Capability Indices. Practical part

Anastasiia Apatenko

Class: Statistical Quality Control

Matriculation Number: 143732

June 23, 2025

Table of contents

Introduction

In this part I will present the practical implementation of my paper "Capability Indices".

To illustrate the theory, discussed in the theoretical part, I decided to use the data base `ss.data.bolts` from the library `SixSigma`. It is a data frame with 50 observations of the diameter of the bolts manufactured in a production line.(Cano et al. 2011)

Variables:

- `diameter` a numeric vector with the diameter of the bolts

I deliberately chose the small data base with just one variable. This makes the illustration of the theory easy to reproduce.

`bolts` dataset

Loading the data

```
library(SixSigma)
data(ss.data.bolts)
data <- ss.data.bolts
sum <- summary(data)
sum
```

```
    diameter
 Min.   : 9.671
 1st Qu.: 9.899
 Median :10.041
 Mean   :10.031
 3rd Qu.:10.146
 Max.   :10.404
```

Assumption check

Before calculation of the indices, I run the check of all four basic assumptions, underlying the process capability indices.

Normal distribution of process data

For normality test I chose the Shapiro–Wilk test from the basic R library.(Liang, Tang, and Chan 2009)

```
shapiro.test(data$diameter)
```

```

    Shapiro-Wilk normality test
```

```
data:  data$diameter
W = 0.98158, p-value = 0.6202
```

Interpretation of the results:

$$H_0 : \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \text{ is a sample from normal } \mathcal{N}_p(\mu, \Sigma)$$
$$H_1 : \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \text{ is not a sample from normal } \mathcal{N}_p(\mu, \Sigma)$$

W(Test statistic) $= 0.98158 \rightarrow$close to 1 $\rightarrow$suggests the normality.

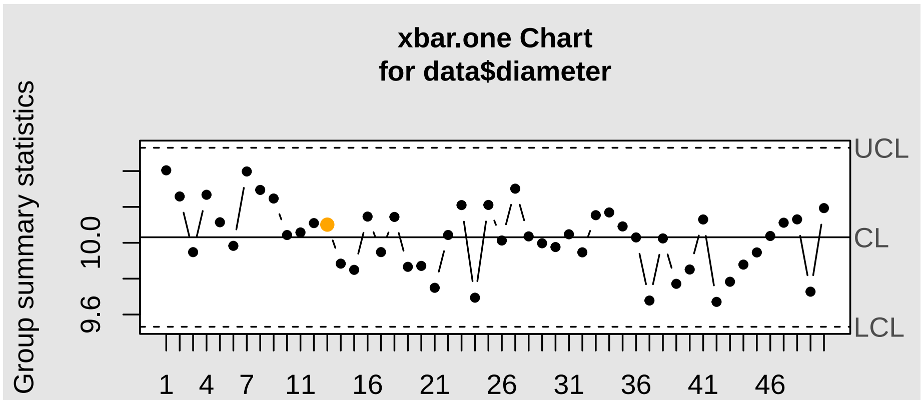
p-value $= 0.6202 > 0.05 \rightarrow$failed to reject H_0

Result: normality proofed

Process is in statistical control

To prove that the process is in statistical control, I will use the `qcc` library. I used the `qcc` function to build the simple $\bar{X}$ (`xbar.one`) chart since I deal with one–at–time data.(Scrucca, n.d.)

```r
library(qcc)
qcc <- qcc(data$diameter, type = "xbar.one")
```



```
Number of groups = 50
Center = 10.0308        LCL = 9.531753     Number beyond limits = 0
StdDev = 0.1663482      UCL = 10.52984     Number violating runs = 1
```

After receiving the results, the following conclutions can be made: - all points fall within the control limits (UCL and LCL);

- there is no clear trend upward or downward;
- there are no patterns and no unusually high or low values;

- only 1 value violates the run rule (7 or more points in a raw on one side of the center line). But it is just 1 point out of 50.

Result: the process is in statistical control

Process mean is centered between specification limits

```
mu <- mean(data$diameter)
m <- qcc$center
mu == m
```

```
[1] TRUE
```

Result: the process is centered

Target value lies midway between USL and LSL ($T = m$)

Since the target value is not provided, I can choose it myself. The best way to do this is to set the T to midway. This gives the 100% chance, that the assumption will be met. We already have $\mu = m$. Therefor by setting the $T = m$ the assumption will be met

```
USL <- qcc$limits[2]
LSL <- qcc$limits[1]
T <- (USL + LSL) / 2
mu == T
```

```
[1] TRUE
```

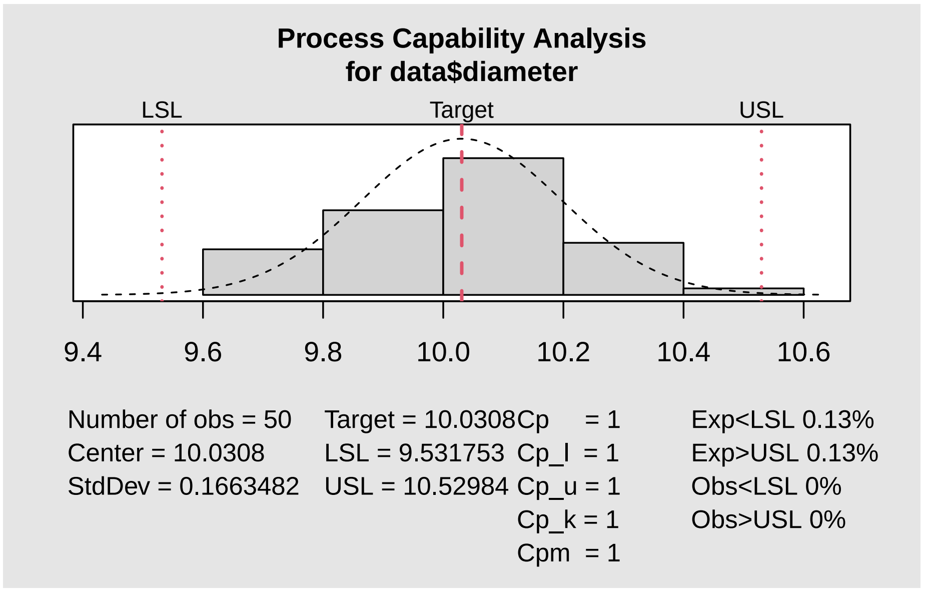
Result: the target value is at the mid-point of the control limits

Calculation of the capability indices

`qcc` library for C_p, C_{pl}, C_{pu}, C_{pk}, C_{pm}

I used the `process.capability` function to calculate the indices: C_p, C_{pl}, C_{pu}, C_{pk}, C_{pm}.

```
cap <- process.capability(qcc, spec.limits = c(LSL, USL))
```



```
Process Capability Analysis

Call:
process.capability(object = qcc, spec.limits = c(LSL, USL))

Number of obs = 50          Target = 10.03
       Center = 10.03          LSL = 9.532
       StdDev = 0.1663         USL = 10.53

Capability indices:

      Value    2.5%  97.5%
Cp        1  0.8025  1.197
Cp_l      1  0.8166  1.183
Cp_u      1  0.8166  1.183
Cp_k      1  0.7815  1.218
Cpm       1  0.8045  1.195

Exp<LSL 0.13%    Obs<LSL 0%
Exp>USL 0.13%    Obs>USL 0%
```

From the output several conclusions can be done:

- $C_p = 1$: the Six Sigma spread of the process fits within the specification width;
- $C_{pk} = 1$: the process mean is exactly centered $\rightarrow C_{pu} = C_{pl} = 1$;

- $C_{pm} = 1$: no penalty is applied, since $T = m$;
- defect rate is 0 $\rightarrow$no data points fall out of the specification limits.

These results show the process stability. Although capability index 1 is just a minimum level of acceptance. $C_p = 1$ gives us the possible mistake value of 2700ppm. In our dataset we did not see any out of limits value, only because the dataset itself is small.

Let's try to fix this. What will definitly not work:

- shift of the target value - only changes the C_{pm}
- shift of the μ - only changes the C_{pk}.

There are several ways to make the indices´ value greater:

- reduce the process variation

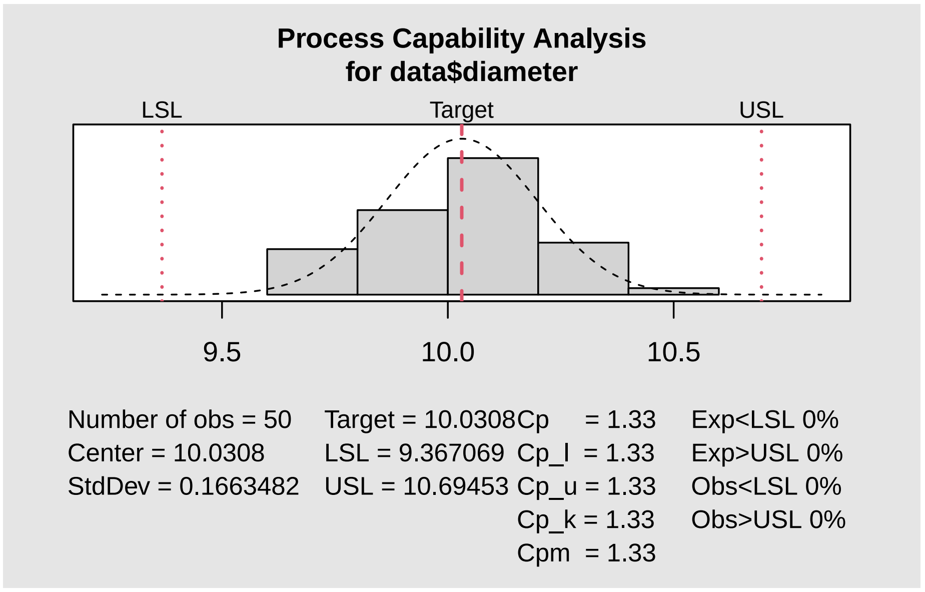
This is a good option, but not in our case. To do so too many changes has to be implemented. Although in the real world this would be the best long-term option.

- widen the spread

This is already much easier and more realistic. This option is the best, since we do not work with the real data. We only testing the possible capability analysis process.

On the first try we had the basic spred at 6σ. Instead of just setting the width to the fixed value, it can be calculated, based on the desired value of the indices. In our case we aim to eigher 1.33 or 2:

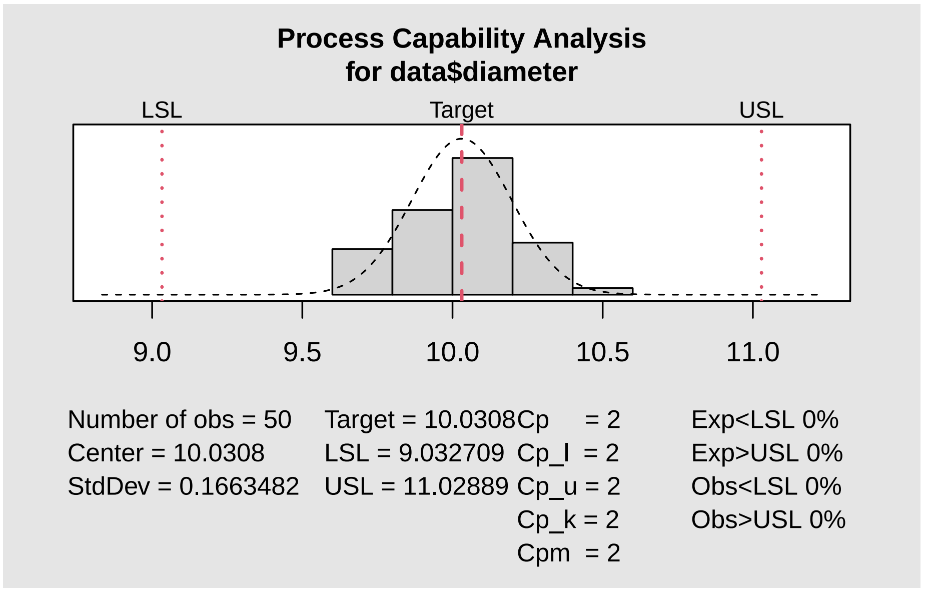
```r
mu <- mean(data$diameter)
sd <- qcc$std.dev

simulate_capability <- function(qcc, sd, mu, cp_target) {
  width <- 6 * sd * cp_target
  LSL <- mu - width / 2
  USL <- mu + width / 2
  cap <- process.capability(qcc, spec.limits = c(LSL, USL))
  return(cap)
}
cap_133 <- simulate_capability(qcc, sd, mu, 1.33)
```



```
Number of obs = 50        Target = 10.0308  Cp   = 1.33    Exp<LSL 0%
Center = 10.0308          LSL = 9.367069    Cp_l = 1.33    Exp>USL 0%
StdDev = 0.1663482        USL = 10.69453    Cp_u = 1.33    Obs<LSL 0%
                                            Cp_k = 1.33    Obs>USL 0%
                                            Cpm  = 1.33
```

```
Process Capability Analysis

Call:
process.capability(object = qcc, spec.limits = c(LSL, USL))

Number of obs = 50           Target = 10.03
       Center = 10.03           LSL = 9.367
       StdDev = 0.1663          USL = 10.69

Capability indices:

      Value   2.5%  97.5%
Cp     1.33  1.067  1.592
Cp_l   1.33  1.096  1.564
Cp_u   1.33  1.096  1.564
Cp_k   1.33  1.051  1.609
Cpm    1.33  1.070  1.590

Exp<LSL 0%   Obs<LSL 0%
Exp>USL 0%   Obs>USL 0%
```

```r
cap_200 <- simulate_capability(qcc, sd, mu, 2)
```



```
Process Capability Analysis

Call:
process.capability(object = qcc, spec.limits = c(LSL, USL))

Number of obs = 50          Target = 10.03
       Center = 10.03          LSL = 9.033
       StdDev = 0.1663         USL = 11.03

Capability indices:

      Value   2.5%  97.5%
Cp        2  1.605  2.394
Cp_l      2  1.659  2.341
Cp_u      2  1.659  2.341
Cp_k      2  1.593  2.407
Cpm       2  1.609  2.390

Exp<LSL 0%   Obs<LSL 0%
Exp>USL 0%   Obs>USL 0%
```

What can be observed from the output:

The widening of the process spread made the capability´s values higher with assumptions still staying satisfied. However this is not the best approach:

- in reality it is way harder to apply;

- in real manufacturing process can lead to the lower quality of the product;
- it is not reflecting the real capability.

Metric	**Cp = 1.0**	**Cp = 1.33**	**Cp = 2.0**
Center (μ)	10.03	10.03	10.03
Target (T)	10.03	10.03	10.03
Std. Dev. (σ)	0.1663	0.1663	0.1663
LSL	9.532	9.367	9.033
USL	10.53	10.69	11.03
C_p	1.00	1.33	2.00
C_{pk}	1.00	1.33	2.00
C_{pm}	1.00	1.33	2.00
Exp $<$ LSL	0.13%	0%	0%
Exp $>$ USL	0.13%	0%	0%

k index

In this part we will look at the k index and see how it works. This index only works for the centeredness of the process.

```r
calc_k <- function(mu, T, USL, LSL) {
  k <- abs(mu - T) / ((USL - LSL) / 2)
  return(k)
}

USL <- qcc$limits[2]
LSL <- qcc$limits[1]

k_value <- calc_k(mu, T, USL, LSL)
k_value
```

```
[1] 0
```

The calculation of it under assumption that the $\mu = m = T$ results in the $k = 0$. Which means that the process is perfectly centered.

Let's shift the T from 10.03 to 9.5

```r
T = 9.5
k_value <- calc_k(mu, T, USL, LSL)
k_value
```

```
[1] 1.063628
```

The k value immediately increased. This illustrates its sensitivity to any shift of the target value.

Manual calculation of the C_{pmk}

Now, when we calculated all first and second-generation indices, we can look at the C_{pmk} index. For this index only the manual calculation is possible.

```r
calc_cpmk <- function(x, USL, LSL, T) {
  mu <- 10.030798
  sd <- 0.1663482
  denom <- 3 * sqrt(sd^2 + (mu - T)^2)
  lower <- (mu - LSL) / denom
  upper <- (USL - mu) / denom
  cpmk <- min(lower, upper)
  return(cpmk)
}
Cpmk_result <- calc_cpmk(data$diameter, USL, LSL, T = mu)
Cpmk_result
```

```
[1] 1
```

Result: C_{pmk} index also equals to 1. No pemalty applied since $\mu = m = T$. This example showed the capability of the process (eventhough not the best one) and the real use of the capability improvements.

Shift of the target value (T) for C_{pm} and C_{pmk}

Let's also prove that the shift in the target value forces the indeces C_{pm} and C_{pmk} to shrink. For this the custom function will be created. There are two reasons for this. Firstly, there are no library in R to show this explisitly on just one function. Seconly, the manual function gives the chabge to go step by step and see the process of index calculation. I made sure to leave all the variable the same, but only shifted th etarget value from midpoint($T = 10.03$) closer to the LSL ($T = 9.5$).

```r
calc_indices_shifted_T <- function(x, USL, LSL, T, mu, sd) {

  Cp <- (USL - LSL) / (6 * sd)
  Cpu <- (USL - mu) / (3 * sd)
  Cpl <- (mu - LSL) / (3 * sd)
  Cpk <- min(Cpu, Cpl)
  Cpm <- (USL - LSL) / (6 * sqrt(sd^2 + (mu - T)^2))
  Cpmk <- min((USL - mu), (mu - LSL)) / (3 * sqrt(sd^2 +
(mu - T)^2))
  return(list(Cp = Cp, Cpk = Cpk, Cpm = Cpm, Cpmk = Cpmk))
}
sd <- qcc$std.dev
calc_indices_shifted_T(data$diameter, USL, LSL, 9.5, mu, sd)
```

```
$Cp
```

```
[1] 1

$Cpk
[1] 1

$Cpm
[1] 0.299051

$Cpmk
[1] 0.299051
```

As it was expected, C_p and C_{pk} did not change at all, since target value is not present in their mathematical formulas. On the other hand C_{pm} and C_{pmk} dropped dramatically, due to the penalization in the denominator.

This custom function showed the role of the target value in the second and third-generation indices.

Conclusion

The practical analysis confirmed the theoretical behavior of capability indices. The process met all assumptions. The initial capability indices equaled to 1, indicating that the process meets minimum capability standards but does not exceed it. By creating the wider tolerance intervals, it was demonstrated how capability indices increase (achieve values of 1.33 and 2). But such an approach may be not applied in the real world. Manually calculated C_{pmk} index proved its consistency under the assumption of centered process. The manually created function showed the importance of the target value in the C_{pm} and C_{pmk} indices.

Overall, this analysis highlighted the usefulness of capability indices in optimizing manufacturing process.

References

Cano, Emilio L., Javier M. Moguerza, Mariano Prieto, and Andrés Redchuk. 2011. "SixSigma: Six Sigma Tools for Quality Control and Improvement." Comprehensive R Archive Network. https://doi.org/10.32614/CRAN.package.SixSigma.

Liang, Jiajuan, Man-Lai Tang, and Ping Shing Chan. 2009. "A Generalized Shapiro–Wilk W<math><mi Is="true">W</Mi></Math> Statistic for Testing High-Dimensional Normality." *Computational Statistics & Data Analysis* 53 (11): 3883–91. https://doi.org/10.1016/j.csda.2009.04.016.

Scrucca, Luca. n.d. "Qcc: An R Package for Quality Control Charting and Statistical Process Control."