AGM Potency Control Charts

Jeffrey R. Weidner

Purpose

Document the **PotCtrlCht.R** script, so that it can be adapted into a Shiny webtool for the Assay Guidance Manual.

This application generates Control Charts for potency data derived from bioassays. Reference compound(s) are tested in each run on an assay and may be tested on one or more plates of the experiment.

Control charts are used to monitor a process in order to demonstrate reproducibility and identify the potential problems which should be investigated. In general control charts track both a measurement and the variability in that measurement. Potency data have a log-Normal distribution, so the data must be transformed to log10 values before performing any statistical analysis. The R package qichart2 will be used for most of the data analysis. The package is available on CRAN, but the source code may be useful for further customization of graphs.

Test Data

A separate script, **PotCCTestData.R**, has been created to generate test potency data as a .csv file. Data files should have a minimum of 2 columns. The first column identifies the Experiment/Run and is recommended that this be a date, though a strings would also be acceptable. (Note the team should discuss requiring this be a date.) The second required column contains the potency measurements for the reference compound. The optional 3rd column would serve to distinguish between replicates run in the same experiment. This column might be useful for labelling points in graphs or as a reference to the user for generated reports.

```
library(tidyverse)
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to

# Seed for reproducible examples remove or change for different random data set
set.seed(19620903)

# Function to generate test data with the same number of replicates per run

tstdata <- function(TrueMeas, TrueMSR, NumRun, NumRep, Dates = TRUE) {

  FileName <- paste('TestData/cc_data', TrueMeas, TrueMSR, NumRun, NumRep, sep = '_' )
  FileName <- paste0(FileName, ".csv")

  TrueMeas <- log10(TrueMeas)
  StdDev <- log10(TrueMSR) / (2 * sqrt(2))
  Run <- if(Dates){
    seq(from = mdy('1/1/2023'), by = '1 week', length.out = NumRun)
  } else {c(1:NumRun)}

  Data <- rnorm(n = NumRun * NumRep, mean = TrueMeas, sd = StdDev)
  Run <- rep(Run, each = NumRep)
  Replicate_ID <- rep(1:NumRep, times = NumRun)

  Output <- if (NumRep == 1)  {
    tibble(Run, Data) %>%
      mutate(Data = 10 ^ Data)
  }  else {
    tibble(Run, Replicate_ID, Data) %>%
    mutate(Data = 10 ^ Data)}

  write_csv(Output, file = FileName)

  Output
}

TestData <- tstdata(TrueMeas = 10, TrueMSR = 3, NumRun = 50, NumRep =1,  Dates = TRUE)
```

The tstdata function generates an ordered set of test data based on the supplied potency (TrueMeas), variability(TrueMSR), number of runs(NumRun), and the number of replicates/run (NumRep). Dates are the default Run identifier.

Application Functions

Functions are primarily derived from the qicharts2 package, which creates the appropriate control charts and stats for individual data versus replicate data. If 6 or more runs are included in the user data, then MSR values are calculated. Currently the 6 run moving window is only implemented for the individual (n=1) data sets. Cummulative MSR's are created for both types of data. The summary of the (qic) object is used to get the summary statistics and rule violations. At this time all graphs and statistics, except MSR, are on the log(10) scale.

```
library(tidyverse)
library(qicharts2)
library(slider)

# Control chart functions --------------------------------
# Charts and Summary stats for individual data

ind_charts <- function(runs, data,usrtitle) {
  IChart <- qic(x = runs,
                y = data,
                chart = 'i',
                xlab = 'Run Date',
                ylab = 'Log10 Potency',
                title = paste0('I Chart ', usrtitle))

  IChartSumm <- summary(IChart)

  MRChart <- qic(x = runs,
                y = data,
                chart = 'mr',
                xlab = 'Run Date',
                ylab = 'Moving Range (Log10 Potency)',
                title = paste0('MR Chart ', usrtitle))

  MRChartSumm <- summary(MRChart)

  Output <- list(IChart = IChart, IChartSumm = IChartSumm, MRChart = MRChart, MRChartSumm
}
```

```
# Charts and data summary for replicate data, n >= 2

rep_charts <- function(runs,data, usrtitle) {
  XbarChart <- qic(x = runs,
                   y = data,
                   chart = 'xbar',
                   xlab = 'Run Date',
                   ylab = 'Mean (Log10 Potency)',
                   title = paste0('Xbar Chart ', usrtitle))

  XbarSumm <- summary(XbarChart)

  SChart <- qic(x = runs,
                   y = data,
                   chart = 's',
                   xlab = 'Run Date',
                   ylab = 'Std. Dev. (Log10 Potency)',
                   title = paste0('S Chart ', usrtitle))

  SSumm <- summary(SChart)

  Output <- list(XbarChart = XbarChart, XBarSumm = XbarSumm, SChart = SChart, SChartSumm =
}

msrchart <- function(runs, data,msrtype, usrtitle) {

  MSRChart <- qic(x = runs,
                  y = data,
                  chart = 'run',
                  xlab = 'Run Date',
                  ylab = msrtype,
                  title = usrtitle)
}

msr <- function(df, startdate, usrtitle) {
  MSRData <- if (length(unique(df$Run)) == length(df)) {
    df %>%
      mutate(Last6_sd = slide_dbl(Log10Pot, sd, .before = 5, .complete = TRUE),
             Last6_MSR = 10 ^ (2 * sqrt(2) * Last6_sd),
             Cumm_sd = slide_dbl(Log10Pot, sd, .before = Inf, .complete = TRUE),
             Cumm_MSR = 10 ^ (2 * sqrt(2) * Cumm_sd)) %>%
      filter(!is.null(Last6_sd))
  } else {
        df %>%
      mutate(Cumm_sd = slide_dbl(Log10Pot, sd, .before = Inf, .complete = TRUE),
             Cumm_MSR = 10 ^ (2 * sqrt(2) * Cumm_sd)) %>%
      group_by(Run) %>%
      summarise(Cumm_MSR = last(Cumm_MSR)) %>%
      ungroup() %>%
      filter(Run > startdate)
  }

  CummMSRChart <- msrchart(runs = MSRData$Run,
                           data = MSRData$Cumm_MSR,
                           msrtype = 'Cummulative MSR',
                           usrtitle = paste0('Run Chart Cummulative MSR ', usrtitle))

  CummMSRSumm <- summary(CummMSRChart)

  Last6MSRChart <- if (length(unique(df$Run)) == length(df)) {
    msrchart(runs = MSRData$Run,
             data = MSRData$Lst6_MSR,
             msrtype = 'MSR(last 6 Runs)',
             usrtitle = paste0('Run Chart MSR(last 6 Runs) ', usrtitle))
  }
  Last6MSRSumm <- if (length(unique(df$Run)) == length(df)) {summary(Last6MSRChart)}

  Output <- if (length(unique(df$Run)) == length(df)) {
    list(MSRData = MSRData, CummMSRChart = CummMSRChart, CummMSRSumm = CummMSRSumm, Last6M
  } else { list(MSRData = MSRData, CummMSRChart = CummMSRChart, CummMSRSumm = CummMSRSumm)
  }
```

User Input

The user will upload a .csv data file with 2 or 3 columns to UsrData. The uploaded file should be consistent with the description in the Test Data section. The user may also specify a string to be used in the report header and subtitle in the generated plots.

```
# User specifies if runs are identified by dates and a title for the generated output
```

```
UsrTitle <- 'Potency 123456'

# User uploads data - 2 or 3 columns Column 1 = Run number or date, last column = Data

UsrData <- read_csv(file = 'TestData/cc_data_10_3_50_1.csv')
## Rows: 50 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): Data
## date (1): Run
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

After user data is uploaded, it should be checked for the following:

1. Data types for columns (Run = *date* or *char*, Data = *dbl*,
2. If less than 6 runs, warn user that they must have at least 6 runs for the MSR chart.

Currently the ReplicateID field is not used in the application, but it may be used to identify specific data points in future reports, if there are outliers.

Data Analysis

Data is sorted by UsrData$Run to ensure time series order, if that column is a *date*, then the $Data column is converted to log(10) values.

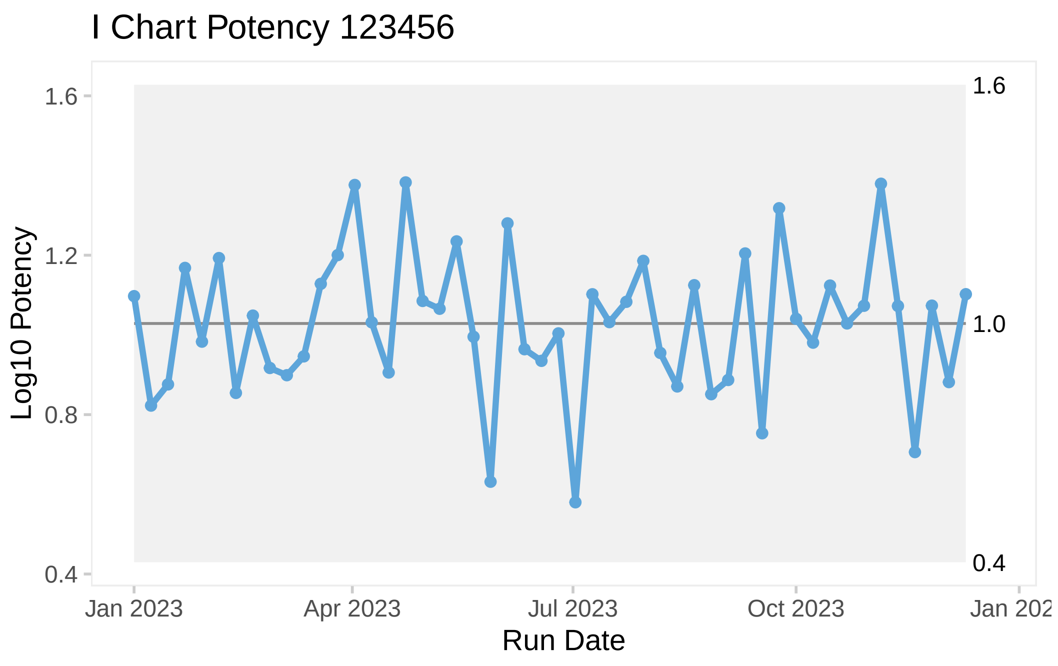
```
# Prepare data for charting

UsrData <- if (is.Date(UsrData$Run)){UsrData %>%
    arrange(Run ) }

UsrData <- mutate(UsrData, Log10Pot = log10(Data))

# Determine number of replicates to select charts to create
RepCount <- UsrData %>%
  group_by(Run) %>%
  summarise(Reps = n()) %>%
  ungroup()

CtrlCharts <- if (max(RepCount$Reps == 1)) {
  ind_charts(UsrData$Run, UsrData$Log10Pot, UsrTitle)
} else{
  rep_charts(UsrData$Run, UsrData$Log10Pot, UsrTitle)
}

MSRChart <- if (length(RepCount > 5)) {
  msr(UsrData, RepCount$Run[5], UsrTitle)
} else { 'A minimum of 6 runs are required to calculate MSR'}
```

User Report

The returned objects (CtrlCharts, and MSRChart) are lists which include data, graphs, and summary statistics that can be used to create the report for the user.

Individual data (no replicates

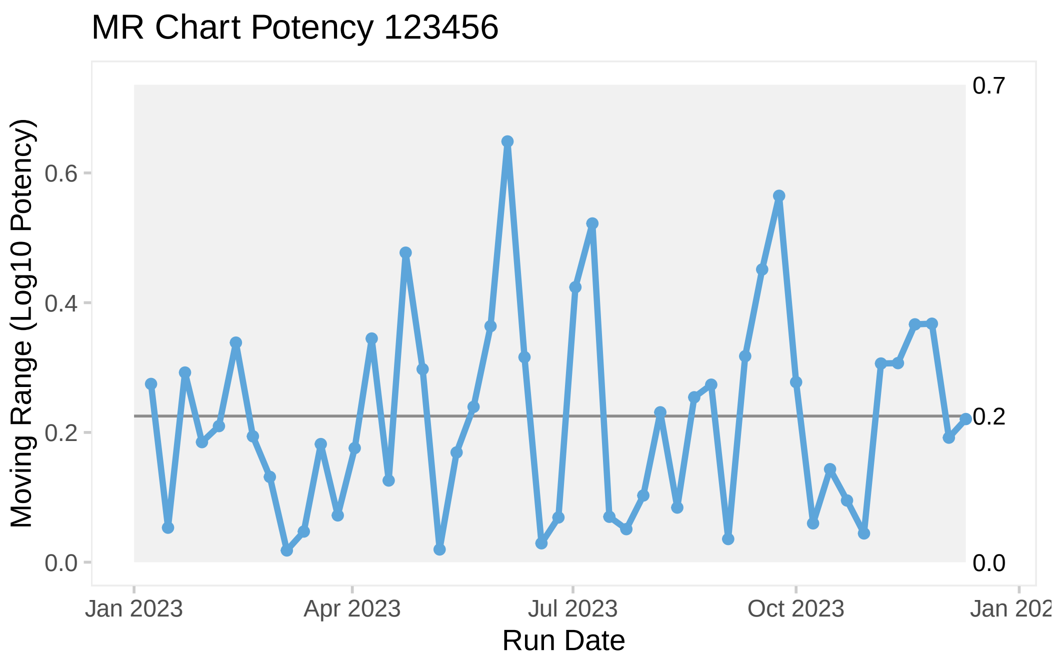
```
UsrData
## # A tibble: 50 x 3
##    Run         Data Log10Pot
##    <date>     <dbl>    <dbl>
##  1 2023-01-01 12.5     1.10
##  2 2023-01-08  6.65    0.823
##  3 2023-01-15  7.51    0.876
##  4 2023-01-22 14.7     1.17
##  5 2023-01-29  9.62    0.983
##  6 2023-02-05 15.6     1.19
##  7 2023-02-12  7.15    0.854
##  8 2023-02-19 11.2     1.05
##  9 2023-02-26  8.26    0.917
## 10 2023-03-05  7.92    0.899
## # i 40 more rows
CtrlCharts['IChart']
## $IChart
```



```
CtrlCharts['IChartSumm']
## $IChartSumm
##   facet1 facet2 part n.obs n.useful longest.run longest.run.max n.crossings
## 1      1      1    1    50       50           5               9          26
##   n.crossings.min runs.signal      aLCL    aLCL.95       CL  aUCL.95     aUCL
## 1              19           0 0.4296705 0.6293492 1.028706 1.428064 1.627742
##   sigma.signal
## 1            0
```

Using the default charts from qicharts2, so data is all on the log10 scale. The grey box represents the 99% confidence interval and the centerline is the median potency. We can transform the statistics from the summary back to the linear scale to generate graphs and reports. Also the summary stats could be parsed to help create different reports for in control and *out of control* (runs.signal or sigma.signal > 0) data.

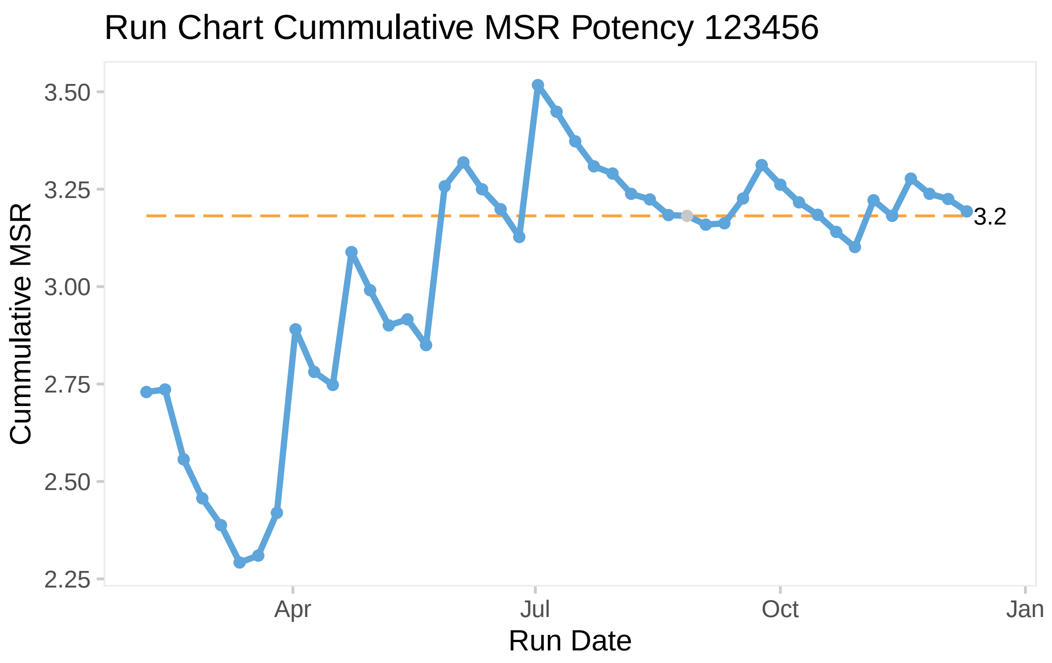
```
CtrlCharts['MRChart']
## $MRChart
```



```
CtrlCharts['MRChartSumm']
## $MRChartSumm
##   facet1 facet2 part n.obs n.useful longest.run longest.run.max n.crossings
## 1      1      1    1    50       49           7               9          21
##   n.crossings.min runs.signal aLCL aLCL.95        CL   aUCL.95     aUCL
## 1              18           0    0       0 0.2252375 0.4905673 0.735851
##   sigma.signal
## 1            0
```

Moving range chart and statistics. These can also be transformed back to the linear scale in the next iteration. Y-axis would be renamed to Moving Ratio(?)

```
MSRChart['MSRData']
## $MSRData
## # A tibble: 45 x 2
##     Run         Cumm_MSR
##     <date>         <dbl>
##   1 2023-02-05      2.73
##   2 2023-02-12      2.74
##   3 2023-02-19      2.56
##   4 2023-02-26      2.46
##   5 2023-03-05      2.39
##  6 2023-03-12     2.29
##  7 2023-03-19     2.31
##  8 2023-03-26     2.42
##  9 2023-04-02     2.89
## 10 2023-04-09     2.78
## # i 35 more rows
MSRChart['CummMSRChart']
## $CummMSRChart
```



```
MSRChart["CummMSRSumm"]
## $CummMSRSumm
##   facet1 facet2 part n.obs n.useful longest.run longest.run.max n.crossings
## 1      1      1    1    45       44          16               8           9
##   n.crossings.min runs.signal aLCL aLCL.95       CL aUCL.95 aUCL sigma.signal
## 1              16           1  NaN     NaN 3.181584     NaN  NaN            0
```

MSR Run charts are generated if there are at least 6 runs. No control limits are generated, since the variance of MSR is not normal. The Cummulative MSR Chart uses all data prior to each run.

```
MSRChart['Last6MSRChart']
## $<NA>
## NULL
MSRChart['Last6MSRSumm']
## $<NA>
## NULL
```

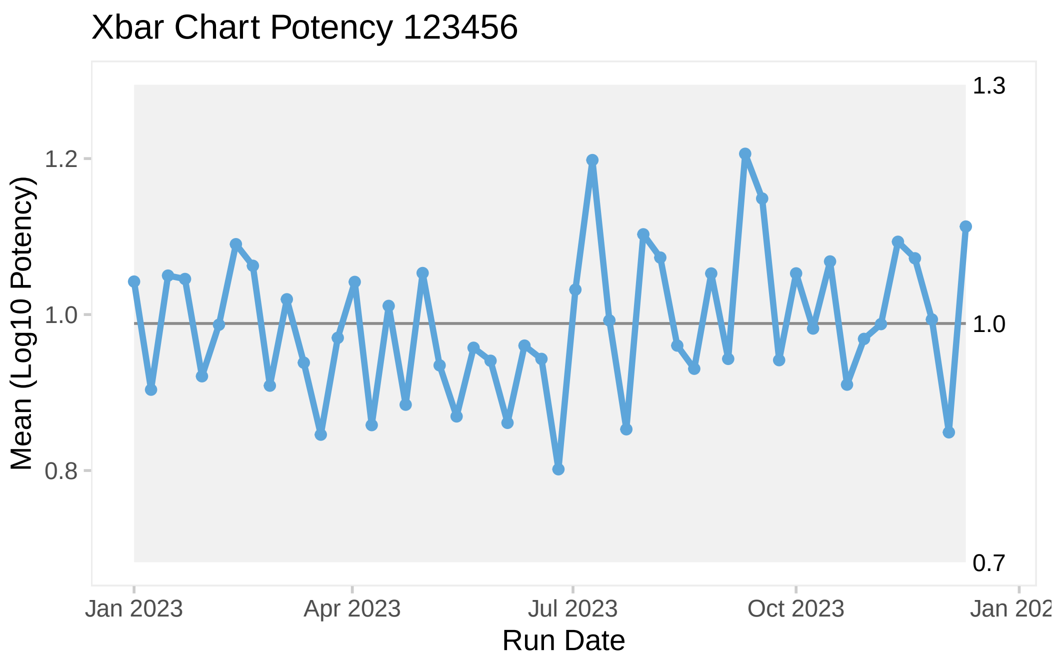
A 6-run moving window for MSR is also calculated for the individual data. Further work is required to enable this for data sets with replicates within the runs.

Replicate Data

Currently any within run replicates trigger this path. I have not thoroughly explored mixed data sets with varying replicate numbers including n=1 yet. This rule will be modified but I want some more input from the statisticians. For now I only recommend using data sets with at least 2 replicates/run.

```
UsrData <- read_csv(file = 'TestData/cc_data_10_3_50_3.csv')
## Rows: 150 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (2): Replicate_ID, Data
## date (1): Run
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

UsrData <- if (is.Date(UsrData$Run)){UsrData %>%
    arrange(Run ) }

UsrData <- mutate(UsrData, Log10Pot = log10(Data))

RepCount <- UsrData %>%
  group_by(Run) %>%
  summarise(Reps = n()) %>%
  ungroup()

CtrlCharts <- if (max(RepCount$Reps == 1)) {
  ind_charts(UsrData$Run, UsrData$Log10Pot, UsrTitle)
} else{
```

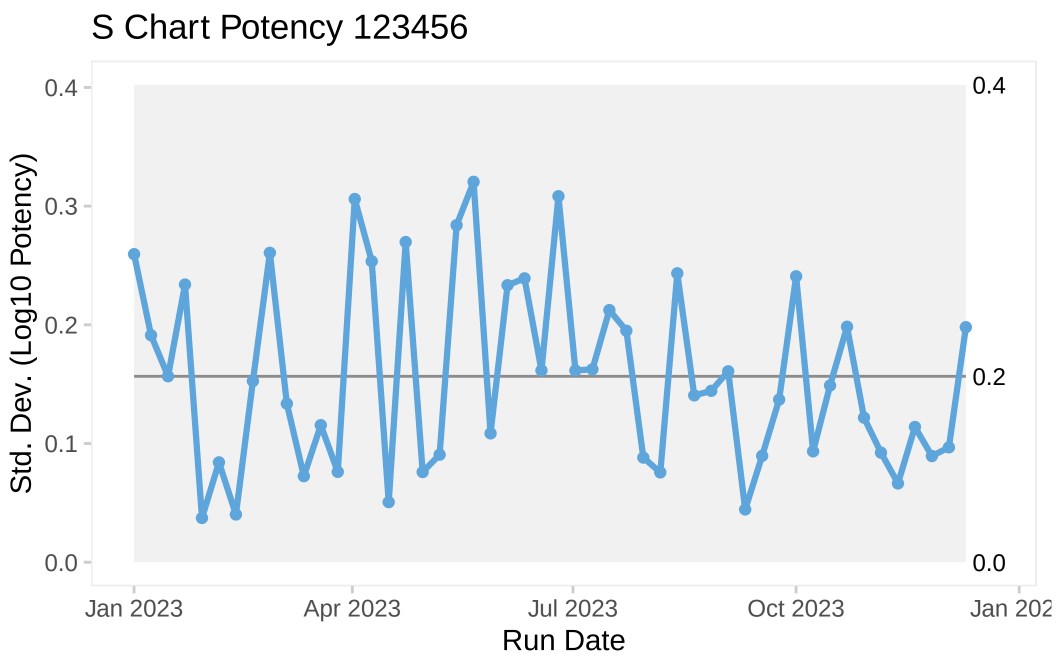
```r
  rep_charts(UsrData$Run, UsrData$Log10Pot, UsrTitle)
}

MSRChart <- if (length(RepCount > 5)) {
  msr(UsrData, RepCount$Run[5], UsrTitle)
} else { 'A minimum of 6 runs are required to calculate MSR'}

UsrData
## # A tibble: 150 x 4
##    Run        Replicate_ID  Data Log10Pot
##    <date>            <dbl> <dbl>    <dbl>
##  1 2023-01-01            1  5.84    0.767
##  2 2023-01-01            2 12.0     1.08
##  3 2023-01-01            3 19.1     1.28
##  4 2023-01-08            1  6.13    0.787
##  5 2023-01-08            2  6.30    0.799
##  6 2023-01-08            3 13.3     1.12
##  7 2023-01-15            1  9.26    0.966
##  8 2023-01-15            2 17.0     1.23
##  9 2023-01-15            3  8.97    0.953
## 10 2023-01-22            1 20.1     1.30
## # i 140 more rows

CtrlCharts["XbarChart"]
## $XbarChart
```



```
CtrlCharts["XBarSumm"]
## $XBarSumm
##   facet1 facet2 part n.obs n.useful longest.run longest.run.max n.crossings
## 1      1      1    1    50       50           8               9          28
##   n.crossings.min runs.signal      aLCL    aLCL.95        CL  aUCL.95     aUCL
## 1              19           0 0.6824358 0.7844714 0.9885426 1.192614 1.294649
##   sigma.signal
## 1            0
```

The Xbar chart plots the mean(log10(potency)) for each run with control limits as the grey box. The center line is the mean of the data.

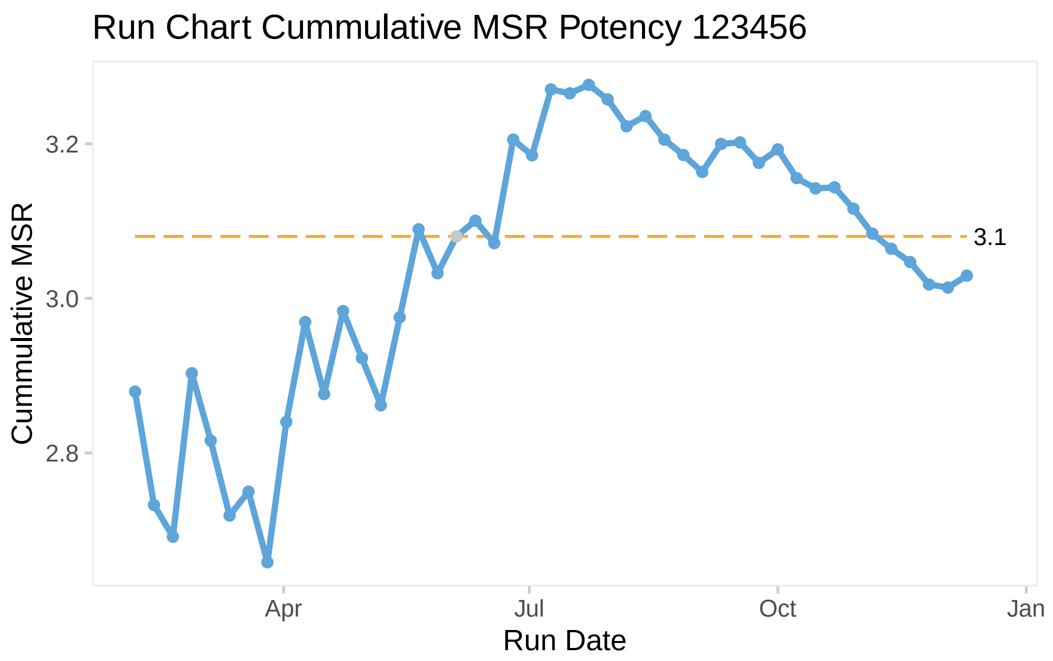
```
CtrlCharts['SChart']
## $SChart
```



```
CtrlCharts['SChartSumm']
## $SChartSumm
##   facet1 facet2 part n.obs n.useful longest.run longest.run.max n.crossings
## 1      1      1    1    50       50           8               9          20
##   n.crossings.min runs.signal aLCL aLCL.95        CL    aUCL.95      aUCL
## 1              19           0    0       0 0.1566236 0.3203652 0.402236
##   sigma.signal
## 1            0
```

Currently qicharts2 can only generate standard deviation charts for replicate data. It should be possible to generate range charts, but I will need some consultation on how to handle variable replicate numbers per run. These plots can also be transformed back to the linear scale, but I need to think a little more about the label for the transformed y-axis.

```
MSRChart['MSRData']
## $MSRData
## # A tibble: 45 x 2
##    Run        Cumm_MSR
##    <date>        <dbl>
##  1 2023-02-05     2.88
##  2 2023-02-12     2.73
##  3 2023-02-19     2.69
##  4 2023-02-26      2.90
##  5 2023-03-05      2.82
##  6 2023-03-12      2.72
##  7 2023-03-19      2.75
##  8 2023-03-26      2.66
##  9 2023-04-02      2.84
## 10 2023-04-09      2.97
## # i 35 more rows
MSRChart['CummMSRChart']
## $CummMSRChart
```



```
MSRChart["CummMSRSumm"]
## $CummMSRSumm
##   facet1 facet2 part n.obs n.useful longest.run longest.run.max n.crossings
## 1      1      1    1    45       44          20               8           6
##   n.crossings.min runs.signal aLCL aLCL.95       CL aUCL.95 aUCL sigma.signal
## 1              16           1  NaN     NaN 3.080088     NaN  NaN            0
```

Currently only cumulative MSR’s are calculated due to limitations in the slider package. I have some ideas on how to do a 6-run moving window with variable replicate number, but I’m still exploring solutions.

Next Steps

1. Create graphs for linear scale with transformed statistics.
2. Explore how qicharts2 handles mixes of individual and replicate data.
3. Create Range charts for replicate data (if needed).
4. 6-run moving window for MSR of replicate data.