RPPA analysis development

Lucy Liu

Contents

Introduction

The aim of this analysis is to guide the development of RPPA data analysis methodology.

The data set that will be used is from an experiment comparing protein expression across various siRNA transfected conditions and two drug compound conditions in the breast cancer cell line HS578T. There are two biological replicates for each condition.

The details of sample conditions are detailed below:

```r
sample_info <- read.xlsx("Data/RunSummary_SIMPSON_RoberyVary_R067_20180521.xlsx",
                         sheet = 4, colNames = TRUE, rowNames = FALSE)
kable(sample_info)
```

Sample	CellLine	Condition	TimePoint	Replicate
RJV37	HS578T	OTP	24hr	1
RJV38	HS578T	OTP	24hr	2
RJV43	HS578T	FIP1L1	24hr	1
RJV44	HS578T	FIP1L1	24hr	2
RJV45	HS578T	PCF11	24hr	1
RJV46	HS578T	PCF11	24hr	2
RJV47	HS578T	CSFT2	24hr	1
RJV48	HS578T	CSFT2	24hr	2
RJV49	HS578T	mock 20-24	24hr	1
RJV50	HS578T	mock 20-24	24hr	2
RJV53	HS578T	OTP	48hr	1
RJV54	HS578T	OTP	48hr	2
RJV59	HS578T	FIP1L1	48hr	1
RJV60	HS578T	FIP1L1	48hr	2
RJV61	HS578T	PCF11	48hr	1
RJV62	HS578T	PCF11	48hr	2
RJV63	HS578T	CSFT2	48hr	1
RJV64	HS578T	CSFT2	48hr	2
RJV65	HS578T	mock 20-48	48hr	1
RJV66	HS578T	mock 20-48	48hr	2
RJV67	HS578T	Drug 20-48	48hr	1
RJV68	HS578T	Drug 20-48	48hr	2
RJV69	HS578T	mock 44-48	48hr	1
RJV70	HS578T	mock 44-48	48hr	2
RJV71	HS578T	Drug 44-48	48hr	1
RJV72	HS578T	Drug 44-48	48hr	2

Read in the raw normalised to secondary AB RFI values and print the beginning of the data:

```r
raw <- read.xlsx("Data/RunSummary_SIMPSON_RoberyVary_R067_20180521.xlsx",
                 sheet = 2, colNames = TRUE, rowNames = TRUE)
#read in raw data
raw <- raw[-1,]
#remove the second row, which contained in-house AB names
raw[] <- sapply(raw, FUN = as.double)
#convert all rows into double
raw <- t(raw)
#transpose (swap rows and columns)
raw[1:5,1:9]
```

```
##            RJV37   RJV38  RJV43  RJV44 RJV45  RJV46   RJV47   RJV48 RJV49
## Akt        0.684 0.81385 0.8990 0.6472 0.796 0.8624 0.78715 0.66457 0.798
## Bax        0.222 0.26018 0.5195 0.2631 0.306 0.4198 0.28066 0.21456 0.255
## Bcl-2      0.000 0.00343 0.0277 0.0291 0.028 0.0292 0.00316 0.00903 0.000
## Bcl-xL     0.122 0.18566 0.3398 0.2222 0.281 0.3708 0.24227 0.33110 0.195
## Beta.Actin 0.316 0.33603 0.5903 0.2755 0.339 0.2946 0.40187 0.20351 0.314
```

Normalisation

There are three primary factors that need to be normalised for in an RPPA experiment (Liu et al. 2014,Wachter et al. (2015)):

1. Spatial bias: Differences in intensity caused by location of the lysate spot on the slide (e.g. rim effects). The Zepto system already accounts for this using the BSA control spots, thus spatial normalisation will not be further considered.
2. Total amount of protein (loading) of different samples on the slides: Although the total proteins in the lysate are gauged before they are printed, this is confounded by lipids and other biological materials in the samples. Thus, the total protein measurement is only a rough estimate.
3. Non linearity of variances: A MA (differences versus means) plot of the differences (M) in intensity of ABs between two samples varies across the spectrum of A, the mean intensity of the two samples. This is especially the case at the upper range and sometimes the lower range of A. This can be seen best when comparing a pair of technical replicates. As there would not be any differential expression, you would expect their intensities to be equal. Some imbalance often occurs and this imbalance may vary depending on the average intensity (A). Variation in the imbalance When normalised well all the points of a MA plot should align with a horizontal line and be evenly distributed about this line.

Normalisation methods

There are a number of software packages specifically developed for analysing RPPA data. The normalisation methodologies offered in each package are detail below and offer insight into the common normalisation techniques used.

- RPPanalyzer:
  - Total protein dye
  - Housekeeping protein
  - Median normalisation
- MIRACLE
  - Housekeeping protein
  - Median normalisation
  - Variable slope

Total protein dye

Include a total protein dye with each experiment to determine the amount of total protein in each sample. A correction factor that reflects the deviation of the protein concentration from the median concentration of all spots is calculated. AB intensities are normalised by dividing by this correction factor. This technique requires running an extra array.

Housekeeping protein

This normalisation methodology is based on the assumption that the levels of housekeeping proteins such as β-Actin is uniform across samples and experiment conditions. Thus any differences in level of these housekeeping proteins is due to differing amount of loading protein. To normalise to housekeeping protein(s) raw intensity values are divided by housekeeping protein RFI values.

Loading control

This approach is utilised by MD Anderson.

Protein effects on intensity are accounted for by dividing all the raw linear intensity by the median for each AB (across all samples). This is the 'median centered ratio'. Sample effects are accounted for by taking the median of the median centered ratios for each sample (across all ABs). This becomes the correction factor for that sample. Raw intensity values are divided by this correction factor to obtain the normalised intensity.

The steps are outlined below:

1. Determine median RFI for each AB (across all samples)
2. Divide each RFI by the median within each AB to get the 'median-centred ratio'.
3. Calculate the median median-cetered ratio for each sample (across all ABs). This is the correction factor for each sample.
4. Divide each median-centred ratio by the correction factor for each sample.

Median normalisation/Global median centering

This method assumes that all measured proteins reflect the total protein amount of one sample. Thus the median AB of a sample estimates sample loading. The median value of all AB signals for a sample is used to normalise the raw intensity values for each AB. This is one of the 'simpliest' methods (involving the least steps) however is biased when the number of ABs is <100 (Liu et al. 2014).

Invariable protein normalisation

This methodology was proposed by Liu et al. (Liu et al. 2014). It proceeds thus:

1. Rank the intensity of ABs for each sample so you have ranked ABs for each sample from the highest expressing AB to lowest expressing AB.
2. Calculate the variance of the ranks for each AB. Remove the AB with the highest rank variance.
3. Re-rank the intensity of ABs.
4. Repeat the steps 2 and 3 until the number of remaining markers reaches a predetermined number (100 kept in Liu et al. study).
5. Trim intensities for each AB e.g. the highest and lowest 25% of values are removed from the data set.
6. Average the remaining values of every AB across all samples and use this as a virtual reference sample.
7. Normalise each sample to the virtual reference sample by lowess smoothing using a MA plot. The normalised values are generated using the residuals of the fit.

The aim of this methodology is to determine the most uniformly expressed proteins and use these as an effective 'Housekeeping' protein to normalise to.

Variable slope

This methodology was developed by MD Anderson. It aims to normalise for between array, between sample and protein effects, which result from variation in the estiamted slope parameters from the calibration curve estiamted in the quantification step. It was designed specifically for quantification (from each dilution series) via the joint sample model, ‘SuperCurve’,. This methodology may not translate to protein expression data not quantified using SuperCurve and will thus not be considered here.

How to evaluate normalisation methodologies

To be able to evaluate each normalisation methodology, one must be able to compare how effectively each method normalises for total protein loading and non-linearity of variances. Normalisation of non-linearity of variances is effectively assessed using the MA plot. Normalisation of total protein loading is more difficult to assess and must be done indirectly. Two techniques have beens suggested and are discussed below.

Pairwise correlation coefficients (Spearman ρ)

This plot is used to evaluate the between protein correlations. For each pair of proteins (for the 29 proteins in this experiment, there are 406 different possible pairs) the Spearman ρ is calculated across all samples. This is then sorted and plotted. We expect the positive and negative correlation coefficients to be largely equal. If there is an abnormally large number of protein pairs with positive correlation, it may be due to a sample loading effect where some proteins are high in all samples, thus are ranked simiarly highly for those samples. Samples with low loading result in low levels of all proteins, and are ranked similarly low for those samples. There would thus be high correlation between a larger proportion of proteins. We expect protein expression to be inherently different due to expression, phosphorylation levels and/or AB affinity.

Distrubution of sample medians

To evaluation effectiveness of loading correction, the distribution of the median raw intensity values for each sample should have a narrow dispersion - i.e. if protein loading is normalised for, you would not expect the median raw intensity value of your samples to vary significantly.

Evaluation of normalisation methodologies

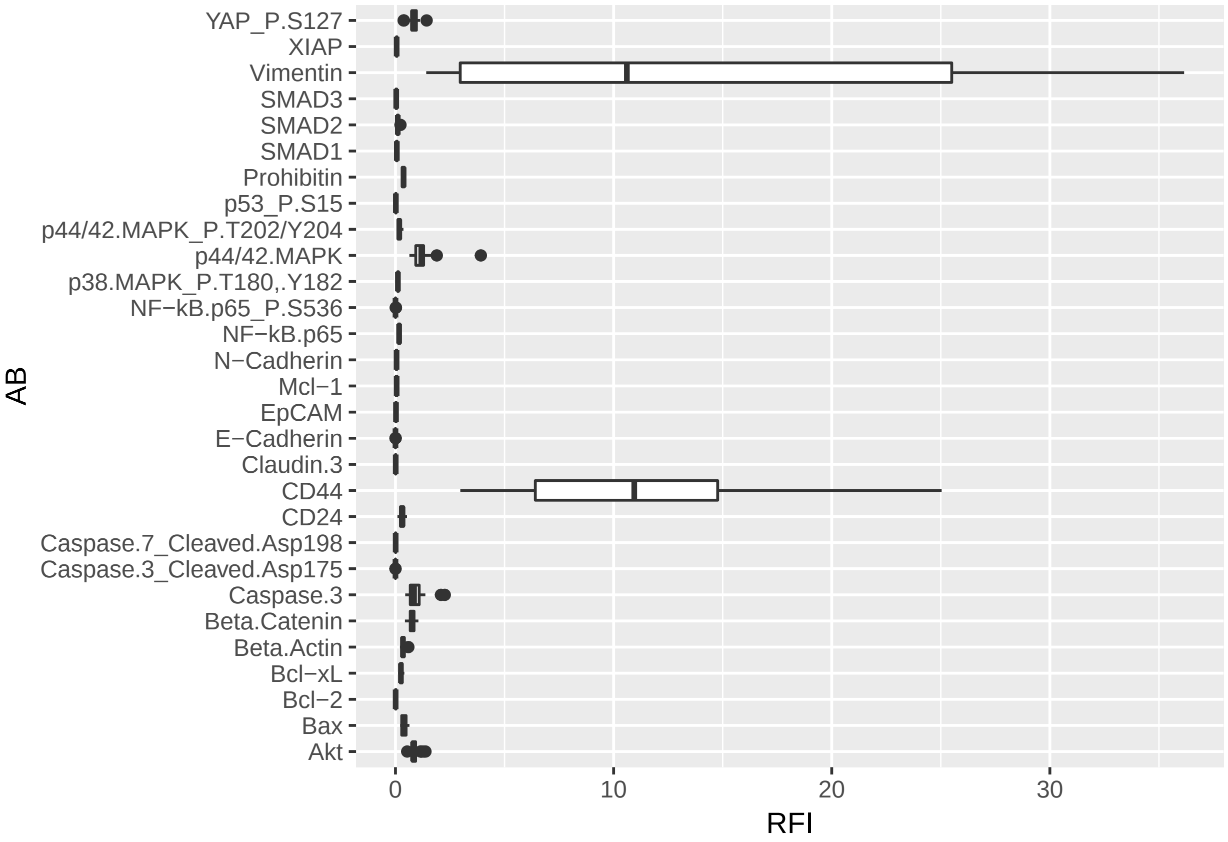
Using our data set we will normalise according to various methods and evaulate their performance.

Raw intensity values

Visualise raw intensity values by AB via bloxplot:

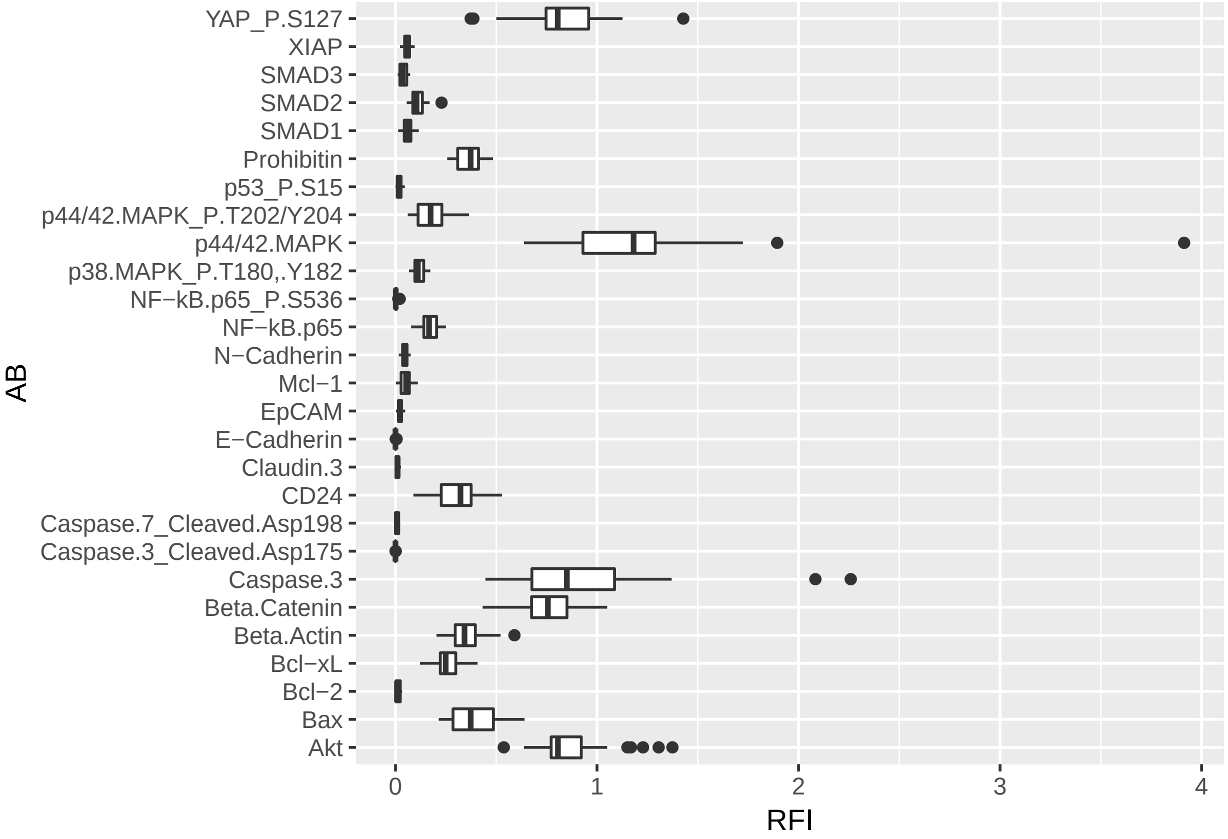
```r
raw_tidy <- melt(raw)
colnames(raw_tidy) <- c("AB", "Sample", "RFI")
#Convert raw matrix into tidy data format
```

```r
ggplot(raw_tidy, aes(y = RFI, x = AB)) +
  geom_boxplot() +
  coord_flip()
```



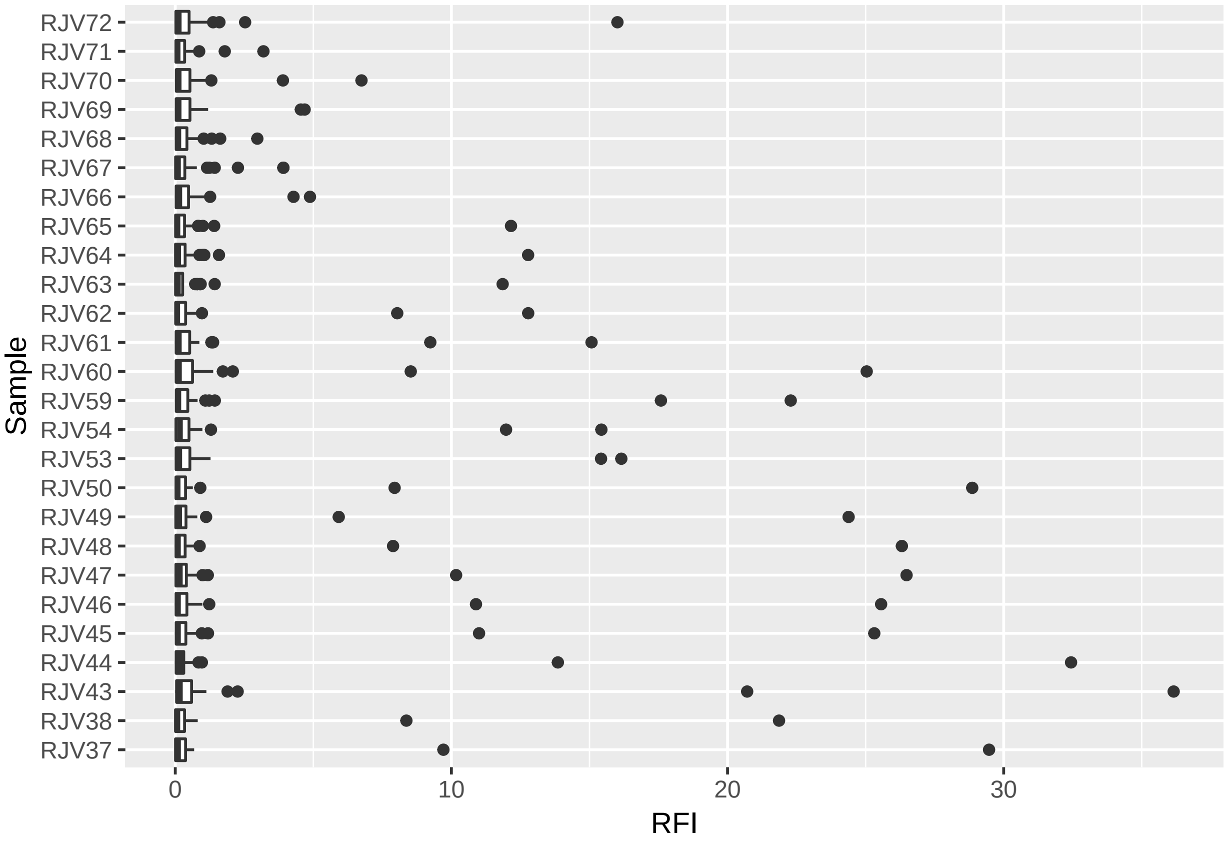
Remove “CD44” and “Vimentin” and remake boxplots to better visualise the other AB’s.

```
raw_tidy %>%
  filter(! AB %in% c("CD44", "Vimentin")) %>%
  ggplot(aes(y = RFI, x = AB)) +
  geom_boxplot() +
  coord_flip()
```



Visualise raw intensity values per sample via boxplot:

```
ggplot(raw_tidy, aes(y = RFI, x = Sample)) +
  geom_boxplot() +
  coord_flip()
```



Housekeeping normalisation

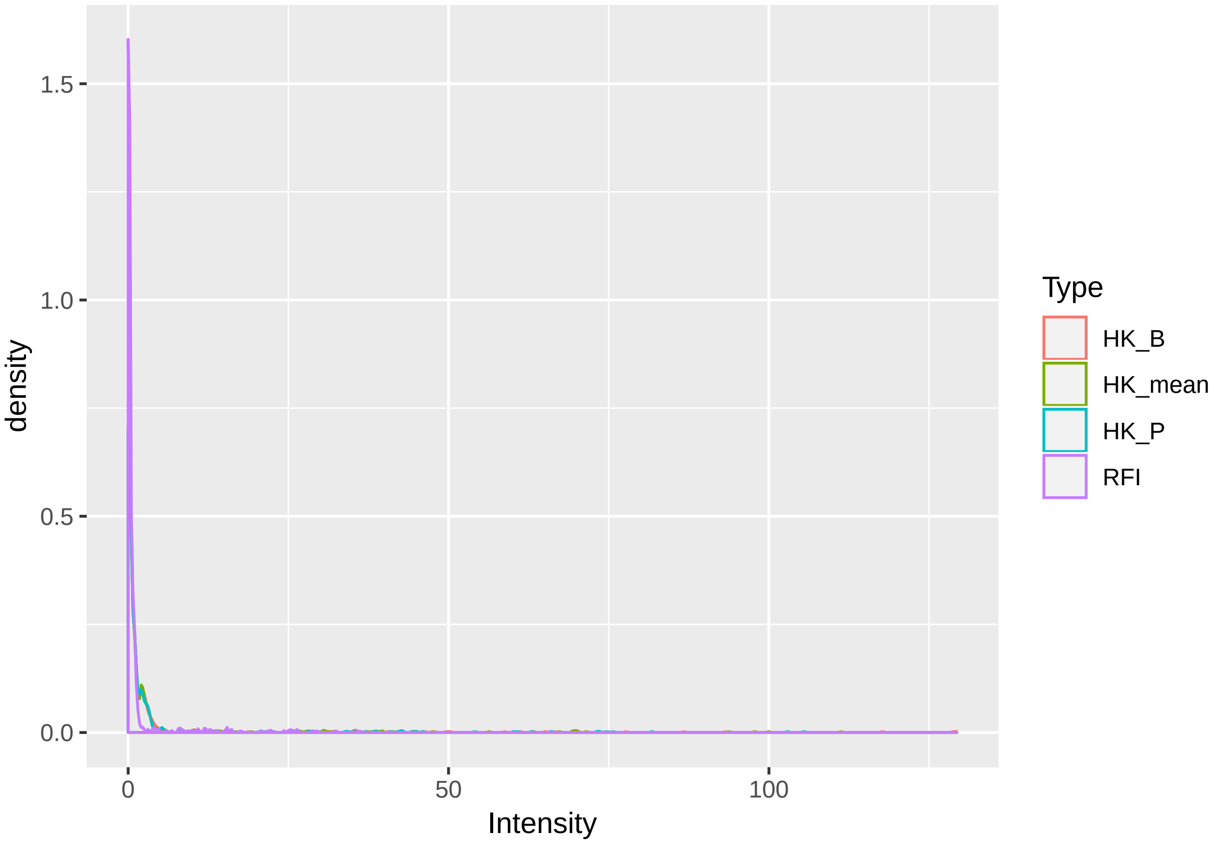
The housekeeping proteins in this experiment were β-Actin and Prohibitin.

Normalisation against β-Actin, Prohibitin and the mean of these two housekeeping proteins will all be performed. The results will then be compared.

```r
hk <- raw_tidy %>%
  group_by(Sample) %>%
  mutate(HK_B = RFI / RFI[AB == "Beta.Actin"]) %>%
  mutate(HK_P = RFI / RFI[AB == "Prohibitin"]) %>%
  mutate(meanHK = mean(RFI[AB %in% c("Beta.Actin", "Prohibitin")])) %>%
  ungroup() %>%
  mutate(HK_mean = RFI / meanHK) %>%
  select(-meanHK)

hk <- hk %>%
  gather(HK_B, HK_P, HK_mean, RFI, key = "Type", value = "Intensity")
```

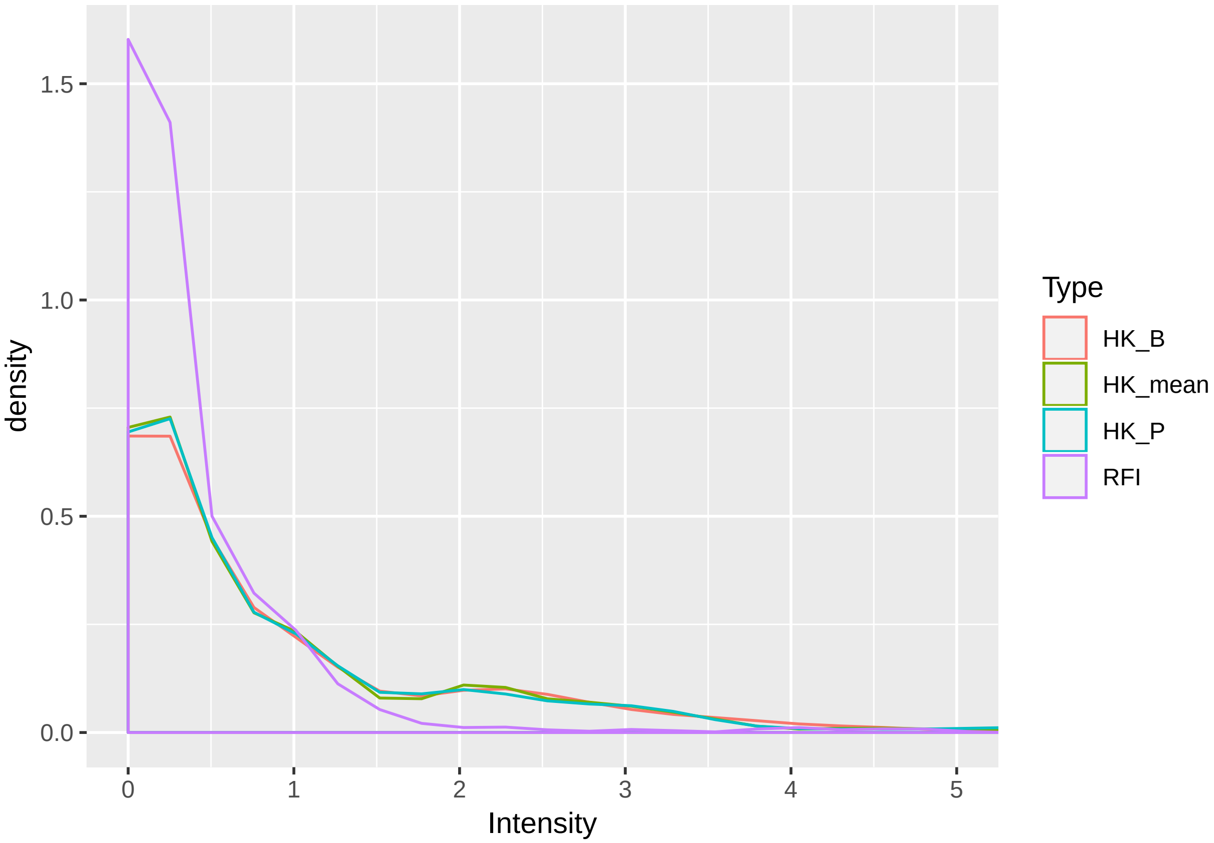
Plot the distribution of the intensity values for each of the normalisation methods and the raw RFI.

```r
ggplot(hk, aes(x = Intensity, colour = Type)) +
  geom_density()
```



Zoom in on the low intensity values to better visualise the data:

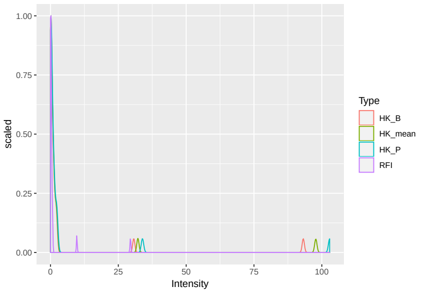
```
ggplot(hk, aes(x = Intensity, colour = Type)) +
  geom_density() +
  coord_cartesian(xlim = c(0,5))
```

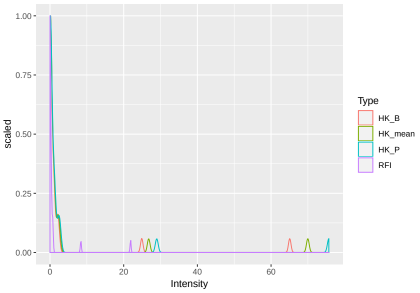


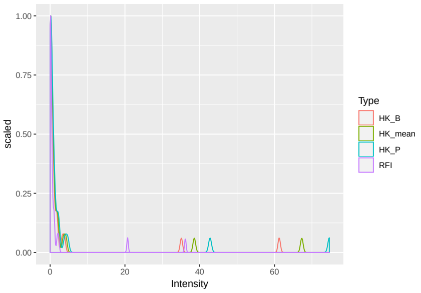
The distribution of the normalised intensity values are very similar for all three housekeeping normalisation methods. Overall, housekeeping normalisation has reduced the number of ABs with low intensity values.

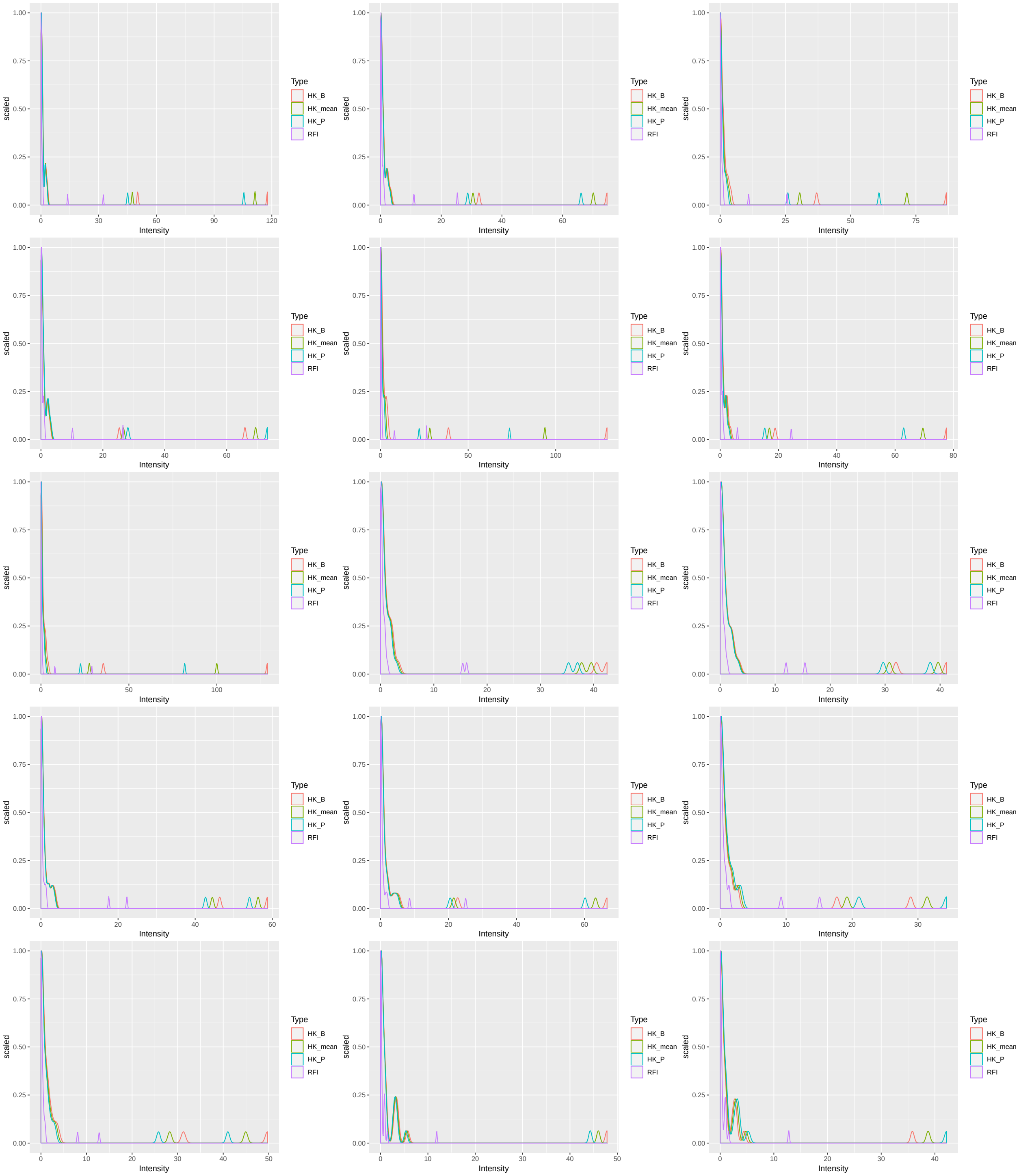
Visualise the change in intensity distribution for each sample:

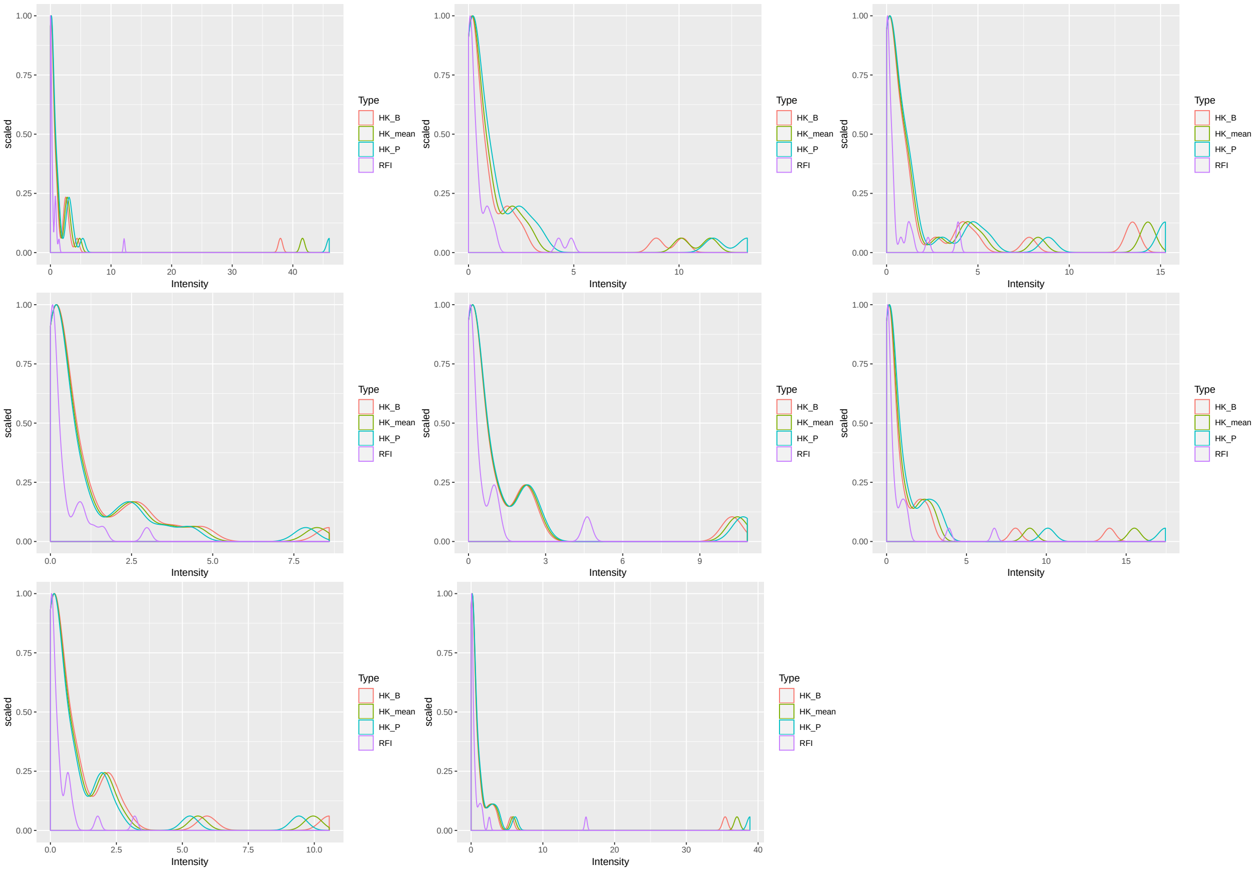
```
samples <- unique(hk$Sample)
for (i in 1:26){
print(
  hk %>%
  filter(Sample == samples[i]) %>%
ggplot(aes(x = Intensity, colour = Type, y = ..scaled..)) +
  geom_density()
)
}
```





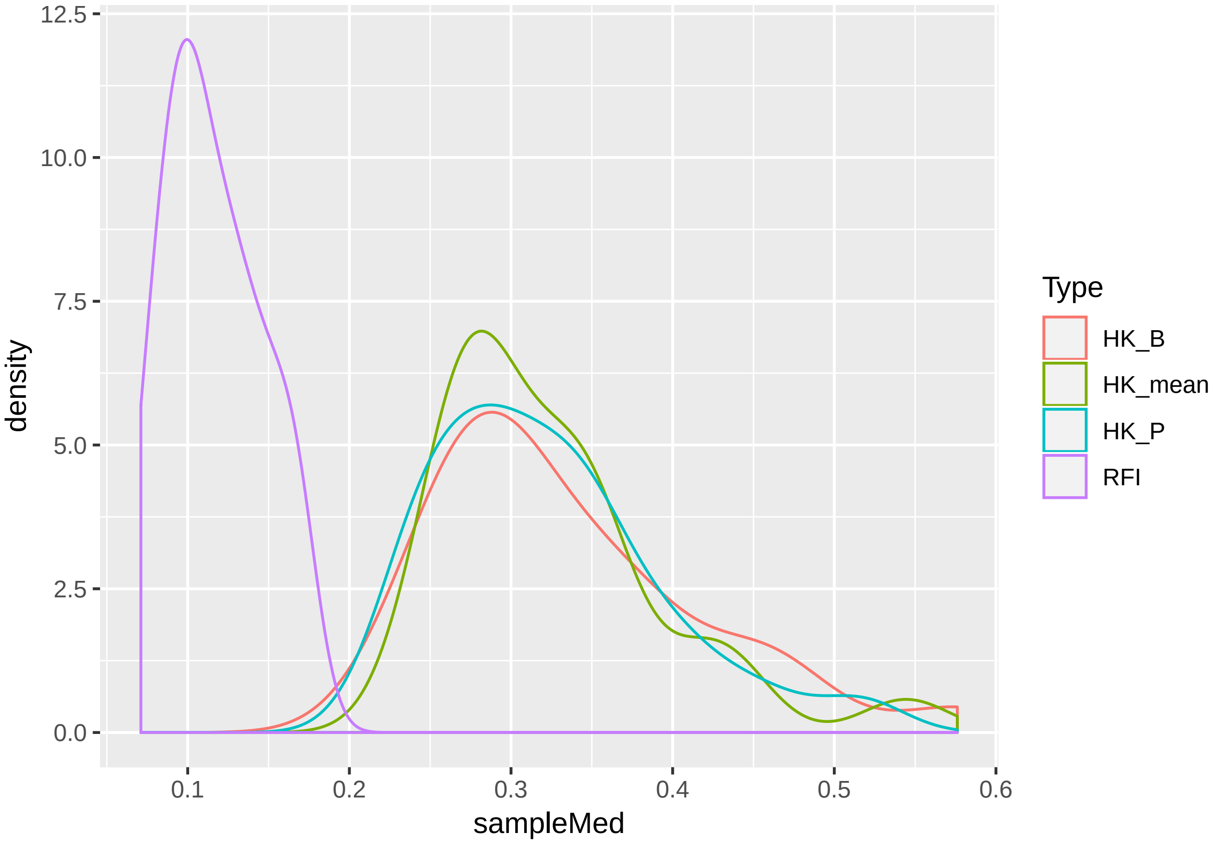



scaled
1.00
0.75
0.50
0.25
0.00
Intensity
Type
HK_B
HK_mean
HK_P
RFI
0 30 60 90 120
0 20 40 60
0 25 50 75
0 20 40 60
0 50 100
0 20 40 60 80
0 50 100
0 10 20 30 40
0 10 20 30 40
0 20 40 60
0 20 40 60
0 10 20 30
0 10 20 30 40 50
0 10 20 30 40 50
0 10 20 30 40



Plot the median raw intensity values per sample:

```
hk %>%
  group_by(Sample, Type) %>%
  summarise(sampleMed = median(Intensity, na.rm = TRUE)) %>%
  ggplot(aes(x = sampleMed, colour = Type)) +
  geom_density()
```

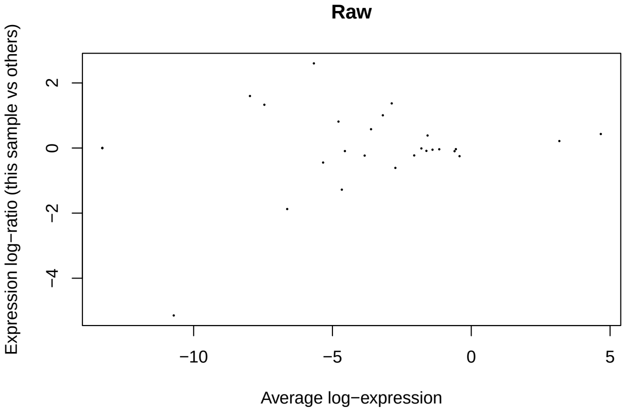


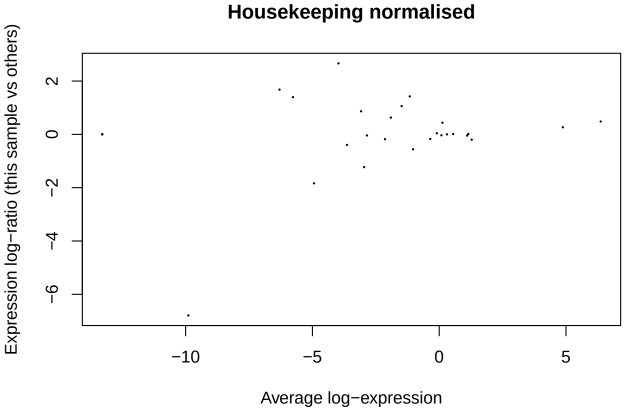
The sample median intensity values are higher (as suggested by the distribution of intensity values above) and more spread out for the housekeeping normalised values when compared to the raw intensity values.

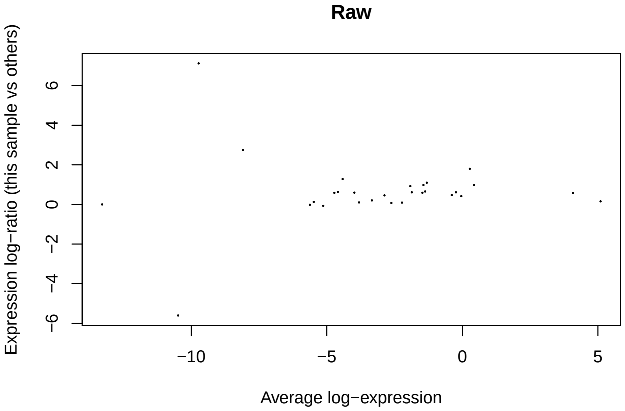
MA plots of the difference and average intensity levels of each pair of biological replicates are plotted below. Each point represents a protein. You expect the points to lie along 0 and the variation of points around 0 to be similar along A.

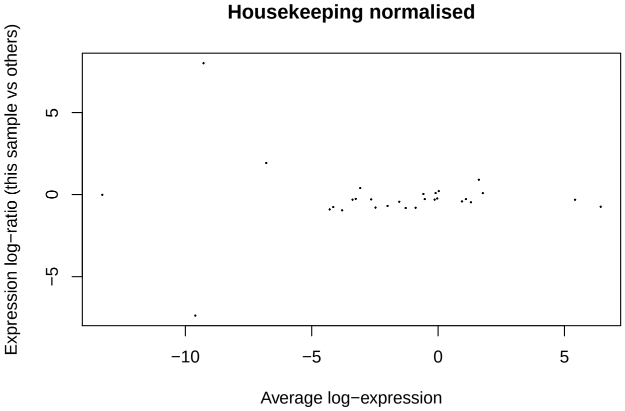
As all three housekeeping normalisations are similar, we will only compare raw values with that normalised to the mean of β-Actin and Prohibitin.

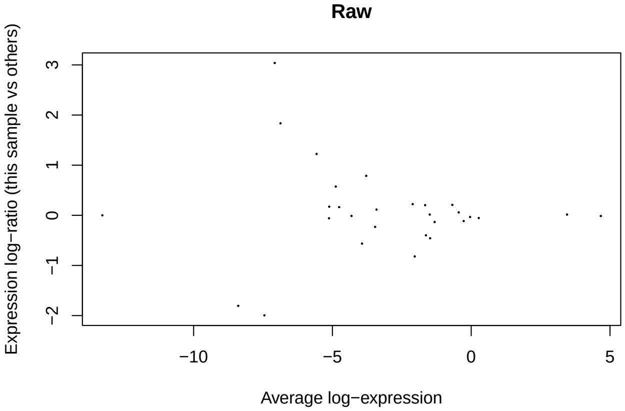
```
raw_matrix <- log(as.matrix(raw) + 0.0001, base = 2)
hk_matrix <- acast(
  hk %>%
  filter(Type == "HK_mean") %>%
  mutate(log2 = log(Intensity + 0.0001, base = 2)) %>%
  select(-c(Type, Intensity)),

  AB~Sample,
  value.var = "log2"
)

for (i in 1:13){
plotMA(raw_matrix[,(2*i-1):(2*i)], main = "Raw")
plotMA(hk_matrix[,(2*i-1):(2*i)], main = "Housekeeping normalised")
}
```

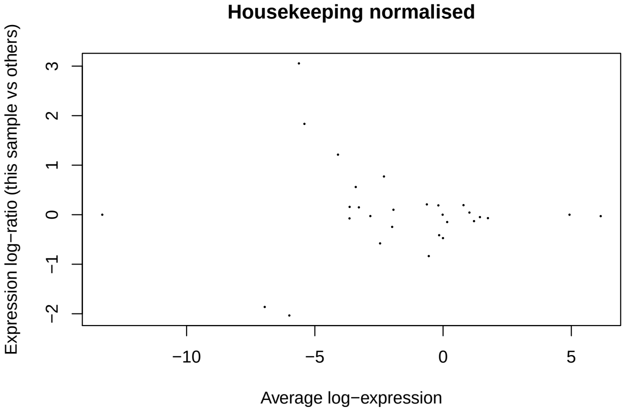

Raw
Expression log−ratio (this sample vs others)
2
0
−2
−4
−10
−5
0
5
Average log−expression

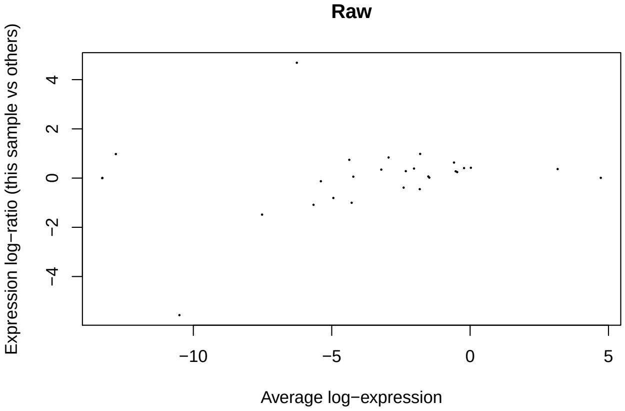

Housekeeping normalised
Expression log−ratio (this sample vs others)
2
0
−2
−4
−6
−10
−5
0
5
Average log−expression

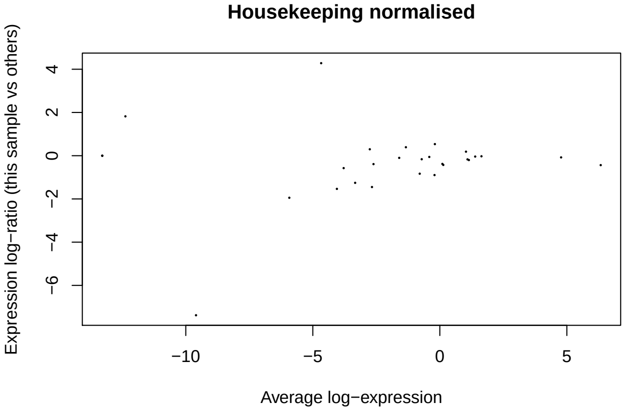

Raw
Expression log−ratio (this sample vs others)
6
4
2
0
−2
−4
−6
−10
−5
0
5
Average log−expression

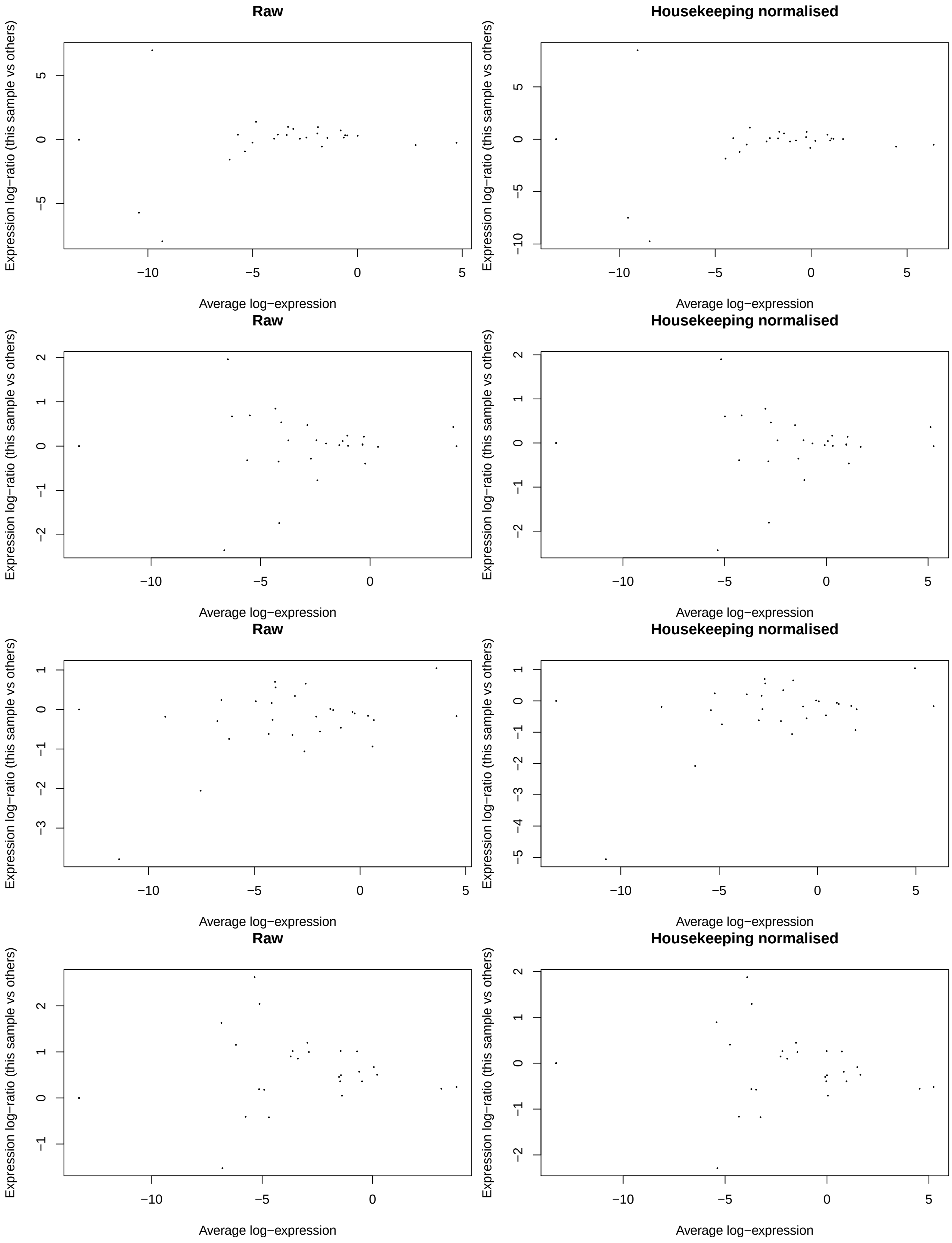

Housekeeping normalised
Expression log−ratio (this sample vs others)
5
0
−5
−10
−5
0
5
Average log−expression

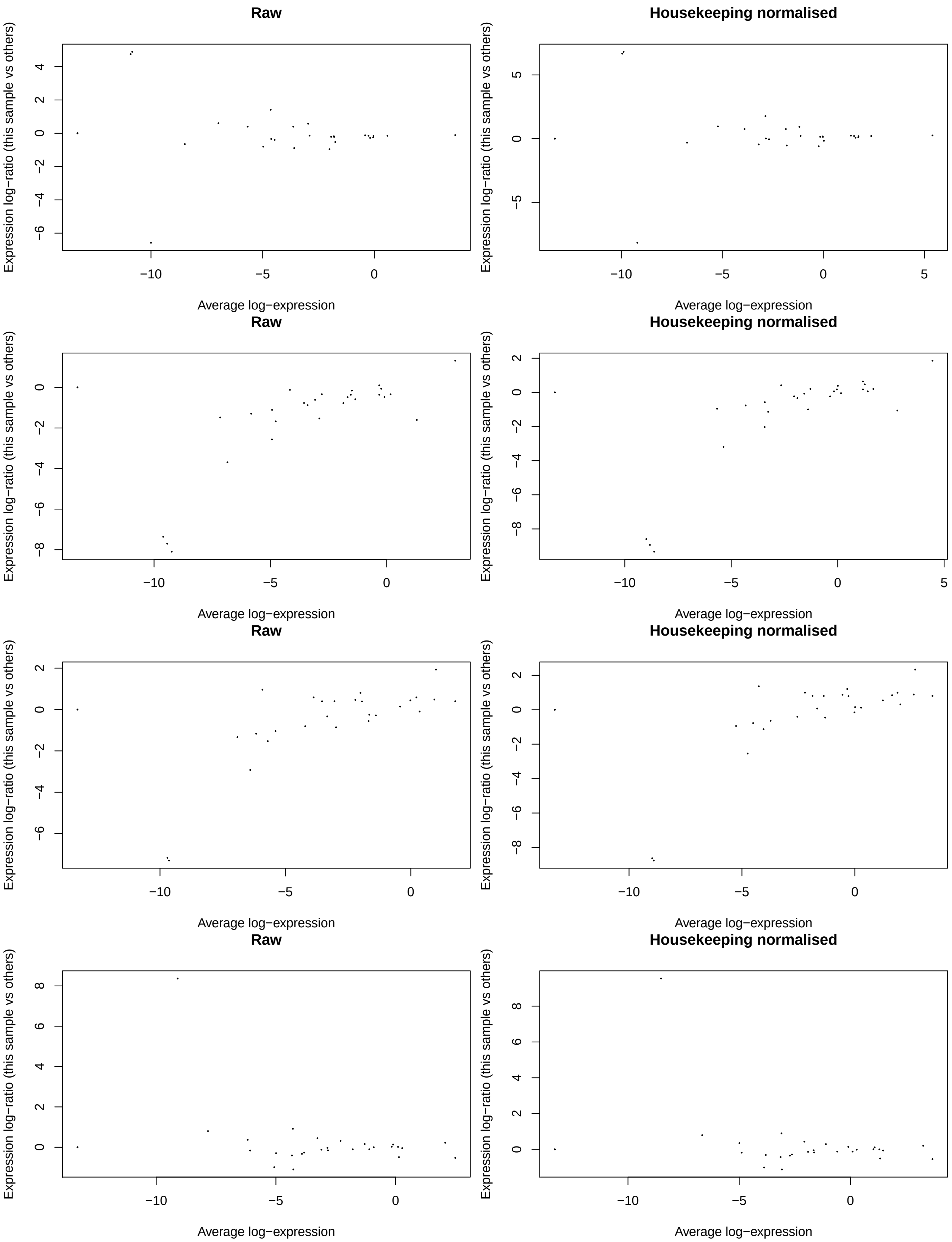

Raw
Expression log−ratio (this sample vs others)
3
2
1
0
−1
−2
−10
−5
0
5
Average log−expression

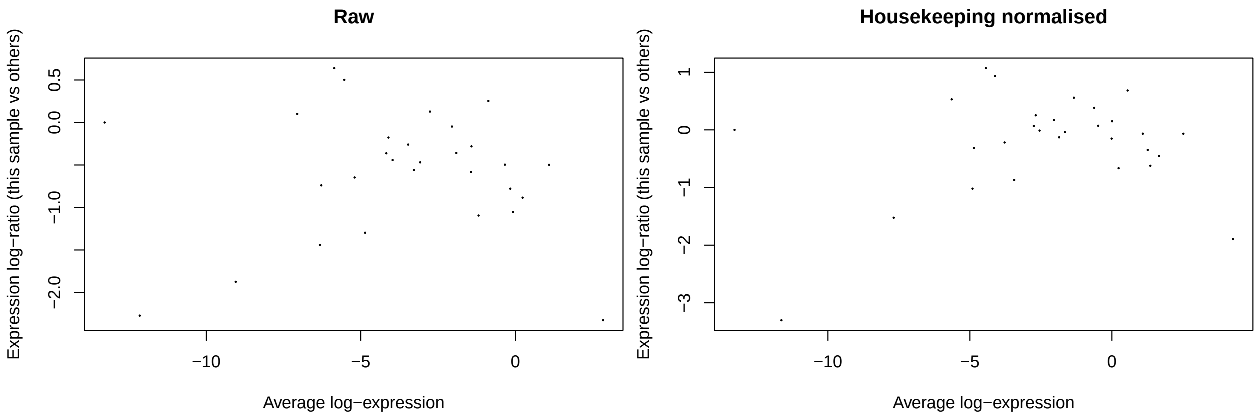

Housekeeping normalised
Expression log−ratio (this sample vs others)
3
2
1
0
−1
−2
−10
−5
0
5
Average log−expression


Raw
Expression log−ratio (this sample vs others)
4
2
0
−2
−4
−10
−5
0
5
Average log−expression


Housekeeping normalised
Expression log−ratio (this sample vs others)
4
2
0
−2
−4
−6
−10
−5
0
5
Average log−expression

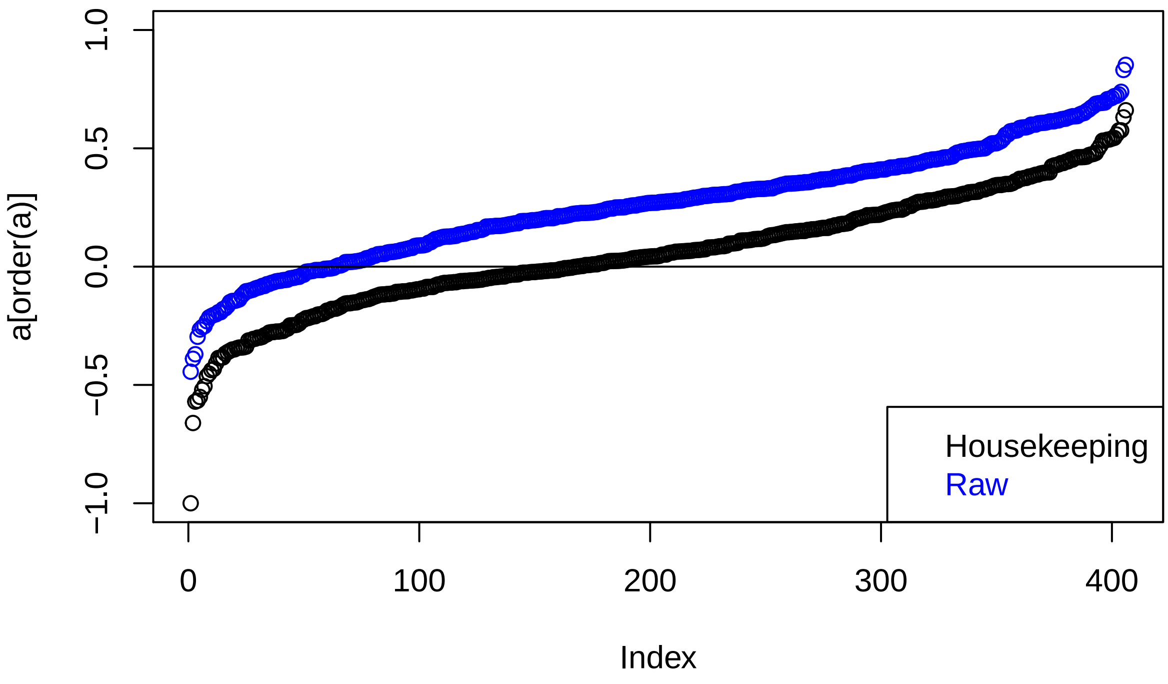

Raw
Expression log−ratio (this sample vs others)
5
0
−5
−10
−5
0
5
Average log−expression
Housekeeping normalised
Expression log−ratio (this sample vs others)
5
0
−5
−10
−10
−5
0
5
Average log−expression
Raw
Expression log−ratio (this sample vs others)
2
1
0
−1
−2
−10
−5
0
Average log−expression
Housekeeping normalised
Expression log−ratio (this sample vs others)
2
1
0
−1
−2
−10
−5
0
5
Average log−expression
Raw
Expression log−ratio (this sample vs others)
1
0
−1
−2
−3
−10
−5
0
5
Average log−expression
Housekeeping normalised
Expression log−ratio (this sample vs others)
1
0
−1
−2
−3
−4
−5
−10
−5
0
5
Average log−expression
Raw
Expression log−ratio (this sample vs others)
2
1
0
−1
−10
−5
0
Average log−expression
Housekeeping normalised
Expression log−ratio (this sample vs others)
2
1
0
−1
−2
−10
−5
0
5
Average log−expression


Raw
Housekeeping normalised
Expression log−ratio (this sample vs others)
Average log−expression
4
2
0
−2
−4
−6
5
0
−5
−10
−5
0
5
Raw
Housekeeping normalised
0
−2
−4
−6
−8
2
Raw
Housekeeping normalised
2
0
−2
−4
−6
−8
Raw
Housekeeping normalised
8
6
4
2
0



A spearman rank correlation plot allows us to evaluate the between protein correlations. We expect a largely equal psotive and negative correlation as discussed above. Blue points are the correlations between proteins for raw RFI values. Black are correlations between proteins for the values normalised to mean of the two housekeeping proteins.

```
rank_hk <- hk %>%
  group_by(Type,AB) %>%
  mutate(Rank = rank(Intensity)) %>%
  select(-Intensity) %>%
  spread(key = Type, value = Rank)


spearmanPlot(rank_hk, "HK_mean", "Housekeeping")
```



After normalisation, there is a much more even spread of correlations above and below 0, as we would expect for data well normalised for total protein loading.

Loading control

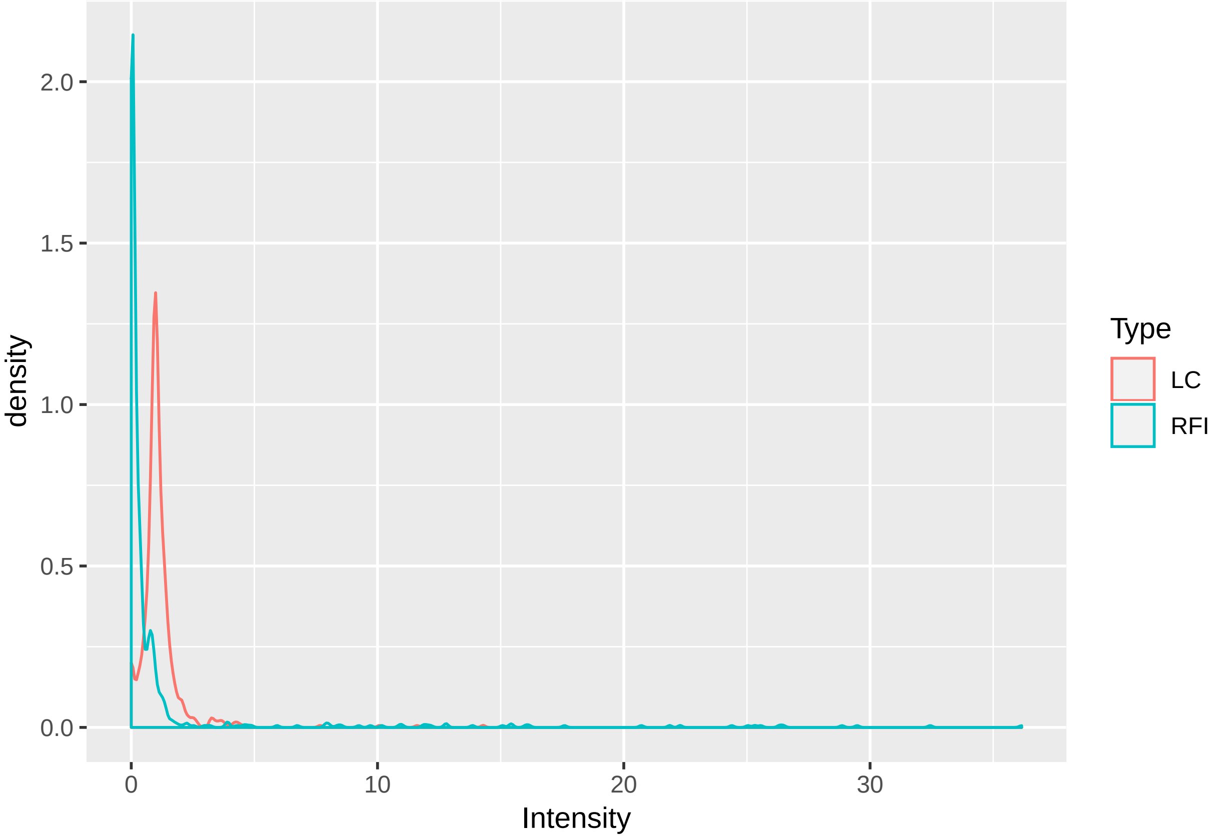
Calculate normalised values using loading control method:

```
lc <- raw_tidy %>%
  group_by(AB) %>%
  mutate(medianAB = median(RFI)) %>%
  #Find median RFI for each AB
  ungroup() %>%
  mutate(medCenter = RFI/medianAB) %>%
  #Divide RFI by the median AB, for each AB
  group_by(Sample) %>%
  mutate(CF = median(medCenter, na.rm = TRUE)) %>%
  #Find the median medCenter for each sample. Remove NA values
  ungroup() %>%
  mutate(LC = medCenter/CF) %>%
  select(c(AB, Sample, RFI, LC))

lc <- lc %>%
  gather(LC, RFI, key = "Type", value = "Intensity")
```

Plot distribution of all intensity values for raw and loading control normalised data:

```
ggplot(lc, aes(x = Intensity, colour = Type)) +
  geom_density()
```

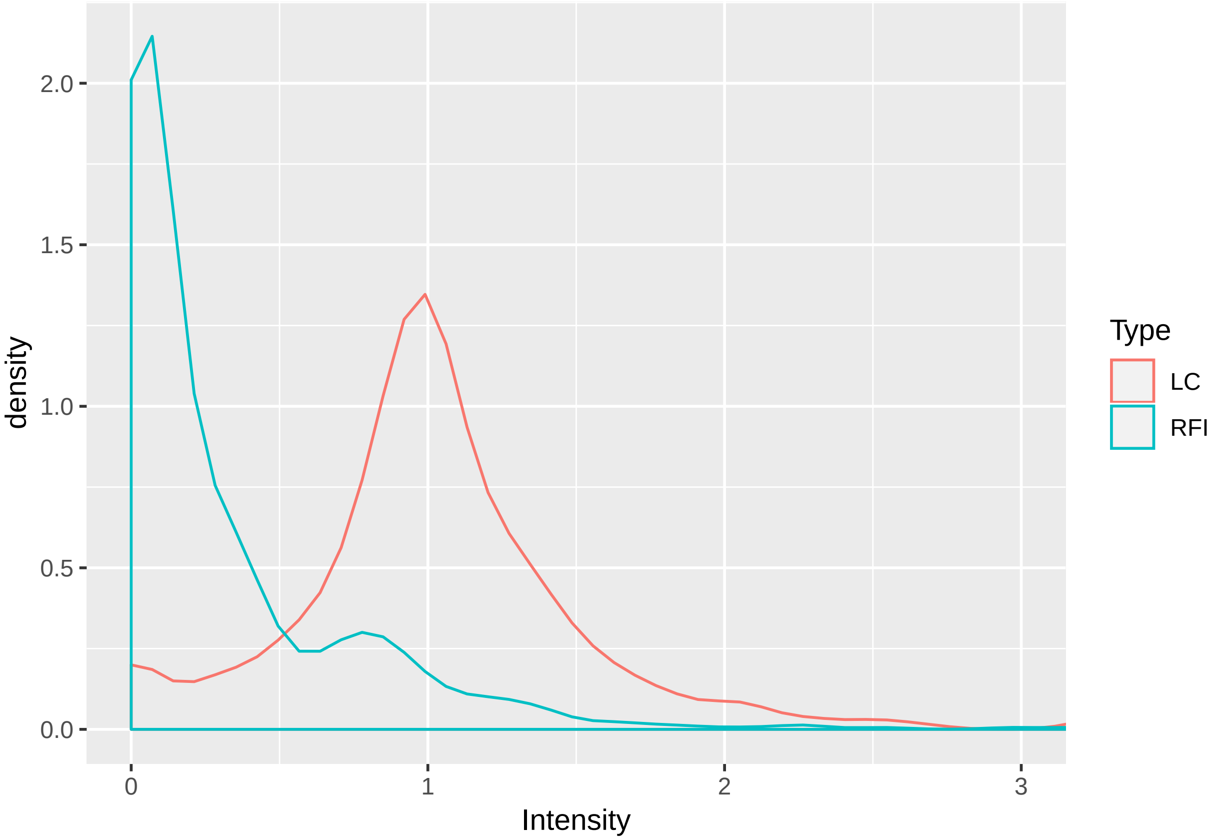
```
## Warning: Removed 52 rows containing non-finite values (stat_density).
```



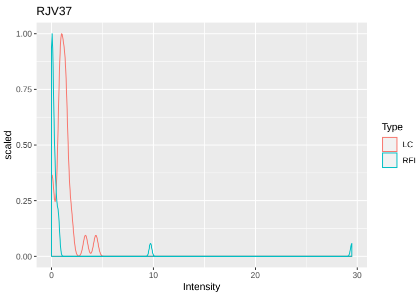
Zoom in on the low intensity values to better visualise the data:

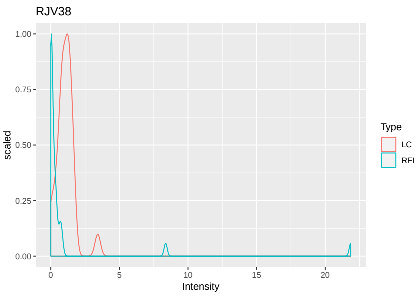
```r
ggplot(lc, aes(x = Intensity, colour = Type)) +
  geom_density() +
  coord_cartesian(xlim = c(0,3))
```

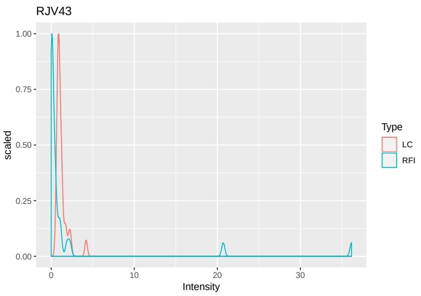
```
## Warning: Removed 52 rows containing non-finite values (stat_density).
```

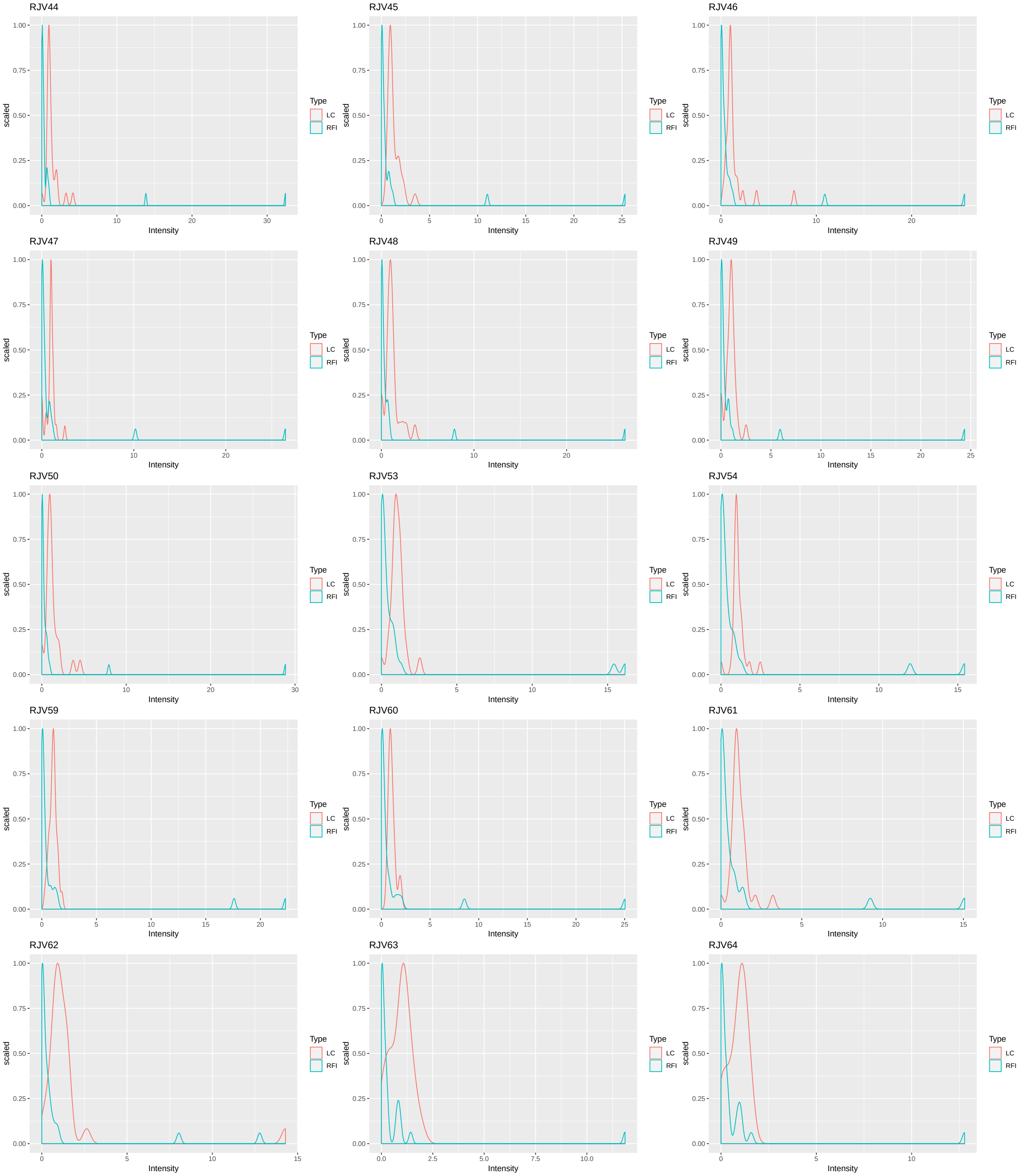


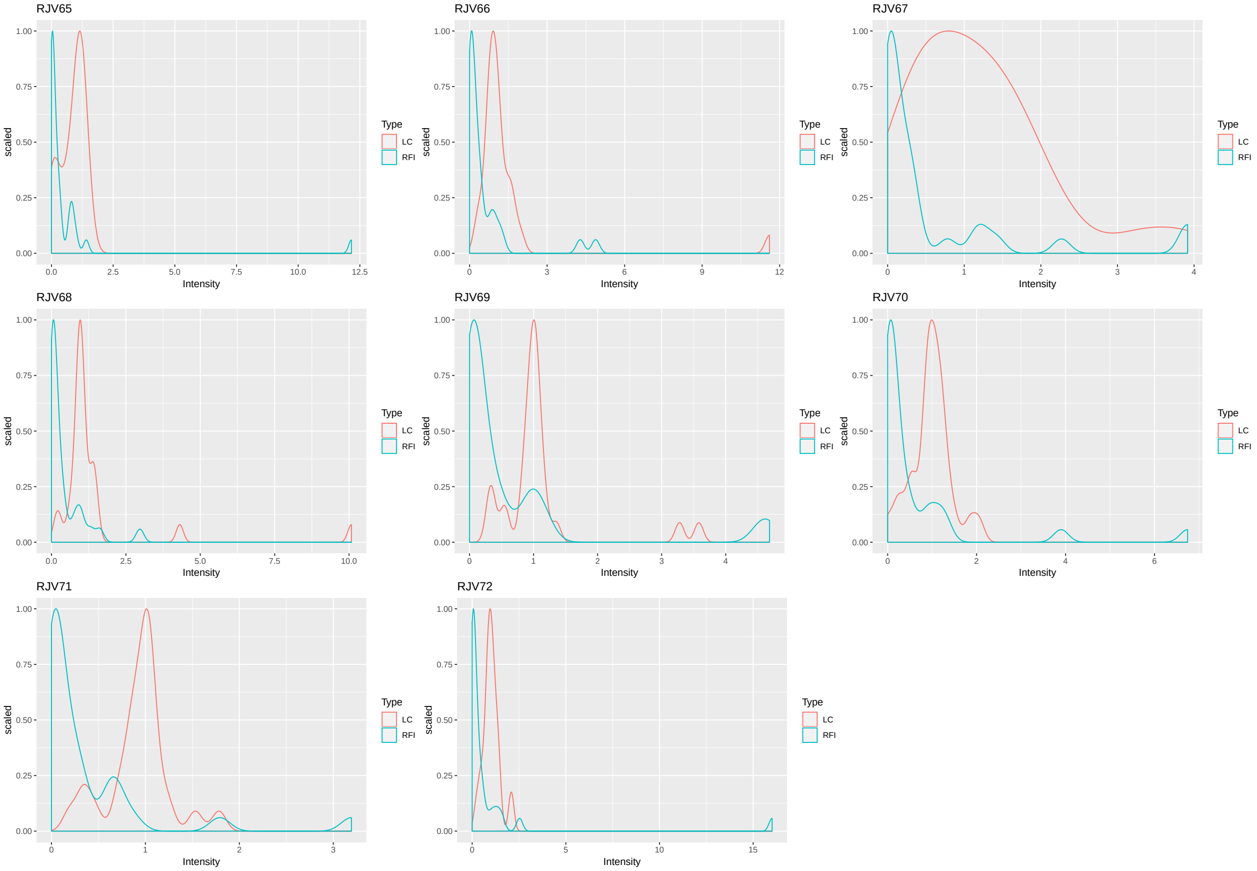
```r
samples <- unique(lc$Sample)
for (i in 1:26){
print(
  lc %>%
  filter(Sample == samples[i]) %>%
ggplot(aes(x = Intensity, colour = Type, y = ..scaled..)) +
  geom_density() +
  labs(title = samples[i])
)
}
```








RJV44
scaled
1.00
0.75
0.50
0.25
0.00
0
10
20
30
Intensity
Type
LC
RFI
RJV45
0
5
10
15
20
25
Intensity
RJV46
0
10
20
Intensity
RJV47
0
10
20
Intensity
RJV48
0
10
20
Intensity
RJV49
0
5
10
15
20
25
Intensity
RJV50
0
10
20
30
Intensity
RJV53
0
5
10
15
Intensity
RJV54
0
5
10
15
Intensity
RJV59
0
5
10
15
20
Intensity
RJV60
0
5
10
15
20
25
Intensity
RJV61
0
5
10
15
Intensity
RJV62
0
5
10
15
Intensity
RJV63
0.0
2.5
5.0
7.5
10.0
Intensity
RJV64
0
5
10
Intensity

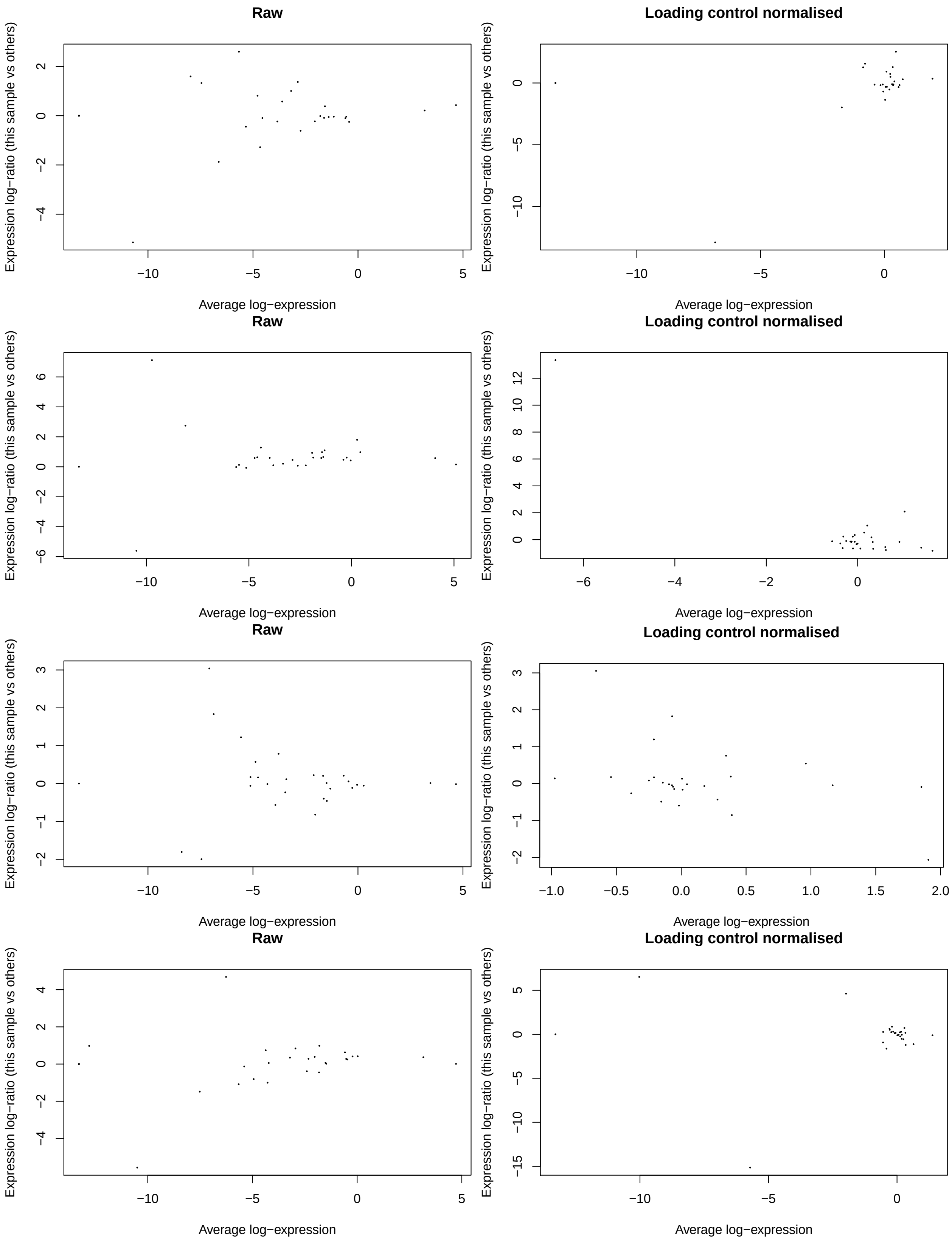


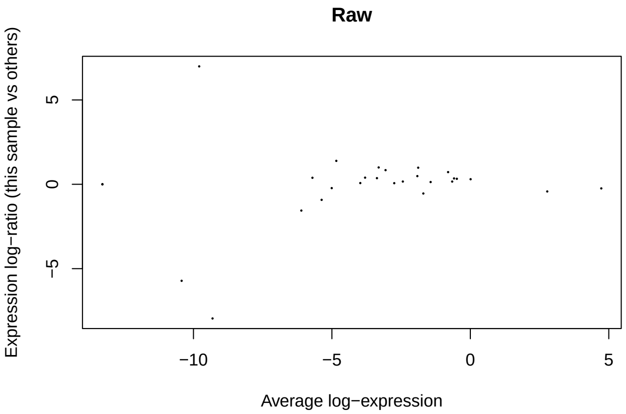
LC normalisation does not appear to significantly change the overall distribution of intensity values.

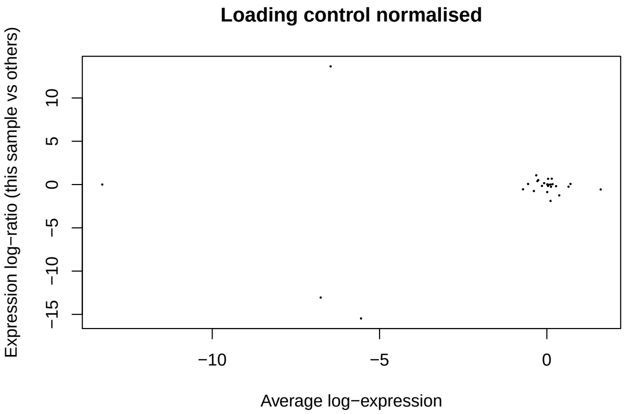
LC normalisation results in the median sample intensities to all be 1 thus the distribution of sample medians is not useful in evaluating its performance.

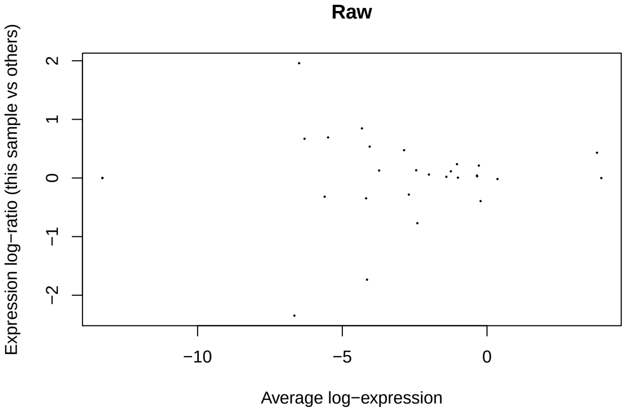
Plot MA graphs for raw and loading control normalised intensities:

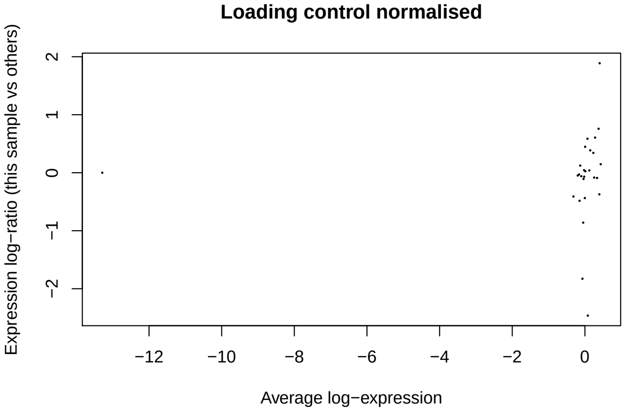
```r
raw_matrix <- log(as.matrix(raw) + 0.0001, base = 2)

lc_matrix <- acast(
  lc %>%
  filter(Type == "LC") %>%
  mutate(log2 = log(Intensity + 0.0001, base = 2)) %>%
  select(-c(Type, Intensity)),

  AB~Sample,
  value.var = "log2"
)

for (i in 1:13){
plotMA(raw_matrix[,(2*i-1):(2*i)], main = "Raw")
plotMA(lc_matrix[,(2*i-1):(2*i)], main = "Loading control normalised")
}
```

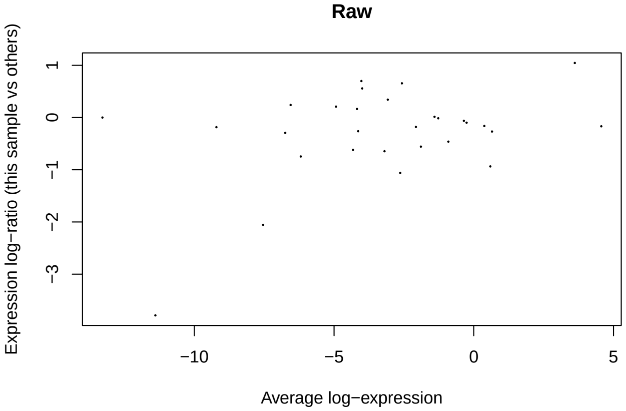

Raw
Loading control normalised
Expression log−ratio (this sample vs others)
Average log−expression
−10
−5
0
5
2
0
−2
−4
−5
−10
Raw
Loading control normalised
6
4
2
0
−2
−4
−6
12
10
8
−6
−4
−2
Raw
Loading control normalised
3
1
−1
−2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Raw
Loading control normalised
−15

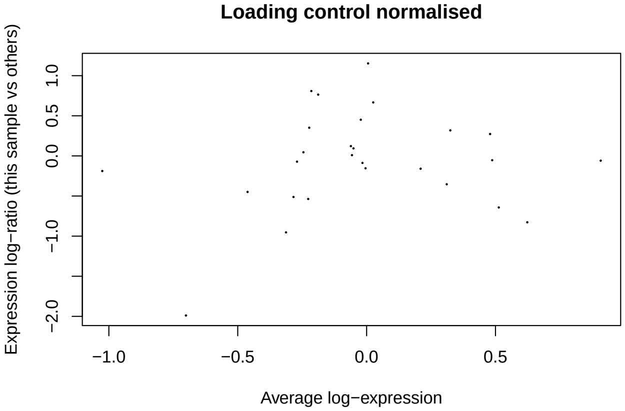

Raw
Expression log−ratio (this sample vs others)
5
0
−5
−10
−5
0
5
Average log−expression

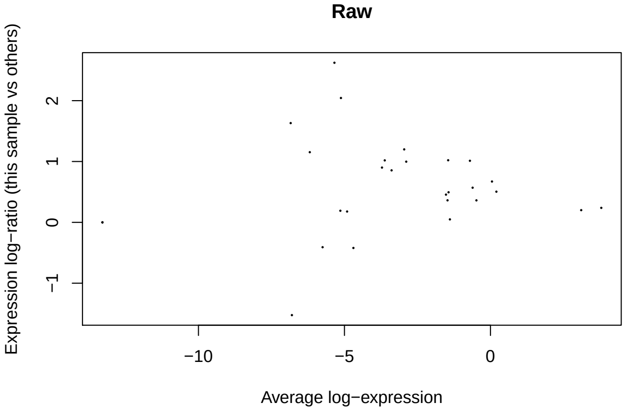

Loading control normalised
Expression log−ratio (this sample vs others)
10
5
0
−5
−10
−15
−10
−5
0
Average log−expression

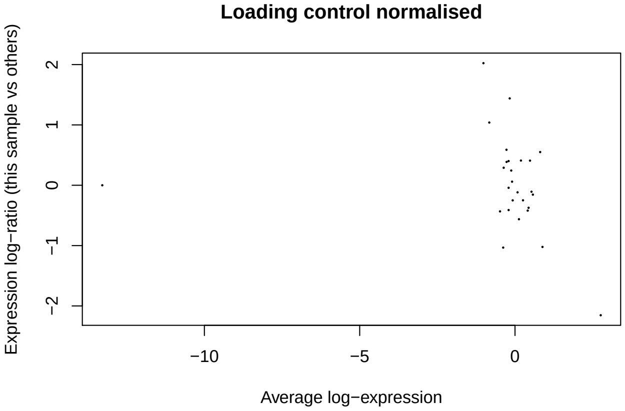

Raw
Expression log−ratio (this sample vs others)
2
1
0
−1
−2
−10
−5
0
Average log−expression

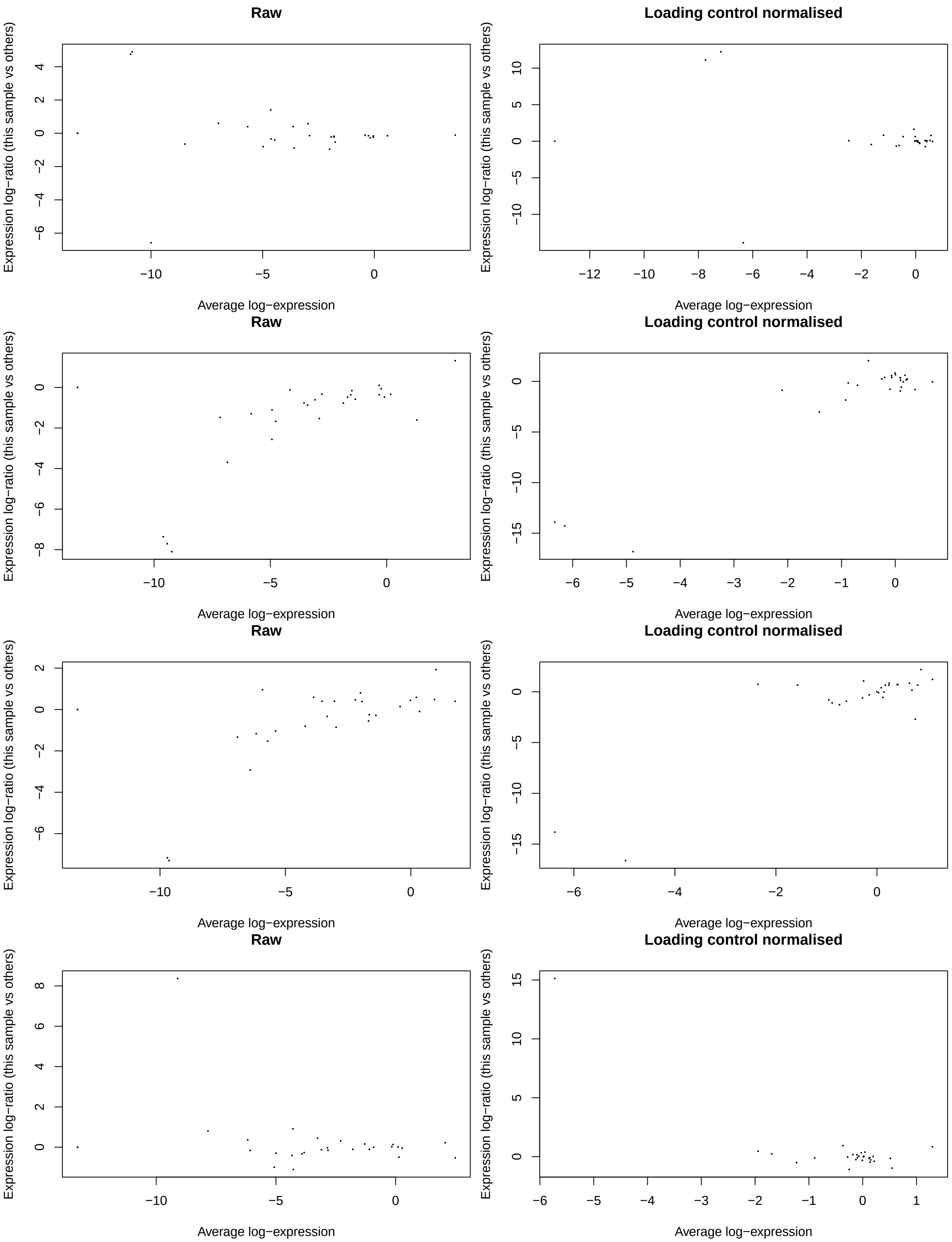

Loading control normalised
Expression log−ratio (this sample vs others)
2
1
0
−1
−2
−12
−10
−8
−6
−4
−2
0
Average log−expression

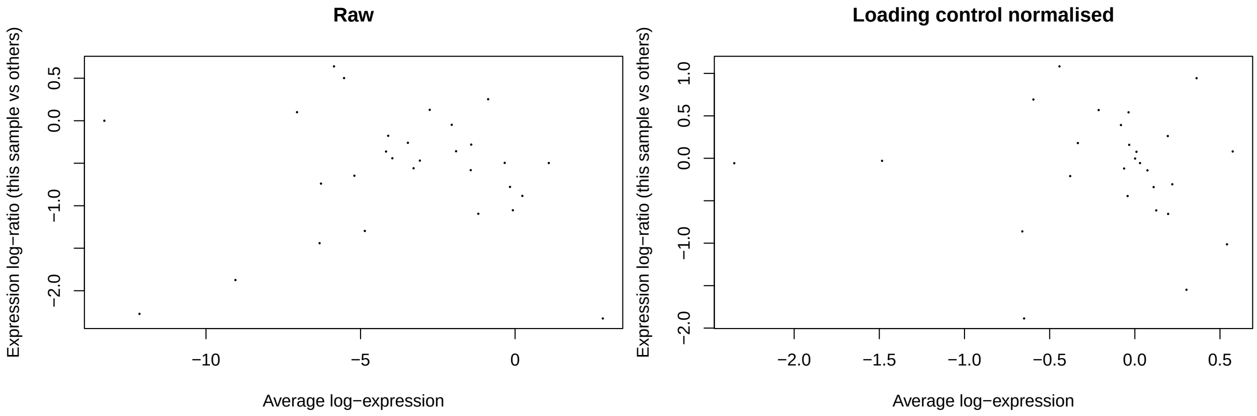

Raw
Expression log−ratio (this sample vs others)
1
0
−1
−2
−3
−10
−5
0
5
Average log−expression


Loading control normalised
Expression log−ratio (this sample vs others)
1.0
0.5
0.0
−1.0
−2.0
−1.0
−0.5
0.0
0.5
Average log−expression


Raw
Expression log−ratio (this sample vs others)
2
1
0
−1
−10
−5
0
Average log−expression

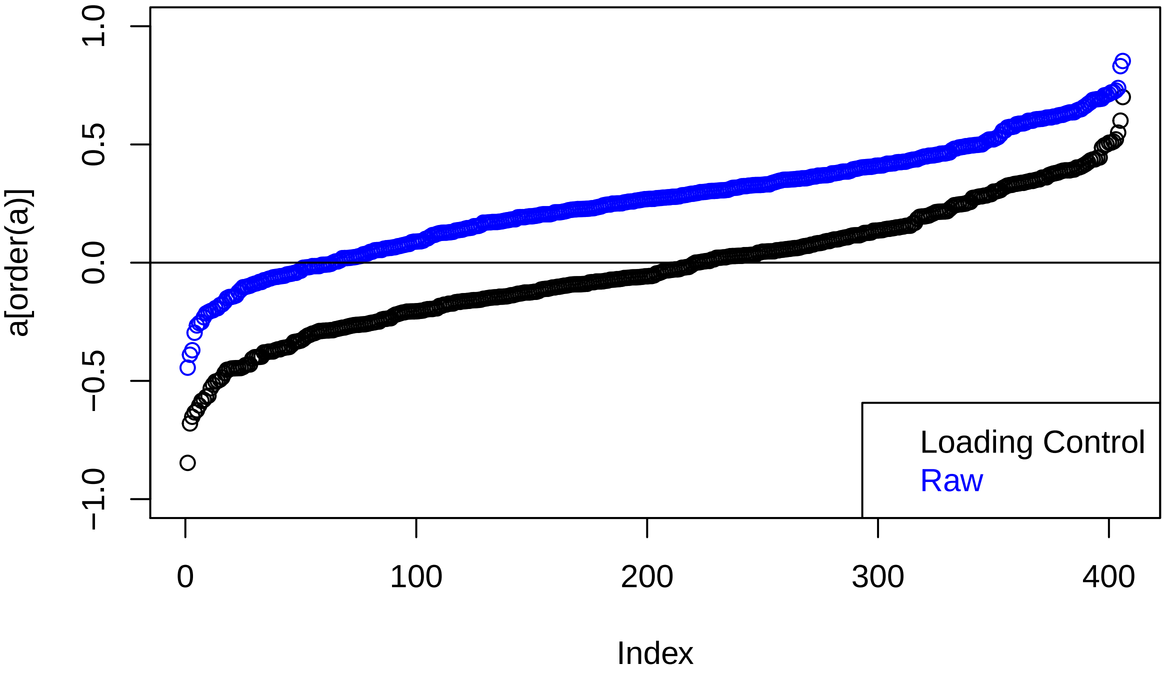

Loading control normalised
Expression log−ratio (this sample vs others)
2
1
0
−1
−2
−10
−5
0
Average log−expression


Raw
Loading control normalised
Expression log−ratio (this sample vs others)
Average log−expression
4
2
0
−2
−4
−6
−10
−5
0
10
5
0
−5
−10
−12
−10
−8
−6
−4
−2
0
Raw
Loading control normalised
0
−2
−4
−6
−8
−10
−5
0
0
−5
−10
−15
−6
−5
−4
−3
−2
−1
0
Raw
Loading control normalised
2
0
−2
−4
−6
−10
−5
0
0
−5
−10
−15
−6
−4
−2
0
Raw
Loading control normalised
8
6
4
2
0
−10
−5
0
15
10
5
0
−6
−5
−4
−3
−2
−1
0
1



Spearman rank correlation plot. We expect a largely equal psotive and negative correlation as discussed above. Blue points are the correlations between proteins for raw RFI values. Black are correlations between proteins for loading control normalised values.

```
rank_lc <- lc %>%
  group_by(Type,AB) %>%
  mutate(Rank = rank(Intensity)) %>%
  select(-Intensity) %>%
  spread(key = Type, value = Rank)

spearmanPlot(rank_lc, "LC", "Loading Control")
```



Again, there is a more even distribution of correlation above and below 0 after loading control normalisation.

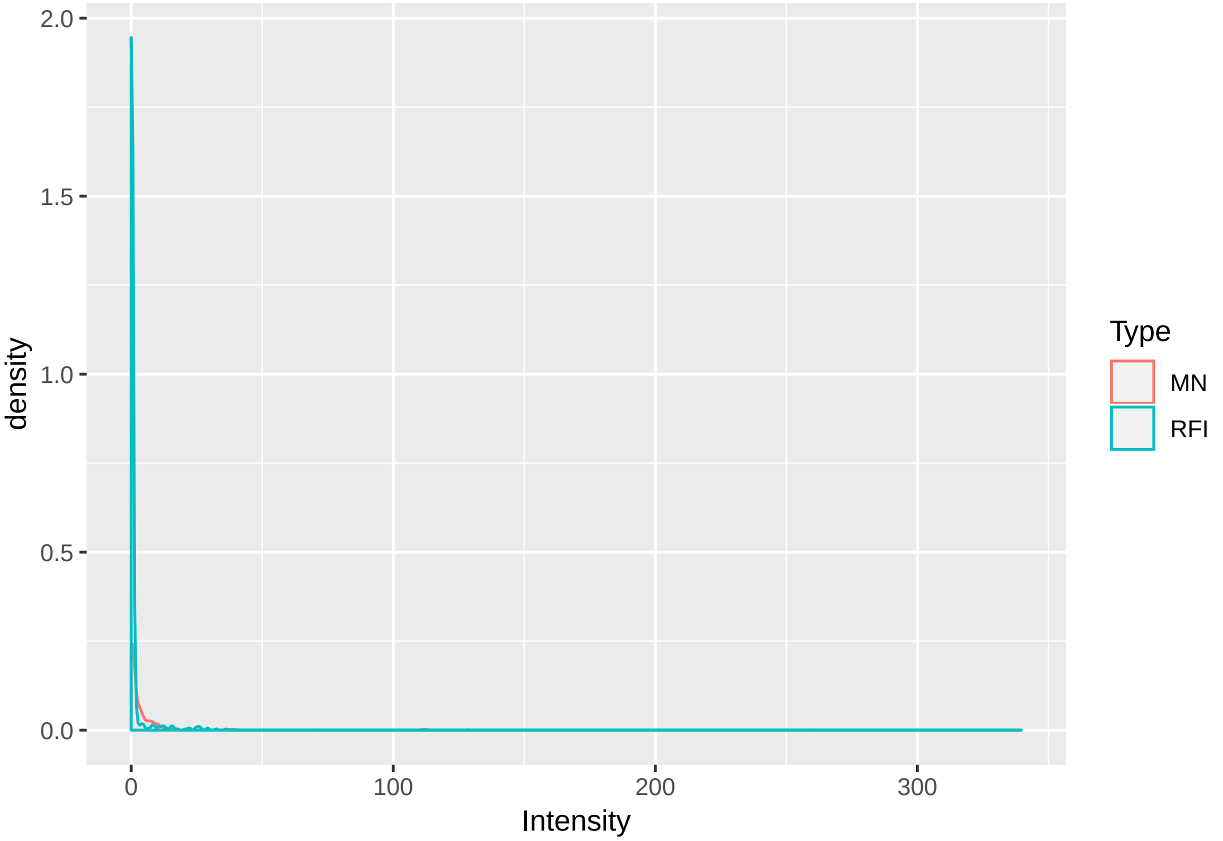
Median normalisation/Global median centering

Calculate median normalised values.

```
mn <- raw_tidy %>%
  group_by(Sample) %>%
  mutate(med = median(RFI)) %>%
  ungroup() %>%
  mutate(MN = RFI/med) %>%
  select(-med)

mn <- mn %>%
  gather(RFI, MN, key = "Type", value = "Intensity")
```

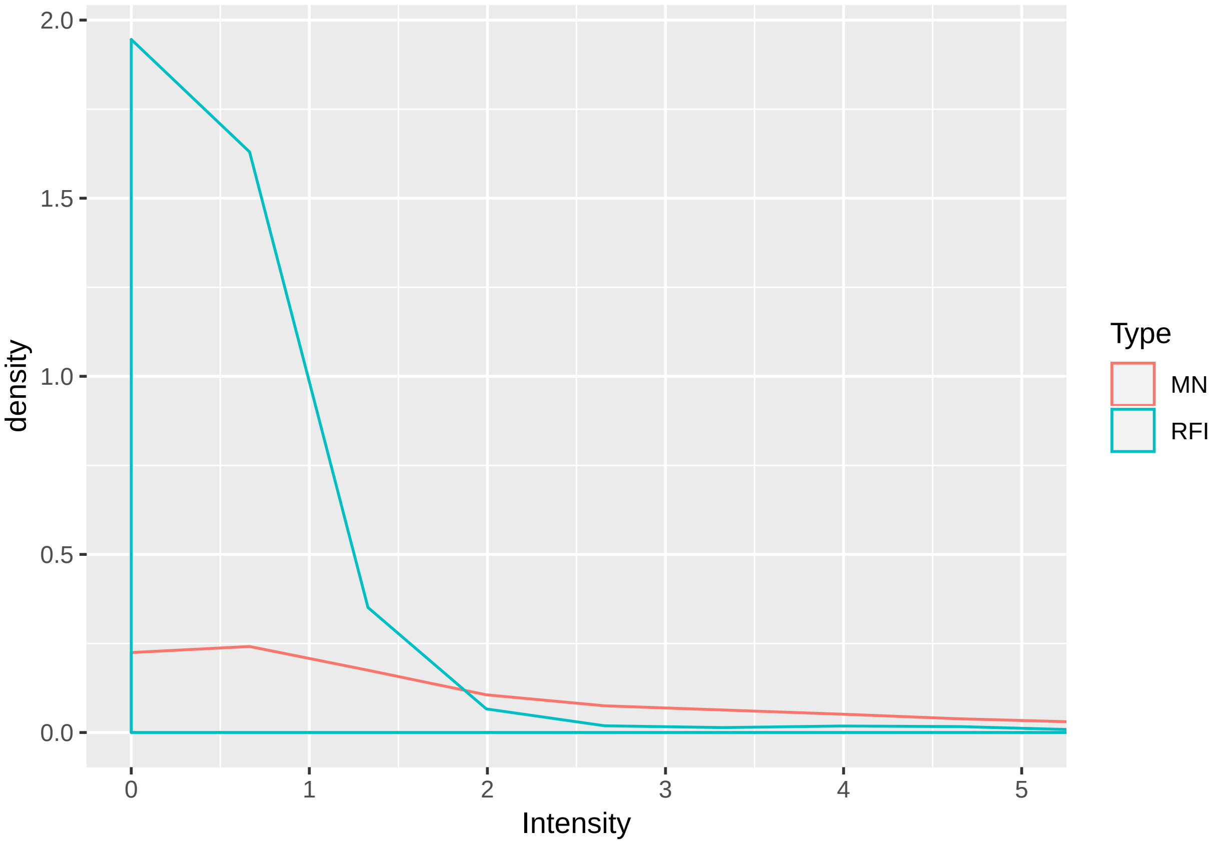
Plot distribution of all intensity values for raw and median normalised data:

```
ggplot(mn, aes(x = Intensity, colour = Type)) +
  geom_density()
```



Zoom in on the low intensity values to better visualise the data:

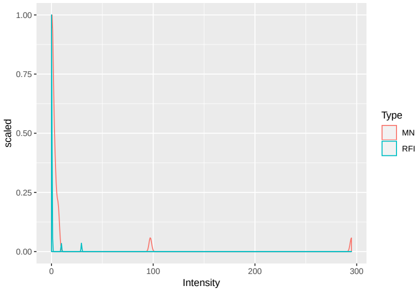
```
ggplot(mn, aes(x = Intensity, colour = Type)) +
  geom_density() +
  coord_cartesian(xlim = c(0,5))
```

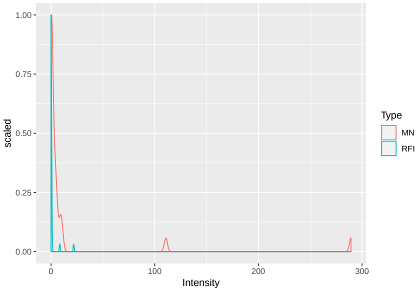


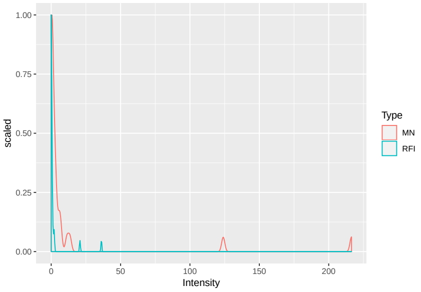
Median normalisation increases intensity values, even more significantly than HK normalisation.

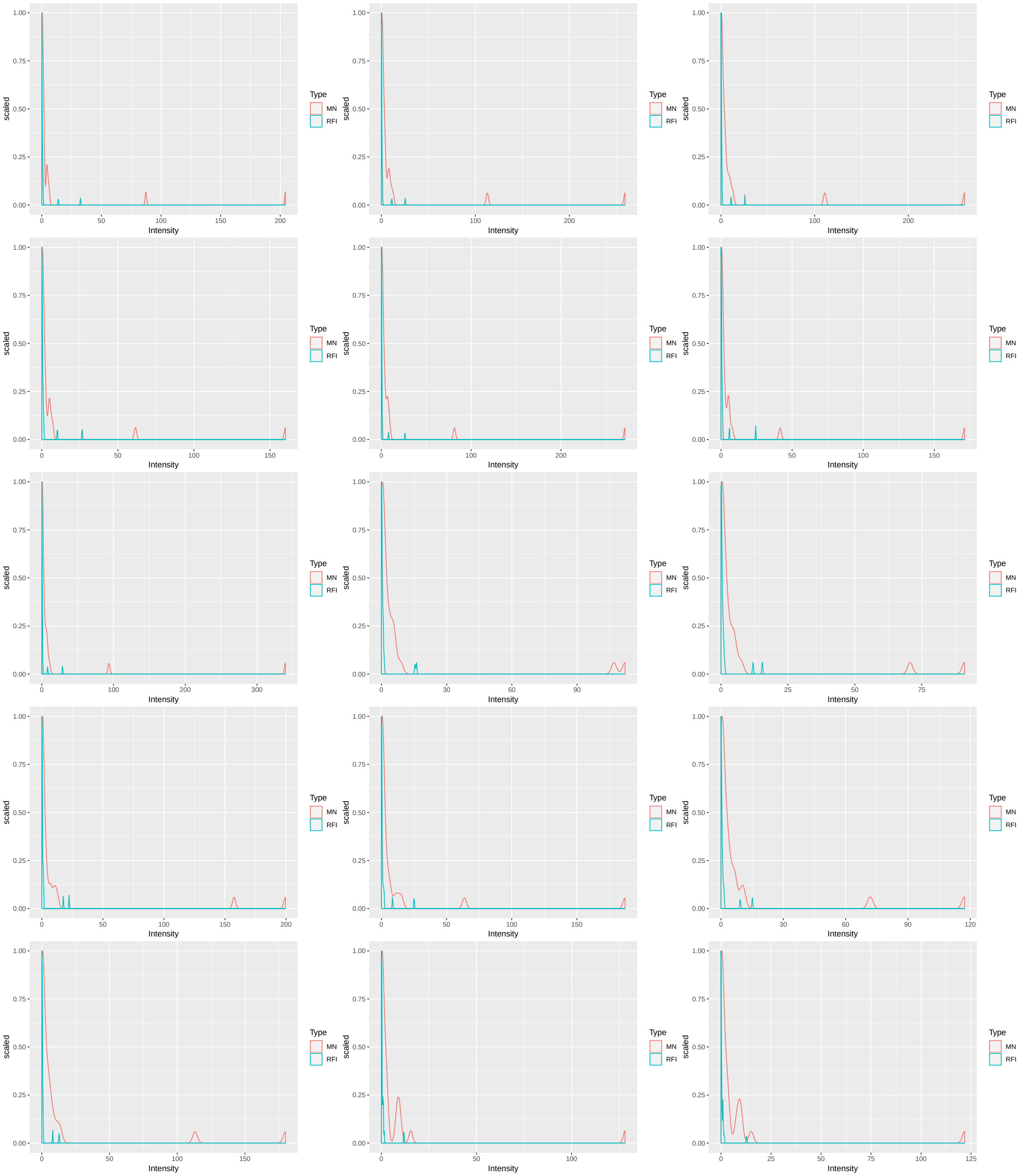
Plot the distribution of intensity values for each sample:

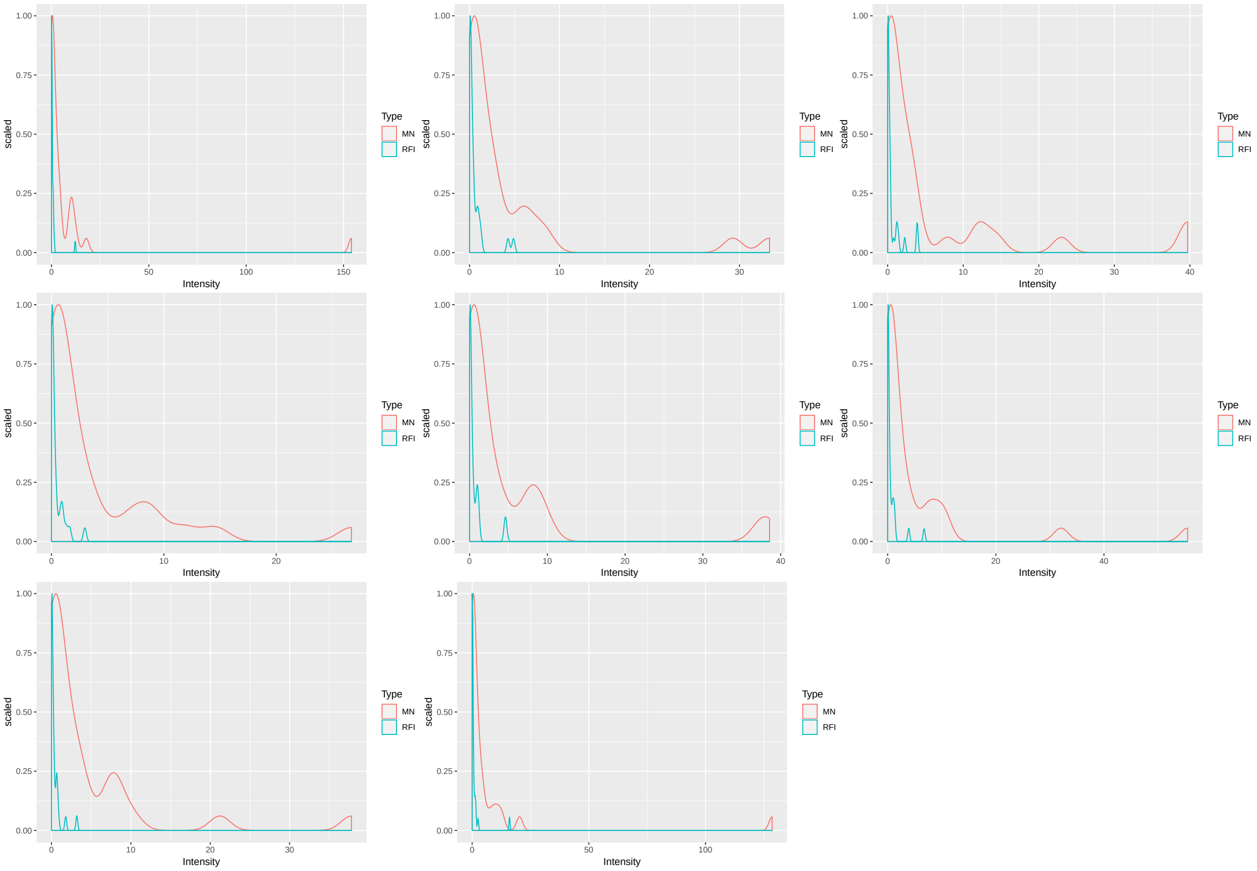
```
samples <- unique(mn$Sample)
for (i in 1:26){
print(
  mn %>%
  filter(Sample == samples[i]) %>%
ggplot(aes(x = Intensity, colour = Type, y = ..scaled..)) +
  geom_density()
)
}
```





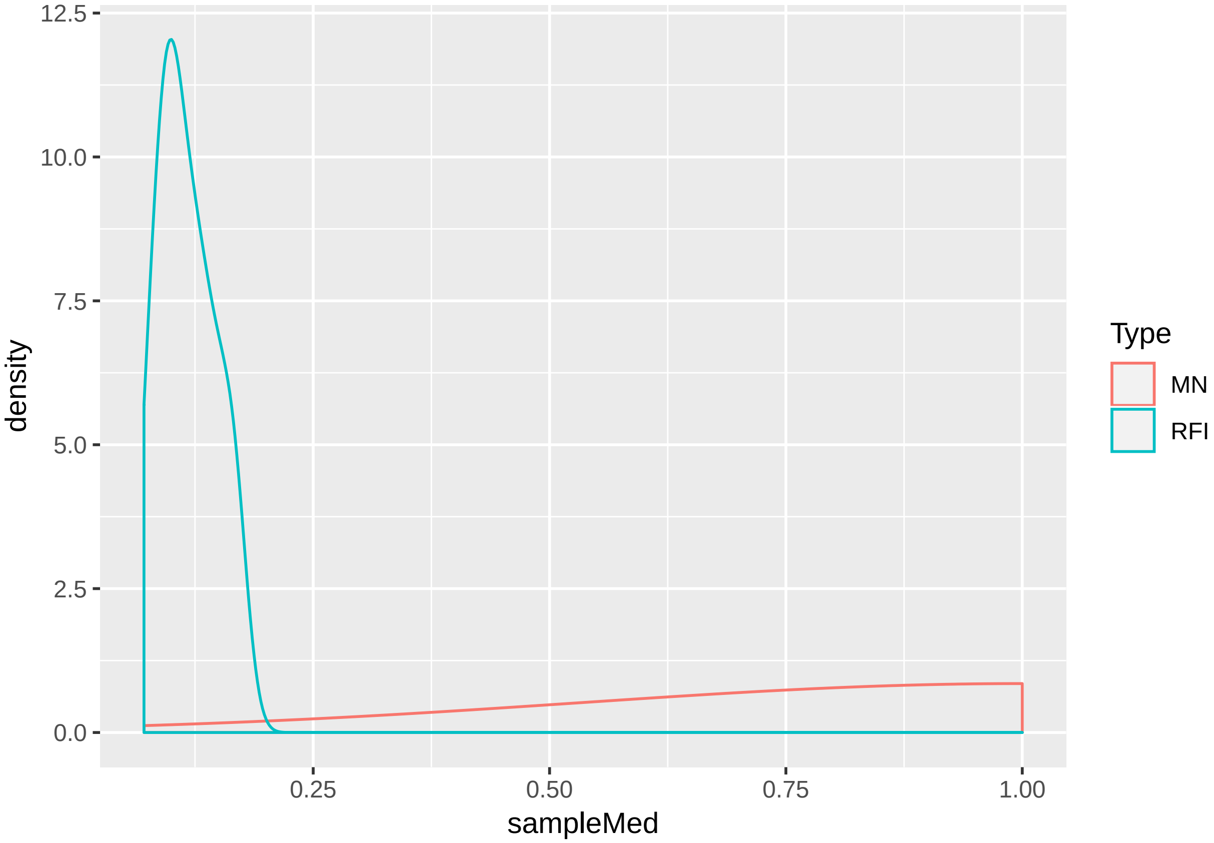



scaled
1.00
0.75
0.50
0.25
0.00
Intensity
Type
MN
RFI



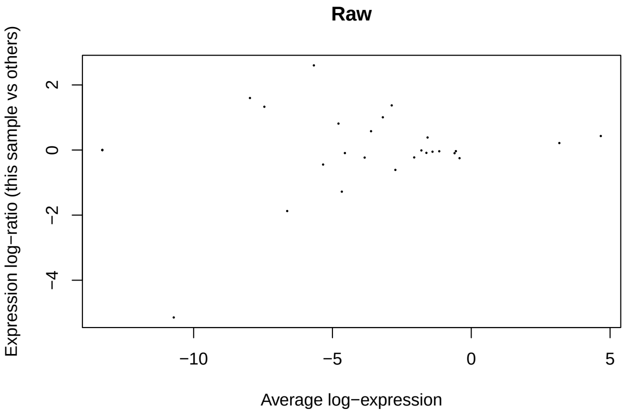
Distribution of sample medians for raw and median normalised intensities:

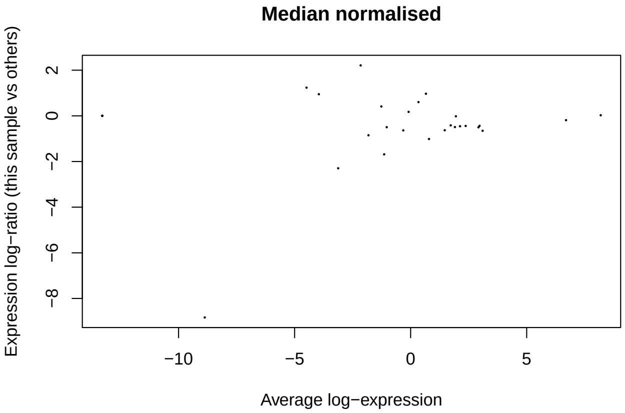
```r
mn %>%
  group_by(Sample, Type) %>%
  summarise(sampleMed = median(Intensity, na.rm = TRUE)) %>%
  ggplot(aes(x = sampleMed, colour = Type)) +
  geom_density()
```

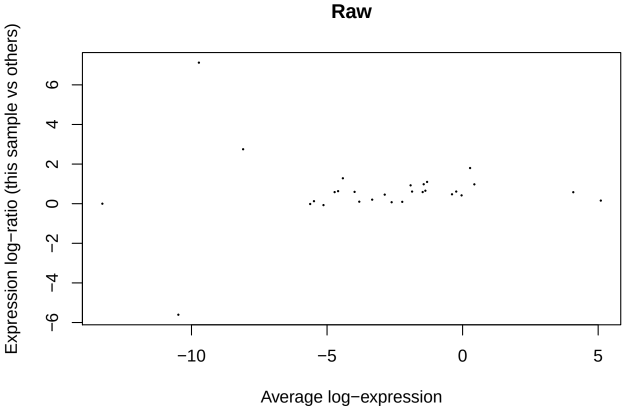


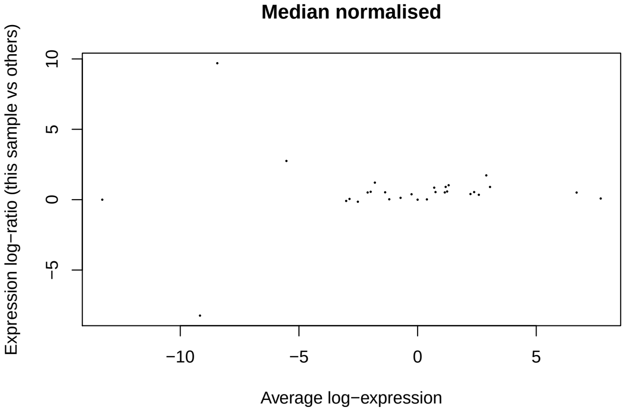
Plot MA graphs for raw and median normalised intensities:

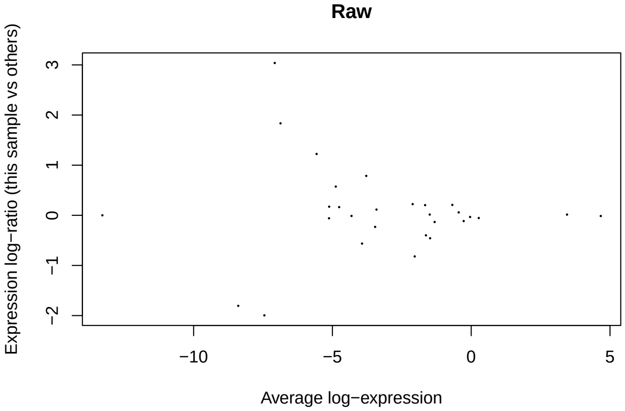
```
mn_matrix <- acast(
  mn %>%
  filter(Type == "MN") %>%
  mutate(log2 = log(Intensity + 0.0001, base = 2)) %>%
  select(-c(Type, Intensity)),

  AB~Sample,
  value.var = "log2"
)

for (i in 1:13){
plotMA(raw_matrix[,(2*i-1):(2*i)], main = "Raw")
plotMA(mn_matrix[,(2*i-1):(2*i)], main = "Median normalised")
}
```

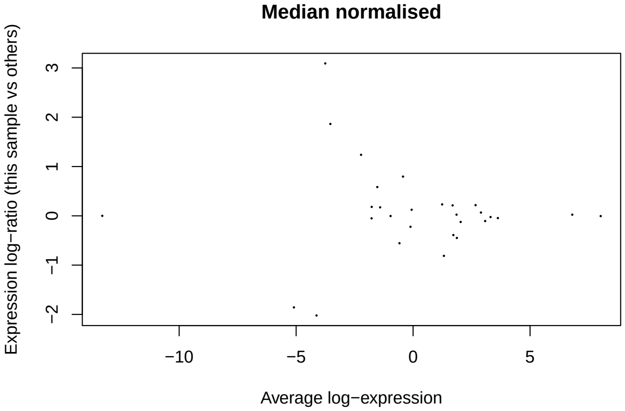

Raw

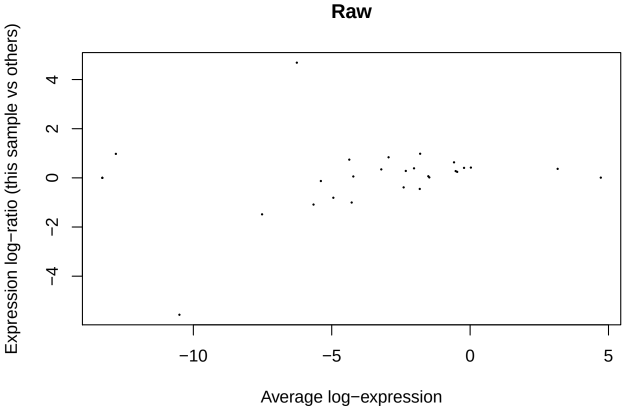

Median normalised

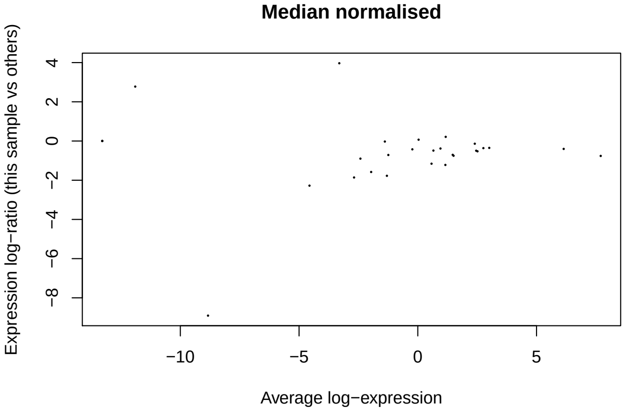

Raw

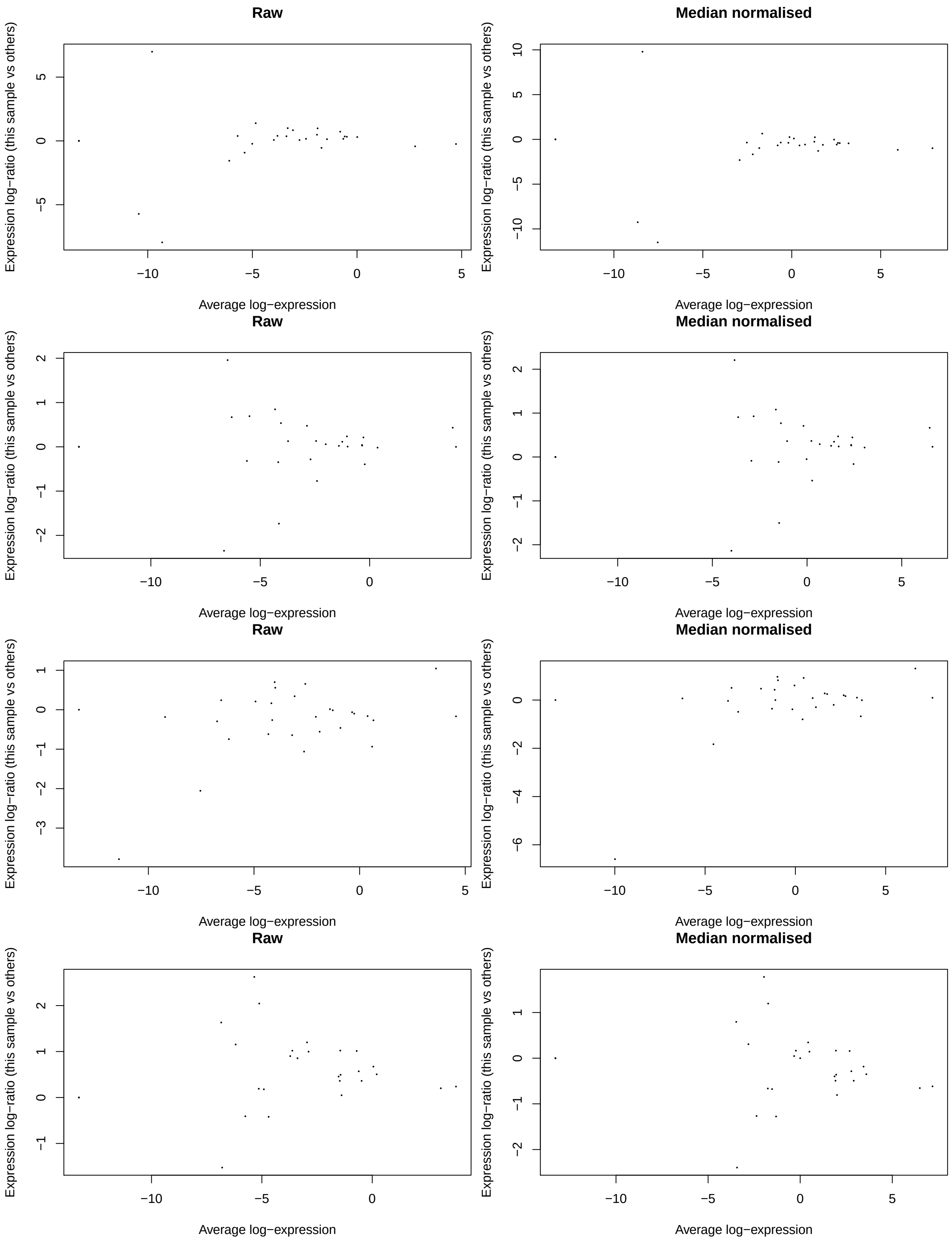

Median normalised

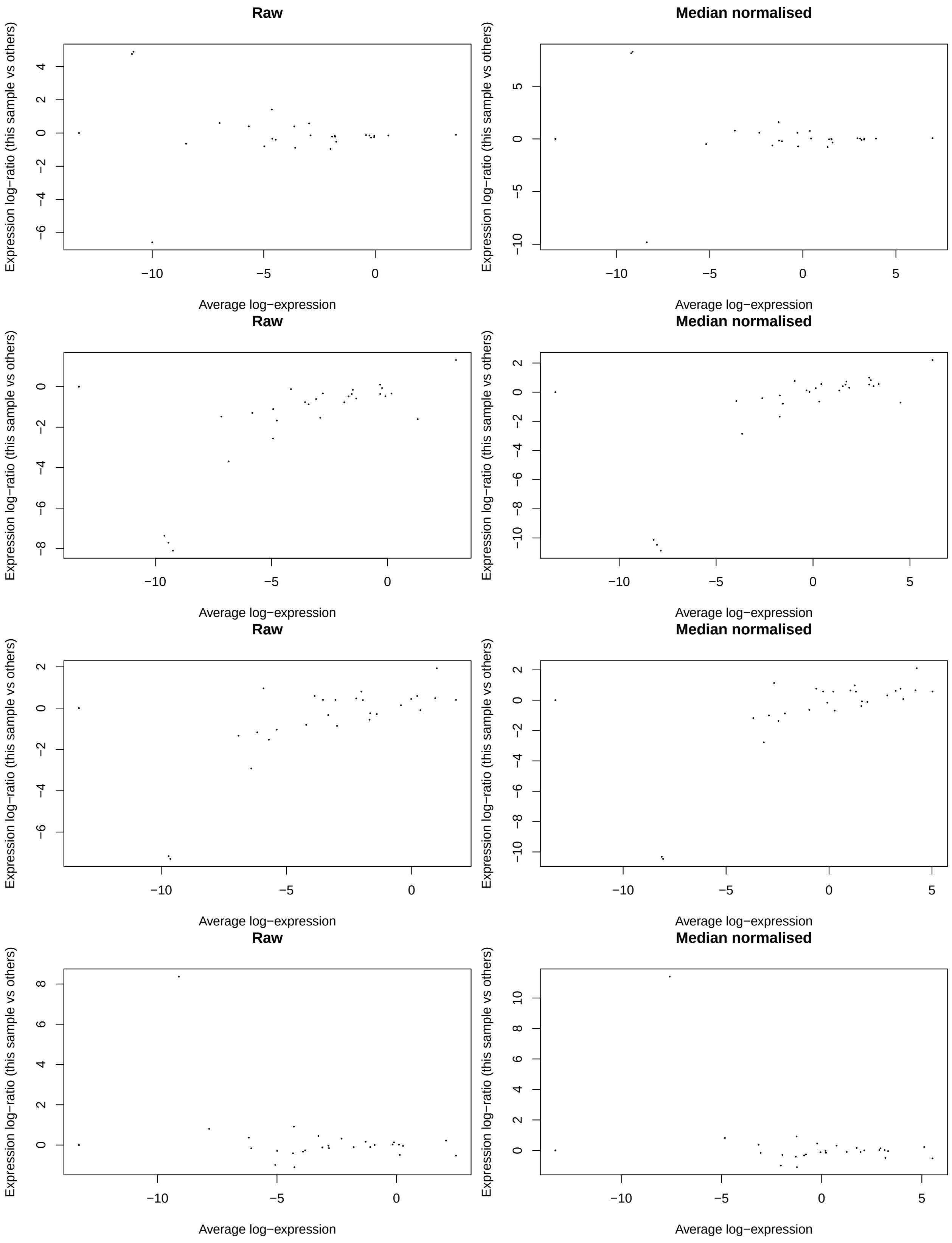

Raw

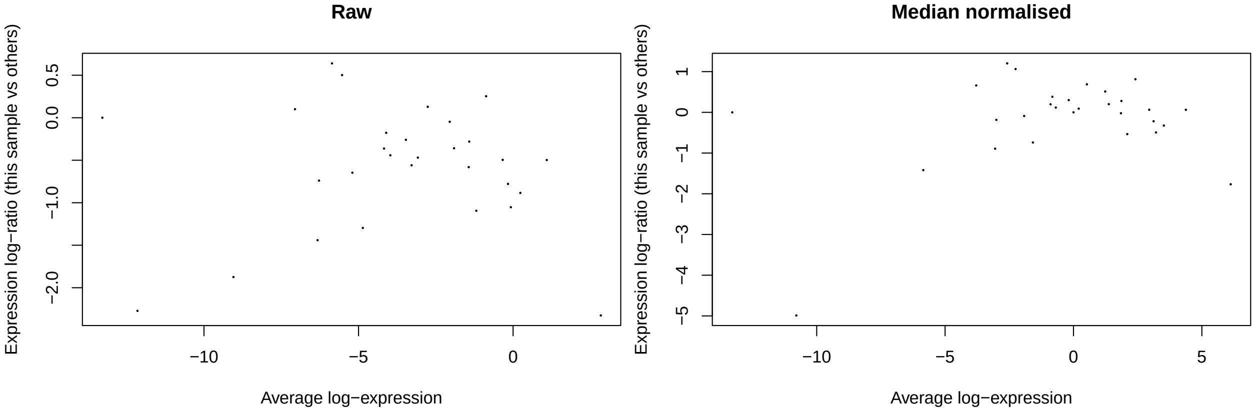

Median normalised


Raw


Median normalised

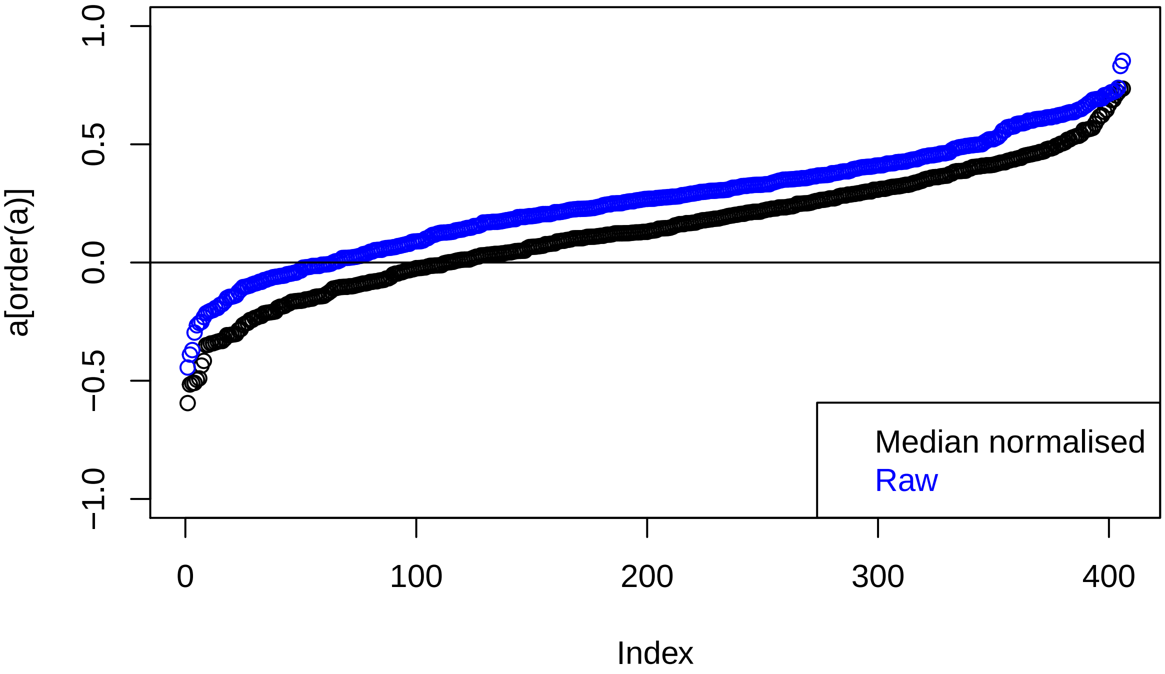

Raw
Median normalised
Expression log−ratio (this sample vs others)
Average log−expression
−10
−5
0
5
10
2
1
−1
−2
−3
−4
−6


Raw
Median normalised
Expression log−ratio (this sample vs others)
Average log−expression
4
2
0
−2
−4
−6
−10
−5
0
5
−10
−5
0
5
Raw
Median normalised
0
−2
−4
−6
−8
2
0
−2
−4
−6
−8
−10
−10
−5
0
5
Raw
Median normalised
2
0
−2
−4
−6
2
0
−2
−4
−6
−8
−10
−10
−5
0
5
Raw
Median normalised
8
6
4
2
0
10
8
6
4
2
0
−10
−5
0
5



Spearman rank correlation plot. We expect a largely equal psotive and negative correlation as discussed above. Blue points are the correlations between proteins for raw RFI values. Black are correlations between proteins for median normalised values.

```r
rank_mn <- mn %>%
  group_by(Type,AB) %>%
  mutate(Rank = rank(Intensity)) %>%
  select(-Intensity) %>%
  spread(key = Type, value = Rank)

spearmanPlot(rank_mn, "MN", "Median normalised")
```



The difference between the distribution of correlations is less pronouced for this normalisation method.

References

Liu, Wenbin, Zhenlin Ju, Yiling Lu, Gordon B. Mills, and Rehan Akbani. 2014. "A Comprehensive Comparison of Normalization Methods for Loading Control and Variance Stabilization of Reverse-Phase

Protein Array Data , A Comprehensive Comparison of Normalization Methods for Loading Control and Variance Stabilization of Reverse-Phase Protein Array Data." *Cancer Informatics* 13 (January): CIN.S13329. doi:10.4137/CIN.S13329.

Wachter, Astrid, Stephan Bernhardt, Tim Beissbarth, and Ulrike Korf. 2015. "Analysis of Reverse Phase Protein Array Data: From Experimental Design Towards Targeted Biomarker Discovery." *Microarrays* 4 (4): 520–39. doi:10.3390/microarrays4040520.