Evaluating Control Charts using Simulated Data

Final Project - Masters of Arts - Biostatistics

Vitaly Druker

Spring 2018

Contents

1 Introduction **2**
- 1.1 General Statistical Control . . . 2
- 1.2 Expansion of the Methodology . . . 2
- 1.3 Report Contents . . . 2

2 Classic Statistical Process Control **3**
- 2.1 Overview . . . 3
- 2.2 Tools Used . . . 3

3 Data Simulation **6**
- 3.1 Overall Data Properties . . . 6
- 3.2 Knobs . . . 6

4 Measuring Success **11**
- 4.1 Ability to Fit . . . 11
- 4.2 False Positive Rate . . . 11
- 4.3 True Positive Rate . . . 12

5 Results and Discussion **12**
- 5.1 Ability of QCC to Fit the Data . . . 12
- 5.2 Ability of QCC to correctly identify changes . . . 12
- 5.3 Ability of QCC to minimize False Positive results . . . 15

6 Conclusion and Next Steps **15**
- 6.1 Discussion . . . 15
- 6.2 Next Steps . . . 16

7 Appendix **17**
- 7.1 Packages . . . 17
- 7.2 Main Functions Used . . . 17
- 7.3 Parameter Grid Creation . . . 21
- 7.4 Data Generation, Model Fitting and Evaluation . . . 22
- 7.5 QCC Helper Functions . . . 22
- 7.6 Software Information . . . 28

Bibliography **30**

1 Introduction

1.1 General Statistical Control

Statistical process control has been used in the manufacturing industries since the 1920s. They were popularized by Walter A. Shewhart at Bell Labs (Shewhart 1931). Statistical process control (SPC) attempts to answer a relatively simple question: is a process changing over time?

Specifically, SPC defines two types of variation: common cause (called chance by Shewhart) and special cause (assignable) variation. Common cause variation can be viewed as process variation that is predictable, stable and (sometimes) unavoidable. This is variation that can be see as being inherent to the process itself. For example, a dart player will not hit a bulls-eye every time. The dart play can lower their variation, but it's unlikely that it will become entirely absent.

On the other hand, special cause variation is variation in either the measurement itself or variation in the variance of the measurement that is outside. This is the variation that SPC tries to identify because it needs to be investigated as a possible issue in a process that was thought to be stable before. In continuing with the dart player example from before, there may be cause for concern if the player begins to consistently miss to the left of the bulls eye. The dart player may be fatigued and needs to retire before hitting one of their friends. On the other hand, a dart player may have had a high variance in hitting the bulls-eye and has started to cluster more towards the center of the bulls eye. Perhaps a fellow player demonstrated the correct way to hold a dart and the dart players precision improved. This could also be noted as special cause variation. Special cause variation does not need to imply a negative outcome - simply that there is a *change* in the process that may be attributed to an event.

The reader may begin to formulate some of the balances that need to be struck in these analyses:

When do we decide that the difference in accuracy is consistently significantly different than before? It can be helpful to imagine an extreme situation. The first dart thrown by the player hits the center of the board. The next hits the wall. Was the second throw 'out of control'? It'd be hard to make that decision. However, if 100/1000 first throws are bulls-eyes and the next 200 are not, the problem becomes more difficult.

1.2 Expansion of the Methodology

An additional problem becomes apparent when the field of application is considered. Shewhart developed his theory to be used in a manufacturing setting where measurements could be done fairly cheaply, repeatedly and in a relatively controlled setting. As more industries were able to set up repeatable measurements with advances in technology, SPC has been applied to software engineering (Team 2006), financial services ("An Exploration of Quality Control in Banking and Finance" 2012) and food control (Dora et al. 2013). The strong adoption of SPC by Six Sigma methodology has not hindered this growth (SixSigma (2016)).

Another field that has not been immune to influence of SPC is healthcare (Thor et al. 2007–10AD). Statistical process control can be applied to everything from patient waiting times to clinical outcomes. However, simple classical SPC is ill equipped to deal with the wide variability of the healthcare field (and some of the other fields mentioned above (Raczynski 2013)). Healthcare data can also suffer from missingness, measurement error and issues in consistent collection. All of these concept lead to wide variability and unsatisfying results.

1.3 Report Contents

This report will begin with an in-depth review of statistical process control methodology, specifically as it applies to highly inconsistent data. Some drawbacks in the methods will be exposed using simulated data.

2 Classic Statistical Process Control

2.1 Overview

The main process for statistical control involves a two stage process:

1. Estimate parameters that describe the 'center' and 'spread' of the process being measured.
2. Apply these parameters to observed data to detect any change.

The recommended method for this is to estimate the parameters from an initial stage (Phase I) and then apply them to new data (Phase 2). This process is analogous to the 'test/train' methodology found in machine learning. It's still possible to estimate these parameters from the same data that is being evaluated, as will be shown below.

Figures 1 and 2 show some examples of typical control charts. Figure 1 shows a typical x-bar chart which assumes a normal distribution for the measurements made. Figure 2 shows an example of a p-chart (or proportion chart) which tracks a binary variable over time. See the figure captions for details.

As the analyst works through the quality control charting process a number of decisions need to be made. In this instance we can assume that the central/spread parameters will be estimated from the data and not predefined.

1. How will the data be grouped (e.g. will the analysis be grouped be weeks or by months)?
2. How will estimates of central tenancy and spread be calculated. This is often tied to the distribution assumed for the process.
3. What are the rules that indicate a process is 'out of control'. This can be seen as finding the balance between type I and type II error.

2.2 Tools Used

The `qcc` package (Scrucca 2017) is used to perform all statistical charting control in this section. Additionally, see Software Information in the Appendix for information about the other tools used. Lastly, the full code for this project can be found on github.

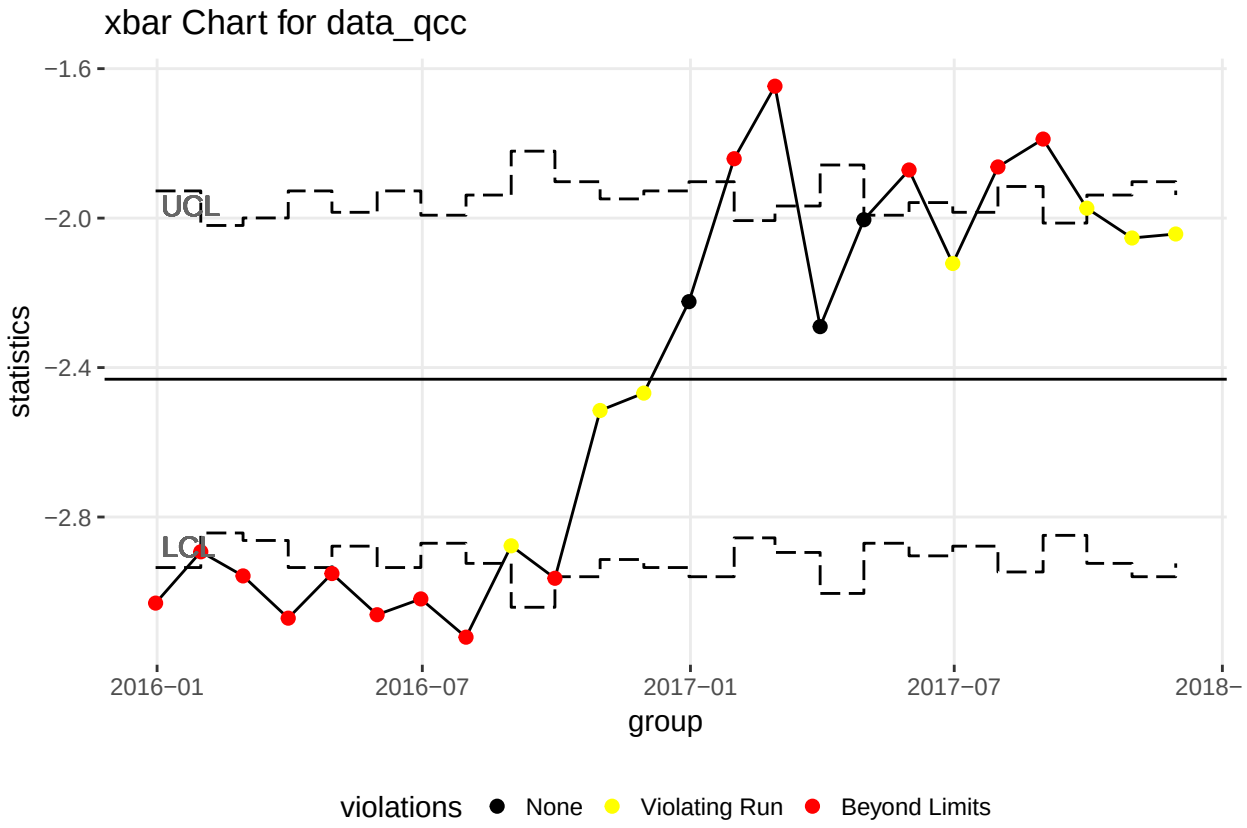

Figure 1: The chart above shows a typical x-bar control chart. It's clear that there is a process change in this example. Note that points outside of the limits are labeled as 'Beyond Limits' while the points that are greater than 6 points on one side of the center line are labeled as 'Violating Runs'.

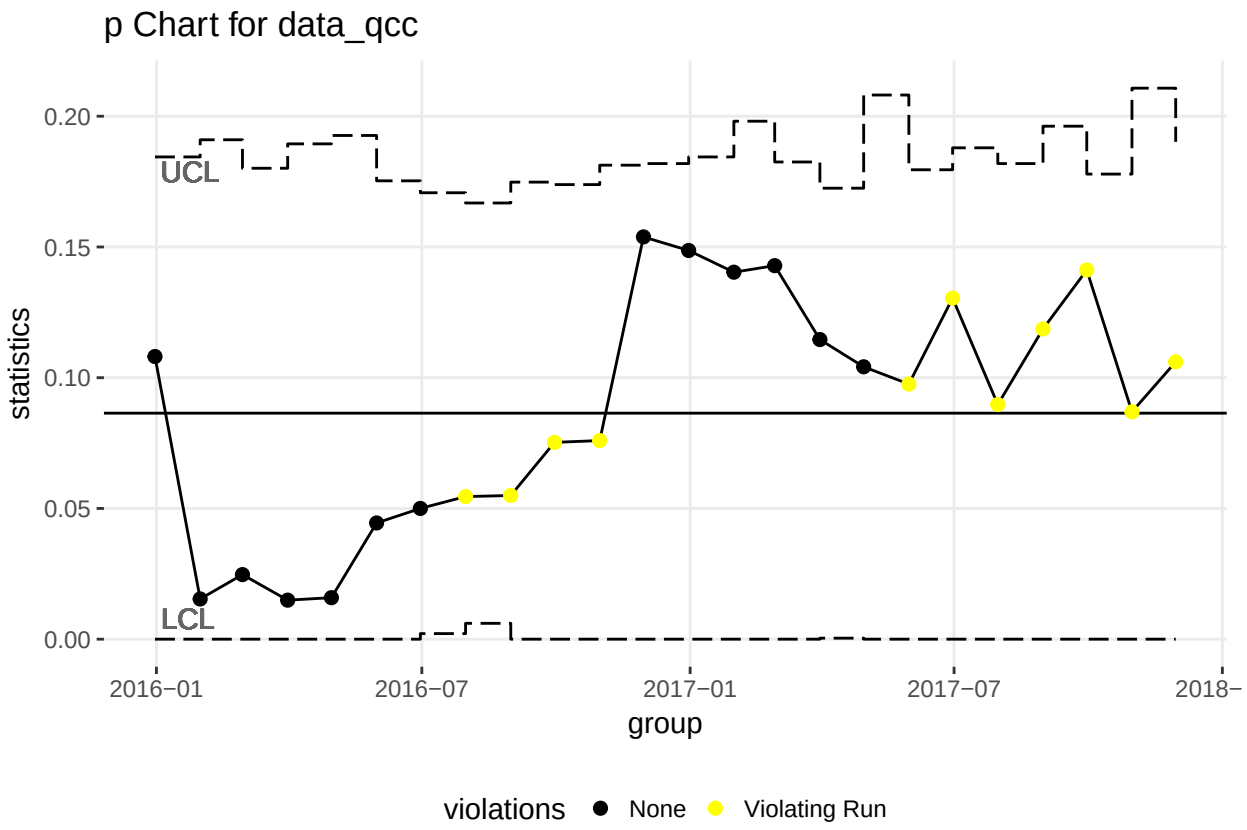

Figure 2: The graph above shows a typical p chart. Note that this figure (unlike figure 1) does not contain any points that are 'Beyond Limits'. Nevertheless there is a clear trend upward which is captured by the 'Violating Run'.

3 Data Simulation

3.1 Overall Data Properties

The main data of interest is some measurement, either continuous or binary, taken at a specific point in time. The range of time is taken as January 1, 2016 - December 31, 2017. The data generating process first picked specific dates (with replacement and according to the parameters listed below) and then assigned those dates the random measurements.

3.2 Knobs

Data generation was done in the spirit of what was completed by (Dorie et al. 2017) where different 'knobs' were defined. These knobs represent distinct features of the data and varying the values for them will create slightly different data sets.

3.2.1 Knobs Controlling Data/Observation Frequency

3.2.1.1 Data Sparsity

The domain of data sparsity is: $x : (0, 1)$.

This parameter ranges from 0 to 1 and describes the number of days in the full range that had an observation. This parameter is used to simulate data that happens rarely, such as rare surgeries. See below in Observations Per Day and Figure 3 for a longer explanation.

3.2.1.2 Observations Per Day

The domain of observations per day is: $x : (0, Inf)$.

This describes a slightly different data property from Data Sparsity in that the days for observation are previously chosen and then a random number (averaging the parameter) is assigned to each of those days. Figure 3 displays these two parameters a they very. See the caption for a deeper look.

3.2.1.3 Data Density (Calculated)

The data density parameter is the result of multiplying Data Sparsity by Observations Per Day. This value will approximate the total observations / 730 days.

3.2.2 Knobs Controlling the Underlying Change

3.2.2.1 Process Start

The domain of process start is: $x : (0, 1)$.

This parameter defines when a process change begins as a percentage of the observations from the beginning of the data set. See Figure 4 for further details.

3.2.2.2 Process Change

The domain of process change is: $x : (0, 1 - process\_start)$.

This parameter defines how long the process will change for. See Figure 4 for further details.

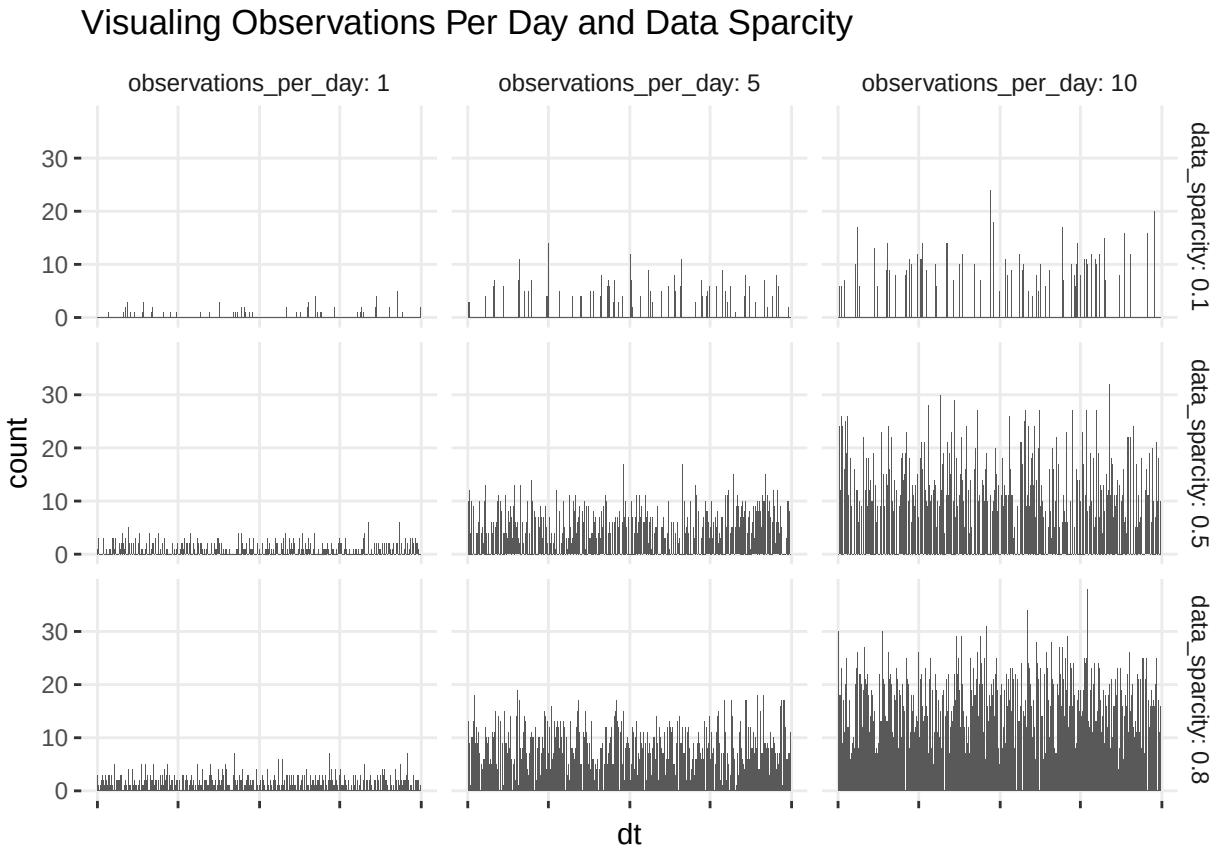

Figure 3: The figure above shows a histogram of observations taken across the two parameters described in Observations Per Day and Data Sparsity. It's important to note the difference between having sparse data with many observations (upper right) and dense data with few observations with few observations (lower left). The data looks different but it has very similar Data Density values suggesting a similar number of observations.

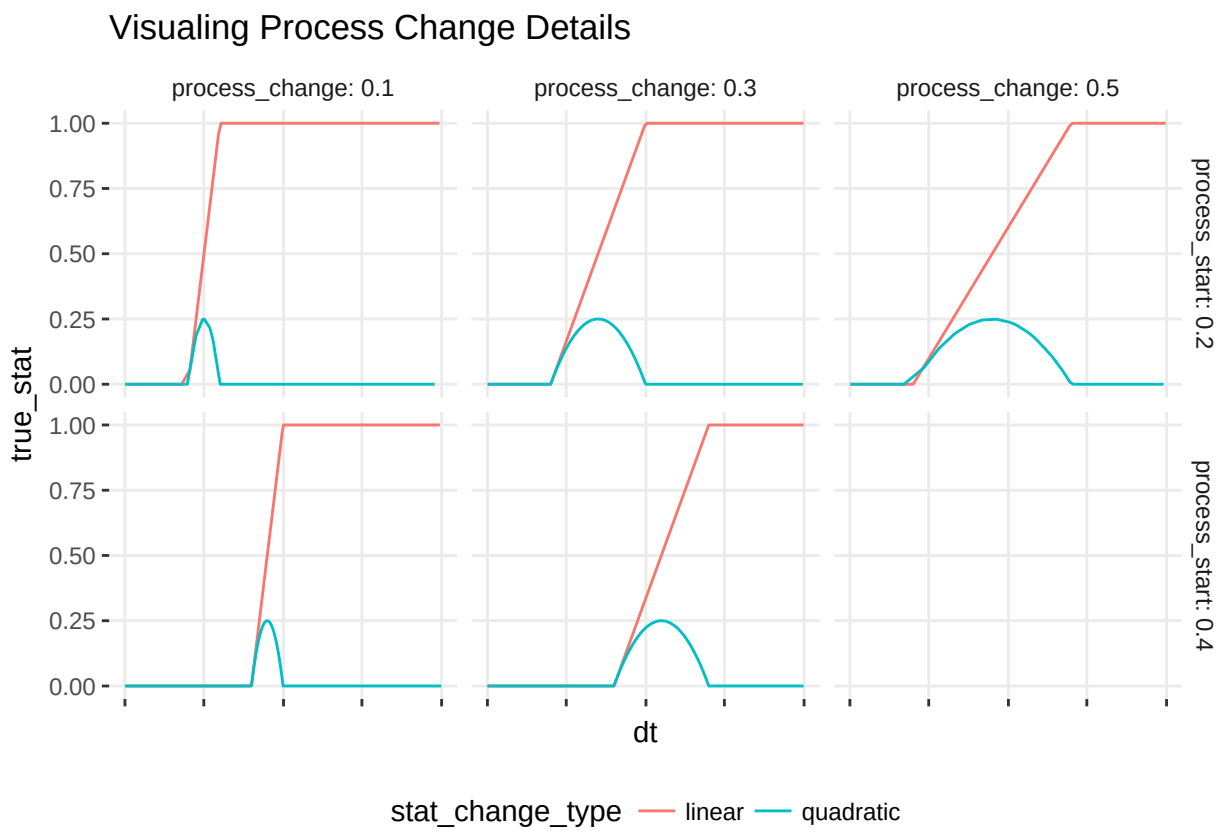

Figure 4: The figure above explains some of the differences in parameters between Process Start, Process Change and Stat Change Type. The graphs show the true value of the statistic of interest. Additionally the magnitude of the change shown below is the same across facets. It's clear that the process change variable modifies the length of the change and the shape of the process change is dictated by the Stat change type variable.

3.2.2.3 Stat Change Type

The domain of stat change type is: $x : (linear, quadratic)$.

This parameter defines what kind of change will occur. A linear change will keep the change while the quadratic change will return back to the normal value. See Figure 4 for further details.

3.2.3 Knobs Controlling the True Values

3.2.3.1 Data Type

The domain of data type is: $x : (measurement, rate)$.

The type of data dictates the underlying distribution of the data which informs how the simulated sample is drawn. If its value is `measurement` then individual values are drawn from the normal distribution. This requires a standard deviation as well which is defined by SD Perc (see below). Values for each observation are then drawn from the distribution .

If the value is `rate` then a similar process happens but with the Bernoulli distribution. First, the current value of the response variable is transformed from $(-Inf, Inf)$ to $(0, 1)$ using the inverse logit operator (from the `boot` package (Canty and Ripley 2017)). This value then becomes the probability with which a Bernoulli trial is sampled.

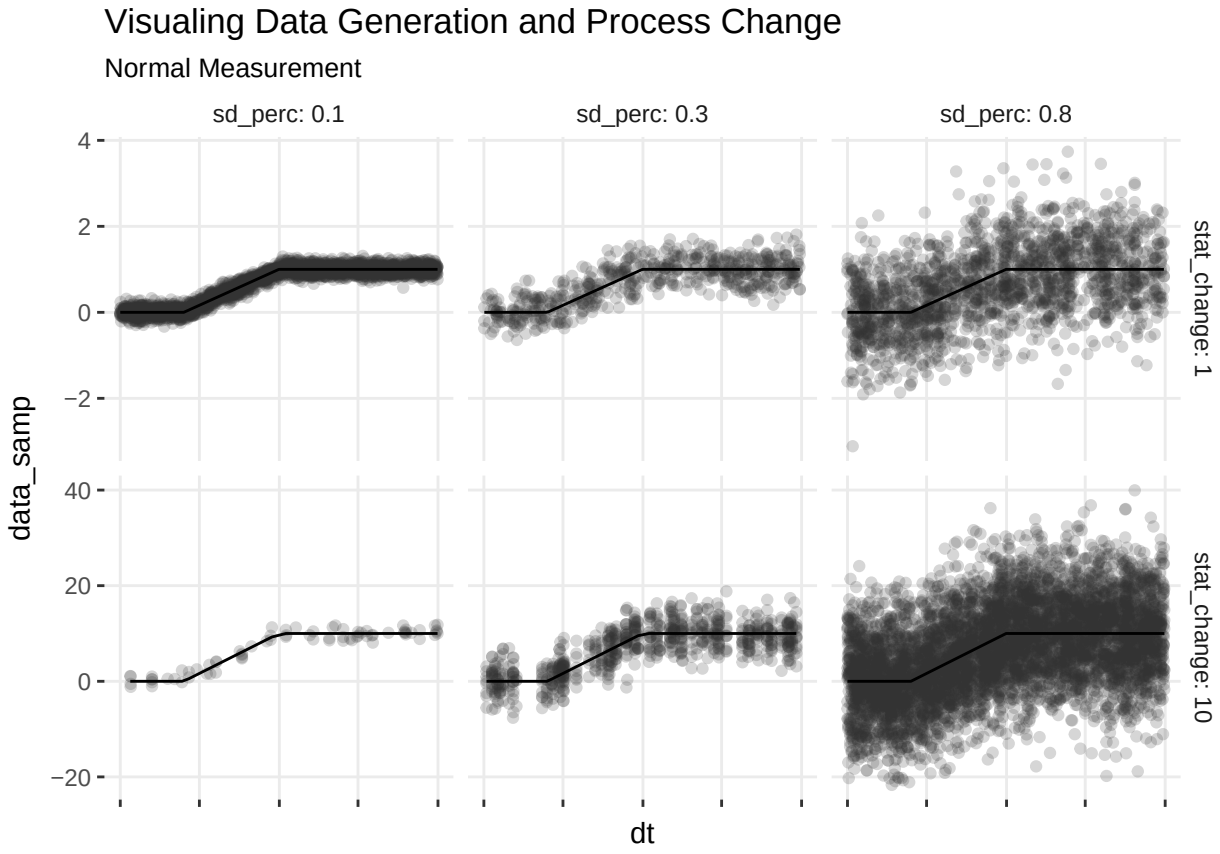

Figure 5: The figure above shows a sample process for a normal measurement. Note that the scale for each row is not the same. Nevertheless the sd_perc describes a similar scale

See Figure 5 for details on the process in regards to normal measurements.

3.2.3.2 Stat Start

The domain of stat start is: $x : (-Inf, Inf)$. This defines the starting/baseline value for the response variable. The definition is made prior to any transformations (such as inverse logit).

3.2.3.3 SD Perc

The domain of SD Perc is: $x : (0, 1)$ and is only applicable when data_type = 'measurement'. It is necessary to define the spread of points when dealing with

3.2.4 Knobs Controlling CC Parameters

3.2.4.1 Date Floor

The domain of Date Floor is $x : (week, month, quarter)$. It represents the grouping variables that are used in the control chart. This 'width' is analogous to a bin width when thinking of a histogram. If `week` is selected there are ~104 groups, while if `quarter` is selected there are only 8 groups.

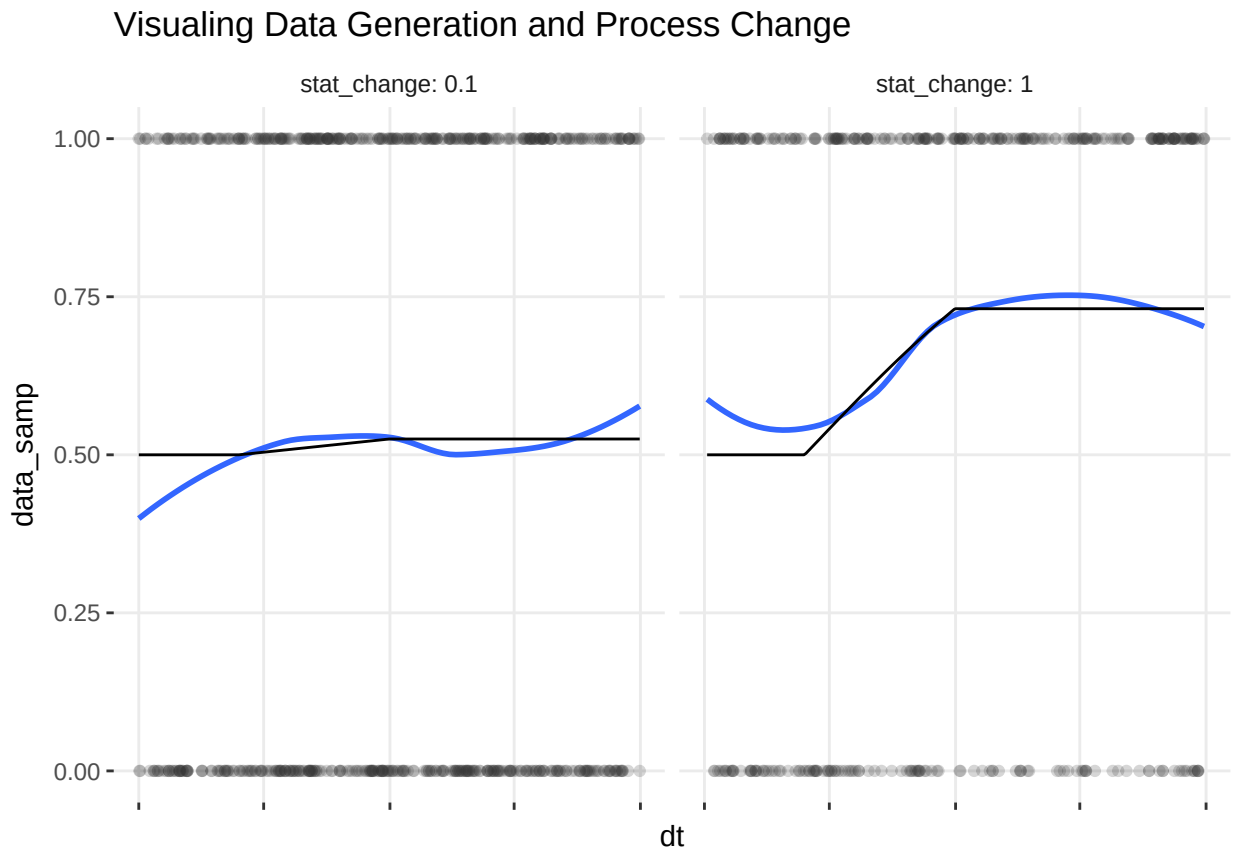

Figure 6: The figure above shows an example a process change for the rate variable. The black line shows the true probability that was used for sampling while the blue line shows a smoothed average based off the the data points (also plotted at 0 and 1 of the graph.)

3.2.5 Actual Parameters Used

While the domains above represent the theoretical possible values for the knobs, only a few were chosen. All combinations of the parameters below were used excluding those that became invalid (i.e. those where the process_change + process_start > 0.9). This process lead to 12,960 different combinations of the parameters to be evaluated.

See the Parameter Grid Creation section of the appendix for details on this process.

- **data_sparcity**: *0.1*, *0.5* and *0.8*
- **observations_per_day**: *1*, *5* and *10*
- **process_start**: *0.2* and *0.4*
- **process_change**: *0.1*, *0.3* and *0.5*
- **stat_start**: *-3*, *0* and *3*
- **stat_change**: *0*, *0.1*, *1* and *10*
- **sd_perc**: *0.1*, *0.3*, *0.8* and *NA*
- **stat_change_type**: *linear* and *quadratic*
- **data_type**: *measurement* and *rate*
- **date_floor**: *week*, *month* and *quarter*

3.2.6 Final Data Generation Process

The final data generation process was completed by seeding and creating 20 sets of data and are included in the github repositroy under the `sim_data` folder. See the Data Generation, Model Fitting and Evaluation section of the appendix for details.

4 Measuring Success

The following sections discuss the main measures of success that will be used to evaluate the success of the control chart methodology. These are discussed in great detail as these could be used as measures of success to empirically evaluate other statistical control methodology.

4.1 Ability to Fit

It is possible for the method developed by the `qcc` package will not complete. This method was developed directly from the standard rules of control charts. This binary outcome will help determine what parameters (whether about the data or the parameters necessary for the method) lead to a successful fit.

The following measures will only be calculated on those versions of the data the successfully fit.

4.2 False Positive Rate

The false positive rate (FPR) is defined as either finding a trend based (6 points in a row on one side of the center line) or an extreme value happening outside of the control limits that occurred during a section where the `true_val` was not different than baseline. A low rate is desirable.

This statistic is a binary value and is only calculated for the models that successfully fit.

4.3 True Positive Rate

The true positive rate (TPR) is defined by having the model accurately predict either a trend change (6 points on either side of the center line) in a linear change model or an extreme value during a quadratic process change.

The distinction was made because, both of these changes should be viewed separately. Control chart theory generally attempts to identify out of control points, whether trend based or extreme value based (Scrucca 2017). Nevertheless, the two types of errors are often attributed to different underlying processes. Namely, trend violations signal a trend change while individual out of control points may signal a random issue that can be 'written off'.

Therefore, for this project the type of change to identify is treated as importantly as the detection of any change.

Additionally only models that successfully fit and those that and had a non-zero process change were evaluated for the true positive statistic as those that don't have any change will not be able to detect a change by definition.

5 Results and Discussion

5.1 Ability of QCC to Fit the Data

One important drawback of the control chart methodology is that there must be at least two points in a group to be able to estimate the standard deviation. Overall, the model was able to fit the data 85.7% of the time. The following discussion refers to Figure 7.

The most important predictor, unsurprisingly is the density of the data. This directly influences. When the data averages about 1 or more (>=0.9) observations/day then the model fits 99% of the time. If the data is not as dense the sampling frequency begins to play an important role. Sampling frequency refers to the width of each group which is decided by the Date Floor knob.

While this is an unsatisfying result of the control chart methodology - it'd be ideal if the model was always able to fit and simply widen confidence intervals accordingly - it's an easily solved issue. The lack of fit can be addressed by removing groups with single observations (although it may result in graphs that bypass those groups). This may be a feature that could be added to the `qcc` package allowing for easier analysis.

5.2 Ability of QCC to correctly identify changes

After appropriately fitting the model, it becomes very important to accurately detect any changes that may be occurring. Figure 8 shows the result of fitting a tree based model to the `true_positive` outcome using the parameters described earlier.

The right side of the figure describes those that had quarter based groups. Therefore, there were only 8 groups analyzed. According to the model, this grouping was very ineffective in detecting linear changes or even changes in rate based data.

The left side of the figure is slightly more complicated improves the base 52% rate of accurate detection to 67%. Finally, looking at continuous data appears to improve the detection rate (left most leaf.) If the data is rate based, there appears to be more fine points to the analysis where the `stat_change` and even the `stat_start` are important in accurate detection. Unsurprisingly, small changes (`stat_change` < 0.55) do not lead to high levels of successful predictions.

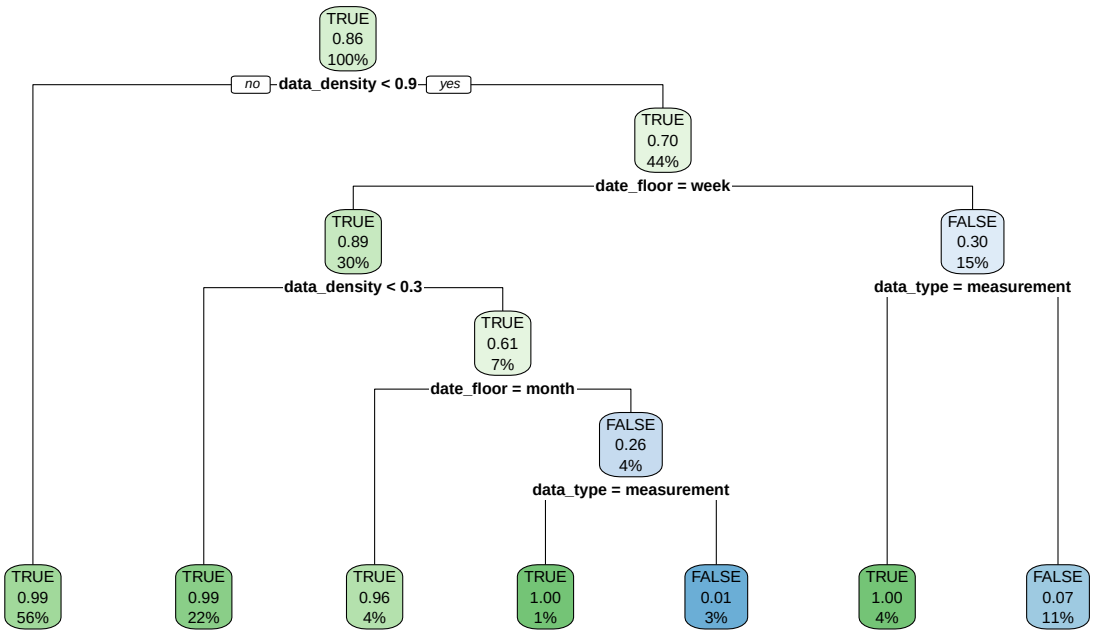

Figure 7: The graph above shows a classification tree model fit using the knobs described in the Knobs section with the dependent variables being `fitted` which is `TRUE` if the control chart model successfully fit. Each square in the graph represents a node. The first line shows the predictor for all data that fall into that node. The second line shows the fraction of data points that fall into the `TRUE` category in that node. The last line shows the proportion of the data that falls into that node. The writing under each node represents a decision, with the affirmative leading to the right. See Ability of QCC to Fit the Data for a deeper discussion on the findings of figure.

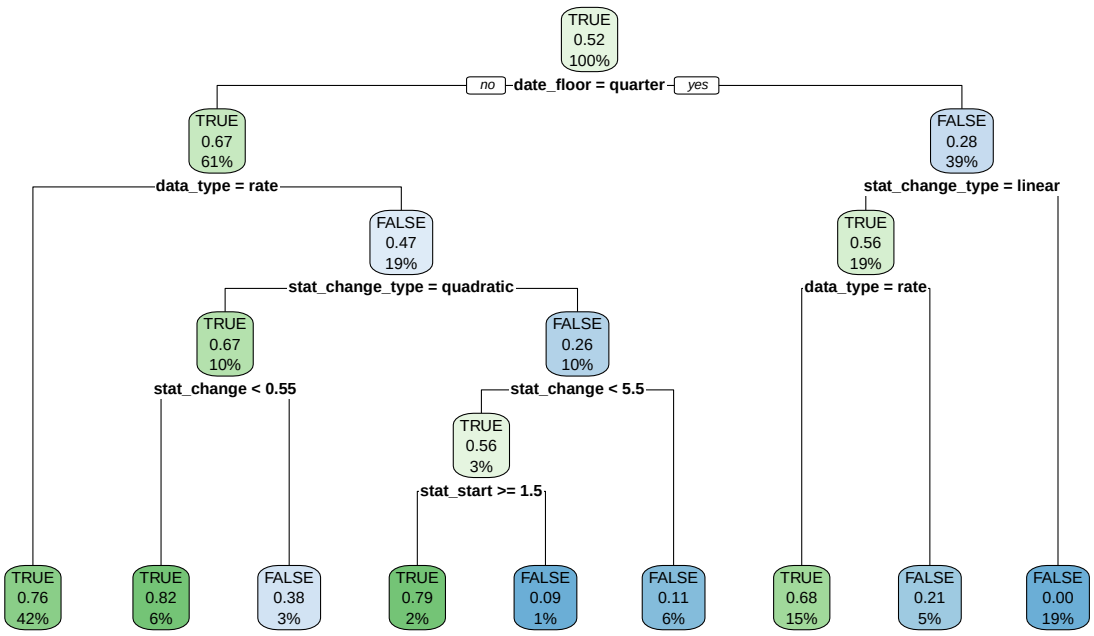

Figure 8: Please see the explanation at figure 7 for details on how to read the graph. The graph above describes the most important parameters in ensuring successful detection of a process change. Only models with an actual change and with a successful fit were included in this analysis.

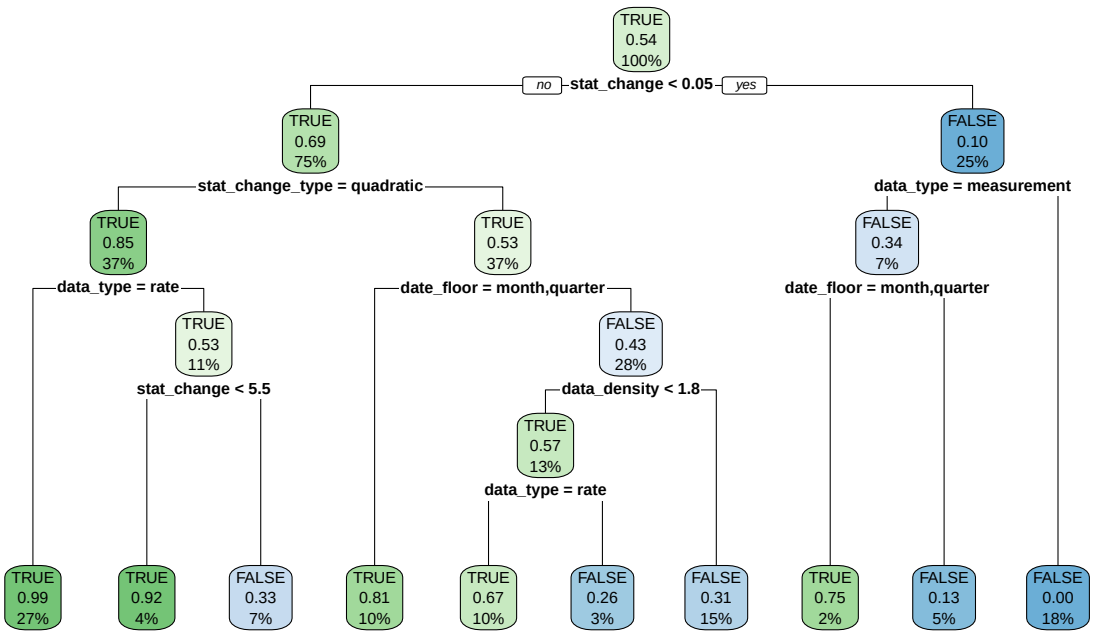

Figure 9: Please see the explanation at figure 7 for details on how to read the graph. The graph above describes the most important parameters in ensuring successful detection of a process change. Only models with a successful fit were included in this analysis.

5.3 Ability of QCC to minimize False Positive results

As important as it is to correctly identify changes, it's important to minimize the number of false positives that are detected. Please review False Positive Rate for details on how this was calculated.

The results of the model that used `false_positive` as an outcome are shown in 9. Surprisingly, a smaller `stat_change` actually lowered the rate of a false postie. This is surprising because it seems that a large change would most likely be detected. However, the larger change may have been detected outside of the limits of the actual process change. Specifically, a large process change may have thrown the estimated central line so far from baseline that the points outside of the process change actually appeared to be in violation of the process limits. An example of this issue can be seen in Figure 1.

Further down the tree it's interesting to see that the parameterization of the grouping variable to the shortest bin width or the most amount of groups results in the largest rate of false positives. Additionally, it appears that measurement based has a higher rate of false positives for linear trend changes.

6 Conclusion and Next Steps

6.1 Discussion

The ability of control charts to appropriately fit are closely related to some of the simple parameters that an analyst has access to. It clearly is related to the bin width chosen by the analyst. These findings follow the general recommendations That there should be ~ 20 groups on a statistical control chart to accurately estimate changes.

These parameterizations are also important to minimize the false positive rate. This makes sense because the more groups lead to more independent tests which leads to a higher chance of false positives.

The full analysis shows that there are some parametric decisions that are needed to be made correctly by the analyst.

6.2 Next Steps

Some possible next steps are outlined to continue exploration of this concept.

6.2.1 Other Knobs

One way to expand the analysis is to create other ways to change the data. This can add more complexity to the analysis and test the robustness of any change method.

1. Auto-correlation Degree
   - Add some degree of auto-correlation to the measurements. Control chart methodology needs to assume that all points are independent. Auto-correlation would break this assumption and may change the ability of this method to appropriately identify change.
2. Cyclical Components
   - It's possible to add some cyclical components which may be present in seasonal data.
3. Count Data Type (using Poisson Distribution)
   - Control chart methodology also allows for analyzing count data. This can be an additional data type included.
4. Adding covariates
   - It would be interesting to add some covariates which can be used to test the reason for trend changes.

6.2.2 Additional Methods

Besides control charts, there are many other methods that may be employed to detect changes and may be more adept at dealing with some of the new details listed in the 'Knobs' section.

For example, it's possible to use forecasting (such as with the `forcast` package) to try to predict points and then detect issues. This package can deal with built in trends automatically. Additionally, it's possible to use more non-parametric methods such as structural change detection (`strucchange` package) which does not rely on any parametric assumptions.

The hope is that the framework developed in this paper may be easily expanded to include other methods. The metrics used can be applied to any change detection method and evaluated for the accuracy and robustness.

7 Appendix

7.1 Packages

```
# Data Cleaning/Manipulation
library(dplyr)
library(tidyr)
library(magrittr)
library(broom)

# Statistical Calculations
library(qcc)
library(caret)
library(boot)
library(rpart)

# Data Presentation
library(ggplot2)
library(pander)
library(rpart.plot)

# Other Utilities
library(lubridate)
library(fs)
library(purrr)
library(future)
plan(multiprocess)
library(formula.tools)

ggplot2::theme_set(
  theme_minimal() %+replace%
    theme(
      legend.position = "bottom",
      panel.grid.minor = element_blank(),
      axis.ticks = element_line(colour = "grey20")
    )
)
```

7.2 Main Functions Used

```
generateData <- function(data_sparcity = 0.1,
                         observations_per_day = 5,
                         process_start = .4,
                         process_change = .2,
                         stat_start  = 0,
                         stat_change = 1,
                         stat_change_type = "linear",
                         data_type  = "measurement",
                         date_floor = "quarter",
                         sd_perc = 0.1,
                         cutoff = NA,
```

```
                        covariates,
                        seed_num = NA,
                        ...) {
if (!is.na(seed_num))
  set.seed(seed_num)


# global options
start_date <- ymd('2016-01-01')
end_date <- ymd('2017-12-31')
date_range <- seq(start_date, end_date, by = 1)
total_dates <- length(date_range)

dates_to_sample <- sample(size = round(data_sparcity * total_dates),
                          x = date_range,
                          replace = FALSE)

total_observations <- round(data_sparcity * total_dates * observations_per_day)
final_dates <- sample(x = dates_to_sample,
                      size = total_observations,
                      replace = TRUE)

if(!missingArg(covariates)){
  d <- covariates %>%
    sample_n(total_observations,
    replace = TRUE) %>%
    mutate(dt = final_dates) %>%
    select(dt, everything())
}else{
  d <- data.frame(dt = final_dates)
}

# Change process definition
if(!between(process_start, .1, .9)) {
  stop("Please provide valid process_start")
}
if (!between(process_change, 0, 1 - process_start)) {
stop("Please provide valid process_change")
}

process_end_change <- process_start + process_change
process_dates <- quantile(as.numeric(date_range),
                          c(process_start, process_end_change))

process_length <- process_dates[2]-process_dates[1]


d <- d %>%
  mutate(dt_int = as.numeric(dt)) %>%
  mutate(dt_group = floor_date(dt, unit = date_floor)) %>%
  mutate(in_process = between(dt_int, process_dates[1],
                              process_dates[2])) %>%
  mutate(baseline = stat_start)
```

```r
  # Process Change Functions
  if (stat_change_type == "linear") {
    d <- d %>%
      mutate(
        true_stat = case_when(
          dt_int < process_dates[1] ~ baseline,
          in_process ~ baseline + stat_change * (dt_int - process_dates[1]) /
            process_length,
          dt > process_dates[2] ~ baseline + stat_change
        )
      )
  } else if (stat_change_type == "quadratic") {
    d <- d %>%
      mutate(
        true_stat =
          case_when(
            dt_int < process_dates[1] ~ baseline,
            in_process ~ baseline - (dt_int - process_dates[1]) *
              (dt_int - process_dates[2]) * stat_change /
              (process_length ^ 2) ,
            dt_int > process_dates[2] ~ baseline
          )
      )

  }

  # Apply Link Functions and Create Data
  if(data_type == "measurement"){
    d <- d %>%
      mutate(data_samp = rnorm(n = nrow(.), mean = true_stat, sd = stat_change*sd_perc))
  }else if(data_type == "rate"){
    d <- d %>%
      mutate(true_stat = boot::inv.logit(true_stat),
             data_samp = rbinom(n = nrow(.), size = 1, prob = true_stat))
  }

  d <- d %>%
    arrange(dt)

  attr(d, "data_type") <- data_type
  attr(d, "stat_change_type") <- stat_change_type
  attr(d, "cutoff") <- cutoff

  d

}

createQCC <- function(d, debug = FALSE){
  data_type <- attr(d, "data_type")
  cutoff <- attr(d, "cutoff")
  type <- case_when(
    data_type == "rate" ~ "p",
    data_type == "measurement" ~ "xbar"
```

```
  )

  if(is.na(cutoff)) {
    out <-
      try(qcc.formula(data_samp ~ dt_group, data = d, type = type),
          silent = !debug)
  } else{
    out <-
      try(qcc.formula(
        data_samp ~ dt_group,
        data = d,
        cutoff = cutoff,
        type = type
      ),
      silent = !debug)
  }

  if(inherits(out, "try-error")) out <- NA

  out
}

evalQCC  <- function(d, qcc_obj,
                     invalid_runs = c("Violating Run", "Beyond Limits"), ...){
  if (is.na(qcc_obj)[[1]]) {
    return(data.frame(
      false_positive = NA,
      true_positive = NA,
      arl = NA
    ))
  }

  stat_change_type <- attr(d, "stat_change_type")

  true_stats <- d %>%
    group_by(dt_group) %>%
    summarise(true_stat = mean(true_stat),
              in_process = any(in_process)) %>%
    mutate(diff_stat = true_stat != first(true_stat))


  calc_df <-  augment.qcc(qcc_obj, gr_func = as.Date) %>%
    rename(dt_group = group) %>%
    mutate(dt_group = dt_group + days(1)) %>%
    left_join(true_stats, by = "dt_group")


  emp_process_start <- calc_df %>%
    filter(diff_stat) %>%
    top_n(-1, dt_group) %>%
    pull(dt_group)

  first_detect <- calc_df %>%
```

```
    filter(diff_stat, violations %in% invalid_runs) %>%
    top_n(-1, dt_group) %>%
    pull(dt_group)

  arl <- ifelse(
    length(first_detect) == 1,
    as.numeric(first_detect - emp_process_start, unit = "days"),
    Inf
  )

  calc_df %>%
    summarise(
      false_positive = any(!diff_stat & violations %in% invalid_runs),
      true_positive = case_when(
        stat_change_type == "linear" ~ any(diff_stat &
                                             violations %in% "Violating Run"),
        stat_change_type == "quadratic" ~ any(diff_stat &
                                                violations %in% "Beyond Limits")
      ),
      arl = arl
    )
}

evalAllQcc <- function(p_grid) {
    p_grid %>%
      mutate(data = pmap(., generateData)) %>%
      mutate(qcc_obj = map(data, ~ createQCC(.x))) %>%
      mutate(qcc_eval = map2(data, qcc_obj, ~ evalQCC(.x, .y))) %>%
      select(-data, -qcc_obj) %>%
      unnest(qcc_eval)
  }
```

7.3 Parameter Grid Creation

```
# Set Up Full Grid
param_grid <- expand.grid(
  data_sparcity = c(.1, .5, .8),
  observations_per_day = c(1, 5, 10),
  process_start = c(.2, .4, .8),
  process_change = c(.1, .3, .5),
  stat_start  = c(-3,0,3),
  stat_change = c(0, .1, 1, 10),
  sd_perc = c(NA, .1, .3, .8),
  stat_change_type = c("linear", "quadratic"),
  data_type  =c("measurement", "rate"),#, "count"),
  date_floor = c("week","month", "quarter"),
  #cutoff = c(as.Date('2017-01-01'), as.Date(NA)),
  stringsAsFactors = FALSE
) %>%
  as_tibble()

param_grid <- param_grid %>%
```

```
  filter(process_change + process_start < .9) %>%
  filter((data_type == "measurement" & !is.na(sd_perc)) |
           (data_type %in% c("rate", "count") & is.na(sd_perc)))
```

7.4 Data Generation, Model Fitting and Evaluation

Most of the important functions from below are defined above in Main Functions Used

```
runs <- 0

## The number of new data sets to create is defined as a
## knitr parameter (default: 0)

# The code below uses the future package to speed up computation using
# multiple processes.

while( runs < params$new_data_sets){
  current_run <- as.numeric(now())
  set.seed(current_run)

  all_data <- param_grid %>%
    # sample_n(16) %>%
    mutate(partition = rep_len(1:4, nrow(.))) %>%
    group_by(partition) %>%
    nest(.key = "p_grid") %>%
    mutate(full_eval = map(p_grid, function(x){future({evalAllQcc(x)})})) %>%
    mutate(full_eval_out = map(full_eval, value)) %>%
    select(full_eval_out) %>%
    unnest()

  all_data <- all_data %>%
    mutate(fitted = !is.na(false_positive))

  save(all_data, file = paste0("sim_data/", current_run*100000,".Rdata" ))

  runs <- runs + 1
}


all_data <- dir_ls("sim_data") %>%
  map_df(function(x){
    ds_name <- load(x)
    get(ds_name)
  }, .id = "strap")

all_data <- all_data %>%
  mutate(data_density = data_sparcity * observations_per_day)
```

7.5 QCC Helper Functions

```r
#' Augment for qcc
#'
#' @param x a qcc object
#' @param include_center should a column for the centerline be included?
#' @param include_stddev should a column for the standard deviation be included?
#' @param gr_func function to apply to group labels, useful for dates or
#'   numbers. Set this to as.numeric or as.Date when your groups are
#'   numbers/dates respectively
#' @param ...
#'
#' @return a dataframe for every row in original data along with control chart
#'   calculations
#' @export
#'
#' @examples
#' library(broom)
#'
#' data("pistonrings", package = 'qcc')
#' diameter <- qcc::qcc.groups(pistonrings$diameter, pistonrings$sample)
#' qcc_obj <- qcc::qcc(diameter[1:25,], type="xbar")
#' augment(qcc_obj)
#'
#' qcc_new <-  qcc::qcc(diameter[1:25,], newdata = diameter[24:40,], type="xbar")
#' augment(qcc_new)
augment.qcc <- function(x,
                        include_center = TRUE,
                        include_stddev = FALSE,
                        gr_func = I, ...){

  if(!(x$type %in% c("xbar", "p", "np", "u")))
    warning("Control Chart Type ", x$type,
            " not directly supported by augment.qcc, use with care!")

  variable_limits <- nrow(x$limits)!=1

  calibrate_data <- data.frame(
    group = names(x$statistics),
    statistics = as.numeric(x$statistics),
    type = "calibrate",
    data_name = x$data.name,
    sizes = x$sizes,
    stringsAsFactors = FALSE
  )


  if(!is.null(x$newdata)){
    new_data <- data.frame(
      group = names(x$newstats),
      statistics = as.numeric(x$newstats),
      type = "newdata",
      data_name = x$newdata.name,
      sizes = x$newsizes,
      stringsAsFactors = FALSE
```

```
    )

    all_data <- rbind(calibrate_data, new_data)

  }else{
    all_data <- calibrate_data
  }

  all_data <- cbind(all_data, x$limits)
  all_data

  if(include_center) all_data$center <- x$center
  if(include_stddev) all_data$stddev <- x$std.dev


  all_data$violations <- "None"
  all_data$violations[x$violations$violating.runs] <- "Violating Run"
  all_data$violations[x$violations$beyond.limits] <- "Beyond Limits"

  # make sure all possibilites are available so that level order is predictable
  all_data$violations <- factor(all_data$violations,
                                levels = c("None", "Violating Run", "Beyond Limits"))

  all_data$group <- gr_func(all_data$group)

  all_data

}

#' plot qcc using ggplot
#'
#' @param x a qcc object
#' @param add.stats currently not used
#' @param chart.all currently not used
#' @inheritParams qcc::plot.qcc
#' @inheritParams augment.qcc
#' @param ... currently not used
#'
#' @return a ggplot2 plot object
#' @export
#' @import ggplot2
#' @examples
#'
#' data("pistonrings", package = 'qcc')
#' diameter <- qcc::qcc.groups(pistonrings$diameter, pistonrings$sample)
#' qcc_obj <- qcc::qcc(diameter[1:25,], type="xbar")
#' plot_gg.qcc(qcc_obj)
#'
#' qcc_new <-  qcc::qcc(diameter[1:25,], newdata = diameter[24:40,], type="xbar")
#' plot_gg.qcc(qcc_new)
#'
plot_gg.qcc <- function(x, add.stats = TRUE, chart.all = TRUE,
                        label.limits = c("LCL ", "UCL"),
```

```
                      gr_func = as.numeric,
                      ...){

augmented <- augment.qcc(x, include_center = FALSE, include_stddev = FALSE,
                         gr_func = gr_func)

calibrated <- length(unique(augmented$data_name)) == 2
if(calibrated){
  calibration_break <- with(augmented, max(group[type=="calibrate"]))
}
min_grp <- min(augmented$group)
max_grp <- max(augmented$group)
first_lcl <- augmented$LCL[1]
first_ucl <- augmented$UCL[1]


g_title <- paste0(x$type, " Chart for ", x$data.name)

p <- ggplot(augmented, aes(x = group, y = statistics)) +
  geom_line() +
  geom_step(aes(y = UCL), linetype = "longdash") +
  geom_step(aes(y = LCL), linetype = "longdash") +
  geom_hline(yintercept = x$center) +
  geom_point(size = 2, aes(color = violations)) +
  scale_color_manual(breaks = c("None", 'Violating Run', 'Beyond Limits'),
                     values = c("black", "yellow", "red")) +
  labs(
    title = g_title
  )


if(calibrated){
  p <- p + geom_vline(xintercept = as.numeric(calibration_break),
                      linetype = "dotted")
}


p <- p +
  geom_text(
    x = as.numeric(min_grp),
    y = first_ucl,
    label = label.limits[2],
    hjust = -.1,
    vjust = 1.2,
    color = "grey40"
  ) +
  geom_text(
    x = as.numeric(min_grp),
    y = first_lcl,
    label = label.limits[1],
    hjust = -.1,
    vjust = -.5,
    color = "grey40"
```

```
    )

  p

}


#' Formula interface to qcc
#'
#' This function provides a simplified interface to the qcc function. The main
#' goal is to remove the need for using the qcc groups function. This allows for
#' much simpler use of the function within piping functions/map.
#'
#'
#' @param formula a formula describing the outcome variable and the grouping
#'   variable
#' @param data a data frame
#' @param cutoff a cutoff date for new data
#' @param ... further arguments passed to \code{qcc:qcc()}
#' @inheritParams qcc::qcc
#' @return a qcc object
#' @export
#'
#' @examples
#' data("pistonrings", package = 'qcc')
#' diameter <- qcc::qcc.groups(pistonrings$diameter, pistonrings$sample)
#'
#' # Without Cutoff
#' qcc_obj <- qcc::qcc(diameter[1:25,], type="xbar")
#' qcc_obj2 <- qcc.formula(diameter ~ sample,
#'     data = dplyr::filter(pistonrings, sample <= 25),
#'     type = "xbar", plot = TRUE)
#'
#' # With Cutoff
#' qcc_new <-  qcc::qcc(diameter[1:25,],
#'     newdata = diameter[24:40,], type="xbar")
#' qcc_new2 <- qcc.formula(diameter ~ sample, data = pistonrings, type = "xbar",
#'     plot = TRUE, cutoff = 25)
#'
#' # Dates and p chart
#' ## use POSIXct, not Date object to create date
#' start <- as.POSIXct('2017-01-01')
#' end <- as.POSIXct('2017-01-15')
#' samp_dates <- seq(from = start, to = end, length.out = 15)
#' samp_dates <- sample(samp_dates, 100, replace = TRUE)
#' old_process <- samp_dates < '2017-01-08'
#' vals <- ifelse(old_process,rbinom(sum(old_process), 1 , p = 0.5),
#'     rbinom(sum(!old_process), 1,  p = 0.2) )
#' d <- data.frame(samp_dates, vals)
#'
#' ## Run the graphs with plot = TRUE
#' qcc.formula(vals ~ samp_dates, data = d, type = "p", plot = TRUE)
#' qcc.formula(vals ~ samp_dates, data = d, type = "p", plot = TRUE,
```

```r
#'     cutoff = '2017-01-08')
qcc.formula <- function(formula, data, type, plot = FALSE, cutoff = NULL, ...){
  data <- as.data.frame(data)
  outcome_name <- formula.tools::lhs(formula)
  group_name <- formula.tools::rhs(formula)

  outcome <- eval(outcome_name, envir = data)
  group <- eval(group_name, envir =  data)

  # standardize group to be a POSIXct
  if(inherits(group, "Date") | inherits(group, "POSIXct")) {
    group <- as.POSIXct(group)
    if(!is.null(cutoff)) cutoff <- as.POSIXct(cutoff)
  }

  groups <- qcc::qcc.groups(outcome, group)

  if(type %in% c("p", "np", "u")){

    data_qcc <- apply(groups, 1, sum, na.rm = TRUE)
    size_qcc <- apply(groups, 1, function(x) sum(!is.na(x)))
  }else{
    data_qcc <- groups
    size_qcc <- apply(groups, 1, function(x) sum(!is.na(x)))
  }

  if(!is.null(cutoff)){
    group_names <- names(data_qcc)
    if(inherits(cutoff, "POSIXct")){
      group_names <- as.POSIXct(group_names)
    }else{
      if(is.null(group_names)) group_names <- 1:nrow(data_qcc)
      else group_names <- as.numeric(group_names)
    }

    if(all(is.na(group_names))){
      stop("the group names could not be converted to either a number or date")
    }


    sel <- group_names < cutoff

  }

  if(!is.null(cutoff)){

    if(is.matrix(data_qcc)){
      calib <- data_qcc[sel, ]
      newdata <- data_qcc[!sel, ]

    }else{
      calib <- data_qcc[sel]
      newdata <- data_qcc[!sel]
```

```
    }

    qcc_obj <- qcc::qcc(data = calib,
                        sizes = size_qcc[sel],
                        type = type,
                        plot = plot,
                        newdata = newdata,
                        newsizes = size_qcc[!sel],
                        ...)

  }else{
    qcc_obj <- qcc::qcc(data = data_qcc,
                        sizes = size_qcc,
                        type = type,
                        plot = plot,
                        ...)
  }

  qcc_obj
}
```

7.6 Software Information

```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.4
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2      formula.tools_1.7.1 future_1.8.1
##  [4] purrr_0.2.4         fs_1.2.2            lubridate_1.7.4
##  [7] rpart.plot_2.1.2    pander_0.6.1        rpart_4.1-13
## [10] boot_1.3-20         caret_6.0-79        ggplot2_2.2.1
## [13] lattice_0.20-35     qcc_2.7             broom_0.4.4
## [16] magrittr_1.5        tidyr_0.8.0         dplyr_0.7.4
##
## loaded via a namespace (and not attached):
##  [1] magic_1.5-8          ddalpha_1.3.3        sfsmisc_1.1-2
##  [4] splines_3.5.0        foreach_1.4.4        prodlim_2018.04.18
##  [7] assertthat_0.2.0     highr_0.6            stats4_3.5.0
## [10] DRR_0.0.3            yaml_2.1.19          robustbase_0.93-0
```

```
## [13] globals_0.11.0       ipred_0.9-6          pillar_1.2.2
## [16] backports_1.1.2      glue_1.2.0           digest_0.6.15
## [19] colorspace_1.3-2     recipes_0.1.2        htmltools_0.3.6
## [22] Matrix_1.2-14        plyr_1.8.4           psych_1.8.3.3
## [25] timeDate_3043.102    pkgconfig_2.0.1      CVST_0.2-1
## [28] listenv_0.7.0        bookdown_0.7         scales_0.5.0
## [31] gower_0.1.2          lava_1.6.1           tibble_1.4.2
## [34] withr_2.1.2          nnet_7.3-12          lazyeval_0.2.1
## [37] mnormt_1.5-5         survival_2.41-3      evaluate_0.10.1
## [40] operator.tools_1.6.3 nlme_3.1-137         MASS_7.3-49
## [43] dimRed_0.1.0         foreign_0.8-70       class_7.3-14
## [46] tools_3.5.0          stringr_1.3.0        kernlab_0.9-26
## [49] munsell_0.4.3        compiler_3.5.0       RcppRoll_0.2.2
## [52] rlang_0.2.0          grid_3.5.0           iterators_1.0.9
## [55] labeling_0.3         rmarkdown_1.9        geometry_0.3-6
## [58] gtable_0.2.0         ModelMetrics_1.1.0   codetools_0.2-15
## [61] abind_1.4-5          reshape2_1.4.3       R6_2.2.2
## [64] knitr_1.20           bindr_0.1.1          rprojroot_1.3-2
## [67] stringi_1.2.2        parallel_3.5.0       Rcpp_0.12.16
## [70] DEoptimR_1.0-8       tidyselect_0.2.4     xfun_0.1
```

Bibliography

"An Exploration of Quality Control in Banking and Finance." 2012. *International Journal of Business and Social Science* 3 (6). https://pdfs.semanticscholar.org/5b1d/fba36a3b0c79a8c5ac5c7a78abe47cc90dbf.pdf.

Canty, Angelo, and B. D. Ripley. 2017. *Boot: Bootstrap R (S-Plus) Functions.*

Dora, Manoj, Maneesh Kumar, Dirk Van Goubergen, Adrienn Molnar, and Xavier Gellynck. 2013. "Food Quality Management System: Reviewing Assessment Strategies and a Feasibility Study for European Food Small and Medium-Sized Enterprises." *Food Control* 31 (2): 607–16. doi:https://doi.org/10.1016/j.foodcont.2012.12.006.

Dorie, Vincent, Jennifer Hill, Uri Shalit, Marc Scott, and Dan Cervone. 2017. "Automated Versus Do-It-Yourself Methods for Causal Inference: Lessons Learned from a Data Analysis Competition."

Raczynski, Bob. 2013. "Is Statistical Process Control Applicable to Software Development Processes? What is Statistical Process Control?" https://www.stickyminds.com/sites/default/files/article/file/2013/XDD14736filelistfilename1{_

Scrucca, Luca. 2017. *Qcc: Quality Control Charts.* https://CRAN.R-project.org/package=qcc.

Shewhart, Walter A. 1931. *Economic Control of Quality of Manufactured Product.* New York: D. Van Nostrand company, inc. //catalog.hathitrust.org/Record/001115960.

SixSigma. 2016. "Six Sigma Dmaic Process - Control Phase - Statistical Process Control." *Six Sigma DMAIC Process - Control Phase - Statistical Process Control - International Six Sigma Institute.* https://www.sixsigma-institute.org/Six_Sigma_DMAIC_Process_Control_Phase_Statistical_Process_Control.php.

Team, SCAMPI Upgrade. 2006. "Appraisal Requirements for Cmmi, Version 1.2 (Arc, V1.2)." CMU/SEI-2006-TR-011. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8107.

Thor, Johan, Jonas Lundberg, Jakob Ask, Jesper Olsson, Cheryl Carli, Karin Pukk Härenstam, and Mats Brommels. 2007–10AD. "Application of Statistical Process Control in Healthcare Improvement: Systematic Review." *Quality & Safety in Health Care* 16 (5). BMJ Group: 387–99. doi:10.1136/qshc.2006.022194.