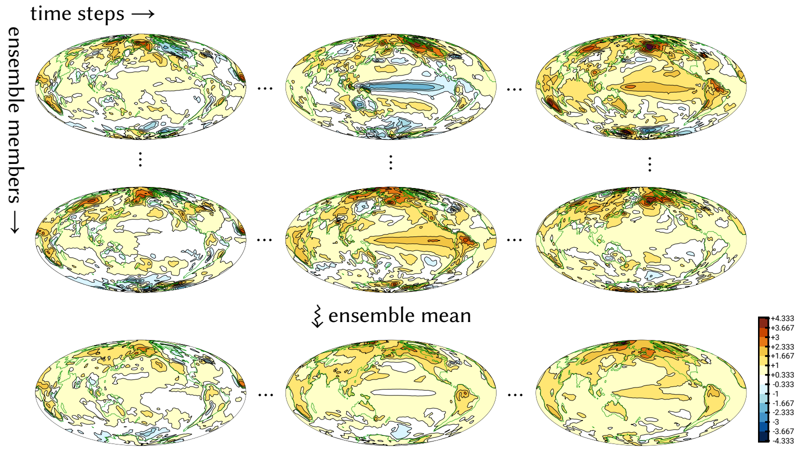
Visual exploration of ensemble variability at the example of decadal climate predictions

Christopher Kappe*, Michael Böttinger†, Heike Leitte*
*TU Kaiserslautern, †Deutsches Klimarechenzentrum
kappe@cs.uni-kl.de

EGU2018 – PICO

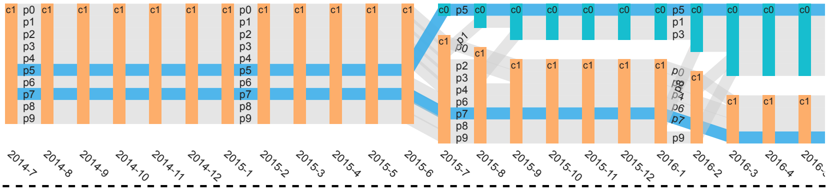
April 9, 2018

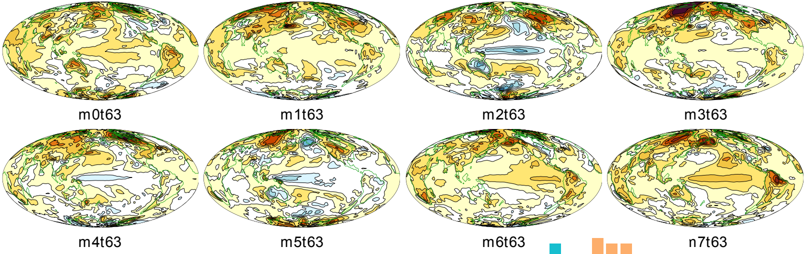
Introduction

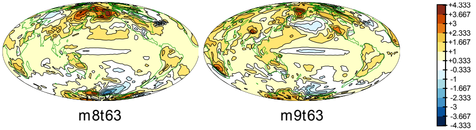


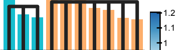
- application data: climate simulation ensembles ($n_m \approx 15$), decadal time scale ($n_t \approx 120$), temperature anomaly and other variables
- www.fona-miklip.de

Teaser





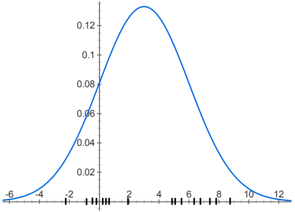


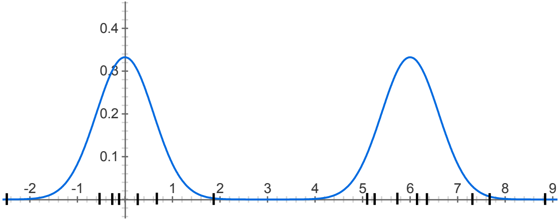


p	7	6	0	4	5	2	3	8	1	9
7	0.000	0.455	0.485	0.721	0.860	0.896	0.666	0.586	0.584	0.605
6	0.455	0.000	0.506	0.692	0.829	0.806	0.617	0.561	0.538	0.590
0	0.485	0.506	0.000	0.686	0.830	0.821	0.668	0.576	0.605	0.517
4	0.721	0.692	0.686	0.000	0.524	0.506	0.557	0.599	0.543	0.557
5	0.860	0.829	0.830	0.524	0.000	0.522	0.694	0.672	0.710	0.670
2	0.896	0.806	0.821	0.506	0.522	0.000	0.704	0.737	0.674	0.658
3	0.666	0.617	0.668	0.557	0.694	0.704	0.000	0.543	0.554	0.567
8	0.586	0.561	0.576	0.599	0.672	0.737	0.543	0.000	0.554	0.456
1	0.584	0.538	0.605	0.543	0.710	0.674	0.554	0.554	0.000	0.557
9	0.605	0.590	0.517	0.557	0.670	0.658	0.567	0.456	0.557	0.000

Challenge

- The mean of an ensemble need not be a good representation of the actual simulation outcomes.
- In the example below the “simulation output” is a single number, so each “ensemble member” can be drawn as a tick on the x-axis.
- The numbers happen to have mean 3 and standard deviation 2, as it was used to draw the normal distribution in the left figure. However, the bi-modal distribution in the right figure describes the data much better.





- We aim at such a smart analysis but for time-dependent scalar fields.

Solution

1. Compute a *clustering* of all the fields to identify distinct outcomes of the simulation and classify the ensemble members accordingly.
   - Needs a suitable clustering algorithm, e.g. k-means or hierarchical clustering.
   - Needs a distance/similarity measure for scalar fields, e.g. Mean Squared Error or Pearson Correlation Coefficient.
2. Provide a comprehensive interactive *visualization* of the results.
   - Show which ensemble member belongs to which cluster at which time step $\longrightarrow$ clustering timeline.
   - Support human evaluation by showing the data behind the clustering such as the $n_m \times n_m$ distance matrix.
   - Make various colormap visualizations available on demand, e.g. for individual ensemble members, ensemble/cluster mean or standard deviation.

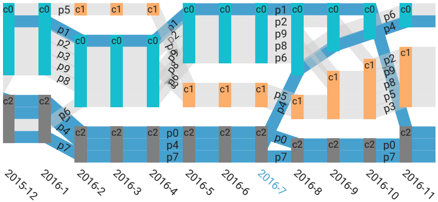
Results 1/2

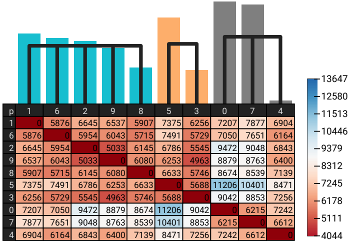
The clustering timeline is a kind of flow diagram that visualizes how the ensemble can be divided into groups and how ensemble members change groups over time.

ensemble members: lines from left to right.
clusters: rectangles of the same color.

Members can be highlighted to trace their path easier.

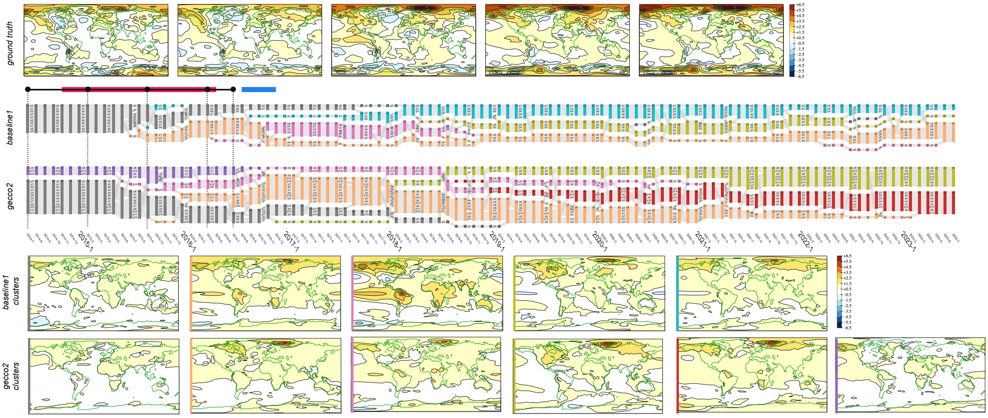
Bars showing how well each member fits into its cluster (silhouette coefficient) are plotted above a heatmap of the pairwise distances.
Black combs represent the overall quality of a cluster.





p	1	6	2	9	8	5	3	0	7	4
1	0	5876	6645	6537	5907	7375	6256	7207	7877	6904
6	5876	0	5954	6043	5715	7491	5729	7050	7651	6164
2	6645	5954	0	5033	6145	6786	5545	9472	9048	6843
9	6537	6043	5033	0	6080	6253	4963	8879	8763	6400
8	5907	5715	6145	6080	0	6633	5746	8674	8539	7139
5	7375	7491	6786	6253	6633	0	5688	11206	10401	8471
3	6256	5729	5545	4963	5746	5688	0	9042	8853	7256
0	7207	7050	9472	8879	8674	11206	9042	0	6215	7242
7	7877	7651	9048	8763	8539	10401	8853	6215	0	6612
4	6904	6164	6843	6400	7139	8471	7256	7242	6612	0

Results 2/2



Comparative analysis of multiple simulation models: (top) ground truth for the time range in the past. Dots on the timeline indicate the respective time point. Color encodes observed El Niño (red) and La Niña (blue) events. (center) Clustering timelines for two models. (bottom) Cluster centers of the two ensembles sorted with respect to matching patterns.

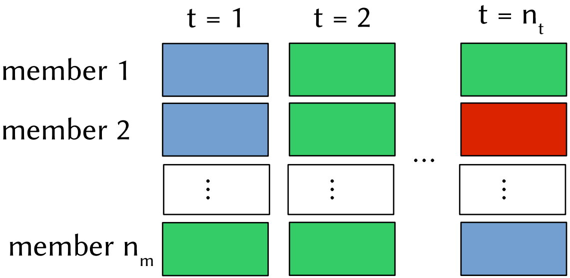
Computation 1/2

- We used k-means to compute the clustering.
  This algorithm begins with some (random) cluster centers, which in our case are hypothetical scalar fields, and assigns each field the cluster to whose center it is nearest. Then the cluster centers are recomputed taking into account the possibly changed assignments. This process is repeated until there are no more changes (it always converges to a local optimum).
- Reasonable results were gained using the simple Manhattan distance as a measure of how near two fields p and q are:
  $\mathrm{d}(p, q) = \sum_{i=1}^{n_v} |p_i - q_i|$,
  where n_v is the number of vertices of the underlying grid.
- The user can set the number of clusters k manually or let the computer decide for which k the distance between clusters is maximal while the distance among members of the same cluster is small.

Computation 2/2

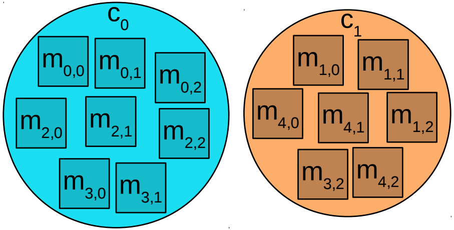
Considering the fact that the different climate states that we suspect to be in the dataset are rather static, while ensemble members may shift from one to the other over time, we do not cluster the n_m time-dependent fields of the ensemble directly but split them into the $n_m \times n_t$ steady fields for clustering.

Schematically a clustering result may look like this (color indicates cluster affiliation):

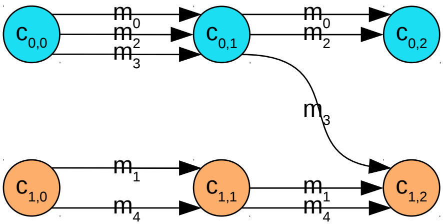


Visualization

We can model the clustering result as a graph and visualize it accordingly. Therefore a variant of the Sugiyama layout algorithm is applied, that distributes the cluster nodes in vertical layers, each representing a time step, while the number of edge crossings is minimized.



various fields (indexed by member ID and time step) in their cluster



the clusters and members laid out with respect to time

Implementation

- We have implemented the described software as a web application based on HTML, SVG, CSS and JavaScript.
  (Currently it runs entirely on the client CPU.)
- Intermediate results like the clustering can be exported so that users may continue an earlier session anytime anywhere.
- The internal architecture is modular so that e.g. the clustering algorithm or distance function can be swapped easily.
- The graphical user interface allows a flexible configuration of the various views.
  - Adding, removing, toggling visibility of views.
  - Panning, zooming, scrolling within views.