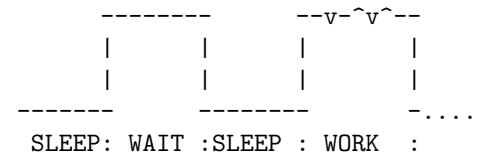
Analysis Procedure

Marco Torchiano

19 November 2015

Computes the average power for the working tasks in a series of measures.

The computation assumes the execution follows a well defined protocol to mark tasks begin and end. A typical profile is:



`SLEEP` : is a period of sleep for a given (marker.length) time

`WAIT` : is a period of busy waiting for a given (marker.length) time

`WORK` : is a period of actual work (the one we aim to measure the energy consumption of)

Load data

The analysis procedure can be applied on data containing Power samples.

Typically file containing current (`I`) values (expressed in Ampere) is the starting point.

The power, given a *constant* `voltage` in output from the generator (5.03 V) – can be computer using the basic relation

$$P = V \cdot I$$

```
  filepath = "data/syscall_mmap2.txt"
  #filepath = "data/syscall_futex.txt"
  #filepath = "data/syscall_gettimeofday.txt"
  #filepath = "data/syscall_mprotect.txt"
  voltage = 5.03 ## Volt

  # read the file containing current values (in A)
  data = read.delim2(filepath, header = FALSE, skip = 7)
  names(data)="I"

#  data = read.csv("~/Dropbox/PhDRifat/ENERGY ESEM 2015/Samples/Java/JavaBubble10000.csv")
#  names(data)[2]="I"
#  marker.length=1000

  # compute the power
  data$P <- with(data, I*voltage )
```

Input parameters

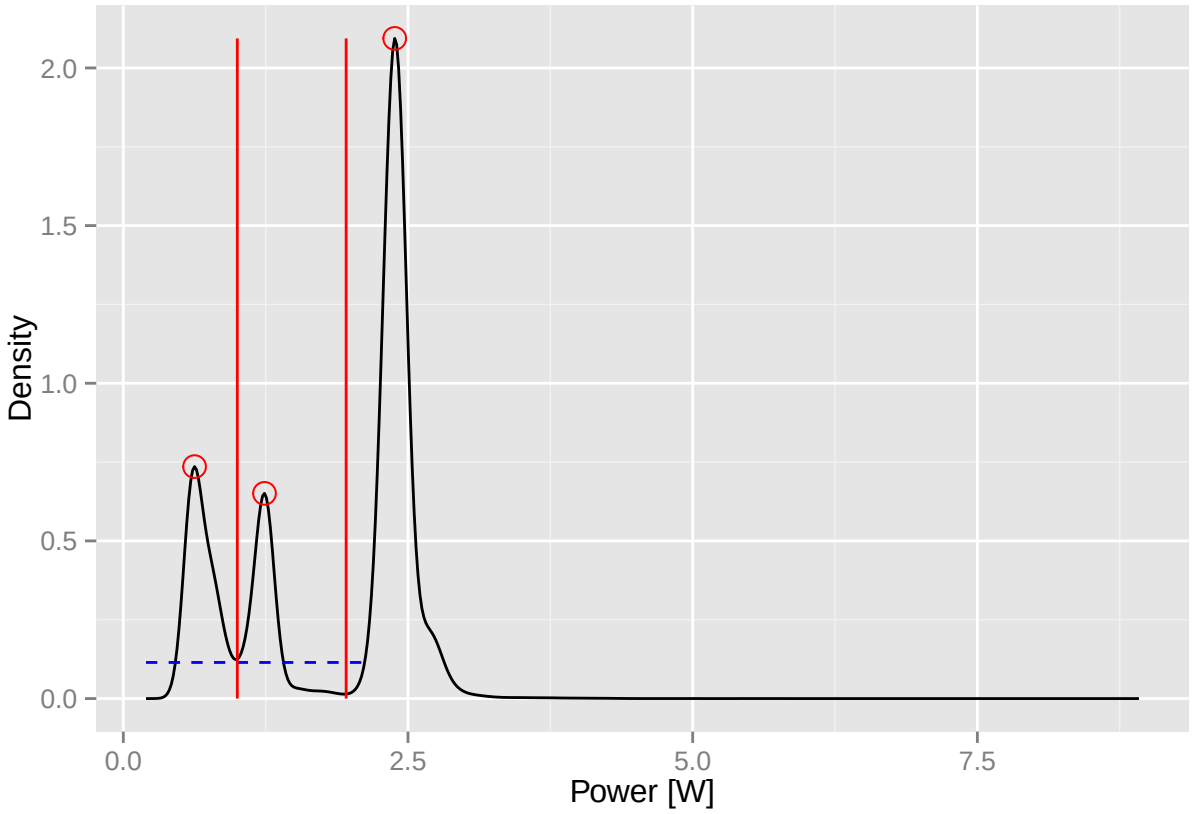
The algorithm takes the following input parameters:

```
adjust = 1.5 ## density parameter
marker.length = 5000
marker.tolerance= 0.1
N = 30 # number of task markers
```

Density analysis

Compute the distribution density of the instant power in order to identify the typical working levels.

```
  ## Distribution density of values
  dens <- density(data$P,adjust=adjust)
  #dens <- density(data$P)

  ## Find peaks in the distribution (the most common power levels)
  dens$peaks <- which(peaks(dens$y) & dens$y>mean(dens$y))

  ## Identify thresholds between peaks as the points of minimum density between peaks
  thresholds = dens$x[apply(cbind(head(dens$peaks,-1),tail(dens$peaks,-1)),1,function(x){
    ss = dens$y[x[1]:x[2]]
    x[1] + which(ss == min(ss))
  })]

  baseline = dens$x[dens$peaks[1]]
```



The baseline, representing the idle level of consumption correspond to the first peak in the density plot: 0.6256976 W.

Tagging

Each peak is assigned to a level, the the thresholds are used to discriminate between consecutive levels. The data points are tagged with the level name.

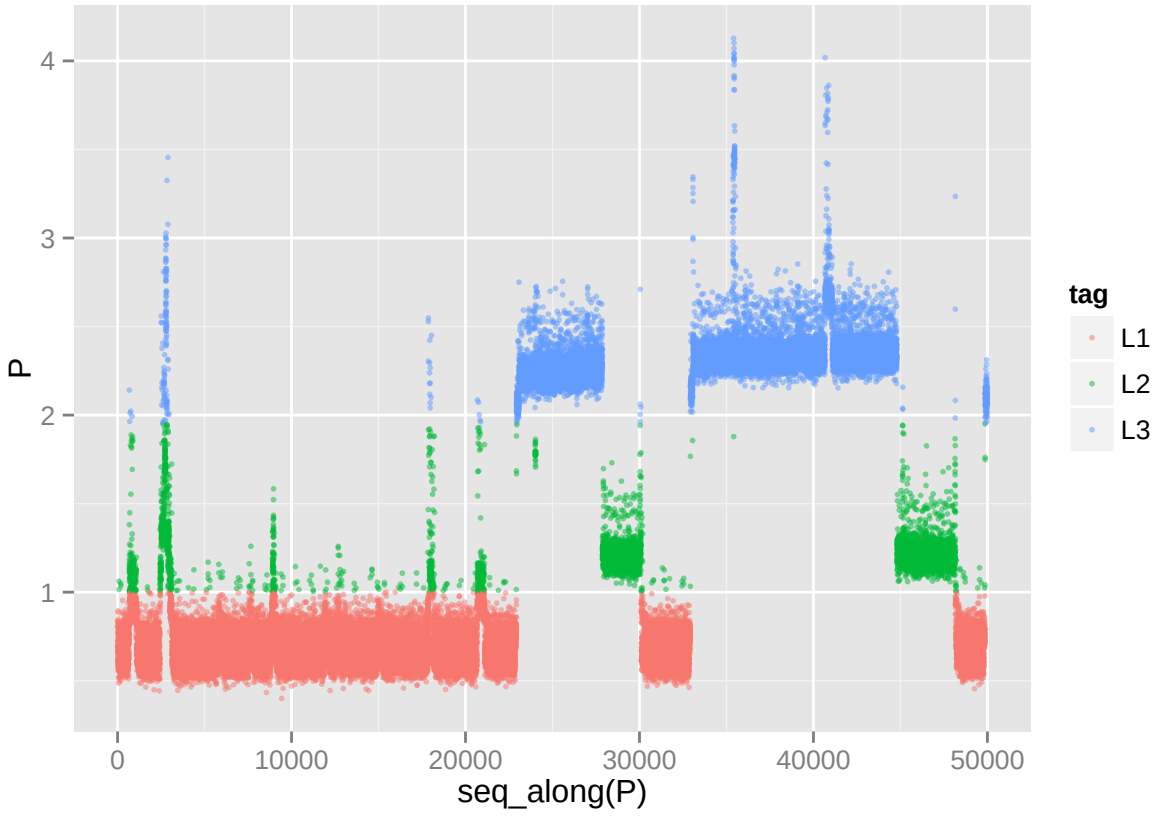
The tagging indetify runs of consecutive points having the same level. The runs are then identified with a unique number.

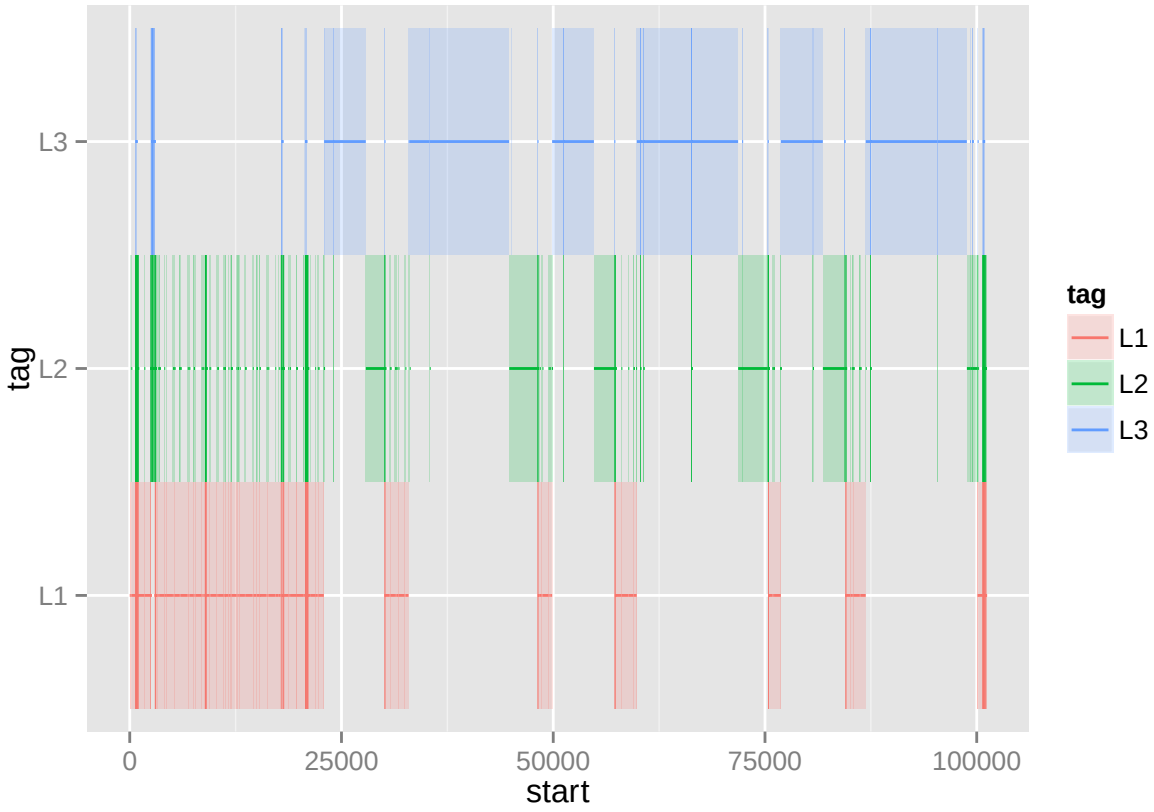
To avoid manipulating the whole series of points, a summary table is computed that reports:

- tag
- run ID
- length
- start position
- end position

```
## tag levels are those corresponding to peaks plus a special NOISE tag
tag.levels <- paste0("L",seq(dens$peaks))
data$tag <- factor(tag.levels[findInterval(data$P,c(0,thresholds))],c(tag.levels,"NOISE"))

## assign unique id to runs
data$runid <- cumsum(c(1,abs(diff(as.numeric(data$tag))!=0) ) )
```

```
## build run summary table
id.tab = subset(melt(with(data,table(tag,runid)),id.vars="runid",value.name="length"),length!=0)
id.tab$start=cumsum(c(1,head(id.tab$length,-1)))
id.tab$end=cumsum(id.tab$length)
```

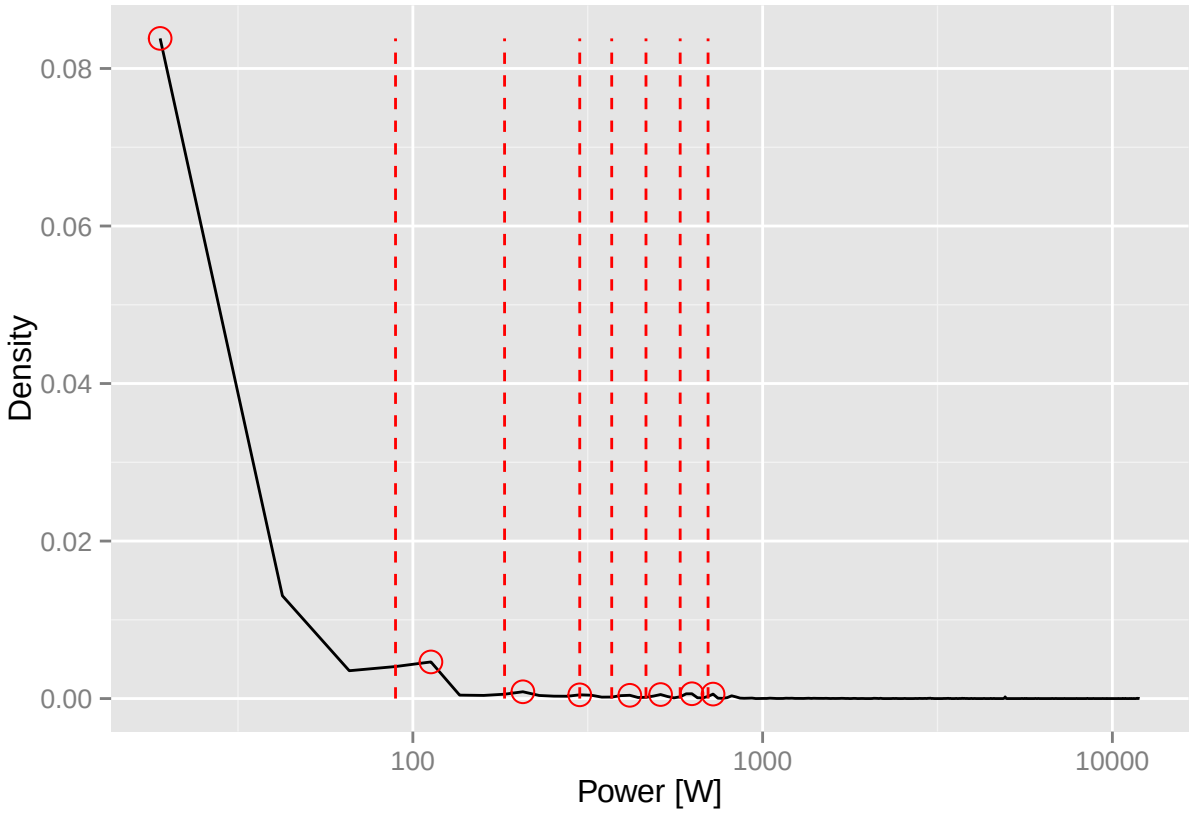




Denoise

The collected measures are subject to noise (as any measure is). Generally there are series of a few points that have a different value.

```
dens = density(id.tab$length)
dens$peaks <- which(peaks(dens$y) & dens$y>mean(dens$y))

## Identify thresholds between peaks as the points of minimum density between peaks
thresholds = dens$x[apply(cbind(head(dens$peaks,-1),tail(dens$peaks,-1)),1,function(x){
  ss = dens$y[x[1]:x[2]]
  x[1] + which(ss == min(ss))
})]
noise=thresholds[1]
```

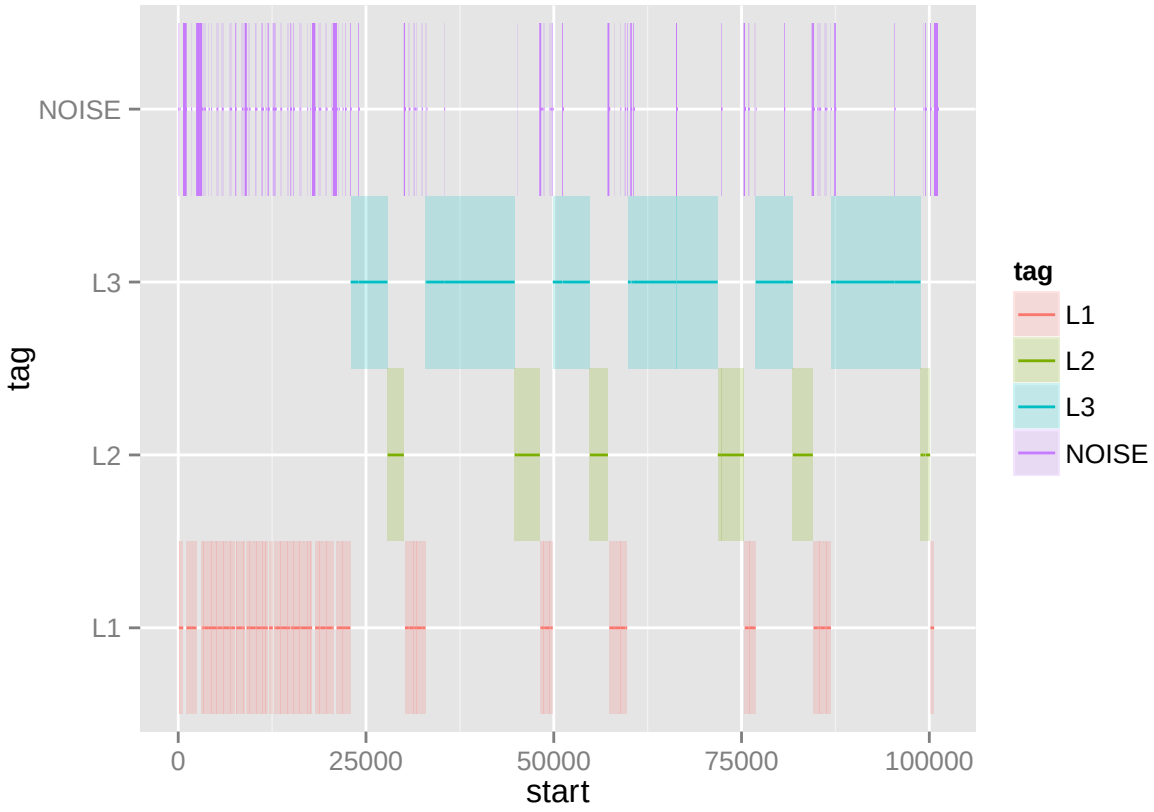


A possible suggested threshold for noise is: 89.2087986

In most cases the nois is a small variation around the expected value. Though, in some cases the noise is so large to cause a different tag to be assigned.

We have, first, to identify the noise using a `noise` threshold on the lenght of the runs.

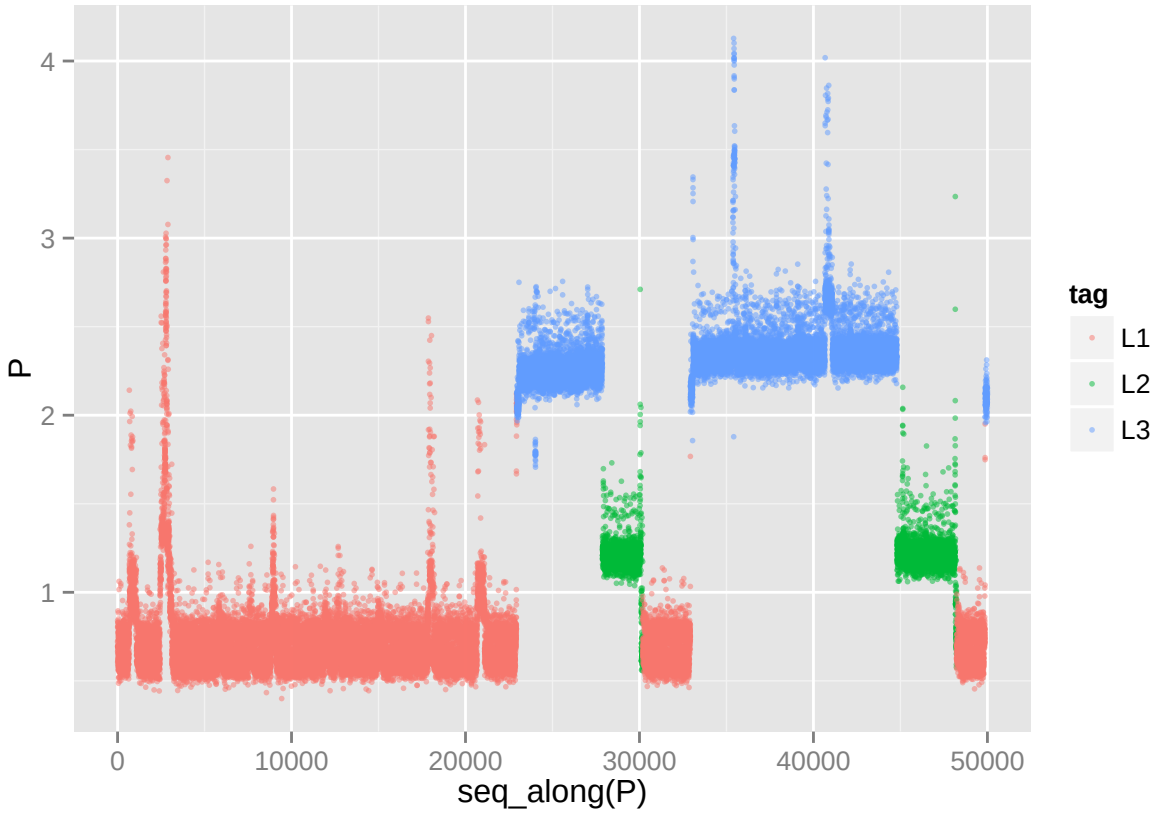
```
## identify noise runs by means of a run length threshold
id.tab$tag[id.tab$length<noise] = "NOISE"
```

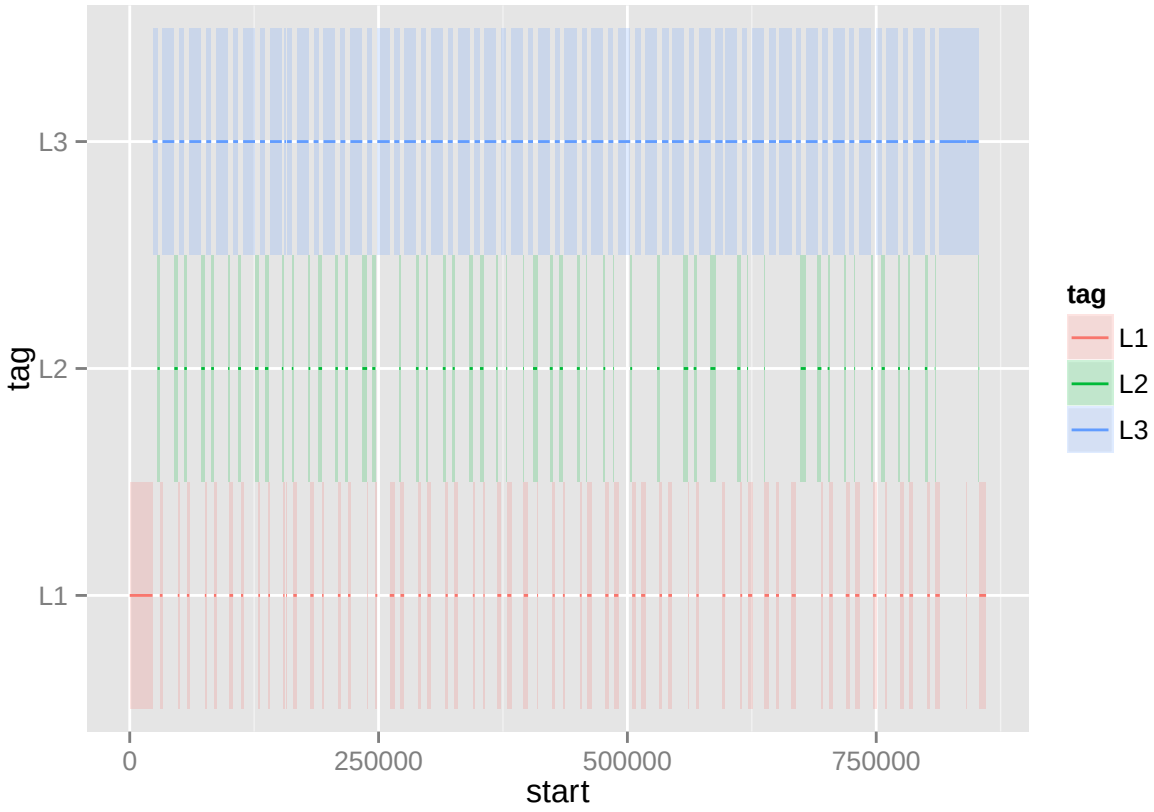


Then, we can remove the noise using some strategy:

- merge noise run with the preceding run
- merge the noise run with either the preceding or the following one, depending which is the largest.

```
## Remove noise
if(id.tab$tag[1]=="NOISE"){
  id.tab$tag[1] = id.tab$tag[min(id.tab$runid[id.tab$tag!="NOISE"])]
}
id.noise = which(id.tab$tag=="NOISE")
while(length(id.noise)>0){
  ##  assign the noise run the same level as the  the preceding run
  id.tab$tag[id.noise] = id.tab$tag[id.noise-1]
  id.noise = which(id.tab$tag=="NOISE")
}
## id.tab = id.tab[order(id.tab$runid),] ### probably not useful

## Merge consecutive runs havin the same tag
## a) recompute the run id
id.tab$runid = cumsum(c(1,abs(diff(as.numeric(id.tab$tag))!=0) ) )
## b) merge all the fragments with the same runid
id.tab = ddply(id.tab,.(runid,tag),summarize,
               length = sum(length),
               start = min(start),
               end = max(end)
               )
```


4
3
2
1
P
0
10000
20000
30000
40000
50000
seq_along(P)
tag
L1
L2
L3



runid	tag	length	start	end
1	L1	22947	1	22947
2	L3	4922	22948	27869
3	L2	2315	27870	30184
4	L1	2742	30185	32926
5	L3	11874	32927	44800
6	L2	3423	44801	48223
7	L1	1646	48224	49869
8	L3	4935	49870	54804
9	L2	2588	54805	57392
10	L1	2474	57393	59866

Identify task markers

Then we need to identify the marker that introduce the experiment.

We do not make any hypothesis as to the level at which markers are found (it should be the highest one...)

The markers are (a subset of) the runs whose lenght is close to the predefined marker length.

First we need to understand at which level (tag) are located the markers.

For each distinct level/run:

- Count how many runs are in the expected range (the interval centered around `marker.length` with emiwidth `marker.tolerance`).

- Compute the median period separating two consecutive runs
- Compute the standard deviation of the period separating two consecutive runs
- Select as marker level/tag the one with the number of potential markers, closer to the expected value (N) and with the smallest variability around of the period.

```
  l.min = marker.length*(1-marker.tolerance)
  l.max = marker.length*(1+marker.tolerance)
  markers = subset(id.tab,length>=l.min & length<=l.max)
  mark.sum = ddply(markers,.(tag),summarize,
        n=length(tag),
        length = median(length),
        t= median(diff(start)),
#         tv = sd(diff(start))
#         tv = 2*diff(quantile(diff(start),c(.25,.75)))
        tv = median(diff(start))*.1
        )
  mark.sum$score = dim(mark.sum)[1]*(rank(abs(mark.sum$n-N))-1)+rank(mark.sum$tv)-1
  marker = subset(mark.sum,score==min(score))
  marker.tag = marker$tag

  kable(mark.sum)
```

tag	n	length	t	tv	score
L1	3	4525.0	274076.5	27407.65	8
L2	4	4960.0	90771.0	9077.10	4
L3	26	4939.5	26959.0	2695.90	0

- the initial set of markers are the runs in the expected range
- the we proceed with all the markers trying to find the next one either in the existing set of markers or in the runs with the market tag

```
  potential.markers = subset(id.tab,tag==marker$tag)

  selected.markers = c()
  initial.markers = subset(markers,tag==marker$tag)
  print(initial.markers$runid)
```

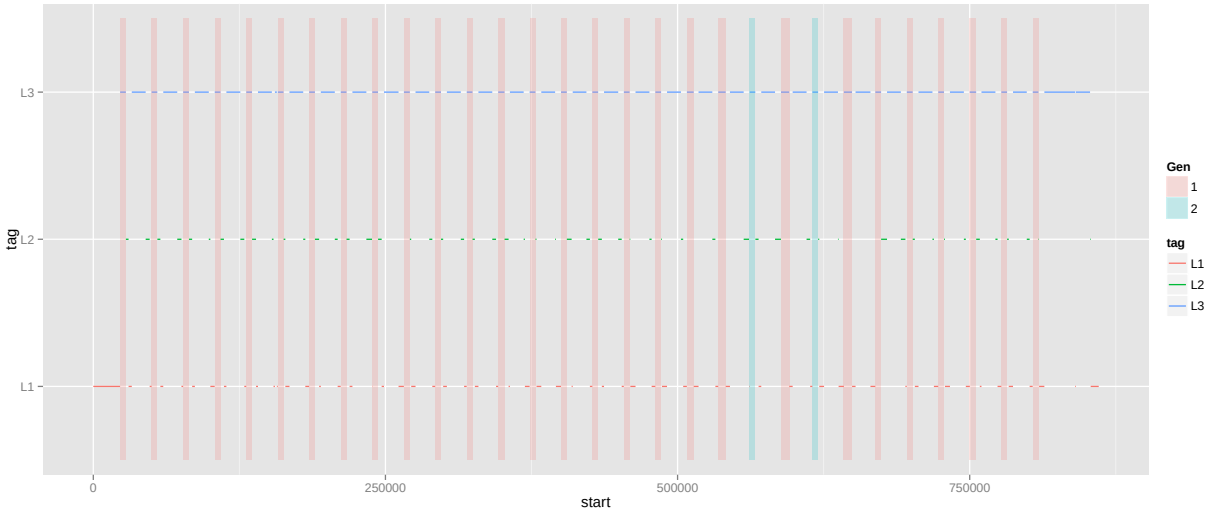
```
##  [1]   2   8  14  20  26  34  40  46  52  57  63  69  75  81  87  93  99
## [18] 105 121 131 141 146 152 158 164 170
```

```
  generation = 0
  while(!is.null(initial.markers)){
    generation <- generation + 1
    if(generation > 5) break
    new.markers=c()
    for(id in initial.markers$runid){
      #cat(id,"\n")
      current.marker = subset(potential.markers,runid==id)
      t.expected = current.marker$start + marker$t
```

```
      t.lower = t.expected - marker$tv
      t.upper = t.expected + marker$tv

      successor = subset(potential.markers,start>=t.lower & start<=t.upper & length>marker.length/2)
      if(dim(successor)[1]>0){
        if(dim(successor)[1]>1){
          successor <- subset(successor,abs(length-marker.length)==min(abs(length-marker.length)))
          #stop("No strategy to prioritize defined yet!!")
        }
        if(! successor$runid%in%selected.markers$runid){
          new.markers <- rbind(new.markers,successor)
        }
      }

      t.expected = current.marker$start - marker$t
      t.lower = t.expected - marker$tv
      t.upper = t.expected + marker$tv
      predecessor = subset(potential.markers,start>=t.lower & start<=t.upper& length>marker.length/2)
      if(dim(predecessor)[1]>0){
        if(dim(predecessor)[1]>1){
          predecessor <- subset(predecessor,abs(length-marker.length)==min(abs(length-marker.length)))
        }
        if(! predecessor$runid%in%selected.markers$runid){
          new.markers <- rbind(new.markers,predecessor)
        }
      }
    }
    new.markers = unique(new.markers)
    initial.markers=new.markers
    if(!is.null(new.markers)){
      print(new.markers$runid)

      new.markers$Gen = generation
      selected.markers=rbind(selected.markers,new.markers)
    }
  }
```

```
##  [1]   8  14   2  20  26  34  40  46  52  57  63  69  75  81  87  93  99
## [18] 105 111 126 116 137 146 152 141 158 164 170
## [1] 131 121
```

```
  selected.markers <- unique(selected.markers)
  selected.markers <- selected.markers[order(selected.markers$runid),]
```



The procedure identified 30 markers. The average length of markers is 5202.3666667, median: 4942, sd: 594.3077389.

	runid	tag	length	start	end	Gen
2	2	L3	4922	22948	27869	1
8	8	L3	4935	49870	54804	1
14	14	L3	4946	76886	81831	1
20	20	L3	4933	103900	108832	1
26	26	L3	4937	130909	135845	1
34	34	L3	4943	157859	162801	1
40	40	L3	4938	184709	189646	1
46	46	L3	4939	211727	216665	1
52	52	L3	4938	238747	243684	1
57	57	L3	5096	265785	270880	1
63	63	L3	4937	292589	297525	1
69	69	L3	4942	319547	324488	1
75	75	L3	5392	346489	351880	1
811	81	L3	4992	373369	378360	1
87	87	L3	4942	400259	405200	1
93	93	L3	5003	427060	432062	1
99	99	L3	4940	454008	458947	1
105	105	L3	4933	480985	485917	1
111	111	L3	5862	507947	513808	1
116	116	L3	6348	534817	541164	1
121	121	L3	4942	561628	566569	2
126	126	L3	7098	588477	595574	1
131	131	L3	4936	615369	620304	2
137	137	L3	7008	642299	649306	1
1411	141	L3	4935	669240	674174	1
146	146	L3	5053	696128	701180	1
152	152	L3	4944	723087	728030	1
158	158	L3	5481	750060	755540	1
164	164	L3	4937	777058	781994	1
170	170	L3	4919	803906	808824	1

Identify work

For each marker, identify the beginning and end of the work run.

Possible options:

1. **End + T *to* Start - T** (E2S): Skip a `marker.length` after marker *end*: that's the beginning of a work; stop `marker.length` before the next marker *start*

```
work = data.frame(
      start = head(selected.markers,-1)$end+marker.length,
      end = tail(selected.markers,-1)$start-marker.length
      )
work <- within(work,
       length <- end-start
       )
summary(work)
```

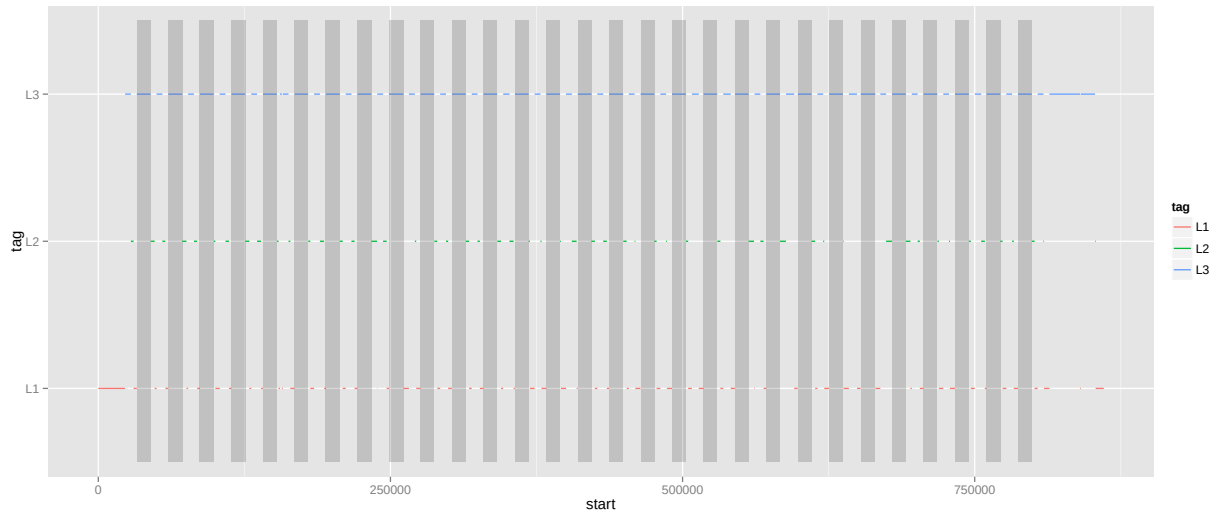
```
##      start             end             length
##  Min.   : 32869   Min.   : 44870   Min.   : 9795
##  1st Qu.:221665   1st Qu.:233747   1st Qu.:11860
##  Median :410200   Median :422060   Median :11954
##  Mean   :410341   Mean   :422059   Mean   :11718
##  3rd Qu.:600574   3rd Qu.:610369   3rd Qu.:12030
##  Max.   :786994   Max.   :798906   Max.   :12101
```

2. **Start + 2 T *to* Start - T** (S2S): Skip two `marker.length`s after marker *end*: that's the beginning of a work; stop `marker.length` before the next marker *start*

```
work.run = data.frame(
      start = head(selected.markers,-1)$start+2*marker.length,
      end = tail(selected.markers,-1)$start-marker.length
      )
work.run <- within(work.run,
       length <- end-start+1
       )
summary(work.run)
```

```
##      start             end             length
##  Min.   : 32948   Min.   : 44870   Min.   :11802
##  1st Qu.:221727   1st Qu.:233747   1st Qu.:11881
##  Median :410259   Median :422060   Median :11943
##  Mean   :410130   Mean   :422059   Mean   :11931
##  3rd Qu.:598477   3rd Qu.:610369   3rd Qu.:11978
##  Max.   :787058   Max.   :798906   Max.   :12039
```

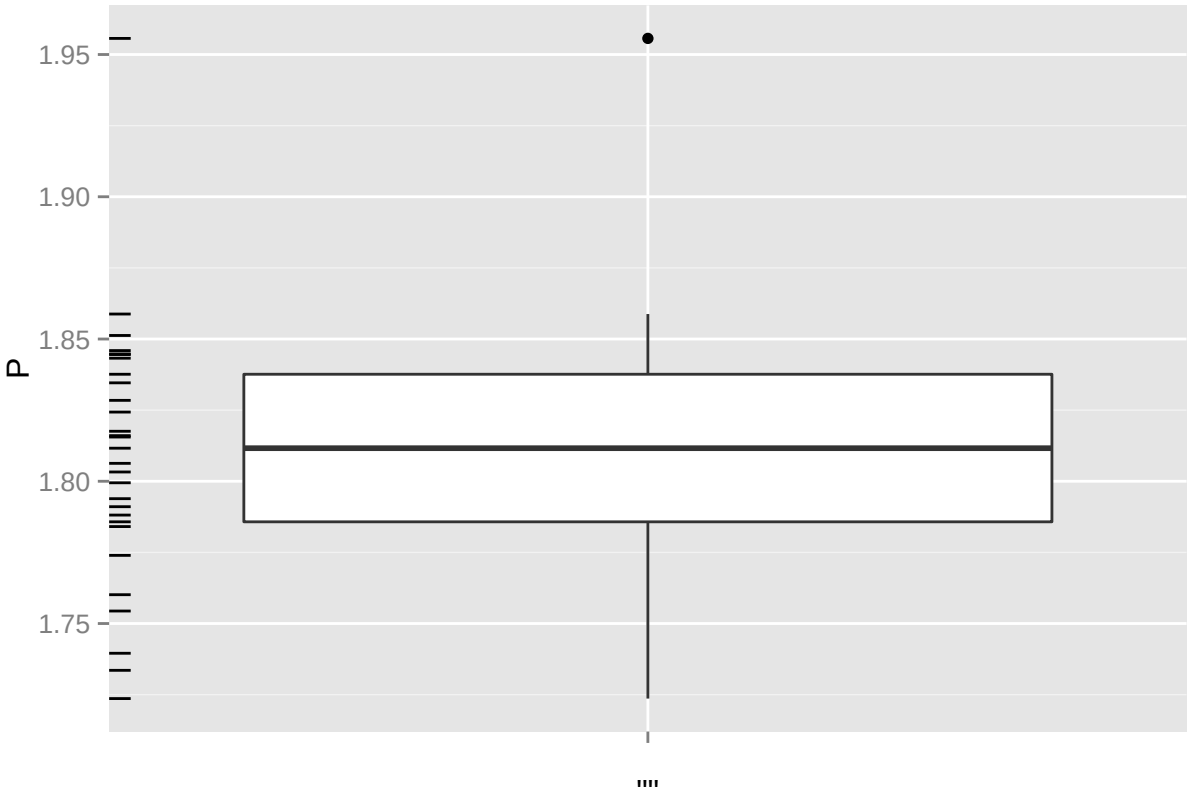
The second approach consistently gives better and more consistent results.



Extract work data

```
work.run$P <- apply(work.run,1,function(x)
                    mean(data[x["start"]:x["end"],]$P)-baseline)
```

```
p <- ggplot(data=work.run,aes(x='',y=P))+geom_boxplot()
p <- p+geom_segment(aes(yend=P),x=0.5,xend=0.52)
p
```


1.95
1.90
1.85
1.80
1.75
P
""