Formatting qicharts2

Calum Polwart

Brief background

I was asked recently about the best visualisation for a quality improvement project. This was a project that had aimed to reduce the frequency of errors detected by a pharmacy team. This was a specific error, and so would be relatively rare. With a common event, you can easily plot something like daily intervention rates and for me I'd usually suggest using the NHS Plot the Dots[1] package as it has a lot of ability to customise and is really rather well written. But for rare events a time between events plot (often called a 'T' plot) is much better. This is the forgotten cousin in SPC charting, and often omitted from the most common SPC tools. NHSRplotthedots can't do 't' plots. But an R package called qicharts can. qichart appears to have been superseded by qicharts2[2] which produces a ggplot2 object which, I thought at least, would mean we could style the graphics more easily using the ggplot grammar of graphics.

Difference between a standard SPC and T plot

This seems limited in description, but the control lines are calculated in a different way. A moving average is used instead of a mean/median. This means you cant just relabel the existing SPC charts other tools make. There is a rather nice NHS PDF available here that outlines the manual process to build the data.

A T plot is plotting the time (usually in days) between events happening. If you introduce a new process to reduce errors, you'd hope to see the time between events **increase**. It is as simple as that. The rest of the SPC rules - more than 6 values above the control line a value outside of the 3 sigma bands etc are the same as normal. Although I've not seen it written down, I think 'impossible' values (less than zero) are also not shown in the sigma ranges. And certainly the tool all check that the time between events isn't zero. If you have two events on the same day, you need to use a fraction of a day to capture the second event. (If that happened lots you might be better using hours as a measure instead of days).

Lets create a very simple example

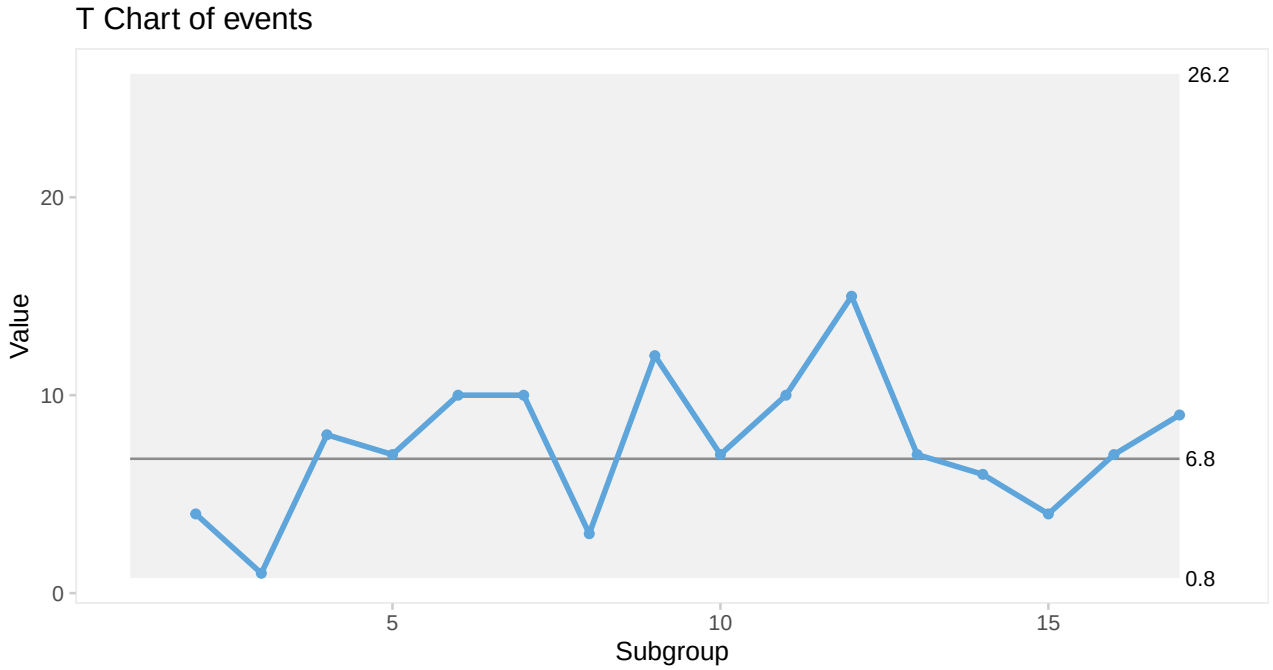
We might as well use the example data provided on that rather nice NHS PDF document. It looks to be measuring time between falls.

```
# Create an object with the dates of the falls
fallsDates <- as.Date(c(
    "2014-03-02", "2014-03-06", "2014-03-07", "2014-03-15", "2014-03-22",
    "2014-04-01", "2014-04-11", "2014-04-14", "2014-04-26",
    "2014-05-03", "2014-05-13", "2014-05-28",
    "2014-06-04", "2014-06-10", "2014-06-14",  "2014-06-21", "2014-06-30" ))

# Calculate the time between events
events <- c(NA, diff(fallsDates))
```

The NHS PDF has one feature I don't like, it uses dates along the x-axis. I don't think that's great practice, as dates on an x-axis should usually be on a time scale and these are actually the specific dates of events. There may be times that's useful - such as picking out events around holiday periods etc. But usually the x-axis for a t-plot should be a sequential line of event numbers.

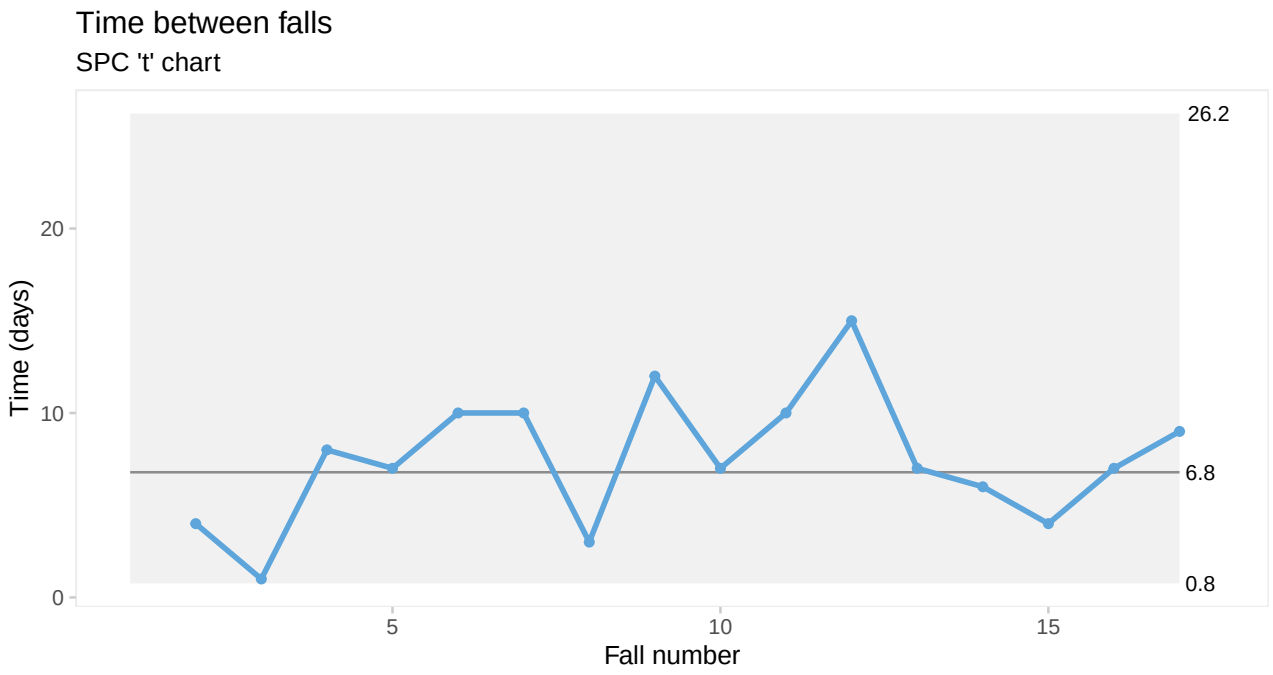
In its simplest form qiplots2 will give as a t-chart with just this code:

```
require(qicharts2)

qic(
    y = events,
    x = 1:length(events),
    chart = "t"
)
```



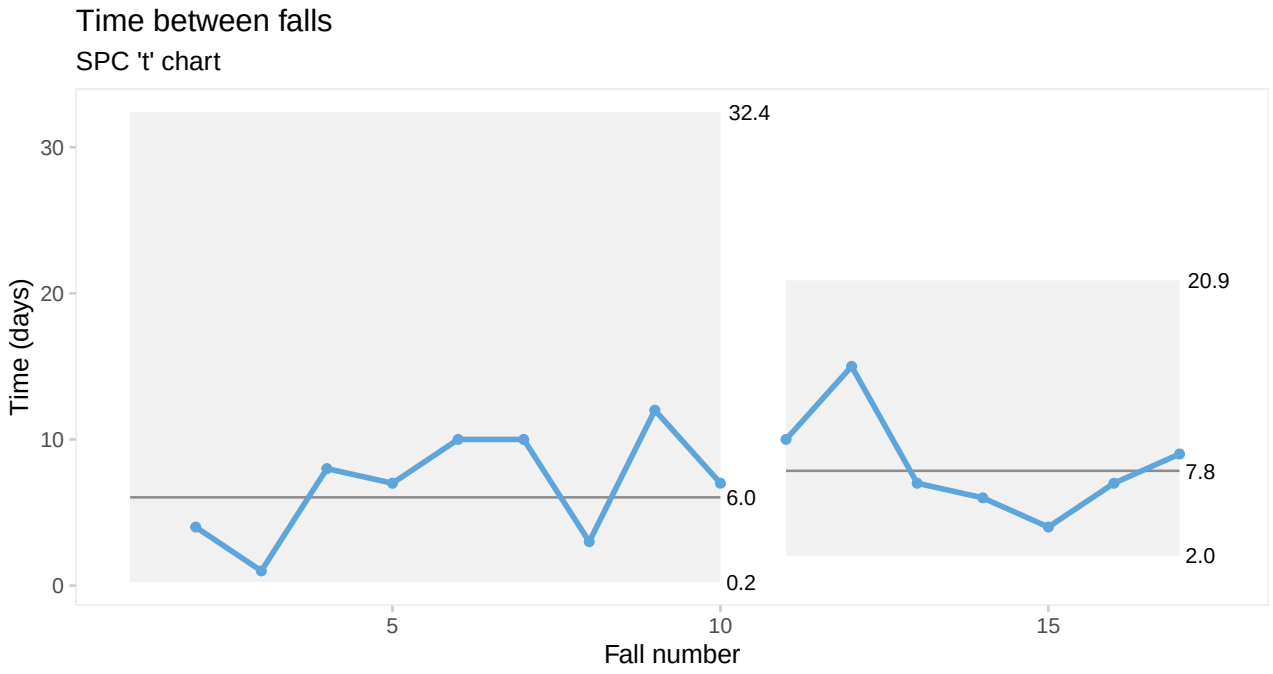
There are clearly documented parameters to change titles, axis labels etc. So you can quickly move to a graph like this:

```
qic(
    y = events,
    x = 1:length(events),
    chart = "t",
    title = "Time between falls",
    subtitle = "SPC 't' chart",
    xlab = "Fall number",
    ylab = "Time (days)"
)
```



And you can either freeze data or split the data at a certain point if you want to see if an intervention is helping. For the sake of demonstration we will assume some falls reduction intervention was introduced after the 10th fall.

```
qic(
    y = events,
    x = 1:length(events),
    chart = "t",
    title = "Time between falls",
    subtitle = "SPC 't' chart",
    xlab = "Fall number",
    ylab = "Time (days)",
    part = 10
)
```



Formatting the result

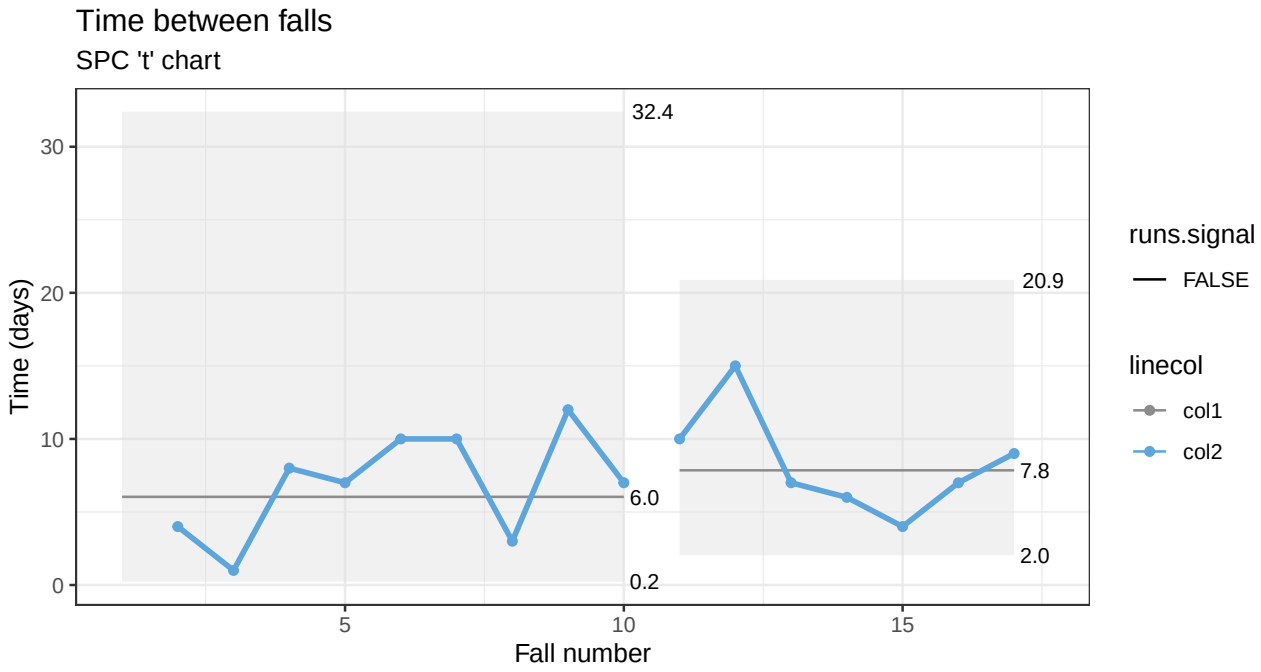
This is a ggplot, but it hasn't been created in a conventional ggplot way. That is you **are not** entering code along the lines of:

```
data |>
    ggplot2(aes(x = .., y = ..)) +
    geom_qichart()
```

If you were, your formatting would usually be controlled in the geom. So what can we do?

Well firstly, it is a ggplot object. So you can apply ggplot theme actions. For example:

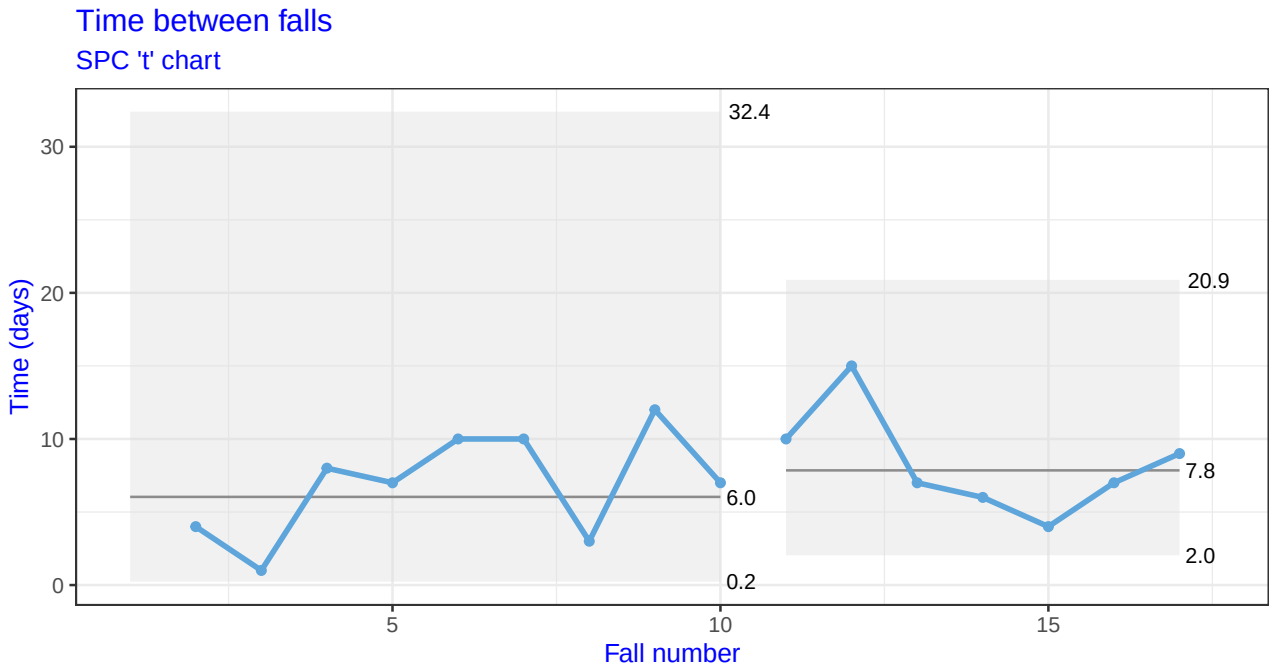
```
require(ggplot2)
qic(
    y = events,
    x = 1:length(events),
    chart = "t",
    title = "Time between falls",
    subtitle = "SPC 't' chart",
    xlab = "Fall number",
    ylab = "Time (days)",
    part = 10
) +
    theme_bw()
```



Adding the theme, means if the theme has a legend it will appear, but we can turn that off as usual with a ggplot theme legend position none statement. All this means we can edit the plotting canvas, the size of text, the fonts, etc. But the actual data lines and the grey shaded 3 sigma zones (produced with a geom_ribbon are beyond easy reach).

```
qic(
    y = events,
    x = 1:length(events),
    chart = "t",
```

```
    title = "Time between falls",
    subtitle = "SPC 't' chart",
    xlab = "Fall number",
    ylab = "Time (days)",
    part = 10
) +
    theme_bw() +
    theme(text = element_text(colour = "blue") ,
             legend.position = "none")
```



Formatting the lines and shading

I couldn't see an easy way to change the lines and shading. It might be possible by hacking a grob produced from the plot but frankly that felt like far too much effort. I might as well try and draw the graph from the raw data! This is open source software - which means we can see the code that produces the graph and thats always worth a look in these situations. The key functionfor plotting lies here.

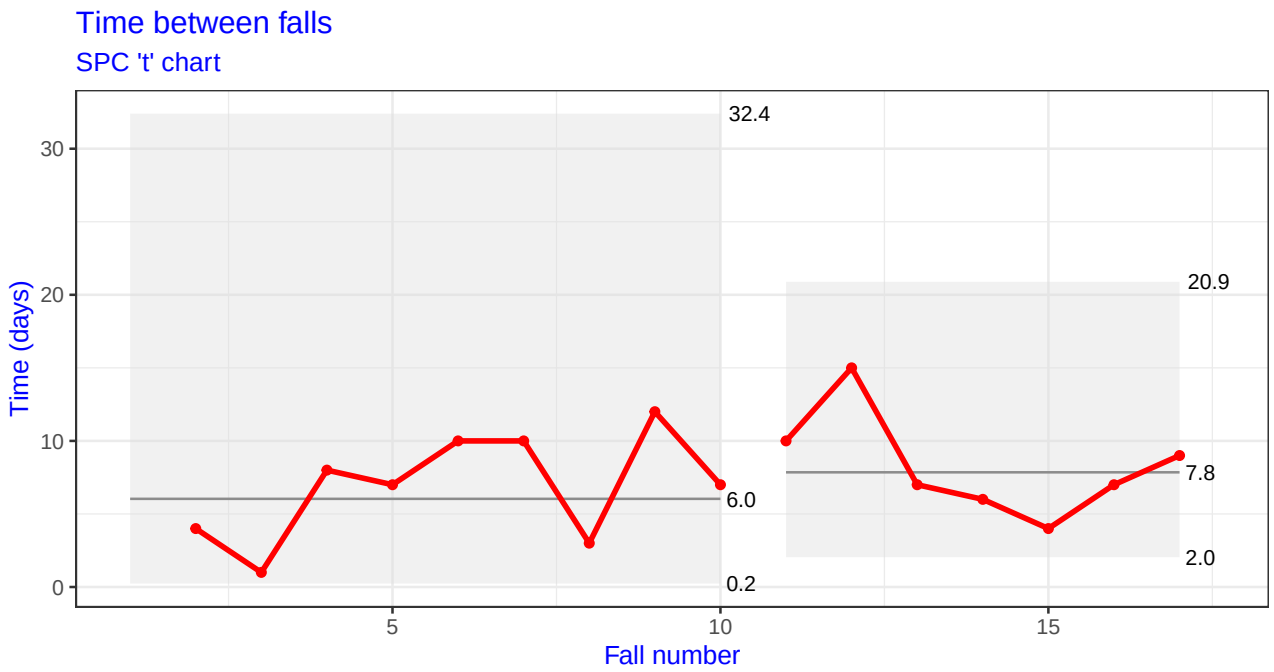
```
plot.qic <- function(x, title, ylab, xlab, subtitle, caption, part.labels,
                     n.facets,
                     nrow, ncol, scales, show.labels, show.grid, show.95,
                     decimals, flip, dots.only, point.size,
                     x.format, x.angle, x.pad,
                     y.expand, y.percent, y.percent.accuracy, strip.horizontal,
                     # col.line = '#5DA5DA',
                     # col.signal = '#F15854',
                     # col.target = '#059748',
                     ...)
```

Which in its-self is interesting. Because there are three commented out variables in the function which may be a future development.

```
  col1      <- '#8C8C8C' # rgb(140, 140, 140, maxColorValue = 255) # grey
  col2      <- getOption('qic.linecol', default = '#5DA5DA')       # blue
  # col3      <- getOption('qic.signalcol', default = '#F15854')     # red
  col3      <- getOption('qic.signalcol', default = '#FAA43A')     # amber
  col4      <- getOption('qic.targetcol', default = '#059748')     # green
  col5      <- '#C8C8C8' # rgb(200, 200, 200, maxColorValue = 255) # light grey
  cols      <- c('col1' = col1,
                 'col2' = col2,
                 'col3' = col3,
                 'col4' = col4,
                 'col5' = col5)
```

So, actually - the plotting function is looking up its colours from a set of options called qic.[variable] and applying some defaults if these are not set. That means we can set those variables and change the colours.

```
currentOptions <- options()
options(qic.linecol = "red")

qic(
    y = events,
    x = 1:length(events),
    chart = "t",
    title = "Time between falls",
    subtitle = "SPC 't' chart",
    xlab = "Fall number",
    ylab = "Time (days)",
    part = 10
) +
    theme_bw() +
    theme(text = element_text(colour = "blue") ,
              legend.position = "none")
```

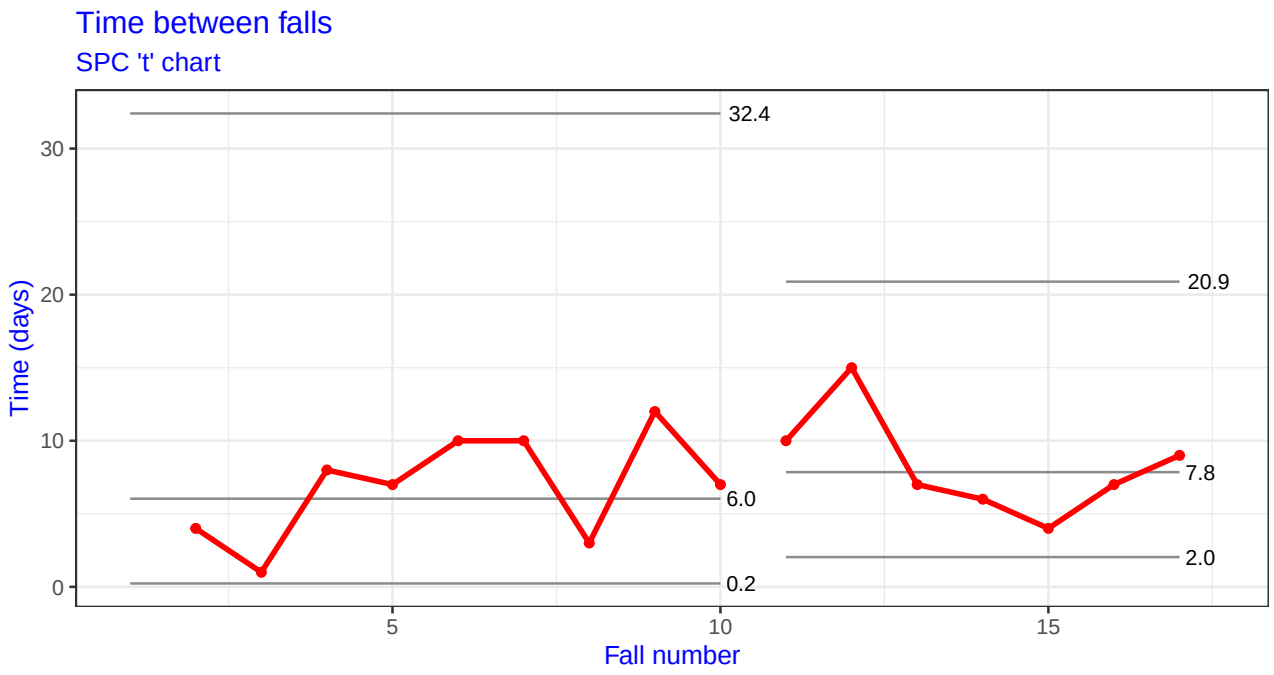


That leaves just the ribbon to try and edit.

```
# Add control limits and centre and target lines
  if (getOption('qic.clshade', default = TRUE) &
      sum(!is.na(x$lcl))) {
    p <- p +
      geom_ribbon(aes_(ymin = ~ lcl, ymax = ~ ucl),
                  fill = 'grey87',
                  alpha = 0.4)
  } else {
    p <- p +
      geom_line(aes_(y = ~ lcl), colour = col1, na.rm = T)

    p <- p +
      geom_line(aes_(y = ~ ucl), colour = col1, na.rm = T)
  }
```

So there is a further option, completely undocumented as far as I can see, that switches on / off the shading.

```
currentOptions <- options()
options(qic.linecol = "red", qic.clshade = F)

qic(
    y = events,
    x = 1:length(events),
    chart = "t",
    title = "Time between falls",
    subtitle = "SPC 't' chart",
    xlab = "Fall number",
    ylab = "Time (days)",
    part = 10
) +
    theme_bw() +
    theme(text = element_text(colour = "blue") ,
             legend.position = "none")
```



And there we have it. A chart with line colours edited, text adjusted and the shading removed. Why the package authors didn't do it with geom_qichart I don't quite know. But if you need a customisable qichart for a time series, I hope these notes help.

References

1. Reading, C., Wellesley-Miller, S., Turner, Z., Jemmett, T., Smith, T., Mainey, C., & MacKintosh, J. (2021). *NHSRplotthedots: Draw XmR charts for NHSE/i 'making data count' programme*. https://CRAN.R-project.org/package=NHSRplotthedots
2. Anhoej, J. (2024). *qicharts2: Quality improvement charts*. https://CRAN.R-project.org/package=qicharts2