CSC

ICT Solutions for Brilliant Minds

Spatial analysis with R in CSC supercomputer Puhti

Kylli Ek, Samantha Wittke, CSC

Zoom, 2.11.2021

Non-profit state organization with special tasks

Turn over in 2020

55M€

Headquarters in Espoo, datacenter in Kajaani

Owned by state (70%)

and all Finnish higher education institutions **(30%)**

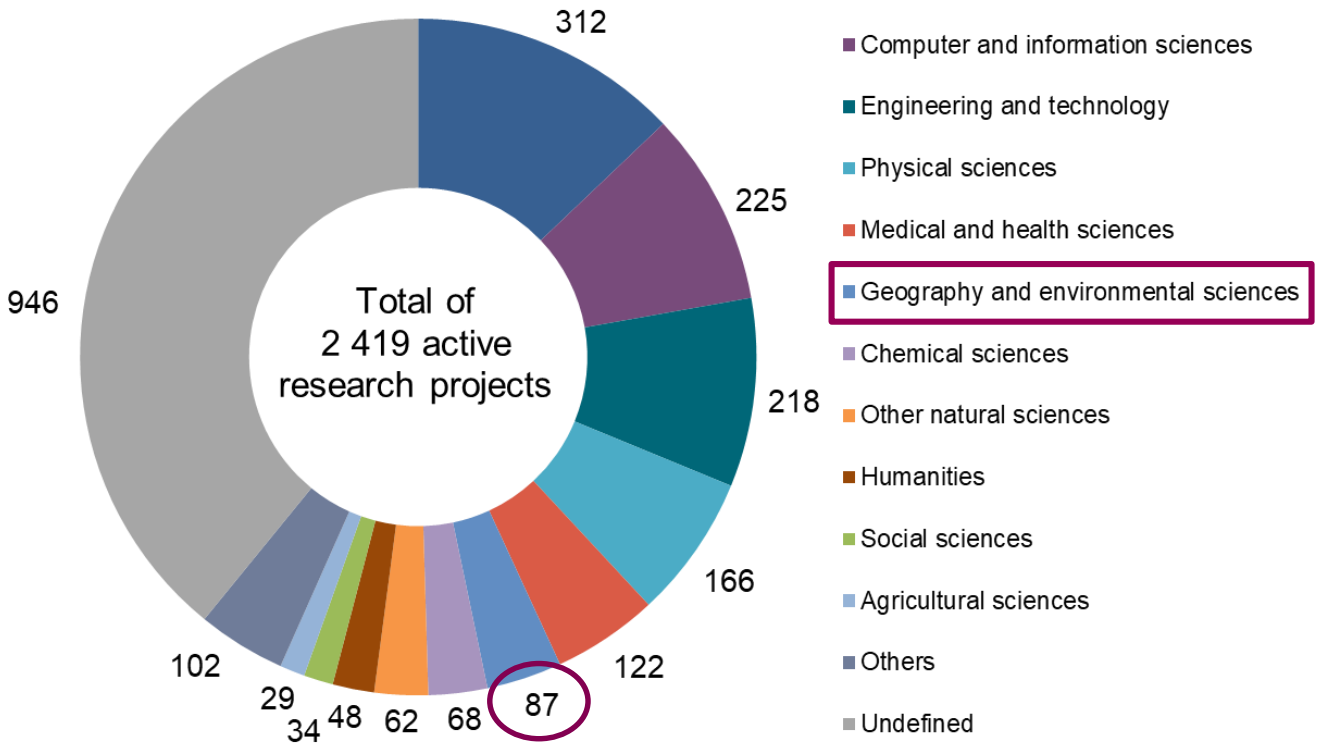
Circa

506

employees in 2021

You are not alone!

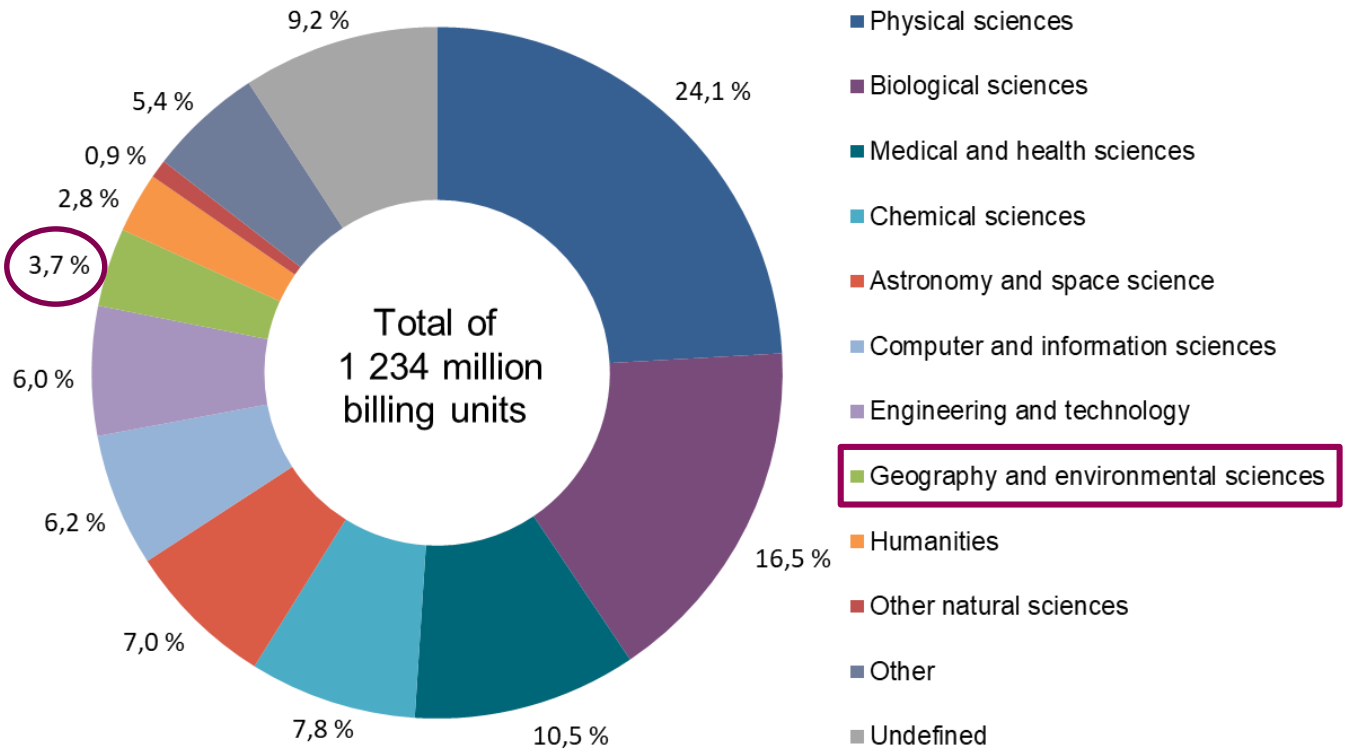
Active research projects by science area in 2020 (all servers)



An active project contains billing unit usage during year 2020.

You are not alone!

Computing resource usage by science area in 2020
(incl. all computational and storage use)

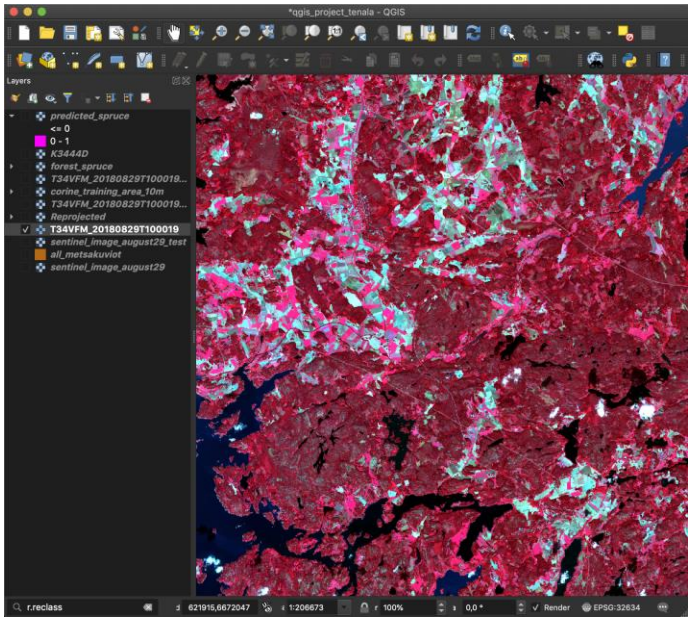


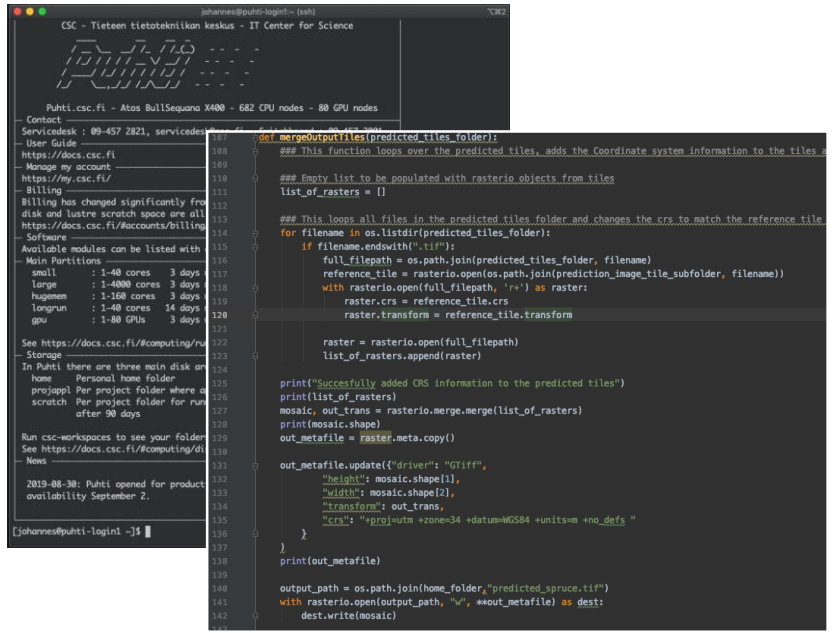
Why using CSC resources?

- Long computing time (> 2 hours)
- High memory usage (> 8 GB)
- Large datasets (> 50 GB)
- Outsource computations, keep own computer free
- Server needs (cPouta)
- Course computers (same setup) (Notebooks)
- Software readily available (Puhti)
- Computing support
- it's free! *

* for open science Finnish university and state research institute users

Requirement for efficient geocomputing





Graphical User Interfaces

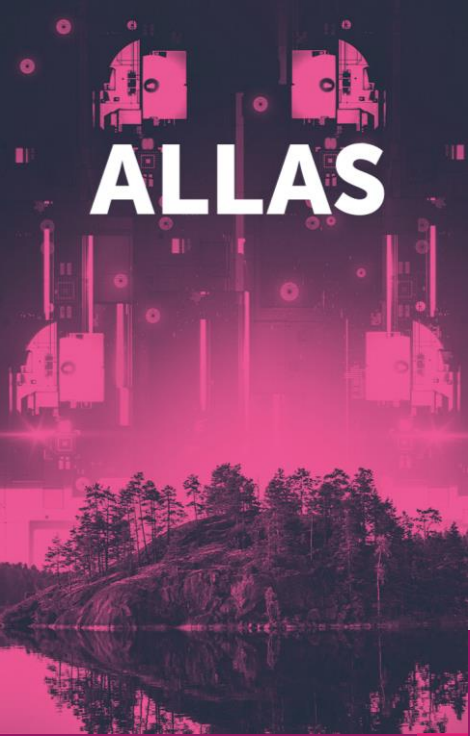
Command Line Interface, scripts

Computing resources for you*

	Puhti	cPouta cloud
System	Supercomputer	Virtual machine cloud
Software	Pre-installed software + user-installed software	User-installed software
Data	Main Finnish datasets	-
Use cases	Run demanding analyses with numerous CPUs or GPUs	Setup your own virtual machine and environment
Max per job / VM / container	**4000** CPUs / **80** GPUs **1500**GB memory	**48** CPUs / **4** GPUs **240**GB memory

Average computer:
4 CPUs /
8 GB memory

* as user of geospatial software



What about data?

- Storage capacity in Puhti/cPouta is limited

→ **Allas object storage**

  - data stored during project runtime
  - Access from
    - CSC computing environments
    - personal computer
  - data is immutable
  - publishable via URL

→ Allas and geospatial data webinar:
https://www.youtube.com/watch?v=mnFXe2-dJ_g

How to get started?

- **Check your eligibility** for using CSC resources: https://research.csc.fi/free-of-charge-use-cases
- Get a **user account**: https://docs.csc.fi/accounts/how-to-create-new-user-account/
- Create or let your PI **create a new project:** https://docs.csc.fi/accounts/how-to-create-new-project/
- **Add members** to your project: https://docs.csc.fi/accounts/how-to-add-members-to-project/
- **Add services** to your project: https://docs.csc.fi/accounts/how-to-add-service-access-for-project/



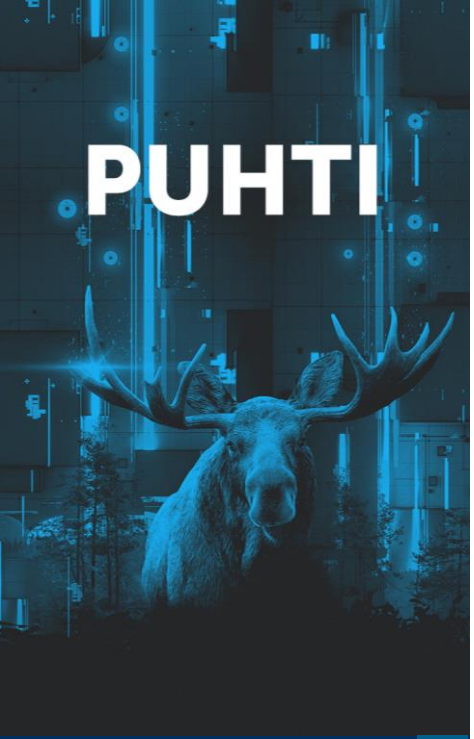
Billing Units (BU)

- Each project has a certain amount of so-called Billing Units (BU)

- Using CSC resources consumes Bus, based on:
  - Computation time and number of resources (CPU, memory)
  - GPUs in Puhti (**a lot of BUs!**)
  - Increased storage quota in Puhti
  - Used storage in Allas
  - Cpouta uptime (flavor dependent)

You can apply for more BUs in **my.csc.fi** by providing a description what what you are doing

Puhti - one of CSC's supercomputers



Keep it real

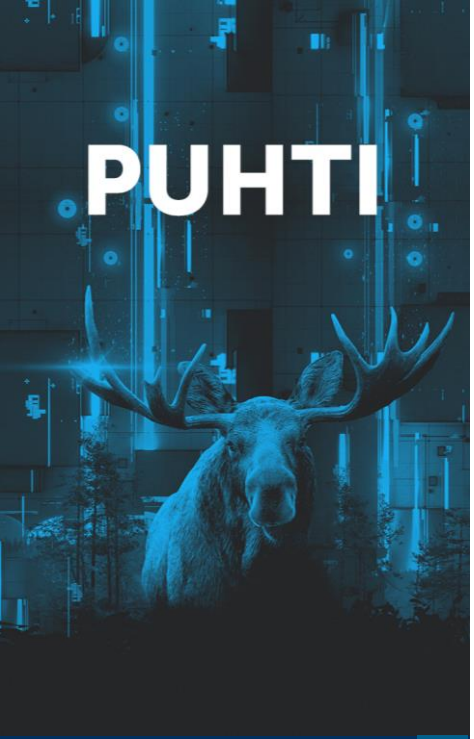
Single core of Puhti = ~speed of laptop

But: Puhti has **many cores**

and **more memory** and **faster input-output (I/O)**

→ Running single core scripts on Puhti does not make it faster

Speedup: multi-core parallel processes and script optimization



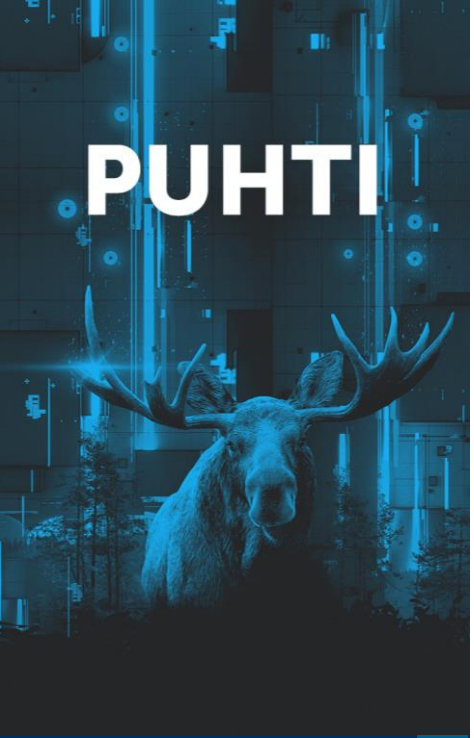
Geospatial Software in Puhti

geo portti
Finnish Geospatial Research and Education Hub

- FORCE & SPLITS
- **GDAL**
- LasTools , also .exe tools with Wine
- MatLab / Octave
- Mapnik
- **OpenDroneMap**
- Orfeo Toolbox
- PCL
- PDAL
- **Python geospatial packages**
- QGIS

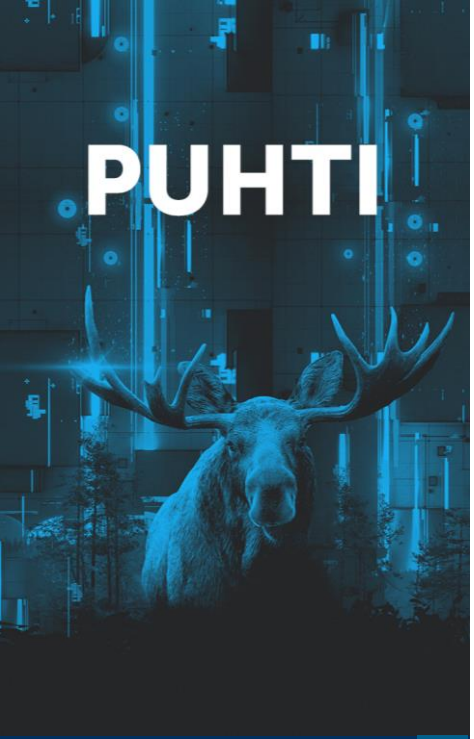
- **R geospatial packages**
- SagaGIS
- **SNAP,** Sen2cor
- WhiteboxTools
- Zonation

- **Something missing?**
  - Ask us :)
    servicedesk@csc.fi



Geospatial software NOT available in Puhti

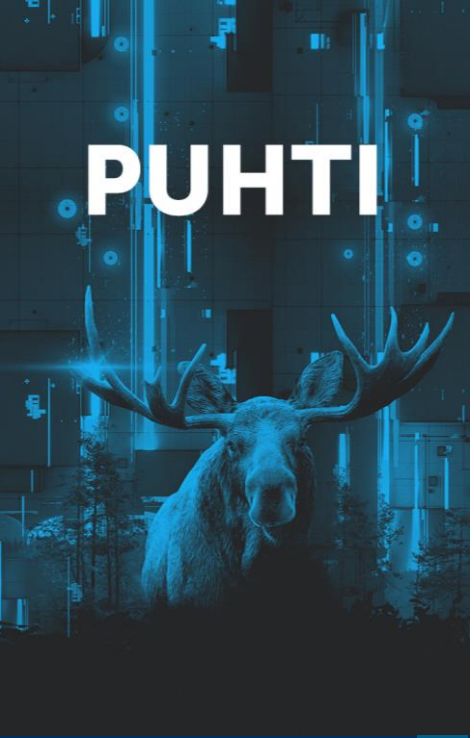
- **Windows** software
  - ArcGIS, TerraScan
- **Map servers**
  - GeoServer, MapServer
- **Database**
  - PostGIS, MongoDB
- **Web map libraries**
  - OpenLayers, Leaflet



Parallelization, GIS software

Some support for parallelization

- GRASS
- FORCE
- OpenDroneMap
- SagaGIS
- SNAP
- R: raster, terra, lidR
- ArcGIS Pro (not in Puhti)



Make code parallel yourself

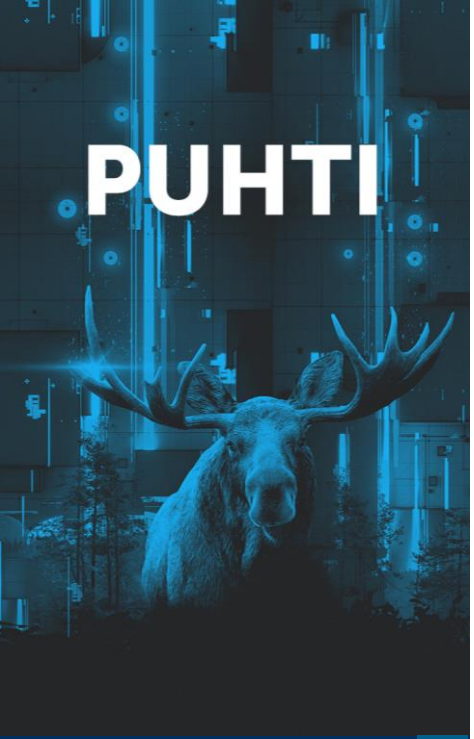
Python libraries

- dask
- multiprocessing
- joblib

R libraries

- future
- (snow, foreach, Rmpi, …)

Matlab Parallel Computing Toolbox



Available geospatial data

geo portti | Finnish Geospatial Research and Education Hub

/appl/data/geo

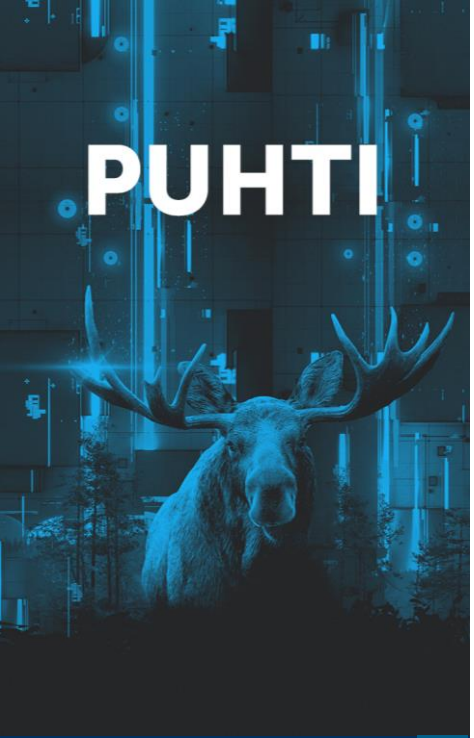
→ large commonly used datasets (open license)
→ no transfer needed
→ all Puhti users have **read** access

11 TB of data available

https://docs.csc.fi/data/datasets/spatial-data-in-csc-computing-env/

Something missing?
→ contact us at servicedesk@csc.fi

- Finnish Digital and Population Data Services Agency
- Finnish Food Agency
- Finnish Meteorological Institute (FMI)
- Finnish Transport Infrastructure Agency, Digiroad
- Institute for the Languages of Finland (KOTUS)
- Karelia University of Applied Sciences
- Latuviitta
- National Land Survey (MML)
- Natural resource institute Finland (LUKE)
- Statistics Finland
- University of Helsinki, Digital Geography Lab

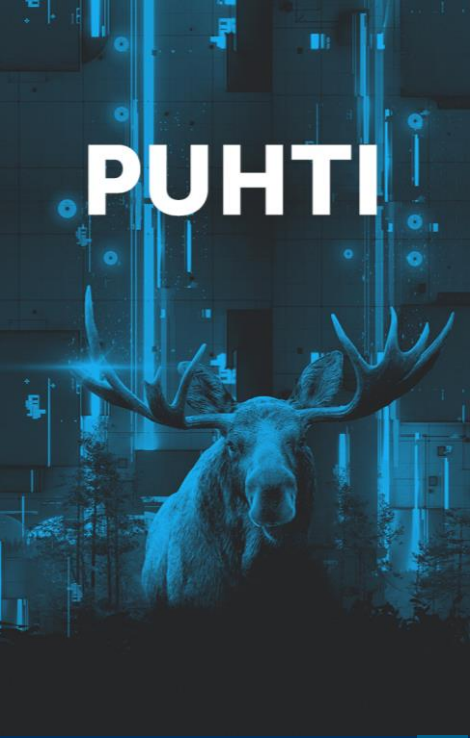


Example scripts

https://github.com/csc/training/geocomputing

- **Batch job files** (requesting resources on computing nodes)
- **R** and **Python**
  - Parallelization libraries
  - Array / Parallel jobs
- **Allas** data transfers with **R** and **Python**
- Copernicus imagery download (Python)
- **SNAP** array job
- Create and use **virtual rasters**

geo portti | Finnish Geospatial Research and Education Hub

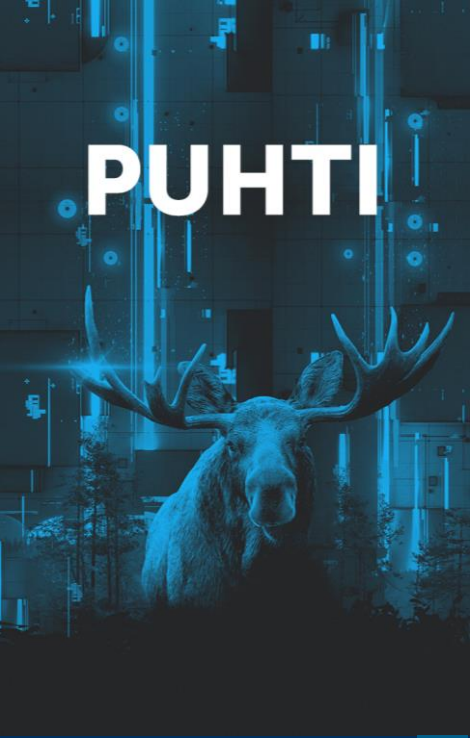


How can I edit/check my data?

https://puhti.csc.fi

→ Puhti web interface for Graphical User Interfaces

- QGIS
- SNAP
- GRASS GIS
- SAGA GIS
- RStudio
- Jupyter
- Visual Studio Code
- Spyder



Steps for running your R script in Puhti

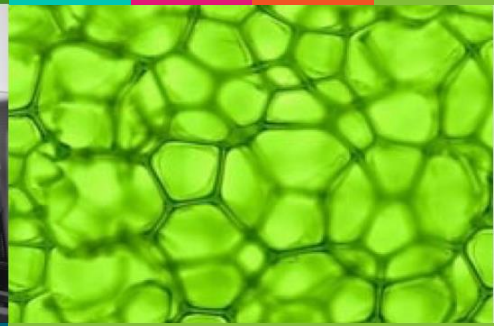
(0. Get **CSC user account**)

1. Log in to **Puhti web interface** (www.puhti.csc.fi).
2. **Move your data and scripts** to Puhti (consider github!).
3. **Open RStudio**.
4. Check **R package availability** (https://docs.csc.fi/apps/r-env-for-gis/)

   * If needed, install it yourself or ask CSC - servicedesk@csc.fi
5. **Fix paths** of your input/output files.
6. **Test your script** with some test data.
7. Write a **batch job script**.
8. **Run** your scripts with all data **as batch job** (or interactively)

(9. **Make use of several cores** using future package in your R code.)

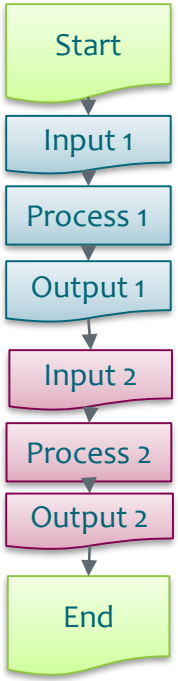
Running R in parallel — principles and practice



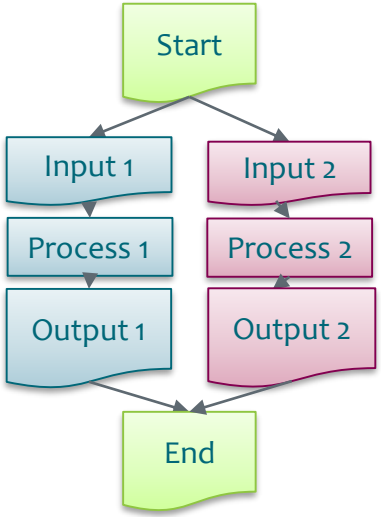
"My R code is slow… what can be done?"

- Try to understand which part of the code takes time and why
  - Use **`system.time()`** or **`tictoc`** package
- Different R packages might have different speed
  - Prefer **sf** over sp and **terra** over raster.
- Always be suspicious of **`for`**-loops!
- Going parallel *may* help
- Unfortunately, increasing the number of cores will *never* decrease the time needed in the same proportion

Serial code



Parallel code

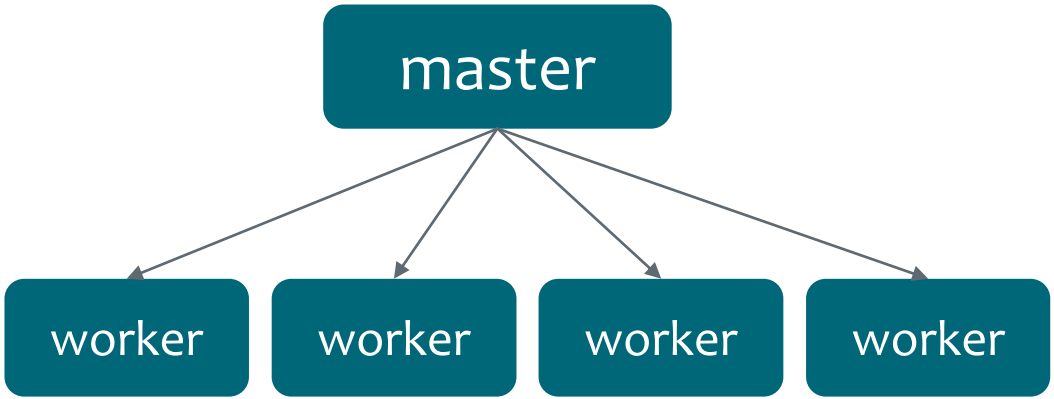


Parallelization in GIS

- Easiest parts for parallelization:
  - for loops
  - map
  - *apply
- Logical level
  - input files / map sheets / chunks of vector data,
  - scenarios, variables, time periods
- In many cases if using map sheets the borders need special care.

future

- A modern, easy-to-use parallel computing package in R
- Older packages for similar use: snow, foreach, Rmpi, pbdMPI, …



future, parallelization options

Name	OS	Description
synchronous:		*non-parallel:*
sequential	all	sequentially and in the current R process
asynchronous:		*parallel*:
multisession	all	background R sessions (on current machine)
multicore	**not Windows/ not RStudio**	forked R processes (on current machine)
cluster	all	external R sessions on current, local, and/or remote machines

function and input

```
slow_function <- function() {
  Sys.sleep(1)
  resutn(i)
}
input <- 1:7
```

for loop

```
a <- 0
for(i in input) {
  a[i] <- slow_function(i)
}
```

purrr, map (tidyverse)

```
library(purrr)
a <- map(input, slow_function)
```

lapply

```
a <- lapply(input, slow_function)
```

lapply

```
a <- lapply(1:7, slow_function)
```

future.apply

```
library(future.apply)
plan(multiprocess)
a <- future_lapply(1:7, slow_function)
```

purrr, map (tidyverse)

```
library(purrr)
a<- map(i:7, slow_function(i))
```

furrr, map

```
library(furrr)
plan(multiprocess)
a<- future_map(1:7, slow_function)
```


Full example: https://github.com/csc-training/r-spatial-course/blob/main/R_parallellization_demo.R

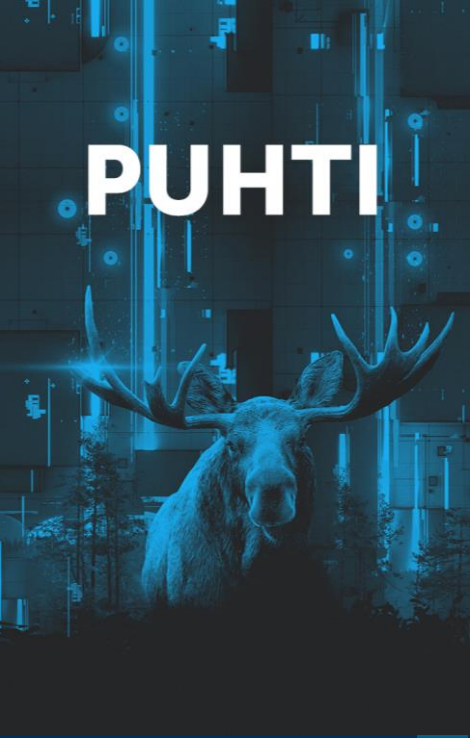
future, libraries and variables

- future exports needed variables and libraries automatically to the workers
- the variables must be serializable
- Terra: https://github.com/rspatial/terra/issues/36
  - Give file names to workers
  - Use wrap() and rast()/vect(), but that inludes moving data between wokrer and master, so be careful

Be careful with simultaneous opening of files

- If you are using the same input or output files in parallel jobs, depending on software it might cause trouble.
- Solutions:
  - Make copies of input files for each processes you use.
  - Write output to smaller files first, join them later

Working in Puhti

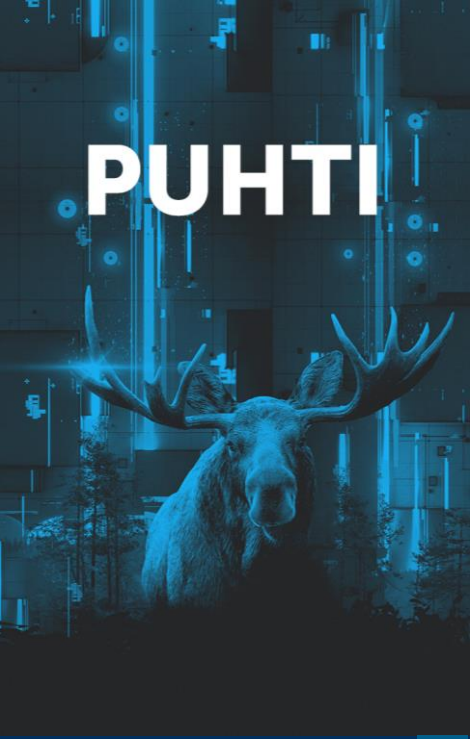


Getting started with Puhti

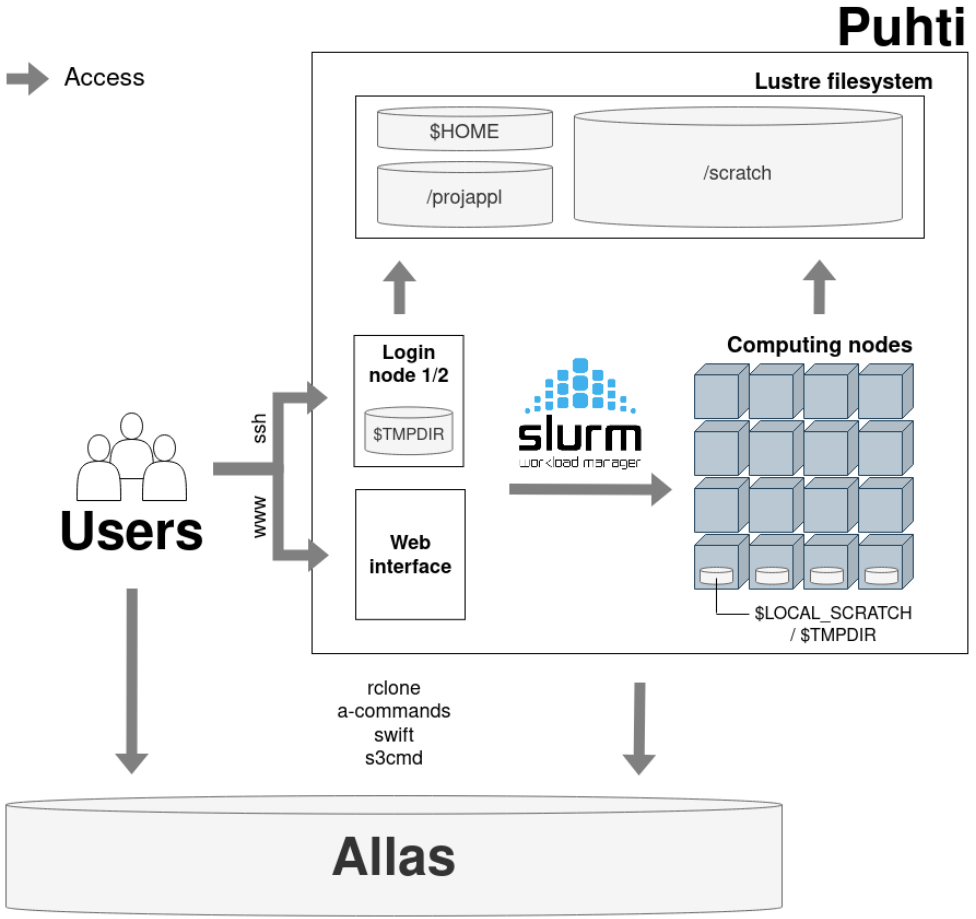
1. **Apply** for an account, project, resources and Puhti access
https://research.csc.fi/accounts-and-projects
https://research.csc.fi/free-of-charge-use-cases

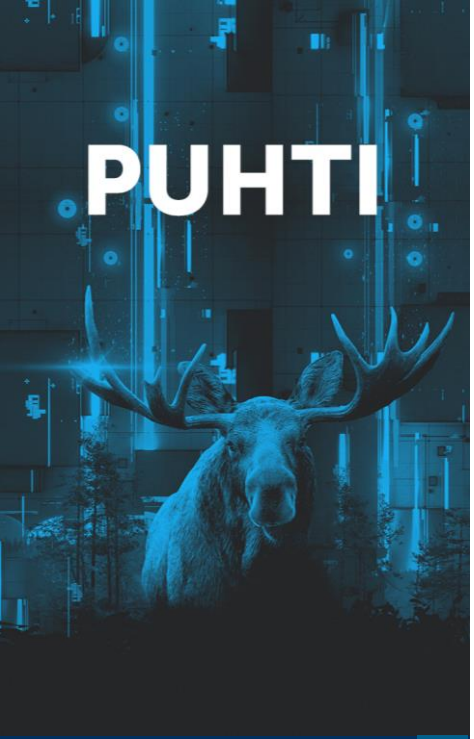
2. **Read/watch**

- **Connecting** to Puhti: https://docs.csc.fi/computing/connecting/
- **Working directories**: https://docs.csc.fi/computing/disk/
- Loading software with **modules:** https://docs.csc.fi/computing/modules/
- **Batch job system** for submitting jobs: https://docs.csc.fi/computing/running/creating-job-scripts-puhti/
- **GIS software** specific pages: https://docs.csc.fi/apps/#geosciences
- CSC **Linux tutorial** for basic Linux commands: https://docs.csc.fi/support/tutorials/env-guide/overview/
- **Geocomputing** webinar: https://www.youtube.com/watch?v=PrgMFna3DKw


PUHTI

Puhti


Access
Lustre filesystem
$HOME
/projappl
/scratch
Login node 1/2
$TMPDIR
Web interface
Computing nodes
slurm
workload manager
ssh
www
Users
$LOCAL_SCRATCH / $TMPDIR
rclone
a-commands
swift
s3cmd
Allas



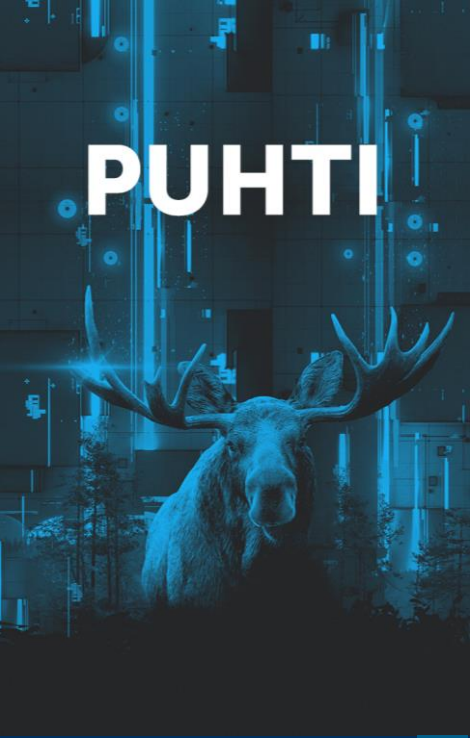
Set up tools for working with Puhti

Minimum:

web-browser for **Puhti web interface**

Advanced:

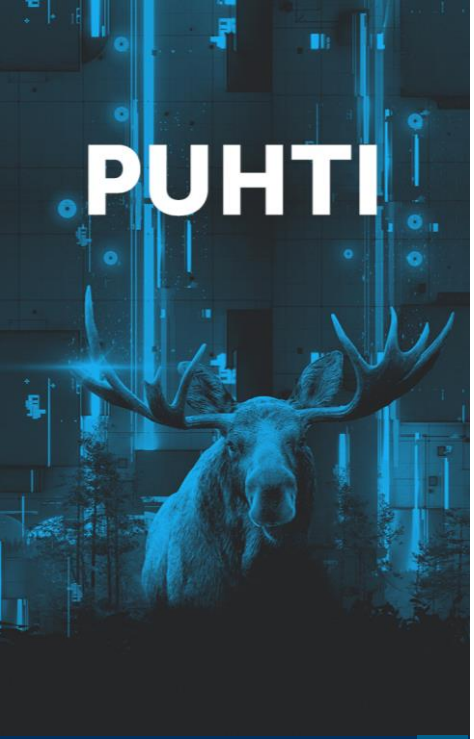
- **Connecting:**
  - Terminal in Linux/Mac
  - PowerShell/Putty/MobaXterm in Windows
- **Moving data:**
  - FileZilla/WinSCP, rsync, wget, curl.



Directories

Directory	Owner	Capacity	Num. files	Path	Cleaning
home ($HOME)	Personal	10 GiB	100 000	/users/<user-name>	No
projappl	Project	50 GiB	100 000	/projappl/<project>	No
scratch	Project	1 TiB	1 000 000	/scratch/<project>	Yes **– 90 days**

- The quota of ***scratch*** and ***projappl*** directories can be increased
- **No back-up**
- More information: https://docs.csc.fi/computing/disk/



The module system

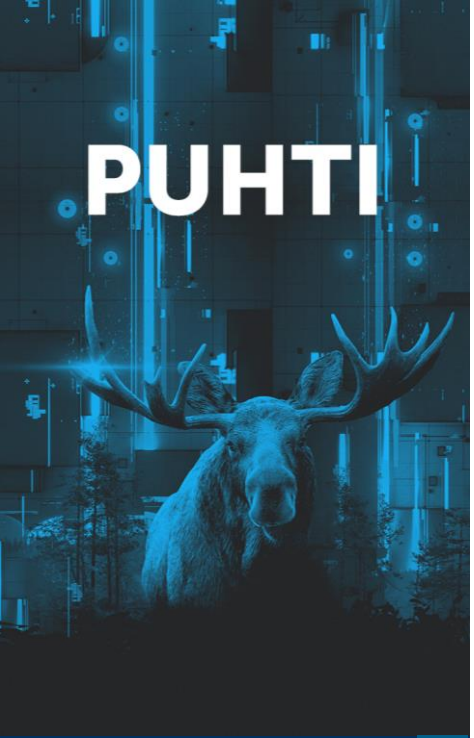
- Puhti is a shared computing environment with hundreds of users

- Software is loaded with **modules**
  - Mutually incompatible software
  - One module: single program or group of similar programs
  - Modules load applications, adjust path settings and set environment variables

- **Example.** Loading module for geospatial Python tools

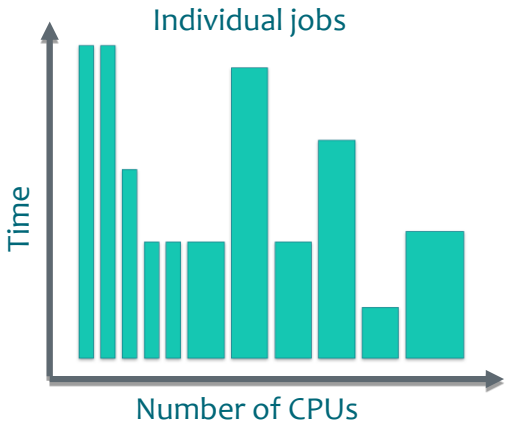
```
module load geoconda
```

- Check: https://docs.csc.fi/apps/#geosciences for module names



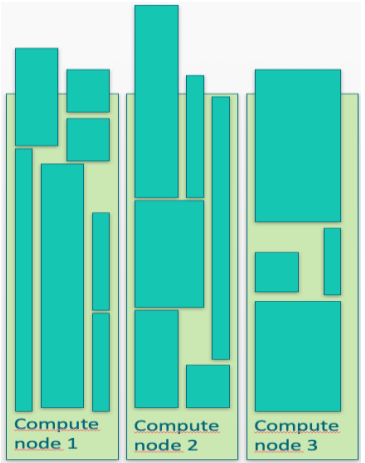
The batch job system (SLURM)

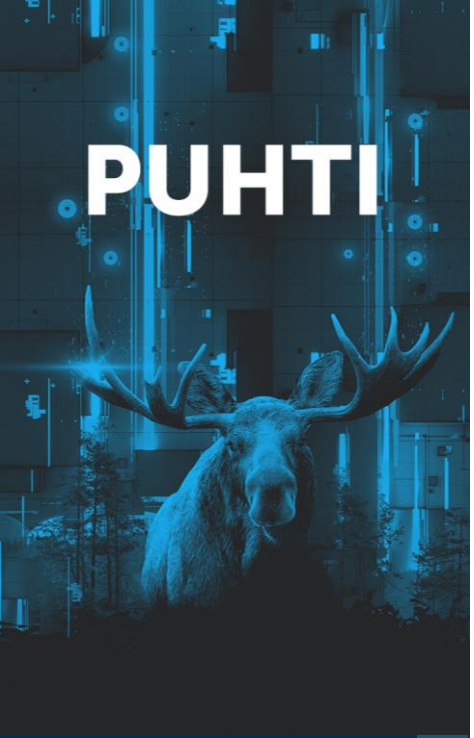
- Running jobs in Puhti requires you to use the batch job system
- You request resources for a batch job
  - CPU cores, memory, GPU etc.
  - Time
- Batch job optimization of the queue by SLURM:



SLURM places jobs on computing nodes

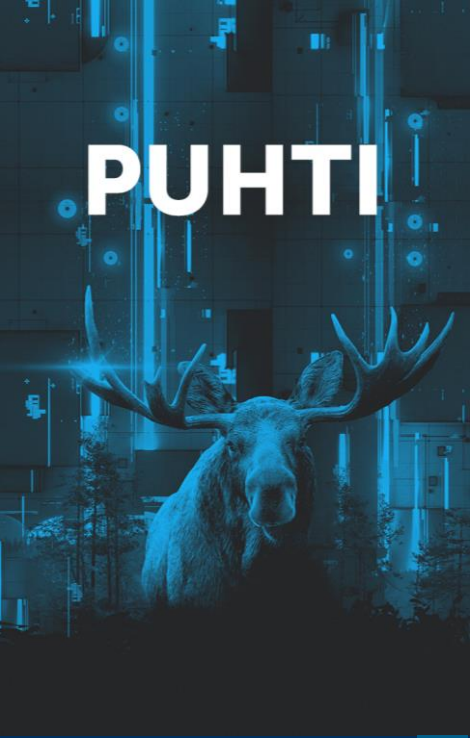
In the most efficient way resource wise





Batch job partitions

Partition	Time limit	Max tasks	Max memory
test	15 minutes	80	190 GiB
interactive	7 days	8	64 GiB
small	3 days	40	382 GiB
large	3 days	1040	382 Gib
longrun	**14 days**	40	382 GiB
hugemem	3 days	160	1534 GiB
hugemem_longrun	7 days	40	**1534 GiB**



Batch job scripts

- For requesting resources and submitting job description
- = bash scripts (.sh)

```
#!/bin/bash
#SBATCH --job-name=myTest
#SBATCH --account=<project>
#SBATCH --time=02:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2000
#SBATCH --partition=small

module load geoconda

srun python my_python_script.py
```

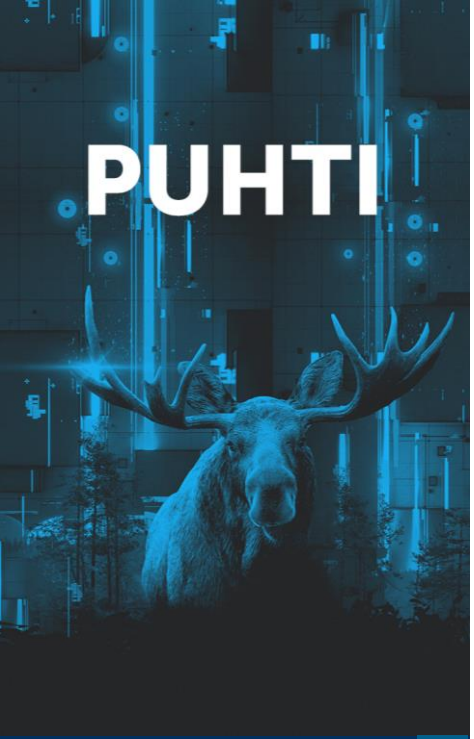
Submit batch job
sbatch <your-batch-job-script>

Cancel a job with
scancel <your-job-id>

Check if job has started running
squeue –u <your-user-name>

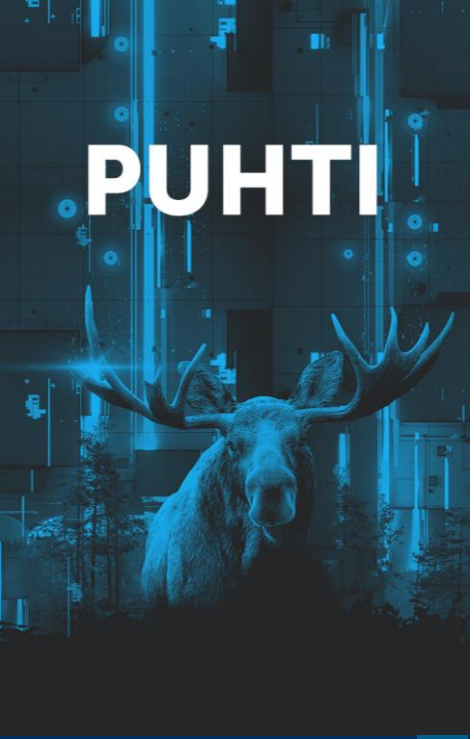
After the job, see resource usage
seff <your-job-id>

R spatial in Puhti



R spatial in Puhti, r-env-singularity

- r-env-singularity is a huge R installation with 1100+ R packages.
- Includes also several spatial analysis R packages.
- Has also GDAL and SagaGIS
- https://docs.csc.fi/apps/r-env-for-gis/
- https://docs.csc.fi/apps/r-env-singularity/
- https://github.com/csc-training/geocomputing/tree/master/R



Installing R libraries

- Everybody can add R libraries for personal use
  - These override the system ones
- If the package depends on some external library, the user installation is likely tricky.
- If you think that the package should be useful also for others or requires some external software to be installed, ask servicedesk@csc.fi
- https://docs.csc.fi/apps/r-env-singularity/#r-package-installations



How to get help?

If you are experiencing problems, have questions or wish to have additional software installed in Puhti, do not hesitate to contact

servicedesk@csc.fi

- More information on geocomputing at CSC

  https://research.csc.fi/geocomputing

- CSC services documentation

  https://docs.csc.fi/

- How to write good service requests:

  https://docs.csc.fi/support/support-howto/

Accounts



my.csc.fi

Self-service for creating and managing

- Accounts
- Projects
- Services
- Billing units
- Log in: HAKA, VIRTU, CSC

my.csc.fi demo