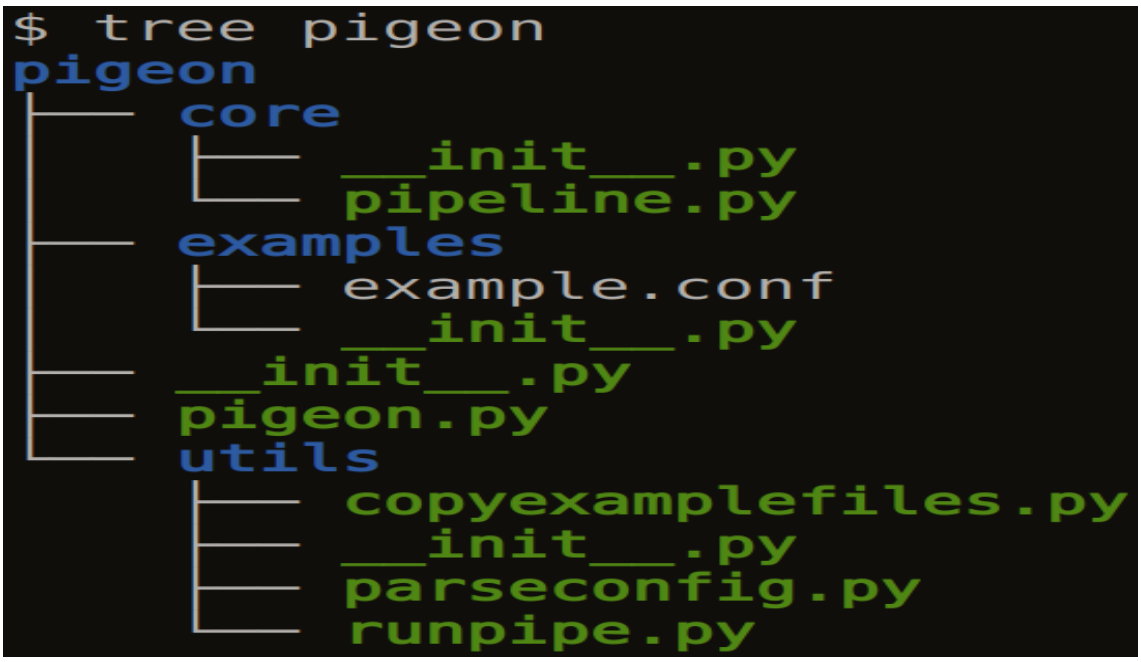
3.2.2. Akış Hattı Yazılımının Programlanması

Akış hattı yazılımı, hazırlanan akış hattı yapılandırma dosyasını okuyarak, bu dosya içerisinde belirtilen tanımlamalara uygun şekilde işlemleri gerçekleştirmek şekilde planlanmıştır. İşlemlerin argümanları ve parametreleri ve çalışma sıraları yapılandırma dosyası üzerinden verilir.

3.2.2.1. Akış Hattı Yazılımının Yapısı

Yazılım, pigeon.py, pipeline.py, parseconfig.py, ve runpipe.py olmak üzere dört temel dosyaya ayrılmıştır (Şekil 3-10).



Şekil 3-10: Akış hattı yazılımının dosya sistemi üzerindeki yapısını gösteren tree komutu çıktısı.

pigeon.py İşlevi

Bu dosya, kullanıcının komut satırı ile programa erişimini sağlar. Kullanıcının verdiği argümanlar burada ayrıştırılarak uygun işlemler çağrılır.

parseconfig.py İşlevi

Bu dosya, yazılıma sağlanan yapılandırma dosyası içerisinde geçen argümanları ayrıştırarak kullanıma hazır hale getirir.

pipeline.py İşlevi

Bu dosya, yazılımın temeli olup sağlanan argümanları bir araya getirerek akış hattında kullanılacak komutları oluşturur.

runpipe.py İşlevi

Bu dosya, akış hattı çalıştırıldığında pipeline.py tarafından oluşturulan komutları kabuğa(shell) geçirerek çalıştırılmasını sağlar.

3.2.2.2. Akış Hattı Yapılandırma Dosya Yapısı

Yapılandırma dosyalarının ayrıştırılması python configparser modülü ile gerçekleştirilir ve dosya yapısı bu modülün dökümantasyonuna uygun olarak geliştirilmiştir. Bu formata ek olarak okunabilirliği arttırmak üzere argümanlar içerisine yeni satır eklenebilir.

Yapılandırma dosyası, işlenecek dosyaların, sonuçların yazılacağı klasörün, çalışacak yazılımların sırasının, her yazılımın kendine özgü argümanlarının ve analizlerde kullanılacak ek verilerin tanımlanabileceği düz metin dosyasıdır. Bu dosya kendi içerisinde, PROJECT, GENERAL, INCLUDE, PIPELINE isimli dört tanımlı bölüm ve kullanıcının her yazılım için ekleyebileceği bölümlerden meydana gelir. Her bölüm ismi, [PROJECT] gibi köşeli parantezler içerisinde belirtilir. Kullanıcı yorumları satıra “#” karakteri ile başlayarak eklenebilir.

PROJECT Bölümü

Bu bölüm içerisinde çalışmalar arası değişebilecek argümanlar tanımlanır (Tablo 3-10). İsteğe bağlı project_name değişkeninin olmaması durumunda çalıştırma tarih ve zamanı proje ismi olarak output_dir değişkeninin olmaması durumunda ise komutun çalıştırıldığı klasör çıktıların yazılacağı klasör olarak kullanılır.

Tablo 3-10: Projeye özgü argümanlar bölümü.

Argüman	Gereklilik	İşlevi
project_name	İsteğe bağlı	Çalışmaya özgü isim
output_dir	İsteğe bağlı	Çalışma sırasında çıktı dosyalarının yazılacağı klasör
input_files	Zorunlu	Akış hattında analizin yapılacağı girdi dosyaları
input_names	İsteğe bağlı	Girdi dosyalarının isimleri

GENERAL Bölümü

Bu bölüm içerisinde çalışmalar arası değişmeyecek argümanlar tanımlanır. Burada referans genom, popülasyon varyasyon verisi ya da kullanılan yazılımların gerektirdiği herhangi bir dosya ya da linke bağlantı verilebilir. Verilen dosyaların, dosya sistemi üzerindeki tam yolları verilmelidir. Bu kısımda tanımlanan değişkenler daha sonra işlem bölümlerinde, ${GENERAL:değişken} formatı ile kullanılabilir.

INCLUDE Bölümü

Bu bölüm içerisinde diğer akış hattı yapılandırma dosyaları eklenebilir. Böylece birden fazla akış hattında kullanılan ya da tekrarlı kısımlar modüllere çevrilerek kullanılabilir. Bu kısımda tanımlanan değişkenler daha sonra [INCLUDE:değişken] formatında bölüm olarak kullanılabilir.

PIPELINE Bölümü

Bu bölüm içerisinde tanımlanan her işlemin çalıştırma sırası pipeline argümanı içerisine yazılır. Bununla beraber işlemlerin kabuk üzerinde çalıştıracak yazılım isimleri ve konteynerler da bu kısımda isteğe bağlı olarak tanımlanabilir. Bu kısımda tanımlanan

yazılım isim değişkenleri işlem kısmının tool argümanı olarak, konteyner değişkenleri de işlem kısımlarında container argümanı olarak kullanılabilir.

İşlem Bölümü

İşlem bölümünde çalıştırılacak her işlemin girdisini, çıktısını, işlem argüman ve parametrelerini işleme özel olarak verilir (Tablo 3-11). Bu bölümlerin isimlendirmesi kullanıcı tarafından yapılır.

Tablo 3-11: İşleme özgü argümanlar bölümü.

Argüman	Gereklilik	İşlevi
tool	Zorunlu	Çalışacak yazılımın kabuğun tanıdığı ismi
input_from	Zorunlu	Çalışacak işlemin girdisini nereden alacağı
args	Zorunlu	İşlemin argüman ve parametreleri
dump_dir	İsteğe bağlı	İşlem çıktısının yazılacağı klasör
ext	İsteğe bağlı	İşlem çıktı dosyalarının uzantısı
suffix	İsteğe bağlı	İşlem çıktı dosya isimlerine ek
input_multi	İsteğe bağlı	İşlemin girdileri çoklu alması
container	İsteğe bağlı	İşlemin çalışacağı konteyner
sub_tool	İsteğe bağlı	İşlemin alt komutu
input_dir	İsteğe bağlı	İşleme bağıl girdi klasörü
secondary_input	İsteğe bağlı	İşleme ikinci çeşit girdi belirteci
secondary_in_dir	İsteğe bağlı	İşleme bağıl ikincil girdinin klasörü
input_flag_repeat	İsteğe bağlı	İşlemin girdi argümanını her girdi için tekrarlama
output_dir	İsteğe bağlı	İşleme bağıl çıktı klasörü
remove	İsteğe bağlı	İşlemden sonra çıktı dosyalarını kaldır
secondary_output	İsteğe bağlı	İkincil çıktı çeşidi belirteci
secondary_dump_dir	İsteğe bağlı	İkincil çıktıların yazılacağı klasör
secondary_ext	İsteğe bağlı	İkincil çıktı dosya uzantısı
secondary_suffix	İsteğe bağlı	İkincil çıktı dosya isimlerine ek

İşlemlerin kullanacağı girdi ve çıktı dosyalarının isimlendirmesi otamatik olarak gerçekleştirilir ve işlemlere tanıtılması her işlem bölümündeki args isimli argümanın içinde yapılır (Tablo 3-12).

Tablo 3-12: Girdi ve çıktıların yerini tuıtan özel değişkenler.

Özel Değişken	Gereklilik	İşlevi
input_placeholder	Zorunlu	İşlemin girdi dosyasının yerini turar
output_placeholder	İsteğe bağlı	İşlemin çıktı dosyasının yerini tutar.
secondary_in_placeholder	İsteğe bağlı	İşlemin ikincil girdi dosyasının yerini tutar.
secondary_out_placeholder	İsteğe bağlı	İşlemin ikincil çıktı dosyasının yerini tutar.

3.2.2.3. Akış Hattı Yazılımının Çalıştırılması

Yazılımın çalıştırılması için ilk olarak yapılandırma dosyasının hazırlanması gereklidir. Örnek olarak Bölüm 3.2.2.2'ye uygun olarak Fastq dosyalarını fastqc yazılımı ile kalite kontrolü ve bwa yazılımı ile genoma hizalama yapacak bir yapılandırma dosyası Şekil 3-11'de verilmiştir.

```
example.conf
[PROJECT]
project_name = example
output_dir = /home/bar/example/output
input_files =   /home/bar/example/input/A_1.fastq
                /home/bar/example/input/A_2.fastq
                /home/bar/example/input/B_1.fastq
                /home/bar/example/input/B_2.fastq
input_names = A B

[GENERAL]
reference_genome = /home/bar/references/hg19.fa

[PIPELINE]
pipeline = QUALITYCHECK ALIGNMENT
bwa = bwa
fastqc = fastqc

[QUALITYCHECK]
tool = ${PIPELINE:fastqc}
input_from = input_files
args = input_placeholder
       --outdir=${PROJECT:output_dir}/fastqc
       --extract
dump_dir = fastqc

[ALIGNMENT]
tool = ${PIPELINE:bwa}
sub_tool = mem
input_from = input_files
args =  -t 8
       ${GENERAL:reference_genome}
       input_placeholder > output_placeholder
named = True
suffix = _bwa
ext = sam
input_multi = paired
```

Şekil 3-11: Kalite kontrol ve genoma hizalama akış hattı yapılandırma dosyası.

Yapılandırma dosyası hazırlandıktan sonra komut satırında "pigeon" komutuyla yazılım çalıştırılabilir. Yazılıma "-c" argümanı ile hazırlanan yapılandırma dosyası tanıtılabilir ve analize başlamadan önce "-d" argümanı ile kabuğa gönderilecek komutlar prova edilebilir (Şekil 3-12). Herhangi bir problem yoksa "-d" argümanı kaldırılarak analize başlanabilir.

```
$ pigeon -c example.conf -d

fastqc /home/bar/example/input/A_1.fastq --outdir=/home/bar/example/output/fastqc --extract

fastqc /home/bar/example/input/A_2.fastq --outdir=/home/bar/example/output/fastqc --extract

fastqc /home/bar/example/input/B_1.fastq --outdir=/home/bar/example/output/fastqc --extract

fastqc /home/bar/example/input/B_2.fastq --outdir=/home/bar/example/output/fastqc --extract

bwa mem -t 8 /home/bar/references/hg19.fa /home/bar/example/input/A_1.fastq /home/bar/example/in
put/A_2.fastq > /home/bar/example/output/A_1_bwa.sam

bwa mem -t 8 /home/bar/references/hg19.fa /home/bar/example/input/B_1.fastq /home/bar/example/in
put/B_2.fastq > /home/bar/example/output/B_1_bwa.sam
```

Şekil 3-12: Akış hattının prova çalıştırılması sonucu.

Akış hattı yazılımının alabileceği bütün argümanlar “pigeon -h” komutu ile komut satırı üzerinde görülebilir.

3.2.3. Vcf Sonrası Analizler Yazılımının Programlanması

Vcf sonrası analizler yazılımında varyant çağırma adımından sonra elde edilen varyantların, anotasyonu, esnek bir şekilde filtrelenebilmesi ve ailesel çalışmalarda aile içi karşılaştırmaların yapılabilmesi hedeflenmiştir.

Varyant çağırma sonrasında elde edilen dosya varyantın genomik pozisyonu, değişimi ve allellerin okuma derinliklerini içerir. Sadece bu bilgilerden yola çıkılarak bir varyantın anlamlandırılması mümkün değildir. Bu yüzden elde edilen varyantların, farklı genetik işaretçilere bağıl pozisyonu, popülasyonlardaki görülme sıklığı, RNA ve protein aşamalarında neden olabileceği bozukluklar gibi bilgilerin anotasyon aşaması ile eklenmesi gerekmektedir. Böylece varyantın fenotip üzerinde yaratabileceği etkiler, genomik, transkriptomik ve proteomik bağlamda popülasyon bilgisi ile birlikte irdelenebilir.

Varyantların anotasyonu sonrasında elde edilen çok sayıdaki varyantın oluşturduğu büyük verinin içerisinden ilgilenilen fenotip ile ilişkili olabilecek varyantları ayrıştırmak gereklidir. Bu ayrıştırma varyantların belli kriterler üzerinden elenmesi ile sağlanır.