Case Study for "SamplingLithics"

R. Peters

28 April 2020

Technical Note & Caveat

This manual presents an example application of **SamplingLithics** workflow. It is based upon the `.R` files which are in the `code\` directory of the GitHub repository SamplingLithics. If you want to apply the workflow on your own data, I recommend you to use these files, as it is possible to individually adjust certain variables of the `SamplingLithics.R` file. However, the code chunks shown in this manual originate from the mentioned `.R` files. It is possible that the files of the GitHub repository have changed, due to improvements. To run the code of the script the package `plyr` is needed, if you haven't please install it using `install.packages("plyr")`. Furthermore, the script was developed under R version 3.6.3.

```
#-- 0. Load Packages ----
require(plyr)
```

```
## Loading required package: plyr
```

Introduction

Archaeologist often face the problem of recording hugh assemblages, for instance of stone artefacts. **SamplingLithics** aims at optimal recording lithic data sets with minimal time investment. It is an implementation of the EasyGilf work routine in R, and credit for inventing the method - originally a MS Access database - belong to the EasyGilf team (Jörg Linstädter, Jürgen Richter, Anja Linstäder, Insitute for Prehistoric Archaelogy, University of Cologne). Please refer to their publication for further information (*Lindstädter/Richter/Lindstädter 2002*).

The goal of **SamplingLithics** is to record quanitativ (mean, median of measurements) and qualitative (percentages) of lithic attributes in a time saving way without reducing validity. The workflow aims at finding a representative sample by calculating running/rolling summary statistics of certain attributes. Already during the phase of data recording one should start with analysing small randomly choosen batches of artefacts. Every new sample is added to the preciding group of batches until the estimates converge. As soons as the estimates of the current batch and the batches before are identical, one can stop recording and the optimal sample size is reached. Put in another way, **SamplingLithics** or EasyGilf are not about a certain sampling strategy but solely concerned with the sampling intensity or the question "How many artefacts do i have to look at to get solid summary statistics of my attributes?".

Working steps:
1. Import Data
2. Prepare Data
3. Define Sampling Variables and Draw Samples
4. Running means of measurements (numerical variables)
5. Running proportions of qualitativ variables
6. Plotting
7. Compare Estimates (samples) and overall values (original data set)

1. Import Data

For this exemplary application, we use a data set of neolithic stone tools - mostly flakes, blades and tools - from a neolithic settlement site. The `data.csv` consists of 510 records and following fields/attributes:

```
#-- 1. Import Data ----

data <- read.csv2("data/data.csv")

head(data)
```

```
##   feature length width thickness weight  type    cortex   burned   mod
## 1      15     17    16         5    1.6 blade no cortex unburned   mod
## 2      30     33    12         3    1.2 blade no cortex unburned unmod
## 3      85     23    20         5    2.0 blade no cortex unburned unmod
## 4      97     44    29         9   15.7 flake    cortex unburned unmod
## 5      97     48    35        14   21.9 flake    cortex unburned unmod
## 6     112     34    31         6    5.1 flake no cortex unburned unmod
```

```
nrow(data)
```

```
## [1] 520
```

`feature` = number of feature

`length` = length of artifact in mm

`width` = width of artifact in mm

`thickness` = thickness of artifact in mm

`weight` = weight of artifact in g

`type` = type of blank (flake, blade or other type of blank)

`cortex` = presence/absence of cortex

`burned` = presence/absence of burning or traces of fire

`mod` = presence/absence of modification (mod = artefact is a tool)

2. Prepare Data

To prepare the data we recode missing values ("999") to `NA` and add an ID column (a concescutive number). Then one can have a look at the summary statistics for the numeric attributes and the counts for the qualitativ attributes.

```
#-- 2. Prepare Data -----

# Recoding Missing Values ("999" to NA)
data$length[data$length==999] <- NA
data$width[data$width==999] <- NA
data$thickness[data$thickness==999] <- NA
data$weight[data$weight==999] <- NA

# add ID column:
data <- cbind(data, SampLithicsID = seq(1, nrow(data)))
head(data)
```

```
##   feature length width thickness weight  type    cortex   burned   mod
## 1      15     17    16         5    1.6 blade no cortex unburned   mod
## 2      30     33    12         3    1.2 blade no cortex unburned unmod
## 3      85     23    20         5    2.0 blade no cortex unburned unmod
## 4      97     44    29         9   15.7 flake    cortex unburned unmod
## 5      97     48    35        14   21.9 flake    cortex unburned unmod
## 6     112     34    31         6    5.1 flake no cortex unburned unmod
##   SampLithicsID
## 1             1
## 2             2
## 3             3
## 4             4
## 5             5
## 6             6
```

```
# Summary statistics per field (numeric variables), for example length
summary(data$length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     9.0    20.0    28.0    29.5    36.0    82.0       3
```

```
# Counts per attribute (qualitativ variables/factors), for example blank type
table(data$type)
```

```
##
## blade flake other
##   187   274    59
```

3. Define Sampling Variables and Draw Samples

Now we have to define the number of batches/training sets `n_batch` and the number of artefacty per batch `A`. Both values depend on your data set, especially the variability of your attirutes and it is advisable to explore these values by running the process several times. Oviously `n_batch` * ´A´ has to be smaller than ´nrow(data)´.

In a real world example one wouldn't draw a sample from a data set but decide on the numer of batches and the number of artefacts beforehand or add batches during find recording.

As a starting point we will beginn with **20** batches containing **5** artefacts totalling to **100** objects, circa **1/5** of the overall data set.

```
# Define nummber of batches/training sets (n_batch) and number of artefacts per batch (A)

n_batch = 20
A = 5

n_batch*A <= nrow(data)
```

```
## [1] TRUE
```

There is a function to split the originall data set into batches/training sets. An indicator column is added to a new data.frame `train` which only contains the artefacts selected randomly.

```r
# Function to produce training sets
copy.data <- data
train <- data.frame()

for (i in 1:n_batch){
  dt = sort(sample(nrow(copy.data), A))
  train  <- rbind(train , cbind(copy.data[dt,], "Train_Set" = rep(i,A)))
  copy.data<-copy.data[-dt,]
}

# Have a look at train set 1:
print(train[train$Train_Set==1,], row.names = FALSE)
```

```
##  feature length width thickness weight  type    cortex   burned   mod
##     2559     34    20         6    3.4 flake no cortex unburned unmod
##     3359     21    16         3    0.8 flake no cortex unburned unmod
##     3567     21    20         7    3.0 blade no cortex unburned unmod
##     4773     16    11         5    1.0 blade no cortex unburned unmod
##     5825     40    20         3    1.8 flake no cortex unburned   mod
##  SampLithicsID Train_Set
##            229         1
##            282         1
##            346         1
##            434         1
##            492         1
```

```r
nrow(train)
```

```
## [1] 100
```

```r
# IDs should be unique / no dublicates:
nrow(train)==length(unique(train$SampLithicsID))
```

```
## [1] TRUE
```

```r
# IDs per batch
table(train$Train_Set)
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
```

4. Running means of measurements (numeric variables)

In the next step we will calculated the summary statistics (mean oder median) for the numeric variables (measurements), for example artifact length. The routine is as follows: the mean value is calculated for the 1st batch, than the 2nd batch is added and the mean of the records in the 1st and 2nd batch is calculated, then the mean of batch 1-3, 1-4, and so on. You could also do it by hand:

```r
# Manual computing:

#Batch 1
mean(train$length[train$Train_Set==1],na.rm = T)
```

```
## [1] 26.4
```

```r
# Batch 1-2
mean(train$length[train$Train_Set%in%c(1,2)],na.rm = T)
```

```
## [1] 33.4
```

```r
# Last Batch containing all samples
mean(train$length[train$Train_Set%in%seq(1,n_batch)],na.rm = T)
```

```
## [1] 29.73737
```

Luckily, we can build a function for that:

```r
# Function get running means of measurments

run_mean <- numeric()
c <- NULL

get_run_var <- function(my_variable){
  for (i in 1:n_batch){
  select <- seq(1,i)
  c = mean(my_variable[train$Train_Set%in%select],na.rm = T)
  #print(c)
  run_mean <- c(run_mean, c)
  }
  return(run_mean)
}
```

Now that we have the summary statistics we can calculate the difference in percentage between the batches:

```r
get_run_diff <- function(my_variable){
  diff_prc <- (diff(my_variable)/(my_variable[-1]/100))
  return(diff_prc)
}
```

Compute means and percental difference for all numeric variables:

```r
weight_mean <- get_run_var(train$weight)
weight_diff_prc <- get_run_diff(weight_mean)

length_mean <- get_run_var(train$length)
length_diff_prc <- get_run_diff(length_mean)

width_mean <- get_run_var(train$width)
```

```
width_diff_prc <- get_run_diff(width_mean)

thickness_mean <- get_run_var(train$thickness)
thickness_diff_prc <- get_run_diff(thickness_mean)
```

We can double-check the results with our manually calculated values:

```
# Compare to manual computing:
mean(train$length[train$Train_Set%in%seq(1,n_batch)],na.rm = T)
```

```
## [1] 29.73737
```

```
length_mean[n_batch]
```

```
## [1] 29.73737
```

5. Running proportions of qualitative variables

Now we will do the same for the qualitative values. We will use `ddply` from the `plyr` package to count the number of blades, burned, cortical and modified pieces.

```
# Proportions/share of qualitative variables

t <- plyr::ddply(train, ~Train_Set, summarize,
           blade_n = sum(table(type, exclude=c("flake", "other"))),
           burned_n = sum(table(burned, exclude=c("unburned"))),
           cortex_n = sum(table(cortex, exclude=c("no cortex"))),
           mod_n = sum(table(mod, exclude=c("unmod"))),
           all_n = sum(table(feature))
)

print(head(t), row.names = F)
```

```
##  Train_Set blade_n burned_n cortex_n mod_n all_n
##          1       2        0        0     1     5
##          2       1        0        3     4     5
##          3       1        0        1     1     5
##          4       3        1        1     1     5
##          5       3        0        1     2     5
##          6       1        0        2     2     5
```

Then we will compute the cummulative percentages, for example % of blades in batch 1, in batch 1 and 2, 1-3, and so on. One can use `diff` to get the interated differences of the percentages, e.g. (% batch 1 - % batch 2), (% bacht 1-2 - % batch 1-3).

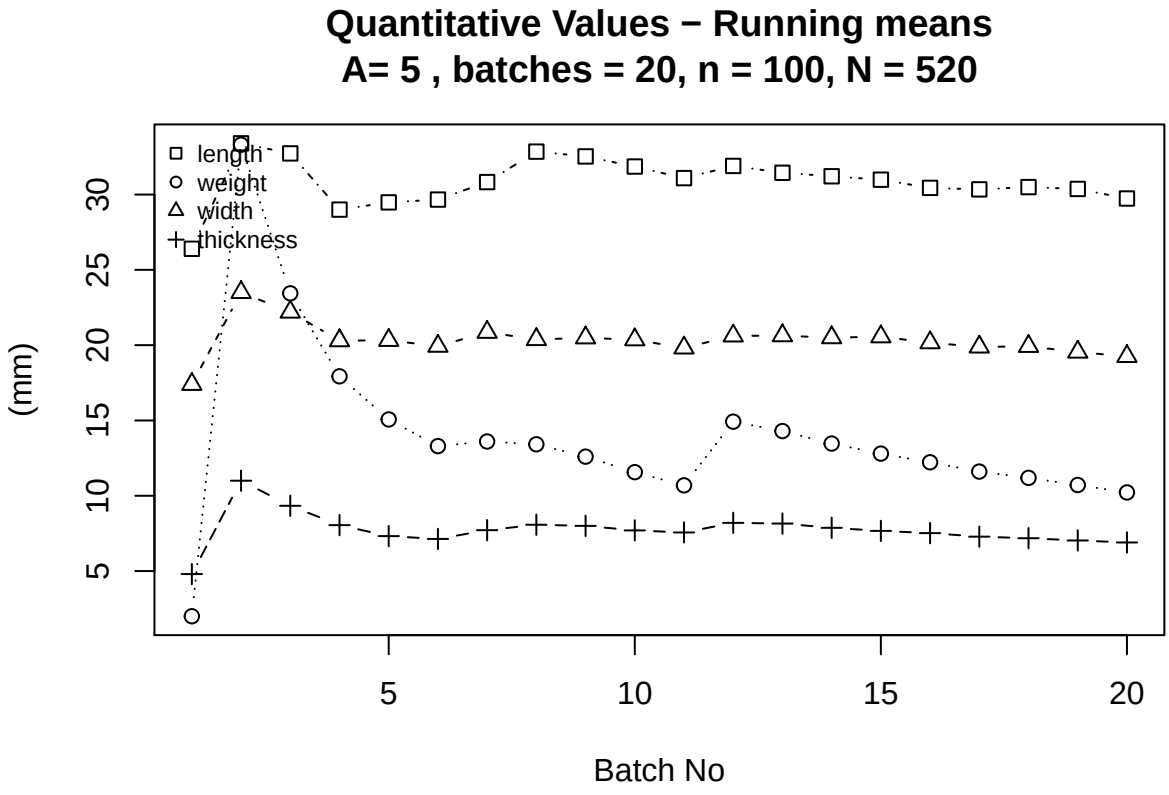
```
# Blades
blade_prc <- cumsum(t$blade_n)/(cumsum(t$all_n)/100)
blade_prc_diff <- diff(blade_prc)/(blade_prc[-1]/100)

# Burned
```

```r
burned_prc <- cumsum(t$burned_n)/(cumsum(t$all_n)/100)
burned_prc_diff <- diff(burned_prc)/(burned_prc[-1]/100)

# Cortex
cortex_prc <- cumsum(t$cortex_n)/(cumsum(t$all_n)/100)
cortex_prc_diff <- diff(cortex_prc)/(cortex_prc[-1]/100)

# Modification
mod_prc <- cumsum(t$mod_n)/(cumsum(t$all_n)/100)
mod_prc_diff <- diff(mod_prc)/(mod_prc[-1]/100)
```

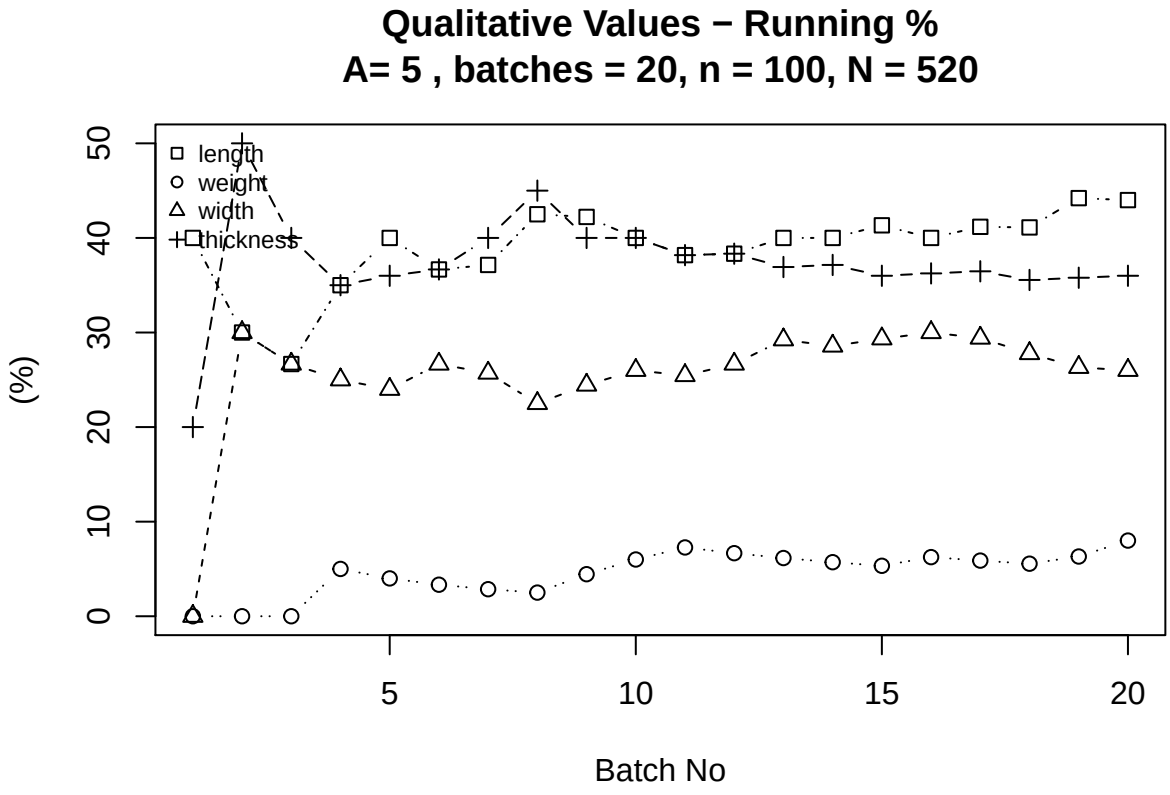
6. Plotting

Finally, we are ready to plot some of our results:

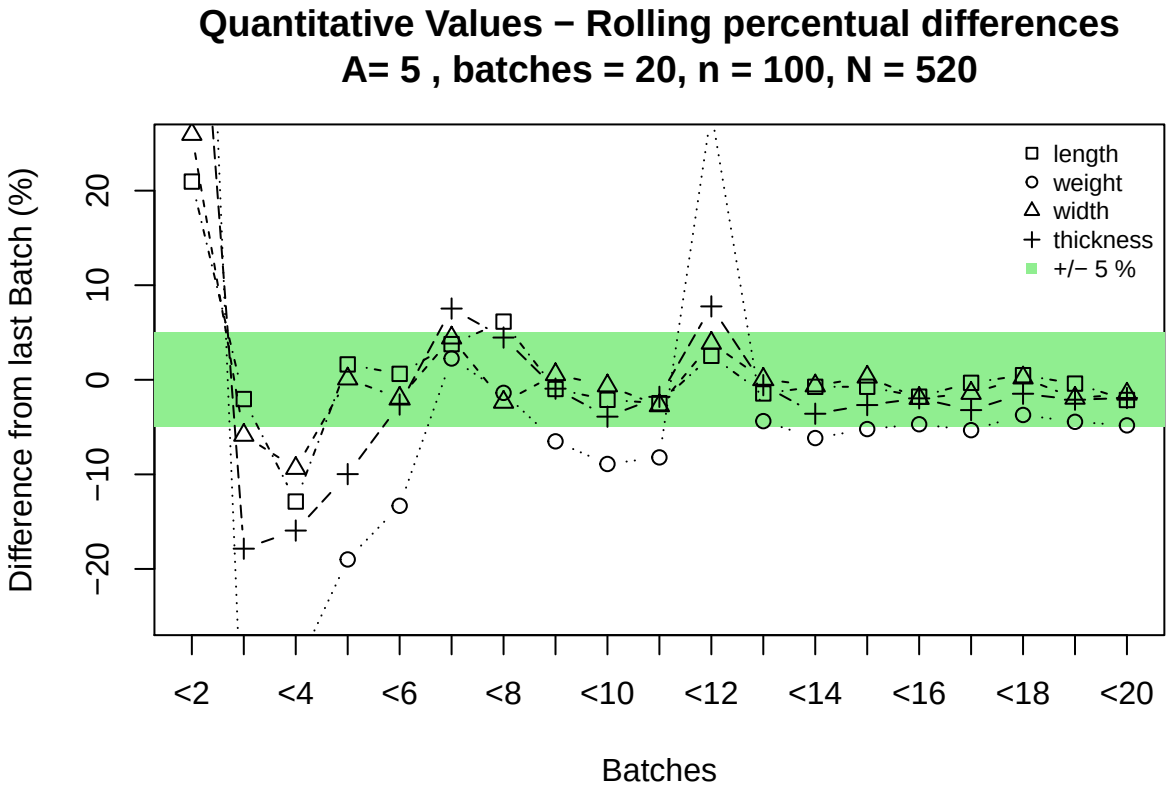
```r
# Quantitative Values
xmin <- min(length_mean, weight_mean, width_mean, thickness_mean)
ymin <-max(length_mean, weight_mean, width_mean, thickness_mean)

plot(1, type="n", xlim=c(1, n_batch), ylim=c(xmin, ymin), ylab="(mm)", xlab = "Batch No")
lines(length_mean, pch = 0, type="b", lty=4)
lines(weight_mean, pch = 1, type="b", lty=3)
lines(width_mean, pch = 2, type="b", lty=2)
lines(thickness_mean, pch = 3, type="b", lty=5)
main_title <- paste0("A= ",A," , batches = ", n_batch, ", n = ", A*n_batch,", N = ",nrow(data))
title(main=paste0("Quantitative Values - Running means\n", main_title))
legend("topleft", c("length", "weight", "width", "thickness"),
       pch = c(0,1,2,3),box.lty = 0, cex = 0.75)
```



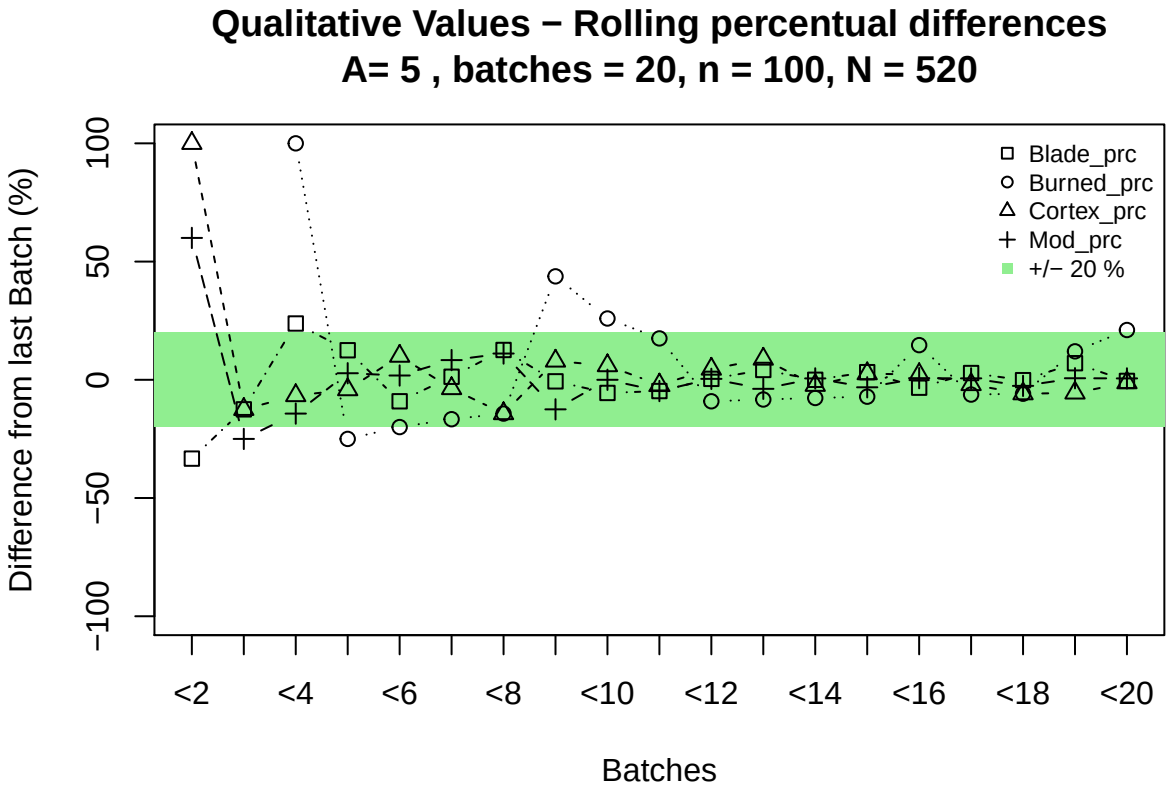

```
# Quantitative Values
xmin <- min(blade_prc, burned_prc, cortex_prc, mod_prc)
ymin <-max(blade_prc, burned_prc, cortex_prc, mod_prc)

plot(1, type="n", xlim=c(1, n_batch), ylim=c(xmin, ymin), ylab="(%)", xlab = "Batch No")
lines(blade_prc, pch = 0, type="b", lty=4)
lines(burned_prc, pch = 1, type="b", lty=3)
lines(cortex_prc, pch = 2, type="b", lty=2)
lines(mod_prc, pch = 3, type="b", lty=5)
main_title <- paste0("A= ",A," , batches = ", n_batch, ", n = ", A*n_batch,", N = ",nrow(data))
title(main=paste0("Qualitative Values - Running %\n", main_title))
legend("topleft", c("length", "weight", "width", "thickness"),
       pch = c(0,1,2,3),box.lty = 0, cex = 0.75)
```




```
# Quantitative Values
plot(1, type="n", xlim=c(1, n_batch-1), ylim=c(-25, 25), ylab="Difference from last Batch (%)", xlab = "
axis(1, at = seq(1, n_batch-1), label=paste0("<",seq(2, n_batch)))
polygon(c(0,n_batch,n_batch,0),c(5,5,-5,-5), col="lightgreen",border = NA)
lines(length_diff_prc, pch = 0, type="b", lty=4)
lines(weight_diff_prc, pch = 1, type="b", lty=3)
lines(width_diff_prc, pch = 2, type="b", lty=2)
lines(thickness_diff_prc, pch = 3, type="b", lty=5)
main_title <- paste0("A= ",A," , batches = ", n_batch, ", n = ", A*n_batch,", N = ",nrow(data))
title(main=paste0("Quantitative Values - Rolling percentual differences\n", main_title))
legend("topright", c("length", "weight", "width", "thickness","+/- 5 %"),
       pch = c(0,1,2,3,15), col=c(rep("black",4),"lightgreen"),box.lty = 0, cex = 0.75)
```




```
# Qualitative Values
plot(1, type="n", xlim=c(1, n_batch-1), ylim=c(-100, 100), ylab="Difference from last Batch (%)", xlab =
axis(1, at = seq(1, n_batch-1), label=paste0("<",seq(2, n_batch)))
polygon(c(0,n_batch,n_batch,0),c(20,20,-20,-20), col="lightgreen",border = NA)
lines(blade_prc_diff, pch = 0, type="b", lty=4)
lines(burned_prc_diff, pch = 1, type="b", lty=3)
lines(cortex_prc_diff, pch = 2, type="b", lty=2)
lines(mod_prc_diff, pch = 3, type="b", lty=5)
title(main=paste0("Qualitative Values - Rolling percentual differences\n", main_title))
legend("topright", c("Blade_prc", "Burned_prc", "Cortex_prc", "Mod_prc","+/- 20 %"),
       pch = c(0,1,2,3,15), col=c(rep("black",4),"lightgreen"),box.lty = 0, cex = 0.75)
```



As you can see the running estimates start of very variable and converge as more batches are beeing added. Linstädter et al. define a corridor of +/- 5 percent for numerical and +/-20 percent for percentages. Please note that the y-axis are scaled differently. If two times two consecutive values fall within this corridor the optimal sampling size has been reached according to Linstädter et al. This limit has to be defined for every variable on its own.

7. Compare Estimates (samples) and overall values (original data set)

Last but not least, in this case study we can compare the estimates computed using the routine by **Linstädter/Richter/Linstädter 2002** with the summary statistics of the original data set. Of course in a real world example there wouldn't be a original data set, because we are trying to avoid recording all artificats of an assemblage.

```
# Put all results in one data.frame:
results <- data.frame(A = rep(A, 4), n_batch = rep(n_batch,4),

                      qual = c("blade_prc","burned_prc","cortx_prc","mod_prc"),

                      qual_estimates =
                        c(round(blade_prc[n_batch],1),
                          round(burned_prc[n_batch],1),
                          round(cortex_prc[n_batch],1),
                          round(mod_prc[n_batch],1)),

                      qual_real = c(
```

```
                        round((prop.table(table(data$type))*100)[1],1),
                        round((prop.table(table(data$burned))*100)[1],1),
                        round((prop.table(table(data$cortex))*100)[1],1),
                        round((prop.table(table(data$mod))*100)[1],1)
                        ),

                      quant = c("length", "width", "thickness", "weight"),

                      quant_estimates =
                        c(round(length_mean[n_batch],1),
                          round(width_mean[n_batch],1),
                          round(thickness_mean[n_batch],1),
                          round(weight_mean[n_batch],1)),

                      quant_real =
                        c(round(mean(data$length,na.rm = T),1),
                          round(mean(data$width,na.rm = T),1),
                          round(mean(data$thickness,na.rm = T),1),
                          round(mean(data$weight,na.rm = T),1))

                      )



results <- cbind(results,
                 qual_diff_prc =
                   round(results$qual_real-results$qual_estimates,1))

results <- cbind(results,
                 quant_diff_prc =
                   round((results$quant_estimates-results$quant_real)/(results$quant_real/100),1))

rownames(results) <- NULL

print(results, row.names = F)
```

```
##  A n_batch       qual qual_estimates qual_real     quant quant_estimates
##  5      20  blade_prc             44      36.0    length            29.7
##  5      20 burned_prc              8      12.5     width            19.3
##  5      20  cortx_prc             26      28.0 thickness             6.9
##  5      20    mod_prc             36      29.2    weight            10.2
##  quant_real qual_diff_prc quant_diff_prc
##        29.5          -8.0            0.7
##        19.6           4.5           -1.5
##         6.9           2.0            0.0
##         7.2          -6.8           41.7
```

Bibliography

J. Linstädter/J. Richter/A. Linstädter, **2002** Easygilf, Optimale Datenerhebung mit minimalen Aufwand. *Archäologische Informationen 25, 1&2, 99-106.*