# Little Handbook of RPPA Analysis in R Using MIRACLE Data

This document describes how you can analyse the RPPA (reverse phase protein array) data using the methods implemented in the R package “miracle”. This R package complements the MIRACLE web-application which allows you to manage RPPA data, as well as to assign slide design / layout information.

# Installation (not necessary when using R-Studio Server):

1. You need to install the R scripts for RPPA analysis:

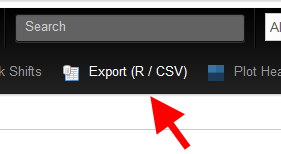
install.packages(file.choose(), type="source", repos=NULL)

S:\CI\ci-mo\Mollenhauer Group\RPPA\R scripts\rppa.zip

1. In R you can initialize any library with the command  
   library(packageName), e.g. library(miracle)
2. You can install additional packages that are required through  
   install.packages(“packageName”), you’ll need
   1. ggplot2
   2. reshape
   3. plyr
   4. manipulate
   5. gridExtra
   6. grid
3. In order to compute protein concentration estimate you can either use the serial dilution curve algorithm (code included) or the SuperCurve package. The installation of the SuperCurve packages is a little more complicated. See <http://bioinformatics.mdanderson.org/Software/supercurve/>

# Tutorial (example of a full-scale analysis):

This tutorial assumes that you have assigned layout information (e.g. information about the sample, cellline, lysis buffer, …) already, as well as processed the raw data. First you need to import the dataset from MIRACLE. Therefore open your slide result in MIRACLE and click on “export to R”:

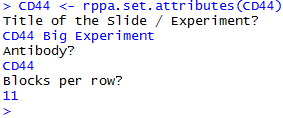


In the next step copy the command given there:



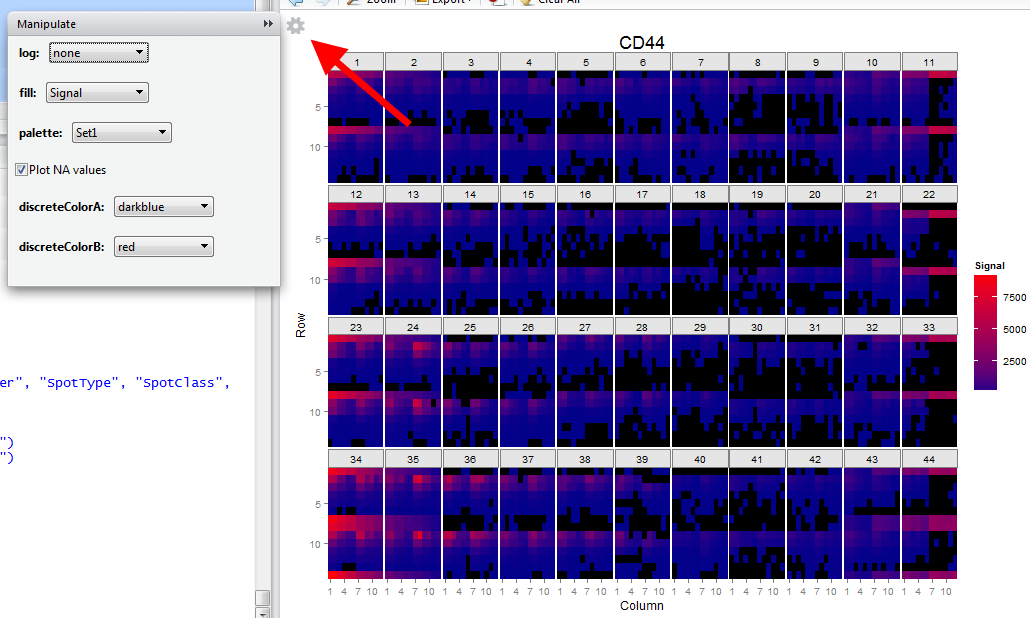
We assume in this tutorial that our dataset is called CD44 and contains layout information about CellLine, LysisBuffer, Inducer and Treatment.

The rppa.load function already does some work for us, such as the filtering of bad signals. It also applies row and column shifts that have already been defined in MIRACLE. However, you can switch these functions off in rppa.load and do this manually. What is left to do, is to assign some attributes to the dataset:



Now we would like to look at the raw data to see if everything looks fine. Therefore we can plot a heatmap:





You can easily change the parameters for plotting by clicking on the little cog-wheel on the top-left corner.

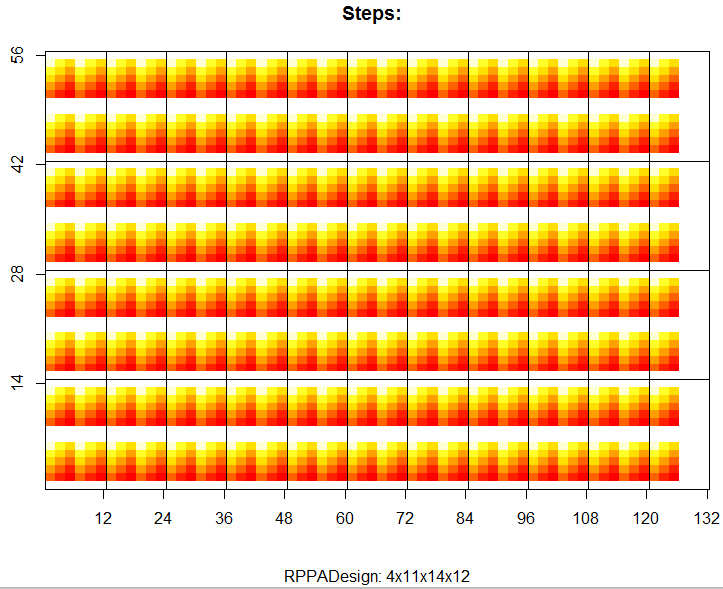
Now that we are happy with our data we would like to estimate the protein concentration of our samples. Since the reaction of the antibody is the same for the whole slide we can use the joint sample information of all dilutions and samples to model the dose-response-curve. This can be done using one of two available methods, the SuperCurve or the serial dilution curve algorithm.

The SuperCurve algorithm is widely used and highly accepted in the science community, but it has to be installed separately (Hu, J. *et al.* Non-parametric quantification of protein lysate arrays. *Bioinformatics (Oxford, England)* **23**, 1986–94 (2007).)

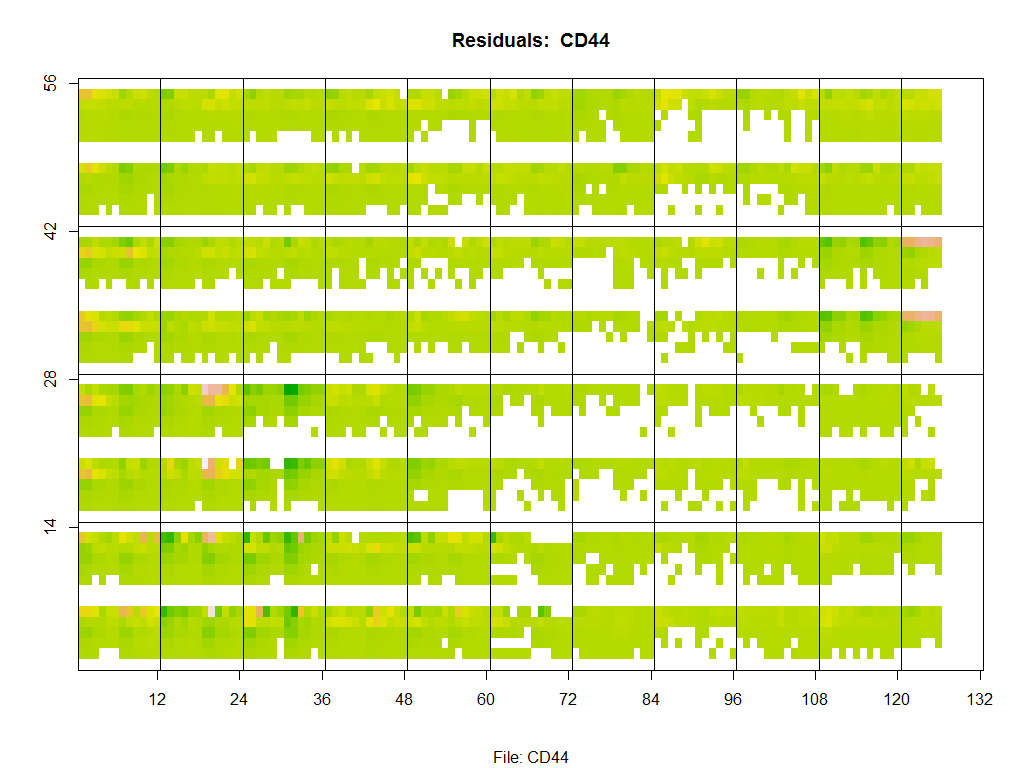
One should make sure that the property “SpotType” has been assigned to the data before using either of algorithms, because otherwise control spots are made part of the model fitting, resulting in a very poor fit.



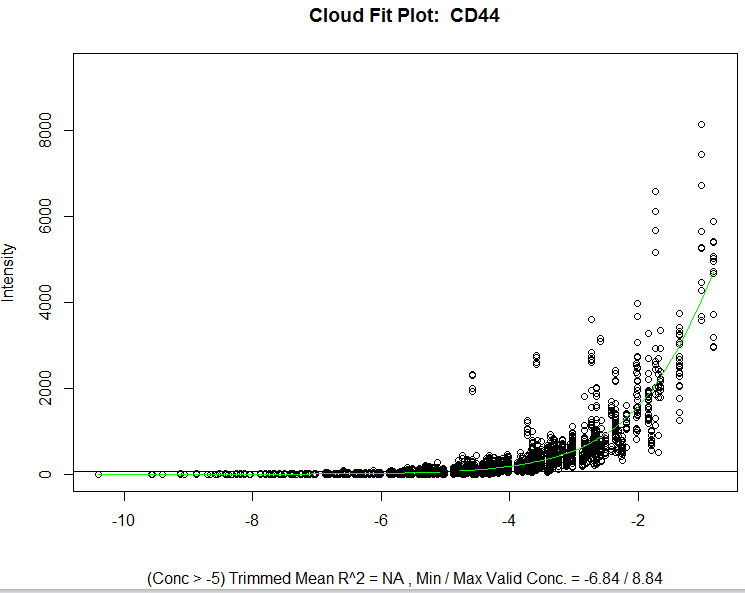
Before the SuperCurve algorithm fits our model, we are presented with the following plot:



This plot shows us how the different dilutions are anticipated. In the default setting, our in-house pattern of depositions in rows and dilutions in columns is used to calculate the actual dilution of the sample. The sample information is then used to group the different dilution series. This behaviour can be changed using the “grouping” argument of the method.

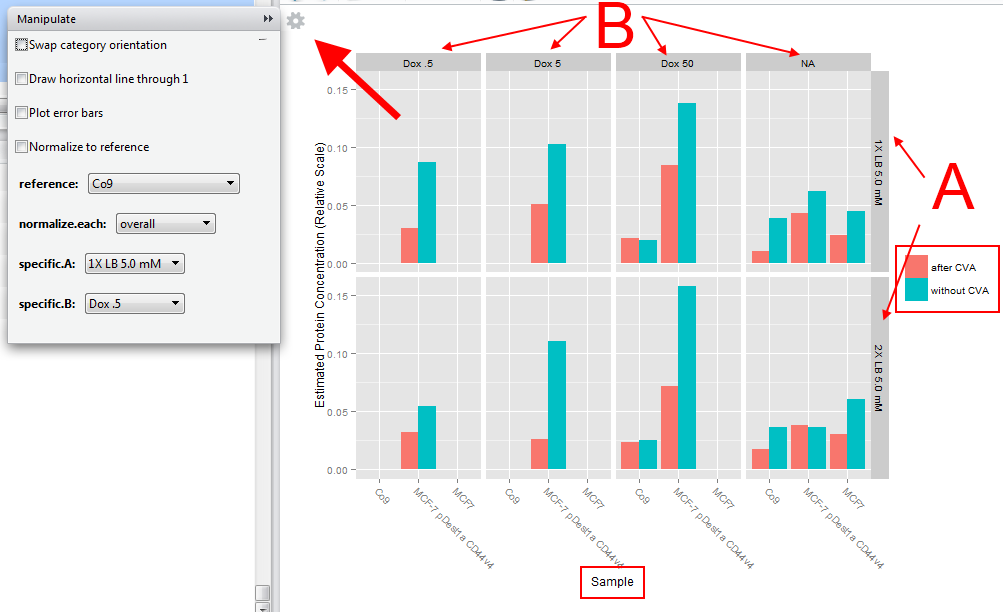


This plot shows us the residuals found in the fitting of the curve. This plot is ideal for spotting outliers that indicate a problem with the staining of the slide or shifts that were not recognized previously.

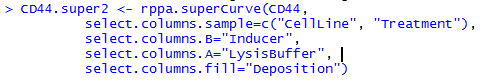


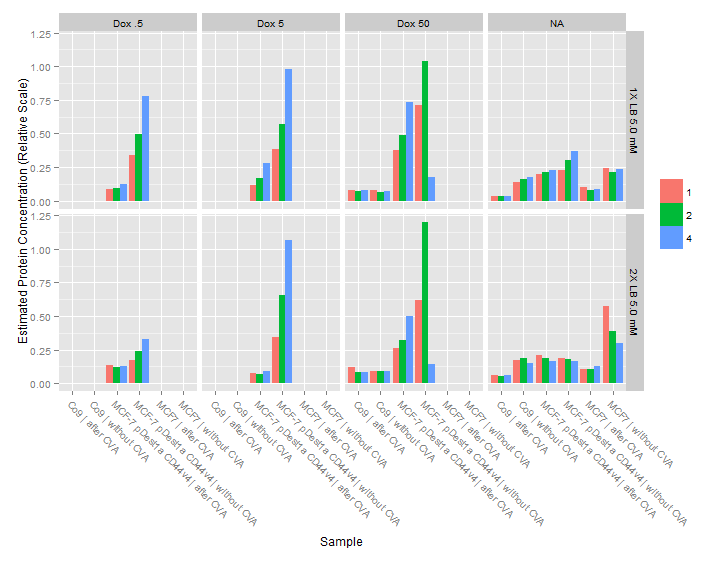
This plot shows us how the response curve is fit onto the data. When the algorithm is done (you might see some warnings, because not all of the dilution series can be fitted correctly), you obtain a new object (here called CD44.super) which contains the protein concentration estimates for all samples generated from the model. You can plot your results using





This plot gives us some insight into experimental categories. You will notice that the sample information is found on the x-axis, but information is also grouped into categories A and B, as well as the colour fill of the bars. You can influence how information is grouped when calling the rppa.superCurve method using all the available layout properties you defined previously in MIRACLE. Presume now we want to change our graph by merging CellLine and Treatment as “Sample” property, so that the color fill property becomes free. We set color fill as property “Deposition” to see how well the different depositions agree. You can choose all categories freely by using either Strings, e.g. “CellLine” or vectors, e.g. c(“CellLine”, “Treatment”). It is also possible to normalize the data either by sample, category A and/or B or by a specific sample using the manipulate window.

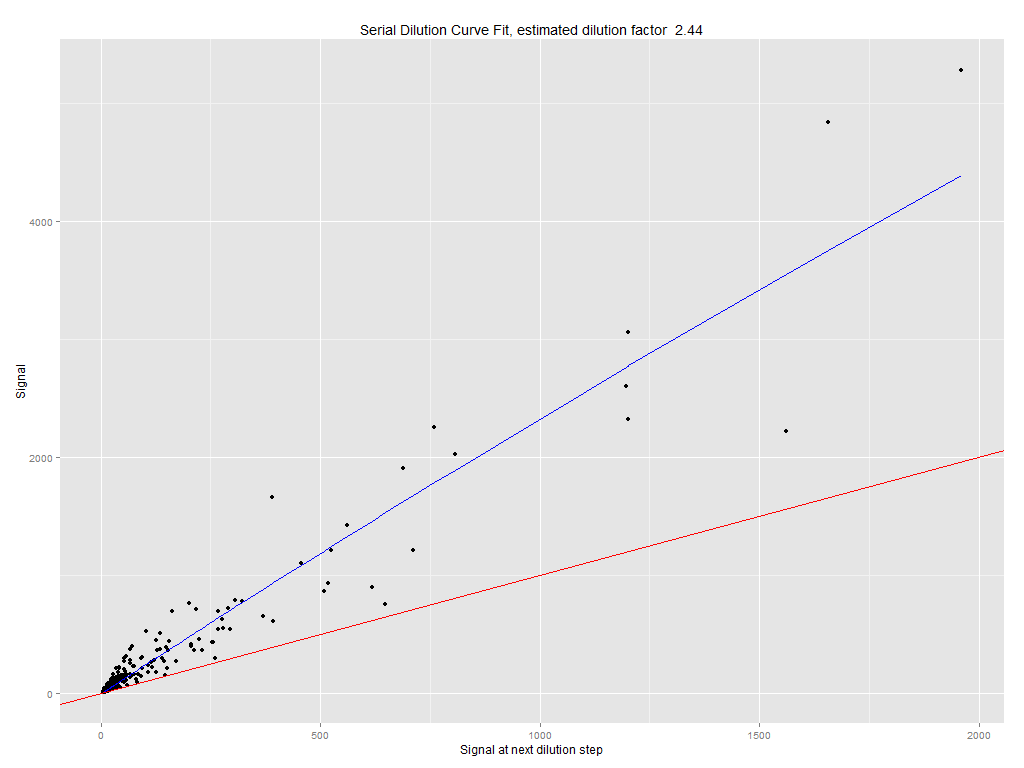




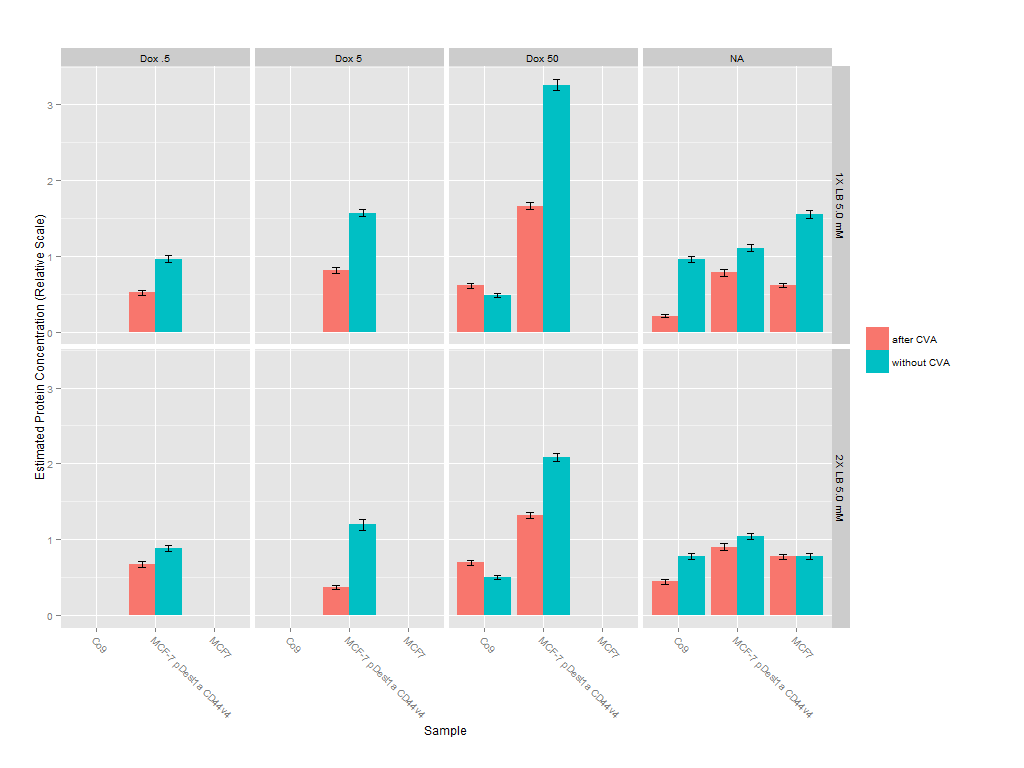
Since we included the deposition count in the dilution for the model fit you would expect all depositions to be on more or less the same level. Samples where this is not the case should be treated as suspicious.

The serial dilution curve algorithm (Zhang, L. *et al.* Serial dilution curve: a new method for analysis of reverse phase protein array data. *Bioinformatics (Oxford, England)* **25**, 650–4 (2009).) is a fairly new method for modelling the antibody binding on RPPA chips. It creates a Sips model using a serial dilution curve. In a serial dilution plot the signal of one dilution step is compared to the signal at the next dilution step. The serial dilution curve it then a fitted curve in this plot and allows for the Sips model to be created from properties found in this curve. It’s implementation is included in the miracle RPPA package and thus it can be used directly. It is very fast compared to the SuperCurve algorithm, but does currently not display residuals. The usage is very similar to rppa.superCurve()





This plot shows the aforementioned serial dilution curve and yields an estimated dilution factor. The intersection of the blue curve with the red line yields the necessary parameters to produce a reasonable model for estimating the sample concentrations. In fact, the result is the same as before where we applied the SuperCurve algorithm to the same data:



## Output of the results

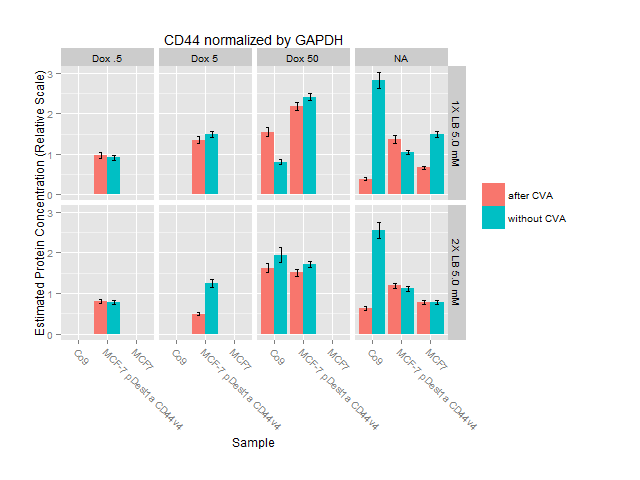
One of the standard features of R is the possibility to export data frames as CSV files. In order to take your results to other programs, e.g. Excel (if you really must), you can just export the result data. In the examples above the result data has been stored in CD44.sdc and CD44.super. To export the data use:



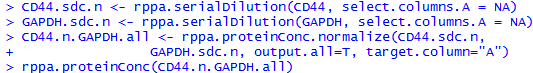
## Normalizing slides

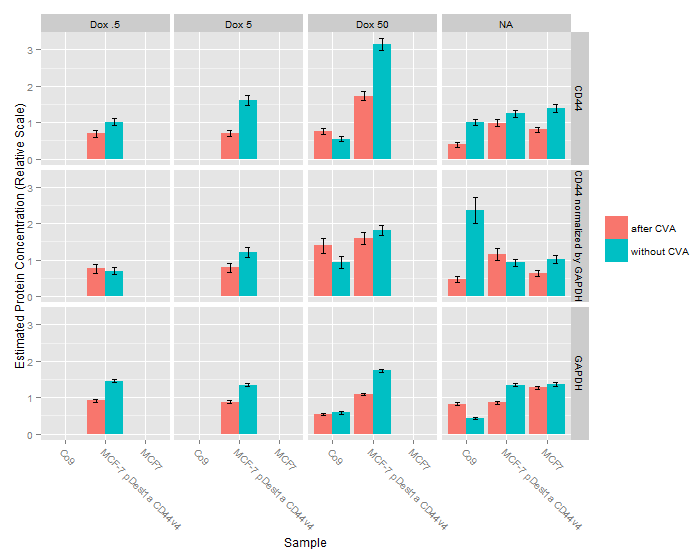
Usually you would like to normalize your RPPA data using a second slide where you use an antibody to stain a housekeeping protein, or a protein where you would not expect any variability except for the amount protein found in the sample. Presume we want to normalize the CD44 stained slide with a GAPDH staining of the same slide. This can be achieved as follows:



This results in the following plot: 

Wouldn’t it be nice if we could compare CD44, GAPDH and the normalized signal side by side? This can easily be done if we leave one category variable free when calculating concentration estimates. In this example we choose A to be that variable, because the difference caused by the different lysis buffers is negligible and the signal can thus be pooled. If we would like to obtain this information we would add it to another category as shown above with cellline and treatment.

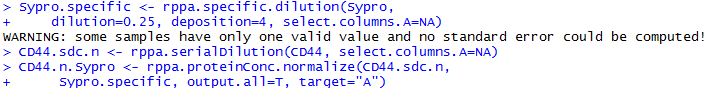




As you can see, this plot allows you to study the effect of the normalization in a convenient manner.

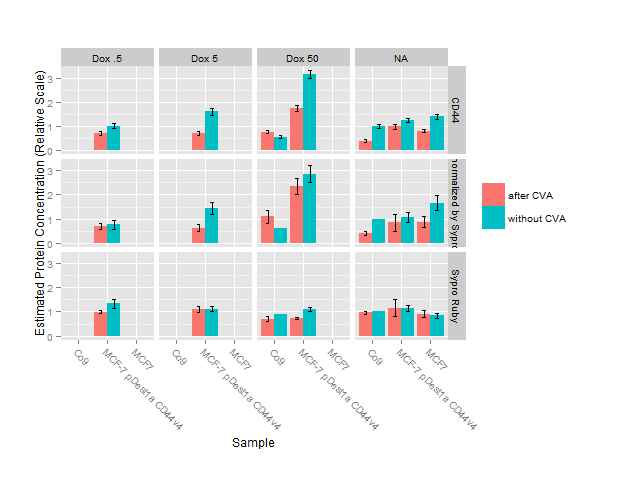
## Sypro Ruby

As an alternative it is common to use a whole-protein-dye such as Sypro Ruby for normalization. Since it is sometimes not possible to use the whole dilution series (the signal is too weak), we create suitable normalization data by only using the strongest dilution. We can create a subset of the Sypro Ruby data and normalize as follows:



Be aware of the warning message that informs you that for some samples only one Sypro Ruby value was given. The error bars for the respective samples will thus not be correctly and are omitted.

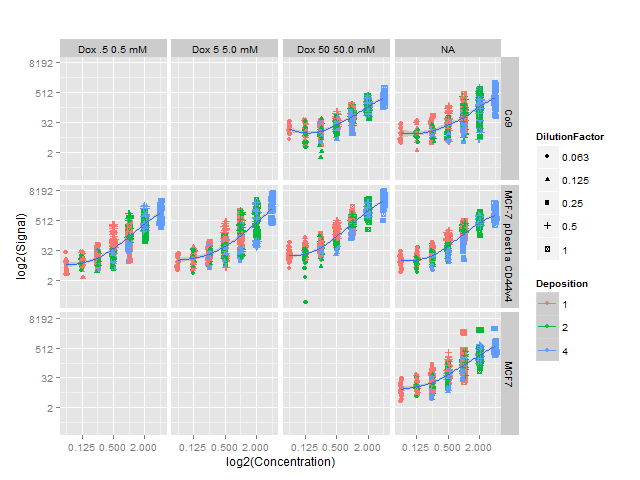




## Slide Concentration Plot:

Sometimes you might want to have a look at how the raw data behaves, more specifically: How does the dilution curve look for different sample categories. Problems will already become clear early in the analysis when plotting these data using:





This plot can be manipulated in a similar fashion as the previous plot by adjusting sample categories A and B. Furthermore, the log2 of the scales can be switched off if needed:

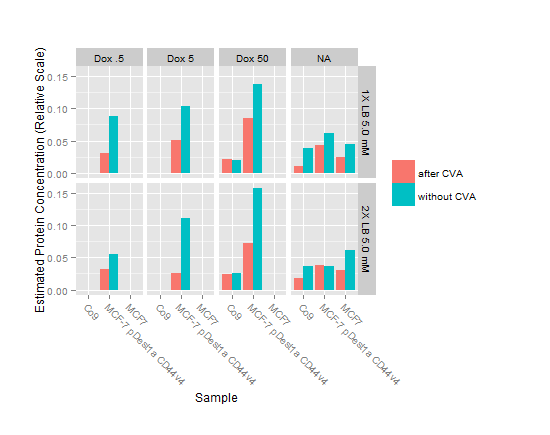


# A Few More Tricks

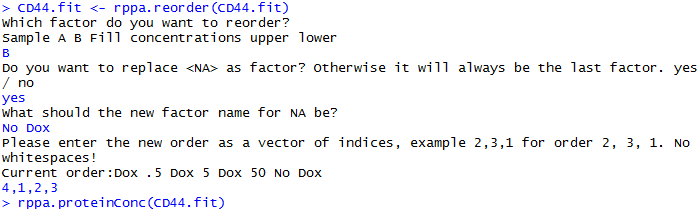
If you realize that there is a partial shift of a block you can take care of this by applying a manual shift in R. For instance if you want to shift rows 14 to 17 of block 12 vertically by -1, the command would be

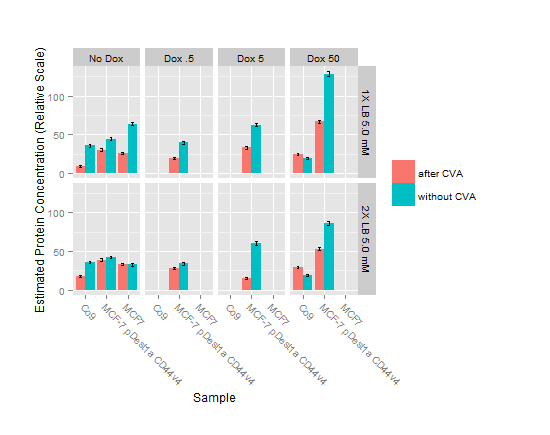


If you want to create a plot and you are not happy with the order of the factors of some category, you can change them easily. Assume we want to change the default order of the inducer dox in our sample data:



NA marks the samples without any inducer levels and should thus be plotted on the left hand side of the plot. We change the factor levels of any data frame like this:



As you can see in the next plot we have not only replaced NA with “No Dox”, we have also corrected the order of the factors.

# Method overview:

Many things have changed. A detailed description of all methods is coming soon ☺