Statistical Process Control in Healthcare

Final Project - Masters of Arts - Biostatistics

Vitaly Druker

Spring 2018

Contents

1 Introduction

1.1 General Statistical Control

Statistical process control has been used in the manufacturing industries since the 1920s. They were popularised by Walter A. Shewhart at Bell Labs (Shewhart 1931). Statisical 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 bullseye 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 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 bullseye 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 percision 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 bullseyes 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 (Bin Jumah, René Burt, and Benjamin Buttram 2012) and food control (Dora et al. 2013). The strong adoption of SPC by Six Sigma methodology has not hindered this growth (SixSigma (n.d.)).

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 equiped to deal with the wide variablity of the healthcare field (and some of the other fields mentioned above (Raczynski, n.d.)). Healthcare data can also suffer from missingness, measurement error and issues in consistent collection. All of these concept lead to wide variablity 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 healthcare data. Some drawbacks in the methods will be exposed using simulated data. Finally, two different versions of SPC beyond the classical charts will be explored.

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 paramaters from an initial stage (Phase I) and then apply them to new data (Phase 2). This process is analgaous 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.

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. ill the analysis be grouped be weeks or by months)?
2. How will estimates of central tendancy 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.

3 Data Simulation

3.1 Overall Data Properties

The main data of interest is some measurement, either continous 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.

These are discussed below:

3.2.1 Data Sparcity

The domain of data sparcity 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 1 for a longer explanation.

3.2.2 Observations Per Day

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

This describes a slightly different data property from Data Sparcity 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 1 displays these two parameters a they very. See the caption for a deeper look.

3.2.3 Data Density (Calculated)

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

3.2.4 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 dataset. See Figure 2 for further details.

3.2.5 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 2 for further details.

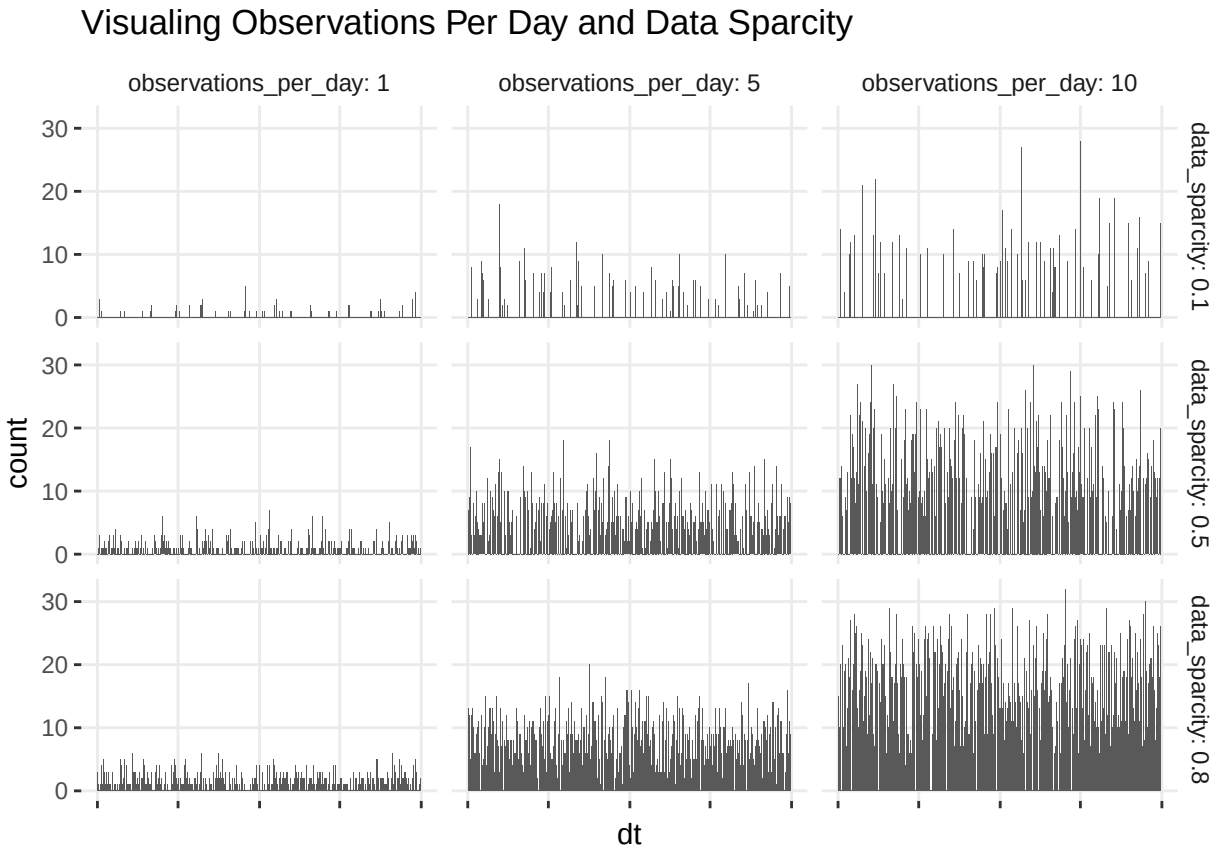


Figure 1: The figure above shows a histograph of observations taken across the two parameters described in Observations Per Day and Data Sparcity. It's important to note the difference between having sparce data with many observations (upper right) and dense data with few observations with few observations (lower left). The data looks different but it has very similar Data Density values suggesting a similar number of observations.

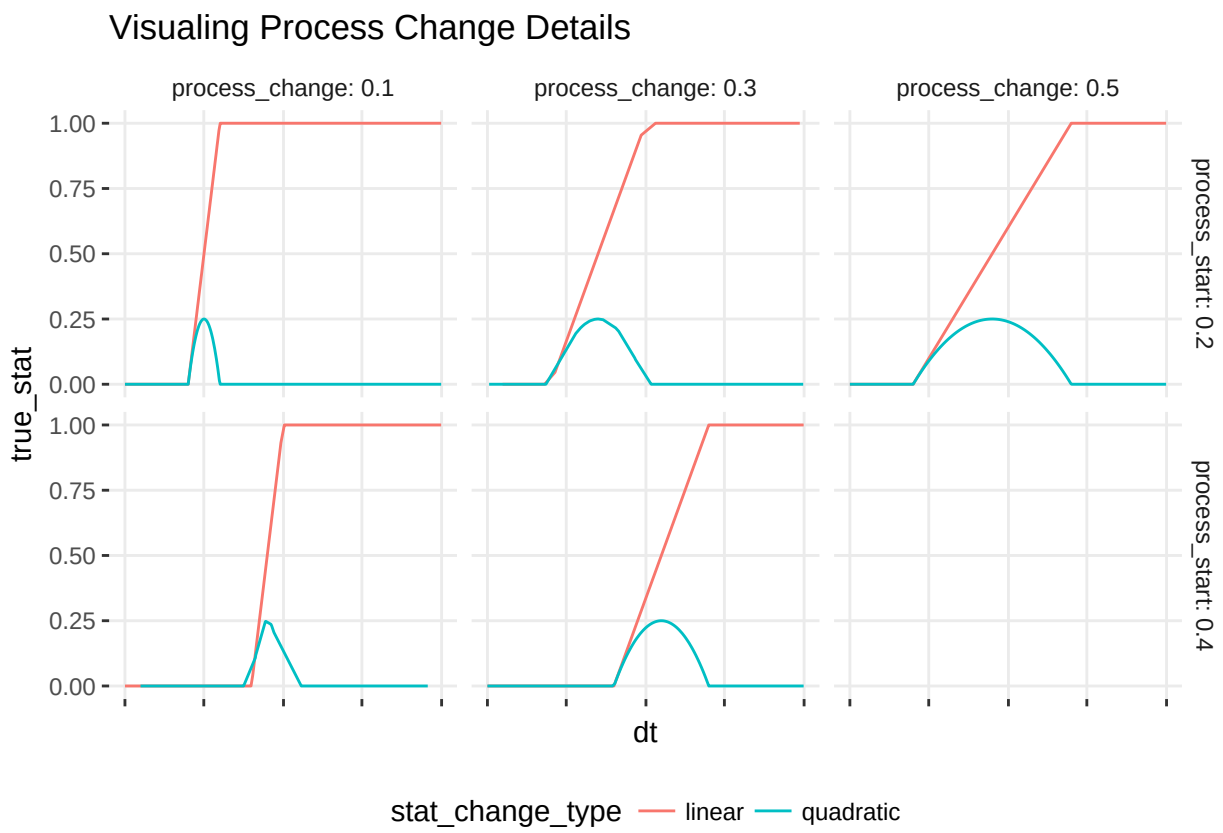


Figure 2: The figure above explains some of the differences in parameters between Process Start, Process Change and Stat Change Type. The graphs show the true value of the statistic of interest. Additionally the magnitude of the change shown below is the same across facets. It's clear that the process change variable modifies the length of the change and the shape of the process change is dictated by the Stat change type variable.

3.2.6 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 2 for further details.

3.2.7 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 $N(true\_val, stat\_change * sd\_perc$.

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 [R-boot]). This value then becomes the probability with which a Bernoulli trial is sampled.

See Figure 3 for details on the process in regards to normal measurements.

```
## `geom_smooth()` using method = 'gam'
```

Figure 3: 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

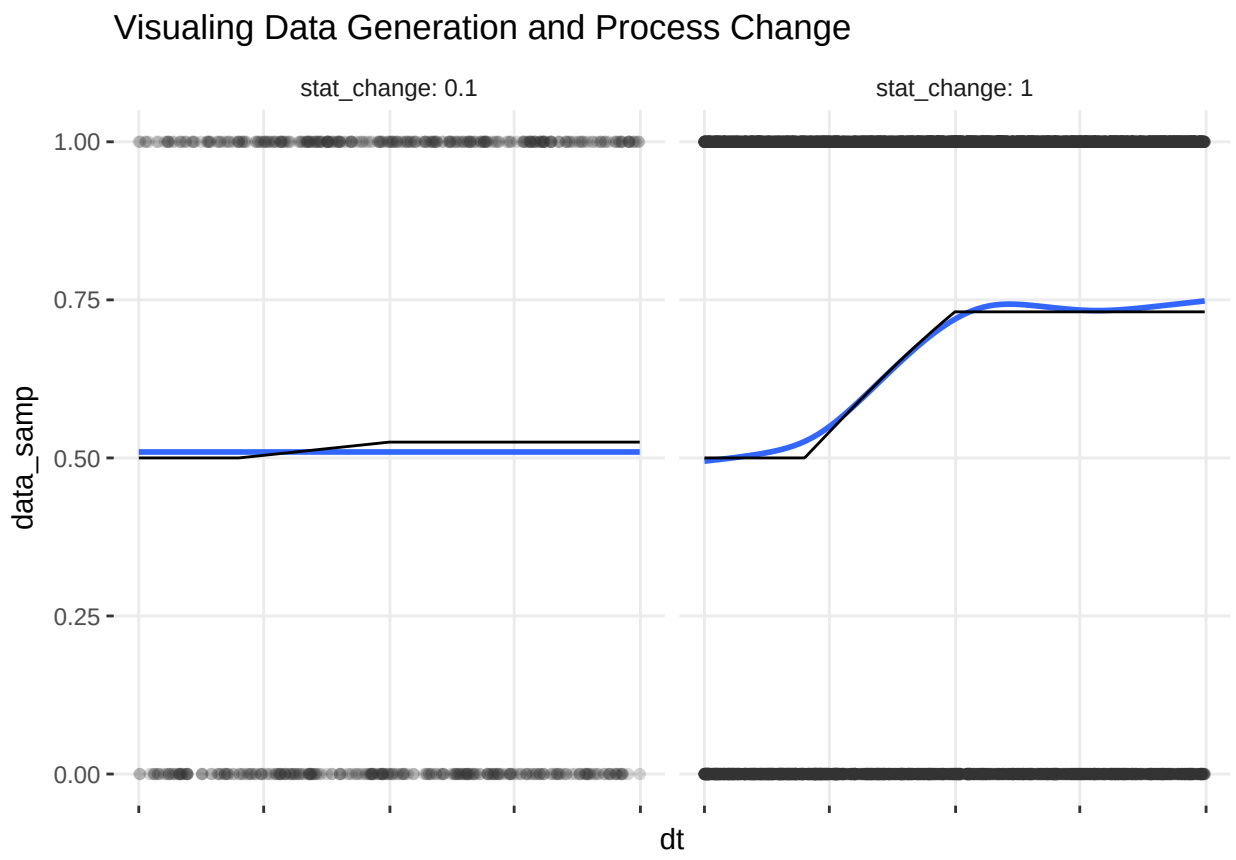


Figure 4: The figure above shows an example a process change for the rate variable. The black line shows the true probability that was used for sampling while the blue line shows a smoothed average based off the the data points (also plotted at 0 and 1 of the graph.)

3.2.8 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).

3.2.9 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

3.2.10 Date Floor

3.2.11 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).

- **data_sparcity**: *0.1*, *0.5* and *0.8*
- **observations_per_day**: *1*, *5* and *10*
- **process_start**: *0.2* and *0.4*

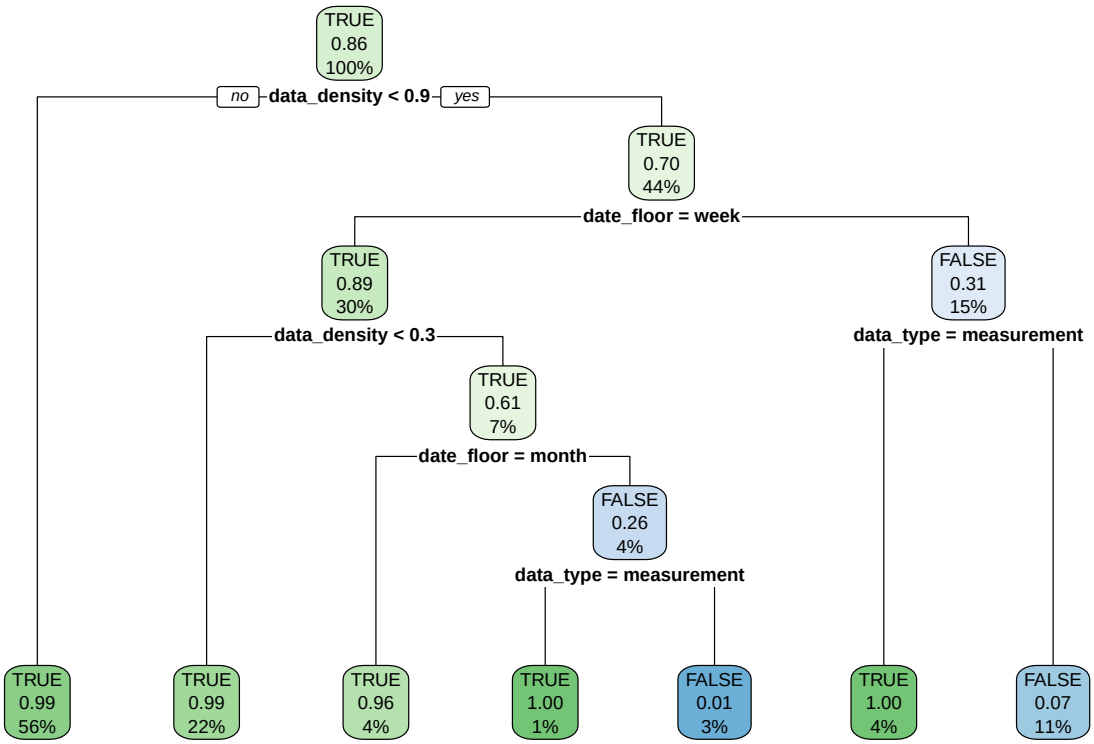
- **process_change**: *0.1*, *0.3* and *0.5*
- **stat_start**: *-3*, *0* and *3*
- **stat_change**: *0*, *0.1*, *1* and *10*
- **sd_perc**: *0.1*, *0.3*, *0.8* and *NA*
- **stat_change_type**: *linear* and *quadratic*
- **data_type**: *measurement* and *rate*
- **date_floor**: *week*, *month* and *quarter*

3.3 Tests

- Detection Rate
- Run Length (RL) (Median and 50% quantile)
- False Positive Rate
- Fitted

4 Results

4.1 Ability of QCC to Fit the Data



4.2 Continous Measurment Analysis

We begin by simulating data for patient wait times

5 Next Steps

Other "Knobs"

1. Autocorrelation Degree
2. Cyclical Comonents 3.1 Amplitude 3.2 Frequency
3. Count Data Type (using Poisson Distribution)
4. Adding covariates

6 Appendix

6.1 Packages

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

# Statistical Calculations
library(qcc)
library(caret)

# Data Presentation
library(ggplot2)
library(pander)

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

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

6.2 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,
                         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)

# 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

  d

}

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

  if(is.null(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
    ))
  }

  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 = any(diff_stat & violations %in% invalid_runs),
      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)
  }
```

6.3 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"),
  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)))
```

6.4 Data Generation, Model Fitting and Evaluation

```
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 %>%
    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)
```

6.5 QCC Helper Functions

```
#' 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)
```

```r
#'
#' 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
```

```r
#' @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 = "
#'
#' # 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 = 
#'
#' # 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 = 
#' 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, 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
}
```

Bibliography

Bin Jumah, Jawaher A, RN P René Burt, and Msie Benjamin Buttram. 2012. "An Exploration of Quality Control in Banking and Finance." *International Journal of Business and Social Science* 3 (6). https://pdfs.semanticscholar.org/5b1d/fba36a3b0c79a8c5ac5c7a78abe47cc90dbf.pdf.

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. n.d. "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. n.d. "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.