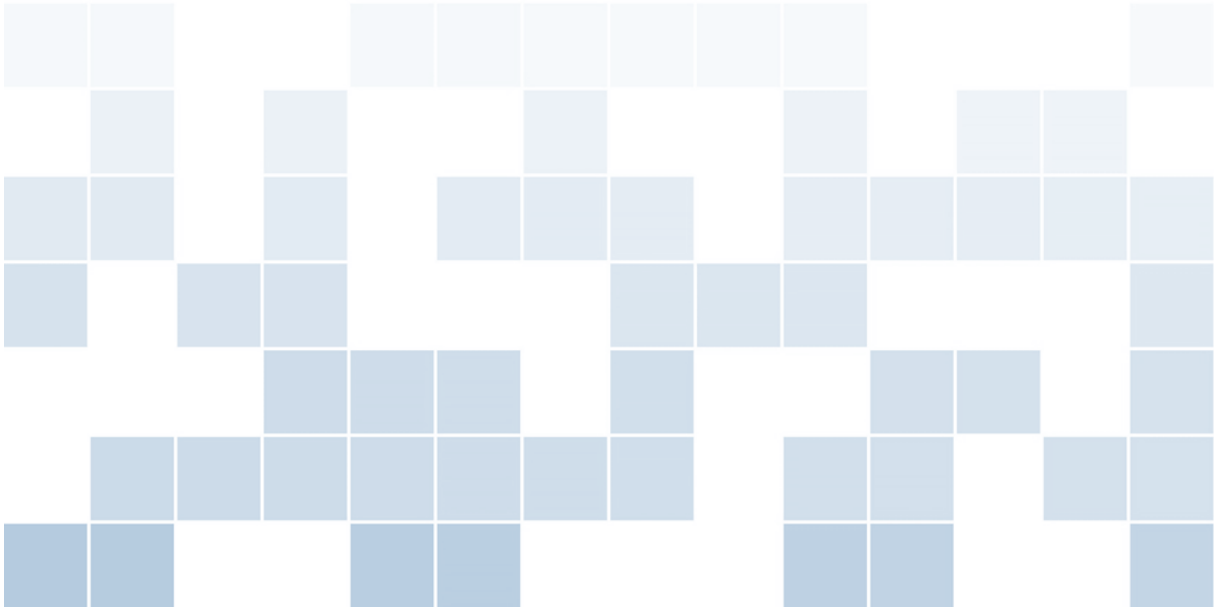
Statistical Process Control in Healthcare with R

July 2019

Dwight Barry, Brendan Bettinger, Andrew Cooper, and Sydney Paul



Contents

Preface 5

0.1 We have a problem **5**

0.2 Common questions **5**

Who is this book for? 5

What if I have never used R? 5

What do I need to start? 6

0.3 About **7**

Who are we? 7

What if I find a typo? 7

I **Part I**

1 Common Mistakes in SPC 9

1.1 Mistake #1 **9**

Not using all the information that you have available to you 9

1.2 Mistake #2 **10**

Forgetting that SPC charts cannot make decisions for you, they can only help *you* to make a decision 10

1.3 Mistake #3 **12**

Skipping to the end of the process 12

1.4 **Mistake #4** 12
Not understanding what it means to have a "stable" process 12

II **Part II**

2 Case Study 15

3 Exploratory Data Analysis 17

4 Checking Assumptions 21
4.1 **Trending** 21
4.2 **Independence / Autocorrelation** 22

5 Run Chart 27

6 Control Chart 30
6.1 **What is the underlying distribution?** 30
6.2 **Which control chart should I use?** 35

III **Part III**

7 Run Charts vs. Control Charts 40
7.1 **Which should I use: a run chart or a control chart?** 40
7.2 **Guidelines for interpreting SPC charts** 41

8 A Guide to Control Charts 44
8.1 **Types of Control Charts** 44
8.1.1 *u*-chart example 45
8.1.2 *p*-chart example 46
8.1.3 *g*-chart example 47
8.1.4 *c*- and *np*-chart details 48
8.1.5 *I-MR* chart 48
8.1.6 $\bar{x}$ and *s* chart 50
8.1.7 *t*-chart example 51
8.2 **Tips and tricks for successful control chart use** 52
8.2.1 When to revise control limits 53
8.3 **Custom SPC function** 56
8.3.1 EWMA chart 56
8.3.2 CUSUM chart 57

9 Additional Resources 60
9.1 Time series EDA 60
9.1.1 Trend 60
9.1.2 Seasonplot 61
9.1.3 Monthplot 62
9.1.4 Autocorrelation 62
9.1.5 Cycles 64
9.1.6 Decomposition 65
9.1.7 Seasonal adjustment 66
9.1.8 Residuals 67
9.1.9 Accumuluation plots 68
9.2 Custom SPC Function 70
9.3 Code used to generate examples 74
9.4 Useful References 77

Preface

0.1 We have a problem

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.

0.2 Common questions

Who is this book for?

This book is geared toward analysts working in the healthcare industry, who are already 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; The section Useful References in the Additional Resources chapter provides a great place to start or continue learning about SPC.

What if I have never used R?

We don't presume familiarity with R, but 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/`.

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 idiosyncrasies and outright annoyances, but when you need to accomplish particular goals, it can be fairly easy!

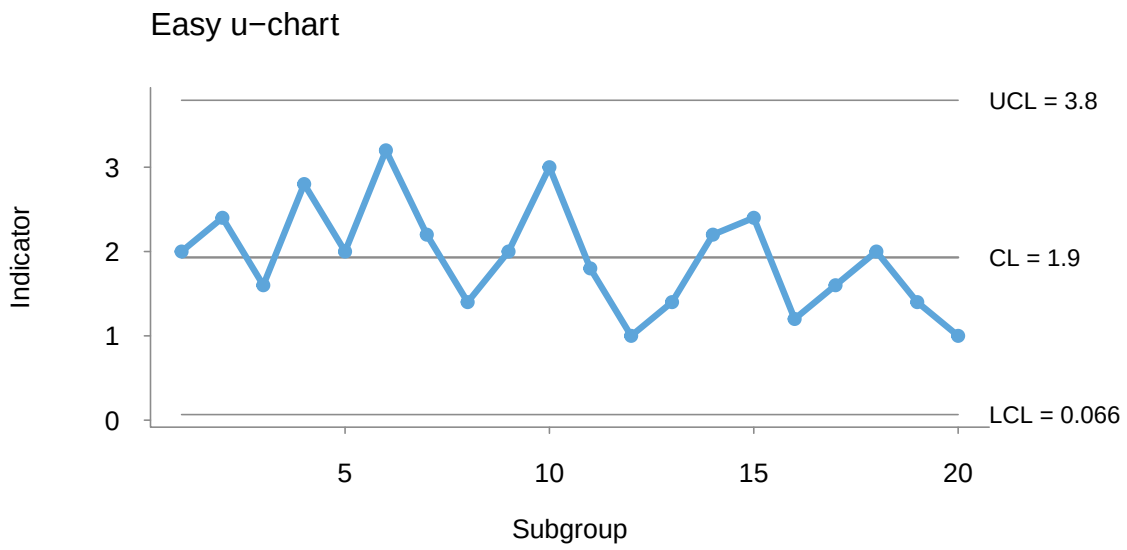
For example, you can create a *u*-chart with only three lines of code!

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

# 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")
```



Another useful feature of R: you can ask for help in R using `?`, followed by the function name, ex. `?qic`

Between the straightforward functions and the help documentation, you'll be coding in R in no time!

What do I need to start?

First, you'll need R and RStudio, which can be downloaded from the links above.

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

```
install.packages("ggplot2", "forecast", "lubridate", "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).

0.3 About

Who are we?

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

- Dwight Barry is a Principal Data Scientist in *Enterprise Analytics*. Twitter: @healthstats-dude
- Brendan Bettinger is a Senior Analyst in *Infection Prevention*.
- Andy Cooper is a Lead Data Scientist in *Enterprise Analytics*. Twitter: @DataSciABC
- Sydney Paul is a Data Science Intern in *Enterprise Analytics*.

What if I find a typo?

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

Part I

1 Common Mistakes in SPC 9
1.1 Mistake #1
1.2 Mistake #2
1.3 Mistake #3
1.4 Mistake #4

1. Common Mistakes in SPC

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.

1.1 Mistake #1

Not using all the information that you have available to you

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 information of a run chart or control chart to understand what it's telling you.

1.2 Mistake #2

Forgetting that SPC charts cannot make decisions for you, they can only help *you* to make a decision

This example illustrates why it is important to think about your data and not blindly listening to the guidelines.

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.

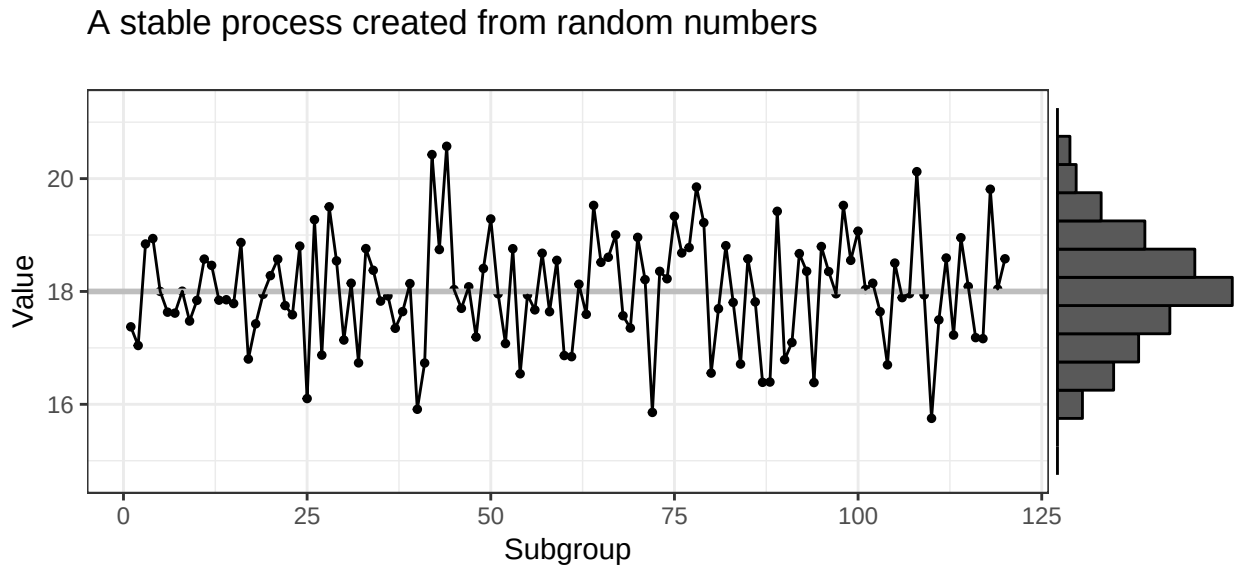


Figure 1: The axis labels are traditional SPC labels. The *value* (i.e., metric) is on the y-axis and the units of observation, traditionally called *subgroups*, are on the x-axis.

The term subgroup was developed in the context of an observation point involved sampling from a mechanical 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.

Another feature of SPC control charts are control limits. The symbol σ is a measure of expected process standard deviation. Figure 2 adds control limits which are at $\pm\ 3\sigma$.

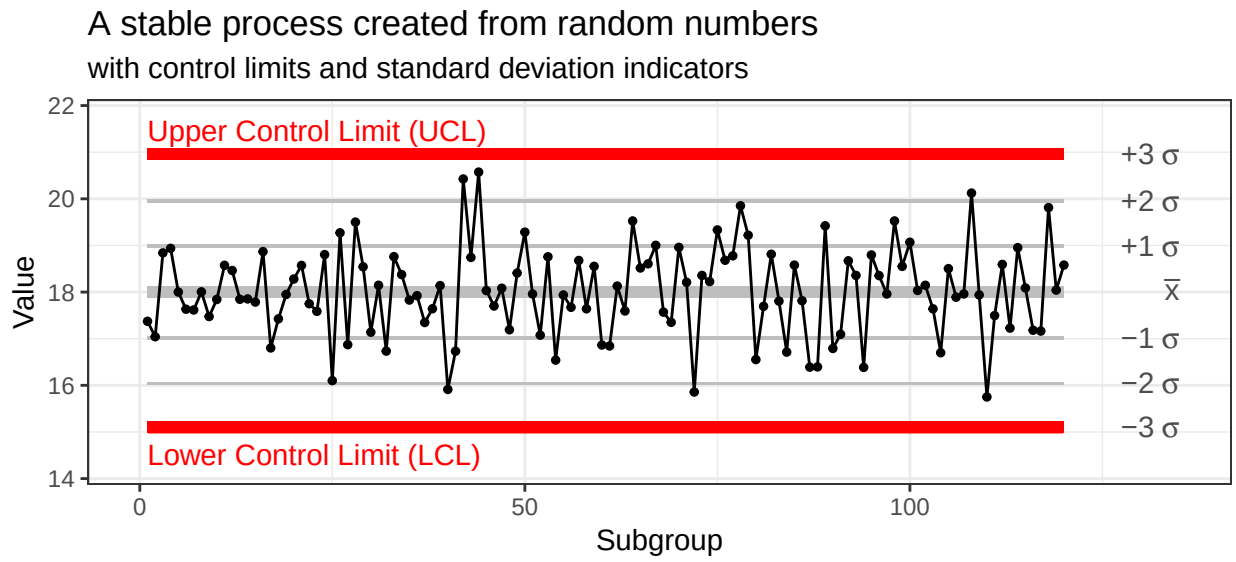


Figure 2: The same plot as in Figure 1, with standard deviation indicators and control limits added.

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 the section labeled Guidelines for interpreting SPC charts in the chapter Run Charts vs. Control Charts.

Applying the guidelines to the chart in Figure 2 reveals 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*
SPC: No alert	*False Negative*	True Negative

Using $\pm 3\sigma$ 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 $\pm 2\sigma$ control limits.

For other metrics where you prefer to be completely certain things are out of whack before

taking action — and are willing to accept more *False Negatives* you to reduce *False Positives* — you could use $\pm 4\sigma$ control limits.

When in doubt, use $\pm 3\sigma$ 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.

1.3 Mistake #3

Skipping to the end of the process

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.

1.4 Mistake #4

Not understanding what it means to have a "stable" process

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. Let's look at two examples to get a more concrete understanding:

Example 1: Some outpatient specialties are facing increasing numbers of referrals, but are not getting more staff to handle them. With increasing patient demand and constrained hospital capacity, we would not expect the 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.

Example 2: Complaints about phone wait time for a call center has led to an increase in full-time employees 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, thus a control chart is not appropriate. So, what would a "stable" KPI look like?

- 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 full-time employees (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.

Part II

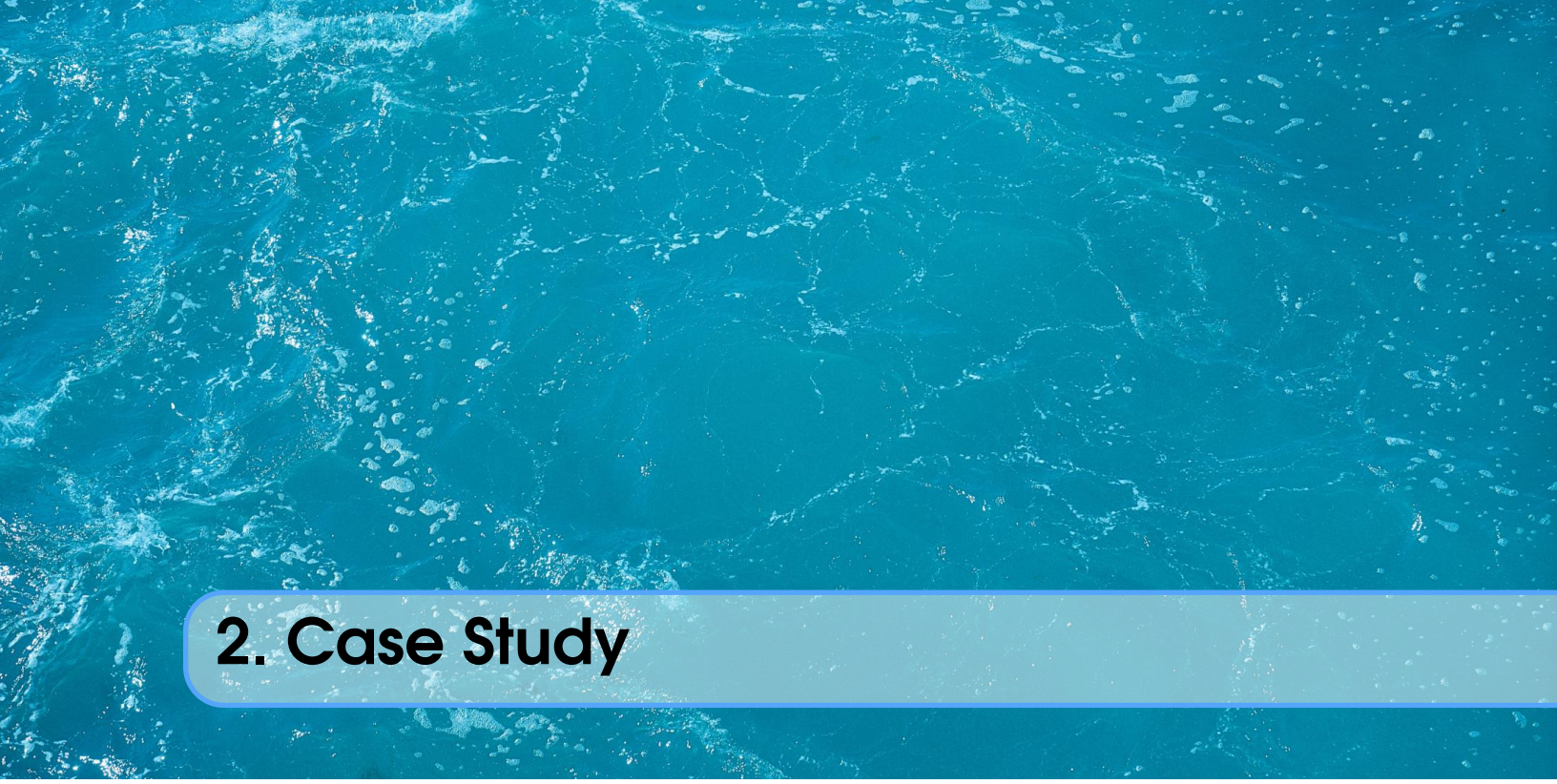
2 Case Study 15

3 Exploratory Data Analysis 17

4 Checking Assumptions 21
4.1 Trending
4.2 Independence / Autocorrelation

5 Run Chart 27

6 Control Chart 30
6.1 What is the underlying distribution?
6.2 Which control chart should I use?



2. Case Study

Rachel is a completely fictional character created for this book. She is a business analyst working in the healthcare industry. Over the next few chapters we will follow Rachel through the SPC process using her datasets (also fictional; the code used to create the datasets are available in the appendix).

Be on the look out for boxes like below. These are checkpoints that will help evaluate your understanding of the material. Let's try one now. Recall what you read about stability in the previous chapter to answer the following questions:

- Scenario 1: Rachel's company has recently implemented a new procedure to reduce the amount of time that patients wait to seen. Her superiors want to know if the change has been effective.
- Scenario 2: Rachel's company is proud that they have the shortest wait times in the state. They want Rachel to monitor patient wait-times to ensure that they do not slip.

Are these stable processes? Can Rachel use control charts?

■

SPC tools are built upon the fundamental assumption of a *stable* process.

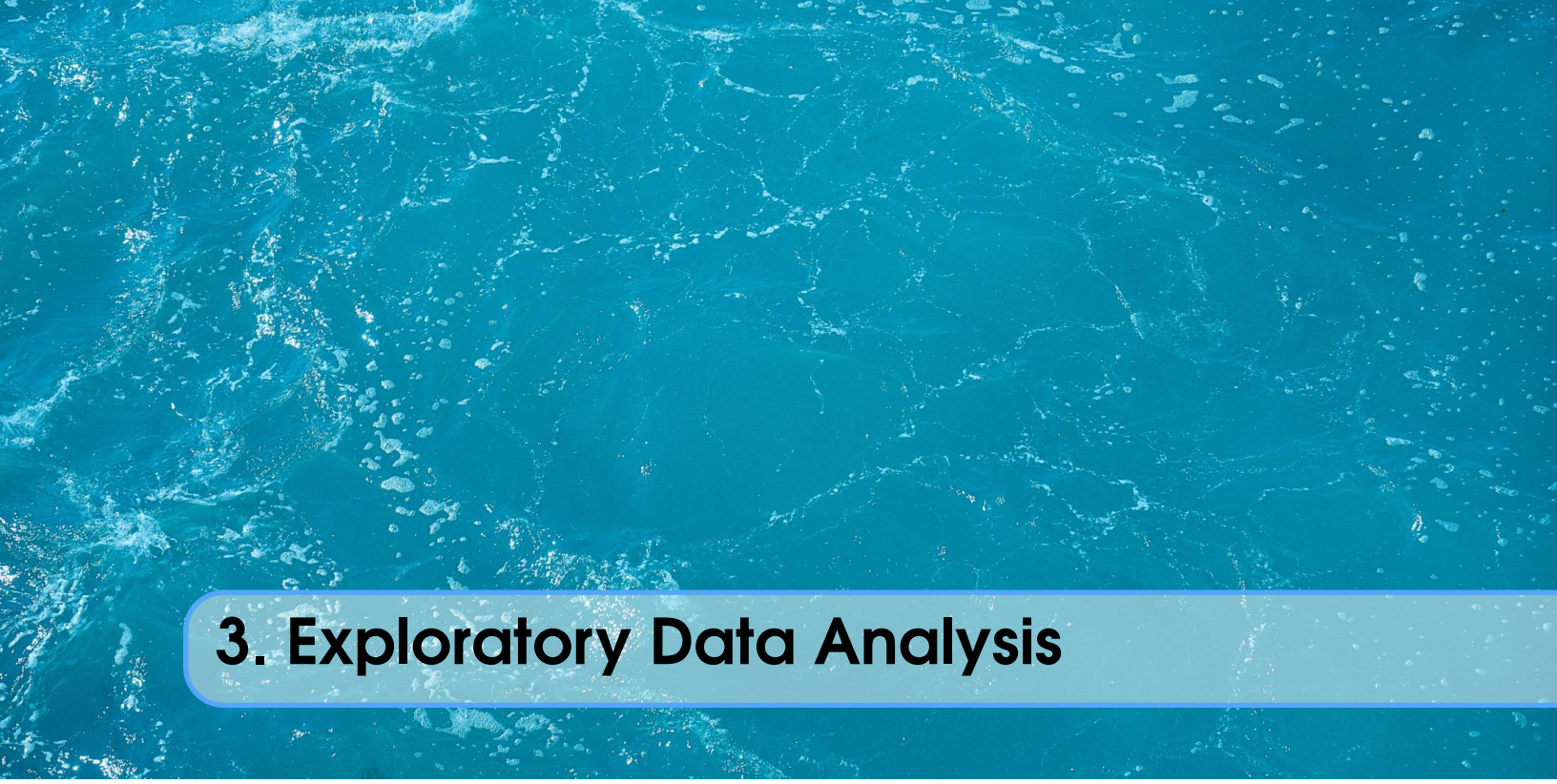
- Scenario 1: Is this a stable process? No, because we expect that the wait time will decrease. Control charts are only applicable when the business goal is to keep that KPI stable. Rachel must define the KPI in terms of something we would want to remain stable in order to proceed with the SPC process. This could be a frequency count of wait times over 45 minutes.
- Scenario 2: Is this a stable process? Yes, because we know that wait times have been stable and we expect them to remain stable.

Ensuring that the KPI is stable is foundational. Rachel cannot simply graph the control chart at this point (see Mistake #3 in the previous chapter). You must go through each step in the SPC process:

Rachel has a dataset of CLABSI (central line associated blood stream infection) data for her hospital, which can be seen below.

months	infections	linedays
2013-10-01	3	1361
2013-11-01	3	1273
2013-12-01	1	1392
...	...	...
2015-07-01	1	1335
2015-08-01	3	1441
2015-09-01	4	1305

Rachel and her team want to ensure that the CLABSI count is stable. In the next chapter we will help Rachel with exploratory data analysis for this dataset.



3. Exploratory Data Analysis

It is important to understand your data. Your data is the foundation for all further analysis. You cannot create any meaningful interpretation from bad data, and not all data is suited for SPC charts. There are many tools for data exploration, and you get to decide how deep to explore. Before you start blindly coding, its important to think about your data.

Take a minute to answer the following questions:

1. What are the typical values of your data, i.e. what do you expect the range of the data to be?
2. What do you think the distribution will look like? Will it be skewed? Will there be a lot of variance?

Our recommendation is to plot your data as a line chart and a histogram (adding a density overlay provides a more "objective" sense of the distribution). In these plots, consider:

1. The shape of the distribution: symmetrical/skewed, uniform/peaked/multimodal, whether changes in binwidth show patterning, etc.

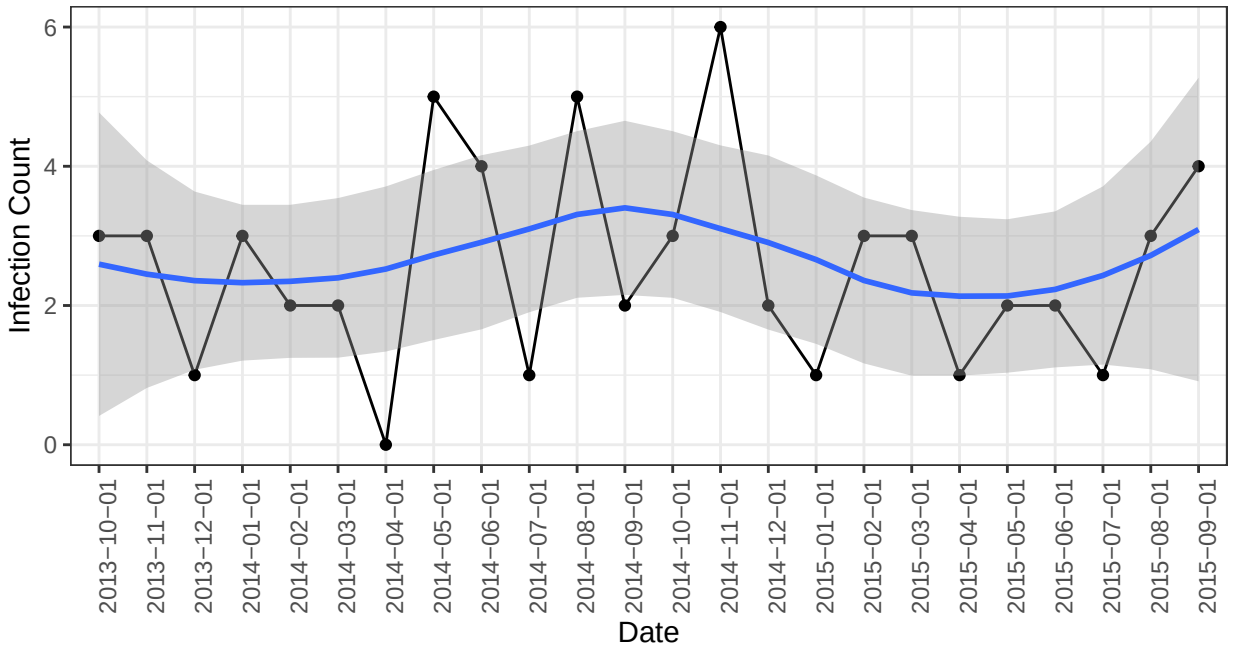
2. Whether you see any trending, cycles, or suggestions of autocorrelation (we will discuss this more in the next step).

3. Whether there are any obvious outliers or inliers—basically, any points deviating from the expected pattern.

R provides a lot of flexibility in plotting your data. This is certainly not the only way, but the code below is a good template for new R users. In order to use this for your data, you will want to replace the `data`, `x`, and `y` arguments and change the x and y labels using `labs`.

Now let's explore Rachel's dataset. First, let's plot the line chart.

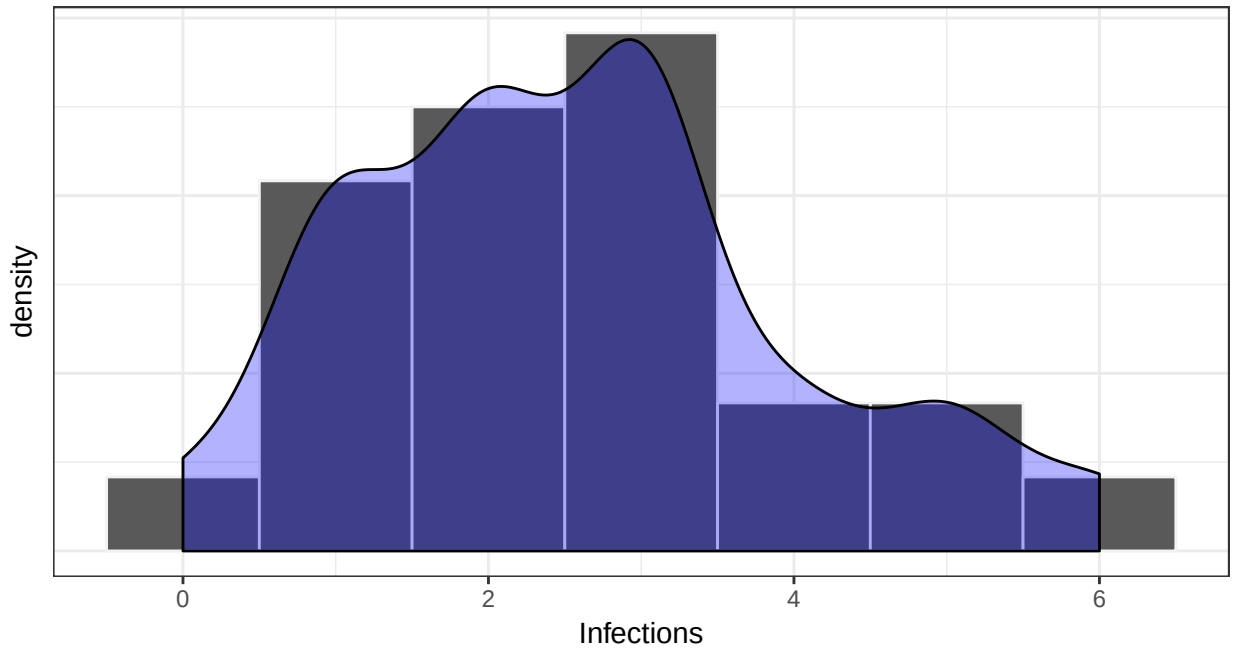
```
# all the data belongs to one group (default creates as many groups as
# observations)
ggplot(rachel_data, aes(x = months, y = infections, group = 1)) +
        geom_point() +
        geom_line() +
        geom_smooth(method = 'loess', formula = y ~ x) +
        theme_bw() +
        theme(axis.text.x = element_text(angle = 90)) +
        labs(x = 'Date', y = 'Infection Count')
```



The black points and line are simply the number of infections plotted over time. The blue trend line and grey shaded area can confuse people. The blue line is the trend line, and the shaded grey area is the confidence interval of the trend line. We can say with 95% confidence that the true, *actual* trend line falls within this grey area. The grey area is *not* a control limit. Remember this is a line chart, not a SPC chart.

Now onto plotting the histogram.

```
ggplot(rachel_data, aes(x = infections)) +
  geom_histogram(aes(y = ..density..), color = "gray95", binwidth = 1) +
  geom_density(fill = "blue", alpha = 0.3) +
  theme_bw() +
  theme(axis.text.y = element_blank(), axis.ticks.y = element_blank()) +
  labs(x = 'Infections')
```



A histogram is an excellent tool for examining the distribution of the data. In R, there are two key arguments that you need to change to explore your data: `binwidth` ***or*** `bins`. The graph above uses a bindwith equal to 1. We have discrete data, so each number gets its own bin. This is nice for this dataset because we only have seven numbers. In this case `binwidth = 1` is the same as `bins = 7`. For larger datasets, you might need to adjust the `bins` argument. R uses a default value of `bins = 30`. This parameter is completely user dependent. It is up to you to change this parameter until *you* think you have a good understanding of the distribution.

Now we refer back to the questions for evaluating these two plots.

1. The shape of the distribution: symmetrical/skewed, uniform/peaked/multimodal, whether changes in binwidth show patterning, etc. The distribution seems mildly skewed to the right. The distribution seems slightly multimodal. Changes in binwidth do not show patterning. In general, you might want evaluation on skew, center, and variation. The distribution is centered around 3, and ranges from 0 to 6.

2. Whether you see any trending, cycles, or suggestions of autocorrelation (we will discuss this more in the next step). It does not appear to be trending up or down. There might be a cycle, not isn't consistent. We will discuss cycles, trends, and autocorrelation in more detail later in the book.

3. Whether there are any obvious outliers or inliers—basically, any points deviating from the expected pattern. There is only one value at six, which could be an outlier, but there are no extreme

values, and no points deviating from the expected pattern. Note that even if we suspect that point to be an outlier, it is still part of our data. We can look for an explanation for it, but we cannot remove it. We acknowledge its existence now, and remember it if it comes up during later analysis.

Now we will move onto the second step, checking your assumptions.

4. Checking Assumptions

There are three main assumptions that must be met for a SPC chart to be meaningful.

1. We assume that the data does not contain a trend.
2. We assume that the data is independent.
3. We assume that the data is not autocorrelated.

4.1 Trending

In the previous step, you already completed a trend test: you looked at the line chart and decided if it was trending or not. You can tell by eye: does it look like it's trending over a large span of the time series? If so, then it probably is trending.

The Mann-Kendall trend test is often used as well. It is a non-parametric test that can determine whether the series contains a monotonic trend, linear or not. The null hypothesis being tested is that the data does not contain a trend. We will implement this test in R below. A caveat is that when sample size is low (n < 20) this test is not useful/accurate.

Let's run the Mann-Kendall trend test on Rachel's data.

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

```
    Mann-Kendall trend test

data:  rachel_data$infections
z = -0.076596, n = 24, p-value = 0.9389
```

```
alternative hypothesis: true S is not equal to 0
sample estimates:
           S         varS          tau
  -4.0000000 1534.0000000   -0.0159455
```

The test gives us a p-value of 0.9389, thus we cannot reject the null hypothesis. We can conclude that the data does not contain a trend. This agrees with our visual test, but keep in mind that we still have a low sample size (n = 24).

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.** An alternative is to use a run chart with a median slope instead, e.g., via quantile regression. 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.

4.2 Independence / Autocorrelation

Independence and autocorrelation are two important, related terms. Independence generally means that the value of the data will not change due to other variables or previous data points, ex. rolling a fair die and flipping a coin. The value that the die lands on should not be affected by the coin flip nor the previous value of the die. Correlation is the tendency for one variable to increase or decrease as a different variable increases. Autocorrelation is a variable that correlates with itself lagged or leading in time, ex. if it rained yesterday, it will be more likely to rain today. If variables are independent, then they do not have any correlation.

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.

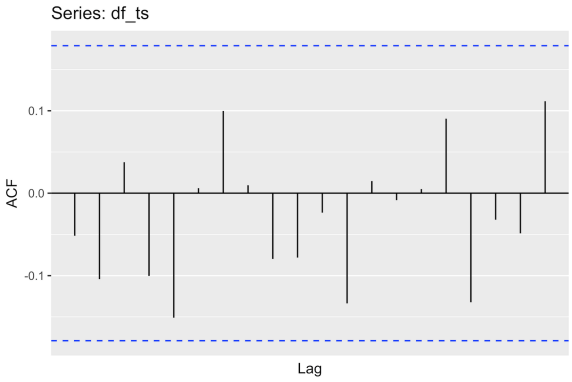
After considering the context, a second way to assess independence is by calculating the autocorrelation function (acf) for the time series.

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.

Autocorrelation values over 0.50 generally indicate problems, as do patterns in the autocorrelation function. 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.

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.

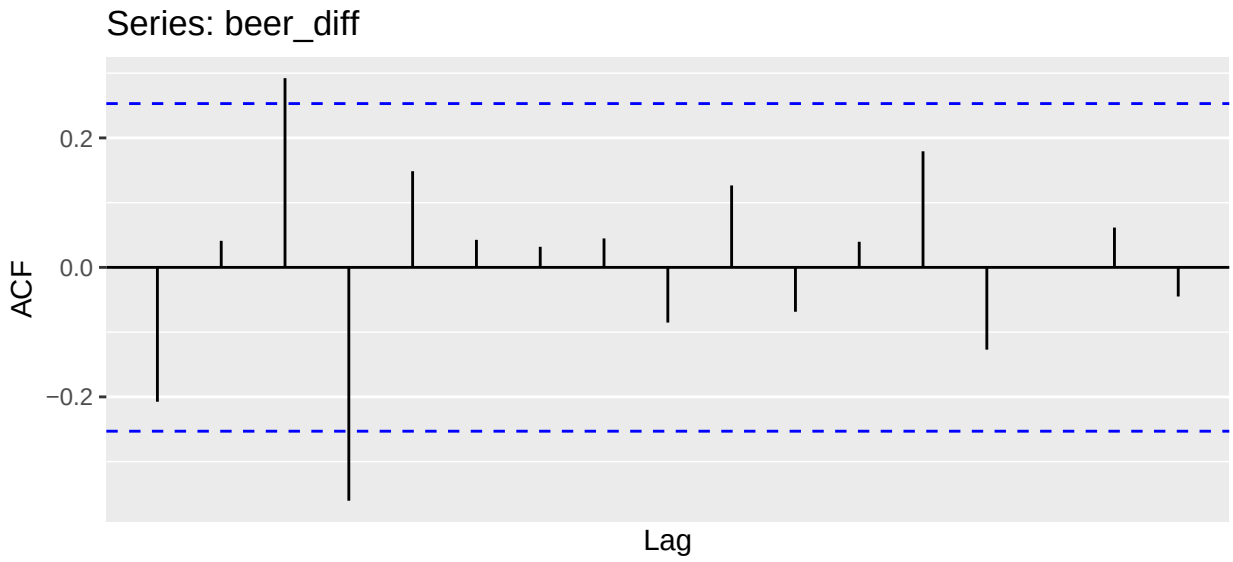
Sometimes, autocorrelation can be removed by changing the sampling or metric's time step: for example, you generally wouldn't expect hospital acquired infection rates in one quarter to influence those in the subsequent quarter.

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

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

# 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.

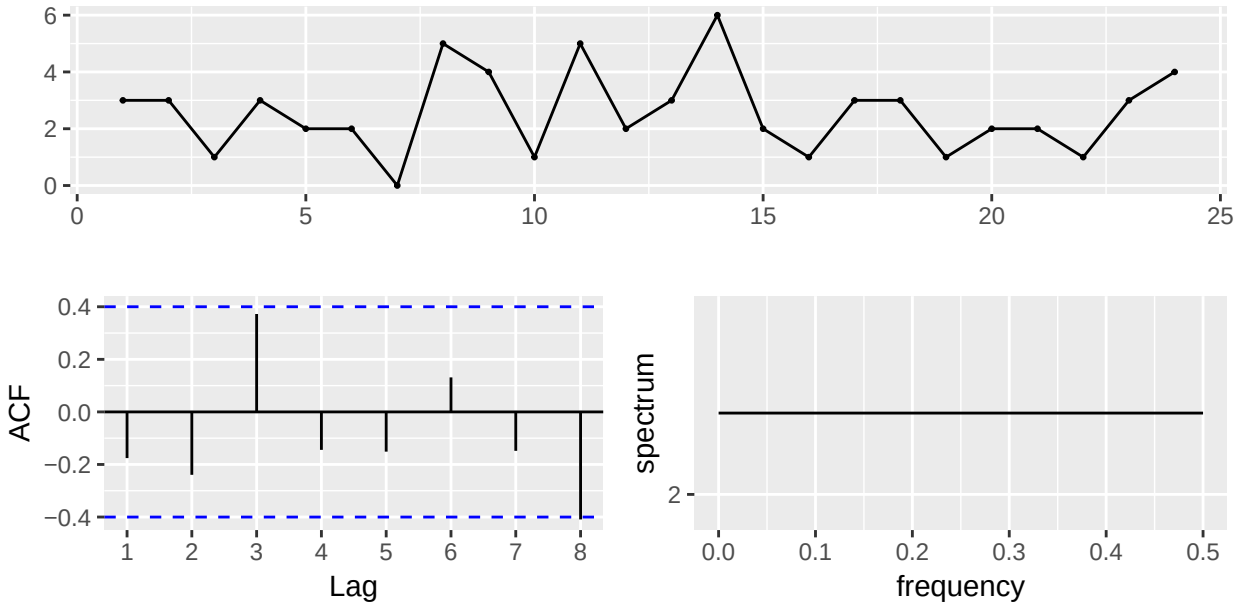
Now we will inspect Rachel's dataset for both independence and autocorrelation.

First we should consider the context. This is CLABSI (central line associated blood stream infection) data. Does the infection of one patient influence the infection of the next patient? Ideally, we would hope not, but infections can spread. Let's now see what the autocorrelation function (acf) has to say.

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).

```
# ggtsdisplay requires the input to be a time series
df_ts <- ts(rachel_data$infections)

# 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). This means that all three assumptions are valid for Rachel's data and she can move on to the next step, creating a run chart.

Before you move on, however, these are very important concepts.

Take a minute to think over the following questions:

1. Why is the Mann-Kendall trend test not always useful?
2. What is independence and why is it important?
3. How can you recognize autocorrelation in control charts?
4. What happens if you make conclusions from a control chart where these assumptions are not valid?

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.

If you still feel uncomfortable with these principles, check out the section Time Series in the Additional Resources chapter for more information and examples, including what these plots can look like when you have time-dependent or otherwise autocorrelated data.

"But what if I understand everything and my data doesn't upheld these assumptions?"

When you do have such data, you cannot use standard SPC tools. Generalized additive models (GAMs or GAMMs) can be useful alternatives; see the section Useful Resources in the Additional Resources chapter for some good initial references.

5. Run Chart

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 (~$\pm$ 1.5σ) changes in process than using the control charts' typical $\pm$ 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.

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

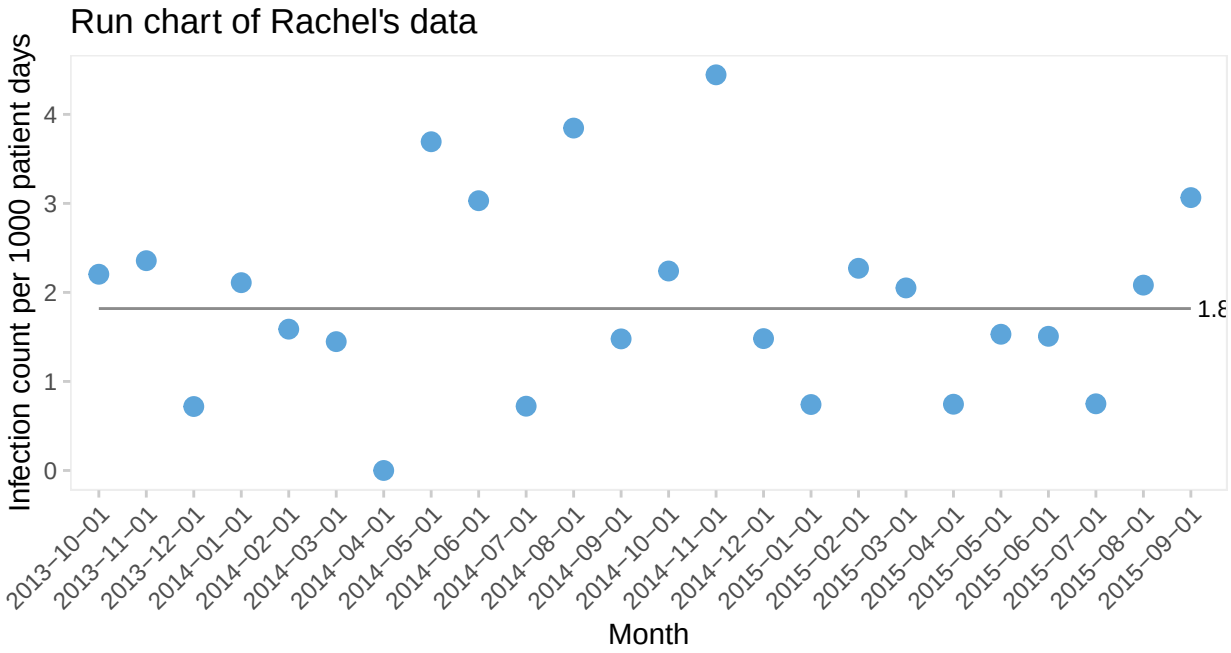
- *Process shift:* A non-random run is a set of $log_2(n)+3$ consecutive data points (rounded to the nearest integer) that are all above or all below the median line, where n is the number of points that do *not* fall directly on the median line. For example, if there are 34 points and 2 fall on the median, then $n = 32$ observations.Plugging this value into the equation: $log_2(32)+3 = 5+3 = 8$. So, 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.

However, this book uses a package called `qicharts2`. This package will create a run chart or control charts for you, and calculate relevant values for you. The best way to understand, is to jump into an example. Let's plot the run chart for Rachel's data. The arguments `data`, `y`, and `x` are straightforward. The `n` argument is only needed for certain applications. CLABSI data is one of those applications where `n` is the number of linedays (which is a column available in Rachel's dataset). The `multiply` argument refers to the number of patient days, in this case we want to look at per 1000 patient days. We encourage you to read through the documentation of the package (`?qicharts2::qic`) to determine which arguments are needed for *your* data.

The argument `print.summary = TRUE` can be added to the qic function *or* the summary function can be called on the plot object (`summary(run_chart)`). Both will give a table of calculated values that you can evaluate.

```
qicharts2::qic(x = months, y = infections, n = linedays, data = rachel_data,
               multiply = 1000, chart = 'run', x.angle = 45,
               title = "Run chart of Rachel's data",
               xlab = "Month",
               ylab = "Infection count per 1000 patient days",
               print.summary = TRUE)
```

```
  facet1 facet2 part n.obs n.useful longest.run longest.run.max
1      1      1    1    24       24           4               8
  n.crossings n.crossings.min runs.signal aLCL       CL aUCL sigma.signal
1          12               8           0  NaN 1.818941  NaN            0
```



By viewing the summary, you can check the number of observations, `n.obs`, to the number of observations useful, `n.useful`. You can ensure that the longest run in the data, `longest.run`, is less than the maximum allowed, `longest.max`. Finally you can ensure that the number of crossing, `n.crossings`, is greater than the minimum expected, `n.crossings.min`.

Looking at the run chart and the summary table for Rachel's data, is any non-random variation suggested?

No, there is not any non-random variation suggested. There are no unusual runs in the data. The longest run is four points when the maximum was eight. Furthermore, there are twelve crossings, when we needed at least eight. So both tests (process shift and number of crossings) suggest that there is no non-random variation in this process.

6. Control Chart

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 offers a great deal more power to find insights and monitor change than a line or run chart.

6.1 What is the underlying distribution?

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*.

There are hundreds of statistical 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

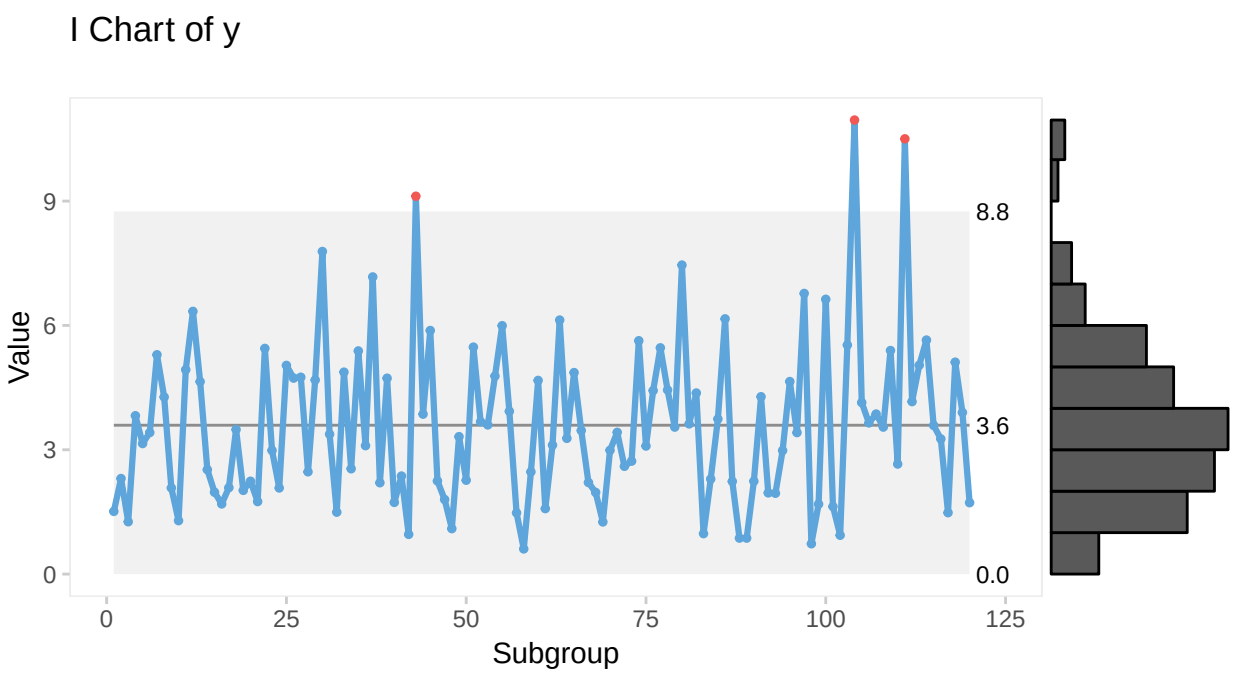
Data Type	Distribution	Range	Skew	Example	SPC chart
	Geometric	$0, \infty$	Right	Number of surgeries between complications	g
Continuous	Normal	$-\infty, \infty$	None	Patient wait times	I, $\bar{x}$, EWMA, CUSUM
	Weibull	$0, \infty$	Right	Time between antibiotic doses	t

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.

So, what happens when you get the mean-variance relationship wrong?

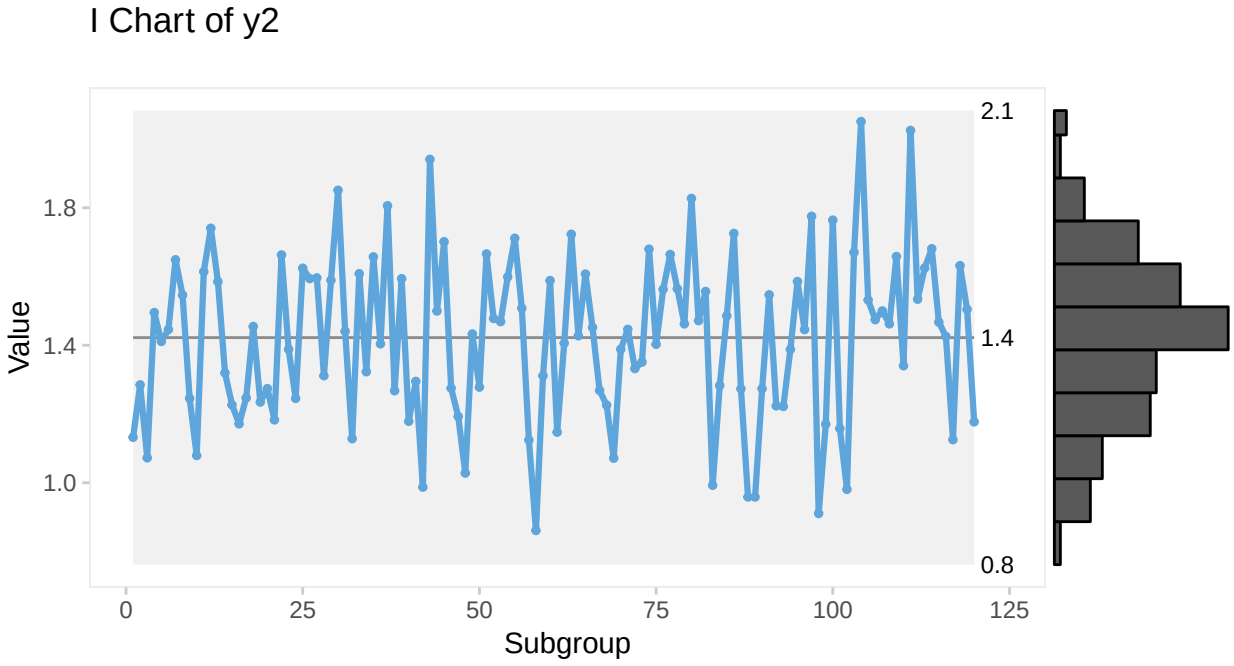
Although control charts are "robust" to some assumption violations and can sometimes work when the mean-variance relationship is incorrect, you won't know unless you explore the differences in implications between the data as-is and that same data transformed to become more in line with the appropriate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



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.



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?

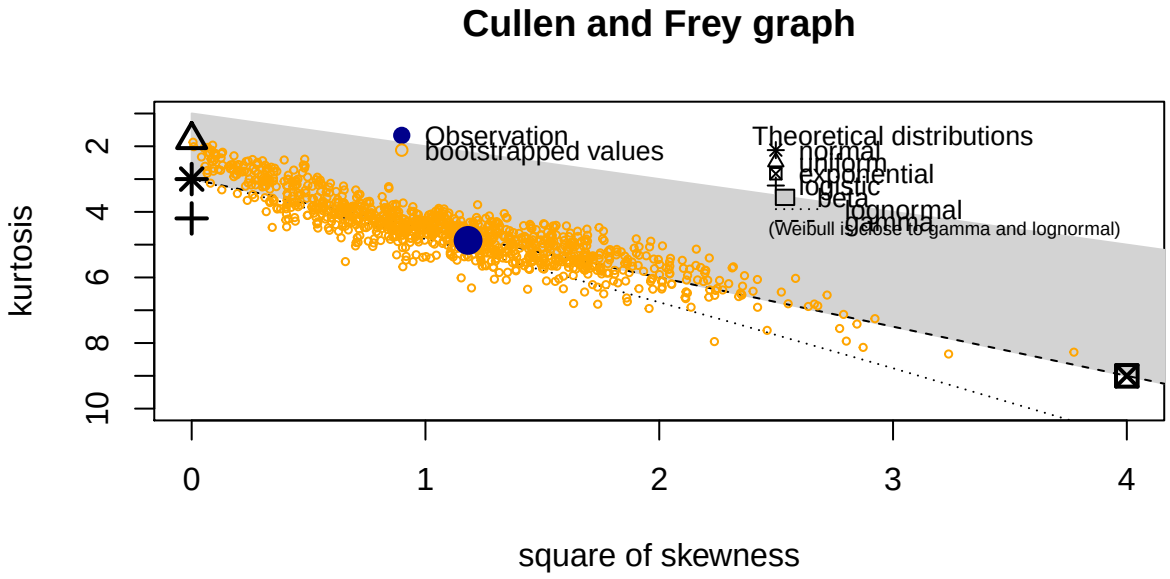
Not necessarily. For more information, see, for example, The arcsine is asinine and Do not log-transform count data.

Consult a statistician if you aren't sure how to proceed.

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-distributed data created above to show how it works.

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



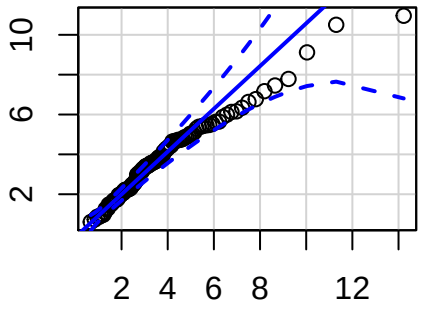
A Cullen and Frey graph compares the data set (blue dot) and bootstrapped replications (orange open circles) to common theoretical distributions. For example, if the blue dot were at or near the * symbol at the top left and more or less surrounded by the orange open circles, it would imply the data are 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 simulations 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:

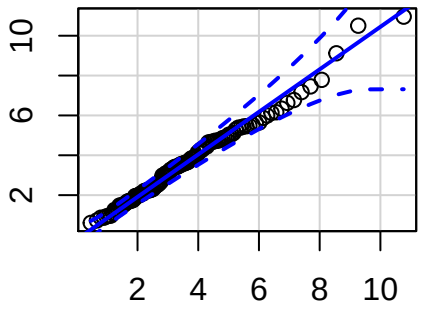
```
# Create objects of the two most-likely distributions
logno = fitdistr(gamma_example$y, "lognormal")
gammo = fitdistr(gamma_example$y, "gamma")

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



lnorm quantiles

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



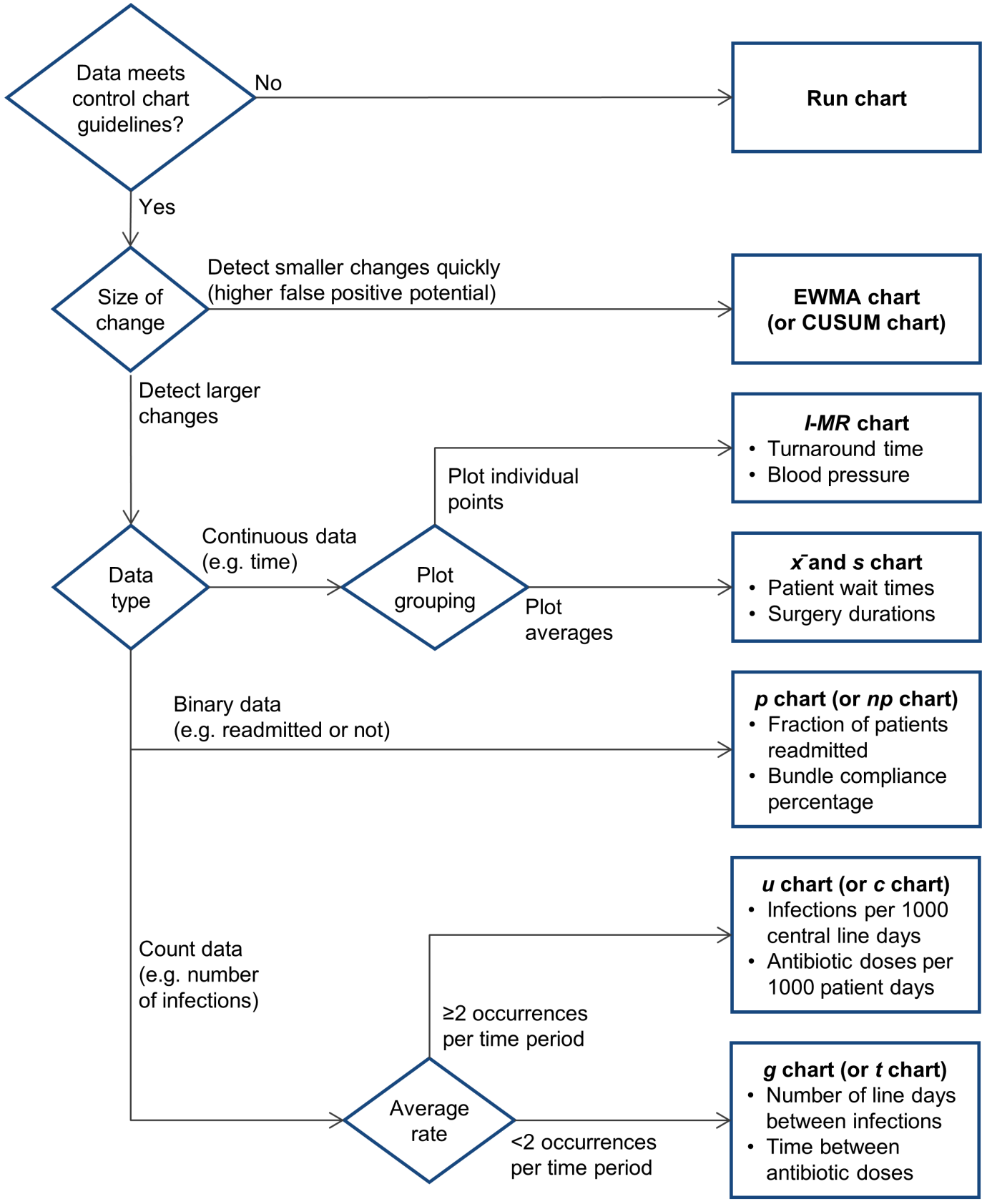
gamma quantiles

Although both distributions fall within the confidence limits (dashed lines), the points fit to a gamma distribution 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.

6.2 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 the chapter A Guide to Control Charts.


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
g chart (or t chart)
• Number of line days between infections
• Time between antibiotic doses
<2 occurrences per time period

Think back to Rachel's dataset. What control chart should we use for her data?

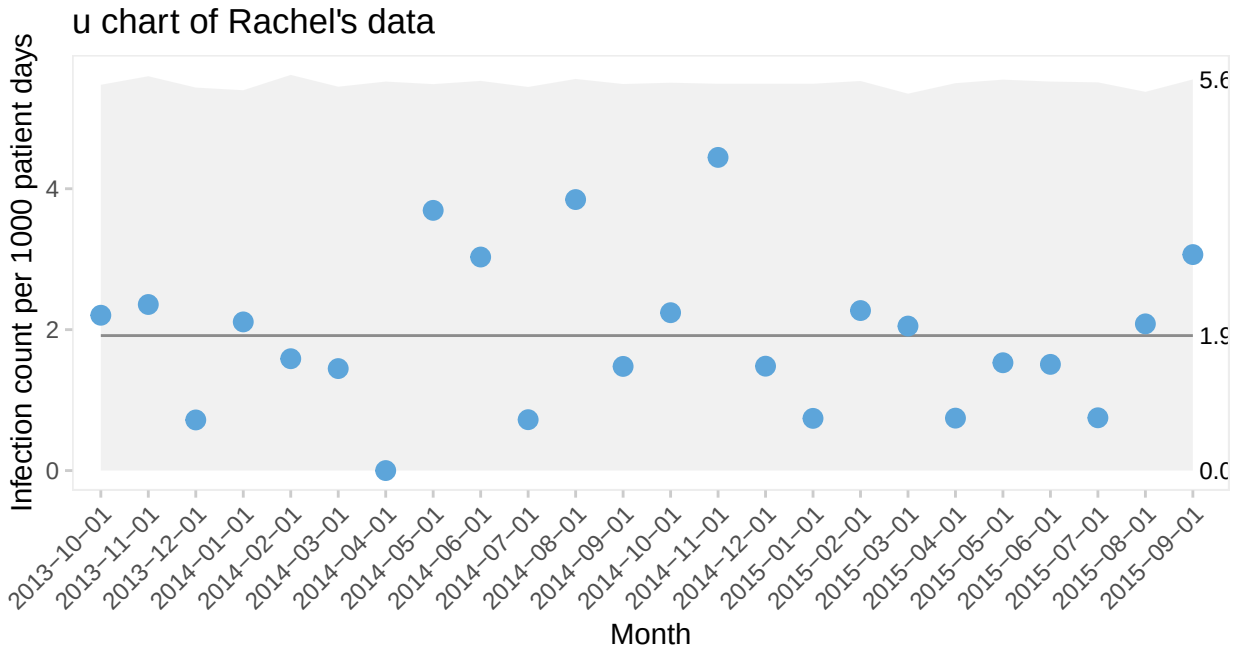
Let's walk through the flow chart to answer this question.

Does the data meet control chart guidelines? Yes. *Size of change* We want to detect larger changes. *Data type* The data is count data. *Average rate* We have more than 2 occurrences per time period.

We want to use a *u* chart! The flowchart also lists common applications for each type of chart. Our decision is backed up by the chart because 'Infections per 1000 central line days' is a common reason for using a *u* control chart.

Now we can use the `qicharts2::qic` function to create the chart. This is the same exact function that we used to create the run chart in the previous chapter. The only difference is that we need to change the `chart` argument to `chart = 'u'`. The documentation for this function contains the full list of chart options.

```
qicharts2::qic(x = months, y = infections, n = linedays, data = rachel_data,
               multiply = 1000, chart = 'u', x.angle = 45,
               title = "u chart of Rachel's data", xlab = "Month",
               ylab = "Infection count per 1000 patient days")
```



From this chart, there does not appear to be any special cause variation. The CLABSI count is stable. In the next chapter we will discuss interpreting run and control charts, as well as clearing up any remaining confusion on which is more appropriate to use for your data.

Part III

7 Run Charts vs. Control Charts ... 40

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

7.2 Guidelines for interpreting SPC charts

8 A Guide to Control Charts 44

8.1 Types of Control Charts

8.2 Tips and tricks for successful control chart use

8.3 Custom SPC function

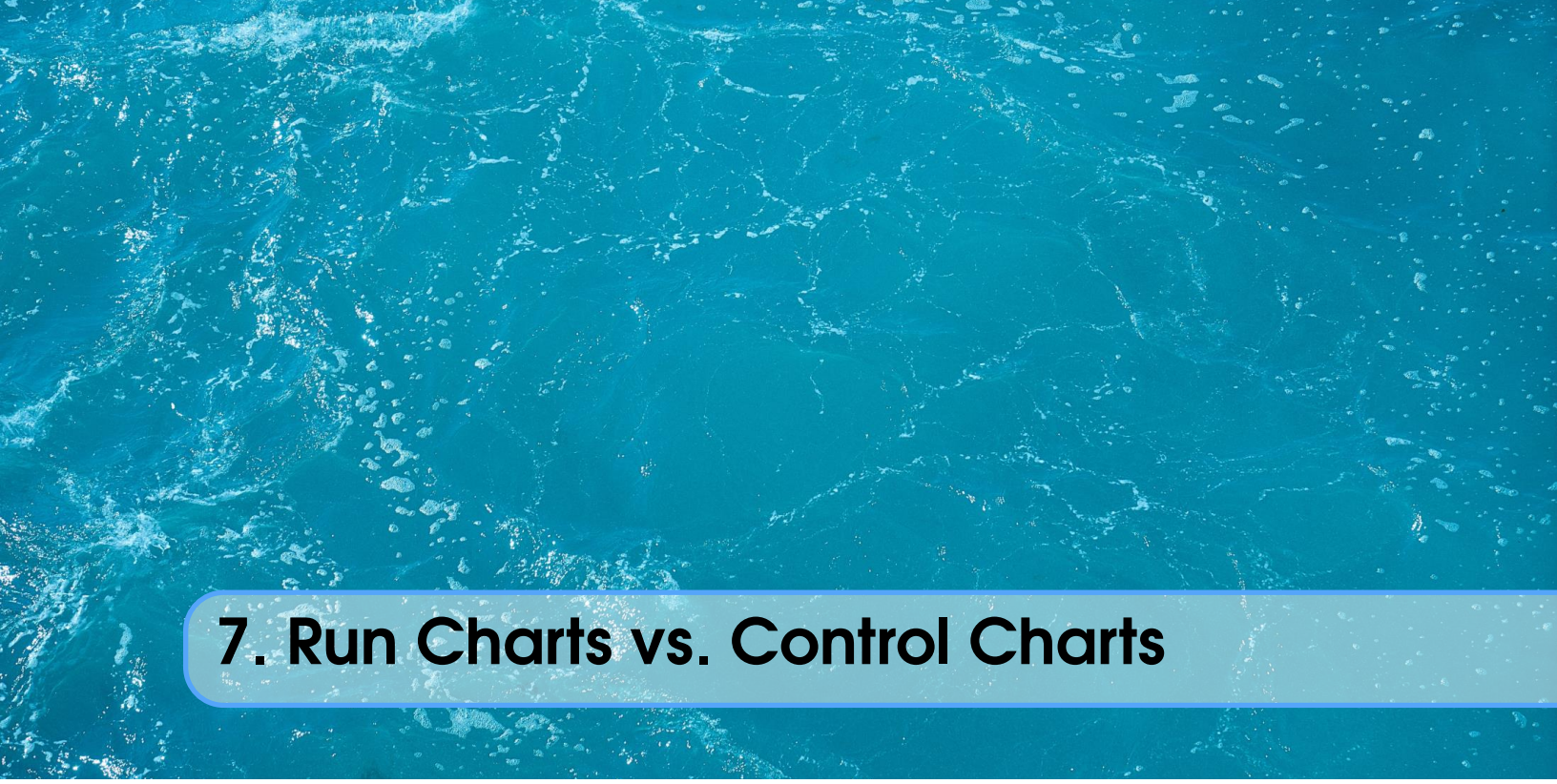
9 Additional Resources 60

9.1 Time series EDA

9.2 Custom SPC Function

9.3 Code used to generate examples

9.4 Useful References



7. Run Charts vs. Control Charts

We have created two useful reference tables that explore the difference between run and control charts. The first elaborates on when to use each chart. The second is how to interpret each chart. More details on specific controls charts can be found in the the chapter A Guide to Control Charts.

7.1 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.

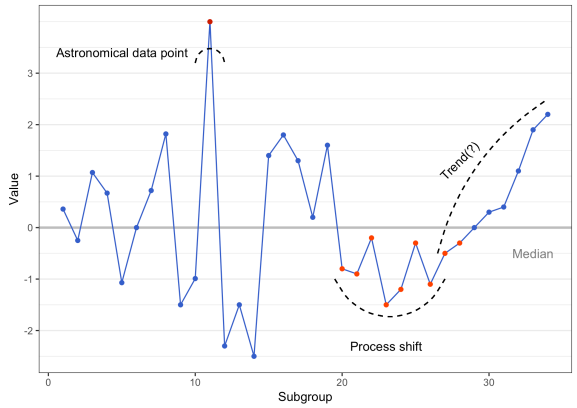
In both cases, the data points must be independent, that is, the position of one point does not influence the position of another point: there is no (serious) autocorrelation. If the data are autocorrelated, the guidelines for testing run or control charts can be invalid, which can lead to poor decision-making.

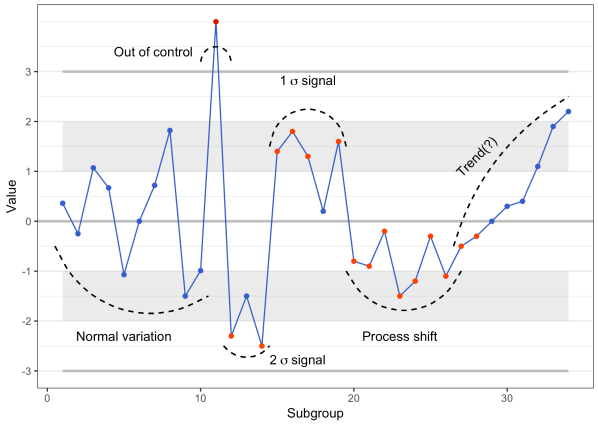
Use a run chart if	Use a control chart (only) if
You may or may not investigate or act when a data point crosses a reference, target, or goal level, or when guidelines suggest a non-random pattern is occurring.	You intend to investigate or act when the process moves outside of control or indicates special 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).

Use a run chart if	Use a control chart (only) if
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.
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.

7.2 Guidelines for interpreting SPC charts

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.

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.

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).

Trend: Seven or more consecutive points all increasing or all decreasing (though this can be an ineffective indicator*).

Cycles: There are obvious cycles of any sort.

Reduced variation: Fifteen or more consecutive points all within 1σ.

Run chart	Control Chart
	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).*

8. A Guide to Control Charts

In this chapter we go over specific types of control charts, discuss tips and tricks for working with control charts, and introduce you to our own custom SPC plot function.

8.1 Types of Control Charts

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
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 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 denomina-

tor, 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 for discrete events/time steps, or *t*-charts for continuous time.

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. Because they are not standard, they are not included in the `qicharts2` package. We did create a custom SPC function which can be used for these charts, shown later in this chapter.

8.1.1 *u*-chart example

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

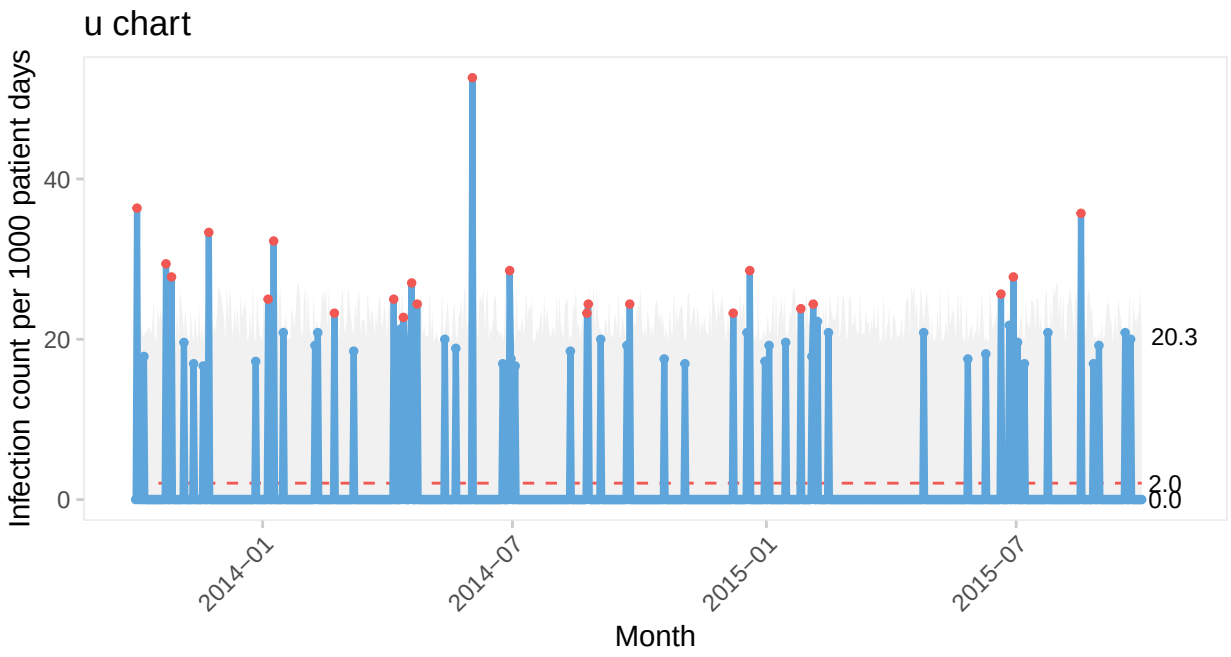
Sometimes, a KPI is based on counts. This is obviously problematic for process monitoring in most healthcare situations because it ignores the risk exposure—for example, counting the number of infections over time is meaningless if you don't account for the change in the number of patients in that same 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

```
qicharts2::qic(x = months, y = infections, n = linedays, data = uchart_data,
               multiply = 1000, chart = 'u', x.angle = 45,
               title = "u chart", xlab = "Month",
               ylab = "Infection count per 1000 patient days")
```



8.1.2 *p*-chart example

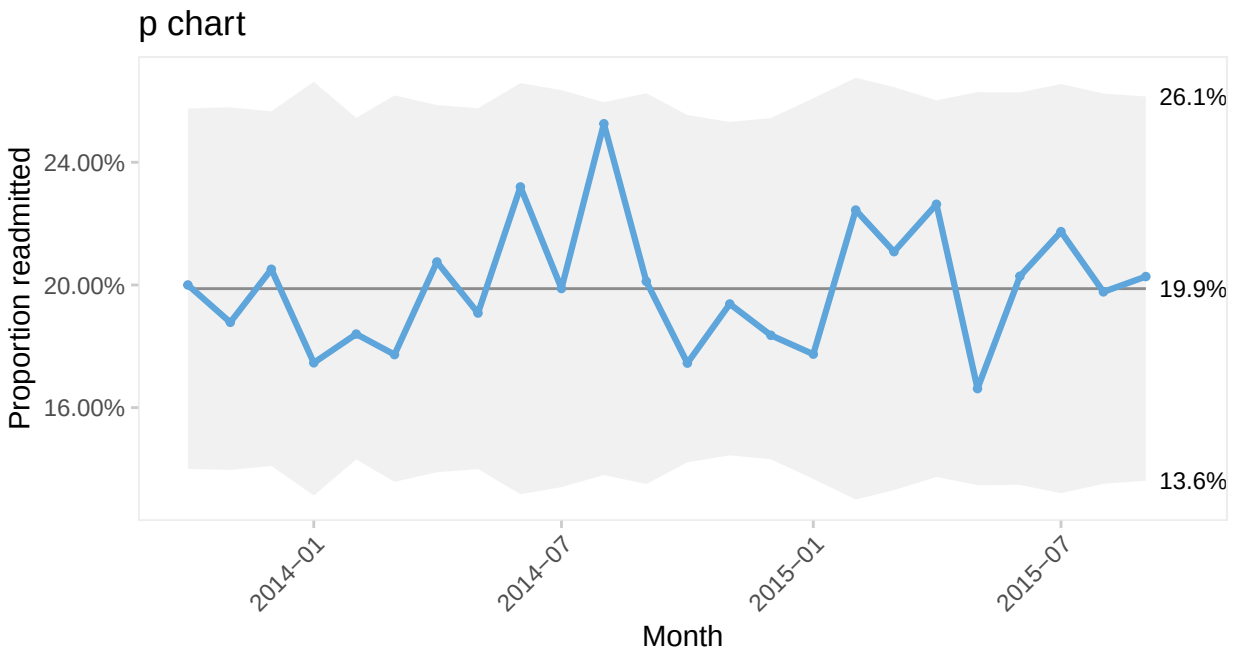
When your metric is a true proportion (and not a rate, e.g., a count per 100), a *p*-chart is the appropriate control chart to use.

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

```
qicharts2::qic(x = dates, y = readmits, n = discharges, data = pchart_data,
               y.percent = TRUE, chart = 'p', x.angle = 45,
               title = "p chart", xlab = "Month",
               ylab = "Proportion readmitted")
```



8.1.3 *g*-chart example

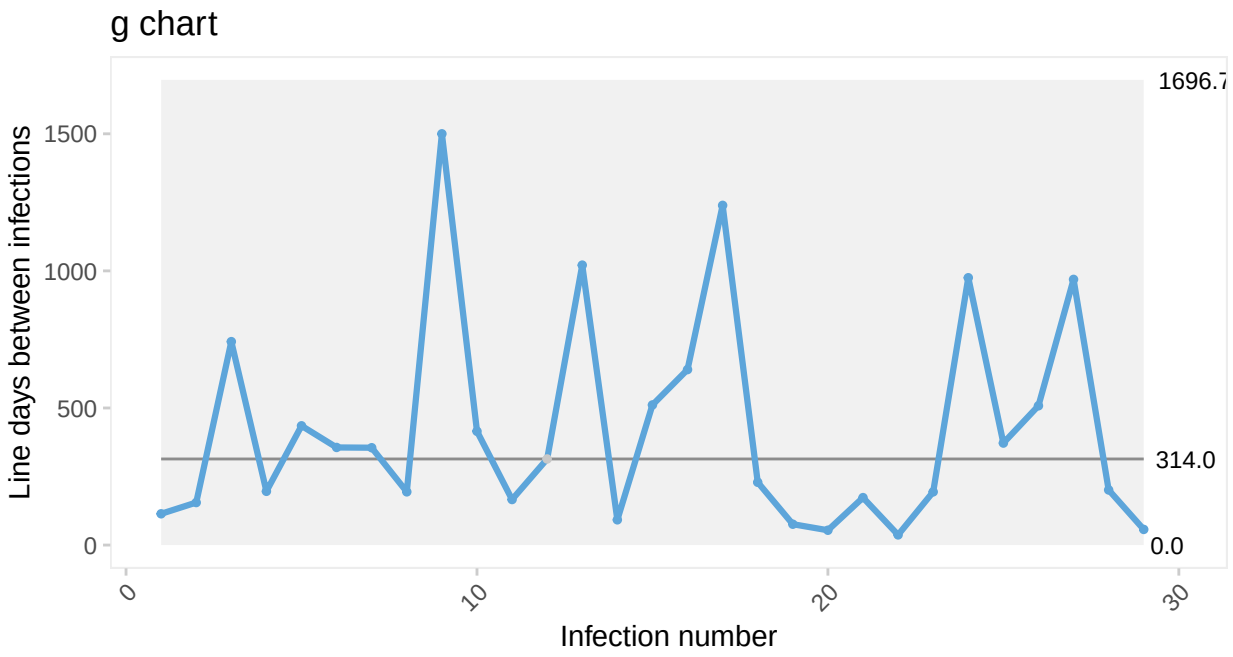
There are important KPIs in healthcare related to rare events, such as is common in patient safety and infection control. These commonly have 0 values for several subgroups within the process time-period. In these cases, you need to use g-charts for a discrete time scale (e.g., days between events) or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

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

```
qicharts2::qic(x = inf_index, y = days_between, data = gchart_data,
               chart = 'g', x.angle = 45, title = "g chart",
               xlab = "Infection number",
               ylab = "Line days between infections")
```



8.1.4 *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): $3\sqrt{c}$

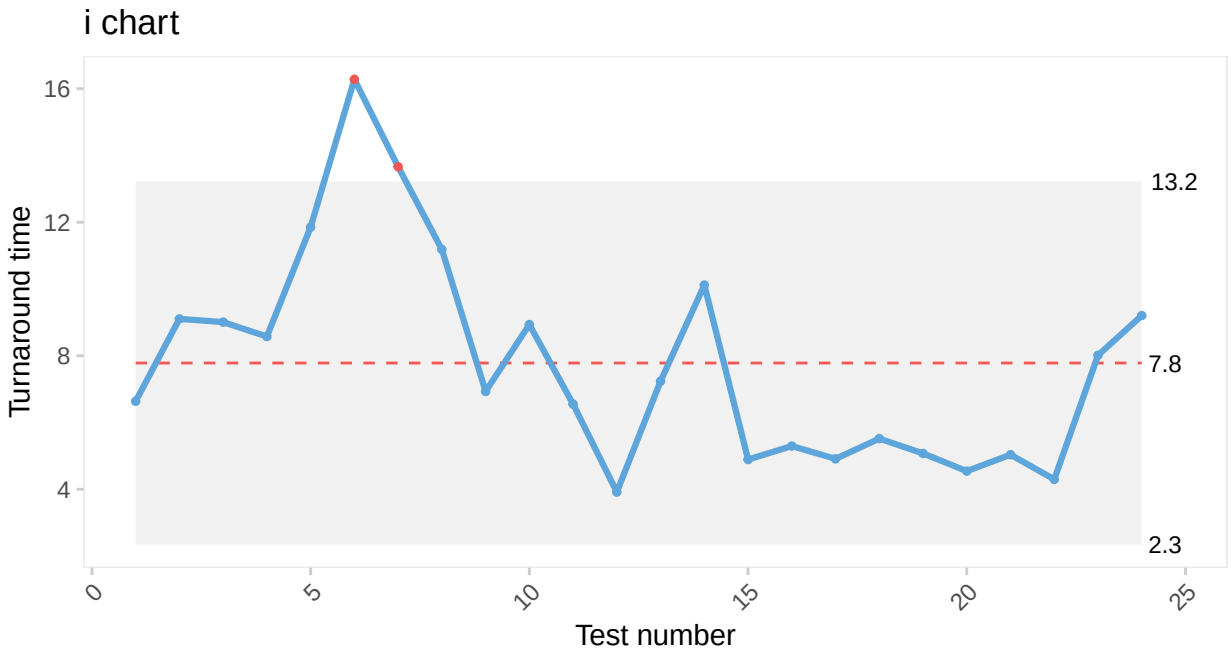
Mean for equal-opporuntity proportions (*np*): $np = \frac{\Sigma y_i}{n}$
where
n is a constant
3σ control limits for equal-opporuntity proportions (*np*): $3\sqrt{np(1-p)}$
where
n is a constant

8.1.5 *I-MR* chart

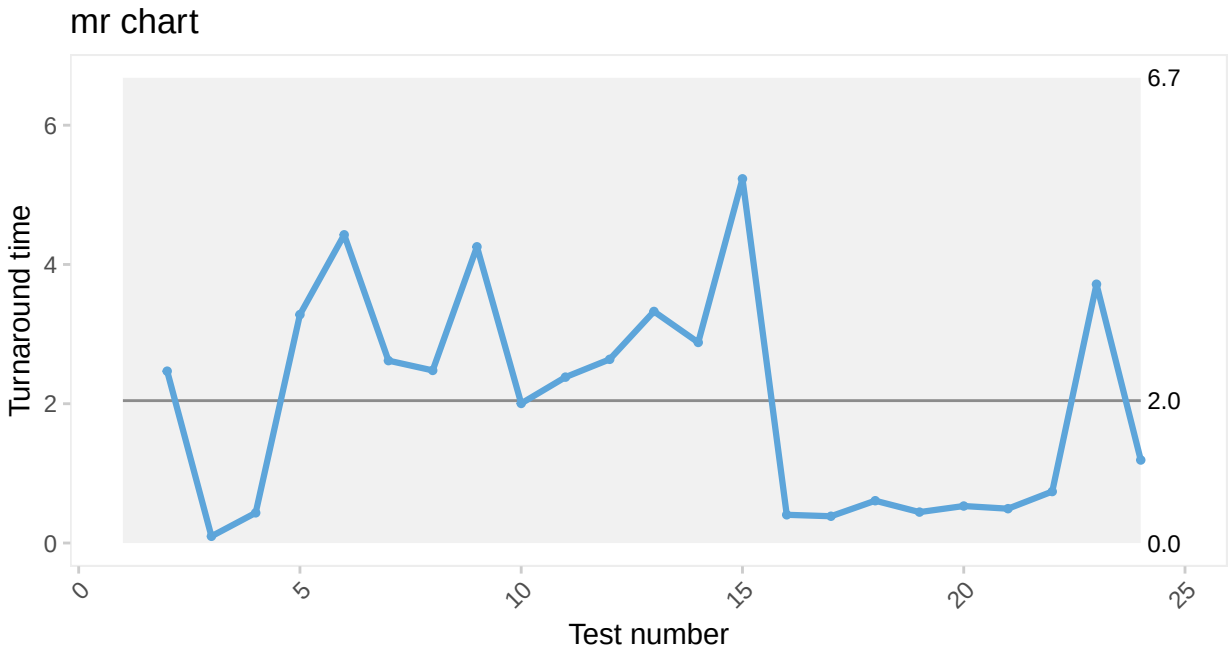
When you have a single measurement per subgroup, the *I-MR* combination chart is appropriate. They should always be used together. When using the `qicharts2` package, this means calling the `qic` function twice, once for each type of plot.

Mean($\bar{x}$): $\bar{x} = \frac{\Sigma x_i}{n}$
Control limits for normal data (*I*): $2.66MR_{bar}$
where
MR_{bar} = average moving range of xs, excluding those > $3.27MR_{bar}$
Lab results turnaround time

```
qicharts2::qic(x = test_num, y = turnaround_time, data = imrchart_data,
               chart = 'i', x.angle = 45, title = "i chart",
               xlab = "Test number", ylab = "Turnaround time")
```




```
qicharts2::qic(x = test_num, y = turnaround_time, data = imrchart_data,
               chart = 'mr', x.angle = 45, title = "mr chart",
               xlab = "Test number", ylab = "Turnaround time")
```



Unlike the attribute control charts, the *I-MR* chart requires a little interpretation. 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.

8.1.6 $\bar{x}$ and *s* chart

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

Control limits (3) are calculated as follows:

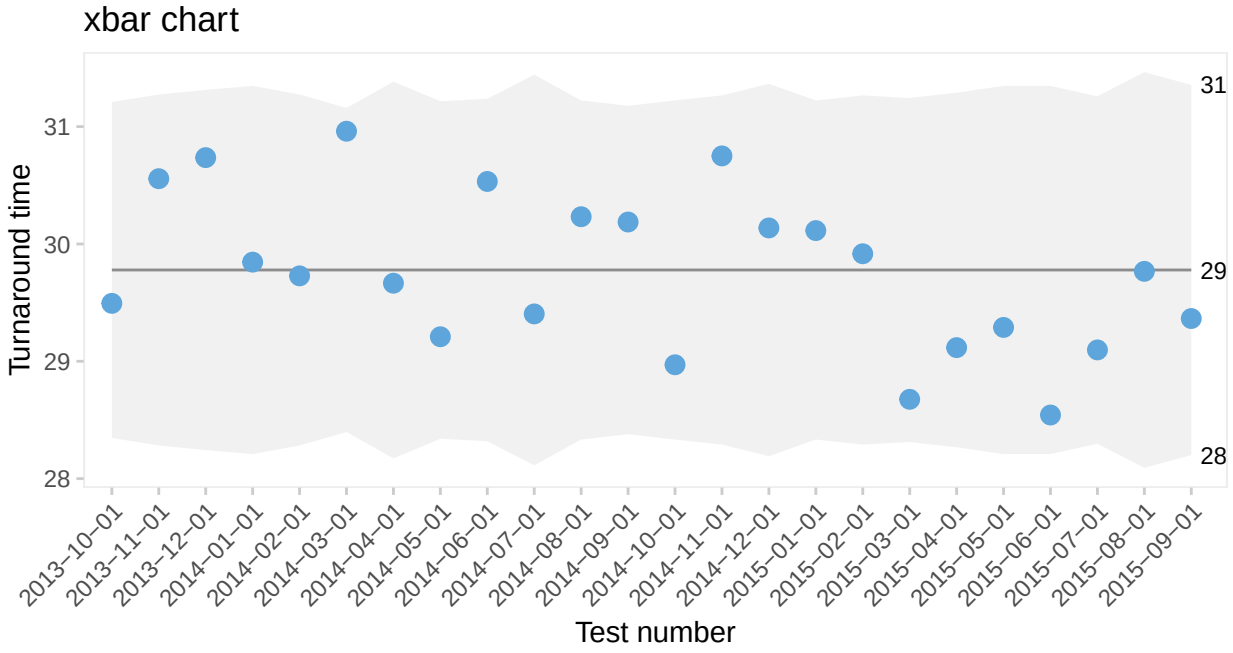
Variable averages ($\bar{x}$): $3\frac{\bar{s}}{\sqrt{n_i}}$

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

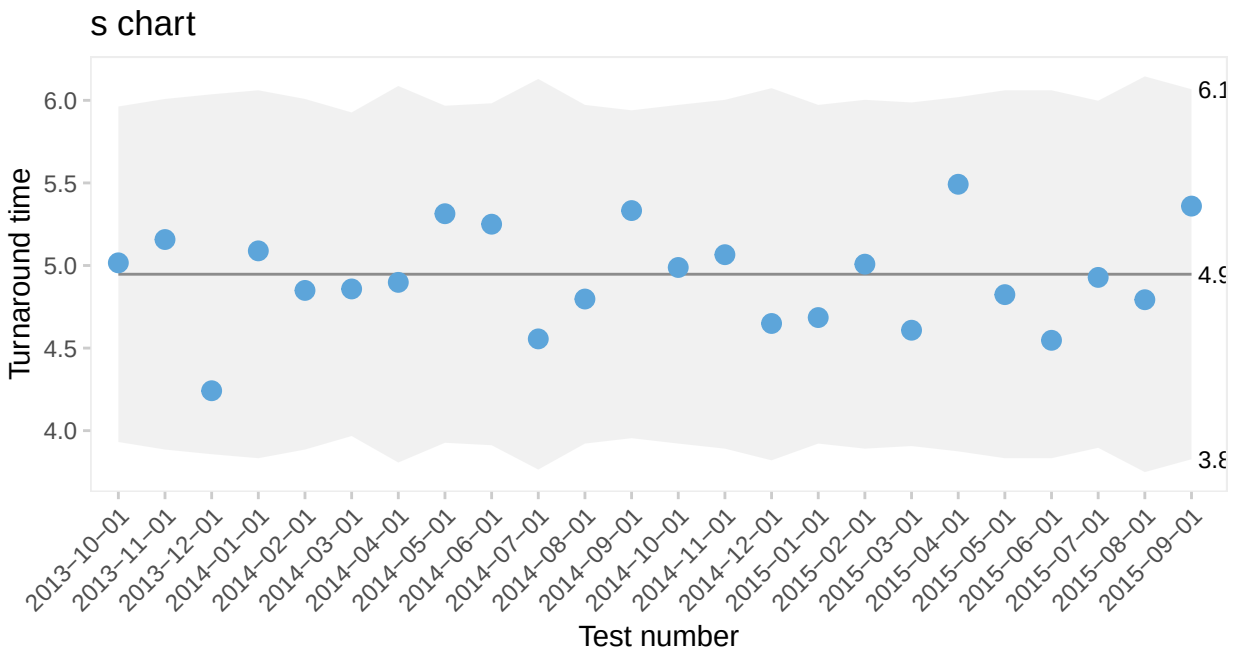
where $c_4 = \sqrt{\frac{2}{n-1}}\frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})}$

Patient wait times

```
qicharts2::qic(x = months, y = waits, data = xbarschart_data,
               chart = 'xbar', x.angle = 45, title = "xbar chart",
               xlab = "Test number", ylab = "Turnaround time")
```




```
qicharts2::qic(x = months, y = waits, data = xbarschart_data, chart = 's',
               x.angle = 45, title = "s chart", xlab = "Test number",
               ylab = "Turnaround time")
```



Just 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.

8.1.7 *t*-chart example

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 may be simpler to implement and easier to interpret without transformation, though a *t*-chart is also acceptable.

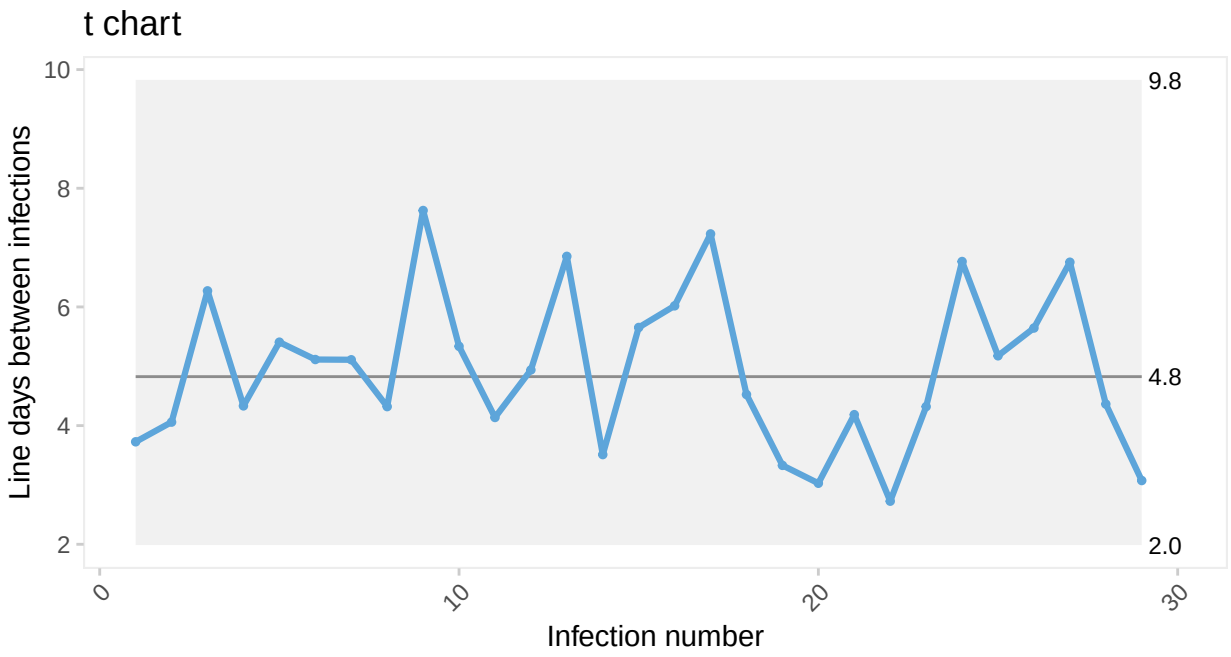
Mean for time between events (*t*)(not shown): $t = \bar{x}(y_i)$
where
t = time between events, where t is always > 0
$y = t^{\frac{1}{3.6}}$

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

```
qicharts2::qic(x = inf_index, y = days_between, data = tchart_data,
               chart = 't', x.angle = 45, title = "t chart",
               xlab = "Infection number",
               ylab = "Line days between infections")
```



8.2 Tips and tricks for successful control chart use

- 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.

8.2.1 When to revise control limits

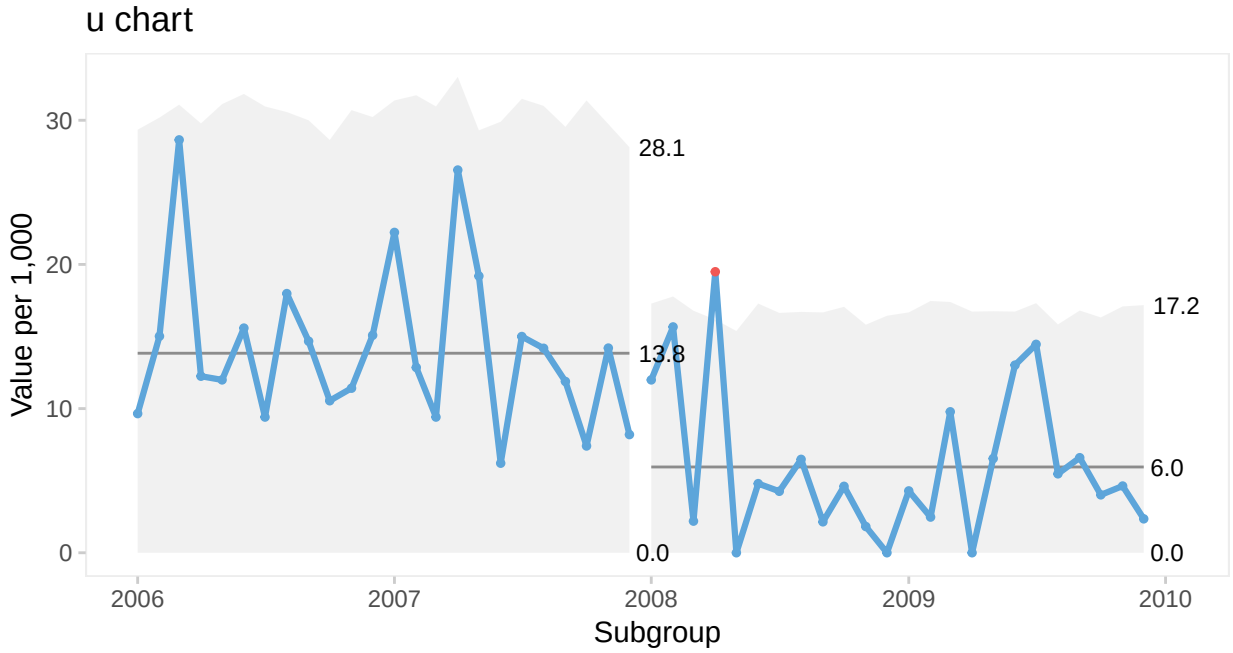
If you need to determine whether an intervention might have worked soon after or even during the improvement process, you shouldn't be using a standard control chart at all. Use a run chart or an EWMA or CUSUM chart to try to detect early shifts.

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.

Let's look at an example using data that we've created.

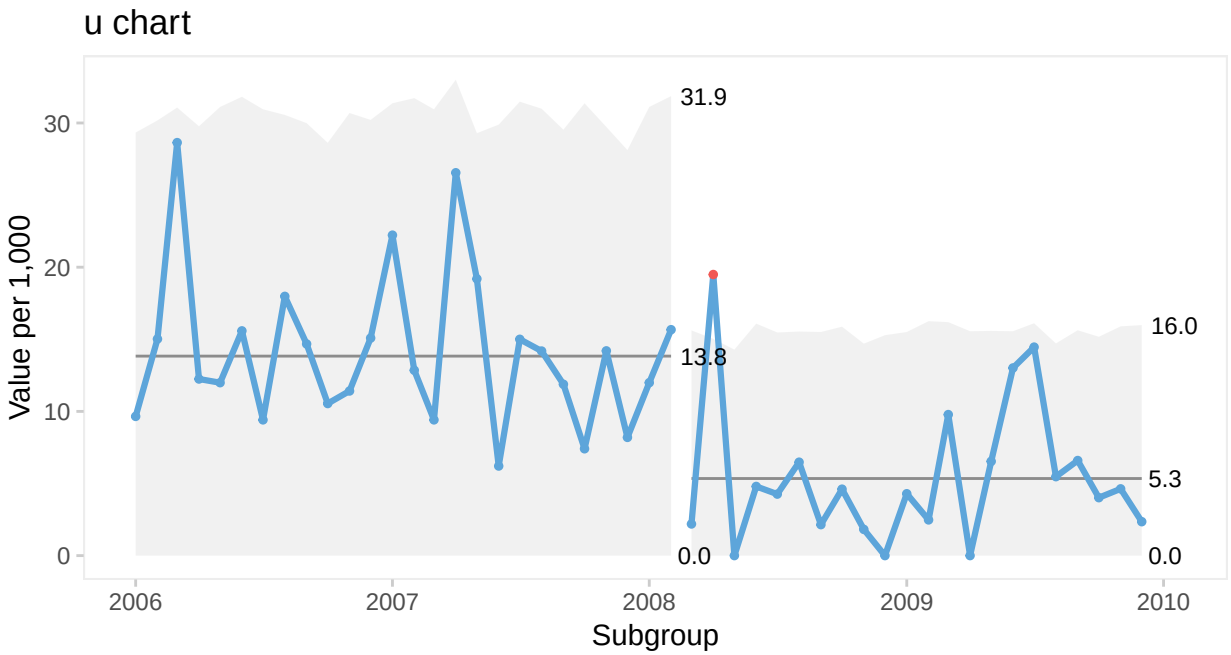
Say that an intervention happened at the start of year 3, but there was a lag between the intervention and when it actually showed up in the data.

```
qicharts2::qic(x = date, y = y, n = n, data = intervention, chart = 'u',
               multiply = 1000, title = "u chart", ylab = "Value per 1,000",
               xlab = "Subgroup", part = 24)
```

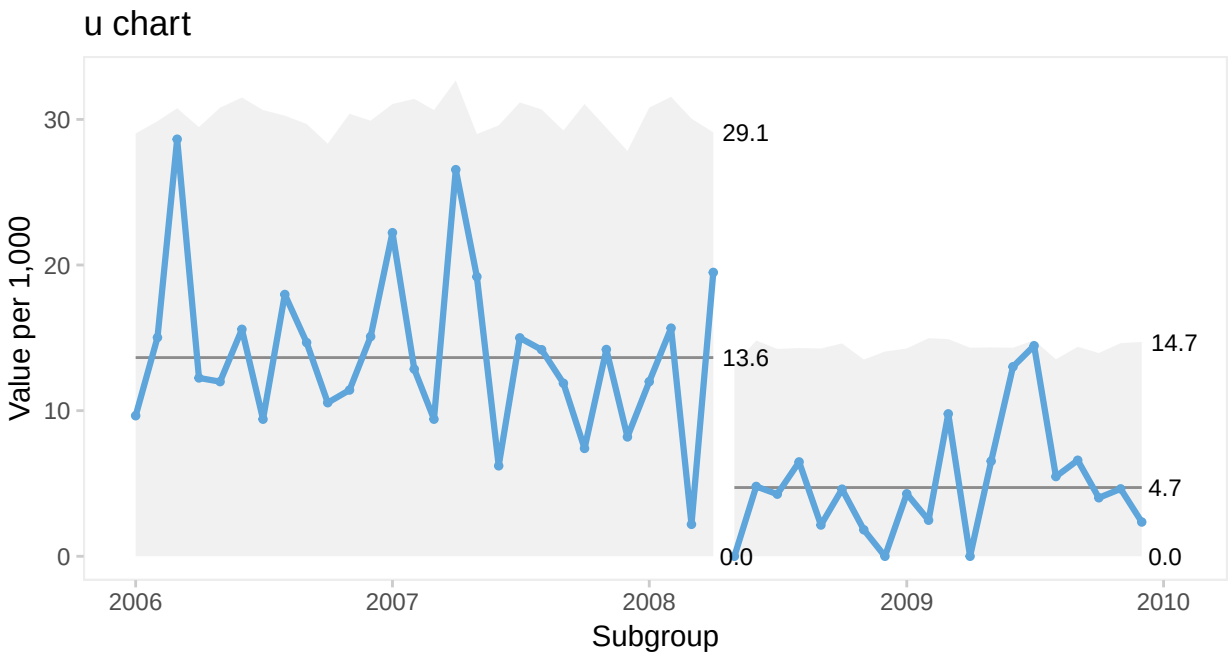


Of course, the change point can be placed arbitrarily in a `qic` graph—with corresponding changes in control limits. For example, using the same data as above, compare those results with those when the change point is moved forward by 2, 4, or 6 time steps (pretending we don't actually know when the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

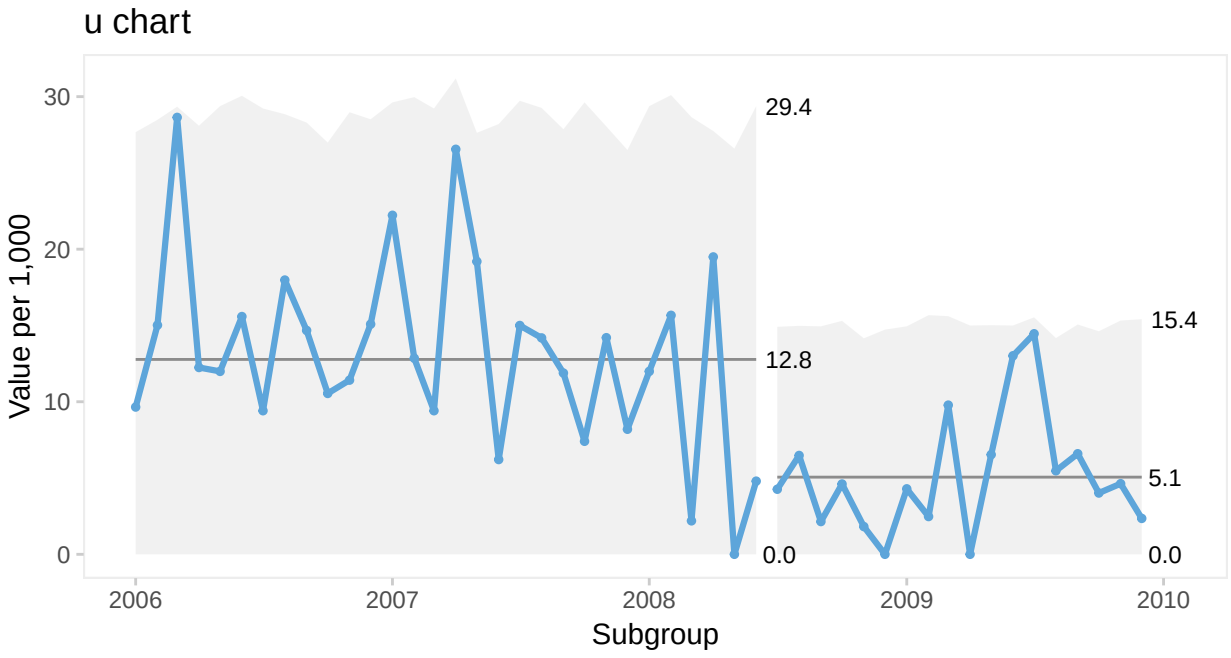
```
qicharts2::qic(x = date, y = y, n = n, data = intervention, chart = 'u',
               multiply = 1000, title = "u chart", ylab = "Value per 1,000",
               xlab = "Subgroup", part = 26)
```




```
qicharts2::qic(x = date, y = y, n = n, data = intervention, chart = 'u',
               multiply = 1000, title = "u chart", ylab = "Value per 1,000",
               xlab = "Subgroup", part = 28)
```




```
qicharts2::qic(x = date, y = y, n = n, data = intervention, chart = 'u',
               multiply = 1000, title = "u chart",  ylab = "Value per 1,000",
               xlab = "Subgroup", part = 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.

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.

8.3 Custom SPC function

The `qicharts2` package is great for plotting a wide variety of charts. However, it does not contain the ability to plot EWMA or CUSUM charts. Also you might want more fine control over your plot. In these situations, you have to create the plot from scratch. We have created the following custom SPC function. Below are the EWMA and CUSUM plots using this function. The full code creating the function and examples of previous charts we have gone over can be found in the section Custom SPC Function.

8.3.1 EWMA chart

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

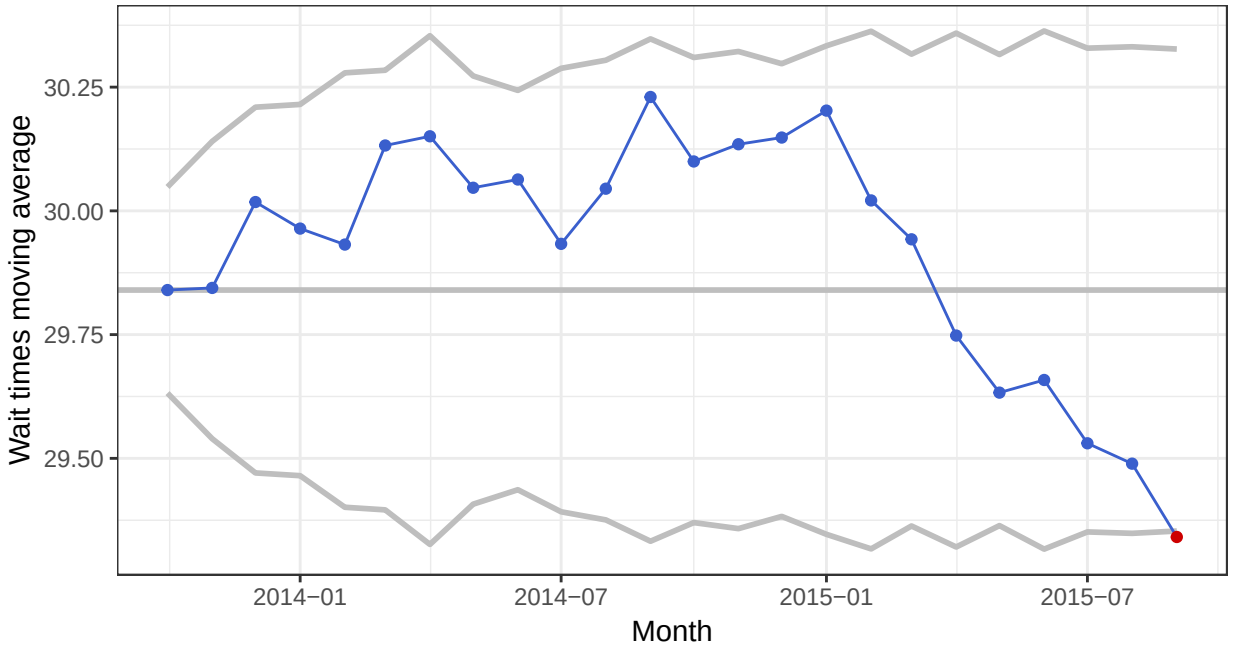
where λ is a weight that determines the influence of past observations. If unsure choose $\lambda = 0.2$, but $0.05 \leq \lambda \leq 0.3$ is acceptable (where larger values give stronger weights to past observations).

Patient wait times (continued)

```
# 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)
```

```
# 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
plotSPC(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")
```



8.3.2 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.

{SKP: finish once EWMA fixed}

Patient wait times (continued)

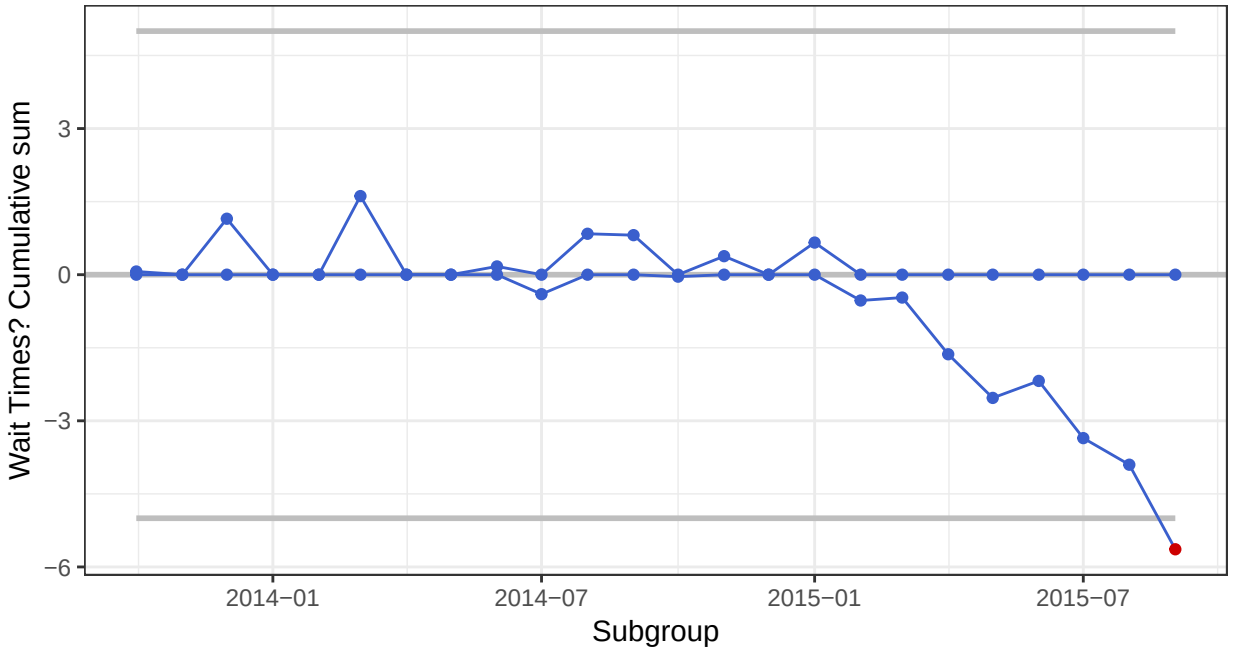
```
# Generate fake patient wait times data
set.seed(777)
waits <- c(rnorm(1700, 30, 5), rnorm(650, 29.5, 5))
months <- strftime(sort(as.Date('2013-10-01') +
                        sample(0:729, length(waits), TRUE)), "%Y-%m-01")
sample.n <- as.numeric(table(months))
dfw <- data.frame(months, waits)

# 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)

# Calculate control chart inputs
subgroup.cusum <- subgroup.x
slack <- 0.5
zscore <- (point.x - mean.x)/sigma.x
point.cusuml <- matrix(nrow = 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(nrow = 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 <- plotSPC(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")
```



9. Additional Resources

9.1 Time series EDA

The chapter Exploratory Data Analysis 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.

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 Rachel's data used in previous chapters has no time series patterns, we'll use the `beer` dataset provided in the `fpp2` package. It has clear time-related patterns to explore with EDA tools.

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

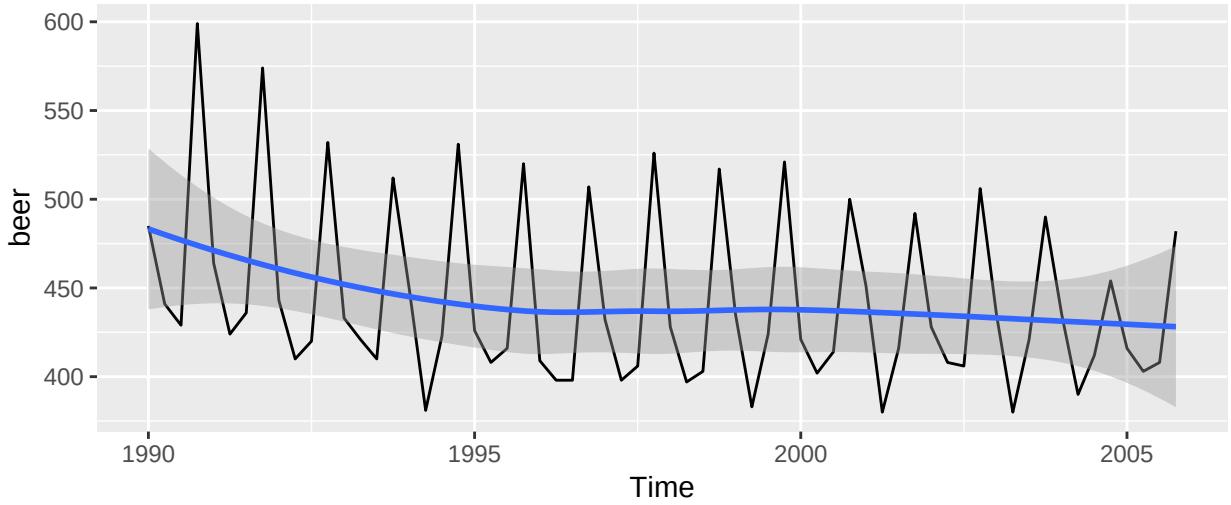
9.1.1 Trend

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.

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

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

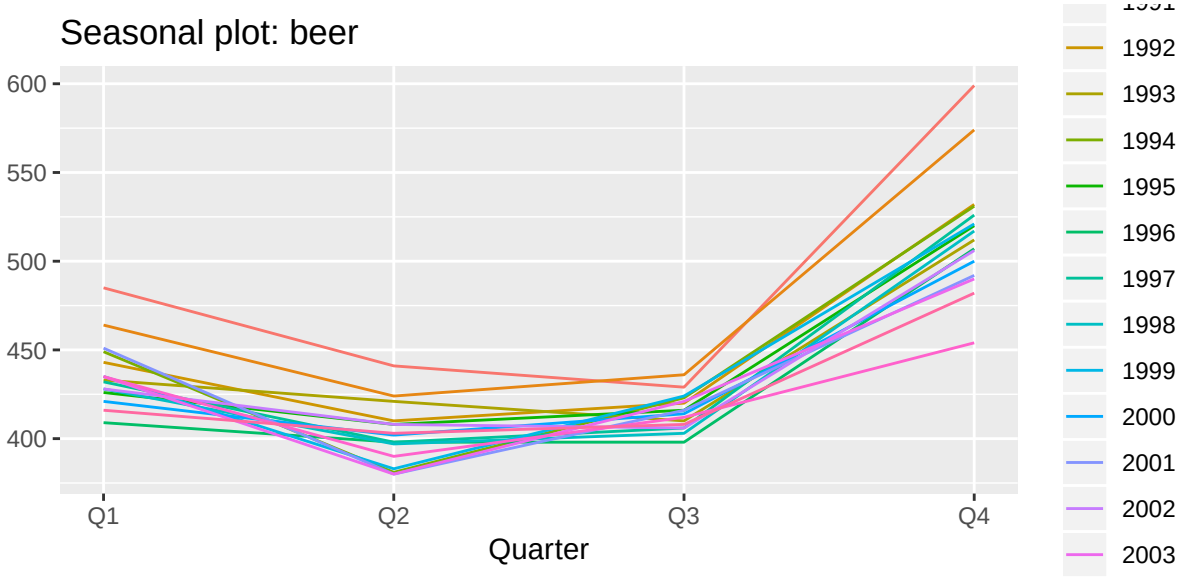


9.1.2 Seasonplot

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.

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)
```

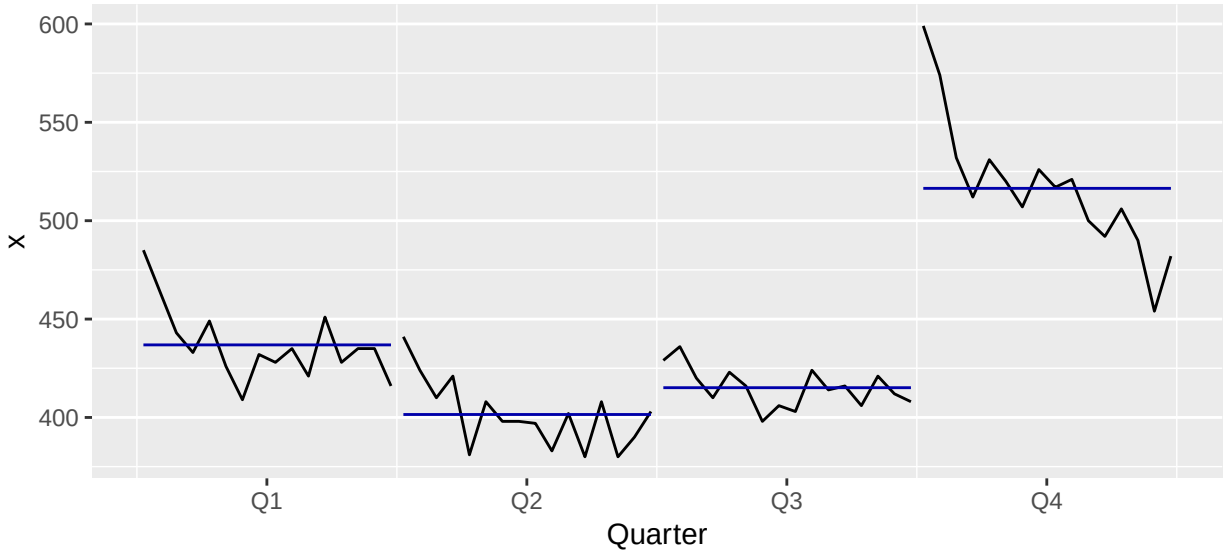


9.1.3 Monthplot

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. 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)
```



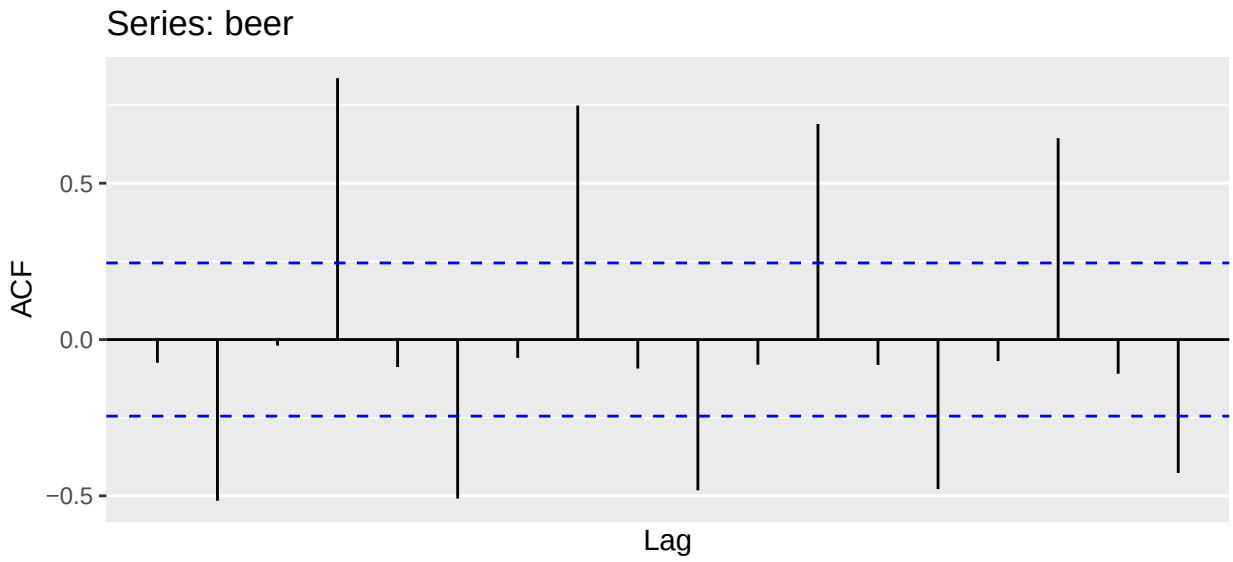
9.1.4 Autocorrelation

We've touched on autocorrelation in other portions of this book.

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 `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))
```

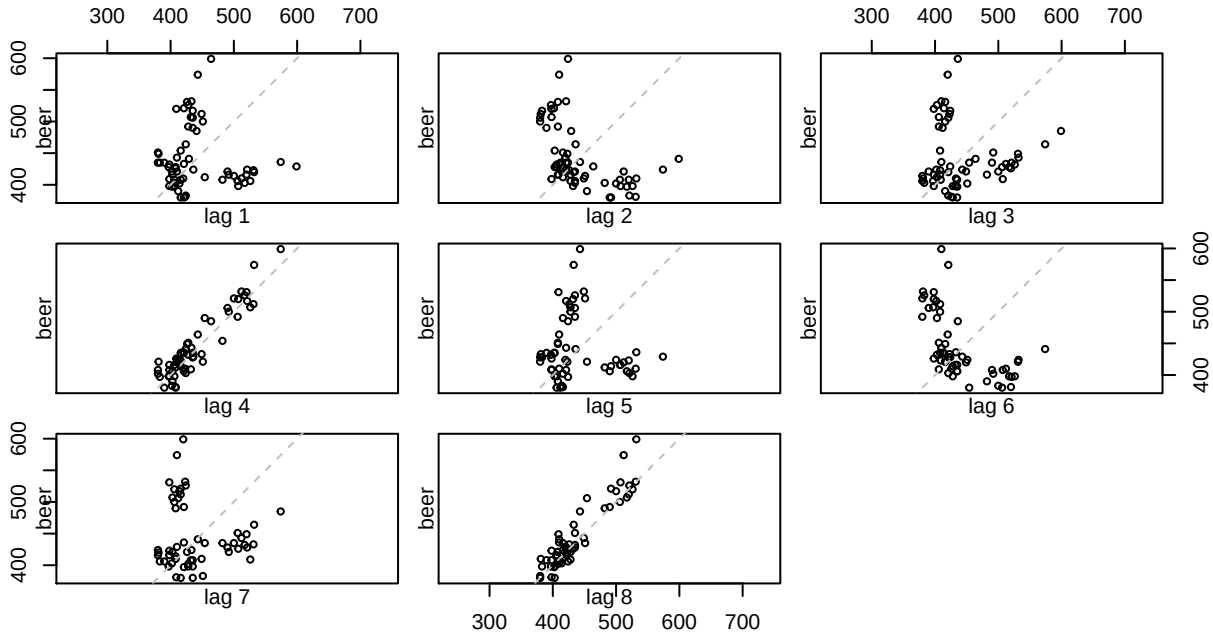


The autocorrelation function is most concisely plotted with the approach above, but you can also plot the increasing lags against an initial value in individual scatterplots. If the points look like a shotgun target, there's no autocorrelation. Patterns in the points indicate autocorrelation in the data. 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.

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

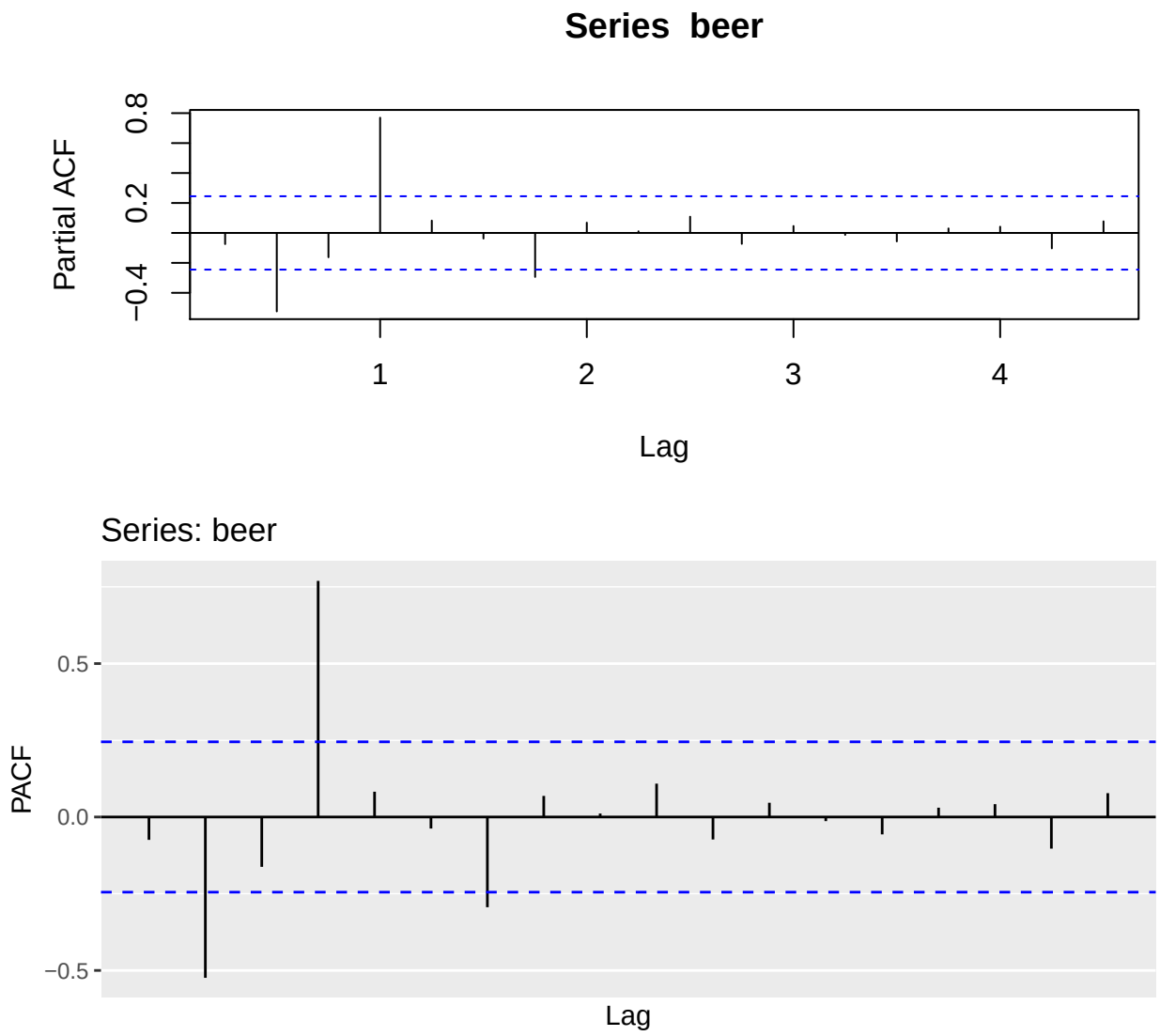
```
# Scatterplot of beer data autocorrelation through first 8 lags
lag.plot(beer, lags = 8, do.lines = FALSE)
```



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.

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))
```



9.1.5 Cycles

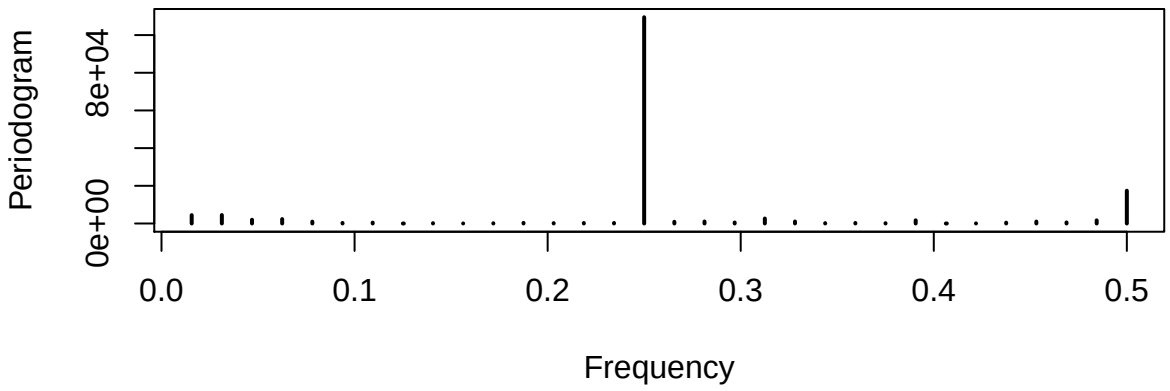
Periodograms allow you to explore a time series for cycles that may or may not be regular in timing (which makes it slightly distinct from seasonality). Sunspot cycles are a classic example at

~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 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.

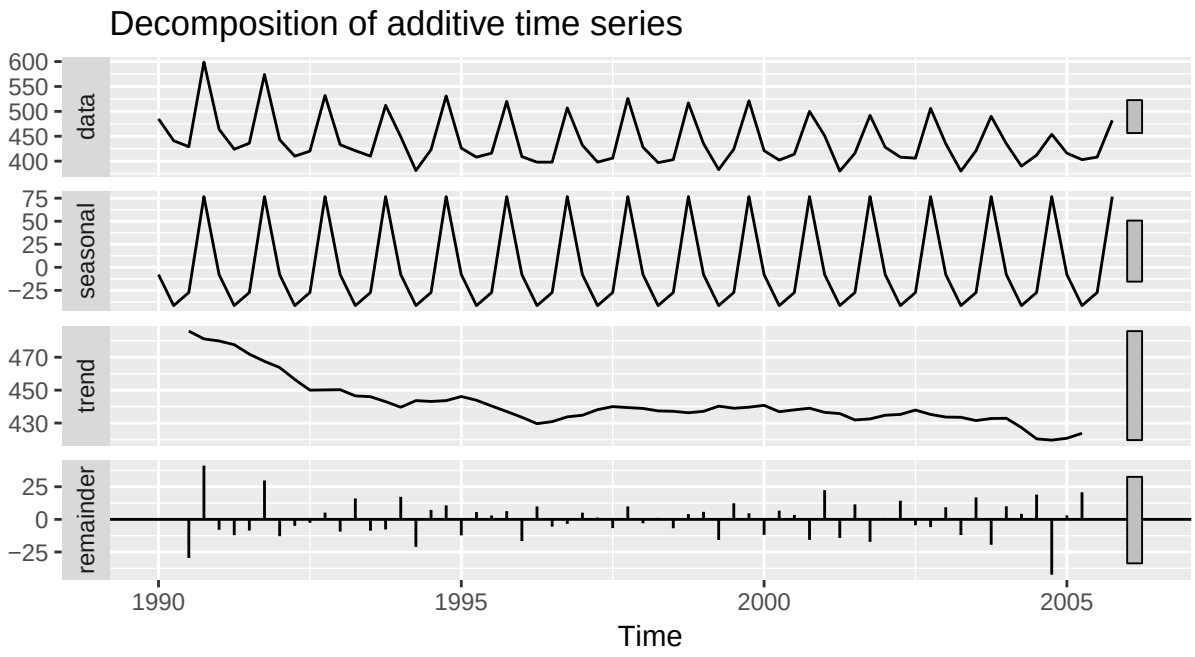
```
TSA::periodogram(beer)
```



9.1.6 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(beer))
```



9.1.7 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.

```
# 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.

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

9.1.8 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.

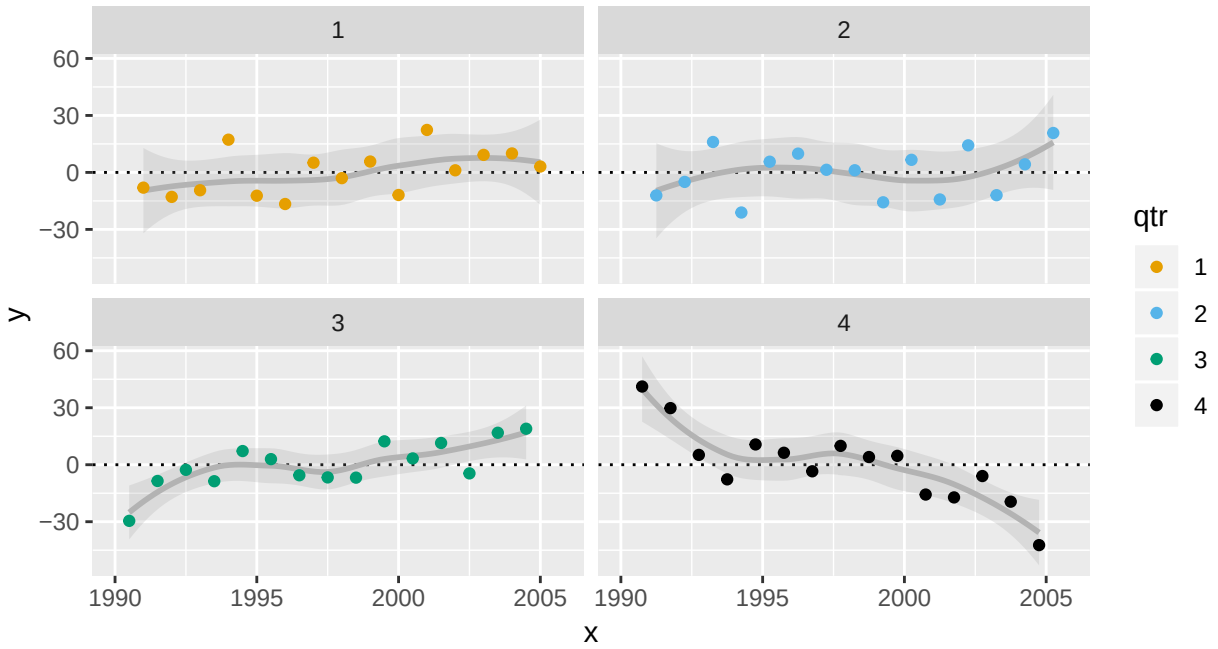
```
# Convert ts residuals to data frame
beer_df_rand = tsdf(decompose(beer)$random)

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

# Plot residuals, with custom colors
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"))
```

We can take that same information and facet by quarter for a different perspective.

```
# 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"))
```



9.1.9 Accumulation 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.

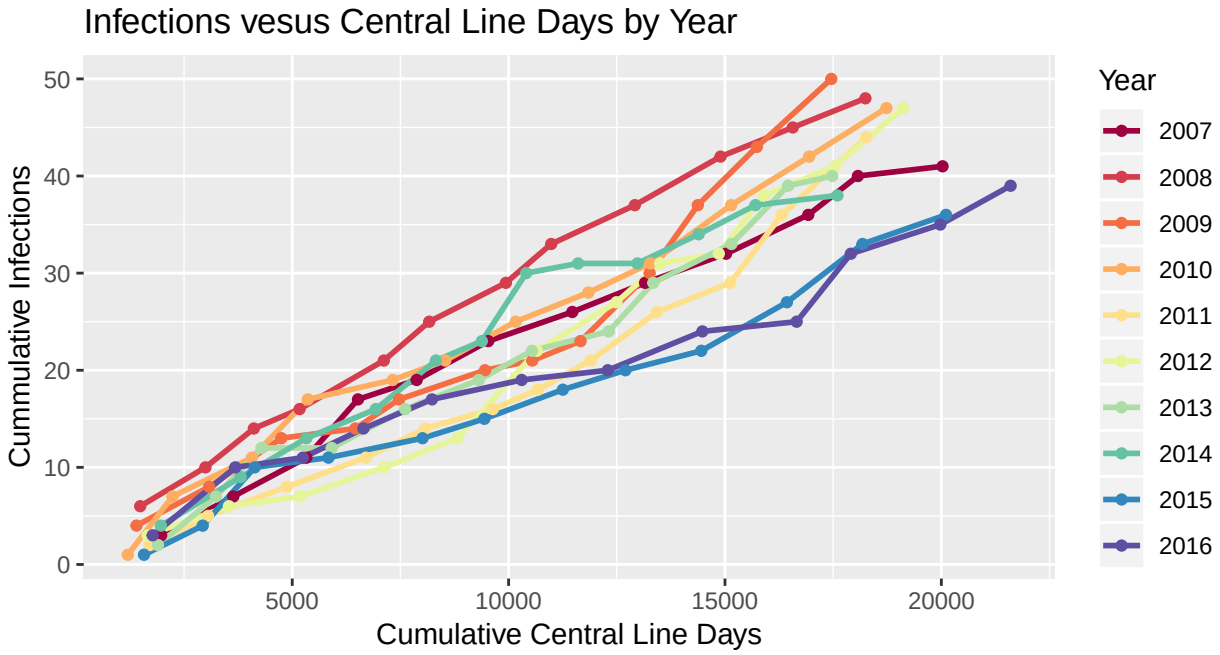
```
# Generate sample data
set.seed(54)
bsi_8yr = data.frame(Linedays = sample(1000:2000, 96),
                     Infections = rpois(96, 4))
bsi_2yr = data.frame(Linedays = sample(1200:2200, 24),
                     Infections = rpois(24, 3))
bsi_10yr = rbind(bsi_8yr, bsi_2yr)
bsi_10yr$Month = seq(as.Date("2007/1/1"),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



9.2 Custom SPC Function

We've created a function that highlights points that may indicate special cause variation. 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.

```
plotSPC <- 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"),
                       guide = FALSE) +
    labs(x = label.x, y = label.y) +
    theme_bw()
}
```

The following code sets up the variables for replicating the control charts in the chapter A

Guide to Control Charts using the simulated data and the custom SPC function instead of qicharts2.

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



# 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
plotSPC(subgroup.p, point.p, central.p, sigma.p,
         label.x = "Month", label.y = "Proportion readmitted")



# 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
plotSPC(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")



# Calculate IMR control chart inputs
subgroup.i <- seq(1, length(exit))
```

```
subgroup.mr <- seq(1, length(exit) - 1)

point.i <- exit - arrival
point.mr <- matrix(nrow = 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))

# Plot MR chart
plotSPC(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
plotSPC(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")




# Calculate XbarS 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)

# Plot s chart
plotSPC(subgroup.s, point.s, mean.s, sigma.s, k = 3,
```

```
         label.x = "Month", label.y = "Wait times standard deviation (s)")

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



# 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
plotSPC(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)")
```

9.3 Code used to generate examples

```
### Rachel's Data
# Set seed for reproducibility
set.seed(2019)

# Generate fake infections data
dates <- strftime(seq(as.Date("2013/10/1"), by = "day", length.out = 730),
                  "%Y-%m-01")
linedays <- sample(30:60,length(dates), replace = TRUE)
infections <- rpois(length(dates), 2/1000*linedays)


# Aggregate the data by month
infections <- aggregate(infections, by = list(dates), FUN = sum,
                        na.rm = TRUE)$x
linedays <- aggregate(linedays, by = list(dates), FUN = sum, na.rm = TRUE)$x
months <- unique(dates)

# Create a tibble
rachel_data = tibble(months, infections, linedays)
```

```
### example control charts

# Set seed for reproducibility
set.seed(72)

#### u chart
# Generate fake infections data
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)

# Aggregate the data by month
infections <- aggregate(infections, by = list(dates), FUN = sum,
                        na.rm = TRUE)$x
linedays <- aggregate(linedays, by = list(dates), FUN = sum, na.rm = TRUE)$x
months <- unique(dates)

# Create a tibble
uchart_data <- tibble(months, infections, linedays)


#### p chart
# 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)

# Create a tibble
pchart_data <- tibble(dates, readmits, discharges)


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

linedays.btwn <- matrix(nrow = length(dfind$start))

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

```
}

gchart_data <- tibble(inf_index = 1:length(linedays.btwn),
                      days_between = linedays.btwn)


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

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

imrchart_data <- tibble(turnaround_time = exit - arrival,
                        test_num = 1:length(exit))



#### XbarS chart
# 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))
xbarschart_data <- tibble(months, waits)



##### t chart
# Generate sample data using g-chart example data
y <- linedays.btwn ^ (1/3.6)
mr <- matrix(nrow = length(y) - 1)
for (i in 1:length(y) - 1) {
  mr[i] <- abs(y[i + 1] - y[i])
}
mr <-  mr[mr <= 3.27 * mean(mr)]
tchart_data <- tibble(inf_index = 1:length(y), days_between = y)
```

```
##### change in process example
# Create fake data with change in process at 28 months
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)))
```

9.4 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; Mohammed et al. 2008, *Plotting basic control charts: tutorial notes for healthcare practitioners*, BMJ Quality & Safety 17:137-145; and Limaye et al. 2008, *A Case Study in Monitoring Hospital–Associated Infections with Count Control Charts*, Quality Engineering 20:404-413. 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)