Investigating Global Simple Means

Mitch Toland

3/27/2020

Introduction

This report documents perhaps my own lack of understanding of how the json-files are created. The data used are those provided in prod_out/p360. The analysis is simple means. I noticed somethings I did not understand. I think I have figured out why I was thinking 0's were being skipped during averaging (see, Problem: of zeros being skipped during averaging: resolved).

Two other issues were with how percentages are being calculated and stored in the 'global' and 'local' forks of the json file (only the mantissa of the floating-point numbers is being manipulated instead of the whole number.); and not being able to identify source data (the input_data structure contains the local_subsites[[1]]_factors data values). (see, Problem: of fractions being multiplied by varying multipliers to make percentages, and Problem: input_data structure does not contain input data used to get simple means).

As to whether or not the simple means are correct. I was able to take a look at this for one overall dataset (18402) for SC. I was able to reproduce the DSR and ANOVA factorLevelId averages by averaging across the data in four SIMPLEMEANSBYSUBSITE json-files. This suggests that the by-subsite simple means are working correctly for SC in the case of single measurement per subsite data.

```
library(jsonlite)
```


The following code-block gives all of the IO and data manipulations for the results below.

```
sourceDirectory <- "/repos/RCB4Cloud/Reference/Data/prod_out/p360/"
inputFileName01 <- "SIMPLEMEANS8957-18367-STLP-outputToP360.json"
sm1        <- fromJSON(paste0(sourceDirectory, inputFileName01))
sm1Data    <- sm1$modelOutputs$simpleMeans$input$data[[1]][,c(3, 4, 8, 9)]
sm1gfactor <- sm1$modelOutputs$simpleMeans$means$global$factors[[1]]
sm1ffactor <- sm1$modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]
mergeCombinedGlobalLocalFactorData <- merge(sm1gfactor, sm1ffactor, by='id', all=T)

value.xToValue.yRatio <- mergeCombinedGlobalLocalFactorData$value.y /
                         mergeCombinedGlobalLocalFactorData$value.x
mergeCombinedGlobalLocalFactorData$Multiplier <- paste0(round(value.xToValue.yRatio))
names(mergeCombinedGlobalLocalFactorData)     <- c("factorLevelId", "Global.value", "Local.value", "Mul
# Reformat multiplier or ratio between value.x and value.y in
which10 <- which(mergeCombinedGlobalLocalFactorData$Multiplier == "1e+10")
which09 <- which(mergeCombinedGlobalLocalFactorData$Multiplier == "1e+09")
which08 <- which(mergeCombinedGlobalLocalFactorData$Multiplier == "1e+08")
which06 <- which(mergeCombinedGlobalLocalFactorData$Multiplier == "1e+06")
which05 <- which(mergeCombinedGlobalLocalFactorData$Multiplier == "1e+05")
pc10 <- paste(c(1,rep(0,10)), sep="",collapse="")
pc09 <- paste(c(1,rep(0, 9)), sep="",collapse="")
pc08 <- paste(c(1,rep(0, 8)), sep="",collapse="")
pc06 <- paste(c(1,rep(0, 6)), sep="",collapse="")
pc05 <- paste(c(1,rep(0, 5)), sep="",collapse="")
mergeCombinedGlobalLocalFactorData$Multiplier[which10] <- pc10
```

```
mergeCombinedGlobalLocalFactorData$Multiplier[which09] <- pc09
mergeCombinedGlobalLocalFactorData$Multiplier[which08] <- pc08
mergeCombinedGlobalLocalFactorData$Multiplier[which06] <- pc06
mergeCombinedGlobalLocalFactorData$Multiplier[which05] <- pc05
# Finished reformating

# Load DSR and ANOVA data to compare with simple means by subsites
inputFileName02      <- "DSR_4757-18402-SC-outputToP360.json"
inputFileName03      <- "ANOVA_GLOBAL_BLUE_696-18402-SC-outputToP360.json"
dsrJson              <- fromJSON(paste0(sourceDirectory, inputFileName02))
globalAnovaJson      <- fromJSON(paste0(sourceDirectory, inputFileName03))
dsrInputData         <- dsrJson$modelOutputs$dsr$input$data[[1]]
globalAnovaInputData <- globalAnovaJson$modelOutputs$rcbBlue$input$data[[1]]
# Check subsite-rep composition of dsrInputData and globalAnovaInputData.
# They are identical and they are multiple-location-single-rep datasets.
print("dsrInputData has multiple-loc-single-rep data (mlsr)")
```

```
## [1] "dsrInputData has multiple-loc-single-rep data (mlsr)"
```

```
 table(dsrInputData$subSiteId, dsrInputData$repId)
```

```
##
##                                      1
##   4bcb4652-860d-4398-9475-156d15f0a224 36
##   90ea907c-ce78-41b9-8600-9b86d8846edc 36
##   a35a1343-5b8d-4ee1-a1b8-b546016ab0d8 32
##   e8e9e9e6-028e-427d-8dcd-c18fa24fd9f8 36
```

```
#                                      1
# 4bcb4652-860d-4398-9475-156d15f0a224 36
# 90ea907c-ce78-41b9-8600-9b86d8846edc 36
# a35a1343-5b8d-4ee1-a1b8-b546016ab0d8 32
# e8e9e9e6-028e-427d-8dcd-c18fa24fd9f8 36

 print("globalAnovaInputData has multiple-loc-single-rep data (mlsr)")
```

```
## [1] "globalAnovaInputData has multiple-loc-single-rep data (mlsr)"
```

```
 table(globalAnovaInputData$subSiteId, globalAnovaInputData$repId)
```

```
##
##                                      1
##   4bcb4652-860d-4398-9475-156d15f0a224 36
##   90ea907c-ce78-41b9-8600-9b86d8846edc 36
##   a35a1343-5b8d-4ee1-a1b8-b546016ab0d8 32
##   e8e9e9e6-028e-427d-8dcd-c18fa24fd9f8 36
```

```
#                                      1
# 4bcb4652-860d-4398-9475-156d15f0a224 36
# 90ea907c-ce78-41b9-8600-9b86d8846edc 36
# a35a1343-5b8d-4ee1-a1b8-b546016ab0d8 32
# e8e9e9e6-028e-427d-8dcd-c18fa24fd9f8 36

# Estimate factorLevelId level means across subsites for both data sets.
factorLevelIdDsrMeans         <- aggregate(value~factorLevelId,
                                           data=dsrInputData,
```

```
                                          function(zx){mean(zx, na.rm=T)})
factorLevelIdGlobalAnovaMeans <- aggregate(value~factorLevelId,
                                          data=globalAnovaInputData,
                                          function(zx){mean(zx, na.rm=T)})
mergeDsrAnovaMeans <- merge(factorLevelIdGlobalAnovaMeans,
                            factorLevelIdDsrMeans,
                            by = "factorLevelId")
mergeDsrAnovaMeans$meanDifference <- mergeDsrAnovaMeans$value.y-mergeDsrAnovaMeans$value.x

# Read the subsite data
inputFileName04 <- "SIMPLEMEANSBYSUBSITE9-18402-SC-outputToP360.json"
inputFileName05 <- "SIMPLEMEANSBYSUBSITE8-18402-SC-outputToP360.json"
inputFileName06 <- "SIMPLEMEANSBYSUBSITE3-18402-SC-outputToP360.json"
inputFileName07 <- "SIMPLEMEANSBYSUBSITE2-18402-SC-outputToP360.json"

smbs9.402    <- fromJSON(paste0(sourceDirectory, inputFileName04))
id9.402 <- smbs9.402$modelOutputs$simpleMeans$input$data[[1]]
if9.402 <- smbs9.402$modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]
names(if9.402) <- c('id', 'value.ss9')

smbs8.402    <- fromJSON(paste0(sourceDirectory, inputFileName05))
id8.402 <- smbs8.402$modelOutputs$simpleMeans$input$data[[1]]
if8.402 <- smbs8.402$modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]
names(if8.402) <- c('id', 'value.ss8')

smbs3.402    <- fromJSON(paste0(sourceDirectory, inputFileName06))
id3.402 <- smbs3.402$modelOutputs$simpleMeans$input$data[[1]]
if3.402 <- smbs3.402$modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]
names(if3.402) <- c('id', 'value.ss3')

smbs2.402    <- fromJSON(paste0(sourceDirectory, inputFileName07))
id2.402 <- smbs2.402$modelOutputs$simpleMeans$input$data[[1]]
if2.402 <- smbs2.402$modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]
names(if2.402) <- c('id', 'value.ss2')

mergeLocalFactorData23       <- merge(if2.402, if3.402, by='id', all=TRUE)
mergeLocalFactorData89       <- merge(if8.402, if9.402, by='id', all=TRUE)
mergeLocalFactorData2389     <- merge(mergeLocalFactorData23,
                                      mergeLocalFactorData89,
                                      by  = 'id',
                                      all = TRUE)
mergeLocalFactorData2389$ave <- apply(mergeLocalFactorData2389[, 2:5],
                                      1,
                                      function(zx){mean(zx, na.rm=TRUE)})
# The average of the 4 subsites' data equals the averages in mergeDsrAnovaMeans
# which where calculated from the DSR and ANOVA input datasets.
mergeMergeMeans <- merge(mergeLocalFactorData2389,
                         mergeDsrAnovaMeans,
                         by.x = 'id',
                         by.y = 'factorLevelId',
                         all  = T)

inputFileName08 <- "SIMPLEMEANSBYSUBSITE12-18382-SC-outputToP360.json"
```

```
smbs1           <- fromJSON(paste0(sourceDirectory, inputFileName08))
smbs1Data       <- smbs1$modelOutputs$simpleMeans$input$data[[1]][,c(3, 4, 8, 9)]
smbs1gfactor    <- smbs1$modelOutputs$simpleMeans$means$global$factors[[1]]
smbs1ffactor    <- smbs1$modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]
mergeBySubsiteGlobalLocalFactorData <- merge(smbs1gfactor,smbs1ffactor, by='id', all=T)

bySsValue.xToValue.yRatio <- mergeBySubsiteGlobalLocalFactorData$value.y /
                             mergeBySubsiteGlobalLocalFactorData$value.x
mergeBySubsiteGlobalLocalFactorData$Multiplier <- paste0(round(bySsValue.xToValue.yRatio))
names(mergeBySubsiteGlobalLocalFactorData)     <- c("factorLevelId", "Global.value", "Local.value", "Mul

# Since the global and local factor data match in this case the multiplier is 1 for all rows
# and as such does not need to be reformatted as was the case in the 'combined' data.
```


The following table shows the DSR and ANOVA means for factorLevelId are the same.

```
names(mergeDsrAnovaMeans) <- c("factorLvelid", "ANOVAmeans", "DSRmeans", "Difference")
mergeDsrAnovaMeans
```

```
##    factorLvelid ANOVAmeans DSRmeans Difference
## 1        127530   48.25000 48.25000          0
## 2        127531   49.00000 49.00000          0
## 3        127532   45.75000 45.75000          0
## 4        127533   47.50000 47.50000          0
## 5        127534   52.25000 52.25000          0
## 6        127535   48.50000 48.50000          0
## 7        127536   50.75000 50.75000          0
## 8        127537   47.50000 47.50000          0
## 9        127538   48.50000 48.50000          0
## 10       127539   50.00000 50.00000          0
## 11       127540   50.00000 50.00000          0
## 12       127541   46.00000 46.00000          0
## 13       127542   47.75000 47.75000          0
## 14       127543   47.00000 47.00000          0
## 15       127544   45.25000 45.25000          0
## 16       127545   48.25000 48.25000          0
## 17       127546   49.75000 49.75000          0
## 18       127547   44.33333 44.33333          0
## 19       127548   48.25000 48.25000          0
## 20       127549   46.75000 46.75000          0
## 21       127550   48.75000 48.75000          0
## 22       127551   48.25000 48.25000          0
## 23       127552   47.25000 47.25000          0
## 24       127553   49.00000 49.00000          0
## 25       127554   49.00000 49.00000          0
## 26       127555   45.25000 45.25000          0
## 27       127556   51.00000 51.00000          0
## 28       127557   48.50000 48.50000          0
## 29       127558   48.50000 48.50000          0
## 30       127559   49.25000 49.25000          0
## 31       127560   46.00000 46.00000          0
## 32       127561   45.75000 45.75000          0
## 33       127562   49.25000 49.25000          0
```

```
## 34       127563   45.33333 45.33333          0
## 35       127564   46.75000 46.75000          0
## 36       127565   48.33333 48.33333          0
```


Problem: input_data structure does not contain input data used to get simple means

Within a json-file, data are stored in three places: input_data, global_factors and local_factors. Underscores are used instead of '$' so Rmarkdown does not interpret them. More specifically, in the modelOutputs sublist the data are found as follows.

```
cat("modelOutputs$simpleMeans$input$data[[1]]", "\n")
```

```
## modelOutputs$simpleMeans$input$data[[1]]
```

```
cat("modelOutputs$simpleMeans$means$global$factors[[1]]", "\n")
```

```
## modelOutputs$simpleMeans$means$global$factors[[1]]
```

```
cat("modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]", "\n")
```

```
## modelOutputs$simpleMeans$means$local$subsites[[1]]$factors[[1]]
```


The ...factors[[1]] data are the same for 'local' or 'by subsite' simple means, but they differ for the 'global' or 'across subsites' simple means. There is also an object in the local json-file under subsites called 'value'. This value appears to be the average of the non-missing entries in modelOutputs_simpleMeans_means_local_subsites[[1]]_factors[[1]]_value.

Again, the global_factors[[1]] and local_subsites[[1]]_factors appear to contain the same data for local analysis scope but different entries in their respective values columns for global analysis scope.

Input_data[[1]]_value contains the same data as in modelOutputs_simpleMeans_means_local_subsites[[1]]_factors[[1]]_value but in a different order. input_data also contains other data related to deactivations and subsite geolocation. # It would be nice if the 'id' levels in the local_subsites[[1]]_factors[[1]] structure were also in the input_data structure to facilitate data handling and checking.

Problem: of fractions being multiplied by varying multipliers to make percentages

In the global analysis scope results for question code, STLP, which means STand Lodging Percent (I think), the values in modelOutputs_simpleMeans_means_global_factors[[1]] are decimal fractions, but the values in modelOutputs_simpleMeans_means_local_subsites[[1]]_factors[[1]] are decimal numbers greater than one.

They look like the global_factor values turned into percentages, but using varying multipliers. It looks like the the mantissa of the floating-point number was extracted and multiplied by 10. This is illustrated in the next table.

```
mergeCombinedGlobalLocalFactorData[1:10, ]
```

```
##    factorLevelId Global.value Local.value Multiplier
## 1         127530 4.727273e-01    47.27273        100
## 2         127531 0.000000e+00          NA         NA
```

```
## 3         127532 0.000000e+00          NA         NA
## 4         127533 0.000000e+00          NA         NA
## 5         127534 4.629630e-07    46.29630  100000000
## 6         127535 4.313726e-05    43.13726    1000000
## 7         127566 0.000000e+00          NA         NA
## 8         127567 6.792453e-02    67.92453       1000
## 9         127568 4.561404e-07    45.61404  100000000
## 10        127569 0.000000e+00          NA         NA
```


Problem: of zeros being skipped during averaging: resolved.

The object, modelOutputs_simpleMeans_means_local_subsites[[1]]_value, appears to be the average of the nonmissing or nonzero values in modelOutputs_simpleMeans_means_local_subsites[[1]]_factors[[1]]_value. Missing values in modelOutputs_simpleMeans_means_local_subsites[[1]]_factors[[1]]_value are turned into 0's in modelOutputs_simpleMeans_means_global_factors[[1]]_value. This is what cause me to think zeros were being skipped during averaging.

The rollup mean, modelOutputs_simpleMeans_means_local_subsites[[1]]_value, skips the missing values in the 'local' fork of the json file: when compared to the 'global' fork of the file, it looks like the 0's are being excluded, because the NA's in the local fork are being converted to 0's in the 'global' fork.

Do the simple means match

I was only able to check this for one case of the 'local' scope, the data sets with '18402' in there files names. There are four 'SIMPLEMEANBYSUBSITE...' json-files for question code, 'SC', which is Stand Count and is recorded as an integer. There are also json-files for DSR and GLOBAL ANOVA with 18402 in there file names. The subsite data combine to give yield the same averages as calculated from teh DSr and ANOVA data. This is shown in the next two tables.

```
names(mergeMergeMeans) <- c("factorLevelId", "valueSubsite2", "valueSubsite3", "valueSubsite8", "valueSu
mergeMergeMeans[, 1:5]
```

```
##    factorLevelId valueSubsite2 valueSubsite3 valueSubsite8 valueSubsite9
## 1         127530            51            46            52            44
## 2         127531            56            43            54            43
## 3         127532            46            45            55            37
## 4         127533            53            45            56            36
## 5         127534            56            47            58            48
## 6         127535            49            44            58            43
## 7         127536            59            44            57            43
## 8         127537            50            37            57            46
## 9         127538            46            45            57            46
## 10        127539            54            46            57            43
## 11        127540            56            40            57            47
## 12        127541            49            37            58            40
## 13        127542            48            44            58            41
## 14        127543            57            33            56            42
## 15        127544            51            38            57            35
## 16        127545            56            39            54            44
## 17        127546            53            41            57            48
## 18        127547            NA            38            56            39
```

```
## 19        127548           52           44           54           43
## 20        127549           53           39           52           43
## 21        127550           54           37           57           47
## 22        127551           53           38           57           45
## 23        127552           47           45           58           39
## 24        127553           56           40           58           42
## 25        127554           NA           44           58           45
## 26        127555           54           37           54           36
## 27        127556           57           43           58           46
## 28        127557           57           37           58           42
## 29        127558           57           41           57           39
## 30        127559           52           47           58           40
## 31        127560           53           37           56           38
## 32        127561           53           32           58           40
## 33        127562           57           43           57           40
## 34        127563           NA           37           56           43
## 35        127564           50           37           57           43
## 36        127565           NA           42           56           47
```



```
mergeMergeMeans[, c(1,6:8)]
```

```
##    factorLevelId averageSubsite DSRaverage ANOVAaverage
## 1         127530       48.25000   48.25000     48.25000
## 2         127531       49.00000   49.00000     49.00000
## 3         127532       45.75000   45.75000     45.75000
## 4         127533       47.50000   47.50000     47.50000
## 5         127534       52.25000   52.25000     52.25000
## 6         127535       48.50000   48.50000     48.50000
## 7         127536       50.75000   50.75000     50.75000
## 8         127537       47.50000   47.50000     47.50000
## 9         127538       48.50000   48.50000     48.50000
## 10        127539       50.00000   50.00000     50.00000
## 11        127540       50.00000   50.00000     50.00000
## 12        127541       46.00000   46.00000     46.00000
## 13        127542       47.75000   47.75000     47.75000
## 14        127543       47.00000   47.00000     47.00000
## 15        127544       45.25000   45.25000     45.25000
## 16        127545       48.25000   48.25000     48.25000
## 17        127546       49.75000   49.75000     49.75000
## 18        127547       44.33333   44.33333     44.33333
## 19        127548       48.25000   48.25000     48.25000
## 20        127549       46.75000   46.75000     46.75000
## 21        127550       48.75000   48.75000     48.75000
## 22        127551       48.25000   48.25000     48.25000
## 23        127552       47.25000   47.25000     47.25000
## 24        127553       49.00000   49.00000     49.00000
## 25        127554       49.00000   49.00000     49.00000
## 26        127555       45.25000   45.25000     45.25000
## 27        127556       51.00000   51.00000     51.00000
## 28        127557       48.50000   48.50000     48.50000
## 29        127558       48.50000   48.50000     48.50000
## 30        127559       49.25000   49.25000     49.25000
## 31        127560       46.00000   46.00000     46.00000
```

```
## 32        127561       45.75000   45.75000     45.75000
## 33        127562       49.25000   49.25000     49.25000
## 34        127563       45.33333   45.33333     45.33333
## 35        127564       46.75000   46.75000     46.75000
## 36        127565       48.33333   48.33333     48.33333
```


This suggests that the values in the local or by subsite datasets are the values for each individual subsite composing the larger dataset used in DSR and ANOVA. These individual values will be the average value for their respective subsites. Thus it seems that the averaging on the local scope is working as intended.

I wasn't able to identify the source data for the global simple means and therefore cannot check them.