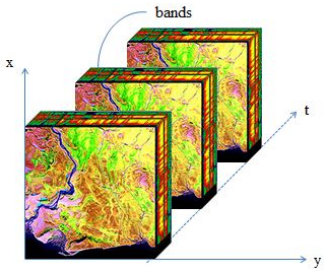
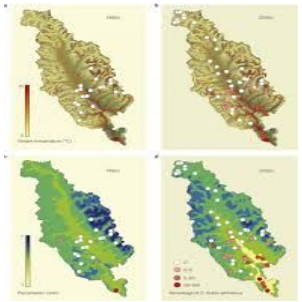
Scalable spatiotemporal data analysis from multidimensional data





Meng Lu, Marius Appel

Spatio-temporal Modeling Lab,
Institute for Geoinformatics,
University of Muenster, Germany

April 27, 2017

Short course overview

Part I: Analysing geoscientific arrays: Overview of data and software

Part II: Hands-on: array-based spatiotemporal change detection with R

Part III: Scalable data management and analysis with SciDB

Part IV: Demonstration: empirical orthogonal function analysis with SciDB

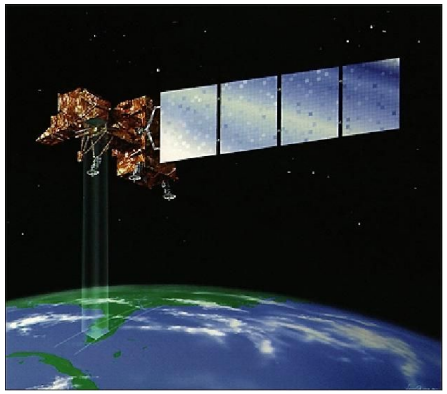
Short course overview

Slides and demonstration scripts will be available at:

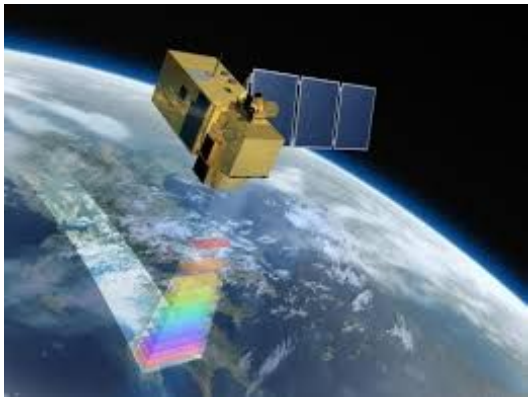
https://github.com/mengluchu/EGU-2017-Scalable-array-analysis

Geoscientific data:

Massive, diverse, and of high dimensionality



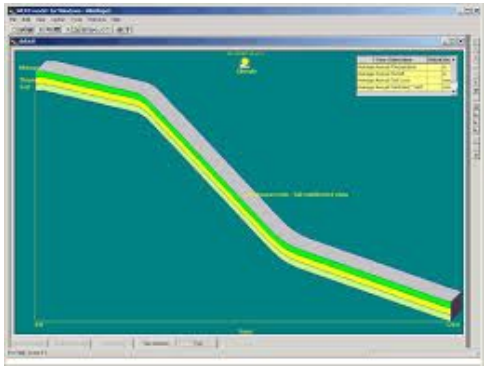
Landsat 7



Sentinel 2

cubesat

Earth observations

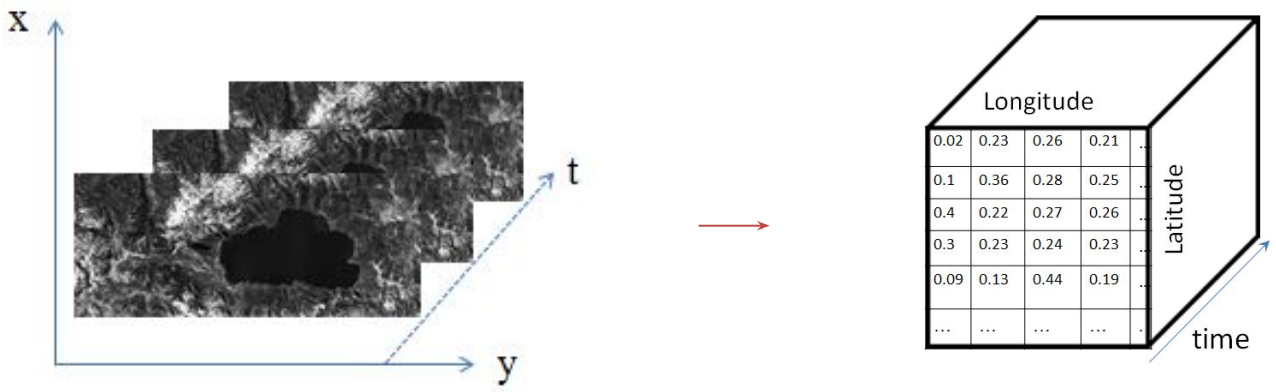


***in-situ* measurements, model simulations, experiment outputs**

Array Structure

Definition:

- A set of elements which are ordered in discretized space.
- A function that maps the Cartesian product of multiple discrete, totally ordered, and finite dimensions to a multidimensional attribute space.



Arrays in Data Management and Analytical Software

	Software / Language	Specific features
Focus on complex analytics	R	• only single-attribute arrays
	Python (NumPy)	• only single-attribute arrays
	GNU Octave	• only single-attribute arrays
	…	
Focus on data Management	Rasdaman	• various indexing methods • efficient external storage
	SciDB	• interfaces ScaLAPACK, • stores arrays in distributed memory environments by chunking • sparse storage
	…	

Arrays in Data Management and Analytical Software

Example: A is a three-dimensional spacetime array, compute mean values for all pixel time series:

Python (NumPy)

```
numpy.apply_along_axis(numpy.mean,0, A)
```

R

```
apply(A,c(2,1), mean)
```

SciDB

```
aggregate(A, avg(val), x, y)
```

Rasdaman

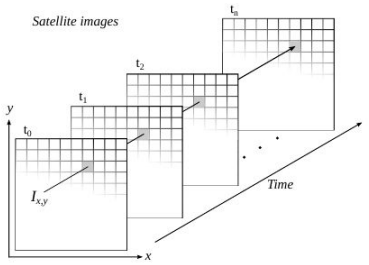
```
select marray B in [0:nx,0:ny]
values condense + over y in [0:nt]
  using avg_cells(A[B[0], B[1], *:*]) / nt
from A
```

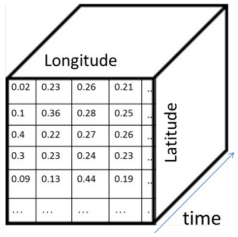
A scalable spatio-temporal change modeling process with multi-dimensional array

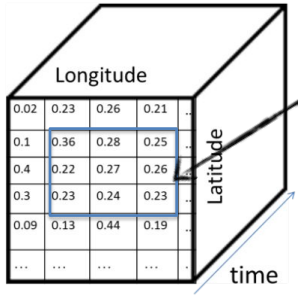
data management

spatio-temporal change detection

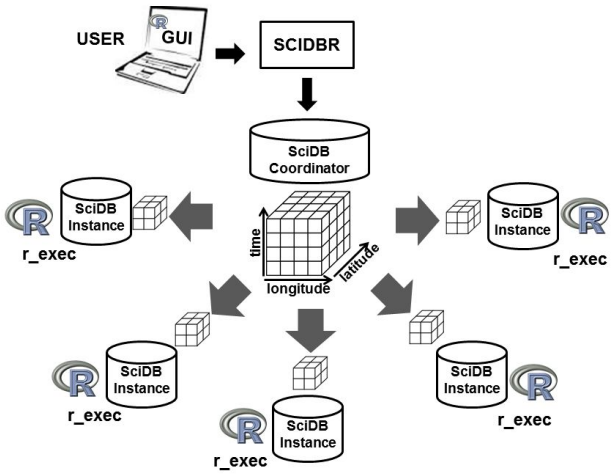
scale the analysis







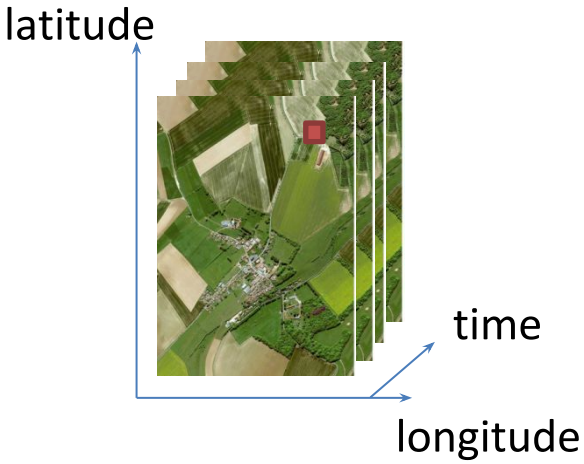
3-D spatio-temporal change detection algorithm

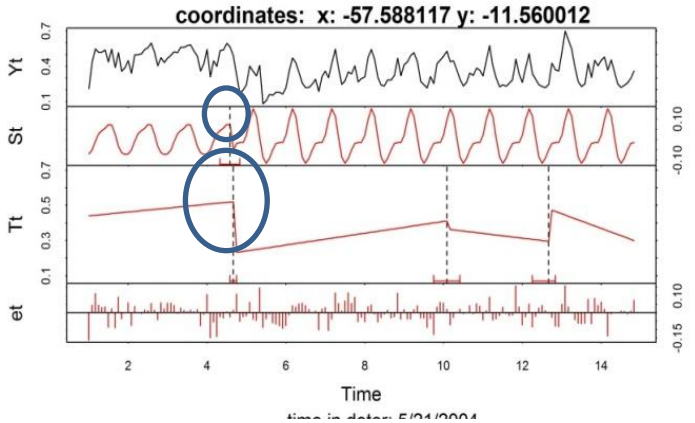


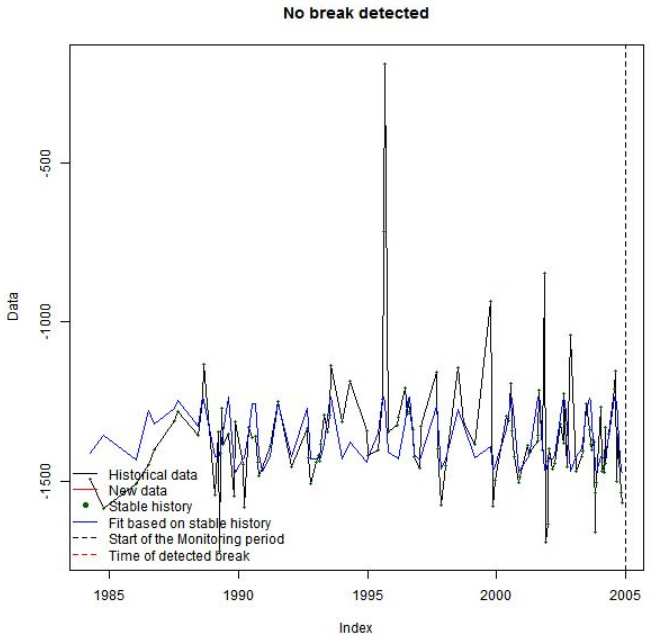
Time series structural change

Three steps:
1. constructing a vegetation index
2. pixel-wise time series analysis
3. summarise the results in space

Example:
BFAST (Breaks For Additive Season and Trend) R package based on Strucchange R package

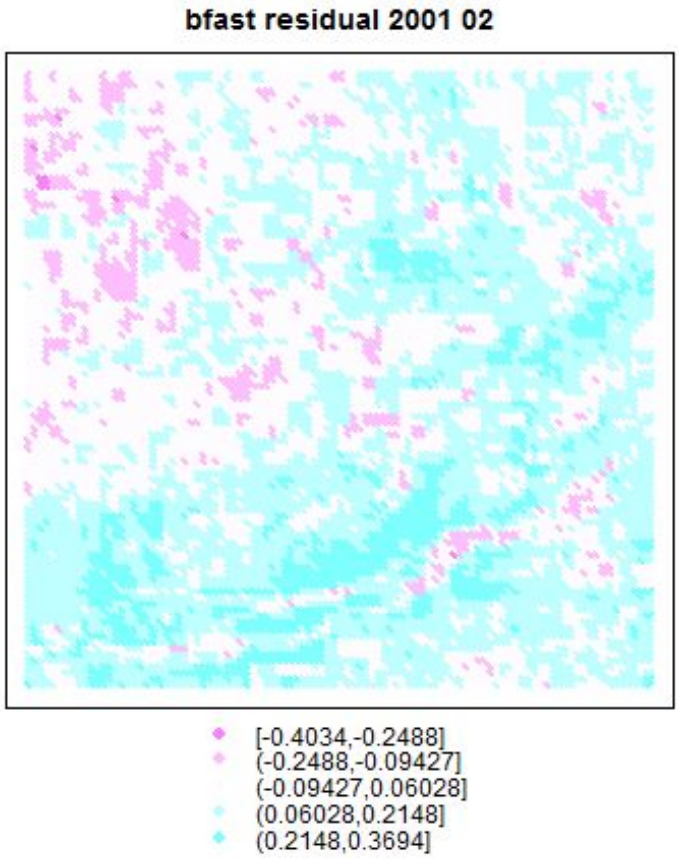




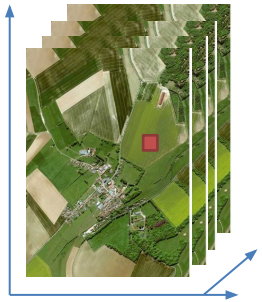


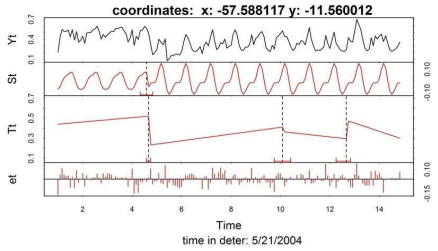
Spatial dependence

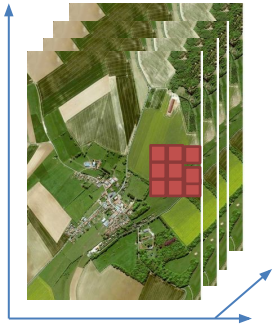
Strong spatial correlation between *time series* residuals

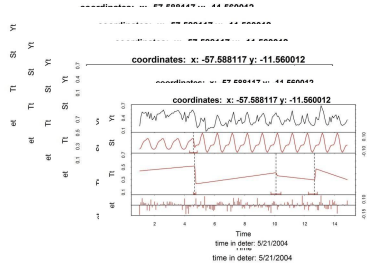


From time to space-time









Short course overview

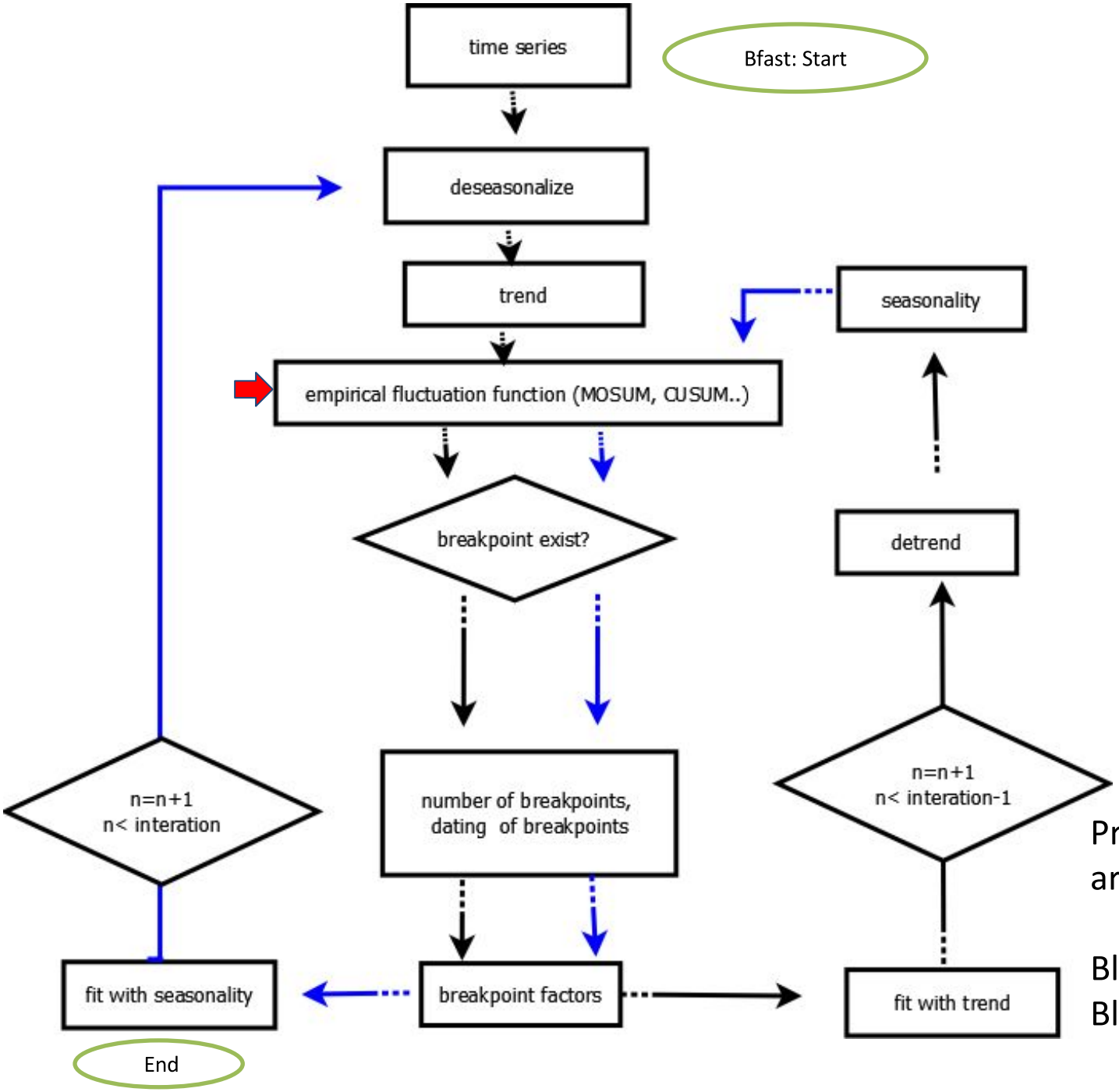
Part I: Analysing geoscientific arrays: Overview of data and software

Part II: Hands-on: array-based spatiotemporal change detection with R

Part III: Scalable data management and analysis with SciDB

Part IV: Demonstration: empirical orthogonal function analysis with SciDB

Bfast Workflow



Procedures follow the arrow of the same color

Blue: seasonality;
Black: trend

Integrate Simultaneous Autoregressive Model (SAR) in the MOSUM/ CUSUM process of *BFAST*

Consider linear regression on each pixel of a spatio-temporal cube:

$$y_{ij} = \beta_j x_i + u_{ij} \qquad (i = 1, \dots, n;\ j = 1, \dots, m)$$

Model the dependence between the residuals with SAR:

$$u_{ij} = B\, u_{ij} + v_{ij} \qquad B = \rho W$$

B is the spatial dependence between residuals.

The spatio-temporal model:

$$y_{ij} = \rho W y_{ij} + \beta_j \rho W x_{ij} + \beta_j x_{ij} + v_{ij}$$
$$(i = 1, \dots, n;\ j = 1, \dots, m)$$

Short course overview

Part I: Analysing geoscientific arrays: Overview of data and software

Part II: Hands-on: array-based spatiotemporal change detection with R

Part III: Scalable data management and analysis with SciDB

Part IV: Demonstration: empirical orthogonal function analysis with SciDB

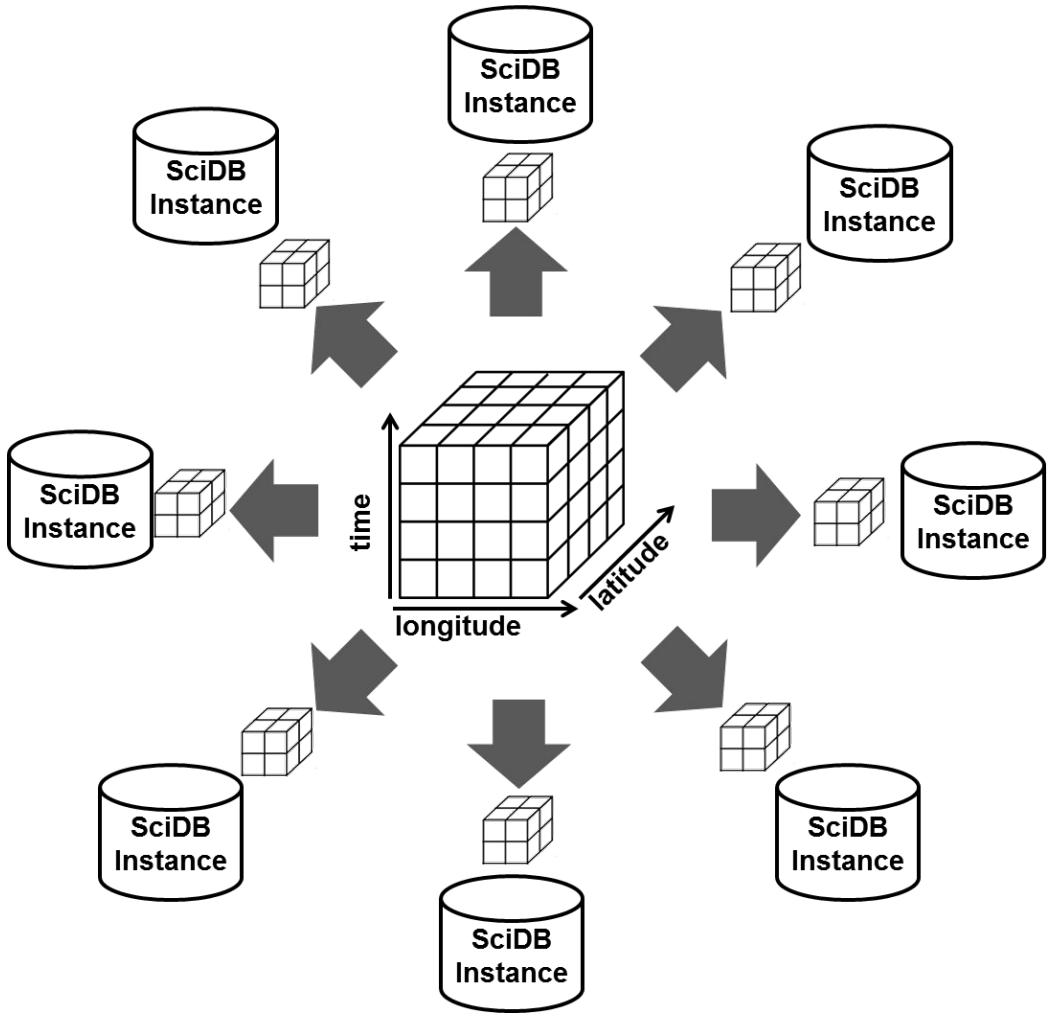
SciDB introduction

- Array-based data management and analytical system [1]
- Runs on single computers as well as on large clusters
- Open-source version available
- Basic data representation as **multidimensional arrays**
- n dimensions, m attributes (with different data types)

[1] Stonebraker, M., Brown, P., Zhang, D., & Becla, J. (2013). SciDB: A database management system for applications with complex analytics. *Computing in Science & Engineering*, *15*(3), 54-62.

Distributing arrays by chunking

- arrays are divided into equally sized chunks
- chunks are distributed over many SciDB instances
- instances may run on the same or different machines in a shared nothing cluster
- distributing storage and computational load
- sparse storage



Query language and functionality

- SciDB query language: Array Functional Language (AFL)

- Native functionality:
  - Load / write arrays from / to files
  - Arithmetic operations
  - subsetting by dimensions and or attributes
  - Aggregations (window, aggregate)
  - Array joins
  - Changing array schemas (repartitioning, redimensioning)
  - Linear algebra routines: (GEMM, GESVD, basic statistics, ScaLAPACK)
  - …

SciDB: extensions for Earth observation data

SciDB

- can load data from CSV and custom-binary files only
- does not understand spatial / temporal reference of arrays

- spacetime extensions [1]:
  - scidb4geo
  - scidb4gdal

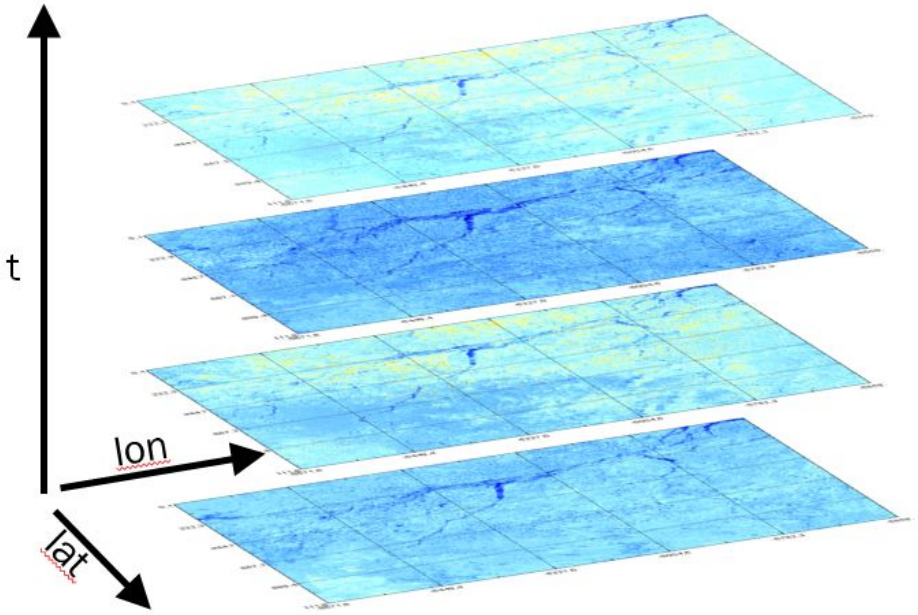
[1] Appel M., Lahn F., Pebesma E., Buytaert W., Moulds S. (2016). Scalable Earth-observation Analytics for Geoscientists: Spacetime Extensions to the Array Database SciDB. accepted for poster presentation at EGU General Assembly 2016, Vienna, Austria April 17-22, 2016.

SciDB: extensions for Earth observation data

New AFL (Array Functional Language) operators	
Operator	**Description**
`eo_arrays()`	Lists geographically referenced arrays
`eo_setsrs()`	Sets the spatial reference of existing arrays
`eo_getsrs()`	Gets the spatial reference of existing arrays
`eo_extent()`	Computes the geographic extent of referenced arrays
`eo_settrs()`	Sets the temporal reference of arrays
`eo_gettrs()`	Gets the temporal reference of arrays
`eo_setmd()`	Sets key value metadata of arrays and array attributes
`eo_getmd()`	Gets key value metadata of arrays and array attributes
`eo_over()`	Overlays two arrays by space and / or time

SciDB: extensions for Earth observation data

- GDAL driver supports ingestion and download of images to and from SciDB
- GDAL supports > 100 raster formats
- ingestion automatically combines images by space and time (mosaicing)



Interfacing R / Python with SciDB

As a client: packages `scidb`[1] and `scidbst`[2] work with proxy objects and lazy evaluation → starts computations when you want to read the data

- overwrites R methods, e.g. %*%
- limited to native SciDB functionality

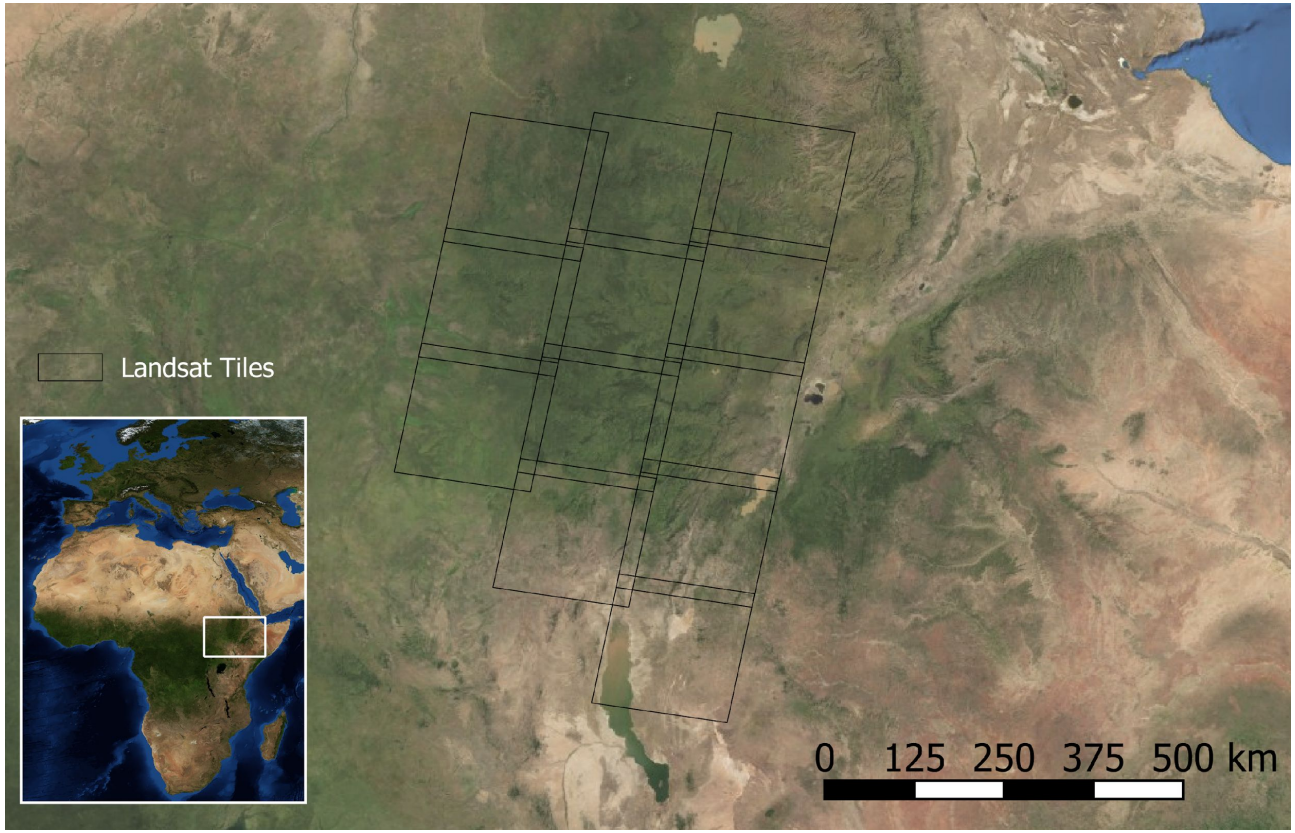
Within SciDB queries: `stream`[3] and `r_exec`[4]

- apply arbitrary scripts in parallel on chunks

[1] https://github.com/Paradigm4/SciDBR
[2] https://github.com/flahn/scidbst
[3] https://github.com/Paradigm4/stream
[4] https://github.com/Paradigm4/r_exec

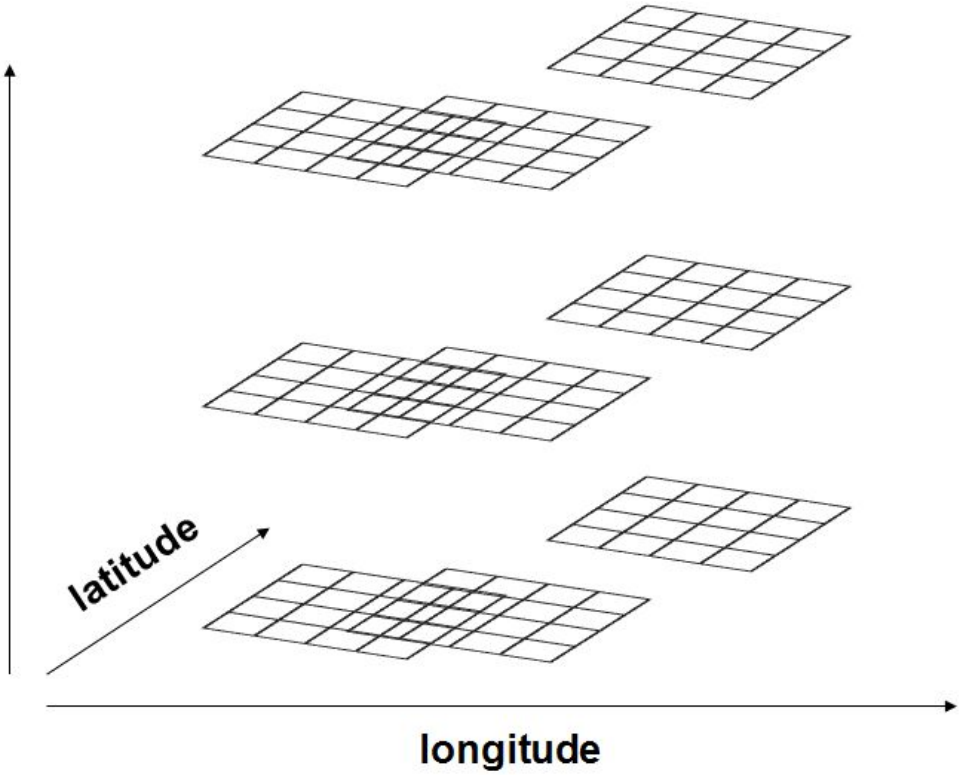
Study case: Landsat 7 in south west Ethiopia

- Landsat 7 data from 12 tiles captured between 2003-07-21 and 2014-12-27 ⟶ 1975 scenes
- Derived NDVI product from ESPA
- approx. 325,000 km^2
- monitor changes after 2010-01-01 using Breaks for Additive Season and Trend (BFAST)



Landsat 7 data in SciDB

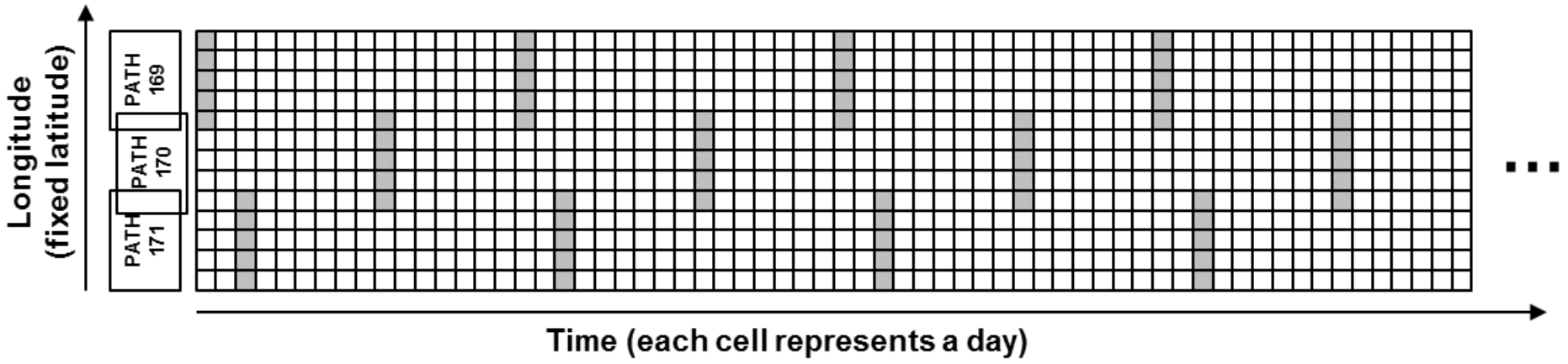
- Chunks contain complete time series of 64x64 pixels
- Preprocessing:
  - remove any values <= -9999 or >10000
  - unscale to -1, 1



Landsat 7 data in SciDB

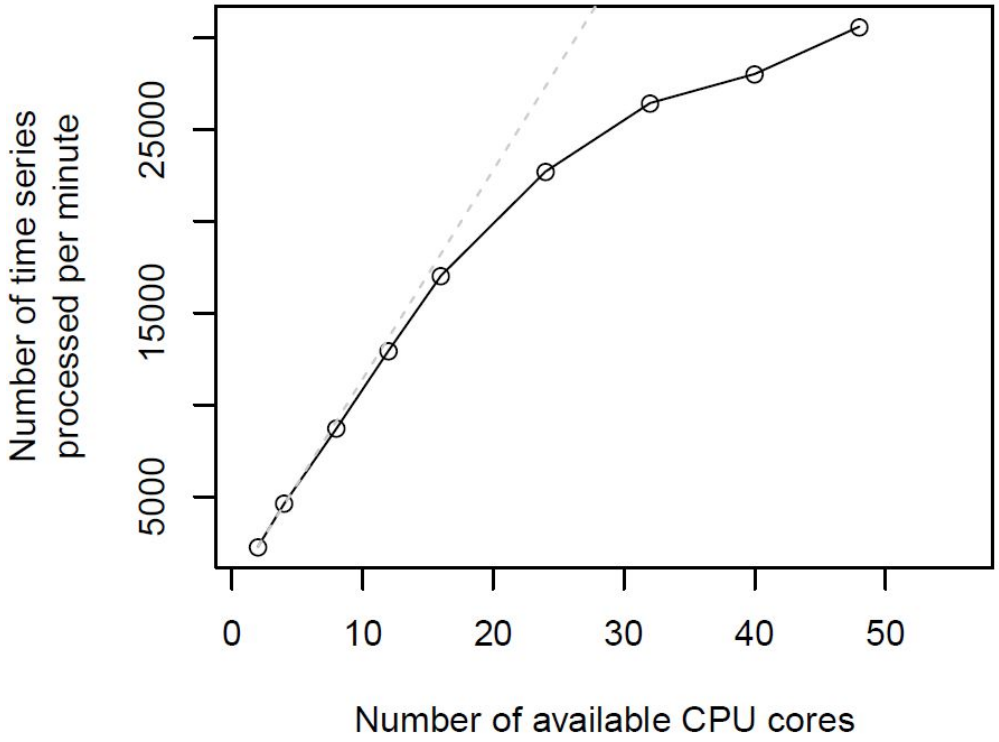
The data is represented in SciDB as a three-dimensional array with **daily temporal resolution** and

- 49548 x 47713 x 4177 cells in total
- 64, 64, 4177 cells per chunk
- Only 0.5% ($54 \cdot 10^9$) of the cells contain data
- SciDB has sparse storage

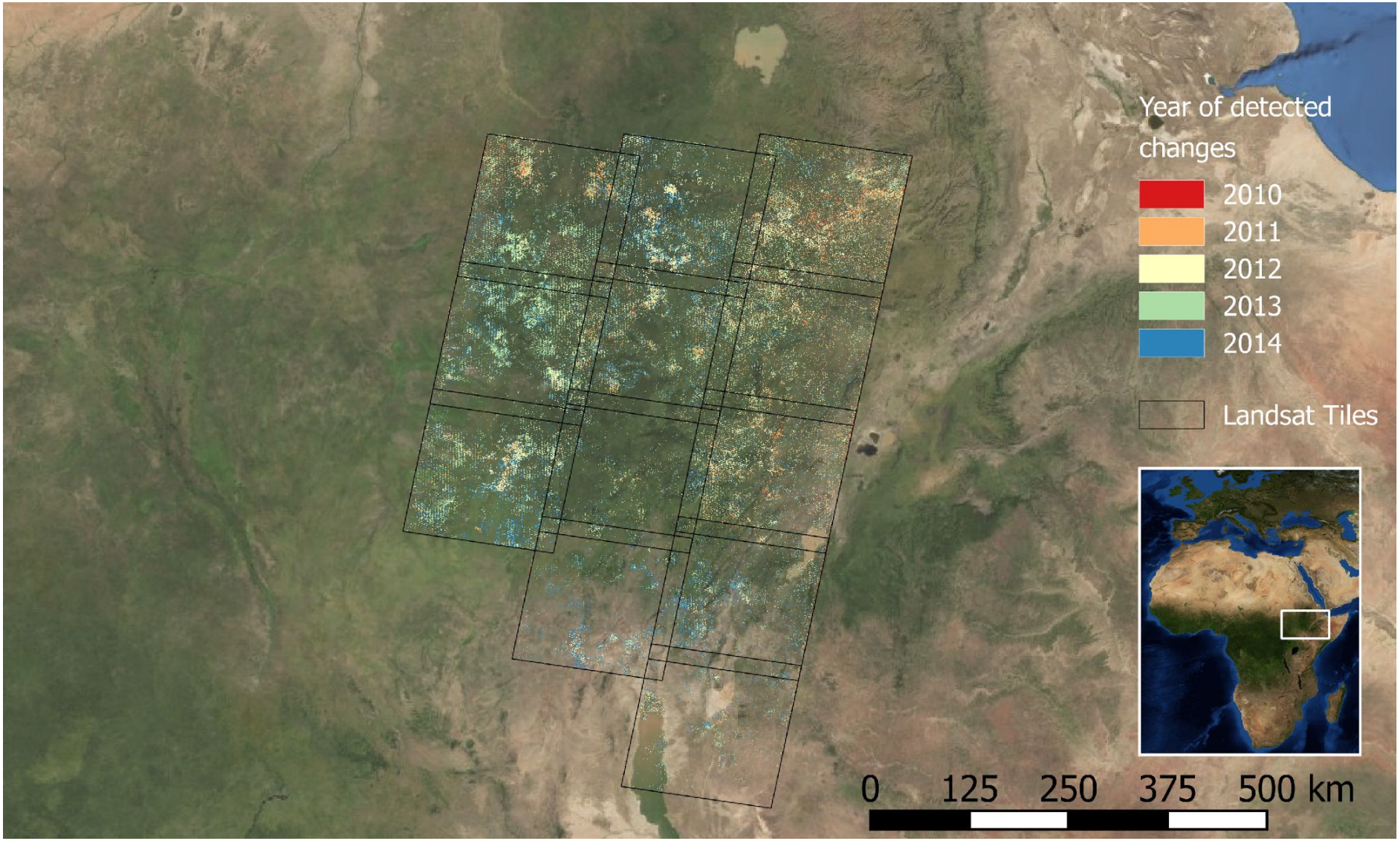


Scalability with SciDB instances

- 16 SciDB instances on one machine used (64 CPU cores, 256 GB main memory)
- running bfast monitor repeatedly with different number of available CPU cores on a small subset



Study case: results



Conclusions

- SciDB is able to scale user defined R / python code
- The multidimensional array model, chunking, and sparse storage are well-suited to represent large EO datasets from many scenes
- Ingestion and data restructuring time consuming
- Installation and data ingestion not straightforward
- Analysis from R / Python easy to learn for R / Python users

Short course overview

Part I: Analysing geoscientific arrays: Overview of data and software

Part II: Hands-on: array-based spatiotemporal change detection with R

Part III: Scalable data management and analysis with SciDB

Part IV: Demonstration: empirical orthogonal function analysis with SciDB

Demonstration: empirical orthogonal function analysis with SciDB

1. Description of the Analysis for 3.
2. An introduction how to work with spacetime arrays with R and SciDB (live demonstration)
3. Example code: Scalable EOF analysis with SciDB

- We use R as a SciDB client, implementation would look similar in
  - Python
  - AFL (array functional language)

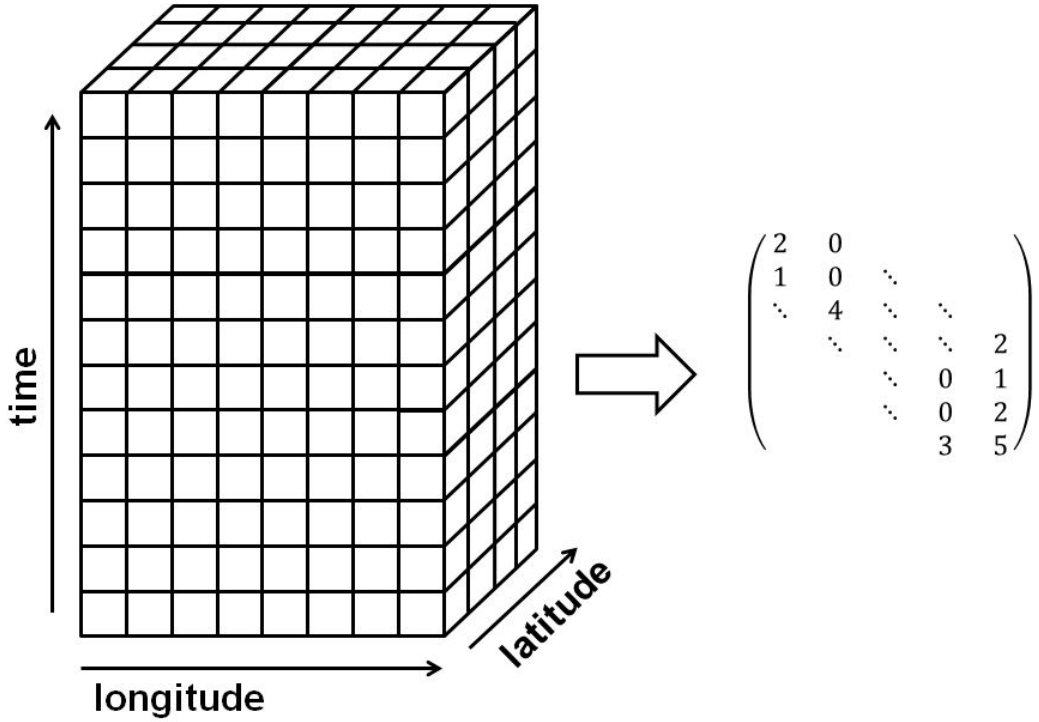
Data and Method

- Tropical Rainfall Measuring Mission (TRMM) daily accumulated precipitation observations from 1998 to 2015
  - What are typical patterns in the data?
  - How can we reduce dimensionality?

- Standard Method: Empirical Orthogonal Function analysis (EOF)

Computation of EOFs

1. Remove images with missing values
2. Reshape the three-dimensional data cube to a two-dimensional data matrix



Computation of EOFs

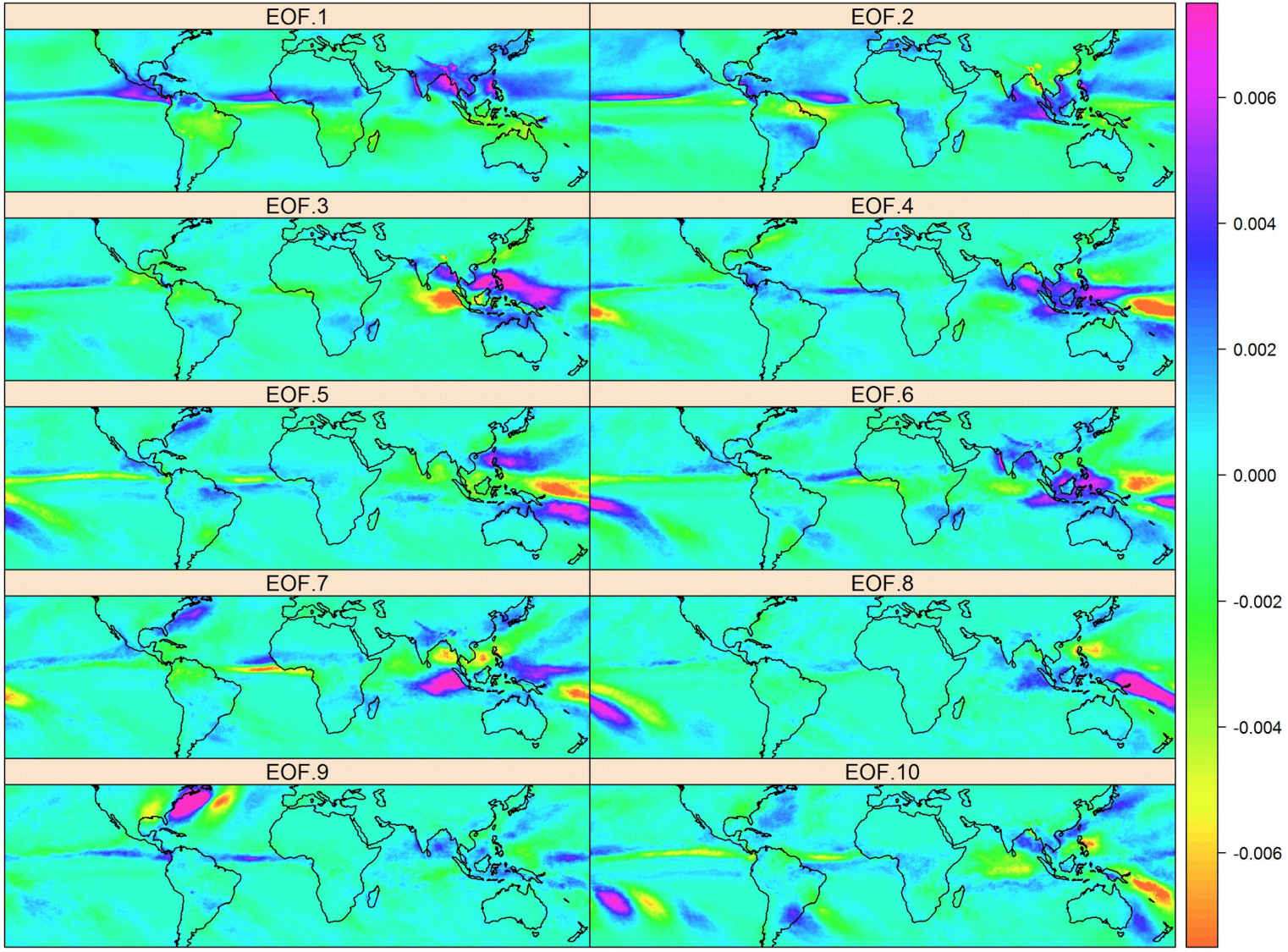
3. Detrend columns of the data matrix

4. Perform a singular value decomposition (SVD)

$$X = U\Sigma V^T$$

5. Extract the first columns (EOFs) of V^T and reshape as images

EOF results


EOF.1
EOF.2
EOF.3
EOF.4
EOF.5
EOF.6
EOF.7
EOF.8
EOF.9
EOF.10
0.006
0.004
0.002
0.000
-0.002
-0.004
-0.006