# Process Capability - bins' colour

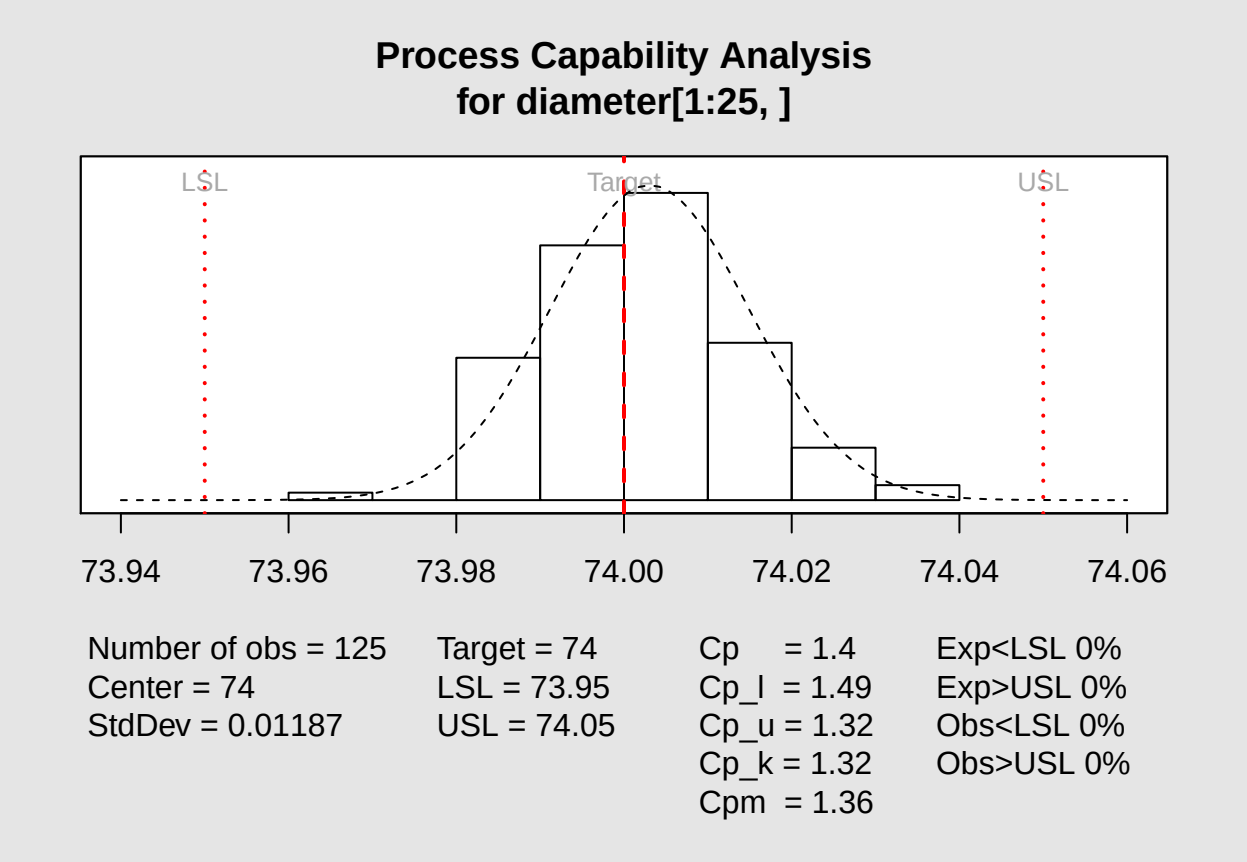
*Emilio López Cano*

*25/09/2014*

Workaround to change colors in a process capability chart, `qcc` package.

Taking the example in ?process.capability

```
library(qcc, quietly = TRUE)
data(pistonrings)
attach(pistonrings)
diameter <- qcc.groups(diameter, sample)
q <- qcc(diameter[1:25,], type="xbar", nsigmas=3, plot=FALSE)
process.capability(q, spec.limits=c(73.95,74.05))
```




```
##
## Process Capability Analysis
##
## Call:
## process.capability(object = q, spec.limits = c(73.95, 74.05))
##
## Number of obs = 125          Target = 74
##        Center = 74              LSL = 73.95
```

```
##        StdDev = 0.01187         USL = 74.05
##
## Capability indices:
##
##       Value   2.5%  97.5%
## Cp    1.405  1.230  1.579
## Cp_l  1.490  1.327  1.653
## Cp_u  1.319  1.173  1.465
## Cp_k  1.319  1.145  1.493
## Cpm   1.360  1.187  1.534
##
## Exp<LSL 0%   Obs<LSL 0%
## Exp>USL 0%   Obs>USL 0%
```

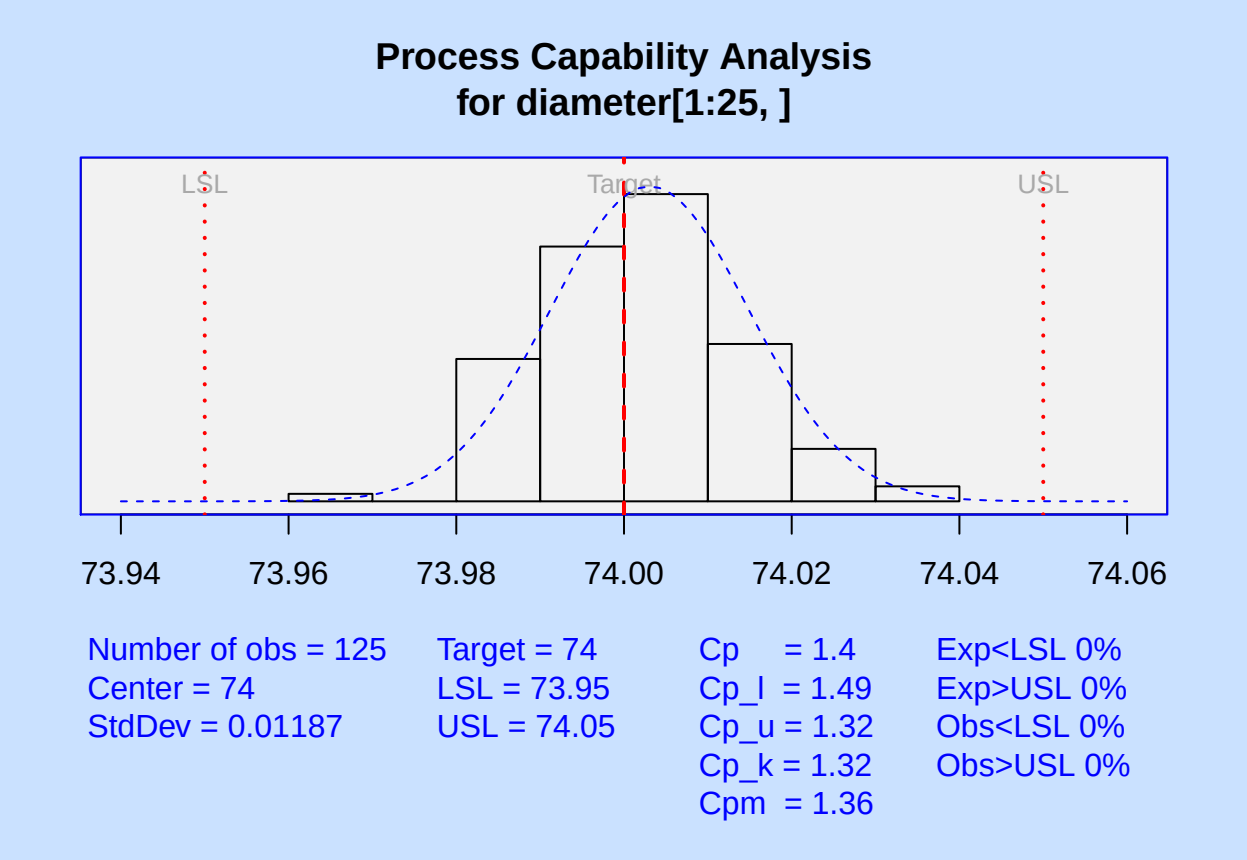
```
detach(pistonrings)
```

Some colors can be changed via `qcc.options()`

```
qcc.options(bg.margin = "lightsteelblue1", bg.figure = gray(.95))
```

To change globally the `col` graphical parameter, but it affects to some text and some lines:

```
oldpar <- par()
par(col = "blue")
process.capability(q, spec.limits=c(73.95,74.05))
```



```
##
## Process Capability Analysis
##
## Call:
## process.capability(object = q, spec.limits = c(73.95, 74.05))
##
## Number of obs = 125          Target = 74
##        Center = 74              LSL = 73.95
##        StdDev = 0.01187         USL = 74.05
##
## Capability indices:
##
##       Value   2.5%  97.5%
## Cp    1.405  1.230  1.579
## Cp_l  1.490  1.327  1.653
## Cp_u  1.319  1.173  1.465
## Cp_k  1.319  1.145  1.493
## Cpm   1.360  1.187  1.534
##
## Exp<LSL 0%   Obs<LSL 0%
## Exp>USL 0%   Obs>USL 0%
```

```
par <- oldpar
```

So, if you need to customize the graphical output, make a copy of the function:

```
process.capability
```

```
## function (object, spec.limits, target, std.dev, nsigmas, confidence.level = 0.95,
##     breaks = "scott", add.stats = TRUE, print = TRUE, restore.par = TRUE)
## {
##     if ((missing(object)) | (!inherits(object, "qcc")))
##         stop("an object of class 'qcc' is required")
##     if (!(object$type == "xbar" | object$type == "xbar.one"))
##         stop("Process Capability Analysis only available for charts type \"xbar\" and \"xbar.one\" cl
##     x <- as.vector(object$data)
##     x <- x[!is.na(x)]
##     sizes <- object$sizes
##     center <- object$center
##     if (missing(std.dev))
##         std.dev <- object$std.dev
##     n <- length(x)
##     title <- paste("Process Capability Analysis\nfor", object$data.name)
##     if (missing(spec.limits))
##         stop("specification limits must be provided")
##     if (!length(spec.limits) == 2)
##         stop("wrong specification limits format")
##     LSL <- min(spec.limits, na.rm = TRUE)
##     USL <- max(spec.limits, na.rm = TRUE)
##     if (missing(target)) {
##         target <- mean(spec.limits, na.rm = TRUE)
##     }
##     if (target < LSL | target > USL)
```

```
##         warning("target value is not within specification limits...")
##     if (missing(nsigmas))
##         if (is.null(object$nsigmas))
##             stop("nsigmas not available in the 'qcc' object. Please provide nsigmas.")
##         else nsigmas <- object$nsigmas
##     if (confidence.level < 0 | confidence.level > 1)
##         stop("the argument confidence.level must be a value between 0 and 1")
##     Cp <- (USL - LSL)/(2 * nsigmas * std.dev)
##     Cp.u <- (USL - center)/(nsigmas * std.dev)
##     Cp.l <- (center - LSL)/(nsigmas * std.dev)
##     Cp.k <- min(Cp.u, Cp.l)
##     Cpm <- Cp/sqrt(1 + ((center - target)/std.dev)^2)
##     alpha <- 1 - confidence.level
##     Cp.limits <- Cp * sqrt(qchisq(c(alpha/2, 1 - alpha/2), n -
##         1)/(n - 1))
##     Cp.u.limits <- Cp.u * (1 + c(-1, 1) * qnorm(confidence.level) *
##         sqrt(1/(9 * n * Cp.u^2) + 1/(2 * (n - 1))))
##     Cp.l.limits <- Cp.l * (1 + c(-1, 1) * qnorm(confidence.level) *
##         sqrt(1/(9 * n * Cp.l^2) + 1/(2 * (n - 1))))
##     Cp.k.limits <- Cp.k * (1 + c(-1, 1) * qnorm(1 - alpha/2) *
##         sqrt(1/(9 * n * Cp.k^2) + 1/(2 * (n - 1))))
##     df <- n * (1 + ((center - target)/std.dev)^2)/(1 + 2 * ((center -
##         target)/std.dev)^2)
##     Cpm.limits <- Cpm * sqrt(qchisq(c(alpha/2, 1 - alpha/2),
##         df)/df)
##     names(Cp.limits) <- names(Cp.k.limits) <- names(Cpm.limits) <- c(paste(round(100 *
##         alpha/2, 1), "%", sep = ""), paste(round(100 * (1 - alpha/2),
##         1), "%", sep = ""))
##     exp.LSL <- pnorm((LSL - center)/std.dev) * 100
##     if (exp.LSL < 0.01)
##         exp.LSL <- 0
##     exp.USL <- (1 - pnorm((USL - center)/std.dev)) * 100
##     if (exp.USL < 0.01)
##         exp.USL <- 0
##     obs.LSL <- sum(x < LSL)/n * 100
##     obs.USL <- sum(x > USL)/n * 100
##     xlim <- range(x, USL, LSL, target)
##     xlim <- xlim + diff(xlim) * c(-0.1, 0.1)
##     xx <- seq(min(xlim), max(xlim), length = 100)
##     dx <- dnorm(xx, center, std.dev)
##     h <- hist(x, breaks = breaks, plot = FALSE)
##     ylim <- range(h$density, dx)
##     ylim <- ylim + diff(ylim) * c(0, 0.05)
##     tab <- cbind(c(Cp, Cp.l, Cp.u, Cp.k, Cpm), rbind(Cp.limits,
##         Cp.l.limits, Cp.u.limits, Cp.k.limits, Cpm.limits))
##     rownames(tab) <- c("Cp", "Cp_l", "Cp_u", "Cp_k", "Cpm")
##     colnames(tab) <- c("Value", names(Cp.limits))
##     oldpar <- par(bg = qcc.options("bg.margin"), cex = qcc.options("cex"),
##         mar = if (add.stats)
##             c(9 + is.null(center) * -1, 2, 4, 2) + 0.1
##         else par("mar"), no.readonly = TRUE)
##     if (restore.par)
##         on.exit(par(oldpar))
##     plot(0, 0, type = "n", xlim = xlim, ylim = ylim, axes = FALSE,
```

```
##         ylab = "", xlab = "", main = title)
##     usr <- par()$usr
##     rect(usr[1], usr[3], usr[2], usr[4], col = qcc.options("bg.figure"))
##     axis(1)
##     box()
##     plot(h, add = TRUE, freq = FALSE)
##     abline(v = c(LSL, USL), col = 2, lty = 3, lwd = 2)
##     text(LSL, usr[4], "LSL", pos = 1, col = "darkgray", cex = 0.8)
##     text(USL, usr[4], "USL", pos = 1, col = "darkgray", cex = 0.8)
##     if (!is.null(target)) {
##         abline(v = target, col = 2, lty = 2, lwd = 2)
##         text(target, usr[4], "Target", pos = 1, col = "darkgray",
##             cex = 0.8)
##     }
##     lines(xx, dx, lty = 2)
##     if (add.stats) {
##         plt <- par()$plt
##         px <- diff(usr[1:2])/diff(plt[1:2])
##         xfig <- c(usr[1] - px * plt[1], usr[2] + px * (1 - plt[2]))
##         at.col <- xfig[1] + diff(xfig[1:2]) * c(0.07, 0.35, 0.56,
##             0.75)
##         mtext(paste("Number of obs = ", n, sep = ""), side = 1,
##             line = 3, adj = 0, at = at.col[1], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##         mtext(paste("Center = ", signif(center, options()$digits),
##             sep = ""), side = 1, line = 4, adj = 0, at = at.col[1],
##             font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
##         mtext(paste("StdDev = ", signif(std.dev, options()$digits),
##             sep = ""), side = 1, line = 5, adj = 0, at = at.col[1],
##             font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
##         if (!is.null(target))
##             msg <- paste("Target = ", signif(target, options()$digits),
##                 sep = "")
##         else msg <- paste("Target = ", sep = "")
##         mtext(msg, side = 1, line = 3, adj = 0, at = at.col[2],
##             font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
##         mtext(paste("LSL = ", signif(LSL, options()$digits),
##             sep = ""), side = 1, line = 4, adj = 0, at = at.col[2],
##             font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
##         mtext(paste("USL = ", signif(USL, options()$digits),
##             sep = ""), side = 1, line = 5, adj = 0, at = at.col[2],
##             font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
##         mtext(paste("Cp     = ", signif(Cp, 3), sep = ""), side = 1,
##             line = 3, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##         mtext(paste("Cp_l  = ", signif(Cp.l, 3), sep = ""), side = 1,
##             line = 4, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##         mtext(paste("Cp_u = ", signif(Cp.u, 3), sep = ""), side = 1,
##             line = 5, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##         mtext(paste("Cp_k = ", signif(Cp.k, 3), sep = ""), side = 1,
##             line = 6, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
```

```
##         if (!is.null(target))
##             mtext(paste("Cpm  = ", signif(Cpm, 3), sep = ""),
##                 side = 1, line = 7, adj = 0, at = at.col[3],
##                 font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
##         mtext(paste("Exp<LSL ", signif(exp.LSL, 2), "%", sep = ""),
##             side = 1, line = 3, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##         mtext(paste("Exp>USL ", signif(exp.USL, 2), "%", sep = ""),
##             side = 1, line = 4, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##         mtext(paste("Obs<LSL ", signif(obs.LSL, 2), "%", sep = ""),
##             side = 1, line = 5, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##         mtext(paste("Obs>USL ", signif(obs.USL, 2), "%", sep = ""),
##             side = 1, line = 6, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
##             cex = qcc.options("cex.stats"))
##     }
##     if (print) {
##         cat("\nProcess Capability Analysis\n")
##         cat("\nCall:\n", deparse(match.call()), "\n\n", sep = "")
##         cat(paste(formatC("Number of obs = ", width = 16), formatC(n,
##             width = 12, flag = "-"), formatC("Target = ", width = 10),
##             formatC(target, digits = options()$digits, flag = "-"),
##             "\n", sep = ""))
##         cat(paste(formatC("Center = ", width = 16), formatC(center,
##             digits = options()$digits, width = 12, flag = "-"),
##             formatC("LSL = ", width = 10), formatC(LSL, digits = options()$digits,
##                 flag = "-"), "\n", sep = ""))
##         cat(paste(formatC("StdDev = ", width = 16), formatC(std.dev,
##             digits = options()$digits, width = 12, flag = "-"),
##             formatC("USL = ", width = 10), formatC(USL, digits = options()$digits,
##                 flag = "-"), "\n", sep = ""))
##         cat("\nCapability indices:\n\n")
##         print(tab, digits = 4, na.print = "", print.gap = 2)
##         cat("\n")
##         cat(paste("Exp<LSL ", format(exp.LSL, digits = 2), "%   ",
##             "Obs<LSL ", format(obs.LSL, digits = 2), "% \n",
##             sep = ""))
##         cat(paste("Exp>USL ", format(exp.USL, digits = 2), "%   ",
##             "Obs>USL ", format(obs.USL, digits = 2), "% \n",
##             sep = ""))
##     }
##     invisible(list(nobs = n, center = center, std.dev = std.dev,
##         target = target, spec.limits = {
##             sl <- c(LSL, USL)
##             names(sl) <- c("LSL", "USL")
##             sl
##         }, indices = tab, exp = {
##             exp <- c(exp.LSL, exp.USL)/100
##             names(exp) <- c("< LSL", "> USL")
##             exp
##         }, obs = {
##             obs <- c(obs.LSL, obs.USL)/100
##             names(obs) <- c("< LSL", "> USL")
```

```
##             obs
##         }))
## }
## <bytecode: 0x1a3b7d8>
## <environment: namespace:qcc>
```

And edit the line where the `hist` function is called

```
process.capability2 <- function (object, spec.limits, target, std.dev, nsigmas, confidence.level = 0.95
    breaks = "scott", add.stats = TRUE, print = TRUE, restore.par = TRUE)
{
    if ((missing(object)) | (!inherits(object, "qcc")))
        stop("an object of class 'qcc' is required")
    if (!(object$type == "xbar" | object$type == "xbar.one"))
        stop("Process Capability Analysis only available for charts type \"xbar\" and \"xbar.one\" char
    x <- as.vector(object$data)
    x <- x[!is.na(x)]
    sizes <- object$sizes
    center <- object$center
    if (missing(std.dev))
        std.dev <- object$std.dev
    n <- length(x)
    title <- paste("Process Capability Analysis\nfor", object$data.name)
    if (missing(spec.limits))
        stop("specification limits must be provided")
    if (!length(spec.limits) == 2)
        stop("wrong specification limits format")
    LSL <- min(spec.limits, na.rm = TRUE)
    USL <- max(spec.limits, na.rm = TRUE)
    if (missing(target)) {
        target <- mean(spec.limits, na.rm = TRUE)
    }
    if (target < LSL | target > USL)
        warning("target value is not within specification limits...")
    if (missing(nsigmas))
        if (is.null(object$nsigmas))
            stop("nsigmas not available in the 'qcc' object. Please provide nsigmas.")
        else nsigmas <- object$nsigmas
    if (confidence.level < 0 | confidence.level > 1)
        stop("the argument confidence.level must be a value between 0 and 1")
    Cp <- (USL - LSL)/(2 * nsigmas * std.dev)
    Cp.u <- (USL - center)/(nsigmas * std.dev)
    Cp.l <- (center - LSL)/(nsigmas * std.dev)
    Cp.k <- min(Cp.u, Cp.l)
    Cpm <- Cp/sqrt(1 + ((center - target)/std.dev)^2)
    alpha <- 1 - confidence.level
    Cp.limits <- Cp * sqrt(qchisq(c(alpha/2, 1 - alpha/2), n -
        1)/(n - 1))
    Cp.u.limits <- Cp.u * (1 + c(-1, 1) * qnorm(confidence.level) *
        sqrt(1/(9 * n * Cp.u^2) + 1/(2 * (n - 1))))
    Cp.l.limits <- Cp.l * (1 + c(-1, 1) * qnorm(confidence.level) *
        sqrt(1/(9 * n * Cp.l^2) + 1/(2 * (n - 1))))
    Cp.k.limits <- Cp.k * (1 + c(-1, 1) * qnorm(1 - alpha/2) *
        sqrt(1/(9 * n * Cp.k^2) + 1/(2 * (n - 1))))
```

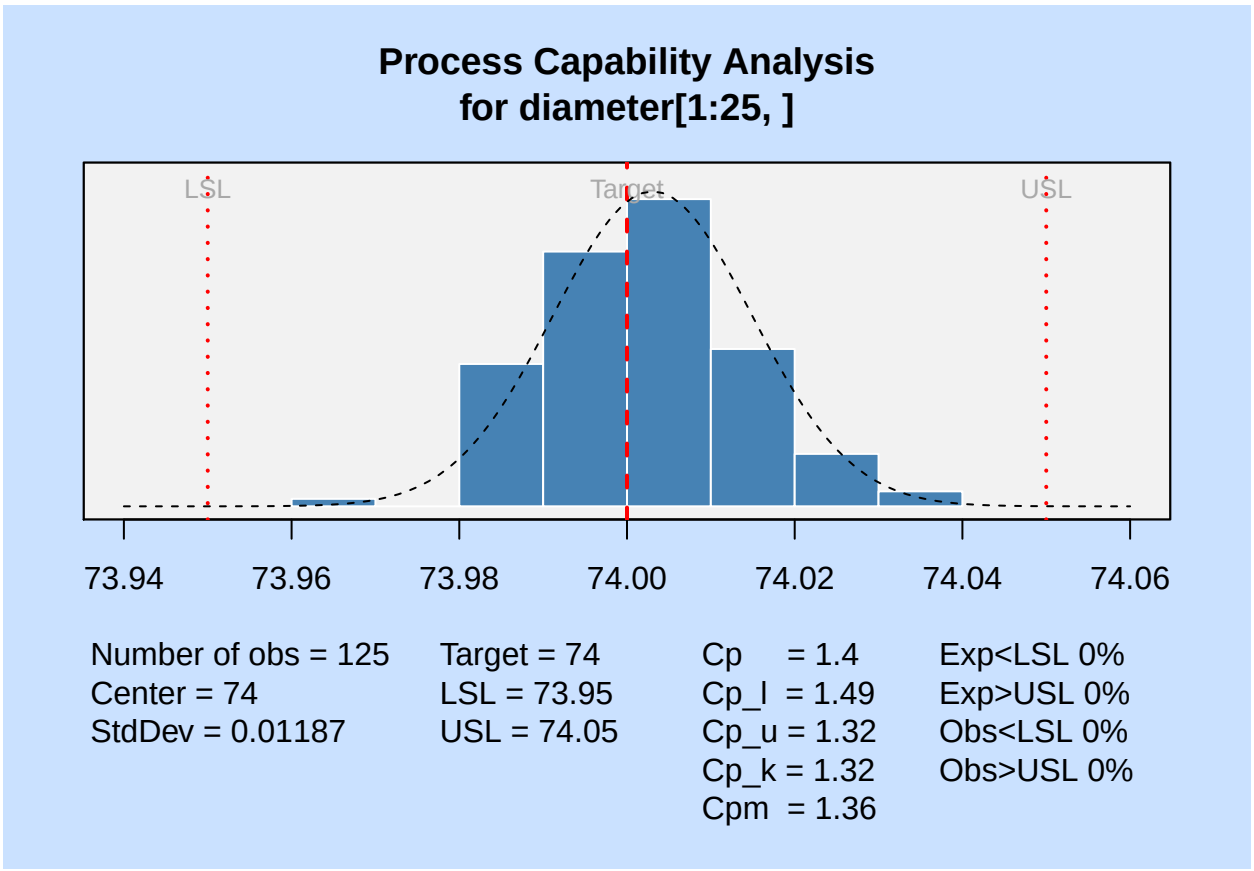
```
    df <- n * (1 + ((center - target)/std.dev)^2)/(1 + 2 * ((center -
        target)/std.dev)^2)
    Cpm.limits <- Cpm * sqrt(qchisq(c(alpha/2, 1 - alpha/2),
        df)/df)
    names(Cp.limits) <- names(Cp.k.limits) <- names(Cpm.limits) <- c(paste(round(100 *
        alpha/2, 1), "%", sep = ""), paste(round(100 * (1 - alpha/2),
        1), "%", sep = ""))
    exp.LSL <- pnorm((LSL - center)/std.dev) * 100
    if (exp.LSL < 0.01)
        exp.LSL <- 0
    exp.USL <- (1 - pnorm((USL - center)/std.dev)) * 100
    if (exp.USL < 0.01)
        exp.USL <- 0
    obs.LSL <- sum(x < LSL)/n * 100
    obs.USL <- sum(x > USL)/n * 100
    xlim <- range(x, USL, LSL, target)
    xlim <- xlim + diff(xlim) * c(-0.1, 0.1)
    xx <- seq(min(xlim), max(xlim), length = 100)
    dx <- dnorm(xx, center, std.dev)
    h <- hist(x, breaks = breaks, plot = FALSE)
    ylim <- range(h$density, dx)
    ylim <- ylim + diff(ylim) * c(0, 0.05)
    tab <- cbind(c(Cp, Cp.l, Cp.u, Cp.k, Cpm), rbind(Cp.limits,
        Cp.l.limits, Cp.u.limits, Cp.k.limits, Cpm.limits))
    rownames(tab) <- c("Cp", "Cp_l", "Cp_u", "Cp_k", "Cpm")
    colnames(tab) <- c("Value", names(Cp.limits))
    oldpar <- par(bg = qcc.options("bg.margin"), cex = qcc.options("cex"),
        mar = if (add.stats)
            c(9 + is.null(center) * -1, 2, 4, 2) + 0.1
        else par("mar"), no.readonly = TRUE)
    if (restore.par)
        on.exit(par(oldpar))
    plot(0, 0, type = "n", xlim = xlim, ylim = ylim, axes = FALSE,
        ylab = "", xlab = "", main = title)
    usr <- par()$usr
    rect(usr[1], usr[3], usr[2], usr[4], col = qcc.options("bg.figure"))
    axis(1)
    box()
#######################################################
##### EDITED
    plot(h, add = TRUE, freq = FALSE,
         col = "steelblue",
         border = "white")
#####
#######################################################
    abline(v = c(LSL, USL), col = 2, lty = 3, lwd = 2)
    text(LSL, usr[4], "LSL", pos = 1, col = "darkgray", cex = 0.8)
    text(USL, usr[4], "USL", pos = 1, col = "darkgray", cex = 0.8)
    if (!is.null(target)) {
        abline(v = target, col = 2, lty = 2, lwd = 2)
        text(target, usr[4], "Target", pos = 1, col = "darkgray",
            cex = 0.8)
    }
```

```
    lines(xx, dx, lty = 2)
    if (add.stats) {
        plt <- par()$plt
        px <- diff(usr[1:2])/diff(plt[1:2])
        xfig <- c(usr[1] - px * plt[1], usr[2] + px * (1 - plt[2]))
        at.col <- xfig[1] + diff(xfig[1:2]) * c(0.07, 0.35, 0.56,
            0.75)
        mtext(paste("Number of obs = ", n, sep = ""), side = 1,
            line = 3, adj = 0, at = at.col[1], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
        mtext(paste("Center = ", signif(center, options()$digits),
            sep = ""), side = 1, line = 4, adj = 0, at = at.col[1],
            font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
        mtext(paste("StdDev = ", signif(std.dev, options()$digits),
            sep = ""), side = 1, line = 5, adj = 0, at = at.col[1],
            font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
        if (!is.null(target))
            msg <- paste("Target = ", signif(target, options()$digits),
                sep = "")
        else msg <- paste("Target = ", sep = "")
        mtext(msg, side = 1, line = 3, adj = 0, at = at.col[2],
            font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
        mtext(paste("LSL = ", signif(LSL, options()$digits),
            sep = ""), side = 1, line = 4, adj = 0, at = at.col[2],
            font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
        mtext(paste("USL = ", signif(USL, options()$digits),
            sep = ""), side = 1, line = 5, adj = 0, at = at.col[2],
            font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
        mtext(paste("Cp     = ", signif(Cp, 3), sep = ""), side = 1,
            line = 3, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
        mtext(paste("Cp_l   = ", signif(Cp.l, 3), sep = ""), side = 1,
            line = 4, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
        mtext(paste("Cp_u = ", signif(Cp.u, 3), sep = ""), side = 1,
            line = 5, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
        mtext(paste("Cp_k = ", signif(Cp.k, 3), sep = ""), side = 1,
            line = 6, adj = 0, at = at.col[3], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
        if (!is.null(target))
            mtext(paste("Cpm  = ", signif(Cpm, 3), sep = ""),
                side = 1, line = 7, adj = 0, at = at.col[3],
                font = qcc.options("font.stats"), cex = qcc.options("cex.stats"))
        mtext(paste("Exp<LSL ", signif(exp.LSL, 2), "%", sep = ""),
            side = 1, line = 3, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
        mtext(paste("Exp>USL ", signif(exp.USL, 2), "%", sep = ""),
            side = 1, line = 4, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
        mtext(paste("Obs<LSL ", signif(obs.LSL, 2), "%", sep = ""),
            side = 1, line = 5, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
```

```
        mtext(paste("Obs>USL ", signif(obs.USL, 2), "%", sep = ""),
            side = 1, line = 6, adj = 0, at = at.col[4], font = qcc.options("font.stats"),
            cex = qcc.options("cex.stats"))
    }
    if (print) {
        cat("\nProcess Capability Analysis\n")
        cat("\nCall:\n", deparse(match.call()), "\n\n", sep = "")
        cat(paste(formatC("Number of obs = ", width = 16), formatC(n,
            width = 12, flag = "-"), formatC("Target = ", width = 10),
            formatC(target, digits = options()$digits, flag = "-"),
            "\n", sep = ""))
        cat(paste(formatC("Center = ", width = 16), formatC(center,
            digits = options()$digits, width = 12, flag = "-"),
            formatC("LSL = ", width = 10), formatC(LSL, digits = options()$digits,
                flag = "-"), "\n", sep = ""))
        cat(paste(formatC("StdDev = ", width = 16), formatC(std.dev,
            digits = options()$digits, width = 12, flag = "-"),
            formatC("USL = ", width = 10), formatC(USL, digits = options()$digits,
                flag = "-"), "\n", sep = ""))
        cat("\nCapability indices:\n\n")
        print(tab, digits = 4, na.print = "", print.gap = 2)
        cat("\n")
        cat(paste("Exp<LSL ", format(exp.LSL, digits = 2), "%   ",
            "Obs<LSL ", format(obs.LSL, digits = 2), "% \n",
            sep = ""))
        cat(paste("Exp>USL ", format(exp.USL, digits = 2), "%   ",
            "Obs>USL ", format(obs.USL, digits = 2), "% \n",
            sep = ""))
    }
    invisible(list(nobs = n, center = center, std.dev = std.dev,
        target = target, spec.limits = {
            sl <- c(LSL, USL)
            names(sl) <- c("LSL", "USL")
            sl
        }, indices = tab, exp = {
            exp <- c(exp.LSL, exp.USL)/100
            names(exp) <- c("< LSL", "> USL")
            exp
        }, obs = {
            obs <- c(obs.LSL, obs.USL)/100
            names(obs) <- c("< LSL", "> USL")
            obs
        }))
}
```

Now you can get the plot with your colours:

```
process.capability2(q, spec.limits=c(73.95,74.05))
```




```
##
## Process Capability Analysis
##
## Call:
## process.capability2(object = q, spec.limits = c(73.95, 74.05))
##
## Number of obs = 125          Target = 74
##        Center = 74              LSL = 73.95
##        StdDev = 0.01187         USL = 74.05
##
## Capability indices:
##
##       Value   2.5%  97.5%
## Cp    1.405  1.230  1.579
## Cp_l  1.490  1.327  1.653
## Cp_u  1.319  1.173  1.465
## Cp_k  1.319  1.145  1.493
## Cpm   1.360  1.187  1.534
##
## Exp<LSL 0%   Obs<LSL 0%
## Exp>USL 0%   Obs>USL 0%
```