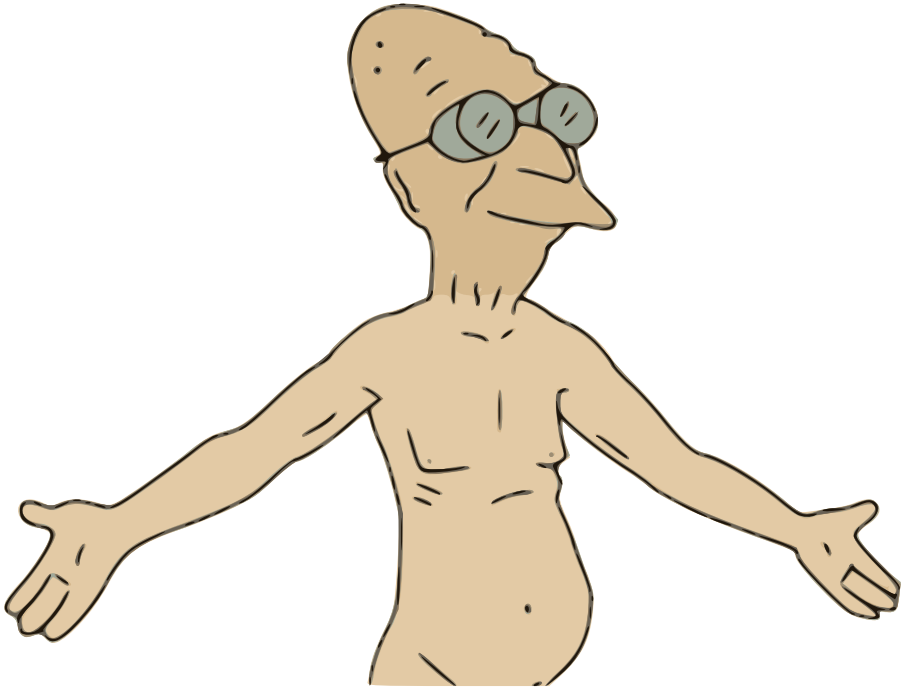
Hubert Farnsworth's Eternal Youth

Christian Diener Max Floettmann

Theoretical Biophysics
Humboldt University of Berlin
Invalidenstraße 42
10115 Berlin

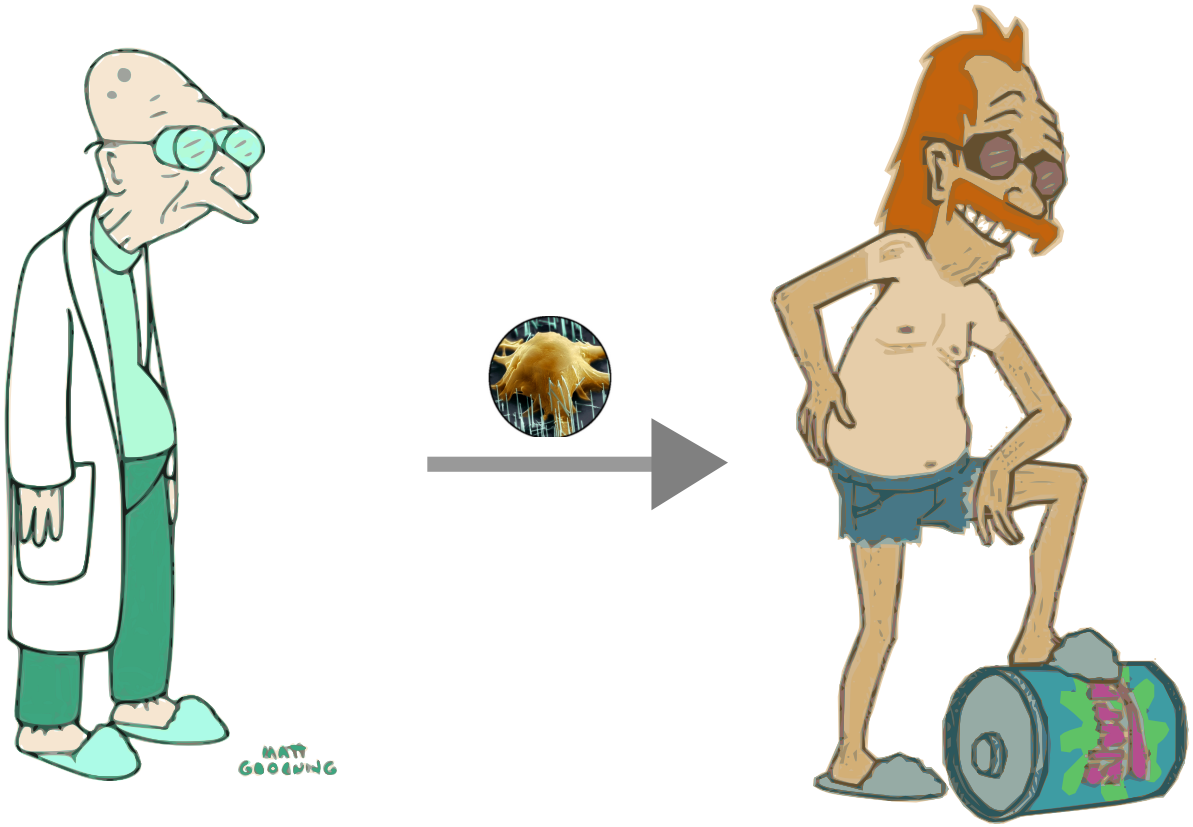
May 19th, 2010

Hubert J. Farnsworth



- Single
- Oldest human beeing on earth (167 years old)
- Professor at Mars University and owner of Planet Express (very busy)
- $\rightarrow$ Relationship trouble

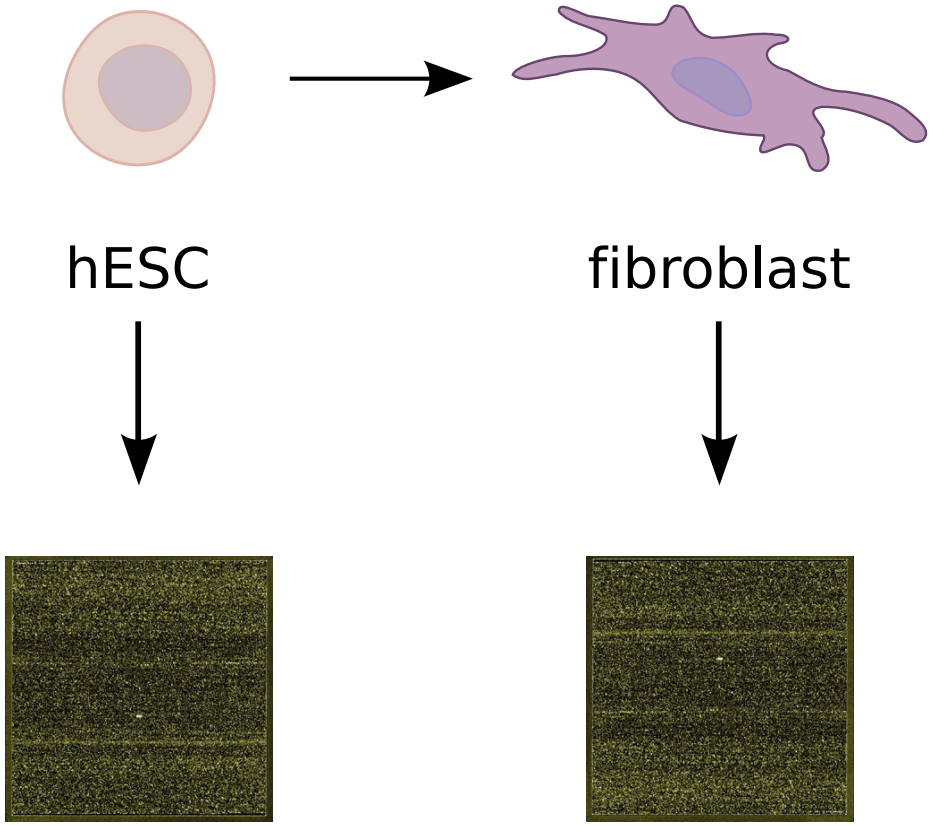
Farnsworth's Solution: Stem Cells



The Stem Cell Experiment

- Farnsworth buys some human embryonic stem cells (hESC) for a small fortune
- The cells make him look incredibly sexy, but the effect wears of quickly
- He decides to produce new stem cells by himself
- To understand the mechanism of stem cell production he decides to do some experiments on the hESCs he has left

The Stem Cell Experiment

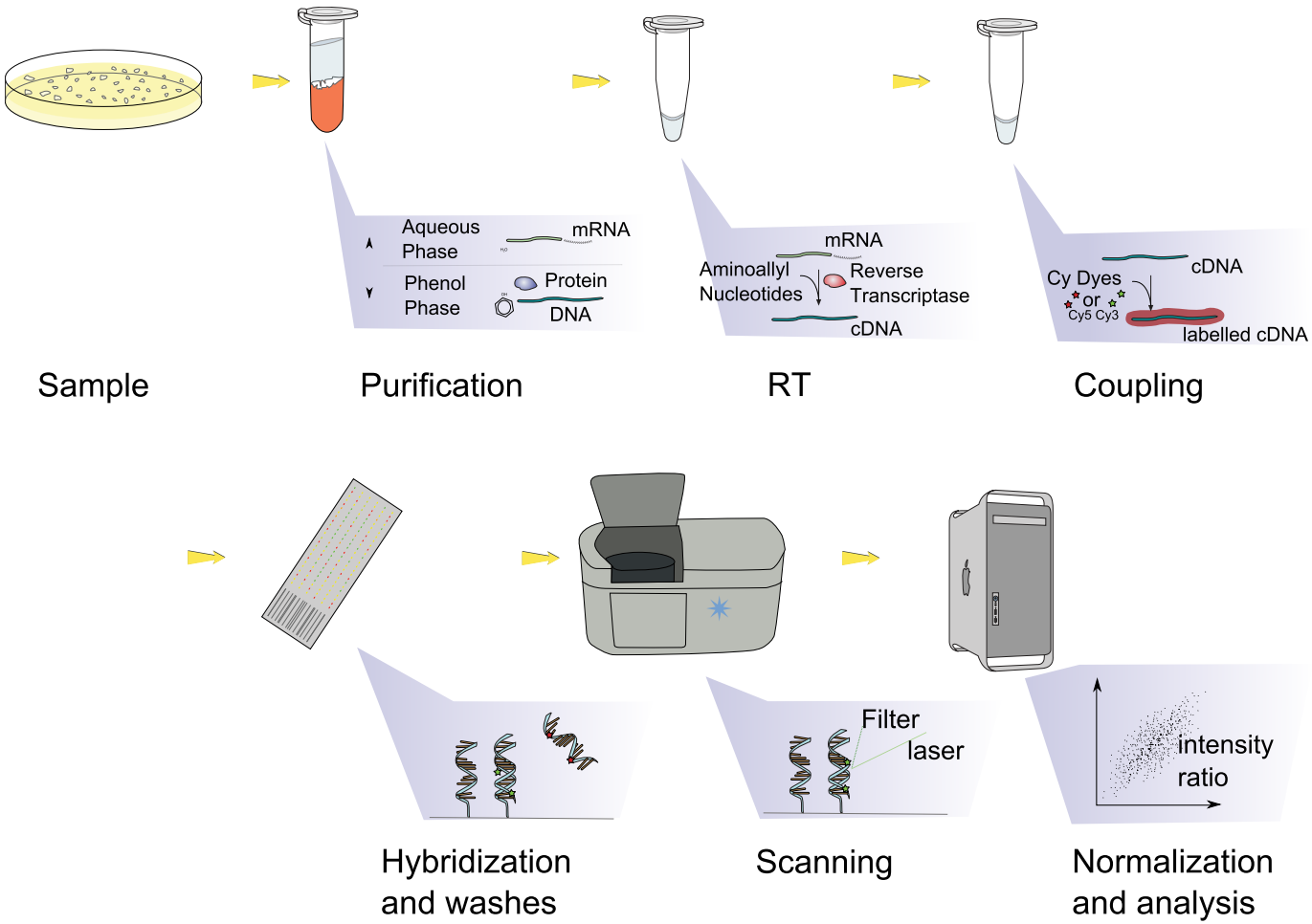


- Differentiation of stem cells by addition of Bmp4 to the media
- Expression profiling of different cell types by single dye microarrays

How do Microarrays Work Again?

1. cDNA or short sequences complementary to a set of genes(in our case all human genes) are printed on a chip
2. mRNA from the cells is transcribed to cDNA by reverse transcriptase
3. cDNA is labled by Cyanine dyes
4. hybridization of labeled *cDNA* and probes on the chip
5. Scanning of images and data analysis

Microarray Workflow



Bioconductor

- Bundle of R-packages for analysis of genomic data
- originally focused on microarrays, now supports a wide range of data
- provides connections to databases like Pubmed and Gene Ontology (GO)

Using Bioconductor

Install basic Bioconductor packages in *R*:

```
source("http://bioconductor.org/biocLite.R")
biocLite()
```

Install special packages:

```
biocLite(affy)
biocLite(c("limma","hgu133plus2.db"))
```

Loading the Data

Import the libraries:

```
library(hgu133plus2.db)
library(limma)
library(affy)
```

Load the Raw result files:

```
Data <- ReadAffy()
```

The function `ReadAffy()` reads all raw data files in the current directory. These files are produced directly by an Affymetrix machine. The function stores the data into an `AffyBatch` object. Have a look at the `AffyBatch` class and try a few of the functions (i.e. `boxplot()` and `hist()`) to become familiar with the dataset.

MA Plots and Normalization

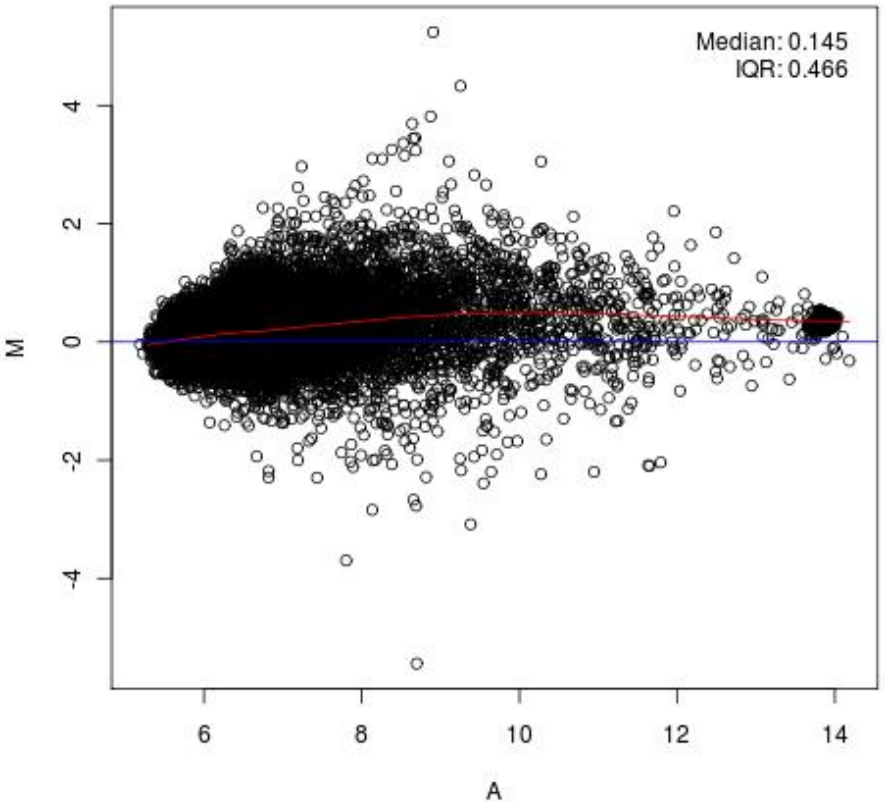
Plot:

```
y <- (exprs(Data)[1:20000, c("hESC1.CEL", "ME1.
    CEL")])
x11()
ma.plot( rowMeans(log2(y)), log2(y[, 1])-log2(y
    [, 2]), cex=1 )
title("Raw data MA plot. hESC1 vs ME1")
```

This code creates an MA plot of the expression values on the chip. MA plots show the relationship between the difference of intensity between experiments and the average expression on the chips for each gene. Try this for a few pairs of arrays, compare hESC vs. hESC and hESC vs. ME. If this takes to long just use subsets of the genes. You can also use look at the help of the `ma.plot()` function and change the `plot.method`for better overview. What does this show you?

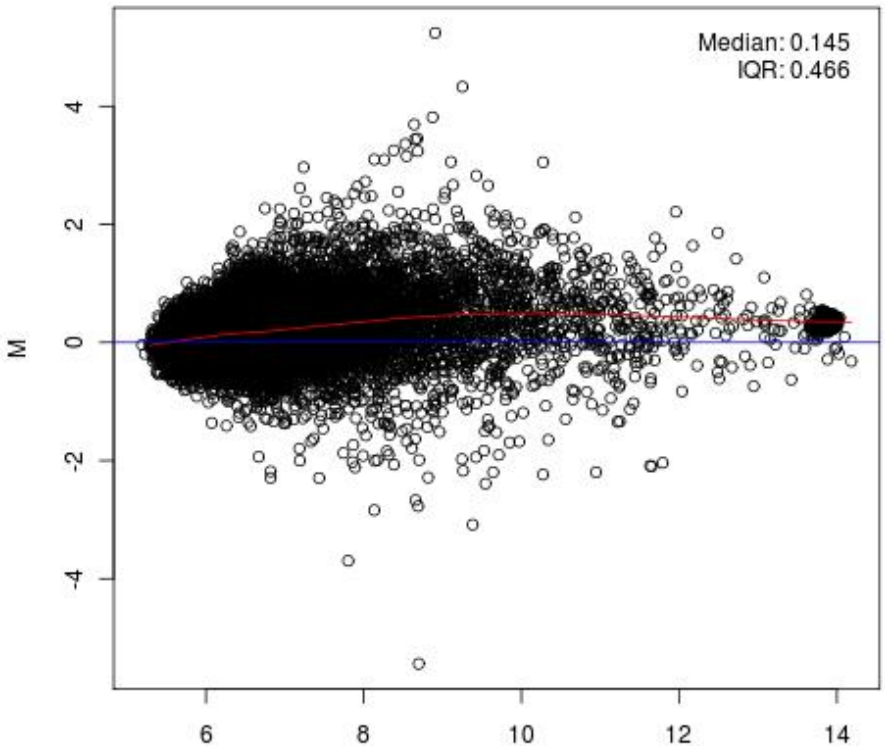
Example

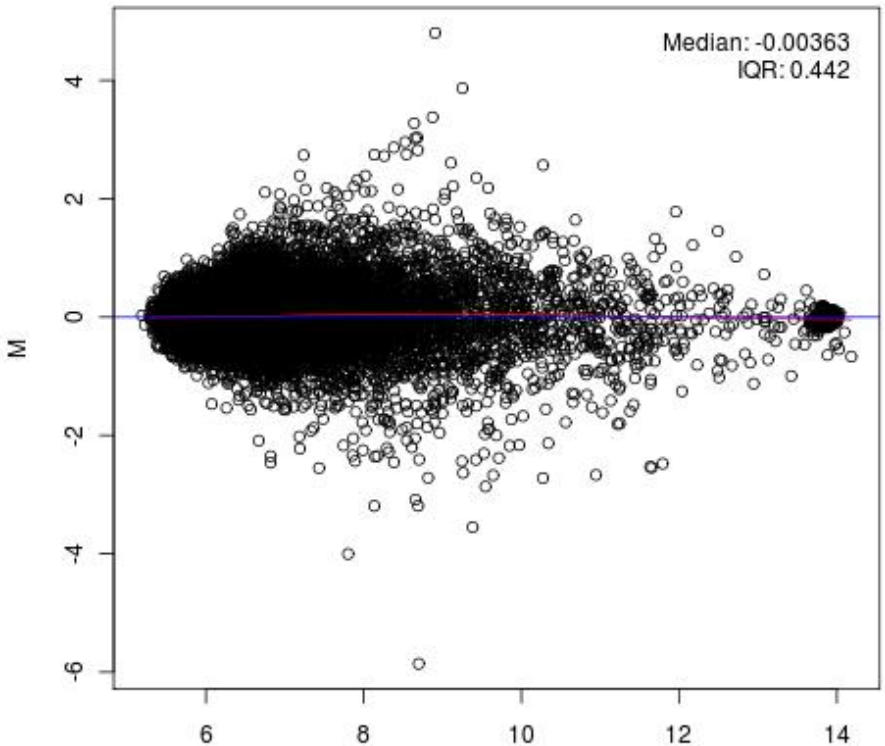
In a perfect situation the log-ratios should be evenly distributed around zero for all intensity values. In our case we clearly see intensity related artefacts that have to be corrected for. The red line in the plot shows the locally weighted linear regression (Lowess) of the dataset.



Normalization

A normalization can be done by subtracting this line from the data. Try to do a `loess.normalize()` on the data and once again plot an `ma.plot()` (try to change `plot.method` to `smoothScatter` again). In the following we use a more sophisticated method to for normalization, but we let Bioconductor to all the work for us.





Normalization

Correction and Normalization:

```
eset <- rma(Data)
```

By calling `rma()` we calculate the Robust Multichip Average expression measure for the given data. This function automatically does a background correction and normalizes before calculating the expression measure.

Normalization

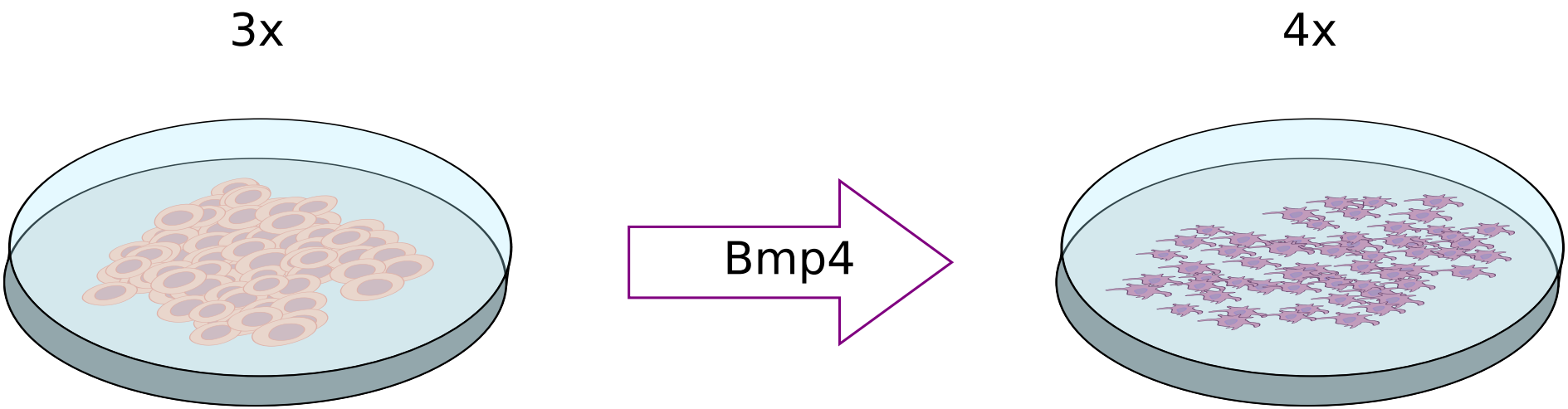
Look at the data:

```
exprs(eset[1:10,])
```

Now the values are normalized and the biologically significant differences in expression can be observed by comparing the values. Now we need an appropriate test for significant difference in expression, to see where our cell types differ the most.

Linear Model

To run a test with Bioconductor, we need to look at how we ran our experiment. Prof. Farnsworth had 4 cell lines. He did expression arrays of all of them in the stem cell state and than added Bmp4 to let them differentiate. After differentiation he did expression arrays again on all cells. Unfortunately he lost one of his stem cell arrays, so we only have 3 stem cell arrays and 4 mesoderm arrays.



Linear Model

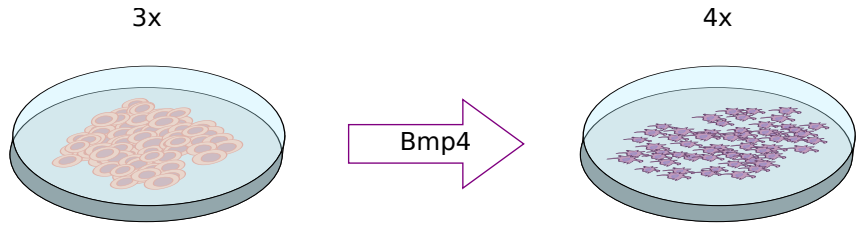
In general we assume that we have a response vector y_g for every gene g that is on the arrays. To build a linear model we also assume that the expected value of y_g can be estimated by:

$$E(y_g) = X\alpha_g \tag{1}$$

where α_g is a vector of coefficients and X is the design matrix of the experiment. The design matrix has n rows which correspond to the n chips we tested and m columns which correspond to the number of different conditions we tested.

Design Matrix

The first coefficient in the design matrix measures the log_2 expression value in the stem cells. The second coefficient measures the fold change in expression when bmp4 is added to the medium.



$$\begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ a \\ a \\ a+b \\ a+b \\ a+b \\ a+b \end{bmatrix}$$

Design Matrix

It is simple to create in our simple case, but can be hard to produce in more complicated experiments.

Create a design matrix:

```
strain <- c("hesc","hesc","hesc","mesc","mesc"
   ,"mesc","mesc")
design <- model.matrix(~factor(strain))
colnames(design) <- c("p","bmp4")
```

Linear Model

We can now fit the linear model defined by the design matrix to the Data we have. After that we can test for differential expression. To define and fit the linear model we use the following `R` commands.

Model and fitting:

```
fit <- lmFit(eset, design)
fit <- eBayes(fit)
```

`lmFit()` builds the linear models for each gene and fits them to the data. `eBayes()` computes test statistics for differential expression. This means that our null hypothesis is $H_0 : \beta_g = 0$.

Results

See the Results:

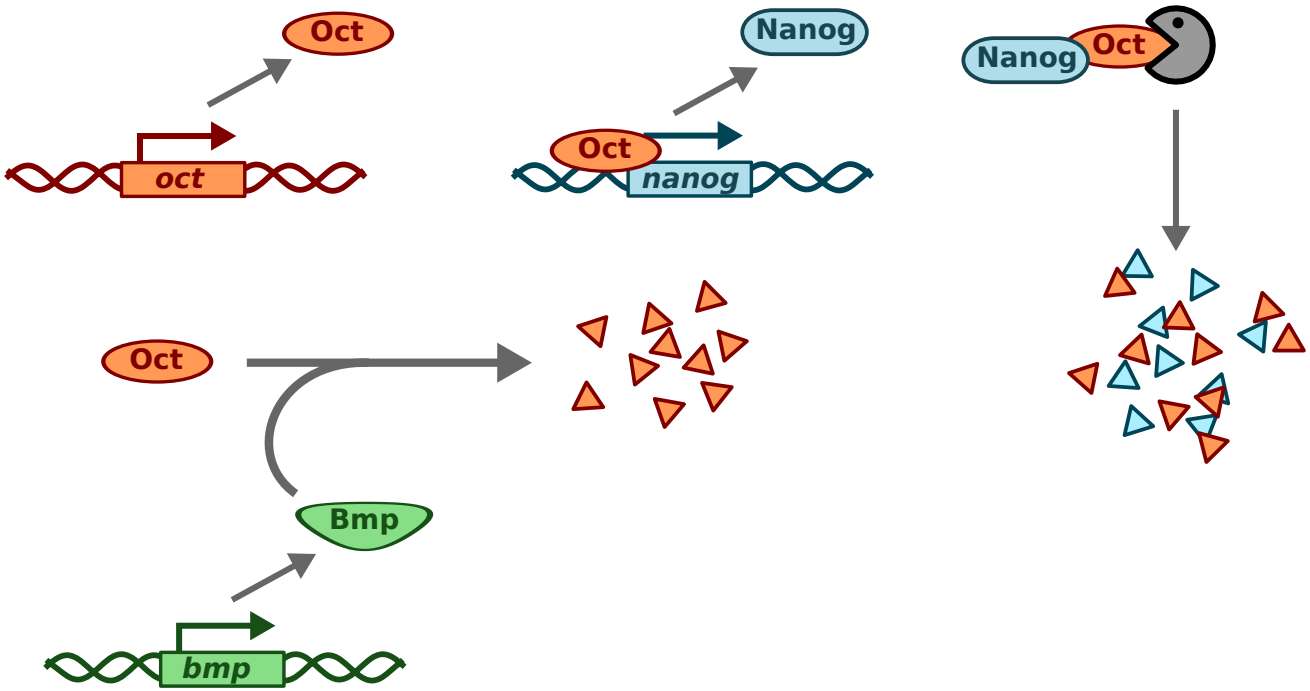
```
toptable <- topTable(fit, coef=2, n=100,
   adjust="BH")
toptable$ID <- mget(toptable$ID,
   hgu133plus2SYMBOL)
```

The code above gives you the list of the genes that show the strongest differential expression between the two cell lines. The second line is only needed to map the commonly used gene symbols to the identifiers used for the features on the chip. Look at the list and try to interpret the columns. If you have some time left try the plotting functions `volcanoplot()` and `tqq()` and think about what they show.

Wernstrom

On the Big Intergalactic Conference for Eternal Youth he hears a talk of his nemesis Prof. Wernstrom about the very same thing he has done. But based on much more experimental data. Wernstrom proposes a mechanism of stochastic differentiation of cells incubated with Bmp4. Farnsworth takes a lot of notes and combines Wernstroms Data with his own and comes up with a model of differentiation.

Model



Good news everyone: Finished!