LIÈGE université
GIGA

Automated Machine Learning-based diagnosis of impaired consciousness: cross-center and protocol generalization of EEG biomarkers.

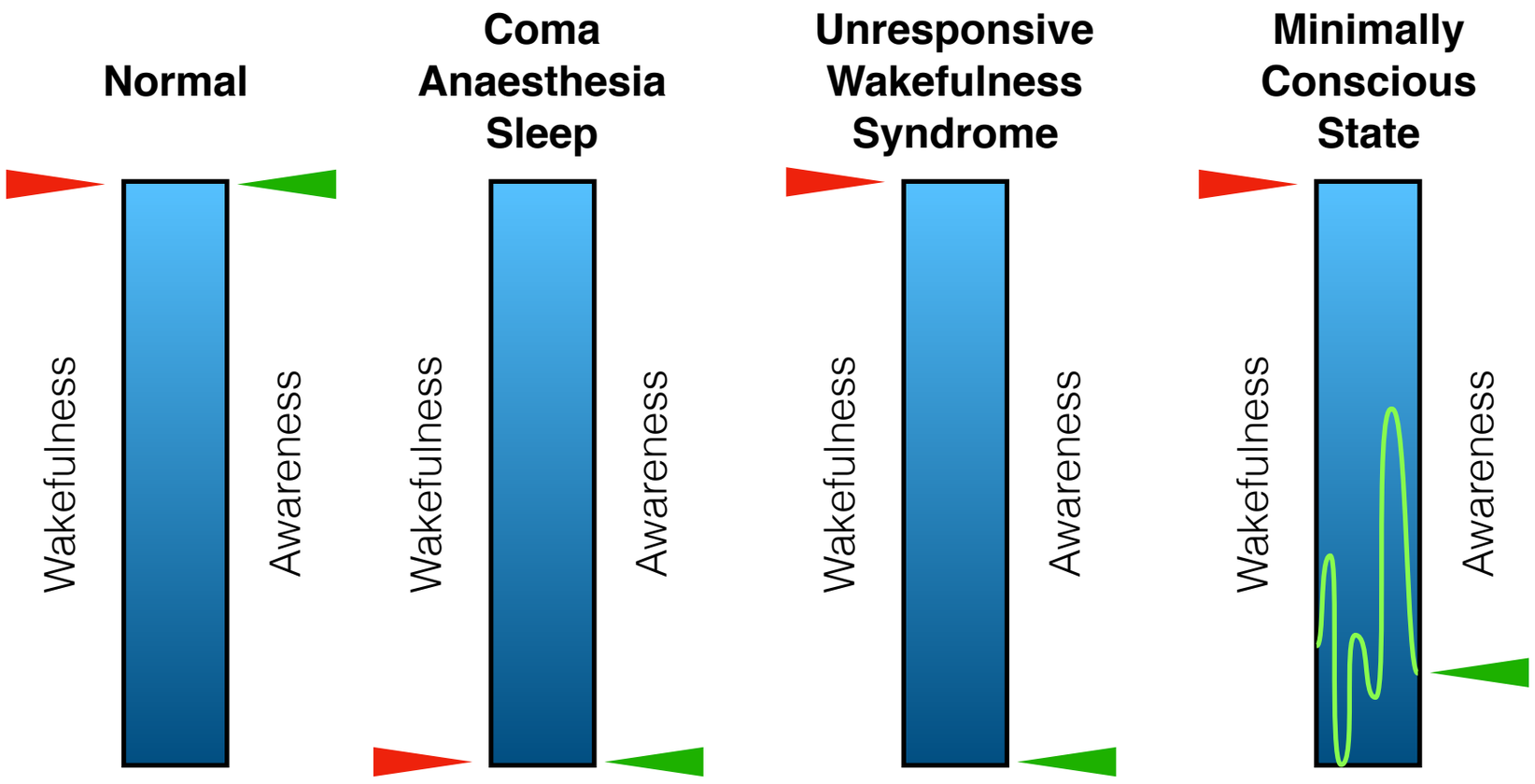
Federico Raimondo

fraimondo@uliege.be

@fraimondo

@RaimondoFede

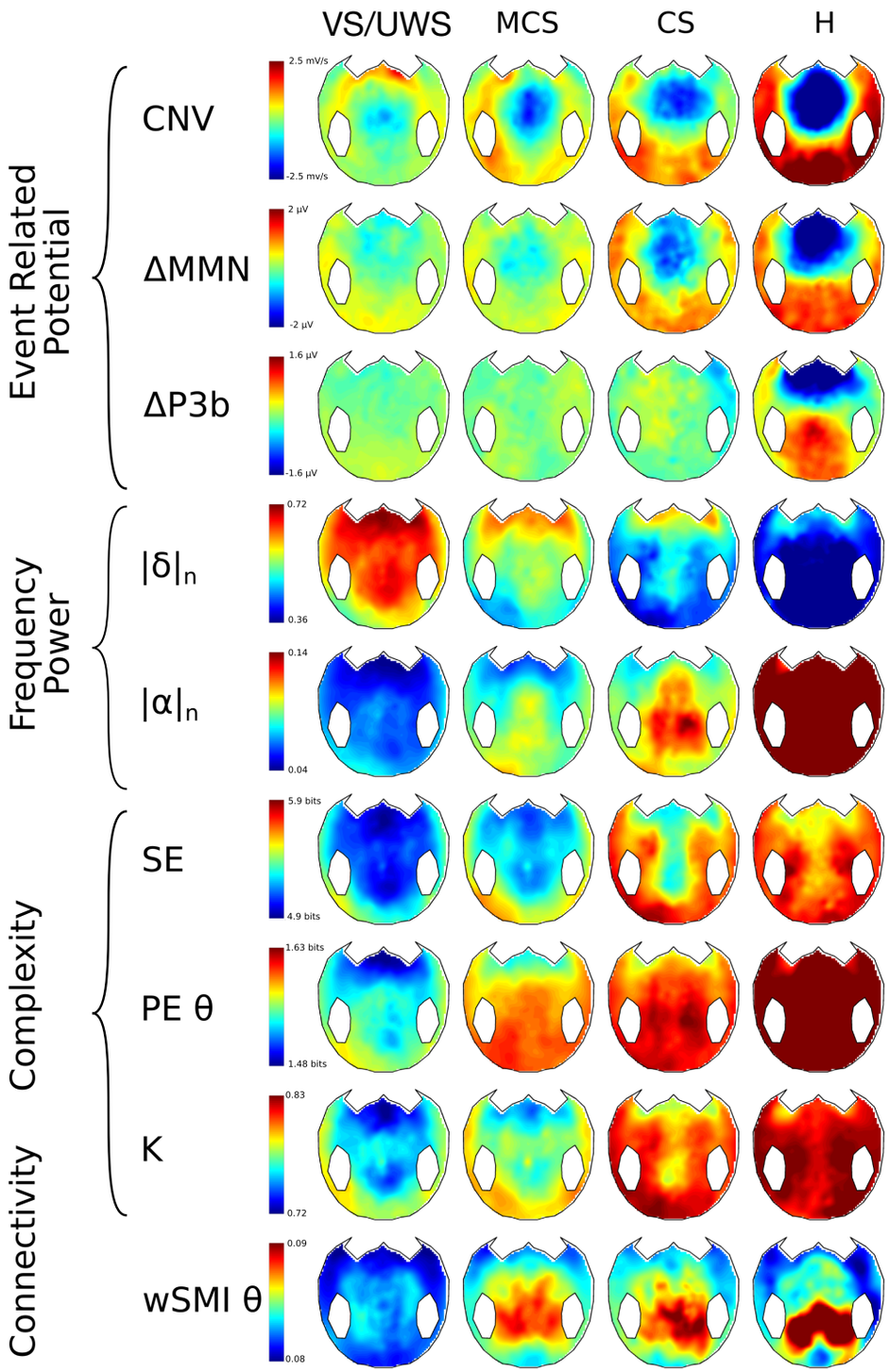
Disorders of Consciousness

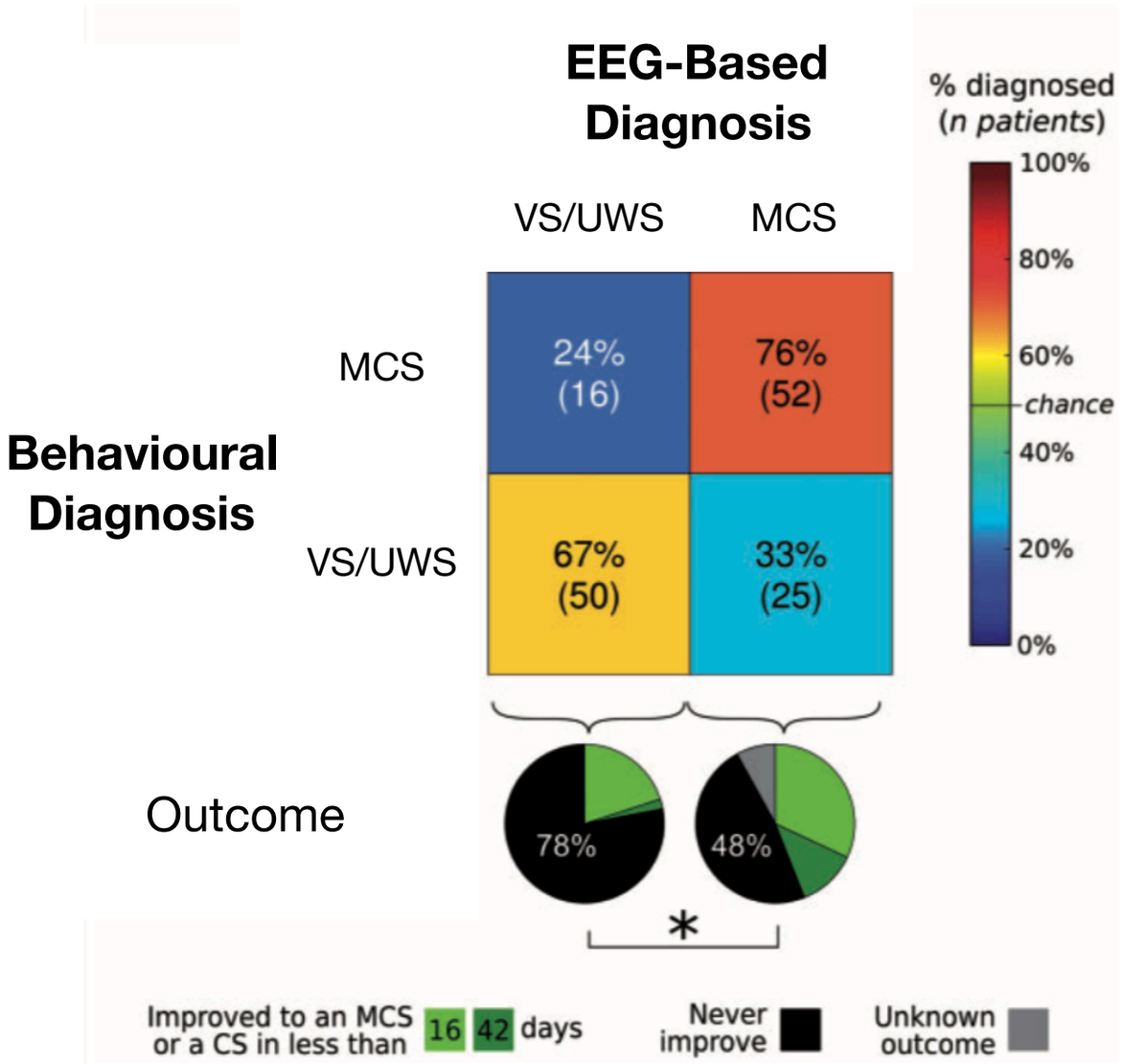


Laureys, Lancet Neurology, 2004

EEG-based diagnosis of DOC

Sitt*, King* et al.
Brain 2014





Does this tool generalise?

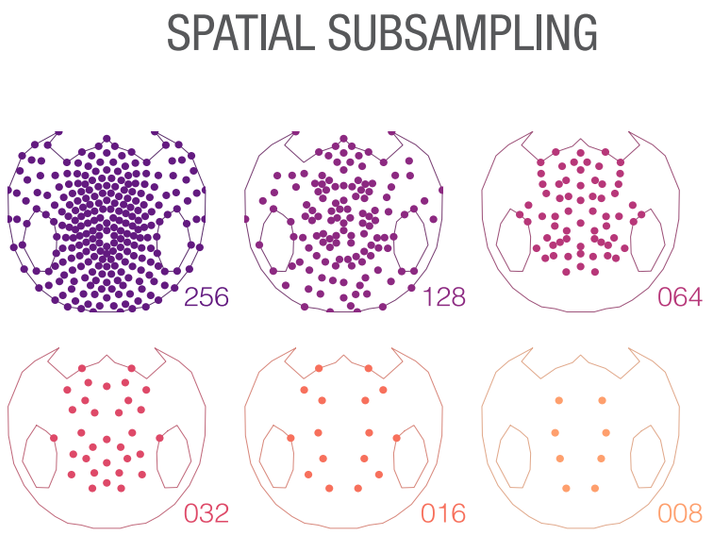
Aim: assess the possibility of creating a universal tool for EEG-based diagnosis of DOC

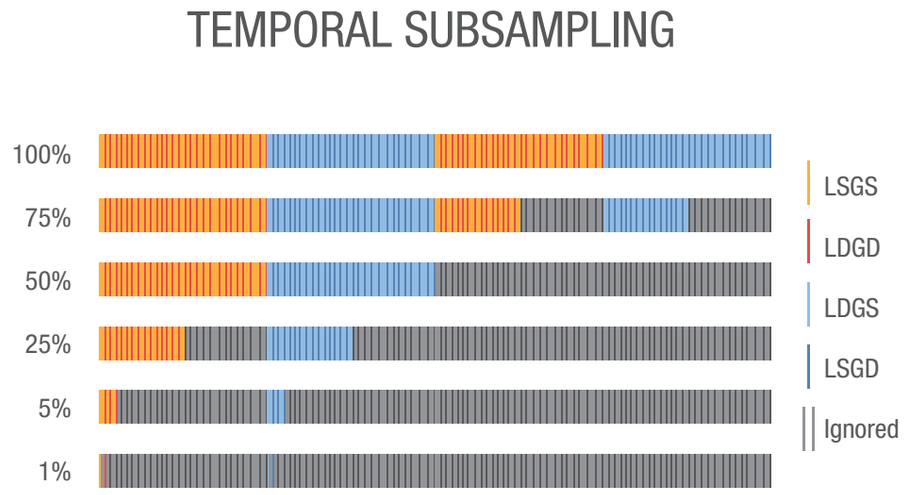
Methods:

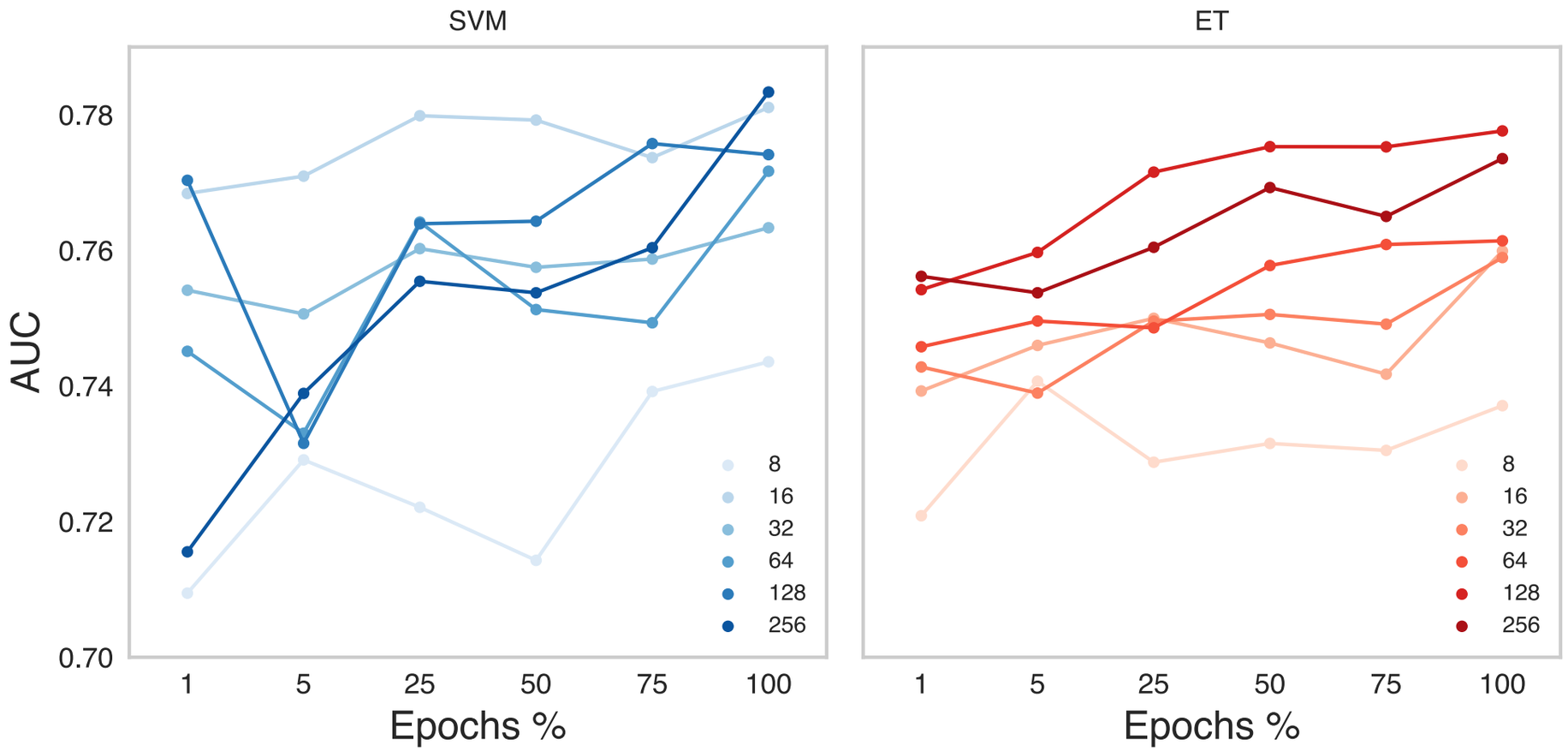
- Classifiers: SVM and Extra-Trees.
- Compare against a dummy classifier.
- Compare data spatially and temporally subsampled.
- Test on Resting State data (Liège).
- Bootstrap to estimate CI (2000 random samples).
- Measure AUC-ROC.

Dataset	Paris 1	Paris 2	Liège
N (EEG)	142	107	78
N (Patients)	98	92	78
N (UWS)	75	52	22
N (MCS)	66	56	57
Sex Ratio	2.0	1.9	1.2
Age (M [SD]) in years	46.5 [17.8]	45.4 [17.7]	38.0 [14.3]
Delay (M [SD]) in days	125.9 [372.9]	299.5 [823.6]	1040.5 [1227.6]
Delay (MD) in days	30.0	40.5	529
Delay (min to max)	6 to 2611	8 to 6570	11 to 5380
Anoxia (%)	29.6	30.4	N/A
Stroke (%)	29.6	15.2	N/A
TBI (%)	23.5	28.2	48.1
Other (%)	18.4	29.4	N/A

Generalisation: Alternative EEG configurations





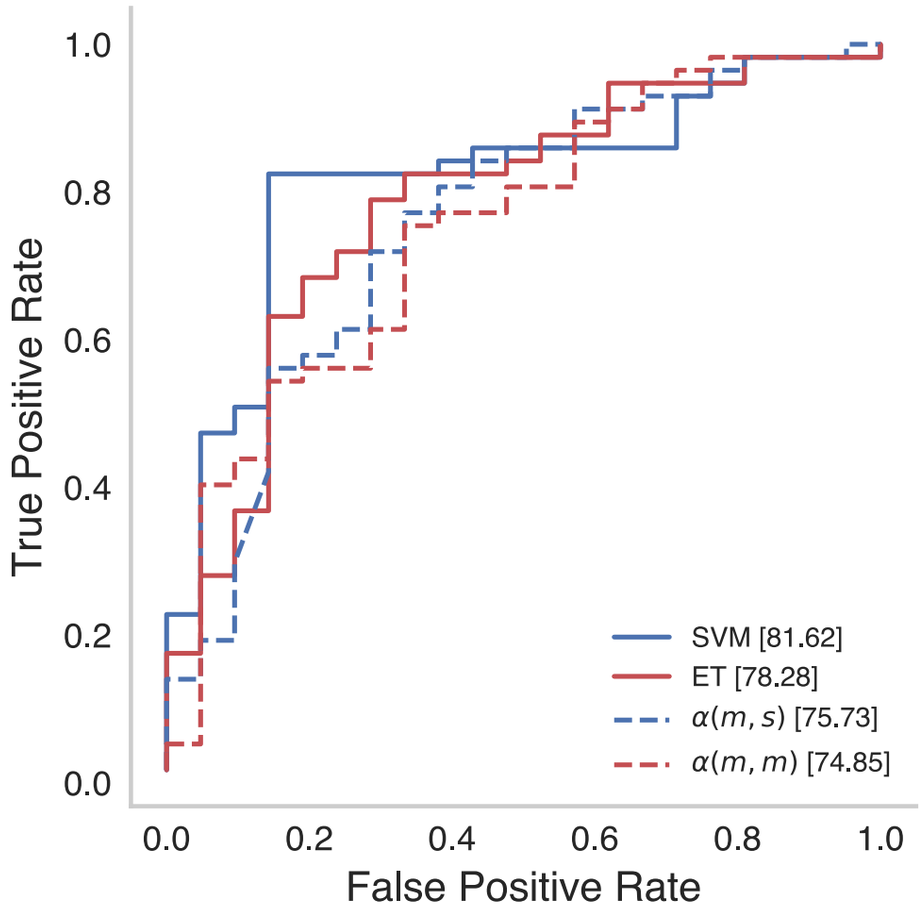


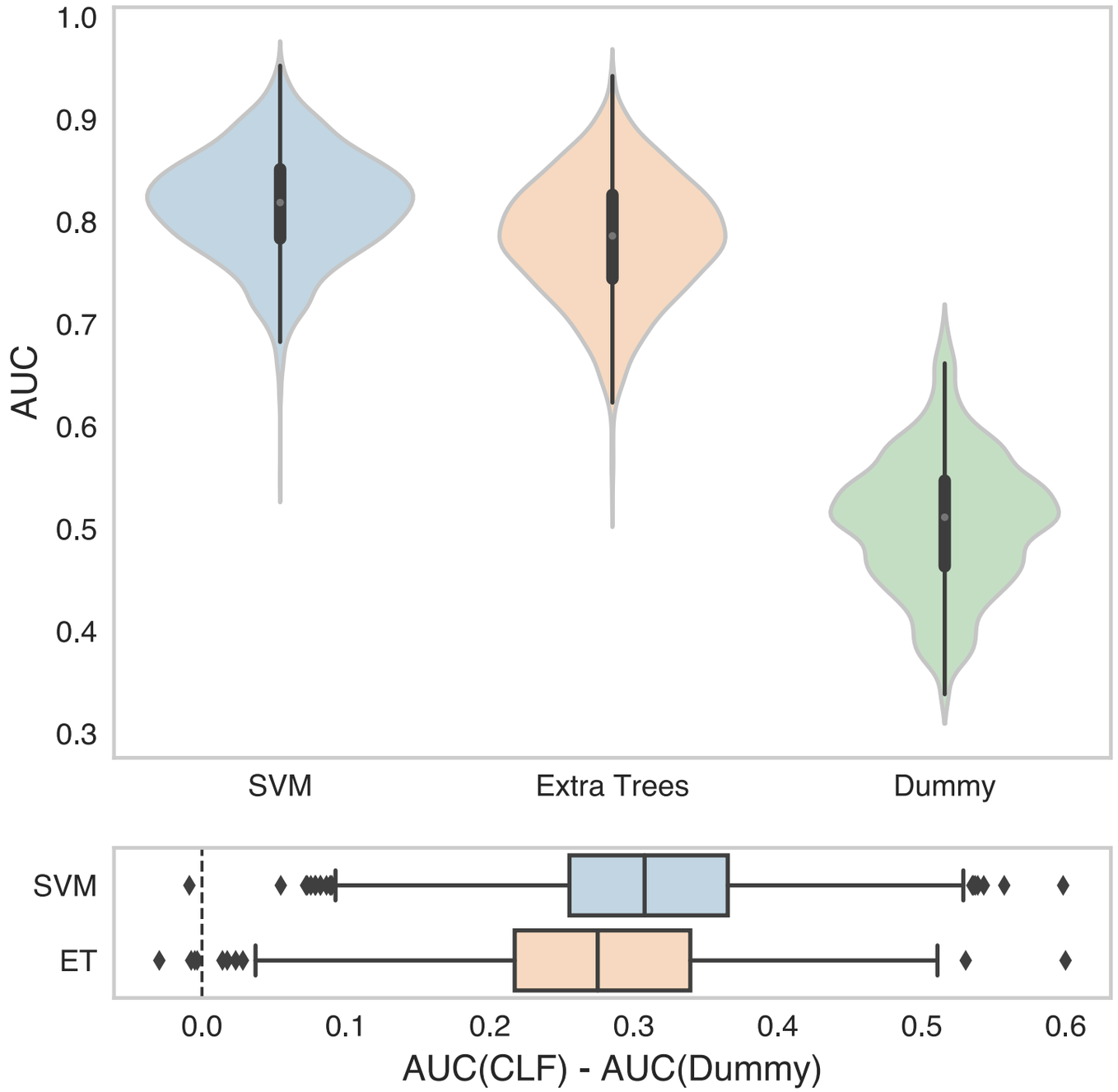
Engeman, Raimondo*, et al.* Brain, 2018

Generalisation: Resting State

Results:

- SVM: 81.62%
- Extra-Trees: 78.28%

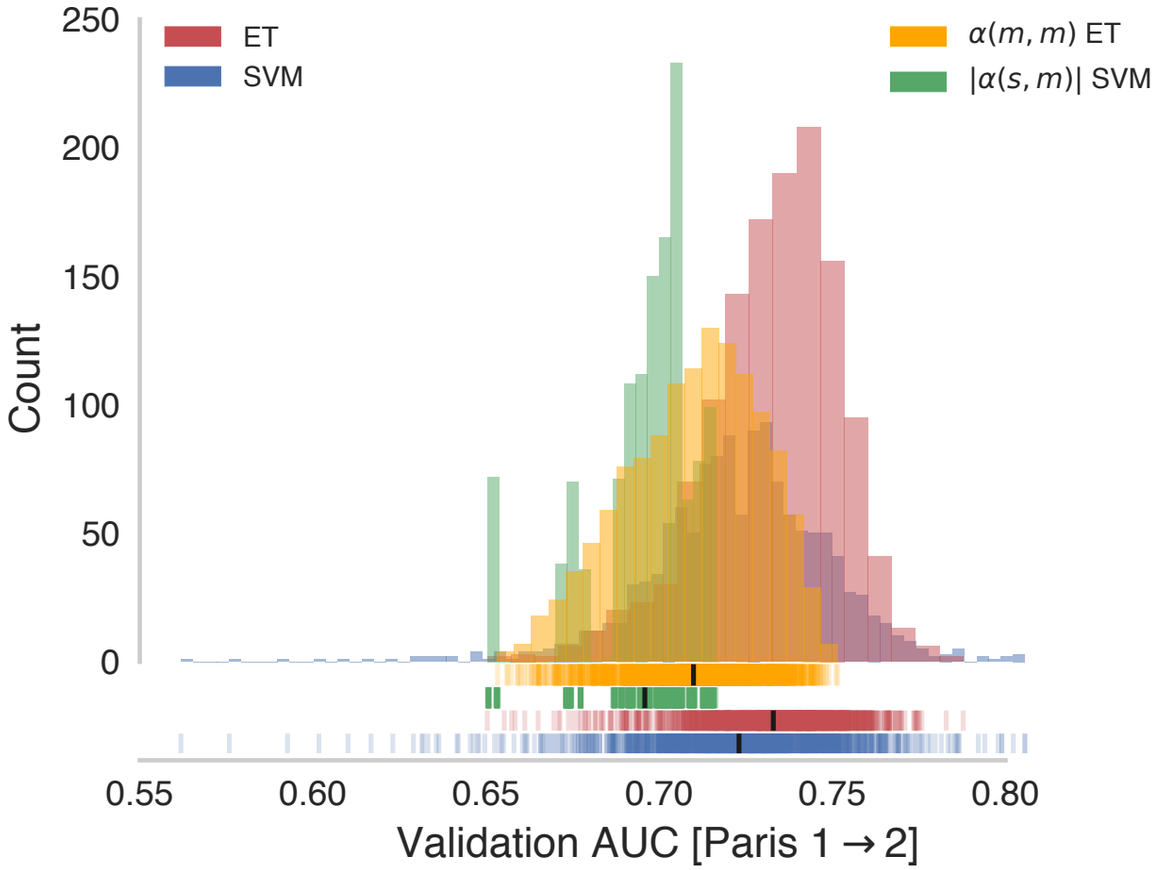


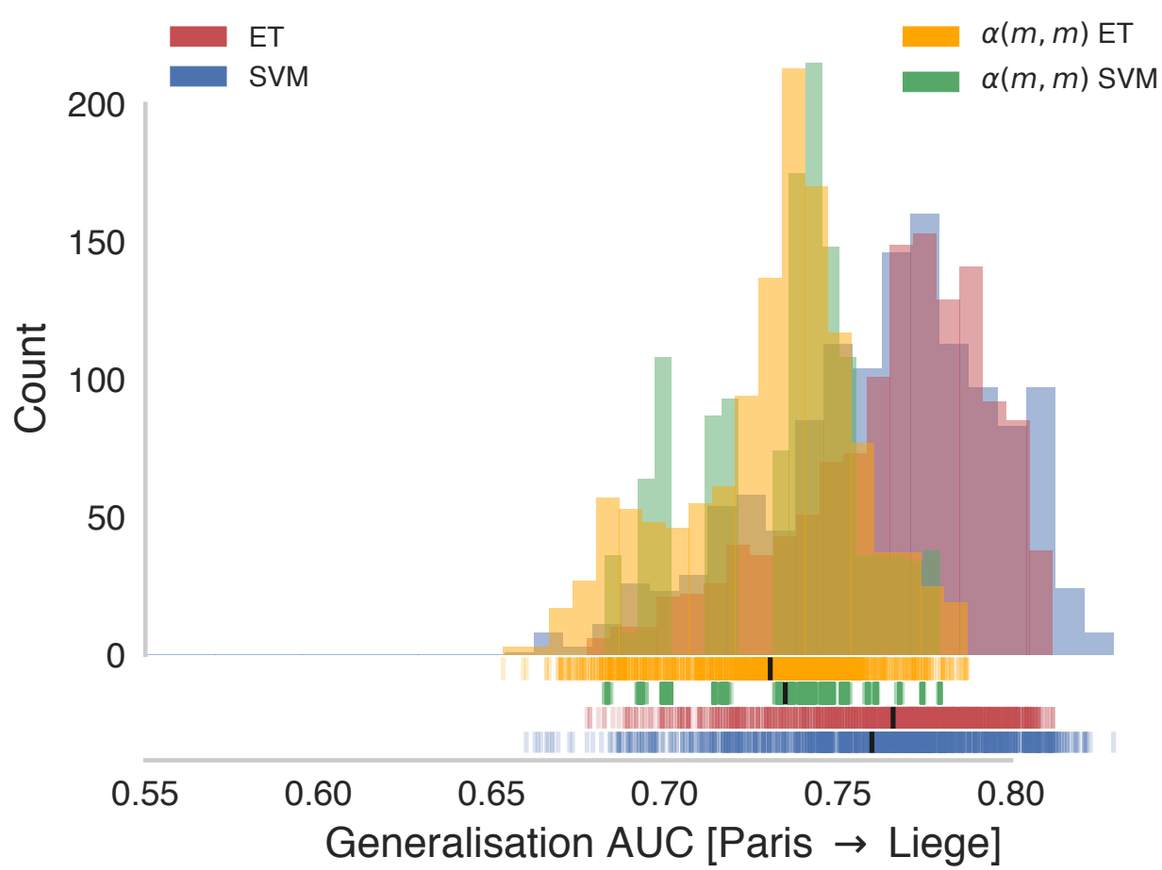


Engeman, Raimondo*, et al.* Brain, 2018

Generalisation: Mixed EEG configurations and acquisition protocols

All the different spatial and temporal configurations considered for training and testing





Engeman, Raimondo*, et al.* Brain, 2018

Conclusions

- Generalisation can be achieved despite changes in spatiotemporal configuration of the EEG and recording protocol.
- Multivariate overperforms univariate models.
    - Particularly with differences between training and testing sets.
- Choosing between ET and SVM classifier depends on the domain.
    - ET: model to Fit them All.
    - SVM: maximises AUC when data variability is low.
- Evoked potentials markers, while de-emphasized by the classifier, still contributes positively.
    - Validation Set: 72.91% with vs 71% without.
- While brain function during task and resting state differ, we could observe significant generalisation from task to resting state EEG.

Why is this impressive?

Filter

1 minute

Cut

0.5 minutes

Clean

2 minutes

Re-reference

0.1 minutes

Compute markers

25 minutes

Reduce Dimensions

5 minutes *

~ 32 minutes

x 500 = 16.000 minutes

= 266,6 hours

= 11,1 days

Combine Subjects data

Do Machine Learning

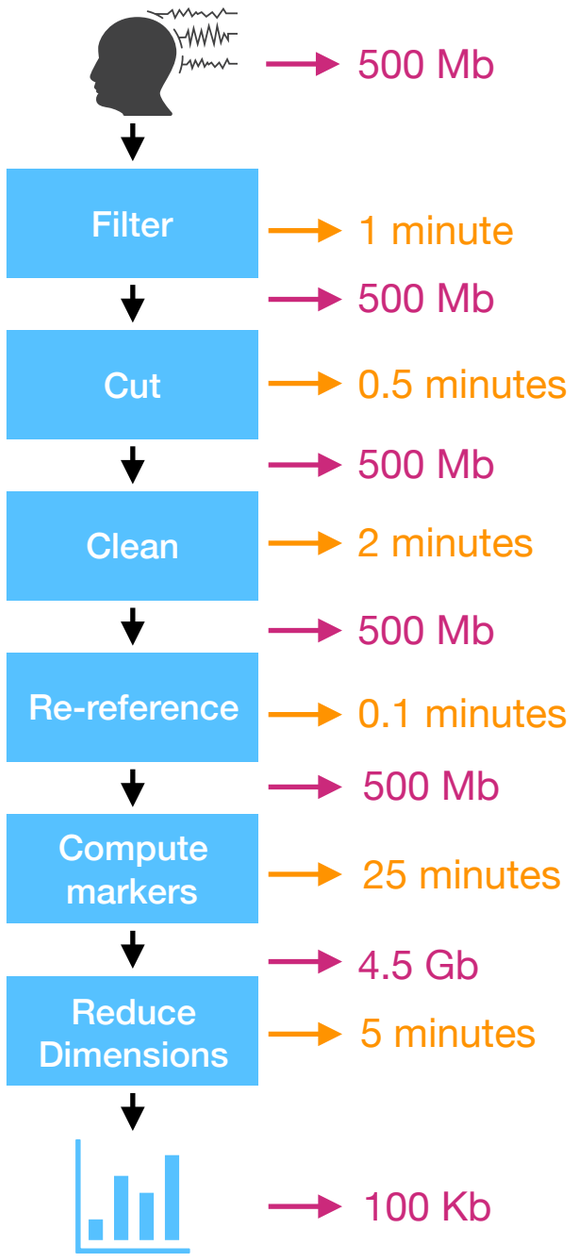
Processing

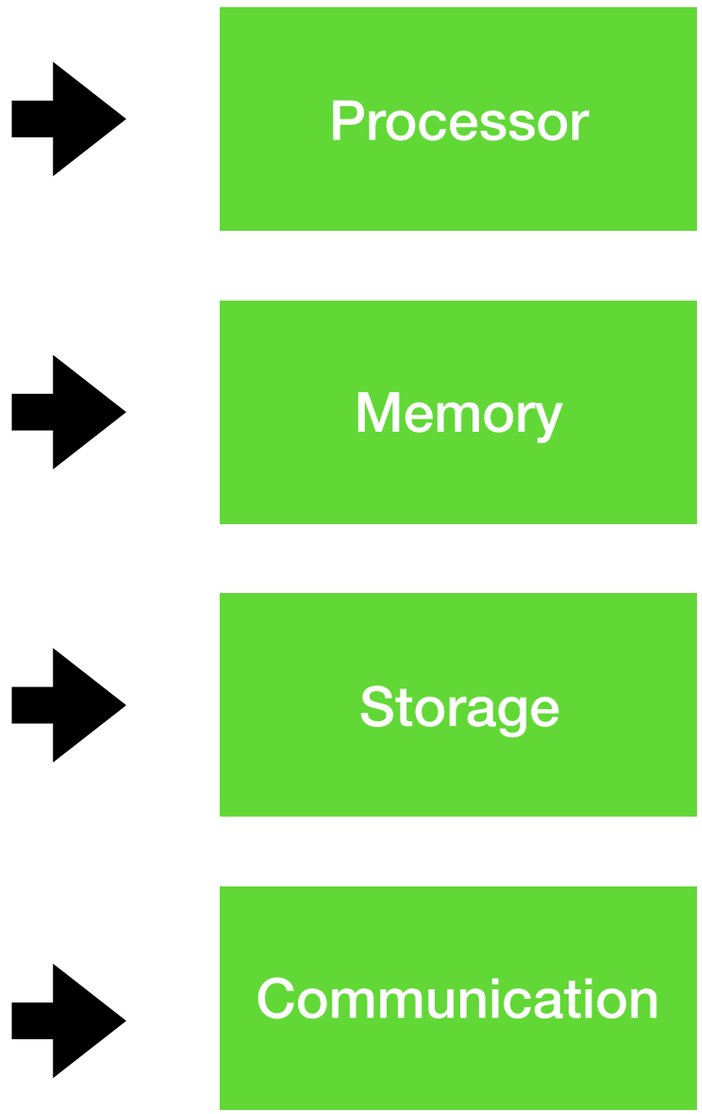
```
1   eegs % Array with EEG file names
2   results = zeros(500)
3
4   for idx=1:500
5       this_eeg = read_eeg(eegs(idx));
6       filtered_eeg = filter_eeg(this_eeg);
7       cut_eeg = epoch(filtered_eeg);
8       clean_eeg = clean(cut_eeg);
9       referenced_eeg = rereference(clean_eeg);
10      markers = compute_markers(referenced_eeg);
11      value = reduce_dimension(markers)
12      results(idx) = value
13  end
14
15  do_machine_learning(results)
```

- Possible problems:
  - Defect → Expected Results ≠ Actual Results
  - Bug → When we know about the defect!
    - Error/Fail → Manifestation of the defect
- Solution: Debug and correct
- Implies: > 11 days again!

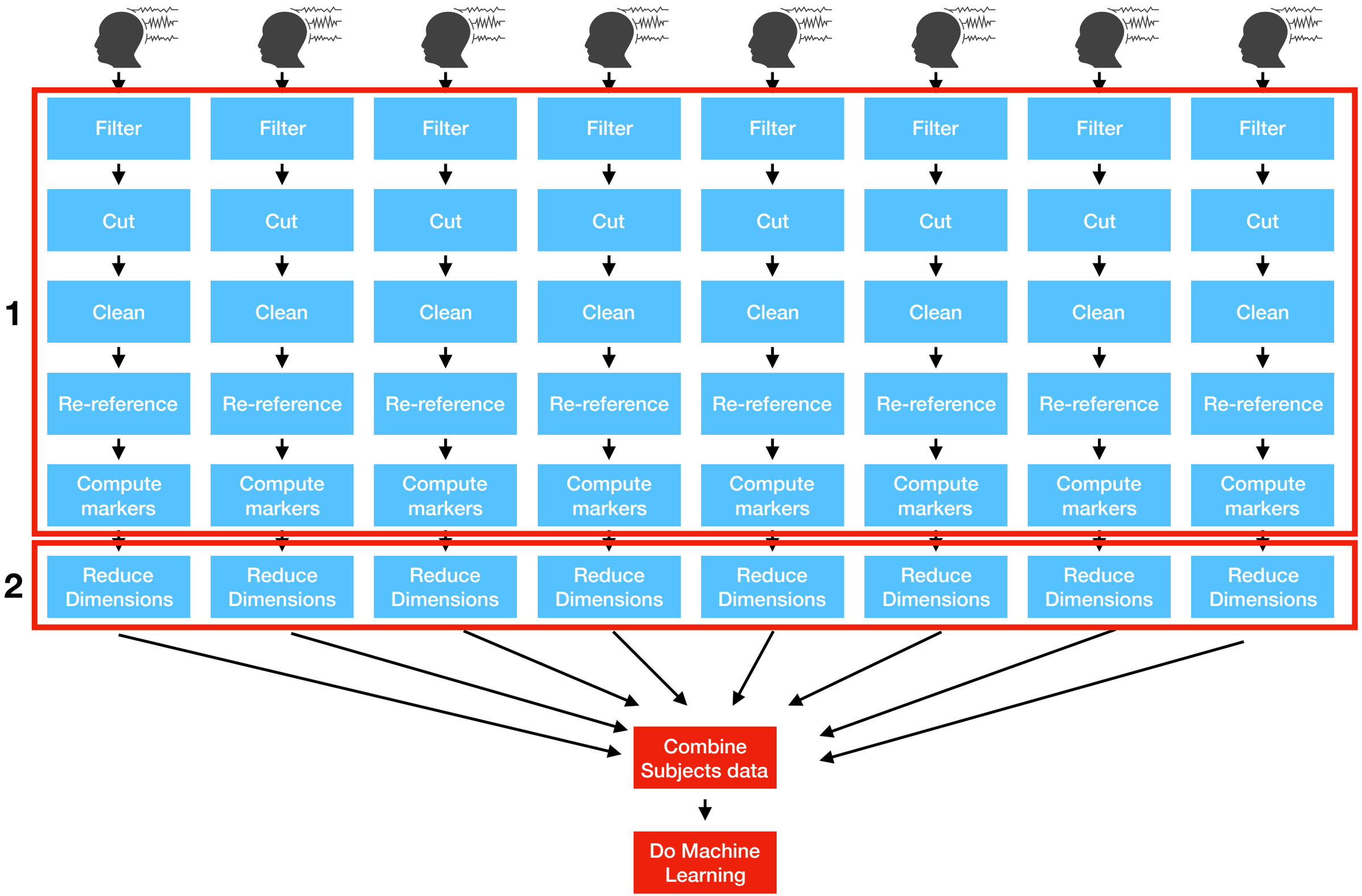
Parallel processing

Parallel processing: considerations

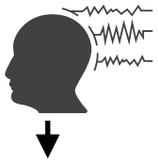




Parallel processing



Parallel processing: considerations



500 Mb

1 minute

Filter

Cut

Clean

Re-reference

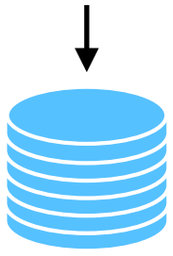
Compute markers

28.6 minutes

8 Gb

2 minutes

4.5 Gb



Processor — 54 cores

Memory — 128 Gb

Storage — 14 Tb
Speed: 2x HDD

Communication — Gigabit network

How much can we parallelise this step?

Parallel processing: considerations

	Available	Required	Overflow?	# Jobs
Processor	54 cores	8 cores for a few seconds 1 most of the time	Some overhead time (not much)	6.75 - 54
Memory	128 Gb	8	Storage is used	16
Storage	14 Tb Speed: 2x HDD	4.5 Gb	Crash (and good luck!)	
Communication	Gigabit network	None	None	

10 Jobs → ~27 Hours
15 Jobs → ~18 Hours

Parallel processing: running

1. Create a script that given a subject/sample number, runs the whole analysis
2. Use GNU parallel to run the script

```
parallel -j10 --joblog job-analysis.log --resume --resume-failed --tag --delay 30 echo {} ::: `seq 1 10`
```

1 **2** **3** **4** **5** **6** **7** **8**

1. Number of Jobs to run in parallel
2. Keep a log of what happens
3. Resume where we left (if stopped)
4. Re-run the failed ones
5. Print a tag with the parameters before the output
6. Waiting time between processes
7. Command
8. Values to replace {} in the command

Parallel processing: running (example)

1. Create a script that given a subject/sample number, runs the whole analysis

run_analysis.m > ...

```
function run_analysis(sample_id)
    disp(['Running sample ', num2str(sample_id)])
end

```

2. Use GNU parallel to run the script (within a bash script)

parallel_analysis.sh

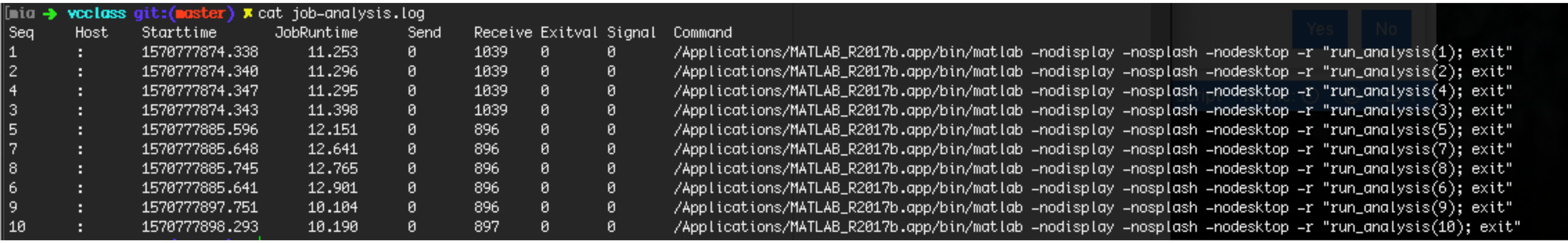
```
#!/bin/bash

njobs=4
samples=`seq 1 20`
command='/Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis({}); exit" '

parallel -j$njobs --joblog job-analysis.log --resume --resume-failed --tag --delay 0 $command ::: $samples

```

- Alternative declaration of samples

```
samples=(Name1 Name4 Sample23)
```

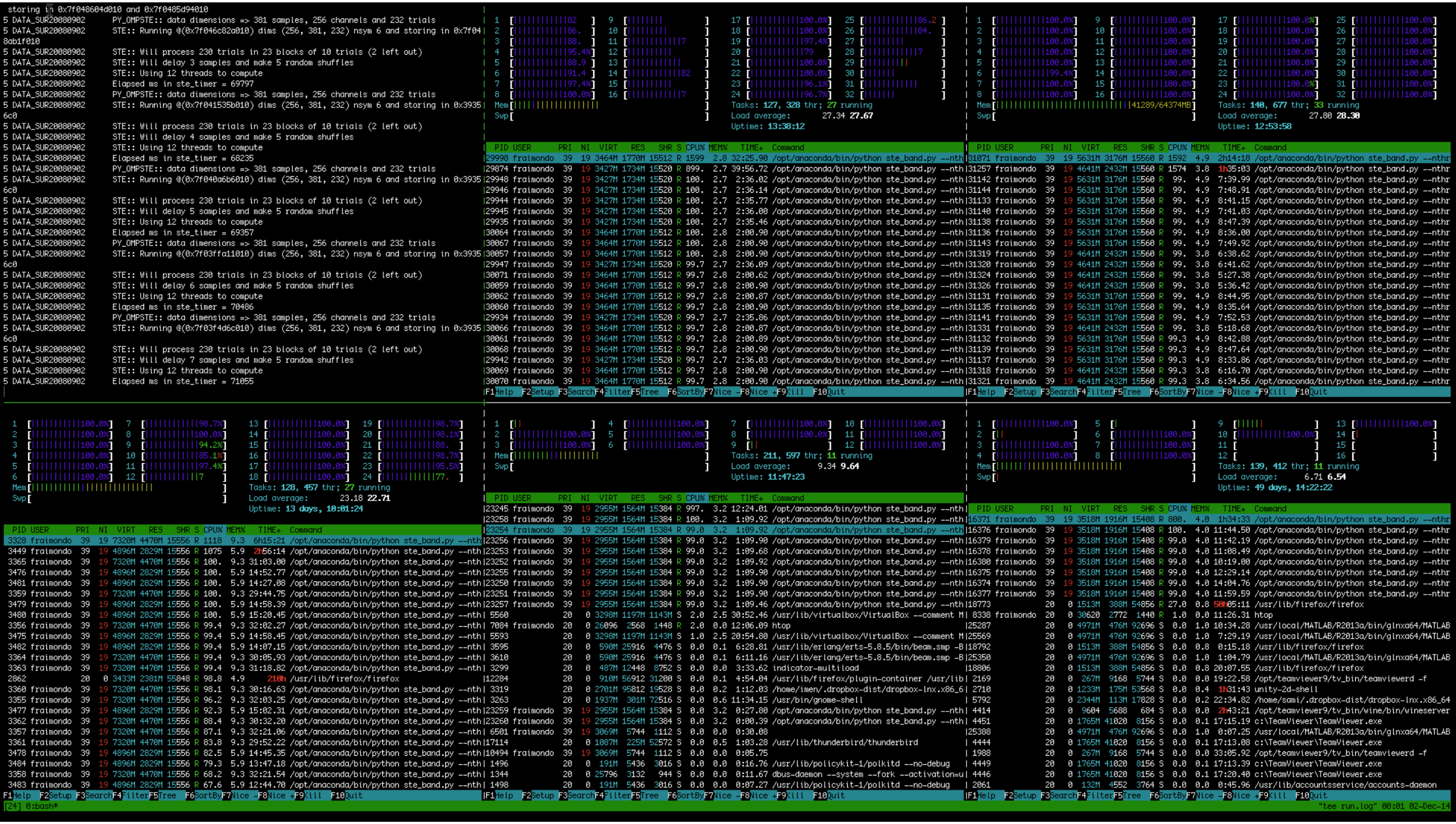
Parallel processing: running (example)

Parallel processing: running (example)



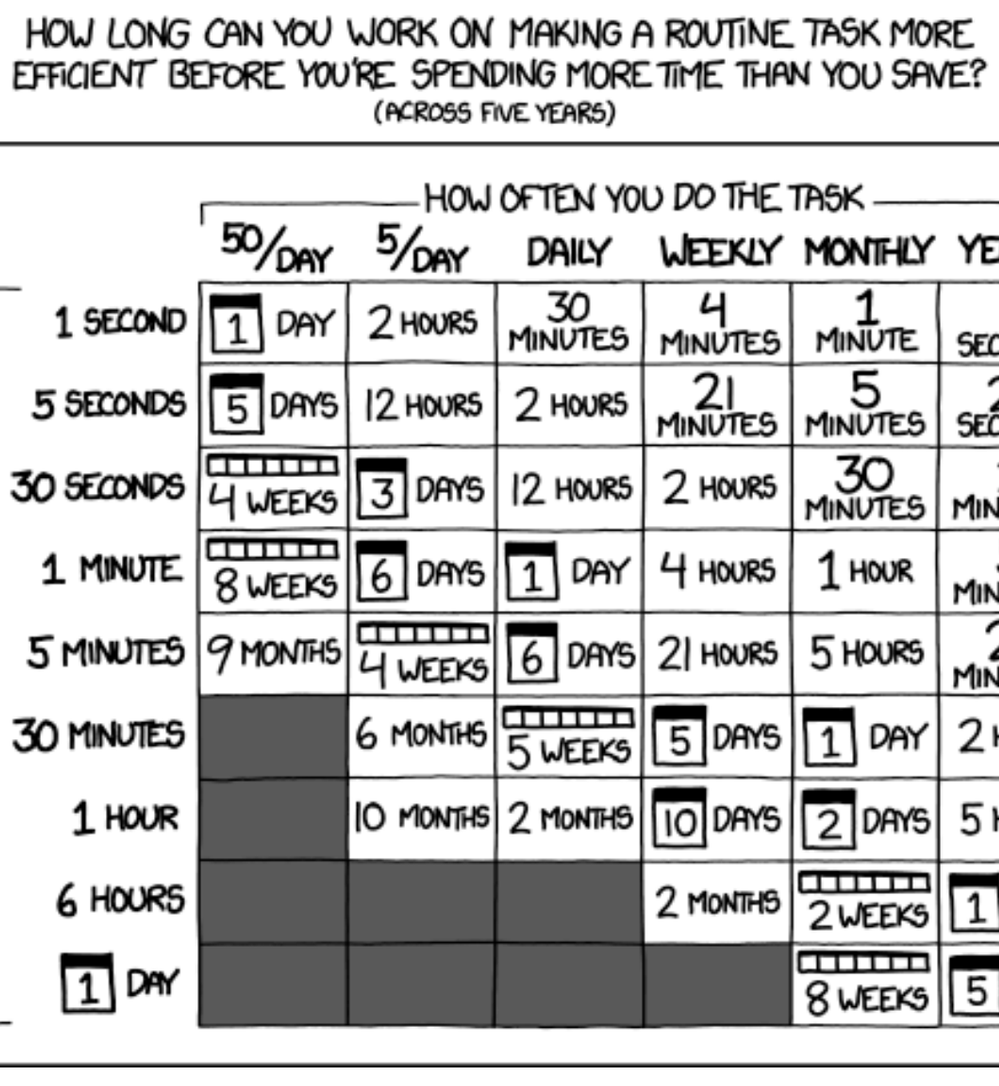
```
mia → vcclass git:(master) ✗ cat job-analysis.log
Seq     Host    Starttime       JobRuntime      Send    Receive Exitval Signal  Command
1       :       1570777874.338     11.253       0       1039    0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(1); exit"
2       :       1570777874.340     11.296       0       1039    0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(2); exit"
4       :       1570777874.347     11.295       0       1039    0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(4); exit"
3       :       1570777874.343     11.398       0       1039    0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(3); exit"
5       :       1570777885.596     12.151       0       896     0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(5); exit"
7       :       1570777885.648     12.641       0       896     0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(7); exit"
8       :       1570777885.745     12.765       0       896     0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(8); exit"
6       :       1570777885.641     12.901       0       896     0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(6); exit"
9       :       1570777897.751     10.104       0       896     0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(9); exit"
10      :       1570777898.293     10.190       0       897     0       0       /Applications/MATLAB_R2017b.app/bin/matlab -nodisplay -nosplash -nodesktop -r "run_analysis(10); exit"
```

Parallel processing: running (my example)



Remarks

1. You will most probably need to learn a bit about Unix/Linux
2. Check local cluster computing service
   - Matlab on those settings is not that easy
3. Use more than one computer
   - GNU parallel supports that!
4. Don’t be afraid of asking for help
5. Invest time in saving time



Source: xkcd.com

fraimondo@uliege.be

@fraimondo

@RaimondoFede