bbeaR - Egg Allergy Example

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

05/23/2020

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 egg allergic children. The levels of IgE to 58 peptides (15-mer, 12 amino acid overlap), covering the entire sequence of hen's egg-white ovomucoid protein, were measured using *BBEA* in 38 allergic children and 6 controls.

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 create a topology plot, to highlight immunodominant regions on the ovomucoid protein. At the end, we will demonstrate an approach to identify epitope-specific (es)IgE that are different between allergic children and controls, using limma modeling framework.

Installation

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

```
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

44 patient samples were assayed in duplicates, ran within the same batch on one 96-well plate. The run additionally inlcuded two positive control pools (aka "PP", a mix of plasma from several allergic patients), a negative pool (aka "NP", a mix of plasma from several healthy patients), and 2 wells without any sample (aka "Buffer", for a 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 a *.csv* file of the one plate.
Note: a separate tutorial (Milk Allergy Example) has an example of importing and processing multiple plates.

```
bbeaEgg <- bbea.read.csv(fname = "BBEA_OVM_58p_IgE_patients.csv")
```

```
## [1] "Reading file: BBEA_OVM_58p_IgE_patients.csv"
```

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

```
names(bbeaEgg)
```

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

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.

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

```
##            OVM_58p_IgE_patients_1(1,A1) OVM_58p_IgE_patients_2(1,A2)
## Analyte 7                           1.0                            0
## Analyte 8                           1.0                            1
## Analyte 10                          0.5                            0
## Analyte 12                          1.0                            1
## Analyte 15                          1.0                            0
```

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

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

```
##            OVM_58p_IgE_patients_1(1,A1) OVM_58p_IgE_patients_2(1,A2)
## Analyte 7                           80                            91
## Analyte 8                           75                            84
## Analyte 10                         132                           114
## Analyte 12                         129                            87
## Analyte 15                         102                            81
```

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

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

```
##                          V1                  V2
## 1                   Program              xPONENT
## 2                     Build            3.1.971.0
## 3                      Date            6/25/2019
## 4
## 5                        SN        LX10014268401
## 6                     Batch OVM_58p_IgE_patients
## 7                   Version                    1
## 8                  Operator
## 9              ComputerName         XPONENT31-PC
```

```
## 10              Country Code                   409
## 11              ProtocolName OVM_58p_c70_6.25.2019
## 12           ProtocolVersion                     1
## 13       ProtocolDescription
## 14 ProtocolDevelopingCompany
## 15              SampleVolume                100 uL
```

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

```
colnames(bbeaEgg$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"
```

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

```
##                              Location  Sample
## OVM_58p_IgE_patients_1(1,A1)  1(1,A1) 1901-r1
## OVM_58p_IgE_patients_2(1,A2)  2(1,A2) 1901-r2
```

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

```
data(Egg)
```

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

```
dim(AnnotEgg)
```

```
## [1] 58  5
```

```
head(AnnotEgg)
```

```
##            Analyte Peptide Protein PeptideName lableName
## OVM-001 Analyte 38 OVM-001     OVM         001       001
## OVM-002 Analyte 40 OVM-002     OVM         002       002
## OVM-003 Analyte 41 OVM-003     OVM         003       003
## OVM-004 Analyte 42 OVM-004     OVM         004       004
## OVM-005 Analyte 44 OVM-005     OVM         005       005
## OVM-006 Analyte 71 OVM-006     OVM         006       006
```

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

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

```
##         OVM_58p_IgE_patients_1(1,A1) OVM_58p_IgE_patients_2(1,A2)
## OVM-009                          1.0                            0
## OVM-031                          1.0                            1
## OVM-011                          0.5                            0
## OVM-014                          1.0                            1
## OVM-020                          1.0                            0
```

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

```
dim(PDegg)
```

```
## [1] 44  4
```

```
head(PDegg)
```

```
##   PTID Age Egg.sIgE          Group
## 1 1901  NA       NA Atopic Control
## 2 1002  16       NA Atopic Control
## 3 1503   2     2.29    Egg Allergy
## 4 1774   7    16.80    Egg Allergy
## 5 1005   7    76.70    Egg Allergy
## 6 1606   4    35.20    Egg Allergy
```

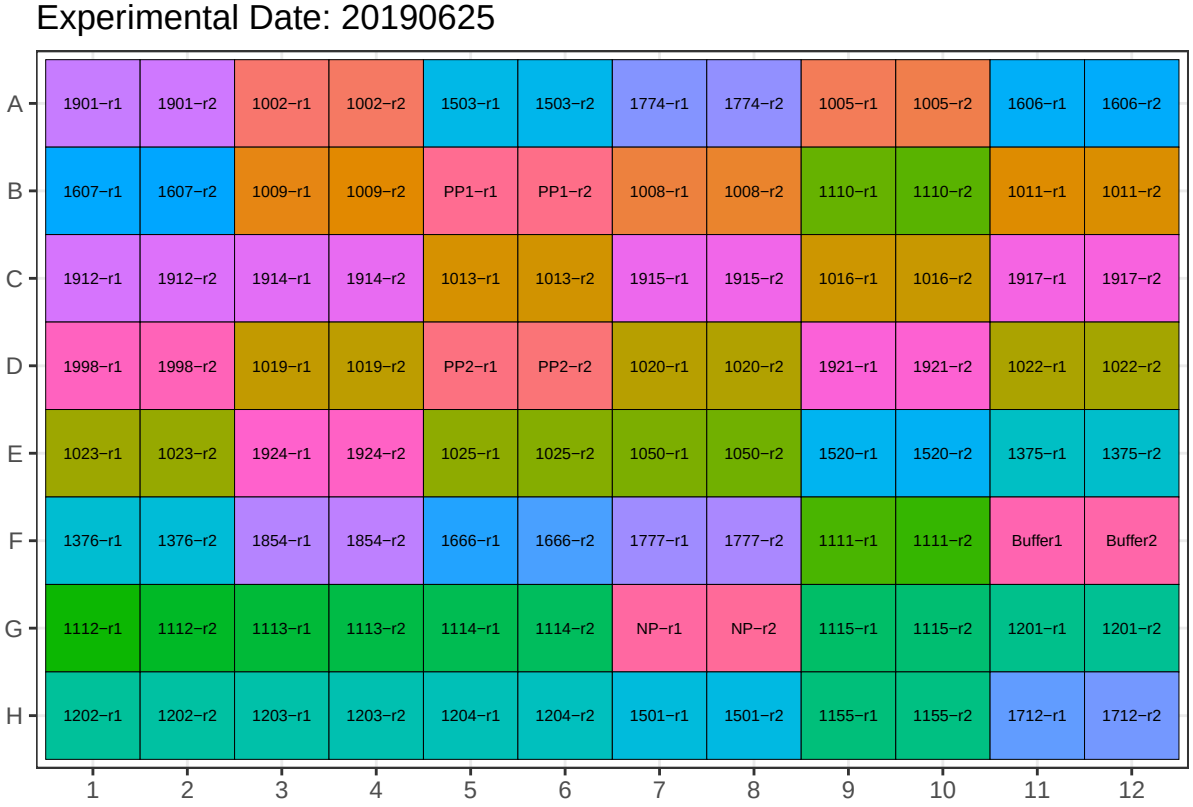
We will clean up the pData and then merge it with PDegg. This step is not necessary, but will be useful when we do statistical modelling.

```
bbeaEgg$pData <- mutate(bbeaEgg$pData,
                        PTID = sapply(strsplit(Sample, "\\-"), head, 1),
                        SampleType = ifelse(grepl("Buff", Sample),"Buffer",
                                            ifelse(grepl("PP|NP", Sample),
                                                   sapply(strsplit(Sample, "\\-"),head, 1), "Patient")))
PDm <- merge(bbeaEgg$pData, PDegg, by = "PTID", all.x = T)
rownames(PDm) <- PDm$File
bbeaEgg$pData <- PDm[colnames(bbeaEgg$Median),] # making sure samples are arranged correctly
```

Quality Control of the Raw Data

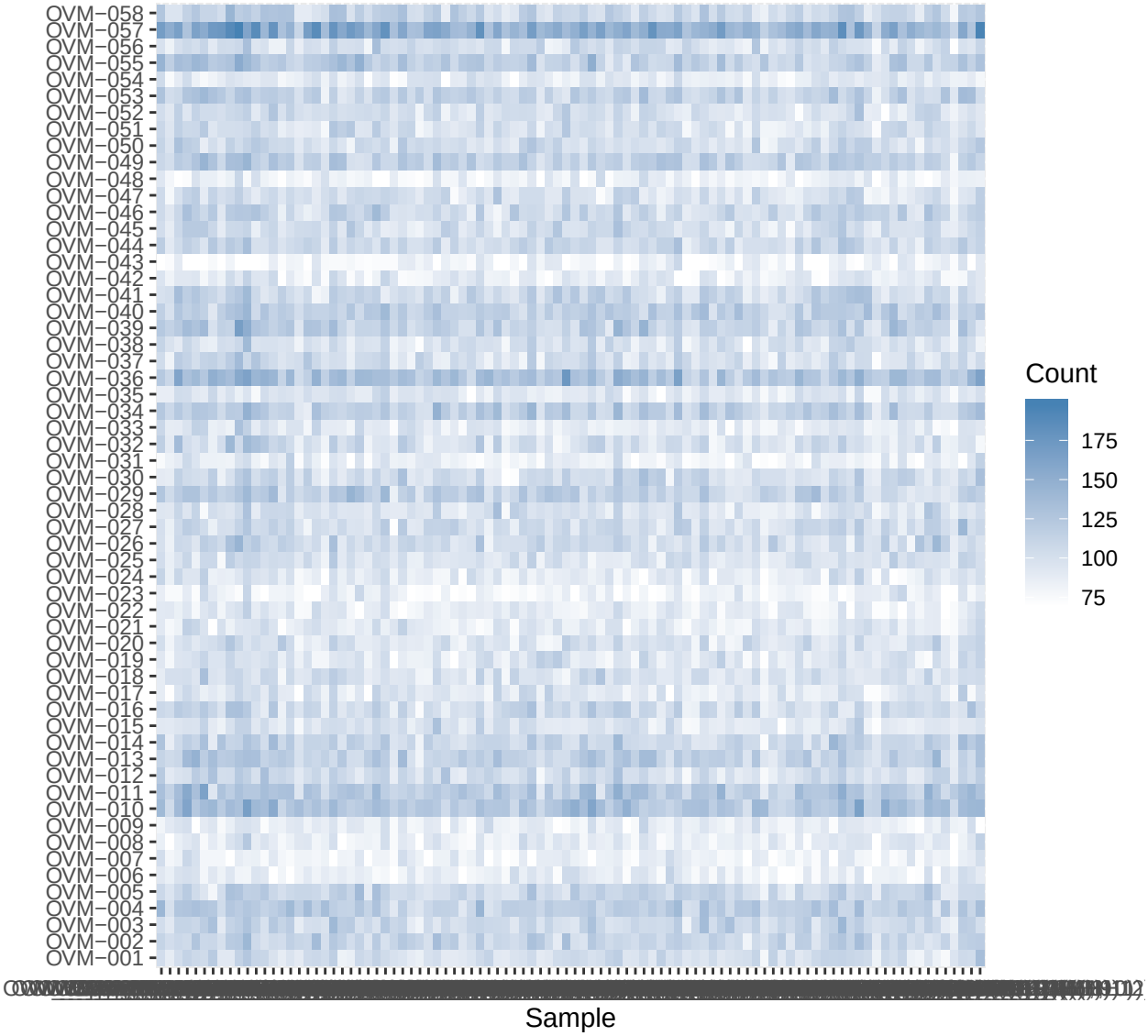
Layout of the experimental plate.
Tip: to make sure no position bias is present in the data, samples have to be randomzied among wells and plates, as outlined in this publication.

```
bbea.QC.Image(bbeaEgg$pData,
              filename = "QC.Image.",
              plate.design.db = plate.design.db)
```



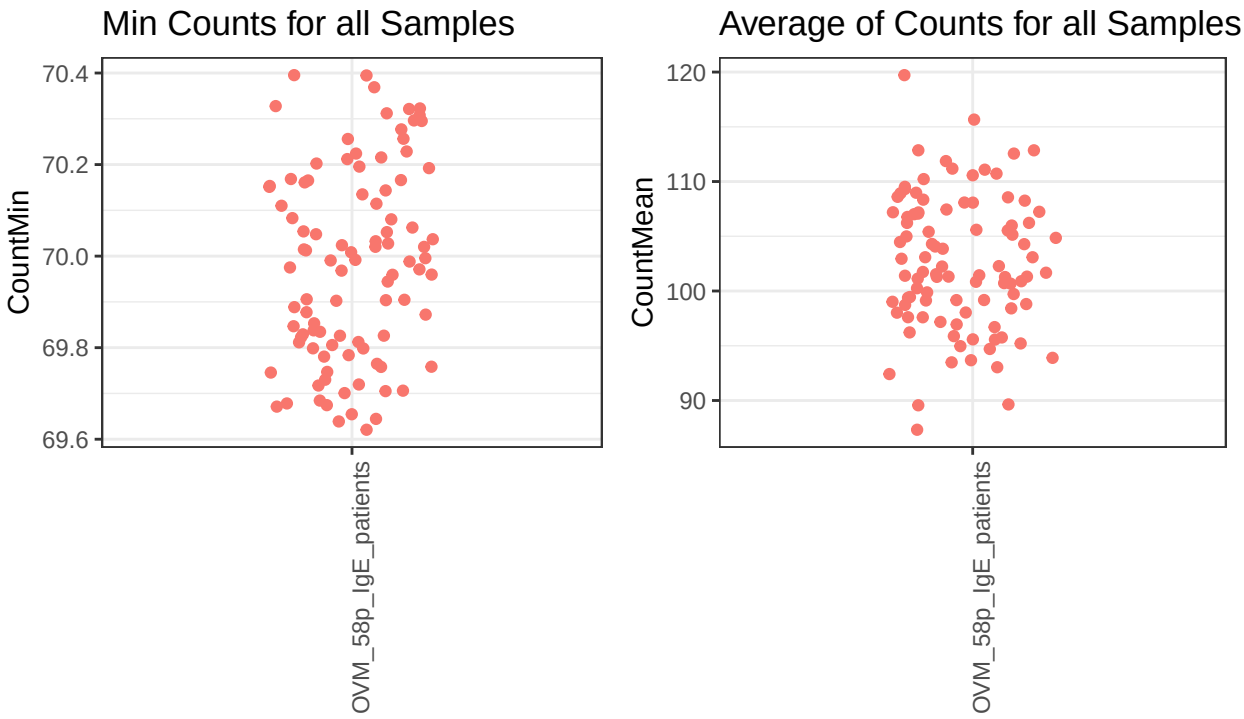
We want to make sure that there are no missing samples or analytes. This would be reflcected by the very low counts (<25).
The heatmap shows that all samples and analytes are were included.

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



Now we look at the overall distribution of counts: the minimum count is ~69, which is pretty good.

```
p<-bbea.QC.Samples(bbeaEgg,
                   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(bbeaEgg, statement = (bbeaEgg$pData$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.

```
bbeaEgg$pData$Sample # samples 71 & 72 are the background wells
```

```
##  [1] "1901-r1" "1901-r2" "1002-r1" "1002-r2" "1503-r1" "1503-r2" "1774-r1"
##  [8] "1774-r2" "1005-r1" "1005-r2" "1606-r1" "1606-r2" "1607-r1" "1607-r2"
## [15] "1009-r1" "1009-r2" "PP1-r1"  "PP1-r2"  "1008-r1" "1008-r2" "1110-r1"
## [22] "1110-r2" "1011-r1" "1011-r2" "1912-r1" "1912-r2" "1914-r1" "1914-r2"
## [29] "1013-r1" "1013-r2" "1915-r1" "1915-r2" "1016-r1" "1016-r2" "1917-r1"
## [36] "1917-r2" "1998-r1" "1998-r2" "1019-r1" "1019-r2" "PP2-r1"  "PP2-r2"
## [43] "1020-r1" "1020-r2" "1921-r1" "1921-r2" "1022-r1" "1022-r2" "1023-r1"
## [50] "1023-r2" "1924-r1" "1924-r2" "1025-r1" "1025-r2" "1050-r1" "1050-r2"
## [57] "1520-r1" "1520-r2" "1375-r1" "1375-r2" "1376-r1" "1376-r2" "1854-r1"
## [64] "1854-r2" "1666-r1" "1666-r2" "1777-r1" "1777-r2" "1111-r1" "1111-r2"
## [71] "Buffer1" "Buffer2" "1112-r1" "1112-r2" "1113-r1" "1113-r2" "1114-r1"
## [78] "1114-r2" "NP-r1"   "NP-r2"   "1115-r1" "1115-r2" "1201-r1" "1201-r2"
## [85] "1202-r1" "1202-r2" "1203-r1" "1203-r2" "1204-r1" "1204-r2" "1501-r1"
## [92] "1501-r2" "1155-r1" "1155-r2" "1712-r1" "1712-r2"
```

```
bbeaN <- MFI2nMFI(bbeaEgg,
                  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] "OVM_58p_IgE_patients"
```

```
names(bbeaN)
```

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

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

```
##         OVM_58p_IgE_patients_1(1,A1) OVM_58p_IgE_patients_2(1,A2)
## OVM-009                    0.7924813                     0.000000
## OVM-031                    1.5849625                     1.584963
## OVM-011                    0.0000000                     0.000000
## OVM-014                    1.5849625                     1.584963
## OVM-020                    0.0000000                     0.000000
```

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: 58 features, 94 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: OVM_58p_IgE_patients_1(1,A1)
##     OVM_58p_IgE_patients_2(1,A2) ... OVM_58p_IgE_patients_96(1,H12) (94
##     total)
##   varLabels: PTID Location ... Group (23 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

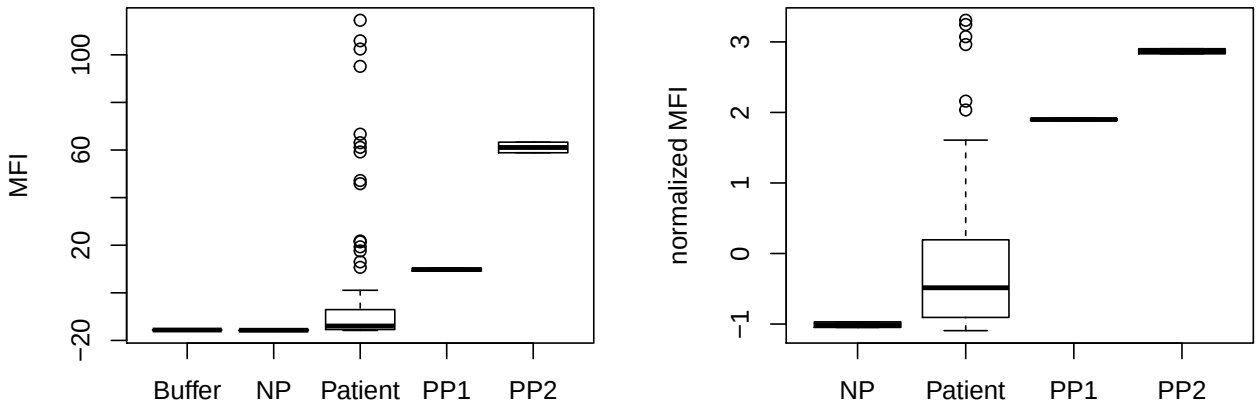
QC of Normalized Data

Distributions

Overall distribution:
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.

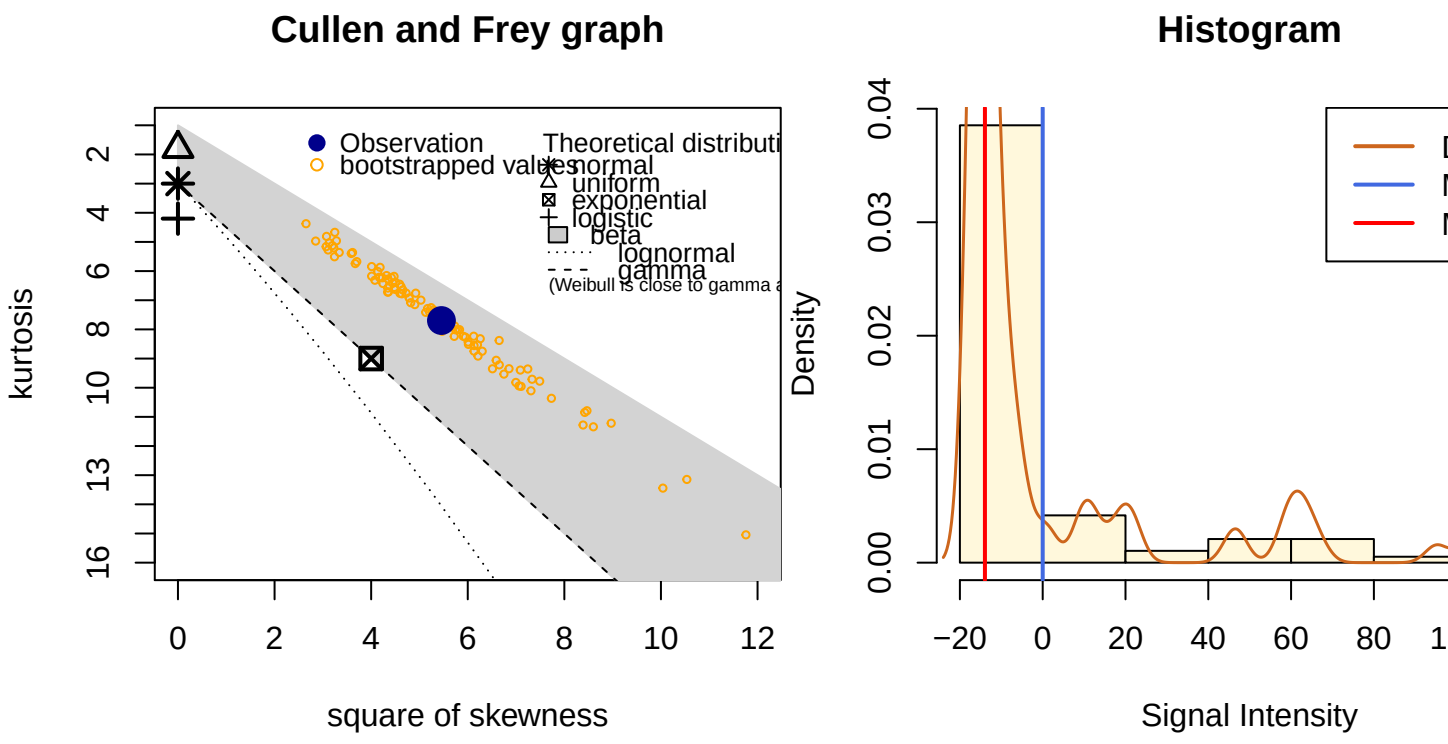
```
adjMFI <- t(apply(as.matrix(bbeaEgg$Median), 1, function(x){x - mean(x, na.rm=T)}))
adjnMFI<-t(apply(as.matrix(exprs(eset)), 1, function(x){x - mean(x, na.rm=T)}))

par(mfrow = c(1, 2))
boxplot(colMeans(adjMFI) ~ bbeaEgg$pData$SampleType,
        xlab = " ", ylab = "MFI")
boxplot(colMeans(adjnMFI) ~ eset$SampleType,
        xlab = " ", ylab = "normalized MFI")
```

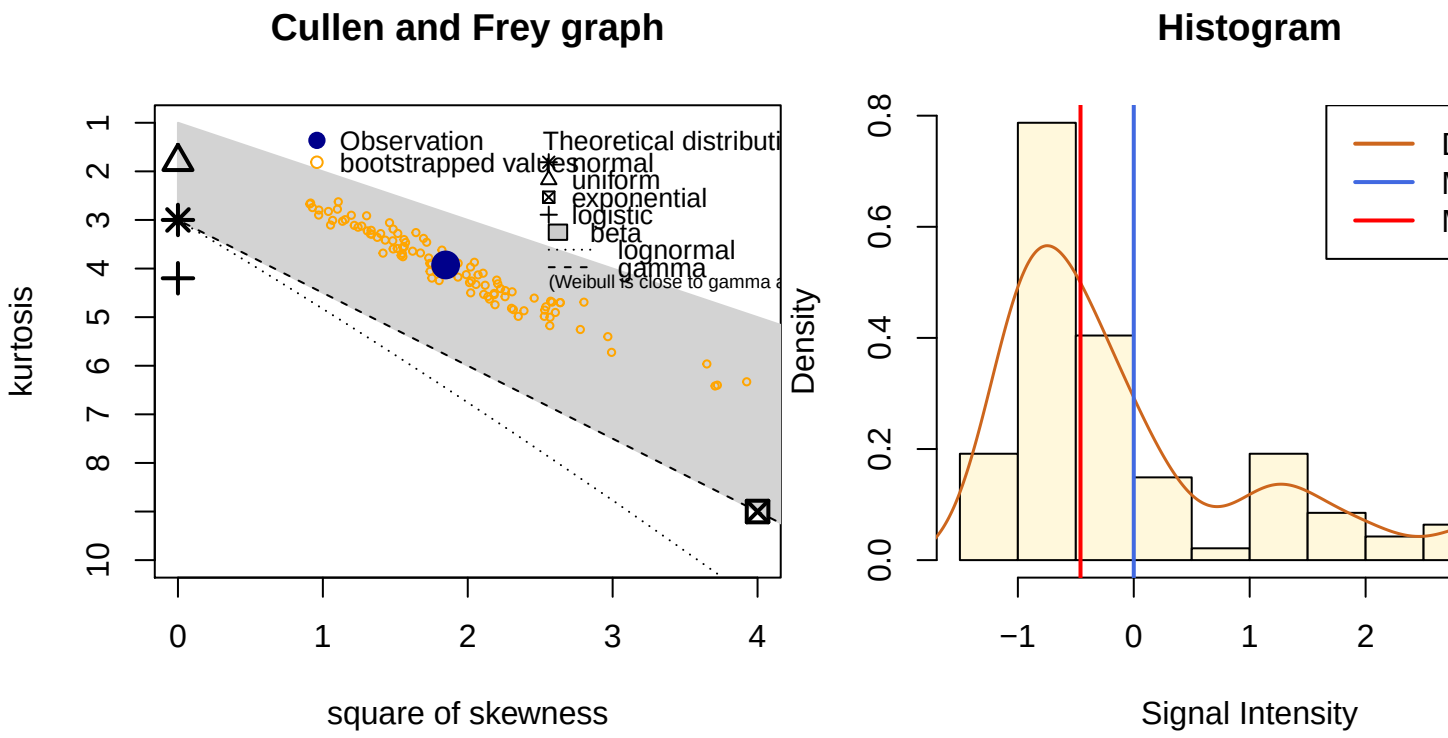


Cullen-Frey plots can be used to evaluate the distribution of the data. It shows how the skewness and kurtosis of our data compare to the theoretical distributions.
CullenFreyPlot() function is a wrapper of the fitdistrplus::descdist().

```
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

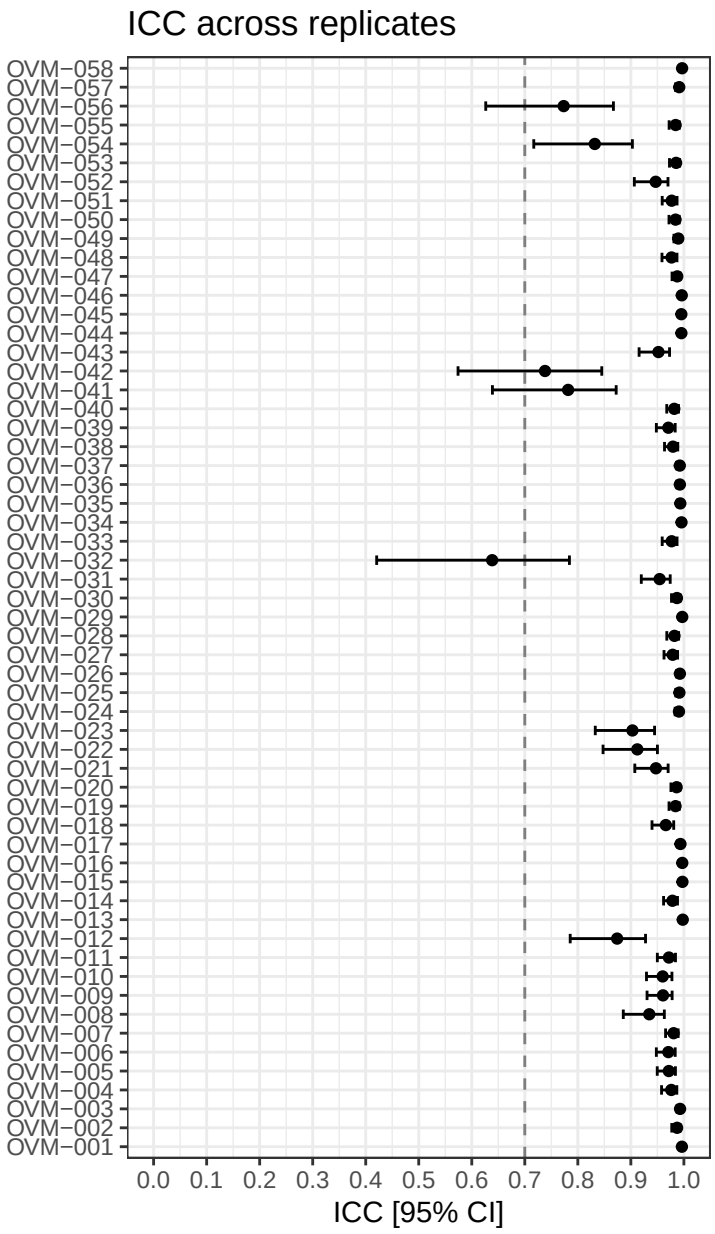
Intraclass Correlation Coefficient (ICC) is used to evaluate agreement among technical replicates for each peptide, where 0 means no agreement, and 1 is a perfect agreement.
The function returns the ICC and a 95% confidence interval.

```
icc.db <- getICCbyPeptide(eset,
                          UR=c("PTID","Plate")) # what is the unit of replication? in this case, it is
head(icc.db)
```

```
##         Peptide       ICC       LCI       UCI
## OVM-009 OVM-009 0.9606181 0.9305577 0.9778404
## OVM-031 OVM-031 0.9542997 0.9197155 0.9742303
## OVM-011 OVM-011 0.9718032 0.9499818 0.9841880
## OVM-014 OVM-014 0.9784793 0.9617347 0.9879478
## OVM-020 OVM-020 0.9862967 0.9756282 0.9923270
## OVM-023 OVM-023 0.9030485 0.8328782 0.9447205
```

```
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%.

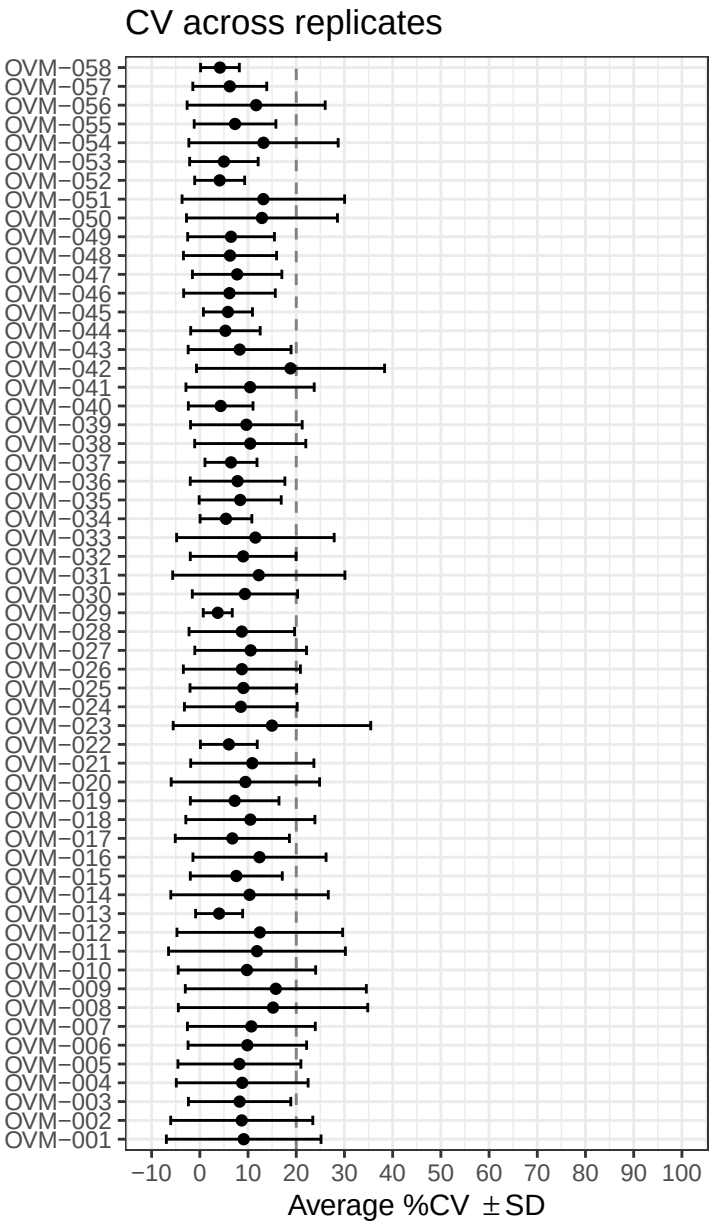
```
cv.db <- getCVbyPeptide.MFI(bbeaEgg,
                            UR = c("PTID","Plate")) # what is the unit of replication? in this case, it
```

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

```
##   Peptide  mean.cv    sd.cv    se.cv median.cv
## 1 OVM-001 9.102478 16.04143 2.291632  1.170530
```

```
## 2 OVM-002 8.692819 14.74045 2.105779  0.000000
## 3 OVM-003 8.253370 10.61832 1.516903  4.876598
## 4 OVM-004 8.788103 13.67184 1.953120  0.000000
## 5 OVM-005 8.209540 12.74839 1.821199  0.000000
## 6 OVM-006 9.849911 12.29611 1.756588  3.142697
```

```
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 replicates can be averaged for the downstream analyses.

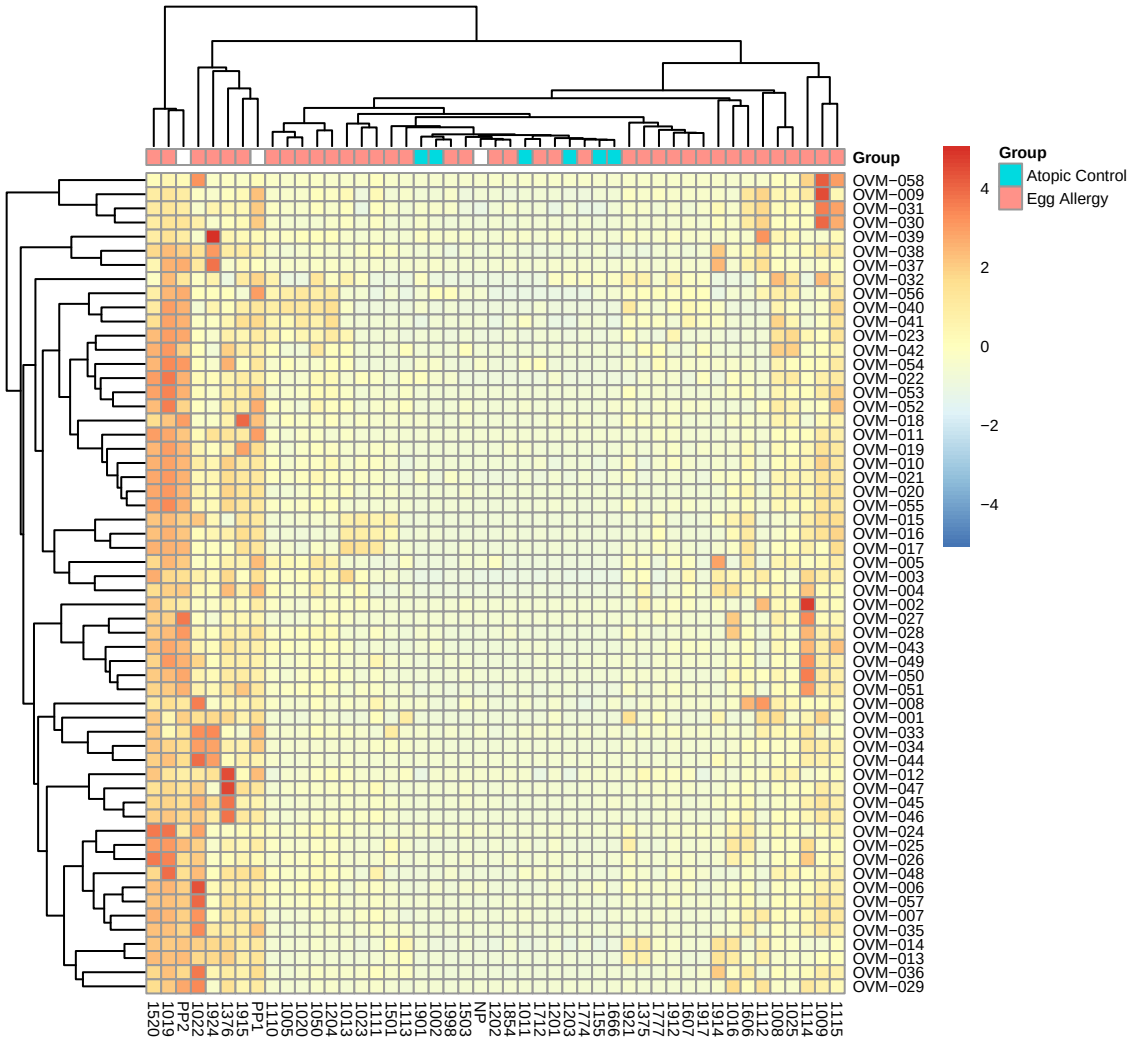
```
eset.avg <- getAveragesByReps(eset, UR = 'PTID')
eset.avg
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 58 features, 47 samples
##   element names: exprs
## protocolData: none
## phenoData
##   rowNames: 1002 1005 ... PP2 (47 total)
##   varLabels: PTID Location ... uf (24 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

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

```
##              1002      1005      1008
## OVM-009 0.3962406 0.7924813 2.1961587
## OVM-031 1.5849625 1.5849625 2.8073549
## OVM-011 0.0000000 0.0000000 0.7369656
## OVM-014 0.7924813 1.5849625 2.8073549
## OVM-020 0.0000000 0.0000000 1.4036775
```

```
pheatmap(exprs(eset.avg), scale = "row", fontsize = 6,
         annotation_col = subset(pData(eset.avg), select = Group))
```

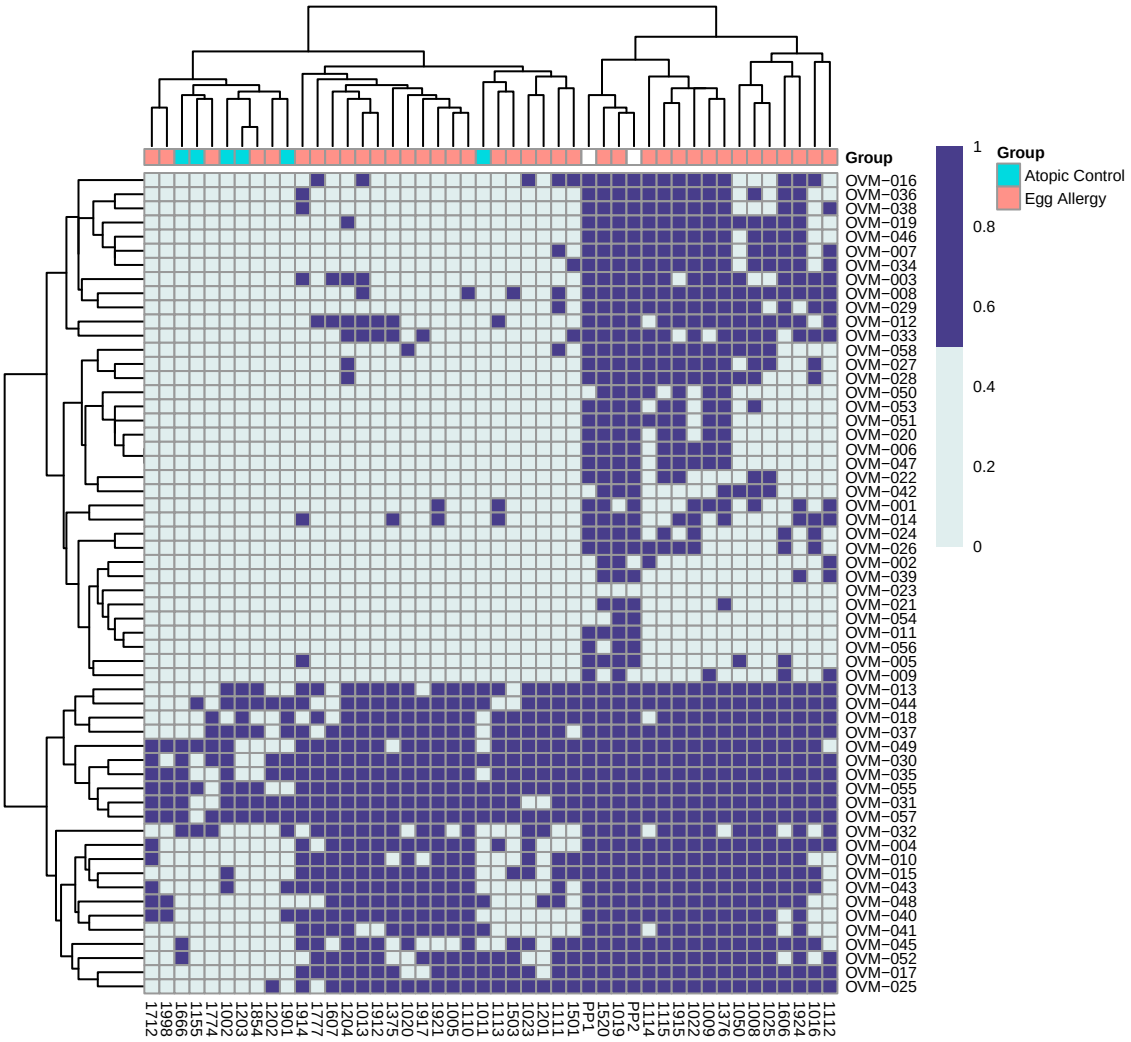


The heatmap shows that few of the patients have higher IgE levels to several ovomucoid peptides.

Another useful statistic of IgE levels is a binary value of whether the IgE an epitope is "present". It can be defined as "present" if the MFI 2-3 standard deviations above the background (aka "Buffer" wells). Sometimes it might be more important to compare the MFI to negative control samples rather than the buffer. This can be easily achieved by specifying a desired string (e.g., "NegativeControl" or "NP") in the *buffer.name* argument of *binarizeMFIbySD()*.

```
binary.db <- binarizeMFIbySD(bbea.object = bbeaEgg,
                             buffer.name = "NP", # a string in the "Sample" column identifying referenc
                             plateVar = "Plate", # column indicating batch/plate
                             UR = 'PTID', # unique sample ID
                             nSD = 3) # numner of standard deviaitons above the buffer
sampleNames(binary.db) <- sapply(strsplit(sampleNames(binary.db), "_"),tail, 1)
# can remove background (Buffer) wells
binary.db <- binary.db[, !grepl("Buffer", sampleNames(binary.db))]

pheatmap(exprs(binary.db), fontsize = 6,
         annotation_col = subset(pData(eset.avg)[sampleNames(binary.db),], select = Group),
         color = c("azure2", "darkslateblue"))
```



The blue color reprsents peptide-specific IgE antibodies detected in our samples.

Differential Analysis - Limma Modeling

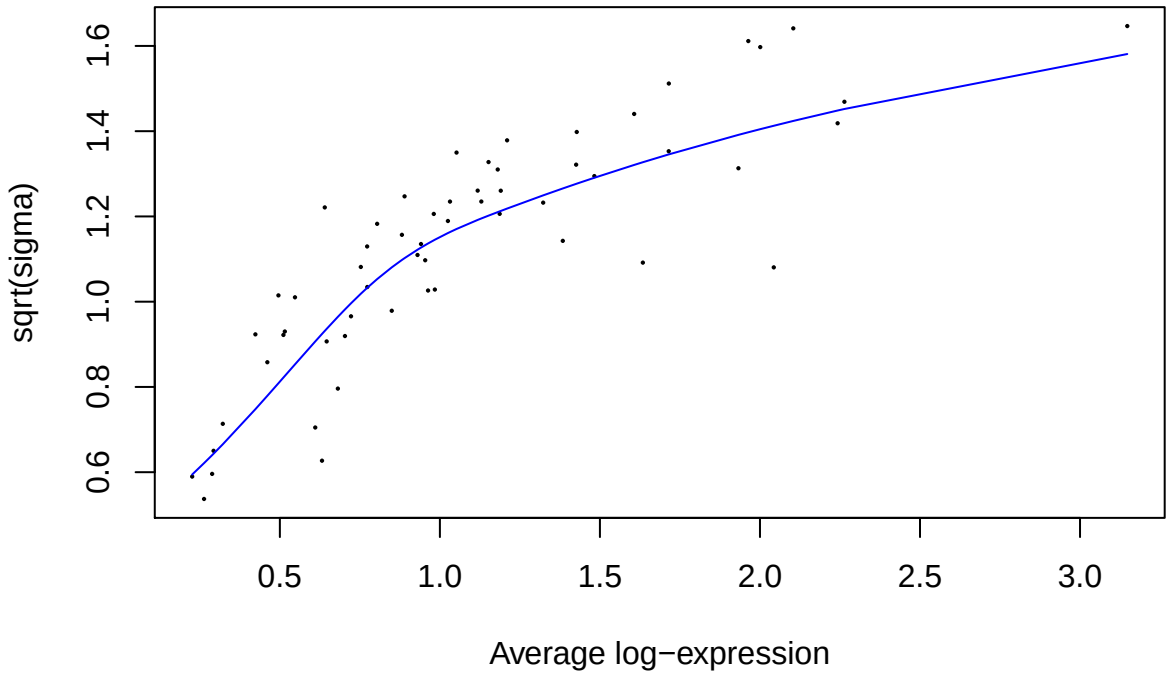
We are interested in identifying peptide-specific IgE antibodies that are different between egg allergic patients and atopic controls. Atopic controls are patients that are not allergic to egg but have other types of allergies.

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

```
# removing control samples
eseti <- eset.avg[, which(!grepl("NP|PP", eset.avg$PTID))]

design <- model.matrix(~ Group, data = pData(eseti))
colnames(design) <- make.names(colnames(design))
fit <- lmFit(eseti, design)
# means
contr_m <- makeContrasts(Controls     = X.Intercept.,
                         EggAllergic = X.Intercept.  + GroupEgg.Allergy, levels = design)
# differences
contr_d <- mutate(as.data.frame(contr_m), Allergic_vs_Controls = EggAllergic - Controls)
```

```
contr_d <- subset(contr_d, select=Allergic_vs_Controls)
fitm <- eBayes(contrasts.fit(fit,contr_m)) # marginal group means
ebfit <- eBayes(contrasts.fit(fit,contr_d), trend=TRUE) # lgFCH allergic vs controls
par(mfrow = c(1, 1))
plotSA(ebfit) # this plot shows that we have a variance trend, so we should keep trend=TRUE in the eBay
```




```
D <- decideTests(ebfit, method = "separate", adjust.method = "none",
                 p.value = 0.05, lfc = log2(1))
t(summary(D)) # number of significant peptides
```

```
##                       Down NotSig Up
## Allergic_vs_Controls    0     51  7
```

```
# Table of top peptide-specific IgE antibodies
topTable(ebfit, number=10)
```

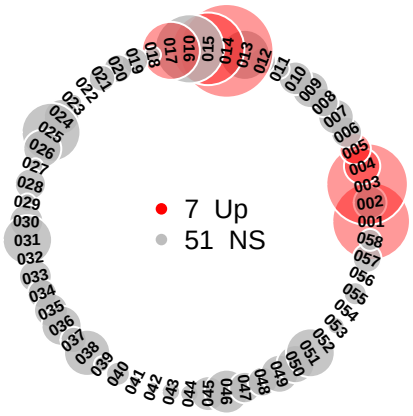
```
##             logFC   AveExpr        t     P.Value adj.P.Val         B
## OVM-003 2.5566883 2.2426355 2.866741 0.005948507 0.1757321 -2.416145
## OVM-014 2.9364076 3.1476426 2.503801 0.015419553 0.1757321 -3.052960
## OVM-004 1.1397038 0.9842896 2.328726 0.023742196 0.1757321 -3.337743
## OVM-005 0.9838123 0.8496561 2.230536 0.029993222 0.1757321 -3.490561
## OVM-001 2.4363516 2.1041219 2.155915 0.035675735 0.1757321 -3.603246
## OVM-015 2.2767442 2.0008656 2.119685 0.038760804 0.1757321 -3.656859
## OVM-017 1.8205973 1.6069205 2.059586 0.044393440 0.1757321 -3.744178
## OVM-016 2.1669335 1.9635633 1.989332 0.051866367 0.1757321 -3.843657
## OVM-051 1.5296773 1.9327391 1.968229 0.054312795 0.1757321 -3.872984
## OVM-040 0.7422013 0.7033638 1.940260 0.057707254 0.1757321 -3.911452
```

We have identified 7 peptide-specific IgE antibodies that are higher in the allergic group. In this exmaple, we have a small control group (n=6), so the p-values were not corrected for multiple comparison.

Significant peptide-specific IgE can be visualized using the "area circle" plot.
Area of the circle is proportional to the absolute value of the difference (Allergic vs Controls). Color of the circle indicates the direction (Higher/Up=red, Lower/Down=blue, Non-Significant (NS)=grey) and significance. 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, D, AnnotEgg, fname = "lmFit.EAvsAC.")
```

IgFCH @ Allergic_vs_Controls



Protein Topology Plot

We can now use our experimental data to overlay it on the amino acid sequence of the protein. Ovomucoid doesn't have a complete 3D crystal structure, so a "topology" plot can be helpful in identifying specific areas on a protein that are recognized by patients' IgE.

First, we need to create a protein layout. We will use *drawProteins* package to download information about the protein sequence from the UniProt database. The meta data inludes amino acid (aa) positions of SIGNAL and CHAIN sequences, as well as information about reactive sites, glycosylation, disulfate bridges.

```
json <- drawProteins::get_features("P01005")
```

```
## [1] "Download has worked"
```

```
ovm_p01005_meta <- drawProteins::feature_to_dataframe(json)
# Ovomucoid has a signal peptide sequence, that was not part of the peptides used in the assay.
ovm_signalEnd <- ovm_p01005_meta$end[which(ovm_p01005_meta$type == "SIGNAL")]
ovm_p01005_meta <- mutate(ovm_p01005_meta,
                          beginMinusSignal = begin - ovm_signalEnd,
                          endMinusSignal = end - ovm_signalEnd)
# complete aa sequence
ovm_p01005_aaseq <- json[[1]]$sequence
# removing signal peptide
ovm_p01005_aaseqCHAIN <- substr(ovm_p01005_aaseq, ovm_signalEnd + 1, (nchar(ovm_p01005_aaseq)))
```

getTopologyPlotDB() function will construct a dataframe that can be used for plotting. The *getEnzymeCuts()* function is based on the R package cleaver and can add sites of enzymatic cuts (by trypsin, pepsin, and chymotrypsin).

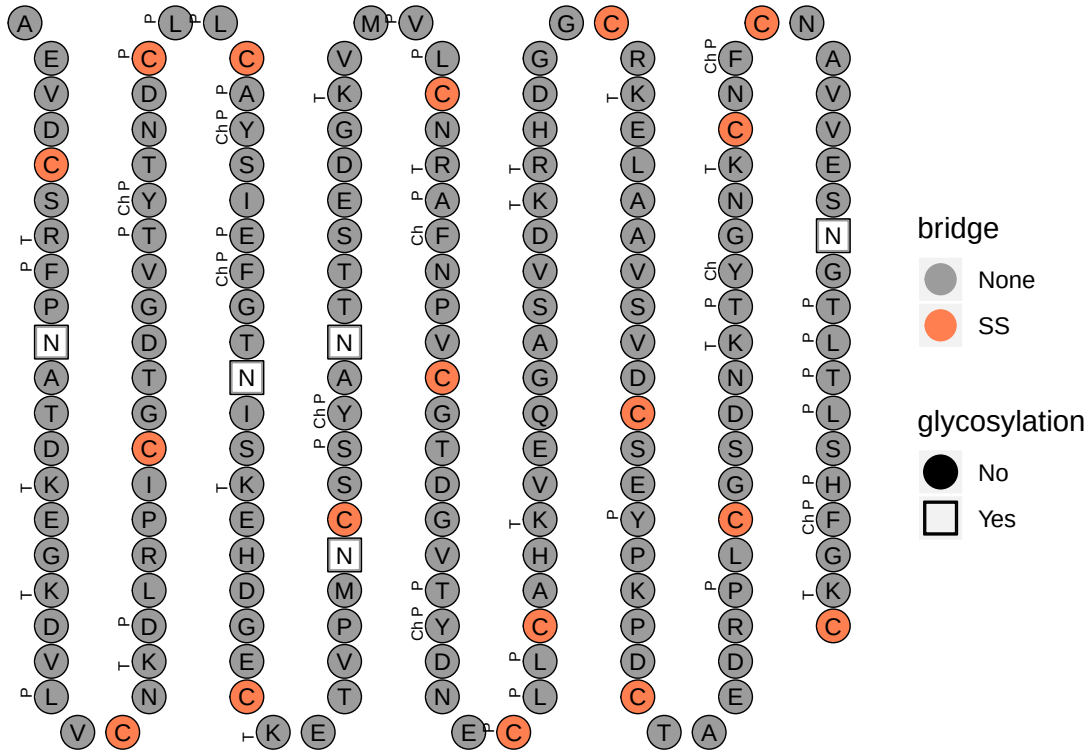
```
ovm_plotDB <- getTopologyPlotDB(ovm_p01005_aaseq, # a character string with amino acid sequence
                                ymax = 20, # number of amino acids in one column
                                offset = 3, # the first few amino acids to be below the rest of the plo
                                meta = TRUE, # do we have the metadata
                                metadb = ovm_p01005_meta) # metadata dataframe

ovm_plotDB <- getEnzymeCuts(ovm_p01005_aaseq, # a character string with amino acid sequence
                            topo = TRUE, # do we have a dataframe generated with getTopologyPlotDB()
                            topodb = ovm_plotDB, # name of the getTopologyPlotDB() dataframe
                            rmsignal = TRUE, # remove the signal peptide chain?
                            signalend = 24) # at what residue does the signal chain end

ovm_plotDB<-mutate(ovm_plotDB,
                   EnzymeRmSignal = ifelse(is.na(EnzymeRmSignal), "", EnzymeRmSignal))

makeTopologyPlotBase(subset(ovm_plotDB, type != "Signal")) +
  geom_point(size = 5.5, color='black', aes(shape = glycosylation)) +
  geom_point(size = 5, aes(color = disulf, shape = glycosylation)) +
  scale_shape_manual(values = c(19, 0, 11), labels = c("No", "Yes")) +
  scale_color_manual("bridge", values = c("grey61", "coral"), labels = c("None","SS")) +
  geom_text(aes(label = AA), size = 3) +
  labs(title = paste("Ovomucoid (P01005)")) +
  geom_text(data = subset(ovm_plotDB, EnzymeRmSignal != ""), aes(label = EnzymeRmSignal),
            size = 2, vjust = -1.7, angle = 90)
```

Ovomucoid (P01005)



The plot shows that several Asparagine (N) residues are glycosilated; the Cysteine residues colored in orange form disulfade bridges; and there are many potential enzymatic cleavage sites (P=Pepsin; T=Trypsin, Ch=Chymotrypsin).

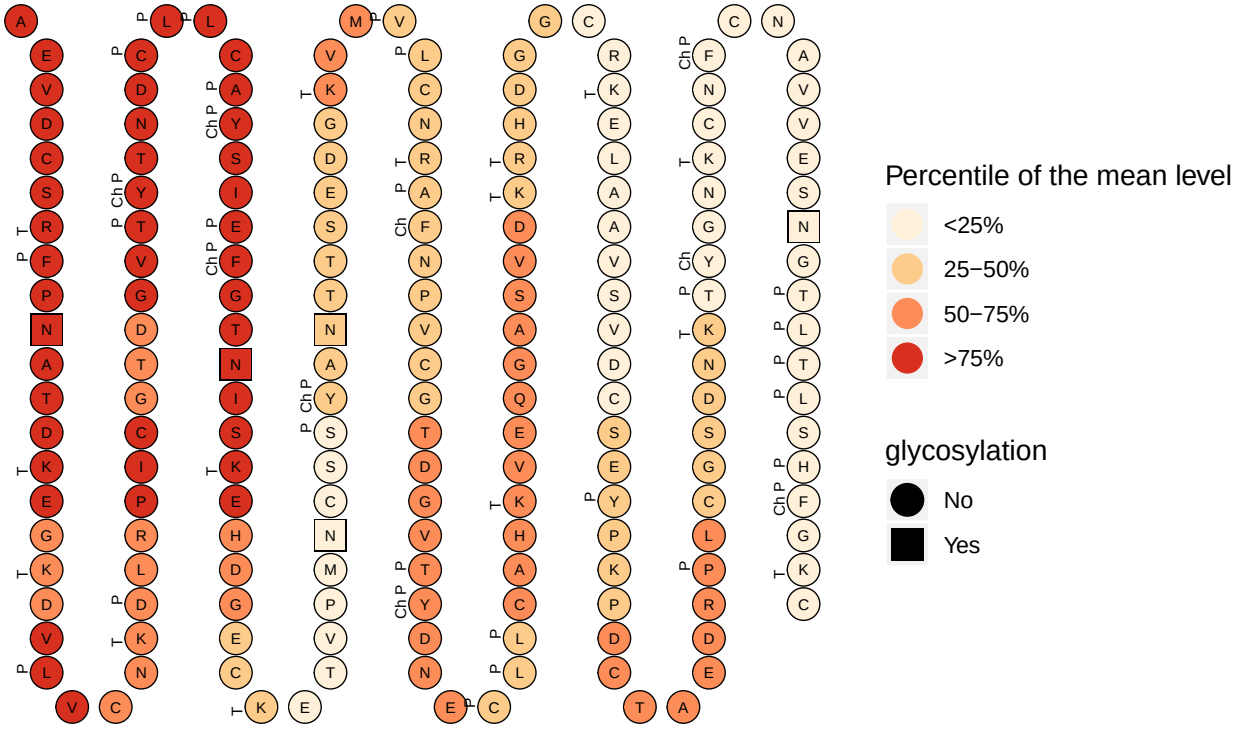
Now we will overlay this topology plot with our experimental data to see if any residues stand out.

```
# Getting sequences of the overlapping peptides (overlap by 12 aa, offset 3)
peptDB_OVM <- do.call("rbind",sapply(seq(1, nchar(ovm_p01005_aaseqCHAIN), 3),
                                     function(i){
                                       substr(ovm_p01005_aaseqCHAIN, i, 14 + i)
                                       }, simplify = FALSE))
peptDB_OVM <- subset(mutate(as.data.frame(peptDB_OVM),
                            Peptide = paste0("OVM-", str_pad(1:nrow(peptDB_OVM), 3, pad = "0")),
                            Peptide.Number = 1 : nrow(peptDB_OVM)),
                     nchar(as.character(V1)) == 15)
colnames(peptDB_OVM)[1] <- "Sequence"

# generate topology dataframe by using the means of the egg allergic patients
dbt <- getTopologynMFI(eset = eset.avg[,which(eset.avg$Group == "Egg Allergy")],
                       peptideDB = peptDB_OVM,
                       topoDB = ovm_plotDB,
                       endSignal = ovm_signalEnd)
# assigning percentiles to the average level for each peptide
dbt$avgp <- cut(dbt$avg,
                breaks = quantile(dbt$avg, probs=seq(0, 1, by = 0.25), na.rm = TRUE),
                include.lowest = TRUE, labels = c("<25%", "25-50%", "50-75%", ">75%"))

makeTopologyPlotBase(subset(dbt, type != "Signal")) +
```

```
  geom_point(size = 5.5, color = 'black',aes(shape = glycosylation)) +
  geom_point(size = 5, aes(color = avgp, shape = glycosylation)) +
  scale_color_brewer("Percentile of the mean level", palette = "OrRd") +
  scale_shape_manual(values = c(19, 15), labels = c("No", "Yes")) +
  geom_text(aes(label = AA), size = 2) +
  geom_text(data = subset(dbt,EnzymeRmSignal != ""), aes(label = EnzymeRmSignal),
            size = 2, vjust = -1.7, angle = 90) +
  labs(title = "Mean nMFI's percentile for each amino acid")
```



The topology plot shows that most IgE antibodies of our egg allergic patients were directed to the several groups of amino acids in the N-terminal of the ovomucoid protein.