(Apache) Spark for physicists

C. Arnault, G. Barrand, J.E. Campagne, J. Peloton and S. Plaszczynski

LAL, Univ. Paris-Sud, CNRS/IN2P3, Université Paris-Saclay, Orsay, France

November 7, 2018

The rise of big data computing

- 2004 `Google`: `mapReduce` programming model foundation of *distributed computing*
- 2006 `Hadoop` open-source framework (ecosystem) `HDFS`,`Hive`, `YARN`...
- 2004 `scala` (`java` ecosystem)
- 2009 `Spark`: research project at UC. Berkeley
- 2015 `Spark` SQL (dataframes)
- today: (`Apache`) `Spark` used by $\gtrsim$ 1000 companies

So what is Spark about?

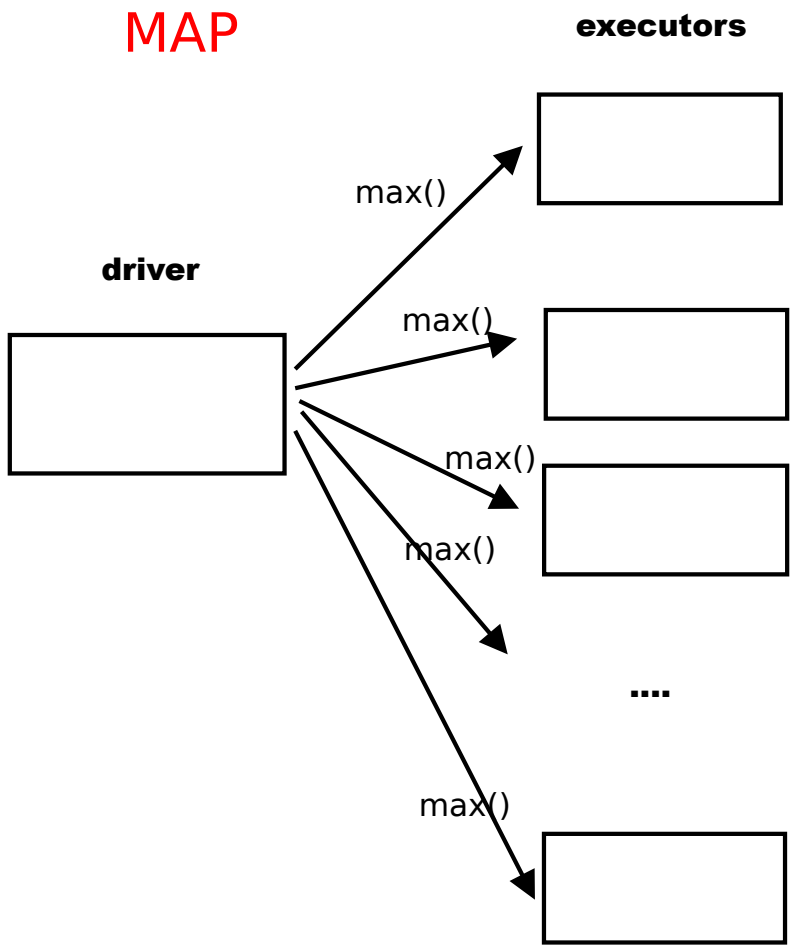
For a **large** volume of data it is more efficient
to **move the computation to the data** than the other way.

Spark = a framework to do it efficiently on distributed architecture
→scala, (java), python, R

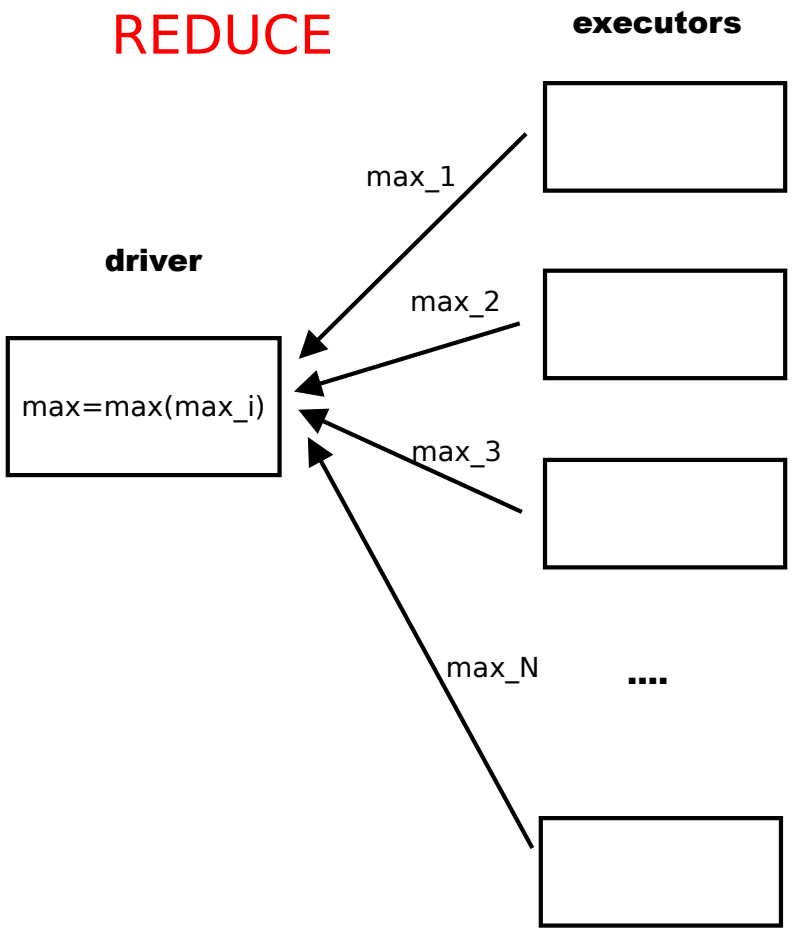
```
dataframe.transform1().transform2()....action()
```

This is Functional Programming (but you don't need to know it!)

Distributed computing



Advantage 1= coarse-grain parallelization



Advantage 2 ?

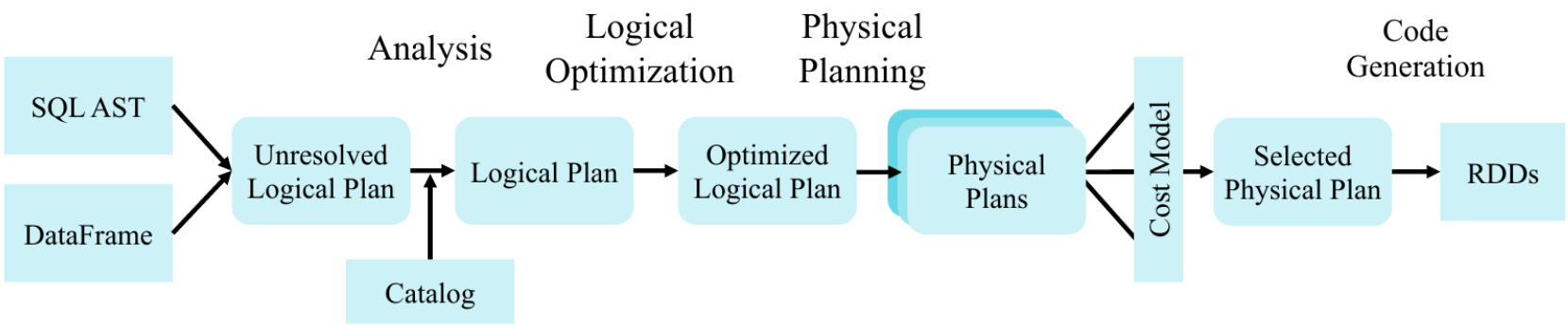
```
> df=spark.read.format("fits")\
            .load("path/to/110GB/of/fits/files")
> df.show(5)
+---------+---------+---------+---------+
|       RA|      Dec|        z|     zrec|
+---------+---------+---------+---------+
|225.80168|18.519966|2.4199903| 2.414322|
|225.73839|18.588171|2.4056022|2.2913096|
|225.79999|18.635067| 2.396816|2.3597262|
|225.49783|18.570776|2.4139786|2.3434482|
|225.57983|18.638515|2.3995044|2.3826954|
+---------+---------+---------+---------+
```

5s !/?

Lazy evaluation →optimization

- you are used to **imperative** languages (C/C++/FORTRAN...)
- here **lazy evaluation**: code is an 'expression-language' that allows to build a Direct Acyclic Graph (DAG)
- transformations (`load,map,filter..`) →update DAG
- actions (`count,collect,show..`) →optimize DAG (`Catalyst`) and run

Advantage 2= Automatic pipeline optimization



the Machine does it better than you! →Spark reason of success

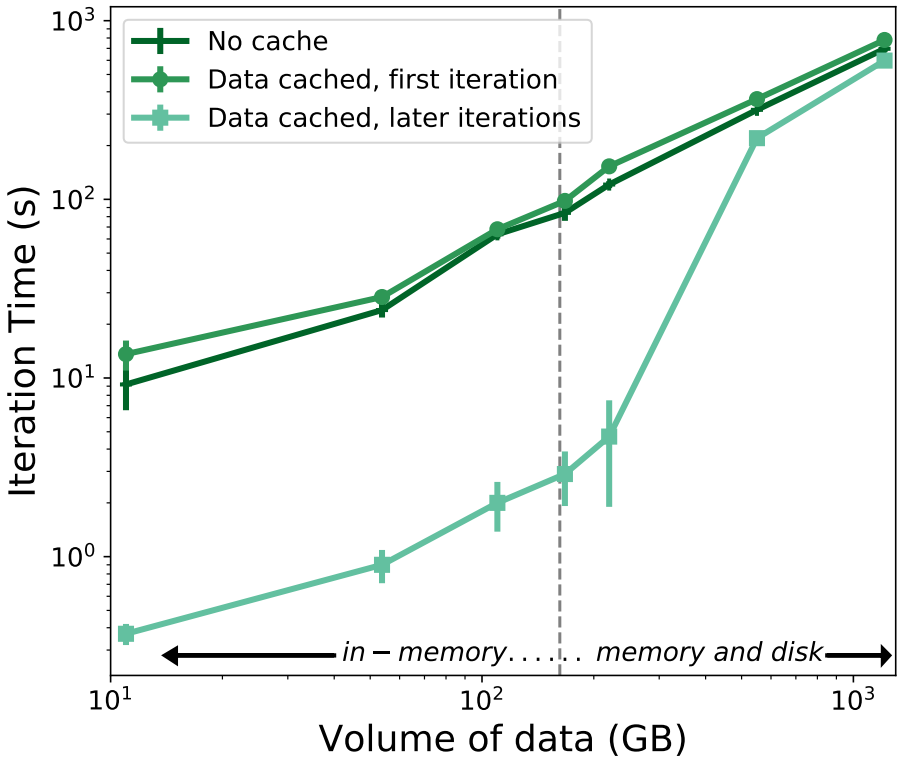
Advantage 3= in memory work (cache)

Put the data in cache as if you had a **huge** RAM

- ex: 110 GB on a small cluster (8 workers)
- 1TB at NERSC (100GB/node, 60% for cache)

Then you can work interactively

Advantage 4= scaling



`spark-fits` high performance connector:

(Peloton,Arnault,Plaszczynski ArXiv:1804.07501)

A use-case in cosmology

- generate **LSST 10Y** of galaxies with fast sim `https://github.com/damonge/CoLoRe.git`
- → **110 GB** of FITS files. 6. 10^9 galaxies
- UPSUD cluster (9 machines)

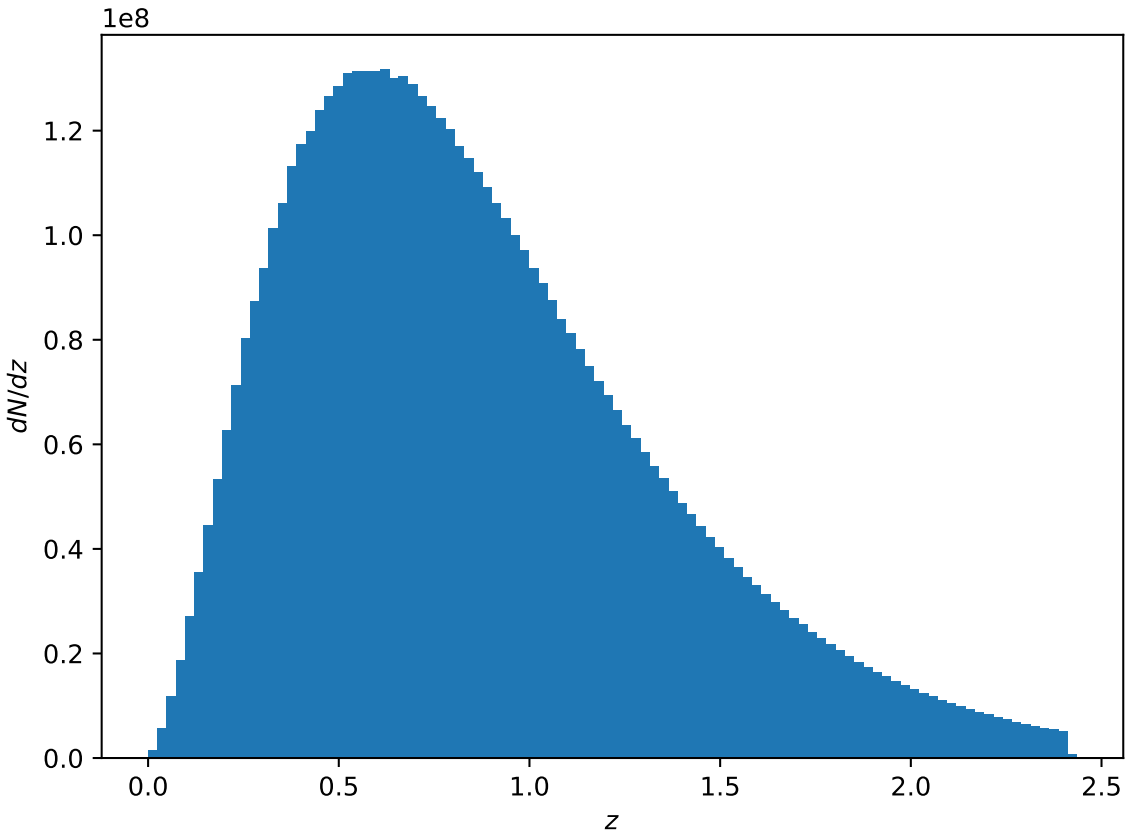
- goal is to have a quick *interactive* look at what was generated (`python`)
- this is different from *developing* software (`scala`)

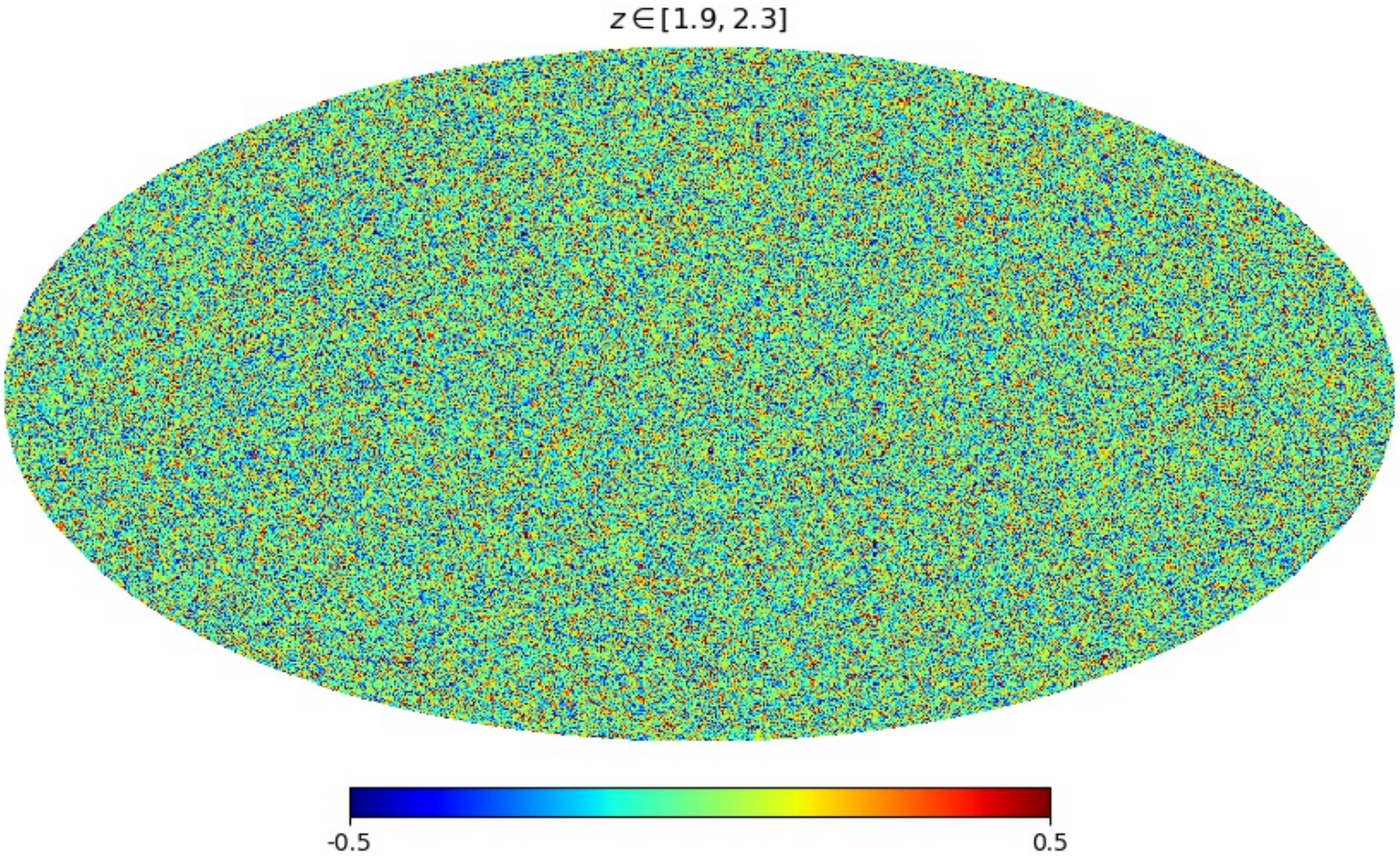
performances (python)

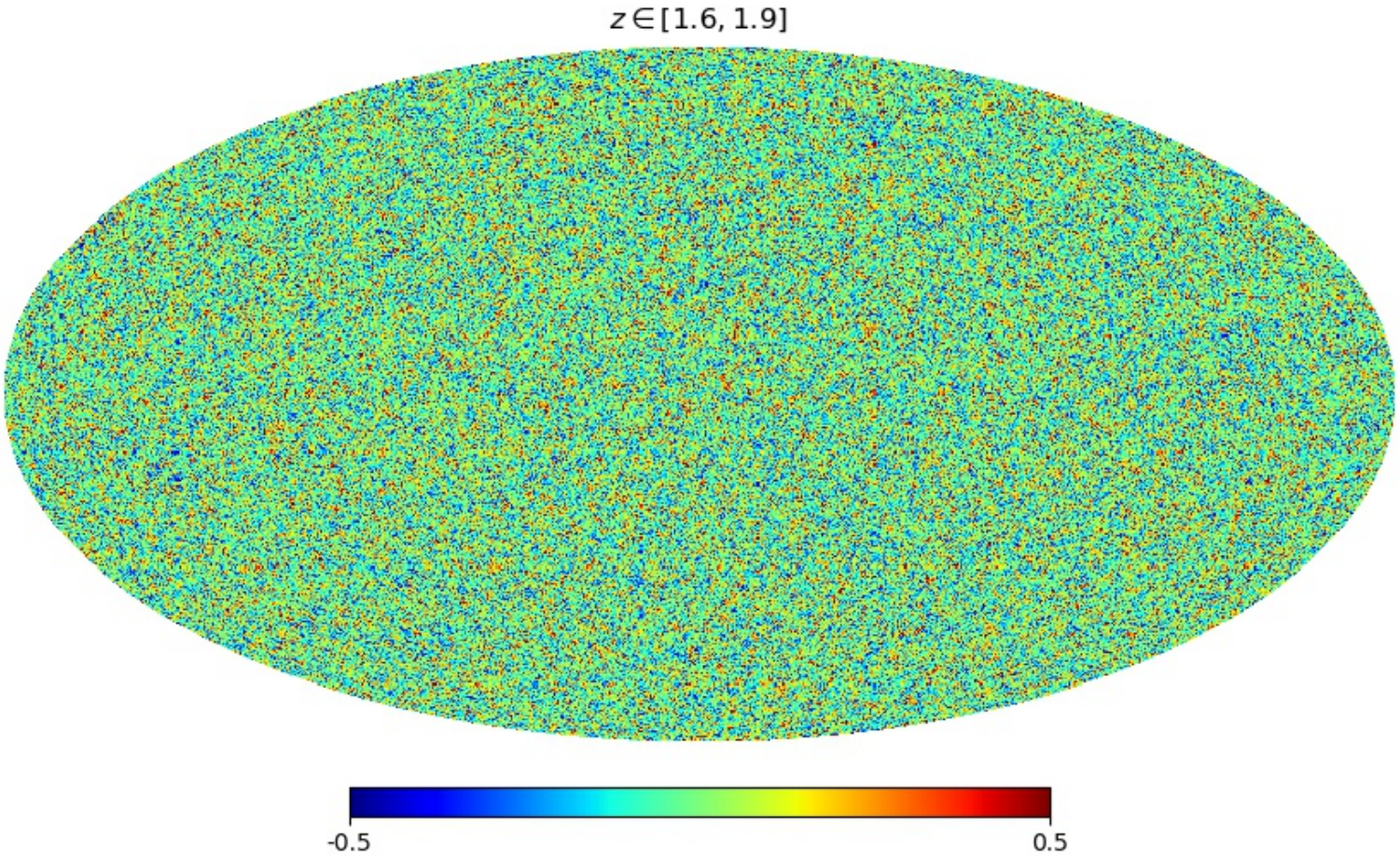
LSST10Y, 6 10^9 galaxies

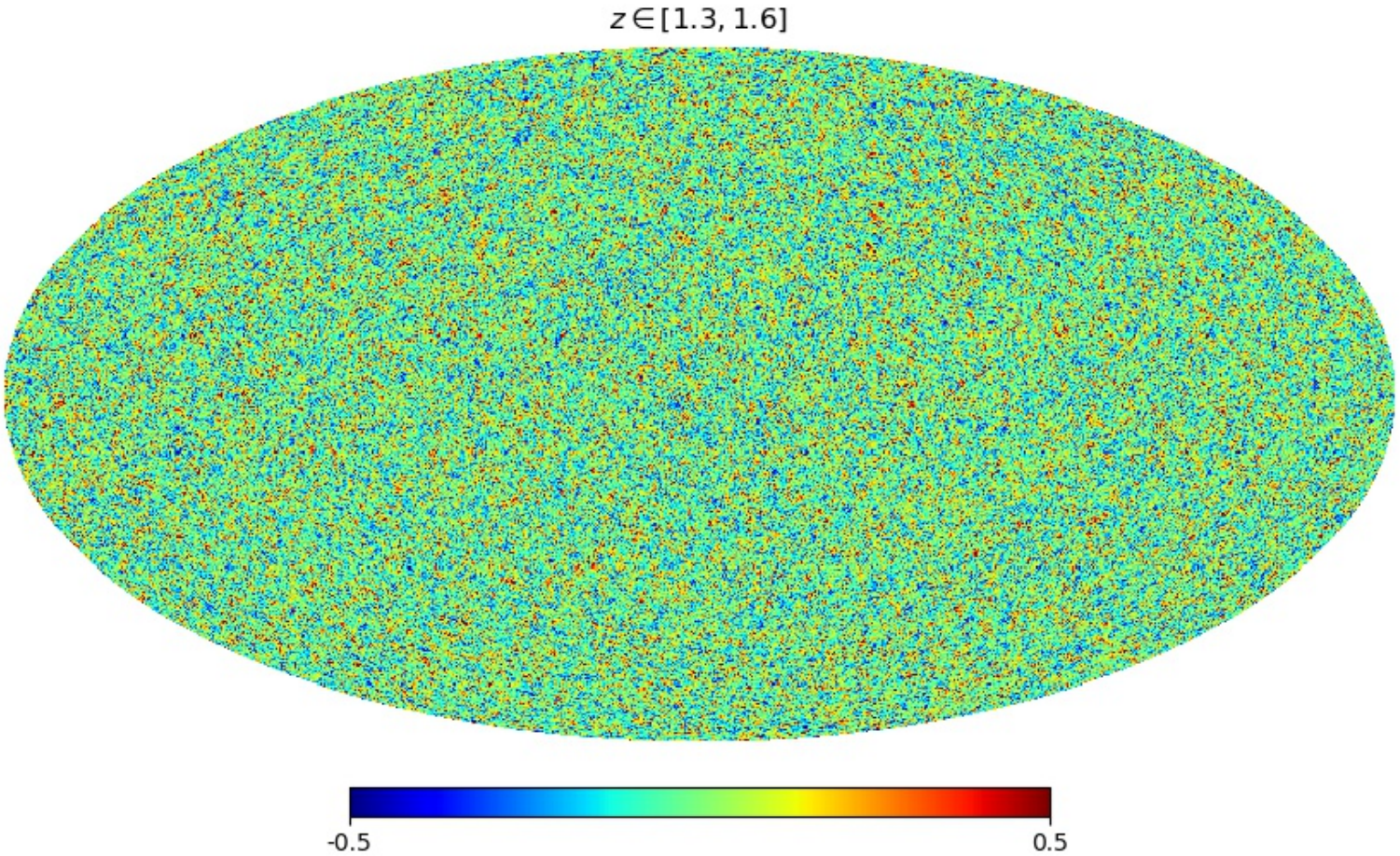
- read+cache : 110 s
- simple statistics on one column: 4 s
- histogram : 11 s (imperative sequential way : 45 mins)
- tomography: 30s/shell

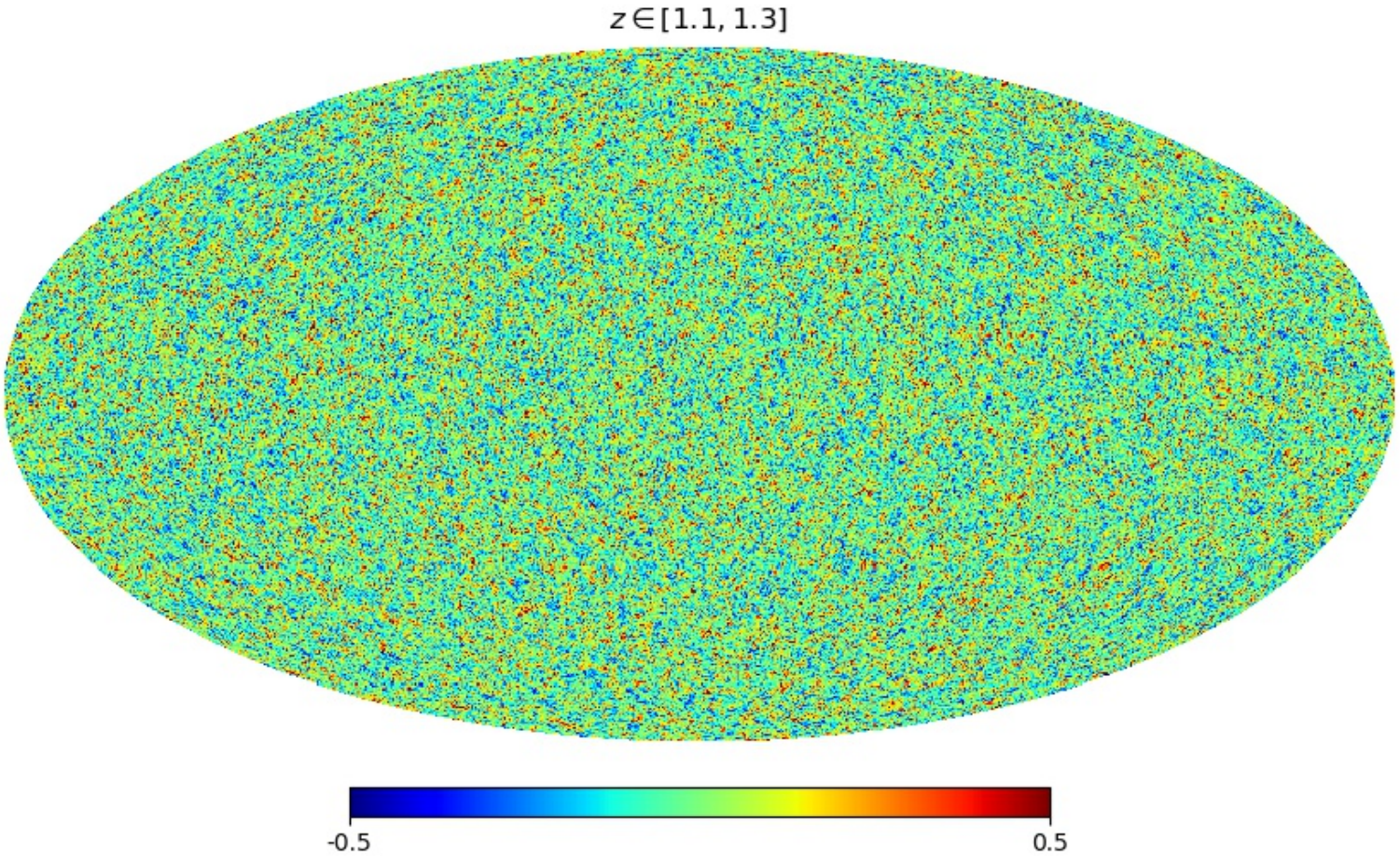
Selection function

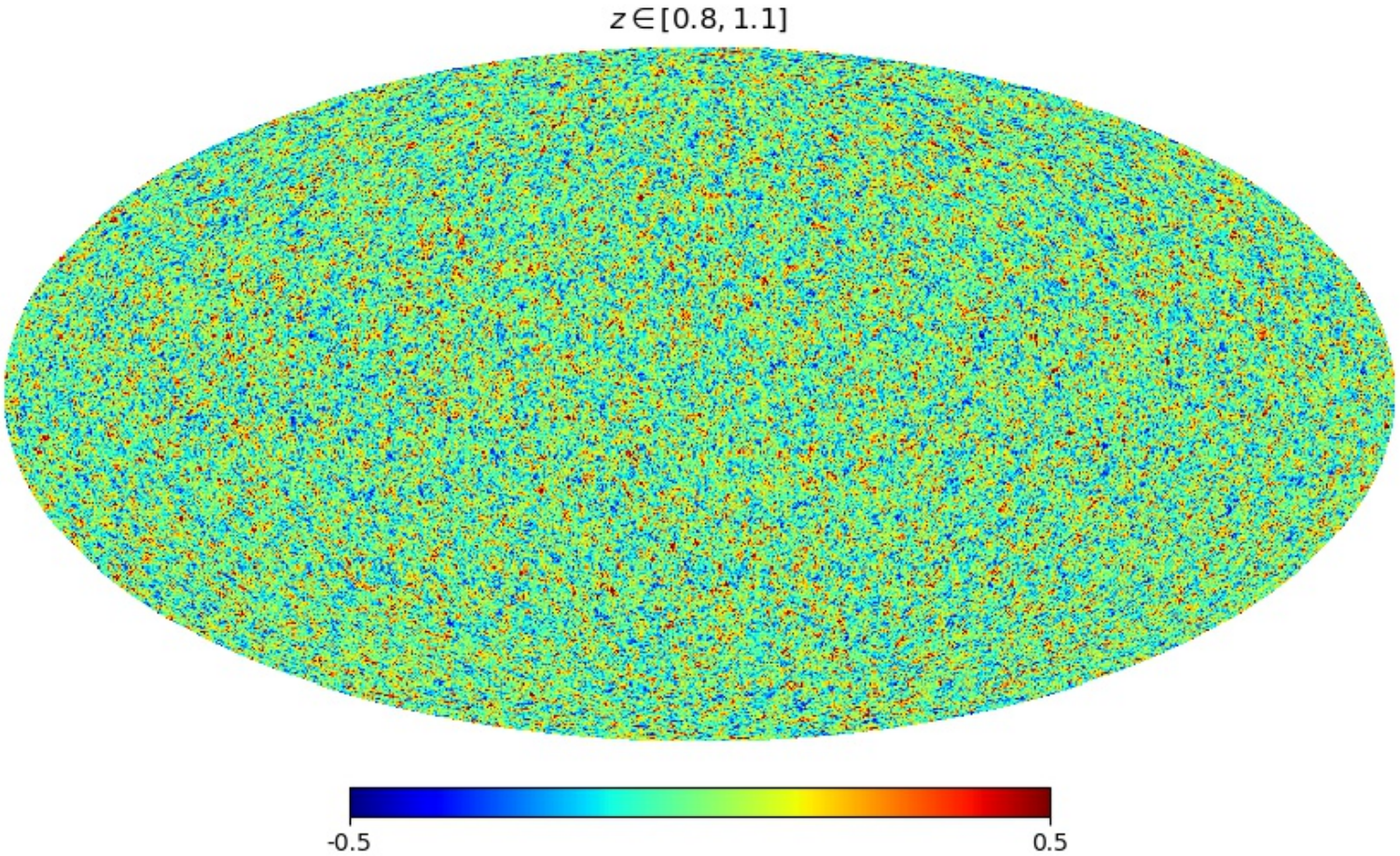


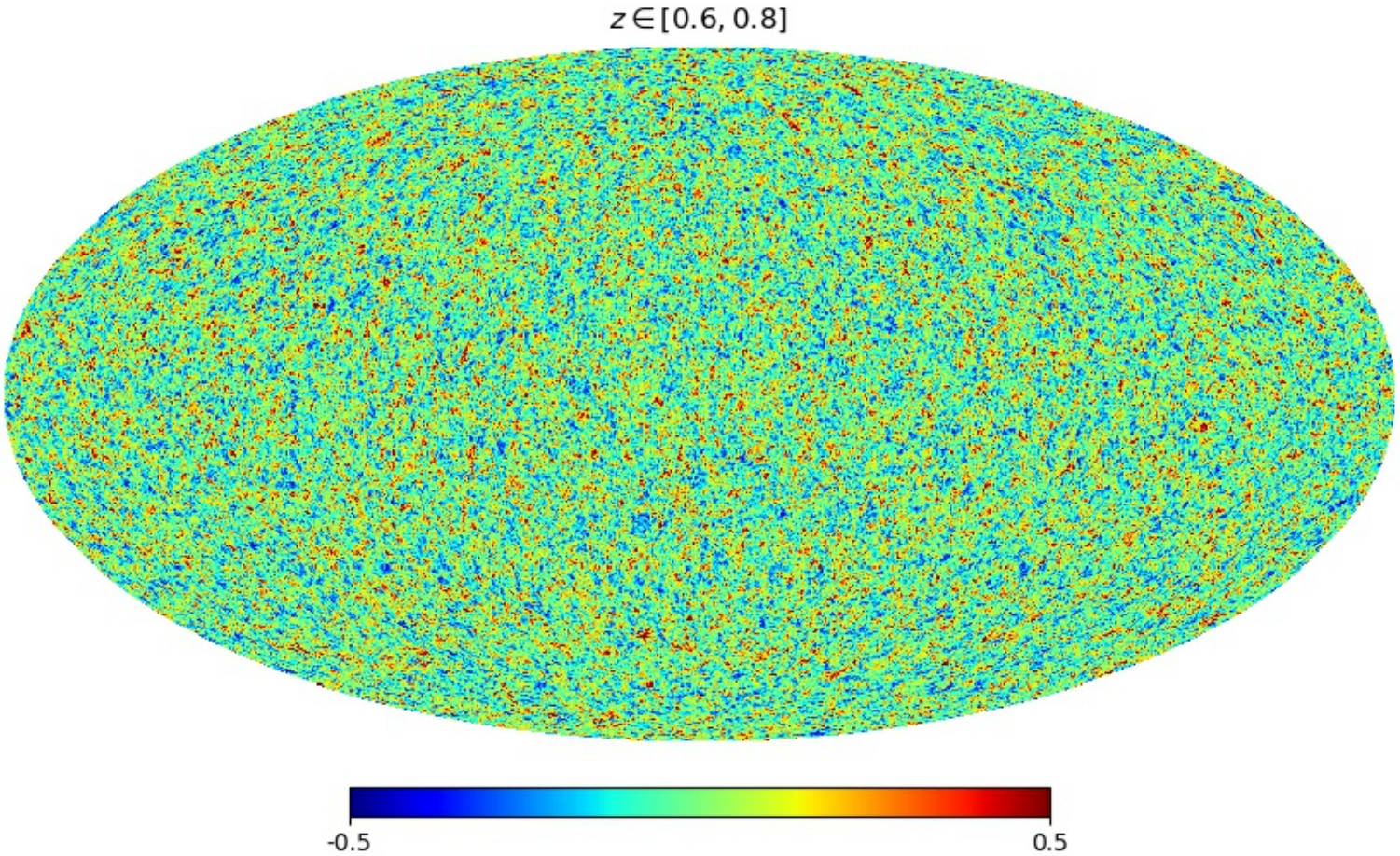

$z \in [1.9, 2.3]$
-0.5
0.5

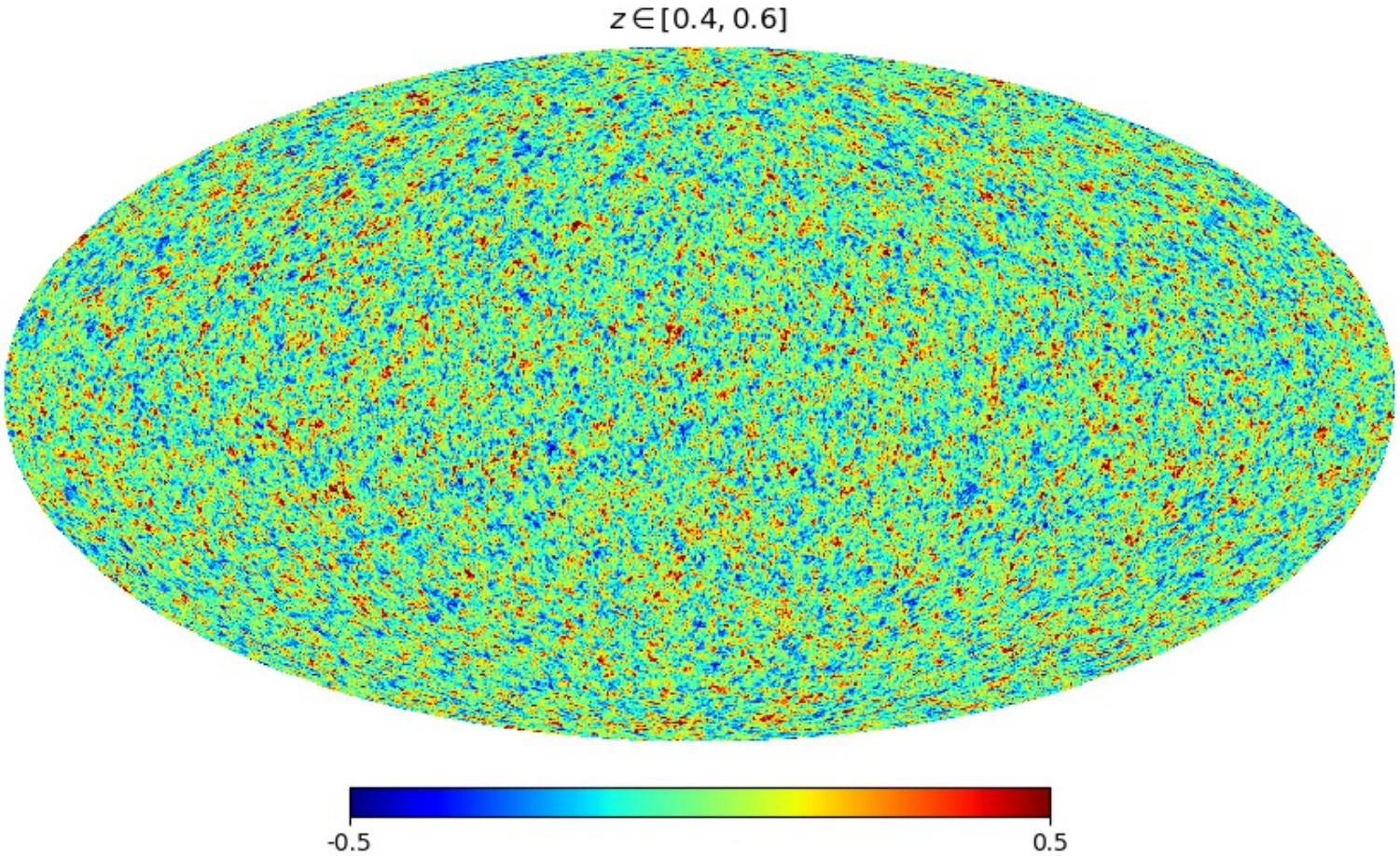

$z \in [1.6, 1.9]$
-0.5
0.5

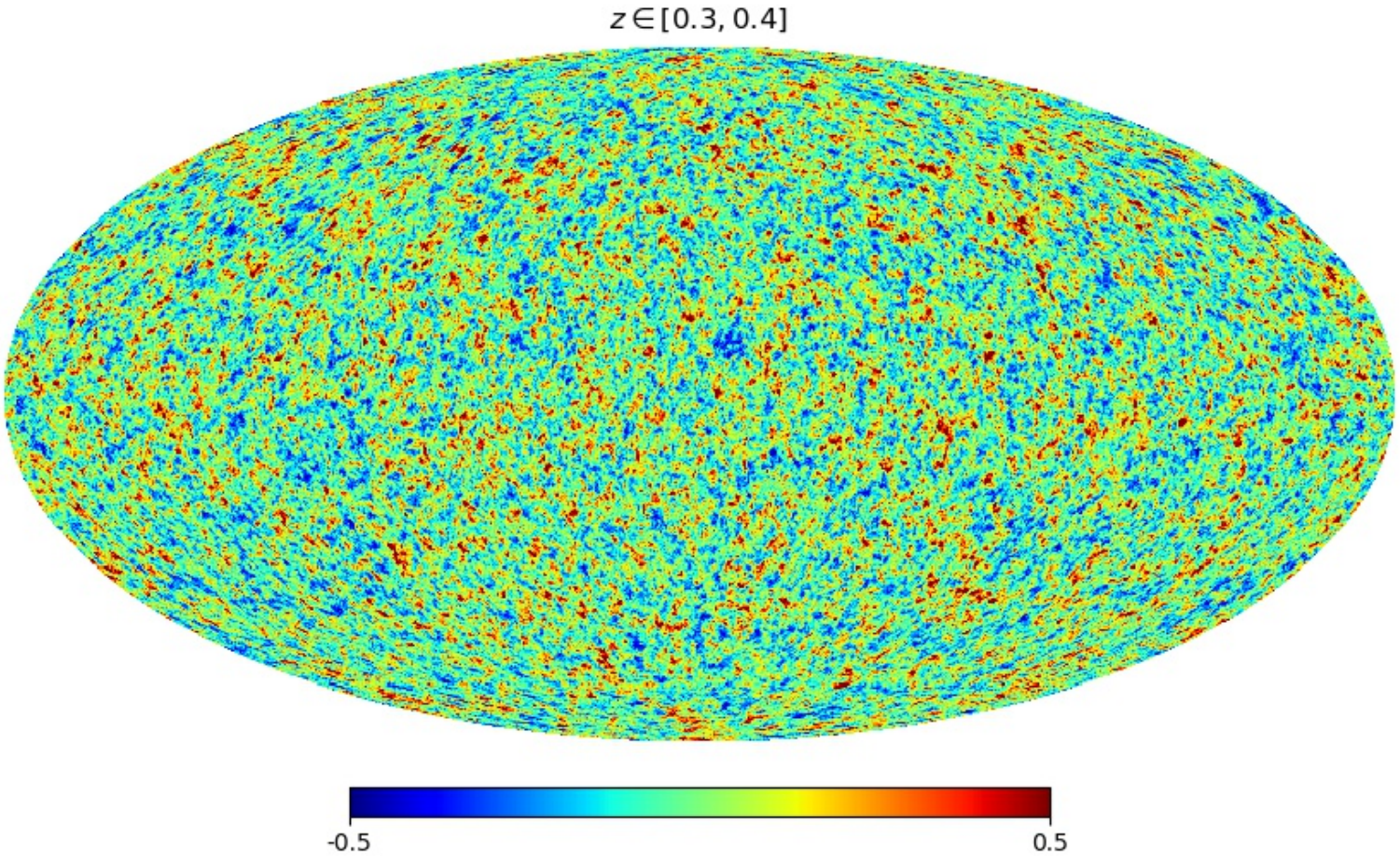

$z \in [1.3, 1.6]$
-0.5
0.5

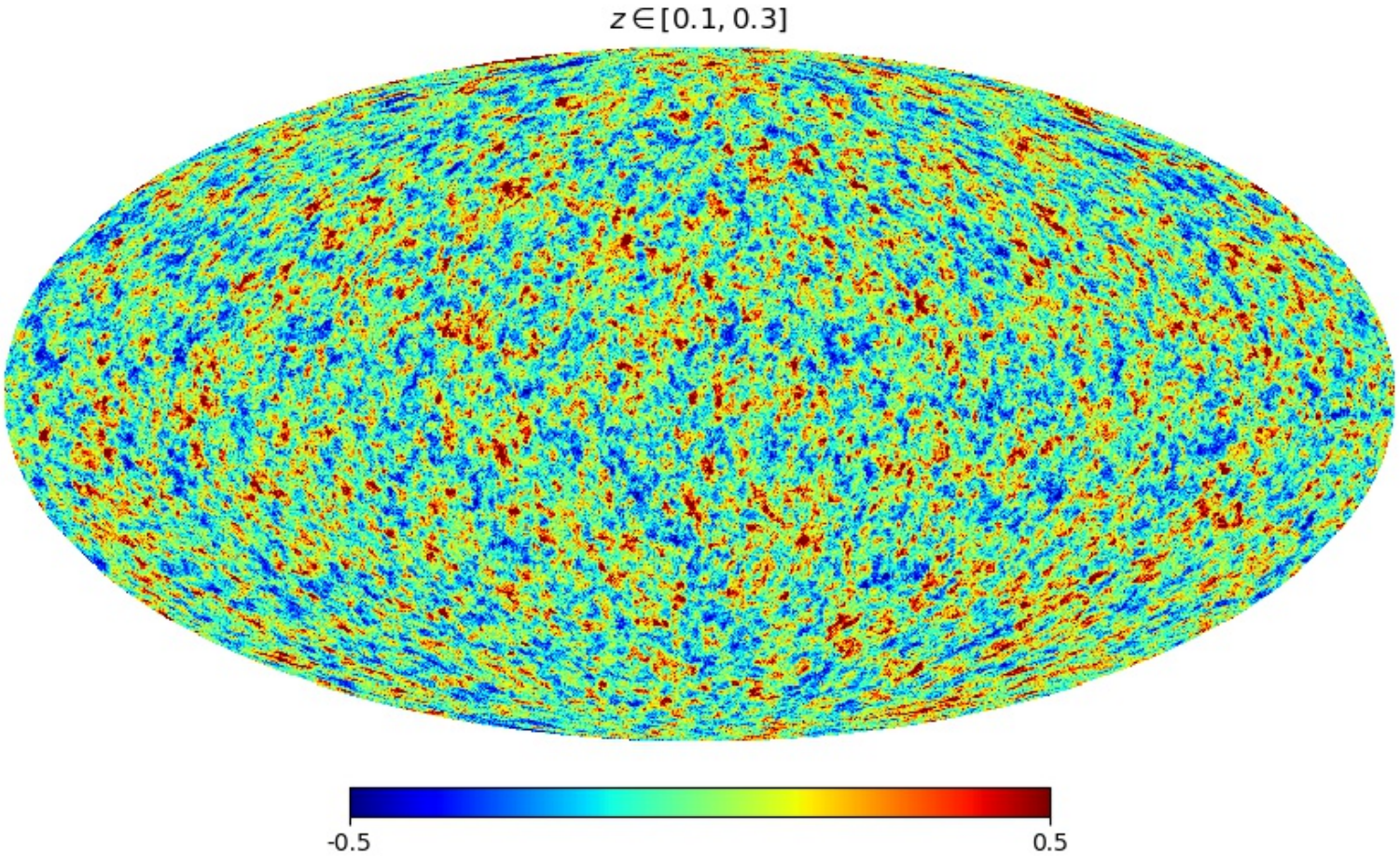

$z \in [1.1, 1.3]$
-0.5
0.5

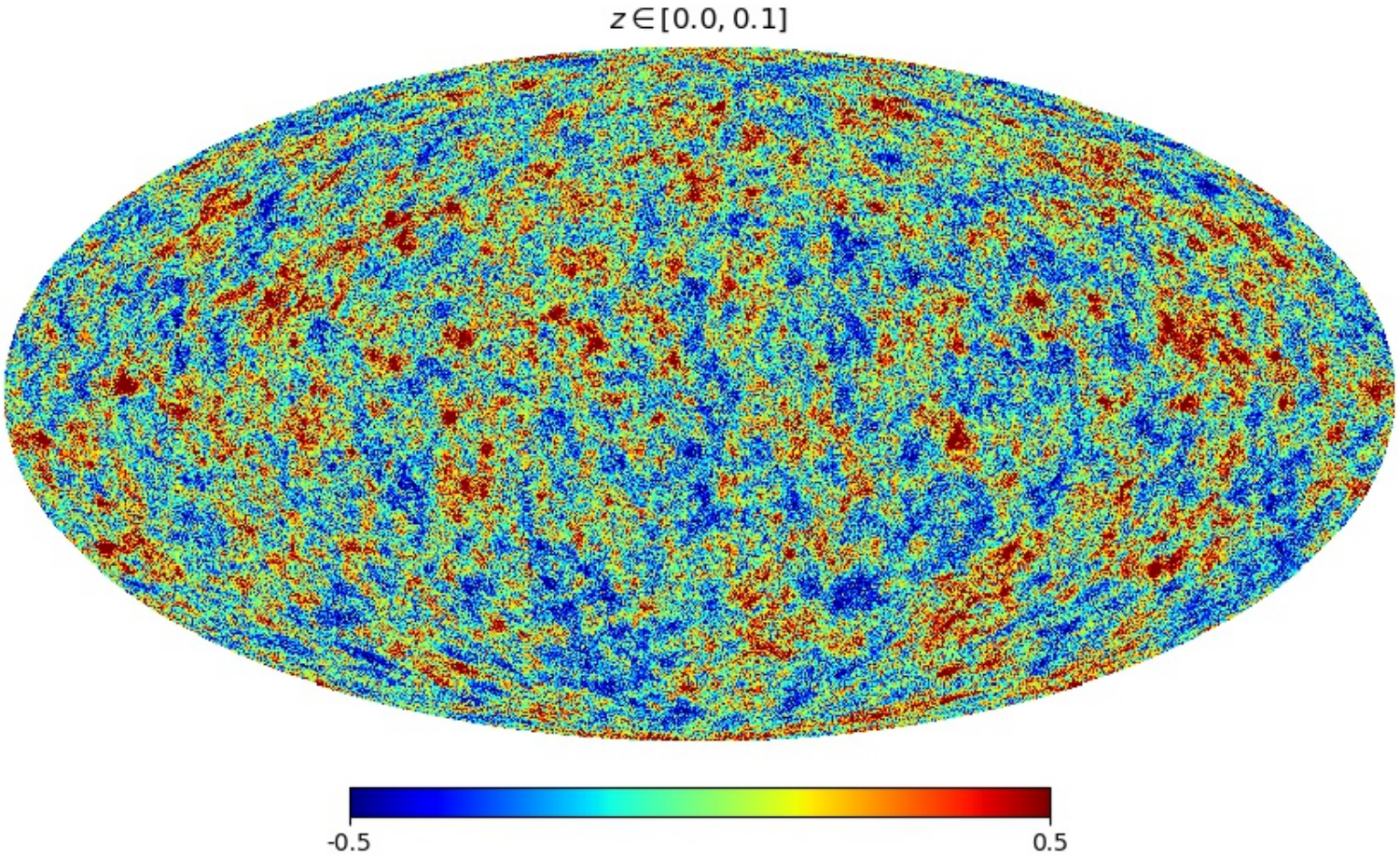

$z \in [0.8, 1.1]$
-0.5
0.5


$z \in [0.6, 0.8]$
-0.5
0.5

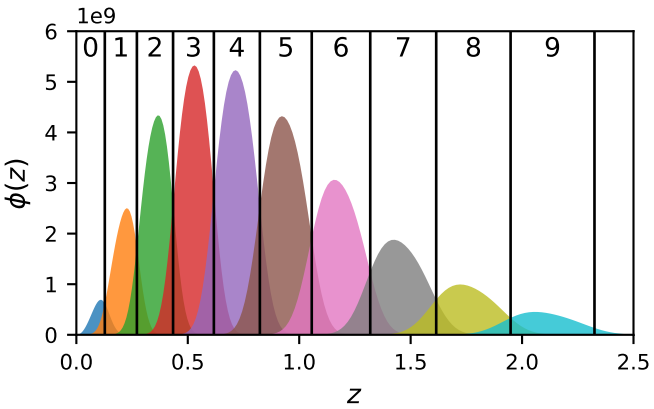

$z \in [0.4, 0.6]$
-0.5
0.5

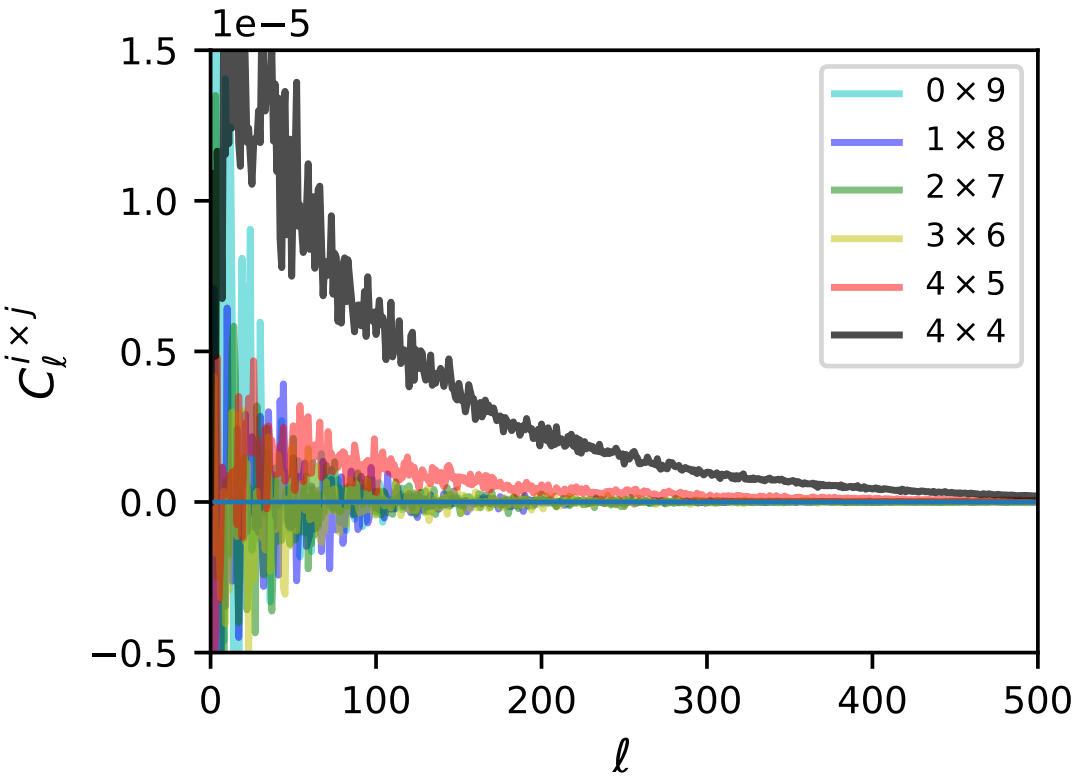

$z \in [0.3, 0.4]$
-0.5
0.5


$z \in [0.1, 0.3]$
-0.5
0.5


$z \in [0.0, 0.1]$
-0.5
0.5

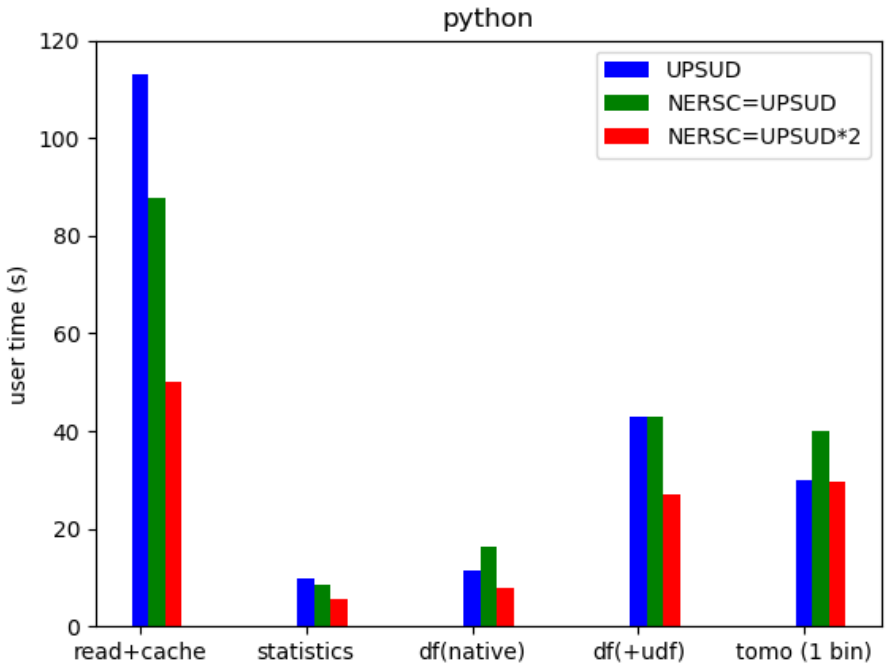
Power spectra





Spark @NERSC

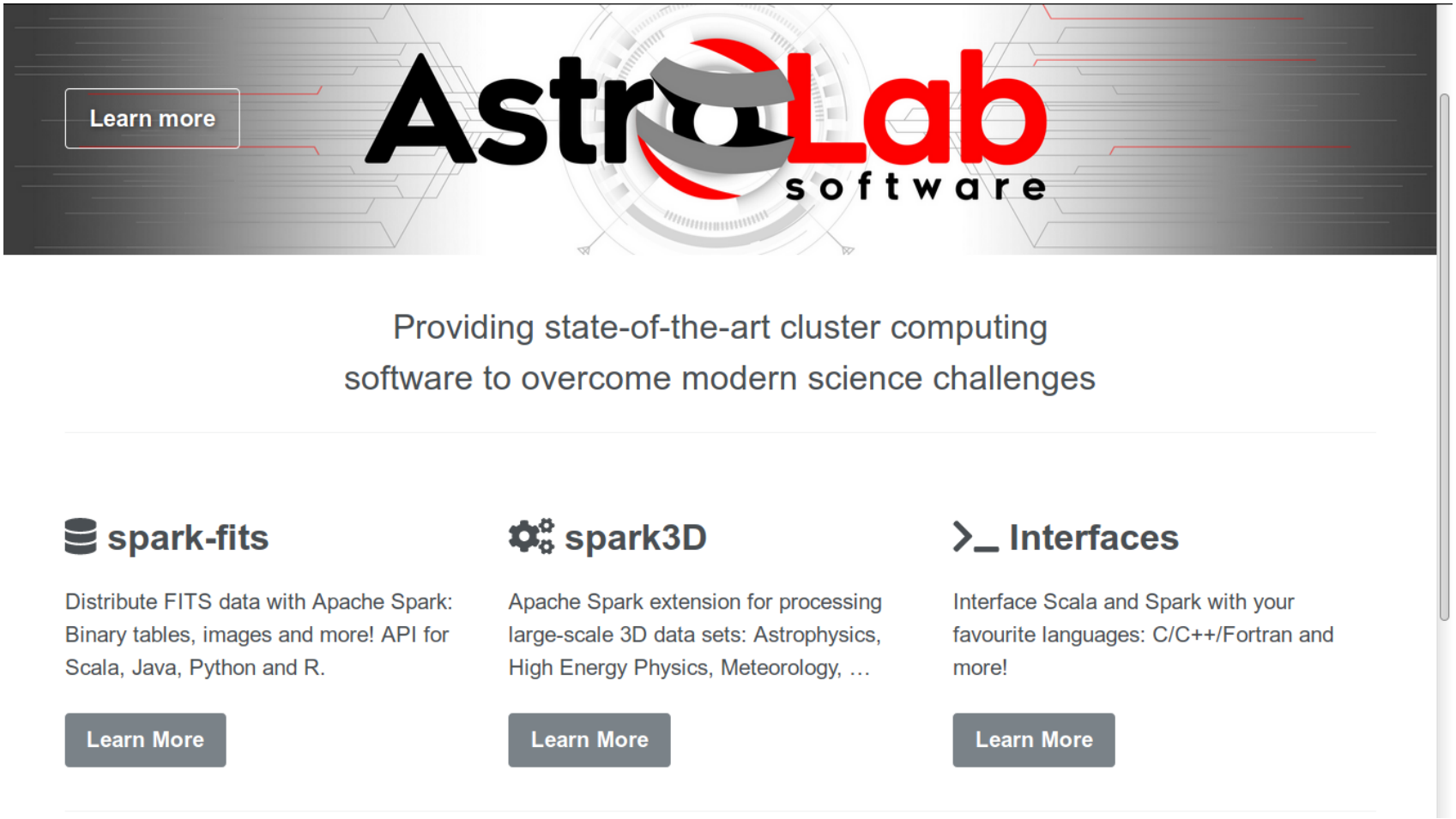
- shifter image + interactive queue (or jupyterlab)
- although on `Lustre` excellent performances



I want to test that immediately!

1. Plaszczynski,Peloton,Arnault,Campagne ArXiv:1807.03078
2. notebooks in LSSTDESC/DC2-production + DC2-analysis
3. LSSTDESC/desc-spark
   - how to run @NERSC (jupyter/interactive/batch)
   - logbook
   - tutorials + bootcamp under construction

https://astrolabsoftware.github.io



Applications

- DC2-production + DC2-analysis (#desc-dc2-data-access+ #3x2pt)
- images→catalog : distributed *SExtractor*
- clustering in Nbody sims (coll with UZH)
- client/server data visualization
- 2(3)-pt computation
- PhD proposed on `Spark` + HPC combination