

# EXAGEOSTAT-R INSTALLATION GUIDE

VERSION 1.0.0

November 11, 2018

## CONTENTS

1	Chapter 1. General Information	2
1.1	Introduction . . . . .	2
1.2	Supported Platforms . . . . .	2
1.3	Software Dependencies . . . . .	2
2	Chapter 2. macOS	3
2.1	Supported Releases . . . . .	3
2.2	Installing the prerequisite Packages . . . . .	3
2.3	ExaGeoStat on R . . . . .	4
2.4	Verifying the Installation . . . . .	5
3	Chapter 3. Linux	5
3.1	Supported Linux Distributions . . . . .	5
3.2	Installing the prerequisite Packages . . . . .	5
3.3	ExaGeoStat on R . . . . .	7
3.4	Verifying the Installation . . . . .	7
4	Chapter 4. Examples	8
4.1	Example 1: Synthetic generation of Geo-spatial data with MLE exact computation, . . . . .	8
4.2	Example 2: Synthetic generation of Geo-spatial data with MLE TLR computation, . . . . .	8
4.3	Example 3: Synthetic generation of Geo-spatial data with MLE DST computation, . . . . .	9
4.4	Example 4: Synthetic generation of measurements based on given Geo-spatial data locations with MLE Exact computation,	9

---

\* Extreme Computing Research Center (ECRC), KAUST, Thuwal, Saudi Arabia

## 1 CHAPTER 1. GENERAL INFORMATION

### 1.1 Introduction

This document describes how to install ExaGeoStatR on various platforms. One chapter is dedicated to each operating system.

### 1.2 Supported Platforms

ExaGeoStatR has been tested and is supported on the following operating systems:

1. MacOS.
2. Linux distributions.

**Note:** ExaGeoStat can be run practically on any Linux-like environment with a decent and fairly up-to-date C++ compiler, for example gcc 5.x. Certain ExaGeoStat-R features (Dense, Diagonal Super Tile(DST), Tile Low Rank(TLR), etc.) depend on the availability of external libraries (Nlopt, StarPU, Chameleon, HiCMA, Stars-H, etc.)

### 1.3 Software Dependencies

1. **Portable Hardware Locality (hwloc):** a software package provides a portable abstraction of the hierarchical topology of modern architectures.
2. **NLopt: a library for nonlinear optimization:** providing a common interface for a number of different free optimization routines available online as well as original implementations of various other algorithms.
3. **GNU Scientific Library (GSL):** a collection of routines for numerical computing.
4. **StarPU:** a task programming library for hybrid architectures.
5. **Chameleon:** a dense linear algebra software relying on sequential task-based algorithms where sub-tasks of the overall algorithms are submitted to a runtime system.
6. **HiCMA:** Hierarchical Computations on Manycore Architectures library, aims to redesign existing dense linear algebra libraries to exploit the data sparsity of the matrix operator.
7. **STARS-H:** a High performance parallel open-source package of Software for Testing Accuracy, Reliability and Scalability of Hierarchical computations.

## 2 CHAPTER 2. MACOS

### 2.1 Supported Releases

This Chapter provides additional information for installing ExaGeoStat on macOS. The following release are covered:

1. macOS high sierra 10.13.

### 2.2 Installing the prerequisite Packages

1. Downloading, Configuring and Building CMake 3.2.3 or higher,

```
$ wget https://cmake.org/files/v3.10/cmake-3.10.0-rc3.tar.gz
$ tar -zxvf cmake-3.10.0-rc3.tar.gz
$ cd cmake-3.10.0-rc3
$ ./configure
$ make -j && make -j install
```

2. Download an Install Intel MKL library from <https://software.intel.com/en-us/mkl> for macOS.

```
$ export MKLROOT=/opt/intel/mkl
```

3. Downloading, Configuring and Building NLOPT library 2.4.2 or higher,

```
$ wget http://ab-initio.mit.edu/nlopt/nlopt-2.4.2.tar.gz
$ tar -zxvf nlopt-2.4.2.tar.gz
$ cd nlopt-2.4.2
$ ./configure --enable-shared --without-guile
$ make -j && make -j install
```

4. Downloading, Configuring and Building GSL library 2.4 or higher,

```
$ wget https://ftp.gnu.org/gnu/gsl/gsl-2.4.tar.gz
$ tar -zxvf gsl-2.4.tar.gz
$ cd gsl-2.4
$ ./configure
$ make -j && make -j install
```

5. Downloading, Configuring and Building hwloc 1.11.5 or higher,

```
$ wget https://www.open-mpi.org/software/hwloc/v1.11/downloads/hwloc-1.11.5.tar.gz
$ tar -zxvf hwloc-1.11.5.tar.gz
$ cd hwloc-1.11.5
$ ./configure
$ make -j && make -j install
```

6. Downloading, Configuring and Building StarPU 1.2.5 or higher,

```
$ wget http://starpup.gforge.inria.fr/files/starpup-1.2.5/starpup-1.2.5.tar.gz
$ tar -zxvf starpu-1.2.5.tar.gz
$ cd starpu-1.2.5
$ ./configure --disable-cuda --disable-mpi --disable-opencl
$ make -j && make -j install
```

## 7. Download Chameleon, HiCMA, and STARS-H,

```
$ git clone https://github.com/ecrc/exageostat.git
$ cd exageostat
$ git submodule update --init --recursive
```

## 8. Configuring and Building Chameleon Software,

```
$ cd exageostat
$ cd hicma
$ cd chameleon
$ mkdir build && cd build
$ cmake .. -DCHAMELEON_USE_MPI=OFF -DBUILD_SHARED_LIBS=ON
-DCLAS_DIR="${MKLR00T}" -DLAPACK_DIR="${MKLR00T}"
-DBLAS_LIBRARIES="-L${MKLR00T}/lib;-lmkl_intel_lp64;-lmkl_core;
-lmkl_sequential;-lpthread;-lm;-ldl"
$ make
$ make install
```

## 9. Configuring and Building STARS-H Library (optional),

```
$ cd exageostat
$ cd stars-h
$ mkdir build
$ cmake .. -DCMAKE_C_FLAGS=-fPIC
$ make -j && make install
```

## 10. Configuring and Building HiCMA library (optional):

```
$ cd ..
# mkdir build && cd build
$ cmake .. -DCMAKE_C_FLAGS=-fPIC -DCLAS_DIR="${MKLR00T}"
-DLAPACK_DIR="${MKLR00T}"
-DBLAS_LIBRARIES="-L${MKLR00T}/lib;-lmkl_intel_lp64;-lmkl_core;
-lmkl_sequential;-lpthread;-lm;-ldl"
$ make
$ make install
```

**Note:** Here, we assume ROOT access during installation. if not,

1. `-DCMAKE_INSTALL_PREFIX=$PWD/installdir` should be added to the `cmake` commands, where `installdir` is your local installation directory.
2. `-prefix=$PWD/installdir` should be added to the `configure` commands, where `installdir` is your local installation directory.
3. `pkgconfig` for `NLOPT`, `GSL`, `hwloc`, `StarPU`, `Chameleon`, `STARS-H`, `HiCMA` paths need to be exported to `PKG_CONFIG_PATH` environment variable.

## 2.3 ExaGeoStat on R

1. Install latest ExaGeoStat R version hosted on GitHub:

```
install.packages("devtools")
library(devtools)
install_git(url="https://github.com/ecrc/exageostatR")
library(exageostat)
```

## 2.4 Verifying the Installation

Run one or more examples from Chapter 4.

# 3 CHAPTER 3. LINUX

## 3.1 Supported Linux Distributions

This chapter provides instructions for installing ExaGeoStat on selected Linux distributions:

1. Ubuntu 16.04 LTS.
2. Fedora Core 25.
3. Red Hat Enterprise Linux Desktop Workstation 7.x
4. OpenSUSE 42.

## 3.2 Installing the prerequisite Packages

1. Downloading, Configuring and Building CMake 3.2.3 or higher,
 

```
$ wget https://cmake.org/files/v3.10/cmake-3.10.0-rc3.tar.gz
$ tar -zxvf cmake-3.10.0-rc3.tar.gz
$ cd cmake-3.10.0-rc3
$ ./configure
$ make -j && make -j install
```
2. Download an Install Intel MKL library from <https://software.intel.com/en-us/mkl> for Linux.
 

```
$ export MKLROOT=/opt/intel/mkl
```
3. Downloading, Configuring and Building NLOPT library 2.4.2 or higher,
 

```
$ wget http://ab-initio.mit.edu/nlopt/nlopt-2.4.2.tar.gz
$ tar -zxvf nlopt-2.4.2.tar.gz
$ cd nlopt-2.4.2
$ ./configure --enable-shared --without-guile
$ make -j && make -j install
```
4. Downloading, Configuring and Building GSL library 2.4 or higher,
 

```
$ wget https://ftp.gnu.org/gnu/gsl/gsl-2.4.tar.gz
$ tar -zxvf gsl-2.4.tar.gz
$ cd gsl-2.4
$ ./configure
$ make -j && make -j install
```

## 5. Downloading, Configuring and Building hwloc 1.11.5 or higher,

```
$ wget https://www.open-mpi.org/software/hwloc/v1.11/
downloads/hwloc-1.11.5.tar.gz
$ tar -zxvf hwloc-1.11.5.tar.gz
$ cd hwloc-1.11.5
$ ./configure
$ make -j && make -j install
```

## 6. Downloading, Configuring and Building StarPU 1.2.5 or higher,

```
$ wget http://starpup.gforge.inria.fr/files/starpup-1.2.5/
starpup-1.2.5.tar.gz
$ tar -zxvf starpu-1.2.5.tar.gz
$ cd starpu-1.2.5
$ ./configure --disable-cuda --disable-mpi --disable-openc
$ make -j && make -j install
```

**StarPU Configuration:** In the case of GPUs systems, both cuda and openc should be enabled using -enable option. If MPI is required, it should be also enabled using -enable option.

## 7. Download Chameleon, HiCMA, and STARS-H,

```
$ git clone https://github.com/ecrc/exageostat.git
$ cd exageostat
$ git submodule update --init --recursive
```

## 8. Configuring and Building Chameleon Software,

```
$ cd exageostat
$ cd hicma
$ cd chameleon
$ mkdir build && cd build
$ cmake .. -DCHAMELEON_USE_MPI=OFF -DBUILD_SHARED_LIBS=ON
-DCLAS_DIR="${MKLR00T}" -DLAPACK_DIR="${MKLR00T}"
-DCLAS_LIBRARIES="-L${MKLR00T}/lib;-lmkl_intel_lp64;-lmkl_core;
-lmkl_sequential;-lpthread;-lm;-ldl"
$ make
$ make install
```

**Chameleon Configuration:** In the case of GPUs systems, -DCHAMELEON\_USE\_CUDA=ON If MPI is required, it should be also enabled using -DCHAMELEON\_USE\_MPI=ON.

## 9. Configuring and Building STARS-H Library (optional),

```
$ cd exageostat
$ cd stars-h
$ mkdir build
$ cmake .. -DCMAKE_C_FLAGS=-fPIC
$ make -j && make install
```

## 10. Configuring and Building HiCMA library (optional):

```

$ cd ..
# mkdir build && cd build
$ cmake .. -DCMAKE_C_FLAGS=-fPIC -DCBLAS_DIR="${MKLR00T}"
-DLAPACK_DIR="${MKLR00T}"
-DBLAS_LIBRARIES="-L${MKLR00T}/lib;-lmkl_intel_lp64;-lmkl_core;
-lmkl_sequential;-lpthread;-lm;-ldl"
$ make
$ make install

```

**Note:** Here, we assume ROOT access during installation. if not,

1. `-DCMAKE_INSTALL_PREFIX=$PWD/installdir` should be added to the `cmake` commands, where `installdir` is your local installation directory.
2. `-prefix=$PWD/installdir` should be added to the `configure` commands, where `installdir` is your local installation directory.
3. `pkgconfig` for `NLOPT`, `GSL`, `hwloc`, `StarPU`, `Chameleon`, `STARS-H`, `HiCMA` paths need to be exported to `PKG_CONFIG_PATH` environment variable.

## 3.3 ExaGeoStat on R

1. Install latest ExaGeoStat R version hosted on GitHub:

```

install.packages("devtools")
library(devtools)
install_git(url="https://github.com/ecrc/exageostatR")
library(exageostat)

```

## 3.4 Verifying the Installation

Run one or more examples from Chapter 4.

## 4 CHAPTER 4. EXAMPLES

### 4.1 Example 1: Synthetic generation of Geo-spatial data with MLE exact computation,

```

library("exageostat")           #Load ExaGeoStat-R lib.
seed      = 0                    #Initial seed to generate XY locs.
theta1    = 1                    #Initial variance.
theta2    = 0.1                  #Initial range.
theta3    = 0.5                  #Initial smoothness.
dmetric   = 0                    #0 --> Euclidean distance, 1--> great circle distance.
n         = 1600                 #n*n locations grid.
ncores    = 2                    #Number of underlying CPUs.
gpu       = 0                    #Number of underlying GPUs.
ts       = 320                  #Tile_size: changing it can improve the performance.
p_grid   = 1                    #More than 1 in the case of distributed systems.
q_grid   = 1                    #More than 1 in the case of distributed systems.
clb      = vector(mode="double",length = 3) #Optimization lower bounds values.
cub      = vector(mode="double",length = 3) #Optimization upper bounds values.
theta_out  = vector(mode="double",length = 3) #Parameter vector output.
globalveclen = 3*n
vecs_out   = vector(mode="double",length = globalveclen) #Z measurements of n locations.
clb       = as.double(c("0.01", "0.01", "0.01")) #Optimization lower bounds.
cub       = as.double(c("5.00", "5.00", "5.00")) #Optimization upper bounds.
vecs_out[1:globalveclen] = -1.99
theta_out[1:3] = -1.99
exageostat_initR(ncores, gpu, ts)           #Initiate exageostat instance.
vecs_out = exageostat_egenzR(n, ncores, gpu, ts, p_grid, q_grid,
theta1, theta2, theta3, dmetric, seed, globalveclen) #Generate Z observation vector.
theta_out = exageostat_emleR(n, ncores, gpu, ts, p_grid, q_grid,
vecs_out[1:n], vecs_out[n+1:(2*n)],
vecs_out[(2*n+1):(3*n)], clb, cub, dmetric, 0.0001, 20) #Exact Estimation (MLE).
exageostat_finalizeR()                       #Finalize exageostat instance
=====

```

### 4.2 Example 2: Synthetic generation of Geo-spatial data with MLE TLR computation,

```

library("exageostat")           #Load ExaGeoStat-R lib.
seed      = 0                    #Initial seed to generate XY locs.
theta1    = 1                    #Initial variance.
theta2    = 0.03                 #Initial range.
theta3    = 0.5                  #Initial smoothness.
dmetric   = 0                    #0 --> Euclidean distance, 1--> great circle distance.
n         = 900                  #n*n locations grid.
ncores    = 4                    #Number of underlying CPUs.
gpu       = 0                    #Number of underlying GPUs.
ts       = 320                  #Tile_size: changing it can improve the performance.
lts      = 600                  #TLR_Tile_size: changing it can improve the performance.
tlr_acc   = 7                   #approximation accuracy 10^-(acc).
tlr_maxrank = 450              #Max rank.
p_grid    = 1                   #More than 1 in the case of distributed systems.
q_grid    = 1                   #More than 1 in the case of distributed systems.

```



```

clb      = vector(mode="double",length = 3) #Optimization lower bounds values.
cub      = vector(mode="double",length = 3) #Optimization upper bounds values.
theta_out = vector(mode="double",length = 3) #Parameter vector output.
globalveclen = 3*n
vecs_out = vector(mode="double",length = globalveclen)#Z measurements of n locations.
clb      = as.double(c("0.01", "0.01", "0.01")) #Optimization lower bounds.
cub      = as.double(c("5.00", "5.00", "5.00")) #Optimization upper bounds.
vecs_out[1:globalveclen] = -1.99
theta_out[1:3] = -1.99
exageostat_initR(ncores, gpus, ts) #Initiate exageostat instance.
vecs_out = exageostat_egenzR(n, ncores, gpus, ts, p_grid, q_grid,
theta1, theta2, theta3, dmetric, seed, globalveclen) #Generate Z observation vector.
theta_out = exageostat_tlrmlrR(n, ncores, gpus, lts, p_grid, q_grid,
vecs_out[1:n], vecs_out[n+1:(2*n)], vecs_out[(2*n+1):(3*n)],
clb, cub, tlr_acc, tlr_maxrank, dmetric, 0.0001, 20) #TLR Estimation (MLE).
exageostat_finalizeR() #Finalize exageostat instance.

```

#### 4.3 Example 3: Synthetic generation of Geo-spatial data with MLE DST computation,

```

library("exageostat") #Load ExaGeoStat-R lib.
seed      = 0 #Initial seed to generate XY locs.
theta1    = 1 #Initial variance.
theta2    = 0.1 #Initial range.
theta3    = 0.5 #Initial smoothness.
dmetric   = 0 #0 --> Euclidean distance, 1--> great circle distance.
n         = 1600 #n*n locations grid.
ncores    = 2 #Number of underlying CPUs.
gpus      = 0 #Number of underlying GPUs.
ts        = 320 #Tile_size: changing it can improve the performance.
p_grid    = 1 #More than 1 in the case of distributed systems.
q_grid    = 1 #More than 1 in the case of distributed systems.
clb      = vector(mode="double",length = 3) #Optimization lower bounds values.
cub      = vector(mode="double",length = 3) #Optimization upper bounds values.
theta_out = vector(mode="double",length = 3) #Parameter vector output.
globalveclen = 3*n
vecs_out = vector(mode="double",length = globalveclen)#Z measurements of n locations.
clb      = as.double(c("0.01", "0.01", "0.01")) #Optimization lower bounds.
cub      = as.double(c("5.00", "5.00", "5.00")) #Optimization upper bounds.
vecs_out[1:globalveclen] = -1.99
theta_out[1:3] = -1.99
exageostat_initR(ncores, gpus, ts) #Initiate exageostat instance.
vecs_out = exageostat_egenzR(n, ncores, gpus, ts, p_grid, q_grid,
theta1, theta2, theta3, dmetric, seed, globalveclen) #Generate Z observation vector.
theta_out = exageostat_dstmlrR(n, ncores, gpus, ts, p_grid, q_grid,
vecs_out[1:n], vecs_out[n+1:(2*n)],
vecs_out[(2*n+1):(3*n)], clb, cub, dmetric, 0.0001, 20) #DST Estimation (MLE).
exageostat_finalizeR() #Finalize exageostat instance

```

#### 4.4 Example 4: Synthetic generation of measurements based on given Geo-spatial data locations with MLE Exact computation,

```

library("exageostat")           #Load ExaGeoStat-R lib.
seed      = 0                    #Initial seed to generate XY locs.
theta1    = 1                    #Initial variance.
theta2    = 0.1                  #Initial range.
theta3    = 0.5                  #Initial smoothness.
dmetric   = 0                    #0 --> Euclidean distance, 1--> great circle distance.
n         = 1600                 #n*n locations grid.
ncores    = 2                    #Number of underlying CPUs.
gpu       = 0                    #Number of underlying GPUs.
ts       = 320                  #Tile_size: changing it can improve the performance.
p_grid   = 1                    #More than 1 in the case of distributed systems.
q_grid   = 1                    #More than 1 in the case of distributed systems.
clb      = vector(mode="double",length = 3) #Optimization lower bounds values.
cub      = vector(mode="double",length = 3) #Optimization upper bounds values.
theta_out  = vector(mode="double",length = 3) #Parameter vector output.
globalveclen = 3*n
x          = rnorm(n = globalveclen, mean = 39.74, sd = 25.09)
#X dimension vector of n locations.
y          = rnorm(n = globalveclen, mean = 80.45, sd = 100.19)
#Y dimension vector of n locations.
vecs_out   = vector(mode="double",length = globalveclen)#Z measurements of n locations.
clb       = as.double(c("0.01", "0.01", "0.01")) #Optimization lower bounds.
cub       = as.double(c("5.00", "5.00", "5.00")) #Optimization upper bounds.
vecs_out[1:globalveclen] = -1.99
theta_out[1:3] = -1.99
exageostat_initR(ncores, gpu, ts) #Initiate exageostat instance.
vecs_out      = exageostat_egenz_glr(n, ncores, gpu, ts, p_grid, q_grid,
x, y, theta1, theta2, theta3,
dmetric, globalveclen) #Generate Z observation vector based on given locations.
theta_out     = exageostat_emleR(n, ncores, gpu, ts, p_grid, q_grid,
vecs_out[1:n], vecs_out[n+1:(2*n)],
vecs_out[(2*n+1):(3*n)], clb, cub, dmetric, 0.0001, 20) #Exact Estimation (MLE).
exageostat_finalizeR() #Finalize exageostat instance

```