İnsan Genomunda Bulunan G-Dörtlülerinin Sınıflandırılarak Korunmuş Motiflerin Keşfi Ve Varyantlarının Analizi

Program Kodu: 3001

Proje No: 216S854

Proje Yürütücüsü:
Dr. Osman DOLUCA

Araştırmacı(lar):
Dr. Oktay I Kaplan

Bursiyer(ler):
Tugay Direk

NİSAN 2019

İZMİR

Önsöz

Bu çalışmada insan genomundaki G-dörtlülerinin tespit ve sınıflandırması amaçlanmış, bulunan G-dörtlülerinin bilinen genetik hastalıklar ile ilişkilendirilerek, bu hastalıkların G-dörtlülerinde yapısal bozulmaya neden olup olmadığı TÜBİTAK destekli 3001 projesi ile araştırılmıştır. Geliştirilen biyoenformatik araçlar ve sonuçları bilim dünyasına kazandırılarak yeni çalışmaların önünün açılması amaçlanmıştır.

İçindekiler

Tablo ve şekil listeleri

Özet

DNA motifileri proteinlerin amino asit dizilerinin saklanmasının yanı sıra genlerin regülasyonunda görev alan kısa DNA dizileridir. Bu diziler transkripsiyon faktörleri ya da diğer enzimlerle etkileşim göstererek gen regülasyonunda rol alabilmektedirler. Bu motiflerden bazıları DNA'nın bilindik 2 zincirli B-formun haricindeki sıradışı formlarını alarak bu etkileşimi tetikleyebilmektedirler. Bu sıradışı formlardan biri de G-dörtlüsü olarak bilinen, guanince zengin DNA yada RNA dizilerince oluşturulabilen yapılardır. G-dörtlüleri kendi aralarında birçok yapısal farklılık gösterebilen ve bu nedenle farklı rollere sahip bir yapı grubudur. Bu yapılar özellikle promotor bölgesi gibi regülasyonda etkili bölgelerde oluşarak transkripsiyonu etkileyebilmektedirler. Ancak bu yapıların tespiti ve rolleri konusunda çalışmalar devam etmekle beraber insan genomunda tahmin edilen 700.000'in üzerindeki G-dörtlüsünün rollerinin ve analizlerinin yapılması zaman ve emek alan bir süreçtir. Bu nedenle biyoenformatik yöntemler ile ön analiz ve tahminler yapılması ilgi çekmekte, varsa bu yapılardaki patojenik etkilerinin ortaya çıkarılması için tıbbi anlamda değer taşımaktadır. Bu çalışmada TÜBİTAK desteği ile G-dörtlülerinin tahmininden, sınıflandırılması ve ilişkilendirilmiş patojenik varyasyonların tespitine kadar gerekli bir seri biyoenformatik teknikler geliştirilmiş ve insan genomu için uygulanmıştır. İlk aşamada G-dörtlüsü tahmini için yeni bir algoritma geliştirilmiş ve başarısı yayınlanan makale ve bildiriler ile gösterilmiştir. Ardından insan genomunda tahmin edilen G-dörtlüleri bulunmuş, sınıflandırılarak farklı G-dörtlüsü motifleri ortaya çıkarılmıştır. Bulunan G-dörtlülerin patojenik etkisi bilinen varyasyonlar ile eşleştirilmiş ve bu varyasyonların etkisinin G-dörtlüsü yapısını etkileyerek gerçekleşip gerçekleşmediğinin tartışılabilmesi için yapısal değişimler deneysel olarak araştırılmıştır. Bulunan sonuçlar gerçekten bazı G-dörtlülerinin patojenik varyasyonlar sonucu yapısal olarak değişebildiğini göstermiştir.

Anahtar kelimeler: G-dörtlüsü, biyoenformatik, sekans analizi, varyant analizi, genetik hastalıklar

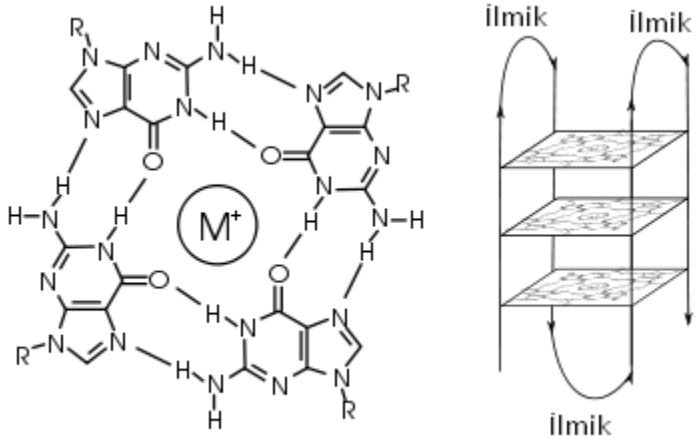
Abstract

DNA motifs are short DNA sequences that is responsible of gene regulation instead of retaining amino acid information. These short sequences function through interaction with transcription factors or other enzymes. G-quadruplexes is one of classes of motifs that are formed by guanine-rich sequences and can take non-canonical shapes of DNA instead of B-form. While G-quadruplexes may show wide range of structural variety, this variety also reflect onto variety of functions. These structures may affect transcription by forming in regulatory regions such as promoters. While many studies are ongoing to enlighten their role, it is a time and effort consuming process considering that there are over 700.000 G-quadruplexes predicted to exist on human genome. For that reason, bioinformatic techniques are attractive for preliminary analysis and predictions for G-quadruplex formation along with investigation of their pathogenic effect for medical purposes. With the support of TUBITAK, this study resulted in development of bioinformatical techniques for prediction, clustering and revealing associated pathogeny and applied them to human genome. Initially a novel algorithm was developed for G-quadruplex prediction and presented in symposiums and a journal article. In the following, G-quadruplexes were predicted for human genome and clustered. The discovered G-quadruplexes were associated with pathological variations and the impact of the variation on the G-quadruplex structure was experimentally analyzed. Results indicated that a number of G-quadruplexes indeed were impacted by these variations.

Keywords: G-quadruplexes, bioinformatics, sequence analysis, variant analysis, genetic diseases

1. **GİRİŞ**

Yaygın olarak bilindiği üzere genetik kalıtımdan sorumlu DNA molekülleri çift sarmallı heliks yapılarına sahiptir. Ancak son yıllarda yapılan birçok biyokimya temelli çalışma bu moleküllerin bazları arasında sıradışı hidrojen bağlarının kurulması ile farklı formlarda bulunabileceğini göstermiştir. Bu yapıların en önemlilerinden biri de G-dörtlüsü yapılarıdır (Neidle, 2016). Bir dizi üzerinde ardışık olarak sıralanmış guanin bazlarının katlanıp kendi aralarında Hoogsten eşleşme bölgesinden hidrojen bağı yapması ile G-tetrad yapıları oluşmaktadır. () Bu düzlemsel yapıdaki G-tetradların üst üste kümelenmesi ile G-dörtlüsü yapıları meydana gelmektedir (Sun vd., 2019; Zhang vd., 2009). G-dörtlüsü oluşumu G-tetrad yapılarının Na^+ ve K^+ gibi monovalent katyonların merkezinde nulunması ile daha da kararlı hale geçerler (Kamel vd., 1998).



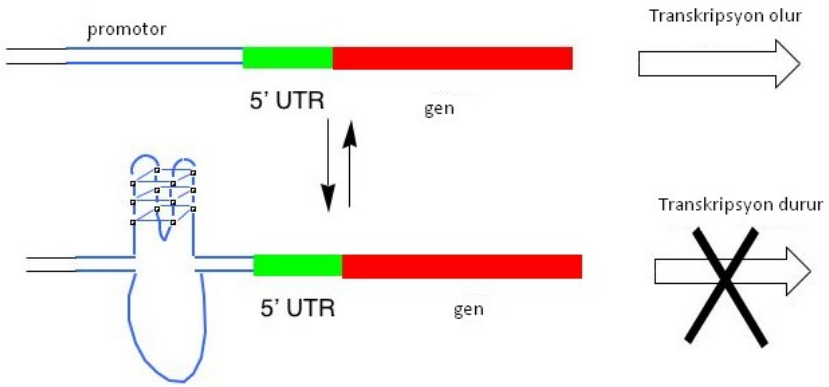
Şekil . G-tetrad oluşumu ve G-dörtlüsü yapısı

Bu yapılar bir yada daha çok DNA yada RNA zincirlerinin biraraya gelmesi ile intra- veya intermoleküler yapılarında gözlemlenebilmektedir. (Kim vd., 2003) Bunun yanı sıra birçok çalışma bu yapıların bilinen çift sarmal yapılarına oranla yüksek termal stabiliteye sahip olduğunu ortaya çıkarmıştır. (Fujimoto vd., 2009) İlk kez *in vitro*'da (Kikin vd., 2006) keşfedilen G-dörtlüsü yapılarının günümüzde tek hücreli canlılardan (Harris ve Merrick, 2015) insana (Hänsel-Hertsch vd., 2017) çeşitli canlıların genomunda sıklıkla bulunduğu tespit edilmiştir. Devam eden çalışmalar doğrultusunda G-dörtlüsülerin genom içerisindeki özel lokasyonları ve ligandları ile yaptığı etkileşimler bu yapıların önemli fonksiyonlara sahip olduğuna işaret etmektedir (Rhodes ve Lipps, 2015).

2. **LİTERATÜR ÖZETİ**

2.1 **Bilinen Fonksiyonel Rolleri**

Hesaplama temelli analizler insan genomunda 700.000'den fazla dizinin G-dörtlüsü yapısı oluşturma potensiyeline sahip olduğunu ortaya çıkarmıştır. (Ding vd., 2018) Ayrıca bu yapıların rastsal bir dağılım göstermediği, genomun fonksiyonel bölgelerinde kümelendikleri saptanmıştır. G-dörtlüsü yapıları insan genomunda en çok telomer bölgelerinde bulunmaktadır. (Lin ve Yang, 2017) Bu bölgelerde bulunan G-dörtlüsü yapılarının telomeraz aktivitesini durdurmaktan sorumlu olduğu ortaya çıkarılmıştır. (Zheng vd., 2011) Ayrıca bu yapılara birçok gen bölgesinin promotör dizilerinde (Balasubramanian vd., 2011), intron-ekson sekanslarının sınırlarında (Weldon vd., 2018), immünoglobulin rekombinasyonundan sorumlu gen bölgelerinde(Sattentau vd., 2018) de rastlanmaktadır. Ayrıca bu yapıların önemli onkogenlerin (c-MYC (You vd., 2015), KRAS (Cogoi ve Xodo, 2006) vb.) promotörlerinde bulunması kanser ile ilişkisi olduğunu ortaya koymaktadır. ()



Şekil . G-dörtlüsünün KRAS geninin kontrol mekanizmasındaki rolü. G-dörtlüsü oluşumu transkripsyonu durdurur

Sıradışı bir şekilde insan genomundaki DNA replikasyon orijinlerinin %90'ı G-dörtlüsü yapıları barındırmaktadır. (Rhodes ve Lipps, 2015) Dahası son yıllardaki çalışmalar G-dörtlüsülerin transkripsiyonda görev alan XPB ve XPD gibi çeşitli helikazlarla etkileşim sergileyerek gen ekspresyonunda önemli rol sahibi olduğunu ortaya koymuştur. (Sauer ve Paeschke, 2017) Bununla beraber yaklaşık 3000 insan gen bölgesinden ekspres olan mRNA moleküllerinin 5'-UTR sekanslarında G-dörtlüsü yapılarına sahip olduğu ve translasyon düzenlenmesini bu yolla gerçekleştirdiği keşfedilmiştir. (Song vd., 2016) Ayrıca telomer bölgelerinin transkripsiyonundan elde edilen ve guanince zengin olan TERRA mRNA ürünlerinde de bu yapılara rastlanmaktadır. (Zhou vd., 2015)

G-dörtlüsü yapılarının önemli onkogenlerin promotörlerinde bulunması ve telomeraz aktivitesini düzenlemesiyle kanser biyolojisi için büyük öneme sahiptir. Bu doğrultuda G-dörtlüsü yapılarının hedeflendiği kansere yönelik ilaç geliştirme çalışmalarına literatürde sıklıkla rastlanmaktadır. (Onel vd., 2014) Ayrıca birçok çalışma bazı hastalıkların G-dörtlüsü yapılarıyla etkileşime giren ligandarı, helikazları ve benzeri protein ürünlerini kodlayan gen bölgelerinde mutasyon oluşması ile meydana geldiğini göstermiştir. (Maizels ve Maizels, 2015) Bununla beraber G-dörtlüsü yapılarının transkripsiyon, translasyon ve replikasyon işlemlerini düzenlemesi metabolik faaliyetler üzerindeki önemini göstermektedir. Bahsi geçen bu sebeplerden dolayı G-dörtlüsü yapılarını, özelliklerini, katlanma ve açılma kinetiklerini, genom içerisindeki lokasyonlarını ve fonksiyonlarını araştırmak genetik bozuklukları inceleyen gelecek çalışmalara ışık tutacaktır.

Bu yapıların memeli hücrelerindeki görevlerini araştıran çalışmaların günden güne artmasına karşın mikroorganizmalardaki görevleri çok fazla aydınlatılmamıştır. Buna rağmen elde edilen bulgular bu yapıların mikroorganizma metabolizmasında ve patojenitelerinde önemli olduğunu ortaya çıkarmıştır. Mikroorganizmaların enfeksiyona kolaylık sağlayan en büyük stratejileri antijenik varyasyon sağlayabilmeleridir. Bu antijenik varyasyon işlemi genom içerisindeki spesifik genlerin rekombinasyona uğramaları ile gerçekleştirilmektedir. (Woude ve Ba, 2004) Bu işlemin gerçekleştiği birçok rekombinasyon bölgelerinde G-dörtlüsü yapılarına rastlanması bu yapıların antijenik varyasyonda büyük önem sahibi olduğunu göstermiştir. (Walia ve Chaconas, 2013) Bununla beraber büyük çaplı ölümlere sebep olan HIV virüsünün genomonun uç sekanslarında ve gen bölgelerinin aralarında uzun terminal tekrarlarına sahip olduğu ve bu tekrarların G-dörtlüsü yapıları barındırdığı ortaya çıkarılmıştır. Bu durum virüsün son derece komplex mekanizmaları G-dörtlüsü yapıları aracılığıyla yönettiğini göstermiştir. (Krafčíková vd., 2017) Ayrıca bu yapılar diğer lentivirüs genomlarının uzun terminal tekrarlarında da gözlemlenmektedir. (Perrone vd., 2017) Benzer şekilde çift sarmallı DNA içeren HHV virüsünde virülans faktörlerden sorumlu gen bölgelerinin promotör bölgelerinde G-quadurplex yapıları saptanmıştır. (Biswas vd., 2016) Yine benzer şekilde serviks kanserine sebep olduğu bilinen insan papillom virüsü'nde (HPV) çeşitli genlerin promotör bölgelerindeki G-dörtlüsü yapıları ile düzenlendiği kaydedilmiştir (Tlučková vd., 2013). Viral latens virüslerin enfeksiyon sonrası konak içerisinde varlığını devam ettirebilmek ve immün yanıttan gizlenebilmek için gerçekleştirdikleri bir mekanizmadır. Bu mekanizma temel olarak virüs genomunun heterokromatinizasyonunu sağlayan ve virüs replikasyonunu engelleyen genlerin ekspresyonuna dayanmaktadır(Grinde, 2013). Şu ana kadar farklı HHV suşlarında viral latensten sorumlu genlerin G-dörtlüsü yapıları ile yönetildiği tespit edilmiştir(Wood ve Royle,

2017). Bütün bu bilgiler ışığında G-dörtlüsü yapılarının viral enfeksiyon ve enfeksiyon sonrası varlığı koruyabilmede virüslere olanak sağladığı belirtilmektedir.

G-dörtlüsü yapılarının bakteri ve mayalardaki görevleri çok fazla aydınlatılmamış olmasına rağmen genomlarındaki görülme sıklıkları bu yapıların önemini vurgulamaktadır. Şu ana kadar elde edilen bulgular G-dörtlüsü yapılarının bu organizmalarda en çok karbonhidrat, amino asit ve nükleotid metabolizmasından sorumlu genlerin promotörlerinde kümelendiğini göstermektedir(Saranathan ve Vivekanandan, 2018). Spesifik bir örnek olarak *Deinococcus radiodurans* bakterisinin radyasyon direnci sağlayan genlerinin promotör bölgelerindeki G-dörtlüsü yapıları aracılığı ile aktivite gösterdiği ortaya çıkarılmıştır(Kota vd., 2015). İlginç olarak bulgular termofil bakterilerin yüksek sıcaklığa dirençlerini sağlayan gen promotörlerinin G-dörtlüsü yapılarınca zengin olduğunu göstermektedir(Long vd., 2017). Ding vd. 2018 yılında yaptıkları çalışmada *Thermales* ve *Deinococales* bakterilerindeki G-dörtlüsü yapılarını araştırmışlardır. Elde ettikleri sonuçlar *Thermales* bakterisindeki guanin'ce zengin dizilerin G-dörtlüsü yapısı oluşturmadığını gösterirken, *Deinococales* stres dirençli bakterisinin gen regülasyon bölgelerinde bu yapılara sıklıkla rastlanmıştır(Ding vd., 2018). Bu bilgiler ışığında özellikle belirlili çevresel koşullarda yaşayan bakterilerin bu özelliklerini G-dörtlüsü yapıları aracılığıyla ilerlettikleri çıkarılmaktadır. Virüslere benzer olarak patojenlik özelliği gösteren bakteri suşlarında da G-dörtlüsü yapılarına rastlanmaktadır. Bu araştırmalara örnek olarak Jain vd. 2018 yılında yaptıkları çalışmada *Salmonella enterica* bakterisindeki G-dörtlüsü yapılarını araştırmışlardır. Biyoinformatik temelli yürütülen çalışmada besin zehirlenmesinde etkin olan bu bakterinin Mg^{+2}, Fe^{+3} iyonları ve maltoz şekeri transportunu sağlayan genlerin düzenleyici bölgelerinde G-dörtlüsü yapıları tespit edilmiştir. Bakteri bu mekanizmalar ile makrofajlar tarafından üretilen reaktif nitrojen/oksijen varlığında canlılığını koruyabilirken insan sindirim sisteminin ana şeker kaynağını kullanabilmektedir(Jain vd., 2018). Bu ve benzeri birçok biyoinformatik temelli çalışma devam ederken model organizma olarak bilinen *E.coli* bakterisinde tekrar ettiği tespit edilen G-dörtlüsü yapıları tarafımızdan keşfedilmiş, bir moleküler anahtar fonksiyonu olabileceği gösterilmiştir. (Oktay I Kaplan vd., 2016)

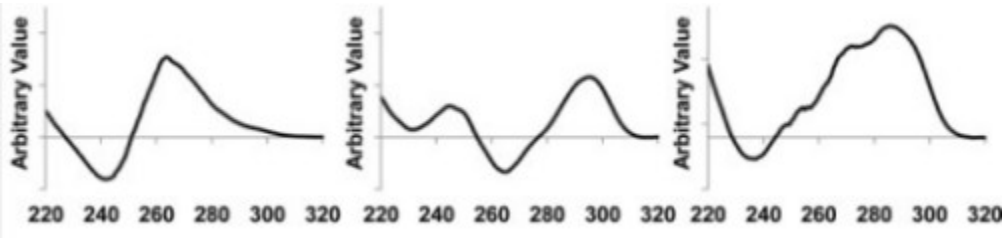
Görüldüğü üzere G-dörtlüsü yapıları memeli hücrelerinde olduğu gibi mikroorganizmaların yaşamsal faaliyetleri, çevre adaptasyonu ve enfeksiyon verimlerini arttırma stratejileri bakımından büyük öneme sahiptir. Bu doğrultuda güncel olarak yürütülen birçok çalışma mikroorganizmalarda yeni G-dörtlüsü yapıları keşfetmeye, bu yapıların görevlerini incelemeye ve bu yapılara bağlanarak stabilize edecek ajanlar üretmeye dayanmaktadır. Bu yapıların stabilize edilebilmesinin bir çok enfeksiyon kökenli hastalığı ve mikroorganizmaların yeni

stratejiler geliştirmesini engelleyebileceği düşünülmektedir. Üretilecek olan bu ajanlar yüksek afiniteli bir ligand gibi G-dörtlüsü yapılarına bağlanarak gen bölgelerini inaktif hale getirecektir(Ligands vd., 2019). Bu doğrultuda yapılan araştırmalar antimikrobiyal ilaç geliştirilmesi çalışmalarına büyük ışık tutacaktır.

2.2 **Tespit yöntemleri**

2.2.1 **Deneysel yöntemler**

G-dörtlüleri yapılarına göre birçok alt sınıfa ayrılabilir. Guanin zincirlerinin komşularına ters istikamette olması durumunda anti-paralel, aynı yöne bakması durumunda paralel G-dörtlüsü adı verilir. Bu durum anti-paralel G-dörtlüsünde birbirine karşı duran zincirlerin aynı yöne bakmasını sağlar. Bu iki G-dörtlüsü çeşidinin yanı sıra, bir de hibrid G-dörtlüsü vardır. Hibrid çeşit G-dörtlüsünde guanin zincirlerinin üçü aynı yönde, diğeri zıt yönde yatmaktadır. Paralel G-dörtlülerinde tüm guaninlerin glikozidik bağı sadece anti yöndedir. Anti-paralel G-dörtlüsünde ise ters yönde olan zincirlerde glikozidik bağın syn konfigürasyonunda olduğu tespit edilmiştir. Glikozidik bağın durumu üst üste duran guaninlerin birbiri ile olan oryantasyonlarını etkilediğinden G-dörtlüsünün polarize ışığı absorbasyon kapasitesini etkilemektedir. Bu durum dairesel dikroizm adı verilen yöntem ile tespit edilebilir ve G-dörtlüsünün yapısı hakkında fikir verebilmektedir. (Jones, 2003; Wagner ve Spelsberg, 1971; Wang, 2008) Dairesel dikroizm ile elde edilen spektrumlar G-dörtlüsünün topolojine ve dolaylı olarak çeşidine göre değişim göstermektedir. Dairesel dikroizm spektrumları parallel G-dörtlülerinde genellikle 240 nm'de negatif ve 260 nm'de pozitif zirve verirken, anti-paralel G-dörtlülerinde 250 nm'de pozitif, 265 nm'de negatif ve 280 nm'de tekrar pozitif zirve vermektedir. () Ancak spektrumlar ilmiklerin konum ve dizisine bağlı olarak değişim gösterebilir. (Kypr vd., 2009; Wang, 2008)



Şekil . Paralel (solda), antiparalel (ortada) ve hibrid (sağda) G-dörtlüsü dairesel dikroizm spektrumları

Dairesel kroizm yöntemi G-dörtlüsü oluşumunun izlenmesinde yegane araçlardan biridir. Alternatif olarak kullanılan NMR yönteminden daha ucuz ve hızlı oluşu nedeniyle sıkça tercih edilmektedir. Ancak bu yöntemin de kendince bazı dezavantajları bulunmaktadır. Bunlardan ilki, G-dürtlülerinin bakıldığı dalga boylarında hücre içindeki diğer moleküllerin de etkileşim göstermesi nedeniyle sadece in vitro ortamda kullanılabilmesidir. Belli bir nükleik asit dizisinin ya

da kombinasyonlarının oluşturacağı G-dörtlülerinin yapısı hakkında fikir elde edilmesi için ilgili dizilerin sentetik olarak elde edilmesi ve fizyolojik ya da yakın şartlarda inkübe edildikten sonra dairesel dikroizm yöntemiyle test edilmesi ile gerçekleştirilir.

2.2.2 **Bilgisayar temelli yöntemler**

Öte yandan hücre içinde ve kromozom üzerinde oluşmuş G-dörtlülerinin tespiti için henüz altın standart oluşturulamamıştır. Ancak 2015 yılında yeni nesil sekanslama teknolojisi polimeraz durdurma yöntemi ile birleştirilerek G4-seq adı verilen yeni bir yöntem geliştirilmiş ve insan genomunda 716,310 G-dörtlüsü tespit edilmiştir. (Chambers vd., 2015)

Yukarıda bahsedildiği gibi biyoinformatik metodlarla da G-dörtlüsü tahmini etmek mümkündür. Genom üzerinde G-dörtlüsü oluşabilmesi için eşit uzunluktaki kısa guanin tekrarlarından dört adet bulunması gereklidir. Bu guanin tekrarları, yada adacıkları, arasında baz çeşidinden bağımsız değişen uzunluklarda diziler bulunabilir. Guanin adacıkları arasında bulunan nükleotidler G-dörtlüsünün ilmiklerini oluşturur. (Wong vd., 2010) Ancak bu kurala uyan her dizi G-dörtlüsü oluşturamamaktadır. Bunun nedeni fizyolojik ortam şartlarında oluşan G-dörtlüsünün kararlılığı ile ilgilidir. Mesela guanin adacıklarının uzunluğu arttıkça istiflenecek G-tetrad sayısı arttığından kararlılığı da artırır. Veyahut, çok uzun ya da çok kısa ilmik uzunluğu G-dörtlüsü kararlılığını ciddi şekilde düşürebilmektedir. (Guédin vd., 2010a) Bu sınırlamalar göz önüne alındığında asgari guanin tekrarı 3 ve ilmik uzunluğu 1 ila 7 arasında değişen dizilerin fizyolojik şartlarda g-dörtlüsü oluşumu için en uygun parametreler olduğu görülmüştür. (Burge vd., 2006) Bu kural G3+N1- 7G3+N1-7G3+N1-7G3+ olarak da ifade edilir. Geliştirilmiş G-dörtlüsü tahmin programları genellikle bu kural etrafında kurulmuştur.

G-dörtlüsü tahmini için geliştirilen programlar kullandıkları algoritma ve parametrelere bağlı olarak ufak farklılıklar gösterebilmektedir. QGRS mapper ve nBMST gibi web tabanlı G-dörtlüsü tahmin araçlarının yanı sıra indirilebilir ve açık kaynaklı programlar da bulunmaktadır. (Cer vd., 2013; Kikin vd., 2006) Bunların içinde quadparser programı hem parametreleri değiştirme seçeneği sunmaktadır. (Wong vd., 2010) Quadparser ile yapılan taramalar sonucunda insan genomunda standart kurala uyan 376.446 dizi ya da potansiyel G-dörtlüsü (pG4) keşfedilmiştir. (Huppert ve Balasubramanian, 2005a) Yaptığımız çalışmalar sonucu aynı parametreler ile E.coli genomunda 52 adet pG4 tespit edilmiştir. Hem insan genomunda hem de E.coli'de G-dörtlüleri protein kodlamayan ve regülatör bölgelerinde yoğunlaşmıştır.

3. GEREÇ VE YÖNTEM

3.1 Kullanılan dil ve kütüphaneler

Kodlama için Python 2.14 ve 3.6 kullanılmıştır. Kullanılan standart python kütüphaneleri multiprocessing, time, sys, subprocess, os, intertools, difflib, string, random, re, regex, memory_profiler, argparse, __future__, operator'dür. Standart dışı olan numpy (1.14.2 and 1.13.3+mkl), h5py (2.8), psutil (5.4.6), scipy (1.1.0), colorama (0.4.1), numba (0.36.1) kütüphaneleri pip veya pycharm üzerinden yüklenmiştir.

3.2 G-dörtlüsü tarama çıktıları için dosya biçimi belirlenmesi

Farklı çıktı formatları kullanan quadparser (Huppert ve Balasubramanian, 2005b) ve G4Hunter (Bedrat vd., 2016) algoritmalarının aynı biçimde çıktı verebilmeleri için herhangi bir kıyaslama çıktıların tekrar işlenmesi gerektirdiğinden kaynak kodlarında ve 'deki görüldüğü gibi çıktı biçimi tekdüze hale getirilmiştir. Bu tekdüzeleştirilmiş biçime sahip G-dörtlüsü tarama rapor dosyası biçimi "G4List" dosya uzantısı ile iliştirilmiştir. Bu biçim 'deki şekilde ve sekme ayraçlı ("tab delimited") olarak tasarlanmıştır. Dosyanın ilk satırı G-dörtlülerinin keşfinde kullanılan algoritma ve parametrelerinin tanımı ve diğer tanımlayıcı bilgiler için ayrılmıştır. İlk satırdan itibaren sütunlar sırasıyla orijinal Fasta dosyasındaki tanım bilgisi, G-dörtlüsü ilk nükleotid pozisyonu, G-dörtlüsü son nükleotid pozisyonu, G-dörtlüsü oluşturan dizi uzunluğu, dizinin bulunduğu iplik, dizinin orijinal Fasta dizisini, G-dörtlüsü oluşturan diziyi ve son sütunda, varsa, G4Hunter puanı listelenmektedir.

Tablo . Örnek tekdüzeleştirilmiş G-dörtlüsü tarama rapor dosyası (G4List) biçimi

>python G4Catchall.py -I 1 -B --G3L 1..3 --XL 1..9 -F							
NC_0000X.1	1205	1220	15	-	CCCTCCCTCCCTCCC	GGGAGGGAGGGAGGG	-2.4
NC_0000X.1	1246	1261	15	+	GGGAGGGTGGGCGGG	GGGAGGGTGGGCGGG	2.567
…	…	…	…	…	…	…	…

Yukarıdaki biçim ile ilgili G-dörtlüsü hakkında muhtemel ihtiyaç duyulabilecek tüm bilgilerin hızla ulaşılabilmesi amaçlanmıştır ve dosya boyutu ikincil öneme sahiptir.

3.3 Toplu G-dörtlüsü tarama betiği geliştirilmesi

Taramaların yapıldığı dizilerin genişliği dolayısıyla kullanılan algoritmalar her izgede farklı bir Fasta dosyasını tarayacak şekilde çok izgeli çalışabilecek bir kod betiğine ihtiyaç duyulmuş ve geliştirilmiştir. () Bu amaçla os ve multiprocessing modülleri kullanılmıştır. Bu betik içinde bulunan klasör içerisindeki tüm "fa" soyadlı Fasta dosyalarını bulup tanımlanmış algoritma ve parametrelerde çalıştırmaktadır. Dosya yazımı çalıştırılan kod tarafından gerçekleştirilmektedir.

3.4 Var olan ihtiyaçları gidermek amacıyla G4Catchall algoritmasının geliştirilmesi

Var olan quadparser ve G4Hunter algoritmalarının yukarıda bahsedilen eksikliklerinden ötürü Python 2.7.14 dili kullanılarak yeni bir algoritma geliştirilmiştir. Bu dil ile beraber kullanılan modüller aşağıdaki gibi listelenmiştir. Algoritmanın Python kodu ’de verilmiştir.

Bu algoritma geliştirilirken daha önceki algoritmaların göz önüne almamış olduğu ancak deneysell olarak gösterilmiş olan G-tekrar bölgeleri içerisindeki çıkıntı ve eksiklikler ve aşırı uzun ilmik bölgelerini de kapsayacak şekilde tasarlanmıştır. Bunlarla beraber, kısa G-dörtlülerinin bu tür değişiklikleri tolere edemediği göz önünde tutularak kısa G-dörtlüleri için ayrı parametrelerin tanınmasına olanak verecek şekilde tasarlanmıştır.

3.4.1 **Algoritmaların kıyaslanması**

Algoritmaların kıyaslanması için daha önce G-dörtlüsü oluşturduğu daha önce tespit edilmiş 298 dizi ile G-dörtlüsü oluşturmadığı tespit edilmiş 94 dizi literatürden alınmıştır. (Bedrat vd., 2016) Her dizi FASTA formatında ve 28’er AT tekrarları arasına gömülerek çalışmalara devam edilmiştir.

Algoritmaların başarısını kıyaslamak için Youden’in J-endeksi kullanılmıştır. Bu indekse J=Gerçek Pozitif Oranı (GPO) -Yanlış Pozitif Oranı (YPO) şeklinde tanımlanmıştır. (Youden, 1950)

Quadparser , G4Hunter, PQSF (Hon vd., 2017) ve G4Catchall algoritmaları farklı parametreler ile daha önceden G-dörtlüsü oluşturup oluşturmadıkları belirlenmiş DNA dizileri ile taramaları sonucunda elde edilen GPO ve YPO değerleri kullanılarak Youden’in J-endeksi belirlenmiş ve sıralanmıştır. () Farklı parametrelerin taranarak J-endeksinin belirlenmesi için yazılmış edilmiş kod betiği ‘te bulunabilir.

3.5 **İnsan genomunun taranması**

Referans olarak belirlenmiş GRCh38.p7 sürüm numaralı insan genom derlemesine ait nükleer kromozom dizileri https://www.ncbi.nlm.nih.gov/assembly/707541’den indirilmiştir. Diziler G4Hunter, Quadparser ve G4Catchall ile taranmış ve sonuçlar kıyaslanmıştır. Kıyaslama için ListG4s.py dosyası kullanılmıştır. Sırasıyla kullanılan parametreler ListG4s içinde command değişkenine sırasıyla 'python quadparser.py ', 'python G4Hunter.py -w 25 -s 1.75 ' ve 'python G4Catchall.py -I 1 -B --G3L 1..3 --XL 1..9' atayarak taranmıştır. Her taramada tüm kromozom dosyalarından tahmin edilen G4’ler tek bir dosyada bölüm 3.3’de olduğu gibi kaydedilmiştir.

3.6 **Benzerlik Matrisi Oluşturulması**

“G4List” dosya uzantısı ile kaydedilmiş olan G-dörtlüsü dizileri arasındaki benzerliklerin bulunması, onların sınıflandırılması için önem arz etmektedir. Bu amaçla her G-dörtlüsünün birbiri ile dizisel benzerlikleri kıyaslanması yapılmıştır. Bu kıyaslama için çiftli hizalama (pairwise alignment) algoritmalarından diziler arasında global benzerlik yerine lokal benzerlik arayan Smith-Waterman yöntemi kullanılmıştır. (Smith ve Waterman, 1981) Lokal benzerlik aranmasının başlıca nedeni keşif algoritmalarının tümünün üst üste gelen G-dörtlülerini birleştirerek bulmalarındandır. Bu nedenle daha uzun gibi gözüken ancak birden çok G-dörtlüsü oluşturan diziler içerisindeki G-dörtlüleri ayrı ayrı kıyaslanmış olacaktır. Bu algoritma ile yapılan hizalama sırasında bulunan benzerlik puanı 0’dan başlar.

3.6.1 **Çiftli hizalama algoritması oluşturulması**

Bir insan genomunda 700.000’in üzerinde G-dörtlüsü olduğu tahmin edildiği göz önüne alındığında her birinin birbiri ile kıyaslanması işlemci açısından uzun ve zahmetli bir iş olduğu açıktır. Python modülleri arasında bilinirliği yüksek olan BioPython modülü (Cock vd., 2009) içerisinde çiftli hizalama algoritması bulunmaktadır, ancak bu adımın hızlandırılması amacıyla çiftli hizalama algoritması numba’nın just-in-time derleme özelliği kullanılarak tekrar yazılmıştır. (Lam vd., 2015) İlgili kod ‘te verilmiştir.

Her G-dörtlüsü çifti için benzerlik çiftli hizalama algoritması için literatürde sıkça tercih edilen, eşleşme için +2, eşleşmeme için -1, boşluklar için -2 puan parametreleri ile kullanılmıştır. Bu değerler ihtiyari olmakla beraber benzeri çiftli hizalama algoritmalarında BLAST algoritması da dahil olmak üzere sıkça kullanılmaktadırlar. (BLAST, 2013; Xia, 2007) G-dörtlüleri için ayrı bir parametre seti henüz gösterilmemiştir. Çiftli hizalama sonucunda elde edilen puanlar, kısa dizinin kendisi ile elde edeceği puana bölümü ile normalize edilmiştir. Bu da “floating point” cinsinden 0 ila 1 arasında bir değer anlamına gelmektedir. Diğer bir deyişle birbirine yüksek benzerlik gösteren diziler arasındaki benzerlik puanı 1’e yakın, yüksek farklılık gösteren diziler arasındaki benzerlik puanı 0’a yakın olarak belirlenmiş olacaktır.

Benzerlik matrisi için gerekli matris boyutu her matris elementinin float cinsinden 24 baytlık bir değer olacağı ve genomdaki G-dörtlüsü sayısının karesi (yaklaşık 700.000x700.000) boyutunda olacağı göz önüne alındığında, yaklaşık 11 terabaytlık bir depolama alanı gerektirmektedir. Bu matrisin büyük çoğunluğunun birbirlerinden çok farklı dizilerin benzerlik değerlerinin “0”’a yakın olacağı ve herhangi bir şekilde gruplanamayacakları göz önüne alınarak bir eşik değeri altında kalan dizilerin benzerlikler 0 olarak belirlenmiş ve bu “0”ların matris içerisinde yer kaplamaması için “sparse” (diğer bir değişle, seyrek) matris kullanılması tercih edilmiştir. Bunun için işlemci tarafından bulunan benzerlikler CSR (“Compressed Sparse Row” matris) ve COO

(“COOrdinate” matris) seyrek matrisleri olarak bellekte ve hafızada kaydedilmiştir. (Fineman vd., 2009)

Böyle büyük bir matrisin doldurulması uzun süre alacağından aynı anda 32 işlemciyi de kullanacak şekilde çok izgeli kodlanarak seyrek matrislerin oluşturulması birkaç güne kadar kısaltılmıştır. İlgili koda ‘de ulaşılabilir. Kod çıktısı “coo” ve “csr” uzantılı dosyalara h5py modülü ile kaydedilerek depolanır. Veriler “parameters”, “data”, “rows” ve “cols” veri seti başlıkları kaydedilmiş olup aynı isimler kullanılarak bu dosyalardan h5py modülü yardımı ile çekilebilir.

3.7 **Sınıflandırma**

Graf teorisine uyumlu olarak G-dörtlüsü bir verteks ve benzerlik matrisinde eşik değerin üzerinde olan her değer ilgili verteksler arası bir bağlantı olacak şekilde bir ağ elde edilebilir. Bu durumda ifade edilen ağın oluşturulması önce “coo” uzantılı dosyadan okuma yapmaktadır. Okunan matris bilgisi verteksler ve bağlantılar olmak üzere iki listeye ayrılarak işlenir. Öncelikle her verteks için en büyük klik (clique) keşfedilir. Ardından her klik’in vertekslerinin bağlantılı olduğu tüm komşuları birer birer değerlendirilir. Bu değerlendirmede bulunan her klikin vertekslerinin komşuları birer birer klikteki diğer bağlı oldukları vertekslerin sayısı tüm vertekslere bölünerek bulunan oran 0.8’nin üzerinde olması durumunda klike eklenmek suretiyle gruplar oluşturulur. Son aşamada bulunan tüm gruplar birbirleri ile ortak verteks sayıları toplam verteks sayısının 0.8’ini aşması şartıyla birleştirilmiştir. Bu gruplar “G4clstr” uzantılı dosyalarda ilgili G-dörtlülerinin orijinal “G4List” dosyasındaki satır indeksleri sıralanarak kaydedilmiştir. Her satırda ilgili grubun G-dörtlüsü indeksleri bulunmaktadır. ()

Tablo . Örnek G-dörtlüsü sınıflandırma rapor dosyası (G4clstr) biçimi

```
> Initial G4 discovery command: >python G4Catchall.v0.5.py -I 1 -B --G3L 1..3 --XL 1..9 -…
{96, 795908, 791012, 5, 466503, 497192, 567562, 100174, 350136, 567576}
{162721, 175170, 380731, 430566, 13832, 433901, 126990, 594259, 86, 472535, 169942, 145177, 548315}
…
…
```

İlgili kod ‘de verilmiştir. İşlemin hızlandırılması adına çok izgeli şekilde multiprocessing modülü ile desteklenmiştir.

3.8 **Motiflerin bulunması**

Her G-dörtlüsü sınıfı içerisindeki G-dörtlüleri doğal olarak birbirlerine büyük benzerlik göstermektedirler. Bu benzerliklerin dizinin neresinde olduğu, hangi bazların ortaklık gösterdiği gibi bilgileri sunulması ve diğer aramalarda kullanılabilmesi için motif haline çevrilmeleri gereklidir. Bu amaçla Glam2 yazılımı Linux ortamında yüklenip derlenmiştir. (Frith vd., 2008) Glam2 programı komut satırından motife ait dizilerin bulunduğu bir dosyayı işleyerek çıktı olarak

bir motif ve motife ait “png” uzantılı görsel bir sunum dosyası oluşturmaktadır. Ancak tüm grupların hızla işlenerek otomatize edilmesi için bir aracı kod betiği ‘deki gibi yazılmıştır. Otomatizasyon “G4clstr” ve “G4List” uzantılı dosyalardan çekilerek oluşturulan geçici ham dosyanın komut satırından Glam2’ye sunulması ile gerçekleşir. Komut satırından kullanılan parametreler “glam2 n geçici_dosya_ismi -o hedef_klasör_ismi -r 1" şeklindedir.

1.1 **G-dörtlülerin genomik gen anotasyonlarına göre filtrelenmesi**

Bu nedenle bu bölgelerde dolayısıyla regülasyonda rolü olan G-dörtlerin elenmeleri için GRCh38 referans genomu üzerindeki gen anotasyonları Ensembl (ftp://ftp.ensembl.org/pub/release-94/gtf/homo_sapiens/Homo_sapiens.GRCh38.94.gtf.gz) ve NCBI veri bankalarından (ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/GFF/ref_GRCh38.p12_top_level.gff3.gz) elde edilmiş, bu iki veri bankasına göre de sadece bir transkripsiyon başlama noktasından -2000 baza kadar olan promotor bölgede, 5’UTR yada 3’UTR bölgede olan her hangi bir veri bankasında genin başka bir bölgesi ile çakışmadığı taktirde seçilerek kaydedilmiştir. Bu filtreleme için ‘deki kod betiği yazılarak kullanılmıştır. Bu kod üç farklı değişken seti ile kullanılmış olup bu setler ‘de belirtilmiştir.

Tablo . G-dörtlüsü filtreleme betiği içinde regülasyon ile ilişkili G-dörtlülerin seçiliminde kullanılmış değişken değerleri

Değişken değerleri	**Çıktı dosyası**
rules.promoter=option.ignore rules.gene=option.ignore rules.five_prime_utr=option.ignore rules.three_prime_utr=option.must_include rules.CDS=option.must_exclude rules.pseudogene=option.must_exclude	Homo sapiens, GRCh38.p7 Primary Assembly_3UTR.G4List
rules.promoter=option.must_include rules.gene=option.must_exclude rules.five_prime_utr=option.ignore rules.three_prime_utr=option.ignore rules.CDS=option.must_exclude rules.pseudogene=option.must_exclude	Homo sapiens, GRCh38.p7 Primary Assembly_Promoter.G4List
rules.promoter=option.ignore rules.gene=option.ignore rules.five_prime_utr=option.must_include rules.three_prime_utr=option.ignore rules.CDS=option.must_exclude rules.pseudogene=option.must_exclude	Homo sapiens, GRCh38.p7 Primary Assembly_5UTR.G4List

3.9 **Filogenetik ağaç oluşturulması**

Filogenetik programları tarafından kullanılma uygun formata çevrilmesi amacıyla ‘deki kod betiği ile “G4List” dosyaları fasta formatına çevrilerek “G4List.fa” uzantılı olarak kaydedilmiştir.

G4List’ten Fasta formatına çevrilmiş dosyalardan MAFFT çoklu dizi hizzalama uygulamasında “--auto --treeout --localpair --reorder" parametreleri ile kullanılarak “G4List.tree” uzantılı newick filogenetik ağaçları elde edilmiştir. (Katoh ve Standley, 2013) Kullanılan yazılım uygulamanın 7.409 nolu sürümü olup boşluk açma puanı (gap opening penaly) -1.53, boşluk uzatma puanı (gap extension penalty) 0 olarak varsayılan değerlerde kullanılmıştır.

3.10 **Varyasyon taraması**

Varyasyon taramasının ilk aşamasında İnsan genomu için en geniş varyasyon veritabanlarından biri olan ClinVar kullanılmıştır. (Landrum vd., 2018) Bu veritabanının tercih edilmesinin en önemli nedeni hastalıklar ile ilişkilendirilmiş varyasyonların etiketlenmiş ve filtrelenebilir olmasıdır. (Landrum vd., 2016) İlgili varyantlardan patojenik (“pathogenic”) ve yüksek ihtimalle patojenik (“highly pathogenic”) olanlar seçilmiştir. GRCh38.p7 insan genomunda hizalanmış varyant bilgileri “tsv” uzuantılı olarak “http://www.ncbi.nlm.nih.gov/variation/view/”’dan her kromozom için ayrı ayrı indirilmiştir. Varyantlardan pozisyonları G-dörtlüsü pozisyonları ile çakışan varyantlar ‘deki kod betiği ile filtrelenmiştir.

ClinVar varyasyon veri bankasına ek olarak TCGA veri bankasında kanserlerle ilişkilendirilmiş varyasyonlar sunulmaktadır. TCGA 45 ayrı projeden elde edilmiş ve kanser tipleri ile ilişkilendirilmiş varyasyonları “https://portal.gdc.cancer.gov/” adresinde kullanıcılara sunmaktadır. (Grossman vd., 2016) TCGA veri bankasından tüm mutasyonlar ile ilgili veriler “JSON” uzantılı olarak indirilmiş, ardından ‘deki kod betiği ile sadece “5’UTR”, “3’UTR”, “upstream”, “downstream” ve “intron” moleküler netice etiketi içeren mutasyonlar filtrelenip, “csv” formatında kaydedilmiştir. G-dörtlüleri ile çakışan varyasyonlar ‘deki kod betiği ile filtrelenmiştir.

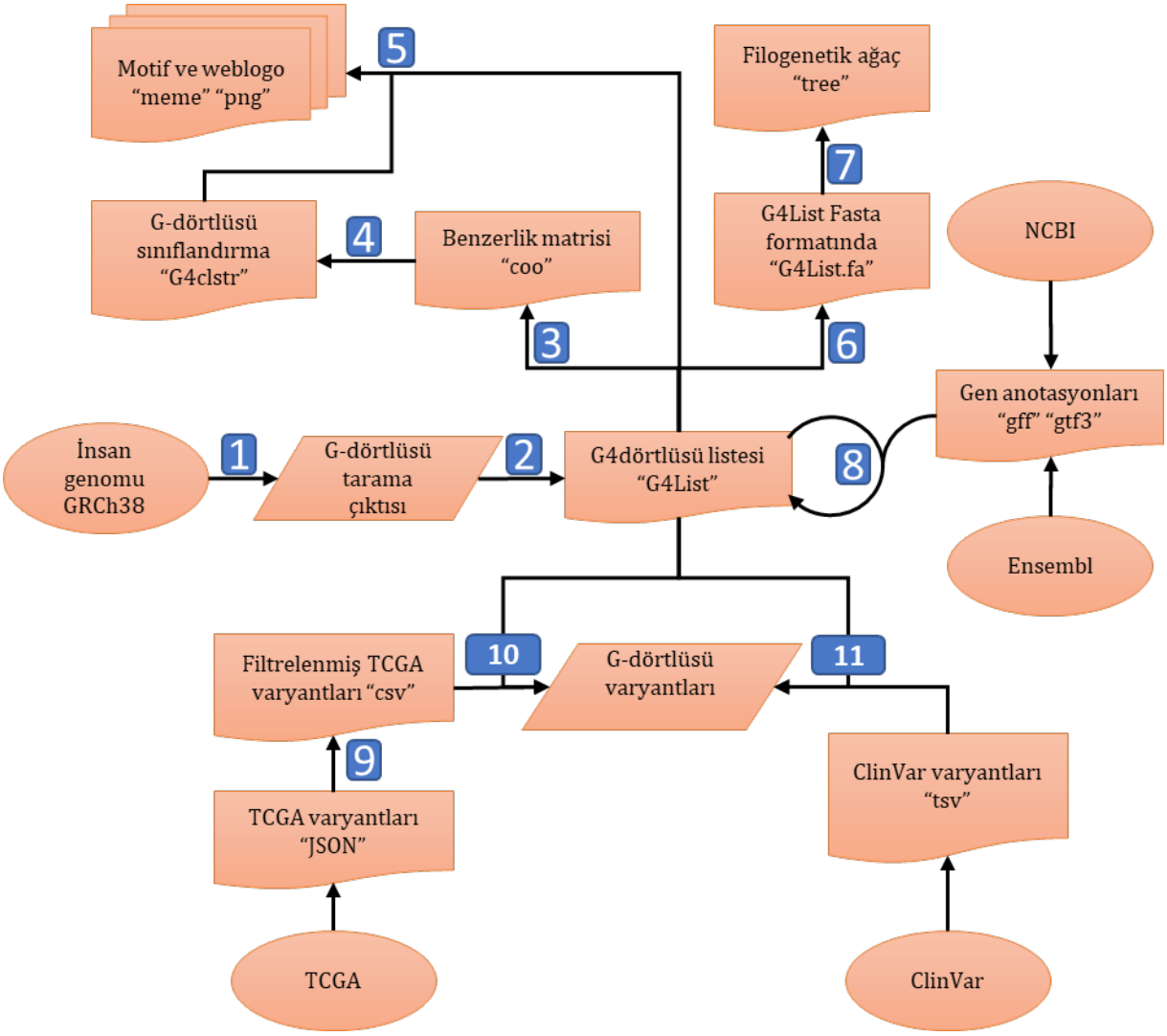
3.11 **Varyasyonların seçimi ve Dairesel Dikroizm ile varyasyon etkisi tespiti**

G-dörtlüleri ile ilişkilendirilmiş varyasyonların yapı üzerine etkisinin çalışılması için özellikle kısa G-dörtlülerinde ve G-dörtlülerinin iç kısmında gerçekleşen varyasyonlar tercih edilmiştir. Bu amaçla G-dörtlülerinin ilk ve son bazlarında var olan varyasyonlar ile G-dörtlüsü oluşturan dizilerin boyutu 40 bazın üzerinde olanlar hariç tutulmuştur. Referans genotip ile varyasyonları oluşturan DNA dizileri 100 nmol skalada ve kartuş ile saflaştırılmış olarak temin edilmiştir. Temin edilen diziler 10mM Tris HCl (pH 7.5) ve 100 mM KCl ile önce 90 °C’de 10 dk inkübe edilip, 16 saat boyunca oda sıcaklığında soğumaya bırakılmıştır. Dairesel Spektrofotometre ile 350 nm ile

200 nm arasında 0.1 nm veri aralıklarında ölçüm yapılmıştır. Varyantlara ait ölçümler referans spektrumdan çıkarılarak elde edilen eğri altı alanın referans eğri altı alanına bölünerek değişim orantısal olarak hesaplanmıştır.

4. **BULGULAR**

G-dörtlülerinin analizi için oluşturulmuş akış şeması ’de gösterilmiş olup akış adımlarının bulguları aşağıda başlıklar altında sıralanmıştır. Her adım için yazılmış olan kod betiği Eklerde listelenmiş ve ilgili adım ile ilişkilendirilmiştir.



Şekil . Proje akış şeması. Adımlar ile iliştirilmiş kodlar şu şekildedir. Adım 1: , yada . Adım 2: . Adım 3: ve . Adım 4: . Adım 5: ve Glam2. Adım 6: . Adım 7: MAFFT. Adım 8: . Adım 9: . Adım 10: . Adım 11: .

4.1 **Var olan ihtiyaçları gidermek amacıyla yeni algoritma geliştirilmesi**

Proje kapsamında G-dörtlüsü tahmin algoritmalarının kullanılarak insan genomundaki G-dörtlülerinin tahmini ilk adımı oluşturmaktadır. Ancak bulgular var olan G-dörtlüsü tahmin algoritmalarının yetersizliğini ortaya koymuştur. Bunlar özellikle G-dörtlülerinini varsayılan dokusuna uymayacak şekilde G-dizisi içerisinde çıkıntı veya eksik guanin bazları (Kaluzhny vd.,

2009; Mukundan ve Phan, 2013; Tomaško vd., 2009; Varizhuk vd., 2017) ve ilmik bölgelerinin beklenenden çok daha yüksek uzunluklarda olabileceğine (Guédin vd., 2010b; Onel vd., 2016) dair bulgular olmuştur. Bu sıradışı G-dörtlüleri göz önünde alınarak, G-dörtlüsü tahmini algoritmalarının eksikliklerini kapatmak amacıyla G4Catchall algoritması geliştirilmiştir. () Bu algoritmanın başarısını kıyaslamak amacıyla G-dörtlüsü oluşturup oluşturmadığı biyofiziksel yöntemlerle tespit edilmiş olan 392 dizi G4Catchall, quadparser (Huppert ve Balasubramanian, 2005b), G4Hunter (Bedrat vd., 2016) ve PQSF (Hon vd., 2017) algoritmaları ile analiz edilmiştir. Her algoritma için doğru sınıflandırma oranları gerçek pozitif oranı (GPO) ve yanlış pozitif oranı (YPO) toplamı olan Youden'in J-endeksi ile hesaplanmıştır. () Quadparser, G4Hunter ve PQSF algoritmaları en çok tercih edilen parametrelerinde test edilirken G4Catchall için çeşitli parametreler kullanılarak en uygun parametrenin bulunması da hedeflenmiştir.

Tablo . Kıyaslanan algoritmaların değişik parametreler için GPO, YPO ve Youden'nin J istatistik indeksleri.

Algoritma ve parametreleri	**J-endeksi**	**En yüksek J-endeksi için**				
		GPO	**YPO**	**G_{3+}GQ için azami ilmik**	**G_2GQ için azami ilmik**	**Azami aşırı ilmik**
G4C G2 + E3 + I2BM	0.83	0.99	0.16	6	3	≥30
G4C G2 + E3 + I2B	0.88	0.97	0.09	≥12	3	≥30
G4C G2 + E3 + I1BM	0.81	0.97	0.16	≥6	3,4	≥30
G4C G2 + E3 + I1B	0.84	0.97	0.13	≥8	4	≥30
G4C G2 + E3 + I0	0.82	0.95	0.13	≥9	4	≥30
G4C G2 + E2 + I2BM	0.82	0.98	0.16	6	2	≥30
G4C G2 + E2 + I2B	0.85	0.96	0.11	≥7	2	≥37
G4C G2 + E2 + I1BM	0.86	0.97	0.11	≥6	2	≥37
G4C G2 + E2 + I1B	0.85	0.96	0.11	≥7	2	≥37
G4C G2 + E2 + I0	0.84	0.95	0.11	≥9	2	≥37
G4C G2 + E-I2BM	0.77	0.93	0.16	≥9	3	–
G4C G2 + E-I2B	0.79	0.9	0.13	≥12	3,4	–
G4C G2 + E-I1BM	0.75	0.91	0.16	≥9	3,4	–
G4C G2 + E-I1B	0.78	0.91	0.13	≥12	4	–

Algoritma ve parametreleri	J-endeksi	En yüksek J-endeksi için				
		GPO	YPO	$G_{3+}GQ$ için azami ilmik	G_2GQ için azami ilmik	Azami aşırı ilmik
G4C G2 + E-I0	0.76	0.89	0.13	≥12	4	–
G4C G3 + E3 + I2BM	0.83	0.96	0.13	6	–	≥30
G4C G3 + E3 + I2B	0.91	0.93	0.02	≥12	–	≥31
G4C G3 + E3 + I1BM	0.87	0.92	0.04	≥9	–	≥30
G4C G3 + E3 + I1B	0.91	0.91	0	≥8	–	≥31
G4C G3 + E3 + I0	0.82	0.82	0	≥9	–	≥31
G4C G3 + E-I2BM	0.77	0.9	0.13	≥9	–	–
G4C G3 + E-I2B	0.83	0.85	0.02	12	–	–
G4C G3 + E-I1BM	0.81	0.86	0.04	≥14	–	–
G4C G3 + E-I1B	0.84	0.84	0	≥12	–	–
G4C G3 + E-I0	0.76	0.76	0	≥13	–	–
QP $G_{2+}N_{\leq 30}$	0.85	0.99	0.18	30	30	n/a
QP $G_{3+}N_{\leq 30}$	0.76	0.81	0	30	n/a	n/a
G4H w = 20 s = 1.2	0.73	0.90	0.16	n/a	n/a	n/a
PQSF	0.88	0.97	0.09	n/a	n/a	n/a

Bulgular J-endeksi cinsinden en yüksek başarının 0.91 ile G4Catchall algoritması içinde G4C G3 + E3 + I2B ve G4C G3 + E3 + I1B modelleri için gösterilmiş olup, bu modellerin her ikisindeki ortak parametreler 1. guanin tekrarlarının en az 3 baz uzunlukta olması, 2. İlmik bölgelerinden birinin aşırı uzun olabilmesi, 3. sırasıyla, bir yada iki adet çıkıntı bulunan guanin tekrarını kabul etmeleri ve 4. guanin tekrarlarında çıkıntı haricinde eksik guanin içerenlerin reddedilmesi olduğu görülmektedir. Bu algoritmalarda ilmik bölgelerinin uzunluklarının, sırasıyla, 12 ve 8 baza kadar olmaları ve aşırı uzun ilmiğin 30 baza kadar olması durumunda bu modeller için azami J-endeksinin elde edildiği görülmüştür.

Bu parametreler kullanıldığında var olan algoritmalara kıyasla daha başarılı sonuçların elde edildiği de gösterilerek elde edilen bulgular yayınlanmıştır. (Doluca, 2019)

4.2 **İnsan genomlarının taranması**

G4Catchall algoritmasının $\mathrm{G4C\ G3+E3+I1B}$ kodlu modeli kullanılarak GRCh38.p7 sürümlü referans genom üzerinde yapılan analiz sonucu 795916 G-dörtlüsü oluşturan dizi keşfedilmiştir. Dizilerin tamamı http://homes.ieu.edu.tr/odoluca/Archive/HumanG4analysis/ sitesinde "completeGenome" klasörü altında "G4List" dosya uzantılı olarak ulaşıma açılmıştır.

4.3 **G-dörtlülerin sınıflandırılması**

G-dörtlüleri tek bir üç boyutlu yapı oluşturmamaktadırlar. G-dörtlüleri oluşturan zincirlerin yönlerine göre paralel, anti-paralel ve hibrid olmak üzere veya ilmiklerin bağlanma şekillerine göre pervane, sandalye, sepet olmak üzere ayrılabilirler. Bu yapısal farklılıkların iki temel kaynağı vardır, çevresel faktörler ve dizi farklılıkları. *İn vitro* ortamda çevresel faktörlerin benzer olduğu göz önüne alınırsa, en önemli faktör dizi farklılıkları olup, buna bağlı olarak 3-boyutlu şekilleri, ve dolayısı ile etkileşim gösterebilecekleri diğer biyomoleküller de değişim göstereceklerinden, benzer yapıda olan G-dörtlülerin benzer fonksyonlara sahip olacağı beklenmektedir. Gerçekten, bu çeşit yaklaşım ile proteinler arasında diziye bağlı olarak sınıflandırılma ve fonksiyonların ortaya çıkarılması mümkün olmaktadır. (Jensen vd., 2003) Benzer şekilde G-dörtlülerin yapısal ve dolayısı ile fonksiyonel benzerliklerinin ortaya çıkarılması için dizilerine göre benzerlerin keşfi, gruplanması ve bilim dünyasına kazandırılması gerekmektedir.

4.3.1 **Benzerlik Matrisi Oluşturulması**

G-dörtlülerinin sınıflandırılması için her G-dörtlüsü birbiri ile Smith-Waterman algoritması kullanılarak kıyaslanmış ve kıyaslama puanları ilgili dizilerin alabileceği en yüksek değere bölünerek bir benzerlik matrisi oluşturulmuştur. () Bu benzerlik matrisi oluşturulurken, matris büyüklüğü nedeniyle belli bir eşik değerin altındakiler 0 olarak alınmış, bu sayede var olan bilgisayar hafızasında tutulamayacak kadar büyük olmasının önüne geçilmiştir. Ortaya çıkan matris "COO" ya da "CSR" dosya uzantılı olarak h5py python modülü ile kaydedilmiştir. Boyutları nedeniyle (>4.3 GB) matris dosyalarının ulaşıma açılabileceği sunucu kaynaklarına ulaşılamamıştır.

4.3.2 **Sınıfların oluşturulması**

Her G-dörtlüsü benzerlik matrisinde bulunan benzerleri ile bir G-dörtlüsü sınıfının üyesi olabilir. Bu sınıfların ortaya çıkarılması bir topluluk keşfi ("community discovery") problemi olarak değerlendirilebilir. (Clauset vd., 2004) Bu amaçla G-dörtlüleri yöntem kısmında verildiği şekilde CFinder benzeri bir algoritma ile sınıflandırılmıştır. (Adamcsek vd., 2006) İlgili kod betiği 'de bulunabilir. Sınıflandırma sonucu

“http://homes.ieu.edu.tr/odoluca/Archive/HumanG4analysis/completeGenome/” adresinde “G4clstr.7z” dosya uzantılı olarak sıkıştırılarak kaydedilmiş ve ulaşıma açılmıştır.

4.3.3 **Motiflerin bulunması**

Keşfedilen G-dörtlüsü sınıflarını temsilen birer motif elde edilmesi ve elde edilen motiflerin dizi logosu şeklinde sunumlarının oluşturulması için Glam2 yazılımında yararlanılmıştır. Glam2 ile diziler “meme” motif formatında kaydedilmiş ve ayrıca görsel olarak kolay anlaşılır “png” dosya uzantılı dizi logoları ile beraber “http://homes.ieu.edu.tr/odoluca/Archive/HumanG4analysis/completeGenome/” adresinde “HumanG4ClstrMotifs.7z” dosya ismi ile sıkıştırılmış olarak ulaşılabilmektedir. Ayrıca iki örnek motif aynı klasör altında listelenmiştir.

Toplamda 5261 G-dörtlüsü sınıflandırılmış ve bu sınıfların kendi içlerindeki toplam varyasyon 69417 olarak kaydedilmiştir. Bu varyasyon sayısı her grubun içerisindeki birbirinden farklı dizilerin sayısını ifade ettiği not edilmelidir.

4.4 **G-dörtlülerinin genetik lokasyona göre filtrelenmesi**

Tespit edilmiş olan G-dörtlülerinin bir kısmı genler arasında iken, bir kısmı promotor, bir kısmı intron ve bir kısmı eksonlarda bulunabilmektedir. G-dörtlülerinin bulundukları pozisyona göre etkileri farklı olacaktır. Özellikle G-dörtlülerinin promotor ve UTR bölgelerinde bulunanların regülasyonda rol alabileceği gösterilmiştir. (Bay vd., 2017; Beaudoin ve Perreault, 2010; Bugaut ve Balasubramanian, 2012; Jodoin vd., 2014; Onel vd., 2016; Siddiqui-Jain vd., 2002) Bu amaçla var olan NCBI ve Ensembl veritabanlarındaki anotasyon bilgileri kullanılarak sadece promotor yada 5’UTR yada 3’UTR bölgelerinde var olan G-dörtlüleri seçilmiş ve seçilen G-dörtlüleri ayrı ayrı “G4List” uzantılı olarak kaydedilmiştir.

4.5 **Filogenetik ağaç oluşturulması**

Bu üç farklı lokasyonda bulunan G-dörtlüleri kendi içlerinde daha önceki çalışmamızda gösterdiğimiz gibi filogenetik ağaç oluşturularak benzerliklerine göre gruplanmıştır. (O.I. Kaplan vd., 2016) İlgili filogenetik ağaçlar MAFFT çoklu dizi hizalama programı ile newick veri formatında, “tree” dosya uzantılı olarak elde edilmiştir. (Katoh vd., 2009; Katoh ve Standley, 2013) “http://homes.ieu.edu.tr/odoluca/Archive/HumanG4analysis/regulatoryG4s/” adresinde ulaşıma açılmıştır.

4.6 **Varyasyon taraması ile hastalıklarla ilişkilendirilmesi**

Tespit edilmiş varyasyonlardan hastalıklarla ilişkilendirilmiş olanlar büyük oranda protein kodlayan bölgede bulunmaktadır. Bu bölgelerde olan varyasyonlar G-dörtlüsü yapısında

değişiklik yapmaktan ziyade amino asit dizisinde değişim yaparak hastalıkların moleküler temelini oluşturmaktadır. Bu nedenle özellikle protein kodlamayan bölgedeki varyasyonlar, promotor, 5'UTR ve 3'UTR bölgelerindeki varyasyonlar yada intron bölgelerindeki varyasyonlar G-dörtlüsü yapısını bozarak regülasyon yada alternatif uçbirleştirmeyi etkilenmesine neden olabilir.

G-dörtlülerin ClinVar veritabanı ile kıyaslanması sonucu 5375 varyasyon G-dörtlüsü içerisinde bulunması nedeniyle yapısal değişim yolu ile etki gösterebileceğini işaret etmektedir. Bu varyasyonlar "http://homes.ieu.edu.tr/odoluca/Archive/HumanG4analysis/completeGenome/" adresinde ClinVar tarafından ilişkilendirilmiş hastalıklar ile beraber "ClinVar_G4Variants.tsv" dosyasında ulaşılabilmektedir.

G-dörtlülerin TCGA veritabanı ile kıyaslanması sonucu kanser tipleri ile ilişkilendirilmiş 12730 varyasyon G-dörtlüsü içerisinde bulunması nedeniyle yapısal değişim yolu ile etki gösterebileceğini işaret etmektedir. İlgili mutasyonlardan 210 tanesi 0.01'in altında p-değeri göstermektedir. İlgili mutasyonlar, ilişkilendirilmiş oldukları hastalıklar ve p-değerleri "http://homes.ieu.edu.tr/odoluca/Archive/HumanG4analysis/completeGenome/" adresinde "GDC_DiseaseAssociatedG4Variants.tsv" dosyasında ve 'te ulaşılabilmektedir.

Tablo . G-dörtlüsü oluşturan dizilerde hastalıklarla ilişkilendirilmiş TCGA varyasyonları ve ilişkilendirildiği hastalıklar

varyasyon	İlişkilendirilmiş hastalık	z-skoru	p-değeri
chr1:g.154963387delG	Uterine Corpus Endometrial Carcinoma	10.15597	1.56E-24
chr10:g.47461391G>A	Cholangiocarcinoma	3.523893	0.000213
chr9:g.69012836delG	Uterine Corpus Endometrial Carcinoma	8.853031	4.26E-19
chr11:g.558069delG	Stomach adenocarcinoma	7.277113	1.71E-13
chr6:g.31541264delG	Colon adenocarcinoma	6.202061	2.79E-10
chr1:g.27306631delC	Uterine Corpus Endometrial Carcinoma	7.117152	5.51E-13
chr3:g.185644590delGGG	Uterine Corpus Endometrial Carcinoma	6.439626	5.99E-11
chr17:g.7260211delC	Stomach adenocarcinoma	6.252684	2.02E-10
chr1:g.201216043delG	Uterine Corpus Endometrial Carcinoma	5.875173	2.11E-09
chr19:g.49103493delC	Uterine Corpus Endometrial Carcinoma	6.309077	1.40E-10
chr18:g.70326559delC	Colon adenocarcinoma	5.667184	7.26E-09
chr21:g.33028236delG	Uterine Corpus Endometrial Carcinoma	6.309077	1.40E-10
chr17:g.70178658delG	Colon adenocarcinoma	6.4628	5.14E-11
chr11:g.113276836delG	Uterine Corpus Endometrial Carcinoma	6.269857	1.81E-10
chrX:g.49881931delG	Colon adenocarcinoma	6.558845	2.71E-11
chr8:g.115668684delC	Uterine Corpus Endometrial Carcinoma	6.254488	1.99E-10
chr17:g.43400345delC	Stomach adenocarcinoma	5.296869	5.89E-08
chr13:g.83881637delG	Colon adenocarcinoma	4.890003	5.04E-07
chr3:g.43366832delC	Uterine Corpus Endometrial Carcinoma	5.539918	1.51E-08

varyasyon	İlişkilendirilmiş hastalık	z-skoru	p-değeri
chr15:g.70079385delGGGCTGGG	Cervical squamous cell carcinoma and endocervical adenocarcinoma	4.746703	1.03E-06
chr5:g.37752629_37752630insT	Uterine Corpus Endometrial Carcinoma	5.101053	1.69E-07
chr17:g.39210647delG	Cholangiocarcinoma	2.843806	0.002229
chr16:g.67828184delG	Stomach adenocarcinoma	4.819627	7.19E-07
chrX:g.74304786delC	Uterine Corpus Endometrial Carcinoma	3.741921	9.13E-05
chr7:g.72948857_72948858insC	Head and Neck squamous cell carcinoma	3.813132	6.86E-05
chr12:g.79934965delG	Uterine Corpus Endometrial Carcinoma	5.101053	1.69E-07
chr6:g.34022043T>C	Cervical squamous cell carcinoma and endocervical adenocarcinoma	3.89452	4.92E-05
chr22:g.39491058delC	Colon adenocarcinoma	4.328531	7.51E-06
chr11:g.45649457G>T	Uterine Corpus Endometrial Carcinoma	5.437186	2.71E-08
chr22:g.26433597delC	Uterine Corpus Endometrial Carcinoma	4.594457	2.17E-06
chr5:g.141646589delC	Colon adenocarcinoma	4.328531	7.51E-06
chr4:g.13484184delG	Colon adenocarcinoma	5.87231	2.15E-09
chr15:g.72346216delC	Stomach adenocarcinoma	4.903269	4.71E-07
chr9:g.88994233delC	Colon adenocarcinoma	4.989518	3.03E-07
chr6:g.38154374delG	Uterine Corpus Endometrial Carcinoma	5.192727	1.04E-07
chr8:g.26291737delC	Colon adenocarcinoma	5.510565	1.79E-08
chr1:g.23344301delC	Uterine Corpus Endometrial Carcinoma	4.708559	1.25E-06
chrX:g.154021864delGG	Uterine Corpus Endometrial Carcinoma	5.192727	1.04E-07
chr5:g.69369558G>A	Skin Cutaneous Melanoma	4.830998	6.79E-07
chrX:g.64189800delC	Colon adenocarcinoma	5.271162	6.78E-08
chr19:g.44759623delG	Colon adenocarcinoma	3.681341	0.000116
chr9:g.20622296delC	Uterine Corpus Endometrial Carcinoma	4.67985	1.44E-06
chr6:g.24357373delG	Uterine Corpus Endometrial Carcinoma	4.67985	1.44E-06
chrX:g.49881949delG	Colon adenocarcinoma	5.271162	6.78E-08
chr3:g.185644590delGGGG	Uterine Corpus Endometrial Carcinoma	4.908608	4.59E-07
chr17:g.64897523C>T	Thymoma	4.044793	2.62E-05
chr5:g.69369559G>A	Skin Cutaneous Melanoma	4.666384	1.53E-06
chr22:g.38822064G>C	Uterine Corpus Endometrial Carcinoma	4.908608	4.59E-07
chr1:g.43966517delG	Uterine Corpus Endometrial Carcinoma	3.278065	0.000523
chr15:g.66703228delC	Colon adenocarcinoma	5.083987	1.85E-07
chr1:g.22091681delC	Cholangiocarcinoma	3.232049	0.000615
chr2:g.127865063delG	Uterine Corpus Endometrial Carcinoma	4.354226	6.68E-06
chr1:g.244056658delC	Colon adenocarcinoma	4.567462	2.47E-06
chr3:g.189789735delG	Colon adenocarcinoma	4.567462	2.47E-06
chr8:g.141141415delG	Stomach adenocarcinoma	4.437284	4.56E-06
chr18:g.46457315delG	Uterine Corpus Endometrial Carcinoma	4.354226	6.68E-06
chr17:g.81166699delC	Uterine Corpus Endometrial Carcinoma	4.570787	2.43E-06
chr22:g.45200965delG	Uterine Corpus Endometrial Carcinoma	4.354226	6.68E-06
chr2:g.219550422delG	Uterine Corpus Endometrial Carcinoma	4.354226	6.68E-06
chr3:g.49108389delC	Uterine Corpus Endometrial Carcinoma	4.181376	1.45E-05

varyasyon	İlişkilendirilmiş hastalık	z-skoru	p-değeri
chr19:g.53963064delC	Kidney renal papillary cell carcinoma	4.289728	8.94E-06
chr19:g.17279140delG	Uterine Corpus Endometrial Carcinoma	4.181376	1.45E-05
chr12:g.54369973delG	Thymoma	2.551119	0.005369
chr22:g.40323724T>G	Sarcoma	3.515914	0.000219
chr14:g.77024793delC	Uterine Corpus Endometrial Carcinoma	4.125398	1.85E-05
chrX:g.17635452delC	Uterine Corpus Endometrial Carcinoma	4.156716	1.61E-05
chr6:g.31271541G>C	Pancreatic adenocarcinoma	3.825946	6.51E-05
chr19:g.45765433delC	Uterine Corpus Endometrial Carcinoma	3.558035	0.000187
chr6:g.21596484delC	Colon adenocarcinoma	4.497621	3.44E-06
chr5:g.3601065delG	Colon adenocarcinoma	2.893645	0.001904
chr6:g.43777706delG	Uterine Corpus Endometrial Carcinoma	3.558035	0.000187
chr19:g.40215422delG	Uterine Corpus Endometrial Carcinoma	4.156716	1.61E-05
chr16:g.2038416T>C	Esophageal carcinoma	3.813159	6.86E-05
chr17:g.31374233delC	Colon adenocarcinoma	2.893645	0.001904
chr20:g.49988448delA	Thymoma	5.392139	3.48E-08
chr19:g.48391405delG	Colon adenocarcinoma	4.497621	3.44E-06
chr17:g.76390876_76390877insG	Stomach adenocarcinoma	2.778963	0.002727
chr12:g.51246272delA	Ovarian serous cystadenocarcinoma	2.789973	0.002636
chr17:g.70178657_70178658insG	Colon adenocarcinoma	4.105688	2.02E-05
chr9:g.20622295_20622296insC	Uterine Corpus Endometrial Carcinoma	3.841062	6.13E-05
chr19:g.7857287delG	Cholangiocarcinoma	3.484173	0.000247
chr17:g.40089537delC	Colon adenocarcinoma	3.950175	3.90E-05
chr17:g.7581557delG	Colon adenocarcinoma	3.060388	0.001105
chr12:g.53380236_53380237insC	Colon adenocarcinoma	4.105688	2.02E-05
chr16:g.71853250delC	Colon adenocarcinoma	4.105688	2.02E-05
chr11:g.32439480delG	Colon adenocarcinoma	4.105688	2.02E-05
chr17:g.82101670delG	Stomach adenocarcinoma	4.016426	2.95E-05
chr5:g.138465720delC	Adrenocortical carcinoma	2.959737	0.00154
chr19:g.49860382delG	Uterine Corpus Endometrial Carcinoma	3.841062	6.13E-05
chr19:g.11453791A>C	Esophageal carcinoma	2.334594	0.009782
chr17:g.39658540delG	Colon adenocarcinoma	3.060388	0.001105
chr2:g.130181854delC	Colon adenocarcinoma	3.950175	3.90E-05
chr5:g.150053836delG	Uterine Corpus Endometrial Carcinoma	3.627101	0.000143
chr19:g.45784220delG	Stomach adenocarcinoma	2.951575	0.001581
chr19:g.41095983G>A	Skin Cutaneous Melanoma	3.956451	3.80E-05
chr17:g.76081507delC	Stomach adenocarcinoma	2.951575	0.001581
chr1:g.19338964delC	Thymoma	2.698934	0.003478
chr2:g.121337954C>A	Uterine Corpus Endometrial Carcinoma	3.841062	6.13E-05
chr17:g.28950997delC	Stomach adenocarcinoma	3.638229	0.000137
chr17:g.7260210_7260211insC	Rectum adenocarcinoma	2.768106	0.002819
chr6:g.139372513delG	Cholangiocarcinoma	3.622573	0.000146
chr6:g.33316976delG	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262

varyasyon	İlişkilendirilmiş hastalık	z-skoru	p-değeri
chr7:g.76981039delC	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr16:g.4258045delG	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr21:g.45941731G>A	Colon adenocarcinoma	3.199506	0.000688
chr12:g.54396134delC	Esophageal carcinoma	2.509482	0.006045
chr1:g.2184898delC	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr8:g.139618246C>T	Skin Cutaneous Melanoma	3.58025	0.000172
chrX:g.53233421delG	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr7:g.100464092T>G	Cervical squamous cell carcinoma and endocervical adenocarcinoma	3.535586	0.000203
chr19:g.10086909delC	Colon adenocarcinoma	3.199506	0.000688
chr1:g.19665631_19665632insG	Stomach adenocarcinoma	3.100133	0.000967
chr12:g.4911301delC	Colon adenocarcinoma	3.199506	0.000688
chr14:g.63045228delG	Stomach adenocarcinoma	3.100133	0.000967
chr19:g.1115966delG	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr7:g.130368696G>A	Uterine Corpus Endometrial Carcinoma	2.904788	0.001838
chrX:g.150768974delG	Stomach adenocarcinoma	3.100133	0.000967
chr17:g.9917204delC	Colon adenocarcinoma	3.199506	0.000688
chr18:g.14543388delC	Colon adenocarcinoma	3.199506	0.000688
chr10:g.100330116delG	Uterine Corpus Endometrial Carcinoma	2.904788	0.001838
chr13:g.30924495delG	Stomach adenocarcinoma	3.100133	0.000967
chr12:g.53380237delC	Colon adenocarcinoma	3.199506	0.000688
chr19:g.39539248delT	Rectum adenocarcinoma	2.768106	0.002819
chr7:g.76626824C>T	Esophageal carcinoma	4.417947	4.98E-06
chr17:g.31374738delG	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr1:g.26467454delG	Colon adenocarcinoma	3.724508	9.78E-05
chr17:g.39210647delGGG	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr4:g.26320982delG	Colon adenocarcinoma	3.199506	0.000688
chr14:g.69380515delC	Colon adenocarcinoma	3.199506	0.000688
chr2:g.232879425C>G	Stomach adenocarcinoma	3.100133	0.000967
chr17:g.76390877delG	Uterine Corpus Endometrial Carcinoma	2.904788	0.001838
chr17:g.42564370delG	Colon adenocarcinoma	3.199506	0.000688
chr11:g.57235733C>T	Uterine Corpus Endometrial Carcinoma	2.904788	0.001838
chr4:g.80197574delC	Cholangiocarcinoma	3.622573	0.000146
chr1:g.26467453_26467454insG	Colon adenocarcinoma	3.199506	0.000688
chr1:g.27306630_27306631insC	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262
chr6:g.45422504A>C	Mesothelioma	3.214366	0.000654
chr11:g.8682709G>A	Skin Cutaneous Melanoma	3.58025	0.000172
chr8:g.8238357C>T	Uterine Carcinosarcoma	3.657341	0.000127
chr1:g.35565746delC	Esophageal carcinoma	2.697039	0.003498
chr11:g.120240309delG	Colon adenocarcinoma	3.179065	0.000739
chr5:g.115177626G>A	Cholangiocarcinoma	3.747161	8.94E-05
chr6:g.36839389delC	Esophageal carcinoma	2.697039	0.003498

varyasyon	İlişkilendirilmiş hastalık	z-skoru	p-değeri
chr5:g.155013811G>T	Adrenocortical carcinoma	3.267873	0.000542
chr1:g.244853186delG	Esophageal carcinoma	2.697039	0.003498
chr19:g.43719965delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr11:g.124748369delC	Stomach adenocarcinoma	3.097839	0.000975
chr19:g.10468482delC	Esophageal carcinoma	2.697039	0.003498
chr16:g.1221731delG	Stomach adenocarcinoma	3.097839	0.000975
chr1:g.58781646A>C	Esophageal carcinoma	2.697039	0.003498
chr17:g.7244148delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr10:g.43373702G>A	Rectum adenocarcinoma	2.940924	0.001636
chr7:g.127593805delG	Colon adenocarcinoma	3.179065	0.000739
chr10:g.31319172delG	Colon adenocarcinoma	3.179065	0.000739
chr19:g.55674124delC	Stomach adenocarcinoma	3.097839	0.000975
chr22:g.26433575delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr16:g.28836524delG	Colon adenocarcinoma	3.179065	0.000739
chr6:g.122997583delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr1:g.171591877delG	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr17:g.39210647delGG	Cholangiocarcinoma	3.747161	8.94E-05
chr17:g.43014325delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr11:g.66287548delC	Esophageal carcinoma	2.697039	0.003498
chr4:g.150584473delCC	Esophageal carcinoma	2.697039	0.003498
chr16:g.81061072delG	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr19:g.5456798delG	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr16:g.71852754C>T	Adrenocortical carcinoma	3.267873	0.000542
chr16:g.67828183_67828184insG	Acute Myeloid Leukemia	2.899826	0.001867
chr8:g.26291736_26291737insC	Colon adenocarcinoma	3.179065	0.000739
chr4:g.144646631delC	Stomach adenocarcinoma	3.097839	0.000975
chr6:g.32128212delC	Esophageal carcinoma	2.697039	0.003498
chr1:g.147625087T>G	Rectum adenocarcinoma	2.940924	0.001636
chr7:g.100890416delG	Esophageal carcinoma	2.697039	0.003498
chr3:g.108551081_108551082insC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr15:g.38253087delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr2:g.240450683delG	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr2:g.130764769delG	Colon adenocarcinoma	3.179065	0.000739
chr7:g.44979054G>A	Mesothelioma	3.361868	0.000387
chrX:g.49002243delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr12:g.123409111delC	Sarcoma	2.48644	0.006451
chr19:g.40215421_40215422insG	Sarcoma	2.48644	0.006451
chr2:g.68042836G>A	Breast invasive carcinoma	2.391528	0.008389
chr19:g.8697429G>A	Rectum adenocarcinoma	2.940924	0.001636
chr15:g.72346223G>A	Rectum adenocarcinoma	2.940924	0.001636
chr15:g.90875734delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr12:g.123415629delC	Colon adenocarcinoma	3.179065	0.000739

varyasyon	İlişkilendirilmiş hastalık	z-skoru	p-değeri
chr16:g.1324898_1324899insG	Esophageal carcinoma	2.697039	0.003498
chr14:g.104803072delGG	Colon adenocarcinoma	3.179065	0.000739
chr19:g.4302399delC	Esophageal carcinoma	2.697039	0.003498
chr14:g.104803078G>T	Thymoma	3.029689	0.001224
chr1:g.156669283G>A	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chrX:g.153669898delG	Colon adenocarcinoma	3.179065	0.000739
chr9:g.32635666delG	Colon adenocarcinoma	3.179065	0.000739
chr7:g.135211535G>A	Skin Cutaneous Melanoma	3.043251	0.00117
chr19:g.4501850G>A	Colon adenocarcinoma	3.179065	0.000739
chr7:g.100485253delC	Colon adenocarcinoma	3.179065	0.000739
chr5:g.140561802G>A	Skin Cutaneous Melanoma	3.043251	0.00117
chr8:g.38969800delGGGA	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr9:g.35749255G>A	Skin Cutaneous Melanoma	3.043251	0.00117
chr7:g.898850delC	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chrX:g.14730588A>G	Uterine Carcinosarcoma	3.657341	0.000127
chr1:g.244056657_244056658insC	Colon adenocarcinoma	3.179065	0.000739
chr12:g.4353864delG	Thymoma	3.029689	0.001224
chr17:g.50135272A>T	Rectum adenocarcinoma	2.940924	0.001636
chr12:g.53712157G>A	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chr19:g.44759622_44759623insG	Stomach adenocarcinoma	3.097839	0.000975
chr1:g.151291739delG	Thymoma	3.029689	0.001224
chr19:g.38208007A>C	Uterine Corpus Endometrial Carcinoma	2.938198	0.001651
chrX:g.64189799_64189800insC	Ovarian serous cystadenocarcinoma	3.105637	0.000949
chr2:g.218985104delG	Colon adenocarcinoma	3.179065	0.000739
chr22:g.37372755delC	Thymoma	3.029689	0.001224

4.7 **Varyasyonların seçimi ve Dairesel Dikroizm ile varyasyon etkisi tespiti**

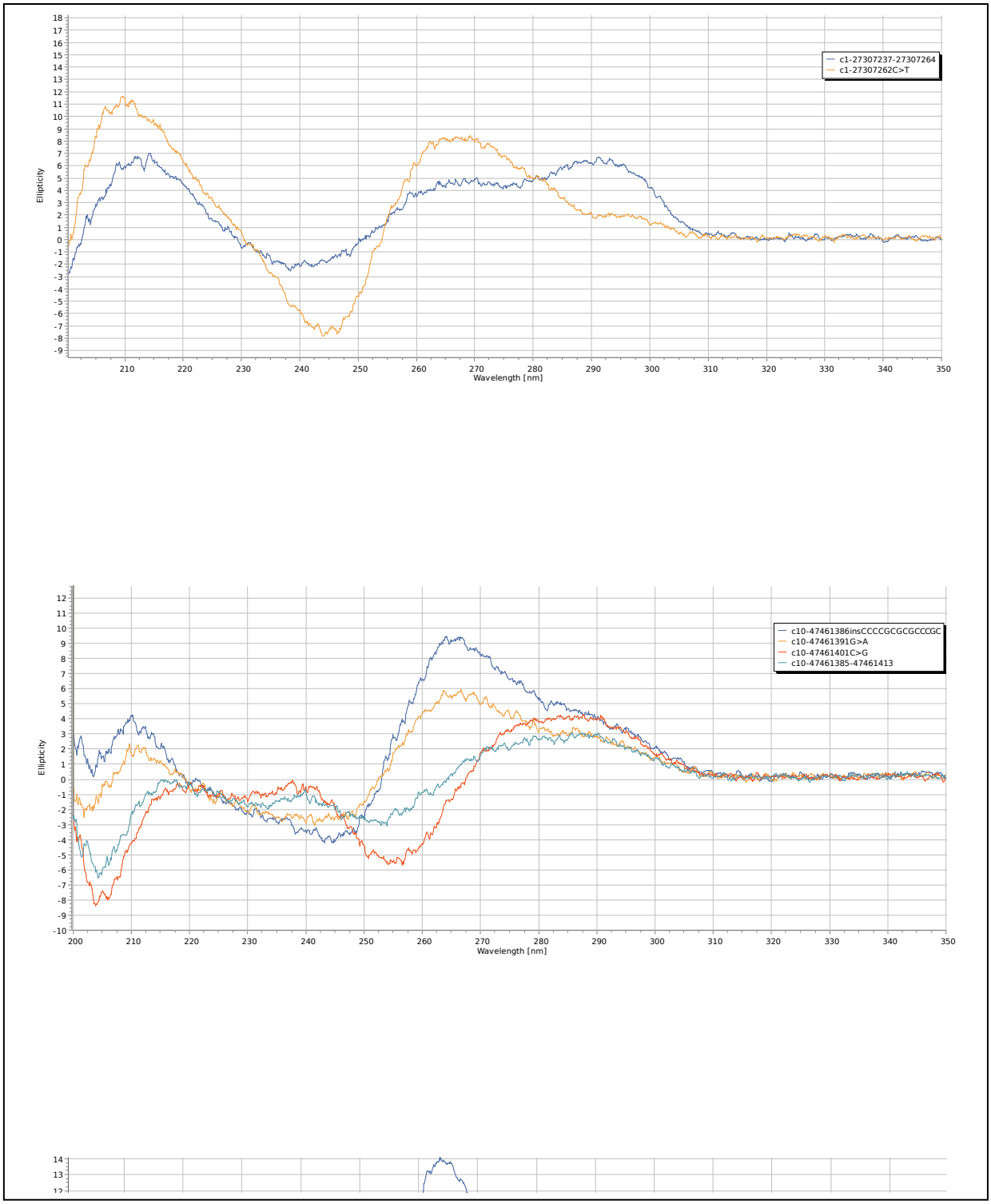
ClinVar ve TCGA veritabanındaki varyantlar ile ilişkilendirilmiş olan G-dörtlülerin varyasyona bağlı olarak yapısal değişimi var mıdır sorusunu cevaplamak için 40 baz altı uzunluğunda referans G-dörtlüsü oluşturan diziler ve onların varyasyonlarını içeren diziler temin edilmiştir. Bu dizilerle G-dörtlüsü oluşturularak dairesel dikroizm ile ölçüm yapılmıştır. Dairesel dikroizm ile elde edilen spektrumlar ölçülen G-dörtlüsüne has olup, yapısal değişimler spektrumda değişimler olarak yansır. Bu değişimleri sayısal olarak ifade etmek için herhangi bir varyasyon ile referans spektrum birbirinden çıkarılarak elde edilen spektrumun eğri altı alanı referans spektrumun eğri altı alanına bölünür. Varyasyonlar, referanslar ve değişim oranları ‘da listelenmiştir. Yapısal değişimin en yüksek olduğu üç G-dörtlüsü ve altı varyasyonun etkisi ’te gösterilmiştir.

Tablo . G-dörtlüsü varyantları ve referans genotiplerinden spektrum değişimi. Spektrum değişim değerinin 0 olması referans dizi anlamına gelmektedir. Tek nükleotid değişimlerinin altı çizilidir.

Varyant yada referans genotip	Sekans (5'->3')	Spektrum değişimi
rs145483167_wt	AGGGGTGGTGTGCGGAGTAGGGGTGGGTGGGGGGA	0
rs145483167A	AAGGGTGGTGTGCGGAGTAGGGGTGGGTGGGGGGA	0.312
rs151344623_wt	TGGGATCAGCGATGGTGCTGGGGGCCGGGCTGGGC	0
rs151344623G	TGGGATCAGCGATGGTGCTGGGGGCGGGGCTGGGC	0.186
rs199422241_wt	CGGGGCGGGAGGAGGCGGGGCCATCCCGGGGGC	0
rs199422241G	CGGGGCGGCAGGAGGCGGGGCCATCCCGGGGGC	0.307
rs199422256_wt	CGGGCGGGCTGGTTGGGGGGAACGGGA	0
rs199422256C	CGCGCGGGCTGGTTGGGGGGAACGGGA	0.587
rs199422257G	CGCGTTGCGGAGGGTGGGCCTGGGAGGGGTGGTGGC	0.242
rs199422259_wt	CGGGTTGCGGAGGGTGGGCCTGGGAGGGGTGGTGGC	0
rs199422259delCCACCAC	CGGGTTGCGGAGGGTGGGCCTGGGAGGGC	0.122
rs267606869_wt	CGGGGCGGCGGGGGCGCTCTAGGGC	0
rs267606869C	CGGGGCGGCGGGGGCGCTCCAGGGC	0.315
rs397515893_wt	TGGGTGGGGGCCGCAGGGAAGTGGC	0
rs397515893T	TGGGTGGGGGCTGCAGGGAAGTGGC	0.34
rs587779182_wt	CGGGGGACGTGGGAGGGGAGGCGGGA	0
rs587779182delTG	CGGGGGACGGGAGGGGAGGCGGGA	0.161
rs6467_wt	AGGAGGTGGGGGCTGGAGGGTGGGA	0
rs6467A	AGGAGTTGGGGGCTGGAGGGTGGGA	0.319
rs6467G	AGGAGCTGGGGGCTGGAGGGTGGGA	0.3
rs724159980_wt	CGGGGGGGCATCGCCGCGGGCCCGGGA	0
rs724159980delCCinsA	CGGGGGTCATCGCCGCGGGCCCGGGA	0.53
rs748749585_wt	AGGGCCTGGGGTGGGGGGGT	0
rs748749585A	AGGGCCTGGGGTAGGGGGGT	0.403
rs748749585C	AGGGCCTGGGGTCGGGGGGT	0.367
rs748749585T	AGGGCCTGGGGTTGGGGGGT	0.223
rs77888940_wt	AGGGCGGGGGGGGGGGGGGC	0
rs77888940G	AGGGCGGGGGGGGGGGGCGC	0.076
rs786204005_wt	AGGGTGGGGCAGGGAGCATCAGGGGGC	0
rs786204005delAG	AGGGTGGGGCAGGGAGCATCGGGGC	0.244
rs794726664_wt	CGGGTGGAAGGGGTCAGGGCAGCTGGGGTCTGGGGC	0
rs794726664G	CGGGTGGAAGGGGTCGGGGCAGCTGGGGTCTGGGGC	0.78
rs863223308_wt	TGGGGGTGAGGGTGGCGCCAGGGGT	0
rs863223308G	TGGGGGTGACGGTGGCGCCAGGGGT	0.449
rs864622197_wt	TGGAGGGGCCGGGGGAGAGGGAGAGAGAGGGC	0
rs864622197T	TGGAGGGGCTGGGGGAGAGGGAGAGAGAGGGC	0.251
rs879254284_wt	AGGTGCGGGGCCGCGGAGGGCAGGGT	0
rs879254284A	AGGTGCAGGGCCGCGGAGGGCAGGGT	0.394
rs879254284C	AGGTGCCGGGCCGCGGAGGGCAGGGT	0.566
rs879254645_wt	TGCGGTATGGGCGGGGCCAGGGTGGGGGCGGGGCGT	0

Varyant yada referans genotip	Sekans (5'->3')	Spektrum değişimi
rs879254645del22	TGGGGGCGGGGCGT	0.543
rs879254647dupT	TGCGGTATTGGGCGGGGCCAGGGTGGGGGCGGGGCGT	0.057
rs879254821_wt	TGGGTGAGCACGGGAAGGCGGCGGGTGGGGGC	0
rs879254821A	TGGGTGAACACGGGAAGGCGGCGGGTGGGGGC	0.141
rs879254821C	TGGGTGACCACGGGAAGGCGGCGGGTGGGGGC	0.162
c1+154963385-154963417	TGGGGGGGGCGGGTGTTGAGAGGGGTCTGGGA	0
c1+154963386insG	TGGGGGGGGGCGGGTGTTGAGAGGGGTCTGGGA	0.494
c1+154963387delG	TGGGGGGGCGGGTGTTGAGAGGGGTCTGGGA	0.391
c1+154963387delGG	TGGGGGGCGGGTGTTGAGAGGGGTCTGGGA	0.345
c10+47461192-47461219	AGGGACCCACATGGGCCAGGGCGGTGC	0
c10+47461194G>T	ATTGGACCCACATGGGCCAGGGCGGTGC	0.375
c10-47461385-47461413	CGGGCGCGCGGGGGCGGGCGCGCGGGGC	0
c10-47461386insCCCCGCGCGCCCGC	CGGGCGCGCGGGGGCGGGCGCGCGGGGGCGGGCGCGCGGGGC	1.586
c10-47461391G>A	CGGGCGCGCGGGGGCGGGCGCGTGGGGC	0.922
c10-47461401C>G	CGGGCGCGCGGGCGCGGGCGCGCGGGGC	0.573
c1-27306383-27306410	AGGGGCTGGGGACCAGGGCTGGAGGGT	0
c1-27306409C>G	ACGGGCTGGGGACCAGGGCTGGAGGGT	0.344
c1-27306409C>T	AAGGGCTGGGGACCAGGGCTGGAGGGT	0.499
c1-27306617-27306646	AGGAGATGGGGGGGGGCATATGGAGGGGT	0
c1-27306630insC	AGGAGATGGGGGGGGGGCATATGGAGGGGT	0.11
c1-27306631delC	AGGAGATGGGGGGGGCATATGGAGGGGT	0.228
c1-27306642C>A	AGGATATGGGGGGGGGCATATGGAGGGGT	0.224
c1-27307237-27307264	TGGGATGGGAAGGAGGGCAGAAAGGGC	0
c1-27307262C>T	TGAGATGGGAAGGAGGGCAGAAAGGGC	0.814
c1-27307505-27307533	AGGGGAAGCAGGGAGGGAGAGAGGGGGA	0
c1-27307507delC	AGGGGAAGCAGGGAGGGAGAGAGGGGA	0.093
c18-70326547-70326571	TGGGCGGGGGGGGCATTGAGGAGA	0
c18-70326558insC	TGGGCGGGGGGGGGCATTGAGGAGA	0.151
c18-70326558insCC	TGGGCGGGGGGGGGGCATTGAGGAGA	0.129
c18-70326559delC	TGGGCGGGGGGGCATTGAGGAGA	0.124
c18-70326567G>C	TGGGGGGGGGGGCATTGAGGAGA	0.217
c3+185644588-185644617	TGGGGGGGGGGGGGTGCGTAGATACGGGA	0
c3+185644589T>G	GGGGGGGGGGGGGGTGCGTAGATACGGGA	0.154
c3+185644590delGG	TGGGGGGGGGGGTGCGTAGATACGGGA	0.059
c3+185644590delGGG	TGGGGGGGGGGTGCGTAGATACGGGA	0.112
c3+185644590delGGGG	TGGGGGGGGGTGCGTAGATACGGGA	0.217
c3+185644595G>A	TGGGGGAGGGGGGGTGCGTAGATACGGGA	0.15
c3+185644613C>T	TGGGGGGGGGGGGGTGCGTAGATATGGGA	0.059
c3+185645294-185645322	TGGGGGGCGGGAGGCGGGGCTCGGTGGT	0
c3+185645296delG	TGGGGGCGGGAGGCGGGGCTCGGTGGT	0.135
c3+185645302C>T	TGGGGGGTGGGAGGCGGGGCTCGGTGGT	0.07

Varyant yada referans genotip	Sekans (5'->3')	Spektrum değişimi
c3-185645343-185645365	AGGGGCCGAGGAGGGGCGGGGA	0
c3-185645349G>A	AGGGGCCGAGGAGGGGTGGGGA	0.191
c3-185645358C>G	AGGGGCCCAGGAGGGGCGGGGA	0.264
c3-185645359G>A	AGGGGCTGAGGAGGGGCGGGGA	0.238
c9+69012834-69012862	TGGGGGGGGGTGAGGGAGCAGCTGGTGT	0
c9+69012836delG	TGGGGGGGGTGAGGGAGCAGCTGGTGT	0.122
c9+69012836delGG	TGGGGGGGTGAGGGAGCAGCTGGTGT	0.261
c9+69012839insCA	TGGGGCAGGGGGTGAGGGAGCAGCTGGTGT	0.42
c9-69012969-69012993	TGGTGGTTTGGGGGATGGGAGGGT	0
c9-69012980C>G	TGGTGGTTTGGGGCATGGGAGGGT	0.403
c9-69012982C>T	TGGTGGTTTGGAGGATGGGAGGGT	0.855



Şekil . Spektrum değişimi oranı 0.8 üzerinde olan varyantlar ve referansların dairesel dikroizm spektrumları

5. **TARTIŞMA/SONUÇ**

Çalışmalarımızın ilk adımında G-dörtlüsü tahmini için bilinen G-dörtlülerine karşı var olan algoritmalardan daha yüksek hassasiyet ve J-endeksi gösteren yeni bir algoritma yazılmıştır. Bu algoritma uluslararası BIONIC Biology of non-canonical nucleic acids: from humans to pathogens sempozyumunda sunulmuş ve SCI endekli Journal of Theoretical Biology dergisinde yayınlanmıştır. Bu algoritmanın python kodu olarak https://github.com/odoluca/G4Catchall ve ayrıca çevrimiçi araç olarak homes.ieu.edu.tr/odoluca/G4Catchall adresinde kullanıma sunulmuştur. Bu algoritmanın yardımı ile insan genomundaki G-dörtlüleri de listelenmiş ve homes.ieu.edu.tr/odoluca/Archive adresinde ulaşılabilmektedir. Bu yazılım ile diğer canlı türlerinin genomları için de benzer tarama yapılmasını mümkün kılmaktadır.

G-dörtlülerinin sınıflandırılması ile 5000'in üzerinde çok geniş G-dörtlüsü aileleri keşfedilmiş olup bu G-dörtlüsü ailelerinin taranmasını kolaylaştırmak adına motif dosyaları yukarıdaki adreste ulaşıma açılmıştır. G-dörtlüleri motiflerinin belirlenmesi ve sınıflandırılması ile bu G-dörtlülerinin ilişkilendirilmiş olduğu genlerin de benzer fonksiyona sahip yolaklarda bulunması olasıdır. Bu ilişkilendirmenin somut olarak ortaya konulması hangi G-dörtlülerinin regülasyonda rolleri olduğunu da ortaya koyacaktır. Bu bağlamda yaptığımız çalışmalar öncü verileri sağlayıp, hedef gösterilmesi açısından önem taşımaktadır.

Aynı zamanda bulduğumuz G-dörtlüleri patojenik varyantlarla ilişkilendirilmiştir. Protein kodlayan bölgelerdeki varyantların birincil etkisinin protein seviyesinde zaten görülebilecek olmasından dolayı tüm patojenik varyantlardan sadece promotor, UTR yada intron bölgelerinde olanlar özellikle ön planda incelenmiştir. Çünkü bu bölgelerde oluşan varyasyonların etkisinin G-dörtlüsünün oluşabildiği DNA yada RNA seviyesinde olması beklenir. Bu varyantların çakıştığı G-dörtlüleri tespit edilmiş ve bu varyasyonların G-dörtlüsünün yapısında bir etkisinin olup olmayacağı dairesel dikroizm spektrometre ile ölçümlenmiştir. Test edilen 58 varyantın birçoğunda yapısal değişim asgari kalmış iken, özellikle guanin tekrarlarının etkilenmesi beklenildiği gibi yapısal bozulmayı en çok tetikleyen mutasyon olmuştur. Her nekadar ilmik bölgelerindeki değişimler yapıyı az oranda etkilemiş olsa da buradaki değişimler, varsa, bağlanan proteinler ile olan etkileşimi değiştirebilir. Ancak bu etkileşimdeki değişimin gözlemlenmesi için öncelikle bağlanan proteinlerin keşfedilmesi gereklidir. Yapısal bozulmasına bağlı hastalıkların görüldüğü G-dörtlülerini bağlayan proteinler hücre ekstraktlarından çekilerek tespit edilirse, bu hastalıkların mekanizmasının anlaşılmasında önemli bir adım atılmış olur.

Test edilen 58 varyant içerisinde yüksek z-skoru gösterenler (z>3.0) ve onların neden olduğu spektrum değişimleri göz önüne alındığında spektrum değişim miktarının değişkenlik gösterdiği, buna rağmen G-dörtlülerinde oluşan mutasyonların hastalıklara yol açtığı yorumlanabilir. () Bu durum G-dörtlülerindeki ufak yapısal değişimlerin bile içerisinde oldukları etkileşimlerin değiştirdiğini göstermektedir. Özellikle DNA'nın G-dörtlüsü yapısına dönmesinin yeterli olmadığının, bu yapının etkileşim içerisinde olduğu proteinlerin bağlanma kararlılıklarının etkilendiği şeklinde yorumlanabilir. Bu durum da G-dörtlülerinin etkilerinin sadece yapıya bağlı değil, G-dörtlüsü ile proteinler arasındaki etkileşime bağlı olduğunu desteklemektedir.

Tablo . Z-skoru 3.0'ün üstünde olup test edilen varyantları ve spektrum değişimleri.

Varyasyon	Hastalık	z-score	p-değeri	Spektrum değişimi
chr1:g.154963387delG	Uterine Corpus Endometrial Carcinoma	10.15597	1.56E-24	0.391
chr10:g.47461391G>A	Cholangiocarcinoma	3.523893	0.000213	0.922
chr1:g.27306631delC	Uterine Corpus Endometrial Carcinoma	7.117152	5.51E-13	0.228
chr1:g.27306630_27306631insC	Uterine Corpus Endometrial Carcinoma	3.468687	0.000262	0.11
chr18:g.70326559delC	Colon adenocarcinoma	5.667184	7.26E-09	0.124
chr3:g.185644590delGGG	Uterine Corpus Endometrial Carcinoma	6.439626	5.99E-11	0.112
chr3:g.185644590delGGGG	Uterine Corpus Endometrial Carcinoma	4.908608	4.59E-07	0.217
chr9:g.69012836delG	Uterine Corpus Endometrial Carcinoma	8.853031	4.26E-19	0.122

Kısaca, bu proje kapsamında geliştirilen biyoenformatik yöntemler ve araçlar ile G-dörtlülerinin kıyaslanması için yeni bir yöntem silsilesi geliştirilmiş ve diğer organizmalarda benzer analizlerin yapılması için öncülük edilmiştir. Ayrıca hastalıklarla ilişkilendirilmiş G-dörtlülerinin hedeflenerek tedavi stratejilerinin geliştirilmesinin önü açılmıştır. Bu amaçla bir proje başvurusu daha gerçekleştirilmiş olup, yapısal değişimi tespit edilmiş ve regülasyondan sorumlu bölgede bulunan G-dörtlülerinin hedeflenmesi amaçlanmaktadır. Ek olarak, bu çalışmanın sonucunda motiflerin ortaya çıkarılması ile, belli G-dörtlülerini hedefleyen, yapıca seçici ilaçların hangi genleri yada genomun hangi noktalarını etkileyebileceğini tahmin etmek kolaylaştırılmıştır. Bu bağlamda G-dörtlülerine yönelik ilaçların etki alanının tahmin edilmesi mümkün olacaktır.

Ekler

Ek . Quadparser algoritma kodu

```
#!/usr/bin/python
import re
import sys
import string
import argparse
parser = argparse.ArgumentParser(description="""

MODIFICATIONS

Output will now include each G4 sequence and its sequence on the + strand.
These
sequences will include one additional base in each direction.

Included ReverseComplement() function.
ReverseComplement() is modified to reverse Ns. This is necessary for G4s
flanking poly-Ns.

DESCRIPTION

    Search for matches to a regex in a fasta file and return a bed file with
the
    coordinates of the match and the matched sequence itself.

    With defaults, quadparser.py searches for putative quadruplexes on
forward
    and reverse strand using the quadruplex rule described at
    http://en.wikipedia.org/wiki/G-
quadruplex#Quadruplex_prediction_techniques.

    The defualt regex is '([gG]{3}\w{1,7}){3,}[gG]{3}' and its complement
    produce the same output as in http://www.quadruplex.org/?
view=quadbaseDownload

    Output bed file has columns:
    1. Name of fasta sequence (e.g. chromosome)
    2. Start of the match
    3. End of the match
    4. ID of the match
    5. Length of the match
    6. Strand
    7. Matched sequence

    Note: Fasta sequences are read in memory one at a time. Also the bed
file
    of of the matches are kept in memeory.

EXAMPLE:
    ## Test data:
    echo '>mychr' > /tmp/mychr.fa
    echo 'ACTGnACTGnACTGnTGAC' >> /tmp/mychr.fa

    quadparser.py -f /tmp/mychr.fa -r 'ACTG'
```

```
        mychr   0       4       mychr_0_4_for   4       +       ACTG
        mychr   5       9       mychr_5_9_for   4       +       ACTG
        mychr   10      14      mychr_10_14_for 4       +       ACTG
        mychr   15      19      mychr_15_19_rev 4       -       TGAC


    ls /tmp/mychr.fa | quadparser.py -f - -r 'A\w\wGn'
        mychr   0       5       mychr_0_5_for   5       +       ACTGn
        mychr   5       10      mychr_5_10_for  5       +       ACTGn
        mychr   10      15      mychr_10_15_for 5       +       ACTGn

DOWNLOAD
    quadparser.py is hosted at http://code.google.com/p/bioinformatics-misc/

TODO

    - Better handling of forward and reverse matches (i.e. other than
complementing
      the forward regex?).
    - Read sequence from stdin ('less myseq.fa | quadparser.py -f - ...').
    """, formatter_class=argparse.RawTextHelpFormatter)

parser.add_argument('--regex', '-r',
                    type=str,
                    help='''Regex to be searched in the fasta input.
Matches to this regex will have + strand. This string passed to python
re.compile(). The default regex is '([gG]{3}\w{1,7}){3,}[gG]{3}' which
searches
for G-quadruplexes.

                    ''',
                    default='\w{1}([gG]{3,}\w{1,7}){3,}[gG]{3,}\w{1}')

parser.add_argument('--regexrev', '-R',
                    type=str,
                    help='''The second regex to be searched in fasta input.
Matches to this regex will have - strand.
By default (None), --regexrev will be --regex complemented by replacing
'actguACTGU' with 'tgacaTGACA'. NB: This means that [a-zA-Z] will be
translated
to [t-zT-Z] and proteins are not correctly translated.

                    ''',
                    default=None)

parser.add_argument('--fasta', '-f',
                    type=str,
                    help='''Input file in fasta format containing one or
more
sequences. Use '-' to get the name of the file from stdin

                    ''',
                    required=True)

parser.add_argument('--noreverse',
                    action='store_true',
                    help='''Do not search the reverse (-) strand. I.e. do
```

```
not use
the complemented regex (or --regexrev/-R). Use this flag to search protein
sequences.                    ''')

args = parser.parse_args()

" -------------------------[ Check and pare
arguments ]---------------------- "

""" Reverse forward match """
intab = 'actguACTGU'
outtab = 'tgacaTGACA'
if args.regexrev is None:
    transtab = string.maketrans(intab, outtab)
    regexrev = args.regex.translate(transtab)
else:
    regexrev = args.regex

if args.fasta == '-':
    args.fasta = sys.stdin.readlines()
    if len(args.fasta) > 1:
        sys.exit('\nquadpareser.py: Only one input file at a time can be
processed:\n--fasta/-f: %s\n' % (args.fasta))
    args.fasta = args.fasta[0].strip()

" ------------------------------
[  Functions ]-------------------------------- "

# modification: reverse complement function.
def ReverseComplement(seq):
    seq1 = 'ATCGNTAGCNatcgntagcn'
    seq_dict = {seq1[i]: seq1[i + 5] for i in range(20) if i < 5 or 10 <= i
< 15}
    return "".join([seq_dict[base] for base in reversed(seq)])



"""                                LIST SORTER
Code to sort list of lists
see http://www.saltycrane.com/blog/2007/12/how-to-sort-table-by-columns-in-
python/
"""
import operator


def sort_table(table, cols):
    """ sort a table by multiple columns
        table: a list of lists (or tuple of tuples) where each inner list
               represents a row
        cols:  a list (or tuple) specifying the column numbers to sort by
               e.g. (1,0) would sort by column 1, then by column 0
    """
    for col in reversed(cols):
        table = sorted(table, key=operator.itemgetter(col))
    return (table)
"""                            END of SORTER
-------------------------------------------------------------------------
```

```
-
"""

psq_re_f = re.compile(args.regex)
psq_re_r = re.compile(regexrev)

ref_seq_fh = open(args.fasta)

ref_seq = []
line = (ref_seq_fh.readline()).strip()
chr = re.sub('^>', '', line)
line = (ref_seq_fh.readline()).strip()
gquad_list = []
while True:
    while line.startswith('>') is False:
        ref_seq.append(line)
        line = (ref_seq_fh.readline()).strip()
        if line == '':
            break
    ref_seq = ''.join(ref_seq)
    for m in psq_re_f.finditer(ref_seq):
        #quad_id = str(chr) + '_' + str(m.start()) + '_' + str(m.end()) +
'_for'
        gquad_list.append([chr, m.start(), m.end(), len(m.group(0)), '+',
m.group(0),
                           m.group(0)])  # modification: added sequence
again
    if args.noreverse is False:
        for m in psq_re_r.finditer(ref_seq):
            #quad_id = str(chr) + '_' + str(m.start()) + '_' + str(m.end())
+ '_rev'
            gquad_list.append([chr, m.start(), m.end(), len(m.group(0)),
'-', m.group(0),
                               ReverseComplement(m.group(0))])  #
modification: added reverse complement
    chr = re.sub('^>', '', line)
    ref_seq = []
    line = (ref_seq_fh.readline()).strip()
    if line == '':
        break

gquad_sorted = sort_table(gquad_list, (0, 1, 2, 3))

for line in gquad_sorted:
    line = '\t'.join([str(x) for x in line])
    print (line)
sys.exit()
```

Ek . G4Hunter algoritma kodu

```
#!/usr/bin/python
################################################################################
```

```
"""
    <G4Hunter - a program to search quadruplex-forming regions in DNA.>
    Copyright (C) <2012>  <Bedrat amina  supervised by Dr.Jean Louis
Mergny>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""
###############################################################################
import os, re, sys, getopt
import time
import shutil
import numpy as np
from matplotlib import pyplot
import matplotlib.pyplot as plt
from Bio import SeqIO


def main(argv):

   if not argv:
        sys.stdout.write("Sorry: you must specify at least an argument\n")
        sys.stdout.write("More help avalaible with -h or --help option\n")
        sys.exit(1)

   inputfile = ''
   outputfile = ''
   try:
      opts, args = getopt.getopt(argv,"hf:o:w:s:",
["help","ffile=","ofile="])
   except getopt.GetoptError:
      print '\033[1m' +'python G4Hunter.py -f <inputfile> -o
<outputrepository> -w <window> -s <score threshold>\n'+'\033[0;0m'
      sys.exit(1)

   for opt, arg in opts:
      if opt in ('-h',"--help"):
          print  '\033[1m' +'\t ----------------------'+'\033[0;0m'
          print  '\033[1m' +'\n\t  Welcome To G4Hunter :'+'\033[0;0m'
          print  '\033[1m' +'\t ----------------------'+'\033[0;0m'

          print 'G4Hunter takes into account G-richness and G-skewness of a
given sequence and gives a quadruplex propensity score as output.'
          print 'To run G4Hunter use the commande line: \n'
          print  '\033[1m' +'python G4Hunter.py -f <inputfile> -o
<outputrepository> -w <window> -s <score threshold>\n'+'\033[0;0m'
          sys.exit()
```

```
      elif opt in ("-f", "--ffile"):
         inputfile= arg
      elif opt in ("-o", "--ofile"):
         outputfile = arg
      elif opt in ("-w", "--wind"):
         window = arg

      elif opt in ("-s", "--score"):
         score = arg

   return  inputfile, outputfile, int(window), float(score)

#calcule le score de chaque base dans un fichier qui peut contenir plusieur
sequences fasta
#le fichier doit comporter la nom de la seq + la sequence en une seul ligne
donc pas de \n ou \r
class Soft(object):

    def __init__(self):
        pass

    def ReadFile(self,Filein):
        ListSeq,LHeader=[],[]
        for record in SeqIO.parse(Filein, "fasta") :
            LHeader.append(record.description)
            ListSeq.append(record.seq)
        return LHeader,ListSeq

    def ReadSeq(self, Seqs):
        ListSeq,LHeader=[],[]
        for record in SeqIO.parse(Seqs, "fasta") :
            LHeader.append(record.description)
            ListSeq.append(record.seq)
        return LHeader,ListSeq

    def GFinder(self,Filein,k):
        LHeader,ListSeq=self.ReadFile(Filein)
        LSeq,LNumber,LScoreSeq,SeqLine=[],[],[],""
        for i in range(len(ListSeq)) :
            Sequence,liste=self.BaseScore(ListSeq[i])
            LSeq.append(Sequence)
            LScoreSeq.append(self.CalScore(liste, k))
            LNumber.append(liste)
        return LScoreSeq, LSeq , LNumber, LHeader

    def BaseScore(self,line):
        item, liste=0, []
        #calcule le item de chaque base et la stock dans liste
        while ( item < len(line)):
            #a la fin d une sequence il est possible d avoir des GGG dans
se cas
            # on verifie si la secore+1<len(line) car il ya un deuxieme G
            #et
            if (item < len(line) and (line[item]=="G" or line[item]=="g")):
                liste.append(1)
                #print liste
                if(item+1< len(line) and (line[item+1]=="G" or
```

```
line[item+1]=="g")):
                    liste[item]=2
                    liste.append(2)
                    if (item+2< len(line) and (line[item+2]=="G" or
line[item+2]=="g")):
                        liste[item+1]=3
                        liste[item]=3
                        liste.append(3)
                        if (item+3< len(line) and (line[item+3]=="G" or
line[item+3]=="g")):
                            liste[item]=4
                            liste[item+1]=4
                            liste[item+2]=4
                            liste.append(4)
                            item=item+1
                        item=item+1
                    item=item+1
                item=item+1
                while(item < len(line) and (line[item]=="G" or
line[item]=="g")):
                        liste.append(4)
                        item=item+1

            elif (item < len(line) and (line[item]=="T" or line[item]=="A"
or line[item]=="t" or line[item]=="a" or line[item]=="U"or line[item]=="u"
or
                                        line[item]=="-" or line[item]=="N"
or line[item]=="_" or line[item]=="Y"
                                        or line[item]=="W" or
line[item]=="R" or
                                       line[item]=="K" or line[item]=="M"or
line[item]=="S" or line[item]=="B"
                                       or line[item]=="V"or
line[item]=="D"or line[item]=="H"or line[item]=="N")):
                liste.append(0)
                item=item+1

            elif(item < len(line) and (line[item]=="C" or line[item]=="c")):
                liste.append(-1)
                if(item+1< len(line) and (line[item+1]=="C" or
line[item+1]=="c" )):
                    liste[item]=-2
                    liste.append(-2)
                    if (item+2< len(line) and (line[item+2]=="C" or
line[item+2]=="c" )):
                        liste[item+1]=-3
                        liste[item]=-3
                        liste.append(-3)
                        if (item+3< len(line) and (line[item+3]=="C" or
line[item+3]=="c"  )):
                            liste[item]=-4
                            liste[item+1]=-4
                            liste[item+2]=-4
                            liste.append(-4)
                            item=item+1
                        item=item+1
                    item=item+1
```

```
                item=item+1
                while(item < len(line) and (line[item]=="C" or
line[item]=="c")):
                    liste.append(-4)
                    item=item+1

            else:
                    item=item+1 #la fin du la ligne ou y a des entrers
        return line, liste

    def CalScore(self,liste, k):
        Score_Liste=[]
        #calcule de la moynne des scores pour toutes les sequences - les
derniers k bases
        for i in range (len (liste)-k):
            j,Sum=0,0
            while (j<k):
                Sum=Sum+liste[i]
                j=j+1
                i=i+1
            Mean=Sum/float(k)
            Score_Liste.append(Mean)
        return Score_Liste
    ########################################################

    def plot2(self,liste, repert):
        # make a little extra space between the subplots
        plt.subplots_adjust(wspace=1.0)
        dt = 1
        t = np.arange(0, len(liste), dt)
        figure= plt.figure()
        plt.plot(t, liste, 'b-')
        plt.xlim(0,len(liste))
        plt.xlabel('Position (ntS)')
        plt.ylabel('Score')
        plt.grid(True)
        figure.savefig(repert+'Score_plot.pdf', dpi=figure.dpi)

    # modification: reverse complement function.
    def ReverseComplement(self,seq):
        seq1 =
'ATCGNWMKRYSVBHDatcgnwmkrysvbhdTAGCNWKMYRSBVDHtagcnwkmyrsbvdh'
        seq_dict = {seq1[i]: seq1[i + 30] for i in range(30)}
        return "".join([seq_dict[base] for base in reversed(seq)])

    ########################################################
    def GetG4(self,line,liste, Window,k, header, Len ):
        LG4=[]
        SEQ=">"+header+"\n Start \t End \t Sequence\t Length \t Score\n"
        for i in range(len(liste)) :
            if (liste[i]>= float(Window) or liste[i]<= - float(Window)):
                seq=line[i:i+k]
                LG4.append(i)
        return LG4

    def WriteSeq(self,line,liste, LISTE,header, F, Len ):
        i,k,I=0,0,0
```

```
        a=b=LISTE[i]
        MSCORE=[]
        if (len(LISTE)>1):
            c=LISTE[i+1]
            while (i< len(LISTE)-2):
                if(c==b+1):
                    k=k+1
                    i=i+1
                else:
                    I=I+1
                    seq=line[a:a+F+k]
                    sequence,liste2=self.BaseScore(seq)
                    self.Write( a, k ,F,0, seq ,len(seq) ,
round(np.mean(liste2),2),header,"")
                    MSCORE.append(abs(round(np.mean(liste2),2)))
                    k=0
                    i=i+1
                    a=LISTE[i]
                b=LISTE[i]
                c=LISTE[i+1]
            I=I+1
            seq=line[a:a+F+k+1]
            sequence,liste2=self.BaseScore(seq)
            self.Write( a, k ,F,1, seq ,len(seq) ,
round(np.mean(liste2),2),header,"")
            MSCORE.append(abs(round(np.mean(liste2),2)))
        return MSCORE

    def Write(self, i, k ,F,X, seq ,long, score, source, strand):
        if score<0:
            strand="-"
            G4sequence=self.ReverseComplement(seq)
        else:
            strand="+"
            G4sequence = seq
       LINE=str(source)+"\t"+str(i)+"\t"+str(i+k+F+X)+"\t"+str(long)
+"\t"+str(strand)+"\t"+str(seq) +"\t"+str(G4sequence)+"\t" #+str(score)
        print(LINE)
    #Len dans le cas ou la sequence fini avec des ----- donc il yaura une
erreur
if __name__ == "__main__":
    inputfile, outputfile , window, score = main(sys.argv[1:])
    #=================================================================
    fname=inputfile.split("/")[-1]
    filefasta=fname.split(".")
    filein=open(inputfile,"r")
    #=========================================

    startTime = time.time()
    soft1=Soft()
    ScoreListe, DNASeq, NumListe, HeaderListe=soft1.GFinder(filein, window)
    for i in range(len(DNASeq)):
        G4Seq=soft1.GetG4(DNASeq[i], ScoreListe[i], float(score),
int(window), HeaderListe[i], len(NumListe[i]))
        if (len(G4Seq)>0):
            MSCORE=soft1.WriteSeq(DNASeq[i],ScoreListe[i], G4Seq,
HeaderListe[i], int(window), len(NumListe[i]))
```

```
    filein.close()
    fin=time.time()
```

Ek . Çok izgeli tarama betiği

```
# -*- coding: utf-8 -*-  #fixes turkish character problems
import sys, subprocess, os
import multiprocessing #use other threads
from multiprocessing.dummy import Pool as ThreadPool #best way to
multiprocess
import time
from common import DateCode
from string import punctuation
pool=ThreadPool(multiprocessing.cpu_count())#use all cores available
#variables
sourceDir=os.curdir
FastaFiles=[] #Change to FastaFiles=[] if you
           # want the file to find all fasta files in the current directory
#constants
corecount=multiprocessing.cpu_count()
command= 'python G4Catchall.py -I 1 -B --G3L 1..3 --XL 1..9 '
# command = 'python quadparser.py ' #shell command for quadparser command.
Can add new regex pattern: -r "\w{1}([gG]{3}\w{1,7}){3,}[gG]{3}\w{1}"

#functions
def Search(file):
    global command
    return subprocess.check_output(command + ' -f "' + file+'"', shell=True)
#Note: strength of the preciding and following base pairing is quite
important to allow double strand to be unwound. So it would only be proper
to include the preceding and following nucleotide in the G-quadruplex
detection.
if __name__=="__main__":
    #listing fasta files
    print ("Finding fasta files...\n
_____________________________________________________________________")
    if FastaFiles==[]:
        Files=os.listdir(sourceDir)
        for File in Files:
            if File.endswith(".fa"):
                print ("> "+File)
                FastaFiles.append(File)
        if len(FastaFiles)<1:
            sys.exit("no file to work with. Place "".fa"" files in the
working directory. quiting.")
    else:
         for file in FastaFiles:
             print(file)
    QA1="N"
    if len(FastaFiles)>1:
        QA1= input("
_____________________________________________________________________\nDo you
want to append results in a single file? (Y/N)")
    starttime=time.time() #to calculate time elapsed
```

```
    #puts all results in separate files
    if str(QA1).capitalize() == "N":
        print ("Results will be separate.\nScanning for G-quadruplexes...")
        results=pool.map(Search,FastaFiles)#multithreading magic
        pool.close()
        pool.join()

        for idx,file in enumerate(FastaFiles): #writes to files
            resultFile = open(file.replace(".fa","").replace("
","").strip(punctuation) +  ".D"+ DateCode() + ".G4List", "w") #overwrites
existing data
            resultFile.write("#" + command + " \n")
            resultFile.write(results[idx].decode())
            resultFile.close()
            print (resultFile.name + " is created.")
    #puts all results in one file
    elif str(QA1).capitalize() == "Y":
        from difflib import SequenceMatcher
        from string import punctuation
        def findCommonFilename(strList):
            commonName = strList[0]
            for name in strList[1:]:
                nameMatch = SequenceMatcher(None, commonName, name)
                match = nameMatch.find_longest_match(0, len(commonName), 0,
len(name))
                if match.size < 5:
                    commonName = "DiscoveredG4s"
                    return commonName
                commonName = commonName[match.a:match.a + match.size]
            return commonName.replace(".fa","").replace("
","").strip(punctuation)
        OutputName = findCommonFilename(FastaFiles)
        print ("Results will be in one file.\nScanning for G-
quadruplexes...")
        results=pool.map(Search,FastaFiles)#multithreading magic
        pool.close()
        pool.join()
        OutputName=OutputName+ ".D"+DateCode()+".G4List"
        resultFile = open(OutputName , "w")#creates a new and clean file
        resultFile.write("#"+command+" \n")
        resultFile.close()

        for idx,file in enumerate(FastaFiles): #writes to a file
            resultFile=open(OutputName,"a") #this is why the results are
appended
            resultFile.write(results[idx].decode()) #rmeove last character
which is probably /n
            resultFile.close()
        print (resultFile.name + " is created.")
    else:
        sys.exit("use Y or N only. quiting.")
    print ("completed in %s seconds" % (time.time()-starttime) )
```

Ek . G4Catchall algoritma kodu

```
from __future__ import division,print_function,absolute_import
import argparse, sys, re, string, regex
" -------------------------[ Parse arguments ]---------------------- "
parser=argparse.ArgumentParser(
    description="""
    DESCRIPTION

        Searches for matches to a G-quadruplex-fitting regex in a fasta
file,
        filters through G4Hunter-like secondary scoring scheme and return a
bed file with
        coordinates of the match, matched sequence, G-quadruplex forming
sequence and the score.

        Output bed file has the following columns:
        1. description of the fasta sequence (e.g. NC_00024.11 Y chromosome)
        2. start of the match
        3. end of the match
        4. size of the match
        5. strand of the match (e.g. +)
        6. positive strand sequence of the match (e.g. CCCTTCCCTTTCCCTCCC)
        7. matched G-quadruplex-forming sequence (e.g. GGGAGGGAAAGGGAAGGG)
        8. score of the matched G-quadruplex-forming sequence based on
selected scoring scheme

    EXAMPLE
        ##Test data:
        echo '>mychr' > /tmp/mychr.fa
        echo 'TTGGGTTGGGACTGGGTACGGGAATAAATAGGTTAGGAATGGATAGGAT' >>
/tmp/mychr.fa

        G4Catchall.py -f /tmp/mychr.fa --G3L 1..3 --G4H
            mychr   2   22  20  +   GGGTTGGGACTGGGTACGGG
GGGTTGGGACTGGGTACGGG    2.11

        G4Catchall.py -f /tmp/mychr.fa --G3L 1..3 --G2L 1..3
            mychr   2   22  20  +   GGGTTGGGACTGGGTACGGG
GGGTTGGGACTGGGTACGGG
            mychr   30  47  17  +   GGTTAGGAATGGATAGG   GGTTAGGAATGGATAGG

        G4Catchall.py -I 0
            ([Gg]{3,})  (\w{1,8})  ([Gg]{3,}) (\w{1,8}) ([Gg]{3,}) (\w{1,8})
([Gg]{3,})

    DOWNLOAD
        G4Catchall.py is hosted at

""",
    formatter_class=argparse.RawTextHelpFormatter)

parser.add_argument('--fasta', '-f',
                    type=str,
help='''Input file in fasta format containing one or more sequences can be
used.
Please note that, if not used, only the regular expression constructed using
```

```
given
arguments will be printed.
'''        )
parser.add_argument('--min_Gtract_for_extreme_loop','-E',
type=int,
help="""Defines the minimum G-tract length for permission of an extreme
loop. Works only with --extreme_loop. Can be set to 2 or 3. Default=3
""", default=3
parser.add_argument('--extreme_loop','--XL',
type=str,nargs='?',const='1..30',
help="""Allows search for an extreme loop. If precedes a secondary argument,
such as "1..20" also defines the limits of the loop. For default values do
not use a second argument. Default="1..30"
""", default=False)
parser.add_argument('--G2GQs_allowed',
action='store_true',
help="""Allows G-quadruplexes with G-tracts of two guanines. Not necessary
with --G2GQ_loop command.
""", default=False)
parser.add_argument('--G2GQ_loop','--G2L',
type=str,nargs='?',const='1..2',
help="""Allows G-quadruplexes with G-tracts of two guanines and defines
limits of loops for such G-quadruplexes if precedes a secondary argument,
such as "1..7". Do not use a secondary argument for default loop limits.
Default="1..2"
""", default=False)
parser.add_argument('--G3GQ_loop','--G3L',
type=str,
help="""Defines limits of loops for typical G-quadruplexes if precedes a
secondary argument, such as "1..7". Do not use for default loop limits.
Default="1..8"
""", default='1..8')
parser.add_argument('--max_imperfect_Gtracts','-I',
type=int,
help="""Defines the number of atypical or "imperfect" G-tracts allowed for
G-quadruplexes with G-tracts of at least 3 guanines. It can be set to 0,1
or 2. Default=1

""", default=1)
parser.add_argument('--bulge_only','-B',
action='store_true',
help="""Defines the nature of the imperfect G-tracts allowed. If used only
bulged G-tracts are allowed. Otherwise, mismatches are also allowed.
""", default=False)
parser.add_argument('--max_GQ_length','--max',
                    type=int,
help='''Maximum allowed G-quadruplex length for a single discovery. This
should be used with caution. If not used together with
--dont_merge_overlapping
discovered sequences may be longer than the given value. This parameter is
essentially designed for limiting cumulative negative impact of long loops.
''', default=False                    )
parser.add_argument('--no_reverse','-R',
action='store_true',
help="""By default the program searches both strands by reversing the regex.

If used only + strand is searched for matches.
```

```
""")
parser.add_argument('--dont_merge_overlapping',
action='store_true',
help="""Putative G-quadruplex-forming sequences may be found overlapping
on the same strand. By default the program merges these sequences. If
used, these are not overlapped. Using may result in huge number of
matches and cause memory issues.
""")
parser.add_argument('--include_flanks','-F',
action='store_true',
help="""By default the program extracts only matching sequences.
If used flanking nucleotides are also included in the search.
Please note if used G-quadruplex-forming sequences at the beginning or
ending of the sequences may be missed. Consider adding "N" to the edges
of the sequence if G-quadruplex forming sequences are expected to be
found at the very edge of the target sequence.
""")
parser.add_argument('--G4H',
action='store_true',
help="""By default the program extracts only matching sequences. If
used, discovered sequences are evaluated based on G4Hunter algorithm.
Low G4Hunter scores can be eliminated using --G4HThreshold argument.
""")
parser.add_argument('--G4HThreshold',
type=float,
help="""Removes G-quadruplex predictions with lower scores than the
preceding
threshold value. If used, --G4H usage is not necessary.
""")
args=parser.parse_args()
" -------------------------[ End of Parse arguments ]---------------------
"
" ----------------------------[  Reverse
complement ]-------------------------------- "
def ReverseComplement(seq):
    seq1 = 'ATCGNTAGCNatcgntagcn'
    seq_dict = {seq1[i]: seq1[i + 5] for i in range(20) if i < 5 or 10 <= i
< 15}
    return "".join([seq_dict[base] for base in reversed(seq)])

" ----------------------------[  End of Reverse
complement  ]-------------------------------- "
""" -----------------------------
[  Sorter ]---------------------------------
Code to sort list of lists
see http://www.saltycrane.com/blog/2007/12/how-to-sort-table-by-columns-in-
python/
"""

def sort_table(table, cols):
    """ sort a table by multiple columns
        table: a list of lists (or tuple of tuples) where each inner list
               represents a row
        cols:  a list (or tuple) specifying the column numbers to sort by
               e.g. (1,0) would sort by column 1, then by column 0
    """
    for col in reversed(cols):
```

```
        table = sorted(table, key=operator.itemgetter(col))
    return (table)

" -------------------------[ End of Sorter ]---------------------- "
" ------------------------------
[ Parameters  ]------------------------------- "
" ------------------------------[ Parse
Arguments ]-------------------------------- "

G4H_scoring=False
max_G4H_score=4
if args.G4HThreshold is not None: args.G4HunterScores=True
G2sAllowed =False
if args.G2GQs_allowed: G2sAllowed=True
shrtLoopMin,shrtLoopMax='1','2'
if args.G2GQ_loop:
    G2sAllowed = True
    shrtLoopMin,shrtLoopMax= args.G2GQ_loop.split("..")
typLoopMin,typLoopMax=args.G3GQ_loop.split("..")
extLoopMin, extLoopMax = '1','30'
ExtremeAllowed=False #args.extreme_loopAllowed
ExtremeAllowedForG2s = False
if args.extreme_loop:
    ExtremeAllowed=True
    extLoopMin,extLoopMax=args.extreme_loop.split("..")
if args.min_Gtract_for_extreme_loop==2:
    ExtremeAllowedForG2s=True
elif args.min_Gtract_for_extreme_loop==3:
    ExtremeAllowedForG2s=False
ImperfectTractsAllowed=args.max_imperfect_Gtracts
BulgedTractsOnly=args.bulge_only

if not G2sAllowed or not ExtremeAllowed:
    ExtremeAllowedForG2s = False

InclFlanks=args.include_flanks
if args.dont_merge_overlapping: MergeOverlapping=False
else: MergeOverlapping=True
NoReverse=args.no_reverse

" ----------------------------[  Define
RegEx  ]-------------------------------- "

bulgeOnly="[Gg]{2,}[ATUCatuc][Gg]+|[Gg]+[ATUCatuc][Gg]{2,}"
mismatch="[Gg]{2,}|[Gg]+[ATUCatuc][Gg]+"
Dimp1='?P<imp1>'
Dimp2='?P<imp2>'
imp='('+mismatch+')'
if BulgedTractsOnly:
    imp='('+bulgeOnly+')'
Timp1='?(imp1)'
Timp2='?(imp2)'
shrt='\w{'+shrtLoopMin+','+shrtLoopMax+'}'
Tshrt='?(G2GQ)'
Dshrt='?P<G2GQ>[Gg]{2}'
typ='\w{'+typLoopMin+','+typLoopMax+'}'
ext='\w{'+extLoopMin+','+extLoopMax+'}'
```

```
Dext='?P<extLoop>'
Text='?(extLoop)'

# Construct Tract 1:
Tract1='[Gg]{3,}'
if ImperfectTractsAllowed>0: Tract1=Tract1+'|('+Dimp1+imp+')'
if G2sAllowed: Tract1=Tract1+'|('+Dshrt+')'

# Construct Loop 1:
Loop1=typ
if ExtremeAllowed:Loop1=Loop1+'|('+Dext+ext+')'
if G2sAllowed:
    shrtAdd = shrt
    if ExtremeAllowedForG2s:
        shrtAdd = shrtAdd + '|(' + Dext + ext+')'
    Loop1 = Tshrt + '(' + shrtAdd + ')|('+Loop1+')'

# Construct Tract 2:
Tract2='[Gg]{3,}'
if ImperfectTractsAllowed>1: Tract2=Tract2+'|('+Dimp2+imp+')'
if ImperfectTractsAllowed>0: Tract2=Timp1+'('+Tract2+')|([Gg]{3,}|
('+Dimp1+imp+'))'
if G2sAllowed:
    Tract2=Tshrt+'[Gg]{2,}|('+Tract2+')'

# Construct Loop 2:
Loop2=typ
if ExtremeAllowed:
    Loop2=Text+Loop2+'|('+typ+'|('+Dext+ext+'))'
if G2sAllowed:
    shrtAdd=shrt
    if ExtremeAllowedForG2s:
        shrtAdd = Text+ shrtAdd + '|('+shrt+'|(' + Dext + ext+'))'
    Loop2=Tshrt+'('+shrtAdd+')|('+Loop2+')'

# Construct Tract 3:
Tract3='[Gg]{3,}'
if ImperfectTractsAllowed>1:
    Tract3=Timp2+Tract3+'|([Gg]{3,}|('+Dimp2+imp+'))'
if ImperfectTractsAllowed>0:
    Tract3=Timp1+'('+Tract3+')|([Gg]{3,}|('+Dimp1+imp+'))'
if G2sAllowed:
    Tract3=Tshrt+'[Gg]{2,}|('+Tract3+')'

# Combine all regions:
reg=r'('+Tract1+')  ('+Loop1+')  ('+ Tract2+') ('+Loop2+') ('+Tract3+')
('+Loop2+') ('+Tract3+')'
if InclFlanks: reg=r'\w'+reg+r'\w'


"""if no fasta is used, only return the regex and exit"""

if args.fasta is None:
    print (reg)
    exit()

""" Reverse forward match """
```

```
intab = 'actguACTGU'
outtab = 'tgacaTGACA'

if args.no_reverse is False:
    transtab = string.maketrans(intab, outtab)
    regrev = reg.translate(transtab)
else:
    regrev = ''
" ------------------------------[ End of Define
RegEx ]---------------------------------- "

" ------------------------------[ Check
Fasta ]---------------------------------- "

if args.fasta == '-':
    args.fasta = sys.stdin.readlines()
    if len(args.fasta) > 1:
        sys.exit('\nquadpareser.py: Only one input file at a time can be
processed:\n--fasta/-f: %s\n' % (args.fasta))
    args.fasta = args.fasta[0].strip()
" ------------------------------[ End of Check
Fasta ]---------------------------------- "

" ------------------------------
[ Functions ]---------------------------------- "

""" Reverse complement function """
def ReverseComplement(seq):
    seq1 = 'ATCGNWSMKRYBDHVatcgnwsmkrybdhv'
    seq2 = 'TAGCNWSKMYRVHDBtagcnwskmyrvhdb'
    seq_dict = {seq1[i]: seq2[i] for i in range(len(seq1))}
    return "".join([seq_dict[base] for base in reversed(seq)])

"""                                 G4Hunter
Modified code to score the discovered sequences based on G4Hunter algorithm.

see Amina Bedrat, Laurent Lacroix, Jean-Louis Mergny; Re-evaluation of G-
quadruplex propensity
 with G4Hunter, Nucleic Acids Research, Volume 44, Issue 4, 29 February
2016, Pages 1746-1759,
https://doi.org/10.1093/nar/gkw006
"""

def G4HScore(seq,minRepeat=2,penalizeGC=True):
    i=0
    baseScore=[]
    while i<len(seq):
        tractScore=[0]
        k=1
        GTract=False
        while seq[i]=="G":
            tractScore=[(min(k,4))] #derivation from original algorithm:
tractScore=[min(k-1,16)]*k
            # region derivation from original algorithm: if prev is "C"
apply bigger penalty. penalizes GCs
            if penalizeGC:
                try:
```

```
                    pass
                    if seq[i-k]=="C":baseScore[-1]=-2
                except:
                    pass
            # endregion
            k+=1
            i+=1
            GTract=True
            if i==len(seq): break
        if not GTract:
            while seq[i]=="C":
                tractScore=[max(-k,-4)] #derivation from original
algorithm: tractScore=[max(-k,-16)]*k
                # region derivation from original algorithm: if prev is "G"
apply bigger penalty. penalizes GCs
                if penalizeGC:
                    try:
                        pass
                        if seq[i - k] == "G": baseScore[-1] = 2
                    except:
                        pass
                # endregion
                k+=1
                i+=1
                GTract=True
                if i == len(seq): break
        baseScore=baseScore.__add__(tractScore)
        if not GTract: i += 1
        # print baseScore
    Score=0
    for value in baseScore:
        Score+=value
    return float(Score)/len(seq)


"""                                LIST SORTER
Code to sort list of lists
see http://www.saltycrane.com/blog/2007/12/how-to-sort-table-by-columns-in-
python/
"""
import operator

def sort_table(table, cols):
    """ sort a table by multiple columns
        table: a list of lists (or tuple of tuples) where each inner list
               represents a row
        cols:  a list (or tuple) specifying the column numbers to sort by
               e.g. (1,0) would sort by column 1, then by column 0
    """
    for col in reversed(cols):
        table = sorted(table, key=operator.itemgetter(col))
    return (table)
" ----------------------------[ End of
Functions ]------------------------------- "
psq_re_f = regex.Regex(reg,regex.VERBOSE|regex.MULTILINE)
psq_re_r = regex.Regex(regrev,regex.VERBOSE|regex.MULTILINE)
```

```
ref_seq_fh = open(args.fasta)
ref_seq = []
line = (ref_seq_fh.readline()).strip()
chr = re.sub('^>', '', line)
line = (ref_seq_fh.readline()).strip()
gquad_list = []
while True:
    while line.startswith('>') is False:
        ref_seq.append(line)
        line = (ref_seq_fh.readline()).strip()
        if line == '':
            break
    ref_seq = ''.join(ref_seq)
    for m in psq_re_f.finditer(ref_seq,overlapped=True):
        if not args.max_GQ_length or len(m.group(0))<=args.max_GQ_length:
            if MergeOverlapping and (len(gquad_list)>0) and gquad_list[-1]
[4]=="+" and (m.start()<=gquad_list[-1][2]) and (chr==gquad_list[-1][0]):
                orj=gquad_list[-1]
                new_seq=orj[5]+m.group(0)[orj[2]-m.start():]
                G4Hscore=G4HScore(new_seq,2,True)
                if abs(G4Hscore)>=(args.G4HThreshold):
                    gquad_list[-1]=[chr, orj[1], m.end(), m.end()-orj[1],
'+', new_seq, new_seq,G4HScore(new_seq,2,True)]
            else:
                G4Hscore=G4HScore(m.group(0),2,True)
                if abs(G4Hscore)>=(args.G4HThreshold):
                    gquad_list.append([chr, m.start(), m.end(),
len(m.group(0)), '+', m.group(0),
                                       m.group(0),G4HScore(m.group(0),2,True)])
# modification: added sequence again
    if args.no_reverse is False:
        if not args.max_GQ_length or len(m.group(0)) <= args.max_GQ_length:
            for m in psq_re_r.finditer(ref_seq,overlapped=True):
                if MergeOverlapping and (len(gquad_list) > 0) and
gquad_list[-1][4]=="-" and (m.start() <= gquad_list[-1][2]) and (chr ==
gquad_list[-1][0]):
                    orj = gquad_list[-1]
                    new_seq = orj[5] + m.group(0)[orj[2] - m.start():]
                    G4Hscore = G4HScore(new_seq, 2, True)
                    if abs(G4Hscore) >= (args.G4HThreshold):
                        gquad_list[-1] = [chr, orj[1], m.end(), m.end() -
orj[1], '-', new_seq, ReverseComplement(new_seq),G4HScore(new_seq,2,True)]
                else:
                    G4Hscore = G4HScore(m.group(0), 2, True)
                    if abs(G4Hscore) >= (args.G4HThreshold):
                        gquad_list.append([chr, m.start(), m.end(),
len(m.group(0)), '-', m.group(0), ReverseComplement(m.group(0)),
G4HScore(m.group(0),2,True)])  # modification: added reverse complement
    chr = re.sub('^>', '', line)
    ref_seq = []
    line = (ref_seq_fh.readline()).strip()
    if line == '':
        break
gquad_sorted = sort_table(gquad_list, (0, 1, 2, 3))
for line in gquad_sorted:
    line = '\t'.join([str(x) for x in line])
    print (line)
```

```
sys.exit()
```

Ek . Referans diziler için parametre tarama kod betiği

```
import os
import subprocess
import time, math
import re
import random as rd

def G4HScore(seq):
    i=0
    baseScore=[]
    while i<len(seq):
        tractScore=[0]
        k=1
        GTract=False
        # print(i)
        while seq[i]=="G":
            # region derivation from original algorithm: if prev is "C" apply bigger
penalty. penalizes GCs
            try:
                if seq[i-k]=="C":baseScore[-1]=-2
            except:
                pass
            # endregion
            tractScore=[(min(k-1,4))]*k #derivation from original algorithm:
tractScore=[min(k-1,16)]*k
            k+=1
            i+=1
            GTract=True
            if i==len(seq): break
        if not GTract:
            while seq[i]=="C":
                # region derivation from original algorithm: if prev is "G" apply
bigger penalty. penalizes GCs
                try:
                    if seq[i - k] == "G": baseScore[-1] = 2
                except:
                    pass
                # endregion
                tractScore=[max(-k+1,-4)]*k #derivation from original algorithm:
tractScore=[max(-k,-16)]*k
                k+=1
                i+=1
                GTract=True
                if i == len(seq): break
        baseScore=baseScore.__add__(tractScore)
        if not GTract: i += 1
        # print baseScore
    Score=0
    for value in baseScore:
        Score+=value
    return float(Score)/len(seq)

file="Mitochondria_NC_012920_1.fasta"

def
ConstructRegex(typLoopMax=7,shrtLoopMax=2,extLoopMax=30,typLoopMin=1,shrtLoopMin=1,e
xtLoopMin=1):
    G2sAllowed=True
```

```
    ExtremeAllowed=False
    ExtremeAllowedForG2s=False
    ImperfectTractsAllowed=2
    BulgedTractsOnly=True
    typLoopMax=str(typLoopMax)
    extLoopMax=str(extLoopMax)
    shrtLoopMax=str(shrtLoopMax)
    typLoopMin = str(typLoopMin)
    extLoopMin = str(extLoopMin)
    shrtLoopMin = str(shrtLoopMin)
    if not G2sAllowed or not ExtremeAllowed:
        ExtremeAllowedForG2s=False
    #region regex construct
    bulgeOnly="[G]{2,}[ATUC][G]+|[G]+[ATUC][G]{2,}"
    mismatch="[G]{2,}|[G]+[ATUC][G]+"
    Dimp1='?P<imp1>'
    Dimp2='?P<imp2>'
    imp='('+mismatch+')'
    if BulgedTractsOnly:
        imp='('+bulgeOnly+')'
    Timp1='?(imp1)'
    Timp2='?(imp2)'
    shrt='\w{'+shrtLoopMin+','+shrtLoopMax+'}'
    Tshrt='?(shrt)'
    Dshrt='?P<shrt>[G]{2}'
    typ='\w{'+typLoopMin+','+typLoopMax+'}'
    ext='\w{'+extLoopMin+','+extLoopMax+'}'
    Dext='?P<lloop>'
    Text='?(lloop)'

    # Construct Tract 1:
    Tract1='[G]{3,}'
    if ImperfectTractsAllowed>0: Tract1=Tract1+'|('+Dimp1+imp+')'
    if G2sAllowed: Tract1=Tract1+'|('+Dshrt+')'

    # Construct Loop 1:
    Loop1=typ
    if ExtremeAllowed:Loop1=Loop1+'|('+Dext+ext+')'
    if G2sAllowed:
        shrtAdd = shrt
        if ExtremeAllowedForG2s:
            shrtAdd = shrtAdd + '|(' + Dext + ext+')'
        Loop1 = Tshrt + '(' + shrtAdd + ')|('+Loop1+')'

    # Construct Tract 2:
    Tract2='[G]{3,}'
    if ImperfectTractsAllowed>1: Tract2=Tract2+'|('+Dimp2+imp+')'
    if ImperfectTractsAllowed>0: Tract2=Timp1+'('+Tract2+')|([G]{3,}|
('+Dimp1+imp+'))'
    if G2sAllowed:
        Tract2=Tshrt+'[G]{2,}|('+Tract2+')'

    # Construct Loop 2:
    Loop2=typ
    if ExtremeAllowed:
        Loop2=Text+Loop2+'|('+typ+'|('+Dext+ext+'))'
    if G2sAllowed:
        shrtAdd=shrt
        if ExtremeAllowedForG2s:
            shrtAdd = Text+ shrtAdd + '|('+shrt+'|(' + Dext + ext+'))'
        Loop2=Tshrt+'('+shrtAdd+')|('+Loop2+')'

    # Construct Tract 3:
```

```
    Tract3='[G]{3,}'
    if ImperfectTractsAllowed>1:
        Tract3=Timp2+Tract3+'|([G]{3,}|('+Dimp2+imp+'))'
    if ImperfectTractsAllowed>0:
        Tract3=Timp1+'('+Tract3+')|([G]{3,}|('+Dimp1+imp+'))'
    if G2sAllowed:
        Tract3=Tshrt+'[G]{2,}|('+Tract3+')'

    # Combine all regions:
    structure=r'('+Tract1+')  ('+Loop1+')  ('+ Tract2+') ('+Loop2+') ('+Tract3+')
('+Loop2+') ('+Tract3+')'
    return structure
    #endregion

def crateRandomList(limitStart,limitEnd,size):
    result=[]
    i=size
    while i>0:
        p=rd.randint(limitStart,limitEnd)
        if p not in result:
            result.append(p)
            i-=1
    return tuple(result)
def iterate(args,test=False):
    typLoopMax=7
    shrtLoopMax=2
    extLoopMax=30
    typLoopMin=1
    shrtLoopMin=1
    extLoopMin=1
    if len(args)==3:
        typLoopMax, shrtLoopMax, extLoopMax=args
    elif len(args)==6:
        typLoopMax, shrtLoopMax, extLoopMax, typLoopMin, shrtLoopMin,
extLoopMin=args

    quadparserCommand = r'python ImGQfinder.v2.py --noreverse -r "' +
ConstructRegex(typLoopMax,shrtLoopMax,extLoopMax,typLoopMin,shrtLoopMin,extLoopMin)
+ '"'
    output = subprocess.check_output(quadparserCommand + ' -f "' + file + '"',
shell=True)

    G4HScoreTreshold=0#.473#473
    TP=0
    FP=0
    G4List=[]
    nonG4List=[]

    useG4List=(52, 282, 67, 71, 133, 114, 90, 155, 63, 32, 124, 196, 26, 176, 184,
5, 287, 252, 147, 154, 128, 2, 1, 85, 293, 277, 88, 197, 172, 113, 77, 212, 14, 123,
258, 236, 265, 270, 246, 37, 9, 152, 86, 187, 170, 220, 243, 129, 189, 11, 24, 60,
193, 110, 87, 262, 80, 255, 36, 232, 8, 156, 198, 45, 263, 249, 218, 75, 93, 16,
217, 102, 141, 27, 120, 4, 104, 12, 219, 92, 190, 41, 74, 58, 195, 158, 138, 175,
18, 259, 247, 201, 62, 53, 54, 191, 43, 188, 185, 205, 20, 226, 70, 165, 275, 231,
108, 57, 202, 178, 145, 199, 222, 294, 122, 118, 61, 94, 139, 149, 146, 33, 95, 192,
79, 257, 285, 166, 292, 82, 210, 279, 107, 281, 66, 241, 125, 31, 163, 148, 106, 10,
251, 127, 143, 235, 213, 101, 103, 299, 264, 186, 223, 164, 271, 261, 204, 266, 267,
34, 40, 171, 121, 221, 157, 227, 6, 215, 142, 83, 200, 291, 216, 162, 240, 42, 237,
272, 283, 168, 268, 136, 91, 278, 173, 161, 30, 245, 48, 286, 203, 68, 132, 297, 3,
151, 225, 159, 22, 98)

    usenonG4List=(330, 313, 328, 314, 388, 315, 375, 346, 387, 333, 308, 371, 302,
344, 342, 393, 351, 319, 372, 321, 369, 312, 359, 376, 367, 365, 368, 341, 317, 383,
```

```
310, 331, 301, 361, 377, 347, 364, 350, 378, 352, 385, 336, 307, 323, 316, 374, 366,
382, 326, 337)
    if test:
        newG4=[]
        newnonG4=[]
        for i in range(1,299):

            if i not in useG4List:
                newG4.append(i)
        for i in range(299,393):
            if i not in usenonG4List:
                newnonG4.append(i)
        useG4List=tuple(newG4)
        usenonG4List=tuple(newnonG4)
    for line in output.splitlines():
        if line.__contains__("not GQ_"):
            G4no=int(re.search(r"[0-9]+",line).group(0))
            if G4no not in G4List and G4no in usenonG4List :
                score=G4HScore(re.search(r"[ATCUG]{5,}",line).group(0))
                if abs(score)>=G4HScoreTreshold:
                    G4List.append(G4no)
                    FP+=1
        elif line.__contains__("GQ_"):
            G4no=int(re.search(r"[0-9]+",line).group(0))
            if G4no not in nonG4List and G4no in useG4List:
                score=G4HScore(re.search(r"[ATCUG]{5,}",line).group(0))
                if abs(score)>=G4HScoreTreshold:
                    nonG4List.append(G4no)
                    TP+=1
    if TP==0 and FP==0: return #exit() #if nothing exists then exit without error.
    FN = len(useG4List)- TP
    TN = len(usenonG4List)- FP
    Youden=float(TP/float(TP+FN))+float(TN/float(TN+FP))-1 #youden's statistics
    precision=float(TP)/(TP+FP)
    if len(args)==3:
        report=
{"typLoopMax":typLoopMax,"shrtLoopMax":shrtLoopMax,"extLoopMax":extLoopMax,"MCC":You
den,"TPR":float(TP)/len(useG4List),"FPR":float(FP)/len(usenonG4List)}

    elif len(args)==6:
        report =
{"typLoopMax":typLoopMax,"shrtLoopMax":shrtLoopMax,"extLoopMax":extLoopMax,"MCC":You
den,"TPR":float(TP)/len(useG4List),"FPR":float(FP)/len(usenonG4List),"typLoopMin":ty
pLoopMin, "shrtLoopMin":shrtLoopMin,"shrtLoopMin":extLoopMin}
    return report

import multiprocessing
import itertools

if __name__=="__main__":
    print("sample regex:")
    print(ConstructRegex(7,4,30))
    p=multiprocessing.Pool(28)
    allParams=list(itertools.product(*[[t for t in range(1,16)],[s for s in
range(1,16)],[e for e in range(15,46)]]))
    results=p.map(iterate,allParams)
    print("----------------------------------------------------------------")

bestSet=[{"typLoopMax":0,"shrtLoopMax":0,"extLoopMax":0,"MCC":0,"TPR":0,"FPR":0}]
    for hit in results:
        if hit!=None:
            if hit["MCC"]>bestSet[0]["MCC"]:
                bestSet=[hit]
```

```
            elif hit["MCC"] == bestSet[0]["MCC"]:
                bestSet.append(hit)
    print("------------------------------------------------------------")

    Tests=[]
    print("parameter sets to test",len(bestSet))
    for params in bestSet:

Tests.append([params["MCC"],iterate((params["typLoopMax"],params["shrtLoopMax"],para
ms["extLoopMax"],),True)])

    bestTests=Tests
    totalJ=0
    for set in bestTests:
        totalJ+=set[1]["MCC"]
        print(set)
    avgJ=float(totalJ)/len(bestTests)
    print("average Youden:",avgJ)

"""LATEST CHANGES FINDS BEST PARAMETERS USING A TRAINING SET AND COMPARES TO A TEST
SET"""
```

Ek . Hızlandırılmış just-in-time derleme özellikli çiftli hizalama algoritması

```
import numpy as np
from numba import cuda,jit,int64
import time

class ProcessTimer:
    def __init__(self):
        self.starttime=time.time()
    def report(self):
        print time.time()-self.starttime
        self.starttime=time.time()
def ConvertToInt(seq):
    set=[0]*len(seq)
    for k in range(len(seq)):
        i=ord(seq[k])
        if (i == 84) or (i == 116): set[k] = 0
        if (i == 65) or (i == 97): set[k] = 1
        if (i == 71) or (i == 103): set[k] = 2
        if (i == 67) or (i == 99): set[k] = 3
        if (i == 78) or (i == 110): set[k] = 4
    return set
    #T:0,A:1,G:2,C:3,N:4,5:gap in seq1,6:gap in seq2, 7: mismatch

@jit(nopython=True,cache=True)
def SWPairwiseScore(seq1, seq2, Smatch, Smismatch, Sgap):
    M = len(seq1)+1
    N = len(seq2)+1
    mat = np.zeros(shape=(M,N),dtype=np.uint8)
    for m in range(1, M):
        for n in range(1, N):
            if seq1[m-1]==seq2[n-1]:
                score=Smatch
            else: score=Smismatch
            mat[m,n] = max(mat[m-1,n-1]+score,mat[m,n-1]+Sgap,mat[m-1,n]
```

```
+Sgap,0)
    return mat.max()

@jit(nopython=True,cache=True)
def SWPairwiseScoreMax(seq1, seq2, Smatch, Smismatch, Sgap):
    M = len(seq1)+1
    N = len(seq2)+1
    mat = np.zeros(shape=(2,N),dtype=np.uint8)
    Max = 0
    for m in range(1, M):
        for n in range(1, N):
            if seq1[m-1]==seq2[n-1]:
                score=Smatch
            else: score=Smismatch
            mat[1,n] = max(mat[0,n-1]+score,mat[1,n-1]+Sgap,mat[0,n]+Sgap,0)
            if Max < mat[1,n]:
                Max = mat[1,n]
        mat[0,:] = mat[1,:]
    return Max

Smatch = 1
Smismatch = -1
Sgap = -2

@cuda.jit
def SWPairwiseScoreCu(seq1, seq2,out):
    for m in range(1, 32):
        for n in range(1, 32):
            if seq1[m-1]==seq2[n-1]:
                score=Smatch
            else: score=Smismatch
            out[m,n] = max(out[m-1,n-1]+score,out[m,n-1]+Sgap,out[m-1,n]
+Sgap,0)
    #print out#.max()

@cuda.jit
def SWPairwiseScoreCuMax(seq1, seq2,Max):
    mat1 = cuda.shared.array(shape=32, dtype=int64)
    mat2 = cuda.shared.array(shape=32, dtype=int64)
    for k in range(0,31):
        mat1[k] = 0
        mat2[k] = 0
    for m in range(1, 32):
        for n in range(1, 32):
            if seq1[m-1]==seq2[n-1]:
                score=Smatch
            else: score=Smismatch
            mat2[n] = max(mat1[n - 1] + score, mat2[n - 1] + Sgap, mat1[n] +
Sgap, 0)
            if Max < mat2[n]:
                Max = mat2[n]
        mat1 = mat2

if __name__ == "__main__":
    timer = ProcessTimer()
    set1 = ConvertToInt("TATTTTCCCAT")
    set2 = ConvertToInt("TATCCCTTTAT")
```

```
    G4List = ['TGGGTTTGGGTTTGGGTTTGGGN', 'TGGGTTTGGGTTTGGGTTTGGGN',
'TGGGTTTGGGTTTGGGTGGGN', 'TGGGTTTGGGGTTGGGTTTGGGN',
              'TGGGTTTGGGGTTGGGTTTGGGN', 'TGGGGTTTGGGTTTGGGTTTGGGN',
'AGGGTTTGGGAAAGGGAAAGGGN',
              'TGGGTTTGGGGTAGGGTTTGGGN', 'TGGGTTTGGGGAAGGGTTTGGGN',
'TGGGTTTGGGGAAGGGTTTGGGN',
              'TGGGTTTGGGGAAGGGTTTGGGN', 'TGGGTTTGGGGAAGGGTTTGGGN']
    timer = ProcessTimer()
    result1 = SWPairwiseScore(set1, set2, 1, -1, -2)
    result2 = SWPairwiseScoreMax(set1, set2, 1, -1, -2)
    timer.report()
    print result1
    print result2
    M = len(set1) + 1
    N = len(set2) + 1
    out = np.zeros(shape=(M, N), dtype=np.int8)
    seq1 = cuda.to_device(np.array(set1))
    seq2 = cuda.to_device(np.array(set2))
    SWPairwiseScoreCu[1, 1](seq1, seq2, out)
    for i in range(100):
        SWPairwiseScoreCu[1,1](seq1,seq2,out)
        seq1 = cuda.to_device(np.array(set1))
        seq2 = cuda.to_device(np.array(set2))
    timer.report()
    print out
```

Ek . Çok izgeli benzerlik matrisi oluşturma betiği

```
from __future__ import print_function
#from Bio import pairwise2 as Pairwise #not required if myPairwise is used
import csv, time, sys, subprocess,os
from multiprocessing import Process,Queue,Manager,cpu_count
import numpy as np
import scipy.sparse as sp
import array
from numpy import int8
from numba import jit
import myPairwise2 as myPairwise
import h5py
# from colorama import init
# init()
import re
from common import DateCode,stripExt
#Variables
UsableThreads = cpu_count()-1
IncludePercentIdentity = True
Treshold = 0.8
SourceDir = os.curdir
Smatch, Smismatch, Sgapopen, Sgapextend = 2, -1, -2, -2     # Scores from
Bioinformatics and the cell, modern
                                                            # computational
approaches in genomics, proeomics and
                                                            #
transcriptomics. Chp 2 Xia X. 2007 ISBN: 978-0-387-71336-6.
# Smatch, Smismatch, Sgapopen, Sgapextend = 5, -4, -14, -14
```

```
G4ListFiles = [] #this could be modified with command line arguments
### Workflow: Find all .G4List files > IncrementalCOOMatrix >
PairwiseCompare() >>> PairwiseThread() > SaveAsCoo() > convert to csr >
SaveAsCsr()
# Moddified from IncrementalCOOMatrix: http://www.scipy-
lectures.org/advanced/scipy_sparse/csr_matrix.html
class IncrementalCOOMatrix(object):

    def __init__(self, shape, dtype):

        if dtype is np.int32:
            type_flag = 'i'
        elif dtype is np.int64:
            type_flag = 'l'
        elif dtype is np.float32:
            type_flag = 'f'
        elif dtype is float:
            type_flag = 'f'
        elif dtype is np.float64:
            type_flag = 'd'
        elif dtype is np.int8:
            type_flag='b'
        else:
            raise Exception('Dtype not supported.')
        self.dtype = dtype
        self.shape = shape
        self.rows = array.array('i')
        self.cols = array.array('i')
        self.data = array.array('f')

    def append(self, i, j, v):
        m, n = self.shape
        if (i >= m or j >= n):
            raise Exception('Index out of bounds')
        self.rows.append(i)
        self.cols.append(j)
        self.data.append(v)

    def appendTuple(self, t,v):
        m,n=self.shape
        if (t[0] >= m or t[1] >= n):
            raise Exception('Index out of bounds')
        self.rows.append(t[0])
        self.cols.append(t[1])
        self.data.append(v)
        if (t[0]!=t[1]): #if not diagonal
            self.rows.append(t[1])#switch around row and column
            self.cols.append(t[0])#switch around row and column
            self.data.append(v)

    def appendTupleVals(self, t):
        #print (sys.getsizeof(t)
        m,n=self.shape
        if (t[0] >= m or t[1] >= n):
            raise Exception('Index out of bounds')
        self.data.append(t[2])
        self.rows.append(t[0])
```

```
        self.cols.append(t[1])
        if (t[0]!=t[1]): #if not diagonal
            self.data.append(t[2])
            self.rows.append(t[1]) #switch around row and column
            self.cols.append(t[0]) #switch around row and column

    def tocoo(self):
        rows = np.frombuffer(self.rows, dtype=np.int32)
        cols = np.frombuffer(self.cols, dtype=np.int32)
        data = np.frombuffer(self.data, dtype=self.dtype)
        return sp.coo_matrix((data, (rows, cols)),
                             shape=self.shape)

    def tocsr(self):
        rows = np.frombuffer(self.rows, dtype=np.int32)
        cols = np.frombuffer(self.cols, dtype=np.int32)
        data = np.frombuffer(self.data, dtype=self.dtype)
        return sp.csr_matrix((data,(rows,cols)), shape=self.shape)

    def __len__(self):
        return len(self.data)

def PairwiseCompare():
    manager=Manager() #manager slows the processes but good against
corruption
    queue=manager.Queue() #all data saved here.
    processes=[]
    threadCount=UsableThreads
    status_activeThreadCount=0
    def Status():
        status = "\ndate: "+time.ctime().__str__() + " active threads: " +
status_activeThreadCount.__str__() + " queue size: " +
queue.qsize().__str__()
        print (status)

    #active threads: 16-18, aligned sequence: 38-45,
    for k in range(threadCount):
processes.append(Process(target=PairwiseThread,args=(k,threadCount,G4List,qu
eue,Smatch,Smismatch,Sgapopen,Sgapextend,Treshold,IncludePercentIdentity,)))
        processes[-1].start()
        status_activeThreadCount+=1
        print ("\r|>active process count: ", status_activeThreadCount," ")
    while (queue.qsize()<1):
        time.sleep(0.1)
    active=True
    while (active):
        if (queue.qsize()==0):
            time.sleep(0.01)
            if (True in [p.is_alive() for p in processes]) == False:
                active = False
                print ("\033[39;49m\n+pairwise alignments are complete.")
            continue
        else:
            SimilarityMatrix.appendTupleVals(queue.get())
            queuesize = queue.qsize()
            if queuesize%1000==0:
                print ("\r\033[30C\033[33;45m|>queue size: ",
```

```
queuesize.__str__(),"\033[K",end="")

    print ("+joining completed processes.")
    for p in processes:
        p.join()
        status_activeThreadCount -= 1
        print ("\r|>active process count: ",status_activeThreadCount,end="")
    print ("\n+collecting last bits from queue.")
    while (queue.qsize() >0):
        SimilarityMatrix.appendTupleVals(queue.get())
        queuesize = queue.qsize()
        if queuesize % 1000 == 0:
            print ("\r\033[33;45m|>queue size: ",
queuesize.__str__(),end="")
    return

def PairwiseThread(thread, threadCount, G4List, queue, Smatch, Smismatch,
Sgapopen, Sgapextend, Treshold,IncludePercentIdentity=False):
    totalG4s = len(G4List)
    IntList=[myPairwise.ConvertToInt(G4List[g]) for g in range(totalG4s)]
    for k in range(thread, totalG4s, threadCount):
        if IncludePercentIdentity:
            queue.put((k, k,1.))
        else:
            queue.put((k, k, 1))
        for l in range(k):
            k_len=len(G4List[k])
            alignmentScore=myPairwise.SWPairwiseScore(IntList[k],
IntList[l], Smatch, Smismatch, Sgapopen)
            Smax = Smatch * min(k_len, len(G4List[l]))
            Pidentity = float(alignmentScore) / Smax
            if Pidentity > Treshold:
                if IncludePercentIdentity:
                    queue.put((k, l, Pidentity))
                    #queue.put((l, k, Pidentity))
                else:
                    queue.put((k, l, 1))
                    #queue.put((l, k, 1))
        if k%100==0:
            print ("\r\033[30;47m|>completed sequence:",k,"  ",end="")
    return

class ProcessTimer:
    def __init__(self):
        self.starttime=time.time()
    def report(self):
        result= time.time()-self.starttime
        self.starttime=time.time()
        print (result)

def SaveAsCoo(coomatrix,summary,filename):
    import h5py
    import numpy as np
    from scipy.sparse import coo_matrix
    if type(filename) is str:
        h5pyFile=h5py.File(filename,"w")
    elif type(filename) is h5py.File:
```

```
        h5pyFile=filename
    else:
        sys.exit("Could not create a file. Check target file name.")
    h5pyFile.attrs["parameters"]=np.string_(summary)

h5pyFile.create_dataset("data",data=coomatrix.data,dtype=coomatrix.dtype)
    h5pyFile.flush()
    h5pyFile.create_dataset("rows",data=coomatrix.rows,dtype=np.int32)
    h5pyFile.flush()
    h5pyFile.create_dataset("cols",data=coomatrix.cols, dtype=np.int32)
    h5pyFile.close()

def SaveAsCsr(csrmatrix,summary,filename):
    import h5py
    import numpy as np
    #from scipy.sparse import csr_matrix
    if type(filename) is str:
        h5pyFile=h5py.File(filename,"w")
    elif type(filename) is h5py.File:
        h5pyFile=filename
    else:
        sys.exit("Could not create a file. Check target file name.")
    h5pyFile.attrs["parameters"]=np.string_(summary)

h5pyFile.create_dataset("data",data=csrmatrix.data,dtype=csrmatrix.dtype)
    h5pyFile.flush()
    h5pyFile.create_dataset("indices",data=csrmatrix.indices,dtype=np.int32)
    h5pyFile.flush()
    h5pyFile.create_dataset("indptr",data=csrmatrix.indptr, dtype=np.int32)
    h5pyFile.close()

if __name__=="__main__":
    # init()  # initiates colorama
    # region Variables
    timer = ProcessTimer()
    G4List = []
    G4positionList = []
    # endregion
    # region Settings printed on screen
    print ("\033[39;49m")
    print ("\n------------------ Settings -----------------")
    print ("Usable thread count=", UsableThreads)
    print ("Identity treshhold=", Treshold)
    print ("Alignment scores, match=", Smatch, "mismatch=", Smismatch, "gap
openning=", Sgapopen, "gap extension=", Sgapextend)
    print ("Include identity matrix=",IncludePercentIdentity)
    # endregion
    # region Find and parse G4List files
    print ("\n----------- Finding G4List files ------------")
    targetFile=""
    G4ListFiles=[]
    for File in os.listdir(SourceDir):
        if File.endswith(".G4List"):
            G4ListFiles.append(File)
    if len(G4ListFiles) < 1:
        sys.exit("No .G4List file detected in the current
directory.quiting.")
```

```
    for i in range(len(G4ListFiles)):
        print (i+1,G4ListFiles[i])
    if len(G4ListFiles)==1:
        targetFile=G4ListFiles[-1]
    else:
        choice=int(input("Please type the number of the G4List file to
analyse."))
        targetFile=G4ListFiles[choice-1]
    pre_params=[]
    fileContent = open(targetFile, "r")
    for line in fileContent:
        if line.startswith("#") == False:
            G4List.append(re.sub('[^a-zA-Z]+', '',line.split("\t")[6]))
#[7] corresponds to G4 forming strand sequence according to G4List format
#re removes non alphabetic characters
        else:
            print(line)
            pre_params.append(line)
    # endregion
    if IncludePercentIdentity:
        SimilarityMatrix =
IncrementalCOOMatrix((len(G4List),len(G4List)),dtype=np.float32)
    else:
        SimilarityMatrix = IncrementalCOOMatrix((len(G4List), len(G4List)),
dtype=np.int8)
    # region G4s from the file are printed on the screen
    print ("\n---------------- Listing G4s ----------------")
    print ("G4s found",len(G4List))
    print (G4List[:30])
    # endregion
    print ("\n------------- Pairwise Comparison -------------")
    PairwiseCompare()
    print ("\n------------------- REPORT -------------------")
    print ("the first entries of Sparse G4 Identity Matrix:")
    print ("matrix.cols:",SimilarityMatrix.cols[:5])
    print ("matrix.rows:",SimilarityMatrix.rows[:5])
    print ("matrix.data:",SimilarityMatrix.data[:5])
    print ("time to complete:",end="")
    timer.report()
    allParameters= "".join(pre_params)+"# Matrix construction parameters:
threshold: "+ Treshold.__str__()+ " match score: "+Smatch.__str__()+ "
mismatch penalty: "+Smismatch.__str__()+ " gap open penalty:
"+Sgapopen.__str__()+" gap extension penalty: "+Sgapextend.__str__()+"
identity percent included: "+IncludePercentIdentity.__str__()

    print ("\n------------------- OUTPUT -------------------")
    filename=stripExt(targetFile)+ ".X"+DateCode() +".coo"
    print ("saving coo matrix data to " + filename)
    SaveAsCoo(SimilarityMatrix,allParameters,filename)
    print ("converting to csr matrix.")
    SimilarityMatrix = SimilarityMatrix.tocsr()
    filename=stripExt(targetFile)+ ".X"+DateCode()+".csr"
    print ("saving csr matrix data to "  + filename)
    SaveAsCsr(SimilarityMatrix, allParameters, filename)
    # print (SimilarityMatrix.todense() #matrix may be too big to handle
this command. Use with caution.
```

Ek . Çok izgeli G-dörtlüsü gruplama betiği

```
from multiprocessing import
Value,Pipe,Process,Pool,Manager,sharedctypes,cpu_count,Array
from time import time,sleep
import ctypes
import numpy as np
import os,sys
import psutil
import linecache
from common import DateCode
stime=time()
threshold=.8
min_comm_size=10

def progress_bar(last_index,total,stage):
    pass
    cur_prog=round(last_index/total*100)
    print("\r|"+"-"*cur_prog+" "*(100-cur_prog)+"| "+stage+" | working
on:"+str(last_index),end="")       #str(psutil.virtual_memory()[4]),end="")

def progress_bar_thread(progress):
    last_value=0
    step=1
    while True:
        progress_value = int(progress.value * 100)
        if progress_value==0 and last_value>90:
            step+=1
        last_value=progress_value
        print("\r|" + "-" * progress_value + " " * (100 - progress_value) +
"| Progress %"+str(progress_value)+" STEP "+str(step), end="")
        sleep(1)

if __name__=="__mp_main__":
    with open("setting.ini","r") as settingFile:
        shared_indices_size,shared_indptr_size= (int(x) for x in
settingFile.read().split(";"))
        settingFile.close()
    fields = [('neighbor', ctypes.c_int32 * shared_indices_size)]
    indices_type = type('indices_type', (ctypes.Structure,), {"_fields_":
fields})
    fields = [('vertex', ctypes.c_int32 * shared_indptr_size)]
    indptr_type = type('indptr_type', (ctypes.Structure,), {"_fields_":
fields})

def set_size_ini(indices_size,indptr_size):
    settingFile=open("setting.ini","w")
    settingFile.write(indices_size.__str__()+";"+indptr_size.__str__())
    settingFile.close()

def ChckPnt(i=-1):
    global stime
    print ("Check point:",i,time()-stime)
    stime=time()
```

```
def LoadFromCSR(filename):
    from numpy import array,string_
    import h5py
    h5pyFile=h5py.File(filename,"r")
    global main_data, main_indices, main_indptr, main_parameters
    main_data=array(h5pyFile.get("data"))
    progress_bar(1,3,"file load")
    main_indices=array(h5pyFile.get("indices"))
    progress_bar(2,3,"file load")
    main_indptr=array(h5pyFile.get("indptr"))
    progress_bar(3,3,"file load")
    main_parameters=str(h5pyFile.attrs["parameters"])[2:-1]
    h5pyFile.close()
    print("\r",filename,"loaded.")

def get_neigbors(index): #RETURNS NEIGHBOURS OF VERTEX
    return
shared_indices.neighbor[shared_indptr.vertex[index]:shared_indptr.vertex[ind
ex + 1]]

def FindCommonSet(mySet,vertex2):
    return mySet.intersection(get_neigbors(vertex2))

def FindFullyConnected_init(indices,indptr,progress_v,lock):
    global shared_indices,shared_indptr
    shared_indices,shared_indptr=indices,indptr
    global fileLock
    fileLock=lock
    global progress
    progress=progress_v

def FindFullyConnected(vertex):
    myNeighbours_list=get_neigbors(vertex)
    myNeighbours_set=set(myNeighbours_list)
    myNeighbours_count=len(myNeighbours_list)
    finalSet={x for x in myNeighbours_list if
len(FindCommonSet(myNeighbours_set,x))/myNeighbours_count>=threshold }
    # finalSet.add(vertex)  # BECAUSE THE MATRIX ALREADY INCLUDES SELF AS
CONNECTED, NO NEED TO ADD IT AGAIN.

    fileSize=GetSizeFromFile()
    if len(finalSet)>=min_comm_size:
        for i in range(fileSize):
            thisSet=ReadFromFile(i)
            if finalSet.issubset(thisSet):
                return
        text = ""
        for s in finalSet:
            text += str(s) + ' '
        text += '\n'
        fileLock.acquire()
        with open("clusters.txt", 'a') as f:
            f.write(text)
            f.close()
        fileLock.release()
```

```
    progress.value=vertex/shared_indptr_size
    # progress_bar(vertex,shared_indptr_size, "STEP 1")

def Merger_init(step2_io,progress_v):
    global step2_out_list
    step2_out_list=step2_io
    global progress
    progress=progress_v

def ReadFromFile(i):
    return set([int(x) for x in linecache.getline("clusters.txt", i).split("
") if x!="" and x!="\n"])

def GetSizeFromFile():
    fname="clusters.txt"
    i=0
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1

def Merger(i):
    # INSTEAD OF TRYING TO DELETE A LINEFROM FILE, THIS FILLS AN ARRAY WITH
0s AND 1s DEPENDING ON IF THE SET TO BE DELETED OR NOT.
    # linecache.clearcache() #ODDLY THIS COMMAND INCREASES MEMORY USE
    length=GetSizeFromFile()
    curSet=ReadFromFile(i)
    for index in range(i+1,length):
        if step2_out_list[index]:
            thisSet=ReadFromFile(index)
            if curSet.issubset(thisSet):
                step2_out_list[i]=False
                break
            if len(thisSet.intersection(curSet))/len(thisSet.union(curSet))
>= threshold:
                step2_out_list[index] = False
                continue
    progress.value=i/length

    # progress_bar(i,length, "STEP 2")
if __name__=="__main__":
    progress=Value('f',0.0)
    progress_process=Process(target=progress_bar_thread,args=(progress,))
    print("----------PARAMETERS----------")
    print("Threshold:",threshold)
    print("Minimum community size:",min_comm_size)
    print("-----DISCOVERED CSR FILES-----")
    # FIND AND CHOOSE A G4LIST FILE TO ANALYSE
    G4ListFiles=[]
    for File in os.listdir(os.curdir):
        if File.endswith(".csr"):
            G4ListFiles.append(File)
    if len(G4ListFiles) < 1:
        sys.exit("No .G4List file detected in the current
directory.quiting.")
    for i in range(len(G4ListFiles)):
        print (i+1,G4ListFiles[i])
```

```
    if len(G4ListFiles)==1:
        target_G4List=G4ListFiles[-1]
    else:
        choice=int(input("Please type the number of the G4List file to
analyse."))
        target_G4List=G4ListFiles[choice-1]
    print("-----LOADING FROM FILE-----")
    LoadFromCSR(target_G4List)
    indices_size=len(main_indices)
    indptr_size=len(main_indptr)
    print("Length of indices:",indices_size)
    print("Length of indptr:",indptr_size)
    progress_bar(0,4,"type declerations")
    set_size_ini(indices_size,indptr_size)
    fields = [('neighbor', ctypes.c_int32 * indices_size)]
    indices_type = type('indices_type', (ctypes.Structure,), {"_fields_":
fields})
    progress_bar(1,4,"type declerations")
    fields = [('vertex', ctypes.c_int32 * indptr_size)]
    indptr_type = type('indptr_type', (ctypes.Structure,), {"_fields_":
fields})
    progress_bar(2,4,"type declerations")
    # CONVERT TYPE OF main_indices AND main_indptr INTO SOMETHING MORE
USEFUL
    main_indices=indices_type((*main_indices,))
    progress_bar(3,4,"type declerations")
    main_indptr=indptr_type((*main_indptr,))
    progress_bar(4,4,"type declerations")
    dataFile=open("clusters.txt","w")
    dataFile.close()
    print("\rTotal processors found:",cpu_count())
    print("-----CANDIDATE DISCOVERY STARTS-----")
    progress_process.start()
    sstime=time() #START OF HEAVY DUTY WORK
    manager=Manager()
    fileLock=manager.Lock()
    # print("Find communities initiates")

pool_step1=Pool(processes=cpu_count(),initializer=FindFullyConnected_init,in
itargs=(main_indices,main_indptr,progress,fileLock)) #FINDCOMMUNITIES
ALGORITHM IS HIDDEN HERE
    pool_step1.imap_unordered(FindFullyConnected,range(indptr_size))
    pool_step1.close()
    pool_step1.join()
    # print("-----CANDIDATE CREATION COMPLETE------")
    step2_out_list=Array("b",[True for x in range(GetSizeFromFile())])

pool_step2=Pool(processes=cpu_count(),initializer=Merger_init,initargs=(step
2_out_list,progress))
    results=pool_step2.imap_unordered(Merger, range(GetSizeFromFile()))#
step1_out_list)
    pool_step2.close()
    pool_step2.join()
    final_list=[ ReadFromFile(x) for x in range(GetSizeFromFile()) if
step2_out_list[x]]
    progress_process.terminate()
    print("\r----------ANALYSIS COMPLETE----------")
```

```
    print("Total time:", time() - sstime)
    print("Clusters found:",len(final_list))
    target_G4List=target_G4List.replace(".csr","")
    target_file=target_G4List+".C"+DateCode()+".G4clstr"
    main_parameters=main_parameters.replace(r"\n","")
    with open(target_file,"w") as f:
        f.write(main_parameters+" # Cluster discovery parameters: threshold:
"+ str(threshold)+" minimum cluster size: "+ str(min_comm_size)+ "\n")
        for a_set in final_list:
            f.write(str(a_set))
            f.write("\n")
        f.close()
    print("Saved to "+target_file)
    print("File size:",os.stat(target_file).st_size,"bytes")
```

Ek . Glam2 otomatizasyonu ile seri motif oluşturma betiği

```
import sys,os,subprocess

G4ListFile=open("NC_000001.11 Homo sapiens chromosome 1, GRCh38.p7 Primary
Assembly.G4List","r")
G4ClstrFile=open("ClstrFile.G4Clstr","r")
parameter=""
parameter=G4ListFile.readline()
print(parameter)
G4List=list()
G4DefList=list()
for line in G4ListFile.readlines():
    G4List.append(line.strip("\n").split("\t")[6] )
    G4DefList.append(":".join(line.strip("\n").split("\t")[:5]))
print ("G4List file loaded: ",G4List[:3],G4DefList[:3])
shortest=""
parameter=G4ClstrFile.readline()
print(parameter)
ClstrList=[ [int(x) for x in line.strip("{}\n").split(", ") ] for line in
G4ClstrFile.readlines()]
ClstrList=sorted(ClstrList,key=len)
print("G4Clstr file loaded: ",ClstrList[:3])
#DOES FOR ONE CLUSTER
totalTypes=0
totalVariants=0
for item in ClstrList[:]:
    clstr=item
    shortest=min([ G4List[pos] for pos in clstr],key=len)
    # print(shortest)
    tempClstrFile=open("tempClstr.fa","w")
    variants=[]
    for entry,index in enumerate(clstr):
        if G4List[index][1:-1] not in variants:
            variants.append(G4List[index][1:-1])
        tempClstrFile.write(">"+G4DefList[index]+"\n")
        tempClstrFile.write(G4List[index]+"\n")
    tempClstrFile.close()
    dirName=shortest[1:-1] #+"_"+str(len(clstr))+"_"+str(len(variants))
    print(dirName)
```

```
    if dirName in os.listdir(os.curdir):
        print("already found:",dirName)
        continue
    #  No need to check if the directory exists, GLAM2 will not overwrite
    bashCommand="/home/osman/meme/bin/glam2 n "+tempClstrFile.name+" -o
"+dirName+" -r 1"
    print(bashCommand)
    process=subprocess.Popen(bashCommand.split(),stdout=subprocess.PIPE)
    output,error=process.communicate()
    print(dirName," G4 cluster. Return code:",process.returncode)
    if process.returncode==1:
        continue
    totalTypes+=1
    totalVariants+=variants.__len__()
    G4ReportFile=open("./"+dirName+"/G4Report.txt","a")
    G4ReportFile.write("size of cluster: "+str(len(clstr))+" variant count:
"+str(len(variants))+"\n")
    G4ReportFile.writelines([x+"\n" for x in variants])
    G4ReportFile.close()
print("total G4 groups:",totalTypes,"\ntotal G4 variants",totalVariants)
```

Ek . G-dörtlülerini gen anotasyonlarındaki pozisyonlarına göre filtreleme betiği

```
import re
from itertools import product,repeat
import itertools
import common
from multiprocessing import Process,Pool,Manager,cpu_count

def tonumeric(value):
    if value[0].isnumeric() or value[0]=="-":
        try:
            return int(value)
        except ValueError:
          try:
              return float(value)
          except ValueError:
              return value
    else:
        return value

def toChr(value):
    chrNames=[str(x) for x in range(1,23)]+["X","Y"]
    txtlen="chromosome".__len__()
    txtstart=value.find("chromosome")
    if txtstart<0:
        txtstart = value.find("chr")
        txtlen="chr".__len__()
    if txtstart < 0: return value
    vals=[]
    for k in value[txtstart+txtlen:]:
        if k in chrNames and len(vals)>0:
            vals[-1]=vals[-1]+k
        else:
            vals.append(k)
```

```
    return tonumeric(vals[0])

def match(g4chr,featurechr):
    featurechr=str(featurechr).strip("chromosome ")
    if len(featurechr)<3:
        featurechr="chromosome "+featurechr
    if featurechr in g4chr:
        return True
    else: return False

#region find gene annotations
class annotations():
    __lines_unread__=0
    def __init__(self,filename=""):
        self.filename=filename
        self.pseudogene=[]
        self.gene=[]
        self.CDS=[]
        self.UTR=[]
        self.five_prime_utr=[]
        self.three_prime_utr=[]
        self.transcript=[]
        self.exon=[]
        self.primary_transcript=[]
        self.promoter=[]
        self.promoterLength=2000
        if filename!="":
            self.read_from_file(filename)

    def List_Features(self):
        return [attr for attr in dir(self) if not attr.startswith("__") and
type(getattr(self,attr))==list]

    def read_from_file(self,filename):
        self.filename = filename
        print("Loading annotation file",filename)
        annotation_file = open(filename, "r")
        for line in annotation_file.readlines():
            line=line.strip("\n").split("\t")
            self.__lines_unread__=0
            try:
                this_feature=line[2]
                getattr(self,this_feature).append([tonumeric(x) for x in
line])
            except:
                self.__lines_unread__+=1
        annotation_file.close()
        self.promoter=self.discoverPromoters(self.promoterLength)
        print("File loaded.")

    def __repr__(self):
        print("\n--Annotation file summary--")
        for annotation_list in self.List_Features():
            print( len(getattr(self,annotation_list)),annotation_list,
"annotations found.")
        print(self.__lines_unread__, "lines could not be read.")
        return
```

```
    def discoverPromoters(self, length=2000):
        promoters = []
        for feature in self.gene:
            if feature[6] == "+": promoters.append(feature[0:2] +
["promoter"] + [feature[3] - length, feature[3]]+feature[5:])
            if feature[6] == "-": promoters.append(feature[0:2] +
["promoter"] + [feature[4], feature[4] + length]+feature[5:])
        return promoters
    def __add__(self, other):
        for attr in self.List_Features():
self.__setattr__(attr,self.__getattribute__(attr)
+other.__getattribute__(attr))
            self.filename=self.filename+";"+other.filename
            return self
#endregion
class G4List():
    def __init__(self,filename):
        self.list=list()
        self.firstline=""
        self.filename=filename
        if type(filename)==str:
            print("Loading:",filename)
            with open(filename,"r") as file:
                file=open(filename,"r").readlines()
                self.firstline=file[0]
            for line in file:
                content=line.strip("\n").split("\t")
                if content.__len__()>=7:
                    self.list.append([tonumeric(x) for x in content])
                elif line.__len__()>0:
                    print(line)

def doesOverlap(G4entry, featureList):
    overlap = False
    for feature in featureList:
        if match(G4entry[0], feature[0]):
            # print(G4entry[0], feature[0])
            if G4entry[1] >= feature[3] and G4entry[2] <= feature[4]:
                overlap = True
                break
    return overlap

#region rules
from enum import Enum
class option(Enum):
    ignore=0
    must_include=1
    must_exclude=2
    cannot_include=3
    cannot_exclude=4

class annotation_filter_rules():
    def __init__(self,for_annotations=annotations()):
        for attr in for_annotations.List_Features():
            setattr(self,attr,option.ignore)
            # print(attr,getattr(self,attr))
```

```
    def __repr__(self):
        return "\n".join([attr+": "+str(self.__getattribute__(attr)) for
attr in dir(self) if not attr.startswith("__") and
type(getattr(self,attr))==option])
    def attrList(self):
        return [attr for attr in dir(self) if not attr.startswith("__") and
type(getattr(self,attr))==option]
#endregion

def chckG4overlaps( tupleEntry):
    (G4entry, annotations, rules)=tupleEntry
    ignore = False
    for featureType in rules.attrList():
        if rules.__getattribute__(featureType) == option.ignore: continue
        featureList = []
        try:
            featureList += annotations.__getattribute__(featureType)
        except:
            pass
        overlap = doesOverlap(G4entry, featureList)
        if rules.__getattribute__(featureType) == option.must_include:
            if overlap:
                continue
            else:
                ignore = True
                break
        elif rules.__getattribute__(featureType) == option.must_exclude:
            if overlap:
                ignore = True
                break
            else:
                continue
    if ignore:
        return None
    else:
        print(G4entry)
        return G4entry

if __name__=="__main__":
    pass
    rules = annotation_filter_rules()
    # rules.promoter=option.must_include
    # rules.gene=option.must_exclude
    rules.five_prime_utr=option.must_include
    # rules.three_prime_utr=option.must_include
    rules.CDS=option.must_exclude
    rules.pseudogene=option.must_exclude
    print("\n--Rules--")
    print(rules)
    print("\n--Annotation file(s)--")
Annotations=annotations("Homo_sapiens.GRCh38.94.gtf")
+annotations("ref_GRCh38.p12_top_level.gff3")
    print("\n--Loading G4List file--")
    G4ListFilename="Homo sapiens, GRCh38.p7 Primary Assembly.G4List"
    G4list=G4List(G4ListFilename)
    G4ListParameterLine=G4list.firstline
    G4list=G4list.list
```

```
    Results=[]
    with Pool(30) as pool:
        iters=[(x,Annotations,rules) for x in G4list]
        Results=pool.map(chckG4overlaps,iters)
newG4ListFile=open(common.stripExt(G4ListFilename)+"_F"+common.DateCode()
+".G4List","w") newG4ListFile.write(G4ListParameterLine.replace("\n","")
+";"+rules.__repr__().replace("\n",",")+" annotation
files:"+",".join(Annotations.filename )+" original G4List
file:"+G4ListFilename+"\n")
    for entry in Results:
        if entry == None: continue
        newG4ListFile.write("\t".join([str(x) for x in entry])+"\n")
```

Ek .G4List dosya formatından fasta formatına çevirme kod betiği

```
filesToConvert=["Homo sapiens, GRCh38.p7 Primary Assembly_FicklW.G4List",
                "Homo sapiens, GRCh38.p7 Primary Assembly_Fj12nt.G4List",
                "Homo sapiens, GRCh38.p7 Primary Assembly_FicbmU.G4list"]

for filename in filesToConvert:
    with open(filename,"r") as file:
        desc=file.readline()
        with open(filename+".fa","w") as target:
            target.write("//"+desc)
            for line in file:
                content=line.split("\t")
                target.write(">"+content[6]+" "+",".join(content[0:4])
+"\n"+content[6]+"\n")
```

Ek . G-dörtlüleri ile çakışan patojenik ve yüksek ihtimalle patojenik ClinVar varyantlar seçilim kod betiği

```
import os
import re
VariantsFolder=os.curdir+"\\"
G4ListFilename="..\\Homo sapiens, GRCh38.p7 Primary Assembly.G4List"
#use this table for accessing variant files
ChrVarFileTable={}
for chrnumber in range(1,23):
    ChrVarFileTable.update({str(chrnumber):"chr"+str(chrnumber)+".tsv"})
ChrVarFileTable.update({"X":"chrX.tsv","Y":"chrY.tsv"})
FilenamesAvailable=os.listdir(VariantsFolder)
for filename in ChrVarFileTable.values():
    if filename not in FilenamesAvailable:
        exit(code="001 Can not find target variant files: "+filename)
G4ListFile=open(G4ListFilename,"r")
CurrentChrNo=1
CurrentChr=list(ChrVarFileTable.keys())[CurrentChrNo-1]
print("Next chromsome:", CurrentChr)
CurrentVariantLineNo=10
CurrentVariantFile=open(VariantsFolder+ChrVarFileTable[CurrentChr],"r").read
lines()
print(G4ListFile,CurrentVariantFile)
G4CountWithVariants=0
Results=[]
ResultsFile=open("G4Variants.tsv","w")
ResultsFile.write("source\tStart\tStop\tlength\tstrand\tstrand sequence\tG4
```

```
sequence\tG4H score\t"+"Variant ID Start  Stop   Variant type   Gene
Molecular consequences Most severe clinical significance  1000G minor allele
1000G MAF  GO-ESP minor allele    GO-ESP MAF ExAC minor allele  ExAC MAF
Publications (PMIDs)   Variant allele Transcript change  RefSeq Protein
change Molecular consequence  HGVS_c HGVS_g HGVS_ng    HGVS_p Condition
Most severe clinical significance  Submitters Highest review status  Last
evaluated\n")
for line in G4ListFile:
    lineParse=line.split("\t")
    # print(lineParse)
    if lineParse.__len__()<3:
        continue
    if not lineParse[0].__contains__("chromosome "+CurrentChr):
        CurrentChrNo+=1
        CurrentChr=list(ChrVarFileTable.keys())[CurrentChrNo-1]
        CurrentVariantLineNo=10
CurrentVariantFile=open(VariantsFolder+ChrVarFileTable[CurrentChr],"r").read
lines()
        print("Next chromsome:",CurrentChr)
    G4start=int(lineParse[1])
    G4end=int(lineParse[2])
    CurVariantFileLength=len(CurrentVariantFile)
    while (CurrentVariantLineNo<CurVariantFileLength):
        variantParse=CurrentVariantFile[CurrentVariantLineNo].split("\t")
        variantStart=int(variantParse[1])
        variantEnd=int(variantParse[2])
        if variantEnd<G4start:
            CurrentVariantLineNo += 1
            continue
        elif variantStart<G4end:
            try:
                if not lineParse[1]==Results[-1][1]:
                    G4CountWithVariants+=1
            except:
                pass
            lineParse[-1]=lineParse[-1].strip("\n")
            Results.append(lineParse+variantParse)
            ResultsFile.write("\t".join(Results[-1]))
            print(lineParse,variantParse)
            CurrentVariantLineNo += 1
            continue
        break
print("total G4s with ClinVar Variants:",G4CountWithVariants)
print("total ClinVar Variants detected:",len(Results))
```

Ek . TCGA varyasyon verileri filtreleme kod betiği

```
import re

ChromosomeNo_dict={"1":1,"2":2,"3":3,"4":4,"5":5,"6":6,"7":7,"8":8,"9":9,"10":10,"11
":11,"12":12,"13":13,"14":14,"15":15,"16":16,"17":17,"18":18,"19":19,"20":20,"21":21
,"22":22,"X":23,"Y":24}

variants_filename="D:\\Users\\doluc\Downloads\\GDC upstream_5UTR_3UTR
mutations.2018-11-04.json"
```

```
variants_file=open(variants_filename,"r")

undesired_consequences={"missense_variant","frameshift_variant","splice_region_variant","synonymous_variant","inframe_variant","stop_retained_variant","incomplete_terminal_codon_variant","stop_lost","incomplete_terminal_codon_variant","protein_altering_variant","inframe_insertion","inframe_deletion","stop_gained","start_lost","coding_sequence_variant","splice_acceptor_variant","splice_donor_variant"}
desired_consequencens=[]
unpaired_square_bracket=0
unpaired_curly_bracket=0
nextline=variants_file.readline()
line_no=0

interested_variants=[]
class variant:
    def __init__(self, keep=False,chr="",position=-
1,mutation="",ssm_id="",mutation_type=""):
        self.keep=keep
        self.chr=chr
        self.position=position
        self.mutation=mutation
        self.ssm_id=ssm_id
        self.mutation_type=mutation_type

    def __str__(self):
        return str(str(self.chr)+","+str(self.position)+","+str(self.mutation)
+","+str(self.ssm_id)+","+str(self.mutation_type)
+","+ChromosomeNo_dict[self.chr[3:]].__str__())

in_variant=False
variant_of_focus=variant()

while (nextline!=""):
    line_no+=1
    if line_no.__mod__(1000000)==0: print(line_no,)
    line=nextline

    if line=='  "consequence": [\n':
        in_variant=True
        # print("in")
        variant_of_focus=variant(keep=True)

    if line.startswith('        "consequence_type":'):
        consequence_type=line[29:-2]
        if consequence_type in undesired_consequences:
            variant_of_focus.keep=False

    if line.startswith('  "genomic_dna_change": '):
        variant_of_focus.chr=re.search(r"(chr[0-9]+)|(chr[XY])",line).group()
        variant_of_focus.position = re.search(r"g.[0-9]+", line).group()[2:]
        # print(line)
        variant_of_focus.mutation = re.search(r"([>ATGCinsdel]{3,})",line).group()
        # print(variant_of_focus)

    if line.startswith('  "ssm_id": '):
        variant_of_focus.ssm_id=line[13:-4]
        # print(variant_of_focus)

    if line.startswith('  "mutation_subtype":'):
        variant_of_focus.mutation_type=line[23:-2]
        # print(variant_of_focus)

    if line=='},{\n' or line=='}]\n':
```

```
        # print(variant_of_focus)
        in_variant=False
        if variant_of_focus.keep:
            # print(variant_of_focus)
            interested_variants.append(variant_of_focus)
        # print("out")

    nextline=variants_file.readline()

target_csv=open("target_variants.csv","w")
target_csv.write("sep=,\nchromosome,position,mutation,ssm_id,mutation type,chrNo\n")

for entry in interested_variants:
    target_csv.write(str(entry))
    target_csv.write("\n")
    print(entry)
```

Ek . G-dörtlüleri ile çakışan TCGA varyantları keşif kod betiği

```
import os
import re

variants_file=open("target_variants.csv","r")
G4List_file=open("..\\Homo sapiens, GRCh38.p7 Primary Assembly.G4List","r")
Chromosome_dict={1:"1",2:"2",3:"3",4:"4",5:"5",6:"6",7:"7",8:"8",9:"9",10:"10",11:"1
1",12:"12",13:"13",14:"14",15:"15",16:"16",17:"17",18:"18",19:"19",20:"20",21:"21",2
2:"22",23:"X",24:"Y"}
ChromosomeNo_dict={"1":1,"2":2,"3":3,"4":4,"5":5,"6":6,"7":7,"8":8,"9":9,"10":10,"11
":11,"12":12,"13":13,"14":14,"15":15,"16":16,"17":17,"18":18,"19":19,"20":20,"21":21
,"22":22,"X":23,"Y":24}

CurrentChr="chr1"
CurrentVariantLineNo=1
Results=[]
variant_line_no=0

def chrToChromosomeNo(chr=str):
    chromosome=ChromosomeNo_dict[chr[3:]]
    return chromosome

def definitionToChrNo(definition=str):
    return ChromosomeNo_dict[re.search(r"(chromosome )[0-9XY]
{1,2}",definition).group(0)[11:]]
variant_line_no+=1
print(variants_file.readline())
variant_line = variants_file.readline()
variant_line_no+=1

for G4_line in G4List_file:
    G4_line_parsed=G4_line.strip("\n").split("\t")
    if G4_line_parsed.__len__()<3:
        continue
    Current_G4_chr=definitionToChrNo(G4_line_parsed[0])
    Current_G4_begin=int(G4_line_parsed[1])
    Current_G4_end=int(G4_line_parsed[2])
    NoMoreForThisG4=False
    while(True):
        variant_line_parsed = variant_line.strip("\n").split(";")
        if variant_line_parsed.__len__()<3:
            break
        Current_variant_chr = chrToChromosomeNo(variant_line_parsed[0])
```

```
        Current_variant_pos = int(variant_line_parsed[1])
        Current_variant_mutation= variant_line_parsed[2]
        Current_variant_ssm_id= variant_line_parsed[3]
        Current_variant_mutation_type = variant_line_parsed[4]

        if (Current_variant_chr<Current_G4_chr):
            variant_line = variants_file.readline()
            variant_line_no += 1
            continue
        if (Current_variant_chr>Current_G4_chr):
            break
        if (Current_variant_pos<Current_G4_begin+1):
            variant_line = variants_file.readline()
            # if variant_line=="":print(Current_G4_chr,Current_variant_chr)
            variant_line_no += 1
            continue
        elif (Current_variant_pos>Current_G4_end-1):
            NoMoreForThisG4=True
            break
        else:            Results.append(",".join(G4_line_parsed)
+","+",".join(variant_line_parsed))
            print(Results[-1])
            variant_line = variants_file.readline()
            variant_line_no += 1
            continue

results_file=open("results.csv","w")
results_file.write("sep=,\n")
for entry in Results:
    results_file.write(entry+"\n")
exit()
```

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Dr. Öğr. Üyesi OSMAN DOLUCA
Proje No:	216S854
Proje Başlığı:	İnsan Genomunda Bulunan G-Dörtlülerinin Sınıflandırılarak Korunmuş Motiflerin Keşfi Ve Varyantlarının Analizi.
Proje Türü:	3001 - Başlangıç AR-GE
Proje Süresi:	16
Araştırmacılar:	OKTAY İSMAİL KAPLAN
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	İZMİR EKONOMİ Ü. MÜHENDİSLİK F.
Projenin Başlangıç ve Bitiş Tarihleri:	15/09/2017 - 15/03/2019
Onaylanan Bütçe:	92288.0
Harcanan Bütçe:	52043.61
Öz:	DNA motifileri proteinlerin amino asit dizilerinin saklanmasının yanı sıra genlerin regülasyonunda görev alan kısa DNA dizileridir. Bu diziler transkripsiyon faktörleri ya da diğer enzimlerle etkileşim göstererek gen regülasyonunda rol alabilmektedirler. Bu motiflerden bazıları DNA?nın bilindik 2 zincirli B-formun haricindeki sıradışı formlarını alarak bu etkileşimi tetikleyebilmektedirler. Bu sıradışı formlardan biri de G-dörtlüsü olarak bilinen, guanince zengin DNA yada RNA dizilerince oluşturulabilen yapılardır. G-dörtlüleri kendi aralarında birçok yapısal farklılık gösterebilen ve bu nedenle farklı rollere sahip bir yapı grubudur. Bu yapılar özellikle promotor bölgesi gibi regülasyonda etkili bölgelerde oluşarak transkripsiyonu etkileyebilmektedirler. Ancak bu yapıların tespiti ve rolleri konusunda çalışmalar devam etmekle beraber insan genomunda tahmin edilen 700.000?in üzerindeki G-dörtlüsünün rollerinin ve analizlerinin yapılması zaman ve emek alan bir süreçtir. Bu nedenle biyoenformatik yöntemler ile ön analiz ve tahminler yapılması ilgi çekmekte, varsa bu yapılardaki patojenik etkilerinin ortaya çıkarılması için tıbbi anlamda değer taşımaktadır. Bu çalışmada TÜBİTAK desteği ile G-dörtlülerinin tahmininden, sınıflandırılması ve ilişkilendirilmiş patojenik varyasyonların tespitine kadar gerekli bir seri biyoenformatik teknikler geliştirilmiş ve insan genomu için uygulanmıştır. İlk aşamada G-dörtlüsü tahmini için yeni bir algoritma geliştirilmiş ve başarısı yayınlanan makale ve bildiriler ile gösterilmiştir. Ardından insan genomunda tahmin edilen G-dörtlüleri bulunmuş, sınıflandırılarak farklı G-dörtlüsü motifleri ortaya çıkarılmıştır. Bulunan G-dörtlülerin patojenik etkisi bilinen varyasyonlar ile eşleştirilmiş ve bu varyasyonların etkisinin G-dörtlüsü yapısını etkileyerek gerçekleşip gerçekleşmediğinin tartışılabilmesi için yapısal değişimler deneysel olarak araştırılmıştır. Bulunan sonuçlar gerçekten bazı G-dörtlülerinin patojenik varyasyonlar sonucu yapısal olarak değişebildiğini göstermiştir.
Anahtar Kelimeler:	G-dörtlüsü, biyoenformatik, sekans analizi, varyant analizi, genetik hastalıklar
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Hayır
Projeden Yapılan Yayınlar:	1- G4Catchall: A G-quadruplex prediction approach considering atypical features (Makale - İndeksli Makale), 2- G-quadruplexes prediction using G4Catchall (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 3- Relaxed formation rules for G-quadruplex prediction (Bildiri - Ulusal Bildiri - Sözlü Sunum),