Statistical Process Control in Healthcare with R

Dwight Barry, Brendan Bettinger, and Andrew Cooper

October 2018

Contents

Chapter 1

Overview

Statistical process control (SPC) was a triumph of manufacturing analytics, and its success spread across a variety of industries—most improbably, into healthcare.

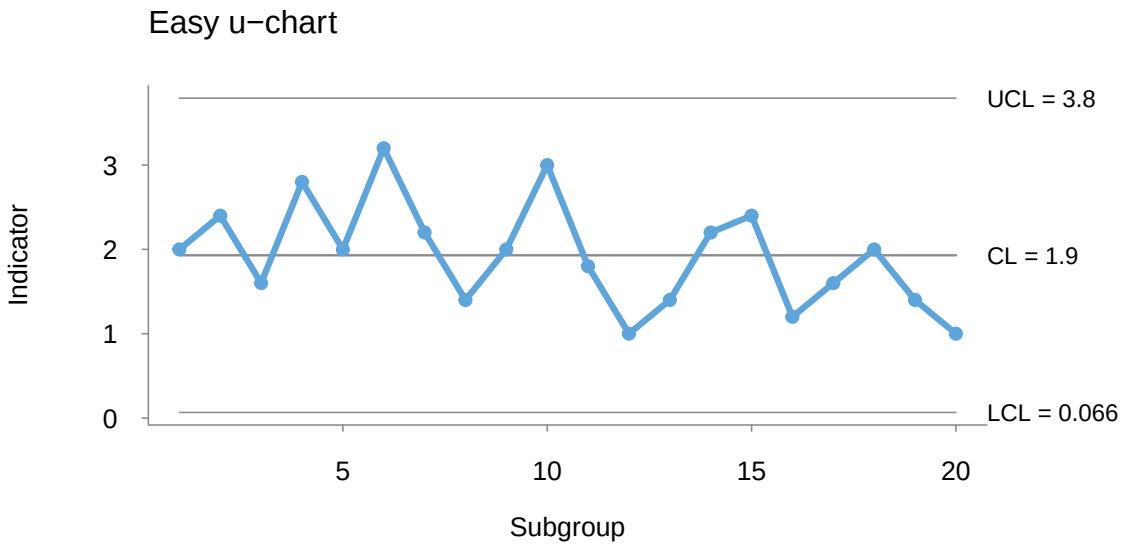
Healthcare is rarely compatible with the idea of an assembly line, but lean manufacturing thinking ("Lean") has taken over healthcare management around the world, and SPC methods are common tools in Lean. Unlike in manufacturing, stability is an inherently tricky concept in healthcare, so this has led to much *misuse* of these methods. Bad methods lead to bad inferences, and bad inferences can lead to poor decisions. This book aims to help analysts apply SPC methods more accurately in healthcare, using the statistical software R.

1.1 If you've never used R

Some BI analysts are apprehensive about getting into R, but if you've ever written a line of SQL or created a formula in an Excel cell, this is no different in concept. Yes, the R language is full of idiosyncracies and outright annoyances, but when you need to accomplish particular goals, it can be fairly easy.

For example, you can create a *u*-chart with only three lines of code, start to finish–load the package, load the data, create the plot:

```
# Load the qicharts package, a simple interface to SPC charts
library(qicharts2)

# Load some example data from another R package as an example
data(pcmanufact, package = "qcc")

# You can look at the data by clicking on the spreadsheet button in the Environment tab,
# or by running `View(pcmanufact)` in the console

# Create the u-chart
qicharts::qic(y = pcmanufact$x, n = pcmanufact$size, chart = "u", main = "Easy u-chart")
```



You can find help in R using `?`, followed by the function name, e.g.,

```
?qic
```

1.2 What you need

We assume that users of this book will already be familiar with basic SPC methods and concepts. We do cover some basics, but we focus primarily on the areas that cause the most misunderstandings and misuse; Chapter 13, Useful References, provides a great place to start or continue learning about SPC.

We don't presume familiarity with R, though of course everything's easier if you've used R before. If you haven't, here's what you need to get started:

- You can download R from https://cran.r-project.org/.
- You can download RStudio from https://www.rstudio.com/products/rstudio/download/.

Open RStudio and install the packages used in this book by copying and pasting this code into the **Console**:

```
install.packages("ggplot2", "forecast", "fpp2", "ggExtra", "ggseas", "gridExtra", "tidyverse",
                 "qcc", "qicharts2", "scales", dependencies = TRUE)
```

This book was created using R version 3.5.1 and RStudio 1.1.456. Code was tested on Mac OS 10.12.6 (aka Sierra).

1.3 Book repo

You can submit pull requests for any errors or typos at https://github.com/Rmadillo/spc_healthcare_with_r.

1.4 About

We are all analysts at *Seattle Children's Hospital* in Seattle, Washington, USA.

- Dwight Barry is a Principal Data Scientist in *Enterprise Analytics*. Twitter: **?**

- Andy Cooper is a Lead Data Scientist in *Enterprise Analytics*. Twitter: **?**

- Brendan Bettinger is a Senior Analyst in *Infection Prevention*.

Chapter 2

Signal, noise, and statistical process control

2.1 Signal and noise

People are really, really good at finding patterns that aren't real, especially in noisy data.

Every metric has natural variation—*noise*—included as an inherent part of that process. True signals only emerge when you have properly characterized that variation. Statistical process control (SPC) charts—run charts and control charts—help characterize and identify non-random patterns that suggest the process has changed.

In essence, SPC tools help you evaluate the stability and predictability of a process or its outcomes. Statistical theory provides the basis to evaluate metric stability and more confidently detect changes in the underlying process amongst the noise of natural variation. Since it is impossible to account for every single variable that might influence a metric, we can use probability and statistics to evaluate how that metric naturally fluctuates over time (aka common cause variation), and construct guidelines around that fluctuation to help indicate when something in that process has changed (special cause variation).

Understanding natural, random variation in time series or sequential data is the essential point of quality assurance or process and outcome improvement efforts. It's a rookie mistake to use SPC tools to focus solely on the values themselves or their central tendency—instead evaluate *all* of the elements of a run chart or control chart to understand what it's telling you. For example, Figure 1 shows a process created using random numbers based on a pre-defined normal distribution. The black points and line are the data itself, the overall mean ($y = 18$) is represented by the grey line, and the overall distribution is shown in a histogram to the right of the run chart.

The axis labels are traditional SPC labels. The *value* (i.e., metric) is on the y-axis and the units of observation (traditionally called *subgroups*) are on the x-axis. The term subgroup was developed in the context of an observation point involved sampling from a mechnical process, e.g., taking 5 widgets from a production of 500. Many SPC examples maintain this label regardless of what the x-axis is actually measuring for simplicity's sake—we follow this convention where appropriate.

Figure 1. A stable process created from random numbers.

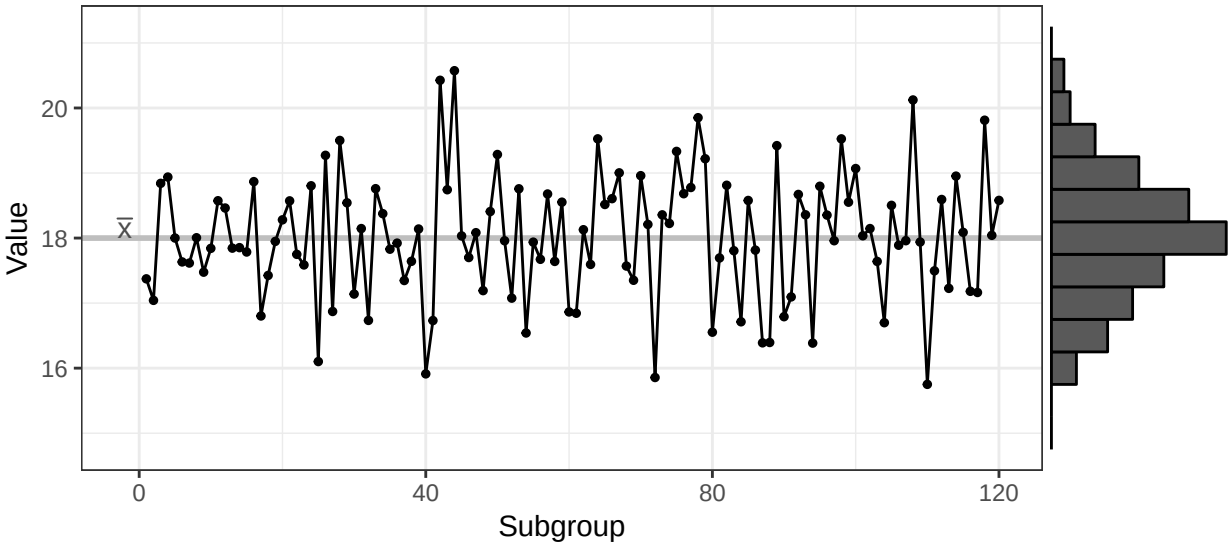
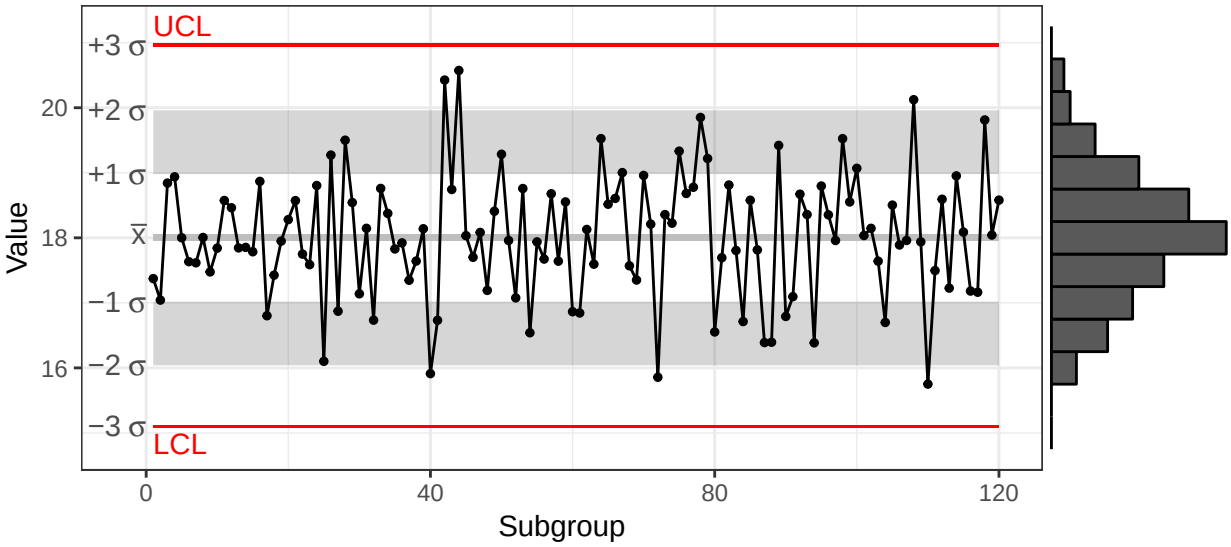


Figure 2 adds control limits and 1-2σ bands (Not sure I understand this notation without looking at the graph and thinking about it. E.g., One band is between 1σ and 2σ above the mean; the other is between 1σ and 2σ below the mean. Do we need to define sigma for the readers?), where σ is a measure of expected process standard devation, for reference. Guidelines on how to use these elements of SPC charts to evaluate the statistical process of a metric in more detail and determine whether to investigate the process for special cause variation are detailed in Chapter 4.

Figure 2. The same plot as in Figure 1, with standard deviation indicators and control limits added.



Note that the control chart guidelines suggest that some special cause variation has occurred in this data. Since this dataset was generated using random numbers from a known, stable, normal distribution, these are *False Positives*: the control chart suggests something has changed when in reality it hasn't.

There is always a chance for *False Negatives*, as well, where something actually happened but the control chart didn't alert you to special cause variation. Consider the matrix of possible outcomes for any given point in an SPC chart:

	Reality: Something Happened	Reality: Nothing Happened
SPC: Alert	True Positive	*False Positive*

	Reality: Something Happened	Reality: Nothing Happened
SPC: No alert	*False Negative*	True Negative

Using 3σ control limits is standard, intended to balance the trade-offs between *False Negatives* and *False Positives*. If you prefer to err on the side of caution for a certain metric (such as in monitoring hospital acquired infections) and are willing to accept more *False Positives* to reduce *False Negatives*, you could use 2σ control limits. For other metrics where you prefer to be completely certain things are out of whack before taking action (need example?) and are willing to accept more *False Negatives* you to reduce *False Positives*, you could use 4σ control limits. When in doubt, use 3σ control limits.

It's important to remember that SPC charts are at heart decision tools which can help you decide how to reduce false signals relative to your use case, but *they can never entirely eliminate false signals*. Thus, it's often useful to explicitly explore these trade-offs with stakeholders when deciding where and why to set control limits.

2.2 SPC tools

Run charts and control charts are the core tools of SPC analysis. Other basic statistical graphs—particularly line charts and histograms—are equally important to SPC work.

Line charts help you monitor any sort of metric, process, or time series data. Run charts and control charts are meant to help you identify departures from a **stable** process. Each uses a set of guidelines to help you make decisions on whether a process has changed or not.

In many cases, a run chart is all you need. In *all* cases, you should start with a line chart and histogram. If—and only if—the process is stable and you need to characterize the limits of natural variation, you can move on to using a control chart.

In addition, *never* rely on a table or year-to-date (YTD) comparisons to evaluate process performance. These approaches obscure the foundational concept of process control: that natural, common cause variation is an essential part of the process. Tables or YTD values can supplement run charts or control charts, but should never be used without them.

Above all, remember that the decisions you make in constructing SPC charts and associated data points (such as YTD figures) *will* impact the interpretation of the results. Bad charts can make for bad decisions.

2.3 Defining *stability*

It's common for stakeholders to want key performance indicators (KPIs) displayed using a control chart. However, control charts are only applicable when the business goal is to keep that KPI stable. SPC tools are built upon the fundamental assumption of a *stable* process, and as an analyst you need to be very clear on the definition of stability in the context of business goals and the statistical process of the metric itself. Because it takes time and resources to track KPIs (collecting the data, developing the dashboards, etc.) you should take time to develop them carefully by first ensuring that SPC tools are, in fact, an appropriate means to monitor that KPI.

In many cases when folks talk about "stability" they mean "constant", and they think of the goal behind the KPI as trying to keep the KPI at some fixed value or achieve some fixed target value. In many cases this makes sense, and a control chart would be appropriate. However, there are times where stability could have different meanings, particularly in a changing environment, and the KPI should be defined accordingly if a control chart is to be used. (Confused by this paragraph, but think maybe it could be deleted since the next two paragraphs are examples addressing the appropriateness of KPI control charts -BB)

For example, perhaps some outpatient specialties are facing increasing numbers of referrals but are not getting more FTEs. With increasing patient demand and constrained hospital capacity, we would not expect the process data (e.g., wait times for appointments) to be constant over time. So, a KPI such as "percent of new patients seen within 2 weeks" might be a goal we care about, but since we expect that value to decline, it is not stable and a control chart is not appropriate. However, if we define the KPI as something like "percent of new patients seen within 2 weeks relative to what we would expect given increased demand and no expansion", we have now placed it into a stable context. Instead of asking if the metric itself is declining, we're asking whether the system is responding as it has in the past. By defining the KPI in terms of something we would want to remain stable, we can now use a control chart to track its performance.

For another example, perhaps complaints about phone wait time for a call center has led to an increase in FTEs to support call demand. You would expect the call center performance—perhaps measured in terms of "percent of calls answered in under 2 minutes"—to improve, so a control chart is not appropriate. So, what would a "stable" call center KPI look like as they add FTEs? Maybe it could be the performance of the various teams within the call center become more similar (e.g., decreased variability across teams). Maybe it could be the frequency of catastrophic events (e.g., people waiting longer than X minutes, where X is very large) staying below some threshold—similar to a "downtime" KPI used to track the stability of computer systems. Maybe it could be the percent change in the previously-defined KPI tracking the percent change in FTEs (though we know this relationship is non-linear).

In both examples, it would not be appropriate to use a control chart for the previously-defined performance metrics, because we do not expect them (or necessarily want them) to be stable. However, by focusing on the process itself, we can define alternate KPIs that conform to the assumptions of a control chart.

Stability means that the system is responding as we would expect to the changing environment and that the system is robust to adverse surprises from the environment. **KPIs meant to evaluate stable processes should be specifically designed to track whether the system is stable and robust**, rather than focusing strictly on the outcome as defined by existing or previous KPIs.

Make sure that metrics meant to measure stability are properly designed from the outset before you spend large amounts of resources to develop and track them.

Chapter 3

Where to start

R has been used by statisticians and data scientists for years, but it is rapidly becoming an essential tool in business intelligence as well. Creating SPC charts can take hours in Excel or Tableau (and can be quite error-prone), but they can be created in seconds with a line or two of R code.

3.1 R packages

This book uses the following R packages:

```
library(ggplot2)     # for general plotting
library(lubridate)   # for easier date/time casting
library(forecast)    # for plotting and forecasts
library(qicharts2)   # for simple run charts and control charts
library(seasonal)    # for seasonal adjusment calculations
library(ggseas)      # for on-the-fly seasonal adjustment plotting
library(ggExtra)     # for making line+histogram marginal plots
library(gridExtra)   # for creating multi-graph plots
```

We'll use fake data throughout this book; below are some data sets that we'll use in this chapter and in a few places later:

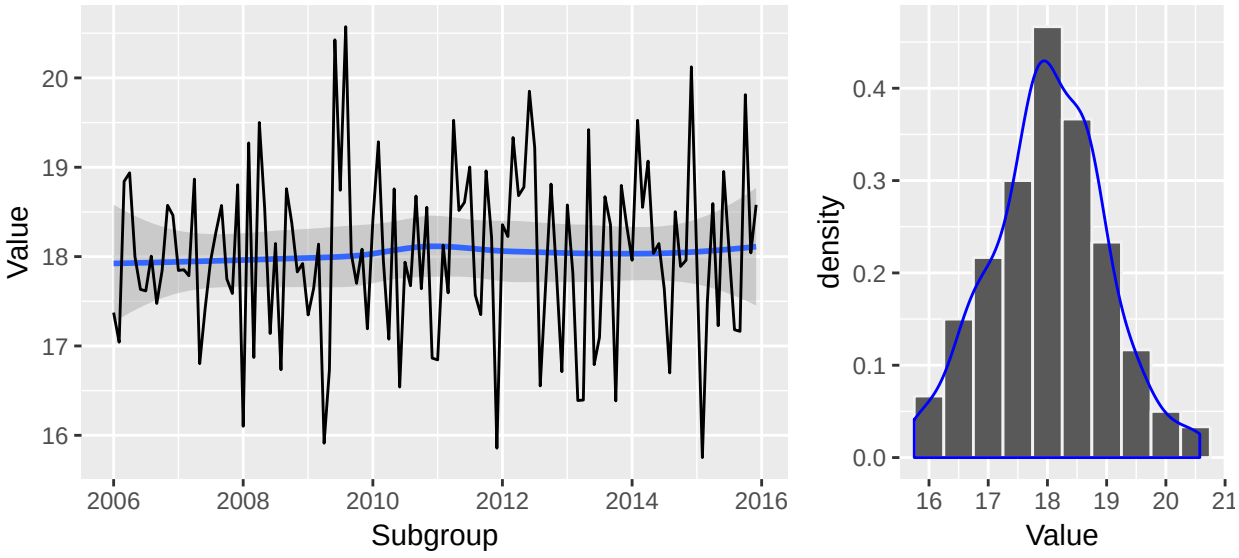
```
# Create fake process data
set.seed(250)
df = data.frame(Subgroup = seq(as.Date("2006-01-01"), by = "month", length.out = 120),
                Value = 18 + rnorm(120))

# Create a time series (`ts`) object from the df data
# (We'll use this later in the chapter)
df_ts = ts(df$Value, start = c(2006,01), frequency = 12)

# Create fake process data with an upward trend
set.seed(81)
n = 36
x = seq(1:n)
mb = data.frame(Subgroup = seq(as.Date("2006-01-01"), by = "month", length.out = n),
            Value = 10000 + (seq(1:n) * 1.25) + (rnorm(n, 0, 5)))

# Create a time series object from the mb data
```

```
mb_ts = ts(mb$Value, start = c(2006,01), frequency = 12)
```

3.2 Start with basic EDA

Before anything else, plot your data as a line chart and a histogram (adding a density overlay provides a more "objective" sense of the distribution).

```
# Line plot with loess smoother for assessing trend
p1 = ggplot(df, aes(x = Subgroup, y = Value)) +
  geom_smooth() +
  geom_line()

# Histogram with density overlay
p2 = ggplot(df, aes(Value)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.5, color = "gray95") +
  geom_density(color = "blue")

grid.arrange(p1, p2, widths = c(0.65, 0.35))
```



In these plots, consider:

- The shape of the distribution: symmetrical/skewed, uniform/peaked/multimodal, whether changes in binwidth show patterning, etc.
- Whether you see any trending, cycles, or suggestions of autocorrelation.
- Whether there are any obvious outliers or inliers—basically, any points deviating form the expected pattern.

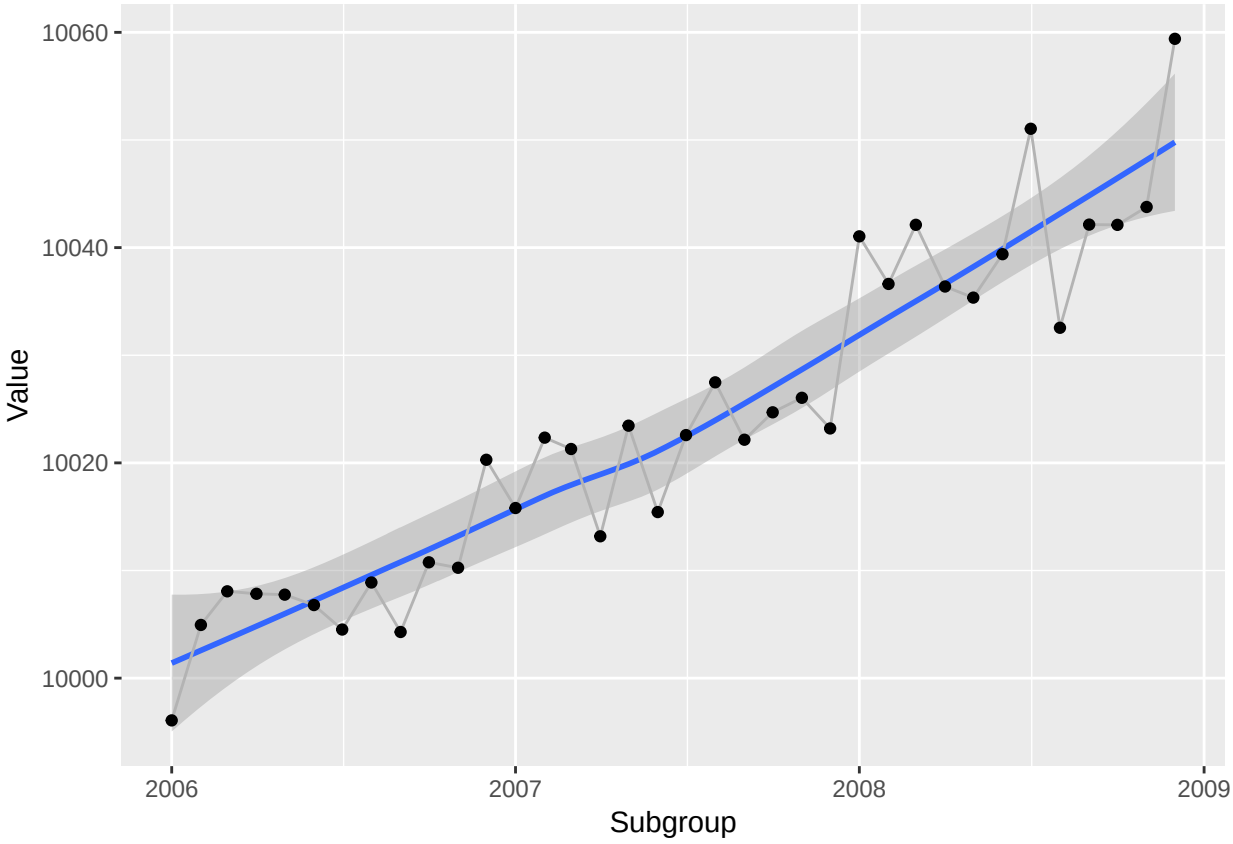
3.3 Testing assumptions

3.3.1 Trending

You can test whether a process is trending first by eye: does it look like it's trending over a large span of the time series? Then it probably is.

We don't see that in the above example, in fact, it's really close to entirely flat: a very stable process in spite of the noise. For comparison, below is an example of assessing the trend on data that is actually trending.

```
# Plot trending data
ggplot(mb, aes(x = Subgroup, y = Value)) +
  geom_smooth() +
  geom_line(color = "gray70") +
  geom_point()
```



The Mann-Kendall trend test is often used as well, a non-parametric test that can determine whether the series contains a monotonic trend, whether linear or not.

```
# Use the trend package's Mann-Kendall trend test
trend::mk.test(mb_ts)
```

```
    Mann-Kendall trend test

data:  mb_ts
z = 6.8513, n = 36, p-value = 7.318e-12
alternative hypothesis: true S is not equal to 0
sample estimates:
```

```
     S   varS    tau
 504.0 5390.0    0.8
```

Because trends can be an indication of special cause variation in a stable process, standard control limits don't make sense around long-trending data, and calculation of center lines and control limits will be incorrect. **Thus, any SPC tests for special causes other than trending will *also* be invalid over long-trending data.** Use a run chart with a median slope instead, e.g., via quantile regression (as seen in Chapter 6).

3.3.2 Independence and autocorrelation

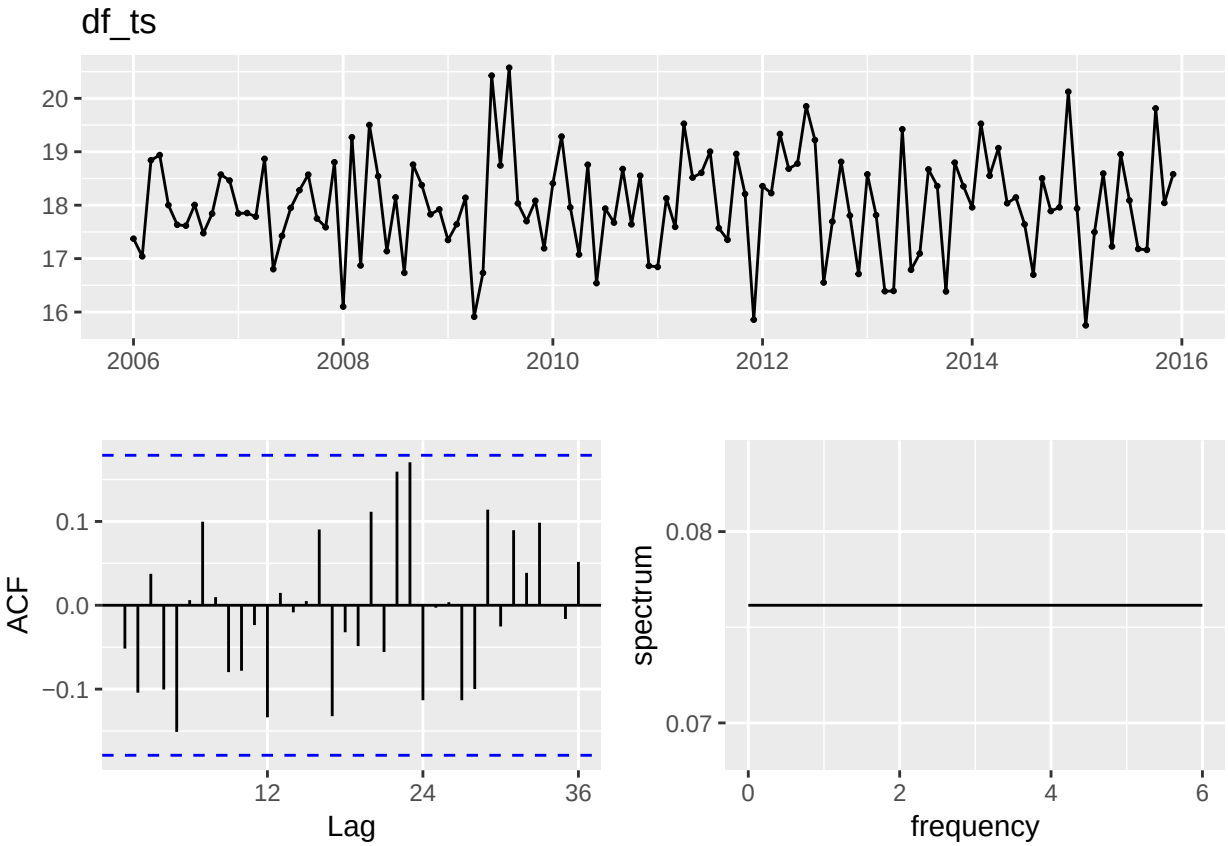
For either run charts or control charts, the data points must be independent for the guidelines to be effective. The first test of that is conceptual—do you expect that one value in this series will influence a subsequent value? For example, the incidence of some hospital-acquired infections can be the result of previous infections. Suppose one happens at the end of March and another happens at the start of April in the same unit, caused by the same organism—you might suspect that the monthly values would not be independent.

After considering the context, a second way to assess independence is by calculating the autocorrelation function (acf) for the time series. Autocorrelation values over 0.50 generally indicate problems, as do patterns in the autocorrelation function (described in Chapter 8). However, *any* significant autocorrelation should be considered carefully relative to the cost of potential false positive or false negative signals. Autocorrelation means that the run chart and control chart interpretation guidelines will be wrong.

For control charts, autocorrelated data will result in control limits that are too small. Data with seasonality (predictable up-and-down patterns) or cycles (irregular up-and-down patterns) will have control limits that are too large. There are diagnostic plots and patterns that help identify each, but the best test is "what does it look like?" If the trend seems to be going up and down, and the control limits don't, it's probably wrong.

Using the `forecast` package's `ggtsdisplay` provides a view of the time series along with the acf and spectral frequency (where frequency is the reciprocal of the time period). Significant autocorrelation is present if there are bars that transgress the blue dashed line in the ACF plot (bottom left). Cycles or seasonality are present if you see a clear peak (or peaks) in the spectrum plot (bottom right).

```
# Plot series, acf plot, and spectal plot
ggtsdisplay(df_ts, plot.type = "spectrum")
```



These plots show that there is no autocorrelation or seasonality/cyclical patterns in the data: there are no obvious patterns nor any bars that cross the blue lines in the acf plot (bottom left), and there are no peaks in the spectral density plot (bottom right). See Chapter 8 for what these plots can look like when you have time-dependent or otherwise autocorrelated data.

When you do have such data, you cannot use standard SPC tools. Generalized additive models (GAMs or GAMMs) can be useful alternatives; see Chapter 13 for some good initial references.

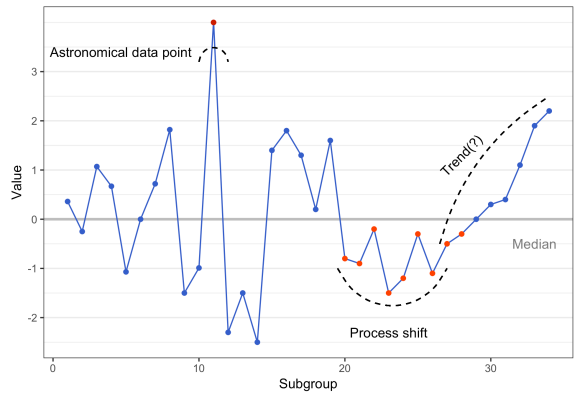
Understanding your data is a fundamental prerequisite of SPC work. Do *not* move on to SPC work until you have explored your data using the techniques demonstrated above and fully understand whether the data are suitable for SPC tools.

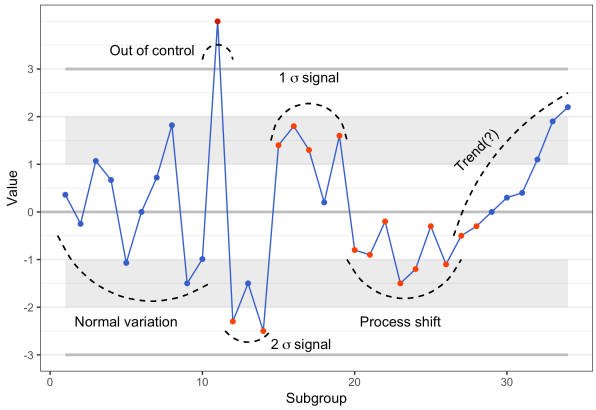
Chapter 4

Guidelines for interpreting SPC charts

(Should Chapter 4 and Chapter 5 be merged, maybe "Chart Selection and Interpretation Basics"? Ideally with a short introduction, then each guide takes up one page -BB)

Run chart	Control Chart





Identifying possible signals of change in run charts

"Astronomical" data point: a point so different from the rest that anyone would agree that the value is unusual.

Process shift: $log_2(n) + 3$ data points are all above or all below the median line, where n is the total number of points that do *not* fall directly on the median line.

Number of crossings: Too many or too few median line crossings suggest a pattern inconsistent with natural variation see Chapter 6.1.

Detecting special cause variation in control charts

One or more points fall outside the control limit: if the data are distributed according to the given control chart's assumptions, the probability of seeing a point outside the control limits when the process has not changed is very low.

Process shift: $log_2(n) + 3$ data points are all above or all below the mean line, where n is the total number of points that do *not* fall directly on the mean(?) line.

Number of crossings: Too many or too few center line crossings suggest a pattern inconsistent with natural variation (see Chapter 6.1).

Run chart	Control Chart
Trend: Seven or more consecutive points all increasing or all decreasing (though this can be an ineffective indicator*).	*Trend:* Seven or more consecutive points all increasing or all decreasing (though this can be an ineffective indicator*).
Cycles: There are obvious cycles that are not linked to known causes such as seasonality.	*Cycles:* There are obvious cycles of any sort.
	Reduced variation: Fifteen or more consecutive points all within 1σ.
	1σ signal: Four of five consecutive points are more than one standard deviation away from the mean.
	2σ signal: Two of three consecutive points are more than two standard deviations away from the mean.

**Note: Although many people use a "trend" test in association with run and control charts, research has shown this test to be ineffective (see Useful References).*

Chapter 5

Which should I use: a run chart or a control chart?

Always create a run chart first. Create a control chart only if you meet the necessary conditions, particularly that of monitoring a *stable* process.

In both cases, the data points must be independent, that is, the position of one point does not influence the position of another point: there is no (serious) autocorrelation. If the data are autocorrelated, the guidelines for testing run or control charts can be invalid, which can lead to poor decision-making.

Use a run chart if	Use a control chart (only) if
You may or may not investigate or act when a data point crosses a reference, target, or goal level, or when guidelines suggest a non-random pattern is occurring.	You intend to investigate or act when the process moves outside of control or indicates special cause variation.
You have little control over or cannot control the metric (e.g., ED volume/acuity).	You have the potential to control the process driving the metric (e.g., ED wait times).
You want to monitor the behavior of individual or groups of data points to a reference, target, or goal level.	You want to monitor the "average" of the system's behavior (i.e., the underlying statistical process) and deviations from expectation.
You are monitoring a metric or process that is generally trending or contains seasonality or other cycles of known cause, as long as you are able to adjust for any seasonality as well as able calculate an appropriate median line (e.g., via quantile regression for trending data).	You are monitoring a *stable* statistical process (there is no trend in the time series, or you have made the appropriate corrections to account or adjust for trends or seasonality).
You have no expectations that normal day-to-day operations will affect the central tendency.	You expect that normal day-to-day operations will keep the process stable within the bounds of common-cause variation.
You do not need to account for the inherent natural variation in the system.	You need to understand and account for the inherent natural variation ("noise") in the system.

Use a run chart if	Use a control chart (only) if
You have at least 12 data points. (Fewer than 12? Just make a line chart, or use an EWMA chart. Run chart guidelines may not be valid.)	You have 20 or more data points that are in a stable statistical process, or you have performed a power analysis that provides the appropriate n for the appropriate time interval(s).
You do not understand one or more of the statistical issues discussed in the control chart column.	You understand the practical trade-offs between the sensitivity and specificity of the control limits relative to your need to investigate or act.
	You know which statistical distribution to use to calculate the control limits to ensure you have the proper mean-variance relationship.

Chapter 6

Run charts

Run charts are designed to show a metric of interest over time. They do not rely on parametric statistical theory, so they cannot distinguish between common cause and special cause variation. Control charts can be more powerful when properly constructed, but run charts are easier to implement where statistical knowledge is limited and still provide practical monitoring and useful insights into the process.

Run charts typically employ the median for the reference line. Run charts help you determine whether there are unusual runs in the data, which suggest non-random variation. They tend to be better than control charts in trying to detect moderate (~1.5σ) changes in process than using the control charts' typical 3σ limits rule alone. In other words, a run chart can be more useful than a control chart when trying to detect improvement while that improvement work is still going on.

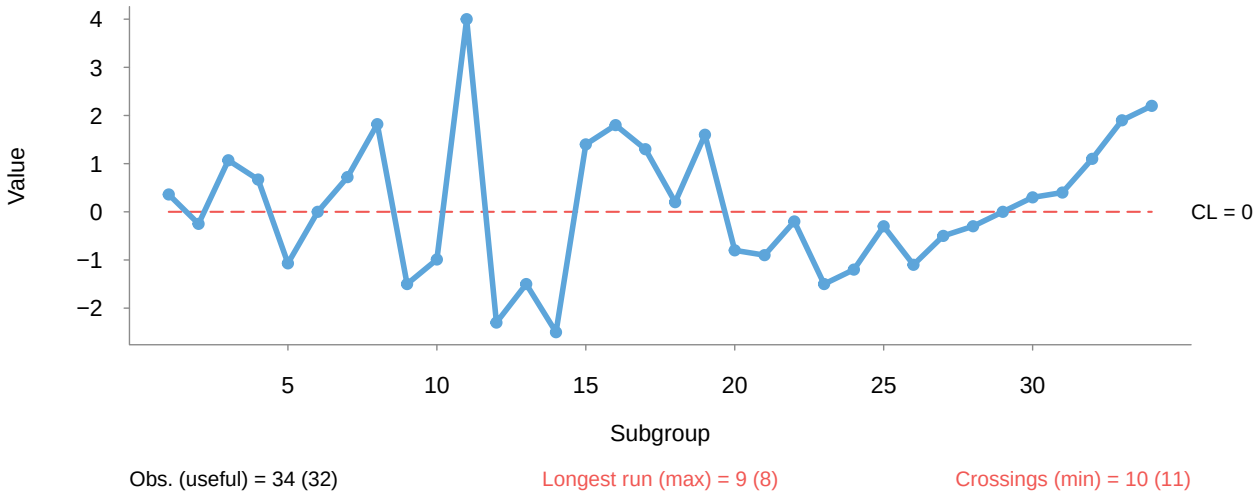
6.1 Interpreting run charts

There are two basic "tests" for run charts (an astronomical data point or looking for cycles aren't tests *per se*):

- *Process shift:* A non-random run is a set of $log_2(n) + 3$ consecutive data points (rounded to the nearest integer) that are all above or all below the median line, where n is the number of points that do *not* fall directly on the median line. For example, if there are 34 points and 2 fall on the median, then $n = 32$ observations. Thus in this case, the longest run should be no more than 8 points.

- *Number of crossings:* Too many or too few median line crossings suggest a pattern inconsistent with natural variation. You can use the binomial distribution (`qbinom(0.05, n-1, 0.50)` in R for a 5% false positive rate and expected proportion 50% of values on each side of the median) or a table (e.g., Table 1 in Anhøj & Olesen 2014) to find the minimum number of expected crossings. Using the same data as in the example introduced in Chapter 1, we would expect the time series to cross the median entirely at least 11 times (points on the median that return to the same side are not crossings).

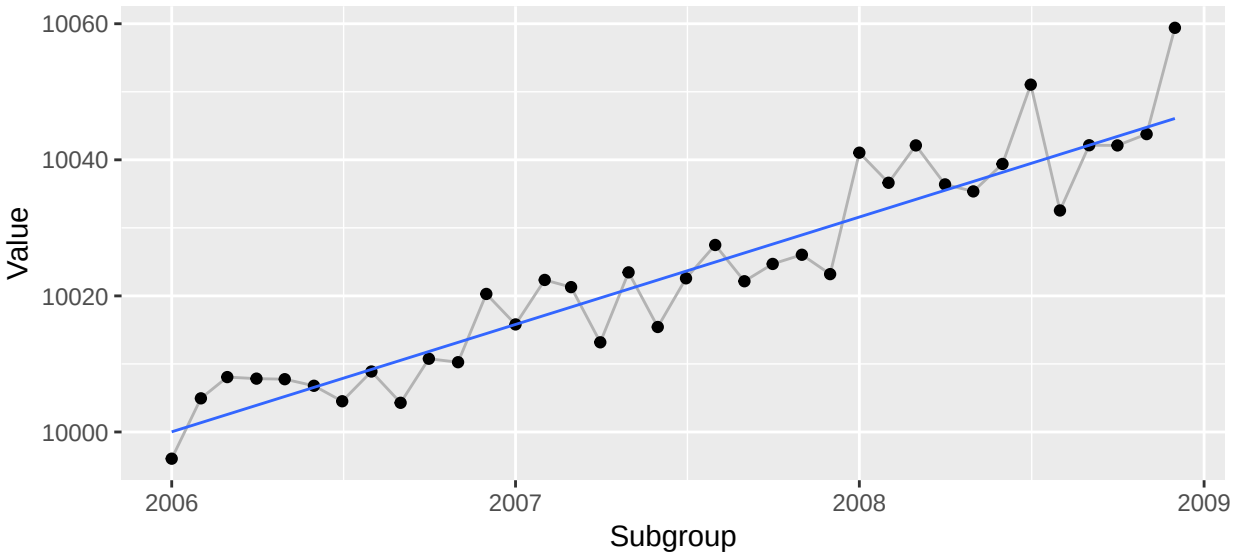
```
# The qic function in the qicharts package creates easy run and control charts
# The runvals option shows the test results on the plot
qicharts::qic(point.signal, runvals=TRUE, ylab="Value", main="")
```



6.2 Run charts with a trend

When you first notice a trend like that seen above, you can generally wait until the process has settled to a new, stable mean and reset the central line accordingly. For a sustained or continuous trend, you can difference the data (create a new dataset by subtracting the value at time t from the value at time $t+1$) to remove the trend or use regression residuals to show deviations from the trend. However, either approach can make the run chart harder to interpret. Perhaps a better idea is use quantile regression to obtain the median line, which allows you to keep the data on the original scale.

```
# Generate fake process data with an upward trend

# Do we need to repeat this since we made it before?
# Might serve as a good reminder, or maybe just delete the first
# instance of it to save space.

set.seed(81)
n = 36
x = seq(1:n)
mb = data.frame(Subgroup = seq(as.Date("2006-01-01"), by = "month", length.out = n),
                Value = 10000 + (seq(1:n) * 1.25) + (rnorm(n, 0, 5)))


# Plot with quantile regression for median line
ggplot(mb, aes(Subgroup, Value)) +
  xlab("Subgroup") +
  ylab("Value") +
  geom_line(color="gray70") +
  geom_point() +
  geom_quantile(quantiles = 0.5)
```



Process shift "test": $log_2(n)+3$ where n is 33 points that do not touch the median is 8.04, so there should be no more than 8 points in any given run. The longest run in this chart is 5 points.

Crossings "test": `qbinom(0.05, 33, 0.50)` is 12, the minimum number of crossings we'd expect. There are 13 crossings in this run chart, as points that fall on the median line and return to the same side are not considered crossings.

Both tests suggest there is no non-random variation in this process.

Chapter 7

Control charts

7.1 Statistical distributions

The primary distinction between run and control charts is that the latter uses parametric statistics monitor additional properties of a data-defined process. If a particular statistical distribution—such as normal, binomial, or Poisson—matches the process you wish to measure, a control chart ofers a great deal more power to find insights and monitor change than a line or run chart.

Parametric distributions are a *useful fiction*—no data will follow an idealized distribution, but as long as it's close, the distribution properties provide useful shortcuts that allow SPC charts to work *in practice*.

7.1.1 Common distributions and their ranges

There are hundreds of statistical distributions DO WE NEED TO WORRY ABOUT COPYRIGHT? MAYBE LINK TO https://en.wikipedia.org/wiki/List_of_probability_distributions?, but only a handful are commonly used in SPC work:

Data Type	Distribution	Range	Skew	Example	SPC chart
Discrete	Binomial	0, N	Any	Bundle compliance percentage	p, np
	Poisson	0, ∞	Right	Infections per 1,000 line days	u, c
	Geometric	0, ∞	Right	Number of surgeries between complications	g
Continuous	Normal	$-\infty$, ∞	None	Patient wait times	I, $\bar{x}$, EWMA, CUSUM
	Weibull	0, ∞	Right	Time between antibiotic doses	t

7.1.2 Mean and variance

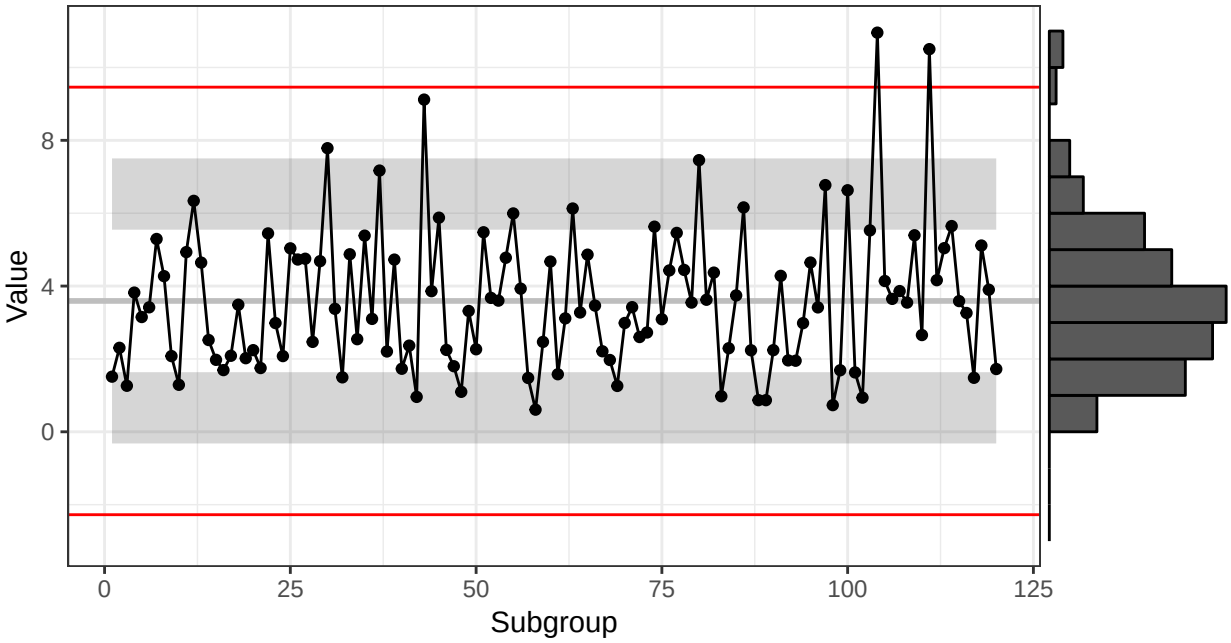
When control charts use the mean to create the center line, they use the arithmetic mean. Rather than using the $\bar{x}$ abbreviation, these mean values are usually named for the type of chart (u, p, etc.) to emphasize the use of control limits that are not based on the normal distribution. The variance used to calculate the control limits differs by distribution.

7.1.3 What happens when you get the mean-variance relationship wrong

Although control charts are "robust" to some assumption violations and can sometimes work when the mean-variance relationship is incorrect, you won't know unless you explore the differences in implications between the data as-is and that same data transformed to become more in line with the appropriate or expected distribution.

For example, if you use the usual normal distribution control limits (an I chart) on gamma-distributed data, you get something like this:

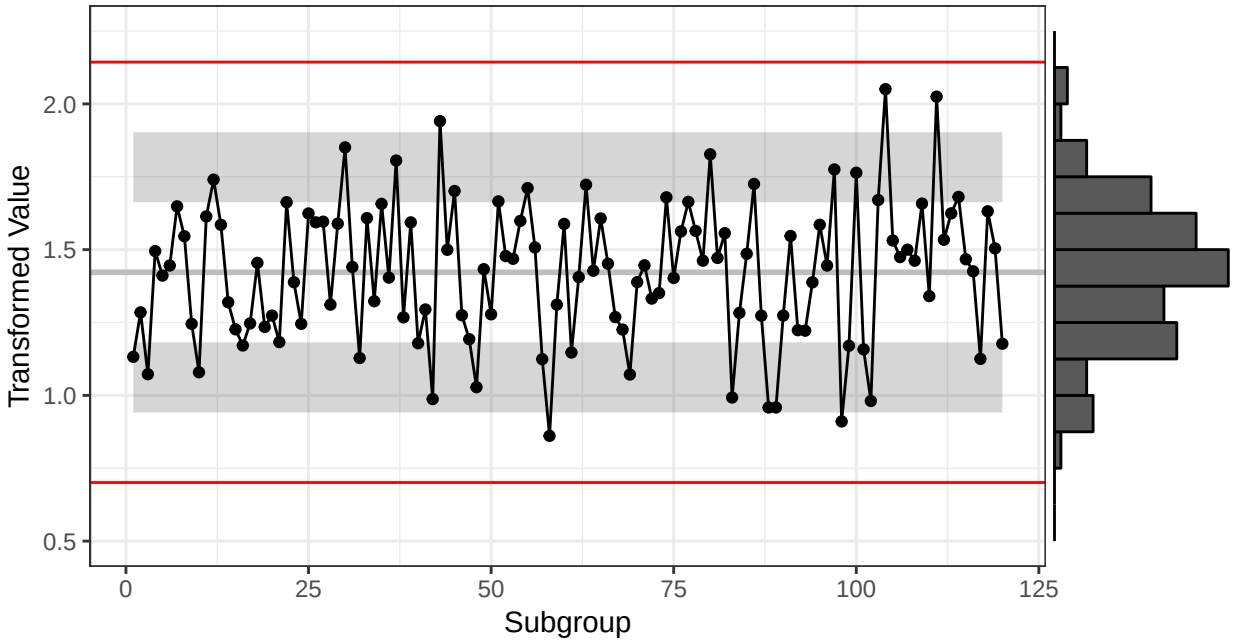
```
# Create some fake gamma-distributed process data
set.seed(3)
df2 = data.frame(x = seq(1:120), y = rgamma(120, shape = 3, rate = 0.8))

# Create plot object
exp_nat_var_cc_plot = ggplot(df2, aes(x, y)) +
  ylim(-3, 11) +
  geom_hline(aes(yintercept=mean(y)), color="gray", size=1) +
  geom_hline(aes(yintercept=mean(y)+(3*sd(y))), color="red") +
  geom_hline(aes(yintercept=mean(y)-(3*sd(y))), color="red") +
  geom_ribbon(aes(ymin = mean(y)-(2*sd(y)), ymax = mean(y)-(1*sd(y))), alpha = 0.2) +
  geom_ribbon(aes(ymin = mean(y)+(1*sd(y)), ymax = mean(y)+(2*sd(y))), alpha = 0.2) +
  xlab("Subgroup") +
  ylab("Value") +
  geom_line() + geom_point() +
  theme_bw()

# Marginal plot
ggMarginal(exp_nat_var_cc_plot, margins="y", type = "histogram", binwidth=1)
```



Clearly something is weird when very few points go below one standard deviation, and none go below two. And do the points above the upper control limit represent *real* anomalous data points, or are they the result of an improper mean-variance relationship?

Using a Box-Cox transformation to make the distribution more symmetrical, we can see that those seemingly out-of-control points are actually well within both control limits, and the variation we see is more in line with (statistical) expectation.

```
# Box-Cox tansformation
bob = data.frame(MASS::boxcox(df2$y ~ 1, lambda=seq(-10, 10, 0.05), plotit=F))
bobmax = bob[which.max(bob[,2]),1]

# Adjustment to make plotting cleaner
df2$y2 = (df2$y ^ bobmax)

# Create plot object
exp_xform_nat_var_cc_plot = ggplot(df2, aes(x, y2)) +
  ylim(0.5, 2.25) +
  geom_hline(aes(yintercept=mean(y2)), color="gray", size=1) +
  geom_hline(aes(yintercept=mean(y2)+(3*sd(y2))), color="red") +
  geom_hline(aes(yintercept=mean(y2)-(3*sd(y2))), color="red") +
  geom_ribbon(aes(ymin = mean(y2)-(2*sd(y2)), ymax = mean(y2)-(1*sd(y2))), alpha = 0.2) +
  geom_ribbon(aes(ymin = mean(y2)+(1*sd(y2)), ymax = mean(y2)+(2*sd(y2))), alpha = 0.2) +
  xlab("Subgroup") +
  ylab("Transformed Value") +
  geom_line() + geom_point() +
  theme_bw()

# Marginal plot
ggMarginal(exp_xform_nat_var_cc_plot, margins="y", type = "histogram", binwidth=0.125)
```



The main drawback is that you now have a chart of essentially uninterptable values—but that's better than assuming a normal distribution will be just fine and inviting false positive signals, potentially wasting time and resources searching for a special cause that doesn't exist.

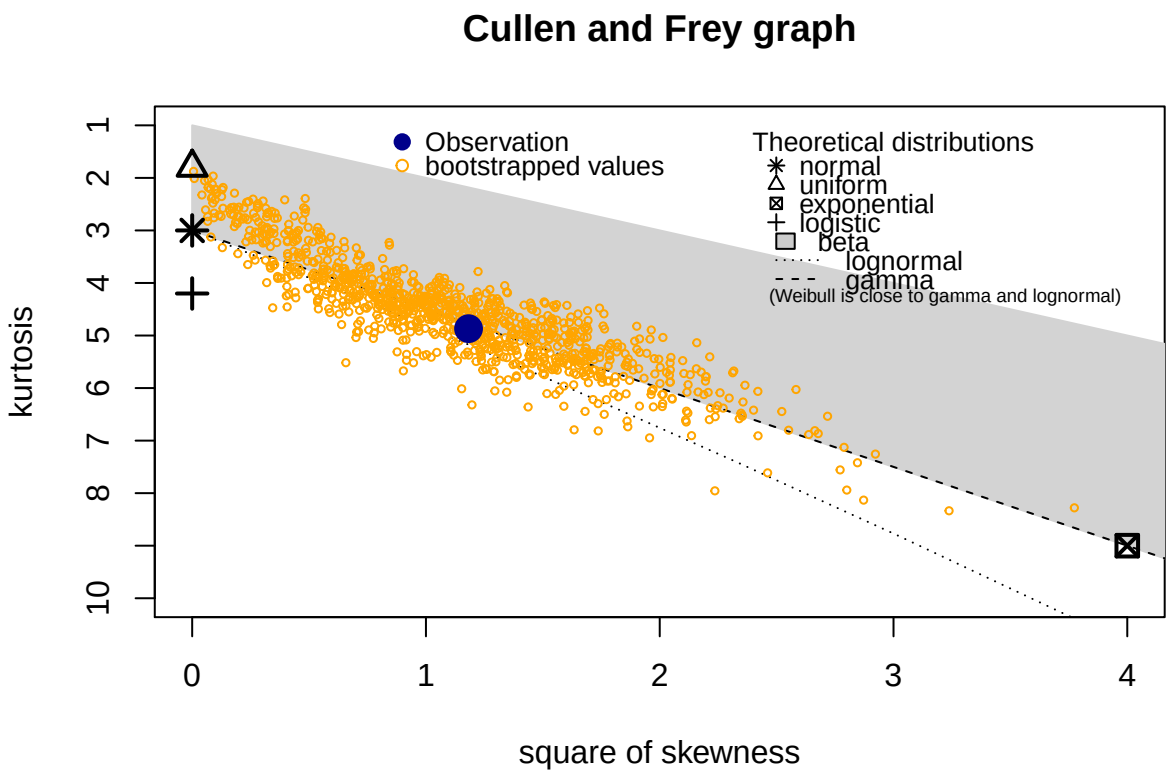
So should you always transform when your data doesn't meet the usual distributions common in control charts? Not necessarily. For more information, see, for example, *The arcsine is asinine* [**?**] and *Do not log-transform count data* [**?**]. DO YOU HAVE A LINK FOR THESE OR A CITATION? Consult a statistician if you aren't sure how to proceed.

7.1.4 What *is* the distribution?

There are R packages and functions to evaluate your data and show what distribution(s) are most consistent with it. This does *not* tell you that your data does follow a given distribution, only that it's consistent with it. Further analysis is usually required; consult a statistician when you're uncertain.

As an example, we can use the gamma-distrubted data created above to show how it works.

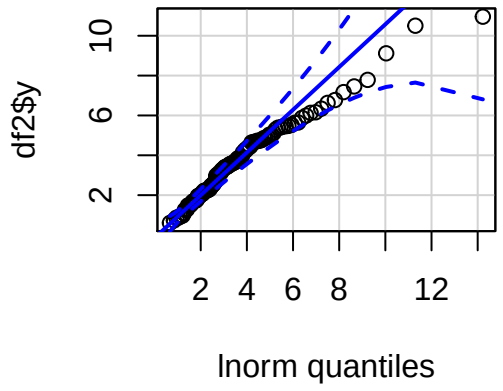
```
library(fitdistrplus)
expo_dist = descdist(df2$y, boot = 1000)
```



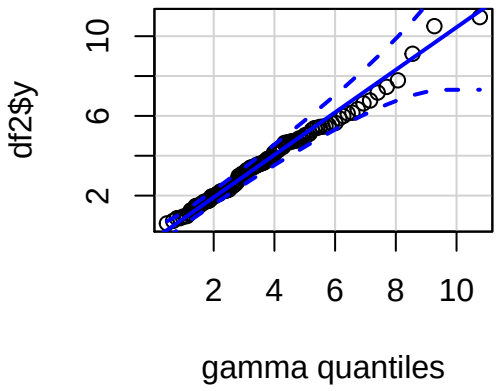
A Cullen and Frey graph compares the data set (blue dot) and bootstrapped replications (orange open circles) to common theoretical distributions. For example, if the blue dot were at or near the * symbol at the top left and more or less surrounded by the orange open circles, it would imply the data are most consistent with a normal distribution. Other common distributions are represented in the graph by points (e.g., the exponential distribution), area (e.g., the beta distribution), or by lines (e.g., the gamma distribution).

This chart shows us that our data (blue dot) and simuations from that data (orange open circles) are most consistent with a gamma distribution and a perhaps a lognormal distribution. Using `qqPlot` lets us evaluate these two options directly:

```
# Create objects of the two most-likely distributions
logno = fitdistr(df2$y, "lognormal")
gammo = fitdistr(df2$y, "gamma")

# The car package has a good quantile-quantile plot function
library(car)
qqPlot(df2$y, "lnorm", meanlog = logno$estimate[1], sdlog = logno$estimate[2], id=FALSE);
```



```
qqPlot(df2$y, "gamma", shape = gammo$estimate[1], rate = gammo$estimate[2],id=FALSE)
```

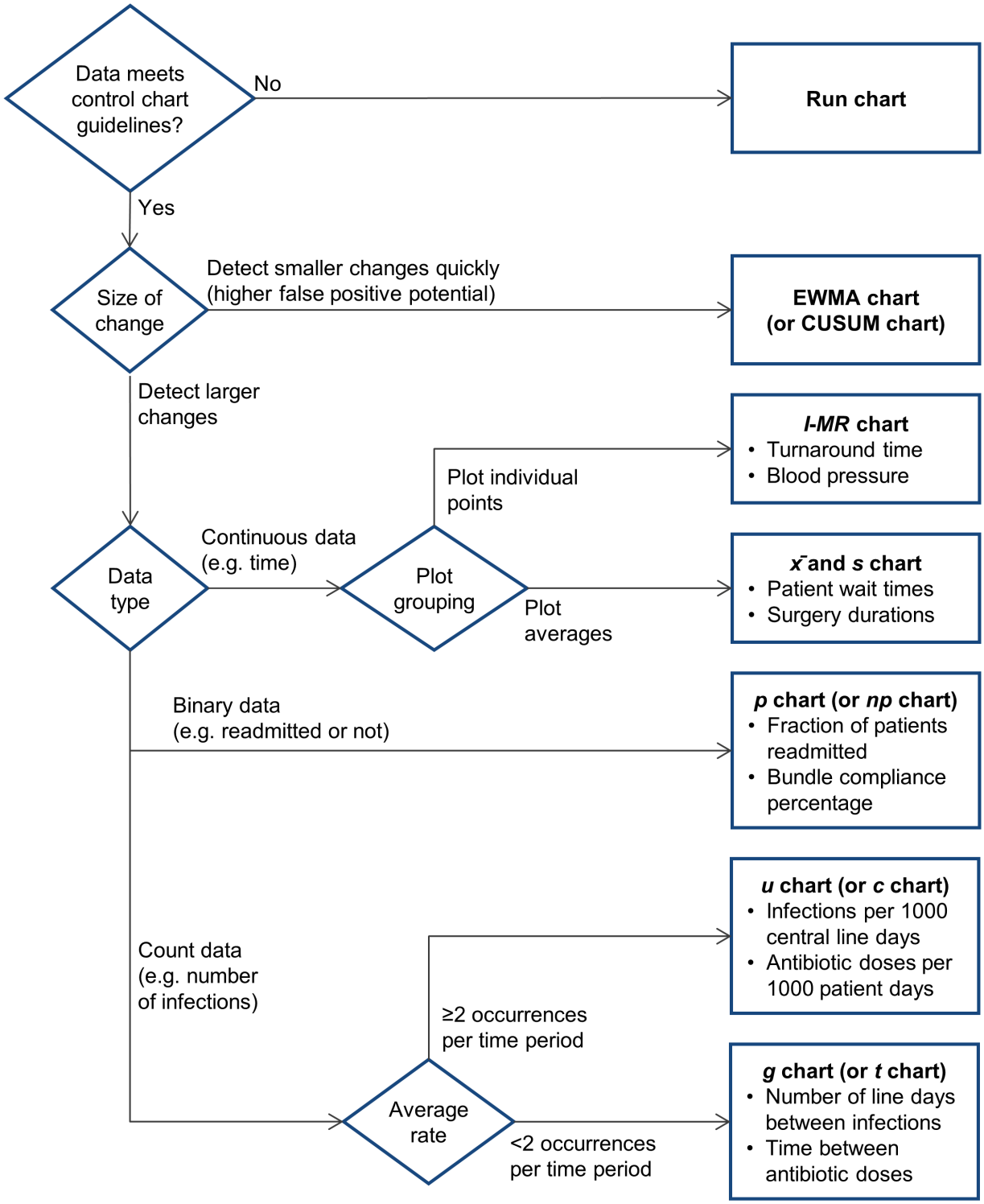


Although both distributions fall within the confidence limits (dashed lines), the points fit to a gamma distribution (right) DOUBLE-CHECK ORIENTATION are closer to the line of best fit.

This is expected for this example with data we created from a gamma distribution. But in practice, when you don't know what distribution the data comes from, using this process can help you determine which distributions are most consistent with the data and plot it appropriately.

7.2 Which control chart should I use?

The following flow chart can help you determine which kind of control chart you might want to use. More details and formulas for each control chart type are provided in the next few chapters.


Data meets control chart guidelines?
No
Run chart
Yes
Size of change
Detect smaller changes quickly (higher false positive potential)
EWMA chart (or CUSUM chart)
Detect larger changes
I-MR chart
• Turnaround time
• Blood pressure
Plot individual points
Data type
Continuous data (e.g. time)
Plot grouping
Plot averages
x̄ and s chart
• Patient wait times
• Surgery durations
Binary data (e.g. readmitted or not)
p chart (or np chart)
• Fraction of patients readmitted
• Bundle compliance percentage
u chart (or c chart)
• Infections per 1000 central line days
• Antibiotic doses per 1000 patient days
Count data (e.g. number of infections)
≥2 occurrences per time period
Average rate
<2 occurrences per time period
g chart (or t chart)
• Number of line days between infections
• Time between antibiotic doses

Chapter 8

Tips and tricks for successful control chart use

8.1 READ ME (or else)

- The definition of your control limits depends on the trade-off between sensitivity and specificity for the question at hand. Typical control charts are built on 3σ limits, which provides a balanced trade-off between sensitivity and specificity, that is, between under- and over-alerting to an indication of special cause variation. When you need to err on the side of caution—for example, in patient safety applications—2σ limits may be more appropriate, while understanding that false positives will be higher. If you need to err on the side of certainty, 4-6σ limits may be more useful.
- With fewer than 20 observations, there is an increased chance of missing special cause variation. With more than 30 observations, there's an increased chance of detecting special cause variation that is really just chance. Knowing these outcomes are possible is useful to help facilitate careful thinking when control charts indicate special cause variation.
- Ensure your data values and control limits make sense. For example, if you have proportion data and your control limits fall above 1 (or above 100%) or below 0, there's clearly an error somewhere. Ditto with negative counts.
- For raw ordinal data (such as likert scores), do not use means or control limits. Just. Don't. If you must plot a single value, convert to a proportion (e.g., "top box scores") first. However, stacked bar or mosaic charts help visualize this kind of data much better, and can be done in the same amount of space.
- Control charts don't measure "statistical significance"—they are meant to reduce the chances of incorrectly deciding whether a process is in (statistical) control or not. Control limits are *not* confidence limits.
- YTD comparisons don't work because they encourage naive, point-to-point comparisons and ignore natural variation—and can encourage inappropriate knee-jerk reactions. There is never useful information about a process in only one or two data points.
- A control chart should measure one defined process, so you may need to create multiple charts stratified by patient population, unit, medical service, time of day, etc. to avoid mixtures of processes.
- With very large sample or subgroup sizes, control limits will be too small, and the false positive rate will skyrocket. Use prime charts THIS LINK DOESN'T GO ANYWHERE instead.

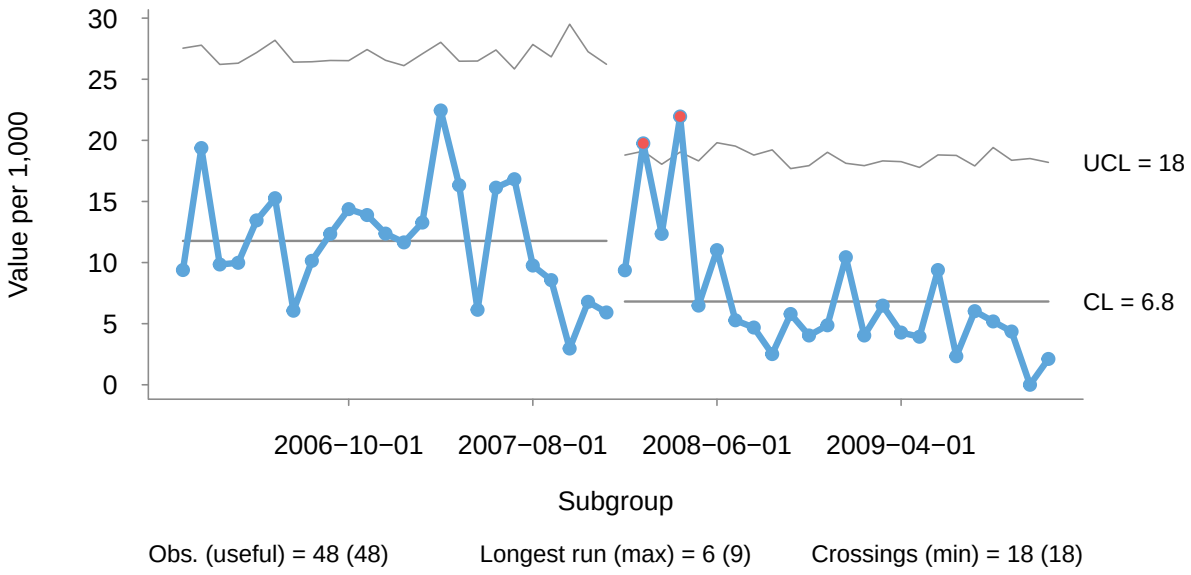
8.2 When to revise control limits

If you need to determine whether an intervention might have worked soon after or even during the improvement process, you shouldn't be using a standard control chart at all. Use a run chart or an EWMA or CUSUM chart to try to detect early shifts.

When you have enough data points after the intervention (about 12-20), with no other changes to the process, you can "freeze" the median and/or mean+control limits at the intervention point and recalculate the median and/or mean+limits on the subsequent data. However, by doing so you are *already assuming* that the intervention changed the process. If there is no evidence of special cause variation after the intervention, you shouldn't recalculate the SPC chart values.

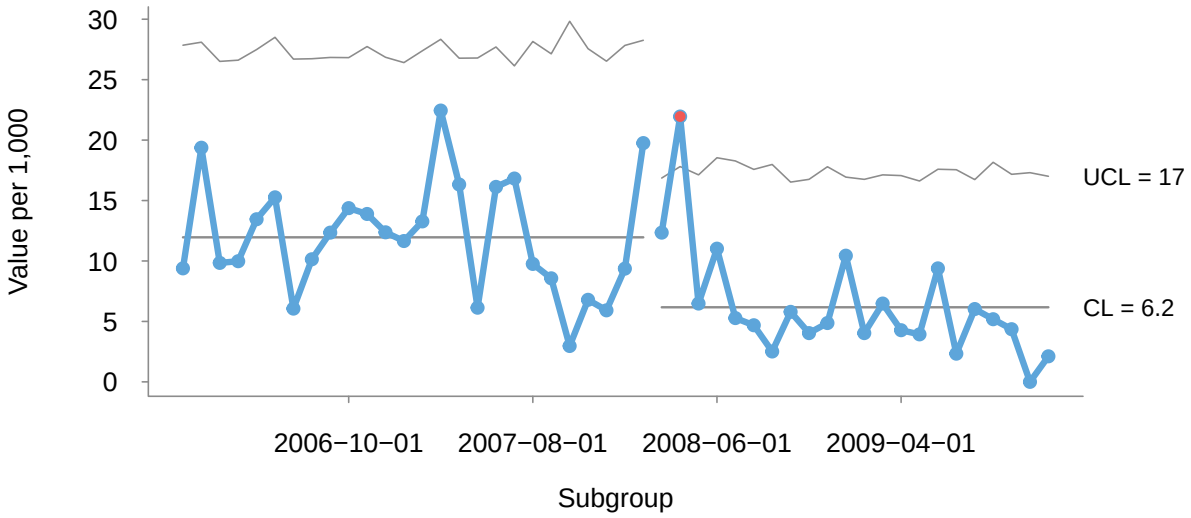
Say that an intervention happened at the start of year 3, but there was a lag between the intervention and when it actually showed up in the data.

```
# Create fake data with change in process at 28 months
set.seed(3)
intervention = data.frame(Date = seq(as.Date("2006-01-01"), by = 'month',
  length.out = 48), y = c(rpois(28, 6), rpois(20, 3)),
  n = round(rnorm(48, 450, 50)))

# Plot control chart with break at intervention
qicharts::qic(y = y, n = n, data = intervention, multiply = 1000, x = Date, chart = "u",
    runvals = TRUE, ylab = "Value per 1,000", main="", breaks = 24)
```



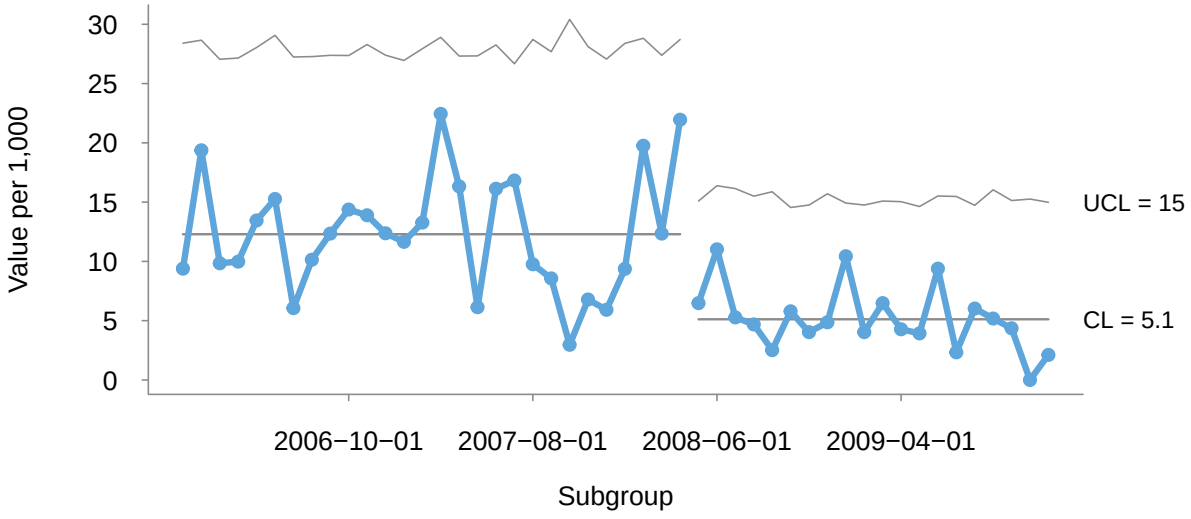
Of course, the change point can be placed arbitrarily in a `qic` graph—with corresponding changes in control limits. For example, using the same data as above, compare those results with those when the change point is moved forward by 2, 4, or 6 time steps (pretending we don't actually know when the process truly changed):

```
# Plot control chart with break 2 months after intevention
qicharts::qic(y = y, n = n, data = intervention, multiply = 1000, x = Date, chart = "u",
    runvals = TRUE, ylab = "Value per 1,000", main="", breaks = 26)
```



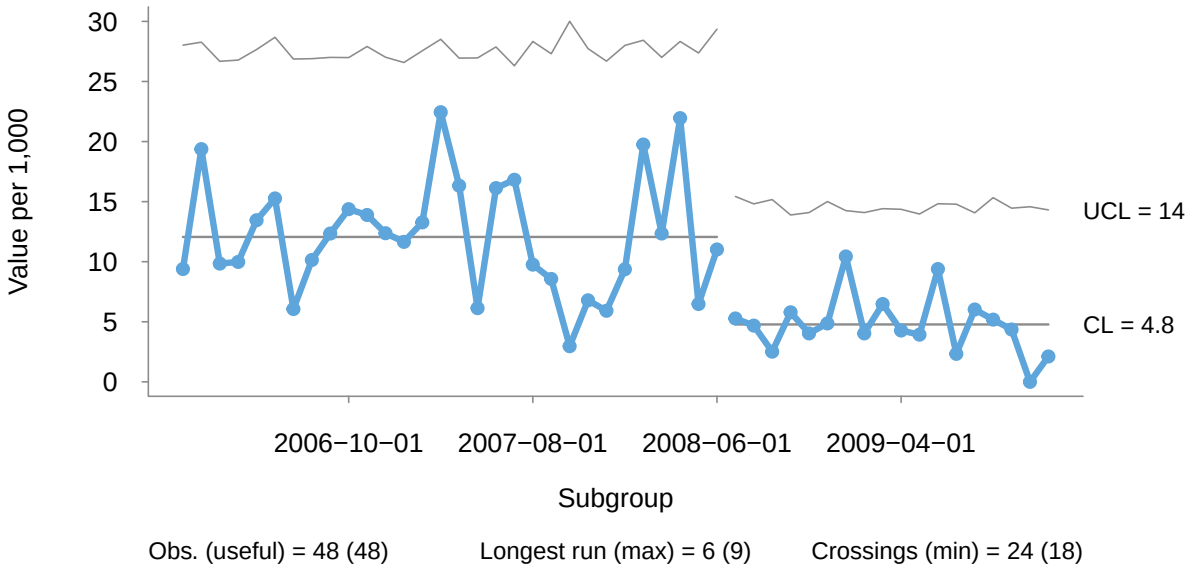
Obs. (useful) = 48 (48) Longest run (max) = 6 (9) Crossings (min) = 18 (18)

```
# Plot control chart with break 4 months after intevention
qicharts::qic(y = y, n = n, data = intervention, multiply = 1000, x = Date, chart = "u",
    runvals = TRUE, ylab = "Value per 1,000", main="", breaks = 28)
```



Obs. (useful) = 48 (48) Longest run (max) = 6 (9) Crossings (min) = 22 (18)

```
# Plot control chart with break 6 months after intevention
qicharts::qic(y = y, n = n, data = intervention, multiply = 1000, x = Date, chart = "u",
    runvals = TRUE, ylab = "Value per 1,000", main="", breaks = 30)
```



As you can see, the conclusions you could draw from a single control chart might be different depending on when the breakpoint is set.

Use common sense and avoid the urge to change medians or means and control limits for every intervention unless evidence is clear that it worked. DO WE WANT TO TALK ABOUT USING THE LOG2(n) + 3 RULE OR THE CROSSINGS RULE TO CHOOSE THE CUT-POINT?

SPC charts are blunt instruments, and are meant to try to detect changes in a process as simply as possible. When there is no clear evidence in SPC charts for a change, more advanced techniques—such as ARIMA models or intervention/changepoint analysis—can be used to assess whether there was a change in the statistical process at or near the intervention point.

Chapter 9

Control charts for count, proportion, or rate data

If your data involve…	use a …	based on the … distribution.
Rates	u chart	Poisson
Counts (with equal sampling units)	c chart	Poisson
Proportions	p chart	binomial
Proportions (with equal denominators)	np chart	binomial
Rare events	g chart	geometric

- For count, rate, or proportion data, carefully define your numerator and denominator. Evaluate each separately over time to see whether there are any unusual features or patterns. Sometimes patterns can occur in one or the other, then disappear or are obscured when coalesced into a rate or proportion.
- For count data, prefer u-charts to c-charts. In most cases, we do not have a constant denominator, so c-charts would not be appropriate. Even when we do, using a u-chart helps reduce audience confusion because you are explicitly stating the "per x".
- For proportion data, prefer p-charts to np-charts. Again, we almost never have a constant denominator, so np-charts would not be appropriate. Even when we do, using a p-chart helps reduce audience confusion by explicitly stating the "per x".
- Rare events can be evaluated either by g-charts (this chapter) for discrete events/time steps, or t-charts (next chapter) for continuous time.

9.1 u-chart example

The majority of healthcare metrics of concern are rates, so the most common control chart is the u-chart.

Sometimes, a KPI is based on counts. This is obviously problematic for process monitoring in most healthcare situations because it ignores the risk exposure—for example, counting the number of infections over time is meaningless if you don't account for the change in the number of patients in that same time period. When KPIs are measuring counts with a denominator that is *truly fixed*, technically a c-chart can be used. This makes sense in manufacturing, but not so much in healthcare, where the definition of the denominator can be very important. You should always use a context-relevant denominator, so in basically all cases a u-chart should be preferred to a c-chart.

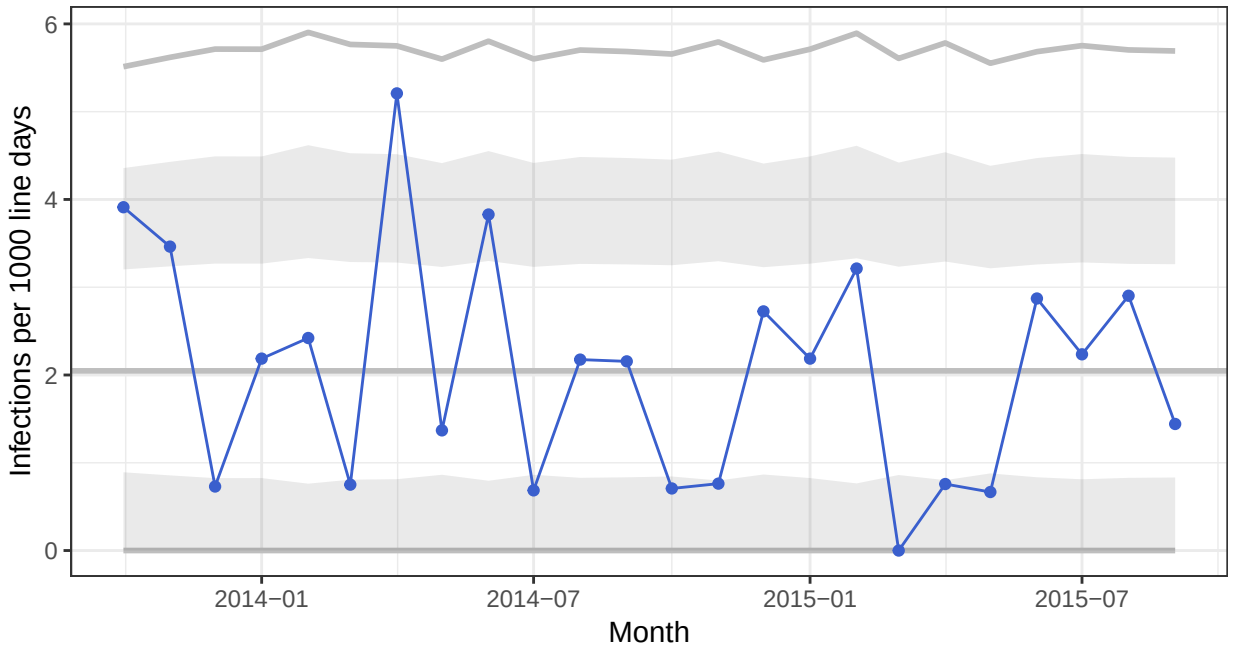
Mean for rates (u): $u = \frac{\Sigma c_i}{\Sigma n_i}$

3σ control limits for rates (u): $3\sqrt{\frac{u}{n_i}}$

Infections per 1000 central line days

```
# Generate fake infections data
set.seed(72)
dates = seq(as.Date("2013/10/1"), by = "day", length.out = 730)
linedays = sample(30:60,length(dates), replace = TRUE)
infections = rpois(length(dates), 2/1000*linedays)

months = as.Date(strftime(dates, "%Y-%m-01"))

dfi = data.frame(months, linedays, infections)

infections.agg = aggregate(dfi$infections, by = list(months), FUN = sum, na.rm = TRUE)$x
linedays.agg = aggregate(dfi$linedays, by = list(months), FUN = sum, na.rm = TRUE)$x

# Calculate u chart inputs
subgroup.u = unique(months)
point.u = infections.agg / linedays.agg * 1000
central.u = sum(infections.agg) / sum(linedays.agg) * 1000
sigma.u = sqrt(central.u / linedays.agg * 1000)
```

```
# Plot u chart
spc.plot(subgroup.u, point.u, central.u, sigma.u, k = 3, lcl.min  = 0,
         label.x = "Month", label.y = "Infections per 1000 line days")
```



See Appendix for details of the spc.plots() function.

9.2 *p*-chart example

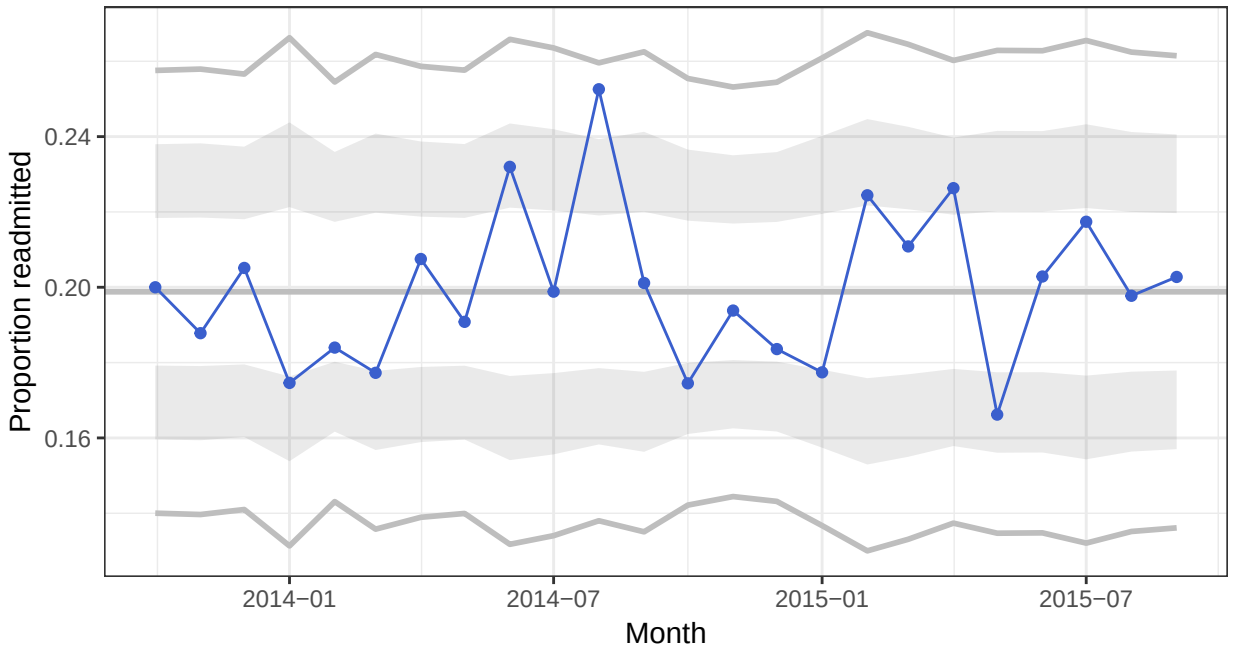
When your metric is a true proportion (and not a rate, e.g., a count per 100), a *p*-chart is the appropriate control chart to use.

Mean for proportions (p): $p = \frac{\Sigma y_i}{\Sigma n_i}$

3σ control limits for proportions (p): $3\sqrt{\frac{p(1-p)}{n_i}}$

Proportion of patients readmitted

```
# Generate sample data
discharges = sample(300:500, 24)
readmits = rbinom(24, discharges, .2)
dates = seq(as.Date("2013/10/1"), by = "month", length.out = 24)

# Calculate p chart inputs
subgroup.p = dates
point.p = readmits / discharges
central.p = sum(readmits) / sum(discharges)
sigma.p = sqrt(central.p*(1 - central.p) / discharges)

# Plot p chart
spc.plot(subgroup.p, point.p, central.p, sigma.p,
         label.x = "Month", label.y = "Proportion readmitted")
```



9.3 Rare events (*g*-charts)

There are important KPIs in healthcare related to rare events, such as is common in patient safety and infection control. These commonly have 0 values for several subgroups within the process time-period. In these cases, you need to use *g*-charts for a discrete time scale (e.g., days between events) or *t*-charts for a continuous time scale (e.g., time between events). See Chapter 10) for details on using *t*-charts to evaluate the time between events.

9.3.1 *g*-chart example

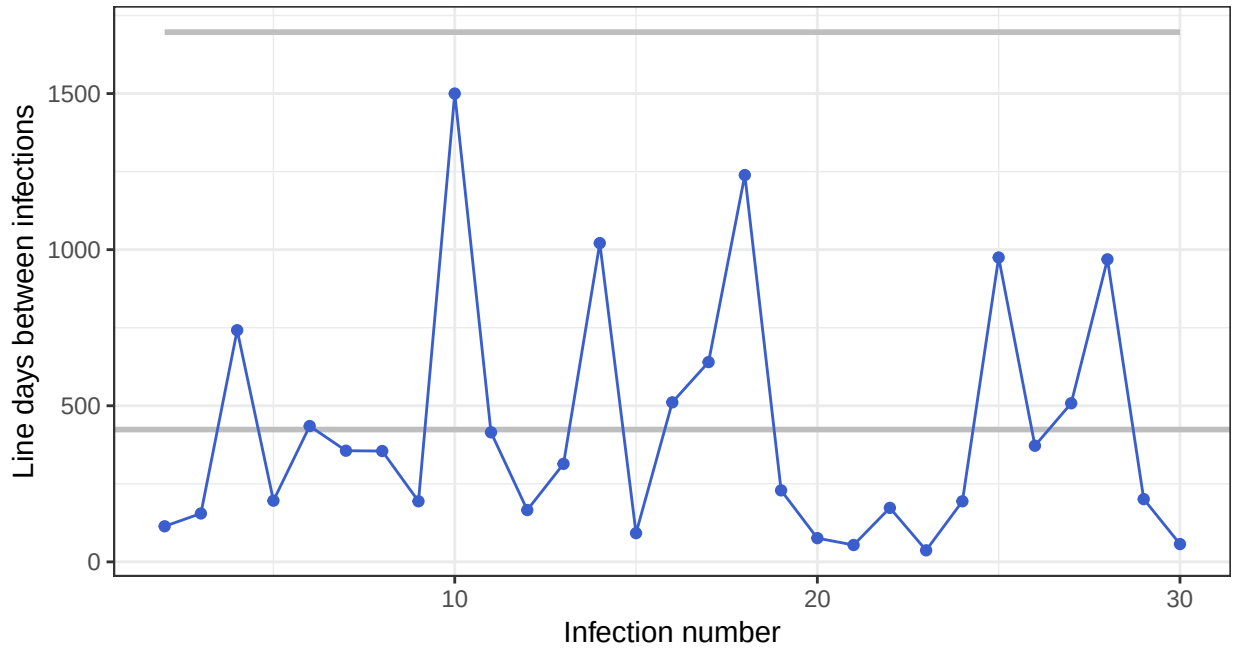
Mean for infrequent counts (g): $g = \frac{\Sigma g_i}{\Sigma n_i}$ *where*
g = units/opportunities between events

3σ limits for infrequent counts (g): $3\sqrt{g(g+1)}$

Days between infections

```
# Generate fake data using u-chart example data
infections.index = which(infections > 0)[1:30]
dfind = data.frame(start = head(infections.index, length(infections.index) - 1) + 1,
                   end = tail(infections.index, length(infections.index) - 1))

linedays.btwn = matrix( , length(dfind$start))

for (i in 1:length(linedays.btwn)) {
    sumover = seq(dfind$start[i], dfind$end[i])
    linedays.btwn[i] = sum(linedays[sumover])
}

# Calculate g chart inputs
subgroup.g = seq(2, length(infections.index))
point.g = linedays.btwn
central.g = mean(point.g)
sigma.g = rep(sqrt(central.g*(central.g+1)), length(point.g))
```

```
# Plot g chart
spc.plot(subgroup.g, point.g, central.g, sigma.g, lcl.show = FALSE,
         band.show = FALSE, rule.show = FALSE,
         lcl.min = 0, k = 3, label.x = "Infection number",
         label.y = "Line days between infections")
```



9.4 *c*- and *np*-chart details

Simply for completeness, means and control limits for *c*- and *np*-charts are presented here. To emphasize that *u*- and *p*-charts should be preferred (respectively), no examples are given.

Mean for counts (*c*): $\frac{\Sigma c_i}{n}$

3σ control limits for counts (*c*)(not shown): $3\sqrt{c}$

Mean for equal-opporuntity proportions (*np*): $np = \frac{\Sigma y_i}{n}$
where
n is a constant

3σ control limits for equal-opporuntity proportions (*np*): $3\sqrt{np(1-p)}$
where
n is a constant

Chapter 10

Control charts for numeric, normally-distributed data

If your data involve…	use a/an …	based on the … distribution.
Individual points	I chart	normal
Subgroup average	$\bar{x}$ and s chart	normal
Exponentially weighted moving average	EWMA chart	normal
Cumulative sum	CUSUM chart	normal
Time between (rare) events	t chart	Weibull

- For continuous data, the definition of the control limits will depend on your question and the data at hand. To detect small shifts in the mean quickly, an EWMA is probably best, while to understand natural variation and try to detect special cause variation, an $\bar{x}$ and s chart will be more useful.
- In the rare cases you may need an individual chart, do *not* use 3σ for the control limits; you must use $2.66MR_{bar}$ instead to ensure the limits are presented correctly.
- Note: EWMA and CUSUM charts aren't "standard" control charts in that the only guideline for detecting special cause variation is a point outside the limits. So while they can't detect special cause variation like control charts, they *can* detect shifts in mean with fewer points than a standard control chart. "MAY" DETECT SHIFTS? LOOK AT THE X-BAR S CHARTS FOR WAIT TIMES VERSUS THE EWMA CHART FOR WAIT TIMES.

10.1 *I-MR* chart

(Think we should move the IMR chart section just above the t-chart section -BB)

When you have a single measurement per subgroup, the *I-MR* combination chart is appropriate. They should always be used together.

Mean($\bar{x}$): $\bar{x} = \frac{\sum_{x_i}}{n}$

Control limits for normal data (*I*): $2.66MR_{bar}$

where

MR_{bar} = average moving range of xs, excluding those $> 3.27MR_{bar}$

I DON'T SEE WHERE YOU'RE EXCLUDING VALUES OVER 3.27*MR OR IS THAT WHAT THE K DOES IN THE FIRST PLOT?

(Does excluding >3.27MR come from the t chart source (Provost and Murray)? I don't see it excluded in most IMR formulations, but added the exclusion the code below -BB)

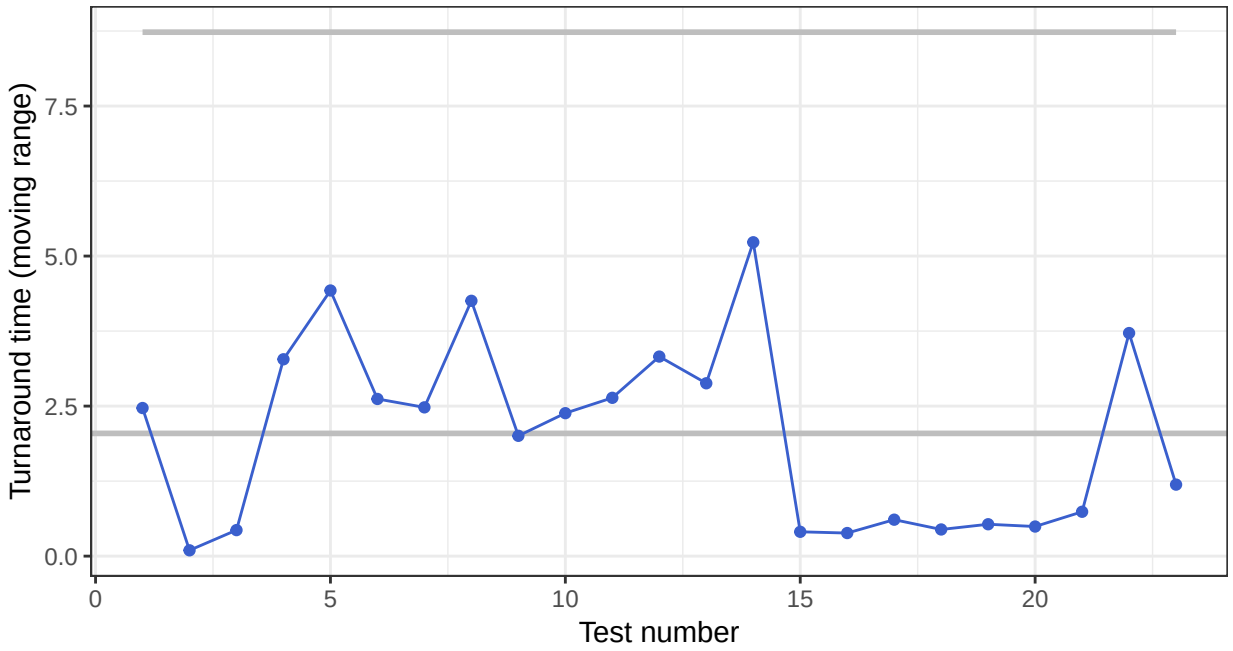
Lab results turnaround time

```
# Generate fake data
arrival = cumsum(rexp(24, 1/10))
process = rnorm(24, 5)
exit = matrix( , length(arrival))
exit[1] = arrival[1] + process[1]

for (i in 1:length(arrival)) {
    exit[i] = max(arrival[i], exit[i - 1]) + process[i]
}

# Calculate control chart inputs
subgroup.i = seq(1, length(exit))
subgroup.mr = seq(1, length(exit) - 1)

point.i = exit - arrival
point.mr = matrix(, length(point.i) - 1)
for (i in 1:length(point.i) - 1) {
    point.mr[i] = abs(point.i[i + 1] - point.i[i])
}

mean.i = mean(point.i)
mean.mr0 = mean(point.mr)
mean.mr = mean(point.mr[point.mr<=3.27*mean.mr0])
sigma.i = rep(mean.mr, length(subgroup.i))
sigma.mr = rep(mean.mr, length(subgroup.mr))
```

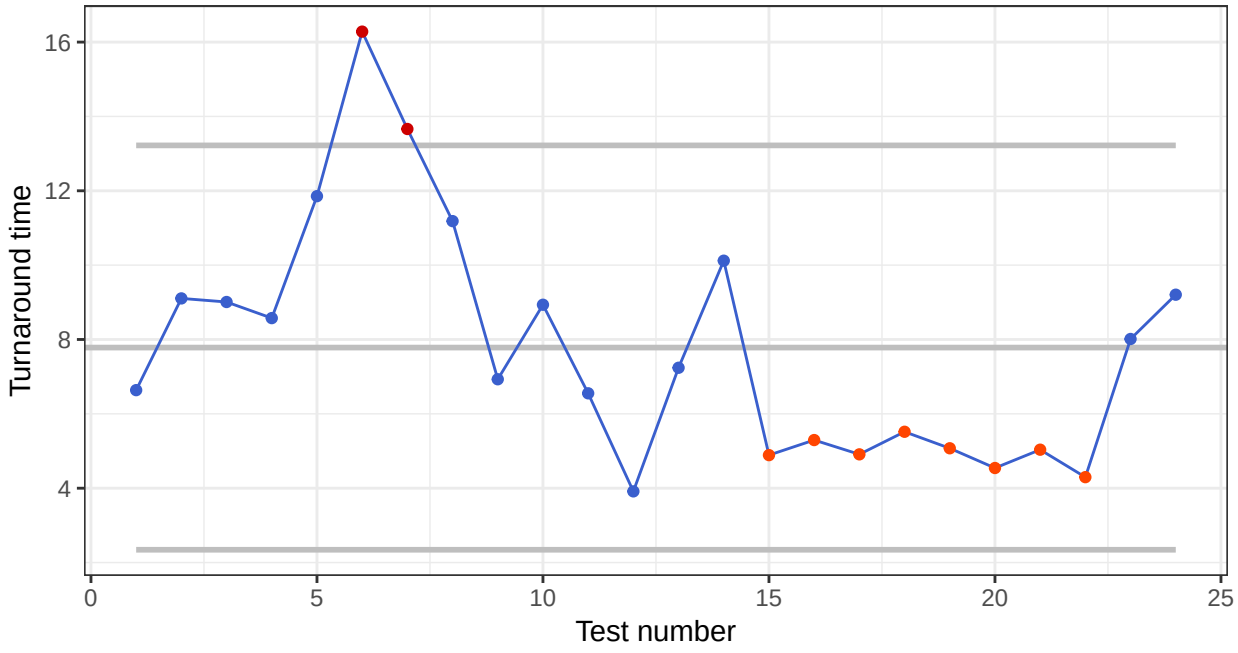
Unlike the attribute control charts, the *I-MR* chart requires a little intepretation. The *I* portion is the data itself, but the *MR* part shows the variation over time, specifically, the range between successive data points.

Look at the *MR* part first; if it's in control, then any special cause variation in the *I* portion can be attributed to a change in process. If the *MR* chart out of control, the control limits for the *I* portion will be wrong, and should not be interpreted. ISN'T THIS THE CASE BELOW?

```
# Plot MR chart
spc.plot(subgroup.mr, point.mr, mean.mr, sigma.mr, k = 3.27,
         lcl.show = FALSE, band.show = FALSE,
         label.x = "Test number", label.y = "Turnaround time (moving range)")
```




```
# Plot I chart
spc.plot(subgroup.i, point.i, mean.i, sigma.i, k = 2.66,
         lcl.min = 0, band.show = FALSE,
         label.x = "Test number", label.y = "Turnaround time")
```



10.2 $\bar{x}$ and s chart

When you have a sample or multiple measurements per subgroup, the $\bar{x}$ and s chart combination is the appropriate choice. As with the *I-MR* chart, they should always be used together.

Control limits (3) are calculated as follows:

Variable averages ($\bar{x}$): $3\frac{\bar{s}}{\sqrt{n_i}}$

Variable standard deviation (s): $3\bar{s}\sqrt{1-c_4^2}$

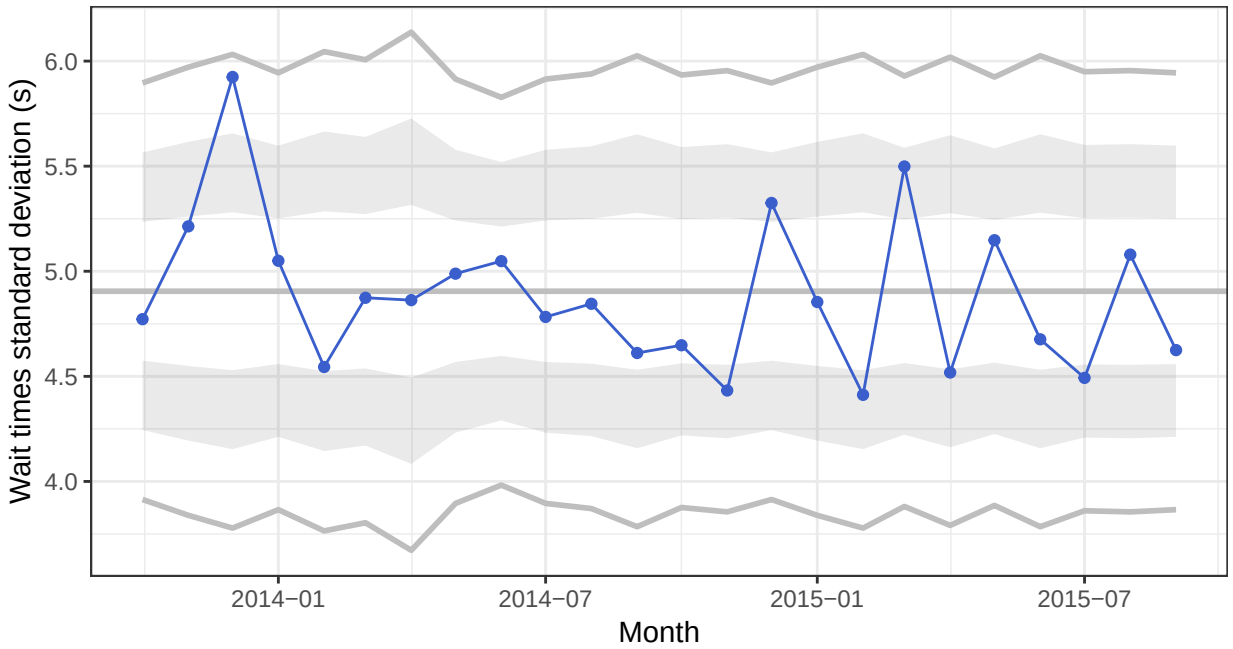
where $c_4 = \sqrt{\frac{2}{n-1}}\frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})}$

Patient wait times

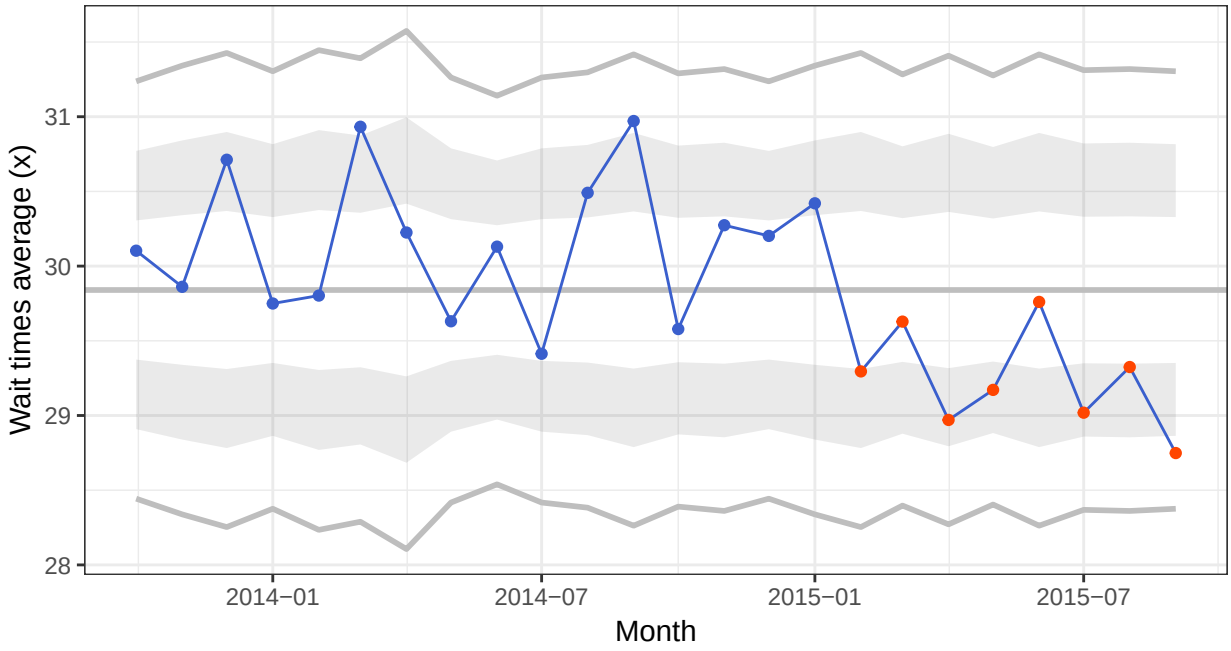
```
# Generate fake patient wait times data
set.seed(777)
waits = c(rnorm(1700, 30, 5), rnorm(650, 29.5, 5))
months = strftime(sort(as.Date('2013-10-01') + sample(0:729,
    length(waits), TRUE)), "%Y-%m-01")
sample.n = as.numeric(table(months))
dfw = data.frame(months, waits)

# Calculate control chart inputs
subgroup.x = as.Date(unique(months))
subgroup.s = subgroup.x
point.x = aggregate(dfw$waits, by = list(months), FUN = mean, na.rm = TRUE)$x
point.s = aggregate(dfw$waits, by = list(months), FUN = sd, na.rm = TRUE)$x
mean.x = mean(waits)
mean.s = sqrt(sum((sample.n - 1) * point.s ^ 2) /
        (sum(sample.n) - length(sample.n)))
sigma.x = mean.s / sqrt(sample.n)
c4 = sqrt(2 / (sample.n - 1)) * gamma(sample.n / 2) /
  gamma((sample.n - 1) / 2)
sigma.s = mean.s * sqrt(1 - c4 ^ 2)
```

As with the *I-MR* chart, you need to look at the s chart first—if it shows special-cause variation, the control limits for the $\bar{x}$ chart will be wrong. If it doesn't, you can go on to interpret the $\bar{x}$ chart.

```
# Plot s chart
spc.plot(subgroup.s, point.s, mean.s, sigma.s, k = 3,
         label.x = "Month", label.y = "Wait times standard deviation (s)")
```



```
# Plot xbar chart
spc.plot(subgroup.x, point.x, mean.x, sigma.x, k = 3,
         label.x = "Month", label.y = "Wait times average (x)")
```



10.3 EWMA chart

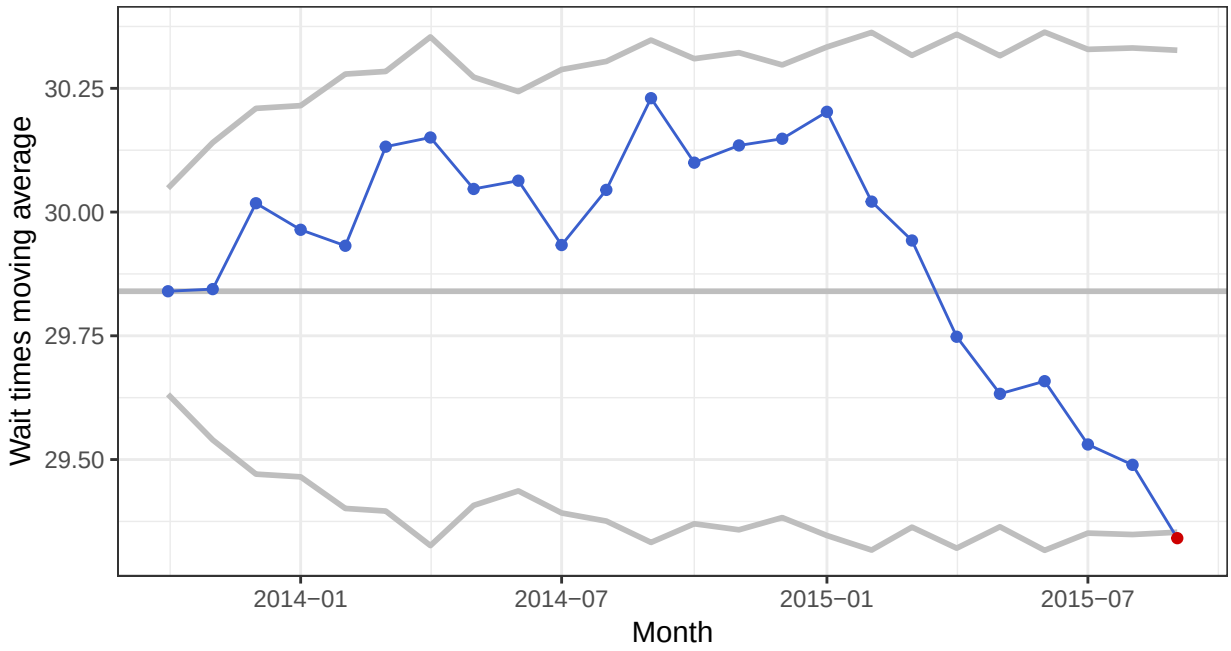
Control limits for exponentially weighted moving average (EWMA): $3\frac{\bar{s}}{\sqrt{n_i}}\sqrt{\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2i}]}$

where λ is a weight that determines the influence of past observations. If unsure choose $\lambda = 0.2$, but $0.05 \leq \lambda \leq 0.3$ is acceptable (where larger values give stronger weights to past observations).

Patient wait times (continued)

```
# Calculate control chart inputs
subgroup.z = subgroup.x
lambda = 0.2
point.z = matrix( , length(point.x))
point.z[1] = mean.x
for (i in 2:length(point.z)) {
     point.z[i] = lambda * point.x[i] + (1 - lambda) * point.z[i-1]
     }
mean.z = mean.x
sigma.z = (mean.s / sqrt(sample.n)) *
     sqrt(lambda/(2-lambda) * (1 - (1-lambda)^(seq(1:length(point.z)))))
```

```
# Plot EWMA chart
spc.plot(subgroup.z, point.z, mean.z, sigma.z, k = 3, band.show = FALSE,
         rule.show = FALSE, label.x = "Month",
         label.y = "Wait times moving average")
```



10.4 CUSUM chart

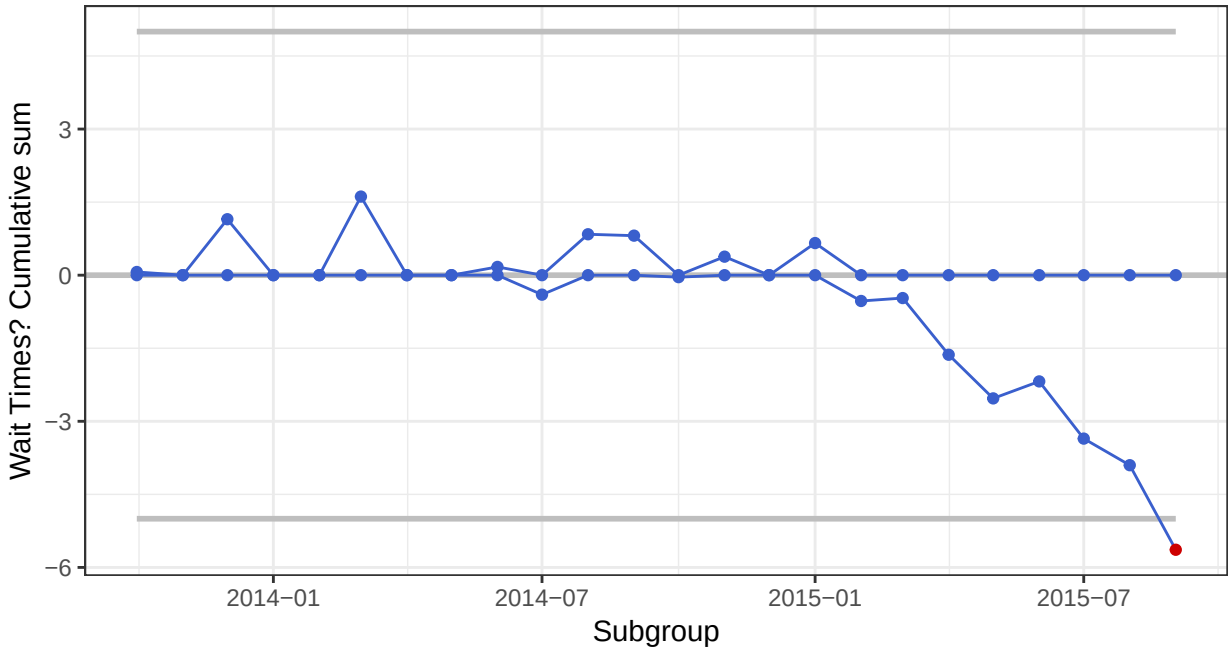
Lower and upper cumulative sums are calculated as follows:

$S_{l,i} = -\max[0, -z_i - k + S_{l,i-1}]$,
$S_{h,i} = \max[0, z_i - k + S_{h,i-1}]$
where z_i is the standardized normal score for subgroup i and $0.5 \leq k \leq 1$ is a slack value.

It is common to choose "decision limits" of ± 4 or ± 5.

```
# Calculate control chart inputs
subgroup.cusum = subgroup.x
slack = 0.5

zscore = (point.x - mean.x)/sigma.x
point.cusuml = matrix(, length(zscore))
point.cusuml[1] = -max(0, -zscore[1] - slack)

for (i in 2:length(point.cusuml)) {
    point.cusuml[i] = -max(0, -zscore[i] - slack - point.cusuml[i-1])
}

point.cusumh = matrix(, length(zscore))
point.cusumh[1] = max(0, zscore[1] - slack)

for (i in 2:length(point.cusuml)) {
    point.cusumh[i] = max(0, zscore[i] - slack - point.cusumh[i - 1])
}

mean.cusum = 0
sigma.cusum = rep(1, length(subgroup.cusum))
```

```
# Plot CUSUM chart
lower.plot = spc.plot(subgroup.cusum, point.cusuml, mean.cusum, sigma.cusum,
        k = 5, band.show = FALSE, rule.show = FALSE,
        label.y = "Wait Times? Cumulative sum")

lower.plot + geom_line(aes(y = point.cusumh), col = "royalblue3") +
    geom_point(aes(y = point.cusumh), col = "royalblue3")
```



10.5 Rare events: *t*-chart

If the time between rare events is best represented by a continuous time scale, use a t-chart. If a discrete time scale is reasonable, a g-chart (see the previous chapter) may be simpler to implement and easier to interpret without transformation, though a t-chart is also acceptable.

10.5.1 *t*-chart example

Mean for time between events (t)(not shown): $t = \bar{x}(y_i)$

where

t = time between events, where t is always > 0

$y = t^{\frac{1}{3.6}}$

Control limits for time between events (t)(not shown): $2.66MR_{bar}$

MR_{bar} = average moving range of ys, excluding those $> 3.27MR_{bar}$

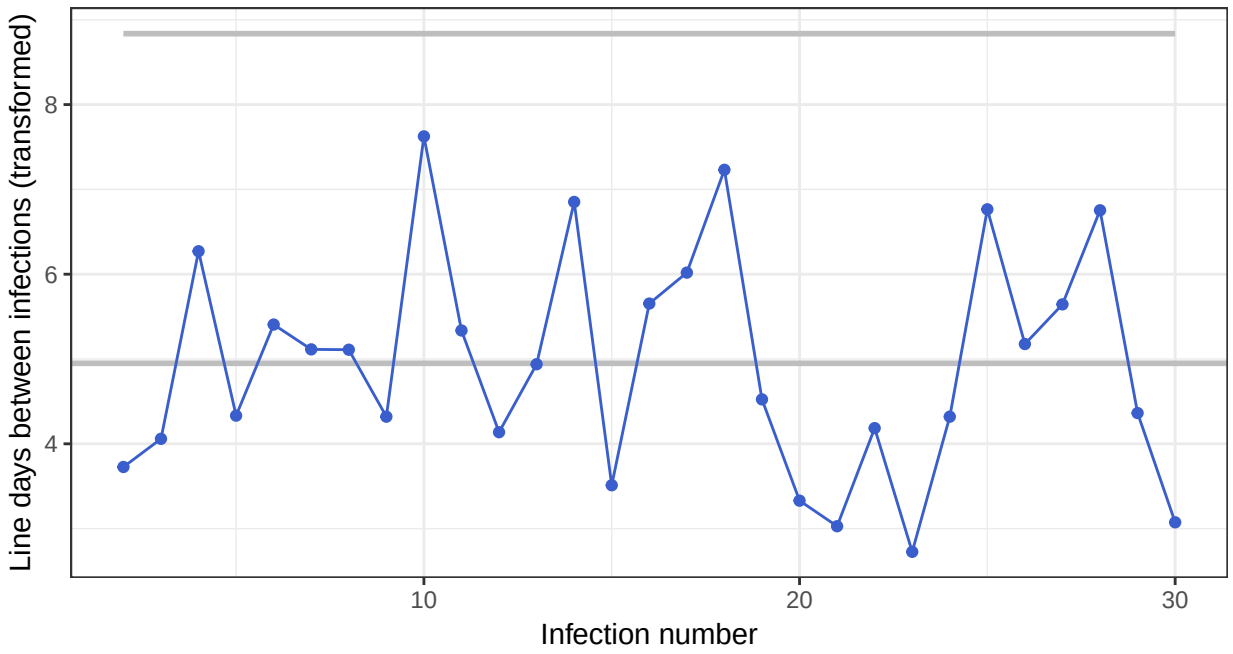
Note: t chart mean and limits can be transformed back to the original scale by raising those values to the 3.6 power. In addition, the y axis can be plotted on a log scale to make the display more symmetrical (which can be easier than explaining how the distribution works to a decision maker).

Days between infections

```
# Generate sample data using g-chart example data
y = linedays.btwn ^ (1/3.6)
```

```
mr = matrix(, length(y) - 1)
for (i in 1:length(y) - 1) {
    mr[i] = abs(y[i + 1] - y[i])
}

#=============================================================================
# Is this the right way to interpret exclude > 3.27MR? Is this exclusion recursive?

#According to Provost and Murray (https://books.google.com/books?id=pRLcaOkswQsC) you first calculate M
#=============================================================================
mr = mr[mr <= 3.27*mean(mr)]
mr_prime = mean(mr)

# Calculate t chart inputs
subgroup.t = subgroup.g
point.t = y
central.t = mean(y)
sigma.t = rep(mr_prime, length(point.t))
```

```
# Plot t chart
spc.plot(subgroup.t, point.t, central.t, sigma.t, lcl.show = FALSE,
         band.show = FALSE, rule.show = FALSE,
         lcl.min = 0, k = 2.66, label.x = "Infection number",
         label.y = "Line days between infections (transformed)")
```



Chapter 11

Time series data exploration

Chapter 3 contains the basic exploratory data analysis you should do before using SPC tools. But there are many other time series-oriented analytic tools available that can help you understand the data more completely.

11.1 Time series EDA

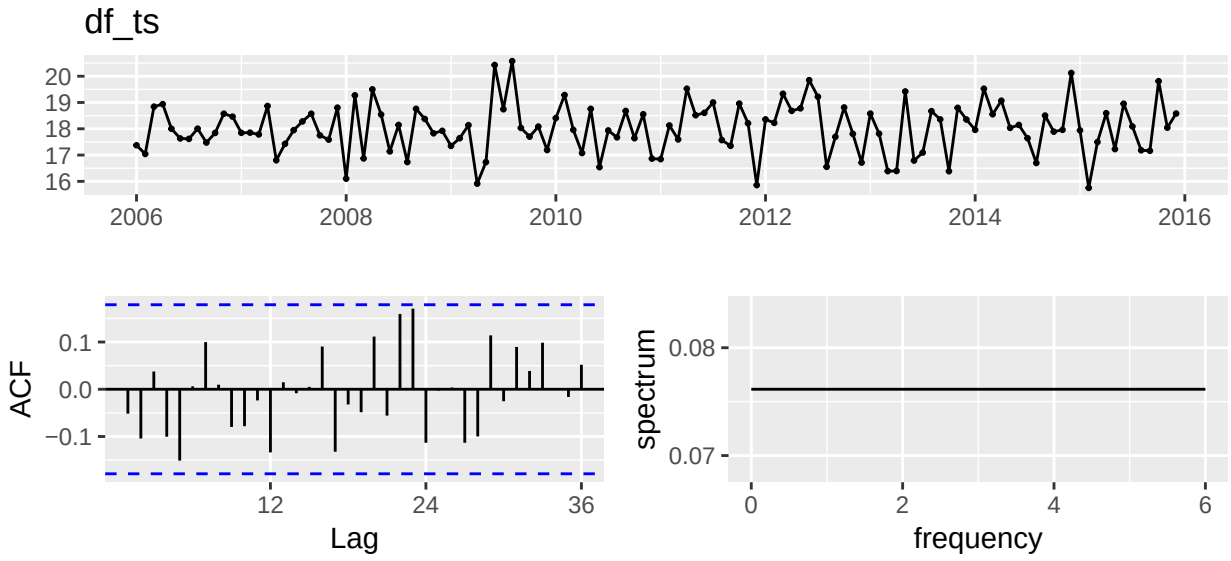
There is usually far more information in a time series than is typically explored with basic SPC methods. You can create a variety of exploratory and diagnostic plots that help you understand the data more thoroughly.

Because the fake data (`df_ts`) used in previous chapters has no time series patterns, we'll use it alongside a data set with clear patterns (`beer`) so we can explore what these EDA tools show with and without time-related patterns. Note the `beer` data is measured quarterly whereas the `df_ts` data is measured monthly.

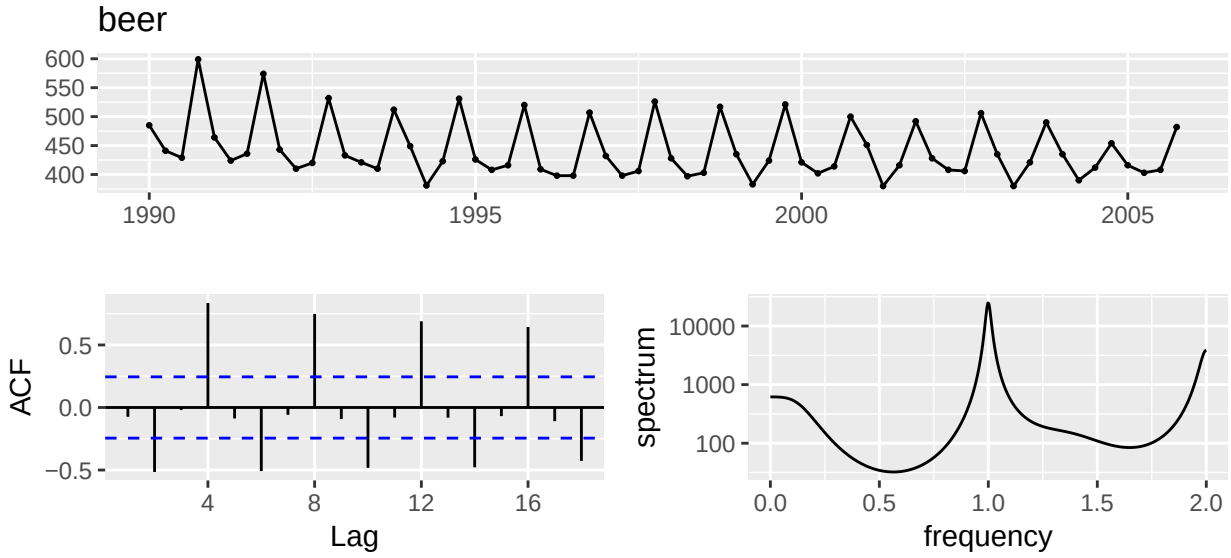
```
# Use Australian beer data, trimmed to a 15 year subset
data(ausbeer, package = "fpp2")
beer = window(ausbeer, start = 1990.00, end = 2005.75)
```

For comparison's sake, here are summaries of each time series that show the time series itself, the autocorrelation function, and a (simplified) periodogram:

```
# No temporal patterns in data
ggtsdisplay(df_ts, plot.type = "spectrum")
```




```
# Temporal patterns in data
ggtsdisplay(beer, plot.type = "spectrum")
```



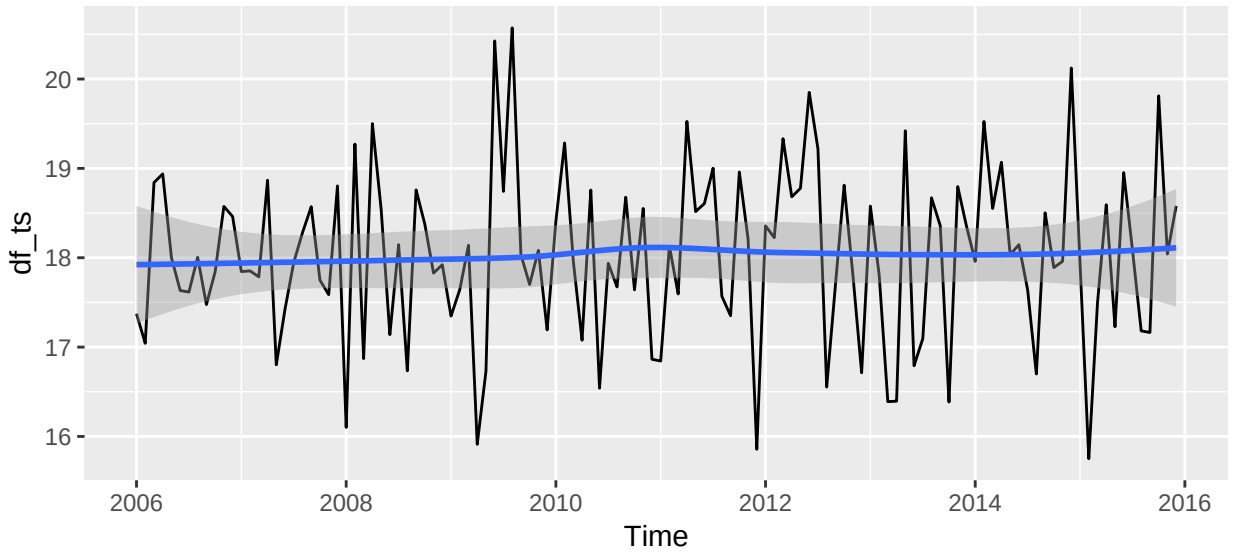
11.1.1 Overall trend (if any)

The first thing to look for is whether there is a trend. The simplest way to let the data speak for this is by using a loess smoother.

The `autoplot` function in the `forecast` package provides several out-of-the-box plots for time series data, and since it's built over `ggplot2`, it can use those functions as well.

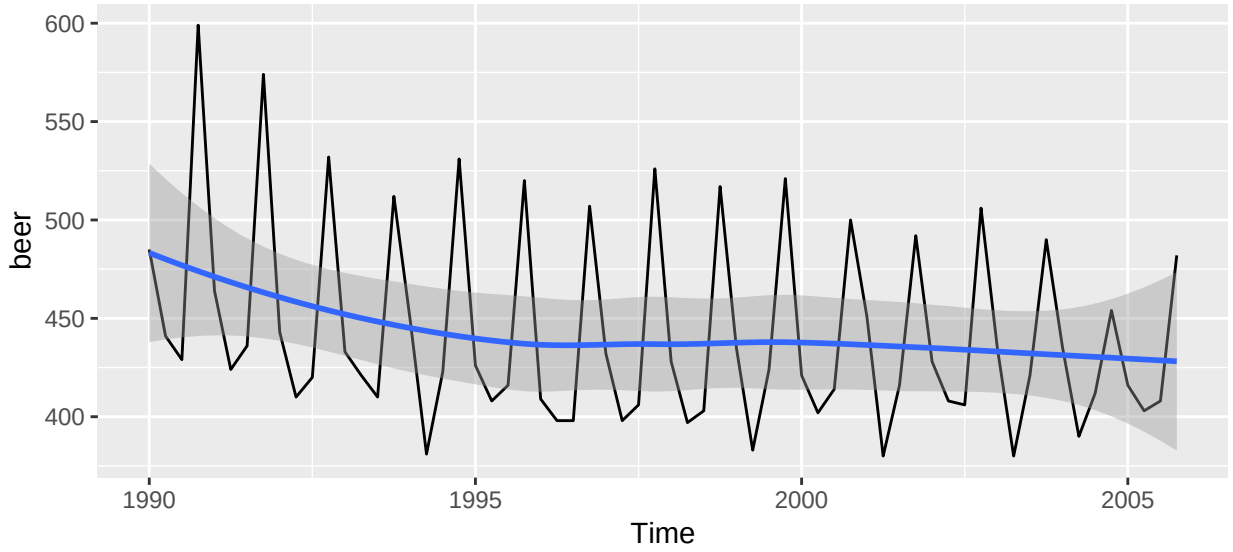
The `df_ts` time series data set has absolutely no trend at all.

```
autoplot(df_ts) +
  geom_smooth()
```



There does seem to be an initial overall declining trend in the `beer` data that seems to flatten out.

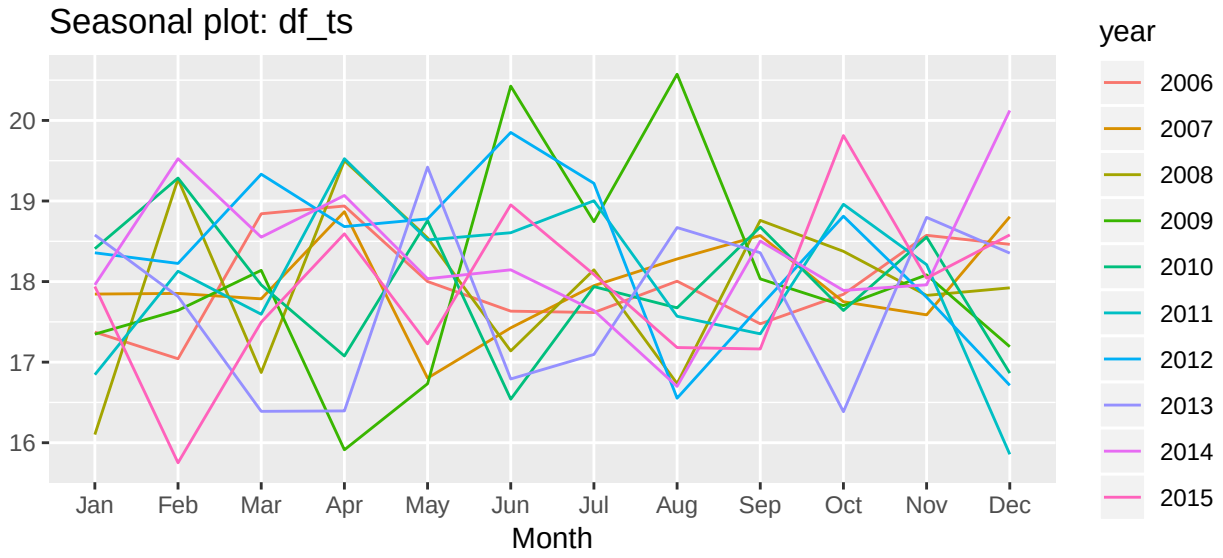
```
autoplot(beer) +
  geom_smooth()
```



11.1.2 Seasonplot

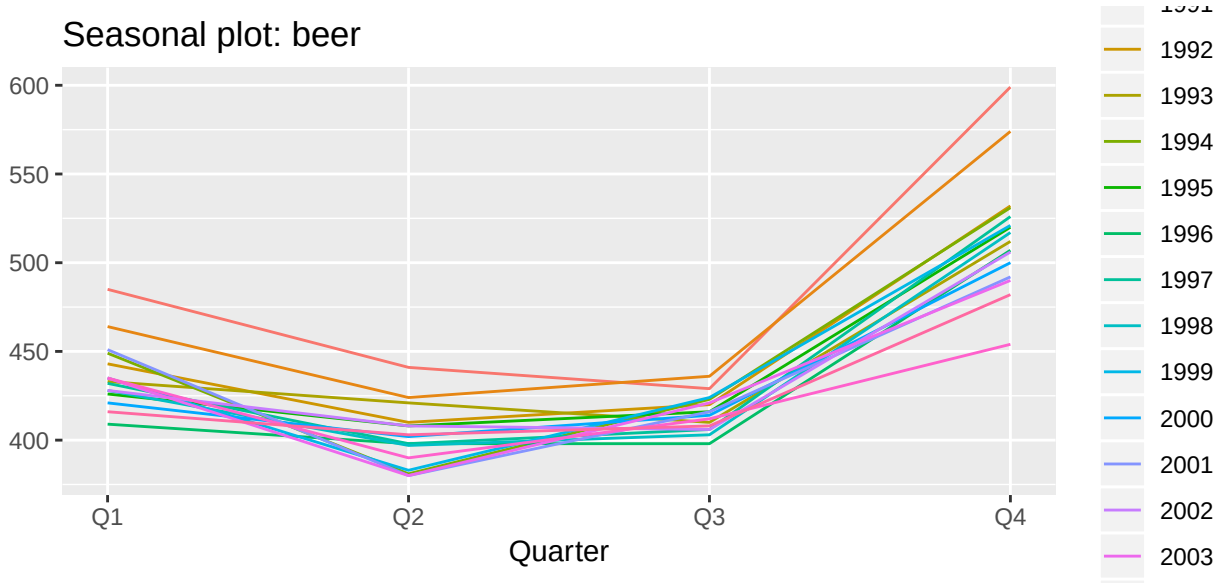
The seasonplot places each year as its own line over an x-axis of the sequential frequency, which defaults to the frequency of the time series. When there's no seasonal pattern across or within that frequency, the plot looks like spaghetti as the result of being driven by natural variation.

```
ggseasonplot(df_ts)
```



When there is a pattern in the time series, patterns emerge. In this case, the fourth quarter increase above the other quarters is quite evident.

```
ggseasonplot(beer)
```

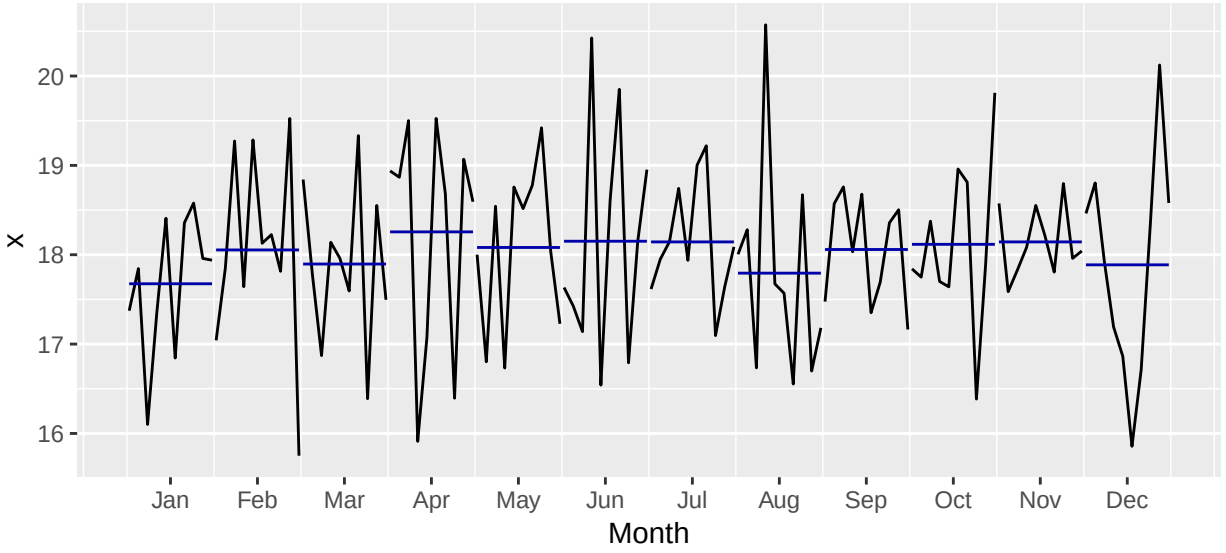


11.1.3 Monthplot

A monthplot puts all years into seasonal groups, where each line is a group (e.g., month) and each point in that line is an individual year. When there is a lengthy trend in the series, you can see it in a consistent up or down pattern in each seasonal group. You can also compare central tendencies across those groups with a mean or median line.

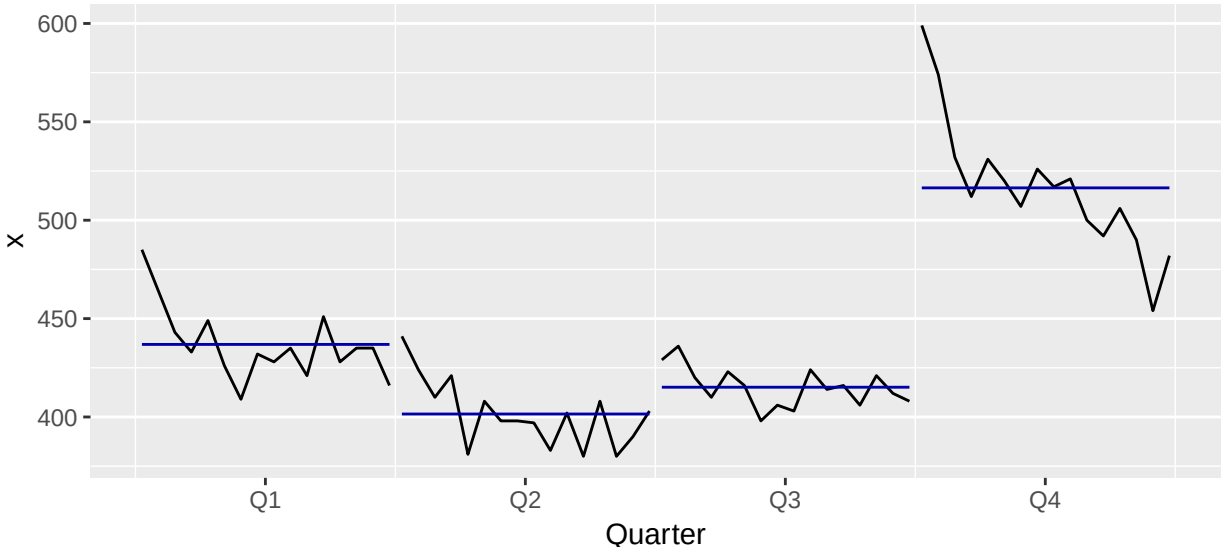
Data with no inherent pattern shows up as noise:

```
ggmonthplot(df_ts)
```



Whereas in a time series with temporal patterns, you can see both the higher levels in Q4 as compared with the other quarters, but you can also see that this quarter's values is declining over the years, a pattern echoed to lesser extent in the early years' values for the other quarters.

```
ggmonthplot(beer)
```



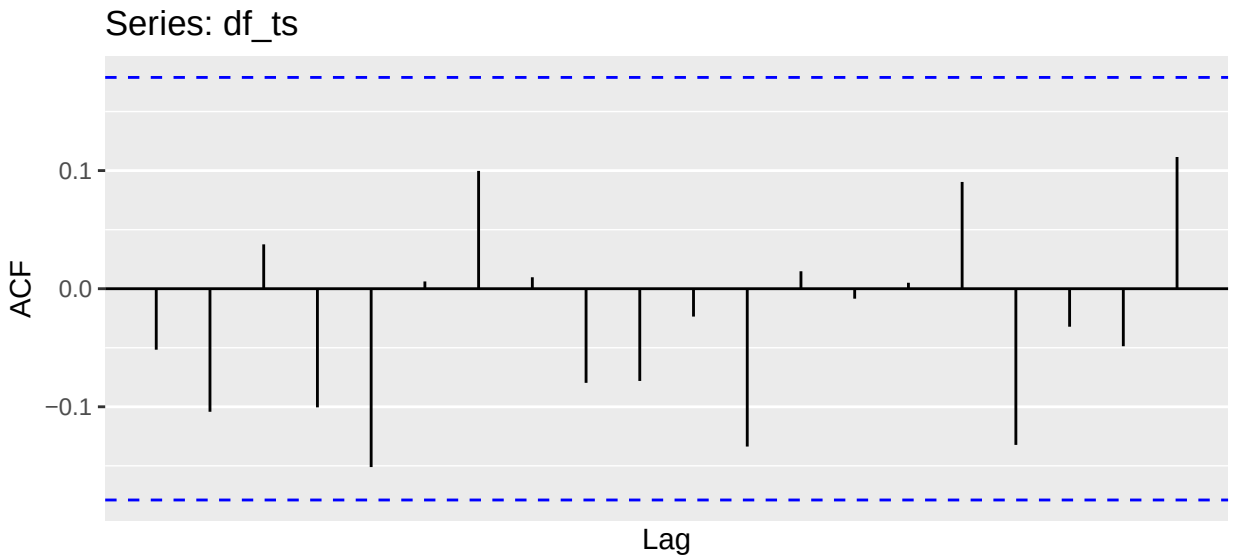
11.1.4 Autocorrelation

We've touched on autocorrelation in other portions of this book, and will discuss it further later in this chapter.

The `acf` function provides a graphical summary of the autocorrelation function, with each data point correlated with a value at increasing lagged distances from itself. Each correlation is plotted as a spike; spikes that go above or below the dashed line suggest that significant positive or negative autocorrelation, respectively, occurs at that lag (at the 95% confidence level). If all spikes occur inside those limits, it's safe to assume that there is no autocorrelation. If only one or perhaps two spikes exceed the limits slightly, it could be due simply to chance. Clear patterns seen in the acf plot can indicate autocorrelation even when the values do not exceed the limits.

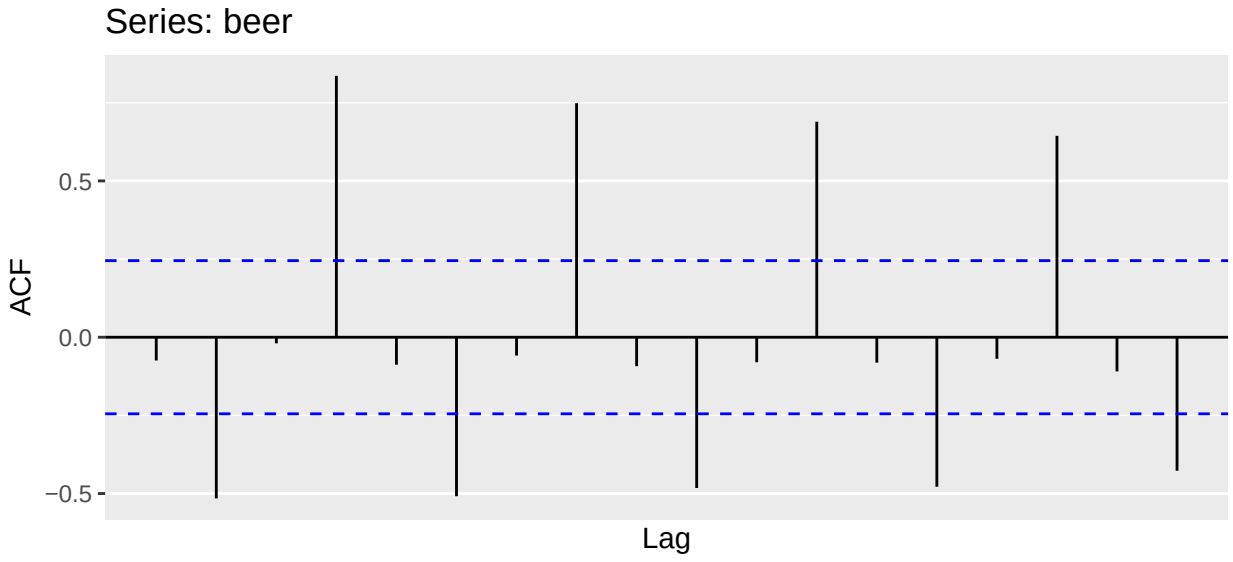
With the `df_ts`, there is no autocorrelation and no obvious pattern, and the correlation values themselves (y-axis) are tiny:

```
# acf plot using the autoplot function instead of base for the ggplot look
autoplot(acf(df_ts, plot = FALSE))
```



But with the `beer` data, the patterning is obvious, especially at lags 2 (6 months apart) and 4 (1 year apart), and the correlation values are quite large.

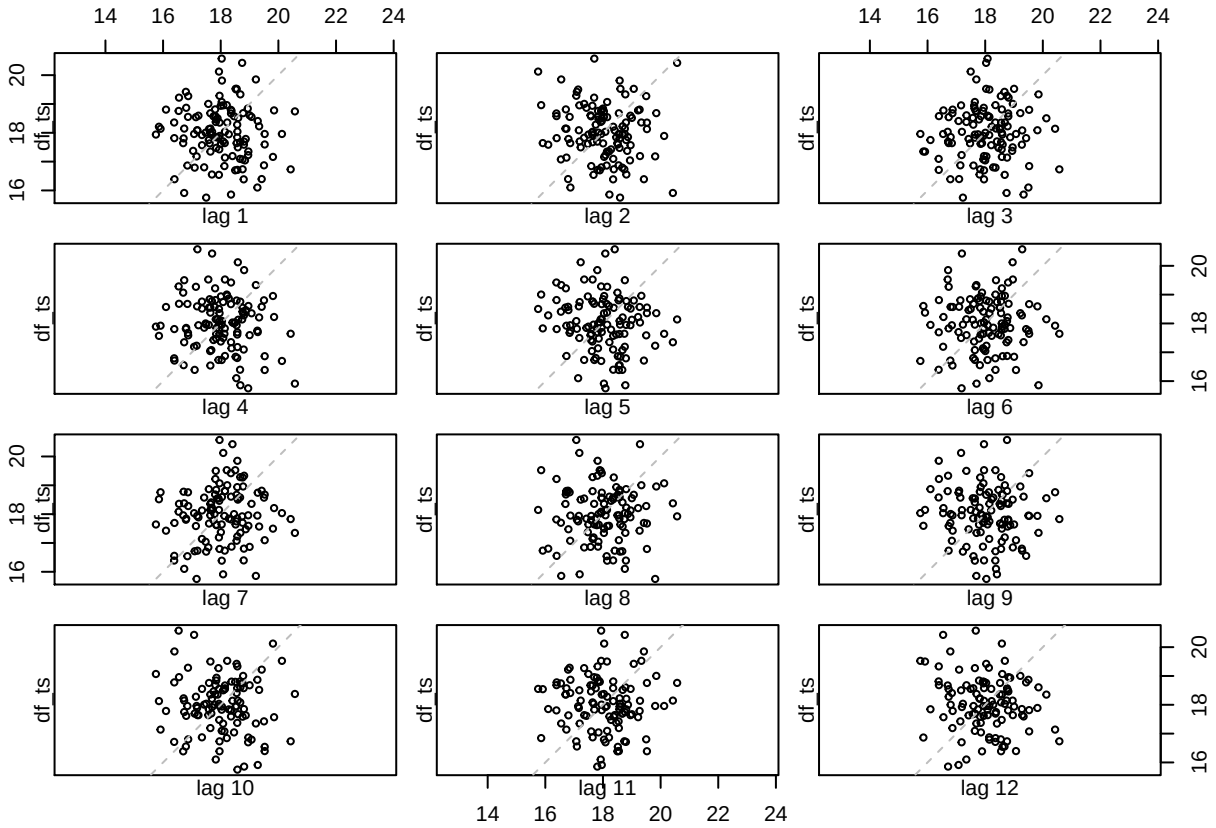
```
# acf plot using the autoplot function instead of base for the ggplot look
autoplot(acf(beer, plot = FALSE))
```



The autocorrelation function is most concisely plotted with the approach above, but you can also plot the increasing lags against an initial value in individual scatterplots. If the points look like a shotgun target, there's no autocorrelation. Patterns in the points indicate autocorrelation in the data. Patterns strung along or perpendicular to the 1:1 dashed line suggest strong positive and negative correlation, respectively, though any sort of pattern is cause for concern.

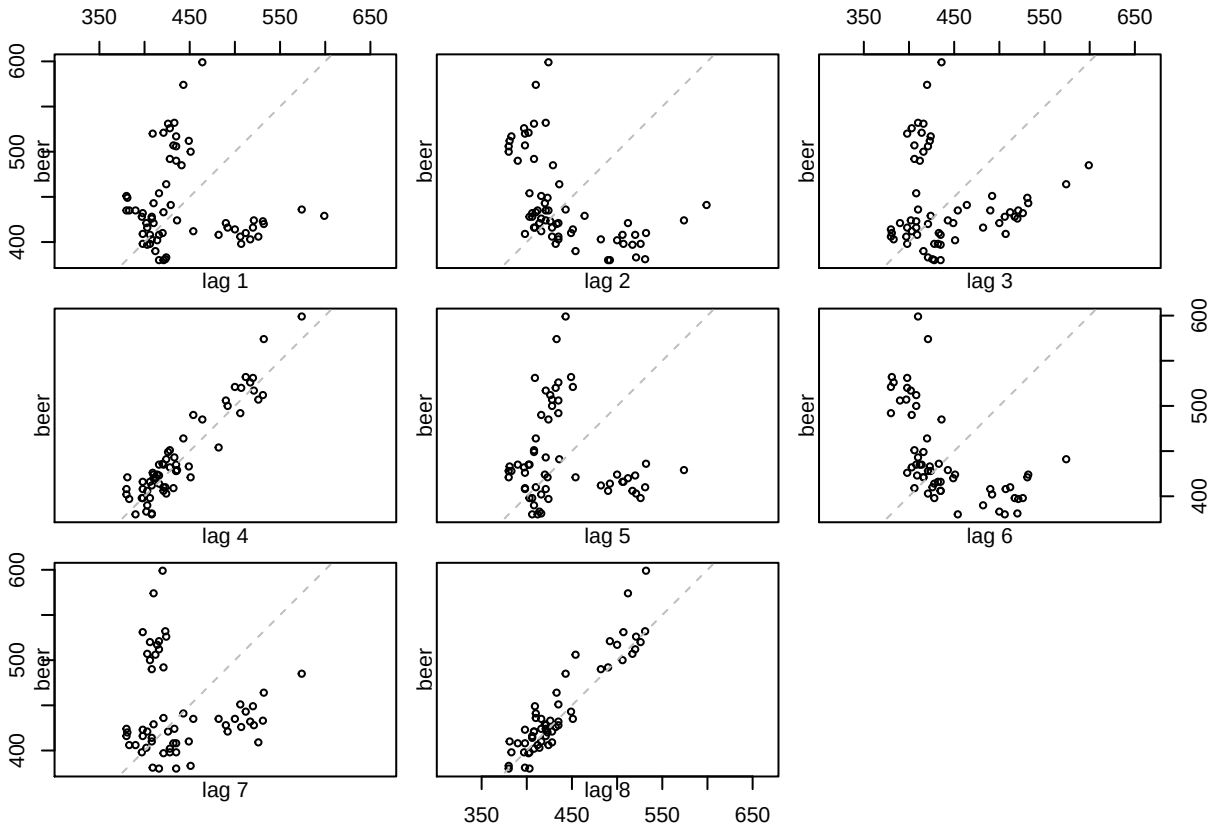
The lagplot for the `df_ts` data shows the shotgun target "pattern" that suggests that only random variation is present.

```
# Scatterplot of df_ts autocorrelation through first 12 lags
lag.plot(df_ts, lags = 12, do.lines = FALSE)
```



But clear patterns emerge—especially at lag 4 (1 year apart)—in the lagplot for the `beer` data.

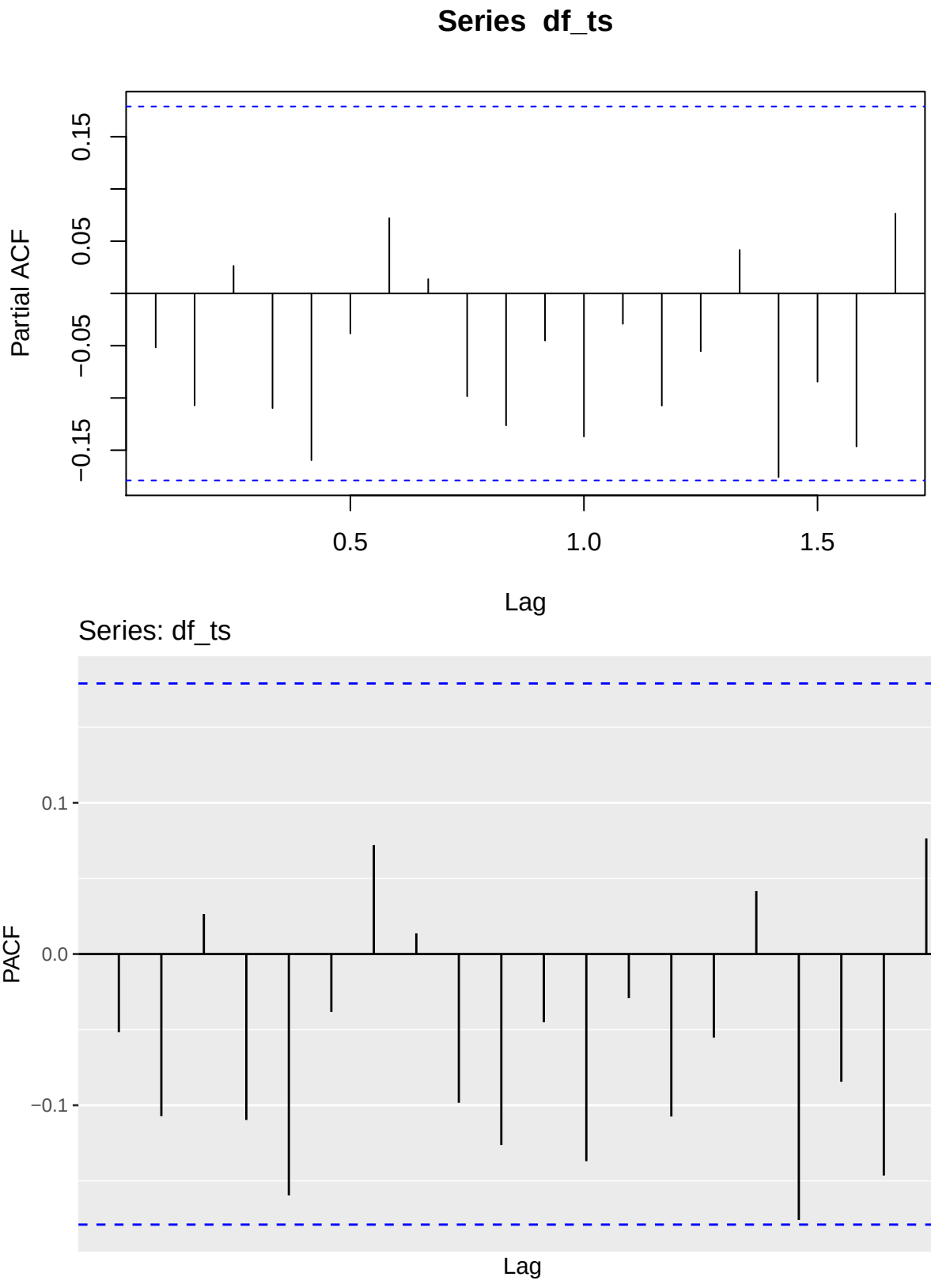
```
# Scatterplot of beer data autocorrelation through first 8 lags
lag.plot(beer, lags = 8, do.lines = FALSE)
```



The `pacf` function gives you a partial autocorrelation plot, which is the correlation between the first value and each individual lag. It's the same information provided by the lag plot, only more compact as it only displays the correlation value itself. This can be quite useful in identifying cycles in data.

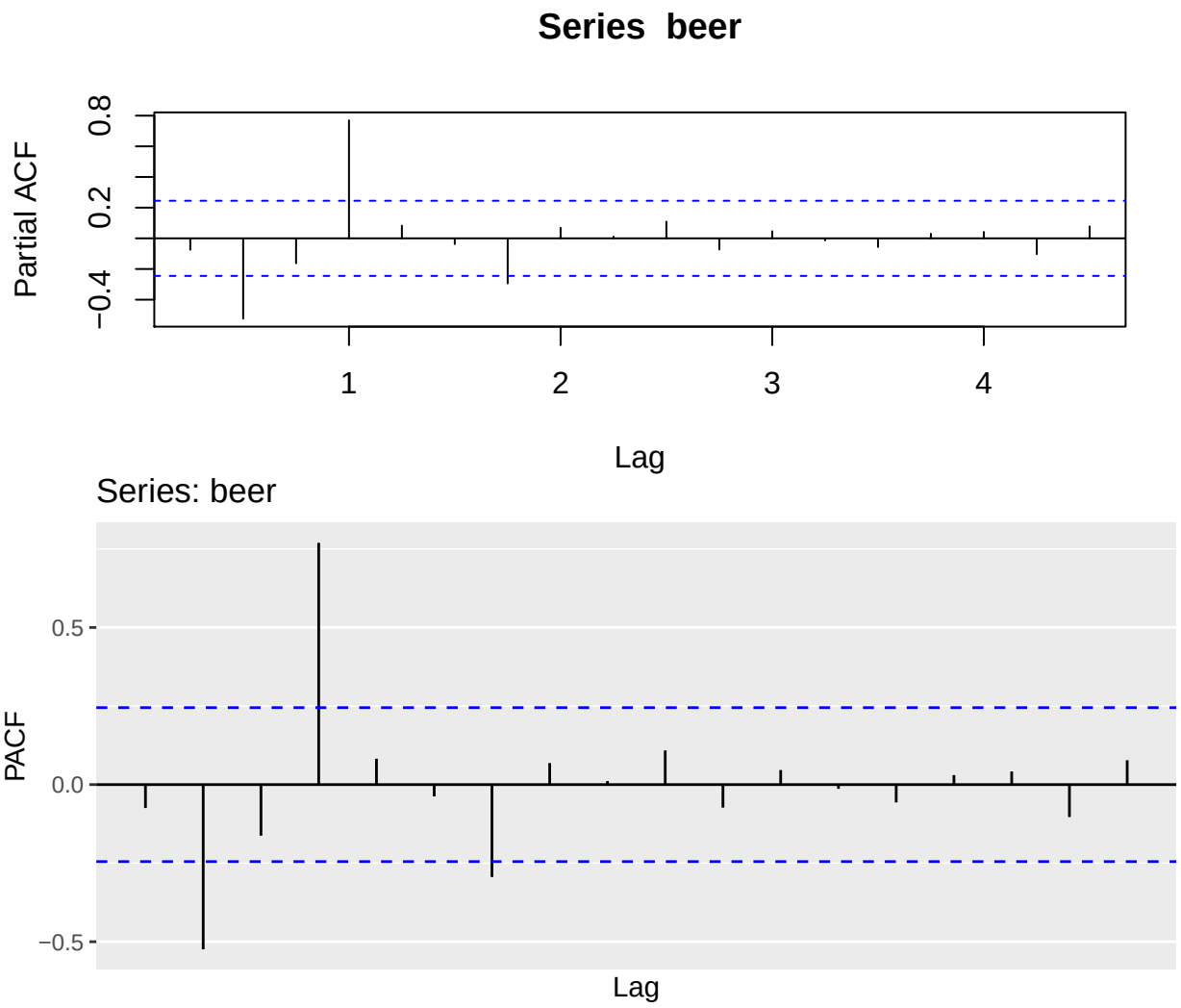
A `pacf` plot for `df_ts` data shows the random noise we'd expect, as well as tiny correlation values.

```
autoplot(pacf(df_ts))
```



Using the `beer` data shows the partial autocorrelation pattern. The spike at the second line indicates that there is a moderate negative relationship in values 6 months (2 quarters) apart, and the spike at the fourth line shows there's a strong positive relationship in values 1 year (4 quarters) apart.

```
autoplot(pacf(beer))
```



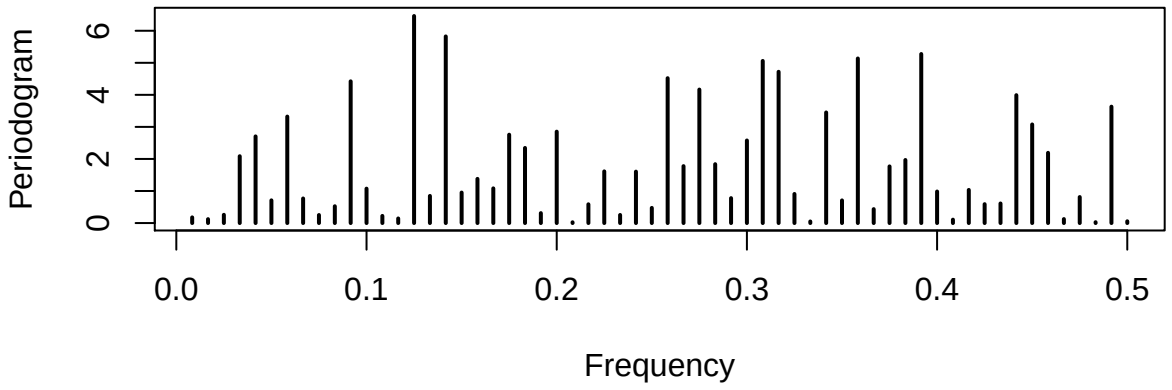
11.1.5 Cycles

Periodograms allow you to explore a time series for cycles that may or may not be regular in timing (which makes it slightly distinct from seasonality). Sunspot cycles are a classic example at ~11 years, a time span that obviously doesn't correspond to calendar seasons and frequencies.

Spikes in the periodogram designate possible cycle timing lengths, where the x-axis is based on frequency. The reciprocal of the frequency is the time period, so a spike in a periodogram for an annual series at a frequency of 0.09 suggests a cycle time of about 11 years.

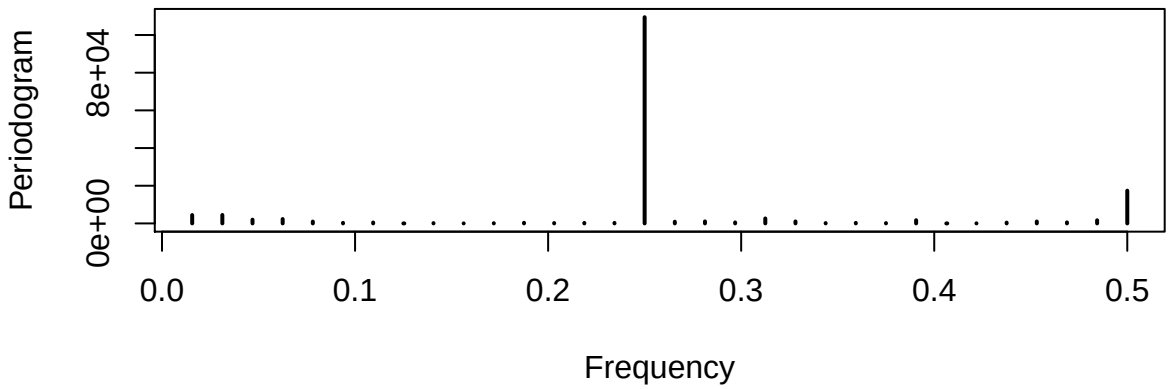
A bunch of spikes scattered across the plot, or a more or less flat line with no real spikes, both suggest that there is no cyclic pattern in the data.

```
TSA::periodogram(df_ts)
```



A clear spike occurs in the `beer` data at a frequency of 0.26, a time period of about 4. Since this is quarterly data, it confirms the annual pattern seen in several plots above.

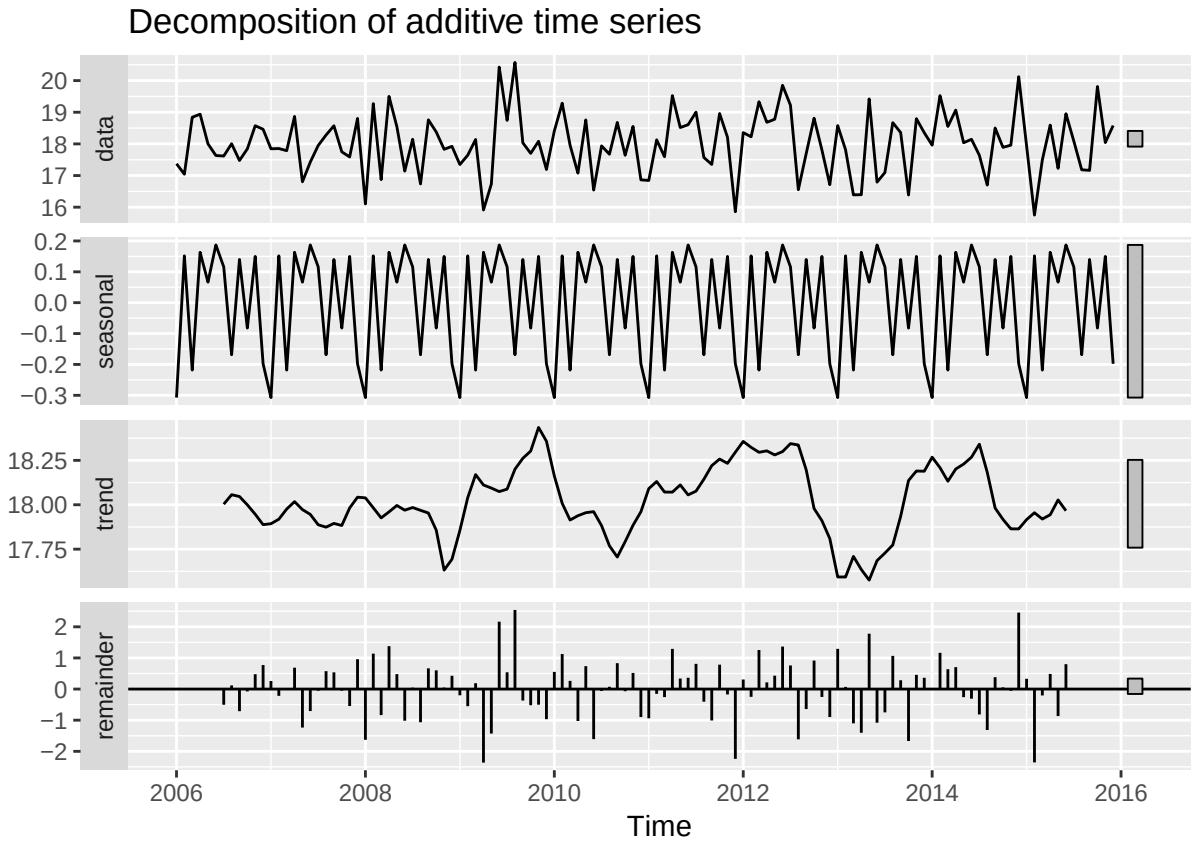
```
TSA::periodogram(beer)
```



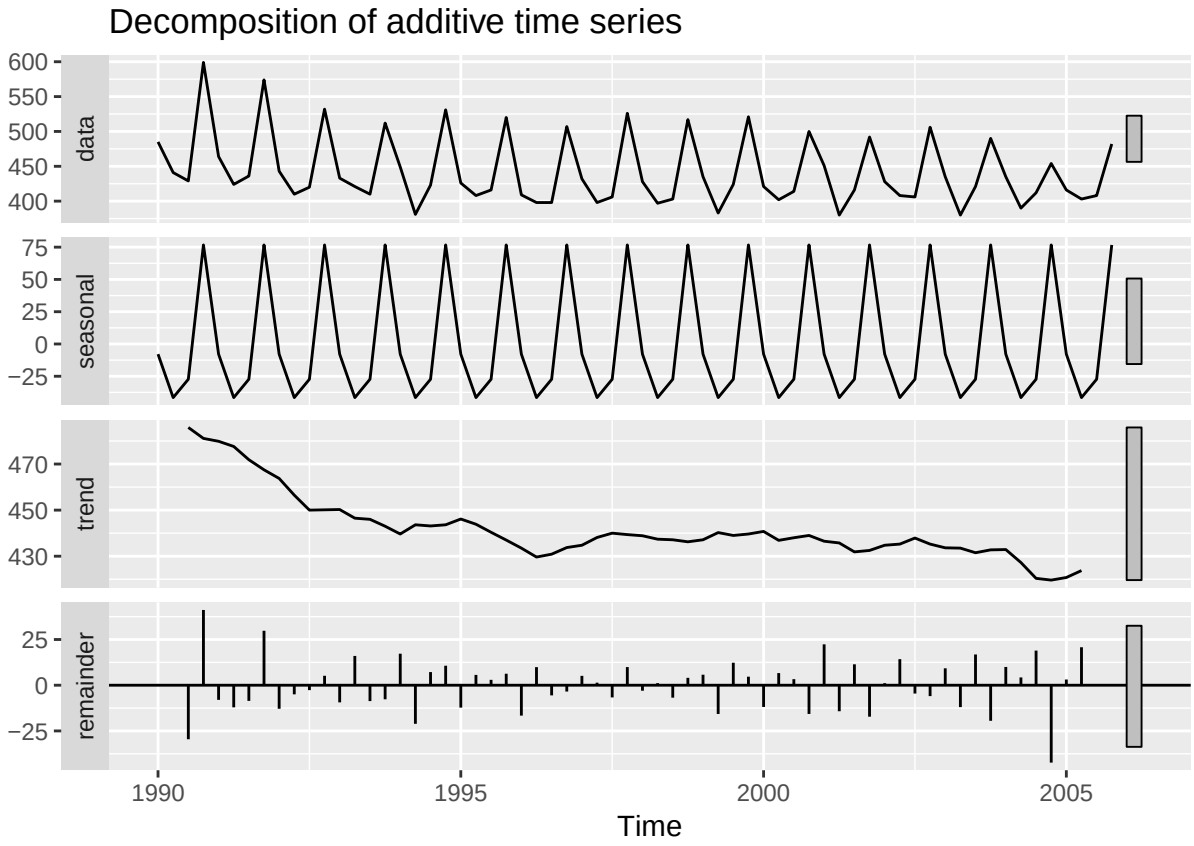
11.1.6 Decomposition

The `decompose` function extracts the major pieces of a time series, while the `autoplot` function presents the results using `ggplot2` for a cleaner look.

```
autoplot(decompose(df_ts))
```




```
autoplot(decompose(beer))
```



11.1.7 Seasonal adjustment

The `seasonal` package uses the U.S. Census Bureau's X-13ARIMA-SEATS method to calculate seasonal adjustment. The `seas` function can be used to view or save the results into another object. WHAT EXACTLY IS A SEASONAL ADJUSTMENT? WHAT IS IT USED FOR?

```
# Convert ts to data frame
beer_df = tsdf(beer)

# Get seasonally-adjusted values and put into data frame
beer_season = seasonal::seas(beer)
beer_df$y_seasonal = beer_season$data[,3]

# Show top 6 lines of data frame
knitr::kable(head(beer_df))
```

x	y	y_seasonal
1990.00	485	492.4053
1990.25	441	490.3219
1990.50	429	467.5433
1990.75	599	495.5152
1991.00	464	482.9547
1991.25	424	460.9614

If you just want to plot it on the fly, `ggseas` provides the `stat_seas` function for use with `ggplot2`. As with all ggplots, you need a data frame first, which the `tsdf` function provides.

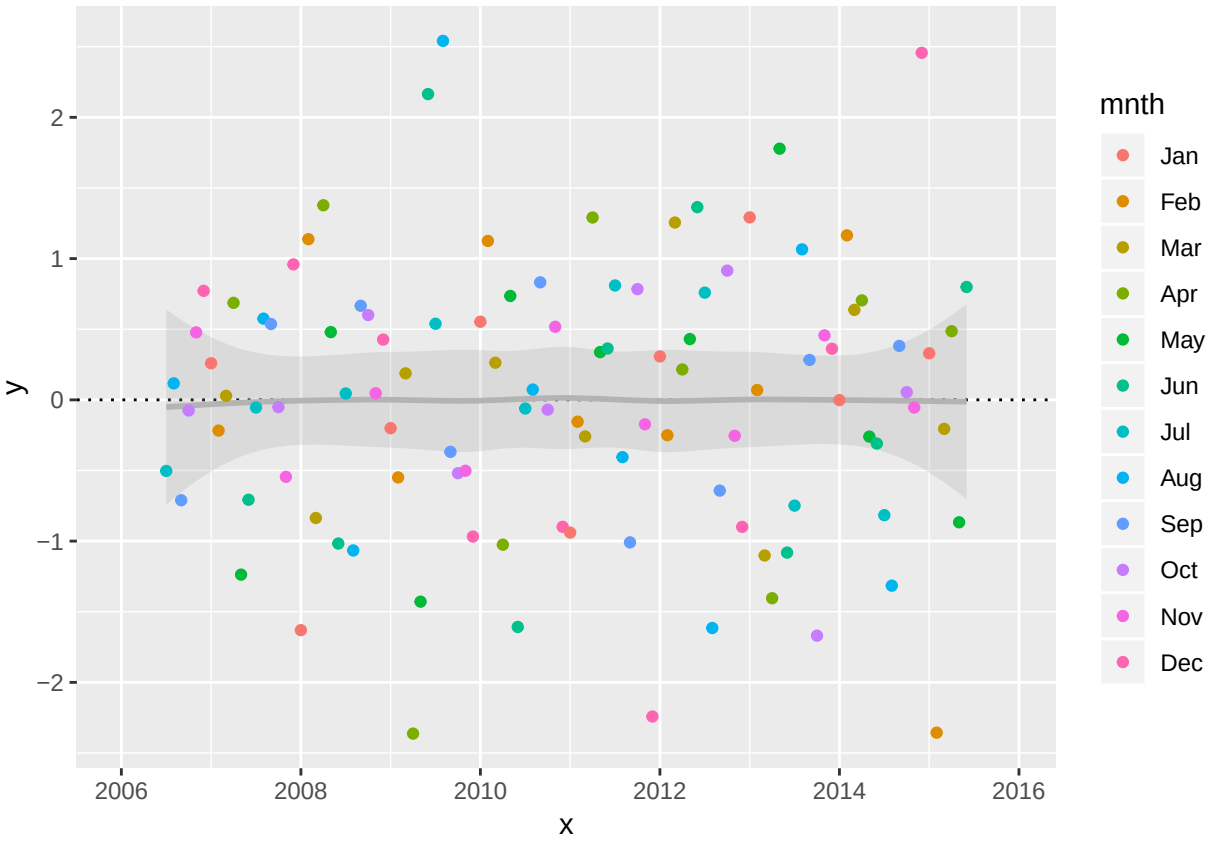
```
# Convert ts to data frame
df_ts_df = tsdf(df_ts)

# Plot original and seasonally adjusted data
ggplot(df_ts_df, aes(x, y)) +
  geom_line(color="gray70") +
  stat_seas(color="blue")
```

```
# Plot original and seasonally adjusted data
ggplot(beer_df, aes(x, y)) +
  geom_line(color="gray70") +
  stat_seas(color="blue")
```

11.1.8 Residuals

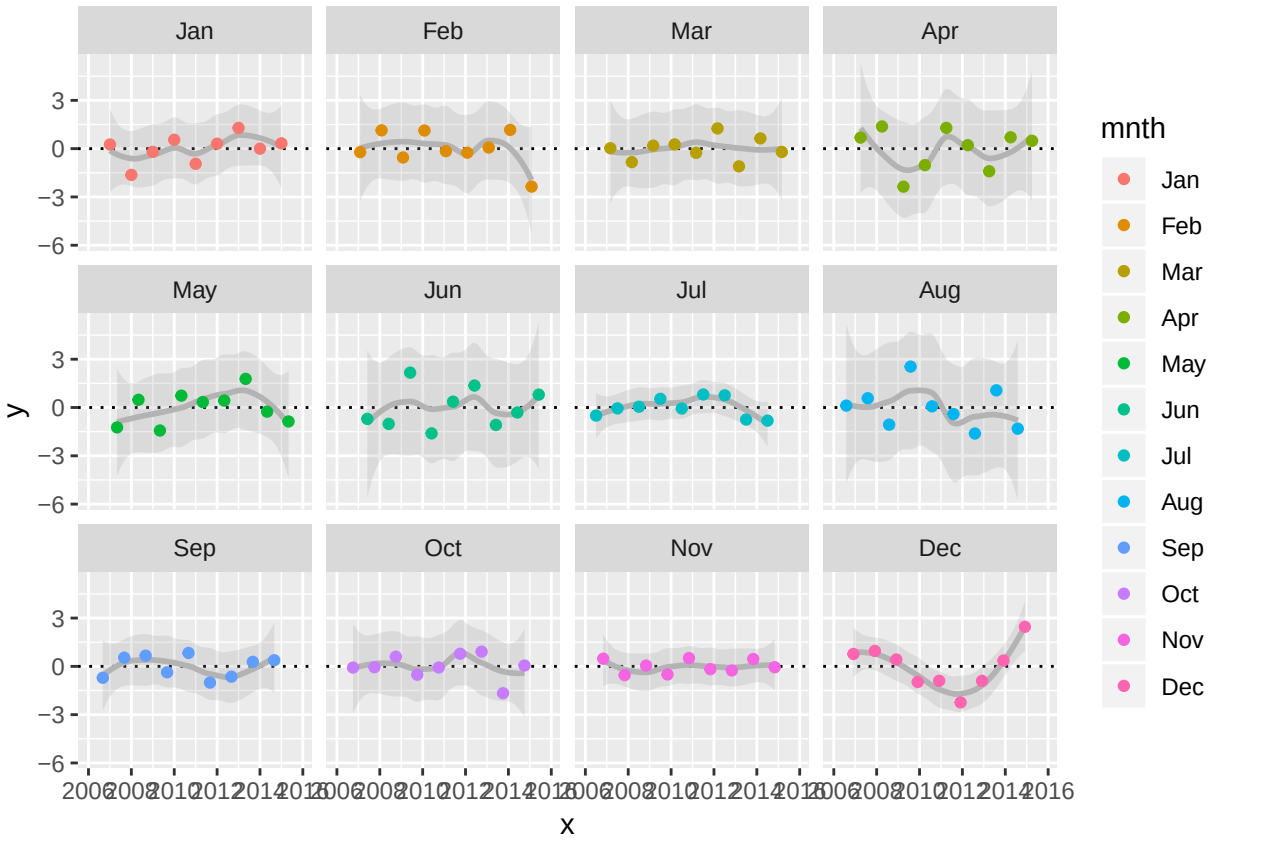
Residuals—the random component of the time series—can also be explored for potential patterns. Ideally, you don't want to see patterns in the residuals, but they're worth exploring in the name of thoroughness.

```
# Convert ts residuals to data frame
df_ts_df_rand = tsdf(decompose(df_ts)$random)

# Add month as a factor
df_ts_df_rand$mnth = factor(rep(month.abb, 10), levels = month.abb)

# Plot residuals
# No apparent patterns
ggplot(df_ts_df_rand, aes(x, y)) +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_smooth(color = "gray70", alpha = 0.2) +
  geom_point(aes(color = mnth))
```

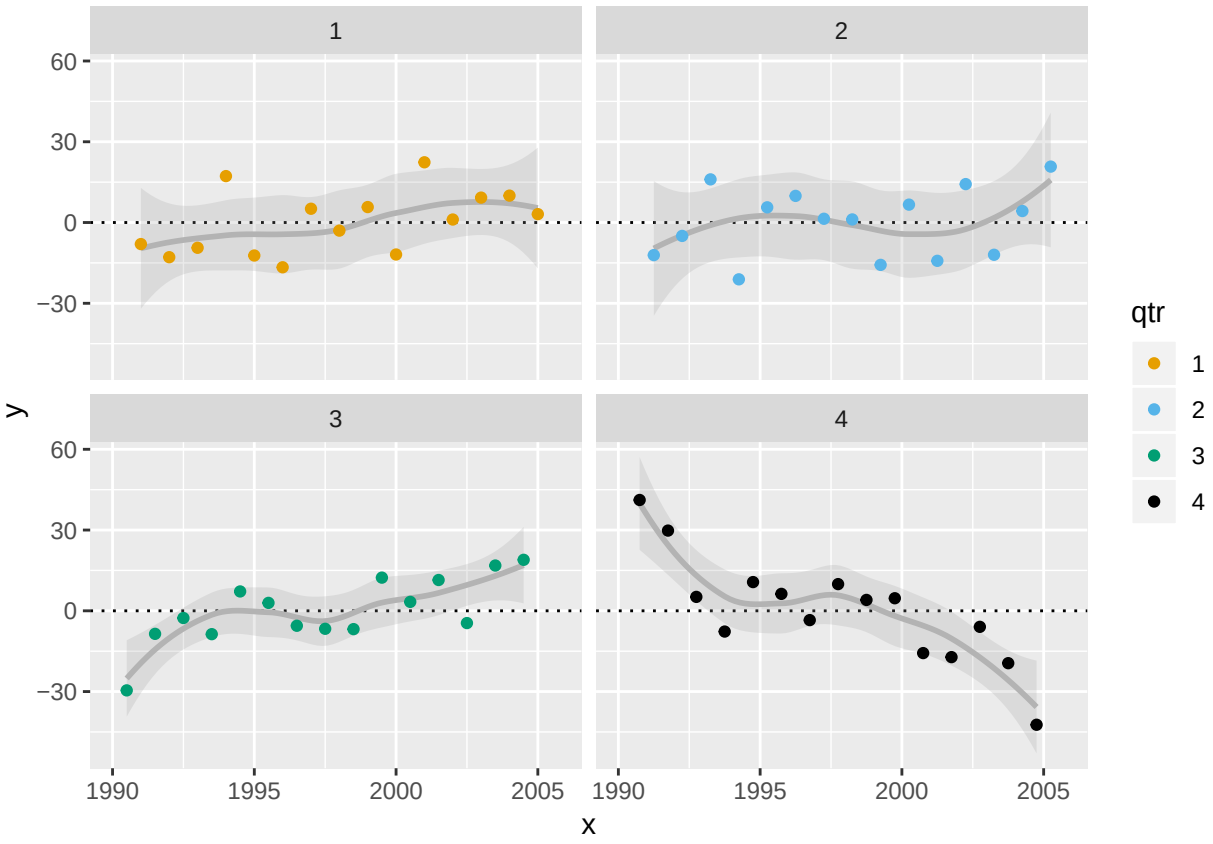



```
# Convert ts residuals to data frame
beer_df_rand = tsdf(decompose(beer)$random)

# Add quarter as a factor
beer_df_rand$qtr = factor(quarter(date_decimal(beer_df_rand$x)))

# Plot residuals, with custom colors

#LOOKS LIKE THERE MIGHT BE A PATTERN.  MAYBE?

ggplot(beer_df_rand, aes(x, y)) +
  geom_hline(yintercept=0, linetype="dotted") +
  geom_smooth(color = "gray70", alpha = 0.2) +
  geom_point(aes(color = qtr)) +
  scale_color_manual(values=c("#E69F00", "#56B4E9", "#009E73", "#000000"))
```


```
# Residuals faceted by month
# Is December weird? Rest seem ok

# DO WE WANT TO SAY SOMETHING ABOUT HOW THESE RESIDUAL PLOTS TRACK THE WITHIN-MONTH PATTERNS WE SAW IN

ggplot(df_ts_df_rand, aes(x, y)) +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_smooth(color = "gray70", alpha = 0.2) +
  facet_wrap(~ mnth) +
  geom_point(aes(color = mnth))
```




```
# Residuals faceted by quarter
ggplot(beer_df_rand, aes(x, y)) +
  geom_hline(yintercept=0, linetype="dotted") +
  geom_smooth(color = "gray70", alpha = 0.2) +
  facet_wrap(~ qtr) +
  geom_point(aes(color = qtr)) +
  scale_color_manual(values=c("#E69F00", "#56B4E9", "#009E73", "#000000"))
```



11.1.9 Accumuluation plots

You can use the EDA tools above on rates, numerators, and denominators alike to explore patterns. When you do have a numerator and a denominator that create your metric, you can also plot them against each other, looking at the accumulation of each over the course of a relevant time frame (e.g., a year).

To illustrate, we'll create a new time series for monthly central line associated infections, set up so that the last two years of a 10 year series are based on a different process.

```
# Generate sample data
set.seed(54)
bsi_8yr = data.frame(Linedays = sample(1000:2000, 96), Infections = rpois(96, 4))
bsi_2yr = data.frame(Linedays = sample(1200:2200, 24), Infections = rpois(24, 3))
bsi_10yr = rbind(bsi_8yr, bsi_2yr)
bsi_10yr$Month = seq(as.Date("2007/1/1"), by = "month", length.out = 120)
bsi_10yr$Year = year(bsi_10yr$Month)
bsi_10yr$Rate = round((bsi_10yr$Infections / bsi_10yr$Linedays * 1000), 2)
```

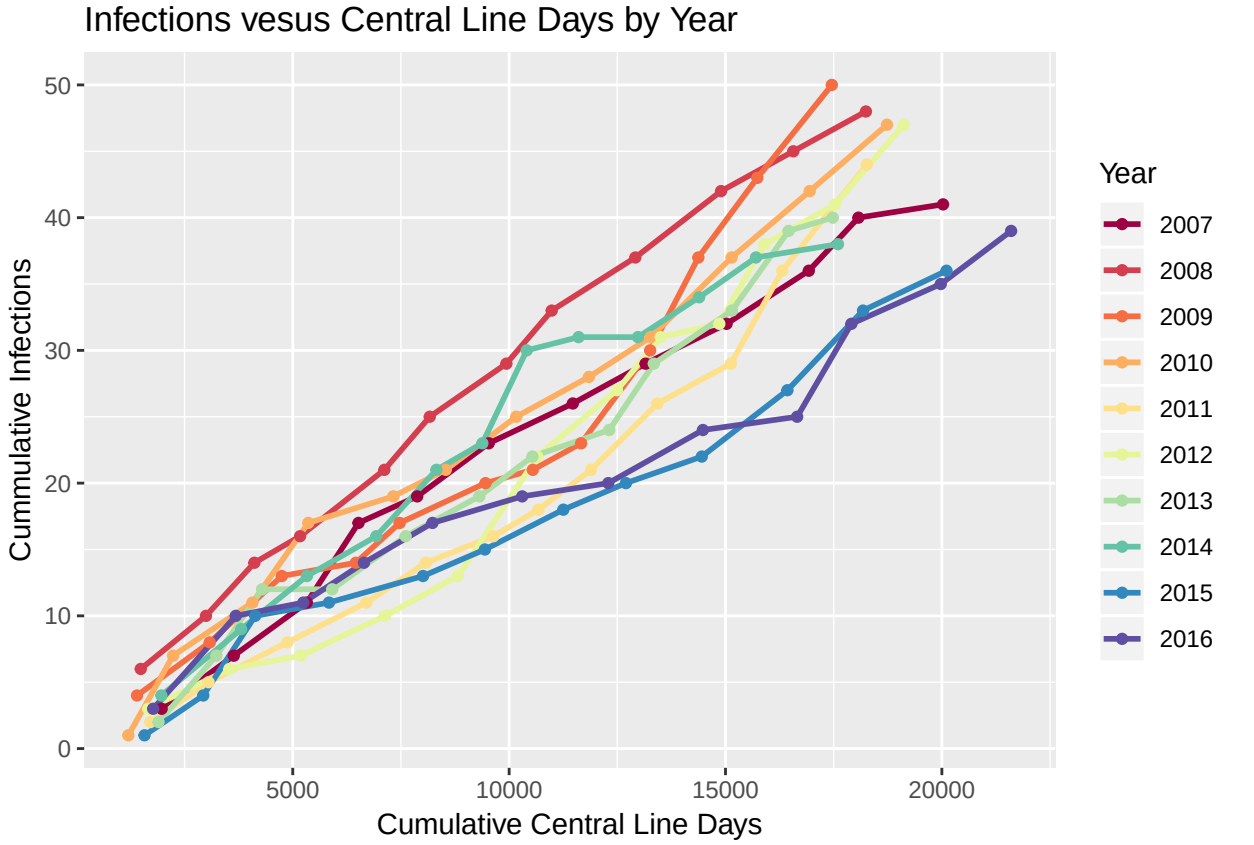
First, calculate the cumulative sums for the numerator and denominator for the time period of interest. Here, we use years.

```
# Calculate cumulative sums by year
accum_bsi_df = bsi_10yr %>%
  group_by(Year) %>%
  arrange(Month) %>%
  mutate(cuml_linedays = cumsum(Linedays), cuml_infections = cumsum(Infections))
```

Then, plot them against each other. Much like a seasonplot, a spaghetti "pattern" indicates that only

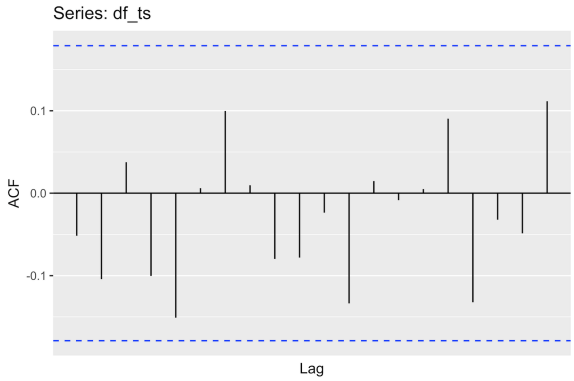
random, common cause variation is acting on the variables. Strands (individual years) that separate from that mess of lines suggest that a different process is in place for those strands.

```
# Accumulation plot
ggplot(accum_bsi_df, aes(x = cuml_linedays, y = cuml_infections, group = as.factor(Year))) +
  geom_path(aes(color = as.factor(Year)), size = 1) +
  geom_point(aes(color = as.factor(Year)))+
  scale_y_continuous(name = "Cummulative Infections", breaks = seq(0,120,10)) +
  scale_x_continuous(name = "Cumulative Central Line Days", breaks = seq(0,40000,5000)) +
  scale_colour_brewer(type = "div", palette = "Spectral") +
  guides(color = guide_legend(title = "Year")) +
  ggtitle("Infections vesus Central Line Days by Year")
```



11.2 More on autocorrelation

For convenience of comparison, here are autocorrelated and non-autocorrelated data already shown above, shown here side-by-side.

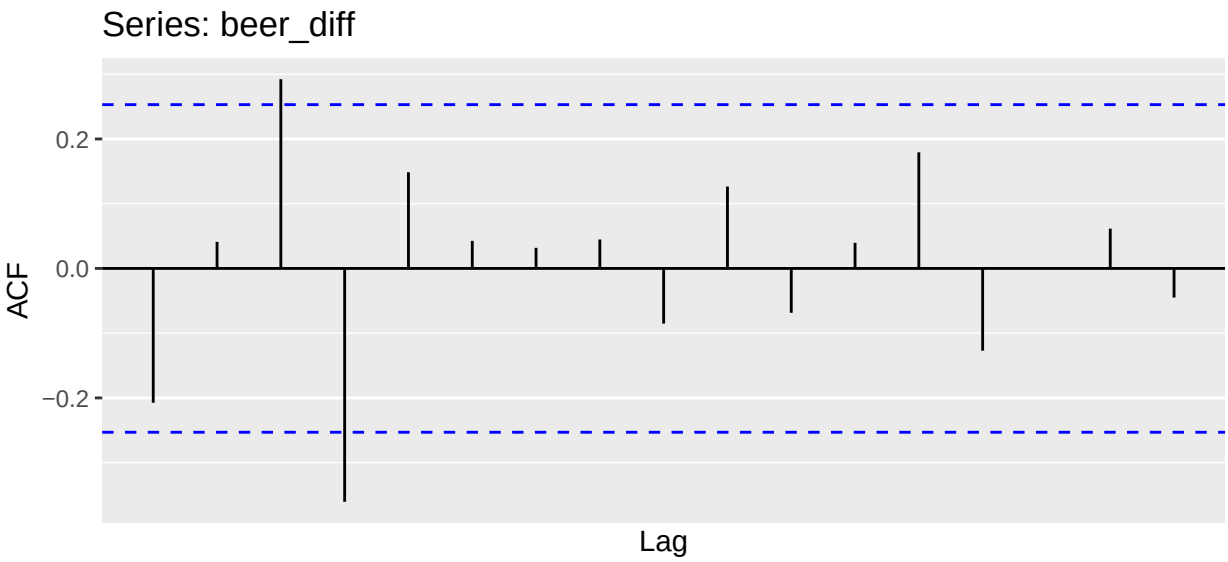
Example autocorrelated data	Example non-autocorrelated data
	

When data are autocorrelated, control limits will be *too small*—and thus an increase in *false* signals of special causes should be expected. In addition, none of the tests for special cause variation remain valid.

Sometimes, autocorrelation can be removed by changing the sampling or metric's time step: for example, you generally wouldn't expect hospital acquired infection rates in one quarter to influence those in the subsequent quarter.

It can also be sometimes removed or abated with differencing, although doing so hurts interpretability of the resulting run or control chart.

```
# Take the fourth lag to difference the beer data
beer_diff = diff(beer, lag = 4)

# Plot the resulting autocorrelation function
autoplot(acf(beer_diff, plot = FALSE))
```



If have autocorrelated data, and you aren't willing to difference the data or can't change the sampling rate or time step, you shouldn't use either run or control charts, and instead use a standard line chart. If you must have limits to help guide decision-making, you'll need a more advanced technique, such as a Generalized Additive Mixed Model (GAMM) or time series models such as ARIMA. It's probably best to work with a statistician if you need to do this.

Chapter 12

I need a shortcut

I'M NOT SURE WHAT THIS SECTION IS SUPPOSED TO BE DOING.

We've created a function that highlights points that may indicate special cause variation, as outlined in Chapter 4.

Using this function does require some thought about setting up the variables, which was done on purpose—you should put as much care into the construction of run and control charts as your nurses put into patient care. To do any less is a disservice to the decision-makers who would rely on your work and the patients that rely on these decision-makers to provide the conditions that support the best care possible.

We also know there are realities to timely decision-making, and sometimes you need a chart now. R does have several packages that provide ready-to-use SPC charts; we've shown examples using `qic` in earlier portions of this book, and we think it does a great job providing the basic needs for quick-and-dirty SPC chart making, particularly with its built-in run rules option. One potential trade-off is that you are stuck with the traditional 3σ control limits.

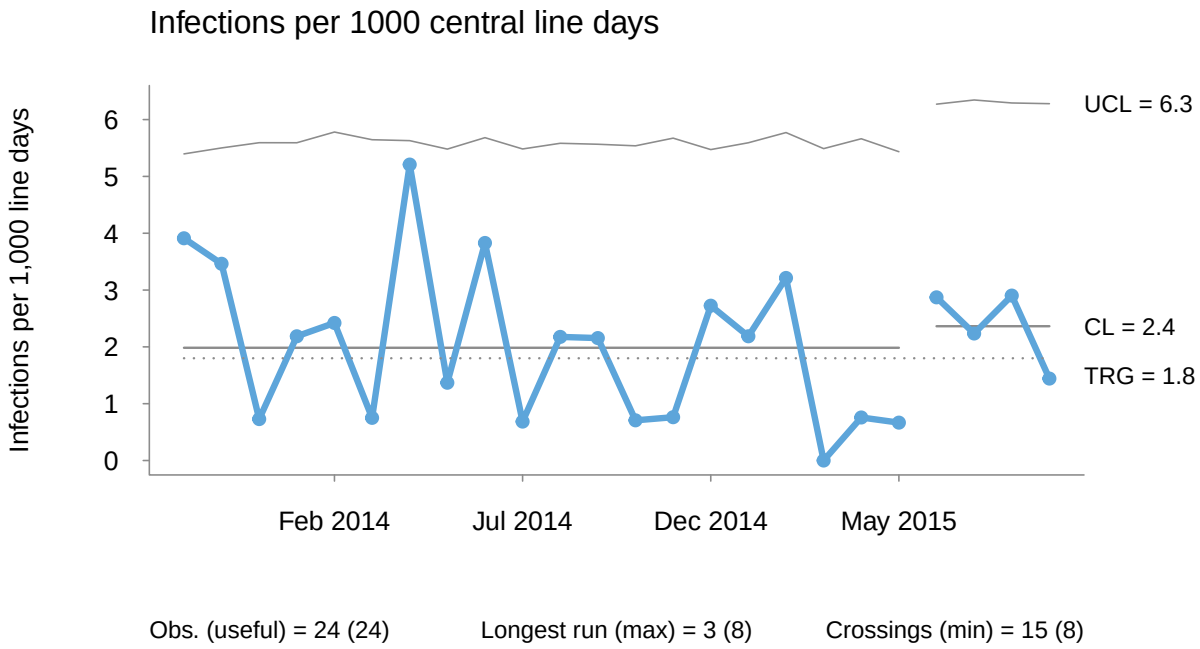
So, if you *always* start with EDA (Chapter 3), and *always* consider the appropriateness of SPC chart use for the given problem (Chapter 4 and [Chapter 5(#which)), and are fine with their built-in assumptions, go ahead and use these packages if they help you do your job better.

We'll use the data from Chapter 8's u-chart example.

```
# this is the same data as used in Chapter 8 for the u chart example
clabsi = data.frame(Month = seq(as.Date("2013/10/1"), by = "month", length.out = 24),
                    Infections = infections.agg, Linedays = linedays.agg)
```

Let's say an intervention was implemented at the end of 20 months of data to meet a target of 1.8 infections per 1,000 line days, and in a hurry to understand whether it works, managers wanted to see a control chart 4 months after the intervention. Clearly the process hasn't changed, and there is no evidence of special cause variation. Many managers might point to the increase in mean; a short-cut toward preventing them from acting on this knee-jerk (but common) reaction is to simply say "That change is not significant."

```
qicharts::qic(y = Infections, n = Linedays, data = clabsi, x = Month, multiply = 1000,
    chart = "u", runvals = TRUE, xlab = "", ylab = "Infections per 1,000 line days",
    x.format = "%b %Y", target = 0.0018, breaks = 20,
    main = "Infections per 1000 central line days")
```


Infections per 1000 central line days
Infections per 1,000 line days
6
5
4
3
2
1
0
Feb 2014
Jul 2014
Dec 2014
May 2015
UCL = 6.3
CL = 2.4
TRG = 1.8
Obs. (useful) = 24 (24)
Longest run (max) = 3 (8)
Crossings (min) = 15 (8)

Chapter 13

Useful References

- For more information, a good overview of run charts can be found in Perla et al. 2011, *The run chart: a simple analytical tool for learning from variation in healthcare processes*, BMJ Quality & Safety 20:46-51.
- A straight-to-the-point reference/tool for doing run charts in R is Anhøj 2016, *Run charts with R.*
- Some good overview papers on control charts include Benneyan et al. 2003, *Statistical process control as a tool for research and healthcare improvement*, BMJ Quality & Safety 12:458-464, and Mohammed et al. 2008, *Plotting basic control charts: tutorial notes for healthcare practitioners*, BMJ Quality & Safety 17:137-145. Wheeler 2010 covers why you shouldn't use 3σ for control limits in I charts.
- A straight-to-the-point reference/tool for doing control charts in R is Anhøj 2016, *Control Charts with qicharts for R.*
- A good basic overview book is Carey and Lloyd 2001, *Measuring Quality Improvement in Healthcare*, American Society for Quality.
- A good book that covers both basic and advanced topics is Provost and Murray 2011, *The Health Care Data Guide*, Jossey-Bass.
- The papers that discuss the uselessness of the trend test in run and control charts include Davis & Woodall 1988, *Performance of the control chart trend rule under linear shift*, Journal of Quality Technology 20:260-262, and Anhøj & Olesen 2014, *Run charts revisited: A simulation study of run chart rules for detection of non-random variation in health care processes*, PLOS One 9(11): e113825.
- Finally, some important warnings about when control charts fail (and a useful alternative, GAMs) can be found in Morton et al. 2009, *Hospital adverse events and control charts: the need for a new paradigm*, Journal of Hospital Infection 73(3):225–231, as well as in Morton et al. 2007, *New control chart methods for monitoring MROs in Hospitals*, Australian Infection Control 12(1):14-18.
- Wikipedia is a good place to start learning about probability distributions and their mean-variance relationships, e.g., (click the name to go to the link):
  - Poisson
  - binomial
  - normal
  - geometric
  - Weibull
  - A gallery of distributions (NIST)
  - Common probability distributions: the data scientist's crib sheet (Cloudera)

Chapter 14

SPC plots with ggspc

```
spc.plot = function(subgroup, point, mean, sigma, k = 3,
                    ucl.show = TRUE, lcl.show = TRUE,
                    band.show = TRUE, rule.show = TRUE,
                    ucl.max = Inf, lcl.min = -Inf,
                    label.x = "Subgroup", label.y = "Value")
{
    # Plots control chart with ggplot
    #
    # Args:
    #   subgroup: Subgroup definition (for x-axis)
    #   point: Subgroup sample values (for y-axis)
    #   mean: Process mean value (for center line)
    #   sigma: Process variation value (for control limits)
    #   k: Specification for k-sigma limits above and below center line.
    #      Default is 3.
    #   ucl.show: Visible upper control limit? Default is true.
    #   lcl.show: Visible lower control limit? Default is true.
    #   band.show: Visible bands between 1-2 sigma limits?  Default is true.
    #   rule.show: Highlight run rule indicators in orange?  Default is true.
    #   ucl.max: Maximum feasible value for upper control limit.
    #   lcl.min: Minimum feasible value for lower control limit.
    #   label.x: Specify x-axis label.
    #   label.y: Specify y-axis label.


    df = data.frame(subgroup, point)
    df$ucl = pmin(ucl.max, mean + k*sigma)
    df$lcl = pmax(lcl.min, mean - k*sigma)

    warn.points = function(rule, num, den) {
        sets = mapply(seq, 1:(length(subgroup) - (den - 1)),
                      den:length(subgroup))
        hits = apply(sets, 2, function(x) sum(rule[x])) >= num
        intersect(c(sets[,hits]), which(rule))
    }
    orange.sigma = numeric()
```

```
    p = ggplot(data = df, aes(x = subgroup)) +
        geom_hline(yintercept = mean, col = "gray", size = 1)

    if (ucl.show) {
        p = p + geom_line(aes(y = ucl), col = "gray", size = 1)
    }

    if (lcl.show) {
        p = p + geom_line(aes(y = lcl), col = "gray", size = 1)
    }

    if (band.show) {
        p = p +
            geom_ribbon(aes(ymin = mean + sigma,
                            ymax = mean + 2*sigma), alpha = 0.1) +
            geom_ribbon(aes(ymin = pmax(lcl.min, mean - 2*sigma),
                            ymax = mean - sigma), alpha = 0.1)

        orange.sigma = unique(c(
            warn.points(point > mean + sigma, 4, 5),
            warn.points(point < mean - sigma, 4, 5),
            warn.points(point > mean + 2*sigma, 2, 3),
            warn.points(point < mean - 2*sigma, 2, 3)
        ))
    }

    df$warn = "blue"
    if (rule.show) {
        shift.n = round(log(sum(point!=mean), 2) + 3)
        orange = unique(c(orange.sigma,
        warn.points(point > mean - sigma & point < mean + sigma, 15, 15),
        warn.points(point > mean, shift.n, shift.n),
        warn.points(point < mean, shift.n, shift.n)))
        df$warn[orange] = "orange"
    }
    df$warn[point > df$ucl | point < df$lcl] = "red"

    p +
        geom_line(aes(y = point), col = "royalblue3") +
        geom_point(data = df, aes(x = subgroup, y = point, col = warn)) +
        scale_color_manual(values = c("blue" = "royalblue3", "orange" = "orangered", "red" = "red3"), gu
        labs(x = label.x, y = label.y) +
        theme_bw()
}
```