‘$’ ile baslayanlar bash komutları.
‘#’ ile baslayanlar yorum ve açıklamalar.

1) GIRIS

bilgisayarımızda hangi shell var?
$ echo $SHELL

$ pwd

$ cd /directory1
$ cd directory1/directory2/directory3

$ cd ..
# üst konum
$cd ../../..

$ ls
$ ls -t
$ ls -l
$ ls -1
$ ls -l -1
$ ls -lt1
# sıralama onemli degil

$ man ls
# koseli parantez icindekiler opsiyonel

$ mkdir isim_soyad
# dosyaya girerken TAB’e basma aliskanligi.

#brace expansion kullanmak
$ echo sekans_{ornek1,ornek2,ornek3}
# ne oldu?

yukarı ok tusunun faydası

mkdir -p zmays_snps/-{data,seqs,scripts,analysis}

 # silmek
$ touch test1 test2
$ ls

$ rm touch1 touch2
# silerken onay istemeyebilir, bu nedenle cok tehlikeli bir komut.
# rm ile bir seyi silerseniz geri gelmez!
# tum dosyalar silebilirsiniz!

#bin/ folder's icini silebilirsiniz

$ touch test
$ rm -i test
# -i parametresi ile sildiğinizde kesinlikle onay ister.

Ctrl + a basa git
Ctrl + e sona
Ctrl + w komuttaki bir onceki kelimeyi sil

# hangi programlar calisiyor, ne kadar CPU kullanıyorlar, 'top' komutu ile öğrenebilirsiniz.
$ top

# devam eden islemler neler:
$ jobs

dosya adlandırma mantigi:
Yanlis: genes_1.txt, genes_2.txt,…,genes_11.txt - numeric sıralamayı bilgisayar algılayamaz bu sekilde numaralandirirsaniz.

#dogru adlandirma:
genes_001.txt, genes_002.txt,…genes_011.txt

2) UNIX DATA ARACLARI'na giris:

SIMSDI ISIMIZE YARAYACAK ASIL KONUYA GIRIYORUZ, unix-bash ile text datalari (sekans datalari da bir besitt text datasi zaten) ile calismak,

cat, grep, cut, awk gibi programlari/komutlari kullanarak data analizinde ilk adimi atacagiz!

Mantik: sekans datasi ustune dusunecegiz (fakat big data analizi icin de kullanabilirsiniz bu araçları). Gb boyutunda, satırlarca text datası. Data'yi acmak, manipüle etmek, memory işgal etmek olasılık disi. Read-only analiz gerekiyor. Linux, bash read-only data analizi icin birebir bilgisayar islemleri ortamı.

CAT:

cat, memory'ye almadan "stream" olarak bash ekranında dosyayı görüntüler

#tga1-protein.fasta dosyasına bakalım:
$ cat tga1-protein.fasta

catkin nasıl calistirildigini öğrenmek icin manual'ine bakalım:

$ man cat

#Birden fazla dosyaya tek komut ile islem uygulayabilirsiniz:
$ cat tb1-protein.fasta tga1-protein.fasta

# * wild card'ini kullanarak deneyelim?
$ cat *.fasta
# .fasta ortak ismine sahip tum dosyalar '*' wild card'ini kullanarak goruntuledik.

Input'u output'a yönlendirmek (yeni bir dosya olarak kaydetmek):
'>' veya '>>' kullanarak yonlendirmek:

'>' girdileri yeni bir dosyaya yonlendirip/kaydeder - eğer ayni isimli baska bir dosya var ise, sormadan uzerine yazar, dikkat!

'>>' yine yonlendiriyor fakat eger o isimde bir dosya varsa, ilgili dosyanın sonuna ekler yeni veriyi.

#CAT programi ile append edebilirsiniz.
$ cat tb1-protein.fasta tga1-protein.fasta > zea_proteins.fasta
$ cat zea_proteins.fasta

HEAD & TAIL

$ man head

$ head Mus_musculus.GRCm38.75_chr1.bed

$ man tail

$ tail Mus_musculus.GRCm38.75_chr1.bed

$ head -17 Mus_musculus.GRCm38.75_chr1.bed
# ilk 17 satiri gösteriyor

$ tail -17 Mus_musculus.GRCm38.75_chr1.bed

$ (head -n 2; tail -n 2) < Mus_musculus.GRCm38.75_chr1.bed

$ head yeast_chr1_orfs.fa.txt

$ tail yeast_chr1_orfs.fa.txt

BED & GTF dosyalari farkli kaliplarda dosyalardir:

$ head -15 Mus_musculus.GRCm38.75_chr1.gtf

LESS

cat ekrandan stream olarak dosya icerigini okutuyordu. Akip-gitmeden bakmak istersek, an basit kullanabileceğimiz program ‘less’. Cok büyük text dosyalarını / genom datası dosyalarni hiç sorun olmadan acabilir (ornegin word ile devasa text dosyalarını acmaya calisirsaniz bilgisayar zorlanır). ‘Less’in hızlı olmasının mantigi memory’ye almadan okuyabilmesi.

$ man less
# yukarı asabi nasıl hareket edebileceğinizi manual’den ogrenin.

$ less contaminated.fastq
# cikis icin 'q'ya basiniz.

#daha büyük bir dosyaya bakalım:
$ less celegans.sam

WC

The WC utility displays the number of lines, words, and bytes contained in each input file, or standard input (if no file is specified) to the standard output.

$ wc Mus_musculus.GRCm38.75_chr1.bed
# 3 adet bilgi verdi. Neler?

$ wc mm_GRCm38.75_protein_coding_genes.gtf

# genelde ilgili datanin/dosyanin kaç satir olduğu ile ilgiliyiz:
$ wc -l Mus_musculus.GRCm38.75_chr1.bed
# uyari: wc empty lines larida sayiliyor.

CUT - SUTUN VERISI ILE CALISMAK
cut’in tab-delimited dosya secimi “by default”

# Once cat ile dosyaya bakalım:
$ cat Mus_musculus.GRCm38.75_chr1.bed

# 2. sutunun icerigine ulaşmak:
$ cut -f 2 Mus_musculus.GRCm38.75_chr1.bed

$ cut -f 2 Mus_musculus.GRCm38.75_chr1.bed | head -14

-f argument birden fazla sutuna ulaşmak icin de kullanilabilir. 3, 4, 5, 6, 7, 8. sutunlarin

hepsine ulaşmak icin -f 3-8 argümanı yeterlidir.

# 1'den 3'e:
$ cut -f1-3 Mus_musculus.GRCm38.75_chr1.bed | head -14
$ cut -f2-3 Mus_musculus.GRCm38.75_chr1.bed | head -14

GREP

grep'in kullanım mantigi: aradiginiz bir 'kalif' (pattern) ister (regular expression olabilir, harf ve rakamları içeren bir kalıp olabilir), bir de kalıbı icinde arayacaginiz datayı ister.

$ man grep

$ grep Man heroes_and_villains.txt
# case sensitive!

$ grep -i Man heroes_and_villains.txt
# 'i' secenegi: case-insensitive arama
# ilgili pattern'i buluyor, tum satiri ekrana veriyor.

$ grep -i -v Man heroes_and_villains.txt
# 'v' aramanin mantigini zit yone ceviriyor. Yani 'Man'i "ignore" ediyor bu komutta.

$ grep -w Man heroes_and_villains.txt
# 'w' sadece tek basina "Man"i iceren satirlari buluyor.

#grep ile bir kalibin kac kere tekrarlandigini bulabilirsiniz:
$ grep -c Villain heroes_and_villains.txt
# grep -c: grep count.

# peki grep ne ise yarayabilir data analizinde:
$ grep -n -i CGTATAT yeast_chr1_orfs.fa.txt
# 7 nukleotidlik kalibin bulunduğu satırları buldu.
#Bu dogru cevap olmayabilir! Dikkat! 'grep', eger sekans iki farkli satira yayilmissa, aranan diziyi bulamaz!

# calistigimiz bir gene dair özellikleri grep ile bulalım:
$ grep 'gene_id "ENSMUSG00000025907"' Mus_musculus.GRCm38.75_chr1.gtf | head -n 1

$ grep "Olfr418-ps1" Mus_musculus.GRCm38.75_chr1_genes.txt

# "partial match" mantigi ile calisiyor:
$ grep Olfr Mus_musculus.GRCm38.75_chr1_genes.txt | head -n 5

# SORU: “Olfr1413” disinda tum "Olfr" genlerine ihtiyaciniz varsa ne yapacaksiniz?
# pipeline oluşturarak, ilk kısımda bir islem yapıyoruz, output’u ‘|’ işaretinden sonra ikinci bir programa input olarak veriyoruz:
$ grep Olfr Mus_musculus.GRCm38.75_chr1_genes.txt | grep -v Olfr1413
#Emin miyiz? Calisti mi? Yazdiginiz komut ve kodların calisip calismadiginin SAGLAMASI her seferinde yapılmalı. Cogu zaman şikayet etmeden komut bir output verebilir. “Garbage in garbage out” deyimi, bu riskli durumu anlatır. Kotu bir komut/kod yazıp, yanlış sonuclar almanız sürekli mumcun. Doğru islem mi yaptirdiniz, sürekli test etmek lazım. Sağlamayı nasıl wc -l ile yaptırabilirsin? Dusun…

# Cozum:
$ grep Olfr Mus_musculus.GRCm38.75_chr1_genes.txt | grep -v Olfr1413 | wc -l
$ grep Olfr Mus_musculus.GRCm38.75_chr1_genes.txt | wc -l

Onemli uyari: bu tip arama islemlerinizi yaparken kuracaginiz mantik minimum hedeflenmeyen kalip bulmaya
gore onceden dusunulerek yapilmali.

grep’in diger ise yarayan bir özelliği, ilgili kalıbın (pattern) önünde (-B), sonrasında (-A), veya hem önünde hem arkasında bulunan daha fazla kismini da gösterebilir.

$ grep -B1 "AGATCGG" contam.fastq | head -n 6
# “Print one line of context before (-B) the matching line.”

GREP cos kuvvetli bir unix data aracidir. Cok hızlı. Bir cok regular expression arayıp bulan programdan daha hızlı. Sebebi de ilgili kalıbı satırda bulunca, satirin gerisini aramadan (daha fazla zaman kaybetmeden) ilgili satiri ciktiya verir.

$ grep -A2 "AGATCGG" contam.fastq | head -n 6

#“Olfr218” veya “Olfr1416.” genlerini aramak icin:
$ grep -E "(Olfr1413|Olfr1411)" Mus_musculus.GRCm38.75_chr1_genes.txt
$ man grep
# -E opsiyonunun ne ise yaradigini okuyun.

$ grep "Olfr141[13]" Mus_musculus.GRCm38.75_chr1_genes.txt

#count secenegi:
$ grep -c "\tOlfr" Mus_musculus.GRCm38.75_chr1_genes.txt
#Ilgili datada Olfr gen ismi kalibindan kaç adet oldugunu bulduk.

Mus_musculus kromozom 1’in özellikleri bilgisini içeren .gtf dosyasının ilk basında bulunan yorum (# ile baslayan) satırları eleyerek, asil ilgilengimiz bilginin olduğu ilk 10 satira bakalım:

$ grep -v "^#" Mus_musculus.GRCm38.75_chr1.gtf | cut -f1-8 | head

# bash ekraninda biraz daha düzenli görmek isterseniz ‘column -t’ programından yardim alabilirsiniz:
$ grep -v "^#" Mus_musculus.GRCm38.75_chr1.gtf | cut -f 1-8 | column -t | head -n 3

CUT + GREP (birden çok programi, pipeline ile calistirmak):

comma separated bir dosyaya (.csv uzantılı) yine column -t ile daha düzenli bakmak mumcun, fakat -s”,” parametresi ile seperator’un ‘,’ (virgül) oldugunu programa bildirmemiz gerekiyor:

# once column -t kullanmadan bakalim.
$ head Mus_musculus.GRCm38.75_chr1_bed.csv

# simdi column -t kullanarak bakalim.
$ column -s"," -t Mus_musculus.GRCm38.75_chr1_bed.csv | head -n 3

— —

Egzersiz: CUT’i kullanarak Mus musculus kromozom 1 özelliklerini içeren .gtf dosyasını 3 sutunun bilgisini içeren, tab-ile-ayrilmis (tab-delimited) ozelliklerin chromosome, start, & end position’i veren daha sade bir dosyaya dönüştürelim:

#problemi adim adim düşünürseniz kolayca çözebilirsiniz:
- adim 1: en basta ‘#’ ile baslayan metadata’dan kurtulalim.
- (chromosome, start, end) bilgileri 1., 4., 5. sutunlarda. CUT kullanarak ayikliyalim.

Cozum:
# öncelikle # içeren satırlardan kurtulalım demistik. # içeren satırlar derken:
$ head "^#" Mus_musculus.GRCm38.75_chr1.gtf | column -t

$ grep -v "^#" Mus_musculus.GRCm38.75_chr1.gtf | cut -f1,4,5 | head -n 3

-v’nin (—invert-match, Selected lines are those not matching any of the specified patterns.) istediğimiz gibi calistigina emin olduktan sonra yeni dosyaya kaydedebiliriz:

$ grep -v "^#" Mus_musculus.GRCm38.75_chr1.gtf | cut -f1,4,5 > test.txt
$ ls -ltr
$ cat test.txt

Mus musculus gif özellikler dosyasında small nuclear RNA sayisini bulmak istiyoruz. snRNA özellikleri nasıl kayıtlı ilgili .gtf dosyasında: gene_biotype "snRNA" - buradan yola cikip bu özellikleri sayalim.

$ grep -c 'gene_biotype "snRNA"' Mus_musculus.GRCm38.75_chr1.gtf

UYARI: ' ' ve " " ayrimina dikkat!

# pseudogene bayisini bulalim:
$ grep -c 'pseudogene' Mus_musculus.GRCm38.75_chr1.gtf

grep hızlı çünkü satırda match bulunca satirin gerisine bakmadan tum satiri veriyor. Sadece istediğiniz kismi vermesi icin ‘-o’ seçeneği kullanılır:

$ grep -o "Olfr.*" Mus_musculus.GRCm38.75_chr1_genes.txt | head -n 3

SORT
#sort alfa-numerik calisir.

$ cat example.bed
# sutunlarin nasıl siralandigini gordunuz.

sort yaparken farklı isteklerimiz var:
- belli bir sutuna gore sort etmek
- sort’a belli sutunlarin “sayısal” oldugunu söylemek.
- birden cok sutunu dikkate alarak sort etmek.

example.bed’i kromozom (sutun 1) & kromozom uzerinde baslangic konumuna (sutun 2) gore sıralamak istediğimizi düşünelim :

$ sort -k1,1 -k2,2n example.bed

n numerical sort
-k1,1 : start,start

$ sort -k1,1 -k2,2n example.bed > example_sorted.bed
# sort edilmiş dosyaya yeni bir dosyaya kaydedelim.

Ornek: daginik bir .gtf dosyasini once kromozom sonra da pozisyona gore siralayalim:
$ sort -k1,1 -k4,4n Mus_musculus.GRCm38.75_chr1_random.gtf > Mus_musculus.GRCm38.75_chr1_sorted.gtf

$ sort -k1,1 -k2,2n example2.bed
# sort computer tabanla sıralama yapıyor, chr2 vs chr11

UNIQ -c

```
 Uniq    -c      Precede each output line with the count of the
number of times the line occurred in the input, followed by a
```

```
single space.
```

# lac ozgun ozellik var, letters.txt dosyasında
$ uniq -c letters.txt
# neden olmadı? Sort etmediğimiz icin.
$ sort letters.txt | uniq -c

# mükemmel ornek. Mus musculus chr1'de bulunan genetik özelliklerin sayısı:
$ grep -v "^#" Mus_musculus.GRCm38.75_chr1.gtf | cut -f3 | sort | uniq -c
# ayni isi çoktan aza doğrular sıralarsak:
$ grep -v "^#" Mus_musculus.GRCm38.75_chr1.gtf | cut -f3 | sort | uniq -c | \
sort -rn

# bir gene ait olan özellikler:
$ grep "ENSMUSG00000033793" Mus_musculus.GRCm38.75_chr1.gtf | cut -f3 | sort \
| uniq -c

# bir baska gene ait özelliklere bakmak:
$ grep "Lypla1" Mus_musculus.GRCm38.75_chr1.gtf | cut -f 3 | sort | uniq -c
# cut, sort,uniq sirasinin mantigi onemli.

AWK
awk data analiz ettigimiz basit bir komut aracı degil ayni zamanda kompleks bir programlama dilidir. Biz sadece komut aracı olarak kullanacağız.

# komut mantigi:
$ awk pattern { action }
# ilk bakışta garip geliyor. pattern disarida, action {} içerisinde.
# pattern TRUE ise, statement'in icindeki action islenir.
# eğer action vermezseniz, pattern'a uyan tum pattern print edilir.

$ awk '{ print $0 }' example.bed
# $0 tum sütunları print eder.

$ awk '{ print $2 }' example.bed
# ikinci satiri print eder (cut -f2 gibi)

$ awk '{ print $2 "\t" $3 }' example.bed
#cut f2,3 ye esdeger.
"\t" tab-delimited olarak print ediyor.

Bir data dosyasinda kac adet sutun var, öğrenmek istiyoruz:
$ tail -n +6 Mus_musculus.GRCm38.75_chr1.gtf | head
$ tail -n +6 Mus_musculus.GRCm38.75_chr1.gtf | awk -F "\t" '{print NF; exit}'

#NF: "number of fields" icin kullanilan bir awk ozelligi.
-F dosyanin tab-aralikli veya virgul-aralikli olduguna gore onemli bir parametre.

awk ile aritmetik islemler: +, -, *, /, %, ^
$ awk '$3 - $2 > 18' example.bed
# belli bir nükleotid uzunluğundan kisa/uzun genleri bulmak vb.

tek bir pattern degil, birçoklarıyla pattern'i birbirine ekleyerek mantıksal operasyonlar kullanarak daha spesifik aramalar/islemler yapabiliriz. Logical operators: && (AND), ll (OR), and !
(NOT).

kromozom 1'de satir uzunlugu 10'u asan satirlari bulmak:
$ awk '$1 ~ /chr1/ && $3 - $2 > 10' example.bed
# Bu komutta ilk sutunu ($1) ve (&&) chr1'i seciyoruz.
# regular expression'lar '/ /' arasına yaziliyor.
# ~ esles (match) anlamina geliyor

sadcece kromozom 2 ve 3 icin özelliklerin uzunluğunu bulmak:
$ awk '$1 ~ /chr2|chr3/ { print $0 "\t" $3 - $2 }' example.bed
# Ciktiya bakip islemi nereye verdigine dikkat edin.

# daha once, awk ve cut ile belirli sütunlar uzerinde calistik. Peki sadece belirli satir araliklarina ulaşmak istersek: orn. 3-5 (3, 4, 5) satırları istiyorsan:
$ awk 'NR >= 3 && NR <= 5' example.bed

AWK icin son orneklerin hepsi yeast_genome.gff data dosyası uzerinden yapilacak. yeast_genome.gff dosyasını suradan indirebilirsiniz:
https://downloads.yeastgenome.org/curation/chromosomal_feature/saccharomyces_cerevisiae.gff

# dataya bakin
$ less yeast_genome.gff

# input dosyasi icerigine awk ile bakmak:
$ awk {'print'} yeast_genome.gff

bir dosyanin ilk sutununa bakmak:
$ awk {'print $1'} yeast_genome.gff

once 5 sonra 1 no'lu sutunlara bakmak:
$ awk '{print $5, $1}' yeast_genome.gff

genomik ozelliklerin ilk-son koordinatlarini yazdirin - 10000 nukleotid'den sonrasinin:
$ awk '{if ($4 > 10000) print $4, $5}' yeast_genome.gff

genomik ozelliklerin ilk-son koordinatlarini ve uzunluklarini yazdirin:
$ awk ‘{print $4, $5, $5 – $4 + 1}’ yeast_genome.gff

aralik birakmadan yapabilirsiniz:
$ awk ‘{print $4,$5,$5-$4+1}’ yeast_genome.gff

# 'intron' iceren satirlari yazdirin:
$ awk ‘/intron/ {print}’ yeast_genome.gff

# 'print' konumu ortuk olarak kullanilabilir:
$ awk ‘/intron/’ yeast_genome.gff

ozelliklerin hangi satirlarda oldugunu 'NR'i kullanarak yazdirin (NR = number of records)
$ awk ‘{print NR, $3}’ yeast_genome.gff

# 'END' komutu ile sadece toplami yazdirin:
$ awk ‘END {print NR}’ yeast_genome.gff

UTR icinde bulunan tum intronlarin uzunluklarinin toplamini yazdirin:
$ awk ‘/UTR_intron/ {lengths += ($4-$3+1)} END {print “total length of UTR introns = “ lengths}’ yeast_genome.gff

———————————————

EXERCISE (zaman kalırsa):

COK FAYDALI, PRATIK BIR ORNEK:

Elinizde en son sekansladiginiz evrim deneyi sonucu elde ettiginiz mayalarin (S. cerevisiae)
kromozom IV'un sonunda bir bolge hic ORF icermiyor. Bu olagan bir durum mu yoksa bir tur
noncoding sekans yayilmasi (expansion) mi var? Database'de bulunan bir maya genomunu indirdiniz: 'yeast_genome.gff'.
16 kromozom var. Sadece kromozom IV'un sonunda genomik ozelliklere bakip karsilastiracaksiniz.
Ornegin son 20 genomik ozellik deneyinizde evrilen mayada oldugu gibi ORF iceriginden yoksun mu?

Her seferde bir command ile grep (1), sort (2), head -n 20 (3), cut (4), sort -tekrar- (5), uniq (6)
adimda 6 farkli input & output kullanarak yapabilirsiniz elbette. Ornegin 400 .gff file

uzerinde
yaptiginizi dunusunun...

Pipeline ile tum girdi-cikti kaydindan, hem de teker teker type etmekten kurtuluyorsunuz.

$ grep -E “^chrIV” yeast_genome.gff | sort -n -k 4 | head -n 20 | cut -f 3 | sort | uniq

Iki uc farkli icerige mi sahip?

$ grep -E “^chrIV” yeast_genome.gff | sort -r -n -k 4 | head -n 20 | cut -f 3 | sort | uniq