Course outline

Day #1: Basic command line interface and reproducible research

Day #2: Command line tools and sequence alignment, Intro to R and RStudio

Day #3: R, data manipulation, and plotting

Let’s play with data

1. Open terminal
2. Go to data folder in IBiS_Bootcamp
3. Look at directory contents
4. .gz is a g-zipped file. Type **`gunzip`** file to unzip it
5. Type “atom” to open a text editor for today’s work - File naming use date YYYYMMDD_file for clarity

```
data — bash — 93×22
~/Desktop/IBiS-Bootcamp/data head -20 DL238.chrIII.fq
@FCC1GWRACXX:6:1103:2609:96765#AGTCAGAA/2
TCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCCTGTCTCTTATACACATCTCTGATGGCGCGAGG
+
^_a\caacgefegihfhhi[_efefgfhifbgfffhheggfhhiiffgff^cgdbaaeeb_behdghhhhgdegddcddbbedbb^^\aa
@FCC1GWRACXX:6:1302:2023:49027#AGTCAGAA/1
GCCTAATCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCCAAGCCTAAGCCTAGGCCTACACCTAAGCCTAAGCCTAAGCCGCA
+
JPJJacJQQ\\bcdd_dd]X[^Re^J[J`QIYYdYPI^I^IIIYIHOOOacX^BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
@FCC1GWRACXX:6:1310:17686:75253#AGTCAGAA/2
TCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCCTGTCTCTTATACACATCTCTGATGGCGCGAGG
+
^PZcce^c]YbRYQ`bY`Z^`dfebg___`Xde[b[XX^c``cfffec^Z^[aaRW\acZMH]edgfhhdggYb]`_cZVZbBBBBBBBB
@FCC1GWRACXX:6:2101:8675:28447#AGTCAGAA/1
TCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCCTGTCTCTTATACACATCTCTGAGCGGGCTGGCAAGGCAGACCGAGTCAGA
+
___eeeeeggggiiihhfhiiiifhhfhihiiiiifghhaggghiagffhiiiiihhihhhhiggdecccccc_[`aacccc[XWbbbb
@FCC1GWRACXX:6:2210:6711:22244#AGTCAGAA/1
TCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAA
+
JP\ceedeggfggfhhiiiiiiiifdhfiibfhhifhhiihihhiihihiihihhiihiihhi_gbddgdd`ggee_bcdd]]]^b`ZZ_
~/Desktop/IBiS-Bootcamp/data
```

`head, tail, clear`

Explore the data

1. How big is the file?
2. What are the permissions?
3. Can we open it with atom?

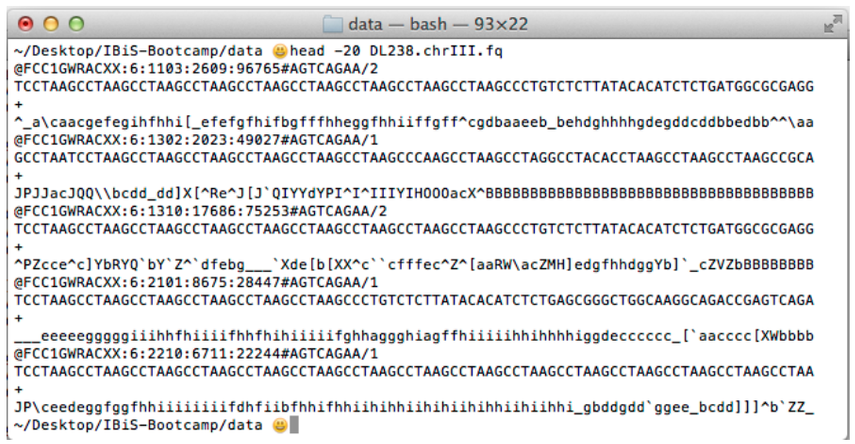
`less`

head, tail, clear

head, tail, clear, less

Illumina sequencing generates FASTQ files

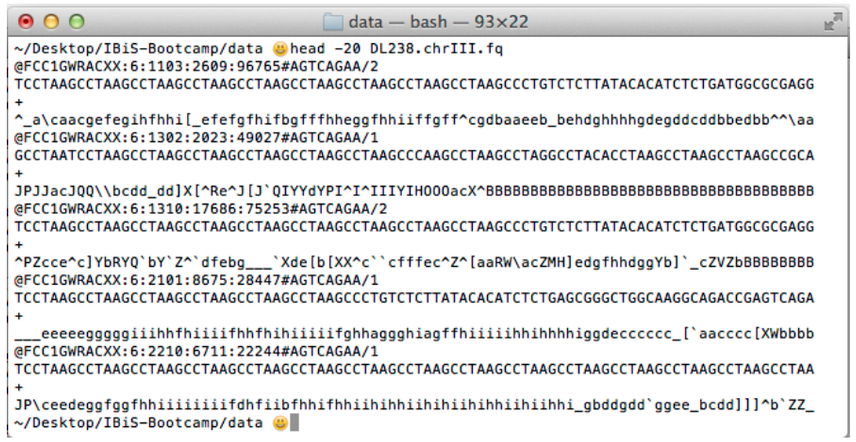
HiSeq2500



Each paired-end lane has four FASTQ files

FASTQ files contain the sequence and quality data, along with sequencer, index, and lane information.

Illumina sequencing analysis pipeline



FASTQ

+

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

FASTA

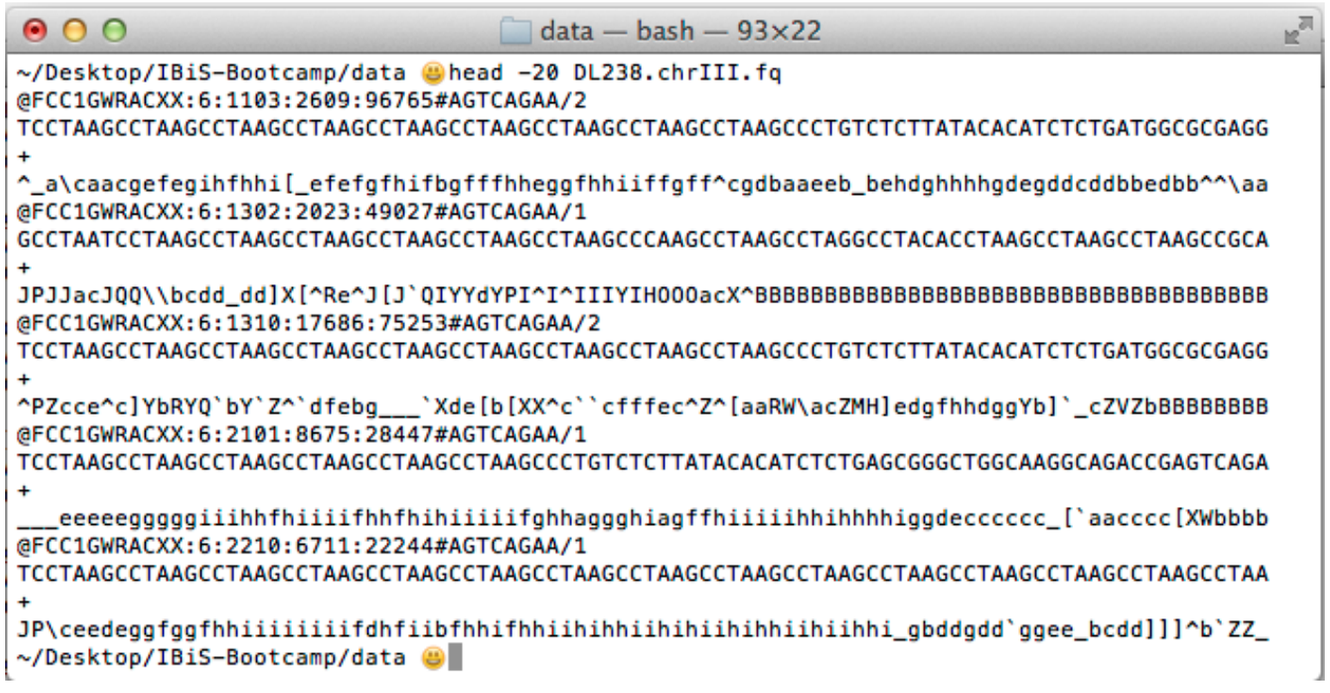
bwa

Binary alignment data
BAM file

samtools/bcftools

Variant Call File
VCF

OK. It's an Illumina sequence file. Now, what?



cp DL238.chrIII.fq seq.fq

cat DL238.chrIII.fq > seq2.fq

rm seq2.fq

gzip DL238.chrIII.fq

head, tail, clear, less

Let’s look for specific data within the FASTQ file

Let’s get all the lines in the file that start with ‘@‘.

`grep ‘^@‘ seq.fq > at_lines.txt`

globally search a regular expression and print

What HiSeq machines were used for sequencing?

`head, tail, clear, less, cp, mv, cat`

`cut` allows you to pull out columnar data from a text file.

delimiter - A character used to separate pieces of data.

field - A column formed by delimiters.

Example

```
@FCC1GWRACXX:6:1103:2609:96765#AGTCAGAA/2
@FCC1GWRACXX:6:1302:2023:49027#AGTCAGAA/1
```

If we make the delimiter a colon (:), then the corresponding fields will look like below.

Field	1		2		3		4		5
	@FCC1GWRACXX	:	6	:	1103	:	2609	:	96765#AGTCAGAA/2
	@FCC1GWRACXX	:	6	:	1302	:	2023	:	49027#AGTCAGAA/1

cut

Field 1 2 3 4 5

```
@FCC1GWRACXX:6:1103:2609:96765#AGTCAGAA/2
@FCC1GWRACXX:6:1302:2023:49027#AGTCAGAA/1
```

Command	Translation	Output
`cut -f 1 -d ":" at_lines.txt`	"cut the first field formed using the colon delimiter."	`@FCC1GWRACXX` `@FCC1GWRACXX`
Save output to machine.txt		
`cut -f 1,3,5 -d ":" at_lines.txt`	"cut the 1st, 3rd, and 5th fields formed using the colon delimiter."	`@FCC1GWRACXX:1103:96765#AGTCAGAA/2` `@FCC1GWRACXX:1302:49027#AGTCAGAA/1`

Let’s look for specific data within the FASTQ file

We’d like to know how many sequencers were used.

```
sort machine.txt > sorted_machine.txt
```

First, we have to sort the data and save to sorted_machine.txt

```
uniq sorted_machine.txt
```

How many lines of data do we have?

```
wc -l sorted_machine.txt
```

head, tail, clear, less, cp, mv, cat, grep, cut

It’s time to put it all together.
Enter the pipe operator.

```
grep ‘^@' seq.fq | cut -f 1 -d ':' | sort | uniq
```

Challenge:
Find all unique barcodes in the seq data
and save to a file called barcode.txt

```
grep '^@' seq.fq | cut -f 2 -d '#' | cut -f 1 -d '/' | sort | uniq > bc.txt
```

head, tail, clear, less, cp, mv, cat, grep, cut, sort, uniq, wc

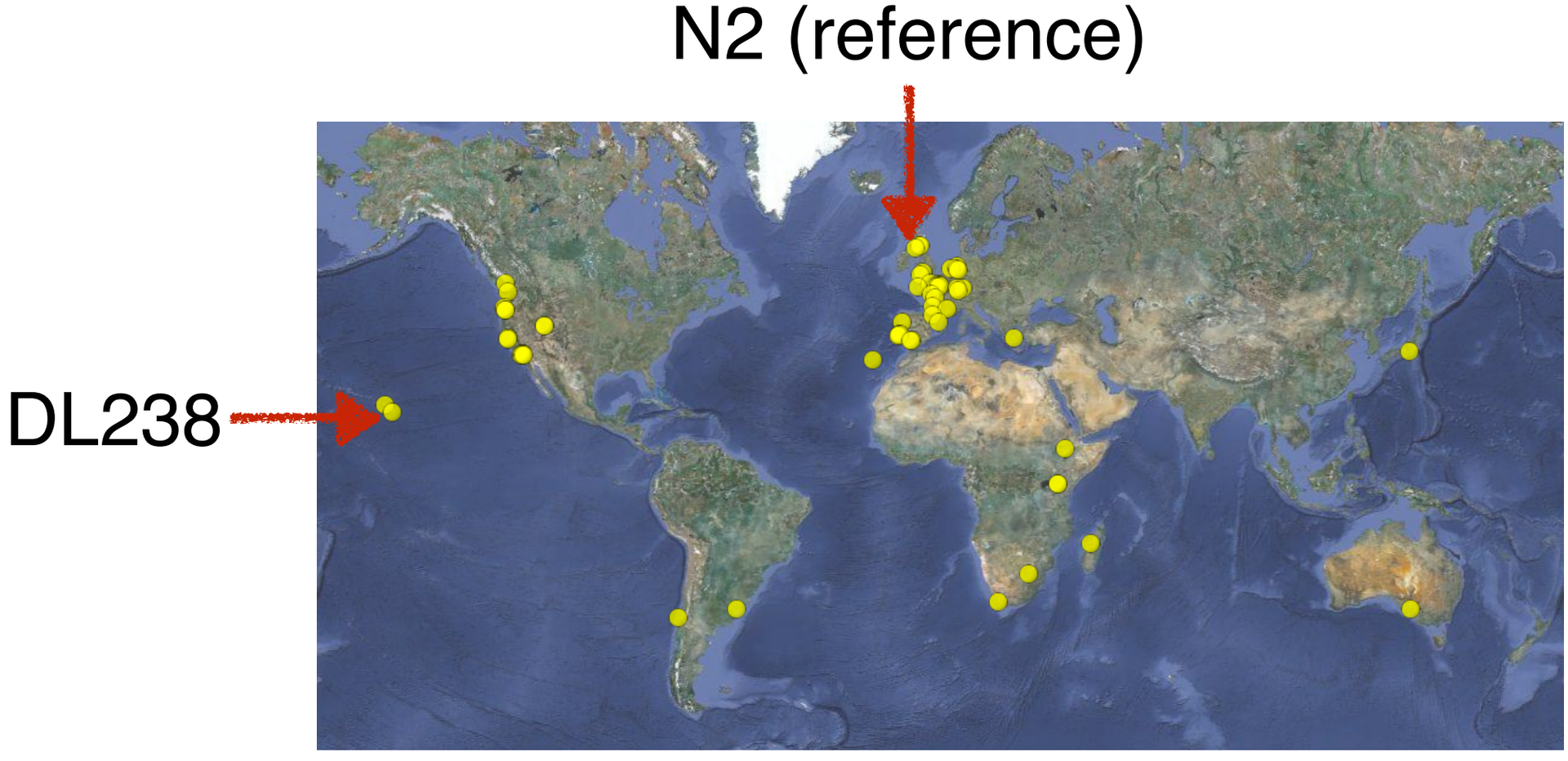
Let’s analyze the sequence data for variation.

Go to scripts folder in IBiS_bootcamp and open DL238.Variant.Calling.sh with Atom

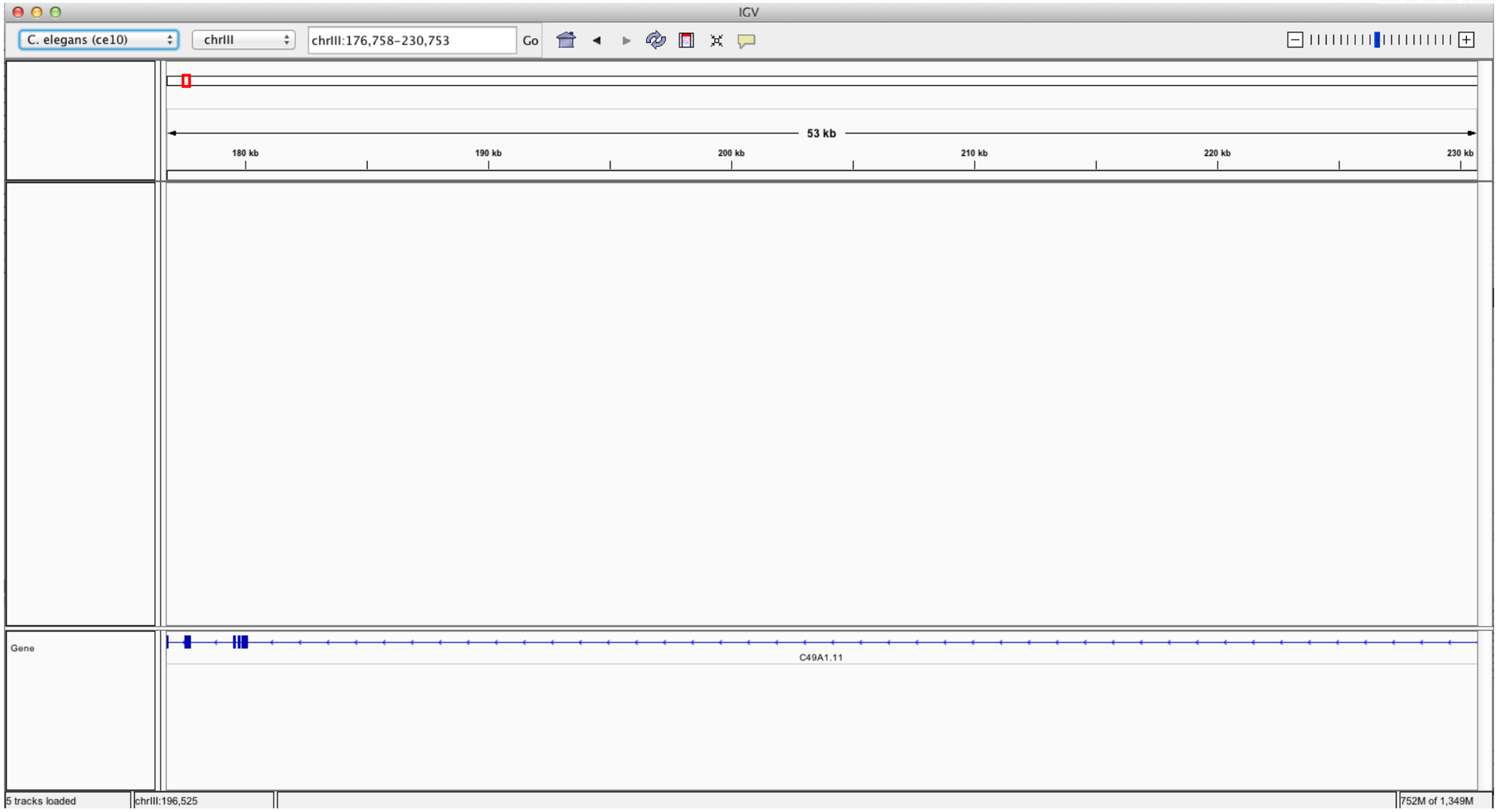
Let's use the script to analyze the sequence data for variation.

Go to the command line
and then go to the data folder in IBiS_Bootcamp

Type: `bash ../scripts/DL238.Variant.Calling.sh`

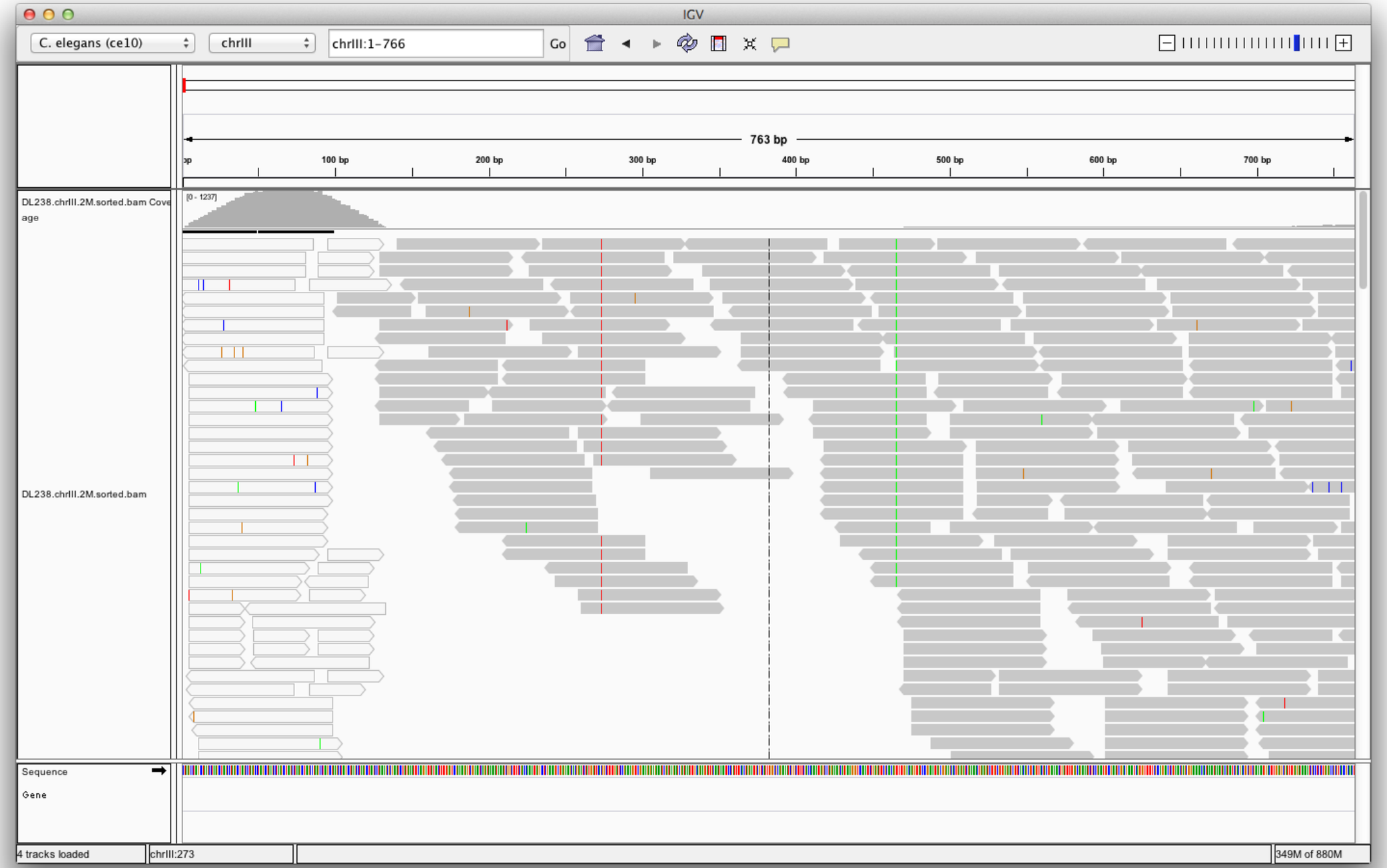


Let's visualize the sequence data



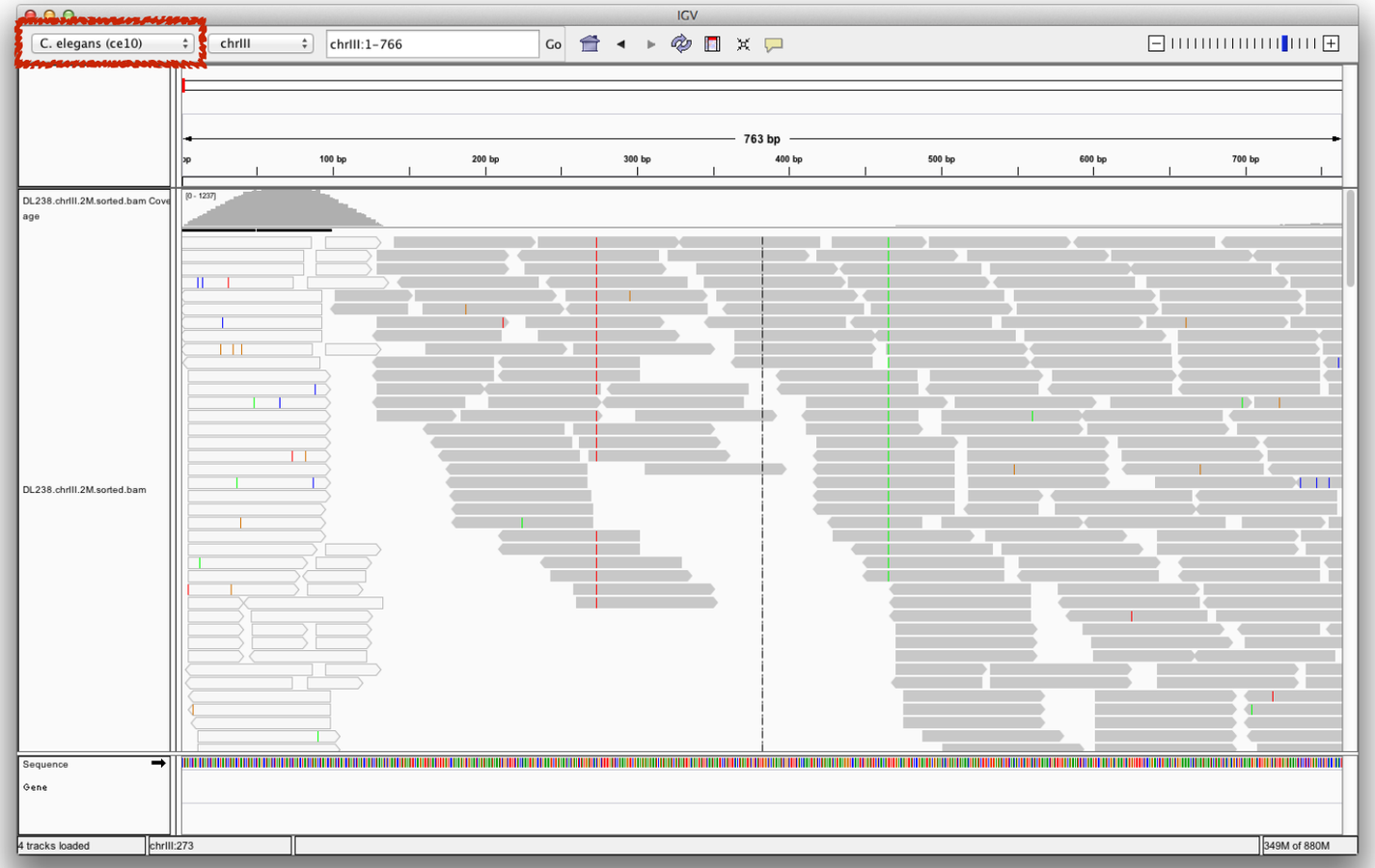
Type `igv` from your command line. IGV will start

Integrative Genomics Viewer

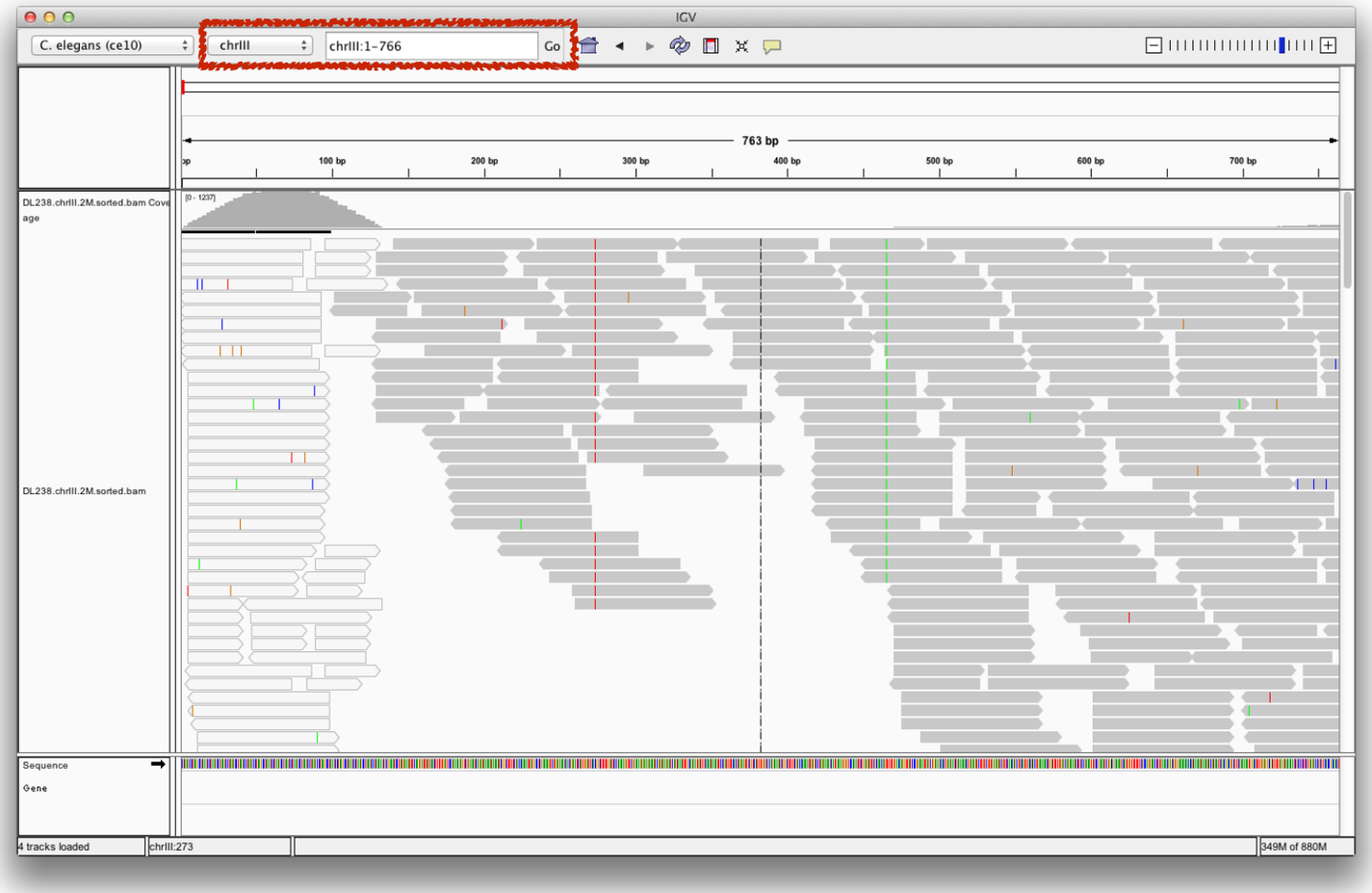


A Genome browser for sequence data

Integrative Genomics Viewer



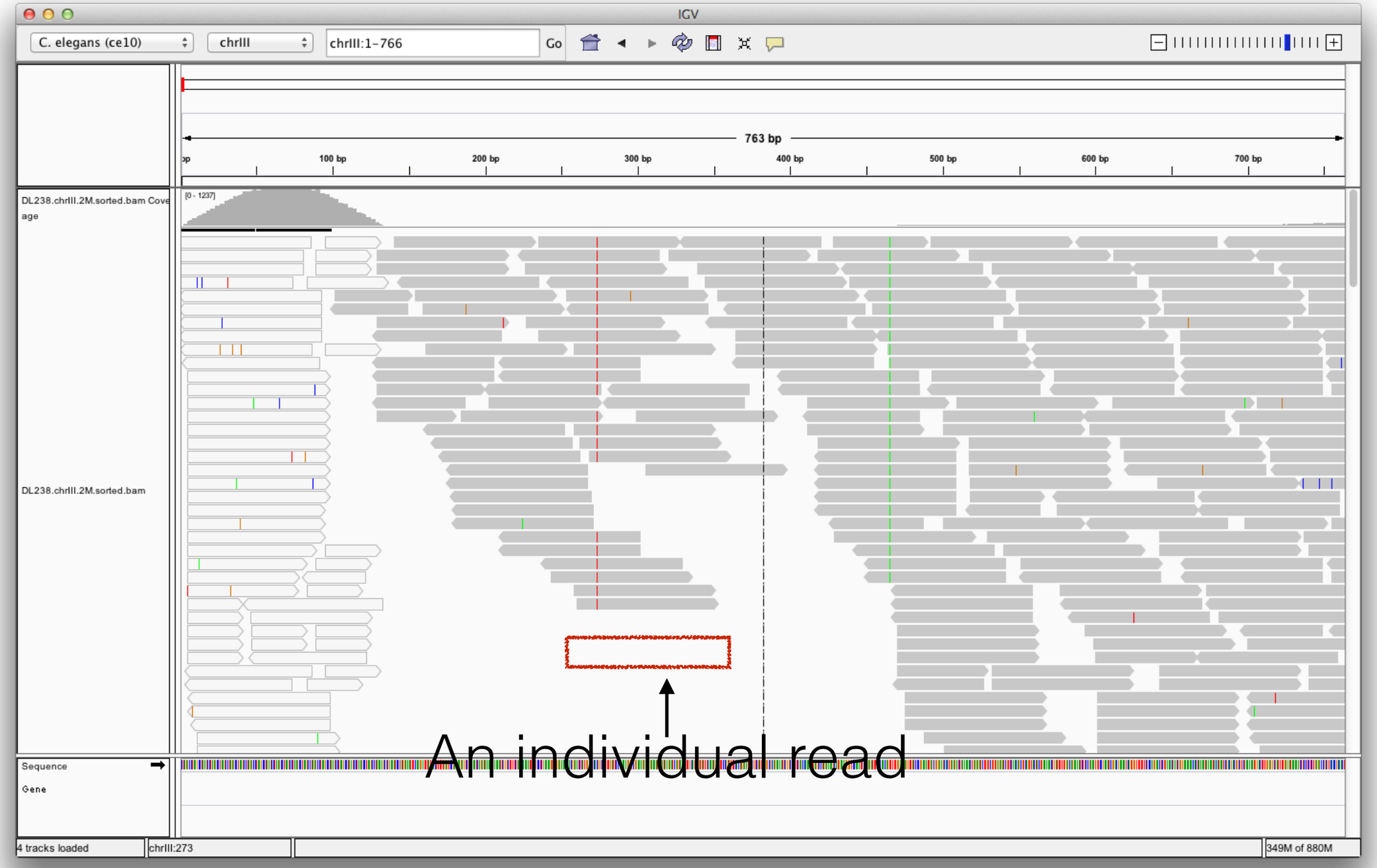
Download *C. elegans* (ce10)



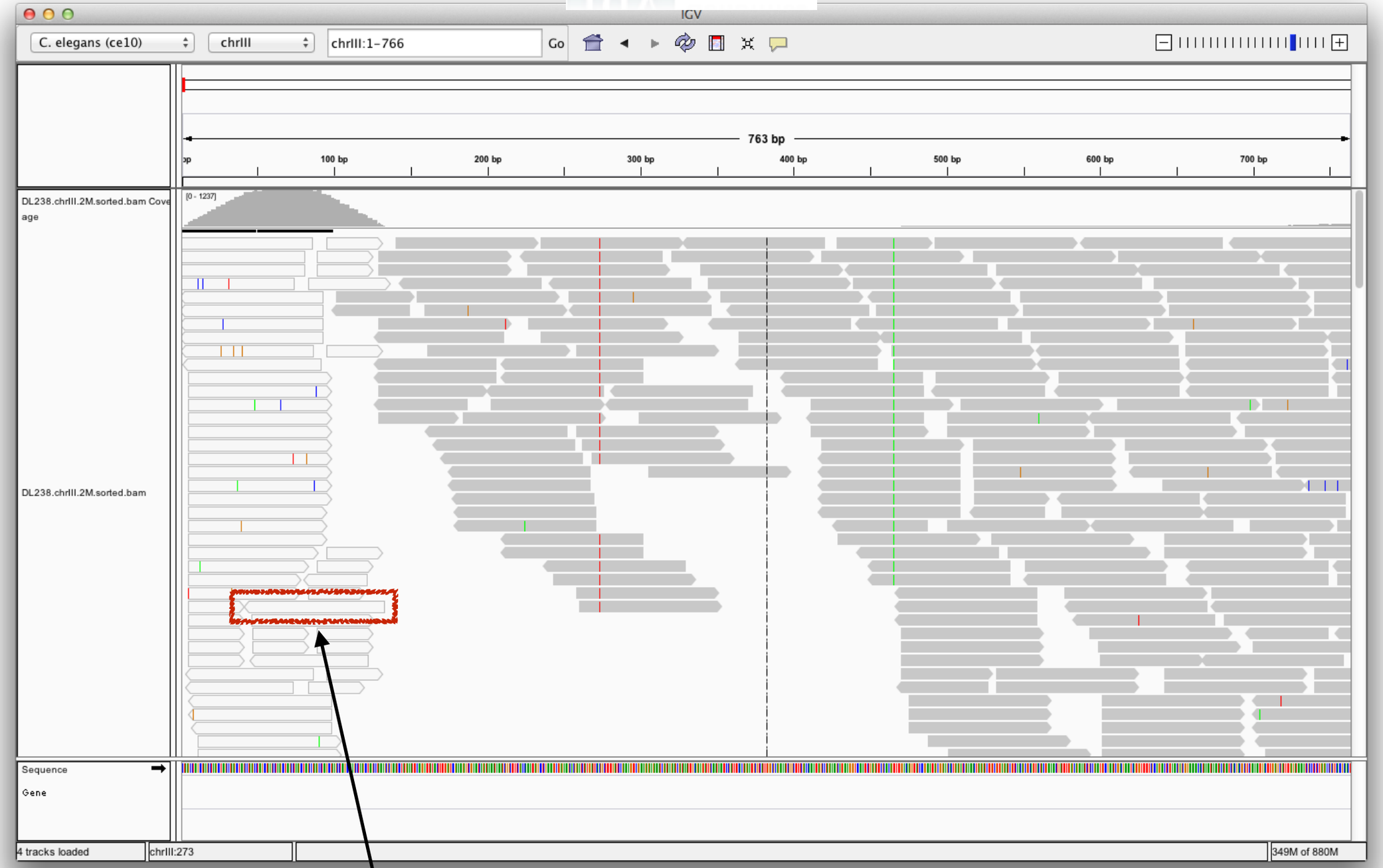
1. Drag DL238.chrIII.2M.sorted.bam into IGV

2. Navigate to chrIII:1-766

Integrative Genomics Viewer



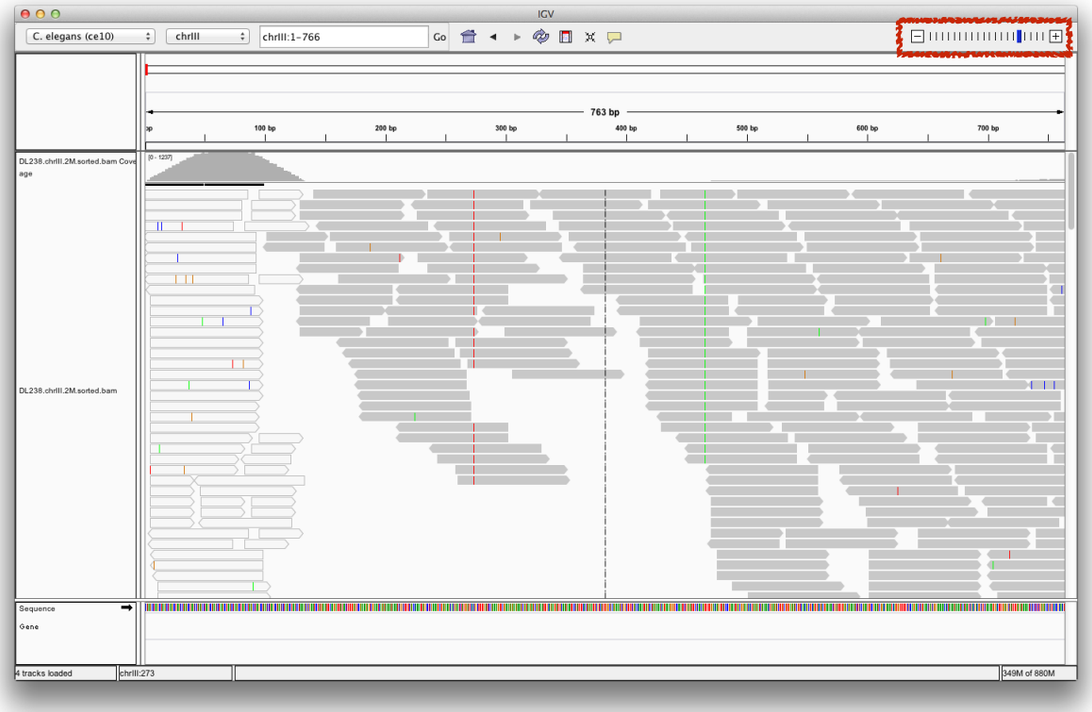
A Genome browser for sequence data



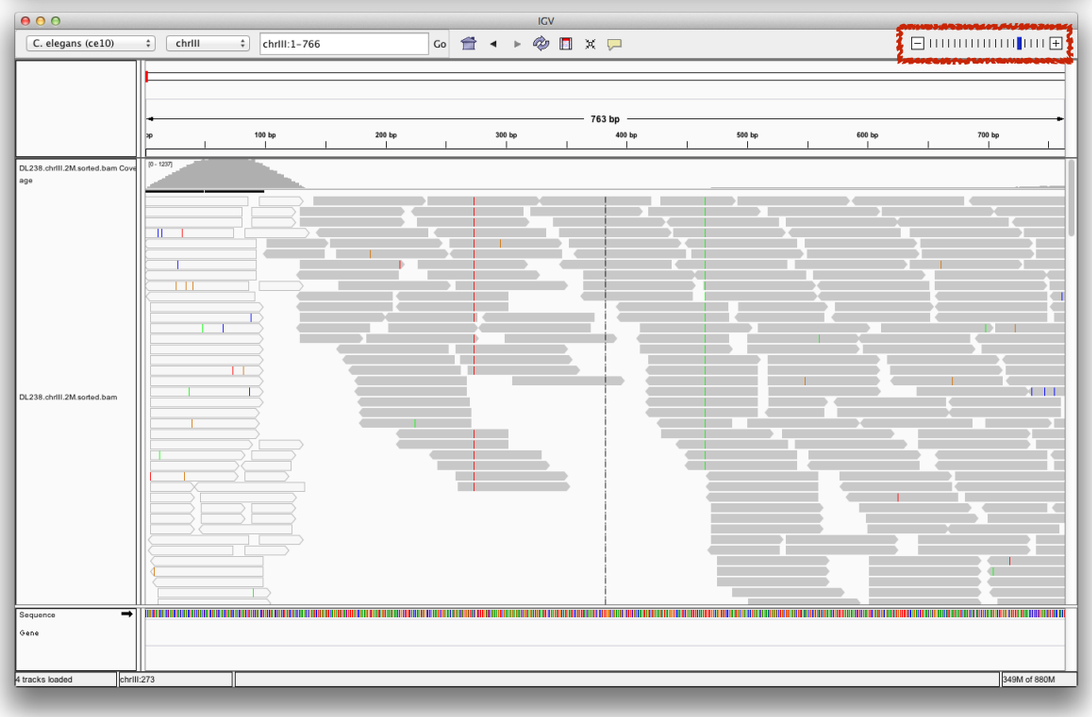
Light Grey = Mapping quality of 0.

Why might the reads on the left side all be gray?

Integrative Genomics Viewer

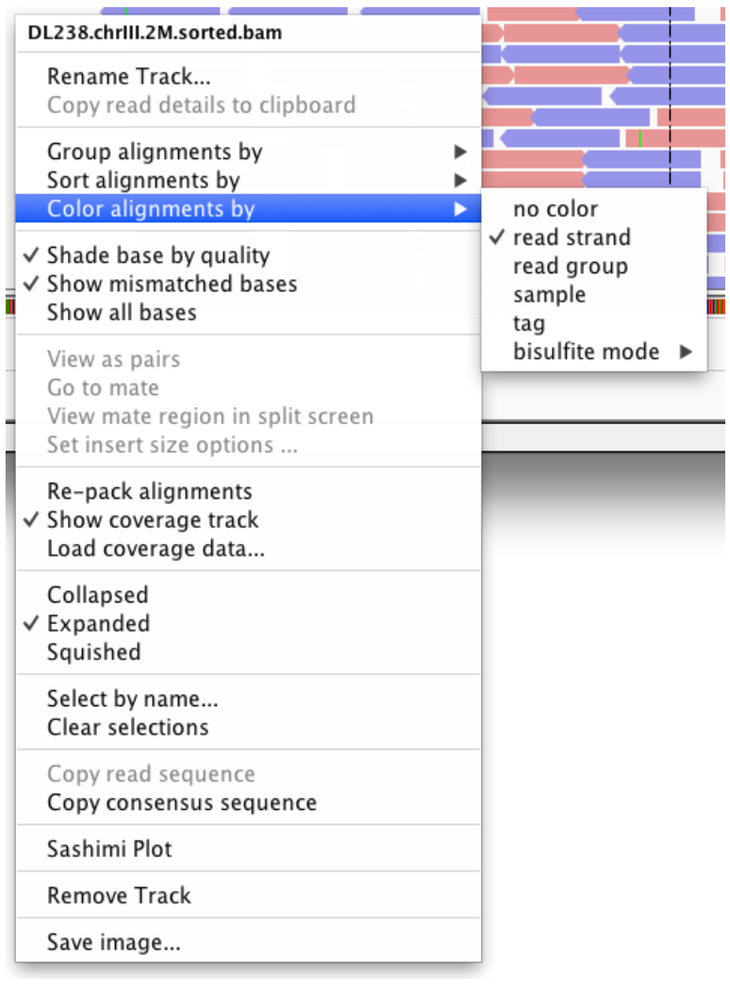


Zoom out



Pan left and right

Integrative Genomics Viewer

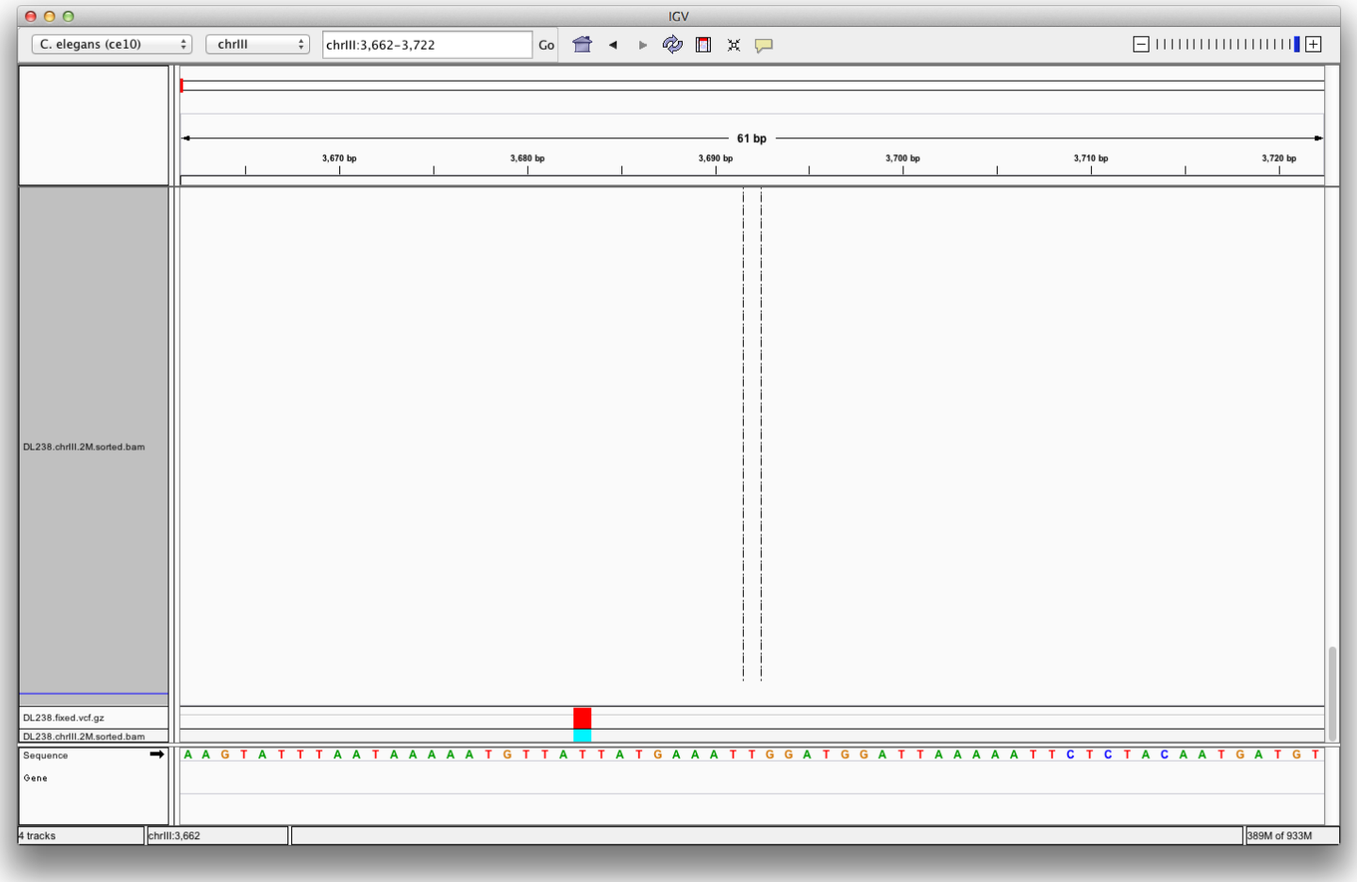


Right click - lots of ways to examine data.

Try setting size to ‘squished’

We also called variants using samtools+bcftools.

1. Drag DL238.fixed.vcf.gz into IGV



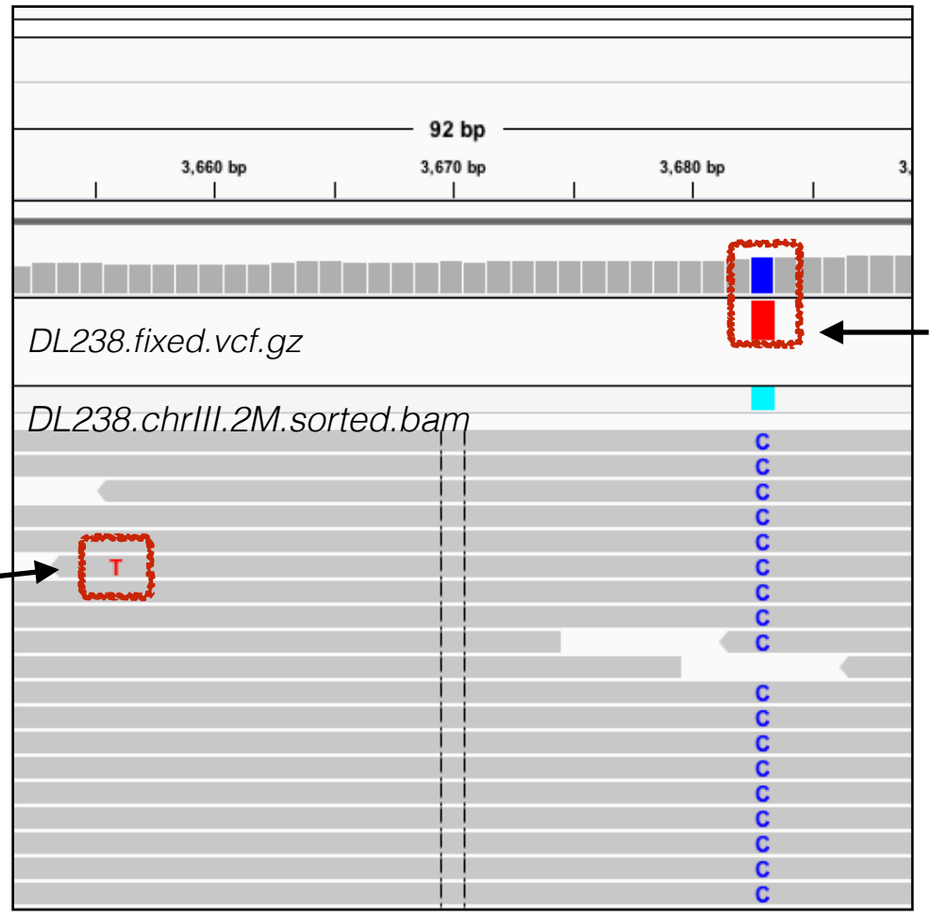
2. Drag the sorted.bam file down so the vcf is on top.

IGV Scavenger Hunt

Within the first 5,000 base pairs, find:

Three SNPs

Two Indels



Legitimate SNP,
Called correctly
and supported by BAM

Sequencing Error

What to do when you are lost?

Google

stackoverflow

Biostars
BIOINFORMATICS EXPLAINED

www.biostars.org

SEQanswers
the next generation sequencing community

www.seqanswers.com

A few additional commands for UNIX

`time` = time how long an operation takes

`screen` = allows multiplexing of terminal windows

`nohup` = no hang ups, process errors go to a log

`sed` = search and edit text

`awk` = search, extract, or transform text

Also, check out datamash utilities

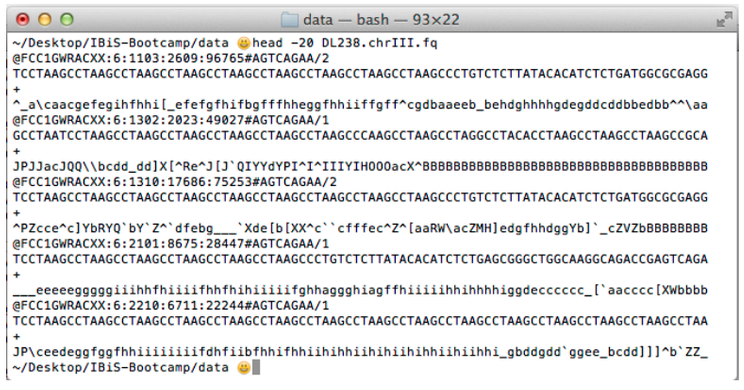
Further experiences:
http://freeengineer.org/learnUNIXin10minutes.html
http://www.ee.surrey.ac.uk/Teaching/Unix/
or just google: “unix command line tools”

Remote commands for UNIX

`ssh` = secure shell, remote access to UNIX systems

`rsync` = remote synchronization, remote copy

The computational biology triumvirate



Free and open source are core principles of science

You ARE learning to “code”!

is a software environment for statistical computing

is free!

Software	Cost
SAS	$8,700 - $140,000 / year
MATLAB	$2150 + $1,000s for modules
R	**$0**

is managed by users!

Comprehensive R Archive Network

http://cran.us.r-project.org

Over 5000 free add-on *packages*

Stack Overflow

http://stackoverflow.com

Free help

is popular for data analysis!

What languages you used for data mining / data analysis?

What programming languages you used for data mining / data analysis in the past 12 months? [570 voters]

R (257)	45%
SQL (184)	32%
Python (140)	25%
Java (139)	24%
SAS (121)	21%
MATLAB (83)	15%
C/C++ (73)	13%
Unix shell/awk/gawk/sed (59)	10%
Perl (45)	7.9%
Hadoop/Pig/Hive (35)	6.1%
Lisp (4)	0.7%
Other (70)	12.0%
None (7)	1.2%

Notes
On average, data miners use 2.1 languages. Among the top 5 languages, only about 15-25% used that language alone.

The breakdown by region is:

- US/Canada, 42%
- Europe, 30%
- Asia, 16%
- Latin America, 4.9%
- AU/NZ, 2.8%
- Africa/MidEast, 3.0%

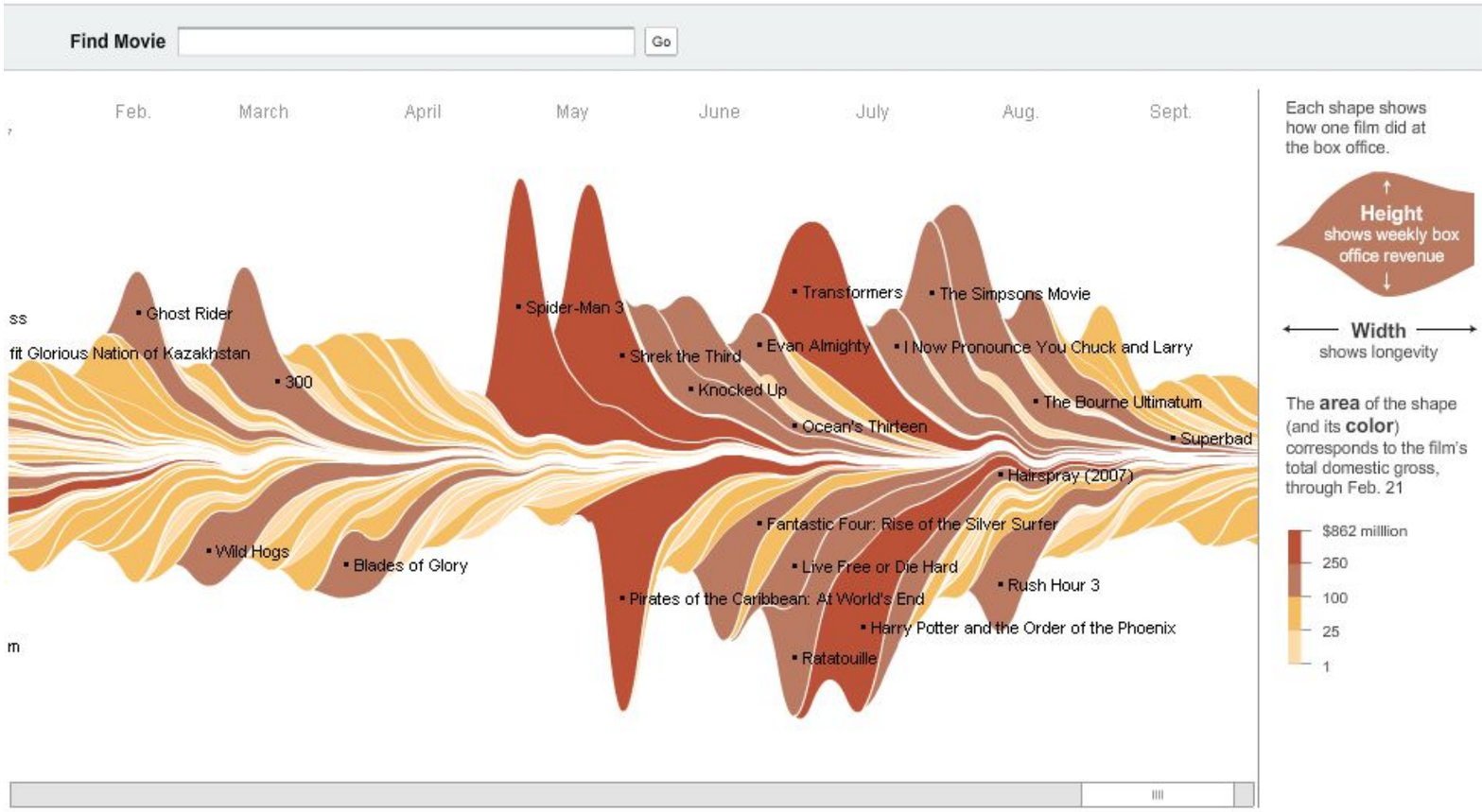
has amazing graphics!

February 23, 2008

SIGN IN TO E-MAIL OR SAVE THIS | FEEDBACK

The Ebb and Flow of Movies: Box Office Receipts 1986 — 2008

Summer blockbusters and holiday hits make up the bulk of box office revenue each year, while contenders for the Oscars tend to attract smaller audiences that build over time. Here's a look at how movies have fared at the box office, after adjusting for inflation.

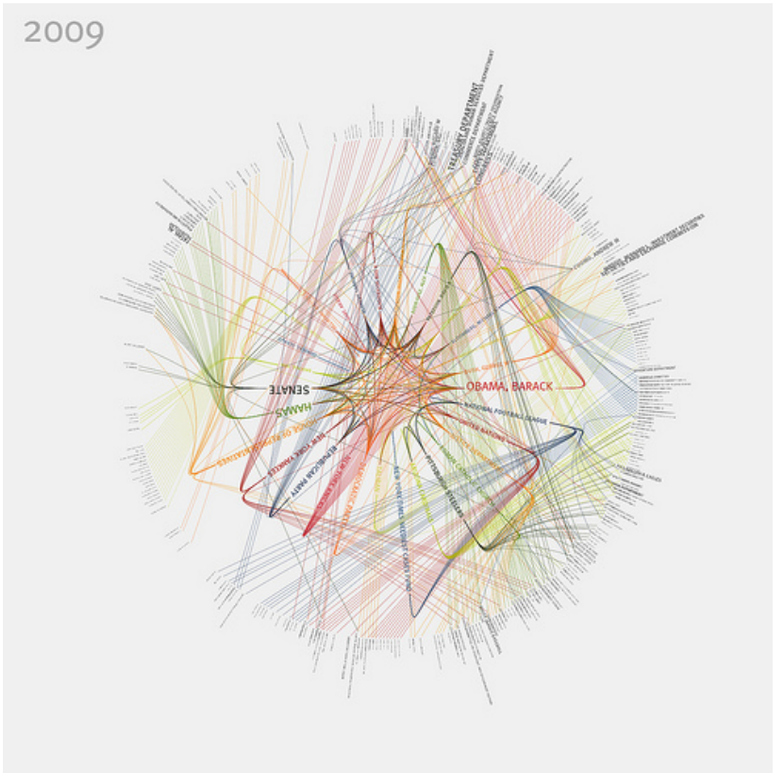


Sources: Baseline StudioSystems; Box Office Mojo

Mathew Bloch, Lee Byron, Shan Carter and Amanda Cox

New York Times favorite tool

has amazing graphics!



is a programming language!

- Integrates into the command line or runs command line utilities
- Integrates with python or runs python
- Integrates with C or runs C
- Integrates with a variety of database systems

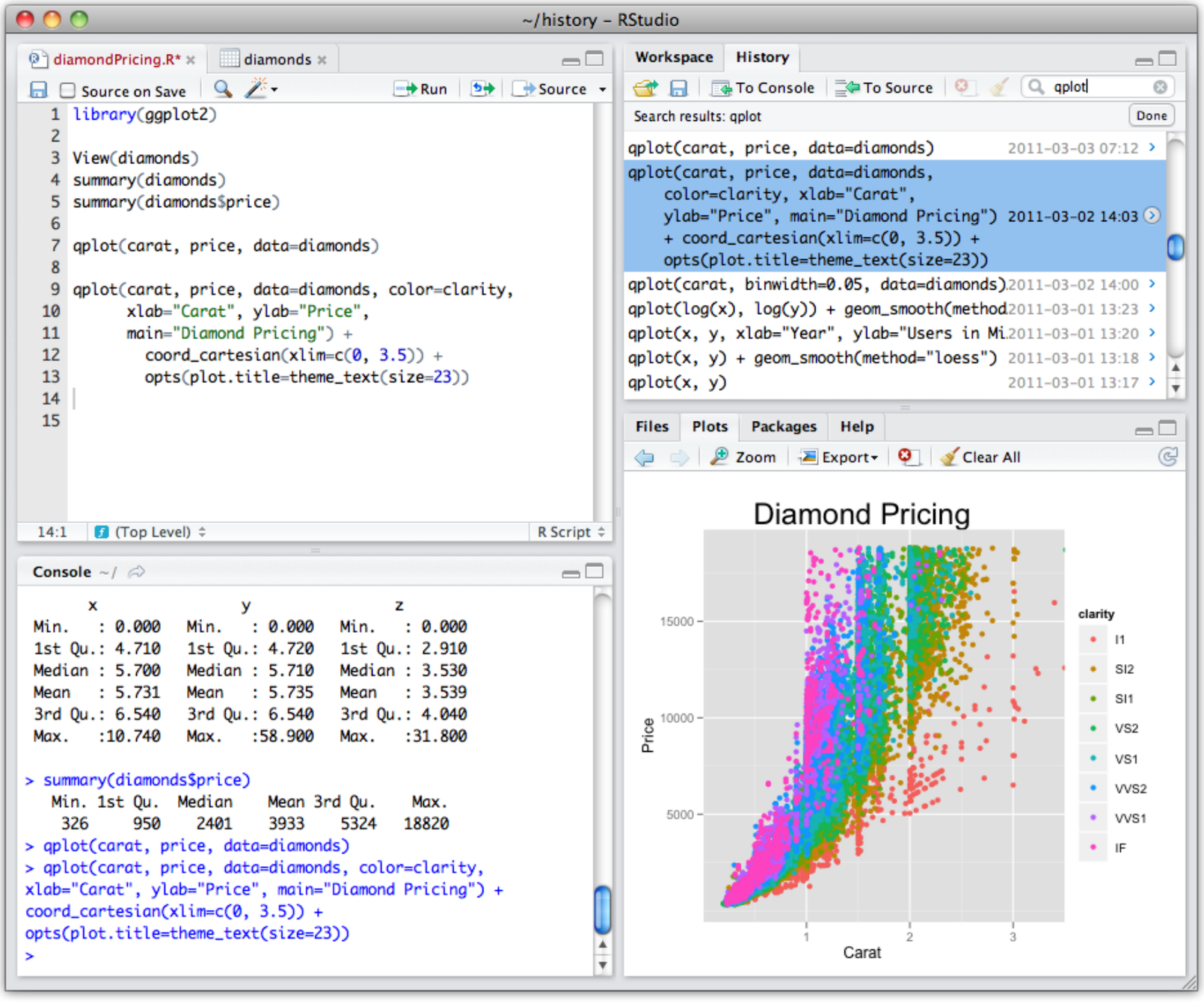
RStudio makes even easier!

Editor

Workspace

Console

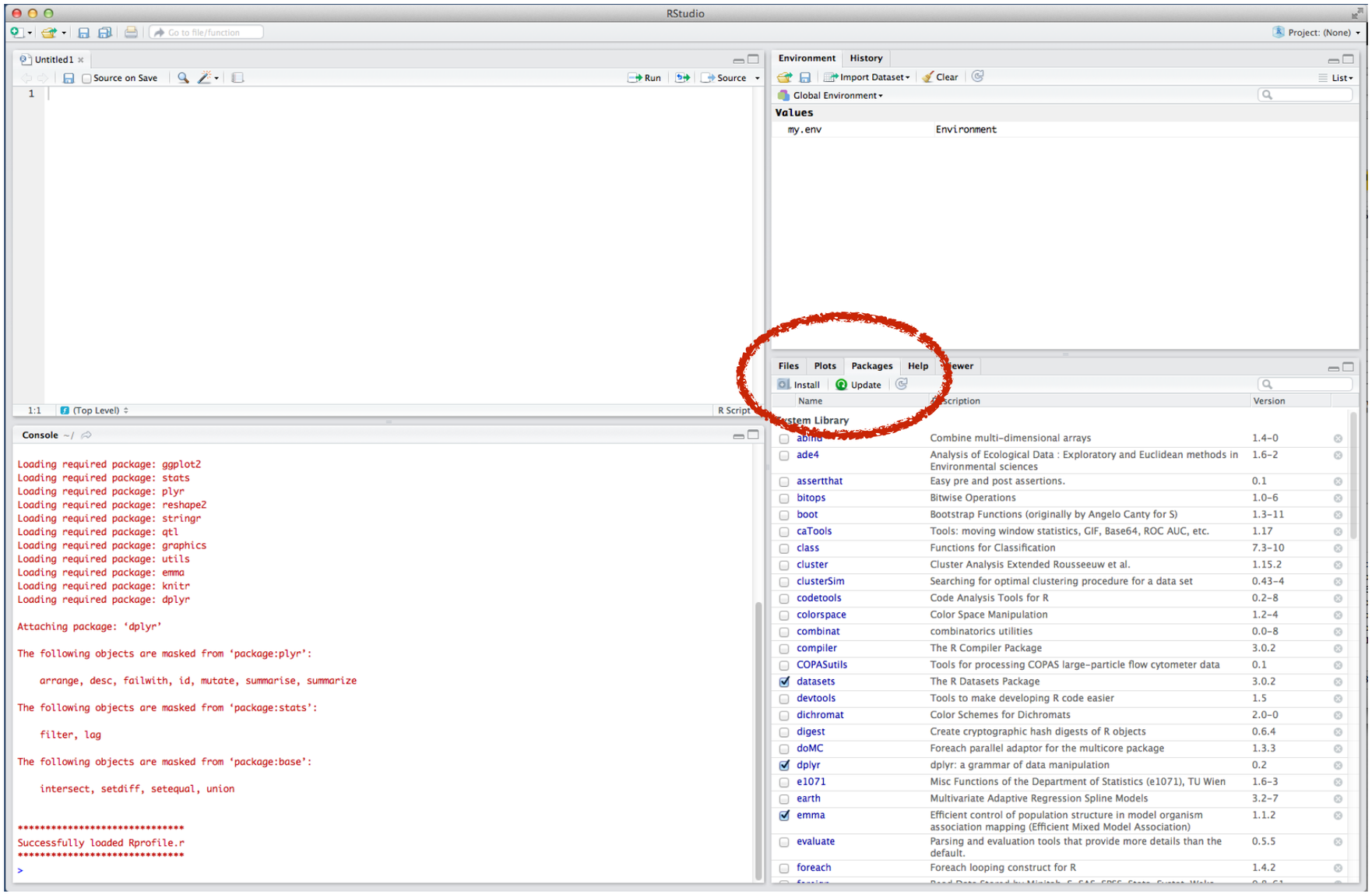
Graphics



We all owe Hadley Wickham a beer.

- Chief Scientist at RStudio
- Made ggplot2 package for beautiful R graphics
- Made dplyr package for grammar of data manipulation
- Made reshape2, stringr, lubridate, tidyr, plyr, among many others

Let’s install some packages



1. Click on Packages in bottom right pane
2. Click on Install
3. Make sure the Install from drop down is CRAN
4. Enter ggplot2, dplyr, stringr, tidyr, knitr

Basic R syntax

> The prompt

Functions are words followed by (), *i.e.* mean(x) for calculating the mean of x

Equations are entered as assignments, *i.e.* y <- mean(x) for y is equal to the mean of x.

If you can't remember a function, use apropos(). For example, apropos("var") will display every item in the workspace with "var"

You can also use ?function. For example, ?mean

Try ????mean

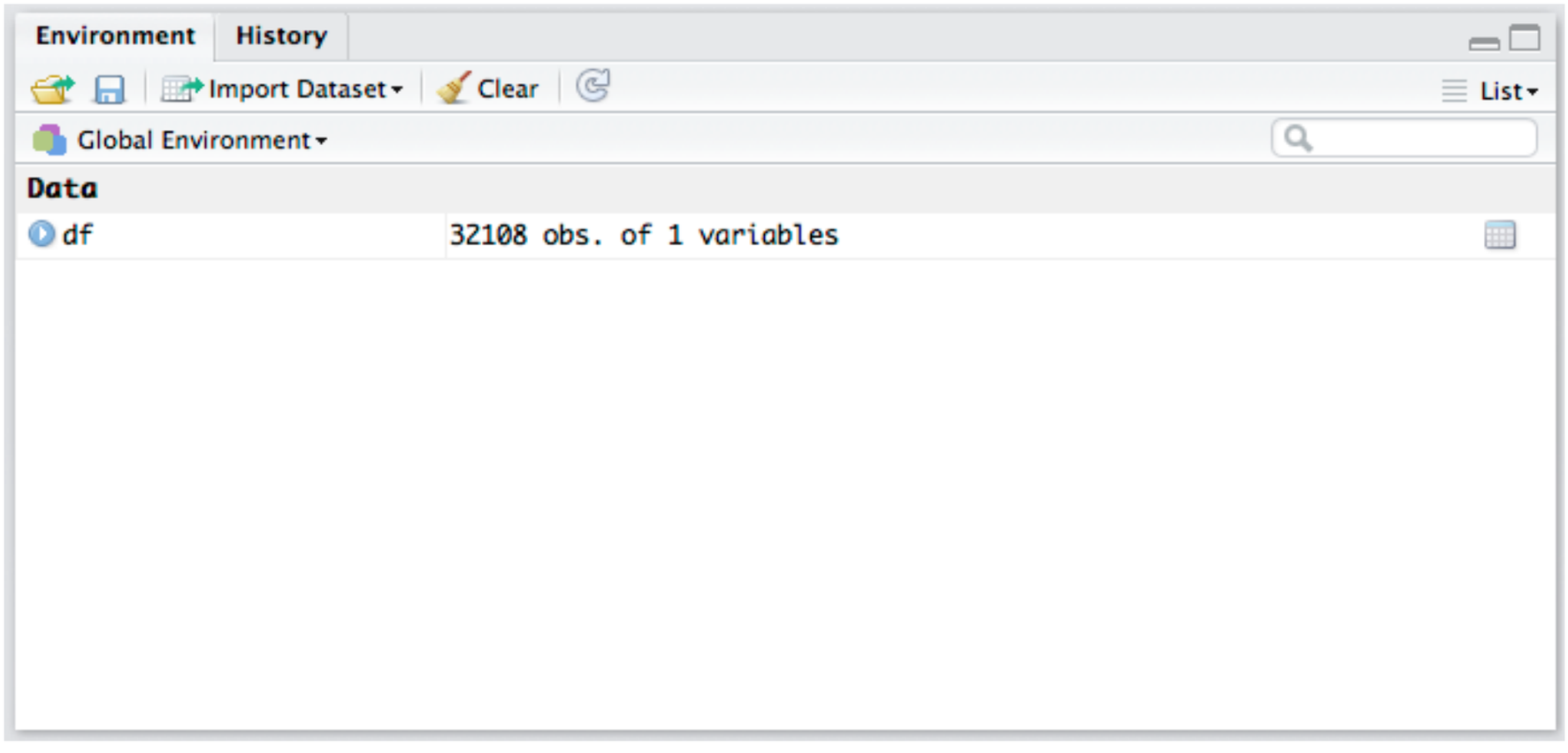
Let’s make our own functions

```
> add20 <- function(x) {x + 20}

> add20(10)

> sqrtAdd20 <- function(x) {sqrt(x) + 20}

> sqrtAdd20(10)
```

Reading in data

`read.delim()` is a generic function to read in simple text files, like csv or tsv files.

```
df <- read.delim(file="~/Desktop/IBiS-Bootcamp/data/sorter_data.txt", header=TRUE, sep="\t")
```



A new data object is in your workspace. Click on it. Type **`df`** to display all of it.
Try **`head(df)`** and **`tail(df)`**.
Try **`summary(df)`**.

Manipulating data

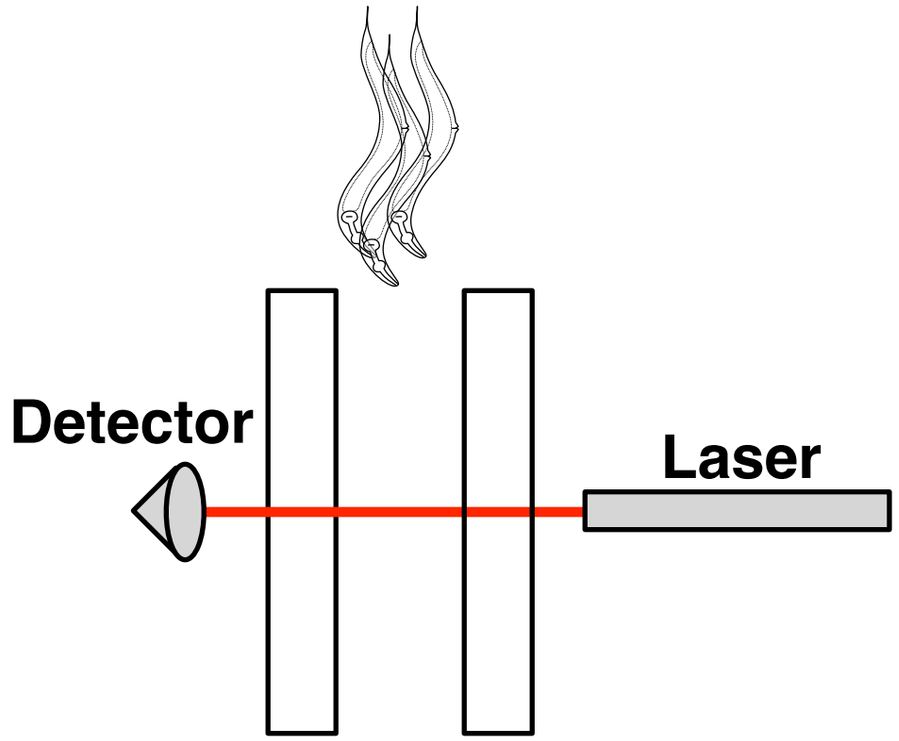
It looks like the tail has some data that do not match the pattern of the rest of the file.

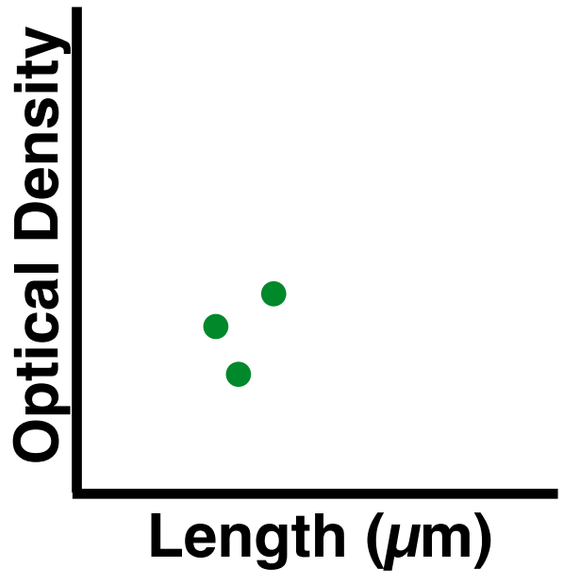
```
df <- read.delim(file="~/Desktop/IBiS-Bootcamp/data/sorter_data.txt", header=TRUE, sep="\t", nrows=32093)
```

Type `str(df)` to look at the structure of the file

```
Console ~/
' ends field 1 on line 1 when detecting types: Green PMT voltage 600
> str(df)
'data.frame':   32093 obs. of  27 variables:
 $ Id         : int  0 1 2 3 4 5 6 7 8 9 ...
 $ Plate      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Row        : chr  "A" "A" "A" "A" ...
 $ Column     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Clog       : chr  "N" "N" "N" "N" ...
 $ Scan.rate  : int  2500 2500 2500 2500 2500 2500 2500 2500 2500 2500 ...
 $ Status.sort: int  0 0 0 0 0 0 0 0 0 0 ...
 $ Status.sel : int  40 40 40 40 40 40 40 40 40 40 ...
 $ TOF        : int  52 556 88 639 531 34 247 152 96 79 ...
 $ EXT        : int  120 2375 130 2697 2520 36 534 474 210 178 ...
 $ Green      : int  2 148 3 205 151 1 16 66 20 4 ...
 $ Yellow     : int  3 193 5 241 170 3 21 26 7 5 ...
 $ Red        : int  0 90 2 102 71 2 14 11 4 2 ...
 $ PH.Ext     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Ext     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PC.Ext     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PH.Green   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Green   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PCGreen    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PH.Yellow  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Yellow  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PCYellow   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PH.Red     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Red     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PCRed      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Time.Stamp : int  2933417 2933417 2933432 2933432 2933463 2933463 2933463 2933495 2933495 2933495 ...
 $ X          : logi  NA NA NA NA NA NA ...
>
```

A worm sorter allows
for automated scoring from 96-well plates





Data structures in R

Data.frames are data organized into rows and columns.

Data frames are organized by [row,column]. You must use the comma!

You can look at a vector of the data frame using $, *i.e.* **`df$TOF`**

Or you can specify the row or column to output, *i.e.* **`df[,9]`** outputs the ninth column and **`df[,"TOF"]`** outputs the column named “TOF”

Console ~/

```
> 
> df[,"TOF"]
  [1]  52 556  88 639 531  34 247 152  96  79  54 133  64 471 417  21 491 400 504 163  25 436  57  36
 [25] 190  29 574 476 394  51 584  74 172 373  26  49  92  31  25 521 177 108 266  59  57 435  26 135
 [49]  62 197  52  38 236  97 545 438  82  28 129  76  67  48 105 140 147  22  88  74  24 265 162  21
 [73]  31 122  50  20  70  53 329  27  71  80 158 517 523 448  40  30  86  49 519 134  26  59  49  78
 [97]  65  46  91 219 259  21  85 128 244 216  59 188 182  56  43  75  51  60 136 114  80  77  52  58
[121] 219  26 203  81  25 127 190  53  63  54  37  40 468 197 172  59  85 196  45  23  24 245  58 209
[145] 314 400 229 488  99  35 260  46 103 199 421 116 481 133 112  40  55 164  51 161 212 161 128 212
[169]  64  23 491  47  20  22  69 471  23 120  28  51 524 124 265  20  69  22  62 481 103  32  53 143
[193]  48  37 203 116  89  31  45 255  73 281 296 105 472  49  20  43  20  46 113  92 143 194  46  87
[217] 194  23  70 525  54  52  51  46  36  59  71  27 506  22 331  73 847 438  58 160  50  55 402  31
[241] 208 524  58  50  78 479  54 102  55  73 295 516 109  38 121 142 127 132 486  44 101 335  66 393
[265]  94 205  77 172 205 565  29 133  54 443 511  34  97 139 182  25  39  27 182  50  60 101  56  59
[289] 146  44 206  48 118 324  36 310 127 537 194  59  46 111 411  61 439  74  32  51  91 115  46 250
[313] 467 229  47  58 181  25 103  35 184  42 295 185 406 119  32  61 549 635  84 186 111 179  49 142
```

Data structures in R

Data frames are two-dimensional data sets of any data type

Vectors are one-dimensional data sets of any one data type

Lists are groups of vectors, data frames, or other lists.

How do we explore data objects?

`names(df)` = gives the names of columns of df

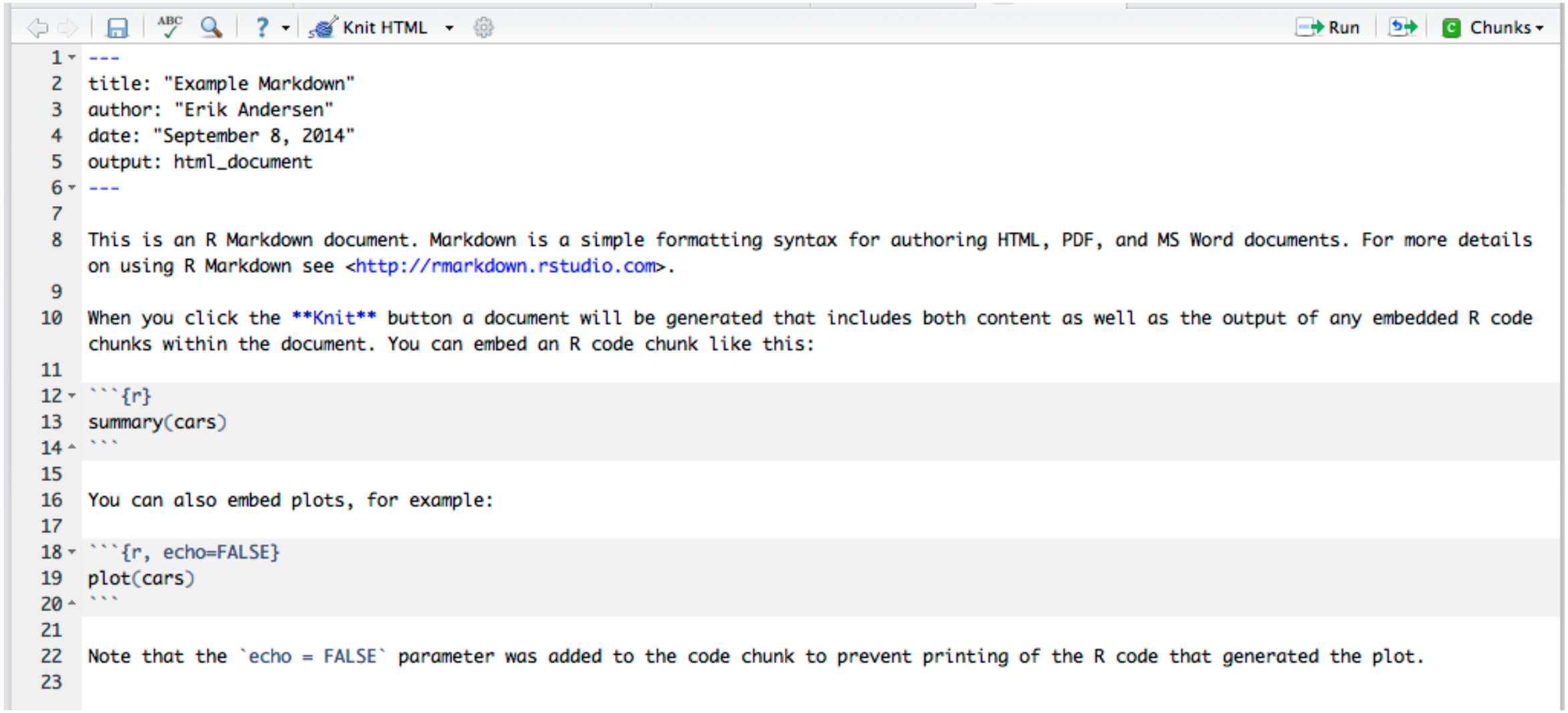
`rownames(df)` = gives the names of rows of df

`colnames(df)` = gives the names of columns of df

`dim(df)` = outputs the number of rows then columns of df

`length(df)` = outputs the length of df

RStudio can generate markdown reports



1. **Go to File, New File, R Markdown**
2. **Install the necessary packages**
4. **Knit an HTML report**

What to do when you are lost?

http://www.r-bloggers.com

http://gettinggeneticsdone.blogspot.com

http://rseek.org

Day #2 Homework

1. Install the swirl package
2. Type `swirl()`
3. Take the introduction to R programming class

Learn R, in R.