Microarray Analyse

Wamelen, J.J. van

Microarray Analyse

Bachelorscriptie - J.J. van Wamelen

Begeleider - Dr. E. van Zwet

Universiteit Leiden

Inleiding

Deze scriptie is geschreven door Jasper van Wamelen met als doel het behalen van de Bachelor wiskunde aan de universiteit Leiden. De scriptie is geschreven onder begeleiding van Erik van Zwet voor het wiskundige deel en Harald Mil voor de biologische aspecten. De gebruikte data is afkomstig van DSM.

In deze scriptie behandel ik, aan de hand van een biologisch experiment op een schimmel, de diverse aspecten die aan de orde zijn bij het verrichten van een microarray analyse. In de eerste hoofdstukken wordt ingegaan op de biologische kant van zowel het experiment als de microarray technologie. Na de biologische achtergrond volgt een hoofdstuk dat de wiskundige kanten van de microarray analyse behandelt. Dit hoofdstuk is echter niet noodzakelijk om de rest van de scriptie te begrijpen en kan eventueel door lezers met alleen kennis op het gebied van de biologie worden overgeslagen. In het laatste deel van de scriptie wordt de analyse van de data uit het experiment uitgevoerd en worden twee mogelijke verbeteringen aangedragen voor de 'standaard' analyse.

Van de lezer wordt verwacht enige kennis te hebben van de biologie (middelbare school niveau is voldoende) en bekend te zijn met enkele simpele begrippen uit de statistiek. De analyse van de data is gedaan met software pakketten bioconductor en R (versie 2.6.2).

Inhoudsopgave

Hoofdstuk 1

Het onderzoek

In dit hoofdstuk wordt kort ingegaan op het onderzoek aan de hand waarvan deze scriptie tot stand is gekomen. Het experiment is uitgevoerd in opdracht van DSM door een onderzoeksgroep binnen de faculteit biologie van de universiteit van Leiden. Na een schets van de biologische achtergrond en de motivatie van het experiment volgen een paar woorden over de schimmel Aspergillus Niger en hey antibioticum tunicamicyne. Voor een uitgebreide bespreking van het onderzoek verwijs ik naar het artikel van Thomas Guillemette et al. [1]

1.1 Achtergrond

Veel soorten draadvormige schimmels beschikken over een effectief secretie systeem. Dit hebben ze nodig om genoeg hydrolitische enymen te produceren, deze enzymen zijn nodig voor het leven op dood organisch materiaal. De secretie is zo effectief omdat de schimmels 'substraten' kunnen produceren uit organisch materiaal. Dit secretie systeem maakt draadvormige schimmels een goede kandidaat voor het industrieel produceren van bepaalde enzymen. Het blijkt echter dat de opbrengst van hetrologe eiwitten (eiwitten die niet natuurlijk door de schimmel worden gemaakt) lager is dan men zou willen. Dit is helemaal het geval als het donor organisme geen schimmel is.

Er is veel onderzoek gedaan naar dit probleem en meerdere studies wijzen in de richting van het secretie pad als mogelijke bottleneck. Kort door de bocht komt het er op neer dat eiwitten in de cel niet goed opgevouwen worden en hierdoor de cel niet uit kunnen. Doordat de eiwitten de cel niet uit kunnen raakt de cel gestrest. Als reactie op de stress zal de cel maatregelen gaan nemen om de eiwit vouwing weer in orde te krijgen. Er zijn een aantal processen die hierbij langs elkaar lopen maar de belangrijkste is de unfolded protein response (verder afgekort met UPR). Dit is ook de reactie die in het experiment zal worden opgewekt bij de schimmel.

Aspergillus niger

Schimmels die veel worden gebruikt in de industrie als zogenaamde celfabriek zijn de Aspergilli. De Aspergillusfamilie bestaat uit een 200 tal draadvormige schimmels die voornamelijk voorkomen op bedorven fruit en vochtige muren (de huis tuin en keukenschimmel).

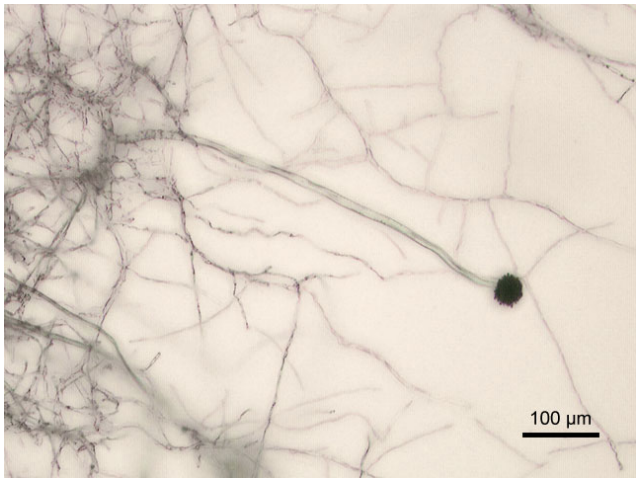
In het onderzoek zal de Aspergillus Niger centraal staan. Deze Aspergillus wordt in de industrie gebruikt (onder andere bij DSM) voor het produceren van citroenzuur (E330) en glucosezuur (E574).

Tunicamycin

Het doel van het onderzoek is het onderzoeken welke genen betrokken zijn bij de UPR in de Aspergillus Niger. Om hier achter te komen wordt tijdens het experiment tunicamycin toegediend aan de schimmel. Tunicamycin is een vorm van antibiotica, geproduceerd door de Streptomyces Iysosuperficus bacterie en het wordt vaak in experimenten gebruikt als opwekker van de UPR.

1.2 Het experiment

Om inzicht te krijgen in de genen die in de Aspergillus een rol spelen bij de UPR is er voor een microarray experiment gekozen. De volgende hoofdstukken gaan hier verder op in. Er is in het onderzoek gekozen om het experiment zes maal uit te voeren. Hierbij is er driemaal daadwerkelijk tunicamycin toegevoegd. De andere drie experimenten zijn gedaan als controle. Voor de gedetailleerde opzet van het experiment verwijs ik wederom naar het artikel van Thomas Guillemette et al. [1].



Aspergillus Niger

Hoofdstuk 2

Microarry's

Zoals in het vorige hoofdstuk is aangegeven, is bij het onderzoek naar de UPR van de Aspergillus gebruik gemaakt van een zestal microarray experimenten. Dit hoofdstuk geeft een korte introductie naar deze onderzoeksmethode die de laatste jaren steeds populairder wordt binnen de experimentele biologie. Microarrays zijn er in vele soorten en maten. In het onderzoek is gebruik gemaakt van zogenaamde High-density Oligonucleotide arrays. Het onderstaande is dan ook vooral van toepassing op deze arrays. Voor andere arrays verwijs ik naar het boek van Dov Stekel. [2]

2.1 Microarray's: een inleiding

In veel biologische onderzoeken is men genteresseerd in de reactie van een organisme op een bepaalde stof. Deze reactie kunnen we aflezen door te kijken naar de eiwit productie in een cel. We zouden dus willen testen welke eiwitten na het experiment extra worden aangemaakt, of juist niet meer worden geproduceerd. Om dit voor elkaar te krijgen maken we gebruik van het proces in de cel dat de aanmaak van eiwitten regelt.

Belangrijk voor het maken van een eiwit is het zogenaamde mRNA. mRNA fungeert als 'boodschapper' (messenger) die twee processen met elkaar verbindt: de transcriptie, waarbij een stuk DNA (een gen) overgeschreven wordt tot mRNA, en de translatie, waarbij het mRNA wordt vertaald naar een keten van aminozuren (een eiwit).

Schematisch:

$$DNA \rightarrow transcriptie \rightarrow mRNA \rightarrow translatie \rightarrow eiwit \tag{2.1}$$

Een microarray (ook wel DNA-microarray) bestaat uit een plaat (chip) met daar op een grote hoeveelheid 'spots'. Afhankelijk van het type array (daar over zo meer) correspondeert elke spot op de chip met een probe. Een probe is een stuk inverse mRNA van ongeveer 25 base paren lang. De probes corresponderen op deze manier met een bepaald gen. Meestal worden er 11 tot 20 probes (verspreid over de hele chip) per gen gebruikt. Aan de hand van de kleuring van de spots /probes is uiteindelijk af te lezen in welke mate een bepaald gen tot expressie komt.

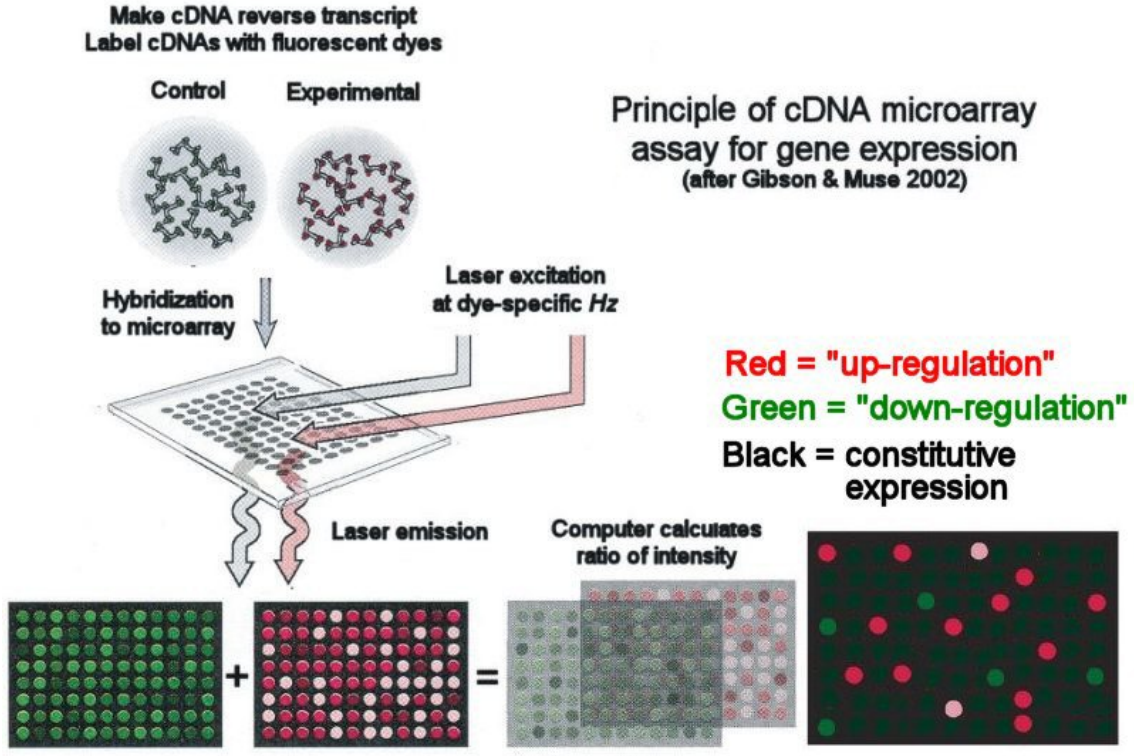
Bij een microarray analyse wordt het mRNA uit een test sample en een control sample gextraheerd en beide worden door inverse transcriptie omgezet in cDNA (copy- of complementDNA). Het is dit cDNA dat uiteindelijk op de microarray chip zal belanden. Door het cDNA van de test en van de controle groep te kleuren zal het verschil in expressie op de chip zichtbaar zijn. Voor de kleuring wordt (wederom voor een deel afhankelijk van de soort microarray chip) gebruik gemaakt van Cy_3 en Cy_5 moleculen die zich kunnen binden aan het cDNA. De speciale eigenschap van deze twee moleculen is dat ze beide onder invloed van een bepaalde laser sterk fluoriseren. De control sample wordt op deze manier groen (dvm Cy_3) en de test sample rood (dvm Cy_5).

Beide gekleurde cDNA-producten worden nu toegevoegd op de microarray chip. Het cDNA zal zich binden op de spots die een complementaire sequentie hebben (de kleuring speelt hier geen rol). Na een tijd worden de niet gebonden cDNA moleculen van de array afgespoeld en houden we een gekleurde chip over. Voor de diverse spots zijn er nu een aantal mogelijkheden;

groen	*Het gen komt alleen tot expressie in de controle.*
rood	*Het gen komt alleen tot expressie in de test.*
zwart	*Het gen komt voor zowel in de test als in de controle.*

In de praktijk hebben spots natuurlijk een kleur die ergens tussen de bovenstaande kleuren in ligt. Bij het doen van een microarray experiment is het dan ook de kunst om die genen te vinden die significant tot expressie komen. Met andere woorden, waarvan de kleur erg groen of erg rood is. Daarom is het gebruikelijk om voor elk gen niet 1 spot te nemen maar (afhankelijk van het experiment) wel tot 20 zogenaamde probes per gen te hebben, dit om om de mogelijke ruis te corrigeren in de productie van de array.

De volgende afbeelding geeft het hele microarray proces schematisch weer;



2.2 Affymetrix

Het uitvoeren van een microarray experiment is niet eenvoudig. De Aspergillus in ons experiment heeft ongeveer 14500 genen, om al deze genen op de chip te krijgen, zonder dat er het een en ander door elkaar loopt is een extreem delicate procedure. Een microarray experiment zoals het bij ons onderzoek wordt gedaan wordt dan ook uitbesteed aan gespecialiseerde bedrijven. Marktleider op dit gebied is Affymetrix en het Aspergillus microarray experiment is dan ook door dit bedrijf uitgevoerd. Affymetrix maakt gebruik van het 'standaard' microarray principe maar wel met een aanpassing.

Affymetrix gebruikt per gen twee soorten probes, perfect match probes (pm) en miss match probes(mm). De perfectmatch probes bevatten de complementaire sequentie van het bij het gen behorende cDNA. De lengte van deze probes is 25 base paren. Door de korte lengte van de probes is het noodzakelijk om meerdere probe paren per gen te hebben. De pm probes zijn ontworpen om alleen de transcripten van het specifieke gen aan zich te binden, in de praktijk is het echter niet te voorkomen dat ook andere moleculen zich binden. Om dit te compenseren is de mm probe. Deze probe is ontworpen om juist de niet specifieke binding van cDNA te meten. De pm en mm probes verschillen dan ook niet veel, ze zijn zelfs bijna identiek in de mm is alleen het 13e base paar vervangen door het complement van het base paar in de pm probe. De pm probes die gebruikt worden zijn niet allemaal identiek. Het mRNA van een gen is een stuk langer dan 25 base paren. Als pm probes worden dan ook verschillende stukjes inverse transcriptie gebruikt.

Er zitten een aantal voordelen aan het gebruik van Affymetrix chips. Door het gebruik van veel probes verspreid over de chip worden lokale ruis effecten opgevangen. Daarnaast leveren de miss match probes een hoop informatie over de 'fout' gebonden cDNA moleculen. Elk voordeel heeft natuurlijk ook een nadeel. De oorspronkelijke opzet van de perfect match en miss match probes blijkt in de praktijk problemen te geven. Affymetrix stelde als probe intensiteit voor om $Y_{probe} = Y_{pm} - Y_{mm}$ te gebruiken. Dit lijkt een logische keuze, het komt echter vaak voor dat Y_{mm} groter is dan de Y_{pm} waardoor er negatieve probe intensiteiten ontstaan. Daarnaast is er het zogenaamde 3'/5' effect. Probes die op het mRNA dichter bij het 3' eind liggen binden makkelijker Cy_3 en probes aan de 5' kant hebben een voorkeur voor Cy_5.

Twee Affymetrix chips

2.3 Output van een microarray experiment

De vorige paragrafen beschrijven in het kort hoe een microarray experiment uitgevoerd wordt. Maar met alleen een gekleurde chip zijn we nog nergens. De eerste stap in het analyseproces betreft het omzetten van de kleuringen op de chips in data. Dit is een delicate procedure, die door Affymetrix zelf uitgevoerd en aangeleverd in speciale .Dat files. Deze files bevatten per probe een rood en een groen intensiteit. Omdat de .dat files alleen geschikt zijn voor software van Affymetrix zelf, zullen we deze buiten beschouwing laten. De analyse in de volgende hoofdstukken zal beginnen vanaf de .CEL files die ook worden aangeleverd door Affymetrix, deze files bevatten per probe slechts een intensiteit. $intensiteit_{probe} = log_2(groenintensiteit_{probe}) - log_2(roodintensiteit_{probe})$.

Hoofdstuk 3

Wiskundige achtergrond

Voordat we kunnen beginnen aan de analyse van de microarray data uit het onderzoek is het noodzakelijk enkele wiskundige begrippen behandeld te hebben. De eerste paragraaf van dit hoofdstuk zal in gaan op de moeilijkheden die ontstaan bij het testen van meerdere (veel) nulhypothese. In paragraaf twee zullen we een aantal aannames doen omtrent de data geproduceerd in het experiment en een lineair model opstellen voor de data.

3.1 Multiple testing

Stel we willen een $n - tal$ nul hypotheses testen, bijvoorbeeld,

$$H_{0,1}, ..., H_{0,n} \quad H_{0,i} = gen\ i\ komt\ niet\ tot\ expressie$$

En we willen dit zo doen dat we het aantal type 1 fouten onder controle willen houden. Met andere woorden, we willen de kans dat er een of meer nulhypothese onterecht worden verworpen kleiner houden dan een niveau α. we noemen dit de FWER (family wise error rate). Laat

$$R_i = \{\ verwerp\ H_{0,i}\}\ en\ \mathbb{P}_{H_{0,i}}(R_i) = \alpha_i$$

.

We defineren nu twee versies van de FWER op niveau α.

Definitie 3.1 (zwakke FWER)**.**

$$\mathbb{P}_{\bigcap_{i=1}^{n} H_{0,i}}(\cup_{i=1}^{n} R_i) \leqslant \alpha$$

Definitie 3.2 (sterke FWER)**.**

$$\mathbb{P}_{\bigcup_{i\in I} H_{0,i}}(\cup_{i\in I} R_i) \leqslant \alpha \qquad \forall\ I \subseteq \{1, 2, .., n\}$$

Het verschil tussen 3.1 en 3.2 is duidelijk. De zwakke variant houdt alleen in de hele verzameling nulhypothese de FWER onder controle waar de sterke variant ook in iedere deelverzameling de FWER kleiner dan α houdt.

Dat het wel degelijk nodig in om de FWER onder controle te houden kunnen we zien aan het volgende voorbeeld; in het experiment hebben we 14500 genen waarvan we willen

weten of ze significant tot expressie komen. We willen de volgende nulhypothese testen, $H_{0,i} = \{$ gen i komt niet tot expressie $\}$. Stel dat we α_i voor elk gen gelijk kiezen aan 0.05 dan vinden we het volgende voor de zwakke FWER;

$$\mathbb{P}_{\bigcup_{i=1}^{14500} H_{0,i}}(\cup_{i=1}^{n} R_i) = 1 - \prod_{i=1}^{14500} (1 - \mathbb{P}_{H_{0,i}}(R_i)) = 1 - \prod_{i=1}^{14500} (1 - \alpha_i)$$

Met $\alpha_i = 0.05$ zien we dat de FWER $1 - (1 - \alpha)^{14500} \approx 1$. Dat wil dus zeggen dat we bijna met kans 1 tenminste een gen foutief significant verklaren.

Alle hypotheses tegen het zelfde significantie niveau testen werkt niet. Er moet dus een correctie komen voor het testen van meerdere hypotheses. Er zijn een aantal methodes die α_i voor een gen i zo kiezen dat er aan 3.1 respectivelijk 3.2 wordt voldaan. We geven eerst twee methodes die de sterke FWER onder controle houden, daarna bekijken we de methode van Benjamini Hochberg die een andere oplossing geeft om de FWER te controleren.

3.1.1 Bonferroni

De eerste methode om de FWER onder controle te houden is van Bonferroni. Het is tevens de meest conservatieve. Bonferroni's methode schaalt het significantie niveau voor een hypothese simpel weg door het gewenste niveau α te delen door het aantal hypotheses;

Definitie 3.3 (Methode van Bonferroni)**.** Stel we testen $H_{0,1}, H_{0,2}, .., H_{0,n}$ nulhypotheses met corresponderende p-waardes $\pi_1, \pi_2, .., \pi_n$. Sorteer de p-waardes $\pi_{(1)} \leq \pi_{(2)} \leq .. \leq \pi_{(n)}$ met $H_{(i)}$ de nulhypothese bij $\pi_{(i)}$. We verwerpen $H_{(i)}$ als $\pi_{(i)} \leq \frac{\alpha}{n}$.

Stelling 3.4. *Bonferroni's methode houdt sterke controle over de FWER.*

Bewijs. Het volgende 1 regel bewijs toont aan dat Bonferroni's methode de FWER onder controle houdt.

$$\mathbb{P}_{\bigcup_{i=1}^{n} H_{0,i}}(\cup_{i=1}^{n} R_i) \leq \sum_{i=1}^{n} \mathbb{P}_{H_{0,i}}(R_i) = \sum_{i=1}^{n} a_i = \sum_{i=1}^{n} \frac{\alpha}{n} = \alpha \quad \forall n$$

QED

Het nadeel aan deze methode is dat er bijna geen onderscheidend vermogen overblijft als we een groot aantal nulhypotheses hebben. Als we weer naar ons experiment met 14500 hypotheses kijken dan moeten we $\alpha_i = 3.45 * 10^{-6}$ kiezen om aan een significatie niveau van $\alpha = 0.05$ te komen. Bonferroni's methode wordt om deze reden dan ook niet veel gebruikt.

3.1.2 Holm Bonferroni

Een tweede reden waarom de methode van Bonferroni niet veel gebruikt wordt is het feit dat de methode van Holm-Bonferroni (HB) beter is. Bij de HB methode wordt begonnen met het sorteren van de p-waardes corresponderend met de verschillende nulhypotheses. Vervolgens verwerpen we alle nulhypotheses vanaf de eerste hypothese waarvoor de p-waarde groter is dan $\frac{\alpha}{n-i+1}$

Definitie 3.5 (Methode van Holm-bonferroni)**.** Stel we testen $H_{0,1}, H_{0,2}, .., H_{0,n}$ nulhypotheses met corresponderende p-waardes $\pi_1, \pi_2, .., \pi_n$. Sorteer de p-waardes $\pi_{(1)} \leq \pi_{(2)} \leq .. \leq \pi_{(n)}$ met $H_{(i)}$ de nulhypothese bij $\pi_{(i)}$. We verwerpen $H_{(i)}$ $i = 1, 2, .., k-1$, voor k de kleinste index zdd $\pi_{(i)} > \frac{\alpha}{n-i+1}$.

Stelling 3.6. *Holms methode houdt sterke controle over de FWER en is meer onderscheidend dan Bonferroni.*

Bewijs. Als Bonferroni verwerpt dan verwerpt Holm-Bonferroni ook. Immers

$$\pi_i \leq \frac{\alpha}{n} \;\Rightarrow\; \pi_1 \leq \frac{\alpha}{n} = \frac{\alpha}{n-1+1} \;\Rightarrow ... \Rightarrow\; \pi_i \leq \frac{\alpha}{n-i+1}$$

.

Stel nu dat alle $H_{0,i}$ waar zijn, de kans dat we tenminste een nulhypothese verwerpen is nu

$$\mathbb{P}_{\cap_{i=1}^{N} H_{0,i}}(\pi_1 < \frac{\alpha}{n}) = 1 - (1 - \frac{\alpha}{n})^n \leq \alpha$$

.

Dit impliceert zwakke controle van de FWER. Stel nu dat $k < n$ nulhypotheses waar zijn. We kunnen nu alleen verwerpen als het minimum van de k p-waardes kleiner is dan $\frac{\alpha}{m-(m-k+1)+1}$ en dit is precies$\frac{\alpha}{k}$. Dus we hebben ook sterke controle van de FWER.

QED

We zien dat HB sterke controle houdt over de FWER maar dat de α_i steeds groter worden. HB heeft dus een groter onderscheidend vermogen dan Bonferroni. De methode van BH is zelfs nog iets te optimaliseren door de $\alpha_i's$ nog slimmer te kiezen (Hochberg, 1986).

3.1.3 Hochberg Benjamini

De laatste methode die we zullen bespreken is die van Benjamini en Hochberg (BH). In plaats van de FWER te willen controleren stelden zij een nieuwe vorm van controleerbaarheid voor: De False Discovery Rate. Deze zullen we nu defineren.

Als we n nulhypotheses testen kunnen we de volgende tabel maken van het aantal fouten dat wordt gemaakt.

	verwerp niet	verwerp	Σ
H_0 waar	U	V	n_0
H_0 niet waar	T	S	$n - n_0$
Σ	$R - n$	R	n

In deze tabel geeft S het aantal type 1 fouten weer, met andere woorden het controleren van de FWER komt neer op het controleren van S. Benjamini en Hochberg stelden dat het, zeker in microarray experimenten, niet zo erg is om een of twee type 1 fouten te maken maar dat het belangrijker is het aantal foute verwerpingen per het totale aantal verwerpingen te controleren, dit noemen we de FDR of False Discovery Rate. Controle van de FDR impliceert zwakke controle van de FWER en is meer onderscheidend dan Holm-Benjamini.

Definitie 3.7. Het deel van de fouten gemaakt door het foutief verwerpen van nulhypotheses kunnen we zien als een random variabele $Q = \frac{V}{V+S}$. (NB, als $V + S = 0$ defineren we $Q = 0$.) De FDR (false discovery rate) defineren we als $Q_\epsilon = \mathbb{E}(Q) = \mathbb{E}(\frac{V}{V+S})$

Definitie 3.8 (Methode van Benjamini-Hochberg)**.** Stel we testen $H_{0,1}, H_{0,2}, .., H_{0,n}$ nulhypotheses met corresponderende p-waardes $\pi_1, \pi_2, .., \pi_n$. Sorteer de p-waardes $\pi_{(1)} \leq \pi_{(2)} \leq .. \leq \pi_{(n)}$ met $H_{(i)}$ de nulhypothese bij $\pi_{(i)}$. Verwerp alle $H_{(i)}$ $i = 1, 2, .., k$, voor k de grootste index waarvoor $\pi_{(i)} \leq \frac{i}{n} q^*$.

Stelling 3.9. *Stelling; Voor onafhankelijke toetsen controleert de methode van Benjamini-Hochberg de FDR op niveau q^*.*

Het bewijs zal volgen uit het volgende lemma.

Lemma 3.10. *Voor elke $0 \leq n_0 \leq n$ onafhankelijke p-waardes behorende bij ware nulhypotheses en voor elke waardes die de $n - n_0$ p-waardes behorende bij de onware nulhypotheses kunnen aannemen, geldt voor de methode van Benjamini-Hochberg de volgende ongelijkheid;*

$$\mathbb{E}(Q|\pi_{(n_0+1)} = \pi_1, ..., \pi_{(n)} = \pi_{n_1}) \leq \frac{n_0}{n} q^* \tag{3.1}$$

Bewijs. Het bewijs van dit lemma gaat met inductie naar n, voor $n = 1$ is het direct duidelijk maar de inductie stap vereist enig werk. Daarom verwijs ik naar appendix A van het oorspronkelijke artikel door Benjamini en Hochberg [3] voor het gehele bewijs. QED

Bewijs stelling 3.9. Stel dat we $n_1 = n - n_0$ onware nulhypotheses hebben. Ongeacht de distributie van de p-waardes behorende bij deze onware hypotheses krijgen we na het integreren van vergelijking 3.1 de volgende uitdrukking;

$$\mathbb{E}(Q) \leq \frac{n_0}{n} q^* \leq q^* \tag{3.2}$$

QED

Het volgende lemma laat zien waarom het nuttig is om te kijken naar de FDR in plaats van de FWER.

Lemma 3.11. *Controle van de FDR $\Rightarrow$ zwakke controle van de FWER. En controle van de FDR is zwakker dan de controle van de FWER*

Bewijs. Stel we testen $H_{0,1}, H_{0,2}, .., H_{0,n}$ nulhypotheses. Als alle hypotheses waar zijn dan geldt in tabel 1 dat $S = 0$ dus $V = R$ en $\frac{V}{R} = 1$ als $V \geq 1$. Dus $\mathbb{P}(V \geq 1) = \mathbb{E}(\frac{V}{R})$. Nu geldt,

$$\mathbb{E}\left(\frac{V}{R}\right) \leq \alpha \Rightarrow \mathbb{P}(V \geq 1) \leq \alpha \Rightarrow zwakke\ FWER$$

Stel nu dat niet alle hypotheses waar zijn maw, $n_0 < n$ dan is $\frac{V}{R} \leq 1$ als $V \geq 1$, dus we hebben $\mathbb{P}(V \geq 1) \geq \mathbb{E}(\frac{V}{R})$ QED

Van de drie methodes die we hier beschrijven zien we dat Benjamini-Hochberg de minste power verliest maar nog wel controle houdt over de zwakke FWER. Bij de analyse van de data uit het experiment zullen we dan ook deze methode gebruiken om te corrigeren voor het testen van meerdere nulhypotheses.

3.2 Moddel fitting

Voor we de data uit het onderzoek kunnen analyseren zullen we een aantal aannames moeten doen over de data. In deze scriptie zullen we er van uit gaan dat de microarray data te beschrijven is met een simpel lineair model. De onderstaande paragraaf geeft een korte samenvatting. Deze samenvatting is gebaseerd op het artikel van Smyth. [4]. Dit artikel geeft een lineair model voor microarray data en een empirische baysiaanse methode voor het vinden van significante genen.

We nemen aan dat we n microarray chips hebben en $Y_g^T = (y_{g1}, .., y_{gn})$ een vector met $y_{gi} = log_2(R_{gi}) - log_2(G_{gi})$ waar R_{gi} en G_{gi} de respectivelijk rood en groen intensiteit zijn voor een gen g op chip i. In het geval van High-density Oligonucleotide chips nemen we aan dat de probes genormaliseerd tot een enkele expressie Y_{gi}. Voor Y_g nemen we het volgende aan;

$$\mathbb{E}(Y_g) = X\alpha_g$$

X is hier de ontwerp matrix en α_g is een vector met cofficinten. Verder nemen we aan dat;

$$cov(Y_g) = W_g\sigma_g^2$$

Met W_g een bekende niet negatieve gewichtsmatrix. (W_g is niet strikt positief om dat er in y_g waardes kunnen missen, in welk geval de corresponderende gewichten in W_g nul zijn).

In de analyse van de microarray data zijn we genteresseerd in contrasten tussen de verschillende chips, bijvoorbeeld test vs. controle. Deze contrasten definieren we met;

$$\beta_g = C^T\alpha_g$$

Hier is C^T de zogenaamde contrast matrix. Stel we hebben een eenvoudig microarray experiment met een controle en een test groep. De contrast matrix wordt in dat geval, $(1, -1)^T$, of duidelijker $test - controle$. Door de keuze van β_g is het interessant als $\beta_g = 0$. In dit geval komt gen g niet verschillend tot expressie.

We nemen aan dat het lineare model de volgende schatters oplevert, $\hat{\alpha}_g$ voor de cofficinten α_g, s_g^2 voor σ_g^2 en een covariantie matrix;

$$cov(\hat{\alpha}_g) = V_g s_g^2.$$

V_g is hier een bekende positief definite matrix die niet afhangt van s_g^2. De schatters voor de contrasten nemen we $\hat{B}_g = C^T\hat{\alpha}_g$ met covariantie matrix;

$$cov(\hat{\beta}_g) = C^T V_g C s_g^2$$

De expressies y_g hoeven niet noodzakelijk normaal verdeeld te zijn en het lineaire model hoeft dan ook niet verkregen te zijn dmv de kleinste kwadraten methode. Van de contrasten nemen we wel aan dat ze normaal verdeeld zijn met gemiddelde β_g en covariantie matrix $C^T V_g C\sigma_g^2$. Neem v_{gj} het $j$de element van $C^T V_g C$ dan nemen we aan dat $\hat{\beta}_{gj}$ en s_g^2 als volgt verdeeld zijn.

$$\hat{\beta}_{gj}|\beta_{gj}, \sigma_g^2 \sim N(\beta_{gj}, v_{gj}\sigma_g^2)$$

$$s_g^2|\sigma_g^2 \sim \frac{\sigma_g^2}{d_g}\chi^2_{d_g}$$

Met deze aannames vinden we de gebruikelijke t-statistiek;

$$t_{gj} = \frac{\hat{\beta}_{gj}}{s_g\sqrt{v_{gj}}}$$

De strategie die we zullen toepassen bij het vinden van significante genen is het testen van de nulhypothese $H_0 : \beta_{gj} = 0$. Het is dus belangrijk de onbekende coefficienten β_{gj} en variantie σ_g^2 te beschrijven. We doen dit met een empirische baysiaanse methode. We nemen hiervoor de volgende prior verdelingen aan voor β_{gj} en σ_g^2. Voor σ_g^2 nemen we dezelfde prior als voor s_g^2 met d_0 vrijheidsgraden;

$$\frac{1}{\sigma_g^2} = \frac{1}{d_0 s_g^2}\chi^2_{d_0}$$

Voor een gegeven j nemen we aan dat een β_{gj} niet nul is met een bekende kans p_j. (in de praktijk wordt hier een kans van 0.01 aangehouden)

$$\mathbb{P}(\beta_{gj} \neq 0) = p_j$$

Nu is p_j de verwachte proportie van het aantal genen dat verschillend tot expressie komt. Voor alle genen met β_{gj} ongelijk aan nul nemen we aan dat de a priori verdeling normaal is met verwachting nul en variantie $v_0 j$ met andere woorden,

$$\beta_{gj}|\sigma_g^2, \beta_{gj} \neq 0 \sim N(0, v_{0j}\sigma_g^2)$$

Dit geeft de verwachte verdeling van de verandering op log-schaal van genen die tot expressie komen. In de analyse van de microarray data van ons experiment zullen we uiteindelijk de β statistiek gebruiken als rangschikking voor de meest significantie expressies. Voor een meer gedetailleerde beschrijving van het bovenstaande en een beschrijving voor het schatten van de diverse parameters verwijs ik naar het artikel van Smythe [4]. of een eerder werk van Lönnstadt en Speed [5].

Hoofdstuk 4

Microarray analyse

In dit hoofdstuk zullen we kijken naar de analyse van het microarray experiment. Zoals al is aangegeven werkt het experiment met chips van Affimetrix. Ik beperk me hier dan ook toe, wat betreft bespreking van de analyse. Voor een meer uitgebreide uitleg over de diverse microarrays verwijs ik naar Dov Stekel [2].

Bioconductor

Er is veel software beschikbaar voor het doen van microarray analyses. Naast de dure commercile pakketen is bioconductor daar van de meest gebruikte. Bioconductor is een package voor het opensource statistiek programma R en is gratis te downloaden op www.bioconductor.org. Door de opensource structuur zijn er veel specifieke packages te downloaden voor bioconductor. Voor een uitgebreide uitleg van microarray analyse met bioconductor verwijs ik naar het boek van Robert Gentelman et al. [6].

4.1 Inlezen van de data

De eerste handeling die gedaan moet worden bij een microarray analyse is het correct inlezen van de data. Zoals al in 2.3 is aangegeven beginnen we de analyse vanaf de .CEL files aangeleverd door Affymetrix. In mijn analyse heb ik gebruik gemaakt van het bioconductor package *affy*, dit is een package dat speciaal is ontworpen om Affymetrix data te kunnen analyseren. De files worden per chip ingelezen en in een Affybatch data object gestopt.

Naast het inlezen van de data is het belangrijk de juiste annotatie (naamgeving voor de genen) toe te voegen. Na de analyse moet er immers een lijst met significante genen uit komen. Annotatie bleek een groot struikelblok in de gehele analyse (daarover later meer). Voor veel organismes is de annotatie zo goed als bekend en zijn er bioconductor packages die de annotatie verzorgen. In het geval van de Aspergillus Niger is dat helaas niet het geval. Onze annotatie komt van DSM en gelukkig bevat het package *makecdfenv* de mogelijkheid een eigen annotatie toe te voegen aan een Affybatch object. De volgende code kan worden gebruikt om de data sets en annotatie in te lezen in R;

>setwd(*dir waar de data files staan*) >library(affy)

```
>library(makecdfenv)
>
>Data← ReadAffy()
>dsmmanigeraancoll ← make.cdf.env("dsmmanigeraancollcdf.cdf")
>Data@cdfName←"dsmmanigeraancoll"
```

Als we nu >Data in typen krijgen we een overzicht van de ingelezen data,

AffyBatch object
size of arrays=712×712 features (9 kb)
cdf=dsmmanigeraancoll (14554 affyids)
number of samples=6
number of genes=14554
annotation=dsmmanigeraancoll
notes=

4.2 Normalisatie en data kwaliteit

Voordat we met de echte analyse van de data kunnen beginnen zullen we de ruwe data van het experiment eerst moeten normaliseren en moeten controleren of er geen rare afwijkingen in de data zitten. Wederom is er een hoop software beschikbaar om deze stappen uit te voeren. We beginnen met een kleine opmerking over de normalisatie en zullen daarna iets dieper in gaan op de kwaliteitsanalyse van onze data.

Normalisatie

Normalisatie van microarray data is een onderwerp waar met gemak een hele scriptie (en misschien wel meer) aan te wijden is. Het proces van experiment naar data bevat een heleboel nauwkeurige handelingen waar (zeker op de kleine schaal van de chips) gemakkelijk ruis kan optreden. Naast deze ruisfactor zijn er nog tal van andere 'problemen' om rekening mee te houden. Een van de vele voorbeelden hier van is het zogenaamde 'banaan' effect wat er op neer komt dat de Cy_5 beter bindt aan probes aan de 3'kant van het mRNA en Cy_3 juist beter aan probes van het andere eind. Voor een uitgebreide bespreking van normalisatie verwijs ik naar Dov Stekel [2] hoofdstuk 5.

In mijn analyse gebruik ik als normalisatie *rma* uit het bioconductor package *limma*. Dit script maakt geen gebruik van de MM probes, dit omdat in veel gevallen de MM waardes hoger zijn dan de PM intensiteiten. De oorspronkelijk door Affymetrix beoogde normalisatie *Mas5.0 background* trekt de MM waardes van de PM waardes af. In ons geval leidt dit dus tot negatieve intensiteiten wat het gebruik van logaritmen onmogelijk maakt. De volgende code kan worden gebruikt om de Affybatch te normaliseren waardoor deze wordt omgezet in een ExpressionSet.

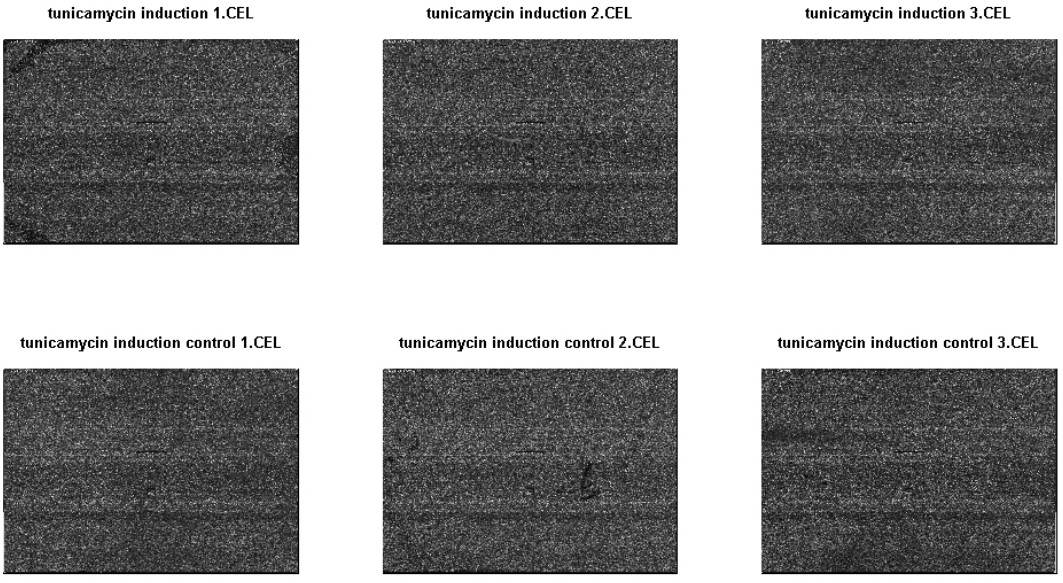
```
>library(limma)
>eset ← rma(Data)
```

>eset
ExpressionSet (storageMode: lockedEnvironment)
assayData: 14554 features, 6 samples
element names: exprs
phenoData
sampleNames: tunicamycin induction 1.CEL, tunicamycin induction 2.CEL, ..., tunicamycin induction control 3.CEL (6 total)
varLabels and varMetadata description:
sample: arbitrary numbering
featureData
featureNames: $AFFX-BioB-3_{at}$, $AFFX-BioB-5_{at}$, ..., $An40h00100_{at}$ *(14554 total)*
fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
Annotation: dsmmanigeraancoll

data kwaliteit controle

Voor het doen van een microarray analyse is het belangrijk dat de data van het experiment geen rare waardes bevat. Het kan daarom geen kwaad om voor het toepassen van normalisatie een aantal plots van de verschillende chips te bekijken. In het bioconductor pakket *affy* zijn een aantal functies voor het maken van data kwaliteit plots. Als eerste kijken we naar een plot van de probe intensiteit zoals de probes op de chip liggen;

>par(mfrow=c(2,3))
>image(Data)



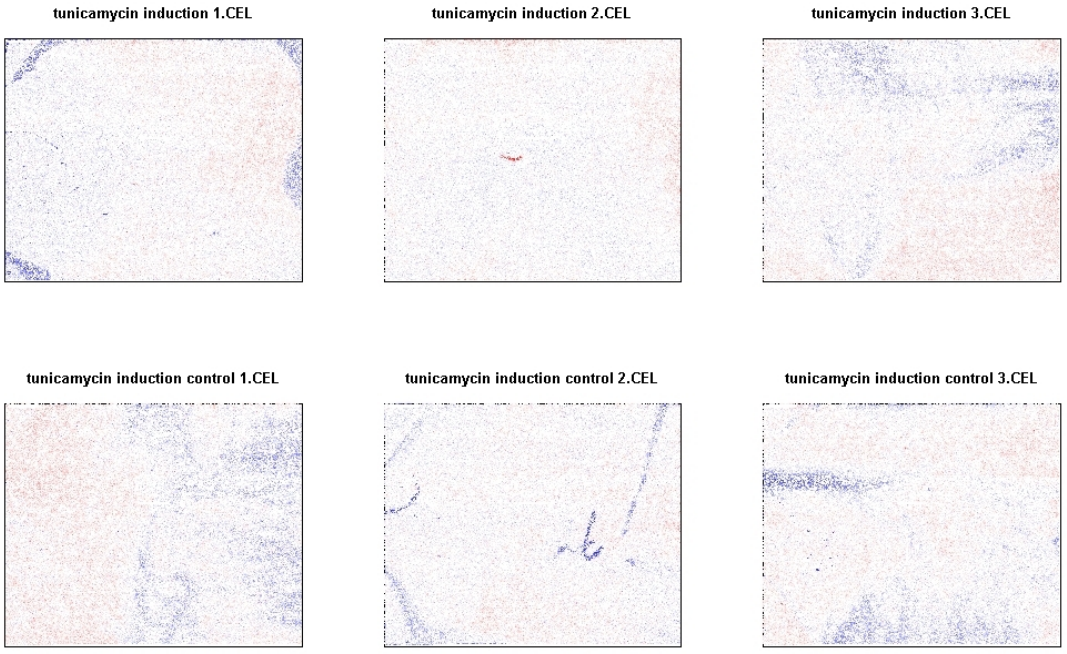
Dit geeft van elk van de zes chips de log intensiteit van de probes op de chips weer. Een aantal dingen vallen op. Op de eerste chip zijn duidelijk twee donkere strepen te zien (links boven en links onder). Daarnaast heeft chip 5 een merkwaardig donker patroon rechts in

het midden. We kunnen deze artefacten duidelijker zichtbaar maken door een zogenaamd Probe Level Model (PLM) op de data te passen. PLM neemt het volgende lineaire model aan voor genormaliseerde probe intensiteiten ($log(Y_{gij})$;

$$log(Y_{gij}) = \Theta_{gi} + \phi_{gj} + \epsilon_{gij}$$

Hier is Θ_{gi} de log intensiteit van gen g op array i, ϕ_{gj} is het effect van probe j op de expressie van gen g en ϵ_{gij} is de meetfout. Met de volgende code passen we PLM toe op de probe data door middel van robuste regressie. Vervolgens krijgen we een plot van de chips naar de residuen (ϕ_{gj}) in het PLM model,

>library("affyPLM")
>PLMData ← fitPLM(Data)
>par(mrow=c(2,3))
>image(PLMData, type="resids")

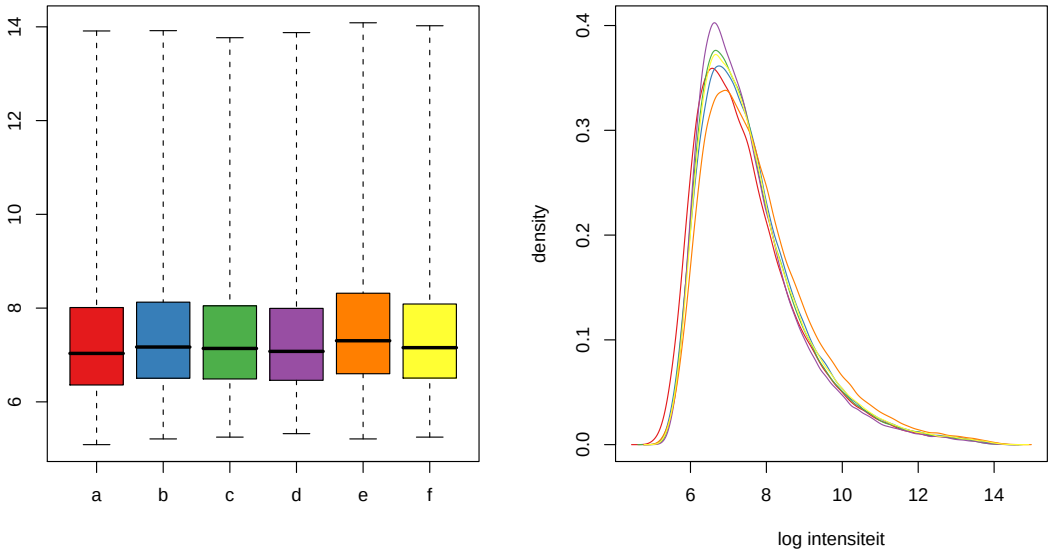


Hier zien we in blauw de probes met een positief residu (rood in het negatieve geval) in het PLM model naar voren komen. Naast de artefact die we in de intensiteit plot al zagen zien we dat de andere chips ook afwijkingen vertonen.

Naast een plot van de intensiteit op elke chip kunnen we de chips natuurlijk ook met elkaar vergelijken. De volgende twee plots geven de dichtheid van de diverse expressies op de chips weer. We verwachten dat deze op de verschillende chips overeen komen. De volgende code levert een boxplot en een histogram;

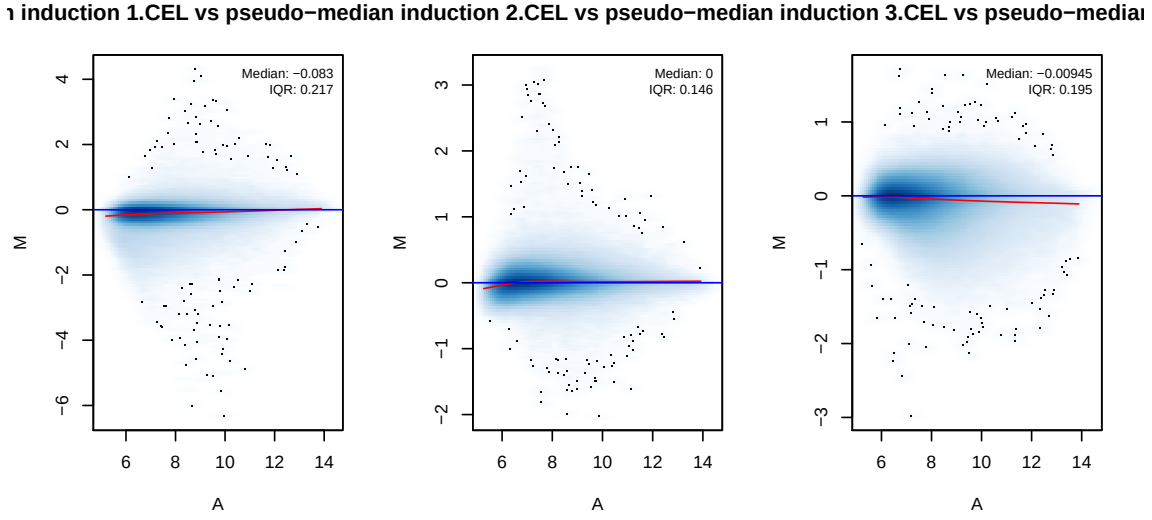
>library("RColorBrewer")
>kleur←brewer.pal(6, "Set1")

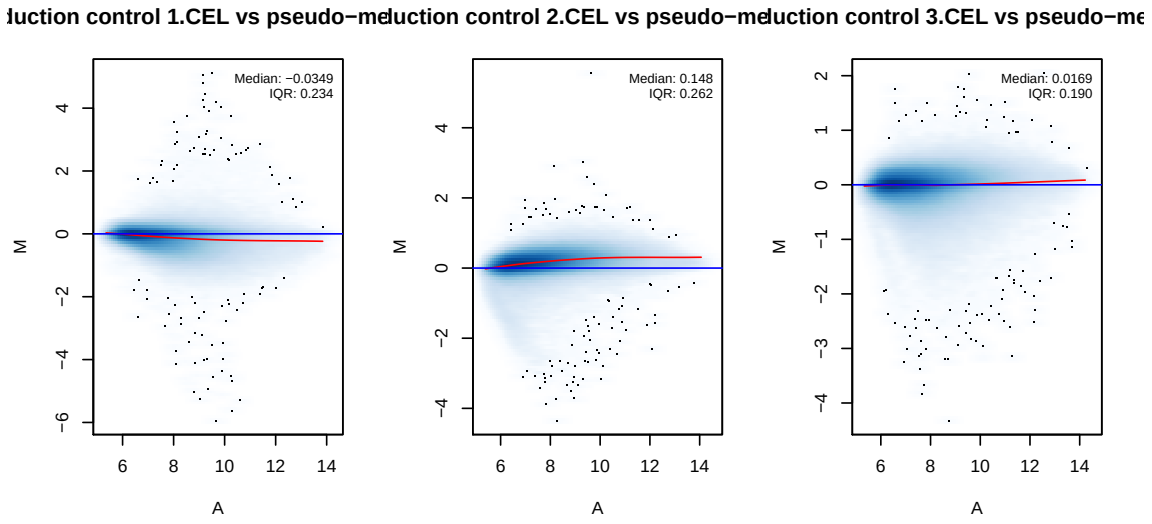
```
>par(mfrow=c(1,2))
>boxplot(Data, col=kleur, names=letters[1:6])
>hist(Data, col=kleur, lty=1, xlab="log intensiteit")
>legend(12,1,letters[1:6], lty=1, col=kleur)
```



We zien inderdaad dat de boxplot en het histogram geen rare uitschieters hebben. Een andere nuttige plot om te maken is de zogenaamde MAplot, de MAplot van twee vectoren Y_1 en Y_2 is een rotatie en een schaling van de scatterplot. In plaats van $Y_{2,j}$ Vs. $Y_{1,j}$ voor $j = 1, ..., J$ plotten we $M_j = Y_{2,j} - Y_{1,j}$ tegen $A_j = \frac{Y_{2,j}+Y_{1,j}}{2}$. Y_1 enY_2 zijn in ons geval logaritmes waardoor M_j de log verandering en A_j de gemiddelde log intensiteit voor gen j weergeeft. Het voordeel van een dergelijke plot is dat in een microarray experiment van de meeste genen verwacht kan worden dat ze niet tot expressie komen (de rood en groen intensiteit zijn ongeveer gelijk). Van een MA-plot kunnen we verwachten dat alle data punten rond $M = 0$ zitten. *affy* bevat een aantal mogelijkheden voor het maken van MA-plots, wij gebruiken *Maplot*. Maplot plot de arrays ten opzichte van een referentie array (andere scripts plotten alle paarsgewijze combinaties van chips wat in het geval met 6 chips niet zo nuttig is.). De referentie array wordt gemaakt door per probe de mediaan van alle chips te nemen. De volgende code geeft een uitgesmeerde MAplot voor onze chips tov een referentie array.

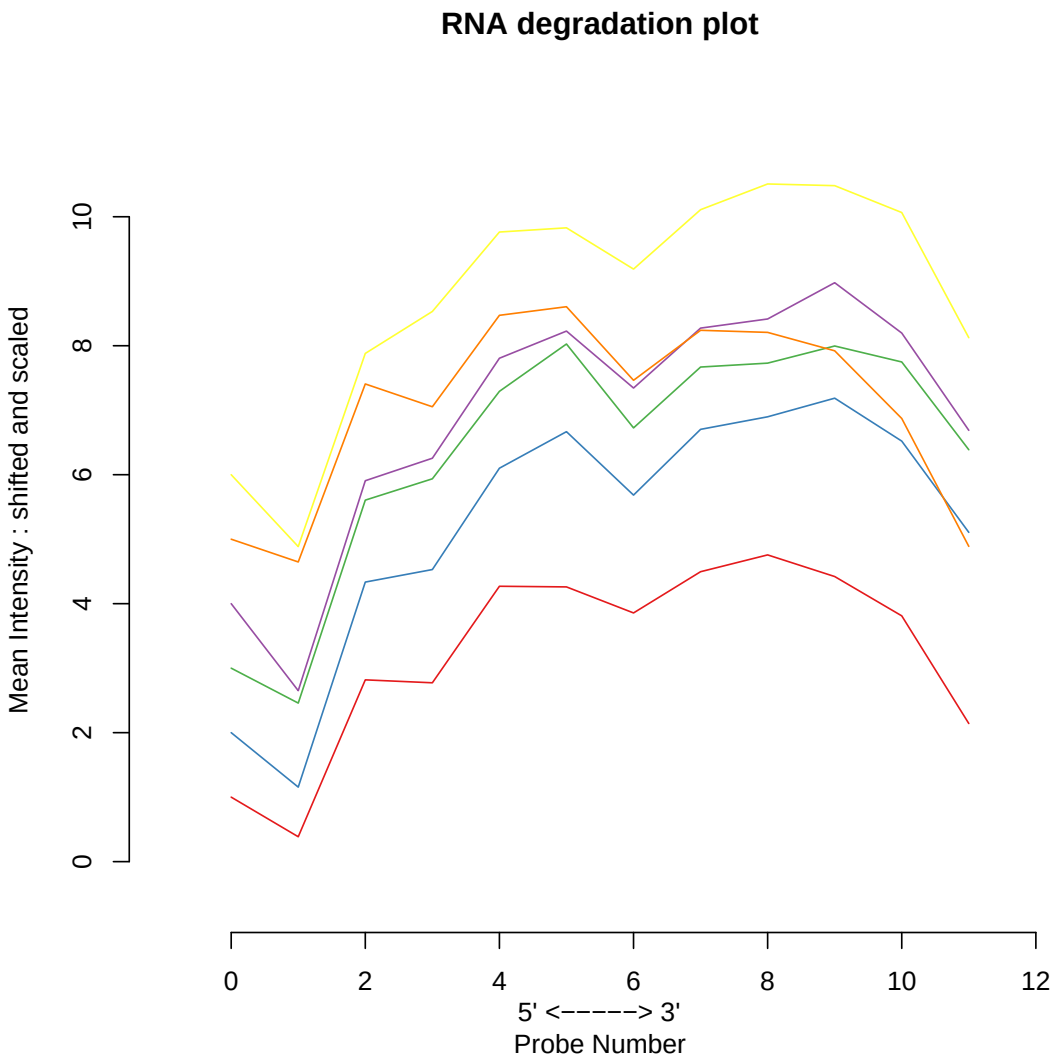
```
>library(geneplotter)
>par(mfrow=c(2,3))
>MAplot(Data, plot.methode="smoothScatter")
```





De MAplot bevat geen rare uitschieters. De laatste plot die we kunnen maken met betrekking tot de data kwaliteit is de RNA afname plot. Zoals al eerder is opgemerkt verschillen de probe intensiteiten afhankelijk van de plek op het RNA. De volgende plot geeft een gemiddelde expressie per probe weer, Laat Y_{ij} intensiteit van probe i van gen j we plotten; $Y_{probe\ i} = \frac{\sum_{j=1}^{15544} Y_{ij}}{15544}$ Wederom biedt *affy* een script om deze plot te maken;

```
>library(affy)
>RNAdeg←AffyRNAdeg(Data)
>plotAffyRNAdeg(RNAdeg)
```



We zien dat de RNA afname per chip gelijk is. Verder zien we dat probes aan de 3' kant een iets hogere intensiteit hebben. Dit is een bekend verschijnsel waar door onder andere *rma* voor gecorrigeerd wordt. Raar aan deze plot is eventueel de dip in intensiteit bij probe 6, maar aangezien alle chips dit vertonen zal het voor de uiteindelijk analyse niet veel uit maken.

4.3 Analyse van de data

Na het succesvol inlezen van de data en controleren van de kwaliteit van de data kunnen we beginnen met de analyse. Alles wat we hier voor nodig hebben is al behandeld, voor de wiskundige onderbouwing van het onderstaande verhaal verwijs ik dan ook terug naar 3.1 en **??**.

Limma

Voor het fitten van een lineair model aan de data gebruiken we het bioconductor package *limma*. In **??** hebben we gezien dat de basis voor het lineaire model de design matrix is. Ons experiment bestaat uit 6 experimenten waarbij 3 maal tunicamycin is toegevoegd en 3 maal een controle is uitgevoerd. De tests zijn niet gepaard (dwz, de controle arrays zijn onafhankelijk van de testarrays geproduceerd). De design matrix voor ons experiment is als volgt;

$$X = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}^T$$

We zijn uiteraard genteresseerd in het verschil in expressie tussen de test en controle samples. De contrast matrix wordt dan ook simpelweg, $C = (-1, 1)^T$ De volgende code leest de disign matrix in en maakt de contrast matrix;

```
>design←cbind(Controle=c(0,0,0,1,1,1), Test=c(1,1,1,0,0,0))
>colnames(design)←c("Controle", "Test")
>cont.matrix←makeContrasts(TestvsControle=Test-Controle, levels=design)
```

Met de design matrix kunnen we met de functie *lmFit* nu aan de genormaliseerde microarray data het lineair model passen, hier voor zijn twee methodes beschikbaar, *lmFit(, ,methode='ls')* fit het model met least squares methode en *lmFit(, ,methode='robust')* gebruikt robuste regressie. Vervolgens geeft *contrasts.fit* het contrast tussen Test en Controle waarin we genteresseerd zijn. Tot slot passen we *eBayes* toe, zoals in **??** past dit script een empirische bayes methode toe om de β statistiek te berekenen. De volgende code voert het bovenstaande uit;

```
>fit←lmFit(eset, design, methode='robust')
>fit2←contrasts.fit(fit, cont.matrix, methode='robust')
>ebayes←eBayes(fit2)
```

De volgende code geeft ons nu de meest significante genen, gesorteerd naar de β statistiek zoals besproken in **??**. De aangepaste p-waardes zijn berekend door middel van Benjamini-Hochberg (3.9), andere correcties zijn ook mogelijk door *topTable(....., adjust.method='holm')* of *topTable(....., adjust.method='bonferroni')*.

```
>topTable(ebayes, number=10, genelist=ebayes$genes, adjust.method='BH')
```

ID	logFC	AveExpr	t	P.Value	adj.P.Val	B
An00g04128	-1.819071	7.845584	-18.37026	6.194134e-07	0.006470295	5.510780
An00g10482	1.050999	10.913212	16.37064	1.310660e-06	0.006470295	5.115698
An00g03712	1.284565	9.421294	16.27569	1.361074e-06	0.006470295	5.094533
An00g05859	1.288754	7.883174	15.61792	1.778286e-06	0.006470295	4.941098
An00g12062	1.014680	10.071636	14.20338	3.283663e-06	0.007150254	4.566075
An00g00175	1.676833	10.966144	14.19632	3.294212e-06	0.007150254	4.564030
An00g05909	1.133622	8.219286	14.10182	3.439039e-06	0.007150254	4.536512
An00g08008	1.228309	9.180138	12.58378	7.143940e-06	0.011618614	4.045398
An00g07353	1.423748	11.818513	12.54356	7.291679e-06	0.011618614	4.031017
An00g07382	-1.059744	8.502887	-12.36697	7.983107e-06	0.011618614	3.966968

In deze tabel staat ID voor de naam van het gen, logFC is de log fold change, AveExpr is de gemiddelde expressie, t is de standaard t statistiek zoals in **??** beschreven, P.Value is de niet voor multipletesting gecorrigeerde p-waarde, adj.P.Value is de p-waarde aangepast voor multiple testing door middel van Benjamini-Hochberg en B is de β statistiek besproken in **??**. Door in *topTable* 'number' te veranderen kunnen meer genen worden weer gegeven. Andere mogelijkheden voor *topTable* zijn te vinden in de bioconductor help file.

Via de vertaal file kunnen we de genen in de topTable terug zoeken in de oorspronkelijke annotatie file. Hierin staat voor elk gen de functie beschreven (voor zover die bekend is). Onze topTable levert het volgende op;

An00g04128	An02g00190	01 METABOLISM
An00g10482	An08g03960	99 UNCLASSIFIED PROTEIN
An00g03712	An02g13410	01 METABOLISM, 40 SUBCELLULAR LOCALISATION
An00g05859	An18g04260	40 SUBCELLULAR LOCALISATION
An00g12062	An08g01740	01 METABOLISM
An00g00175	An11g04180	05 PROTEIN SYNTHESIS, 06 PROTEIN FATE
An00g05909	An09g05420	40 SUBCELLULAR LOCALISATION
An00g08008	An05g00880	06 PROTEIN FATE
An00g07353	An01g08420	06 PROTEIN FATE, 40 SUBCELLULAR LOCALISATION
An00g07382	An03g05200	01 METABOLISM

We zien dat 06 PROTEIN FATE relatief vaak voorkomt, we zullen daar straks verder op in gaan. De volgende tabel geeft het aantal significante genen weer bij verschillende α grenzen voor de FDR.

α	0.01	0.05	0.1	0.15	0.20
aantal	7	26	83	115	195

We zien dat relatief weinig genen significant kunnen worden verklaard. Dit hangt natuurlijk in grote maten samen met het feit dat we meer dan 14000 nulhypotheses tegelijk willen testen. In het volgende hoofdstuk zullen we twee strategien bespreken om het aantal nulhypotheses omlaag te krijgen.

Hoofdstuk 5

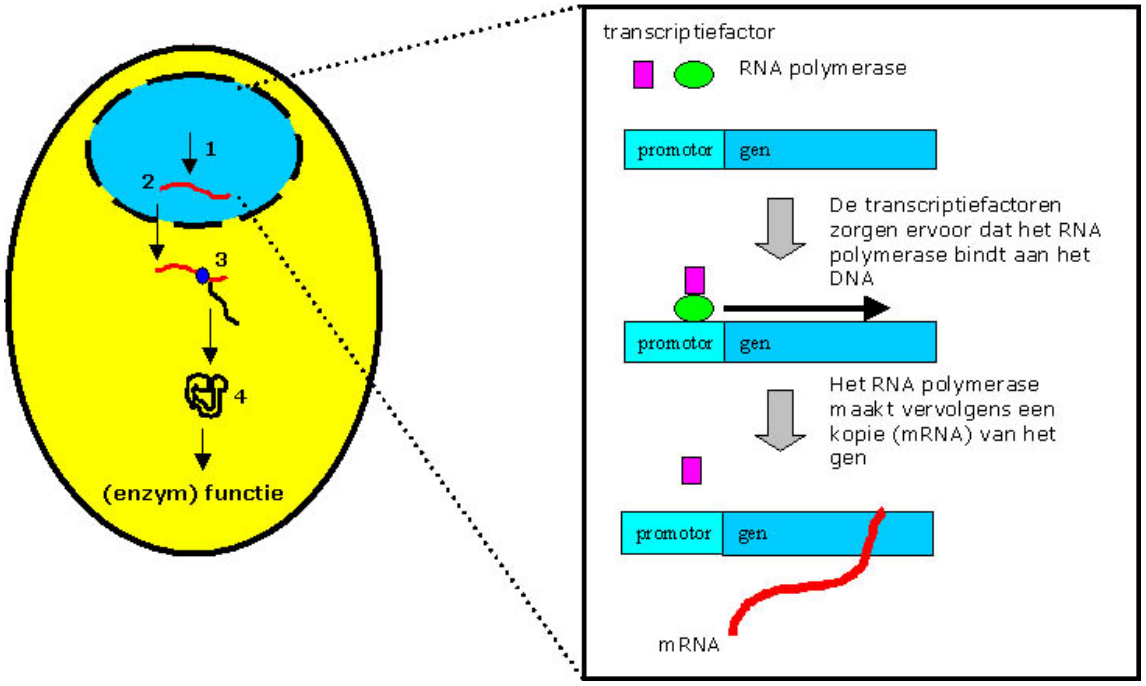
Verbeteringen

In het vorige hoofdstuk hebben we gezien dat de analyse van microarray data wordt bemoeilijkt door de grote hoeveelheid nulhypotheses die worden getest. Dit hoofdstuk geeft twee mogelijke methodes die de analyse kunnen verbeteren. De eerste paragraaf kijkt naar zogenaamde transcriptie factoren. De tweede paragraaf kijkt naar de functie van bepaalde genen. In ons experiment zijn we genteresseerd in de eiwit vouwing. Door de genen waarop we testen te beperken tot een groep genen die alleen aan eiwit vouwing doen testen we minder nulhypotheses en houden we meer power over.

5.1 Promotor factoren en het MEME algoritme

Bij een microarray analyse zijn we genteresseerd in genen die anders dan normaal tot expressie komen. Maar zoals al is gebleken bemoeilijkt het grote aantal genen het vinden van significante expressies. In deze paragraaf kijken we naar gezamenlijke transcriptie factoren. Transcriptie factoren zijn eiwitten die de productie van andere eiwitten reguleren.

Een promotor is een gebied van tussen de 500 en 1000 base paren voor het gen op het DNA. In dit promotor gebied bevinden zich zogenaamde 'binding sites'. Hieraan kunnen transcriptie factoren binden die op hun beurt de expressie van het gen reguleren. Deze transcriptie factoren worden vaak door meerdere genen gedeeld. Het volgende plaatje geeft het proces schematisch weer.



Het vinden van gezamenlijke binding-sites in de promotor gebieden van een groepje genen dat in eerste instantie niet significant tot expressie komt kan een indicatie zijn dat de niet significatie expressie toch het gevolg is van het experiment. Aan de andere kant kunnen informatie of transcriptie factoren helpen om genen die onterecht significant zijn er uit te filteren.

MEME algoritme

In 5.1 hebben we gezien dat een van de mogelijke verbeteringen in microarray analyse kan liggen in het zoeken naar gelijke transcriptie factoren. Voor ons experiment zullen we dit doen met het MEME algoritme. Het MEME algoritme is gebaseerd op het EM of expectation maximalisation algoritme uit 1990 en voor het eerst gepubliceerd door Bailey en Elkan [7]. Voor een uitgebreide uitleg over het algoritme verwijs ik dan ook naar dit artikel

Het MEME algoritme baseert op kans basis de meest waarschijnlijke transcriptie factor gegeven de lengte van de transcriptie factor en een willekeurig aantal promotor gebieden. Schematisch;

1. **MEME**(Data, Lengte, Aantal) {
2. Voor $i = 1$ *tot aantal* {
2. Kies random Q zdd $Q_{ij} \neq 0\ \forall i, j$
3. doe {
3. Bepaal KV uit Q
4. Maak Q uit KV
5. } tot $\Delta Q \leq \tau$
6. $Q = Q_i$ }
7. voor Q_i bepaal $\sum$
8. return Q_i *met grootste LLH*

9. }

Transcriptie factoren zijn net als het DNA opgebouwd uit vier elementen, voor het gemak, de letters T, A, C en G. Het MEME algoritme bepaald voor deze letters de kans dat ze op een bepaalde plaats in de transcriptie factor staan. Dit gebeurd op de volgende manier. Q is de matrix met letter kansen voor bepaalde plekken in het motief, iaw, $Q_{(i,j)} = \mathbb{P}(\textit{letter } i \textit{ staat op positie } j)$. De eerste Q matrix kiezen we willekeurig, maar wel zo dat er geen $Q_{(i,j)}$ nul is. Met de Q matrix kunnen we voor elke positie in de promotor de kans berekenen dat de transcriptie factor daar begint. De matrix met deze kansjes noemen we dat 'kans matrix' $KV_{(i,j)} = \mathbb{P}(\textit{transcriptie factor begint op plek } i \textit{ in promotor van gen } j)$.

$$KV_{(i,j)} = \prod_{k=i}^{k=i+l} Q_{(A,k-i)}1_{D(k,j)=A} + Q_{(T,k-i)}1_{D(k,j)=T} + Q_{(C,k-i)}1_{D(k,j)=C} + Q_{(G,k-i)}1_{D(k,j)=G}$$

$D_{(i,j)}$ is hier de data matrix met $D_{(i,j)} = \textit{letter op plek } i \textit{ in promotor gen } j$, 1 is de indicator functie en l is de lengte van de transcriptiefactor. De strategie van MEME is om vanuit de KV matrix een nieuwe Q te produceren enz enz. Dit gebeurd met de kans(jes) uit de KV matrix als weeg factor. schematisch gaat het als volgt.

```
voor gen j {
  voor alle posities in in de promotor van het gen {
    voor de posities 1 tot lengte van de transcriptie factor {
      als positie == A {
        Q(A,positie) = Q(A,positie) + KV(positie, j) }
      als positie == T {
        Q(T,positie) = Q(T,positie) + KV(positie, j) }
      als positie == C {
        Q(C,positie) = Q(C,positie) + KV(positie, j) }
      als positie == G {
        Q(G,positie) = Q(G,positie) + KV(positie, j) }
       }
    }
  }
```

Merk op dat deze nieuwe matrix Q niet direct een kans matrix is en dus nog genormaliseerd moet worden voor een nieuwe KV matrix kunnen berekenen. Het MEME algoritme gaat op deze wijze door tot de Q matrix niet (nauwelijks) meer veranderd of een maximaal aantal stappen zijn bereikt.

Resultaat

In het vorige hoofdstuk staat een korte schets van het MEME-algoritme. Zelf heb ik in matlab een programma geschreven dat het algoritme uitvoert op een gegeven aantal promotor gebieden. We hebben het algoritme toegepast op de volgende genen.

An01g10930	An03g06550	An04g06910	An04g06920	An11g03340
An15g00310	An15g03940	An15g04060	inuA	sucA

De vraag was of er voor deze genen een gezamenlijke transcriptiefactor te vinden is. Ik heb het MEME algoritme geprogrammeerd in matlab en daarna uitgevoerd op de 10 genen. Van een mogelijke transcriptiefactor was bekend dat deze bestaat uit drie base paren, acht willekeurige base paren en tot slot weer drie base paren. Met 20 verschillende start plekken vindt het programma de volgende promotiefactoren;

$$\text{TFINAL} = \begin{matrix} \text{TCCCCC} \\ \text{TCCCCC} \\ \text{TCCCCC} \\ \text{TCCCCC} \\ \text{TCCCCC} \\ \text{TCCTCC} \\ \text{TCCCCC} \\ \text{TCCCCC} \\ \text{TCCCCC} \\ \text{TCCACC} \end{matrix} \qquad \text{QFINAL} = \begin{pmatrix} 0.0000 & 0.0000 & 0.0000 & 0.1307 & 0.0000 & 0.0000 \\ 1.0000 & 0.0000 & 0.0000 & 0.2576 & 0.0004 & 0.0076 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 1.0000 & 0.6117 & 0.9996 & 0.9924 \end{pmatrix}$$

$$\text{SFINAL} = \begin{pmatrix} 51 & 857 & 810 & 552 & 444 & 861 & 491 & 740 & 825 & 47 \end{pmatrix}$$
$$\text{TESTFINAL} = -0.9642$$

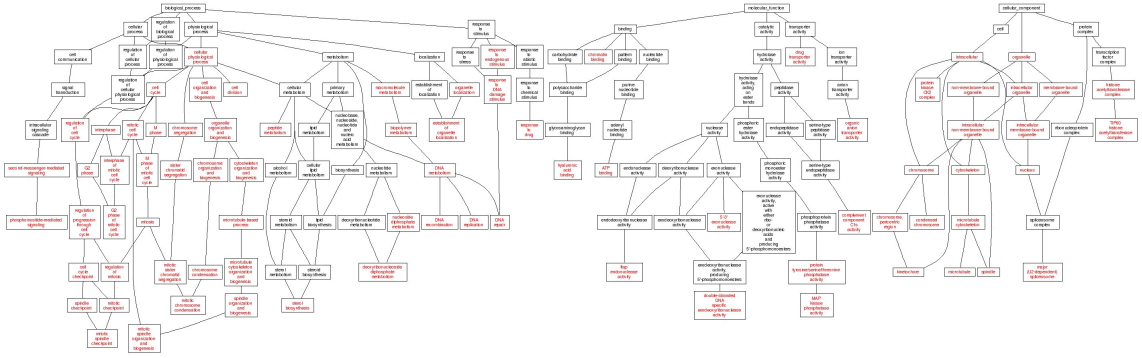
QFINAL is hier de uiteindelijke Q matrix met net als hier boven $Q_{ij} = \mathbb{P}(letter\ i\ op\ plek\ j)$. We zien dat de eerste letter met kans 1 een T is, de tweede met kans 1 een C etc. TFINAL geeft de uiteindelijk best passende transcriptie factoren. SFINAL geeft van elk array de begin plaats van de transcriptie factor en TESTFINAL geeft de log likelihood van QFINAL over de gehele data.

Helaas levert dit resulaat niet zoveel op. TCCCCC is geen bekende transcriptie factor en de kans dat we hiermee een nieuwe ontdekking hebben gedaan lijkt klein. Transcriptie factoren zijn vaak palindromen en TCCCCC is dit natuurlijk niet. Het liefst zou ik nu MEME een aantal maal draaien met genen uit de topTable van het vorige hoofdstuk. Een gezamenlijke transcriptiefactor zou een indicatie kunnen zijn dat genen die misschien niet in de topTable voorkomen maar wel dezelde transcriptiefactor delen toch ook tot expressie komen. Helaas gooien de annotatie problemen hier roet in het eten. Veel promotorsequenties zijn niet te vinden voor genen uit de topTable.

5.2 Testen op bomen

Geno Ontology

Een van de in het vorige hoofdstuk besproken verbeteringen voor microarray data analyse is het gebruik van gen annotatie. Annotatie maakt het mogelijk om genen met een vergelijkbare functie of plaats op het dna te groeperen en gezamenlijk te analyseren. De belangrijkste en populairste bron voor deze annotatie is de Gene Ontology (GO) database. Go is vooral populair dankzij zijn structuur. De elementen van Go zijn geordend als een acylise graaf.



voorbeeld van een GO-graaf

5.2.1 Testen in de GO-graaf

Elke gen set in de Go-graaf heeft een bijbehorende nulhypothese die we kunnen testen. We gaan er van uit dat deze nulhypotheses zo zijn geformuleerd dat ze de relaties in de GO-graaf respecteren. Kort gezegd, elke deelverzameling relatie tussen twee sets genen impliceert een logische relatie tussen de bijbehorende nulhypotheses.

Definitie 5.1. Laat A, B, C gen verzamelingen en $H_{0,A}, H_{0,B}, H_{0,C}$ de bijbehorende nulhypotheses we nemen aan;

1. Als $B \subseteq A$ en $H_{0,A}$ is waar dan is $H_{0,B}$ waar.
2. Als $A = B \cup C$ en $H_{0,B}$ en $H_{0,C}$ zijn waar dan is $H_{0,A}$ waar.

We zien dat de nulhypothese behorende bij de verzameling van alle genen verworpen wordt als er ook maar een nulhypothese wordt verworpen. Verder moet worden opgemerkt dat de logische structuur die geldt voor de nulhypotheses niet noodzakelijk geldt voor onaangepaste (niet-gecorrigeerd voor multiple testing) test resultaten. Het kan bijvoorbeeld voorkomen dat $H_{0,B}$ significant is maar $H_{0,A}$ niet terwijl $B \subseteq A$.

Bottom-up

De eerste methode voor het testen in de GO-graaf die we bekijken is de 'bottom-up' methode. Deze methode zoals de naam al zegt doorloopt de boom (of eigenlijk de gerichte acylische graaf) vanuit de takken richting de top. Door de logische implicaties hoeven we alleen de bladeren van de boom te testen.

Definitie 5.2 (bottom-up procedure)**.** Beschouw een GO-graaf met bladeren $A_1, A_2, .., A_n$ en kies een significatie niveau α. We testen de nulhypotheses $H_{0,A_1}, H_{0,A_2}, .., H_{0,A_n}$ met de methode van Holm-Bonferoni 3.1.2. Na het significant (of niet) verklaren van de bladeren ligt de significantie van de andere gen-verzamelingen vast met 5.1.

Het voordeel van de bottom-up methode is dat we in plaats van alle knopen van de Go-graaf te testen kunnen volstaan met alleen de bladeren. We hoeven dus minder nulhypotheses te testen en dit betekent dat we meer power over houden. Het is een snelle procedure omdat we na 1 keer testen al klaar zijn. Het nadeel is dat de GO-graaf nog steeds een groot aantal bladeren heeft waardoor de correctie voor het multiple testen nog veel invloed heeft. Daarnaast vallen sommige globale effecten (een groot aantal genen dat een klein beetje tot expressie komt) minder op.

Top-down

Het alternatief voor de bottom-up methode is (niet zo verwonderlijk) de top-down methode. De methode is gebaseerd op de gesloten testprocedure van Marcus, Peritz and Gabriel [8]. We testen van boven af. Als de top significant is dan test de procedure de knopen er onder, dit gaat door tot er geen significante knopen meer zijn of de bladeren van de boom zijn bereikt.

Definitie 5.3 (top-down procedure)**.** Beschouw een GO-graaf met n knopen. Stap 1, we maken een nieuwe graaf met de knopen van de Go-graaf die gesloten is onder vereniging. Maw, als $A, B \subseteq GO - graaf \Rightarrow A \cup B \subseteq GO - graaf$. De procedure begint bij de wortel en test vervolgens alle verzamelingen waarvan de bovenliggende verzamelingen niet significant zijn bevonden.

Alle testen in deze procedure worden uitgevoerd gedaan op een niveau α, toch houdt de methode sterke controle over de FWER. Voor het bewijs verwijs ik naar [8]. Het feit dat we elke hypothese kunnen testen tegen een niveau α maakt de top-down procedure natuurlijk erg aantrekkelijk. Het nadeel is echter dat het construeren van een graaf G uit de Go-graaf die gesloten is onder vereniging problemen kan opleveren. Als we als voorbeeld de GO-graaf van de mens nemen, deze heeft 2865 knopen. Als we deze graaf willen uitbreiden zo doende dat deze gesloten is onder vereniging levert dat een graaf met $8.5 * 10^{270}$ [9] knopen op. De topdown methode is wat betreft het vinden van significante genen het tegenovergestelde van de bottom-up. Topdown zal meer globale effecten vinden en lokaal sterk tot expressie gekomen genen over het hoofd zien als de superset niet significant is.

Focuslevel methode

Naast de bestaande bottom-up en top-down methode is er nog een derde alternatief voor het testen op de GO-graaf die de sterktes en zwaktes van beide methodes combineert. De zogenaamde focuslevel methode beschreven door Goeman [9] kiest een focuslevel in de boom, en voert vanuit dit focuslevel zowel bottum-up als top-down uit. Hiermee is de methode beter in het vinden van globale effecten dan bottom-up en beter in het vinden van lokale uitschieters dan de top-down methode. Daarnaast is een top-down analyse vaak niet mogelijk door het grote aantal knopen in de gecomplementeerde graaf.

5.2.2 06 PROTEIN FATE boom

We hebben nu drie methodes gezien om te testen in de GO-graaf. Ondanks dat er wederom diverse paketten in bioconductor zijn om deze tests uit te voeren (bijvoorbeeld *globaltest*) heeft dit geen nut voor ons onderzoek. Net als bij de annotatie is er geen GO-graaf van de Aspergillus.

We zullen dus zelf een GO-graaf moeten bouwen en daar vanuit gaan testen. Zoals al in het eerste hoofdstuk te lezen is zijn we in het onderzoek genteresseerd in de UPR. Het deel van de GO-graaf waar tegen we willen testen is dat ook BP 06 PROTEINFATE. Uit de beschikbare annotatie (dsmanigeraancoll.exel) is met perl een matrix geconstueerd die de boom bevat. Nadeel is dat de exel file gebruik maakt van een andere naamgeving voor

genen dan de cdf annoatie die in bioconductor is gebruikt. Gelukkig is ook dit te omzeilen door middel van een vertaalfile.

Met een kort perl script heb ik uit de exeltext file een textfile boom.txt gemaakt. Boom.txt beschrijft de 06 PROTEINFATE boom als een tabspaced matrix met als rijen de verschillende genen en als kolommen de verschillende knopen in de boom. Omdat dit format anders is dan de invoer bij de bekende GO-pakketten voor bioconductor (bijvoorbeeld *mulTest*) kwam er wederom een custom scriptje aan te pas. De volgende code voert de bottom-up methode uit op de 06 PROTEINFATE boom. Door onze zelf gemaakte boom zullen we alleen de bottom-up methode uitvoeren. Top-down testen zou betekenen dat we de boom moeten complementeren zodat deze gesloten is onder vereniging, dit zou een enorme graaf opleveren. Daarnaast is een groot deel van de code specefiek geschreven voor deze boom en een aanpassing voor de gecomplementeerde boom zou veel werk zijn.

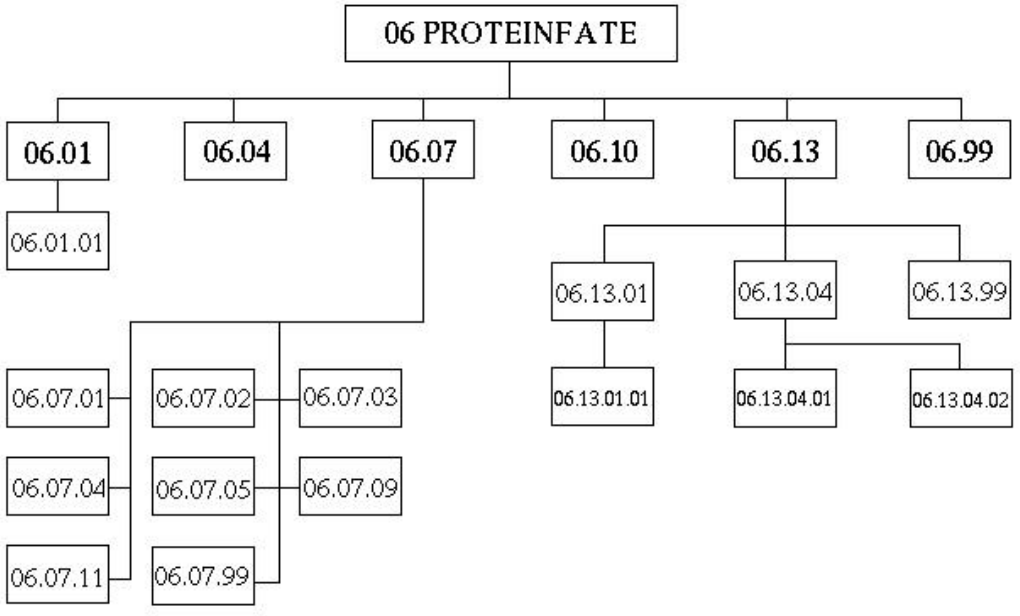
```
>library(multest)
>colnam ← read.table(name.txt)
>boom ← read.table(boom.txt, colClasses=colclas)
>colnames(boom)← colnam
>for i in 1:length(colnam)
blad[i]←subset(boom, boom[i]==1, select=gen)
rauwep[i]←ebayes(p.value$[blad[i]gen,])
adjp[i]←mt.rawp2adjp(rauwep[i], "BH")
lijstje[i]←cbind(blad[i][adjp[i]$index[0:5,], adjp[i]$adjp[0:5,2])
```

Lijstje[] bevat nu de top 5 significante genen met bijbehorende gecorrigeerde p-waardes. Er zijn 15 bladeren aan de boom, hieronder volgt van elk blad de top 5;

An00g00175	An00g08008	An00g07353	An00g03066	An00g03054	*06.01*
0.0005	0.0005	0.0005	0.0005	0.0006	
An00g04128	An00g05909	An00g08008	An00g07353	An00g03066	*06.04*
0.0003	0.0008	0.0009	0.0009	0.0010	
An00g04872	An00g04261	An00g10626	An00g08499	An00g10171	*06.07.01*
0.0376	0.0496	0.1392	0.1392	0.2759	
An00g11355	An00g09668	An00g10484	An00g12028	An00g11344	*06.07.02*
0.0013	0.0013	0.0022	0.023	0.0023	
An00g06491	An00g06579	An00g12429	An00g11008	An00g11256	*06.07.03*
0.2818	0.2818	0.2818	0.2923	0.3280	
An00g08662	An00g10063	An00g06914	An00g09956	An00g04355	*06.07.04*
0.7480	0.7480	0.7480	0.7480	0.7480	
An00g12008	An00g08267	An00g13947	An00g10394	An00g12014	*06.07.05*
0.0586	0.2254	0.2254	0.2254	0.2254	
An00g10229	An00g10100	An00g08234	An00g03567	An00g09710	*06.07.09*
0.0202	0.2704	0.5100	0.5100	0.5100	

An00g08069	An00g09725	An00g00163	An00g10557	An00g13947	*06.07.11*
0.0009	0.0103	0.1273	0.1273	0.1464	
An00g03066	An00g08069	An00g04358	An00g04261	An00g05962	*06.07.99*
0.0006	0.0006	0.1107	0.1223	0.1703	
An00g00175	An00g08008	An00g12011	An00g08069	An00g06075	*06.10*
0.0010	0.0010	0.0010	0.0013	0.0172	
An00g12011	An00g06679	An00g06607	An00g11256	An00g06695	*06.13.01.01*
0.0005	0.0470	0.0865	0.1317	0.1317	
An00g08224	An00g11644	An00g00154	An00g11815	An00g09628	*06.13.04.01*
0.0330	0.0330	0.3014	0.3014	0.3014	
An00g07382	An00g08071	An00g07381	An00g08502	An00g10628	*06.13.04.02*
0.0002	0.0964	0.0964	0.0964	0.3533	
An00g05909	An00g08787	An00g07786	An00g08224	An00g11030	*06.13.99*
0.0002	0.0096	0.0107	0.0223	0.0223	
An00g007080	An00g07083	An00g10149	An00g11611	An00g10150	*06.99*
0.0048	0.0048	0.2457	0.2766	0.5532	

De volgende afbeelding geeft de hele 06 PROTEIN FATE boom weer;



06.	PROTEIN FATE
06.01	protein folding and stabilization
06.04	protein targeting. sorting and transporting
06.07	protein modification
06.07.01	modification with fatty acids
06.07.02	modification with suger residues
06.07.03	modification by phosphorylation, dephosphorylation
06.07.04	modification by acetylation, deacetylation
06.07.05	modification by uniquitination, deubiquitination
06.07.09	posttranslational modification of amino acids
06.07.11	protein processing (proteolytic)
06.07.99	other protein modifications
06.10	assembly of protein complexes
06.13	proteolytic degration
06.13.01	Cytoplasmic and nuclear degradation
06.13.01.01	Proseasomal degradation
06.13.04	Iysosomal and vacuolar degration
06.13.04.01	Isosomal degration
06.13.04.02	Vacuolar degration
06.13.99	other proteolytic degration

Resultaat

Als we voor de FDR $\alpha = 0.1$ nemen dan zien we dat 8 van de 15 bladeren significant zijn. Met de logische structuur die we eerder hebben gedefinieerd kunnen we niet anders dan concluderen dat ook de top van de graaf significant is. Maw, 06 PROTEIN FATE is op basis van het experiment betrokken bij de response op het experiment. De oplettende lezer zal al hebben opgemerkt dat we dit ook hadden kunnen concluderen zonder de test op de boom uit te voeren. An00g08008, komt namelijk ook voor in 4.3 (topTable)

Hoofdstuk 6

Conclusie

Microarray analyse is een krachtige methode om binnen de experimentele biologie onderzoek te doen. De mogelijkheid om met een test alle genen van een organisme te kunnen testen is geweldig. Dit alles komt natuurlijk wel tegen een prijs. Het testen van vele nulhypotheses en de correcties die er voor nodig zijn om dit mogelijk te maken zorgen dat er weinig power over blijft om genen significant te verklaren. Daarnaast kan, zoals bij ons het geval was, het gebrek aan annotatie voor een hoop problemen zorgen.

Wat betreft het experiment kunnen we concluderen dat het secretie pad inderdaad betrokken lijkt bij de reactie op de tunicamycin. In de Go-graaf zien we, net als in de topTable meerdere genen in de 06 PROTEIN FATE groep naar voren komen. Feit blijft echter wel dat andere genen in de topTable nog hoger scoorden. An00g10482 is unclassified, misschien is dit een hint dat dit gen mogelijk iets met de secretie te maken heeft.

Het verbeteren van de microarray analyse door het testen op relevantie takken van de GO-graaf werkt goed. Waar bij de standaard analyse veel genen niet significant konden worden verklaard omdat er gecorrigeerd moet worden voor multiple testing, is dit bij het testen in de GO-graaf geen probleem. Een mooi vervolg aan deze scriptie zou dan ook het verder uitwerken van de Go-graaf voor de Aspergillus Niger zijn. Door middel van een topdown of focuslevel analyse zouden dan eventueel enkele globale effecten bij de UPR naar boven komen.

Bibliografie

[1] Thomas Guillemette et al. Genomic analysis of the secretion stress response in the enzyme-producing cell factory aspergillus niger. *BMC Genomics*, 8, 2007.

[2] Dov Stekel. *Microarray Bioinformatics*. Cambridge University Press, 2003.

[3] Yoav Benjamini and Yosef Hochberg. Controling the false discovert rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistics Society*, 57(1), 1995.

[4] G.K. Smyth. Linear models and empirical bayes methodes for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3(1), 2004.

[5] Lonnstedt and Speed. Replicated microarray data. *Statistica Sinica*, 12, 2002.

[6] Robert Gentleman et al. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, 2005.

[7] T.L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using experctation maximization. *Machine Learning*, 21, 1995.

[8] R. Marcus et al. On closed testing procedures with special reference to ordered analysis of variance. *Biometrika*, 63, 1976.

[9] Jelle Goeman and Ulrich mansmann. Multiple testing on the directed acyclic grap of gene ontology. *Bioinformatics*, 24(4), 2008.