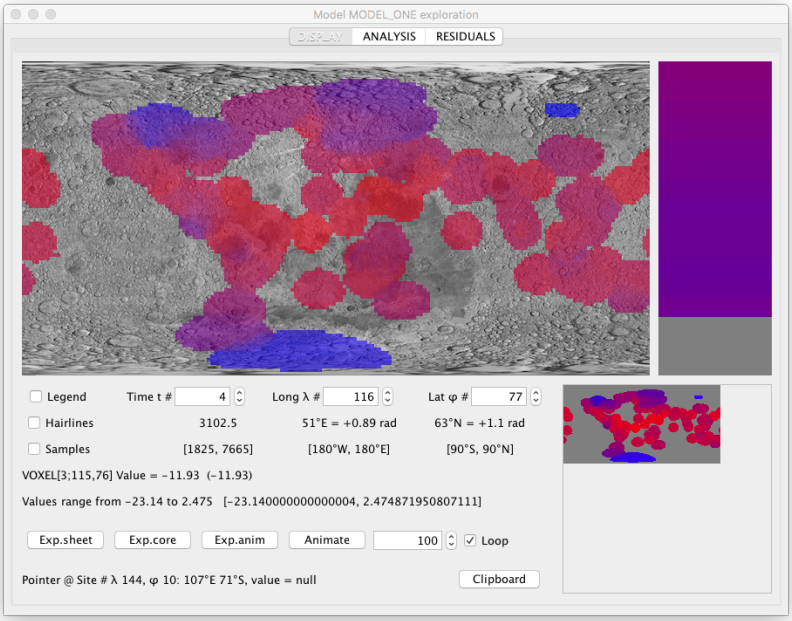


Fig. 38

Setting a radius e.g. $R = 1737.1$ km for the Moon (above) or $R = 3389.5$ km for Mars (below) is all what one needs to do. Just an aesthetic concern: it is not advisable to plot such exotic surfaces over the Earth's planisphere.

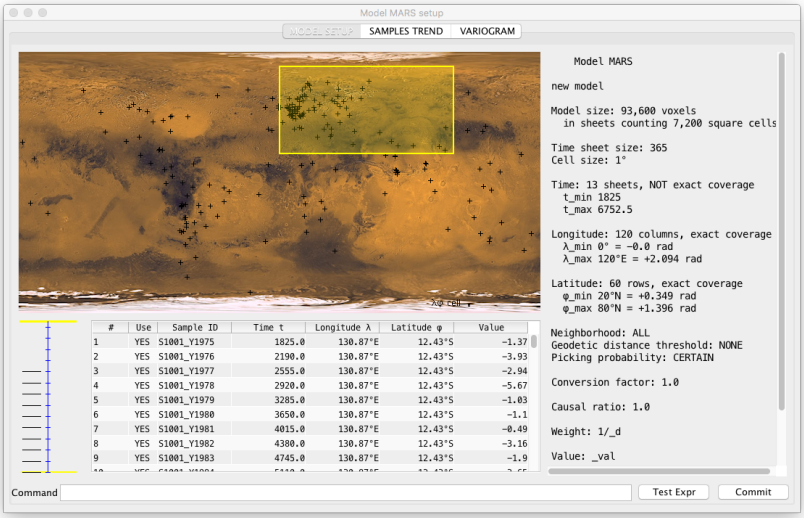


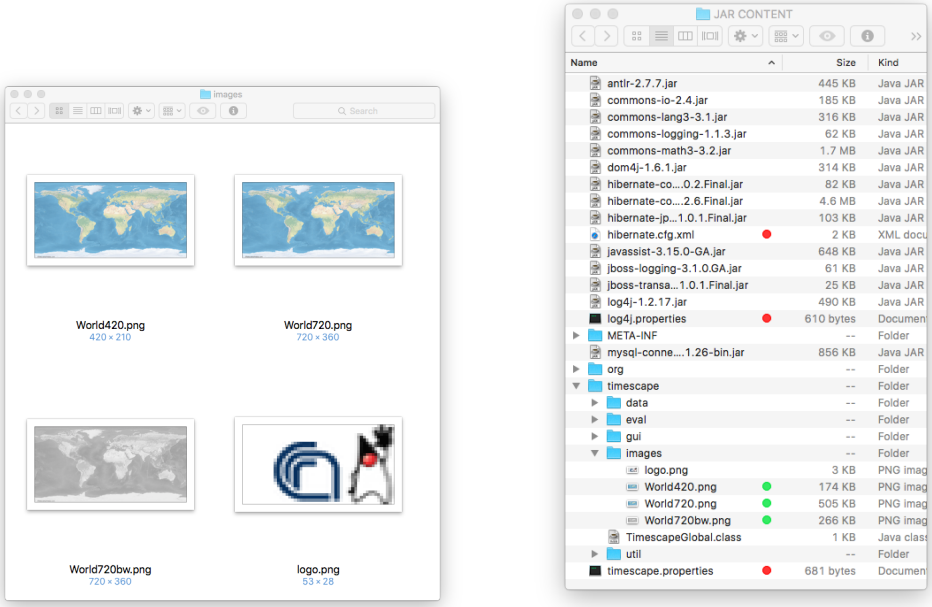
Fig. 39

CI *Nearest* should be understood in space-time terms.

It is possible to change the graphic layers underneath the plotting panels of the application. It requires a bit of surgery but is a harmless operation: first find a suitable image of the desired planet, this must span the entire longitude and latitude intervals. Prepare the image so that the upper left corner corresponds to $\lambda = 180°W$ and $\varphi = 90°N$ The image should have a width which is twice its height, with a reasonable resolution.

From this image, prepare the following files:

- scale the image to 720 (w) × 360 (h), convert it to .png format and label[CII] it `World720.png` (this image is used in the **Setup panel**)
- make a copy of the latter called `World720bw.png`, convert it to grayscale and apply a contrast lowering filter (this image is used in the **Display panel**)
- make a copy of `World720.png` rescaled to 420 (w) × 210 (h) called `World420.png` (this image is used in the **Residual panel**)



(a) **(b)**

Fig. 40

Now the surgery: open up the application .jar file (see section 5.4) and locate the `images` directory inside the main `timescape` classes folder. Substitute the original images (marked with a green dot in the figure above) with the newly created ones. Pack it all into a new .jar: that's all. The user who wants to personalize his/her copy of **Timescape Global** can also change the logo icon,[CIII] it is allowed by the licensing policy (see section 2).

Note the configuration files marked with red dots: they can be tuned via the **Configurator** application (see section 5.5) but, having the .jar unpacked, the user will probably edit them directly by hand. The radius parameter is located in the `timescape.properties` file.

15 Sample data

TimescapeGlobal comes with a sample dataset of stable isotopes in precipitations. This dataset comes form the Global Network of Isotopes in Precipitation [6], it covers roughly the Mediterranean area, from January 1970 to December 1999.

CII Names are case sensitive.
CIII It must be a 53 (w) × 28 (h) .png icon called `logo.png`.

Fig. 41

The red dots correspond to the spacial location of the dataset records. The main `VALUE` field contains the $\delta^{18}O\ {}^{0}/_{00}$ values.[CIV] The example file id `gnip_med.csv`; the file head has a few comment lines about the content, which is actually made of a few thousands of records:[CV]

```
#########################################################################
# Data from the Global Network of Isotopes in Precipitation
# url http://www-naweb.iaea.org/napc/ih/IHS_resources_gnip.html
# Mediterranean Area (8 deg W to 29 deg E, 28 deg N to 47 deg N)
# Years from 1970 to 1999
# VALUE is referred to delta 18O (permil)
# Ancillary fields:
#   YEAR year 1970 to 1999
#   MONTH month 1 to 12
#   ELEV elevation (m)
#   D2H delta 2H (permil)
#   PREC precipitation of the month (mm)
#   TEMP mean temperature (deg Celsius)
#   PRES vapour pressure (mb)
# ID schema ccppyymm
#   cc country
#   pp place within country (ccpp is unique)
#   yy year, last two digits
#   mm month, two digits
#########################################################################
ID,TIME,LONG,LAT,VALUE,YEAR,MONTH,ELEV,D2H,PREC,TEMP,PRES
ATGZ7301,1080,15.45,47.08,-16.26,1973,1,366,-121.90,40.0,,
ATGZ7302,1110,15.45,47.08,-9.97,1973,2,366,-66.70,29.0,,
        ...
```

The ID is a label composed as `ccppyymm`, where `cc` is a country code, `pp` labels a place within the country, `yy` is the year and `mm` is the month of observation. The time is expressed in days form the first day of the year 1970, longitude and latitude are given in decimal degrees (two significant digits). Ancillary data may or may not be known for each sample record.

This recordset exhibits interesting variations of the values over time, there is both a seasonal (within-year) and a global (year-by-year) pattern, as the images below suggest:

CIV This is the relative abundance of $^{18}O/^{16}O$, with respect to a given standard, generally quoted in ‰ units.

CV The data parser ignores all comment lines beginning with the # sign.

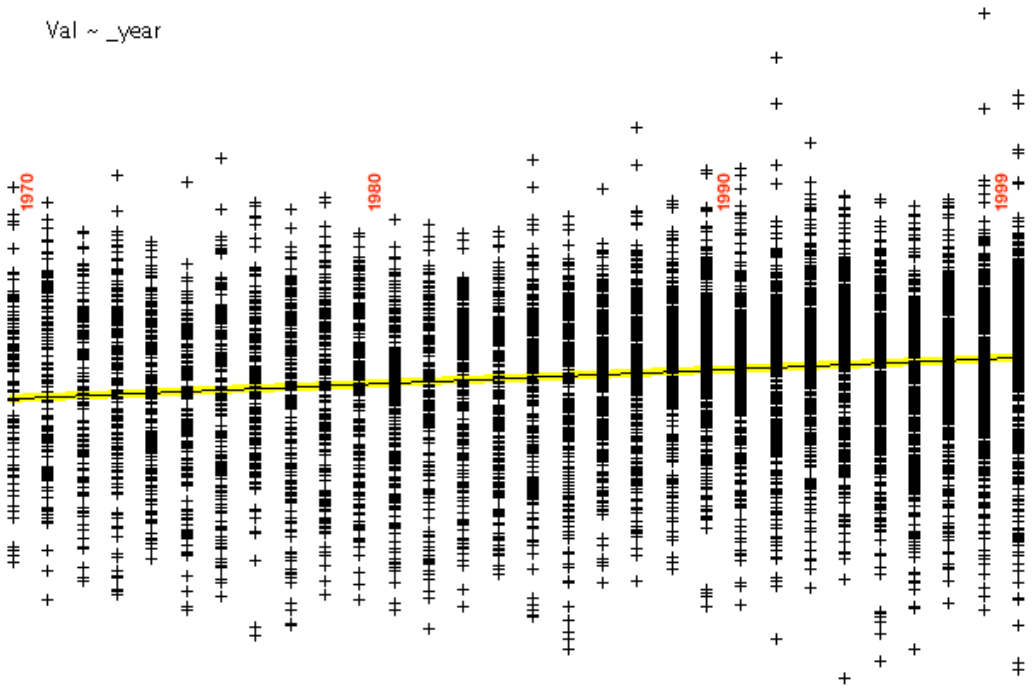


Fig. 42

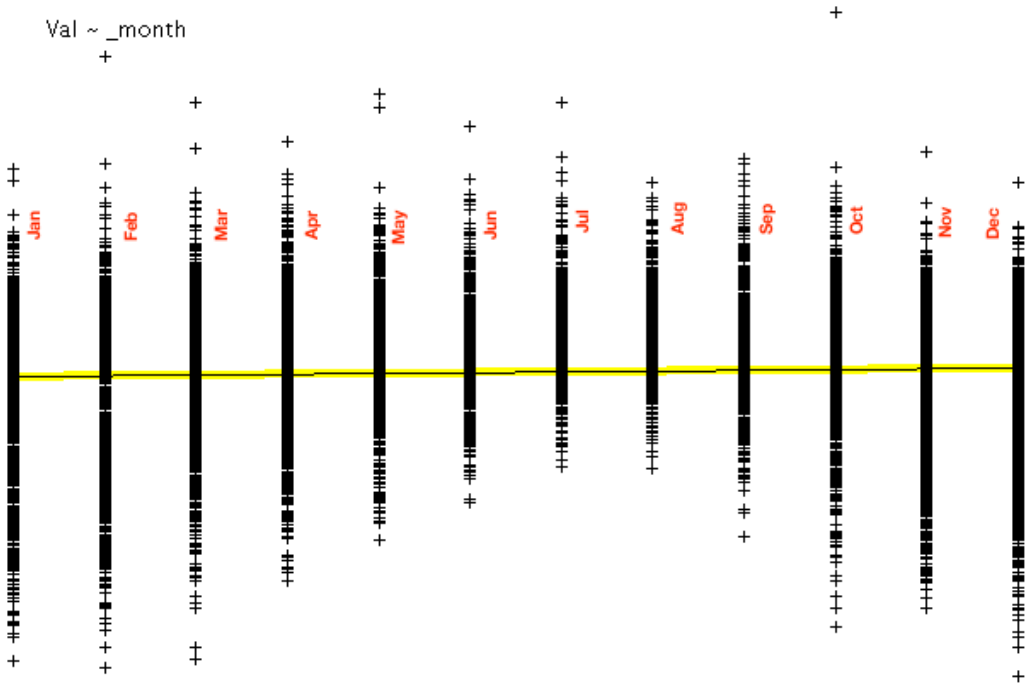


Fig. 43

These images have been exported from the TREND analysis panel (see section 7.3). The linear trend is not significant for the monthly variations, while there is a slight increase of the values with time, but among-years variations are greater.

The spatiotemporal distribution of data is sketched in the following image

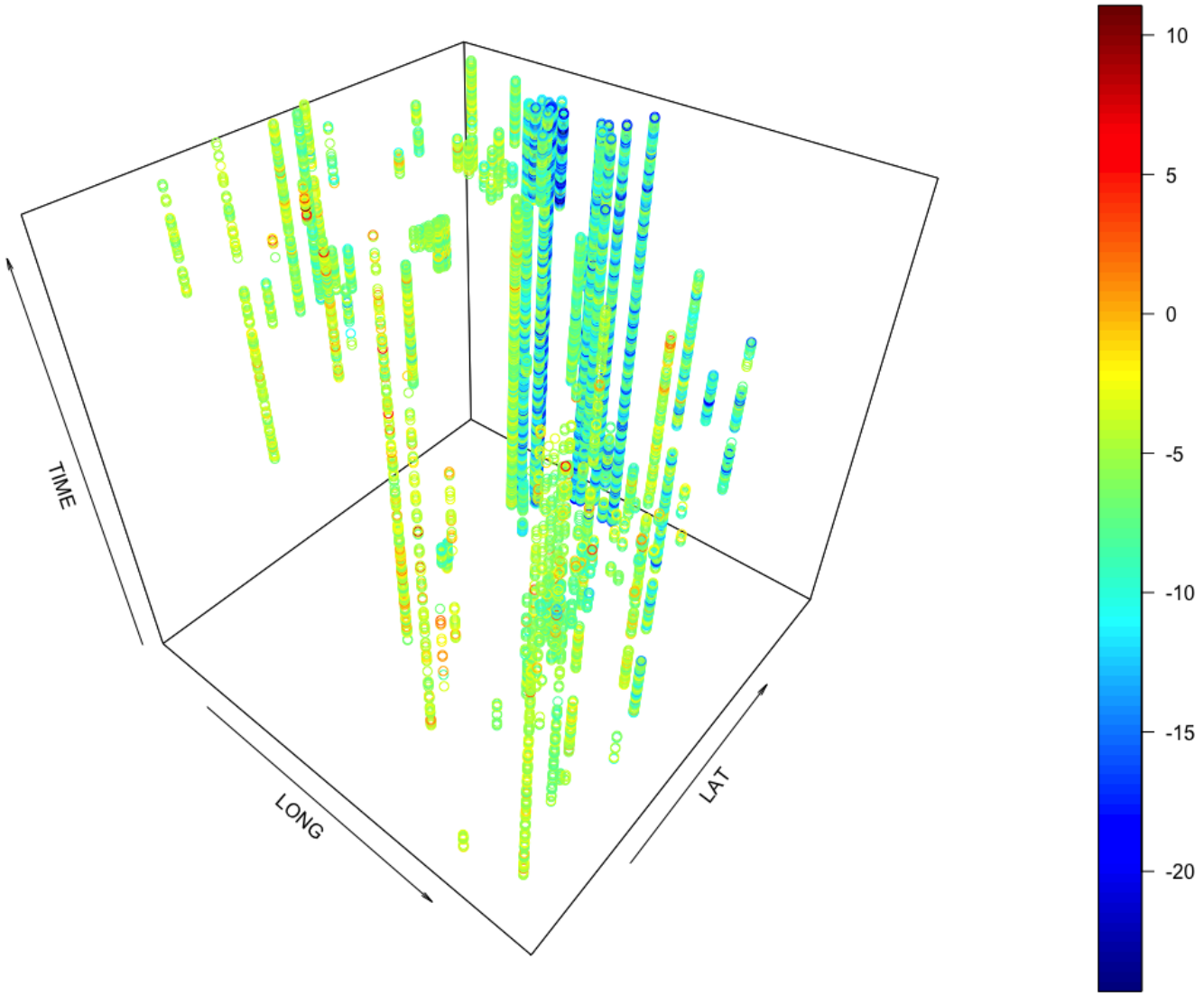


Fig. 44

The colours are referred to the values (see the scale on the right). Time is plotted on the vertical axis, while longitude and latitude are plotted (flat) horizontally. This is the typical distribution of repeated measurements in the same place, as it often occurs in ecological studies. The above scatterplot has been obtained in R [9,10].

References

1 N. A. C. Cressie, Statistics for Spatial Data, John Wiley & sons, 1990.

2 N. A. C. Cressie, C. K. Wikle, Statistics for Spatio-temporal Data, in: Probability and Statitics.

3 The TimescapeLocal project site: https://sourceforge.net/projects/timescapelocal/.

4 M. Mattioni, M. Ciolfi, F. Chiocchini, M. Lauteri, Timescape local, SMART eLAB 10 (2018) 19–38. doi:10.30441/smart-elab.v10i0.201.

5 GNU General Public License: http://www.gnu.org/licenses/gpl-3.0.en.html.

6 Global Network of Isotopes in Precipitation (GNIP) of the International Atomic Energy Agency (IAEA): http://www-naweb.iaea.org/napc/ih/IHS_resources_gnip.html.

7 An ECMAScript reference: http://www.ecma-international.org/publications/standards/ Ecma-262.htm.

8 QGIS open source Geographic Information System: http://www.qgis.org.

9 The R Project for Statistical Computing: https://www.r-project.org.

10 RStudio is an open source user-friendly way to use R: https://www.rstudio.com.

11 Rob J Hyndman (Rob.Hyndman@monash.edu) CRAN Task Time Series Analysis: https://cran.r-project.org/web/views/TimeSeries.html.

12 Apache [TM]logging services: https://logging.apache.org.

13 Oracle ™software download main page:
http://www.oracle.com/technetwork/indexes/downloads/ It also allows for the download of MySQL database management system.

14 The SPATIAL (SPAtio-Temporal Isotope Aanalytics Lab) group, University of Utah:
http://wateriso.utah.edu/spatial/.

15 Isoscapes, Understanding Movement, Pattern and Process on Earth Through Isotope Mapping, Springer, 2010.