Power Trace Analysis

Marco Torchiano

Politecnico di Torino (Italy)

30 August 2016

SoftEng
http://softeng.polito.it

Contents

Goals

Analyze power traces from software execution

- understand what are the issues and challenges
- know the tools and methods
- set up a workflow

Case study

Bubble sort of a 10k elements `int` array.
Raspberry Pi 1A

- ARM v6 architecture
- 32-bit CPU 700 MHz single-core

30 repetitions of sorting task

- duration 817 ms each
- total experiment 71.3 s
- overall 712,698 samples

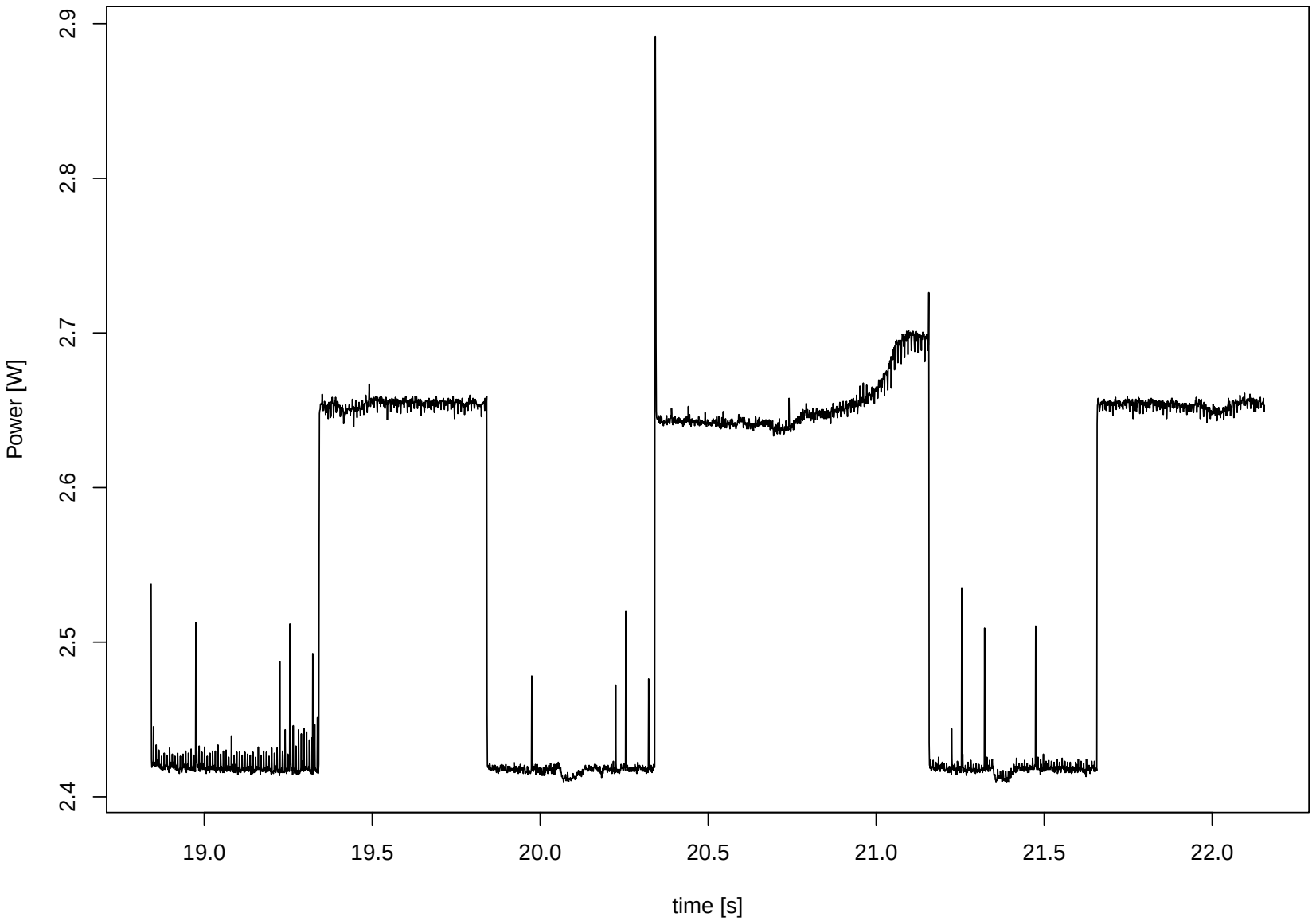
Repetitions

- Several repetitions are needed to average out measurement error
- How to isolate individual repetitions:
  - Strict coordinated timings bewteen SUT and MD
  - Synchronization signals between SUT and MD
  - Markers added in the trace

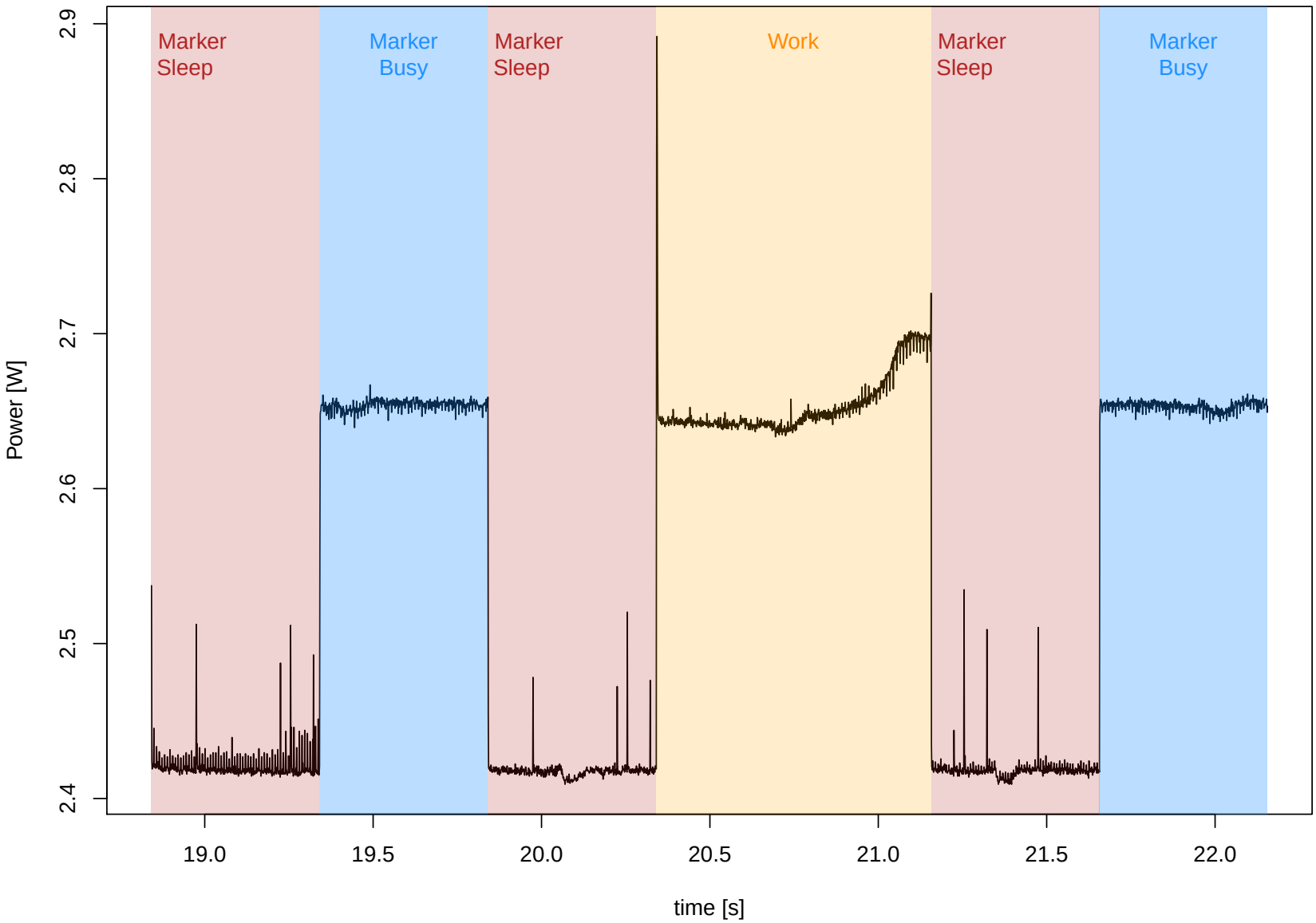
Generate a marker

```
Thread.sleep(markerLength);              // SLEEP
long temp=System.currentTimeMillis()+ markerLength;
while(temp>System.currentTimeMillis()){ // BUSY
  int count = 0;
  for(int i=0; i<markerLength; ++i){
    count += i;
  }
}
Thread.sleep(markerLength);      // SLEEP
```

First look at the trace



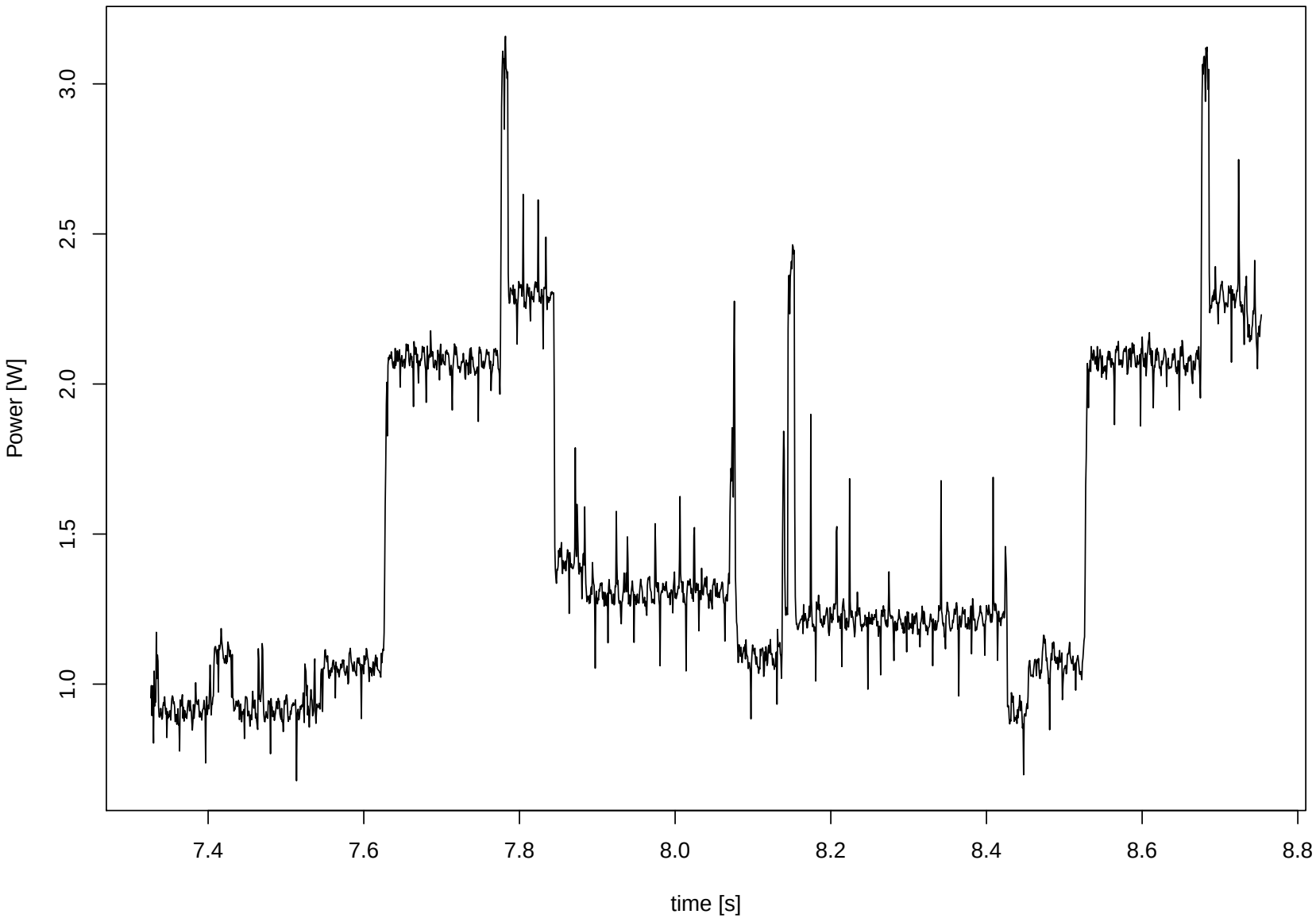
First look at the trace explained



Issues

- Noise:
  - difficult to detect marker step
  - instant power is hard to detect
- Size
  - easily more than 1 M samples
  - plot with care (downsample!)

Another example

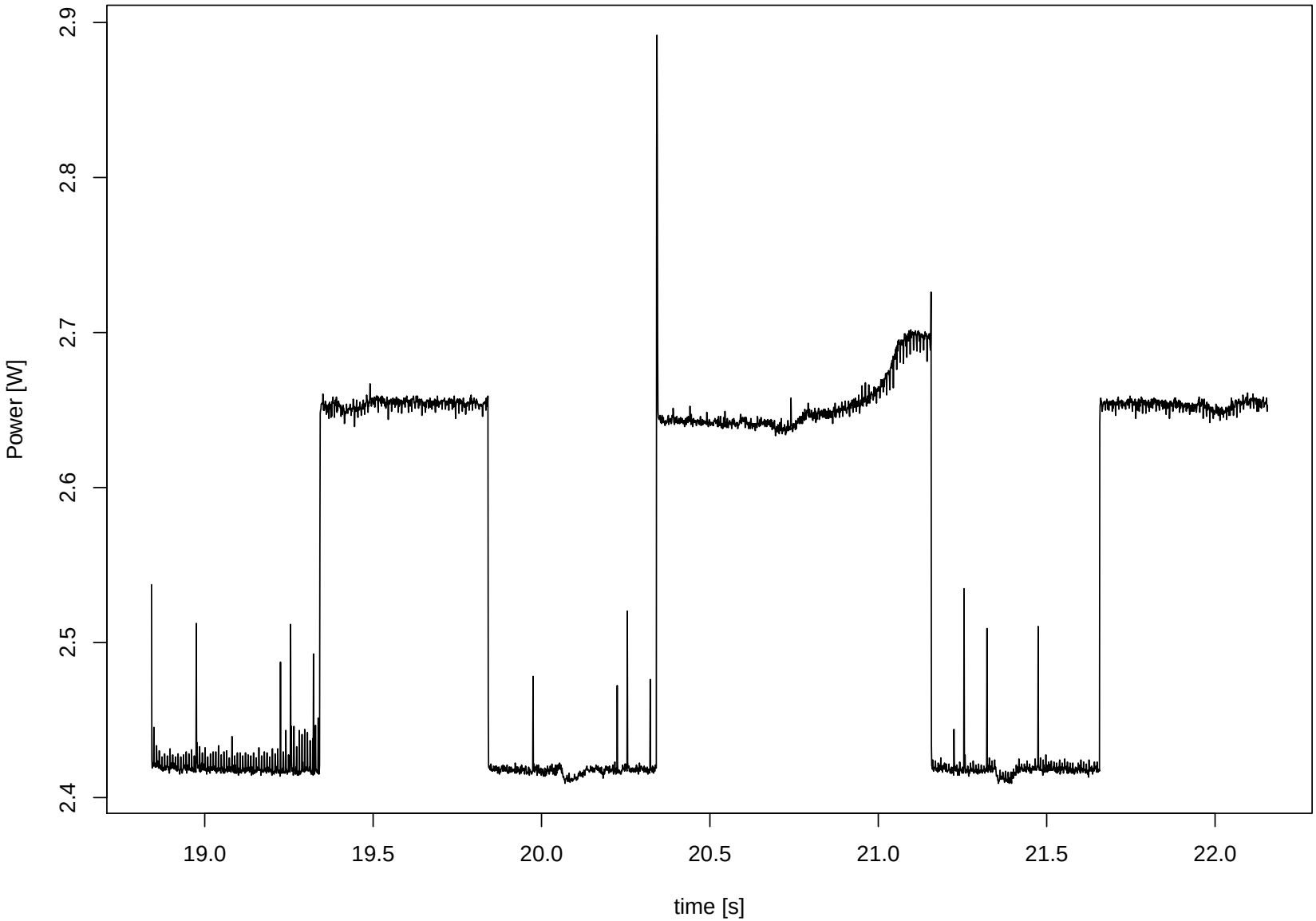


Another example

This was from an Android (Nexus LG 4) quicksorting 50k `ints`.
Again 30 repetitions:

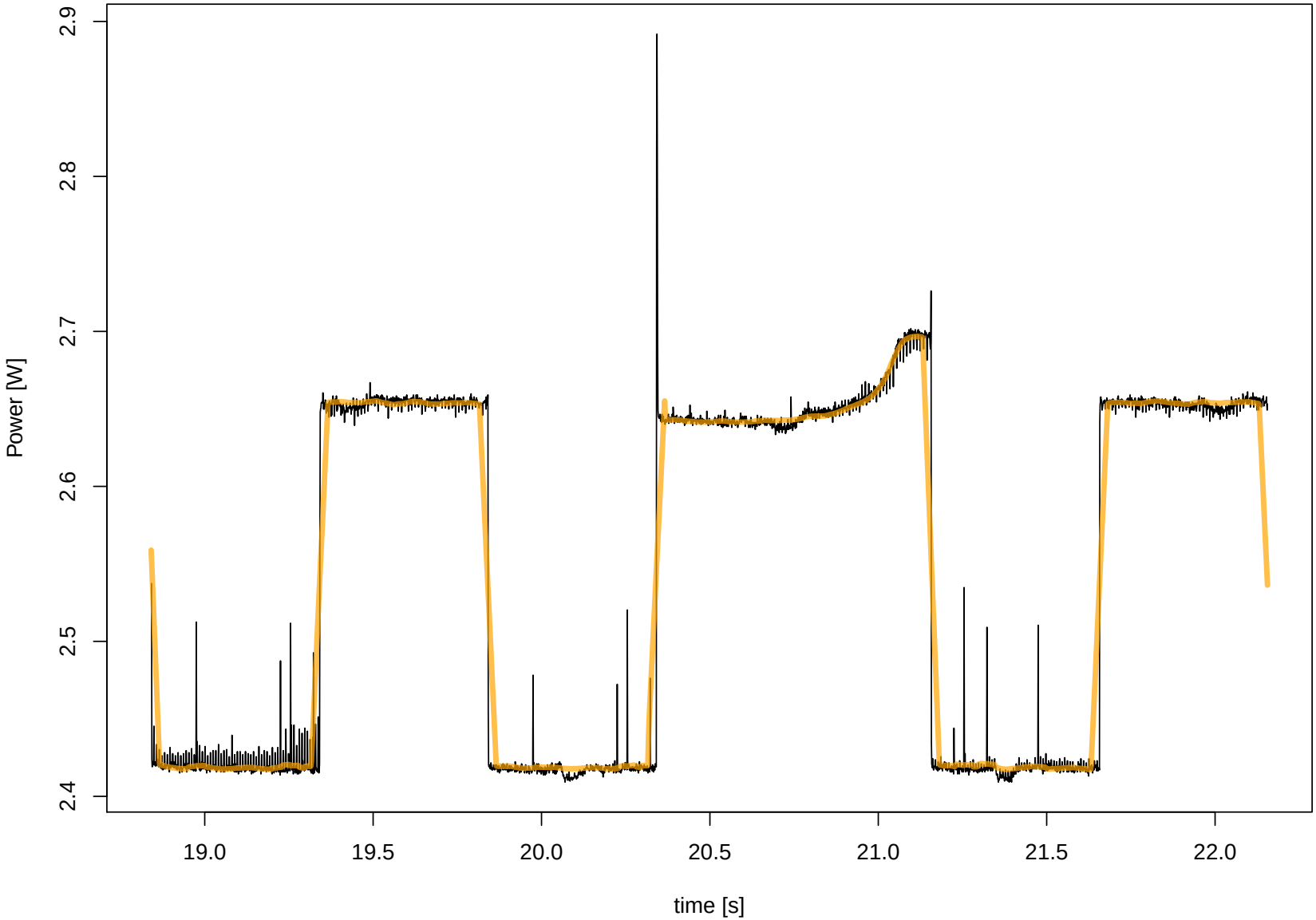
- duration 85 ms each
- total experiment 370.3 s
- overall 3,703,000 samples
  - took 14.831s to load' on my MBP

Detecting the edges



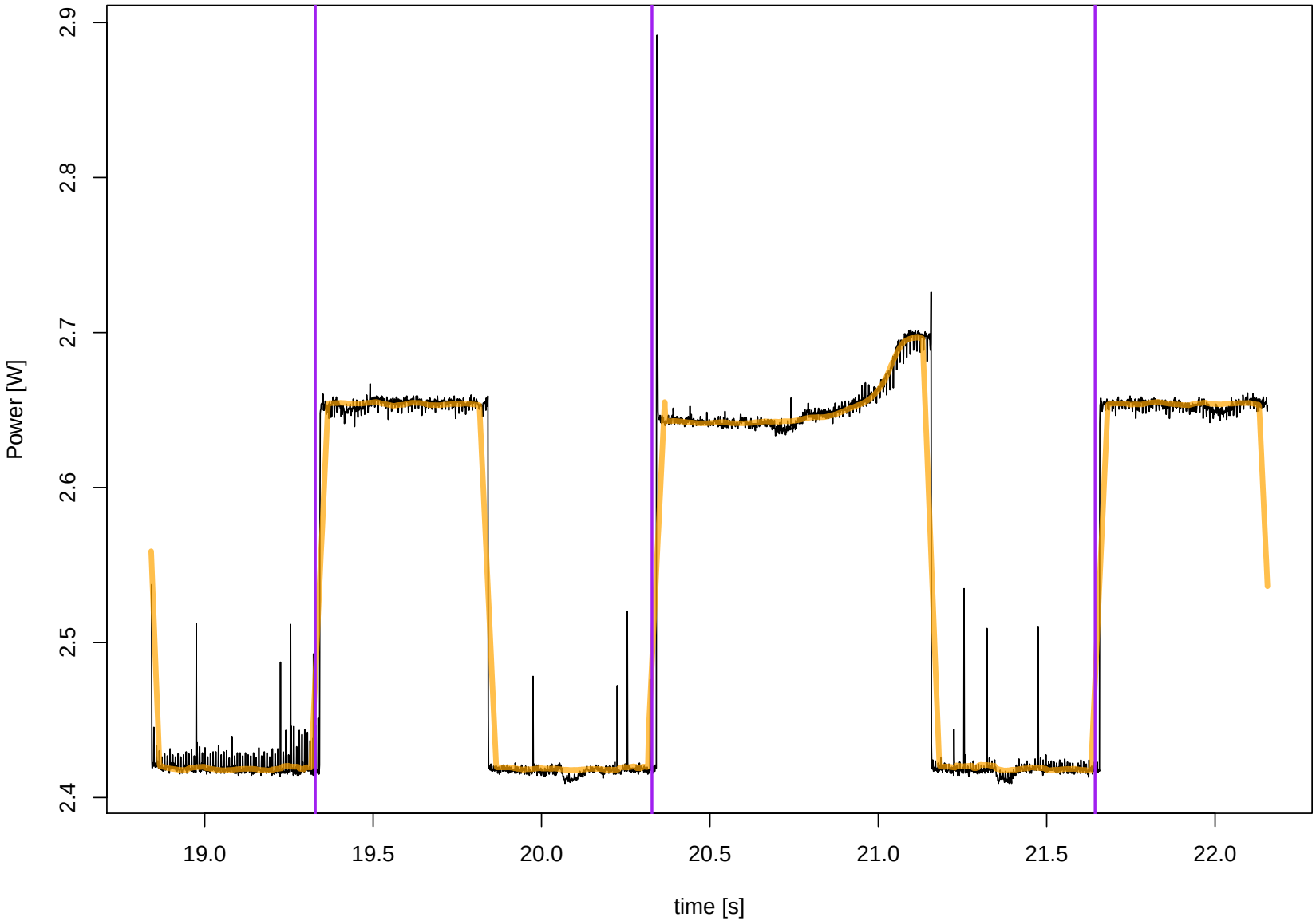
How to avoid spurious edges?

Detecting the edges: the idea



Apply low-pass filter (or computationally faster moving average)

Detecting the rising edges: the idea



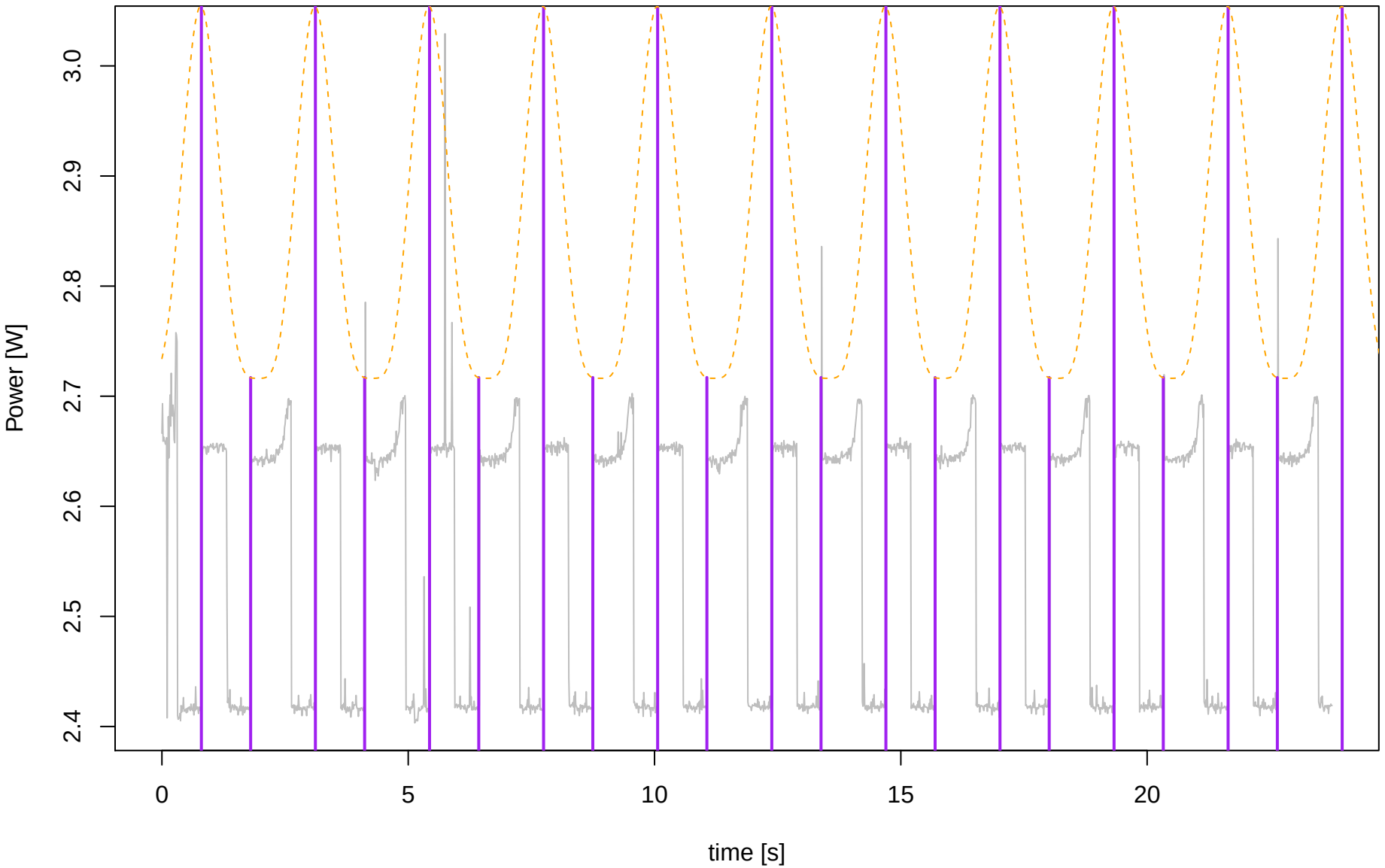
Look for edges in the filtered samples.

Identification of markers

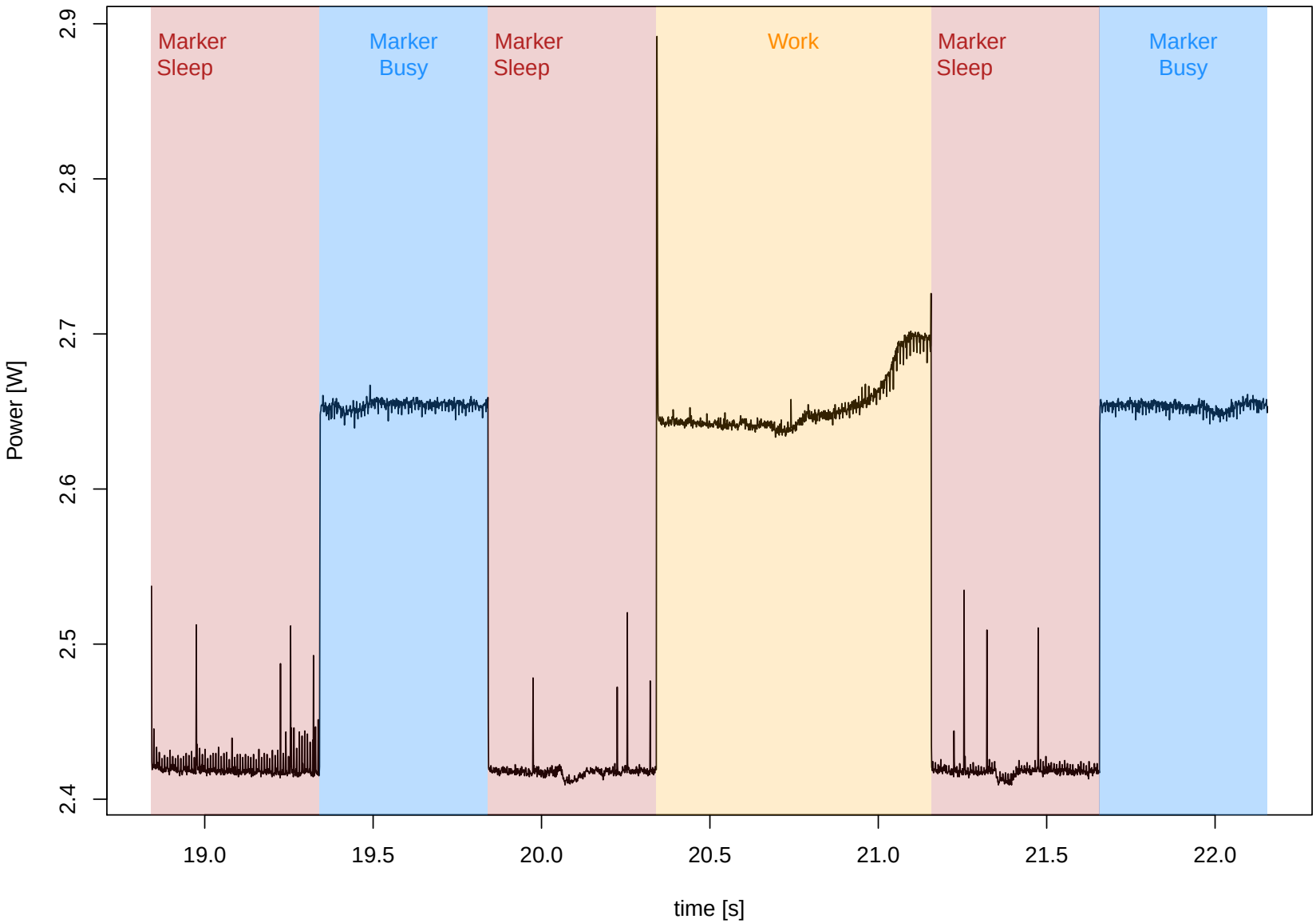
- Edges should precede a step with the marker length's width (+ or -)
- Fit a cyclic pattern
  - Frequency: should fit 30 or less cycles
  - Starting from the first to the last candidate edge

$$\left(1 + cos\left((x - first) \cdot \frac{n \cdot 2\pi}{last - first}\right)\right)^2$$

Identification of markers



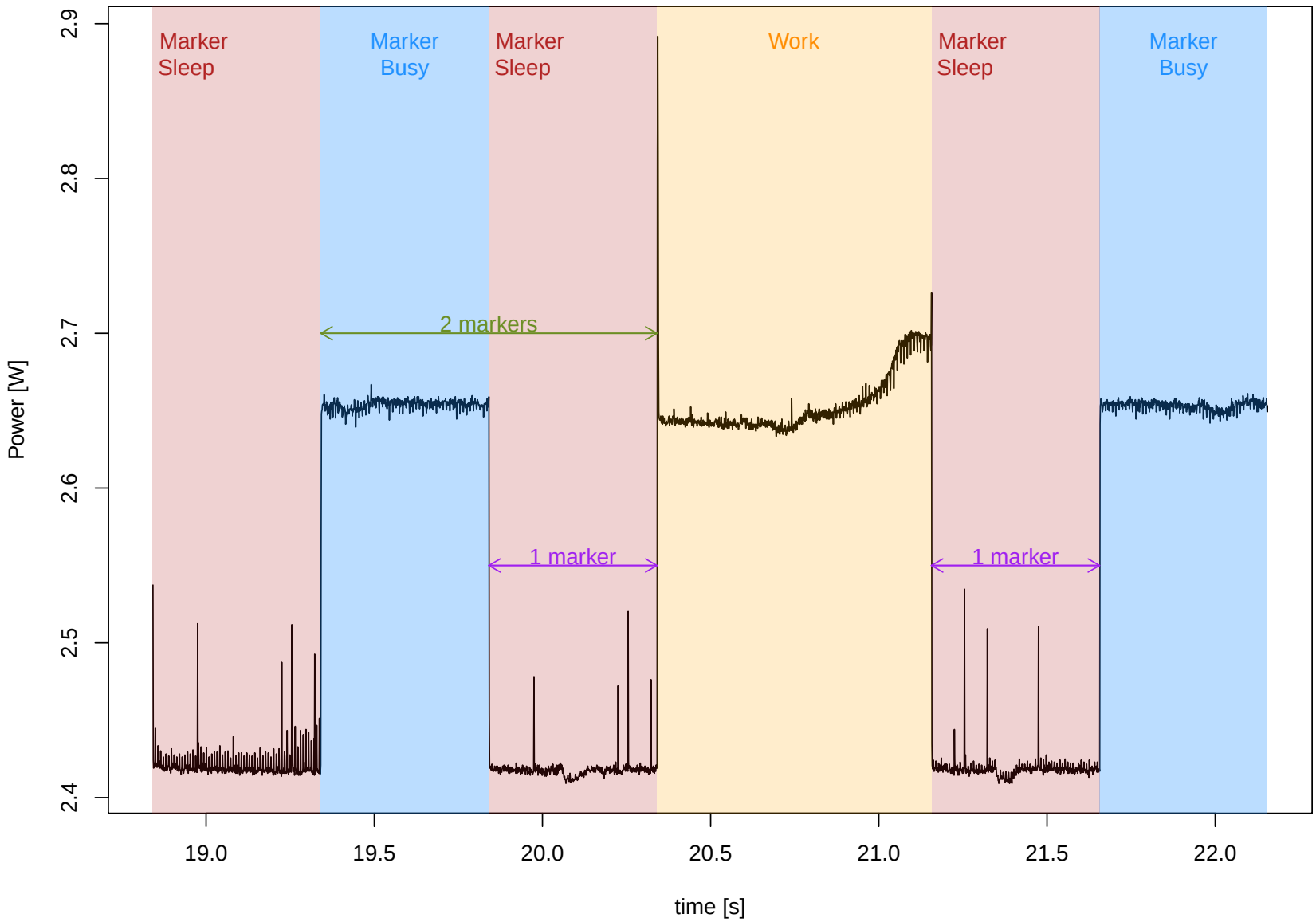
Identification of work units



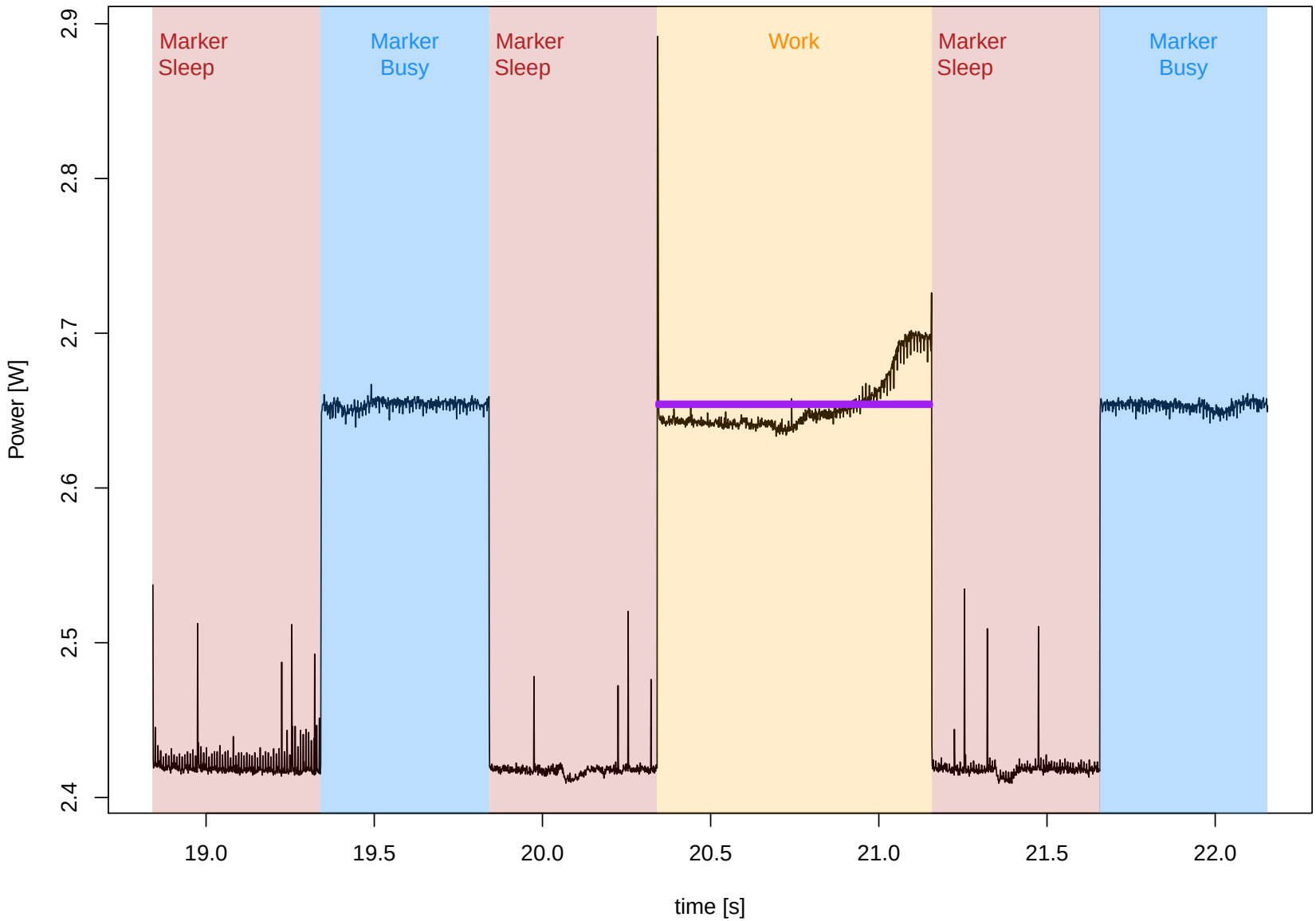
Processing the work units

- when does the work start and when does it end?
- what to measure?
  - average vs. single samples
- what is the energy attributable to the program under test?
  - difference between real and effective power

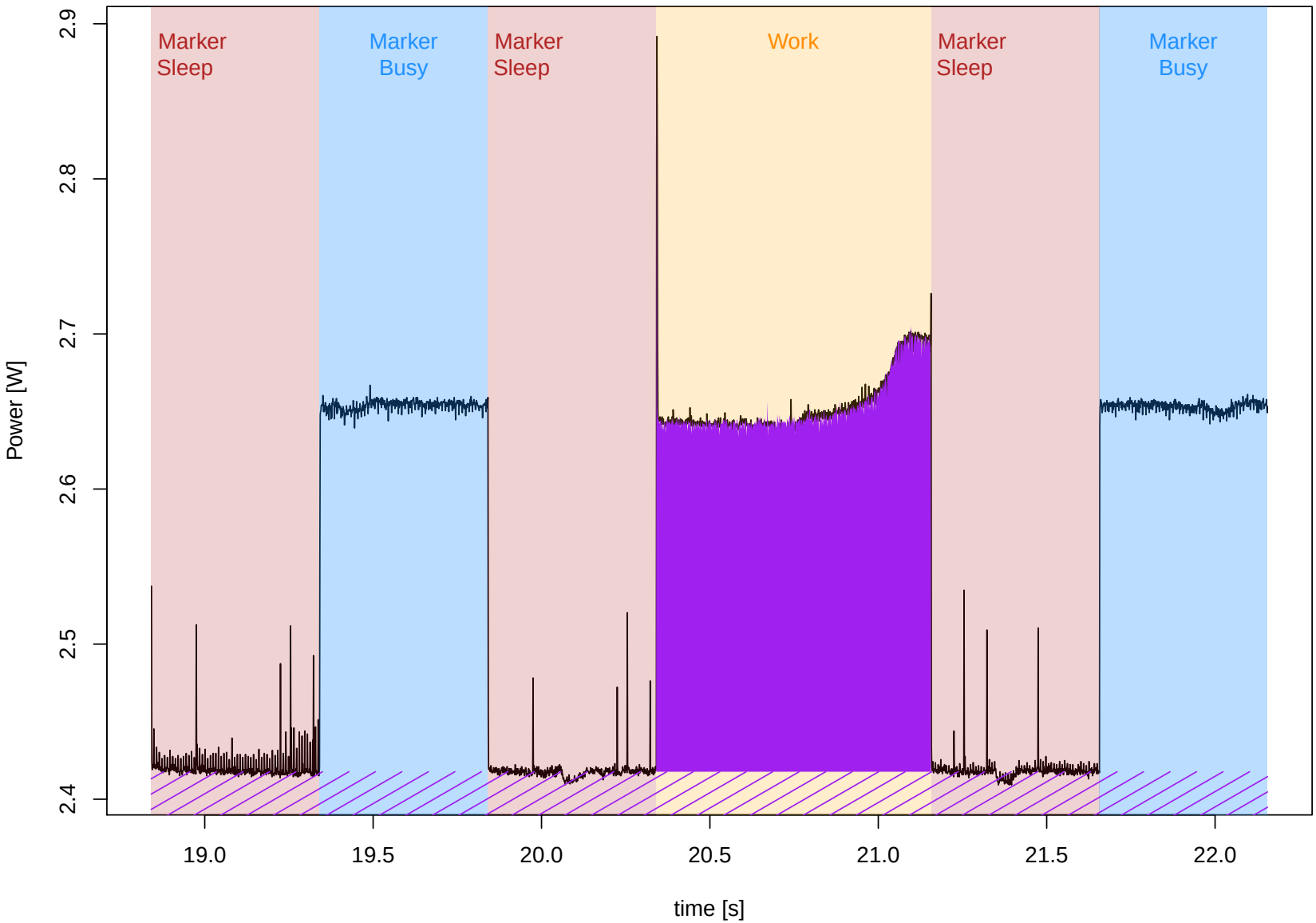
When does the work take place?



What to measure?



What is effective work?



Effective work

Effective work = time * (power - baseline)
Basline power can be computed:

- Locally
  - the left sleep period
  - the right sleep period
  - the average of the two
- Globally
  - the average of all left sleep periods
  - the average of all right sleep periods
  - the average of all sleep periods

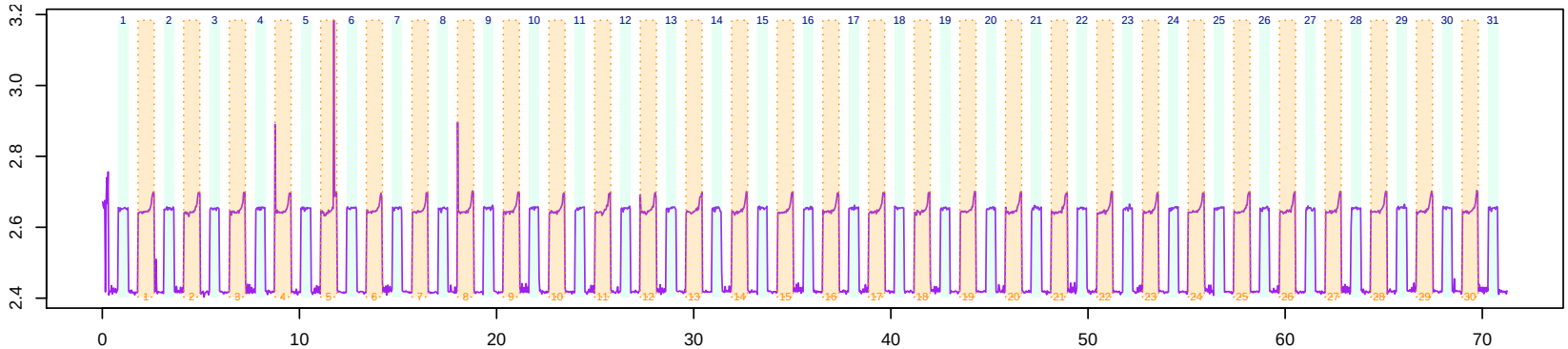
Asking a RQ and analyzing the data

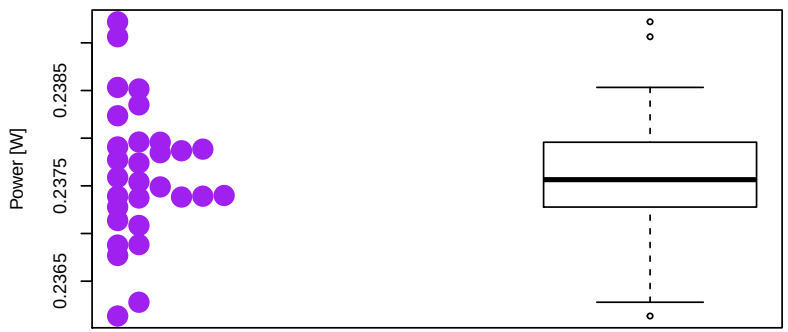
Now that you have the final aggregate data you can answer some interesting research questions.
Look at your data first:

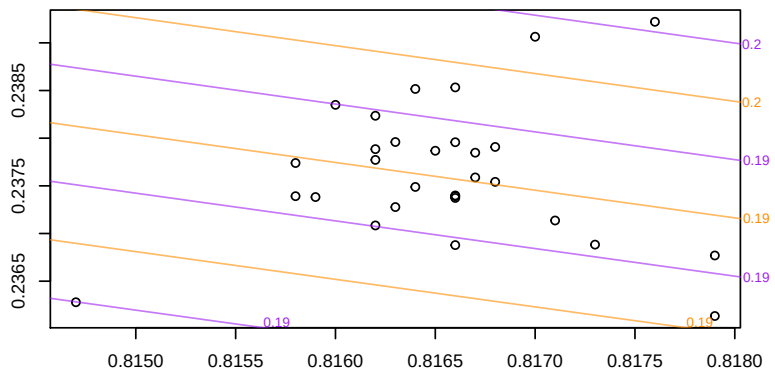
- how much are they dispersed?
- what is their range?
- look at time, power, energy?

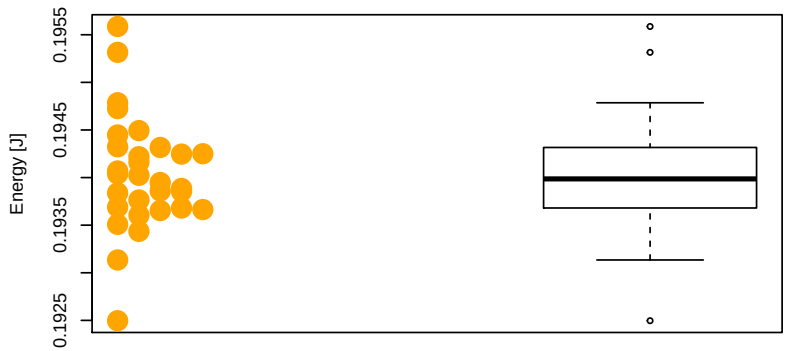
Then you can formulate your hypotheses and expect to get some reasonable answer

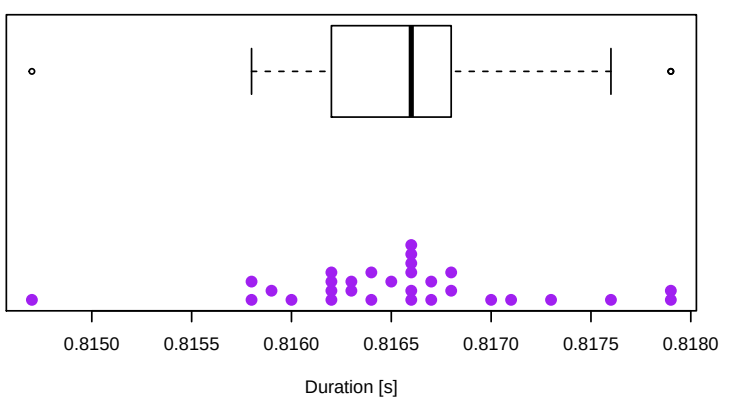
Look at your data



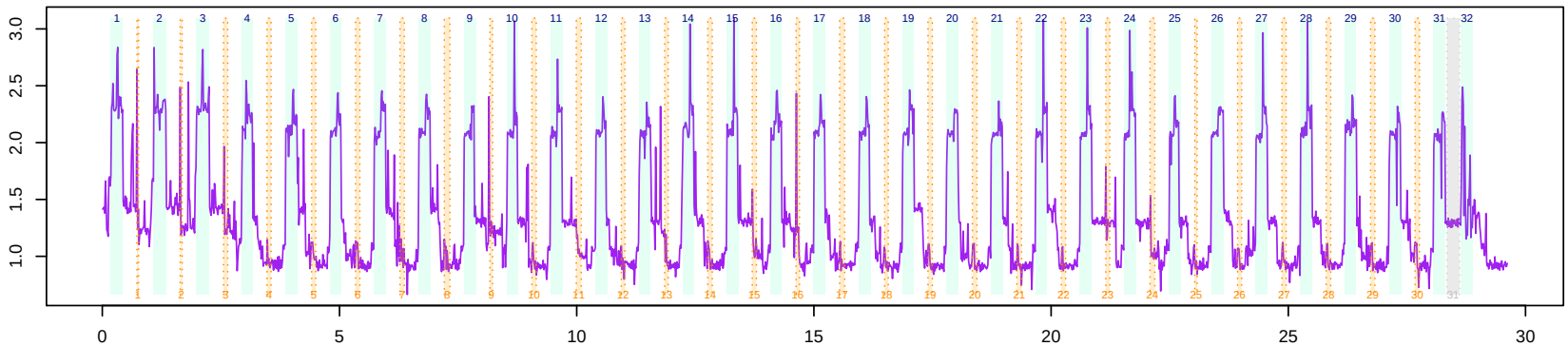


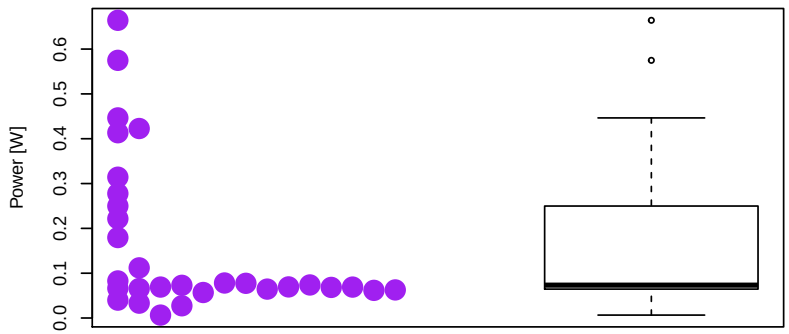


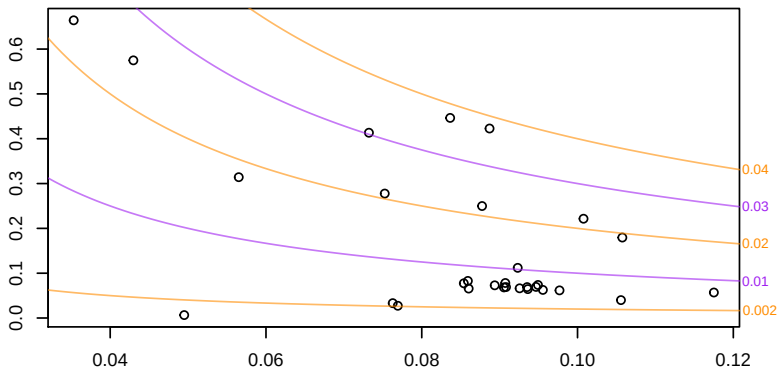


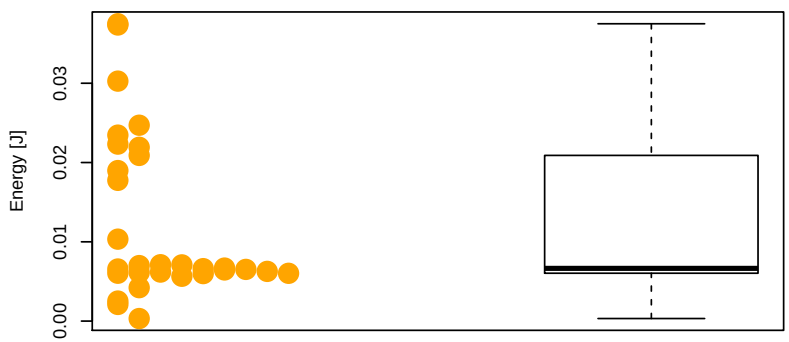


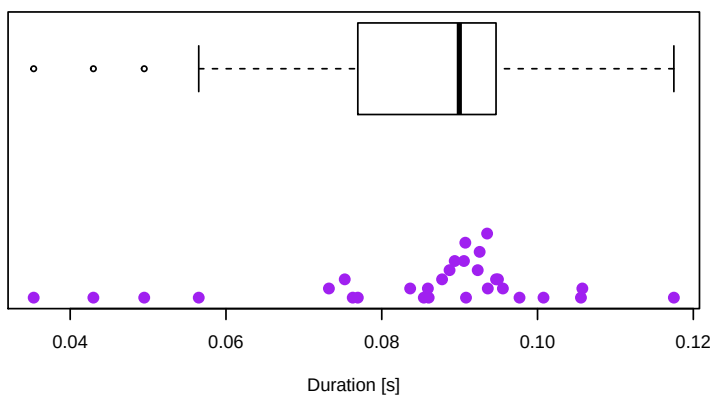
Look at your data











The package `powtran`

R package, contains all the above steps conveniently packed:

- *just* 700 lines of code
- tested on several different traces
- optimized (3.315 s to process 3.7M samples)

Download it from:
`http://softeng.polito.it/torchiano/powtran_0.1.tgz`
Available on github: `https://github.com/SoftengPoliTo/powtran`

powtran usage

Usage:

```
url="http://softeng.polito.it/torchiano/powtran_0.1.tgz"
install.packages(url, repo=NULL)
library(powtran)
res = extract.power(data,         # samples
                    t.sampling,   # sampling period
                    N,            # num. repetitions (30)
                    marker.length, # marker step duration
                    baseline      # method for baseline
                                  # computation
                    )
```

powtran results

`res$work`:

start	end	t	P real	P left	P right	P	E
18136	26282	0.815	2.653	2.417	2.417	0.236	0.192
41276	49454	0.818	2.653	2.416	2.416	0.236	0.193
64446	72604	0.816	2.654	2.418	2.417	0.237	0.194
87596	95759	0.816	2.654	2.418	2.418	0.237	0.194
110756	118931	0.818	2.656	2.418	2.418	0.239	0.196
133926	142092	0.817	2.654	2.418	2.418	0.238	0.194
157086	165255	0.817	2.655	2.418	2.417	0.239	0.195
180246	188410	0.816	2.654	2.418	2.418	0.238	0.194
203406	211563	0.816	2.654	2.418	2.418	0.237	0.194
226556	234721	0.817	2.654	2.418	2.418	0.237	0.194

Basic workflow

1. Load your data (it can take a while. . .)
2. Process it to extract power/energy info
3. Analyze your data

Load your data

You can load you data from within a zip file.

```
# given a full path in zipfilename
all.files = unzip(zipfilename,list=TRUE)
name = all.files$Name[1] # pick the 1st
conn = unz(zipfilename,name)
samples.m = read.csv(conn)$Power
      # pick only the Power column
```

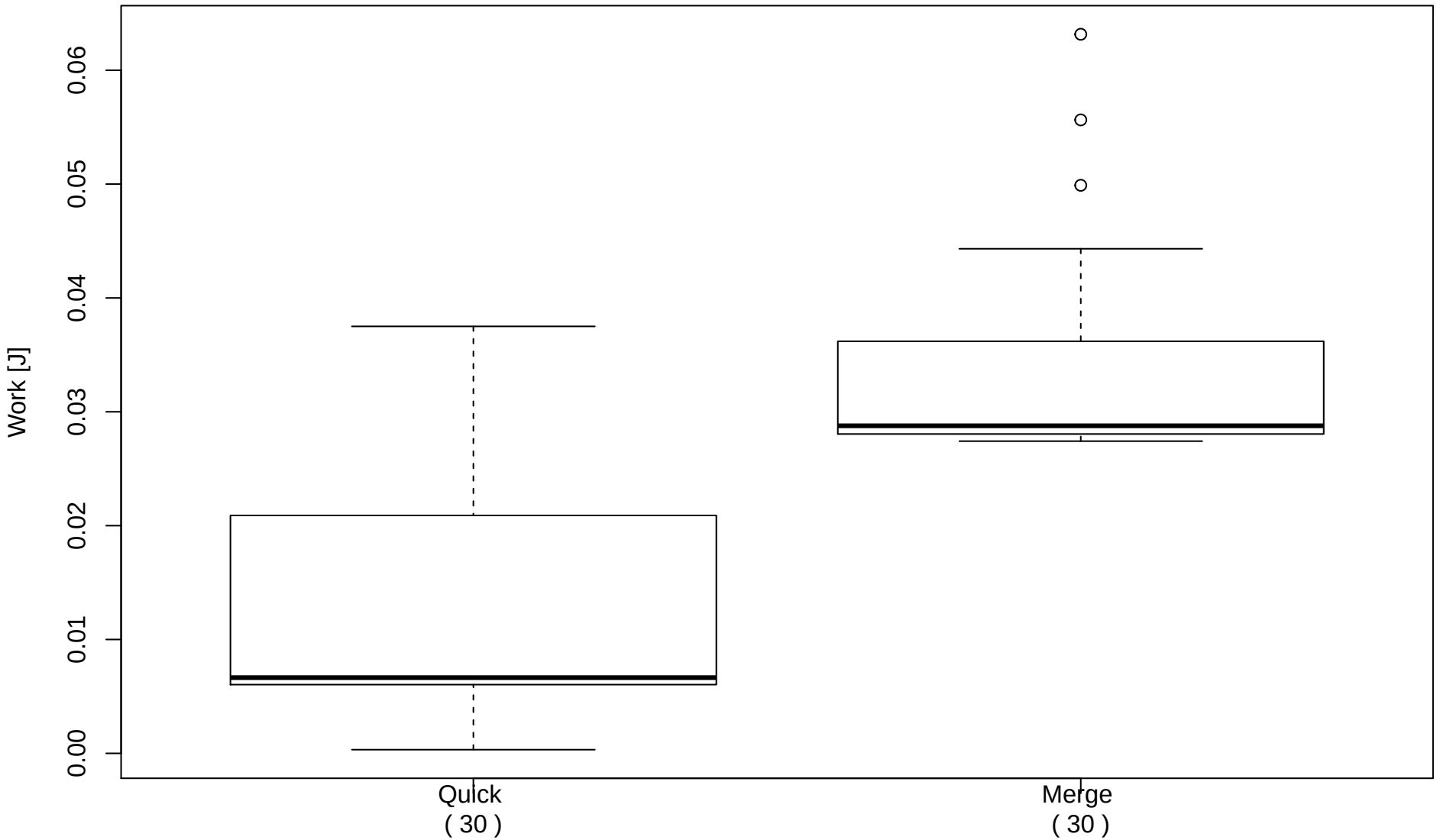
Process your data

Use the `extract.power` function to process the samples

- sampling freq is 125 kHz
- marker length is 300 ms
- baseline is computed as global minimum (`"gmin"`)

```
library(powtran)
res.merge = extract.power(samples.m,
                          t.sampling = 1/125000,
                          marker.length = .300,
                          baseline="gmin")
```

Analyze your data



Analyze your data

Use any suitable statistical approach, e.g. non-parametric stats.

```
wilcox.test(res.quick$work$E,res.merge$work$E,conf.int = TRUE)
```

```
##
##  Wilcoxon rank sum test
##
## data:  res.quick$work$E and res.merge$work$E
## W = 69, p-value = 4.126e-10
## alternative hypothesis: true location shift is not equal to
## 95 percent confidence interval:
##  -0.02278451 -0.02100447
## sample estimates:
## difference in location
##             -0.0219367
```

That's all folks

Remember:

- Plan carefully you experiments
- Take noise and size into consideration
- **R** and **RStudio** are your friends

License