Monte Carlo Simulations

- What are Monte Carlo Simulations and why ones them?
- Pseudo Random Number generators
- Creating a realization of a general PDF
- The Bootstrap approach
- A “real life” example: LOFAR simulations

Random Number generators

“Computer generated random numbers” is conceptually a contradictory notion. Computer generate random numbers based on a fixed recipe that is set by the programmer, how can a fixed formula generate an infinitely large set of completely random numbers? Obviously, we'll not solve this complicated issue here, however, from this opening you can probably realize that there are no truly random number generators produced by computer (example will follow momentarily) their proper name is actually “pseudo-Random number generators”.

Putting the philosophical issues aside, this issue has practical side, one has to be extra careful with random number generators. In the history of computer analysis of data there are many examples of very badly written random number generators that led to a vast many wrong conclusions, the most infamous of such routines is the one called RANDU that was widespread on IBM mainframe computers.

Despite all what have been said, random number generators, provided they are well tested, constitute one of the main tools scientists have in their disposal to analyse and model data.

A typical random number generator is a function or subroutine RAN(seed) that requires the user to provide an "initial random number" called the seed from which the routine generates a number, the next input seed is automatically generated by the previous step.

Almost all supplied random number generators fall under the name congruential generators (Lehmer 1948), which create a sequence of integers from the following simple recipe

$$I_{j+1} = aI_j + c \; (mod\, m)$$

Pros: This algorithm is fast in generation of numbers and requires very few operations per call.

Cons: Not free of sequential correlation on successive calls.

Routine RANDU (IBM Corp.):

"We guarantee that each number is random individually,
but we don't guarantee that more than one of them is random."

The Transformation method

- We know that if y=y(x), then: $$p(y) = p(x)\left|\frac{dx}{dy}\right|$$

- We know how to generate a uniform random number, so that the probability of it being between x and x+dx is:

$$p(x)dx = \begin{cases} dx & \text{if } 0 \leq x \leq 1; \\ 0 & \text{otherwise.} \end{cases}$$

- Therefore we need to solve the differential equation:

$$\frac{dx}{dy} = f(y) \ \ (\equiv p(y))$$

- The solution is: $y(x) = F^{-1}(x)$ where $F = \int f(y)dy$

- This method is used to generate normal, exponential and other types of distributions where the inverse is well defined and easy to obtain.

The Transformation method: exponential deviates

As an simple example consider the case of an exponential distribution

$$p(y)dy = e^{-y}dy$$

From the previous relation we can produce a realization of this distribution from a uniform deviate x through the transformation:

$$y(x) = F^{-1}(x) = -\ln x$$

The Transformation method: Gaussian deviates

An important example for the application of the transformation method is the Box-Müller method for generating random gaussian deviates with a Gaussian (normal) distribution.

$$p(y)dy = \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy$$

This method makes use of the fact that it is possible to find a function that generates the 2-dimensional Gaussian distribution

$$p(y_1, y_2)dy_1 dy_2 = \frac{1}{2\pi} \exp\left(-\frac{(y_1^2 + y_2^2)}{2}\right) dy_1 dy_2$$

from two uniform deviates x_1 and x_2

$$y_1 = \sqrt{-2\ln x_1} \cos 2\pi x_2$$
$$y_2 = \sqrt{-2\ln x_1} \sin 2\pi x_2$$

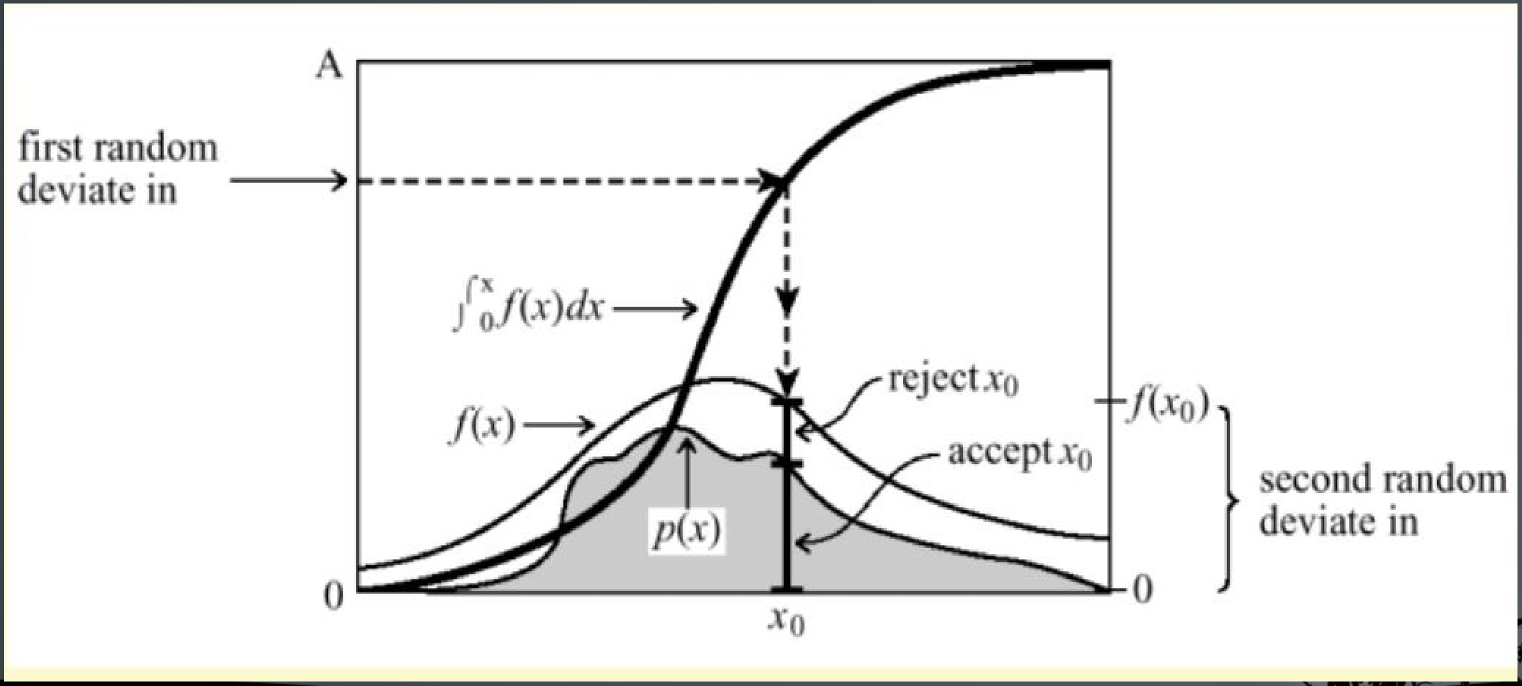
Acceptance rejection method

– Generate a random deviate of f(x) (more tractable function).

– Generate a second deviate to decide whether to accept or reject that x.

– Ratio of accepted to rejected points is the ratio of the area under p to the area between p and f.

- This is very useful for generating: Gamma distribution deviates, Poisson deviates and binomial deviates.



Bootstrap

- The bootstrap is a name generically applied to statistical resampling schemes that allow uncertainty in the data to be assessed from the data themselves, in other words, "pulling yourself up by your bootstraps".
- Given n observations z_i, i=1,....,n and a calculated statistic S, e.g., the mean , what is the uncertainty in S?
- The procedure:
  - Draw n values z'_i from the original data with replacement
  - Calculate the statistic S' from the "bootstrapped" sample
  - Repeat L times to build up a distribution of uncertainty in S

Bootstrap Assumptions

1. Your sample is a valid representative of the population.

2. Bootstrap method will take sampling with replacement from the sample. Each sub sampling is independent and identical distribution (i.i.d.). In other word, it assumes that the sub samples come from the same distribution of the population, but each sample is drawn independently from the other samples.

Bootstrap: Applications

Here are some typical statistical examples of problems that you can use Bootstrap method to solve

1. Suppose you have some sample data but your sample is quite small that you are not sure the population theoretical distribution of your sample. How could you estimate the variance of the mean average of your sample?

2. You have two samples from unknown distribution, name them X and Y. You want to know the distribution of ratio Z = X/Y and want to derive some useful statistics (such as mean and standard deviation) from the distribution of the ratio.

3. You have two samples A and B and you want to test whether they come from the same population

4. You have regression model $y = \alpha x + \beta$ and you want to get the confidence interval of the parameters α and β.

Jackknife analysis

For a given statistic, one often wants to calculate the bias and error with both are as small as possible. One practical way of doing so, in the absent of knowledge of underlying distribution, is from the data itself through the so called Jackknife analysis. The basic idea is calculate the statistic repeatedly while each excluding one (or more) data points from the estimation of the statistic.

Jackknife analysis is related, albeit less general, to the bootstrap method discussed earlier. Its main advantage, however, is in its simplicity.

Jackknife analysis

Here we'll rigorously proof that the Jackknife analysis works for a certain case. Assume we are after a statistic *s* which we want to estimate from n data points, $E(s_n)$. We'll assume that the estimator is biased although asymptotically unbiased. For example, assume that the bias in the estimation is given by:

$$E(s_n) - s = \sum_{i=1}^{\infty} a_i/n^i$$

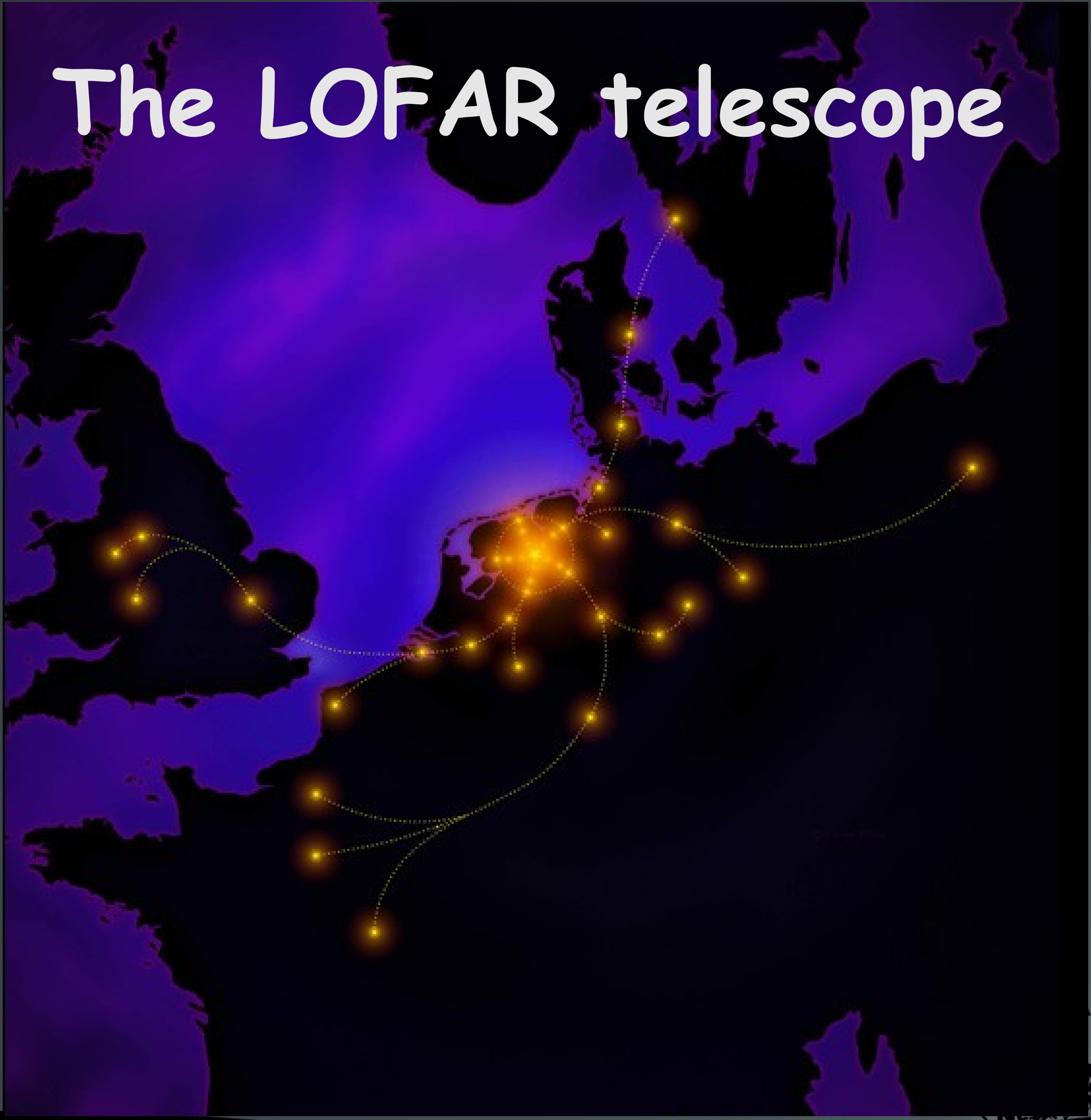
We can make n samples of n-1 observations, define a new statistic:

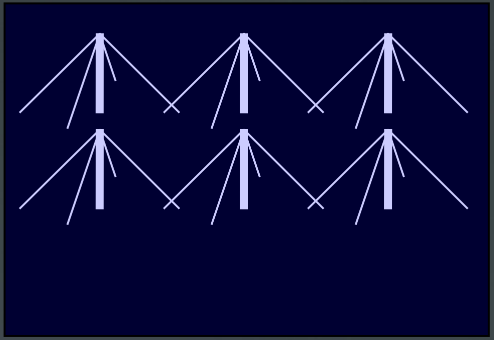
$$s'_n = ns_n - (n-1)s_{n-1,AV} = s_n + (n-1)(s_n - s_{n-1,AV})$$

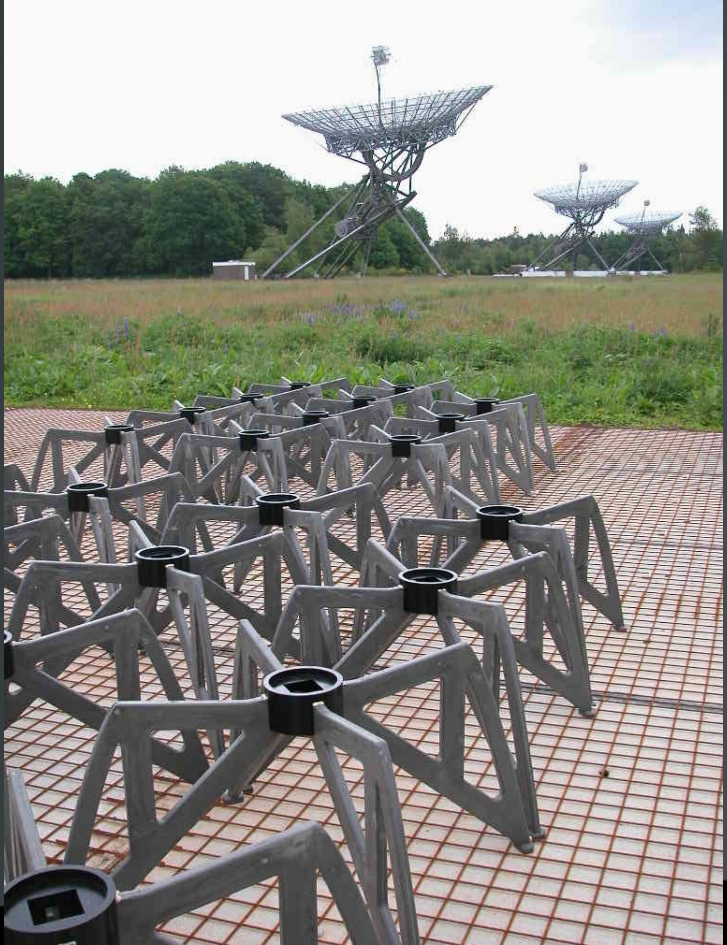
Which less biased than $E(s_n)$:

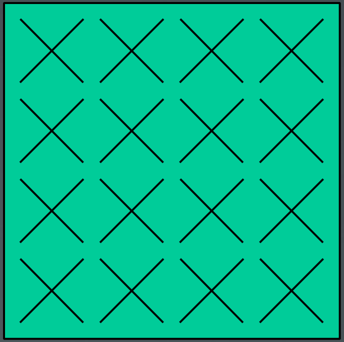
$$E(s'_n) - s = -a_2/n^2 + O(n^{-3})$$

The LOFAR telescope

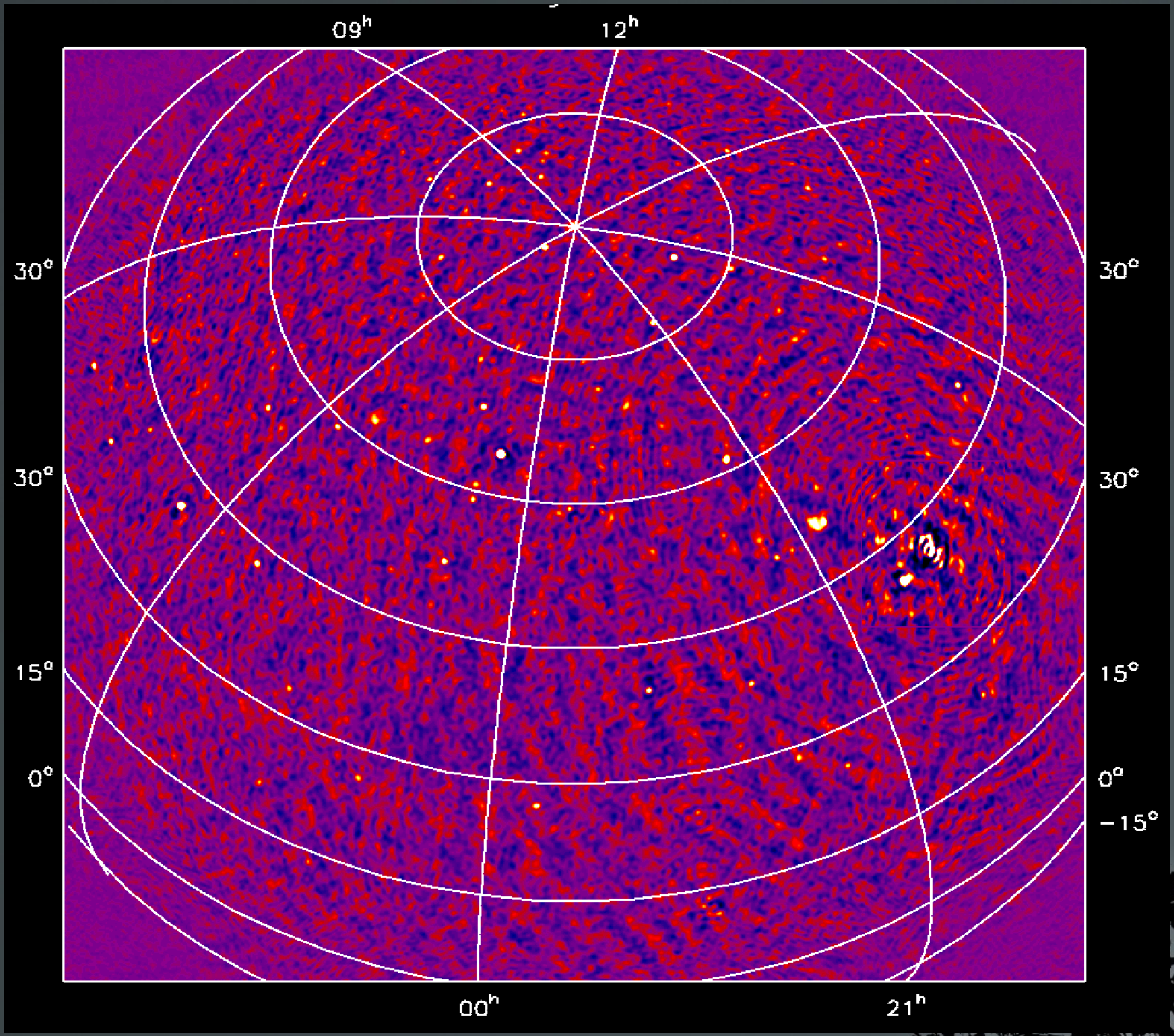


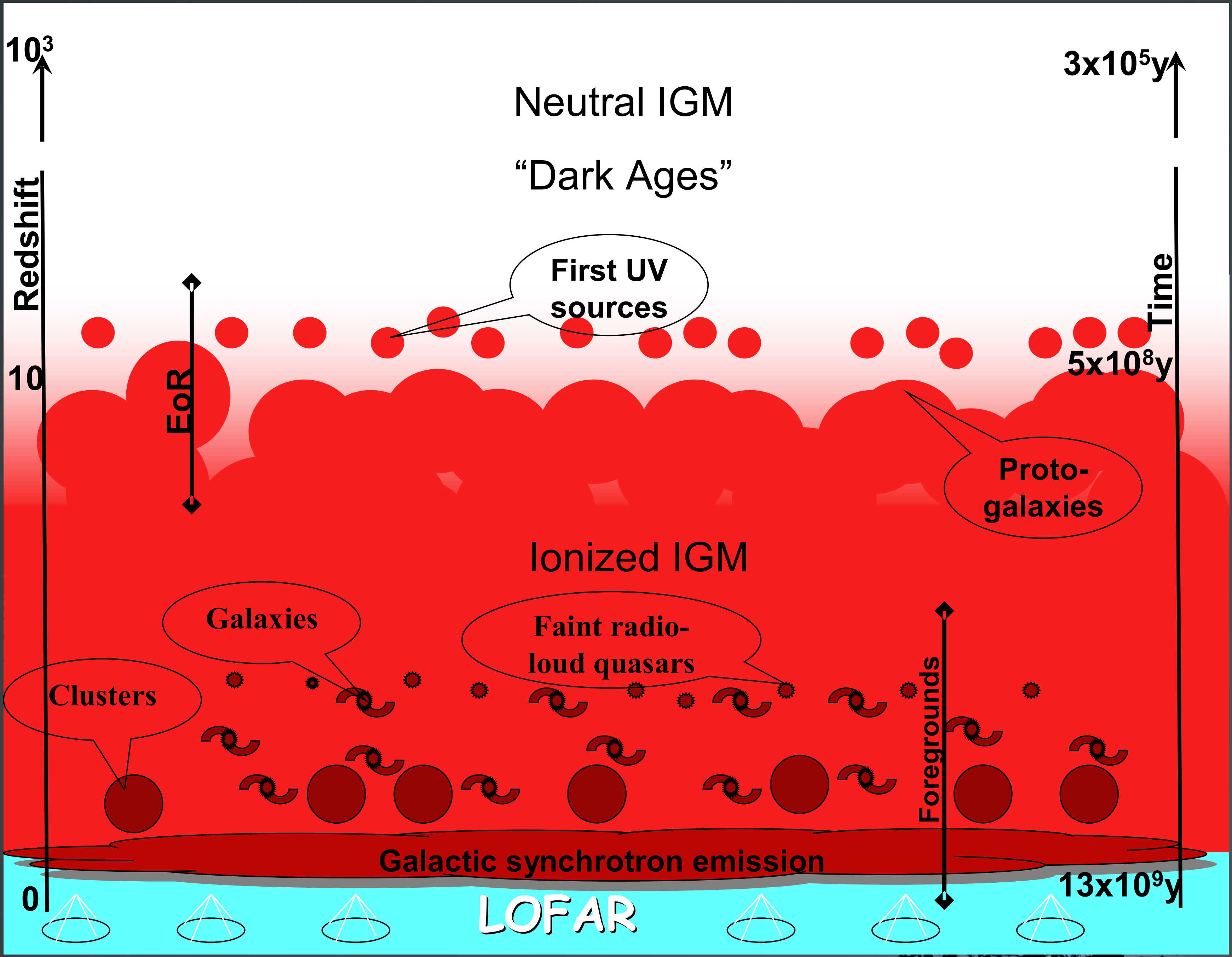






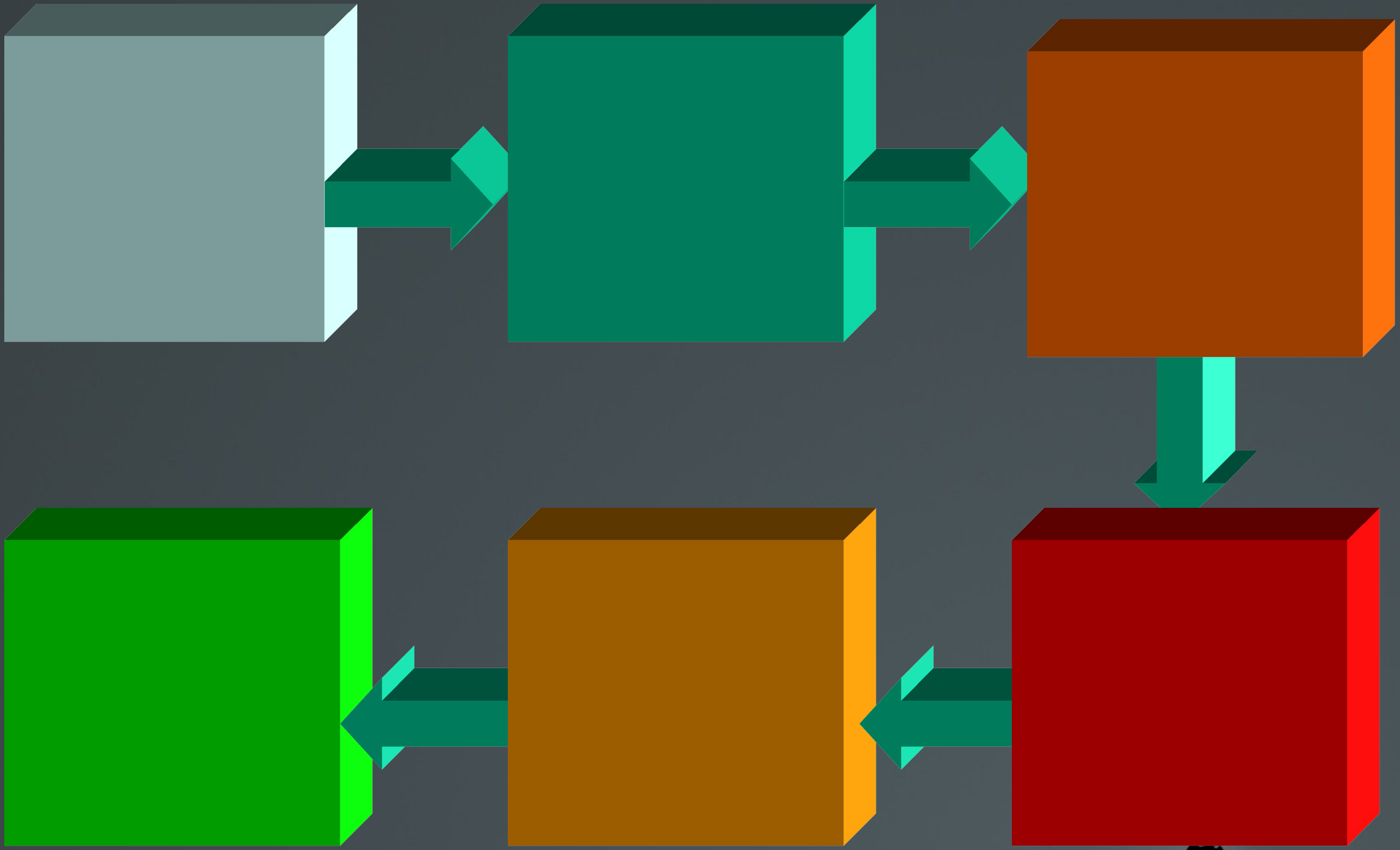
LOFAR test stations images

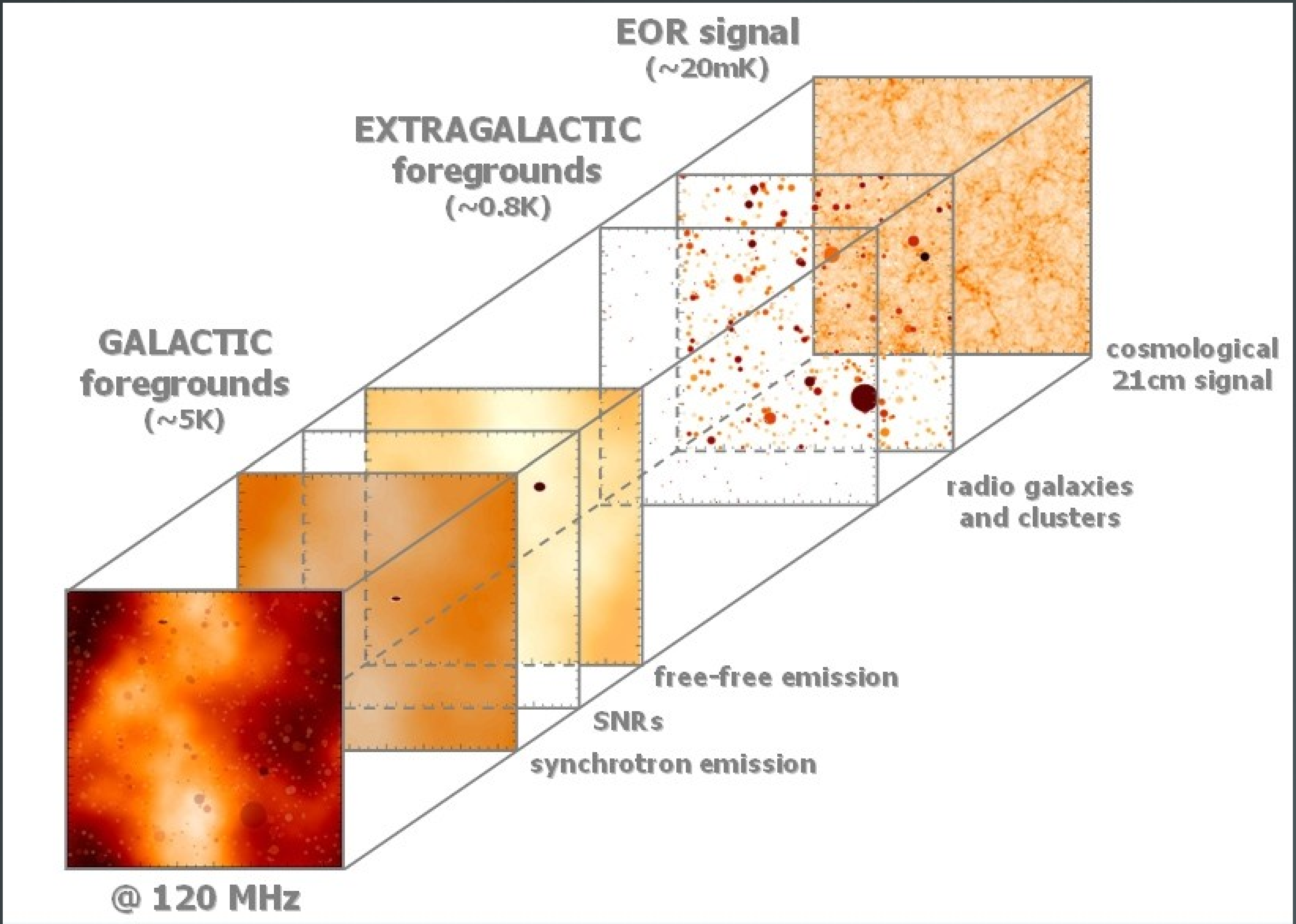




10^3
Redshift
10
0
$3x10^5y$
Time
$5x10^8y$
$13x10^9y$
Neutral IGM
"Dark Ages"
First UV sources
EoR
Proto-galaxies
Ionized IGM
Galaxies
Faint radio-loud quasars
Clusters
Foregrounds
Galactic synchrotron emission
LOFAR

Creating a dataset for LOFAR




EOR signal
(~20mK)
EXTRAGALACTIC
foregrounds
(~0.8K)
GALACTIC
foregrounds
(~5K)
cosmological
21cm signal
radio galaxies
and clusters
free-free emission
SNRs
synchrotron emission
@ 120 MHz