POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

# Quality Data Analysis - Project Work [FULL PROJECT]

Project Report of Quality Data Analysis - Mathematical Engineering

**Team number:** 1
**Team members:** Giulia Mezzadri, Federico Angelo Mor, Abylaikhan Orynbassar, Federica Rena
**Academic year:** 2023-2024
**Professor:** Panagiotis Tsiamyrtzis

# Phase 1

## 1.1. Introduction 3.5k chars

Ensuring product quality and operational efficiency is critical in modern manufacturing processes as the ability to detect defects in real-time during production, known as in-line detection, is essential for maintaining high standards and minimizing waste. For this purpose, 40 Voronoi filters were printed using HP Jet Fusion 580 Color 3D printer, to be used as sample to propose a method for the in-line detection of defects. Printed filters were placed on ten trays, each containing four filters, to take a top view gray-scale image of the objects at MADE Competence center.

The purpose of the project was to create a robust statistical process monitoring (SPC) method specifically for the in-line detection of defects. It consisted of two phases: in the first we were given objects without any defects to develop our statistical process control method, while the second phase involved testing our proposed method to identify objects with various defects. We used the Shewhart univariate control chart and multivariate control chart to design our SPC method, as they provide a visual and statistical method for monitoring process stability and detecting abnormal variation, helping organizations maintain consistent quality and identify opportunities for process improvement. In the following sections, we will detail the project's methodology, results, and implications, offering insights into how this innovative SPC method can be adopted across various industries to drive quality improvements and operational excellence.

## 1.2. Assumptions and preliminary data analysis 5k chars

## Data manipulation

Once completed the image acquisition we ran the python script provided on webeep to generate the images' statistics, obtaining the first dataset, which we will refer to as `df_old`.

At that initial stage the unique "key" which identified each of the four objects inside each image was the combination of the `Image Name` and `Position` variables, so we opted to combine them into a single identifier, `Part_ID`, using a simple function. This way we could more easily refer to single objects in the subsequent analysis.

The python script also included a segmentation part, which automatically isolated the objects by cropping a region around them. However, the crop was quite large and the objects inside were tilted, so we decided to implement our own code to do the segmentation. We deemed this step necessary since we thought that to compare more fairly the statistics of the images we needed a "common ground" (that is, perfect framing) on which they should be computed.

To perform this correction we binarized the images using Otsu treshold, detected the borders using a Canny filter, selected the main outer edge, and from that we derived the parameters for the proper rotation matrix and crop region (see Figure 1). After this refinement we re-computed their statistics and we obtained the dataset `df_new`.

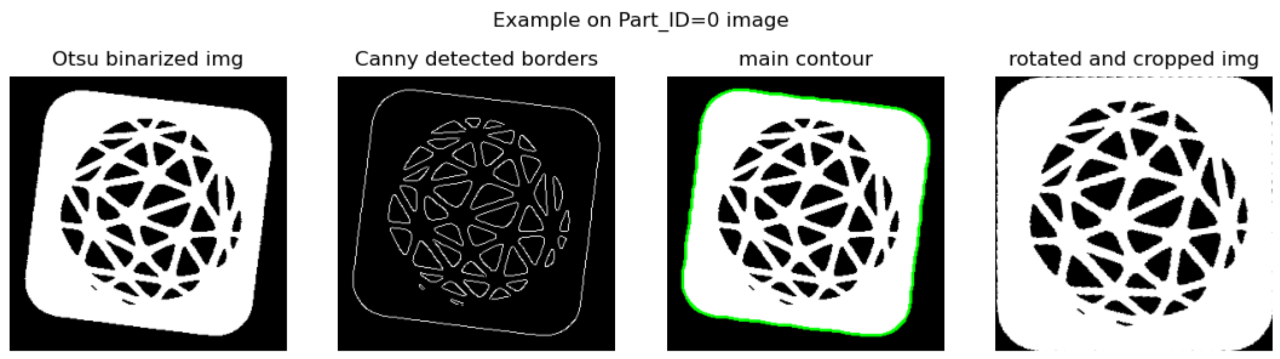


Figure 1: Visualization of the steps of our segmentation algorithm.

Before continuing with the analysis, we also deemed appropriate to split both old and new datasets into a *part* subset and a *void* subset, since they are constitutionally different features (the full object versus the small holes inside it), and we also saw a clear separation in the distribution of the variables, as depicted in Figure 2 and confirmed by ANOVA tests about the difference of the variables in the void and part groups, of which we report the pvalues of the tests in Table 1.

| Variable Name | Pvalue of ANOVA test |
|---|---|
| Area [pixels] | 0.0 |
| Perimeter [pixels] | 0.0 |
| Eccentricity | $4.3781482 \cdot 10^{-146}$ |
| Orientation [radians] | $4.6979078 \cdot 10^{-8}$ |
| Solidity | $2.8502201 \cdot 10^{-157}$ |
| Extent | $2.4072606 \cdot 10^{-23}$ |
| Major Axis Length [pixels] | 0.0 |
| Minor Axis Length [pixels] | 0.0 |
| Equivalent Diameter [pixels] | 0.0 |

Table 1: Pvalues of the ANOVA test, on each variable generated from the original python script, using the `df_new` dataset. The null hypothesis is that, for a certain variable $i$, the two groups have the same population mean.

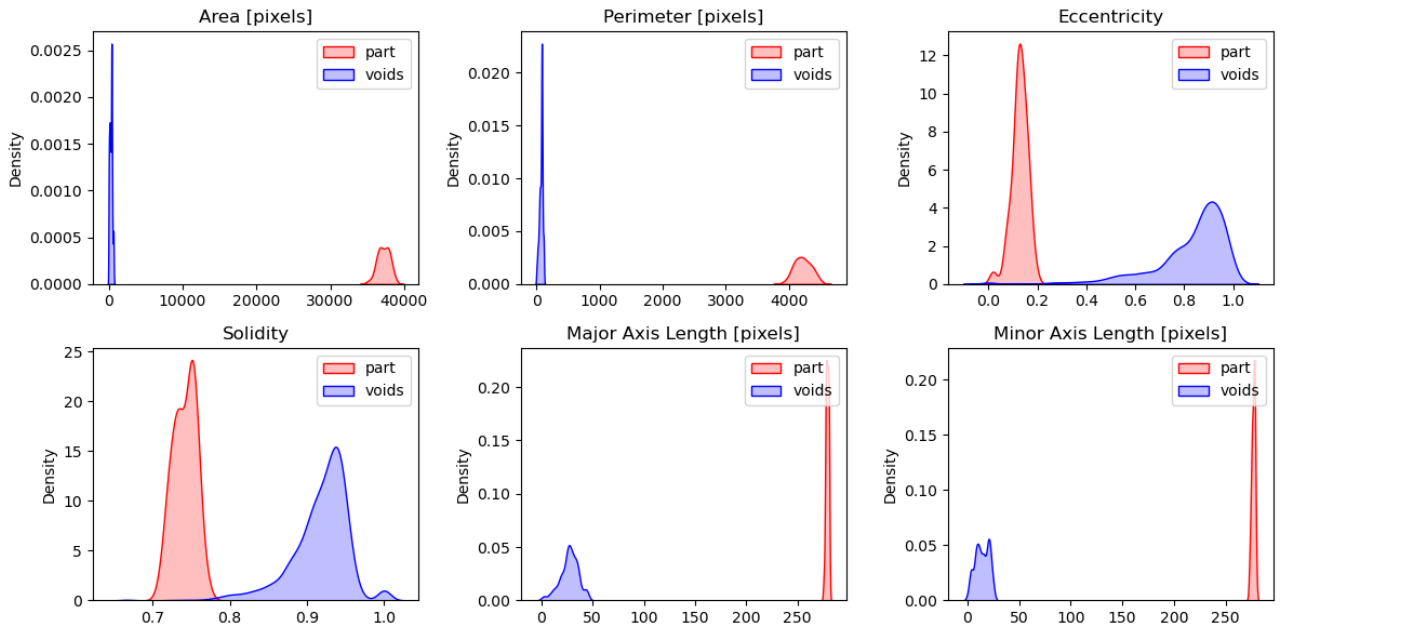


Figure 2: Visual comparison of the approximated distributions between parts and voids, for some variables of the `df_new` dataset. Similar results were also present for the `df_old` one.

## Dataset comparison

The two datasets, the old and the new one, seemed to provide different values, so we investigated this difference through nonparametric intervals (Boostrap t-intervals) and the respective tests. In this preliminary phase of our analysis we did not assume normality: that's why we chose to use a nonparametric approach to perform the comparison. We studied separately the `df_part` dataset and the `df_void` one.

About `df_parts` we tested, for each variable $i$,

$$H_0 : \mathbb{E}(\texttt{df\_old\_parts}[i]) = \mathbb{E}(\texttt{df\_new\_parts}[i]) \quad \text{vs} \quad H_1 : H_0^c$$

and we obtained the following confidence intervals (where `Voids number` is a new variable that we quickly created store how many voids each object had):

| Variable name | Lower CI | Point Estimate | Upper CI | $0 \in$ CI? |
|---|---|---|---|---|
| Area [pixels] | -1539.53 | -1452.3 | -1362.32 | no |
| Perimeter [pixels] | 87.112 | 96.9639 | 105.541 | no |
| Eccentricity | -0.0022843 | 0.0025 | 0.00738961 | yes |
| Orientation [radians] | -0.456515 | -0.057225 | 0.0647833 | yes |
| Solidity | -0.0227009 | -0.02125 | -0.0197547 | no |
| Extent | 0.00990142 | 0.017275 | 0.029109 | no |
| Major Axis Length [pixels] | 1.02045 | 1.1686 | 1.3052 | no |
| Minor Axis Length [pixels] | 0.914555 | 1.03865 | 1.15556 | no |
| Equivalent Diameter [pixels] | -4.4616 | -4.20975 | -3.94027 | no |
| Voids number | -0.550939 | -0.2 | -0.0646082 | no |

Table 2: Confidence intervals for `df_old_parts` against `df_new_parts` variables.

As the majority of intervals do not include zero, we can confidently reject the null hypothesis in most cases with a confidence level of 95%, which means there is a substantial difference between `df_parts` and `df_new_parts`.

Regarding the `df_void` dataset the same test was conducted, albeit with a variation: in this case, we considered 40 distinct datasets, each corresponding to a different image (since originally we had ten images of trays, each of which with four objects) in order to compare the different variables related to the same image. So for each variable $i$ referred to the image $j$ we tested

$$H_0 : \mathbb{E}(\texttt{df\_old\_voids}^{(j)}[i]) = \mathbb{E}(\texttt{df\_new\_voids}^{(j)}[i]) \quad \text{vs} \quad H_1 : H_0^c$$

and we obtained the following intervals (for the sake of example we show only the ones related to the first image, i.e. `Part_ID` $= 0$):

| Variable name | Lower CI | Point Estimate | Upper CI |
|---|---|---|---|
| Area [pixels] | -27.3227 | 18.8049 | 63.3162 |
| Perimeter [pixels] | -3.93578 | 3.15383 | 9.72439 |
| Eccentricity | -0.0363188 | -0.00497561 | 0.0276549 |
| Orientation [radians] | -0.462142 | -0.0247073 | 0.382791 |
| Solidity | -0.0273247 | -0.0168537 | -0.00623376 |
| Extent | -0.0241511 | 0.00826829 | 0.0376097 |
| Major Axis Length [pixels] | -1.79098 | 0.688024 | 2.91621 |
| Minor Axis Length [pixels] | -1.0699 | 0.578268 | 2.20956 |
| Equivalent Diameter [pixels] | -1.0602 | 0.667683 | 2.34608 |

Table 3: Confidence intervals for the difference of variables between `df_old_voids` and `df_new_voids`; reported for `Part_ID` $= 0$.

Almost all the intervals seem to include the zero and so we cannot reject the null hypothesis in the majority of the cases.

These tests show how there is a substantial difference between `df_old` and `df_new` (or regarding what we will focus on hereafter, `df_old_parts` and `df_new_parts`). However, this difference should be read as an improvement step, since the new dataset proved to have better regularity and properties with respect to the old one, for example granting gaussianity to some variables that didn't have it in the old dataset, not even after a box-cox transformation.

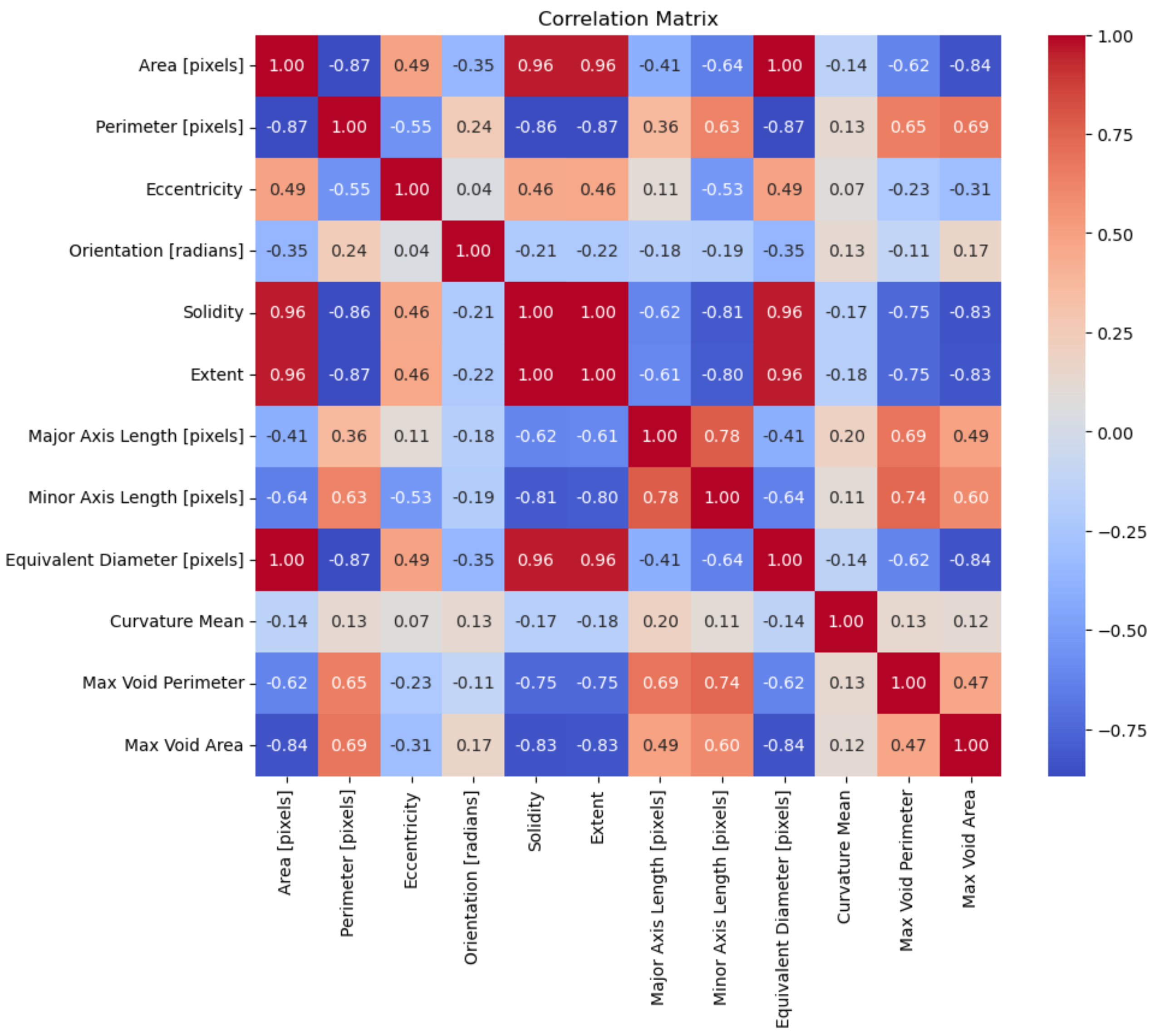
## Variables Analysis

In order to reduce the dimensionality and select only the necessary variables, we started from the ones respecting the assumptions of gaussianity (through a Shapiro-Wilk test) and excluding `Voids number` which being discrete couldn't be easily implemented in control charts.

As we will better describe in Section 1.3, we studied correlation both graphically, through a scatter plot, and quantitatively, through the correlation matrix (of which Figure 3 gives a visualization). From that we noticed how we could exclude some highly correlated variables and filter just a few "representatives" among them, in order to have a single variable carrying the information shared by multiple ones (e.g. `Area`, `Extent` and `Solidity` all relate to the amount of white pixels in the image). Indeed some variables conveyed the same information as others, a perfectly explainable result since the rotation and cropping had "standardized" the images, thus making some measurements redundant.

Another attempt at dimensionality reduction was brought through Principal Component Analysis. We applied PCA on the standardization of the dataset consisting of the previously selected uncorrelated columns, but the remaining dimensionality was still pretty high, and the loadings didn't allow a simple interpretation (and were not gaussian).

A PCA had also been made on the complete dataset, but even if it highly reduced the number of dimensions needed, the problems with normality of the selected PCs and interpretability made the attempt unsuccessful.



| | Area [pixels] | Perimeter [pixels] | Eccentricity | Orientation [radians] | Solidity | Extent | Major Axis Length [pixels] | Minor Axis Length [pixels] | Equivalent Diameter [pixels] | Curvature Mean | Max Void Perimeter | Max Void Area |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Area [pixels] | 1.00 | -0.87 | 0.49 | -0.35 | 0.96 | 0.96 | -0.41 | -0.64 | 1.00 | -0.14 | -0.62 | -0.84 |
| Perimeter [pixels] | -0.87 | 1.00 | -0.55 | 0.24 | -0.86 | -0.87 | 0.36 | 0.63 | -0.87 | 0.13 | 0.65 | 0.69 |
| Eccentricity | 0.49 | -0.55 | 1.00 | 0.04 | 0.46 | 0.46 | 0.11 | -0.53 | 0.49 | 0.07 | -0.23 | -0.31 |
| Orientation [radians] | -0.35 | 0.24 | 0.04 | 1.00 | -0.21 | -0.22 | -0.18 | -0.19 | -0.35 | 0.13 | -0.11 | 0.17 |
| Solidity | 0.96 | -0.86 | 0.46 | -0.21 | 1.00 | 1.00 | -0.62 | -0.81 | 0.96 | -0.17 | -0.75 | -0.83 |
| Extent | 0.96 | -0.87 | 0.46 | -0.22 | 1.00 | 1.00 | -0.61 | -0.80 | 0.96 | -0.18 | -0.75 | -0.83 |
| Major Axis Length [pixels] | -0.41 | 0.36 | 0.11 | -0.18 | -0.62 | -0.61 | 1.00 | 0.78 | -0.41 | 0.20 | 0.69 | 0.49 |
| Minor Axis Length [pixels] | -0.64 | 0.63 | -0.53 | -0.19 | -0.81 | -0.80 | 0.78 | 1.00 | -0.64 | 0.11 | 0.74 | 0.60 |
| Equivalent Diameter [pixels] | 1.00 | -0.87 | 0.49 | -0.35 | 0.96 | 0.96 | -0.41 | -0.64 | 1.00 | -0.14 | -0.62 | -0.84 |
| Curvature Mean | -0.14 | 0.13 | 0.07 | 0.13 | -0.17 | -0.18 | 0.20 | 0.11 | -0.14 | 1.00 | 0.13 | 0.12 |
| Max Void Perimeter | -0.62 | 0.65 | -0.23 | -0.11 | -0.75 | -0.75 | 0.69 | 0.74 | -0.62 | 0.13 | 1.00 | 0.47 |
| Max Void Area | -0.84 | 0.69 | -0.31 | 0.17 | -0.83 | -0.83 | 0.49 | 0.60 | -0.84 | 0.12 | 0.47 | 1.00 |

Figure 3: Correlation matrix of the variables of the dataset `df_new_parts`. The newly introduced variables `Curvature Mean`, `Max Void Perimeter` and `Max Void Area` will be defined in Section 1.3.

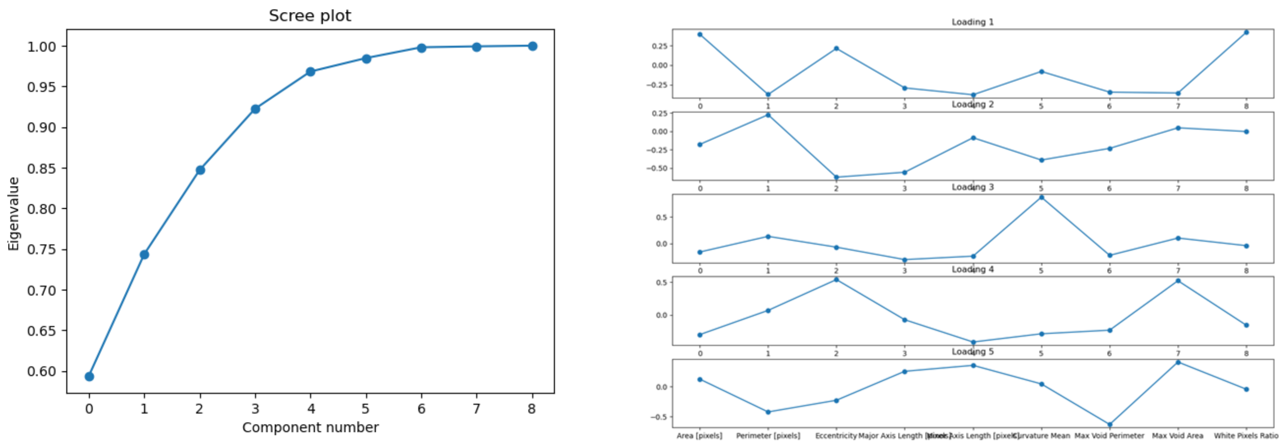


Figure 4: PCA results on

add caption on PCA img

## 1.3. Proposed metodology 10k chars

Before describing in details the steps we performed, we would like to highlight a first important modeling choice. The structure of the collected data led us thinking about two possible approaches:

1. using the original images, in which there were four objects in each tray;
2. using the objects-isolating images, obtained trough the segmentation code.

Having in total 40 objects, case 1 would have implied to use $m = 10$ samples with $n = 4$ as sample size; while in case 2 we would have set $m = 40$ and $n = 1$, i.e. considering individual observations.

Since the printer prints objects individually, we though the second approach to be more useful in a real-life scenario, since it would imply a more accurate monitoring on each object, rather than waiting for a set of them to be printed and then control that set.

This clarification will be important only in Idea 1 and 2 sections; while now we illustrate the work we performed on the dataset variables.

### Variable selection and definition

As mentioned before, towards the building of control charts we decided to consider only the necessary variables, looking for the ones respecting the assumptions of gaussianity (through a Shapiro-Wilk test) and randomness (through a Runs test), and that seemed worthy and meaningful to be included.

Therefore, our main idea was to select a minimal set of variables (i.e., avoiding redundancies), which could be useful to detect objects' defects, taking into account both mathematical assumptions and qualitative reasoning.

In this direction, we chose to remove the following variables:

- `Orientation [radians]`, since the new dataset cropping removed any difference in orientation between images.
- `Solidity` and `Extent`, as the information was already captured by `Area [pixels]` after the re-centering correction.
- `Eccentricity`, as we were not sure about how the computation behind it worked, nor if it could be useful.
- `Minor Axis Length [pixels]`, since we decided to keep `Major Axis Length [pixels]` instead, which together with the minor (trough a ratio) were highly correlated to `Eccentricity`; therefore including also the minor axis would have morally meant to include the eccentricity information, which instead we discarded.

So we were left having saved `Area [pixels]`, `Perimeter [pixels]` and `Major Axis Length [pixels]`.

To support our argument about `Eccentricity` ineffectiveness, we created an image with a visible deformation, but the obtained value seemed to be (wrongly) close to the correct ones. So in the end we did not valued this variable as interesting.

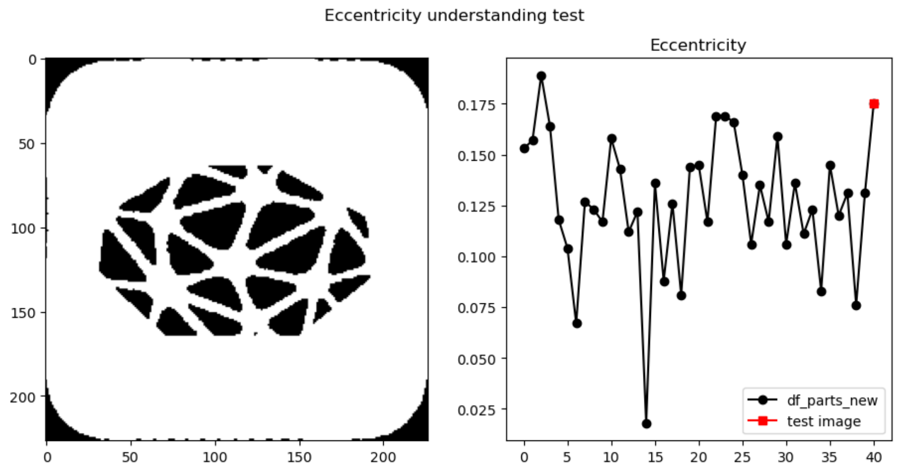


Figure 5: Plot of the test that we performed to understand the eccentricity variable.

Related to the `df_new_voids`, we decided not to keep any of the proposed variables because it made sense for us to include a variable if and only if it had a normal distribution in all images. However, we did not find any variable that satisfied this constraint. Moreover, we couldn't think the variables in that dataset as i.i.d. since the voids were different in terms of location and dimension, and problematic to group and compare. So we did not use this dataset as it was, but processed it to generate two new variables illustrated below.

After experimenting a bit trying to extract some more information from the images, we generated some new variables, to improve the descriptive capabilities of our model, and here we report the ones that made it to the control charts:

1. `Curvature mean`. From our own segmentation code, we took the section about the main contour extraction and from there we computed the curvature for each set of three consecutive points. This new variable stores the the mean of all those values.
2. `Max void area`. The voids of the objects were not i.i.d., due to different sizes and positions, so we could not use them altogether. Instead, we decided to consider, for each object, only the void with the biggest area (and not the one of smallest area because it wouldn't have been much informative as the segmentation of smaller voids was heavily affected by pixels noise).
3. `Max void perimeter`. With a reasoning similar to the previous variable, we decided to include the information about the perimeter of the biggest void inside each object. In fact, the coefficient of correlation with `Max void area` was just 0.47, therefore we thought this variable as also worthy to be included.

Regarding voids but being related to each object individually, the last two variables were computed from `df_new_voids` but stored as new variable in `df_new_parts`, to have a single dataset to work on. Another new variable not included but still worth mentioning is `White pixels ratio`, created as an attempt to obtain a more robust equivalent of `Area`, being the new one a percentage of white pixels rather than a simple count.

The following table summarizes how these new variables, together with those already present, could detect defected pieces:

| Variable | Kind of defect it could help to detect |
|---|---|
| Area | too many holes; broken pieces |
| Perimeter | wrong dimensions; presence of bulges |
| Major Axis Length [pixels] | flattening deformations |
| Curvature mean | cracks along perimeter |
| Max Void Area | too big void in the inside; wrongly shaped voids |
| Max Void Perimeter | voids collapsed together; weirdly shaped voids |

Table 4: Interpretation about which kind of defects these selected variables could help to detect, through their control charts.

| Variable | Shapiro Test pvalues | Dasaset |
|---|---|---|
| Area [pixels] | 0.7310 | `df_new_parts` |
| Perimeter [pixels] | 0.8690 | `df_new_parts` |
| Major Axis Length [pixels] | 0.0967 | `df_new_parts` |
| Curvature Mean | 0.5501 | `df_new_parts` |
| Max Void Area | 0.5666 | `df_new_parts` |
| Max Void Perimeter | 0.9064 | `df_new_parts` |

Table 5: Pvalues of the Shapiro test to check normality on the variables we selected for the control charts.

However, we had some problems regarding the independence assumption, and we proposed two approaches to deal with them, based on the different weight and role given to time.

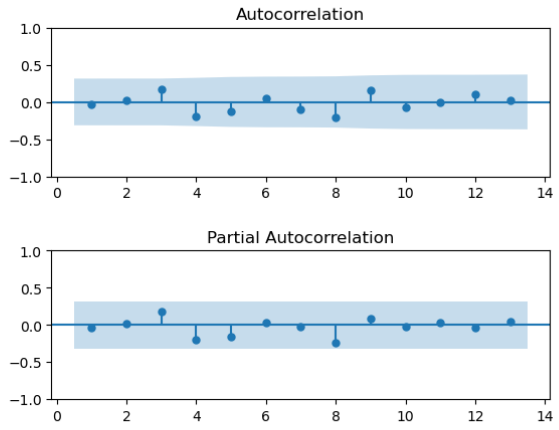
## Idea 1: FVC and SCC

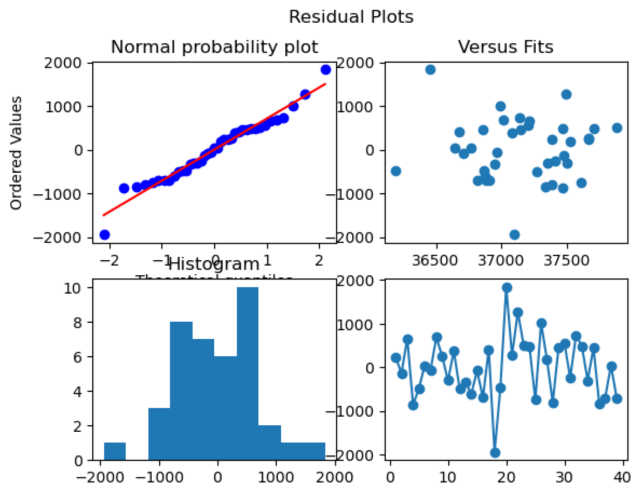
The first approach we tried out consisted in treating the data as time series, assuming that issues with independence were caused by underlying models and patterns.
We started with I-MR-R charts, noticing an improvement already, but dealing whit the independence problems and building Fitted-Values chart (FVC) and Special-Cause chart (SCC), where needed, was a better idea.
We present the results a variable at a time as ordered in Table 5.

### Area

Looking at the pvalue of the runs test (0.025) and at the auto-correlation plots, we noticed that the independence assumption was not reached.

We decided to build an autoregressive model of order one (AR(1)) to model the data and obtain SCC and FVC control charts. The residuals were indipendent, with a pvalue for the runs test of 0.254, gaussian with pvalue of shapiro-wilk of 0.496, and homoschedastic as shown in the plots below.





The obtained control charts are:

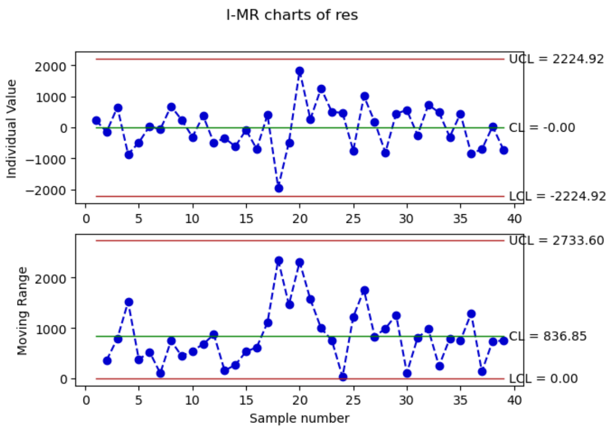


Figure 6: SCC for Area

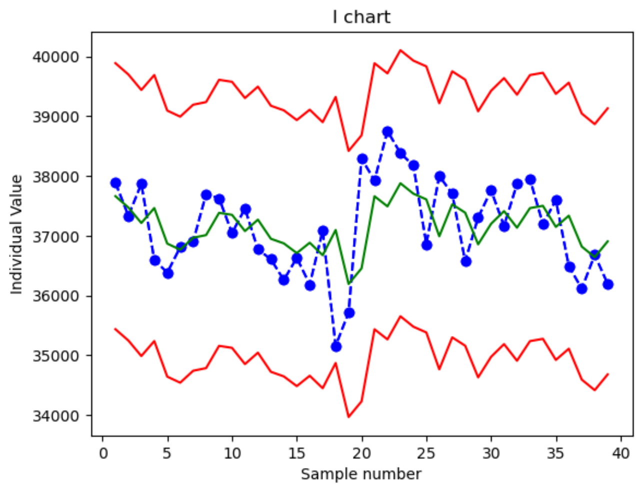


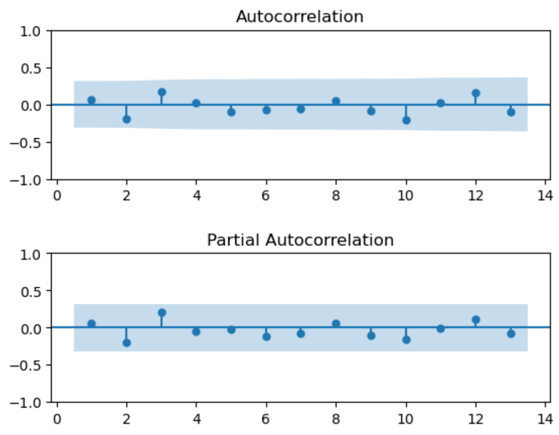
Figure 7: FVC for Area

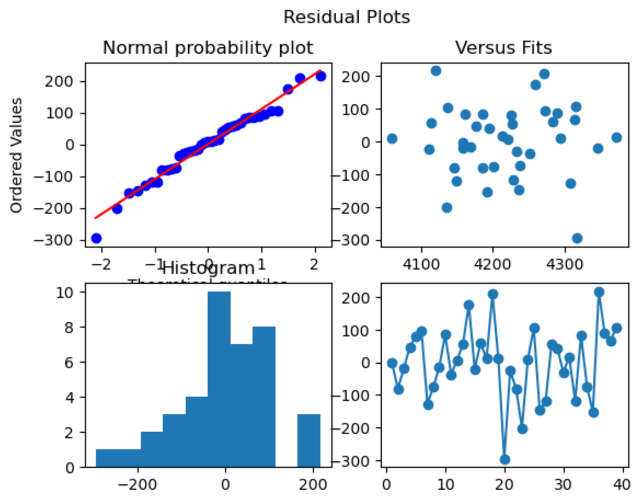
No strange pattern, outliers or out of control observations appear.

## Perimeter

In the following variable we observed the same situation we experienced with the `Area` variable: we had Gaussian distribution but we missed independence (the pvalue of the run test was equal to 0.026).

As done before, we modeled the data with an AR(1) model obtaining independent residuals (the pvalue of the run test was equal to 0.436) with Gaussian distribution (Shapiro-Wilk test p-value equal to 0.685) and same variance.





The related control charts are:

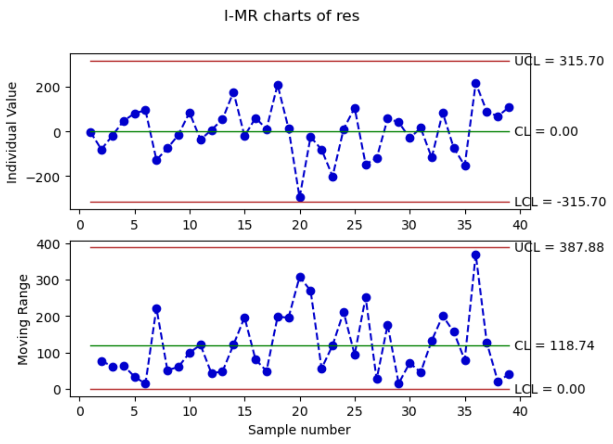


Figure 8: SCC for Area

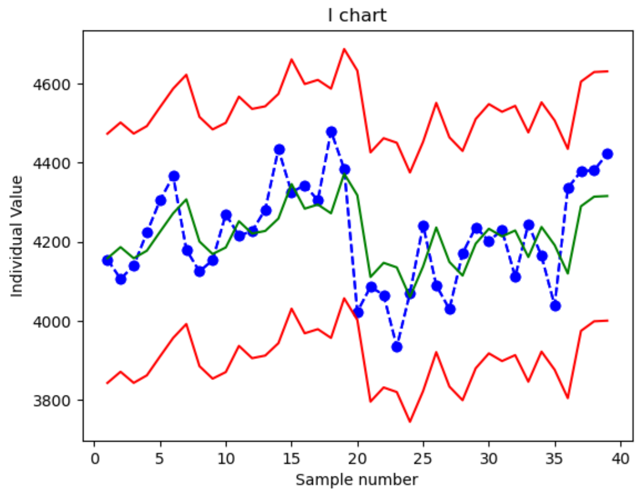


Figure 9: FVC for Area

As before, no strange pattern, outliers or out of control observations appear.

## Major Axis Length, Curvature Mean, Max Void Area and Max Void Perimeter

For all the remaining variables considered, both the assumptions, gaussianity and independence were satisfied. We measured, in fact, the following pvalues of the runs test:

| Variable | Runs Test pvalues |
|---|---|
| Major Axis Length | 0.749 |
| Curvature Mean | 0.522 |
| Max Void Area | 0.749 |
| Max Void Perimeter | 0.161 |

Table 6: Pvalues of the runs test to check independence.

So, we were able to build the "classical" SPC control charts on the original variables and not on the residuals.

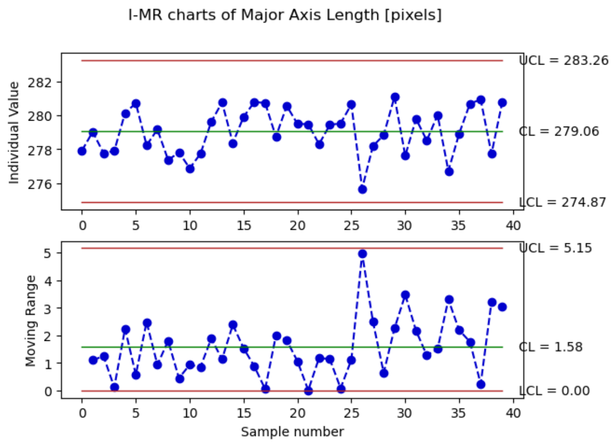


Figure 10: SPC for Major Axis Length

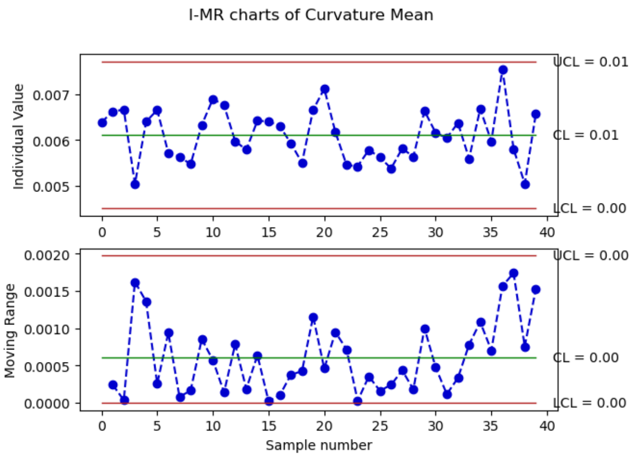


Figure 11: SPC for Curvature Mean

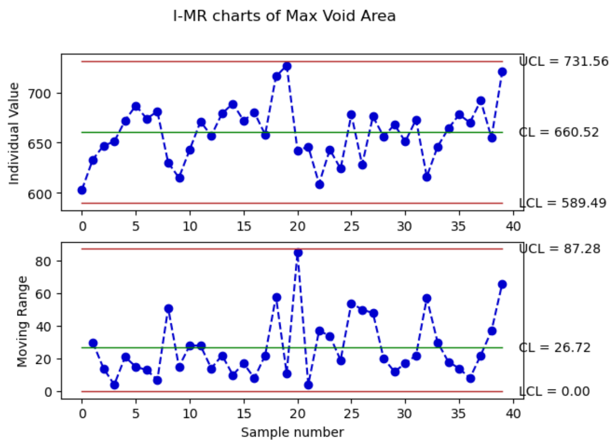


Figure 12: SPC for Max Void Area

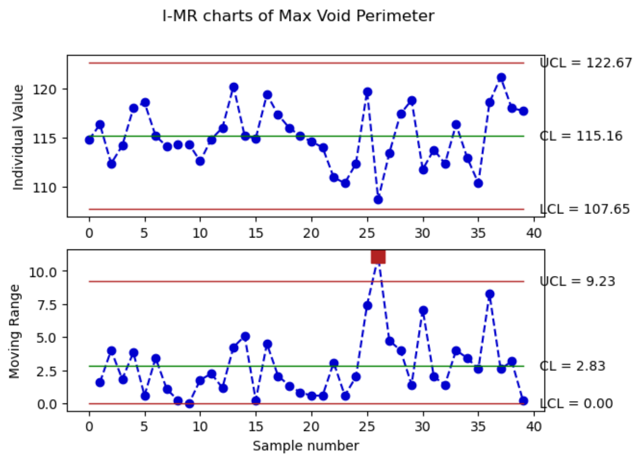


Figure 13: SPC for Max Void Perimeter

As we can see, only an out of control observation seems to be present in our control charts but no assignable cause were found. In addition to this, it is related to the Moving Range control chart that, as explained below, it will not useful for our scope. So, all the previous control charts are acceptable.

## Idea 2: Charts on shuffled data

Due to the mechanism of the image acquisition, we could assume that the obtained data should not be considered as a time series. In fact, the objects to be placed in the trays were selected in a random order; so even if originally existed a particular time ordering the acquisition procedure made it to be lost. Consistently with this we could also assume that there should be no problems with independence in the data. Indeed, all the objects were in control, by design of the project, and they were disposed in the trays according not to a particular order but randomly. So any independence problem would be caused by chance.

For this reason, since the order in which objects were originally stored in the dataset caused problems with independence through the Runs test, as mentioned before, we decided to shuffle the data. This suited the new context since it had no problems with respect to time issues, as being the objects randomly placed at the beginning meant that we could also re-order them in any way without altering the design and construction of the experiment. This shuffling made us

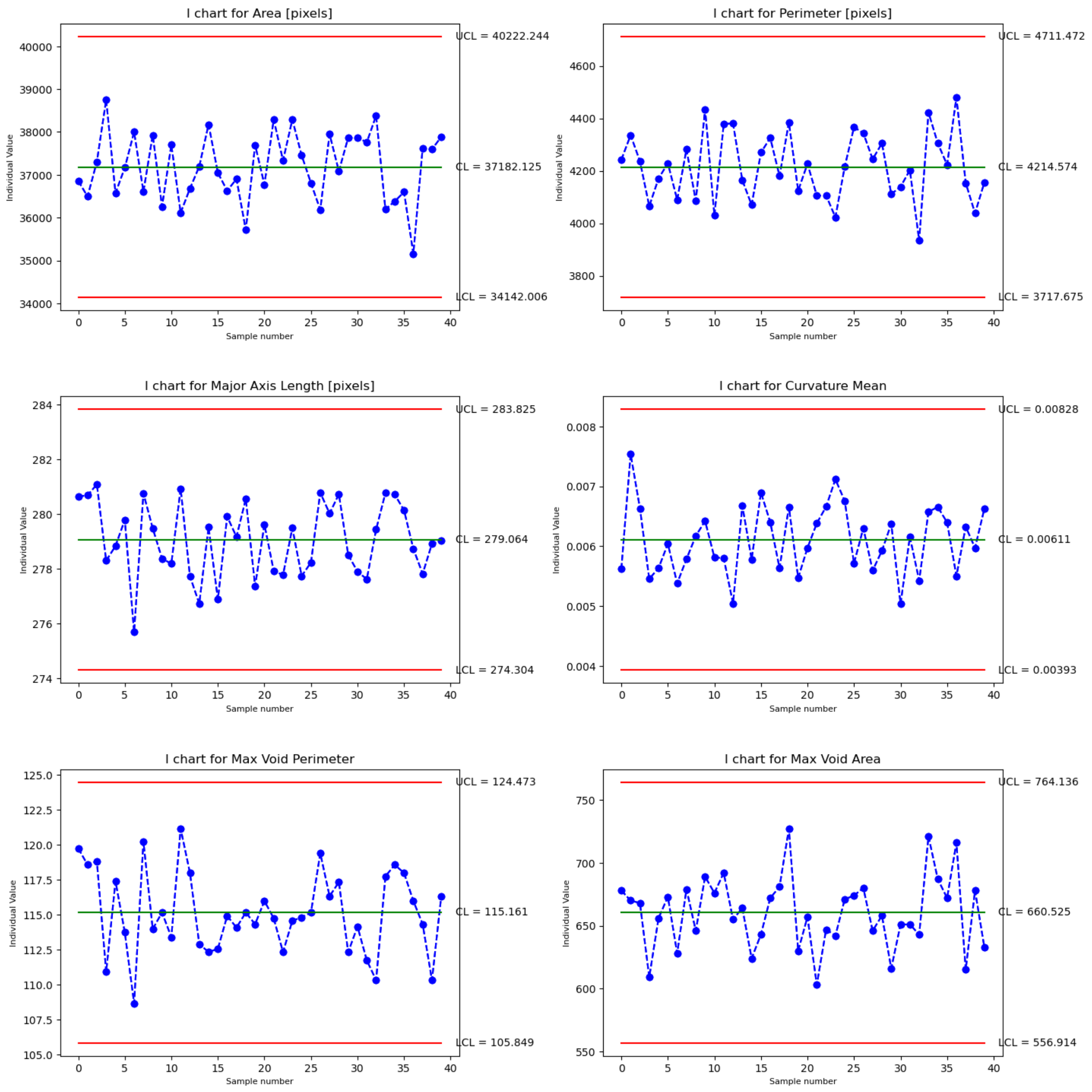


Figure 14: I charts, for the six variables we selected, generated according to the Idea 2.

obtain independence (as the Runs tests showed in Table 7) which according to the experiment design should have been there "by default".

| Variable | Runs Test pvalues |
|---|---|
| Area [pixels] | 0.521 |
| Perimeter [pixels] | 0.193 |
| Major Axis Length [pixels] | 0.200 |
| Curvature Mean | 0.749 |
| Max Void Perimeter | 0.789 |
| Max Void Area | 0.109 |

Table 7: Pvalues of the Runs test to check randomness assumption in the shuffled dataset.

As we saw in the previous sections we had no problems with the gaussianity assumption, and

say that we decided to consider only the control chart for the mean and therefore we built the control chart using this shuffled `df_new_parts` dataset.

The random permutation introduced slightly changes the MR computation, as the random re-ordering made that the moving range values changed from shuffle to shuffle. But on average they were very close, so this was not considered a problem. Due to this slight variability, however, we just focused on the I charts.

We also applied a Bonferroni correction in the end, since we made multiple (six) univariate charts on the same data, so that each chart was individually built with a $\alpha = \frac{\alpha_{\text{tot}}}{6} = \frac{0.0027}{6}$, which gave a $K = 3.5$. The correction leads to larger value of K, so a more precise choice could be to try a single multivariate chart, but allows better interpretability in identifying the variable causing the defect.

### 1.4. Results 10k chars

#### Individual Control Charts

Since we couldn't know whether the Phase 2 dataset would be given to us in the form of trays again or as individual objects, we chose to build both I-MR and Xbar-R control charts to be able to handle both cases. In this second case, we had to resize the dataset to consider each tray (the subsample) as a row, while the choice of R instead of S was to grant robustness, as trays were expected to contain a constant of 4 objects, as a different number would probably mean a defect itself.

In the end, however, after all the attempts and charts, among the multiple charts we selected the individual ones obtained with the shuffling procedure and bonferroni correction as our final result. They are, in fact, the ones with the most plausible and consistent hypothesis, as well as lacking issues of hugging or false alarms.

#### Multivariate Control Charts

As for what concerns future development, we could try with more powerful multivariate models or different kinds of less conservative corrections. A last idea worth exploring would be trying to to consider the variable `Num_voids` which we had excluded for being discrete, but could now tackle using "by attribute" control charts.

## Phase 2

### 2.1. Preliminary data analysis

### 2.2. Test of your proposed approach on new data

### 2.3. Discussion

## References

[1] J. Hao. Segmentation implementation: Given an image and a 4 points in the image (no 3 points are co-linear). find the rotated rectangle enclosing the polygon formed by the 4 points and crop the rotated rectangle from the image. all done by opencv. `https://gist.`

github.com/jdhao/1cb4c8f6561fbdb87859ac28a84b0201, 2018.

[2] D. C. Montgomery. *Introduction to Statistical Quality Control*. Wiley, 8th edition, August 2019.

[3] S. User. Segmentation implementation: How to straighten a rotated rectangle area of an image using opencv in python? https://lc.cx/v8Pl4c, 2012. The reported link has been reduced.

[4] Wikipedia contributors. Curvature — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Curvature, 2024.