AGM Potency Control Charts

Jeffrey R. Weidner

Purpose

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

This application generates Control Charts (I-MR) 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. This version of CtrlChart.R assumes that the reference compound is run only once in an experiment, but later versions can be updated to address replicates on multiple plates within an experiment. If more than one reference compound is used, multiple data sets can be uploaded and analyzed independently.

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.

Test Data

Generate a set of test potency data as a tibble with 2 columns. Run should either be an integer or a date.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
```

```
-- 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 becor
```

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

# Function to generate test data

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

  Measure <- rnorm(n = SmplSize, mean = TrueMeas, sd = StdDev)

  Output <- tibble(Run, Measure) %>%
    mutate(Measure = 10 ^ Measure)
}

TestData <- tstdata(TrueMeas = 10, TrueMSR = 3, SmplSize = 50, Dates = TRUE)

TestData
```

```
# A tibble: 50 x 2
   Run        Measure
   <date>       <dbl>
 1 2023-01-01   12.5
 2 2023-01-08    6.65
 3 2023-01-15    7.51
 4 2023-01-22   14.7
 5 2023-01-29    9.62
 6 2023-02-05   15.6
 7 2023-02-12    7.15
 8 2023-02-19   11.2
 9 2023-02-26    8.26
10 2023-03-05    7.92
# i 40 more rows
```

The tstdata function generates an ordered set of test data based on the supplied potency (TrueMeas), variability(TrueMSR), and sample size. Dates are the default Run identifier.

User Input

The user will upload a .csv data file with 2 columns to UsrData.The first column contains the run identifier as either an integer or date and the second column is a numeric value for the measurement. The the user should specify the run identifier type and data should be checked for value types. 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

Dates = TRUE
UsrTitle <- 'Potency 123456'

# User uploads data - 2 columns Column 1 = Run number or date, Column 2 = Measurement
```

Data Analysis

Data is sorted by UsrData$Run to ensure time series order then the $Measure column is converted to log(10) values. The slider package is used to iterate functions over 3 moving windows of the $Measure data:

- Cum - the current row and all previous rows.
- Lst6 - the current row and the previous 5 rows.
- Ratio - the current value and previous value.

The mean, sd, control limits and MSR are calculated for the Cum and Lst6 windows. The ratio is used to calculate the difference (log10($Measure_n$) - log10($Measure_{n-1}$) which becomes the ratio when the data is transformed back to linear.

```
library(tidyverse)
library(slider)

# numdiff subtracts the value from the previous run when called within slider
numdiff <- function(x) {
  x[2] - x[1]
}
```

```r
UsrData <- TestData %>%
  arrange(Run) %>%
  mutate(Measure = log10(Measure),
         CumMean = slide_dbl(Measure, mean, .before = Inf),
         CumSD = slide_dbl(Measure, sd, .before = Inf),
         CumUCL = CumMean + 3 * CumSD,
         CumLCL = CumMean - 3 * CumSD,
         CumMSR = 2 * sqrt(2) * CumSD,
         Lst6Mean = slide_dbl(Measure, mean, .before = 6, .complete = TRUE),
         Lst6SD = slide_dbl(Measure, sd, .before = 6, .complete = TRUE),
         Lst6UCL = Lst6Mean + 3 * Lst6SD,
         Lst6LCL = Lst6Mean - 3 * Lst6SD,
         Lst6MSR = 2 * sqrt(2) * Lst6SD,
         Ratio = slide_dbl(Measure, numdiff, .before = 1, .complete = TRUE),
         RatioUCL = 3 * CumSD,
         RatioLCL = 3 * CumSD * (-1),
         across(-Run, function(x) 10^x))


UsrData
```

```
# A tibble: 50 x 15
   Run        Measure CumMean CumSD CumUCL CumLCL CumMSR Lst6Mean Lst6SD Lst6UCL
   <date>       <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>    <dbl>  <dbl>   <dbl>
 1 2023-01-01   12.5    12.5  NA       NA     NA     NA       NA     NA       NA
 2 2023-01-08    6.65    9.12  1.56    34.9   2.38   3.54     NA     NA       NA
 3 2023-01-15    7.51    8.55  1.40    23.4   3.13   2.58     NA     NA       NA
 4 2023-01-22   14.7     9.80  1.47    31.2   3.08   2.98     NA     NA       NA
 5 2023-01-29    9.62    9.76  1.40    26.6   3.58   2.57     NA     NA       NA
 6 2023-02-05   15.6    10.6   1.43    30.6   3.64   2.73     NA     NA       NA
 7 2023-02-12    7.15    9.98  1.43    29.0   3.43   2.74      9.98   1.43    29.0
 8 2023-02-19   11.2    10.1   1.39    27.4   3.74   2.56      9.82   1.41    27.8
 9 2023-02-26    8.26    9.90  1.37    25.7   3.82   2.46     10.1    1.37    26.0
10 2023-03-05    7.92    9.68  1.36    24.4   3.85   2.39     10.2    1.36    25.6
# i 40 more rows
# i 5 more variables: Lst6LCL <dbl>, Lst6MSR <dbl>, Ratio <dbl>,
#   RatioUCL <dbl>, RatioLCL <dbl>
```

Output

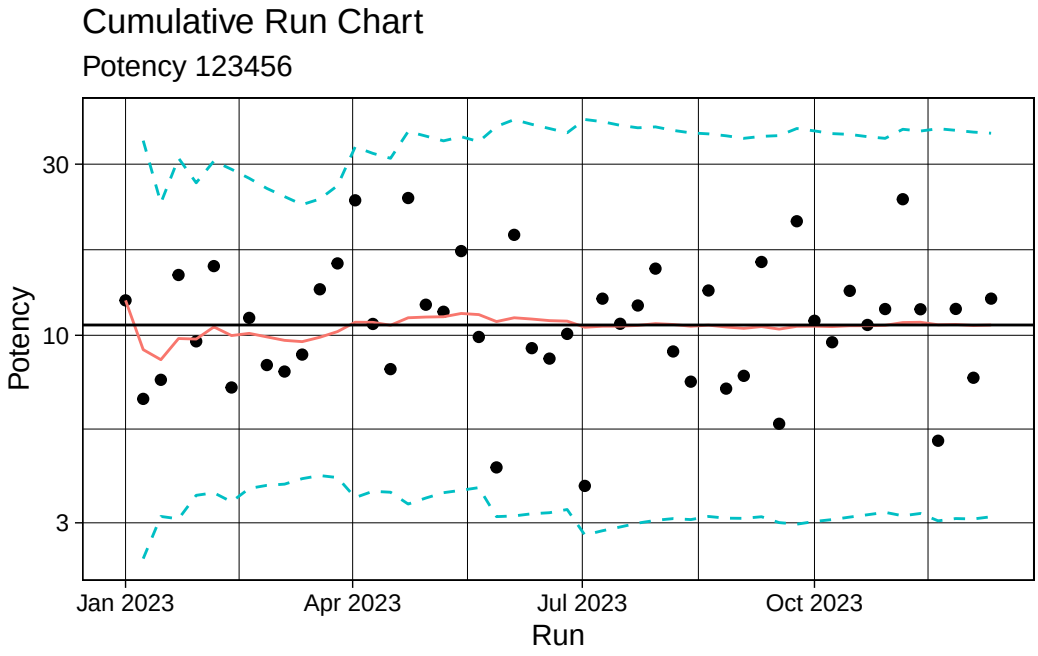
Cumulative Measure

These plots are based on an expanding window which includes all of the potency measurements up to each run and will provide the best estimates of the overall variability. However, as the size of the data set increases, the sensitivity to outliers decreases in the MSR plot.

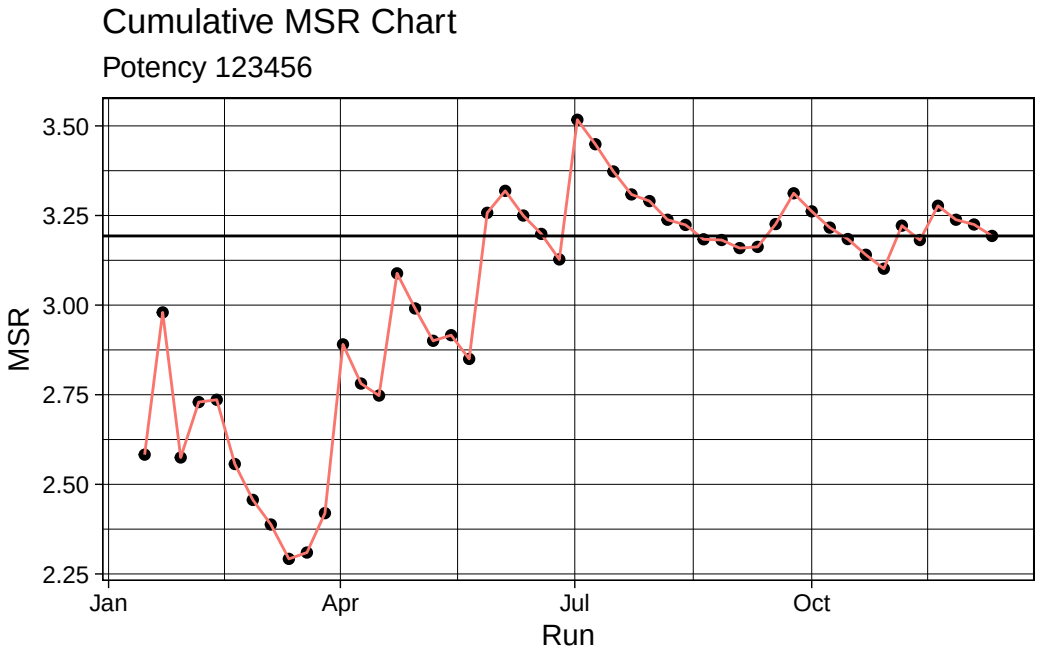
```
XLabel <- if_else(Dates, 'Date', 'Run')

CumRunPlot <- ggplot(UsrData, aes(x = Run, y = Measure)) +
  geom_point() +
  geom_line(aes(y = UsrData$CumMean, color = 'blue')) +
  geom_hline(yintercept = tail(UsrData$CumMean, 1)) +
  geom_line(aes(y = CumUCL, color = 'red'), linetype = 'dashed') +
  geom_line(aes(y = CumLCL, color = 'red'), linetype ='dashed') +
  scale_y_continuous(trans = "log10") +
  labs(title = 'Cumulative Run Chart',
       subtitle = UsrTitle,
       y = 'Potency') +
  theme_linedraw() +
  theme(legend.position = 'none')

CumMSRPlot <- ggplot(slice(UsrData, -(1:2)), aes(x = Run, y = CumMSR)) +
  geom_point() +
  geom_line(aes(color = 'blue')) +
  geom_hline(yintercept = tail(UsrData$CumMSR, 1)) +
  labs(title = 'Cumulative MSR Chart',
       subtitle = UsrTitle,
       y = 'MSR') +
  theme_linedraw() +
  theme(legend.position = 'none')

CumRunPlot
```

```
Warning: Use of `UsrData$CumMean` is discouraged.
i Use `CumMean` instead.
```

```
Warning: Removed 1 row containing missing values (`geom_line()`).
Removed 1 row containing missing values (`geom_line()`).
```


Cumulative Run Chart
Potency 123456
30
10
3
Potency
Jan 2023
Apr 2023
Jul 2023
Oct 2023
Run

```
CumMSRPlot
```


Cumulative MSR Chart
Potency 123456
3.50
3.25
3.00
2.75
2.50
2.25
MSR
Jan
Apr
Jul
Oct
Run

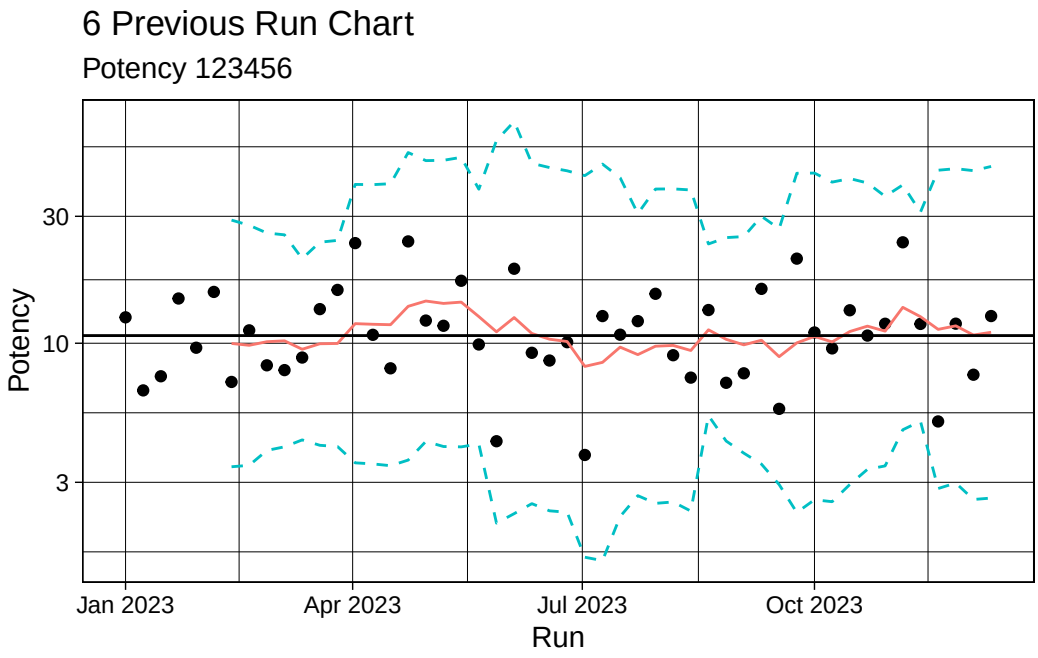
Last 6 Runs

These plots are based on a moving window that includes the the 5 previous observations. Since the sample size is smaller there will be more variability in both the mean(Potency) and MSD values. The smaller sample size makes these plots more sensitive to detecting outliers.

```
Lst6RunPlot <- ggplot(UsrData, aes(x = Run, y = Measure)) +
  geom_point() +
  geom_line(aes(y = UsrData$Lst6Mean, color = 'blue')) +
  geom_hline(yintercept = tail(UsrData$CumMean, 1)) +
  geom_line(aes(y = Lst6UCL, color = 'red'), linetype = 'dashed') +
  geom_line(aes(y = Lst6LCL, color = 'red'), linetype ='dashed') +
  scale_y_continuous(trans = "log10") +
  labs(title = '6 Previous Run Chart',
       subtitle = UsrTitle,
       y = 'Potency') +
  theme_linedraw() +
  theme(legend.position = 'none')

Lst6MSRPlot <- ggplot(UsrData, aes(x = Run, y = Lst6MSR)) +
  geom_point() +
  geom_line(aes(color = 'blue')) +
  geom_hline(yintercept = tail(UsrData$CumMSR, 1)) +
  labs(title = 'MSR (last 6 Runs) MSR Chart',
       subtitle = UsrTitle,
       y = 'MSR') +
  theme_linedraw() +
  theme(legend.position = 'none')

Lst6RunPlot
```

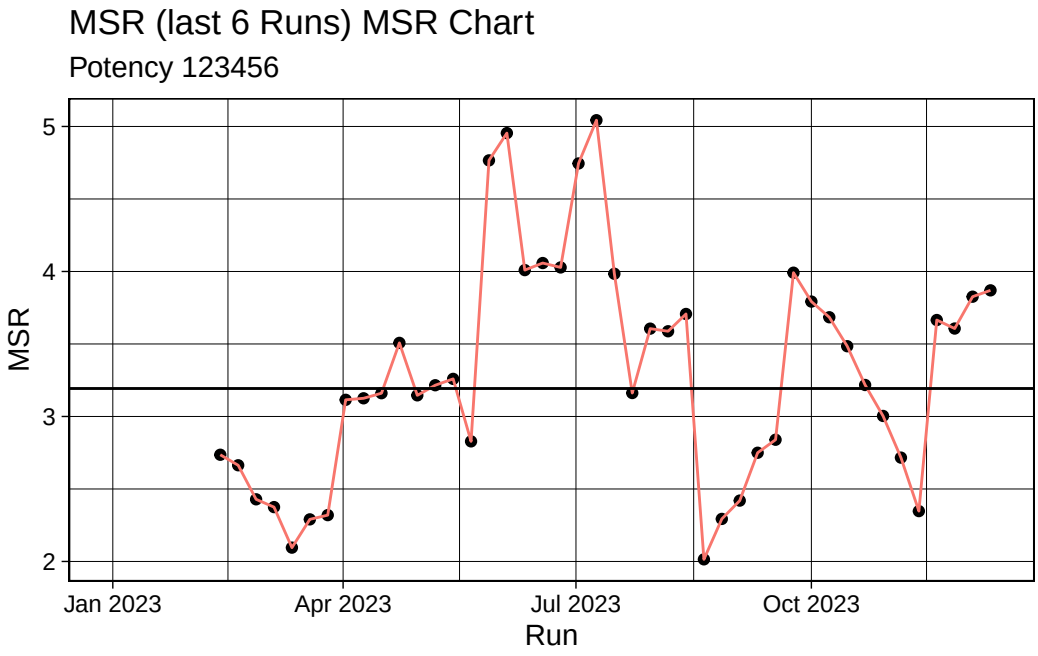
```
Warning: Use of `UsrData$Lst6Mean` is discouraged.
i Use `Lst6Mean` instead.
```

```
Warning: Removed 6 rows containing missing values (`geom_line()`).
Removed 6 rows containing missing values (`geom_line()`).
Removed 6 rows containing missing values (`geom_line()`).
```




```
Lst6MSRPlot
```

```
Warning: Removed 6 rows containing missing values (`geom_point()`).
Removed 6 rows containing missing values (`geom_line()`).
```



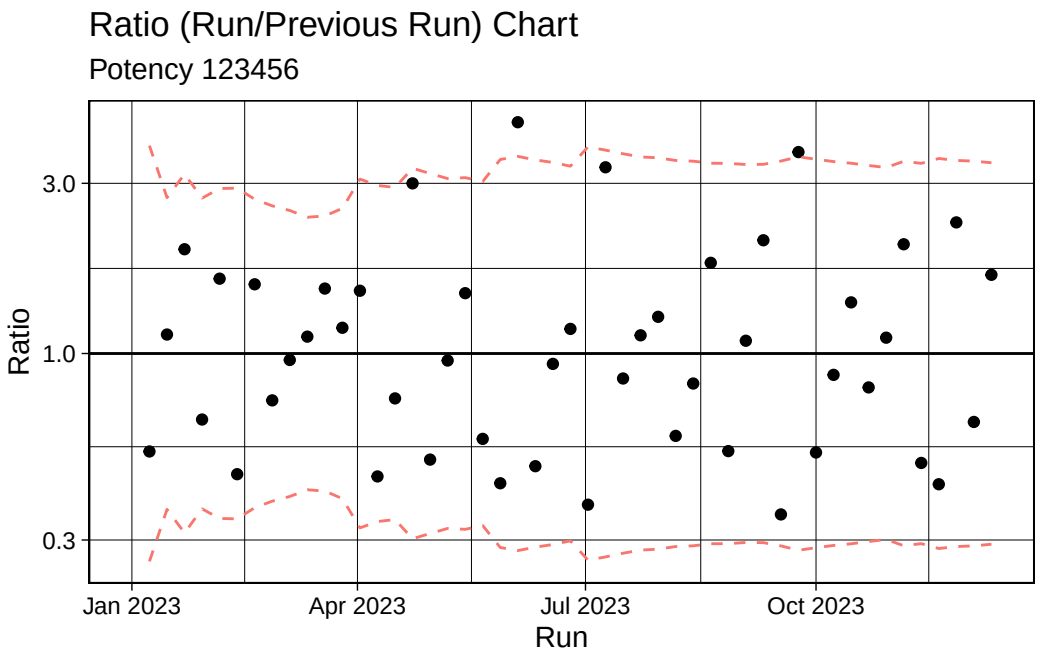
Ratio plot

This is the ratio of the potency divided by the previous potency value. Since this is the smallest sample window (n=2) it is the most sensitive to detecting outliers.

```
RatioPlot <- ggplot(UsrData, aes(x = Run, y = Ratio)) +
  geom_point() +
  geom_hline(yintercept = 1) +
  geom_line(aes(y = RatioUCL, color = 'red'), linetype = 'dashed') +
  geom_line(aes(y = RatioLCL, color = 'red'), linetype ='dashed') +
  scale_y_continuous(trans = "log10") +
  labs(title = 'Ratio (Run/Previous Run) Chart',
       subtitle = UsrTitle,
       y = 'Ratio') +
  theme_linedraw() +
  theme(legend.position = 'none')

RatioPlot
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
```

```
Warning: Removed 1 row containing missing values (`geom_line()`).
Removed 1 row containing missing values (`geom_line()`).
```



Summary Statistics

```
DataSummary <- UsrData %>%
  summarise(across(where(is.numeric), list(Min = ~min(.x, na.rm = TRUE),
                                           Max = ~max(.x, na.rm = TRUE),
                                           Last = ~last(.x)))) %>%
  pivot_longer((everything()), names_to = c('Column', 'Statistic'), names_sep = '_', value
  pivot_wider(names_from = Column, values_from = Value) %>%
  mutate_if(is.numeric, ~round(., 2)) %>%
  select(-CumSD, -Lst6SD)
DataSummary
```

```
# A tibble: 3 x 13
  Statistic Measure CumMean CumUCL CumLCL CumMSR Lst6Mean Lst6UCL Lst6LCL
  <chr>       <dbl>   <dbl>  <dbl>  <dbl>  <dbl>    <dbl>   <dbl>   <dbl>
1 Min           3.8    8.55   23.1   2.38   2.29     8.18    20.8    1.52
2 Max          24.1   12.5    40.0   4.06   3.54    14.4     68.2    5.34
3 Last         12.6   10.7    36.6   3.12   3.19    11       46.2    2.62
# i 4 more variables: Lst6MSR <dbl>, Ratio <dbl>, RatioUCL <dbl>,
#   RatioLCL <dbl>
```