Faculteit Bio-ingenieurswetenschappen
Academiejaar 2015 – 2016

Next generation telomeerlengtebepaling

Dries Godderis

Promotor & tutor: Prof. dr. ir. Tim De Meyer

Masterproef voorgedragen tot het behalen van de graad van
Master in de bio-ingenieurswetenschappen: cel-en genbiotechnologie

Faculteit Bio-ingenieurswetenschappen
Academiejaar 2015 – 2016

Next generation telomeerlengtebepaling

Dries Godderis

Promotor & tutor: Prof. dr. ir. Tim De Meyer

Masterproef voorgedragen tot het behalen van de graad van
Master in de bio-ingenieurswetenschappen: cel-en genbiotechnologie

Woord vooraf

Na mijn studies van een Bachelor in de Biomedische Laboratoriumtechnologie – Bio-Informatica en een Master in de Industriële Wetenschappen – Biochemie stond ik voor de keuze om al dan niet nog verder te studeren in de Bio-Ingenieurswetenschappen. Het werkveld leek na vijf jaar lang studeren echter de meest logische keuze. Een jaar later en een eerste werkervaring rijker ben ik dan uiteindelijk toch aan de Coupure terechtkomen. Het was al sinds middelbare studies duidelijk dat mijn interesses binnen de biotechnologie-*scene* lagen. Slechts tijdens mijn eerste studie in 2008 wist de bio-informatica benadering van biologische problemen mij erg te interesseren. Het afwerken van een bio-informatica masterscriptie in 2016 is voor mij dan ook wel een zeer bijzonder moment. Ik kan wel zeggen dat ik acht jaar geleden nooit had verwacht dat ik hier ging belanden. Maar het was een leerrijke reis waar ik met plezier en voldoening op terugkijk.

De persoon waar alle dank naartoe gaat om deze masterscriptie te realiseren is mijn promotor en tutor Prof. Tim De Meyer. *Research* is geen simpel gegeven, maar bij hem kon ik steeds terecht - week na week - om resultaten te bespreken en te interpreteren. Met zijn grote passie voor telomeerbiologie was het niet moeilijk voor me om dit afgelopen jaar gemotiveerd te blijven en er steeds weer voor te gaan. In het bijzonder wens ik Tim te bedanken voor de kritische kijk die hij me meegaf gedurende de scriptie. Ook de aangeleerde statistische concepten en werkwijzes die ik bij aanvang minder goed beheerste zie ik als zeer verrijkend voor me in mijn verdere carrière. Het geduld dat hij steeds weer heeft opgebracht om mij methodologieën haarfijn te beschrijven en waar nodig het me nog eens opnieuw uit te leggen was ongezien.

Ik wens ook Prof. Wim Van Criekinge te bedanken voor zijn boeiende lessen bio-informatica en de mogelijkheid om aan BioBix mijn scriptie uit te voeren. De enthousiaste manier waarop zijn lessen gegeven werden maar ook hoe hij opportuniteiten creëert en grijpt is iets waar ik persoonlijk erg naar opkijk. Een algemene dank-jullie-wel gaat uit naar iedereen van de BioBix-groep met in het bijzonder Jeroen Galle om alle technische aspecten te voorzien om deze masterproef mogelijk te maken. Ik wens Tine, Steven en Volodimir erg veel succes in hun verdere jaren van doctoreren. Ik wens ook aan Louis en Simon veel succes, ongeacht welk verder carrièrepad zij kiezen.

Tenslotte wil ik mijn ouders bedanken. Gedurende al deze jaren hebben ze me steeds volledig bijgestaan en me alle kansen gegeven die ik wou. Het zijn ook vooral zij die achter de keuze stonden om voor de titel van bio-ingenieur te gaan, en dus ook deze masterscriptie mogelijk maakten. Ook mijn vrienden bedank ik die de eeuwige student in hun vriendenkring steeds door dik en dun steunden.

Dries Godderis
Gent, juni 2016

Inhoudsopgave

Lijst van afkortingen

aTL:	*absolute telomere length quantitation*
BAM:	*Binary Alignment Map*
CAD:	*coronary artery disease*
CV:	coefficiënt van variatie
D-loop:	*displacement loop*
DNA:	desoxyribonucleïnezuur
ds:	dubbelstrengig
Flow-FISH:	*Fluorescent In Situ Hybridization*
GA:	Genome Analyzer
GCC:	*GNU Compiler Collection*
HPA:	*Hybridization Protection Assay*
Kb:	kilobasen
Mb:	megabasen
MMqPCR:	*monochrome multiplex quantitative* PCR
OPA:	*overhang protection assay*
PENT:	*Primer-Extension Nick Translation*
PNA:	*Peptide Nucleic Acid*
Q-FISH:	*Quantitative Fluorescence In Situ Hybridization*
qPCR:	*quantitative polymerase chain reaction*
RAP1	*Human Repressor Activator Protein* 1
RNA:	ribonucleïnezuur
S:	*single copy gene copy number*
SAM:	*Sequence Alignment Map*
SNP:	*Single-Nucleotide Polymorphism*
STELA:	*Single Telomere Length Analysis*
T/S:	*Telomere-to-Single Copy Gene*
T:	*telomere repeat copy number*
TCAB1:	*telomerase* Cajal *body protein* 1
TERC:	integraal telomerase RNA
TERRA:	telomerische *repeat-containing* RNA's
TERT:	telomerase reverse transcriptase proteïne
T-loop:	telomerische loop
T-OLA:	*Telomere Oligo Length Assistance*
TRF:	*terminal restriction fragment*
TRF1/TRF2:	TTAGGG *repeat binding factor* 1/2
WGS:	*Whole Genome Sequence*

Samenvatting

De telomeren, herhalingen van een specifiek hexameer op de uiteindes van chromosomen, ondergaan een graduele verkorting tijdens het leven van een individu door celdelingen en oxidatieve stress. Vanaf dat er een kritisch korte telomeerlengte bereikt wordt is het echter niet langer mogelijk om een D-loop-T-loop te vormen wat de herkenning van een DNA-breuk en replicatieve senescentie in de hand werkt. Ondanks de verkorting van de telomeren reeds duidelijk in verband gebracht werden met hart-en vaatziekten zijn de statististische associatie en biomerkerwaarde eerder beperkt. De mogelijke verklaring die hier onderzocht wordt houdt in dat de meest gebruikte meetmethodes voor telomeerlengte (TRF *Southern blot* en qPCR) ook vatbaar zijn voor de aanwezigheid van gedegenereerde, niet-functionele telomeren. Ondanks dit eerder werd aangetoond werden de gevolgen hiervan nog niet goed ingeschat. Mede doordat deze gedegenereerde *repeats* met een verschillende frequentie voorkomen tussen individuen maakt dit de hypothese van aspecifieke TRF-en qPCR-metingen plausibel.

Recent werd aangetoond dat uit de aanwezigheid van telomerische *repeats* binnen *whole genome sequence* data er een gemiddelde telomeerlengte kan worden berekend. Dit zorgt ervoor dat deze methodologie geschikt is om op een exploratieve basis de niet-functionele telomeren in kaart te brengen via een bio-informatica benadering. Door gebruik te maken van vrij beschikbare data uit het 1000 Genomes Project worden profielen opgesteld van gedetecteerde SNPs en hexameren, en worden telomeerlengtes vergeleken tussen de verschillende aanwezige populaties binnen de dataset. Er konden kleine variaties teruggevonden worden, maar door de overheersende technische variatie is verder onderzoek noodzakelijk. Wel steunen de resultaten het idee dat telomerische mutaties frequenter voorkomen dan vroeger ingeschat, en dat dit tot interindividuele verschillen in “functionele” telomeerlengte kan leiden die niet detecteerbaar is met de klassieke methodes.

1. Literatuurstudie

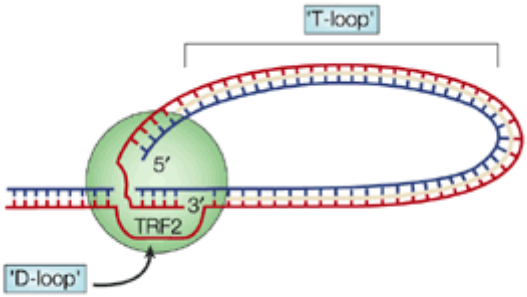
1.1. Inleiding: telomeren, telomeerlengte en zijn determinanten

De telomeren zijn de natuurlijke uiteindes van lineaire chromosomen die aan de hand van hun complexe structuur door de cel van dubbelstrengige breuken kunnen worden onderscheiden. Hierdoor hebben telomeren in de eerste plaats een beschermende functie tegen exonuclease degradatie en *end-to-end* fusie van chromosomen. Ze bestaan uit nucleoproteïnecomplexen met als DNA component species-specifieke repeats die, in de mens, variëren in lengte van typisch 5kb tot 15kb. Humane telomeren bestaan uit TTAGGG hexameren maar er zijn ook andere *repeats* terug te vinden met een lagere frequentie: TTGGGG, TGAGGG en TTTAGGG (Allshire, et al., 1989; Moyzis, et al., 1988). Studies hebben uitgewezen dat telomerisch DNA tot transcriptie wordt gebracht met vorming van telomerische *repeat-containing* RNA's (TERRAs), maar hun functie is vooralsnog relatief onbekend (Azzalin, et al., 2007; Reig-Viader, et al., 2013).

Normale humane cellen die in cultuur gebracht worden hebben een beperkt aantal delingscycli. Dit fenomeen is gekend als de Hayflick limiet en is een rechtstreeks gevolg van het *end-replication problem*. Hier worden de telomeren bij elke celdeling korter omdat het DNA-polymerase het 3'-einde van de DNA-streng niet volledig kan repliceren. Bij het bereiken van een kritisch korte telomeerlengte gaat de complexe structuur verloren en wordt het telomeer disfunctioneel. Dit zorgt voor de activatie van celcylcus *checkpoints* wat tot replicatieve senescentie leidt. Studies hebben aangetoond dat het niet de gemiddelde telomeerlengte is, maar een individueel kort telomeer dat de celcyclus *checkpoint* activeert (Hemann, et al., 2001).

Stamcellen, geslachtscellen en de meeste kankercellen kunnen deze verkorting tegengaan door de activatie van een enzym, telomerase. Dit is een ribonucleoproteïne reverse transcriptase complex dat op basis van zijn RNA component hexanucleotide repeats kan toevoegen aan de telomerische 3'-uiteinden (zie 1.3: Telomerase) (Shay & Wright, 2010). In normale somatische human cellen wordt telomerase volledig of deels onderdrukt waardoor de telomeren bij elke celdeling inkorten (Bryan & Cech, 1999).

Ondermeer het *end-replication problem* zorgt voor het ontstaan van een 3' *overhang* met een lengte van 100-200 nucleotiden (Wright, et al., 1997). Deze *overhang* nestelt zich terug in het eerder dubbelstrengig DNA-gebied en vormt zo een *displacement-loop* (D-loop) en telomerische-loop (T-loop) (Figuur 1). Er zijn sterke aanwijzingen dat specifiek deze structuren voorkomen dat de 3' *overhang* herkend wordt als een DNA-breuk. Ze kunnen ook fungeren als primer voor het telomerase of de toegankelijkheid voor transcriptiefactoren en proteïnes regelen. Er werd ontdekt dat een telomeerloop zover als 10Mb van het chromosoomeinde kan reiken waarbij de chromatine-opvouwing en genexpressie kan beïnvloed worden (Kalmbach, et al., 2013; Robin, et al., 2014). Het amalgaam van telomeer-geassocieerde proteïnes, zoals TRF2, die interageren met deze loops en een stabiliserende functie hebben wordt het shelterine complex genoemd (zie 1.2: Shelterine/telosoom) (Greider, 1999; Griffith, et al., 1999; Stansel, et al., 2001).



Figuur 1: De vorming van een D-loop en T-loop door de innesteling van een 3' G-rijke overhang in het duplex gedeelte van de telomeren. Dit wordt in stand gehouden door verschillende telomeer-geassocieerde proteïnes zoals TRF2 (Mathon & Lloyd, 2001).

De rol van telomeerbiologie in de Hayflicklimiet heeft geleid tot de hypothese dat telomeerlengte een biomerker is voor veroudering, en deze wordt ondersteund door verschillende factoren. Zo is er de afname in telomeerlengte aan 20-60bp per jaar in perifere bloedleukocyten bij cross-sectioneel onderzoek (Vaziri, et al., 1994; von Zglinicki & Martin-Ruiz, 2005). De telomeren van volwassen mannen zijn ook korter dan bij vrouwen, wat het verschil in levensverwachting weerspiegelt. Een mogelijke verklaring hiervoor is een snellere afname van telomeerlengte bij mannen, bv. omdat de hogere oestrogeen-waarde bij vrouwen de activiteit van telomerase stimuleert (Jeanclos, et al., 2000; Nawrot , et al., 2004; Bekaert, et al., 2007; Farzaneh-Far, et al., 2010; De Meyer, et al., 2011; Gardner, et al., 2014). Er zijn echter ook aanwijzingen dat vrouwen met langere telomeren geboren worden (Factor-Litvak, et al., 2016).

Systemische inflammatie is geassocieerd met kortere telomeerlengte, op z'n minst in leukocyten, wat vermoedelijk kan verklaard worden door het hogere aantal celdelingen (Sohal & Dubey, 1994; Wong, et al., 2014). Inflammatie kan ook leiden tot apoptose en necrose, en bijgevolg een vrijgave van reactieve zuurstofverbindingen, i.e. oxidatieve stress. Oxiderende agentia hebben een potentieel schadelijke invloed op het genoom onder de vorm van dubbelstrengige breuken en chemische modificaties in telomerische sequenties. In combinatie met een niet-optimaal herstel van deze breuken versnelt dit de verkorting van de telomeren (Petersen, et al., 1998; von Zglinicki, et al., 2000; von Zglinicki & Martin-Ruiz, 2005). Oxidatieve stress, al dan niet geassocieerd met inflammatie is dus vermoedelijk de belangrijkste oorzaak voor telomeerlengteverkorting *in vivo*. Ook deze fenomenen werden gelinkt aan de langere telomeren in vrouwen, aangezien deze minder reactieve zuurstofbindingen produceren en ook oestrogeen antioxidatieve eigenschappen vertoont (Nawrot , et al., 2004; Gardner, et al., 2014). Een ongezonde levensstijl heeft tevens een invloed op het verkorten van de telomeren, en steunt vermoedelijk op dezelfde moleculaire principes van inflammatie en oxidatieve stress, hoewel ook een hogere telomerase-activiteit werd voorgesteld (Ornish, et al., 2008; De Meyer, et al., 2011). Er wordt verondersteld dat het slechts de omgevingsfactoren in een vroege ontwikkeling zijn die bepalend zijn voor de telomeerlengte (Hjelmborg, et al., 2015).

Ondanks bovenstaande factoren vormt erfelijkheid de belangrijkste determinant van telomeerlengte. Reeds tijdens de vroege ontwikkeling van het embryo wordt telomeerlengte grotendeels vastgelegd. Zo hebben studies met tweelingen aangetoond dat individuele telomeerlengteverschillen voor een groot deel te wijten zijn aan erfelijkheid (Slagboom, et al., 1994; Graakjaer, et al., 2003; Bekaert, et al., 2005). Hierdoor hebben verschillende weefsels van eenzelfde individu ook heel gelijkaardige telomeerlengtes (Takubo, et al., 2002). Echter,

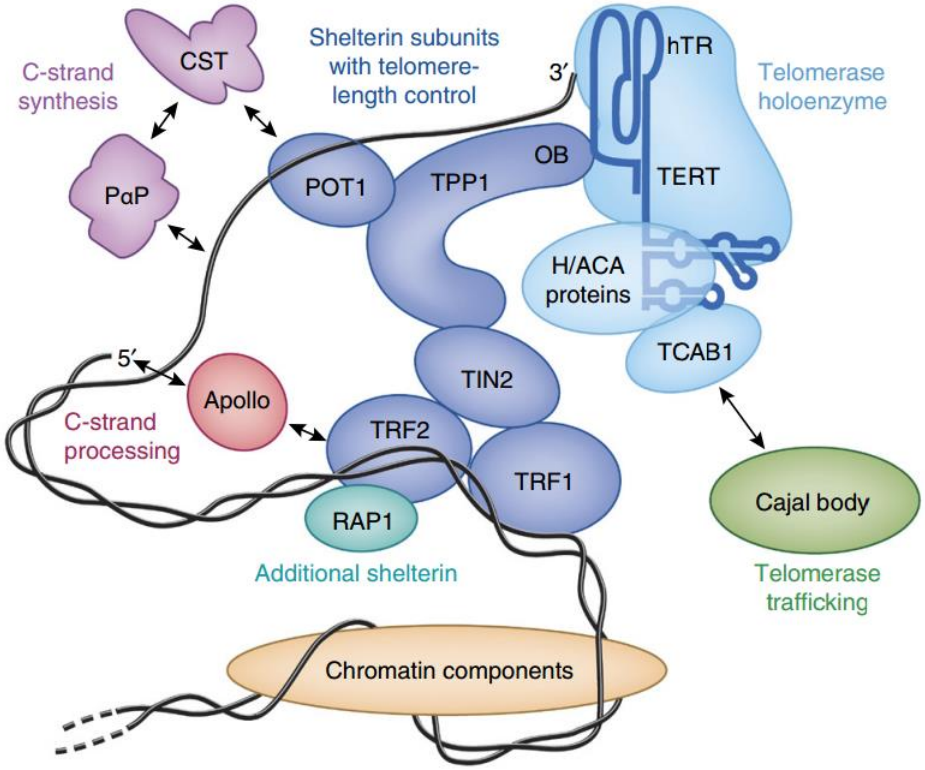
deze erfelijkheid wordt slechts deels genetisch bepaald, waarbij ondermeer varianten in telomerase en andere eiwitten betrokken in telomeerbiologie in beperkte mate telomeerlengte beïnvloeden. Hoewel normale genetische variatie een beperkt effect heeft, kan de aanwezigheid van pathogene mutaties wel de levensduur van het individu aantasten (Vulliamy, et al., 2001). De belangrijkste erfelijke factor is vermoedelijk het telomeerlengteprofiel van de gameten die de zygote zullen vormen. Tijdens de vroege ontwikkeling worden deze telomeerlengtes opnieuw verlengd door telomerase, maar de verdere impact van genetische verschillen lijkt hier beperkt (De Meyer, et al., 2014).

Een andere relevante telomeerlengtedeterminant is het het *paternal age effect*, waarbij kinderen met oudere vaders bij hun geboorte ook een langere telomeerlengte hebben. De telomeerlengte zou toenemen met ca. 15bp per jaar dat de vader ouder is bij conceptie (Unryn, et al., 2005; De Meyer, et al., 2007). Het feit dat spermacellen een langere telomeerlengte hebben met een hogere leeftijd van de vader is aanvullend bewijs voor de overerfbaarheid van telomeerlengte los van genetica (Unryn, et al., 2005). Daarnaast werd ook etniciteit als determinant van leukocytentelomeerlengte geïdentificieerd, met bv. langere telomeerlengtes bij Afro-Amerikanen dan bij blanke personen, hoewel de onderliggende oorzaak misschien een cumulatief effect is van één van bovenstaande factoren, bv. een cumulatief *paternal age effect* (De Meyer, et al., 2007; Hunt, et al., 2008).

1.2. Shelterine/telosoom

De telomerase-activiteit gebeurt slechts in associatie met het proteïnecomplex shelterine (telosoom). Dit complex is opgebouwd uit een zestal basisproteïnes: TRF1, TRF2, POT1, TIN2, TPP1 en RAP1 (Figuur 2). Hiervan zijn het enkel de homodimeren TRF1 en TRF2 (TTAGGG *repeat binding factor*) die binden met het telomerische DNA (Nishikawa, et al., 2001; Cech, 2004; Court, et al., 2005; Sfeir & de Lange, 2012). Ze hebben beiden een carboxyterminaal DNA-bindend motief dat een hoge homologie heeft met het Myb DNA-bindend motief. Daarnaast hebben beide proteïnes ook een aminoterminaal TRFH domein dat ingrijpt op de homodimerisatie, en hebben ze eenzelfde α-helicale structuur (Broccoli, et al., 1997; Fairall, et al., 2001). Ondanks de gelijkenissen in structuur hebben de proteïnes een andere functie. Overexpressie van TRF1 zorgt voor een verkorting in telomeerlengte, terwijl bij een dominant-negatieve mutant van TRF1 er een verlenging in telomeerlengte werd waargenomen. Dit wijst erop dat TRF1 een negatieve regulator is van de telomeerlengte (Smogorzewska & de Lange, 2004). Een dominant-negatieve mutant van TRF2 zorgt ervoor dat cellen de telomerische eindes herkennen als DNA-breuk, de 3'-*overhang* verliezen en de telomeren fuseren, wat uiteindelijk tot p53-afhankelijke apoptose leidt. Dit wijst op een rol van het TRF2 in het beschermen van de chromosoomeindes (van Steensel, et al., 1998; Karlseder, et al., 1999).

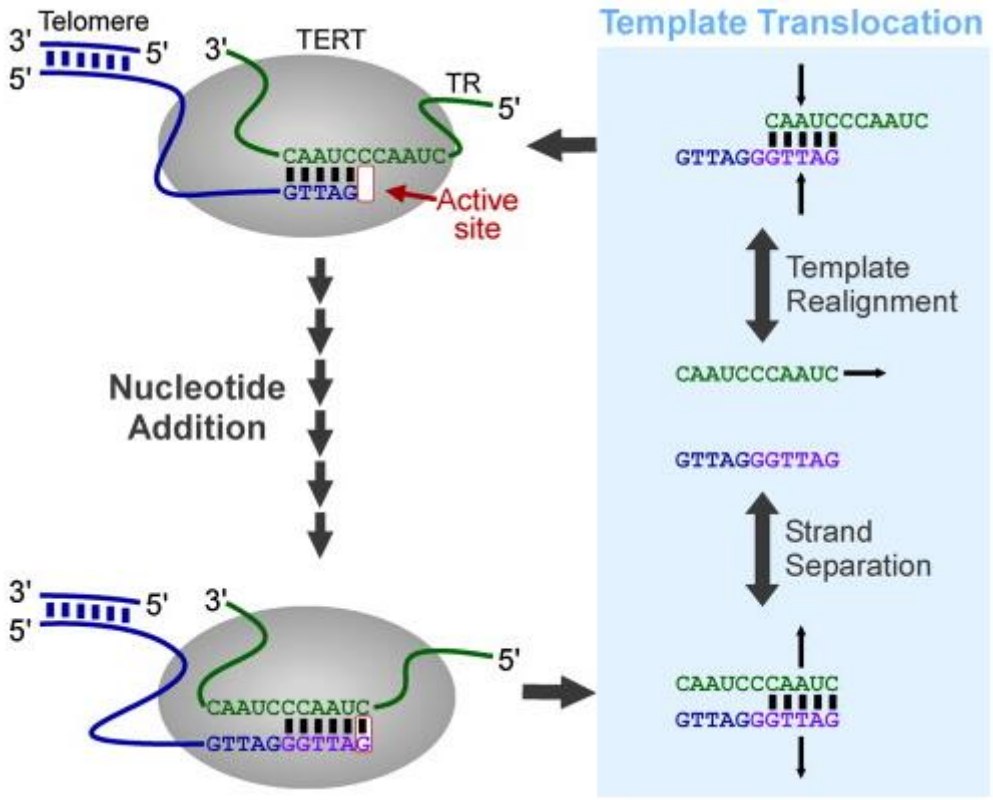
TRF1 en TRF2 vertonen een hoge dynamiek in hun voorkomen en verblijftijd (Mattern, et al., 2004). Post-translationele modificaties van TRF1 zoals poly-ADP-ribosylatie, ubiquinatie, sumoylatie of fosforylatie kunnen de associatie met het telomerische DNA beïnvloeden. Bij TRF2 werden fosforylatie en sumoylatie eerder beschreven (Smith, et al., 1998; Diotti & Loayza, 2011). POT1 bindt met de telomerische DNA primer en TPP1 bindt met POT1, TIN2 en het telomerase reverse transcriptase proteïne (TERT). Dit complex houdt de DNA-primer dichtbij het telomerase en zorgt voor een lagere primerdissociatie en een hogere translocatie-efficiëntie (Cristofari, et al., 2007; Latrick & Cech, 2010; Podlevsky & Chen, 2012). Het RAP1 (Human Repressor Activator Protein 1) vormt een complex met TRF2 en verhoogt de selectiviteit van TRF2 met het telomerisch DNA (Janouskova, et al., 2015).



Figuur 2: Interactie van het humane shelterine en de telomerase subunits. De TRF1 en TRF2 proteïnes zorgen voor een koppeling van de dubbelstrengige telomeerrepeats met het shelterinecomplex. POT1 is op zijn beurt gebonden verbonden met TRF1 en TRF2 via de proteïnes TIN2 en TPP1. Een additioneel shelterine RAP1 kan rechtstreeks binden aan TRF2. Het katalytisch actief humaan telomerase bevat het TERT, het TER en de H/ACA proteïnes dyskerine, NHP2, NOP10 en GAR1. TCAB1 interageert met het telomerase via het H/ACA-domein en medieert de ribonucleoproteïne in de Cajal-lichamen. TRF2 bindt Apollo, een C-streng 5'-3' exonuclease (Hockemeyer & Collins, 2015).

1.3. Telomerase

Telomerase is een ribonucleoproteïne enzym met twee katalytisch essentiële subeenheden: het TERT en het integraal telomerase RNA (TERC) (zie Figuur 3). Het humane TERT-gen is gelegen op chromosoom 5, het humane TERC-gen op chromosoom 3. Het TERT bevat de katalytische site voor DNA synthese en gebruikt een *template* van het TERC om telomerische DNA-*repeats* te vormen (Greider & Blackburn, 1989). De initiatie van dit proces gebeurt door basepaarvorming tussen het 3'-einde van de telomerische DNA primer en de 5'-regio van het RNA *template*. Het humaan telomerase plaatst vervolgens via reverse transcriptie een 5'-GGTTAG-3' hexameer op het 3'-einde van de DNA primer. Voor een volgende additie van telomerische DNA-repeats wordt het *template* geregenereerd door een translocatie van de RNA-streng. Deze translocatie vindt plaats buiten de actieve site van het telomerase en vormt de snelheidsbepalende stap (Morin, 1989; Greider, 1991; Qi, et al., 2011). Wanneer het einde van het *template* wordt bereikt gebeurt er een regeneratie van dit *template* voor een volgende synthese, of zal het telomerase dissociëren van het DNA (Podlevsky & Chen, 2012).



Figuur 3: De cyclus waarbij het humaan telomerase een hexamere telomeerrepeat toevoegt aan het 3'-einde van de DNA-primer wordt opgedeeld in een nucleotide additie (links) en een template translocatie (rechts). Op de figuur zijn het TERT (cirkel), het TER (op de figuur "TR"), de DNA primer (GTTAG) en de toegevoegde zes nucleotiden te zien (Podlevsky & Chen, 2012).

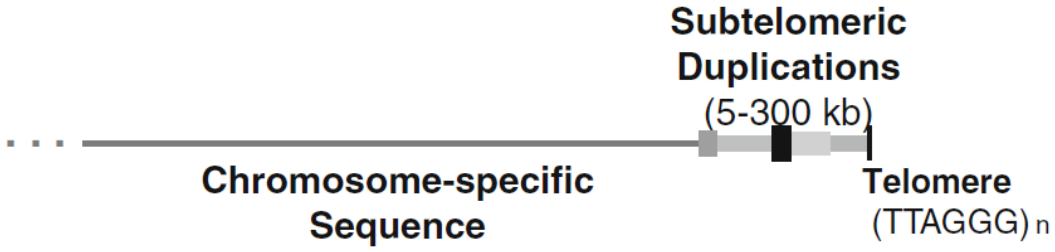
Het TERT-proteïne bestaat uit vier geconserveerde structurele domeinen: het telomerase N-terminaal domein, het telomerase RNA-bindend domein, het reverse transcriptase en een C-terminale extensie. Er zijn verschillende DNA-motieven aanwezig binnen deze domeinen die ervoor zorgen dat de telomerische DNA-primer beter behouden blijft (Wyatt, et al., 2007). Er zijn ook motieven aanwezig die een belangrijke rol spelen in de templatetranslocatie (Lue, et al., 2003; Qi, et al., 2011; Podlevsky & Chen, 2012). Bij alle species bevat het TERC twee geconserveerde structuren: een *template/pseudoknot* domein en een CR4/5 domein. Het *template/pseudoknot* domein bevat een *triple helix* en is in staat het *template* op de actieve site te positioneren (Lin, et al., 2004; Shefer, et al., 2007). Het humane TERC bevat een bijkomend H/ACA domein dat als proteïnebindend domein fungeert en essentieel is voor de telomerase biogenese en lokalisatie (Mitchell, et al., 1999; Armanios, et al., 2005; Parry, et al., 2011; Podlevsky & Chen, 2012). Er zijn reeds verschillende deleties en nucleotidesubstitutie mutaties ontdekt in het TERC en TERT gen. Verschillende van deze mutaties leiden tot haploinsufficiëntie en *in vivo* verkorting van telomeerlengte, ook over generaties (Savage & Bertuch, 2010).

De biogenese van het telomerase is afhankelijk van de assemblage van beide subeenheden tot een actief ribonucleoproteïne enzym. De expressie van het TERT proteïne gebeurt door een klassieke opeenvolging van mRNA transcriptie, maturatie en een cytoplasmatische translatie. Het proteïne wordt vervolgens naar nucleoli en Cajal-lichamen gebracht. Het TERC wordt gevormd als een precursor RNA polymerase II transcript. Het RNA helicase RHAU bindt op het 5'-einde terwijl een complex van dyskerine, NHP2, NOP10 en GAR1 op het 3'-einde bindt. De RNA-streng wordt getrimd, intern gemodificeerd en het mature TERC wordt naar Cajal-lichamen gebracht waar beide subeenheden een matuur telomerase vormen. Tenslotte wordt dit telomerase getransloceerd naar de telomeren om zijn functie uit

te voeren (Zhu, et al., 2004; Collins, 2006; Podlevsky & Chen, 2012). Vooral het TPP1/POT1 complex zou noodzakelijk zijn voor deze translocatie. Daarnaast zijn er ook aanwijzingen dat TCAB1 (telomerase Cajal body protein 1), dat in grote mate aanwezig is in Cajal-lichamen, een belangrijke rol hierin heeft (Xin, et al., 2007; Venteicher, 2009).

1.4. Subtelomeren

Subtelomeren zijn het gebied tussen de unieke chromosoomspecifieke sequenties en de telomerische *repeats* aan het einde van de chromosomen. Over het algemeen kunnen ze worden onderverdeeld in een distaal (terminaal) gebied en een proximaal gebied (Figuur 4). De distale regio bestaat uit segmentale duplicaties die een hoge sequentiegelijkenis (>90%) kunnen hebben en variëren in grootte (1kb tot >200kb). Deze duplicaties kunnen uniek zijn, gedeeld met pericentromerische *repeat* regio's of gedeeld met interstitiële chromosomale loci (Riethman, et al., 2001). De proximale gebieden bestaan uit chromosoom-specifieke sequenties. In tegenstelling tot normale polymorfismen in het distale gebied, kunnen variaties in de proximale gebieden leiden tot ziektebeelden. Gekende voorbeelden hiervan zijn de 3q29 microdeletie, deletie van de 9q subtelomeer en 22q13 deletie. Fenotypische kenmerken zijn vooral intellectuele achterstand en geboorteafwijkingen (Flint, et al., 1997; Phelan, et al., 2001; Mefford & Trask, 2002; Harada, et al., 2004; Willat, et al., 2005; Rudd, 2012).



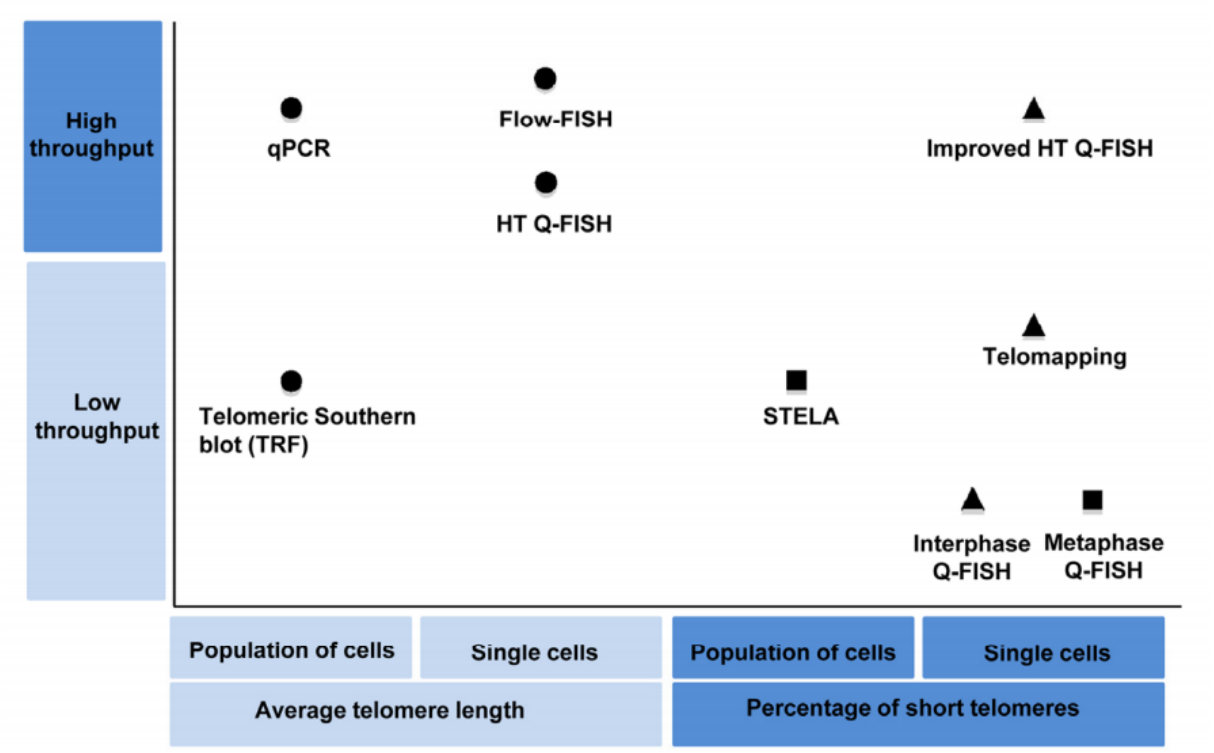
Figuur 4: Schematisch voorbeeld van een chromosoomeinde. Het einde van de chromosoom bestaat uit pure telomerische TTAGGG repeats. Dit wordt begrensd door het distale subtelomeerdomein dat bestaat uit segmentele duplicaties. Het proximale subtelomeerdomein zijn chromosoom-specifieke sequenties (Rudd, 2012).

Het precieze mechanisme waarbij subtelomerische sequenties gedupliceerd en verspreid worden is niet gekend maar is waarschijnlijk gebaseerd op meerdere verschillende processen als translocaties, recombinaties en duplicaties. Ook over de functionele significantie van de subtelomeren bestaan er enkele hypotheses. Doordat het zeer dynamische regio's zijn, zou dit een belangrijk mechanisme kunnen zijn dat diversiteit van genen schept en een snelle adaptieve evolutie toelaat. Bewijs hiervan is terug te vinden bij P. *falciparum* dat het immuunsysteem van de gastheer ontwijkt door antigenen te variëren die tot expressie komen op de celmembraan van geïnfecteerde cellen. De 50 genen in de var-genfamilie die voor deze antigenen coderen bevinden zich bijna allemaal in de subtelomeren (Corcoran, et al., 1988; Trask, et al., 1998; Mefford & Trask, 2002).

Een andere hypothese is dat de subtelomeren een manier zijn om het telomeer-positie effect tegen te gaan. Hierbij worden genen die dicht aanliggen bij de telomeren transcriptioneel *gesilenced* (Baur, et al., 2001; Mefford & Trask, 2002). Een afwezigheid van telomerase-activiteit zorgt ervoor dat cellen een beperkt aantal delingscycli hebben. *Drosophila* heeft geen telomeren of telomerase, maar het verdwijnen van eindes van chromosomen wordt tegengegaan door retrotranspositie van HeT-A en TART elementen. Mogelijks bestaat er in gistcellen en humane cellen een analoog systeem voor telomeeronderhoud via recombinatie van subtelomerische sequenties (Mason & Biessmann, 1995; Mefford & Trask, 2002).

1.5. Lengtebepaling telomeren

Er bestaan meerdere technieken om telomeerlengte te meten, voor zowel individuele chromosomen als het gemiddelde voor een collectie cellen (bv. leukocyten) (Figuur 5). De belangrijkste en meest gebruikte methoden worden opgesomd met hun voor-en nadelen.



Figuur 5: Classificatie van de belangrijkste methoden bij het meten van de telomeerlengte (Vera & Blasco, 2012).

1.5.1. TRF

Terminal Restriction Fragment (TRF) analyse wordt vaak aangegeven als de gouden standaardmethode voor het bepalen van de telomeerlengte. Opgezuiverd genomisch DNA wordt behandeld met een mengsel van restrictie-enzymes die geen restrictiesite hebben in telomerische *repeats*. *Hinf*1/*Rsa*1 maar ook *Hph*I/*Mn*II zijn beide vaak gebruikte combinaties van frequente knippers die het niet-telomerische DNA digereren (Figuur 6). Hierbij blijven de telomerische fragmenten zelf intact. Vervolgens gebeurt er een scheiding op grootte via een agararose gelelektroforese waarbij het DNA naar de positieve pool migreert. Hierna wordt er een *Southern blot* gedaan waarbij de fragmenten worden getransfereerd naar een nylon of nitrocellulosemembraan en er via een gelabelde probe specifiek voor telomeersequenties de gemiddelde lengte van telomeren kan worden bepaald (Kimura, et al., 2010).

Figuur 6: restrictiesites van de vaakst gebruikte enzymen bij TRF-methodes.

De voordelen van TRF zijn dat dit een veelgebruikte techniek is en er geen dure reagentia of materiaal nodig zijn. Het is ook een erg precieze en nauwkeurige techniek. Dit maakt het erg geschikt voor *proof-of-concept* studies. Daartegenover staat er dat TRF arbeids-en

tijdsintensief is, moeilijk te kwantificeren en er relatief veel DNA nodig is van goede kwaliteit (minimum 2 µg). Er wordt ook slechts een gemiddelde telomeerlengte bepaald voor het volledige genoom. Korte telomeren die aanwezig zijn op een klein aantal chromosomen zijn moeilijk te detecteren omdat deze gemaskeerd worden door de aanwezige lange telomeren (met een groter signaal). Dit vormt een groot nadeel voor ouderdomsstudies. Daarnaast worden ook subtelomerische fragmenten als telomerisch beschouwd met deze methode, aangezien deze niet noodzakelijk veel restrictiesites bevatten. Interindividuele variabiliteit in de subtelomerische sequenties kan bovendien voor een *bias* zorgen. (Tollefsbol, 2007; Kimura, et al., 2010; Montpetit, et al., 2014).

1.5.2. qPCR

Quantitative PCR (qPCR) is een vorm van PCR waarbij in *real-time* kan gevolgd worden hoeveel DNA er gevormd wordt door gebruik te maken van dsDNA-bindende kleurstof zoals Sybr Green of een *molecular beacon* zoals de Taqman-probe. Deze laatste is een enkelstrengige molecule met interne complementariteit waarbij het ene uiteinde de fluorescentie van het andere uiteinde onderdrukt. Als de probe kan hybridiseren, gebeurt er een openplooiing en vindt er fluorescentie plaats bij excitatie. Voor elk staal wordt een C_T-waarde bepaald, dit is het cyclusnummer waarbij de gevormde fluorescentie boven een opgelegde drempelwaarde komt. Door gebruik te maken van standaardurves met gekende verdunningen van target-DNA kan er kwantitatief gewerkt worden (Gheysen, et al., 2014).

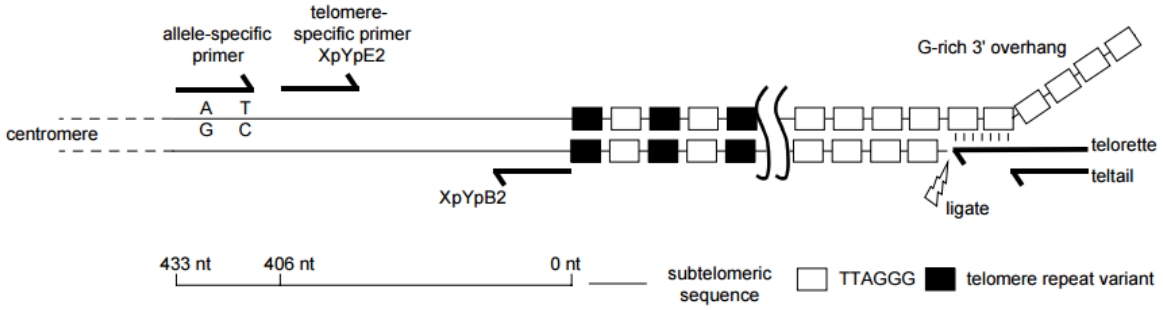
Voor telomeerlengtebepaling wordt er een *Telomere-to-Single Copy Gene* (T/S) ratio opgesteld van de *telomeer repeat copy number* (T) en de *single copy gene copy number* (S). De mate waarin de ratio van een staal dan verschilt met de ratio van een referentiestaal is proportioneel met de gemiddelde telomeerlengte. Voor de T-reactie zouden perfect *matching primers* voor de repetitieve telomerische DNA-sequentie (voornamelijk) tot de vorming van primerdimeren leiden, en een foutieve inschatting van de telomeerlengte. Dit probleem werd opgelost door gebruik te maken van primers die *mismatches* bevatten, waardoor de vorming van primerdimeren sterk bemoeilijkt wordt (de *mismatches* komen op verschillende plaatsen voor) terwijl de primers wel nog voldoende affiniteit hebben voor de eigenlijke telomerische sequentie.

Als keuze voor het *single copy gene* is het belangrijk dat de lengte van het *single copy gene* dat PCR-amplificatie ondergaat, identiek is in alle stalen. In een studie van Cawthon (2002) werd hiervoor het *36B4*-gen genomen, dat codeert voor ribosomaal fosfoproteïne PO. Door limitaties van precisie is er in deze methode echter steeds variatie in hoeveelheid DNA tussen de “T” en “S” reacties. De *monochrome multiplex quantitative* PCR (MMqPCR) omzeilt dit probleem door in eenzelfde tube te werken (Cawthon, 2009). Een verdere aanpassing van de qPCR methode is de *absolute telomere length* (aTL) *quantitation*. Hier is het mogelijk om een schatting te doen van telomeerlengte op basepaarniveau door gebruik te maken van een standaardcurve van gekende telomeerlengtes (Callaghan & Fenech, 2011). In tegenstelling met TRF is qPCR sneller en efficiënter door zijn *high-throughput*. Door de PCR-stap is er ook slechts enkele nanogram DNA nodig. Volgens een studie van Aviv et al. is er echter een hoge coëfficient van variatie (CV) van 6,45% voor qPCR waar de CV voor *Southern blots* slechts 1,74% is (Cawthon, 2002; Aviv, et al., 2011; Montpetit, et al., 2014).

1.5.3. STELA

TRF en qPCR-gebaseerde methoden hebben als gemeenschappelijk kenmerk dat ze enkel een gemiddelde telomeerlengte meten. Sinds telomeren met een kritisch korte

telomeerlengte een belangrijke aanleiding zijn tot cellulaire senescentie en apoptose, is het noodzakelijk dat er methoden zijn die deze telomeren op een chromosoomspecifieke wijze kunnen meten (Hemann, et al., 2001). *Single Telomere Length Analysis* (STELA) is een PCR-gebaseerde methode die dit toelaat (Figuur 7). Hier wordt een linker ontwikkeld bestaande uit zeven basen die TTAGGG homologie bezitten, en een niet-complementaire staart met een lengte van twintig nucleotiden. Deze linker zal hybridiseren met het 3' overhangend deel van het telomeer. Vervolgens wordt deze linker geligeerd aan het 5' eind van de complementaire streng waardoor de telomeer een *tag* bezit. Dit maakt het mogelijk om een *long-range* PCR uit te voeren waarbij een eerste primer bindt aan de niet-complementaire staart en een chromosoom-specifieke tweede primer in het subtelomerisch gebied (Baird, et al., 2003).



Figuur 7: Principe van STELA. Een linker kan worden geligeerd aan het 5'-eind van de telomeer waarna een PCR kan worden uitgevoerd met specifieke primers (Baird, et al., 2003).

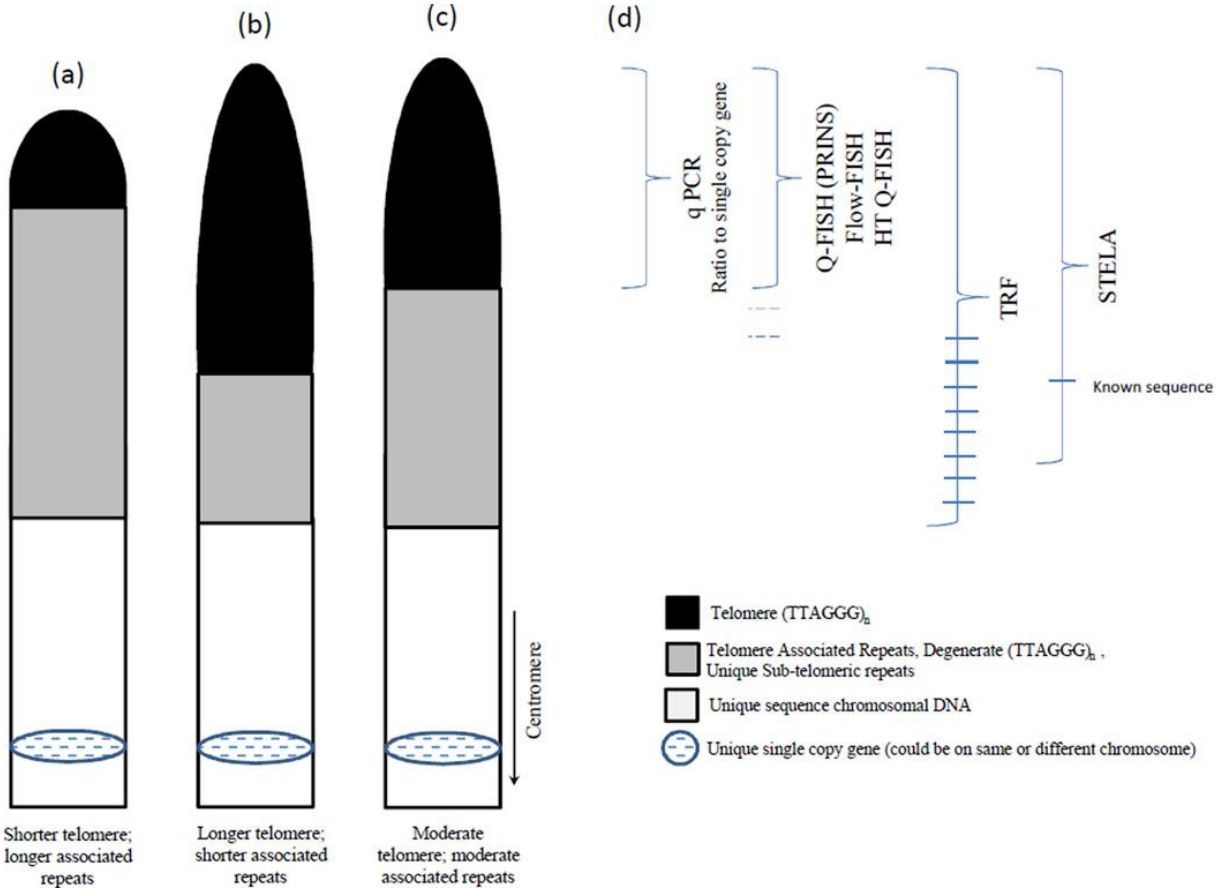
Deze methode was initieel chromosoom-specifiek of chromosoomarm-specifiek voor een kleine subset van chromosomen (XpYp,2p,11q,12q en 17p), door de noodzaak om primers voor individuele subtelomerische regio's te kunnen maken. *Universal* STELA is een aanpassing van de STELA methode die toelaat om telomeerlengte te bepalen van elk chromosoom. Hierbij wordt het geëxtraheerd DNA geknipt door *Mse*I en *Nde*I-restrictie-enzymes. Een dubbelstrengig oligo dat een niet-complementaire staart bevat, wordt geligeerd aan de *sticky ends* van het geknipt DNA. Volgens het STELA-principe wordt er een linker aan het 5'-eind van de telomeer geligeerd. De oligo is zo ontworpen dat amplificatie van intra-genomische fragmenten onderdrukt wordt door het principe van *suppression*-PCR (Bendix, et al., 2010).

STELA en *Universal* STELA vereisen geen specifiek materiaal en er is geen nood aan een grote hoeveelheid genomisch DNA. Een groot nadeel is dat lange telomeren of een afwezigheid van telomeer niet gedetecteerd kunnen worden. De methodes zijn ook beiden *low-throughput* en arbeidsintensief. Tenslotte kan een te grote hoeveelheid *template* DNA leiding geven tot veel onafgewerkte amplicons, die te zien zijn op de gel als smeer (Montpetit, et al., 2014).

1.5.4. Andere methodes voor het bepalen van telomeerlengte

Quantitative Fluorescence In Situ Hybridization (Q-FISH of telomapping) is een kwantitatieve techniek voor het meten van telomeersequenties in metafase of interfase chromosomen. Metafase of interfase cellen worden hierbij gefixeerd en gehybridiseerd met gelabelde *peptide nucleic acid* (PNA) probes via een geoptimaliseerde FISH techniek. Vervolgens wordt er via digitale fluorescentiemicroscopie beelden genomen waaruit informatie over telomeerfluorescentie en chromosoomidentificatie geëxtraheerd kan worden (Poon & Lansdorp, 2001). *Fluorescent In Situ Hybridization* (*Flow*-FISH) is een methode die op Q-FISH verderbouwt en de hybridisatie van een telomerische $(CCCTAA)_3$ PNA probe

combineert met flowcytometrie. In *Flow*-FISH wordt er dan ook een gemiddelde telomeerlengte bepaald van de celpopulatie (Baerlocher, et al., 2006). De *Hybridization Protection Assay* (HPA) is een techniek die steunt op het vergelijken van het aantal telomeerrepeats met het aantal Alu-herhalingen in een specimen. Door variatie in Alu-herhalingen tussen stalen wordt deze methode weinig gebruikt (Lin & Yan, 2007; Montpetit, et al., 2014). In tegenstelling tot de eerder beschreven methoden die de volledige telomeerlengte schatten, zijn er ook technieken die de lengte van de telomerische 3'-overhang kwantificeren. Voorbeelden zijn *Telomere Oligo Length Assistance* (T-OLA), *G-tail* HPA, *overhang protection assay* (OPA), *single-strand electron microscopy*, *Primer-Extension Nick Translation* (PENT) en *double-strand specific nuclease* (Montpetit, et al., 2014). Figuur 8 toont een overzicht van de meest gebruikte technieken en de regio's die hierbij gemeten worden. Tot slot werden ook een aantal methodes ontwikkeld die gebruik maken van DNA-sequenering, deze worden in het volgend hoofdstuk besproken (zie 1.7: Sequeneringsmethodes).



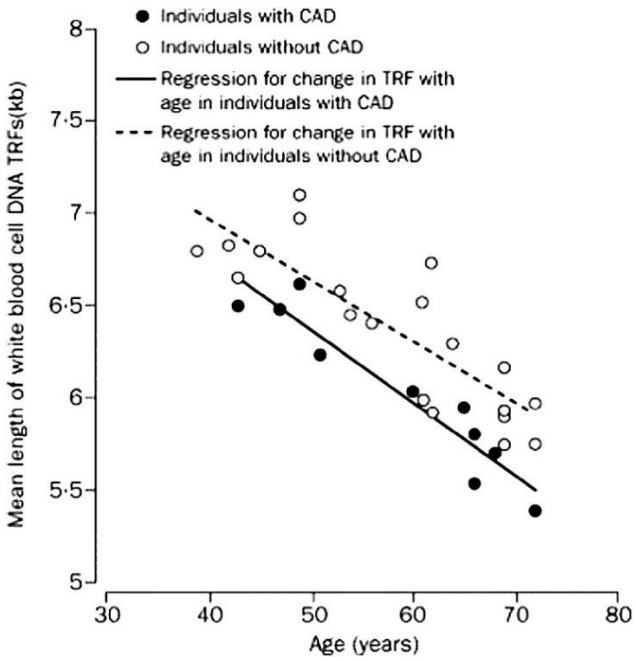
Figuur 8: Schematisch overzicht van de regio's die gemeten worden bij qPCR, FISH-methodes, TRF en STELA. Humane telomeren en subtelomerische regio's verschillen tussen chromosomen en individuen (a-c). In het zwart zijn de pure TTAGGG repeats te zien. Grijswaarden tonen telomeergeassocieerde repeats aan, gedegenereerde $(TTAGGG)_n$ repeats en unieke subtelomerische repeats. Witte waarden zijn unieke chromosomaal-specifieke sequenties. TRF meet de pure telomeren en een stuk van de subtelomeren, afhankelijk van de mogelijke variaties hierbinnen. De STELA assay gebruikt ook sequenties van de subtelomerische regio, maar is sequentie-specifiek. Q-FISH methodologiën gebruiken een probe die specifiek is voor de telomerische regio om de lengte te schatten. De mogelijkheid bestaat echter dat deze probe bindt met een regio binnen de gedegenereerde $(TTAGGG)_n$ repeats. qPCR gebruikt primers en een single copy gene om de telomeerlengte te berekenen (Montpetit, et al., 2014).

1.6. Klinische relevantie

De klinische relevantie van het korter worden van de telomeren is aangetoond door verschillende ziektes met een grote variatie in symptomen. Het bekendste voorbeeld is dyskeratose congenita, dat een gevolg is van een mutatie in het TERT-gen. Zoals hoger vermeld leidt dit tot haploinsufficiëntie die aanleiding geeft tot de geleidelijke verkorting van de telomeerlengte, eventueel over generaties, en bij kritische verkorting het ziektebeeld (Vulliamy, et al., 2001). Mutaties in dit gen verklaren ook zo'n 10% van de patiënten met aplastische anemie (Garcia, et al., 2007; Savage & Bertuch, 2010). Progeroïde syndromen als Werner, Bloom, Hutchinson-Gilford zijn ook reeds in verband gebracht met een snellere verkorting van telomeren. Daarnaast zijn een hogere kans op kanker, osteoarthritis, slechtere wondheling, immuunfunctie, diabetes mellitus en de ziekte van Alzheimer hiermee geassocieerd (Bekaert, et al., 2005; Willeit, et al., 2010; Testa, et al., 2011; Cai, et al., 2013).

In verschillende epidemiologische studies zijn kortere telomeerlengtes aangetoond in patiënten met cardiovasculaire ziektes. Telomeerlengte werd zelfs aangetoond als onafhankelijke predictor voor hartziekte-gerelateerde *events* (bv. hartaanval) (Cawthon, et al., 2003; Brouilette, et al., 2008; De Meyer, et al., 2011). Een studie vond dat personen al met vroege ischemische hartklachten (coronary artery disease, CAD) een kortere gemiddelde telomeerlengte hadden dan gezonde personen van eenzelfde geslacht en leeftijd (Figuur 9) (Nilsson, et al., 2013). Ook werd replicatieve senescentie aangetoond in atherosclerotische lesies, wat een sterke aanwijzing is voor een functionele rol van telomeren in hart-en vaatziekten (Minamino, et al., 2002).

Dit heeft geleid tot de hypothese waarbij individuen met kortere geërfde telomeren een verhoogde kans hebben op vroege senescentie van vasculair weefsel met het ontstaan van atherosclerose. Uit de verschillende epidemiologische onderzoeken blijkt echter dat geërfde kortere telomeren sowieso maar een heel beperkt effect zouden hebben op het voorkomen van atherosclerose en hart- en vaatziekten. Dit maakt het onwaarschijnlijk dat telomeerbiologie een causale rol zou spelen. Er werd daarom geopperd dat systemische verkorting van telomeren als predictieve merker zou kunnen worden gebruikt aangezien deze het cumulatief effect van andere factoren, zoals oxidatieve stress en inflammatie, kan capteren en dus ook van het cardiovasculair ouder worden. Dit vereist uiteraard longitudinale studies (De Meyer, et al., 2009; De Meyer, et al., 2011).



Figuur 9: Gemiddelde telomeerlengtes versus leeftijd van gezonde individuen en individuen die leiden aan CAD (Nilsson, et al., 2013).

Bij de meeste studies wordt er een systemische telomeerlengte berekend van perifere bloedleukocyten. Sinds de variatie in telomeerlengte tussen weefsels veel kleiner is dan de interindividuele variatie, is de telomeerlengte van perifere bloedleukocyten een goede voorspeller van de vasculaire telomeerlengte (Wilson, et al., 2008). Omdat de verkorting van telomeren een longitudinaal proces is en bepaald wordt door de erfelijkheid en omgevingsverschillen, kan een grote interindividuele variatie ervoor zorgen dat er niet genoeg *power* is om significante verschillen waar te nemen (De Meyer, et al., 2011). Om die reden zijn longitudinale studies met een grote *sample size* in tegenstelling tot de vaker voorkomende cross-sectionele onderzoeken erg waardevol. De Asklepios studie is een voorbeeld daarvan waarin de systematische leukocyttelomeerlengte als merker gebruikt wordt voor biologische ouderdom (De Meyer, et al., 2009).

Ook de keuze van onderzoeksmethode speelt een belangrijke rol. Een kleinere meetfout bij TRF *Southern blot* methodes dan bij PCR methodes heeft een belangrijke invloed op de reproduceerbaarheid (Aviv, et al., 2011). Zo is het geslachtsverschil in telomeerlengte enkel bij TRF *Southern blot* methodes te zien, maar niet bij *Real-Time* PCR of *Flow*-FISH methodes. Dit leidt tot een vraagstelling waar het niet duidelijk is of de experimentele variabiliteit van PCR en *Flow*-FISH methodes verantwoordelijk is voor deze bevindingen, of een methodologische *bias* van de *Southern blot* techniek (Gardner, et al., 2014). Sinds de telomeerlengte gerelateerd is met de leeftijd is het ook niet steeds duidelijk of dit een indicator is van een normaal ouderdomsproces of een merker is van een prodromale leeftijdsgerelateerde ziekte (Sprott, 2010; Mather, et al., 2011). Andere aspecten van telomeerbiologie zoals de telomerase activiteit kunnen ook een significante invloed hebben. Om die reden is het ook interessant deze factoren op te nemen in het onderzoek (Nilsson, et al., 2013).

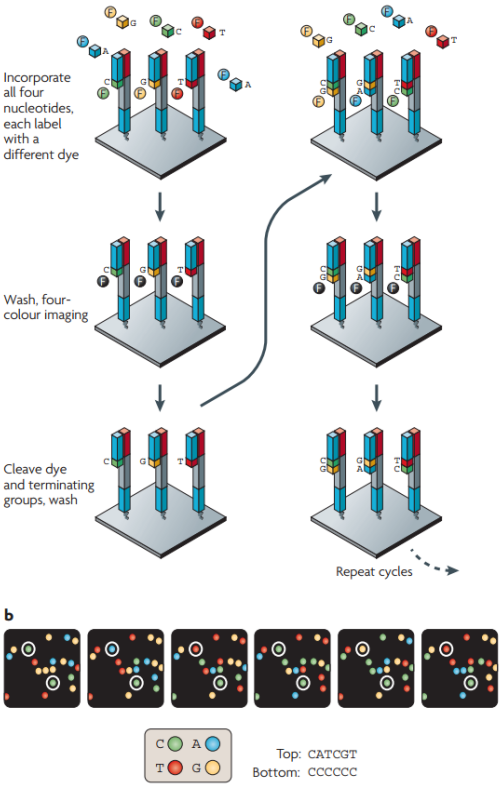
Uit de eerste grootschalige longitudinale metingen bleek echter dat telomeerlengtes slechts beperkt beïnvloed worden door verschillen in telomeerattritie (Benetos, et al., 2013), wat de plausibiliteit van de telomeerattritie-hypothese in hart- en vaatziekten sterk heeft

gereduceerd. Aan de andere kant werd intussen wel aangetoond dat verschillende van de genetische determinanten van telomeerlengte ook geassocieerd zijn met het risico op hart- en vaatziekten, wat - samen met de prospectieve studies - toch weer sterk wijst op causaliteit (Codd, et al., 2013). Concluderend kan dus gesteld worden dat verschillende studies een causale relatie tussen telomeerlengte en hart- en vaatziektes suggereren, maar dat de biomerkerwaarde van telomeerlengte wel zeer zwak is. Er werden nog geen verklarende factoren voor deze duidelijke discrepantie gevonden, maar twee mogelijkheden zijn het negeren van de impact van kritisch korte telomeren in individuele chromosomen (voorheen werd vooral gemiddelde telomeerlengte bepaald) en de kans dat de huidige methodes telomeerlengte niet precies genoeg bepalen door de aanwezigheid van subtelomeren.

1.7. Sequeneringsmethodes

Sinds het verschijnen van automatische *sequencers* in 1987, gebaseerd op Sanger *sequencing*, was het sequeneren van het volledige menselijke genoom in 2000 een werk van lange adem. Een enkele *sequencing run* was slechts in staat om maximaal 96 DNA-molecules te sequeneren van ca. 700 basen. Tegenwoordig bestaan er *second generation*, *high-throughput* sequeneringsmethodes die in parallel miljoenen molecules kunnen sequeneren. De voornaamste techniek die gebruikt wordt is Illumina *sequencing* (oorspronkelijk ontwikkeld door Solexa). Andere *second generation sequencing* platformen zijn 454 *sequencing* (Roche), SOLiD *sequencing* (Applied Biosystems – Life Technologies) en Ion Torrent (Life Technologies), waarbij enkel de laatste nog frequent wordt gebruikt. Deze methodes zijn gebaseerd op de aanmaak van een *library*, een amplificatiestap, en *sequencing-by-synthesis*. In deze masterproef wordt er vooral met Illumina-data gewerkt.

In de eerste stap van Illumina *sequencing* wordt dubbelstrengig DNA gefragmenteerd, en worden er specifieke adapters aan beide eindes geligeerd. Daarna wordt het DNA gedenatureerd met vorming van enkelstrengig DNA. De amplificatiestap is gebaseerd op *bridge*-PCR waarbij er een *slide* (*flowcell*) gebruikt wordt met *forward* en *reverse* primers complementair aan de geligeerde adapters. Vervolgens wordt er een PCR-reactie uitgevoerd waarbij elk DNA-fragment een brug vormt tussen een *forward* en *reverse* primer. Herhaaldelijke denaturatie en extensie zorgt ervoor dat DNA-fragmenten lokaal worden geamplificeerd met vorming van miljoenen geïsoleerde clusters. In de sequeneringsstap wordt gebruik gemaakt van fluorescent gelabelde terminator-nucleotiden, waarvan de fluorescente groep afgeknipt worden. Daarnaast zijn de nucleotiden reversibel gemodificeerd met een terminatorgroep aan hun 3'-OH einde zodat telkens slechts een enkele nucleotide kan ingebouwd worden per DNA-fragment (Figuur 10). Het aflezen van de fluorescentie maakt het mogelijk te detecteren welke base in elk fragment ingebouwd werd. Daarna worden de niet-gebonden nucleotiden weggewassen, en worden de fluorescente groepen en terminator-groepen verwijderd van de aanwezige terminatornucleotiden. Het proces herhaalt zich waarbij er afhankelijk van het toestel *reads* van 100-700bp gevormd worden (Illumina, 2016). Roche 454 *sequencing* is gebaseerd op een amplificatie via emulsie-PCR en een *sequencing-by-synthesis* via *pyrosequencing*. In vergelijking met Illumina-*sequencing* zijn de *reads* langer (tot 1kb per streng), maar zijn er duurdere reagentia en is er een relatief hoge *error rate* in homopolymeren. SOLiD *sequencing* is gebaseerd op di-base gelabelde probes en het gebruik van een kleurencode. Er worden *reads* gegenereerd met een lengte van slechts 25-50bp (Gheysen, et al., 2014).



Figuur 10: Principe van de Illumina reversible terminator technologie. Na binding met een gepaste reversible terminatornucleotide is het mogelijk om de fluorescente groep te verwijderen en het 3'-OH einde te regenereren (Metzker, 2010).

Voor de Illumina Genome Analyzer II/IIx is er een gemiddelde *error rate* van grootteorde 10^{-2}-10^{-3} (Kircher & Kelso, 2010). De *error rate* varieert van 0.3% in het begin van de *read* tot 3.8% op het einde van de *reads*. Verkeerde *basecalls* worden voornamelijk voorafgegaan door een G-nucleotide. De A>C transversie is het meest frequent terwijl de C>G transversie het minst frequent voorkomt (Dohm, et al., 2008). Nakamura et al. suggereert dat GGC-motieven vaak error-gevoelige regio's voorafgaan (Nakamura, et al., 2011). De indel *error rate* is zeer laag en heeft een voorkomen van zo'n $4*10^{-6}$ (Minoche, et al., 2011). De gemiddelde substitutie *error rate* van Roche 454 *Sequencing* is 0.5%, het voorkomen van indels is 0.38% (Loman, et al., 2012). Een groot deel van de errors zijn kleine indels door een foute lengte *calling* van een homopolymeer. Luo et al., vond een *error rate* van 25% voor homopolymeren van 7 nucleotiden en bijna 70% voor homopolymeren van 11 nucleotiden. A's en T's zorgen voor een hogere homopolymeer error dan C's en G's (Luo, et al., 2012) (Marinier, et al., 2015). De gemiddelde *error rate* bij SOLiD *sequencing* hangt af van de aanwezigheid van een referentiegenoom voor errorcorrectie (10^{-3} - 10^{-4} t.o.v. 10^{-2} - 10^{-3}) (Kircher & Kelso, 2010).

Recentere 3rd *generation sequencing* methoden (*single molecule sequencing*) vereisen niet langer een amplificatiestap en zorgen dusdanig niet voor PCR-fouten of een GC-*bias*. De belangrijkste toestellen zijn de Heliscope *sequencer* (Helicos), de PacBio *sequencer* (Pacific Biosciences) en de ION systemen (bv. GridION, MinION; Oxford Nanopore Technologies). De *reads* die gegeneerd worden zijn doorgaans minimum 5kb lang, hoewel ook fragmenten van meer dan 40kb worden gesequeneerd (Surekha & Rao, 2016). De grootste tekortkoming is het hoge percentage *sequencing errors*. In het kader van deze masterproef zijn deze methodes zeer interessant gezien de repetitieve aard van de telomeren, en in mindere mate de subtelomeren (Chaisson, et al., 2014; Gheysen, et al., 2014).

1.8. Bowtie 2

Na het verkrijgen van gesequeneerde fragmenten is het meestal nodig om te achterhalen van welke locus deze exact afkomstig zijn, deze stap wordt mapping of aligniering genoemd. Aangezien voor deze masterscriptie Bowtie 2 werd gebruikt wordt enkel deze tool hier besproken, maar veel recente mappers (bv. BWA) gaan op een gelijkaardige manier te werk. Bowtie 2 is een C/C++ tool die het genoom indexeert op een manier waarbij het gebruik van computergeheugen minimaal blijft. Dit gebeurt door gebruik te maken van een FM-index, die gebaseerd is op de Burrows-Wheeler transformatie waardoor data gecompresseerd en geïndexeerd kan worden (Ferragina & Manzini, 2005; Langmead, 2013). In tegenstelling tot Bowtie, is Bowtie 2 sneller en gevoeliger voor *reads* die langer zijn dan 50bp, en is er een mogelijkheid voor een *gapped* aligniering (toelaten indels), lokale aligniering (versus globaal, het volledige fragment) en *paired-end* aligniering (voor *paired-end sequencing* data, waarbij beide uiteinden van een DNA fragment worden gesequeneerd). Meerdere processoren kunnen gebruikt worden bij deze aligniering, en de *output* wordt gegeven in een *Sequence Alignment Map* (SAM) *format*.

Bij elke *read* worden er *seed substrings* genomen van zowel de *read* als zijn reverse complement die vervolgens gealigneerd worden met de referentiesequentie. Dit levert aligneringen op onder de vorm van Burrows-Wheeler *ranges*. Door middel van *dynamic programming* worden de *seeds* tot volledige aligneringen verlengd tot er voldoende *seeds* onderzocht zijn, en wordt de beste aligniering geselecteerd (Langmead & Salzberg, 2012).

1.9. Telomeerlengtebepaling met behulp van sequenering

In een studie uitgevoerd door Castle et al. (2010) werd er ontdekt dat uit het aantal TTAGGG of CCCTAA-herhalingen in *reads* een telomeerlengte kan geschat worden. Sindsdien werden verschillende tools ontwikkeld om deze mogelijkheid verder te verkennen. Telseq is een *open source* software pakket dat het aantal telomeermotieven telt, en dit vertaalt in een absolute gemiddelde telomeerlengte. *Reads* worden in deze methode telomerisch beschouwd indien ze *k* of meer TTAGGG *repeats* bevatten (met *k* standaard gelijk aan zeven). De methode werd gevalideerd op 260 leukocytenstalen van de TwinsUK cohort. Deze data bevatte zowel Illumina 100bp *paired-end whole genome sequence* (WGS) data als telomeerlengtemetingen met *Southern blot* mTRFs. Telseq werd ook toegepast op 96 stalen van het 1000 Genomes Project (zie 1.10: 1000 Genomes Project) dat zowel WGS als *exome* (alle proteïne-coderende regio's in het humaan genoom) *sequencing* data omvat. Hieruit blijkt dat de methode ook toepasbaar is op *exome sequencing* data (Ding, et al., 2014).

Net zoals Telseq is Computel een softwarepakket dat de telomeerlengte schat uitgaande van WGS data. Deze berekening is in essentie een *alignment*-gebaseerde methode via Bowtie 2 waarbij de *reads* worden gealigneerd aan een telomerische index van CCCTAA-*repeats*. De validatie werd uitgevoerd op basis van WGS data van tumor en gezond weefsel van vijf neuroblastoma en twee osteosarcoma patiënten. Voor deze stalen werd het verschil in telomeerlengte tussen gezond en tumorweefsel eerder al geschat via kwantitatieve *real-time* PCR. Daarnaast werd de performantie van Computel vergeleken met Telseq. Hieruit blijkt dat op verschillende vlakken Computel de betere methode is. Zo is er geen afhankelijkheid van een opgelegde *k-threshold* en wordt de berekende telomeerlengte minder beïnvloed door een andere readlengte dan 100 nucleotiden. De aligniering-gebaseerde methode van Computel is in tegenstelling tot Telseq minder gevoelig voor *sequencing errors*. Tenslotte kan Computel ook een onderscheid maken tussen de pure telomeren en de subtelomeren. Enkel de *reads* die gealigneerd zijn met het telomerische deel van de index dragen bij tot de

berekende telomeerlengte. Hierdoor wordt de *bias* van subtelomerische regio's verminderd (Nersisyan & Arakelyan, 2015).

1.10. 1000 Genomes Project

Het 1000 Genomes Project, opgestart in januari 2008, is een internationaal project dat als doel heeft geno- en haplotype informatie in kaart te brengen van alle humane DNA polymorfismen binnen verschillende humane populaties. Specifiek betekent dit dat voor elk van vijf verschillende belangrijke bevolkingsgroepen (Europa, Oost-Azië, Zuid-Azië, West-Afrika en Amerika) 95% van alle genomische detecteerbare varianten met een allelfrequentie van minstens 1% gekarakteriseerd moeten worden (The 1000 Genomes Project Consortium, 2010). Dit project werd onderverdeeld in een proefonderzoek gevolgd door drie verschillende fasen. Het proefonderzoek op zich werd nogmaals onderverdeeld in drie studies: een *low coverage* (2-4X) onderzoek (pilot 1), een *high coverage* (20-60X) onderzoek (pilot 2) en een exongerichte pilot (50X) (pilot 3). In de eerste fase werd er gefocust op een lage *coverage* WGS data en exoom data analyse van 1092 individuen. Fase 2 omvatte een grotere groep van 1700 individuen, en fase 3 bevatte 2535 individuen van 26 verschillende populaties (1000 Genomes, 2012).

In deze masterproef wordt er gebruik gemaakt van de data uit de eerste fase van het onderzoek. Omdat er gewerkt wordt met meerdere *sequencing centers* zijn er ook verschillende methodologiën toegepast. De belangrijkste sequencers waarvan gebruik werd gemaakt zijn de Illumina HiSeq 2000, Illumina GA (I, II, IIX), SOLiD (V3, V3+, V4) maar ook Roche LS454. Het mappen aan het referentiegenoom GRCh37 werd reeds uitgevoerd door het 1000 Genomes Project team met BWA (Illumina-data), SSAHA (Roche 454-data) en BFAST (SOLiD). Voor de genotypering werd er afhankelijk van het *sequencing center* gebruik gemaakt van verschillende algoritmes en pipelines die vaak gebruik maken van een *Bayesian likelihood* model om de meest waarschijnlijke genotypes en allelfrequenties te schatten. Na het annoteren gebeurde er een validatie met verschillende *sequencers* op 300 SNP loci van chromosoom 20.

1.11. Doelstellingen thesis

Het belang van telomeerbiologie is reeds veelvuldig aangetoond. Zo is de lengte van telomeren een biomerker van veroudering en is er een duidelijk (maar onopgehelderd) verband met o.a. cardiovasculaire ziektes. De lengte van telomeren wordt vaak gemeten via de TRF techniek, die gezien wordt als gouden standaard binnen dit domein. In deze masterscriptie wordt de hypothese van een foutieve en aspecifieke meting via TRF onderzocht. Variaties binnen de subtelomeren worden niet altijd geknipt met de restrictie-enzymen, wat ertoe kan leiden dat subtelomerische fragmenten mee worden opgepikt door TRF en gezien worden als functionele telomeren. Aaneensluitend hierbij is dat de aanwezigheid van deze mutaties binnen de telomeren de fragiele D-loop-T-loop vorming kunnen bemoeilijken. Bij het verlies van deze structuur worden de chromosoomeindes herkend als dubbelstrengige breuken met als gevolg dat er replicatieve senescentie van de cel optreedt.

Door gebruik te maken van de Phase 1 dataset van het 1000 Genomes Project worden de variaties binnen de functionele telomeren en subtelomeren via een bio-informatica benadering gekarakteriseerd. Er wordt een vergelijking uitgevoerd tussen de verschillende bevolkingsgroepen waarin de vaakst voorkomende hexameren beschreven en onderzocht worden.

2. Materiaal en methoden

2.1. Proefopzet

Een eerste luik van deze thesis begint met een *proof of concept*. Bij het testen van de hypothese wordt er gespecifieerd met welke dataset er gewerkt zal worden. Er wordt een vergelijking gemaakt tussen de telomeerlengteberekening met Computel en Telseq waarna er slechts met een enkele methode wordt verdergewerkt. Mogelijke determinanten van de gemeten telomeerlengtes in de dataset worden geanalyseerd.

In de tweede fase wordt op basis van de aligneringsdata van de telomerische fragmenten een globaal telomerisch profiel opgesteld. De *sequencing error rate* en indel *error rate* worden beschreven en het voorkomen van indels en SNPs wordt vergeleken. Vervolgens wordt er een beeld geschetst van de vaakst voorkomende SNPs en hexameren binnen de telomerische fragmenten.

In de derde fase worden de telomerische fragmenten verdeeld in drie verschillende fracties: “pure” telomeren, pseudotelomeren en hybride telomeren. In elke fractie zullen recurrente SNPs en indels beschreven worden in functie van de etnische afkomst van het individu.

2.2. Hardware en software

Bij het maken van deze masterscriptie werd een laptop (CPU: i5 2.5GHz, 8GB DDR3 RAM, OS: Windows 10) en een desktop (CPU: i5 3.0 GHz, 8GB DDR3 RAM, OS: Windows 10) gebruikt. De meeste computaties werden uitgevoerd op de *midas* server van BioBix. Deze Linux server gebruikt als OS Fedora21 en beschikt over 320GB RAM en 64 processoren. Voor *scripting* werd gebruik gemaakt van Python v2.7.10. Statistische bewerkingen werden gedaan met R v3.2.3. MobaXterm v8.2 werd gebruikt als SSH-*client* om op de *midas* server te werken. De scriptie zelf werd opgesteld met het softwarepakket Office 2010.

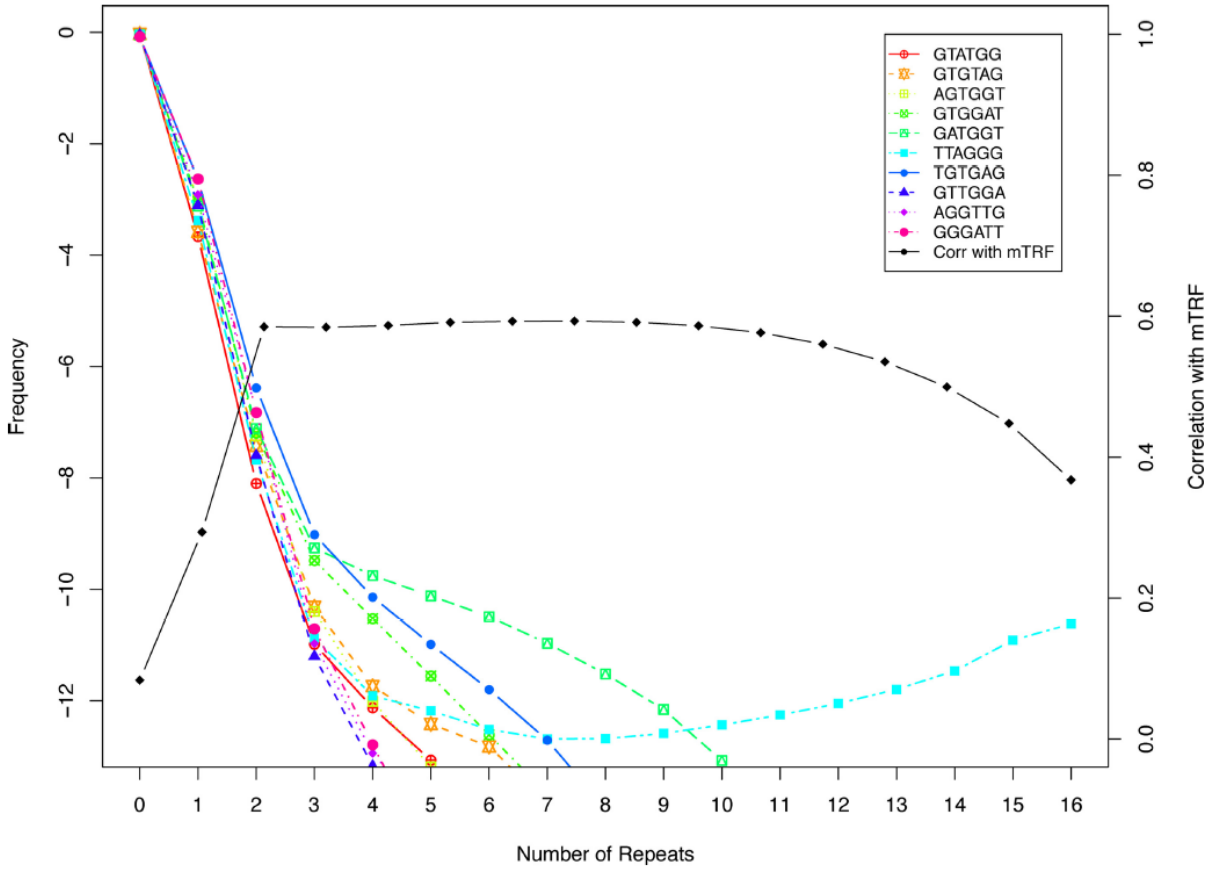
2.3. Telseq

Telseq (Ding, et al., 2014) is een softwarepakket dat de telomeerlengte berekent waarbij WGS data onder de vorm van *Binary Alignment Map* (BAM) bestanden als input worden ingegeven. Deze software is in C++ geïmplementeerd en gebruikt de bamtools *library* en *GNU Compiler Collection* (GCC), dat de programmeertaal vertaalt tot machinetaal. De telomeerlengte wordt bepaald door de formule:

$l = t_k * s * c_g$

In deze vergelijking stelt l de telomeerlengte (kbp) voor, t_k het aantal telomerische *reads* bij een *threshold k* (standaard 7), en is c_g een constante voor de genoomlengte gedeeld door het aantal telomerische eindes (46). De waarde k is het minimale aantal telomerische *repeats* die er aanwezig moeten zijn in een *read* zodat de *read* als telomerisch wordt beschouwd (Figuur 11). De fractie van *reads* met een GC-compositie tussen 48 en 52% wordt voorgesteld door s. Deze *range* wordt gebruikt omdat de telomerische sequenties $((TTAGGG)_n)$ een GC-waarde van 50% vertonen. Eerdere studies hebben aangetoond dat amplificatie tijdens PCR afhankelijk is van de GC-compositie (Dohm, et al., 2008). Bij de berekening van de telomeerlengte wordt het BAM-bestand ingeladen, en wordt in elke *read* het aantal TTAGGG of CCCTAA *repeats* geteld. Elke *read* wordt ook beoordeeld op zijn GC-percentage waarbij er verschillende GC-“*bins*” worden gemaakt waarin elke *read* terecht

komt. Het aantal *reads* binnen GC4 en GC5 (48-52% GC-compositie) wordt gebruikt voor de berekening van de uiteindelijke telomeerlengte (Ding, et al., 2014).



Figuur 11:Identificatie van telomerische reads. Voor 260 Twins UK stalen werd de aanwezigheid van alle permutaties van TTAGGG-repeats nagegaan. De frequenties van reads met het aantal hexameren zijn te zien op een logaritmische schaal. In zwart is de correlatie te zien van Telseq met mTRF als functie van threshold k (Ding, et al., 2014).

2.4. Computel

De *open source* tool Computel (Nersisyan & Arakelyan, 2015) is geschreven in R, maakt gebruik van drie externe programma's (Bowtie2-2.1.0, Samtools 0.1.19 en Picard tools 1.108). Zoals hoger vermeld (zie 1.8: Bowtie2) wordt Bowtie2 gebruikt voor de alignering van de *reads* t.o.v. het referentiegenoom. Samtools wordt aangewend om handelingen met SAM-bestanden uit te voeren en een *coverage* te berekenen via de *depth* functie. De Picard software is in staat een SAM-bestand naar een FASTQ-bestand om te zetten. Computel is onderverdeeld in vier verschillende scripts:

- **computel.R**: standaard gebruiksvriendelijk overzicht waar de meeste parameters waarmee Computel werkt aangepast kunnen worden. De *paths* naar de tools die gebruikt worden en de andere scripts dienen hier ingegeven te worden. Naast de standaard computel.R is er ook een computel.cmd.R script voorzien dat identiek is aan computel.R maar vanuit de *command* lijn opgeroepen kan worden.
- **validate.options.R**: bij het opstarten van computel.R wordt alle data doorgesluisd naar validate.options.R dat alle opties valideert, en bij ongeldigheid of afwezigheid

van nodige informatie een standaard waarde meegeeft. Alles wordt daarna doorgestuurd naar pipeline.R.

- **functions.R**: een opsomming van alle functies die Computel gebruikt zoals het bouwen van een telomerische index, het berekenen van de *coverage* en de berekening van de telomeerlengte.
- **pipeline.R**: dit script krijgt alle startwaardes van validate.options.R en overloopt stapsgewijs de verschillende stappen. Alle benodigde functies worden opgeroepen vanuit functions.R. Uiteindelijk wordt o.a. de *base coverage* en telomeerlengte opgeslagen in een .xls bestand.

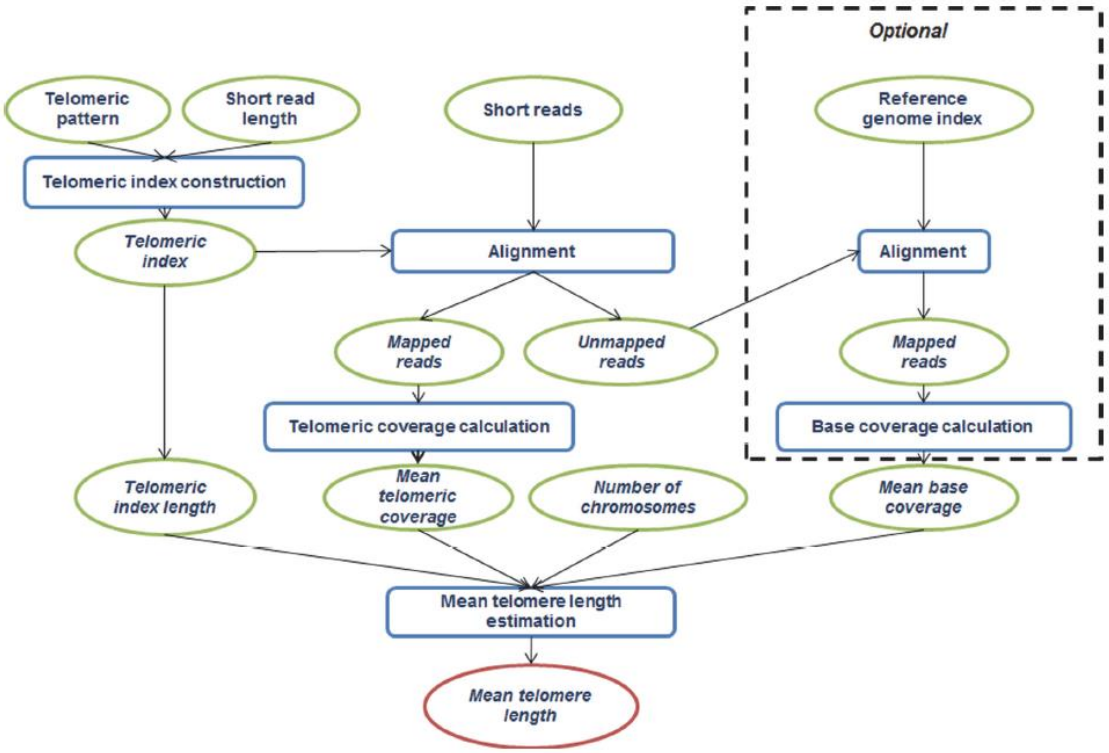
Een overzicht van de *workflow* is te zien op Figuur 12. Computel vereist een FASTQ-bestand om mee te werken. Sinds de gebruikte Illlumina sequentiedata beschikbaar is als een BAM-bestand dient dit nog omgezet te worden. Vanuit een opgegeven *read* lengte wordt een pure telomerische sequentie opgesteld met *bowtie2-build*: een aaneenschakeling van CCCTAA-repeats met het totale aantal herhalingen = [(*read* lengte + patroonlengte – 1)/patroonlengte]. Na deze telomeersequentie worden een aantal N-nucleotiden aan het 3'-einde van deze index gehecht met een totale lengte van (*read* lengte – min.seed) herhalingen. Het toevoegen van deze N-nucleotiden heeft als bedoeling *reads* te detecteren die deels telomeerrepeats bevatten. Deze min.seed waarde (standaard als 12 geconfigureerd) verhogen zal zorgen voor een hogere specificiteit maar ook een lagere gevoeligheid in het detecteren van telomerische herhalingen.

De telomerische index zal gealigneerd worden aan elke *read* door gebruik te maken van *bowtie2-align*. De parameters die in Computel als standaard geconfigureerd staan zijn een –*end-to-end* aligning op –*very-sensitive* mode met –N 1 (dat *mismatches* toelaat), en een –L die varieert afhankelijk van de *read* lengte. De –L bepaalt de lengte van *seed substrings* die worden gealigneerd tijdens de *multiseed* alignering en wordt berekend door (*read* lengte / 3) met als maximum- en minimumwaarde respectievelijk 22 en 6. De *output* van deze alignering wordt opgeslaan in een SAM-bestand.

Voor de berekening van een referentiegenoom *coverage* kunnen ongemapte reads afkomstig van het SAM-bestand gealigneerd worden met een opgegeven referentiegenoom via Bowtie2. Daarna kan de *output* gesorteerd worden, en kan er een *base coverage* berekend worden met *samtools depth*. Door het tijdsintensieve aspect van deze bewerking kan deze *base coverage* ook geschat worden door [(#*reads* * *read* lengte) / genoomlengte]. Deze parameter wordt dan meegegeven als startwaarde in computel.R.

Voor de berekening van een telomerische *coverage* wordt het SAM-bestand gesorteerd en volgt er een berekening met *samtools depth*. De uiteindelijke relatieve *coverage* die gebruikt wordt voor de berekening van de telomerische index is een verhouding van de telomerische *coverage* met de *base coverage*. De uiteindelijke telomeerlengte wordt dan als volgt berekend:

$$\text{Gemiddelde telomeerlengte} = \frac{\text{gemiddelde relatieve coverage} * (\text{read lengte} + \text{patroonlengte} - 1)}{2 * \text{aantal chromosomen}}$$



Figuur 12: schematische voorstelling van het Computel-algoritme voor de berekening van de telomeerlengte. WGS data onder de vorm van reads worden gealigneerd met een opgestelde pure telomerische index. Hiervan wordt er een telomerische coverage berekend waarmee de telomeerlengte wordt bepaald (Nersisyan & Arakelyan, 2015).

2.5. BAM/SAM-bestandsformaat

In het Phase 1 onderdeel van het 1000 Genomes Project is de sequentiedata beschikbaar onder de vorm van een BAM-bestand. Dit is een binaire, geïndexeerde en gecompresseerde versie van het SAM-bestand. Beide bestanden bevatten een optionele *header* (aangegeven door "@") die informatie meegeeft over de referentiesequentie, *read groups*, sequentie mapping software en instellingen. De *header* wordt gevolgd door een aligneringssectie waarin o.a. de gealigneerde sequenties te vinden zijn, een *base quality* (QUAL-score), *mapping quality* (MAPQ-score) en positie van alignering, indels en substituties. De voornaamste parameters die in deze masterproef worden gebruikt worden hier kort besproken (The SAM/BAM Format Specification Working Group, 2015).

- De CIGAR-code is een sequentie die meegeeft welke basen van de *read* aligneren met de referentie-sequentie (M: match/mismatch) en waar er zich inserties (I) of deleties (D) bevinden bv. "7M1D29M1I36M1I6M1I20M".
- De MD-code bevat informatie over de posities van *mismatches* en deleties, maar negeert inserties bv. "7^C38T0A48A2".
- De NM-code houdt bij hoeveel verschillen er zijn van de *reads* met de referentie-sequentie. Het aantal inserties, deleties en *mismatches* wordt bijgehouden bv. "NM:i:7".

Er werd vooral gebruik gemaakt van SAMtools in linux om met deze bestanden om te gaan.

2.6. Sample selectie

De Phase 1 data van het 1000Genomes Project bestaat uit een grote variëteit aan o.a. *coverages*, *read* lengtes en *sequencing* apparaten. Om die reden werd er een eerste selectie gedaan volgens de *read* lengtes sinds Telseq gevalideerd werd op WGS data bestaande uit 100bp *read* lengtes. Er werd gebruik gemaakt van een tekstbestand afkomstig van het 1000 Genomes Project (phase1.alignment.index.bas) waaruit voor elk individu de *read* lengte kan worden berekend. Er werd een duidelijk verschil in kwaliteit gezien tussen de 100/101bp *reads* (n=161) en de 108bp *reads* (n=145). Hierdoor wordt er een opsplitsing gemaakt in deze datasets. Hiervoor werden de scripts in Bijlage 5A en Bijlage 5B gebruikt.

Het voorkomen van verschillende *read* lengtes binnen eenzelfde invidu geeft een gemiddelde *read* lengte dat problemen geeft bij het opstellen van een telomerische index. Omdat deze index dan niet op een optimale manier kan aligneren met elke *read* wordt deze data bij voorkeur niet gebruikt. Bij het opstellen van een histogram worden dus alle gemiddelde *read* lengtes die slechts een enkele maal voorkomen verwijderd.

2.7. Berekening van de telomeerlengte

Door de manier waarop Telseq en Computel opgesteld zijn was het nodig om dit proces van telomeerlengteberekening te automatiseren. Het Computel-script werd om deze reden in een while-loop geplaatst dat als enige *input* de URL van de BAM-bestanden van de gemapte en ongemapte *reads* vereiste. Het script werd verder aangepast zodat ook Telseq toegepast werd op de *sequencing* data (Bijlage 4). Algemeen gesteld was dit een opeenvolging van volgende stappen die herhaald werd voor elk individu:

- Downloaden van gemapte *reads* voor specifiek individu.
- Downloaden van ongemapte *reads* voor specifiek individu.
- Fuseren van beide BAM-bestanden naar een enkel BAM-bestand via *samtools merge.*
- Telseq toepassen op het BAM-bestand.
- Omzetting van de .bam file naar een FASTQ-bestand via *samtools bam2fq.*
- Berekening van de *read* lengte en *base coverage*.
- Computel toepassen op het FASTQ-bestand.
- Wegschrijven van de aligneringsdata.
- Na het wegschrijven van de *output*, het verwijderen van de BAM-en FASTQ-bestanden.

Het wegschrijven van de aligneringsdata is noodzakelijk voor het analyseren van de gecapteerde telomerische reads. Er werd gekozen om dit naar een tekstbestand om te zetten sinds de handelingen hiermee eenvoudiger zijn dan met een SAM-bestand. Door een beperkte opslagcapaciteit is het nodig om de BAM- en FASTQ bestanden vervolgens te verwijderen.

De URL's werden verkregen door een python script dat via het tekstbestand phase1.alignment.index.bas een URL genereert naar het BAM-bestand op de EBI FTP server. Dit script vereist als *input* enkel het individunummer (bv. HG00187, NA19452) waarvoor de URL gewenst is (Bijlage 5C).

2.8. Telomerisch profiel

Het opstellen van een telomerische distributie maakt het mogelijk om verschillen in het aantal SNPs tussen de 100/101-data en 108-data te vergelijken. Voor elk individu worden alle SNPs binnen de MD-code opgeteld met uitzondering van nucleotiden die aligneren met het N-terminale deel van de telomerische index. Het aantal gedetecteerde substituties worden ook telkens gedeeld door het totale aantal *reads* zodat een frequentie bekomen wordt. Dit wordt gedaan omdat aligneringsbestanden onderling verschillen in grootte of aantal gecapteerde *reads*. Door het uitdrukken in een frequentie en nadien te delen door het totaal aantal individuen wordt er dus een niet-gewogen gemiddelde verkregen van de totale telomerische distributie (Bijlage 5P).

2.9. Poisson distributie

Bij het schatten van de *sequencing error rate* werd er gebruik gemaakt van een Poisson distributie. Er wordt uitgegaan dat deze *sequencing error* gebeurtenissen willekeurig en onafhankelijk van elkaar gebeuren op een constante wijze. De Poisson probabiliteitsdistributie steunt op het gebruik van een Poisson *random variable* λ. Hierin wordt de probabiliteit om x aantal *events* (hier *sequencing errors*) te beschouwen berekend als:

$$P(X = x) = \frac{\lambda^x * e^{-\lambda}}{x!}$$

2.10. SNP – indel ratio

Het beschouwen van de SNP-indel ratio's wordt opgesplitst tussen de 100/101-data en de 108-data. Voor elk individu wordt bij elke *read* het aantal indels en mutaties bijgehouden. Bij het bereiken van het einde van het bestand wordt de ratio berekend en opgeslagen in een een lijst. Tenslotte worden de SNP-indel ratio's van alle 100/101-data invididuen en 108-data individuen in een histogram gevisualiseerd (Bijlage 5H, Bijlage 5I en Bijlage 5J).

2.11. Beschrijven van SNPs

Voor elke *read* in een aligneringsbestand wordt de sequentie, MD-code en CIGAR-code geïsoleerd. Omdat de MD-code geen rekening houdt met aanwezige inserties, wordt deze MD-code gemodificeerd. Wanneer er een insertie in de CIGAR-code is, zal het juiste element binnen de MD-code worden aangepast met het aantal geïnsereerde nucleotiden. Het aanpassen van deze code is essentieel om een juiste relatie te hebben met de sequentie. Vervolgens wordt de sequentie gescand op aanwezige nucleotidesubstituties binnen de MD-code. De SNP wordt weggeschreven onder de vorm van [base in de $(CCCTAA)_n$-polymeer] > [base in de telomerische *read*] (Bijlage 5K, Bijlage 5L en Bijlage 5M).

2.12. Beschrijven van hexameren

Het beschrijven van de voorkomende hexameren gaat analoog te werk als (2.11: Beschrijven van SNPs). De sequentie, MD-code en CIGAR-code worden in elke *read* gecapteerd. De MD-code wordt op eenzelfde manier gecorrigeerd voor de aanwezigheid van inserties. Elke plaats waar er een nucleotidesubstitutie wordt gedetecteerd in de sequentie wordt als een alternatief hexameer beschouwd. De omringende basen worden geïsoleerd en weggeschreven (bv. CCC**C**AA waarin de 4^e C de nucleotidesubstitutie voorstelt). Er wordt

een uitzondering gemaakt wanneer de mutatie helemaal voorin of achterin de *read* valt. In dat geval worden respectievelijk de eerste of laatste zes nucleotiden van de *read* gecapteerd (Bijlage 5N).

2.13. Opsplitsing van telomerische fracties

In het fase 3-onderdeel wordt de aligneringsdata opgesplitst in drie verschillende telomerische fracties. Dit gebeurt via verschillende scripts die rekening houden met de aanwezige mutaties binnenin deze *reads*. De “pure” telomerische fractie omvat alle telomerische *reads* die maximaal drie SNPs bevatten. Deze toegelaten SNPs zijn deel van een arbitrair gekozen *sequencing error rate* van 1.5% die hoog is, maar rekening houdt met de beginperiode van *second generation* methodes. Inserties en deleties zijn niet toegestaan, net zoals er geen twee mutaties naast elkaar worden geaccepteerd. De informatie over SNPs en indels wordt gehaald uit respectievelijk de MD-code en CIGAR-code (Bijlage 5D). De pseudotelomerische fractie omvat alle *reads* die meer dan drie SNPs hebben of indels bevatten. In deze fractie zijn ook geen *reads* toegestaan die twee nucleotidesubstituties naast elkaar hebben (Bijlage 5E). De hybridetelomerische fractie laat enkel *reads* toe die twee naast elkaar liggende nucleotidesubstituties hebben (Bijlage 5F). Het cumulatieve aantal *reads* van de drie afzonderlijke telomerische fracties is gelijk aan het totaal aantal *reads* in het originele aligneringsbestand.

Algemeen gesteld wordt er vertrokken vanuit de gegenereerde aligneringsbestanden van Computel. Er wordt gebruik gemaakt van een script in Python en een script in R. Het Python script is een lus dat voor elk aligneringsbestand volgende stappen uitvoert:

- *Read*-lengte isoleren vanuit het aligneringsbestand.
- Een LN-waarde berekenen voor in de SAM-header vanuit de *read*-lengte.
- Een *base coverage* berekenen aan de hand van het totale aantal *reads* waarmee het genoom van het individu gesequeneerd is. Dit is terug te vinden in 20101123.sequence.index.
- Een geschikte *header* plaatsen, noodzakelijk voor het SAM-bestandsformaat.
- Alle *reads* van het aligneringsbestand scannen en de *reads* selecteren die voldoen aan de gewilde telomerische fractie: “pure” telomeren, pseudotelomeren of hybride telomeren.
- Omvormen van een tekstbestand naar een volwaardig SAM-bestand. In de naamgeving wordt het individu weergegeven, de *read* lengte en de *base coverage*. Het tekstbestand wordt ook opgeslagen voor verder onderzoek op de telomerische fracties.

Het R-script is een opvolging van het Python script en is ook gebaseerd op een lus waarbij voor elk aligneringsbestand er een telomeerlengte wordt berekend (Bijlage 5G):

- Voor elk individu worden de meegegeven argumenten binnen de naamgeving van het SAM -bestand geïsoleerd: individu, *read* lengte en *base coverage*.
- Omvormen van het SAM-bestand naar een BAM-bestand met *samtools view*.
- Het BAM-bestand sorteren met *samtools sort*.
- De *depth* berekenen met *samtools depth*.
- Het berekenen van de telomeerlengte (bp) en de *output* in de linux *terminal* weergeven.

2.14. Bepalen van biologische relevantie

Bij de bepaling van biologisch relevante nucleotidesubstituties (versus *sequencing errors*) werd er onderzocht met welke frequentie nucleotidesubstituties voorkwamen op elke positite in het telomerisch hexameer CCCTAA. Het capteren van hexameren in telomerische *reads* hangt af van waar de nucleotidesubstitutie in de *read* gelegen is. Indien dit in het begin van de *read* was, werd er gekeken naar de nucleotiden na het gemuteerd nucleotide om de positie binnen het hexameer te bepalen. Indien dit op het eind van de *read* was, werd er gekeken naar de nucleotiden voor de nucleotidesubstitutie. Indien de nucleotidesubstitutie meer middenin de *read* lag, was dit een combinatie van deze twee. Door de kwaliteit van de *reads* worden er ook mutaties naar een onbekend nucleotide N waargenomen. Deze worden op het eind van het script hieruit verwijderd. Er komen soms kleine frequenties (grootteorde 0,01%) voor aan specifieke nucleotidesubsitituties waardoor het script de juiste positie niet kan bepalen. Sinds dit slechts in zeer kleine mate voorkomt, wordt ook dit verwijderd (Bijlage 5Q).

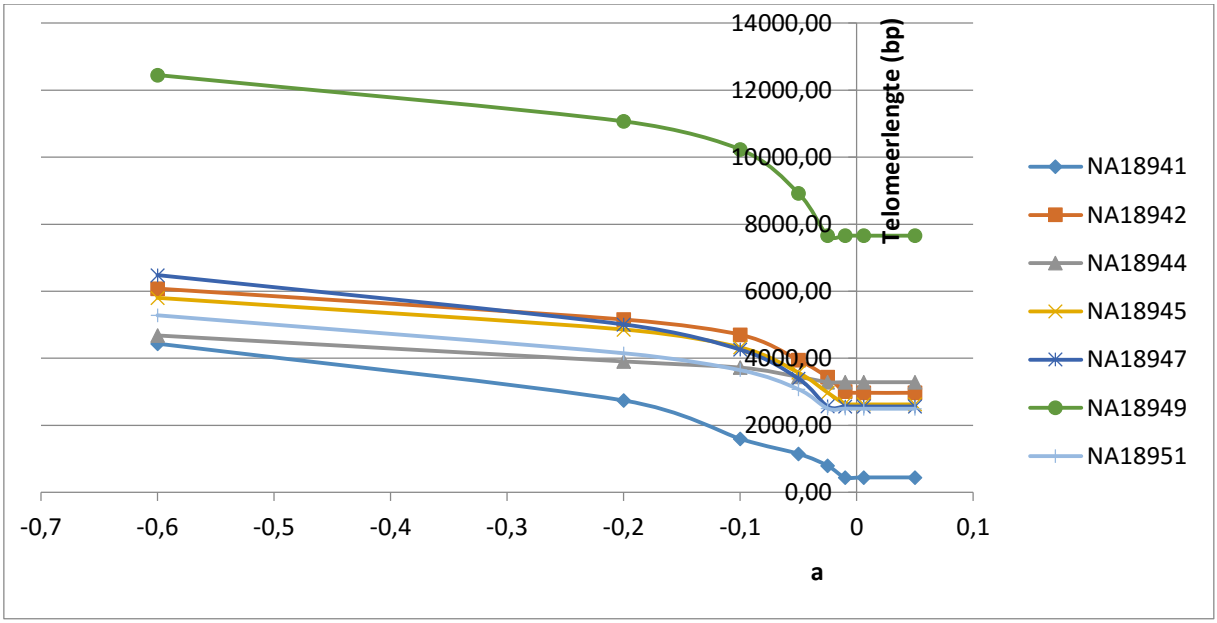
Er moet opgemerkt worden dat er verschillende kleinere R scripts doorheen de thesis zijn gebruikt geweest. Deze zijn te vinden in Bijlage 5R.

3. Resultaten

3.1. Fase 1

3.1.1. Proof of concept

De hypothese die in deze masterscriptie wordt getest is dat TRF-methodes, de gouden standaard van telomeerlengtemetingen, niet accuraat zijn om de pure telomeren te meten. Dit zijn telomeren die in staat zijn om de D-loop-T-loop structuur te vormen. Op Figuur 13 werden zeven *ad random* gekozen individuen onafhankelijk van *read* lengte maar allen gesequeneerd in het Wellcome Trust Sanger Institute, uit het 1000 Genomes Project gekozen. Van deze individuen werden de telomeerlengtes telkens gemeten bij een stijgende stringentie van alignering aan de hand van een *a*-parameter. Bij *a* = -0.6 worden de telomeerlengtes van de individuen geschat bij standaardinstellingen. Dit zijn ook de lengtes die gevalideerd werden op Telseq dat op zijn beurt gevalideerd was op TRF-metingen. Wanneer de stringentie (*a*) verhoogd wordt en er minder mutaties toegelaten worden in de *reads*, daalt de telomeerlengte tot er een minimum bereikt wordt die de “pure” mutatievrije telomeerlengte voorstelt. Er kan dus gesteld worden dat TRF methodes naar alle waarschijnlijkheid niet specifiek de telomeerlengte meten maar ook vatbaar zijn voor subtelomerische regio’s, zoals vroeger reeds gerapporteerd (zie 1.5.1: TRF). Belangrijk is hierbij op te merken dat een hogere stringentie niet bij elk individu een gelijkaardig effect heeft, zo heeft individu NA18944 zo goed als de kortste telomeerlengte bij a = -0,6 maar ook de tweede hoogste telomeerlengte bij a = 0. Dit suggereert dat interindividuele verschillen in de lengte van subtelomeren ook biologisch relevant kunnen zijn.

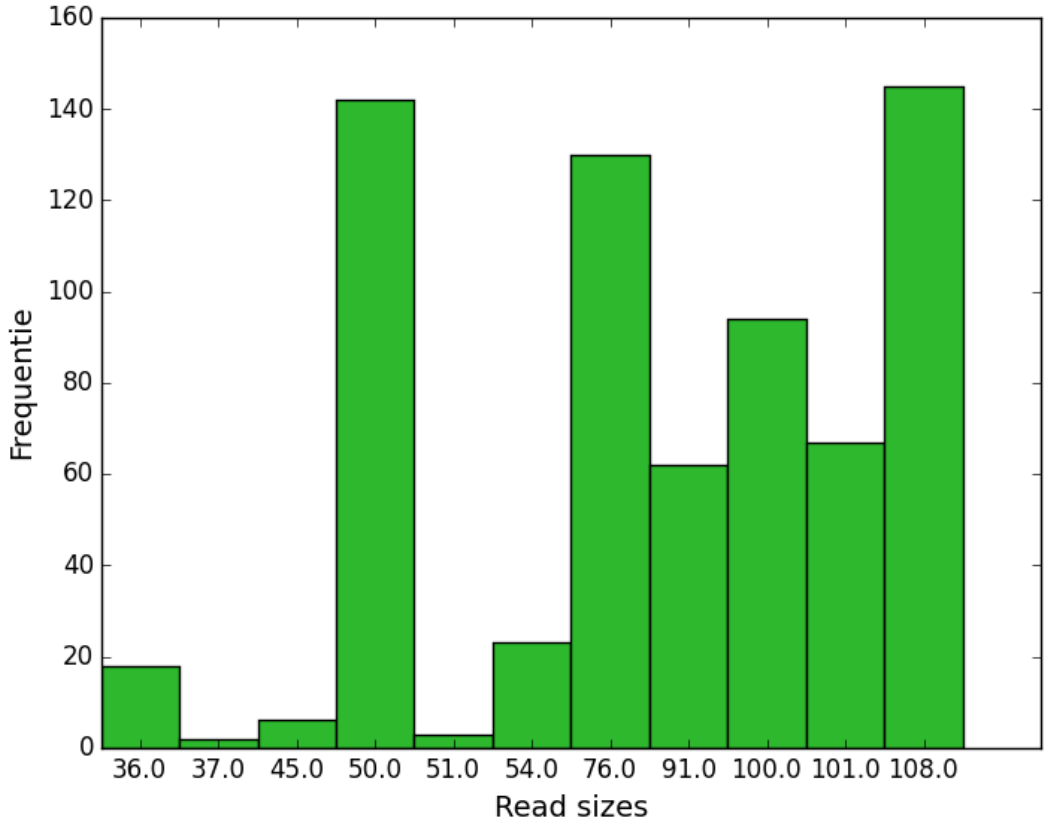


*Figuur 13: Proof of concept waarbij er een groot verschil is tussen de standaard gemeten telomeerlengtes bij TRF (a=-0,6) en de mutatievrije "pure" telomeren bij a=0. De parameter die gevarieerd wordt is “a” en bepaalt de stringentie waarmee Bowtie2 de pure telomerische index aligneert met elke read volgens de formule score-min = -0.6 + a*x waarbij x de read lengte voorstelt.*

3.1.2. Keuze van de sample set

Figuur 14 is een exploratieve weergave van de *read* lengtes van alle WGS data binnen de eerste fase van het 1000 Genomes Project. De sequeneringsdata voor *read* lengtes van

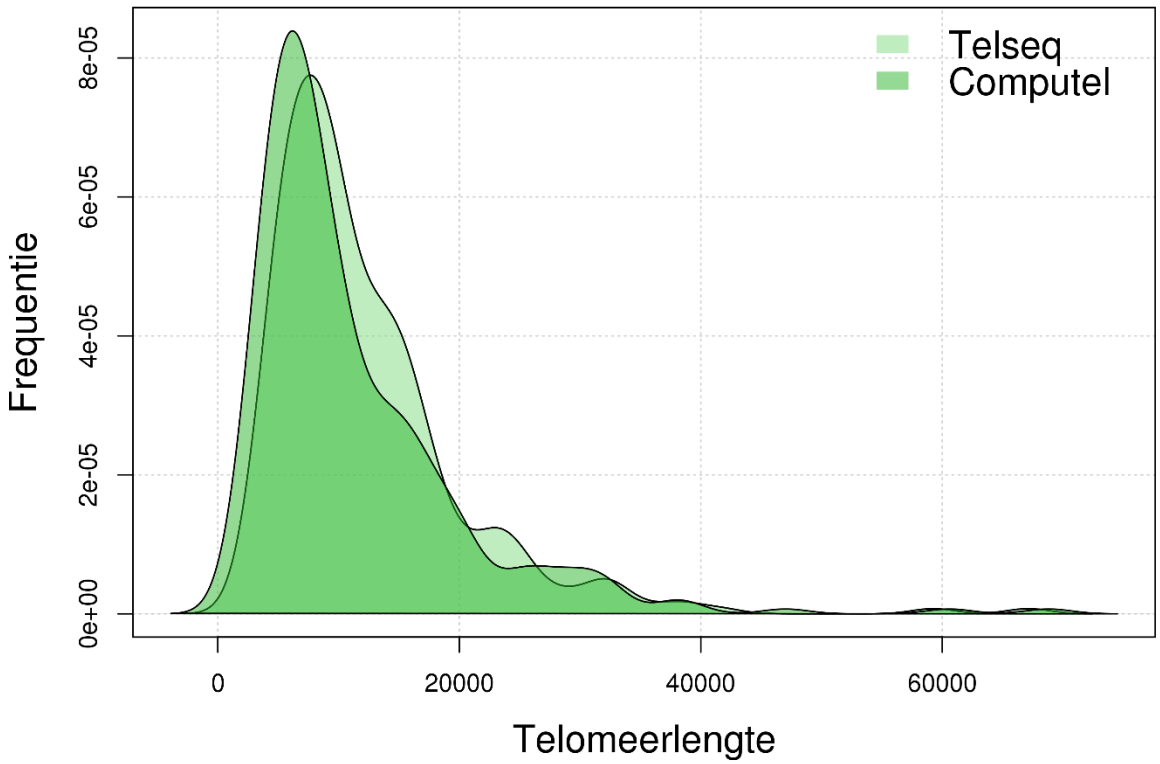
100bp, 101bp en 108bp omvatten samen 306 individuen en zullen verder gebruikt worden bij de berekening van telomeerlengte. De reden hiervoor is dat Computel en Telseq beiden geoptimaliseerd zijn op sequeneringsdata rond deze *read* lengtes. Hiernaast zijn er ook andere *read* lengtes binnen deze fase. Deze data is echter vooral afkomstig van Roche 454 of Illumina-*sequencing* data met verschillende *read* lengtes, ook binnen hetzelfde individu. De verklaring hiervoor is dat er bij de kwaliteitscontrole vaak enkele terminale nucleotiden worden verwijderd van de *read* wegens een lagere kwaliteit. *Reads* van 36, 50 en 54bp zijn afkomstig van SOLiD-*sequencing*. *Reads* van 76 en 91bp zijn eveneens Illumina-*sequencing* data.



Figuur 14: Een overzicht van de verschillende read lengtes binnen de eerste fase van het 1000 Genomes Project. In deze masterscriptie wordt er enkel gewerkt met de lengtes 100bp, 101bp en 108bp. Dit omvat 306 individuen.

3.1.3. Vergelijking Computel en Telseq

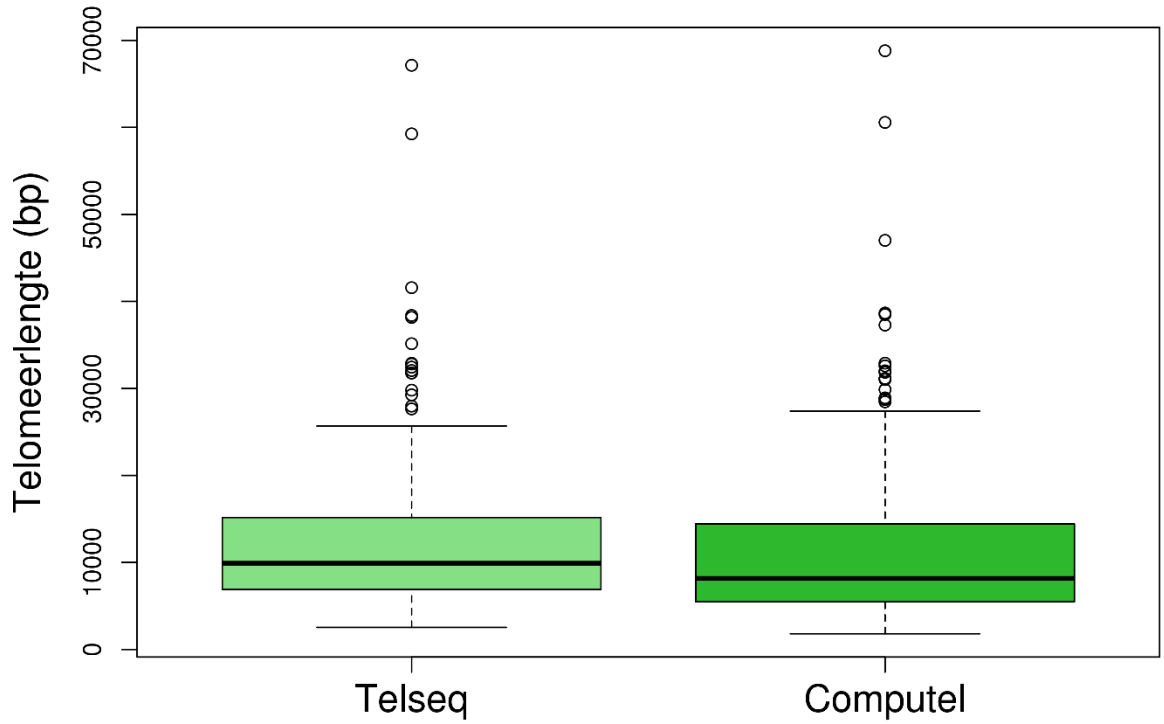
Er wordt voor zowel Telseq als Computel van alle 306 individuen een telomeerlengte berekend die te zien is op Figuur 15 onder de vorm van twee dichtheidscurves. De dichtheidscurve van Telseq is in grote mate identiek aan de dichtheidscurve van Computel, maar heeft een lagere piek en een kleine horizontale verschuiving naar rechts. Deze bevinding, ook geïdentificeerd door de Computel auteurs, kan worden verklaard door het feit dat Telseq ook subtelomerische gebieden meerekent, en zo de telomeerlengte overschat met 2,5-4kb (Nersisyan & Arakelyan, 2015). De mediaan van telomeerlengte berekend door Telseq bedraagt 9908bp terwijl de mediaan bij Computel 8164bp bedraagt (Figuur 16) wat dus op een lager verschil wijst dan voorheen aangegeven.



Figuur 15: Dichtheidscurves van de berekende telomeerlengtes voor Computel en Telseq. Een kleine horizontale verschuiving van de Telseqdata over de horizontale as is waarschijnlijk een gevolg van een hogere gevoeligheid voor subtelomerische gebieden (Nersisyan & Arakelyan, 2015).

Een gepaarde T-test gaf aan dat de gemeten telomeerlengtes via Telseq ($\bar{x}_{Telseq}$ = 12261bp) en Computel ($\bar{x}_{Computel}$ = 11202bp) wel significant verschillen van elkaar ($p < 2.2*10^{-16}$).

Voor beide methodes werd een boxplot opgesteld (Figuur 16). Voor zowel Telseq als Computel zijn er een gelijkaardig aantal uitschieters. Sinds Computel stringenter de telomeerlengte schat (waar er hierbij vanuitgegaan wordt dat deze nog te weinig stringent is), de tool in R geprogrammeerd is en de aligneringsdata eenvoudiger te isoleren is dan bij Telseq, wordt er dan ook voor gekozen om in de verdere masterscriptie enkel met Computel verder te werken.



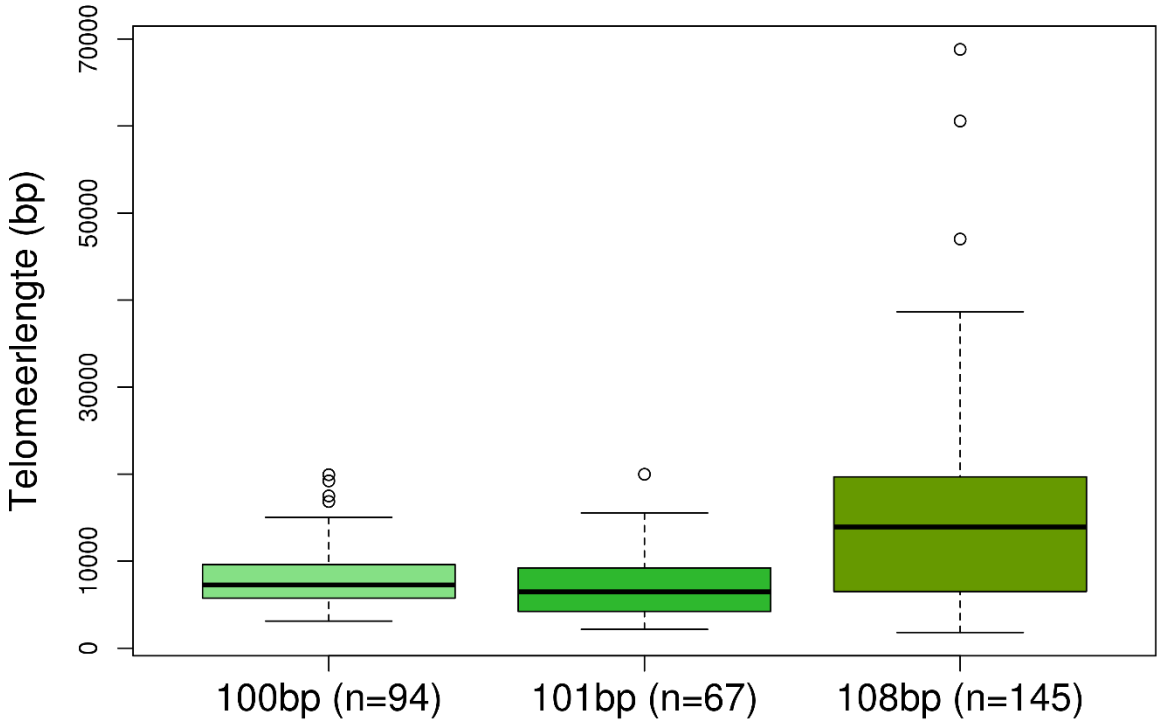
Figuur 16: Een boxplot waarin Computel en Telseq worden vergeleken, waarbij telomeerlengtes bij Telseq iets langer worden geschat dan bij Computel.

3.1.4. Vergelijking tussen determinanten

Er zijn verschillende technische en biologische factoren die een invloed kunnen hebben op de berekende telomeerlengtes, en deze worden dan ook eerst geëvalueerd. Het gaat hier over de *read* lengte, het *sequencing center*, en het *sequencing* apparaat, maar ook populatie en het geslacht. Leeftijden en andere fenotypische data werd omwille van privacy redenen voor de 1000 Genomes populatie niet bekend gemaakt.

3.1.4.1. Read lengte

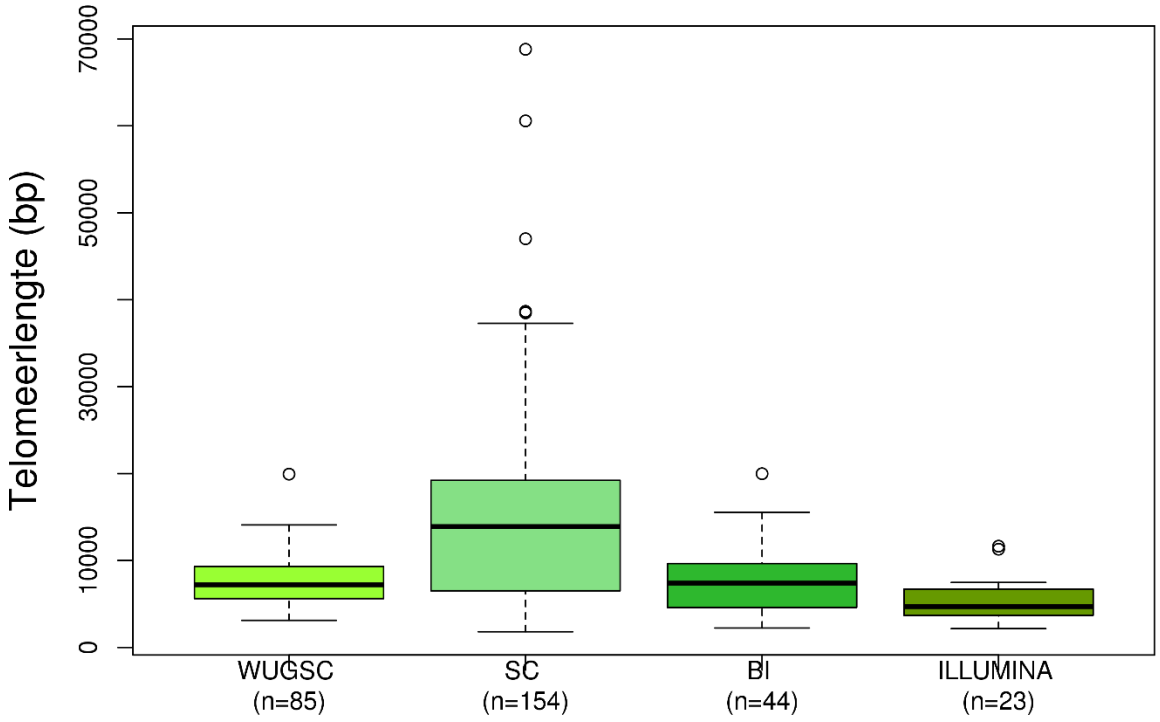
Op Figuur 17 worden de telomeerlengtes weergegeven voor de *sequencing* data in functie van de *read* lengtes waaruit deze data bestaat. Er is duidelijk te zien dat er een hogere telomeerlengte gemeten wordt bij data die bestaat uit *reads* van 108bp (100bp: 8017, 101bp: 6973bp, 108bp: 15221bp). De variabiliteit binnen deze data is ook erg verschillend van de 100/101-data (100bp: 3406bp, 101bp: 3551bp, 108bp: 10908bp). Het aantal uitschieters bij de *read* lengtes is hier echter niet opvallend verschillend. Een *one-way* ANOVA binnen de read-lengtes ($p=3.98*10^{-15}$) geeft aan dat er een significant verschil is tussen de groepsgemiddeldes bij een $\alpha = 0.05$. Een *pairwise t-test* laat toe om te bepalen welke *read* lengte voor dit significant verschil zorgt. Na een Bonferroni-correctie bevestigt de p-waarde tussen de *read* lengtes 100bp en 108bp ($p=3.3*10^{-10}$) en 101bp en 108bp ($p=1.4*10^{-10}$) dat de telomeerlengte voor 108bp *read* lengte data significant verschilt van zowel 100bp als 101bp data, terwijl deze laatste onderling niet significant verschillend waren ($p>0.99$).



Figuur 17: Vergelijking van de telomeerlengtes over de verschillende read lengtes. Waar de 100-en 101-data een gelijkaardige boxplot vertonen, lijkt de 108-data erg verschillend met een hogere mediaan en variabiliteit.

3.1.4.2. Sequencing center

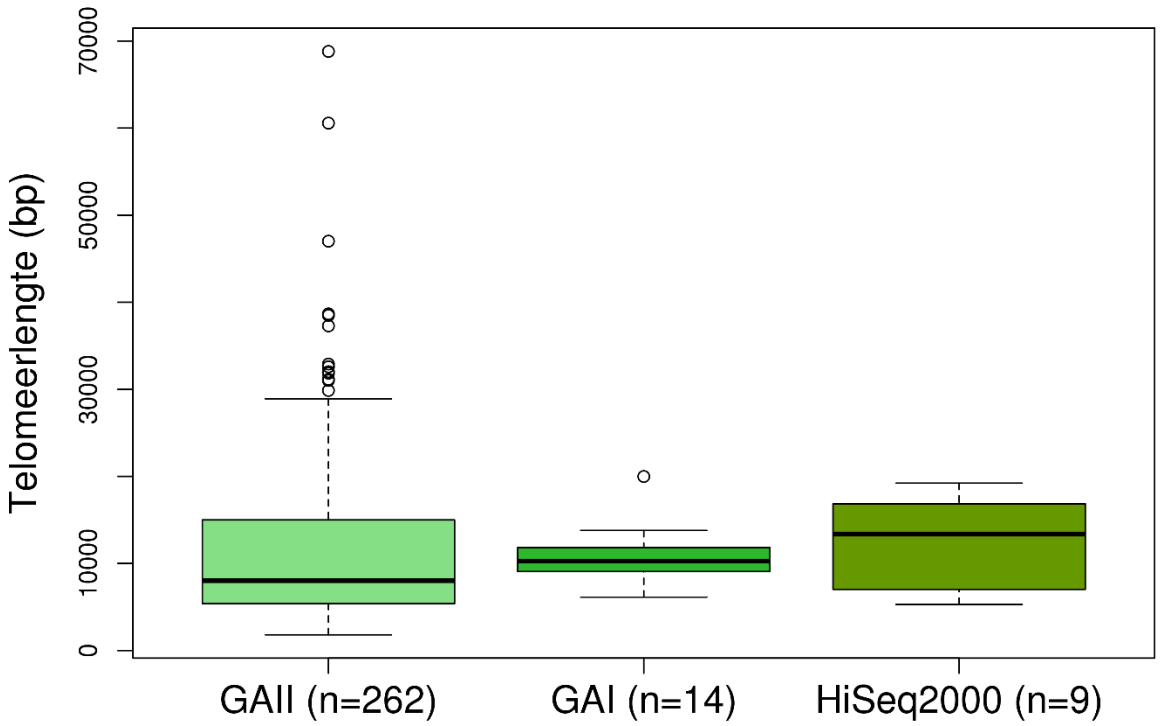
De data die gebruikt werd van de 306 individuen is afkomstig van vier verschillende centra: het Washington University Genome Sequencing Center (WUGSC), het Wellcome Trust Sanger Institute (SC), het Broad Institute (BI) en Illumina. Een vergelijking werd gemaakt op Figuur 18**.** De medianen van WUGSC (7199bp) en BI (7404bp) zijn gelijkaardig. WUGSC bevat echter een lagere standaarddeviatie (2788bp) in tegenstelling tot BI (3761bp). De mediaan van Illumina is het laagst van alle *sequencing centers* (4694bp) met een standaarddeviatie van 2487bp. SC meet hogere telomeerlengtes dan de overige *sequencing centers* (mediaan 13908bp, s = 10670bp). Dit verschil is gelijkaardig aan dat in Figuur 17, wat vermoedelijk gerelateerd is aan het feit dat SC alle 108bp data genereert. Een *one-way* ANOVA geeft ook hier aan dat er een significant verschil is tussen de *sequencing centers* ($p=1.3*10^{-14}$).



Figuur 18: De 306 individuen zijn van vier verschillende sequencing centers afhankelijk: WUGSC, SC, BI en Illumina. Vooral SC vertoont een hogere mediaan en variabiliteit.

3.1.4.3. Sequencing apparaat

De sequentiedata van de 306 individuen werd geanalyseerd door drie verschillende apparaten: de Illumina Genome Analyzer II (n=262), de Illumina Genome Analyzer (n=14) en de Illumina HiSeq 2000 (n=9) (Figuur 19). Voor 21 stalen werd er geen informatie over het toestel ter beschikking gesteld. Er is te zien dat de Illumina Genome Analyzer II data veel uitschieters bevat, wat eventueel te wijten is aan de beduidend grotere *sample size*. Een *one-way* ANOVA gaf aan dat er geen significant verschil waar te nemen is tussen de groepsgemiddeldes van de *sequencing* apparaten (p=0.208). Hoewel de gemiddelde telomeerlengtes dus zeer gelijkaardig zijn, lijkt er wel een verschil in standaardafwijking te zijn. Een *Bartlett* test bevestigt dit vermoeden van heteroscedasticiteit (p=0.0001284). Alle 108bp *reads* werden gegenereerd door de Genome Analyzer II. Er dient echter wel te worden opgemerkt dat het verschil tussen de *sequencing* toestellen niet betrouwbaar kan worden geëvalueerd door de beperkte *sample sizes*.



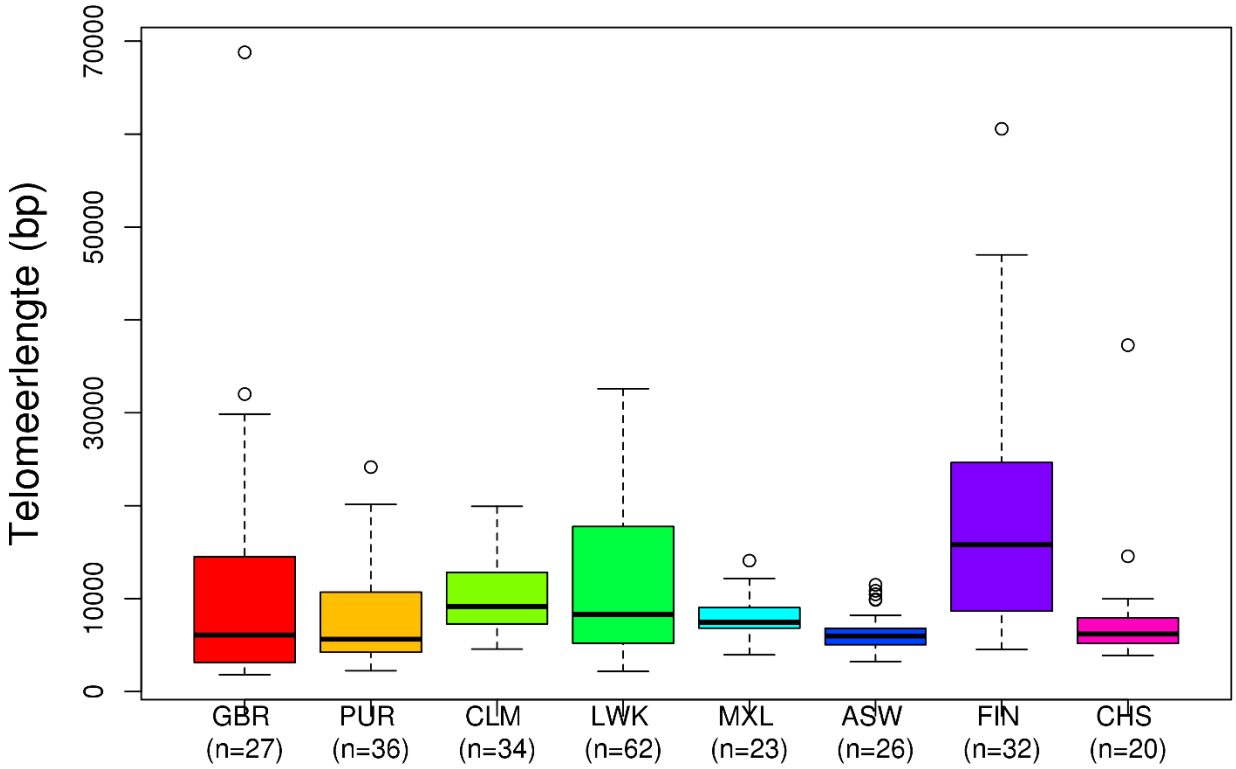
Figuur 19: Er werden drie verschillende sequencers gebruikt in de dataset van de 306 individuen: de Genome Analyzer II, Genome Analyzer I en de HiSeq 2000. De Genome Analyzer II was echter het meest gebruikte toestel binnen de geselecteerde dataset.

3.1.4.4. Populatie

De 306 individuen zijn van 13 verschillende populaties afkomstig:

- GBR: Brititen in Groot-Brittannië en Schotland (n=27)
- PUR: Puerto Ricanen van Puerto Rico (n=36)
- CLM: Colombianen van Medellin, Colombia (n=34)
- IBS: Iberische Populatie in Spain (n=6)
- YRI: Yoruba in Ibadan, Nigeria (n=8)
- LWK: Luhya in Webuye, Kenya (n=62)
- MXL: Mexicaanse afkomst van Los Angeles, USA (n=23)
- ASW: Amerikanen van Afrikaanse origine in SW USA (n=26)
- FIN: Finnen in Finland (n=32)
- CHS: Southern Han Chinesen (n=20)
- CEU: Utah bewoners met Noord-en West- Europese origine(n=4)
- CHB: Han Chinesen in Bejing, China (n=12)
- JPT: Japanesen in Tokya, Japan (n=16)

Er wordt vergeleken of er opmerkelijke verschillen zijn tussen de populaties sinds etnische afkomst een belangrijke rol kan spelen in telomeerlengte (Hunt, et al., 2008). Door de beperkte hoeveelheid data worden de IWS, YRI, CEU, CHB en JPT populaties niet opgenomen in de vergelijking op Figuur 20.

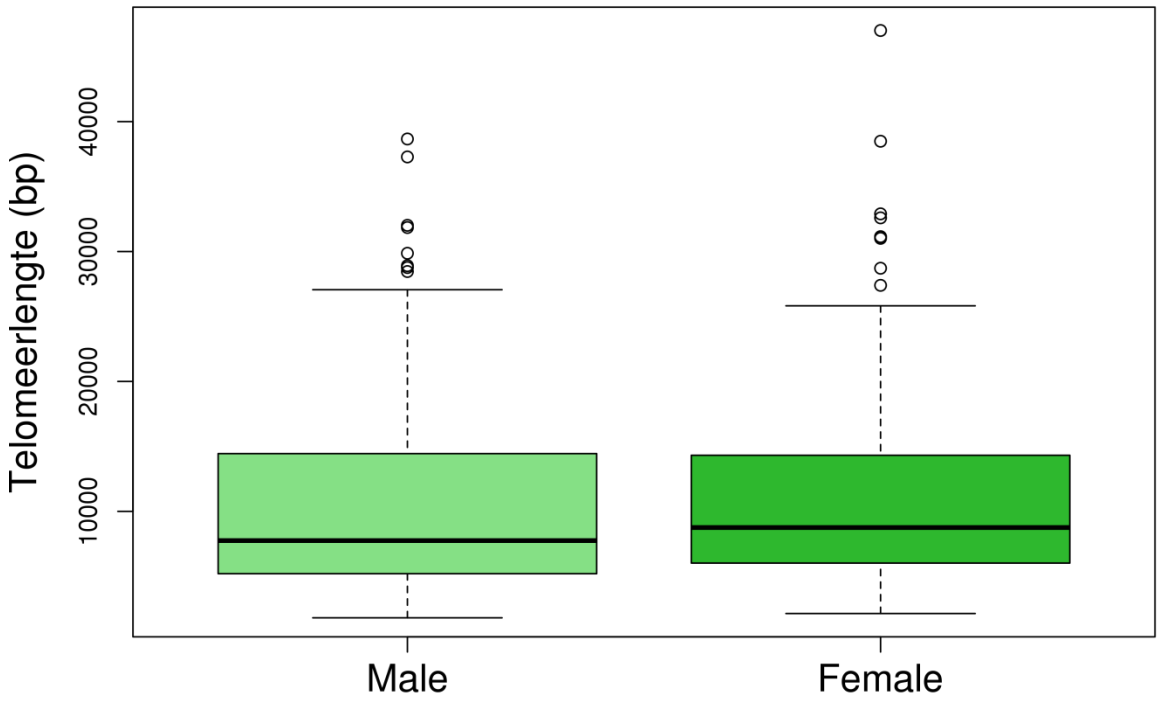


Figuur 20: Visuele weergave van de telomeerlengtes bij de belangrijkste populaties van de gekozen dataset.

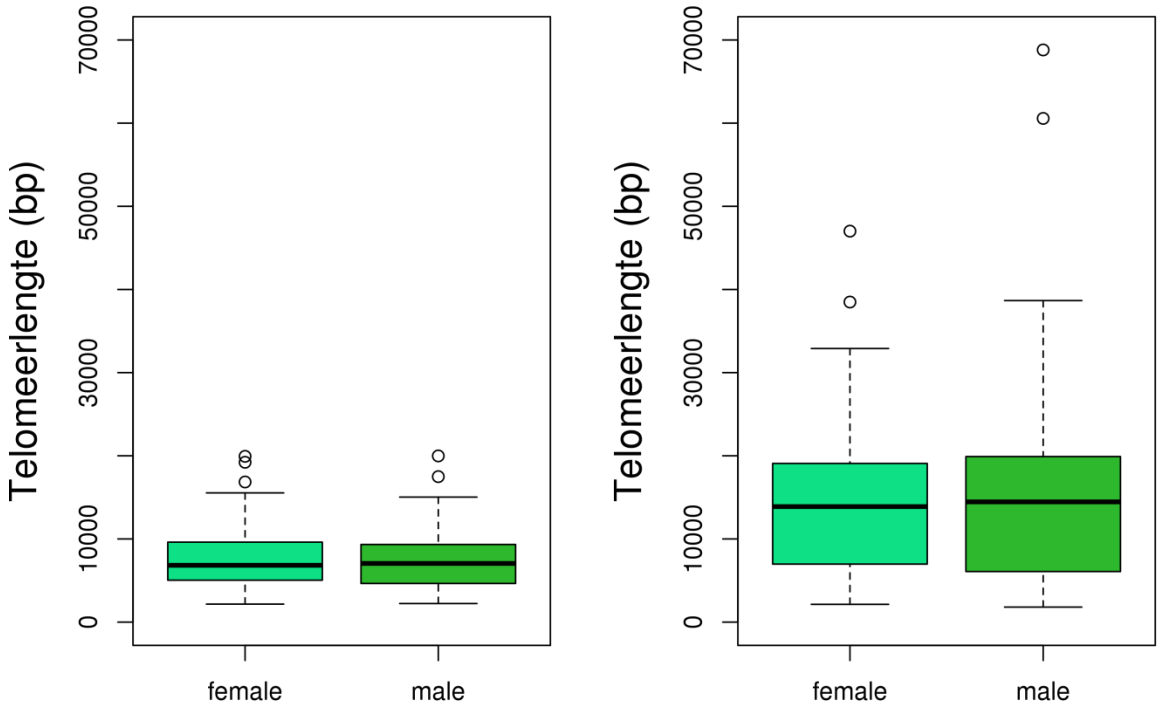
Een *one-way* ANOVA toont aan dat er wel een significant verschil waar te nemen is tussen de groepsgemiddeldes van deze populaties ($p=2.083*10^{-7}$). Ondanks de grote onderlinge verschillen in variabiliteit en medianen kan er niet uitgesloten worden dat deze variaties afkomstig zijn van eerdere beschreven technische aspecten.

3.1.4.5. Geslacht

Zoals in de literatuurstudie uitgelegd (zie 1.1: Inleiding: telomeren, telomeerlengte en zijn determinanten) zou er een significant verschil moeten zijn in telomeerlengte tussen man en vrouw. Op Figuur 21 is de telomeerlengtedata hiervoor weergegeven in twee boxplots. Het zeer beperkte verschil in gemiddelde telomeerlengte tussen man ($\bar{x}_M$ = 11225bp) en vrouw ($\bar{x}_F$ = 11183bp) is echter niet significant volgens een ongepaarde t-test (p = 0.968). De data afkomstig van de 108bp-*reads* heeft een zeer grote spreiding en kan de data erg beïnvloeden. Om die reden wordt de data opgesplitst in twee groepen: de 100/101-data, en de 108-data (Figuur 22).



Figuur 21: Voor de 306 individuen werd de gemeten telomeerlengte (bp) uitgezet in functie van het geslacht.



Figuur 22: Na het opsplitsen van de dataset in enerzijds 100/101-data en anderzijds 108-data wordt de telomeerlengte (bp) opnieuw uitgezet in functie van het geslacht. Er is een zichtbaar hogere mediaan en variabiliteit in telomeerlengtes bij de 108-data.

Na het opsplitsen was er nog steeds geen significant verschil tussen de gemiddeldes van man ($\bar{x}_{M\_100/101}$ = 7377bp) en vrouw ($\bar{x}_{F\_100/101}$ = 7762bp) via een ongepaarde t-test (p=0.4873) bij de 100/101-data. Ook blijkt geslachtsverschil in de 108-data: $\bar{x}_{M\_108}$ = 15881bp en $\bar{x}_{F\_108}$ = 14729bp via een ongepaarde t-test (p=0.5526) niet significant. Het verschil in gemiddeldes bij de 108-data vertoont zelfs een hogere telomeerlengte bij mannen waar de literatuur een algemeen hogere telomeerlengte bij vrouwen aanwijst. Dit is vermoedelijk te wijten aan de hoge variantie in de 108-data. De verklaring dat de 100/101-data geen significant verschil kan aantonen kan liggen aan het feit dat Computel zowel pseudotelomeren, subtelomeren als pure telomeren gebruikt bij de berekening van een gemiddelde telomeerlengte. Idealiter zouden de pure telomeren een duidelijker en statistisch significant geslachtsverschil moeten tonen. In ieder geval is de dataset waarmee hier gewerkt wordt relatief klein en wordt de benodigde *power* om het geslachtsverschil te bevestigen hier niet bereikt.

3.1.5. Technische variatie

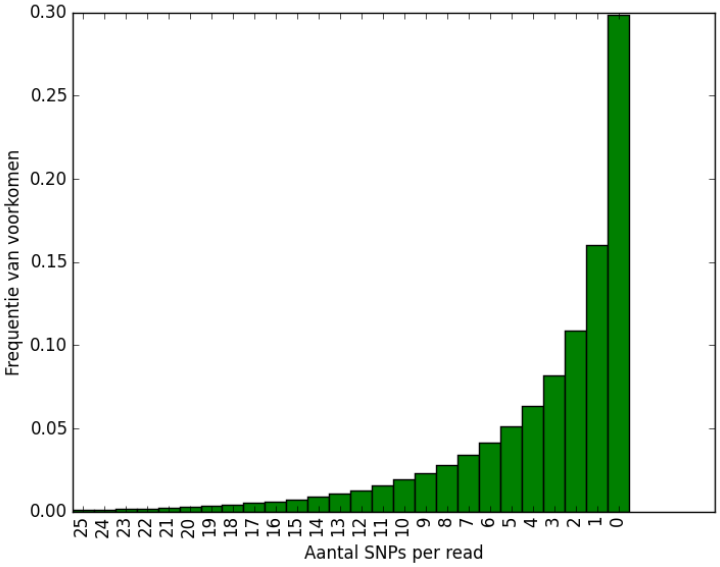
Binnen de dataset van de 306 individuen worden er grote verschillen in telomeerlengte waargenomen omwille van technische factoren. Aangezien de *read* lengte en het *sequencing center* echter niet onafhankelijk zijn, werd een *three-way* ANOVA uitgevoerd met deze technische informatie als factoren. Hieruit bleek dat enkel de *read* lengte ($p=1.05*10^{-15}$) maar niet het *sequencing center* (p=0.148) en *sequencing* apparaat (p=0.135) voor significante verschillen zorgen bij $\alpha = 0.05$.

Gezien de 108bp *read* lengte voor een significant verschil zorgt in vergelijking met de 100bp en 101bp *reads* (zie 3.1.4.1: Read lengte) wordt ervoor gekozen om de volledige dataset van 306 individuen op te splitsen in een 100/101bp dataset (n=161) en een 108bp dataset (n=145). In beide datasets zijn er individuen die van GBR, LWK en PUR-populaties afkomstig zijn. Hierdoor kunnen er vergelijkingen worden gemaakt waarbij enkel het technische aspect van de *read* data zou mogen verschillen.

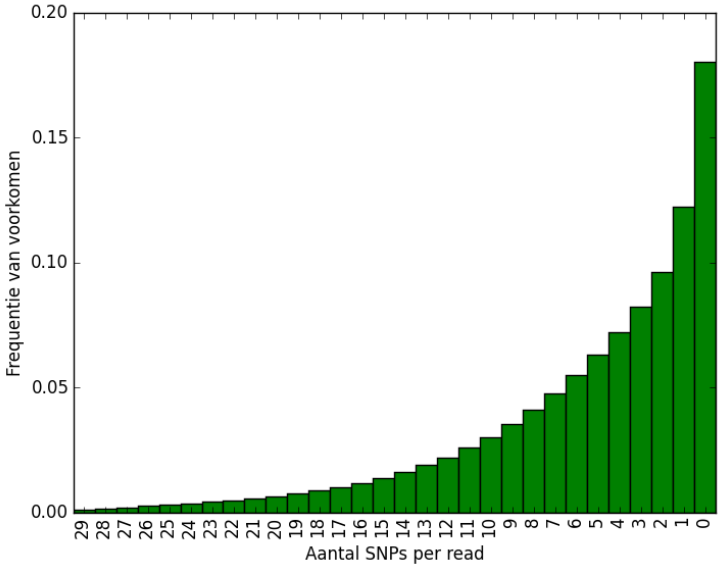
3.2. Fase 2

3.2.1. Telomerisch profiel

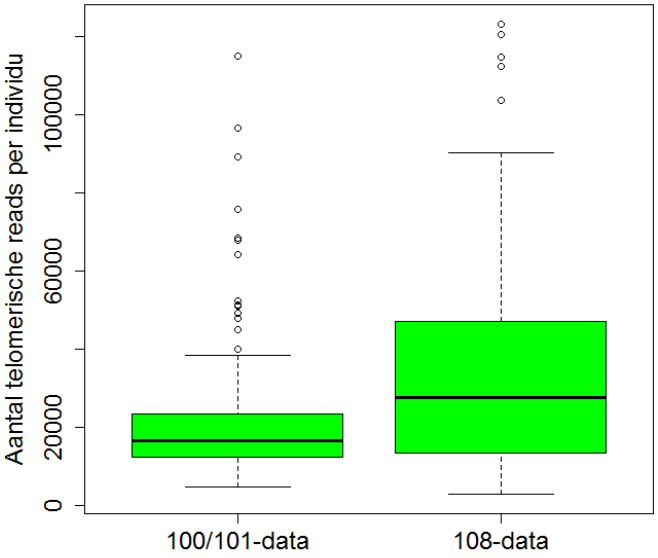
Het duidelijk verschil tussen de 100/101-data en 108-data kan ook gezien worden in de aligneringsdata. Hiervoor werd het aantal SNPs per *read* opgeteld over alle individuen heen voor de 100/101-data en de 108-data (Figuur 23 en Figuur 24). Op deze manier wordt een algemeen telomerisch profiel bekomen waardoor verschillen tussen beide datasets zichtbaar worden gemaakt. Sinds de hoeveelheid aligneringsdata verschilt tussen individuen, wordt aan elk individu eenzelfde gewicht toegekend. Op beide figuren worden frequenties lager dan 0.01 niet weergegeven. Er is een groot verschil waarneembaar tussen beide distributies waarbij de 108-data meer *reads* bevat met meer SNPs. Er zou verwacht worden dat een hogere aanwezigheid van SNPs in de data voor een lagere berekende telomeerlengte zorgt sinds er minder *reads* worden gealigneerd met de telomerische index. Echter wordt hier het tegenovergestelde waargenomen (Figuur 17). De verklaring hiervoor is dat de aanwezigheid van een hogere frequentie SNPs minder van belang is voor de berekende telomeerlengte als er meer gecapteerde telomerische *reads* zijn voor de 108-data. Een figuur waarin het aantal gecapteerde telomerische fragmenten voor elk individu wordt vergeleken tussen de 100/101-data en 108-data lijkt dit te bevestigen (Figuur 25).



Figuur 23: Telomerisch profiel van de 100/101-data. Van elke read binnen alle aligneringsbestanden wordt het aantal gedetecteerde SNPs opgeteld. Er wordt bijgehouden hoeveel reads er de verschillende aantallen SNPs bevatten. Sinds de aligneringsdata zeer heterogeen in grootte is tussen verschillende individuen, werd aan elk individu een gelijk gewicht toegekend zodat een niet-gewogen gemiddeld profiel verkregen wordt.



Figuur 24: telomerisch profiel van de 108-data. Net als in de 100/101-data wordt voor elk aligneringsbestand het aantal reads bijgehouden met de hoeveelheid SNPs die ze bevatten. Er werd een modificatie uitgevoerd waardoor er een niet-gewogen gemiddeld telomerisch profiel verkregen wordt.



Figuur 25: Vergelijking tussen het aantal gecapteerde telomerische fragmenten voor elk individu. Er is een duidelijk verschil in de hoeveelheid reads van elk aligneringsbestand tussen de 100/101-data (links) en de 108-data (rechts).

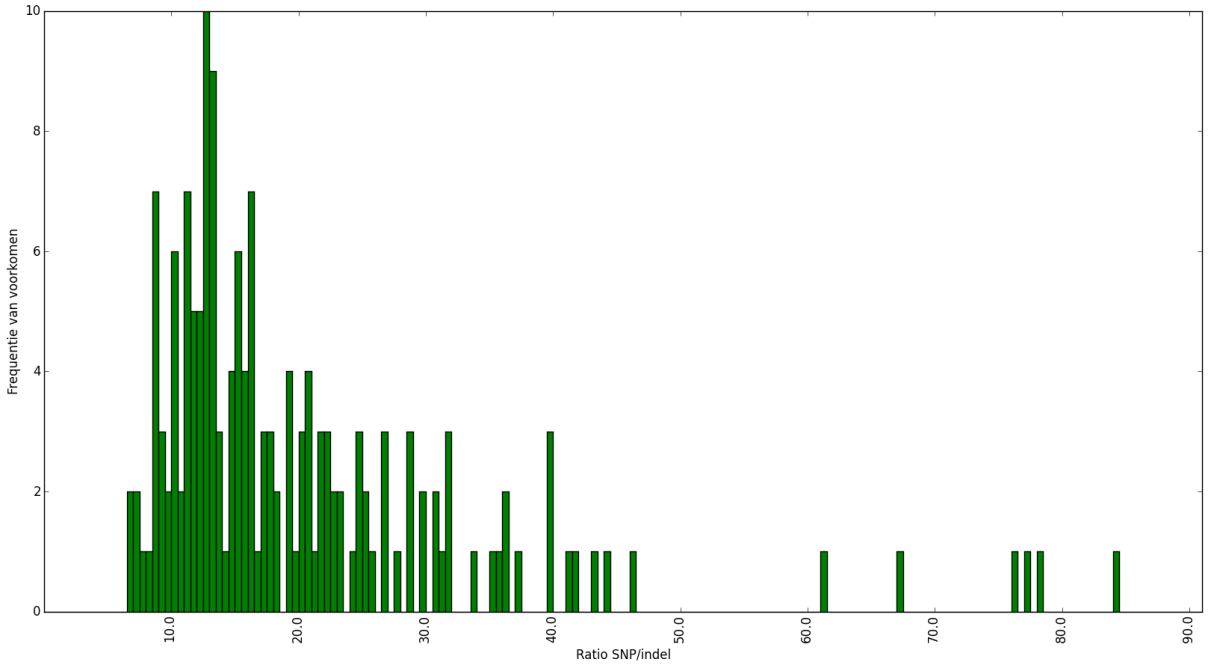
3.2.2. Indel error rate

Reads gegenereerd door Illumina sequencing hebben een zeer kleine indel *error rate* van $4*10^{-6}$ (Minoche, et al., 2011). Om die reden nemen we aan dat een waargenomen indel zo goed als zeker geen technisch artefact is en moet er dus geen correctie gebeuren voor een indel *error rate*. In een *ad random* gekozen individu NA18947 is de vaakst voorkomende indel TTA**G**GGG, gevolgd door TTA**A**GGG, TTAGG en TT**T**AGGG. Eerder werd gerapporteerd (zie 1.1: Inleiding) dat TTTAGGG een vaak voorkomende *repeat* was. De

vondsten in dit artikel werden echter bekomen door gebruik te maken van een probe met dit specifieke *repeat* (i.e. $(TTTAGGG)_4$) en sluiten ook niet uit dat er andere indels voorkomen binnen de telomerische fracties.

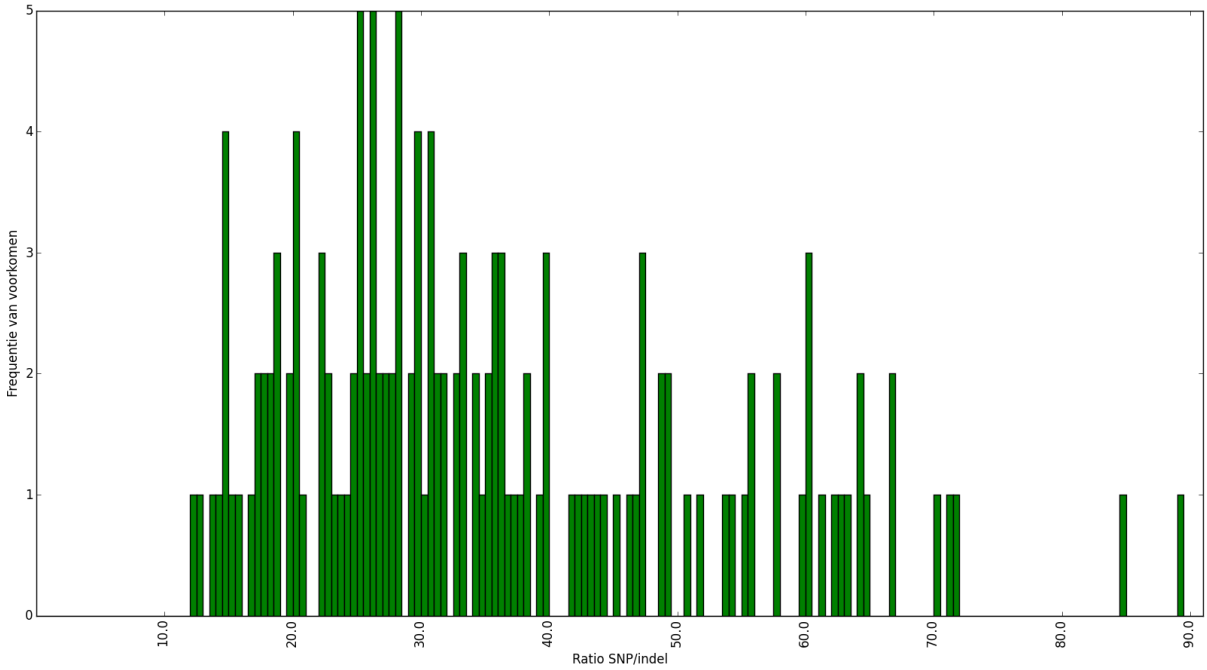
3.2.3. Verhouding SNPs – indels

Ter exploratie wordt de relatie tussen SNPs en indels voor elk van de 306 individuen onderzocht door middel van een SNP/indel ratio op te stellen en dit te visualiseren in een histogram. Dit wordt afzonderlijk weergegeven voor de 100/101-data en de 108-data (Figuur 26 en Figuur 27). Er dient opgemerkt te worden dat er bij de indels gekeken wordt naar de indel-gebeurtenissen, en niet het aantal geïnsereerde nucleotiden. Indien er bij de ratio's een drempelwaarde van 54 SNPs/indel wordt aangenomen, valt het op dat er bij de 108-data 25 uitschieters zijn waarvan 14 individuen van de LWK (Luhya in Webuye, Kenya) populatie. Bij de 100/101-data gaat het over 6 uitschieters waarvan 4 van de PUR (Puerto Ricans van Puerto Rico) populatie.



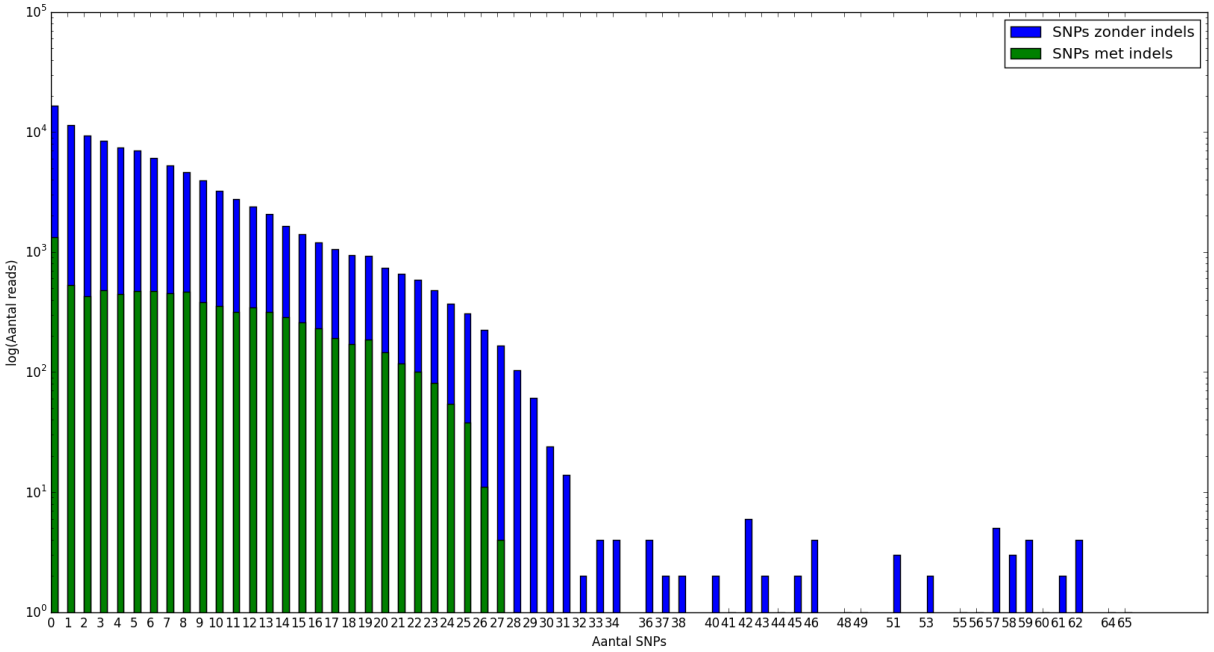
Figuur 26: Histogram van de SNP/indel ratio's van de 100/101-data individuen. De modus is te zien bij een SNP/indel-ratio = 13. Er is indicatie van een rechtse skew.

Voor de 108-data zijn twee individuen niet weergegeven op de figuur: NA19431 met een SNP/indel ratio van 114 en NA19438 met een SNP/indel ratio van 170.5. Er lijkt een verschil te zijn tussen de 100/101-data en de 108-data gezien de duidelijk verschillende modi (100/101-data: 13, 108-data: 26). Een *Wilcoxon rank-sum test* bevestigt dit vermoeden van verschil ($p = 2.817*10^{-9}$). Dit valt in lijn met de eerdere bevindingen waarin de 108-data globaal gezien meer nucleotidesubsitituties bevat. De grote variatie aan verschillende SNP/indel ratio's kan de variantie gezien op Figuur 17 deels verklaren.



Figuur 27: Histogram van de SNP/indel ratio's van de 108-data individuen. De modus is te zien bij een SNP/indel-ratio = 26.

Er wordt gekeken of het aantal *reads* met SNPs en indels verschilt met het aantal *reads* met SNPs zonder indels. Dit wordt voor een *ad random* gekozen individu NA19010 gedaan en is te zien op Figuur 28 waarbij het aantal *reads* in log-schaal geplaatst is.

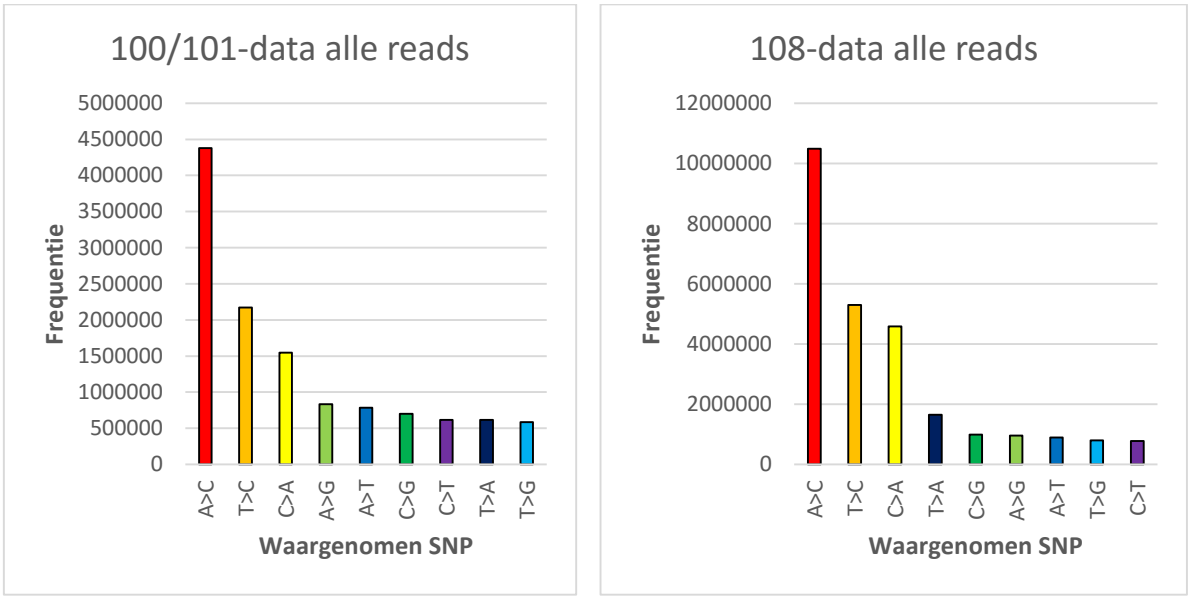


Figuur 28: Histogram waarin voor het individu NA19010 het aantal reads (log-schaal) met SNPs en indels vergeleken wordt met het aantal reads met SNPs zonder indels.

Een *Wilcoxon rank-sum test* op deze data voor het NA19010 individu wijst erop dat er geen significant verschil aantoonbaar is tussen het aantal *reads* met SNPs en indels en het aantal *reads* met SNPs zonder indels ($p = 0.8002$). Aangezien SNPs dus veel frequenter voorkomen en een gelijkaardige afwijking ten opzichte van pure telomeren vertonen als indels wordt vanaf hier enkel nog gefocust op SNPs.

3.2.4. Type SNPs

Voor het verkrijgen van een globaal overzicht wordt voor elke dataset en elke SNP-type het absolute aantal bepaald (Figuur 29). Er is te zien dat de drie meest voorkomende nucleotidesubstituties (A>C, T>C en C>A) in beide datasets gelijk zijn. Zoals eerder aangetoond is er echter een hoger aantal SNPs in de 108-dataset terwijl het aantal individuen in de 108-dataset (n=145) toch lager is dan het aantal individuen in de 100/101-dataset (n=161). De belangrijkste SNP is tevens de meest voorkomende *sequencing error* in Illumina-sequeneringsmethodes.



Figuur 29: Absolute aantal SNPs samen met het type voor de 100/101-dataset (links) en 108-dataset (rechts).

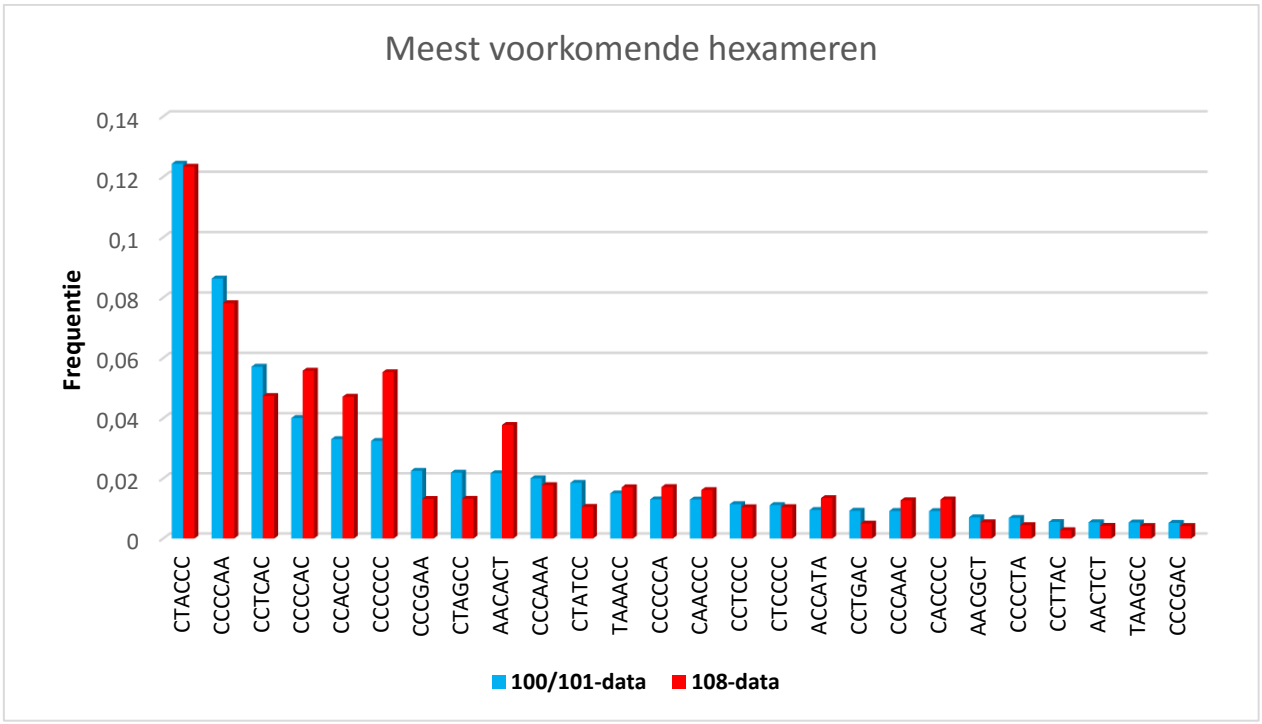
3.2.5. Type hexameren

Er werd eerder al ontdekt dat er binnen de telomerische fragmenten een aanwezigheid is van hexameren die verschillend zijn van CCCTAA (zie 1.1: Inleiding). Net als voor de SNPs wordt er een algemeen beeld geschetst welke de meest voorkomende alternatieve hexameren zijn in de telomerische fracties. Dit wordt apart weergegeven voor zowel de 100/101-data als de 108-data. Er wordt een frequentie opgesteld van de meest voorkomende hexameren over het totaal aantal getelde hexameren. De resultaten hiervan zijn te zien in Figuur 30. Een belangrijke opmerking is dat het script zo is ontwikkeld dat de vierde base in de hexameer de gemuteerde base is. In essentie komt het CTA**C**CC hexameer dus overeen met het muteren van het telomerische CCCTAA naar het hexameer CCCTA**C**. Een probleem met deze methode is dat hetzelfde voorkomende hexameer dubbel kan worden geteld, zoals het geval is met de hexameren CTCCCC en CCTCCC. Dit heeft als gevolg dat de frequenties van andere voorkomende hexameren heel beperkt onderschat worden.

De drie belangrijkste hexameren komen met een gelijkaardige frequentie terug bij de 100/101-data en de 108-data (CTACCC, CCCCAA en CCTCAC). Er kan gezien worden dat er een hoger aantal CCCCCC-meren (3 puntmutaties naar een cytosine), CCCCAC-en CCACCC-meren (2 puntmutaties naar een cytosine) worden waargenomen in de 108-data. Er is dus indicatie dat de mutatie naar een cytosine een belangrijk voorkomende *sequencing error* is in deze 108-data. In een eerder onderzoek (zie 1.1: Inleiding) werd ontdekt dat er TTGGGG en TGAGGG *repeats* voorkomen binnen de telomerische fracties. Deze *repeats*

werden ook hier teruggevonden onder de vorm van CCCCAA en CCTCAC en zijn respectievelijk de tweede en derde meest voorkomende hexameren. De bevestiging van deze hexameren via experimentele methodes betekent dat de *sequencing error* niet in dermate aanwezig is dat de biologisch relevante mutaties volledig worden gemaskeerd.

TRF-methodes zijn gebaseerd op een behandeling van genomisch DNA met een combinatie van restrictie-enzymes (zie 1.5.1: TRF). Idealiter knippen deze enzymen niet in telomerische fracties, maar worden wel alle subtelomerische fracties gedigereerd. Specifieke sets van restrictie-enzymen kunnen eventueel in dit laatste verschillen. Binnen dit globaal profiel van hexameren worden er inderdaad zo'n sites waargenomen, zelfs met een relatief hoge frequentie. Het hexameer CCTCAC bevat een restrictiesite voor het restrictie-enzym *Mn*II en komt voor met een frequentie van ca. 5%. Daarnaast bevat ook het hexameer CCTCCC deze restrictiesite en komt dit voor met een frequentie van ca. 1%. Alhoewel het restrictiepaar *Hph*I/*Mn*II minder gebruikt wordt dan het restrictiepaar *Hinf*I/*Rsa*I bevestigt deze figuur wel dat de aanwezigheid van deze specifieke hexameren een TRF-meting kunnen beïnvloeden. Dit wordt ook bevestigd door eerdere bronnen waarin experimentele methodes met *Hph*I/*Mn*II kortere restrictiefragmenten vormen dan met *Hinf*I/*Rsa*I, hoewel de gevolgen hiervan niet werden ingeschat (Baird, et al., 2005; Hunt, et al., 2008).

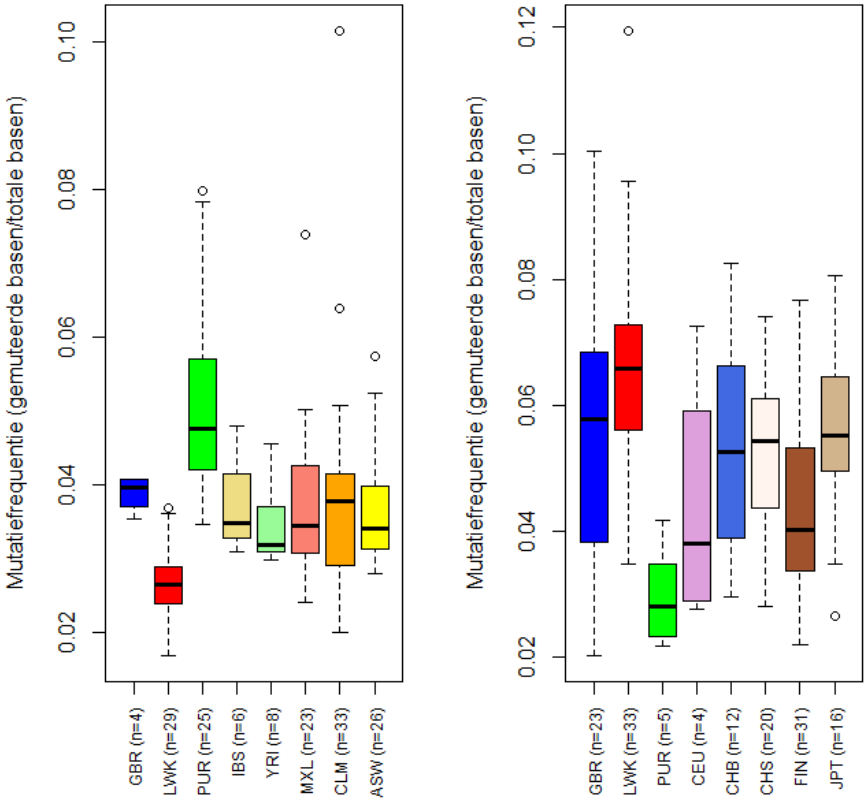


Figuur 30: Weergave van de vaakst voorkomende hexameren binnen de telomerische reads van de 306 individuen.

3.2.6. Mutatiefrequentie

Verschillende mutatiefrequenties tussen populaties kunnen eventueel gerelateerd zijn aan de geobserveerde verschillen in telomeerlengte. Op Figuur 31 zijn alle mutatiefrequenties te zien over de verschillende groepen heen, opgesplitst tussen de 100/101-data en 108-data. Als de gemeenschappelijke populaties tussen de 100/101-data en 108-data vergeleken worden zijn er grote verschillen waar te nemen. Het feit dat er meer nucleotidesubstituties te zien zijn binnen de 108-data dan binnen de 100/101-data (zie 3.2.1: Telomerisch profiel) heeft hier zeker mee te maken. Daarnaast bevatten de populaties GBR100 en PUR108 te weinig individuen waardoor onderstaande data als inconsistent wordt beschouwd. De

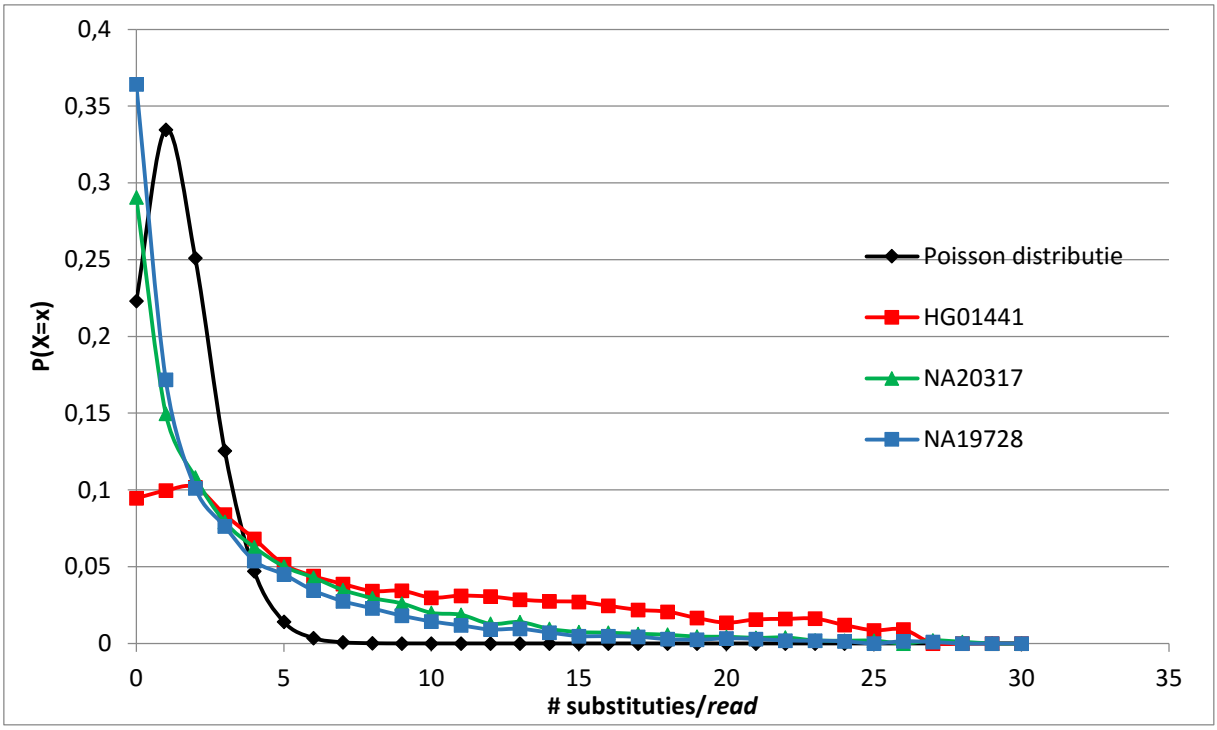
mutatiefrequenties van de 108-data worden waarschijnlijk teveel beïnvloed door onderliggende *sequencing errors* om uitspraken te kunnen doen over een biologisch relevante mutatiefrequentie.



Figuur 31: Mutatiefrequenties over alle telomerische fracties werden over de verschillende populaties weergegeven en opgedeeld tussen de 100/101-data (links) en 108-dat a (rechts).

3.2.7. Sequencing error rate

Naast mutaties bevatten de telomerische *reads* ook nog *sequencing errors*, die - gezien de aanwezigheid van verschillende technische verschillen tussen samples (*read* lengte, toestel & *sequencing center*) - moeilijk eenduidig te kwantificeren vallen. Bij het analyseren van recurrente SNPs is het wel belangrijk dat er hiervoor gecorrigeerd wordt. Voor een gegeven *error rate* kan via een Poisson-distributie er bepaald worden welke *reads* er vooral ware SNPs bevatten, dan wel *sequencing errors* (Figuur 32). Hier wordt arbitrair voor een *sequencing error rate* van 1.5% gekozen, dat ook in de Poisson-distributie als λ gehanteerd wordt. Deze relatief hoge waarde werd gekozen omdat de data nog in de begintijden van *second generation sequencing* werd gegenereerd. Van drie *ad random* gekozen individuen wordt er vanuit de aligneringsdata voor elke *read* het aantal nucleotidesubstituties bijgehouden, en worden de relatieve frequenties hiervan uitgezet tegen de Poisson-distributie.



Figuur 32: Een Poisson-distributie van de verwachte sequencing error wordt uitgezet tegen de waargenomen nucleotidesubstituties in reads van 3 ad random gekozen individuen (HG01441, NA20317 en NA19728). De kans op een sequencing error is het hoogst bij reads met minder of gelijk aan drie SNPs per read.

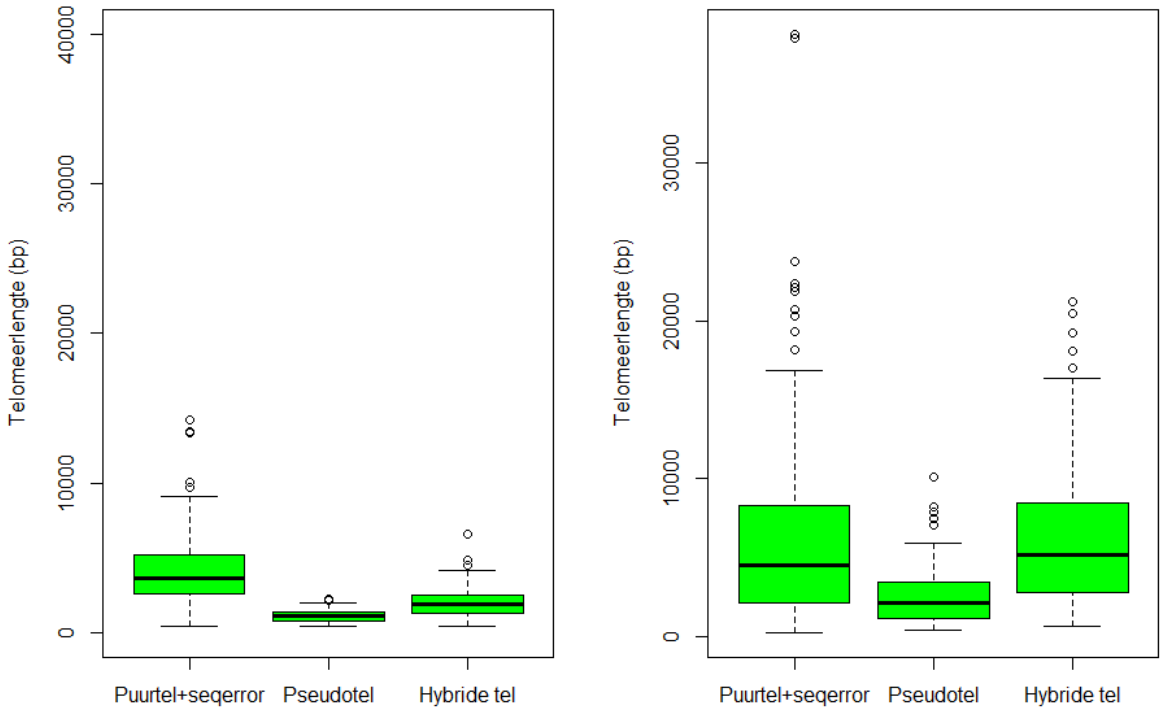
Gegeven een gemiddelde *sequencing error rate* van 1.5%, kan er waargenomen worden dat de kans op een *sequencing error* het hoogst is bij *reads* die minder of gelijk aan drie nucleotidesubstituties bevatten. In dat opzicht wordt ervoor gekozen om als initiële arbitraire grens de *reads* te kiezen met minder dan vier nucleotidesubstituties als pure telomerische *reads* met *sequencing errors*. Voor de individuen HG01441, NA20317 en NA19728 (allen vanuit de 100/101bp dataset) gaat dit dan over respectievelijk 38%, 62.8% en 71.3% van de *reads*. Aangezien op basis van TRF telomeerlengtes van grootteorde 10 kbp worden bekomen (Bekaert, et al., 2007), en de subtelomeerlengte op enkele kbp wordt geschat lijkt de keuze van deze *threshold* op het eerste zicht in overeenstemming met de literatuur wat betreft de NA20317 en NA19728 individuen (Counter, et al., 1992; Greider & Harley, 1996). Voor het individu HG01441 wordt er echter een lager percentage (38%) teruggevonden. Dit impliceert dat ongeveer $2/3^{e}$ van de gemeten telomeren in feite subtelomeren zijn, terwijl grootteorde zowat $1/3^{e}$ wordt verwacht (subtelomeerlengte van ca. 3kbp op 10kbp) (Counter, et al., 1992). Het nog verhogen van de geselecteerde *sequencing error rate* (waardoor meer “pure” telomeren zouden worden gecapteerd) wordt echter niet opportuun geacht aangezien deze al hoog wordt ingeschat.

3.3. Fase 3

3.3.1. Opsplitsing in telomerische fracties

Op basis van de vooropgezette *sequencing error rate* in sectie 3.2.7 worden de telomeren opgesplitst in drie verschillende fracties: de ‘pure’ telomeren die verondersteld enkel *sequencing errors* bevatten, de pseudotelomeren met naast *sequencing errors* ook mutaties, en de hybride telomeren (vanaf twee naast elkaar voorkomende nucleotidesubstituties). De pseudotelomeren en hybride telomeren worden samen beschouwd als de subtelomeren. Belangrijk is hierbij op te merken dat pseudo- en hybride telomeren in essentie met distale resp. proximale subtelomeren overeenkomen, maar wel conceptueel verschillend zijn: bij de laatste gaat het over de positie t.o.v. van het telomeer, terwijl het bij de eerste over de aanwezigheid van enkele mutaties gaat dan wel grootschalige afwijkingen van de telomerische sequentie en hun impact op telomeerlengtebepalingen..

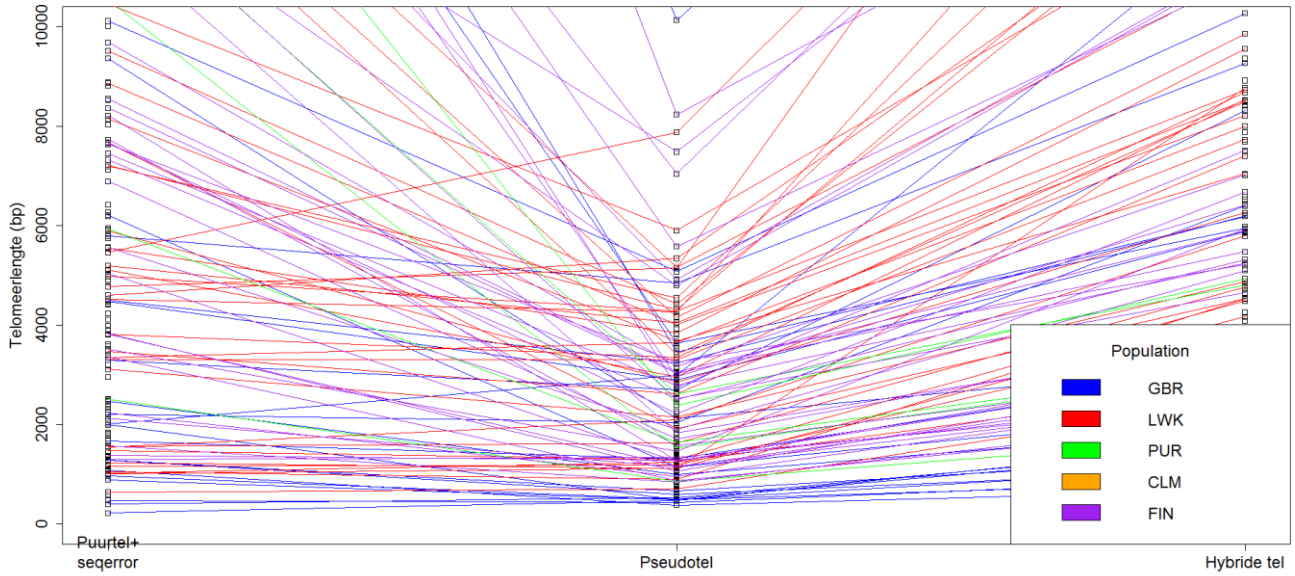
Bij vergelijkingen in telomeerlengtes worden acht individuen verwijderd uit de totale dataset (HG00731, HG00272, HG01048, HG01055, HG01066, HG01067, HG01079 en HG01125) waarbij de lengtes van de aparte telomerische fracties samen niet overeenstemmen met de totale telomeerlengte. Dit is te wijten aan een functionele tekortkoming in het script waardoor wellicht de *coverages* (waarmee telomeerlengtes berekend worden) verkeerd bepaald werden. Op Figuur 33 is te zien dat er bij de 108-data duidelijk meer hybride telomeren gecapteerd zijn. Dit zou het verschil in telomeerlengte tussen de 100/101-en 108-datasets deels kunnen verklaren. De grotere varianties die te zien zijn in de 108-data zijn verspreid over de verschillende telomerische fracties en hebben een gelijkaardige vorm als in Figuur 17.



Figuur 33: Weergave van de telomeerlengtes voor de pure telomeren + sequentie-error, de pseudotelomeren en de hybride telomeren. Dit werd uitgevoerd voor de 100/101-data (links) en voor de 108-data (rechts)

3.3.2. Vergelijking van telomerische fracties volgens populatie

Omdat de lengtes van afzonderlijke telomerische fracties kunnen variëren tussen individuen wordt aan de hand van *stripcharts* de relatie tussen deze verschillende telomerische fracties weergegeven voor elk individu. Dit is te zien op Bijlage 1: Stripchart van 100/101-data en Bijlage 2: Stripchart van 108-data. Binnen de 100/101-data volgen de telomerische fracties een gemeenschappelijk patroon waarbij de puurtelomerische telomeerlengte globaal gezien hoger is dan de pseudotelomerische telomeerlengte. In de 108-data daarentegen is te zien hoe bij sommige individuen de pseudotelomerische fractie hoger is dan de puurtelomerische fractie. Dit betekent dat er minder *reads* als puurtelomerisch worden beschouwd dan als pseudotelomerisch. Mogelijks is dit groepsspecifiek en biologisch relevant, maar meer waarschijnlijk heeft dit te maken met het hoger aantal *sequencing errors* in de 108 data waardoor minder *reads* als pure telomeren worden geclassificeerd. De *stripchart* wordt voor de belangrijkste populaties weergegeven binnen de 108-data (Figuur 34).



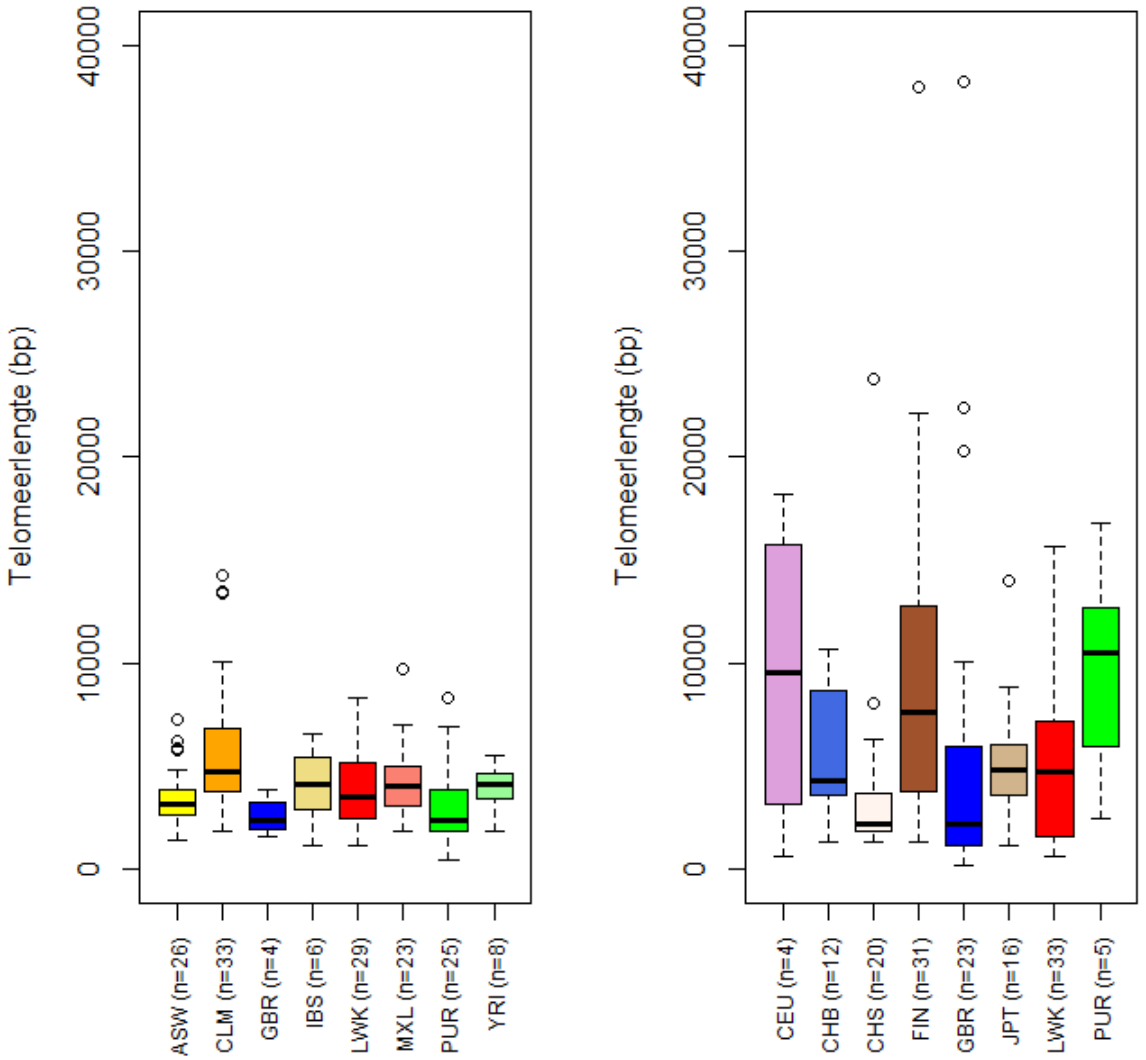
Figuur 34: Voor de grootste populaties binnen de 108-dataset werd de relatie tussen de lengte van de pure telomeren + sequentie-error, pseudotelomeren en hybridetelomeren in kaart gebracht. De y-as werd herschaald zodat de meeste verbindingen duidelijk zichtbaar zijn.

Op deze exploratieve figuur gaan diverse individuen van de GBR en LWK populatie tegen de algemene dalende trend in tussen de pure telomerische en pseudotelomerische fragmenten. Dit verschijnsel wordt echter niet gezien bij LWK-individuen van de 100/101-dataset. Waarschijnlijk is dit dus geen biologisch maar een technisch artefact. Er dient te worden opgemerkt dat het ook LWK-individuen waren die de meest extreme SNP/indel ratios hadden. Om verdere verschillen tussen populaties waar te nemen worden de absolute lengtes van de aparte telomerische fracties voor elke populatie afzonderlijk weergegeven (Figuur 35, Figuur 36 en Figuur 37).

Eerder gerapporteerde verschillen in telomeerlengte (op basis van TRF) kunnen eventueel te wijten zijn aan verschillen in pseudotelomeren (i.e. mutaties) in plaats van de functionele, pure telomeren, daarom wordt hier bekeken of de lengtes verschillen tussen de populaties in het 1000 Genomes Project.

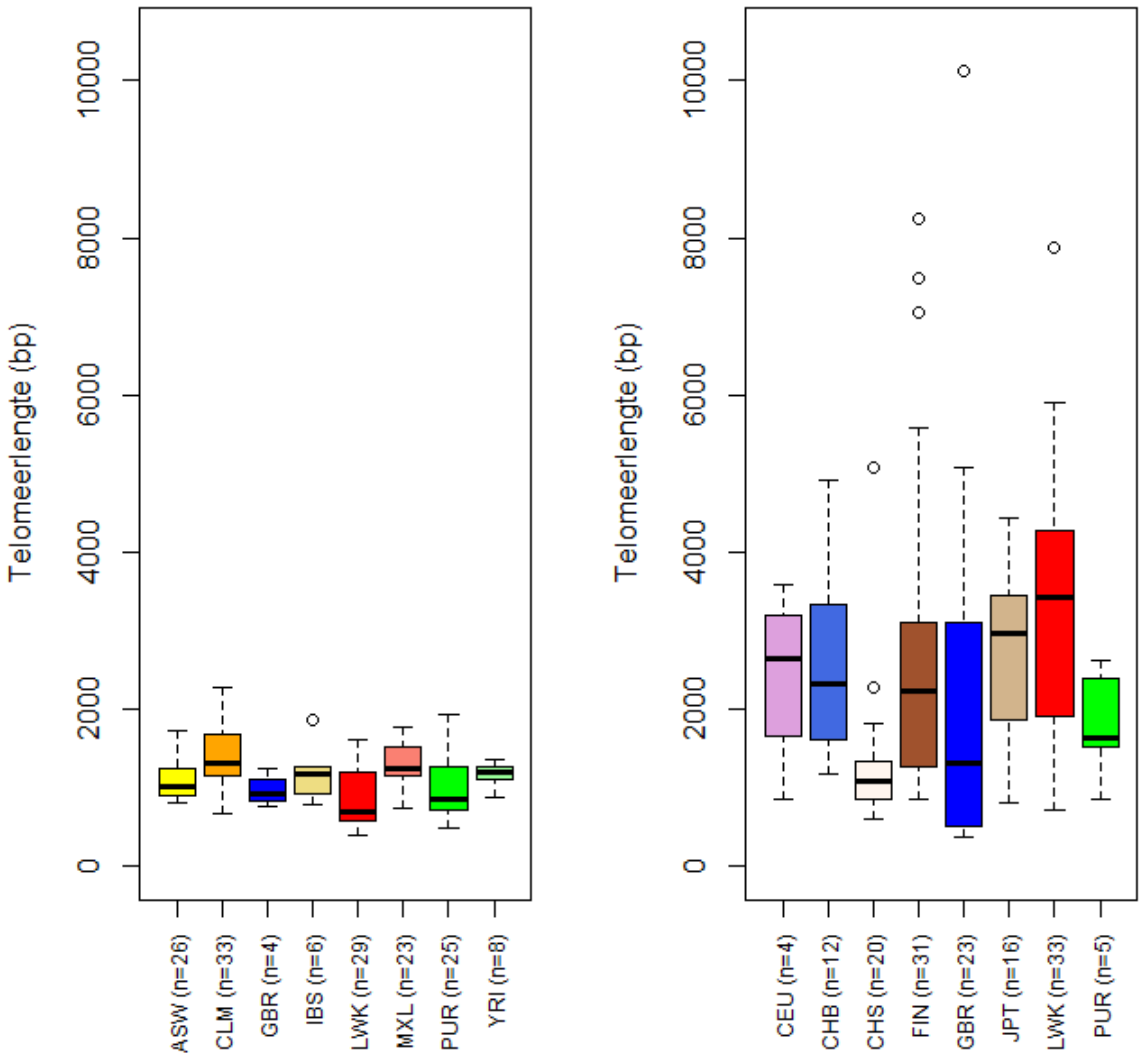
Op Figuur 35 zijn de verschillen in lengte van de pure telomeren te zien voor de verschillende populaties. Bij de 100/101-data en 108-data zijn er drie gemeenschappelijke populaties: GBR, LWK en PUR. Echter is de GBR-populatie bij de 100/101-data en de PUR-populatie bij de 108-data klein, respectievelijk vier en vijf individuen. Er is te zien dat de mediaan bij de GBR en LWK-populatie tussen beide datasets ongeveer overeenstemt. Het grote verschil in de PUR-populatie tussen de 100/101-data en 108-data kan verklaard worden door de kleine populatie in de 108-data. Een extra waarde kan een groot effect hebben op de variantie en dus de boxplot. Over het algemeen wordt er opnieuw een hogere variantie waargenomen in de 108-data. Een *Bartlett* test geeft aan dat de varianties van beide datasets zowel bij de 100/101-data ($p=0.0005106$) als de 108-data ($p=9.684*10^{-7}$) heterogeen zijn. *One-way* ANOVA's van de 100/101-data ($df=7$, $p=0.0001071$) als de 108-data ($p=0.02831$) geven aan dat de gemiddeldes van de groepen niet allen gelijk zijn. Doordat er geen homogeniteit in varianties is wordt gebruik gemaakt van de *Games-Howell* test. Bij de 100/101-data is er een verschil in gemiddeldes van de ASW, GBR en PUR groep t.o.v. de CLM groep. Bij de 108-data is er een verschil tussen de CHS en FIN-groep. Hoewel

zeker relevant, blijft onduidelijk in welke mate deze verschillen effectief aan onderliggende biologie te wijten zijn.



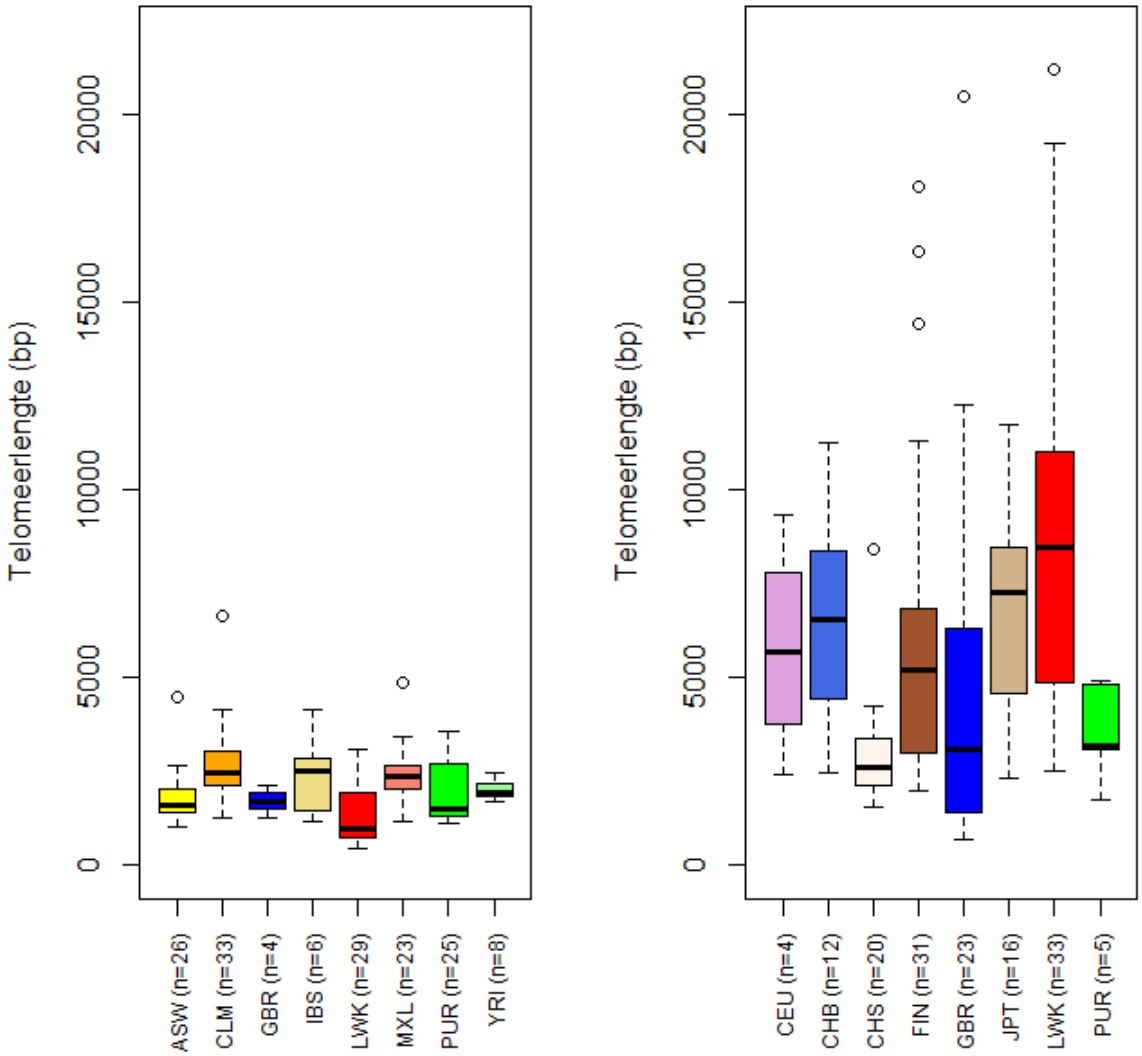
Figuur 35: Op deze figuur zijn de lengtes van de "pure" telomeren (met sequencing error) te zien voor de populaties binnen de 100/101-data (links) en de 108-data (rechts).

Het verschil in lengte van de pseudotelomeren tussen de 100/101-data en 108-data lijkt groot (Figuur 36). Bij de LWK-populatie hebben de medianen idealiter eenzelfde waarde. Helaas is dit niet het geval (ca. 1000bp en ca. 3500bp). Hetzelfde is te zien op Figuur 36 dat de lengtes van hybride telomeren vergelijkt tussen de populaties. Hier verschillen de medianen van de 100/101-data en 108-data in de LWK populatie zelfs sterker: respectievelijk ca. 1000bp en ca. 8000bp. Dit bevestigt Figuur 24**,** waar er meer *reads* zijn met SNPs in de 108-data. De varianties van beide datasets hebben volgens een *Bartlett* test opnieuw ongelijke varianties (100/101-data: p=0.01974 en 108-data: p=0.0003849). *One-way* ANOVA's van de 100/101-data ($p=7.887*10^{-9}$) en 108-data (p=0.00841) kunnen wijzen op een groter verschil in gemiddeldes binnen de pseudotelomeren dan de pure telomeren. Echter, naast de grote absolute verschillen tussen 100/101 en 108 data, vertonen ook de algemene trends voor de gemeenschappelijke populaties (GBR, LWK en PUR) geen gezamenlijk patroon. Het gebrek aan consistentie in de resultaten toont daarom ook aan dat de impact van technische variantie vermoedelijk groter is dan deze van de onderliggende biologie.



Figuur 36: Weergave van de lengte van de pseudotelomeren voor de verschillende populaties binnen de 100/101-data (links) en de 108-data (rechts).

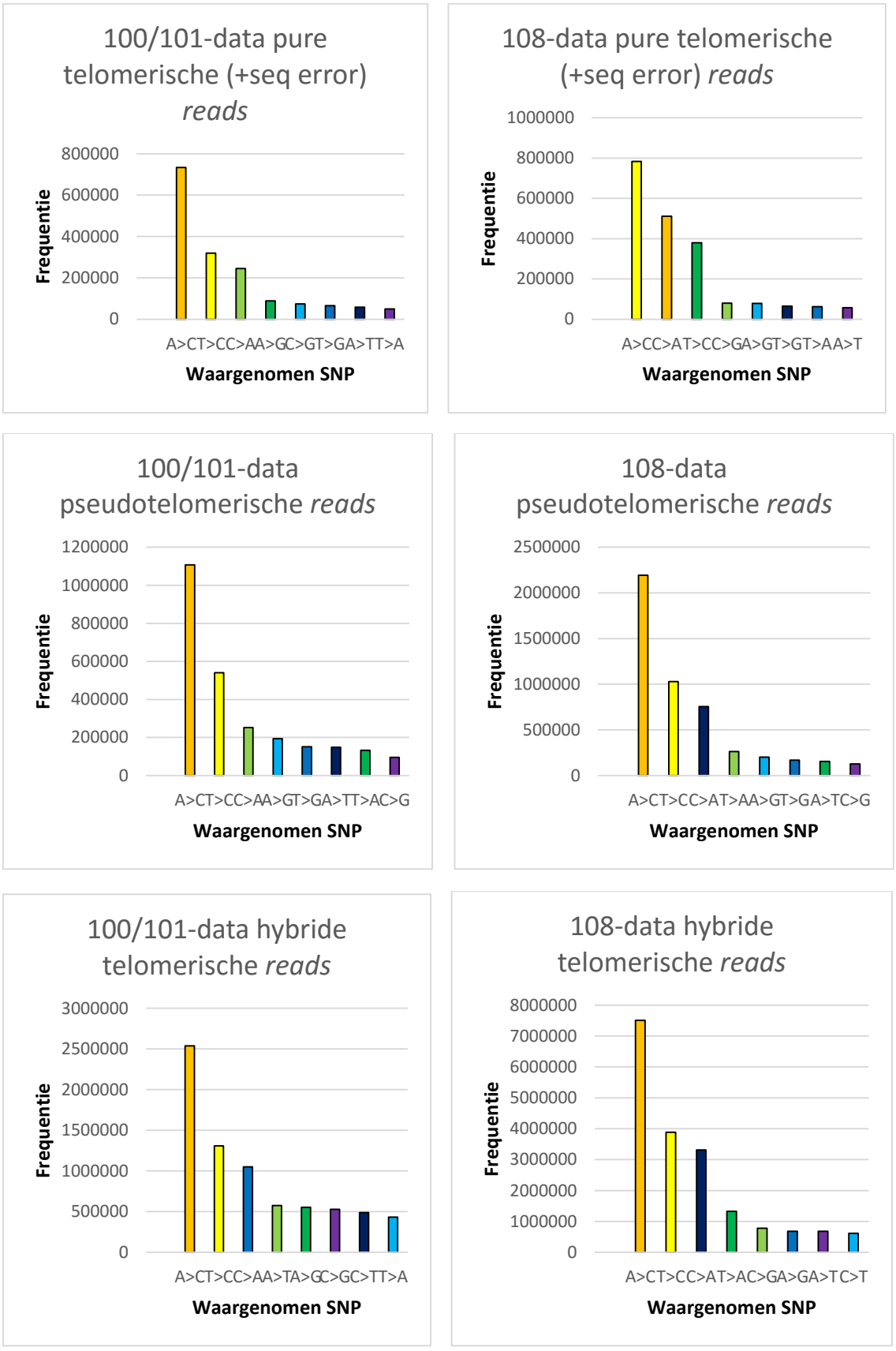
Voor de volledigheid worden ook hybride telomeren bestudeerd voor elke populatie, zowel voor de 100/101-data als de 108-data (Figuur 37). Er is volgens een *Bartlett* test opnieuw een ongelijkheid in varianties van de 100/101-data ($p=0.003619$) en de 108-data ($p=1.938*10^{-5}$). De *one-way* ANOVA's van de 100/101-data ($p=1.366*10^{-8}$) en de 108-data ($p=2.879*10^{-5}$) vertonen gelijkaardige resultaten als bij de pseudotelomeren. Er is een indicatie dat verschillende populaties grotere verschillen tonen in gemeten telomeerlengtes bij de pseudotelomeren en hybride telomeren dan bij de pure telomeren. Aangezien er een groot verschil is tussen de gemeenschappelijke populaties GBR, LWK en PUR moeten deze bevindingen ook hier met een korrel zout genomen worden.



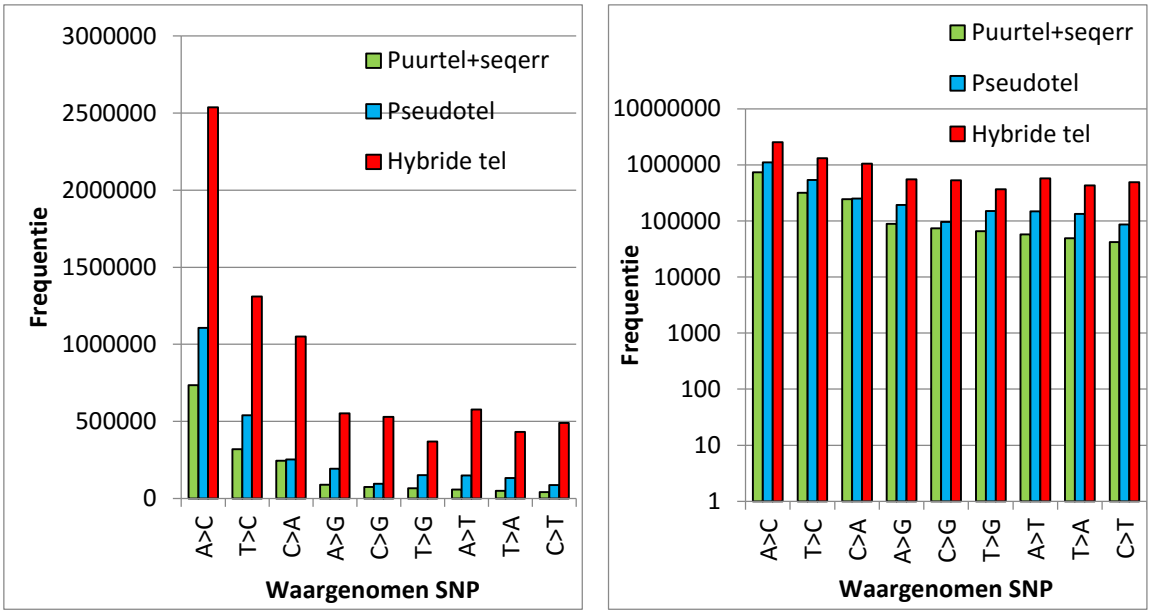
Figuur 37: De lengte van de hybride telomeren werden uitgedrukt voor de verschillende populaties binnen de 100/101-data (links) en de 108-data (rechts).

3.3.3. Karakterisatie van SNPs

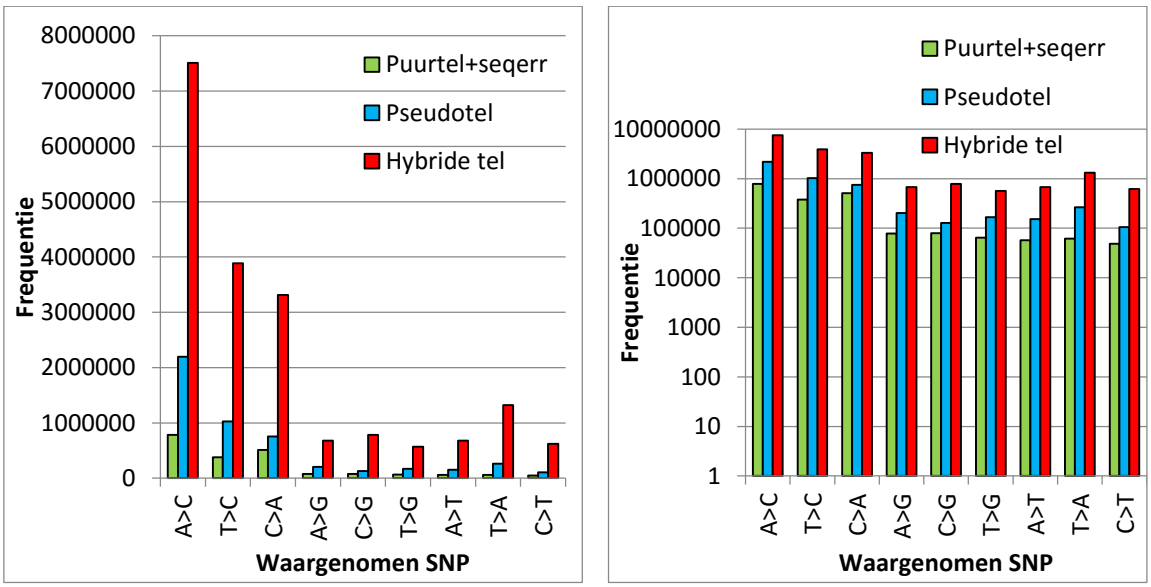
De karakterisatie van de gedetecteerde nucleotidesubstituties over de verschillende telomerische fracties zou duiding moeten geven welke de specifieke *sequencing errors* zijn (pure telomeren) en welke de ware voorkomende mutaties (pseudo en hybride telomeren). Op Figuur 38 werden voor de verschillende telomerische fracties alle verschillende SNPs gevisualiseerd. Figuur 39 en Figuur 40 vatten deze resultaten samen. Er is te zien dat de belangrijkste nucleotidesubstituties (A>C, T>C en C>A) over de verschillende fracties gelijk blijven, wat er op kan wijzen dat ofwel (i) dit enkel technische fouten zijn (*sequencing errors*), dan wel dat (ii) dezelfde mutaties over de volledige telomerische *read* (puur + pseudo + hybridetelomeer) voorkomen, en dat een meer strikte definitie van de pure telomeren noodzakelijk was. Binnen de pure telomerische *reads* van de 108-data komt de C>A substitutie uitzonderlijk meer voor dan de T>C substitutie. SNPs die minder voorkomen zijn wel in een hogere mate zichtbaar in de hybride telomeren. Sinds deze telomeerfractie het overgangsgebied vormt tussen functionele genen en de pure telomeren lijkt het normaal om hier een grotere verscheidenheid aan nucleotiden terug te vinden.



Figuur 38: Overzicht van de frequentie SNPs binnen de verschillende telomerische fracties voor zowel de 100/101-data als de 108-data.



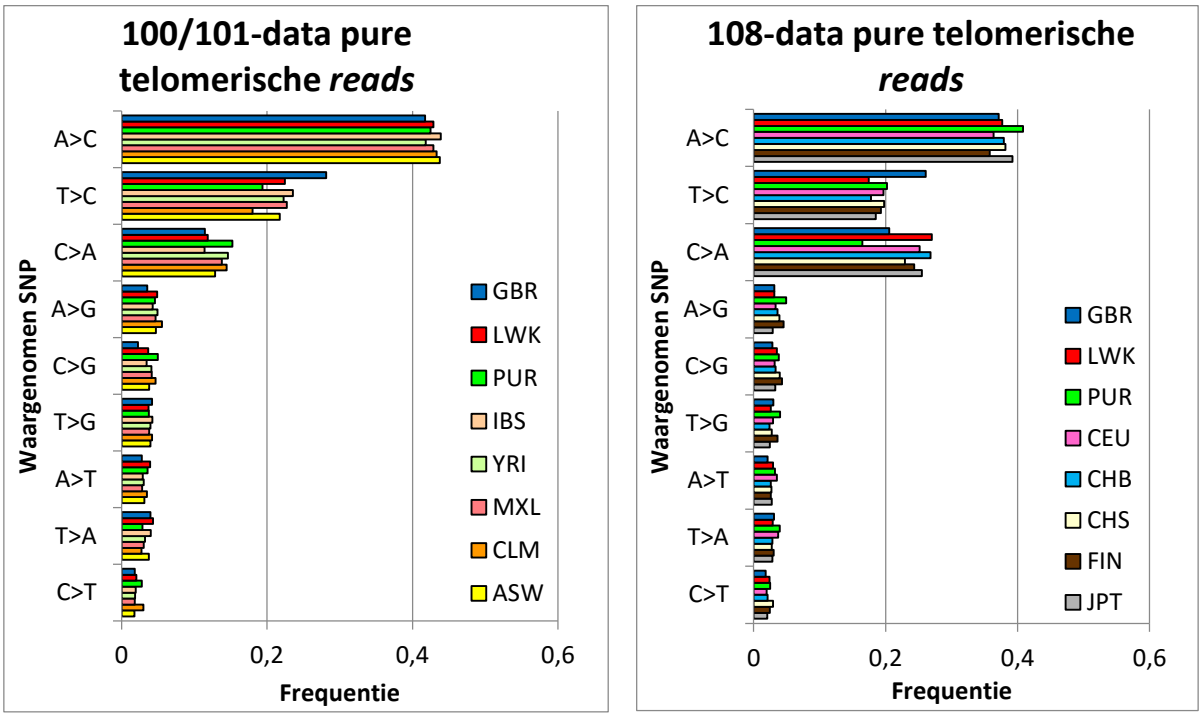
Figuur 39: Overzicht van de frequentie SNPs die in elke verschillende telomeerfractie te vinden zijn binnen de 100/101-data.

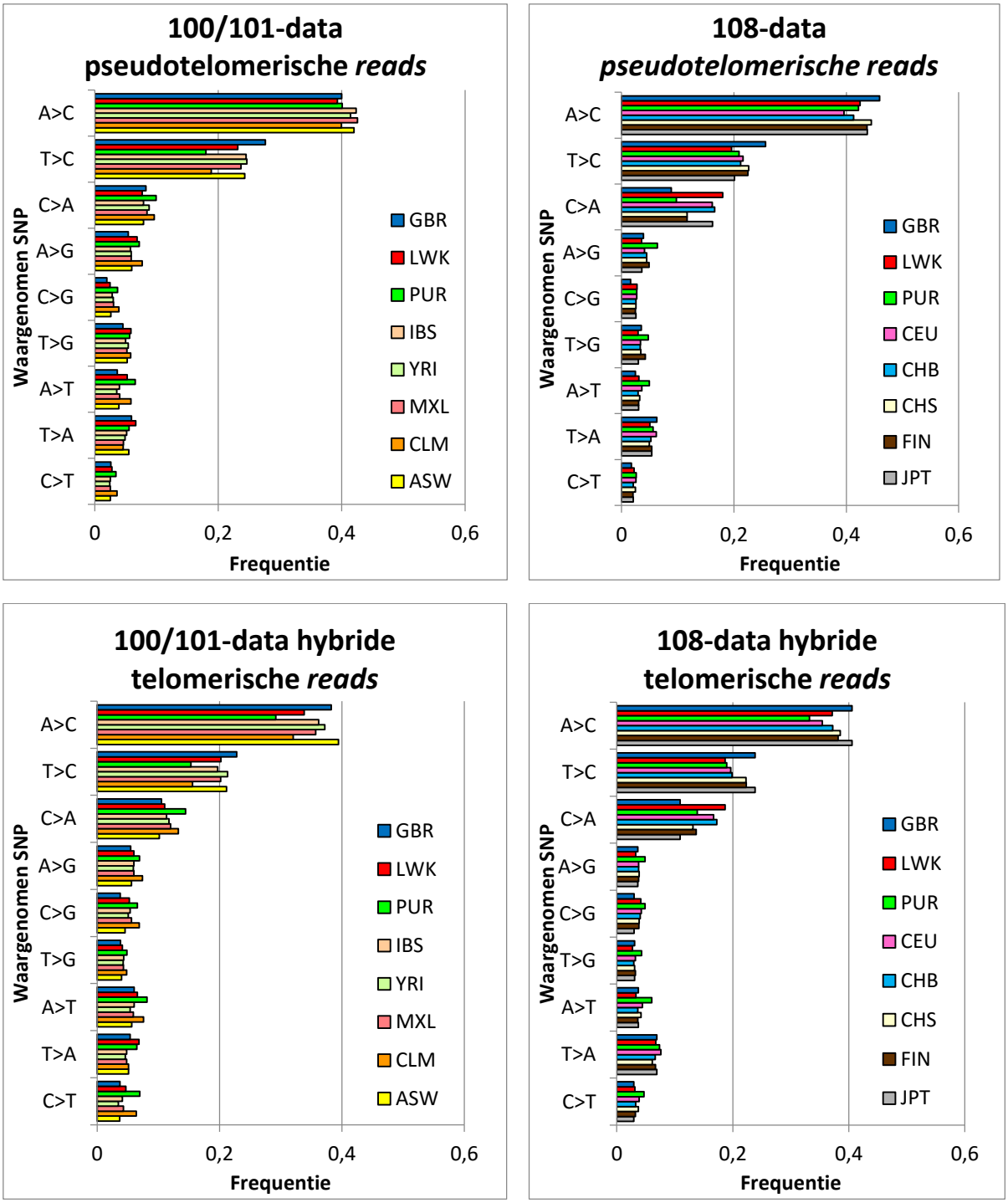


Figuur 40: Overzicht van de frequentie SNPs die in elke verschillende telomeerfractie te vinden zijn binnen de 108-data.

3.3.4. Karakterisatie van SNPs volgens populatie

In 3.3.3 werden de SNPs gekarakteriseerd onafhankelijk van de populaties die elk in zekere mate biologische variatie bezitten. Eventuele populatiegebonden verschillen worden om deze reden exploratief vergeleken voor elke telomerische fractie en zijn te zien op Figuur 41. Door technische variatie zijn de verhoudingen in SNP-frequenties tussen de gemeenschappelijke populaties GBR, LWK en PUR niet gelijk. Om die reden dienen onderstaande resultaten dus kritisch te worden bekeken. Er kan gezien worden dat het voorkomen van de T>C substitutie het hoogst is bij de GBR-populatie in beide datasets. Sinds de GBR-data binnen de 100/101-data (WUGSC) in contrast staat met de GBR-data binnen de 108-data (SC) is het op het eerste zicht aannemelijk dat dit een gevolg van biologische variatie kan zijn. Er moet echter opnieuw opgemerkt worden dat de GBR-groep van de 100/101-data slechts vier individuen bevat en dus erg kan variëren.

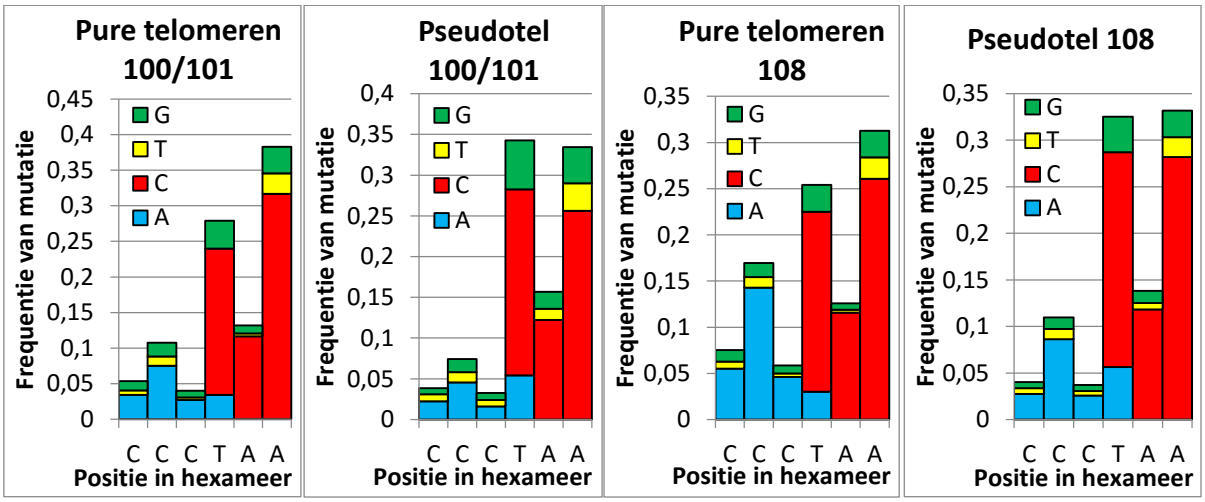




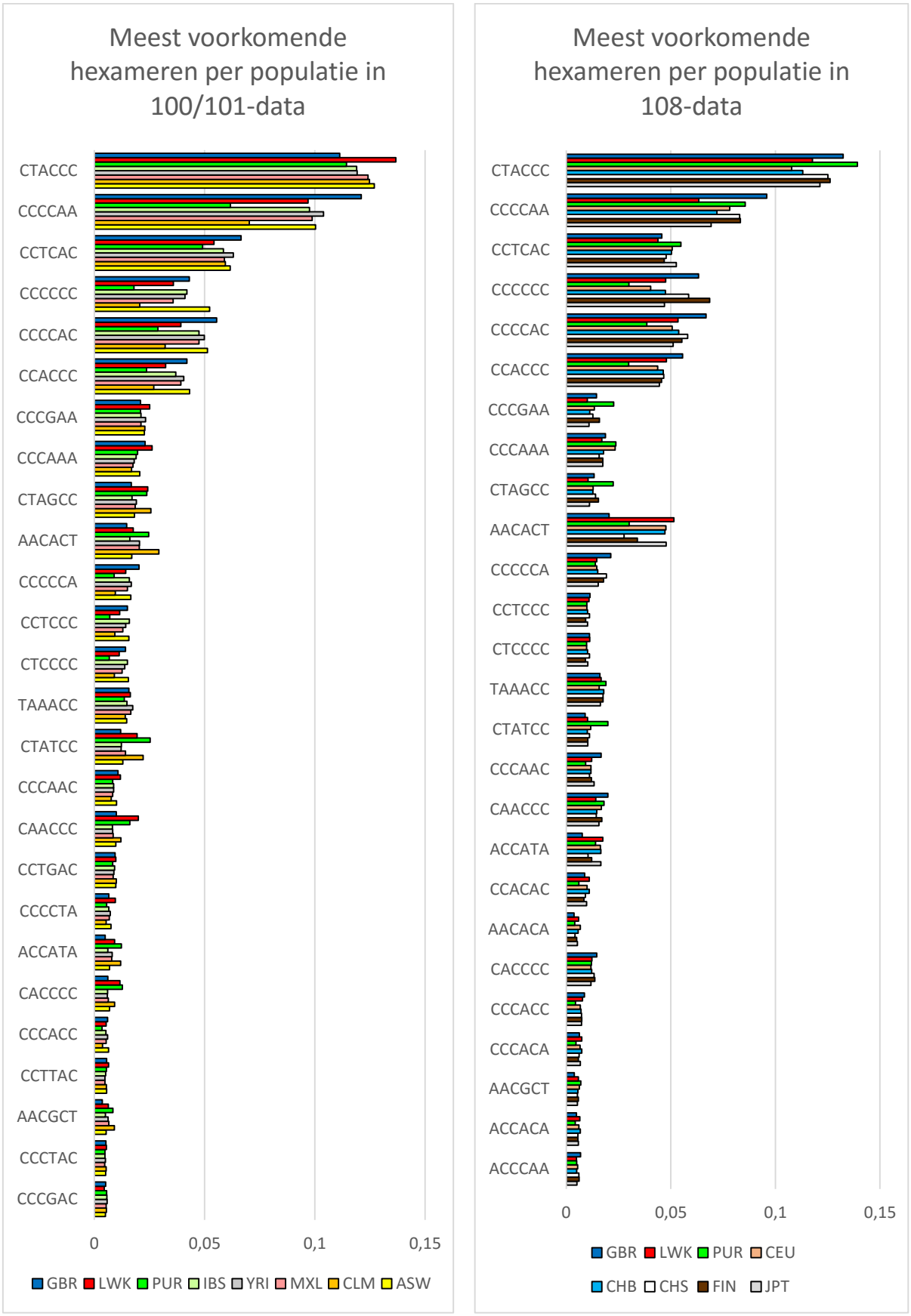
Figuur 41: Weergave van de voorkomende SNPs binnen de verschillende telomerische fracties en populaties.

3.3.5. Karakterisatie van hexameren volgens populatie

Net als in 3.2.5 worden de hexameren gekarakteriseerd, maar nu met het oog op de telomerische fractie waarin ze gedetecteerd zijn en de populatie waarvan de sequentiedata afkomstig is. Op Figuur 43 en Bijlage 3: Overzicht van hexameren in puurtelomerische, pseudotelomerisch en hybridetelomerische fractie zijn de hexameren te zien die met de hoogste frequentie voorkomen in elke telomerische fractie. Op deze figuren zijn er opnieuw grote verschillen tussen de 100/101-data en 108-data, ook wat betreft de gemeenschappelijke populaties GBR, LWK en PUR. Er kan gezien worden dat het CCCCAA-hexameer in beide datasets het meeste voorkomt bij de GBR groep. Dit valt in lijn met Figuur 41 waarin er binnen de GBR-groep meer T>C nucleotidesubstituties werden waargenomen. Het hexameer CCTCAC (en in mindere mate CCTCCC) dat een restrictiesite bevat voor het enzym *Mn*II is in elke telomerische fractie terug te vinden en varieert in beperkte (vermoedelijk niet biologisch relevante) mate tussen populaties. Omdat de hexameren binnen de pure telomeren en pseudotelomeren voor het grootste deel slechts uit een enkele nucleotidesubstitutie bestaan in vergelijking met het standaard CCCTAA-hexameer wordt dit overzichtelijker weergegeven in Figuur 42.



Figuur 42: Globaal overzicht van de vaakst voorkomende nucleotidesubstituties binnen het hexameer. Elk percentage is de frequentie waarbij het nucleotide in het hexameer gesubstitueerd wordt voor een ander nucleotide. Dit is te zien voor de pure telomeren met sequencing-error (links) en voor de pseudotelomeren (rechts).

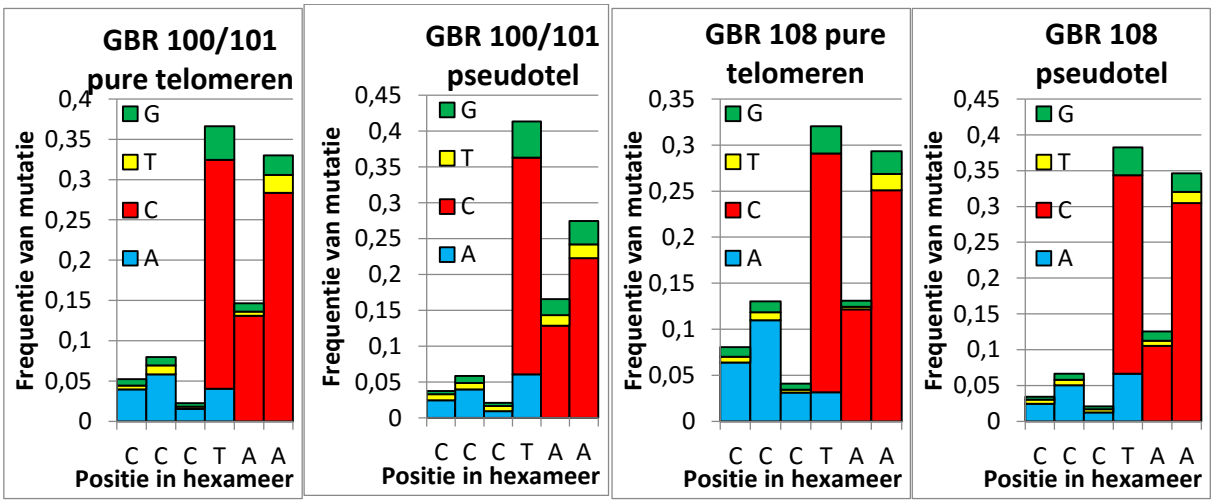


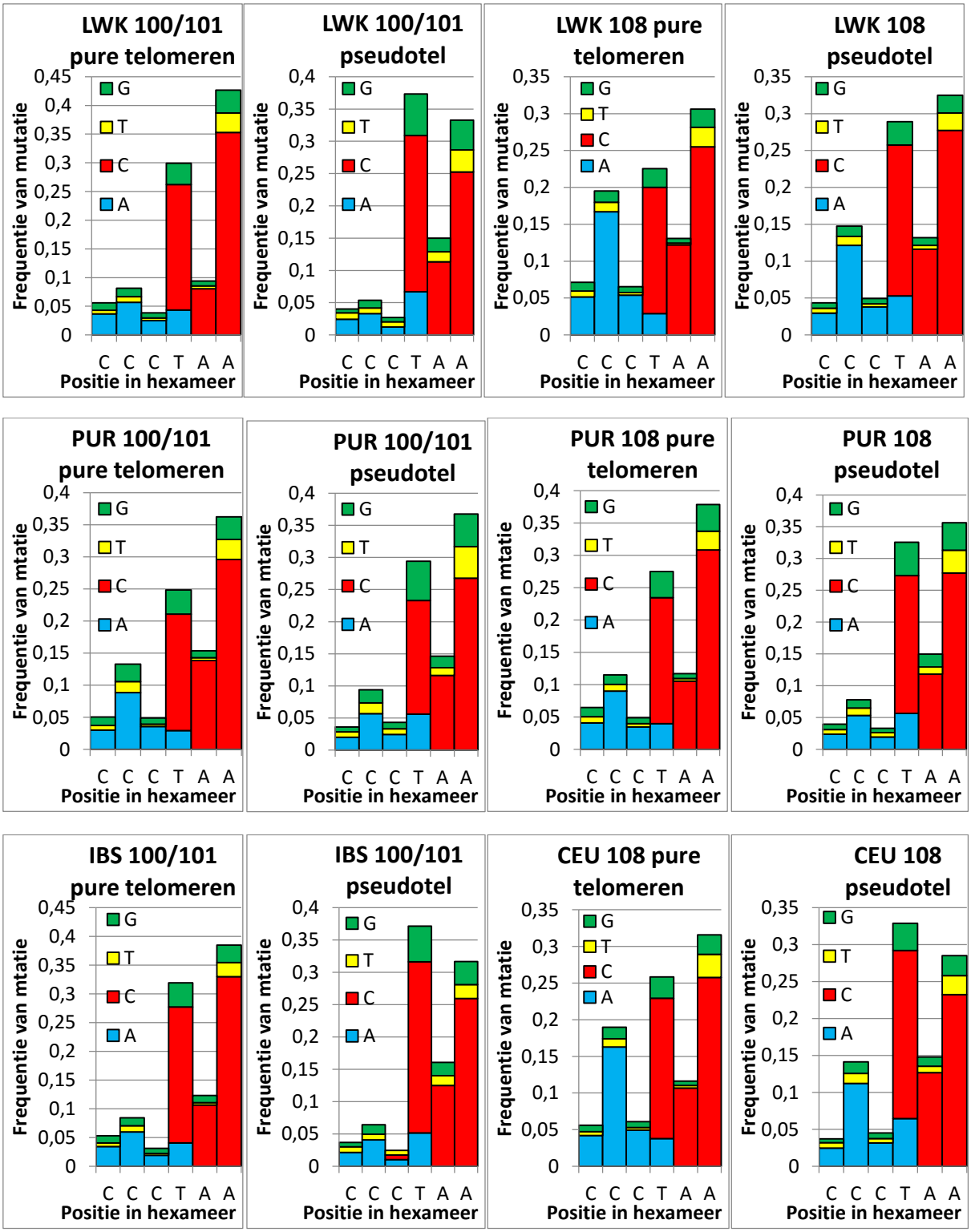
Figuur 43: Overzicht van de vaakst voorkomende hexameren in de 100/101-data (links) en 108-data (rechts).

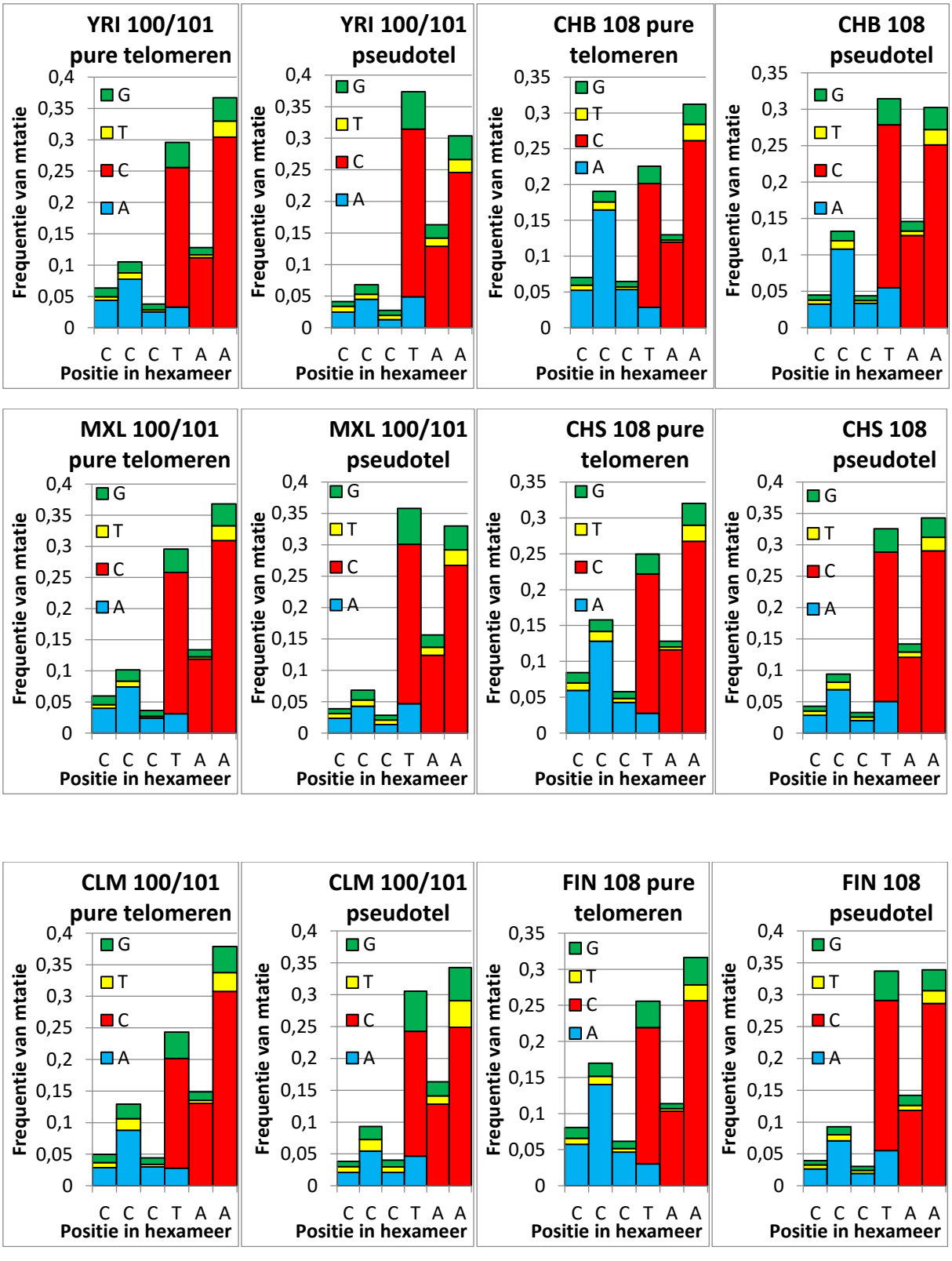
In Figuur 42 is te zien hoe bepaalde nucleotiden meer vatbaar zijn voor een nucleotidesubstitutie dan andere nucleotiden. Zo komt de SNP A>C van het tweede adenine in relatieve frequentie ongeveer twee maal zoveel voor als het eerste adenine. In de pseudotelomeren is te zien hoe de mutaties die voorkomen van dezelfde aard zijn als die binnen de pure telomeren met *sequencing error*. Dit suggereert opnieuw dat er bij het kiezen van een *cut-off* in 3.2.7 een meer stringente optie mocht gekozen worden (met nog kortere pure telomeren als gevolg).

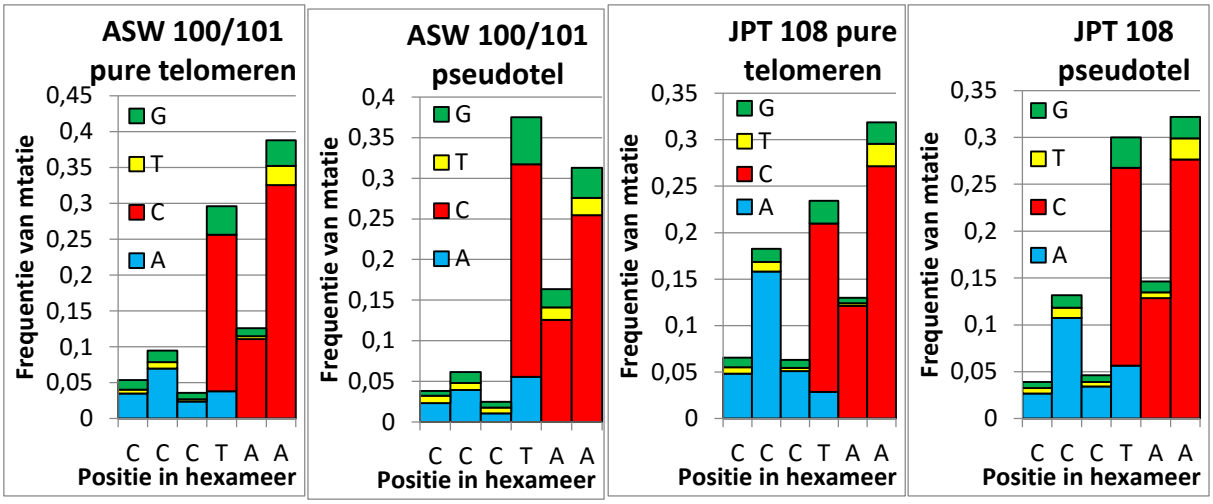
In de vergelijking tussen de 100/101-data en 108-data valt het op hoe het tweede cytosine in het hexameer twee maal meer muteert naar een adenine bij de 108-data. In een studie waarin de *sequencing error* in Illumina-platformen onderzocht werd was dit tevens de meest voorkomende fout. Aangezien langere *reads* vaak meer *sequencing errors* bevatten op het einde is het zeker mogelijk dat deze fout preferentieel in de laatste acht nucleotiden van de 108bp data voorkomt. Ook de mutatie A>C kwam in dit onderzoek vaak voor, zoals hier ook duidelijk is (Dohm, et al., 2008; Sleep, et al., 2013). Omwille van deze redenen kan niet worden uitgesloten dat de verschillen die in Figuur 42 worden gezien van technische aard zijn.

Echter, het is minder waarschijnlijk dat het grote verschil waarbij beide adenines gesubstitueerd worden voor een ander nucleotide een gevolg is van enkel een *sequencing error*. Daarbij verklaarde Sleep et al. (2013) dat de *sequencing error* T>A drie maal minder voorkomt dan de *sequencing error* C>A, iets wat hier helemaal niet het geval is. Om die reden is er een vermoeden dat de geobserveerde mutaties wel biologisch van aard zijn, maar dat de pure telomeren nog veel te weinig strict werden geïdentificeerd. Voor alle populaties binnen de dataset werden analoge figuren opgesteld (Figuur 44).




LWK 100/101 pure telomeren
LWK 100/101 pseudotel
LWK 108 pure telomeren
LWK 108 pseudotel
PUR 100/101 pure telomeren
PUR 100/101 pseudotel
PUR 108 pure telomeren
PUR 108 pseudotel
IBS 100/101 pure telomeren
IBS 100/101 pseudotel
CEU 108 pure telomeren
CEU 108 pseudotel
Frequentie van mutatie
Frequentie van mtatie
G
T
C
A
C C C T A A
Positie in hexameer


YRI 100/101 pure telomeren
YRI 100/101 pseudotel
CHB 108 pure telomeren
CHB 108 pseudotel
MXL 100/101 pure telomeren
MXL 100/101 pseudotel
CHS 108 pure telomeren
CHS 108 pseudotel
CLM 100/101 pure telomeren
CLM 100/101 pseudotel
FIN 108 pure telomeren
FIN 108 pseudotel
Frequentie van mtatie
Positie in hexameer
C C C T A A
G
T
C
A



Figuur 44: Binnen het telomerische hexameer werd elke nucleotidesubstitutie weergegeven. Elk percentage is de relatieve frequentie waarbij het nucleotide in het hexameer gesubstitueerd wordt voor een ander nucleotide. Dit werd gedaan voor alle populaties, opgesplitst tussen de 100/101-data, 108-data, pure telomeer (links) en pseudotelomeer (rechts).

Door het aannemen van een *sequencing error rate* binnen de pure telomeren wordt er verwacht dat het aantal *sequencing errors* binnen deze pure telomeren ook hoger zou moeten liggen dan deze bij de pseudotelomeren. Eerdere vondsten (zie 3.3.3: Karakterisatie van SNPs) duiden echter aan dat de pure telomeren niet stringent genoeg werden gedefinieerd. Om deze reden zijn de ware nucleotidesubstituties moeilijker te zien op Figuur 44. Om statistisch aan te tonen dat de pseudotelomeren meer mutaties vertonen wordt voor elk individu een standaarddeviatie berekend. Dit wordt gedaan op basis van de zes relatieve frequenties waarbij een nucleotide binnen het hexameer muteert naar een ander nucleotide. Het verschil in standaarddeviatie tussen de pseudotelomeren en pure telomeren geeft aanleiding tot het gebruik van een binomiale test. Hiermee kan er worden getest indien er een verschil is tussen beide telomerische fracties waarbij de nulhypothese aangeeft dat er geen verschil is. Er werden 212 grotere waarden waargenomen (waarbij de standaarddeviatie van de pseudotelomeren groter was dan die van de pure telomeren) en 94 negatieve waarden. De binomiale test geeft aan dat er wel degelijk een significant verschil is tussen beide fracties ($p=1.264*10^{-11}$). Dit toont aan dat er meer *sequencing errors* bij de echte telomeren worden waargenomen (die meer *random* zijn, dus lagere standaarddeviatie), en meer ware nucleotidesubstituties bij de pseudotelomeren (die meer verschillen tussen de basen, dus hogere standaarddeviatie).

Zoals eerder beschreven in 1.1 werden eerder al TTGGGG en TGAGGG *repeats* gedetecteerd in de telomerische sequentie. Deze hexameren worden ook hier teruggevonden in elke telomerische fractie (zie Bijlage 3: Overzicht van hexameren in puurtelomerische, pseudotelomerisch en hybridetelomerische fractie) onder de vorm van CCCCAA en CCCTCA (op de figuur CCTCAC). De mutatie die in het hexameer CCCCAA voorkomt is een T>C SNP en komt inderdaad meer voor bij de pseudotelomeren (die bestaan uit een *sequencing error* en mutaties) dan bij de pure telomeren (Figuur 42). Er is dus een sterke aanwijzing dat deze SNP inderdaad biologisch relevant is. Het hexameer CCCTCA wordt gevormd door de A>C SNP op de plaats van de eerste adenine. Hoewel hiervoor geen duidelijk verschil te zien is in voorkomen tussen pseudo en pure telomeren (Figuur 42) en dit bovendien de meest voorkomende *sequencing error* is in Illumina data, geeft dit aan dat ook deze SNP biologisch relevant kan zijn.

4. Discussie

Telomeren, herhalingen van een specifiek hexameer op de uiteindes van chromosomen, ondergaan gedurende het leven van een individu een graduele verkorting door celdelingen en oxidatieve stress. Dit gaat door tot een kritisch korte lengte wordt bereikt en het telomeer disfunctioneel wordt. De D-loop-T-loop die aanwezig is bij functionele telomeren kan bij deze kritische lengte niet langer gevormd worden en geeft zo aanleiding tot replicatieve senescentie (zie 1.1: Inleiding). Telomeerlengteverkorting werd reeds in verband gebracht met veroudering en er wordt geopperd dat ze ook als predictieve biomerker voor hart- en vaatziekten zou kunnen dienen (Cawthon, et al., 2003; De Meyer, et al., 2011). Ondanks sterke aanwijzingen voor causaliteit is de statistische associatie echter eerder beperkt. De verklaring die in deze masterscriptie wordt onderzocht is dat TRF *Southern blot*, de gouden standaard wat betreft telomeerlengtemetingen, de functionele telomeren meet maar ook gevoelig is voor de gedegenereerde, niet-functionele telomeersequenties (pseudotelomeren). Omdat de aanwezigheid van deze pseudotelomeren mogelijks varieert tussen individuen verlaagt de precisie van de methode en kan de TRF-methode onvoldoende specifiek worden genoemd (Baird, et al., 2005; Hunt, et al., 2008). Hierbij wordt ervan uitgegaan dat deze pseudotelomeren aanleiding kunnen geven tot moeilijkheden bij de vorming van de D-loop-T-loop wanneer de functionele telomeren kritisch verkort zijn. Dit wordt dan als een DNA-breuk herkend waardoor er replicatieve senescentie optreedt (Hemann, et al., 2001).

Recent werd er aangetoond dat de aanwezigheid van telomerische herhalingssequenties in WGS-data of exoom-data kan vertaald worden naar een geschatte gemiddelde telomeerlengte. Twee *tools* die dit realiseren, Telseq (Ding, et al., 2014) en Computel (Nersisyan & Arakelyan, 2015), zijn beide *open-source* en werden in deze masterthesis gebruikt om de niet-functionele gecapteerde telomerische fragmenten te karakteriseren. Dit werd gedaan voor een bepaalde set individuen binnen het Phase 1 deel van het 1000 Genomes Project. Ondanks het hier gaat om *low coverage* (4X) data, is dit geschikt voor exploratief onderzoek. Via een bio-informatica benadering werden gedetecteerde nucleotidesubstituties en indels gekarakteriseerd, en werden de telomeerlengtes van functionele telomeren en pseudotelomeren geschat. Vervolgens gebeurde er een vergelijking voor de verschillende populaties die aanwezig waren in de gekozen dataset.

In een eerste fase van de masterproef werd er bij de keuze van de dataset gezien dat veel stalen in het Phase 1 deel afweken van de 100bp *read* lengtes waarvoor zowel Computel en Telseq geoptimaliseerd werden (Figuur 14). Er werden uiteindelijk slechts 306 individuen gekozen van de totale 1092 individuen binnen het Phase 1 deel. Hierdoor werd er veel *power* verloren, maar was er een hoge waarschijnlijkheid dat de telomeerlengteschatting correct verliep. Omdat Telseq minder stringent is qua inclusie van subtelomerische regio's (Nersisyan & Arakelyan, 2015) werd ervoor gekozen om enkel met Computel verder te werken. Een vergelijking van determinanten bracht aan het licht dat de geschatte telomeerlengte erg afhankelijk was van de *read* lengte (zie 3.1.4.1: Read lengte). In tegenstelling tot de 100bp *reads* en 101bp *reads* die gelijkaardige telomeerlengtes voorstelden, was er een duidelijk significant verschil met de 108bp *reads* waardoor een verdere opsplitsing van de data noodzakelijk was (zie 3.1.5: Technische variatie). Het geslachtsverschil in telomeerlengtes kon statistisch niet worden aangetoond door een waarschijnlijk gebrek aan *power* van de test (zie 3.1.4.5: Geslacht).

In de tweede fase werd bij het opstellen van een telomerisch profiel duidelijk wat verantwoordelijk was voor het grote verschil in variaties en medianen tussen de 100/101bp-

data en 108bp-data. De 108bp-data bestaat over het algemeen over meer *reads* (Figuur 25), maar ook over een hoger aantal gedetecteerde nucleotidesubstituties in vergelijking met de 100/101bp-data (Figuur 23 en Figuur 24). Bij de vergelijking van de (veronderstelde) SNP/indel ratio werd er gezien dat er ook hier een groot verschil was tussen de 100/101bp-data en 108bp-data. Dit valt tevens te verklaren door de aanwezigheid van een hoog aantal *sequencing errors* binnen de 108bp-data. Er werden binnen de 108bp-data 25 extreme waarden gevonden waarvan 14 behorend tot de LWK (Luhya in Webuye, Kenya) populatie. Er was ook geen statistisch verschil aantoonbaar tussen het aantal *reads* met veronderstelde SNPs en indels en het aantal *reads* met veronderstelde SNP's zonder indels op een willekeurig gekozen individu (zie 3.2.3: Verhouding SNPs - indels). De indel TTTAGGG die eerder gerapporteerd werd (zie 1.1: Inleiding) werd ook in deze data teruggevonden, hetzij niet als meest voorkomende indel. Aangezien deze resultaten erop wezen dat SNP's veel meer abundant zijn dan indels werd ervoor gekozen om verder op SNP's te focussen.

Bij het karakteriseren van de voorkomende (veronderstelde) SNPs was te zien dat de vaakst voorkomende SNP's (A>C, T>C en C>A) in zowel de 100/101bp-data als de 108bp-data gelijk waren (Figuur 29). De A>C SNP is echter ook de meest voorkomende *sequencing error* in Illumina-sequeneringsmethodes (Dohm, et al., 2008). Dit was opnieuw een bevestiging dat het aantal *sequencing errors* niet te onderschatten is. Bij de alternatieve hexameren werden er twee hexameren ontdekt (CCTCAC en CCTCCC) die een restrictiesite bevatten voor het *Mn*II restrictie-enzym en samen voorkomen met een frequentie van ca. 6% van alle hexameren (Figuur 30). Ondanks het feit dat het restrictiepaar *Hph*I/*Mn*II minder vaak gebruik wordt dan het restrictiepaar *Hinf*I/*Rsa*I in TRF *Southern blot* methodes was dit wel een bevestiging dat deze hexameren een TRF-meting kunnen beïnvloeden. Omdat de lengtes van de gedegenereerde telomerische fracties tussen individuen verschillen in een mate die niet gekend is, kan dit aanleiding geven tot een verlaagde precisie van TRF-methodes.

In een poging om echte SNPs van *sequencing errors* te onderscheiden werd gepoogd om de telomerische *reads* verder op te splitsen. Hiervoor werd een arbitraire *sequencing error rate* van 1.5% gebruikt, waardoor met behulp van de Poisson-distributie de fractie *reads* met SNPs van deze met enkel *sequencing errors* te onderscheiden. Deze 1.5% werd gekozen naar de verwachting van *error rate* die optrad in de begintijden van *second generation sequencing* (en wanneer het 1000 Genomes Project plaatsvond) (zie 3.2.7: Sequencing error rate). In de derde fase van het onderzoek werden de gecapteerde telomerische *reads* dan ook in drie afzonderlijke telomerische fracties opgesplitst: de functionele telomeren die verondersteld wordt enkel *sequence errors* te bevatten, pseudotelomeren met naast *sequencing errors* ook mutaties, en de hybride telomeren die werden gedefinieerd op basis van de aanwezigheid van twee naburige mutaties. Bij de vergelijking van de verschillende populaties over de verschillende telomerische fracties heen was er indicatie dat de pseudotelomeren onderling meer verschil toonden dan de pure telomeren (zie 3.3.2). Door inconsistentie van de data (wellicht door technische variatie) was het echter niet mogelijk dit hard te maken. De vergelijking van lengtes tussen pure telomerische regio's en pseudotelomerische regio's toonde aan dat sommige individuen meer pseudotelomerische *reads* bevatten dan pure telomerische *reads* (Figuur 34), waar volgens de literatuur een omgekeerde trend is verwacht (Counter, et al., 1992; Greider & Harley, 1996). Bij de vergelijking in SNPs tussen de afzonderlijke telomerische fracties werden ook geen grote *shifts* waargenomen in type en frequentie van voorkomende SNPs (Figuur 38). Dit wees er waarschijnlijk op dat de definitie van "pure" telomeren (alle *reads* met minder dan vier voorkomende nucleotidesubstituties) onvoldoende streng was, en er bij voorkeur een meer stringente *cut-off* voor *sequencing errors* gekozen moest worden. De *sequencing error* speelde ook bij het karakteriseren van de hexameren een belangrijke rol sinds het verschil

tussen de *sequencing errors* en biologische nucleotidesubstituties niet eenvoudig te onderscheiden waren. Er werd via het gebruik van een binomiale test statistisch aangetoond dat de pseudotelomeren, zoals eerder vooropgesteld, naar alle waarschijnlijkheid meer mutaties bevatten dan de pure telomeren (zie 3.3.5). De alternatieve hexameren (TTGGGG en TGAGGG) die eerder via experimentele methodes ontdekt werden (zie 1.1: Inleiding), werden in alle telomerische fragmenten teruggevonden als respectievelijk tweede (CCCCAA) en derde (CCCTCA) meest frequent. Er wordt dus vermoed dat naast de voorkomende *sequencing errors* er duidelijk ook biologische relevante mutaties te zien zijn.

De mogelijkheid om uit WGS data een telomeerlengte te schatten bleek in deze masterscriptie vanuit een exploratief standpunt erg interessant. Van de twee tools die dit mogelijk maakten, Telseq en Computel, was Computel het meest relevant voor deze thesis gezien er stringenter werd omgegaan (maar nog steeds niet stringent genoeg) met de aanwezigheid van subtelomerische regio's (Nersisyan & Arakelyan, 2015). Het was duidelijk dat er grote technische variaties in de data aanwezig waren. In een eerste instantie was een opsplitsing tussen 100/101bp-data en 108bp-data noodzakelijk omdat er een hoge discrepantie werd waargenomen in varianties en medianen. Er moet opgemerkt worden dat de uniformiteit van de data erg belangrijk is omdat individuele *reads* als basis liggen voor het onderzoek. Het Phase 1 deel van het 1000 Genomes Project was om die reden minder geschikt gezien het voorkomen van verschillende *sequencing* methodologieën en *read* lengtes. Zelfs zonder externe validatie kunnen we dus concluderen dat de telomeerlengteschattingen door Computel, maar vermoedelijk ook Telseq, niet als absoluut dienen geïntepreteerd te worden en duidelijk onderhevig zijn aan technische variatie. De recent publiek gemaakte Phase 3 data zou voor dit onderzoek dan ook beter geweest zijn omdat er exclusief gewerkt wordt met Illumina-*sequencing* data, en de *reads* ook langer en meer uniform zijn. Daarnaast dwong de aanwezigheid van een *sequence error* er toe om de *reads* op te splitsen met de bedoeling mutaties van deze *errors* te onderscheiden. Uiteindelijk bleek dat de gekozen *cut-off* voor deze *sequence errors* niet stringent genoeg was en de verondersteld pure telomeren nog steeds een te groot aantal mutaties bevatten. Omdat er een hogere *sequencing error rate* wordt waargenomen bij de terminale nucleotiden van de *read* zou er eventueel gekeken kunnen worden of het *trimmen* van de terminale nucleotiden van de 108bp *reads* een aanzienlijke verbetering geeft. Daarnaast zou het gebruik van meer data positief zijn om de *power* van de testen te verbeteren.

De werkwijze die hier werd toegepast bestond uit het opsplitsen van de gecapteerde telomerische *reads* tot pure telomeren, pseudotelomeren en hybride telomeren waaruit tenslotte een lengte berekend werd. Dit is echter exploratief en kan niet als representatief worden beschouwd voor de ware lengte van de afzonderlijke fracties, vnl. door de keuze van een te hoge arbitraire *sequencing error rate*. Hierbij is het belangrijk op te merken dat dit zal leiden tot een (grote) verdere afname van de lengte aan pure telomeren, terwijl de subtelomeren (pseudo- en hybride telomeren samen) nu al vaak langer ingeschat worden dan de functionele telomeren. Hierbij werd gebruik gemaakt van Computel, een programma dat al conservatiever is dan Telseq, software die werd geoptimaliseerd om optimaal met telomeerlengte op basis van TRF overeen te komen. Gecombineerd met de observatie dat de telomerische mutaties naar alle waarschijnlijkheid vooral biologisch van herkomst zijn, geeft dit aan dat de werkelijk "pure" telomeren vermoedelijk veel korter zijn dan aanvankelijk gedacht. Indien de vermelde telomerische mutaties effectief D-loop-T-loop vorming verhinderen kan dit een grote biologische impact hebben. Bovendien kunnen de huidige meest gebruikte methodes (TRF, maar ook qPCR) dit nu niet meten, terwijl er vermoedelijk sterke interindividuele verschillen aanwezig zijn (Figuur 13). Rekening houdend met bovenstaande opmerkingen moet het dan ook mogelijk worden om op termijn te gaan kijken of dit de onderliggende reden is voor de grote discrepantie tussen de veronderstelde causale

link tussen cardiovasculaire ziekten en telomeerlengte en de momenteel beperkte biomerkerwaarde.

Hoewel een beter design van de studie (zie hoger) deze analyses sterk zou kunnen verbeteren, zijn er ook andere opties mogelijk. Eerst en vooral is er de aanwezigheid van trio's (man,vrouw,kind) binnen de nieuwste data van het 1000 Genomes Project. Een studie van de mate van overerfbaarheid van de telomerische mutaties moet toelaten in te schatten in welke mate de vroeger geobserveerde verschillen effectief biologisch van aard zijn. Een betere techniek om telomeren en andere *high repetitive* regio's te karakteriseren is echter gebruik te maken van de recente *single molecule sequencing* technologie (Chaisson, et al., 2014). Dit zou het mogelijk maken om de overgang tussen hybride telomeren, pseudotelomeren en pure telomeren beter te beschrijven, een algemeen betere annotatie te voorzien van de telomerische regio's maar ook een absolute chromosoomspecifieke telomeerlengte te berekenen. Door de nog steeds aanzienlijke *error rate* en afwezigheid van grote hoeveelheden publiek beschikbare data afkomstig van deze *3rd generation sequencing* was het in deze masterscriptie echter niet mogelijk om dit toe te passen. Binnen afzienbare tijd zou het echter wel moeten mogelijk worden om in te schatten of telomeren nu effectief een goede biomerker voor (cardiovasculaire) veroudering zijn.

5. Bibliografie

1000 Genomes. (2012) *1000 Genomes.* www.1000genomes.org
URL bezocht op 6 november 2015

Allshire, R. C., Dempster, M. & Hastie, N. D., 1989. Human telomeres contain at least three types of G-rich repeat distributed non-randomly. *Nucleic Acids Research,* 17(12), pp. 4611-4627.

Armanios, M. et al., 2005. Haploinsufficiency of telomerase reverse transcriptase leads to anticipation in autosomal dominant dyskeratosis congenita. *Proc Natl Acad Sci,* 102(44), pp. 15960-15964.

Aviv, A. et al., 2011. Impartial comparative analysis of measurement of leukocyte telomere length/DNA content by Southern blots and qPCR. *Nucleic Acids Research,* 39(20), p. 5.

Azzalin, C. M. et al., 2007. Telomeric repeat containing RNA and RNA surveillance factors at mammalian chromosome ends. *Science,* 318(5851), pp. 798-801.

Baerlocher, G. M., Vulto, I., de Jong, G. & Lansdorp, P. M., 2006. Flow cytometry and FISH to measure the average length of telomeres (flow FISH). *Nature protocols,* 1(5), pp. 2365-2376.

Baird, D. M. et al., 2005. Telomere instability in the male germline. *Human Molecular Genetics,* 15(1), pp. 45-51.

Baird, D. M., Rowson, J., Wynford-Thomas, D. & Kipling, D., 2003. Extensive allelic variation and ultrashort telomeres in senescent human cells. *Nature Genetics,* 33(2), pp. 203-207.

Baur, J. A., Zou, Y., Shay, J. W. & Wright, W. E., 2001. Telomere position effect in human cells.. *Science,* 292(5524), pp. 2075-2077.

Bekaert, S. et al., 2007. Telomere length and cardiovascular risk factors in a middle-aged population free of overt cardiovascular disease. *Aging Cell,* 6(5), pp. 639-647.

Bekaert, S., De Meyer, T. & Van Oostveldt, P., 2005. Telomere attrition as ageing biomarker.. *Anticancer Res,* 25(4), pp. 3011-3021.

Bendix, L. et al., 2010. The load of short telomeres, estimated by a new method, Universal STELA, correlates with number of senescent cells. *Aging Cell,* 9(3), pp. 383-397.

Benetos, A. et al., 2013. Tracking and fixed ranking of leukocyte telomere length across the adult life course. *Aging Cell,* 12(4), pp. 615-621.

Benetos, A. et al., 2001. Telomere length as an indicator of biological aging: the gender effect and relation with pulse pressure and pulse wave velocity. *Hypertension,* 37(2), pp. 381-385.

Broccoli, D., Smogorzewska, A. & de Lange, T., 1997. Human telomeres contain two distinct Myb-related proteins, TRF1 and TRF2. *Nat Genet.,* 17(2), p. 5.

Brouilette, S. W. et al., 2008. Telomere length is shorter in healthy offspring of subjects with coronary artery disease: support for the telomere hypothesis.. *Heart,* 94(4), pp. 422-425.

Bryan, T. M. & Cech, T. R., 1999. Telomerase and the maintenance of chromosome ends. *Current Opinion in Cell Biology,* 11(3), pp. 318-324.

Cai, Z., Yan, L. J. & Ratka, A., 2013. Telomere shortening and Alzheimer's disease.. *Neuromolecular Med,* 15(1), pp. 25-48.

Callaghan, N. J. & Fenech, M., 2011. A quantitative PCR method for measuring absolute telomere length. *Biological Procedures Online,* 13(3), p. 10.

Castle, C. J. et al., 2010. DNA copy number, including telomeres and mitochondria, assayed using next-generation sequencing. *BMC Genomics,* Volume 11, p. 244.

Cawthon, R. M., 2002. Telomere measurement by quantitative PCR. *Nucleic Acids Research,* 30(10), p. 6.

Cawthon, R. M., 2009. Telomere length measurement by a novel monochrome multiplex quantitative PCR method. *Nucleic Acids Research,* 37(3), p. 7.

Cawthon, R. M. et al., 2003. Association between telomere length in blood and mortality in people aged 60 years or older. *Lancet,* 361(9355), pp. 393-395.

Cech, T., 2004. Beginning to understand the end of the chromosome. *Cell,* 116(2), pp. 273-279

Chaisson, M. J. et al., 2014. Resolving the complexity of the human genome using single-molecule sequencing. *Nature,* 517(7536), pp. 608-611.

Codd, V. et al., 2013. Identification of seven loci affecting mean telomere length and their association with disease. *Nat Genet,* 45(4), pp. 422-427.

Collins, K., 2006. The biogenesis and regulation of telomerase holoenzymes. *Nat Rev Mol,* 7(7), pp. 484-494.

Corcoran, L. M., Thompson, J. K., Walliker, D. & Kemp, D. J., 1988. Homologous recombination within subtelomeric repeat sequences generates chromosome size polymorphisms in P. falciparum. *Cell,* 53(5), pp. 807-813.

Counter, C. M. et al., 1992. Telomere shortening associated with chromosome instability is arrested in immortal cells which express telomerase activity. *EMBO Journal,* 11(5), pp. 1921-1929.

Court, R., Chapman, L., Fairall, L. & Rhodes, D., 2005. How the human telomeric proteins TRF1 and TRF2 recognize telomeric DNA: a view from high-resolution crystal structures.. *EMBO,* 6(1), p. 6.

Cristofari, G., Sikora, K. & Lingner, J., 2007. Telomerase unplugged. *ACS Chem Biol,* 2(3), pp. 155-158.

De Meyer, T. et al., 2007. Paternal age at birth is an important determinant of offspring telomere length. *Human Molecular Genetics,* 16(24), pp. 3097-3102.

De Meyer, T. et al., 2009. Systemic telomere length and preclinical atherosclerosis: the Asklepios Study. *Eur Heart J,* 30(24), pp. 3074-3081.

De Meyer, T. et al., 2011. Telomere length and cardiovascular aging: The means to the ends?. *Ageing Research Reviews,* 10(2), pp. 297-303.

De Meyer, T. et al., 2014. A non-genetic, epigenetic-like mechanism of telomere length inheritance?. *Eur J Hum Genet,* 22(1), pp. 10-11.

Ding, Z. et al., 2014. Estimating telomere length from whole genome sequence data. *Nucleic Acids Research,* 42(9), p. 4.

Diotti, R. & Loayza, D., 2011. Shelterin complex and associated factors at human telomeres. *Nucleus,* 2(2), pp. 119-135.

Dohm, C. J., Lottaz, C., Borodina, T. & Himmelbauer, H., 2008. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research,* 36(16).

Factor-Litvak, P. et al., 2016. Leukocyte Telomere Length in Newborns: Implications for the Role of Telomeres in Human Disease. *Pediatrics,* 137(4), p. e20153927.

Fairall, L. et al., 2001. Structure of the TRFH dimerization domain of the human telomeric proteins TRF1 and TRF2. *Mol Cell.,* 8(2), p. 10.

Farzaneh-Far, R. et al., 2010. Telomere length trajectory and its determinants in persons with coronary artery disease: longitudinal findings from the heart and soul study. *PLoS One,* 5(1), p. e8612.

Ferragina, P. & Manzini, G., 2005. Indexing Compressed Text. *JACM,* 52(4), pp. 552-581.

Flint, J. et al., 1997. Sequence comparison of human and yeast telomeres identifies structurally distinct subtelomeric domains. *Hum Mol Genet.,* 6(8), pp. 1305-1313.

Garcia, C. K., Wright, W. E. & Shay, J. W., 2007. Human diseases of telomerase dysfunction: insights into tissue aging. *Nucleic Acids Res,* 35(22), pp. 7406-7416.

Gardner, M. et al., 2014. Gender and telomere length: Systematic review and meta-analysis. *Experimental Gerontology,* Volume 51, pp. 15-27.

Gheysen, L., Van Damme, E. & Kyndt, T., 2014. *Gentechnologie en moleculaire diagnostiek.* Gent: Faculteit Bio-Ingenieurswetenschappen.

Graakjaer, J. et al., 2003. The pattern of chromosome-specific variations in telomere length in humans is determined by inherited, telomere-near factors and is maintained throughout life. *Mechanisms of Ageing and Development,* 124(5), pp. 629-640.

Greider, 1999. Telomeres Do D-loop-T-loop. *Cell,* 97(4), pp. 419-422.

Greider, C. W., 1991. Telomerase is processive. *Mol Cell Biol,* 11(9), pp. 5683-4580.

Greider, C. W. & Blackburn, E. H., 1989. A telomeric sequence in the RNA of Tetrahymena telomerase required for telomere repeat synthesis. *Cell,* 337(6205), pp. 331-337.

Greider, C. W. & Harley, C. B., 1996. Telomere length regulation in mammalian cells. In: N. J. Holbrook, G. R. Martin & R. A. Lockshin, eds. *Cellular Aging and Cell Death.* New York: Wiley-Liss, p. 131.

Griffith, J. D. et al., 1999. Mammalian Telomeres End in a Large Duplex Loop. *Cell,* 97(4), p. 11.

Harada, N. et al., 2004. A 1-Mb critical region in six patients with 9q34.3 terminal deletion syndrome. *J Hum Genet,* 49(8), pp. 440-444.

Hemann, M. T., Strong, M. A., Hao, L.-Y. & Greider, C. W., 2001. The Shortest Telomere, Not Average Telomere Length, Is Critical for Cell Viability and Chromosome Stability. *Cell,* 107(1), pp. 67-77.

Hjelmborg, J. B. et al., 2015. The heritability of leucocyte telomere length dynamics. *J Med Genet,* 52(5), pp. 297-302.

Hockemeyer, D. & Collins, K., 2015. Control of telomerase action at human telomeres. *Nature Struct Mol Biol,* 22(11), pp. 848-852.

Hunt, S. C. et al., 2008. Leukocyte telomeres are longer in African Americans than in whites: the National Heart, Lung, and Blood Institute Family Heart Study and the Bogalusa Heart Study. *Aging Cell,* 7(4), pp. 451-458.

Illumina. (2016) *Illumina DNA Sequencing.* http://www.illumina.com/
URL bezocht op 3 april 2016.

Janouskova, E. et al., 2015. Human Rap1 modulates TRF2 attraction to telomeric DNA. *Nucleic Acids Res,* 43(5), pp. 2691-2700.

Jeanclos, E. et al., 2000. Telomere length inversely correlates with pulse pressure and is highly familial. *Hypertension,* 36(2), pp. 195-200.

Kalmbach, K. H. et al., 2013. Telomeres and human reproduction. *Fertil Steril,* 99(1), pp. 23-29.

Karlseder, J. et al., 1999. p53- and ATM-dependent apoptosis induced by telomeres lacking TRF2. *Science,* 283(5406), pp. 1321-1325.

Kimura, M. et al., 2010. Measurement of telomere length by the Southern blot analysis of terminal restriction fragment lengths. *Nature protocols,* 5(9), p. 11.

Kim, W. et al., 2012. Single-variant and multi-variant trend tests for genetic association with next-generation sequencing that are robust to sequencing error. *Hum Hered,* 74(3-4), pp. 172-183.

Kircher, M. & Kelso, J., 2010. High-throughput DNA sequencing--concepts and limitations.. *Bioessays,* 32(6), pp. 524-536.

Langmead, B. (2013) *Introduction to the Burrows-Wheeler Transform and FM Index.* http://www.cs.jhu.edu/~langmea/resources/bwt_fm.pdf
URL bezocht op 10 februari 2016.

Langmead, B. & Salzberg, S., 2012. Fast gapped-read alignment with Bowtie 2. *Nature methods,* 9(4), pp. 357-359.

Latrick, C. M. & Cech, T. R., 2010. POT1-TPP1 enhances telomerase processivity by slowing primer dissociation and aiding translocation. *EMBO,* 29(5), pp. 924-933.

Lin, J. et al., 2004. A universal telomerase RNA core structure includes structured motifs required for binding the telomerase reverse transcriptase protein. *Proc Natl Acad Sci,* 102(36), pp. 14713-14718.

Lin, K.-W. & Yan, J., 2007. The telomere length dynamic and methods of its assessment. *Journal of Cellular and Molecular Medicine,* 9(4), pp. 977-989.

Loman, N. J. et al., 2012. Performance comparison of benchtop high-throughput sequencing platforms. *Nature Biotechnology,* 30(5), pp. 434-439.

Lue, N. F., Lin, Y. C. & Mian, I. S., 2003. A conserved telomerase motif within the catalytic domain of telomerase reverse transcriptase is specifically required for repeat addition processivity.. *Mol Cell Biol,* 23(23), pp. 8440-8449.

Luo, C. et al., 2012. Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample. *PLoS One,* 7(2), p. e30087.

Marinier, E., Brown, D. G. & McConkey, B. J., 2015. Pollux: platform independent error correction of single and mixed genomes. *BMC Bioinformatics,* 16(10).

Mason, J. M. & Biessmann, H., 1995. The unusual telomeres of Drosophila.. *Trends Genet,* 11(2), pp. 58-62.

Mather, K. A., Jorm, A. F., Parslow, R. A. & Christensen, H., 2011. Is telomere length a biomarker of aging? A review. *J Gerontol A Biol Sci Med Sci,* 66(2), pp. 202-213.

Mathon, N. F. & Lloyd, A. C., 2001. Milestones in cell division : Cell senescence and cancer. *Nature Reviews Cancer,* Volume 1, pp. 203-213.

Mattern, K. A. et al., 2004. Dynamics of Protein Binding to Telomeres in Living Cells: Implications for Telomere Structure and Function. *Mol Cell Biol,* 24(12), pp. 5587-5594.

Mefford, H. C. & Trask, B. J., 2002. The complex structure and dynamic evolution of human subtelomeres. *Nature,* 3(2), pp. 91-102.

Metzker, M. L., 2010. Sequencing technologies - the next generation. *Nature Rev Genet,* 11(1), pp. 31-46.

Minamino, T. et al., 2002. Endothelial cell senescence in human atherosclerosis: role of telomere in endothelial dysfunction. *Circulation,* 105(13), pp. 1541-1544.

Minoche, A. E., Dohm, J. C. & Himmelbauer, H., 2011. Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and Genome Analyzer systems. *Genome Biol,* 12(11).

Mitchell, J. R., Wood, E. & Collins, K., 1999. A telomerase component is defective in the human disease dyskeratosis congenita. *Nature,* 402(6761), pp. 551-555.

Montpetit, A. J. et al., 2014. Telomere Length: A Review of Methods for Measurement. *Nursing Research,* 63(4), p. 10.

Montpetit, A. J. et al., 2014. Telomere Length: A Review of Methods for Measurement. *Nurs Res,* 63(4), pp. 289-299.

Morin, G. B., 1989. The human telomere terminal transferase enzyme is a ribonucleoprotein that synthesizes TTAGGG repeats. *Cell,* 59(3), pp. 521-529.

Moyzis, R. K. et al., 1988. A highly conserved repetitive DNA sequence, (TTAGGG)n, present at the telomeres of human chromosomes. *Proc. Natl. Acad. Sci.,* 85(18), pp. 6622-6626.

Nakamura, K. et al., 2011. Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Research,* 39(13), p. e90.

Nawrot , T. S., Staessen, J. A., Gardner, J. P. & Aviv, A., 2004. Telomere length and possible link to X chromosome. *Lancet,* 363(9408), pp. 507-510.

Neidle, S. & Parkinson, G. N., 2003. The structure of telomeric DNA. *Current Opinion in Structural Biology,* 13(3), pp. 275-283.

Nersisyan, L. & Arakelyan, A., 2015. Computel: Computation of Mean Telomere Length from Whole-Genome Next-Generation Sequencing Data. *PLOS one,* 10(4), p. 14.

Nilsson, P. M., Tufvesson, H., Leosdottir, M. & Melander, O., 2013. Telomeres and cardiovascular disease risk: an update 2013. *Translational Research,* 162(6), pp. 371-380.

Nishikawa, T. et al., 2001. Solution Structure of a Telomeric DNA Complex of Human TRF1. *Structure,* 9(12), pp. 1237-1251.

Ornish, D. et al., 2008. Increased telomerase activity and comprehensive lifestyle changes: a pilot study. *Lancet Ocol,* 9(11), pp. 1048-1057.

Parry, E. M. et al., 2011. Syndrome complex of bone marrow failure and pulmonary fibrosis predicts germline defects in telomerase. *Blood,* 117(21), pp. 5607-5611.

Petersen, S., Saretzki, G. & von Zglinicki, T., 1998. Preferential accumulation of single-stranded regions in telomeres of human fibroblasts. *Exp Cell Res,* 239(1), pp. 152-160.

Phelan, M. C. et al., 2001. 22q13 deletion syndrome. *Am J Med Genet,* 101(2), pp. 91-99.

Podlevsky, J. D. & Chen, J. J.-L., 2012. It all comes together at the ends: Telomerase structure, function, and biogenesis. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis,* 730(1-2), pp. 3-11.

Poon, S. S. & Lansdorp, P. M., 2001. Quantitative fluorescence in situ hybridization (Q-FISH). *Curr Protoc Cell Biol,* pp. 18.4.1-18.4.21.

Prescott, J. et al., 2012. Paternal age at birth is associated with offspring leukocyte telomere length in the nurses' health study. *Hum Reprod,* 27(12), pp. 3622-3631.

Qi, X. et al., 2011. RNA/DNA hybrid binding affinity determines telomerase template-translocation efficiency. *EMBO,* 31(1), pp. 150-161.

Reig-Viader, R. et al., 2013. Telomeric repeat-containing RNA and telomerase in human fetal oocytes.. *Hum Reprod,* 28(2), pp. 414-422.

Riethman, H. C. et al., 2001. Integration of telomere sequences with the draft human genome sequence. *Nature,* Volume 409, pp. 948-951.

Robin, J. D. et al., 2014. Telomere position effect: regulation of gene expression with progressive telomere shortening over long distances. *Genes & Development,* 28(22), pp. 2464-2476.

Rudd, M. K., 2012. Structural variation in subtelomeres. *Methods Mol Biol,* Volume 838, pp. 137-149.

Savage, S. A. & Bertuch, A. A., 2010. The genetics and clinical manifestations of telomere biology disorders. *Genetics in Medicine,* 12(12), pp. 753-764.

Sfeir, A. & de Lange, T., 2012. Removal of shelterin reveals the telomere end-protection problem. *Science,* Volume 336, p. 6.

Shay, J. W. & Woodring, W. E., 2010. Telomeres and telomerase in normal and cancer stem cells. *FEBS Lett,* 584(17), pp. 3819-3825.

Shay, J. W. & Wright, W. E., 2010. Telomeres and telomerase in normal and cancer stem cells. *FEBS Lett,* 584(17), pp. 3819-3825.

Shefer, K. et al., 2007. A triple helix within a pseudoknot is a conserved and essential element of telomerase RNA. *Mol Cell Biol,* 27(6), pp. 2130-2143.

Slagboom, P. E., Droog, S. & Boomsma, D. I., 1994. Genetic determination of telomere size in humans: a twin study of three age groups. *Am J Hum Genet,* 55(5), pp. 876-882.

Sleep, J. A., Shreiber, A. W. & Baumann, U., 2013. Sequencing error correction without a reference genome. *BMC Bioinformatics,* 14(367).

Smith, S., Giriat, I., Schmitt, A. & de Lange, T., 1998. Tankyrase, a poly(ADP-ribose) polymerase at human telomeres. *Science,* 282(5393), pp. 1484-1487.

Smogorzewska, A. & de Lange, T., 2004. Regulation of telomerase by telomeric proteins.. *Annu Rev Biochem,* 73(10), p. 11.

Sohal, R. S. & Dubey, A., 1994. Mitochondrial oxidative damage, hydrogen peroxide release, and aging. *Free Radical Biology and Medicine,* 16(5), pp. 621-626.

Sprott, R. L., 2010. Biomarkers of aging and disease: introduction and definitions. *Exp Gerontol,* 45(1), pp. 2-4.

Stansel, R. M., de Lange, T. & Griffith, T. D., 2001. T-loop assembly in vitro involves binding of TRF2 near the 3′ telomeric overhang. *The EMBO Journal,* 20(19), pp. 5532-5540.

Surekha, C. & Rao, N., 2016. Next-Generation Sequencing and Metagenomics. In: K. Wong, ed. *Computational Biology and Bioinformatics: Gene Regulation.* Hong Kong: CRC Press, pp. 334-335.

Takubo, K. et al., 2002. Telomere lengths are characteristic in each human individual.. *Exp Gerontol,* 37(4), pp. 523-531.

Testa, R. et al., 2011. Leukocyte telomere length is associated with complications of type 2 diabetes mellitus. *Diabet Med,* 28(11), pp. 1388-1394.

The 1000 Genomes Project Consortium, 2010. A map of human genome variation from population-scale sequencing. *Nature,* 467(7319), p. 12.

The SAM/BAM Format Specification Working Group. (2015) *Sequence Alignment/Map Format Specification.*
https://samtools.github.io/hts-specs/SAMv1.pdf
URL bezocht op 18 mei 2016.

Tollefsbol, T. O., 2007. *Biological Aging - Methods and Protocols.* Totowa, New Jersey: Humana Press.

Trask, B. J. et al., 1998. Members of the olfactory receptor gene family are contained in large blocks of DNA duplicated polymorphically near the ends of human chromosomes.. *Hum Mol Genet,* 7(1), pp. 13-26.

Unryn, B. M., Cook, L. S. & Riabowol, K. T., 2005. Paternal age is positively linked to telomere length of children. *Aging Cell,* 4(2), pp. 97-101.

van Steensel, B., Smogorzewska, A. & de Lange, T., 1998. TRF2 protects human telomeres from end-to-end fusions. *Cell,* Volume 92, p. 12.

Vaziri, H. et al., 1994. Evidence for a mitotic clock in human hematopoietic stem cells: loss of telomeric DNA with age.. *Proc Natl Acad Sci USA,* 91(21), pp. 9857-9860.

Venteicher, A. S., 2009. A human telomerase holoenzyme protein required for Cajal. *Science,* 323(5914), pp. 644-648.

Vera, E. & Blasco, M. A., 2012. Beyond average: potential for measurement of short telomeres. *Aging,* 4(6), pp. 379-392.

von Zglinicki, T. & Martin-Ruiz, C. M., 2005. Telomeres as Biomarkers for Ageing and Age-Related Diseases. *Current Molecular Medicine,* 5(2), pp. 197-203.

von Zglinicki, T., Pilger, R. & Sitte, N., 2000. Accumulation of single-strand breaks is the major cause of telomere shortening in human fibroblasts. *Free Radic Biol Med,* 28(1), pp. 64-74.

Vulliamy, T. J., Knight, S. W., Mason, P. J. & Dokal, I., 2001. Very short telomeres in the peripheral blood of patients with X-linked and autosomal dyskeratosis congenita.. *Blood Cells Mol Dis,* 27(2), pp. 353-357.

Willat, L. et al., 2005. 3q29 microdeletion syndrome: clinical and molecular characterization of a new syndrome. *Am J Hum Genet,* 77(1), pp. 154-160.

Willeit, P. et al., 2010. Telomere length and risk of incident cancer and cancer mortality.. *JAMA,* 304(1), pp. 69-75.

Wilson, W. R. et al., 2008. Blood leucocyte telomere DNA content predicts vascular telomere DNA content in humans with and without vascular disease.. *Eur Heart J,* 29(21), pp. 2689-2694.

Wong, J. Y. et al., 2014. The Relationship between Inflammatory Biomarkers and Telomere Length in an Occupational Prospective Cohort Study. *PLoS One,* 9(1), p. e87348.

Wright, W. E. et al., 1997. Normal human chromosomes have long G-rich telomeric overhangs at one end. *Genes Dev.,* Volume 11, p. 8.

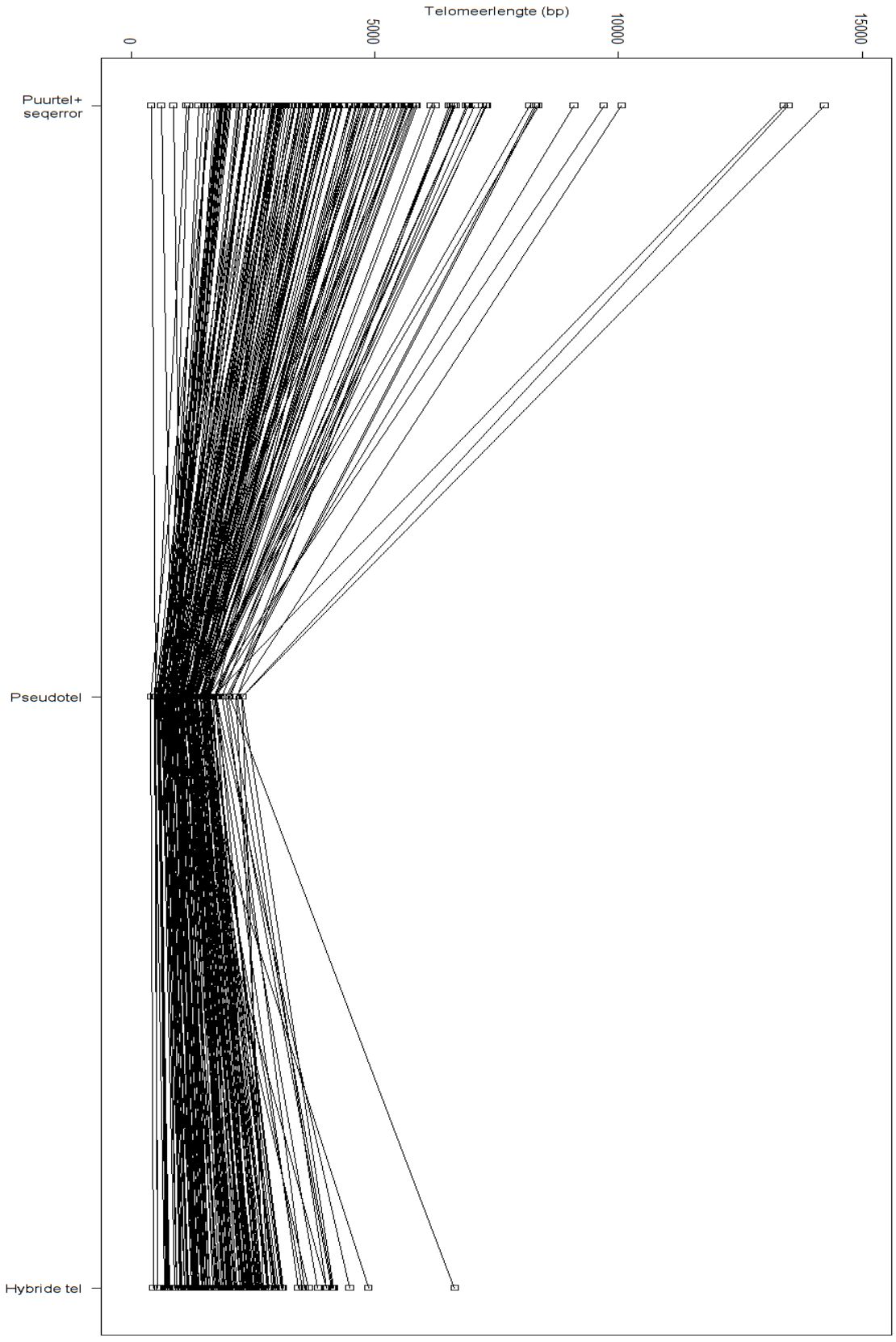
Wyatt, H. D., Lobb, D. A. & Beattie, T. L., 2007. Characterization of Physical and Functional Anchor Site Interactions in Human Telomerase. *Molecular and Cellular Biology,* 27(8), pp. 3226-3240.

Xin, H. et al., 2007. TPP1 is a homologue of ciliate TEBP-beta and interacts with POT1 to recruit telomerase.. *Nature,* 445(7127), pp. 559-562.

Zhu, Y. et al., 2004. Telomerase RNA accumulates in Cajal bodies in human cancer cells. *Mol Biol Cell,* 15(1), pp. 81-90.

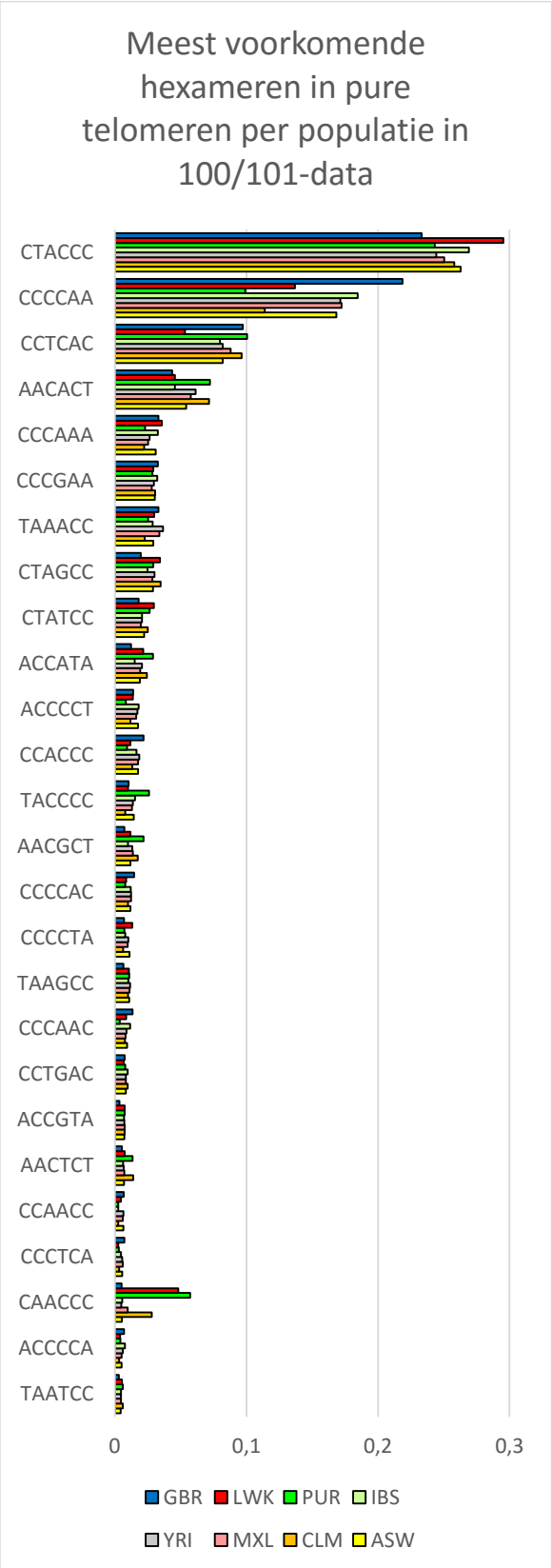
6. Bijlage

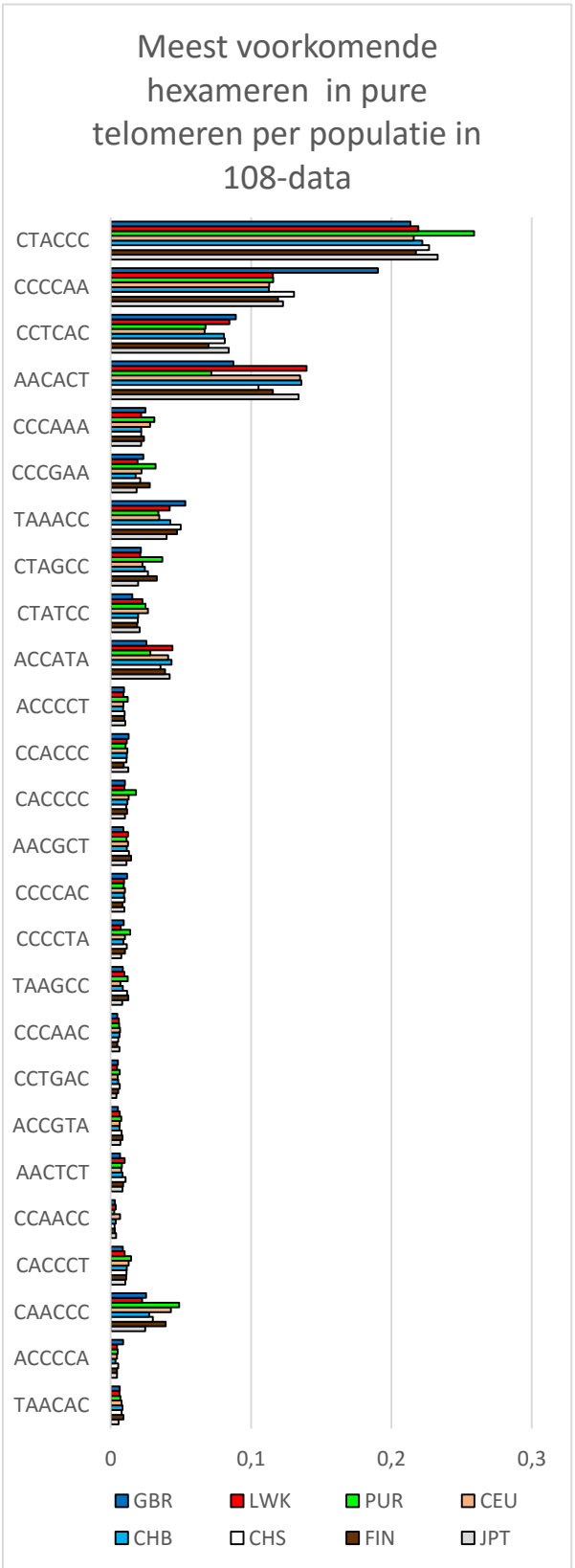
Bijlage 1: Stripchart van 100/101-data

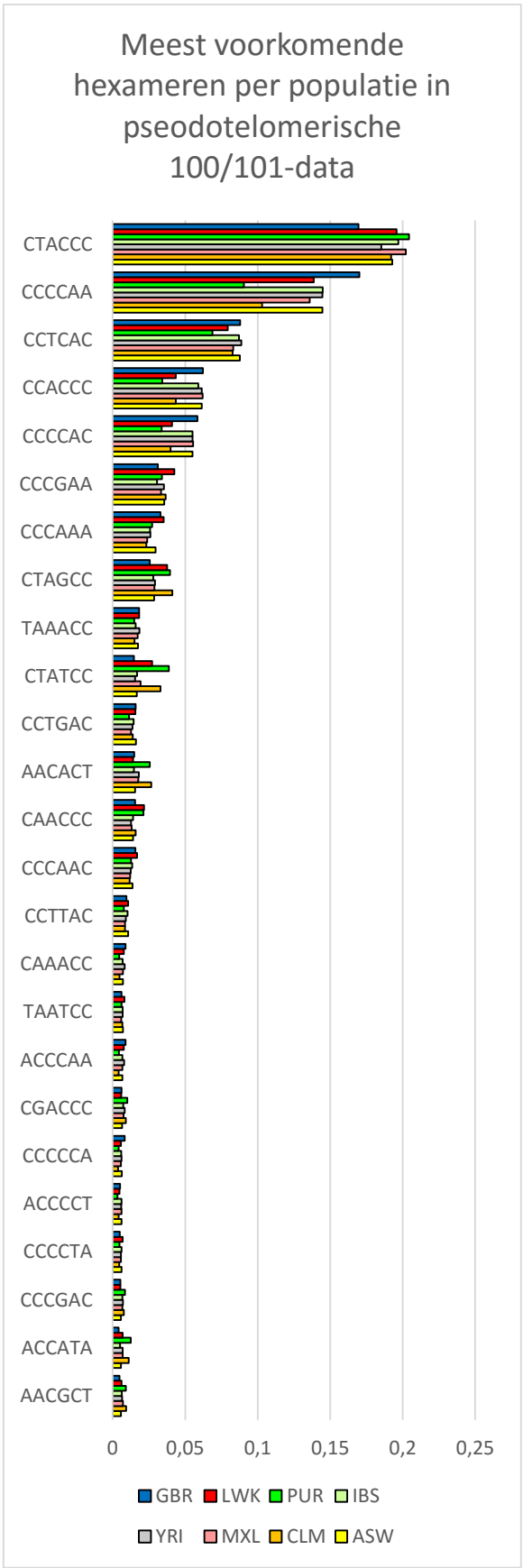


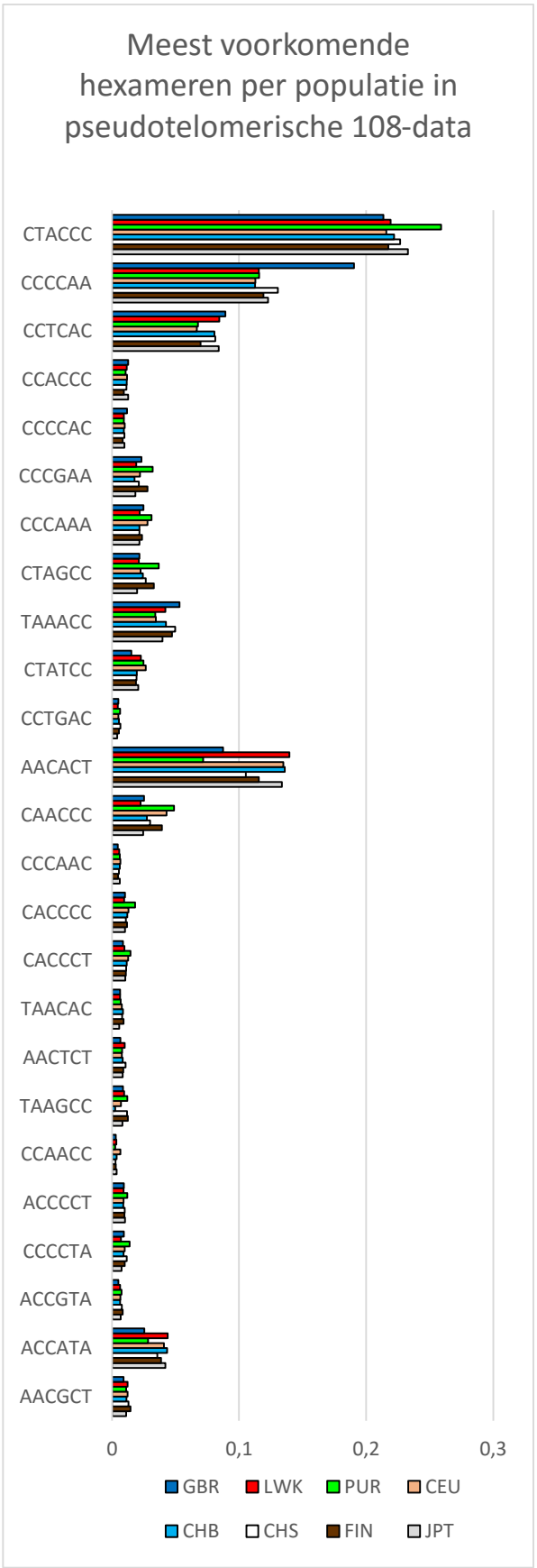
Bijlage 2: Stripchart van 108-data

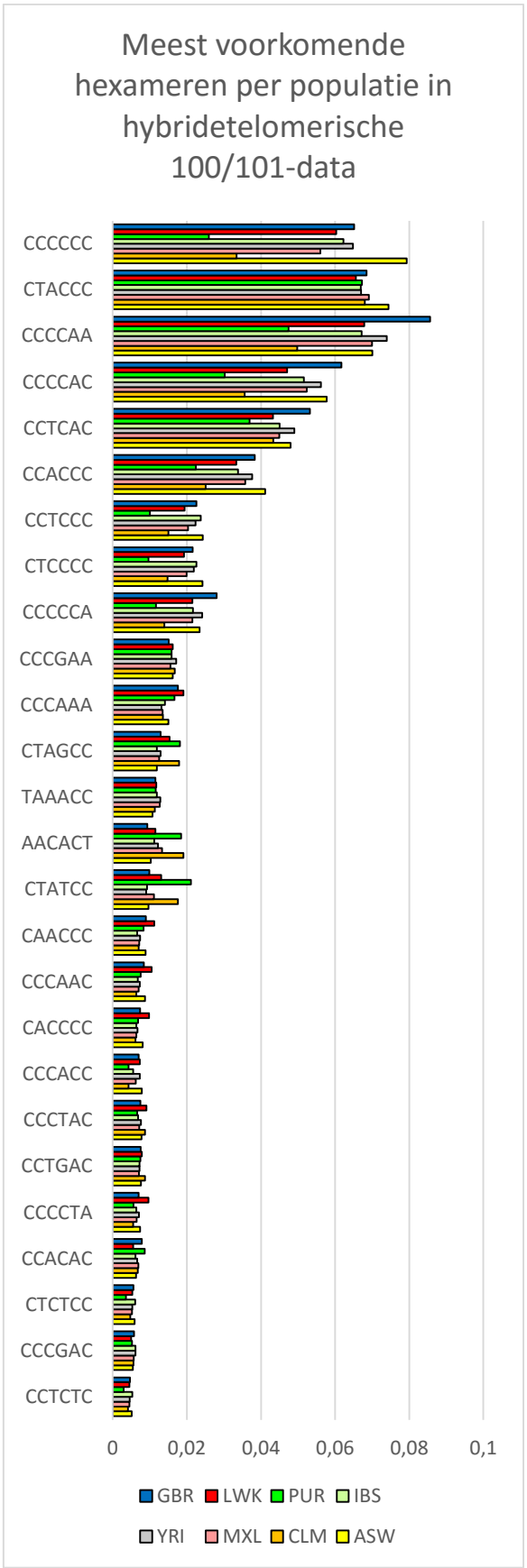
Bijlage 3: Overzicht van hexameren in puurtelomerische, pseudotelomerisch en hybridetelomerische fractie

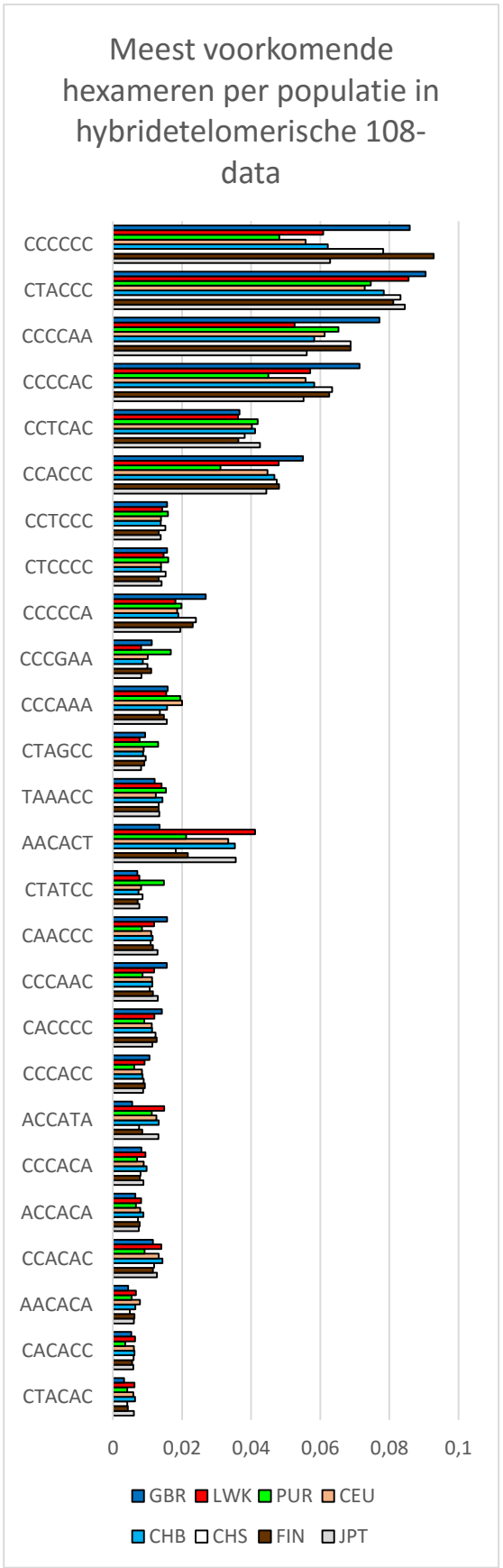





Meest voorkomende hexameren per populatie in pseodotelomerische 100/101-data
CTACCC
CCCCAA
CCTCAC
CCACCC
CCCCAC
CCCGAA
CCCAAA
CTAGCC
TAAACC
CTATCC
CCTGAC
AACACT
CAACCC
CCCAAC
CCTTAC
CAAACC
TAATCC
ACCCAA
CGACCC
CCCCCA
ACCCCT
CCCCTA
CCCGAC
ACCATA
AACGCT
0
0,05
0,1
0,15
0,2
0,25
GBR
LWK
PUR
IBS
YRI
MXL
CLM
ASW


Meest voorkomende hexameren per populatie in pseudotelomerische 108-data
CTACCC
CCCCAA
CCTCAC
CCACCC
CCCCAC
CCCGAA
CCCAAA
CTAGCC
TAAACC
CTATCC
CCTGAC
AACACT
CAACCC
CCCAAC
CACCCC
CACCCT
TAACAC
AACTCT
TAAGCC
CCAACC
ACCCCT
CCCCTA
ACCGTA
ACCATA
AACGCT
0
0,1
0,2
0,3
GBR
LWK
PUR
CEU
CHB
CHS
FIN
JPT


Meest voorkomende hexameren per populatie in hybridetelomerische 100/101-data
CCCCCC
CTACCC
CCCCAA
CCCCAC
CCTCAC
CCACCC
CCTCCC
CTCCCC
CCCCCA
CCCGAA
CCCAAA
CTAGCC
TAAACC
AACACT
CTATCC
CAACCC
CCCAAC
CACCCC
CCCACC
CCCTAC
CCTGAC
CCCCTA
CCACAC
CTCTCC
CCCGAC
CCTCTC
0
0,02
0,04
0,06
0,08
0,1
GBR LWK PUR IBS
YRI MXL CLM ASW


Meest voorkomende hexameren per populatie in hybridetelomerische 108-data
CCCCCC
CTACCC
CCCCAA
CCCCAC
CCTCAC
CCACCC
CCTCCC
CTCCCC
CCCCCA
CCCGAA
CCCAAA
CTAGCC
TAAACC
AACACT
CTATCC
CAACCC
CCCAAC
CACCCC
CCCACC
ACCATA
CCCACA
ACCACA
CCACAC
AACACA
CACACC
CTACAC
0
0,02
0,04
0,06
0,08
0,1
GBR LWK PUR CEU
CHB CHS FIN JPT

Bijlage 4: Computel scripts

Bijlage 4A: computel_download_verwerk_mapped_unmapped.R

```
#Alle .bam URL's die zullen worden gedownload
URLarray <- c(
"ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1/data/HG00101/alignment/HG00101.mapped.SOLID.bfast.GBR.low_coverage.20101123.bam",
"ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1/data/HG00101/alignment/HG00101.unmapped.SOLID.bfast.GBR.low_coverage.20101123.bam")

#variabele count benoemen (ipv i) dat het array zal overlopen, per loop 1.
count<-1

while (count<=length(URLarray))
{

###Individunummer isoleren vanuit de URL: HG00099, NA00145 etc.
individu<-substr(URLarray[count],54,60)

###Vertellen aan de gebruiker welke file er zal worden gedownload
cat("\n==========\nStarting with file ",count," of ",length(URLarray),":",individu)
cat("\n==========\n")
###Plaats selecteren waar de .bam file opgeslaan wordt
saveplace <- paste('/home/driesg/workplace/',individu,'.bam',sep="")
saveplacemapped <- paste('/home/driesg/workplace/',individu,'mapped.bam',sep="")
saveplaceunmapped <- paste('/home/driesg/workplace/',individu,'unmapped.bam',sep="")

###.Bam file downloaden en dit vertellen aan de gebruiker
#cat("\nDownloading WGS data from ",URLarray[count]," and saving in ",saveplacemapped,"\n\n")
#download.file(url=URLarray[count],destfile=saveplacemapped)
cat("\nDownloading WGS data from ",URLarray[count+1]," and saving in ",saveplaceunmapped,"\n\n")
download.file(url=URLarray[count+1],destfile=saveplaceunmapped)

###Fuseren van de 2 .bam files naar 1 .bam file
cat("\nConverging the unmapped and mapped reads to one file.\n\n")
argsystem2<-paste('merge ',saveplace,' ',saveplacemapped,' ',saveplaceunmapped,sep="")
system2('/home/driesg/computel/samtools-0.1.19-linux/samtools', args = argsystem2)

###Hier telseq met read length aanpassen!
#cat("Using telseq to calculate the telomere length.\n")
#argsystem2<-paste(saveplace,' -u -r108 -o /home/driesg/workplace/',individu,'telseq.txt',sep="")
#argsystem2<-paste(saveplace,' -u -o /home/driesg/workplace/',individu,'telseq.txt',sep="")
#system2('/home/driesg/telseq/src/Telseq/telseq', args = argsystem2)

###.Bam omzetten naar .fq en dit vertellen aan de gebruiker. Een variabele argsystem2 aanmaken dat alle argumenten bevat voor system2
cat("Using samtools to convert the .bam file to .fq.\n")
argsystem2<-paste('bam2fq ',saveplace,' >/home/driesg/workplace/',individu,'.fq',sep="")
```

```
system2('/home/driesg/computel/samtools-0.1.19-linux/samtools', args = argsystem2)

###Bepalen waar de .fq file zit van het gedownloade en omgezette genoom
fastq <- paste('/home/driesg/workplace/',individu,'.fq',sep="")

###Read length berekenen van de eerste 4 lijntjes van de omgezette .fq file en dit meegeven aan de gebruiker
argsystem2<-paste(fastq," | head -4 | paste - - - - | cut -f 2 | tr -d '\n' | wc -c",sep="")
#read.length=76
read.length <- system2('cat', args = argsystem2, stdout=TRUE)
cat("\nThe read length was calculated to be ",read.length," bases.\n")

###Aantal reads bepalen en de daaruitvolgende base coverage: (aantal reads * read length) / genome length
argsystem2<-paste('$(cat ',fastq,' | wc -l) / 4',sep="")
read.number <- system2('expr', args = argsystem2, stdout=TRUE)
cat("\nThe read number was calculated to be ",read.number,".\n")
genome.length=3101804739
base.cov=((as.numeric(read.length) * as.numeric(read.number))/genome.length)
#base.cov=""
cat("\nThe base coverage was calculated to be ",base.cov,".\n")


#####################################################################
#################################
      ######      Configuration options                                    ######

#####################################################################
#################################
      ###directories

      scripts.dir="/home/driesg/computel/src/scripts/"
      bowtie.build.path="/home/driesg/computel/bowtie2-2.1.0-linux/bowtie2-build"
      bowtie.align.path="/home/driesg/computel/bowtie2-2.1.0-linux/bowtie2-align"
      samtools.path="/home/driesg/computel/samtools-0.1.19-linux/samtools"
      picard.samtofastq.jar="/home/driesg/computel/SamToFastq.jar"

      ###input_reads
      quals = "--phred33" ##default: --phred33, alternatives: --phred64, --solexa-quals

      #treat reads as single-end (single set to T) or paired-end (single set to F)
      single=T
      # note: paired-end reads can effectively be treated as single-end, thus single is 'T' by default
      # note: compressed files can only be supplied as single-end reads

      ## single-end reads
      # if files.with.prefix is 'T', files should be specified by filename prefix;
      # if it is 'F' - by absolute paths
      files.with.prefix =  F

      # single-end reads specified by filename prefix (files.with.prefix is set to T)
      fastq.prefix=""
      fastq.dir=""

      #single-end reads specified by file paths (files.with.prefix is set to F)
```

```
	# file compression type (can be 'F' (not compressed), "gz" or "bz2")
	# works only for single-end reads!

	file.compression = 'F'

	# if paired-end read alignment is prefered, 'single' should be set to 'F', and read pairs specified below:
	# paired-end reads file paths (set single to F)
	fastq1=""
	fastq2=""

	###base_coverage_calculation_options
	compute.base.cov=F
  #the base coverage is given (F) or should be computed (T)
	base.index.pathtoprefix=""

	estimate.base.cov =F
	#if files are compressed and the base coverage needs to be estimated during unzipping, set "estimate.base.cov" to T

	###output_options

	output.dir='/home/driesg/workplace/'

	###algorithm_options

	pattern='TTAGGG'
	num.haploid.chr=23
	min.seed=12
  #min.seed=as.numeric(read.length)
	mode.local=F
	###system_options

	num.proc=3
	ignore.err=F

	###additional_options

	quals="--phred33" #default: --phred33, alternatives: --phred64, --solexa-quals
	ignore.err = T



##############################################################
#########
	####     assemble options in config.table for validation     ####

##############################################################
#########

	config.table = NULL
	config.table['scripts.dir'] = scripts.dir
	config.table['bowtie.build.path'] = bowtie.build.path
	config.table['bowtie.align.path'] = bowtie.align.path
	config.table['samtools.path'] = samtools.path
	config.table['picard.samtofastq.jar'] = picard.samtofastq.jar
```

```
        config.table['fastq1'] = fastq1
        config.table['fastq2'] = fastq2
        config.table['single'] = single
        config.table['fastq'] = fastq
        config.table['files.with.prefix'] = files.with.prefix
        config.table['fastq.dir'] = fastq.dir
        config.table['fastq.prefix']=fastq.prefix
        config.table['file.compression']=file.compression
        config.table['read.length'] = read.length

        config.table['pattern'] = pattern
        config.table['num.haploid.chr'] = num.haploid.chr
        config.table['min.seed'] = min.seed
        config.table['mode.local'] = mode.local

        config.table['compute.base.cov'] = compute.base.cov
        config.table['estimate.base.cov'] = estimate.base.cov
        config.table['genome.length'] = genome.length
        config.table['base.cov'] = base.cov
        config.table['base.index.pathtoprefix'] = base.index.pathtoprefix

        config.table['output.dir'] = output.dir
        config.table['num.proc'] = num.proc
        config.table['quals']=quals
        config.table['ignore.err']=ignore.err

        config.table = as.matrix(config.table)

        validate.R = file.path(scripts.dir, "validate.options.R")
        if (!file.exists(validate.R))
        validate.R = "validate.options.R"

        if (!file.exists(validate.R)){
          stop("validate.options.R not found.\n Provide scripts.dir containing the script.")
        } else {
          config.set = T
          source(validate.R)
        }

        dir.create(output.dir, showWarnings=F)

        if (!config.set){
          stop("configuration not set successfully. Scripts will not execute.\n")
        } else {
          source(pipeline.R)
        }

count<-count+2
}
```

Bijlage 4B: functions.R

```
call.cmd <- function(command){
  OS.name = tolower(Sys.info()["sysname"])
  if (OS.name == "windows")
    shell(command, wait = T)
  else
    system(paste("bash -c", "'", command, "'"), wait = T)
}

build.tel.index <- function(pattern, read.length, min.seed,
                            tel.index.dir, tel.index.prefix,
                            bowtie.build.path, ignore.err = F){
  pattern = string2vector(pattern)
  pattern = complement(pattern)
  pattern.length = read.length+length(pattern)-1;
  reps = floor(pattern.length/length(pattern))
  if(pattern.length-reps*length(pattern) > 0)
    left.rem = pattern[1:(pattern.length-reps*length(pattern))] else
      left.rem=NULL
  tel.index = c(rep(pattern,reps), left.rem)
  tel.index = c(tel.index, rep("N", read.length-min.seed))

  tel.index.fasta = paste(tel.index.prefix, ".fasta",sep="")

  write.fasta(sequences=tel.index, names=c(">fwd: DNA"),
              file=file.path(tel.index.dir, tel.index.fasta))
  this = getwd()
  setwd(tel.index.dir)
  command = paste(bowtie.build.path, '--quiet', tel.index.fasta,  tel.index.prefix)
  print(command)
  call.out = call.cmd(command)
  setwd(this)
  success = file.exists(paste(tel.index.dir,"/", tel.index.prefix, ".1.bt2",sep=""))
  if (success){
    if (!ignore.err){
      success = (call.out == 0)
    }
  }
  return(success)

}

complement <- function(pattern) {
  pattern = rev(toupper(pattern))
  letters =      c("A", "T", "G", "C", "W", "S", "M", "K", "R", "Y", "N");
  comp.letters = c("T", "A", "C", "G", "S", "W", "K", "M", "Y", "R", "N");
  comp.pattern = pattern;
  for (i in 1:length(letters)){
    ind = which(pattern == letters[i]);
    if(length(ind)>0)
      comp.pattern[ind] <-comp.letters[i];
  }
  return(comp.pattern);
}

tel.length <- function(coverage.file, rl, pl,
```

```
                    base.cov, num.haploid.chr, out.file, mutationfrequency){

  out.list = list()
  out.list$coverage.file = coverage.file
  out.list$read.length = rl
  out.list$pattern.length = pl
  out.list$base.cov = base.cov
  out.list$num.haploid.chr = num.haploid.chr
  out.list$tel.length = 0

  ###mutationfrequency toevoegen
  out.list$mutationfrequency = mutationfrequency



  table.tel = read.table(file=coverage.file,col.names=c("position","orientation", "depth"));
  tl = length(table.tel$depth)
  range = c(1:(rl+pl-1))

  if(length(table.tel[,1]) < length(range))
    depth.tel=table.tel$depth/2/base.cov
  else
    depth.tel = table.tel$depth[range]/2/base.cov

  mean.length.rl.pl = mean(depth.tel)*(rl+pl-1)
  tel.length = mean.length.rl.pl/num.haploid.chr

  out.list$tel.length = tel.length
  out.table = as.matrix(out.list)

  cat("Computing telomere length from the followin parameters:\n")
  cat("coverage.file= ", coverage.file,"\n")
  cat("rl= ", rl,"\n")
  cat("pl= ", pl,"\n")
  cat("base.cov= ", base.cov,"\n")
  cat("num.haploid.chr= ", num.haploid.chr,"\n")
  cat("tel.length=",  tel.length, "\n")

  ###mutationfrequency toevoegen
  cat("mutationfrequency=",  mutationfrequency, "\n")

  write.table(out.table, file = out.file, sep="\t")

  return(tel.length)
}

base.coverage <- function(base.coverage.file){
  coverage.table       =       read.table(file=coverage.file,col.names=c("position","orientation",
"depth"));
  depth = coverage.table$depth
  mean.cov = mean(depth)
  return(mean.cov)
}

string2vector <- function(pattern){
  pattern = toupper(pattern)
  if(length(pattern)==1)
```

```
    #     pattern = substring(pattern, seq(1,nchar(pattern),1), seq(1,nchar(pattern),1))
    pattern = strsplit(pattern,"")[[1]];
  return(pattern)
}


bowtie.align <- function(bowtie.align.path, x, m1=NA, m2=NA, U=NA, S = "align.sam",
                         mode="--end-to-end",
                         additional.options = "", ignore.err = F) {
  args = ls();
  index = as.character(paste('-x', x))

  if (!'U' %in% unlist(args) || is.na(U))
    reads = as.character(paste('-1', m1,'-2', m2))
  else
    reads = as.character(paste('-U', U));


  samout = as.character(paste('-S', S))
  if (file.exists(S))
    file.remove(S)

  options = as.character(paste('-q', mode, '--quiet', additional.options))
  cat("\nPerforming alignment with command:\n")
  command = as.character(paste(bowtie.align.path, options,index, reads, samout));
  cat(command)
  call.out = call.cmd(command)
  success = (file.exists(S) && file.info(S)$size > 0)
  if (success){
    if (!ignore.err){
      success = (call.out == 0)
    }
  }
  return(success)
}
#unzippes the files to stdout and pipes them to bowtie stdin as input.
#command template:
#cat fastq1 fastq2 ... | bunzip2(gunzip) - | tee >(wc -l > length) |
# bowtie2-align -p additional.options -x index -U - -S samout

bowtie.align.compressed <- function(bowtie.align.path, x, U=NA, S = "align.sam",
                         mode="--end-to-end",
                         additional.options = "", ignore.err = F,
                         file.compression, estimate.base.cov) {
  args = ls();
  index = as.character(paste('-x', x))


  reads = as.character(paste('-U', U));
  U.gz = gsub(",", replacement=" ", U)


  samout = as.character(paste('-S', S))
  if (file.exists(S))
    file.remove(S)

  options = as.character(paste('-q', mode, '--quiet', additional.options))
```

```
  if(file.compression == "gz")
    unzipper = "gunzip - "
  else
    unzipper = "bunzip2 - "
  cat("\nPerforming alignment with command:\n")
  if(estimate.base.cov){
    command = as.character(paste("cat", U.gz, "|", unzipper, "|",
                                 "tee >(wc -l > length) |", bowtie.align.path,
                                 options, index, "-U - ", samout))

    cat(command)
    call.out = call.cmd(command)
    success = (file.exists(S) && file.info(S)$size > 0)
    if (success){
      if (!ignore.err){
        success = (call.out == 0)
      }
    }
    if(success){
      success = (file.exists("length") && file.info("length")$size > 0)
      if(!success){
        cat("could not find file length for base coverage estimation \n")
      }
    }
  } else {
    command = as.character(paste("cat", U.gz, "|", unzipper, "|",
                                 bowtie.align.path,
                                 options, index, "-U - ", samout))

    cat(command)
    call.out = call.cmd(command)
    success = (file.exists(S) && file.info(S)$size > 0)
    if (success){
      if (!ignore.err){
        success = (call.out == 0)
      }
    }
  }
  return(success)
}

coverage.from.sam <- function(samtools.path, sam.file, bam.file.name, dir){
  #Convert sam to bam
  ####################
  #bam.file = file.path(dir, "tel.align.bam")
  bam.file = file.path(dir, bam.file.name)
  samtobam = paste(samtools.path, "view -bSh", sam.file, ">", bam.file)
  cat("\nConverting sam file", sam.file,"to bam file", bam.file, "...\n")
  call.cmd(samtobam)

  #Sort bam
  ####################
  sorted.file = file.path(dir, paste(bam.file.name, "_sorted",sep=""))
  sort = paste(samtools.path, "sort", bam.file, sorted.file)
  cat("\nSorting bam file", bam.file, "to", sorted.file, "\n")
  call.cmd(sort)
```

```
  #Calculate depth
  ###################
  sorted.file = paste(sorted.file, ".bam", sep="")
  coverage.file = file.path(dir, paste(bam.file.name, ".coverage.txt", sep=""))
  depth = paste(samtools.path, "depth", sorted.file, ">", coverage.file)
  cat("\nCalculating depth of coverage from bam file", sorted.file, "to", coverage.file, "\n")
  call.cmd(depth)
  return(coverage.file)
}
```

Bijlage 4C: pipeline.R

```
if (!"seqinr" %in% installed.packages()) {

  #   if (tolower(OS.name) != "windows") {
  #     lib.paths = .libPaths()
  #     user.dir.ind = which(regexpr("/home/", lib.paths, fixed=T)==1)
  #     if(length(user.dir.ind) > 0 ) {
  #       install.lib = lib.paths[ind]
  #     } else {
  #
  #     }
  #
  #   }

  install.packages("seqinr", repos = "http://cran.rstudio.com/")
}
if (!"psych" %in% installed.packages())
  install.packages("psych", repos = "http://cran.rstudio.com/")

library("seqinr")
source(file.path(scripts.dir, "functions.R"))

# Formatting variables
#####################################################################
#################################
#####################################################################
#################################

tel.index.prefix = "tel_index"
tel.index.fasta = "tel_index.fasta"
tel.index.dir = file.path(output.dir, "index")
dir.create(tel.index.dir, showWarnings=F)

base.index.prefix="base_index"
tel.genome.fasta = "telomeric.genome.fasta"

align.dir = file.path(output.dir, "align")
dir.create(align.dir, showWarnings=F)

# Build telomeric index
#####################################################################
#################################
#####################################################################
#################################
cat("\nBuilding telomeric index in the directory", tel.index.dir, "\n")
success = build.tel.index(pattern, rl, min.seed, tel.index.dir, tel.index.prefix,
                          bowtie.build.path, ignore.err)
if (!success){
  stop("Problem building telomeric index")
} else {

  # Align reads to telomeric index

#####################################################################
#################################
```

```
##################################################################
#################################
  align.args = list()
  align.args$bowtie.align.path = bowtie.align.path
  align.args$x = file.path(tel.index.dir, tel.index.prefix)
  align.args$S = file.path(align.dir, "tel.align.sam")

 l = floor(rl/3)
 if (l < 6)
   l = 6
 if (l > 22)
   l = 22

  no.unal = ifelse(compute.base.cov,"", "--no-unal")

  align.args$additional.options = paste("-p", num.proc,quals,
                                        no.unal, "--n-ceil", rl-min.seed,
                                        "-D 20 -R 3 -N 1 -L", l, "-i S,1,0.5")


  #align.args$additional.options = paste("-p", num.proc,quals,
  #                                      no.unal, "--n-ceil ", rl-min.seed,
  #                                      " -D 15 -R 2 -N 0 -L", l, "-i S,1,1.15")


  align.args$mode = ifelse(mode.local, "--local", "--end-to-end")

  if(single){
    align.args$U = paste(fastqs,collapse=",")
    reads = as.character(paste('-U', align.args$U))
  } else {
    align.args$m1 = fastq1
    align.args$m2 = fastq2
    reads = as.character(paste('-1', align.args$m1,'-2', align.args$m2))
  }
  align.args$ignore.err = ignore.err
  if(compressed){
    align.args$file.compression = file.compression
    align.args$estimate.base.cov = estimate.base.cov
    success = do.call(bowtie.align.compressed, args=align.args)
    if(estimate.base.cov){
      length = as.numeric(read.table("length", header=F))
      base.cov = length/4*rl/genome.length
      cat("\nesimated base coverage: ", base.cov, "\n")
    }
  } else {
    success = do.call(bowtie.align, args=align.args)
  }
  if (!success){
    stop("Problem aligning reads to telomeric index\n")
  } else {

    #If compute.base.cov=T, separate mapped from unmapped reads

##################################################################
####
```

```
    if(!compute.base.cov){
      reads.mapped = align.args$S
    } else {
      cat("Separating mapped from unmapped reads\n")
      sam.file = align.args$S
      reads.mapped = file.path(align.dir, "tel.align.mapped.sam")
      reads.unmapped = file.path(align.dir, "tel.align.unmapped.sam")

      command.unmapped = paste(samtools.path, "view -Sh -f 4", sam.file,'>',
reads.unmapped)
      command.mapped = paste(samtools.path, "view -Sh -F 4", sam.file,'>', reads.mapped)
      call.cmd(command.unmapped)
      call.cmd(command.mapped)

      cat("Converting unmapped reads to fastq with command\n")
      samtofastq.command = paste("java -jar", picard.samtofastq.jar,
                                 "VALIDATION_STRINGENCY=LENIENT",
                                 paste("INPUT=", reads.unmapped, sep=""))
      base.dir = file.path(output.dir, "base")
      dir.create(base.dir, showWarnings=F)
      file.prefix = paste(base.dir, "/reads.unmapped", sep="")
      if (!single){
        base.fastq1 = paste(file.prefix, "_1.fastq", sep="")
        base.fastq2 = paste(file.prefix, "_2.fastq", sep="")
        base.fastq.unpaired = paste(file.prefix, "_unpaired.fastq", sep="")

        base.fastq1.opt = paste("FASTQ=", base.fastq1, sep="")
        base.fastq2.opt = paste("SECOND_END_FASTQ=", base.fastq2, sep="")
        base.fastq.unpaired.opt = paste("UNPAIRED_FASTQ=", base.fastq.unpaired, sep="")
        samtofastq.command = paste(samtofastq.command, base.fastq1.opt,
                                   base.fastq2.opt, base.fastq.unpaired.opt)
      }
      else {
        base.fastq = paste(file.prefix, ".fastq", sep="")
        base.fastq.opt = paste("FASTQ=", base.fastq, sep="")
        samtofastq.command = paste(samtofastq.command, base.fastq.opt, sep=" ")
      }
      print(samtofastq.command)
      call.cmd(samtofastq.command)
    }

    # Compute telomeric alignment coverage

#########################################################################
#################################

#########################################################################
#################################
    cat("Computing telomeric index coverage")
    tel.coverage = coverage.from.sam(samtools.path, sam.file= reads.mapped,
bam.file.name="tel.align.bam", dir=align.dir)

    # Compute base coverage

#########################################################################
#################################
```

```
##############################################################
##################################

   if(compute.base.cov){
     cat("Computing base coverage\n")

     #Calculate base coverage from alignment if compute.base.cov = T

##############################################################
##########
     cat("Aligning unmapped reads to reference\n")
     #Align unmapped reads to reference

##############################################################
##################################
     base.align.args = list()
     base.align.args$bowtie.align.path = bowtie.align.path
     base.align.args$x = file.path(base.index.pathtoprefix)
     base.align.args$S = file.path(base.dir, "base.align.sam")

     base.align.args$additional.options = paste("-p", num.proc, quals)
     base.align.args$mode = ifelse(mode.local, "--local", "--end-to-end")

     if(single){
       base.align.args$U = base.fastq
       reads = as.character(paste('-U', base.align.args$U))
     } else {
       base.align.args$m1 = base.fastq1
       base.align.args$m2 = base.fastq2
       reads = as.character(paste('-1', base.align.args$m1,'-2', base.align.args$m2))
     }

     cat("Performing alginment with command: \n\n")
     success = do.call(bowtie.align, args=base.align.args)
     if (!success){
       stop("Problem aligning reads to base index")
     } else {

       cat("Computing base index coverage\n")
       coverage.file = coverage.from.sam(samtools.path, sam.file= base.align.args$S,
bam.file.name="base.align.bam", dir=base.dir)

       base.cov = base.coverage(coverage.file)
     }
   }
   # Compute telomere length

##############################################################
##################################

##############################################################
##################################

  ###Mutatiefrequentie berekenen
```

```
  argsystem2<-paste("view -h /home/driesg/workplace/align/tel.align.bam  | awk -F'\t' '{
if($18 ~ /MD/) {print $18} else if ($19 ~ /MD/) {print $19} }' | grep -Eo 'A|C|T|G' | wc
-m",sep="")
  numberofmutations <- system2('samtools', args = argsystem2, stdout=TRUE)
  cat("\nSuccesfully calculated the number of mutations: ",numberofmutations,"\n")
  mutationfrequency = as.numeric(numberofmutations)/genome.length
  cat("\nSuccesfully calculated the mutationfrequency: ",mutationfrequency,"\n")

  cat("estimating telomere length \n")
      #tel.length.out<-paste(output.dir,individu,'.xls',sep="")
      #tel.length.out = file.path(output.dir, "tel.length.xls")
      tel.length.out = file.path(output.dir,"/",individu,".xls",fsep="")
      tel.length  = tel.length(tel.coverage, rl, pl=  nchar(pattern),base.cov, num.haploid.chr,
tel.length.out, mutationfrequency)

      cat("Telomere computation successfully complete.\n")
      cat("Output stored at ", tel.length.out, "\n")

  ###Files .bam en .fq verwijderen, .xls wordt behouden
  #argsystem2<-paste('/home/driesg/workplace/',individu,'.fq
/home/driesg/workplace/',individu,'.bam  /home/driesg/workplace/',individu,'mapped.bam
/home/driesg/workplace/',individu,'unmapped.bam',sep="")
  #system2('rm', args = argsystem2)
      #cat("\nSuccesfully deleted the two ",individu,".fq and .bam datafiles.\n")

  ###tel.align.bam omzetten naar een .txt - dit wordt automatisch behouden
  argsystem2<-paste('view  /home/driesg/workplace/align/tel.align.bam  >
/home/driesg/workplace/align/tel.align.',individu,'.txt',sep="")
  system2('/home/driesg/computel/samtools-0.1.19-linux/samtools', args = argsystem2)

 }
}
```

Bijlage 4D: validate.options.R

```
set.property <- function(matrix, prop.name, default.value){
  result = tryCatch({
    matrix[prop.name,1]
  }, warning = function(w) {
    cat("property \"", prop.name, "\" not found. Default value assigned: ", default.value,
"\n")
    default.value
  }, error = function(e) {
    cat("property \"", prop.name, "\" not found. Default value assigned: ", default.value,
"\n")
    default.value
  })
  return(result)
}

set.property.double <- function(matrix, prop.name, default.value){
  result = set.property(matrix, prop.name, default.value)
  result = tryCatch({
    as.double(result)
  }, warning = function(w) {
    cat("Warning: Illegal value for", prop.name, ": ", result,
        "Should be double. Default value assigned: ", default.value, "\n")
    default.value
  }, error = function(e) {
    cat("Error: Illegal value for", prop.name, ": ", result,
        "Should be double. Default value assigned: ", default.value, "\n")
    default.value
  })
}

set.property.integer <- function(matrix, prop.name, default.value){
  result = set.property(matrix, prop.name, default.value)
  result = tryCatch({
    as.integer(result)
  }, warning = function(w) {
    cat("Warning: Illegal value for", prop.name, ": ", result,
        "Should be integer. Default value assigned: ", default.value, "\n")
    default.value
  }, error = function(e) {
    cat("Error: Illegal value for", prop.name, ": ", result,
        "Should be integer. Default value assigned: ", default.value, "\n")
    default.value
  })
}
set.property.logical <- function(matrix, prop.name, default.value){
  result = set.property(matrix, prop.name, default.value)
  result = tryCatch({
    as.logical(result)
  }, warning = function(w) {
    cat("Warning: Illegal value for", prop.name, ": ", result,
        "Should be logical. Default value assigned: ", default.value, "\n")
    default.value
  }, error = function(e) {
    cat("Error: Illegal value for", prop.name, ": ", result,
        "Should be logical. Default value assigned: ", default.value, "\n")
```

```
    default.value
  })
}

set.property.sequence <- function(matrix, prop.name, default.value){
  result = set.property(matrix, prop.name, default.value)
  result = tryCatch({
    as.character(result)
    result = toupper(result)
    dna.chars = c ("A", "G", "C","T")
    chars = strsplit(result,"")[[1]]
    is.dna = all(chars %in% dna.chars)
    if (!is.dna)
      cat("Illegal value for ", prop.name, ": ", result,
          "Should contain DNA sequence characters ", dna.chars, ". Defualt value assigned:
",
          default.value, "\n")
    result
  }, warning = function(w) {
    cat("Warning: Illegal value for", prop.name, ": ", result,
        "Should be numeric. Default value assigned: ", default.value, "\n")
    default.value
  }, error = function(e) {
    cat("Error: Illegal value for", prop.name, ": ", result,
        "Should be numeric. Default value assigned: ", default.value, "\n")
    default.value
  })
}

set.property.executable <- function(matrix, prop.name, default.value){
  result = tryCatch({
    matrix[prop.name,1]
  }, warning = function(w) {
    cat("property \"", prop.name, "\" not found. Default value assigned: ", default.value,
"\n")
    default.value
  }, error= function(e) {
    cat("property \"", prop.name, "\" not found. Default value assigned: ", default.value,
"\n")
    default.value
  })
  if (length(grep("./", result, fixed = T)) > 0
      || length(grep(".\\", result, fixed = T)) > 0){
    wd = getwd()
    setwd("..")
    parent = getwd()
    setwd(wd)
  }

  if (length(grep(pattern="../", result, fixed = T)) > 0){
    result = sub(pattern="..", replacement=parent, result, fixed = T)
  } else if (length(grep(pattern="./", result, fixed = T)) > 0){
    result = sub(pattern=".", replacement=wd, result, fixed = T)
  } else   if (length(grep(pattern="..\\", result, fixed = T)) > 0){
    result = sub(pattern="..", replacement=parent, result, fixed = T)
  } else if (length(grep(pattern=".\\", result, fixed = T)) > 0){
    result = sub(pattern=".", replacement=wd, result, fixed = T)
```

```
  }

  return(result)
}

defaults = list(
  rl = 100,
  pattern = "TTAGGG",
  base.cov = 1,
  num.haploid.chr = 23,
  output.dir = "output",
  compute.base.cov = F,
  estimate.base.cov = F,
  genome.length = 3101804739,
  mode.local = F,
  single = T,
  files.with.prefix = F,
  file.compression = F,
  min.seed = 12,
  num.proc = 3,
  scripts.dir = "./",
  bowtie.dir = "../bowtie2-2.1.0/",
  bowtie.build.path = "..bowtie2-2.1.0/bowtie2-build.exe",
  bowtie.align.path = "../bowtie2-2.1.0/bowtie2-align.exe",
  samtools.path = "../samtools-0.1.19/samtools.exe",
  quals = "--phred33",
  ignore.err = F
)
rl = set.property.integer(config.table, "read.length", defaults$rl)
pattern = set.property.sequence(config.table, "pattern", defaults$pattern)

num.haploid.chr = set.property.integer(config.table, "num.haploid.chr",
defaults$num.haploid.chr)
compute.base.cov = set.property.logical(config.table, "compute.base.cov",
defaults$compute.base.cov)

mode.local = set.property.logical(config.table, "mode.local", defaults$mode.local)
single = set.property.logical(config.table, "single", defaults$single)
files.with.prefix = set.property.logical(config.table, "files.with.prefix", defaults$files.with.prefix)
file.compression = set.property(config.table, "file.compression", defaults$file.compression)
compressed = F
OS.name = tolower(Sys.info()["sysname"])
if (OS.name == "windows") {
  if(file.compression != "F"){
    config.set = F
    cat("file.compression should be set to F for windows systems\n")
  }
} else {
  if(file.compression != "F"){
    if(file.compression == "gz"){
      compressed = T
    } else if (file.compression == "bz2"){
      compressed = T
    } else {
      compressed = F
      cat("file.compressed should be either gz or bz2\n")
      config.set = F
```

```
    }
  }
  if(compressed){
    if(!single){
      cat("compressed fastq files should be given with \"single\" parameter set to T")
      config.set = F
    }
  }
}
estimate.base.cov = set.property.logical(config.table, "estimate.base.cov",
defaults$estimate.base.cov)
if(estimate.base.cov){
  if(!compressed){
    cat("Warning: base coverage is estimated only from gzip or bzip2 compressed files.
Default base coverage", defaults$base.cov, "assigned.\n")
    estimate.base.cov = F
    base.cov = defaults$base.cov
  }
  if(compute.base.cov){
    cat("Error: compute.base.cov and estimate.base.cov cannot be true at the same
time\n")
    config.set = F
  }
}
genome.length = set.property.double(config.table, "genome.length", defaults$genome.length)
min.seed = set.property.integer(config.table, "min.seed", defaults$min.seed)
num.proc = set.property.integer(config.table, "num.proc", defaults$num.proc)

quals = set.property(config.table, "quals", defaults$quals)
if (quals != "--phred33"){
  if (quals != "--phred64"){
    if(quals != "--solexa-quals")
      quals = defaults$quals
  }
}

ignore.err = set.property.logical(config.table, 'ignore.err', defaults$ignore.err)

if(single){
  if (!files.with.prefix){
    fastq = tryCatch({
      config.table["fastq",1]
    }, warning = function(w) {
      cat("\"fastq\" parameter not specified for single mode alignment\n")
      config.set = F
    }, error = function(e) {
      cat("\"fastq\" parameter not specified for single mode alignment\n")
      config.set = F
    })
    names(fastq) <- NULL

    fastqs = unlist(strsplit(fastq, split='[ ,]' , fixed=F))
    fastqs = fastqs[!duplicated(fastqs)]
    empty.fastqs = c()
    for (i in 1:length(fastqs)){
      if (identical(fastqs[i], "")) {
        empty.fastqs = c(empty.fastqs,i)
```

```
    } else if (!file.exists(fastqs[i])){
      cat("fastq file ", fastqs[i],
          " does not exist. Please, specify a valid file for single mode alignment.\n")
      config.set = F
    }
  }
  if(length(empty.fastqs) > 0)
    fastqs=fastqs[-empty.fastqs]
} else {
  fastq.prefix = tryCatch({
    config.table["fastq.prefix",1]
  }, warning = function(w) {
    cat("\"fastq.prefix\" parameter not specified for single mode alignment\n")
    config.set = F
  }, error = function(e) {
    cat("\"fastq.prefix\" parameter not specified for single mode alignment\n")
    config.set = F
  })
  fastq.dir = tryCatch({
    config.table["fastq.dir",1]
  }, warning = function(w) {
    cat("\"fastq.dir\" parameter not specified for single mode alignment\n")
    config.set = F
  }, error = function(e) {
    cat("\"fastq.dir\" parameter not specified for single mode alignment\n")
    config.set = F
  })
  if(compressed){
    suffix = file.compression
    zip.pattern = paste(fastq.prefix, ".*","\\.", suffix,"$", sep="")
    fastq.files = list.files(path=fastq.dir, zip.pattern)
  } else {
    suffix = ""
    fastq.files = list.files(path=fastq.dir, paste(fastq.prefix, ".*\\.f.*q$", sep=""))
  }


  fastqs = file.path(fastq.dir, fastq.files)
  if (length(list.files) == 0)
    cat("no files with pattern ", zip.pattern, " were present in the directory ", fastq.dir)
}

#   check for compression type
if(compressed){
  suffix = file.compression
  for (fastq in fastqs){
    sout = system(paste("file",fastq), intern = T)
    if(suffix == "gz")
      grout = grep("gzip compressed", sout)
    else if(suffix == "bz2")
      grout = grep("bzip2 compressed", sout)
  }
  if(length(grout) == 0){
#     zip.pattern = paste(".*\\.", suffix, "$",sep="")
#     if(length(grep(zip.pattern, fastqs)) < length(fastqs))
    cat("fastq file ", fastq, " was not ", suffix, " compressed", "\n")
    config.set = F
```

```
    }
  }
} else {
  fastq1 = tryCatch({
    config.table["fastq1",1]
  }, warning = function(w) {
    cat("\"fastq1\" parameter not specified for paired mode alignment\n")
    config.set = F
  }, error = function(e) {
    cat("\"fastq1\" parameter not specified for paired mode alignment\n")
    config.set = F
  })

  if (!file.exists(fastq1)){
    cat("fastq1 file ", fastq1,
        " does not exist. Please, specify a valid file for paired mode alignment.\n")
    config.set = F
  }

  fastq2 = tryCatch({
    config.table["fastq2",1]
  }, warning = function(w) {
    cat("\"fastq2\" parameter not specified for paired mode alignment\n")
    config.set = F
  }, error = function(e) {
    cat("\"fastq2\" parameter not specified for paired mode alignment\n")
    config.set = F
  })

  if (!file.exists(fastq1)){
    cat("fastq2 file ", fastq1,
        " does not exist. Please, specify a valid file for paired mode alignment.\n")
    config.set = F
  }
}


base.index.exists <- function(prefix, suffix){
  filename = paste(prefix, suffix, sep="")
  if (!file.exists(filename)){
    cat("base index ", filename, " could not be found\n")
    return(F)
  } else {
    return(T)
  }
}

if (compute.base.cov){
  base.index.pathtoprefix = set.property(config.table, "base.index.pathtoprefix", "n/a")
  if(base.index.pathtoprefix == "n/a"){
    cat("compute.base.cov was equal to T, but base.index.pathtoprefix was not
specified.")
    base.index.set = F
  } else{
    base.index.set = F
    if (base.index.exists(base.index.pathtoprefix, ".1.bt2"))
      if (base.index.exists(base.index.pathtoprefix, ".2.bt2"))
```

```
        if (base.index.exists(base.index.pathtoprefix, ".3.bt2"))
          if (base.index.exists(base.index.pathtoprefix, ".4.bt2"))
            if (base.index.exists(base.index.pathtoprefix, ".rev.1.bt2"))
              if (base.index.exists(base.index.pathtoprefix, ".rev.2.bt2"))
                base.index.set = T
  }
  if (!base.index.set)
    config.set = F
  picard.samtofastq.jar = set.property(config.table, "picard.samtofastq.jar", "n/a")
  if(picard.samtofastq.jar == "n/a"){
    cat("compute.base.cov was equal to T, but base.index.pathtoprefix was not
specified.")
    config.set = F
  } else{
    if(!file.exists(picard.samtofastq.jar)){
      cat("Specified picard.samtofastq.jar file ", picard.samtofastq.jar, "does not exist\n")
      config.set = F
    }
  }
} else if(estimate.base.cov){
      base.cov = defaults$base.cov
} else {
  base.cov = set.property.double(config.table, "base.cov", defaults$base.cov)
}

output.dir = set.property(config.table, "output.dir", defaults$output.dir)
if(is.na(file.info(output.dir)$isdir || !file.info(output.dir)$isdir)){
  dc = dir.create(output.dir, showWarnings=F)
  if (!dc){
    cat("Warning: could not create output directory", output.dir,
        "results will be kept in default directory", defaults$output.dir, "\n")
    output.dir = defaults$output.dir
    dir.create(output.dir, showWarnings=F)
  }
}

scripts.dir = set.property(config.table, "scripts.dir", defaults$scripts.dir)
scripts.dir.set = F
#if ( file.info(scripts.dir)$isdir){
if (file.exists(file.path(scripts.dir, "pipeline.R")))
  if (file.exists(file.path(scripts.dir, "functions.R"))){
    scripts.dir.set = T
  }
#}
if (!scripts.dir.set){
  cat("scripts.dir does not exist or does not contain the required scripts\n")
  config.set = F
}

bowtie.build.path = set.property.executable(config.table, "bowtie.build.path",
defaults$bowtie.build.path)
if (!file.exists(bowtie.build.path)){
  cat("bowtie.build executable could not be found at ", bowtie.build.path, "\n")
  config.set = F
}
```

```
bowtie.align.path = set.property.executable(config.table, "bowtie.align.path",
defaults$bowtie.align.path)
if (!file.exists(bowtie.align.path)){
  cat("bowtie.align executable not be found at ", bowtie.align.path, "\n")
  config.set = F
}

samtools.path = set.property.executable(config.table, "samtools.path", defaults$samtools.path)
if (!file.exists(samtools.path)){
  cat("samtools.path", samtools.path, "does not exist\n")
  config.set = F
}


if (config.set) {

  cat("\nConfiguration successful\n")
  cat("Computel will execute with the following parameters:\n")
  cat("scripts.dir =", scripts.dir, "\n")
  cat("bowtie.build.path =", bowtie.build.path, "\n")
  cat("bowtie.align.path =", bowtie.align.path, "\n")
  cat("samtools.path =", samtools.path, "\n")
  if(compute.base.cov)
    cat("picard.samtofastq.jar =", picard.samtofastq.jar, "\n")
  cat("single =", single, "\n")
  if (single) {
    cat("fastqs =", fastqs, "\n")
  } else {
    cat("fastq1 =", fastq1, "\n")
    cat("fastq2 =", fastq2, "\n")
  }
  if(compressed){
    cat("file.compression =", file.compression, "\n")
  }
  cat("read.length = ", rl,"\n")
  cat("pattern = ", pattern,"\n")
  cat("num.haploid.chr = ", num.haploid.chr,"\n")
  cat ("min.seed =", min.seed, "\n")
  cat ("mode.local =", mode.local, "\n")
  cat("compute.base.cov =", compute.base.cov, "\n")
  if (compute.base.cov){
    cat("base.index.pathtoprefix =", base.index.pathtoprefix, "\n")
  } else if(estimate.base.cov){
    cat("base.cov =  will be estimated from the zipped files\n")
  } else {
    cat("base.cov = ", base.cov,"\n")
  }

  cat("output.dir= ", output.dir,"\n")
  cat ("num.proc=", num.proc, "\n")
  cat("quals=", quals, "\n")
  cat("ingore.err=", ignore.err, "\n")
  pipeline.R = file.path(scripts.dir, "pipeline.R")

}
```

Bijlage 5: Python scripts en R scripts

Bijlage 5A: 1000Genomes_overzicht.py

```
#Dit script gebruikt als input de 1000Genomes data van phase1 om hier informatie uit te extraheren.
from collections import Counter
from operator import itemgetter
import re
import numpy as np
import matplotlib.pyplot as plt

"""
import sys
import urllib

linenumber = 0
totalnumberofbases = 0
totalnumberofreads = 0

for link in sys.argv:
    testfile = urllib.URLopener()
    testfile.retrieve(link,"file.bas")

with open("file.bas",'r') as fp:
    for line in fp:
        if linenumber > 0:
            #print(line)
            splittedline = line.split('\t')
            print(splittedline)
            numberofbases = splittedline[7]
            numberofreads = splittedline[9]
            print("Controle: dit mag geen nummer zijn maar library :",splittedline[6])
            totalnumberofbases += float(numberofbases)
            totalnumberofreads += float(numberofreads)
        linenumber+=1

print ("Totalnumberofbases: ",totalnumberofbases)
print ("Totalnumberofreads: ",totalnumberofreads)
read_length = totalnumberofbases/totalnumberofreads
print ("Read length = ",read_length)
"""

i = 0
linenumber = 0
read_length = 0
numberofbases = 0
numberofreads = 0
individuallist = []
numberofbaseslist = []
numberofreadslist = []
totalindividuallist = []
totalnumberofbaseslist = []
totalnumberofreadslist = []
totalread_lengthlist = []
readsize = []
WANTEDREADSIZE = []
```

```
with open(r'/home/driesg/workplace/scripts/phase1.alignment.index.bas') as fp:
    for line in fp:
        if linenumber > 0:
            splittedline = line.split('\t')
            #if splittedline[0][8] == "m": #enkel werken met mapped reads
                #print(splittedline[0])
            individuallist.append(splittedline[0][0:7]) #NA20828
            numberofbaseslist.append(splittedline[7])
            numberofreadslist.append(splittedline[9])
        linenumber+=1

for i in range(len(individuallist)):
    if i == len(individuallist)-1: #einde van lijst
        numberofbases += int(numberofbaseslist[i])
        numberofreads += int(numberofreadslist[i])
        totalindividuallist.append(individuallist[i-1])
        totalnumberofbaseslist.append(numberofbases)
        totalnumberofreadslist.append(numberofreads)
    elif individuallist[i] == individuallist[i-1]:
        numberofbases += int(numberofbaseslist[i])
        numberofreads += int(numberofreadslist[i])
    elif i > 0: #Dus nieuw individu!
        #print(individuallist[i])
        totalindividuallist.append(individuallist[i-1])
        totalnumberofbaseslist.append(numberofbases)
        totalnumberofreadslist.append(numberofreads)
        numberofbases = 0 + int(numberofbaseslist[i])
        numberofreads = 0 + int(numberofreadslist[i])

#lijst met de totale read lengte aanmaken per individu.
for i in range(len(totalindividuallist)):
    read_length = float(totalnumberofbaseslist[i])/float(totalnumberofreadslist[i])
    totalread_lengthlist.append(read_length)

"""
#afzonderen van de reads die we willen
for position in range(len(totalread_lengthlist)):
    if str(totalread_lengthlist[position]) == "100.0" or str(totalread_lengthlist[position])
== "101.0" or str(totalread_lengthlist[position]) == "108.0":
        WANTEDREADSIZE.append(totalindividuallist[position])

print (WANTEDREADSIZE)
print len(WANTEDREADSIZE)
"""

#telt hoeveel een bepaalde readlengte voorkomt
readsizelibrary = Counter(totalread_lengthlist)
print (readsizelibrary)

#verwijderen van elementen met een voorkomen van minder dan X maal:
for element in list(readsizelibrary):
    if readsizelibrary[element] < 2:
        del readsizelibrary[element]

labels, values = zip(*readsizelibrary.items())
```

```
indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, color = '#2eb82e')
plt.xticks(indexes + width * 0.5, labels)
plt.xlabel('Read sizes',fontsize=14)
plt.ylabel('Frequentie',fontsize=14)
#plt.title('Read eigenschappen')
plt.show()
```

Bijlage 5B: 1000Genomes_overzicht_detail.py

```
#Dit script dient om van een bepaald individu de precieze populatie, sequencing toestel en sequencing center te achterhalen
#Input is "/home/driesg/workplace/scripts/20101123.sequence.index"

import re

rawdata = ['HG00100', ..., 'NA19470']

seqcenter = "test"
population = "test"
seqmachine = "test"
sequencestrategy = "low coverage"

for individu in rawdata:
    with open(r"/home/driesg/workplace/scripts/20101123.sequence.index") as fp:
        for line in fp:
            splittedline = line.split('\t')
            m = re.search('^data/%s' % individu, line)
            if m:
                sequencestrategy = splittedline[25]
                sequencestrategy = sequencestrategy.rstrip() #newline verwijderen
                if sequencestrategy == "low coverage":
                    seqcenter = splittedline[5]
                    population = splittedline[10]
                    seqmachine = splittedline[13]
                    #print (splittedline[25])
    #print (individu)
    #print (seqcenter)
    #print (population)
    #print (seqmachine)
```

Bijlage 5C: URLretriever.py

```
#Dit script dient om uit een gegeven aantal individuen de precieze URL te verkrijgen waar het gedownload kan worden.
#Deze URL kan dan meegegeven worden in een ander script die dit dan kan downloaden en bewerkingen op kan voeren.
#Gebruikt als input phase1.alignment.index.bas.

import sys
import re


rawdata = ['NA18615','NA18616','NA18617','NA18618','NA18619','NA18626','NA18627','NA18628','NA18630','NA18631','NA18634','NA18941','NA18987','NA19002','NA19009','NA19010','NA19074','NA19076','NA19077','NA19079','NA19080','NA19081','NA19082','NA19083','NA19084','NA19087','NA19088','NA19311','NA19312','NA19313','NA19429','NA19430','NA19431','NA19434','NA19435','NA19436','NA19437','NA19438','NA19439','NA19440','NA19443','NA19444','NA19445','NA19446','NA19448','NA19449','NA19451','NA19452','NA19453','NA19455','NA19456','NA19457','NA19461','NA19462','NA19463','NA19466','NA19467','NA19468','NA19469','NA19470']

rawdatanumber = 0
URLmappedindividu = "test"
URLunmappedindividu = "test"

log = open("./Output.txt", "w")
sys.stdout = log

for individu in rawdata:
    with open(r'/home/driesg/workplace/scripts/phase1.alignment.index.bas') as fp:
        for line in fp:
            splittedline = line.split('\t')
            m = re.search('^%s' % individu, line)
            if m:
                #print (splittedline[0])
                if splittedline[0][8] == "m":
                    URLmappedindividu = "\"ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1/data/" + individu + "/alignment/" + splittedline[0] + ".bam\","

                if splittedline[0][8] == "u":
                    URLunmappedindividu = "\"ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1/data/" + individu + "/alignment/" + splittedline[0] + ".bam\","


    print(URLmappedindividu)
    #print("\n")
    print(URLunmappedindividu)
```

Bijlage 5D: alignmentfile_to_puretelome_and_seqerror.py

```
#Dit script is om vanuit een gewone alignment file (vb HG00096.txt) enkel de pure telomeren + seq error over te houden.
#Er zal gefilterd worden waarbij de pure telomeren overblijven, de telomeren met 3 en minder SNPs maar GEEN inserten

import os #voor de .txt file te renamen naar een .sam file & fuseren
import sys #for output in een nieuwe file
import re
import subprocess
import glob

individu306collectie = ['HG00096', ..., 'NA20356']

#Individu declareren
for individu in individu306collectie:
    log = open("./../align/puurtelomeer_met_seqerror.txt", "w")
    sys.stdout = log
    individupath = "/home/driesg/workplace/align/tel.align." + str(individu) + ".txt"
    numberofreads = 0 #variabele dat aantal reads bijhoudt

    #Read lengte isoleren!
    with open(individupath) as fp:
        for line in fp:
            splittedline = line.split()
            sequentie = (splittedline[9])
            rl=len(sequentie)
            break

    #LN isoleren voor in de header
    LN=(rl+6-1)+(rl-12)

    #Base coverage berekenen!
    with open(r"/home/driesg/workplace/scripts/20101123.sequence.index") as fp:
        for line in fp:
            splittedline = line.split('\t')
            m = re.search('^data/%s' % individu, line)
            if m:
                if splittedline[25] == "low coverage\n":
                    numberofreads += int(splittedline[23])

    basecov=float(numberofreads)*float(rl)/float(3101804739)
    basecov = "%.3f" % round(basecov, 3)

    #Header aan de file plakken of dit zal foutmelding geven in SAM file!
    print ("@HD        VN:1.0  SO:unsorted")
    print("@SQ SN:>fwd:        LN:"+str(LN))
    print("@PG ID:bowtie2      PN:bowtie2      VN:2.1.0")

    with open(individupath) as fp:
        for line in fp:
            snpnumber = 0 #variabele dat aantal snps bijhoudt in mdcode
            splittedline = line.split()
            sequentie = (splittedline[9])
            cigarcode = (splittedline[5])
            n = re.search(r"MD:Z:([\w]+)",line)
```

```
            if n:
                mdcode = n.groups(1)[0]
                if 'A0' not in mdcode and 'C0' not in mdcode and 'G0' not in mdcode and 'T0' not in mdcode and 'N0' not in mdcode[0:-1]:
                    if 'I' not in cigarcode and 'D' not in cigarcode:
                        snpnumber = mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')+mdcode.count('N')
                        if snpnumber <= 3:
                            line=line.strip()
                            print(line)

    infostring = "mv ./../align/puurtelomeer_met_seqerror.txt ./../align/puurtelomeer_met_seqerror_metheader_"+str(individu)+"_"+str(rl)+"_"+str(basecov)+".sam"
    os.system(infostring)
```

Bijlage 5E: alignmentfile_to_absolute_pseudotel.py

```
#Dit script is om vanuit een gewone alignment file (vb HG00096.txt) enkel de pure pseudotel over te houden.
#Dus de pure telomeren zullen worden weggelaten
#Reads met #SNPs < 4 weggelaten
#Reads met N's weggelaten
#Reads waarin er mutaties naast elkaar zijn zijn subtelomeren en worden weggelaten
#Er zal gefilterd worden waarbij N's van subtelomeren niet geaccepteerd worden, en de pseudotels geen 2 naast mekaar voorkomende SNP's accepteren

import os #voor de .txt file te renamen naar een .sam file & fuseren
import sys #for output in een nieuwe file
import re
import subprocess
import glob

individu306collectie = ['HG00096', ..., 'NA20356']

#Individu declareren
for individu in individu306collectie:
    log = open("./../align/absolute_pseudotelomeer.txt", "w")
    sys.stdout = log
    individupath = "/home/driesg/workplace/align/tel.align." + str(individu) + ".txt"
    numberofreads = 0 #variabele dat aantal reads bijhoudt

    #Read lengte isoleren!
    with open(individupath) as fp:
        for line in fp:
            splittedline = line.split()
            sequentie = (splittedline[9])
            rl=len(sequentie)
            break

    #LN isoleren voor in de header
    LN=(rl+6-1)+(rl-12)

    #Base coverage berekenen!
    with open(r"/home/driesg/workplace/scripts/20101123.sequence.index") as fp:
        for line in fp:
            splittedline = line.split('\t')
            m = re.search('^data/%s' % individu, line)
            if m:
                if splittedline[25] == "low coverage\n":
                    numberofreads += int(splittedline[23])

    basecov=float(numberofreads)*float(rl)/float(3101804739)
    basecov = "%.3f" % round(basecov, 3)


    #Header aan de file plakken of dit zal foutmelding geven in SAM file!
    print ("@HD        VN:1.0  SO:unsorted")
    print("@SQ SN:>fwd:        LN:"+str(LN))
    print("@PG ID:bowtie2      PN:bowtie2      VN:2.1.0")

    with open(individupath) as fp:
        for line in fp:
            splittedline = line.split()
```

```
            sequentie = (splittedline[9])
            cigarcode = (splittedline[5])
            n = re.search(r"MD:Z:([\w]+)",line)
            if n:
                mdcode = n.groups(1)[0]
                snpnumber =
mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')+mdcode.count
('N')
                if 'A0' not in mdcode and 'C0' not in mdcode and 'G0' not in mdcode and 'T0'
not in mdcode and 'N0' not in mdcode[0:-1]: #geen 10,20 of 30 meetellen!!
                    #geen sequentie-errors of pure telomeren:
                    #reads die <4 SNPs hebben maar indels zijn ook pseudo!
                    if snpnumber > 3 or 'I' in cigarcode or 'D' in cigarcode:
                        line=line.strip()
                        print(line)


    infostring = "mv ./../align/absolute_pseudotelomeer.txt
./../align/absolute_pseudotelomeer_metheader_"+str(individu)+"_"+str(rl)+"_"+str(basecov)+".sam
"
    os.system(infostring)
```

Bijlage 5F: alignmentfile_to_absolute_subtel.py

```
#Dit script is om vanuit een gewone alignment file (vb HG00096.txt) enkel de hybride telomeren over te houden.
#Reads waarin er mutaties naast elkaar zijn subtelomeren
#OF
#Reads waar er "No" in wordt gedetecteerd zijn hybride telomeren

import os #voor de .txt file te renamen naar een .sam file & fuseren
import sys #for output in een nieuwe file
import re
import subprocess
import glob

individu306collectie = ['HG00096', ... , 'NA20356']

#Individu declareren
for individu in individu306collectie:
    log = open("./../align/absolute_subtelomeer.txt", "w")
    sys.stdout = log
    individupath = "/home/driesg/workplace/align/tel.align." + str(individu) + ".txt"
    numberofreads = 0 #variabele dat aantal reads bijhoudt

    #Read lengte isoleren!
    with open(individupath) as fp:
        for line in fp:
            splittedline = line.split()
            sequentie = (splittedline[9])
            rl=len(sequentie)
            break

    #LN isoleren voor in de header
    LN=(rl+6-1)+(rl-12)

    #Base coverage berekenen!
    with open(r"/home/driesg/workplace/scripts/20101123.sequence.index") as fp:
        for line in fp:
            splittedline = line.split('\t')
            m = re.search('^data/%s' % individu, line)
            if m:
                if splittedline[25] == "low coverage\n":
                    numberofreads += int(splittedline[23])

    basecov=float(numberofreads)*float(rl)/float(3101804739)
    basecov = "%.3f" % round(basecov, 3)


    #Header aan de file plakken of dit zal foutmelding geven in SAM file!
    print ("@HD        VN:1.0  SO:unsorted")
    print("@SQ SN:>fwd:        LN:"+str(LN))
    print("@PG ID:bowtie2      PN:bowtie2      VN:2.1.0")

    with open(individupath) as fp:
        for line in fp:
            splittedline = line.split()
            sequentie = (splittedline[9])
            cigarcode = (splittedline[5])
```

```
            n = re.search(r"MD:Z:([\w]+)",line)
            if n:
                mdcode = n.groups(1)[0]
                if 'A0' in mdcode or 'C0' in mdcode or 'G0' in mdcode or 'T0' in mdcode or 'N0'
in mdcode[0:-1]:
                    line=line.strip()
                    print(line)

    infostring                    =                    "mv                    ./../align/absolute_subtelomeer.txt
./../align/absolute_subtelomeer_metheader_"+str(individu)+"_"+str(rl)+"_"+str(basecov)+".sam"
    os.system(infostring)
```

Bijlage 5G: alignment_length_calculator.R

```
# Dit script zal de vervanger zijn van het Computelscript
# Met als enige doel om vanuit alignmentfile_to_XXX.py een telomeerlengte te berekenen

call.cmd <- function(command){
  OS.name = tolower(Sys.info()["sysname"])
  if (OS.name == "windows")
    shell(command, wait = T)
  else
    system(paste("bash -c", "'", command, "'"), wait = T)
}

alignmentpaths<-c('puurtelomeer_met_seqerror_metheader_HG00731_101_6.772.sam')

path = alignmentpaths[1]

#ALL
individu = "HG00731" #zonder /home/driesg/workplace/align/
rl = as.integer(101)
base.cov = as.double(6.772)
num.haploid.chr=23
pl=6

#Declaren van allerlei variabelen
samtools.path="/home/driesg/computel/samtools-0.1.19-linux/samtools"
#sam.file="/home/driesg/workplace/align/puurtelomeer_metheader.sam"
#sam.file=path
sam.file=paste("/home/driesg/workplace/align/",path,sep="")
bam.file="/home/driesg/workplace/align/puurtelomeer_metheader.bam"
sorted.file="/home/driesg/workplace/align/puurtelomeer_metheader_sorted"
coverage.file="/home/driesg/workplace/align/puurtelomeer_metheader_sorted_coverage.txt"

#Convert sam to bam
####################
samtobam = paste(samtools.path, "view -bSh", sam.file, ">", bam.file)
#cat("\nConverting sam file", sam.file,"to bam file", bam.file, "...\n")
call.cmd(samtobam)

#Sort bam
####################
sort = paste(samtools.path, "sort", bam.file, sorted.file)
#cat("\nSorting bam file", bam.file, "to", sorted.file, "\n")
call.cmd(sort)

#Calculate depth
####################
sorted.file = paste(sorted.file, ".bam", sep="")
depth = paste(samtools.path, "depth", sorted.file, ">", coverage.file)
#cat("\nCalculating depth of coverage from bam file", sorted.file, "to", coverage.file, "\n")
call.cmd(depth)

#Calculate tel length
####################
#cat("estimating telomere length \n")
table.tel = read.table(file=coverage.file,col.names=c("position","orientation", "depth"));
tl = length(table.tel$depth)
range = c(1:(rl+pl-1))
```

```
depth.tel = table.tel$depth[range]/2/base.cov
mean.length.rl.pl = mean(depth.tel)*(rl+pl-1)
tel.length = mean.length.rl.pl/num.haploid.chr
cat(tel.length)
```

Bijlage 5H: ratio_snp_indel.py

```
#INDIVIDUGEWIJS
#Dit script om te kijken of bepaalde SNP's of indels met een bepaalde frequentie samenkomen
#Eerst statistisch weergeven van aantal SNP's, indels en deleties.

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt

outputlist = []

#variabelen die bijhouden hoeveel inserts, deleties en snps's er in totaal voorkomen
global_insert_count = 0
global_deletion_count = 0
global_snp_count = 0

with open(r'/home/driesg/workplace/align/tel.align.HG01495.txt') as fp:
    for line in fp:

        #opsplitsing tussen 2 counts: sinds inserten in de cigarcode niet worden bijgehouden in de MDcode moet dit gecorrigeerd worden
        cigarcodecount = 0 #variable dat positie in de cigarcode bijhoudt
        mdcodecount = 0 #variabele dat positie in de mdcode bijhoudt

        insert_count=0 #variable dat aantal geinsereerde nucleotiden bijhoudt
        deletion_count=0 #variabele dat aantal deleted nt bijhoudt
        splittedline = line.split()
        cigarcode = (splittedline[5])
        sequentie = (splittedline[9])
        read_length = len(sequentie)
        cigarcodelist = re.split("([\d]+[\D])",cigarcode)
        cigarcodelist = [element for element in cigarcodelist if element.strip()]

        #Elk element in de lijst overlopen.
        for element in cigarcodelist:
            if 'I' in element:
                insert_count+=int(element[0])
                global_insert_count+=int(element[0])

            if 'D' in element:
                deletion_count+=int(element[0])
                global_deletion_count+=int(element[0])

        n = re.search(r"MD:Z:([\w]+)",line)
        if n:
            mdcode = n.groups(1)[0]
            snpnumber = mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')+mdcode.count('N')
            global_snp_count += snpnumber

        #print('Aantal SNPs zijn ',snpnumber,'inserts :',insert_count,'deletions :',deletion_count)
        output = '%sSNPs %sI %sD' % (snpnumber, insert_count, deletion_count)
```

```
        outputlist.append(output)

        #if interesse:
        if output == '1SNPs 1I 0D':
            print (sequentie)
            print (cigarcode)
            print (mdcode)

            mdcodelist = re.split("([\d]+[\D])",mdcode)
            mdcodelist = [x for x in mdcodelist if "A" in x or "C" in x or "T" in x or "G" in x]
            for element in mdcodelist:
                snpref = re.sub('[\d]', '', element)
                print ('SNP:',snpref)

            for element in cigarcodelist:
                #Als M is (=match of mismatch, nummer van basen in variabele steken
                if 'M' in element:
                    element = element.replace("M","")
                    cigarcodecount += int(element)
                    print (cigarcodecount)
                elif 'I' in element:
                    element = element.replace("I","")
                    cigarcodecount += int(element)
                    print ('Hier een extra insert: ',sequentie[cigarcodecount-
int(element):cigarcodecount])
                    print (cigarcodecount)
                elif 'D' in element:
                    element = element.replace("D","")
                    cigarcodecount += int(element)
                    print (cigarcodecount)

counteroutputlist = Counter(outputlist)

print('SNPs:',global_snp_count,'inserts:',global_insert_count,'deletions:',global_deletion_count)

#verwijderen van elementen met een voorkomen van minder dan X maal:
#for element in list(counteroutputlist):
    #if counteroutputlist[element] < 50:
        #del counteroutputlist[element]

c = counteroutputlist.items()
c.sort(key=itemgetter(1))
labels, values = zip(*c)

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('Afwijkingen')
plt.ylabel('Frequentie')
plt.title('Afwijking per read')
plt.show()
```

Bijlage 5I: ratio_snp_indel_versie2.py

```
#POPULATIE GEWIJS!!!!
# Ratio bepalen van INDELS / SNPS (zonder N) voor elk individu.
# (aan de hand van CIGARcode en MDcode)
#Dit in histogram weergeven.

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import sys
import glob

"""
#Dit script berekent de ratio van aantal SNP's / aantal indels PER individu en geeft dit terug.
ratio_snp_indel = []

with open(r'/home/driesg/workplace/align/tel.align.NA19728.txt') as fp:
    for line in fp:
        snpnumber = 0
        indelnumber = 0
        ratio=0
        splittedline = line.split()

        #Opslaan van de indels (insertie + deletie)
        #Focus op aantal events! Dus niet aantal nucleotiden die geinsereerd zijn!
        cigarcode = (splittedline[5])
        cigarcodelist = re.split("([\d]+[\D])",cigarcode)
        cigarcodelist = [element for element in cigarcodelist if element.strip()]
        for element in cigarcodelist:
            if 'I' in element or 'D' in element:
                indelnumber+=1

        #Opslaan van de SNP's
        #Zonder aantal N's!
        n = re.search(r"MD:Z:([\w]+)",line)
        if n:
            mdcode = n.groups(1)[0]
            snpnumber = mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')

        #Berekenen van de ratio en in een list steken
        if indelnumber == 0:
            ratio = 0
        else:
            ratio=snpnumber/indelnumber
        ratio_snp_indel.append(ratio)

counter_ratio_snp_indel = Counter(ratio_snp_indel)

c = counter_ratio_snp_indel.items()
c.sort(key=itemgetter(1))
labels, values = zip(*c)
```

```
indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('Ratio')
plt.ylabel('Frequentie van voorkomen')
plt.title('Frequentie van SNP/indel-ratios')
plt.show()
"""

##########################################################################
##########################################
##########################################################################
##########################################
##########################################################################
##########################################

"""
#Dit script berekent SNP/INDEL ratio als gemiddelde voor een individu!
#Geeft dit tenslotte weer in een histogram voor alle individuen

path = '/home/driesg/workplace/align/tel.align.*.txt'
files = glob.glob(path)
ratio_snp_indel = []

aantalindividuen=0
#head_lines = 0

def my_round(x):
    return round(x*2)/2

for name in files:
    aantalindividuen +=1
    #head_lines +=1
    #if head_lines > 10:
        #break
    print("Aantal individuen aanwezig: "+str(aantalindividuen))
    snpnumber = 0
    indelnumber = 0
    ratio=0
    with open(name) as f: # No need to specify 'r': this is the default.
        for line in f:
            splittedline = line.split()

            #Opslaan van de indels (insertie + deletie)
            #Focus op aantal events! Dus niet aantal nucleotiden die geinsereerd zijn!
            cigarcode = (splittedline[5])
            cigarcodelist = re.split("([\d]+[\D])",cigarcode)
            cigarcodelist = [element for element in cigarcodelist if element.strip()]
            for element in cigarcodelist:
                if 'I' in element or 'D' in element:
                    indelnumber+=1

            #Opslaan van de SNP's
            #Zonder aantal N's!
            n = re.search(r"MD:Z:([\w]+)",line)
```

```
            if n:
                mdcode = n.groups(1)[0]
                snpnumber                                                    +=
mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')

        #Na alle lijnen van individu bekeken te hebben:
        #Berekenen van de ratio en in een list steken
        if indelnumber == 0:
            ratio = 0
        else:
            ratio=float(snpnumber)/float(indelnumber)
        ratio=my_round(ratio)
        ratio_snp_indel.append(ratio)
        print(ratio)


#Na alle individuen bekeken te hebben:
counter_ratio_snp_indel = Counter(ratio_snp_indel)

i = 0
#Nulwaarden toevoegen aan de ratioreeks voor barplot zodat mooi wordt weergegeven
while i <= 100:
    if counter_ratio_snp_indel[i]:
        i=i+0.5
    else:
        counter_ratio_snp_indel[i]=0
        i=i+0.5

c = counter_ratio_snp_indel.items()
c.sort(key=itemgetter(0))
#c.sort(key=itemgetter(1))
labels, values = zip(*c)

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')

plt.xticks(indexes + width * 0.5, labels, rotation='vertical', fontsize = 8)
plt.xlim(0,180)
plt.xlabel('Ratio')
plt.ylabel('Frequentie van voorkomen')
plt.title('Frequentie van SNP/indel-ratios')
plt.show()

"""

##########################################################################
##################################
##########################################################################
##################################
##########################################################################
##################################

#Om te kijken of 100/101 reads en 108 reads gelijk zijn van indel/snp's.

path = '/home/driesg/workplace/align/tel.align.*.txt'
files = glob.glob(path)
```

```
ratio_snp_indel = []

reads100_101 = ['HG00096', ... ,'NA19664']

aantalindividuen=0
#head_lines = 0

def my_round(x):
    return round(x*2)/2

for name in files:
    individu = name[39:46] #HG00096, NA19278 etc.

    if individu in reads100_101:
        aantalindividuen +=1
        #head_lines +=1
        #if head_lines > 10:
            #break
        #print("Aantal individuen aanwezig: "+str(aantalindividuen))
        snpnumber = 0
        indelnumber = 0
        ratio=0
        with open(name) as f: # No need to specify 'r': this is the default.
            for line in f:
                splittedline = line.split()

                #Opslaan van de indels (insertie + deletie)
                #Focus op aantal events! Dus niet aantal nucleotiden die geinsereerd zijn!
                cigarcode = (splittedline[5])
                cigarcodelist = re.split("([\d]+[\D])",cigarcode)
                cigarcodelist = [element for element in cigarcodelist if element.strip()]
                for element in cigarcodelist:
                    if 'I' in element or 'D' in element:
                        indelnumber+=1

                #Opslaan van de SNP's
                #Zonder aantal N's!
                n = re.search(r"MD:Z:([\w]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    snpnumber                                                    +=
mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')

            #Na alle lijnen van individu bekeken te hebben:
            #Berekenen van de ratio en in een list steken
            if indelnumber == 0:
                ratio = 0
            else:
                ratio=float(snpnumber)/float(indelnumber)
            ratio=my_round(ratio)
            ratio_snp_indel.append(ratio)
            #print(ratio)
            if(int(ratio)>54):
                print(individu)
                print(ratio)
                print(snpnumber)
                print(indelnumber)
```

```
#Na alle individuen bekeken te hebben:
counter_ratio_snp_indel = Counter(ratio_snp_indel)
print (counter_ratio_snp_indel)

i = 0
#Nulwaarden toevoegen aan de ratioreeks voor barplot zodat mooi wordt weergegeven
while i <= 100:
    if counter_ratio_snp_indel[i]:
        i=i+0.5
    else:
        counter_ratio_snp_indel[i]=0
        i=i+0.5

print (counter_ratio_snp_indel)
c = counter_ratio_snp_indel.items()
c.sort(key=itemgetter(0))
#c.sort(key=itemgetter(1))
labels, values = zip(*c)

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')

xticks=[20,40,60,80,100,120,140,160,180]
labellist = list(labels[i] for i in [20,40,60,80,100,120,140,160,180])

plt.xticks (xticks, labellist, rotation='vertical', fontsize = 12)
plt.xlim(0,182)
plt.xlabel('Ratio SNP/indel')
plt.ylabel('Frequentie van voorkomen')
plt.show()
```

Bijlage 5J: ratio_snp_indel_versie3.py

```
#INDIVIDU GEWIJS!
#Dit script dient om te kijken of het (aantal SNP's met INDELS) significant verschilt met (aantal SNP's ZONDER INDELS)
#Hierna dan Wilcoxon test om te vergelijken!
#De reads met subtelomeren worden verwijderd: enkel pure telomeren en pseudotelomeren beschouwen

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt

snplist_zonder_indel=[]
snplist_met_indel=[]

with open(r'/home/driesg/workplace/align/tel.align.NA19010.txt') as fp:
    for line in fp:
        snpnumber = 0
        splittedline = line.split()
        cigarcode = (splittedline[5])
        sequentie = (splittedline[9])
        n = re.search(r"MD:Z:([\w]+)",line)
        if n:
            mdcode = n.groups(1)[0]
            #print (sequentie)
            #print (cigarcode)
            #print (mdcode+"\n")

            #reads met subtelomeren worden niet beschouwd
            if 'N' not in mdcode:
                cigarcodelist = re.split("([\d]+[\D])",cigarcode)
                cigarcodelist = [element for element in cigarcodelist if element.strip()]
                snpnumber =
mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')
                for element in cigarcodelist:
                    #Opsplitsing
                    #Reads met INDEL aanwezig!!!
                    if 'I' in element or 'D' in element:
                        snplist_met_indel.append(snpnumber)

                    #Reads zonder INDEL aanwezig!
                    else:
                        snplist_zonder_indel.append(snpnumber)

counter_snplist_met_indel = Counter(snplist_met_indel)
counter_snplist_zonder_indel = Counter(snplist_zonder_indel)

print (counter_snplist_met_indel)
print (counter_snplist_zonder_indel)

c = counter_snplist_met_indel.items()
c.sort(key=itemgetter(1))
labels1, values1 = zip(*c)
```

```
d = counter_snplist_zonder_indel.items()
d.sort(key=itemgetter(1))
labels2, values2 = zip(*d)

width = 0.4
indexes1 = np.arange(len(labels1))
indexes2 = np.arange(len(labels2))
fig, ax = plt.subplots()

rects1 = ax.bar(labels2, values2, width, facecolor='blue')
rects2 = ax.bar(labels1, values1, width, facecolor='green')
ax.set_yscale('log')
ax.set_xticks(labels2)
ax.legend((rects1[0], rects2[0]), ('SNPs zonder indels', 'SNPs met indels'))
plt.xlabel('Aantal SNPs')
plt.ylabel('log(Aantal reads)')
plt.show()
```

Bijlage 5K: telomeren_SNP.py

```
#Geeft voor de algemene populatie van 100/101 of 108reads een distributie over SNP distributie bv. A>C

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import subprocess
import glob

individuen100_101 = ['HG00096',..., 'NA19664']
individuen108 = ['HG00100', ...,  'NA19470']

path = '/home/driesg/workplace/align/absolute_subtelomeer_*.txt'
files = glob.glob(path)
aantalindividuen = 0

#break_lines=0

snplist = []
countersnplist_temp = Counter()
countersnplist_perm = Counter()

for name in files:
    #tel.align.txt
    #individu = name[39:46] #HG00096, NA19278 etc.
    #absolute_pseudotelomeer.txt
    #individu = name[53:60]
    #puurtel+seqerror.txt
    #individu = name[55:62]
    #absolute_subtelomeer.txt
    individu = name[50:57]
    #print (individu)
    #if individu in individuen108:
    if individu in individuen100_101:
        print (individu)
        #if break_lines > 5:
            #break
        #break_lines+=1

        with open(name) as f:
            for line in f:
                basecount = 0 #positie bij overlopen van sequentiestring
                splittedline = line.split()
                cigarcode = (splittedline[5])
                sequentie = (splittedline[9])
                #print(line)
                n = re.search(r"MD:Z:([\S]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    mdcodelist = re.split("([\d]+[\D]+)",mdcode) #Hier!!!
                    mdcodelist = [x for x in mdcodelist if "A" in x or "C" in x or "T" in x or "G" in x]
                    cigarcodelist = re.split("([\d]+[\D])",cigarcode)
```

```
                cigarcodelist = [element for element in cigarcodelist if element.strip()]
                cigarposition = 0 #variabele om de plaats in cigarstring bij te houden
                basecount = 0
                #print ('mdcodelistbefore',mdcodelist)
                #print (cigarcodelist)

                #corrigeren van de mdcodelist bij aanwezigheid van insert: alles moet opschuiven op de juiste plaats!
                for element1 in cigarcodelist:
                    numberofindel = int(re.sub('[\D]', '', element1))
                    cigarposition += int(re.sub('[\D]', '', element1))
                    #print ('cigarposition:',cigarposition)
                    if 'I' in element1:
                        #alles van mdcodes vermeerderen dat > cigarposition!
                        mdpositiontotal = 0 #variabele om plaats in mdcode bij te houden
                        i = 0 #plaats in mdcodelist: [0] of 1, 2
                        for element2 in mdcodelist:
                            mdsnp = re.sub('[\d]','', element2) #A, C, T of G
                            mdposition = int(re.sub('[\D]', '', element2)) #hoeveel plaatsen tussen SNP's bv. 0A4T
                            mdpositiontotal += int(re.sub('[\D]','', element2)) #plaats in totaal 8,12,23
                            mdpositiontotal+=1
                            #print ('mdposition',mdpositiontotal)
                            if (mdpositiontotal+numberofindel) > cigarposition:
                                mdposition=mdposition+numberofindel
                                var=str(mdposition)+str(mdsnp)
                                mdcodelist[i]=var
                                break #na toevoegen van extra nt, uit de for-lus breken
                            i+=1

                #print ('mdcodelistafter',mdcodelist)
                for element in mdcodelist:
                    snpref = re.sub('[\d]', '', element)
                    snppos = re.sub('[\D]', '', element)
                    basecount += int(snppos)
                    #print (basecount)
                    if '^' in element: #corrigeren voor deletie
                        basecount-=1
                    #print('basecount:',basecount)
                    #print('sequentie',sequentie)
                    snpread = sequentie[basecount]
                    #print ("Oude base van referentie : ",snpref, snppos, "Nieuwe base van read : ",snpread)
                    basecount+=1 #telkens bij 1SNP staat de letter ook voor 1 plaats, maar dit wordt niet meegerekend in totale nummering
                    snp = snpref + ">" + snpread
                    if '^' not in element:
                        #print (snp)
                        snplist.append(snp)

            countersnplist_temp = Counter(snplist)
            countersnplist_perm = countersnplist_perm + countersnplist_temp
            print (countersnplist_perm)
            snplist = []
            countersnplist_temp = []

c = countersnplist_perm.items()
```

```
c.sort(key=itemgetter(1))
labels, values = zip(*c)

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('Aantal SNPs')
plt.ylabel('Frequentie van voorkomen')
#plt.title('Afwijking per read')
plt.show()
```

Bijlage 5L: telomeren_SNP_withoutweight.py

```
#Geeft voor de algemene populatie van 100/101 of 108reads een distributie over SNP distributie bv. A>C
#Dit is niet-weighted! Dus voor elk individu wordt aantal snp's opgeteld. Na het tellen van de snps, wordt de verhouding berekend. Dus elke individu heeft ongeacht van de grootte van de file eenzelfde weight.

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import subprocess
import glob

individuen100_101 = ['HG00096', ..., 'NA19664']
individuen108 = ['HG00100', ..., 'NA19470']
ASW100 = ['NA19985', ... , 'NA20356']
CLM100 = ['HG01112', ..., 'HG01492']
GBR100 = ['HG00096',...,'HG00233']
IBS100 = ['HG01515',...,'HG01522']
LWK100 = ['NA19318',...,'NA19428']
MXL100 = ['NA19728',...,'NA19664']
PUR100 = ['HG01097',...,'HG01204']
YRI100 = ['NA18917',...,'NA19223']
CEU108 = ['NA12272',...,'NA12718']
CHB108 = ['NA18614',...,'NA18634']
CHS108 = ['HG00403',...,'HG00635']
FIN108 = ['HG00171',...,'HG00377']
GBR108 = ['HG00100',...,'HG00265']
JPT108 = ['NA18941',...,'NA19088']
LWK108 = ['NA19311',...,'NA19470']
PUR108 = ['HG00553',...,'HG00640']

path = '/home/driesg/workplace/align/absolute_subtelomeer_*.txt'
#path = '/home/driesg/workplace/align/puurtelomeer_met_seqerror_*.txt'
#path = '/home/driesg/workplace/align/absolute_pseudotelomeer_*.txt'

files = glob.glob(path)
aantalindividuen = 0

#break_lines=0

snplist = []
countersnplist_temp = Counter()
countersnplist_perm = Counter()
aantalindividuen = 0

for name in files:
    #tel.align.txt
    #individu = name[39:46] #HG00096, NA19278 etc.
    #absolute_pseudotelomeer.txt
    #individu = name[53:60]
    #puurtel+seqerror.txt
    #individu = name[55:62]
    #absolute_subtelomeer.txt
```

```
    individu = name[50:57]
    #print (individu)
    if individu in PUR108:
    #if individu in individuen108:
        print (individu)
        totaalaantalsnps=0
        aantalindividuen+=1
        #if break_lines > 0:
            #break
        #break_lines+=1

        with open(name) as f:
            for line in f:
                basecount = 0 #positie bij overlopen van sequentiestring
                splittedline = line.split()
                cigarcode = (splittedline[5])
                sequentie = (splittedline[9])
                #print(line)
                n = re.search(r"MD:Z:([\S]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    mdcodelist = re.split("([\d]+[\D]+)",mdcode) #Hier!!!
                    mdcodelist = [x for x in mdcodelist if "A" in x or "C" in x or "T" in x or "G" in
x]
                    cigarcodelist = re.split("([\d]+[\D])",cigarcode)
                    cigarcodelist = [element for element in cigarcodelist if element.strip()]
                    cigarposition = 0 #variabele om de plaats in cigarstring bij te houden
                    basecount = 0
                    #print ('mdcodelistbefore',mdcodelist)
                    #print (cigarcodelist)

                    #corrigeren van de mdcodelist bij aanwezigheid van insert: alles moet opschuiven op
de juiste plaats!
                    for element1 in cigarcodelist:
                        numberofindel = int(re.sub('[\D]', '', element1))
                        cigarposition += int(re.sub('[\D]', '', element1))
                        #print ('cigarposition:',cigarposition)
                        if 'I' in element1:
                            #alles van mdcodes vermeerderen dat > cigarposition!
                            mdpositiontotal = 0 #variabele om plaats in mdcode bij te houden
                            i = 0 #plaats in mdcodelist: [0] of 1, 2
                            for element2 in mdcodelist:
                                mdsnp = re.sub('[\d]','', element2) #A, C, T of G
                                mdposition = int(re.sub('[\D]', '', element2)) #hoeveel plaatsen tussen
SNP's bv. 0A4T
                                mdpositiontotal += int(re.sub('[\D]','', element2)) #plaats in totaal
8,12,23
                                mdpositiontotal+=1
                                #print ('mdposition',mdpositiontotal)
                                if (mdpositiontotal+numberofindel) > cigarposition:
                                    mdposition=mdposition+numberofindel
                                    var=str(mdposition)+str(mdsnp)
                                    mdcodelist[i]=var
                                    break #na toevoegen van extra nt, uit de for-lus breken
                                i+=1

                    #print ('mdcodelistafter',mdcodelist)
                    for element in mdcodelist:
```

```
                    snpref = re.sub('[\d]', '', element)
                    snppos = re.sub('[\D]', '', element)
                    basecount += int(snppos)
                    #print (basecount)
                    if '^' in element: #corrigeren voor deletie
                        basecount-=1
                    #print('basecount:',basecount)
                    #print('sequentie',sequentie)
                    snpread = sequentie[basecount]
                    #print ("Oude base van referentie : ",snpref, snppos, "Nieuwe base van read : ",snpread)
                    basecount+=1 #telkens bij 1SNP staat de letter ook voor 1 plaats, maar dit wordt niet meegerekend in totale nummering
                    snp = snpref + ">" + snpread
                    if '^' not in element:
                        #print (snp)
                        snplist.append(snp)
                        totaalaantalsnps+=1

            countersnplist_temp = Counter(snplist)

            #omzetten naar percentages
            for element in list(countersnplist_temp):
                countersnplist_temp[element] = float(countersnplist_temp[element])/totaalaantalsnps

            countersnplist_perm = countersnplist_perm + countersnplist_temp
            print (countersnplist_perm)
            snplist = []
            countersnplist_temp = []

#percentages corrigeren voor aantal individuen:
for element in list(countersnplist_perm):
    countersnplist_perm[element] = float(countersnplist_perm[element])/aantalindividuen

print (countersnplist_perm)

for element in list(countersnplist_perm):
    var = str(element) + "\t" + str(countersnplist_perm[element])
    print (var)

c = countersnplist_perm.items()
c.sort(key=itemgetter(1))
labels, values = zip(*c)

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('Aantal SNPs')
plt.ylabel('Frequentie van voorkomen')
#plt.title('Afwijking per read')
plt.show()
```

Bijlage 5M: telomeren_SNP_withoutweight_obsexp.py

```
#Geeft voor de algemene populatie van 100/101 of 108reads een distributie over SNP distributie bv. A>C
#Dit is niet-weighted! Dus voor elk individu wordt aantal snp's opgeteld. Na het tellen van de snps, wordt de verhouding berekend. Dus elke individu heeft ongeacht van de grootte van de file eenzelfde weight. De verhouding die hier berekend wordt is aantal SNP's / totaal aantal basen!!!

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import subprocess
import glob

individuen100_101 = ['HG00096', ..., 'NA19664']
individuen108 = ['HG00100', ..., 'NA19470']
ASW100 = ['NA19985',...,'NA20356']
CLM100 = ['HG01112',...,'HG01492']
GBR100 = ['HG00096',...,'HG00233']
IBS100 = ['HG01515',...,'HG01522']
LWK100 = ['NA19318',...,'NA19428']
MXL100 = ['NA19728',...,'NA19664']
PUR100 = ['HG01097',...,'HG01204']
YRI100 = ['NA18917',...,'NA19223']
CEU108 = ['NA12272',...,'NA12718']
CHB108 = ['NA18614',...,'NA18634']
CHS108 = ['HG00403',...,'HG00635']
FIN108 = ['HG00171',...,'HG00377']
GBR108 = ['HG00100',...,'HG00265']
JPT108 = ['NA18941',...,'NA19088']
LWK108 = ['NA19311',...,'NA19470']
PUR108 = ['HG00553',...,'HG00640']

#path = '/home/driesg/workplace/align/absolute_subtelomeer_*.txt'
#path = '/home/driesg/workplace/align/puurtelomeer_met_seqerror_*.txt'
#path = '/home/driesg/workplace/align/absolute_pseudotelomeer_*.txt'
path = '/home/driesg/workplace/align/tel.align.*.txt'

files = glob.glob(path)
aantalindividuen = 0

#break_lines=0

snplist = []
countersnplist_temp = Counter()
countersnplist_perm = Counter()
aantalindividuen = 0

for name in files:
    #tel.align.txt
    individu = name[39:46] #HG00096, NA19278 etc.
    #absolute_pseudotelomeer.txt
    #individu = name[53:60]
    #puurtel+seqerror.txt
    #individu = name[55:62]
```

```
    #absolute_subtelomeer.txt
    #individu = name[50:57]
    #print (individu)
    #if individu in individuen100_101:
    if individu in individuen108:
        print (individu)
        totaalaantalsnps=0
        totaalaantalbasen = 0
        aantalindividuen+=1
        #if break_lines > 0:
            #break
        #break_lines+=1

        with open(name) as f:
            for line in f:
                basecount = 0 #positie bij overlopen van sequentiestring
                splittedline = line.split()
                cigarcode = (splittedline[5])
                sequentie = (splittedline[9])
                totaalaantalbasen+=len(sequentie)
                #print(line)
                n = re.search(r"MD:Z:([\S]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    mdcodelist = re.split("([\d]+[\D]+)",mdcode) #Hier!!!
                    mdcodelist = [x for x in mdcodelist if "A" in x or "C" in x or "T" in x or "G" in
x]
                    cigarcodelist = re.split("([\d]+[\D])",cigarcode)
                    cigarcodelist = [element for element in cigarcodelist if element.strip()]
                    cigarposition = 0 #variabele om de plaats in cigarstring bij te houden
                    basecount = 0
                    #print ('mdcodelistbefore',mdcodelist)
                    #print (cigarcodelist)

                    #corrigeren van de mdcodelist bij aanwezigheid van insert: alles moet opschuiven op
de juiste plaats!
                    for element1 in cigarcodelist:
                        numberofindel = int(re.sub('[\D]', '', element1))
                        cigarposition += int(re.sub('[\D]', '', element1))
                        #print ('cigarposition:',cigarposition)
                        if 'I' in element1:
                            #alles van mdcodes vermeerderen dat > cigarposition!
                            mdpositiontotal = 0 #variabele om plaats in mdcode bij te houden
                            i = 0 #plaats in mdcodelist: [0] of 1, 2
                            for element2 in mdcodelist:
                                mdsnp = re.sub('[\d]','', element2) #A, C, T of G
                                mdposition = int(re.sub('[\D]', '', element2)) #hoeveel plaatsen tussen
SNP's bv. 0A4T
                                mdpositiontotal += int(re.sub('[\D]','', element2)) #plaats in totaal
8,12,23
                                mdpositiontotal+=1
                                #print ('mdposition',mdpositiontotal)
                                if (mdpositiontotal+numberofindel) > cigarposition:
                                    mdposition=mdposition+numberofindel
                                    var=str(mdposition)+str(mdsnp)
                                    mdcodelist[i]=var
                                    break #na toevoegen van extra nt, uit de for-lus breken
                                i+=1
```

```
                    #print ('mdcodelistafter',mdcodelist)
                    for element in mdcodelist:
                        snpref = re.sub('[\d]', '', element)
                        snppos = re.sub('[\D]', '', element)
                        basecount += int(snppos)
                        #print (basecount)
                        if '^' in element: #corrigeren voor deletie
                            basecount-=1
                        #print('basecount:',basecount)
                        #print('sequentie',sequentie)
                        snpread = sequentie[basecount]
                        #print ("Oude base van referentie : ",snpref, snppos, "Nieuwe base van read : ",snpread)
                        basecount+=1 #telkens bij 1SNP staat de letter ook voor 1 plaats, maar dit wordt niet meegerekend in totale nummering
                        snp = snpref + ">" + snpread
                        if '^' not in element:
                            #print (snp)
                            snplist.append(snp)
                            totaalaantalsnps+=1

            countersnplist_temp = Counter(snplist)

            #omzetten naar percentages
            for element in list(countersnplist_temp):
                countersnplist_temp[element] = float(countersnplist_temp[element])/totaalaantalbasen

            countersnplist_perm = countersnplist_perm + countersnplist_temp
            print (countersnplist_perm)
            snplist = []
            countersnplist_temp = []

#percentages corrigeren voor aantal individuen:
for element in list(countersnplist_perm):
    countersnplist_perm[element] = float(countersnplist_perm[element])/aantalindividuen

print (countersnplist_perm)

for element in list(countersnplist_perm):
    var = str(element) + "\t" + str(countersnplist_perm[element])
    print (var)

c = countersnplist_perm.items()
c.sort(key=itemgetter(1))
labels, values = zip(*c)

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('Aantal SNPs')
plt.ylabel('Frequentie van voorkomen')
#plt.title('Afwijking per read')
plt.show()
```

Bijlage 5N: telomeren_hexamere.py

```
#Geeft voor de algemene populatie van 100/101 of 108reads de hexameren die het meest voorkomen (op de plaats van SNP)

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import subprocess
import glob

individuen100_101 = ['HG00096', ..., 'NA19664']
individuen108 = ['HG00100', ..., 'NA19470']
ASW100 = ['NA19985',...,'NA20356']
CLM100 = ['HG01112',...,'HG01492']
GBR100 = ['HG00096',...,'HG00233']
IBS100 = ['HG01515',...,'HG01522']
LWK100 = ['NA19318',...,'NA19428']
MXL100 = ['NA19728',...,'NA19664']
PUR100 = ['HG01097',...,'HG01204']
YRI100 = ['NA18917',...,'NA19223']
CEU108 = ['NA12272',...,'NA12718']
CHB108 = ['NA18614',...,'NA18634']
CHS108 = ['HG00403',...,'HG00635']
FIN108 = ['HG00171',...,'HG00377']
GBR108 = ['HG00100',...,'HG00265']
JPT108 = ['NA18941',...,'NA19088']
LWK108 = ['NA19311',...,'NA19470']
PUR108 = ['HG00553',...,'HG00640']

path = '/home/driesg/workplace/align/absolute_subtelomeer_*.txt'
#path = '/home/driesg/workplace/align/puurtelomeer_met_seqerror_*.txt'
#path = '/home/driesg/workplace/align/absolute_pseudotelomeer_*.txt'
#path = '/home/driesg/workplace/align/tel.align.*.txt'

files = glob.glob(path)
aantalindividuen = 0

break_lines=0
aantalindividuen = 0
snphexamerelist = []
countersnplist_temp = Counter()
countersnplist_perm = Counter()

for name in files:
    #tel.align.txt
    #individu = name[39:46] #HG00096, NA19278 etc.
    #absolute_pseudotelomeer.txt
    #individu = name[53:60]
    #puurtel+seqerror.txt
    #individu = name[55:62]
    #absolute_subtelomeer.txt
    individu = name[50:57]
    #print (individu)
```

```
    if individu in PUR108:
    #if individu in individuen108:
        print (individu)
        totaalaantalhexameres=0
        aantalindividuen+=1
        print (aantalindividuen)

        #if break_lines > 3:
            #break
        #break_lines+=1

        with open(name) as f:
            for line in f:

                basecount = 0 #positie bij overlopen van sequentiestring
                splittedline = line.split()
                cigarcode = (splittedline[5])
                sequentie = (splittedline[9])
                n = re.search(r"MD:Z:([\S]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    mdcodelist = re.split("([\d]+[\D]+)",mdcode) #Hier!!!
                    mdcodelist = [x for x in mdcodelist if "A" in x or "C" in x or "T" in x or "G" in
x]
                    cigarcodelist = re.split("([\d]+[\D])",cigarcode)
                    cigarcodelist = [element for element in cigarcodelist if element.strip()]
                    cigarposition = 0 #variabele om de plaats in cigarstring bij te houden
                    basecount = 0
                    #print ('mdcodelistbefore',mdcodelist)
                    #print (cigarcodelist)

                    #corrigeren van de mdcodelist bij aanwezigheid van insert: alles moet opschuiven op
de juiste plaats!
                    for element1 in cigarcodelist:
                        numberofindel = int(re.sub('[\D]', '', element1))
                        cigarposition += int(re.sub('[\D]', '', element1))
                        #print ('cigarposition:',cigarposition)
                        if 'I' in element1:
                            #alles van mdcodes vermeerderen dat > cigarposition!
                            mdpositiontotal = 0 #variabele om plaats in mdcode bij te houden
                            i = 0 #plaats in mdcodelist: [0] of 1, 2
                            for element2 in mdcodelist:
                                mdsnp = re.sub('[\d]','', element2) #A, C, T of G
                                mdposition = int(re.sub('[\D]', '', element2)) #hoeveel plaatsen tussen
SNP's bv. 0A4T
                                mdpositiontotal += int(re.sub('[\D]','', element2)) #plaats in totaal
8,12,23
                                mdpositiontotal+=1
                                #print ('mdposition',mdpositiontotal)
                                if (mdpositiontotal+numberofindel) > cigarposition:
                                    mdposition=mdposition+numberofindel
                                    var=str(mdposition)+str(mdsnp)
                                    mdcodelist[i]=var
                                    break #na toevoegen van extra nt, uit de for-lus breken
                                i+=1

                    for element in mdcodelist:
                        snpref = re.sub('[\d]', '', element)
```

```
                    snppos = re.sub('[\D]', '', element)
                    basecount += int(snppos) #
                    if '^' in element: #corrigeren voor deletie
                        basecount-=1
                    if '^' not in element:
                        #3bp voor en na indel nemen! Controleren of de indel niet helemaal vanvoor valt, anders krijg je een lege string in grafiek.
                        if basecount > 2 and basecount < (len(sequentie) - 2):
                            nieuweseq = sequentie[basecount-3:basecount+3]
                            #print (nieuweseq)
                            snphexamerelist.append(nieuweseq)
                            totaalaantalhexameres+=1
                        elif basecount <= 2:
                            nieuweseq = sequentie[0:6]
                            #print (nieuweseq)
                            snphexamerelist.append(nieuweseq)
                            totaalaantalhexameres+=1
                        elif basecount >= (len(sequentie) - 2):
                            nieuweseq = sequentie[len(sequentie)-6:len(sequentie)]
                            #print (nieuweseq)
                            snphexamerelist.append(nieuweseq)
                            totaalaantalhexameres+=1

                    basecount+=1 #telkens bij 1SNP staat de letter ook voor 1 plaats, maar dit wordt niet meegerekend in totale nummering

            countersnplist_temp = Counter(snphexamerelist)

            #omzetten naar percentages
            for element in list(countersnplist_temp):
                countersnplist_temp[element] = float(countersnplist_temp[element])/totaalaantalhexameres

            countersnplist_perm = countersnplist_perm + countersnplist_temp
            #print (countersnplist_perm)
            snphexamerelist = []
            countersnplist_temp = []

#percentages corrigeren voor aantal individuen:
for element in list(countersnplist_perm):
    countersnplist_perm[element] = float(countersnplist_perm[element])/aantalindividuen

#verwijderen van elementen met een voorkomen van minder dan X maal:
for element in list(countersnplist_perm):
    if countersnplist_perm[element] < 0.0025:
        del countersnplist_perm[element]

print(countersnplist_perm)

for element in list(countersnplist_perm):
    var = str(element) + "\t" + str(countersnplist_perm[element])
    print (var)

c = countersnplist_perm.items()
c.sort(key=itemgetter(1))
labels, values = zip(*c)

#labels, values = zip(*snplibrary.items())
```

```
indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('SNPs')
plt.ylabel('Frequentie')
plt.title('SNPs')
plt.show()
```

Bijlage 5O: mutationfrequency.py

```
#Dit script geeft een algemene mutatiefrequentie per geselecteerde groep:
#mutatiefreq = aantal detected SNP's/totaal aantal basen

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import subprocess
import glob

individuen100_101 = ['HG00096', ..., 'NA19664']
individuen108 = ['HG00100', ..., 'NA19470']
ASW100 = ['NA19985',...,'NA20356']
CLM100 = ['HG01112',...,'HG01492']
GBR100 = ['HG00096',...,'HG00233']
IBS100 = ['HG01515',...,'HG01522']
LWK100 = ['NA19318',...,'NA19428']
MXL100 = ['NA19728',...,'NA19664']
PUR100 = ['HG01097',...,'HG01204']
YRI100 = ['NA18917',...,'NA19223']
CEU108 = ['NA12272',...,'NA12718']
CHB108 = ['NA18614',...,'NA18634']
CHS108 = ['HG00403',...,'HG00635']
FIN108 = ['HG00171',...,'HG00377']
GBR108 = ['HG00100',...,'HG00265']
JPT108 = ['NA18941',...,'NA19088']
LWK108 = ['NA19311',...,'NA19470']
PUR108 = ['HG00553',...,'HG00640']

path = '/home/driesg/workplace/align/absolute_subtelomeer_*.txt'
#path = '/home/driesg/workplace/align/puurtelomeer_met_seqerror_*.txt'
#path = '/home/driesg/workplace/align/absolute_pseudotelomeer_*.txt'
#path = '/home/driesg/workplace/align/tel.align.*.txt'
files = glob.glob(path)

#break_lines=0
mutationfreqlist = []

for name in files:
    #tel.align.txt
    #individu = name[39:46] #HG00096, NA19278 etc.
    #absolute_pseudotelomeer.txt
    #individu = name[53:60]
    #puurtel+seqerror.txt
    #individu = name[55:62]
    #absolute_subtelomeer.txt
    individu = name[50:57]
    #print (individu)
    #if individu in individuen100_101:
    if individu in PUR108:
        print (individu)
        snpnumber = 0
        totaalaantalbasen = 0
        #if break_lines > 0:
```

```
            #break
        #break_lines+=1

        with open(name) as f:
            for line in f:
                splittedline = line.split()
                cigarcode = (splittedline[5])
                sequentie = (splittedline[9])
                totaalaantalbasen+=len(sequentie)
                #print(line)
                n = re.search(r"MD:Z:([\S]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    snpnumber                                                                    +=
mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')

            mutationfrequency = float(snpnumber)/float(totaalaantalbasen)
            mutationfreqlist.append(mutationfrequency)

print (mutationfreqlist)

for element in list(mutationfreqlist):
    print (element)
```

Bijlage 5P: telomerisch_profiel.py

```
#Dit dient om per individu een algemeen (gemiddeld) profiel te verkrijgen.
#Dus voor 1 individu: 500fragmenten met 1SNP, 1000fragmenten met 4SNP's.
#Omdat data met meer reads meer doorwegen bij gemiddeldes, wordt er gewerkt met percentages
#Dit resulteert in niet-gewogen gemiddelde

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import subprocess
import glob

individuen100_101 = ['HG00096', ..., 'NA19664']

individuen108 = ['HG00100', ..., 'NA19470']

path = '/home/driesg/workplace/align/tel.align.*.txt'
files = glob.glob(path)
aantalindividuen = 0

break_lines=0

#aanmaken van lege totalcountersnplistvector
totalcountersnplist=Counter()

for name in files:
    individu = name[39:46] #HG00096, NA19278 etc.
    #if individu in individuen100_101:
    if individu in individuen108:

        #if break_lines > 1:
            #break
        #break_lines+=1
        aantalreads=0
        snplist = [] #geeft alle counts terug, niet geordend
        countersnplist = []
        aantalindividuen +=1
        print(individu)
        print(aantalindividuen)

        with open(name) as f:
            for line in f:
                aantalreads+=1
                n = re.search(r"MD:Z:([\w]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    #N's worden niet beschouwd
                    snpnumber                                                    =
mdcode.count('A')+mdcode.count('C')+mdcode.count('T')+mdcode.count('G')
                    output = "%s" % (snpnumber)
                    snplist.append(output)

        countersnplist = Counter(snplist)
```

```
        #print(countersnplist)

        #omzetten naar percentages
        for element in list(countersnplist):
            #print (element) #bv.13SNPs
            #print (countersnplist[element]) #bv.576
            countersnplist[element] = float(countersnplist[element])/aantalreads

        #print (countersnplist)

        #percentages naar een andere vector die alle percentages samenvat
        for element in list(countersnplist):
            totalcountersnplist[element] += float(countersnplist[element])

#percentages corrigeren voor aantal individuen:
for element in list(totalcountersnplist):
    totalcountersnplist[element] = float(totalcountersnplist[element])/aantalindividuen

#verwijderen van elementen met een voorkomen van minder dan X maal:
for element in list(totalcountersnplist):
    if totalcountersnplist[element] < 0.001:
        del totalcountersnplist[element]

print (totalcountersnplist)

c = totalcountersnplist.items()
c.sort(key=itemgetter(1))
labels, values = zip(*c)

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('Aantal SNPs per read')
plt.ylabel('Frequentie van voorkomen')
#plt.title('Afwijking per read')
plt.show()
```

Bijlage 5Q: telomeren_hexameren_biol_relevance.py

```
#Dit script bepaalt hoeveel elke nt in het standaard hexameer muteert naar andere nt
#dit om biologische relevantie aan te duiden
#bv. CCCTAA
# C nr 1: X maal gemuteerd naar A,X maal naar G etc.
# C nr 2: idem

from collections import Counter #telt hoeveel patroon voorkomt
from operator import itemgetter #sorteren van de lijst

import re
import numpy as np
import matplotlib.pyplot as plt
import subprocess
import glob

individuentot = ['HG00096', ..., 'NA19470']
individuen100_101 = ['HG00096', ..., 'NA19664']
individuen108 = ['HG00100', ..., 'NA19470']
ASW100 = ['NA19985',...,'NA20356']
CLM100 = ['HG01112',...,'HG01492']
GBR100 = ['HG00096',...,'HG00233']
IBS100 = ['HG01515',...,'HG01522']
LWK100 = ['NA19318',...,'NA19428']
MXL100 = ['NA19728',...,'NA19664']
PUR100 = ['HG01097',...,'HG01204']
YRI100 = ['NA18917',...,'NA19223']
CEU108 = ['NA12272',...,'NA12718']
CHB108 = ['NA18614',...,'NA18634']
CHS108 = ['HG00403',...,'HG00635']
FIN108 = ['HG00171',...,'HG00377']
GBR108 = ['HG00100',...,'HG00265']
JPT108 = ['NA18941',...,'NA19088']
LWK108 = ['NA19311',...,'NA19470']
PUR108 = ['HG00553',...,'HG00640']

#path = '/home/driesg/workplace/align/absolute_subtelomeer_*.txt'
#path = '/home/driesg/workplace/align/puurtelomeer_met_seqerror_*.txt'
path = '/home/driesg/workplace/align/absolute_pseudotelomeer_*.txt'
#path = '/home/driesg/workplace/align/tel.align.*.txt'

files = glob.glob(path)
aantalindividuen = 0

break_lines=0
aantalindividuen = 0
snpmutlist = []
countersnplist_temp = Counter()
countersnplist_perm = Counter()
totaalaantalhexameres=0
stdevlist = []

case = "" #geeft mee indien hexameer aan begin ligt, in midden of aan einde

def checkmut(hexameer,case):
    #print (hexameer[2],hexameer[3],hexameer[4])
```

```
    if case == "midden":
        #Standaard sequentie is CCCTAA - CCCTAA. Eerst positie bepalen
        #Posities zijn C(1)C(2)C(3)T(4)A(5)A(6)
        if hexameer[2] == "A" and hexameer[4] == "C": #pos1 or pos6
            if hexameer[1] == "A" and hexameer[0] == "T": #pos1
                return("1"+hexameer[3])
            elif hexameer[1] == "T" and hexameer [0] == "C": #pos6:
                return("6"+hexameer[3])
        elif hexameer[2] == "C" and hexameer[4] == "C": #pos2
            return("2"+hexameer[3])
        elif hexameer[2] == "C" and hexameer[4] == "T": #pos3
            return("3"+hexameer[3])
        elif hexameer[2] == "C" and hexameer[4] == "A": #pos4
            return("4"+hexameer[3])
        elif hexameer[2] == "T" and hexameer[4] == "A": #pos5
            return("5"+hexameer[3])

    if case == "begin":
        #Standaard sequentie is CCCTAA - CCCTAA. Eerst positie bepalen
        #hexameer geeft base terug + 5 opeenvolgende nt:
        if hexameer[1] == "C" and hexameer[2] == "C": #pos1 of pos6
            if hexameer[3] == "T": #pos1
                return("1"+hexameer[0])
            elif hexameer[3] == "C": #pos6
                return("6"+hexameer[0])
        elif hexameer[1] == "C" and hexameer[2] == "T": #pos2
            return("2"+hexameer[0])
        elif hexameer[1] == "T" and hexameer[2] == "A": #pos3
            return("3"+hexameer[0])
        elif hexameer[1] == "A" and hexameer[2] == "A": #pos4
            return("4"+hexameer[0])
        elif hexameer[1] == "A" and hexameer[2] == "C": #pos5
            return("5"+hexameer[0])

    if case == "einde":
        #Standaard sequentie is CCCTAA - CCCTAA. Eerst positie bepalen
        #hexameer geeft 5 eerdere nt terug + gemuteerde base op einde:
        if hexameer[4] == "A" and hexameer[3] == "A": #pos1
            return("1"+hexameer[5])
        elif hexameer[4] == "C" and hexameer[3] == "A": #pos2
            return("2"+hexameer[5])
        elif hexameer[4] == "C" and hexameer[3] == "C": #pos3 of pos4
            if hexameer[2] == "A": #pos3
                return("3"+hexameer[5])
            elif hexameer[2] == "C": #pos4
                return("4"+hexameer[5])
        elif hexameer[4] == "T" and hexameer[3] == "C": #pos5
            return("5"+hexameer[5])
        elif hexameer[4] == "A" and hexameer[3] == "T": #pos6
            return("6"+hexameer[5])

for name in files:
    #tel.align.txt
    #individu = name[39:46] #HG00096, NA19278 etc.
    #absolute_pseudotelomeer.txt
    individu = name[53:60]
    #puurtel+seqerror.txt
```

```
    #individu = name[55:62]
    #absolute_subtelomeer.txt
    #individu = name[50:57]
    #print (individu)

    if individu in individuentot:
    #if individu in individuen100_101:
    #if individu in individuen108:
        print (individu)
        totaalaantalhexameres=0
        aantalindividuen+=1
        print (aantalindividuen)

        #if break_lines > 3:
            #break
        #break_lines+=1

        with open(name) as f:
            for line in f:

                basecount = 0 #positie bij overlopen van sequentiestring
                splittedline = line.split()
                cigarcode = (splittedline[5])
                sequentie = (splittedline[9])
                n = re.search(r"MD:Z:([\S]+)",line)
                if n:
                    mdcode = n.groups(1)[0]
                    mdcodelist = re.split("([\d]+[\D]+)",mdcode) #Hier!!!
                    mdcodelist = [x for x in mdcodelist if "A" in x or "C" in x or "T" in x or "G" in
x]
                    cigarcodelist = re.split("([\d]+[\D])",cigarcode)
                    cigarcodelist = [element for element in cigarcodelist if element.strip()]
                    cigarposition = 0 #variabele om de plaats in cigarstring bij te houden
                    basecount = 0
                    #print ('mdcodelistbefore',mdcodelist)
                    #print (cigarcodelist)

                    #corrigeren van de mdcodelist bij aanwezigheid van insert: alles moet opschuiven op
de juiste plaats!
                    for element1 in cigarcodelist:
                        numberofindel = int(re.sub('[\D]', '', element1))
                        cigarposition += int(re.sub('[\D]', '', element1))
                        #print ('cigarposition:',cigarposition)
                        if 'I' in element1:
                            #alles van mdcodes vermeerderen dat > cigarposition!
                            mdpositiontotal = 0 #variabele om plaats in mdcode bij te houden
                            i = 0 #plaats in mdcodelist: [0] of 1, 2
                            for element2 in mdcodelist:
                                mdsnp = re.sub('[\d]','', element2) #A, C, T of G
                                mdposition = int(re.sub('[\D]', '', element2)) #hoeveel plaatsen tussen
SNP's bv. 0A4T
                                mdpositiontotal += int(re.sub('[\D]','', element2)) #plaats in totaal
8,12,23
                                mdpositiontotal+=1
                                #print ('mdposition',mdpositiontotal)
                                if (mdpositiontotal+numberofindel) > cigarposition:
                                    mdposition=mdposition+numberofindel
                                    var=str(mdposition)+str(mdsnp)
```

```
                        mdcodelist[i]=var
                        break #na toevoegen van extra nt, uit de for-lus breken
                    i+=1

            for element in mdcodelist:
                snpref = re.sub('[\d]', '', element)
                snppos = re.sub('[\D]', '', element)
                basecount += int(snppos) #
                if '^' in element: #corrigeren voor deletie
                    basecount-=1
                if '^' not in element:
                    snpread = sequentie[basecount]
                    #3bp voor en na indel nemen! Controleren of de indel niet helemaal vanvoor valt, anders krijg je een lege string in grafiek.
                    if basecount > 2 and basecount < (len(sequentie) - 2):
                        case = "midden"
                        nieuweseq = sequentie[basecount-3:basecount+3]
                        checkedmut = checkmut(nieuweseq,case)
                        #snp = snpref + ">" + snpread
                        #print (snp)
                        #print (checkedmut)
                        if checkedmut:
                            snpmutlist.append(checkedmut)
                            totaalaantalhexameres+=1

                    elif basecount <= 2:
                        case = "begin"
                        nieuweseq = sequentie[basecount:basecount+6]
                        checkedmut = checkmut(nieuweseq,case)
                        if checkedmut:
                            snpmutlist.append(checkedmut)
                            totaalaantalhexameres+=1

                    elif basecount >= (len(sequentie) - 2):
                        case = "einde"
                        nieuweseq = sequentie[basecount-5:basecount+1]
                        checkedmut = checkmut(nieuweseq,case)
                        if checkedmut:
                            snpmutlist.append(checkedmut)
                            totaalaantalhexameres+=1

                basecount+=1 #telkens bij 1SNP staat de letter ook voor 1 plaats, maar dit wordt niet meegerekend in totale nummering

        countersnplist_temp = Counter(snpmutlist)

        #omzetten naar percentages
        for element in list(countersnplist_temp):
            countersnplist_temp[element] = float(countersnplist_temp[element])/totaalaantalhexameres

        #print (countersnplist_temp)

        #frequenties opzetten
        frequentie1=0
        frequentie2=0
        frequentie3=0
        frequentie4=0
```

```
            frequentie5=0
            frequentie6=0

            for element in list(countersnplist_temp):
                if element == "1T" or element == "1A" or element == "1G":
                    frequentie1+= float(countersnplist_temp[element])
                elif element == "2T" or element == "2A" or element == "2G":
                    frequentie2+= float(countersnplist_temp[element])
                elif element == "3T" or element == "3A" or element == "3G":
                    frequentie3+= float(countersnplist_temp[element])
                elif element == "4C" or element == "4A" or element == "4G":
                    frequentie4+= float(countersnplist_temp[element])
                elif element == "5T" or element == "5C" or element == "5G":
                    frequentie5+= float(countersnplist_temp[element])
                elif element == "6T" or element == "6C" or element == "6G":
                    frequentie6+= float(countersnplist_temp[element])


frequentietot=frequentie1+frequentie2+frequentie3+frequentie4+frequentie5+frequentie6
            #print (frequentietot) #zou zo dicht mogelijk bij 1 moeten liggen

            stdev                                                                    =
np.std([frequentie1,frequentie2,frequentie3,frequentie4,frequentie5,frequentie6], ddof=1)
            print (stdev)
            stdevlist.append(stdev)

            countersnplist_perm = countersnplist_perm + countersnplist_temp
            #print (countersnplist_perm)
            snpmutlist = []
            countersnplist_temp = []

#percentages corrigeren voor aantal individuen:
for element in list(countersnplist_perm):
    countersnplist_perm[element] = float(countersnplist_perm[element])/aantalindividuen

print(countersnplist_perm)

for element in list(stdevlist):
    print (element)

c = countersnplist_perm.items()
c.sort(key=itemgetter(1))
labels, values = zip(*c)

#labels, values = zip(*snplibrary.items())

indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width, facecolor='green')
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.xlabel('SNPs')
plt.ylabel('Frequentie')
plt.title('SNPs')
plt.show()
```

```
setwd("/home/driesg/workplace")
df <- read.table("Telomeerlengtes.txt",header = TRUE)
telseqdata = df[,c('Telseq')]
computeldata = df[,c('Computel')]

t.test(telseqdata,computeldata,paired=TRUE)

hist(telseqdata,xlab="Telomeerlengte",ylab="Frequentie",breaks=50,col="#2EB82E")

***COMPUTEL VS TELSEQ***
densTelseq <- density(telseqdata)
densComputel <- density(computeldata)
xlim <- range(densTelseq$x,densComputel$x)
ylim <- range(densTelseq$y,densComputel$y)
telseqCol <- rgb(133/255, 224/255, 133/255, 0.5)
computelCol <- rgb(46/255, 184/255, 46/255, 0.5)
plot(densTelseq,cex.lab=1.5,xlim=xlim,ylim=ylim,xlab="Telomeerlengte
(bp)",ylab="Frequentie",main = '',panel.first=grid())
polygon(densTelseq,density = -1, col=telseqCol)
polygon(densComputel,density = -1, col=computelCol)
legend('topright',c('Telseq','Computel'),fill=c(telseqCol,computelCol),bty='n',border=NA,cex
=1.5)
dev.copy(png,"histogramtelseqcomputel.png",width=8,height=6,units="in",res=500)
dev.off()

***COMPUTEL VS TELSEQ***
boxplot(telseqdata,computeldata,names=c("Telseq","Computel"),col=c(rgb(133/255,
224/255,      133/255),rgb(46/255,      184/255,      46/255)),ylab="Telomeerlengte
(bp)",cex.lab=1.5,ylim=limits,cex.axis=1.5,yaxt="n")
axis(2,cex.axis=1)
dev.copy(png,"boxplottelseqcomputel.png",width=8,height=6,units="in",res=500)
dev.off()

***READLENGTH***
setwd("/home/driesg/workplace")
df <- read.table("Telomeer_readlength.txt",header = TRUE,check.names=FALSE)
readlength100data = df[,c('100')]
readlength101data = df[,c('101')]
readlength108data = df[,c('108')]
boxplot(readlength100data,readlength101data,readlength108data,names=c("100bp
(n=94)","101bp       (n=67)","108bp       (n=145)"),col=c(rgb(133/255,       224/255,
133/255),rgb(46/255,          184/255,          46/255),rgb(102/255,          153/255,
0)),ylab="Telomeerlengte (bp)",cex.lab=1.5,cex.axis=1.5,yaxt="n")
axis(2,cex.axis=1)
dev.copy(png,"boxplotreadlengths.png",width=8,height=6,units="in",res=500)
dev.off()

***POPULATIES***
setwd("/home/driesg/workplace")
df <- read.table("Telomeer_populaties.txt",header = TRUE)
GBR = df[,c('GBR')]
PUR = df[,c('PUR')]
CLM = df[,c('CLM')]
LWK = df[,c('LWK')]
MXL = df[,c('MXL')]
```

```
ASW = df[,c('ASW')]
FIN = df[,c('FIN')]
CHS = df[,c('CHS')]
boxplot(GBR,PUR,CLM,LWK,MXL,ASW,FIN,CHS,col = rainbow(8),names=c("GBR \n(n=27)","PUR \n(n=36)","CLM \n(n=34)","LWK \n(n=62)","MXL \n(n=23)","ASW \n(n=26)","FIN \n(n=32)","CHS \n(n=20)"),ylab="Telomeerlengte (bp)",cex.lab=1.5,yaxt="n")
axis(2,cex.axis=1)
dev.copy(png,"telomeerpopulaties.png",width=8,height=6,units="in",res=500)
dev.off()

***SEQMACHINE***
setwd("/home/driesg/workplace")
df <- read.table("Telomeer_seqmachine.txt",header = TRUE)
GAII = df[,c('Illumina_Genome_Analyzer_II')]
GAI = df[,c('Illumina_Genome_Analyzer')]
HISEQ = df[,c('Illumina_HiSeq_2000')]
boxplot(GAII,GAI,HISEQ,names=c("GAII (n=262)","GAI (n=14)","HiSeq2000 (n=9)"),col=c(rgb(133/255, 224/255, 133/255),rgb(46/255, 184/255, 46/255),rgb(102/255, 153/255, 0)),ylab="Telomeerlengte (bp)",cex.lab=1.5,cex.axis=1.5,yaxt="n")
axis(2,cex.axis=1)
dev.copy(png,"boxplotseqmachine.png",width=8,height=6,units="in",res=500)
dev.off()

***SEQCENTER***
setwd("/home/driesg/workplace")
df <- read.table("Telomeer_seqcenter.txt",header = TRUE)
WUGSC = df[,c('WUGSC')]
SC = df[,c('SC')]
BI = df[,c('BI')]
ILLUMINA = df[,c('ILLUMINA')]
boxplot(WUGSC,SC,BI,ILLUMINA,names=c("WUGSC \n(n=85)","SC \n(n=154)","BI \n(n=44)","ILLUMINA \n(n=23)"),col=c(rgb(153/255, 255/255, 51/255),rgb(133/255, 224/255, 133/255),rgb(46/255, 184/255, 46/255),rgb(102/255, 153/255, 0)),ylab="Telomeerlengte (bp)",cex.lab=1.5,cex.axis=1,yaxt="n")
axis(2,cex.axis=1)
dev.copy(png,"boxplotseqcenter.png",width=8,height=6,units="in",res=500)
dev.off()

***GENDER***
setwd("/home/driesg/workplace")
df <- read.table("Telomeer_gender.txt",header = TRUE)
male = df[,c('male')]
female = df[,c('female')]
boxplot(male,female,names=c("Male","Female"),col=c(rgb(133/255, 224/255, 133/255),rgb(46/255, 184/255, 46/255)),ylab="Telomeerlengte (bp)",cex.lab=1.5,ylim=range(female),cex.axis=1.5,yaxt="n")
axis(2,cex.axis=1)
dev.copy(png,"boxplotgender.png",width=8,height=6,units="in",res=500)
dev.off()

***GENDER TUSSEN 100 EN 108***
df108 <- df[df[, "Read_length"]=="108",]
df100_101 <- df[df[, "Read_length"]=="kort",]

par(mfrow=c(1,2))
```

```
boxplot(data=df100_101,df100_101$Computel~df100_101$Gender,col=c(rgb(13/255,
224/255,      133/255),rgb(46/255,      184/255,      46/255)),ylab="Telomeerlengte
(bp)",cex.lab=1.5,ylim=c(0,70000))
boxplot(data=df108,df108$Computel~df108$Gender,col=c(rgb(13/255,              224/255,
133/255),rgb(46/255,              184/255,              46/255)),ylab="Telomeerlengte
(bp)",cex.lab=1.5,ylim=c(0,70000))
dev.copy(png,"boxplotgenderverschil100_101_108.png",width=8,height=6,units="in",res=
500)
dev.off()

t.test(df100_101$Computel~df100_101$Gender,paired=FALSE)
t.test(df108$Computel~df108$Gender,paired=FALSE)

***GLM***
setwd("/home/driesg/workplace")
df <- read.table("Telomeer_GLM.txt",header = TRUE)
> par(mfrow=c(1,2))
> boxplot(df[,c('Computel')],ylim=c(0,70000),main="Computel_origineel")
> boxplot(df[,c('Computel_edited')],ylim=c(0,70000),main="Computel_corrected")
> boxplot(df[,c('Computel_edited')]~df[,c('Read_length')])

***TESTWILCOXON VAN INSERTEN PER SNP***
setwd("/home/driesg/workplace")
df <- read.table("Hypothese_indel_snp.txt",header = TRUE)

***GENDER_PUURTELOMEER***
setwd("/home/driesg/workplace")
df <- read.table("Telomeerlengtes_mdas.txt",header = TRUE)
df108 <- df[df[, "Read_length"]=="108",]
df100_101 <- df[df[, "Read_length"]=="100",]
df100_101 <- rbind(df100_101,df[df[, "Read_length"]=="101",])
par(mfrow=c(1,2))
boxplot(data=df100_101,df100_101$Computel_puurtelomeer~df100_101$Gender,col=c(rgb(13
/255,   224/255,   133/255),rgb(46/255,   184/255,   46/255)),ylab="Telomeerlengte
(bp)",cex.lab=1.5,ylim=c(0,70000))
boxplot(data=df108,df108$Computel_puurtelomeer~df108$Gender,col=c(rgb(13/255,
224/255,      133/255),rgb(46/255,      184/255,      46/255)),ylab="Telomeerlengte
(bp)",cex.lab=1.5,ylim=c(0,70000))

t.test(df100_101$Computel_puurtelomeer~df100_101$Gender,paired=FALSE)
t.test(df108$Computel_puurtelomeer~df108$Gender,paired=FALSE)

***PUUR - PSEUDO - SUB***
setwd("C:/Users/Dries/Dropbox/School/School Sem 13 (Ir)/Masterproef/Documenten")
df <- read.table("Telomeerlengtes_mdas.txt",header = TRUE)
puurtel <- df[,c('Computel_puurtelomeer')]
puurtel_seqerror <- df[,c('Computel_puurtelomeer_seqerror')]
pseudotel <- df[,c('Computel_pseudotelomeer')]
tel <- df[,c('Computel')]
boxplot(tel,pseudotel,puurtel_seqerror,puurtel,names=c("Subtel","Pseudotel","Puurtel+seqerr
or","Puurtel"),col="green",ylab="Telomeerlengte (bp)")
stripchart(list(tel,pseudotel,puurtel_seqerror,puurtel),vertical=TRUE,group.names=c("Subtel","
Pseudotel","Puurtel+seqerror","Puurtel"))
for(i in 1:nrow(df)){segments(1, tel, 2, pseudotel)}
for(i in 1:nrow(df)){segments(2, pseudotel, 3, puurtel_seqerror)}
for(i in 1:nrow(df)){segments(3, puurtel_seqerror, 4, puurtel)}
```

```
stripchart(list(tel,pseudotel,puurtel_seqerror,puurtel),vertical=TRUE,group.names=c("Subtel","
Pseudotel","Puurtel+seqerror","Puurtel"),ylim=c(0,20000))
stripchart(list(tel,pseudotel,puurtel_seqerror,puurtel),vertical=TRUE,group.names=c("Subtel","
Pseudotel","Puurtel+seqerror","Puurtel"),ylim=c(0,10000))

...populaties...
segments(1,      df[df[,      "Population"]=="GBR",][,c('Computel')],      2,      df[df[,
"Population"]=="GBR",][,c('Computel_pseudotelomeer')],col="blue")
segments(1,      df[df[,      "Population"]=="LWK",][,c('Computel')],      2,      df[df[,
"Population"]=="LWK",][,c('Computel_pseudotelomeer')],col="red")
segments(1,      df[df[,      "Population"]=="PUR",][,c('Computel')],      2,      df[df[,
"Population"]=="PUR",][,c('Computel_pseudotelomeer')],col="green")
segments(1,      df[df[,      "Population"]=="CLM",][,c('Computel')],      2,      df[df[,
"Population"]=="CLM",][,c('Computel_pseudotelomeer')],col="orange")
segments(1,      df[df[,      "Population"]=="FIN",][,c('Computel')],      2,      df[df[,
"Population"]=="FIN",][,c('Computel_pseudotelomeer')],col="purple")
...andere populaties...

...etnisch...niet alle groepen weergegeven...
segments(1,      df[df[,      "Population"]=="GBR",][,c('Computel')],      2,      df[df[,
"Population"]=="GBR",][,c('Computel_pseudotelomeer')],col="blue")
segments(1,      df[df[,      "Population"]=="IBS",][,c('Computel')],      2,      df[df[,
"Population"]=="IBS",][,c('Computel_pseudotelomeer')],col="blue")
segments(1,      df[df[,      "Population"]=="FIN",][,c('Computel')],      2,      df[df[,
"Population"]=="FIN",][,c('Computel_pseudotelomeer')],col="blue")
segments(1,      df[df[,      "Population"]=="CEU",][,c('Computel')],      2,      df[df[,
"Population"]=="CEU",][,c('Computel_pseudotelomeer')],col="blue")
segments(1,      df[df[,      "Population"]=="YRI",][,c('Computel')],      2,      df[df[,
"Population"]=="YRI",][,c('Computel_pseudotelomeer')],col="black")
segments(1,      df[df[,      "Population"]=="LWK",][,c('Computel')],      2,      df[df[,
"Population"]=="LWK",][,c('Computel_pseudotelomeer')],col="black")
segments(1,      df[df[,      "Population"]=="ASW",][,c('Computel')],      2,      df[df[,
"Population"]=="ASW",][,c('Computel_pseudotelomeer')],col="black")
segments(1,      df[df[,      "Population"]=="CHS",][,c('Computel')],      2,      df[df[,
"Population"]=="CHS",][,c('Computel_pseudotelomeer')],col="orange")
segments(1,      df[df[,      "Population"]=="CHB",][,c('Computel')],      2,      df[df[,
"Population"]=="CHB",][,c('Computel_pseudotelomeer')],col="orange")
segments(1,      df[df[,      "Population"]=="JPT",][,c('Computel')],      2,      df[df[,
"Population"]=="JPT",][,c('Computel_pseudotelomeer')],col="orange")

segments(2,   df[df[,   "Population"]=="GBR",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="GBR",][,c('Computel_puurtelomeer_seqerror')],col="blue")
segments(2,   df[df[,   "Population"]=="IBS",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="IBS",][,c('Computel_puurtelomeer_seqerror')],col="blue")
segments(2,   df[df[,   "Population"]=="FIN",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="FIN",][,c('Computel_puurtelomeer_seqerror')],col="blue")
segments(2,   df[df[,   "Population"]=="CEU",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="CEU",][,c('Computel_puurtelomeer_seqerror')],col="blue")
segments(2,   df[df[,   "Population"]=="YRI",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="YRI",][,c('Computel_puurtelomeer_seqerror')],col="black")
segments(2,   df[df[,   "Population"]=="LWK",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="LWK",][,c('Computel_puurtelomeer_seqerror')],col="black")
segments(2,   df[df[,   "Population"]=="ASW",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="ASW",][,c('Computel_puurtelomeer_seqerror')],col="black")
segments(2,   df[df[,   "Population"]=="CHS",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="CHS",][,c('Computel_puurtelomeer_seqerror')],col="orange")
```

```
segments(2,  df[df[,  "Population"]=="CHB",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="CHB",][,c('Computel_puurtelomeer_seqerror')],col="orange")
segments(2,  df[df[,  "Population"]=="JPT",][,c('Computel_pseudotelomeer')],3,df[df[,
"Population"]=="JPT",][,c('Computel_puurtelomeer_seqerror')],col="orange")

segments(3,df[df[,
"Population"]=="GBR",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="GBR",][,c('Computel_puurtelomeer')],col="blue")
segments(3,df[df[,
"Population"]=="IBS",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="IBS",][,c('Computel_puurtelomeer')],col="blue")
segments(3,df[df[,
"Population"]=="FIN",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="FIN",][,c('Computel_puurtelomeer')],col="blue")
segments(3,df[df[,
"Population"]=="CEU",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="CEU",][,c('Computel_puurtelomeer')],col="blue")
segments(3,df[df[,
"Population"]=="YRI",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="YRI",][,c('Computel_puurtelomeer')],col="black")
segments(3,df[df[,
"Population"]=="LWK",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="LWK",][,c('Computel_puurtelomeer')],col="black")
segments(3,df[df[,
"Population"]=="ASW",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="ASW",][,c('Computel_puurtelomeer')],col="black")
segments(3,df[df[,
"Population"]=="CHS",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="CHS",][,c('Computel_puurtelomeer')],col="orange")
segments(3,df[df[,
"Population"]=="CHB",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="CHB",][,c('Computel_puurtelomeer')],col="orange")
segments(3,df[df[,
"Population"]=="JPT",][,c('Computel_puurtelomeer_seqerror')],4,df[df[,
"Population"]=="JPT",][,c('Computel_puurtelomeer')],col="orange")

*** PUURTEL + SEQERROR - PSEUDOTEL - SUBTEL ***
setwd("C:/Users/Dries/Dropbox/School/School Sem 13 (Ir)/Masterproef/Documenten")
df <- read.table("Telomeerlengtes_mdas.txt",header = TRUE)
puurtel_seqerror <- df[,c('Computel_puurtelomeer_seqerror')]
pseudotel <- df[,c('Computel_absoluut_pseudotelomeer')]
subtel <- df[,c('Computel_absoluut_subtelomeer')]
boxplot(puurtel_seqerror,pseudotel,subtel,names=c("Puurtel+seqerror","Pseudotel","Subtel")
,col="green",ylab="Telomeerlengte (bp)")
stripchart(list(puurtel_seqerror,pseudotel,subtel),vertical=TRUE,group.names=c("Puurtel+seqe
rror","Pseudotel","Subtel"),ylim=c(0,20000))
for(i in 1:nrow(df)){segments(1, puurtel_seqerror, 2, pseudotel)}
for(i in 1:nrow(df)){segments(2, pseudotel, 3, subtel)}

...onderscheid tss 100/101 en 108...
setwd("C:/Users/Dries/Dropbox/School/School Sem 13 (Ir)/Masterproef/Documenten")
df <- read.table("Telomeerlengtes_mdas.txt",header = TRUE)
df108 <- df[df[, "Read_length"]=="108",]
df100 <- df[df[, "Read_length"]=="100",]
df101 <- df[df[, "Read_length"]=="101",]
df100_101 <- rbind(df100,df101)
```

```
puurtel_seqerror100_101 <- df100_101[,c('Computel_puurtelomeer_seqerror')]
pseudotel100_101 <- df100_101[,c('Computel_absoluut_pseudotelomeer')]
subtel100_101 <- df100_101[,c('Computel_absoluut_subtelomeer')]
stripchart(list(puurtel_seqerror100_101,pseudotel100_101,subtel100_101),vertical=TRUE,group.
names=c("Puurtel+\nseqerror","Pseudotel","Hybride
tel"),ylim=c(0,15000),ylab="Telomeerlengte (bp)",las=2)
for(i in 1:nrow(df)){segments(1, puurtel_seqerror100_101, 2, pseudotel100_101)}
for(i in 1:nrow(df)){segments(2, pseudotel100_101, 3, subtel100_101)}

puurtel_seqerror108 <- df108[,c('Computel_puurtelomeer_seqerror')]
pseudotel108 <- df108[,c('Computel_absoluut_pseudotelomeer')]
subtel108 <- df108[,c('Computel_absoluut_subtelomeer')]
stripchart(list(puurtel_seqerror108,pseudotel108,subtel108),vertical=TRUE,group.names=c("Puu
rtel+\nseqerror","Pseudotel","Hybride tel"),ylab="Telomeerlengte (bp)",las=2)
#stripchart(list(puurtel_seqerror108,pseudotel108,subtel108),vertical=TRUE,group.names=c("P
uurtel+\nseqerror","Pseudotel","Hybride tel"),ylim=c(0,10000))
for(i in 1:nrow(df)){segments(1, puurtel_seqerror108, 2, pseudotel108)}
for(i in 1:nrow(df)){segments(2, pseudotel108, 3, subtel108)}

par(mfrow=c(1,2))
boxplot(puurtel_seqerror100_101,pseudotel100_101,subtel100_101,names=c("Puurtel+seqerro
r","Pseudotel","Hybride tel"),col="green",ylab="Telomeerlengte (bp)",ylim=c(0,40000))
boxplot(puurtel_seqerror108,pseudotel108,subtel108,names=c("Puurtel+seqerror","Pseudotel"
,"Hybride tel"),col="green",ylab="Telomeerlengte (bp)")

...populaties 108...
stripchart(list(puurtel_seqerror108,pseudotel108,subtel108),vertical=TRUE,group.names=c("Puu
rtel+\nseqerror","Pseudotel","Hybride          tel"),ylim=c(0,10000),ylab="Telomeerlengte
(bp)",cex.lab=1.5, cex.axis=1.5, cex.main=1.5, cex.sub=1.5)
segments(1,                                                    df108[df108[,
"Population"]=="GBR",][,c('Computel_puurtelomeer_seqerror')],     2,     df108[df108[,
"Population"]=="GBR",][,c('Computel_absoluut_pseudotelomeer')],col="blue")
segments(1,                                                    df108[df108[,
"Population"]=="LWK",][,c('Computel_puurtelomeer_seqerror')],     2,     df108[df108[,
"Population"]=="LWK",][,c('Computel_absoluut_pseudotelomeer')],col="red")
segments(1,                                                    df108[df108[,
"Population"]=="PUR",][,c('Computel_puurtelomeer_seqerror')],     2,     df108[df108[,
"Population"]=="PUR",][,c('Computel_absoluut_pseudotelomeer')],col="green")
segments(1,                                                    df108[df108[,
"Population"]=="CLM",][,c('Computel_puurtelomeer_seqerror')],     2,     df108[df108[,
"Population"]=="CLM",][,c('Computel_absoluut_pseudotelomeer')],col="orange")
segments(1,                                                    df108[df108[,
"Population"]=="FIN",][,c('Computel_puurtelomeer_seqerror')],     2,     df108[df108[,
"Population"]=="FIN",][,c('Computel_absoluut_pseudotelomeer')],col="purple")
segments(2,                                                    df108[df108[,
"Population"]=="GBR",][,c('Computel_absoluut_pseudotelomeer')],    3,     df108[df108[,
"Population"]=="GBR",][,c('Computel_absoluut_subtelomeer')],col="blue")
segments(2,                                                    df108[df108[,
"Population"]=="LWK",][,c('Computel_absoluut_pseudotelomeer')],    3,     df108[df108[,
"Population"]=="LWK",][,c('Computel_absoluut_subtelomeer')],col="red")
segments(2,                                                    df108[df108[,
"Population"]=="PUR",][,c('Computel_absoluut_pseudotelomeer')],    3,     df108[df108[,
"Population"]=="PUR",][,c('Computel_absoluut_subtelomeer')],col="green")
segments(2,                                                    df108[df108[,
"Population"]=="CLM",][,c('Computel_absoluut_pseudotelomeer')],    3,     df108[df108[,
"Population"]=="CLM",][,c('Computel_absoluut_subtelomeer')],col="orange")
```

```
segments(2, df108[df108[,"Population"]=="FIN",][,c('Computel_absoluut_pseudotelomeer')], 3, df108[df108[,"Population"]=="FIN",][,c('Computel_absoluut_subtelomeer')],col="purple")
legend("bottomright",title = "Population",bg="white",c("GBR","LWK","PUR","CLM","FIN"),fill=c("blue","red","green","orange","purple"),cex=1.5)

...populaties 100/101...
stripchart(list(puurtel_seqerror108,pseudotel108,subtel108),vertical=TRUE,group.names=c("Puurtel+\nseqerror","Pseudotel","Hybride tel"),ylim=c(0,10000),ylab="Telomeerlengte (bp)",las=2)
#segments(1, df100_101[df100_101[,"Population"]=="GBR",][,c('Computel_puurtelomeer_seqerror')], 2, df100_101[df100_101[,"Population"]=="GBR",][,c('Computel_absoluut_pseudotelomeer')],col="blue")
segments(1, df100_101[df100_101[,"Population"]=="LWK",][,c('Computel_puurtelomeer_seqerror')], 2, df100_101[df100_101[,"Population"]=="LWK",][,c('Computel_absoluut_pseudotelomeer')],col="red")
#segments(1, df100_101[df100_101[,"Population"]=="PUR",][,c('Computel_puurtelomeer_seqerror')], 2, df100_101[df100_101[,"Population"]=="PUR",][,c('Computel_absoluut_pseudotelomeer')],col="green")
#segments(1, df100_101[df100_101[,"Population"]=="CLM",][,c('Computel_puurtelomeer_seqerror')], 2, df100_101[df100_101[,"Population"]=="CLM",][,c('Computel_absoluut_pseudotelomeer')],col="orange")
#segments(1, df100_101[df100_101[,"Population"]=="FIN",][,c('Computel_puurtelomeer_seqerror')], 2, df100_101[df100_101[,"Population"]=="FIN",][,c('Computel_absoluut_pseudotelomeer')],col="purple")

#segments(2, df100_101[df100_101[,"Population"]=="GBR",][,c('Computel_absoluut_pseudotelomeer')], 3, df100_101[df100_101[,"Population"]=="GBR",][,c('Computel_absoluut_subtelomeer')],col="blue")
segments(2, df100_101[df100_101[,"Population"]=="LWK",][,c('Computel_absoluut_pseudotelomeer')], 3, df100_101[df100_101[,"Population"]=="LWK",][,c('Computel_absoluut_subtelomeer')],col="red")
#segments(2, df100_101[df100_101[,"Population"]=="PUR",][,c('Computel_absoluut_pseudotelomeer')], 3, df100_101[df100_101[,"Population"]=="PUR",][,c('Computel_absoluut_subtelomeer')],col="green")
#segments(2, df100_101[df100_101[,"Population"]=="CLM",][,c('Computel_absoluut_pseudotelomeer')], 3, df100_101[df100_101[,"Population"]=="CLM",][,c('Computel_absoluut_subtelomeer')],col="orange")
#segments(2, df100_101[df100_101[,"Population"]=="FIN",][,c('Computel_absoluut_pseudotelomeer')], 3, df100_101[df100_101[,"Population"]=="FIN",][,c('Computel_absoluut_subtelomeer')],col="purple")

...boxplots van populaties puurtel + seqerror...

par(mfrow=c(1,2))
```

```
df100_101_puurtelomeer_ASW <- df100_101[df100_101[,"Population"]=="ASW",][,c('Computel_puurtelomeer_seqerror')]
df100_101_puurtelomeer_CLM <- df100_101[df100_101[,"Population"]=="CLM",][,c('Computel_puurtelomeer_seqerror')]
df100_101_puurtelomeer_GBR <- df100_101[df100_101[,"Population"]=="GBR",][,c('Computel_puurtelomeer_seqerror')]
df100_101_puurtelomeer_IBS <- df100_101[df100_101[,"Population"]=="IBS",][,c('Computel_puurtelomeer_seqerror')]
df100_101_puurtelomeer_LWK <- df100_101[df100_101[,"Population"]=="LWK",][,c('Computel_puurtelomeer_seqerror')]
df100_101_puurtelomeer_MXL <- df100_101[df100_101[,"Population"]=="MXL",][,c('Computel_puurtelomeer_seqerror')]
df100_101_puurtelomeer_PUR <- df100_101[df100_101[,"Population"]=="PUR",][,c('Computel_puurtelomeer_seqerror')]
df100_101_puurtelomeer_YRI <- df100_101[df100_101[,"Population"]=="YRI",][,c('Computel_puurtelomeer_seqerror')]

boxplot(
df100_101_puurtelomeer_ASW,
df100_101_puurtelomeer_CLM,
df100_101_puurtelomeer_GBR,
df100_101_puurtelomeer_IBS,
df100_101_puurtelomeer_LWK,
df100_101_puurtelomeer_MXL,
df100_101_puurtelomeer_PUR,
df100_101_puurtelomeer_YRI,
col = c("yellow","orange","blue","lightgoldenrod","red","salmon","green","palegreen"),
names=c("ASW (n=26)","CLM (n=33)","GBR (n=4)","IBS (n=6)","LWK (n=29)","MXL (n=23)","PUR (n=25)","YRI (n=8)"),ylab="Telomeerlengte (bp)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2,ylim=c(0,40000))
axis(2,cex.axis=1)

df108_puurtelomeer_CEU <- df108[df108[,"Population"]=="CEU",][,c('Computel_puurtelomeer_seqerror')]
df108_puurtelomeer_CHB <- df108[df108[,"Population"]=="CHB",][,c('Computel_puurtelomeer_seqerror')]
df108_puurtelomeer_CHS <- df108[df108[,"Population"]=="CHS",][,c('Computel_puurtelomeer_seqerror')]
df108_puurtelomeer_FIN <- df108[df108[,"Population"]=="FIN",][,c('Computel_puurtelomeer_seqerror')]
df108_puurtelomeer_GBR <- df108[df108[,"Population"]=="GBR",][,c('Computel_puurtelomeer_seqerror')]
df108_puurtelomeer_JPT <- df108[df108[,"Population"]=="JPT",][,c('Computel_puurtelomeer_seqerror')]
df108_puurtelomeer_LWK <- df108[df108[,"Population"]=="LWK",][,c('Computel_puurtelomeer_seqerror')]
df108_puurtelomeer_PUR <- df108[df108[,"Population"]=="PUR",][,c('Computel_puurtelomeer_seqerror')]

boxplot(
df108_puurtelomeer_CEU,
df108_puurtelomeer_CHB,
df108_puurtelomeer_CHS,
df108_puurtelomeer_FIN,
df108_puurtelomeer_GBR,
df108_puurtelomeer_JPT,
df108_puurtelomeer_LWK,
```

```
df108_puurtelomeer_PUR,
col = c("plum","royalblue","seashell","sienna","blue","tan","red","green"),
names=c("CEU (n=4)","CHB (n=12)","CHS (n=20)","FIN (n=31)","GBR (n=23)","JPT (n=16)","LWK (n=33)","PUR (n=5)"),
ylab="Telomeerlengte (bp)",yaxt="n",ylim=c(0,40000),cex.lab=1, cex.axis=0.75,las=2)
axis(2,cex.axis=1)
```

...boxplots van populaties pseudotel...

```
par(mfrow=c(1,2))
df100_101_pseudotelomeer_ASW <- df100_101[df100_101[,"Population"]=="ASW",][,c('Computel_absoluut_pseudotelomeer')]
df100_101_pseudotelomeer_CLM <- df100_101[df100_101[,"Population"]=="CLM",][,c('Computel_absoluut_pseudotelomeer')]
df100_101_pseudotelomeer_GBR <- df100_101[df100_101[,"Population"]=="GBR",][,c('Computel_absoluut_pseudotelomeer')]
df100_101_pseudotelomeer_IBS <- df100_101[df100_101[,"Population"]=="IBS",][,c('Computel_absoluut_pseudotelomeer')]
df100_101_pseudotelomeer_LWK <- df100_101[df100_101[,"Population"]=="LWK",][,c('Computel_absoluut_pseudotelomeer')]
df100_101_pseudotelomeer_MXL <- df100_101[df100_101[,"Population"]=="MXL",][,c('Computel_absoluut_pseudotelomeer')]
df100_101_pseudotelomeer_PUR <- df100_101[df100_101[,"Population"]=="PUR",][,c('Computel_absoluut_pseudotelomeer')]
df100_101_pseudotelomeer_YRI <- df100_101[df100_101[,"Population"]=="YRI",][,c('Computel_absoluut_pseudotelomeer')]

boxplot(
df100_101_pseudotelomeer_ASW,
df100_101_pseudotelomeer_CLM,
df100_101_pseudotelomeer_GBR,
df100_101_pseudotelomeer_IBS,
df100_101_pseudotelomeer_LWK,
df100_101_pseudotelomeer_MXL,
df100_101_pseudotelomeer_PUR,
df100_101_pseudotelomeer_YRI,
col = c("yellow","orange","blue","lightgoldenrod","red","salmon","green","palegreen"),
names=c("ASW (n=26)","CLM (n=33)","GBR (n=4)","IBS (n=6)","LWK (n=29)","MXL (n=23)","PUR (n=25)","YRI (n=8)"),ylab="Telomeerlengte (bp)",yaxt="n",ylim=c(0,10500),cex.lab=1, cex.axis=0.75,las=2)
axis(2,cex.axis=1)

df108_pseudotelomeer_CEU <- df108[df108[,"Population"]=="CEU",][,c('Computel_absoluut_pseudotelomeer')]
df108_pseudotelomeer_CHB <- df108[df108[,"Population"]=="CHB",][,c('Computel_absoluut_pseudotelomeer')]
df108_pseudotelomeer_CHS <- df108[df108[,"Population"]=="CHS",][,c('Computel_absoluut_pseudotelomeer')]
df108_pseudotelomeer_FIN <- df108[df108[,"Population"]=="FIN",][,c('Computel_absoluut_pseudotelomeer')]
df108_pseudotelomeer_GBR <- df108[df108[,"Population"]=="GBR",][,c('Computel_absoluut_pseudotelomeer')]
df108_pseudotelomeer_JPT <- df108[df108[,"Population"]=="JPT",][,c('Computel_absoluut_pseudotelomeer')]
df108_pseudotelomeer_LWK <- df108[df108[,"Population"]=="LWK",][,c('Computel_absoluut_pseudotelomeer')]
```

```
df108_pseudotelomeer_PUR <- df108[df108[,"Population"]=="PUR",][,c('Computel_absoluut_pseudotelomeer')]

boxplot(
df108_pseudotelomeer_CEU,
df108_pseudotelomeer_CHB,
df108_pseudotelomeer_CHS,
df108_pseudotelomeer_FIN,
df108_pseudotelomeer_GBR,
df108_pseudotelomeer_JPT,
df108_pseudotelomeer_LWK,
df108_pseudotelomeer_PUR,
col = c("plum","royalblue","seashell","sienna","blue","tan","red","green"),
names=c("CEU (n=4)","CHB (n=12)","CHS (n=20)","FIN (n=31)","GBR (n=23)","JPT (n=16)","LWK (n=33)","PUR (n=5)"),
ylab="Telomeerlengte (bp)",yaxt="n",ylim=c(0,10500),cex.lab=1, cex.axis=0.75,las=2)
axis(2,cex.axis=1)
```

...boxplots van populaties hybride tel...

```
par(mfrow=c(1,2))
df100_101_subtel_ASW <- df100_101[df100_101[,"Population"]=="ASW",][,c('Computel_absoluut_subtelomeer')]
df100_101_subtel_CLM <- df100_101[df100_101[,"Population"]=="CLM",][,c('Computel_absoluut_subtelomeer')]
df100_101_subtel_GBR <- df100_101[df100_101[,"Population"]=="GBR",][,c('Computel_absoluut_subtelomeer')]
df100_101_subtel_IBS <- df100_101[df100_101[,"Population"]=="IBS",][,c('Computel_absoluut_subtelomeer')]
df100_101_subtel_LWK <- df100_101[df100_101[,"Population"]=="LWK",][,c('Computel_absoluut_subtelomeer')]
df100_101_subtel_MXL <- df100_101[df100_101[,"Population"]=="MXL",][,c('Computel_absoluut_subtelomeer')]
df100_101_subtel_PUR <- df100_101[df100_101[,"Population"]=="PUR",][,c('Computel_absoluut_subtelomeer')]
df100_101_subtel_YRI <- df100_101[df100_101[,"Population"]=="YRI",][,c('Computel_absoluut_subtelomeer')]

boxplot(
df100_101_subtel_ASW,
df100_101_subtel_CLM,
df100_101_subtel_GBR,
df100_101_subtel_IBS,
df100_101_subtel_LWK,
df100_101_subtel_MXL,
df100_101_subtel_PUR,
df100_101_subtel_YRI,
col = c("yellow","orange","blue","lightgoldenrod","red","salmon","green","palegreen"),
names=c("ASW (n=26)","CLM (n=33)","GBR (n=4)","IBS (n=6)","LWK (n=29)","MXL (n=23)","PUR (n=25)","YRI (n=8)"),ylab="Telomeerlengte (bp)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2,ylim=c(0,22000))
axis(2,cex.axis=1)

df108_subtel_CEU <- df108[df108[,"Population"]=="CEU",][,c('Computel_absoluut_subtelomeer')]
df108_subtel_CHB <- df108[df108[,"Population"]=="CHB",][,c('Computel_absoluut_subtelomeer')]
```

```
df108_subtel_CHS <- df108[df108[,"Population"]=="CHS",][,c('Computel_absoluut_subtelomeer')]
df108_subtel_FIN <- df108[df108[,"Population"]=="FIN",][,c('Computel_absoluut_subtelomeer')]
df108_subtel_GBR <- df108[df108[,"Population"]=="GBR",][,c('Computel_absoluut_subtelomeer')]
df108_subtel_JPT <- df108[df108[,"Population"]=="JPT",][,c('Computel_absoluut_subtelomeer')]
df108_subtel_LWK <- df108[df108[,"Population"]=="LWK",][,c('Computel_absoluut_subtelomeer')]
df108_subtel_PUR <- df108[df108[,"Population"]=="PUR",][,c('Computel_absoluut_subtelomeer')]

boxplot(
df108_subtel_CEU,
df108_subtel_CHB,
df108_subtel_CHS,
df108_subtel_FIN,
df108_subtel_GBR,
df108_subtel_JPT,
df108_subtel_LWK,
df108_subtel_PUR,
col = c("plum","royalblue","seashell","sienna","blue","tan","red","green"),
names=c("CEU (n=4)","CHB (n=12)","CHS (n=20)","FIN (n=31)","GBR (n=23)","JPT (n=16)","LWK (n=33)","PUR (n=5)"),
ylab="Telomeerlengte (bp)",yaxt="n",cex.lab=1, cex.axis=0.75,las=2,ylim=c(0,22000))
axis(2,cex.axis=1)

*** AANTAL READS PER INDIVIDU ***
setwd("C:/Users/Dries/Dropbox/School/School Sem 13 (Ir)/Masterproef/Documenten")
data100101 <- c(17888, 13874, 12334, 6421, 19501, 8784, 27490, 12823, 7932, 15107,
4825, 15610, 64173, 12986, 16061, 8110, 75810, 29991, 23910, 18123, 26142, 14523,
18519, 10204, 13476, 16905, 13846, 13454, 25660, 12855, 13882, 15199, 96642,
7959, 7028, 23038, 21968, 49279, 16070, 16237, 19750, 9828, 17506, 18344, 51332,
13526, 31945, 19081, 7504, 8090, 19309, 16715, 15116, 16136, 12333, 20106, 13572,
25371, 15492, 14733, 68358, 13000, 67825, 18844, 9588, 28860, 6430, 17643, 21807,
8662, 19018, 11868, 38382, 28379, 5302, 14280, 5836, 22703, 8275, 51074, 40054,
32411, 9031, 11009, 10198, 52420, 15190, 21270, 13295, 28636, 21642, 19301,
19875, 19455, 14421, 16598, 30243, 34598, 9023, 28134, 15810, 12740, 47977,
12018, 18555, 10552, 26891, 21110, 20897, 17079, 15506, 5334, 22954, 13127,
29710, 12410, 10681, 35554, 16868, 13582, 114954, 12403, 12434, 21914, 9328,
12599, 23361, 15841, 18100, 16841, 17756, 25823, 22292, 14552, 28274, 23142,
17129, 7688, 25991, 4819, 22115, 9879, 9557, 9418, 17365, 15289, 14992, 12268,
26806, 30181, 15141, 89343, 27590, 12374, 8970, 25519, 12054, 24004, 8484, 13736,
45019)
data108 <- c(90283, 19714, 37369, 23867, 19559, 38467, 70415, 13575, 22555, 10063,
4853, 31631, 11706, 50028, 44284, 52140, 25449, 20717, 3157, 45035, 47197, 20356,
69841, 2869, 40748, 27574, 12117, 33029, 6584, 59201, 15144, 36461, 30097, 24625,
30637, 7908, 49430, 8276, 48206, 9962, 70335, 4447, 103633, 35132, 33943, 22468,
8397, 22314, 28719, 12775, 76537, 17349, 18729, 45381, 28688, 40144, 29369,
87966, 53887, 31893, 12844, 11051, 18732, 27715, 123321, 18922, 13857, 11309,
49196, 39707, 5191, 23819, 19508, 75944, 28097, 9610, 38445, 43653, 6586, 52896,
28370, 4728, 8832, 13669, 8439, 11367, 15796, 12282, 40386, 69067, 16081, 14666,
22238, 11582, 39086, 15157, 112398, 50889, 45104, 25667, 82856, 29850, 82283,
56940, 30412, 30070, 10706, 35518, 11729, 21695, 6426, 8200, 12646, 120526,
21667, 63464, 13474, 9942, 64475, 114728, 39596, 61770, 47028, 9132, 27322,
```

```
72286, 9474, 9351, 4219, 16954, 28161, 62242, 24599, 38123, 31958, 26512, 19951,
21873, 55728, 70519, 53745, 15976, 8770, 68404, 50868)
boxplot(data100101,data108,names=c("100/101-data","108-
data"),col="green",ylab="Aantal telomerische reads per
individu",cex.axis=1.5,cex.lab=1.5)

*** VERGELIJKEN VAN POPULATIES VIA ANOVA ***
totaldf100_101puur =
c(df100_101_puurtelomeer_ASW,df100_101_puurtelomeer_CLM,df100_101_puurtelomeer_GBR,df
100_101_puurtelomeer_IBS,df100_101_puurtelomeer_LWK,df100_101_puurtelomeer_MXL,df100_
101_puurtelomeer_PUR,df100_101_puurtelomeer_YRI)
groups100_101 = factor(rep(letters[1:8], c(26,33,4,6,29,23,25,8)))
bartlett.test(totaldf100_101puur,groups100_101)
fit <- aov(formula = totaldf100_101puur ~groups100_101)
anova(fit)
TukeyHSD(fit)
posthocTGH(totaldf100_101puur,groups100_101,method=c("games-howell", "tukey"),
digits=2)

totaldf108puur =
c(df108_puurtelomeer_CEU,df108_puurtelomeer_CHB,df108_puurtelomeer_CHS,df108_puurtelom
eer_FIN,df108_puurtelomeer_GBR,df108_puurtelomeer_JPT,df108_puurtelomeer_LWK,df108_puur
telomeer_PUR)
groups108 = factor(rep(letters[1:8], c(4,12,20,31,23,16,33,5)))
bartlett.test(totaldf108puur,groups108)
fligner.test(totaldf108puur,groups108)
fit <- aov(formula = totaldf108puur ~groups108)
anova(fit)
TukeyHSD(fit)
posthocTGH(totaldf108puur,groups108,method=c("games-howell", "tukey"), digits=2)

totaldf100_101pseudo =
c(df100_101_pseudotelomeer_ASW,df100_101_pseudotelomeer_CLM,df100_101_pseudotelomeer
_GBR,df100_101_pseudotelomeer_IBS,df100_101_pseudotelomeer_LWK,df100_101_pseudotelom
eer_MXL,df100_101_pseudotelomeer_PUR,df100_101_pseudotelomeer_YRI)
groups100_101 = factor(rep(letters[1:8], c(26,33,4,6,29,23,25,8)))
bartlett.test(totaldf100_101pseudo,groups100_101)
fligner.test(totaldf100_101pseudo,groups100_101)
fit <- aov(formula = totaldf100_101pseudo ~groups100_101)
anova(fit)
TukeyHSD(fit)
posthocTGH(totaldf100_101pseudo,groups100_101,method=c("games-howell", "tukey"),
digits=2)

totaldf108pseudo =
c(df108_pseudotelomeer_CEU,df108_pseudotelomeer_CHB,df108_pseudotelomeer_CHS,df108_ps
eudotelomeer_FIN,df108_pseudotelomeer_GBR,df108_pseudotelomeer_JPT,df108_pseudotelomeer
_LWK,df108_pseudotelomeer_PUR)
groups108 = factor(rep(letters[1:8], c(4,12,20,31,23,16,33,5)))
bartlett.test(totaldf108pseudo,groups108)
fligner.test(totaldf108pseudo,groups108)
fit <- aov(formula = totaldf108pseudo ~groups108)
anova(fit)
TukeyHSD(fit)
posthocTGH(totaldf108pseudo,groups108,method=c("games-howell", "tukey"), digits=2)
```

```
totaldf100_101subtel =
c(df100_101_subtel_ASW,df100_101_subtel_CLM,df100_101_subtel_GBR,df100_101_subtel_IBS,
df100_101_subtel_LWK,df100_101_subtel_MXL,df100_101_subtel_PUR,df100_101_subtel_YRI)
groups100_101 = factor(rep(letters[1:8], c(26,33,4,6,29,23,25,8)))
bartlett.test(totaldf100_101subtel,groups100_101)
fligner.test(totaldf100_101subtel,groups100_101)
fit <- aov(formula = totaldf100_101subtel ~groups100_101)
anova(fit)
TukeyHSD(fit)
posthocTGH(totaldf100_101subtel,groups100_101,method=c("games-howell",  "tukey"),
digits=2)

totaldf108subtel =
c(df108_subtel_CEU,df108_subtel_CHB,df108_subtel_CHS,df108_subtel_FIN,df108_subtel_GBR,df
108_subtel_JPT,df108_subtel_LWK,df108_subtel_PUR)
groups108 = factor(rep(letters[1:8], c(4,12,20,31,23,16,33,5)))
bartlett.test(totaldf108subtel,groups108)
fligner.test(totaldf108subtel,groups108)
fit <- aov(formula = totaldf108subtel ~groups108)
anova(fit)
TukeyHSD(fit)
posthocTGH(totaldf108subtel,groups108,method=c("games-howell", "tukey"), digits=2)

***VERGELIJKEN VAN MUTATIEFREQ ***
...all...
setwd("C:/Users/Dries/Dropbox/School/School Sem 13 (Ir)/Masterproef/Documenten")
df <- read.table("Mutationfreq.txt",header = TRUE)
df100_101 = c(df[,1],df[,2],df[,3],df[,4],df[,5],df[,6],df[,7],df[,8])
groups100_101 = factor(rep(letters[1:8], c(26,34,4,6,29,23,31,8)))
df100_101 <- df100_101[!is.na(df100_101)]
fit <- aov(formula = df100_101 ~groups100_101)
bartlett.test(df100_101,groups100_101)
boxplot(df[,1:8])
anova(fit)
library(userfriendlyscience)
posthocTGH(df100_101,groups100_101,method=c("games-howell", "tukey"), digits=2)

df108 = c(df[,9],df[,10],df[,11],df[,12],df[,13],df[,14],df[,15],df[,16])
groups108 = factor(rep(letters[1:8], c(4,12,20,32,23,16,33,5)))
df108 <- df108[!is.na(df108)]
bartlett.test(df108,groups108)
fit <- aov(formula = df108 ~groups108)
anova(fit)
posthocTGH(df108,groups108,method=c("games-howell", "tukey"), digits=2)
fit <- aov(formula = df108 ~groups108)

par(mfrow=c(1,2))
boxplot(
df[,c('GBR100')],
df[,c('LWK100')],
df[,c('PUR100')],
df[,c('IBS100')],
df[,c('YRI100')],
df[,c('MXL100')],
df[,c('CLM100')],
df[,c('ASW100')],
col = c("blue","red","green","lightgoldenrod","palegreen","salmon","orange","yellow"),
```

```
names=c("GBR (n=4)","LWK (n=29)","PUR (n=25)","IBS (n=6)","YRI (n=8)","MXL
(n=23)","CLM (n=33)","ASW (n=26)")
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)

boxplot(
df[,c('GBR108')],
df[,c('LWK108')],
df[,c('PUR108')],
df[,c('CEU108')],
df[,c('CHB108')],
df[,c('CHS108')],
df[,c('FIN108')],
df[,c('JPT108')],
col = c("blue","red","green","plum","royalblue","seashell","sienna","tan"),
names=c("GBR (n=23)","LWK (n=33)","PUR (n=5)","CEU (n=4)","CHB (n=12)","CHS
(n=20)","FIN (n=31)","JPT (n=16)")
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)

...puur...
setwd("C:/Users/Dries/Dropbox/School/School Sem 13 (Ir)/Masterproef/Documenten")
df <- read.table("Mutationfreq_puur.txt",header = TRUE)
df100_101 = c(df[,1],df[,2],df[,3],df[,4],df[,5],df[,6],df[,7],df[,8])
groups100_101 = factor(rep(letters[1:8], c(26,34,4,6,29,23,31,8)))
df100_101 <- df100_101[!is.na(df100_101)]
boxplot(df[,1:8], col="")
fit <- aov(formula = df100_101 ~groups100_101)
bartlett.test(df100_101,groups100_101)
anova(fit)
library(userfriendlyscience)
posthocTGH(df100_101,groups100_101,method=c("games-howell", "tukey"), digits=2)

df108 = c(df[,9],df[,10],df[,11],df[,12],df[,13],df[,14],df[,15],df[,16])
groups108 = factor(rep(letters[1:8], c(4,12,20,32,23,16,33,5)))
df108 <- df108[!is.na(df108)]
bartlett.test(df108,groups108)
fit <- aov(formula = df108 ~groups108)
anova(fit)
posthocTGH(df108,groups108,method=c("games-howell", "tukey"), digits=2)

par(mfrow=c(1,2))
boxplot(
df[,c('GBR100')],
df[,c('LWK100')],
df[,c('PUR100')],
df[,c('IBS100')],
df[,c('YRI100')],
df[,c('MXL100')],
df[,c('CLM100')],
df[,c('ASW100')],
col = c("blue","red","green","lightgoldenrod","palegreen","salmon","orange","yellow"),
names=c("GBR (n=4)","LWK (n=29)","PUR (n=25)","IBS (n=6)","YRI (n=8)","MXL
(n=23)","CLM (n=33)","ASW (n=26)")
```

```
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)

boxplot(
df[,c('GBR108')],
df[,c('LWK108')],
df[,c('PUR108')],
df[,c('CEU108')],
df[,c('CHB108')],
df[,c('CHS108')],
df[,c('FIN108')],
df[,c('JPT108')],
col = c("blue","red","green","plum","royalblue","seashell","sienna","tan"),
names=c("GBR (n=23)","LWK (n=33)","PUR (n=5)","CEU (n=4)","CHB (n=12)","CHS
(n=20)","FIN (n=31)","JPT (n=16)")
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)

...pseudo...
df <- read.table("Mutationfreq_pseudo.txt",header = TRUE)
df100_101 = c(df[,1],df[,2],df[,3],df[,4],df[,5],df[,6],df[,7],df[,8])
groups100_101 = factor(rep(letters[1:8], c(26,34,4,6,29,23,31,8)))
df100_101 <- df100_101[!is.na(df100_101)]
boxplot(df[,1:8])
fit <- aov(formula = df100_101 ~groups100_101)
bartlett.test(df100_101,groups100_101)
anova(fit)
library(userfriendlyscience)
posthocTGH(df100_101,groups100_101,method=c("games-howell", "tukey"), digits=2)

df108 = c(df[,9],df[,10],df[,11],df[,12],df[,13],df[,14],df[,15],df[,16])
groups108 = factor(rep(letters[1:8], c(4,12,20,32,23,16,33,5)))
df108 <- df108[!is.na(df108)]
boxplot(df[,9:16])
bartlett.test(df108,groups108)
fit <- aov(formula = df108 ~groups108)
anova(fit)
posthocTGH(df108,groups108,method=c("games-howell", "tukey"), digits=2)

par(mfrow=c(1,2))
boxplot(
df[,c('GBR100')],
df[,c('LWK100')],
df[,c('PUR100')],
df[,c('IBS100')],
df[,c('YRI100')],
df[,c('MXL100')],
df[,c('CLM100')],
df[,c('ASW100')],
col = c("blue","red","green","lightgoldenrod","palegreen","salmon","orange","yellow"),
names=c("GBR (n=4)","LWK (n=29)","PUR (n=25)","IBS (n=6)","YRI (n=8)","MXL
(n=23)","CLM (n=33)","ASW (n=26)")
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)
```

```
boxplot(
df[,c('GBR108')],
df[,c('LWK108')],
df[,c('PUR108')],
df[,c('CEU108')],
df[,c('CHB108')],
df[,c('CHS108')],
df[,c('FIN108')],
df[,c('JPT108')],
col = c("blue","red","green","plum","royalblue","seashell","sienna","tan"),
names=c("GBR (n=23)","LWK (n=33)","PUR (n=5)","CEU (n=4)","CHB (n=12)","CHS (n=20)","FIN (n=31)","JPT (n=16)")
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)

...hybride...
df <- read.table("Mutationfreq_hybride.txt",header = TRUE)
df100_101 = c(df[,1],df[,2],df[,3],df[,4],df[,5],df[,6],df[,7],df[,8])
groups100_101 = factor(rep(letters[1:8], c(26,34,4,6,29,23,31,8)))
df100_101 <- df100_101[!is.na(df100_101)]
boxplot(df[,1:8])
fit <- aov(formula = df100_101 ~groups100_101)
bartlett.test(df100_101,groups100_101)
anova(fit)
library(userfriendlyscience)
posthocTGH(df100_101,groups100_101,method=c("games-howell", "tukey"), digits=2)

df108 = c(df[,9],df[,10],df[,11],df[,12],df[,13],df[,14],df[,15],df[,16])
groups108 = factor(rep(letters[1:8], c(4,12,20,32,23,16,33,5)))
df108 <- df108[!is.na(df108)]
boxplot(df[,9:16])
bartlett.test(df108,groups108)
fit <- aov(formula = df108 ~groups108)
anova(fit)
posthocTGH(df108,groups108,method=c("games-howell", "tukey"), digits=2)

par(mfrow=c(1,2))
boxplot(
df[,c('GBR100')],
df[,c('LWK100')],
df[,c('PUR100')],
df[,c('IBS100')],
df[,c('YRI100')],
df[,c('MXL100')],
df[,c('CLM100')],
df[,c('ASW100')],
col = c("blue","red","green","lightgoldenrod","palegreen","salmon","orange","yellow"),
names=c("GBR (n=4)","LWK (n=29)","PUR (n=25)","IBS (n=6)","YRI (n=8)","MXL (n=23)","CLM (n=33)","ASW (n=26)")
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)

boxplot(
df[,c('GBR108')],
```

```
df[,c('LWK108')],
df[,c('PUR108')],
df[,c('CEU108')],
df[,c('CHB108')],
df[,c('CHS108')],
df[,c('FIN108')],
df[,c('JPT108')],
col = c("blue","red","green","plum","royalblue","seashell","sienna","tan"),
names=c("GBR (n=23)","LWK (n=33)","PUR (n=5)","CEU (n=4)","CHB (n=12)","CHS
(n=20)","FIN (n=31)","JPT (n=16)")
,ylab="Mutatiefrequentie (gemuteerde basen/totale basen)",yaxt="n",cex.lab=1,
cex.axis=0.75,las=2)
axis(2,cex.axis=1)
```