Assignment2

Quentin JONNEAUX - R00274704

2024-12-21

Introduction

In this report, we are helping a fuel company to give advice on the quality assurance strategy testing the fuel quality of each outlet. Fuel samples are collected from each stations and placed in trays that are placed in a conveyer for an automated testing process. The tray is passed in length through the machine that test the sample using a needle. This needle is aligned at the beginning of testing process but is subject to deviations after each test, which may result in test being inaccurate or machine becoming damaged, stopping the process and causing costly repairs. Engineers from the company were able to provide distributions and descriptives statistics for the deviations after observing the process.

We are tasked with analysing the distribution of needle alignments at the end of the day, through simulation. We will provide a scatterplot to visualise it and calculate the likelihood of the machine becoming innacurate or failing at any point during the day and define the distribution of the number of test that can be completed before machine become innacurate or misaligned.

After that, we are provided costing information for the quality testing process. We will conduct a simulation to find the best strategy for testing. We will find the optimal number of test to maximise profits and give the expected net profit for this optimal number.

Finally, we will give a summary and recommendations on our findings.

Insights provided by engineers and computing an end of day simulation

Thanks to the engineer, we are provided enough information to perform a Monte Carlo simulation for a batch. Let's see the details:

- Trays provided for testing are 20mm x 10 mm: the placement of the needle within the tray determine if the test is accurate or not. It is crucial to understand the zones where the machine becomes innacurate. If the needle is within 1.5mm of the tray edge, it results in an inaccurate reading. Therefore, if the needle deviates 8.5mm from the centre on the length axis ((20mm/2) - 1.5mm = 8.5mm) or if the needle deviates 3.5mm from the centre on the width axis ((10mm/2) - 1.5mm = 8.5mm), the test are innacurate. In addition, if the needle is misaligned by 10mm in length or 5mm in width (or more) from centre, then the needle is damaged resulting in machine failure.

- Angle of needle movement for each tray test ($\delta\theta$) follows a uniform distribution with a minimum of $-22.5o$ ($-\pi/8$ radians) and a maximum $67.5o$ ($3\pi/8$ radians), where $0o$ corresponds to the direction of conveyer movement (positive x-direction). We will be able to compute the value of an angular deviation for each sample, by passing the min and max value and use the distribution information.

- Distance of needle movement for each tray test (δr) follows a normal distribution with a mean of 0.05mm and a standard deviation of 0.13mm (negative values are permitted here and correspond to movements in the opposite direction). We will be able to compute the value of a needle movement deviation for each sample, by passing the mean and standard deviation and use the distribution information.

- We will use the formulae to convert the deviations from Circular Polar (r, θ) to Cartesian (x, y) coordinates:

  $\delta x = \delta r * \cos(\delta\theta)$

  $\delta y = \delta r * \sin(\delta\theta)$

Simulating 1000 times the process

Based on the information from engineers, we can compute a simulation:

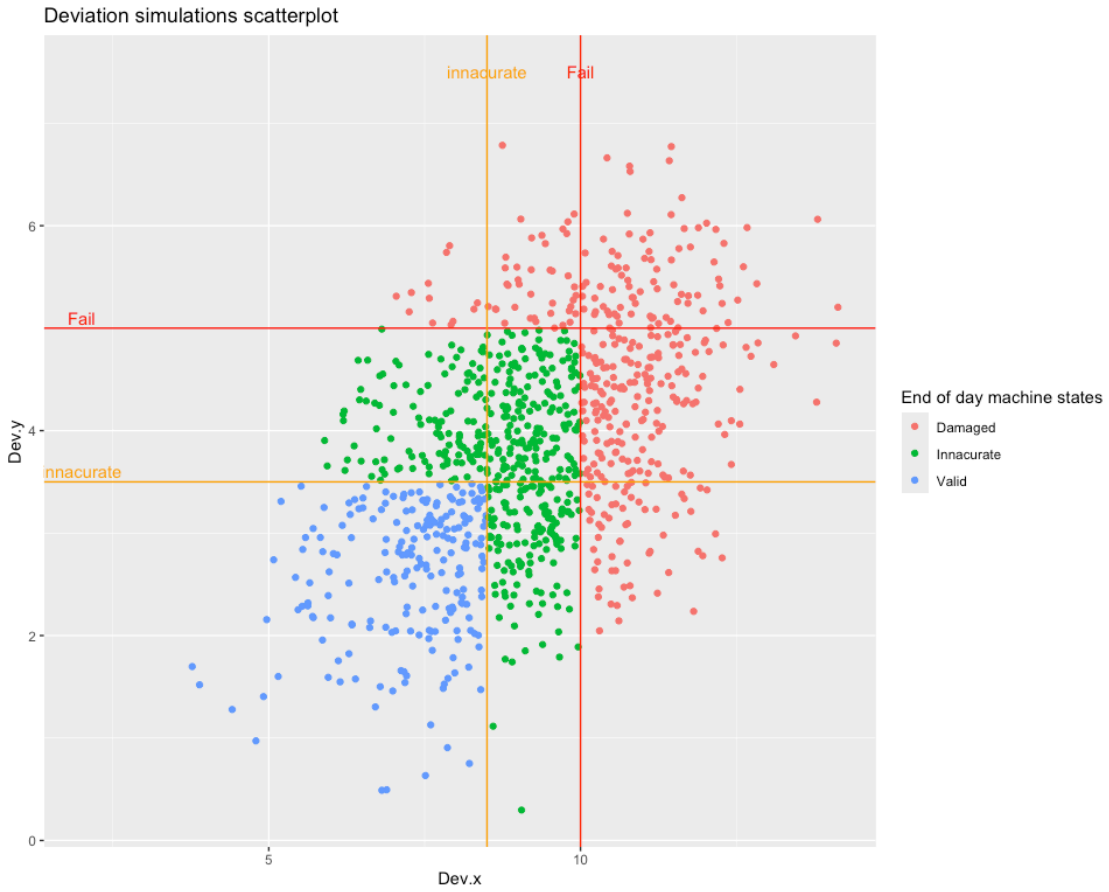
```
# Create empty vectors to store 1000 simulations of deviations of x and y
axis for each day
End.day.dev.x <- rep(NA,1000)
End.day.dev.y <- rep(NA,1000)

for(j in 1:1000){# run the random walk 1000 times

  align.x <- 0 # x aligned at start of day
  align.y <- 0 # y aligned at start of day
  # For a batch of 220 samples
  for(i in 1:220){
    # Apply a random deviation per distribution and angles provided
    delta.r <-rnorm(1,0.05, 0.13) # Store needle movement deviation
    delta.theta <- runif(1, (-pi/8), (3*pi/8)) # Store needle angular
deviation
    dev.x <- delta.r * cos(delta.theta) # random deviation in x direction
(applying formula to convert polar to cartesian coords)
    dev.y <- delta.r * sin(delta.theta) # random deviation in y direction
(applying formula to convert polar to cartesian coords)
    align.x <- align.x+dev.x # add this to previous steps - current alignment
x
    align.y <- align.y+dev.y # add this to previous steps - current alignment
y
  }
  # Store deviations for each dimensions in vectors
  End.day.dev.x[j] <- align.x
  End.day.dev.y[j] <- align.y
}

#create data frame for ggplot
dev.data <- data.frame(Dev.x = End.day.dev.x, Dev.y = End.day.dev.y)
```

Let's see the dataframe compiling the deviations:

First 5 rows devation data

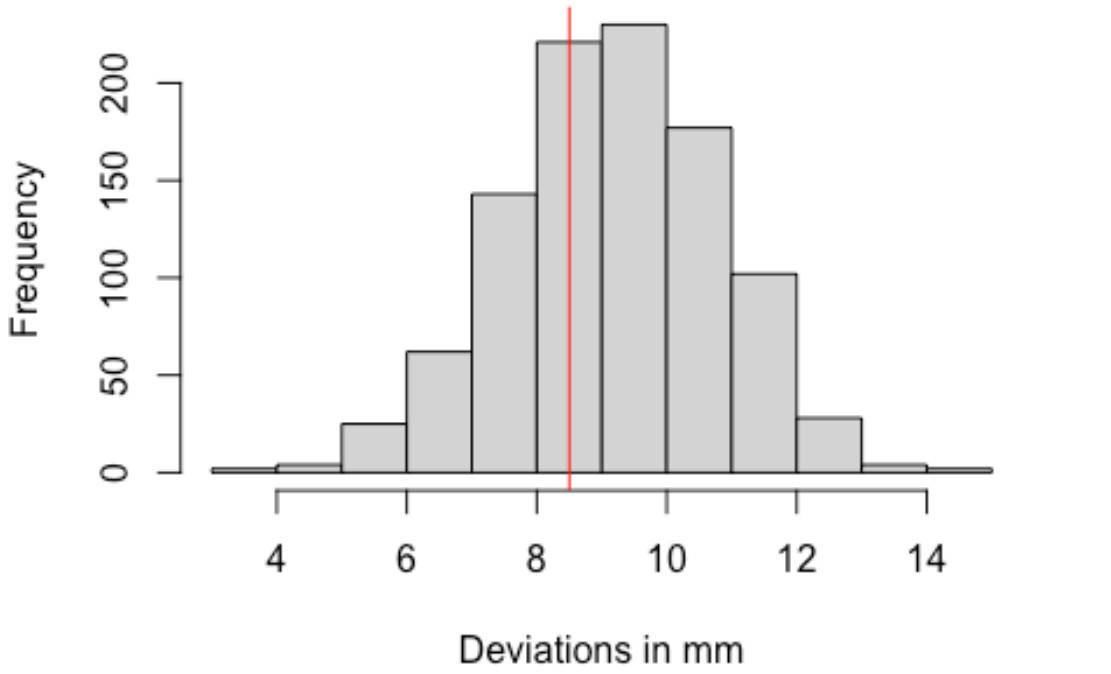
Dev.x	Dev.y
7.761997	3.937827
8.723921	2.890066
8.783807	5.048729
10.956991	5.356097
11.487300	4.327950

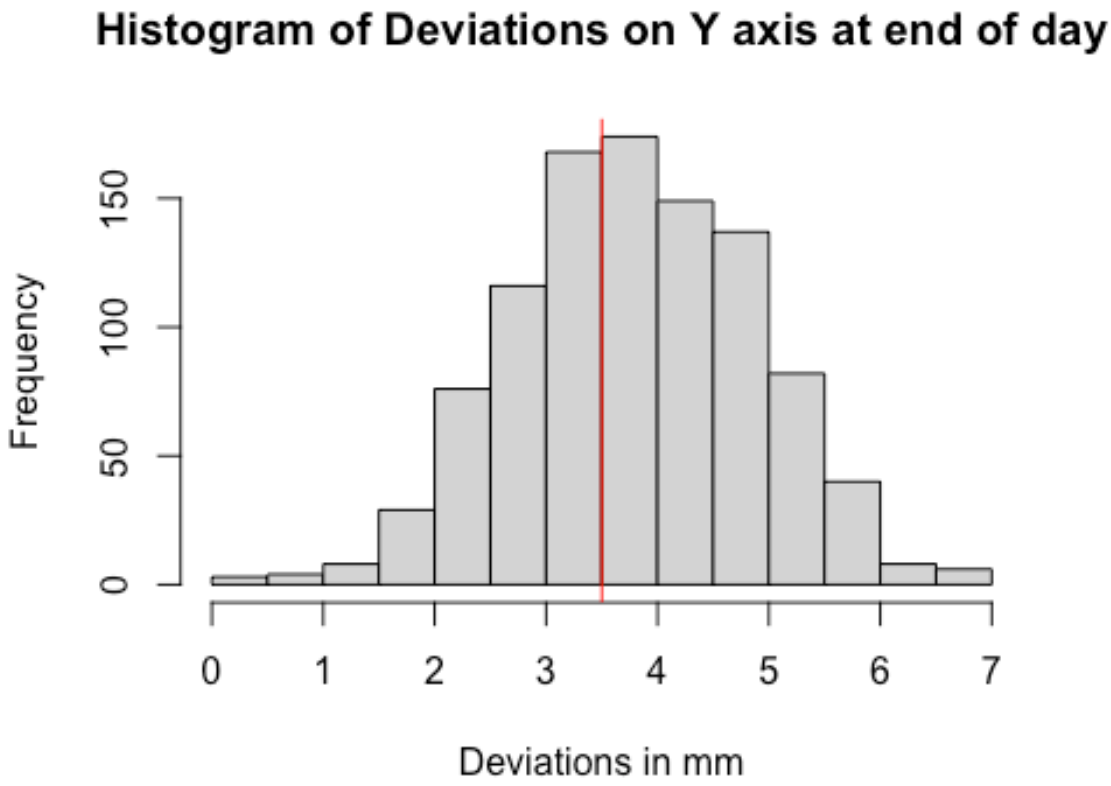
Here is a scatterplot of the needle alignment simulation at the end of a batch. We have higlighted the innacurate limits in orange and fail limits in red. The blue dots corresponds to a valid state at end of batch, the green dots to an innacurate state and the red dots to a damaged state.



We can see that there is a significant amount of batches that leaves the machine not valid by the end of the day. Only about 1/4 of batches leaves the machine in a valid state. Let's compute the likelihood of becoming not valid and visualise the distribution of the number of tests that can be completed before.

```
library(dplyr)
# Filter data for machine being inacurate or damaged
in.or.fail.data <- filter(dev.data,End.day.dev.x>=8.5 | End.day.dev.y>=3.5)
# Dividing count on simulation total
likelihood <- nrow(in.or.fail.data)/1000
likelihood

## [1] 0.793
```


Histogram of Deviations on X axis at end of day
Frequency
0
50
100
150
200
4
6
8
10
12
14
Deviations in mm



After simulating the batch 1000 times, the likelihood of the machine becoming damaged or innacurate at any point during the day is 0.793. We can also see that distribution for both deviations in x and y look normal as there is a bell shape in the frequency distribution. they are a little symmetrical passed the inaccurate limit of both lines, meaning more than half of cases are likely to be inaccurate or fail. That correlates to the likelihood we computed.

Simulating profits

Now that we can simulate, we can assess the strategy for profits. We are given the following information:

- The machine goes offline (excessive misalignment, no further batches can be tested for the rest of the day) costs €10,000
- An inaccurate batch test needs to be retested at an additional cost of €300.
- If a batch is ready for testing but the machine is offline, there is an average cost of €500 for storage and/or alternate testing of each untested batch under the target number of tests per day.
- Each sample successfully tested results in a positive value of €200.

Let's create a function we can use to replicate in the simulation. The function will return a vector with the number of runs that can be done until machine becomes innacurate and when it becomes damaged:

```
#function to just calculate how many runs it takes before alignment is out of
spec
# if it remains within spec, function return the value n+1
when.outside <- function(n){#simulate if/when misalignment occurs
  delta.r <-rnorm(n,0.05, 0.13) # Store needle movement deviation
  delta.theta <- runif(n, (-pi/8), (3*pi/8)) # Store needle angular deviation
  dev.x <- delta.r * cos(delta.theta) # random deviation in x direction
(applying formula to convert polar to cartesian coords)
  dev.y <- delta.r * sin(delta.theta) # random deviation in y direction
(applying formula to convert polar to cartesian coords)
  # Store cumulative sums of each deviations
  cum.dev.x <- cumsum(dev.x)
  cum.dev.y<- cumsum(dev.y)

  index <- 1:n
  # Store cumulative sum of test when machine becomes inacurate (if 221,
always passing)
  nretest.dev.x <-
ifelse(is.na(index[cum.dev.x>=8.5][1]),n+1,index[cum.dev.x>=8.5][1])
  nretest.dev.y <-
ifelse(is.na(index[cum.dev.y>=3.5][1]),n+1,index[cum.dev.y>=3.5][1])
  nretest <- ifelse(nretest.dev.x >= nretest.dev.y, nretest.dev.y,
nretest.dev.x)
  # Store cumulative sum of test when machine becomes damaged (if 221, always
passing)
  nfail.dev.x <-
ifelse(is.na(index[cum.dev.x>=10][1]),n+1,index[cum.dev.x>=10][1])
  nfail.dev.y <-
ifelse(is.na(index[cum.dev.y>=5][1]),n+1,index[cum.dev.y>=5][1])
```

```
  nfail <- ifelse(nfail.dev.x >= nfail.dev.y, nfail.dev.y, nfail.dev.x)

  # return vectors of number of tests when inacurate and when damaged
  nums <- c(nretest,nfail)
}
```

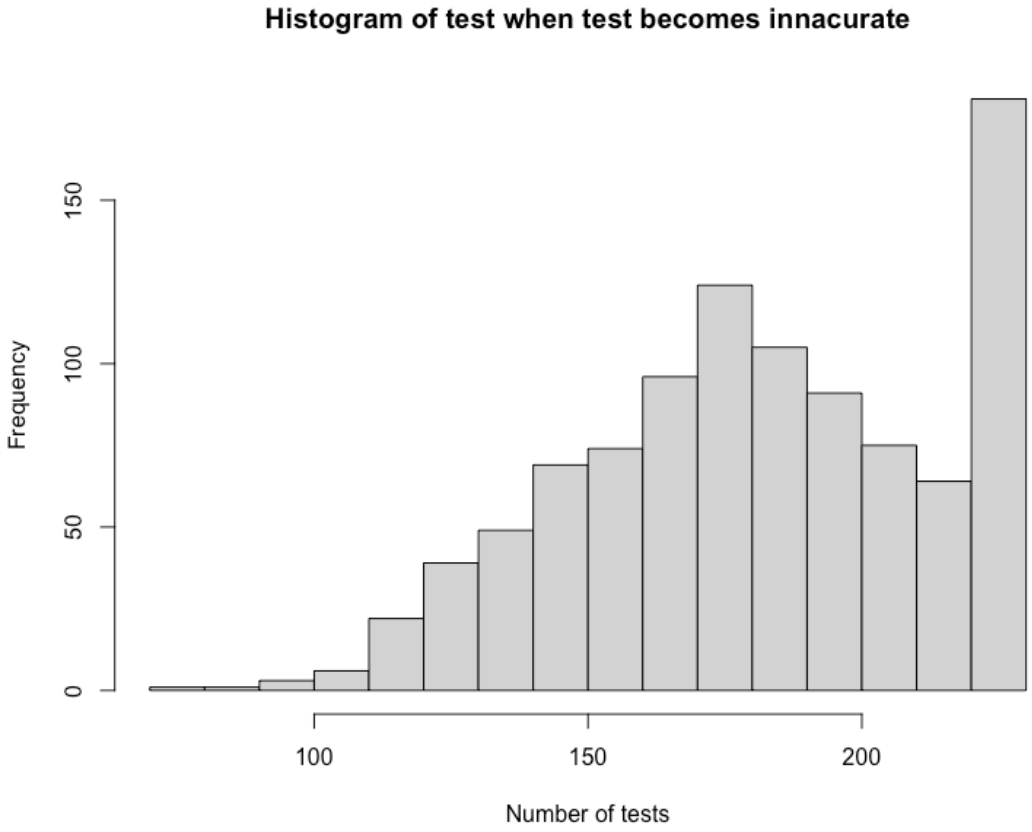
Let's do a 1000 simulations a create a data frame with the result (a value of 221 means the machine does not become inaccurate and/or damaged):

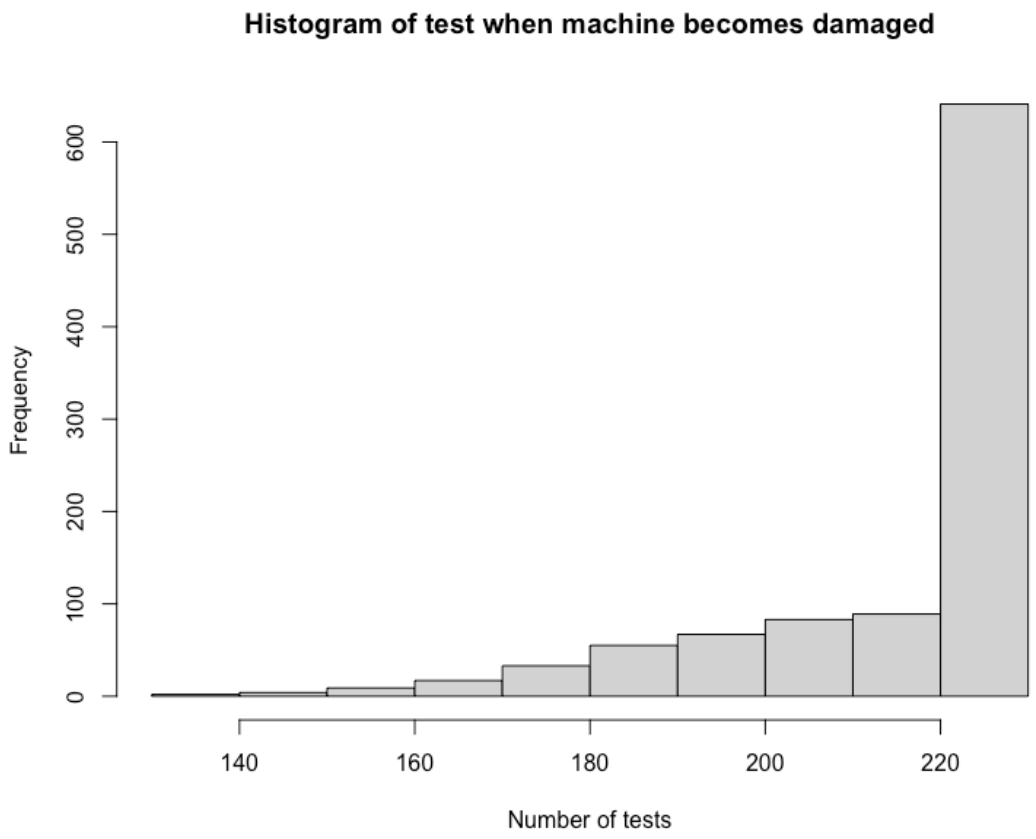
```
# Replicate 1000 for simulation
sim <- replicate(1000, when.outside(220))
#Store each vector
retest.row <-sim[1,]
fail.row <-sim[2,]
# Create a data frame with those vector
sim2 <-data_frame(retest=numeric(1000),fail=numeric(1000))
sim2$retest <- retest.row
sim2$fail <- fail.row
```

First 5 rows devation data

retest	fail
203	221
143	221
209	221
181	221
154	201

Let’s visualise the distributions of the innacurate and fail data:





We can see that the inaccurate distribution is bell shaped around the 170-180 bin and a much higher peak in the last bin (where the machine never becomes inaccurate). The fail distribution is left skewed with a much higher peak in the last bin (where the machine never becomes damaged). We can assume therefore machine is not very likely to fail but there is optimal value to work around for inaccurate data.

Let's compute the likelihood of each:

```
retest.likelihood <-sum(sim2$retest<221)/1000
fail.likelihood <-sum(sim2$fail<221)/1000
retest.likelihood

## [1] 0.819

fail.likelihood

## [1] 0.359
```

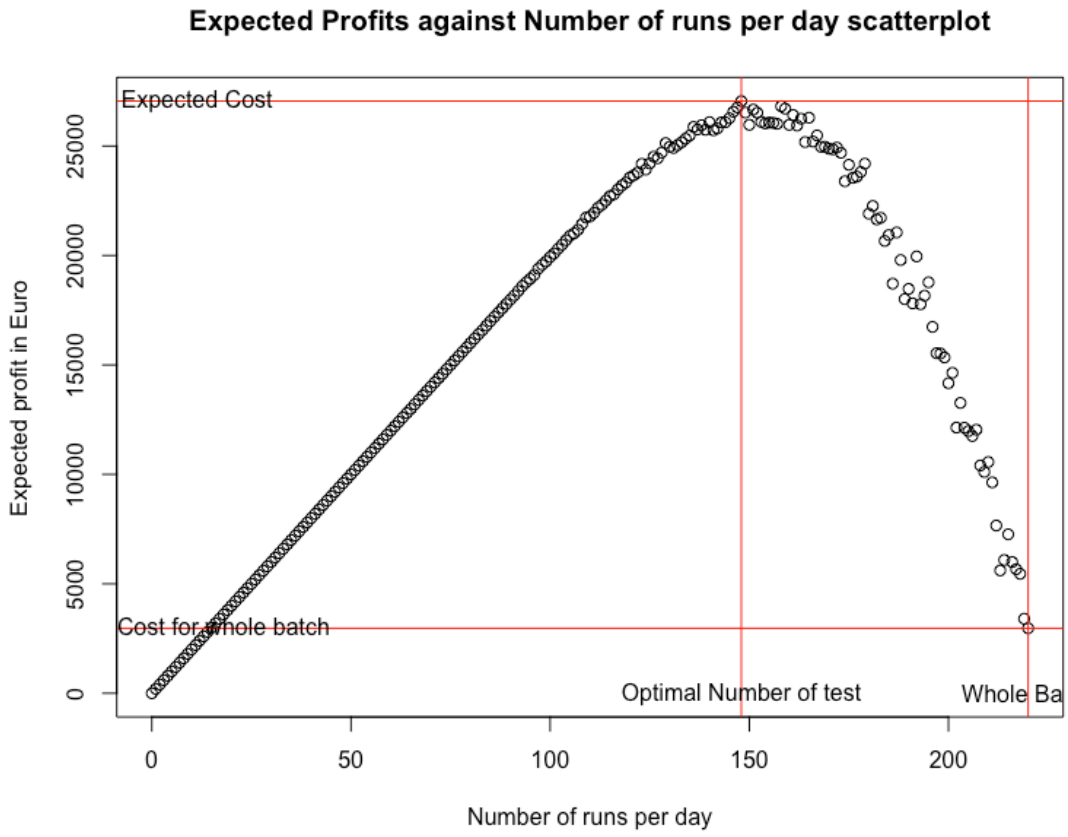
According to the simulation, the probabilities of the machine becoming inaccurate is 0.819 and failing is 0.359. Now we have an understanding of the distribution, let's create a function calculating the profits based on the results of the simulation and run it for a batch of 220 samples. The function will return the cost/profit mean:

```
profit <- function(n){#function to simulate profit/loss
  ofline.cost <- 10000
  run.profit <- 200
  run.retest <- 300
  run.loss <- 500
  sim <- replicate(1000, when.outside(n))#simulate 1000 times
  # Create a cost vector with profit calculation depending on scenario
  costvec <- ifelse(sim[1,]<=n,
                    ifelse(sim[2,]<=n,
                           run.profit*sim[1,]-run.retest*((n-sim[1,])-(n-
sim[2,]))-run.loss*(n-sim[1,])-ofline.cost,
                           run.profit*sim[1,]-run.retest*(n-sim[1,])-
run.loss*(n-sim[1,])),
                    n*run.profit)
  mean(costvec)#calculate mean profit over all simulations
}

# Expected profits for simulation
pr.220 <- profit(220)
pr.220

## [1] 4820.5
```

The expected cost for running 220 samples in one batch is 4820.5. It is profitable, but let's see how to maximise profits.

Let's perform profits simulations for batches ranging from 0 to 220 samples per batch and visualise its distribution.

```
#run this expected mean profit simulation for various values if target n
N=0:220
ECOST = sapply(N, profit)
```



We can see a parabolic distribution, meaning we can identify the maximum expected profit (around 26000) associated with an optimal number of runs (around 153). The profits are increasing until this number of runs and then decrease passed it. Let's compute the value to have these values.

```
# Expected Max Profit
Ex.max.profit <- max(ECOST)
# Optimal number of runs per day
Opt.num.runs <- N[ECOST==max(ECOST)]
Ex.max.profit
```

```
## [1] 27053.2
```

```
Opt.num.runs
```

```
## [1] 148
```

The maximum expected profit is 27053.2, associated with an optimal number 148 runs.

Summary

According to this 1000 simulation of a run of 220 samples batches:

- Running 220 samples batches is profitable
- The distributions of needle alignments at the end of each batch seem concentrated in inaccurate or fail values.
- The likelihood that the machine fails or become inaccurate at any point during the day is 0.793.
- Number of tests distributions seem normal on the x axis (length) and y axis (width).
- The distribution of expected profits against number of runs is parabolic. Expected profits rise until optimal number is reached, then drops significantly.
- The maximum expected profit is 27053.2
- The optimal number of runs per day before realignment is 148.

Profits can therefore increase significantly if the strategy is adjusted around maximum number.

Recommendations:

- Run the tests in batches 148 samples.
- Combine the remainder of current batch with the next batch until 148 is scheduled.
- It is less profitable to run a very small batch (1-30) samples than a whole batch of 220. Distribute runs to avoid batches that are less than 30 samples.