bbeaR - Milk Allergy Example

Maria Suprun, Randall J. Ellis, Mayte Suárez-Fariñas

5/27/2020

Contents

About

The **Bead-Based Epitope Assay** *(BBEA)* can be used to quantify the amount of epitope- or peptide-specific antibodies (e.g., IgE) in plasma or serum samples. A detailed assay description is outlined in this publication

In this tutorial, we will analyze a dataset of the immuniglobulin (Ig)E profiles in children allergic to milk that were treated with milk oral immunotherapy (OIT). The levels of 66 epitope-specific IgE were measured using *BBEA* in plasma.

We will start by installing *bbeaR* and reading in the *BBEA's* raw data (alternatively, one can load an R dataset that comes with this package). We will look at several quality control (QC) measures and normalize the data. Then we will demonstrate an approach to identify epitope-specific (es)IgE that is different between children treated with Placebo or Omalizumab as adjuvants for milk Oral Immunotherapy (mOIT), using limma modeling framework.

Installation

bbeaR is an R package and is installed using a standard github command:

```
library(devtools)
install_github('msuprun/bbeaR')
```

Loading additional packages that will be used in the analyses.

```
library(bbeaR)
library(plyr)
library(stringr)
library(ggplot2)
library(gridExtra)
library(pheatmap)
library(RColorBrewer)
library(limma)
```

Raw Data Import

47 patients were assayed before and after treatment, for a total of 94 samples, each ran in triplicate on four 96-well plates. The runs additionally included three negative control well, without any sample (aka "Buffer", for background quantification).

The original *.csv* files from the Luminex-200 assay, generated with the xPONENT® software, can be downloaded here [link].

We need to create a plate layout using the *create.plate.db()* function. This layout will be used in the import and some of the plotting functions. The only input to this function is the direction of the plate read (horizorntal or vertical).

```
l <- create.plate.db(direction = "horizontal")
plate.design.db <- l$plate.design.db
plate.design <- l$plate.design
plate.design
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,] "A1" "A2" "A3" "A4" "A5" "A6" "A7" "A8" "A9" "A10" "A11" "A12"
## [2,] "B1" "B2" "B3" "B4" "B5" "B6" "B7" "B8" "B9" "B10" "B11" "B12"
## [3,] "C1" "C2" "C3" "C4" "C5" "C6" "C7" "C8" "C9" "C10" "C11" "C12"
## [4,] "D1" "D2" "D3" "D4" "D5" "D6" "D7" "D8" "D9" "D10" "D11" "D12"
## [5,] "E1" "E2" "E3" "E4" "E5" "E6" "E7" "E8" "E9" "E10" "E11" "E12"
## [6,] "F1" "F2" "F3" "F4" "F5" "F6" "F7" "F8" "F9" "F10" "F11" "F12"
## [7,] "G1" "G2" "G3" "G4" "G5" "G6" "G7" "G8" "G9" "G10" "G11" "G12"
## [8,] "H1" "H2" "H3" "H4" "H5" "H6" "H7" "H8" "H9" "H10" "H11" "H12"
```

Then read all four *.csv* files.
Note: a separate tutorial (Egg Allergy Example) has an example of importing and processing a single plate.

```
file_names <- dir(path = "./", pattern=".csv",
                  all.files = FALSE, full.names = TRUE)
bbea <- bbea.read.csv.all(file_names[grepl("MOIT_IgE",file_names)])
```

```
## [1] "Reading file: MOIT_IgEplate_02.csv"
## [1] "Reading file: MOIT_IgEplate_03.csv"
## [1] "Reading file: MOIT_IgEplate_04.csv"
## [1] "Reading file: MOIT_IgEplate01.csv"
```

The *bbea* list object contains several elements, extracted from the assay's output.

```
names(bbea)
```

```
## [1] "Median"    "NetMFI"    "Count"     "pData"     "AssayInfo"
```

The **Median**, **NetMFI**, and **Count** are matrices with rows as Analytes (epitopes) and columns as Samples.

The **Median** are the Median Flourescence Intensities (MFIs) and the **NetMFI** are the Medians normalized to background. In our example, **Median** and **NetMFI** have exactly same values, since normalization was not selected during the assay run. This post has more details about the Luminex outputs.

```
bbea$Median[1:5, 1:2]
```

```
##            Plate2.1.1.A1. Plate2.9.1.A2.
## Analyte 10              5              4
## Analyte 12              1              1
## Analyte 14              2              2
## Analyte 15              5              4
## Analyte 16              1              0
```

The **Count** are the numbers of beads counted per analyte, and is important quality control measure.

```
bbea$Count[1:5, 1:2]
```

```
##            Plate2.1.1.A1. Plate2.9.1.A2.
## Analyte 10            161            149
## Analyte 12            147            140
## Analyte 14            191            165
## Analyte 15            175            143
## Analyte 16            194            144
```

The **AssayInfo** saves parameters of the assay.

```
bbea$AssayInfo[1:15, 1:2]
```

```
##                          V1
## 1                   Program
## 2                     Build
## 3                      Date
## 4
## 5                        SN
## 6                     Batch
## 7                   Version
## 8                  Operator
## 9              ComputerName
```

```
## 10              Country Code
## 11              ProtocolName
## 12           ProtocolVersion
## 13       ProtocolDescription
## 14 ProtocolDevelopingCompany
## 15              SampleVolume
##                                                                  V2
## 1                                                           xPONENT
## 2                                                          3.1.971.0
## 3                                                            7/22/15
## 4
## 5                                                      LX10010124401
## 6                                                             Plate2
## 7                                                                  1
## 8
## 9                                                       XPONENT31-PC
## 10                                                               409
## 11                                    Milk OIT IgE Plate2 07.22.15
## 12                                                                 1
## 13 Run by Gustavo Gimenez, Peiatric Allergy and Immunology Department
## 14
## 15                                                           90 uL
```

Finally, *bbea.read.csv()* creates a **phenotype (p)Data** that has some basic information about the samples and the assay run.

```
colnames(bbea$pData)
```

```
##  [1] "Location"          "Sample"            "filename"
##  [4] "File"              "Plate"             "SampleNumber"
##  [7] "Well.Number"       "Well.Letter"       "Well_coord"
## [10] "print.plate.order" "letters_numeric"   "Plate.Date"
## [13] "Plate.Time"        "Plate.TimeHR"      "CountSum"
## [16] "CountMean"         "CountMin"          "CountMax"
```

```
bbea$pData[1:2, 1:2]
```

```
##                Location   Sample
## Plate2.1.1.A1.  1(1,A1) Unknown1
## Plate2.9.1.A2.  9(1,A2) Unknown1
```

We can now add extrenal information about the samples and analytes.
Load already imported data that inlcudes **p**henotype (clinical) **d**ata *PD* and an annotation file *Annot*. Note: this will also load a *bbea* list object (generated in the previous steps).

```
data(Milk)
```

The annotation **Annot** dataset contains the mapping of Luminex beads (Analytes) to the peptides/eptiopes.

```
dim(Annot)
```

```
## [1] 66  6
```

```
head(Annot)
```

```
##                Analyte     Peptide Protein lableName Bead PeptideName
## alphas1-p01  Analyte 5 alphas1-p01 alphas1    a1.p01    5 alphas1-p01
## alphas1-p02 Analyte 58 alphas1-p02 alphas1    a1.p02   58 alphas1-p02
## alphas1-p03  Analyte 7 alphas1-p03 alphas1    a1.p03    7 alphas1-p03
## alphas1-p04  Analyte 8 alphas1-p04 alphas1    a1.p04    8 alphas1-p04
## alphas1-p05 Analyte 60 alphas1-p05 alphas1    a1.p05   60 alphas1-p05
## alphas1-p06 Analyte 10 alphas1-p06 alphas1    a1.p06   10 alphas1-p06
```

Changing the *bbea* object to have peptide names instead of analyte numbers.

```
bbea <- bbea.changeAnnotation(bbea.obj = bbea,
                              annotation = Annot,
                              newNameCol = "Peptide",
                              AnalyteCol = "Analyte")
bbea$Median[1:5, 1:2]
```

```
##             Plate1.1.1.A1. Plate1.2.1.A2.
## alphas1-p06              4              4
## alphas1-p08             18             19
## alphas1-p09             35             33
## alphas1-p13             22             21
## alphas1-p17              2              2
```

The **PD** dataset has clinical information about our samples.

```
dim(PD)
```

```
## [1] 285   5
```

```
head(PD)
```

```
##   SUBJECT.ID    Visit Treatment  Plate  Location
## 1     MILK01 Baseline   Placebo Plate1 34(1,C10)
## 2     MILK01 Baseline   Placebo Plate1 35(1,C11)
## 3     MILK01 Baseline   Placebo Plate1 36(1,C12)
## 4     MILK01  Month32   Placebo Plate1 46(1,D10)
## 5     MILK01  Month32   Placebo Plate1 47(1,D11)
## 6     MILK01  Month32   Placebo Plate1 48(1,D12)
```

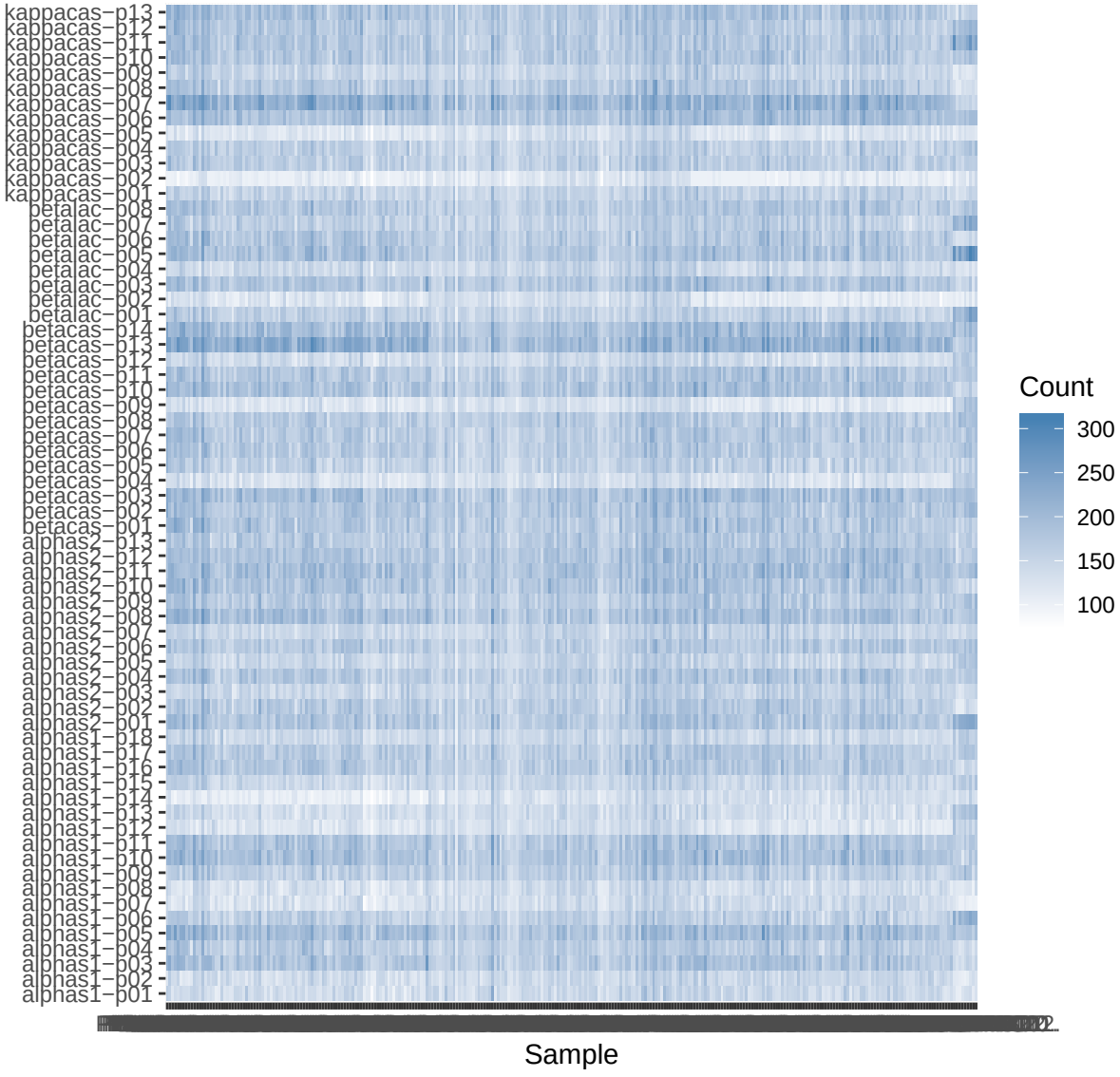
We will clean up the pData and then merge it with PD. This step is not necessary, but will be useful when we do statistical modelling.

```
bbea$pData$Rows <- rownames(bbea$pData)
bbea$pData <- mutate(merge(bbea$pData,PD,by=c("Location","Plate"),all.x=T),
                     Sample=mapvalues(SUBJECT.ID,NA,"Buffer"))
rownames(bbea$pData) <- bbea$pData$Rows
```

Quality Control of the Raw Data

We want to make sure that there are no missing samples or analytes. This would be reflcected by the very low counts (<25).
The heatmap shows that all samples and analytes are were included.

```
bbea.QC.heatmap.counts(bbea,
                       getlog2=FALSE,
                       filename="QC.CountsHeatmap.pdf",
                       plateVar="Plate", ann=NULL)
```

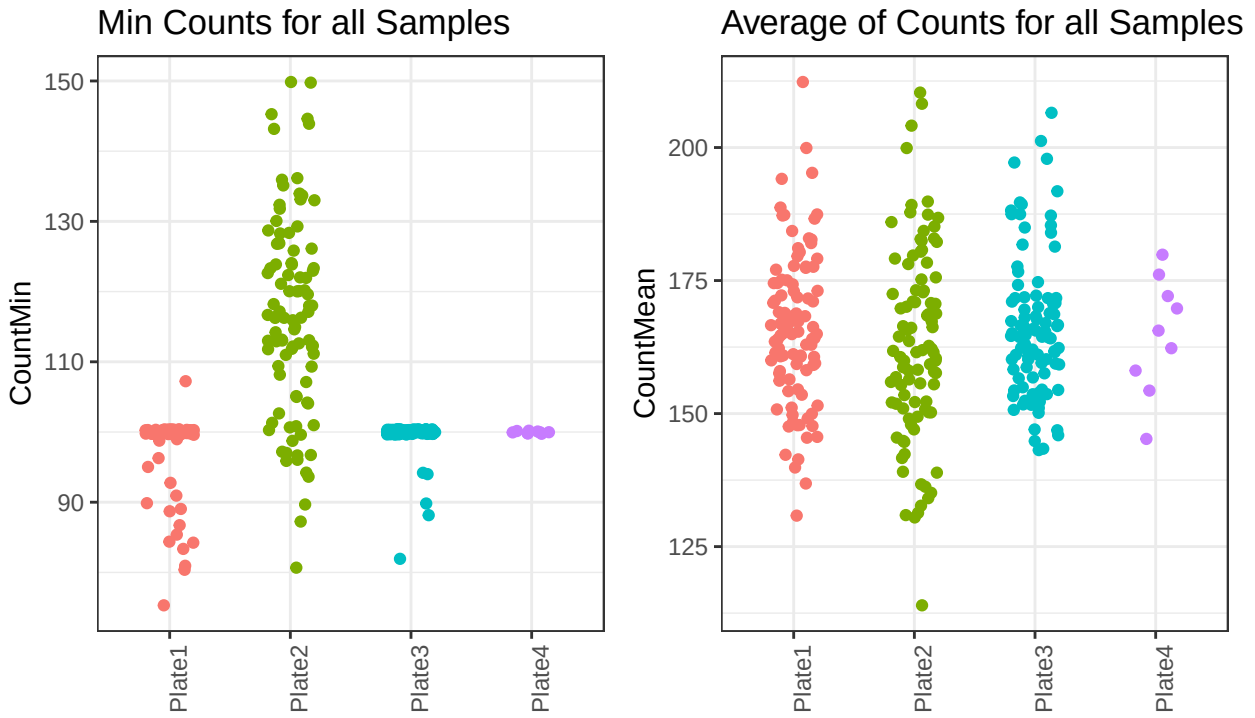


All counts seem to be high and we don't see any specific sample or epitope to be missing.
However, when having multiple plates, it might be more useful to have a counts heatmap for each plate separately. This can be achived by running *bbea.QC.heatmap.counts.byPlate()* function.

```
bbea.QC.heatmap.counts.byPlate(bbea, getlog2=FALSE,
                               filename="QC.CountsHeatmap",
                               plateVar="Plate", ann=NULL, he=7, wi=9)
```

Now we look at the overall distribution of counts: the minimum count is ~75, which is pretty good.

```
p<-bbea.QC.Samples(bbea,
                   filename = "QC.",
                   plateVar = 'Plate',
                   gt = 25)
grid.arrange(p$pmin, p$pavg, nrow=1)
```



Our counts looks good, so we don't need to exclude any samples. However, if this were not the case, samples with low counts can be removed using the *bbea.subset()* function, only keeping samples with average counts > 25.

```
bbea.sub <- bbea.subset(bbea, statement = (bbea$pData$CountMean > 25))
```

Data Normalization

We convert the **Median** to normalized **nMFI** by taking the $\log_2$ of values and subtracting the average of the background wells. Note: the *Sample* column of the **pData** should include a *"Buffer"* string in the wells dedicated for the background.

```
bbeaN <- MFI2nMFI(bbea,
                  offset = 0.5, # a constant to add to avoid taking a log of 0
                  rmNeg = TRUE) # if a value of a sample is below background, assign "0"
```

```
## [1] "Plate1"
## [1] "Plate2"
## [1] "Plate3"
## [1] "Plate4"
```

```
names(bbeaN)
```

```
## [1] "nMFI"      "NetMFI"    "Count"     "pData"     "AssayInfo"
```

```
bbeaN$nMFI[1:5, 1:2]
```

```
##             Plate1.1.1.A1. Plate1.2.1.A2.
## alphas1-p06       2.641604       2.641604
## alphas1-p08       4.681133       4.757081
## alphas1-p09       5.621426       5.537768
## alphas1-p13       4.435211       4.369623
## alphas1-p17       1.793607       1.793607
```

R object of class ExpressionSet *eset* is convenient for a high-thoughtput data analysis.
bbea object can be converted to eset:

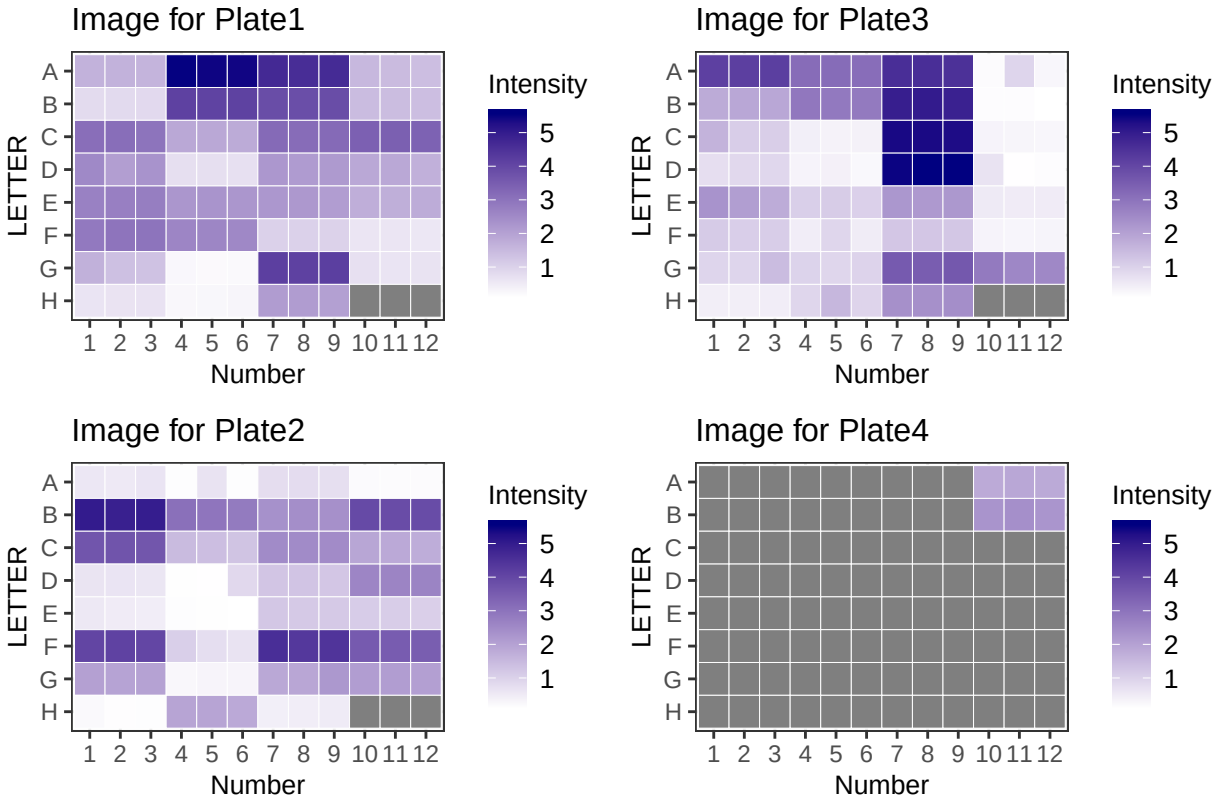
```
eset <- nMFI2Eset(nMFI.object = bbeaN)
eset
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 66 features, 285 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: Plate1.1.1.A1. Plate1.2.1.A2. ... Plate4.24.1.B12. (285
##     total)
##   varLabels: Location Plate ... Treatment (22 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

QC of Normalized Data

We can print the layouts of all experimental plates, to quicky do a visual inspection of the samples.

```
Image.Plate(bbeaN)
```



In this figure, we can clearly see triplicates and several wells (grey) that were used for the background calculaiton. We can also notice that laste plate (plate #4) had only two samples.

Batch Effect

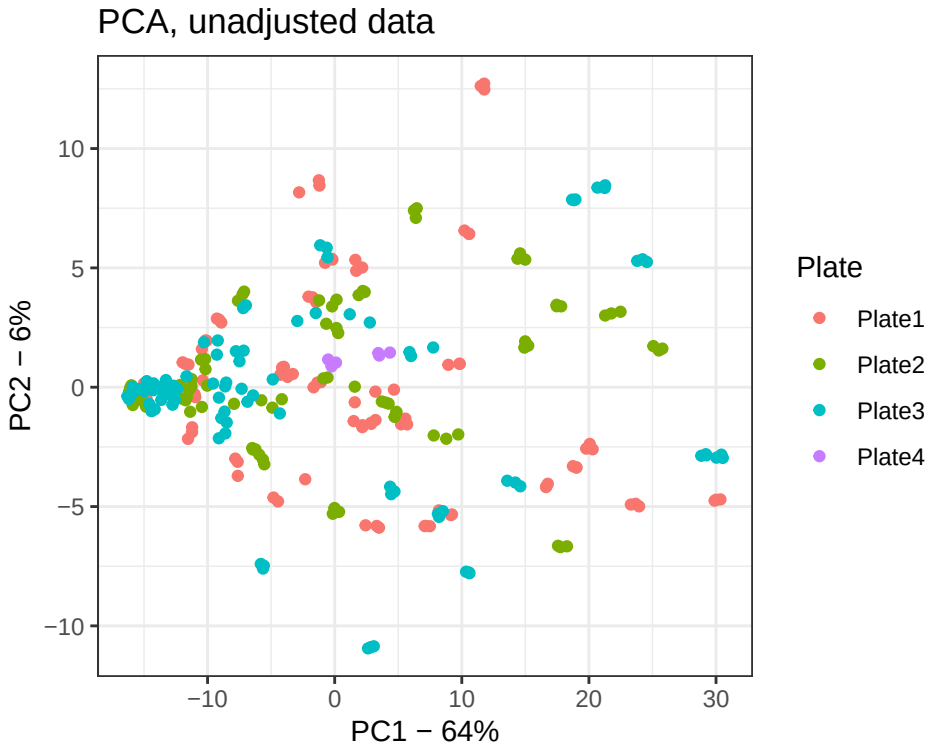
Batch effects are a well-known phenomenon in the high-throughput expreiments, i.e. Microarrays, RNAseq, Luminex. For the **BBEA**, batch effects are individual microplate runs. Those are the effects that capture experimental rather than biological variability. Batch effetcs are easy to detetc and eliminate, if experimental conditions are randomized across plate runs.

Principal Component Analysis (PCA) can be used to viually inspect the data.

```
pca.db<-getPCAs(eset)
pca.db$varex[1:10]
```

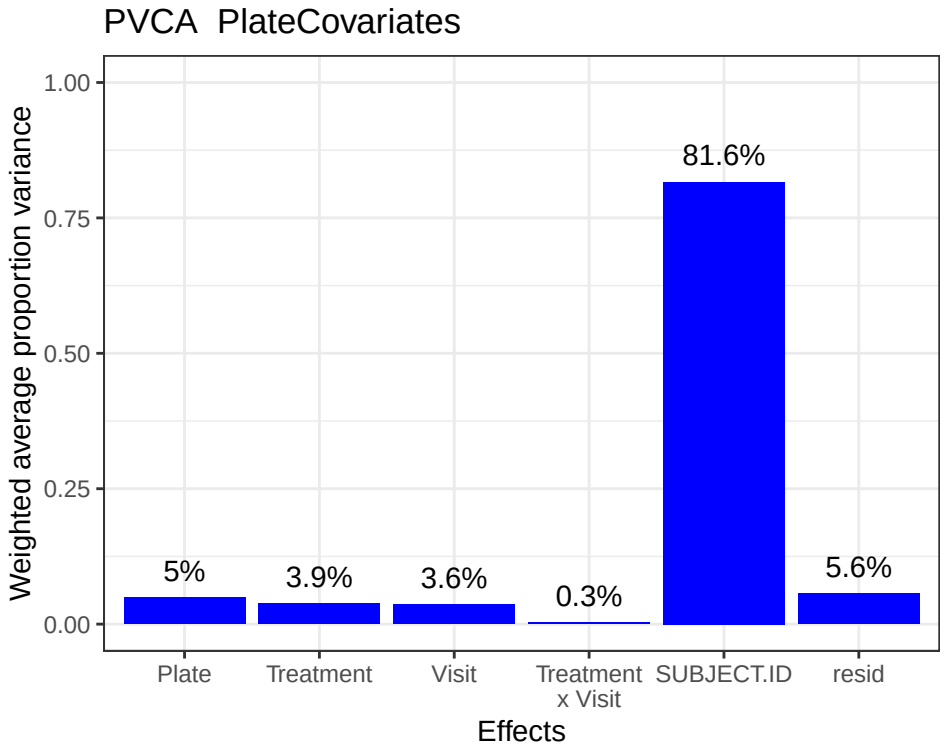
```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
##   64    6    5    3    2    2    2    2    1    1
```

```
ggplot(pca.db$db,aes(x=PC.1,y=PC.2,color=Plate)) + geom_point() + theme_bw() +
  labs(x=paste0("PC1 - ",pca.db$varex[1],"%"),
       y=paste0("PC2 - ",pca.db$varex[2],"%"),
       title='PCA, unadjusted data')
```



Principal Variance Component Analysis (PVCA) to quantify the amount of varibaility attributed to the plate. *pvcaBatchAssess.bbea()* function is based on the pvca R package available through Bioconductor.

```
pvca.obj<-pvcaBatchAssess.bbea(eset,
                               threshold=0.8,
                               batch.factors=c('Plate','Treatment','Visit',"SUBJECT.ID"),
                               include.inter='Treatment:Visit')
pvca.plot(pvca.obj, fname='PVCA.Plate.Covariates', ht=4, wd=5.5,
          order=c('Plate','Treatment','Visit','Treatment x Visit',"SUBJECT.ID",'resid'))
```



The PVCA analysis shows that 5% of the variability is attributed to the "plate" effect. There are many ways to remove this variability. We will show the exmaple of fitting a limma model for each epitope with Plate as a covariate and then subtracting the plate coeffcient. More detailed desciropn of this method is described in this publication.

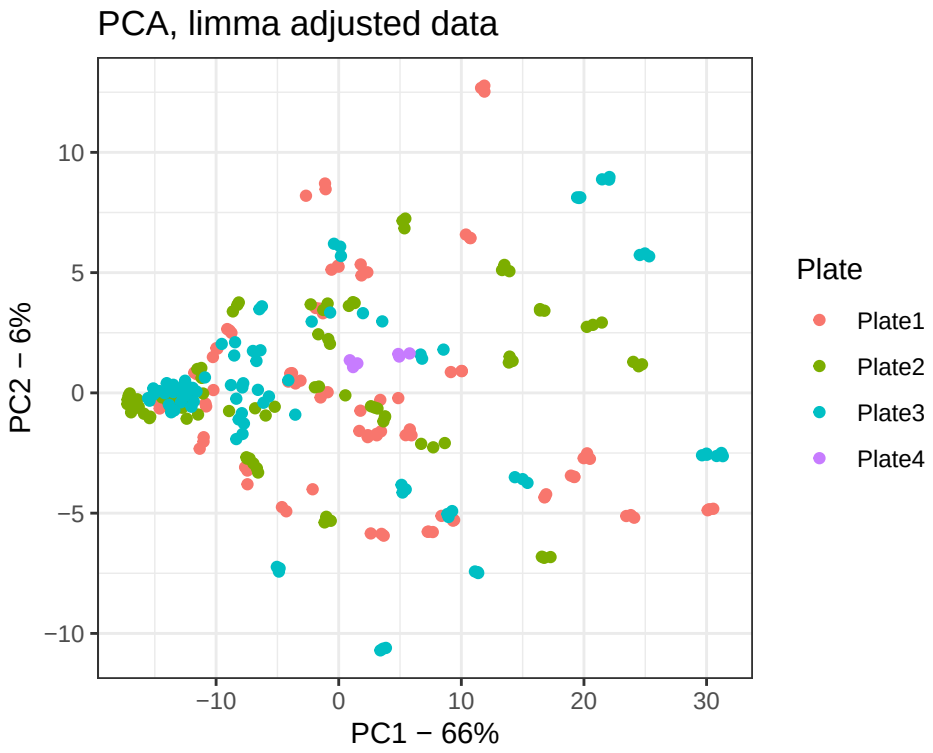
```
design <- model.matrix(~ Plate + Visit*Treatment, data=pData(eset))
cor <- duplicateCorrelation(eset, design, block=eset$SUBJECT.ID) # this estimates the correlation betwe
fit<-lmFit(exprs(eset), design, block=eset$SUBJECT.ID, correlation=cor$consensus)
coefs2adjust <- colnames(fit$coefficients)[grep('Plate', colnames(fit$coefficients))]
adj.exprs <- exprs(eset) - fit$coefficients[,coefs2adjust] %*% t(design[,coefs2adjust])
eset.lm <- eset
exprs(eset.lm) <- adj.exprs
```

PCA after the adjustment looks almost identical, as oftentimes the batch effect is not easily detectable "by eye".

```
pca.db<-getPCAs(eset.lm)
pca.db$varex[1:10]
```

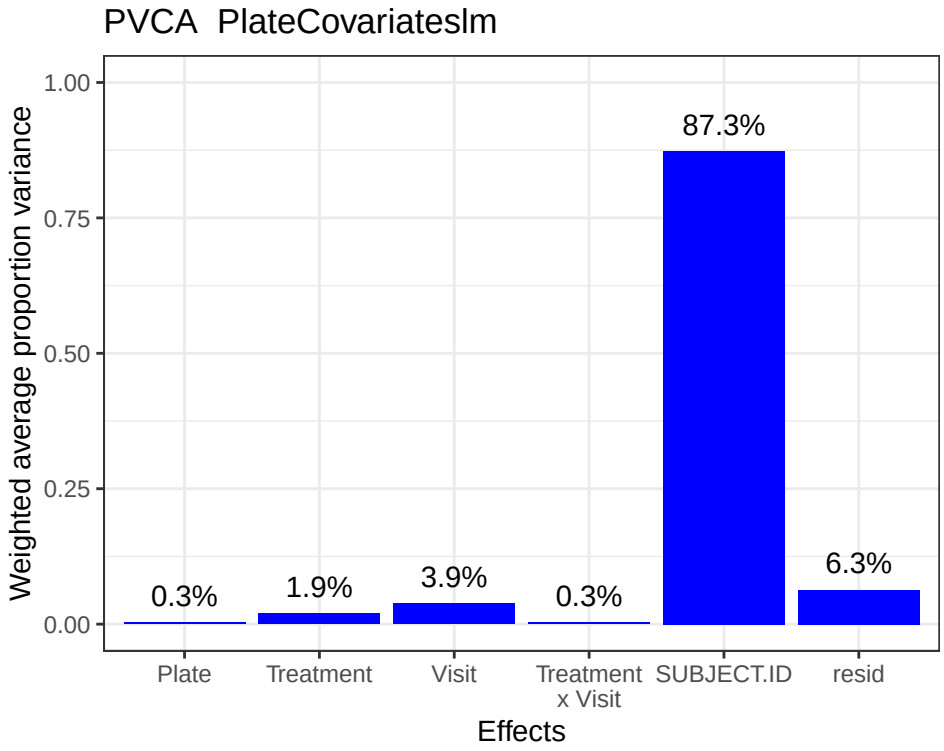
```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
##   66    6    4    3    2    2    2    2    1    1
```

```
ggplot(pca.db$db,aes(x=PC.1,y=PC.2,color=Plate)) + geom_point() + theme_bw() +
  labs(x=paste0("PC1 - ",pca.db$varex[1],"%"),
       y=paste0("PC2 - ",pca.db$varex[2],"%"),
       title='PCA, limma adjusted data')
```



However, the PVCA analysis shows that after the adjustment, "plate" accounts for 0.3% of the overall variability.

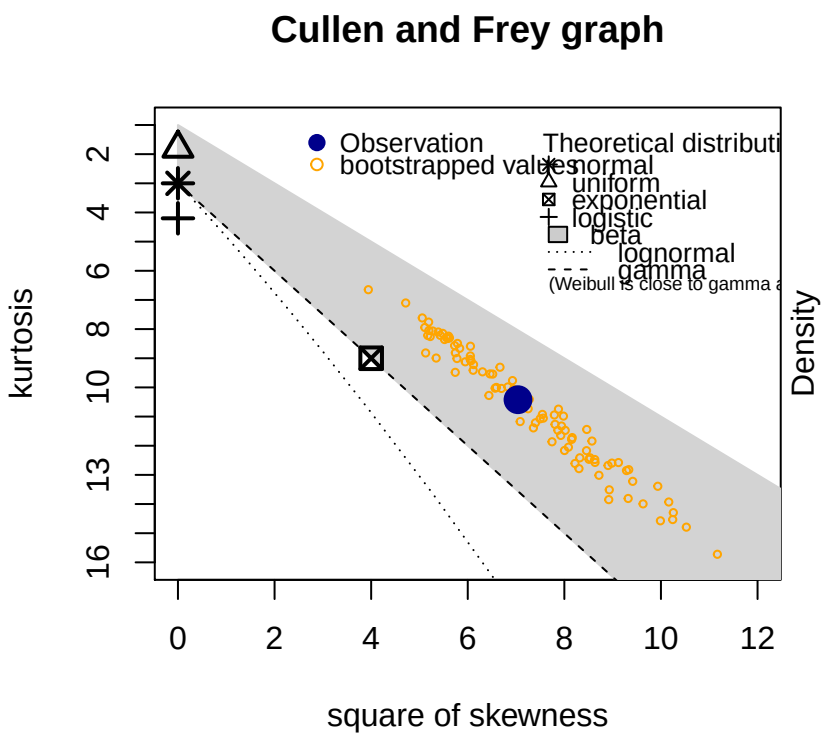
```
pvca.obj<-pvcaBatchAssess.bbea(eset.lm,
                               threshold=0.8,
                               batch.factors=c('Plate','Treatment','Visit',"SUBJECT.ID"),
                               include.inter='Treatment:Visit')
pvca.plot(pvca.obj, fname='PVCA.Plate.Covariates.lm', ht=4, wd=5.5,
          order=c('Plate','Treatment','Visit','Treatment x Visit',"SUBJECT.ID",'resid'))
```

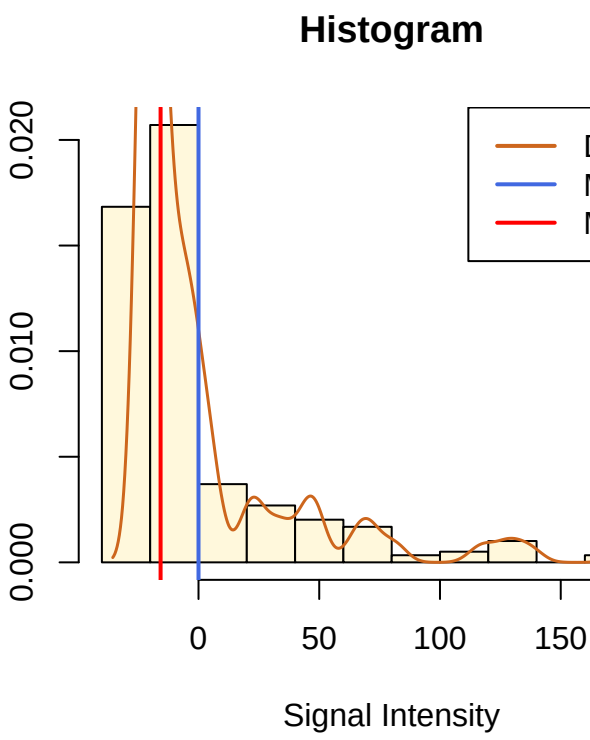


Distributions

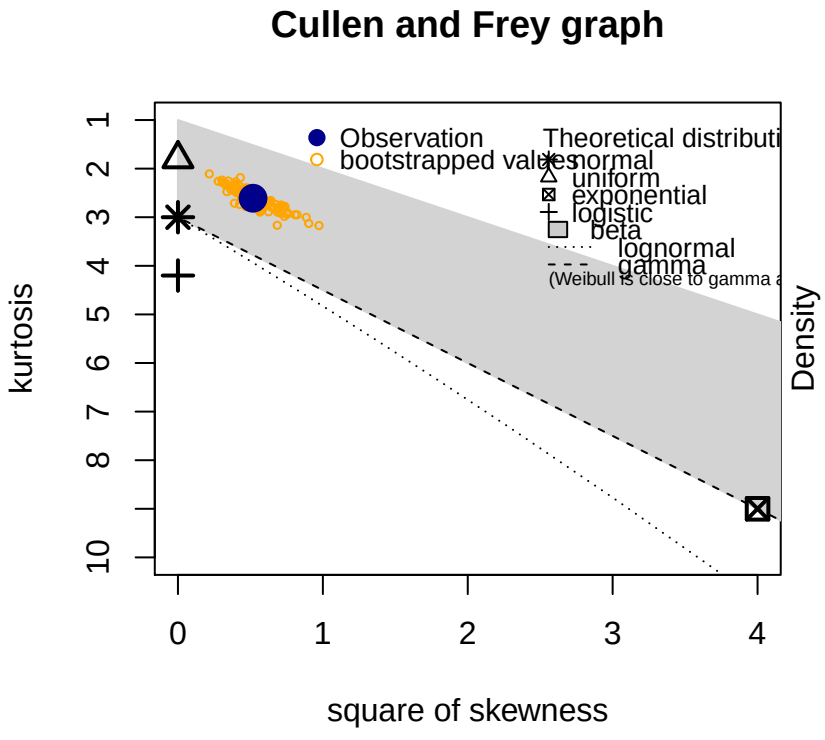
Cullen-Frey plots can be used to evaluate the distribution of the data. It shows how the skewness and kurtosis of our data compare to the theoretical distributions.
CullenFreyPlot() function is a wrapper of the fitdistrplus::descdist(). Since there are different levels of the antibody to each peptide, the data will be scaled before plotting. Y-axis of the boxplot represents a mean MFI or nMFI of 50 scaled peptides.

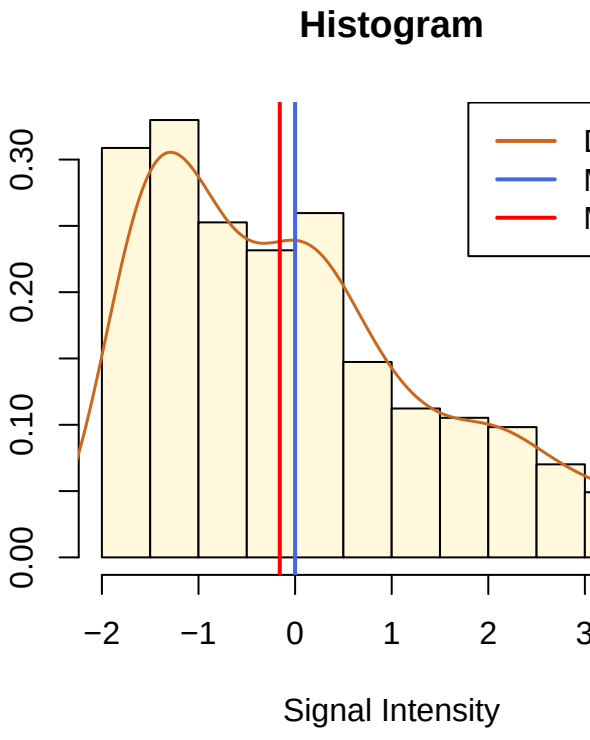
```
adjMFI <- t(apply(as.matrix(bbea$Median), 1, function(x){x - mean(x, na.rm=T)}))
adjnMFI<-t(apply(as.matrix(exprs(eset)), 1, function(x){x - mean(x, na.rm=T)}))

CullenFreyPlot(adjMFI, filename = "QC.CullenFrey.MFI")
```






```
CullenFreyPlot(adjnMFI, filename = "QC.CullenFrey.nMFI")
```





We can see that while both MFI and nMFI data are skewed, nMFI data is closer to log-normal rather than exponential distributions.

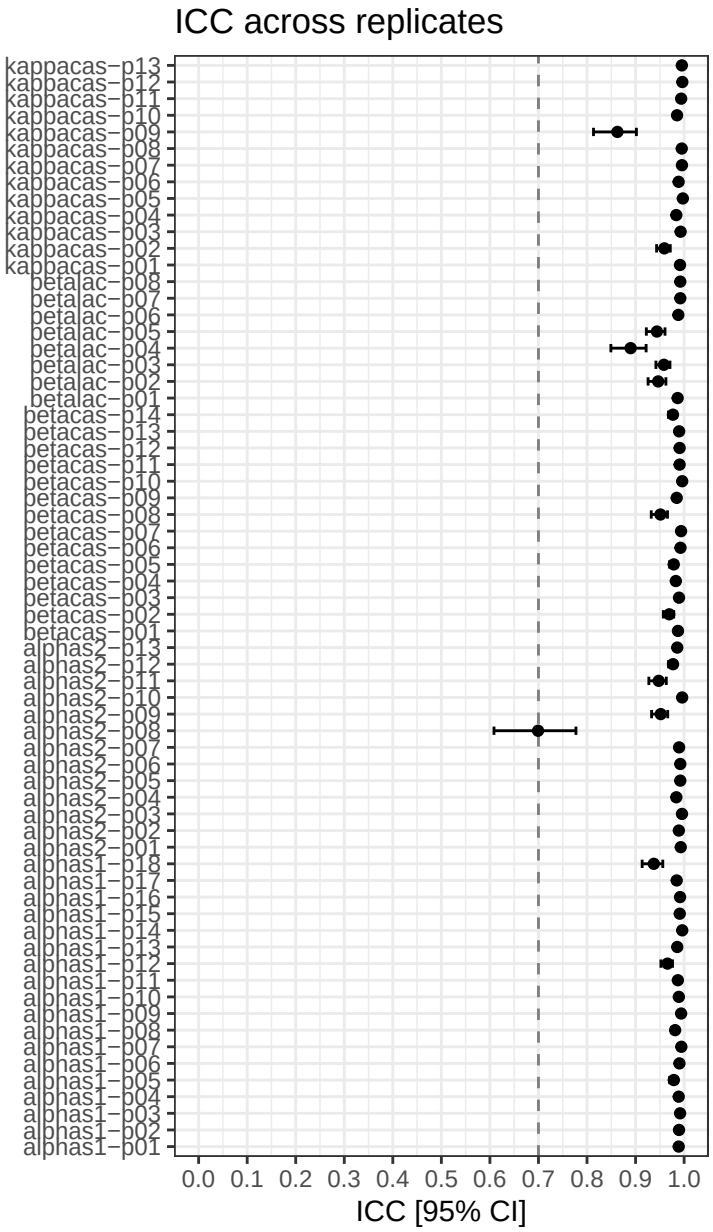
Technical Replicates

Intraclass Correlation Coefficient (ICC) is used to evaluate agreement among technical replicates for each peptide, where 0 means no agreement, and 1 is a perfect agreement.
The function returns the ICC and a 95% confidence interval.

```
icc.db <- getICCbyPeptide(eset, UR=c("SUBJECT.ID","Visit")) # what is the unit of replication? in this
head(icc.db)
```

```
##                 Peptide       ICC       LCI       UCI
## alphas1-p06 alphas1-p06 0.9903578 0.9864010 0.9933039
## alphas1-p08 alphas1-p08 0.9815516 0.9741442 0.9871304
## alphas1-p09 alphas1-p09 0.9938350 0.9913312 0.9957099
## alphas1-p13 alphas1-p13 0.9858467 0.9801419 0.9901349
## alphas1-p17 alphas1-p17 0.9847886 0.9786641 0.9893949
## alphas1-p18 alphas1-p18 0.9375927 0.9135439 0.9560780
```

```
ggplot(icc.db, aes(x = Peptide, y = ICC)) +
  geom_hline(yintercept = 0.7, color = "grey50", linetype = 2) +
  geom_point() +
  geom_errorbar(aes(ymax = UCI, ymin = LCI),width = 0.5) +
  scale_y_continuous(limits = c(0, 1), breaks = seq(0, 1, 0.1)) +
  labs(x = "", y = ("ICC [95% CI]"), title = "ICC across replicates") +
  theme_bw() + coord_flip()
```



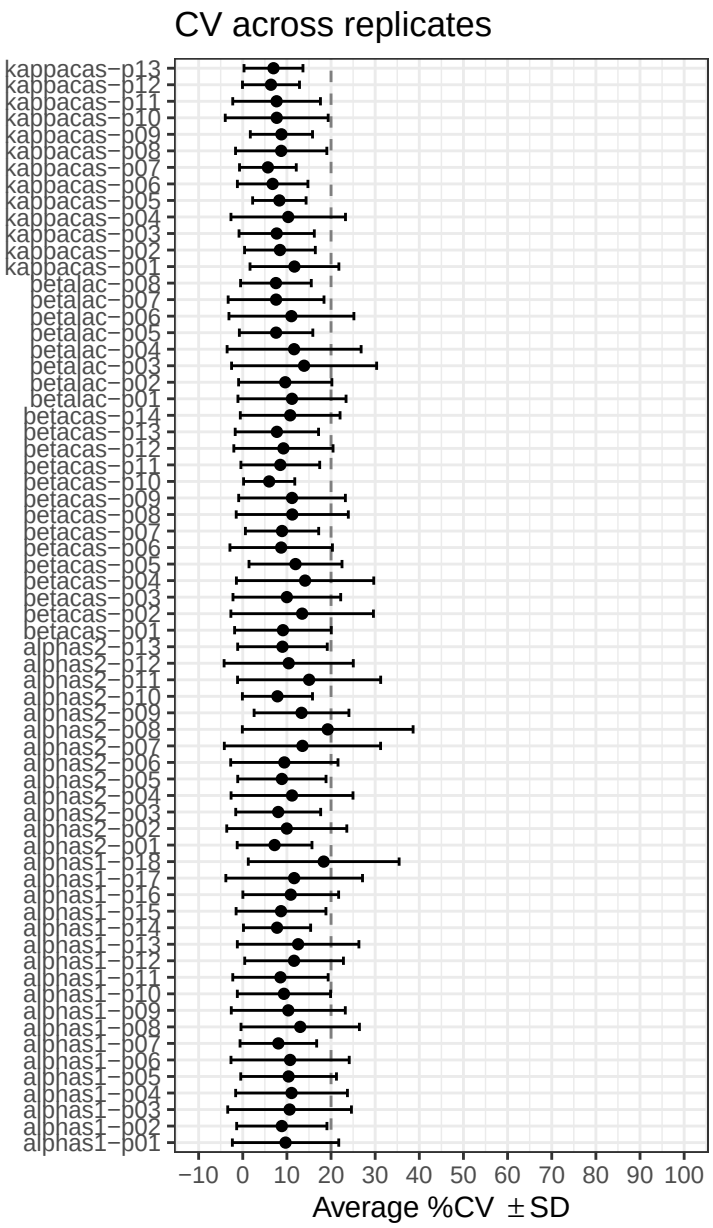
Coefficient of Variation (CV) is used to estimate varibaility of the replicates. Generally, for biological assays the desired CV is below 20%.

```
cv.db <- getCVbyPeptide.MFI(bbea, UR = c("SUBJECT.ID","Visit")) # what is the unit of replication? in t
```

```
head(cv.db)
```

```
##       Peptide   mean.cv    sd.cv    se.cv median.cv
## 1 alphas1-p01  9.694102 12.05228 1.236538  5.555556
## 2 alphas1-p02  8.842593 10.22483 1.049046  5.808490
## 3 alphas1-p03 10.605762 14.01331 1.437736  5.412659
## 4 alphas1-p04 11.050888 12.66247 1.299142  5.972589
## 5 alphas1-p05 10.385327 10.84504 1.112677  7.530656
## 6 alphas1-p06 10.724139 13.39413 1.374209  5.555556
```

```
ggplot(cv.db, aes(x = Peptide,y = mean.cv)) +
  geom_hline(yintercept = 20, color = "grey50", linetype = 2) +
  geom_point() +
  geom_errorbar(aes(ymax = mean.cv + sd.cv, ymin = mean.cv - sd.cv),width = 0.5) +
  scale_y_continuous(limits = c(-10,100),breaks = seq(-10,100,10)) +
  labs(x = "", y = expression("Average %CV "%+-%"SD"), title = "CV across replicates") +
  theme_bw() + coord_flip()
```



Averaging Technical Replicates

If there are no samples to exclude based on ICC, CV and QC of counts, technical triplicates can be averaged for the downstream analyses.

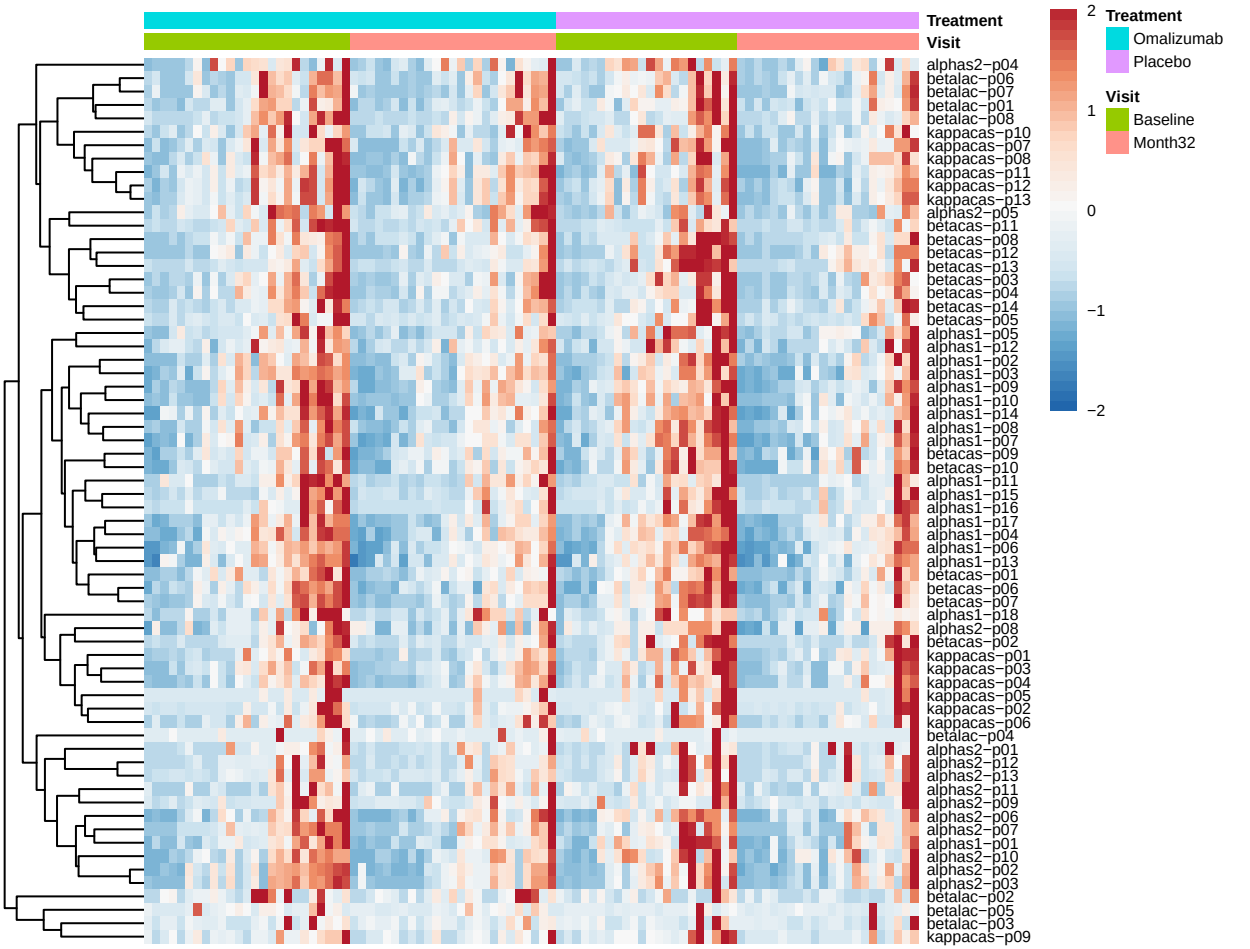
```
eset.avg <- getAveragesByReps(eset, UR = c('SUBJECT.ID','Visit'))
eset.avg
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 66 features, 94 samples
##   element names: exprs
## protocolData: none
## phenoData
##   rowNames: MILK01_Baseline MILK01_Month32 ... MILK77_Month32 (94
##     total)
##   varLabels: SUBJECT.ID Visit ... uf (23 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

```
exprs(eset.avg)[1:5,1:3]
```

```
##             MILK01_Baseline MILK01_Month32 MILK02_Baseline
## alphas1-p06        6.747561       4.447867        6.317169
## alphas1-p08        5.685712       2.983762        4.935668
## alphas1-p09        3.559142       1.955416        8.414214
## alphas1-p13        3.379424       1.427095        8.261503
## alphas1-p17        2.839442       1.302297        7.474694
```

```
# will calculate an average of 66 epitopes for each sample
eset.avg$esIgE.avg <- colMeans(exprs(eset.avg))

ord <- with(pData(eset.avg),order(Treatment, Visit,esIgE.avg))
pheatmap(exprs(eset.avg)[,ord], scale = "row", fontsize = 6,
         annotation_col = subset(pData(eset.avg)[ord,], select = c(Visit,Treatment)),
         cluster_cols = FALSE, breaks = seq(-2,2,0.1),
         color=rev(colorRampPalette(brewer.pal(7,"RdBu"))(length(seq(-2,2,0.1)))),
         show_colnames = FALSE)
```



The heatmap shows that maybe epitope-specific IgE levels somewhat decreased post treatment. To test this statistically, we will do some modelling.

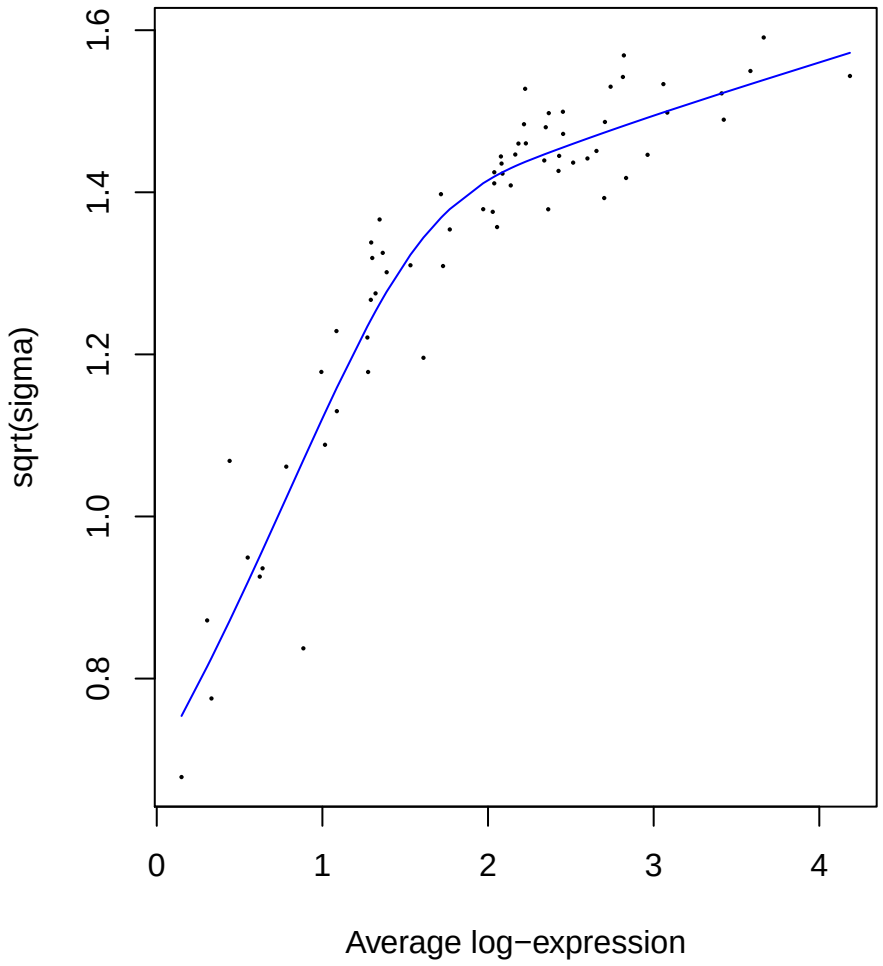
Differential Analysis - Limma Modeling

We are interested whether epitope-specific IgE antibodies decreased after patients were treated with an adjuvant Omalizumab or Placebo in addition to milk OIT.

For this, we will use a limma framework where a linear model is fit to every peptide.

```
design <- model.matrix(~ Visit*Treatment, data=pData(eset.avg))
colnames(design) <- make.names(colnames(design))
cor <- duplicateCorrelation(exprs(eset.avg), design, block=eset.avg$SUBJECT.ID)
fit <- lmFit(exprs(eset.avg), design, block=eset.avg$SUBJECT.ID, correlation=cor$consensus)
contrm <- makeContrasts(Placebo.Baseline = X.Intercept. + TreatmentPlacebo,
                        Placebo.Post = X.Intercept. + TreatmentPlacebo + VisitMonth32 + VisitMonth32.Tr
                      Omalizumab.Baseline = X.Intercept.,
                      Omalizumab.Post = X.Intercept.+VisitMonth32,
                      levels=design)
contrd <- mutate(as.data.frame(contrm),
                 PostvsBaseline.Placebo = Placebo.Post - Placebo.Baseline,
                 PostvsBaseline.Omalizumab = Omalizumab.Post - Omalizumab.Baseline,
                 TreatmentEffect = PostvsBaseline.Omalizumab - PostvsBaseline.Placebo)
```

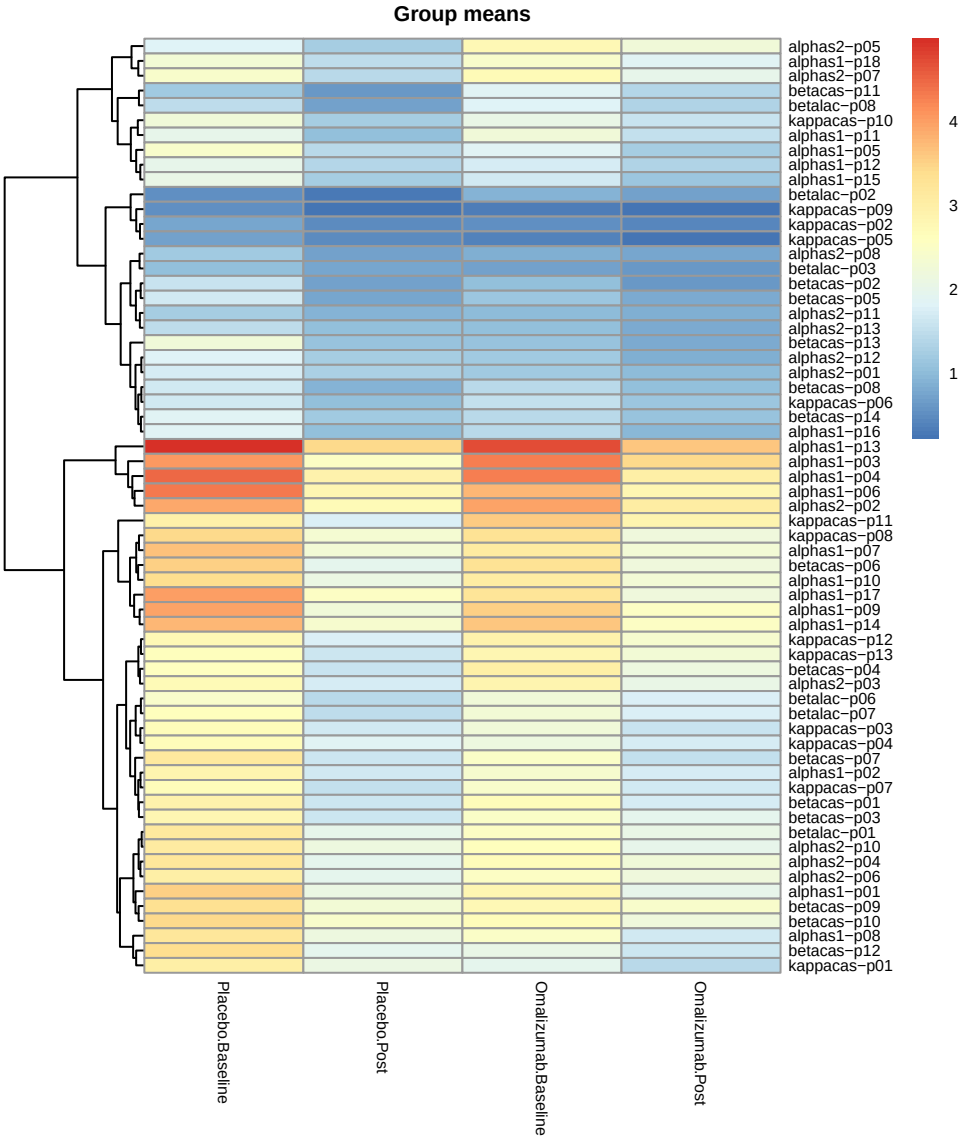
```
contrd <- contrd[,!colnames(contrd)%in%colnames(contrm)]

fitm <- eBayes(contrasts.fit(fit,contrm)) # marginal group means
ebfit <- eBayes(contrasts.fit(fit,contrd),trend=T) # lgFCH of changes
plotSA(ebfit)
```




```
D <- decideTests(ebfit,method="separate",adjust.method="BH",p.value=0.05,lfc=log2(1))
t(summary(D)) # number of significant epitopes
```

```
##                            Down NotSig Up
## PostvsBaseline.Placebo       63      3  0
## PostvsBaseline.Omalizumab    45     21  0
## TreatmentEffect               0     66  0
```

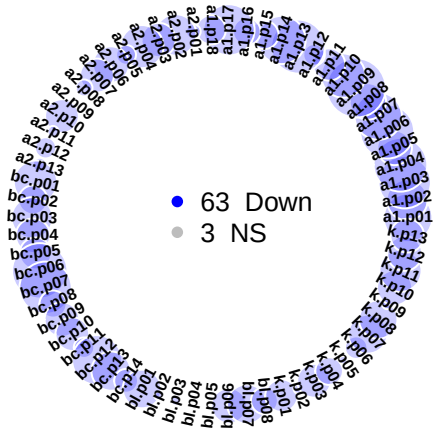
```
# significant epitopes (FDR<0.05)
sepits <- rownames(D)[which(D[,"PostvsBaseline.Placebo"]!=0)]
pheatmap(fitm$coefficients[sepits,], cluster_cols = F,
         main = "Group means", fontsize = 6)
```



We can see that epitope-specific IgE decreased in both Omalizumab and Placebo groups, with more changes (in 63 epitope-specific IgE) in the Placebo group. User needs to make sure that the Annotation file has a column named "lableName", as it will be used to print names of the epitopes.

```
Run_doNetCirclePlot(ebfit[,1], D[,1], Annot, fname = "lmFit.Placebo.")
```

lgFCH @ PostvsBaseline.Placebo




```
Run_doNetCirclePlot(ebfit[,2], D[,2], Annot, fname = "lmFit.Omalizumab.")
```

lgFCH @ PostvsBaseline.Omalizumab