**R1**

SCRIPT\_STR('

x <- .arg1

paste(x)

', 2)

**R1 with param**

SCRIPT\_STR('

x <- .arg1

paste(2 \* x)

', [Parameter for R])

**R2**

SCRIPT\_STR('

x <- .arg1

y <- length(x)

paste(x, y, sep = "~")

', SUM([Performance]))

**R2 with param**

SCRIPT\_STR('

x <- .arg1

y <- length(x)

z <- .arg2

l <- length(z)

paste(x, y, z, l, sep = "~")

', SUM([Performance]), [Parameter for R])

**R3**

SCRIPT\_STR('

x <- .arg1

y <- length(x)

z <- .arg2

m <- round(mean(x, na.rm = T), z)

n <- 1:10

paste(y, m, z, n, sep = "~")

# n

', SUM([Performance]), [Rounding])

**Rolling count**

WINDOW\_SUM(COUNT([Performance]), -11, 0)

**Rolling sum**

WINDOW\_SUM(SUM([Performance]), -11, 0)

**Rolling average**

[Rolling sum] / [Rolling count]

**Smoother**

SCRIPT\_STR('

x <- as.Date(.arg1)

y <- .arg2

ci <- .arg3

if(.arg4 == 0) {

model <- lm(y ~ x)

fit <- predict(model, interval = "confidence", level = ci)

paste(round(fit[,1], 5), round(fit[,2], 5), round(fit[,3], 5), sep = "~")

} else {

model <- loess(y ~ as.numeric(x), span = .arg5[1])

fit <- predict(model, se = T)

paste(round(fit$fit, 5),

round(fit$fit - qt(.5 + ci/2, fit$df) \* fit$se.fit, 5),

round(fit$fit + qt(.5 + ci/2, fit$df) \* fit$se.fit, 5), sep = "~")

}

', MAX([Date]), MAX([Performance]), [CI], [Smoothing method], [Span])

**fit**

FLOAT(LEFT([Smoother], FIND([Smoother], "~") - 1))

**Lower CI bound**

FLOAT(SCRIPT\_STR('

aList <- unlist(strsplit(.arg1, "~"))

aList[seq(2, length(aList), by = 3)]

', [Smoother]))

**Upper CI bound**

FLOAT(SCRIPT\_STR('

aList <- unlist(strsplit(.arg1, "~"))

aList[seq(3, length(aList), by = 3)]

', [Smoother]))

**R rolling**

SCRIPT\_REAL('

# This script requires that date pill has "Show missing values" checked!

#

library(lubridate)

df <- data.frame(date = as.Date(.arg1), perf = .arg2)

df$res <- NA

idx <- which(!is.na(df$date))

if(length(idx) != 0) {

if(idx[1] < nrow(df)) {

# Going down from first not NA

for(i in (idx[1] + 1):nrow(df)) {

if(is.na(df$date[i])) {

df$date[i] <- df$date[i - 1]

month(df$date[i]) <- month(df$date[i]) + 1

}

}

}

if(idx[1] > 1) {

# Going up from first not NA

for(i in (idx[1] - 1):1) {

if(is.na(df$date[i])) {

df$date[i] <- df$date[i + 1]

month(df$date[i]) <- month(df$date[i]) - 1

}

}

}

}

mindate <- df$date[1]

day(mindate) <- 1 # so that it preceeding or equal the maxdate

maxdate <- df$date[nrow(df)]

enddate <- seq(mindate, maxdate, by = "month")

day(enddate) <- 1

month(enddate) <- month(enddate) + 1

day(enddate) <- day(enddate) - 1

startdate <- enddate

day(startdate) <- day(startdate) + 1

month(startdate) <- month(startdate) - 12

if(.arg3 == 0) {

fun <- mean

} else {

fun <- median

}

for(i in 1:length(startdate)) {

sset <- df$date >= startdate[i] & df$date <= enddate[i]

val <- fun(df$perf[sset], na.rm = T)

df$res[year(df$date) == year(enddate[i]) & month(df$date) == month(enddate[i])] <- ifelse(is.nan(val), NA, val)

}

df$res

# paste(df$date, df$perf, df$res, sep = "~")

', MAX([Date]), MAX([Performance]), [Rolling function])

* **The data we’ll be working with**
  + Plot data aggregated. Partition by Process.
  + Deaggregate data on the plot, show multiple points for the same date
  + Change to text view. Check the number of dates, number of iterations. -> more data points than dates.
  + Deaggregate datapoints
  + Check Show Missing Values in Date pill. Observe missing months.
* **First R calculation.**
  + Calculated field R1. Take constand as input, return it. Then return after some calculation.
* **Second R calculation**
  + Duplicate R1, rename to R1 with param. Pass the param to the R calculated field
* **Third R calculation**
  + Create new calculated field R2. Pass Performance to it. Observe it need an aggregation function.
  + Put R2 field to the sheet - one point is displayed
  + Deaggregate data - 242 datapoints sent to R. Observe R calculated field is table calculation. In this case it is one data cell containing 242 values.
  + Aggregate data again. Partition data by Date, exact, discreete. Looks like Tableau is calculating by cell. Change to cell - result is the same.
  + Change calculation to Table, then to Date - same result for both, but different than before
  + Now deaggregate, calculate by Date and Table - R gets 242-items long vector, returns the same length. Tableau assigns the result to proper cells.
* **Clear the sheet. Check the aggregation is on.**
  + Add R2, add Process to color. Compute using Table. R called 6 times with 1 value. Change to Cell. Looks like partitioning by process created 6 sub-cells each with one value (or six variants of a single cell)
  + Change to Process. Now a Process vector of 6 values is passed to R.
  + Change back to Table. Deaggregate. Now again we have 6 variants of a table with single cell.
  + Change to cell - the result is as above.
  + Change to Process - now R receives the whole process vector of 242 unaggregated data points. Coloring by Process has no effect on R data.
* **One last experiment with data passed to R before you’ll get bored. This will prove usefull when we proceed to the use cases part.**
* **Clear the sheet. Check the aggregation is on.**
  + Add R2 to Text. Add Date to Rows, exact, discrete. Compute using equivalent to cell. For each existing Process entry a cell variant is created and after aggregation passed to R. Change to Cell - thesame result.
  + Change to Table. For each date an entry variant for a Process is aggregated or NA passed, if no data for that process. We get 170 entries per process. The same result is for Date. Date acts as x variable, Process partitions the data.
  + Change to Process. Process acts as x variable for the calculations, dates partition the data.
  + Now the same without aggregation:
    - by Table: for each Process a vector with at least one data point per date. NAs if necessary
    - by Cell: for each existing process data at given date there will be a vector of lenght equal to number of process data points at that date. No NAs.
    - by Date: as by Table
    - by Process: for each Date the vector consisting of unaggregated datapoints irrespective of Process
* **Duplicate R2, rename to R2 with param. pass additional parameter to R - Tableau parameter.**
  + Play a bit with different layouts, aggregations. Observe parameter gets passed as a vector. Important when addressing parameter value inside R
* **Calculation result is a scalar. Duplicate R2, rename to R3. Return mean, rounded by** **parameter**
  + Play with R3. Observe issue when there are NAs passed to ‘mean’. Modify adding na.rn = T. Play again.
  + Return ‘n’ in ‘paste’. Observe R is autorepeating shorter verctors to mach the longest
  + Return only ‘n’. Observe error in Tableau
* **Use case 1. Smoothing**
  + Plot Performance scatterplot. Add trend line.
  + Create Smoother. First with Date and Performance
  + Fit the model, return REAL vector of fit
  + Extract fit in its own calculated field
  + Plot fit on previous plot
  + Modify script to return Lower CI and Upper CI
  + Add two calculated fields to extract its values. Add to plot
  + Add CI parameter to script. Show its working
  + Now create a method selection parameter and add it to the script.
  + Switch between both methods.
  + Add CI parametrization to loess. Show.
  + Add span parameter. Show.
* **Use case 2. Rolling calculations by months.**
  + Show two approaches to RA in plain Tableau – does anybody know why they differ?
  + Create data frame, add res column
  + Check dates R is receiving (without and with split by Process)
  + Prepare date border values for input data subsetting: end dates set to last day of the month, start dates set to first day of the month. Attention to month arithmetics.
  + Assign ‘mean’ to ‘fun’
  + Loop over the end months for 12MRA, evaluate fun, assign to proper months in res.
  + Add choice of rolling function
  + Check partitioning by Process
  + Modify dates recreation chunk
  + Parition by Process
  + Filter by Process
  + Filter by Date