SLIM : GeoClimate tool tutorial
for Urban Mapping on HPC environment

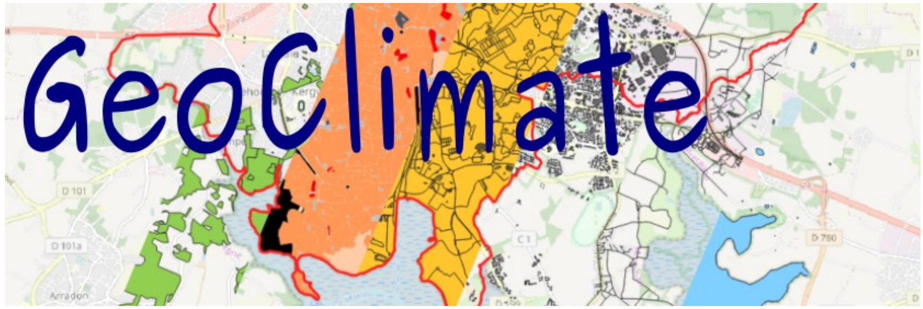
Emmanuel Renault, Elisabeth Le Saux, Erwan Bocher, Jérémy Bernard

Mai 2021

Knowledge Transfer

GeoClimate and HPCF

JAR	**Environment** How to **install** the GeoClimate library.
⚙	**Workflow** How to **configure** and **process** the workflow to execute the GeoClimate chain.
▶	**Demo** Application to SLIM domains.



Installation Requirements (HPCF)

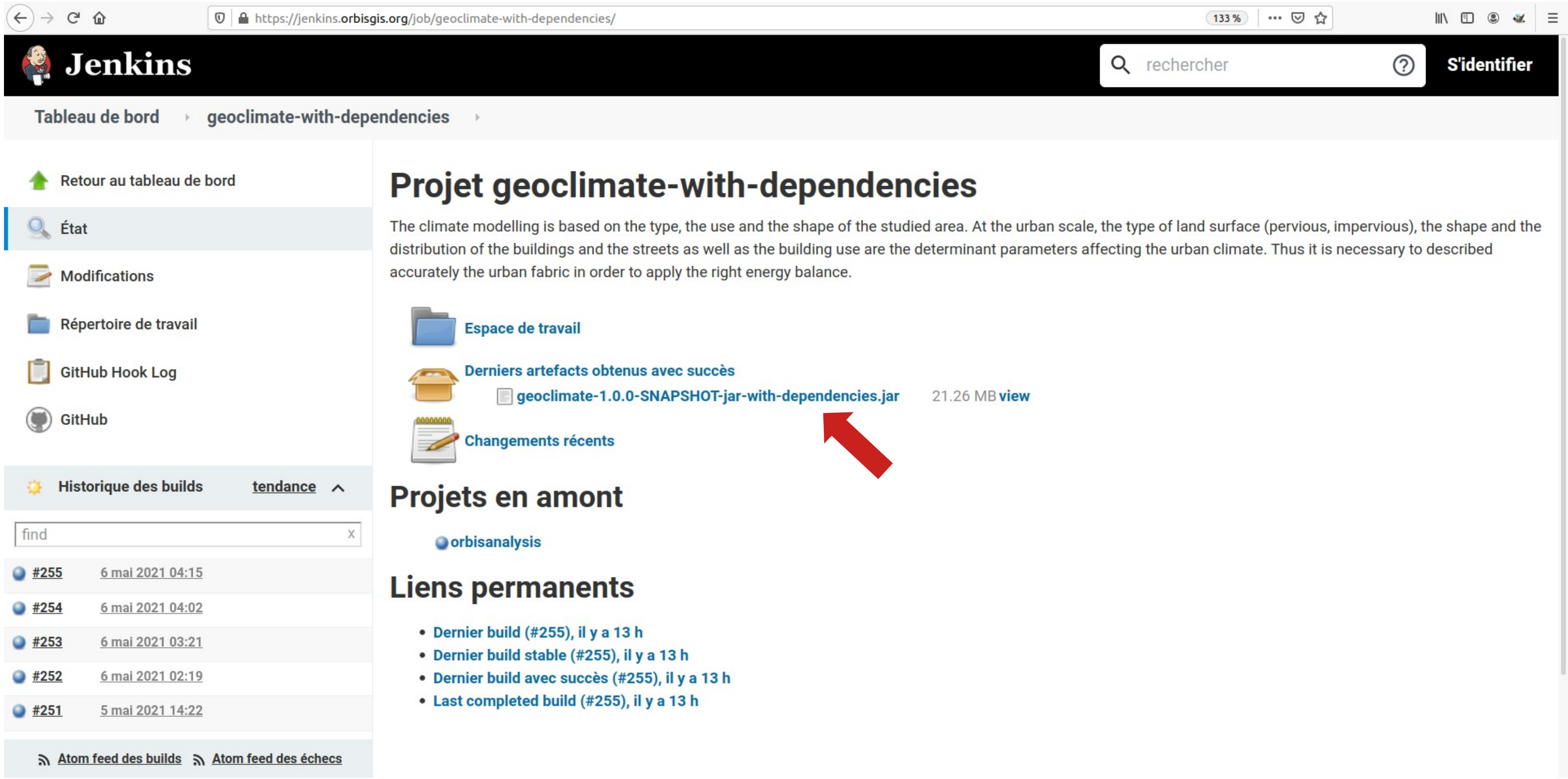
- Java / Groovy softwares (versions 11.0.6 / 3.0.7)
- External dependencies:

  Download the last available release of the GeoClimate library as part of CI/CD component on Jenkins platform.

  Copy the binary jar file to the appropriate location :

  $HOME/.groovy/grapes/org.orbisgis.orbisprocess/geoclimate/jars/
  → geoclimate-1.0.0-SNAPSHOT-jar-with-dependencies.jar

- Full description of the setup:
  https://confluence.ecmwf.int/display/SLIM/Documentation+and+user+manuel



https://jenkins.orbisgis.org/job/geoclimate-with-dependencies/

Historique des builds — tendance

find

#255	6 mai 2021 04:15
#254	6 mai 2021 04:02
#253	6 mai 2021 03:21
#252	6 mai 2021 02:19
#251	5 mai 2021 14:22

Atom feed des builds Atom feed des échecs

Projet geoclimate-with-dependencies

The climate modelling is based on the type, the use and the shape of the studied area. At the urban scale, the type of land surface (pervious, impervious), the shape and the distribution of the buildings and the streets as well as the building use are the determinant parameters affecting the urban climate. Thus it is necessary to described accurately the urban fabric in order to apply the right energy balance.

Espace de travail

Derniers artefacts obtenus avec succès

geoclimate-1.0.0-SNAPSHOT-jar-with-dependencies.jar 21.26 MB view

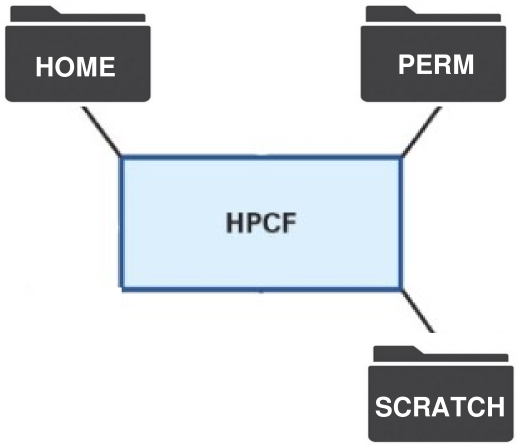
Changements récents

Projets en amont

orbisanalysis

Liens permanents

- Dernier build (#255), il y a 13 h
- Dernier build stable (#255), il y a 13 h
- Dernier build avec succès (#255), il y a 13 h
- Last completed build (#255), il y a 13 h



File system	Quota
Home	480 MB
Perm	26 GB
Scratch	30 TB (between all users)

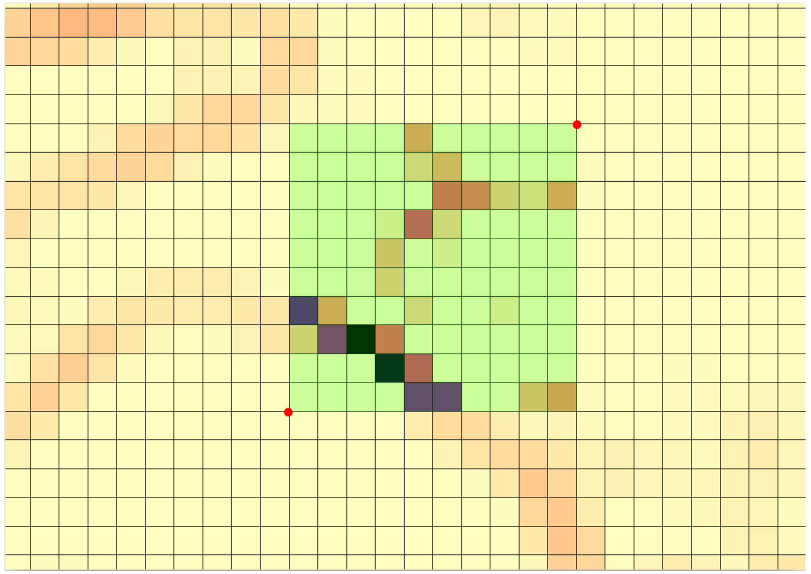
GeoClimate Delivrables

- HOME : GeoClimate scripts
  - **run_batch**
  - **geoclimate_batch**
  - **geoclimate_batch.gvy**
  - **slim_domains_1.xlsx**

- SCRATCH :
  GeoClimate chain output data stored in folders
  File formats: **.asc** **.geojson** **.prj**

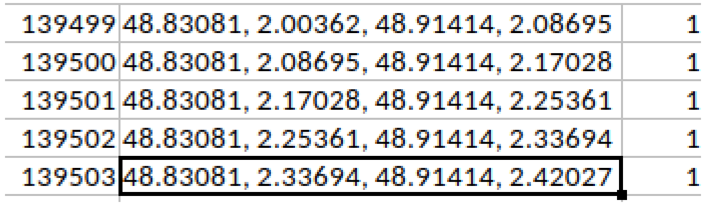
- PERM :
  → .zip data files moved to PERM on ECGATE for disk quota (100 GB)

SLIM Domains



NetCDF grid and SLIM domain over Paris area

- Domain : bounding box defined by its SW and NE lat-lon coordinates (red dots on the figure)
- SLIM Domains : Contiguous domains of 10x10 pixels extracted from the 1km NetCDF LSM grid.
- For each domain is assigned an identifier and a code (land-sea filtering).



id	bbox	code
139499	48.83081, 2.00362, 48.91414, 2.08695	1
139500	48.83081, 2.08695, 48.91414, 2.17028	1
139501	48.83081, 2.17028, 48.91414, 2.25361	1
139502	48.83081, 2.25361, 48.91414, 2.33694	1
139503	48.83081, 2.33694, 48.91414, 2.42027	1

XLSX → HPCF

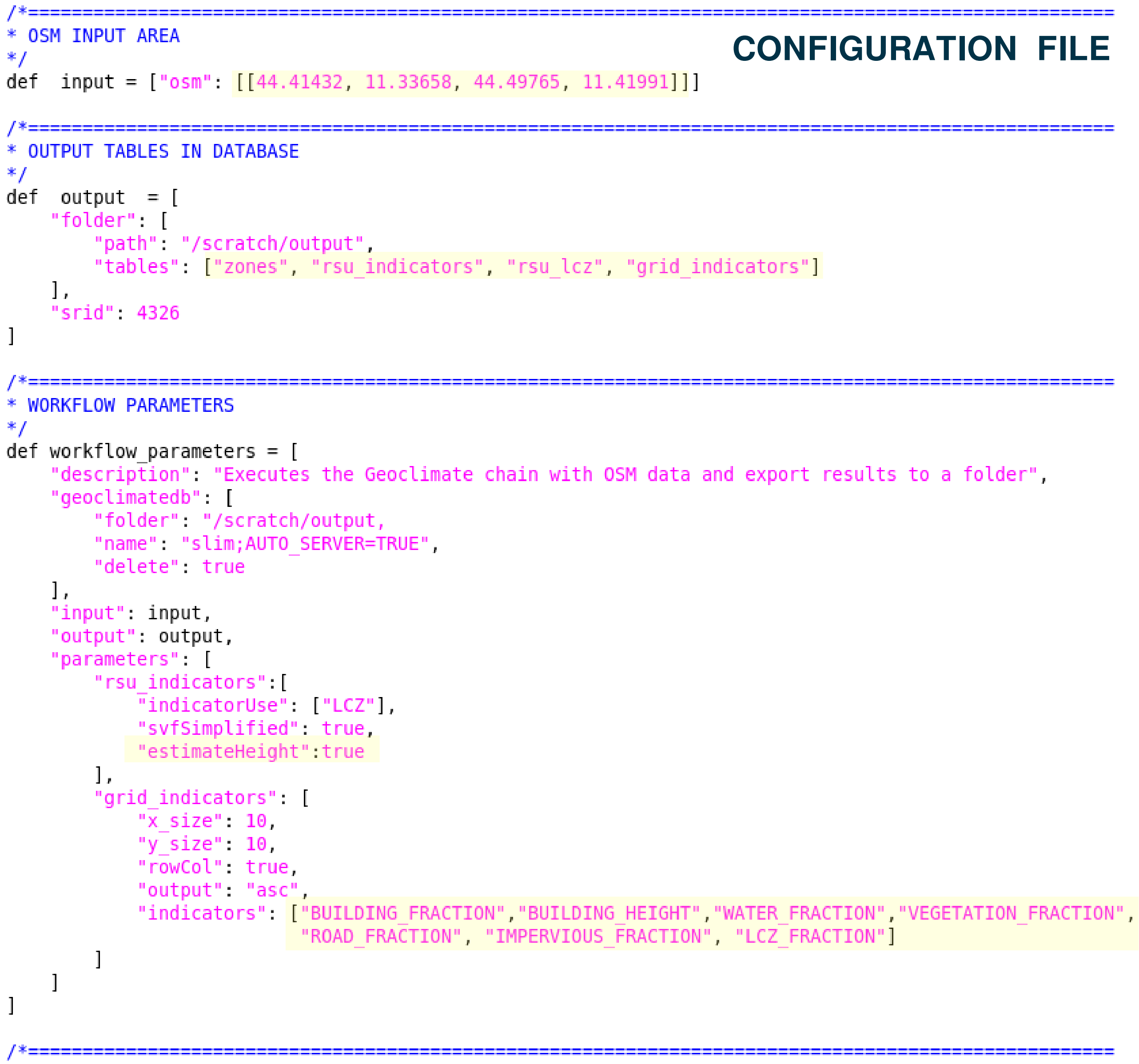
CONFIGURATION FILE

List of SLIM domains

List of attributes stored in database + Geojson files

Use of ML algorithm for heights estimates

List of indicators saved for the rasterization process: ASCII Grid files



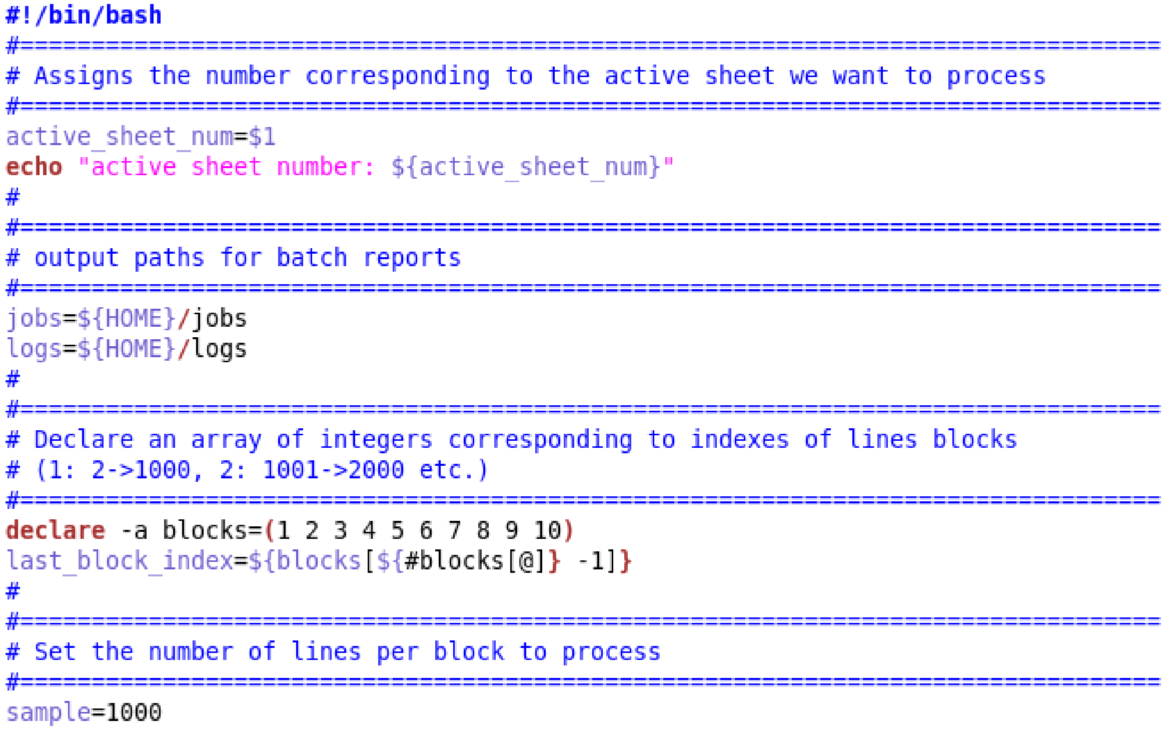
```
/*==========================================================================================
* OSM INPUT AREA
*/
def  input = ["osm": [[44.41432, 11.33658, 44.49765, 11.41991]]]

/*==========================================================================================
* OUTPUT TABLES IN DATABASE
*/
def  output  = [
    "folder": [
        "path": "/scratch/output",
        "tables": ["zones", "rsu_indicators", "rsu_lcz", "grid_indicators"]
    ],
    "srid": 4326
]

/*==========================================================================================
* WORKFLOW PARAMETERS
*/
def workflow_parameters = [
    "description": "Executes the Geoclimate chain with OSM data and export results to a folder",
    "geoclimatedb": [
        "folder": "/scratch/output,
        "name": "slim;AUTO_SERVER=TRUE",
        "delete": true
    ],
    "input": input,
    "output": output,
    "parameters": [
        "rsu_indicators":[
            "indicatorUse": ["LCZ"],
            "svfSimplified": true,
            "estimateHeight":true
        ],
        "grid_indicators": [
            "x_size": 10,
            "y_size": 10,
            "rowCol": true,
            "output": "asc",
            "indicators": ["BUILDING_FRACTION","BUILDING_HEIGHT","WATER_FRACTION","VEGETATION_FRACTION",
                           "ROAD_FRACTION", "IMPERVIOUS_FRACTION", "LCZ_FRACTION"]
        ]
    ]
]

/*==========================================================================================
```

RUN_BATCH



```
#!/bin/bash
#================================================================================
# Assigns the number corresponding to the active sheet we want to process
#================================================================================
active_sheet_num=$1
echo "active sheet number: ${active_sheet_num}"
#
#================================================================================
# output paths for batch reports
#================================================================================
jobs=${HOME}/jobs
logs=${HOME}/logs
#
#================================================================================
# Declare an array of integers corresponding to indexes of lines blocks
# (1: 2->1000, 2: 1001->2000 etc.)
#================================================================================
declare -a blocks=(1 2 3 4 5 6 7 8 9 10)
last_block_index=${blocks[${#blocks[@]} -1]}
#
#================================================================================
# Set the number of lines per block to process
#================================================================================
sample=1000
```

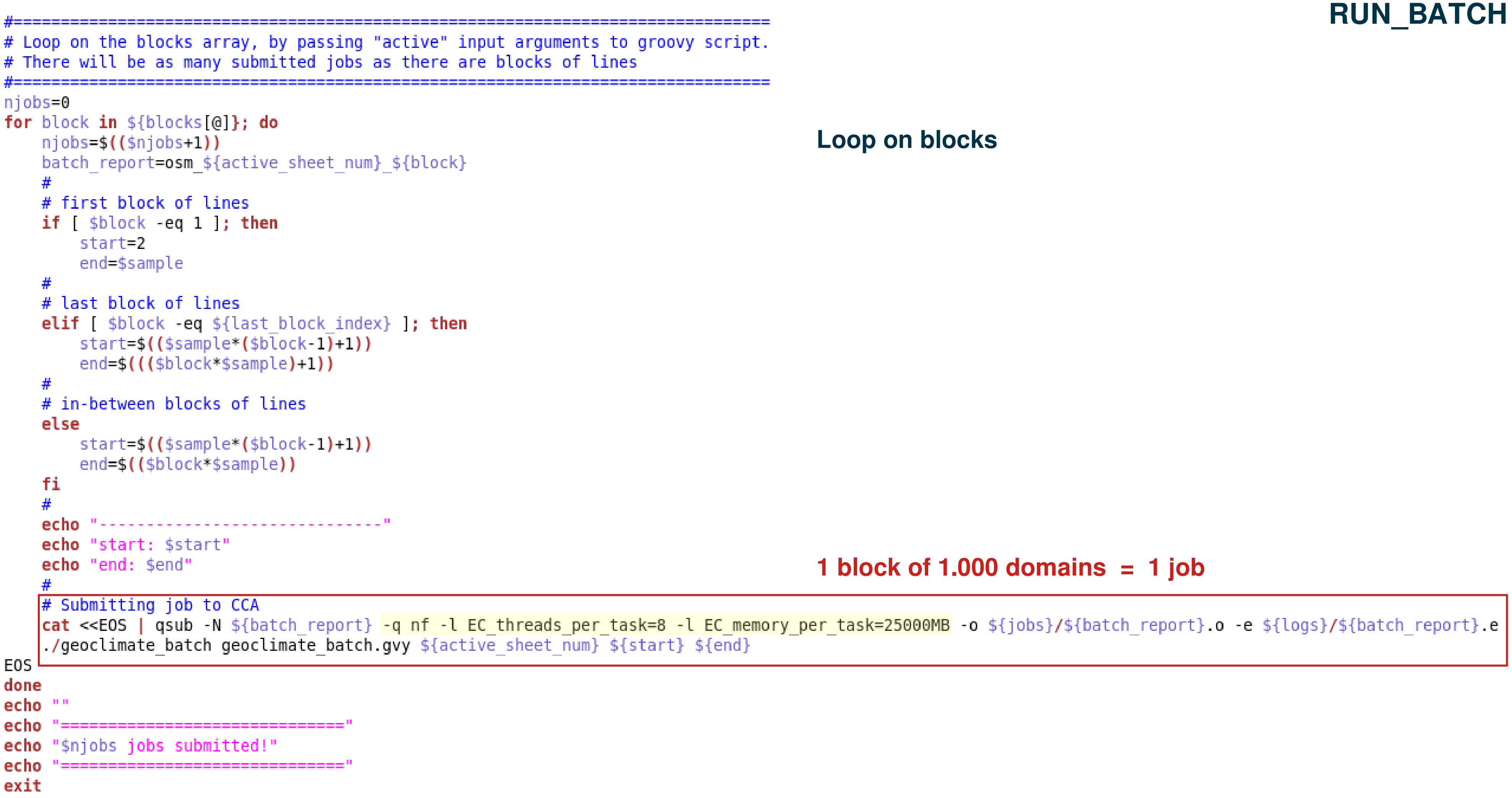
Workbook sheet

Redirected standard outputs

Blocks of domains

Number of domains per block

RUN_BATCH

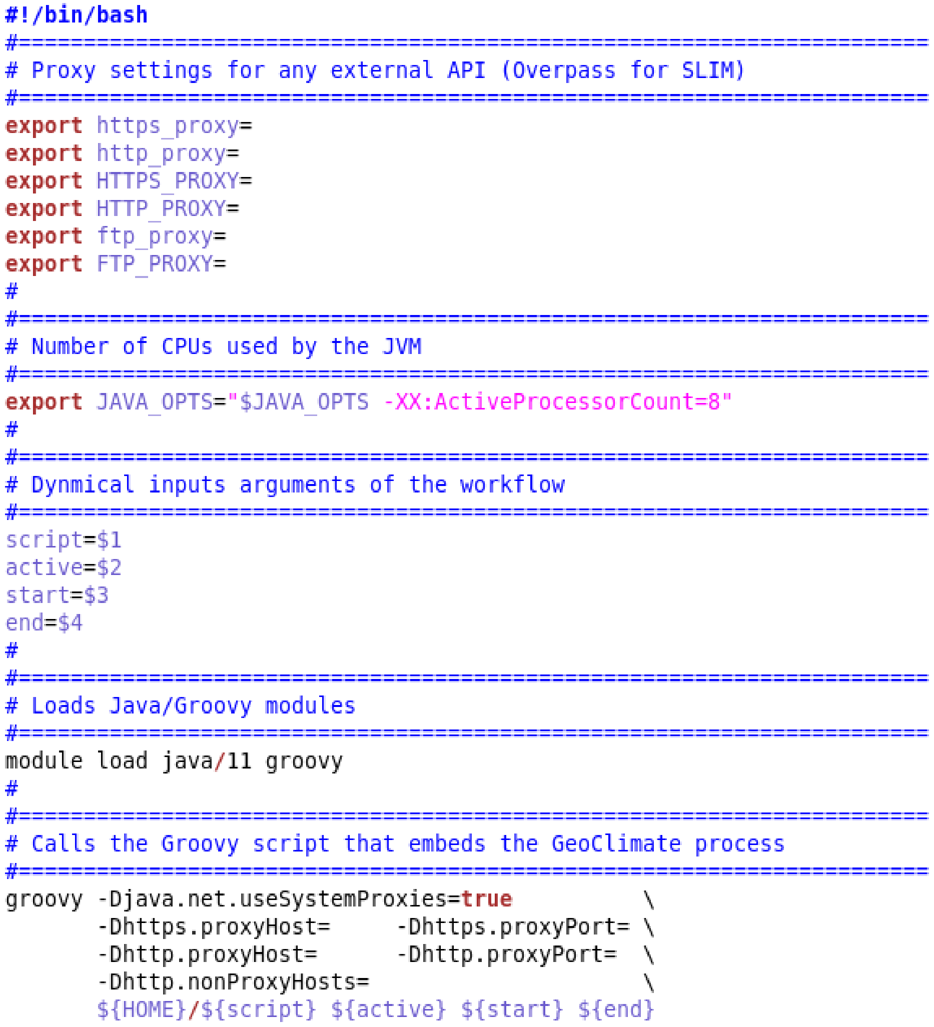


```
#==============================================================================
# Loop on the blocks array, by passing "active" input arguments to groovy script.
# There will be as many submitted jobs as there are blocks of lines
#==============================================================================
njobs=0
for block in ${blocks[@]}; do
    njobs=$(($njobs+1))
    batch_report=osm_${active_sheet_num}_${block}
    #
    # first block of lines
    if [ $block -eq 1 ]; then
        start=2
        end=$sample
    #
    # last block of lines
    elif [ $block -eq ${last_block_index} ]; then
        start=$(($sample*($block-1)+1))
        end=$((($block*$sample)+1))
    #
    # in-between blocks of lines
    else
        start=$(($sample*($block-1)+1))
        end=$(($block*$sample))
    fi
    #
    echo "------------------------------"
    echo "start: $start"
    echo "end: $end"
    #
    # Submitting job to CCA
    cat <<EOS | qsub -N ${batch_report} -q nf -l EC_threads_per_task=8 -l EC_memory_per_task=25000MB -o ${jobs}/${batch_report}.o -e ${logs}/${batch_report}.e
    ./geoclimate_batch geoclimate_batch.gvy ${active_sheet_num} ${start} ${end}
EOS
done
echo ""
echo "=============================="
echo "$njobs jobs submitted!"
echo "=============================="
exit
```

Loop on blocks

1 block of 1.000 domains = 1 job

GEOCLIMATE_BATCH



```
#!/bin/bash
#======================================================================
# Proxy settings for any external API (Overpass for SLIM)
#======================================================================
export https_proxy=
export http_proxy=
export HTTPS_PROXY=
export HTTP_PROXY=
export ftp_proxy=
export FTP_PROXY=
#
#======================================================================
# Number of CPUs used by the JVM
#======================================================================
export JAVA_OPTS="$JAVA_OPTS -XX:ActiveProcessorCount=8"
#
#======================================================================
# Dynmical inputs arguments of the workflow
#======================================================================
script=$1
active=$2
start=$3
end=$4
#
#======================================================================
# Loads Java/Groovy modules
#======================================================================
module load java/11 groovy
#
#======================================================================
# Calls the Groovy script that embeds the GeoClimate process
#======================================================================
groovy -Djava.net.useSystemProxies=true          \
       -Dhttps.proxyHost=     -Dhttps.proxyPort= \
       -Dhttp.proxyHost=      -Dhttp.proxyPort=  \
       -Dhttp.nonProxyHosts=                     \
       ${HOME}/${script} ${active} ${start} ${end}
```

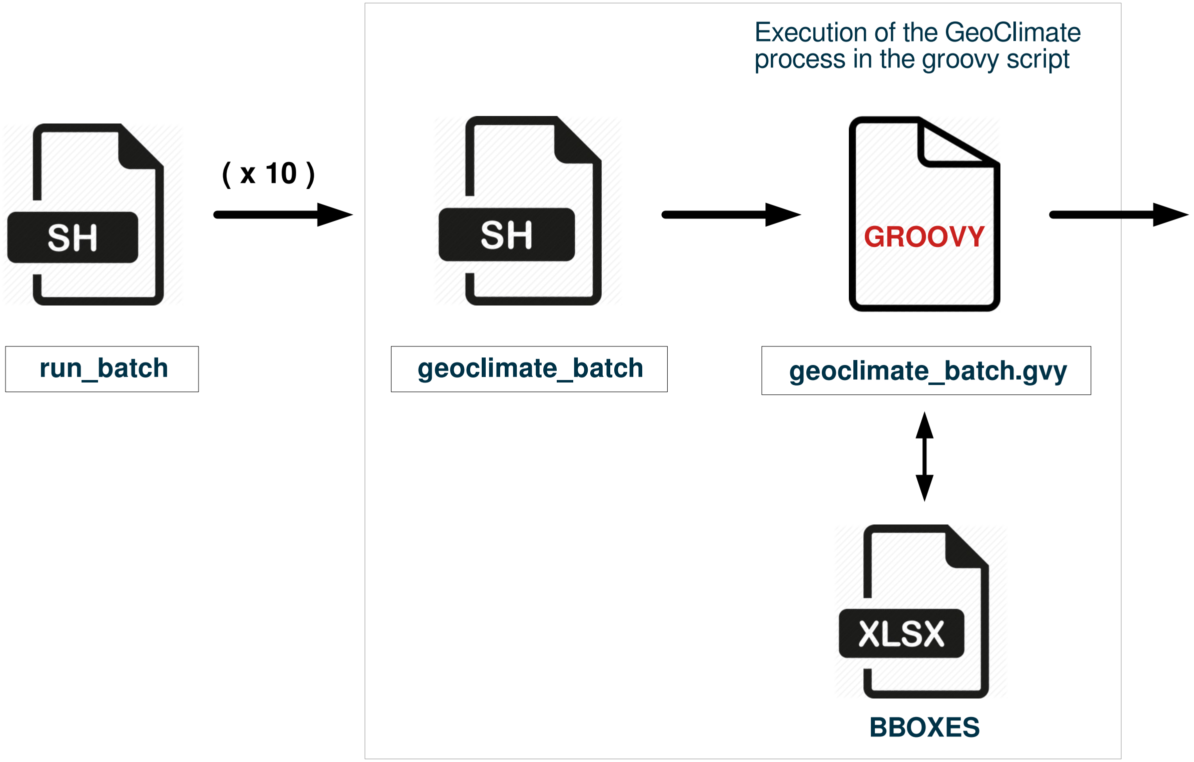
Proxy (OSM Raw Data)

CPU threshold for the JVM

Positional arguments

Required modules

Execution of the groovy script

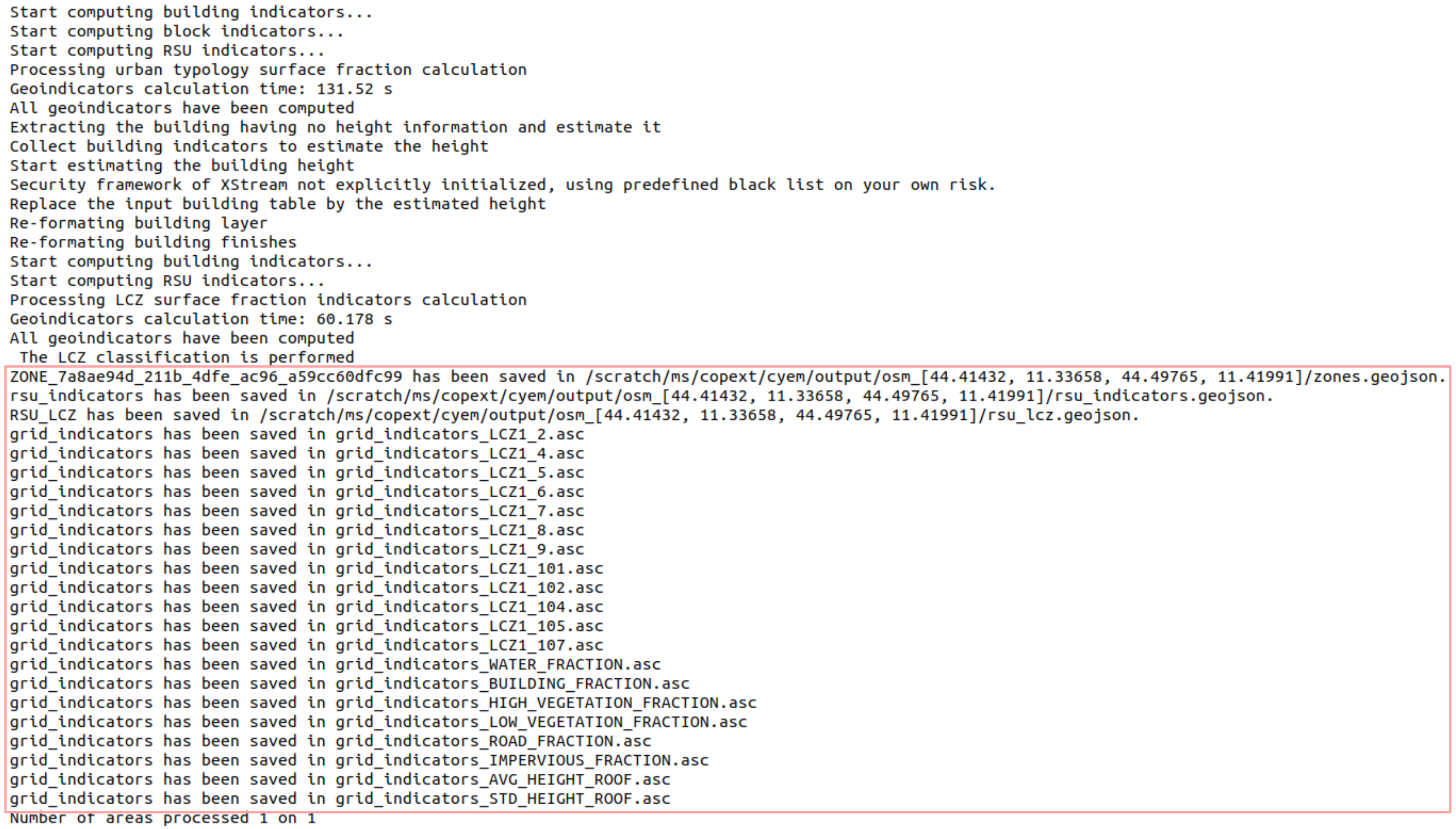


HPCF

Credit : Enrico Fucile, ECMWF

- Access: tsh/ssh
- File systems: HOME, PERM, SCRATCH
- Clusters (x 2): cca and ccb
- Batch scheduler : PBS directives
- Jobs submission: up to 20 per queue (nf)

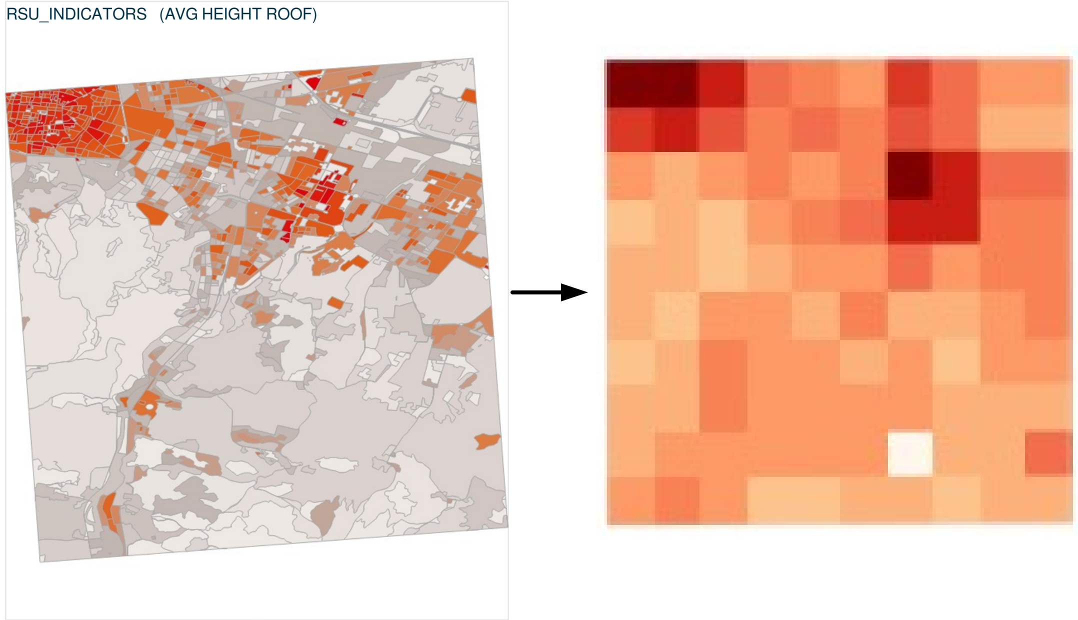
./demo_geoclimate

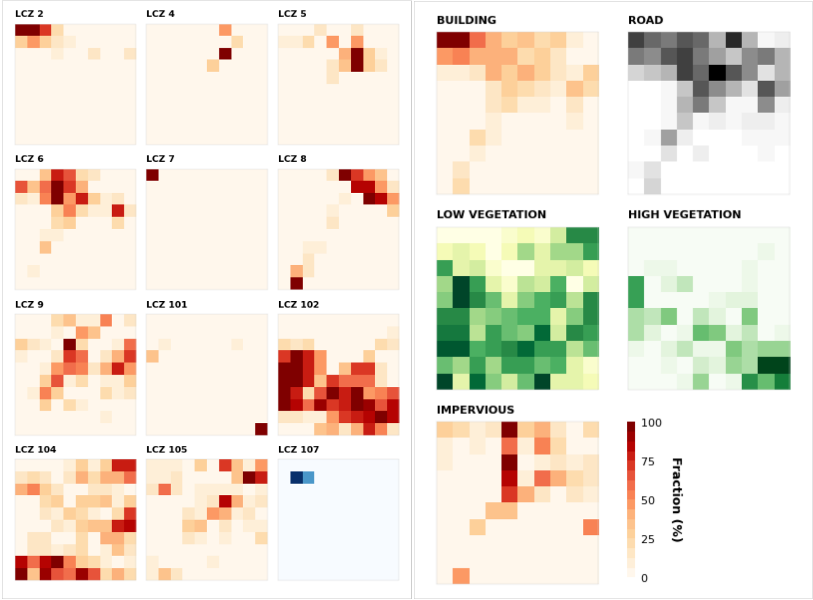


```
Start computing building indicators...
Start computing block indicators...
Start computing RSU indicators...
Processing urban typology surface fraction calculation
Geoindicators calculation time: 131.52 s
All geoindicators have been computed
Extracting the building having no height information and estimate it
Collect building indicators to estimate the height
Start estimating the building height
Security framework of XStream not explicitly initialized, using predefined black list on your own risk.
Replace the input building table by the estimated height
Re-formating building layer
Re-formating building finishes
Start computing building indicators...
Start computing RSU indicators...
Processing LCZ surface fraction indicators calculation
Geoindicators calculation time: 60.178 s
All geoindicators have been computed
 The LCZ classification is performed
ZONE_7a8ae94d_211b_4dfe_ac96_a59cc60dfc99 has been saved in /scratch/ms/copext/cyem/output/osm_[44.41432, 11.33658, 44.49765, 11.41991]/zones.geojson.
rsu_indicators has been saved in /scratch/ms/copext/cyem/output/osm_[44.41432, 11.33658, 44.49765, 11.41991]/rsu_indicators.geojson.
RSU_LCZ has been saved in /scratch/ms/copext/cyem/output/osm_[44.41432, 11.33658, 44.49765, 11.41991]/rsu_lcz.geojson.
grid_indicators has been saved in grid_indicators_LCZ1_2.asc
grid_indicators has been saved in grid_indicators_LCZ1_4.asc
grid_indicators has been saved in grid_indicators_LCZ1_5.asc
grid_indicators has been saved in grid_indicators_LCZ1_6.asc
grid_indicators has been saved in grid_indicators_LCZ1_7.asc
grid_indicators has been saved in grid_indicators_LCZ1_8.asc
grid_indicators has been saved in grid_indicators_LCZ1_9.asc
grid_indicators has been saved in grid_indicators_LCZ1_101.asc
grid_indicators has been saved in grid_indicators_LCZ1_102.asc
grid_indicators has been saved in grid_indicators_LCZ1_104.asc
grid_indicators has been saved in grid_indicators_LCZ1_105.asc
grid_indicators has been saved in grid_indicators_LCZ1_107.asc
grid_indicators has been saved in grid_indicators_WATER_FRACTION.asc
grid_indicators has been saved in grid_indicators_BUILDING_FRACTION.asc
grid_indicators has been saved in grid_indicators_HIGH_VEGETATION_FRACTION.asc
grid_indicators has been saved in grid_indicators_LOW_VEGETATION_FRACTION.asc
grid_indicators has been saved in grid_indicators_ROAD_FRACTION.asc
grid_indicators has been saved in grid_indicators_IMPERVIOUS_FRACTION.asc
grid_indicators has been saved in grid_indicators_AVG_HEIGHT_ROOF.asc
grid_indicators has been saved in grid_indicators_STD_HEIGHT_ROOF.asc
Number of areas processed 1 on 1
```

OVERVIEW FOR A SINGLE SLIM DOMAIN

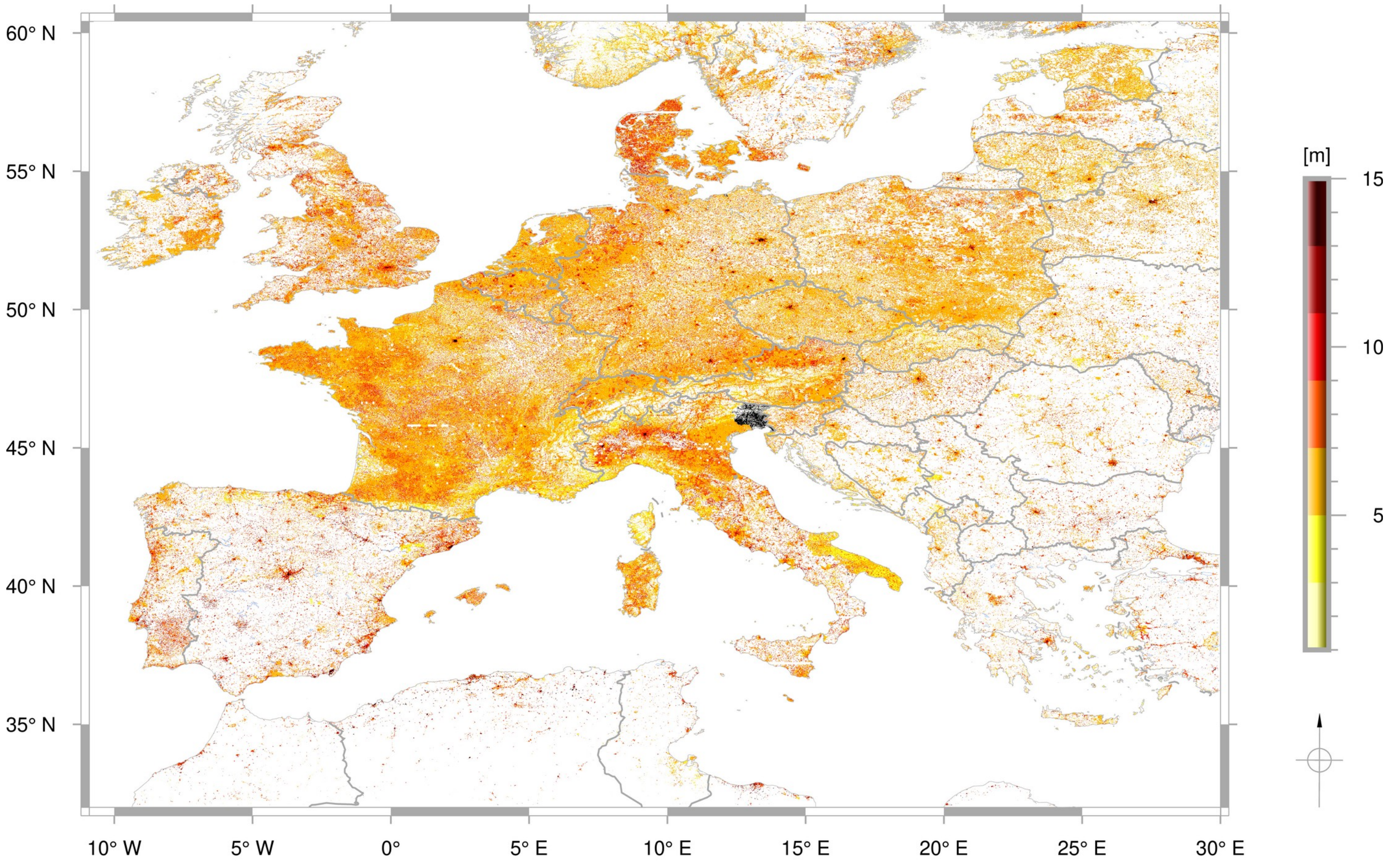
Bologna, ITA





OVERVIEW ON EUROPE AREA

AVERAGE OF ESTIMATED ROOF HEIGHTS



Thank you for your attention