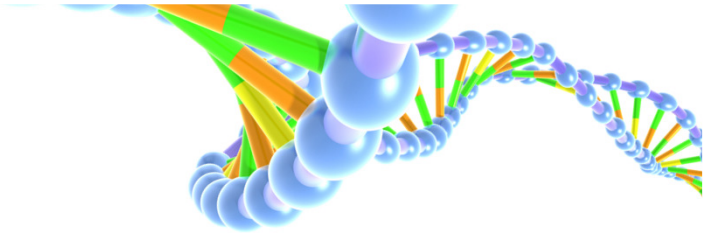
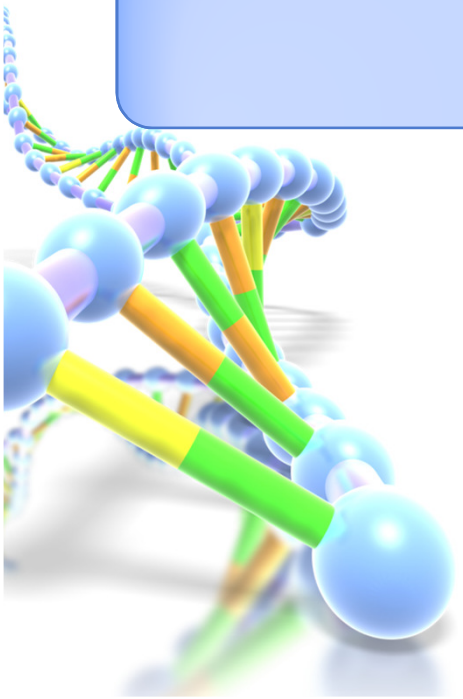


Les 14 - Microarray's en differentiële gen expressie (3)

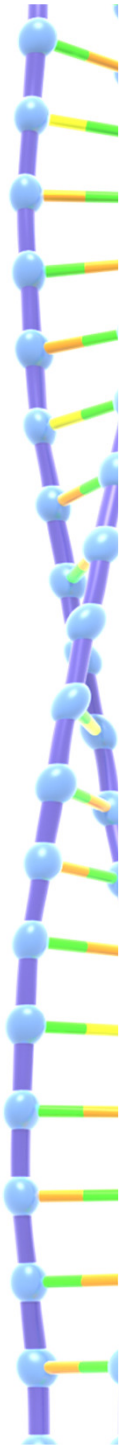
Emile Apol



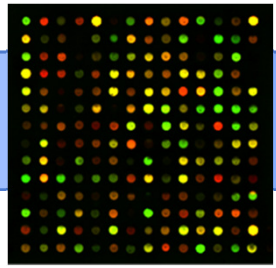
Hanze University Groningen
Applied Sciences
Institute for
Life Science & Technology

LES 14

- Preprocessing
  - Data selectie/manipulatie in R (**`all`**, **`any`**, **`which`** etc.)
  - Normalisatie (**`mean`**, **`lowess`**)
  - Regressie: **`lm`**, plotten: **`plot`**, **`lines`**, **`predict`**
- Statistische analyse per gen
  - *t*-toets
  - Wilcoxon's toets
  - 1-way ANOVA



MICROARRAY ANALYSE: STAPPENPLAN



- Background correctie
- Log transformatie
- Normalisatie (bijv. loess)
- Toetsen op DEG's:
  - *t*-toets, 1-way ANOVA, …
  - Wilcoxon's toets, Kruskall-Wallis toets, …
- Aanpassen *p*-waarden voor multiple toetsing
- Clustering van DEG's:
  - Hiërarchisch clusteren
  - *k*-means
  - Principale Componenten Analyse (PCA)
- Toetsen op functionaliteit genen binnen clusters

DATA SELECTIE IN R

- Enkele handige functies:

```
x <- c(10, 1, 1, 2, 3, 4, 5, 4, 5, 7, 8, 9, 10)
unique(x)          # unieke elementen overhouden
any(x > 2)         # is er een element in x > 2?
any(x > 10)        # is er een element in x > 10?
all(x > 2)         # zijn alle elementen in x > 2?
all(x > -1)        # zijn alle elementen in x > -1?
which(x ==10)      # welke elementen (index) in x zijn 10?
which(x != 1)      # welke elementen (index) in x zijn niet 1?
```

```
> x <- c(10, 1, 1, 2, 3, 4, 5, 4, 5, 7, 8, 9, 10)
> unique(x)         # unieke elementen overhouden
[1] 10  1  2  3  4  5  7  8  9
> any(x > 2)        # is er een element in x > 2?
[1] TRUE
> any(x > 10)       # is er een element in x > 10?
[1] FALSE
> all(x > 2)        # zijn alle elementen in x > 2?
[1] FALSE
> all(x > -1)       # zijn alle elementen in x > -1?
[1] TRUE
> which(x ==10)     # welke elementen (index) in x zijn 10?
[1]  1 13
> which(x != 1)     # welke elementen (index) in x zijn niet 1?
 [1]  1  4  5  6  7  8  9 10 11 12 13
```

DATA SELECTIE IN R

- Verwijderen van elementen: negatieve index

```
x <- c(10, 1, 1, 2, 3, 4, 5, 4, 5, 7, 8, 9, 10)
x.clean <- x[-1] # verwijder 1e element uit x
x.clean
(x.clean <- x[-c(1, 3)]) # verwijder 1e en 3e element uit x en laat zien
x.clean <- x[-which(x > 7)] # verwijder alle elementen uit x > 7
x.clean
```

```
> x <- c(10, 1, 1, 2, 3, 4, 5, 4, 5, 7, 8, 9, 10)
> x.clean <- x[-1] # verwijder 1e element uit x
> x.clean
 [1]  1  1  2  3  4  5  4  5  7  8  9 10
> (x.clean <- x[-c(1, 3)]) # verwijder 1e en 3e element uit x en laat zien
 [1]  1  2  3  4  5  4  5  7  8  9 10
> x.clean <- x[-which(x > 7)] # verwijder alle elementen uit x > 7
> x.clean
[1] 1 1 2 3 4 5 4 5 7
```

DATA SELECTIE IN R

- **`which( )`** functie op matrices/dataframes:

```
M <- matrix(21:32, nrow=3)
View(M)
which(M > 27)
which(M > 27, arr.ind=T)
which(M > 27, arr.ind=T)[,"row"]
which(M > 27, arr.ind=T)[,"col"]
unique(which(M > 27, arr.ind=T)[,"row"])
```

Data: M

File

	V1	V2	V3	V4
1	21	24	27	30
2	22	25	28	31
3	23	26	29	32

```
> which(M > 27)
[1]  8  9 10 11 12
> which(M > 27, arr.ind=T)
     row col
[1,]   2   3
[2,]   3   3
[3,]   1   4
[4,]   2   4
[5,]   3   4
> which(M > 27, arr.ind=T)[,"row"]
[1] 2 3 1 2 3
> which(M > 27, arr.ind=T)[,"col"]
[1] 3 3 4 4 4
> unique(which(M > 27, arr.ind=T)[,"row"])
[1] 2 3 1
```

GENERAL position

MATRIX position

1	4	7	10
2	5	8	11
3	6	9	12

rows met waarden > 27

unieke rows met waarden > 27

6

DATA SELECTIE IN R

- Verwijder uit dit dataframe alle rows met een flag “F1” of “F2” die anders zijn dan “0”:

Data: M

File

	row.names	M1	F1	M2	F2
1	gene1	1.5	0	1.1	0
2	gene2	2.5	0	2.1	1
3	gene3	-0.5	1	0.2	0
4	gene4	-3.2	1	-2.5	1
5	gene5	1.1	0	2.3	0

```
# Remove rows with any flag F* != 0 from data

# Make names of flag columns
flagNames <- paste("F", 1:2, sep="")
# select columns with flags
M[,flagNames]
M.flags <- M[,flagNames]
View(M.flags)
# which indices are != 0? in GENERAL position
which(M[,flagNames] != 0)
# which indices are != 0? in MATRIX (= row/col) position
which(M[,flagNames] != 0, arr.ind=T)
# select all rows with wrong flags
which(M[,flagNames] != 0, arr.ind=T)[,"row"]
# remove all duplicates in rows
unique(which(M[,flagNames] != 0, arr.ind=T)[,"row"])
wrongRows <- unique(which(M[,flagNames] != 0, arr.ind=T)[,"row"])
# remove wrong rows from data frame
M.clean <- M[-wrongRows,]
View(M.clean)
```

negatieve index: verwijderen

DATA SELECTIE IN R

- Resultaat:

Data: M

File

	row.names	M1	F1	M2	F2
1	gene1	1.5	0	1.1	0
2	gene2	2.5	0	2.1	1
3	gene3	-0.5	1	0.2	0
4	gene4	-3.2	1	-2.5	1
5	gene5	1.1	0	2.3	0

Data: M.flags

File

	row.names	F1	F2
1	gene1	0	0
2	gene2	0	1
3	gene3	1	0
4	gene4	1	1
5	gene5	0	0

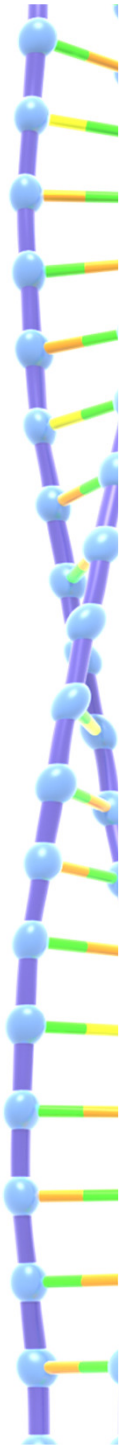
Data: M.clean

File

	row.names	M1	F1	M2	F2
1	gene1	1.5	0	1.1	0
2	gene5	1.1	0	2.3	0

DATA MANIPULATIE IN R

- Toevoegen van extra kolomvector **`newCol`** aan dataframe **`myData`** onder de naam “extra”:
  - **`myData["extra"] <- newCol`**
  - **`cbind(myData, extra=newCol)`**
  - **`myData <- data.frame(myData, extra=newCol)`**



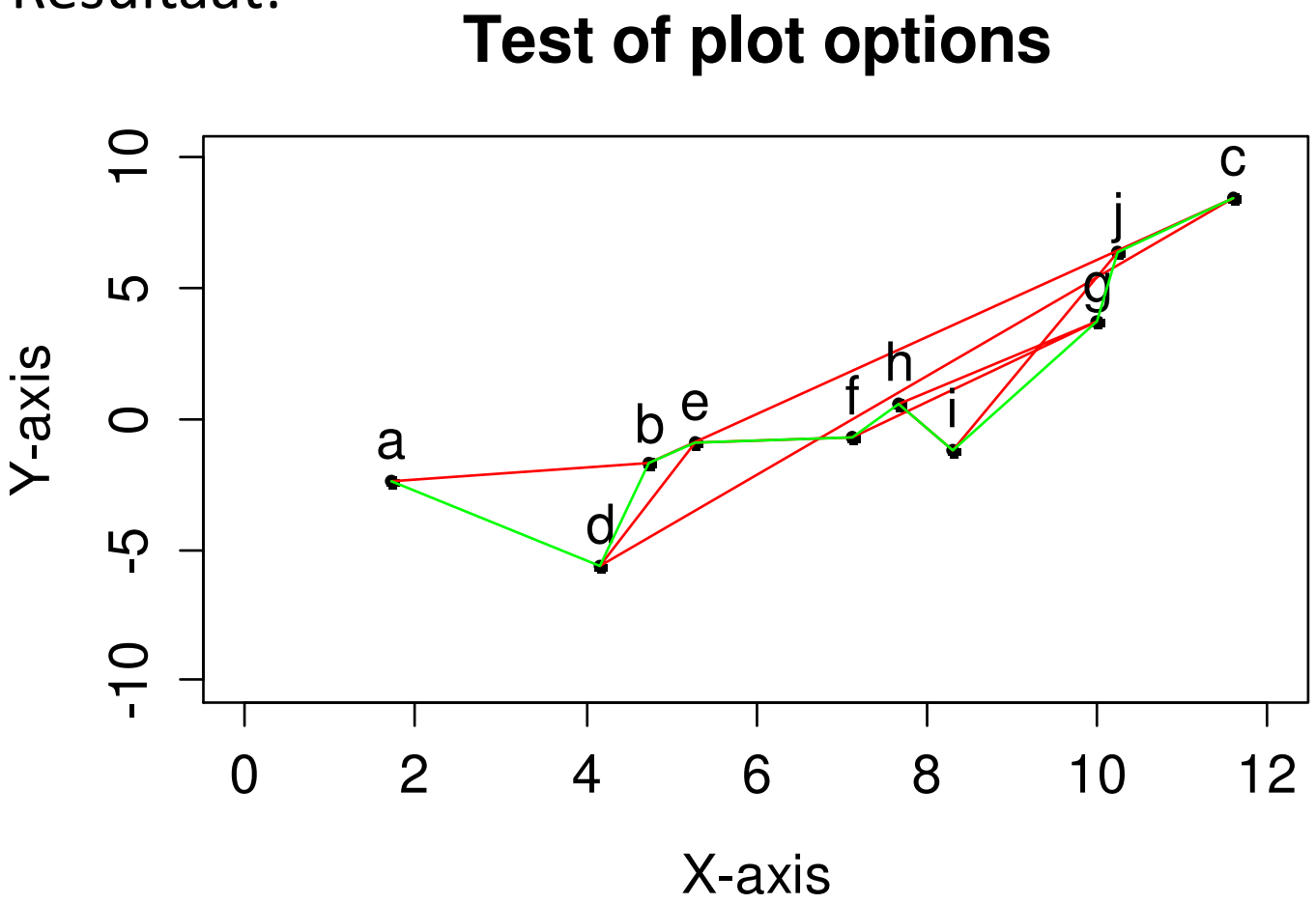
GRAFIEKEN: PLOT

- Voorbeeld: 1) leeg “canvas” maken, 2) punten, 3) lijnen en 4) tekst toevoegen:

```
# plot without data: only frame
plot(x, y, type="n",
     xlab="X-axis", ylab="Y-axis",
     xlim=c(0,12), ylim=c(-10,10),
     main="Test of plot options")
# add points
points(x, y, pch=20)
# connect by lines
lines(x, y, col="red")
# add text ABOVE each point (pos=3)
text(x, y, pos=3,
     labels=c("a", "b", "c", "d", "e", "f", "g", "h", "i", "j"))
ORDER <- order(x)
lines(x[ORDER], y[ORDER], col="green")
```

GRAFIEKEN: PLOT

- Resultaat:



REGRESSIE: LINEAIRE MODELLEN

- Model: $y = a_0 + a_1 * x$ (rechte lijn)

```
# fit and plot straight line y = a0 + a1*x
fit.lm <- lm(y ~ x)
summary(fit.lm)

y.lm.1 <- fit.lm$fitted.values
y.lm.1 <- predict(fit.lm)
lines(x, y.lm.1, col="green")
lines(y.lm.1 ~ x, col="green")

xplot <- seq(0,12,0.1)
y.lm.2 <- predict(fit.lm, newdata=data.frame(x=xplot))
lines(xplot, y.lm.2, col="black")
lines(y.lm.2 ~ xplot, col="black")

abline(fit.lm, col="blue")
```

"info" regressie

predict: bereken op basis van fit

"formule" manier

"formule" manier

rechte lijn y = a + b x

REGRESSIE: LINEAIRE MODELLEN

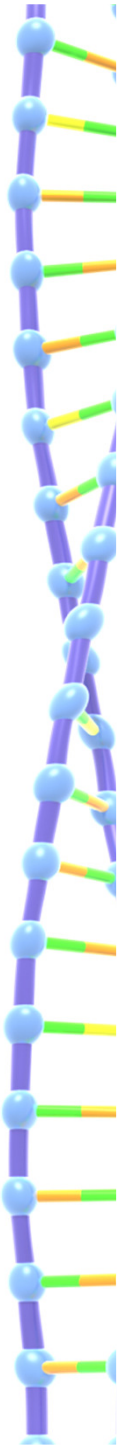
- Output:

$$y = a_0 + a_1 \cdot x$$

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3073 -1.2307 -0.0328  1.6126  3.3240

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7.7331     1.9136  -4.041  0.00373 **
x             1.1857     0.2495   4.753  0.00144 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.323 on 8 degrees of freedom
Multiple R-squared: 0.7385, Adjusted R-squared: 0.7058
F-statistic: 22.59 on 1 and 8 DF,  p-value: 0.001439
```

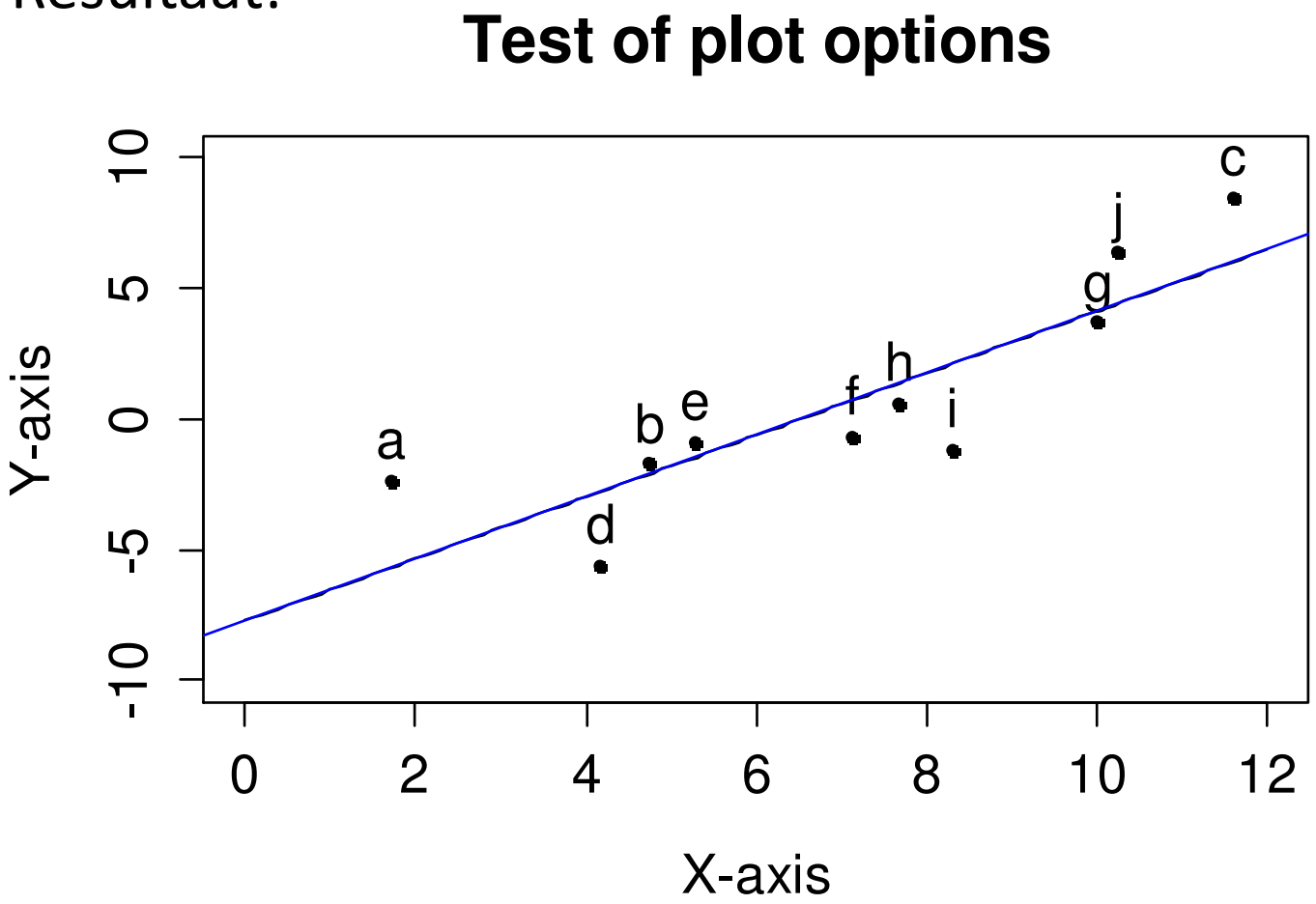
a_0

a_1



REGRESSIE: LINEAIRE MODELLEN

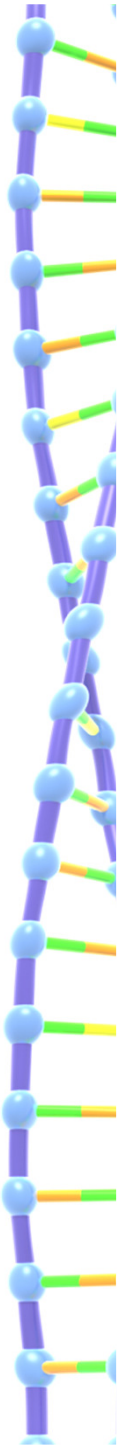
- Resultaat:



REGRESSIE: LINEAIRE MODELLEN

- Model: $y = a_0 + a_1*x + a_2*x^2$ (parabool)

```
# fit and plot parabola y = a0 + a1*x + a2*x^2
fit.lm <- lm(y ~ x + I(x^2))
summary(fit.lm)

y.lm.1 <- fit.lm$fitted.values
y.lm.1 <- predict(fit.lm)
lines(x, y.lm.1, col="green")
lines(y.lm.1 ~ x, col="green")

xplot <- seq(0,12,0.1)
y.lm.2 <- predict(fit.lm, newdata=data.frame(x=xplot))
lines(xplot, y.lm.2, col="black")
lines(y.lm.2 ~ xplot, col="black")

abline(fit.lm, col="blue")
```



REGRESSIE: LINEAIRE MODELLEN

- Output:

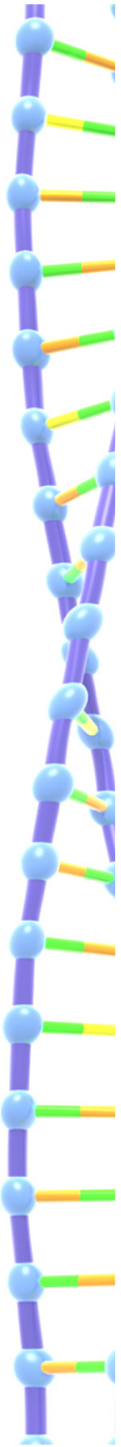
$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2$$

```
Call:
lm(formula = y ~ x + I(x^2))

Residuals:
    Min      1Q  Median      3Q     Max
-2.5078 -0.5672  0.1918  1.0679  1.7548

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.92817    2.58935  -0.358   0.7306
x           -1.28159    0.82287  -1.557   0.1633
I(x^2)       0.18136    0.05911   3.068   0.0181 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.622 on 7 degrees of freedom
Multiple R-squared: 0.8885, Adjusted R-squared: 0.8566
F-statistic: 27.88 on 2 and 7 DF,  p-value: 0.0004633
```

a_0

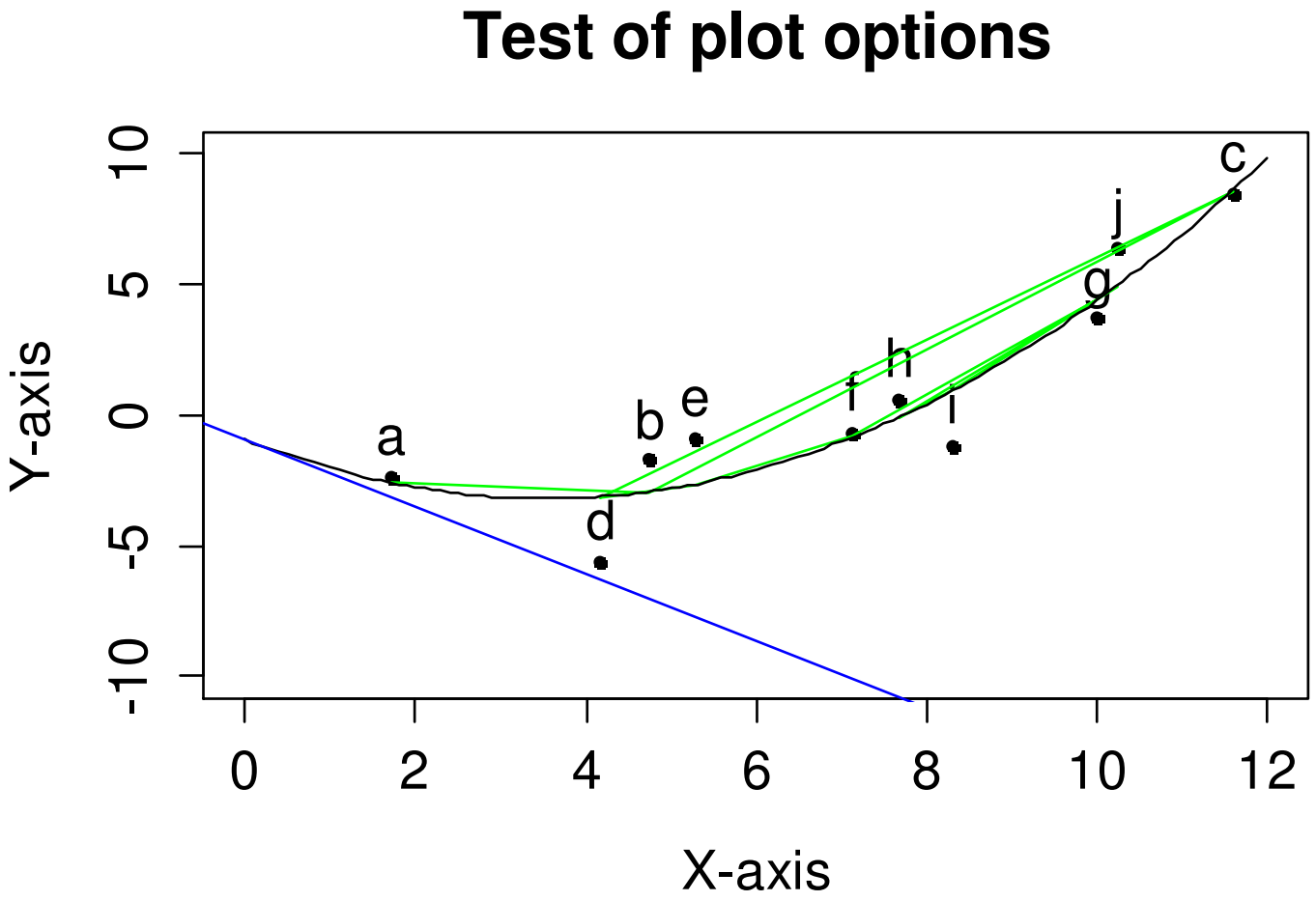
a_1

a_2



REGRESSIE: LINEAIRE MODELLEN

- Resultaat:



REGRESSIE: LO(W)ESS

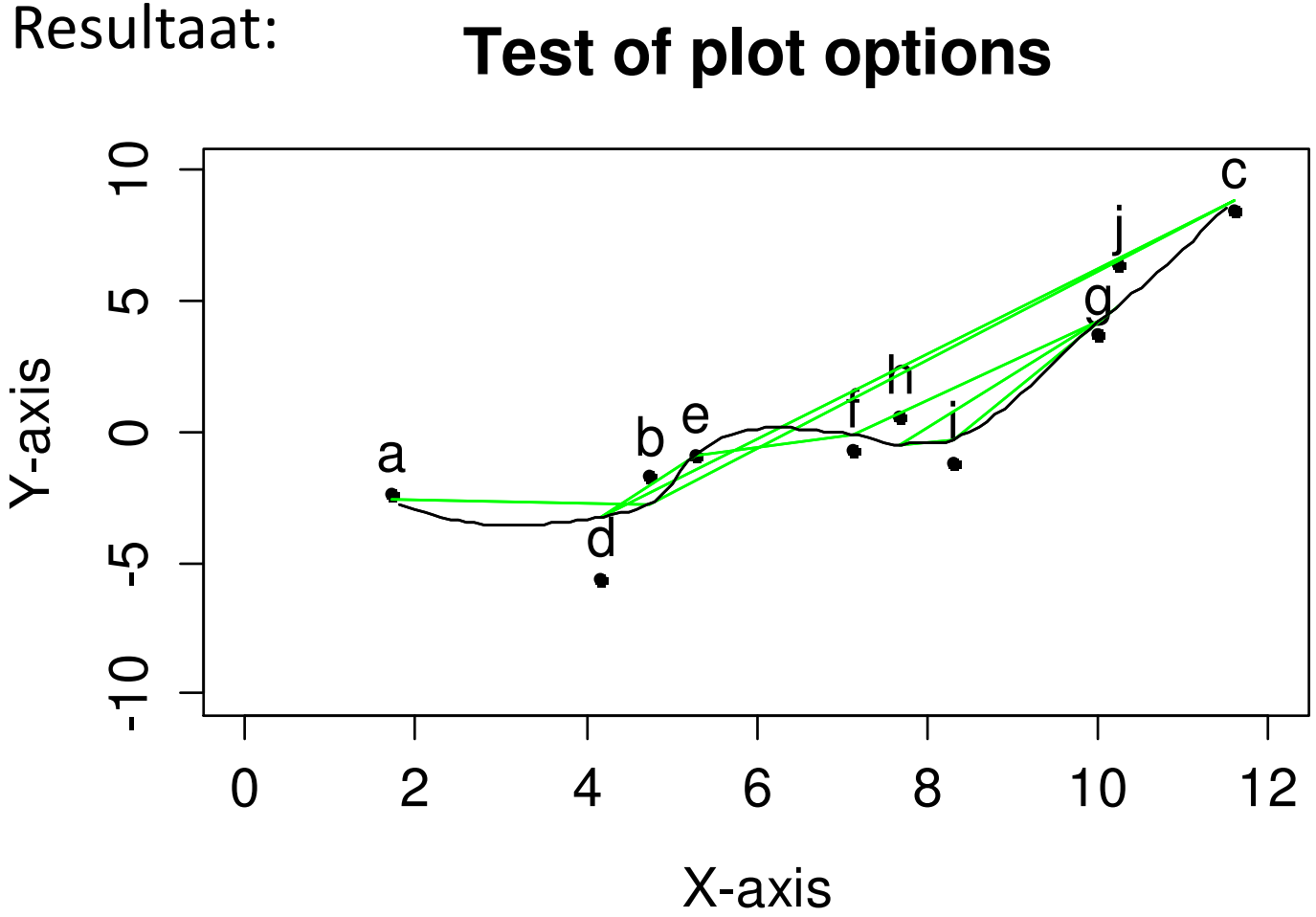
- Model: lowess (locale regressie)

```
# fit and plot a loess function
fit.loess <- loess(y ~ x)

y.loess.1 <- fit.loess$fitted
y.loess.1 <- predict(fit.loess)
lines(x, y.loess.1, col="green")
lines(y.loess.1 ~ x, col="green")

xplot <- seq(0,12,0.1)
y.loess.2 <- predict(fit.loess, newdata=data.frame(x=xplot))
lines(xplot, y.loess.2, col="black")
lines(y.loess.2 ~ xplot, col="black")

# in 1 stap data en loess curve plotten:
scatter.smooth(x, y, xlab="X-axis", ylab="Y-axis")
scatter.smooth(y ~ x, xlab="X-axis", ylab="Y-axis")
```

Belangrijk: omdat in de (loess) fit “x” de verklarende variabele is, moet je de variabele bij newdata OOK “x” noemen!

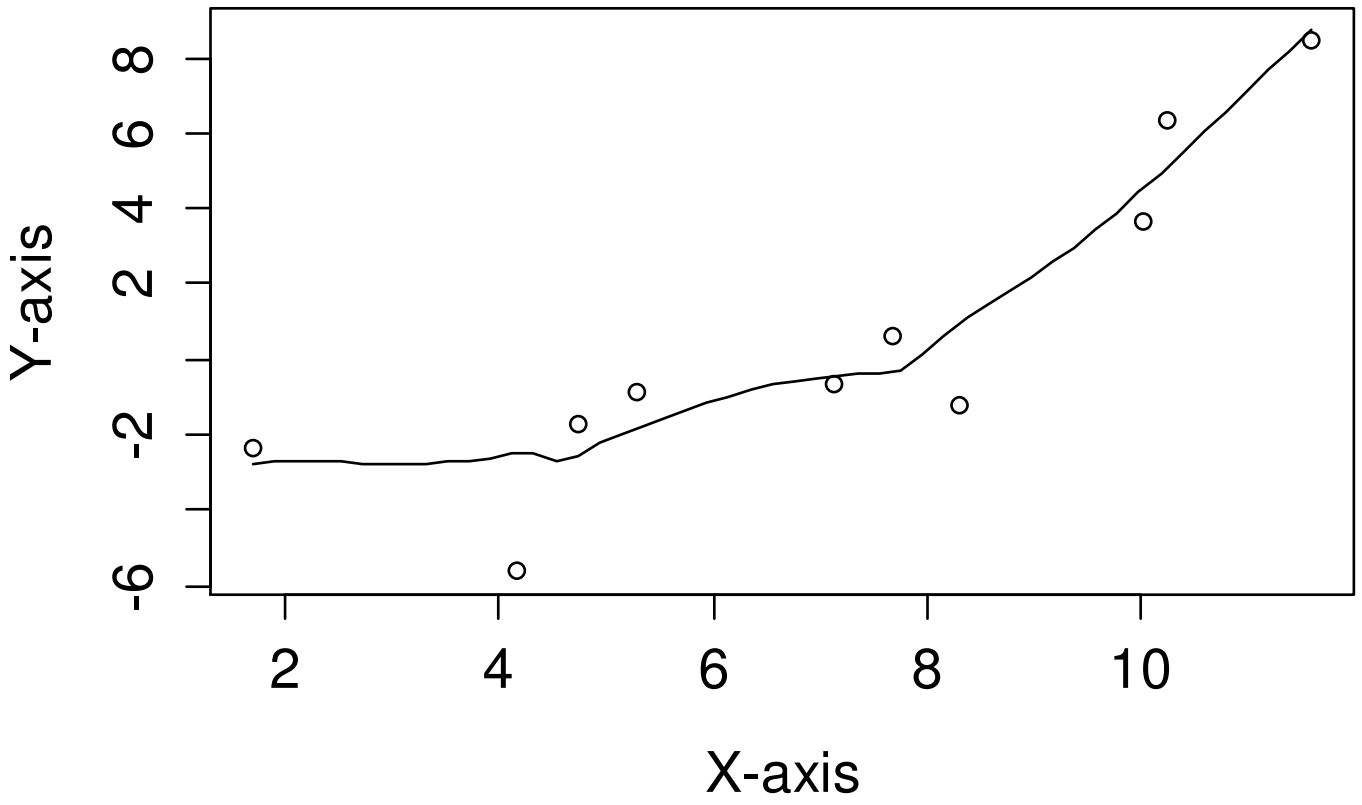
REGRESSIE: LO(W)ESS

- Resultaat:



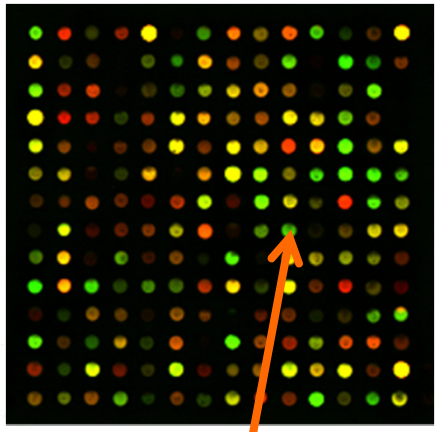
SCATTER.SMOOTH

- Resultaat:

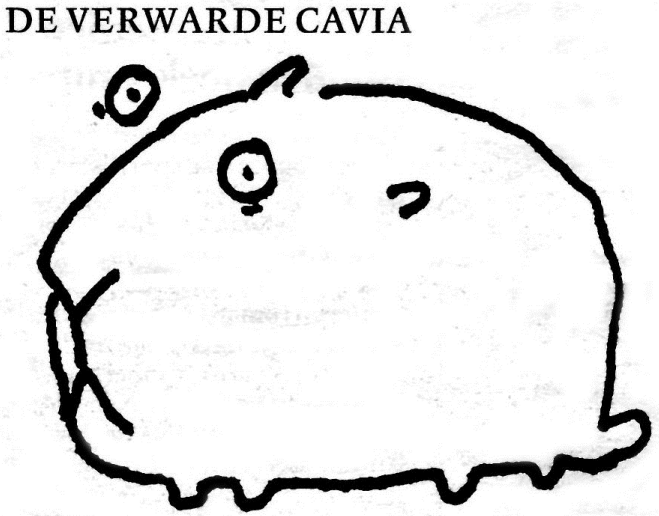


DATA PRE-PROCESSING

- Verschillende data pre-processing stappen
- Niet altijd elke stap en niet altijd in zelfde volgorde
  - background correctie
  - log transformatie
  - normalisatie



spot *i*



NORMALISATIE

- Dual channel (*R* = Cy5, *G* = Cy3) array
- Voer per channel background correcties uit per gen (spot) i $\quad R_i \equiv R_{i,\mathrm{raw}} - R_{i,\mathrm{bg}} \quad G_i \equiv G_{i,\mathrm{raw}} - G_{i,\mathrm{bg}}$
- Bereken per microarray per gen (spot) i de log expressie ratio ("log fold change")

ook wel "*R*"

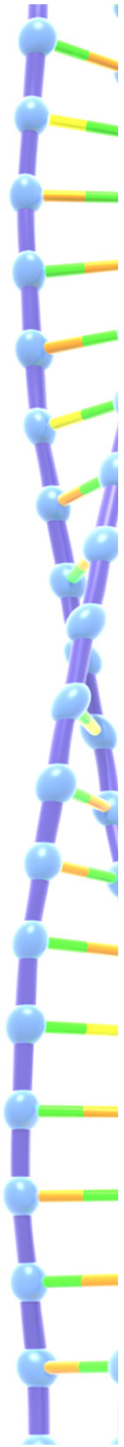
$$M_i \equiv {}^2\log\left(\frac{R_i}{G_i}\right) = {}^2\log(T_i) = \left[{}^2\log(R_i) - {}^2\log(G_i)\right]$$

- Bereken per microarray over alle genen (spots) i de log gemiddelde expressie ("log intensity")

ook wel "*I*"

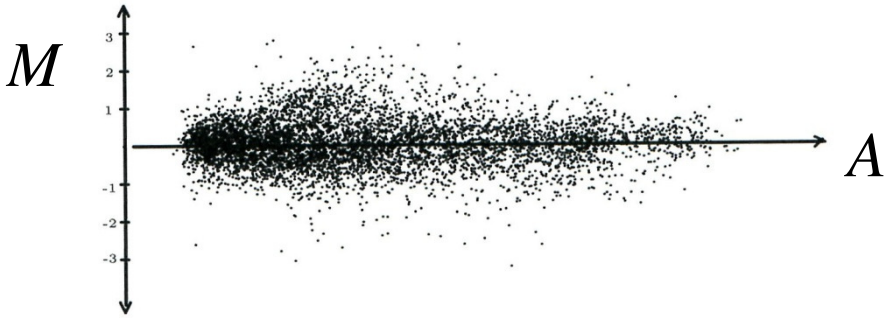
$$A_i \equiv {}^2\log\left(\sqrt{R_i \cdot G_i}\right) = \tfrac{1}{2}\left[{}^2\log(R_i) + {}^2\log(G_i)\right]$$

geometrisch gemiddelde



NORMALISATIE

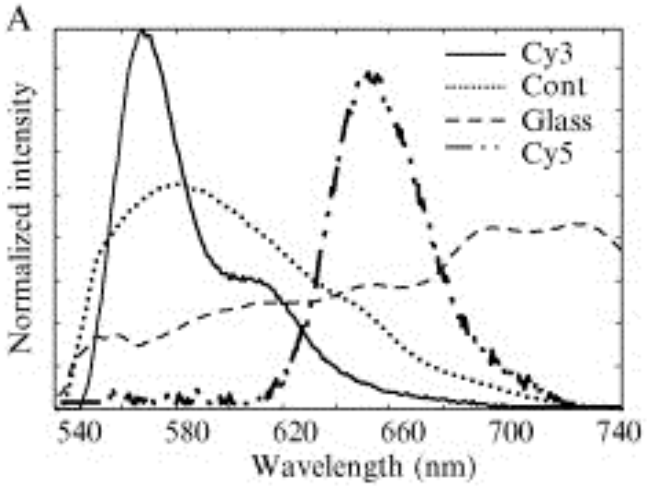
- Verwachting:
  - M vs A (= R vs I) is een puntenwolk rondom M = 0, en ongeveer symmetrisch in A



- Zo niet: dan effect van (technische) bias
  - ongelijke hoeveelheden mRNA sample
  - verschil in labeling efficiëntie dyes
  - verschil in detectie efficiëntie dyes
  - …

PROBLEMEN: BRONNEN VAN VARIATIE

- Verschil tussen array's
- Verschil in probe dichtheid tussen spots binnen array
- Delen array gemaakt door verschillende printer tips
- Background correctie
- Verschil hybridisatie tussen genen
- Effect van dye
- Interactie gen - dye
- Dag/tijdstip/analist/…



- Verschil tussen weefsel
- Verschil tussen individuen

- Differentieel verschil tussen genen

NORMALISATIE

- Per microarray meestal per gen (spot) i
  - mean normalisatie

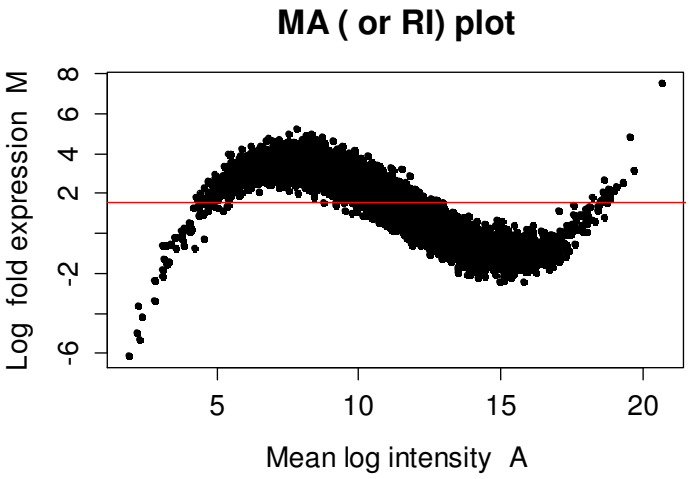
$$M_i' = M_i - \overline{M}$$

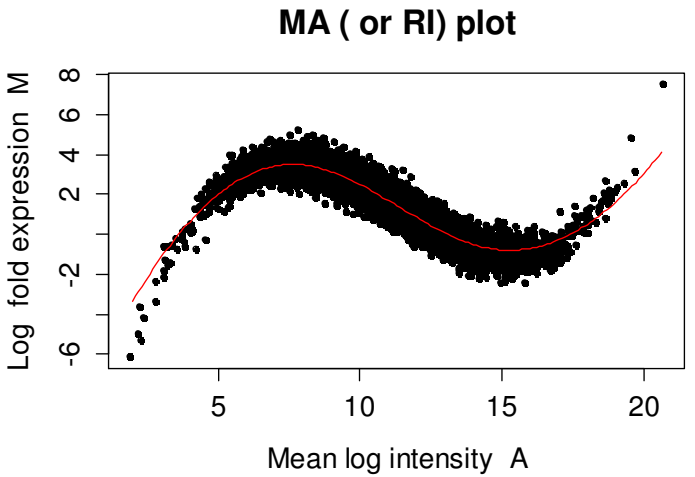
gemiddeld over hele array

  - lowess (loess) normalisatie

$$M_i' = M_i - \text{lowess}(A_i)$$

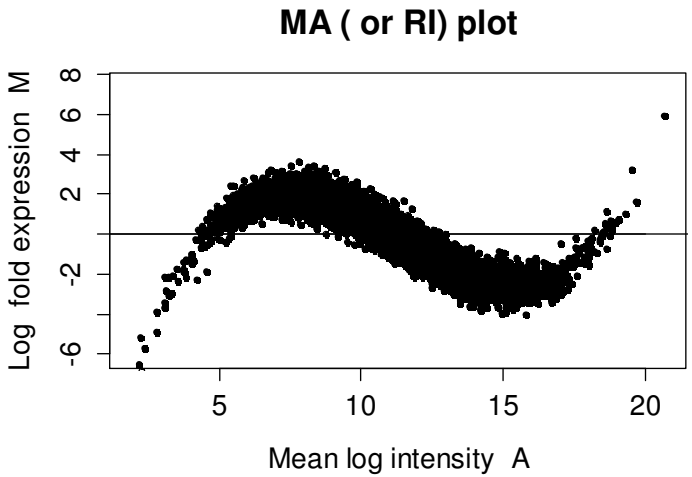
locale fit door puntenwolk



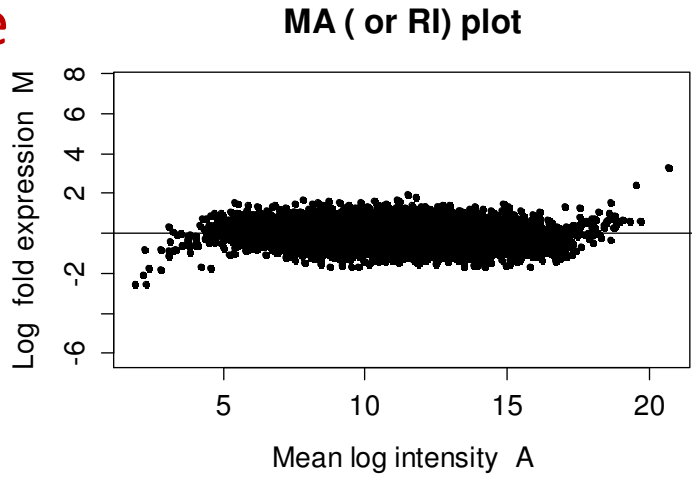


NORMALISATIE

- Resultaat:
  - mean normalisatie



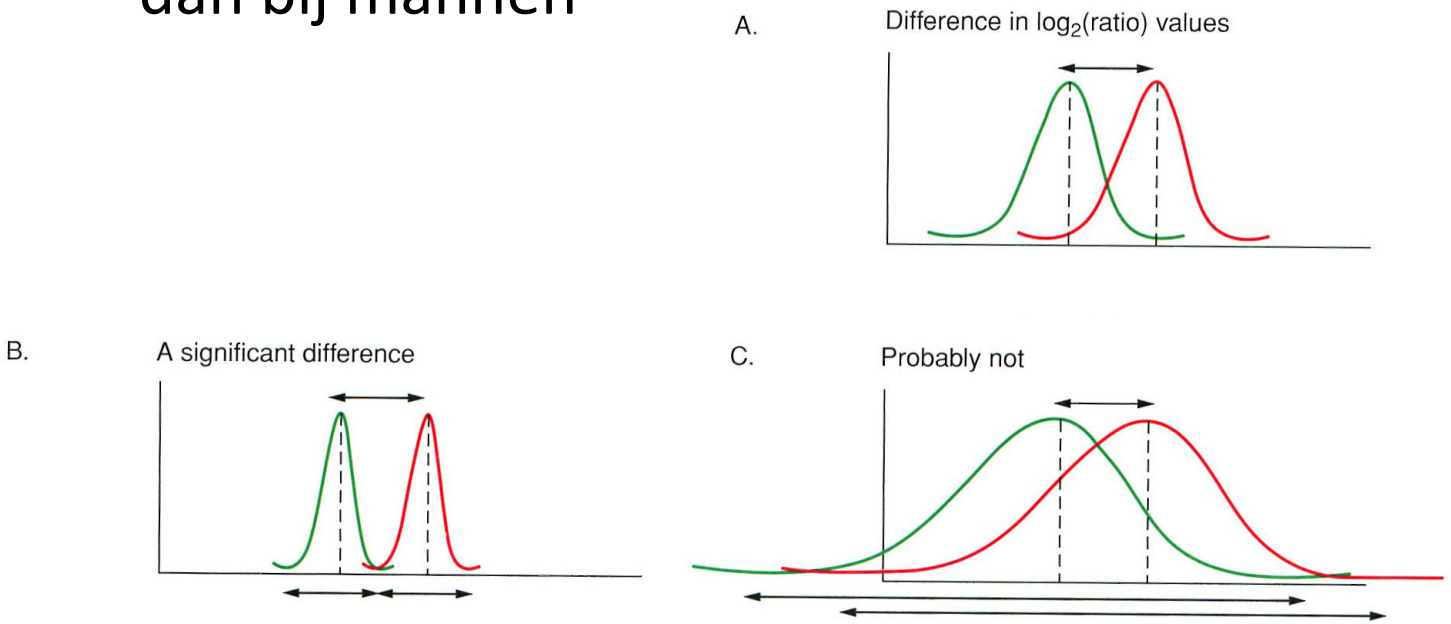
  - lowess (loess) normalisatie



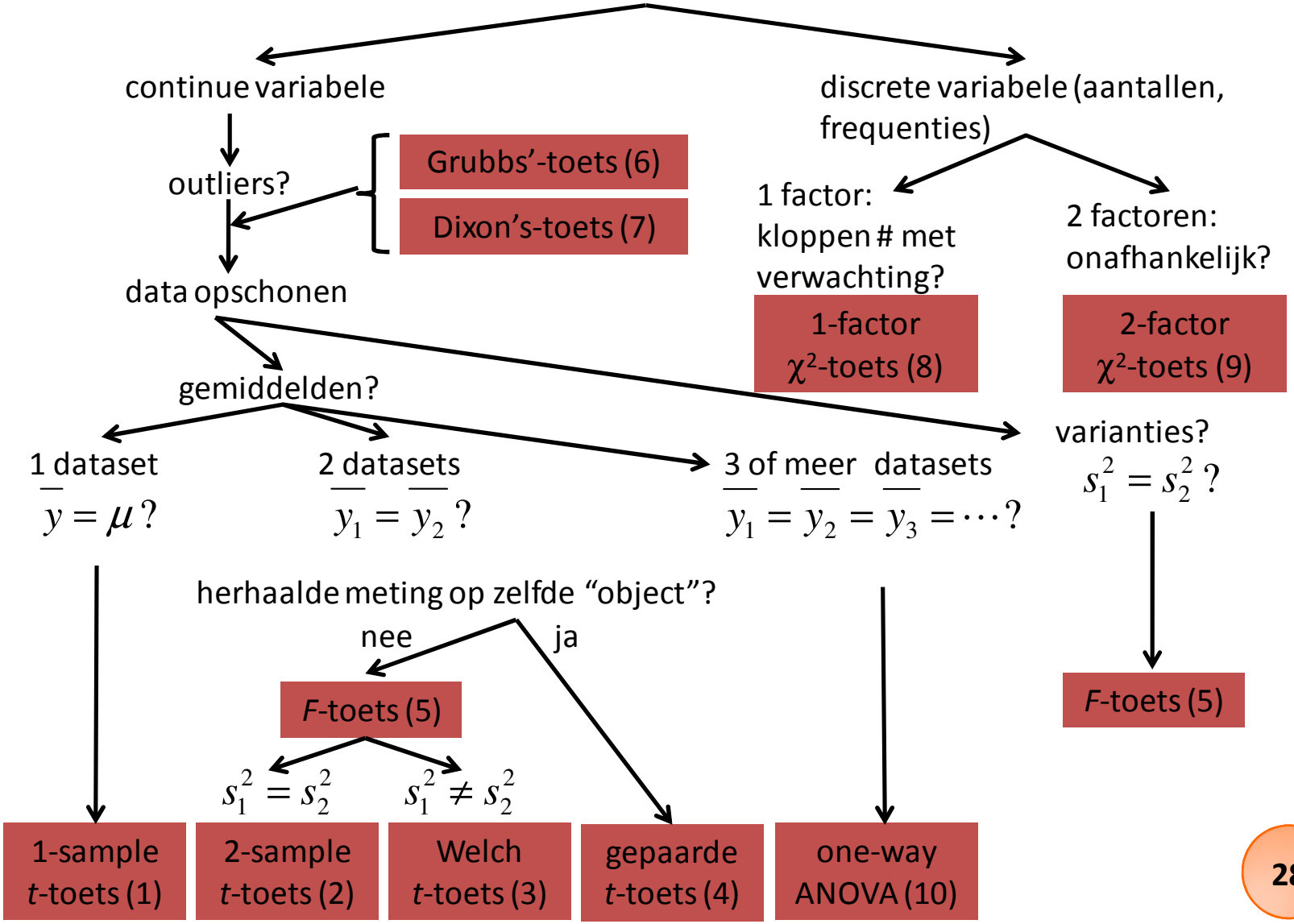
Je update dus ALLEEN de *M* (= *R*) = log(*T*) waarden, NIET de intensiteiten per channel!

DIFFERENTIALLY EXPRESSED GENES (DEGS)

- Genen met verschillende expressie tussen sample groep(en)
  - Bijv. gen X komt *meer tot expressie* bij gezonde mensen ten opzichten van mensen met kanker
  - Bijv. gen Y komt *minder tot expressie* bij vrouwen dan bij mannen

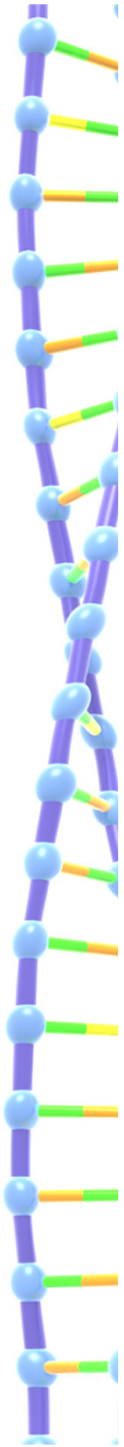


STATISTIEK 2: BESLISSCHEMA



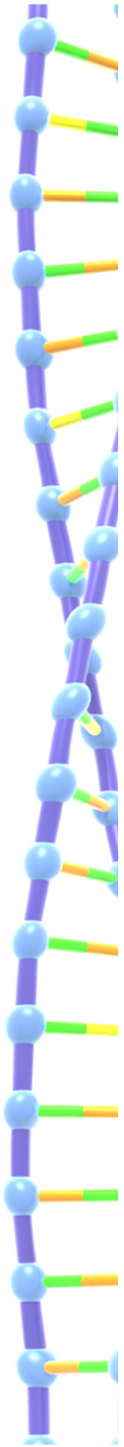
PARAMETRISCHE TOETSEN

- *t*-toets (1-sample/2-sample/Welch/gepaard)
  - **`t.test(y ~ x, …)`**
  - $p < \alpha$ (= 0.05) $\rightarrow$ significant verschil in gemiddelde tussen groepen gedefinieerd in **`x`** / significant effect van **`x`** op **`y`**
- *F*-toets
  - **`var.test(y ~ x)`**
  - $p < \alpha$ (= 0.05) $\rightarrow$ significant verschil in varianties tussen groepen gedefinieerd in **`x`** / significant effect van **`x`** op varianties van **`y`**



PARAMETRISCHE TOETSEN

- 1-way ANOVA
  - `aov(y ~ x)`
  - $p < \alpha$ (= 0.05) $\rightarrow$ significant verschil in gemiddelde tussen groepen gedefinieerd in **`x`** / significant effect van **`x`** op **`y`**
- Bartlett-toets
  - `bartlett.test(y ~ x)`
  - $p < \alpha$ (= 0.05) $\rightarrow$ significant verschil in varianties tussen groepen gedefinieerd in **`x`** / significant effect van **`x`** op varianties van **`y`**

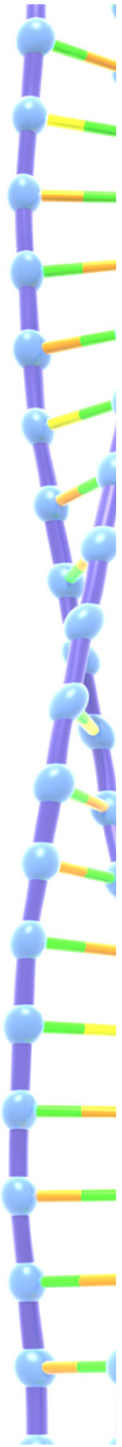


DIFFERENTIËLE GEN EXPRESSIES: 1 GEN, 2 SAMPLES

- Data format 1: r replica's van M' waarden per gen:

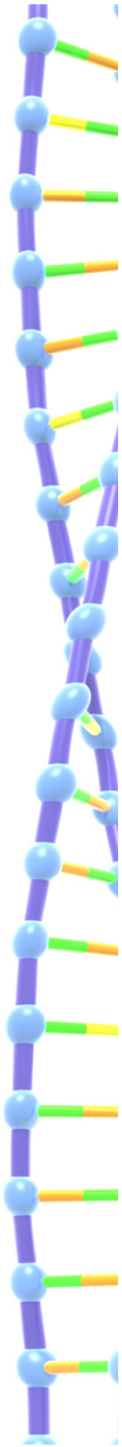
Gene	M'_1	M'_2	…	M'_i	…	M'_r
gene 1	0.51	0.34	…	0.55	…	0.44
gene 2	-0.14	-0.31	…	0.11	…	-0.27
…						
gene g	0.78	0.85	…	0.69	…	0.75
…						
gene G	1.15		…	0.66	…	0.91

1-sample *t*-toets voor μ = 0



(NIET-)PARAMETRISCHE TOETSEN

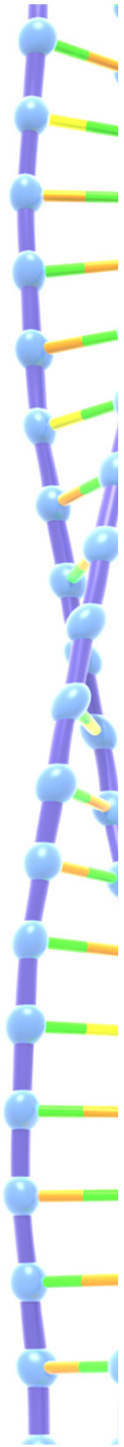
- Parametrische toetsen
  - Aanname: ruis = normaalverdeeld
  - Aanname: ruis gelijk voor verschillende groepen
  - Meer power (= 1 - β) als normaalverdeeld
- Niet-parametrische toetsen
  - Aanname: verdeling gelijk voor versch. groepen
  - Niet afhankelijk van normaalverdeling



(NIET-)PARAMETRISCHE TOETSEN

- Voor de meeste parametrische toetsen zijn ook niet-parametrische versies:

Data	Parametrisch	Niet-parametrisch
1 dataset	1-sample *t*-toets `t.test(y, mu=10)`	Wilcoxon signed-rank toets `wilcox.test(y, mu=10)`
2 datasets	2-sample/Welch *t*-toets `t.test(y ~ sample, var.eq=T/F)`	Wilcoxon rank-sum toets = Mann-Whitney toets `wilcox.test(y ~ sample)`
2 datasets, gepaard	gepaarde *t*-toets `t.test(y ~ sample, paired=T)`	Wilcoxon signed-rank toets `wilcox.test(y ~ sample, paired=T)`
>2 datasets	1-way ANOVA `aov(y ~ sample)`	Kruskal-Wallis rank-sum toets `kruskal.test(y ~ sample)`



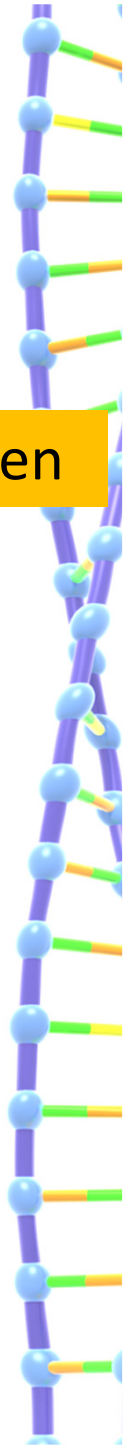
DIFFERENTIËLE GEN EXPRESSIES: 1 GEN, 2 SAMPLES

- Statistische analyse per gen op $M = \log(T)$ waarden:
  - 1-sample *t*-toets voor μ= 0
  - Wilcoxon signed-rank toets voor μ= 0

  } *p*-waarde per gen
- *p*-waarden per gen aanpassen voor multiple testing:
  - Holm methode (FWER)
  - Benjamini-Hochberg methode (FDR)
- Genen i met significante differentiële expressie:

$$p_{\text{adjust},i} < \alpha$$

- Doe vervolg analyse met deze significante DEGs:
  - clustering
  - …

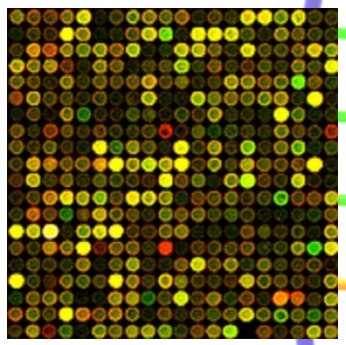


DIFFERENTIËLE GEN EXPRESSIES: 1 GEN, >2 SAMPLES

- Met >2 samples (bijv. controle, tumor stage I, tumor stage II) niet één enkele $M = \log(T)$ waarde…
- Mogelijke aanpak: bereken M-waarden t.o.v. gezamenlijk referentie sample, bijv. "gemiddelde" van samples (= mengsel van mRNA samples):
- Bijv. 4 channel array (Cy2, Cy3, Cy5, Cy7 labels)
  - Cy2 = controle, Cy3 = stage I, Cy5 = stage II, Cy7 = ref
  - Background correctie per channel
  - Berekenen M_{contr}, $M_{\text{stage I}}$ en $M_{\text{stage II}}$

$$M_{\text{contr}} = {}^2\log(\text{Cy2}/\text{Cy7}), \qquad M_{\text{stageI}} = {}^2\log(\text{Cy3}/\text{Cy7}), \ \ldots$$

  - Normalisatie voor controle, stage I en stage II



DIFFERENTIËLE GEN EXPRESSIES: 1 GEN, >2 SAMPLES

- Data format 1: k samples, r replica's j van $M_{ij} = \log(T_{ij}')$ waarden per gen per sample i :

	sample 1				sample k		
Gene	M'_{11}	…	M'_{1r}	…	M'_{k1}	…	M'_{kr}
gene 1	0.51	…	0.34	…	0.55	…	0.44
gene 2	-0.14	…	-0.31	…	0.11	…	-0.27
…							
gene g	0.78	…	0.85	…	0.69	…	0.75
…							
gene G	1.15	…	0.45	…	0.66	…	0.91

DIFFERENTIËLE GEN EXPRESSIES: 1 GEN, >2 SAMPLES

- Data format 1: k samples, r replica's j van $M_{ij} = \log(T_{ij}')$ waarden per gen per sample i :

	sample 1				sample k		
Gene	M'_{11}	…	M'_{1r}	…	M'_{k1}	…	M'_{kr}
gene 1	0.51	…	0.34	…	0.55	…	0.44
gene 2	-0.14	…	-0.31	…	0.11	…	-0.27
…							
gene g	0.78	…	0.85	…	0.69	…	0.75
…							
gene G					0.66	…	0.91

1-way ANOVA

DIFFERENTIËLE GEN EXPRESSIES: 1 GEN, >2 SAMPLES

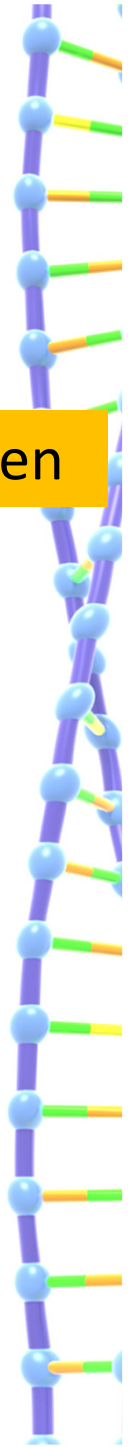
- Statistische analyse per gen op $M = \log(T)$ waarden:
  - 1-way ANOVA
  - Kruskal-Wallis rank-sum toets

} *p*-waarde per gen

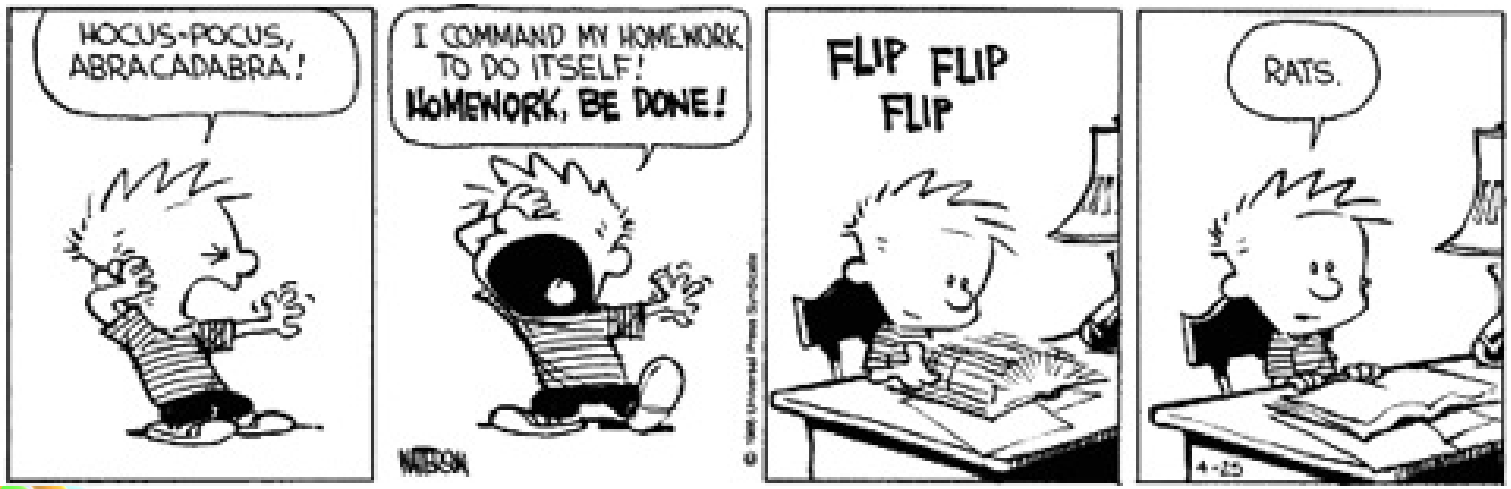
- *p*-waarden per gen aanpassen voor multiple testing:
  - Holm methode (FWER)
  - Benjamini-Hochberg methode (FDR)
- Genen i met significante differentiële expressie:

$$p_{\text{adjust},i} < \alpha$$

- Doe vervolg analyse met deze significante DEGs:
  - clustering
  - …



Jullie kunnen nu de opdrachten van les 14 maken



Hanze University Groningen
Applied Sciences
Institute for
Life Science & Technology