Quality Control–based Robust LOESS Signal Correction in R

Wanchang Lin

2025-04-02

Contents

0.1 Load libraries

The first step is to load necessary libraries. `mt` is used to plot PCA, PLS and LDA plots to access performance of signal correction. `tictoc` records the running time, especially for the optimisation of LOESS.

```
pkgs <- c("qcrlscR", "mt", "tidyverse", "tictoc")
## install.packages(pkgs)
invisible(lapply(pkgs, library, character.only = TRUE))
```

0.2 Read data

The data used is `man_qc` in package `qcrlscR`. This data set is a list of two data frames, `data` and `meta`.

```
names(man_qc)
#> [1] "data" "meta"
t(sapply(man_qc, dim))
#>      [,1] [,2]
#> data  462  656
#> meta  462    2
```

Get meta and data matrix

```
meta <- man_qc$meta
data <- man_qc$data %>%
  mutate_if(is.character, as.numeric)
```

Extract group information of batch and sample types

```
names(meta)
#> [1] "batch"       "sample_type"
cls.qc <- factor(meta$sample_type)
table(cls.qc)
#> cls.qc
#>     QC Sample
#>    110    352

cls.bl <- factor(meta$batch)
table(cls.bl)
#> cls.bl
#>   1   2   3   4
#> 119 114 119 110
```

0.3 Missing value filter and fill

Check missing value rates

```r
tail(sort(mv.perc(data)), 20)
#> V1986  V562 V2098 V1602  V348 V1902  V975 V2017 V2020  V163 V1021 V1676 V1540
#> 0.156 0.158 0.160 0.162 0.165 0.167 0.169 0.169 0.169 0.171 0.171 0.171 0.173
#> V1321 V1935 V1079  V610 V1691 V2077  V926
#> 0.182 0.182 0.190 0.197 0.197 0.197 0.199
```

Filter based on missing values

```r
filter_qc <- FALSE      # filter on qc missing values or all missing values
thres <- 0.15           # threshold for filtering

if (filter_qc) {      # filter using all missing values
  ret <- mv.filter(data, thres = thres)
} else {              # filter using qc missing values
  ret <- mv.filter.qc(data, cls.qc, thres = thres)
}
```

Update data matrix

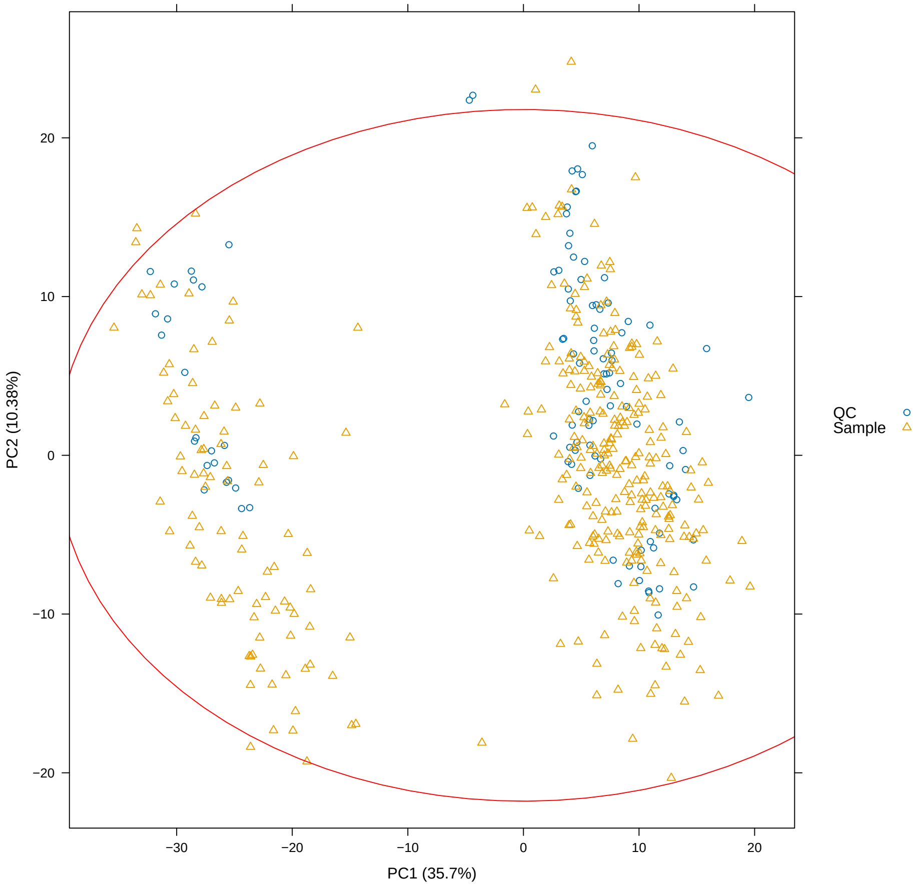
```r
dat <- ret$dat
```

Missing values filling for visualisation `'mv.fill` is in R package `mt`)

```r
dat_fill  <- dat %>% mv.fill(method = "median", ze_ne = T) %>% as_tibble()
```

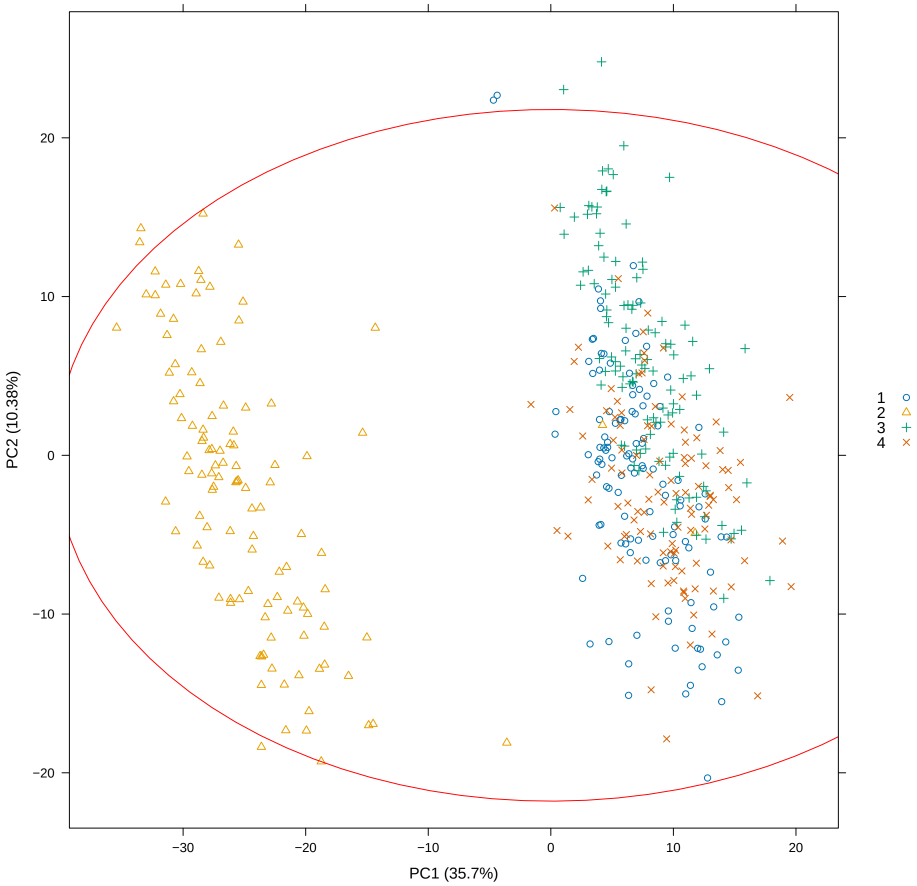
Data screening before signal correction PCA plot for sample types

```r
pcaplot(dat_fill, cls.qc, pcs = c(2, 1), ep = 1)
```



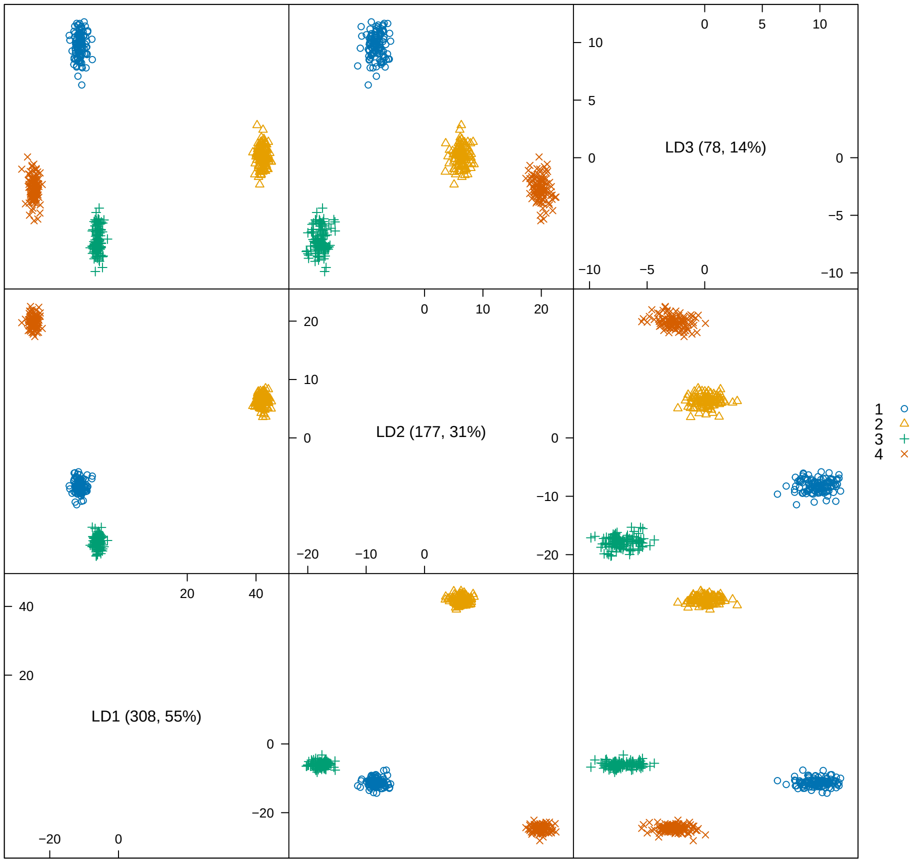
PCA plot for batches

```
pcaplot(dat_fill, cls.bl, pcs = c(2, 1), ep = 1)
```



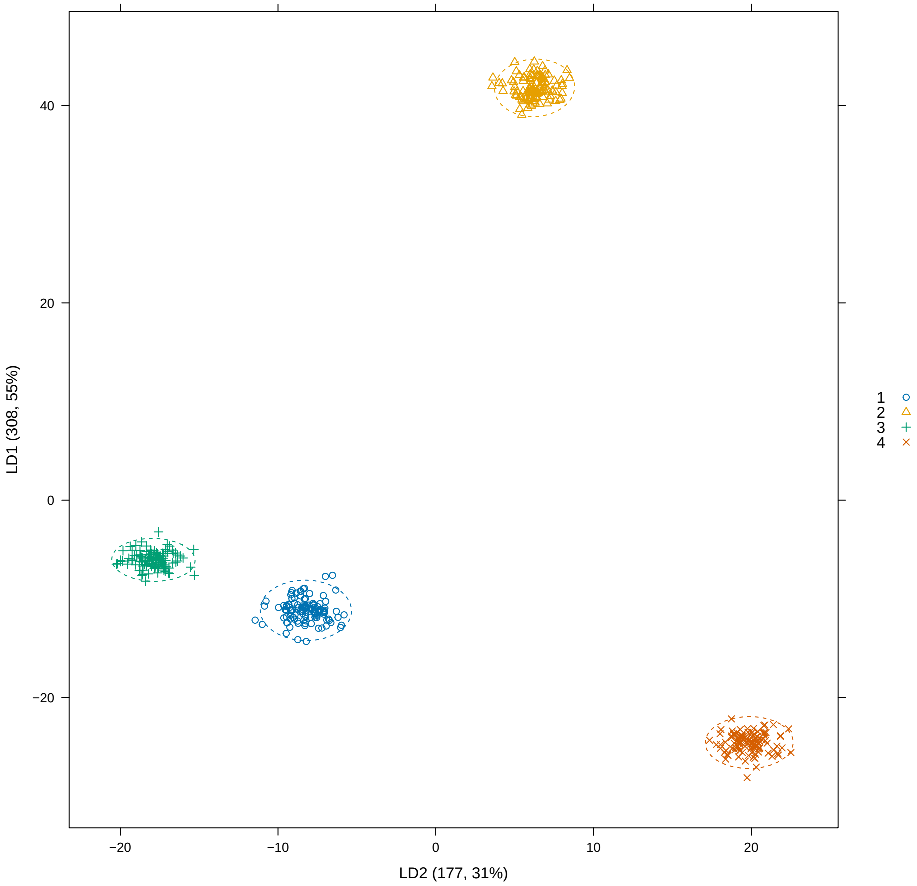
LDA plot for batches

```
plot(pcalda(dat_fill, cls.bl))
```



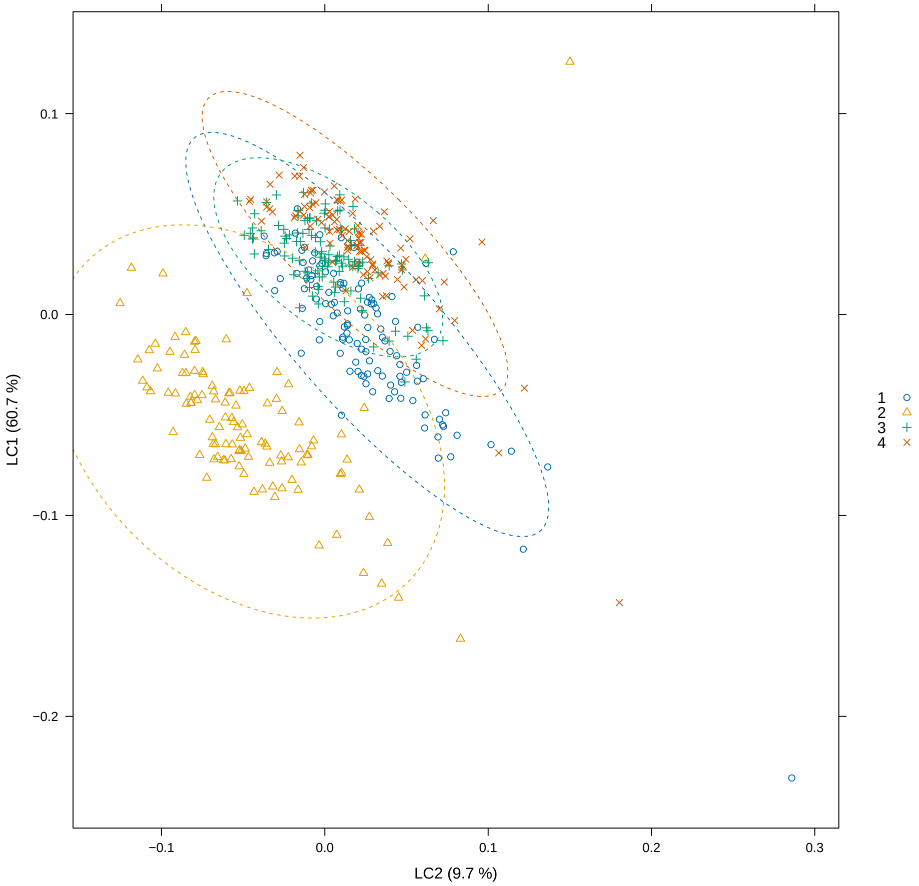
LDA plot of batches: LD1 vs LD2 (only for batch groups larger than 2)

```
plot(pcalda(dat_fill, cls.bl), dimen = c(1:2), ep = 2)
```



PLS plot of batches: LC1 vs LC2

```
plot(plslda(dat_fill, cls.bl), dimen = c(1:2), ep = 2)
```



0.4 Set parameters for QC-RLSC

```
method <- "subtract"  # two methods: "subtract", "divide"
intra <- F            # signal correction within batch or not
opti <- T             # optimise smooth parameter  or not
log10 <- T            # log 10 transform data or not
outl <- T             # outlier detect in qc samples or not
shift <- T            # batch shift or not
```

0.5 QC outlier detection

log transformation

```
if (log10) {
  dat[dat == 0] <- NA
  dat <- log10(dat)
}
```

outlier detection based on QC

```
if (outl) {
  dat <- sapply(dat, function(x) { #' x <- dat[, 6, drop = T]
    qc_ind <- grepl("qc", cls.qc, ignore.case =  TRUE, perl = TRUE)
    ## get median of qc data
    qc_dat <- x[qc_ind]
    qc_median <- median(qc_dat, na.rm = TRUE)
    ## assign other data as NA for QC outlier detection
    tmp <- x
    tmp[!qc_ind] <- NA
    ## QC outlier detection
    out_ind <- outl.det.u(tmp)
    ## assign outlier as qc median
    x[out_ind] <- qc_median
    return(x)
  }) %>% as_tibble()
}
dat
#> # A tibble: 462 x 620
#>       V3   V18   V19   V20   V22   V23   V25   V26   V31   V33   V34   V39   V45
#>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#>  1  5.94 NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA     5.54
#>  2  6.08  5.74  8.18  5.66  4.97  7.46  4.74  7.28  4.88  6.59  6.88  6.36  5.75
#>  3  5.96  5.70  8.15  5.62  4.96  7.47 NA     7.28  4.87  6.64  6.90  6.38  5.71
#>  4  6.04  5.69  8.12  5.60  4.98  7.48  4.75  7.29  4.89  6.61  6.91  6.39  5.78
#>  5  5.99  5.68  8.14  5.61  4.92  7.45  4.71  7.27  4.72  6.57  6.91  6.39  5.75
#>  6  6.04  5.67  8.13  5.60  4.95  7.47  4.74  7.28  4.75  6.56  6.92  6.41  5.76
#>  7  6.05  5.68  8.13  5.60  4.99  7.47  4.75  7.28  4.78  6.56  6.91  6.37  5.76
#>  8  5.95  5.65  8.09  5.57  4.96  7.46  4.73  7.28  4.71  6.56  6.93  6.40  5.73
#>  9  5.97  5.63  8.10  5.57  4.95  7.47 NA     7.28  4.93  6.61  6.90  6.37  5.75
#> 10  6.02  5.61  8.08  5.54  4.98  7.47  4.73  7.29  4.97  6.65  6.91  6.39  5.76
#> # i 452 more rows
#> # i 607 more variables: V48 <dbl>, V51 <dbl>, V66 <dbl>, V68 <dbl>, V71 <dbl>,
#> #   V72 <dbl>, V73 <dbl>, V74 <dbl>, V104 <dbl>, V106 <dbl>, V112 <dbl>,
#> #   V115 <dbl>, V116 <dbl>, V121 <dbl>, V122 <dbl>, V123 <dbl>, V124 <dbl>,
#> #   V125 <dbl>, V126 <dbl>, V128 <dbl>, V134 <dbl>, V138 <dbl>, V140 <dbl>,
#> #   V142 <dbl>, V147 <dbl>, V149 <dbl>, V157 <dbl>, V158 <dbl>, V159 <dbl>,
#> #   V160 <dbl>, V162 <dbl>, V164 <dbl>, V167 <dbl>, V169 <dbl>, V171 <dbl>, ...
```

0.6 QC-RLSC

perform qc-rlsc within each batch or not

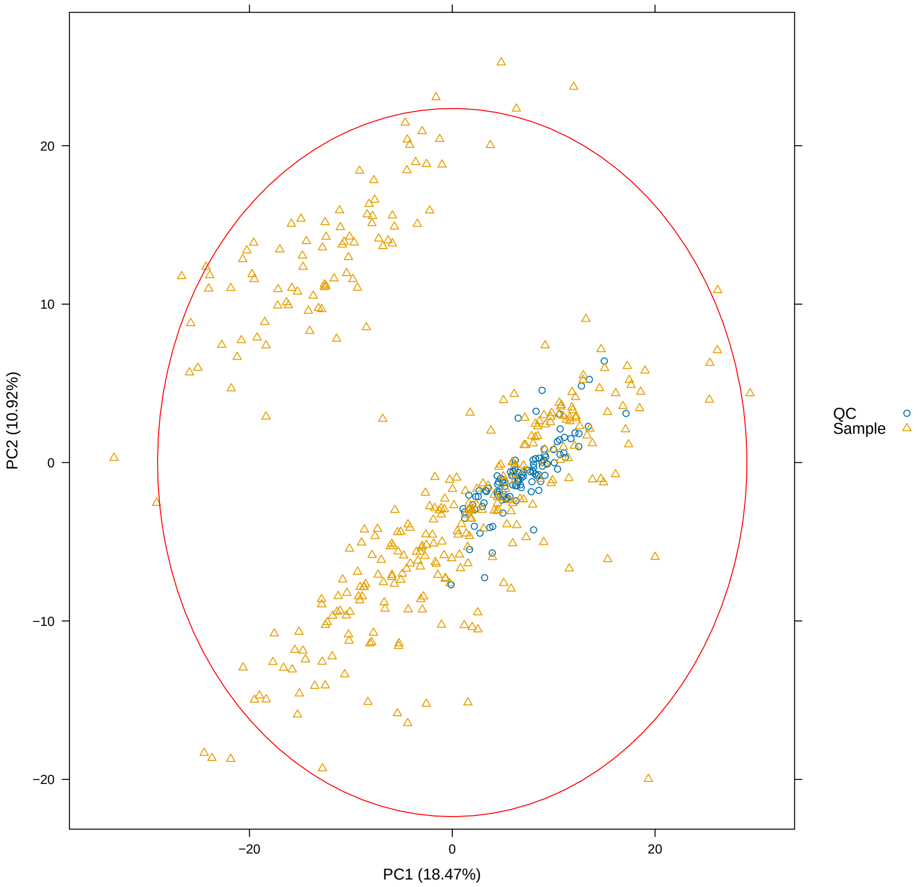
```
tic()
if (!intra) {
  res <- qc.rlsc(dat, cls.qc, method = method, opti = opti)
} else { # do signal correction inside each batch
  res <- lapply(levels(cls.bl), function(x) {
    idx <- cls.bl %in% x
    tmp <- qc_rlsc(dat[idx,], cls.qc[idx], method = method, opti = opti)
  })
  res <- bind_rows(res)
}
toc()
#> 17.11 sec elapsed
```

Data visualisation after signal correction

```
res_fill  <- res %>% mv.fill(method = "median", ze_ne = T) %>% as_tibble()
```

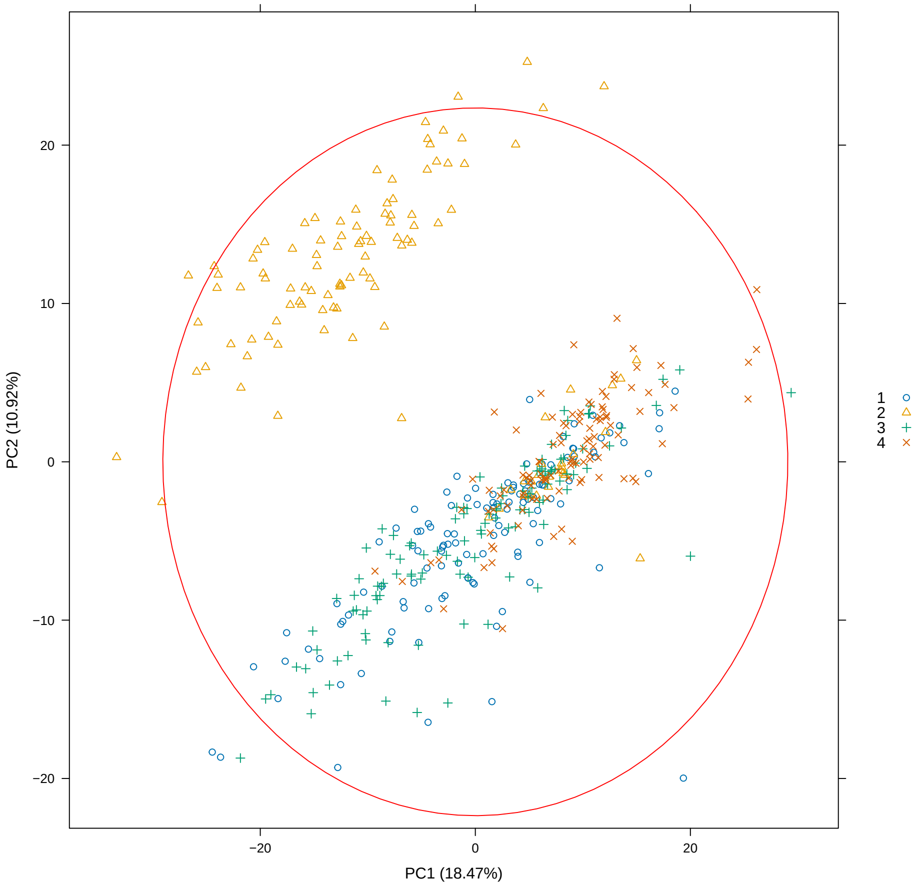
PCA plot for sample types

```
pcaplot(res_fill, cls.qc, pcs = c(2, 1), ep = 1)
```



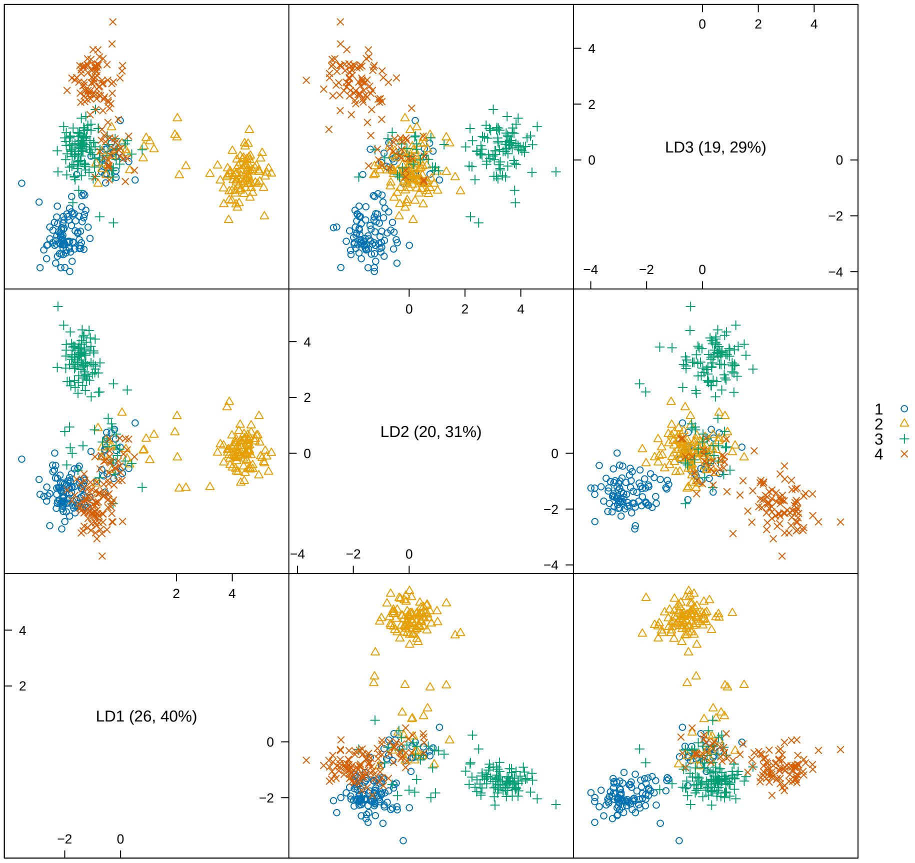
PCA plot for batches

```
pcaplot(res_fill, cls.bl, pcs = c(2, 1), ep = 1)
```



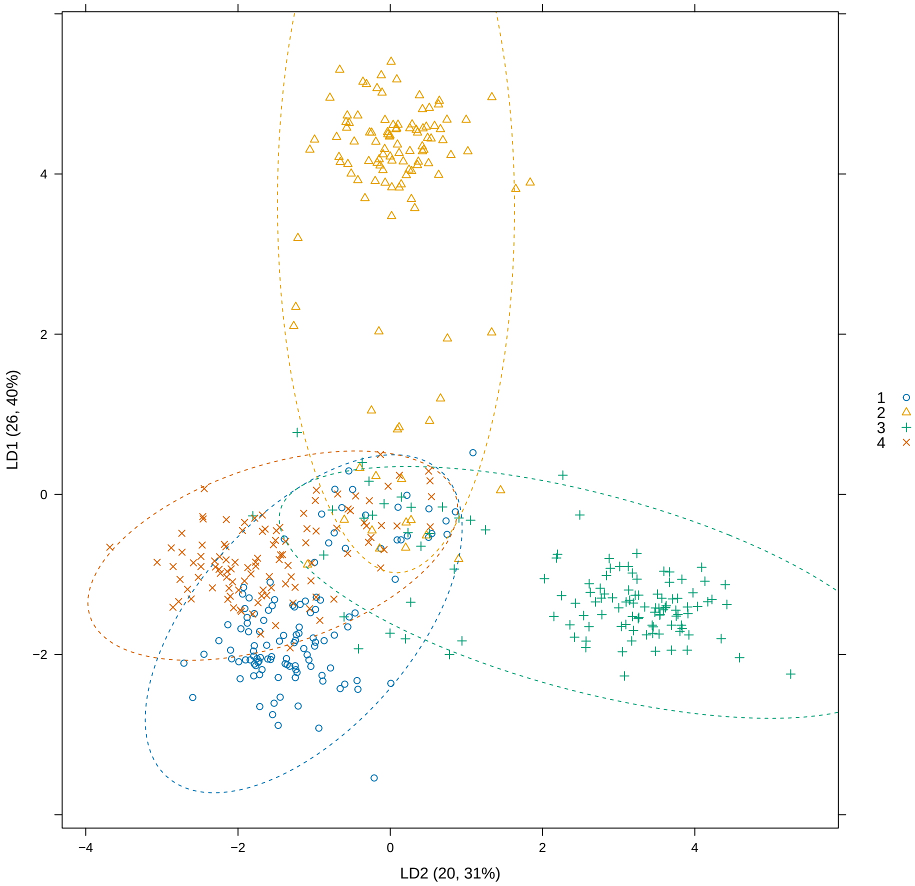
LDA plot for batches

```
plot(pcalda(res_fill, cls.bl))
```



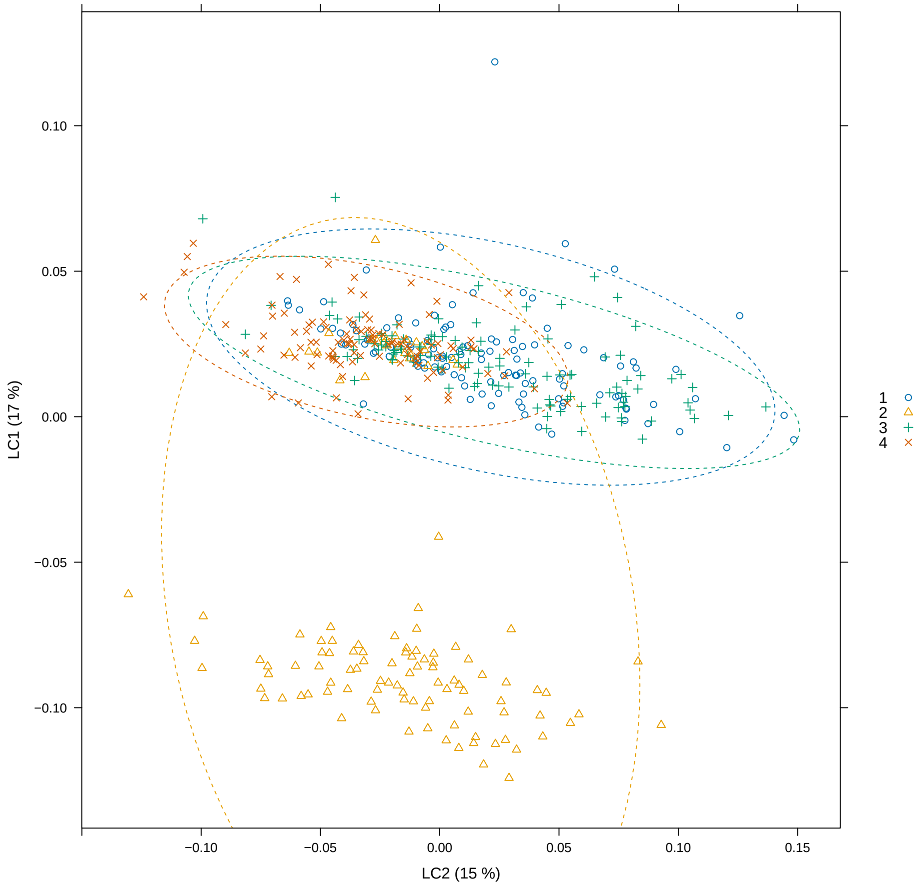
LDA plot of batches: LD1 vs LD2 (only for batch groups larger than 2)

```
plot(pcalda(res_fill, cls.bl), dimen = c(1:2), ep = 2)
```



PLS plot of batches: LC1 vs LC2

```
plot(plslda(res_fill, cls.bl), dimen = c(1:2), ep = 2)
```



0.7 Batch shift

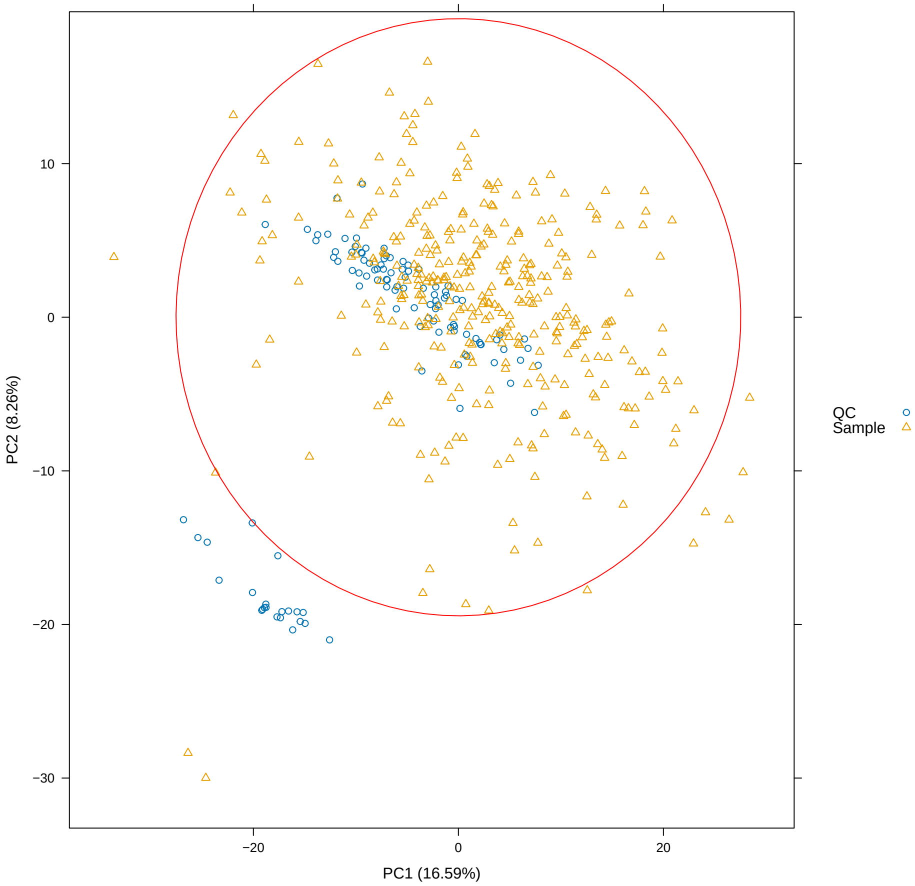
```
if (shift) {
  res <- batch.shift(res, cls.bl, overall_average = T) %>% as_tibble()
}
```

Data visualisation after batch shift

```
res_fill  <- res %>% mv.fill(method = "median", ze_ne = T) %>% as_tibble()
```

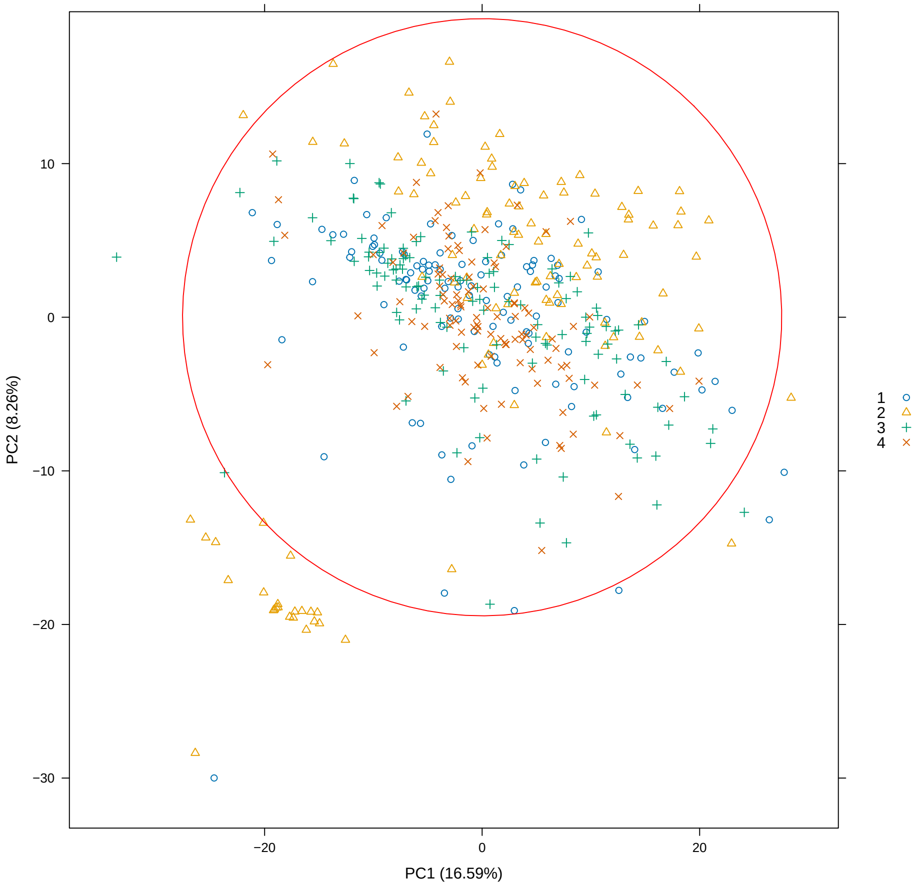
PCA plot for sample types

```
pcaplot(res_fill, cls.qc, pcs = c(2, 1), ep = 1)
```



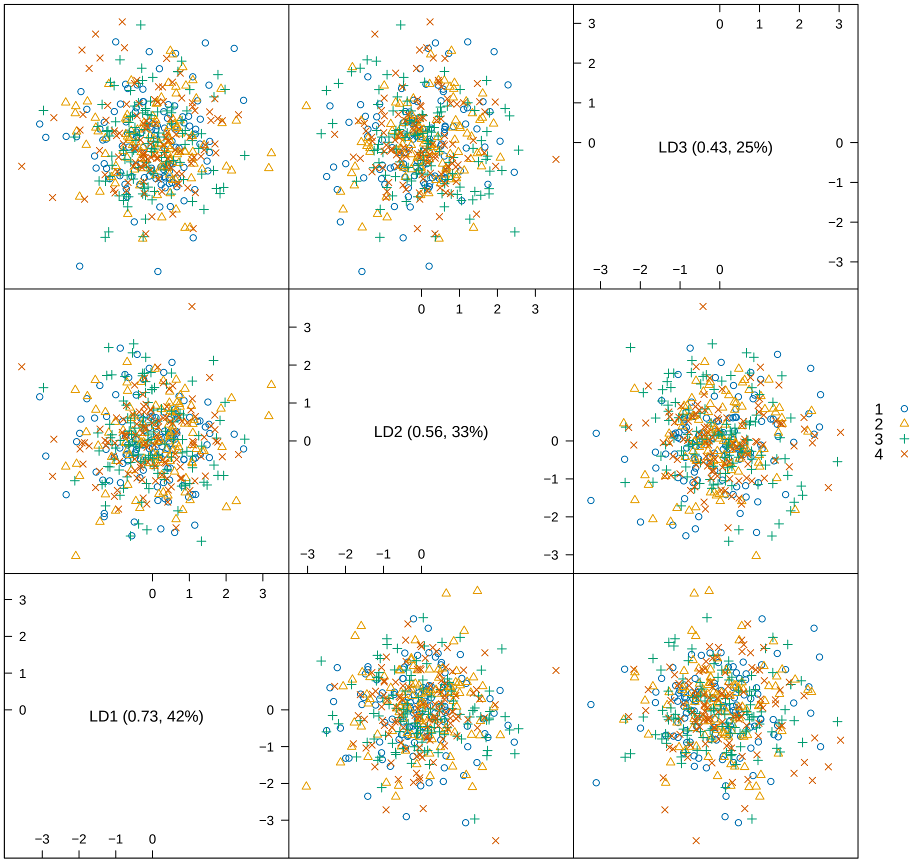
PCA plot for batches

```
pcaplot(res_fill, cls.bl, pcs = c(2, 1), ep = 1)
```



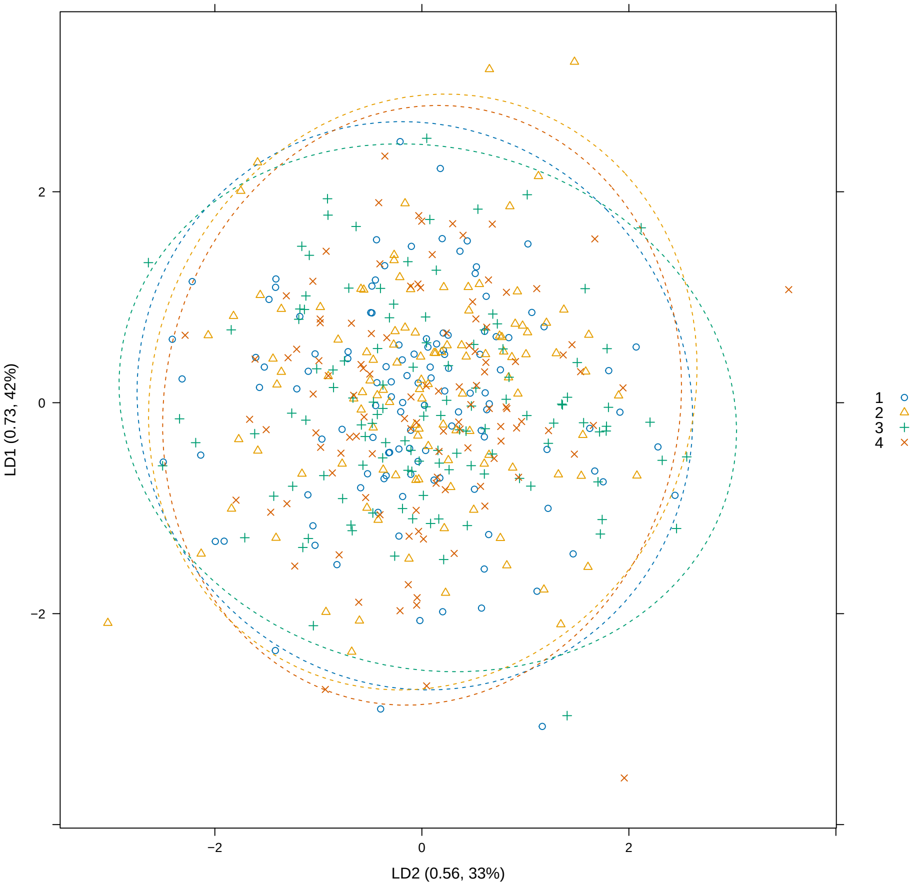
LDA plot for batches

```
plot(pcalda(res_fill, cls.bl))
```



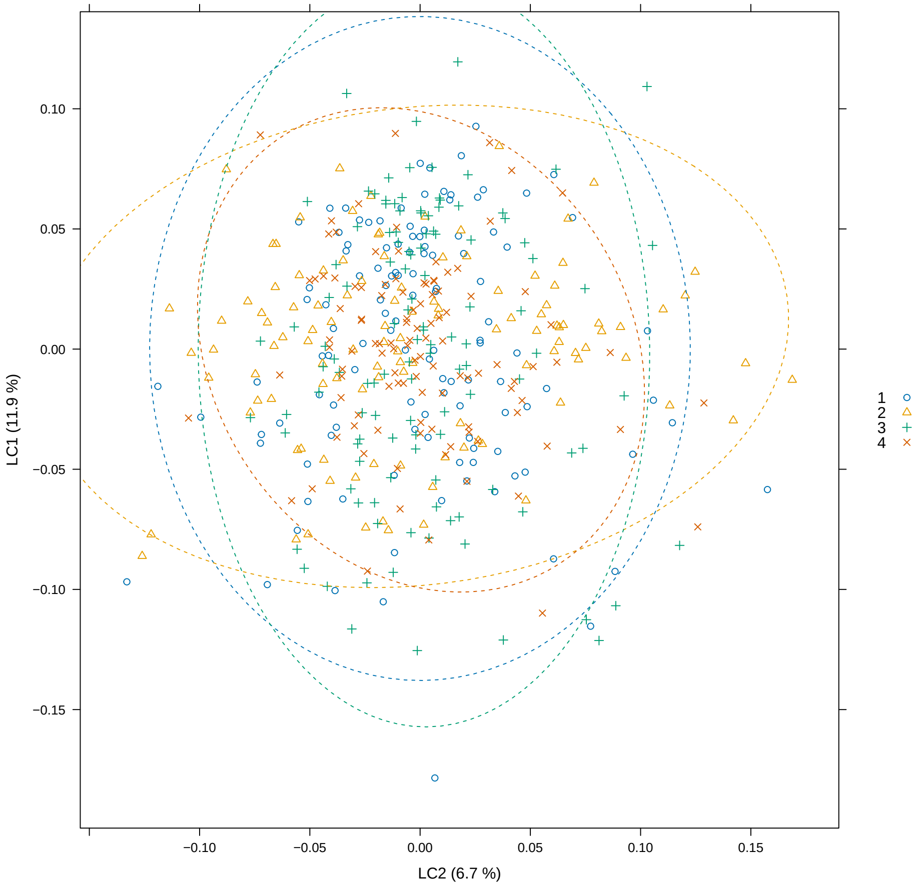
LDA plot of batches: LD1 vs LD2 (only for batch groups larger than 2)

```
plot(pcalda(res_fill, cls.bl), dimen = c(1:2), ep = 2)
```



PLS plot of batches: LC1 vs LC2

```
plot(plslda(res_fill, cls.bl), dimen = c(1:2), ep = 2)
```



0.8 Save results

inverse log10 transformation

```
res <- 10^res %>% as_tibble()

tmp <- list(data =  res, meta = meta)
## write.xlsx(tmp, file = here::here("data", paste0(FILE, "_res.xlsx")),
##            asTable = F, overwrite = T, rowNames = F, colNames = T)
```

0.9 QC-RLSC wrapper function

or use wrapper function `qc_rlsc_wrap` directly

```
## res <- qc.rlsc.wrap(dat, cls.qc, cls.bl, method, intra, opti, log10, outl,
##                      shift)
## tmp <- list(data =  res, meta = meta)
## write.xlsx(tmp, file = here::here("data", paste0(FILE, "_res.xlsx")),
##            asTable = F, overwrite = T, rowNames = F, colNames = T)
```