Final Report: Proteome-wide Screen for RNA-dependent Proteins
non-synchronized A549 cells

Anastasia Möller, Johannes Schadt, Sylviane Verschaeve, Tine Limberg

17.07.2023

Contents

Loading the data:

1. Introduction

1.1. Importance of RNA-binding proteins

1.2. Experimental Setup

2. Methods

2.1. Data cleanup

2.2. Normalization methods

2.3. Reproducibility and Batch Effect

2.4. Scaling and Reduction of Dataset

2.5. Our limitations with Gaussian fit

description of problem with local peaks and inaccuracy of overlap

Alternatively, we …

2.6. Data description via Parameters

The control sample and the RNase sample for each protein will be compared via the

Via the parameters the differences between the RNase and the Control are depicted manually via the parameters.

2.6.1. Parameter 1: Significant change of protein amount under global peak

2.6.2. Parameter 2: Significant change of protein amount under local peaks

2.6.3. Parameter 3: Significant fraction-shift of global peak

2.6.4. Parameter 4: Significant difference in position of shoulderregions

2.6.5. Precipitated proteins

2.7. K-means clustering

2.8. Regression analysis

3. Results

3.1. Cleaned Dataset

3.2. RNA-dependent Proteins

3.3. Comparison of the normalization methods

3.3. K-means clustering

3.4. Regression analysis

3.5. Comparison with Database

4. Discussion

5. Literature

1.1. Check for missing values

1.2. Check data format

1.3. Deleting rows with only zeros

-> da die Summe der Zeileneinträge keines Proteins 0 entspricht, wurde ein Dataframe aus False erstellt. Einträge ausschließlich False, werden durch die sum Funktion als 0 aufaddiert.

1.4. Rearranging of Data

1.4.1. Reordering columns

1.4.2. Separate Ctrl and RNase

2. Reproducibility

Here we test whether the replicates are similar to each other. This would mean, that the experiment is reproducible, thus the data is reliable. Proteins that do not satisfy this condition will be removed from the dataset and will not be analysed.

2.1 Pearson Correlation To facilitate the calculation of the correlation between each replicate, we design 6 separate data frames, one for each replicate

Here we calculate the correlation between the replicates and put them together in one data frame (ctrl.cor and rnase.cor) (?)

Now we eliminate proteins which have NA-correlations (this happens when they contain replicates with only 0s). We then create new separate data frames for each replicate.

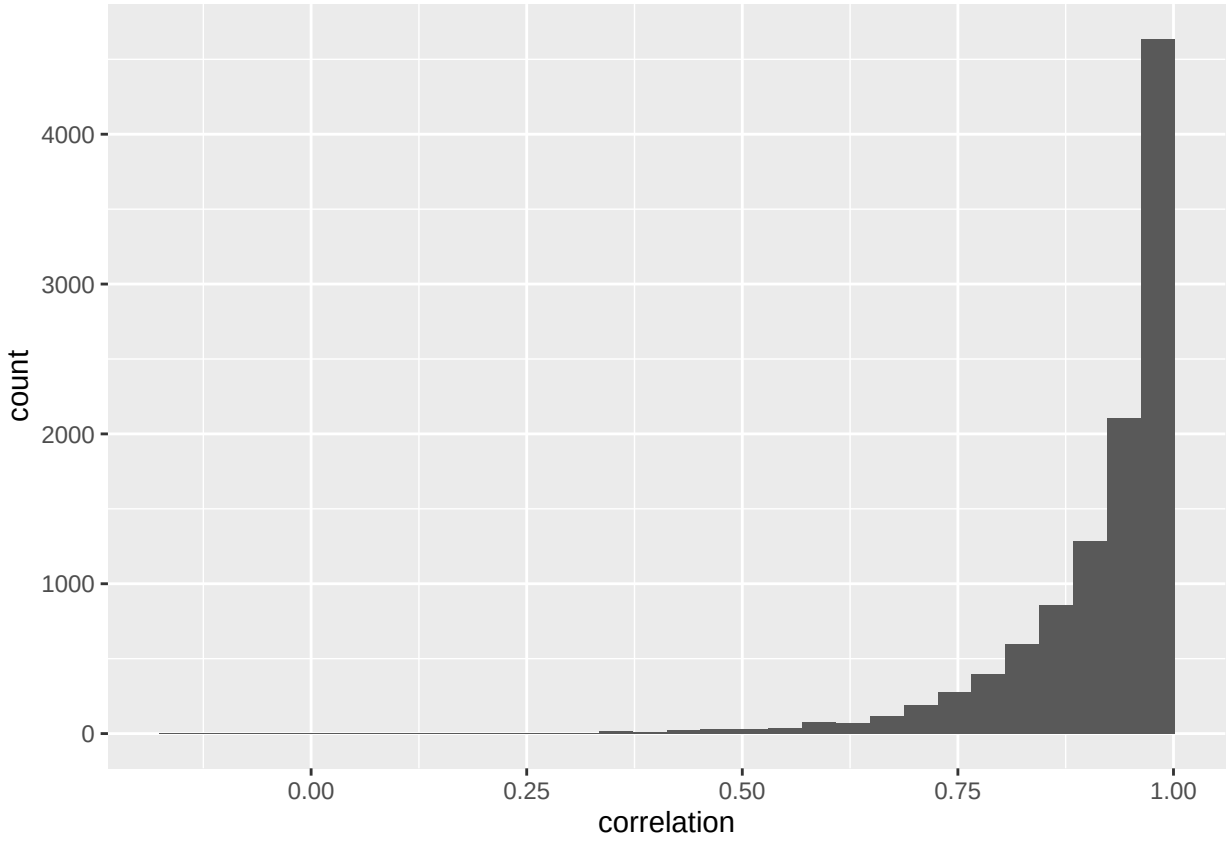
Now we calculate the correlation of the replicates. This time the proteins that containes replicates with only 0 are eliminated, so there should be no NAs anymore.

The following plot shows us the general distribution of correlation.

In total we look at 3*(3680-83) correlations This has to be taken into account, when looking at the graphs. It is import to figure out if the 3 cor are for one protein or for 3 different ones.

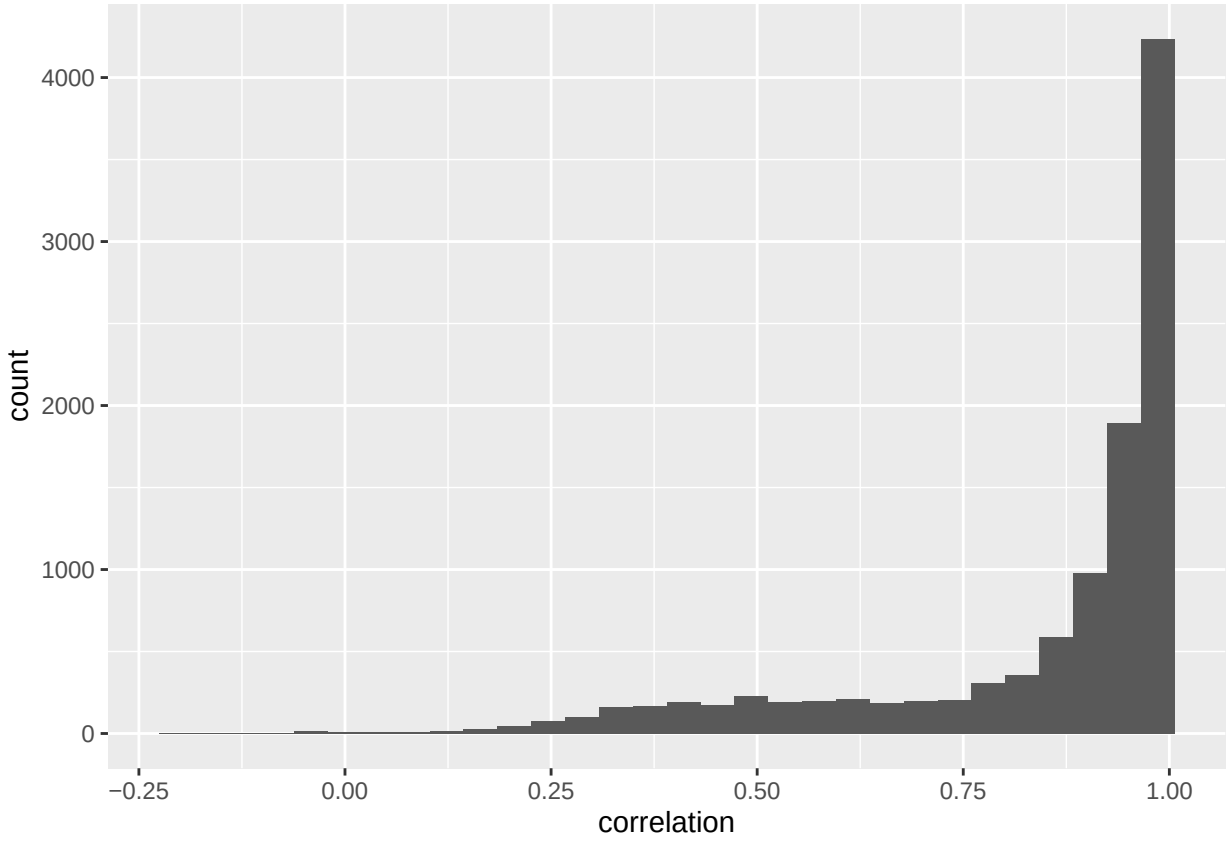
```
## Warning: Paket 'ggplot2' wurde unter R Version 4.2.3 erstellt
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We do the same for the RNase group:

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Now we select the proteins which have correlations beneath 0.9. Those are not reproducible, thus the data is not safe enough to be used further.

First we determine the proteins that only have correlations under 0.9.

Now we eliminate the proteins that only have correlations under 0.9.

Other proteins are a bit trickier. Some proteins have two replicates similar to each other (correlation < 0.9) and a third one that completely differs. These Proteins have one very high and two smaller correlations. The different replicate is often the third one (mabye batch effect). To avoid loosing too many proteins and still to still have safe data, we try to ignore the bad replicates. For this we first set them to NA: After the normalization-set we can ignore them.

We now have 3074 Proteins left. They are stored in new variables:

We now have clean data, with proteins that have reproducible data we can use for further analysis.

3. Scaled and Reduced Dataset

For the normalization each replicate has to be separated, therefore we design 6 separate dataframes.

3.1. Mean Value Method

3.1.1. Normalization We perform the mean-value-method (mvm) on each replicate, both control and RNase:

3.1.2. Reduction To reduce we take the mean value between each replicate. Here we must consider the NA-values of non-reproducible replicates.

3.1.3. Scaling To test whether we have “lost” our scaling during the merge, and find out whether scaling back to 100 is necessary, we scale the control to 100 and compare it with the original control.

-> scaling back to 100 is necessary

Because scaling back to 100 is necessary, we do it for the RNase too:

Now we have normalized our data using the mean-value-method, and scaled it to 100. The two variables that will be used later on either contain the normalized (mvm) and scaled data of the control: **ctrl.mvm** or the normalized (mvm) and scaled data of the rnase: **rnase.mvm**