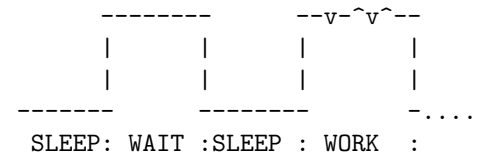
Analysis Procedure

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

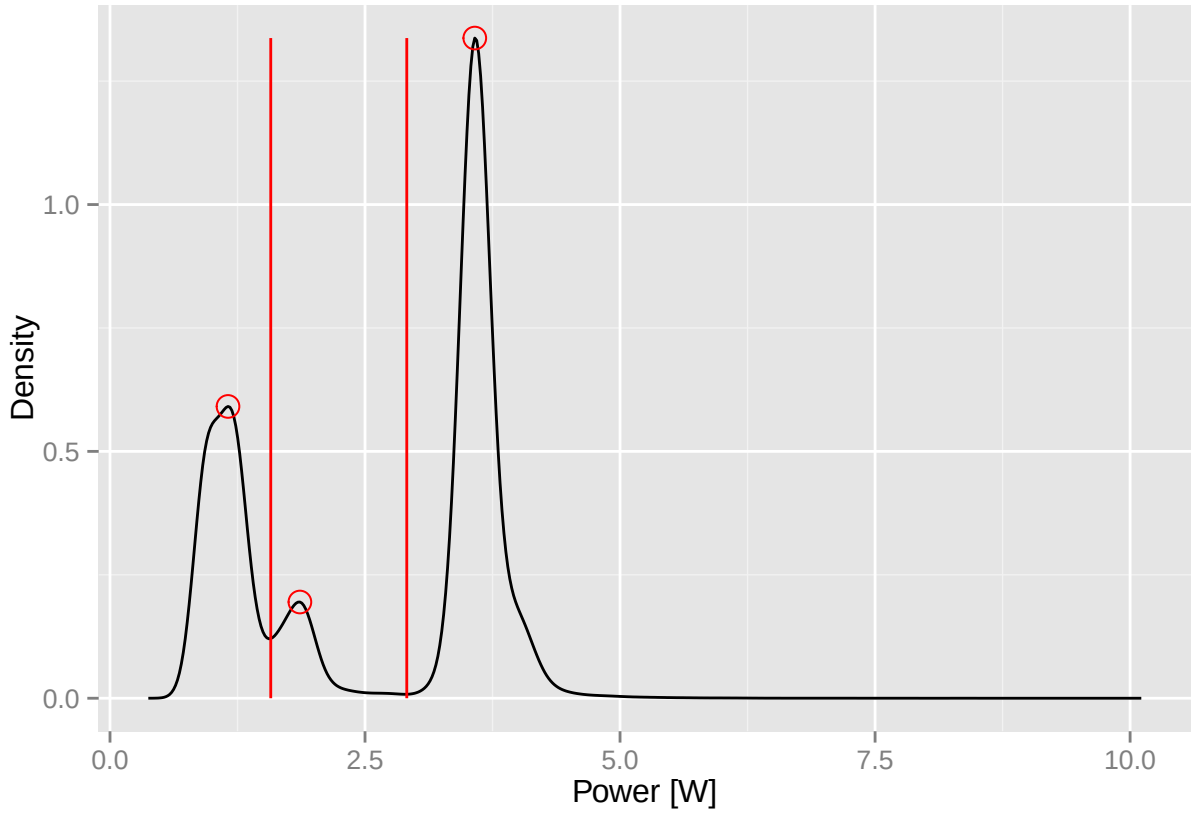
The algorithm takes the following input parameters:

```
adjust = 1.5 ## density parameter
marker.length = 5000
marker.tolerance= 0.02
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: 1.1566972 W.

Tagging

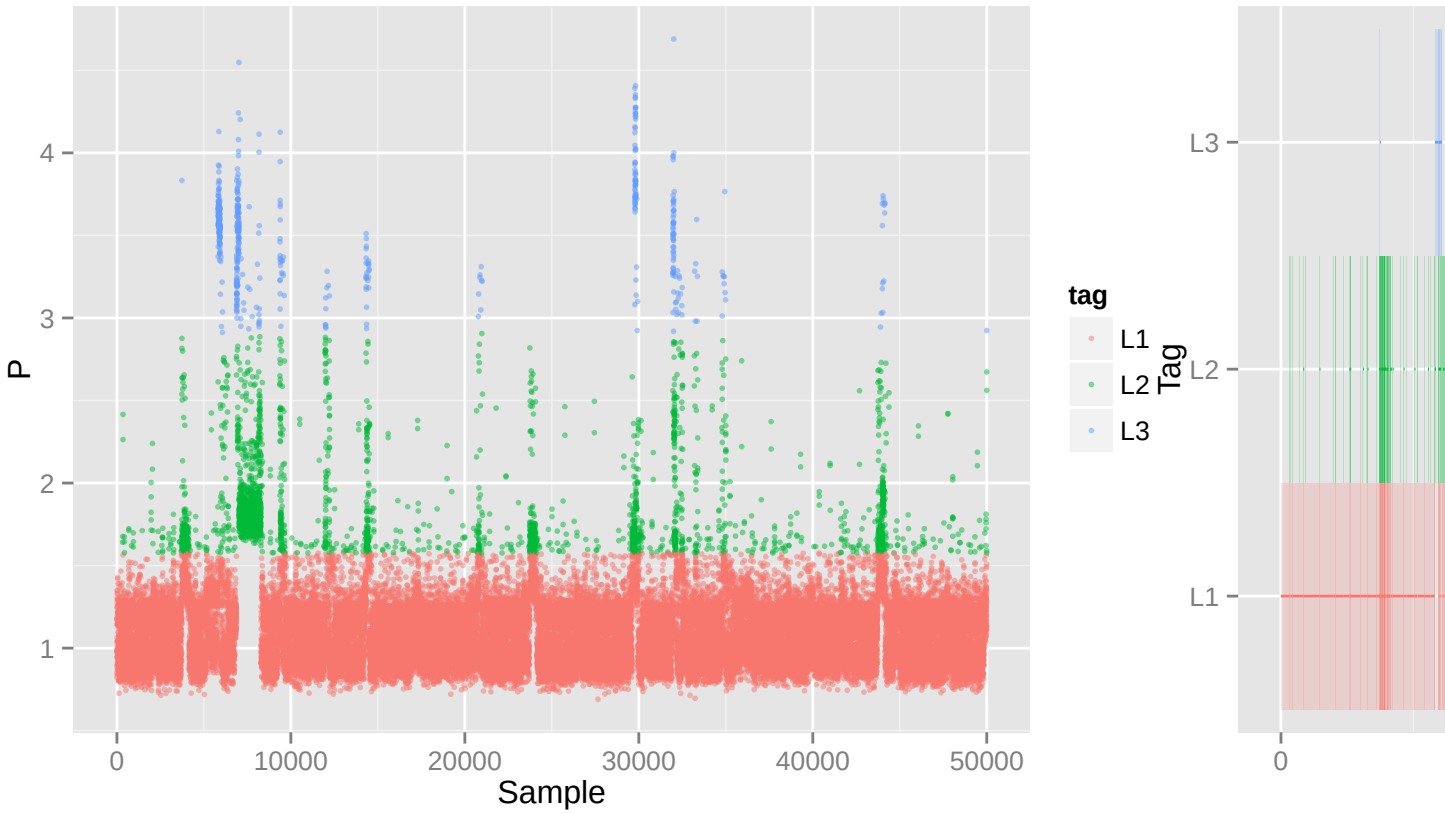
Each peak is assigned to a level, the the thresholds are used to discriminate between consecutive levels. The data points are tagged with the level name.

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

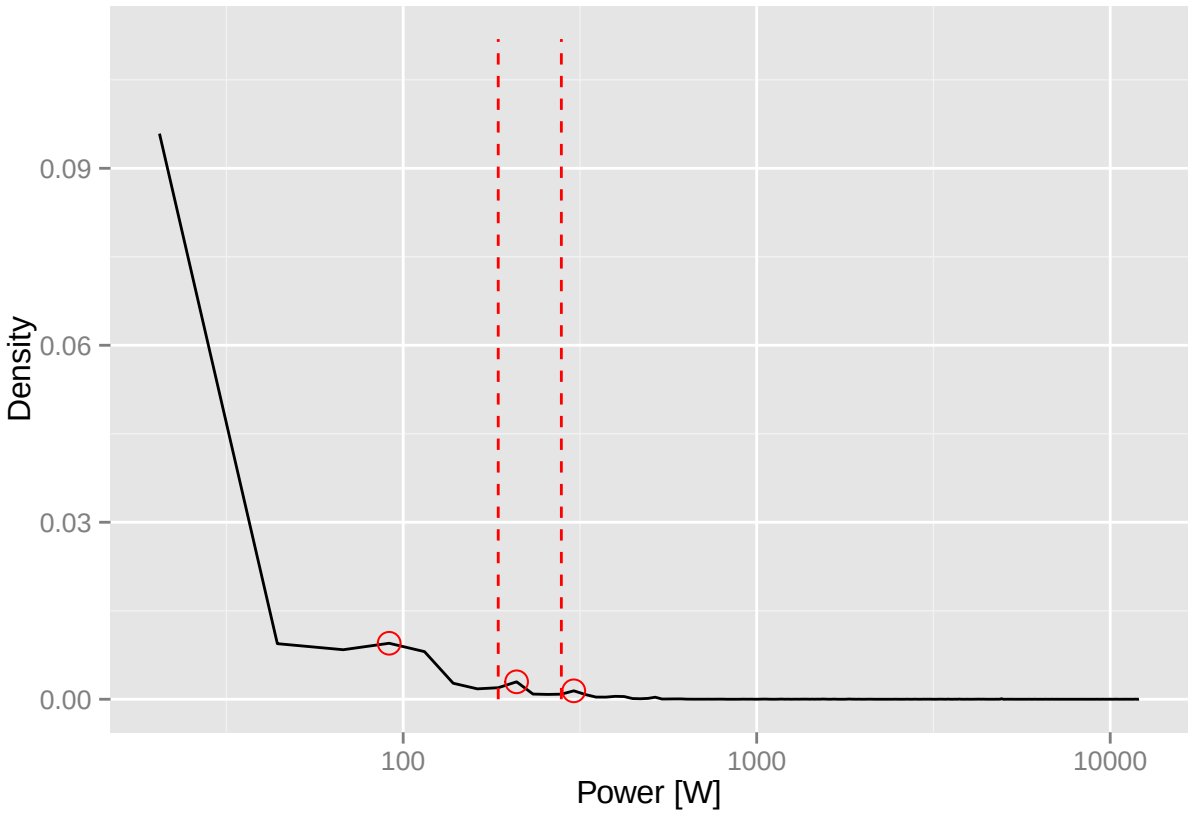
```
## tag levels are those corresponding to peaks plus a special NOISE tag
tag.levels <- paste0("L",seq(dens$peaks))
data$tag <- factor(tag.levels[findInterval(data$P,c(0,thresholds))],c(tag.levels,"NOISE"))

## 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]
```

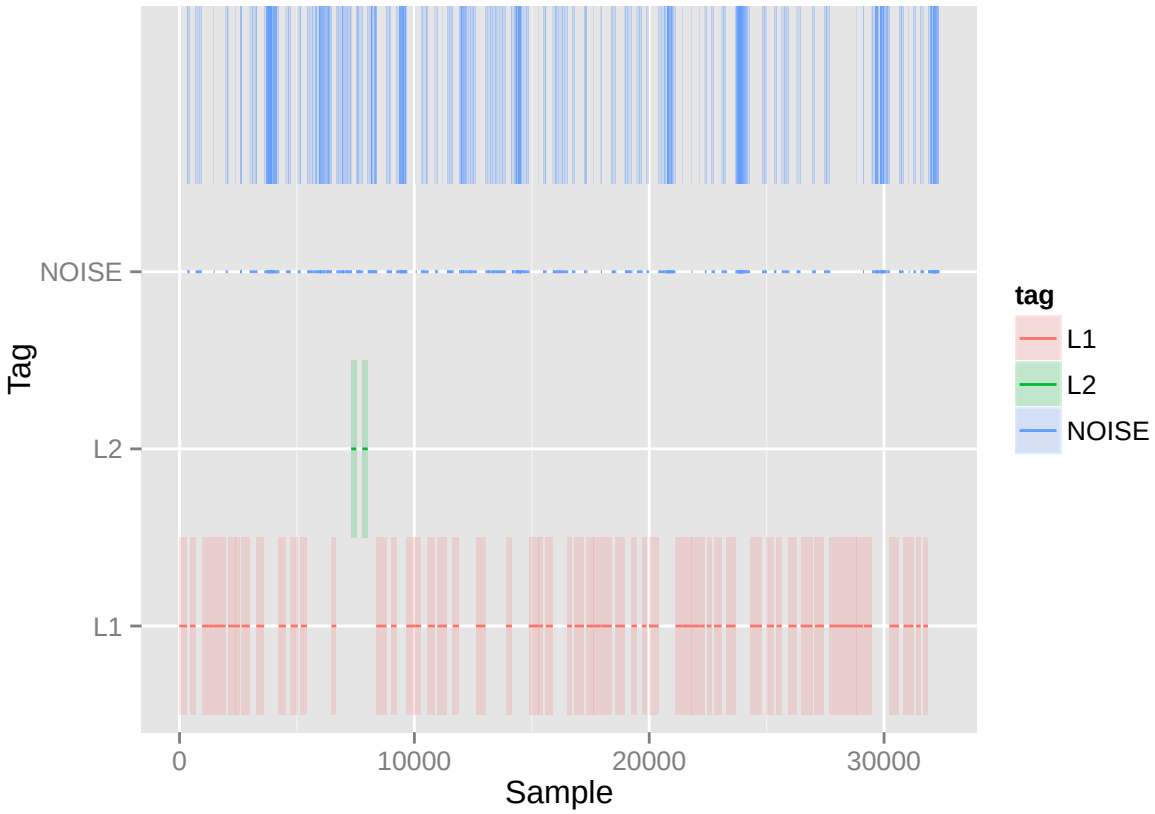


A possible suggested threshold for noise is: 185.7085706

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.

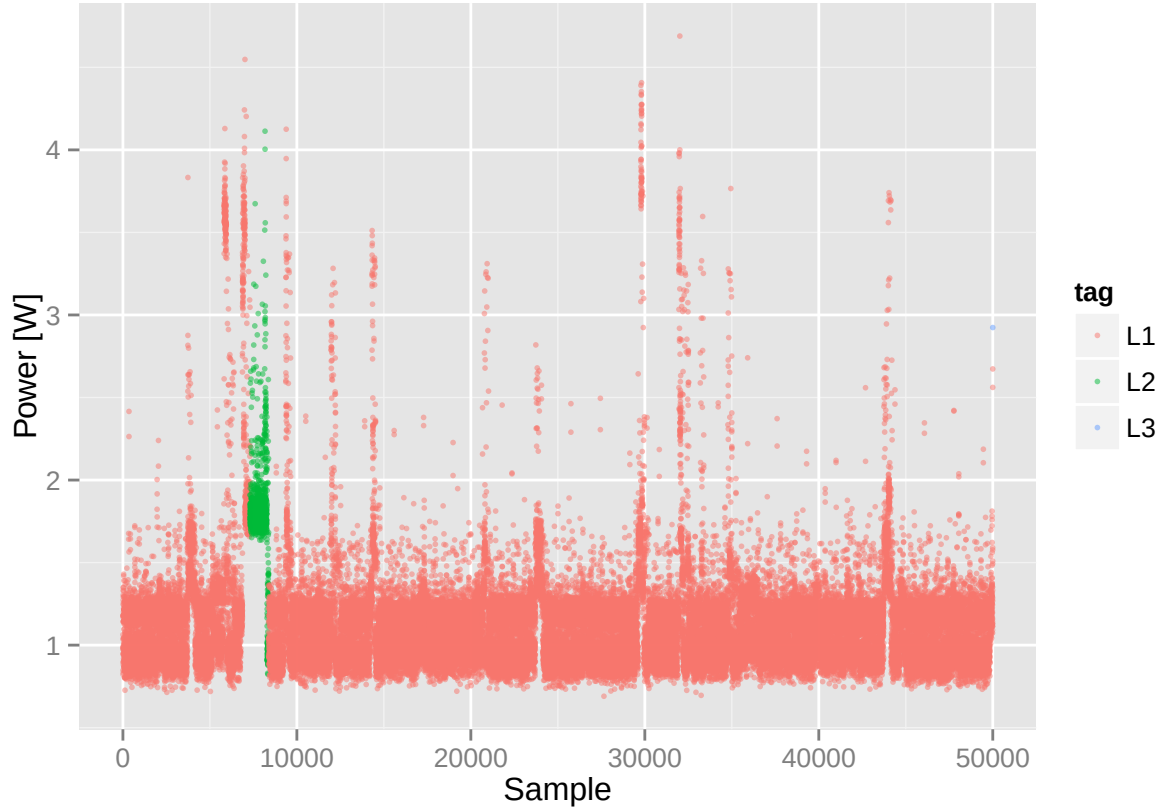
```
## identify noise runs by means of a run length threshold
id.tab$tag[id.tab$length<noise] = "NOISE"
```

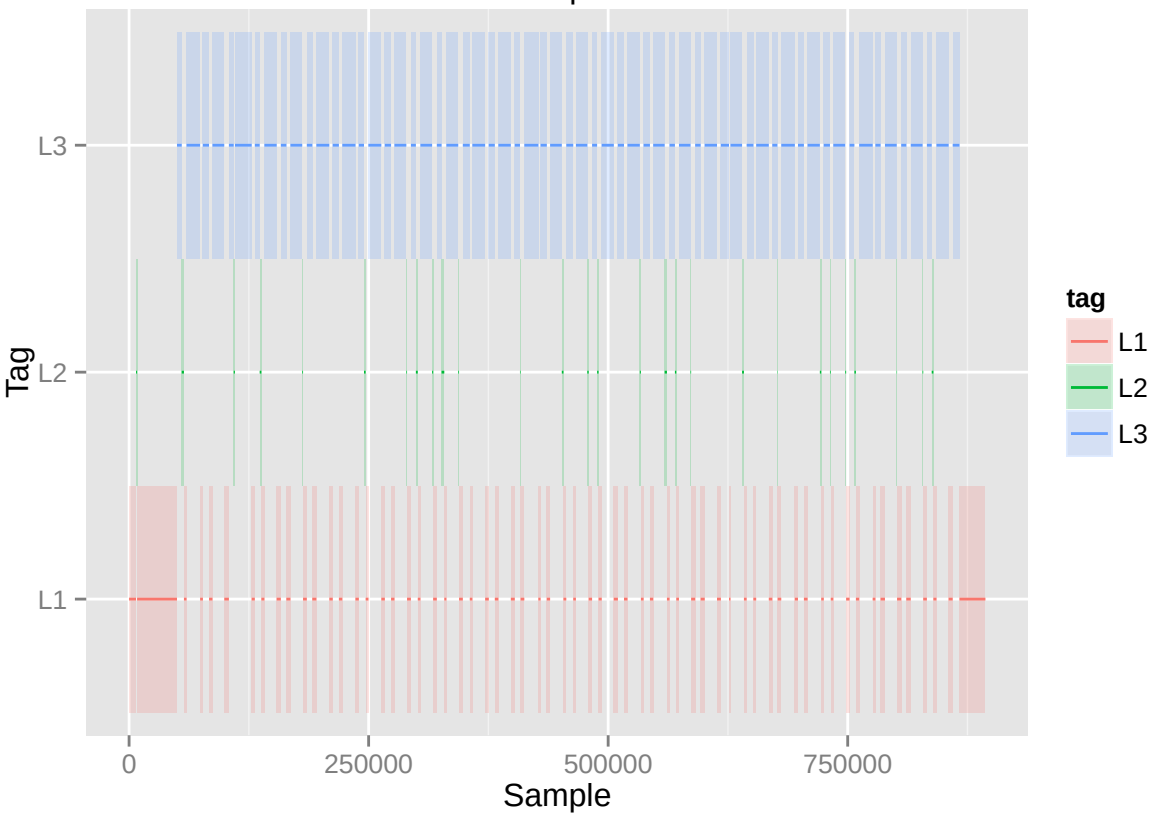


Then, we can remove the noise using some strategy:

- 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)
               )
```





runid	tag	length	start	end
1	L1	7305	1	7305

runid	tag	length	start	end
2	L2	1070	7306	8375
3	L1	41624	8376	49999
4	L3	4910	50000	54909
5	L2	2355	54910	57264
6	L1	2725	57265	59989
7	L3	14171	59990	74160
8	L1	2872	74161	77032
9	L3	6448	77033	83480
10	L1	3559	83481	87039

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))*.02
        )
  mark.sum$score = dim(mark.sum)[1]*(rank(abs(mark.sum$n-N))-1)+rank(mark.sum$tv)-1
  marker = subset(mark.sum,tag==max(tag.levels))
  marker.tag = marker$tag

  kable(mark.sum)
```

tag	n	length	t	tv	score
L3	6	5037.5	80680	1613.6	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)
```

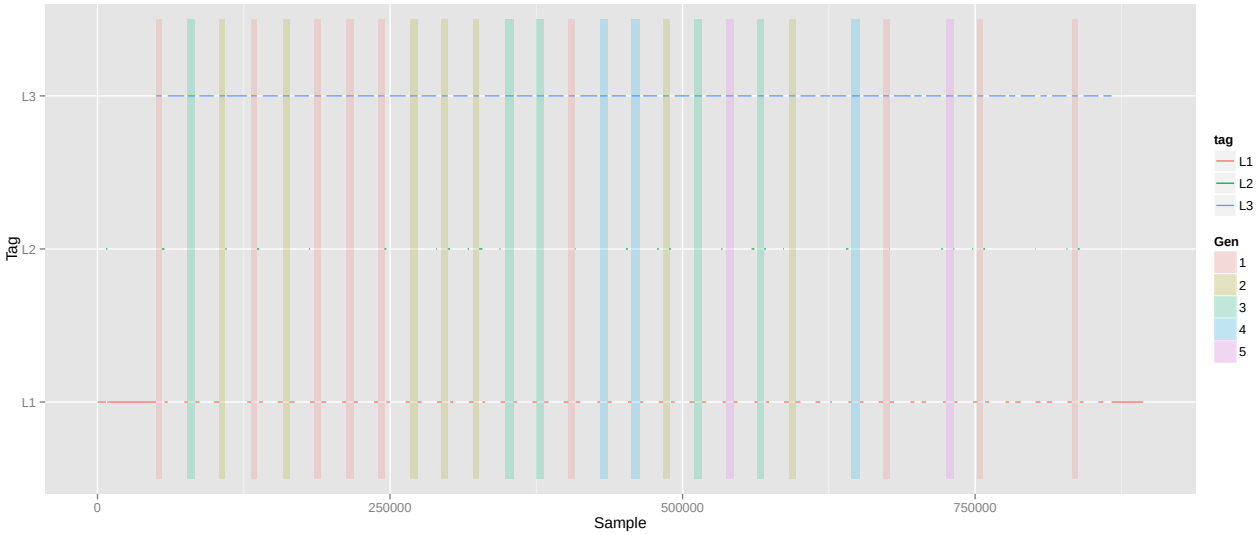
```
## [1]   4  13  17  51 130 145
```

```
  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]  17  27  31   4  65  35 145 114 130
## [1]  40  13  45  80  51  22 101
## [1] 57 61 95  9 85
## [1]  70  75 110
## [1]  90 124
```

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



The procedure identified 26 markers. The average length of markers is 5873.2692308, median: 5735, sd: 757.292364.

	runid	tag	length	start	end	Gen
4	4	L3	4910	50000	54909	1
9	9	L3	6448	77033	83480	3
13	13	L3	5037	104139	109175	2
17	17	L3	5083	131211	136293	1
22	22	L3	5798	158363	164160	2
27	27	L3	5738	185470	191207	1
31	31	L3	6414	212542	218955	1
35	35	L3	5398	239762	245159	1
40	40	L3	6530	266903	273432	2
45	45	L3	5188	294112	299299	2
51	51	L3	4953	321150	326102	2
57	57	L3	7268	348290	355557	3
61	61	L3	6281	375390	381670	3
65	65	L3	5446	402502	407947	1
70	70	L3	6512	429432	435943	4
75	75	L3	7278	456313	463590	4
80	80	L3	5218	483249	488466	2

	runid	tag	length	start	end	Gen
85	85	L3	6392	510141	516532	3
90	90	L3	6583	536983	543565	5
95	95	L3	5601	564060	569660	3
101	101	L3	5732	590792	596523	2
110	110	L3	7074	644492	651565	4
114	114	L3	5244	671302	676545	1
124	124	L3	6455	725003	731457	5
130	130	L3	5086	751932	757017	1
145	145	L3	5038	832612	837649	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.   : 59909   Min.   : 72033   Min.   : 9659
##  1st Qu.:223955   1st Qu.:234762   1st Qu.:10495
##  Median :386670   Median :397502   Median :11310
##  Mean   :392128   Mean   :407527   Mean   :15399
##  3rd Qu.:548565   3rd Qu.:559060   3rd Qu.:12036
##  Max.   :762017   Max.   :827612   Max.   :65595
```

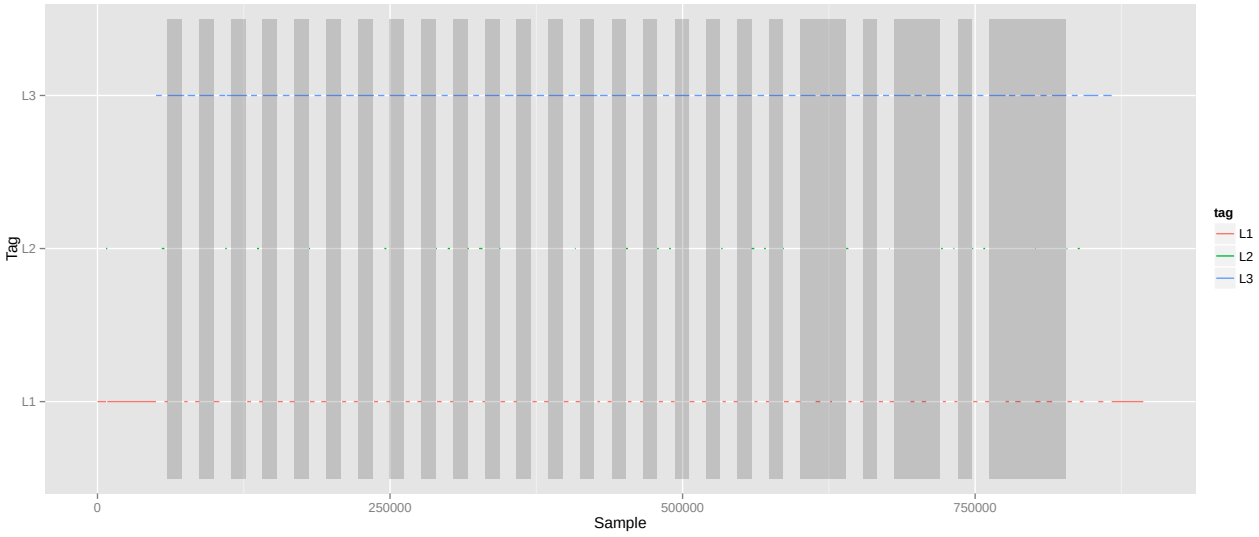
2. **Start + 2 T *to* Start - T** (S2S): Skip two `marker.length`s after marker *end*: that's the beginning of a work; stop `marker.length` before the next marker *start*

```
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.   : 60000   Min.   : 72033   Min.   :11733
##  1st Qu.:222542   1st Qu.:234762   1st Qu.:11931
##  Median :385390   Median :397502   Median :12078
##  Mean   :391223   Mean   :407527   Mean   :16305
##  3rd Qu.:546983   3rd Qu.:559060   3rd Qu.:12142
##  Max.   :761932   Max.   :827612   Max.   :65681
```

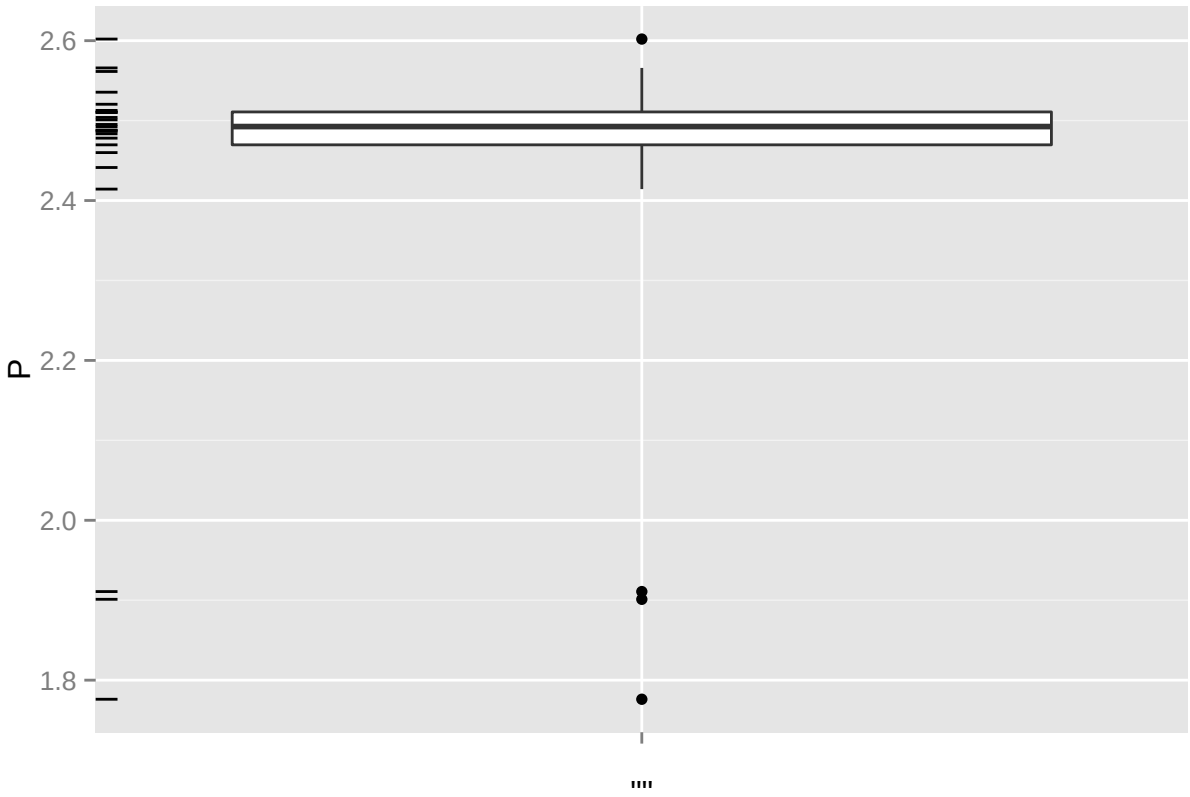
The second approach consistently gives better and more consistent results.



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
```


2.6
2.4
2.2
2.0
1.8
P
""