

THE MISSING GUIDE

The
**BIOSTAR
HANDBOOK**

BIOINFORMATICS DATA ANALYSIS GUIDE

RNA-SEQ

ANALYSIS

GENE

EXPRESSION

NGS

SEQUENCING

BLAST

SEARCH

GENOME

ANALYSIS

Book updated on February 7, 2020

Contents

I Preface	3
1 Welcome to the Biostar Handbook	5
1.1 The Biostar Handbook Collection	5
1.2 News: Shell casting (January 2020)	6
1.3 How to download the book?	6
1.4 Get notified	7
1.5 Current online courses	7
1.6 How was the book developed?	8
1.7 How is this book different?	9
1.8 Who is a Biostar?	9
1.9 Access your account	10
2 About the author	11
3 Why bioinformatics?	15
3.1 What is this book about?	15
3.2 What is covered in the book?	16
4 What is bioinformatics?	17
4.1 How has bioinformatics changed?	17
4.2 What subfields of bioinformatics exist?	18
4.3 Is there a list of functional assays used in bioinformatics?	19
4.4 But what is bioinformatics, <i>really</i> ?	21
4.5 Are bioinformaticians data janitors?	21
4.6 Is creativity required to succeed?	22
4.7 Are analyses all alike?	23
4.8 Should life scientists know bioinformatics?	24
4.9 What type of computer is required?	24

4.10 Is there data with the book?	25
4.11 Who is the book for?	25
4.12 Is bioinformatics hard to learn?	25
4.13 Can I learn bioinformatics from this book?	25
4.14 How long will it take me to learn bioinformatics?	26
5 Biology for bioinformaticians	27
5.1 What is DNA?	28
5.2 Is there a directionality of DNA?	31
5.3 What is sense/antisense?	31
5.4 What is DNA sequencing?	32
5.5 What gets sequenced?	32
5.6 What is a genome?	32
5.7 What is a genome's purpose?	33
5.8 How big is a genome?	33
5.9 What is RNA?	34
5.10 How does a genome function?	34
5.11 What is a protein?	35
5.12 How are proteins made?	36
5.13 What is an ORF?	37
5.14 What is a gene?	37
5.15 Do genomes have other features?	38
5.16 What is homology?	39
6 How is bioinformatics practiced?	41
6.1 What is the recommended computer for bioinformatics?	42
6.2 How much computing power do we need?	42
6.3 Does learning bioinformatics need massive computing power?	42
6.4 What about the cloud?	43
6.5 Do I need to know Unix to do bioinformatics?	44
6.6 Do I need to learn a programming language?	44
6.7 Are there alternatives to using Unix?	44
6.8 What is Bioconductor?	45
6.9 What is Galaxy?	46
6.10 What is BaseSpace?	47
6.11 Are commercial bioinformatics software packages expensive?	48
6.12 Should I freelance as a bioinformatician?	49
6.13 What do bioinformaticians look like?	50

7 How to solve it	52
7.1 What is holistic data analysis?	52
7.2 How do I perform a holistic analysis?	53
7.3 What are the rules of a bioinformatics analysis?	54
7.4 What does “make it work” mean?	55
7.5 Why is fast so important?	55
7.6 What does “simple” mean?	55
7.7 How to deal with anxiety and stress?	56
8 How not to waste your time	59
8.1 The False Peaks of Biology	59
8.2 Don’t Let Architecture Astronauts Scare You	61
8.3 The Curse of Docker	62
8.4 Cloud computing: Amazon Compute Cloud	63
8.5 Large scale data analysis platforms	64
8.6 The New York Times Test	65
8.7 Designed by a committee: Common Workflow Language (CWL)	65
8.8 Time to pay the Iron Price	67
II Installation	69
9 How to set up your computer	71
9.1 How do I set up my computer?	71
9.2 Is this going to be difficult?	72
9.3 How do I prepare my computer?	72
9.4 How do I initialize my terminal?	73
10 How to install conda	74
10.1 What are environments?	74
10.2 How do I install conda?	75
10.3 Step-by-step instruction for installing conda	75
10.4 How to update conda	76
10.5 What is bioconda ?	76
11 How to install software	78
11.1 Installation overview	78
11.2 How to create a bioinformatics environment	78

CONTENTS	5
-----------------	----------

11.3 How to activate and install bioinformatics software	79
11.4 How do I check that Entrez Direct works?	79
11.5 How do I verify that all other programs work?	80
11.6 How do I fix installation problems?	81
11.7 How do I report installation problems?	81
11.8 How do I use <code>conda</code> in general?	82
11.9 How do I install a new tool?	82
11.10 How do I upgrade a tool?	83
11.11 Do the automatically installed tools always work?	83
11.12 How do I update <code>conda</code> ?	84
11.13 How should I set up my file structure?	84
11.14 What to do if I get stuck?	85
11.15 Why is installation so tedious?	85
12 Choose and install a text editor	87
12.1 What features should my text editor have?	87
12.2 Viewing whitespace	88
12.3 Super annoying behaviors	88
12.4 Which text editor to choose?	89
12.5 Watch the line endings on Windows!	90
III UNIX COMMAND LINE	91
13 Introduction to Unix	93
13.1 What is the command line?	93
13.2 What does the command line look like?	94
13.3 What are the advantages of the command line?	94
13.4 What are the disadvantages of the command line?	94
13.5 Is knowing the command line necessary?	95
13.6 Is the command line hard to learn?	95
13.7 Do other people also make many mistakes?	95
13.8 How much Unix do I need to know to be able to progress?	96
13.9 How do I access the command line?	96
13.10 What is a shell?	96
13.11 What is the best way to learn Unix?	99
13.12 How do I troubleshoot errors?	99
13.13 Where can I learn more about the shell?	100

14 The Unix bootcamp	101
14.1 Introduction	101
14.2 Why Unix?	101
14.3 Typeset Conventions	102
14.4 1. The Terminal	102
14.5 2. Your first Unix command	104
14.6 3: The Unix tree	105
14.7 4: Finding out where you are	106
14.8 5: Making new directories	106
14.9 6: Getting from ‘A’ to ‘B’	107
14.107: The root directory	108
14.118: Navigating upwards in the Unix filesystem	109
14.129: Absolute and relative paths	109
14.1310: Finding your way back home	110
14.1411: Making the <code>ls</code> command more useful	111
14.1512: Man pages	112
14.1613: Removing directories	113
14.1714: Using tab completion	113
14.1815: Creating empty files with the <code>touch</code> command	114
14.1916: Moving files	114
14.2017: Renaming files	115
14.2118: Moving directories	116
14.2219: Removing files	116
14.2320: Copying files	117
14.2421: Copying directories	118
14.2522: Viewing files with <code>less</code> (or more)	118
14.2623: Viewing files with <code>cat</code>	119
14.2724: Counting characters in a file	120
14.2825: Editing small text files with <code>nano</code>	120
14.2926: The <code>\$PATH</code> environment variable	121
14.3027: Matching lines in files with <code>grep</code>	122
14.3128: Combining Unix commands with pipes	124
14.32Miscellaneous Unix power commands	124
15 Data analysis with Unix	126
15.1 What directory should I use?	126
15.2 Where are we getting the data from?	127
15.3 How do I obtain a data file that is online?	128

15.4 How many feature types are in this data?	135
15.5 The single most useful Unix pattern	136
15.6 One-liners	137
16 Using Makefiles	138
16.1 What is a Makefile?	138
16.2 Why doesn't my Makefile work?	139
16.3 Can I use bash shell constructs in a Makefile?	140
16.4 Why does my Makefile print every command?	140
16.5 Why does my Makefile stop on an error?	140
16.6 Why is the first action executed?	141
16.7 Is there more to Makefiles?	141
16.8 Are there are alternatives to Makefiles?	142
17 Data compression	143
17.1 Quick summary	143
17.2 What is a compressed format?	144
17.3 What objects may be compressed?	144
17.4 What are some common compression formats?	145
17.5 Is there a bioinformatics-specific compression format?	145
17.6 How do I compress or uncompress a file?	146
17.7 How do I compress or uncompress multiple files?	146
17.8 What is a tarbomb?	148
17.9 How do we use tar again?	149
IV DATA SOURCES	150
18 What is data?	152
18.1 So what is data?	152
18.2 Essential properties of data	153
18.3 How life scientists think about bioinformatics	153
18.4 What bioinformatics is in reality	154
18.5 What is the state of data in bioinformatics?	154
18.6 What kind of problems does bioinformatics data have?	155
18.7 How complete is the data that will I obtain?	158
18.8 Final thoughts on data	159

19 Biological data sources	160
19.1 Where is biomedical data stored?	160
19.2 What are the major DNA data repositories?	161
19.3 What kind of other data sources are there?	162
19.4 Is there a list of “all” resources?	164
19.5 What’s in a name?	164
19.6 Project systematic names	165
20 Common data types	167
20.1 A Quick look at the GENBANK format.	167
20.2 A quick look at the FASTQ format	169
20.3 A quick look at the GFF/GTF/BED formats	170
20.4 A quick look at the SAM/BAM formats	170
20.5 Can I convert between formats.	171
20.6 What is reference data?	172
20.7 What are genomic builds?	172
20.8 Is there a list of “all” resources?	173
20.9 Should download data first or get data on-demand?	173
21 Human and mouse genomes	175
21.1 How many genomic builds does the human genome have?	175
21.2 Why is the genomic build hg19 still in use?	176
21.3 Should we use the old version of a genome?	176
21.4 How do we transfer genomic coordinates between builds?	177
21.5 Human gene naming	178
21.6 Which human data source should I trust?	179
21.7 Which human/mouse genome data should I be using?	179
21.8 Is there a better resource for human annotations?	180
21.9 How do I access NCBI RefSeq and GenBank?	180
21.10 What can I get from ENSEMBL?	181
21.11 A working strategy for finding reference information	181
22 Automating access to NCBI	183
22.1 Note	183
22.2 What is Entrez?	184
22.3 How is Entrez pronounced?	184
22.4 How do we automate access to Entrez?	184
22.5 How is data organized in NCBI?	185

CONTENTS	9
----------	---

22.6 How do I use Entrez E-utils web API?	186
22.7 How do I use Entrez Direct?	186
22.8 How do we search with Entrez Direct?	187
22.9 How to do more work with Entrez Direct?	188
23 Entrez Direct by example	189
23.1 How do I use <code>efetch</code> ?	189
23.2 How do I use <code>esearch</code> to obtain project related data?	190
23.3 How do I get run information on a project?	191
23.4 How do I get even more information on a project?	191
23.5 How do I extract taxonomy information?	192
V DATA FORMATS	193
24 Introduction to data formats	195
24.1 What is a data format?	195
24.2 Should I re-format (transform) my data?	195
24.3 When to re-format (transform) data?	198
25 The GenBank format	199
25.1 What is the GenBank format?	199
25.2 When do we use the GenBank format?	200
25.3 What is RefSeq?	200
25.4 How are RefSeq sequences named?	202
26 The FASTA format	204
26.1 What is the FASTA format?	204
26.2 Are there problems with this format?	205
26.3 Is there more information in FASTA headers?	206
26.4 Is there more information in the FASTA sequences?	207
26.5 Where do I get a fasta file?	208
27 The FASTQ format	209
27.1 What is the FASTQ format?	209
27.2 How to recognize FASTQ qualities by eye	211
27.3 Are there different versions of the FASTQ encoding?	211
27.4 Is there more information in FASTQ headers?	212
27.5 What is a Phred score?	213

27.6 How do I convert FASTQ quality codes at the command line?	213
27.7 Closing thoughts on FASTQ	214
28 Advanced FASTQ processing	216
28.1 Example data	217
28.2 How to produce an overview of FASTQ files?	217
28.3 How do I get the GC content?	218
28.4 How do I get the percentage for custom bases?	218
28.5 How to extract a subset of sequences with name/ID list file?	219
28.6 How do I find FASTA/Q sequences containing degenerate bases and locate them?	219
28.7 How do I remove FASTA/Q records with duplicated sequences?	220
28.8 How do I locate motif/subsequence/enzyme digest sites in FASTA/Q sequence?	221
28.9 How do I sort a huge number of FASTA sequences by length?	221
28.10 How do I split FASTA sequences according to information in the header?	222
28.11 How do I search and replace within a FASTA header using character strings from a text file?	223
28.12 How do I extract paired reads from two paired-end reads files?	224
28.13 How to concatenate two FASTA sequences in to one?	225
VI VISUALIZING DATA	228
29 Visualizing biological data	230
29.1 What are the challenges of visualization?	230
29.2 What is a genome browser?	231
29.3 Why are default browser screens so complicated?	231
29.4 What types of data may be visualized in a genome browser?	232
29.5 How do I interpret glyphs?	233
29.6 Which standalone genome browsers should I use?	234
29.7 What about online genome browsers?	234
30 Using the Integrative Genomics Viewer	235
30.1 What is IGV?	236
30.2 How to run IGV on Mac and Linux?	236
30.3 How to run IGV on Windows Bash?	236

CONTENTS	11
-----------------	-----------

30.4 What does the IGV interface look like?	237
30.5 What data does IGV come with?	237
30.6 How do I create a custom genome in IGV?	238

VII SEQUENCE ONTOLOGY	240
------------------------------	------------

31 What do the words mean?	242
31.1 Why is the ontology necessary?	243
31.2 Are there other ontologies?	243
31.3 Who names the genes?	244
31.4 What will our data tell us?	244

32 Sequence ontology	246
32.1 What is the Sequence Ontology (SO)?	246
32.2 Where do I see Sequence Ontology (SO) terms used?	246
32.3 How do I access the Sequence Ontology browser?	247
32.4 Does all sequencing data obey the rules of SO?	250
32.5 How are the SO relationships defined?	251
32.6 Will I need to access the SO data directly?	251
32.7 How can I investigate the SO data?	251
32.8 How many Sequence Ontology terms are there?	252
32.9 How can I quickly search the Sequence Ontology?	252
32.10 How to search for other information?	253

VIII GENE ONTOLOGY	255
---------------------------	------------

33 Gene ontology	257
33.1 What is the Gene Ontology (GO)?	257
33.2 How is the GO designed?	257
33.3 What kind of properties do annotated gene products have ? .	259
33.4 Where can access the GO online?	259
33.5 How are GO terms organized?	260
33.6 Where can the visualize GO terms online?	262

34 Understanding the GO data	265
34.1 What format is the GO data in?	265

34.2 What does a GO term file contain?	266
34.3 What is a GO association file?	266
34.4 Where can I find the association files for different organisms? .	267
34.5 How to get the human gene association file?	267
34.6 What format does the GO association file have?	268
34.7 Do the association files represent all of the accumulated biological knowledge?	269
34.8 What kind of properties does the GO data have?	269
34.9 What are the most annotated human genes and proteins? . .	270
34.10 What are the ten most highly annotated genes in the GO dataset?	272
34.11 Do the GO annotations change?	273
34.12 How complete is the GO?	273
35 Functional analysis	276
35.1 The sorry state of data categorization	276
35.2 What is a functional analysis?	277
35.3 What is an Over-Representation Analysis (ORA)?	278
35.4 Are there different ways to compute ORA analyses?	279
35.5 What are problems with the ORA analysis?	281
35.6 Why do we still use the ORA analysis?	281
35.7 What is a Functional Class Scoring (FCS)?	282
35.8 Should I trust the results of functional analyses?	283
36 Gene set enrichment	284
36.1 What is a gene set enrichment analysis?	284
36.2 What tools are used to perform enrichment analysis?	284
36.3 Will different tools produce different results?	286
36.4 How do I perform a gene set enrichment analysis?	287
37 Using the AgriGO server	289
37.1 Authors note	289
37.2 How to use AgriGO	289
37.3 What is the Singular Enrichment Analysis (SEA) in AgriGo? .	290
37.4 How to use SEA?	290
37.5 What are the outputs of AgriGO?	292
37.6 How do I prepare a custom annotation for AgriGO?	293

CONTENTS	13
----------	----

38 Using the g:Profiler server	296
38.1 What is the g:Profiler?	296
38.2 What is a standout feature of the g:Profiler?	297
38.3 What functionality does the g:Profile have?	297
38.4 How to use g:profiler at the command line	300
39 Using the DAVID server	303
39.1 Authors note	303
39.2 What is DAVID?	304
39.3 What does DAVID do?	304
39.4 What are the different steps in a DAVID analysis?	305
39.5 How do I start an analysis with DAVID?	305
39.6 What is the Functional Annotation Tool?	307
39.7 What is the Functional Annotation Summary?	308
39.8 How do I explore the Functional Annotation Summary?	309
39.9 What is a Functional Annotation Chart ?	309
39.10 What is Functional Annotation Clustering?	309
39.11 What is a Functional Annotation Table?	311
39.12 What is the Gene Functional Classification Tool?	313
39.13 What is an EASE Score?	313
39.14 What is the Kappa statistic?	314
39.15 What does the Gene ID Conversion Tool do?	314
39.16 What are some pros and cons of DAVID?	315
40 Using the goatools package	317
40.1 Install goatools	317
40.2 Plot GO terms	318
40.3 Finding enrichment with goatools	323
41 Using the ErmineJ tool	327
41.1 Authors note	327
41.2 How to use ErmineJ	328
41.3 What type of analyses does ErmineJ offer?	328
41.4 How is ErmineJ different from other options?	328
41.5 What are the input files in ErmineJ?	329
41.6 What are the annotation files?	329
41.7 What is a gene score file?	330
41.8 What is the expression data?	331

41.9 How do I start an analysis in ErmineJ?	331
41.10 What is an Over-Representation Analysis (ORA)?	332
41.11 How do I run an ORA?	333
41.12 What are the results of an ORA analysis?	334
41.13 What is a multi-functionality score?	336
41.14 Is multi-functionality good or bad?	336
41.15 What is Gene Score Resampling (GSR)?	337
41.16 What is Correlation Analysis in ErmineJ?	338
42 Student reports on functional enrichment	339
42.1 David Northover is amused	340
42.2 Kristin Passero notes that the presentation matters	341
42.3 Jeremy Held gets somewhat different answers but all make sense	342
42.4 Kelly Gomez Campo works on her thesis	344
42.5 Tomas Lopez Londono finds that even randomly selected genes may be enriched	345
IX REPRODUCIBILITY	348
43 Scientific reproducibility	350
43.1 What does the word “reproducible research” mean?	350
43.2 What is the “red herring” of reproducibility?	351
43.3 Is science really facing a reproducibility crisis?	352
43.4 How do scientists attempt to define what reproducibility is?	352
43.5 So what does reproducibility mean?	353
43.6 How difficult is to re-analyze data?	353
43.7 What is the best strategy to reproduce results?	354
43.8 Are the challenges of data reproducibility recognized?	355
44 Redo: Genomic surveillance elucidates Ebola virus origin	357
44.1 Quick links	357
44.2 How to get the information for the Ebola paper?	358
44.3 Is it possible to access the data for this analysis?	359
44.4 Where can we find out more about this dataset?	359
44.5 How do we download results for this paper?	361
45 Redo: Zika virus targets human cortical neural precursors	363

CONTENTS	15
45.1 Quick links:	363
45.2 How to get the information for the Zika paper?	364
45.3 How do we find the data for this paper?	364
45.4 What chapters cover the Zika data?	367
46 Redo: A synthetic-diploid benchmark for accurate variant-calling evaluation	368
46.1 Quick links:	368
46.2 Why are we reproducing this paper?	368
46.3 Will the best bioinformaticians on the planet produce reproducible analyses?	369
46.4 What problem does the paper attempt to solve?	369
46.5 Where is the data?	370
46.6 Where is the code?	371
46.7 Is the analysis reproducible?	371
46.8 The gods must be crazy	373
47 Redo: Explore the genome of a paleolithic-era archaic human	375
X SEQUENCING INSTRUMENTS	377
48 Sequencing instruments	379
48.1 Is there a document that compares sequencing instruments?	379
48.2 What is in a name?	380
48.3 What type of sequencing instruments are in use?	380
48.4 What are obsolete sequencing instruments I should know about?	381
48.5 How accurate are sequencing instruments?	382
48.6 How do sequencing instruments work?	382
48.7 Can reads be in different orientations?	385
48.8 What is paired-end sequencing?	385
48.9 What are the advantages and disadvantages of paired-end sequencing?	386
48.10 What is mate-pair sequencing?	387
49 Illumina sequencers	388
49.1 What types of sequencers does Illumina manufacture?	388
50 PacBio sequencers	391

50.1 What resources are available for PacBio data analysis?	391
50.2 What is an SMRT cell?	392
50.3 What is a Zero Mode Waveguide (ZMW)?	392
50.4 What is a subread?	392
50.5 How many subreads are produced from a template?	392
50.6 What is a Circular Consensus Sequence (CCS) read?	393
50.7 What is the output of a PacBio run?	393
50.8 What other information is encoded in the BAM file?	394
50.9 What SAM headers does the instrument set?	394
50.10 How is the BAM file formatted?	394
50.11 What SAM tags are added to the reads?	394
50.12 How do I get subreads from a single ZMW?	395
50.13 Why would I want to split a BAM file by ZMWs?	395
50.14 How do I get the consensus sequence from the subreads?	395
50.15 What is the per-base quality of PacBio reads?	396
50.16 How good are the consensus corrected reads?	397
51 Minion sequencers	398
51.1 What is MinION?	398
51.2 What is MinKNOW?	399
51.3 What is the Metrichor Agent?	399
51.4 What is the output of a MinION run?	399
51.5 What do ‘pass’ or ‘fail’ subfolders contain?	400
51.6 What is different between 1D and 2D reads?	400
51.7 How do I extract data from a FAST5 file?	401
51.8 What is HDFView?	401
51.9 How can I extract data from a FAST5 file using poretools? . .	402
52 Sequencing data preparation	403
52.1 From sample submission to receiving sequence data	403
52.2 A recap of high-throughput sequencing	403
52.3 Where can I find illustrated summaries of sequencing technologies?	404
52.4 How do I select proper sample identifiers?	405
52.5 How to choose the right sequencing method?	405
52.6 Why is a submission of sample metadata critical?	406
52.7 Why is sample/library QC essential?	407

52.8 What can large-scale projects do to ensure sample data integrity?	407
52.9 What happens during a sequencing run?	407
52.10 What does an original Illumina sequence data folder contain?	408
52.11 How does one get “finished” data from “raw” sequence data?	408
52.12 What should I expect to get from a sequencing provider?	409
52.13 How is the sequencing data delivered?	410
52.14 What should I do with the raw data?	410
52.15 Where can I download Illumina software?	411
XI SEQUENCING DATA	412
53 Sequencing coverage	414
53.1 How much data do I need?	414
53.2 What are typical coverages for DNA sequencing?	414
53.3 What are typical coverages for RNA sequencing?	415
53.4 How is genome coverage computed?	415
53.5 How many bases have not been sequenced?	416
53.6 Do theoretical coverages describe reality?	417
54 Accessing the Short Read Archive (SRA)	419
54.1 What is the Short Read Archive (SRA)?	419
54.2 What are the SRA naming schemes?	419
54.3 How do we download data from SRA?	420
54.4 Are there alternatives to the SRA?	420
54.5 Where is the SRA documentation?	420
54.6 How does the sratoolkit work?	421
54.7 Is it possible to download the SRA files from a server?	422
54.8 Where do downloads go?	422
54.9 What is a “spot” in SRA?	423
54.10 How do we get information on the run?	423
55 Automating access to SRA	424
55.1 How to automate downloading multiple SRA runs?	424
55.2 How to filter for various metadata?	427
55.3 Is there even more metadata?	427
55.4 How do I process the docsum format?	428

56 How much data in the SRA?	429
56.1 How many sequencing runs are in the SRA?	430
56.2 How do we extract columns from a comma separated file?	431
56.3 How many sequencing runs per organism?	432
56.4 How many sequencing runs for each sequencing platform?	433
56.5 How many runs per sequencing instrument model?	434
XII QUALITY CONTROL	435
57 Visualizing sequencing data quality	437
57.1 What part of the FASTQ file gets visualized?	437
57.2 Are FASTQ errors values accurate?	438
57.3 What is the FastQC tool?	438
57.4 How does FastQC work?	439
57.5 How do we run FastQC?	439
57.6 Should I be worried about the “stoplight” symbols?	440
57.7 What does the sequence quality visualization tell us?	440
57.8 What does the sequence length histogram show?	442
57.9 What does the sequence quality histogram tell us?	443
57.10 What is the next step after visualizing the quality?	443
58 Sequencing quality control	445
58.1 What is quality control?	445
58.2 How critical is quality control?	445
58.3 What can go wrong with the data?	446
58.4 When do we perform quality control?	446
58.5 How do we perform quality control?	447
58.6 How reliable are QC tools?	447
58.7 Can quality control introduce new problems?	448
58.8 Is there a list of QC tools?	448
58.9 How does read quality trimming work?	450
59 Sequencing adapter trimming	452
59.1 What are sequencing adapters?	452
59.2 How do I visualize the sequencing adapters?	452
59.3 Can we customize the adapter detection?	453
59.4 Why do we need to trim adapters?	454

59.5 How do we trim adapters?	455
59.6 How do we perform multiple steps at once?	456
59.7 Do we have to remove adapters from data?	456
59.8 How do I cut a different adapter?	457
60 Sequence duplication	458
60.1 What is sequence duplication?	458
60.2 How do we detect sequence duplication?	458
60.3 What is the primary concern with duplicates?	459
60.4 Should we remove duplicates?	459
60.5 What does the FastQC duplicate plot mean?	460
60.6 How do I remove duplicates?	461
61 Advanced quality control	462
61.1 Are there different types of quality control?	462
61.2 How to combine the results into a single report.	463
61.3 Why did my <code>multiqc</code> fail with an error?	465
61.4 What else is <code>multiqc</code> capable of?	466
61.5 Would we ever want to toss out some of the good quality data?	466
61.6 Can I combine reads into a single sequence?	467
61.7 When should I merge the reads?	467
61.8 How do I merge reads?	467
61.9 How well do mergers work?	470
61.10 Is there anything newer than <code>fastqc</code> ?	471
61.11 What is error correction?	472
61.12 How do we correct errors?	473
XIII WRITING SCRIPTS	474
62 The Art of Bioinformatics Scripting	476
XIV PATTERNS	477
63 Sequence patterns	479
63.1 What is a sequence pattern?	479
63.2 Can adapters be identified by a pattern?	479

63.3 How can I search genomic sequences for patterns?	480
64 Regular expressions	482
64.1 What are regular expressions?	482
64.2 What are some challenges I may face when using regular expressions?	484
65 Sequence k-mers	485
65.1 What is a k-mer?	485
65.2 What are k-mers good for?	485
65.3 Should I use the k-mer report produced by FastQC?	486
XV ALIGNMENTS	487
66 Introduction to alignments	489
66.1 What is a sequence alignment?	489
66.2 What are alignments used for?	490
66.3 What governs an alignment?	490
66.4 The inconvenient truth about alignments	490
66.5 How are alignments displayed?	491
66.6 Does it matter which sequence is on top?	492
66.7 How are alignments generated?	492
66.8 How does alignment scoring work?	493
66.9 How do I choose the scores?	494
66.10 What kind of scoring matrices are there?	495
66.11 What other properties do scoring matrices have?	495
66.12 Are there other ways to quantify alignments?	496
66.13 What is a CIGAR string?	496
66.14 Where can I learn more about alignments?	498
67 Global and local alignments	499
67.1 Install the helper scripts	499
67.2 What is a global alignment?	500
67.3 What is a local alignment?	501
67.4 Why are scoring values integer numbers?	503
67.5 What is a semi-global alignment?	503
68 Misleading alignments	505

68.1 Will alignments always indicate the “correct” variation?	505
XVI BLAST	508
69 BLAST: Basic Local Alignment Search Tool	510
69.1 What is BLAST?	510
69.2 What are the BLAST tools?	511
69.3 What are the fundamental concepts of BLAST?	511
69.4 Can BLAST be used for pairwise alignments?	512
69.5 How do I use BLAST?	513
69.6 What are the blast tools?	516
69.7 What is the Blast terminology?	516
69.8 What is an E-Value	517
70 BLAST use cases	519
70.1 How do I build custom blast databases?	519
70.2 How to run BLAST queries?	520
70.3 How do I format the output differently?	521
70.4 What are blast tasks?	522
70.5 Will blast find all alignments?	523
71 BLAST databases	526
71.1 How do we make a BLAST database?	526
71.2 Can we download BLAST databases?	527
71.3 Can we automate the download of BLAST databases?	527
71.4 What information can be obtained from a BLAST database? .	528
71.5 How do I reformat sequences in a BLAST database?	529
71.6 How do I extract a specific entry from a database?	530
71.7 How do I process all entries in a database?	530
XVII SHORT READ ALIGNMENT	532
72 Short read aligners	534
72.1 How are short reads different from long reads?	534
72.2 What are short read aligners?	535
72.3 How do short read aligners work?	535

72.4 What are the limitations of short read aligners?	536
72.5 What is read mapping?	536
72.6 How do I distinguish between a mapper and an aligner?	537
72.7 How do we pick the best short read aligner?	538
72.8 What features do we look for in a short read aligner?	538
73 The bwa aligner	540
73.1 How does bwa work?	541
73.2 How do I use bwa?	541
73.3 How do I find help on bwa?	543
74 The bowtie aligner	545
74.1 How does bowtie work?	545
74.2 How do I build an index with bowtie?	546
74.3 How do I align with bowtie?	546
75 How do I compare aligners?	549
75.1 How can I tell which aligner is “better”?	549
75.2 How do I choose the right aligner?	551
76 Multiple sequence alignments	553
76.1 How do I align more than two sequences?	553
76.2 What programs can be used to align multiple sequences?	555
XVIII SAM/BAM Format	556
77 The SAM/BAM/CRAM formats	558
77.1 What is a SAM file?	558
77.2 What is a BAM file?	559
77.3 What is the CRAM format?	559
77.4 Will the CRAM format replace BAM?	560
77.5 Is it SAM, BAM or CRAM now?	560
77.6 What are SAM files used for?	560
77.7 Is the SAM an optimal format?	561
77.8 What information is stored in a SAM file?	561
77.9 Why do we need the SAM format?	562
77.10 How important is to understand the details of the SAM format?	562
77.11 How do I create SAM/BAM files?	562

CONTENTS	23
77.12 How do I create a CRAM file?	563
77.13 Can “unaligned” data be stored in a SAM file?	563
78 How to make a BAM file	566
79 The SAM format explained	571
79.1 What is a SAM file?	571
79.2 Where do I get a BAM file?	571
79.3 What is the SAM header?	572
79.4 What is stored in the SAM alignment section?	574
79.5 Column 1: Query Name (QNAME)	575
79.6 Column 2: FLAG	575
79.7 What are primary, secondary and chimeric (supplementary) alignments	579
79.8 Columns 3-4: Reference Name (RNAME) and Position (POS)	580
79.9 Column 5-6: Mapping Quality (MAPQ) and Compact Idiosyncratic Gapped Alignment Representation (CIGAR) . . .	581
79.10 Columns 7-9: RNEXT, PNEXT, TLEN	582
79.11 Columns 10-11: SEQ and QUAL	584
79.12 Optional Fields: Columns 12, 13 and on	585
79.13 What do the SAM tags mean?	585
80 Working with BAM files	587
80.1 How can I better understand the FLAGS?	587
80.2 Where do I get a BAM file?	588
80.3 How can I extract a section of the BAM file?	589
80.4 How can I select from or filter data from BAM files?	590
80.5 How do I valid (mapped) alignments?	590
80.6 How do I select forward or reverse alignments?	591
80.7 How to separate alignments into a new file?	593
80.8 How to get an overview of the alignments in a BAM file?	593
80.9 What is a proper-pair?	595
80.10 How do I use flags?	596
80.11 How do I combine multiple flags?	597
80.12 How should I interpret seemingly impossible alignments?	598
80.13 How do I filter on mapping quality?	599
80.14 How can I find out the depth of coverage?	599
80.15 What is a “SECONDARY” alignment?	601

80.16 What is a “SUPPLEMENTARY” alignment?	601
80.17 What is a “PRIMARY” or “REPRESENTATIVE” alignment?	603
81 The SAM reference sheet	604
81.1 What are the required SAM columns?	604
81.2 What do the flags mean?	605
81.3 How to build compound flags?	605
81.4 What are SAM tags?	606
81.5 What do the SAM header tags mean?	606
81.6 What do the SAM alignment tags mean?	607
81.7 What are BWA aligner specific tags?	609
82 Sequence variation in SAM files	611
82.1 How to mutate a sequence from the command line?	612
82.2 How do I reconstruct alignment from CIGAR and MD strings?	613
82.3 What will deletions look like in a SAM format?	615
82.4 What will insertions look like in a SAM format?	615
82.5 How long is an alignment?	616
82.6 How to get the alignment lengths with a tool?	616
82.7 How to visualize a SAM file as pairwise alignment?	617
XIX Genomic Variation	618
83 An introduction to genomic variation	620
83.1 What is a variant?	620
83.2 How do we classify variants?	621
83.3 What are SNPs?	621
83.4 Is the SNP term used consistently?	622
83.5 Do SNPs cause disease?	623
83.6 What is a genotype?	623
83.7 What is a haplotype?	623
84 Online Mendelian Inheritance of Man	625
84.1 How does OMIM work?	625
84.2 Can the OMIM data be obtained?	626
84.3 What format is the OMIM data in?	626
84.4 What terms and concepts does OMIM know about?	627

84.5 How many phenotypes of Mendelian inheritance?	628
84.6 How many phenotypes can be connected to genes?	629
84.7 How many unique pairs of gene and phenotypes are there?	629
84.8 What genes have the most annotated phenotypes?	631
85 Why simulating reads is a good idea	633
85.1 What types of data simulators exist?	633
85.2 Are the simulations realistic?	634
85.3 What is <code>wgsim</code> and <code>dwgSim</code> ?	634
85.4 How do I simulate sequencing data with <code>wgsim</code> ?	635
85.5 How do I simulate sequencing data with <code>dwgSim</code> ?	635
85.6 How do I mutate a sequence with <code>msbar</code> ?	636
85.7 How do I simulate mutations with <code>biosed</code> ?	637
85.8 How do I simulate reads with <code>readsim</code> ?	638
85.9 How do I simulate reads with <code>art</code> ?	638
86 Visualizing large scale genomic variations	640
86.1 The reference genome is not “real”	640
86.2 Which alignments are informative?	641
86.3 What would perfect data look like?	642
86.4 What would realistic and useful data look like?	643
86.5 What would a deletion from the genome look like?	644
86.6 What would copy number variation look like?	645
86.7 How can you tell when parts of the genome are swapped?	646
86.8 What if the genome contains new regions?	647
86.9 What if we reverse complement parts of the genome?	648
XX Variation Calling	649
87 Introduction to variant calling	651
87.1 What is the process for variant calling?	651
87.2 What is the ploidy?	652
87.3 How many possible alleles will I see?	652
87.4 What is a haplotype?	652
87.5 What is a genotype?	652
87.6 What is the best method to call variants?	653
87.7 What are the most commonly used variant callers?	653

87.8 How are variants represented?	654
87.9 How do I generate VCF files?	654
88 Variant calling example	657
88.1 Which differences should I be looking at?	657
88.2 How do I prepare the reference genome for variant calling?	658
88.3 How do I obtain the sequencing data?	659
88.4 How do I align sequencing data against a reference?	659
88.5 How do I use <code>bcftools</code> to call variants?	659
88.6 How do I use the FreeBayes variant caller?	660
88.7 What is GATK (The Genome Analysis Toolkit)?	660
88.8 How to install GATK?	661
88.9 Are there multiple versions of GATK?	662
88.10 How do I use GATK?	662
88.11 How can I improve the quality of variant calls?	663
88.12 Can I customize the variant calling process?	663
89 Multi-sample variant calling	665
89.1 What is multi-sample variant calling	665
89.2 How do I perform multi-sample variant calling process?	665
89.3 How do I interpret the visualization?	667
90 Variant normalization	668
90.1 What is variant normalization?	668
90.2 Should variant callers report normalized variants by default?	669
90.3 How do I normalize a variant?	669
91 The Variant Call Format (VCF)	670
91.1 What is the VCF format?	670
91.2 How important is it to understand the VCF format?	670
91.3 What is a VCF header?	671
91.4 What is the difference between the <code>INFO</code> and the <code>FORMAT</code> fields?	671
91.5 What do the words mean?	672
91.6 What are VCF records?	673
91.7 How to interpret the <code>FORMAT</code> field?	674
91.8 What is represented in the <code>REF/ALT</code> columns.	674
91.9 How are genotypes described?	675
91.10 What are genotype likelihoods?	676

91.11 What is a BCF file?	677
91.12 What are additional resources?	677
92 Filtering information in VCF files	678
92.1 How can I filter VCF files?	678
92.2 How do I get the example data?	679
92.3 How do I extract information from VCF in a custom format? .	679
92.4 How do I list all samples stored in a VCF file?	680
92.5 How do I extract all variants from a particular region?	680
92.6 How do I extract variants excluding a specific region?	680
92.7 How do I get variants present in all samples?	681
92.8 How do I select INDELS only?	682
92.9 How do I extract per-sample information from a multi-sample VCF file?	682
92.10 How to print genotype (GT) of all samples at each position. .	683
92.11 How do I select specific samples?	683
92.12 How do I exclude specific samples?	683
92.13 How do I get variants for which allele count is above a specific value?	684
92.14 How do I select sites where all samples have homozygous geno- type?	684
92.15 How do I select sites where ‘n’ samples have heterozygous variants?	684
92.16 How do I select variant sites based on quality and read depth? .	685
92.17 How do I merge multiple VCF/BCF files?	685
92.18 How do I find variant sites present in all vcf files?	686
92.19 How do I extract variant sites present in at least 2 files? . .	686
92.20 How do I find variants unique to one sample but absent in all other samples?	686
93 Variant annotation and effect prediction	687
93.1 What is variant annotation?	687
93.2 Are are consequences of “variant effects”?	688
93.3 What kind of effects can variant effect predictors find?	688
93.4 What kind of variant annotators can I use?	689
93.5 How do I use snpEff?	689
93.6 Can I build custom annotations for snpEff?	691
93.7 How do I use the Ensemble Variant Effect Predictor (VEP) .	692

94 Why is variant calling challenging?	694
94.1 Why is it so difficult to call variants?	694
94.2 How do variant calls overcome these challenges?	695
94.3 How to explore the effect of mutations on variant callers?	695
94.4 What happens when the variation is more complicated?	697
XXI RNA-SEQ PRINCIPLES	699
95 Introduction to RNA-Seq analysis	701
95.1 What is RNA-Seq when reduced to its essence?	701
95.2 What is quantifying?	703
95.3 What is RNA-Seq analysis?	703
95.4 Why should I consider RNA-seq (over alternatives, e.g., microarray, qPCR)?	703
95.5 What are the main methods for quantifying RNA-Seq reads?	704
95.6 What is the typical outcome of an RNA-Seq analysis?	705
95.7 What complicates RNA-Seq analysis?	705
95.8 What are gene isoforms?	706
95.9 What kind of splicing events exist?	707
95.10 How does RNA-seq analysis work?	707
95.11 How many replicates do I need?	708
95.12 Will there ever be an optimal RNA-Seq method?	709
95.13 What is the first decision when performing an RNA-Seq analysis?	710
95.14 Should I quantify against a genome or a transcriptome?	710
95.15 What are the steps of an RNA-Seq analysis?	711
95.16 What is a splice-aware aligner?	711
95.17 Which splice-aware aligner should I use?	712
95.18 How do I quantify mRNA abundances?	713
95.19 How do I compare mRNA abundances?	713
96 RNA-Seq terminology	715
96.1 What is a sample?	715
96.2 What is normalization?	715
96.3 What is a library size normalization?	716
96.4 What is the effective length?	716
96.5 What gets counted in a gene level analysis?	716

96.6 What is the RPKM?	717
96.7 What the FPKM is that?	718
96.8 Why are RPKM and FPKM still used?	719
96.9 What is TPM?	719
96.10 What is TMM (edgeR) normalization?	720
96.11 What is DESeq normalization?	720
96.12 Do I always need an advanced statistical analysis?	720
96.13 What is a “spike-in” control?	721
96.14 How should I name samples?	721
97 Statistical analysis in RNA-Seq	724
97.1 Why do statistics play a role in RNA-Seq?	725
97.2 When do I need to make use of statistics?	725
97.3 What kind of questions can we answer with a statistical test?	726
97.4 What types of statistical tests are common?	726
97.5 Do I need to involve a statistician?	727
97.6 What is R?	727
97.7 How do I install R?	728
97.8 Can I also install R from command line?	728
97.9 What is Bioconductor?	729
97.10 What does a p-value mean?	730
97.11 So how do I deal with p-values?	732
97.12 Do I need to compute and discuss p-values?	734
98 Useful R scripts	735
98.1 How can I run RNA-Seq differential expression scripts from the command line?	735
98.2 How to the helper scripts work?	736
98.3 Do I need to run all three scripts?	737
98.4 What is the <code>norm-matrix</code> file?	737
98.5 How can I plot my normalized matrix?	738
99 The RNA-Seq puzzle	740
99.1 How would I even solve this puzzle?	740
99.2 The Pledge	741
99.3 The Turn	742
99.4 The Prestige	742
99.5 How to solve it (a hint)	743

XXII RNA-SEQ EXAMPLE	744
100Understand your data	746
100.1Which publication is reanalyzed?	746
100.2What type of data is included?	746
100.3How do I download the example data?	747
101Alignment based RNA-Seq of control samples	749
101.1What is a spike-in control?	749
101.2How do I align an RNA-seq sample?	751
101.3How do I automate my code for all samples?	752
101.4How to better automate the process?	753
101.5How do I estimate the abundance for a single sample?	755
101.6Are there different ways to count overlaps?	756
101.7How do I estimate abundances for all samples?	756
101.8How do I find differential expression?	757
101.9What does a differential expression file look like?	758
101.10Did our RNA-Seq analysis reproduce the expected outcomes?	759
102Alignment based RNA-Seq of biological data	761
102.1How do I adapt scripts to new data?	761
102.2How do I generate the list of differentially expressed features?	762
102.3What does the differential expression file look like?	763
102.4How do I interpret the results?	763
103Classification based RNA-Seq of control samples	765
103.1What are Kallisto and Salmon?	765
103.2What is the main difference between alignments and classification based RNA-Seq?	766
103.3Where do we start?	766
103.4How do I build a Kallisto index?	766
103.5How do I quantify transcripts with Kallisto?	767
103.6What are the files that kallisto produces?	767
103.7How do I quantify all samples with Kallisto?	768
103.8How to interpret the files?	769
103.9How do I run a differential expression study?	769

CONTENTS	31
XXIII RNA-SEQ ZIKA	771
104Understand the Zika data	773
104.1What data does the project contain?	773
104.2How to obtain the data?	775
105What will the accession number files contain?	777
105.1How much data is there in total?	778
105.2How do I analyze data generated on different sequencing platforms?	778
106Alignment based RNA-Seq of Zika data	779
106.1How to get the reference genomes?	779
106.2What will be done in this section?	780
106.3What are the steps to align the data?	781
106.4How do I generate feature counts?	783
106.5How do I compute differentially expressed genes?	784
106.6How do I visualize the differentially expressed genes?	785
106.7How do I make sense of this data?	785
106.8What is the result of the differential expression analysis?	787
106.9How do the results relate to expected results?	788
107Classification based RNA-Seq of Zika data	790
107.1What are the steps of a classification based RNA-Seq?	790
107.2How do I build a Kallisto index?	791
107.3How do I quantify transcripts with Kallisto?	791
107.4What are the outputs of Kallisto?	793
107.5How do I run kallisto for all the samples?	793
107.6How do I quantify the results?	794
107.7How do I find differentially expressed transcripts?	795
XXIV CHIP-SEQ Analysis	797
108Introduction to ChIP-Seq analysis	799
108.1What is ChIP-Seq?	799
108.2What are the challenges of ChIP-Seq?	800
108.3What the heck is a peak?	800
108.4How are RNA-Seq studies different from ChIP-Seq studies?	801

108.5 What will the data look like?	802
108.6 What does a “peak” represent?	803
108.7 Do peaks represent binding?	804
108.8 How do we determine peaks from sequencing data?	805
108.9 What does a ChIP-Seq data measure?	806
108.10 What type of ChIP-Seq studies are performed?	809
108.11 Does the data need to be compared to control?	809
108.12 Should my ChIP-Seq aligner be able to detect INDELS?	810
108.13 What are the processing steps for ChIP-Seq data?	811
109 Aligning ChIP-Seq data	812
109.1 Which paper is going to be re-analyzed?	812
109.2 How do I obtain the data for project PRJNA306490?	813
109.3 How do I find out more information each sample?	814
109.4 What other information do I need to start a ChIP-Seq analysis?	815
109.5 How do I get the “standard” yeast genome data and annotation?	816
109.6 How do I align the ChIP-Seq data?	817
109.7 Do I need to trim data to borders?	818
109.8 How do I visualize the alignments?	818
109.9 Are there other ways to generate bigwig files?	820
109.10 What is the next step of analyzing the data?	822
110 Chip-Seq peak calling	823
110.1 How do I reanalyze a ChIP-Seq experiment?	823
110.2 Can/should I contact the author of a study?	824
110.3 Should I first summarize my data?	825
110.4 What tools can I use to predict peaks?	826
110.5 How do I refine the peaks?	827
110.6 What else can I do to refine the peaks?	828
110.7 How did the paper call peaks?	829
111 Chip-Seq motifs	830
111.1 Why do scientists consider sequence motifs to be important?	830
111.2 What makes motifs difficult to evaluate?	831
111.3 How do I find known motifs?	831
111.4 How many “degenerate” motifs in the genome?	834
111.5 How do I call motifs?	834
111.6 What is the conclusion on finding motifs?	836

CONTENTS	33
111.7 Have motifs ever misled scientists?	836
XXV Ming Tang's ChIP Seq	838
112Ming Tang's guide to ChIP-Seq analysis	840
112.1 What is ChIP-seq?	840
112.2 What are the processing steps for ChIP-seq data?	841
112.3 What are IgG control and input control for ChIP-seq?	841
112.4 Data sets	842
112.5 How to obtain the data?	842
112.6 How to align ChIP-seq reads?	844
112.7 How do I call peaks?	845
112.8 How do I call super-enhancers?	846
112.9 How do I get a normalized bigWig track for visualizing the raw signal?	847
112.10 How do I put all the steps together?	849
112.11 What are the black-listed regions?	854
112.12 How do I visualize the peaks?	855
113ChIP-Seq Downstream Analysis 1	857
113.1 How do I compare the peak sets?	857
113.2 How do I annotate my peaks?	860
113.3 How do I do pathway enrichment analysis for the peaks?	862
113.4 How do I do motif analysis with the peaks?	864
113.5 How to call differential peaks?	868
114ChIP-Seq Downstream Analysis 2	869
114.1 How do I generate a heatmap with ChIP-seq data?	869
114.2 How do I generate a meta profile plot with ChIP-seq data?	872
114.3 Where can I get public available ChIP-seq data sets?	877
XXVI Sequence Assembly	880
115Introduction to sequence assembly	882
115.1 What does “sequence assembly” mean?	882
115.2 What applications does assembly have?	882

115.3 What are the challenges of assembly?	883
115.4 What is the “white whale” of assembly?	884
115.5 What can be “assembled”?	884
116 Concepts in sequence assembly	886
116.1 How do “assemblers” work?	886
116.2 What makes genome assembly difficult?	886
116.3 Beyond technical limitations what else makes genome assembly difficult?	887
116.4 How good is good enough?	887
116.5 Is data quality control more important?	888
116.6 Does the sequencing platform matter?	888
116.7 What is a hybrid assembly?	888
116.8 Does more data lead to better assembly?	889
116.9 What does a genome assembly rely on?	889
116.10 Why are repetitive regions challenging to assemble?	890
117 Redo: GAGE A critical evaluation of genome assemblies and assembly algorithms.	892
117.1 Where is the supporting material?	894
117.2 How does GAGE differ from the other assembly bake-offs?	894
117.3 How do I get the data for the paper?	894
118 Whole genome assembly	897
118.1 What are the steps of Genome Assembly?	897
118.2 How do we quantify assembly quality?	898
118.3 What type of assembly errors will I get?	898
118.4 What is the N50 statistic?	899
118.5 So what is the N50 statistic?	900
118.6 What are the problems with N50?	901
118.7 So what is your N50?	902
119 How to improve the assembly process	903
119.1 Why are assemblers so sensitive to parameters?	903
119.2 Should I error correct my data?	904
119.3 What are the k-mer sizes?	904
119.4 How to tune the expected coverages?	905

120The lucky bioinformatician's guide to genome assembly	906
120.1How do I run an assembly?	906
120.2Is quality control necessary?	907
120.3Do I need a lot of data?	907
120.4What does it look like when I am getting lucky?	907
120.5How do I get "luckier"?	909
120.6How do I evaluate the assembly?	910
120.7What do I see in this assembly that I wouldn't otherwise? . .	910
120.8How do I get lucky?	912
121How to perform a genome assembly	913
121.1How do I run an assembly?	913
121.2How to evaluate a genome assembly	913
121.3What are the steps of genome assembly?	914
121.4How do I visualize a resulting assembly?	917
121.5Can I improve the assembly?	918
121.6What assembler should I use?	919
121.7What do I do next?	920
XXVII METAGENOMICS	921
122Introduction to metagenomics	923
122.1What is metagenomics?	923
122.2What are the challenges of metagenomics?	924
122.3Do humans host ten times as many bacteria as their cells? .	924
122.4What type of answers does metagenomics produce?	926
122.5What are the metagenomics approaches?	927
122.6What is 16S sequencing?	927
122.7What is whole-genome metagenomics?	928
122.8How many bacteria have been fully sequenced?	928
122.9How many bacteria are unknown?	929
123What is a metagenomics analysis?	931
123.1What type of analyses will I need to perform?	931
123.2What online tools can I use?	931
123.3How can I quantify and compare results?	932

123.4 What command line tools may be used to analyze metagenomics data?	933
123.5 What are typical steps of an analysis?	934
124 Introduction to taxonomies	936
124.1 What is a taxonomy?	936
124.2 What the heck is an OTU?	936
124.3 How many taxonomies are available?	937
124.4 Which taxonomy should I use?	938
124.5 How to view the NCBI taxonomy?	938
124.6 What is the story with the NCBI disclaimer?	938
124.7 How to get the NCBI taxonomy file?	939
124.8 How many taxonomical ranks are there?	940
124.9 How do I search the NCBI taxonomy from the command line?	941
124.10 How do I search the taxonomies?	941
124.11 How do I find the taxid of a taxonomy name?	942
124.12 How do I get the taxid for an accession number?	943
124.13 How do I get the lineage of a taxid?	943
124.14 How do I list all taxids of a given rank?	944
124.15 How many species of viruses are in the current taxonomy?	945
125 How do I obtain sequences based on taxonomies?	946
125.1 How do I get all known bacterial genomes?	946
125.2 Are there other ways to get all bacterial/viral genomes?	947
125.3 How do I set up the BLAST for taxonomy operations?	949
125.4 How do I extract sequences by taxonomy from a BLAST database?	950
125.5 What does the env_nt blast database contain?	950
126 Classifying 16S sequences {#16S-classification}	952
126.1 What data will be analyzed?	952
126.2 Can I classify 16S sequences via alignment?	953
126.3 How do I visualize the 16S classification?	954
126.4 Are there better ways to classify 16S sequences?	954
126.5 How can I visually compare classifications?	956
126.6 Are there other 16S classification methods?	957
126.7 How do I classify multiple samples?	957

126.8 How do I determine statistically relevant differences between the classification counts?	959
127 Classifying whole genome data	962
127.1 How do I evaluate how well a given method works?	962
127.2 How do I obtain the data?	963
127.3 What are the expected abundances of the data?	964
127.4 Are there different kinds of abundances?	965
127.5 What percent of a measurements originate from a given species?	966
127.6 What is the coverage?	966
127.7 How can I tell how many bacteria are in the sample?	967
127.8 How do I match the results of a classifier?	967
127.9 How does Kraken classifier work?	968
127.10 How do I match back the bacteria?	969
127.11 Would the full kraken database be more accurate?	971
127.12 How to use sourmash?	971
127.13 What is binning?	974
127.14 How do I assemble metagenomes?	974
128 Alaska Oil Reservoir Metagenome study	975
128.1 Which data are we going to re-analyze?	975
128.2 How do I get more information on the project?	975
128.3 How do I classify this data?	977
128.4 How many species were present?	977
128.5 Does quality control improve the classification?	978
128.6 Are there other methods to classify samples?	979
128.7 How do I use “sourmash” to understand my data?	980
129 Analyzing the neonatal microbiome	981
129.1 How to classify all the data for the project PRJNA46337?	981
129.2 What is the most common misunderstanding (paradox) of metagenomics?	984
129.3 How reliable are these results?	985
XXVIII Appendix	986
129.4 Do I need to compute and discuss p-values?	988

130Setting up MacOS	989
130.1How do I get started?	989
130.2Change the shell to BASH!	989
130.3What other App Store software is needed?	990
130.4What is Homebrew?	991
130.5Do I need Homebrew?	991
130.6How do I install Homebrew?	991
130.7What software are required?	992
130.8How do I install Java?	992
130.9How do I upgrade software?	993
130.10How do I view my home directory in the Finder?	993
130.11Will the Finder show all files?	995
130.12What is the next step?	995
131Setting up Linux	996
131.1Which distributions of Linux does this book support?	996
131.2What is the first step for Linux?	996
131.3How do I update a Linux based system?	996
131.4What are the required libraries for Linux?	997
131.5What is the next step?	997
132Setting up Windows	998
132.1How do I install Ubuntu Linux on Windows?	998
132.2How do I start Ubuntu Linux?	999
132.3How do I see the Unix filesystem from Windows?	999
132.4Install Java for Windows	1000
132.5What does not work on Windows?	1000
132.6How do I customize the Windows Bash terminal?	1000
132.7How do I finish setting up Ubuntu on Windows?	1001
133Setting up your Bash profile	1002
133.1What is a “shell”?	1002
133.2What is Bash?	1002
133.3What are shell profiles?	1003
133.4Can we have multiple shell profiles?	1003
133.5What’s the best setup for multiple shell profiles?	1004
133.6How can I create the minimal settings?	1004
133.7What should my .bashrc file contain?	1004

CONTENTS	39
-----------------	-----------

133.8 Can I customize the terminal colors and fonts?	1005
133.9 How do I edit my shell profile?	1006
133.10 How do I activate a shell profile?	1006
133.11 Troubleshooting the PATH	1007
134 How do I set up my PATH?	1010
134.1 Solution 1: Use the full program path	1010
134.2 Solution 2: Extend the search PATH to include the installation directory	1011
134.3 Solution 3: Create shortcuts	1012
135 Installing wgsim and dwgsim	1013
135.1 Installing DWGSim	1013
135.2 Testing	1014
135.3 Information on mutations	1014
135.4 Source code installation	1014
136 Jim Kent's bioinformatics utilities	1016
136.1 Installation	1016
136.2 Manual Installation	1017

Part I

Preface

Chapter 1

Welcome to the Biostar Handbook

Last updated on **February 07, 2020**

The Biostar Handbook introduces readers to bioinformatics, the scientific discipline at the intersection of biology, computer science, and statistical data analytics dedicated to the digital processing of genomic information.

The Biostar Handbook has been developed, improved and refined over more than a half decade in a research university setting while used in an accredited Ph.D. level training program. The contents of this book have provided the analytical foundation to thousands of students, many of whom have become full-time bioinformaticians and work at the most innovative companies in the world.

1.1 The Biostar Handbook Collection

The Biostar Handbook has grown huge, it is close to 1000 pages! To manage this complexity we have started reworking the various chapters into independent books that can be studied separately.

- **The Biostar Handbook**¹ - The main introduction to Bioinformatics.

¹<https://www.biostarhandbook.com/>

- **The Art of Bioinformatics Scripting**² - Learn Unix and Bash scripting.
- **RNA-Seq by Example**³ - Learn RNA-Seq data analysis.

Access to all new books and materials is included with your subscription!

1.2 News: Shell casting (January 2020)

The book now includes **shell casts** to demonstrate command line usage. The **shell casts** are not videos! These animations are textual, captured from a terminal exactly as typed, with sharp visibility, will zoom with your screen font sizes, may be paused at any time, and may be selected for copy-paste from the screen! Here is an example:

Shell casts are only visible via the web. PDF and ebooks do not have these animations embedded in them. Animations are a new feature in 2020. As we re-work the book we will be adding **shell casts** into each chapter.

1.3 How to download the book?

The book is available to registered users. The latest versions can be downloaded from:

- Biostar Handbook, 2nd Edition, PDF⁴
- Biostar Handbook, 2nd Edition, eBook⁵

We recommend reading and accessing the book via the website! The web version will always contain the most recent additions and most up-to-date content. A few times a year we send out emails that describe the new additions.

²<https://www.biostarhandbook.com/books/scripting/index.html>

³<https://www.biostarhandbook.com/books/rnaseq/index.html>

⁴biostar-handbook.pdf

⁵biostar-handbook.epub

1.4 Get notified

Want to know when new content is published? Enter your email below:

1.5 Current online courses

Access to all courses is included with the book for the duration of your subscription.



- Course: Learn Bioinformatics the Right Way (Fall 2019)⁷

This course has been launched in September of 2019. The course tracks the 2019 course BMMB 852: Applied Bioinformatics at Penn State. Slides are provided.



- Course: Bioinformatics Data Analysis (Spring 2019)⁹

This course has 18 lectures, video presentations. It is our recommended course that follows the 2nd edition of the Biostar Handbook.



⁶<https://www.biostarhandbook.com/edu/course/5/>

⁷<https://www.biostarhandbook.com/edu/course/5/>

⁸<https://www.biostarhandbook.com/edu/course/4/>

⁹<https://www.biostarhandbook.com/edu/course/4/>

¹⁰<https://www.biostarhandbook.com/edu/course/1/>

- Course: Learn Bioinformatics in 100 hours (2017)¹¹

The course has 30 lectures, slides, assignments and quizzes (no videos). This course was developed using the 1st edition of the Biostar Handbook.



- Course: Bioinformatics Recipes Course (2018)¹³

This course comes with six lectures, video presentations. Some of the materials for this course have been merged into the Bioinformatics Data Analysis (2019)¹⁴ course.



- Course: Python Programming (2018)¹⁶

The course has four lectures with videos, slides, assignments, and quizzes. The course is incomplete and will be rebooted in the fall of 2019 or early spring 2020.

1.6 How was the book developed?

We have been teaching bioinformatics and programming courses to life scientists for many years now. We are also the developers and maintainers of Biostars: Bioinformatics Question and Answer¹⁷ website the leading resource

¹¹<https://www.biostarhandbook.com/edu/course/1/>

¹²<https://www.biostarhandbook.com/edu/course/3/>

¹³<https://www.biostarhandbook.com/edu/course/3/>

¹⁴<https://www.biostarhandbook.com/edu/course/4/>

¹⁵<https://www.biostarhandbook.com/edu/course/2/>

¹⁶<https://www.biostarhandbook.com/edu/course/2/>

¹⁷<https://www.biostars.org>

for helping bioinformatics scientists with their data analysis questions.

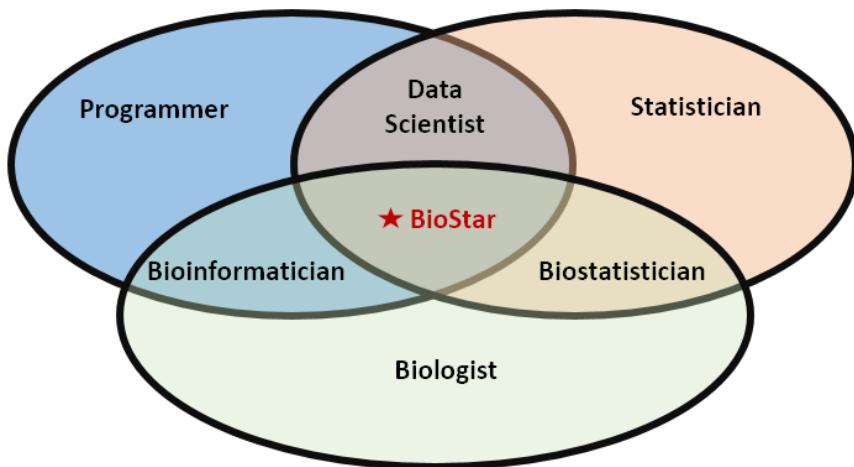
We wrote this book based on these multi-year experiences in training students and interacting with scientists that needed help to complete their analyses. We are uniquely in tune with the challenges and complexities of applying bioinformatics methods to real problems, and we've designed this book to help readers overcome these challenges and go further than they have ever imagined.

1.7 How is this book different?

We use a Question/Answer based format to explain concepts and demonstrate methods simply and practically. We found this to be the most efficient training method for the topics that we cover.

1.8 Who is a Biostar?

It is not a what; it is a who. And it could be you. It is the person whose responsibilities go beyond clear boundaries. This book is for them.



Visualization inspired by a blog post by Anthony Fejes: Who is a bioinformatician?¹⁸

¹⁸<http://blog.fejes.ca/?p=2418>

1.9 Access your account

Logged in users may manage their accounts via the link below.

Access Your Account

You may also change your email or log out via this page.

Chapter 2

About the author

Dr. István Albert is renowned for being at the forefront of the Bioinformatics frontier and pushing the field forward in new directions.

He helped pioneer the Galaxy Bioinformatics Platform¹, an open source web-based tool that allows users to perform and share data-intensive biomedical research. He also developed BooleanNet², a biological system simulation software and the GeneTrack³ software platform that automates and processes large sets of biological data.

Currently, Dr. Albert is the lead software developer, maintainer and administrator of the website, Biostars: Bioinformatics Questions and Answers⁴, the most comprehensive information source in Bioinformatics that is visited by millions of people every year.

Besides being well-known for his scientific innovations⁵ in three different scientific fields: Physics, Computer Science and Biology with publications that gathered over 10 thousand citations, Dr. Albert is a celebrated educator. The courses he develops are consistently highly ranked and revered by students.

In 2014, István was awarded the Paul M. Althouse Outstanding Teaching

¹<https://usegalaxy.org/>

²<https://scfbm.biomedcentral.com/articles/10.1186/1751-0473-3-16>

³<http://bioinformatics.oxfordjournals.org/content/24/10/1305.long>

⁴<https://www.biostars.org/>

⁵https://scholar.google.com/citations?hl=en&user=hordfUUAAAAJ&view_op=list_works&sortby=pubdate

Award⁶ at Penn State for the superb quality of his lectures. The lecture information that has garnered him such high praise is the very same material that you will find in the Biostar Handbook⁷.

Dr. Albert is currently a Research Professor of Bioinformatics⁸ at Pennsylvania State University and the Director of the Bioinformatics Consulting Center at this institute. István established this cutting-edge research facility where he has advised and assisted dozens of brilliant young scientists in their ambitious endeavors. He also leads the Graduate Certificate Program in Applied Bioinformatics⁹, a comprehensive Bioinformatics training course offered through the Penn State World Campus¹⁰ that allows anyone, anywhere, to attain in-depth knowledge of this exciting new field.

The Handbook's main author and editor is Dr. Istvan Albert¹¹.

- Personal website: <https://www.ialbert.me>¹²
- Email: istvan.albert@gmail.com¹³

2.0.1 Contributors

- Aswathy Sebastian, MS, Bioinformatics Analyst, Bioinformatics Consulting Center, Pennsylvania State University, USA
- Reka Albert¹⁴, Ph.D., Professor of Physics and Biology, Department of Physics, Pennsylvania State University, USA
- Jeremy Leipzig¹⁵, MS, Senior Data Integration Analyst, The Children's Hospital of Philadelphia, USA
- Hemant Kelkar¹⁶, Ph.D., Research Associate Professor, Department of

⁶<http://bmb.psu.edu/about/news-articles/istvan-albert-recognized-for-outstanding-teaching>

⁷<https://www.biostarhandbook.com>

⁸<https://www.ialbert.me/>

⁹<http://www.worldcampus.psu.edu/degrees-and-certificates/applied-bioinformatics-certificate/overview>

¹⁰<http://www.worldcampus.psu.edu/>

¹¹<https://www.ialbert.me>

¹²<https://www.ialbert.me>

¹³<mailto:%20istvan.albert@gmail.com>

¹⁴<https://www.ralbert.me/>

¹⁵<http://dbhi.chop.edu/index.php/people/40-people/90-jeremy-leipzig-ms.html>

¹⁶<https://bioinformatics.unc.edu/personnel/>

Genetics and Center for Bioinformatics, University of North Carolina, USA

- Ming Tang¹⁷, Ph.D., Computational biologist in Genomic Medicine Department, MD Anderson Cancer Center, USA
- Ram Srinivasan¹⁸, MS, Bioinformatician, Icahn School of Medicine and Mount Sinai, New York, NY, USA
- Wei Shen¹⁹, Ph.D. student, Third Military Medical University, China.
- Wouter De Coster²⁰, Ph.D. Student, University of Antwerp, Belgium
- Madelaine Gogol²¹, BS, Programmer Analyst, Stowers Institute, Kansas City, MO, USA
- John Dale, retired physician, biohacker, Ham Radio enthusiast(VE7QJD), commercial pilot, Nelson, BC, Canada

2.0.2 Can I contribute to the book?

Join our effort to build a comprehensive and up to date compendium for genomic data analysis.

It is a simple process that works through GitHub via simple, text-based, Markdown files. The only permission we ask from authors are the rights to publish their content under our own terms (see below).

Authors retain re-publishing rights for the material that they are the principal author of and may re-distribute the content that they have created for this book under any other terms of their choosing.

2.0.3 What are the licensing terms?

The Handbook's content is copyrighted material owned by Biostar Genomics LLC²². Republishing any part of the Handbook is permitted only on a limited basis and must follow the terms of the Fair Use²³ policy unless other, prior

¹⁷<https://www.linkedin.com/in/ming-tang-40650014>

¹⁸<https://www.linkedin.com/in/ramakrishnanrs>

¹⁹<http://shenwei.me/>

²⁰<https://www.uantwerpen.be/en/staff/wouter-decoster/>

²¹<https://www.linkedin.com/in/madelainegogol>

²²<https://www.biostargenomics.com>

²³https://en.wikipedia.org/wiki/Fair_use

agreements have been made.

2.0.4 What can I reuse from the book?

If you have purchased the book or have had at any time an account on this site that means that you have license to apply, modify and reuse information from the book as it pertains to your own work and research.

2.0.5 Terms of use

You may not share your login information with others and you may not re-distribute the book, or any part of the book to others.

2.0.6 Citations

In general, when it comes to citations we recommend that you cite the authors of the tools and techniques that are using. We also welcome acknowledgements of the book in your work.

Chapter 3

Why bioinformatics?

In the modern world, it often seems that the age of exploration is over. We have been on the Moon, there are research bases on Antarctica, we can look at atoms. You may ask yourself are there wonders waiting to be discovered? After all, it may seem that all we are left with is to refine and improve on prior observations.

I have good news for you all. The information age has opened up an entirely new domain of science, that of understanding what life itself consists of. We are all beginners in this new science, we are all starting from the same knowledge and principles. Everyone has the chance to make unique and fascinating discoveries using little more than their own computer.

3.1 What is this book about?

The answers to some of the greatest questions of life lie within ourselves. Bioinformatics is a new science created from the marriage of Data Science and Biology. Through this emerging field of study, scientists are able to find and decode hidden information in our very own genes, allowing us to understand what none before us have known.

This book teaches you practical skills that will allow you to enter this fast expanding industry. Beginning with fundamental concepts such as understanding data formats and how analysis is done in general and what conclusions

can be drawn from data, the Handbook eases you into the limitless world of possibilities.

With the help of the book, you will be reproducing results from realistic data analysis scenarios such as genome assembly or gene expression analysis. The methods and tools you will find in the Handbook were refined in world-class research facilities and will allow you to break into this new field and tackle some of the most significant challenges we are facing in the scientific frontier of the 21st century.

3.2 What is covered in the book?

The Handbook is divided into several sections. We cover both the foundations and their applications to realistic data analysis scenarios.

Bioinformatics analysis concepts:

- Data formats and repositories.
- Sequence alignments.
- Data visualization.
- Unix command line usage.

Bioinformatics protocols:

- Genome variation and SNP calling.
- RNA-seq and gene expression analysis
- Genome assembly
- Metagenomics classification
- ChIP-Seq analysis

Software tool usage:

- Using short read aligners
- Using quality control tools
- Manipulating sequence data

The table of contents on the left allows you to jump to the corresponding sections.

Chapter 4

What is bioinformatics?

Bioinformatics is a new, computationally-oriented Life Science domain. Its primary goal is to make sense of the information stored within living organisms. Bioinformatics relies on and combines concepts and approaches from biology, computer science, and data analysis. Bioinformaticians evaluate and define their success primarily in terms of the new insights they produce about biological processes through digitally parsing **genomic** information.

Bioinformatics is a data science that investigates how information is *stored within* and *processed by* living organisms.

4.1 How has bioinformatics changed?

In its early days—perhaps until the beginning of the 2000s—bioinformatics was synonymous with **sequence analysis**. Scientists typically obtained just a few DNA sequences, then analyzed them for various properties. Today, sequence analysis is still central to the work of bioinformaticians, but it has also grown well beyond it.

In the mid-2000s, the so-called *next-generation, high-throughput* sequencing instruments (such as the Illumina HiSeq) made it possible to measure the full genomic content of a cell in a single experimental run. With that, the quantity of data shot up immensely as scientists were able to capture a

snapshot of *everything* that is DNA-related.

These new technologies have transformed bioinformatics into an entirely new field of *data science* that builds on the “classical bioinformatics” to process, investigate, and summarize massive data sets of extraordinary complexity.

4.2 What subfields of bioinformatics exist?

DNA sequencing as initially valued for revealing the DNA content of a cell. It may come as a surprise to many, however, that the most significant promise for the future of bioinformatics might lie in other applications. In general, most bioinformatics problems fall under one of four categories:

1. **Classification:** determining the species composition of a population of organisms
2. **Assembly:** establishing the nucleotide composition of genomes
3. **Resequencing:** identifying mutations and variations in genomes
4. **Quantification:** using DNA sequencing to measure the functional characteristics of a cell

The Human Genome Project¹ fell squarely in the **assembly** category. Since its completion, scientists have assembled the genomes of thousands of others species. The genomes of many millions of species, however, remain entirely unknown.

Studies that attempt to identify changes relative to known genomes fall into the **resequencing** field of study. DNA mutations and variants may cause phenotypic changes like emerging diseases, changing fitness, different survival rates, and many others. For example, there are several ongoing efforts to compile all variants present in the human genome—these efforts would fall into the resequencing category. Thanks to the work of bioinformaticians, massive computing efforts are underway to produce clinically valuable information from the knowledge gained through resequencing.

Living micro-organisms surround us, and we coexist with them in complex collectives that can only survive by maintaining interdependent harmony. **Classifying** these mostly-unknown species of micro-organisms by their genetic material is a fast-growing subfield of bioinformatics.

¹https://en.wikipedia.org/wiki/Human_Genome_Project

Finally, and perhaps most unexpectedly, bioinformatics methods can help us better understand biological processes, like gene expressions, through **quantification**. In these protocols, the sequencing procedures are used to determine the relative abundances of various DNA fragments that were made to correlate with other biological processes.

Over the decades biologists have become experts at manipulating DNA and are now able to co-opt the many naturally-occurring molecular processes to copy, translate, and reproduce DNA molecules and connect these actions to biological processes. Sequencing has opened a new window into this world, new methods and sequence manipulations are continuously discovered. The various methods are typically named as **Something-Seq** for example **RNA-Seq**, **Chip-Seq**, **RAD-Seq** to reflect what mechanism was captured/connected to sequencing. For example, **RNA-Seq** reveals the abundance of RNA by turning it into DNA via reverse transcription. Sequencing this construct allows for simultaneously measuring the expression levels of all genes of a cell. The **RAD-Seq** uses restriction enzymes to cleave DNA at specific locations and only the fragments around these locations are then sequenced. This method produces very high coverage around these sites, thus is suited for population genetics studies.

4.3 Is there a list of functional assays used in bioinformatics?

In the Life Sciences, an **assay** is an investigative procedure used to *assess* or measure the presence, amount, or function of some target (like a DNA fragment). Dr. Lior Pachter², professor of Mathematics at Caltech, maintains a list of “functional genomics” assay technologies on the page called Star-Seq³.

All of these techniques fall into the **quantification** category. Each assay uses DNA sequencing to quantify another measure, and many are examples of connecting DNA abundances to various biological processes.

Notably, the list now contains nearly 100 technologies. Many people, us

²<https://liorpachter.wordpress.com>

³<https://liorpachter.wordpress.com/seq/>

4.3. IS THERE A LIST OF FUNCTIONAL ASSAYS USED IN BIOINFORMATICS?57

included, believe that these applications of sequencing are of greater importance and impact than identifying the base composition of genomes.

Below are some examples of the assay technologies on Dr. Pachter's list:

dsRNA-Seq: Qi Zheng et al., "Genome-Wide Double-Stranded RNA Sequencing Reveals the Functional Significance of Base-Paired RNAs in *Arabidopsis*," *PLoS Genet* 6, no. 9 (September 30, 2010): e1001141, doi:10.1371/journal.pgen.1001141.

FRAG-Seq: Jason G. Underwood et al., "FragSeq: Transcriptome-wide RNA Structure Probing Using High-throughput Sequencing," *Nature Methods* 7, no. 12 (December 2010): 995–1001, doi:10.1038/nmeth.1529.

SHAPE-Seq: (a) Julius B. Lucks et al., "Multiplexed RNA Structure Characterization with Selective 2'-hydroxyl Acylation Analyzed by Primer Extension Sequencing (SHAPE-Seq)," *Proceedings of the National Academy of Sciences* 108, no. 27 (July 5, 2011): 11063–11068, doi:10.1073/pnas.1106501108.

(b) Sharon Aviran et al., "Modeling and Automation of Sequencing-based Characterization of RNA Structure," *Proceedings of the National Academy of Sciences* (June 3, 2011), doi:10.1073/pnas.1106541108.

PARTE-Seq: Yue Wan et al., "Genome-wide Measurement of RNA Folding Energies," *Molecular Cell* 48, no. 2 (October 26, 2012): 169–181, doi:10.1016/j.molcel.2012.08.008.

PARS-Seq: Michael Kertesz et al., "Genome-wide Measurement of RNA Secondary Structure in Yeast," *Nature* 467, no. 7311 (September 2, 2010): 103–107, doi:10.1038/nature09322.

Structure-Seq: Yiliang Ding et al., "In Vivo Genome-wide Profiling of RNA Secondary Structure Reveals Novel Regulatory Features," *Nature* advance online publication (November 24, 2013), doi:10.1038/nature12756.

DMS-Seq: Silvi Rouskin et al., "Genome-wide Probing of RNA Structure Reveals Active Unfolding of mRNA Structures in Vivo," *Nature* advance online publication (December 15, 2013), doi:10.1038/nature12894.

4.4 But what is bioinformatics, *really*?

So now that you know what bioinformatics is all about, you're probably wondering what it's like to practice it day-in-day-out as a bioinformatician. The truth is, it's not easy. Just take a look at this "Biostar Quote of the Day" from Brent Pedersen in *Very Bad Things*⁴:

I've been doing bioinformatics for about 10 years now. I used to joke with a friend of mine that most of our work was converting between file formats. We don't joke about that anymore.

Jokes aside, modern bioinformatics relies heavily on file and data processing. The data sets are large and contain complex interconnected information. A bioinformatician's job is to simplify massive datasets and search them for the information that is relevant for the given study. Essentially, bioinformatics is the art of finding the needle in the haystack.

4.5 Are bioinformaticians data janitors?

Oh, yes. But then, make no mistake, all data scientists are. It is not a unique feature of this particular field.

I used to get worked up about the problems with data, but then I read the New York Times opinion piece: *'For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights'*⁵, where they state:

For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights

[...] Yet far too much handcrafted work — what data scientists call “data wrangling,” “data munging” and “data janitor work” — is still required. Data scientists, according to interviews and expert

⁴<https://www.biostars.org/p/63336/>

⁵http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights.html?_r=0

estimates, spend from 50 percent to 80 percent of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets [...]

You will have problems like that all the time. For example, you'll download a file from an official database like NCBI. It turns to be a large file with 1.5 million entries. But you won't be able to use it, programs that ought to work fine with data of this size crash spectacularly. You'll scratch your head, dive in, few hours later it turns out that one single line, out of the 1.5 million, has the wrong number of fields - one line has the wrong format! Now you'll need to find a way to fix or get rid of that single line - then everything works again.

It helps to be prepared and know what to expect.

4.6 Is creativity required to succeed?

Bioinformatics requires a dynamic, creative approach. Protocols should be viewed as guidelines, not as rules that guarantee success. Following protocols by the letter is usually entirely counterproductive. At best, doing so leads to sub-optimal outcomes; at worst, it can produce misinformation that spells the end of a research project.

Living organisms operate in immense complexity. Bioinformaticians need to recognize this complexity, respond dynamically to variations, and understand when methods and protocols are not suited to a data set. The myriad complexities and challenges of venturing at the frontiers of scientific knowledge always require creativity, sensitivity, and imagination. Bioinformatics is no exception.

Unfortunately, the misconception that bioinformatics is a *procedural* skill that anyone can quickly add to their toolkit rather than a scientific domain in its own right can lead some people to underestimate the value of a bioinformatician's contributions to the success of a project.

As observed in the Nature paper Core services: Reward bioinformaticians⁶

⁶<http://www.nature.com/news/core-services-reward-bioinformaticians-1.17251>

Biological data will continue to pile up unless those who analyze it are recognized as creative collaborators in need of career paths.

Bioinformatics requires multiple skill sets, extensive practice, and familiarity with multiple analytical frameworks. Proper training, a solid foundation and an in-depth understanding of concepts are required of anyone who wishes to develop the particular creativity needed to succeed in this field.

This need for creativity and the necessity for a bioinformatician to think “outside the box” is what this Handbook aims to teach. We don’t just want to list instructions: “do this, do that.” We want to help you establish that robust and reliable foundation that allows you to be creative when (not if) that time comes.

4.7 Are analyses all alike?

Most bioinformatics projects start with a “standardized” plan, like the ones you’ll find in this Handbook. However, these plans are never set in stone. Depending on the types and features of observations and results of analyses, additional tasks inevitably deviate from the original plan to account for variances observed in the data. Frequently, the studies need substantive customization.

Again, as the authors of Core services: Reward bioinformaticians⁷ note the following:

“No project was identical, and we were surprised at how common one-off requests were. There were a few routine procedures that many people wanted, such as finding genes expressed in a disease. But 79% of techniques applied to fewer than 20% of the projects. In other words, most researchers came to the bioinformatics core seeking customized analysis, not a standardized package.”

⁷<http://www.nature.com/news/core-services-reward-bioinformaticians-1.17251>

In summary, almost no two analyses are precisely the same. Also, it is quite common for projects to deviate from the standardized workflow substantially.

4.8 Should life scientists know bioinformatics?

Yes!

The results of bioinformatic analyses are relevant for most areas of study inside the life sciences. Even if a scientist isn't performing the analysis themselves, they need to be familiar with how bioinformatics operates so they can accurately interpret and incorporate the findings of bioinformaticians into their work. All scientists informing their research with bioinformatic insights should understand how it works by studying its principles, methods, and limitations—the majority of which is available for you in this Handbook.

We believe that this book is of great utility even for those that don't plan to run the analysis themselves.

4.9 What type of computer is required?

All tools and methods presented in this book have been tested and will run on all three major operating systems: **MacOS**, **Linux** and **Windows 10**. See the Computer Setup page.



Figure 4.1

For best results, Windows 10 users will need to join the Windows Insider⁸ program (a free service offered by Microsoft) that will allow them to install the newest release of “Windows Subsystem for Linux (WSL)”

⁸<https://insider.windows.com/>

4.10 Is there data with the book?

Yes, we have a separate data site at <http://data.biostarhandbook.com>. Various chapters will refer to content distributed from this site.

4.11 Who is the book for?

The Biostar Handbook provides training and practical instructions for students and scientists interested in data analysis methodologies of genome-related studies. Our goal is to enable readers to perform analyses on data obtained from high throughput DNA sequencing instruments.

All of the Handbook's content is designed to be simple, brief, and geared towards practical application.

4.12 Is bioinformatics hard to learn?

Bioinformatics engages the distinct fields of biology, computer science, and statistical data analysis. Practitioners must navigate the various philosophies, terminologies, and research priorities of these three domains of science while keeping up with the ongoing advances of each.

Its position at the intersection of these fields might make bioinformatics more challenging than other scientific subdisciplines, but it also means that you're exploring the frontiers of scientific knowledge, and few things are more rewarding than that!

4.13 Can I learn bioinformatics from this book?

Yes you can!

The questions and answers in the Handbook have been carefully selected to provide you with steady, progressive, accumulating levels of knowledge.

*4.14. HOW LONG WILL IT TAKE ME TO LEARN BIOINFORMATICS?*63

Think of each question/answer pair as a small, well-defined unit of instruction that builds on the previous ones.

- Reading this book will teach you what bioinformatics is all about.
- Running the code will demonstrate you the skills you need to perform the analyses.

4.14 How long will it take me to learn bioinformatics?

About **100** hours.

Of course, a more accurate answer depends on your background preparation, and each person is different. Prior training in at least one of the three fields that bioinformatics builds upon (biology, computer science, and data analysis) is recommended. The time required to master all skills also depends on how you plan to use them. Solving larger and more complex data problems will require more advanced skills, which need more time to develop fully.

That being said, based on several years of evaluating trainees in the field, we have come to believe that an active student would be able to perform publication quality analyses after dedicating about **100** hours of study.

This is what this book is really about – to help you put those **100** hours to good use.

Chapter 5

Biology for bioinformaticians

Biology is a domain of the Life Sciences, which include, but are not limited to, organic chemistry, ecology, botany, zoology, physiology, and so on. Biology seeks to understand the structures, functions, origins, interactions, and taxonomies of living organisms.

As a science, biology is still relatively immature. It is eclectic, encompassing several scientific domains that are each insufficiently understood and described.

Organisms are immensely complex and always changing. For centuries, we haven't had precise enough tools to measure, describe, or understand the full extent of their complexity. Digital technologies are changing this. Bioinformatics is at the frontier of these changes, and its potential contributions to Biology and the Life Sciences more broadly are quite exciting. As bioinformatics and other innovative methodologies advance, we expect Life Sciences to mature and develop rich, accurate vocabularies and models to understand and describe living organisms.

For these reasons, current concepts and definitions in the Life Sciences are still just approximations. While they can be sufficiently accurate in some contexts, they may be woefully inadequate in others. In what almost sounds like a “scientific fight club”, the “rules” of bioinformatics are as follows:

1. There are no “universal” rules.
2. Every seemingly fundamental paradigm has one or more exceptions.
3. The validity of a bioinformatics method depends on unknown charac-

teristics of the data.

4. Biology is always more complicated than you think, *even after taking this rule into account.*

Below, we have attempted to describe the biological concepts that we deem important to understand the types of information encoded in data. Each of our definitions is short and terse, and we recommend additional self-guided study of each concept of interest.

5.1 What is DNA?

DNA stands for Deoxyribo Nucleic Acid.

It is a macromolecule (a molecule constructed of smaller molecules) that carries the genetic instructions required for the development, functioning and reproduction of all known living organisms. In eukaryotic organisms (like animals, plants, and fungi), DNA is present in the nucleus of each cell. In prokaryotic organisms (single-celled organisms like bacteria and mitochondria), DNA is present in the cell's cytoplasm.

5.1.1 What is DNA composed of?

DNA is made up of two strands of smaller molecules coiled around each other in a double-helix structure. If you uncoiled DNA, you could imagine it looking somewhat like a ladder. Ladders have two important parts: the poles on either side and the rungs that you climb. The “poles” of DNA are made of alternating molecules of deoxyribose (sugar) and phosphate. While they provide the structure, it’s the “rungs” of the DNA that are most important for bioinformatics.

To understand the “rungs” of the DNA, imagine a ladder split in half down the middle so that each half is a pole with a bunch of half-rungs sticking out. In DNA, these “half-rungs” are a molecule called nucleotides. Genetic information is encoded into DNA by the order, or sequence, in which these nucleotides occur. The “ladder” of DNA is held together by the bonds between each “half-rung.”

5.1.2 What are nucleotides?

Nucleotides are the building blocks of nucleic acids (DNA and RNA—we’ll get to that one later on). In DNA, there are four types of nucleotide: Adenine, Cytosine, Guanine, and Thymine. Because the order in which they occur encodes the information biologists try to understand, we refer to them by their first letter, A, C, G and T, respectively.

A	Adenine
C	Cytosine
G	Guanine
T	Thymine

Back to the ladder analogy. “Nucleotide” is the *type* of molecule that makes up each half-rung of our ladder. Because they function as the units that encode genetic information, each letter is also called a base.

For an example of how we represent the sequence of DNA bases using these letters, if we took the DNA of the *Bacillus anthracis* bacteria that causes Anthrax disease, unfolded the double-helix into a ladder, and then split the ladder in two, the top half (the forward strand—we’ll get to that in a bit) would be written like this:

ATATTTTTCTTGTCCCCATATCCACAAACTCTTTT

5.1.3 What are base pairs?

When you put both sides of the ladder back together, each rung is made by the bond between two bases. When they are bonded, we call them a base pair (or sometimes “bp”).

When it comes to base pairs, it’s important to remember that Adenine only ever bonds with Thymine, and Guanine with Cytosine. In the ladder analogy, this means that each rung will be denoted as “A-T,” “T-A,” “G-C,” or “C-G.”

Certain groups of nucleotides that share some common attribute may be designated by so called *ambiguity* codes, for example, W stands for A or T:

Y	Pyrimidine (C or T)
R	Purine (A or G)
W	Weak (A or T)
S	Strong (G or C)

K	Keto (T or G)
M	Amino (C or A)
D	A, G, T (not C - remember as after C)
V	A, C, G (not T - remember as after T/U - We'll get to "U" soon)
H	A, C, T (not G - remember as after G)
B	C, G, T (not A - remember as after A)
N	Any base
-	Gap

5.1.4 What are DNA strands?

Remember when we split the DNA “ladder” in half down the middle so that each side was a pole with half-rungs sticking out? These two halves are called strands. To distinguish the two strands, scientists label one the forward strand and the second, the reverse strand. Above, we gave the example of the forward strand in the *Bacillus anthracis* bacteria. Here’s what it looks like paired with its reverse strand:

```
forward --> ATATTTTTCTTGTAAAAATATCCACAAACTCTTT
                  ||||||| ||||||| ||||||| ||||||| |||||
                  TATAAAAAAGAACAAAAAATAGGTGTTGAGAAA <-- reverse
```

The lines that connect the bases on either side denote a basepair relationship.

“Forward” and “reverse” are just labels. The choice of labeling is arbitrary and does not depend on any inherent property of the DNA. The forward strand is not “special.” Scientists decide which to call “forward” and which to call “reverse” when they first analyze the DNA of an organism. Even though the decision is arbitrary, it’s important to maintain consistency with that decision for the sake of clear communication.

The forward and reverse strands may also be denoted with different terms. For example, in some datasets you may find them labeled as + and -. They might also be called *top* and *bottom* strands, or even *Watson* and *Crick* strands.

In our opinion, these variances are needlessly confusing. Please avoid referring to strands with any other terms than *forward* and *reverse*.

5.2 Is there a directionality of DNA?

Yes, there is a directionality to the DNA determined by the polarity of the molecules. This direction runs in opposite ways for each strand. Typically, we indicate this polarity with arrows:

```
----->
ATATTTTTCTTGTAAAAATATCCACAAACTCTTT
||||||| | | | | | | | | | | | | | | | |
TATAAAAAAGAACAAAAAATATAGGTGTTGAGAAAA
-----<
```

Most biological mechanisms (but not all) take place on a single strand of the DNA and in the direction of the arrow. Hence, sequences of the DNA above will be “seen” by the biochemical machinery as either:

ATATTTTTCTTGTAAAAATATCCACAAACTCTTT

or as:

AAAAGAGTTTGTGGATATAAAACAAAGAAAAATAT

This last sequence is called the *reverse complement* of the first and is formed by reversing the order of the letters then swapping A for T and swapping C for G (and vice-versa).

Hence a DNA sequence AAACT may need to be considered:

- in reverse, TCAAA
- as a complement, TTTGA
- as a reverse-complement, AGTTT

5.3 What is sense/antisense?

When a process occurs in the expected direction, its directionality may be called *sense*; if it is going against the normal direction, its directionality may be called *anti-sense*.

It is very important not to collate the concepts of *forward/reverse* with *sense/anti-sense*, as these concepts are completely unrelated. The *sense/anti-*

sense is relative to a sequence's direction; the sequence, in turn, may come from a *forward* or *reverse* strand.

5.4 What is DNA sequencing?

DNA sequencing is the “catch-all” terminology that describes all the processes for identifying the composition of a DNA macromolecule.

The results of a DNA sequencing process are data files stored in an unprocessed format, typically either **FASTA**, **FASTQ**, or unaligned **BAM** (called **uBAM**) files. Most published papers also store their data in repositories that can be downloaded for reanalysis.

5.5 What gets sequenced?

It is essential to note that instruments do not directly sequence the DNA in its original form. Sequencing requires a laboratory process that transforms the original DNA into a so-called “sequencing library” - an artificial construct based on the original DNA. The process of creating this sequencing library introduces a wide variety of limitations and artificial properties into the results. Also, the method of building the sequencing library will also limit the information that can be learned about the original DNA molecule.

Most (perhaps all) life scientists use the term “sequencing” over-generously and are often under the assumption that it produces more precise information than what it can deliver.

5.6 What is a genome?

A genome is all of an organism’s DNA sequence. Each cell typically contains a copy of the entire genome. More accurately, each cell has one or more nearly identical copies of the genome. Replication is the process of duplicating the genome when a cell divides.

While, due to complementarity, the number of **A** and **T** nucleotides and the **C** and **G** nucleotides is equal, the relative ratios of **AT** pairs vs **CG** pairs may be

very different. Some genomes may contain more AT pairs while others may contain more CG pairs.

5.7 What is a genome's purpose?

The genome contains the information that makes the functioning of an organism possible. In cellular organisms, for example, the genome has regions that contain the instructions for making proteins. These are typically called “coding regions”.

The genome may also have regions used to produce molecules other than proteins, as well as regions that regulate the rates by which other processes take place.

All genomes are subject to evolutionary principles. Hence, some (or even substantial) parts of a healthy genome may be non-functional and may no longer serve any obvious purpose. Some percent of a genome may consist of copies of various kinds of interspersed, repeat sequences. At some point, biologists labeled these regions “junk DNA”, but the term has since become a lightning rod of controversy.

Identifying non-functional regions has turned out to be more difficult, and quite more controversial than biologists originally anticipated. The so-called C Value Paradox¹ captures the observation that the size of a genome does not directly determine the genome's complexity.

5.8 How big is a genome?

Functional genomes range from as short as the 300 base pairs of the prions that cause the mad-cow disease to as long as the 150 billion base pairs of *Paris japonica*, a rare, perennial, star-like flower from Japan. It is common to refer to genome sizes in terms of kilo-bases (thousands), mega-bases (millions), and giga-bases (billions).

Here are some other genome sizes:

- Ebola virus genome: 18 thousand basepairs (18Kb)

¹https://en.wikipedia.org/wiki/C-value#C-value_paradox

- E.coli bacteria genome: 4 million basepairs (4Mb)
- Baker's yeast fungus genome: 12 million basepairs (12Mb)
- Fruit fly genome: 120 million basepairs (120Mb)
- Human genome: 3 billion basepairs (3Gb)
- Some salamander species: 120 billion basepairs (120 Gb)

5.9 What is RNA?

Whereas DNA is the blueprint, RNA is a smaller interval of this plan translated into a new kind of molecule. This molecule is similar to DNA in that it differs only owing to a few chemical modifications that change its properties relative to DNA. For example, in RNA, the base T (Thymine) is replaced by the nucleotide U (Uracil).

RNA is a polymeric, single-stranded molecule that often performs a function and is believed to exist transiently. Unlike DNA, there are many classes of RNA: mRNA, tRNA, rRNA, and many others. DNA is continuously present in the cell, whereas RNA degrades quickly over time (minutes).

5.10 How does a genome function?

The genome has numerous functions, most of which are too complicated (or insufficiently studied) to describe accurately. Instead, scientists employ a “biological narrative” to describe genomic functions and processes.

The “narrative story” is made very simple by design, often accompanied by a simplified illustration of events: step 1 followed by step 2 followed by step 3, etc. While this helps initially, it also runs of risk of explaining concepts in an oversimplified manner. To give you an example of how this looks in practice, here is a “biological narrative” of what is perhaps the most studied genomic phenomenon: *primary mRNA transcription* in eukaryotic cells.

“The cell begins by transcribing a “gene” (see later) into an RNA molecule. After transcription, pieces of the RNA are cut out and discarded in a process called “splicing.” Each discarded piece is referred to as an *intron*. Each piece between consecutive introns

is known as an *exon*, and the RNA molecule with the introns removed is known as messenger RNA, or *mRNA*.

Part of the cell's method for identifying introns is the presence of **GT** and **AG**, the so-called "splice signals" that usually occur as the first and last dinucleotides of an intron. But the mere presence of these dinucleotides is not sufficient for splicing to occur. Perhaps 35% of human genes are alternatively spliced, meaning that under different circumstances, various combinations of exons are selected.”

Remember, again, that the above explanation is a simplistic description of a far more complex phenomenon. Any single detail of it opens a series of yet unexplainable questions: why is it that only a small subset of **GT AG** pairs cause splicing and how come the polymerase that initiates a splicing event at a **GT** knows that there will be a **AG** later? Or perhaps it does not know and it is all random - but that would have other implications. You see how the rabbit hole only gets deeper and deeper.

5.11 What is a protein?

A protein is a three-dimensional macromolecule built from a series of so-called “amino acid” molecules that can form a 3D structure. There are 20 kinds of amino acids that can form a protein; these are labeled as letters in the “alphabet” of protein sequences, where each letter is an amino acid. The “alphabetical order of the amino acids alphabet is:

ARNDCEQGHILKMFPSTWYZ

Where **A** stands for Alanine, **R** for Arginine, **N** for Asparagine and so on²

Proteins can be described by their sequence alone. But in our current state of understanding, the sequence itself is typically insufficient to fully determine the protein’s 3D structure or function.

Whereas DNA and mRNA usually carry *information*, proteins are the actual, physical building blocks of life. Every living organism is built out of proteins

²https://en.wikipedia.org/wiki/DNA_codon_table

and functions via the interactions of proteins that are being continuously produced.

A short series (less than 40) of amino acids without a well-defined 3D structure is called a polypeptide (peptides).

5.12 How are proteins made?

The process of reading *DNA* and creating *mRNA* out of it is called *transcription*. Then, in eukaryotes, the *mRNA* is transported out of the nucleus (to the cell's cytoplasm), where it is converted to a protein sequence in a process called *translation*. Multiple proteins (even hundreds) may be translated from a single mRNA molecule.

- transcription: DNA → mRNA
- translation: mRNA → Protein

To perform the translation, the mRNA is partitioned into units of three consecutive letters, each called a *codon*. A *codon* is then translated via a translation table³ into an *amino acid*. A protein is a sequence of amino acids.

There is a genetic code for translating a codon into an amino acid. For example, the codon TCA (or UGA if we describe it as an RNA sequence) codes for S, the amino acid *Serine*. There are 64 combinations of codons but only 20 amino acids. Some codons code for the same amino acid, this is called the codon degeneracy⁴. For example:

CGU, CGC, CGA, CGG, AGA, AGG --> Arginine (Arg/R)

Most (but not all) amino acids are encoded by more than one codon. A few codons, in most cases TAA, TAG and TGA are the so called *stop codons*; translation stops when one of them is reached.

The translation process begins with the so-called start codon ATG that corresponds to M (the *Methionine* amino acid).

For example the subunit B of the Cholera Toxin protein complex responsible for the massive, watery diarrhea characteristic of cholera infection has the

³https://en.wikipedia.org/wiki/DNA_codon_table

⁴https://en.wikipedia.org/wiki/Codon_degeneracy

following sequence:

MIKLKFGVFFTVLSSAYAHGTPQNIIDLCAYHNTQIYTLNDKIFSYTESLAGKREMAI
ITFKNGAIFQVEVPGSQHIDSQQKAIERMKDTLRIAYLTEAKVEKLCVWNKTPHAAIASMAN

It starts with an M, the Methionine, which as we just stated is also the start codon. You may think all protein sequences should start with M.

But wait, remember how every rule has exceptions? That applies here as well. While the overwhelming majority of proteins start with an M it is possible for this M to get cut off during later processes.

5.13 What is an ORF?

An **ORF**, or *open reading frame*, is a sequence of at least, say, 100 consecutive codons without a stop codon.

It is important also to note that while there is a “standard” translation table, some organisms may use slight variations of it. These are called genetic codes⁵. When translating from genome to protein, the use of the correct genetic code is essential.

5.14 What is a gene?

In our experience, the word gene is one of the most misused words in the entirety of biology. It is usually interpreted quite differently by different scientists. Some think of genes as the DNA that codes for proteins; others think genes may include upstream elements, and so on. The “official” definition of the term gene⁶ in the Sequence Ontology is

A region (or regions) that includes all of the sequence elements necessary to encode a functional transcript. A gene may include regulatory regions, transcribed regions and other functional sequence regions.

⁵https://en.wikipedia.org/wiki/List_of_genetic_codes

⁶http://www.sequenceontology.org/browser/current_svn/term/SO:0000704

5.15 Do genomes have other features?

Genomic regions may have a wide variety of functions. Here are a few often-studied types:

5.15.1 Untranslated regions

The region of the mRNA before the start codon (or the corresponding genomic region) is called the 5' UTR (5 prime UTR), or untranslated region; the portion from the stop codon to the start of the poly-A tail is the 3' UTR (three prime UTR).

5.15.2 Promoter regions

The genomic region just before the 5' UTR may contain patterns of nucleotides, called the promoter, that are used to position the molecular machinery that performs the transcription to RNA. For about 60% of human genes, the promoter occurs in a CpG island: a region of DNA where C and G are far more frequent than A and T. Other patterns in the DNA tell the cell when (how frequently and in what tissues) to transcribe the gene; that is, they regulate transcription. A pattern that increases the frequency of transcription operations is an **enhancer**, while one that decreases the rate is a **silencer**.

5.15.3 CpG islands

CpG islands are regions of DNA where a C (cytosine) nucleotide is followed by a G guanine nucleotide in the linear sequence of bases along its 5' -> 3' direction. CpG is shorthand for 5'-C-phosphate-G-3', that is, cytosine and guanine separated a phosphate; phosphate links any two nucleosides together in DNA. Promoter regions may contain CpG islands.

Cytosines in CpG dinucleotides can be methylated. In turn, methylated cytosines within a gene may change the gene's expression, a mechanism that is part of a larger field of science studying gene regulation, called epigenetics.

Below is a section of a CpG island from chromosome 1 of the human genome. It has 30 CpGs and GC% is 77.14.

```
>gi|568815597:36306860-36307069 Homo sapiens chromosome 1, GRCh38.p7 Primary Ass
CGGGGCTCCGGAGAGGCGCGGAGGCCGCCTGTGCGCGCCGCCGAGGTGAGCGCAAGGGCGGGACGGC
GCCGGTGGGCGGGTGCACGGAGCCAGTGCACCGCCGGCTCTCCGGCTTTAGTGACGGCGCGCTCTG
GGCGGGACCTCGGGGCCGCCCTGCGGTCTGTGATTGGTTCTCGAGTGCAATGCTCCGCCCTGGGGCGGG
```

We obtained the above via:

```
efetch -db=nuccore -id=NC_00001.11 -format=fasta -seq_start=36306860 -seq_stop=
```

This tool will be covered in later sections.

5.15.4 Enhancers

Enhancers help a gene turn on and regulate where, when, and how much the gene should be expressed for the cell to function properly. One enhancer can regulate many genes, and one gene can have many enhancers. Since DNA is folded and crumpled up in a cell, an enhancer can be far away from a gene it regulates in terms of distance measured along the DNA strand, but physically very close to the gene promoter. The best way to identify a sequence as an enhancer is to experimentally disrupt it and observe gene expression changes. Since this is cumbersome in practice, we use proxies like histone modifications (chemical decorations on the structural backbone upon which DNA is coiled) to identify enhancers. Silencers work similarly – they can be far from a gene, many-to-many relationships, hard to find – but cause a gene to be less, rather than more, expressed.

5.16 What is homology?

Two regions of DNA that evolved from the same sequence (through processes of duplication of genomic regions and separation of two species) are **homologous**, or **homologs** of one another.

More specifically, regions in the genomes of two species that are descended from the same area in a common ancestor's genome are **orthologs**. These regions are said to be **orthologous**.

On the other hand, **paralogous** sequences or **paralogs** were separated by duplication of a genomic region within the same genome.

Standard pet peeve of many scientists:

Homology is not a synonym of sequence similarity!

Homologous sequences are usually similar to one another, but similarity of sequences does not indicate homology.

Chapter 6

How is bioinformatics practiced?

Bioinformatics requires a broad skill set. The diverse tasks undertaken by bioinformaticians can be roughly divided into three tiers:

1. Data management.

Data management requires accessing, combining, converting, manipulating, storing, and annotating data. It requires routine data quality checks, summarizing large amounts of information, and automating existing methods.

2. Primary data analysis.

Analysis of the data requires running alignments, variation callers, and RNA-Seq quantification, as well as finding lists of genes. Strategically, the bioinformatician must anticipate potential pitfalls of planned analyses, find alternatives, and adapt and customize the methods to the data.

3. Data interpretation.

Data management and analysis are meaningless without accurate and insightful interpretation. Bioinformaticians discover or support biological hypotheses via the results of their analyses, and so they must be able to interpret their findings in the context of ongoing scientific discourse.

6.1 What is the recommended computer for bioinformatics?

In our experience, the most productive setup for a bioinformatician is using a Mac OS based computer to develop and test the methods and then using a high-performance Linux workstation—or cluster—to execute these pipelines on the data. With the release of Windows 10 Creator Edition (accessible for users that are part of the Windows Insider program), a fully functional Linux Bash shell can be installed and run on Windows. See the Computer Setup page for more details.

6.2 How much computing power do we need?

Bioinformatics methodologies are improving at a rapid pace. A regular workstation is often all you need to analyze data from a typical, high-volume sequencing run (hundreds of millions of measurements). For example, you could handle most RNA-Seq data analysis required for a given day by using only the `hisat2` aligner on a standard iMac computer (32GB RAM and eight cores used in parallel). More substantial analyses like genome assembly, however, typically require more of memory than what is generally available on a standard desktop computer.

As a rule, low-quality data (contamination, incorrect sample prep, etc.) takes substantially longer to analyze than high-quality data.

6.3 Does learning bioinformatics need massive computing power?

No!

Mastering bioinformatics requires nothing more than a standard, UNIX-ready laptop. You can learn bioinformatics using only a \$100 Chromebook.

Of course, the computational needs for applying the bioinformatics methods on particular problems will depend on the amount and scale of data being analyzed. A \$100 Chromebook has the tools one needs but probably doesn't

have the compute power, storage or memory to run analyses of the massive datasets that may need to be processed.

You might be surprised, however, just how much can be done on just a Chromebook!

Remember this; you don't need an expensive or powerful system to become one of the best bioinformaticians on the planet! You can learn, practice and gain a profound understanding of this entire field on any simple, Unix capable computing device.

6.4 What about the cloud?

Cloud computing is becoming an increasingly common platform for bioinformatics, in particular for efforts that allow bioinformaticians to “bring the tools to the data” - applying your algorithms to massive sequence databases. Cloud services such as Amazon Web Services also enable anyone to host web applications and services on powerful, accessible machines while only paying for the compute time they use.

Running a cloud service involves learning about object stores, virtual private clouds, file systems, and security. Building and maintaining a cloud instance requires you become something like a part-time systems administrator, which can distract you from learning bioinformatics. In later chapters, we will introduce both cloud computing and cloud-based stacks - commonly referred to as software-as-a-service (SaaS), platform-as-a-service (PaaS, and infrastructure-as-a-service (IaaS). These offerings, combined with *containerized* applications and analyses, have become valued for their potential to provide a scalable platform for reproducible research.

If you are comfortable with these concepts, a cloud-based service can be a companion to this guide, but learning bioinformatics *does not require* the cloud.

6.5 Do I need to know Unix to do bioinformatics?

Bioinformatics has primarily been developed via freely available tools written on the Unix platform. The vast majority of new advances are published with software written for Unix-based operating systems.

It's unlikely you'll be in a position for long-term career advancement as a bioinformatician without a basic understanding of the command line. Starting with 2016, the Windows platform has begun to offer the so-called "Bash for Windows" application that allows the Windows 10 operating system to run almost all of the Unix-specific software.

The good news is that using Unix is not that complicated. While somewhat unforgiving, the Unix philosophy is logical and precise. In our experience, even people with no prior computational background can become confident Unix users in a matter of weeks.

6.6 Do I need to learn a programming language?

Yes — an introductory level of programming ability (in any programming language) will be necessary.

Fortunately, programming languages function by similar underlying logic — even when they seem entirely different on the surface. While symbols, syntax, and procedures may differ among programming languages, an introductory level of programming ability in any language is necessary to understand the thought processes required for computational analysis.

We do cover programming concepts in this book. See the section titled Some programming required and the corresponding subsections for a primer.

6.7 Are there alternatives to using Unix?

Alternatives to Unix are limited but do exist. They fall into two categories:

6.7.1 Software that provides a web interface to the command line tools

These software systems run the same command line tools that one could install into a Unix-based system, but the interfaces to these tools are graphical and provide better information “discoverability.”

1. Galaxy (open source)
2. GenePattern (open source)
3. BaseSpace (commercial)
4. DNA Nexus (commercial)
5. Seven Bridges Genomics (commercial)

6.7.2 Systems that offer custom implementations of bioinformatics methods.

These can be standalone software that runs on your local system — they are compatible with web-based applications. They often run open-source tools via a GUI.

1. CLC Genomics Workbench
2. Partek
3. Golden Helix
4. SoftGenetics
5. DNA Star
6. Geneious

6.8 What is Bioconductor?

Bioconductor¹ is an open-source software project for the analysis and comprehension of genomic data. Bioconductor is based primarily on the R statistical programming language²

Users of Bioconductor typically write scripts using R that make calls to

¹<https://www.bioconductor.org/>

²<https://www.r-project.org/>

functions defined in the Bioconductor library. For example, a Bioconductor-based script that runs an RNA-Seq data analysis could look like this:

```
biocLite("DESeq")
library(DESeq)
count = read.table("stdin", header=TRUE, row.names=1 )
cond1 = c("control", "control", "control")
cond2 = c("treatment", "treatment", "treatment")
conds = factor(c(cond1, cond2))
cdata = newCountDataSet(count, conds)
esize = estimateSizeFactors(cdata)
edisp = estimateDispersions(esize)
rdata = nbinomTest(edisp, "control", "treatment")
write.table(rdata, file="", sep="\t", row.name=FALSE, quote=FALSE)
```

6.9 What is Galaxy?

Galaxy³ is a web-based scientific workflow and data analysis platform that aims to make computational biology accessible to research scientists with no computer programming experience.

Galaxy can generate and present a web-based user interface to existing command-line tools, making them available to those unable to run the tools themselves.

Important: Galaxy is not a data analysis tool! It is an **Interface** and a **Platform** that provides computational capacity that runs **command line** bioinformatics software on the users' behalf. Galaxy makes it seem simple to launch tools and manage data – but it is the user's responsibility to understand the underlying concepts and use these tools correctly.

Pet peeve: *Please don't say I used Galaxy to analyze data.* That is like saying I used a computer to do bioinformatics. As stated above Galaxy is an interface and a platform that uses tools developed by scientists around the world. Recognize the tools authors, cite the publications for the software, then mention and recognize Galaxy.

Galaxy can also serve as graduation moment in a bioinformaticians' life.

³<http://www.usegalaxy.org>

We've never met an expert bioinformatician that would voluntarily want to use Galaxy for their own needs. You see the "ease" of use of Galaxy is, in the end, nothing more than *an illusion*. A skilled scientist will always prefer to specify their work via set words such as:

```
bwa mem file1.fq file2.fq | samtools view -F 68 | samtools sort > result.bam
```

instead of describing it as a series of clicks in an interface.

In our opinion, your comfort level of using Galaxy relative to using the command line quantifies your skill level. The moment you feel the drag, the slowdown, the limitations and that "straight-jacket" feel on your creativity that Galaxy imposes on you is the moment where you have graduated from the *Elementary School of Galaxy* and you've entered the *Academy of the Command Line*.

But by all means, if Galaxy works for you keep using it, it is a progression step.

And let's make one thing clear, the above sentiment is valid for just about all "easy to use" graphical bioinformatics data analysis platforms. There may be a time when you too will recognize that "easy-to-use" is, in reality, a restriction, a curse, it means that you are limited in what you can do and that you will have to operate inside of a glass box with invisible walls.

Rant over :-). Back to our regularly scheduled content.

Galaxy uses a three-panel system where the left panel lists the tools, the middle panel describes the tools, and the right panel displays user data.

6.10 What is BaseSpace?

BaseSpace⁴ is a cloud-based genomics analysis and storage platform provided by Illumina that directly integrates with all Illumina sequencers.

The system can operate only with sequencing data produced by Illumina and only when the data is in the formats generated by the sequencers.

⁴<http://www.illumina.com/informatics/research/sequencing-data-analysis-management/basespace.html>

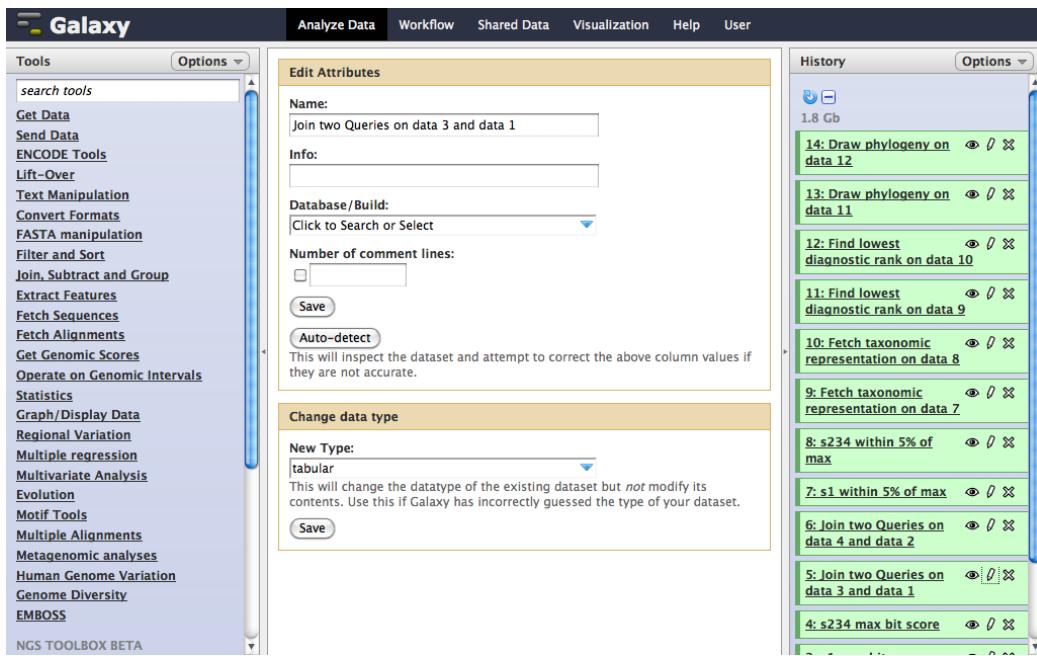


Figure 6.1: Galaxy Interface

6.11 Are commercial bioinformatics software packages expensive?

From an individual, consumer-level perspective, the cost of commercial bioinformatics software is high — typically measured in thousands of dollars. The price tag means that few people could or would buy them directly.

For larger organizations, the software costs may pay off quickly in productivity gains as long as the software is chosen carefully to match specific research needs.

Most academicians and open-source developers treat commercial software with some distrust. In our opinion, this distrust may be unwarranted. Many commercial tools cater to specific needs and need to be operated by individuals with diverse levels of training. Also, the goals and resource allocations in academia are typically very different than those of private organizations.

6.12 Should I freelance as a bioinformatician?

In our opinion, the answer is **no** unless the terms are generous or your clients appear to recognize the potential complexities that you may run into.

Successful bioinformatics analysis requires more than skill alone: the information content of the data is a crucial second piece. The world's best bioinformatician won't be able to extract information that is not present in the data and may have trouble demonstrating that they have done their best.

Our experiences line up with the observations in Core services: Reward bioinformaticians, Nature, (2015)⁵ that state that analyses always end up being substantially more complicated than anticipated.

It is unexpectedly difficult to estimate the effort that a project requires accurately.

Amusing story

I once got called up by a company offering custom RNA-Seq analysis. It was a quite basic service as far as analyses go. They ran the Tuxedo pipeline and provided the files it generated. Once we established what the service consisted of I asked how much it cost. The answer was surprising:

“For educational institutions, we charge *two fifty* per sample.”

I was both stupefied and elated by this answer. Here I am, I thought I knew all about how to estimate the cost of bioinformatics analyses. Yet I am unable to decide what *two fifty* meant. A typical RNA-Seq at that time might have had ten samples. So:

1. Was it \$2.50 per sample, that would sum up to a ridiculously low \$25, that adds up to less than the cost of owning the computing power for an analysis of this type.
2. Or was it \$250 per sample, that would sum up to a ridiculously high \$2500, for executing a straightforward command line tool that at most would take an hour to set up and maybe some hours to run.

⁵<http://www.nature.com/news/core-services-reward-bioinformaticians-1.17251>

I meekly inquired: *What is two fifty? Is it \$2.50?*

*** glacial pause **

Answer: *No it is \$250*

Moral of the story, if someone pays \$250 per sample, you should take the job. If someone pays \$2.50 then better don't.

6.13 What do bioinformaticians look like?

Handsome of course!

Here are some participants at the NSF-funded “Analyzing Next Generations Sequencing Data” workshop at Michigan State University (2014). Graduate students, postdocs, researchers, faculty, were all represented.



Figure 6.2: MSU Workshop

Chapter 7

How to solve it

This section has been inspired by the book titled How to solve it¹ by George Pólya describing methods of problem-solving. His advice can be summarized via:

1. First, you have to understand the problem.
2. After understanding, then make a plan.
3. Carry out the plan.
4. Look back on your work. How could it be better?

The advice is general and timeless, but how does it apply to Bioinformatics?

7.1 What is holistic data analysis?

In our opinion, one of the most essential characteristics of the analysis is to think about it “holistically” – considering all steps that need to be performed.

The best analogy is drawing - usually artists will start with a rough draft. In time, they will add more detail and accuracy to their work. The same applies to any data analysis - do not get bogged down with the initial details - make it first work on a rough scale - so that you see all the moving components that will make up your final result. Only when you have all the parts together will you have a firm understanding of the strengths and weaknesses of the entire process.

¹https://en.wikipedia.org/wiki/How_to_Solve_It

Imagine and visualize what your result will need to look like and work toward that goal.

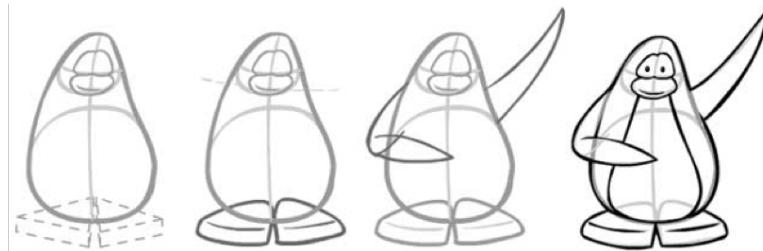


Figure 7.1

7.2 How do I perform a holistic analysis?

The most straightforward approach is to isolate a small section of your data and build your analysis around that. Don't start with processing hundreds of gigabytes of data against genomes with billions of bases.

Start small and stay in control. You will be able to build pipelines that run in minutes (or even seconds), and you will be continually aware of what is taking place, what problems are cropping up, and what the results look like. You'll be able to identify very early if something is amiss. Your analysis will run quickly even on a small laptop.

Do not try to optimize a process too early. Build your pipeline to be flexible and robust that you can rerun with other techniques if you need to. You will only really understand the problems that you need to solve when you get near the end.

How to select these subsets? Here are some examples we employ in this book:

1. Download just a subset of the data. For instance in all our cases that use `fastq-dump` we limit data to values between 10,000 or 1 million reads. You can tweak these numbers to find that balance where the results are quick and allow you to understand and iterate over the results.
2. Build random samples out of your data. Tools such as `seqtk`, `seqkit` and others will let you randomly select a much smaller subset of data.

7.3. WHAT ARE THE RULES OF A BIOINFORMATICS ANALYSIS?91

3. Use a smaller segment of the target genome. If you study large genomes, use just a single chromosome of it. For example, build your pipeline but execute it using only human chromosome 22 instead of the entire human genome. Then swap out the reference file when you are ready to do it all.
4. Generate data with known properties. We have a chapter on simulating data - and it is, in fact, the best way to understand the features and limits of the various processes that you encounter. When you analyze data with known properties, you can validate and understand how your methods work.

7.3 What are the rules of a bioinformatics analysis?

In our center, written on the whiteboard is the motto and credo of how we believe progress is made in this field:

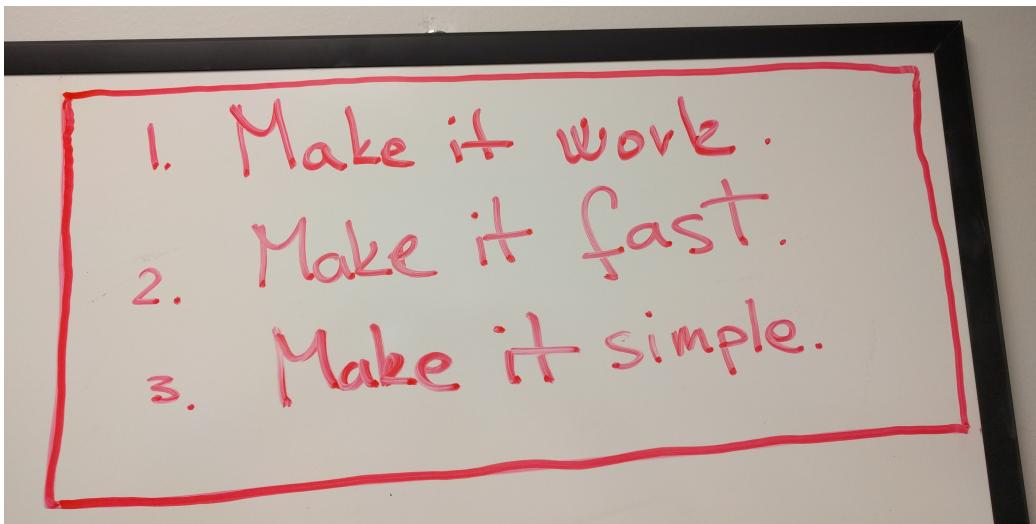


Figure 7.2

1. Make it work.
2. Make it fast.
3. Make it simple.

7.4 What does “make it work” mean?

Well, it means get the entire process moving - not just the beginning - the whole thing. It will be hacky, it will be squeaky, it may need to be held together with duct tape. Go all the way! Find a way to reduce your problem so that it finishes in minutes, perhaps to a maximum one hour. Then you can iterate on it.

7.5 Why is fast so important?

On the one hand, doing it fast sort of make sense. Who wouldn’t want to do things fast?

But there is more to it than just getting home by six. It is about the way we think about problems in the subconscious. When a process takes three days, we instinctively think up questions where we factor in the time it takes to get the answer. We try to ask more “difficult” questions that are “worth” the time. If a solution takes just two minutes, we can maintain mental focus on the problem, and we will investigate it with a very different approach.

So make your data small, analyze it quickly and ask quick and easy questions - but lots of them. You will get much further.

7.6 What does “simple” mean?

Simple means identifying the commonality of the processes, and generalizing the solution to take fewer steps. It is very likely that once you understand the solution, once it works and once it is fast, you can find better ways to do the achieve the same results with even fewer assumptions and requirements. Your building blocks should be like “lego blocks” that are reusable for building any other structure. Turn a specific solution into a generic one and you’ll find yourself immensely more productive.

7.7 How to deal with anxiety and stress?

It may feel strange to talk about anxiety in a data analysis book. But make no mistake about it - it is something many of us experience.

Bioinformaticians often work alone because other members of the group do not have sufficient expertise to contribute. In that case the data analyst alone needs to make dozens if not hundreds of decisions and they alone are responsible for applying these correctly.

In addition most tools can be cryptic and terse, with features that you may know nothing about. When you save the commands that you type for posterity (and yes, you should), anyone at any time can review and criticize these, and any error you make will be visible and apparent. Being the sole decision maker of this analysis puts the burden on you alone.

With time the added pressure of this sole responsibility can wear you down psychologically. After a while, you won't remember why you made certain decisions, and you might have increasing doubts and fears. When many months later the paper is accepted and the publication comes around you may ask yourself: *Did I do it right? What if now, that many people will look at it, they will instantly find and point out that trivial and glaring mistake that renders everything invalid?*

One of the saddest, public, episodes of science are described in the article Doubts about Johns Hopkins research have gone unanswered, scientist says² a story that follows what we believe a situation where the added pressure of being the sole data analyst of a significant paper has led to a tragic outcome. Notably, a recent article points out that Ph.D. students face significant mental health challenges³. The risk of being affected by the [impostor syndrome] is high.

Doubts about Johns Hopkins research have gone unanswered, scientist says ⁴

²https://www.washingtonpost.com/business/economy/doubts-about-johns-hopkins-research-have-gone-unanswered-scientist-says/2013/03/11/52822cba-7c84-11e2-82e8-61a46c2cde3d_story.html

³<http://www.sciencemag.org/careers/2017/04/phd-students-face-significant-mental-health-challenges>

⁴<https://www.washingtonpost.com/business/economy/>

For me the stories above are eerily familiar - I wish I had known about them sooner. You see, earlier in my career I've struggled with similar issues. I've rarely enjoyed the publication of what later turned out to be some of my most cited papers - in fact; I've dreaded the publication date - I saw it as the date when everyone finds out that I made a mistake, it is all wrong and that I am incompetent.

Today I know better - and I tell you the same. If you do your due diligence, follow good practices, keep it simple, keep it straight, everything will be ok, it does not matter that much anyway.

And you know what? If you have messed up because among the hundreds of commands you've entered by accident

```
samtools view -f 266
```

instead of the “correct” command:

```
samtools view -f 226
```

then whose fault is it? Do you see how hard it is to catch this error? Especially when it might be buried among hundreds of equally unreadable statements. I believe that above it is the absurdity of the tool that bears all responsibility. A well-designed tool should never allow you to make errors of this type and this magnitude accidentally. As you will see later mistyping a single digit will radically alter the entire meaning of the command, yet there is no way to recognize that from just looking at it. The number that you see there needs to be interpreted as the binary bits encoded into that integer $226 = 11100010$ vs. $266 = 100001010$.

Gee, thank why you `samtool` creators (sarcasm) for providing me with all the rope I need to shoot myself into the foot!

Psychology says that cursing helps: `samtools` you suck and worse, the SAM format is amusingly ill-designed. GATK? That tool is bonkers ... I won't go on, but the list is practically endless. Bioinformatics, unfortunately, represents all too well the disconnect of the Ivory Tower from reality, where the tool makers are “too smart for their own good” and create software that is fragile, brittle, difficult to use then trivial to misuse, and frankly often just plain wrong.

doubts-about-johns-hopkins-research-have-gone-unanswered-scientist-says/2013/03/11/52822cba-7c84-11e2-82e8-61a46c2cde3d_story.html

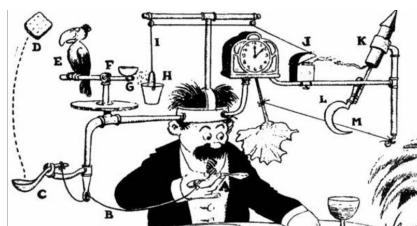
So once you get that off your chest, you can go back to enjoy data analysis - and if you do mess up... well, rest assured many, many others did too. You may have been alone in the lab, but when it comes to people that published at least one incorrect analysis, you aren't alone.

There are a lot of us out there.

Chapter 8

How not to waste your time

Every time I tried to use any of the tools and techniques listed below, I found that I've wasted a lot of time and gained no benefit at all. I want to save you the trouble.



I have come to believe that tools and solutions below are bioinformatics equivalents of Rube Goldberg machines¹, basically overcomplicated gadgets performing what should be straightforward tasks.

Your solutions will always be more natural and more straightforward when done and automated with a simple shell script or a makefile.

8.1 The False Peaks of Biology

In mountaineering, a false peak or false summit is a peak that appears to be the pinnacle of the mountain, but upon reaching, it

¹https://en.wikipedia.org/wiki/Rube_Goldberg

turns out the summit is higher. False peaks can have significant effects on climbers' psychological states by inducing feelings of dashed hopes or even failure.

There is a simple explanation of why the techniques and methods listed on this page even exist. Some approaches are even supported by continually renewed, multi-million dollar governmental research grants.

You see, the vast majority of life scientists mistakenly believe that the primary limitation to understanding biology is their inability to run some existing software. All these scientists believe that if they could get over this first hurdle, the *software mountain* that they see right in front of them, everything would start making sense and that they would be well on their way to making scientific discoveries. They believe all that because they are *computationally illiterate* - they have never been taught computation. Understandably so, since the field of data-oriented biology is new, going back perhaps ten years.

Compared to the real challenges of bioinformatics, the computing obstacle that they experience is nothing more than a false peak²:



The false peaks of biology are responsible for deeply misguided efforts like CWL, CyVerse, and Galaxy. These solutions create the illusion of providing people with computational skills. In reality, the use of these tools does not provide any of the training required to scale the next summit. Ironically, the mere existence of these methods hinders science instead of promoting discovery.

The challenges of biology are part of a much larger **computational challenge**.

We lack the proper representation, terminology, and interpretation of the phenomena that we observe. We have to continually adapt and make decisions then choose the correct methods then apply them appropriately based

²https://en.wikipedia.org/wiki/False_peak

on a deeper understanding of what we did.

Becoming successful in bioinformatics means to be **computationally proficient**. Only then you can scale the subsequent computational peaks independently, rather than using the crutches and dead-end solutions that well-meaning, yet computationally illiterate scientists have built for you.

You are in the right place, though, from this book, you'll learn computation the way Mother Nature intended it to work.

8.2 Don't Let Architecture Astronauts Scare You



We all want to adopt the best practices, keep up with development, embrace progress. But we have to be careful when doing so. Someone being smarter than us does not always mean they are doing more intelligent things. Let me refer you to the seminal blog titled:

- Don't Let Architecture Astronauts Scare You³

In that blog Joel Spolsky makes the following salient point:

When great thinkers think about problems, they start to see patterns.

[...]

Sometimes smart thinkers just don't know when to stop, and they create these absurd, all-encompassing, high-level pictures of the universe that are all good and fine but don't actually mean anything at all.

[...]

³<https://www.joelonsoftware.com/2001/04/21/dont-let-architecture-astronauts-scare-you/>

These are the people I call **Architecture Astronauts**. It's very hard to get them to write code or design programs because they won't stop thinking about **Architecture**.

If you want actually to get things done, you have to recognize the astronauts early on.

Bioinformatics is fertile ground for these astronauts. Biologists are not experts in computing; they all “want to believe.” They always hold out the hope out there is a silver bullet, that a single and magical solution explains everything. As an example just about every biologist seems to hope that their incredibly complicated phenotype can be traced back to a few SNPs.

No wonder that **Architecture Astronauts** thrive in these environments.

8.3 The Curse of Docker



What is Docker⁴? Wikipedia says:

Docker is a set of platform-as-a-service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines.

I'll make this brief by formulating a recommendation that I will call: “The curse of Docker”:

⁴[https://en.wikipedia.org/wiki/Docker_\(software\)](https://en.wikipedia.org/wiki/Docker_(software))

In my experience when you hear “docker” mentioned in any positive way in the context of bioinformatics, it is exceedingly likely that at least one (and often all) of the following statements below are true:

1. The person that uttered those words does not understand **any bioinformatics at all**.
2. The person that recommends Docker will directly or indirectly benefit from you using Docker. They are hoping to lock you into a solution that only they can provide, thus you become dependent on them.
3. The bioinformatics software that needs Docker is usually terrible, poorly written, full of bugs and overall follows awful software practices. That’s why it requires Docker! Software written by smart people does not need Docker. By using Docker, you are selecting for worse.
4. The more time you spend with Docker, the worse off you’ll be. You dig a hole that will only get deeper (see the sunken cost fallacy⁵).



Biostar Theorem of the Day

Any software that needs Docker can be replaced by better, faster, and simpler software that does not need Docker.

Thus, if at any time, if you hear the word **Docker** recommended in any bioinformatics context, run away as fast as possible. Don’t stop, don’t look back - or you’ll never get anything done.

8.4 Cloud computing: Amazon Compute Cloud



Everything written for the section titled The Curse of Docker applies the same way.

⁵https://en.wikipedia.org/wiki/Sunk_cost

Using either Docker or the compute cloud will demand excessive efforts and putting up with massive information overhead - all gratuitous and nonsensical, all driven by corporate politics, commercial interests, and bureaucratic agendas.

You'll be neither smarter nor understand anything more about biology than you did before embarking on understanding these processes.

The people that teach you Docker and Amazon Cloud *before* teaching you bioinformatics cause you more intellectual harm than good.

8.5 Large scale data analysis platforms

When someone promises you everything, the most likely outcome is that you get nothing.



For example, take CyVerse⁶, funded by the National Science Foundation's Directorate for Biological Sciences note how they claim:

CyVerse provides life scientists with powerful computational infrastructure to handle huge datasets and complex analyses, thus enabling data-driven discovery.

Astronautics is quite strong here; buzzwords abound, amusingly, they even have Atmosphere⁷ product.

Note how the page discussing the terms of using their logo⁸ is more expansive than the explanation of a typical data analysis protocol.

Poke around the site; note how they don't teach you how to do something yourself. You cannot learn anything meaningful about science or bioinformat-

⁶<https://www.cyverse.org/>

⁷<https://www.cyverse.org/atmosphere>

⁸<https://www.cyverse.org/use-logo>

ics. It is always about click here, click there, no lasting knowledge, all about clicking, as if their interface and whatever half-baked approach they have listed there would be the perfect data analysis process. Not surprisingly and quite tellingly: *they love Docker**, they offer regular training for Docker!

At no point in their rambling, are they close to anything that could be considered a rational scientific thought process. Everyone visiting the site is now dumber for having clicked the link that led them there. They deserve no points, and may God have mercy on their soul.

8.6 The New York Times Test



Some of you are looking for projects that you can graduate from. Some promising plans and research directions do not pan out. Through tireless work and dedication, I have discovered a fool-proof method that has high accuracy in predicting if a scientific discovery is false.



95% of life-science oriented discoveries mentioned in a New York Times⁹ news release are false.

So there you have a straightforward way to pick winning projects.

8.7 Designed by a committee: Common Workflow Language (CWL)



I was giving a presentation on scripting, showed off recipes that automated the re-analysis of entire publications in less than ten lines of code. A biologist in the audience asked me the following question:

-Wouldn't it have been a better option to use the Common Workflow Language?

I asked him in what way would the CWL be of help. It turns out he did not know anything at all about the CWL - just that he thought I would be better off had I used it.

Wow! I thought to myself. Now that's excellent marketing! They sold a biologist on CWL, with him being unable to articulate a single benefit of it.

Ok, so let's talk CWL. By their own definition:

The Common Workflow Language (CWL)¹⁰ is an open standard for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high-performance computing (HPC) environments. CWL is designed to meet the needs of data-intensive science, such as Bioinformatics, Medical Imaging, Astronomy, High Energy Physics, and Machine Learning.

Let's see how it works in practice. Imagine that you want to print "Hello World" to the screen. Normally, in a shell you could do it like this:

```
echo "Hello World"
```

and it would print:

```
Hello World
```

Done!

If you were *to solve* this same task via the Common Workflow Language¹¹ you would first need to write a document called `hello.cwl`:

```
#!/usr/bin/env cwl-runner
```

```
cwlVersion: v1.0
```

¹¹https://www.commonwl.org/user_guide/02-1st-example/index.html

```

class: CommandLineTool
baseCommand: echo
inputs:
  message:
    type: string
    inputBinding:
      position: 1
outputs: []

```

In the next step, you would need to create another file called `echo-job.yml` that contains:

```
message: Hello world!
```

Finally, in the next step you would *execute* the code with:

```
cwl-runner 1st-tool.cwl echo-job.yml
```

that in turn will print the following:

```
[job 1st-tool.cwl] /tmp/tmpmM5S_1$ echo \
  'Hello world!'
Hello world!
[job 1st-tool.cwl] completed success
{}
Final process status is success
```

Compare the two solutions! I find this latter both ridiculous and frankly ineffectual.

Of course, some could (and I am sure many will) argue that the example above is *too simple* and that in a *realistic scenario* CWL will pay other dividends. Alas no.

My experience is the opposite. It is only downhill from here on out.

On the other hand, do you know what works well with **CWL? Docker** and **Cloud Computing!**

8.8 Time to pay the Iron Price

I could go on, but I think I made enough enemies as is.

I have been laughing writing this chapter, but I know I am going to *pay the iron price* for this little moment of fun.

Scientists have long memories!

Part II

Installation

Chapter 9

How to set up your computer

Practicing bioinformatics will require the installation of software tools distributed in various formats:

1. Binary “executable” code that can be run directly on the target computer.
2. “Source code” that needs to be “compiled” to create a “binary” program.
3. Programs that require the presence of another programming language: `java`, `python`, or `perl`.

9.1 How do I set up my computer?

Follow each of the steps below. We may instruct you to visit other pages but after you do visit those make sure to come back to this page.

Ensure that you followed each of the steps and the corresponding subsections.

1. Prepare your computer.
2. Install conda and activate bioconda.
3. Install the software.

9.2 Is this going to be difficult?

Well, truth to be told it can be a bit tedious. But you only need to do this once, so bear with it.

The installation process may take an hour or more, along the way it will print copious amounts of information on your screen. While you don't have to watch the cryptic sentences go by in real time, the process will, at times stop and ask you permissions or some questions (yes/no). The choice is to select or type yes unless otherwise instructed by the book.

You may need to enter a password, this is either the Linux user password or your Mac user password. In each case, the user must have administrative privileges.

Important: Follow the instructions in order!

9.3 How do I prepare my computer?

Before installing bioinformatic software, you have to prepare your computer for the tasks ahead.

We have separate instructions for MacOS, Linux and Windows computers. Follow only the instructions for your platform.

There will be multiple steps that you have to follow, the processes may take some time, possibly hours. Make sure to complete the system setup before venturing forth:

- Setting up a Mac OS computer
- Setting up a Windows 10 computer
- Setting up a Linux computer

Once you followed the instruction above continue on below.

9.4 How do I initialize my terminal?



Do not run this terminal initialization command above more than once!

Now that your computer is set up as described above you will need a few lines in your setup files. Open a Terminal and run the following:

```
curl http://data.biostarhandbook.com/install/bash_profile.txt >> ~/.bash_profile  
curl http://data.biostarhandbook.com/install/bashrc.txt >> ~/.bashrc
```

These commands will update your so-called “shell” (see later) to behave more consistently. If you want the details read the Setting the Bash profile page.

The new settings won’t apply right away to your current terminal window. To have them applied, you will need to either close then open the terminal again or type:

```
source ~/.bash_profile
```

Once your system is set up properly continue on with the section How to install conda.

Chapter 10

How to install conda

Steps that need to be followed:

1. Install conda
2. Update conda to the latest version
3. Activate the bioconda channels

10.1 What are environments?

Modern software packages are interdependent. Each program may rely on the presence of one or more other software. In turn, each required software may also need another program to be present - and so on - down the rabbit hole we go,

Then, software A may only inter-operate appropriately with a specific version of software B. For that reason there may be conflicts between the different requirements. These conflicts are solved by creating isolated installations where only the compatible software are grouped.

The isolated installations are called *environments*.

To set up and manage our environment we will use the conda installer on all platforms. On MacOS we may also use the `homebrew` package manager in addition to `conda`.

Conda¹ is an open source package management system and environment management system for installing multiple versions of software packages and their dependencies and switching easily between them. It works on Linux, OS X and Windows, and was created for Python programs but can package and distribute any software.

We will use a version of it called `bioconda` customized to install bioinformatics packages. You will need to install `conda` then tap the channels that correspond to `bioconda`.

10.2 How do I install conda?

Start with by installing miniconda² a simplified version of `conda` itself.

Click the animation below to see what the conda installation might look like on your computer, then follow the steps listed below.

10.3 Step-by-step instruction for installing conda

For MacOS the link to the installer is :

URL=https://repo.continuum.io/miniconda/Miniconda3-latest-MacOSX-x86_64.sh

For Linux and Windows Bash the URL is:

URL=https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh

To download the installer type:

```
# Download the installer.  
curl $URL > miniconda-installer.sh
```

```
# Run the installer.  
bash miniconda-installer.sh -b
```

¹<https://conda.io/miniconda.html>

²<https://conda.io/miniconda.html>

Once the installation completes you will need to initialize the conda environment with:

```
miniconda3/condabin/conda init
```

Note how it states:

```
==> For changes to take effect, close and re-open your current shell. <==
```

Close the terminal and open a new one to make conda activate. Check that it worked by typing:

```
conda
```

10.4 How to update conda

The downloaded package may not even be the most up-to-date version. Every few months it may be a good idea to check for an updated version of conda.

Click the next animation to start it. It shows the steps needed to finalize the installation process.

Check the version:

```
conda -V
```

update the conda package itself to bring it to the latest version:

```
conda update -y -n base conda
```

See if your version changed:

```
conda -V
```

Close and reopen the terminal so that the latest conda activates.

10.5 What is bioconda?

Bioconda is a channel for distributing bioinformatics software that can be enabled in the Conda package manager. Run the following in your terminal:

```
conda config --add channels bioconda  
conda config --add channels conda-forge
```

The **bioconda** channel is now active within the **conda** package manager.

Now move onto the How to install software page.

Chapter 11

How to install software

Steps that need to be followed:

1. Create the bioinformatics environment
2. Activate the bioinformatics environment
3. Install the bioinformatics software

11.1 Installation overview

Click the next animation to start it. It shows the steps needed to install the software on your computer.

11.2 How to create a bioinformatics environment

Once you have installed and updated `conda` create an environment.

```
# Create a bioinformatics enviroment called bioinfo.  
conda create -y --name bioinfo python=3.6
```

Close and reopen your terminal. This is the final time you need to reopen your terminal during installation :-)

11.3 How to activate and install bioinformatics software

The list of bioinformatics tools installed with the command below can be seen in conda.txt¹. To proceed follow the commands below:

```
# Activate the bioinformatics environment.
```

```
conda activate bioinfo
```

```
# Install most bioinformatics tools covered in the Biostar Handbook.
```

```
curl http://data.biostarhandbook.com/install/conda.txt | xargs conda install -y
```

The installation above will take a bit of time to complete. Note that you will need to run:

```
conda activate bioinfo
```

once in every new terminal you open to activate the bioinformatics environment.

11.4 How do I check that Entrez Direct works?

Since Entrez Direct is the tool that seems to cause most problems for our readers we recommend that you verify right away that it works. Once the installation has completed, run the following to verify that you can access NCBI from the command line:

```
efetch -db nuccore -id 2 -format gb
```

if the command above produces:

```
LOCUS      A00002          194 bp  DNA  linear  PAT 10-FEB-1993
DEFINITION B.taurus DNA sequence 1 from patent application EP0238993.
ACCESSION  A00002
VERSION    A00002.1
...
JOURNAL    Patent: EP 0238993-A2 1 30-SEP-1987;
```

¹<http://data.biostarhandbook.com/install/conda.txt>

...

then your installation is successful, and you are done. Move onto the next chapter. Before you go note how this sequence (numbered with 2, so it is the second sequence ever deposited in the NCBI) comes from a patent application applied to the cow genome submitted initially in 1987. You see the legality of patenting biological information is an issue that is not yet fully resolved. You may also want to reflect on the appropriateness of patenting life itself.

11.5 How do I verify that all other programs work?

We have created a script that verifies that your settings are correct. Throughout the book, if any of our instructions causes an error, your first choice ought to be to check what the doctor says :-) In our experience when the “doctor” is happy all is well on your system:

Download and run our `doctor.py`² script from a terminal:

```
mkdir -p ~/bin
curl http://data.biostarhandbook.com/install/doctor.py > ~/bin/doctor.py
chmod +x ~/bin/doctor.py
```

When run from a terminal the `doctor.py` will tell you what (if anything) is out of order:

```
~/bin/doctor.py
```

The doctor will report errors that you might have.

There may be a few optional programs that the `doctor.py` will detect and report for example:

```
Optional program not found: global-align.sh
```

These messages do not indicate a setup problem. At the beginning of the relevant chapters you will be instructed to install these optional programs.

²<http://data.biostarhandbook.com/install/doctor.py>

11.6 How do I fix installation problems?

Occasionally your installation may break for various reasons. Perhaps you have installed different tools, explored other software that changed your setup. Sometimes you may find that even after uninstalling tools your system may exhibit various errors that were not present before. It is typically quite challenging to fix these so in our experience the best solution is to start anew, make a new environment and reinstall everything. For us, this usually solves the problem. The steps to do a full reinstall by removing the previous environment and resetting everything are:

```
conda create -y --name bioinfo python=3.6
conda activate bioinfo
curl http://data.biostarhandbook.com/install/conda.txt | xargs conda install -y
```

Alternatively if you can create a new environment with a different name and install into that.

11.7 How do I report installation problems?

While we are doing our best to keep the book up-to-date certain problems may go unnoticed for short periods of time.

Twice a year the installation instructions are tested in a classroom setting where wide variety of computing platforms are in use: Windows, Mac and Linux. So we know the instructions are tested twice a year in a wide range of systems, but even so problems may still occur from time to time.

The book has an issue tracker that you may consult to see whether other people had problems similar to yours. This is also the place where you can report new issues and problems that appear to have not been discussed yet:

- Biostar Handbook Issue Tracker³

The tracker above may also be used to report errors or inconsistencies in the content of the book.

³<https://github.com/biostars/biostar-handbook-issues>

11.8 How do I use `conda` in general?

Once `conda` is installed as above you can create environments, for example we created the `bioinfo` with:

```
conda create -y --name bioinfo python=3.6
```

To activate the `bioinfo` environment (this needs to be done in each new window you open) type :

```
conda activate bioinfo
```

You may create multiple environments on your system, sometimes creating new environments is necessary to run software with different requirements and dependencies.

Note: in prior versions of `conda` the command to activate environment was `source activate bioinfo`, if you see this advice remember to replace it with `conda activate`

You can list all existing environments with:

```
conda info --envs
```

You can learn more about managing environments on the `conda` webpage⁴

11.9 How do I install a new tool?

While our instructions install all the tools that are used in the book, you may occasionally find yourself needing a program that we did not include. In that case, your first approach would be to see if `conda` already covers the tool you want to install. Most of the time you may be able to install into the environment `bioinfo`

```
conda activate bioinfo  
conda install toolname
```

This command will collect the requirements and it will ask you if you want to proceed. Read the output carefully and make a note of any warning it indicates. It is possible that some tools cannot operate together. In those cases, you can create a new environment just for the new tool.

⁴<https://conda.io/docs/using/envs.html>

```
conda create --name mynewthing -y  
conda activate mynewthing  
conda install toolname
```

To run the new tool you'd have to be in `mynewthing` environment. To avoid having to switch back and forth, you may open a terminal and activate the `bioinfo` environment then open another terminal and activate the `mynewthing` environment there.

Now in some cases, software needs to be installed from the “source” code. Source code installation typically involves downloading the source code, unpacking the software and following the installation instructions. Sometimes this means running either `make` or some other command. We do list the source code installation for a few tools in the appendix.

11.10 How do I upgrade a tool?

Tools may have updated versions released.

```
# Upgrade a package.  
conda update toolname
```

11.11 Do the automatically installed tools always work?

Alas no. Bioinformatics software can become outdated. Tools packagers (`conda` or `brew`) are not always aware or able to integrate these changes. As a result, tools installed via these may suddenly not work until the package maintainers update their formulas.

In those cases, visit the source code instruction page in the web and install manually from the source.

11.12 How do I update conda?

Periodically it may be necessary to update your conda manager. You can do so via

```
conda update -n base conda
```

11.13 How should I set up my file structure?

Many people have strong feelings on how a filesystem should be set up when using Unix command line for scientific projects. Here are some opinions:

- A Quick Guide to Organizing Computational Biology Projects⁵
- A research workflow based on GitHub⁶
- Directory layout advice⁷

In our view, the most important concept is that of simplicity. We typically create the following directories:

- Links to binaries go into `~/bin`.
- Reference genomes go into `~/refs`.
- Source code for tools go into `~/src`.

You can come up with appropriate names for other folders. We like to use `work`, `project` etc.

Just keep the names short and simple because you will have to enter or read them a lot. We often need to type full paths to various files, and in those cases, it helps if the file system paths look like

- `~/refs/hg19.fa`

What you don't want are paths that look like:

- `/My Bioinformatics Work/Documents and Folders/Unfinished Projects/2016/factor Binding/todo3/mygenome/b19/human.fa`

You will feel the urge to make your names very long and descriptive. You will want to do this because you'll be afraid that you will forget what you

⁵<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424>

⁶<http://www.carlboettiger.info/2012/05/06/research-workflow.html>

⁷<http://nicercode.github.io/blog/2013-04-05-projects/>

did. While that is a valid concern, long file names are not a right approach to handle this information complexity.

What you need is a proper track record of what you did. We'll teach you how to do that. You'll learn to write scripts where the script itself documents what each file contains. So don't worry about short file names. Keep it simple.

11.14 What to do if I get stuck?

While we are doing our best to keep the book current, some problems may go unnoticed for short periods. The software is updated all the time whereas installation takes place only once on each computer - and you typically do not need to redo that again. Hence instructions that were valid last month may be outdated later.

Twice a year the installation instructions are tested in a classroom setting where a wide variety of computing platforms are in use: Windows, Mac, and Linux.

The book has an issue tracker that you may consult to see whether other people had problems similar to yours. The issue tracker is also the place where you can report new issues and problems that have not yet been discussed:

- Biostar Handbook Issue Tracker⁸

11.15 Why is installation so tedious?

Finally, you may ask yourself, why is it that we can put people on the Moon, our phone is a supercomputer in our pocket, yet we still have to close and reopen a terminal just to make an installation stick.

Not to mention the endless hoops that we have to jump through...



It reminds me of the fable where God created all creatures, his favorite creature was the donkey, an unassuming, modest, hardworking animal.

⁸<https://github.com/biostars/biostar-handbook-issues>

One day the donkey asks for an audience with God and tells him the following.

- *My Lord! I have become the laughing stock humanity. They make jokes on my behalf, they use me as an example for stubborn stupidity, they are even calling me an “Ass”. My Lord! You are so powerful! Can you please make the stigma go away?*

God is saddened, ponders the situation, sighs, then says:

- *Well, I'm really sorry! Changing that is beyond my powers...*

The donkey is puzzled:

- *But why? What could be more powerful than God?*

To which God replies:

- *Consensus opinion*



Figure 11.1

When it comes to software installation an analogous situation occurs.

Individually each step may be tolerable, and the consensus opinion is that all it takes a bit of effort to get through. Yet all together it becomes a miserable experience one that is impossible to change as a large number of distinct people and organizations each with their own beliefs and agendas would need to agree at the same time on the same solution.

Chapter 12

Choose and install a text editor

This section is short but so essential that we placed it into a separate chapter. Many examples, scripts, and assignments will ask you to view, edit or generate text files.

To create, investigate and modify these files you will need a proper **text** editor. And no, Microsoft Word is not a text editor. Using Word to edit text will eventually cause (devious) errors. It's best if you learn to use a proper text editor from the start.

Keep in mind that a substantial number of bioinformatics file formats use the so-called “tab” character to separate columns. Thus the need to use an editor that supports working with tab-delimited data.

12.1 What features should my text editor have?

Essential features that your editor needs to have:

- The editor needs to have the ability to show so-called **tab** characters versus **space** characters. Both are invisible, one-byte characters, that by eye will look like one or more spaces. Data that, for some reason, end up using a tab instead of space (or vice versa) will cause incredibly confusing errors. You will have a hard time tracking down these errors unless you can visualize the two normally invisible characters. Note a

common gotcha: pasting from a web page will turn tabs into spaces even if the author of the web page wrote tabs into the page! Your editor does not always have to show these, but you need to know how to turn on the visualization of these white spaces.

- The editor should be able to display line numbers. Frequently this feature lets you identify the line where the error occurs.
- The editor has to be able to switch line endings. Line endings have variants: Unix, Windows. Since Mac OS X (late 2002) the Mac is using Unix line endings. As a rule, all your files should always have Unix line endings, and you should easily be able to convert line endings to Unix format.

12.2 Viewing whitespace

Here is a situation where words on both rows appear to be separated by tabs when in reality the two lines differ. On the first line, three spaces separate the words. On the second line, a single tab character separates the two words.

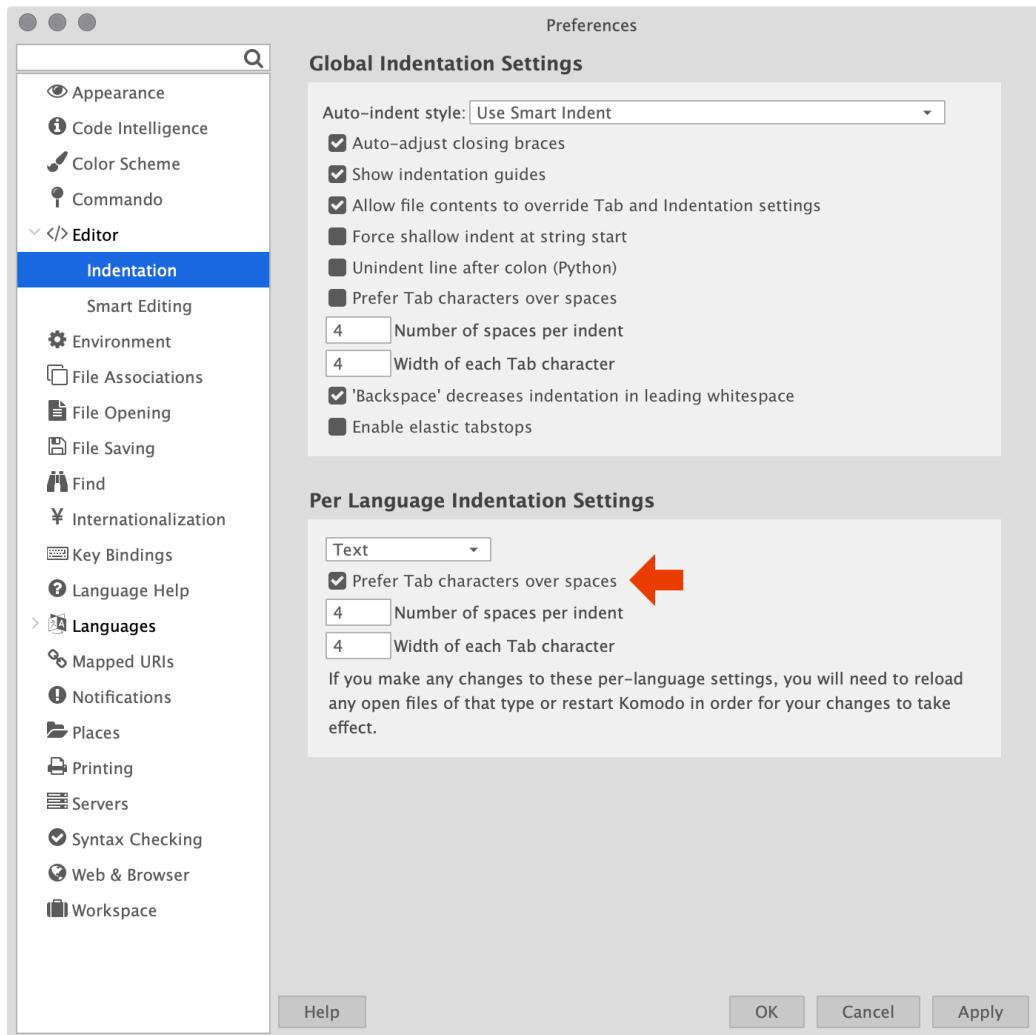
The left image shows the typical view in Komodo Edit. The right-hand side image shows the same file with “View Whitespace” option turned on. The so-called “white-space” content between the words is now visible.



12.3 Super annoying behaviors

Some text editors (Komodo Edit is one of them), will, by default, insert multiple spaces even when you press the TAB key. You have to explicitly configure Komodo to have it enter the TAB character when you press the TAB key. This behavior is immensely confusing (and annoying) at first. Hence make sure to set up your text editor to insert TAB when you press TAB.

In Komodo Edit you need to navigate to preferences and tick a checkbox to make it do what it is supposed to.



12.4 Which text editor to choose?

The choice of an editor is somewhat personal preference. You may already have an editor that you like. If not the following options are a good starting point:

- Komodo Edit¹ - works on all operating systems
- Sublime Text 2² - all operating systems
- NotePad++³ - works on Windows only (would be our preference on Windows)

I use PyCharm⁴ for both text editing and programming.

12.5 Watch the line endings on Windows!

Currently there are two kinds of line endings in use: UNIX (LF) style and Windows style (CRLF) line termination characters. If you are running Windows Bash and use a Windows text editor from Windows you have to ensure that the files that you create will use UNIX line endings. This is typically a setting on the file, and is only necessary when you start a new file.

Having Windows line endings instead of Unix line endings will lead to various mysterious errors.

Below we link to a short explanation:

- The Great Newline Schism⁵

Google the concept if you want to learn more.

¹<https://www.activestate.com/komodo-ide/downloads/edit>

²<http://www.sublimetext.com/2>

³<https://notepad-plus-plus.org/>

⁴<https://www.jetbrains.com/pycharm/>

⁵<https://blog.codinghorror.com/the-great-newline-schism/>

Part III

UNIX COMMAND LINE

Chapter 13

Introduction to Unix

The terms “**Unix**” and “**Command Line**” are used interchangeably both in this Handbook and in the real world. Technically speaking, Unix is a class of operating system that originally used the command line as the primary mode of interaction with the user. As computer technologies have progressed, the two terms have come to refer to the same thing.

13.1 What is the command line?

People interact with computers via so-called **user interfaces**. Clicking an icon, scrolling with the mouse, and entering text into a window are all user interfaces by which we instruct the computer to perform a specific task.

When it comes to data analysis, one of the most “ancient” of interfaces—the text interface—is still the most powerful way of passing instructions to the computer.

In a nutshell, instead of clicking and pressing buttons you will be entering *words* and *symbols* that will act as direct commands and will instruct the computer to perform various processes.

13.2 What does the command line look like?

For the uninitiated command line instructions may look like a magical incantation:

```
cat sraids.txt | parallel fastq-dump -O sra --split-files {}
```

In a way it is modern magic - with just a few words we can unleash complex computations that would be impossible to perform in any other way. With the command line, we can slay the dragons and still be home by six.

As for the above – and don’t worry if you can’t follow this yet – it is a way to download all the published sequencing data that corresponds to the ids stored in the `sraids.txt` file. The command as listed above invokes three other commands: a funny one named `cat`, a more logical sounding one `parallel` and a mysterious `fastq-dump`.

This series of commands will read the content of the `sraids.txt` and will feed that line by line into the `parallel` program that in turn will launch as many `fastq-dump` programs as there are CPU cores on the machine. This parallel execution typically speeds up processing a great deal as most computers can execute more than one process at a time.

13.3 What are the advantages of the command line?

The command line presents a word by word representation of a series of actions that are explicit, shareable, repeatable and automatable. The ability to express actions with words allows us to organize these actions in the most appropriate way to solve every problem.

13.4 What are the disadvantages of the command line?

Running command line tools requires additional training and a more indepth understanding of how computers operate. Do note that this knowledge is quite valuable beyond the domain of bioinformatics.

13.5 Is knowing the command line necessary?

Yes. While it is entirely possible to perform many bioinformatics analyses without running a command line tool most studies will benefit immensely from using a Unix like environment. Most importantly understanding Unix develops a way of thinking that is well suited to solving bioinformatics problems.

13.6 Is the command line hard to learn?

That is not the right way to think about it.

Understand that learning the command line (and programming in general) is not just a matter of learning the concepts. Instead, it is primarily about learning to think a certain way - to decompose a problem into very simple steps each of which can be easily solved. Some people have an affinity to this – they will pick it up faster, while others will need more practice. In our observation, prior preparation/experience does not affect the speed at which people can learn the command line.

At the same time, you have to realize that you can only succeed at changing how you think when you perform these actions. Besides it also means that you can't just learn it all in a day or even a week. So keep at it for a little while longer building your skills. In the end, it is only a matter of time and practice.

Try to actively make use of command line tools by integrating them into your daily work. Making use of command line tools is usually quite easy to do since Unix tools are useful for just about any type of data analysis.

13.7 Do other people also make many mistakes?

When you start out using the command line you will continuously make errors. That can be very frustrating. You could ask yourself, “Do other people make lots of mistakes too?”

Yes, we all do. We correct these errors faster.

Your skill should be measured not just in making fewer mistakes but primarily in how quickly you fix the errors that you do make. So get on with making mistakes, that's how you know you are on the right path.

13.8 How much Unix do I need to know to be able to progress?

A beginner level Unix expertise is sufficient to get started and to perform the majority of analyses that we cover in the book.

You should have a firm grasp of the following:

1. Directory navigation: what the directory tree is, how to navigate and move around with `cd`
2. Absolute and relative paths: how to access files located in directories
3. What simple Unix commands do: `ls`, `mv`, `rm`, `mkdir`, `cat`, `man`
4. Getting help: how to find out more on what a unix command does
5. What are “flags”: how to customize typical unix programs `ls` vs `ls -l`
6. Shell redirection: what is the standard input and output, how to “pipe” or redirect the output of one program into the input of the other

13.9 How do I access the command line?

When using graphical user interfaces the command line shell is accessed via an application called “Terminal”. When the “Terminal” runs it launches a so-called “shell” that by default opens into a folder that is called your “home” directory.

13.10 What is a shell?

The shell is an interface that interprets and executes the commands that entered into it via words. There are different types of shells, the most commonly used shell is called `bash`. Below we run the `echo` command in `bash`

```
echo 'Hello World!'
```

and we see it executed in a Mac OSX Terminal below. The default terminal that opens up for you will probably look a bit different. There are options to change the fonts, background colors etc. Besides, our so-called command prompt (the line where you enter the commands), is probably simpler than yours. See the Setup profile for details on how to change that.

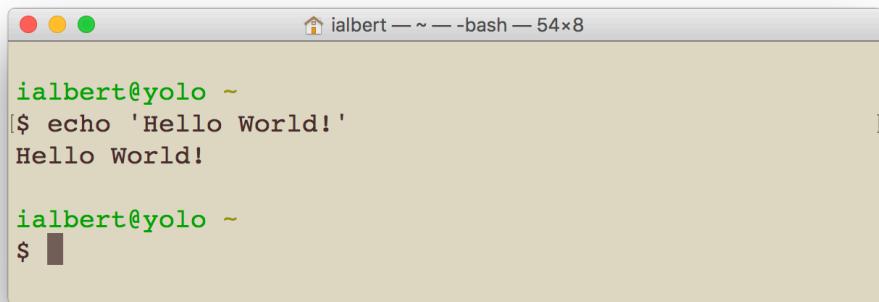


Figure 13.1

It is essential to recognize that the shell is not just a way to launch commands but a **full programming language** that provides many powerful constructs to streamline its use. Alas, initially these powers tend to get in the way and cause some frustrations. Suppose that instead of

```
echo 'Hello World!'
```

you've enclosed the text with double quotes:

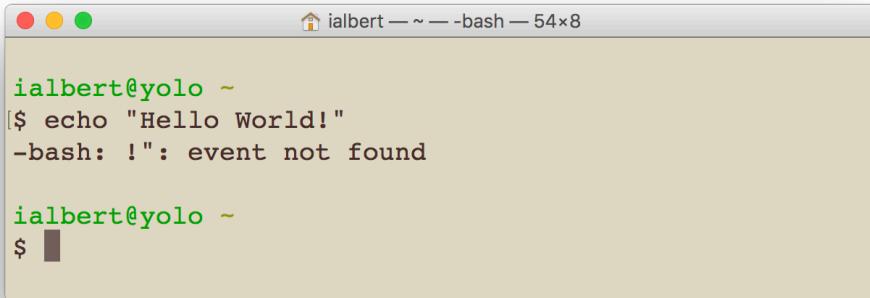
```
echo "Hello World!"
```

now instead of echoing the output, it will produce:

Maddeningly enough the error occurs only if the string contains the ! character as the last element. For example, this works just fine.

```
echo "Hello World"
```

Understandably getting mysterious errors when writing seemingly simple text can be quite confusing. The error occurred because in `bash` the ! character allows you to rerun a previous command. For example



A screenshot of a macOS terminal window titled "ialbert — ~ — bash — 54x8". The window shows a command-line session:

```
ialbert@yolo ~
$ echo "Hello World!"
-bash: !": event not found

ialbert@yolo ~
$ █
```

Figure 13.2

!e

Will rerun the last command that starts with e. In this case the first echo 'Hello World!'. The feature was added as help to avoid retyping long commands. Tip: try !! see what happens.

Then, as it turns out in bash text enclosed within double quotes is interpreted differently than text enclosed in single quotes. Single quoted content is passed down exactly as written, whereas in double-quoted text some letters that have may special meaning like: !, \$ (and others) will be interpreted as bash commands. Initially this gets in the way, as you don't know which letter is "special". Later this feature allows you to build more powerful commands.

As a rule, in bash type single quotes if you want the text the stay the same, and type double quotes when the text needs to be modified later on.

Gotcha's like these are numerous. Everyone gets tripped up at some time. Google is your friend. Bash has been around for a long time (decades) and did not change much. Every problem you run into has most likely been experienced by others as well, and will have solutions written out in great detail.

13.11 What is the best way to learn Unix?

We offer a number of tutorials right here in this book.

In addition there are excellent online tutorials, some free others at cost, each with a different take and philosophy on how to teach the concepts. We recommend that you try out a few and stick with the one that seems to work the best for you.

- CodeAcademy: Learn the command line¹
- Software Carpentry: The Unix shell²
- Command line bootcamp³
- Unix and Perl Primer for Biologists⁴
- Learn Enough Command Line to Be Dangerous⁵
- The Command Line Crash Course⁶
- Learn Bash in Y minutes⁷

And there are many other options.

13.12 How do I troubleshoot errors?

If you don't know how to do or fix something try Googling it. Put the relevant section of the error message into your search query. You will find that other people will have had the same problem and someone will have told them what the answer was. Google will likely direct you to one of these sites

- Ask Ubuntu⁸
- StackOverflow⁹

¹<https://www.codecademy.com/learn/learn-the-command-line>

²<http://swcarpentry.github.io/shell-novice/>

³<http://korflab.ucdavis.edu/bootcamp.html>

⁴http://korflab.ucdavis.edu/Unix_and_Perl/current.html

⁵<https://www.learnenough.com/command-line-tutorial>

⁶<http://cli.learncodethehardway.org/book/>

⁷<https://learnxinyminutes.com/docs/bash/>

⁸<http://askubuntu.com/>

⁹<http://stackoverflow.com/>

13.13 Where can I learn more about the shell?

- Command Reference¹⁰
- Explain Shell¹¹

¹⁰<http://files.fosswire.com/2007/08/fwunixref.pdf>

¹¹<http://explainshell.com>

Chapter 14

The Unix bootcamp

The following document has been adapted from the Command-line Bootcamp¹ by Keith Bradnam² licensed via Creative Commons Attribution 4.0 International License. The original content has been substantially reworked, abbreviated and simplified.

14.1 Introduction

This ‘bootcamp’ is intended to provide the reader with a basic overview of essential Unix/Linux commands that will allow them to navigate a file system and move, copy, edit files. It will also introduce a brief overview of some ‘power’ commands in Unix.

14.2 Why Unix?

The Unix operating system³ has been around since 1969. Back then, there was no such thing as a graphical user interface. You typed everything. It may seem archaic to use a keyboard to issue commands today, but it’s much easier to automate keyboard tasks than mouse tasks. There are several variants of

¹<http://korflab.ucdavis.edu/bootcamp.html>

²<http://www.keithbradnam.com/>

³<http://en.wikipedia.org/wiki/Unix>

Unix (including Linux⁴), though the differences do not matter much for most basic functions.

Increasingly, the raw output of biological research exists as *in silico* data, usually in the form of large text files. Unix is particularly suited to working with such files and has several powerful (and flexible) commands that can process your data for you. The real strength of learning Unix is that most of these commands can be combined in an almost unlimited fashion. So if you can learn just five Unix commands, you will be able to do a lot more than just five things.

14.3 Typeset Conventions

Command-line examples that you are meant to type into a terminal window will be shown indented in a constant-width font, e.g.

```
ls -lrh
```

The lessons from this point onwards will assume very little apart from the following:

1. You have access to a Unix/Linux system
2. You know how to launch a terminal program on that system
3. You have a home directory where you can create/edit new files

14.4 1. The Terminal

A *terminal* is the common name for the program that does two main things. It allows you to type input to the computer (i.e. run programs, move/view files etc.) and it allows you to see output from those programs. All Unix machines will have a terminal program available.

Open the terminal application. You should now see something that looks like the following:

Your terminal may not look like the image above. Depending on how your system is set up, it may even look like the image below:

⁴<http://en.wikipedia.org/wiki/Linux>

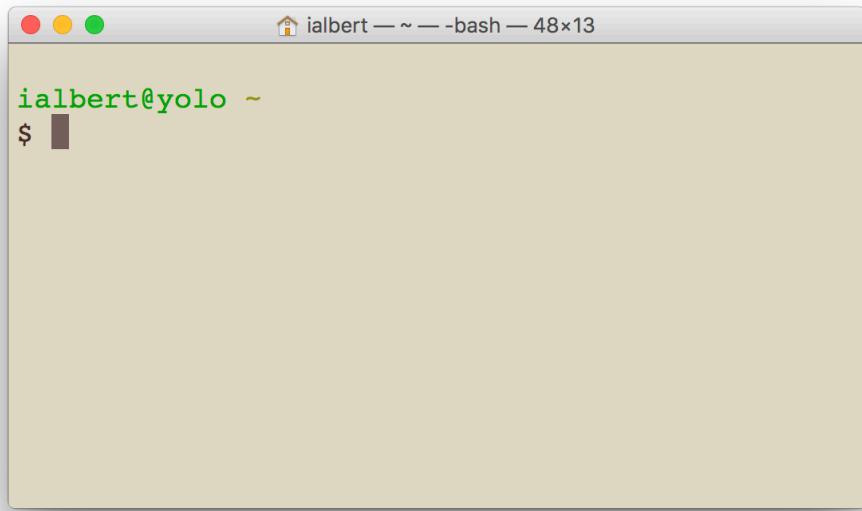


Figure 14.1

If your terminal is not easily readable customize it. Make the fonts BIGGER, make the background lighter. On a Mac select Preferences, on Windows right click the top bar and set more appropriate colors and font.

Understand that learning Unix is also a “physical” skill that requires a sort of hand-eye coordination. You have to teach your body to cooperate. You have to be comfortable, you have to see the commands clearly and with no effort. The presence of a single character in a multiline pipeline may change the meaning of the entire process. Ensure that you have minimal distractions and that the text in your terminal is eminently legible.

There will be many situations where it will be useful to have multiple terminals open and it will be a matter of preference as to whether you want to have multiple windows, or one window with multiple tabs (there are typically keyboard shortcuts for switching between windows, or moving between tabs).

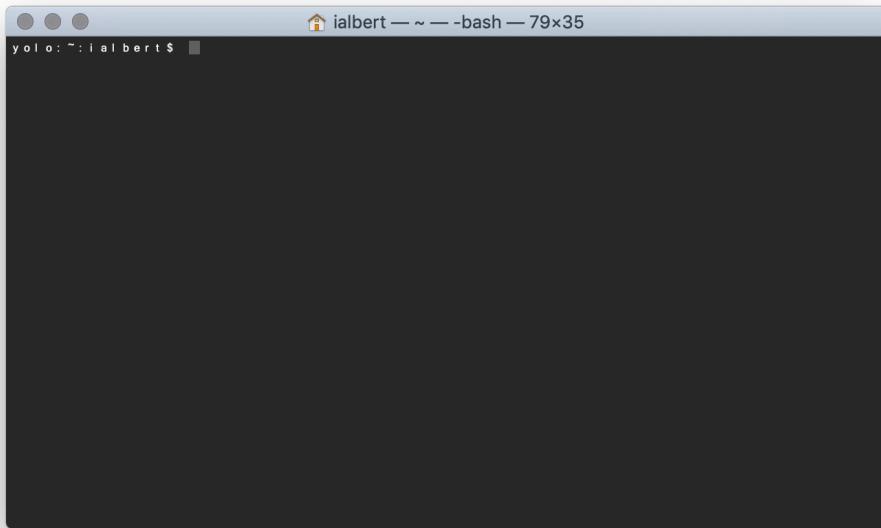


Figure 14.2

14.5 2. Your first Unix command

It is important to note that you will always be *inside* a single directory when using the terminal. The default behavior is that when you open a new terminal you start in your own *home* directory (containing files and directories that only you can modify). To see what files and directories are in your home directory, you need to use the `ls` command. This command lists the contents of a directory. If we run the `ls` command we should see something like:

```
ls
```

prints (this depends on what computer you use):

`Applications Desktop Documents Downloads`

There are four things that you should note here:

1. You will probably see different output to what is shown here, it depends on your operating system and computer setup. Don't worry about that

for now.

2. In your case, you may see a \$ symbol. This is called the **Unix command prompt**. Note that the command prompt might not look the same on every Unix system. We do not include the command prompt in our examples so that you can copy-paste code. In general you use the command prompt to be able to tell that the computer is waiting on commands to be entered.
3. The output of the `ls` command lists two types of objects: files and directories. We'll learn how to tell them apart later on.
4. After the `ls` command finishes it produces a new command prompt, ready for you to type your next command.

The `ls` command is used to list the contents of *any* directory, not necessarily the one that you are currently in. Try the following:

```
ls /bin
```

it will print a lot of program names, among them,

```
bash pwd mv
```

14.6 3: The Unix tree

Looking at directories from within a Unix terminal can often seem confusing. But bear in mind that these directories are exactly the same type of folders that you can see if you use any graphical file browser. From the *root* level (/) there are usually a dozen or so directories. You can treat the root directory like any other, e.g. you can list its contents:

```
ls /
```

prints:

```
bin dev initrd.img    lib64      mnt  root software tmp vmlinuz
boot etc initrd.img.old lost+found opt  run  srv    usr vmlinuz.old
data home lib          media      proc  sbin sys      var
```

You might notice some of these names appear in different colors. Many Unix systems will display files and directories differently by default. Other colors may be used for special types of files. When you log in to a computer, you are typically placed in your home directory, which is often inside a directory called 'Users' or 'home'.

14.7 4: Finding out where you are

There may be many hundreds of directories on any Unix machine, so how do you know which one you are in? The command `pwd`⁵ will print the working directory. That's pretty much all this command does:

```
pwd  
prints:  
/Users/ialbert
```

When you log in to a Unix computer, you are typically placed into your *home* directory. In this example, after we log in, we are placed in a directory called ‘ialbert’ which itself is a *subdirectory* of another directory called ‘Users’. Conversely, ‘Users’ is the *parent* directory of ‘ialbert’. The first forward slash that appears in a list of directory names always refers to the top level directory of the file system (known as the *root directory*). The remaining forward slash (between ‘Users’ and ‘ialbert’) delimits the various parts of the directory hierarchy. If you ever get ‘lost’ in Unix, remember the `pwd` command.

As you learn Unix you will frequently type commands that don’t seem to work. Most of the time this will be because you are in the wrong directory, so it’s a really good habit to get used to running the `pwd` command a lot.

14.8 5: Making new directories

If we want to make a new directory (e.g. to store some lecture related files), we can use the `mkdir` command:

```
mkdir edu  
ls
```

shows:

```
edu
```

⁵<http://en.wikipedia.org/wiki/Pwd>

14.9 6: Getting from ‘A’ to ‘B’

We are in the home directory on the computer but we want to work in the new `edu` directory. To change directories in Unix, we use the `cd` command:

```
cd edu  
pwd
```

will print:

```
/Users/ialbert/edu
```

Let’s make two new subdirectories and navigate into them:

```
mkdir lecture  
cd lecture  
pwd
```

prints:

```
/Users/ialbert/edu/lecture
```

then make a data directory:

```
mkdir data  
cd data/  
pwd
```

prints:

```
/Users/ialbert/edu/lecture/data
```

We created the two directories in separate steps, but it is possible to use the `mkdir` command in way to do this all in one step.

Like most Unix commands, `mkdir` supports *command-line options* which let you alter its behavior and functionality. Command-like options are – as the name suggests – optional arguments that are placed after the command name. They often take the form of single letters (following a dash). If we had used the `-p` option of the `mkdir` command we could have done this in one step. E.g.

```
mkdir -p ~/edu/lecture/docs
```

Note the spaces either side of the -p!

14.10 7: The root directory

Let's change directory to the root directory:

```
cd /
cd Users
cd ialbert
```

In this case, we may as well have just changed directory in one go:

```
cd /Users/ialbert
```

The leading / is incredibly important. The following two commands are very different:

```
cd /step1/step2
cd step1/step2/
```

The first command specifies a so called *absolute path*. It says go the root directory (folder) then the **step1** folder then the **step2** folder that opens from the **step1** directory. Importantly there can only be one **/step1/step2** directory on any Unix system.

The second command specifies a so called *relative path*. It says that from the *current location* go to the **step1** directory then the **step2** directory.

There can potentially be many **step1/step2** directories that open from different directories.

Learn and understand the difference between these two commands. Initially it is very easy to miss the leading / yet it is essential and fundamentally important. With a little time and practice your eyes will be trained and you will quickly notice the difference.

14.11 8: Navigating upwards in the Unix filesystem

Frequently, you will find that you want to go ‘upwards’ one level in the directory hierarchy. Two dots .. are used in Unix to refer to the *parent* directory of wherever you are. Every directory has a parent except the root level of the computer. Let’s go into the lecture directory and then navigate up two levels:

```
cd  
cd edu  
pwd  
prints:  
/Users/ialbert/edu  
but now:  
cd ..  
pwd  
prints:  
/Users/ialbert
```

What if you wanted to navigate up *two* levels in the file system in one go? Use two sets of the .. operator, separated by a forward slash:

```
cd ../../..
```

14.12 9: Absolute and relative paths

Using cd .. allows us to change directory *relative* to where we are now. You can also always change to a directory based on its *absolute* location. E.g. if you are working in the ~/edu/lecture directory and you want to change to the ~/edu/tmp directory, then you could do either of the following:

```
# Go down one level. Use a relative path.  
cd ../../tmp  
pwd
```

For me it prints:

```
/Users/ialbert/edu/tmp
```

or:

```
# Use an absolute path.  
cd ~/edu/tmp  
pwd
```

the last command also prints:

```
/Users/ialbert/edu/tmp
```

The ~ is a shorthand notation that gets substituted to my home directory /Users/ialbert/. You can think of it as a mixture of relative and absolute paths. It is a full path, but it expands to different locations for different people. If I were to specify the path as:

```
# This is an "even more" absolute path.  
cd /Users/ialbert/edu/tmp  
pwd
```

Then the example would only work on my computer (or other computers where the home directory is also named ialbert). Using the ~ allows us to get the best of both worlds. It is an absolute path with a little bit of *relativity* in it.

All examples achieve the same thing, but the later examples require that you know about more about the location of your directory of interest. It is important that you understand the slight differences here, as the absolute and relative paths are important concepts that you will have make use of frequently.

Sometimes it is quicker to change directories using the relative path, and other times it will be quicker to use the absolute path.

14.13 10: Finding your way back home

Remember that the command prompt shows you the name of the directory that you are currently in, and that when you are in your home directory it shows you a tilde character (~) instead? This is because Unix uses the tilde character as a short-hand way of [specifying a home directory][home directory].

See what happens when you try the following commands (use the `pwd` command after each one to confirm the results):

```
cd /
cd ~
cd
```

Hopefully, you should find that `cd` and `cd ~` do the same thing, i.e. they take you back to your home directory (from wherever you were). You will frequently want to jump straight back to your home directory, and typing `cd` is a very quick way to get there.

You can also use the `~` as a quick way of navigating into subdirectories of your home directory when your current directory is somewhere else. I.e. the quickest way of navigating from anywhere on the filesystem to your `edu/lecture` directory is as follows:

```
cd ~/edu/lecture
```

14.14 11: Making the `ls` command more useful

The `..` operator that we saw earlier can also be used with the `ls` command, e.g. you can list directories that are ‘above’ you:

```
cd ~/edu/lecture/
ls ../../
```

Time to learn another useful command-line option. If you add the letter ‘l’ to the `ls` command it will give you a longer output compared to the default:

```
ls -l ~
```

prints (system-dependent):

```
drwx----- 5 ialbert staff 170B May 13 15:24 Applications
drwx-----+ 7 ialbert staff 238B Aug 20 05:51 Desktop
```

```
drwx-----+ 30 ialbert staff 1.0K Aug 12 13:12 Documents  
drwx-----+ 14 ialbert staff 476B Aug 24 10:43 Downloads
```

For each file or directory we now see more information (including file ownership and modification times). The ‘d’ at the start of each line indicates that these are directories. There are many, many different options for the `ls` command. Try out the following (against any directory of your choice) to see how the output changes.

```
ls -l  
ls -R  
ls -l -t -r  
ls -lh
```

Note that the last example combines multiple options but only uses one dash. This is a very common way of specifying multiple command-line options. You may be wondering what some of these options are doing. It is time to learn about Unix documentation....

14.15 12: Man pages

If every Unix command has so many options, you might be wondering how you find out what they are and what they do. Well, thankfully every Unix command has an associated ‘manual’ that you can access by using the `man` command. E.g.

```
man ls  
man cd  
man man # Yes, even the man command has a manual page
```

When you are using the `man` command, press `space` to scroll down a page, `b` to go back a page, or `q` to quit. You can also use the up and down arrows to scroll a line at a time. The `man` command is actually using another Unix program, a text viewer called `less`, which we’ll come to later on.

14.16 13: Removing directories

We now have a few (empty) directories that we should remove. To do this use the [rmdir] command. This will only remove empty directories, so it is quite safe to use. If you want to know more about this command (or any Unix command), then remember that you can just look at its man page.

```
cd ~/edu/lecture/
rmdir data
cd ..
rmdir lecture
ls
```

Note, you have to be outside a directory before you can remove it with `rmdir`

14.17 14: Using tab completion

Saving keystrokes may not seem important now, but the longer that you spend typing in a terminal window, the happier you will be if you can reduce the time you spend at the keyboard. Especially as prolonged typing is not good for your body. So the best Unix tip to learn early on is that you can [tab complete] the names of files and programs on most Unix systems. Type enough letters to uniquely identify the name of a file, directory, or program and press tab – Unix will do the rest. E.g. if you type ‘tou’ and then press tab, Unix should autocomplete the word to ‘touch’ (this is a command which we will learn more about in a minute). In this case, tab completion will occur because there are no other Unix commands that start with ‘tou’. If pressing tab doesn’t do anything, then you have not have typed enough unique characters. In this case pressing tab *twice* will show you all possible completions. This trick can save you a LOT of typing!

Navigate to your home directory, and then use the `cd` command to change to the `edu` directory. Use tab completion to complete directory name. If there are no other directories starting with ‘e’ in your home directory, then you should only need to type ‘`cd`’ + ‘`e`’ + ‘`tab`’.

Tab completion will make your life easier and make you more productive!

Another great time-saver is that Unix stores a list of all the commands that you have typed in each login session. You can access this list by using the **history** command or more simply by using the up and down arrows to access anything from your history. So if you type a long command but make a mistake, press the up arrow and then you can use the left and right arrows to move the cursor in order to make a change.

14.18 15: Creating empty files with the touch command

The following sections will deal with Unix commands that help us to work with files, i.e. copy files to/from places, move files, rename files, remove files, and most importantly, look at files. First, we need to have some files to play with. The Unix command **touch** will let us create a new, empty file. The **touch** command does other things too, but for now we just want a couple of files to work with.

```
cd edu
touch heaven.txt
touch earth.txt
ls
```

prints:

```
earth.txt  heaven.txt
```

14.19 16: Moving files

Now, let's assume that we want to move these files to a new directory ('temp'). We will do this using the Unix [mv] (move) command. Remember to use tab completion:

```
mkdir temp
mv heaven.txt temp
mv earth.txt temp
ls temp
```

For the `mv` command, we always have to specify a source file (or directory) that we want to move, and then specify a target location. If we had wanted to, we could have moved both files in one go by typing any of the following commands:

```
mv *.txt temp
mv *t temp
mv *ea* temp
```

The asterisk `*` acts as a *wild-card character*, essentially meaning ‘match anything’. The second example works because there are no other files or directories in the directory that end with the letter ‘`t`’ (if there were, then they would be moved too). Likewise, the third example works because only those two files contain the letters ‘`ea`’ in their names. Using wild-card characters can save you a lot of typing.

The `?` character is also a wild-card but with a slightly different meaning. See if you can work out what it does.

14.20 17: Renaming files

In the earlier example, the destination for the `mv` command was a directory name (`temp`). So we moved a file from its source location to a target location, but note that the target could have also been a (different) file name, rather than a directory. E.g. let’s make a new file and move it whilst renaming it at the same time:

```
touch rags
ls
```

```
mv rags temp/riches  
ls temp/
```

prints:

```
earth.txt  heaven.txt  riches
```

In this example we create a new file ('rags') and move it to a new location and in the process change the name (to 'riches'). So `mv` can rename a file as well as move it. The logical extension of this is using `mv` to rename a file without moving it. You may also have access to a tool called `rename`, type `man rename` for more information.

```
mv temp/riches temp/rags
```

14.21 18: Moving directories

It is important to understand that as long as you have specified a 'source' and a 'target' location when you are moving a file, then it doesn't matter what your *current* directory is. You can move or copy things within the same directory or between different directories regardless of whether you are in any of those directories. Moving directories is just like moving files:

```
mv temp temp2  
ls temp2
```

14.22 19: Removing files

You've seen how to remove a directory with the `rmdir` command, but `rmdir` won't remove directories if they contain any files. So how can we remove the files we have created (inside `temp`)? In order to do this, we will have to use the `rm` (remove) command.

Please read the next section **VERY** carefully. Misuse of the **rm** command can lead to needless death & destruction

Potentially, **rm** is a very dangerous command; if you delete something with **rm**, you will not get it back! It is possible to delete everything in your home directory (all directories and subdirectories) with **rm**. That is why it is such a dangerous command.

Let me repeat that last part again. It is possible to delete **EVERY** file you have ever created with the **rm** command. Are you scared yet? You should be. Luckily there is a way of making **rm** a little bit safer. We can use it with the **-i** command-line option which will ask for confirmation before deleting anything (remember to use tab-completion):

```
cd temp  
ls  
rm -i earth.txt heaven.txt rags
```

will ask permission for each step:

```
rm: remove regular empty file 'earth.txt'? y  
rm: remove regular empty file 'heaven.txt'? y  
rm: remove regular empty file 'rags'? y
```

We could have simplified this step by using a wild-card (e.g. **rm -i *.txt**) or we could have made things more complex by removing each file with a separate **rm** command.

14.23 20: Copying files

Copying files with the **cp** (copy) command has a similar syntax as **mv**, but the file will remain at the source and be copied to the target location. Remember to always specify a source and a target location. Let's create a new file and make a copy of it:

```
touch file1  
cp file1 file2  
ls
```

What if we wanted to copy files from a different directory to our current directory? Let's put a file in our home directory (specified by `~`, remember) and copy it to the lecture directory (`~/edu/lecture`):

```
$ touch ~/edu/file3  
$ cp ~/edu/file3 ~/edu/lecture/
```

In Unix, the current directory can be represented by a `.` (dot) character. You will often use for copying files to the directory that you are in. Compare the following:

```
ls  
ls .  
ls ./
```

In this case, using the dot is somewhat pointless because `ls` will already list the contents of the current directory by default. Also note how the trailing slash is optional.

14.24 21: Copying directories

The `cp` command also allows us (with the use of a command-line option) to copy entire directories. Use `man cp` to see how the `-R` or `-r` options let you copy a directory *recursively*.

14.25 22: Viewing files with less (or more)

So far we have covered listing the contents of directories and moving/copying/deleting either files or directories. Now we will quickly cover how you can look at files. The `more` or `less` commands let you view (but not edit) text files. We will use the `echo` command to put some text in a file and then view it:

```
echo "Call me Ishmael."  
Call me Ishmael.
```

```
echo "Call me Ishmael." > opening_lines.txt  
ls
```

prints:

```
opening_lines.txt
```

we can view the content of the file with:

```
more opening_lines.txt
```

On its own, `echo` isn't a very exciting Unix command. It just echoes text back to the screen. But we can redirect that text into an output file by using the `>` symbol. This allows for something called file *redirection*.

Careful when using file redirection (`>`), it will overwrite any existing file of the same name

When you are using `more` or `less`, you can bring up a page of help commands by pressing `h`, scroll forward a page by pressing `space`, or go forward or backwards one line at a time by pressing `j` or `k`. To exit `more` or `less`, press `q` (for quit). The `more` and `less` programs also do about a million other useful things (including text searching).

14.26 23: Viewing files with `cat`

Let's add another line to the file:

```
echo "The primroses were over." >> opening_lines.txt  
cat opening_lines.txt
```

prints:

```
Call me Ishmael.  
The primroses were over.
```

Notice that we use `>>` and not just `>`. This operator will **append** to a file. If we only used `>`, we would end up overwriting the file. The `cat` command displays the contents of the file (or files) and then returns you to the command line. Unlike `less` you have no control on how you view that text (or what you do with it). It is a very simple, but sometimes useful, command. You

can use `cat` to quickly combine multiple files or, if you wanted to, make a copy of an existing file:

```
cat opening_lines.txt > file_copy.txt
```

14.27 24: Counting characters in a file

```
$ ls  
opening_lines.txt  
  
$ ls -l  
total 4  
-rw-rw-r-- 1 ubuntu ubuntu 42 Jun 15 04:13 opening_lines.txt  
  
$ wc opening_lines.txt  
2 7 42 opening_lines.txt  
  
$ wc -l opening_lines.txt  
2 opening_lines.txt
```

The `ls -l` option shows us a long listing, which includes the size of the file in bytes (in this case ‘42’). Another way of finding this out is by using Unix’s `wc` command (word count). By default this tells you many lines, words, and characters are in a specified file (or files), but you can use command-line options to give you just one of those statistics (in this case we count lines with `wc -l`).

14.28 25: Editing small text files with nano

Nano is a lightweight editor installed on most Unix systems. There are many more powerful editors (such as ‘emacs’ and ‘vi’), but these have steep learning curves. Nano is very simple. You can edit (or create) files by typing:

```
nano opening_lines.txt
```

You should see the following appear in your terminal:

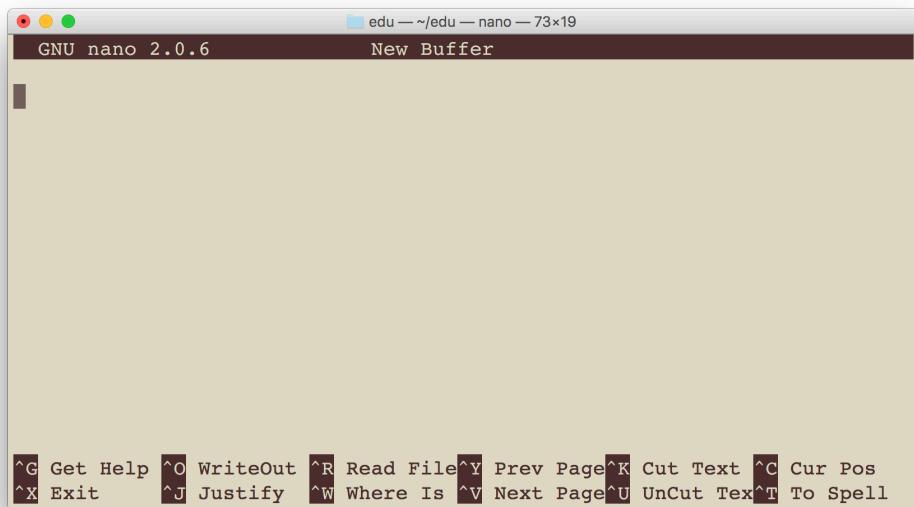


Figure 14.3

The bottom of the nano window shows you a list of simple commands which are all accessible by typing ‘Control’ plus a letter. E.g. Control + X exits the program.

14.29 26: The \$PATH environment variable

One other use of the `echo` command is for displaying the contents of something known as *environment variables*. These contain user-specific or system-wide values that either reflect simple pieces of information (your username), or lists of useful locations on the file system. Some examples:

```
$ echo $USER  
ialbert  
$ echo $HOME  
/Users/ialbert  
echo $PATH  
/usr/local/sbin:/usr/local/bin:/Users/ialbert/bin
```

The last one shows the content of the \$PATH environment variable, which displays a — colon separated — list of directories that are expected to contain programs that you can run. This includes all of the Unix commands that you have seen so far. These are files that live in directories which are run like programs (e.g. ls is just a special type of file in the /bin directory).

Knowing how to change your \$PATH to include custom directories can be necessary sometimes (e.g. if you install some new bioinformatics software in a non-standard location).

14.30 27: Matching lines in files with grep

Use nano to add the following lines to opening_lines.txt:

```
Now is the winter of our discontent.  
All children, except one, grow up.  
The Galactic Empire was dying.  
In a hole in the ground there lived a hobbit.  
It was a pleasure to burn.  
It was a bright, cold day in April, and the clocks were striking thirteen.  
It was love at first sight.  
I am an invisible man.  
It was the day my grandmother exploded.  
When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow.  
Marley was dead, to begin with.
```

You will often want to search files to find lines that match a certain pattern. The Unix command grep does this (and much more). The following examples show how you can use grep's command-line options to:

- show lines that match a specified pattern

- ignore case when matching (`-i`)
- only match whole words (`-w`)
- show lines that don't match a pattern (`-v`)
- Use wildcard characters and other patterns to allow for alternatives (*, ., and [])

Show lines that match the word `was`:

```
$ grep was opening_lines.txt
The Galactic Empire was dying.
It was a pleasure to burn.
It was a bright, cold day in April, and the clocks were striking thirteen.
It was love at first sight.
It was the day my grandmother exploded.
When he was nearly thirteen, my brother Jem got his arm badly broken at the el
Marley was dead, to begin with.
```

Use:

```
grep --color=AUTO was opening_lines.txt
```

to highlight the match.

Show lines that do not match the word `was` :

```
$ grep -v was opening_lines.txt
Now is the winter of our discontent.
All children, except one, grow up.
In a hole in the ground there lived a hobbit.
I am an invisible man.
```

`grep` has a great many options and applications.

14.31 28: Combining Unix commands with pipes

One of the most powerful features of Unix is that you can send the output from one command or program to any other command (as long as the second command accepts input of some sort). We do this by using what is known as a **pipe**. This is implemented using the ‘|’ character (which is a character which always seems to be on different keys depending on the keyboard that you are using). Think of the pipe as simply connecting two Unix programs. Here’s an example which introduces some new Unix commands:

```
grep was opening_lines.txt | wc -c  
# 316  
  
grep was opening_lines.txt | sort | head -n 3 | wc -c  
# 130
```

The first use of **grep** searches the specified file for lines matching ‘was’. It sends the lines that match through a pipe to the **wc** program. We use the **-c** option to just count characters in the matching lines (316).

The second example first sends the output of **grep** to the Unix **sort** command. This sorts a file alphanumerically by default. The sorted output is sent to the **head** command which by default shows the first 10 lines of a file. We use the **-n** option of this command to only show 3 lines. These 3 lines are then sent to the **wc** command as before.

Whenever making a long pipe, test each step as you build it!

14.32 Miscellaneous Unix power commands

The following examples introduce some other Unix commands, and show how they could be used to work on a fictional file called **file.txt**. Remember, you can always learn more about these Unix commands from their respective

man pages with the `man` command. These are not all real world cases, but rather show the diversity of Unix command-line tools:

- View the penultimate (second-to-last) 10 lines of a file (by piping `head` and `tail` commands):

```
#tail -n 20 gives the last 20 lines of the file, and piping that to head will give the penultimate 10 lines
tail -n 20 file.txt | head
```

- Show the lines of a file that begin with a start codon (ATG) (the `^` matches patterns at the start of a line):

```
grep '^ATG' file.txt
```

- Cut out the 3rd column of a tab-delimited text file and sort it to only show unique lines (i.e. remove duplicates):

```
cut -f 3 file.txt | sort -u
```

- Count how many lines in a file contain the words ‘cat’ or ‘bat’ (`-c` option of `grep` counts lines):

```
grep -c '[bc]at' file.txt
```

- Turn lower-case text into upper-case (using `tr` command to ‘transliterate’):

```
cat file.txt | tr 'a-z' 'A-Z'
```

- Change all occurrences of ‘Chr1’ to ‘Chromosome 1’ and write changed output to a new file (using `sed` command):

```
cat file.txt | sed 's/Chr1/Chromosome 1/' > file2.txt
```

Chapter 15

Data analysis with Unix

In this section, we will use simple Unix tools to obtain and investigate a genome data file that represents all the annotated features of the yeast genome.

This section is not meant to be a full introduction to all the utility of the Unix command line.

What we attempt to show you are the concepts, methods and the manner in which Unix tools are typically employed to probe and poke at data.

15.1 What directory should I use?

When you open a terminal, it starts in a specific location (directory, folder) on your computer. You can, of course, move around and navigate to other places, but you have to remember that every action that you take is always relative to the location from which you are giving the command.

For each project, lecture or exploration create a separate folder that you use only for that task. Starting with an empty directory will help you keep track of the files you generate at each step.

Never start typing commands from your home directory. Tools create several ancillary files, and you do not want these mixed in with your other work-related data.

Your two BFFs (best friends forever) are `pwd` and `ls`. These two commands will tell you the most important information:

1. `pwd` will tell you *where* you are.
2. `ls` will tell you *what* you have there.

Especially at the beginning a lot of confusion is caused by not knowing these two pieces of information. When in doubt ask yourself:

1. Where am I on my computer?
2. What do I have in the current folder?

15.2 Where are we getting the data from?

We'll visit the Saccharomyces Genome Database (SGD)¹ webpage. What you will notice that while clicking around "makes sense" it is not easy to properly describe where you would get a given dataset. All too often you will hear people say: "*I've downloaded data from the XYZ database*" as if there was only one data to be downloaded. Often you are left scratching your head as to what exactly did they download.

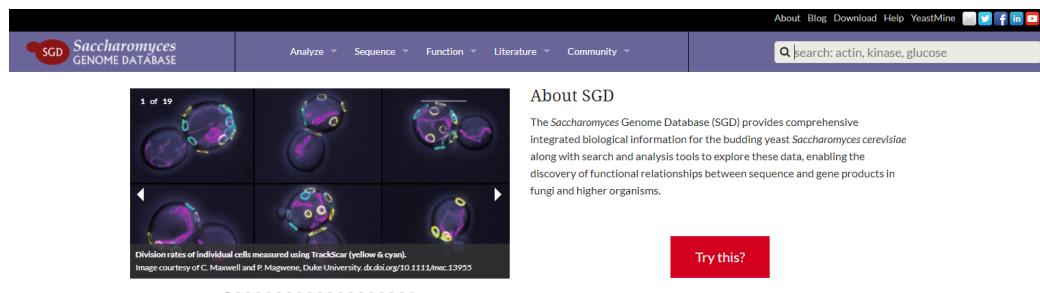


Figure 15.1

Note: Websites may change unexpectedly and without notice. When a site changes the links may break. To avoid the problems caused by changes outside of our control, we host some of the data on the handbook website.

¹<http://www.yeastgenome.org/>

Visit the “Download”² link then “Curated Data”³ then “**Chromosomal Features**”. From that section we want to investigate the file called `SGD_features.tab` that is described as “*Chromosomal features, coordinates, gene descriptions.*”. Let’s get it into our terminal.

Chromosomal Features

Information about *S. cerevisiae* chromosomal features annotated in SGD.

Name	Description	README
SGD_CDS_xref.txt	SGD cross references for CDS entries in GenBank/EMBL/DDBJ	
SGD_features.tab	Chromosomal features, coordinates, gene descriptions.	

Figure 15.2

The first thing you need to do is obtain the URL of this file. While hovering over the link right click and select “Copy link address.” You now have the location of the data:

`https://downloads.yeastgenome.org/curation/chromosomal_feature/
SGD_features.tab`

15.3 How do I obtain a data file that is online?

We can use either the `curl` or `wget` command line tool (see `Curl vs wget`⁴) to download the content of URLs.

You may use the link above, but for simplicity and to avoid potential errors outside of our control, we also host these files ourselves; thus you can get them from the handbook data site.

- `http://data.biostarhandbook.com/data/SGD_features.tab`
- `http://data.biostarhandbook.com/data/SGD_features README`

²<https://www.yeastgenome.org/download-data>

³<https://www.yeastgenome.org/download-data/curation>

⁴<https://daniel.haxx.se/docs/curl-vs-wget.html>

```

# Create a work directory for a unix tutorial work.
mkdir unixtut

# Switch to the new directory you will work in.
cd unixtut

# Get the data from SGD.
wget http://data.biostarhandbook.com/data/SGD_features.tab

# Also get the README file.
wget http://data.biostarhandbook.com/data/SGD_features README

# Quick check. Where are you and what have you got?
pwd
ls

```

Simple Unix tools let us answer a wide variety of questions.

15.3.1 What are flags (parameters)?

A “flag” in Unix terminology is a parameter added to the command. See for example

`ls`

versus

`ls -l`

Typically flags make programs behave differently or report their information in different ways.

15.3.2 How to find out what flags are available?

You can use the manual to learn more about a command:

`man ls`

will produce

NAME

ls -- list directory contents

SYNOPSIS

ls [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz] [file ...]

DESCRIPTION

For each operand that names a file of a type other than directory, **ls** displays its name as well as any requested, associated information. For each operand that names a file of type directory, **ls** displays the names of files contained within that directory, as well as any requested, associated information.

...

-l (The lowercase letter ``ell'').) List in long format. (See below.) If the output is to a terminal, a total sum for all the file sizes is output on a line before the long listing.

...

We can see above that **ls** takes the **-l** parameter (letter ell, not to be confused with the number 1 as the two can look the same depending on the fonts used by your screen). The **-l** parameter makes **ls** report the long form.

You don't need to remember all flags, not even the majority of them. Most of the time we use just a few, in some cases when you find yourself needing some specific use of a tool you may want to investigate the manual, perhaps the tool already supports the behavior that you seek.

15.3.3 What if the tools does not have manual page?

Only tools that come with Unix will have a manual. For other software the **-h**, **--help** or **--help** command will typically print instructions for their use.

hisat2 --help

If the help flag is not recognized you will need to find the manual for the software.

15.3.4 What are flag formats?

Traditionally Unix tools use two flag forms:

- short form: single minus - then one letter, like `-g`, '`-v`'
- long form: double minus -- then a word, like `--genome`, `--verbosity`

In each case the parameter may stand alone as a toggle (on/off) or may take additional values after the flag. `-p` or `-p 100`

Now some bioinformatics tools do not follow this tradition and use a single - character for both short and long options. `-g` and `--genome`.

15.3.5 Is using flags important?

Using flags is a essential to using Unix. Bioinformatics tools typically take a large number of parameters specified via flags. The correctness of the results critically depends on the proper use of these parameters and flags.

15.3.6 How do I view the data one page at a time?

```
more SGD_features README
```

press "space" to move forward, "b" to move backward, q or ESC to exit.

15.3.7 How do I open a stream from a data?

The `cat` command concatenates one or more files and stars producing them on the output stream:

```
cat SGD_features.tab
```

The command above will print the contents of the file to the current output stream which happens to be your terminal window.

Tips:

- Use the manual `man` for more information on commands.
- If you want to rerun a command you can use the up-arrow key to recall the previous entry.

- You can press TAB to ask your shell to try to complete a file name. It is convenient! Always try to have the shell fill in the filename for you, that way you will never mistype these.

15.3.8 How many lines does the file have?

You can pipe the output of your stream into another program rather than the screen. Use the | character to connect the programs. For example, the `wc` program is the word counter.

```
cat SGD_features.tab | wc
```

prints the number of lines, words, and characters in the stream:

```
16454 425719 3264490
```

You can customize what the `wc` program does:

```
cat SGD_features.tab | wc -l
```

so that it will only print the number of lines:

```
16454
```

Now many programs also operate on files directly like so:

```
wc -l SGD_features.tab
```

The command above is fully equivalent to:

```
cat SGD_features.tab | wc -l
```

In general, it is a better option to open a stream with `cat` then pipe the flow into the next program. Later you will see that it is easier to design, build and understand more complex pipelines when the data stream is opened at the beginning as a separate step.

Some people have strong negative feelings about this second form. They, feel that it is redundant, it even has a name: Useless Use of Cat⁵. For them, it feels superfluous, and indeed it is ever so slightly less efficient as it opens two processes instead of just one.

⁵<http://porkmail.org/era/unix/award.html>

The main problem with `wc -l SGD_features.tab` is that it makes the process less reusable if you wanted to run another tool with the same file you'd modify the beginning of the command, whereas when using `cat` you can delete the last command and add a new one.

15.3.9 How does the file start?

```
cat SGD_features.tab | head
```

prints:

S000002143	ORF	Dubious YAL069W	chromosome 1	1	335
S000031098	CDS		YAL069W	1	335
S000028594	ORF	Dubious YAL068W-A	chromosome 1	1	1
S000031372	CDS		YAL068W-A	1	538
					792

15.3.10 Which lines match a specific pattern?

Suppose you wanted to find information on gene YAL060W

```
cat SGD_features.tab | grep YAL060W
```

15.3.11 How can I color the pattern that is matched?

```
cat SGD_features.tab | grep YAL060W --color=always | head
```

15.3.12 How can we tell how many lines DO NOT match a given pattern?

Adding the `-v` flag to `grep` reverses the action of `grep` it shows the lines that do not match.

```
cat SGD_features.tab | grep -v Dubious | wc -l
```

15.3.13 How do I store the results in a new file?

The > character is the redirection.

```
cat SGD_features.tab | grep YAL060W > match.tab
```

now check how many files do you have:

```
match.tab  
SGD_features README  
SGD_features.tab
```

15.3.14 How many lines in the file match the word gene?

```
cat SGD_features.tab | grep gene | wc -l
```

Important: note how the above matches the word gene anywhere on the line.

15.3.15 How do I select genes?

It looks like this file uses the feature type (column 2) ORF for protein-coding genes. You will need to cut the second field (by tabs).

```
cat SGD_features.tab | cut -f 2 | head
```

Build your commands one step at a time, always checking that you are on the right track:

```
cat SGD_features.tab | head  
cat SGD_features.tab | cut -f 2 | head  
cat SGD_features.tab | cut -f 2 | grep ORF | head
```

15.3.16 How many genes are there?

```
cat SGD_features.tab | cut -f 2 | grep ORF | wc -l
```

15.3.17 Can I select multiple columns?

```
cat SGD_features.tab | cut -f 2,3,4 | grep ORF | head
```

What does this do?

```
cat SGD_features.tab | cut -f 2,3,4 | grep ORF | grep -v Dubious | wc -l
```

15.4 How many feature types are in this data?

We are going to use this data a lot, so place it into a separate file for now.

```
cat SGD_features.tab | cut -f 2 > types.txt
```

Sorting places identical consecutive entries next to one another.

```
cat types.txt | sort | head
```

Find unique words. The `uniq` command collapses consecutive identical words into one.

```
cat types.txt | sort | uniq | head
```

Using `-c` flag to `uniq` will not only collapse consecutive entries it will print their counts.

```
cat types.txt | sort | uniq -c | head
```

prints:

```

352 ARS
196 ARS_consensus_sequence
  6 blocked_reading_frame
7074 CDS
  16 centromere
  16 centromere_DNA_Element_I
  16 centromere_DNA_Element_II
  16 centromere_DNA_Element_III
  8 external_transcribed_spacer_region
24 five_prime_UTR_intron

```

To find out the most frequent unique elements, we need to sort the output of uniq.

```
cat types.txt | sort | uniq -c | sort -rn | head
```

it now prints:

```

7074 CDS
6604 ORF
484 noncoding_exon
383 long_terminal_repeat
377 intron
352 ARS
299 tRNA_gene
196 ARS_consensus_sequence
  91 transposable_element_gene
  77 snoRNA_gene

```

15.5 The single most useful Unix pattern

The pattern `sort | uniq -c | sort -rn` is perhaps the most useful simple unix command pattern that you will ever learn.

Time and again we are surprised by just how many applications it has, and how frequently problems can be solved by sorting, collapsing identical values, then resorting by the collapsed counts.

The skill of using Unix is not just that of understanding the commands themselves. It is more about recognizing when a pattern, such as the one

that we show above, is the solution to the problem that you wish to solve. The easiest way to learn to apply these patterns is by looking at how others solve problems, then adapting it to your needs.

15.6 One-liners

We could have also done the last step in one single step to find out what are the most frequent unique types of features in this file:

```
cat SGD_features.tab | cut -f 2 | sort | uniq -c | sort -rn | head
```

The commands above are what we will call a pipeline, or a segment of a pipeline.

When you see a command like the one above, rebuild it yourself, one step at a time from the left to right:

```
cat SGD_features.tab | head  
cat SGD_features.tab | cut -f 2 | head  
cat SGD_features.tab | cut -f 2 | sort | head  
cat SGD_features.tab | cut -f 2 | sort | uniq -c | head  
cat SGD_features.tab | cut -f 2 | sort | uniq -c | sort -rn | head
```

When you understand each step, you have mastered the pattern.

Chapter 16

Using Makefiles

Whereas writing scripts is useful and helps immensely, there may soon come a time when you want to keep all related functionality in one file, yet only run a sub-section of it.

The tool named `make` and its corresponding `Makefile` are designed to do that (and a lot more as well). `make` will allow you to set up a veritable *command and control center* for your data analysis from where you will be able to keep track of what you have done and how.

In our opinion the path to reproducible research starts with `Makefiles`. Perhaps at some point in the future when scientists submit their data analysis results, they will also be required to also attach their `Makefiles` that describe the steps they took in a clear and explicit way.

16.1 What is a Makefile?

You can think of makefiles as scripts with “targets”.

A `Makefile` is a simple text file that lists commands grouped by so called “targets”, which you can think of as labels. We indicate the commands that belong to a rule by indenting them via a TAB character:

```
foo:  
    echo Hello John!
```

```
bar:  
    echo Hello Jane!  
    echo Hello Everyone!
```

This `Makefile` has two targets: `foo` and `bar`. And once you create this file you may type:

```
make foo  
and it will print  
Hello John!
```

Note how `make` automatically executed the `Makefile` as long as it was called just that. If we wanted to execute a different makefile we would have to specify it like so `make -f othermakefile`. If we were to type:

```
make bar  
it will print:  
Hello Jane!  
Hello Everyone!
```

And that's it. This one feature of Makefile is sufficient to greatly simplify your life and package your entire workflow into a single file.

For every analysis you should create a single `Makefile` and perhaps other scripts with it. Then you basically have a “central command script” from which you can orchestrate your analysis.

```
make fastqc  
make align  
make plots  
...
```

16.2 Why doesn't my Makefile work?

One common error to check for is neglecting to put tabs in front of the commands. Many text editors insist on inserting a number of space characters even if you press TAB and not SPACE on your keyboard. View the whitespace in your editor.

16.3 Can I use bash shell constructs in a Makefile?

The syntax used inside makefiles may superficially look like a bash shell, but only the commands themselves are passed and executed as a bash shell. There is a fine distinction there - one that you may not ever need to bridge - if you do, remember that a **Makefile** is not a **bash** script. For example, variables in Makefiles are specified the same way as in bash but NOT executed in bash:

```
NAME=Jane
hello:
    echo Hello ${NAME}
```

16.4 Why does my Makefile print every command?

By default **make** echoes every command. This helps you see what it is trying to do. To turn that off add a @ symbol to the line:

```
NAME=Jane
hello:
    @echo Hello ${NAME}
```

16.5 Why does my Makefile stop on an error?

Remember, that is a good thing. You really want to stop on any error so you can examine and fix it.

In the odd case when you really don't care whether a command succeeded or not and you don't want the make to stop add the - sign in front of the command:

```
hello:
    -cp this that
    echo "Done"
```

Will always print “Done” even if the file `this` is not present.

16.6 Why is the first action executed?

If you run `make` with no task label it will execute the first label it sees. It is a good idea to place usage information there so that you can remind yourself of it just by running `make`.

```
NAME=Jane
usage:
    @echo "Usage: make hello, goodbye, ciao"

hello:
    @echo Hello ${NAME}
```

16.7 Is there more to Makefiles?

We only covered the basic features of `make`, the ones that we believe will give you the most benefits with the least amount of overhead.

Rest assured that `make` has several other very handy features. One important feature is that it can track the so-called dependencies between files. It can be told to automatically detect which sections need to be re-run when some of the inputs change.

In our personal observation these features of `make` are only useful when data analysis is performed at very large scale. In our typical work we almost never need to specify these dependencies. That keeps the use of `make` a lot simpler. But we encourage you to explore and learn more about `make`. A good start would be the document titled Automation and Make¹ on Software Carpentry².

But remember - the simplest feature that you learned above already saves you a lot of effort.

¹<http://swcarpentry.github.io/make-novice/>

²<http://swcarpentry.github.io/make-novice/>

16.8 Are there are alternatives to Makefiles?

The concept of `make` goes back a long way and in time many alternatives have been proposed. One that seems to have taken off with bioinformaticians is `snakemake`³ If you find yourself having to develop overly complex Makefiles it might be worth exploring the alternatives. On our end we get a lot done with simple and uncomplicated Makefiles.

³<https://bitbucket.org/snakeyed/snakeyed/wiki/Home>

Chapter 17

Data compression

This chapter deals with compressed files that may be referred to as zip files, gzip files, tar files, compressed files, archives.

17.1 Quick summary

This at the beginning is just a reminder of how the commands work if you forget them. We recommend that you consult the full chapter at least once if you are unfamiliar with the concepts.

17.1.1 How do I uncompress a `tar.gz` archive?

You extract, a zipped, file with verbose output that shows you what is going on:

```
tar xzfv archive.tar.gz
```

unpacks the file `archive.tar.gz` and you'll have both the original file `archive.tar.gz` and what was inside it. If there were directories inside the archive these will be created.

17.1.2 How do I create a compressed archive?

You create, a zipped, file with verbose output that shows you what is going on:

```
tar czfv archive.tar.gz path/to/file1 path/to/file2
```

produces the file `archive.tar.gz`

17.2 What is a compressed format?

Compressed formats reduce file disk space requirements. A compressed file contains the same information as an uncompressed but has a smaller size. On the downside, it needs to be decompressed to access its content.

Compression (creating a compressed file) requires substantially more computational resources than decompression. It is always much faster (may even be an order of magnitude faster) to restore a file from a compressed form than to create that condensed form.

Some compressed formats (like the `.gz` format) can be concatenated (or appended to) without being decompressed. For example

```
cat file1.gz file2.g file3.gz > bigfile.gz
```

Will create a valid compressed file that contains all three files. When the files are large, the time to compress may be hours (or more) thus concatenating files as done above can quite useful as it avoids the most time-consuming step.

17.3 What objects may be compressed?

We can compress single files or entire directories. Compressed directories are often called “archives”.

- A “compressed file” is a single file reduced in size. The name may end with `.gz`, `.bz`, `.bz2`, `.zip` or `xz`
- A “compressed archive” is a folder containing multiple files combined, and then compressed. The name may end with `zip`, `.tar.gz`, `.tar.bz2`, `tar.xz`.

17.4 What are some common compression formats?

- ZIP, extension `.zip`, program names are `zip/unzip`. Available on most platforms.
- GZIP extension `.gz`, program names are `gzip/gunzip`. The most common compression format on Unix-like systems.
- BZIP2 extension `.bz/.bz2`, program names are `bzip2/bunzip2`.
- XZ extension `.xz`. A more recent invention. Technically `bzip2` and `xz` are improvements over `gzip`. `gzip` is still quite prevalent for historical reasons and because it requires less memory.

In the majority of cases, bioinformatics data is distributed in `gz` and `tar.gz` formats.

17.5 Is there a bioinformatics-specific compression format?

Yes. The `bgzip` (Blocked GNU Zip Format) is a variant of the GZIP format that allows random access to its content. BCF, BAM, and TABIX formats are in BGZIP compressed formats.

A `bgzip` file can be decompressed with `gzip` but only the specialized `bgzip` command can create a file in the Blocked GNU Zip Format file. The `bgzip` utility is installed when you install the `htslib` package.

Most bgzipped files carry the `gz` extension, thus you can't always tell which compression was used. You would need to create `bgzip` files yourself when compressing VCF and TABIX files. For all other cases `gzip` will suffice.

```
# Get a sequence file.  
efetch -db=nuccore -format=fasta -id=AF086833 > AF086833.fa  
# Create bgzipped file.  
bgzip AF086833.fa
```

More about `bgzip`: <http://www.htslib.org/doc/tabix.html>

17.6 How do I compress or uncompress a file?

```
# Get a sequence file.  
efetch -db=nuccore -format=fasta -id=AF086833 > AF086833.fa  
  
# Compress with gzip.  
# Creates the file AF086833.fa.gz.  
gzip AF086833.fa  
  
# Some tools can operate on GZ files without unpacking them.  
# Note: the tool is called gzcat on macOS  
gzcat AF086833.fa.gz | head  
  
# Use zcat on Linux  
# zcat AF086833.fa.gz | head  
  
# Uncompress the file. Creates AF086833.fa.  
gunzip AF086833.fa.gz
```

17.7 How do I compress or uncompress multiple files?

Zip-files may contain one or multiple files. Not knowing beforehand what an archive contains files or directories can be confusing and annoying in that you can't tell from the file name alone whether it is a single file or many files.

In the Unix world the most commonly used approach is to create a so-called “tape archive” even though it does not typically involve tape anymore. The resulting file is called a **tar** file (Tape Archive) file. The extension for gzip-compressed tap archive files is **.tar.gz**.

Suppose you had two files and wanted to create an archive from them:

```
# Get two files
efetch -db=nuccore -format=fasta -id=AF086833 > AF086833.fa
efetch -db=nuccore -format=gb -id=AF086833 > AF086833.gb
```

The archive creation command works by subcommands that labels by the actions that it performs: create will be `c` , a file will be `f` like so:

```
tar cf archive-name files-to-be-compressed
```

Thus if we want to create `c`, a compressed `z`, file `f`, in verbose `v` mode and name the archive `sequences.tar.gz` we would use the following construct:

```
tar czfv sequences.tar.gz AF086833.fa AF086833.gb
```

The first name listed on the command `sequences.tar.gz` is the file to be created, the other two are the files that should be added, equivalent to:

```
tar czvf sequences.tar.gz AF086833.*
```

17.7.0.1 Warning, a super annoying behavior!

If you accidentally put the file that you wanted to compress as the first to `tar`, then it will be destroyed as tar will try to create that file!

```
# SUPER ANNOYING ERROR! Destroys the file you wanted to archive!
tar czvf AF086833.fa sequences.tar.gz
```

Unless we have very few files, adding them into an archive is not a good way to package files – upon unpacking, we may not remember which files were in the archive to begin with.

The recommended practice to create archives is to put all files into a subdirectory then compress that entire directory:

```
mkdir sequences
mv AF086833.* sequences/
```

then compress that entire directory like so:

```
tar czvf sequences.tar.gz sequences/*
```

The added advantage is that `tar` will not overwrite a directory as a file even if you list it in the wrong order, thus the super annoying behavior is not an issue, you'll just get an error that it cannot create the archive.

17.8 What is a tarbomb?

A tarbomb is a tar file, usually created by an unwitting user, that contains a vast number of files.

Here is an example:

```
mkdir -p tarbomb
cd tarbomb
curl -O http://data.biostarhandbook.com/data/tarbomb.tar.gz
tar xzf tarbomb.tar.gz
```

Upon unpacking it “explodes” all over your directory, perhaps instead of a bomb we should call it a “flashbang” grenade. It is quite disorienting once you look at the directory and you can’t recognize what is yours and what was unpacked.

Disarming a tarbomb:

1. First, view the content of an archive:

```
tar tzvf tarbomb.tar.gz
```

1. If it is a tarbomb move it to a separate directory and unpack it there.

Usually it is a good habit to always make a new directory for tar files you don’t know about and unpack them there. If you make that a habit, you’ll never be surprised, it is like defensive driving, always expect the worst.

17.9 How do we use tar again?

You may forget which `tar` flags you need. Come back to this chapter or Google it quickly.

Take solace that it is not just you. See <http://xkcd.com/1168/> for a different kind of tarbomb:

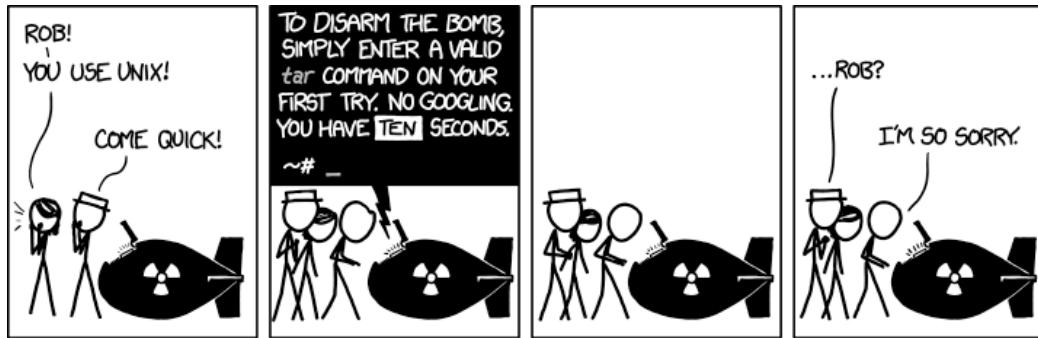


Figure 17.1

Part IV

DATA SOURCES

Chapter 18

What is data?

Intuitively people believe that they have a good practical understanding of what data is, but we found that it can be unexpectedly difficult to articulate even what the word *data* means.

Reflect for a moment, what is data?

We went around and asked a bunch of people the hard question **What is data?.**

Note the most common definition:

DATA is, uhhhh ... DATA

18.1 So what is data?

The simplest definition relies on clarifying that “data” and “data format” are closely related concepts:

1. A symbolic representation of **information**.
2. A **design**, and **optimization** of that information.

The second point is, in my opinion, the essential property of data that most people can't quite articulate.

Depending on the format, information that is present in the data may be readily accessible or difficult to obtain. Some information may be impossible to extract, even though the information is there! Many scientists are unaware that information organization is just as important as the information itself.

18.2 Essential properties of data

- The same information may be represented in different formats (optimized differently).
- The same format (optimization) may be used to describe different types of information.
- There may be information in data that is not readily accessible (optimized).

Often, it can be surprisingly difficult to predict the types of new information that we can derive from a dataset. Pushing the limits of what we can extract from data is an essential part of science.

Understanding data formats, what information is encoded in each, and when it is appropriate to use one format over another is an essential skill of a bioinformatician.

18.3 How life scientists think about bioinformatics

For a typical life scientist bioinformatics means a software transforms “data” into an “answer”.

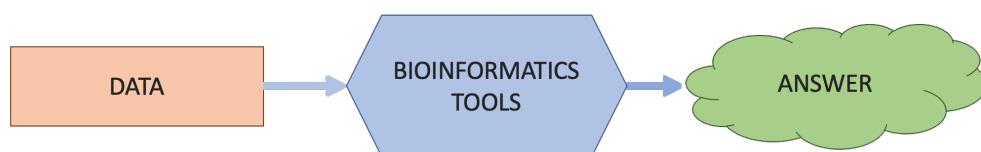


Figure 18.1

We call this “a tool-centric” world view, where the software is a defining component. Running a software is expected to immediately produce the insight and explanation that the scientist needs.

18.4 What bioinformatics is in reality

In reality, as a bioinformatician you need to understand that there is **only data**. Software tools merely transform data from one format to another. The new, resulting data will usually have different “optimization”. It may be richer in that it may contain a combination of the information from other data. But it is still data - and the biological insight comes from understanding and investigating this data.

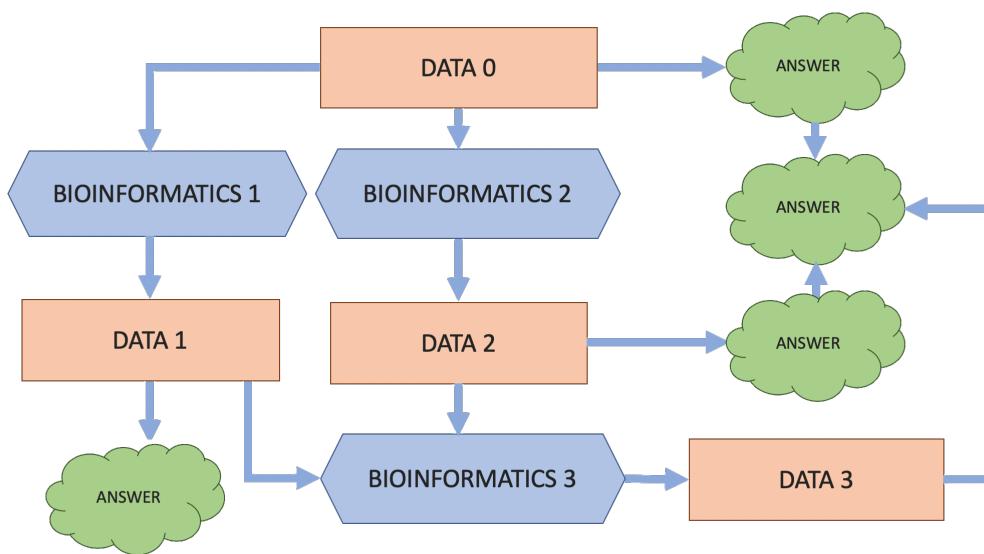


Figure 18.2

18.5 What is the state of data in bioinformatics?

It is kind of a mess, but fear not:

Everything's gonna be alright Everything's gonna be okay It's gonna be a good, good, life That's what my therapists say

Here is what is happening. Scientists that control, influence and shape the field, as well as society as a whole has not yet fully realized the complexity, the challenges and the importance of data within this field of science. The current organizational structure, the way we choose to represent, store and disseminate data are all woefully inadequate. The amount of newly generated information is continually overflowing the “containers” and “storage” that were designated to hold and organize them.

Your skill as a bioinformatician will be measured in part by your ability to cut a path through the increasingly diverging information and subjectivity to identify and isolate the relevant information.

18.6 What kind of problems does bioinformatics data have?

Let us demonstrate the ever growing magnitude of the problem. A little background first.

Have you heard about the Human Genome Project? The one “completed” in the year 2000¹? Perhaps you heard that ten years later² we realized that the job was not quite done, but that the best is yet to come³.

That was about ten years ago. So what is the status now? How well do we understand the Human Genome? Let us reproduce here a blog post made by Heng Li on Oct 13, 2017. It merely tries to answer the question Which human reference genome to use⁴

Which human reference genome to use

TL;DR: If you map reads to GRCh37 or hg19, use hs37-1kg:

¹<https://www.nature.com/articles/35057062>

²<https://www.nature.com/collections/vqfbhrpqcd>

³<https://www.nature.com/articles/470140a>

⁴<http://lh3.github.io/2017/11/13/which-human-reference-genome-to-use>

18.6. WHAT KIND OF PROBLEMS DOES BIOINFORMATICS DATA HAVE?193

ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/technical/reference/human_g1k_v37.fasta.

If you map to GRCh37 and believe decoy sequences help with better variant calling, use `hs37d5`:

ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/technical/reference/phase2_reference_ass

If you map reads to GRCh38 or `hg38`, use the following:

ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_f

There are several other versions of GRCh37/GRCh38. What's wrong with them? Here are a collection of potential issues:

1. Inclusion of ALT contigs. ALT contigs are large variations with very long flanking sequences nearly identical to the primary human assembly. Most read mappers will give mapping quality zero to reads mapped in the flanking sequences. This will reduce the sensitivity of variant calling and many other analyses. You can resolve this issue with an ALT-aware mapper, but no mainstream variant callers or other tools can take the advantage of ALT-aware mapping.
2. Padding ALT contigs with long “N”s. This has the same problem with 1 and also increases the size of genome unnecessarily. It is worse.
3. Inclusion of multi-placed sequences. In both GRCh37 and GRCh38, the pseudo-autosomal regions (PARs) of chrX are also placed on to chrY. If you use a reference genome that contains both copies, you will not be able to call any variants in PARs with a standard pipeline. In GRCh38, some alpha satellites are placed multiple times, too. The right solution is to hard mask PARs on chrY and those extra copies of alpha repeats.
4. Not using the rCRS mitochondrial sequence. rCRS is widely used in population genetics. However, the official GRCh37 comes with a mitochondrial sequence 2bp longer than rCRS. If you want to analyze mitochondrial phylogeny, this 2bp insertion will cause troubles. GRCh38 uses rCRS.
5. Converting semi-ambiguous IUB codes to “N”. This is a very minor issue, though. Human chromosomal sequences contain few semi-ambiguous bases.
6. Using accession numbers instead of chromosome names. Do you know CM000663.2 corresponds to `chr1` in GRCh38?

7. Not including unplaced and unlocalized contigs. This will force reads originated from these contigs to be mapped to the chromosomal assembly and lead to false variant calls.

Now we can explain what is wrong with other versions of human reference genomes:

- hg19/chromFa.tar.gz from UCSC: 1, 3, 4 and 5.
- hg38/hg38.fa.gz from UCSC: 1, 3 and 5.
- GCA_000001405.15_GRCh38_genomic.fna.gz from NCBI: 1, 3, 5 and 6.
- Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz from Ensembl: 3.
- Homo_sapiens.GRCh38.dna.toplevel.fa.gz from EnsEMBL: 1, 2 and 3.

Using an inappropriate human reference genome is usually not a big deal unless you study regions affected by the issues. However,

1. other researchers may be studying in these biologically interesting regions and will need to redo alignment;
2. aggregating data mapped to different versions of the genome will amplify the problems. It is still preferable to choose the right genome version if possible.

Well, welcome to bioinformatics!

(end of blog post)

Reflect on the above a little bit. Almost twenty years after “completing” the human genome project we have a hard time deciding “which” human genome should be used to begin with. There are several alternatives. Many of the published and widely used genomics builds contain inconsistencies that make comparing results across them more challenging and less reliable. All these “builds” are being simultaneously used to generate increasing amount of information, that all gets “mixed” together propagating errors and inconsistencies. But fear not:

18.7 How complete is the data that will I obtain?

Let's take the sequence data for chromosome 22 of the human genome, which is distributed from the UCSC data site at <http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/>

```
# Download and unzip the file on the fly.
```

```
curl http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz | gunzip -
```

```
# Look at the file  
cat chr22.fa | head -4
```

it produces:

Ok, first let's deal with a little surprise. N stands for "unknown base". It looks like chromosome 22 starts with a series of unknown bases in it. Whatever we were about to investigate got derailed, what's with all the Ns? Thankfully we have Unix tools at our disposal to poke and prod at this data. Let's see just how many unknown bases there are for this chromosome? We can use grep with the flags -o (only matching) on the N pattern:

```
cat chr22.fa | grep -o N | head
```

Now count the lines:

```
cat chr22.fa | grep -o N | wc -l
```

The result comes at 11,658,691, over 11 million bases! Is that a lot? Ok, so how big is chromosome 22? We have installed many tools that could be used to print that information

```
seqkit stat chr22.fa
```

or.

infoseq chr22.fa

or, why not do it the hard way with bioawk:

```
cat chr22.fa | bioawk -c fastx '{ print length($seq) }'
```

The above code produces 50,818,468. In summary, 11 million out of 50 million bases (~ 23%) of chr22 are currently unknown.

Let's think about that a bit. Two decades after the announcement of the successful assembly of the entire human genome, 23% of chromosome 22 is listed as unknown. Ask a biologist to estimate how much of chromosome 22 is labeled as Ns see what they say. You will most likely experience a disconnect between their belief and reality.

But is it truly unknown? Well, not exactly. There are pieces of DNA that have been sequenced but have not yet been localized. Often these regions will fall into centromeres and telomeres. These un-localized pieces may be distributed separately. Yet in the vast majority of publications, when authors state that they are using the human genome, they will use only the chromosome 22 as indicated above. The one were 22% of it will be considered as "unknown" which is just a scientifically palatable word for "missing."

You have two choices: you can go out there and shout into the void: "*somebody should be doing something!*" or perhaps As Heng Li puts it, keep on the sunny side and say: "*Welcome to bioinformatics!*"

18.8 Final thoughts on data

Knowing how to solve these challenges is what makes bioinformaticians valuable. That's why you are here, no? The Biostar Handbook will guide you in the next chapters so that you can look like the genius you actually are.

Chapter 19

Biological data sources

19.1 Where is biomedical data stored?

Walter Goad of the Theoretical Biology and Biophysics Group at Los Alamos National Laboratory and others established the Los Alamos Sequence Database in 1979, which culminated in 1982 with the creation of the public GenBank. (source: Wikipedia)

GenBank can be considered as the first large-scale, public repository of biomedical data. Today, it is integrated with the services provided by the *National Center for Biotechnology Information (NCBI)*.

Today, just about all biomedical data is distributed via various web-based services. You may already be familiar with one or more such services; for example, NCBI Home Page¹ is an important entry point to a wealth of information.

Early warning: when visiting bioinformatics data sites, you are likely to be subjected to an information overload coupled with severe usability problems. Typically, it is surprisingly difficult to find the relevant, up-to-date, and correct information on any entity of interest. With the growth of data, we anticipate the situation only to worsen.

¹<http://www.ncbi.nlm.nih.gov/>

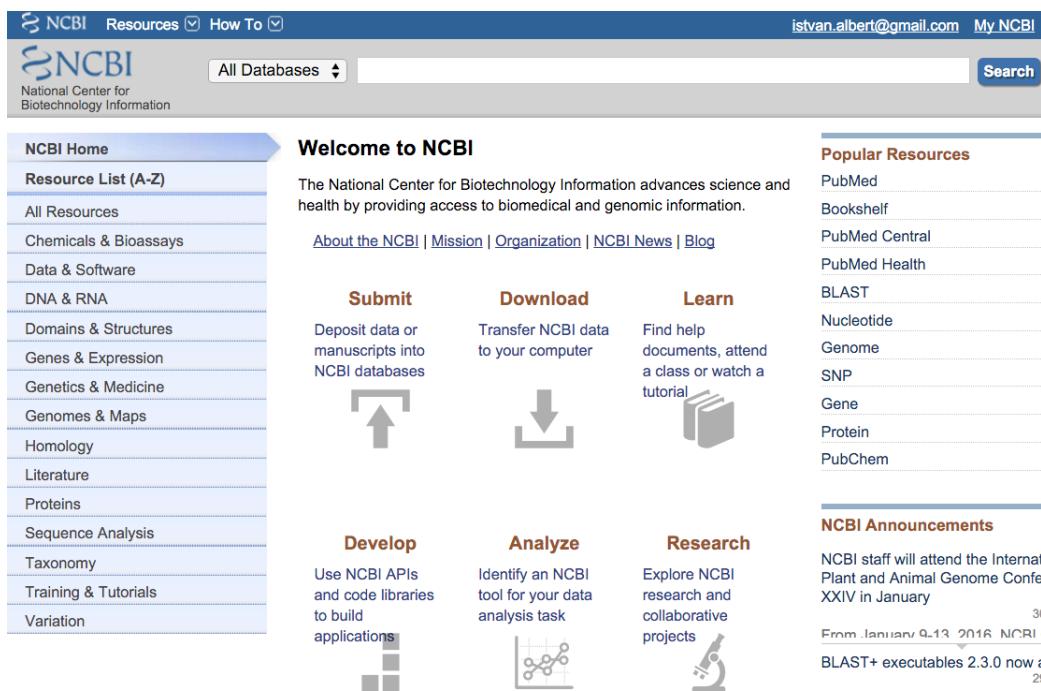


Figure 19.1

It is not your fault: it is theirs. Persevere and keep at it. You will eventually find what you need.

19.2 What are the major DNA data repositories?

DNA sequences collected by scientists are deposited in databases and made available to the public via the Internet. The INSDC: International Nucleotide Sequence Database Collaboration² has assumed stewardship for maintaining copies of the so-called “primary DNA data.” The member organizations of this collaboration are:

- NCBI: National Center for Biotechnology Information³

²<http://www.insdc.org/>

³<http://www.ncbi.nlm.nih.gov/>

- EMBL: European Molecular Biology Laboratory⁴
- DDBJ: DNA Data Bank of Japan⁵

The INSDC has set up rules on the types of data that will be mirrored. The most important of these from a bioinformatician's perspective are:

- GenBank⁶ contains all annotated and identified DNA sequence information
- SRA: Short Read Archive⁷ contains measurements from high throughput sequencing experiments
- UniProt: Universal Protein Resource⁸ is the most authoritative repository of protein sequence data.
- Protein Data Bank (PDB)⁹ is the major repository of 3D structural information about biological macromolecules (proteins and nucleic acids). PDB contains structures for a spectrum of biomolecules - from small bits of proteins/nucleic acids all the way to complex molecular structures like ribosomes.

19.3 What kind of other data sources are there?

Beyond the primary DNA data, various institutes may also hosts a variety of services and vast numbers of annotated datasets that may be supporting the work performed in each organization.

Also, there are different tiers of other organizations that have taken it upon themselves to reorganize, rename, and process existing data and provide additional functionalities and visualizations. Scientific interest groups formed around model organisms may also maintain their resources:

- UCSC Genome Browser¹⁰ invented the graphical browser visualization of genomes. Today it offers comprehensive comparative genomics data

⁴<http://www.embl.org/>

⁵<http://www.ddbj.nig.ac.jp/>

⁶<http://www.ncbi.nlm.nih.gov/genbank/>

⁷<http://www.ncbi.nlm.nih.gov/sra>

⁸<http://www.uniprot.org/>

⁹<http://www.rcsb.org/pdb/home/home.do>

¹⁰<http://hgdownload.soe.ucsc.edu/downloads.html>

across vertebrate genomes.

- FlyBase¹¹ is the database of *Drosophila* (fruit fly) genes and genomes.
- WormBase¹² is the primary resource for nematode biology.
- SGD: *Saccharomyces* Genome Database¹³ provides comprehensive integrated biological information for the budding yeast *Saccharomyces cerevisiae* along with search and analysis tools to explore these data.
- RNA-Central¹⁴ is a meta-database that integrates information from several other resources.
- TAIR¹⁵ The Arabidopsis Information Resource is the primary resource for genetic and molecular data about *Arabidopsis thaliana*, a model higher plant.
- EcoCyc¹⁶ (Encyclopedia of *E. coli* Genes and Metabolic Pathways) is a scientific database for the bacterium *Escherichia coli* K-12 MG1655.

Beyond the second tier of comprehensive databases, there are a large number of specialized data resources, often started and/or managed by single individuals with little-to-no support.

Describing each resource in detail is beyond the scope of this book. Most claim to provide “self-discoverable” interface elements, though their usability is typically far from optimal.

It is also hard to shake the feeling that each data provider’s ultimate goal is to be “too big to fail”; hence, there is an eagerness to integrate more data for the sole purpose of growing larger. It does not help that interface design is often supervised by scientists and this typically means jamming way too much information into way too little space. As a rule, using bioinformatics data resources is frustrating, requires patience, and often requires the suspension of disbelief.

Among the most significant challenges for a newcomer, however, is where to look for authoritative and up-to-date information. No database will ever admit that their data is not quite up to date

¹¹<http://flybase.org/>

¹²<https://www.wormbase.org>

¹³<http://www.yeastgenome.org/>

¹⁴<http://rnacentral.org/>

¹⁵<https://www.arabidopsis.org/>

¹⁶<http://ecocyc.org/>

or incomplete.

In addition to using different naming schemes, the data formats and data content will vary from resource to resource, adding no small confusion when trying to combine and reconcile information gleaned from different sources.

19.4 Is there a list of “all” resources?

Once a year the journal Nucleic Acids Research publishes its so-called “database issue”. Each article of this issue of the journal will provide an overview and updates about a specific database written by the maintainers of that resource.

- View the NAR: 2019 Database Issue¹⁷.

19.5 What’s in a name?

Two submitters using the same systematic names to describe two different genes would lead to very confusing situation. This makes it necessary to associate unique ids with each biological entity.

On the other hand, making gene names mnemonics that reflect their function, such as calling a *pigment dispersing factor* a *PDF*, or *Protein Kinase A* as *PKA* helps immensely during the biological interpretation process. The problem, of course, is that there could be many different variants of *PKA* in different genomes; moreover, potentially different terms could be shortened to similar or even identical mnemonics.

In general, all biological terms are defined by multiple names: there is a common name, a systematic name, a database-specific name, an accession number and sometimes a so-called locus. The common names typically carry a biological meaning, whereas systematic names are built with rules that make them unique. The accession and loci values may also follow various rules, though it is common that the original design of the rules turned out

¹⁷<https://academic.oup.com/nar/issue/47/D1>

to be insufficient to cover all use cases. Some identifiers fall out of use. Once widely used, GenBank gi numbers are now considered obsolete and were replaced by accession numbers. Other identifiers such as “locus” lack a proper definitions and should be avoided. The word locus is more appropriate in identifying a region in a genome, rather than a certain biological entity.

Most community projects formed around model organisms have adopted custom gene naming conventions that may only apply to data produced by that community.

19.6 Project systematic names

Different organizations may come up with their own naming schemes that are unique for an organism. For example, the *Saccharomyces* Genome database project has detailed documentation on naming conventions¹⁸ for every element. For example:

- Gene naming ADE12: The gene name should consist of three letters (the gene symbol) followed by an integer
- Open reading frame naming YGR116W: First letter ‘Y’ (‘Yeast’); the second letter denotes the chromosome number (‘A’ is chr I, etc.); the third letter is either ‘L’ or ‘R’ for left or right chromosome arm; next is a three digit number indicating the order of the ORFs on that arm of a chromosome starting from the centromere (like a highway exit), irrespective of strand; finally, there is an additional letter indicating the strand, either ‘W’ for Watson (forward) or ‘C’ for Crick (reverse). Thus, YGR116W is the 116th ORF right of the centromere on chromosome VII on the forward strand.

You may stop for a second and think about whether this ORF naming convention is an optimal long term solution (hint: it is not).

For example, naming strands of DNA to be W/C after Watson¹⁹ and Crick²⁰ is inappropriate (and not just because James Watson turned out to be a jerk²¹). The DNA has orientation, that may be forward and reverse, these

¹⁸<http://www.yeastgenome.org/help/community/nomenclature-conventions>

¹⁹https://en.wikipedia.org/wiki/James_Watson

²⁰https://en.wikipedia.org/wiki/Francis_Crick

²¹<https://liorpachter.wordpress.com/2018/05/18/james-watson-in-his-own-words/>

words make sense, they express something. There is no reason to introduce a new terminology, that adds no new information, in addition to the existing one.

Then why call chromosome 1 as A, what is the benefit of introducing another layer of remapping, 1 is A, 2 is B ... what is the 13th letter? It does not quite help. It even used to be that chromosomes were labeled with roman numerals `chrI`. Thankfully the practice using roman numeral has been changed, though note how the instructions above that we copied from the YGD website still indicate roman numerals.

Then what about the Y there, what problem does that solve? Do scientists often confuse human ORFs with yeast ones? I have a hard time imagining that. Telling apart ORFs from two yeast strains is a far more common need for a scientist working in this field. The present naming scheme does not help there.

Do you see how the For Big-Data Scientists, ‘Janitor Work’ Is Key Hurdle to Insights²² rings true?

We wrote this all up not to criticize (well maybe a little bit) but to demonstrate to readers that they will often have to deal with peculiarities that appear to make little sense. It helps to remember that many of these problems originate in local optimizations, ad-hoc conventions or even regrettable historical rationale, rather than well designed and well thought out decisions.

²²http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html?_r=0

Chapter 20

Common data types

A typical biological data analysis will require you to obtain and manipulate data that may be categorized as:

1. Data that captures prior knowledge (aka reference: FASTA, GFF, BED)
2. Experimentally obtained data (aka sequencing reads: FASTQ)
3. Data generated by the analysis (aka results: BAM, VCF, formats from point 1 above, and many nonstandard formats)

Below we list some of the most common bioinformatics formats. We will discuss each of these formats in more detail; here we'll first give you a quick overview of their purpose. Remember that each of these formats may contain information that overlaps with other forms. The optimization of each format makes it suited to specific goals. With time you will immediately be able to recognize the format from its extension or

20.1 A Quick look at the GENBANK format.

The “first” and most ancient, generic biological data format. Extensions .gb, .genbank. The GenBank format is optimized for reading by humans. Not suited for large scale data processing.

```
efetch -db nuccore -id NM_000020 -format gb | head
```

will produce output in GenBank:

```
LOCUS      NM_000020          4263 bp  mRNA  linear  PRI 23-DEC-2018
```

```

DEFINITION Homo sapiens activin A receptor like type 1 (ACVRL1), transcript
          variant 1, mRNA.
ACCESSION NM_000020
VERSION NM_000020.2
KEYWORDS RefSeq.
SOURCE Homo sapiens (human)
ORGANISM Homo sapiens
          Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
          Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;

...
regulatory      4241..4246
          /regulatory_class="polyA_signal_sequence"
          /gene="ACVRL1"
          /gene_synonym="ACVRL1; ALK-1; ALK1; HHT; HHT2; ORW2;
          SKR3; TSR-I"
polyA_site       4263
          /gene="ACVRL1"
          /gene_synonym="ACVRL1; ALK-1; ALK1; HHT; HHT2; ORW2;
          SKR3; TSR-I"
ORIGIN
  1 aggaaacggt ttatttaggag ggagtgggtgg agctggcca ggcaggaaga cgctggaata
  61 agaaaacatTT ttgctccagc ccccatccca gtccccggag gctgccgcgc cagctgcgcc
  121 gagcgagccc ctcccccggct ccagccccggt ccggggccgc gcccgaccc cagccccggc
  181 tccagcgctg gcggtgcaac tgcggccgcg cggtggagggg gaggtggaccc cggtccggccg
...

```

As you can see above it is a fairly complex format where information is distributed throughout the file. For that reason, very few tools can operate on GenBank files.

20.1.1 Quick look at the FASTA format

The same GenBank record above may be formatted in the FASTA format. Typical extension .fa, .fasta, .fna, sometimes .seq. This format retains only some of the sequence information of a GenBank record.

```
efetch -db nuccore -id NM_000020 -format fasta | head
```

will produce

```
>NM_000020.2 Homo sapiens activin A receptor like type 1 (ACVRL1), transcript var-
AGGAAACGGTTATTAGGAGGGAGTGGTGGAGCTGGGCCAGGCAGGAAGACGCTGGAATAAGAACATTT
TTGCTCCAGCCCCATCCCAGTCCCAGGCTCCGCGCCAGCTGCGCCAGCGAGGCCCTCCCGCT
CCAGCCCGGTCCGGGCCCGCAGCCCGACCCAGCCCGCCGTCAGCGCTGGCGGTGCAACTGCGCCCG
CGGTGGAGGGAGGTGGCCCCGGTCCGCCAAGGCTAGCGCCCCGCCACCCGAGAGCGGGCCAGAGGG
```

Note how the header seems to have context-related information as free text.

20.2 A quick look at the FASTQ format

The FASTQ format is meant to store experimental sequence measurements produced by a sequencing instrument. It can be thought of as FASTA with qualities.

```
fastq-dump -X 100 -Z SRR1553607 | head
```

will produce:

```
@SRR1553607.1 1 length=202
GTTAGCGTTGTTGATCGCGACGCAACAACTGGTAAAGAATCTGGAAGAAGGATATCAGTTCAAACGCTCAAG
+SRR1553607.1 1 length=202
BB@FFFFFHBBBBJJJJJJJJJJJJJJJJGHIJJJJJJJJHHHHHFFFFEEEEEEDDDDDDDDD
@SRR1553607.2 2 length=202
GGTGTAAAGCACAGTACTCGGCCACATCGCCTTGTTAATGAAGTTGGGTATCAACTTCATCCCCAAT
+SRR1553607.2 2 length=202
?@;BDDDDFHFHFFFGIIGHIIJJGJIGIJIIIGDGGGHEIGJIIIGIJJ5@FGHJJIEGGEEHHFFFFF
```

(the above input was truncated a little bit for legibility). The header lines may contain extra information for example, instead of

```
@SRR1553607.1 1 length=202
```

the instrument will typically specify more information on the context of how this measurement was made.

```
@EAS139:136:FC706VJ:2:2104:15343:197393 1:N:18:1
```

20.3 A quick look at the GFF/GTF/BED formats

The GFF/GTF/BED formats are the so-called “interval” formats that retain only the coordinate positions for a region in a genome. Each is tab delimited and will contain information on the chromosomal coordinate, start, end, strand, value and other attributes, though the order of the columns will depend on the feature.

Three column BED has the columns for chromosome, start, end

```
chr7    127471196 127472363
chr7    127472363 127473530
chr7    127473530 127474697
```

Six column BED add name, value and strand:

```
chr7    127471196 127472363 Pos1 0 +
chr7    127472363 127473530 Pos2 0 +
chr7    127473530 127474697 Pos3 0 +
```

Up to 12 columns may be specified for BED formats. The coordinates start at 0 (that is the first valid coordinate).

The GFF/GTF formats¹ are 9 column tab-delimited formats. The coordinates start at value 1.

```
chr1 . mRNA      1300 9000 . + . ID=mRNA0001;Name=sonichedgehog
chr1 . exon      1300 1500 . + . ID=exon00001;Parent=mRNA0001
chr1 . exon      1050 1500 . + . ID=exon00002;Parent=mRNA0001
```

See the details on the UCSC formats² page. In the GTF format, the layout of the 9th (last column) is different. Also, GTF formats are valid only if they have values for “gene_id” and “transcript_id” specified on the 9th column.

20.4 A quick look at the SAM/BAM formats

The SAM/BAM formats are so-called Sequence Alignment Maps. These files typically represent the results of aligning a FASTQ file to a reference FASTA file

¹<https://useast.ensembl.org/info/website/upload/gff3.html>

²<https://genome.ucsc.edu/FAQ/FAQformat.html>

and describe the individual, pairwise alignments that were found. Different algorithms may create different alignments (and hence BAM files).

```
samtools view http://data.biostarhandbook.com/bam/demo.bam | head -5
SRR1553425.13617    163 AF086833.2 46 60 101M   = 541 596 GAATAACTATGAGGAAGATT
SRR1553425.13755    99  AF086833.2 46 60 101M   = 46 101 GAATAACTATGAGGAAGATT
SRR1553425.13755    147 AF086833.2 46 60 101M   = 46 -101  GAATAACTATGAGGAAGA
SRR1553425.11219    2227   AF086833.2 47 60 71H30M = 146 171 AATAACTATGAGGAAGA
```

20.4.1 Quick look at the VCF format

The VCF (Variant Call Format) describes the variation of alignments relative to a reference. Extensions: `.vcf`, `vcf.gz`, `.bcf`. If the extension ends with `.gz` it is a block gzipped file (see the chapter on Data Compression). A VCF file is typically created from a BAM file, whereas the BAM file was created from a FASTQ and a FASTA file.

Thus think of the VCF file as a file that captures the differences of for each of the sequences in the FASTQ file relative to the genome in the FASTA file.

```
bcftools view -H http://data.biostarhandbook.com/variant/subset_hg19.vcf.gz | head
19 400410 rs540061190 CA C 100 PASS AC=0;AF=0.00179712;AN=12;NS=2504;DP=777
19 400666 rs11670588 G C 100 PASS AC=5;AF=0.343251;AN=12;NS=2504;DP=8445;K
19 400742 rs568501257 C T 100 PASS AC=0;AF=0.000199681;AN=12;NS=2504;DP=15
19 400819 rs71335241 C G 100 PASS AC=0;AF=0.225839;AN=12;NS=2504;DP=10365
19 400908 rs183189417 G T 100 PASS AC=1;AF=0.0632987;AN=12;NS=2504;DP=1316
```

20.5 Can I convert between formats.

If one format contains information that another format stores then you can typically convert it. For example you can make a FASTA or GFF file from a GENBANK format. Reversing the process would create an incomplete GENBANK file.

```
# Get a genbank file.
efetch -db nucleotide -format=gb -id=AF086833 > AF086833.gb

# We can convert the GenBank file into GFF3.
```

```
cat AF086833.gb | seqret -filter -feature -osformat gff3 > AF086833.gff  
  
# We can convert the GenBank file into FASTA.  
cat AF086833.gb | seqret -filter -feature -osformat fasta > AF086833.fa
```

20.6 What is reference data?

Reference data represents the snapshot of the accumulated knowledge at a given point in time.

Here is worth to note that the information about the human genome, because of its importance to society, is treated quite differently than information for just about all other genomes. The human genome related data has most “standardization” applied to it - though mostly by necessity.

Even so painful silly disagreements remain, for example, some organization may label human chromosomes by word + number (for example `chr1`, `chr2`, ...) other organizations name chromosomes by numbers alone (for example `1`, `2`, ...). The reasons for these and the many other discrepancies are numerous, in a nutshell, it can be condensed down to the following: it takes a lot of work to standardize, yet there too few incentives to reward those that choose to do so. From your perspective as a bioinformatician, it means that you will need to navigate these troubled waters by learning to recognize and navigate between the many shoals.

For other organisms that have a large following, various organizations stepped in to standardize the representation of this accumulated knowledge. The “local” standards are occasionally better enforced, often that only means that the amount of accumulated data is too low, it is easier to “police” it.

20.7 What are genomic builds?

As more information is discovered previous representations of the genome may need to be corrected and rearranged. Genomic builds represent an “edition,” a snapshot of the information in time. Especially when obtaining data from disparate sources, it is essential to ensure that all evidence refers to the same baseline information.

When a genomic build has changed the information associated with the genome must be changed as well. For example, just adding a single base at the beginning of a genome means that the coordinates of all subsequent elements must be changed, shifted by one.

Because genomes get rearranged in various and complex ways: inversions, insertions, deletions, relocations, sometimes in an overlapping manner, remapping a coordinate to a new location turns out to be a surprisingly challenging operation. Note that even though if the sequence were not to change substantially the coordinates may end up altered in a manner that could be challenging or even impossible to reconcile with prior data. Some locations in the previous version of a genome may “not exist” in the new genome.

20.8 Is there a list of “all” resources?

Once a year the journal Nucleic Acids Research publishes its so-called “database issue”. Each article of this issue of the journal will provide an overview and updates about a specific database written by the maintainers of that resource.

- View the NAR: 2019 Database Issue³.

20.9 Should download data first or get data on-demand?

The size and frequency of the need for a data will determine your use case. Typically we only perform on-demand download for small (<10M) datasets and during exploratory data analyses. We recommend creating a directory for your reference data that you would store the references for access across all your projects).

```
# Make a directory of a human reference build.  
mkdir -p ~/refs/hg38  
  
# Get chromosome 22 and store it locally.
```

³<https://academic.oup.com/nar/issue/47/D1>

20.9. SHOULD DOWNLOAD DATA FIRST OR GET DATA ON-DEMAND?211

```
curl http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz | gunzip -c  
  
# Look at the file  
cat ~/refs/hg38/chr22.fa | head -4  
  
# Print statistics on the sequence.  
infoseq ~/refs/hg38/chr22.fa
```

As always remember that rule “Make it fast.” Always test your processes on a subset of the genome like chromosome 22 to have a fast turnaround.

Chapter 21

Human and mouse genomes

As we mentioned before the information about the human and mouse genomes are treated in a more standardized fashion than the data from just about all other genomes. Even if you are not working with the human and mouse genomes, it helps if you familiarize yourself with the design/distribution choices made for this data.

The mouse genome has a special designation as it serves as the most studied experimental animal model for the human genome. For that reason, many of the standards of the human genome have been adopted for the mouse genome as well.

21.1 How many genomic builds does the human genome have?

38 as of today. Alas not only do different genomic builds exist, but various data distribution resources may also disagree on how to label the same data. For example, here are the last four genomic builds as marked at NCBI vs. UCS Genome Browser

- Build 35 released in 2004 is labeled as NCBI35 and hg17
- Build 36 released in 2006 is labeled as NCBI36 and hg18
- Build 37 released in 2009 is labeled as GRCh37 and hg19
- Build 38 released in 2013 is labeled as GRCh38 and hg38

Note the painful inconsistency of earlier years; it took until 2013 to get to the point where at least the number matches across. Then note just how long ago was build 37 released. It is about a decade old. You'd be surprised to know that this genome build is still in active use even though a better build has been available.

Within each year subsequent corrections (patches)¹ may be applied to the genome that does not alter the length of the genome or the coordinates of elements.

21.2 Why is the genomic build hg19 still in use?

Scientists that want to build upon results that integrate with information obtained older genome build may use that old genome as a reference, rather than using a new genome then “transferring” the results to the new genome.

21.3 Should we use the old version of a genome?

No. We would actively discourage the practice. While it is conceivable that there are legitimate uses of relying on obsolete genomic build such as hg19, we can't help but wonder whether it is a lack of awareness and ignorance that allows publications still using hg19 to clear the review process

Scientists keep following the practice; hence you will need to know how to remap information between genomes.

¹<https://www.ncbi.nlm.nih.gov/grc/help/patches/>

21.4 How do we transfer genomic coordinates between builds?

The term used to describe this process is called “lifting over” coordinates. There are web and command line applications that you may use:

- liftOver² from UCSC (web and command line)
- remap³ from NCBI (web)
- crossmap⁴ (command line)

The “lifting over” process requires the presence of a precomputed “chain” file (chain data) that describes how the new genome is different from the old one.

While the web interfaces may suffice for some tasks, we recommend command line tools that allow you to generate reproducible analyses.

```
conda install crossmap -y
```

Run the tool:

```
CrossMap.py
```

will print:

```
Program: CrossMap (v0.3.3)
```

Description:

```
CrossMap is a program for convenient conversion of genome coordinates and genome
format.
```

```
...
```

To perform a remapping [get a chainfile][chains]

```
# Get the chain file that maps from hg19 to hg38.
```

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg19/liftOver/hg19ToHg38.over.chain.gz
```

```
# Get a test data file that will be remapped.
```

```
wget http://data.biostarhandbook.com/data/ucsc/test.hg19.bed
```

```
# Run the remapping process.
```

²<https://www.ncbi.nlm.nih.gov/genome/tools/remap>

³<https://www.ncbi.nlm.nih.gov/genome/tools/remap>

⁴<http://crossmap.sourceforge.net/>

```
CrossMap.py bed hg19ToHg38.over.chain.gz test.hg19.bed test.hg38.bed
```

The commands above will remap `test.hg19.bed` to `test.hg38.bed`. As an example the coordinates:

```
chr1      65886334      66103176
```

were changed to:

```
chr1      65420651      65637493
```

21.5 Human gene naming

The HUGO Gene Nomenclature Committee⁵ is the only worldwide authority that assigns standardized nomenclature to human genes. The names assigned by HGNC are typically formed from a so-called stem (or root) symbol that is used as the basis for a series of approved symbols that are defined as members of either a functional or structural gene family. For example *CYP*: cytochrome P450; *HOX*: homeobox; *DUSP*: dual specificity phosphatase; *SCN2A*: sodium channel voltage-gated type II alpha 2 polypeptides etc.

Since the mouse and rat genomes are the closest human models and are widely studied, making human and mouse gene names follow similar rules has always been a priority of the Mouse Gene Nomenclature Committee (MGNC)⁶

The Vertebrate Gene Nomenclature Committee (VGNC)⁷ is an extension of the established HGNC (HUGO Gene Nomenclature Committee) project that names human genes. VGNC is responsible for assigning standardized names to genes in vertebrate species that currently lack a nomenclature committee. The current prototype VGNC naming species is the chimpanzee.

⁵<http://www.genenames.org/>

⁶<http://www.informatics.jax.org/mgihome/nomen/>

⁷<http://vertebrate.genenames.org/>

21.6 Which human data source should I trust?

First, recognize that while some data is nearly identical (for example the sequence data differs mainly in its completeness, information derived from processing results may be notably different. Also, different groups may rearrange data to serve their needs better, further complicating the picture.

When it comes to the human genome scientists typically use one of the following resources:

- GENCODE⁸
- NCBI: RefSeq⁹
- UCSC¹⁰

Ensembl¹¹, the joint project between EMBL-EBI and the Wellcome Trust Sanger Institute to publish data and software distributes the GENCODE¹² data as their human and mouse annotations. The data in Ensembl is published via release versions, currently at version 95:

- Ensembl FTP: <ftp://ftp.ensembl.org/pub/release-95/>

21.7 Which human/mouse genome data should I be using?

Most scientists are selecting a data source to integrate with other data that is present in that data source. The choice relates to the domain of application and the subculture that a scientist is a part of.

For example projects with medical and clinical applications tend to use RefSeq¹³ as typically being the most conservative yet best integrated with all available knowledge. Scientists interested in comparative genomics may fa-

⁸<https://www.gencodegenes.org/>

⁹<https://www.ncbi.nlm.nih.gov/refseq/>

¹⁰<http://hgdownload.soe.ucsc.edu/downloads.html>

¹¹<https://useast.ensembl.org/index.html>

¹²<https://www.gencodegenes.org/>

¹³<https://www.ncbi.nlm.nih.gov/refseq/>

vor UCSC¹⁴. GENCODE¹⁵ is the choice of those that work on so “model” organism or those that wish to use a more frequently updated and more comprehensive database.

21.8 Is there a better resource for human annotations?

The unfortunately named CHESS¹⁶ project, a name that is an acronym for **C**omprehensive **H**uman **E**xpressed **S**equence**S** was published as CHESS: a new human gene catalog curated from thousands of large-scale RNA sequencing experiments reveals extensive transcriptional noise (Genome Biology, 2018)¹⁷ claims to have created a new list of human genes and transcripts.

Pet peeve: Naming this resource as CHESS was incredibly shortsighted and unfortunate as it will make searching and finding relevant hits for it online virtually impossible.

This paper and data repository are worth studying in more detail. It is well written article that provides a detailed background on the current challenges for identifying and annotating the human genome. Besides, it offers what we believe to be the best current annotations for the human genome. We do foresee other resources integrating the information that this paper has published.

21.9 How do I access NCBI RefSeq and GenBank?

Entrez is the interface into the data stored at NCBI.

- NCBI web: <https://www.ncbi.nlm.nih.gov/>

¹⁴<http://hgdownload.soe.ucsc.edu/downloads.html>

¹⁵<https://www.gencodegenes.org/>

¹⁶<http://ccb.jhu.edu/chess/>

¹⁷<https://genomebiology.biomedcentral.com/articles/10.1186/s13059-018-1590-2>

- NCBI FTP: <ftp://ftp.ncbi.nlm.nih.gov/>
- See the RefSeq publication¹⁸ in the database issue of NAR

NCBI and Entrez (the interface into GenBank) are the best resources to obtain any sequence that humanity has ever found and stored. But in the cost of handling that immense variety is that you can forget about expecting any naming scheme that “makes sense.” Chromosome 1 of the human genome will not be called as such; instead it will be called GCF_000001405.38 - suffice to say you’ll have a hard time working with data from outside of NCBI.

Entrez is excellent at getting you any version of any gene that was ever known - even if this means getting thousands of hits, most of which may be unreliable.

21.10 What can I get from ENSEMBL?

Ensembl is the interface into the data store at EBI.

- Ensembl web: <http://useast.ensembl.org/index.html>
- Ensembl FTP: <ftp://ftp.ensembl.org/pub/>

Ensembl has the most detailed annotations of the genomes. Ensembl operates via releases¹⁹. Compared to GenBank and RefSeq Ensembl stores only a subset of the most studied organisms.

21.11 A working strategy for finding reference information

1. Locate the resource
2. Understand the file structure
3. Write the commands to get the file you need

Example: Visit <https://www.gencodegenes.org/human/>. Say you want the transcript sequences in FASTA format. Find the URL that you need. Make a note and store these commands for the future:

¹⁸<https://academic.oup.com/nar/article/44/D1/D733/2502674>

¹⁹<http://useast.ensembl.org/info/website/news.html>

21.11. A WORKING STRATEGY FOR FINDING REFERENCE INFORMATION219

```
wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_29/gencode.v29.ti  
gunzip gencode.v29.transcripts.fa.gz
```

Sometimes/often the descriptions are insufficient and you need to download and investigate the data to fully understand what it contains.

Chapter 22

Automating access to NCBI

22.1 Note

On December 2018 NCBI has introduced a limit to how many requests one may make per second. When we chain command like discussed in some examples below, we may end up unwittingly exceeding this limit if some commands finish sooner than a second. To avoid rate limit errors you may need to occasionally slow down your commands by suspending execution with `sleep 1`. To sleep before doing another task connect the two commands with the `&&` symbols and place them within parentheses like so:

```
# Sleeps one second before counting the number of lines.  
ls -1 | (sleep 1 && wc -l)
```

Alternatively, a new version of Entrez Direct is available that works correctly but is not yet available in bioconda. To install the source code version of `entrez-direct` see the page on How to install Entrez-Direct from source code. Using the solutions above will help you avoid rate limit errors.

22.2 What is Entrez?

The NCBI data store is a repository of gargantuan proportions that stores data far beyond the primary DNA data that the NCBI's original mission called for.

Currently, NCBI provides storage, categorization, and context on just about all life-sciences-oriented information, from raw datasets to curated results. It even stores the content of most of the scientific publications themselves.

Entrez is NCBI's primary text search and retrieval system that integrates the PubMed database of biomedical literature with 39 other literature and molecular databases including DNA and protein sequence, structure, gene, genome, genetic variation, and gene expression.

With the large size come the challenges - the interfaces can be cumbersome and complicated; it is typically not easy to find the information that one is looking for and is easy to get lost across the many pages. The challenges of data access make command line access even more important as it makes all actions explicit and reproducible.

22.3 How is Entrez pronounced?

Technically, it is a French word, and the French pronunciation would sound similar to “*on tray*.” But if you say “*let’s search on-tray*” no will know what you mean. Thus we recommend pronouncing it as it written: “*en-trez*”

22.4 How do we automate access to Entrez?

From early on NCBI offered a web API interface called Entrez E-utils¹ then later released a toolset called Entrez Direct².

¹<http://www.ncbi.nlm.nih.gov/books/NBK25500/>

²<http://www.ncbi.nlm.nih.gov/books/NBK179288>

Biostar News of the Day

Ncbi Releases Entrez Direct, The Entrez Utilities On The Unix Command Line³

NCBI has just released Entrez Direct, a new software suite that enables users to use the UNIX command line to access NCBI databases directly, as well as to parse and format the data to create customized downloads.

22.5 How is data organized in NCBI?

The diversity of data sources and the need to keep up with an evolving body of knowledge poses challenges when trying to identify current and past information unambiguously.

As an example, search the NCBI nucleotide database for the words “Ebola virus 1976⁴”. Among the first hits will be data that starts with:

```
LOCUS      AF086833          18959 bp   cRNA   linear  VRL 13-FEB-2012
DEFINITION Ebola virus - Mayinga, Zaire, 1976, complete genome.
ACCESSION   AF086833
VERSION     AF086833.2  GI:10141003
```

Note the accession number AF086833 and the version number AF086833.2. Because the number does not contain an underscore we know that this data is in GenBank but is not in RefSeq.

An accession number, for example, AF086833, applies to the complete database record and remains stable even if updates/revisions are made to the record.

The version number is formed by adding a dotted number such as .2 to the accession number to form: AF086833.2. This number is a unique identifier for the sequence data within its record.

⁴<https://www.ncbi.nlm.nih.gov/nuccore/?term=Ebola+virus+1976>

If any change occurs to the sequence data, no matter how large or small, the version number for that sequence is incremented by one decimal.

Starting in 2015, NCBI has decided to phase out an older standard called GI numbers that were integer numbers associated with a record. An older document may still refer to GI or gi numbers.

22.6 How do I use Entrez E-utils web API?

Entrez web API allows us to query NCBI data sources via a specially constructed URL. A query URL will be of the form `https://service.nih.gov?param1=value1¶m2=value2`.

Since the & character has a special meaning for the shell, we need to either put the URL within single quotes, or we need to “escape/protect” the & character by placing a \ in front of it like so \&

Returns the the data for accession number AF086833.2 in the FASTA format:

```
curl -s https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?id=AF086833.2&db=nucleotide
```

or equivalently:

```
curl -s 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?id=AF086833.2&db=nucleotide'
```

Read the Entrez E-utils⁵ documentation for information on all parameters.

22.7 How do I use Entrez Direct?

The web-based Entrez E-utils can be unwieldy and are not well suited for exploratory analysis. The tool suite called `entrez-direct` (EDirect) simplifies this web access via a series of tools, including `efetch` and others:

```
efetch -db=nuccore -format=gb -id=AF086833 | head
```

We can retrieve the same information in different formats:

```
# Accession number AF086833 in Genbank format.
```

```
efetch -db=nuccore -format=gb -id=AF086833 > AF086833.gb
```

⁵<http://www.ncbi.nlm.nih.gov/books/NBK25500/>

```
# Accession number AF086833 in Fasta format.
efetch -db=nuccore -format=fasta -id=AF086833 > AF086833.fa

# efetch can take additional parameters and select a section of the sequence.
efetch -db=nuccore -format=fasta -id=AF086833 -seq_start=1 -seq_stop=3
```

It can even produce the sequence from reverse strands:

```
efetch -db=nuccore -format=fasta -id=AF086833 -seq_start=1 -seq_stop=5 -strand=1
efetch -db=nuccore -format=fasta -id=AF086833 -seq_start=1 -seq_stop=5 -strand=2
```

But just what exactly did `strand=2` do? Is it the complement or reverse complement? Always ensure you understand what parameters do. Don't assume you understand their function. Below are the actual outputs of the last two commands. We can see that the second command produces the reverse complement as indicated by the `c5-1` tag in the FASTA file header.

```
>gb|AF086833.2|:1-5 Ebola virus - Mayinga, Zaire, 1976, complete genome
CGGAC
```

```
>gb|AF086833.2|:c5-1 Ebola virus - Mayinga, Zaire, 1976, complete genome
GTCCG
```

22.8 How do we search with Entrez Direct?

Once we have a project accession number, say PRJNA257197, we can use it to search for the data that comes with it. We can locate project accession numbers in published papers, or we find them in the supporting information.

```
esearch -help
esearch -db nucleotide -query PRJNA257197
esearch -db protein -query PRJNA257197
```

The `esearch` command produces a so-called “environment” that will be passed into other Entrez direct programs. You can get a sense of what the results are from looking at the results of the search:

```
<ENTREZ_DIRECT>
<Db>nucleotide</Db>
<WebEnv>NCID_1_17858181_130.14.18.34_9001_1472654178_1691087027_0MetA0_S_Mega
```

```
<QueryKey>1</QueryKey>
<Count>249</Count>
<Step>1</Step>
</ENTREZ_DIRECT>
```

To fetch the data for a search, pass it into `efetch`:

```
esearch -db nucleotide -query PRJNA257197 | efetch -format=fasta > genomes.fa
```

To get all proteins:

```
esearch -db protein -query PRJNA257197 | efetch -format=fasta > proteins.fa
```

22.9 How to do more work with Entrez Direct?

Entrez Direct is a very sophisticated tool with surprising powers - though to fully use it, one needs to have a firm understanding of the XML file format that `efetch` returns.

Specifically, the `xtract` tool in Entrez Direct allows navigating and selecting parts of an XML file. `xtract` is conceptually very similar to the so-called XML XPATH constructs that allow access to various nodes of an XML document. It is somewhat of a reinvention of the same concepts, designed to serve more data processing needs.

As an example here is a construct that matches taxonomy ids to more readable names:

```
efetch -db taxonomy -id 9606,7227,10090 -format xml | xtract -Pattern Taxon -first TaxId $
```

Produces:

```
9606 Homo sapiens human Primates
7227 Drosophila melanogaster fruit fly Invertebrates
10090 Mus musculus house mouse Rodents
```

More in-depth documentation may found in the NCBI manual titled: Entrez Direct: E-utilities on the UNIX Command Line⁶

⁶<https://www.ncbi.nlm.nih.gov/books/NBK179288/>

Chapter 23

Entrez Direct by example

In-depth documentation may found in the NCBI manual titled: Entrez Direct: E-utilities on the UNIX Command Line¹

Note: This chapter will be collect entrez-direct code examples from the book and recipes.

23.1 How do I use efetch?

```
efetch -db=nuccore -format=gb -id=AF086833 | head
```

We can retrieve the same information in different formats:

```
# Accession number AF086833 in Genbank format.
```

```
efetch -db=nuccore -format=gb -id=AF086833 > AF086833.gb
```

```
# Accession number AF086833 in Fasta format.
```

```
efetch -db=nuccore -format=fasta -id=AF086833 > AF086833.fa
```

```
# efetch can take additional parameters and select a section of the sequence.  
efetch -db=nuccore -format=fasta -id=AF086833 -seq_start=1 -seq_stop=3
```

It can even produce the sequence from reverse strands:

¹<https://www.ncbi.nlm.nih.gov/books/NBK179288/>

```
efetch -db=nuccore -format=fasta -id=AF086833 -seq_start=1 -seq_stop=5 -strand=1
efetch -db=nuccore -format=fasta -id=AF086833 -seq_start=1 -seq_stop=5 -strand=2
```

But just what exactly did `strand=2` do? Is it the complement or reverse complement? Always make sure to understand what parameters do. Don't just assume you understand their function. Below are the actual outputs of the last two commands. We can see that the second command produces the reverse complement as indicated by the `c5-1` tag in the FASTA file header.

```
>gb|AF086833.2|:1-5 Ebola virus - Mayinga, Zaire, 1976, complete genome
CGGAC
```

the parameter `strand=2` will return the reverse complement:

```
>gb|AF086833.2|:c5-1 Ebola virus - Mayinga, Zaire, 1976, complete genome
GTCCG
```

23.2 How do I use efetch to obtain project related data?

Once we have a project accession number, say PRJNA257197, we can use it to search for the data that comes with it. Project accession numbers are included in published papers, or we find them in the supporting information.

```
esearch -help
esearch -db nucleotide -query PRJNA257197
esearch -db protein -query PRJNA257197
```

These commands produce a so-called “environment” that can be passed into other Entrez direct programs. You can get a sense of what the results are from looking at the results of the search:

```
<ENTREZ_DIRECT>
  <Db>nucleotide</Db>
  <WebEnv>NCID_1_17858181_130.14.18.34_9001_1472654178_1691087027_0Meta0_S_MegaStore_F
  <QueryKey>1</QueryKey>
  <Count>249</Count>
  <Step>1</Step>
</ENTREZ_DIRECT>
```

To fetch the data for a search, pass it into `efetch`:

```
esearch -db nucleotide -query PRJNA257197 | efetch -format=fasta > genomes.fa
```

To get all proteins:

```
esearch -db protein -query PRJNA257197 | efetch -format=fasta > proteins.fa
```

23.3 How do I get run information on a project?

```
esearch -db sra -query PRJNA257197 | efetch -format runinfo > runinfo.csv
```

Now we have a file with many columns, here are a few:

```
cat runinfo.csv | cut -d , -f 1,2,16 | head -3
```

prints:

```
Run,ReleaseDate,LibraryLayout
SRR1972917,2015-04-14 13:59:24,PAIRED
SRR1972918,2015-04-14 13:58:26,PAIRED
```

23.4 How do I get even more information on a project?

Another way (and why are there more ways to do the same thing?) to get run information is via the “summary” format:

```
esearch -db sra -query PRJNA257197 | efetch -format summary > summary.xml
```

This creates a file that technically speaking is an invalid XML file (gee thanks NCBI, sarcasm) so you cannot visualize it a browser, but can still be processed with xtract

```
cat summary.xml | xtract -pattern RUN_SET -element @accession | head
```

will produce:

```
SRR1972976
SRR1972975
SRR1972974
SRR1972973
```

This XML file contains more information than runinfo.

23.5 How do I extract taxonomy information?

Match taxonomy ids to more readable, common names:

```
efetch -db taxonomy -id 9606,7227,10090 -format xml > output.xml
```

Open up the file output.xml in a web browser to understand its content. It will look like

```
<TaxaSet>
  <Taxon>
    <TaxId>9606</TaxId>
    <ScientificName>Homo sapiens</ScientificName>
    <OtherNames>
      <GenbankCommonName>human</GenbankCommonName>
      <CommonName>man</CommonName>
    <Name>
      <ClassCDE>authority</ClassCDE>
      <DispName>Homo sapiens Linnaeus, 1758</DispName>
    </Name>
    <Name>
      <ClassCDE>misspelling</ClassCDE>
      <DispName>Home sapiens</DispName>
    </Name>
  ...

```

It is a full XML document with lots of content in it. The **xtract** tool matches and extracts elements from this document:

```
xtract -Pattern Taxon -first TaxId ScientificName GenbankCommonName Division
Produces:
```

```
9606   Homo sapiens   human   Primates
7227   Drosophila melanogaster fruit fly   Invertebrates
10090  Mus musculus   house mouse Rodents
```

Part V

DATA FORMATS

Chapter 24

Introduction to data formats

24.1 What is a data format?

As we stated before data is defined by:

1. The **information** it contains
2. The **design**, and **optimization** of that information.

The subsequent chapters will discuss in detail the most commonly used data formats. We will cover what each format was designed to represent and what optimizations it has that make the data suited for specific tasks.

These data format chapters do not need to be read in order, we have grouped them under one heading to make locating them later a more straightforward task.

24.2 Should I re-format (transform) my data?

A question that you will frequently face:

1. Is it better to reformat existing data into the format that you need?
2. Is it better to obtain data in different forms from the data source?

Here is an example. Take the Murine Leukemia Virus GenBank file:

```
# Fetch the sequence from NCBI.  
efetch -db nucore -id NC_001501 -format gb > NC_001501.gb
```

View the sequence with

```
cat NC_001501.gb | head
```

produces:

```
LOCUS      NC_001501          8332 bp ss-RNA    linear  VRL 13-AUG-2018
DEFINITION Moloney murine leukemia virus, complete genome.
ACCESSION  NC_001501
VERSION    NC_001501.1
DBLINK     BioProject: PRJNA485481
KEYWORDS   RefSeq.
SOURCE     Moloney murine leukemia virus (MoMLV)
ORGANISM   Moloney murine leukemia virus
Viruses; Ortervirales; Retroviridae; Orthoretrovirinae;
Gammaretrovirus; Murine leukemia virus.
...
...
```

Suppose you wanted the sequence in FASTA format. There are different methods to get this information. You could fetch the FASTA file directly with:

```
efetch -db nuccore -id NC_001501 -format fasta > NC_001501-version1.fa
```

or you could transform the existing GenBank with seqret:

```
cat NC_001501.gb | seqret -filter -osformat fasta > NC_001501-version2.fa
```

Are the two files the same? Let's look at the first two lines in each with:

```
cat NC_001501-version1.fa | head -2
cat NC_001501-version2.fa | head -2
```

The first command will show:

```
>NC_001501.1 Moloney murine leukemia virus, complete genome
TATGCGCCTGCGTCGGTACTAGTTAGCTAACTAGCTCTGTATCTGGCGGACCCGTGGAACTGACGAG
```

whereas the second command produces:

```
>NC_001501 NC_001501 Moloney murine leukemia virus, complete genome.
tatgcgcctgcgtcggtactagttagctaaactagctctgtatctggcggacccgtgg
```

Does not look like quite the same information.

Now the sequence itself may be the same, though you can't tell that by eye. Let's first see if the sequences are identical. We can run a pairwise

alignment to verify this. You could go about it in many ways, and here I will run `blastn` (covered in subsequent lecture) in a manner that it is only producing certain attributes of interest that can help me determine whether the sequences match:

```
blastn -query NC_001501-version1.fa -subject NC_001501-version2.fa -outfmt "7 qa"
will produce:
```

```
# BLASTN 2.7.1+
# Query: NC_001501.1 Moloney murine leukemia virus, complete genome
# Database: User specified sequence set (Input: NC_001501-version2.fa)
# Fields: query acc., subject acc., query length, subject length, alignment length,
# 5 hits found
NC_001501.1 NC_001501    8332     8332     8332    100.000
```

Above we only display the first alignment. It tells me that both sequences are 8332 bases long, the alignment is over 8332 bases and the percent identity is 100%. So I can see that the sequences are identical.

The most impactful difference is that the first file the sequence is named by accession version number `NC_001501.1`, whereas in the second file it is named by accession number `NC_001501` alone. This difference is significant as the naming of sequences can cause many problems and annoyances later on. We note that another difference is, of course, the capitalization of the sequence - that too may matter later.

The situation above captures a typical scenario that you will face all the time. You are out there, looking for data, collecting information, then out of the blue you need to quickly solve a problem you did expect to have to deal with: are two sequences identical?

Moreover, there may be multiple options to do transformations; you could be using a different tool, let's use `readseq`. Our transformation then would work like this:

```
cat NC_001501.gb | readseq -p -format=FASTA > NC_001501-version3.fa
```

let's see what we get:

```
cat NC_001501-version3.fa | head -2
```

produces:

```
>NC_001501 Moloney murine leukemia virus, complete genome. 8332 bp
```

```
tatgcgcctgcgtcggtacttagttagctaactagctctgtatctggcggaccctgg
```

Guess what, we have slightly different output yet again. I bet you did not see that coming ;-):

But the GenBank to FASTA transformation is one of the most straightforward changes out there. The sequence is fully embedded into the GenBank file. Once transformations are more complicated, you can expect even more differences.

24.3 When to re-format (transform) data?

In general, we found that if the data source offers data in different formats, the recommended approach is to obtain each data format individually and not convert between them. Each conversion is fraught with potential problems. This way, at least you can blame the data source.

When the data source does not offer data in different formats, or when you are not entirely sure that the data that you have represents the same genomic build, you should do the re-formatting work yourself. Make sure you do it right. Double check and be defensive. Nothing is more dangerous in data analysis than slightly “wrong” data.

Chapter 25

The GenBank format

25.1 What is the GenBank format?

GenBank format is one of the oldest bioinformatics data formats, originally invented to bridge the gap between a human-readable representation and one that can be efficiently processed by the computer. The format (defined here¹) has a so-called fixed-width format, where the first ten characters form a column that serves as an identifier and the rest of the lines are information corresponding to that identifier.

```
LOCUS      AF086833          18959 bp   cRNA   linear  VRL 13-FEB-2012
DEFINITION Ebola virus - Mayinga, Zaire, 1976, complete genome.
ACCESSION  AF086833
VERSION    AF086833.2  GI:10141003
KEYWORDS   .
SOURCE     Ebola virus - Mayinga, Zaire, 1976 (EBOV-May)
ORGANISM   Ebola virus - Mayinga, Zaire, 1976
           Viruses; ssRNA viruses; ssRNA negative-strand viruses;
           Mononegavirales; Filoviridae; Ebolavirus.
REFERENCE  1  (bases 1 to 18959)
AUTHORS   Bukreyev,A.A., Volchkov,V.E., Blinov,V.M. and Netesov,S.V.
TITLE     The VP35 and VP40 proteins of filoviruses. Homology between Marburg
           and Ebola viruses
```

¹<http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html>

JOURNAL FEBS Lett. 322 (1), 41-46 (1993)
PUBMED 8482365
REFERENCE 2 (bases 1 to 18959)
AUTHORS Volchkov,V.E., Becker,S., Volchkova,V.A., Ternovoij,V.A.,
Kotov,A.N., Netesov,S.V. and Klenk,H.D.
TITLE GP mRNA of Ebola virus is edited by the Ebola virus polymerase and
by T7 and vaccinia virus polymerases

There were times (up to 1987) when the GenBank database was printed and published as a book.

Today, if that stack of books were to be printed it would be a few miles tall. According to the Wikipedia entry on GenBank²:

As of 15 June 2018, GenBank release 226.0 has 209,775,348 loci, 263,957,884,539 bases, from 209,775,348 reported sequences.

25.2 When do we use the GenBank format?

The GenBank format has the advantage of being a generic format that can represent a wide variety of information while still keeping this information human-readable. And that was its intended purpose.

For this same reason, however, it is not optimized for any particular data analysis task. Hence, it is used mainly to exchange information and is almost never suited as input to an analysis protocol.

We typically convert a GenBank file to some other, simpler format to work with it. ReadSeq³ is a useful conversion tool to use for this purpose.

25.3 What is RefSeq?

The NCBI Reference Sequence (RefSeq) project provides sequence records and related information for numerous organisms and provides a baseline for

²<https://en.wikipedia.org/wiki/GenBank>

³<https://sourceforge.net/projects/readseq/files/readseq/2.1.19/>

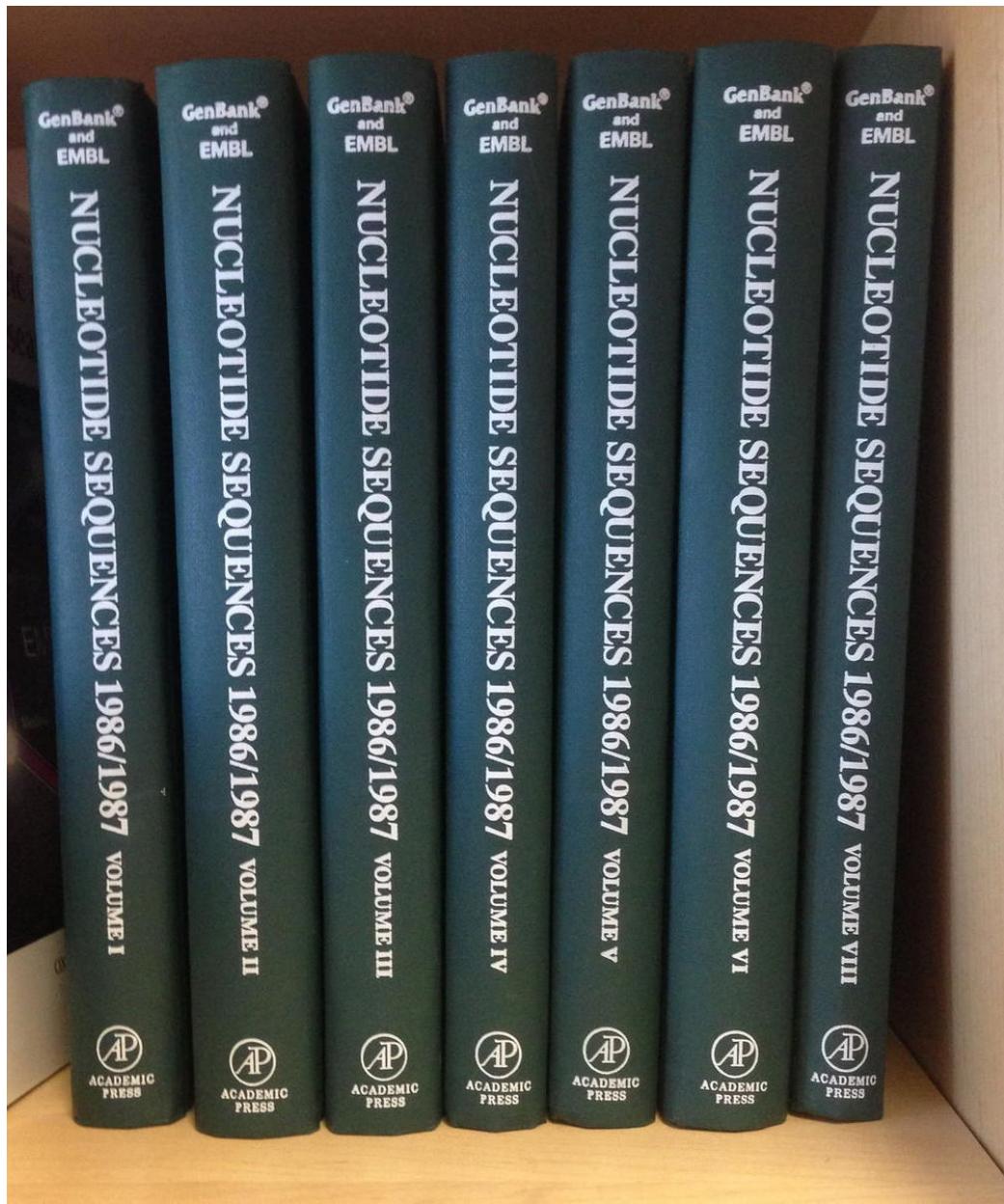
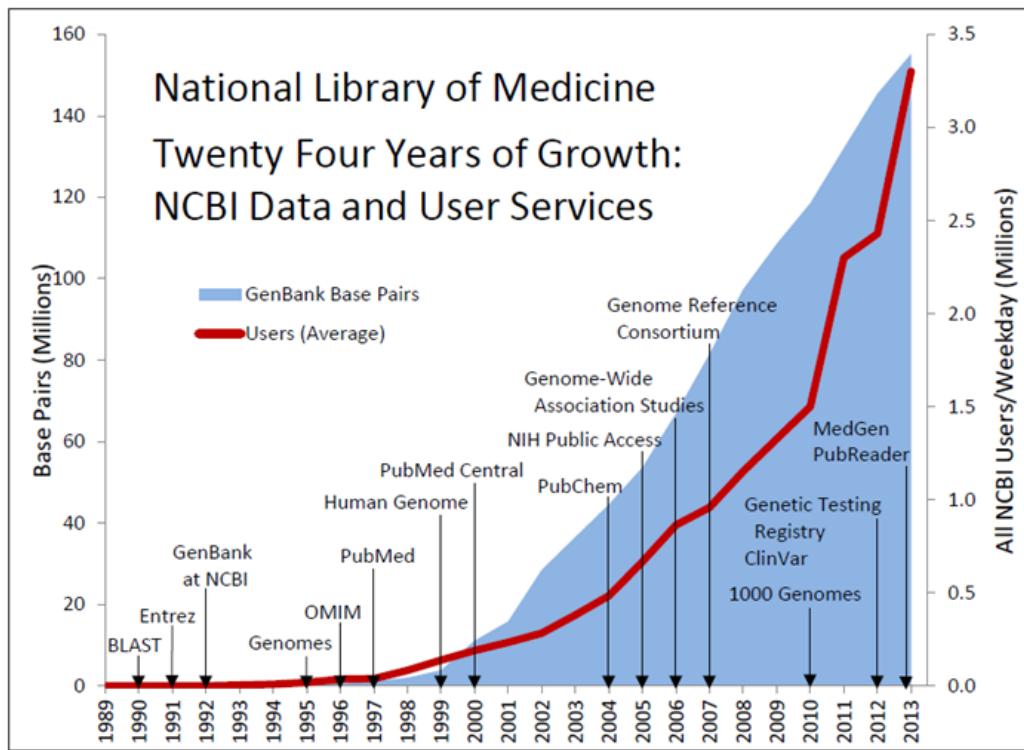


Figure 25.1: GenBank as a book

**Figure 25.2**

medical, functional, and comparative studies. The RefSeq database is a non-redundant set of reference standards derived from all the data deposited in GenBank that includes chromosomes, complete genomic molecules (organelle genomes, viruses, plasmids), intermediate assembled genomic contigs, curated genomic regions, mRNAs, RNAs, and proteins.

25.4 How are RefSeq sequences named?

The most distinguishing feature of a RefSeq record is the distinct accession number format that begins with two characters followed by an underscore (e.g., NP_).

Thus if a sequence name has an underscore _ then the sequence is part of RefSeq.

Accession prefix	Molecule type	Comment
AC_	Genomic	Complete genomic molecule, usually alternate assembly
NC_	Genomic	Complete genomic molecule, usually reference assembly
NG_	Genomic	Incomplete genomic region
NT_	Genomic	Contig or scaffold, clone-based or WGS ^a
NW_	Genomic	Contig or scaffold, primarily WGS ^a
NS_	Genomic	Environmental sequence
NZ_ ^b	Genomic	Unfinished WGS
NM_	<u>mRNA</u>	
NR_	<u>RNA</u>	
XM_ ^c	<u>mRNA</u>	Predicted model
XR_ ^c	<u>RNA</u>	Predicted model
AP_	Protein	Annotated on AC_ alternate assembly
NP_	Protein	Associated with an NM_ or NC_ accession
YP_ ^c	Protein	
XP_ ^c	Protein	Predicted model, associated with an XM_ accession
ZP_ ^c	Protein	Predicted model, annotated on NZ_ genomic records

Figure 25.3

Chapter 26

The FASTA format

26.1 What is the FASTA format?

The FASTA format is the “workhorse” of bioinformatics. It is used to represent sequence information. The format is deceptively simple:

- A `>` symbol on the FASTA header line indicates a *fasta record* start.
- A string of letters called the *sequence id* may follow the `>` symbol.
- The header line may contain an arbitrary amount of text (including spaces) on the same line.
- Subsequent lines contain the sequence.

The sequence is assumed to come from an *alphabet* that corresponds to a biological entity. For example, the standard alphabet for nucleotides would contain: ATGC. An extended alphabet may also contain the N symbol to indicate a base that could be any of ATGCN. A different extension of a nucleotide alphabet may allow extended symbols such as W that corresponds to nucleotide that is either an A or T etc. Gaps may be represented via `.` or `-` symbols. Search the web for “IUPAC nucleotides” to get a list of all such symbols that may be included in an alphabet. The same rationale needs to be followed by protein alphabets. The following is a FASTA file with two FASTA records:

```
>foo
ATGCC
>bar other optional text could go here
```

CCGTA

As it happens, the FASTA format is not “officially” defined - even though it carries the majority of data information on living systems. Its origins go back to a software tool called Fasta¹ written by David Lipman² (a scientist that later became, and still is, the director of NCBI) and William R. Pearson³ of the University of Virginia. The tool itself has (to some extent) been superceded by the BLAST suite but the format (named after the tool itself) not only survived, but has become the de facto standard.

26.2 Are there problems with this format?

The lack of a definition of the FASTA format and its apparent simplicity can be a source of some of the most confounding errors in bioinformatics. Since the format appears so exceedingly straightforward, software developers have been tacitly assuming that the properties they are accustomed to are required by some standard - whereas no such thing exists. In general, when creating FASTA files, the following rules should be observed:

- Sequence lines should not be too long. While, technically, a Fasta file that contains the sequence of the entire human chromosome 1 on a single line (330 million bases) is a valid FASTA file, most tools that run on such a file would fail in various, occasionally spectacular ways (like bringing your entire computer to a screeching halt). Making a FASTA file like that would be useful solely as a practical joke on April 1st.
- Some tools may tacitly(!) accept data containing alphabets beyond those that they know how to deal with. When your data contains alphabets beyond the 4 nucleotides or 20 aminoacid letters, you need to ensure that whatever tool you use is able to recognize the extended values.
- Use upper-case letters. Whereas both lower-case and upper-case letters are allowed by the specification, the different capitalization may carry additional meaning and some tools and methods will (tacitly again)

¹<http://fasta.bioch.virginia.edu/>

²https://en.wikipedia.org/wiki/David_J._Lipman

³<http://www.people.virginia.edu/~wrg/>

operate differently when encountering upper- or lower-case letters. Curiously enough – and curious is a word we use to avoid mentioning how we feel about it – some communities chose to designate the lower-case letters as the non-repetitive and the upper-case letters to indicate repetitive regions. Needless to say, confusion can rule the game.

- The sequence lines should always wrap at the same width (with the exception of the last line). Some tools will fail to operate correctly and may not even warn the users if this condition is not satisfied. The following is technically a valid FASTA but it may cause various subtle problems.

For example:

```
>foo
ATGCATGCATGCATGCATGC
ATGCATGCA
TGATGCATGCATGCATGCA
```

should be reformatted to:

```
>foo
ATGCATGCATGCATGCATGC
ATGCATGCATGATGCATGCA
TGCATGCA
```

26.3 Is there more information in FASTA headers?

Some data repositories will format FASTA headers to include structured information. Tools may operate differently when this information is present in the FASTA header. For example, NCBI may include both gi and gb accession numbers >gi|10141003|gb|AF086833.2|.

Specifically the Blast tool from NCBI is able to query data in more granular fashion when this extra information embedded via header formatting is available. Below is a list of the recognized FASTA header formats.

Type	Format(s) ¹	Example(s)
local	lcl integer lcl string	lcl 123 lcl hmm271
GenInfo backbone seqid	bbs integer	bbs 123
GenInfo backbone moltype	bbm integer	bbm 123
GenInfo import ID	gim integer	gim 123
GenBank	gb accession locus	gb M73307 AGMA13GT
EMBL	emb accession locus	emb CAM43271.1
PIR	pir accession name	pir G36364
SWISS-PROT	sp accession name	sp P01013 OVAX_CHICK
patent	pat country patent sequence	pat US RE33188 1
pre-grant patent	pgp country application-number seq-number	pgp EP 0238993 7
RefSeq ²	ref accession name	ref NM_010450.1
general database reference	gnl database integer gnl database string	gnl taxon 9606 gnl PID e1632
GenInfo integrated database	gi integer	gi 21434723
DDBJ	dbj accession locus	dbj BAC85684.1
PRF	prf accession name	prf 0806162C
PDB	pdb entry chain	pdb 1I4L D
third-party GenBank	tpg accession name	tpg BK003456
third-party EMBL	tpe accession name	tpe BN000123
third-party DDBJ	tpd accession name	tpd FAA00017

Figure 26.1

26.4 Is there more information in the FASTA sequences?

Lower-case letters may be used to indicate repetitive regions for the genome for example ATGCATGCagctagctATGTATGC. In this case agctagct is indicated as being present in multiple location of the larger genome. That being said, it is typically not easy to find out what exactly the “repetitiveness” consists of and how it was determined. Usually, it is the case that the “repetitiveness” in a sequence has been determined by a tool run with certain parameters and that has marked the sequence as such.

Importantly, some tools like `lastz` will, by default, skip over regions that are lower-cased - this can cause lots of confusion if not recognized right away.

26.5 Where do I get a fasta file?

We will cover and example data access in more detail, for now if you knew the accession number of a sequence deposited at NCBI you can obtain it with:

```
efetch -db nucore -id NM_000020 -format fasta | head
```

will produce

```
>NM_000020.2 Homo sapiens activin A receptor like type 1 (ACVRL1), transcript variant 1, mRNA
AGGAAACGGTTTATTAGGAGGGAGTGGTGGAGCTGGGCCAGGCAGGAAGACGCTGGAATAAGAAACATTT
TTGCTCCAGCCCCATCCCAGTCCCAGGGCTGCCCGCAGCTGCCCGAGCGAGCCCTCCCCGGCT
CCAGCCCGTCCGGGCGCCGGACCCCAGCCCGCTCCAGCGCTGGCGGTGCAACTGCCGCCGCG
CGGTGGAGGGAGGTGGCCCCGGTCCGCCGAAGGCTAGCGCCCCGCCACCCGAGAGCGGGCCAGAGGG
ACCATGACCTTGGCTCCCCAGGAAAGGCCTCTGATGCTGCTGATGGCCTTGGTACCCAGGGAGACC
CTGTGAAGCCGTCTCGGGGCCGCTGGTACCTGCACGTGAGAGCCCACATTGCAAGGGCCTACCTG
CCGGGGGGCCTGGTCACAGTAGTGCTGGTGCAGGGAGGGAGGGAGGCACCCCCAGGAACATCGGGCTGC
GGGAACCTGCACAGGGAGCTCTGCAGGGGGCGCCCCACCGAGTTCGTCAACCAACTACTGCTGCGACAGCC
ACCTCTGCAACCACAACGTGCTCCCTGGTGGAGGCCACCCAACCTCTGGAGCAGCCGGAACAGA
```

Chapter 27

The FASTQ format

27.1 What is the FASTQ format?

The FASTQ format is the de facto standard by which all sequencing instruments represent data. It may be thought of as a variant of the FASTA format that allows it to associate a quality measure to each sequence base, FASTA with QUALITIES.

In simpler terms, it is a format where for every base, we associate a reliability measure: base is A and the probability that we are wrong is 1/1000. Conceptually, the format is very similar to FASTA but suffers from even more flaws than the FASTA format.

The FASTQ format¹ consists of 4 sections (and these are usually produced a single line each):

1. A FASTA-like header, but instead of the > symbol it uses the @ symbol. This is followed by an ID and more optional text, similar to the FASTA headers.
2. The second section contains the measured sequence (typically on a single line), but it may be wrapped until the + sign starts the next section.
3. The third section is marked by the + sign and may be optionally followed by the same sequence id and header as the first section

¹https://en.wikipedia.org/wiki/FASTQ_format

4. The last line encodes the quality values for the sequence in section 2, and must be of the same length as section 2. It should (must?) also be wrapped the same way as the section 2.

An example of a single FASTQ record as seen in the Wikipedia entry:

```
@SEQ_ID
GATTTGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTGTTCAACTCACAGTTT
+
! ''*((((*++))%%%++)(%%%%) .1***-+* '')**55CCF>>>>CCCCCCC65
```

The weird characters in the 4th section (line) are the so called “encoded” numerical values. In a nutshell, each character `! ''*(((` represents a numerical value: a so-called Phred score², encoded via a single letter encoding. It is, frankly, a overly convoluted way of doing something that should be very simple. Perhaps a cautionary tale of what happens when scientists start to design data formats.

The idea behind the Phred score is to map two digit numbers that represent error likelihoods to single characters so that the length of the quality string stays the same as the length of the sequence.

Each character has a numerical value, say `!` will be mapped to 0, `+` will be remapped to mean 10, `5` will be mapped to 30 and `I` will be mapped to 40. In most situations we don’t need to remember the mapping, and we just take the remappings “as they are” below. The scale of characters to value mappings looks like this:

<code>! "#\$%&' ()*+, -./0123456789: ;<=>?@ABCDEFGHI</code>	<code> </code>
<code>0....5....10....15....20....25....30....35....40</code>	<code> </code>
<code>worst.....best</code>	

The quality values of the FASTQ files are on top. The numbers in the middle of the scale from 0 to 40 are called Phred scores. The numbers represent the error probabilities in a somewhat convoluted manner via the formula `Error=10^(-P/10)` basically summarized as:

- $P=0$ means $1/1$ (100% error)
- $P=10$ means $1/10$ (10% error)

²https://en.wikipedia.org/wiki/Phred_quality_score

- P=20 means 1/100 (1% error)
- P=30 means 1/1000 (0.1% error)
- P=40 means 1/10000 (0.01% error)

This if A has the quality of * we can see that * corresponds to the number 9 that in turn via the formula $10^{(-9/10)}$ is around 0.12 that is 12%. If the quality were E, the value is 36 and then we would have $10^{(-36/10)}=0.0002$ that is 0.02%

27.2 How to recognize FASTQ qualities by eye

When qualities look like “swearing” in a comic book !#\$%&'()*+,-. it means that the data is of low quality. Imagine the sequencing instrument telling you: *Hey, look at all this #!#!# data!*

When the quality scores are jumbled but readable letters your data has great quality. Imagine the sequencing instrument telling you: *Hey, look at all this ABCDE data!*

27.3 Are there different versions of the FASTQ encoding?

Unfortunately, yes. There was a time when instrumentation makers could not decide at what character to start the scale. The current standard shown above is the so-called Sanger (+33) format where the ASCII codes are shifted by 33. There is the so-called +64 format that starts close to where the other scale ends. In this scaling @ is mapped to 0 and i is mapped to 40 like so:

```
@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
|   |   |   |   |   |   |   |   |
0....5...10...15...20...25...30...35...40
|   |   |   |   |   |   |   |
worst.....best
```

Most confusingly, in this encoding the characters that indicate the highest base quality in Sanger will indicate one of the lowest qualities. It is easy to

recognize this, though. When our FASTQ quality contains lower case letters `abcdefghijklm` it means that your data is in this older format. Some tools will allow you to set parameters to account for this different encoding. For others, we will need to convert the data to the correct encoding. You can use `seqtk` for example:

```
seqtk seq -Q64 input.fq > output.fa
```

27.4 Is there more information in FASTQ headers?

Just as with FASTA headers, information is often encoded in the “free” text section of a FASTQ file. For example (see the same Wikipedia entry on FASTQ formats):

```
@EAS139:136:FC706VJ:2:2104:15343:197393 1:Y:18:ATCACG
```

Contains the following information:

- EAS139 the unique instrument name
- 136 the run id
- FC706VJ the flowcell id
- 2 flowcell lane
- 2104 tile number within the flowcell lane
- 15343 ‘x’-coordinate of the cluster within the tile
- 197393 ‘y’-coordinate of the cluster within the tile
- 1 the member of a pair, 1 or 2 (paired-end or mate-pair reads only)
- Y Y if the read is filtered, N otherwise
- 18 0 when none of the control bits are on, otherwise it is an even number
- ATCACG index sequence

This information is specific to a particular instrument/vendor and may change with different versions or releases of that instrument.

This information can be useful, however, to identify quality control problems or systematic technical issues that might affect the instruments.

27.5 What is a Phred score?

A Phred score³ Q is used to compute the probability of a base call being incorrect by the formula $P=10^{(-Q/10)}$.

You don't need to remember the formula - there is a much simpler way to remember it; like so (where the multiplier to 10 tells you the order of magnitude):

Q	Error	Accuracy
0	1 in 1	0%
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10,000	99.99%

27.6 How do I convert FASTQ quality codes at the command line?

When it comes to visually evaluating and inspecting data we rarely need to find out precisely what a value actually decodes to. The simplest is to remember that:

- !"#\$%&'()*+,-. means low quality 1/10
- -./0123456789 means medium quality 1/100
- ABCDEFGHI means high quality 1/1000

If you need to convert exactly, you may use the following command line shortcuts:

Find the integer value that corresponds to character A (prints 65):

```
python -c 'print (ord("A"))'
```

Find character that corresponds to value 65 (prints A):

```
python -c 'print (chr(65))'
```

Find Phred quality score by the Sanger encoding of letter A (prints 32)"

```
python -c 'print (ord("A")-33)'
```

³https://en.wikipedia.org/wiki/Phred_quality_score

Find Sanger encoded letter that corresponds to Phred quality score 32 (prints A):

```
python -c 'print (chr(32+33))'
```

Compute the error probability represented by the Pred quality score 32 (prints 0.00063095734448):

```
python -c 'from math import*; print (10**-(32/10.0))'
```

Compute the error probability represented by letter A (prints 0.00063095734448):

```
python -c 'from math import*; print (10**-((ord("A")-33)/10.0))'
```

Compute the Phred quality score that corresponds to probability p=0.00063095734448 (prints 32)

```
python -c 'from math import*; print (int(round(-10*log(0.00063095734448)/log(10))))'
```

Compute the Sanger encoding that corresponds to probability p=0.00063095734448 (prints A)

```
python -c 'from math import*; print (chr(int(round(-10*log(0.00063095734448)/log(10))+3)))'
```

Due to the imprecision of the floating point representation of numbers, a rounding of the results is required before converting to integer. The extra call to `int` is required only when using Python 2. When using Python 3 `round` alone will do.

27.7 Closing thoughts on FASTQ

We have to recognize that most bioinformatics data formats are adopted via a mixture of past consensus and historical precedent. Most formats may appear to be good enough when first proposed, yet the fast advancements may render them quickly inadequate. The lack of central authority means that even inefficient formats endure past their prime.

Notably the FASTQ format was originally designed as a multi-line format just as the FASTA format. In the early days of high-throughput sequencing, instruments always produced the entire FASTQ sequence on a single line (on account of the sequences being so short 35-50bp).

A surprising number of bioinformaticians still operate under the assumption that the FASTQ format is line-oriented and that the entire sequence must

always be on a single line, and that the FASTQ record consists of 4 lines. In addition, a large number of “guides” and “tutorials” will tacitly assume this and will show you code that operates on a FASTQ file in a line by line basis.

While having sequences on a single line greatly simplifies operations, do remember that FASTQ format is not required to be line-oriented! As sequencing technology evolves and the sequencing reads become longer, stuffing the entire sequence on a single line will start to become more inappropriate. For now this problem is yet acute, but it may get there.

Chapter 28

Advanced FASTQ processing

Note: This chapter presents advanced topics that may not be immediately useful for beginners. Feel free to skip it if you are new to bioinformatics. We have added this chapter right after the FASTQ format to make it locating it easier later on, when you will need the information in it. Eventually, most bioinformaticians run into challenges that will require the techniques that are presented below.

Author: **Wei Shen**¹

This page illustrates advanced FASTA/Q manipulations using SeqKit². For more complete documentation visit:

- Code: <https://github.com/shenwei356/seqkit>
- Manual: <https://bioinf.shenwei.me/seqkit/usage/>

Some other utilities, including csvtk³ (CSV/TSV toolkit) and shell commands were also used.

Note: SeqKit seamlessly supports FASTA and FASTQ formats both in their original form as well as gzipped compressed format. We list FASTA or FASTQ depending on the more common usage but you can always use it

¹<http://shenwei.me/>

²<https://bioinf.shenwei.me/seqkit/>

³<https://bioinf.shenwei.me/csvtk/>

on the other type as well.

If you do not have the tools installed yet you can do so by visiting the release page

- <https://github.com/shenwei356/csvtk/releases>

then downloading and copying the executable to your `~/bin` directory. We do not recommend installing with `conda` as at this time the conda dependencies are not properly set and may affect the use of other tools.

For example on MacOS you could do a:

```
URL1=https://github.com/shenwei356/seqkit/releases/download/v0.10.0/seqkit_darwin_amd64
(cd ~/bin && curl -L $URL1 | tar zxv -)
chmod +x ~/bin/seqkit

URL2=https://github.com/shenwei356/csvtk/releases/download/v0.17.0/csvtk_darwin_amd64
(cd ~/bin && curl -L $URL2 | tar zxv -)
chmod +x ~/bin/csvtk
```

28.1 Example data

One FASTQ file (sample reads, 1M) and two FASTA files (Virus DNA and protein sequences from NCBI RefSeq database, 60+40M) are used.

```
wget -nc http://data.biostarhandbook.com/reads/duplicated-reads.fq.gz
wget -nc ftp://ftp.ncbi.nih.gov/refseq/release/viral/viral.2.1.genomic.fna.gz
wget -nc ftp://ftp.ncbi.nih.gov/refseq/release/viral/viral.2.protein.faa.gz
```

As the data may change in time, the output of the commands may change accordingly.

28.2 How to produce an overview of FASTQ files?

Sequence format and type are automatically detected.

```
seqkit stat *.gz
```

prints:

```
file          format type  num_seqs  sum_len min_len avg_len max_len
viral.2.1.genomic.fna.gz  FASTA  DNA      7,942  212,999,370    200  26,819.4  2,243,109
viral.2.protein.faa.gz   FASTA  Protein  256,686  64,293,716     8    250.5    8,573
```

28.3 How do I get the GC content?

`seqkit fx2tab` converts FASTA/Q to 3-column tabular format (1st: name/ID, 2nd: sequence, 3rd: quality), and can also provide various information in new columns, including sequence length, GC content/GC skew, alphabet.

GC content:

```
seqkit fx2tab -name -only-id -gc viral*.fna.gz | head
```

generates:

NC_029548.1	45.44
NC_029549.1	31.66
NC_029550.1	30.12
NC_029551.1	31.09
NC_029552.1	30.73
...	

28.4 How to I get the percentage for custom bases?

Suppose we wish to find the percentage for A, C and A+C:

```
seqkit fx2tab -H -n -i -B a -B c -B ac viral.2.1.genomic.fna.gz | head -5
```

produces:

#name	seq	qual	a	c	ac
NC_029548.1			25.65	13.71	39.35
NC_029549.1			31.75	16.24	47.99
NC_029550.1			30.12	16.93	47.05
NC_029551.1			27.05	17.00	44.05

28.5 How to extract a subset of sequences with name/ID list file?

This is a frequently used manipulation. Let's create a sample ID list file, which may also come from some other method like a mapping result.

```
seqkit sample --proportion 0.001 duplicated-reads.fq.gz | seqkit seq --name --on
ID list file:
```

```
head id.txt
```

shows:

```
SRR1972739.2996
SRR1972739.3044
SRR1972739.3562
```

Searching by ID list file:

```
seqkit grep --pattern-file id.txt duplicated-reads.fq.gz > duplicated-reads.subs
```

28.6 How do I find FASTA/Q sequences containing degenerate bases and locate them?

`seqkit fx2tab` converts FASTA/Q to tabular format and can output the sequence alphabet in a new column. Text searching tools can then be used to filter the table.

```
seqkit fx2tab -n -i -a viral.2.1.genomic.fna.gz | csvtk -H -t grep -f 4 -r -i -p "[
prints:
```

gi 446730228 ref NC_019782.1	ACGNT
gi 557940284 ref NC_022800.1	ACGKT
gi 564292828 ref NC_023009.1	ACGNT

Long-option version of the command:

```
seqkit fx2tab --name --only-id --alphabet viral.1.1.genomic.fna.gz | csvtk --no-
```

You can then exclude these sequences with `seqkit grep`:

```
# save the sequence IDs.
seqkit fx2tab -n -i -a viral.2.1.genomic.fna.gz | csvtk -H -t grep -f 4 -r -i -p "[^ACGT]"

# search and exclude.
seqkit grep --pattern-file id2.txt --invert-match viral.1.1.genomic.fna.gz > clean.fa
```

Or locate the degenerate bases, e.g., N and K

```
seqkit grep --pattern-file id2.txt viral.2.1.genomic.fna.gz | seqkit locate --ignore-case
```

shows:

28.7 How do I remove FASTA/Q records with duplicated sequences?

```
seqkit rmdup --by-seq --ignore-case duplicated-reads.fq.gz > duplicated-reads.uniq.fq.gz
```

If the FASTA/Q file is very large, you can use the flag `-m`/`--md5`, which uses MD5 instead of the original sequences to reduce memory usage during sequence comparison.

You can also deduplicate according to sequence ID (default) or full name (`--by-name`).

28.8 How do I locate motif/subsequence/enzyme digest sites in FASTA/Q sequence?

Related posts: Question: Count and location of strings in fastq file reads⁴, Question: Finding TATAWAA in sequence⁵.

Assuming you have a list of motifs (enzyme digest sites) in FASTA format that you would like to locate in your sequences:

```
cat enzymes.fa
```

```
>EcoRI
GAATTC
>MmeI
TCCRAC
>SacI
GAGCTC
>XcmI
CCANNNNNNNNTGG
```

Flag `--degenerate` is on because our patterns of interest contain degenerate bases. Command:

```
seqkit locate --degenerate --ignore-case --pattern-file enzymes.fa viral.2.1.genome
```

Sample output (simplified and reformatted by `csvtk -t uniq -f 3 | csvtk -t pretty`)

seqID	patternName	pattern	strand	start	end	matches
gi 526245010 ref NC_021865.1	MmeI	TCCRAC	+	1816	1821	TCC
gi 526245010 ref NC_021865.1	SacI	GAGCTC	+	19506	19511	GA
gi 526245010 ref NC_021865.1	XcmI	CCANNNNNNNNTGG	+	2221	2235	

28.9 How do I sort a huge number of FASTA sequences by length?

Sorting a FASTA file in order of sequence size (small to large).

⁴<https://www.biostars.org/p/204658/>

⁵<https://www.biostars.org/p/221325/>

28.10. HOW DO I SPLIT FASTA SEQUENCES ACCORDING TO INFORMATION IN THE HEADER?

```
seqkit sort --by-length viral.2.1.genomic.fna.gz > viral.1.1.genomic.sorted.fa
```

If the files are too big, use flag `--two-pass` which will consume less memory.

```
seqkit sort --by-length --two-pass viral.2.1.genomic.fna.gz > viral.2.1.genomic.sorted.fa
```

You can also sort by sequence ID (default), full header (`--by-name`) or sequence content (`--by-seq`).

28.10 How do I split FASTA sequences according to information in the header?

Related posts: Question: extract same all similar sequences in FASTA based on the header⁶.

For example, the FASTA header line of `viral.2.protein.faa.gz` contains the species name in square brackets.

Overview of FASTA Headers:

```
seqkit head -n 3 viral.2.protein.faa.gz | seqkit seq --name
```

generates:

```
gi|526245011|ref|YP_008320337.1| terminase small subunit [Paenibacillus phage phiIBB_Pl1]
gi|526245012|ref|YP_008320338.1| terminase large subunit [Paenibacillus phage phiIBB_Pl1]
gi|526245013|ref|YP_008320339.1| portal protein [Paenibacillus phage phiIBB_Pl23]
```

`seqkit split` can split FASTA/Q files according to ID, number of parts, size of each part, or sequence region. In this case, we'll split according to sequence ID (species name) which can be specified by flag `--id-regexp`.

Default ID:

```
seqkit head -n 3 viral.2.protein.faa.gz | seqkit seq --name --only-id
```

outputs:

```
gi|526245011|ref|YP_008320337.1|
gi|526245012|ref|YP_008320338.1|
gi|526245013|ref|YP_008320339.1|
```

New ID:

⁶<https://www.biostars.org/p/223937/>

```
seqkit head -n 3 viral.2.protein.faa.gz | seqkit seq --name --only-id --id-regex
prints:
```

```
Paenibacillus phage phiIBB_P123
Paenibacillus phage phiIBB_P123
Paenibacillus phage phiIBB_P123
```

Split:

```
seqkit split --by-id --id-regex "\[(.+)\" viral.1.protein.faa.gz
```

28.11 How do I search and replace within a FASTA header using character strings from a text file?

Related posts: Question: Replace names in FASTA file with a known character string from a text file⁷, Question: Fasta header, search and replace...?⁸.

`seqKit replace` can find substrings in FASTA/Q headers using regular expressions and replace them with strings or corresponding values of found substrings provided by a tab-delimited key-value file.

For example, to unify names of protein with unknown functions, we want to rename “hypothetical” to “putative” in lower case. The replacing rules are listed below in tab-delimited file:

```
cat changes.tsv
Hypothetical    putative
hypothetical    putative
Putative        putative
```

Overview the FASTA headers containing “hypothetical”:

```
seqkit grep --by-name --use-regex --ignore-case --pattern hypothetical viral.1.
gi|526245016|ref|YP_008320342.1| hypothetical protein IBBP123_06 [Paenibacillus
gi|526245019|ref|YP_008320345.1| hypothetical protein IBBP123_09 [Paenibacillus
gi|526245020|ref|YP_008320346.1| hypothetical protein IBBP123_10 [Paenibacillus
```

⁷<https://www.biostars.org/p/221962/>

⁸<https://www.biostars.org/p/205044/>

A regular expression, `^([^\s]+)(\w+)`, was used to specify the key to be replaced, which is the first word after the sequence ID in this case. Note that we also capture the ID (`^([^\s]+)`) so we can restore it in “replacement” with the capture variable `${1}` (more robust than `$1`). And flag `-I/--key-capt-idx` (default: 1) is set to 2 because the key (`\w+`) is the second captured match. Command:

```
seqkit replace --kv-file changes.tsv --pattern "^( [^\s]+ )(\w+)" --replacement "\${1}{kv}"
```

28.12 How do I extract paired reads from two paired-end reads files?

Let’s create two unbalanced PE reads files:

```
seqkit rmdup duplicated-reads.fq.gz | seqkit replace --pattern ".+" --replacement "1" |
```

```
seqkit rmdup duplicated-reads.fq.gz | seqkit replace --pattern ".+" --replacement "2" |
```

Overview:

```
# number of records
seqkit stat read_1.fq.gz read_2.fq.gz
```

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
read_1.fq.gz	FASTQ	DNA	9,033	912,333	101	101	101
read_2.fq.gz	FASTQ	DNA	8,965	905,465	101	101	101

```
# sequence headers
seqkit head -n 3 read_1.fq.gz | seqkit seq --name
```

```
SRR1972739.1 1
SRR1972739.3 1
SRR1972739.4 1
```

```
seqkit head -n 3 read_2.fq.gz | seqkit seq --name
```

```
SRR1972739.1 2
SRR1972739.2 2
SRR1972739.3 2
```

Firstly, extract sequence IDs of the two file and compute the intersection:

```
seqkit seq --name --only-id read_1.fq.gz read_2.fq.gz | sort | uniq -d > id.txt

# number of IDs
wc -l id.txt

8090 id.txt
```

Then extract reads using `id.txt`:

```
seqkit grep --pattern-file id.txt read_1.fq.gz -o read_1.f.fq.gz
seqkit grep --pattern-file id.txt read_2.fq.gz -o read_2.f.fq.gz
```

Check if the IDs in two files are the same by `md5sum`:

```
seqkit seq --name --only-id read_1.f.fq.gz > read_1.f.fq.gz.id.txt
seqkit seq --name --only-id read_2.f.fq.gz > read_2.f.fq.gz.id.txt

md5sum read_*.f.fq.gz.id.txt
```

```
537c57cfdc3923bb94a3dc31a0c3b02a  read_1.f.fq.gz.id.txt
537c57cfdc3923bb94a3dc31a0c3b02a  read_2.f.fq.gz.id.txt
```

Note that this example assumes that the IDs in the two reads file have same order. If not, you can sort them after previous steps. GNU `sort` can sort large file using disk, so temporary directory is set as current directory by option `-T ..`

```
gzip -d -c read_1.f.fq.gz | seqkit fx2tab | sort -k1,1 -T . | seqkit tab2fx | gzip -
gzip -d -c read_2.f.fq.gz | seqkit fx2tab | sort -k1,1 -T . | seqkit tab2fx | gzip -
```

28.13 How to concatenate two FASTA sequences in to one?

Related posts: Combining two fasta sequences into one⁹

Data (not in same order):

```
cat 1.fa
```

⁹<https://www.biostars.org/p/231806/>

28.13. HOW TO CONCATENATE TWO FASTA SEQUENCES IN TO ONE?263

```
>seq1
aaaaaa
>seq2
ccccc
>seq3
ggggg

cat 2.fa
>seq3
TTTT
>seq2
GGGGG
>seq1
CCCCC
```

Step 1. Convert FASTA to tab-delimited (3 columns, the 3rd column is blank (no quality for FASTA)) file:

```
seqkit fx2tab 1.fa > 1.fa.tsv
seqkit fx2tab 2.fa > 2.fa.tsv
```

```
cat -A 1.fa.tsv
seq1^Iaaaaaa^I$
seq2^Iccccc^I$
seq3^Iggggg^I$
```

Step 2. Merge two table files:

```
csvtk join -H -t 1.fa.tsv 2.fa.tsv | cat -A
seq1^Iaaaaaa^I^ICCCCC^I$
seq2^Iccccc^I^IGGGGG^I$
seq3^Iggggg^I^TTTTT^I$
```

Step 3. Note that there are two TAB between the two sequences, so we can remove them to join the sequences

```
csvtk join -H -t 1.fa.tsv 2.fa.tsv | sed 's/\t\t//'
seq1      aaaaaaCCCCC
seq2      cccccGGGGG
seq3      gggggTTTTT
```

Step 4. Convert tab-delimited file back to FASTA file:

```
csvtk join -H -t 1.fa.tsv 2.fa.tsv | sed 's/\t\t//' | seqkit tab2fx  
>seq1  
aaaaaCCCCC  
>seq2  
cccccGGGGG  
>seq3  
gggggTTTTT
```

All in one command:

```
csvtk join -H -t <(seqkit fx2tab 1.fa) <(seqkit fx2tab 2.fa) | sed 's/\t\t//' | seqkit tab2fx
```

Part VI

VISUALIZING DATA

Chapter 29

Visualizing biological data

For all the algorithmic power at our disposal, visually inspecting data has been and will always be an essential ingredient to scientific discovery. The human eye has pattern recognition abilities that are unmatched by any computer. Our minds are the most potent analytic instrument that we have access to. We are just not fast enough, and we're not physically capable of processing all the data at our disposal. The limitation is imposed by the quantity of data.

Trust your own abilities, your eye and mind are able to analyze data far better than any software. You will see that eventually, you'll disagree with a computed result – when that happens, almost always it means that you are correct and the computer/software is wrong.

29.1 What are the challenges of visualization?

It is essential to recognize that visualizing data so that it displays the attributes that are interesting to us at any given moment is a far more complicated task than we may realize. A meaningful visualization is one that shows us *only* what we are interested in at any given moment – but that need changes over time.

For this reason, building useful visualization software is far more challenging

than it may seem at first – and it is more difficult than building tools that have objective measures (speed/accuracy etc.) of quality.

No wonder that biological visualization tools have a “graveyard” that is larger than that for most other software. Many, including yours truly¹, have tried to build a better, simpler and nicer visualization approach, and many, including yours truly, have failed to deliver a long term solution.

29.2 What is a genome browser?

A genome browser is a graphical interface to display information from a database of genomic data. Most genome browsers draw linear tracks that represent the forward strand of the genome from its 5' (left) towards the 3' (right) direction. The genomic features drawn over this linear track are called **glyphs**.

A glyph is a pictogram that corresponds to a particular genomic feature.

No universal standard specifies all visualization tools. Interpreting glyphs comes with experience and a level of familiarity with each tool. For example, here is a default screenshot of the UCSC genome browser:

29.3 Why are default browser screens so complicated?

Visit any genome online genome browser site, and you may be overwhelmed by the information that it shows by default. As an example click here to see a default UCSC Genome Browser view².

Note how most of this information will most likely not be relevant to you. Unfortunately, scientists are not always that great at creating practical and straightforward services, and many operate under the mistaken assumption that “more” must be “better.”

Typically, there is nothing special about the data that a browser shows you by default. In the tracks above some features of the elephant genome are shown,

¹<https://academic.oup.com/bioinformatics/article/24/10/1305/178363>

²<https://genome.ucsc.edu/cgi-bin/hgTracks>

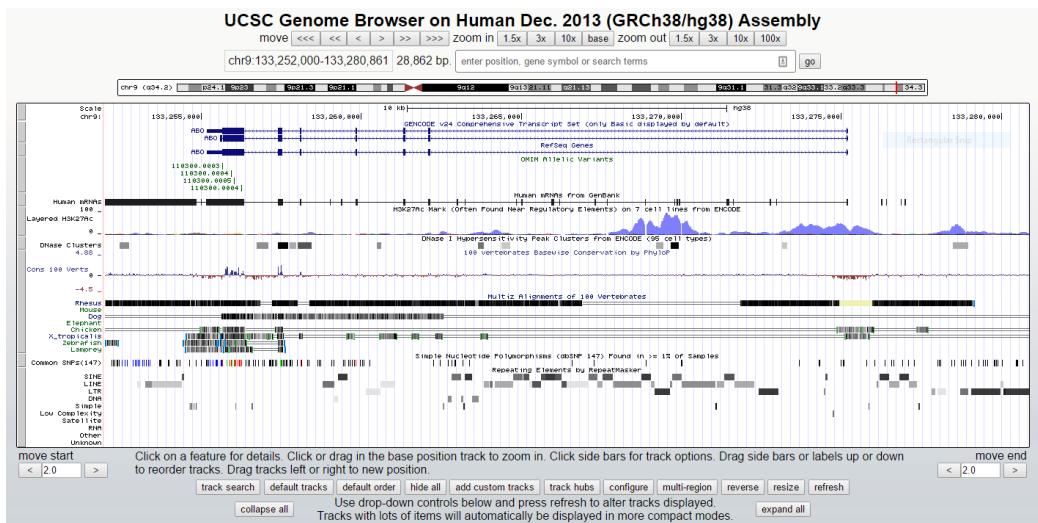


Figure 29.1

together with a so-called “Layered H2K27AC.” track without us having ever asked to see it. I mean no disrespect to you, elephant, I am not interested in seeing your track, nor am I interested in layered H2K27AC, whatever that is - some histone with modifications. Perhaps it is the lead developer’s (or their boss’ boss’) personal data, so they want to show it off. Their fallacy is to operate under the assumption that everyone must want to see the same.

Needless to say – this arbitrary, initial complexity of most browsers makes their use more a lot more difficult than necessary. It often takes inordinate amounts of time to find and visualize what you want.

29.4 What types of data may be visualized in a genome browser?

In most cases, only data in simple line-oriented formats work: FASTA, BED, GFF, SAM/BAM.

Data formats that contain more complex information such as GenBank or EMBL typically need to be transformed and simplified into a line-oriented format such as BED or GFF.

Most visualization tools come with some data pre-loaded, and with different genomic builds for widely used organisms: human, mouse, fruit fly, etc. That means that data for many genomic features will likely already be present. For other organisms, we may need to build our custom genome and add our custom annotations to the browser.

29.5 How do I interpret glyphs?

Glyphs are a visual representation of some genomic features:

- Horizontal intervals: directions, genes, alignments
- Values over intervals: coverages, probabilities
- Attributes at locations: mutations, deletions, junctions

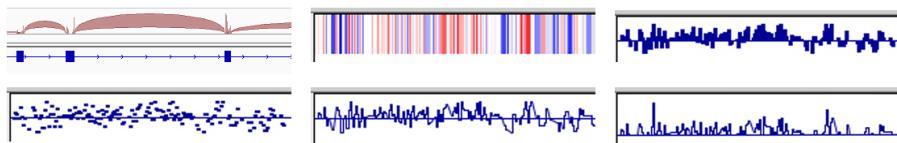


Figure 29.2

What you will find is that often it is not possible to locate a precise definition for the reasons that a feature is drawn a specific way.

For example, why is one linear feature drawn thicker than the other? Probably because it is labeled differently. Perhaps it is a coding sequence, and this tool chose to draw those differently. Alas, this information may not always be present or easy to find in the tool's documentation. Often you need to rely on intuition, some detective work, and your interpretation.

The take-home message here is that behind every visual feature that you see, no matter how small, it is the visualizer trying to tell you something more. For example, is an alignment outline of an ever-so-slightly different color? Perhaps it is a chimeric read, perhaps it is a multi-mapped read, maybe it is duplicated - and this is how this visualizer shows these - try to find out what the visualizer is trying to tell you

29.6 Which standalone genome browsers should I use?

A surprising number of standalone genome visualization tools become abandoned within a few years of their announcement.

We list those standalone applications that have withstood the test of time and appear to have decent institutional support:

- IGView³, Integrative Genomics Viewer³ by the Broad Institute.
- IGB, Integrated Genome Browser⁴ from the University of North Carolina, Charlotte
- Artemis⁵ by the Wellcome Trust Sanger Institute. One of the few browsers that can handle GenBank and EMBL formats!

29.7 What about online genome browsers?

- UCSC Genome/Table Browser⁶ from the University of California at Santa Cruz,
- Ensembl Genome Browser⁷ from the partnership of the European Bioinformatics Institute and the Wellcome Trust Sanger Institute

For each of these services, it is possible to display your data within the context of the data that is stored within that service. See next chapters for more detail.

It should be noted here, while NCBI offers many facilities to visualize data that is already deposited into the service, it offers few options to visualize our data within the same context.

³<http://software.broadinstitute.org/software/igv/>

⁴<http://bioviz.org/igb/index.html>

⁵<http://www.sanger.ac.uk/science/tools/artemis>

⁶<https://genome.ucsc.edu/>

⁷<http://useast.ensembl.org/index.html>

Chapter 30

Using the Integrative Genomics Viewer

NOTE: The usage instructions for IGV and all graphical programs are kept short intentionally.

In our opinion, when presenting graphical user interfaces, describing steps in their annoying minutiae (click here, then click there) have questionable effectiveness. Step by step instructions that need to be followed blindly take away your ability to *understand* what is going on.

Different people may begin from different starting points, and they may get stuck at various stages. To succeed you will need a conceptual understanding of what can be accomplished and what outcomes will look like, rather than following a click-by-click “handholding.”

What we will state in this book are possible actions, and we will discuss many of potential pitfalls. But you, yourself will have to explore and use the interface to learn and successfully perform the operations.

Take an active approach:

- Explore the menu
- Right-click on glyphs, windows, and tabs.
- Try to drag and drop tracks or features.

30.1 What is IGV?

IGV, Integrative Genomics Viewer¹ by the Broad Institute is both a visualizer and a data integration platform where your data can be displayed in the context of other information.

30.2 How to run IGV on Mac and Linux?

You can download and double click on IGV – it is simple to run it that way and initially you might use it that way.

In our experience though, this is not the more optimal way to run it. When working with large datasets, IGV may get bogged down. The software may hang, and its error recovery is impacted when run in the “double-click” option. We have had much better luck downloading and running the source code distribution of IGV. For example after unpacking it into `src` we can run the shell script from the command line:

```
bash ~/src/IGV_2.3.85/igv.sh
```

Then minimize this window and come back to it only if there is trouble. You will be able to see in this window when trouble is brewing, and you’ll be able to exit IGV more readily if it hangs/breaks.

30.3 How to run IGV on Windows Bash?

Under windows Bash, you will need to run IGV under as a Windows application! Download the Windows version, double click it to start. When you need to load data navigate via Windows to the Unix folder of interest. IGV was one reason why, during setup, we wanted you to locate where your Unix files are when accessing them from Windows.

¹<http://software.broadinstitute.org/software/igv/>

30.4 What does the IGV interface look like?

Most visualization tools (and IGV is no different) come with data preloaded, with different genomic builds of widely used organisms: human, mouse, fruit-fly, etc. That means that some information on your organism may already be present. For all other organisms, we typically build our custom genome and add our custom annotations.

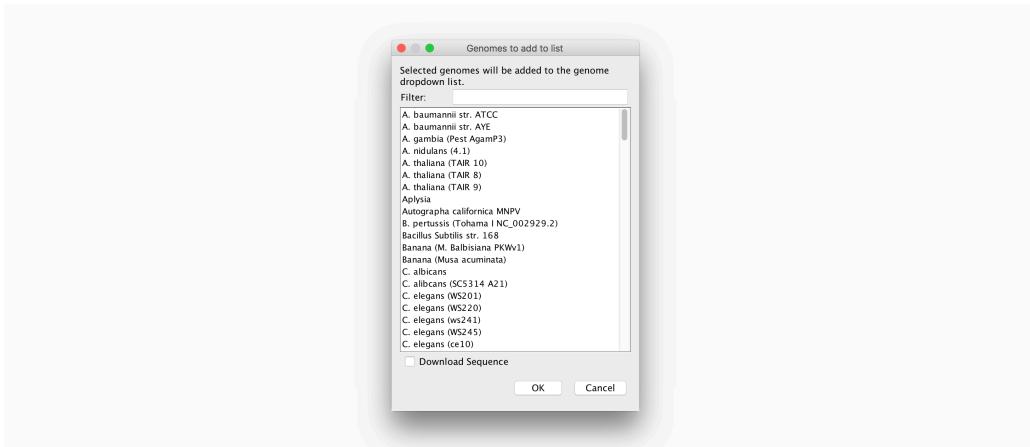


Figure 30.1

Note the colors, the orientations; some reads have different outlines; some regions have different colors. All of these are IGV specific **glyphs** by which IGV is trying to display information about the data.

30.5 What data does IGV come with?

The Broad Institute distributes data for IGV from a vast number of organisms and data sources. Do make a note – to be able to display their data, your nomenclature for chromosomes has to match theirs (more on this in the *What is in a name* (todo) chapter):

**Figure 30.2**

30.6 How do I create a custom genome in IGV?

To operate on a genome file that is not preloaded with IGV, we need to build it ourselves. Making a custom genome will typically require the following:

1. Identify the genome file and optionally a feature file you want to use.
2. Create an index for the genome
3. Load the new genome into IGV

How to make a gene file:

```
# Get a genbank file. Unfortunately we cannot load a Genbank file into IGV.
efetch -db nucleotide -format=gb -id=AF086833 > AF086833.gb

# We must convert the GenBank file into GFF3.
cat AF086833.gb | seqret -filter -feature -osformat gff3 > AF086833.gff

# We must also convert the GenBank file into FASTA.
cat AF086833.gb | seqret -filter -feature -osformat fasta > AF086833.fa

# Create an index of the FASTA file
samtools faidx AF086833.fa
```

You can now select the menu “Genomes -> Load Genome from File” to load

your FASTA file. Select the “File -> Load from File” option (or drag and drop) to select the GFF3 file onto your view. Success looks like this:

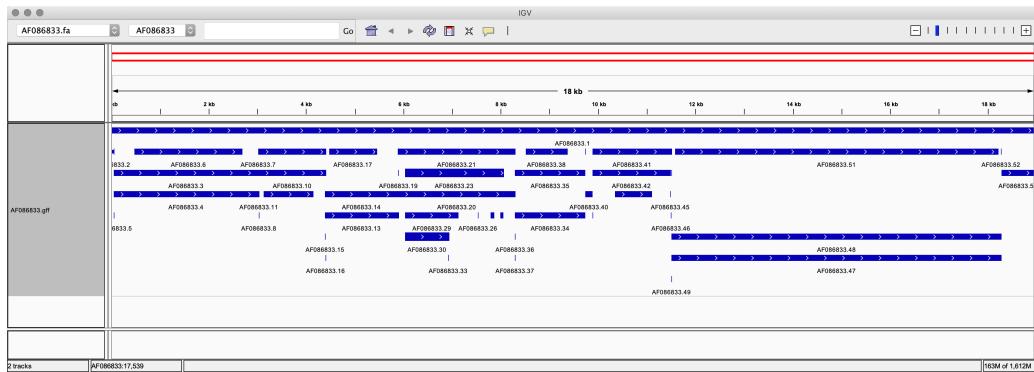


Figure 30.3

Keep at it, and you'll eventually succeed. Read the manual, explore the options, click on menus and importantly right click on fields and surfaces to see what type of contextual options are does the software offer.

Part VII

SEQUENCE ONTOLOGY

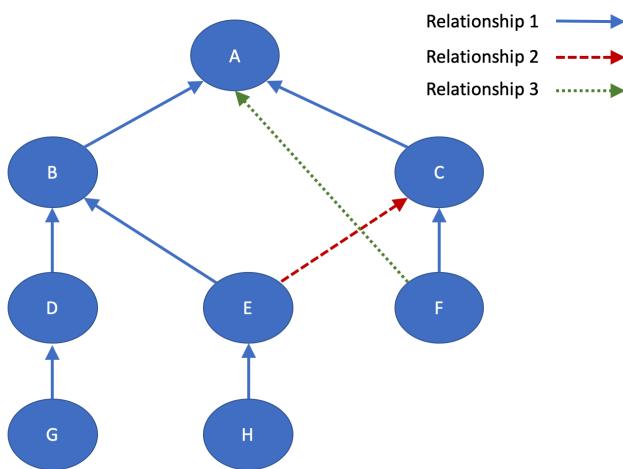
Chapter 31

What do the words mean?

Computational concepts and algorithms require exact definitions. It is exceedingly important that all scientists use the same words to describe the same information.

An *ontology* is a controlled vocabulary of *terms* or *concepts* and a restricted set of *relationships* between those terms. The most commonly used ontologies for bioinformatics data are:

- The Sequence Ontology (SO): a vocabulary for information related to sequence features.
- The Gene Ontology (GO), a vocabulary for information related to gene functions.



31.1 Why is the ontology necessary?

In the paper titled The Sequence Ontology: a tool for the unification of genome annotations¹, the authors state:

Unfortunately, biological terminology is notoriously ambiguous; the same word is often used to describe more than one thing, and there are many dialects. For example, does a coding sequence (CDS) contain the stop codon or is the stop codon part of the 3'-untranslated region (3' UTR)?

There really is no right or wrong answer to such questions, but consistency is crucial when attempting to compare annotations from different sources, or even when comparing annotations performed by the same group over an extended period of time.

It is essential to recognize that consistency matters more than “correctness” - there is no universally “correct” answer in biology.

31.2 Are there other ontologies?

Yes. Possibly too many. See a list at the The Open Bioinformatic Ontologies (OBO) Foundry².

Creating an ontology is a difficult, thankless job. Scientists often underestimate the difficulty of coming up with logically consistent definitions. For that reason, they often embark on overly ambitious ontology building adventures. Hence, there are many biomedical ontologies left in various states of abandon and disrepair. See the sorry situation of the RNA Ontology³ for example.

¹<https://genomebiology.biomedcentral.com/articles/10.1186/gb-2005-6-5-r44>

²<http://www.obofoundry.org/>

³<http://roc.bgsu.edu/>

31.3 Who names the genes?

A few different organizations have taken upon themselves the obligation to maintain a consistent naming scheme.

- For example HUGO Gene Nomenclature Committee (HGNC)⁴ available via <http://www.genenames.org/> is the only worldwide authority that assigns standardized nomenclature to human genes.

Other model and non-model organisms might have their consortia that may follow similar or very different naming conventions to label the genes for that organism.

Whereas the gene names are standardized other names, such transcript names, regulatory elements, etc. may have competing labeling. Converting between these across different organizations and their data releases continue to be one of the most annoying and unsolved problems of bioinformatics.

31.4 What will our data tell us?

In general, most bioinformatics-oriented analysis results fall into two categories (and combinations thereof):

1. **What a segment of DNA is**, described with the Sequence Ontology
-> annotation, classification
2. **What a segment of DNA does**, expressed with the Gene Ontology
-> functional analyses, pathway analysis

For example, a **de-novo genome assembly** attempts to reconstruct a genome that was measured after being broken into many small fragments. Once assembled some part of the genome may be annotated with terms such as the capacity to encode for proteins or the ability to regulate RNA expression. The ideal result of this analysis would be a genome where every base carries annotations that describe its role.

In contrast a typical **RNA-Seq** study, which aims to explain phenotypes by identifying the transcripts that show variations in their expression levels. Since transcripts have functional roles associated with them, the change in the abundance of a transcript is assumed to affect its functional role, and

⁴<http://www.genenames.org/>

hence to influence the phenotype of the organism. The ideal outcome of an RNA-Seq experiment is the identification of the mechanisms by which the DNA functions and those by which it produces an observed phenotype.

Chapter 32

Sequence ontology

32.1 What is the Sequence Ontology (SO)?

The Sequence Ontology (abbreviated as SO)¹ located at

- <http://www.sequenceontology.org/>²

is a collaborative ontology project for the definition of sequence features used in biological sequence annotation.

32.2 Where do I see Sequence Ontology (SO) terms used?

Most data sources will tag information with words that are defined in an ontology. For sequence data the ontology will be that of Sequence ontology (SO) Let's revisit the yeast feature file from the Unix tutorial:

```
wget http://data.biostarhandbook.com/data/SGD_features.tab
```

the second column shows what each feature “is”:

```
cat SGD_features.tab | cut -f 2 | head
```

prints:

¹<http://www.sequenceontology.org>

²<http://www.sequenceontology.org>

```

ORF
CDS
ORF
CDS
ARS
telomere
telomeric_repeat
X_element
X_element_combinatorial_repeat

```

or to summarize the most common ones:

```
cat SGD_features.tab | cut -f 2 | sort | uniq -c | sort -rn | head
```

we get:

```

7074 CDS
6604 ORF
484 noncoding_exon
383 long_terminal_repeat
377 intron
352 ARS
299 tRNA_gene
196 ARS_consensus_sequence
91 transposable_element_gene
77 snoRNA_gene

```

but what do these words mean? What is a “transposable_element_gene”? That is what the Sequence ontology is for.

32.3 How do I access the Sequence Ontology browser?

Search the Sequence Ontology Browser³ for the words that need to be defined.

For example, to find out what the term `X_element_combinatorial_repeat` means within a sequencing data context, we can visit the Sequence Ontology

³<http://www.sequenceontology.org/browser/obob.cgi>

32.3. HOW DO I ACCESS THE SEQUENCE ONTOLOGY BROWSER?285

Browser⁴ and find its definition⁵ to be:

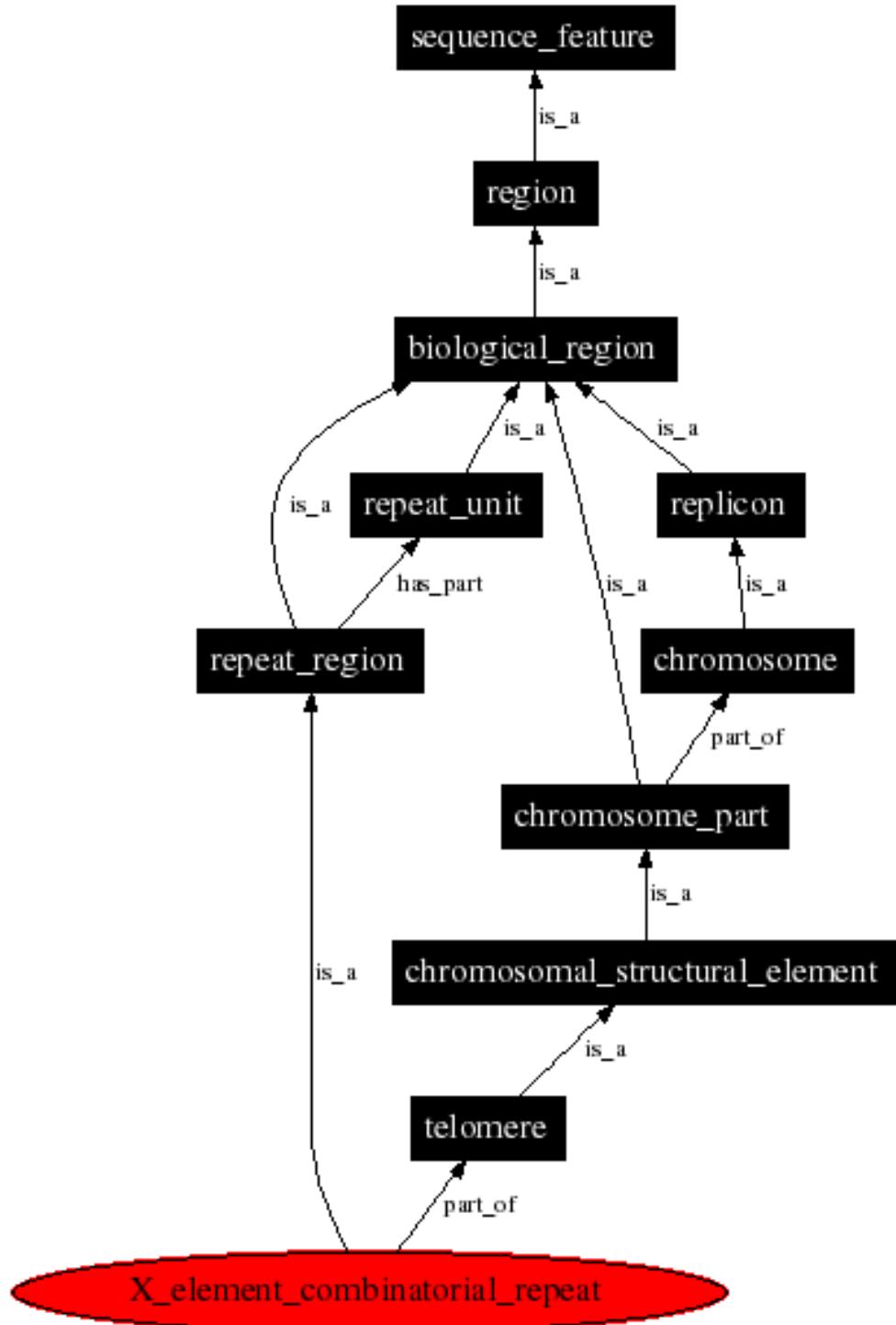
X element combinatorial repeat

An X element combinatorial repeat is a repeat region located between the X element and the telomere or adjacent Y' element.

A graphical display shows the relationship of the terms to its parents:

⁴<http://www.sequenceontology.org/browser/obob.cgi>

⁵http://www.sequenceontology.org/browser/current_svn/term/SO:0001484



32.4. DOES ALL SEQUENCING DATA OBEY THE RULES OF SO? 287

Note how the description above makes use of the word **X Element**, that is a new term, that in turn can also be found via another search in the browser. What is an **X Element**:

** X element**

The X element is a conserved region, of the telomere, of ~475 bp that contains an ARS sequence and in most cases an Abf1p binding site.

The definition has two new words **telomere** and **ARS**. It may feel like a wild goose chase. It won't really be like that. As you go higher in the tree of words, there will be fewer new terms. Keep searching until you find all term definitions.

Tip: click the words on the image to jump to another term - this makes navigating much more manageable.

32.4 Does all sequencing data obey the rules of SO?

It should, but sometimes it does not. You may some words used in a data are not "officially" defined. You see, the SO is more of a recommendation rather than a mandate. There is SO Police cracking down on misfits.

Adopt a defensive mindset and verify your data. Other odd and unexpected situations may occur. For example, we noticed that often data distribution sources will not include the sequence for the **stop codon** in a sequence labeled as **CDS** although based on the definition they should:

CDS

A contiguous sequence which begins with, and includes a start codon and ends with, and includes a stop codon.

Even data from Ensembl used to make this mistake - though I believe this has been corrected in the meantime. As a rule, spot check your data and verify that it is what you think it is.

32.5 How are the SO relationships defined?

Whereas SO terminology is precise, the rationale for the SO relationships between terms are more complicated to understand - and they feel more subject to interpretation. For that and other reasons, the majority of life scientists do not make use of the SO relationships and use the SO terms only.

For details we recommend consulting the publication:

The Sequence Ontology: A tool for the unification of genome annotations in Genome Biology, 2005⁶

32.6 Will I need to access the SO data directly?

While during regular use bioinformaticians typically don't need to access the original representation of the SO data but it is worth spending a bit a bit of time to understand how it is organized and what it contains.

Ontologies capture the state in which the science has reached in a moment of time.

Other, more complicated ontologies work the same way.

32.7 How can I investigate the SO data?

Finding the SO data is somewhat circuitous. There is a GitHub repository at <https://github.com/The-Sequence-Ontology/SO-Ontologies>. But even on that site, there is little information on how this data is formatted.

```
URL=https://raw.githubusercontent.com/The-Sequence-Ontology/SO-Ontologies/master/ontologies/obo/so.obo
wget $URL
```

The obo format is plain text and so-called “record oriented” rather than being “column oriented”. What this means is that information is distributed

⁶<https://genomebiology.biomedcentral.com/articles/10.1186/gb-2005-6-5-r44>

over many lines, and so called record separators divide the information into units. Here is an example for one record:

```
[Term]
id: SO:0000004
name: interior_coding_exon
subset: SOFA
synonym: "interior coding exon" EXACT []
is_a: SO:0000195 ! coding_exon
```

32.8 How many Sequence Ontology terms are there?

We can easily count how many SO terms are there:

```
cat so-simple.obo | grep 'Term' | wc -l
```

The entire Sequence Ontology is just 2359 terms. Is that a lot? Is that little? Are we close to be finished or not with naming things? It is hard to tell. Let's say that based on our current understanding of the genome we have 2359 terms to associate with sequences.

32.9 How can I quickly search the Sequence Ontology?

Grepping the raw data is probably the fastest way to find what you are looking for.

How many record definitions match the word `gene`:

```
cat so-simple.obo | grep 'name: gene'
```

prints:

```
name: gene_sensu_your_favorite_organism
```

```
name: gene_class
name: gene_part
name: gene_by_transcript_attribute
name: gene_by_polyadenylation_attribute
name: gene_to_gene_feature
name: gene_array_member
```

What we want is the name that matches `gene` only. We can specify that as `gene` followed by the end of the line. In pattern matching (and you will see details of this later) the *end of line* character may be specified as the `$` character.

```
cat so-simple.obo | grep 'name: gene$'
```

prints only one entry:

```
name: gene
```

The `-B` and `-A` flags make `grep` print the lines before and after a match. These two flags can be very handy.

```
bash cat so-simple.obo | grep 'name: gene$' -B 1 -A 6
```

will print:

```
id: SO:0000704
name: gene
def: "A region (or regions) that includes all of the sequence elements necessary to
comment: This term is mapped to MGED. Do not obsolete without consulting MGED ontology.
subset: SOFA
xref: http://en.wikipedia.org/wiki/Gene "wiki"
is_a: SO:0001411 ! biological_region
relationship: member_of SO:0005855 ! gene_group
```

32.10 How to search for other information?

While the website offers a visual representations, you can mine the Sequence ontology at the command line various pieces of information that otherwise would be more challenging to obtain. Which records match the word PCR:

```
cat so-simple.obo | grep 'PCR' -B 2 -A 2
```

We get information of the form:

```
...
--
id: SO:0000345
name: EST
def: "A tag produced from a single sequencing read from a cDNA clone or PCR product; typica
comment: This term is mapped to MGED. Do not obsolete without consulting MGED ontology.
subset: SOFA
--

id: SO:0001481
name: RAPD
def: "RAPD is a 'PCR product' where a sequence variant is identified through the use of PCR
synonym: "Random Amplification Polymorphic DNA" EXACT []
is_a: SO:0000006 ! PCR_product
created_by: kareneilbeck
creation_date: 2009-09-09T05:26:10Z
...
```

Part VIII

GENE ONTOLOGY

Chapter 33

Gene ontology

33.1 What is the Gene Ontology (GO)?

The Gene Ontology (GO)¹ is a controlled vocabulary that connects each gene to one or more functions.

As it happens, the name “Gene Ontology” is misleading. The ontology is intended to categorize **gene products** rather than the genes themselves. Different products of the same gene may play very different roles, labeling and summarizing all of these functions under the same gene name may (and often does) lead to uncertainty and confusion.

33.2 How is the GO designed?

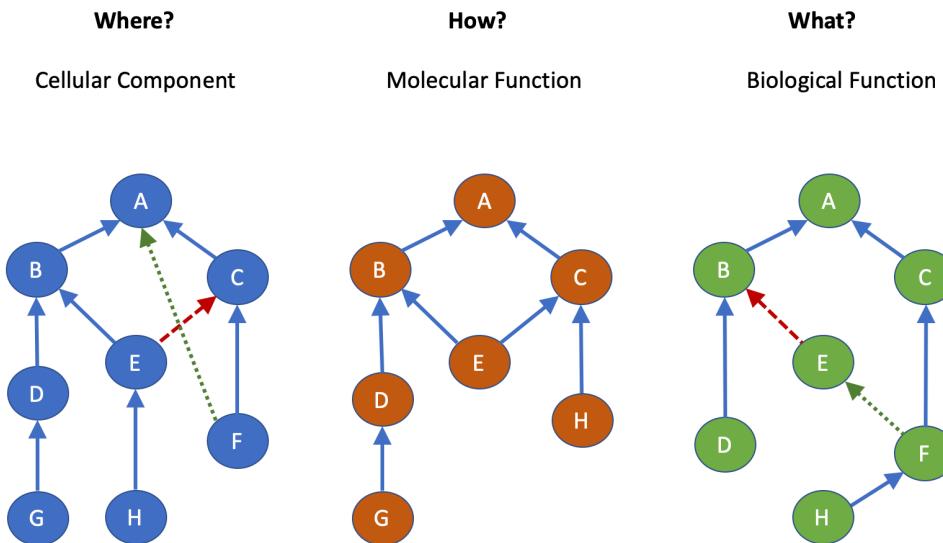
The GO ontology is a directed, acyclic graph structure where nodes are the GO terms and the edges are the relationships between these terms. Each child node is more “specific” than its parent, and function is “inherited” down the line.

To further separate the concepts the GO project has developed no one but three structured, independent sub-ontologies that describe gene products in a species-independent manner. The sub-ontologies characterize the gene products as follows:

¹<http://geneontology.org/>

- **Where does the product exhibit its effect?** -> Cellular Component (CC) This ontology relates the gene product to a component of a cell, that is, part of a larger object, for example, *cell*, *nucleus*, *Golgi membrane*, *SAGA complex*
- **How does it work?** -> Molecular Function (MF) Which biochemical mechanism characterizes the product? These terms describe activities that occur at the molecular level: *lactase activity*, *actin binding*
- **What is the purpose of the gene product?** -> Biological Process (BP) The general rule to assist in distinguishing between a biological process and a molecular function is that a process must have **more than one** definite step: *cholesterol efflux*, *transport*, *mitotic prophase*

A graphical representation for GO is the following:



Each sub-graph above is independent of the others. When we perform functional annotation we need treat each sub-graph separately. In a sense we perform functional annotation three times, once for each category. The same gene product may have fewer or more annotations in different categories.

33.3 What kind of properties do annotated gene products have ?

- Each gene product may have entries in each of the different categories.
- Each gene product may have multiple entries in each of the categories
- Leaf nodes (nodes with no children) are the most specific representation of knowledge.

For example a product may be annotated as:

- Cellular Component GO:0032580: Golgi cisterna membrane
- Molecular Function GO:0008107: galactoside 2-alpha-L-fucosyltransferase activity
- Biological Process GO:0005975: carbohydrate metabolic process
- Biological Process: GO:0006486: protein glycosylation

There is typically no reason to believe that the above characterization is complete or that it captures all the representative functions for that product. When you first start to grasp what the GO data is, what it can and cannot tell us, is where you start to realize just how pioneering our work is, how early in the discovery process we all are. We are still trying to invent the wheel, and we have come up with a triangular shape so far.

33.4 Where can access the GO online?

Various organization have taken up the task of visualizing GO terms, with various success when it comes to usability and informativeness:

- GeneOntology.org² is the “official” data originator.
- Quick Go³ by EMBL
- AmiGO⁴ Gene Ontology browser

²<http://www.geneontology.org/>

³<https://www.ebi.ac.uk/QuickGO/>

⁴<http://amigo.geneontology.org/amigo>

33.5 How are GO terms organized?

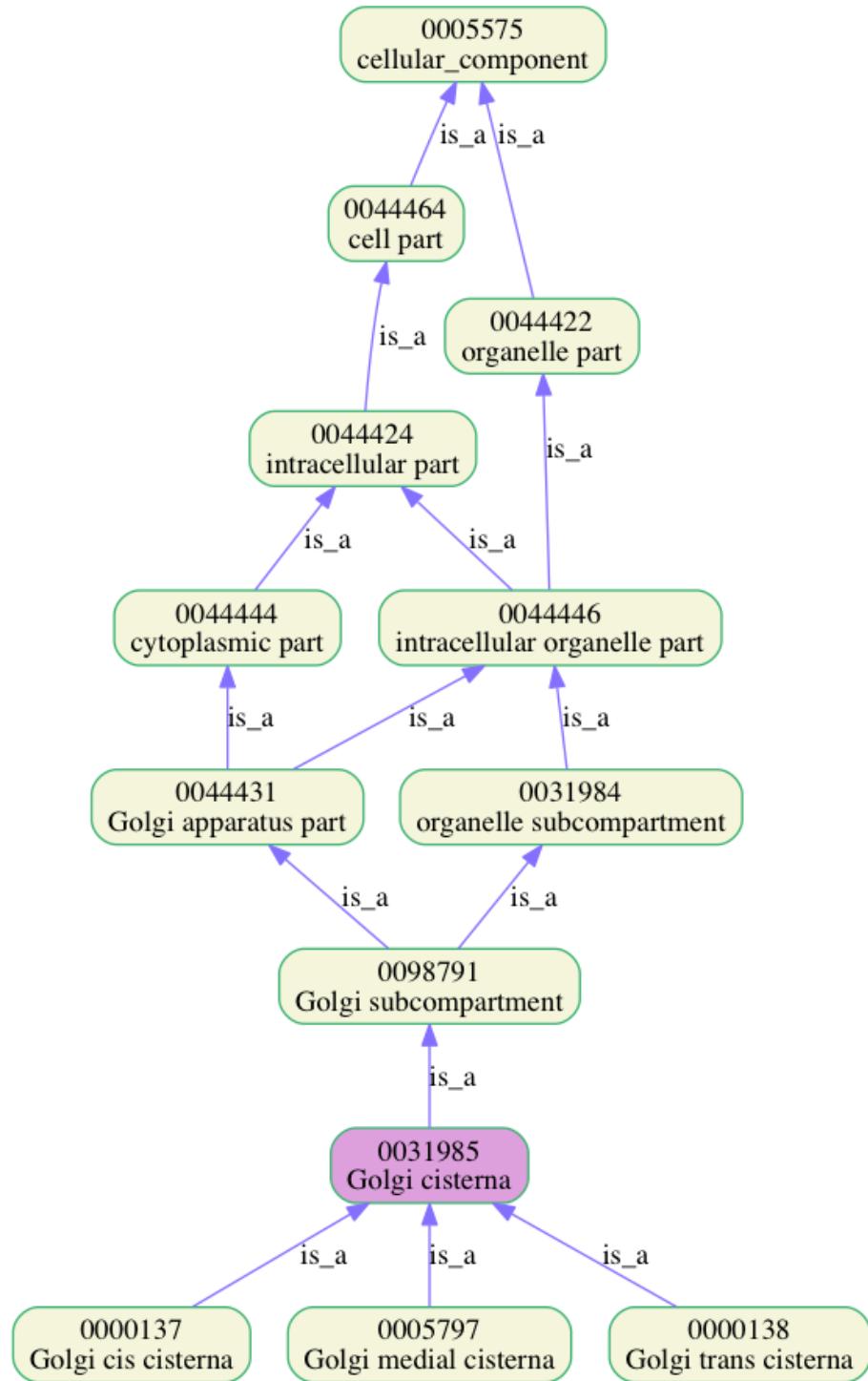
The GO ontology is structured as a directed, acyclic graph where each term has defined relationships to one or more other terms in the same domain, and sometimes to other domains. The GO vocabulary is designed to be species-agnostic and includes terms applicable to prokaryotes and eukaryotes, and to single and multicellular organisms.

Here is the definition of the Golgi Cisterna (GO:0031985) visualized with goatools⁵ (see the chapter for `goatools` usage for installation instructions):

```
plot_go_term.py --term GO:0031985 go-basic.obo
```

produces the following depiction for the terms:

⁵<https://github.com/tanghaibao/goatools>



33.6 Where can the visualize GO terms online?

The Gene Ontology⁶ website is the authoritative source for definitions but is not particularly well suited for data interpretation.

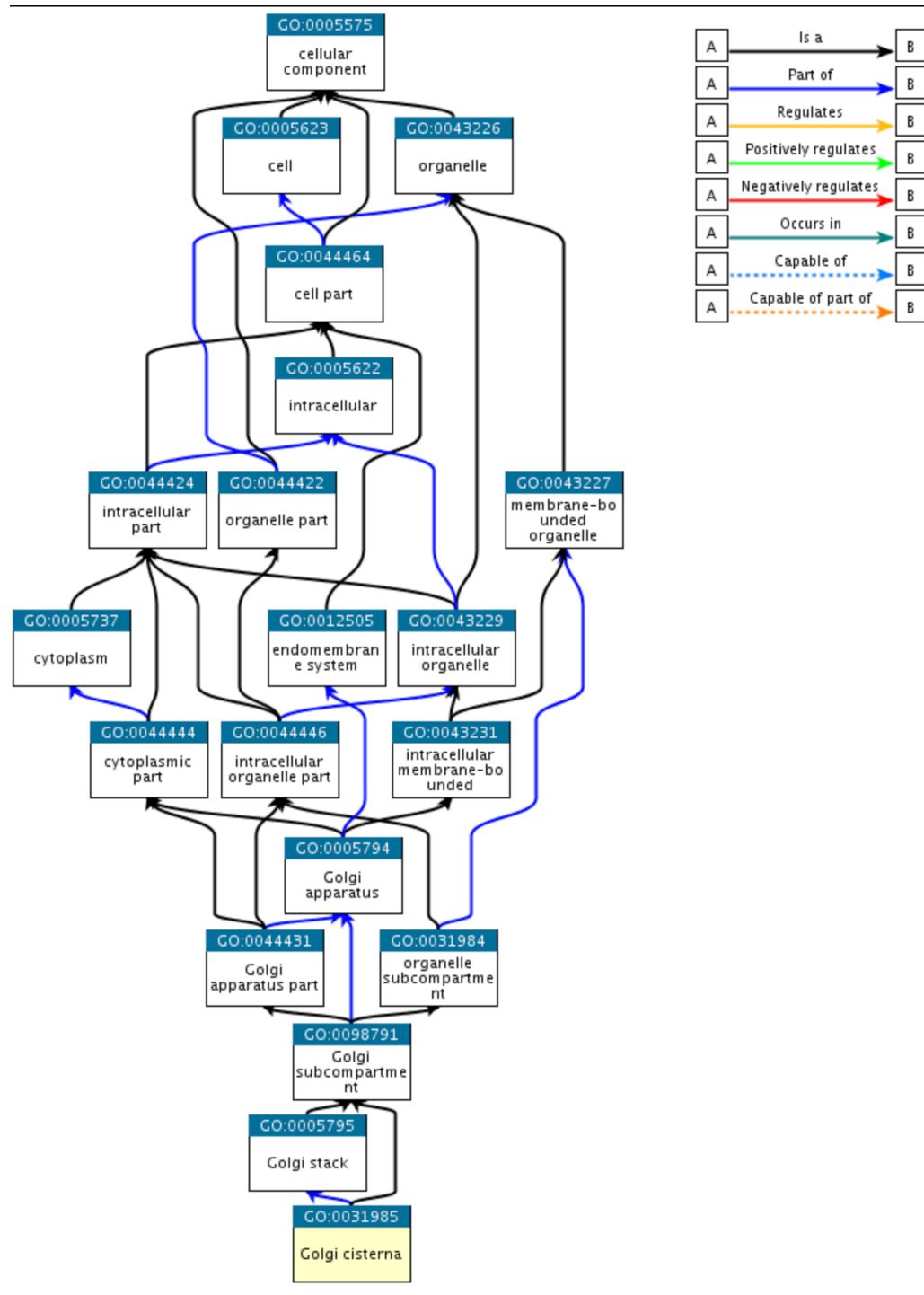
The Quick GO⁷ service from the European Bioinformatics Institute offers a web interface with more user-friendly functionality.

The same GO term Golgi Cisterna (GO:0031985) viewed in QuickGO⁸ shows

⁶<http://geneontology.org/>

⁷<https://www.ebi.ac.uk/QuickGO/>

⁸<https://www.ebi.ac.uk/QuickGO/>



When compared to the other visualization, the images are similar, but they do not represent identical information. Observing differences when visualizing ontology data is a common theme when working with Gene Ontology. It is often not easy to identify how an annotation is being generated/filtered or expanded.

There are so-called reduced (slim) vocabularies that simplify the data to broader terms at the cost of losing some of its lower level granularity.

Chapter 34

Understanding the GO data

While during regular use we may not need to access the original GO data it is worth spending just a bit a time to understand what it contains and how it is laid out. It will give you a different perspective, and a deeper understanding of what is possible to infer in biology based on the way you chose to store the data.

Remember every time you do a so-called “functional analysis” you will make use of the information stored in this data! Do you see now how *everything is data*, we take one data, produce results using that data, then look it up against another data.

34.1 What format is the GO data in?

GO data are available for each organism from the GO Download¹ page. The underlying data for the gene ontology consists of two files:

1. Gene ontology definition file.
2. A gene association file.

These two pieces of information connect the ontology terms to gene product names. The GO files are surprisingly small and can be easily downloaded:

¹<http://geneontology.org/page/download-annotations>

```
wget http://purl.obolibrary.org/obo/go.obo
```

34.2 What does a GO term file contain?

The content of the core go.obo file is constructed of records in the form:

```
[Term]
id: GO:0000002
name: mitochondrial genome maintenance
namespace: biological_process
def: "The maintenance of the structure and integrity of
      the mitochondrial genome; includes replication and
      segregation of the mitochondrial chromosome." [GOC:ai, GOC:vw]
is_a: GO:0007005 ! mitochondrion organization
```

What is this information?

It is *biology* encoded into a computational form. Information that is not yet encoded via the GO cannot be found automatically with tools. The correctness of biological data analysis will depend on the quality of this file.

34.3 What is a GO association file?

An association file connects a gene product to a GO term. Like so:

```
IGLV3-9          GO:0005886
IGLV3-9          GO:0003823
IGLV3-9          GO:0002250
...
```

This would mean that gene product IGLV3-9 is annotated with the terms GO:0005886, GO:0003823 and GO:0002250. What is GO:0005886? It is defined in go.obo:

```
cat go.obo | grep "id: GO:0005886" -A 5
```

produces:

```

id: GO:0005886
name: plasma membrane
namespace: cellular_component
alt_id: GO:0005904
def: "The membrane surrounding a cell that separates the cell from its external en

```

You can see how the 7 digits long GO ids in the form of `GO:xxxxxxx` are defined in the `go.obo` file. There will be a different association file for each organism and the file will contain more columns beyond the two shown above. Sometimes there is a date, an evidence code, other ids etc.

34.4 Where can I find the association files for different organisms?

The list of association files is maintained at:

- <http://geneontology.org/page/download-go-annotations>

34.5 How to get the human gene association file?

The gene association file for human gene products can be obtained as:

```

# Get the file.
wget http://geneontology.org/gene-associations/goa_human.gaf.gz

# Unzip the compressed file.
gunzip goa_human.gaf.gz

```

The file names may occasionally change (sometimes suddenly and with little prior notice) If the link is nonfunctional please visit the Gene Ontology download page² and find and substitute the correct location.

²<http://geneontology.org/page/download-annotations>

34.6. WHAT FORMAT DOES THE GO ASSOCIATION FILE HAVE?305

34.6 What format does the GO association file have?

The file is in the so-called GAF format that is defined at <http://geneontology.org/page/go-annotation-file-gaf-format-21> that states:

Annotation File Fields

The annotation flat file format is comprised of 17 tab-delimited fields.

Column Content	Required?	Cardinality	Example
1 DB	required	1	UniProtKB
2 DB Object ID	required	1	P12345
3 DB Object Symbol	required	1	PHO3
4 Qualifier	optional	0 or greater NOT	
5 GO ID	required	1	GO:0003993
6 DB:Reference (IDB:Reference)	required	1 or greater	PMID:2676709
7 Evidence Code	required	1	IMP
8 With (or) From	optional	0 or greater	GO:0000346
9 Aspect	required	1	F
10 DB Object Name	optional	0 or 1	Toll-like receptor 4
11 DB Object Synonym ((Synonym))	optional	0 or greater	tToll Tollbooth
12 DB Object Type	required	1	protein
13 Taxon((taxon))	required	1 or 2	taxon:9606
14 Date	required	1	20090118
15 Assigned By	required	1	SGD
16 Annotation Extension	optional	0 or greater	part_of(CL:0000576)
17 Gene Product Form ID	optional	0 or 1	UniProtKB:P12345-2

Figure 34.1: GAF Format

Comments are specified with ! the rest are tab separated and column-oriented data of the form:

UniProtKB AOA024QZP7 CDC2 GO:0004672 GO_REF:0000002 ...

Here the first few columns indicate the gene product name, then its corresponding GO association is listed. Since GO terms have a hierarchical representation, a GO term for a gene name implies that the gene is annotated with all the parent nodes as well. The parent nodes typically describe the similar characteristics but in a broader sense.

In the gene association file, each gene product may be associated with one or more GO terms in each category.

34.7 Do the association files represent all of the accumulated biological knowledge?

To some extent yes. While life sciences knowledge is much richer and the domain expertise of individuals is typically more nuanced, the GO associations represent the “automatically searchable” dimension of this knowledge.

The `go.obo` file measures 33Mb whereas the compressed gene association file is a mere 7Mb. It is nothing short of mind-boggling to realize that these files are a road map to and product of an entire scientific field.

NOTE: The GO files represent the accumulated and distilled results of the work of tens of thousands of scientists over many decades! Every automated, genome-based diagnosis, data interpretation, functional characterization, or genetic test relies on the content of this data. The life science data interpretation relies on Gene Ontology being correct, well annotated, and available to all.

34.8 What kind of properties does the GO data have?

We will investigate the Gene Ontology.

```
# Get the GO data and their associations.  
wget http://purl.obolibrary.org/obo/go.obo  
wget http://geneontology.org/gene-associations/goa_human.gaf.gz  
  
# How big are these files  
ls -lh  
  
# Uncompress and remove lines starting with the comments character!  
# from the annotation file. This makes subsequent commands simpler.  
gunzip -c goa_human.gaf.gz | grep -v '^!' > tmp
```

```

# Move the temporary files under the original name.
mv tmp goa_human.gaf

# How many gene to association mappings are there?
cat goa_human.gaf | wc -l
# 474213 (this number changes when the data is updated)

# Gene names are in column 3.
cat goa_human.gaf | cut -f 3 | head -20

# Gene names are in column 3. Find the unique ones.
cat goa_human.gaf | cut -f 3 | sort | uniq -c | head -20

# How many unique genes are present?
cat goa_human.gaf | cut -f 3 | sort | uniq -c | wc -l
# 19719 (this number changes when the data is updated)

```

Intuitively we would expect an increase of annotations as science progresses, yet over the years we often noted decreases in the numbers of annotations in this file.

34.9 What are the most annotated human genes and proteins?

The association file contains both gene and protein ids.

```

# What are the ten most highly annotated proteins in the GO dataset?
cat goa_human.gaf | cut -f 2 | sort | uniq -c | sort -k1,1nr > prot_counts.txt

# What are the ten most highly annotated gene names in the GO dataset?
cat goa_human.gaf | cut -f 3 | sort | uniq -c | sort -k1,1nr > gene_counts.txt

```

The structure of the file is:

```
cat gene_counts.txt | head
```

Produces:

```
724 TP53
669 GRB2
637 EGFR
637 UBC
580 RPS27A
570 UBB
565 UBA52
511 CTNNB1
422 SRC
```

A neat little tool called `datamash` lets us do data analytics at the command line.

```
datamash --help
```

Unfortunately the `uniq -c` command pads numbers with a variable number of spaces. We need to `squeeze` those into a single space. The command `tr -s` can do that.

```
cat gene_counts.txt | tr -s ' ' | head
```

Even after squeezing there is still one empty space in front of each column, so when later we split by the space character the numbers will be into column 2. This can be super confusing if you don't notice the little padding and you try to read column 1 the way it looks logical! Remember when we recommended that you make the terminal light, the fonts big, and legible? This another reason why. Now knowing that the information is in column 2, we can use `datamash` to answer a few questions:

What is the average number of annotations per gene?

```
cat gene_counts.txt | tr -s ' ' | datamash -t ' ' mean 2
# 24.048582585324
```

Looks like on average there are 24 annotation per gene. Now, what are the mean, minimum, maximum and standard deviation for annotations per gene:

```
cat gene_counts.txt | tr -s ' ' | datamash -t ' ' mean 2 min 2 max 2 sstdev 2
# 24.048582585324 1 830 33.83047198404
```

This indicate that the number of annotations is highly variable.

34.10 What are the ten most highly annotated genes in the GO dataset?

The commands

```
cat gene_counts.txt | head
```

Produce:

```
724 TP53
669 GRB2
637 EGFR
637 UBC
580 RPS27A
570 UBB
565 UBA52
511 CTNNB1
422 SRC
```

Showing us that TP53 has the most annotations. Interestingly in December 2016, when we first wrote this chapter the list was the following:

```
1337 HLA-B
918 HLA-A
819 HLA-DRB1
789 TP53
...
```

Today the HLA-B gene produces radically different output:

```
cat gene_counts.txt | grep HLA-B
# 48 HLA-B
```

Whoa! What is going on? How did HLA-B drop from 1337 annotations to just 48?

(For a short period of time I was even suspecting some sort of prank. You see 1337 is the so called “leet” code (elite) in “leet speak”. Where 1337 is the designation of a person skilled at computer programming or hacking. Is this a prank from the Bioinformatics Gods punishing me? Maybe, or maybe it just bioinformatics... :-))

34.11 Do the GO annotations change?

As we mentioned the latest GO data download produces a starkly different output than it originally did. Notably, HLA genes are not in the top ten anymore. It reinforces the transient nature of annotations - as new information or different filtering parameters are applied the information content can drastically change.

Why did the HLA genes disappear? Was their presence an error. Perhaps ... but if so what does it mean for the publications that we produced during that time that the error was present? Was anyone ever publicly notified about these changes? The previous was a rhetorical question, but if you wanted to know then short answer is “pfff, of course not, are you kidding?”. We discovered this discrepancy by accident as we were rerunning the same command a year later. There it goes to show how not everything is all and well in the GO world...

34.12 How complete is the GO?

We may wonder: how complete are these GO annotations? Would it be possible to estimate what percent of all functional annotations have been discovered so far?

First we could tabulate the dates assigned to each evidence and see how many pieces of evidence are produced per year.

It is “reasonably easy” to extract and tabulate the date information from the two files that we downloaded to get the rate of information growth in the GO:

Don’t worry about this command if you don’t get it yet. We’ll cover `awk` programming later, but we want to demonstrate some results. What we do is extract the year from the evidence codes listed in the 14th column. The `\` is the so called line continuation command, it says to ignore the line end and imagine the whole command is on one line. It is used to wrap commands that would stretch to far out and make them more readable:

```
#  
cat goa_human.gaf \  
| cut -f 14 \  
| awk '{ print substr($1, 1, 4) }' \  
| sort \  
| uniq -c \  
| sort -k 2,2 -n \  
| awk '{ print $2 "\t" $1 }'
```

The code below produces the number of annotations verified in a given year.

Here are the outputs when run in 2016

```
...  
2008    9682  
2009    13322  
2010    15026  
2011    23490  
2012    16428  
2013    60555  
2014    34925  
2015    33096  
2016    235077
```

And when run at the beginning of 2019:

```
...  
2008    8694  
2009    12512  
2010    13011  
2011    21186  
2012    14693  
2013    57417  
2014    23636  
2015    19656  
2016    24109  
2017    56695  
2018    190351
```

This result is quite surprising, and we don't quite know what to make of it. It seems that most evidence is assigned to the latest year. Or maybe there is some feature to the system that attaches the newest date to every evidence that is being re-observed? We don't quite know.

There is also a surprising dip in the number of annotations in 2012, then a subsequent bump in 2013.

Finally we can note that the number of annotations assigned to previous years always decreases.

Chapter 35

Functional analysis

For the majority of people, the reason for studying bioinformatics is to make sense of the data. After all, we all want to gain insights into biological phenomena.

35.1 The sorry state of data categorization

Surprisingly enough, whereas scientists are eager to fund each other when it comes to gathering data, the same people are unexpectedly short-sighted when it comes to funding the organization and categorization of the information that has already been collected. It seems that in their minds once data has been obtained, it becomes a solved problem: “just put it in a database.” In reality, organizing existing knowledge is difficult, thankless and greatly unappreciated effort - that everyone dearly needs, but few can understand and appreciate the magnitude and complexity of the task. It is a manifestation of the so-called tragedy of commons¹ where: “individual users acting independently according to their self-interest behave contrary to the common good.”

As an example, the Kyoto Encyclopedia of Genes and Genomes (KEGG)² database had to change its licensing and started to charge thousands of dollars for bulk data access because they were unable to fund their service

¹https://en.wikipedia.org/wiki/Tragedy_of_the_commons

²<http://www.genome.jp/kegg/>

otherwise.

Companies have recognized the bottleneck and the commercial potential of the inefficiencies of the process. Many offer services that solve the increasingly painful process of summarizing data. Of course, the services that they offer come with eye-popping pricing and that makes them practically inaccessible to students, publicly funded groups or citizen scientists. We want to make it clear that we are not against the commercialization of services. The concerning issue is the increasing challenges of data interpretation makes a commercial services almost a necessity in some fields, rather than a choice.

35.2 What is a functional analysis?

A functional analysis typically means interpreting data in a context that explains observed phenomena, supports or rejects a hypothesis, discovers a mechanism.

As it happens most life science experiments do not directly measure the attributes of interests. To use an analogy, for example, to weigh an object, we could place it on a scale, and we could measure the weight instantly. But what if the object was not accessible - we would have to infer its weight from observing, other related information. Perhaps as an astronomer does, maybe the light bends when going around the object. Of course by making indirect observations we have to contend with many other facets that will greatly complicate the process. Most life sciences observations are just like that, we always have to consider multiple factors, and then combine our data with that collected by others to make sense of the observation.

In the most common interpretation, functional analysis means interpreting data in the context of all known information gathered up present time.

For a review, see Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges³ from 2012, Plos Computational Biology.

³<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375>

35.3 What is an Over-Representation Analysis (ORA)?

An Over-Representation Analysis (ORA) attempts to find representative functions of a list of genes. It does this by comparing the number of times a function has been observed in an experiment to a baseline. it is one of the most frequently used method to identify common functions in a group of genes and transcripts.

In Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges⁴ the authors summarize ORA as:

... over-representation analysis (ORA), which statistically evaluates the fraction of genes in a particular pathway found among the set of genes showing changes in expression. It is also referred to as “2×2 table method” in the literature. ORA uses one or more variations of the following strategy: first, an input list is created using a certain threshold or criteria. For example, a researcher may choose genes that are differentially over- or under-expressed in a given condition at a false discovery rate (FDR) of 5%. Then, for each pathway, input genes that are part of the pathway are counted. This process is repeated for an appropriate background list of genes (e.g., all genes measured on a microarray).

In a nutshell, an ORA analysis computes the following numbers:

1. How many functions do all genes have in total?
2. How many functions do my selected (subset) genes have?
3. How many functions should we observe if we were to select a random subset of genes?
4. How many functions did we observe for my selected genes? Is that number larger (functional enrichment) or smaller (functional depletion) than the expectation?

An ORA analysis will compute how likely is that the last number, that of the functions in our subset could be obtained by random chance.

⁴<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375>

If the number of functions is more extensive than expected, it is usually interpreted as the subset being enriched with respect to the function.

35.4 Are there different ways to compute ORA analyses?

If you followed the section on Gene Ontology, you know that the GO files have a relatively simple format to them. Moreover, the data structure is that of a (network) tree. Building a program that assigns counts to each node and can traverse the tree is a standard problem that requires only moderately advanced programming skill. Yet deciding under what circumstance to call a count over-represented is not settled **at all!** Different methods may produce wildly different outcomes.

Since data interpretation is such an acute problem, a “cottage industry” of ORA tools exists, perhaps even hundreds of such tools. Alas the large number only means fragmentation and lack of support for each, the vast majority of the tools become abandoned, are failing or even worse give the wrong answer tacitly.

Moreover, the accuracy of a method will critically depend on it integrating the most up to date information. As a tale of caution, we note that the DAVID: Functional Annotation Tool⁵ was not updated from 2010 to 2016! Over this time it was unclear to users whether the data that the server operates on includes the most up to date information. New functions and annotations are added on an almost weekly basis, over the many years this difference increased considerably. We believe that by the end of the 2016 DAVID operated on a mere 20% of the total information known in biology. Nevertheless, it had gained thousands of citations every single year.

As you can see above in 2015 alone, DAVID collected over 5000 citations, yet at that time it contained only 20% of the known annotations. The “good” news is that there was an update in 2016 when the information has caught up hence the data is not as far behind as it used to.

The sorry state of DAVID has been a sore point for many. The “update” might have been in response to the growing chorus of complaints:

⁵<https://david.ncifcrf.gov/home.jsp>

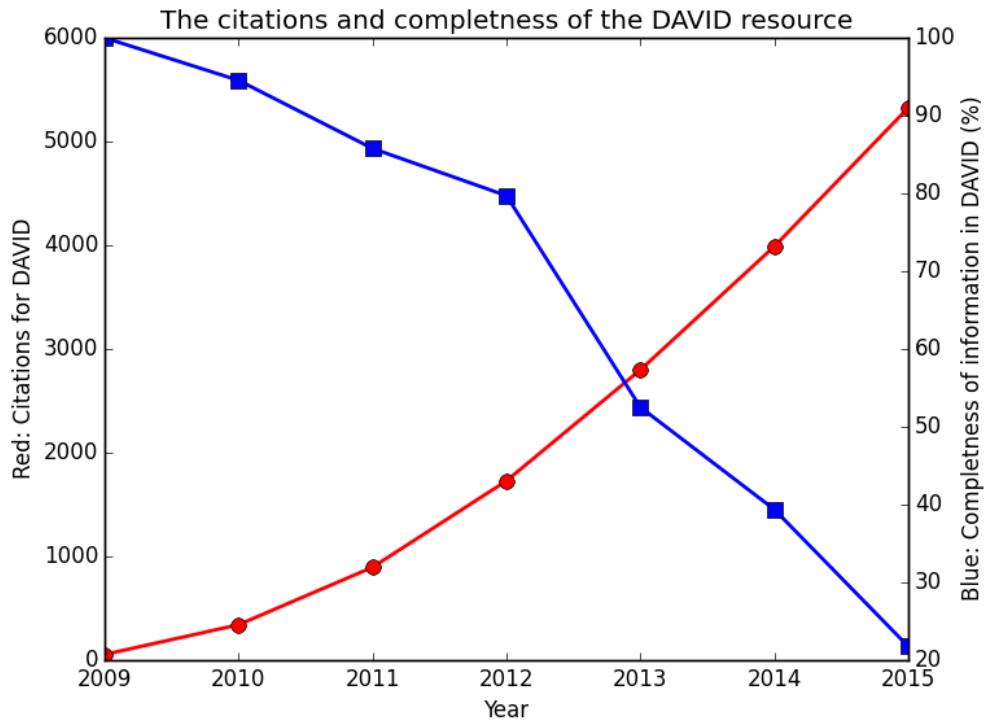


Figure 35.1

- Impact of outdated gene annotations on pathway enrichment analysis⁶
Nature Methods, 2016

We also refer the reader to several Biostar posts:

- Is the DAVID knowledgebase not updated anymore?⁷
- GO analysis: DAVID vs. GREAT vs. GOrilla⁸
- Finally the much awaited DAVID update has taken place⁹

While it is great that there was an update in 2016, by 2019 the data has again become obsolete. As Yogi Berra would say: “It’s like déjà vu all over again.”

⁶<http://www.nature.com/nmeth/journal/v13/n9/full/nmeth.3963.html>

⁷<https://www.biostars.org/p/100427/>

⁸<https://www.biostars.org/p/177998/>

⁹<https://www.biostars.org/p/193803/>

35.5 What are problems with the ORA analysis?

The shortcomings of the overlap analysis are that:

1. ORA analysis does not account for the magnitude of expression levels.
The gene is either in the list or not.
2. ORA typically uses only a subset of genes - the cutoffs are “arbitrary” in the sense that they are based on convention rather than an objective measure.
3. Genes and functions are all considered independent of one another. For statistical assumptions to work, this is an essential requirement. If the independence constraint does not hold, then the mathematical basis for the test does not hold either. As we all know many functions in the cell are strongly interdependent.

35.6 Why do we still use the ORA analysis?

If you read the page so far you can that ORA analyses have many pitfalls. Why do we still use it?

I think the reason for this is that the concept is simple, quick to compute, and most scientists understand what takes place. For many other techniques the internal workings are so complicated and “black-box”-like that often you will be left with nothing more than a belief that that box probably does what it claims to.

In our opinion, ORA analyses should be used as a hypothesis generation and testing framework but not necessarily as a method that provides the final answer to a problem.

35.7 What is a Functional Class Scoring (FCS)?

In Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges¹⁰ the authors summarize FCS as:

1. “A gene-level statistic is computed using the molecular measurements from an experiment.”
2. “The gene-level statistics for all genes in a pathway are aggregated into a single pathway-level statistic.”
3. “The final step in FCS is assessing the statistical significance of the pathway-level statistic.”

FCS methods use this information to detect coordinated changes in the expression of genes in the same pathway. Finally, by considering the correlated changes in gene expression, FCS methods account for dependence between genes in a pathway, which ORA does not.

Now you can see another reason why ORA analyses are still used. At least we understand what the ORA does - whereas the text above could be applied broadly and generically to cover a great variety of techniques - most of which could be complicated and nearly impossible to understand for a life scientist. When people use an FCS tool they become mere “passengers” of it - they run the tool then hope for the best that it takes them where they want to go.

There are many implementations of each method and approach some more explicit than others - do refer to the paper¹¹ for a list.

¹⁰<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375>

¹¹<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375>

35.8 Should I trust the results of functional analyses?

We advise caution. The statistical methods used to determine the values that biologists are so fond of and use as cutoffs (p-values, FDR) are particularly ill-suited when derived from functional annotation. It is not the methods that are flawed, it is the data that does not conform to assumptions that the methods make.

For a statistical method to work correctly, it needs access to reasonably unbiased and representative data. To determine what “is enriched”, we need to know what “not-enriched” looks like. But data on gene function does not conform to either requirement; discoveries are driven by the need to find unusual behaviors. Genes under study are selected by scientists interested in certain phenomena and will be published only when something interesting is found. Thus there is “bias” in the selection and a “bias” in what makes into a function repository.

Long story short, don’t put too much stock into p-values and other measures. Use the statistical measures more as a suggestion or estimation rather than precise, scientific criteria.

Chapter 36

Gene set enrichment

36.1 What is a gene set enrichment analysis?

“Gene set enrichment analysis” refers to the process of discovering the common characteristics potentially present in a list of genes. When these characteristics are GO terms, the process is called “functional enrichment.”

Gene set enrichment analysis can be performed in several ways. One of the most commonly used is approach is the *over-representation analysis (ORA)* that we have described in a previous chapter. An ORA examines the genes in a list, summarizes the GO annotations for each, then determines whether there any annotations are statistically over-represented (compared to an expectation) in that list.

36.2 What tools are used to perform enrichment analysis?

Of all bioinformatics tools developed to-date, the GO enrichment tools are in the worst shape. Many are in various states of abandon. Unlike other software that, once implemented correctly, keeps working over the years, a GO tool analysis critically depends on using the most up-to-date GO annotations. Hence, GO tools require ongoing maintenance and effort, which their creators are often unable to provide long term.

Most enrichment tools seem to incorporate the word “GO” into various “cute” and humorous” tool names, such as GOOSE, GOrilla, AmiGO, BINGO, QuickGO. Alas the joke gets old very quickly, most of these tools are surprisingly hard to use and may produce inconsistent results. We recommend lots of caution when using GO-based tools, as all are notoriously difficult to evaluate for correctness.

Over the years, we have used and recommended (with some caveats though) the following tools:

- AgriGO¹ Web-based GO Analysis Toolkit and Database for Agricultural Community. We love the visualizations that this produces. Short review: ” “A++, would do business again.”
- DAVID² This is the GO tool biologists love. It is the “most generous” of them all, as it produces copious amounts of output. DAVID was nearly abandoned for more than six years but was finally updated in 2016. In the meantime (between 2009 and 2016), thousands of publications used it to produce GO enrichment analyses on increasingly outdated GO data. After 2016 the data has not been updated again despite the title on the main page proudly stating **Welcome to DAVID 6.8, 2003-2018!** Is that misleading title just an unfortunate interface choice? false advertising? scientific misconduct? You decide!
- Panther³ is offered directly from the GO website. It produces very-limited information and no visualization. At least it is probably not completely wrong. It is the equivalent of “better than nothing.” The bar is set very low.
- goatools⁴ a command line Python scripts. Published as GOATOOLS: A Python library for Gene Ontology analyses⁵ in Scientific Reports, 2018.
- ermineJ⁶ Standalone tool with easy to use interface. It has detailed documentation. Used to be our first choice.

¹<http://bioinfo.cau.edu.cn/agriGO/>

²<https://david.ncifcrf.gov/home.jsp>

³<http://geneontology.org/>

⁴<https://github.com/tanghaibao/goatools>

⁵<https://www.nature.com/articles/s41598-018-28948-z>

⁶<http://erminej.chibi.ubc.ca/>

- GOrilla⁷ Web-based; good visualization; downloadable results (as Excel files); easily see which genes contribute to which enriched terms; results pages indicate the date of last update GO database (often within the last week).
- ToppFun⁸ - a suite of tools for human genomes that integrates a surprising array of data sources.

36.3 Will different tools produce different results?

Biostar Quote of the Day: Why does each GO enrichment method give different results?⁹

I'm new to GO terms. In the beginning, it was fun, as long as I stuck to one algorithm. But then I found that there are many out there, each with its own advantages and caveats (the quality of graphic representation, for instance).

[...] As a biologist, what should I trust? Deciding on this or that algorithm may change the whole story/article!"

Figuring out how which method to trust is a challenge with no clear solution. Unlike most other computational approaches where there is an objective way to validate the quality of a result, in most GO analyses we don't know how to tell when it worked correctly.

Overall GO enrichment is surprisingly subjective and different methods will produce different results.

Note: When using different tools to find enrichment, the two different results could be both correct, at the same time. Or both could be wrong, at the same time. Usually, they agree on about

⁷<http://cbl-gorilla.cs.technion.ac.il/>

⁸<https://toppgene.cchmc.org/>

⁹<https://www.biostars.org/p/122893/>

50% of targets. If you get over 50% agreement with different methods, we envy you. Show it off on Instagram.

36.4 How do I perform a gene set enrichment analysis?

Most tools follow the same principles:

1. Enter the group of gene names to be studied.
2. Enter the group of genes that are the “background” (perhaps all genes for this organism)
3. Select a statistical method for comparison.

For example, let’s find genes that match HLA ::

```
# Get human annotation files.  
wget http://geneontology.org/gene-assessments/goa_human.gaf.gz  
  
# Unzip the file  
gunzip goa_human.gaf.gz  
  
# Match the genes named HLA  
cat goa_human.gaf. | cut -f 3 | grep HLA | sort | uniq | head
```

it produces the following output:

```
HHLA1  
HHLA2  
HHLA3  
HLA-A  
HLA-B  
HLA-C  
HLA-DMA  
HLA-DMB  
HLA-DOA  
HLA-DOB
```

36.4. HOW DO I PERFORM A GENE SET ENRICHMENT ANALYSIS?325

Enter this list into a GO enrichment tool shown here for Panther¹⁰ via the main GO¹¹ website:

Bonferroni count: 8723
[Export results](#)

Displaying only results for Bonferroni-corrected for P < 0.05, [click here to display all results](#)

Homo sapiens (REF)	upload <small>1 (▼ Hierarchy, NEW! ⓘ)</small>				
#	# expected	Fold Enrichment	+/-	P value	
2	2	> 100	+	1.07E-02	
4	2	> 100	+	2.67E-02	
6	2	> 100	+	4.97E-02	
1557	7	.74	9.44	+	1.07E-02
5	2	> 100	+	3.73E-02	
3	2	> 100	+	1.78E-02	
4	2	> 100	+	2.67E-02	
4	2	> 100	+	2.67E-02	
5	2	> 100	+	3.73E-02	
6	2	> 100	+	4.97E-02	
136	7	.06	> 100	+	6.01E-10
167	7	.08	88.01	+	2.43E-09
2687	8	1.28	6.25	+	2.24E-02
99	4	.05	84.83	+	9.76E-04
101	4	.05	83.15	+	1.06E-03
4	2	> 100	+	2.67E-02	
7	3	> 100	+	8.12E-05	
7	3	> 100	+	8.12E-05	

Congrats, you've just completed a gene set enrichment process. Next chapters discuss other alternatives.

¹⁰<http://geneontology.org/>

¹¹<http://geneontology.org/>

Chapter 37

Using the AgriGO server

Authors: Aswathy Sebastian, Istvan Albert

AgriGO stands for - “Go Analysis Toolkit and Database for Agricultural Community” but it has been expanded to support more organisms.

- AgriGo service: <http://systemsbiology.cau.edu.cn/agriGOv2/>
- AgriGO publication: agriGO v2.0: a GO analysis toolkit for the agricultural community, NAR (2017)¹

37.1 Authors note

AgriGO is one of our favorite tools for functional analysis.

It is fairly simple to execute, the visualizations are detailed and nicely drawn. We typically start a functional analysis with an AgriGO visualization.

37.2 How to use AgriGO

AgriGo is a web-based tool for gene ontology analysis. The original goal of the service was to support the functional analysis of agricultural species. The

¹<https://academic.oup.com/nar/article/3796337/agriGO-v2-0-a-GO-analysis-toolkit-for-the>

37.3. WHAT IS THE SINGULAR ENRICHMENT ANALYSIS (SEA) IN AGRIGO?327

site and services that it offers have been useful beyond the original purpose, today a large number of the widely studied genome like mouse; chicken can also be analyzed.

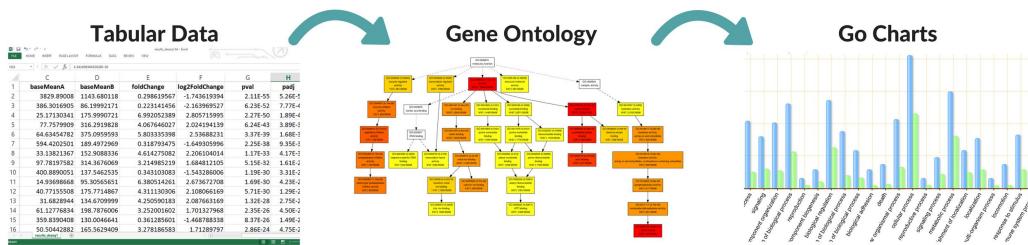


Figure 37.1

Strengths of AgriGO are:

- Usable and straightforward interface.
- Great graphical visualizations.

37.3 What is the Singular Enrichment Analysis (SEA) in AgriGo?

Singular Enrichment Analysis (SEA) in AgriGo can be used to find the enriched Gene Ontology terms in user's list. The input consists of a list of genes. After performing statistical tests with pre-computed or customized backgrounds, the user will be provided with enriched GO terms. These results can be visualized in a GO hierarchical tree structure along with its significance level.

37.4 How to use SEA?

When SEA as selected as the analysis tool, the user is required to specify the species and the gene list in the required format. When species are chosen, allowed forms for gene list will be displayed on the right-hand side box. For example, if **Mus musculus** is selected, then the gene list should be submitted.

The figure shows a screenshot of the AgriGO server interface. The top navigation bar includes links for HOME, ANALYSIS TOOL, SEARCH, DOWNLOADS, MANUAL, and FAQ. Below the navigation bar, there are four main sections:

- 1. Select analysis tool:** This section contains a dropdown menu with the following options: Singular Enrichment Analysis (SEA) (selected), Parametric Analysis of Gene Set Enrichment (PAGE), Transfer IDs by BLAST (BLAST4ID), Cross comparison of SEA (SEACOMPARE), Customized comparison, and Reduce + Visual Gene Ontology (REVIGO). A tooltip for SEA explains it as a traditional and widely used method.
- 2. Select the species:** This section includes a radio button for "Supported species" (selected) and "Customized annotation". A dropdown menu shows "Mus musculus" selected. Below this is a "Query list [Example]" box containing a list of MGI IDs: 2147810, 97268, 97267, 1352494, 98001, 99501, 104597, 102844, 102853, 87958, and 105102. A red arrow points from the "Select species" label to this dropdown.
- 3. Select reference:** This section includes a radio button for "Suggested backgrounds" (selected), "Customized reference [Example]", and "Customized annotated reference". A dropdown menu shows "Mouse Genome Informatics ID" selected. A red arrow points from the "Select reference" label to this dropdown.
- 4. Advanced options (optional):** This section contains two buttons: "Submit" and "Reset".

A red arrow also points from the "Gene list is the specified format" label to the "Query list" box. Another red arrow points from the "Allowed Formats for selected species" label to the "MGI" tooltip in the "Supported species" section.

Figure 37.2

The data file used here can be obtained using the commands below. Here a mouse gene list is selected for AgriGO analysis by using an expression p-value cut off of 0.05.

```
curl -O http://data.biostarhandbook.com/rnaseq/mouse-gene-expression.txt
cat mouse-gene-expression.txt | awk '$2 < 0.05 {print $1}' > genelist.txt
```

Since AgriGo requires mouse gene list to be in 'MGI' format, submit this to the Batch Query in Mouse Genome Informatics² website.

The MGI ids are obtained from here can be pasted in the query list box. Once the reference was chosen, submit the list.

²<http://www.informatics.jax.org/batch>

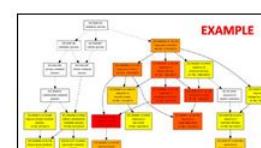
37.5 What are the outputs of AgriGO?

A summary report and a table of enriched GO terms along with their statistical significance will be displayed once the gene list is processed. The site offers alternative options for generating graphical visualizations.

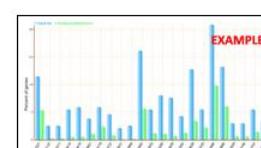
Analysis Brief Summary

Job ID: 656410826 [Useful within 7 days]
 Job Name: 656410826
 Species: *Mus musculus*
 GO type: Completed GO
 Background/Reference: Mouse genome informatics ID
 Annotated number in query list: 67 [[Download](#)]
 Annotated number in background/reference: 35008
 Significant GO terms: 204 [[Details](#)]

Graphical Results 

Select Category: Biological Process Cellular Component Molecular Function
 Advanced Parameter Settings
 Graphic result format: PNG PDF JPEG GIF SVG
 Graphic rank direction: Top to Bottom Left to Right Bottom to Top Right to Left
 Graphic font size (pt): 7 8 9 10 11 12
 

GO flash Chart 

Select Category: Biological Process Cellular Component Molecular Function
 Advanced Parameter Settings
 Bar style: Glass Bar Filled Bar 3D Bar Cylinder Bar
 Query bar color: #1ABCFF Bg/Ref bar color: #66FF33 [HEX format only] [default]
 X legend content: GO annotation GO accession font: 14
 X legend rotation: 300 [270 to 315 is suggested]
 

Detail information 

You can [[Browse in tree traversing mode](#)] [[Browse all GO terms](#)] [[Download](#)] [REVIGO ]
 Or select following significant terms to [Draw graphical results ] [Create bar chart ]

GO term	Ontology	Description	Number in input list	Number in BG/Ref	p-value	FDR
GO:0006936	P	muscle contraction	13	119	3.5e-19	3.4e-16
GO:0003012	P	muscle system process	13	130	1e-18	5e-16
GO:0003015	P	heart process	11	86	4.6e-17	1.1e-14
GO:0060047	P	heart contraction	11	86	4.6e-17	1.1e-14

Figure 37.3

Shown below is the graphical view of enriched GO terms in *Molecular Function* category. Different colors denote the level of significance. Red means highly significant and orange means of medium importance. The significance values are shown inside the boxes.

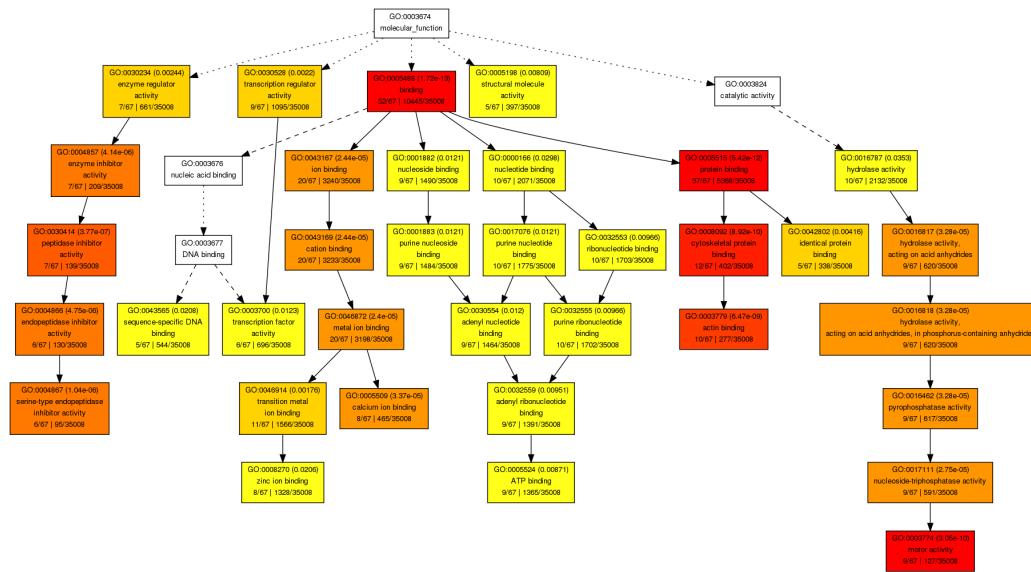


Figure 37.4

37.6 How do I prepare a custom annotation for AgriGO?

To use custom annotation with Agrigo, you need to submit a tab delimited data where one column is an ID and the other is a GO term. For example, to use human annotations, you would need to first get the association file you get

```
# Obtain the human annotation files.
wget http://geneontology.org/gene-associations/goa_human.gaf.gz
gunzip goa_human.gaf.gz
```

Note that you can obtain a protein or gene-name GO pair via:

```
cat goa_human.gaf | grep -v '!' | cut -f 3,5 | head
```

and that looks like:

```
DNAJC25-GNG10 GO:0003924
```

37.6. HOW DO I PREPARE A CUSTOM ANNOTATION FOR AGRIGO?331

```
DNAJC25-GNG10    GO:0007186
NUDT4B           GO:0003723
NUDT4B           GO:0005829
NUDT4B           GO:0008486
```

Now sort this so that it later can be joined with another file.

```
cat goa_human.gaf | grep -v '^!+' | cut -f 3,5 | sort -k 1b,1 > pairs.txt
```

If you had another set of differentially expressed gene names (using the same naming scheme) then you can sort those too then join them on this file. Take for example this enrichment file deseq-output.txt³

```
wget http://data.biostarhandbook.com/ontology/deseq-output.txt
```

Take the first 100 differentially expressed genes and sort them:

```
cat deseq-output.txt | head -100 | sort -k 1,1 > top.txt
```

You can join the two files and cut only the first two columns; this subselects from the pairs file just those lines that appear in the top.txt file. The action before produces the AgriGO input file:

```
join pairs.txt top.txt -t $'\t' | cut -f 1, 2 | head
```

and yes the same term can be there multiple times since each is an association to a function.

```
AARS    GO:0000049
AARS    GO:0001942
AARS    GO:0002161
AARS    GO:0003676
```

You can load the resulting GO annotation file into AgriGO as the new annotation file

³<http://data.biostarhandbook.com/ontology/deseq-output.txt>

```
join -t $'\t' pairs.txt top.txt | cut -f 1,2 > agrigo.txt
```

Chapter 38

Using the g:Profiler server

Quick access:

- g:profiler service: <http://biit.cs.ut.ee/gprofiler>
- g:profiler publication: <https://www.ncbi.nlm.nih.gov/pubmed/27098042>

38.1 What is the g:Profiler?

A useful website sporting a counterintuitive and somewhat confusing naming scheme, the **g:Profiler** is a suite of several tools.

For the full details we recommend consulting the publication g:Profiler – a web server for functional interpretation of gene lists¹ Nucleic Acids Research 2016;

The **g:Profiler** can perform: “functional enrichment analysis and mine additional information. These tools analyse flat or ranked gene lists for enriched features (g:GOSt; g:Cocoa), convert gene identifiers of different classes (g:Convert), map genes to orthologous genes in related species (g:Orth) and find similarly expressed genes from public microarray datasets (g:Sorter). An additional tool g:SNPense maps human single nucleotide polymorphisms (SNP) to gene names, chromosomal locations and variant consequence terms from Sequence Ontology”

¹<https://www.ncbi.nlm.nih.gov/pubmed/27098042>

38.2 What is a standout feature of the g:Profiler?

In our opinion g:Profiler visualizes the evidence code for GO terms in the most transparent way.

As an example load the following human gene-list.txt² into the g:Profiler³ and view the results.

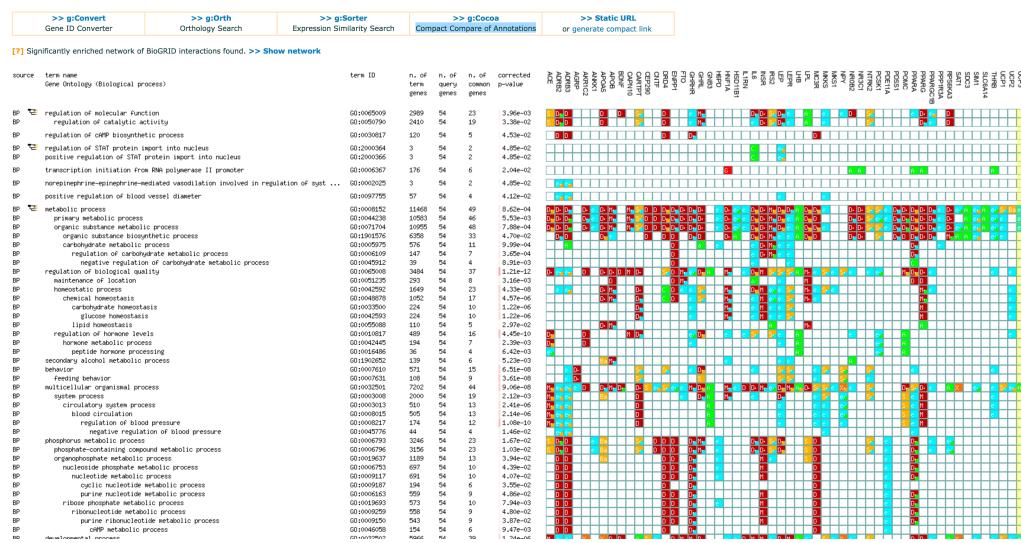


Figure 38.1

38.3 What functionality does the g:Profile have?

38.3.1 g:GOSt

Performs functional enrichment analysis:

²<http://data.biostarhandbook.com/courses/2017-852/media/05/gene-list.txt>

³<http://biit.cs.ut.ee/gprofiler/index.cgi>

g:GOSt performs pathway enrichment analysis and is the central tool of the g:Profiler web server. It maps a user-provided gene list to various sources of functional information and determines significantly enriched pathways, processes, and other annotations.

38.3.2 g:COCOA

COCOA (Compact Compare of Annotations) offers simultaneous enrichment analysis of multiple gene lists:

g:Cocoa provides means to analyze several gene lists at the same time and compare their characteristic enriched terms. This is useful in scenarios where an experimental design involves many comparisons of samples or individuals, or when one wants to compare directly different clusters of genes arising from the analysis. Each gene list is analysed for functional enrichments similarly to g:GOSt and resulting terms are aligned vertically into a matrix highlighting strongest findings for every gene list.

38.3.3 g:Convert

Allows for automatic conversion of gene identifiers.

g:Convert provides a convenient service to translate identifiers (IDs) of genes, proteins, microarray probesets and many other types of namespaces. The seamless translation process works on a mixed set of diverse identifiers and maps these through Ensembl gene identifiers (ENSG) as reference. In cases of multiple identifiers, all relevant combinations are highlighted. At least 13 types of IDs are supported for all of the 213 species available in g:Profiler, and at least 40 types of IDs for more than 50 species.

38.3.4 g:Orth

Maps related genes across species

g:Orth allows the user to map a list of genes of interest to homologous genes in another related organism. Many experiments are conducted in model organisms and knowledge from such experiments is transferred to other organisms to compare or complement previous findings. g:Orth uses g:Convert to map gene IDs to Ensembl ENSG identifiers. Further mapping to orthologous genes in other organisms is also based on Ensembl data. We provide cross-references between all organisms in g:Profiler. Queries are limited to tiers according to classes of species (animals, plants, fungi).

38.3.5 g:Sorter

Findins similar genes in transcriptomics data

g:Sorter is a tool for finding lists of co-expressed genes from public transcriptomics datasets. Thousands of microarray and RNA-seq experiments have been conducted in the past decades. The majority of published studies have been accumulated in databases like ArrayExpress and Gene Expression Omnibus. We have downloaded 7878 datasets for 18 species from ArrayExpress and provide gene co-expression similarity searches using six most common statistical methods. The datasets can be searched rapidly with keywords. The input of g:Sorter is a single gene and a dataset of interest and the result is a sorted list of genes that are similarly expressed with the gene of interest. These lists can be integrated into functional enrichment analysis. For comprehensive global gene expression similarity queries, as well as support for more species and platforms we suggest to use Multi Experiment Matrix (MEM) tool

38.3.6 g:SNPense

SNP identifier mapping

With the rapid growth of whole genome sequencing technology, researchers are uncovering extensive genetic variation and large collections of known SNP are available for human and other species. In order to easily map SNP identifiers (e.g. rs4244285) to gene names, chromosomal coordinates and retrieve their functional consequences we now provide a new service called g:SNPense. Information about genome variants is retrieved from dbSNP and mapped to NCBI Genes. Potential functional consequences of the variants are retrieved from Ensembl Variation data and grouped into 35 Sequence Ontology terms of various severity. g:SNPense is a potential entry point to g:GOST and its functional annotation pipeline and enrichment analysis.

38.4 How to use g:profiler at the command line

An alternative to the web interface is the command line client to g:profiler. First install the command line client:

```
# Install the client.
pip install gprofiler-official

# Get help on the client parameters.
gprofiler_cli.py --help

# Example run from the documentation.
gprofiler_cli.py -o scerevisiae "swi4 swi6"
```

Let's run the g:profiler on the up regulated genes from the Zika redo chapter. Get the data

```
wget http://data.biostarhandbook.com/redo/zika/zika-up-regulated.csv
cat zika-up-regulated.csv | cut -f 1 -d , | head
```

produces the list of up regulated genes

```
gene
ELAVL3
FAM53C
SLC22A17
WDFY1
TBL1XR1
SON
EIF4G2
SOGA3
SDCBP
```

The file above has too many candidate genes. Usually this means that the output will be overly complicated, will pull in too many alternative explanations. Biologists tend to subfilter the results to let them isolate one particular function. One typical filtering is that on the magnitude of change. For example, we may want to select genes that change more than 4 fold (this would correspond to a log2 of 2). In addition the g:profiler client requires a list of symbols, all on the same line so will replace the new line with a space character.

```
cat zika-up-regulated.csv | grep -v gene | awk -F , '$6 > 2 { print $1 }' | tr "\n"
```

like so:

```
NOG SNORA18 SESN2 JDP2 ASNS RAB3IL1 ...
```

run the profiler

```
gprofiler_cli.py -o hsapiens -q query.txt > results.txt
```

The resulting file contains a large number of entries, the first few columns, when filtered for biological process are:

```
cat results.txt | cut -f 9-12 | grep BP | head
```

produces:

```
GO:0034976 BP 1 response to endoplasmic reticulum stress
GO:0070059 BP 1 intrinsic apoptotic signaling pathway in response to endoplasmic reticulum stress
GO:0036003 BP 1 positive regulation of transcription from RNA polymerase II promoter
```

GO:0006986 BP 1 response to unfolded protein
GO:0035966 BP 1 response to topologically incorrect protein
GO:0030968 BP 1 endoplasmic reticulum unfolded protein response
GO:0010033 BP 1 response to organic substance
GO:0034620 BP 1 cellular response to unfolded protein
GO:0035967 BP 1 cellular response to topologically incorrect protein
GO:0071310 BP 1 cellular response to organic substance

Chapter 39

Using the DAVID server

Authors: Aswathy Sebastian, Istvan Albert

The word DAVID is an acronym for “The Database for Annotation, Visualization and Integrated Discovery”

- DAVID service: <https://david.ncifcrf.gov/home.jsp>
- DAVID publication: <https://www.ncbi.nlm.nih.gov/pubmed/19131956>

39.1 Authors note

DAVID is the tool we love to hate.

Amongst all functional analysis tools we feel DAVID is the wonkiest. It is awkward to use, has a repetitive, redundant and circularly confusing naming scheme. Within five clicks we lose our way and need to start a new.

Yet biologists **LOVE** DAVID.

There is something about the way this tool works and how it presents information that matches the way life scientists are trained to think. It “speaks” to biologists in a way no other

tool does. It is also a tool with the most “generous” enrichment protocols.

DAVID will dole out seemingly relentless sweet suggestions of what the data is about. DAVID will not limit your imagination with mundane statistics. As a matter of fact we believe that DAVID does not even correct for multiple comparisons - perhaps one of the cardinal sin of statistics. On the other hand, it is not yet proven that the Gene Ontology should be described with statistical models in the first place! Suggestion by DAVID when coupled with biological intuition and other supporting information can help scientists discover phenomena that other approaches miss. At the same time though, it can also lead the unwitting deep down into the tar pits of irreducibility.

Try DAVID and see how you like it. Did we mention that we love to hate it? When we want to “get rid” of someone asking too many questions about functional analysis we send them to DAVID. Nobody ever comes back.

39.2 What is DAVID?

DAVID is a functional enrichment analysis tool that can be used to understand the biological relevance of a list of genes. The tool allows users to:

- Identify functionally essential genes in a gene list.
- Get the enriched biological terms in a gene list.
- Obtain the biological processes that a set of up-regulated and down-regulated genes are involved in.

The tool is a valuable resource - but like many other scientific software, it suffers from very confusing interface elements and redundant windows that pop over others. It is very easy to get disoriented while using it.

39.3 What does DAVID do?

When presented a gene list like the one shown below

Tmem132a
Myl3
Myl4
Tnnt2
Hspb7
Tnni3
Actc1
...

DAVID will generate functional interpretations:

- 14 genes (Myl3, Myl4, Tnni3,...) from this list are involved in *Cardiac Muscle Contraction*
- 4 genes (Actc1, Myh6, Myh7, Tnnt2) have the role of *ATPase Activity*

and so on. The value added by the tools is that it helps find groups of genes that share specific properties that might be interesting to a life scientist.

39.4 What are the different steps in a DAVID analysis?

A typical gene enrichment and functional annotation workflow in DAVID has the following steps:

1. Load a gene list into the site.
2. Explore details through the annotation table, chart or clustering reports.
3. Export and save the results.

39.5 How do I start an analysis with DAVID?

Every analysis in DAVID starts with the submission of a gene list.

A gene list is a selected set of genes from an experiment that you might be interested in learning more about. This gene list will be evaluated against a so-called “background” list to detect attributes that are only present in the gene list.

Often, the background is taken as all genes of an organism, but, in some cases, it is more appropriate to use a different gene set as background. For example, if the list of interesting genes was taken from a microarray experiment, the background should be only those genes queried by the microarray. Gene lists from sequencing assays generally can use all the organism's genes as the background.

To start an analysis, users can either click on the 'Start Analysis' tab or choose one of the tools from 'Shortcut to DAVID Tools' tab. Both will take you to the upload wizard:

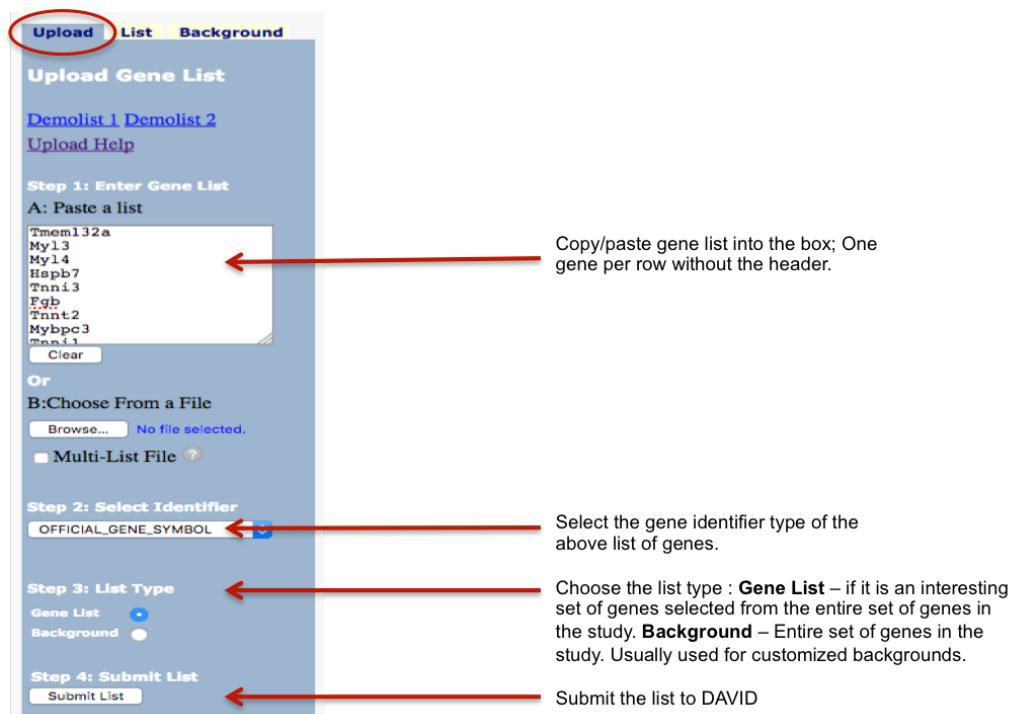


Figure 39.1

An example gene list can be obtained using the commands below. Here a mouse gene list is selected for DAVID analysis by using an expression p-value cut off of 0.05.

```
curl -O http://data.biostarhandbook.com/rnaseq/mouse-gene-expression.txt  
cat mouse-gene-expression.txt | awk '$2 < 0.05 {print $1}' >genelist.txt
```

Once the list is submitted, a selection of the input species might be needed in case there are ambiguities, and the input gene names may refer to multiple species.

A note on what is happening behind the scenes when a gene list is submitted:

- Each input gene id gets converted into a so-called *DAVID Gene ID*, a unique identifier specific to DAVID. All the subsequent analysis will be applied to the DAVID IDs in the selected gene list.
- The DAVID Gene ID connects the input genes across all other databases in DAVID. DAVID supports nearly 30 different input id types and a broad set of annotation categories.

Once the gene list is submitted, you will need to explore the submitted gene list with DAVID tools and their results are named somewhat redundantly and confusingly (at least we have a hard time remembering which one does what):

- Functional Annotation.
- Gene Functional Classification.

39.6 What is the Functional Annotation Tool?

These tools assist users in the biological interpretation of their data. It maps the genes to annotation content from different databases DAVID computes the enriched functionality of the gene list.

The results of a Functional Annotation process is a Functional Annotation Summary where the user has options to further select annotation categories of interest and explore them in detail.

39.7 What is the Functional Annotation Summary?

The selections of the Functional Annotation Tool and Annotation Summary page are shown below:

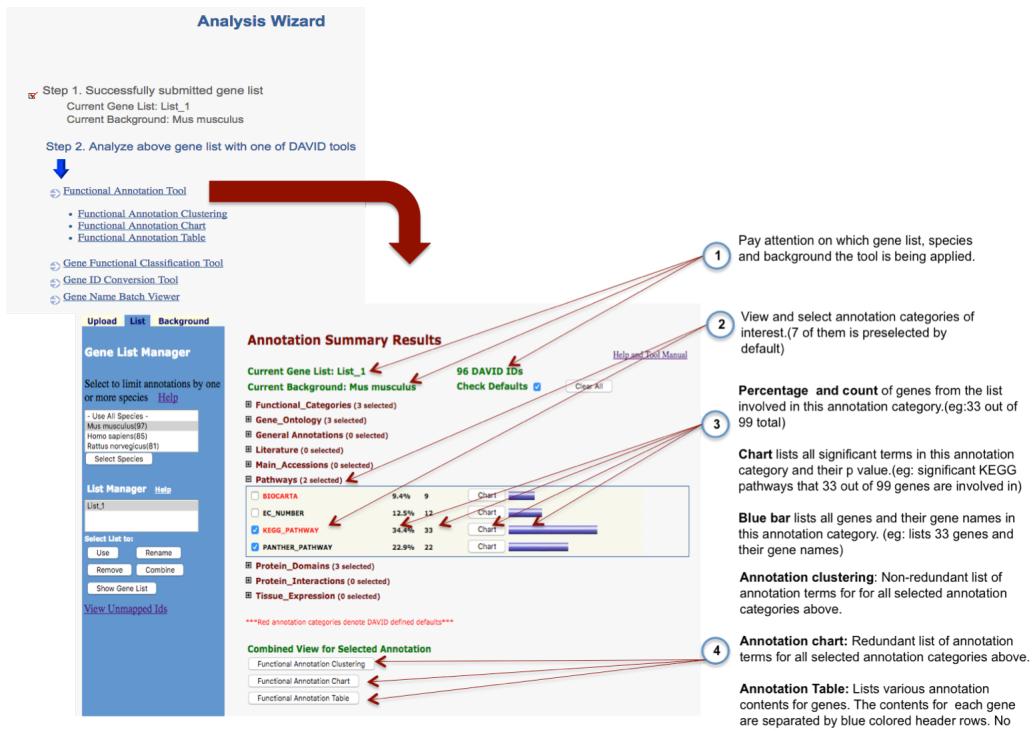


Figure 39.2

Expanding on each selected category gives the annotation details of the genes involved in it.

The combined view allows you to explore the combined annotation content of all selected categories.

39.8 How do I explore the Functional Annotation Summary?

The selected functional categories of interest in the Functional Annotation Summary page can be analyzed through any of the following reports:

1. **Functional Annotation Chart:** A view focused on annotation terms that lists all enriched annotation terms and their associated genes.
2. **Functional Annotation Clustering:** Groups similar annotation terms into clusters based on their similarity and the genes they share in common.
3. **Functional Annotation Table:** Gene-centric view that lists all genes and their associated annotation terms (selected only). No statistics are provided here.

39.9 What is a Functional Annotation Chart ?

The functional annotation chart is a tabular form of all enriched annotation terms associated with a gene list. The enriched annotation terms presented here can be redundant. The enrichment P-value is the so-called *Ease Score*, calculated via Fisher's Exact test.

In the example shown here, 233 annotation terms are associated with 96 input genes. 18 of these 96 genes are involved in muscle protein functionality. Clicking on the blue bar lists these genes. The related terms (RT) search shows other annotation terms that these 18 genes are associated with. In other words, RT lists all annotation terms above a certain similarity threshold that are shared by these genes.

39.10 What is Functional Annotation Clustering?

Annotation terms associated with genes can be redundant. DAVID measures the degree of similarity between these terms using kappa statistics based on

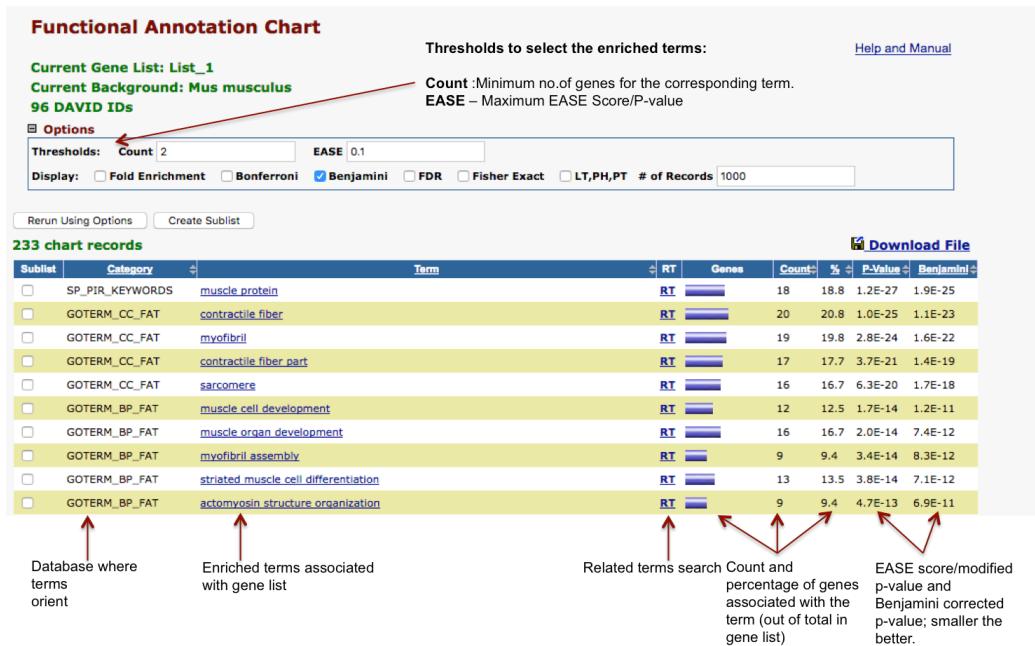
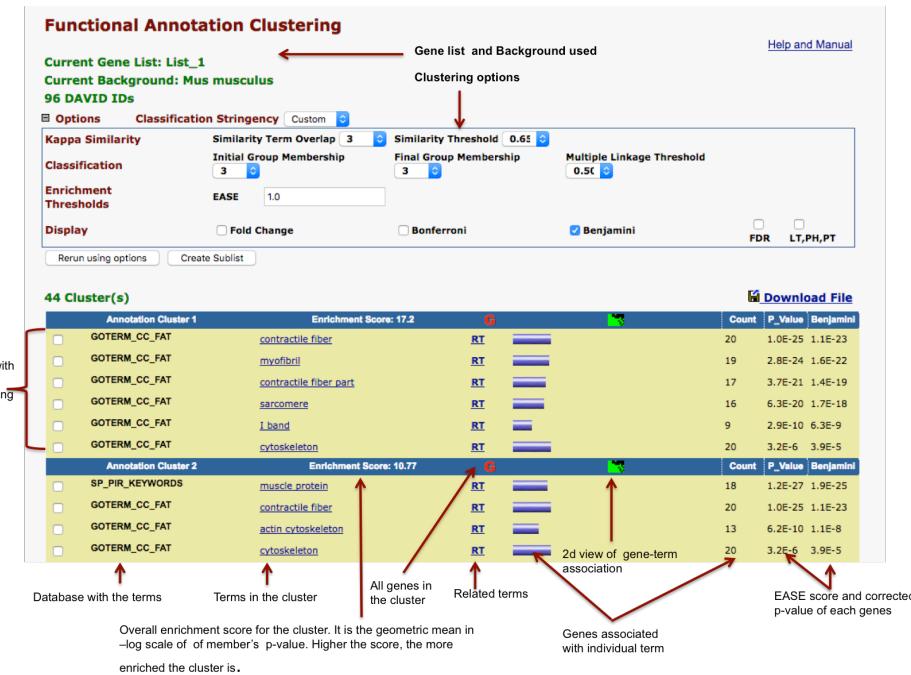


Figure 39.3

the number of genes they share. All the terms above a certain similarity threshold are then grouped together using a clustering algorithm. Thus each annotation cluster groups terms with similar biological meaning.

The clustering report below shows that 233 enriched annotation terms associated with 96 DAVID input ids are clustered into 44 groups.



Cluster1 in the example shown here has six annotation terms that share the associated genes. The relationship between the genes and these annotation terms is shown in the image below. The green area represents the sharing of genes by these annotation terms. The black area denotes that there is no association between the terms and the genes. A large green area means that more genes are shared, implying a greater similarity between these terms. The terms are thus grouped together in a single cluster.

39.11 What is a Functional Annotation Table?

The Functional Annotation Table is a third form of reporting the annotation content (selected in annotation summary) of a gene list. It provides a gene-centric view by listing the genes and their associated terms. No statistics are reported on this page. A blue header row separates each gene.

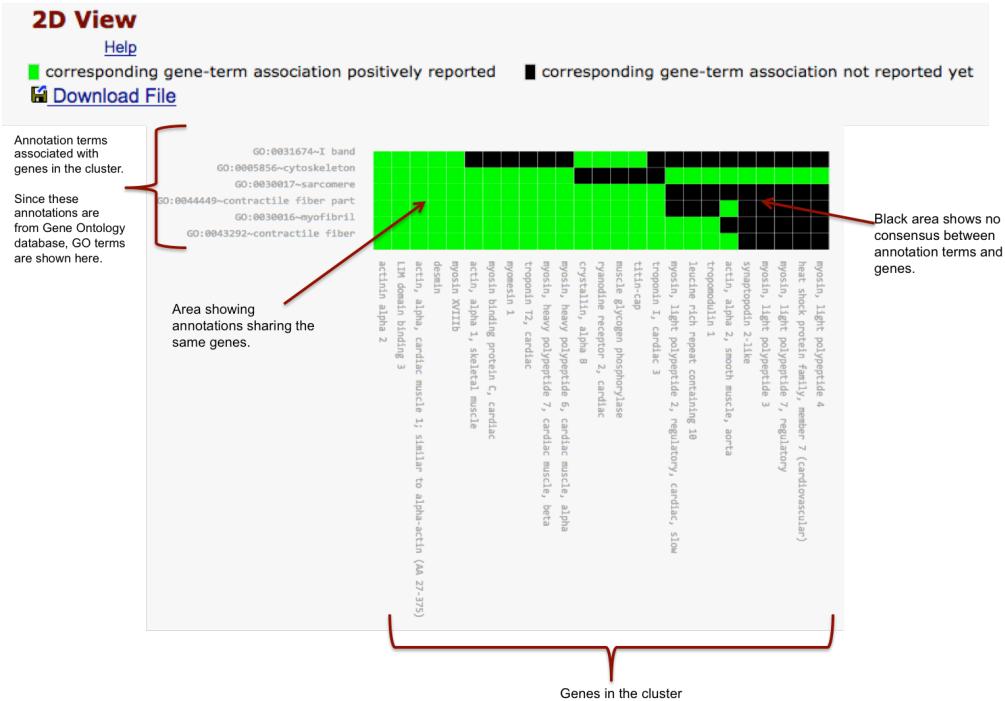


Figure 39.4

Functional Annotation Table

Current Gene List: List_1
Current Background: *Mus musculus*
871 DAVID IDs

[Help and Manual](#)

[Download File](#)

94 record(s)

Adprhl1	ADP-ribosylhydrolase like 1	Related Genes	<i>Mus musculus</i>
GOTERM_BP_FAT	protein amino acid de-ADP-ribosylation,		
GOTERM_MF_FAT	magnesium ion binding, ADP-ribosylarginine hydrolase activity, hydrolase activity, hydrolyzing N-glycosyl compounds, ion binding, cation binding, metal ion binding,		
INTERPRO	ADP-ribosylation/Crystallin J1, ADP-ribosylarginine hydrolase,		
PIR_SUPERFAMILY	PIRSF016939:ADP-ribosylarginine hydrolase, PIRSF016939:ADP_ribslarg_hdr,		
SP_PIR_KEYWORDS	hydrolase,		
UP_SEQ_FEATURE	chain:[Protein ADP-ribosylarginine] hydrolase-like protein 1,		
Bcl11a	B-cell CLL/lymphoma 11A (zinc finger protein)	Related Genes	<i>Mus musculus</i>
GOTERM_BP_FAT	cell activation, immune system development, leukocyte differentiation, transcription, hemopoiesis, lymphocyte differentiation, B cell differentiation, T cell differentiation, T cell activation, B cell activation, leukocyte activation, regulation of transcription, lymphocyte activation, hemopoietic or lymphoid organ development,		
GOTERM_MF_FAT	transcription cofactor activity, transcription corepressor activity, transcription factor binding, zinc ion binding, transcription repressor activity, transcription regulator activity, ion binding, cation binding, metal ion binding, transition metal ion binding,		
INTERPRO	Zinc finger, C2H2-type, Zinc finger, C2H2-type/integrase, DNA-binding, Zinc finger, C2H2-like,		

Figure 39.5

39.12 What is the Gene Functional Classification Tool?

DAVID has yet another tool to help the user interpret the biological meaning of their data: the Gene Functional Classification Tool. This tool provides a view similar to annotation clustering, except that the genes are clustered instead of the annotation terms.

The Gene Functional Classification tool reports all the functionally related genes in the user's gene list. Only genes not mapped to any of the functional categories are omitted from the table.

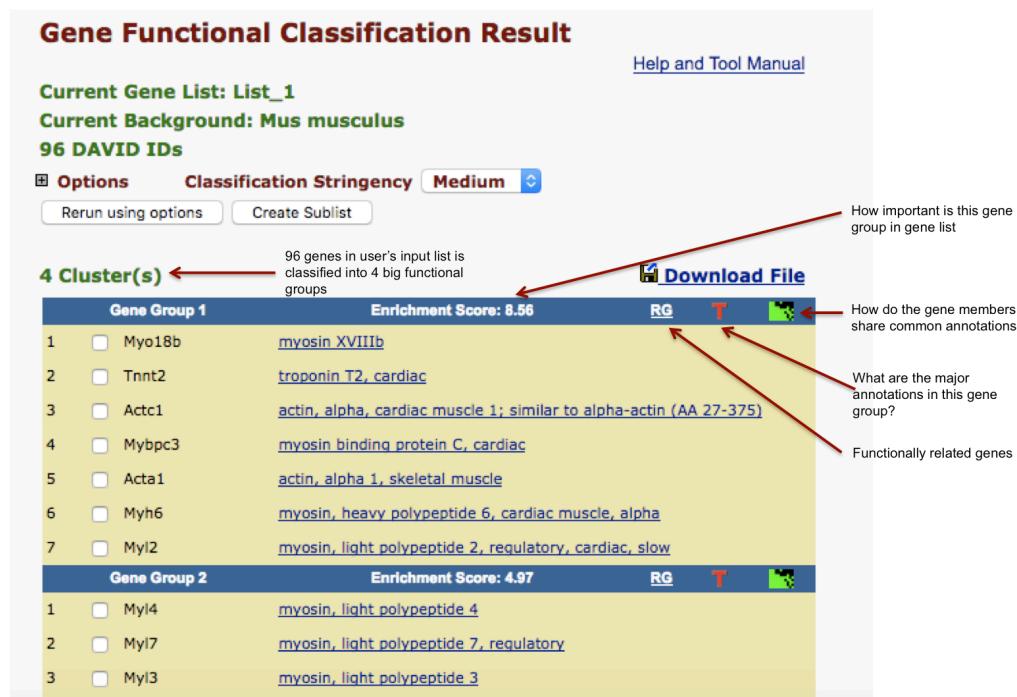


Figure 39.6

39.13 What is an EASE Score?

In the DAVID annotation system, Fisher's Exact Test is used to measure gene-enrichment in annotation terms. The Ease score is a slightly modified

Fisher's Exact p-value. The smaller the value, the more enriched the associated term. Consider,

- a = number of genes in the user's list that belong to a particular pathway, Z.
- b = total number of genes in the user's list.
- A = number of genes in the background that belong to pathway Z
- B = total number of genes in the background.

Fisher's Exact Test would compare a/b genes in the user's list with A/B genes in the background to see if the genes in the user's list belong to pathway 'Z' by anything more than random chance.

DAVID compares $(a-1)/b$ with the A/B in Fisher's Exact test to check significance. The p-value obtained is reported as the EASE score.

39.14 What is the Kappa statistic?

The clustering algorithm in DAVID is based on the hypothesis that similar annotations should have similar gene members. It uses the Kappa statistic to measure the degree of shared genes between two annotations. This is followed by heuristic clustering to group similar annotations according to Kappa values.

Cohen's kappa coefficient (k)¹ measures the degree of agreement between qualitative items. DAVID uses kappa values to estimate the functional relationship between gene pairs.

A higher Kappa value in DAVID means that genes are more similar in their annotation content. A zero or a negative value for Kappa indicates that genes share no functionality.

39.15 What does the Gene ID Conversion Tool do?

This tool enables the conversion of an accession number from one specification to another. There are around 30 different supported id types, which makes

¹https://en.wikipedia.org/wiki/Cohen%27s_kappa

this tool quite useful. The gene id conversion page and the results page are shown below.

The screenshot shows the 'Gene Accession Conversion Tool' interface. On the left, two red brackets group sections: 'Summary' (containing 'Conversion Summary' and 'Total Unique User IDs: 10') and 'Possible choices for ambiguous genes' (containing 'Summary of Ambiguous Gene IDs' and 'All Possible Sources For Ambiguous IDs'). Below these are four red arrows pointing to specific rows in the main table:

- User's list:** Points to the first row of the table, which contains Myl4 and its details.
- Converted ids:** Points to the second row, Tnni1.
- Species of converted ids:** Points to the third row, Myl3.
- Gene names of converted ids:** Points to the fourth row, Tnnt2.

Conversion Summary			Submit Converted List to DAVID as a Gene List		Submit Converted List to DAVID as a Background	
ID Count	In DAVID DB	Conversion	From	To	Species	David Gene Name
10	Yes	Successful	Myl4	ENSMUSG00000061086	Mus musculus	myosin, light polypeptide 4
0	Yes	None	Tnni1	ENSMUSG00000026418	Mus musculus	troponin I, skeletal, slow 1
0	No	None	Myl3	ENSMUSG00000059741	Mus musculus	myosin, light polypeptide 3
0	Ambiguous	Pending	Tnnt2	ENSMUSG00000026414	Mus musculus	troponin T2, cardiac
Total Unique User IDs: 10			Tnni3	ENSMUSG00000035458	Mus musculus	troponin 1, cardiac 3
Summary of Ambiguous Gene IDs			Fgb	ENSMUSG00000033831	Mus musculus	fibrinogen beta chain
All Possible Sources For Ambiguous IDs			Actc1	ENSMUSG00000068614	Mus musculus	actin, alpha, cardiac muscle 1; similar to alpha-actin (AA 27-375)
Ambiguous ID			Mybpc3	ENSMUSG00000002100	Mus musculus	myosin binding protein C, cardiac
Possibility			Hspb7	ENSMUSG0000006221	Mus musculus	heat shock protein family, member 7 (cardiovascular)
Convert			Tmem132a	ENSMUSG00000024736	Mus musculus	transmembrane protein 132A

Figure 39.7

39.16 What are some pros and cons of DAVID?

39.16.1 Pros

- The database in DAVID provides the user with “many interesting” results. DAVID is very well suited for exploratory analyses as it does not filter the results.
- Since it is an online resource, no installation is required.
- The statistical model is simple and straightforward.
- Every tool can be used independently of another. In other words, you don’t need to follow a set series of actions to get the results.

39.16.2 Cons

- All results are in “flat” tabular form. No visualizations are provided.
- It’s hard to gain a higher level overview of the relationship between the terms that are enriched.

- Every click in DAVID opens a new window. You can feel lost after just a few clicks.
- DAVID presents the user with many different views of the same result with the aim to help better interpret the data. However, this can be very confusing; sometimes it feels like running in circles.

Chapter 40

Using the goatools package

`goatools` was originally designed as a Python package to be used from within other Python programs. The package also supports command line usage and offers utility in form of standalone programs that get installed when you install `goatools`. The library has been published as GOATOOLS: A Python library for Gene Ontology analyses¹

- `goatools` code: <https://github.com/tanghaibao/goatools>
- `goatools` publication: <https://www.nature.com/articles/s41598-018-28948-z>

40.1 Install `goatools`

```
source activate bioinfo
conda install pygraphviz
pip install goatools pydot statsmodels fisher

# Make a directory for reference data.
mkdir -p refs

# Obtain the gene ontology term definitions.
```

¹<https://www.nature.com/articles/s41598-018-28948-z>

```
# Gets the file only once (-nc) and places into the prefix (-P)
wget -nc -P ~/refs http://geneontology.org/ontology/go-basic.obo
```

40.2 Plot GO terms

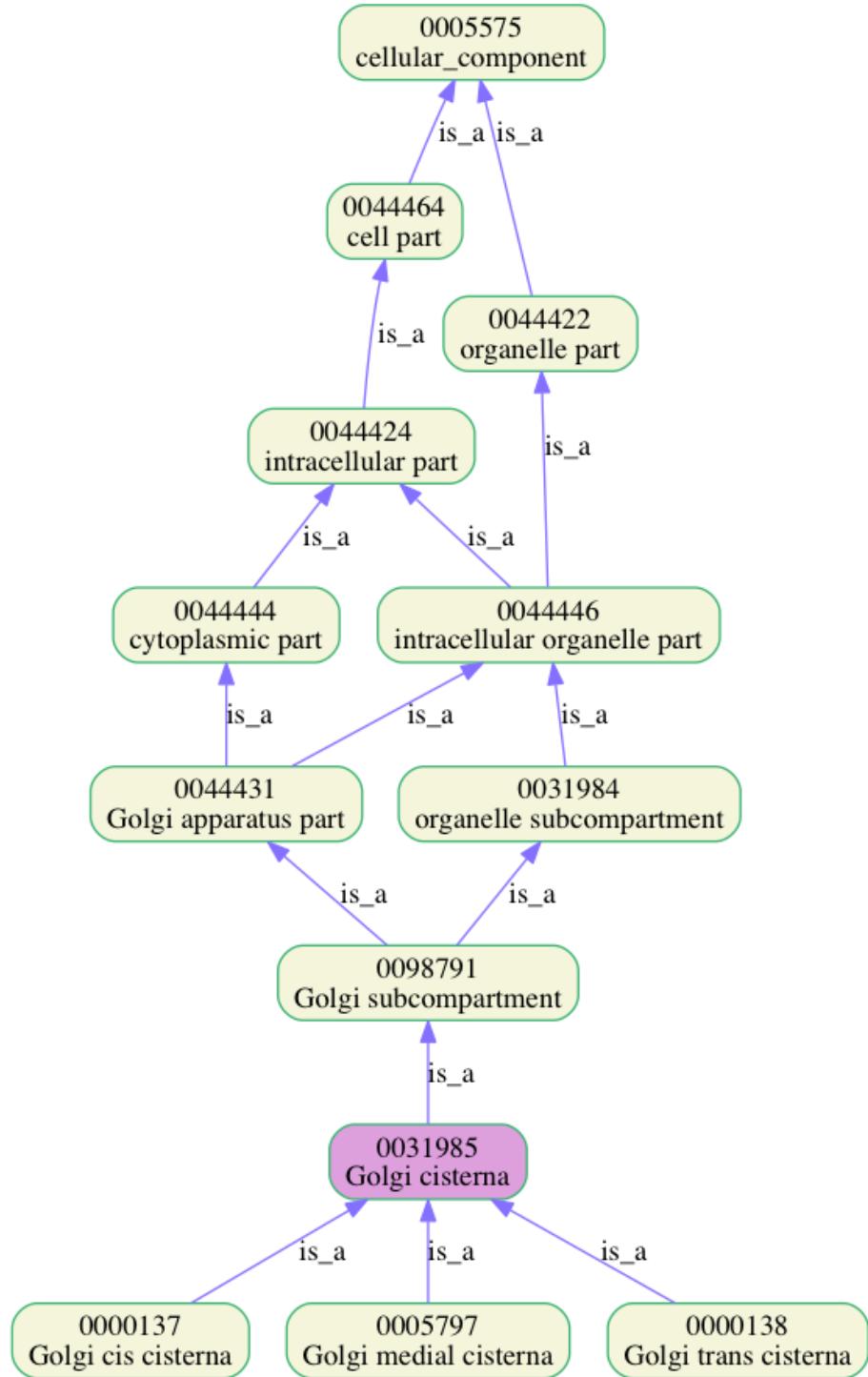
There are several levels of visualizations that you might need, even when wanting to see a single term in context. The `goatools` package offers various visualization capabilities. More examples can be seen at:

- https://github.com/tanghaibao/goatools/blob/master/doc/md/README_go_plot.md

below we reproduce a few examples from the page above:

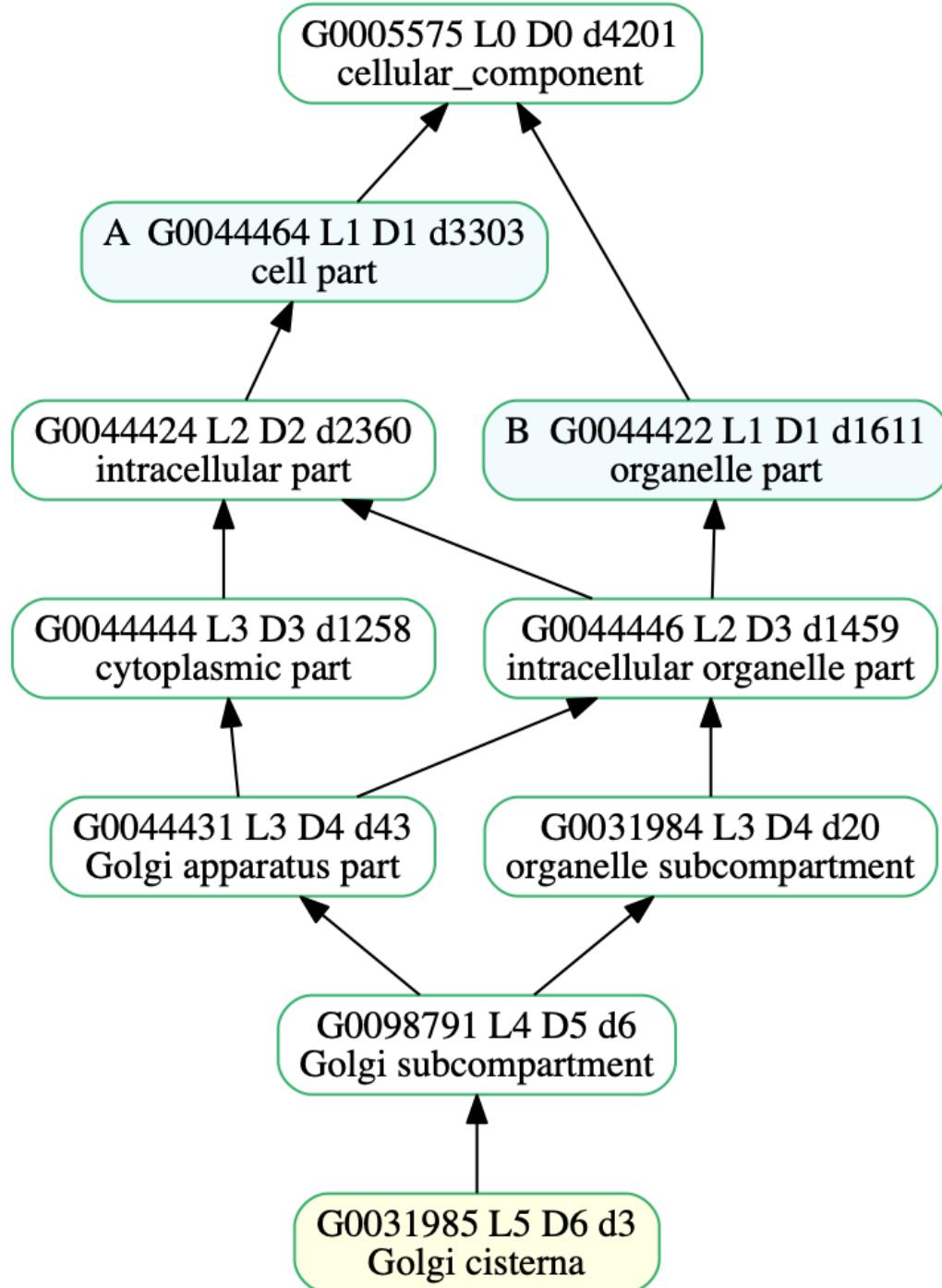
```
# Plot a GO term. Creates the GO_lineage.png file.
plot_go_term.py --term GO:0031985 ~/refs/go-basic.obo
```

Will render an image like:



There is a second way to plot the same information.

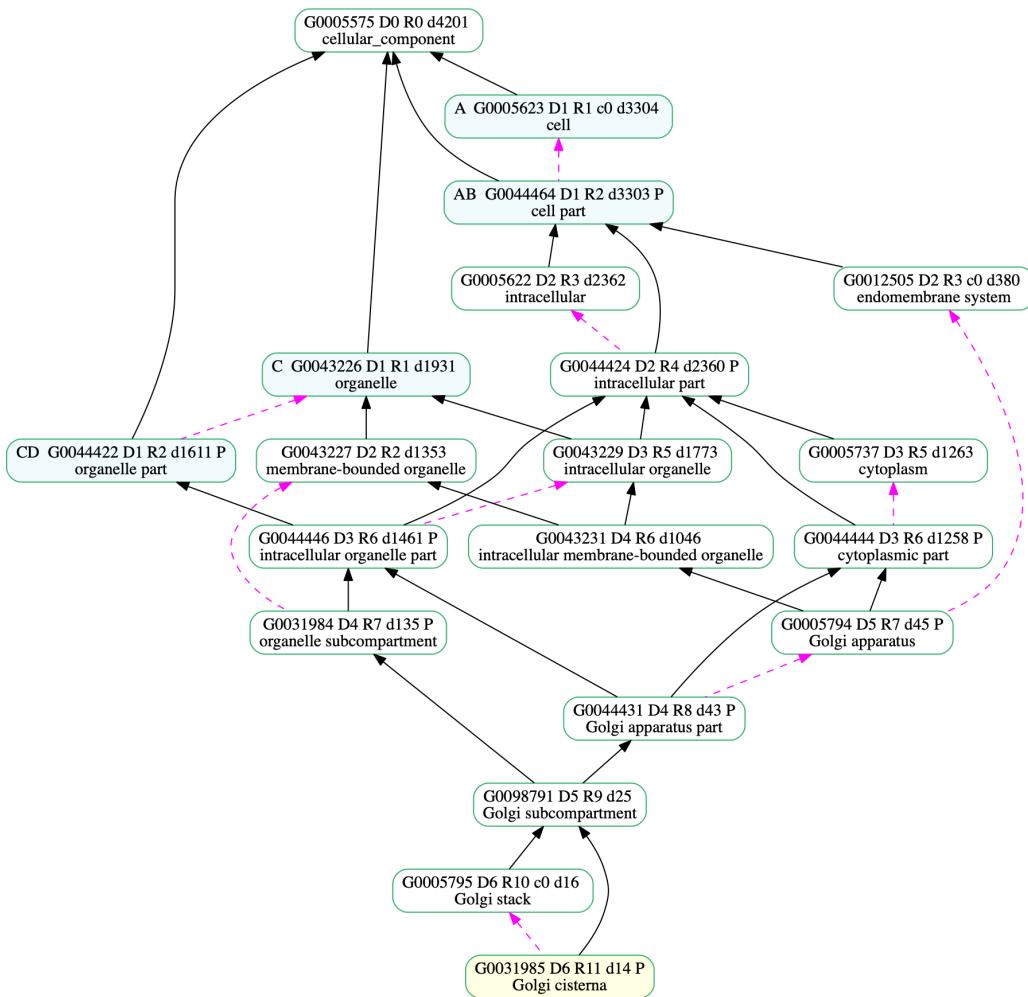
```
go_plot.py --obo ~/refs/go-basic.obo -o output.png GO:0031985
```



Add the `-r` flag to plot all relationships:

```
go_plot.py --obo ~/refs/go-basic.obo -r -o output.png GO:0031985
```

will now add even more context:



The `goatools` offer a variety of options for coloring and displaying the nodes in your image. When it comes to publishing your results, this package can help you produce compelling plots that show the various functional terms in the proper context.

40.3 Finding enrichment with `goatools`

Suppose you have a list of protein or gene ids and you wanted to know what function is representative of them. For example, if we were to take the five most annotated genes in GO is there something common about them? Also, ask yourself should there be anything in common about them? We can find enrichment for a group of genes with `goatools` but the process is a little more tedious than you may anticipate.

You see the `goatools` package suffers from what I call “over-engineering”. It is a common disease that affects the best and the brightest of programmers even more so than beginners. The design of the library is overcomplicated, the examples in the manual lack in specificity. What is a population file, what is an association file, what is a study file? It is a shame, the package has been in development for almost a decade, and clearly, the programmers that work on it are quite talented. Still, curiously simple features and explanations are missing.

For example, the enrichment tool cannot operate on the basic data format that GO is distributed as. Think about that for a second. The Nature Methods journal published a paper on GO data processing and visualization, one essential component of that software cannot process a standard GO annotation file. To use `goatools` on the “official” data format, the dear readers would first need to transform the GAF file into a so-called association file. We’ll do that here:

```
# Get the human gene annotations.
wget -nc -P ~/refs http://geneontology.org/gene-associations/goa_human.gaf.gz
gunzip ~/refs/goa_human.gaf.gz
```

We now need to transform this `gaf` file into an association file where row contains all the GO terms for an id, like so:

`id GO:1;GO2;GO10`

the following `awk` script will do that:

```
# Simplify the data into two columns first, on ID and GO term per line.
cat ~/refs/goa_human.gaf | grep -v ! | cut -f 3,5 > pairs.txt
```

```

# Linearize the file with one ID and multiple GO terms on each line.
cat pairs.txt | awk '{if(a[$1])a[$1]=a[$1]";"$2; else a[$1]=$2;} END {for (i in a) pr
```

```

# Let's consider the population (background) as all of the existing products.
cat pairs.txt | cut -f 1 > population

# We want to know what functions characterize the five most annotated genes.
# This takes a bit of work, we need to squeeze the spaces and find the right column
cat pairs.txt | cut -f 1 | sort | uniq -c | sort -rn | tr -s ' ' | cut -f 3 -d ' ' |

```

The study file contains:

```
TP53
EGFR
GRB2
UBC
CTNNB1
```

We now have all the data, the association, the background and the study file that contains the ten most annotated gene names. Finally, let's get the answer, what are common functions that these genes have? Lets run the `find_enrichment.py` script:

```
find_enrichment.py study population association --obo ~/refs/go-basic.obo > results.txt
```

The file has substantial information, at the beginning it states:

```
13 GO terms found significant (< 0.05=alpha) after multitest correction: local bonferroni
```

Let's visualize these terms.

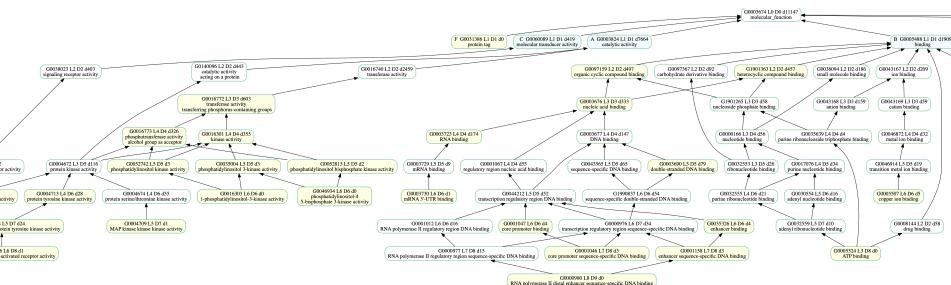
```

# Subselect molecular function (MF).
cat results.txt | grep GO: | grep MF | cut -f 1 > terms.txt

# Plot the terms
go_plot.py --obo ~/refs/go-basic.obo --go_file terms.txt

```

Produces a large image (we show it cropped, here):



It has many functions. Note how we only had five genes under study (though these are the most annotated genes) The `results.txt` file has 1206 functions just for molecular function. Out of these those that contain the word binding are:

- regulation of euchromatin binding
- positive regulation of DNA-binding transcription factor activity
- positive regulation of core promoter binding
- regulation of core promoter binding
- regulation of DNA-binding transcription factor activity
- regulation of chromatin binding
- positive regulation of transcription regulatory region DNA binding
- nucleotide-binding oligomerization domain containing signaling pathway
- nucleotide-binding domain, leucine rich repeat containing receptor signaling pathway
- regulation of transcription regulatory region DNA binding
- positive regulation of DNA binding
- regulation of DNA binding
- positive regulation of binding
- protein phosphatase binding
- phosphatase binding
- protein kinase binding
- kinase binding
- enzyme binding
- disordered domain specific binding
- ubiquitin protein ligase binding
- ubiquitin-like protein ligase binding
- chromatin binding
- signaling receptor binding
- protease binding
- protein domain specific binding

RNA polymerase II transcription factor binding
epidermal growth factor binding
neurotrophin TRKA receptor binding
neurotrophin TRK receptor binding
cadherin binding
alpha-catenin binding
neurotrophin receptor binding
TFIID-class transcription factor complex binding
I-SMAD binding
insulin receptor substrate binding
RNA polymerase II basal transcription factor binding
RNA binding
...

The results indicate that the selected genes have all have roles in regulation via DNA binding.

Chapter 41

Using the ErmineJ tool

ErmineJ¹ is a functional analysis tool that can be used to determine the functional categories or biological pathways that are enriched in a list of genes generated by an experiment. The gene list will be tested with a reference set of Gene Ontology (GO) terms to identify relevant biological pathways.

- ErmineJ code: <http://erminej.chibi.ubc.ca/download/>
- ErmineJ publication: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1310606/>

41.1 Authors note

There was a time when ErmineJ was the “GO to” choice for GO analysis (see how easy it is to make puns using the word “GO”?). Sí, AmiGO!

Those times are past, but it still a unique program that it runs offline on your computer.

Besides, it is one of the few tools that account for “multifunctionality” (see below) and offers a visualization that includes tiny heatmaps for each entry. Quite the unique feature! Recently we

¹<http://erminej.chibi.ubc.ca/download/>

found ourselves using ErmineJ a lot less, we still include it here as a useful alternative.

41.2 How to use ErmineJ

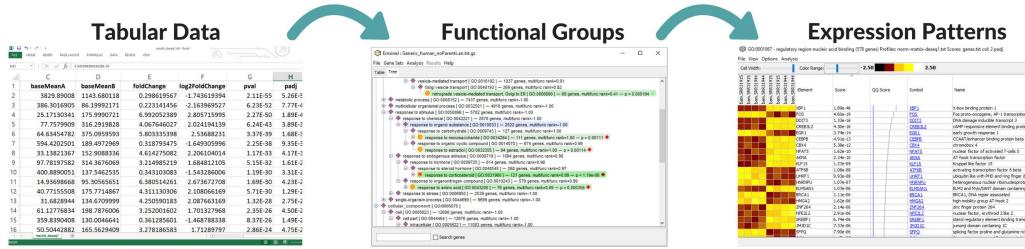


Figure 41.1

In the simplest of terms, you start with a text file that contains genes on each row and measurements in each column. ErmineJ's purpose is to find the common functions across the genes that are above a certain score.

41.3 What type of analyses does ErmineJ offer?

ErmineJ offers different ways to analyze the gene sets. These include,

- **ORA** - Over-Representation Analysis (finds enriched functions).
- **GSR** - Examines the distribution of gene scores in each set.
- **CORR** - Uses correlation of expression profiles.

41.4 How is ErmineJ different from other options?

- ErmineJ runs on Java, hence it is easy to install and run.
- It runs on a desktop (not browser based).
- It has a simple, easy-to-use graphical user interface.

- ErmineJ provides data visualization.
- It computes a multi-functionality score.

41.5 What are the input files in ErmineJ?

The main input files for ErmineJ are

1. The annotation file. These specify the gene-to-function information.
2. The gene score file. The gene score file specific the scores assigned to genes.
3. The optional expression data. This file assigns numbers across multiple conditions for each gene.

41.6 What are the annotation files?

The annotations represent the names and relationships among GO terms. ErmineJ comes with a GO XML file containing the names and relationships among GO terms. The ErmineJ website offers options to download annotations for many commonly studied organisms (see details below).

- Download ErmineJ annotation files²

There quite many files there. Our examples work on with the generic human and generic mouse files, those can be downloaded from the command line with:

```
curl -O http://chibi.ubc.ca/microannots/Generic_human_noParents.an.txt.gz
```

and

```
curl -O http://chibi.ubc.ca/microannots/Generic_mouse_noParents.an.txt.gz
```

When you start ErmineJ, you will need to load one of these files into the system. Only one annotation can be active at any given moment though.

²<http://chibi.ubc.ca/microannots/>

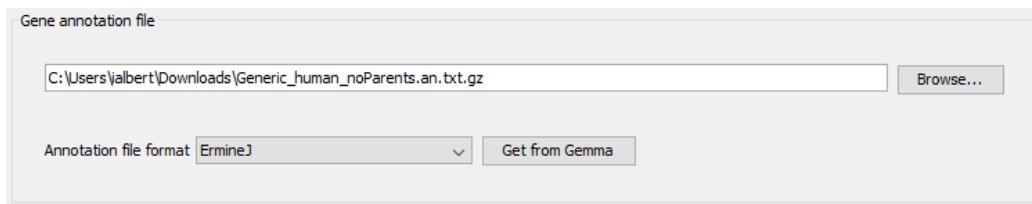


Figure 41.2

41.7 What is a gene score file?

Gene scores are used to select genes, then based on the selection the tool determines which functions are enriched. It is a tab delimited file with unique gene identifiers as the first column and score values as the second column.

For example, if you had a differential expression file (we'll show you how to make these in the RNA-Seq section) that looked like this³ then selecting and saving the first and last columns of would create a gene score file.

A score may be any value applied to a gene that represents some measure of interest, frequently that is a p-value or fold-change. Non-numeric values such as 'NaN' or '#NUM!' will be interpreted as zero. The first line is considered to be a header line.

id	padj
WSB1	5.26199960214757e-51
TPBG	7.77064023248874e-48
XBP1	1.88906929176218e-46
ASNS	3.89107811001364e-39
SLC7A5	1.68112640483516e-35

You can make this gene score yourself with:

```
curl -O http://data.biostarhandbook.com/ontology/deseq-output.txt
cat deseq-output.txt | cut -f 1,8 > gene_scores.txt
```

³<http://data.biostarhandbook.com/ontology/deseq-output.txt>

41.8 What is the expression data?

Expression data contain the measurements (e.g., read counts) associated with each gene and condition in a tab-delimited text file format. It is not used to determine functional enrichment. Instead, it is displayed when groups of genes are visualized.

The first row is the header line. Missing values must be indicated by blank, not by ‘NA’ or any other non-numerical values.

An example raw data file may look like this:

gene	C1	C2	C3	M1	M2	M3
Myl3	1207.16	1345.04	1247.69	2222.91	3041.39	2819.01
Tnnt2	2401.01	2542.16	2712.17	3648.55	5133.96	4505.65
Mybpc3	1689.82	2026.43	2173.50	3070.45	4439.41	3540.34
...						

To get our example expression⁴ data from the command line do:

```
curl -O http://data.biostarhandbook.com/ontology/normalized-matrix.txt
```

You will specify this expression data at the same time that we enter the gene score file.

41.9 How do I start an analysis in ErmineJ?

When we start up ErmineJ, the following page will be presented:

****GO XML File:**** ErmineJ will automatically download and store this file in the user’s ermineJ.data directory. This file is required even if you are not using GO.

Annotation File The annotation file appropriate to the experiment can be *downloaded from ErmineJ website using this⁵ link*. It is recommended to use ‘No parents’ versions of annotation file (parent GO terms are not explicitly listed in the file), as it will result in a faster startup time.

⁴<http://data.biostarhandbook.com/ontology/normalized-matrix.txt>

⁵<http://www.chibi.ubc.ca/microannots/>

41.10. WHAT IS AN OVER-REPRESENTATION ANALYSIS (ORA)?369

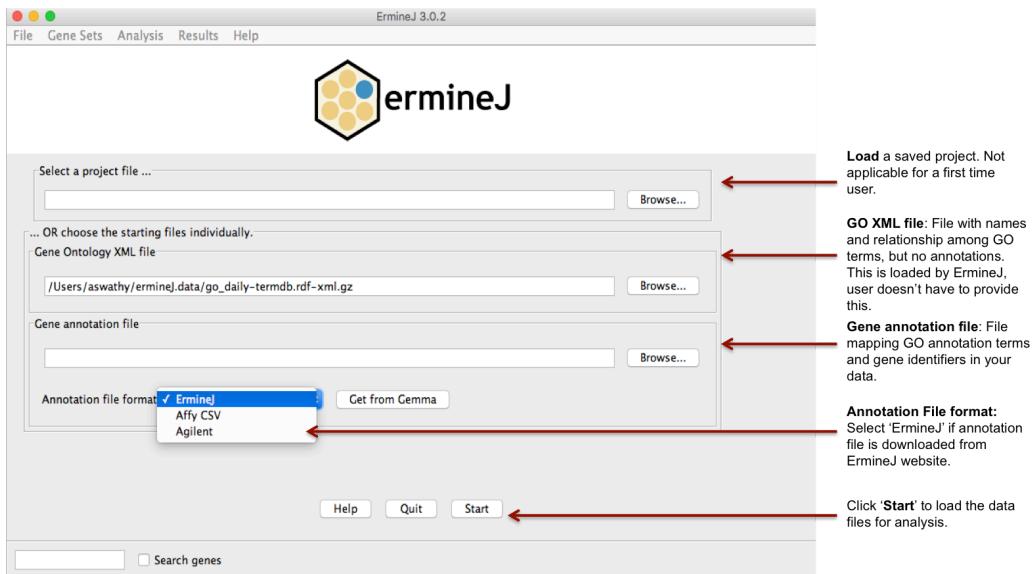


Figure 41.3

Get From Gemma used to be another option to get the annotation file. At the time of writing this document, the option appears to be non-functional.

Upon clicking 'start,' these files will be loaded and displayed in a table view. The different columns include the name and description of a group (e.g., GO ids and description), the number of elements in the group, and a score denoting the multi-functionality bias of the group. More columns will be added to this as you run an analysis. The details of the table view and tree view are explained below (see *Results of ORA analysis*).

Once these files are loaded, choose an analysis to run from the 'Analysis' tab.

41.10 What is an Over-Representation Analysis (ORA)?

Over-representation Analysis (ORA) is a method to identify statistically significant functional groups (e.g., Gene Ontology terms) among high-scoring genes in a gene list.

This method requires a gene-score file (see *input files* section) of all genes in

the experiment. The user needs to set a threshold of gene scores to ‘select’ the genes in the gene list to be examined for functional enrichment. The null hypothesis is that genes in the functional groups are randomly distributed between ‘selected’ and ‘non-selected’ genes. It uses hyper-geometric distribution to identify the functional groups that are significant among the genes passing the threshold.

41.11 How do I run an ORA?

To start an ORA analysis, choose ORA from Analysis ->Run Analysis tab.

ORA requires a complete list of gene scores, not just the selected gene list. Once a complete list of gene scores is uploaded, the user needs to specify a threshold to determine the genes to be selected for analysis. Raw gene expression data file can also be provided, though this is optional. The following parameters may be set by the user:

- **Maximum and minimum gene set sizes** - This refers to the number of genes in the Gene Ontology groups to be considered for analysis. It is recommended not to use very small or huge numbers here. Having a large number of functional groups can worsen the multiple-testing issues.
- **ORA-specific settings** - The ORA-specific settings dialog box is shown below. Depending on the gene scores in the input file, the user needs to select the appropriate choice. For example, if the scores are p-values, then the “take the negative log of the gene scores” box should be checked, and “larger scores in your score file are better” should be unchecked.
- **Gene score threshold** refers to the score value in the input file, and this determines how genes are selected. For example, if the input gene list has p-values, then the threshold might be 0.05 or 0.001, but not 4. The threshold refers to the values *prior* to log-transforming.

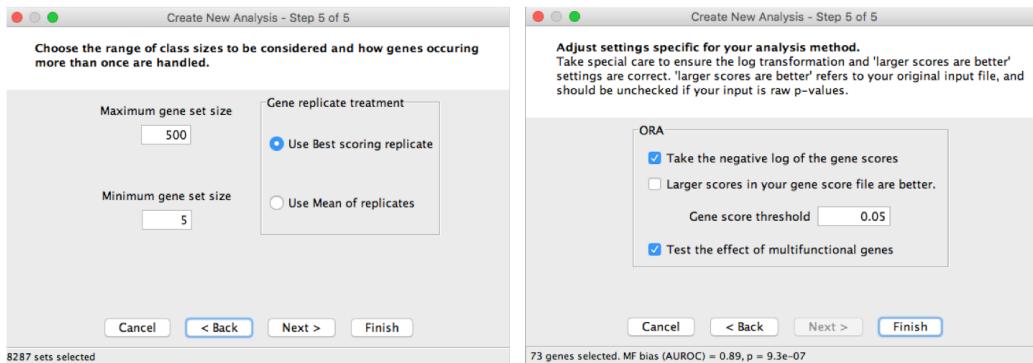


Figure 41.4

41.12 What are the results of an ORA analysis?

The ORA analysis displays the functional groups that are over-represented in the gene list and the associated p-value. The results can be viewed either in a table format (see below) or in a tree format.

The figure shows a screenshot of the ErmineJ software interface. The menu bar includes 'File', 'Gene Sets', 'Analysis', 'Results', and 'Help'. The main window has tabs 'Table' (selected) and 'Tree'. The table view displays the following data:

Name	Description	Size	Multifunc	ORA run on 'pad'
GO:0055003	cardiac myofibril assembly	12 [13]	0.63	7.10e-09 ♦
GO:008016	regulation of heart contraction	105 [106]	0.91	7.40e-09 ♦
GO:0043462	regulation of ATPase activity	28 [29]	0.72	8.90e-09 ♦
GO:0010927	cellular component assembly	145 [150]	0.92	1.26e-08 ♦
GO:006937	regulation of muscle contraction	77 [78]	0.90	1.81e-08 ♦
GO:0090257	regulation of muscle system process	110 [112]	1.00	1.82e-08 ♦
GO:005833	hemoglobin complex	7 [8]	0.22	5.30e-08 ♦
GO:005861	troponin complex	8	0.30	1.06e-07 ♦
GO:0032275	actin-myosin filament sliding	8	0.33	1.06e-07 ♦
GO:0003779	actin binding	300 [306]	0.91	1.55e-07 ♦
GO:1903522	regulation of blood circulation	157 [158]	1.00	2.00e-07 ♦
GO:0030029	actin filament-based process	302 [307]	0.93	2.45e-07 ♦
GO:005344	oxygen transporter activity	12	0.09	4.90e-07 ♦
GO:0075712	adult heart development	12 [13]	0.71	7.32e-07 ♦
GO:0015671	oxygen transport	13	0.08	7.32e-07 ♦
GO:0032982	myosin filament	15	0.30	7.32e-07 ♦
GO:0019825	oxygen binding	19 [20]	0.57	1.99e-06 ♦
GO:0070925	organelle assembly	247 [253]	0.92	2.16e-06 ♦
GO:0070252	actin-mediated cell contraction	35	0.84	2.24e-06 ♦
GO:0014866	skeletal myofibril assembly	5	0.38	2.48e-06 ♦
GO:0030240	skeletal muscle thin filament	5	0.38	2.48e-06 ♦
GO:0048747	muscle fiber development	38	0.77	2.60e-06 ♦
GO:0015669	gas transport	17	0.29	2.64e-06 ♦
GO:0030036	actin cytoskeleton organization	264 [269]	0.92	4.35e-06 ♦
GO:0030049	muscle filament sliding	6	0.22	4.94e-06 ♦
GO:0003209	cardiac atrium morphogenesis	23	0.86	8.47e-06 ♦
GO:0005859	muscle myosin complex	8	0.41	8.61e-06 ♦
GO:0048739	cardiac muscle fiber development	7	0.41	8.61e-06 ♦
GO:0003230	cardiac atrium development	24	0.86	1.03e-05 ♦
GO:0030509	BMP signaling pathway	62	0.79	1.16e-05 ♦
GO:0030048	actin filament-based movement	49	0.86	1.28e-05 ♦
GO:0048821	erythrocyte development	24 [27]	0.70	1.48e-05 ♦
GO:0071772	response to BMP	69 [70]	0.81	2.30e-05 ♦

Below the table is a search bar with the placeholder 'Search genes'.

1. Name/ID of the functional group.
2. Description of functional group.
3. No. of elements in the functional group.
4. Multi-functionality score ranging from 0 to 1. It indicates the degree to which the group is biased towards multifunctional genes. 1 is the highest.
5. P-value associated with functional group.
6. A '.' means the functional group has the same exact members as one or more groups.

Figure 41.5

The first four columns are standard to an ErmineJ table view. As each analysis is run, the results from each run will be added to the standard table as additional columns.

The p-values associated with each functional group, as obtained from the ORA run, are given in the 5th column. Benjamini-Hochberg correction is used to correct the p-values for multiple testing. Different shades of green color denote the strength of enrichment. A green diamond next to the p-value indicates a low effect of multi-functionality correction (low sensitivity) and a red diamond indicates that the group is sensitive to multi-functionality correction (see below for details).

Additional information on each functional group is obtained by double-clicking on the corresponding p-value. This displays the details of all genes in the functional group. Expression data is optional, however, if it was loaded, the expression values will be presented as a heatmap on the left-hand side as shown below.

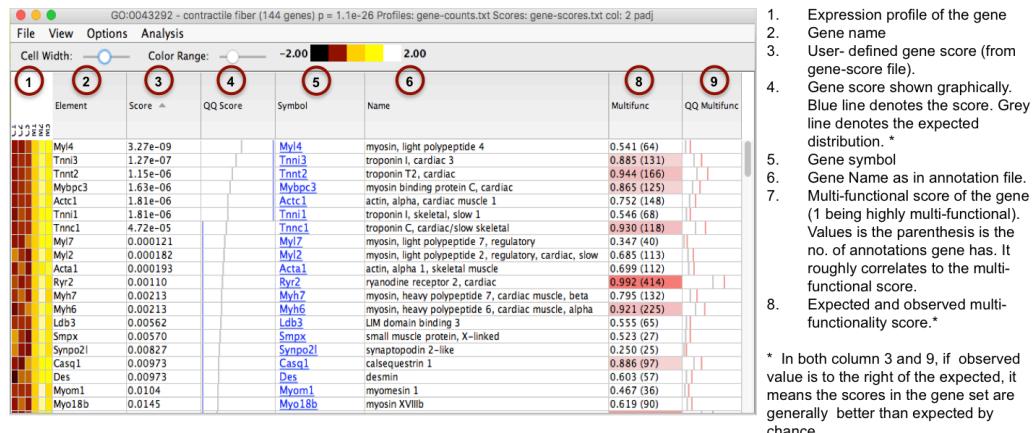


Figure 41.6

An example expression data file can be obtained using the commands below. This file contains mouse gene expression raw data for 6 samples (3 controls - C1,C2,C3 and 3 mutants - M1,M2,M3).

```
curl -O http://data.biostarhandbook.com/rnaseq/mouse-gene-expression.txt
cat mouse-gene-expression.txt | cut -f 1,3,4,5,6,7,8 >gene_counts.txt
```

41.13 What is a multi-functionality score?

Multi-functionality refers to the fact that a gene can have more than one function. In ErmineJ, function applies to the annotations for a gene. Some genes will have more functions (and hence annotations) than others, but we must recognize that this information is incomplete. Moreover, we can't even assume that the missing annotations are evenly distributed. Some genes are more “popular” and better-studied than others. Consequently, there is more information available about “popular” genes than “less popular” genes.

ErmineJ calculates a multi-functionality score for each gene. The equation takes into account the size of the functional group. In general, however, the multi-functionality score of a gene correlates highly with the number of annotations it has.

The fourth column in the ErmineJ table format is the multi-functionality score of a functional group. It is a normalized rank that shows the degree to which a group has multi-functional genes. The higher the score, the more generic the functional group is. Details on how the multi-functionality score for a functional group is calculated can be found here⁶.

41.14 Is multi-functionality good or bad?

From solely a data interpretation perspective, multifunctional genes are more likely to be false positives. We can think of this as a multiple testing problem. When a gene is multi-functional, it has more opportunities to be shown as enriched just by chance.

Depending on the analysis method used, ErmineJ employs different approaches to correct for multi-functionality. In ORA, genes are iteratively removed in the order of their multi-functionality to test the effect this removal has on the enriched groups. The idea here is that if any functional group is enriched merely due to the multi-functionality of the genes, then such groups will fall out as you remove those genes.

⁶<http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0017258>

41.15 What is Gene Score Resampling (GSR)?

Gene Score Resampling (GSR) is another method in ErmineJ (similar to ORA) to identify significant functional groups in a gene list. GSR is different from ORA in that it does not require a threshold. Instead, GSR computes an aggregate score (e.g., a geometric mean of gene scores) for a functional group and calculates its significance by resampling the data.

The results of both ORA and GSR methods applied to the same input datasets are shown below. GSR gave fewer effects compared to ORA in this analysis.

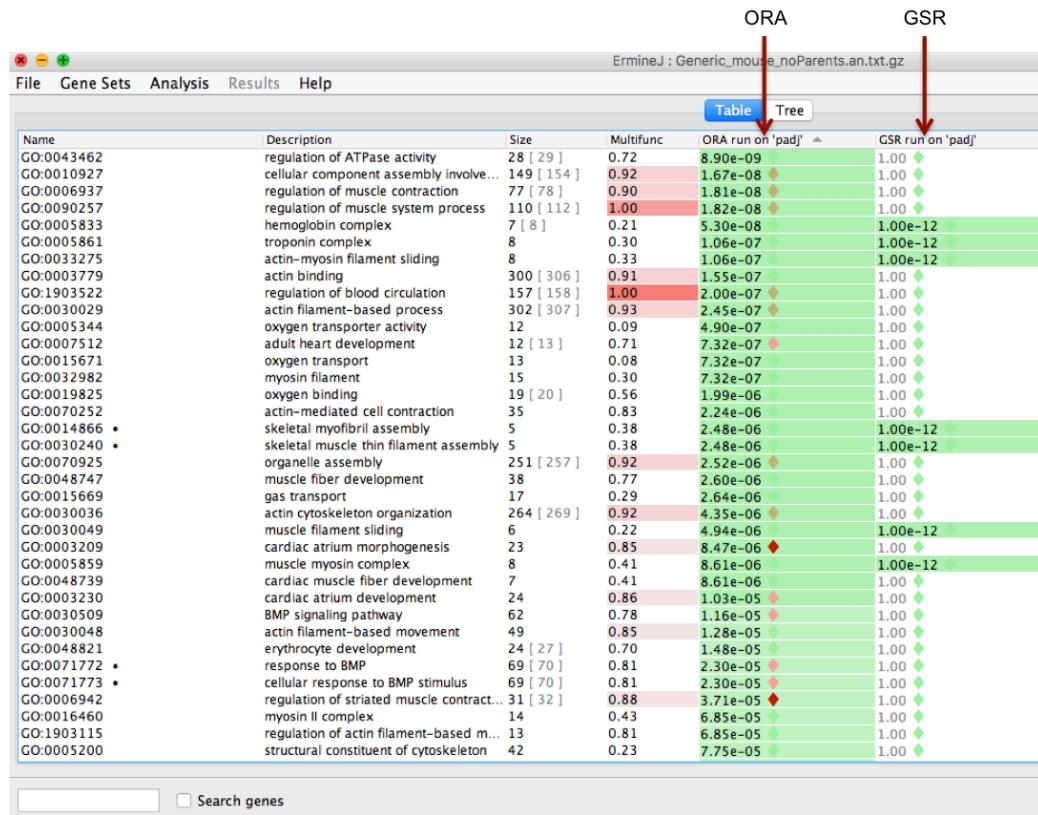


Figure 41.7

41.16 What is Correlation Analysis in ErmineJ?

The correlation analysis (CORR) differs from other ErmineJ methods by assessing how well the genes in a functional group are clustered together. This method does not use gene scores. Instead, it uses the expression profiles of the genes themselves. CORR checks if the correlation among the members of a functional group is more than expected by chance. The raw score for a functional group is the absolute mean of the pairwise correlation of its members.

Unlike other methods in ErmineJ, the CORR analysis is computationally intensive. Hence, one needs to be careful while setting the parameters when running this analysis.

Chapter 42

Student reports on functional enrichment

The following entries are submissions by students taking the **BMMB 852: Applied Bioinformatics** class at Penn State. I selected a entries that go beyond a typical ontology analysis and make additional interesting observations. The assignment stated the following:

Take a list of genes and perform a gene list over-representation or a gene set enrichment study with a method of your choice.

In a few paragraphs discuss what you have found. What functionality appears to characterize the list? You may use the genes in the files from here:

[http://data.biostarhandbook.com/courses/2017-852/
media/05/](http://data.biostarhandbook.com/courses/2017-852/media/05/)

Alternatively, you may use any other list of genes that you have.

Within each section the content is taken verbatim from the submission by the author named in the title. Only minor corrections were made if these were necessary. My own assessment and reply are written in italics at the end of each section.

42.1 David Northover is amused

After examining the gene list given in Enrichr, there were a few common enriched threads, which suggests that there were a couple of distinct groups of genes associated with specific functions.

The clearest signal was in relation to adipogenesis, the generation of fat stores (adipose cells and tissues). Pathway analyses showed strong enrichment for adipogenesis, transcription factors related to adipogenesis, as well as several insulin related pathways. This is also consistent with the identified ontologies, matching insulin and appetite response, as well as hormone activity and glucose transport. Cell types matched to the Pituitary (hormone) and Liver, above adipose tissue, so the genes appear to be more focused on the regulatory side. Disease screening was (given the rest) unsurprisingly associated with obesity, cholesterol, and insulin disorders (such as Hyperglycemia and Hyperinsulism).

A possible secondary function may be related to neural activity, as while there was no overexpression of nerve cells, there were KEGG hits for neuroactive ligand-receptor interaction and GO axon cell components. This may in some way be related to sending/recieving neural signals related to adipogenesis in some way, however.

As an amusing side note about the dangers of over-interpreting limited enrichment results, and of the issues with self-reported data, it was also enriched for the UK Biobank term “Never Had Laser Treatment for Glaucoma/High Eye Pressure”, which is likely to be a very strange coincidence. There are other UK Biobank terms that may be leading, though - “Relative Age Voice Broke”, “Comparative Body Size (Age 10)” and “Age When Periods Started”, though less enriched, match with a “Precocious Puberty” listing elsewhere. I would be uncomfortable making too much out of these - given the number of comparisons scanned through - but it would be worth trying to find which genes triggered those items in specific but not making a conclusion solely out of this kind of data.

Great points are made here, we are never quite certain of what information makes it into the GO or other resources and what the vetting process is.

42.2 Kristin Passero notes that the presentation matters

I took the following gene list from the DESeq Output file

- <http://data.biostarhandbook.com/ontology/deseq-output.txt>

available on the [Using the AGRIGO server][#agrigo] page in the BioStar Handbook. I selected the top 19 differentially expressed genes.

Genes: WSB1, TPBG, XBP1, ASNS, SLC7A5, SFRP2, HERPUD1, SLC3A2, HES6, SLC7A11, CARS, VEGFA, SHMT2, TOP2A, SAFB2, MTHFD2, YARS, DNAJB9, HYOU1

I used both PANTHER and AgriGO to perform gene set enrichment for biological process. One thing I noticed is that the GO terms produced by PANTHER were far more specific than those produced by AgriGO for the same list of genes. PANTHER's terms seemed more "event" oriented (response to stress, to apoptosis, etc.) while AgriGO's all includes the words "metabolic process" preceded by a different compound name (amino acid, cellular amine, oxyacid, etc.) There was only one shared term among them, "carboxylic acid metabolic process, GO:0019752." However, were I to have used these tools without comparison, AgriGO would have given the impression that my subset of genes was enriched in basic metabolic process, whereas PANTHER would have suggested the genes were involved primarily in cellular-stress-response pathways.

PANTHER lists significant enrichment in processes such as PERK-mediated unfolded protein response, negative regulation of ER-unfolded protein response, negative regulation of endoplasmic reticulum stress-induced intrinsic apoptotic signaling pathway, IRE1-mediated unfolded protein response. However, it does list more "generic" processes such as amino acid transport across the plasma membrane, tryptophan transport, and branched-chain amino acid transport. As such, knowing nothing about the actual function of these genes, I would predict they're involved in collecting free amino acids for unfolded protein repair. The AgriGO processes include cellular amino acid metabolic process, oxoacid metabolic process, carboxylic acid metabolic process. Based on that information I might presume that these proteins have something to do with amino acid modifications.

The PANTHER results give a picture with greater context; however that

is based on my own presumptions, knowing nothing except these terms I have been given. As such, PANTHER has a greater ability to lead good-bad conclusions just because it gives enough information to make the user dangerous. On the other hand, AgriGO offers a more general overview of processes, so the conclusions drawn from them may be more correct due to the fact the processes listed could be “shared” by different cellular processes. However, if one looks at all the AgriGO results, further along the list you begin to see terms involving cell death and stress — they are just not FDR significant. At first I thought maybe they were using different statistical methods, but I found they are both using Fisher’s exact test and some variant of the FDR correction. As such, I am not sure of the origin of the discrepancy in significance.*

What I like the most about this entry that it notes that the presentation of a complex information can in itself shape our thinking and rationalization later on. The choice of words, the order of listing information, the visualization of the data may have substantial effects on what we come to believe about our data.

42.3 Jeremy Held gets somewhat different answers but all make sense

I am generating my own gene list from the model plant *Arabidopsis thaliana*. To generate the list, I first took out the top 2000 genes with annotations, which equated to 20 annotations at the low end. The average for *Arabidopsis* is only 7. From my list of 2000, I shuffled the gene names and took 20:

OCP3 AT5G11270 DMR6 AT5G24530 DELTA-OAT AT5G46180 ARAC5
AT1G75840 CBF1 AT4G25490 NRPD1B AT2G40030 DDF1 AT1G12610
KT2/3 AT4G22200 HBT AT2G20000 RBR1 AT3G12280 PHB2 AT1G03860
ATR1 AT4G24520 EIN3 AT3G20770 MKK3 AT5G40440 CLPP6 AT1G11750
ROXY1 AT3G02000 PBP1 AT3G16420 PEPR1 AT1G73080 GAPA-2
AT1G12900 VAB1 AT1G76030

Using Panther Using the TAIR IDs, PANTHER was able to uniquely map all 20 genes. The reference list contained 27,581 genes. I only looked at enriched biological process terms with an FDR <0.05.

The highest enrichment was in ‘regulation of cell growth’ (over 40-fold). Only 102 genes in the reference have this annotation, which seems low. 5 genes in my list are annotated as ‘defense response to bacterium’, and 4 as ‘innate immune response’, indicating my list is enriched for plant defense. I also have enrichments for more vague terms including ‘response to acid chemical’ and ‘response to hormone’.

Using AgriGO AgriGO shows many of the same terms as PANTHER, including defense genes and chemical/hormone responses. Calculation of FDR is different between PANTHER and AgriGO. For example, the lowest FDR in AgriGO is ‘response to oxygen-containing compound’ (1.1E-05). In PANTHER, this term is not the lowest, with its FDR at 1.77E-04. Also interesting, AgriGO does not pull out ‘regulation of cell growth’, even though it had the largest enrichment in PANTHER. Instead, new terms pop up, albeit at high FDRs, including ‘regulation of transcription, DNA-templated’ and ‘cell communication’.

Using G:Profiler This service employs the strictest formula, as it only found 16 significant biological process terms. However, many of the same defense and chemical response terms found in PANTHER and AgriGO are identified here. The right side of the table shows that the function of these genes has been verified experimentally for most of my genes (indicated by red squares). It also shows that some genes have broad functions, while others only have a few. This may be because certain genes have received more study than others.

Conclusion This exercise has clearly shown the variability in a gene enrichment analysis depending on the program you use. However, terms relating to plant defense and responses to chemicals and hormones were identified in each program, suggesting those designations may be accurate. Additionally, as these genes were randomly selected, we would not expect strong associations for particular functions, as perhaps we would see if our gene list came from an experiment. As shown on G:Profiler, no GO term encompasses every gene.

This is an example where albeit different, the answers seem to indicate similar biological mechanisms.

42.4 Kelly Gomez Campo works on her thesis

The study case is based on an experiment from my PhD thesis. I was testing the hypothesis that phenotypic plasticity is associated to methylome changes in a reef-building coral “*Acropora*”. Because coral genomes are annotated based on human genomes, I chose 288 genes from one of my outputs

- (genes not included here as this study is still in progress)

and use *Homo sapiens* as the reference list. A large list of genes in coral genomes are still unknown or not annotated, or are annotated as uncharacterized proteins. An statistical overrepresentation test was performed in PANTHER. The functionality that appears to characterize the list is cell organization. This analysis showed a significant enrichment as high as 114 (5850 REF) in GO biological process cellular component organization or biogenesis (ID GO:0071840). According to the source, is a process that results in the biosynthesis of constituent macromolecules, assembly, arrangement of constituent parts, or disassembly of a cellular component. The most significant enrichment functions are related to this biological process, organelle organization, regulation of cellular component organization, regulation of organelle organization. Response to ion calcium ion (ID GO:0051592) was a differences enriched biological function from the output, relates to any process that results in a change in state or activity of a cell or an organism (in terms of movement, secretion, enzyme production, gene expression, etc.) as a result of a calcium ion stimulus. According to the ancestor chart is a biological process that relates to response to a inorganic chemical stimulus, specifically, to metal ion. It's an interesting enriched function since Calcium ions (Ca^{2+}) are key second messengers in a variety of eukaryotic cell signaling pathways, and they function in the regulation of diverse cellular processes. While it has become clear that Ca^{2+} stimulates many cellular processes, such as muscle contraction, cellular proliferation, gene expression, secretion of hormones and neurotransmitters, exocytosis, and chemotaxis, it has also been realized that Ca^{2+} is very toxic (Schnellmann and Covington, 2010). Thus, the free intracellular Ca^{2+} concentration ($[\text{Ca}^{2+}]$) must be highly regulated to achieve a proper balance between Ca^{2+} -mediated cell function and Ca^{2+} -mediated cell death. Double-strand break repair via homologous recombination (ID GO: 0000724) is another biological function that was significantly enriched. The function is defined as the error-free repair of a double-strand break in DNA in which the broken DNA molecule is repaired using homologous sequences.

A strand in the broken DNA searches for a homologous region in an intact chromosome to serve as the template for DNA synthesis. The restoration of two intact DNA molecules results in the exchange, reciprocal or nonreciprocal, of genetic material between the intact DNA molecule and the broken DNA molecule. The enriched functions go accordingly to the treatment in my experiment and the phenotypic response I have for the organisms. The treatment was a switch from low light conditions to a high light conditions. According to this analysis, the high light treatment induced a cell reorganization, DNA damage and repair, with calcium ions as a stimulator of cellular processes.

What I love about this submission is that it takes the concepts, methods that we cover in class and applies them right away to a realistic problem. This is one of the paradigms that I am building the entire book around. I hope to cover subjects in a manner that allows readers to apply what they learn right away to “realistic” situations. To go beyond toy examples and canned demos. I am always pleased to see it work in practice

42.5 Tomas Lopez Londono finds that even randomly selected genes may be enriched

Script (gene_counts.txt file was created from goa_human.gaf):

```
cat gene_counts.txt | head -n 1000 | cut -f 3 -d ' ' | shuf | head -n 20
```

Resulting gene list:

```
PKD2
KAT2B
HGF
GAB1
YES1
PSMA7
STXBP1
PSMB8
HSPB1
NCBP2
```

42.5. TOMAS LOPEZ LONDONO FINDS THAT EVEN RANDOMLY SELECTED GENES MAY BE

SMURF2
MAD1L1
PPARGC1A
DDX58
ZC3H12A
PPP3CA
PSMD8
GATA4
PARP1
FADD

ORA method: Panther14.1 (<http://pantherdb.org/tools/compareToRefList.jsp>) Analysis performed in terms of biological processes, with false discovery rate of observations (FDR) set by default (FDR<0.05)

Discussion: What functionality appears to characterize the list?

When organizing the results from highest (<0.05) to lowest false discovery rate (FDR) values, it is observed that the function with the highest chance of FDR is “blood vessel development”, with 4 genes of the uploaded list (20 genes) out of the 487 genes in the reference list that map to this particular function. The Fold Enrichment for this function is 8.21, which indicates that it is overrepresented. The definition of this function, according to AmiGO, is “the process whose specific outcome is the progression of a blood vessel over time, from its formation to the mature structure”. The total number of annotations for this function is 7073, with a significant portion of uncharacterized proteins for each gene/product annotated.

The function with the lowest FDR is “response to organic substance”, which is defined as “any proves that results in a change in state or activity of a cell or an organism as a result of an organic substance stimulus”. All genes of the uploaded list (20) out of the 2992 in the reference list mapped this particular function. Total number of annotations for this function is 82892, also with a significant portion of uncharacterized proteins for each gene/product annotated. The fold Enrichment value is similar to the previous function (6.68), indicating that this function is also overrepresented. In fact, all functions selected with FDR<0.05 were overrepresented. This may indicate that the fraction of observed genes in the uploaded list associated to particular pathways always exceed the number of expected genes associated to those pathways when considering FDR values below 0.05.

One of the functions with highest Fold Enrichment is the “regulation of ATP biosynthetic process”, with a value >100 . This value indicates that this is probably one of the most over-represented functions. 2 genes of the uploaded list out of the 19 genes in the reference list that map to this particular function. The definition of this function is “any process that modulates the frequency, rate or extent of ATP biosynthetic process”. It has only 162 annotations, compared to the previously discussed functions with 7073 and 82892 annotations. Interestingly, most proteins encoded by the genes/products associated with this function have been characterized, opposite to the other two functions with significant portions of uncharacterized proteins. This may indicate that this pathway is much better described and understood. The extremely complex chart with the hierarchical classification of gene ontology relationships can be interpreted as proof of that.

This example goes to demonstrate that even “randomly” selected genes may exhibit features that distinguish them from a background. Was Mr. Londono “lucky” to selecte genes with common functions or was the statistical method unable to cope with the data representation - an undecidable question in my opinion. In general, statistical methods don’t work all that well for functional enrichment because the information that is being put into context has been collected in deliberate and systematic way, aiming for certain goals - hence violate assumptions that the statistical methods rely upon

Part IX

REPRODUCIBILITY

Chapter 43

Scientific reproducibility

The reproducibility of scientific analyses is (or rather should be) a cornerstone of science. The scientific method itself depends on scientists' ability to verify statements made by others.

In reality, the reproducibility process is fraught with many pitfalls, and solutions are few and far between. The problems we face are in part due to rapid and radical transformations in different scientific domains, especially Life Sciences.

At the same time the very meaning of the word “reproducibility” has been distorted. By using the terms incorrectly, and focusing on the “outrageous” yet rare and non-representative events, well intentioned scientists, the very champions of “reproducibility” cause more harm than good.

43.1 What does the word “reproducible research” mean?

In what might be the most ironic of situations - scientists have a hard time agreeing what the words “reproducible research” mean.

The very definition of “*reproducible research*” itself seems not to be reproducible! There is a surprising amount of uncertainty and confusion even what the word ought to mean.

43.2 What is the “red herring” of reproducibility?

We invite you to perform a Google search on the words scientific reproducibility. Here is what our first hit is:

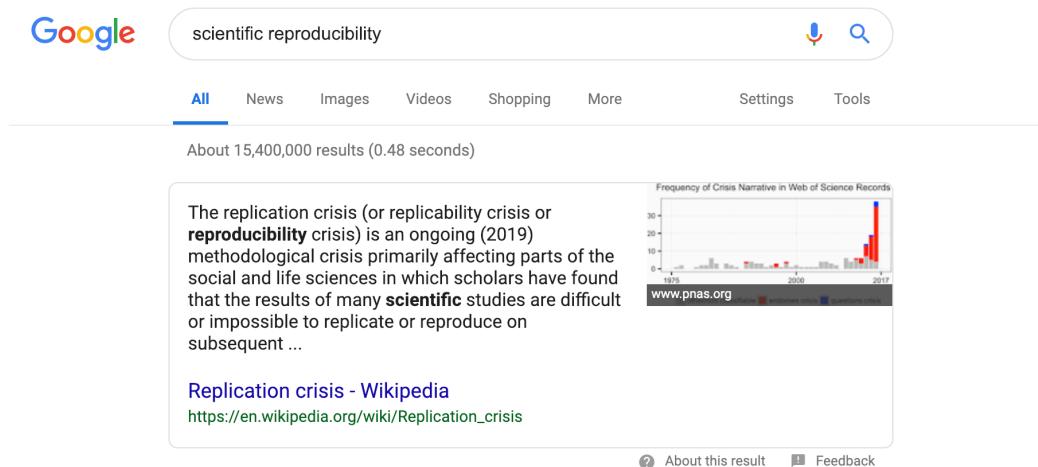


Figure 43.1

There is a systematic mischaracterization of the very concept of reproducibility to imply that there is a “crisis” in science and that reproducibility is about “fixing” broken science. Examples abound, here are the leading scientific publications pushing the same line of thought:

- Editorial: Reproducibility¹ in the Science
- Challenges in irreproducible research² in Nature

In the above framing, reproducibility is a sort of “defense against dark arts” a way to catch you-know-who, plus their posse of other cheaters, liars and evil scientists. Alas it is the painfully wrong way to frame the problem of reproducibility that only invites fruitless solutions.

¹<http://science.sciencemag.org/content/343/6168/229>

²<https://www.nature.com/collections/prbfkwvwz>

43.3 Is science really facing a reproducibility crisis?

With all the alarms going off, here is a different take:

- Opinion: Is science really facing a reproducibility crisis, and do we need it to?³ PNAS (2018)

Here are the closing thoughts that we fully agree with:

Therefore, contemporary science could be more accurately portrayed as facing “new opportunities and challenges” or even a “revolution”. Efforts to promote transparency and reproducibility would find complete justification in such a narrative of transformation and empowerment, a narrative that is not only more compelling and inspiring than that of a crisis, but also better supported by evidence.

There is no crisis in the sense of “cheating” and invalid science being published at increasing rates.

There is an alarming trend in publishing research results that are not properly explained. Framing reproducibility to address the transparency of the processes would be a far more fruitful direction to pursue.

43.4 How do scientists attempt to define what reproducibility is?

Here we present the ideas that scientists came up with, in the order of least demands to most demands that they put on authors:

1. A brief description of the decisions that made during the analysis.
2. A detailed description of the commands that were used to generate the report.
3. A list of all instructions and a snapshot of the intermediate data that the commands produced.

³<https://www.pnas.org/content/115/11/2628>

4. Publishing the analysis as a single document that one can easily re-run.
5. All the above with the added information of the exact computer setup, operating system configuration, and software version that was used to generate the analysis.
6. All the above packaged into a single “virtual” machine that emulates the same setup the authors had at their disposal that can be re-run.

43.5 So what does reproducibility mean?

First you should not conflate the different interpretations of the word “reproducibility”. In this book we focus solely on the reproducibility of an analytical process where:

1. Reproducibility is not a binary classification: *reproducible* vs *irreproducible*. Reproducibility is a *scale*.

Reproducibility goes from *easy to follow* to *practically impossible to redo*; the scale should be evaluated in the context of using the information that the scientists provided.

2. Reproducibility is not a measure of scientific validity. It is a measure of *transparency*. Do we know what took place?

In our opinion a result is reproducible when we have been provided with all information required to make a the same scientific discovery. The benefit is that we can now build on this previous discovery and take it further. Reproducibility, as used in this book, quantifies the effort it takes to follow along the discovery process. See also: Will the best bioinformaticians on the planet produce reproducible analyses?

43.6 How difficult is to re-analyze data?

As you will see it yourself, it is usually far more straightforward to re-analyze data with your methods than to reproduce and retrace the same steps as published in a scientific paper.

In the majority of cases, you’d want to know how the data has been analyzed not to rerun the same commands, but to assess the strengths and weaknesses

of the analysis and to identify any possible errors or inconsistencies of that process.

43.7 What is the best strategy to reproduce results?

As a bioinformatician, you may often need to reproduce published results. We must warn everyone that doing so by following the descriptions and explanations found within research papers is usually a tedious and frustrating process.

A typical scientific paper spends far more time trying to convince readers that its findings are significant rather than explaining the steps of how that analysis took place. The focus on “selling the result” is a normal consequence of how scientific publications are judged. Novelty is the most crucial requirement.

But there is another and more devious complicating factor. The so-called “*scientific narrative*” is simplified and misleading - it depicts the bioinformatics analysis as a linear chain of decisions: “*first we did this, then we did that*.” The descriptions paint a picture that does not correspond to reality. Recall the plot from What is data?:

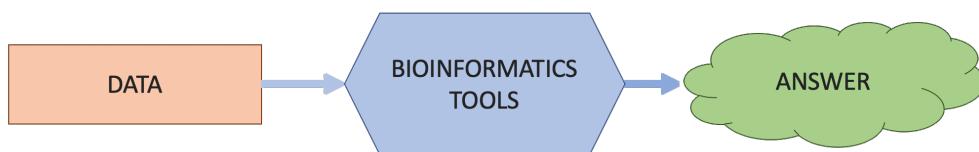


Figure 43.2

Alas the image above is not what the authors actually did. Most of the time the authors explored several, perhaps dozens of alternatives, ran into many dead ends, backtracked and tried their analyses in several different ways. After quite a bit of work, they’ve finally honed in on what they believe to be the correct approach for making the discovery. But within the scientific paper, the long and branching chains of decisions and actions are described as

straightforward choices, and the failed attempts are never mentioned. This is what most likely they have done:

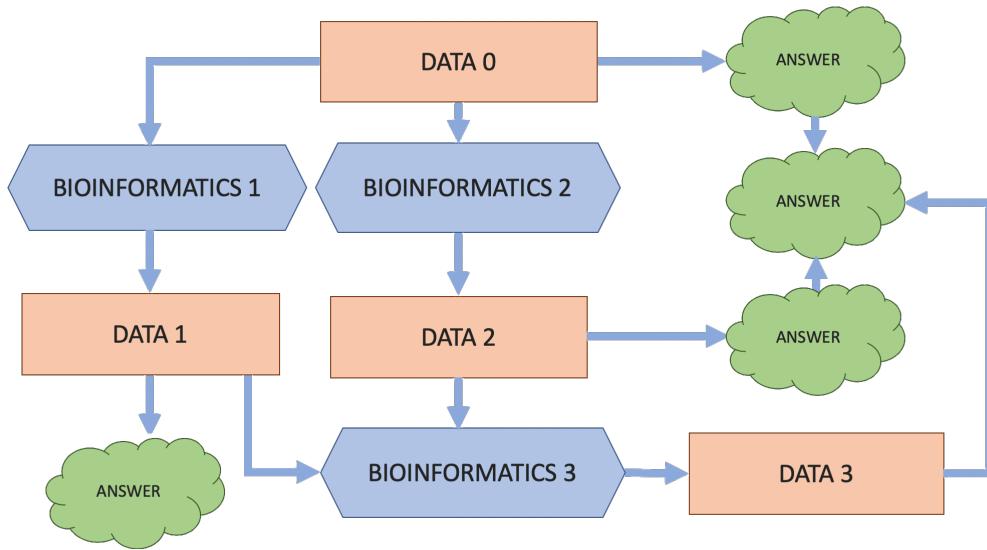


Figure 43.3

Your aim should always be to re-analyze the data in a more straightforward way. “Re-doing it faster and better” is the approach that we champion in this book and it is the one that has served us well. With time methods improve and the algorithms change, then all scientific observations should be discoverable in different ways. Sticking to one protocol because that’s what one always used before and that is what “my group is using” is not always a good choice. It may still be the optimal choice, but you should ensure that you understand the alternatives. Once you know what type of insights are in your data, there is almost always a simpler and more efficient method to find them.

43.8 Are the challenges of data reproducibility recognized?

Most organizations funding science recognize that the low level of reproducibility of results are a growing problem in bioinformatics and other areas of

43.8. ARE THE CHALLENGES OF DATA REPRODUCIBILITY RECOGNIZED?393

science. These organizations try to implement what they believe is a “carrot and stick” approach, policies of offering a combination of rewards and punishment to induce behavior that should result in more reproducibility.

In our opinion, these policies, at least those that we are aware of, rely mostly on “stick” and have less of the “carrot” component.

In summary, you should be mindful that the problem is widely recognized, but that it is also widely misunderstood. The current mitigation efforts appear to be inadequate.

Chapter 44

Redo: Genomic surveillance elucidates Ebola virus origin

44.1 Quick links

- SRA BioProject: PRJNA257197¹
- All variants: iSNV-all.vcf²
- 2014 strain specific variants: table-S4-2014_specific_snps.csv³

Accession numbers:

- AF086833 Ebola virus - Mayinga, Zaire, 1976, complete genome.
- KM233118 Zaire ebolavirus isolate Ebola virus/H.sapiens-wt/SLE/2014/Makona-NM042.3, complete genome

Getting the runinfo for the project:

```
esearch -db sra -query PRJNA257197 | efetch -format runinfo > runinfo.csv
```

How did we find the links listed above? Read on.

¹<https://www.ncbi.nlm.nih.gov/bioproject/PRJNA257197>

²<http://data.biostarhandbook.com/redo/ebola/iSNV-all.vcf>

³http://data.biostarhandbook.com/redo/ebola/table-S4-2014_specific_snps.csv

44.2 How to get the information for the Ebola paper?

We were interested in accessing the data for the study titled

- Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak⁴

published in 2014 in the Science research journal.

The screenshot shows the Science journal website. The main header features the word 'Science' in large white letters on a black background, with 'AAAS' in smaller letters to its right. Below the header is a red navigation bar with links for 'Home', 'News', 'Journals', 'Topics', and 'Careers'. Underneath the navigation bar, there's a black footer bar with links for 'Science', 'Science Translational Medicine', 'Science Signaling', and 'Science Advances'. On the left side of the main content area, there are social sharing icons for Facebook, Twitter, and Google+. The main title of the article is 'Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak'. Below the title is a list of authors and their affiliations. The article is categorized as a 'REPORT'.

SHARE REPORT

Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak

Stephen K. Gire^{1,2,*}, Augustine Goba^{3,*†}, Kristian G. Andersen^{1,2,*†}, Rachel S. G. Sealfon^{2,4,*}, Daniel J. Park^{2,*}, Lansana Kanneh³, Simbirie Jalloh³, Mambu Momoh^{3,5}, Mohamed Fullah^{3,5†}, Gytis Dudas⁶, Shirlee Wohl^{1,2,7}, Lina M. Moses⁸, Nathan L. Yozwiak^{1,2}, Sarah Winnicki^{1,2}, Christian B. Matranga², Christine M. Malboeuf², James Qu², Adrienne D. Gladden², Stephen F. Schaffner^{1,2}, Xiao Yang², Pan-Pan Jiang^{1,2}, Mahan Nekoui^{1,2}, Andres Colubri¹, Moinya Ruth Coomber³, Mbalu Fonne^{3,‡}, Alex Moigboi^{3,‡}, Michael Gbakie³, Fatima K. Kamara³, Veronica Tucker³, Edwin Konuwa³, Sidiki Saffa³, Josephine Sellu³, Abdul Aziz Jalloh³, Alice Kovoma^{3,‡}, James Koninga³, Ibrahim Mustapha³, Kandeh Kargbo³, Momoh Foday³, Mohamed Yillah³, Franklyn Kanneh³, Willie Robert³, James L. B. Massally³, Sinéad B. Chapman², James Bochicchio², Cheryl Murphy², Chad Nusbaum², Sarah Young², Bruce W. Birren², Donald S. Grant³, John S. Scheiffelin⁸, Eric S. Lander^{2,7,9}, Christian Happi¹⁰, Sahr M. Gevao¹¹, Andreas Gnirke^{2,§}, Andrew Rambaut^{6,12,13,§}, Robert F. Garry^{8,§}, S. Humarr Khan^{3,‡§}, Pardis C. Sabeti^{1,2,†§}

Figure 44.1

Now you could read this paper all the way through and not be able to find the information that point to where the data is located. If you don't believe

⁴<http://www.sciencemag.org/content/early/2014/08/27/science.1259657.full>

us, see it for yourself. See if you could locate the link to the sequencing data. There is even a section called Figures & Data, common sense indicates that the data access were listed there!

44.3 Is it possible to access the data for this analysis?

As a matter of fact we were only able to find the link to the data because we knew beforehand that information deposited in the Short Read Archive is typically designated with a bio project code that starts with PRJN:. A text search for this pattern leads to a tiny, small entry at the very end of the paper, past the references, in the acknowledgments - way out of sight – in the section that we could call a scientific “*no-mans land*.” It is there where we finally found what many might consider one of the most important deliverable of this project:

Sequence data are available at NCBI (NCBI BioGroup: **PR-JNA257197**).

There is no link, no other information there. You are just supposed to know what this means. Now make a note of this number PRJNA257197. It will be our primary means to get all the data for this project, helps us avoid clicking around on websites and gives us a systematic, automated and repeatable way to get the data.

44.4 Where can we find out more about this dataset?

Now that we have the id number PRJNA257197 we can search for it at the NCBI website⁵, to find a Bioproject titled:

⁵<http://www.ncbi.nlm.nih.gov/>

44.4. WHERE CAN WE FIND OUT MORE ABOUT THIS DATASET?397

- Zaire ebolavirus sample sequencing from the 2014 outbreak in Sierra Leone, West Africa.⁶

At the time of writing, the project summary included 891 experiments that produced 249 nucleotide and 2240 protein sequences.

Project Data:

Resource Name	Number of Links
SEQUENCE DATA	
Nucleotide (Genomic RNA)	249
SRA Experiments	891
Protein Sequences	2240
PUBLICATIONS	
PubMed	1
PMC	1
OTHER DATASETS	
BioSample	716

Figure 44.2

Once you are on the Bioproject site, it is no less shocking just how little information is disseminated on what exactly has been deposited or how these results were obtained.

- What are these 249 nucleotide and 2240 protein sequences?
- How reliable is the information?
- How are they different from what was known before?
- What methods and practices were used to derive them?

The rather lengthy Supplementary Information⁷ does not help much, either. This 29-page document has an extensive list of tools and techniques, but it severely lacks specificity. It contains statements such as:

EBOV reads were extracted from the demultiplexed Fastq files

⁶<https://www.ncbi.nlm.nih.gov/bioproject/PRJNA257197/>

⁷<https://www.sciencemag.org/cgi/content/full/science.1259657/DC1>

using Lastal against a custom-made database containing all full-length EBOV genomes

This description is inadequate for reproducing the results: it only poses more questions.

- What is this custom made database?
 - Why isn't this custom database deposited here as well?
 - Which full-length EBOV genomes were included?
 - What does it mean to be extracted with Lastal?

Note another missing element here: who do we contact if we have a question? While papers do have corresponding authors, those are almost never the ones that know the intricate details of the analysis.

44.5 How do we download results for this paper?

We will learn in later chapters of automated ways to get the sequencing data for this project but what about the results themselves? Can we get those in a flexible format? Here again, the standards vary widely. In this case, the approach is borderline ridiculous. You need to visit the publisher's website, navigate to an arbitrary location and then individually download files with obscure file names, without knowing precisely what these files contain.

You can now get these to look at from the command line (via ridiculously named URLs) because we went through the tedious process of copy-pasting each and collected them for you below. Stop and think about how absurd these URLs look and how inefficient is to distribute data this way:

Unzip a file and see what it contains - note how uninformative the name of the file name is:

```
unzip 1259657_table_s1.zip
```

As you will see later, you can unzip all files in one command (and don't worry if you don't get this yet, we'll talk about this later):

```
ls -1 *.zip | xargs -n 1 unzip
```

Some of the files are in PDF format. One such PDF contains nothing more than a table with four rows and five columns! Since it is distributed as a PDF and not as plain text or an Excel sheet that we could export into other formats we cannot automatically process the information contained in this file without manually transcribing it into a machine-readable form!

Other files are in Excel but follow an arbitrary ad-hoc format unsuited for automated processing. Again those will need to be reformatted differently to comparable to information standard formats.

We think we made our point - today data distribution is an ad-hoc and superficially supervised process, a bottleneck to innovation that demands future users do substantial extra work that should not be necessary.

Chapter 45

Redo: Zika virus targets human cortical neural precursors

45.1 Quick links:

Links for easy access:

- SRA BioProject: PRJNA313294¹.
- The GEO series: GSE78711².
- Up-regulated genes: zika-up-regulated.csv³
- Down-regulated genes: zika-down-regulated.csv⁴

Getting the runinfo for the project:

```
# Crosslink to sra. Need to "sleep" to slow things down so a bit so NCBI does not ban
esearch -db gds -query GSE78711 | (sleep 1 && elink -target sra) | efetch -format r
```

```
# Other sample information
esearch -db gds -query GSE78711 | efetch > sampleinfo.txt
```

How did we found the links listed above? Read on.

¹<https://www.ncbi.nlm.nih.gov/bioproject/PRJNA313294>

²<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE78711>

³<http://data.biostarhandbook.com/redo/zika/zika-up-regulated.csv>

⁴<http://data.biostarhandbook.com/redo/zika/zika-down-regulated.csv>

45.2 How to get the information for the Zika paper?

We were interested in accessing the data and the results for the study titled

- Zika Virus Targets Human Cortical Neural Precursors and Attenuates Their Growth⁵

published in the journal Cell Stem Cell in 2016.

The screenshot shows the Cell Stem Cell journal website. At the top, there is a search bar with 'Search' and dropdown menus for 'All Content', 'Advanced Search', 'Cell Stem Cell', and 'All Journals'. Below the header, there are navigation links: 'Explore', 'Online Now', 'Current Issue', 'Archive', 'Journal Information', and 'For Authors'. The main content area displays the article 'Zika Virus Infects Human Cortical Neural Progenitors and Attenuates Their Growth' by Hengli Tang et al. The article is a 'Brief Report' from Volume 18, Issue 5, p587–590, 5 May 2016. It includes a DOI link (<http://dx.doi.org/10.1016/j.stem.2016.02.016>) and a CrossMark logo. On the right side, there are download options: 'PDF (1 MB)', 'Extended PDF (1 MB)', 'Download Images(.ppt)', and 'About Images & Usage'. Below these are links for 'Email Article', 'Add to My Reading List', 'Export Citation', 'Create Citation Alert', 'Cited by in Scopus (0)', 'Request Permissions', and 'Order Reprints (100 minimum order)'. At the bottom left, there is an Altmetric score of 1,954.

Figure 45.1

45.3 How do we find the data for this paper?

At the end of the paper we find the following statement in a more visible form (though we only found it by searching for the word GEO)

The accession number for RNA-seq data reported in this paper is GEO: **GSE78711**

⁵<https://www.ncbi.nlm.nih.gov/pubmed/26952870>

The GEO stands for the Gene Expression Omnibus⁶ and is a public functional genomics data repository that accepts array- and sequence-based data. Somewhat confusingly only the gene expression level results are actually stored in GEO, the sequence data is deposited into the Short Read Archive SRA. The projects are then linked but it also means that there are two locations for the data.

```
# Produce run information.
esearch -db gds -query GSE78711 | efetch
```

The default output is somewhat of a hodge podge information, if you need higher granularity use the `docsum` format, then `xtract`.

Connect the data to the sequencing data deposited to SRA:

```
# Crosslink to sra.
esearch -db gds -query GSE78711 | elink -target sra | efetch -format runinfo > runi
```

```
# Get the SRR run IDS for the data
cat runinfo.csv | cut -f 1,22 -d , | head
```

produces

```
Run,BioProject
SRR3191542,PRJNA313294
SRR3191543,PRJNA313294
SRR3191544,PRJNA313294
SRR3191545,PRJNA313294
SRR3194428,PRJNA313294
SRR3194429,PRJNA313294
SRR3194430,PRJNA313294
SRR3194431,PRJNA313294
```

So we found the two locations for the data:

- SRA BioProject PRJNA313294⁷.
- The GEO series GSE78711⁸.

Clearly we now have two numbers that we need to keep track of: PRJNA313294 and GSE78711. In addition, the scientific journal that

⁶<https://www.ncbi.nlm.nih.gov/geo/>

⁷<https://www.ncbi.nlm.nih.gov/bioproject/PRJNA313294>

⁸<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE78711>

published the results also distributes some information from another location. This third location, has no programmatic access to allow searches. You have to click around the site and find the URLs with a manual process. If you've made it this far and you have the prior knowledge, you can then save those links as command line actions like so:

```
curl -Okl http://www.cell.com/cms/attachment/2055417636/2061019276/mmc2.xlsx  
curl -Okl http://www.cell.com/cms/attachment/2055417636/2061019275/mmc1.pdf
```

The first file above is an Excel sheet. You would need to download, open and export each of the pages into a separate file. We did this for you

- List of regulated genes: <http://data.biostarhandbook.com/redo/zika/zika-up-regulated.csv>
- List of down regulated genes: <http://data.biostarhandbook.com/redo/zika/zika-down-regulated.csv>

You can now operate on the results of this paper from command line like so:

```
# Get the gene names for the study.  
wget http://data.biostarhandbook.com/redo/zika/zika-up-regulated.csv  
cat zika-up-regulated.csv | cut -f 1 -d , | head
```

produces the list of up regulated genes

```
gene  
ELAVL3  
FAM53C  
SLC22A17  
WDFY1  
TBL1XR1  
SON  
EIF4G2  
SOGA3  
SDCBP
```

Even identifying the location of the data and results is quite cumbersome, and obviously not quite ready for full automation. We think we have made our point again - currently data distribution is an ad-hoc and superficially supervised process that places a undue burden on the process of reproducibility.

45.4 What chapters cover the Zika data?

- Understand the Zika data
- Alignment based RNA-Seq of Zika data
- Classification based RNA-Seq of Zika data

Chapter 46

Redo: A synthetic-diploid benchmark for accurate variant-calling evaluation

46.1 Quick links:

- EBI BioProject: PRJEB13208¹
- SRA BioProject: PRJEB13208²
- Publication link: A synthetic-diploid benchmark for accurate variant-calling evaluation³ Nature Methods (2018)

Getting the runinfo for the project:

```
esearch -db sra -query PRJEB13208 | efetch -format runinfo > runinfo.csv
```

46.2 Why are we reproducing this paper?

We believe that some of the authors of this paper, for example Dr Heng and Dr. McArthur are among the most experienced practitioners of this field. We

¹<https://www.ebi.ac.uk/ena/data/view/PRJEB13208>

²<https://www.ncbi.nlm.nih.gov/bioproject/PRJEB13208>

³<https://www.nature.com/articles/s41592-018-0054-7>

consider them to be among the most skilled, most talented, and best trained computational scientists that work in life sciences.

46.3 Will the best bioinformaticians on the planet produce reproducible analyses?

Above we take the word “reproducible” in what we believe is the correct context. We believe that reproducibility means the effort needed to redo the same analysis and obtain the same results. Whether or not the results are scientifically valid is completely different question that we do not investigate here.

Now, having been told that these authors are the best of the best, of the few and proud, what do you think? Can we, at least, in the first step, obtain the same results as the authors did? Have the authors provided sufficiently detailed descriptions to reproduce the results?

Your Majesties! This humble ant wishes to follow your gigantic footsteps. Will you permit that?

46.4 What problem does the paper attempt to solve?

The accuracy of variant calling is of utmost importance. It is typically quite challenging to determine which method works best as we don’t know what the correct answer is. This paper states in the abstract:

We derived a new benchmark dataset from the de novo PacBio assemblies of two fully homozygous human cell lines, which provides a relatively more accurate and less biased estimate of small-variant-calling error rates in a realistic context



Brief Communication | Published: 16 July 2018

A synthetic-diploid benchmark for accurate variant-calling evaluation

Heng Li , Jonathan M. Bloom, Yossi Farjoun, Mark Fleharty, Laura Gauthier, Benjamin Neale  & Daniel MacArthur 

Nature Methods **15**, 595–597 (2018) | Download Citation 

Abstract

Existing benchmark datasets for use in evaluating variant-calling accuracy are constructed from a consensus of known short-variant callers, and they are thus biased toward easy regions that are accessible by these algorithms. We derived a new benchmark dataset from the de novo PacBio assemblies of two fully homozygous human cell lines, which provides a relatively more accurate and less biased estimate of small-variant-calling error rates in a realistic context.

46.5 Where is the data?

The paper clearly states that:

Illumina reads from this study were deposited in the European Nucleotide Archive under accession PRJEB13208⁴.

Listing the data explicitly is a great step in the right direction. Not only tells us the number but provides a link. The authors chose to upload the data

into EBI; thankfully the datasets are mirrored at NCBI and we are able to access the run information with `entrez direct` using:

```
esearch -db sra -query PRJEB13208 | efetch -format runinfo > runinfo.csv
```

46.6 Where is the code?

Unlike most other publications, the authors chose to create not just one but two GitHub repositories, and they state that:

Evaluation scripts are available from

- <https://github.com/lh3/CHM-eval>

The variant-calling pipeline and hard filters are implemented in

- <https://github.com/lh3/unicall>

We have the data and code repositories. Excellent!

46.7 Is the analysis reproducible?

While we commend the authors for publishing Github repositories that act alone is insufficient to reproduce the analysis. If you were to visit the repositories above you will be overwhelmed with overcomplicated commands. Among the many things we'd like to call out is the following abomination that demonstrates a total disconnect regarding common sense:

```
wget -q0- ftp://ftp.sra.ebi.ac.uk/vol1/ERA596/ERA596361/bam/CHM1_CHM13_2.bam \
| freebayes -f hs37.fa - > CHM1_CHM13_2.raw.vcf
```

There are so many things wrong with the code above that we have a hard figuring out where to start our critique. First, above the authors suggest that you download the BAM file each time you might want to run this evaluation. Which is an absurd idea, you would pay the cost of download over and over again. So let's fix that first, separate the download from the variant calling. Let's get the data first:

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/ERA596/ERA596361/bam/CHM1_CHM13_2.bam
```

Can you guess what happens? You'll get an error:

```
No such directory 'vol1/ERA596/ERA596361/bam'.
```

The link in the repository is incorrect. The first command of the published code on how to reproduce the analysis does not work! Gee thanks! After some investigation you'd find that the correct link is different:

- `ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR134/ERR1341796/CHM1_CHM13_2.bam`

Thus the correct download command is:

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR134/ERR1341796/CHM1_CHM13_2.bam
```

The code above shows me an estimated download of 3 hours. Note how the original command would have you pay the 3 hours each time you ran the command. Now, suppose that you have downloaded the file, run the second command:

```
freebayes -f hs37.fa CHM1_CHM13_2.bam > CHM1_CHM13_2.raw.vcf
```

Will it work? I am sure you know what to expect. Now you will get the following error:

```
could not open hs37.fa
```

Neither the first nor the second commands work as published. The file that `freebayes` attempts to open is the reference genome used in the variant calling process. There is no explanation of what the author used or how to get that data. We might guess from the name of `hs37.fa` that it is the *Homo Sapiens build 37*, but we do not know which particular variant of the genome it is. Also, should we really be guessing, after all this is a top notch and major publication and we can't even get past the first two commands without guesstimating what the author might have meant.

Ironically the author of the repository is the same person who, in past has pointed out, quite emphatically, how important it is to use the “correct” genomic build: Which human reference genome to use?⁵ So Dr. Heng which human genome are you using? And since we are discussing this choice, why are you using genome build that is more than ten years old? What sense does that make to evaluate your variants against a ten-year-old genome?

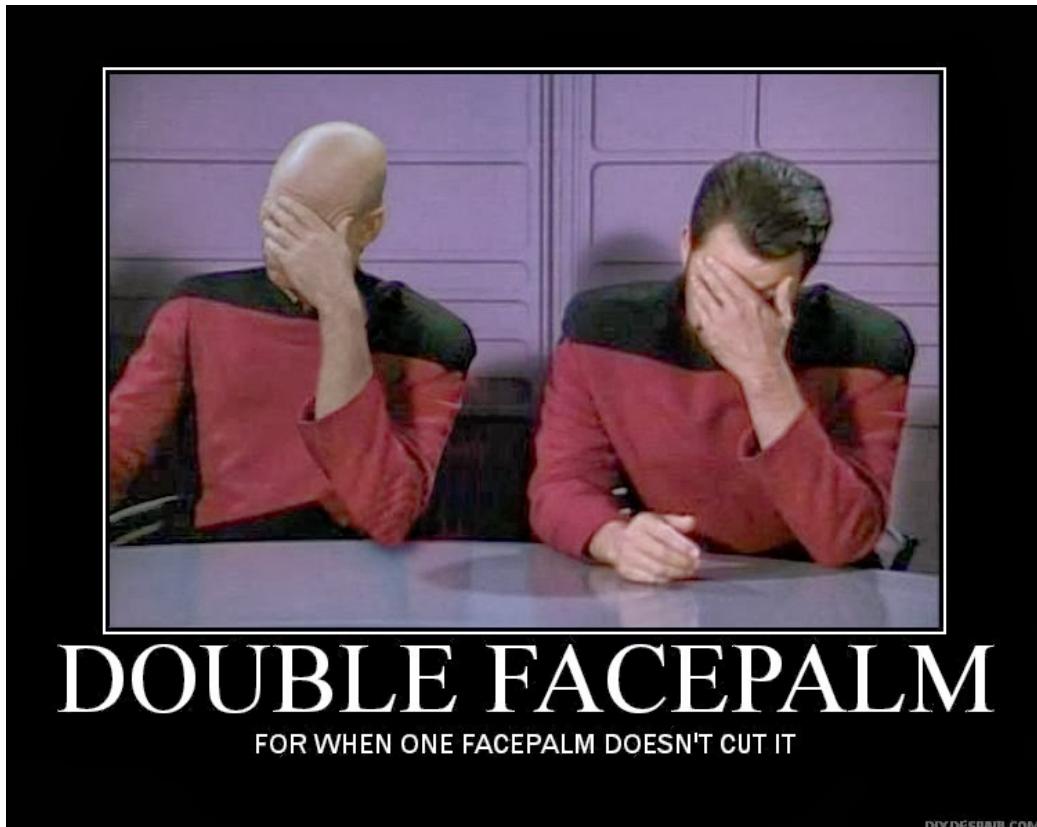
⁵<http://lh3.github.io/2017/11/13/which-human-reference-genome-to-use>

I hope we managed to demonstrate how even the very best bioinformaticians fail, often *spectacularly*, at producing reproducible analyses.

The pervasive lack of awareness when it comes to properly describing an analytical process has immense negative repercussions for life sciences, and is the problem that most urgently needs solutions. Even for an advanced bioinformatician reproducing the analysis that took place in this paper is mostly a mystery that they could only uncover at significant additional effort. From that point of view the analysis in this paper is no more reproducible than the Ebola paper written five years ago by scientists with supposedly far less awareness of computational reproducibility.

46.8 The gods must be crazy

We will revisit the results from this paper when we evaluate the variant calling accuracy. In the meantime we'll leave you with a sentiment that best captures the state of analysis reproducibility in bioinformatics:



DOUBLE FACEPALM

FOR WHEN ONE FACEPALM DOESN'T CUT IT

DIY.DESPAIR.COM

Chapter 47

Redo: Explore the genome of a paleolithic-era archaic human

(Work in progress, unfinished)

The Denisovans are a Paleolithic-Era members of the genus *Homo* that may belong to a previously unknown species. The genome sequence of a Denisovan individual which lived about 41,000 years ago was generated from a small fragment of a finger bone discovered in Denisova Cave in southern Siberia in 2008.

- Website: <http://www.eva.mpg.de/denisova>
- Data: <http://cdna.eva.mpg.de/denisova/>

Extract alignments that overlap with the 1 million bp wide interval starting on chromosome 3, at 120 million.

URL=http://cdna.eva.mpg.de/denisova/alignments/T_hg19_1000g.bam

```
# Get a subset of the data that overlaps with a region.  
samtools view -b $URL 3:120,000,000-121,000,000 > subset.bam  
  
# Index the BAM file.  
samtools index subset.bam
```

The Aryl Hydrocarbon Receptor binds to the location 17,375,390 on chromosome 7 of the human genome. Does the denisovan genome show variants in the reads that overlap with this position?



NOTE:

URL based web access is only helpful when you need to check a small subset of the data. If you need some properties of the entire dataset, you are better off downloading the data first, since you would need to stream it off the web anyway.

Web access can be useful to answer well-defined queries against a large number of datasets available on the web.

Part X

SEQUENCING INSTRUMENTS

Chapter 48

Sequencing instruments

Sequencing instrumentation has evolved at a dramatic pace.

In 2005 the 454 sequencing instrument jump-started the so-called “next-gen” sequencing revolution. Within just ten years, the 454 methodologies and the entire platform has been discontinued and supplanted by more efficient instruments.

48.1 Is there a document that compares sequencing instruments?

Travis Glenn’s Field Guide to Next Generation DNA Sequencer¹ (written in 2011) attempted to summarize the state of instrumentation at that time. There are yearly updates to this guide that (unlike the paper itself) are available for free.

- 2016: Updates to the NGS Field Guide²

¹<http://onlinelibrary.wiley.com/doi/10.1111/j.1755-0998.2011.03024.x/abstract>

²<http://www.molecularecologist.com/next-gen-fieldguide-2016/>

48.2 What is in a name?

You may have heard words such as next-gen, high throughput sequencing, massively parallel sequencing etc. For an eminently readable overview of the terminologies we recommend the blog post by Keith Robison titled:

- Beyond Generations: My Vocabulary for Sequencing Tech³

In the post Dr. Robison makes the case that sequencing instruments should be grouped by the technologies that they represent:

1. DNA source: Single Molecule vs. Clonal
2. Chemistry: Cyclic vs. Continuous
3. Physical surface: Beads, Wells, Surfaces, or Membranes
4. Signal type: Optical vs. Electrical
5. Measurement: Synthesis vs. Ligation vs. Pore Passing vs. Digestion vs. Hybridization vs.

Overall the blog post above will give you an quick overview of the methods of past, present and future.

48.3 What type of sequencing instruments are in use?

The following instruments are currently in use:

48.3.1 Illumina MiSeq, MiSeq, NextSeq, HiSeq

Illumina is the current undisputed leader in the high-throughput sequencing market. Illumina currently offers sequencers that cover the full range of data output. More details are available on the Illumina sequencers page.

- Up to 300 million reads (HiSeq 2500)
- Up to 1500 GB per run (GB = 1 billion bases)

³<http://omicsomics.blogspot.com/2019/02/beyond-generations-my-vocabulary-for.html>

48.3.2 IonTorrent PGM, Proton

The IonTorrent platform targets more specialized clinical applications.

- Up to 400bp long reads
- Up to 12 GB per run

48.3.3 PacBio Sequel

This company is the leader in long read sequencing. More details on the PacBio sequencers page.

- Up to 12,000 bp long paired-end reads
- Up to 4 GB per run

48.3.4 MinION

A portable, miniaturized device that is not yet quite as robust and reliable as the other options. More details on the MinION sequencers page.

- Up to 10,000 long reads
- Up to 240 MB per run (MB = 1 million bases)

48.4 What are obsolete sequencing instruments I should know about?

Two significant platforms – SOLiD and 454 – have been discontinued. Over time it is becoming increasingly difficult to analyze data produced by these platforms, as newer software packages do not support them. Older software sometimes fails to run at all.

Specifically, the SOLiD platform uses so-called “colorspace” data, a convoluted and indirect method to measure base identities that makes data analysis even more complicated.

48.5 How accurate are sequencing instruments?

Typically the longer read instruments operate at substantially higher error rates than those producing short reads.

- Illumina: 0.1% error rates (1/1000)
- PacBio: 10% error rates(1/10)
- MinION: 20% error rates (1/5)

Errors in long reads are easier to correct and account for than errors in short reads. Compare the error rates between the Illumina (left) and PacBio (right) instruments.

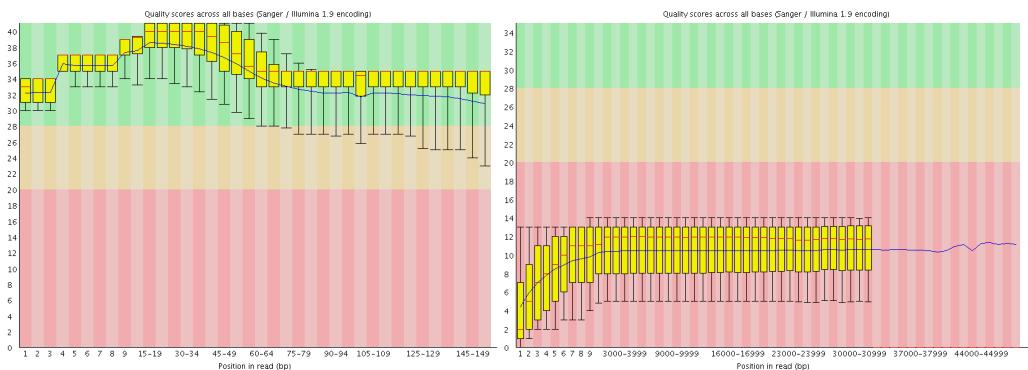


Figure 48.1

Read lengths are radically different across platforms:

48.6 How do sequencing instruments work?

A typical sequencer measures a single strand DNA fragment and from that fragment produces a “sequencing read” of a certain length. The term “read” is very widely used to define this single measurement.

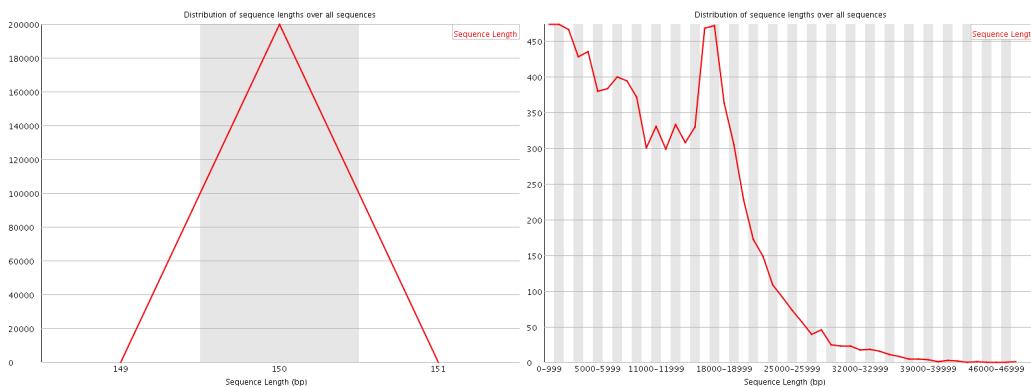


Figure 48.2

Essential detail: The resulting sequencing read is typically shorter but can be longer than the original fragment that it measures.

The read will end up shorter when the sequencing protocol reaches its limit before reading the end of the fragment. Having reads shorter than the DNA fragment is the most common expectation for short read sequencing approaches. The read may end up being longer than the original DNA fragment (some may also call this “template”) when the read runs into artificial constructs required by the sequencing protocols.

Below we will provide a sketch for the steps that go into a sequencing library for the HiSeq sequencer.

Suppose that the original, double-stranded DNA fragment obtained by isolating and fragmenting DNA from a cell is the following:

AAAACCCC
TTTTGGGG

During library preparation, uniquely designed DNA adapters of lengths that are typically over 30 bases are added to the 5' (right), and 3' (left) ends of each sequence.

Below we label these as XXXX and YYYY. Hence after adding the adapters the single-stranded sequences that make it onto the instrument flowcell will be

of the form:

XXXXAAAACCCCYYYY

the reverse strand's directionality is reversed; hence it will form:

XXXGGGGTTTYYYY

For any given fragment, both or neither of these strands may be sequenced.

The sequencer will typically recognize the XXXX at the beginning and will not report it. We can illustrate the process with arrows that show the direction of sequencing. Suppose that the sequencer can generate reads of lengths 5:

```
---->
AAAAT
AAAATGGGG
TTTAGGGG
AGGGG
<----
```

It is important to note that the size distribution of DNA fragments varies although the range typically is within 20-50bp. What this means, however, is that the fragment may occasionally be shorter than the measurement. When the read length is exactly as long a given fragment, we get to the following situation:

```
----->
AAAACCCYY
YYYYTTTGGGG
<-----
```

As the read length grows longer than the DNA fragment size, it will start running into the 3' adapter. Neither the sequencer nor software can immediately differentiate artificial DNA from the natural sequence, if for example the run-in is just one base long:

```
----->
AAAACCCYY
YYYYTTTGGGG
<-----
```

Hence our measurements above have artifacts at their ends. It is a situation of a so-called “read-through”, where the sequencing is longer than the frag-

ment. If the read-through is sufficiently long, say at least Five bases, we may attempt to remove these computationally after sequencing with tools that recognize the start of the adapter sequence.

48.7 Can reads be in different orientations?

Depending on sequencing protocols and instrumentations, data may end up with read orientations such as:

---->
---->

or even:

<----
---->

The vast majority of software tools cannot handle data with these orientations and may treat them in various incorrect ways.

It is relatively simple to convert data orientation. A simple post-processing operation can be used to reverse complement the reads into the most commonly used ----> <---- orientation.

48.8 What is paired-end sequencing?

Paired-end (PE) sequencing is a method to sequence both ends of a fragment and to make the pairing information available in the data.

DNA fragments are typically longer than the measured read lengths. For many applications, it is hugely advantageous to be able to measure (if not the entire piece) at least both ends of it. Many biological processes start at specific locations in the genome, knowing exactly where the fragment starts and ends can provide critically important information.

For that reason, some instruments offer the option of running the device differently to achieve two measurements from a single strand DNA.

48.9. WHAT ARE THE ADVANTAGES AND DISADVANTAGES OF PAIRED-END SEQUENCING

In the Illumina sequencing protocol, for example, the first round of reads are called the “single end” (SE) or “first” reads.

---->

AAAATTTGGGGCCCC

If the paired-end protocol is employed, after producing the “first” reads the *same* sequence is flipped over within the instrument, it is reverse complemented and a second measurement is taken to produce another set of reads. These are called the “second” reads.

---->

GGGGCCCCAAAATTTT

The net effect is that we obtain two measurements from one single stranded fragment:

---->

AAAATTTGGGGCCCC

TTTTAAAACCCCGGGG

<----

The two reads are typically stored in separate FASTQ files and are synchronized by name and order. Each read in file 1 has a corresponding entry in file 2.

Important: It is our responsibility to maintain this order and perform all operations on these files in such a manner that they are kept in sync and each contains the reads in the same order.

48.9 What are the advantages and disadvantages of paired-end sequencing?

Paired-end sequencing is more expensive than single end sequencing but not radically so (perhaps 20% more).

The ability to identify the ends of a DNA fragment typically carries a lot of importance, as biological processes always have a directionality. Besides, two measurements of the same fragment offer the user a better chance to correctly identify the location and composition of the original DNA fragment.

We recommend that genomic variation and genome assembly analyses use paired-end sequencing as much as possible.

On the downside, paired-end sequencing measures the same fragment twice, hence at the same genomic coverage, it will use half as many unique fragments. For analysis methods that use sequencing for quantification such as RNA-Seq and ChIP-Seq, this may be a disadvantage. Typically single end sequencing is used, as it produces more uniform and higher resolution coverage.

48.10 What is mate-pair sequencing?

Mate-pair sequencing typically has the same goals as the paired-end approach – it attempts to measure two ends of a fragment. The implementation, however, is entirely different and, unlike the paired-end sequencing, requires a better understanding of each particular instrument.

Notably, mate-pair DNA fragments are much much longer than PE methods and the read orientations are usually different, for example:

---->
AAAATTTGGGGCCCC
---->

Paired-end data is mainstream, and most tools support it directly. Mate-paired data is only supported by specialized software.

Chapter 49

Illumina sequencers

49.1 What types of sequencers does Illumina manufacture?

MiniSeq - the smallest bench-top sequencer Illumina sells (as of 2016). It only makes sense for those labs who have limited budgets and want to get started with high-throughput sequencing.

- Run time 4-24h (depending on the type of run)
- Up to 150bp long paired-end reads
- Up to 25 million reads per run
- Up to 8 GB per run (GB = 1 billion bases)

MiSeq - is perhaps the most popular instrument that started the trend for truly “bench-top” sequencers. MiSeq is small enough to fit on a regular lab bench and only requires a network connection (if you wish to use BaseSpace or off-load data from the instrument). Otherwise, it is entirely self-contained.

- Run time 4-55h
- Up to 300bp long paired-end reads
- Up to 25 million reads per run
- Up to 15 GB per run (GB = 1 billion bases)

MiSeqDX is a particular variant of MiSeq that is the first FDA-cleared high-throughput sequencer for *in vitro* diagnostic (IVD) testing.

- Currently there are two Cystic Fibrosis assays approved for use with MiSeqDX.
 - A panel of 139 clinically relevant variants for CFTR
 - A test for a comprehensive view of the CFTR gene

NextSeq 500/550 bench-top sequencer

- Run time 12-30h
- Up to 150bp paired-end reads
- Up to 400 million reads per run
- Up to 120 GB per run (GB = 1 billion bases)

HiSeq 2500/3000/4000 sequencers are the workhorses of sequencing. HiSeq 2500 is meant for large-scale genomics, and HiSeq 3000/4000 are meant for production-scale (think core facilities).

- Run time 1-3.5d (HiSeq 3000/4000)
- Run time 7h-6d (HiSeq 2500)
- Up to 150bp paired end reads (HiSeq 3000/4000)
- Up to 250bp paired end reads (HiSeq 2500 rapid mode)
- Up to 5 billion reads (HiSeq 3000/4000)
- Up to 300 million reads (HiSeq 2500)
- Up to 1500 GB per run (GB = 1 billion bases)

HiSeq X Five/X Ten sequencers: As the name suggests, there are 5/10 sequencers (sold as a package) to enable population-level genome sequencing. The HiSeq X instruments were initially only certified for human DNA sequencing.

- Run time 3d
- Up to 150bp paired end reads
- Up to 6 billion reads per run
- Up to 1800 GB per run (GB = 1 billion bases)

NovaSeq 5000 : Meant for counting applications

- New flow cells (S1, S2)
- Two-color chemistry like NextSeq 500
- 50, 100 and 150bp paired-end
- Up to 2 terabases data (1.6 billion reads) in 2.5 d
- Two flowcells per sequencer, can be run independently
- Same flowcells can be used for single-end or paired-end runs
- One library pool per flow cell

49.1. WHAT TYPES OF SEQUENCERS DOES ILLUMINA MANUFACTURE?427

- Four lanes per flowcell, visually distinct but optically identical
- On-board cluster generation

NovaSeq 6000 : High sequence coverage applications

- New flow cells (S1, S2, S3, and S4)
- S3 and S4 flow cells only usable in NovaSeq 6000
- Two-color chemistry like NextSeq 500
- 50, 100 and 150bp paired-end
- Up to 6 terabases data (10 billion reads) in 2 d
- Two flowcells per sequencer, can be run independently
- Same flowcells can be used for single-end or paired-end runs
- One library pool per flow cell
- Four lanes per flowcell, visually distinct but optically identical
- On-board cluster generation
- 132 exomes/transcriptomes per run

Chapter 50

PacBio sequencers

Pacific Biosciences is the leader in Single Molecule Real Time (SMRT) sequencing technology. Their *Sequel* platform is a high-throughput long read sequencer.

Note: On 1 November 2018, Illumina entered into a purchase agreement to buy PacBio, hence we expect substantial changes to take place with respect of this instrument.

50.1 What resources are available for PacBio data analysis?

Since PacBio offers longer reads than typical short read technologies, a new set of tools and pipelines are required for data analysis.

Resources:

- <https://github.com/PacificBiosciences/>
- <https://github.com/PacificBiosciences/Bioinformatics-Training/wiki>
- <http://www.pacb.com/products-and-services/analytical-software/devnet/>

50.2 What is an SMRT cell?

A SMRT cell consists of an array of measurement devices called Zero-Mode Waveguides (ZMWs). At the time of writing, *Sequel* system has one million ZMWs per SMRT cell.

50.3 What is a Zero Mode Waveguide (ZMW)?

A ZMW is a small hole with a unique structure in a SMRT cell that enables real time sequencing observation. It is a nanophotonic device with a diameter too small to permit propagation of light in the wavelength range used for detection.

50.4 What is a subread?

A subread is an individual measurement (read) that corresponds to the DNA template between the adapter sequences. The PacBio methodology allows for the same template to be measured multiple times, hence multiple measurements called subreads can be produced by a single ZMW waveguide.

50.5 How many subreads are produced from a template?

Divide the length of the DNA fragment (template) by the length of a read. For example, if the templates are 1KB and the sequencer produces 10KB reads, 10 subreads can ideally be produced from each template. This is a highly simplified case. Realistically we tend to observe: (TODO)

50.6 What is a Circular Consensus Sequence (CCS) read?

A CCS read is a consensus sequence obtained by the alignment of all subreads. When the template is shorter than the read length the polymerase will loop back, sequencing the template again. A sequence from a single pass is called a subread and multiple subreads can be used to obtain a single consensus sequence with much higher accuracy than that of the individual subreads. More than 2 full pass subreads are required to generate a CCS read.

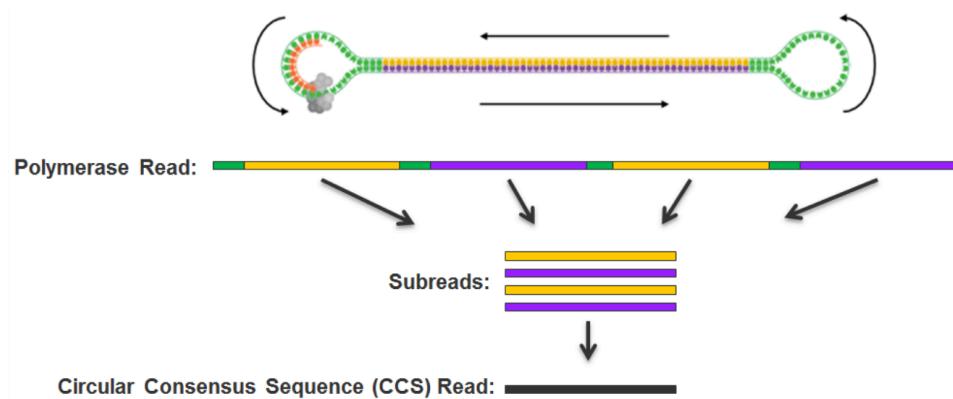


Figure 50.1

50.7 What is the output of a PacBio run?

A PacBio run folder contains unaligned bam files, adapter fasta file and metadata in xml format.

The primary output of a PacBio run is an unaligned BAM file called `subreads.bam`. The data in this file is to be used for downstream analysis. All rejected subreads, excised barcodes, and adapters are combined in a file called `scraps.bam`.

50.8 What other information is encoded in the BAM file?

PacBio-produced BAM files also encode instrument-specific information in the header and alignment sections.

50.9 What SAM headers does the instrument set?

- RG (read group) - 8 character string read group identifier for PacBio data.
- PL (platform) - contains “PacBio”.
- PM (platform model) - contains PacBio instrument series eg: SEQUEL.
- PU (platform unit) - contains PacBio movie name.
- DS (description) - contains some semantic information about the reads in the group, encoded as a semicolon-delimited list of “Key=Value” strings.

50.10 How is the BAM file formatted?

The query template name convention for a subread is:

{movieName}/{zmw-holeNumber}/{qStart}_{qEnd}

The query template name convention for a CCS read is:

{movieName}/{holeNumber}/ccs

50.11 What SAM tags are added to the reads?

- **qs** - 0-based start of the query in the ZMW read (absent in CCS).
- **qe** - 0-based end of the query in the ZMW read (absent in CCS).
- **zm** - ZMW hole number.

- **np** - NumPasses (1 for subreads, for CCS, it encodes the number of complete passes of the insert).
- **rq** - expected accuracy of the read.
- **sn** - average signal-to-noise ratio of A, C, G, and T (in that order) over the HQRegion

50.12 How do I get subreads from a single ZMW?

We can use the `zm` tag in the bam file to extract all subreads from a single zmw. This can be done with the *bamtools filter* command.

```
bamtools filter -in subreads.bam -out zmw.bam -tag 'zm:10027159'
```

The command will read the entire BAM file to extract the matching entries. This approach is reasonable only if we need to do this for one or two ZMWs. To split the BAM file into parts just by ZMWs, a custom program needs to be written.

50.13 Why would I want to split a BAM file by ZMWs?

The consensus caller (see below) operates on data for a single ZMW at a time. You can massively speed up the process by running the consensus caller in parallel on data that corresponds to each ZMW separately.

50.14 How do I get the consensus sequence from the subreads?

The `ccs` command from the Unanimity C++ library can be used to generate CCS reads. This command requires a minimum of 3 full pass subreads to a generate a CCS read.

```
# ccs [options] INPUT OUTPUT
```

```
ccs subreads.bam ccs.bam
```

50.15 What is the per-base quality of PacBio reads?

Whereas the quality of the original reads is low, the quality of consensus corrected reads improves dramatically.

The FastQC report plot below shows the per-base quality of some CCS reads.

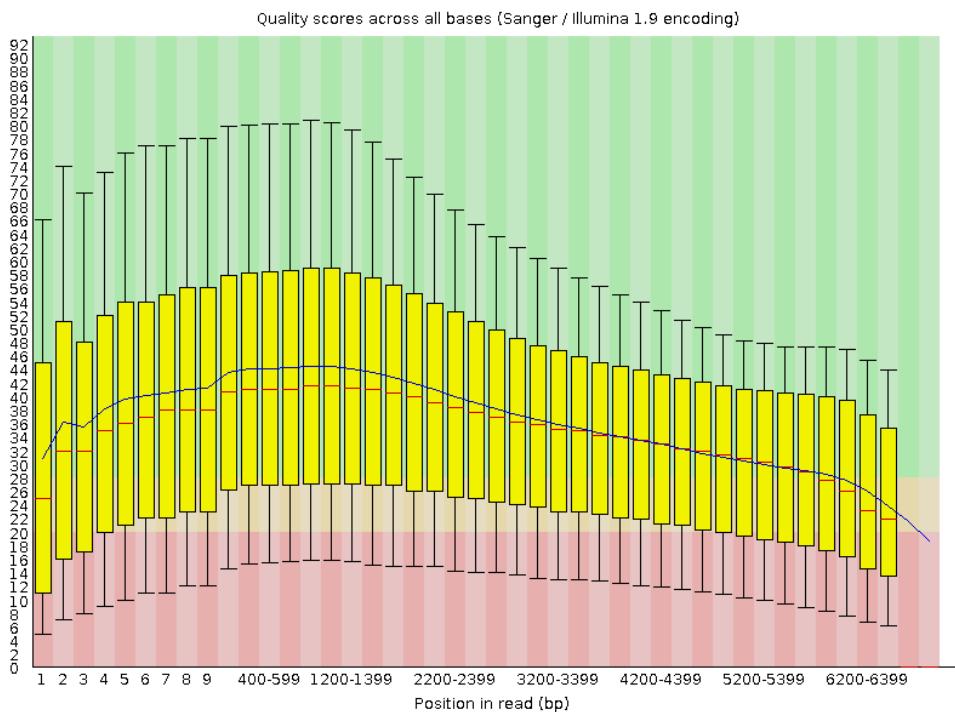


Figure 50.2

50.16 How good are the consensus corrected reads?

On the lower left is the alignment of subreads to the genome and on the right is the alignment of CCS reads.

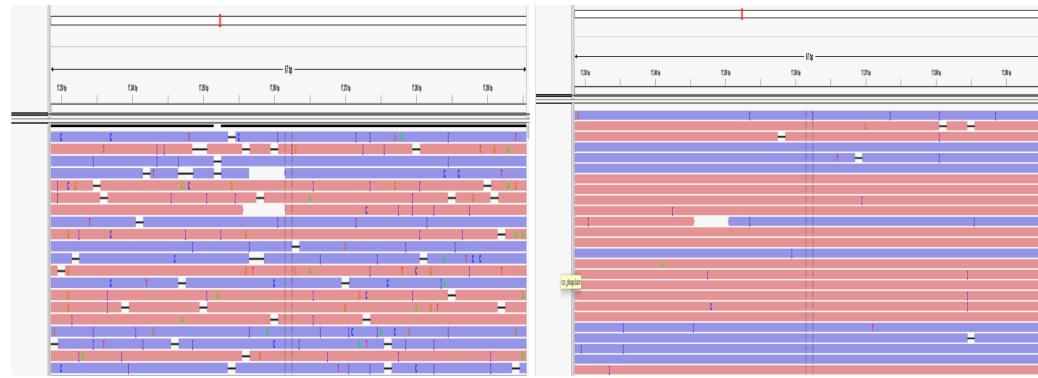


Figure 50.3

Chapter 51

Minion sequencers

Note: The MinION platform is undergoing rapid changes that we are unable to keep up with in a timely fashion.

51.1 What is MinION?

The MinION is a pocket-sized long read sequencing instrument from Oxford Nanopore technologies powered by a USB cable. It uses [nanopore sequencing] technology. Sequencing occurs by measuring changes in the ionic current as DNA translocates through protein nanopores in an insulated membrane on the flow cell. Below each nanopore channel is a very sensitive electrical sensor. A characteristic electric signal is generated by the flow of ions when no molecule is in the pore. When a strand of DNA or RNA passes through the pore, partially blocking the flow of ions, the current pattern will be disturbed in a manner characteristic to the molecule and dependent on the nucleotide sequence. The technology has essentially no size limitations – if you manage to keep a molecule intact during the library prep, reads up to hundreds of kilobases can be obtained. A standard library prep will routinely result in the majority of reads being greater than 10kb. Over the past few years, Nanopore sequencing technology has seen a rapid development with drastic improvements in throughput and accuracy. [nanopore sequencing]: https://en.wikipedia.org/wiki/Nanopore_sequencing

51.2 What is MinKNOW?

MinKNOW is the software that controls the MinION sequencer. The scripts in MinKNOW software carry out several tasks including device configuration, platform QC and calibration, data acquisition, etc. During the platform QC, the quality of the pores for sequencing is assessed and the most optimal pores will be selected for starting the run. A sequencing script starts the sequencing process. As the run progresses, it produces one FAST5 file per DNA molecule and stores it in the specified run directory on the SSD disk of your sequencing computer. It is possible to modify the (python) sequencing scripts to tweak the run and obtain higher yields, but this is something that is best left to those with considerable experience with the technology.

51.3 What is the Metrichor Agent?

The Metrichor Agent manages the connection to the base-calling service in the cloud (hosted by Metrichor). When Metrichor Agent is started on the sequencing laptop, it moves the pre-base-called FAST5 files to the specified uploads folder and then transfers a copy to the base-calling service. The FAST5 file with original information and base calls is then returned and stored in the specified download folder on the sequencing computer. In addition to cloud base-calling, local alternatives are available. It's expected that cloud base-calling will be discontinued in the near future. Other applications are available in the cloud – such as the What's In My Pot (WIMP) workflow, in which a metagenomic sample is classified based on the obtained sequence to identify species contained therein.

51.4 What is the output of a MinION run?

Each read sequenced on the MinION will generate a FAST5 file, which is a variant of the [HDF5] file format. This format is essentially a container for storing a variety of data types (eg. integers, floats, strings, arrays) in a single file. The format has a hierarchical structure with ‘groups’ for organizing data objects and ‘datasets’ which contain a multidimensional array

of data elements. These files contain voltage and time information, metadata identifying the pore and flow cell used for sequencing, among other things. This data allows for base-calling of the reads, either in the cloud or on your local system. In addition, when new base-calling software is available, the analysis can be repeated. Third-party base-callers for nucleotide modifications such as methyl-cytosine are also available. [HDF5]: <https://support.hdfgroup.org/HDF5/>

51.5 What do ‘pass’ or ‘fail’ subfolders contain?

If you activate the quality filter at the start of a run, the base-called data will get stored into ‘pass’ or ‘fail’ subfolders in the specified download folder. The pass/fail classification system is intended to help the user distinguish higher and lower quality data. Lower quality data isn’t necessarily useless – depending on your application this data may still be valuable. The classification criteria may change between different versions of Metrichor.

51.6 What is different between 1D and 2D reads?

In general two library preparations are possible, termed 1D and 2D sequencing. The highest accuracy is obtained by ligating a hairpin adapter on one side of the molecule, hence sequencing the double-stranded DNA in both directions (2D). The first strand is called the template read, the second strand the complement. Since a consensus read can be created based on both strands, the accuracy of a 2D read is higher. For a 1D library prep, no hairpins are ligated and sequencing is performed for each strand separately, effectively increasing the throughput but sacrificing some accuracy. Since May 2016, the accuracy of 1D reads has reached acceptable levels.

51.7 How do I extract data from a FAST5 file?

Different commands and tools are available to extract data from FAST5 file.

* HDFView * poretools * poRe

51.8 What is HDFView?

HDFView provides a Graphical User Interface (GUI) to browse the contents of a FAST5 file. Shown below is an HDFView display of a 2D read sequence.

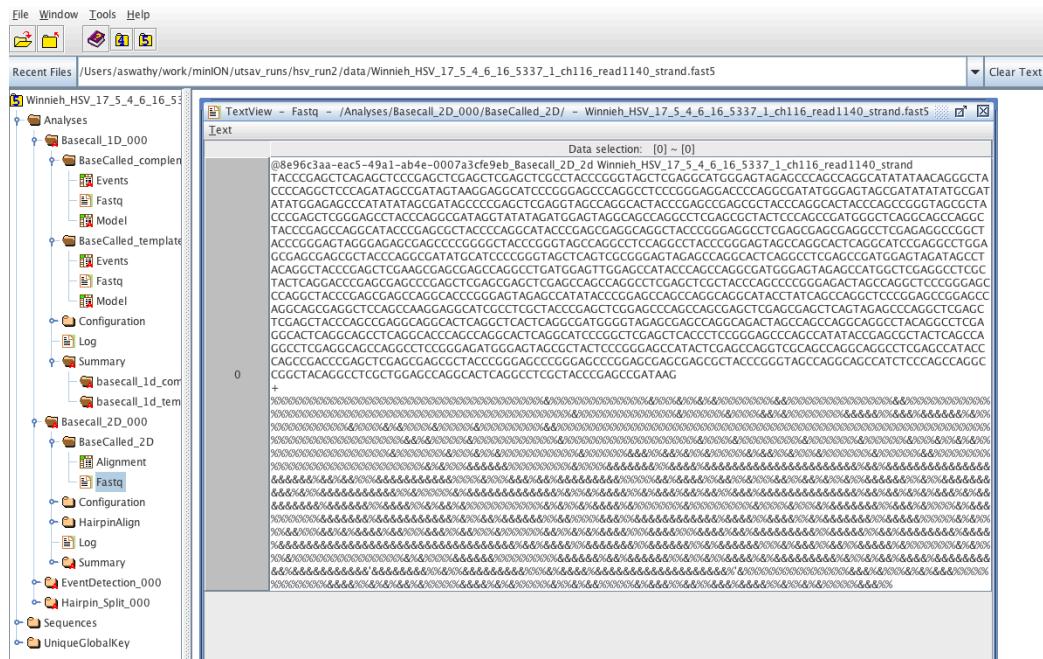


Figure 51.1

51.9 How can I extract data from a FAST5 file using poretools?

- Poretools is a toolkit for extracting data from fast5 data¹, written by Nick Loman and Aaron Quinlan

Type the command with no arguments to get the basic usage.

```
poretools
```

```
usage: poretools [-h] [-v]
{combine,fastq,fasta,stats,hist,events,readstats,tabular,nucdist,qualdist,winner,squiggle,times,yield_plot,occupancy}
...
poretools: error: too few arguments
```

To extract FASTQ sequences from a set of FAST5 files in a directory ‘fast5-data’

```
poretools fastq fast5-data/
```

Extract all 2D reads from FAST5 files:

```
poretools fastq --type 2D fast5-data/
```

To get the length, name, sequence, and quality of the sequence in one or a set of FAST5 files, we can use the ‘tabular’ command.

```
poretools tabular data.fast5
```

¹<http://bioinformatics.oxfordjournals.org/content/early/2014/08/19/bioinformatics.btu555.abstract>

Chapter 52

Sequencing data preparation

52.1 From sample submission to receiving sequence data

Author: **Hemant Kelkar**

Note: Discussion below draws upon personal observations over the past few years. Your experience may vary due to practices prevalent at the sequencing facility you use.

Getting samples ready for high-throughput sequencing is a result of a days/weeks/months-long effort on your part but suffice it to say that a significant amount of money and labor has already been expended on that process.

52.2 A recap of high-throughput sequencing

High-throughput sequencing data broadly fall into two categories:

- Short reads (300 bp or smaller).
- Long reads (>350 bp to several kilobases).

What goes into a high-throughput sequencing experiment may be familiar to many wet-lab scientists.

Here is a quick summary:

52.3. WHERE CAN I FIND ILLUSTRATED SUMMARIES OF SEQUENCING TECHNOLOGIES?44

- Sequencers sequence sheared and purified DNA fragments.
- Addition of technology-specific adapter sequences (oligos) create “sequencing-ready libraries”.
- Controlled amplification (PCR) is required with most prep methods (PCR-free methods are available for some applications).
- Sequencing is done by DNA synthesis (except for Oxford Nanopore and some optical mapping technologies).
- Detection of synthesis is done using light (Illumina, PacBio) or non-light (Oxford Nanopore, Ion) methods.
- Post-processing of “raw” data is required to generate the actual sequence.

Note: Direct RNA sequencing has been demonstrated with some technologies, but no commercial applications are currently available.

Illumina sequencing is by far the most common high-throughput sequencing technology currently in use.

- It has been commercially available for almost a decade.
- Sequencers and sequencing kits have evolved to cover the entire spectrum of data yields.
- Error profiles are reasonably well understood.
- Sequencing is relatively cost-effective.

Ion technology is used for specific applications and when fast data turnaround is a high priority.

Pacific Biosciences implements a mature long read technology that has been on the market for some time. Oxford Nanopore (another long read method) is gaining interest in academic labs at a rapid pace.

52.3 Where can I find illustrated summaries of sequencing technologies?

The following videos illustrate currently popular sequencing technologies:

- Illumina, Intro to Sequencing by Synthesis (HiSeq 2500)¹

¹<http://www.youtube.com/watch?v=HMyCqWhwB8E>

- Illumina, Patterned Flow Cell Technology (HiSeq 3000,4000,X)²
- Ion Torrent next-gen sequencing technology³
- Pacific Biosciences (SMRT Technology)⁴ and Introduction to SMRT Sequencing⁵
- Oxford Nanopore Sequencing Technologies⁶

52.4 How do I select proper sample identifiers?

Do NOT use spaces or the following characters (? () [] / \ = + < > : ; " ' , * ^ | & .) in sample identifiers. These characters can cause problems with Illumina (and/or other technology) data pre-processing software and should be avoided.

DO use sample names or identifiers that make immediate sense to you. It is best to avoid personal identifiers of any kind to prevent regulatory issues. If you use “Sample1/Sample2” for multiple submissions, the data files that you receive will have those exact names. While facilities will keep the data segregated, you may have a hard time keeping track of ‘what is what’ down the road.

Sequencing cores may track samples using internal IDs (as part of a LIMS or otherwise), but the sample name you submit will be carried forward through the process. In the case of Illumina technology, this name may get incorporated into the sequence file itself.

52.5 How to choose the right sequencing method?

The number of cycles of sequencing and read type (single-end/paired-end) determines the kind of data you will receive. The kind of library you made

²<https://www.youtube.com/watch?v=pfZp5Vgsbw0>

³<http://www.youtube.com/watch?v=WYBzbxIfuKs>

⁴<http://www.youtube.com/watch?v=v8p4ph2MAvI>

⁵<http://www.youtube.com/watch?v=NHCJ8PtYCFC>

⁶<http://www.youtube.com/watch?v=3UHw22hBpAk>

52.6. WHY IS A SUBMISSION OF SAMPLE METADATA CRITICAL?443

is also important. All of these choices have a direct impact on the cost of sequencing. In general, longer and paired-end reads (Illumina) and very long reads (PacBio) are comparatively more expensive.

It is not possible to cover the entire breadth of sequencing choices in this section.

Here is a representative example. If you are only interested in read counts and differential gene expression (e.g., RNAseq), then single-end reads of a reasonable size may be enough. You will not get spatial information about fragments unless you choose to do paired-end sequencing. If you were interested in identifying full-length alternately spliced transcripts, then Pacific Biosciences IsoSeq would be the method of choice (assuming there are no budget limitations).

52.6 Why is a submission of sample metadata critical?

Sample metadata (the type of sample, specific sample characteristics, experiment type) that you include with your sample submission may play a crucial role in the success of sequencing.

Illumina sequencing technology assumes a reasonably uniform distribution of nucleotides (A/C/T/G, ~25% each) in your samples. If your sample is known to deviate from this assumption (e.g., amplicons, metagenomics, high GC/AT samples) then make a note of this during sample submission. A “spike-in” may need to be added to such samples (generally phiX is used, but any “normal” genomic DNA may be used) to address “low nucleotide diversity” (for sample examples noted above).

Ion/Oxford Nanopore technologies are not very robust when homopolymers (e.g., AAAAA) of significant length are present in samples. They would not be appropriate for such samples.

52.7 Why is sample/library QC essential?

Qubit and Bioanalyzer analysis are preferred options (qPCR may also be used) for analyzing the quality of starting material and finished libraries. Not all labs have access to these technologies, and hence they may use alternate means for QC (e.g., nanodrop). The amount of material you think you are submitting invariably turns out to be less (sometimes 50% or more), which can directly affect data yields.

Final sequencing data quality depends on the quality of samples (garbage in, garbage out). It may be worth asking the sequencing facility to do QC via their methods – some facilities may do this as a part of their standard sequencing protocol. There may be a charge for this QC, but it sure beats having sequencing fail because of samples with lousy quality.

Making good sequencing libraries is an art. If you intend to keep making libraries for many samples, then it would be cost effective to acquire this expertise in house. Otherwise, it is best to get a professional (at a core or elsewhere) to take care of this crucial step.

52.8 What can large-scale projects do to ensure sample data integrity?

If you are submitting a large number of samples (e.g., samples in multi-well plates), then it is useful to think about having an independent means of identifying samples from sequence data by correlation with an independent parameter (e.g., genotype). While sequencing facilities do take precautions in handling plates/large batches of samples, a human is involved in some steps and errors will sometimes occur.

52.9 What happens during a sequencing run?

Once your sample libraries clear these initial checks, they are ready to run on the sequencer. As noted elsewhere the machine run times may vary from a few hours to a week depending on the type of run and model of the sequencer.

Illumina sequencers use an elegant combination of precision engineered components, motors, manifolds, lasers, optics, and proprietary onboard software to capture sequence data. It is possible to monitor the sequencing run both locally and remotely (with the right set up) in real time.

Illumina flowcells are imaged and processed in real time on the workstation (or internal computer) attached to the sequencer. Per-cycle base calls generated are saved in an elaborate folder structure that contains data for each flowcell. The naming convention for the folders is in this format: YYDDMM_SequencerName_RunNo_FlowcellBarcode (FlowcellBarcode is tagged in front with A or B in case of sequencers that can run two flowcells at one time).

52.10 What does an original Illumina sequence data folder contain?

Depending on the run type (length, flowcell, sequencer type) a raw data folder can contain between 25 to ~550 GB of data. There can be hundreds of thousands of files, in multiple, nested folders, that follow an identical layout for every flowcell. This folder structure allows for scripting and workflow based processing of data. Individual file sizes range from a few bytes to tens of megabytes. Some sequencing instruments (e.g., MiSeq) may store some imaging data that are useful for diagnostic purposes in case of run failure.

52.11 How does one get “finished” data from “raw” sequence data?

Smaller sequencing cores may use the onboard data analysis capabilities of sequencers (MiniSeq/MiSeq/NextSeq) and BaseSpace, Illumina’s cloud computing solution for management of sequence data. If an institution can’t use third-party cloud computing solutions, a locally installed variant of BaseSpace called “BaseSpace Onsite” is also available as an option. BaseSpace uses “apps” to provide added analytics functionality beyond the processing of raw sequence data.

Larger cores generally use **bcl2fastq**, which is Illumina's software for converting the per-cycle base calls (stored in BCL files) into the fastq format. **bcl2fastq** is currently available in two versions (v. 1.8.4 for older sequencers, v. 2.18.x for newer sequencers), but is backward compatible with all current Illumina sequencers.

bcl2fastq does per-cycle BCL file to per-read FASTQ format conversion and simultaneous demultiplexing of samples. **bcl2fastq** can optionally trim adapters and remove Unique Molecular Identifiers (UMI) from reads.

bcl2fastq conversion requires access to a cluster (or a high-performance multicore computer). These conversion runs may take between 30 min to a few hours (depending on the type of run). The result of a **bcl2fastq** run is a folder of sequence files and several report/results files/folders. These are used by sequencing facilities to do internal quality control of sequence data. Once a sample passes these internal QC measures, it is generally ready for release to the submitter.

52.12 What should I expect to get from a sequencing provider?

The primary deliverable for high throughput sequencing is a gzip compressed FASTQ formatted sequence data file(s) for each sample. You will get as many files as the number of samples you had submitted.

Filename formats may vary depending on how the local pipeline is run. The file may be named something like:

`SampleID_ACGGCTGAC_L001_R1_001.fastq.gz`

If you requested paired-end sequencing, then you should expect to get two files per sample. The order of the reads within the files indicate the read pairings.

`SampleID_ACGGCTGAC_L001_R1_001.fastq.gz`
`SampleID_ACGGCTGAC_L001_R2_001.fastq.gz`

Note: In a rare instance, you may receive R1/R2 reads in an interleaved format in a single file (R1_r1, R2_r1, R1_r2, R2_r2, etc).

In case your samples are not multiplexed “Barcode/Tag” (ACGGCTGAC above) in the filename may be replaced by the word “NoIndex”. Newer versions of `bcl2fastq` do not put the “Barcode” in the file name by default, so it may not always be present.

L001 indicates the lane your sample ran in. 001 indicates a single data file per sample. Illumina software allows sequence data files to be split into defined chunks (2 Million reads per file is default). If that is the case, you will get a series of files (with the same SampleID) where 001 in the file name will be incremented for each file chunk for that particular sample (001 through nnn).

52.13 How is the sequencing data delivered?

This will largely depend on your sequencing provider. Sequencing data files (even when compressed) can be fairly large (tens of GB per sample). You will need adequate internet bandwidth to download the files directly. Ideally, your provider will create MD5 hashes for your files so that you can verify the file integrity after download.

Commonly used data delivery options are :

- Delivery via a storage partition mounted on a local computer cluster (if you are using an internal sequencing facility).
- Via the Internet (web links)
- Illumina BaseSpace (if it is in use at the sequencing facility). You will need a free BaseSpace account to access data.
- Amazon/Google/Microsoft/Other cloud storage solution. Preferred, if data needs to be shared with many people.
- Shipping a copy of the data on external hard drives may still be an (or only) option for some who are security conscious, don’t want their data via the internet, or do not have adequate bandwidth

52.14 What should I do with the raw data?

You should ensure there is a backup copy for internal needs. External drives are fine for small lab groups (keep two copies on separate drives, for added

security). Consider submitting the raw data to NCBI SRA as soon as is practical. You can request an embargo period (before the data becomes publicly available) to allow you to complete the analysis and write a paper.

Many facilities keep internal backups of raw sequence and processed data files for varying periods (ask about your facilities' data policy). Bear in mind that retrieval of data may incur an additional charge, beyond a specified period.

52.15 Where can I download Illumina software?

`bcl2fastq` (and many other software packages) can be obtained from Illumina's portal (iCom⁷) by creating a free account. Most Illumina software is available at no cost. Software obtained from iCom is NOT open source. The only software hosted on iCom is software supported by Illumina technical support.

Illumina maintains a GitHub repository (Illumina Repo⁸) specifically for open source software. Open source software on GitHub repo is NOT supported by Illumina technical support.

⁷<https://accounts.illumina.com/?ReturnUrl=http://icom.illumina.com/>

⁸<https://github.com/Illumina>

Part XI

SEQUENCING DATA

Chapter 53

Sequencing coverage

53.1 How much data do I need?

The question of how much data one needs is one of the first questions that everyone asks. The answers to this question are always a bit complicated and depend on the goals of the experiment.

The amount of data is typically expressed as “coverage” and represents how many measurements (on average) will be taken from each base of the genome. The coverage is typically expressed as “x” (times) symbol. 10x would mean each base is, on average sequenced ten times.

53.2 What are typical coverages for DNA sequencing?

- Genome assembly 50x and above
- Variation calling (resequencing) 20x and above.

Always consider how many measurements you might need to support a statement. For example, at a 20x resequencing project, in a diploid organism, a heterozygous mutation would be, on average covered, by ten reads.

53.3 What are typical coverages for RNA sequencing?

In RNA-Seq experiments different transcripts are expressed at different levels; hence a formula based coverage is not directly applicable. Consult the RNA-Seq data analysis chapter for more details.

In a nutshell, and at the risk of oversimplifying the problem, for the same sequencing throughput, the number of replicates is more important than the number of reads (coverage) per sample. For human/mouse genomes 10 million reads per replicate (and 3 or more replicates) usually suffice. For different transcriptome sizes scale this number accordingly. Ten times smaller transcriptome will usually need ten times fewer reads per sample.

The most reliable way to asses the required coverage for a specific and perhaps novel analysis is to download published datasets on the same organism and tissue and evaluate the results. Thankfully the process of obtaining published data lends itself to automation as demonstrated in Automating access to SRA.

53.4 How is genome coverage computed?

The simple definition of coverage is

$$C = \text{number of sequenced bases} / \text{total genome size}$$

The only slightly confounding factor is that the number of sequenced bases come in units of “read lengths” rather than one base at a time.

Typically the coverage is expressed with the \times symbol. For example, 10 \times indicates that on average, each base of the genome will be covered by 10 measurements.

For a sequencing instrument that produces variable reads lengths we have:

$$C = \text{SUM}(L_i) / G$$

Where the sum goes over N the read count:

$$L_i = \text{length of read } i$$

$$G = \text{size of the genome}$$

For an instrument with fixed read length L , that produces N reads the formula simplifies to:

$$C = N * L / G$$

Example

What is the coverage of a 20KB virus if we measured 1000 reads of 50bp each?

$$C = 50 * 1000 / 20000 = 2.5$$

The coverage is $2.5x$ that is on average each base is sequenced 2.5 times. If we needed to raise that coverage to $10x$ we could do the following:

1. Increase the read lengths.
2. Produce more reads.
3. A combination of both of the above.

We can invert the formula and compute it, but usually, it is straightforward to see by eye what needs to be done. To get to $10x$ we need to increase the coverage by a factor of 4. We could choose to:

1. Increase $L=200$ and keep $N=1000$
2. Keep $L=50$ and increase $N=4000$

Usually, there are various real-life constraints on how the instrument operates and a combination ends up being chosen such as $L=100$ and $N=2000$

53.5 How many bases have not been sequenced?

The coverage is an average metric and has variability in it. At a given coverage some bases will likely not get covered. There are formulas to compute it based on a theoretical model called the Lander/Waterman model (random fragmentation).

At a coverage C the probability of a base not being sequenced is:

$$P = \exp(-C)$$

Example

How many bases will not be sequenced when we cover a 20KB virus at 10x coverage.

```
P = exp(-10)
```

From command line:

```
python -c "import math; print (math.exp(-10))"  
# Prints: 4.53999297625e-05
```

To figure out how many bases will not be sequenced, we need to multiply this result by the genome length:

```
python -c "import math; print (math.exp(-10) * 20000)"  
# Prints: 0.90799859525
```

So at 10x coverage, we will miss about one base. When we cover the human genome at the same coverage:

```
python -c "import math; print (math.exp(-10) * 3E9)"  
136199.789287
```

We will miss 136,199 bases. Note, though, that these gaps will be theoretically dispersed across the entire genome.

53.6 Do theoretical coverages describe reality?

Not really. Think of theoretical coverages as the ideal, absolute best case outcomes. Reality is typically a lot more complicated. DNA does not fragment randomly and may exhibit various non-random patterns. We don't sequence the genome directly; instead, a series of complex material extraction and preparation protocols need to be performed that will show preferences toward some parts of the DNA. Notably:

1. Increase your coverages well over the theoretical limits.
2. Sometimes some parts of the genome do not show up in the data. These are called *hard to sequence* regions, but it is not always clear why they are hard to sequence.
3. What part of the genome is uniquely coverable with a given read size? Repeat-rich regions of the genome require longer reads to resolve.

Scientists often use terms such as “accessible,” “mappable,” “effective” genome to indicate that only some parts of the genome can be easily studied.

Chapter 54

Accessing the Short Read Archive (SRA)

54.1 What is the Short Read Archive (SRA)?

The Short Read Archive (SRA)¹ is a data storage service provided by NCBI. The history of the SRA’s development is relatively convoluted and includes a widely announced, promoted, then retracted cancellation of the service. In parallel, the data design and interface were standardized prematurely, before the community fully understood how scientists might need to access the data. The incredible rate of growth of the deposited data locked and froze those standards in place. Today it seems that it would be hard to change them without inconveniencing a large number of people. No wonder there is an entire “cottage industry” of tools trying to bypass and automate access to SRA.

54.2 What are the SRA naming schemes?

Data in the SRA is organized under a hierarchical structure where each top category may contain one or more subcategories:

¹<http://www.ncbi.nlm.nih.gov/sra>

- NCBI BioProject: PRJN**** (example: PRJNA257197) contains the overall description of a single research initiative; a project will typically relate to multiple samples and datasets.
- NCBI BioSample: SAMN**** or SRS**** (example: SAMN03254300) describe biological source material; each physically unique specimen is registered as a single BioSample with a unique set of attributes.
- SRA Experiment: SRX**** a unique sequencing library for a specific sample
- SRA Run: SRR**** or ERR**** (example: SRR1553610) is a manifest of data file(s) linked to a given sequencing library (experiment).

54.3 How do we download data from SRA?

You can download data via a web browser (after installing a web-plugin) or from the command line via the `sratoolkit`² package that can be installed via `conda`.

54.4 Are there alternatives to the SRA?

The European Nucleotide Archive (ENA)³ provides an alternative repository to store the world's nucleotide sequencing information, covering raw sequencing data, sequence assembly information, and functional annotation. The naming schemes and the organization of the data mirror those of the SRA, though the files are stored directly as FASTQ and BAM formats and do not need conversion via the `ssratoolkit`.

54.5 Where is the SRA documentation?

- The SRA Handbook⁴ contains a detailed description of the Short Read Archive.

²<https://www.ncbi.nlm.nih.gov/sra/docs/toolkitsoft/>

³<http://www.ebi.ac.uk/ena>

⁴<http://www.ncbi.nlm.nih.gov/books/NBK47528/>

- The SRA Toolkit Documentation⁵ describes the command line tools that may be used to access data.
- SRA Toolkit GitHub⁶ may have more up-to-date information on troubleshooting.

54.6 How does the sratoolkit work?

There are several tools (see the tool directory). The most commonly used ones are:

- `fastq-dump`, downloads data in FASTQ format
- `sam-dump`, downloads data in SAM alignment format

A typical use case is

```
fastq-dump SRR1553607
```

This creates the file:

```
SRR1553607.fastq
```

By default, the SRA data will concatenate the paired reads. For paired-end reads the data needs to be separated into different files:

```
fastq-dump --split-files SRR1553607
```

This creates the paired end files:

```
SRR1553607_1.fastq
SRR1553607_2.fastq
```

The `fastq-dump` command does two things in sequence. It first downloads the data in the so-called SRA format; then it converts it to FASTQ format. We can ask the tool to convert just a subset of data; for example, the first 10,000 reads:

```
fastq-dump --split-files -X 10000 SRR1553607
```

But we note that even in that case, `fastq-dump` needs first to download the full data file.

⁵https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc

⁶<https://github.com/ncbi/sra-tools>

54.7 Is it possible to download the SRA files from a server?

Yes, there is a website called `ftp://ftp-trace.ncbi.nih.gov/sra/sra-instant` that uses an encoding system to map an SRR number to a URL. For example, SRR1972739 can be downloaded as:

```
wget -nc ftp://ftp-trace.ncbi.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR197/SRR1972739.sra
```

The command above will download a file named `SRR1972739.sra`. The downloaded file can be operated on the same way with `fastq-dump`:

```
fastq-dump -X 10000 --split-files SRR1972739.sra
```

54.8 Where do downloads go?

When any download via `fastq-dump` is initiated, the ** complete** data for that run id will be downloaded to a location on your computer. By default this location is at `~/ncbi/public/sra/`. There may be other sequences related to an SRA file, in which case those too will be downloaded. To investigate the downloaded SRA data do:

```
ls -l ~/ncbi/public/sra/
```

After the download is completed, a conversion takes place according to the parameters.

Note: Even if we only asked for the first ten reads of a FASTQ file, the entire FASTQ content would need to be downloaded to the “hidden” location then out of that the first ten reads will be shown. For large datasets, this may take a considerable amount of time and may use up a substantial amount of disk space.

Now, if we were to access the same SRA run the second time, the data would already be there, and conversion would be much faster.

54.9 What is a “spot” in SRA?

SRA tools use the word “spots” and not as “reads” (measurements), though the two almost always seem to mean the same thing. Answers to the Biostar question: What Is A “Spot” In Sra Format?⁷ shed some light:

“The spot model is Illumina GA centric. [...] All of the bases for a single location constitute the spot.”

As it turns out, the terminology is both confusing and may not apply to other instruments.

In the Illumina sequencers, a single location on the flow cell generates the first read, the barcode (if that exists), and the second (paired) read (if the run was a paired-end sequencing run).

We almost always split the spots (via the `--split-files` option) into originating files.

54.10 How do we get information on the run?

The `sra-stat` program can generate an XML report on the data.

```
sra-stat --xml --quick SRR1553610
```

Get read length statistics on a PacBio dataset:

```
sra-stat --xml --statistics SRR4237168
```

⁷<https://www.biostars.org/p/12047/>

Chapter 55

Automating access to SRA

In this chapter, we demonstrate how to automate access to the Short Read Archive for the purpose of obtaining data deposited with scientific publications.

Around the end of the chapter we show how to get information on all the data stored in SRA and compile various statistics on them! The first version of the book was written in the Fall of 2016. We revisited this same chapter and updated the numbers in the Spring of 2019. How much did the numbers change? A lot more than I expected. The data floodgates are open my friends!

55.1 How to automate downloading multiple SRA runs?

The primary strategy is to first obtain the project PRJN number, then from the project numbers get the SRR run numbers, then run `fastq-dump` on each SRR number.

So start with the bio-project number found in the publication. For example, in Redo: Genomic surveillance elucidates Ebola virus origin we noted the that project number is PRJNA257197

```
esearch -db sra -query PRJNA257197
```

The command above returns a so called “search result” that tells us that there are 891 sequencing runs for this project id. The search result is an XML file that you can later pipe into other programs:

```
<ENTREZ_DIRECT>
  <Db>sra</Db>
  <WebEnv>NCID_1_74789875_130.14.18.34_9001_1474294072_501009688_0Meta0_S_Mega
  <QueryKey>1</QueryKey>
  <Count>891</Count>
  <Step>1</Step>
</ENTREZ_DIRECT>
```

We can now efetch and format the search result as so-called `runinfo` data type. The `runinfo` type is in CSV, comma separated value, format:

```
esearch -db sra -query PRJNA257197 | efetch -format runinfo > runinfo.csv
```

The resulting `runinfo.csv` file is quite rich in metadata and contains a series of attributes for the run. You may open the file in Excel for an easier visualization though with time you’ll be just as comfortable viewing it at the command line:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Run	ReleaseDate	LoadDate	spots	bases	spots_with_avgLength	size_MB	Assembly	Nar download_p	Experiment	LibraryName	LibraryStrat	LibraryStrat_Lit
2	SRR1972917	4/14/15	4/14/15	4377867	884329134	4377867	202	486	http://sra-dosRX994194	G5723.1.l1	RNA-Seq	cD	
3	SRR1972918	4/14/15	4/14/15	3856384	778989568	3856384	202	457	http://sra-dosRX994195	G5731.1.l1	RNA-Seq	cD	
4	SRR1972919	4/14/15	4/14/15	4816440	972920880	4816440	202	541	http://sra-dosRX994196	G5732.1.l1	RNA-Seq	cD	
5	SRR1972920	4/14/15	4/14/15	320773	64796146	320773	202	38	http://sra-dosRX994197	G5735.2.l1	RNA-Seq	cD	
6	SRR1972921	4/14/15	4/14/15	5720830	1155607660	5720830	202	677	http://sra-dosRX994198	G5737.1.l1	RNA-Seq	cD	
7	SRR1972922	4/14/15	4/14/15	5244734	1059436268	5244734	202	605	http://sra-dosRX994199	G5738.1.l1	RNA-Seq	cD	
8	SRR1972923	4/14/15	4/14/15	2093369	422860538	2093369	202	240	http://sra-dosRX994200	G5739.2.l1	RNA-Seq	cD	
9	SRR1972924	4/14/15	4/14/15	3509670	708953340	3509670	202	395	http://sra-dosRX994201	G5740.1.l1	RNA-Seq	cD	
10	SRR1972925	4/14/15	4/14/15	5374781	1085705762	5374781	202	651	http://sra-dosRX994202	G5743.1.l1	RNA-Seq	cD	
11	SRR1972926	4/14/15	4/14/15	5131420	1036546840	5131420	202	534	http://sra-dosRX994203	G5748.1.l1	RNA-Seq	cD	
12	SRR1972927	4/14/15	4/14/15	6097111	1231616422	6097111	202	727	http://sra-dosRX994204	G5749.1.l1	RNA-Seq	cD	
13	SRR1972928	4/14/15	4/14/15	26185365	5289443730	26185365	202	3212	http://sra-dosRX994205	G5756.1.l1	RNA-Seq	cD	
14	SRR1972929	4/14/15	4/14/15	4478755	904708510	4478755	202	514	http://sra-dosRX994206	G5759.1.l1	RNA-Seq	cD	
15	SRR1972930	4/14/15	4/14/15	1812806	366186812	1812806	202	191	http://sra-dosRX994207	G5760.1.l1	RNA-Seq	cD	

Figure 55.1

Cutting out the first column of this file

```
cat runinfo.csv | cut -f 1 -d ',' | head
```

Produces:

55.1. HOW TO AUTOMATE DOWNLOADING MULTIPLE SRA RUNS?463

```
Run  
SRR1972917  
SRR1972918  
SRR1972919  
...
```

To filter for lines that match SRR and to store the first ten run ids in a file we could write:

```
cat runinfo.csv | cut -f 1 -d ',' | grep SRR | head > runids.txt
```

Our `runids.txt` file contains:

```
SRR1972917  
SRR1972918  
SRR1972919  
...
```

We now have access to the SRR run ids for the experiment. In the next step we wish to generate a series of commands of the form:

```
fastq-dump -X 10000 --split-files SRR1972917  
fastq-dump -X 10000 --split-files SRR1972918  
fastq-dump -X 10000 --split-files SRR1972919  
...
```

Note how the commands are mostly the same, we only substitute some parts of each. In the UNIX world, different techniques may be used to generate the commands above. Our choice for automation uses the `parallel` tool that executes commands in “parallel,” one for each CPU core that your system has. It is an extremely versatile command that has a large number of unique features, well suited to most tasks you will face. It is worth spending some time to understand how `parallel` works to fully appreciate the productivity gains you can make in your work.

- See: Tool: Gnu Parallel - Parallelize Serial Command Line Programs Without Changing Them¹

For example, here we run the `echo` command on each element of the file:

```
cat runids.txt | parallel "echo This is number {}"
```

will produce:

¹<https://www.biostars.org/p/63816/>

```
This is number SRR1972917
This is number SRR1972918
This is number SRR1972919
This is number SRR1972920
...
```

We can plug the `fastq-dump` command in the place of the `echo`:

```
cat runids.txt | parallel fastq-dump -X 10000 --split-files {}
```

The command above will generate twenty files, (two files for each of the ten runs).

55.2 How to filter for various metadata?

We also have options for filtering for various metadata in this file. We can, for example, filter for runs with the release date of August 19, 2014:

```
cat info.csv | grep '2014-08-19' | cut -f 1 -d ',' | grep SRR | head -10 > ids.txt
```

55.3 Is there even more metadata?

Yes there is. The runinfo format is only a subset of all the information SRA maintains on the run, sometimes does not include essential information that you might need. When that happens you will need to process the metadata downloaded in another format, called a document summary (docsum). What? Another format for the same information? Thanks, NCBI!

Let's get the docsum XML then:

```
esearch -db sra -query PRJNA257197 | efetch -format docsum > docsum.xml
```

The docsum data is an awkwardly overcomplicated format that will make you sweat for every piece of information that you want to wrangle out of it. View the docsum and brace yourself.

```
cat docsum.xml | more
```

55.4 How do I process the docsum format?

The strategy for processing docsum files is to transform the XML with the `xtract` command. If you knew what tags the information is stored under you could do a

```
cat docsum.xml | xtract -pattern DocumentSummary -element Bioproject,Biosample,Run@acc |
```

The command produces the output:

```
PRJNA257197 SAMN03253746      SRR1972976  
PRJNA257197 SAMN03253745      SRR1972975  
PRJNA257197 SAMN03253744      SRR1972974  
PRJNA257197 SAMN03254300      SRR1972973  
PRJNA257197 SAMN03254299      SRR1972972  
...
```

Finding the right tags is not an easy task, especially since no effort was made to make the docsum formatted in a humanly readable way. You can “prettyify” the format by setting up the following alias:

```
alias pretty="python -c 'import sys;import xml.dom.minidom;s=sys.stdin.read();print(xml.
```

using the

```
cat docsum.xml | pretty | more
```

allows you to investigate the XML file in a somewhat more manageable rearrangement.

Chapter 56

How much data in the SRA?

As you gain a better understanding of SRA is you may ask yourself, how much data is there? What technologies are in use, etc. We will go through a few of these questions below. Notably, we have run the same analyses at the end of 2016 as well as at the start of 2019 (roughly two years later); hence it will be interesting to see what changed within just two years.

While theoretically, you could get the runinfo on all SRA runs with a single `esearch` query, in reality, that approach will not work. The service will fail in various ways. It was just not designed for this use case. In general, when downloading large amounts of data, the best approach, whenever possible, is to divide the request into smaller queries. For example, we can limit the query to a single month with:

```
esearch -db sra -query '"2015/05" [Publication Date]' | efetch -format runinfo > 201505.runinfo
```

To use for the entire period, we will need to automate this line of code to iterate over each year and month pair. For that purpose. We wrote a script called `sra-runinfo.sh`¹ that downloads a single month's worth of data and ran it for every range. You don't need to run the script yourself though; you may just as well obtain the final data from the handbook data site. We include the command for educational reasons, as a potential starting point for other scripts of similar functionality:

```
# Generate the input sequences
seq 2007 2019 > years.txt
```

¹<http://data.biostarhandbook.com/sra/sra-runinfo.sh>

```

seq 1 12 > months.txt

# Get the script that downloads one months' worth of data.
wget -nc http://data.biostarhandbook.com/sra/sra-runinfo.sh

# Make the script executable
chmod +x sra-runinfo.sh

# Run on each pair, but just request in parallel at a time so the NCBI does not ban you.
parallel -j 1 --eta --progress --verbose ./sra-runinfo.sh {1} {2} :::: years.txt :::: months.txt

```

As mentioned before you can also directly download our results of running the script with:

```

# Obtain the runinfo data.
wget -nc http://data.biostarhandbook.com/sra/sra-runinfo-2019-01.tar.gz

# Unpack the runinfo data.
tar xzvf sra-runinfo-2019-01.tar.gz

# How big is the unpacked directory?
du -hs sra

```

It is a reasonably large file (512MB compressed, 2.7GB unpacked) that doubled in size since 2016!

We can now answer a series of interesting questions regarding how much data is in the SRA.

56.1 How many sequencing runs are in the SRA?

Combine all files into a single file; keep only lines that contain SRR numbers:

```
cat sra/*.csv | grep SRR > allruns.csv
```

How many runs are there:

```
cat allruns.csv | wc -l
```

Prints the number of sequencing runs stored in SRA, in 2016 it was almost 1.5 million, today it clocks in at almost 4.5 million sequencing runs:

```
4,373,757
```

In just two years the total number of sequencing runs tripled! Note that these are data only for the published sequencing runs deposited at NCBI!

56.2 How do we extract columns from a comma separated file?

Cutting columns on tabs is usually fine, as tabs are not part of a standard text. Commas, on the other hand, are commonly used in different contexts as well. When splitting directly on commas with a tool like `cut`, we may end up lining up the wrong columns. Hence, we need a more sophisticated approach that cuts the file correctly: at the “column delimiter” commas and not at the commas that are within a text field (quotes).

In other words, we need a tool that operates on `csv` files, for example:

- `csvkit`
- `miller`
- `csvtk`

and other similar tools.

In the following commands, we will replace `cut` with `csvcut` command that comes with the `csvkit` package. You can install `csvkit` with

```
pip install csvkit
```

we now run `datamash` to summarize the number of sequenced bases across all runs:

```
cat allruns.csv | csvcut -c 5 | datamash sum 1
```

in 2016 this number was `4.97e+15`, today is:

```
2.25e+16
```

Note how in two years the amount of data grew by order of magnitude! from 5 peta bases to 2.2 exa bases.

As you have seen before we often make use of the pattern:

```
sort | uniq -c | sort -rn
```

to count unique entries in a file. We can create a shortcut to this construct to run it more readily. In the Unix world a shortcut is called an alias:

```
alias tabulate='sort | uniq -c | sort -rn'
```

Now we can use the shortcut named `tabulate` to make our commands look a little simpler. The downside of shortcuts like this is that they make commands more abstract and a little less explicit, there is “hidden” information on what `tabulate` does. Other subtle “gotchas” may also trip you up, aliases for example won’t be visible in a subshell (a shell opened from a shell) etc. We use the aliases here as a demonstration.

56.3 How many sequencing runs per organism?

Column 29 of the run info file contains the organism:

```
cat allruns.csv | csvcut -c 29 | tabulate | head
```

in 2016 the list was:

```
480,622 Homo sapiens
141,135 Mus musculus
47,929 human metagenome
37,644 soil metagenome
36,740 human gut metagenome
26,440 Salmonella enterica
24,641 Drosophila melanogaster
21,396 Arabidopsis thaliana
19,577 Salmonella enterica subsp. enterica
19,370 Plasmodium falciparum
```

in 2019 the list is:

```
1,201,332 Homo sapiens
569,555 Mus musculus
129,382 soil metagenome
119,229 human gut metagenome
117,301 Salmonella enterica
```

```
102,580 human metagenome
 62,858 Escherichia coli
 52,308 gut metagenome
 51,124 Salmonella enterica subsp. enterica
 51,078 Drosophila melanogaster
```

I have added commas into the integers for readability, the command itself won't do that.

56.4 How many sequencing runs for each sequencing platform?

Column 19 of the run info file contains the sequencing platform name:

```
cat allruns.csv | csvcut -c 19 | tabulate | head
```

in 2016 the list was:

```
1,231,420 ILLUMINA
 182,407 LS454
 31,865 ABI_SOLID
 31,223 PACBIO_SMRT
 14,550 ION_TORRENT
 3,736 COMPLETE_GENOMICS
   869 HELICOS
   398 CAPILLARY
    96 OXFORD_NANOPORE
```

in 2019 the list is:

```
3,960,154 ILLUMINA
 248,304 LS454
 57,243 PACBIO_SMRT
 52,161 ION_TORRENT
 40,204 ABI_SOLID
 54,83 BGISEQ
 45,89 COMPLETE_GENOMICS
 2,493 CAPILLARY
 2,119 OXFORD_NANOPORE
 1,005 HELICOS
```

56.5 How many runs per sequencing instrument model?

Column 20 of the run info file contains the instrument model name:

```
cat allruns.csv | csvcut -c 20 | tabulate | head
```

in 2016 the list was:

```
633,701 Illumina HiSeq 2000
203,828 Illumina MiSeq
174,695 Illumina HiSeq 2500
120,715 454 GS FLX Titanium
99,268 Illumina Genome Analyzer II
47,066 Illumina Genome Analyzer IIx
33,187 454 GS FLX
26,527 NextSeq 500
25,243 Illumina Genome Analyzer
17,558 PacBio RS II
```

in 2019 the list is:

```
1,086,037 Illumina HiSeq 2000
1,013,347 Illumina MiSeq
915,073 Illumina HiSeq 2500
286,384 NextSeq 500
192,972 Illumina HiSeq 4000
158,024 454 GS FLX Titanium
142,459 HiSeq X Ten
108,684 Illumina Genome Analyzer II
61,321 Illumina Genome Analyzer IIx
44,219 454 GS FLX
```

Part XII

QUALITY CONTROL

Chapter 57

Visualizing sequencing data quality

The first step after receiving the sequencing data is to evaluate its quality. It is how you “get to know” your data.

57.1 What part of the FASTQ file gets visualized?

The FASTQ format contains sequenced bases and a quality value associated with each. For example, consider a record that looks like this:

```
@id  
ATGC  
+  
05:I
```

The characters 05:I, when decoded, corresponds to quality measures of 15 20 25 40. Basically it contains:

```
@id  
ATGC  
+  
10 20 25 40
```

Were each of the numbers corresponds to an estimate generated by the instrument as to how reliable the basecall is. The formula is to divide the number by 10 and raise 10 to that power,

$$20 \rightarrow 10^{(20/10)} \rightarrow 10^{-2} \rightarrow 1/100 \rightarrow 0.01 \rightarrow 1\%$$

thus a 20 means that the instrument thinks that 1% of time the basecall would be incorrect. Thus the FASTQ file contains the data:

```
@id
ATGC
+
10% 1% 0.3% 0.01%
```

when transformed into errors. Data quality visualizations generate various plots out of these numbers, for example, distribution of values by each base or by the average for each sequence.

Also, the composition of sequences may also be used to identify characteristics that could indicate errors.

57.2 Are FASTQ errors values accurate?

Not really. In our observation the numbers are quite unreliable - treat them as an advisory rather than accurate measurements.

57.3 What is the FastQC tool?

The undisputed champion of quality control visualization is a tool named FastQC¹ developed by Babraham Institute², an independent, charitable life sciences institute involved in biomedical research.

Even though it is a de-facto standard of visualization, its results are not always the simplest to interpret. On the positive side, the tool is easy to run (requires only Java), simple, reasonably efficient (though it is tuned for the Illumina platform and may be unstable on other types of data) and produces aesthetically pleasing plots. On the downside, some of its beautiful charts

¹<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

²<http://babraham.ac.uk/>

are uninformative and occasionally confusing. For example, in some cases, plots will switch from non-binned to binned columns within the same panel, easily misleading casual observers.

Then there are plots, such as the K-MER plot and the Overrepresented Sequences plot, that don't show what most people assume they do. There is also the difficulty of having to open HTML files one by one for result interpretation. In the current state of the field, where dozens of samples are being investigated in any given sequencing run, even just opening the results of the QC analysis becomes exceedingly tedious. The seemingly simple task to check the data quality requires clicking and scrolling through each file individually.

To some extent, even the tool's name is confusing. FASTQC does not perform quality control: it only visualizes the quality of the data.

But even with its flaws, FastQC³ is still by far the best FASTQ quality visualization tool. It is one of the tools that scientific funding agencies should seek to fund even if the author were not to apply for funding.

57.4 How does FastQC work?

FastQC generates its reports by evaluating a small subset of the data and extrapolating those findings to the entirety of the dataset. Many of the metrics are only computed on the first 200,000 measurements then are being tracked through the rest of the data.

The tool help³ contains detailed (although not entirely satisfactory) descriptions of what each data analysis plot is supposed to show.

57.5 How do we run FastQC?

Obtain and unpack the data for different sequencing platforms:

```
wget http://data.biostarhandbook.com/data/sequencing-platform-data.tar.gz
tar xzvf sequencing-platform-data.tar.gz
```

³<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/>

Run the FastQC tool:

```
fastqc illumina.fq
```

This action generates an HTML file called `illumina_fastqc.html` that contains the results of the run. There are options you can pass to `fastqc`:

```
fastqc illumina.fq --nogroup
```

that will change the plots in some way, the `--nogroup` option, for example, turns off the binning on columns.

There is data from several platforms in the file you downloaded above. As of 2016, one will fail to run with `fastqc`

```
fastqc minion.fq
```

Will raise a memory error. This is just an example of how some tools are too closely tuned for one specific sequencing instrument. Even our quality control tool can fail on a ‘standard’ dataset that happens to be from a different platform.

57.6 Should I be worried about the “stop-light” symbols?

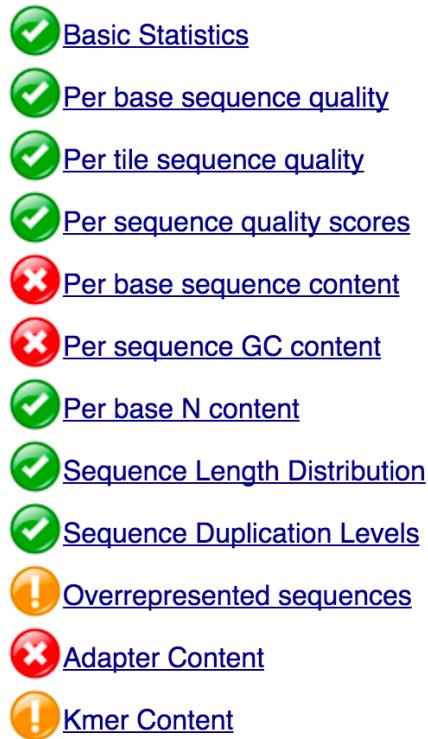
Usually not.

When FastQC runs, it generates “stoplight” icons for each analysis having “pass,” “warning,” and “error” symbols. Most of the time, these symbols are not meaningful. They were developed for only a particular class of samples and library preparation methods and just for certain types of instruments.

Though it is fair to say if most or all your stop light icons are red, then you probably have a data problem.

57.7 What does the sequence quality visualization tell us?

The simplest way to snapshot the quality of a sequencing run is via a chart that plots the error likelihood at each position averaged over all measure-

**Figure 57.1**

ments.

The vertical axis are the FASTQ scores that represent error probabilities:

- 10 corresponds to 10% error (1/10),
- 20 corresponds to 1% error (1/100),
- 30 corresponds to 0.1% error (1/1,000) and
- 40 corresponds to one error every 10,000 measurements (1/10,000) that is an error rate of 0.01%.

The three-colored bands illustrate the typical labels that we assign to these measures: reliable (30-40, green), less reliable (20-30, yellow) and error prone (1-20, red). The yellow boxes contain 50% of the data, the whiskers indicate the 75% outliers.

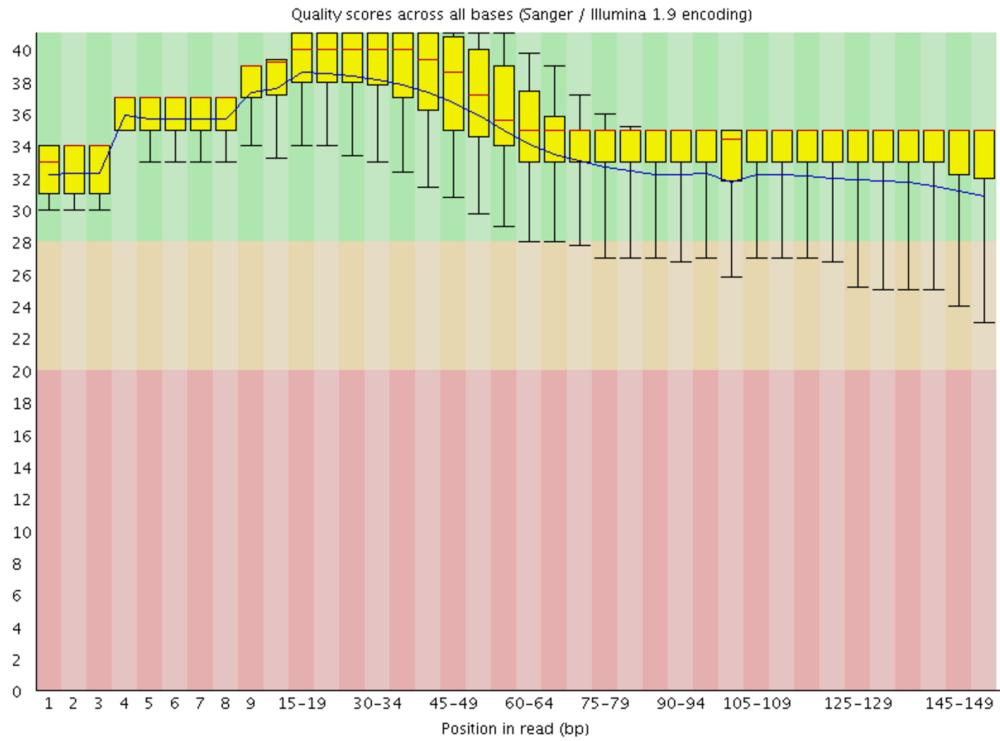


Figure 57.2

57.8 What does the sequence length histogram show?

The sequence length distribution shows how many sequences of each length the data contains. For fixed read length instruments, like the Illumina sequencer, all read lengths are the same. For long read technologies like the PacBio and MinION, the distribution can be a lot more varied.



Figure 57.3

57.9 What does the sequence quality histogram tell us?

Another way to visualize data quality is to generate the histograms of the average qualities. The horizontal scales are the quality scores; the vertical axis indicates the number of reads of that quality.

57.10 What is the next step after visualizing the quality?

After visualization, you need to decide whether the data should be corrected. This process is called Quality Control (QC) and typically follows this process:

57.10. WHAT IS THE NEXT STEP AFTER VISUALIZING THE QUALITY?481

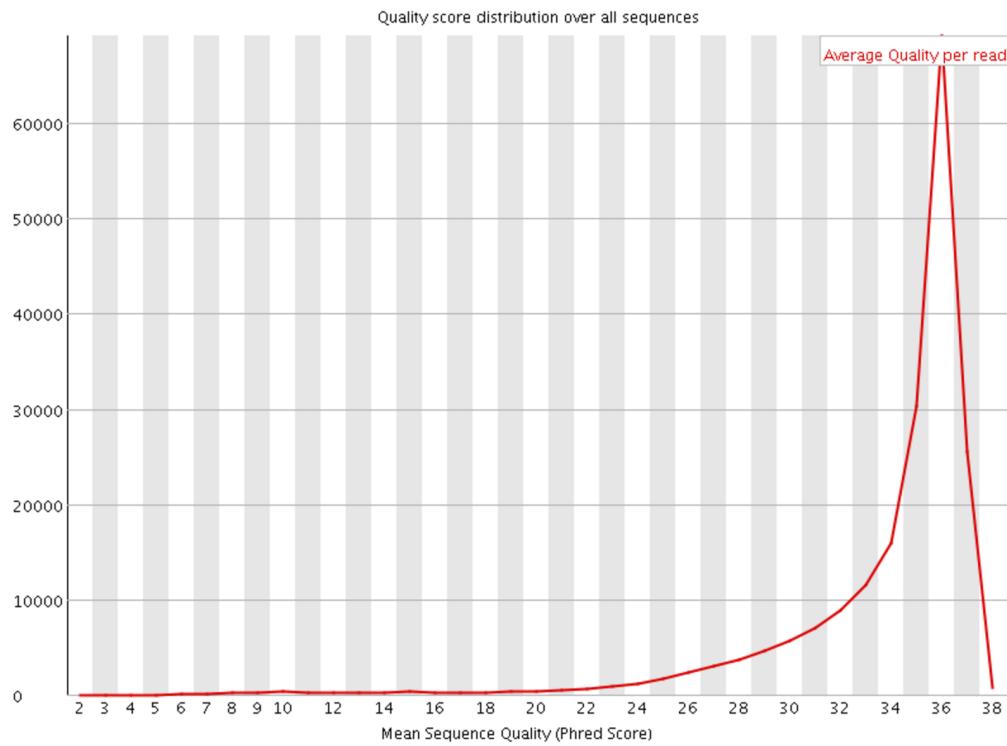


Figure 57.4

1. Evaluate (visualize) data quality.
2. Stop QC if the quality appears to be satisfactory.
3. If the quality is still inadequate, execute one or more data altering steps then go to step 1.

There are many tools that you can use to alter the data. We will cover a few in the subsequent chapters. Each tool has benefits but also may require various tradeoffs.

Chapter 58

Sequencing quality control

58.1 What is quality control?

Quality control (abbreviated as QC from now on) is the process of improving data by removing identifiable errors from it. It is typically the first step performed after data acquisition.

Since it is a process that alters the data, we must be **extremely cautious** not to introduce new features into it inadvertently. Ideally, we want the same data (information) only better (more accurate). But please do note:

It is common to expect too much from quality control. Remember that QC cannot turn bad data into useful data and we can never salvage what appears to be a total loss.

When the sequencing data looks terrible from the beginning, it is best to move on and collect new data. No amount of QC will save it. Only wasted work and frustrations lie that way.

58.2 How critical is quality control?

The more unknowns about the genome under study, the more critical it is to correct any errors.

When aligning against well-studied and known genomes, we can recognize and identify errors by their alignments. When assembling a de-novo genome, mistakes can derail the process; hence, it is more important to apply a higher stringency for filtering.

58.3 What can go wrong with the data?

Probably the greatest challenge of QC is our inability to foresee all the possible ways that data can be affected. Good data are all alike, but once things go awry, the potential problems that may need correcting can run a whole gamut: from puzzling to fascinating to outright disturbing.

For some people, performing quality control is also somewhat of a psychological barrier. Many feel that they should only move to the next step after they have done everything possible to improve the data. They may run, rerun, alter the data, and spend weeks and months on it. Don't be one of these people:

Data improvement via quality control is an incremental process with ever-diminishing returns. Once simple corrections have been made, no amount of tweaking will produce radically better outcomes.

Besides, a common pitfall is to keep tweaking QC in a manner that seems to “improve” the final results; the danger with this is overfitting¹ - making the data match the desired outcome.

58.4 When do we perform quality control?

Quality control may be performed at different stages

- Pre-alignment: “raw data” - the protocols are the same regardless of what analysis will follow
- Post-alignment: “data filtering” - the protocols are specific to the analysis that is performed.

¹<https://en.wikipedia.org/wiki/Overfitting>

58.5 How do we perform quality control?

QC typically follows this process.

1. Evaluate (visualize) data quality.
2. Stop and move to the next analysis step if the quality appears to be satisfactory.
3. If not, execute one or more data altering steps then go to step 1.

58.6 How reliable are QC tools?

First, let's me start with an observation that might blow your mind. Are you ready? Here it comes. Most quality control tools are of very low software quality themselves.

There was an era (one that may not even be over yet) when writing a high-throughput sequencing quality control tool seemed to be “in fashion” because such tools were considered “easy” to publish. A literature search may turn up dozens of such software publications. Unfortunately, in our experience, a majority of these tools are naïve solutions that perform surprisingly poorly in real-world scenarios.

It also doesn't help that there is a surprising amount of inconsistency in the way QC tools work. Even though the concepts that the tools implement appear to be well defined, we have never been able to produce the same outcomes when employing different tools with seemingly identical algorithms.

Finally, the truth of the matter is that an objective assessment of the data quality is inherently tricky. What you will often see and hear are subjective, almost hand-waving assessments of what good data “looks like”. Beyond that, there are few objective measures, standards, and recommendations that we can test against.

58.7 Can quality control introduce new problems?

This might be the second mind-blowing observation we make in this same chapter - one that just about every bioinformatician seems to conveniently forget.

Remember - every quality control process introduce errors. We dearly hope that the impact of the new errors that the QC creates to be much smaller than the impact of errors that the QC corrects. But see how at some point it is almost a philosophical issue - would you rather be dealing with errors that an instrument caused or mistakes that you introduced when correcting for the machine errors?

Do not error correct data that does not need it.

58.8 Is there a list of QC tools?

A considerable number of QC tools have been published. We have only tested a few of these and out of those we recommend **Trimmomatic**, **BBDuk**, **flexbar** and **cutadapt**. While all tools implement basic quality control methods, each often includes unique functionality specific to that tool.

Quality control tools often provide more utility, some are full sequence manipulation suites. Here is a list (in alphabetical order) of tools that we have had some experiences with:

- BBDuk² part of the BBMap³ package
- BioPieces⁴ a suite of programs for sequence preprocessing
- CutAdapt⁵ application note in Embnet Journal, 2011⁶
- fastq-mcf⁷ published in The Open Bioinformatics Journal, 2013⁸

²<https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/>

³<https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/>

⁴<https://code.google.com/p/biopieces/>

⁵<http://code.google.com/p/cutadapt/>

⁶<http://journal.embnet.org/index.php/embnetjournal/article/view/200>

⁷<http://code.google.com/p/ea-utils/>

⁸<http://benthamscience.com/open/openaccess.php?tobioij/articles/V007/>

- Fastx Toolkit: collection of command line tools for Short-Reads FASTA/FASTQ files preprocessing⁹ - one of the first tools
- FlexBar, Flexible barcode and adapter removal¹⁰ published in Biology, 2012¹¹
- NGS Toolkit¹² published in Plos One, 2012¹³
- PrinSeq¹⁴ application note in Bioinformatics, 2011¹⁵
- Scythe¹⁶ a bayesian adaptor trimmer
- SeqPrep¹⁷ - a tool for stripping adaptors and/or merging paired reads with overlap into single reads.
- Skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads.¹⁸
- TagCleaner¹⁹ published in BMC Bioinformatics, 2010²⁰
- TagDust²¹ published in Bioinformatics, 2009²²
- Trim Galore²³ - a wrapper tool around Cutadapt and FastQC to consistently apply quality and adapter trimming to FastQ files, with some extra functionality for MspI-digested RRBS-type (Reduced Representation Bisulfite-Seq) libraries
- Trimmomatic²⁴ application note in Nucleic Acid Research, 2012, web server issue²⁵

There are also many libraries via R (Bioconductor) for QC such as: PIQA²⁶, ShortRead²⁷

⁹http://hannonlab.cshl.edu/fastx_toolkit/

¹⁰<http://sourceforge.net/projects/flexbar/>

¹¹<http://www.mdpi.com/2079-7737/1/3/895>

¹²<http://www.nipgr.res.in/ngsqctoolkit.html>

¹³<http://www.plosone.org/article/info:doi/10.1371/journal.pone.0030619>

¹⁴<http://prinseq.sourceforge.net/>

¹⁵<http://www.ncbi.nlm.nih.gov/pubmed/21278185>

¹⁶<https://github.com/vsbuffalo/scythe>

¹⁷<https://github.com/jstjohn/SeqPrep>

¹⁸<https://www.biostars.org/p/104320/>

¹⁹<http://tagcleaner.sourceforge.net/>

²⁰<http://www.biomedcentral.com/1471-2105/11/341>

²¹<http://genome.gsc.riken.jp/osc/english/software/>

²²<http://www.ncbi.nlm.nih.gov/pubmed/19737799>

²³http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/

²⁴<http://www.usadellab.org/cms/index.php?page=trimmmatic>

²⁵<http://nar.oxfordjournals.org/content/40/W1.toc>

²⁶<http://www.bioinfo.uh.edu/PIQA/>

²⁷<http://www.bioconductor.org/packages/2.12/bioc/html/ShortRead.html>

58.9 How does read quality trimming work?

Historically, for early instruments, the reliability of sequencing decreased sharply along the read. A typical correction was to work backward from the end of each read and remove low-quality measurements from it. The read shortening process is called “trimming”.

```
# Get the data
fastq-dump --split-files -X 10000 SRR1553607

# Run trimmomatic in single end mode
trimmomatic SE SRR1553607_2.fastq trimmed_2.fq SLIDINGWINDOW:4:30

# Generate fastqc reports on both datasets.
fastqc SRR1553607_1.fastq trimmed_2.fq
```

The resulting plot shows the original file on the left and the quality-trimmed version on the right.

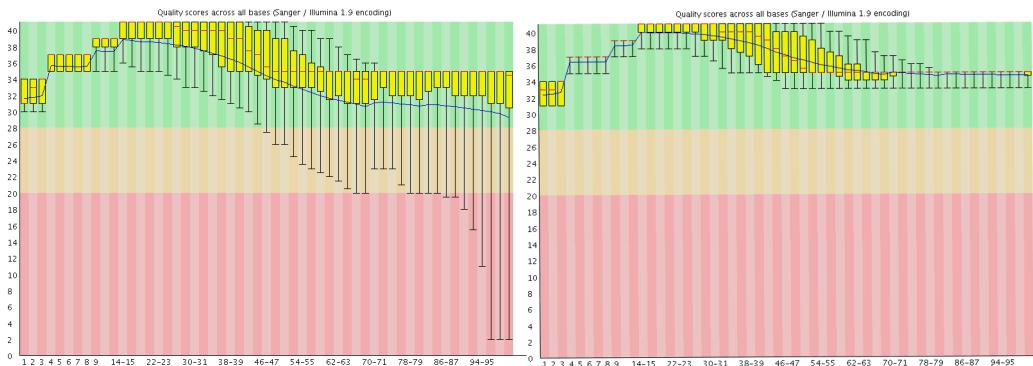


Figure 58.1

A similar (but not identical) result would be found using `bbduk` to trim adapters:

```
bbduk.sh in=SRR1553607_2.fastq out=bbduk.fq qtrim=r overwrite=true qtrim=30
```

To perform the same process on both files at the same time, we need a more complicated command.

```
trimmomatic PE SRR1553607_1.fastq SRR1553607_2.fastq trimmed_1.fq unpaired_1.fq trimmed_2.fq unpaired_2.fq
```

Or with BBduk:

```
bbduk.sh in1=SRR1553607_1.fastq in2=SRR1553607_2.fastq outm1=bbduk_1.fq out1=un
```

Chapter 59

Sequencing adapter trimming

59.1 What are sequencing adapters?

During library preparation, uniquely designed DNA adapters of lengths that are typically over 30 bases are added to the 5' and 3' ends of each sequence.

Hence, after adding the adapters, the single-stranded sequences that make it into the instrument will be of the form:

XXXXAAAATTTGGGGCCCCYYYY

where XXXX and YYYY are the sequencing adapters. The instrument can recognize the starting adapters but usually won't detect the end adapters if these make it into the measurement. When the read length exceeds the length of the DNA fragment, the adapter sequence may appear in the data.

59.2 How do I visualize the sequencing adapters?

FastQC will detect adapters based on Illumina sequences and will display them in the following manner:

The rising line indicates the number of adapter sequences contained at that base.

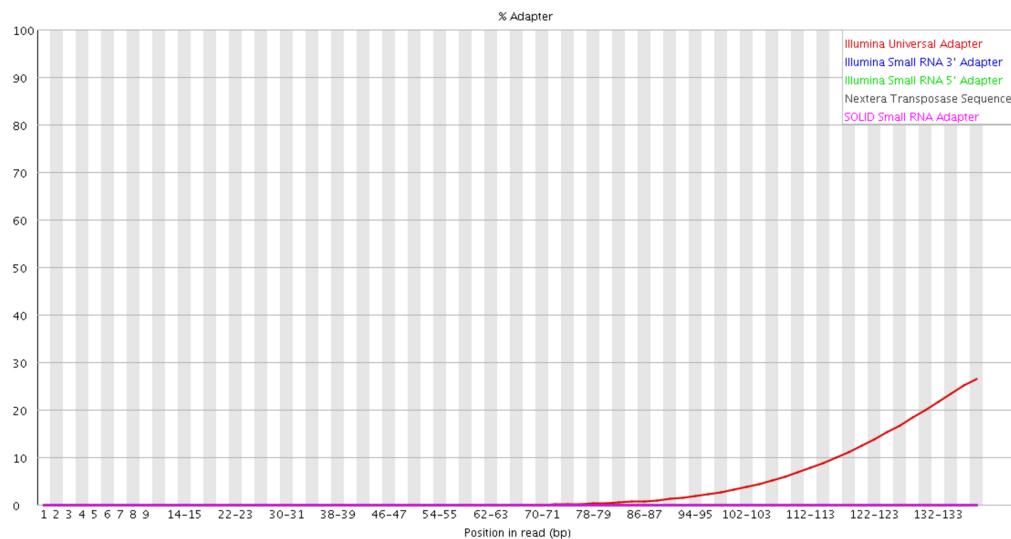


Figure 59.1

59.3 Can we customize the adapter detection?

Yes. FASTQC has files that specify the adapters to search for.

The location of the adapter file depends on the installation (and version number) but would be somewhere like `~/miniconda3/envs/bioinfo/opt/fastqc-0.11.8/Config`

The default content of the adapter file is:

Illumina Universal Adapter	AGATCGGAAGAG
Illumina Small RNA 3' Adapter	TGGAATTCTCGG
Illumina Small RNA 5' Adapter	GATCGTCGGACT
Nextera Transposase Sequence	CTGTCTCTTATA
SOLID Small RNA Adapter	CGCCTTGGCCGT

Adding your adapters into that list will allow you to have them displayed in the FASTQC plot.

59.4 Why do we need to trim adapters?

As you recall, within the sequencing instrument, each read has artificial sequences attached to its end. If the sequence is AAAAAAAA and the adapter is TTTTTT then depending on the library size and read length the sequencing may run into the adapter.

```
AAAAAAAATTTTT
---->
----->
----->
```

The last read will have the adapter sequence TTTTT at the end.

If the overlap of bases into the adapter is sufficiently long (5-6), we can detect the adapter itself and cut the sequence at the start of the adapter.

To recognize the adapters, we need to know what their sequence is. This information is surprisingly difficult to find out from those that put these adapters on. Most companies consider these as “trade secrets” and you would be hard pressed to find clear documentation of it. When found, it is usually buried deep in other information and needs to be fished out from it. For example, the Illumina TruSeq index adapters will have the following sequence:

```
# TruSeq Indexed Adapter
GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATCTGTATGCCGTCTCTGCTTG
```

To complicate matters even more, an additional A is ligated to the end of the sequences before the adapter is attached. Hence, the artificial sequence present at the end of some reads will typically be some fraction of the sequence:

```
AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC
```

To see other adapters that for example the FastQC tool detects see:

```
# See configuration files (your version number may be different)
ls -l ~/miniconda3/envs/bioinfo/opt/fastqc-0.11.5/Configuration/
```

```
# The adapter configuration file.
cat ~/miniconda3/envs/bioinfo/opt/fastqc-0.11.5/Configuration/adapter_list.txt
```

A full list of adapters may be found on the data webpage at:

- <http://data.biostarhandbook.com/data/sequencing-adapters.fa>

59.5 How do we trim adapters?

You can create your adapter or use the ones that come with Trimmomatic. Let's create an adapter format for Illumina instrument

```
echo ">illumina" > adapter.fa
echo "AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC" >> adapter.fa
```

Use this to filter another SRA file:

```
fastq-dump -X 10000 --split-files SRR519926
fastqc SRR519926_1.fastq
```

Trim adapters with `trimmmatic`:

```
trimmmatic SE SRR519926_1.fastq output.fq ILLUMINACLIP:adapter.fa:2:30:5
```

The left plot shows the adapter content of the data before removing the adapters. The right-hand side shows the effect of the command above.

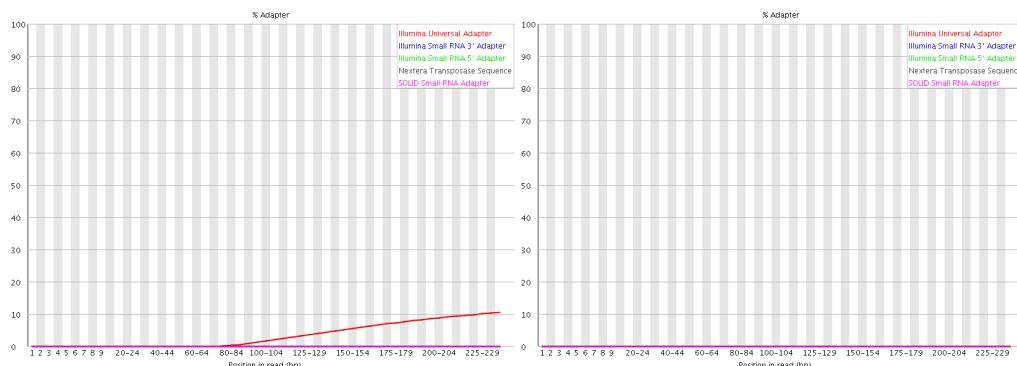


Figure 59.2

With `bbduk`, the trimming would be:

```
bbduk.sh in=SRR519926_1.fastq ref=adapter.fa out=bbduk.fq
```

59.6 How do we perform multiple steps at once?

Most tools allow us to chain together multiple actions, though usually the resulting command line is quite an eyeful.

```
trimmmatic PE SRR519926_1.fastq SRR519926_2.fastq trimmed_1.fq unpaired_1.fq trimmed_2
```

It is important to recognize that the order of action matters and results may be different depending on which action is performed first:

1. Do we first trim for quality then cut the adapters?
2. Do we cut the adapters first and then trim for quality?

Since the adapters have to overlap if we trim by quality first, we may run the risk of removing a base that will not allow us to detect the adapter. On the other hand, if the base is of low quality, it may not align to the adapter; hence, the adapter won't be detected. Often the order of operation does not make a notable difference, but on problematic data it's best to experiment.

59.7 Do we have to remove adapters from data?

Adapter read-through problems may not need to be dealt with explicitly if the genomes that the data originated from are known.

Many high-throughput aligners can automatically clip the alignments and trim off adapters during the alignment phase.

On the other hand, the presence of adapter sequences may cause substantial problems when assembling new genomes or transcriptomes; hence they should be removed before starting that process.

There is no full consensus on the matter, although Brian Bushnell makes a strong case that adapter trimming dramatically improves the quality of data.

Read on the Biostar question of the day Trimming adapter sequences - is it necessary?¹

¹<https://www.biostars.org/p/212136/>

59.8 How do I cut a different adapter?

Depending on the library preparation and experiment design other artificial sequences may be attached at the end of a DNA fragment:

```
fastq-dump -X 10000 SRR1553606 --split-files
```

The FastQC report will show Nextera adapter contamination:

```
fastqc SRR1553606-1.fastq
```

Verify what FastQC can detect to begin with:

```
cat ~/miniconda3/envs/bioinfo/opt/fastqc-0.11.5/Configuration/adapter_list.txt
```

Create the adapter that you need to cut (this may need searching for information) for convenience here is the Nextera adapter:

```
echo ">nextera" > nextera.fa
echo "CTGTCTTATACACATCTCCGAGCCCACGAGAC" >> nextera.fa
```

Trim the sequences with this new adapter.

Chapter 60

Sequence duplication

60.1 What is sequence duplication?

Duplication usually means having the identical sequences in the data. Depending on the expected coverages duplicates may be correct measurements or errors.

Right off the bat the word “duplicate” is misleading; it seems to imply that there is “two” of something, what it really means is “more than one”. No wonder there is a surprising amount of confusion of what duplication is and most importantly what the appropriate action is when you observe it.

Duplicates fall into two classes:

1. Natural duplicates - these were identical fragments present in the sample. These should not be removed.
2. Artificial copies - these are produced artificially during the sequencing process, PCR amplification, detection errors. These should be removed.

But how can we distinguish between the two cases? As it turns out the process is fraught with challenges.

60.2 How do we detect sequence duplication?

There are two main approaches:

1. Sequence identity - where we find and remove sequences that have the exact sequence.
2. Alignment identity - where we find and remove sequences that align the same way.

As counterintuitive as that might sound, by filtering on sequence identity, we are running the risk of rewarding reads with errors in them. We are removing identical reads, yet keep those reads that were supposed to be equal but were mismeasured. There is a danger of actually enriching our data with incorrect measurements.

On the other hand, alignment-based identity is only feasible if there is a known genome to align the data against.

There is a new class of methods that detect subsequence composition (called k-mers) that can be used to correct errors.

60.3 What is the primary concern with duplicates?

The adverse effect of read duplication manifests itself mostly during variation detection. This process assigns a reliability score to each variant based on the number of times it has been observed. If a single read with a variant happens to be duplicated artificially, it runs the risk of producing a seemingly more reliable result than in reality.

60.4 Should we remove duplicates?

For SNP calling and genome variation detection, the answer is usually yes. For other processes, the answer is often no.

Duplicate removal is a substantial alteration of the data - one should have additional evidence that none of the duplicates are of natural origin before doing it.

As always the severity of the problem and whether it impacts the study should be evaluated and weighted.

60.5 What does the FastQC duplicate plot mean?

It is not all that simple to interpret the FastQC duplication report. We have struggled with it.

First of all, that small number circled at the top is one of the most important values. It shows what percent of the data is distinct.

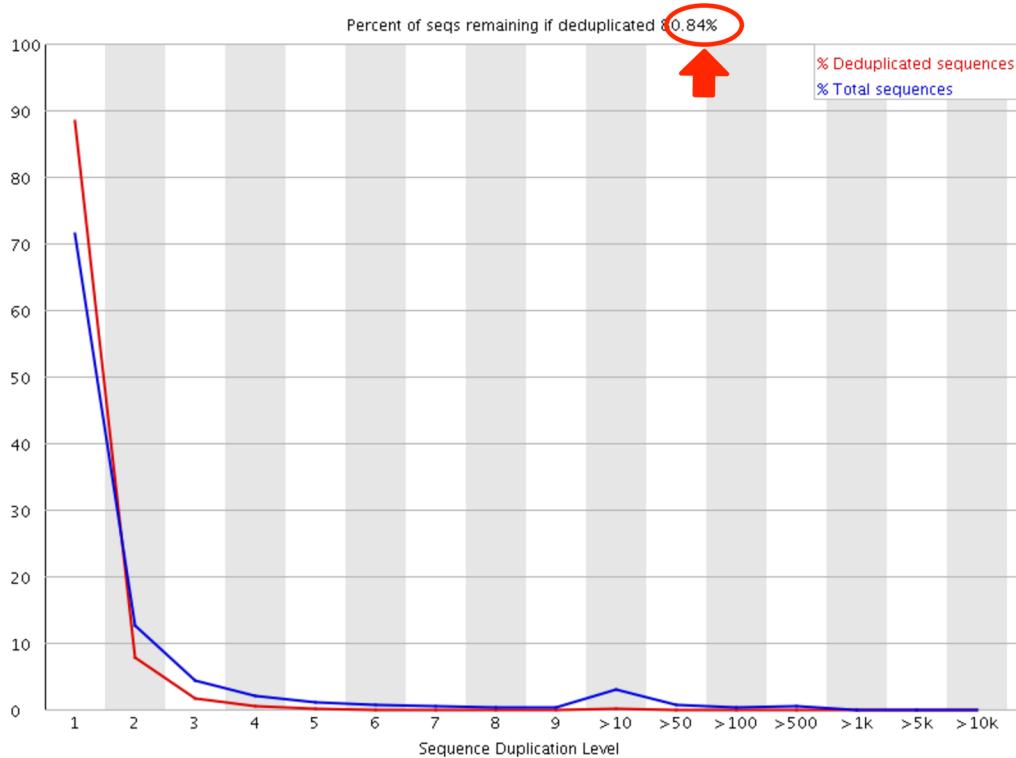


Figure 60.1

- Biostar question of the day: Revisiting the FastQC read duplication report¹

There are two kinds of duplicate counts: one for all sequences, and another for distinct sequences.

¹<https://www.biostars.org/p/107402/>

As an example, take a situation where two sets would have to be built as

A A A A B B B B

and

A A A A A A A B

Both examples have only two distinct sequences: A and B and the duplication rate would be the same. We have two distinct options and eight observations: 2/8, But obviously, the sets' internal structure is entirely different. We should not use the same measure to quantify the duplication of the first set as the second.

This set structure is what the blue and red lines try to capture and visualize for us.

- The blue lines show the percent (Y-axis) of **all sequences** that are duplicated at a given rate (X-axis).
- The red line indicates the number of **distinct sequences** that are replicated at a given frequency.

To work this out explicitly. In the first example, we have two sequences each duplicated at a rate of 4, whereas in the second case we have one unique sequence and one sequence reproduced at the rate of 7.

60.6 How do I remove duplicates?

In general, avoid removing duplicated sequences by sequence identify alone unless you have good reasons to do so. Once you have an alignment file (BAM that we will cover later), you may use the picard MarkDuplicates² tool to mark (label) duplicated alignments.

²<https://broadinstitute.github.io/picard/command-line-overview.html#MarkDuplicates>

Chapter 61

Advanced quality control

61.1 Are there different types of quality control?

Yes. Up to this point, we only covered the QC of the so-called “raw” (original) data that is produced directly by the instrument.

But quality control may be performed later in the process as well. Often though at that point it typically becomes a different type of tasks, more of a quality assessment methods to answer questions of “How well does the data follow expected characteristics?”. For example, does the RNA-Seq data align to transcripts as we expect it to?

RNA-Seq analysis for that matter has two (quite unfortunately named) software packages, which you will, like us, permanently confuse and conflate:

- RSeQC¹
- RNA-SeqC²

Author’s note: By the way, if you ever end up writing a QC software tool, please do us all a service and find a memorable and simple name for it. Call it `speedyQC`, call it `monsterQC` call it `sevenQC`, but please, please don’t make

¹<http://rseqc.sourceforge.net/>

²<http://archive.broadinstitute.org/cancer/cga/rna-seqc>

it an awkward variation of an existing, already awkward name. While we are on this topic, here are more suggestions on to avoid:

- Crac: Funny And Weird Names For Bioinformatics Tools³

61.2 How to combine the results into a single report.

An inconvenience of `fastqc` (and many other tools) is that each report goes into a separate, standalone file. For more extensive studies that may involve dozens or even hundreds of samples, consulting each of these reports becomes tedious to impossible.

The MultiQC⁴ tool is designed to facilitate combining different reports into a single one.

Aggregate results from bioinformatics analyses across many samples into a single report

MultiQC searches a given directory for analysis logs and compiles an HTML report. It's a general use tool, perfect for summarising the output from numerous bioinformatics tools.

At the time of writing this document `multiqc` supported the integration of reports generated by 51 different tools. The documentation of MultiQC⁵ reads almost like a table of contents describing the tools and techniques you could use to assess the quality of your data. Our original installation did not include this package since it has many requirements thus you will have to install `multiqc` yourself. Thankfully there is a `conda` package for it.

```
conda activate bioinfo  
pip install multiqc
```

Get and unpack the data:

³<https://www.biostars.org/p/67874/>

⁴<http://multiqc.info/>

⁵<http://multiqc.info/>

61.2. HOW TO COMBINE THE RESULTS INTO A SINGLE REPORT.501

```
wget http://data.biostarhandbook.com/data/sequencing-platform-data.tar.gz  
tar zxvf sequencing-platform-data.tar.gz
```

Run fastqc on two datasets to compare the Illumina platform to that of IonTorrent's. The PacBio data is so radically different that is probably not worth putting on the same plot. The command below will create two `fastqc` reports and the `--extract` flag instructs `fastqc` to keep the data directories for each report:

```
fastqc --extract illumina.fq iontorrent.fq
```

Now `multiqc` can combine the data from the three data report directories into one:

```
multiqc illumina_fastqc iontorrent_fastqc
```

The resulting plot shows us a clear comparison of the two runs:

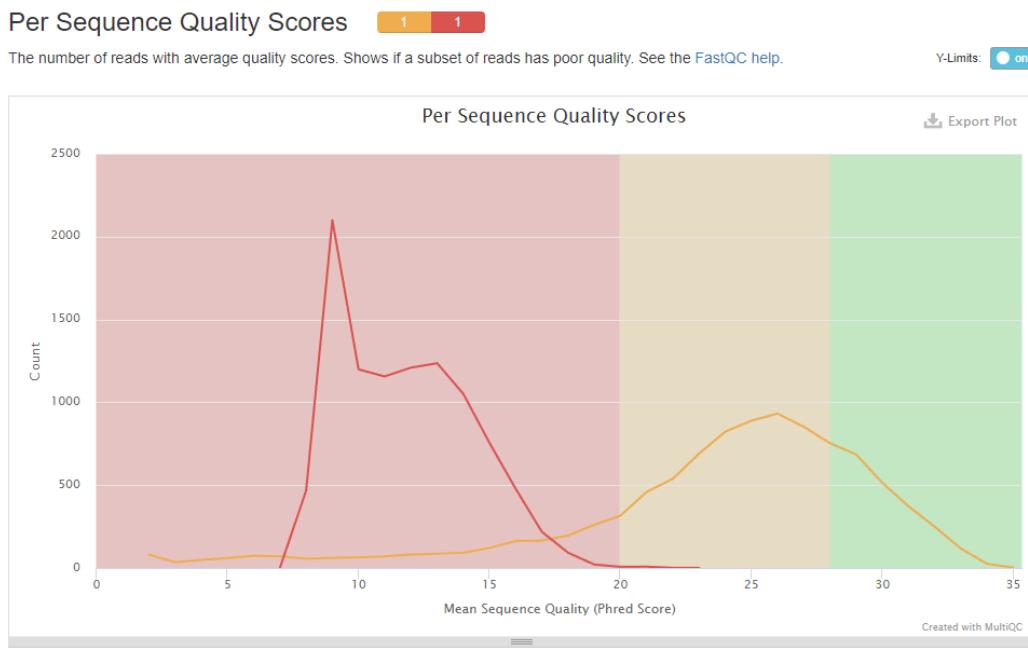


Figure 61.1

As you can imagine when you have many sequencing runs to summarize `multiqc` is quite the handy tool.

61.3 Why did my `multiqc` fail with an error?

Here is a tale of caution when it comes to bioinformatics tools.

The very next day after this guide was published readers started reporting that `multiqc` did not work at all. Even just by invoking the tool `multiqc` they got errors such as:

```
TypeError: add_edge() takes 3 positional arguments but 4 were given
```

What happened is that a new version of the `networkx` software was released, one that changed the way the `add_edge` method call works. Now `multiqc` does not use `networkx` directly. But it uses a library called `spectra` that in turn uses a library called `colormath` that in turn uses `networkx`. So there is a “network” (pun intended) of dependencies in play.

Then there is a flaw in the way the `colormath` library is installed. It does not “pin” the library to a version of `networkx`. The standard and proper software engineering practice is to specify not just a dependency but also the version of it that is proven to work correctly. Now in the meantime the `colormath` library also added code to support both versions of `networkx` the library. Alas the `colormath` library installed with `conda` is a prior version and not the latest that includes the fixes.

Hence the two compounding problems lead to a complete failure to run the program. Eventually, once sufficient number of people complain the problem will be fixed. By the time you read this the problem might not exist. But it exhibits an acute problem that will re-occur under different circumstances. We, scientists, usually don’t recognize the importance of proper software maintenance practices.

In the meantime if you experience this problem you may downgrade to a previous version of `networkx` with:

```
source activate multiqc
conda install networkx=1.11
```

61.4 What else is *multiqc* capable of?

If you need to process more than a few (2-3 datasets) we recommend that you consult the *multqc* documentation⁶ and look for ways it can help you interpret the results.

61.5 Would we ever want to toss out some of the good quality data?

Surprising as it sounds too much data can make processes operate worse. A saturation of information can take place, after which the errors get magnified.

In those cases, you could, perhaps, choose to select a random subset of the entire data and that may work well if your data happens to cover the DNA evenly. However, the typical situation can be more complicated, where some regions may be extremely highly covered, while others still be under shallow coverage. For some studies, such as genome assembly, you would want to remove data from the high coverage regions while keeping the low coverages intact.

So you'd have to apply some downsampling but without knowing beforehand where each measurement belongs. This seemingly impossible task can be accomplished by an algorithm that keeps track of the frequency of small patterns in each sequencing read. In a nutshell, reads built from short patterns that have already been observed many times over can be ignored.

Some tools internally implement such techniques, for example, the Trinity transcriptome assembler will downsample the data. In other cases, external tools such as khmer⁷ may be used for this purpose.

⁶<http://multiqc.info/>

⁷<http://khmer.readthedocs.io>

61.6 Can I combine reads into a single sequence?

Under certain circumstances, for paired end sequencing, when the DNA fragment is shorter than the sum of the read lengths the reads may overlap:

```
-----read1----->
===== DNA =====
<----read2-----
```

The two opposing reads that have an overlapping region may be combined using the common regions:

```
-----read1----->
||||||| | | | |
<----read2-----
```

To produce a single, longer and more accurate read:

```
----->
```

This process is called “merging.”

61.7 When should I merge the reads?

Combining reads only makes sense when the majority of the reads overlap. Otherwise, you’ll end up with a mixed data of overlapped single-end reads, and non-overlapping paired-end reads - only adding to the complexity of the subsequent tasks.

61.8 How do I merge reads?

In Biostar Post of the Day: Tools to merge overlapping paired-end reads⁸, Rafay Khan writes:

⁸<https://www.biostars.org/p/225683/>

In very simple terms, current sequencing technology begins by breaking up long pieces of DNA into lots more short pieces of DNA. The resultant set of DNA is called a “library” and the short pieces are called “fragments”. Each of the fragments in the library are then sequenced individually and in parallel. There are two ways of sequencing a fragment - either just from one end, or from both ends of a fragment. If only one end is sequenced, you get a single read. If your technology can sequence both ends, you get a “pair” of reads for each fragment. These “paired-end” reads are standard practice on Illumina instruments like the GAIIX, HiSeq, and MiSeq.

Now, for single-end reads, you need to make sure your read length (L) is shorter than your fragment length (F) or otherwise the sequence will run out of DNA to read! Typical Illumina fragment libraries would use $F \sim 450\text{bp}$, but this is variable. For paired-end reads, you want to make sure that F is long enough to fit two reads. This means you need F to be at least $2L$. As $L=100$ or 150bp these days for most people, using $F \sim 450\text{bp}$ is fine, there is still a safety margin in the middle.

However, some things have changed in the Illumina ecosystem this year. Firstly, read lengths are now moving to $>150\text{bp}$ on the HiSeq (and have already been on the GAIIX) and to $>250\text{bp}$ on the MiSeq, with possibilities of longer ones coming soon! This means that the standard library size $F \sim 450\text{bp}$ has become too small, and paired-end reads will overlap. Secondly, the new enzymatic Nextera library preparation system produces a wide spread of F sizes compared to the previous TruSeq system. With Nextera, we see F ranging from 100bp to 900bp in the same library. So some reads will overlap, and others won’t. It’s starting to get messy.

The whole point of paired-end reads is to get the benefit of longer reads without actually being able to sequence reads that long. A paired-end read (two reads of length L) from a fragment of length F , is a bit like a single-read of length F , except a bunch of bases in the middle of it are unknown, and how many of them there are

is only roughly known (as libraries are only nominally of length F, each read will vary). This gives the reads a more extended context, and this mainly helps in de novo assembly and in aligning more reads unambiguously to a reference genome. However, many software tools will get confused if you give them overlapping pairs, and if we could overlap them and turn them into longer single-end reads, many tools will produce better results, and faster.

Multiple tools are listed in the post above. One of the most popular choices FLASH (Fast Length Adjustment of SHort reads)⁹ is a tool to merge paired-end reads that were generated from DNA fragments whose lengths are shorter than twice the length of reads. Merged read pairs result in unpaired longer reads. Longer reads are more desired in genome assembly and genome analysis processes.

```
conda activate bioinfo
conda install flash -y
```

Get the 1% error test data from the tool website (there is 1%, 2%, 3%, and 5% error test data as well, give those a go!):

```
# Get an example dataset.
wget https://ccb.jhu.edu/software/FLASH/error1.tar.gz
```

```
# Unpack the data
tar zxvf error1.tar.gz
```

This dataset now contains `frag1.fq` and `frag2.fq`

```
flash frag_1.fastq frag_2.fastq
```

The tools create some ancillary files, while reporting:

```
[FLASH] Read combination statistics:
[FLASH]      Total pairs:      1000000
[FLASH]      Combined pairs:  701191
[FLASH]      Uncombined pairs: 298809
[FLASH]      Percent combined: 70.12%
```

Note how at 1% error rate only 70% of the data could be merged.

⁹<http://ccb.jhu.edu/software/FLASH/>

You may also use `bbmerge.sh` to perform a merge:

```
bbmerge.sh in1=frag_1.fastq in2=frag_2.fastq out=merged.fq
```

that prints:

Pairs:	1000000	
Joined:	310150	31.015%
Ambiguous:	689850	68.985%
No Solution:	0	0.000%
Too Short:	0	0.000%

61.9 How well do mergers work?

A detailed guide to validating merged data¹⁰ was written by Brian Bushnell, the author of BBTools¹¹ one of a fascinating bioinformatics software suites (and now you also know where `bb` letters come from in: `bbmap`, `bbrename`, `bbmerge` etc).

We reproduce that guide below in a shortened format.

Get the E.Coli genome.

```
efetch -db nuccore -id MF521836 -format fasta > ecoli.fa
```

Generate random sequences from the E.Coli genome with the `randomreads.sh` tool:

```
randomreads.sh ref=ecoli.fa reads=100000 paired interleaved out=reads.fq.gz len=150 min=10 max=20
```

Rename each read by its length. The names are used later as the validator will checks that the merged fragment matches this size. To use the `bbrename.sh` program, you will have to fully install the `bbtools` package manually.

```
bbrename.sh in=reads.fq.gz int renamebyinsert out1=r1.fq out2=r2.fq
```

Merge reads with `flash`. Creates the `out.extendedFrags.fastq` file.

```
flash -M 150 -x 0.25 r1.fq r2.fq
```

Validate the output. How many wrong merges?

```
grademerge.sh in=out.extendedFrags.fastq
```

¹⁰<http://seqanswers.com/forums/showthread.php?t=43906>

¹¹<https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/>

Prints:

Correct:	99.85591%	74153 reads
Incorrect:	0.14409%	107 reads
Too Short:	0.00943%	7 reads
Too Long:	0.13466%	100 reads

Merge the reads with `bbmerge.sh`. Creates the `merged.fq` file.

```
bbmerge.sh in1=r1.fq in2=r2.fq out=merged.fq
```

Validate the output. How many wrong merges?

```
grademerge.sh in=bb_merged.fq
```

Produces:

Correct:	100.00000%	62346 reads
Incorrect:	0.00000%	0 reads
Too Short:	0.00000%	0 reads
Too Long:	0.00000%	0 reads

Note how `flash` merged 11897 more reads than `bbmap` but it has also made more mistakes when doing so. 214 reads were merged incorrectly. Which one is better?

Above you can see an interesting example of the tradeoffs and decisions that you may need to make along the way.

61.10 Is there anything newer than `fastqc`?

For the sake of revisiting QC yearly, we looked for a new tool for 2017 and found AfterQC: automatic filtering, trimming, error removing and quality control for fastq data.¹² BMC Bioinformatics 2017 18(Suppl 3):80

Let's remind you what we said before,

First, let's start with a mind-blowing observation: Many quality control tools are of meager software quality.

¹²<https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1469-3>

Sure enough, the AfterQC tool requires the installation of Python 2.7 even though Python 3.0 has been released nine years ago! Reviewers still accept tools written in a language version that is now considered obsolete.

Lucky for us, that's why we have environments at our disposal. While we cannot install afterqc into the `bioinfo` environment, we can make a new, Python 2 based environment. Let's get to it:

```
conda create --name py2 python=2.7 -y
```

Activate the new environment:

```
conda activate py2
```

Install AfterQC into this environment

```
conda install afterqc
```

Run the program on

```
after.py -1 illumina.fq
```

Do consult this plot and the documentation for details.

61.11 What is error correction?

Data produced by sequencing instruments may contain errors. The quality value in a FASTQ process is meant to indicate the reliability of a given measurement.

Since the errors are random, it is unlikely that they will occur in the same locations. As a consequence, if the data has high coverage, it is possible to recognize errors by identifying regions that have many measurements in consensus and the very few disagreements are likely to be errors.

The error correction thus can be achieved even without knowing the reference genome by computing the so-called k-mer density of the data. In simple terms, a k-mer is a short subsequence of size k from the data. k-mers that come from real data will be represented in a considerable number of times whereas k-mers that contain errors will be rare.

A FASTQ error correction program will attempt to correct or remove reads that look to have errors in them. Many firmly believe that error correction can make some analysis (especially genome assembly) more efficient.

Also when data is of high coverage, error correction might have a more pronounced beneficial effect.

61.12 How do we correct errors?

The bbmap package includes the `tadpole.sh` error corrector:

```
fastq-dump -X 10000 --split-files SRR519926
```

```
# Tadpole is part of the bbmap package
tadpole.sh in=SRR519926_1.fastq out=tadpole.fq mode=correct out=r1.fq out2=r2.fq
Error correction with bfc https://github.com/lh3/bfc
bfc SRR519926_1.fastq > bfc.fq
```

Part XIII

WRITING SCRIPTS

Chapter 62

The Art of Bioinformatics Scripting

This chapter has been moved into a separate, standalone book.

- The Art of Bioinformatics Scripting¹



¹<https://www.biostarhandbook.com/books/scripting/index.html>

²<https://www.biostarhandbook.com/books/scripting/index.html>

Part XIV

PATTERNS

Chapter 63

Sequence patterns

(This section is incomplete, we are planning to significantly expand it in the future)

63.1 What is a sequence pattern?

A sequence pattern is a sequence of bases described by certain rules. These rules can be as simple as a direct match (or partial matches) of a sequence, like:

- Find locations where ATGC matches with no more than one error

Patterns may have more complicated descriptions such as:

- All locations where ATA is followed by one or more GCs and no more than five Ts ending with GTA

Patterns can be summarized probabilistically into so called **motifs** such as:

- Locations where a GC is followed by an A 80% of time, or by a T 20% of the time, then is followed by another GC

63.2 Can adapters be identified by a pattern?

Yes. Adapters in the data are among the simplest types of sequence pattern that may be present in a data set.

63.3. HOW CAN I SEARCH GENOMIC SEQUENCES FOR PATTERNS? 517

When the our sequences are “line oriented”, that is the entire sequence is on a single line, the `grep` with the `--color=always` parameter will allow us to extract and color the matches:

```
fastq-dump -X 10000 SRR1553606 --split-files  
  
cat SRR1553606_1.fastq | grep --color=always CTGTCTCTTATACA | head -2
```

will produce:



The screenshot shows a terminal window titled "work — ~/edu/work — bash — 103x14". The command entered is \$ cat SRR1553606_1.fastq | grep --color=always CTGTCTCTTATACA | head. The output displays several lines of FASTQ sequence data, with the pattern "CTGTCTCTTATACA" highlighted in red. The user's name is listed as ialbert@yolo.

Figure 63.1

Most FASTQ files can be processed with line oriented tools.

Do not use `--color=always` when saving data in a new file for other downstream processing. When you use coloring, additional color-related data is added to the file, and that will affect downstream processing.

63.3 How can I search genomic sequences for patterns?

`grep` and extended `grep` are tools that operate on one line at a time and can be used to identify lines with patterns. When sequences wrap over

many lines we need dedicated tools like dreg¹ or fuzznuc² (Emboss Tools). See the following chapter [Regular expressions][regexp.md]

```
# This will miss patterns that wrap around new lines.  
cat KU182908.fa | grep AAAAAA
```

```
# The dreg tool matches and reports the locations.  
cat KU182908.fa | dreg -filter -pattern AAAAAA
```

```
# Search a pattern with ambiguous N bases.  
cat KU182908.fa | fuzznuc -filter -pattern 'AANAA'
```

¹<http://emboss.sourceforge.net/apps/cvs/emboss/apps/dreg.html>

²<http://emboss.sourceforge.net/apps/cvs/emboss/apps/fuzznuc.html>

Chapter 64

Regular expressions

(This section is incomplete, we are planning to significantly expand it in the future)

dreg¹ or fuzznuc² (Emboss Tools) are tools that allow you to search the genome for complex patterns.

64.1 What are regular expressions?

Regular expressions are special sequences of characters that define a possibly complex search pattern.

For example

- ^ matches the beginning of the line
- . matches any character
- {m,n} matches the preceding elements at least m but not more than n times.

The Regexp pattern language is unlike most other computing languages. It can be best learned via an interactive service like <https://regexone.com/> or many others where the pattern and its effect are instantly visualized.

Example regular expression searches:

¹<http://emboss.sourceforge.net/apps/cvs/emboss/apps/dreg.html>

²<http://emboss.sourceforge.net/apps/cvs/emboss/apps/fuzznuc.html>

```
# Get a FASTQ dataset.  
fastq-dump --split-files SRR519926  
  
# Find an ATG anchored at the start of the line  
cat SRR519926_1.fastq | egrep "^ATG" --color=always | head  
  
# Find an ATG anchored at the end of the line  
cat SRR519926_1.fastq | egrep "ATG\$" --color=always | head  
  
# Find TAATA or TATTA patterns, this is a range of characters  
cat SRR519926_1.fastq | egrep "TA[A,T]TA" --color=always | head  
  
# Find TAAATA or TACCTA, these are groups of words  
cat SRR519926_1.fastq | egrep "TA(AA|CC)TA" --color=always | head  
  
# Quantify matches with metacharacters  
# Find TA followed by zero or or more A followed by TA  
cat SRR519926_1.fastq | egrep "TA(A*)TA" --color=always | head  
  
# Find TA followed by one or or more A followed by TA  
cat SRR519926_1.fastq | egrep "TA(A+)TA" --color=always | head  
  
# Find TA followed by two to five As followed by TA  
cat SRR519926_1.fastq | egrep "TAA{2,5}TA" --color=always | head  
  
# Practice RegExp matches on online regexp testers  
# http://regexpal.com/  
  
# Match Illumina adaptors at the end of the reads  
# Match AGATCGG anywhere followed by any number of bases  
cat SRR519926_1.fastq | egrep "AGATCGG.*" --color=always | head
```

64.2. WHAT ARE SOME CHALLENGES I MAY FACE WHEN USING REGULAR EXPRESSIONS??

64.2 What are some challenges I may face when using regular expressions?

Think of a regular expression as a computing engine. There are different implementations of it, and each may operate with slightly different syntax, and at different speeds or efficiencies.

Get chromosome 22 of the Human Genome

```
curl http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz | gunzip >
```

Let's find the telomeric repeats TTAGGG in the human genome. In principle this is easy: build and match the pattern. In practice, when we search at a genomic scale, things don't always go as planned.

Let's first check that this pattern is present. Use the `-i` flag to make the match case insensitive since the genome may lowercase regions (for example the repeating regions are marked as such in the human genome):

```
cat chr22.fa | egrep -i '(TTAGGG)' --color=always
```

The above won't work perfectly since this is line oriented matching and the genome wraps over many lines. But it is a good start. We can refine it more

```
cat chr22.fa | egrep -i '(TTAGGG){3,10}' --color=always
```

Then we can “linearize” the genome by removing the new line wrapping characters from it. That way we can look for the actual pattern:

```
cat chr22.fa | tr -d '\n' | egrep -o -i '(TTAGGG){20,30}' --color=always
```

To produce:

The same can be done with `dreg`, though note that its regular expression engine is less powerful and may not be able to process all patterns:

```
cat chr22.fa | dred -filter -pattern '(TTAGGG){20,30}'
```

Chapter 65

Sequence k-mers

(This section is incomplete, we are planning to significantly expand it in the future)

65.1 What is a k-mer?

A **k-mer** typically refers to all the possible substrings of length **k** that are contained in a string. For example if the sequence is ATGCA then

- The 2 base long **k-mers** (**2-mers**) are AT, TG, GC and CA
- The 3 base long **k-mers** (**3-mers**) are ATG, TGC and GCA
- The 4 base long **k-mers** (**4-mers**) are ATGC, TGCA
- The 5 base long **k-mer** (**5-mer**) is ATGCA

65.2 What are k-mers good for?

The potential information gained from k-mers is evolving quickly. Interesting applications using k-mers are:

- Error correction: rare k-mers are more likely to be caused by sequence errors.
- Classification: certain k-mers may uniquely identify genomes.
- Pseudo-alignment: new pseudo-aligners can match reads to locations based solely on the presence of common k-mers.

Computing k-mers is much faster than producing alignments – hence they can massively speed up data interpretation.

Using the jellyfish¹ k-mer counter:

```
# Get some sequence data.  
efetch -id KU182908 -db nucleotide -format fasta > KU182908.fa  
  
# Count the k-mers up to size 10.  
jellyfish count -C -m 10 -s10M KU182908.fa  
  
# Show a histogram of k-mers.  
jellyfish histo mer_counts.jf  
  
# The k-mers present at least 7 times.  
jellyfish dump -L 7 mer_counts.jf  
  
# Pick one k-mer, say TTAAGAAAAAA - is it present 7 times?  
cat KU182908.fa | dred -filter -pattern TTAAGAAAAAA
```

65.3 Should I use the k-mer report produced by FastQC?

The k-mer plot that FastQC produces should be interpreted with caution as it can be misleading. Both the scaling of the plot and the ratios of observed versus expected counts are flawed. Longer k-mers will be reported many times, but shifted by one base.

¹<http://www.cbcu.umd.edu/software/jellyfish/>

Part XV

ALIGNMENTS

Chapter 66

Introduction to alignments

66.1 What is a sequence alignment?

Sequence alignment (specifically pairwise alignment) means arranging two sequences in a way so that regions with similarities line up. For example alignment of GATTACA and GATCA could look like:

```
GATTACA  
||| |  
GATCA--
```

When judged by eye it does not take much for alignments to make sense intuitively, but be warned that our intuition and the apparent clarity is grossly misleading.

Our brains are hardwired to see patterns and similarities whenever possible, and once a similarity is shown, we are often unable to recognize other alternatives that may be just as good (if not better) as the one we are currently observing. It is hard to “un-see” a pattern. For example, we could have arranged our example words like this

```
GATTACA  
|||  ||  
GAT--CA
```

or we could have shown them like this:

```
GATTACA
```

|| | ||

GA-T-CA

Now we have two more alignments for the same words, that, when displayed look to be matching up just as well. Once you look at one pattern, it may take quite a bit of effort to notice that you could have also arranged the letters in other ways that seem just as appropriate.

66.2 What are alignments used for?

Alignments have two primary use cases:

1. Finding similar regions between two sequences - probing the relatedness of sequences.
2. Finding which sequence (from many alternatives) is most similar to a query (input) sequence

66.3 What governs an alignment?

The following two factors determine every alignment:

1. Alignment algorithm: global, local, semi-global
2. Alignment scoring (parameters): the numerical values that tune the arrangement

Thus always remember that:

- Different algorithms will typically produce different alignments for the same sequences.
- Different scoring will usually generate different alignments for the same sequences

66.4 The inconvenient truth about alignments

When the sequences are very similar (nearly identical) the choice of scoring or even algorithms may not matter at all. The results will be *robust* - producing

identical alignments across different algorithms and parameter settings.

When the sequences are dissimilar the choice of algorithm and scoring will typically have radical impacts on the results. In general, the more different the sequences, the more sensitive the alignment becomes to parameter choices.

In our opinion, the lack of understanding of the above constraints is the root cause of quite numerous invalid scientific reports. Scientists, having explored situations where the sequences were similar, and the results were robust, develop a false sense of understanding and “comfort” with the process and will start applying the same strategies for situations where the alignment processes are susceptible to parameter choices.

66.5 How are alignments displayed?

There are several ways that alignments can be reported, and there is no simple, universal format that can present all the information encoded in an alignment.

Often, we use a visual display that employs various extra characters to help us “interpret” the lineup. For example:

- the - character may indicate a “gap” (space),
- the | character is used to display a match,
- the . character may be used to display a mismatch.

For example the following is an alignment between two sequences ATGCAAATGACAAATCGA and ATGCTGATAACTGCGA :

```
ATGCAAATGACAAAT-CGA
||||| |||.||.| ||
ATGC---TGATAACTGCGA
```

Above 13 bases are the same (13 identities), 5 bases are missing (5 gaps), 2 bases are different (2 mismatches).

66.6 Does it matter which sequence is on top?

Usually, if not stated explicitly otherwise, we read and interpret the alignment as if we were comparing the bottom sequence against the top one.

```
ATGCAAATGACAAAT-CGA
||||| ||||.||.|| |
ATGC---TGATAACTGCGA
```

Whereas the word gap is generic and refers to either sequence, once we want to be more specific we would say that the above is an alignment with three deletions of A and one insertion of G. The word “deletion” means that the second sequence has missing bases relative to the first.

We could generate and display this same alignment the other way around:

```
ATGC---TGATAACTGCGA
||||| ||||.||.|| |
ATGCAAATGACAAAT-CGA
```

The alignment would now be described as one that contains three insertions of As followed later by a deletion a G relative to the top sequence. One sequences’ deletion is the other sequences’ insertion - it all depends on what the study is about.

66.7 How are alignments generated?

Suppose you had the three alignments GATTACA vs GATCA:

GATTACA	GATTACA	GATTACA
.		
GATCA--	GAT--CA	GA-T-CA

Which one is the “best”, “right”, “correct”, “proper”, “meaningful” alignment?

What would you consider to be the *best* alignment?

You see the best alignment will depend on how you “value” the way bases line up. Do you consider one mismatch to be less disruptive than gaps? The *value* you associate with a match, mismatch or a gap is called *scoring*.

The most important concept of alignments is the following:

There is no *universally best* alignment. There is only the *best alignment* relative to a *scoring choice*. Changing the score will usually change what is selected as the best alignment. Alignment algorithms find the arrangement that produces the maximal score over the entire alignment.

Sequence alignment algorithms speed up the process of selecting the alignment with the best score. You can think of aligners as methods for quickly evaluating all possible combinations then, out of those picking the alignments with the best score.

66.8 How does alignment scoring work?

The scores are the values, both positive and negative, that an algorithm will assign to various bases when they line up a certain way. An aligner attempts to produce an arrangement that **maximizes** the score.

For example, we could choose the following scores:

- 5 points for a match.
- -4 points for a mismatch.
- -10 points for opening a gap.
- -0.5 points for extending an open gap.

We then instruct the aligner to find the best alignments using the scores above. With this scoring system, the alignments would be scored as (scores shown at the bottom):

GATTACA	GATTACA	GATTACA
.		
GATCA--	GAT--CA	GA-T-CA

5.5

14.5

5

With this scoring, the second alignment has the highest score and will be considered the “best” (highest scoring) alignment. But we have a peculiar situation to deal with here. Most biological protocols produce errors towards the ends of sequences. Thus in practice, we frequently end up with sequences

that are shorter than we expect. To account for this type of error a scoring ‘adjustment’ is frequently applied, one that will not penalize a gap at the end of either sequence. If we apply that correction the scores become:

GATTACA	GATTACA	GATTACA
.		
GATCA--	GAT--CA	GA-T-CA
16	14.5	5

Now the first alignment ended up with a higher score than the others, and with that, it became the “best alignment”.

66.9 How do I choose the scores?

In most cases, we start with known scores, computed by observing the substitution rates across evolutionary history. Since there are different ways scientists can consolidate their observations - there will be various alternative scoring matrices. Appropriate starting points already exist. See for example `ftp://ftp.ncbi.nlm.nih.gov/blast/matrices`

For example, the EDNAFULL¹ scoring matrix is the default nucleotide scoring choice for just about all aligners. Typically, unless you set the scores yourself, you will use these values:

```
curl -O ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/NUC.4.4
```

then print it with:

```
cat NUC.4.4
```

The actual matrix is more extensive than what we show below, as it includes all the ambiguous bases as well (refer to the biology intro for information on those); the following section is relevant:

	A	T	G	C
A	5	-4	-4	-4
T	-4	5	-4	-4
G	-4	-4	5	-4

¹<ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/NUC.4.4>

C -4 -4 -4 5

This section shows that the score contribution from matching an A with A is 5 points, whereas the score contribution from matching an A with a T is -4 points (i.e., a score penalty of 4 points).

Choosing the right matrix, especially for protein sequences is more complicated. Refer to the following publication for more details:

- Selecting the Right Similarity-Scoring Matrix² by William Pearson, the author of the FASTA program.

66.10 What kind of scoring matrices are there?

There are two types: nucleotide scoring matrices and protein scoring matrices. The protein scoring matrices come in many variants computed with different assumptions on what similarity means. Also, (if that previous wasn't enough of complexity) the scoring matrices can be normalized and non-normalized. Care must be taken when comparing alignment scores computed with different scoring matrices, even if they seem to have the same name! One scoring matrix could be normalized and the other not.

Visit: <ftp://ftp.ncbi.nlm.nih.gov/blast/matrices> for a list of matrices.

66.11 What other properties do scoring matrices have?

Observe that the scoring matrix does not include the information on the gap penalties. It is also worth noting that the gap opening and extension penalties are typically different.

Specifically, the gap extension penalty is usually much smaller than the gap opening one. This scoring method (also called affine gap penalty³) has a biologically relevant rationale that you may want to read up on your own.

²<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3848038/>

³https://en.wikipedia.org/wiki/Gap_penalty

66.12. ARE THERE OTHER WAYS TO QUANTIFY ALIGNMENTS?533

It is essential to recognize just how impactful the choice of scoring is and how profound its effects are because scoring governs how many gaps versus mismatches the aligner produces.

A final note: a custom scoring matrix can only work in a biologically meaningful way if it sums up to negative values along both of its rows and columns. It is also worth keeping in mind that the scores are typically on a logarithmic scale.

66.12 Are there other ways to quantify alignments?

Beyond the score of an alignment, scientists may also make use of concepts such as

- Percent Identity: What percentage of the two sequences are the same
- Percent Similarity: What proportion of the two sequences have similar bases/amino acids
- E-Values: Number of observed alignments by chance
- Mapping quality: The likelihood of incorrect alignment

66.13 What is a CIGAR string?

The CIGAR string (Compact Idiosyncratic Gapped Alignment Report) is an alignment format used within the Sequence Alignment Map (SAM) files that form the backbone of most bioinformatics high-throughput analyses. For the same alignment from above:

```
ATGCAAATGACAAATAC
||||| ||||.||.|
ATGC---TGATAACT--
```

the reported CIGAR format would be:

4M3D3M1X2M1X1M2D

We read out this “idiosyncratic” construct like so 4M + 3D + 3M + 1X + 2M + 1X + 1M + 2D:

- 4 matches followed by
- 3 deletions,
- 3 matches,
- 1 mismatch,
- 2 matches,
- 1 mismatch,
- 1 match,
- 2 deletions.

The format above is in a so-called “Extended CIGAR,” meaning that it employs the X symbol for mismatches.

Another variant where *both* matches and mismatches are designated by an M (as odd as that might sound) is the format used within the Sequence Alignment Map (SAM). In that format the extended CIGAR string of

4M3D3M1X2M1X1M2D

would be written as:

4M3D7M2D

Note how the 3M + 1X + 2M + 1X will be 7M forming the the sum of consecutive matches or mismatches. The value of this latter CIGAR form is its compactness and an emphasis on insertions and deletions - though in our opinion, in most cases it sacrifices too much information to be practical.

Finally, there is yet another idiosyncratic way to write CIGARs (don’t you love it when a funny, self-deprecating name ends up being quite deservedly so?). In that format for a single base long change the number is not shown like so:

10M1D10M

would be written as

10MD10M

You may see CIGAR notations in any of the variants above.

66.14 Where can I learn more about alignments?

There is a large body of work and information on alignments and scoring that are beyond the scope of this book. There are good starting points in Wikipedia, as well as in many other easy-to-find web resources.

- Wikipedia: Sequence Alignment⁴
- Wikipedia: Gap Penalty⁵

⁴https://en.wikipedia.org/wiki/Sequence_alignment

⁵https://en.wikipedia.org/wiki/Gap_penalty

Chapter 67

Global and local alignments

67.1 Install the helper scripts

For this book, we have prepared two scripts:

- global-align.sh¹
- local-align.sh²

that will allow us to demonstrate alignments from the command line. Both of these scripts rely on programs in the Emboss³ package. The same programs can also be run online via the Ensembl Pairwise Alignment Tool⁴ webpage. To install the alignment helper scripts run the following:

```
# Store the program in the bin folder.  
mkdir -p ~/bin  
  
# Install the wrapper for the EMBOSS alignment tools.  
curl http://data.biostarhandbook.com/align/global-align.sh > ~/bin/global-align.sh  
curl http://data.biostarhandbook.com/align/local-align.sh > ~/bin/local-align.sh  
  
# Make the scripts executable.  
chmod +x ~/bin/*-align.sh
```

¹<http://data.biostarhandbook.com/align/global-align.sh>

²<http://data.biostarhandbook.com/align/local-align.sh>

³<http://emboss.sourceforge.net/what/>

⁴<http://www.ebi.ac.uk/Tools/psa/>

Verify that the script works with:

```
local-align.sh THISLINE ISALIGNED
```

The scripts are written in such a way that you may pass either text strings to it or FASTA file names instead of the words.

```
local-align.sh sequence1.fa sequence2.fa
```

Besides, you may pass scoring matrices and other parameters with:

```
wget -nc ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/BLOSUM30
local-align.sh THISLINE ISALIGNED --data BLOSUM30 -gapopen 5
```

Note: we found these helper scripts to be quite useful outside of the classroom as well. Every once in a while when we quickly need to align and visualize two sequences, we found ourselves reaching for them. Even though we know of many other alternatives, the two scripts above are still among the most straightforward and simple choices.

In the examples below, we will use hypothetical protein sequences **THISLINE** and **ISALIGNED**, as these form real words that are easy to read and so better demonstrate the differences between alignments. These sequences were first used for a similar purpose, albeit in a different context, in *Understanding Bioinformatics* by Marketa Zvelebil and Jeremy Baum

67.2 What is a global alignment?

In global alignments, the bases of both sequences are arranged next to one another over their entire length. Each base of the first sequence is matched to another base or a “gap” of the second sequence.

We use global alignments when we need to look for the similarities over the entire length of both sequences.

For example,

```
global-align.sh THISLINE ISALIGNED
```

produces:

```
THISLI--NE-
 ||.: ||
--ISALIGNED
```

We may also override the gap open and gap extension penalty:

```
global-align.sh THISLINE ISALIGNED -gapopen 7
```

Now, this choice of parameter produces a different alignment:

```
THIS-LI-NE-
|| || ||
--ISALIGNED
```

Note how radically different the second alignment is from the first one. All we did is reduce the penalty of opening a gap to from 10 to 7. The alignment is longer but has more gaps. The tradeoff is readily apparent.

The full list of parameters to the scripts are:

```
-gapopen
-gapextend
-data
```

Where the `-data` parameter is used to pass a different scoring matrix to the aligner.

```
wget -nc ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/BLOSUM30
global-align.sh THISLINE ISALIGNED -data BLOSUM30
```

will produce:

```
THIS--LINE-
|| : ||
ISALIGNED
```

67.3 What is a local alignment?

Local alignments are used when we need to find the region of maximal similarity between two sequences. When performing local alignments, the algorithms look for the highest scoring (partial) interval between the two sequences :

```
local-align.sh THISLINE ISALIGNED
```

When run as above the local alignment generated with the default parameters will be surprisingly short:

```
NE  
||  
NE
```

The algorithm is telling us that these two matching amino acids produce the highest possible score (11 in this case) and any other local alignment between the two sequences will produce a score that is worse than 11.

We can use other scoring matrices as shown in <ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/>:

```
# DNA matrices  
# This matrix is the "standard" EDNAFULL substitution matrix.  
wget ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/NUC.4.4  
  
# Protein matrices  
# Get the BLOSUM30, BLOSUM62, and BLOSUM90 matrices.  
wget ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/BLOSUM30  
wget ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/BLOSUM62  
wget ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/BLOSUM90
```

Note how the local alignment is affected by the scoring scheme.

```
local-align.sh THISLINE ISALIGNED -data BLOSUM90
```

Using the BLOSUM90 scoring scheme produces a much longer alignment:

```
SLI-NE  
:|| ||  
ALIGNE
```

How do you picking the right matrix? Let us again refer to the paper:

- Selecting the Right Similarity-Scoring Matrix⁵ by William Pearson, the author of the FASTA program.

Here are a few lines from the abstract:

While “deep” matrices provide very sensitive similarity searches, they also require longer sequence alignments and can sometimes produce alignment overextension into non-homologous regions. Shallower scoring matrices are more effective when searching for

⁵<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3848038/>

short protein domains, or when the goal is to limit the scope of the search to sequences that are likely to be orthologous between recently diverged organisms.

Likewise, in DNA searches, the match and mismatch parameters set evolutionary look-back times and domain boundaries.

67.4 Why are scoring values integer numbers?

You could wonder, rightfully so, how come the scores are all integers? And also what does 3 vs. 5 mean in a scoring matrix?

In a nutshell, the score reflects a probability. Besides, the scores are represented as log 2 odds. A substitution score of 3 means $2^3=8$ whereas a substitution score of 5 means $2^5=32$, thus a four-fold increase of probability relative to one another.

Thus a substitution with a score of 3 is four times more likely to occur than a change with a score of 5. For simplicity and to keep our sanity the probabilities log2 odds were all rounded to nearest integers.

Note how the score is not absolute; they are relative to one another. Any other number pars with the same ratios would have the same effect.

67.5 What is a semi-global alignment?

Semi-global alignments (also known as global-local, glocal) are cross between the global and local alignments.

Semi-global alignments attempt to completely align a shorter sequence against a longer (reference) one.

Semi-global aligners are used when matching sequencing reads produced by sequencing instruments against reference genomes. Most high throughput data analysis protocols that we cover in this book rely on tools that use this type of alignment. ## Are scores computed the same way?

One commonly used variant of both global and semi-global alignments is the free-end-gap method, where gaps at the end of the alignment are scored differently and with lower penalties.

It is not always clear from the description of the software when this choice is made. Do note that most tools used to investigate biological phenomena use free-end-gap scoring.

Chapter 68

Misleading alignments

68.1 Will alignments always indicate the “correct” variation?

We want to show a situation that you may encounter to demonstrate the limitations of using mathematical concepts to identify biological phenomena.

Imagine that the sequence below is subjected to two insertions of Cs at the locations indicated with carets:

CCAAACCCCCC[^]TCCCCCGCTTC
 ^ ^

The two sequences, when placed next to one another, would look like this:

CCAAACCCCCC[^]TCCCCCGCTTC
CCAAACCCCCC[^]C[^]TCCCCCGCTTC

A better way to visualize what's happening in this example is shown in the expected alignment that would reflect the changes that we have introduced:

CCAAA-CCCCCC[^]T-CCCCGCTTC
||||| ||||| | | | | | | | |
CCAAACCCCCC[^]CTCCCCCGCTTC

Now suppose we did not know what the changes were. We want to recover

68.1. WILL ALIGNMENTS ALWAYS INDICATE THE “CORRECT” VARIATION?543

the alignment above to identify the two insertions from the sequences alone. Can we discover the variation that we introduced by using a global aligner? Let's see:

```
global-align.sh CCAAACCCCCCTCCCCGCTTC CCAAACCCCCCTCCCCCGCTTC
```

Here is what we obtain:

```
CCAAACCCCCC--TCCCCCGCTTC  
||||| .|||||||  
CCAAACCCCCCCCCTCCCCCGCTTC
```

Whoa! What just happened?

Our aligner reports our two separate and distant insertions of Cs as a single insertion of a CT followed by a mismatch of T over C.

To add insult to injury, the change is shown to be in a completely *different location*, not even where we have modified the sequence! That's just crazy! How did that happen? What can we do about it?

In a nutshell, there is a “simpler” explanation to reconcile the differences between the two sequences - and the aligner, using mathematical reasoning produces the simplest explanation - even though this explanation does not reflect what happened. Some people use the word “misalignment”, but that would be incorrect - the aligner did its job correctly, but there was no way for it to know that a more complicated explanation was the correct one.

The main reason for the existence of this “simpler explanation” is that the region between the two insertions has low information content; the same base is repeated: CCCCCC. This repetition throws off the neatly-crafted mathematical alignment concept of rewarding matching bases. When shifted by one base, we still get the same type of bases lined up, hence producing a positive score even though it is the “newly inserted” C that matches the “original” C. Let's compute the scores.

In our “correct” alignment, and when using the EDNAFULL matrix, the score will be formed from 23 matches and 2 gap openings:

$$(23 * 5) - (2 * 10) = 95$$

In the “incorrect” second alignment we have only 22 matches with 1 gap open, 1 gap extension and 1 mismatch leading to a score of

$$(22 * 5) - 10 - 0.5 - 4 = 95.5$$

An optimal aligner finds the alignment that maximizes the score. We now see that the second, “incorrect” alignment produces a slightly better score; hence, it will be reported as the most likely match between the two sequences.

We could recover the “correct” alignment by modifying the parameters to reduce the gap open penalty:

```
global-align.sh CCAAACCCCCCCTCCCCCGCTTC CCAAACCCCCCCTCCCCCGCTTC -gapopen 9
```

Now it will produce the expected output

```
CCAAA-CCCCCCCT-CCCCCGCTTC
||||| ||||||||| |||||||||
CCAAACCCCCCCCCTCCCCCGCTTC
```

Does this mean that from now on we should run all alignments with `gapopen=9`?

Absolutely not!

As we mentioned before, tweaking alignment parameters can have far-reaching consequences on all other alignments that are typically correct. Using this setting means that two matches $5 + 5 = 10$ will overcome the penalty of a gap open 9; hence, the aligner will open gaps any time it can find two matches later on. Sometimes that is a good idea, but that is most likely not what we intended to do.

The problem above was caused not by the incorrect alignment parameters but by the reduced information content of the series of Cs (aka homopolymeric) region.

Situations such as the above will generally produce incorrect variation calls and are an ongoing, profound, and ubiquitous challenge when identifying genomic variations. As we shall see, the latest SNP calling tools have features that allow them to recognize and correct “misalignments” - though as you can imagine the problems are far from being solved.

A universal take-home message:

Alignment reliability also depends on the information content of the aligned sequence itself.

Alignments that operate on low complexity regions are generally produce less reliable variation calls.

Part XVI

BLAST

Chapter 69

BLAST: Basic Local Alignment Search Tool

69.1 What is BLAST?

BLAST is an acronym for **Basic Local Alignment Search Tool**. BLAST represents both an algorithm and a suite of tools that implement said algorithm. The primary purpose of BLAST is to search a collection of target sequences for alignments that match a query sequence.

Using BLAST is a bit like using Google when searching for information. When we perform a Google query, we are looking to find documents that contain the words that we are seeking. Similarly, BLAST search results are sequences that have similarities to the query sequence. As the name says the results of a BLAST search are local alignments, thus a blast result is generally a partial match of the query sequence to a target sequence in the database.

BLAST can be run both as a web interface from NCBI¹ and as standalone downloadable tools. NCBI also maintains a dull to read Blast Handbook² that contains further information on how BLAST works.

¹<http://blast.ncbi.nlm.nih.gov/Blast.cgi>

²<https://www.ncbi.nlm.nih.gov/books/NBK279690/>

69.2 What are the BLAST tools?

BLAST consists of a suite of tools

- `blastn`, `blastp`, `blastx`, `tblastn`, `tblastx` and others

Note that each tool then can be further customized via so-called tasks specified with the flag `-task`. Think of these tasks as important tuning parameters: `megablast`, `blastp-short` that will radically change how the search operates. When communicating research findings generally speaking it is insufficient to state which blast program one run; the task should be mentioned as well. Remember that even if you don't explicitly choose a task, the software tool will select one for you by default. For example, the '`blastn`' tool, will, by default run the with the `megablast` task.

BLAST is not an optimal aligner. What this means is that it may not find all alignments and there are limits for how short and how long sequences may be for BLAST to operate correctly. Also, BLAST is tuned for performance and efficiency; its primary purpose is to search a large body of known information for similarities (hits). To quote the World's Most Interesting Man: *I don't generally misuse BLAST, but when I do I make huge mistakes!*. What this means is that BLAST will not typically miss a hit, or produce a wrong alignment. What happens more frequently is that it is used in a manner and context that is entirely wrong. Thankfully “big” mistakes are easier to catch.

69.3 What are the fundamental concepts of BLAST?

- A search may take place in nucleotide space, protein space or translated spaces where nucleotides are translated into proteins.
- Searches may implement search “strategies”: optimizations to a specific task. Different search strategies will produce different alignments.
- Searches use alignments that rely on scoring matrices
- Searches may be customized with many additional parameters.

BLAST has many parameters that most users may never need to use or set. Knowing BLAST well can be a “full-time job” - there are job positions that

mostly consists of using BLAST in various contexts. There are books and courses on just how to tune and operate BLAST searches.

NOTE: Biologists love BLAST! It is easy to see why. When it works, it feels like hitting the jackpot, similar to when you are stuck and Googling a word takes you to the answer you were looking for all along. When we search with BLAST the NT or NR databases we are navigating through the accumulated knowledge that mankind has collected, it is incredibly empowering.

But there is a downside to all this power - as our knowledge grows the possible rate of false discovery increases faster than what most scientists assume. From a purely evolutionary point of view, every sequence ought to be “similar” to any other sequence as they share common ancestry (distant as it may be). As we add more sequences onto the same pile the picture gets fuzzier and less reliable for making precise statements. Knowing where to draw the line, which similarities are extreme events of random chance and which are meaningful is a far more complicated task than most scientists care to admit.

69.4 Can BLAST be used for pairwise alignments?

While pairwise alignment is not the primary use case of BLAST, the tools in the suite do support the function. Here is an example of running BLAST to align the coding sequence for the nucleoprotein named NP across the 1976 and 2018 Ebola strains.

```
# The Ebola nucleoprotein in the 1997 strain
efetch -db nuccore -id AAD14590 --format fasta > AAD14590.fa

# The Ebola nucleoprotein in the 2018 strain
efetch -db nuccore -id ARG44037 --format fasta > ARG44037.fa

# Run a pairwise BLAST alignment
blastp -query AAD14590.fa -subject ARG44037.fa
```

that will produce the output:

```
Score = 1512 bits (3915), Expect = 0.0, Method: Compositional matrix adjust.
Identities = 728/739 (99%), Positives = 730/739 (99%), Gaps = 0/739 (0%)

Query 1 MDSRPQKIWMAPSLTESDMYHKILTAGLSVQQGIVRQRVIPVYQVNNLEEICQLIIQAF 60
        MDSRPQK+WM PSLTESDMYHKILTAGLSVQQGIVRQRVIPVYQVNNLEEICQLIIQAF
Sbjct 1 MDSRPQKVWMTPSLTESDMYHKILTAGLSVQQGIVRQRVIPVYQVNNLEEICQLIIQAF 60

Query 61 EAGVDFQESADSFLMLCLHHAYQGDYKLFLESGAVKYLEGHGFRFEVKKRDGVKRLEEL 120
        EAGVDFQESADSFLMLCLHHAYQGD+KLFLESGAVKYLEGHGFRFEVKKRDGVKRLEEL
Sbjct 61 EAGVDFQESADSFLMLCLHHAYQGDHKLFFLESGAVKYLEGHGFRFEVKKRDGVKRLEEL 120

Query 121 LPAVSSGKNIKRTLAAMPEEETTEANAGQFLSFASLFLPKLVVGEKACLEKVQRQIQVHA 180
        LPAVSSGKNIKRTLAAMPEEETTEANAGQFLSFASLFLPKLVVGEKACLEKVQRQIQVHA
Sbjct 121 LPAVSSGKNIKRTLAAMPEEETTEANAGQFLSFASLFLPKLVVGEKACLEKVQRQIQVHA 180
...
```

The report states that the sequences match at the 99% level and 728 out of 739 bases are identical. Now reading default BLAST outputs is also a skill that develops over time. Thankfully there are various ways we can reformat the output. If all we wanted was to know the percent identities we could have run `blastp` using a column based output:

```
blastp -query AAD14590.fa -subject ARG44037.fa -outfmt '6 pident'
```

and that would have produced just one value:

98.512

69.5 How do I use BLAST?

The most common use of blast requires the following three steps:

1. Prepare a BLAST database with `makeblastdb`. This task only needs to be done once per sequence data.
2. Select the appropriate BLAST tool: `blastn`, `blastp`. Depending on your needs you may need to tune the parameters.
3. Run the tool and format the output as necessary.

For example, suppose we want to investigate in what way has the sequence of the polymerase gene VP35 changed since 1976. To do so, we will align the 1976 version of VP35 to all protein sequences obtained during the 2014 outbreak. We also know that the data for the 1976 strain is deposited under the accession number AF086833.

First, we need to find the accession number for the VP35 polymerase gene from 1976. We could, or we could see it inside the GenBank file.

```
# Get the Ebola genome as genbank.  
efetch -db nuccore -id AF086833 -format gb > AF086833.gb  
  
# Look through the file for gene name VP35 and context around it.  
cat AF086833.gb | grep -A 1 VP35
```

You can start with more additional lines for the `-A` flag, in this case a single line is sufficient. Among other hits we see that:

```
/product="VP35"  
/protein_id="AAD14582.1"
```

Thus we have the accession number. Will get that data.

```
# Fetch the sequence for the VP35 protein.  
efetch -db protein -id AAD14582 -format fasta > AAD14582.fa
```

We will now get all proteins deposited for project PRJNA257197 described in Redo: Genomic surveillance elucidates Ebola virus origin:

```
# Make a directory to hold the sequences  
mkdir -p db  
  
# Fetch the sequences for the project.  
esearch -db protein -query PRJNA257197 | efetch -format fasta > db/proteins.fa  
  
# Run a sanity check to verify it all went well.  
seqkit stats db/proteins.fa
```

it produces showing 2240 sequences with over a million bases in total. This will be our search space:

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
db/proteins.fa	FASTA	Protein	2,240	1,367,025	217	610.3	2,212

In summary, our BLAST query will be `AAD14582.fa` and we want to search the sequences in `db/proteins.fa`.

```
# Make the blast database
makeblastdb -dbtype prot -in db/proteins.fa -out db/proteins
```

Our query is a protein sequence, and the target database is a protein sequence; thus the appropriate BLAST tool is `blastp`.

```
# Run the blastp tool
blastp -db db/proteins -query AAD14582.fa > results.txt

cat results.txt | wc -l
# prints 8021
```

The results will be saved into the file `results.txt` - as you will see it is a file with 8000 lines, lots of alignments, overall not so easy to make sense of - there are just too many hits. It is best to reformat the output to list only the columns you might be interested in. Let's reformat to tabular form:

```
blastp -db db/proteins -query AAD14582.fa -outfmt 7 | head
```

will produce:

```
# BLASTP 2.7.1+
# Query: AAD14582.1 VP35 [Ebola virus - Mayinga, Zaire, 1976]
# Database: db/proteins
# Fields: query acc.ver, subject acc.ver, % identity, alignment length, mismatche
# 249 hits found
AAD14582.1 AIG95885.1 98.824 340 4 0 1 340 1 340 0.0 702
AAD14582.1 AIG95894.1 98.824 340 4 0 1 340 1 340 0.0 702
AAD14582.1 AIG95903.1 98.824 340 4 0 1 340 1 340 0.0 702
AAD14582.1 AIG95912.1 98.824 340 4 0 1 340 1 340 0.0 702
AAD14582.1 AIG95921.1 98.824 340 4 0 1 340 1 340 0.0 702
```

Now the task of making sense of this file is where the challenges begin. How do I identify what is essential, what is not, which columns should I have (there are many more available), which columns are relevant, and so on. At this point running BLAST was the “easy” part - making sense of data? That is always an uphill battle.

```
# Three best alignments
blastp -db db/proteins -query AAD14582.fa -outfmt "6 qseqid sseqid pident" | sort
```

```
# The worst alignments.  
blastp -db db/proteins -query AAD14582.fa -outfmt "6 qseqid sseqid pident" | sort -k3 -rn
```

69.6 What are the blast tools?

Blast is an entire suite of somewhat confusingly named programs.

- `blastn`, `blastp`, `tblastn`, `blastx` etc. Each tool performs a certain type of query (nucleotide or peptide) versus a certain type of target (nucleotide or peptide) using a certain type of translation (nucleotide or peptide). It is often difficult to remember which blast tool does what as the naming is not all that logical.

As strange as that might sound when figuring out which blast tool is more appropriate, it makes more sense to try to remember what each tool should have been named (using the strategy below) instead of what they currently named.

1. A blast search tool where both the query and target are in *nucleotide* space should have been called `blastNN` (but instead it is called just `blastn`).
2. A blast tool that searches a *peptide* sequence against a *nucleotide* sequence in the translated space should have been called `blastPN` but instead, it is called `tblastn`

Here's a helpful table that clarifies the blast tools:

69.7 What is the Blast terminology?

- **Query:** the sequence that we are looking to match.
- **Target:** the database (collection of sequences) that we are searching for matches.
- **Subject:** the sequence entry in the target that produced an alignment.
- **Score:** the alignment score.
- **E-Value:** the Expect value (E) describes the number of hits better than the current hit that one can “expect” to see just by chance when searching a database of a particular size.

Query sequence type	Database sequence type	Alignment level type	What the program should be called	What the program is actually called
nucleotide	nucleotide	nucleotide	blastNN	blastn
peptide	peptide	peptide	blastPP	blastp
nucleotide	peptide	peptide	blastNP	blastx
peptide	nucleotide	peptide	blastPN	tblastn
nucleotide	nucleotide	peptide	blastNNP	tblastx

Figure 69.1

69.8 What is an E-Value

The e-values are among the most misunderstood and misused concepts of BLAST. E-values were designed as the means of bringing some level of confidence to the search results, and are defined as the number of hits one can “expect” to see by chance when searching a database of a particular size. A smaller the e-value is “better”.

The e-value was intended to be used as a filtering and threshold parameter, where, supposedly by using them we could identify more “trustworthy” hits. Fundamentally the problem with e-values is that they depend on the database size and content, hence are not a measure of validity, reliability or correctness. Detailed definitions for e-values come in many shapes and forms, from [overcomplicated mathematics and statistics][evaluestats] that few people could possibly follow to hand-waving rationalizations like the ones seen for example Chapter 9: BLAST QuickStart³

³<https://www.ncbi.nlm.nih.gov/books/NBK1734/>

The alignments found by BLAST during a search are scored, as previously described, and assigned a statistical value, called the “Expect Value.” The “Expect Value” is the number of times that an alignment as good or better than that found by BLAST would be expected to occur by chance, given the size of the database searched. An “Expect Value” threshold, set by the user, determines which alignments will be reported. A higher “Expect Value” threshold is less stringent and the BLAST default of “10” is designed to ensure that no biologically significant alignment is missed. However, “Expect Values” in the range of 0.001 to 0.0000001 are commonly used to restrict the alignments shown to those of high quality.

After reading the above you may ask yourself what is a “high quality” alignment? And why would an alignment with an e-value of 0.0001 be of “lower quality” than one with an e-value of 0.0000001? It very much sounds like “quality” is just a synonym for, surprise!, e-values! Why would one ever set an e-value to 0.0000001 when searching for a single hit?

You see e-values in their current form are designed for projects that attempt to brute-force a problem by aligning even the kitchen sink to everything else, with the hopes that something interesting eventually pops up. For any other project with a realistic basis, e-values are just a hindrance and little more than a chance to derail the information you get..

Ranking blast search results by e-values is pervasive and prevalent. Since both the alignment lengths and the alignment scores are ingredients to the e-value formula, ranking by e-values typically shows a strong correlation to ranking by score and lengths. E-values end up as a single “convenient” number that empirically seems to work well.

Chapter 70

BLAST use cases

In this section we demonstrate a few use cases for BLAST and we will explore common problems and errors that you might encounter.

70.1 How do I build custom blast databases?

Let's build a database out of all features of the 2014 Ebola genome deposited under accession number KM233118.

```
# Database directory.  
mkdir -p db  
  
# Get the 2014 Ebola genome as GenBank file.  
efetch -db nucleotide -id KM233118 --format gb > db/KM233118.gb  
  
# Get the 2014 Ebola genome as a FASTA file.  
efetch -db nucleotide -id KM233118 --format fasta > db/KM233118.fa  
  
# Extract all features of the GenBank file as separate sequences.  
cat db/KM233118.gb | extractfeat -filter -describe gene > db/KM233118-features.fa  
  
I often run statistics on files to ensure that they are what I expect them to  
be:  
  
seqkit stats db/KM233118-features.fa
```

produces:

```
file          format type num_seqs sum_len min_len avg_len max_len
db/KM233118-features.fa  FASTA  DNA      40  71,382      9 1,784.6 18,613
```

To see short and long help for the blast database builder run the following:

```
makeblastdb -h
makeblastdb -help
```

Construct the blast database with:

```
# Run the blast database builder.
makeblastdb -dbtype nucl -in db/KM233118-features.fa -out db/KM233118-features
```

70.2 How to run BLAST queries?

Now that we have the database let's run a query against it. Will make our query from the start region of the genome.

```
# Take the first sequence, keep the first 45 bases, rename the sequence to test, make it uppercase
cat db/KM233118-features.fa | seqret -filter -firstonly -sbegin 1 -send 45 -sid test -sup
```

```
# Look at the query file.
cat query.fa
```

produces:

```
>test [source] Zaire ebolavirus isolate Ebola virus/H.sapiens-wt/SLE/2014/Makona-NM042.
AATCATACTGGTTGTTTCAGAGCCATATCACCAAGATAGAGAA
```

Run the `blastn` tool

```
blastn -db db/KM233118-features -query query.fa
```

It generates a fairly lengthy output. The alignment specific section is:

```
> KM233118_1_2905 [mRNA] (gene="NP") Zaire ebolavirus isolate Ebola
virus/H.sapiens-wt/SLE/2014/Makona-NM042.3, complete genome.
Length=2905
```

```
Score = 84.2 bits (45),  Expect = 1e-19
Identities = 45/45 (100%), Gaps = 0/45 (0%)
Strand=Plus/Plus
```

```

Query 1  AATCATACCTGGTTGTTCAGAGCCATATCACCAAGATAGAGAA 45
         ||||||| | | | | | | | | | | | | | | | | | | | |
Sbjct  1  AATCATACCTGGTTGTTCAGAGCCATATCACCAAGATAGAGAA 45

```

70.3 How do I format the output differently?

The need to extract information on alignments is widespread. BLAST allows us to format the outputs into tabular and other formats. Change it to output format 6 or 7 (tabular form with or without comments and headers):

```
blastn -db db/KM233118-features -query query.fa -outfmt 7
```

it produces:

```

# BLASTN 2.7.1+
# Query: test [source] Zaire ebolavirus isolate Ebola virus/H.sapiens-wt/SLE/2014
# Database: db/KM233118-features
# Fields: query acc.ver, subject acc.ver, % identity, alignment length, mismatche
# 3 hits found
test  KM233118_1_2905    100.000 45 0 0 1 45 1 45 9.75e-20 84.2
test  KM233118_1_2905    100.000 45 0 0 1 45 1 45 9.75e-20 84.2
test  KM233118_1_18613   100.000 45 0 0 1 45 1 45 9.75e-20 84.2

```

We can even add custom fields to the output. Run `blastn -help`, then scroll through the output to find more details:

```

...
qseqid means Query Seq-id
qgi means Query GI
qacc means Query accession
qaccver means Query accession.version
qlen means Query sequence length
sseqid means Subject Seq-id
sallseqid means All subject Seq-id(s), separated by a ';'
sgi means Subject GI
sallgi means All subject GIs
...

```

The command now reads:

```
blastn -db db/KM233118-features -query query.fa -outfmt "6 qseqid sseqid pident"
```

That produces reduced information, query id, subject id, percent identity:

```
test    KM233118_1_2905 100.000
test    KM233118_1_2905 100.000
test    KM233118_1_18613 100.000
```

70.4 What are blast tasks?

Tasks are an algorithmic modification of the method (or its parameters) that make a specific blast tool better suited for finding certain types of alignments.

Confusingly, the task itself may be named the same as the tool. For example, `blastn` can have the following tasks:

- `blastn` - finds more divergent sequences
- `megablast` - finds less divergent sequences
- `blastn-short` - short queries

Even more confusingly, the default task for the `blastn` tool is the `megablast` task.

Let's make a shorter sequence than before, again; we will take the beginning of the genome:

```
# Take the first sequence record, make it upper case, keep the first 12 bases, rename the s
```

```
cat db/KM233118-features.fa | seqret -filter -firstonly -sbegin 1 -send 12 -sid test -sup
```

where the `short.fa` file is now:

```
>test [source] Zaire ebolavirus isolate Ebola virus/H.sapiens-wt/SLE/2014/Makona-NM042.
AATCATACCTGG
```

The search now is:

```
blastn -db db/KM233118-features -query short.fa
```

and it produces:

```
***** No hits found *****
```

We have to run `blastn` with a different search strategy in this case `blastn` and `blastn-short` produce a hit:

```
blastn -db db/KM233118-features -query short.fa -task blastn
```

Make the sequence even shorter at 7 bases like so:

```
echo ">mini" > mini.fa
echo "AATCATA" >> mini.fa
```

Now, only the `blastn-short` search strategy will produce results:

```
blastn -db db/KM233118-features -query mini.fa -task blastn-short
```

70.5 Will blast find all alignments?

You should know that the default operation of BLAST was tuned to the most common use cases. These common use cases can produce unexpected results that sometimes seem to defy common sense. Here is an example, let's take chromosome 1 of the yeast genome.

```
# The directory to store blast databases in.
mkdir -p db

# Fetch chromosome 1 of the yeast genome.
efetch -id NC_001133 -db nucleotide -format fasta > db/NC_001133.fa

# Make a blast database out of chromosome 1 of the yeast genome.
makeblastdb -dbtype nucl -in db/NC_001133.fa -out db/NC_001133
```

Create a query from the first two lines of chromosome 1 of the yeast genome (70 bases).

```
head -2 db/NC_001133.fa > start.fa
```

the file `start.fa` contains:

```
>NC_001133.9 Saccharomyces cerevisiae S288C chromosome I, complete sequence
CCACACCACACCCACACACACCACACACCACACACCACACACACACACATCTAACAA
```

Lets query the yeast genome for this sequence:

```
blastn -db db/NC_001133 -query start.fa
```

you get:

```
***** No hits found *****
```

Do a pairwise alignment with `blastn`:

```
blastn -subject db/NC_001133.fa -query start.fa
```

it returns:

```
***** No hits found *****
```

Now it is the time that desperation sets in! We know that the sequence is in the genome, we took it from the genome! Look:

```
cat db/NC_001133.fa | head -2
```

The start of the sequence is the same sequence as `start.fa`. Why won't BLAST find it???????

```
>NC_001133.9 Saccharomyces cerevisiae S288C chromosome I, complete sequence
CCACACCACACCCACACACCCACACACCACACCACACACCACACACACATCCTAAC
```



Let's do a local alignment with an optimal aligner as in `local-align.sh`:

```
local-align.sh start.fa db/NC_001133.fa
```

it produces:

NC_001133.9	1 CCACACCACACCCACACACCCACACACCACACCACACCACACCACACC	50
NC_001133.9	1 CCACACCACACCCACACACCCACACACCACACCACACCACACCACACC	50
NC_001133.9	51 CACACACACACATCCTAAC	70
NC_001133.9	51 CACACACACACATCCTAAC	70

YES! Finally some sanity in the world. This is the match we expected to see. So BLAST what's up?

Here is what happens, the query sequence has repetitive elements (also called low complexity regions) and BLAST automatically and silently filters out hits to these. Look at the sequence, see how little variation is there?

```
CCACACCACACCCACACACCCACACACCACACACCACACACCACACACACATCCTAAC
```

Since BLAST is tuned to help scientists with no computational background, and these scientists would continuously use and publish these low quality hits the developer of BLAST turned off presenting these. Was that a wise decision? We'll let you think about that, for now, let mention that to turn off this behavior you will need to set the flag `-dust no`

```
blastn -query start.fa -subject db/NC_001133.fa -dust no
```

it now produces the output:

```
Score = 130 bits (70), Expect = 7e-33
Identities = 70/70 (100%), Gaps = 0/70 (0%)
Strand=Plus/Plus
```

```
Query 1 CCACACCACACCCACACACCCACACACCACACACCACACACCACACACACA 60
       ||||||| | | | | | | | | | | | | | | | | | | | | | | | | | |
Sbjct 1 CCACACCACACCCACACACCCACACACCACACACCACACACCACACACACA 60
```

```
Query 61 CATCCTAAC 70
       ||||| | | |
Sbjct 61 CATCCTAAC 70
```

Let this be a reminder to all on how easy it is to end up with unintended consequences.

Chapter 71

BLAST databases

For BLAST to operate successfully, it needs to build a database of the target sequences. Different databases are necessary for nucleotide and peptide sequences.

- `makeblastdb` creates blast databases.
- `blastdbcmd` queries blast databases.
- `update_blastdb.pl` updates prebuilt blast databases.

71.1 How do we make a BLAST database?

Let's create a BLAST database. Get all proteins for project PRJNA257197:

```
# Make a directory to store the index
mkdir -p db

# Download the proteins from NCBI.
esearch -db protein -query PRJNA257197 | efetch -format fasta > db/proteins.fa

# Build the blast index.
makeblastdb -in db/proteins.fa -dbtype prot -out db/proteins -parse_seqids

# List the content of the blast database.
blastdbcmd -db db/proteins -entry 'all' -outfmt "%a" | head
```

71.2 Can we download BLAST databases?

While we can always create our own blast databases with `makeblastdb`, this task may take a considerable amount of time (days or even weeks) when the total number of bases to be indexed is very large.

To facilitate the process, prebuilt databases can be downloaded from the NCBI webpage.

```
ftp://ftp.ncbi.nlm.nih.gov/blast/db/
```

71.3 Can we automate the download of BLAST databases?

NCBI offers ready-made databases for many organisms and even the entire non-redundant database of all known sequences is available for download.

Make a dedicated storage for your blast databases:

```
# This directory will store all downloaded databases.  
# Note how it opens from your home directory.  
mkdir -p ~/blastdb  
  
# Switch to that folder.  
cd ~/blastdb  
  
# The blast package includes a script that can be used to download  
# ready made databases. List them all:  
update_blastdb.pl --showall  
  
# Download the 16 microbial database.  
update_blastdb.pl 16SMicrobial --decompress  
  
# Download the taxonomy database.  
# Required only if you wish to use taxonomy related information..  
update_blastdb.pl taxdb --decompress
```

We can also tell blast where to look for files via the `BLASTDB` variable. When you have the `BLASTDB` variable set then using blast tools becomes simpler,

you would not need add the full path to the BLAST database, specifying the name would be enough.

In addition, this is required for the taxonomy searches to work.

```
export BLASTDB=$BLASTDB:~/blastdb
```

If the above is set then instead of having to set the full path as in:

```
blastdbcmd -db ~/blastdb/16SMicrobial -info
```

BLAST can access the databases by name alone like so:

```
blastdbcmd -db 16SMicrobial -info
```

In addition, when the BLASTDB variable is set the taxonomy information can be also accessed by BLAST.

71.4 What information can be obtained from a BLAST database?

We can extract a wide variety of interesting information from blast databases. Sometimes it is worth creating a BLAST database just to extract specific information from sequences.

```
# Get some information on the 16S Microbial database.  
blastdbcmd -db ~/refs/refseq/16SMicrobial -info  
  
# What is the first sequence in the file  
blastdbcmd -db ~/refs/refseq/16SMicrobial -entry 'all' | head -1  
  
# Which publication links to this sequence?  
esearch -db nucore -query NR_118889.1 | elink -target pubmed | efetch
```

blastdbcmd is a command with high utility. Read the help on it:

```
blastdbcmd -help | more
```

71.5 How do I reformat sequences in a BLAST database?

The `-outfmt` parameter of the `blastdbcmd` allows printing a variety of information. For example, the following will print the sequence ids and their length:

```
blastdbcmd -db ~/refs/refseq/16SMicrobial -entry 'all' -outfmt '%a %l'
```

There are many more formatting options (see the detailed help for the tool) that can be used to answer potentially interesting questions:

We replace the ‘,’ (comma) separator with tabs since it is easier to manipulate files with tabs than with commas. The `tr` command translates each comma into a tab

```
blastdbcmd -db ~/refs/refseq/16SMicrobial -entry 'all' -outfmt '%a,%l,%T,%L' | tr
```

This file lists the genes, lengths, and common taxonomical ranks (this last part will only work if your taxonomy file has been downloaded as above and if the `BLASTDB` variable has been set properly).

NR_118889.1	1300	36819	Amycolatopsis azurea
NR_118899.1	1367	1658	Actinomyces bovis
NR_074334.1	1492	224325	Archaeoglobus fulgidus DSM 4304
NR_118873.1	1492	224325	Archaeoglobus fulgidus DSM 4304
NR_119237.1	1492	224325	Archaeoglobus fulgidus DSM 4304
NR_118890.1	1331	1816	Actinokineospora fastidiosa
NR_044838.1	1454	1381	Atopobium minutum
NR_118908.1	1343	1068978	Amycolatopsis methanolica 239
NR_118900.1	1376	1655	Actinomyces naeslundii
NR_044822.1	1255	2075	Pseudonocardia nitrificans

What organism has the longest 16S gene?

```
cat 16genes.txt | sort -k2,2 -rn | head
```

This sorts the file above by the second column:

NG_046384.1	3600	1104324	Pyrobaculum sp. 1860
NG_041958.1	2211	178306	Pyrobaculum aerophilum str. IM2
NG_041961.1	2207	121277	Pyrobaculum arsenaticum
NG_042068.1	2197	56636	Aeropyrum pernix

NG_042067.1	2192	70771	Pyrobaculum neutrophilum
NG_041957.1	2192	698757	Pyrobaculum oguniense TE7
NG_041951.1	2168	698757	Pyrobaculum oguniense TE7
NG_041954.1	2130	70771	Pyrobaculum neutrophilum
NG_044969.1	2089	477696	Thermogladius shockii
NG_042881.1	1833	215	Helicobacter fennelliae

71.6 How do I extract a specific entry from a database?

Pass the `-entry accession` parameter to `blastdbcmd`. Here, the particular blast database construction choices become essential, specifically whether or not the `-parse_seqids` was used. Accession numbers are only available in the latter case. Otherwise, the full sequence name needs to be used.

Get the first 20 bases of a specific 16S gene. We are asking for one specific accession number:

```
blastdbcmd -db ~/refs/refseq/16SMicrobial -entry 'NR_118889.1' -range 1-20
```

This produces the output:

```
>gi|645322056|ref|NR_118889.1|:1-20 Amycolatopsis azurea strain NRRL 11412 16S ribosomal RNA (16S rRNA) gene product
GGTCTNATAACCGGATATAAC
```

We can format the output for it as well.

```
blastdbcmd -db ~/refs/refseq/16SMicrobial -entry 'NR_118889.1' -range 1-20 -outfmt "%s"
```

This produces the following output:

```
GGTCTNATAACCGGATATAAC
```

71.7 How do I process all entries in a database?

Pass the `-entry all` parameter to `blastdbcmd` to get, for example, the first 50 bases of each gene.

```
blastdbcmd -db ~/refs/refseq/16SMicrobial -entry 'all' -range 1-50 -outfmt "%s"
cat starts.txt | head
```

This will produce a file containing the first 50 bases of each 16S gene, like so:

```
GGTCTNATACCGGATATAACAACCATGGCATGGTTGGTAGTGGAAAGCT
GGGTGAGTAACACGTGAGTAACCTGCCCNACTTCTGGATAACCGCTTG
ATTCTGGITGATCCTGCCAGAGGCCGCTGCTATCCGGCTGGGACTAAGCC
ATTCTGGTTGATCCTGCCAGAGGCCGCTGCTATCCGGCTGGGACTAAGCC
ATTCTGGTTGATCCTGCCAGAGGCCGCTGCTATCCGGCTGGGACTAAGCC
TACTTGGGATAAGCCTGGAAACTGGGTCTNATACCGGATATGACAAC
TTGAACCGGAGAGTTCGANCCTGGCTCAGGATGAACGCTGGCGCGCCT
GTGAGTGGCGAACGGGTGAGTAACACGTGGTAACCTCNNTGTACTTG
GTGAGTAACCTGCCCTTCTTCTGGATAACCGCATGAAAGTGTGGCTAAT
ACCGGATANGACCACNTGCATNTCGGTGGGTGGAAAGTTTTTCGGT
```

How many unique gene starts of length 50 are there?

```
cat starts.txt | sort | uniq -c | sort -rn | head
```

The output of the above is:

```
298 AGAGTTGATCCTGGCTCAGGACGAACGCTGGCGCGTGCTAACACATG
161 AGAGTTGATCCTGGCTCAGGACGAACGCTGGCGCGTGCTAACACATG
126 GACGAACGCTGGCGCGTGCTAACACATGCAAGTCGAGCGGTAAAGGCC
102 ATTCCGGTTGATCCTGCCGGAGGCCATTGCTATGGAGTCCGATTAGCC
101 AGAGTTGATCATGGCTCAGATTGAACGCTGGCGGCAGGCCTAACACATG
 91 GACGAACGCTGGCGCGTGCTAACACATGCAAGTCGAGCGGAAAGGCC
 84 AGAGTTGATCCTGGCTCAGGACGAACGCTGGCGCGTGCTAACACATG
 75 AGAGTTGATCCTGGCTCAGAACGAACGCTGGCGGCAGGCCTAACACATG
 72 TAGAGTTGATCCTGGCTCAGGACGAACGCTGGCGCGTGCTAACACAT
 69 AGAGTTGATCCTGGCTCAGATTGAACGCTGGCGGCAGGCCTAACACATG
```

The above result shows that out of all 16S genes in the NCBI database 298 have the same 50 basepairs as starting sequences.

Part XVII

SHORT READ ALIGNMENT

Chapter 72

Short read aligners

Sequencing used to be a costly proposition. The resulting few, and relatively long, sequences (1000 bp) were very “precious” and were meant to be studied extensively. For that reason, aligners were initially designed to perform near-optimal alignments and to enumerate most if not all alternative arrangements that all fell within a level tolerance. The landscape changed dramatically around 2005 with the arrival of the so-called high throughput short-read technologies.

72.1 How are short reads different from long reads?

Modern sequencing instruments brought about two significant changes:

1. The read lengths became very short (50-300bp)
2. The number of reads grew extremely high (hundreds of millions)

Hence, whereas initial algorithms were designed to find most (if not all) alignments, scientists needed newer and faster methods that could very quickly select the best location for each of the measurements at the expense of not investigating all potential alternative alignments.

The rationale behind the new paradigm was to treat sequencing reads as small, independent measurements to be subsequently matched either against

a known genome (called re-sequencing) or against one another (called assembly). This new use case and its requirements lead to the development of an entire subfield of alignment algorithms, often referred to as “short read aligners” or “short read mappers.”

72.2 What are short read aligners?

Short read aligners software tools designed to align a vast number of short reads (billions, or lately even trillions¹). The dominance of the Illumina instrumentation means that most short-read software were designed to work best for sequences of lengths in the range of 50 to 300 bp that the Illumina instruments produce.

72.3 How do short read aligners work?

First, let us make something abundantly clear: short read aligners are marvels of modern scientific software engineering. A well-optimized short read aligner can match over ten thousand sequences per second (!) against the 3 billion bases of the human genome. Their performance, implementation, and level of execution is nothing short of astounding.

But there is another side to the story. The rational expectation is that the published aligners would produce reasonably similar alignments and that the difference between various implementation would manifest primarily in their performance, hardware requirements, and small fluctuations in accuracy and precision.

In reality, the results that two aligners produce can be substantially different. Surprisingly, it seems almost impossible to unambiguously understand, describe, and characterize the changes in the results that different algorithm’s implementations provide. While there is no shortage of scientific publications that claim to compare the accuracy and performance of various tools, most of these papers fail to capture the essential differences between the methods. It is not for lack of trying - accurately characterizing the “quality” of alignments turned out to be a lot more complicated than anticipated.

¹<https://www.nature.com/articles/s41588-018-0273-y>

A typical bioinformatics analysis requires finding exact matches, missing locations, and partial and alternative matches/overlaps. An method that is efficient at one of these tasks trades that performance gain for inefficiency in another requirement.

Hence, the performance of software methods depends critically on the type of data they are used to analyze, which, in turn, is domain-specific. Different alignment methods will, therefore, be best suited to different domains.

72.4 What are the limitations of short read aligners?

1. Most short read aligners will find only alignments that are reasonably similar to the target. The algorithm will generally “gives up” searching beyond a certain threshold.
2. Short read aligners are designed to find regions of high similarity.
3. Most short read aligners typically cannot handle long reads or become inefficient when doing so.
4. There is also a limit to how short a read can be.

The minimum length for the read is algorithm- and implementation-dependent; it is rarely (if ever) mentioned in the documentation. Many tools stop working correctly when the read lengths drop under 30 bases. When studying small RNAs, for example, microRNAs, we have to look for tools that can align very short reads; the selection is surprisingly sparse.

72.5 What is read mapping?

Conceptually, alignment and mapping appear to mean the same thing, but there are subtle differences. The word “mapper” is often used to emphasize that the optimal alignment of a read is not guaranteed. The purpose of a mapper tool is locating a region in a genome, not producing an optimal alignment to that region.

The following definition is taken from a presentation by Heng Li:

Mapping

- A mapping is a region where a read sequence is placed.
- A mapping is regarded to be correct if it overlaps the true region.

Alignment

- An alignment is the detailed placement of each base in a read.
- An alignment is regarded to be correct if each base is placed correctly.

Interestingly, and we invite the reader to come up with their examples, we could have a situation where a read has a correct mapping with an incorrect alignment or vice versa - has an incorrect mapping with a proper alignment.

Ideally, of course, the mapping and the alignment should coincide — but it's important to remember that this is not always the case.

72.6 How do I distinguish between a mapper and an aligner?

It used to be that tools could be readily categorized into the two classes of aligner vs. mapper. As the field advanced and matured, however, the read lengths became longer, and the software tools started to make use of combined techniques that made tools behave as both mappers and aligners.

It is essential, however, to remember that the distinction between mapping and alignment does exist, and to recognize further that different studies have different requirements in regards to the use of these concepts.

For example, studies examining SNPs and variations in a genome would be primarily alignment-oriented. By contrast, studies using ChIP-Seq data would be essentially mapping-oriented.

72.7 How do we pick the best short read aligner?

So how do we pick the best aligner? There are good starting choices, such as `bwa` and `bowtie2`, that will perform well in all domains of application. Later, you may want to explore other tools based on observed performance, reviews of scientific literature, and the experience of other scientists studying the same type of data. The optimal choice will be specific to the data you are analyzing.

In the end, the choice of the aligner is a bit “faith-based,” coupled to personal observations and empirical evidence. There is also a cultural and historical element to it. Scientists at the Broad Institute will likely use `bwa` because it was developed there; others who work at the Beijing Genomics Institute will probably prefer the `novoalign` tool because as it was created at that institute, and so on.

Then there is an individualistic feel to almost every high-throughput aligner. Typically, a sole genius is behind each, an individual with uncommon and extraordinary programming skill, a person that has set out to solve a problem that is important to them. Due to the sheer complexity of the task and requirements, the software’s unique characteristics will reflect their developer personality, virtues, and some of their quirks, as well.

You don’t just run an aligner; you perform a Vulcan style “mind-meld²” (see Star Trek) with the author who wrote it.

72.8 What features do we look for in a short read aligner?

When choosing an aligner, you will need to perform a mental checklist of what features the aligner of your choice should provide:

- Alignment algorithm: global, local, or semi-global?
- Can the aligner filter alignments based on external parameters?
- Can aligner report more than one alignment per query?

²[https://en.wikipedia.org/wiki/Vulcan_\(Star_Trek\)#Mind_melds](https://en.wikipedia.org/wiki/Vulcan_(Star_Trek)#Mind_melds)

- How will the aligner handle INDELS (insertions/deletions)?
- Can the aligner skip (or splice) over intronic regions?
- Will the aligner find chimeric alignments?

Looking at this list above, it is probably clear what makes comparing different tools so tricky. Various techniques excel for different requirements; moreover, as we shall see, we are only scratching the surface when it comes to the information that alignment record may contain.

Chapter 73

The `bwa` aligner

The `bwa` (aka Burrows-Wheelers Aligner) aligner was written by Heng Li¹, a research scientist at the Broad Institute. The `bwa` aligner is possibly the most widely used short read alignment tool and is well suited for a broad number of applications.

- Github page: <https://github.com/lh3/bwa>

Interestingly, the algorithm that the latest version of `bwa` implements has not yet been published in a peer-reviewed journal (as of 2016). There is a fascinating story of why that is so², and it illustrates the shortcomings of the “scientific method” itself. The new algorithm is called the `bwa mem` algorithm (where `mem` stands for Maximally Exact Matches).

But note that there may be quite a bit of old documentation on the previous alignment method called `bwa aln`. In the following, we will focus solely on the `bwa mem` algorithm, the results of which will differ (often substantially and are much improved) compared to results that of the `bwa aln` algorithm. See the pitfalls of comparing tools? Even the same software may later choose to implement different algorithms that may produce very different results.

¹<http://lh3lh3.users.sourceforge.net/>

²<https://gist.github.com/ialbert/3164967c853b7fd8f44e>

73.1 How does bwa work?

In general, all short read aligners operate on the same principles:

1. First build an “index” from a reference genome (this only needs to be done once).
2. The reads in FASTA/FASTQ files are then aligned against the index created in step 1.

Index building consists of pre-processing the reference genome so that the program can search it efficiently. Each program will build a different type of index; sometimes it may produce multiple files with odd-looking names or extensions. For this reason, it is best to place the reference genome in separate directories.

The time and computational resources required for building an index will depend on the genome size. These times can be substantial (many days or more) for large genomes. For some software it is possible to download pre-built indices from their websites.

73.2 How do I use bwa?

Suppose you were interested in processing the sequencing data from Redo: Genomic surveillance elucidates Ebola virus origin (#ebolaredo). And that you wished to investigate the changes relative to the 1976 outbreak.

Make a new director for the reference genomes:

```
mkdir -p refs
```

Get the ebola genome in FASTA format.

```
efetch -db=nuccore -format=fasta -id=AF086833 > ~/refs/AF086833.fa
```

Build an index with bwa:

```
bwa index ~/refs/AF086833.fa
```

The indexing above will finish rather quickly, as the genome is a mere 18K base long. Note that in scripts you should make use of environment variables to store names. It will allow you to keep the command the same and only change the value of the variable. For example, we could have written,

```
# Assign the file name to a variable  
ACC=AF086833.fa  
  
# The reference genome stored locally.  
REF=refs/$ACC.fa  
  
# Fetch the sequence.  
efetch -db=nuccore -format=fasta -id=$ACC > $REF.fa  
  
# Build an index for the genome so that we can view in IGV  
samtools faidx $REF  
  
# Build the index.  
bwa index $REF
```

The `bwa` program provides a neat self-discovery of its parameters. It is generally run as `bwa command`, and it will perform very different tasks depending on which command is used. To discover all of the `bwa` can run execute it on its own like so:

```
bwa
```

Then we can find out more on each command, like so:

```
bwa index  
bwa mem
```

Note the instructions that it provides. To align reads, we need sequence data files. Let's obtain data from SRA. Let's find out all runs for the Ebola project.

```
esearch -db sra -query PRJNA257197 | efetch -format runinfo > runinfo.csv
```

Pick a run from this file, say `SRR1972739`, and we'll only subset the data to 10K reads to get through quickly:

```
fastq-dump -X 10000 --split-files SRR1972739
```

We can create shortcuts to each of our files. This will allow us to shorten and simplify the commands greatly.

```
R1=SRR1972739_1.fastq  
R2=SRR1972739_2.fastq
```

You can always check what the variable stands for with,

```
echo $R1
```

To align one of the read pairs with `bwa mem` in single end mode, we could run the command with:

```
bwa mem $REF $R1 > output.sam
```

The resulting file is in a so-called SAM (Sequence Alignment Map) format, one that you will undoubtedly love and hate (all at the same time). It is one of the most recent bioinformatics data formats, one that by today has become the standard method to store and represent all high-throughput sequencing results. The SAM file looks like this:

```
@SQ SN:gi|10141003|gb|AF086833.2| LN:18959
@PG ID:bwa PN:bwa VN:0.7.12-r1039 CL:bwa mem /Users/ialbert/refs/ebola/1976.fa
SRR1972739.1 83 gi|10141003|gb|AF086833.2| 15684 60 69M32S = 15600 -153
```

A SAM file encompasses all known information about the sample and its alignment; typically, we never look at the FastQ file again, since the SAM format contains all (well almost all) information that was also present in the FastQ measurements.

Since this is a paired-end dataset, the proper way to align the data is in paired-end mode. For that purpose, we list both files on the command line:

```
bwa mem $REF $R1 $R2 > bwa.sam
```

Note how the same data can be aligned in either single end or paired end mode. This latter alignment will contain more information as pair related information will be added to the alignment file.

73.3 How do I find help on bwa?

To see all the ways that we could tune `bwa mem` alignment run

```
bwa mem
```

Among the many options, note those that set the scoring matrix:

```
-A INT      score for a sequence match, [...] [1]
-B INT      penalty for a mismatch [4]
-O INT[,INT] gap open penalties for deletions and insertions [6,6]
```

```
-E INT[,INT] gap extension penalty; a gap of size k cost '{-0} + {-E}*k' [1,1]
-L INT[,INT] penalty for 5'- and 3'-end clipping [5,5]
-U INT       penalty for an unpaired read pair [17]
...
```

When compared to the more traditional EDNAFULL matrix discussed in the alignment chapter, we note that in the default setting, the reward for sequence match is much smaller; hence, the default alignment is tuned to find long stretches if bases match.

Gap open penalties are also smaller (6 instead of 10) whereas gap extension penalties are a bit larger.

There are other parameters where a single word will set multiple values. For example:

- note how `-x ont2d` is equivalent to setting `-k14 -W20 -r10 -A1 -B1 -O1 -E1 -L0`. These are the recommended settings when aligning data from an Oxford Nanopore MinION instrument.

The way the parameters alter the final alignment gets complicated. Here is where the choice becomes more "faith-based". We place of faith in the author of the tool. In this case, we trust that Heng Li knows why these parameters are the best, even if we could not fully explain their interplay.

What is essential to keep in mind is that decisions may have been made on our behalf and if things don't go the way we expect, we know where to look for potential solutions.

Chapter 74

The bowtie aligner

The first version of bowtie aligner¹ was the first implementation of the Burrows-Wheeler algorithm for short read alignment, and with that it has opened a new era in processing high-throughput data.

The latest version of the algorithm `bowtie2`, is almost always preferred. In this book when we talk about the bowtie program we mean bowtie version 2.

74.1 How does bowtie work?

In general, all short read aligners operate on the same principles:

1. First build an “index” from a reference genome (this only needs to be done once).
2. The reads in FASTA/FASTQ files are then aligned against the index created in step 1.

Index building consists of pre-processing the reference genome so that the program can search it efficiently. Each program will build a different type of index; sometimes it may produce multiple files with odd-looking names or extensions. For this reason, it is best to place the reference genome in separate directories.

¹<http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>

74.2 How do I build an index with bowtie?

We will align ebola sequencing data against the 1976 Mayinga reference genome. We recommend keeping reference genomes in a more general location rather than the local folder, since you will reuse them frequently. For example, use `~/refs/ebola` for all ebola genomes and annotations.

This directory will hold the reference genome and all indices:

```
mkdir -p refs
```

```
# Assign the file name to a variable  
ACC=AF086833.fa  
  
# The reference genome stored locally.  
REF=~/refs/$ACC.fa  
  
# Fetch the sequence.  
efetch -db=nuccore -format=fasta -id=$ACC > $REF.fa  
  
# Build an index for the genome so that we can view in IGV  
samtools faidx $REF
```

For bowtie2, we need to specify both the input and output prefix name on the command line. Confusingly, these can also be the same because many prefixes are the same. The first `$REF` is the reference file; the second `$REF` indicates the prefix name of the index; it could be anything, but it is a good habit to keep it the same or similar to the actual reference.

```
bowtie2-build $REF $REF
```

See for yourself how the ancillary files multiply like rabbits.

```
ls ref/*
```

74.3 How do I align with bowtie?

We can pick an ebola sequencing run id, for example, `SRR1972739`, and we'll select just 10K reads to get through quickly:

```
fastq-dump -X 10000 --split-files SRR1972739
```

We can create shortcuts to each of our files. This will allow us to shorten and simplify the commands greatly.

```
R1=SRR1972739_1.fastq
R2=SRR1972739_2.fastq
```

To align with bowtie,

```
bowtie2 -x $REF -1 $R1 -2 $R2 > bowtie.sam
```

When bowtie runs, it also gives us a report:

```
10000 reads; of these:
 10000 (100.00%) were paired; of these:
   5086 (50.86%) aligned concordantly 0 times
   4914 (49.14%) aligned concordantly exactly 1 time
   0 (0.00%) aligned concordantly >1 times
   ----
   5086 pairs aligned concordantly 0 times; of these:
     827 (16.26%) aligned discordantly 1 time
     ----
 4259 pairs aligned 0 times concordantly or discordantly; of these:
  8518 mates make up the pairs; of these:
    7463 (87.61%) aligned 0 times
    1055 (12.39%) aligned exactly 1 time
    0 (0.00%) aligned >1 times
62.69% overall alignment rate
```

Not to be outdone in the shock and awe department, `bowtie2`, too, can be set up in myriad ways.

```
bowtie2
```

Run and enjoy the view:

```
[ ... lots of lines ... ]
Presets:           Same as:
For --end-to-end:
--very-fast        -D 5 -R 1 -N 0 -L 22 -i S,0,2.50
--fast              -D 10 -R 2 -N 0 -L 22 -i S,0,2.50
--sensitive        -D 15 -R 2 -N 0 -L 22 -i S,1,1.15 (default)
--very-sensitive   -D 20 -R 3 -N 0 -L 20 -i S,1,0.50
```

For --local:

```
--very-fast-local      -D 5 -R 1 -N 0 -L 25 -i S,1,2.00
--fast-local          -D 10 -R 2 -N 0 -L 22 -i S,1,1.75
--sensitive-local     -D 15 -R 2 -N 0 -L 20 -i S,1,0.75 (default)
--very-sensitive-local -D 20 -R 3 -N 0 -L 20 -i S,1,0.50
```

[... lots of lines ...]

Chapter 75

How do I compare aligners?

75.1 How can I tell which aligner is “better”?

The answer to this is a little more complicated. As you will see, we can't quite make it a “fair” comparison to begin with. Let's try to evaluate the alignment on simulated sequences:

```
# Get the reference genome and build an index from it.  
mkdir -p refs  
REF=refs/AF086833.fa  
efetch -db=nuccore -format=fasta -id=AF086833 > $REF  
  
# Index the reference with bwa  
bwa index $REF  
# Index the reference with bowtie2.  
bowtie2-build $REF $REF  
  
# Simulate 100,000 reads from the genome with 1% error rate  
dwgsim -N 100000 -e 0.01 -E 0.01 $REF data  
  
# Set up shortcuts to the reads.  
R1=data.bwa.read1.fastq  
R2=data.bwa.read2.fastq
```

Align with `bwa mem` while timing the run:

```
time bwa mem $REF $R1 $R2 > bwa.sam
```

Align with `bowtie2`:

```
time bowtie2 -x $REF -1 $R1 -2 $R2 > bowtie.sam
```

The result is that:

- `bwa` aligns 95.4% of reads in 4 seconds
- `bowtie2` aligns 94.7% of reads in 10 seconds.

So `bowtie2` was slower and a hair less efficient. We’ll call it a tie, leaning towards `bwa`

Now let’s raise the error rate to 10%:

```
dwgsm -N 100000 -e 0.1 -E 0.1 $REF data
```

Running the same alignments now take longer and fewer reads align:

- `bwa` aligns 83.3% of reads in 6 seconds
- `bowtie2` aligns 28.9% of reads in 3 seconds.

The discrepancy is now huge. `bowtie2` is way off the mark! The first lesson to learn here is that properties of the input data may have major effects on both runtime and mapping rates that you may observe. But we can tune `bowtie2` and make it more sensitive by using the ... drumroll ... `--very-sensitive-local` flag:

```
bowtie2 --very-sensitive-local -x $REF -1 $R1 -2 $R2 > bowtie.sam
```

In this case

- `bwa` aligns 83.3% of reads in 6 seconds
- `bowtie2` aligns 63.86% of reads in 11 seconds.

Maybe we could tune `bowtie2` even more. Hmm ... how about this for tuning:

```
time bowtie2 -D 20 -R 3 -N 1 -L 20 -x $REF -1 $R1 -2 $R2 > bowtie.sam
```

We are getting somewhere; `bowtie2` now aligns more reads than `bwa`:

- `bwa` aligns 83.3% of reads in 6 seconds
- `bowtie2` aligns 87.14% of reads in 10 seconds.

Thus as you can see, it is not that simple to compare aligners. A lot can depend on how aligners are configured - though all things considered we like to start with and recommend `bwa` as apparently it is a lot less finicky and gets the job done without adding elaborate decorations.

But does it mean that `bowtie2` is a second class citizen? Well, let's not rush to judgment yet. Let's see what tags are present in the `bwa` alignment file:

```
cat bwa.sam | cut -f 12-20 | head
```

prints:

```
AS:i:0 XS:i:0
AS:i:0 XS:i:0
NM:i:6 MD:Z:2A12G5A14G6G23A2 AS:i:44 XS:i:0
NM:i:6 MD:Z:7G8G32A8T0T3A6 AS:i:43 XS:i:0
```

the alignments made with `bowtie2`

```
cat bowtie.sam | cut -f 12-20 | head
```

contain:

```
AS:i:-29      XN:i:0 XM:i:10 X0:i:0 XG:i:0 NM:i:10 MD:Z:OC0A6A17C4T7T1A6G1T13A
AS:i:-27      XN:i:0 XM:i:10 X0:i:0 XG:i:0 NM:i:10 MD:Z:2C8A8C5T1T5A6T14T3C2T6
AS:i:-18      XN:i:0 XM:i:6 X0:i:0 XG:i:0 NM:i:6 MD:Z:2A12G5A14G6G23A2 YS:i:-
AS:i:-16      XN:i:0 XM:i:6 X0:i:0 XG:i:0 NM:i:6 MD:Z:7G8G32A8T0T3A6 YS:i:-
```

Do you see the difference? Bowtie2 fills in a lot more information; it also offers more ways to format the output and filter the alignments. So it has its applications and may be indispensable in some situations.

By the way, do you want to know how we figured out that it should be `-D 20 -R 3 -N 1 -L 20` and what does that even mean?

The task of finding settings where for this simulation `bowtie2` outperforms `bwa` was a homework assignment.

The help for `bowtie2` lists the parameters that are set when `--very-sensitive-local` is set, so students started with those and kept tweaking and re-running the alignments. It was a brute force search, not a particularly deep insight into what these parameters mean and how they interact. Few people (if anyone) understands these at that level of granularity.

75.2 How do I choose the right aligner?

The most important lesson is not to treat the tools as “better” or “worse,” but instead as more or less appropriate for a given task. Also, it is essential

to understand that not all tools operate at the same levels of sensitivity or specificity when using them on their default setting.

Finally note that, depending on the data and its characteristics, different tools may be better suited to different types of data. The requirement to have a particular kind of information in the BAM file may also be a significant factor. The good news is that it is easy to benchmark different aligners as needed. As we've seen conceptually, the tools are very similar.

Chapter 76

Multiple sequence alignments

A multiple sequence alignment is one that uses three or more sequences. Since you already saw that even pairwise alignments can be substantially customized it should come at no surprise that since multiple sequence alignment involves reconciling differences across more than two sequences there are several different ways to go about it.

Just as with pairwise alignment there are optimal and non-optimal but much speedier methods to perform the same analysis.

76.1 How do I align more than two sequences?

Suppose we wanted to align several full length ebola viruses.

```
# Store the genomes in this folder.  
mkdir -p genomes
```

```
# Get all genomes for the ebola project.
```

```
esearch -db nuccore -query PRJNA257197 | efetch -format fasta > genomes/ebola.fa
```

The file contains 249 full length genomes:

```
seqkit stat genomes/ebola.fa
```

that produces:

```
file      format type num_seqs  sum_len min_len avg_len max_len
genomes/ebola.fa FASTA  DNA    249 4,705,429 18,613 18,897.3 18,959
```

Initially let's just align the first ten full ebola sequences against one another.

Find the ID of the first ten sequences:

```
seqkit seq -n genomes/ebola.fa | cut -f 1 -d ' ' | head -10 > ids.txt
```

The `ids.txt` file now contains ten accession numbers:

```
KR105345.1
KR105328.1
KR105323.1
...
...
```

Using these accession numbers we can extract the sequences that correspond to these ids:

```
seqkit grep --pattern-file ids.txt  genomes/ebola.fa > small.fa
```

Perform a multiple sequence alignment with the `mafft` tool:

```
mafft --clustalout small.fa > alignment.maf
```

View the alignment:

```
head -20 alignment.maf
```

Displays a visual alignment of all sequences

```
KR105345.1      -----ataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105328.1      --gattaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105323.1      --gattaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105302.1      --gattaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105295.1      ---attaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105294.1      --gattaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105282.1      --gattaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105266.1      ---attaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105263.1      aagattaataatttcctctcattgaaatttatatcgaaatttaattgaaattgttact
KR105253.1      ---attattaatyttcctctcattgaaatttatatcgaaatttaattgaaattgttact
                                         **** ****
```

We can see that one difference between the genomes is how complete the assemblies are at their edges.

```
tail -20 alignment.maf
```

Similarly towards the ends:

KR105345.1	ggaaaaatggcacacacaaaattaaaaataaatctattcttcttttgtgtgt
KR105328.1	ggaaaaatggtcgcacacaaaattaaaaataaatctattcttcttttgtgtgt
KR105323.1	ggaaaaatggtcgcacacaaaattaaaaataaatctattcttcttttgtgtgt
KR105302.1	ggaaaaatggtcgcacacaaaattaaaaataaatctattcttcttttgtgtgt
KR105295.1	ggaaaaatggtcgcacacaaaattaaaaataaatctattt-----
KR105294.1	ggaaaaatggtcgcacacaaaattaaaaataaatctattcttcttttgtgtgt
KR105282.1	ggaaaaatggtcgcacacaaaattaaaaataaatctattcttcttttgtgtgt
KR105266.1	ggaaaaatgg-----
KR105263.1	ggaaaaatggtcgcacac-----
KR105253.1	ggaaaaatggtcgcacacaaaattaaaaataatctattcttctt-----

The * character indicates a consensus.

Play around and perform multiple sequence alignments on 10% of the genomes:

```
seqkit sample --proportion 0.1 genomes/ebola.fa > small.fa
mafft --clustalout small.fa > alignment.maf
```

76.2 What programs can be used to align multiple sequences?

Tools include `clustal-omega`, `mafft`

(TODO: COMING-SOON)

Part XVIII

SAM/BAM Format

Chapter 77

The SAM/BAM/CRAM formats

77.1 What is a SAM file?

The SAM file contains information on alignments.

The SAM format is a TAB-delimited, line-oriented text format consisting of a

1. **Header** section, where each line contains some metadata
2. **Alignment** section where each line provides information on an alignment

The SAM format¹ specification lists the required and optional content for each of these sections.

In general, the quality of the information within a SAM file determines the success of analysis. Hence, producing this file so that it contains the information we need, and the ability to investigate the data on our own should always be our most pressing concern.

Note: Understanding the details of SAM will require that you occasionally consult the official documentation:

- The SAM format specification² is the specification of the SAM format.

¹<http://samtools.github.io/hts-specs/SAMv1.pdf>

²<http://samtools.github.io/hts-specs/SAMv1.pdf>

- The SAM tag specification³ is the specification of the SAM tags.
- Published as The Sequence Alignment/Map format and SAMtools⁴ in Bioinformatics. 2009

77.2 What is a BAM file?

A BAM file is a binary, compressed (and almost always sorted) representation of the SAM information. Generally, BAM files are sorted, usually by the alignment coordinate and more rarely by the read names.

- Sorting by coordinate allows fast query on the information by location.
- Sorting by read name is required when the identically named read pairs need to be accessed quickly as read pairs will then be stored in adjacent lines.

77.3 What is the CRAM format?

The file format was designed and released in 2014 by the European Bioinformatics Institute see CRAM goes mainline⁵

CRAM files are conceptually similar to BAM files. CRAM files represent a more efficient version of the information in BAM files, where the gain is made by storing the reference separately. The essential difference between BAM and CRAM format is that a CRAM file can only be read if the reference sequence is present at a given location on the target computer. In contrast, the information in the BAM file is complete.

The CRAM format is supported directly by the `samtools sort`:

```
bwa mem $REF $R1 $R2 | samtools sort --reference $REF -O CRAM > bwa.cram
```

The downside of this approach is that losing the reference sequence means losing the data (that is, being unable to decode it). For many use-cases this is not a problem. But there are situations where having the alignment

³<http://samtools.github.io/hts-specs/SAMtags.pdf>

⁴<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2723002/>

⁵<http://genomeinformatician.blogspot.com/2014/08/cram-goes-mainline.html>

data dependent on the presence and availability of the reference genome adds another layer of complexity.

77.4 Will the CRAM format replace BAM?

Time will tell (probably not).

77.5 Is it SAM, BAM or CRAM now?

The format that the data is in is called SAM. When we store it in a compressed binary form, we call it BAM or CRAM. Most of the time we exchange the data in a binary format because it is much more efficient and faster to query.

Note that when we view or query a BAM file with a tool like `samtools` we see the results converted into the SAM format.

We will usually call the format and the data as SAM with the understanding that the representation may be either that SAM, BAM or CRAM. Each format contains the same information; the differences are in their storage methods and performance characteristics.

77.6 What are SAM files used for?

SAM files were designed for two primary use cases:

1. To store alignments in a standardized and efficient manner.
2. To allow quick access to alignments by their coordinates.

For example, if you had 100 million alignments in a file and wanted those alignments that overlap with coordinate 1,200,506 - the BAM format can return that information in milliseconds, without having to read through the entire file.

77.7 Is the SAM an optimal format?

Unfortunately no.

The SAM format has all the features of a product designed by a committee (which it was), marred by premature optimizations (the root of all evil according to Donald Knuth), crippled by flawed conceptual design, impaired by an incomplete specification. Other than that it is fantastic.

But we are stuck with SAM. Like it or not, we have to learn how to deal with the format.

77.8 What information is stored in a SAM file?

Even practicing bioinformaticians frequently have misconceptions about SAM files. Among them, the most prevalent is the assumption that BAM files produced by different tools will have a comparable amount of information and the differences are mostly in the *accuracy* or *performance* characteristics of the alignments.

Nothing could be further from the truth.

The misconception perhaps caused by the SAM specification⁶ itself. This specification appears to be the “standard” requirement that seems to prescribe what exactly should go into each SAM file:

1. each SAM file has to have 11 columns
2. each column contains the following information⁷ ...

Yet, as it turns out the “required” information is quite limited, and a file that would only fill the required columns would be useless in many cases. Most often the “optional” tags and other fields of the SAM file carry all the information needed for analysis. Thus there may be substantial differences between alignment information produced by different tools

The take-home message is that a SAM format is still somewhat nebulous. It prescribes the presence of very little information. Each aligner will fill in as

⁶<http://samtools.github.io/hts-specs/SAMv1.pdf>

⁷<http://samtools.github.io/hts-specs/SAMv1.pdf>

much as it knows how to.

77.9 Why do we need the SAM format?

The SAM format was introduced to support use cases presented by the high throughput sequencing instrumentation:

- Fast access to alignments that overlap with a coordinate. For example, select alignments that overlap with coordinate 323,567,334 on chromosome 2
- Quick selection and filtering of reads based on attributes. For example, we want to be able to quickly select alignments that align on the reverse strand (though we want to warn you that the actual implementation of the filtering borderlines the absurd)
- Efficient storage and distribution of the data. For example, have a single compressed file containing data for all samples each labeled in some manner.

77.10 How important is to understand the details of the SAM format?

On the one hand, it is conceivable that most of the time you will not need to manipulate SAM/BAM files directly. There is an extensive library of programs and tools designed to process and analyze SAM/BAM files for you.

On the other hand, many problems cannot be understood and solved without understanding and investigating the SAM files in more detail. Time and again we find ourselves looking at a few lines of a SAM file to understand a seemingly mysterious event taking place.

77.11 How do I create SAM/BAM files?

We have a separate chapter titled How to make a BAM file, below we point out only the relevant lines: Typically you would generate a SAM file, then

sort that file and convert it into a BAM format. Finally, you will need to index the resulting BAM file.

```
# Create a SAM file.
bwa mem $REF $R1 $R2 > alignments.sam

# Convert SAM to sorted BAM.
samtools sort alignments.sam > alignments.bam

# Index the BAM file.
samtools index myfile.bam
```

The `samtools` package has come a long way in its usability and since version 1.3 will both convert and sort a SAM file into BAM in one step:

```
# Create a sorted BAM file in one line.
bwa mem $REF $R1 $R2 | samtools sort > alignments.bam

# Index the BAM file.
samtools index alignments.bam
```

77.12 How do I create a CRAM file?

You need to add the reference to the sorting step and change the output format:

```
# Create a sorted CRAM file in one line.
bwa mem $REF $R1 $R2 | samtools sort --reference $REF -O cram > bwa.cram

# Index CRAM file.
samtools index bwa.cram
```

77.13 Can “unaligned” data be stored in a SAM file?

Because the BAM files are compact and can store additional information they have become a de-facto standard not only for representing and storing

alignment information but also for storing raw “unaligned” FASTQ files.

It is a somewhat idiosyncratic usage (welcome to bioinformatics folks): a format that was initially designed to store alignments is now used to represent “unaligned” data. In all honesty that is more a testament of the shortcomings of the FASTQ files.

The advantage of the BAM file is that each read can be tagged with additional, user-created information, for example, the sample name.

When used in this manner a single BAM file can be used to store all samples from an experiment, hence greatly simplifying file management. Instead of having to deal with 100 paired-end samples distributed over 200 files we can save a single BAM file.

Here is an example of converting paired-end FASTQ files in an unaligned BAM format, where each read in the pair is tagged as “FOO”

```
# Get the data.  
fastq-dump -X 10000 --split-files SRR1972739
```

```
# Set up shortcuts.  
R1=SRR1972739_1.fastq  
R2=SRR1972739_2.fastq
```

```
# Convert FASTQ to BAM file. Tag each sample as "FOO"  
picard FastqToSam F1=$R1 F2=$R2 O=SRR1972739.bam SM="FOO"
```

Look at the headers of the resulting file:

```
 samtools view -H SRR1972739.bam
@HD VN:1.5  SO:queryname
@RG ID:A    SM:FOO
```

The so-called “readgroup” A will be tagged as “FOO”:

```
 samtools view SRR1972739.bam | head -1
```

And the file contains (shown wrapped below):

SRR1972739.1 77 * 0 0 * * 0 0
TCTAAAAGGAGGACTCAATTGCAATATCTCATTGAGTCCTCCTT
CCCCFFFFFHHHHFIJJJJJJJJJJJJJJJJJJJJGIGIJJJJJ
RG:Z:A

Note again the RG:Z:A tag at the end of the read that maps the read to id A that in turn is mapped to sample SM:FOO in the header.

To analyze the data stored in an unaligned BAM file we typically need to unpack it into FASTQ format again like so:

```
 samtools fastq SRR1972739.bam -1 pair1.fq -2 pair2.fq
```

Chapter 78

How to make a BAM file

Several chapters will start with by analyzing a BAM file. This section shows the code for creating a BAM file using published data. The script aligns sequencing data from the Redo: Genomic surveillance elucidates Ebola virus origin project using both `bwa` and `bowtie2`.

You can download this script from

- <http://data.biostarhandbook.com/scripts/make-bam.sh>

You may also obtain it from the command line with:

```
wget -nc http://data.biostarhandbook.com/scripts/make-bam.sh
```

In addition check the Bioinformatics Data Analysis recipes¹ for further code examples.

The script mentioned above contains the following commands:

```
# Stop on any error.  
set -uex  
  
# Accession number for the reference genome  
ACC=AF086833  
  
# SRA run number.  
SRR=SRR1972739
```

¹<https://www.bioinformatics.recipes/recipe/list/bio-data-analysis/>

```
# How many reads to extract from the dataset.  
N=10000  
  
# The reference genome in three different formats: GenBank, FASTA and GFF  
GB=refs/$ACC.gb  
FA=refs/$ACC.fa  
GFF=refs/$ACC.gff  
  
# Make the reference directory.  
mkdir -p refs  
  
# Get a genbank file.  
efetch -db nucleotide -format=gb -id=$ACC > $GB  
  
# Convert the GenBank file into GFF3.  
cat $GB | seqret -filter -feature -osformat gff3 > $GFF  
  
# Convert the GenBank file into FASTA.  
cat $GB | seqret -filter -feature -osformat fasta > $FA  
  
# Create an index of the FASTA file  
samtools faidx $FA  
  
# Obtain the dataset.  
fastq-dump -X $N --split-files $SRR  
  
# Index reference with bwa  
bwa index $FA  
  
# Index the reference with samtools  
samtools faidx $FA  
  
# Shortcuts to read names  
R1=${SRR}_1.fastq  
R2=${SRR}_2.fastq  
  
# Align with bwa mem.  
bwa mem $FA $R1 $R2 | samtools sort > $SRR.bwa.bam
```

```
# Index the BAM file generated with bwa.  
samtools index $SRR.bwa.bam  
  
# Index reference with bowtie2.  
bowtie2-build $FA $FA  
  
# Align the same data with bowtie2.  
bowtie2 -x $FA -1 $R1 -2 $R2 | samtools sort > $SRR.bowtie.bam  
  
# Index the BAM file produced with bowtie2.  
samtools index $SRR.bowtie.bam
```

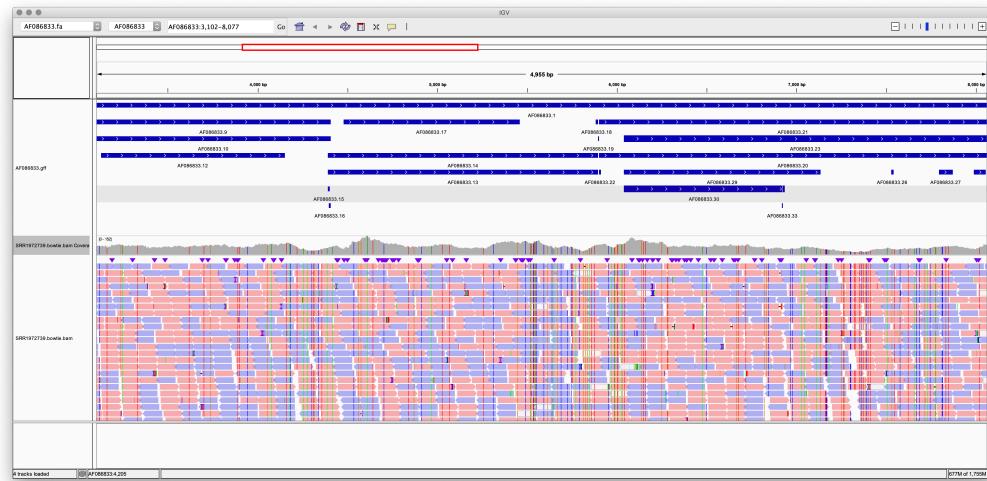
Running the script above produces 25 files:

```
find . | wc -l  
# prints 25
```

Out of those the following are of interest information:

- `refs/AF086833.fa` - the reference genome sequence.
- `refs/AF086833.gff` - is the GFF feature file for the genome.
- `SRR1972739.bwa.bam` - is the alignment file produced with `bwa`.
- `SRR1972739.bowtie.bam` - is the alignment file produced with `bowtie2`.

The script also produces a GFF file that we can visualize with IGV together with our alignment files:



We can produce statistics on each BAM file with:

```
samtools flagstat SRR1972739.bwa.bam
```

produces:

```
20740 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
740 + 0 supplementary
0 + 0 duplicates
15279 + 0 mapped (73.67% : N/A)
20000 + 0 paired in sequencing
10000 + 0 read1
10000 + 0 read2
14480 + 0 properly paired (72.40% : N/A)
14528 + 0 with itself and mate mapped
11 + 0 singletons (0.05% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

whereas

```
samtools flagstat SRR1972739.bowtie.bam
```

prints:

```
20000 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
```

```
0 + 0 supplementary
0 + 0 duplicates
12537 + 0 mapped (62.69% : N/A)
20000 + 0 paired in sequencing
10000 + 0 read1
10000 + 0 read2
9828 + 0 properly paired (49.14% : N/A)
11482 + 0 with itself and mate mapped
1055 + 0 singletons (5.27% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

Chapter 79

The SAM format explained

79.1 What is a SAM file?

The SAM format is a TAB-delimited, line-oriented text format consisting of a

1. **Header** section, where each line contains some metadata
2. **Alignment** section where each line provides information on an alignment

The SAM format¹ specification lists the required and optional content for each of these sections.

79.2 Where do I get a BAM file?

The commands in this section assume that your BAM was generated in the manner described in the How to make a BAM file section.

You may also choose a remotely hosted BAM file. For example select data from publication discussed in Redo: Explore the genome of a paleolithic-era archaic human:

```
# Location of the BAM file.  
URL=http://cdna.eva.mpg.de/denisova/alignments/T_hg19_1000g.bam
```

¹<http://samtools.github.io/hts-specs/SAMv1.pdf>

```
# Download alignments for chromosome 22.  
samtools view -b $URL 22 > denisova.chr22.bam
```

```
# Index the downloaded alignments.  
samtools index denisova.chr22.bam
```

Alternatively you may explore a subset of data from Redo: A synthetic-diploid benchmark for accurate variant-calling evaluation (takes about 15 minutes to download)

```
# Location of the BAM file.  
URL=ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR134/ERR1341796/CHM1_CHM13_2.bam
```

```
# Download alignments for chromosome 22.  
samtools view -b $URL 22 > syndip.chr22.bam
```

```
# Index the downloaded alignments.  
samtools index syndip.chr22.bam
```

79.3 What is the SAM header?

While we could temporarily save BAM as SAM files then treat them as 11 column text files, in practice, we never do that. Almost exclusively we only “view” BAM files in the SAM format. The tool that does the “live” conversion of BAM to SAM is called **samtools**.

To view the header of a SAM file we would type:

```
samtools view -H SRR1972739.bwa.bam
```

The lines that start with the @ symbols are the so-called “SAM header” rows:

```
@HD VN:1.6 SO:coordinate  
@SQ SN:AF086833 LN:18959  
@PG ID:bwa PN:bwa VN:0.7.17-r1188 CL:bwa mem refs/AF086833.fa SRR1972739_1.fastq SRR19
```

The headers are described in the SAM Specification². Various information encoded here, for example, SN is the sequence name that we aligned against

²<http://samtools.github.io/hts-specs/SAMv1.pdf>

(the name of the sequence in the Fasta reference file), LN is the length of this sequence, PG is the program version that we ran. But if you were to look at the alignment file produced with `bowtie`

```
 samtools view -H SRR1972739.bowtie.bam
```

you would see a slightly different header:

@HD VN:1.0 SO:coordinate

@SQ SN:AF086833 LN:18959

©PG ID:bowtie2 PN:bowtie2 VN:2.3.4.3 CL:"/Users/ialbert/miniconda3/envs/bioin

Depending on what kinds of post-processing information went into a BAM file the headers may be quite extensive. Besides, samtools can access remote data as well. In the Redo: A synthetic-diploid benchmark for accurate variant-calling evaluation (#syndipredo) project we noted a BAM file under the URL

- `ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR134/ERR1341796/CHM1_CHM13_2.bam`. We can readily view its header:

get the header for the remote BAM file with:

```
URL=ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR134/ERR1341796/CHM1_CHM13_2.bam  
samtools view -H $URL > header.txt
```

Note how we cautiously placed the header into a separate file. why?

```
cat header.txt | wc -l  
# prints 171
```

It has 171 lines, with content that looks like this:

```
generate_nWayOut_md5s=false check_early=false noPGTag=false keepPGTags=false indelsFile
statisticsFileForDebugging=/seq/picard_aggregation/G94794/CHMI_CHMI3_WGS1/v3/1/CHMI_C
@PG    ID:MarkDuplicates    VN:5589b97c07fd801bd8afb2999f2bf66a81dede7c    CL:picard.s
READ_ONE_BARCODE_TAG=RX INPUT=[/pod/pod2temp/pod2/tmp/flowcell/HJCMTCCXX/C1-310_2016-
OUTPUT=/pod/pod2temp/pod2/tmp/flowcell/HJCMTCCXX/C1-310_2016-01-13_2016-01-16/5/Pond-
METRICS_FILE=/pod/pod2temp/pod2/tmp/flowcell/HJCMTCCXX/C1-310_2016-01-13_2016-01-16/5
OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500 TMP_DIR=[/local/scratch/HJCMTCCXX, /pod/pod2temp
MAX_SEQUENCES_FOR_DISK_READ_ENDS_MAP=50000 MAX_FILE_HANDLES_FOR_READ_ENDS_MAP=8000 SOR
DUPLICATE_SCORING_STRATEGY=SUM_OF_BASE_QUALITIES PROGRAM_RECORD_ID=MarkDuplicates PROG
...

```

It is a monster header! The PG tag lists information on what programs were run and what parameters each had. It was intended to help us better understand what has been done to produce the BAM file. But looking at this file does it really? ... it is convoluted and complicated, most likely needlessly so. If there is one thing I have learned from watching geniuses at work is that they frequently lack the ability, or perhaps the need (?), to simplify. Look at file naming as evidence:

/pod/pod2temp/pod2/tmp/

the people that analyzed this data use a computer that has a root level directory `pod` that contains another directory called `pod2temp` that contains another directory called `pod2` that contains a directory called `tmp`. That's a bit silly arrangement - I am noting it here as an example of how unrelated details of a computer "leak" into metadata.

79.4 What is stored in the SAM alignment section?

The SAM Specification³ contains a detailed description of each field and is a required read for all bioinformaticians. Whenever in doubt, consult it.

The simplest way to understand a SAM file is to go over it a few columns at a time. The SAM spec has a name for each column, for example, column 1 is also called `QNAME`, which corresponds to the words "Query name". Let's look at the

³<http://samtools.github.io/hts-specs/SAMv1.pdf>

79.5 Column 1: Query Name (QNAME)

Cut the first column:

```
samtools view SRR1972739.bwa.bam | cut -f 1 | head -5
```

will produce:

```
SRR1972739.9152
SRR1972739.4087
SRR1972739.1404
SRR1972739.7291
SRR1972739.8376
```

Whereas the original FASTQ file had the reads labeled in order:

```
SRR1972739.1
SRR1972739.2
SRR1972739.3
```

the alignments in the BAM file are sorted by position. The SAM file tells us that the read that was the 9152th in the file aligned in the leftmost position.

79.6 Column 2: FLAG

As it happens a substantial amount of information necessary to figure out which read aligns in what way is included in the second column of the SAM format, in the “magical” number called FLAG:

```
samtools view SRR1972739.bwa.bam | cut -f 2 | head -5
```

that produces the following:

```
163
163
163
163
99
```

The FLAG is the field of the SAM spec where the designers of the format have strayed the furthest from what one might call *common sense*. It was a way to “cram” (no pun intended) as much information as possible into a single number. To do so, they came up with a contorted rationalization that

no biologist will ever manage to remember, hence ensuring the long-term employment of bioinformaticians across the globe. Perhaps that was the plan all along, in which case I must admit, the FLAG is not such a bad thing after all.

Here is how the rationale works: Imagine that we associate an attribute to each power of 2 starting from zero. Let's say

- 1 means “red”,
- 2 means “blue”,
- 4 means “tall”,
- 8 means “short”,
- 16 means “fast”,
- 32 means “cheap”,

At this point, it is not important what the attributes are just that they correspond to the information we wish to know. If we were to describe an object as a 5 we would decompose that into powers (factors) of 2 like so $1 + 4$, in essence stating that when we label something as 5 what mean is that it is “red” and “tall”. If we had a 10 that would be $2 + 8$ meaning that its attributes are “blue” and “short”. If it is 37 it is “red”, “tall” and “fast”.

The justification behind this encoding was that instead of having to store two labels, 2 and 8, we can just store a single number, 10, yet still represent two attributes. Expanding one this we can now store up to 8 factors in a byte (8 bits).

It is essential to remember that if we wanted to identify the items that are “blue” (that is, 2) we would need to select ALL the numbers that, when decomposed into powers of 2 would contain 2. And that is where it ends up quite counter-intuitive as it includes the numbers 3,6,7 and 10 but would NOT include 4, 9 or 16 etc.

See why: $1 + 2 = 3$, $2 + 4 = 6$, $1 + 2 + 4 = 7$, $2 + 8 = 10$ and so on. So everything described as 3,6,7,10 is blue (among other attributes). But of course, all these objects will be different in one or more of their other attributes. While the method may look elegant and fast to people familiar with binary representations - like the computational geniuses that came up with it - it does not match at all how humans think about these problems.

First, it is challenging to see by eye which flags are set and which are not:

is flag 2 set in the numbers 19 or 35? Moreover, what if some of the attributes are mutually exclusive but others are not? For example, red and blue may be mutually exclusive but “fast” and “cheap” are not. When selecting for attributes and lack thereof, we end up with surprisingly hard to parse statements.

One of the curses of the FLAG system is that it is exceedingly easy to mistakenly generate a wrong type of query as one adds more terms into it.

Imagine handling eight different attributes, some of which are exclusive and others not; some traits are filled in by the aligner; others are not. Selecting or removing alignments by these attributes becomes unnecessarily complicated and error-prone. The flaw of the FLAG system is that it accentuates rather than solves an already difficult problem. Using the FLAG system correctly is surprisingly challenging.

The SAM specification⁴ lists the meaning of the flags, and the `flags` command will list them at the command line:

```
samtools flags
```

generating:

Flags:

0x1	PAIRED	.. paired-end (or multiple-segment) sequencing technology
0x2	PROPER_PAIR	.. each segment properly aligned according to the aligner
0x4	UNMAP	.. segment unmapped
0x8	MUNMAP	.. next segment in the template unmapped
0x10	REVERSE	.. SEQ is reverse complemented
0x20	MREVERSE	.. SEQ of the next segment in the template is reversed
0x40	READ1	.. the first segment in the template
0x80	READ2	.. the last segment in the template
0x100	SECONDARY	.. secondary alignment
0x200	QCFAIL	.. not passing quality controls
0x400	DUP	.. PCR or optical duplicate
0x800	SUPPLEMENTARY	.. supplementary alignment

Adding insult to injury, these are now listed in “octal” representation. Let’s see what our flags mean, our list contained 163 and 99. We can ask for a description for each flag:

⁴<http://samtools.github.io/hts-specs/SAMv1.pdf>

```
samtools flags 163
```

and that results in:

```
0xa3      163 PAIRED,PROPER_PAIR,MREVERSE,READ2
```

We can read this out loud. You could say:

When I see an alignment with the FLAG set to 163 I know right away that it came from a paired read. Both the read and its mate map in in a way that a “proper pair” should. The mate aligns on the reverse strand, and the read sequence comes from the second file.

Whoa! A lot of information in just one number, wouldn't you say?

We are not done yet - and in my opinion - this captures the fundamental flaw of the FLAG system. To understand the alignment above you have also to enumerate some of the flags that are not set. Let's look at the unset flags: UNMAP, MUNMAP, REVERSE, READ1, SECONDARY, QCFAIL, DUP, SUPPLEMENTARY. Mental gymnastics is now necessary to list those attributes that, when not set, are meaningful.

For example, READ1 is not meaningful, we already know that the read comes from the second pair as it says READ2. We have learned nothing new by saying, it is not READ1 we knew that already. But we do learn something from seeing how REVERSE is not set. It means that the current alignment is on the forward strand. Then we can also learn something from seeing how the flags for neither SUPPLEMENTARY and SECONDARY alignments are not present. It means that this alignment is a primary alignment on the forward strand. So the story of this alignment now becomes:

When I see an alignment with the FLAG set to 163 I know right away that: it aligns (UNMAP not set), it aligns on the forward strand (REVERSE not set), it is a primary alignment (SECONDARY and SUPPLEMENTARY not set), it came from a paired read (PAIRED is set), where both the read and its mate map in in a way that a “proper pair” should (PROPER_PAIR is set), the mate aligns on the reverse strand (MREVERSE is set), and the sequence comes from the second file (READ1 is set).

How about DUP, QCFAIL? Those are not meaningful in this case since we don't know if we have attempted to detect these flaws or not. A read not being a duplicate is only meaningful if we tried to label duplicates - which most none of the aligners do. And since we are here what does a PROPER_PAIR even mean? I have a surprise for you, it could mean anything. The SAM spec only says "*each segment properly aligned according to the aligner*". So there you have it, it is a proper pair according to bwa.

In the wonderful world of SAM you will know something is primary when it is neither secondary nor supplementary, and you know something must be forward from the simple fact that it is not labeled reverse. If it is all Yoda approved, use the Force to understand it:



Now take a look at the flags 99 it says:

```
0x63      99  PAIRED,PROPER_PAIR,MREVERSE,READ1
```

To generate statistics on the flags:

```
samtools flagstat SRR1972739.bwa.bam
```

79.7 What are primary, secondary and chimeric (supplementary) alignments

The most important FLAGS characterizing the alignment type are:

1. **Secondary alignment:** a linear alignment that covers different regions of a read.

2. **Supplementary alignment:** a read that cannot be represented as a single linear alignment and matches two different locations without overlap. These are also known as “chimeric” alignments. What gets called as chimeric alignment is software/implementation dependent.

an alignment not labeled as secondary or supplementary is then a:

1. **Primary alignment:** the best alignments by the score. If multiple alignments match equally well, the aligner will designate one (software dependent) as the primary alignment.

79.8 Columns 3-4: Reference Name (RNAME) and Position (POS)

The 3rd and 4th columns of a SAM file indicate the reference name (RNAME) and the position (POS) of where the query, the sequence named in column 1 (QNAME) aligns:

```
samtools view SRR1972739.bwa.bam | cut -f 1,3,4 | head -5
```

In our case, there is only one reference sequence (chromosome):

SRR1972739.9152	AF086833	46
SRR1972739.4087	AF086833	62
SRR1972739.1404	AF086833	80
SRR1972739.7291	AF086833	103
SRR1972739.8376	AF086833	108

We can see that each sequence aligns to the same reference at different positions: 46, 62, 80 and so on.

Everything in a SAM file is relative to the forward strand!

The essential thing to remember about the POS field is that it reflects the **leftmost** coordinate of the alignment. Even when the alignment matches on the reverse strand the POS coordinate is reported relative to the forward strand.

It is the FLAG column that tells us which strand the data aligns to FLAG=4

79.9 Column 5-6: Mapping Quality (MAPQ) and Compact Idiosyncratic Gapped Alignment Representation (CIGAR)

These columns reflect the Mapping Quality (MAPQ) and the so-called Compact Idiosyncratic Gapped Alignment Representation (CIGAR).

```
samtools view SRR1972739.bwa.bam | cut -f 5,6 | head -5
```

to produce:

```
60 101M
60 101M
60 101M
60 101M
60 101M
```

The values in the MAPQ column here are all 60. This column was designed to indicate the likelihood of the alignment being placed incorrectly. It is the same Phred score that we encountered in the FASTQ files. And we read it the same way, $60/10 = 6$ so the chance of seeing this alignment being wrong is 10^{-6} or 1/1,000,000 one in a million.

There is one caveat though. The numbers that an aligner puts into the MAPQ field are typically estimates. It is not possible to mathematically compute this value. What this field does is to inform us on a guess by the aligner's algorithm. This guess is more of a hunch that should not be treated as a continuous, numerical value. Instead, it should be thought of as an ordered label, like "not so good" or "pretty good". Aligner developers even generate unique MAPQ qualities to mark individual cases. For example, bwa will create a MAPQ=0 if a read maps equally well to more than one location.

The CIGAR string is a different beast altogether. It is meant to represent the alignment via numbers followed by letters:

- M match of mismatch
- I insertion
- D deletion
- S soft clip
- H hard clip
- N skipping

To find CIGAR strings other than 101M you could write:

```
samtools view SRR1972739.bwa.bam | cut -f 5,6 | grep -v "101" | head -5
```

to produce

```
60 18S83M
60 92M9S
60 90M11S
60 98M3S
60 3S98M
```

These are also meant to be “readable”; the 18S83M says that 18 bases soft clipped and the next 83 are a *match or mismatch*.

The CIGAR representation is a neat concept, alas it was developed for short and well-matching reads. As soon as the reads show substantial differences the CIGAR representations are much more difficult to read.

There are also different variants of CIGAR. The “default” CIGAR encoding describes both the match and mismatch the same way, with an M. There are some unfortunate rationale and explanation for having adopted this choice - one that complicates analysis even more.

The extended CIGAR encoding adopted by others uses the symbols of = and X to indicate matches and mismatches.

79.10 Columns 7-9: RNEXT, PNEXT, TLEN

Columns 7,8 and 9 deal with the alignment of the mate pair. Since both reads that from a pair originated from the same fragment, it is very informative to know right away where both ends align. And that is what these columns do.

Let us also include the QNAME and POS fields and look at the data:

```
samtools view SRR1972739.bwa.bam | cut -f 7,8,9 | head -5
```

We see the following:

=	230	281
=	224	263
=	205	226

=	238	236	
=	194	187	

The = symbol for the RNEXT field indicates that the query mate aligns to the same reference anchor (chromosome) as the query. The PNEXT fields indicate the POS field of the **primary(!)** alignment of the mate. (see later what primary alignment means). Note how every PNEXT always has a corresponding POS column on a different line since the mate is also in the file. Finally, the TLEN column indicates the distance between the outer ends of mate alignments, that is, how long the fragment **appears** to be! Making this distinction is imperative. TLEN is not the actual length of the fragment - it is the length that it would be if the genome under study was identical to the reference and the mapping was correct. The specification calls it observed template length.

Working out how the TLEN is computed takes some effort. You usually don't need to do it manually, but it is good to know how it works. To do it you need the POS and CIGAR information:

```
samtools view SRR1972739.bwa.bam | cut -f 1,2,4,6,7,8,9 | grep SRR1972739.9152
```

We have isolated the information for the two reads that form a pair:

SRR1972739.9152	163	46	101M	=	230	281
SRR1972739.9152	83	230	97M4S	=	46	-281

The flag 163 indicates that the alignment is on the forward strand (not reverse) and the leftmost position of it is at 46 with its mate aligning at position 230 on the reverse strand. The read has a CIGAR match of 101 bases, so its right coordinate is at $46+101=147$. The mate aligns at 230 and has a CIGAR of 97M4S, so it aligns over 97 bases only. This means that the rightmost coordinate of the second read is at $230 + 97 = 327$ following:

46	147	230	327	
=====>		<=====		

Note how from the above picture SAM only reports 46 and 230 (a major flaw in my opinion by the way) it takes a lot of effort to find the other two coordinates. How long is the distance between the outermost coordinates? $327 - 46 = 281$, and that is the template length.

46	147	230	327	
----	-----	-----	-----	--

```

|       |       |
=====>   <=====
|----- 281 -----|

```

So the distance between the outer edges is 281 that if under ideal conditions corresponds to the length of the fragment that was sequenced. For the mate, the value is reported as -281 to indicate that the mate is on the reverse strand.

It is exceedingly common to need the alignment length information, and we typically have to resort to tools such as `bedtools` to convert

```
cat SRR1972739.bwa.bam | bedtools bamtobed | head -1
```

That prints:

```
AF086833 45 146 SRR1972739.9152/2 60 +
```

But of course, as typical in bioinformatics this does not precisely match what we computed above. It is now a zero-based coordinate system - where the leftmost coordinate is off by one 45 instead of 46. Gee thanks `bedtools`. Will the slightly confusing behaviors ever end? Wouldn't bet on it.

79.11 Columns 10-11: SEQ and QUAL

These columns store the sequence and the FASTQ quality measures associated with the query.

```
samtools view SRR1972739.bwa.bam | cut -f 10,11 | head -1
```

will produce the read sequence and its FASTQ quality:

```
GAATAACTATGAGGAAGATTAATAATTTCCTCTCATTGAAATTATCGGAATTAAATTGAAATTGTTACTGTAATCATACC
```

The most important detail (not quite explained well in the SAM specification) is that these columns store the sequences relative to the *forward* strand. What this means is that queries that align to the reverse strand will be shown as their reverse complement.

The soft and hard clip qualifiers of the `CIGAR` field describe whether the clipped and unaligned sequence is still included (soft clip) or not (hard clip) in the `SEQ` column.

The QUAL column contains those sequence qualities that directly correspond to the SEQ columns.

79.12 Optional Fields: Columns 12, 13 and on

Here is where it gets interesting and results produced by different aligners start to diverge. All optional fields follow the TAG:TYPE:VALUE format for example, where TAG is a two letter word, TYPE is one character long (like i, Z, f) while the VALUE can be arbitrarily long. Let us look at some tags:

```
samtools view SRR1972739.bwa.bam | cut -f 12,13,14,15 | head -1  
will produce:
```

```
NM:i:1      MD:Z:81C19      MC:Z:97M4S      AS:i:96
```

As a rule, TAGs that start with an X or Y are tool specific and are (should be) documented by the aligner.

79.13 What do the SAM tags mean?

The definitions for SAM tags are distributed as a standalone document called SAMtags.pdf⁵

- Tags are optional.
- Only some upper cased tags have pre-defined meanings.
- Tags that start with X are aligner
- Tags containing lowercase letters are always specific to the tool that filled them in.

For example, we can see there that NM is defined as “edit distance to the reference, including ambiguous bases but excluding clipping”. And MD is *string for mismatching positions*, yet another representation of the alignment itself, similar to CIGAR but not quite the same. There is a long story to the MD tag, one that will be covered later.

⁵<http://samtools.github.io/hts-specs/SAMtags.pdf>

The take-home lesson from the above is that the SAM format represents alignments and stores a relatively complex set of information on the way sequences align.

Chapter 80

Working with BAM files

The SAM/BAM format is the workhorse format of bioinformatics. The information in the SAM files (or lack thereof) may determine the success or failure of a project.

Several tool sets have been created to manipulate BAM files:

1. `samtools` - one of the most powerful data analysis tools in all of bioinformatics. Complete data analytics may be performed using `samtools` alone.
2. `bamtools` - originally conceived of to fill in functionality missing from `samtools`.
3. `picard` - a toolset developed at the Broad Institute (named after Captain Picard in the Star Trek Next Generation series); Sometimes used in close conjunction with the GATK variant caller.

Many operations on BAM files fall into one of these two categories:

- Select alignments that match an attribute: strand, mate information, mapping quality
- Select alignments that align in a certain region of the genome.

80.1 How can I better understand the FLAGS?

The best resource to explore SAM flags is a site called

- Explain SAM Flags¹.

The service allows you to iteratively build flags and to reverse their meaning.

80.2 Where do I get a BAM file?

The commands in this section assume that your BAM was generated in the manner described in the How to make a BAM file section.

You may also choose a remotely hosted BAM file. For example select data from publication discussed in Redo: Explore the genome of a paleolithic-era archaic human:

```
# Location of the BAM file.  
URL=http://cdna.eva.mpg.de/denisova/alignments/T_hg19_1000g.bam  
  
# Download alignments for chromosome 22.  
samtools view -b $URL 22 > denisova.chr22.bam  
  
# Index the downloaded alignments.  
samtools index denisova.chr22.bam  
  
Alternatively you may explore a subset of data from Redo: A synthetic-diploid benchmark for accurate variant-calling evaluation (takes about 15 minutes to download)  
  
# Location of the BAM file.  
URL=ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR134/ERR1341796/CHM1_CHM13_2.bam  
  
# Download alignments for chromosome 22.  
samtools view -b $URL 22 > syndip.chr22.bam  
  
# Index the downloaded alignments.  
samtools index syndip.chr22.bam
```

¹<https://broadinstitute.github.io/picard/explain-flags.html>

80.3 How can I extract a section of the BAM file?

The need to isolate a subsection of a bam file is exceedingly common. The SAM format was designed around this use case. The “slice” your SAM file over a region pass a “query” string at the end of the command. The query string should be in the format “chrom:start-end” in a one based coordinate system:

```
samtools view SRR1972739.bwa.bam AF086833:1000-1010 | wc -l
# Prints 82
```

to produce another BAM file, that contains only data for the “slice” pass the -b flag to `samtools` like so:

```
# Create a new BAM file.
samtools view -b SRR1972739.bwa.bam AF086833:1000-1010 > selected.bam

# Index the new BAM file
samtools index selected.bam
```

since we tend to forget to re-index the BAM file when running quick examples, it makes sense to combine the two commands into a single one, separated by the `&&` symbols that bash interprets as “AND”:

```
 samtools view -b SRR1972739.bwa.bam AF086833:1000-1010 > selected.bam && samtools
```

In the future examples we will combine the two commands into one. When visualized in IGV the resulting BAM file would look like this:



As the above image shows, after taking the slice only those alignments were retained that overlapped with the coordinate 1000–1010.

80.4 How can I select from or filter data from BAM files?

First, let's clarify some wording:

1. Selecting means to keep alignments that match a condition.
2. Filtering means to remove alignments that match a condition.

Filtering on FLAGS can be done via `samtools` by passing the parameters `-f` and `-F` that operate the following way:

- `-f flag` includes only alignments where the bits match the bits in the flag
- `-F flag` includes only alignments where the bits DO NOT match the bits in the flag.

80.5 How do I validate (mapped) alignments?

First recall what flag 4 means:

```
samtools flags 4
```

It states:

```
0x4 4 UNMAP
```

This means that when this flag is set, the read is unmapped (unaligned). You may wonder why this alignment is included at all in the BAM file if there is no actual alignment for it. We often need to know which read did not produce an alignment, thus some aligners include them by default. Others do not. Again one of the differences between aligners. To select and view alignments where the read did not align you can do:

```
samtools view -f 4 SRR1972739.bwa.bam | head
```

Counting alignments with a property is so frequently needed that samtools provides flag `-c` to do that:

```
samtools view -c -f 4 SRR1972739.bwa.bam
```

to produce:

```
5461
```

Thus 5461 reads in our file are not actually aligned. To count those that are aligned reverse the flags:

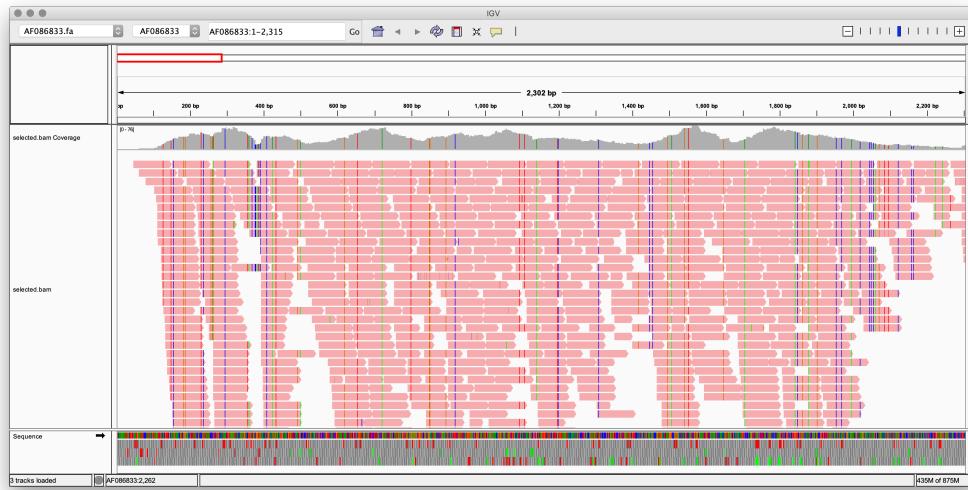
```
# How many don't have the UNMAP flag?
samtools view -c -F 4 SRR1972739.bwa.bam
# 15279
```

80.6 How do I select forward or reverse alignments?

To select alignments on the forward strand we would filter out UNMAP (4) and REVERSE (16) ($4+16=20$):

```
samtools view -F 20 -b SRR1972739.bwa.bam > selected.bam && samtools index selected.bam
```

80.6. HOW DO I SELECT FORWARD OR REVERSE ALIGNMENTS?629



To select alignments on the reverse strand we would filter out UNMAP(4) but select for REVERSE (14):

```
samtools view -F 4 -f 16 -b SRR1972739.bwa.bam > selected.bam && samtools index selected.bam
```



80.7 How to separate alignments into a new file?

Removing the `-c` parameter will produce the alignments, so if you wanted to create another BAM file that only contains the aligned reads you could do:

```
samtools view -b -F 4 SRR1972739.bwa.bam > aligned.bam  
samtools index aligned.bam
```

80.8 How to get an overview of the alignments in a BAM file?

1. `samtools flagstat SRR1972739.bwa.bam` - produces a report on flags
2. `samtools idxstats SRR1972739.bwa.bam` - produces a report on how many reads align to each chromosome
3. `bamtools stats -in SRR1972739.bwa.bam` - produces a report on flags

One observation that we made over the years is that the numbers reported by the various statistics programs never quite match! The numbers are similar but never all the same in each category.

This is an indication that even at the basic level there is ambiguity in what the different terms mean to different people. Note, for example, how below `samtools flagstat` reports 14480 as “proper pairs” whereas `bamtools stats` reports 15216 for the same quantity.

The output of `samtools flagstat`

```
20740 + 0 in total (QC-passed reads + QC-failed reads)  
0 + 0 secondary  
740 + 0 supplementary  
0 + 0 duplicates  
15279 + 0 mapped (73.67% : N/A)  
20000 + 0 paired in sequencing  
10000 + 0 read1  
10000 + 0 read2  
14480 + 0 properly paired (72.40% : N/A)
```

80.8. HOW TO GET AN OVERVIEW OF THE ALIGNMENTS IN A BAM FILE?631

```
14528 + 0 with itself and mate mapped
11 + 0 singletons (0.05% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

The output of `bamtools stats`

```
*****
```

Stats for BAM file(s):

```
*****
```

Total reads:	20740
Mapped reads:	15279 (73.6692%)
Forward strand:	14393 (69.3973%)
Reverse strand:	6347 (30.6027%)
Failed QC:	0 (0%)
Duplicates:	0 (0%)
Paired-end reads:	20740 (100%)
'Proper-pairs':	15216 (73.3655%)
Both pairs mapped:	15268 (73.6162%)
Read 1:	10357
Read 2:	10383
Singlets:	11 (0.0530376%)

How could you verify which one is correct? Take the definition of proper pair as flag 2:

```
samtools view -c -f 2 SRR1972739.bwa.bam
# 15216
```

If you count the PROPER_PAIR flag the `bamtools stats` is correct.

So how does `samtools flagstat` come up with 14480? It turns out that it counts only the properly mapped AND primary alignment reads. Basically, it is counting each read only once in the cases that a read produces multiple alignments:

How do we produce the alignment the primary alignments? It is a bit circuitous, we filter out the unmapped, secondary and supplementary alignments

```
samtools flags UNMAP,SECONDARY,SUPPLEMENTARY
0x904 2308 UNMAP,SECONDARY,SUPPLEMENTARY
```

all we need is to filter (remove alignments) with flag 2308

```
samtools view -c -f 2 -F 2308 SRR1972739.bwa.bam
# 14480
```

80.9 What is a proper-pair?

“Proper pair”, called “concordant” in bowtie is a fuzzy and under specified term . The SAM specification only states that PROPER_PAIR “*each segment properly aligned according to the aligner*” which is no definition at all. The aligner, may decide to start label certain alignments as PROPER_PAIR.

Typically what proper pair means is that the read pair aligns in an expected manner. The distance between the outer edges is within expected ranges, the orientation of the reads are towards one another ---> <----.

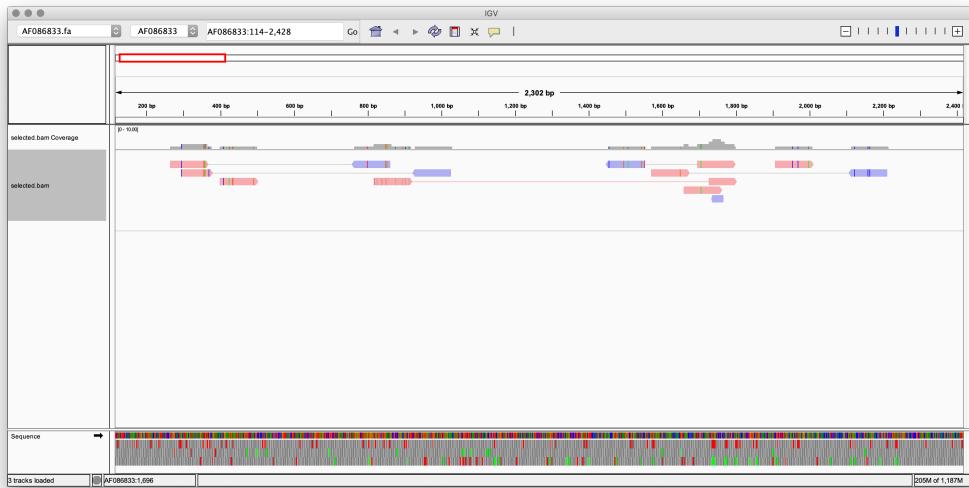
The word PROPER_PAIR is loaded and somewhat misleading, it appears to suggest that these alignments are more “correct” than others. That’s not what it supposed to mean or imply. Always remember that when we align against a reference, we align real data against an imaginary, hypothetical reference. If our real data has a genome reorganization, it will produce sequences that when forced over the hypothetical reference may show up as “improper pairs”.

It is also true, of course, that errors in the protocols may produce flawed data that also aligns incorrectly. Our job is to recognize and distinguish between the two cases.

Often it is interesting to see which alignments are not “proper” or “primary”. These often carry information on the differences relative to the reference genome. Let’s filter out the UNMAP (4) and PROPER_PAIR (2)

```
samtools view -F 6 -b SRR1972739.bwa.bam > selected.bam && samtools index selected
```

the command above will produce the selection



Few reads, distributed all across the genome. The alignments do not seem to indicate a genomic reorganization.

80.10 How do I use flags?

The most important rule is to ensure that you filter on flag 4, the UNMAP. Even though it makes little sense that a read could be simultaneously unmapped and on a reverse strand, this can be reported as such:

```
# Alignments that are not mapped to a location
samtools view -c -f 4 SRR1972739.bwa.bam
# 5461
```

```
Alignments that are mapped to a location.
samtools view -c -F 4 SRR1972739.bwa.bam
# 15279
```

```
# Reads that align to reverse strand.
samtools view -c -F 4 -f 16 SRR1972739.bwa.bam
# 6343
```

```
# Alignments to forward strand.
```

```
samtools view -c -F 4 -F 16 SRR1972739.bwa.bam
# 8936

# Alignments that are unmapped and on the reverse strand! Say what?
samtools view -c -f 4 -f 16 SRR1972739.bwa.bam
4
```

Various curious situations may occur:

- Biostar Post of the Day: How can an unmapped read align on the reverse strand?²

80.11 How do I combine multiple flags?

This is where it gets even more complicated.

When investigating any attribute we may need to select for and against certain features. But as we combine the flags, the way we order the flags is not symmetric anymore when using `-f` and `-F` together. To select all reads that align (oppose 4) and are mapped to the reverse strand (select for 16) we do this:

```
# Alignments on the reverse strand.
# Oppose unmapped (4) and select for reverse strand (16).
samtools view -c -F 4 -f 16 SRR1972739.bwa.bam
# 6343

# Alignments on the forward strand.
# Oppose unmapped (4) and oppose the reverse strand (16).
samtools view -c -F 4 -F 16 SRR1972739.bwa.bam
# 8936

# Typically the above is combined into a single number: 4 + 16 = 20
samtools view -c -F 20 SRR1972739.bwa.bam
# 8936
```

As you select for more attributes the overhead of the mental gymnastics adds up. Here is an example that seems contrived yet is a common task if you

²<https://www.biostars.org/p/14569/#14570>

wanted to separate strand-specific data (where the second pair has the same orientation as the transcript). In that case if we wanted to separate our alignments to correspond to forward and reverse transcripts we would do:

```
# Select alignments for reads that are second in pair if they map to the forward strand.
samtools view -b -F 20 -f 128 data.bam > fwd1.bam

# Select alignments for reads that are the first in pair if they map to the reverse strand.
samtools view -b -F 4 -f 80 data.bam > fwd2.bam
```

Not surprisingly, the above quite challenging to figure out and apply correctly.

It gets even more complicated for paired-end strand-specific data. We had to write a tutorial (mainly so that we also remember it) on this:

- Biostar Post of the Day: How To Separate Illumina Based Strand Specific Rna-Seq Alignments By Strand³

80.12 How should I interpret seemingly impossible alignments?

It is possible to create selections that seemingly make no sense. For example, let us select alignments where the read and its mate both align to the reverse strand.

```
samtools flags PAIRED,PROPER_PAIR,REVERSE,MREVERSE
# 0x33 51 PAIRED,PROPER_PAIR,REVERSE,MREVERSE
```

There are 124 of these:

```
# Unique alignments, mapped, paired, proper pair, both reads
# in a pair align to the reverse strand.
samtools view -c -q 1 -F 4 -f 51 SRR1972739.bwa.bam
# 124
```

If our genome matched the reference and the alignments were correct this would be impossible to observe. Given that we do see these, there are two explanations.

1. Incorrect alignments: The reads have been misplaced.

³<https://www.biostars.org/p/92935/>

2. Genomic rearrangement: The real genome has a sequence inversion that makes it “look like” one read of the pair is inverted.

Remember that the real genome does not need to match the reference - real, biological changes could look like incorrect mapping.

80.13 How do I filter on mapping quality?

The mapping quality column contains a number that the aligner fills in with a value that ought to reflect the likelihood of errors. This number is rarely an objective quantity, it rather reflects a subjective value designated by the tool developer. For example, for `bwa` a mapping quality of 0 indicates a read that maps to multiple locations. Does this make sense? Not really, the likelihood that a read can be in multiple locations should be closer to $1/N$ (N is the number of possible alignments) - but `bwa` reports it as zero.

Do other tools use the same rationale? Typically not. You have to check the documentation for the particular software you are using to know for sure.

The `-q` flag selects alignments whose mapping quality is at or above the value

```
# Select uniquely mapped reads when these were aligned with BWA.
samtools view -c -q 1 SRR1972739.bwa.bam
# 15279
```

Selection options can be combined:

```
# Mapped alignments with quality over 1 and on the forward strand.
samtools view -c -q 1 -F 4 -F 16 SRR1972739.bwa.bam
# 8936
```

80.14 How can I find out the depth of coverage?

There are several tools that can compute the number of reads that overlap each base. With `samtools` you can do:

```
# Compute the depth of coverage.
samtools depth SRR1972739.bwa.bam | head
```

This produces a file with name, coordinate and coverage on that coordinate:

```
AF086833    46  1
AF086833    47  1
AF086833    48  1
...
...
```

Run `bamtools coverage`:

```
bamtools coverage -in SRR1972739.bwa.bam | head
```

The output seems identical

```
AF086833    45  1
AF086833    46  1
AF086833    47  1
...
...
```

Except note how it is “one off”! The index starts at 0 rather than 1 in the latter case. These type of inconsistencies are extraordinarily common in bioinformatics and are the source of endless errors.

You could ask for all positions to be shown with `samtools` and you can verify that it starts at 1

```
samtools depth -a SRR1972739.bwa.bam | head
```

it produces:

```
AF086833    1  0
AF086833    2  0
AF086833    3  0
...
...
```

You could sort by depth, though this could take a long for a large genome size. To find the most covered region:

```
samtools depth SRR1972739.bwa.bam | sort -k 3 -rn | head
```

Produces:

```
AF086833    4609    163
AF086833    4608    163
AF086833    4611    157
...
...
```

80.15 What is a “SECONDARY” alignment?

A secondary alignment refers to a read that produces multiple alignments in the genome. One of these alignments will be typically referred to as the “primary” alignment.

```
samtools flags SECONDARY
# 0x100 256 SECONDARY

samtools view -c -F 4 -f 256 SRR1972739.bwa.bam
# 0
```

There are no reads that map in multiple locations in this data set.

The SAM specification states:

Multiple mappings are caused primarily by repeats. They are less frequent given longer reads. If a read has multiple mappings, and all these mappings are almost entirely overlapping with each other; except the single-best optimal mapping, all the other mappings get mapping quality <Q3 and are ignored by most SNP/INDEL callers.

To be honest we don’t fully understand the statement above. The definition is somewhat meandering and fuzzy. Oh look, I even have the gall to criticise someone else’s writing, especially considering the quantity of typos and broken sentences that I myself produce.

80.16 What is a “SUPPLEMENTARY” alignment?

A supplementary alignment (also known as a chimeric alignment) is an alignment where the read partially matches different regions of the genome without overlapping the same alignment:

If this the original read was say 9 bp long:

123456789

Then these alignments would be reported as two alignments, one of them chimeric:

--- -----
123 456789

Above the entire sequence aligns in two different locations with no overlap. Another arrangement as below would not be reported as chimeric:

12345

6789

Typically, one of the linear alignments in a chimeric alignment is considered the “representative” or “primary” alignment, and the other is called “supplementary”. The decision regarding which linear alignment is representative (primary) is not all that well documented. We assume it is the alignment with higher score.

The SAM specification states:

Chimeric alignment is primarily caused by structural variations, gene fusions, misassemblies, RNA-seq or experimental protocols. [1]. For a chimeric alignment, the linear alignments consisting of the alignment are largely non-overlapping; each linear alignment may have high mapping quality and is informative in SNP/INDEL calling.

To select supplementary alignments:

```
samtools flags SUPPLEMENTARY
# 0x800 2048     SUPPLEMENTARY

samtools view -c -F 4 -f 2048 SRR1972739.bwa.bam
# 740
```

80.17 What is a “PRIMARY” or “REPRESENTATIVE” alignment?

Typically these represent the “best” alignment although the word “best” may not be well defined. Each read will only have one primary alignment and other secondary and supplemental alignments.

There is no flag to select for primary alignments, the opposite of “SUPPLEMENTARY” and “SECONDARY” will be the primary alignment. To select primary alignments:

```
samtools flags SUPPLEMENTARY,SECONDARY  
# 0x900 2304      SECONDARY,SUPPLEMENTARY
```

```
samtools view -c -F 4 -F 2304 SRR1972739.bwa.bam  
# 14539
```

Chapter 81

The SAM reference sheet

This page collects information on SAM files that is often needed.

- The SAM format specification¹ is the official specification of the SAM format.
- The SAM tag specification² is the official specification of the SAM tags.

81.1 What are the required SAM columns?

1. QNAME String [!-?A-~]{1,254} Query template NAME
2. FLAG Int [0,65535] bitwise FLAG
3. RNAME String *|[!-()+-<>~-][!-~]* Reference sequence NAME
4. POS Int [0,2147483647] 1-based leftmost mapping POSition
5. MAPQ Int [0,255] MAPping Quality
6. CIGAR String *|([0-9]+[MIDNSHPX=]) CIGAR string
7. RNEXT String *|=|[!-()+-<>~-][!-~]* Ref. name of the mate/next read
8. PNEXT Int [0,2147483647] Position of the mate/next read
9. TLEN Int [-2147483647,2147483647] observed Template LENGTH
10. SEQ String *|[A-Za-z.=.] segment SEQuence
11. QUAL String [!-~]+ ASCII of Phred-scaled base QUALity+33

¹<http://samtools.github.io/hts-specs/SAMv1.pdf>

²<http://samtools.github.io/hts-specs/SAMtags.pdf>

81.2 What do the flags mean?

Binary	Integer	Name	Meaning
000000000001	1	PAIRED	Read paired
000000000010	2	PROPER_PAIR	Read mapped in proper pair
000000000100	4	UNMAP	Read unmapped
000000001000	8	MUNMAP	Mate unmapped
000000010000	16	REVERSE	Read reverse strand
000000100000	32	MREVERSE	Mate reverse strand
000001000000	64	READ1	First in pair, file 1
000010000000	128	READ2	Second in pair, file 2
000100000000	256	SECONDARY	Not a primary alignment
001000000000	512	QCFAIL	Read fails platform/vendor quality checks
010000000000	1024	DUP	Read is PCR or optical duplicate
100000000000	2048	SUPPLEMENTARY	Supplementary alignment

81.3 How to build compound flags?

The best resource is a site called Explain SAM Flags³. The site allows us to not only build a series of flags but to instantly reverse their meaning, a process that can be very error-prone otherwise.

It is worth reflecting for a moment that the format is so complicated that we need a separate, standalone website just to understand one of its columns.

You can also build the flag that has all the right bits at the command line:

```
samtools flags PAIRED,PROPER_PAIR,REVERSE
```

will print the integers value 19 :

```
0x13    19    PAIRED,PROPER_PAIR,REVERSE
```

You can also go in “reverse”:

```
samtools flags 19
```

prints:

```
0x13    19    PAIRED,PROPER_PAIR,REVERSE
```

³<https://broadinstitute.github.io/picard/explain-flags.html>

81.4 What are SAM tags?

There are two types of tags:

- Header tags: These are predefined in the header and each one is present for each alignment.
- Alignment tags: These are filled in by the aligner and may be predefined or custom tags.

81.5 What do the SAM header tags mean?

Header tags start with the @ symbol and are at the beginning of the SAM file.

81.5.1 @HD: The header line

The first line if present.

- VN Format version
- SO Sorting order of alignments
- GO Grouping of alignments

81.5.2 @PG: Program

- ID Program record identifier
- PN Program name
- CL Command line
- PP Previous @PG-ID
- DS Description
- VN Program version

81.5.3 @RG: Read group

This is one of the most important headers in a SAM file.

Unordered multiple @RG lines are allowed.

- ID Read group identifier

- CN Name of sequencing center producing the read
- DS Description
- DT Date the run was produced (ISO8601 date or date/time)
- F0 Flow order
- KS The array of nucleotide bases that correspond to the key sequence of each read
- LB Library
- PG Programs used for processing the read group
- PI Predicted median insert size
- PL Platform/technology used to produce the reads
- PM Platform model
- PU Platform unit, a unique identifier
- SM Sample

81.5.4 @SQ: Reference sequence dictionary

The order of @SQ lines defines the alignment sorting order.

- SN Reference sequence name
- LN Reference sequence length
- AS Genome assembly identifier
- M5 MD5 checksum of the sequence in the uppercase
- SP Species
- UR URI of the sequence

81.6 What do the SAM alignment tags mean?

The definitions for SAM tags are distributed as a standalone document called SAMtags.pdf⁴

- Tags are optional.
- Only some upper case tags have pre-defined meanings.

⁴<http://samtools.github.io/hts-specs/SAMtags.pdf>

- Tags that start with X, Y' and Z or are in lowercase letters are always specific to the tool that filled them in. You need to consult the tool documentation for what they mean.

The current specification calls for the following pre-defined tags but please do note that no aligner fills in all and most aligners only fill very few of them:

- AM The smallest template-independent mapping quality of segments in the rest
- AS Alignment score generated by aligner
- BC Barcode sequence
- BQ Offset to base alignment quality (BAQ)
- CC Reference name of the next hit
- CM Edit distance between the color sequence and the color reference (see also NM)
- C0 Free-text comments
- CP Leftmost coordinate of the next hit
- CQ Color read base qualities
- CS Color read sequence
- CT Complete read annotation tag, used for consensus annotation dummy features.
- E2 The 2nd most likely base calls
- FI The index of segment in the template
- FS Segment suffix
- FZ S Flow signal intensities
- GC Reserved for backwards compatibility reasons
- GQ Reserved for backwards compatibility reasons
- GS Reserved for backwards compatibility reasons
- H0 Number of perfect hits
- H1 Number of 1-difference hits (see also NM)
- H2 Number of 2-difference hits
- HI Query hit index
- IH Number of stored alignments in SAM that contains the query in the current record
- LB Library
- MC CIGAR string for mate/next segment
- MD String for mismatching positions
- MF Reserved for backwards compatibility reasons
- MQ Mapping quality of the mate/next segment

- NH Number of reported alignments that contains the query in the current record
- NM Edit distance to the reference
- OC Original CIGAR
- OP Original mapping position
- OQ Original base quality
- PG Program
- PQ Phred likelihood of the template
- PT Read annotations for parts of the padded read sequence
- PU Platform unit
- QT Barcode (BC or RT) phred-scaled base qualities
- Q2 Phred quality of the mate/next segment sequence in the R2 tag
- R2 Sequence of the mate/next segment in the template
- RG Read group
- RT Barcode sequence (deprecated; use BC instead)
- SA Other canonical alignments in a chimeric alignment
- SM Template-independent mapping quality
- SQ Reserved for backwards compatibility reasons
- S2 Reserved for backwards compatibility reasons
- TC The number of segments in the template
- U2 Phred probability of the 2nd call being wrong conditional on the best being wrong
- UQ Phred likelihood of the segment, conditional on the mapping being correct
- X? Reserved for end users
- Y? Reserved for end users
- Z? Reserved for end users

81.7 What are BWA aligner specific tags?

Out of the pre-defined tags `bwa` fills the following:

- NM Edit distance
- MD Mismatching positions/bases
- AS Alignment score
- BC Barcode sequence

Lowercase tags or those that start with a X, Y or Z are application specific.

The **bwa** aligner fills in the following **bwa** specific tags:

- X0 Number of best hits
- X1 Number of suboptimal hits found by BWA
- XN Number of ambiguous bases in the reference
- XM Number of mismatches in the alignment
- X0 Number of gap opens
- XG Number of gap extinctions
- XT Type: Unique/Repeat/N/Mate-sw
- XA Alternative hits; format: (chr, pos, CIGAR, NM;)*
- XS Suboptimal alignment score
- XF Support from forward/reverse alignment
- XE Number of supporting seed

Chapter 82

Sequence variation in SAM files

The section discusses how variants are represented inside SAM files. The concepts we cover are somewhat technical but don't let that discourage you. Being confident about your files means that you fully understand how information is represented inside them.

Thankfully in the majority of cases, you won't need to operate on your files like this. But I guarantee you there will be a time when you do need to look inside the SAM file. At that time knowing where to look will save you substantial effort.

For example, when variation calling goes awry, you will need to investigate the alignment file and attempt to understand the information encoded in the file.

Let's prepare some data for this chapter.

```
# Get the reference genome.  
mkdir -p refs  
REF=refs/AF086833.fa  
efetch -db=nuccore -format=fasta -id=AF086833 > $REF  
  
# Index reference with bwa  
bwa index $REF
```

82.1 How to mutate a sequence from the command line?

The `msbar` EMBOSS tool was designed to generate mutations in sequences. It can produce point mutations with the flag:

```
-point Types of point mutations to perform
  0 (None);
  1 (Any of the following)
  2 (Insertions)
  3 (Deletions)
  4 (Changes)
  5 (Duplications)
  6 (Moves)
```

Or block mutations with the flags

```
-block Types of block mutations to perform
  0 (None)
  1 (Any of the following)
  2 (Insertions)
  3 (Deletions)
  4 (Changes)
  5 (Duplications)
  6 (Moves)
```

Block mutations are a series of consecutive mutations that span from ranges specified with `-minimum` and `-maximum`.

Instead of aligning millions of reads will use our short read aligner to align just one with known properties. We will then look at how the variant is reflected within the SAM file.

Get a 50bp query from the reference file:

```
cat $REF | seqret -filter -send 50 > query.fa

# Introduce a point deletion
cat query.fa | msbar -filter -point 3 > mutated.fa
```

```
# Visualize the output:
global-align.sh query.fa mutated.fa
```

It will produce:

```
CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTGTGTGCGAATA
||||||| | | | | | | | | | | | | | | | | | | | | | | |
CGGACACACAAAAAGAAAGAAGAATTTTA-GATCTTGTGTGCGAATA
```

Perhaps you'd want to make a different mutation, with a 2 base block mutation:

```
cat query.fa | msbar -filter -block 3 -minimum 2 -maximum 2 > mutated.fa
```

that case produced on my system the following:

```
CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTGTGTGCGAATA
||||||| | | | | | | | | | | | | | | | | | | | | | |
CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTGTGT--GAATA
```

We can introduce more than one mutation with the `-count` parameter:

```
cat query.fa | msbar -filter -count 2 -block 3 -minimum 2 -maximum 2 > mutated.fa
```

that will produce:

```
CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTGTGTGCGAATA
||||||| | | | | | | | | | | | | | | | | | | | | |
CGGACACACAAAA--AAAGAAGAATTTTAGGA--TTTGTGTGCGAATA
```

Since the tool selects mutations randomly, when choosing mutations like substitutions it is possible to mutate the sequence back to itself i.e. replacing an A with another A

One flaw of the `msbar` tool is that it does not tell us what changes it has made.

82.2 How do I reconstruct alignment from CIGAR and MD strings?

The encoding of the SAM file (add cross-reference to SAM format information) makes reconstructing the actual alignment a little more difficult than one might imagine. Let us walk through a simple example of mutating a

50bp sequence from the Ebola genome, then reconstructing that information from its alignment back to reference.

In a nutshell we will create the mutation:

```
CGGACACACAAAAAGAAAGAATTTAGGATCTTGTGTGCGAATA
|||||||.||||||||||||||||||||||||||||||||| |
CGGACACATAAAAAGAAAGAAGAATTTAGGATCTTGTGTGCGAATA
```

and this above will be represented in the CIGAR 50M and MD tag of **8C41**. You can see advantages of the shortened representations the entire long stretch above is stored in the short word **8C41**

The **-point 4** flag creates a substitution in a random location of the sequence (in your case the mutation will be somewhere else and it is also possible that the mutation generates the existing base. In that case rerun it):

```
cat query.fa | msbar --filter --point 4 > mutated.fa
```

To see what the change is we can run our global aligner:

```
global-align.sh query.fa mutated.fa
```

it will print:

```
CGGACACACAAAAAGAAAGAATTTAGGATCTTGTGTGCGAATA
|||||||.||||||||||||||||||||||||||||| |
CGGACACATAAAAAGAAAGAAGAATTTAGGATCTTGTGTGCGAATA
```

We can see how **msbar** above replaced a C with a T. So what will this look like in a SAM record? Let us generate the alignment.

```
bwa mem $REF mutated.fa > align.sam
```

Then view the CIGAR, SEQUENCE and MD tags (columns 6,10 and 13):

```
samtools view align.sam | cut -f 6,10,13
```

This prints:

```
50M CGGACACATAAAAAGAAAGAAGAATTTAGGATCTTGTGTGCGAATA MD:Z:8C41
```

The SAM specification states that the MD field aims to achieve SNP/INDEL calling without looking at the reference. What this means is that from the CIGAR, SEQUENCE and MD columns we can reconstruct the original data and we don't need to know the reference.

The MD tag is value is 8C41. It says that 8 bases match then the 9th is a mismatch and the reference contains a C at that location. The 9th base in the SEQUENCE is a T hence this we now know this is a C to T variation.

82.3 What will deletions look like in a SAM format?

Let's look at a block deletion:

```
cat query.fa | msbar -filter -count 2 -block 3 -minimum 2 -maximum 2 > mutated.fa
```

That produces:

When aligned with bwa the resulting CIGAR and MD tags are 13M2D18M2D15M and 13^AG18^TC15:

13M2D18M2D15M CGGACACACAAAAAGAAGAATTTAGGATTTGTGTGCGAATA MD:Z:13^AG18^TC

Note how the CIGAR string tells us that we have 2D but it is the MD tag that tells us what has been deleted: ^AG

82.4 What will insertions look like in a SAM format?

```
cat query.fa | msbar -filter -count 2 -block 2 -minimum 2 -maximum 2 > mutated.fa  
produces:
```

CGGACACACAAAAAGAAAG--AAGAATTTTAGGATCTTTGTG--TGCGAATA
||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| |||||
CGGACACACAAAAAGAAAGATAAGAATTTTAGGATCTTTGTGCTTGCATA

the corresponding SAM CIGAR and MD fields are 19M2I23M2I8M and 50:

19M2I23M2I8M CGGACACACAAAAAGAAAGATAAGAATTTTAGGATCTTTGTGCTTGCGAATA MD:Z:50

82.5 How long is an alignment?

This question can have different interpretations. What it typically means is the distance between the start and end coordinates on the reference.

For either case, we can compute the alignment lengths from the CIGAR string alone following these rules:

Dealing with matches and mismatches cigar strings such as 50M or in extended format 24=1X25=

These will produce an alignment that is the sum of the integers in the formula: 50. We now see some of the logic behind labeling both matches and mismatches with M. They produce the same alignment lengths. Though this labeling is hopefully on the way out - as being a grossly misleading and frankly dumb way of describing alignments. When using the extended cigar string the = will be used for matches and the X sign for mismatches and that too would also produce a $24 + 1 + 25 = 50$ bp long alignments.

Dealing with deletions say we have 3S28M2D15M the MD tag is 28^TC15. Soft clipping means that this region is not aligned. Then the alignment length is $28 + 2 + 15 = 45$ so the alignment will start at column POS and will go to POS + 45.

Dealing with insertions for example: 14M2I19M2I17M the MD tag is 50. The insertions are relative to the query and do not alter the alignment length corresponding to the reference thus it will be $14 + 19 + 17 = 50$.

82.6 How to get the alignment lengths with a tool?

The `bedtools` suite can do this:

```
bedtools bamtobed -i align.bam
```

though be warned that by default this suite uses the BED format - a zero-based data representation hence coordinates will go from zero.

82.7 How to visualize a SAM file as pairwise alignment?

Matt LaFave's tool `sam2pairwise` does this:

- <https://github.com/mlafave/sam2pairwise>
- <https://www.biostars.org/p/110498/>

usage:

```
samtools view align.bam | sam2pairwise
```

will produce:

```
AF086833      0    AF086833      6    60  3S28M2D15M * 0 0
```

```
CGGACACAAAAAGAAAGAAGAATTTTAGGA--TTTTGTGTGCGAATA
|||||||||||||||||||||  |||||||||||||  |||||||  |||||
NNNACACAAAAAGAAAGAAGAATTTTAGGATCTTGTGTGCGAATA
```

Part XIX

Genomic Variation

Chapter 83

An introduction to genomic variation

83.1 What is a variant?

When it comes to defining genomic variations, the nomenclature and definitions show a surprising level imprecision. We all know what variation is *supposed* to mean of course. It should be a difference from what we expect to see. The problem is we rarely know reality is and our expectations may be different in the future. At any given moment we only have imperfect information that changes in time.

Moreover as our understanding of genomes evolves concepts such as “reference genome” become harder to apply correctly. What should be considered a “reference genome”? After all what we call a reference genome changes too - albeit usually in a more systematic way. Come to think of it just how different are different reference genomes from one another? Can we visualize that? As it turns out it is surprisingly challenging. Then will a variant detected for the previous version of a genome still be a variant in the updated genome?

83.2 How do we classify variants?

Genomic variations are typically categorized into different classes and are often denoted with a shortened acronym:

- SNP, a single nucleotide polymorphism - A change of a single base.
- INDEL, an insertion or a deletion - A single base added or removed.
- SNV, a single nucleotide variant. A SNPs or an INDEL. The change is still single basepair long.
- MNP, a multi-nucleotide polymorphism - Several consecutive SNPs in a block.
- MNV, a multi-nucleotide variant - Multiples SNPs and INDELS as a block.
- Short variations. Variations (MNVs) that are typically less than **50bp** in length.
- Large-scale variations. Variations that are larger **50bp**.
- Structural variants. Variations on the kilobase scale that involve (one or more) chromosomes.

Radically different techniques need to be used to study problems in each class. For example for short variations, individual reads can capture them in entirety. Large-scale variants are typically detected from the relative positioning of reads about their read pairs. For structural variations entirely new techniques (e.g. nano-channel arrays, Bionanogenomics) may be needed.

As a rule, SNPs are the most reliably detectable from data. In contrast very short MNVs that include insertions and deletions are the most difficult to resolve correctly. As you shall see in one of the next chapters, for example, a deletion followed by another variation can often be better aligned in alternate positions and that that can “fool” variant callers.

83.3 What are SNPs?

SNPs (Single Nucleotide Polymorphisms) are a particular case of small scale variations. Originally were defined as:

A single nucleotide polymorphism, also known as simple nucleotide polymorphism, (SNP, pronounced snip; plural snips) is a DNA sequence variation occurring commonly within a population (e.g. 1%) in which a single nucleotide - A, T, C or G of a shared sequence differs between members of a biological species (or paired chromosome of the same individual).

Note how a SNP is defined as only a polymorphism (a change of a base) and also implies a frequency of observing it in a population.

83.4 Is the SNP term used consistently?

Not at all. Previously I have mentioned how “gene” was the most misused word in biology. Well “SNP” is a close second. In our observations, most bioinformaticians and most reference and training materials will often use the word SNP to refer to SNVs or even MNVs, sometimes to **all** short variants. For a start, the word SNPs (snip) is easy to pronounce. It’s fun, it just rolls off your tongue. Then the 1% limitation is, of course, arbitrary. The first cutoff for SNPs was at 5% a limit later reduced to 1%.

Another problem with the definition of SNPs is that scientists are unable to randomly sample populations and with that, they introduce the so-called selection bias. It could easily happen that the frequency of a variation that initially appears to be over 1% then later turns out to be less than that value.

We believe that even the scientific language is dynamic and we should accept the reality that most scientists use the word SNP to refer to polymorphisms where the frequency in a population is not known. In this book, we will do the same. Also, we need to remember that what others call SNPs may often include not just polymorphisms but also single (or very short) nucleotide insertions and deletions.

83.5 Do SNPs cause disease?

SNPs are in a sense the holy grail of genomics. Their “popularity” perhaps is a manifestation of our hopes and wishful thinking. After all wouldn’t it be great if every disease or human condition could be distilled down to a few changes and mutations in the genome? We would need to find that one SNP, and with that, we solve the problem, collect our reward and move onto the next challenge.

And sometimes (rarely though) things do work out this way. Some individual mutations do indeed cause disease(s). The OMIM database (Online Mendelian Inheritance in Man) is an online catalog of human genes and genetic disorders collect this data.

Visit the chapter called Online Mendelian Inheritance in Man for details on what is currently known about inheritable diseases and how we can interpret the data that describes the inheritance.

83.6 What is a genotype?

A genotype is the genetic constitution of an individual organism.

In the context of genomic variations, the genotyping typically refers to identifying one, more than one, or all mutations of the sequence under investigation.

Specifically, the word “genotyping” seems to be used very liberally in publications and training materials. Most of the time, when reading these it is not clear what the authors mean when they state that they have “genotype” data. What they probably and usually mean is that they have established with some confidence that a variant is present.

83.7 What is a haplotype?

A haplotype is a group of genes in an organism that is inherited together from a single parent.

In the context of genomic variations, a haplotype is a group of mutations that are inherited together. But just as with the “genotyping,” the word “haplotyping” may be used in a broader and narrower sense.

Chapter 84

Online Mendelian Inheritance of Man

The Online Mendelian Inheritance in Man (OMIM) is a continuously updated catalog of human genes and genetic disorders and traits, with the particular focus on the molecular relationship between genetic variation and phenotypic expression. It contains information about Mendelian disorders¹.

84.1 How does OMIM work?

Whereas the <http://www.omim.org/> website offers useful search and visualization functions, a different type of data mining can be performed from the command line. Do note that OMIM website is primarily intended to assist physicians.

Here I want to make a point. Providing a service to clinicians is a high-risk task as the responsibilities associated are much higher than that of other services. OMIM tries to manage this risk by producing humongous amounts of information that may even distantly related to a search or term of interest. In essence, it organizes information to some extent, but it also tries to transfer all responsibility to the end user. Basically, it makes it their job to figure out what all that means. Just about any searchable term will produce pages and pages of results. I believe that this is by design so - but remember the actual

¹https://en.wikipedia.org/wiki/Mendelian_inheritance

useful information is probably a tiny fraction (if any) of all those results. When we investigate the data that underlies OMIM do we fully realize how far from completion this information is.

84.2 Can the OMIM data be obtained?

Whereas the main web site is the entry point most people - the command line allows us to see and investigate the data that connects it all together. Hence it may give different insights and deeper understanding and appreciation. The data itself has a relatively flat and straightforward structure and is amenable to be processed via the Unix command line.

Do note however that only the gene-to-phenotype data is available for download. The content of the pages that provide clinical advice is not included.

Registering at <http://www.omim.org/> offers download access to their data in files named such as

- `mim2gene.txt`
- `mimTitles.txt`
- `genemap.txt`,
- `morbitmap.txt`
- `genemap2.txt`

files that we can investigate from the Unix command line as well. Getting the OMIM files is somewhat ludicrous it requires a process of registering, feels a bit discriminative where you can't have a gmail account etc.

84.3 What format is the OMIM data in?

The OMIM file formats (as typical in bioinformatics) are somewhat obscure and counter-intuitive, going back many decades well before the internet era. Identifiers carry markers used to inform readers of the meaning of each term:

- Asterisk (*) Gene
- Plus (+) Gene and phenotype, combined
- Number Sign (#) Phenotype, molecular basis known
- Percent (%) Phenotype or locus, molecular basis unknown

- NULL () Other, main phenotypes with a suspected Mendelian basis
- Caret (^) Entry has been removed from the database or moved to another entry

As seen above, for example, the “percent” symbol was used initially as a marker to indicate a certain mendelian phenotype or phenotypic locus for which the underlying molecular basis is not known. Today, in the data files the word Percent will be written out explicitly. It looks (and is) a bit ridiculous:

```
Percent 100070 AORTIC ANEURYSM, FAMILIAL ABDOMINAL
```

So here we are in the 21st century, exploring the magnificent science of biology, but to do so we have to write: “Asterix” when we mean “Gene” ... oh well.

84.4 What terms and concepts does OMIM know about?

The `mimTitles.txt` file list the terms in the OMIM database. Think of it as table of contents. It connects a so called MIM number to a concept.

```
# Copyright (c) 1966-2016 Johns Hopkins University. Use of this file adheres to the
# Generated: 2016-11-05
# Prefix      Mim Number      Preferred Title; symbol Alternative Title(s); symbol(
NULL      100050  AARSKOG SYNDROME, AUTOSOMAL DOMINANT
Percent    100070  AORTIC ANEURYSM, FAMILIAL ABDOMINAL, 1; AAA1  ANEURYSM, ABDOMINAL
Number Sign  100100  PRUNE BELLY SYNDROME; PBS      ABDOMINAL MUSCLES, ABSENCE OF
NULL      100200  ABDUCENS PALSY
Number Sign  100300  ADAMS-OLIVER SYNDROME 1; AOS1  AOS;; ABSENCE DEFECT OF LIME
Caret     100500  MOVED TO 200150
Percent    100600  ACANTHOSIS NIGRICANS
```

The number of total terms in OMIM:

```
cat mimTitles.txt | grep -v '#' | wc -l
# 24970
```

84.5 How many phenotypes of Mendelian inheritance?

Phenotypes where the molecular origin is known:

```
cat mimTitles.txt | grep 'Number' | wc -l  
# 4869
```

Phenotypes where the molecular origin is unknown:

```
cat mimTitles.txt | grep 'Percent' | wc -l  
# 1615
```

What are the phenotypes of unknown molecular origin:

```
cat mimTitles.txt | grep 'Percent' | head
```

Producing the file:

```
Percent 100070 AORTIC ANEURYSM, FAMILIAL ABDOMINAL, 1; AAA1      ANEURYSM, ABDOMINAL AORTIC  
Percent 100600 ACANTHOSIS NIGRICANS  
Percent 100820 ACHOO SYNDROME AUTOSOMAL DOMINANT COMPELLING HELIOOPHTHALMIC OUTBURST SYNDROME  
Percent 101850 PALMOPLANTAR KERATODERMA, PUNCTATE TYPE III; PPKP3      ACROKERATOELASTOIDES  
Percent 102100 ACROMEGALOID CHANGES, CUTIS VERTICIS GYRATA, AND CORNEAL LEUKOMA      ROSSETT  
Percent 102150 ACROMEGALOID FACIAL APPEARANCE SYNDROME AFA SYNDROME;; THICK LIPS AND ORAL  
Percent 102300 RESTLESS LEGS SYNDROME, SUSCEPTIBILITY TO, 1; RLS1      ACROMELALGIA, HEREDITARY  
Percent 102350 ACROMIAL DIMPLES      SUPRASPINOUS FOSSAE, CONGENITAL  
Percent 102510 ACROPECTOROVERTEBRAL DYSPLASIA; ACRPV      F SYNDROME  
Percent 102800 ADENOSINE TRIPHOSPHATASE DEFICIENCY, ANEMIA DUE TO
```

The file can be searched (case insensitive) for diseases:

```
cat mimTitles.txt | egrep -i sickle
```

to produce:

```
NULL    143020 HPA I RECOGNITION POLYMORPHISM, BETA-GLOBIN-RELATED; HPA1      RESTRICTION  
Number Sign      603903 SICKLE CELL ANEMIA
```

The number 603903 is the so-called MIM number, a unique identifier within the framework. To find the gene connected to sickle cell Anemia

```
cat genemap2.txt | grep 603903
```

produces a fairly lengthy (but still single line entry with):

```

chr11 5225465 5227070 11p15.4      141900 HBB   Hemoglobin beta HBB  3043
HBBP1 between HBG and HBD loci      Delta-beta thalassemia, 141749 (3), Autosomal c
Heinz body anemias, beta-, 140700 (3), Autosomal dominant; Hereditary persistence
Autosomal dominant; {Malaria, resistance to}, 611162 (3); Methemoglobinemias, bet
(3), Autosomal recessive; Thalassemia-beta, dominant inclusion-body, 603902 (3);
Hbb-bt,Hbb-bs (MGI:5474850,MGI:5474852)

```

To see just a first few columns:

```
cat genemap2.txt | grep 603903 | cut -f 1-8
```

produces:

```
chr11 5225465 5227070 11p15.4      141900 HBB   Hemoglobin beta
```

84.6 How many phenotypes can be connected to genes?

The 13th column of the `genemap2.txt` files connects a gene to a phenotype. But not all those columns have values:

```
cat genemap2.txt | cut -f 13 | wc -l
# 16086
```

```
cat genemap2.txt | cut -f 13 | egrep '.+' | wc -l
# 4996
```

Out of 16086 genes in this file just 4996 have a non-zero field for phenotypes.

84.7 How many unique pairs of gene and phenotypes are there?

As it turns out this is unexpectedly difficult to extract. Columns 6 and 13 contain what we need, but the latter is a free text format from which each 6 digit MIM number needs to be parsed out.

```
cat genemap2.txt | grep 603903 | cut -f 13
```

The output of which is:

84.7. HOW MANY UNIQUE PAIRS OF GENE AND PHENOTYPES ARE THERE?667

Delta-beta thalassemia, 141749 (3), Autosomal dominant; Erythremias, beta- (3); Heinz body anemias, beta-, 140700 (3), Autosomal dominant; Hereditary persistence of fetal hemoglobin, 141749 (3), Autosomal dominant; {Malaria, resistance to}, 611162 (3); Methemoglobinemias, beta- (3); Sickle cell anemia, 603903 (3), Autosomal recessive; Thalassemia-beta, dominant inclusion-body, 603902 (3); Thalassemias, beta-, 613985 (3)

What we need from this are the numbers 141749, 140700 and preferably a file that lists them like so:

```
603903 141749
603903 140700
603903 141749
...
.
```

The above roadblock is one of those typical moments when a bioinformatician needs to write their simple code to get them past the obstacle. There are many ways to solve this problem, for example with an `awk` program that would look like this:

```
# Split input by tabs.
# Format output by tabs.
BEGIN { FS = OFS = "\t" }

{
    # Gene MIM number is column 6.
    genemim = $6
    geneid = $8

    # The phenotypes are in column 13.
    split($13, elems, " ")

    # Go over each element and print the matches to numbers.
    for (i in elems){
        word = elems[i]

        # MIM numbers have exactly 6 digits.
        if (word ~ /^[0-9]{6}/ ){
            print genemim, geneid, word
        }
    }
}
```

```
}
```

Running this code on the entire data and tabulating at the end. How many disease/phenotype entries are there:

```
cat genemap2.txt | awk -f prog.awk | wc -l
# 6964
```

So the 4996 annotated genes produce a total of 6964 phenotypes.

84.8 What genes have the most annotated phenotypes?

These are the genes where mutations appear to cause the most phenotypes:

```
cat genemap2.txt | awk -f prog.awk | cut -f 1,2 | sort | uniq -c | sort -rn | head
```

produces:

```
15 120140 Collagen II, alpha-1 polypeptide
14 134934 Fibroblast growth factor receptor-3
12 176943 Fibroblast growth factor receptor-2 (bacteria-expressed kinase)
12 150330 Lamin A/C
11 601728 Phosphatase and tensin homolog (mutated in multiple advanced cancers 1)
11 191170 Tumor protein p53
11 171834 Phosphatidylinositol 3-kinase, catalytic, alpha polypeptide
11 109270 Solute carrier family 4, anion exchanger, member 1 (erythrocyte membrane)
10 605427 Transient receptor potential cation channel, subfamily V, member 4 (vanilloid receptor 3)
10 300017 Filamin A, alpha (actin-binding protein-280)
```

So there are 15 entries in this file for gene with MIM number 120140 that corresponds to gene *Collagen II, alpha-1 polypeptide*. What are these phenotypes?

```
cat genemap2.txt | awk -F '\t' '$6 ~ /120140/ { print $13 } '
```

That produces

```
Achondrogenesis, type II or hypochondrogenesis, 200610 (3), Autosomal dominant; Achondroplasia, 608805 (3), Autosomal dominant; Czech dysplasia, 609162 (3), Autosomal dominant; myopia and deafness, 132450 (3), Autosomal dominant; Kniest dysplasia, 156550 (3); Legg-Calve-Perthes disease, 150600 (3), Autosomal dominant; Osteoarthritis with rickets, 605427 (3), X-linked recessive
```

84.8. WHAT GENES HAVE THE MOST ANNOTATED PHENOTYPES?669

Autosomal dominant; Otospondylomegaepiphyseal dysplasia, 215150 (3), Autosomal recessive dysplasia, Torrance type, 151210 (3), Autosomal dominant; SED congenita, 183900 (3), Auto type, 184250 (3), Autosomal dominant; Spondyloepiphyseal dysplasia, Stanescu type, 61658 Spondyloperipheral dysplasia, 271700 (3), Autosomal dominant; Stickler syndrome, type I, n Autosomal dominant; Stickler syndrome, type I, 108300 (3), Autosomal dominant; Vitreoret epiphyseal dysplasia (3)

Chapter 85

Why simulating reads is a good idea

Simulating data allows us to evaluate the performance of our tools and techniques. By knowing what the data contains, we can verify that the tools do indeed operate as we expect them to.

When coupled with recipes that automate an entire workflow data simulations allows you to understand the whole process.

We recommend that you start exploring all bioinformatics methods with simulated data to see the strengths and weaknesses of each. Unfortunately, while data simulators are essential for learning data analysis, this domain of bioinformatics is neglected with quite a few abandoned software. We tried to collect a few practical examples though each has shortcomings.

85.1 What types of data simulators exist?

There are simulators available for the whole genome, exome, RNA-seq and chip-seq data simulation. Some tools run from command line; others have graphical user interfaces.

- `wgsim/dwgsim` simulates obtaining sequencing reads from whole genomes.
- `msbar` EMBOSS tool can simulate mutations in a single sequence.

- `biosed` EMBOSS tool provides control on what mutations we introduce.
- `ReadSim` is a sequencing read simulator to target long reads such as PacBio or Nanopore.
- `Art` is the most sophisticated and up to date simulator that attempts to mimic the errors that sequencing instruments produce.
- `Metasim` simulates reads from metagenomes (from a diverse taxonomical composition present in different abundances).
- `Polyester` an RNA-Seq¹ measurement simulator

85.2 Are the simulations realistic?

The level of “realism” depends on what the simulator aims to capture. Some properties of the sequencing are easier to model, but there will always be features that are unique to real data that no simulator can fully capture.

In general, data simulators are best suited for exploring how various events may manifest themselves within sequencing data. For example: suppose we have a “large scale inversion” what will that data look like when aligned against the non-inverted reference genome.

85.3 What is `wgsim` and `dwgsim`?

`wgsim` and `dwgsim` are tools for simulating sequence reads from a reference genome. The tools can simulate diploid genomes with SNPs and insertion/deletion (INDEL) polymorphisms, and affect reads with uniform sequencing errors.

`wgsim` is easier to install but it only produces the mutations in text format that is harder to visualize. `dwgsim` on the other hand will produce VCF files in addition to the text file.

¹<http://bioconductor.org/packages/release/bioc/vignettes/polyester/inst/doc/polyester.html>

85.4 How do I simulate sequencing data with `wgsim`?

```
# Get a reference genome.
efetch -db=nuccore -format=fasta -id=AF086833 > genome.fa

# Simulate a read pair
wgsim -N 1000 genome.fa sim1.fq sim2.fq > mutations.txt
```

The commands above will create the following files:

- First in pair: `sim1.fq`
- Second in pair: `sim2.fq`
- Mutations as a text file: `mutations.txt`

85.5 How do I simulate sequencing data with `dwgsim`?

You may install `dwgsim` as instructed on the page [Installing `wgsim` and `dwgsim`](#):

```
# Get a reference genome.
efetch -db=nuccore -format=fasta -id=AF086833 > genome.fa

# Simulate sequencing reads from the reference genome.
dwgsim -N 1000 genome.fa sim
```

This will create a number of files:

- First in pair: `sim.bwa.read1.fastq`
- Second in pair: `sim.bwa.read2.fastq`
- Mutations as text file: `sim.mutations.txt`
- Mutations as a VCF file: `sim.mutations.vcf`

You may align the paired-end reads against a genome to explore aligner behavior.

85.6 How do I mutate a sequence with msbar?

The `msbar` EMBOSS tool was designed to generate mutations in sequences. It can produce point mutations with the flag:

```
-point Types of point mutations to perform
  0 (None);
  1 (Any of the following)
  2 (Insertions)
  3 (Deletions)
  4 (Changes)
  5 (Duplications)
  6 (Moves)
```

Or block mutations with the flags

```
-block Types of block mutations to perform
  0 (None)
  1 (Any of the following)
  2 (Insertions)
  3 (Deletions)
  4 (Changes)
  5 (Duplications)
  6 (Moves)
```

Block mutations are a series of consecutive :mutations that span from ranges specified with `-minimum` and `-maximum`.

Get a 50bp query from the reference file and introduce a point mutation:

```
# Get a reference genome.
efetch -db=nuccore -format=fasta -id=AF086833 > genome.fa

# Extract the first 50 bases as a query.
cat genome.fa | seqret -filter -send 50 > query.fa

# Introduce a point deletion.
cat query.fa | msbar -filter -point 3 > mutated.fa

# Visualize the output.
```

```
global-align.sh query.fa mutated.fa
```

on our system this produces the following (the mutations are random so you will get something different):

CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTTGTGTGCGAATA
||||| ||||| ||||| ||||| ||||| ||||| ||||| |||||
CGGACACACAAAAAGAAAGAAGAATTTTA-GATCTTTGTGTGCGAATA

Mutate the query to contain a 2 base block mutation:

```
cat query.fa | msbar -filter -block 3 -minimum 2 -maximum 2 > mutated.fa
```

The alignment now shows:

CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTTGTGTGCGAATA
||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| |||||
CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTTGTGT--GAATA

We can introduce more than one mutation with the `-count` parameter:

```
cat query.fa | msbar -filter -count 2 -block 3 -minimum 2 -maximum 2 > mutated.fa
```

one result could be:

CGGACACACAAAAAGAAAGAATTTTAGGATCTTTGTGTGCGAATA
||||| ||||| ||||| ||||| |||||
CGGACACACAAAA--AAAGAAGAATTTTAGGA--TTTTGTGTGCGAATA

Since the tool selects mutations randomly, when choosing substitutions it is possible to mutate the sequence back to itself i.e. replacing an A with another A

One (serious) flaw of the `msbar` tool is that it does not inform us what changes it has made. Hence, it is a bit harder to use it in a context of validating a variant caller.

85.7 How do I simulate mutations with biosed?

```
# Get a reference genome.  
efetch -db=nuccore -format=fasta -id=AF086833 | head -3 > genome.fa
```

The file contains:

```
>AF086833.2 Ebola virus - Mayinga, Zaire, 1976, complete genome
CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTGTGCGAATAACTATGAGGAAGATTAATAA
TTTCCTCTCATTGAAATTATCGGAATTAAATTGAAATTGTTACTGTAATCACACCTGGTTGTT
```

Now replace the first ten bases with biosed as:

```
cat genome.fa | biosed -filter -target CGGACACACA -replace GGGGGGGGGG
```

produces:

```
>AF086833.2 Ebola virus - Mayinga, Zaire, 1976, complete genome
GGGGGGGGGGAAAAGAAAGAAGAATTTTAGGATCTTGTGCGAATAACTATGAGGA
AGATTAATAATTTCCTCTCATTGAAATTATCGGAATTAAATTGAAATTGTTACTG
TAATCACACCTGGTTGTT
```

It also will slightly reformat the fasta file; the lines are wrapping at a shorter length.

85.8 How do I simulate reads with `readsim`?

The `readsim` program has a large variety of options, it claims to be able to produce realistic errors that correspond to instruments, it can account for higher ploidy (to simulate variants from populations) and so on.

Manual: <https://sourceforge.net/p/readsim/wiki/manual/>

The program will only work with Python 2.

```
# Get a reference genome.
efetch -db=nuccore -format=fasta -id=AF086833 > genome.fa

# Simulate pacbio technology reads into a file named reads.fq
readsim.py sim fq --ref genome.fa --pre reads --tech pacbio
```

85.9 How do I simulate reads with `art`?

Manual: <http://www.niehs.nih.gov/research/resources/software/biostatistics/art/>

The `Art` suite is the most sophisticated simulator that we know of.

```
art_illumina -ss HS25 -sam -i genome.fa -p -l 150 -f 20 -m 200 -s 10 -o reads
```

The output contains even an “optimal” SAM alignment file for the simulated reads that we can compare against:

Quality Profile(s)

```
First Read: HiSeq 2500 Length 150 R1 (built-in profile)
First Read: HiSeq 2500 Length 150 R2 (built-in profile)
```

Output files

FASTQ Sequence Files:

```
the 1st reads: reads1.fq
the 2nd reads: reads2.fq
```

ALN Alignment Files:

```
the 1st reads: reads1.aln
the 2nd reads: reads2.aln
```

SAM Alignment File:

```
reads.sam
```

Chapter 86

Visualizing large scale genomic variations

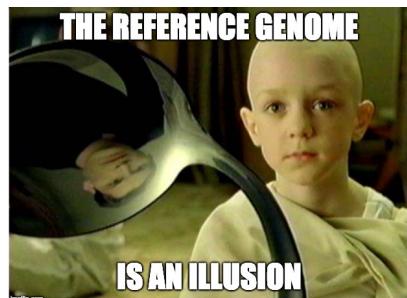
One of the most unexpectedly challenging tasks of a bioinformatics visualization is correctly interpreting the effects that large-scale genomic variations have on alignments.

86.1 The reference genome is not “real”

The confusion starts right at the terminology – we call one of our sequences the “reference” and we compare our data to it. The word “reference” implies that we are comparing against a “ground truth” – but it is the other way around. It is our sequenced data that exists; the reference is often not even a “real” sequence. For example the human reference is built from a pooled sample from several donors. It almost certainly the case that the resulting human reference does not fully match either of the donors; instead, it is a mosaic of them all.

It is also very likely that the human reference genome is not even a viable - meaning that if the sequence were “implanted” into a human cell, it would not function. Now, knowing that, think about the repercussions of this for a second. Scientists are using a non-existing, potentially non-viable human reference to characterize all other “real” human genomes. And when we compare two real genomes, we do so by comparing them via this third, non-

real one through a process that is not “transitive” - knowing how A vs B and how A vs C are different from one another is not sufficient to characterize how B and C are different from one another.



I found it useful to think of the reference genome as a “mirror” in which we “reflect” our real data to make it visible to us. It is an imperfect mirror, and what we see is often in some sense the opposite of reality.

To run the software for this chapter prepare some datasets first:

```
mkdir -p refs
efetch -db=nuccore -format=fasta -id=AF086833 > refs/reference.fa
bwa index refs/reference.fa
```

86.2 Which alignments are informative?

When it comes to structural variations the main rule to remember:

Alignments that **cross over** the variation carry the most information on the nature of the change.

Alignments that are entirely contained within a continuous segment do not know about what is “outside” of that segment, and hence are less informative. The very fact that paired-end sequencing does not fully cover the fragment and has “holes” in the middle becomes, in this case, is quite useful. It does two things for us:

1. It allows the read to cover a longer distance.
2. The span and orientation of the pairs will indicate the type and direction of variation.

It is also worth noting that typically the vast majority of alignments will fall outside of these so-called junctions – hence, only a small subset of the reads will carry useful information.

86.3 What would perfect data look like?

Ask any bioinformatician what ideal, “perfect” data should look like - chances are you’ll get very different answers. We practitioners never see ideal data – we don’t quite know how even to define it. For this book, we wrote a simple Python script called the `perfect_coverage.py` that simulates sequencing every 500bp long fragment of a genome exactly once, with no errors, with an Illumina paired-end protocol.

```
# Get the script
curl -O http://data.biostarhandbook.com/align/perfect_coverage.py

# Generate perfect data.
cat refs/reference.fa | python perfect_coverage.py
```

The code above produces the files `R1.fq` and `R2.fq`. Align these reads with.
`bwa mem refs/reference.fa R1.fq R2.fq | samtools sort > perfect.bam`
`samtools index perfect.bam`

Visualize the output in IGV. The image below is what we saw:



Figure 86.1

Note the coverages at the beginning and end of the genome. These drops are the so-called *edge effects* and will show up even in ideal data. In essence, even

under ideal conditions there will be less coverage at the ends of a chromosome because fewer reads can overlap. The range where the effect will show up is as long as the fragment size – 500bp in this case.

Since a chromosome has only two ends and the fragments are typically much smaller than the sequence itself these artifacts generally are ignored. But that for an analysis that measures lots of short sequences (like an RNA-Seq experiment) there may be tens or hundreds of thousands of transcripts, and each transcript will have its ends affected. Moreover, transcripts have varying lengths whereas the edge effect is fragment size dependent. Thus, shorter transcripts will be more affected and will get less coverage than expected.

Dealing with edge effects in transcripts is more complicated, often solved by introducing a new concept called “effective length normalization” – treating all transcripts as if they were shorter than they are in reality. We’ll discuss this further in the RNA-seq section. It is important to note that the effect is present even in this idealized case.

86.4 What would realistic and useful data look like?

The data simulation methods are explained in more detail in Why simulating reads is a good idea. For the rest of the chapter we will repeat the following steps:

1. We will make a copy of our reference file. We will call this the `genome.fa` and we will use it as the source of our simulations.
2. We will modify our genome file, generate reads from it, and compare it to the reference.

Initially, the `reference.fa` and `genome.fa` files are the same. We will mimic evolution by modifying `genome.fa`, then we will align data obtained from it against `reference.fa`. The process of simulating and aligning can be done step-by-step or via a recipe. When using a recipe, comment out the steps that obtain or index the reference as those don’t change.

```
cp refs/reference.fa genome.fa
wgsim -N 10000 genome.fa r1.fq r2.fq
```

Now run the alignment and bam file creation steps. The result will look like:

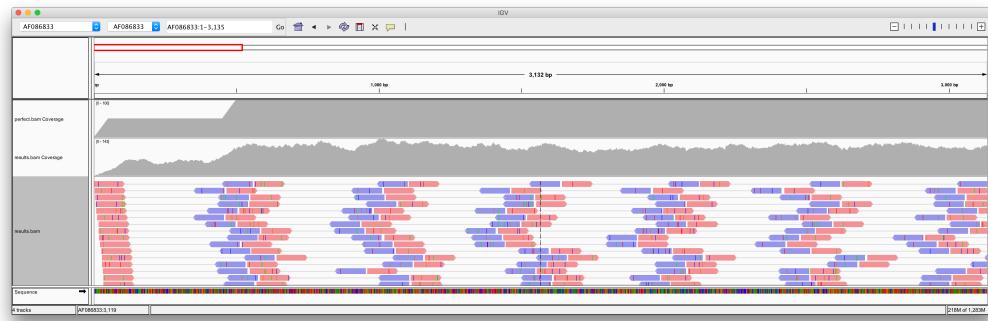


Figure 86.2

The is more realistic data than our perfect alignment. We can observe the stochasticity of the coverage; it varies across the genome.

86.5 What would a deletion from the genome look like?

Open the `genome.fa` file in a text editor and delete a big chunk of it. Turning it from ABCD to say ACD. You could also perform that from command line like so:

```
cat refs/reference.fa | seqret --filter -sbegin 1 -send 2000 > part1
cat refs/reference.fa | seqret --filter -sbegin 3000 > part2
cat part1 part2 | union -filter > genome.fa
```

Then rerun the alignment. Here is what we get:

This image makes a lot of sense. The parts that we deleted are missing. But note something else: there are read pairs that start on one end and continue on the others. From the fragment (template size) we can figure out how long the deletion is. Note how the fragment is around 1500bp? Since our fragment size should be 500 on average, we can tell that the deletion should be around 1000bp this way it adds up to 1500.

From the above example, we note that the deletion made the fragments look larger. It is because we are “stretching” a shorter real genome over a longer “reference” genome.

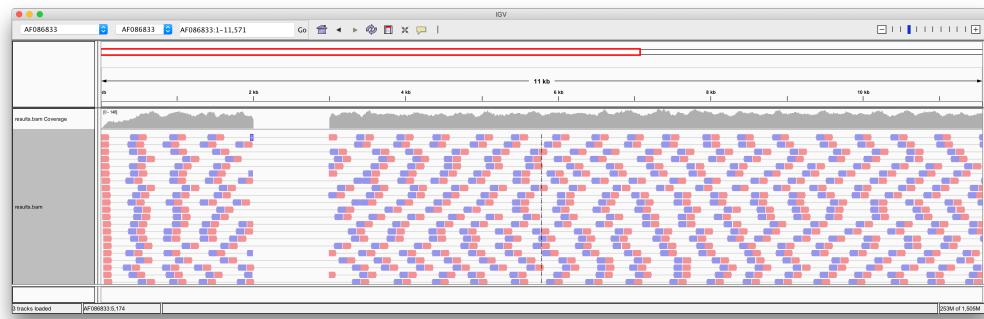


Figure 86.3

86.6 What would copy number variation look like?

For this, add half of the beginning of the genome back to the front of it. Basically imagine the genome is ABC then make it AABC. Again you can do this in an editor or from command line with:

```
cat refs/reference.fa | seqret --filter -sbegin 1 -send 2000 > part1
cat refs/reference.fa | seqret --filter -sbegin 1 -send 2000 > part2
cat part1 part2 refs/reference.fa | union -filter > genome.fa
```

These events are called duplications or more generically copy number variation.

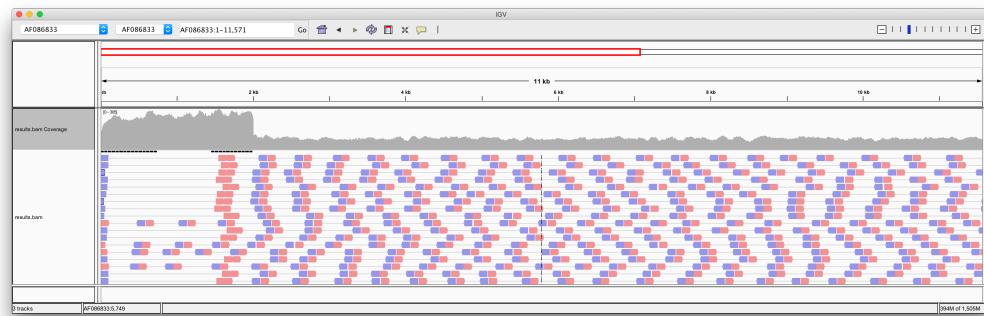


Figure 86.4

86.7 How can you tell when parts of the genome are swapped?

Now we modify our genome to go from ABCD to CDAB.

```
cat refs/reference.fa | seqret --filter -send 5000 > part1
cat refs/reference.fa | seqret --filter -sbegin 5000 > part2
cat part2 part1 | union -filter > genome.fa
```

How will this data align? The image that we see has more subtle changes:

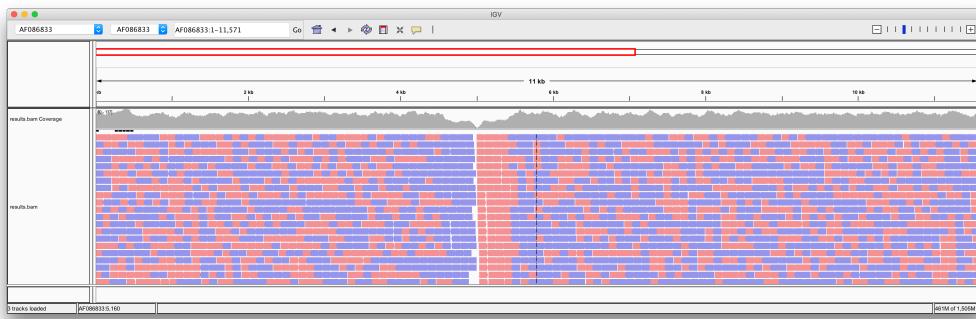


Figure 86.5

It is only when we fully expand the pairs that we start to understand the complexity of it all.

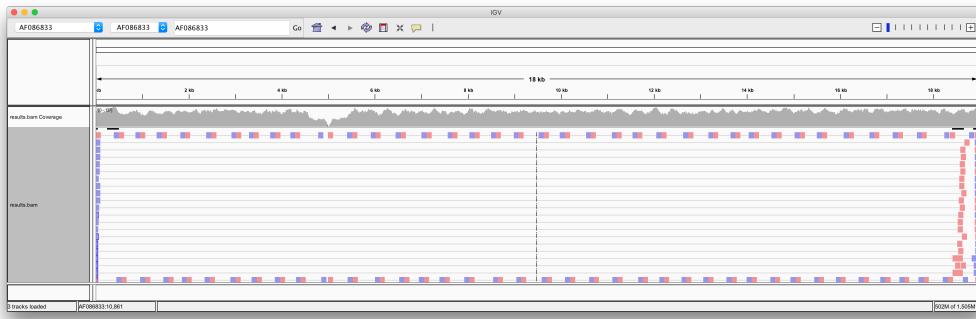


Figure 86.6

86.8 What if the genome contains new regions?

What if would turn our genome from ABCD to ABFCD. How will this show up in our data? Let's insert a sequence of a different virus, in this case, HIV-1 into our genome

```
cat refs/reference.fa | seqret --filter -send 1000 > part1
efetch -db nuccore -id NC_001802.1 -format fasta -seq_stop 1000 > middle
cat refs/reference.fa | seqret --filter -sbegin 1000 > part2
cat part1 middle part2 | union -filter > genome.fa
```

The read pairs that come from the novel genome will not align anywhere in the reference. These reads will stay unmapped. For read pairs where one of the pairs comes from the reference and the other from the novel genome, the matching one will be aligned and the mate lost. These are called “orphan” alignments. From the “orphans” we can see that a new region must be present.

Also, reads falling over the junctions may be soft-clipped all at the same coordinate. When we look at the alignment in paired-end mode a large number of orphan reads show up:

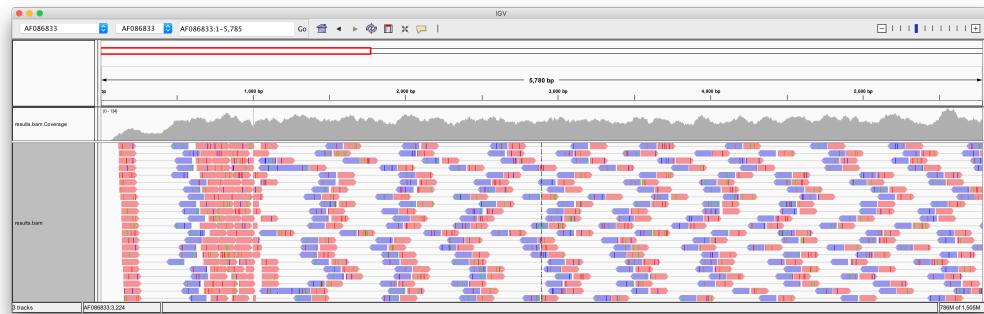


Figure 86.7

86.9 What if we reverse complement parts of the genome?

In this example, we reverse complement some section of genome, but keep the rest the same:

```
cat refs/reference.fa | seqret --filter -sbegin 1 -send 1000 > part1
cat refs/reference.fa | seqret --filter -sbegin 1000 -send 2000 -sreverse1 > part2
cat refs/reference.fa | seqret --filter -sbegin 2000 > part3
cat part1 part2 part3 | union -filter > genome.fa
```

The resulting image is best described as a “quite the head-scratcher”. Bioinformaticians look at one another and say: “What the heck is THAT?”. Since we know what we put in we can explain what we see – had this been an unknown reorganization, figuring out what happened would have been a lot more difficult.

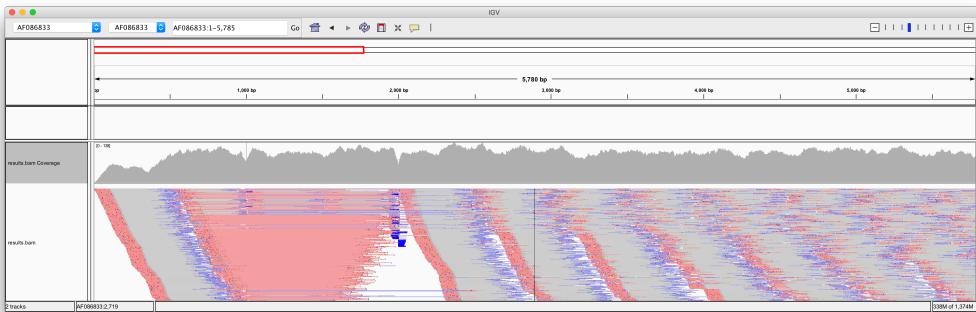


Figure 86.8

Part XX

Variation Calling

Chapter 87

Introduction to variant calling

Variant calling is the process of identifying and cataloging the differences between the observed sequencing reads and a reference genome.

87.1 What is the process for variant calling?

The process is relatively straightforward as long as you recognize that the process is a series of interconnected yet still separate and independent tasks, rather than a single one-step analysis.

At the simplest level variants are usually determined (“called”) from alignments (BAM) files. A typical process involves the following:

1. Align reads to the reference.
2. Correct and refine alignments (optional).
3. Determine variants from the alignments.
4. Filter the resulting variants for the desired characteristics.
5. Annotate filtered variants.

Let us mention that each of the steps above allows for substantial customizations that in turn may significantly alter the outcomes in the later stages.

In the variant calling process you will run into terminologies that are not always used consistently. We will attempt to define a few of these terms while making a note that these definitions are not “textbook” definitions

designed to accurately capture the proper terminology of Genetics as science. Our definitions are what people ‘call’ certain features of their data.

87.2 What is the ploidy?

The ploidy is the number of complete sets of chromosomes in a cell and hence represent the number of possible alleles (variant forms) of the DNA of a **single** cell.

87.3 How many possible alleles will I see?

It is essential that you identify early on the constraints of the problem under study. If all the cells carry the same DNA then the number of alleles will correspond to the ploidy of the organism. If the cells represent a population of N distinct individuals then the maximum possible alleles will be $N \times$ ploidy.

87.4 What is a haplotype?

A haplotype (haploid genotype) is a group of alleles in an organism that are (or may be) inherited together from a single parent. Variants that occur on the same DNA molecule form a haplotype. We also call such variants as “phased” variants.

87.5 What is a genotype?

We found that the word “genotype” is used quite inconsistently, often with wildly different meanings. The simplest way to think about a genotype is as the collection of all known alleles. Both genotypes and haplotypes may be complete or incomplete (partially known).

The word “genotyping” usually means that we can assign one or more sequence reads to a known group of variants (genotype).

87.6 What is the best method to call variants?

There is no universal rule, method or protocol that would always produce correct answers or guarantee some range of optimality. Moreover, any method, even a simple evaluation or even an incorrect rationale may be able to create seemingly right calls when the signal in the data is robust and unambiguous.

Much ink and effort are spent generating and comparing results across massive datasets; you can find these as being labeled as “concordance.” of SNP callers. Alas, in our finding none of these reports ever provide a clear picture or even advice on which tool to pick or which works better under what circumstance.

This is not to say that we don’t have access to reasonably reliable variant callers. Scientists have been working feverishly in this field, and radical improvements have been and are continually being made.

What you need to recognize is that the task of describing differences may in some cases turn out to be far more complicated than anticipated - yet there is no way to tell - apriori - which category a given study will fall into.

87.7 What are the most commonly used variant callers?

There are “generic” variant callers:

- bcftools
- FreeBayes
- GATK
- VarScan2

Besides, there are specialized software designed to account for specific properties of the data, such as somatic mutations, germline mutations, family trios, and many others. Specialized tools typically can make use of other features of the data and produce more accurate variant calls.

87.8 How are variants represented?

The “standard” representation for variants is the so-called VCF format described in a subsequent chapter: VCF: The Variant Call Format. In this section we jump ahead and show you how to create VCF files, then later we explain how the format represents the information.

87.9 How do I generate VCF files?

Typically a VCF file is a result of running a “variant caller” (or “SNP caller” as some people call it) on one or more BAM alignment files. The result of running the variant caller will be a VCF file that contains a column for each sample. Samples may be stored in different BAM files or may a single BAM file may include multiple samples tagged via read-groups.

We have used the `wgsim` simulator before, another simulator based on `wgsim` called `dwgsim` (install with that command `conda install dwgsim`) also produces a VCF file that lists the mutations

As before there is a lot of value in generating known mutations as it allows us to evaluate and benchmark different variant calling approaches. Below we show how to generate a mutated genome then use a variant caller to predict the known mutations.

```
# Reference accession numbers.  
ACC=AF086833  
  
# Create the directory for reference file.  
mkdir -p refs  
  
# The name of the reference.  
REF=refs/${ACC}.fa  
  
# The name of the BAM file  
BAM=align.bam  
  
# Obtain the reference genome.  
efetch -db=nuccore -format=fasta -id=$ACC > $REF
```

```
# Create a bwa index for the reference.
bwa index $REF

# Create a samtools index for the reference.
samtools faidx $REF

# Simulate reads from the reference file.
dwgsim $REF simulated
```

Running `dwgsim` as above will produce a file called `simulated.mutations.vcf`. Take a moment to investigate this file (your file will be different since it will contain different random changes)

```
##fileformat=VCFv4.1
##contig=<ID=AF086833,length=18959>
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=pl,Number=1,Type=Integer,Description="Phasing: 1 - HET contig 1, #2 - HET contig 2">
##INFO=<ID=mt,Number=1,Type=String,Description="Variant Type: SUBSTITUTE/INSERTION/DELETION">
#CHROM POS ID REF ALT QUAL FILTER INFO
AF086833 1093 . T C 100 PASS AF=0.5;pl=2;mt=SUBSTITUTE
AF086833 1195 . A T 100 PASS AF=0.5;pl=1;mt=SUBSTITUTE
AF086833 2566 . G T 100 PASS AF=0.5;pl=2;mt=SUBSTITUTE
AF086833 3345 . A AC 100 PASS AF=0.5;pl=2;mt=INSERT
```

The file above is a VCF file discussed in the section: VCF: The Variant Call Format.

Let's now generate sequencing reads from the mutated genome.

In this example we will use the software package called `bcftools`. This `bcftools` suite has a very broad number of use cases beyond variant calling that we will explore in other chapters.

```
# This is the data naming generated by dwgsim.
R1=simulated.bwa.read1.fastq
R2=simulated.bwa.read2.fastq

#
# Generate the alignment from the data simulated above.
#
```

```
bwa mem $REF $R1 $R2 | samtools sort > $BAM

# Index the BAM file
samtools index $BAM

# Compute the genotypes from the alignment file.
bcftools mpileup -Ovu -f $REF $BAM > genotypes.vcf

# Call the variants from the genotypes.
bcftools call -vc -Ov genotypes.vcf > observed-mutations.vcf
```

The resulting file called `observed-mutations.vcf` may be visualized in IGV and compared to the file `simulated-mutations.vcf`:

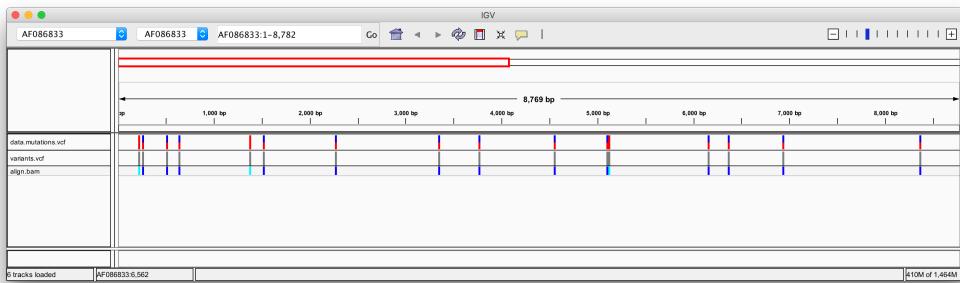


Figure 87.1

Study these outputs, perform multiple runs, tune `dwgsim` to produce more variations, perhaps larger errors and so on. You will develop a sense of what types of mutations are reliably called and which ones are less so.

Experiment with different genomes, repetitive and more redundant genomes will produce data that produces less accurate results.

Chapter 88

Variant calling example

We demonstrate variant calling for the data from Redo: Genomic surveillance elucidates Ebola virus origin.

88.1 Which differences should I be looking at?

Remember that “variants” are always defined relative to “something”. Variants do not exist on their own, only relative to another, similar sequence. You decide which comparison needs to be made. Sometimes there is a commonly used “reference” - other times it is up to you to decide which comparison is the most appropriate.

Let us investigate the genomic changes of the 2014 outbreak relative to the sequence of the 1976 outbreak stored under the accession number AF086833. We will now have to make use of all the techniques that we’ve learned so far to find out what has changed in 38 years. We will do the following:

1. Obtain and prepare the reference genomes
2. Obtain the sequencing data
3. Perform the alignments
4. Perform the variant calling

88.2 How do I prepare the reference genome for variant calling?

Since there are quite many steps, the workflow is well suited for automation and scripting. The following code is structured as a script that you could create as a “starter variation calling” script.

First, we set up the filenames as variables. Doing so helps reuse the code for other data and makes the steps shorter and easier to see what file goes where:

```
# Reference genome accession number.  
ACC=AF086833  
  
# SRR run number.  
SRR=SRR1553500  
  
# The directory to store the reference in.  
mkdir -p refs  
  
# The name of the file that stores the reference genome.  
REF=refs/$ACC.fa  
  
# The resulting alignment file name.  
BAM=$SRR.bam
```

Obtain the sequences:

```
# Get the reference sequence.  
efetch -db=nuccore -format=fasta -id=$ACC | seqret -filter -sid $ACC > $REF  
  
# Index reference for the aligner.  
bwa index $REF  
  
# Index the reference genome for IGV  
samtools faidx $REF
```

88.3 How do I obtain the sequencing data?

We will use `fastq-dump` to access data deposited in SRA:

```
# Get data from an Ebola sequencing run.
fastq-dump -X 100000 --split-files $SRR
```

88.4 How do I align sequencing data against a reference?

We may use different aligners; we chose `bwa` below:

```
# Shortcut to read names.
R1=${SRR}_1.fastq
R2=${SRR}_2.fastq

# Tag the alignments. GATK will only work when reads are tagged.
TAG="@RG\tID:$SRR\tSM:$SRR\tLB:$SRR"

# Align and generate a BAM file.
bwa mem -R $TAG $REF $R1 $R2 | samtools sort > $BAM

# Index the BAM file.
samtools index $BAM
```

88.5 How do I use `bcftools` to call variants?

Samtools/BCFTools can be used to call SNPs in the following way:

```
# Determine the genotype likelihoods for each base.
bcftools mpileup -Ovu -f $REF $BAM > genotypes.vcf

# Call the variants with bcftools.
bcftools call --ploidy 1 -vm -Ov genotypes.vcf > variants1.vcf
```

You may also run both commands on one line to avoid having to store the intermediate, often fairly large genotype file:

```
bcftools mpileup -Ou -f $REF $BAM | bcftools call --ploidy 1 -vm -Ov > variants1.vcf
```

The genomes have changed a lot, view the resulting VCF file in IGV:



Figure 88.1

88.6 How do I use the FreeBayes variant caller?

On this same data the FreeBayes variant caller may be invoked as:

```
freebayes -f $REF $BAM > variants2.vcf
```

88.7 What is GATK (The Genome Analysis Toolkit)?

GATK is not just a variant caller - it is a toolset of epic complexity that offers features and functionality that most of us won't ever understand (or need for that matter).

The Genome Analysis Toolkit may be one of the best variant callers and would be our first choice especially for higher order organisms with complex genomes (human, mouse etc) had it not been for the unnecessarily awkward usage and incredible mental overhead that comes with it.

When you use GATK you have to submit to their will, jump through hoops, bend the knee, drink the kool-aid all at the same time. Every step ends up overcomplicated with the many small inconsistencies adding up to create perhaps the worst user experience of all bioinformatics.

In our opinion, GATK is the worst offender when it comes to capricious and seemingly ludicrous behaviors, tedious and odious program interfacing with people. Now, if you learn how to put up with it all other tools will be a joy to use - you will burst into signing every time it is not GATK that you have to use.

To their credit, the GATK team maintains a forum where they do their best to address the myriad of legitimate questions that their user-unfriendly tool generates:

- Main website: <https://software.broadinstitute.org/gatk/>

GATK also has a “best practices” site that presents recommended workflows for variant calling using GATK.

- GATK best practices: <https://software.broadinstitute.org/gatk/best-practices/>

Good luck comrade, you are going to need it!

88.8 How to install GATK?

Thanks to the many attempts to monetize a product that was developed primarily with public resources GATK cannot be installed as other tools. . You will need to download GATK as instructed here separately:

- <https://bioconda.github.io/recipes/gatk/README.html>

once you download the GATK program you will have to install the conda wrapper with:

```
conda install gatk
```

then call:

```
gatk-register downloaded-stuff
```

88.9 Are there multiple versions of GATK?

Oh yes.

Version 3.8 has many features that version 4.0 does not support and vice versa. In addition, the way we make use of GATK version 3.8 is only remotely similar to the way GATK version 4.0 works.



Figure 88.2

Long story short, GATK is not for the faint-hearted and the problems that you have to deal with when using it have nothing to do with bioinformatics. Compare how you run `freebayes` to GATK...

88.10 How do I use GATK?

GATK operate via “tools”. Tools are independent programs that provide specific functionality. These are typically run as

```
gatk SomeToolName -R $REF -I $BAM -O something -X 200
```

Here the tool called `SomeToolName` will be run with the parameters listed on the right.

Besides, a separate program named `picard` (a homage to Capt. Picard of Star Trek Next-Generation) is often required to prepare and manipulate data.

For our example, the Genome Analysis Toolkit could be used with the `HaplotypeCaller` tool as:

```
# Index the reference with samtools.  
samtools faidx $REF
```

```
# It needs yet another so-called Dictionary index.  
picard CreateSequenceDictionary REFERENCE=$REF OUTPUT=refs/$ACC.dict  
  
# Generate the variants.  
gatk HaplotypeCaller -R $REF -I $BAM -O variants3.vcf
```

88.11 How can I improve the quality of variant calls?

Most pipelines and protocols recommend a series of quality filtering steps and processes that will improve the quality of the variant calls. These are too numerous to enumerate here but can be found under titles such as:

- GATK best practices: <https://software.broadinstitute.org/gatk/best-practices/>

You would also need to read a few documents, papers, and manuals that summarize other researcher's experiences with the exact domain that you are interested in. For example, the VarScan2 manual states that when investigating trios (2 parents + child):

the ideal de novo mutation call will have high depth (>20x) in all three samples, with good support in the child (40-50% of reads for autosomal calls) and no variant-supporting reads in either parent.”

Do understand that best-practices merely improve on a process with diminishing returns but cannot, on their own, solve some of the most significant problems of variant calling.

88.12 Can I customize the variant calling process?

Just as with aligners the process of calling variants can be significantly customized. Some parameters (e.g., ploidy), reflect certain expectations of the

88.12. CAN I CUSTOMIZE THE VARIANT CALLING PROCESS? 701

genome and sample collection under study and will need to be set to match those expectations.

Chapter 89

Multi-sample variant calling

89.1 What is multi-sample variant calling

A multi-sample variant call is an extension of a single sample variant call, where all the samples are evaluated at the same time, and the presence and absence of a variant is indicated for all samples in parallel.

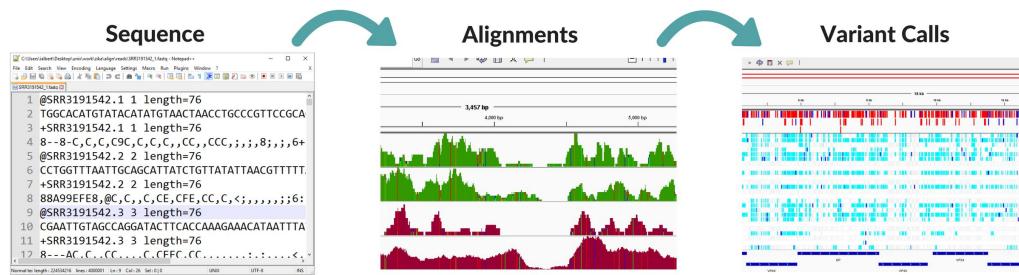


Figure 89.1

89.2 How do I perform multi-sample variant calling process?

There are two ways to perform multi-sample variant calls:

1. Call the variants simultaneously on all BAM files

2. Merge variants called individually in separate VCF files

In general if possible use the first method. Merging VCF files may be fraught by subtle problems, especially whenever different versions of tools created the VCF files

Recall that we found the SRR **runids** for the Ebola project by searching the SRA database with PRJNA257197 and formatting the output as **runinfo**.

If you were to follow the variant calling as presented in Variant calling example you would end up with a BAM file called **SRR1553500**. Place the code (or consult the corresponding recipe) to create other BAM files (one or more) for ids that you found when searching for SRR numbers that belong to the same project. For example:

```
SRR1972917
SRR1972918
SRR1972919
SRR1972920
```

Now you have a BAM file for each of the SRR runs.

```
bcftools mpileup -Ou -f $REF bam/*.bam > genotypes.vcf
bcftools call --ploidy 1 -vm -Ou | bcftools norm -Ov -f $REF -d all - > variants.vcf
```

to produce a multisample variant call that contains the information on all samples:



Figure 89.2

89.3 How do I interpret the visualization?

Read more on how IGV visualizes VCF files below:

- Viewing VCF Files¹

¹http://software.broadinstitute.org/software/igv/viewing_vcf_files

Chapter 90

Variant normalization

Conceptually the “word” variation captures unexpectedly complex information. Once we move past simple, single step changes, the same results could be represented in alternative ways.

90.1 What is variant normalization?

In Unified representation of genetic variants, Bioinformatics 2015¹ the author state:

A genetic variant can be represented in the Variant Call Format (VCF) in multiple different ways. Inconsistent representation of variants between variant callers and analyses will magnify discrepancies between them and complicate variant filtering and duplicate removal.

For example, the same variant may be represented as

GAC --> AC
TGAC --> AC
GACC --> ACC

¹<http://bioinformatics.oxfordjournals.org/content/31/13/2202>

Variant normalization is the process of simplifying the representation of a variant to obey the following rules:

- Represent the variant with as few letters as possible.
- No allele may be zero length.
- Variant must be “left aligned” (this also applies to repetitive regions).

A variant is “left aligned” if it is no longer possible to shift its position to the left while keeping the length of all its alleles constant. For example, an insertion of C into

CCCC

would need to be reported at the leftmost position.

90.2 Should variant callers report normalized variants by default?

Yes and eventually they will do so by default. GATK for example already produces normalized variants. The output of other tools ought to be normalized.

90.3 How do I normalize a variant?

The `bcftools` and the `vt` tools have actions that perform variant normalization.

```
bcftools norm -f $REF samples.vcf
```

and

```
vt normalize -r $REF samples.vcf
```

Chapter 91

The Variant Call Format (VCF)

91.1 What is the VCF format?

The variant call format (VCF) is a data representation format used to describe variations in the genome. A VCF file may contain information on any number of samples, thousands even and can be thought of as a single database that summarizes the final results of multiple experiments in a single file.

As a format, VCF is composed of two sections: a **header** section and a **record** section.

While it is a plain text format, its contents can go from being easy to read and understand to seemingly impossible to visually decipher.

The challenge of reading out variants by eye is (probably) not the format's "fault" - the variations in the genome can have relatively complex representations.

91.2 How important is it to understand the VCF format?

In many cases, it is possible to perform sophisticated analyses without ever directly looking inside a VCF file. Many tools take VCF files as inputs and

produce other reports out of them. At other times a cursory understanding of what a VCF file contains and how the information is laid inside of them out is sufficient to complete an entire analysis.

But just as with the SAM format - understanding the VCF format is essential for anyone that needs to venture off the beaten path. There will be times when crucial information is buried deep inside a VCF file with hundreds of samples. You may have to use toolsets like `bcftools` to filter, combine and subselect VCF files in every which way imaginable. At those times understanding the small details of the VCF format become indispensable.

91.3 What is a VCF header?

VCF header sections are located at the beginning of the VCF files and consists of lines that start with the ## symbols like so:

```
##fileformat=VCFv4.1
##contig=<ID=AF086833,length=18959>
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
```

As variant calling is a more complex operation than alignment, additional information needs to be recorded about the process.

The VCF headers contain the specifications on various terms used throughout the file. For example, from the above header, we learn that AF in the VCF file will be used to describe a variant and it means “Allele Frequency” and seeing AF=0.5 would mean that the allele frequency for a variant was measured to be 0.5.

91.4 What is the difference between the INFO and the FORMAT fields?

The starting word `INFO` in the definition line above means that the ID will describe a variant across **all samples**. The definition from the header applies to (or has been derived from) all samples together and those characterize the variant.

When the word **FORMAT** is used it would mean that, for this ID you will have a different value for each sample.

91.5 What do the words mean?

What does **Allele Frequency** mean precisely? And where is the actual definition of how the value for this term was computed?

Some words are well-defined concepts from related scientific fields such as population genetics and will be relatively easy to find a meaning for. For example, if you did not know beforehand, a Google search could tell you that “allele frequency” is the frequency of a variant in a population. It can be defined as the fraction of all chromosomes that carry the allele.

But there will be terms for which it will be much harder to find a proper explanation. For example:

```
## INFO=<ID=VDB,Number=1,Type=Float,Description="Variant Distance Bias for  
## filtering splice-site artefacts in RNA-seq data (bigger is better)",Version="3">
```

So what is VDB then? It states that it is: *Variant Distance Bias for ## filtering splice-site artefacts in RNA-seq data (bigger is better)*. Ok what does that even mean? Turns out you’d have to do a lot of legwork to figure it out, perhaps read the manual or publication of the variant caller that produced this value. And occasionally, for some terms, you may not be able to find (or understand) the explanation on your own. To some extent, this is the antithesis of what science is. Some number that someone thought is relevant...

Note how even the creator of the VDB ID felt the need to mention that for this value “bigger is better.” That is not all that encouraging from the perspective of a scientist - you are given a number that you don’t even know which way is “better”. You might ask yourself the question, do I even have to know about VDB in the first place? To which the answer usually is, no, you don’t.

Some (many or even most) IDs might be immaterial to the study that you are conducting. Variant callers are very prolific and liberally sprinkle values of all kinds of quantities and measures that the creators of these tools felt that should always be mentioned. Only a few are these will be relevant most of the time you neither need to nor would want to know them.

91.6 What are VCF records?

The record section of a VCF file consists of at least eight tab-delimited columns where the first eight columns describe a variant and the rest of the columns describe the properties of each sample. The 9th column is the **FORMAT** and each column past the 9th will represent a sample.

As mentioned before, but it is worth reiterating, a VCF file may contain any number of sample columns, thousands even and can be thought of as a single database that represents all variations in all samples.

The first nine columns of a VCF file are:

1. **CHROM**: The chromosome (contig) on which the variant occurs
2. **POS**: The genomic coordinates on which the variant occurs. For deletions, the position given here are on of the bases preceding the event.
3. **ID**: An identifier for the variant (if it exists). Typically a dbSNP database if that is known.
4. **REF**: The reference allele on the forward strand.
5. **ALT**: The alternate allele(s) on the forward strand. More than one may be present.
6. **QUAL**: A probability that the **REF/ALT** variant exists at this site. It is in Phred scale, just as the FASTQ quality and the MAPQ field in the SAM file are.
7. **FILTER**: The name of filters that the variant fails to pass, or the value **PASS** if the variant passed all filters. If the **FILTER** value is **.**, then no filtering has been applied to the record.
8. **INFO**: Contains the site-specific annotations represented as **ID=VALUE** format.
9. **FORMAT**: Sample-level annotations as colon separated TAGS.

For example:

```
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT align.bam
AF086833 60 . A T 54 . DP=43 GT:PL 0/1:51,0,48
```

91.7 How to interpret the FORMAT field?

This field specifies the meaning of the numbers in the each column of the sample. The fields are colon : separated and map each field of the **FORMAT** to each value in the sample column. Suppose that the following were columns 9,10 and 11 of a VCF file:

FORMAT	sample1	sample2
GT:PL	0/1:51,0,48	1/1:34,32,0

We interpret it in the following way.

- The variant observed for **sample1** has the values **GT=0/1** and **PL=51,0,48**
- This same variant when observed in **sample2** has the values **GT=1/1** and **PL=34,32,0**

What if you wanted to know what do **GT** and **PL** mean? You'd have to go to the header for their definition, and there you would find

```
##FORMAT=<ID=GT,Number=1>Type=String,Description="Genotype">
##FORMAT=<ID=PL,Number=G>Type=Integer,Description="List of Phred-scaled genotype likelihoods"
```

91.8 What is represented in the REF/ALT columns.

The **REF** column represents the reference allele at position **POS**

The **ALT** column represent **all** variants observed at that site. When there is only a single simple variant, say a SNP at position 60 where an **A** was replaced by a **T** the VCF will look eminently readable. The variant record would state:

```
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT align.bam
AF086833 60 . A T 54 . DP=43 GT:PL 0/1:51,0,48
```

It also tells us that the genotype is **GT=0/1** and that Phred-scaled genotype likelihoods are **PL=51,0,48** (we will explain both concepts in more detail a bit later).

Now when there are two possible variants in the same location, it starts to be a little more complicated. This site now has two alternates at the same site:

```
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT align.bam
AF086833 60 . A T,C 43.2 . DP=95 GT:PL 1/2:102,124,42,108,0,48
```

There are two alternates, and we see from the GT field that our sample was heterozygous for the alternates. Neither of the alleles matched the reference.

Alas, the complexity does not stop there. This is where the VCF format becomes a lot harder to interpret. Previously we only mutated base 60. But what if beyond just the mutation, some members of this population had a three-base deletion at coordinates 58,59 and, 60. Now the our VCF record will show:

```
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT align.bam
AF086833 55 . TGAGGA TGA 182 . INDEL;IDV=42 GT:PL 0/1:215,0,255
AF086833 60 . A C,T 39.8 . DP=126; GT:PL 1/2:99,87,47,86,0,49
```

What is important to note here that even though the deletion started at coordinate 58 is shown at coordinate 55 in the file, but since the both the ALT and REF start with identical bases the TGA difference between them begin only after three bases. So when TGAGGA is replaced by TGA the actual change is the three base deletion starting from 58 and will overlap with coordinate 60. But it is clear that you can't easily see that from the VCF file.

You may even ask yourself why is the variant reported as TGAGGA --> TGA? Why not GAGGA --> GAGGA or even GGA ->? Clearly, we could report these difference in many different ways. You'll be surprised to know that this is still a not fully resolved problem in variant reporting.

There is a method called “variant normalization” that attempts to ensure that variants are reported identically. To explore that and understand how variants are determined and represented see the chapter [NEXT WEEK]

91.9 How are genotypes described?

Whereas the REF and ALT columns show the change, we need to know how many of the copies of the DNA carry the variant or variants. The variant is

encoded in the field called **GT** that indicates the genotype of this sample at this site.

It is constructed out of slash-separated numbers where:

- 0 indicates the **REF** field,
- 1 indicates the first entry in the **ALT** field,
- 2 indicates the second entry in the **ALT** field and so on.

For example for a diploid organism the **GT** field indicates the two alleles carried by the sample:

- 0/0 - the sample is a homozygous reference
- 0/1 - the sample is heterozygous, carrying one of each the **REF** and **ALT** alleles
- 1/2 - would indicate a heterozygous carrying one copy of each of the **ALT** alleles.
- 1/1 - the sample is homozygous for the first alternate
- 2/2 - the sample is heterozygous for the second alternate

and so on. For samples of higher ploidy (multiple copies of chromosomes), there would be as many / characters as there are copies of each chromosome. For example, a tetraploid genotype could be 0/1/0/2.

91.10 What are genotype likelihoods?

The genotype likelihoods are normalized probabilities that describe the likelihood that each of the possible genotypes occurs. The value is normalized to 0. So the most likely genotype is 0, and the rest are computed relative to that. Since it is a statistical measure (a p-value), it is the chance of not observing the genotype hence the “best” value is 0.

So let's take the:

GT:PL 0/1:51,0,48

PL field will contain three numbers, corresponding to the three possible genotypes (0/0, 0/1, and 1/1). What it says the following:

- The support is highest for genotype 0/1 and that is shown as zero.
- The chance that this genotype is 0/0 instead of 0/1 is Phred score 51 meaning 0.000001

- The chance that this genotype is 1/1 instead of 0/1 is Phred score 48 meaning 0.00001

We got the 0.000001 from the formula $10^{-(-51/10)}$ it is the same Phred score that we grew to love (not really) where you drop a digit from it and that's how many numbers you go to the left 40 -> 0.0001

91.11 What is a BCF file?

The BCF format is a binary representation of the VCF format. It has a similar relationship as the SAM/BAM but with the exception that the BCF files are not sorted and indexed and cannot be queried the same way as typical BAM files do.

91.12 What are additional resources?

- VCF Poster: <http://vcftools.sourceforge.net/VCF-poster.pdf>
- VCF short summary: <http://www.htslib.org/doc/vcf.html>
- VCF Specification: <http://samtools.github.io/hts-specs/>
- A detailed description of the VCF: What is a VCF file¹ on the GATK forums.

¹<http://gatkforums.broadinstitute.org/gatk/discussion/1268/what-is-a-vcf-and-how-should-i-interpret-it>

Chapter 92

Filtering information in VCF files

If you produce VCF file, sooner or later you will need to filter this file, to keep or to remove variants that satisfy certain constraints.

Note: In this section we have collected examples that demonstrate the tools and techniques that perform specific tasks. Use these examples as starting points to achieve what you might need.

92.1 How can I filter VCF files?

There is a cottage industry of tools that may be used to manipulate VCF files, from simple pattern matching with `grep`, to `bedtools` that can intersect files by coordinate, to tools that seem to implement a “mini-programming languages” like `bcftools`

Recommended software that can be used for filtering and manipulating VCF files include

- `bcftools`¹ - utilities for variant calling and manipulating VCFs and BCFs
- `SnpSift`² - a toolbox that allows you to filter and manipulate annotated

¹<https://samtools.github.io/bcftools/bcftools.html>

²<http://snpeff.sourceforge.net/SnpSift.html>

files.

- bedtools³ - a powerful toolset for genome arithmetic

Most `bcftools` commands accept VCF, bgzipped VCF and BCF. However, some commands (e.g.:`bcftools query`) require a tabix⁴ indexed and hence `bgzip` compressed file.

SnpSift filtering tool is especially useful if you have a ‘SnpEff’ annotated VCF file.

92.2 How do I get the example data?

Download the VCF file and its index used in the examples with:

```
wget http://data.biostarhandbook.com/variant/subset_hg19.vcf.gz
wget http://data.biostarhandbook.com/variant/subset_hg19.vcf.gz.tbi
```

This file contains variants in chromosome 19:400kb-500kb.

92.3 How do I extract information from VCF in a custom format?

Bcftools query command with the `-f` flag can be used to extract fields from VCF or BCF files. To extract just the CHROM, POS, REF and ALT fields:

```
bcftools query -f '%CHROM %POS %REF %ALT\n' subset_hg19.vcf.gz | head -3
```

to produce:

```
19 400410 CA C
19 400666 G C
19 400742 C T
```

³<https://bedtools.readthedocs.io/en/latest/>

⁴<http://www.htslib.org/doc/tabix.html>

92.4 How do I list all samples stored in a VCF file?

```
bcftools query -l subset_hg19.vcf.gz
```

produces:

```
HG00115
HG00116
HG00117
HG00118
HG00119
HG00120
```

92.5 How do I extract all variants from a particular region?

To specify the region, use `-r` option with region specified in the format
`chr|chr:pos|chr:from-to|chr:from-[,...]`

```
bcftools query -r '19:400300-400800' -f '%CHROM\t%POS\t%REF\t%ALT\n' subset_hg19.vcf.gz
```

This will produce:

```
19 400410  CA  C
19 400666  G   C
19 400742  C   T
```

If you want to output the results as a VCF file, use `filter` or `view` command instead of `query`. Multiple regions can be listed together in a bed file and can be loaded with `-R` flag to extract variants in those intervals.

92.6 How do I extract variants excluding a specific region?

To exclude a specific region, prefix it with `^`. The `-r` option cannot be used in this case instead use `-t`.

```
bcftools query -t '^19:400300-400800' -f '%CHROM\t%POS\t%REF\t%ALT\n' subset_hg19.vcf.gz
```

To produce:

```
19 400819 C G
19 400908 G T
19 400926 C T
```

To exclude multiple regions, compile them into a bed file and use with `-T` and `^`.

```
bcftools query -T ^exclude.bed -f '%CHROM\t%POS\t%REF\t%ALT\n' subset_hg19.vcf.gz
```

```
19 400742 C T
19 400819 C G
19 401013 G A
```

where `exclude.bed` looks like this

```
#cat exclude.bed
19 400300 400700
19 400900 401000
```

92.7 How do I get variants present in all samples?

We can exclude sites with one or more missing genotype and one or more homozygous reference call to extract variant sites that are present in all samples. Bcftools `view` command with `-g,--genotype` option helps to achieve this.

```
bcftools view -e 'GT= ." | GT=="0|0"' subset_hg19.vcf.gz | bcftools query -f '%POS
```

```
402556 0|1 0|1 1|1 1|0 0|1 1|1
402707 0|1 0|1 1|1 1|0 0|1 1|1
402723 0|1 0|1 1|1 1|0 0|1 1|1
```

The output of `bcftools view` command is piped into `query` command to print it in a user-friendly format.

92.8 How do I select INDELS only?

query command with `-v`, `--types` option can be used to select a variant type of interest. Here a site is chosen if any of the ALT alleles is of the type requested. In this case, types are determined by comparing the REF and ALT alleles in the VCF record not INFO tags like INFO/INDEL or INFO/SNP.

Another way to extract a specific variant is to use the information given in INFO tag. For this we can use `-i`, `--include` option along with an expression.

In our example, we do not have any INDEL sites.

```
bcftools view -v indels subset_hg19.vcf.gz | bcftools query -f '%POS\t%TYPE\n' | wc -l
141
```

```
bcftools view -i 'TYPE="indel"' subset_hg19.vcf.gz | bcftools query -f '%POS\t%TYPE\n' | wc -l
140
```

The difference in the two commands is caused by the position 485135 as marked in the INFO/TYPE field as

```
485135 SNP,INDEL
```

Hence when searched with an expression for exact match in the INFO field, this variant won't be included.

Multiple variant types can be selected with `-v` by listing them separated by a comma. To exclude a variant type use `-V`,`--exclude-types`.

92.9 How do I extract per-sample information from a multi-sample VCF file?

FORMAT which is the 9th column in VCF file specifies how to interpret the colon separated values of each sample given in 10th column onwards. These FORMAT tags can be extracted using the square brackets [] operator, which loops over all samples.

92.10 How to print genotype (GT) of all samples at each position.

```
bcftools query -f '%CHROM\t%POS[\t%GT\t]\n' subset_hg19.vcf.gz | head -3
```

Produces:

```
19 400410 0|0      0|0      0|0      0|0      0|0      0|0
19 400666 1|0      0|1      0|1      0|0      0|0      1|1
19 400742 0|0      0|0      0|0      0|0      0|0      0|0
```

If you want to print header too, use -H

```
bcftools query -H -f '%CHROM\t%POS[\t%GT\t]\n' subset_hg19.vcf.gz | head -3
```

92.11 How do I select specific samples?

`view` command with `-s`, `--samples` option can be used to subset a VCF file based on samples.

To select samples HG00115 and HG00118 from the sample vcf file.

```
bcftools view -s HG00115,HG00118 subset_hg19.vcf.gz | bcftools query -H -f '%POS[
```

produces:

```
# [1]POS      [2]HG00115:GT    [3]HG00118:GT
400410  0|0 0|0
400666  1|0 0|0
400742  0|0 0|0
```

92.12 How do I exclude specific samples?

To exclude samples HG00115 and HG00118 from VCF file.

```
bcftools view -s ^HG00115,HG00118 subset_hg19.vcf.gz | bcftools query -H -f '%POS[
```

produces:

92.13. HOW DO I GET VARIANTS FOR WHICH ALLELE COUNT IS ABOVE A SPECIFIC VALUE

```
# [1]POS [2]HG00116:GT [3]HG00117:GT [4]HG00119:GT [5]HG00120:GT
400410 0|0 0|0 0|0 0|0
400666 0|1 0|1 0|0 1|1
400742 0|0 0|0 0|0 0|0
```

92.13 How do I get variants for which allele count is above a specific value?

`view` command with `-c`, `--min-ac` helps to set the minimum allele count of sites to be printed.

The command below removes sites where allele count (AC tag) < 5.

```
bcftools view -c 5 subset_hg19.vcf.gz | grep -v "#" | wc -l
# 185
```

92.14 How do I select sites where all samples have homozygous genotype?

`view` command with `-g`, `--genotype` helps to include or exclude one or more homozygous (hom), heterozygous (het) or missing (miss) genotypes.

```
bcftools view -g ^het subset_hg19.vcf.gz | bcftools view -g ^miss | bcftools query -f '%P'
# 3891
```

In the command above `-g ^het` and `-g ^miss` exclude sites with one or more heterozygous variant or a missing genotype.

92.15 How do I select sites where ‘n’ samples have heterozygous variants?

SnpSift `filter` command below selects sites where 3 or more samples have heterozygous genotypes.

In the command below `snpsift` is an alias to “`java -jar ~/bin/SnpSift.jar`”

```

alias snpsift="java -jar ~/bin/SnpSift.jar"
snpsift filter "countHet() >=3" subset_hg19.vcf.gz |snpsift extractFields - "POS"
will produce

#POS      GEN[*].GT
400666  1|0 0|1 0|1 0|0 0|0 1|1
400941  0|1 1|0 0|1 1|0 0|0 0|0
401235  0|0 0|0 0|1 0|0 0|1 1|0

```

92.16 How do I select variant sites based on quality and read depth?

If we want to extract sites with `QUAL > 50` and `read depth > 5000`, then these conditions can be included in an expression like this.

```
QUAL>50 && DP>5000
```

This can be used with `-i,--include` to extract sites satisfying the expression. The command would then be

```
bcftools query -i 'QUAL>50 && DP>5000' -f '%POS\t%QUAL\t%DP\n' subset_hg19.vcf.gz
```

```

400410  100 7773
400666  100 8445
400742  100 15699

```

92.17 How do I merge multiple VCF/BCF files?

If we had multiple VCF files, then we can merge them into a single multi-sample VCF with bcftools `merge` command.

```
bcftools merge -l samplelist > multi-sample.vcf
```

where sample list is

```
#cat samplelist
sample1.vcf.gz
```

```
sample2.vcf.gz  
sample3.vcf.gz  
....
```

92.18 How do I find variant sites present in all vcf files?

If we have 3 samples and a vcf file for each of them, then to find sites that are common to all 3

```
bcftools isec -p outdir -n=3 sample1.vcf.gz sample2.vcf.gz sample3.vcf.gz
```

n=3 in the command specifies to output sites present in 3 files. The output files will be in a directory named ‘outdir’.

92.19 How do I extract variant sites present in at least 2 files?

```
bcftools isec -p outdir -n+2 sample1.vcf.gz sample2.vcf.gz sample3.vcf.gz
```

n+2 in the command specifies to output sites present in 2 or more files. The output files will be in a directory named ‘outdir’.

92.20 How do I find variants unique to one sample but absent in all other samples?

Bcftools isec command with -C, --complement option output positions present only in the first file but missing in others.

```
bcftools isec -p outdir -C sample1.vcf.gz sample2.vcf.gz sample3.vcf.gz
```

The above command will produce sites unique to sample1.

Chapter 93

Variant annotation and effect prediction

93.1 What is variant annotation?

Variant annotation means predicting the effects of genetic variants (SNPs, insertions, deletions, copy number variations (CNV) or structural variations (SV)) on the function of genes, transcripts, and protein sequence, as well as regulatory regions.

Variant effect annotators typically require additional information from other data sources. The predicted results will depend on the type of other data that is known about the organism or genome under study. The starting points for a variant annotator are the variants (SNPs, insertions, deletions, and MNPs) usually in VCF format.

The results of variant annotation may be reported in text, HTML or pdf format along with an annotated VCF file that now contains additional information on each variant. The text/HTML reports contain substantially more information than the VCF files as this latter format limits the type of content that may be added.

93.2 Are are consequences of “variant effects”?

Whereas the way the word is used seems to imply a “universal” effect, keep in mind that all effects are “relative” to the information that is currently present in a database.

The same way as in an alignment an insertion in one of the sequences could be reported as being a deletion in the other - it all depends on what we choose to be the reference, any effect predicted by an annotator refers to the information already known and stored in the database.

93.3 What kind of effects can variant effect predictors find?

The variant effects (sometimes called consequences) fall into multiple categories:

- Correlate the location of the variant with other genomic annotations (e.g., upstream of a transcript, in the coding sequence, in non-coding RNA, in regulatory regions)
- List which genes and transcripts are affected by the variant.
- Determine consequence of the variant on protein sequence (e.g., stop_gained, missense, stop_lost, frameshift, other property changes)
- Match known variants that may have been found in different projects such as the 1000 Genomes Project.

Several terms are used to express the effect (also called consequence) of variants. The actual definition of each word is part of the Sequence Ontology that we covered in the first chapters. Also, each effect is associated with a somewhat subjective qualifier (LOW, MODERATE, HIGH) that attempts to characterize the severity and impact of the impact on a biological process.

Here are a few potential “effects”:

- **stop_gained:** A sequence variant whereby at least one base of a codon is changed, resulting in a premature stop codon, leading to a shortened transcript, SO:0001587, HIGH

- **inframe_insertion:** An inframe nonsynonymous variant that inserts bases into in the coding sequence, SO:0001821, MODERATE
- **missense_variant:** A sequence variant, that changes one or more nucleotides, resulting in a different amino acid sequence but where the length is preserved, SO:0001583, MODERATE
- **protein_altering_variant:** A sequence_variant which is predicted to change the protein encoded in the coding sequence, SO:0001818, MODERATE

Variant Effect Predictor (VEP) from Ensembl maintains a list of consequence terms¹ and their meaning when used in VEP.

93.4 What kind of variant annotators can I use?

Variant effect annotation is a burgeoning field with several tools and techniques:

- VEP: Variant Effect Predictor² A web-based tool from Ensembl that integrates with the Ensembl data.
- snpEff³: Command line genetic variant annotation and effect prediction toolbox.
- AnnoVar⁴: a software tool to functionally annotate genetic variants detected from diverse genomes.
- VAAST⁵: Variant annotation, analysis, and search tool.

93.5 How do I use snpEff?

We will compare two strains of the Ebola virus, and we will attempt to determine which changes are likely to impact the function of the genome.

¹http://useast.ensembl.org/info/genome/variation/predicted_data.html#consequences

²<http://www.ensembl.org/info/docs/tools/vep/index.html>

³<http://snpeff.sourceforge.net/index.html>

⁴<http://annovar.openbioinformatics.org/en/latest/>

⁵<http://www.yandell-lab.org/software/vaast.html>

For all annotation tools, we need to know which genome build the annotation database is using. We will need to align our data against the same genome build.

Find what databases does snpEff know about the present.

```
snpEff databases > listing.txt
```

Find the information on ebola

```
cat listing.txt | grep ebola
```

Produces the output:

```
ebola_zaire Ebola Zaire Virus KJ660346.1 OK
```

```
http://downloads.sourceforge.net/project/snpeff/databases/v4_1/snpEff_v4_1_ebola_zair
```

We need to download that database to use it

```
snpEff download ebola_zaire
```

We can visualize the content of a database for example to check the chromosomal naming and have an understanding of what the database contains.

```
snpEff dump ebola_zaire | more
```

Note the version number of the accession number KJ660346. As a rule, we have to ensure that our alignments are performed using the same build. Using this accession number means that our effects will be in reference to this genomic build.

Get the script that we used before:

```
curl -O http://data.biostarhandbook.com/variant/find-variants.sh
```

The quality of a script becomes evident when we need to modify it to run on a different data set. Fewer the modifications we need to make, the less likely it is that we will introduce an error. Ideally, we change just a small section or a parameter, and the whole script should run with no other changes. Think about how much time and effort this saves you. Within seconds you can re-run the entire analysis on a different genome (or with different SRR data).

We also need to select SRR runs that we wish to study with the same commands that we demonstrated in the chapter Variant calling Ebola data

```
curl http://data.biostarhandbook.com/sra/ebola-runinfo.csv > runinfo.txt
cat runinfo.txt | grep "04-14" | cut -f 1 -d ',' | grep SRR | head -5 > samples.txt
```

The file `samples.txt` contains five sample run ids:

```
SRR1972917
SRR1972918
SRR1972919
SRR1972920
SRR1972921
```

Run the script:

```
bash find-variants.sh KJ660346 samples.txt
```

And finally to produce the annotated results:

```
snpEff ebola_zaire combined.vcf > annotated.vcf
```

Now to show these annotations we also need to build the custom genome in IGV.

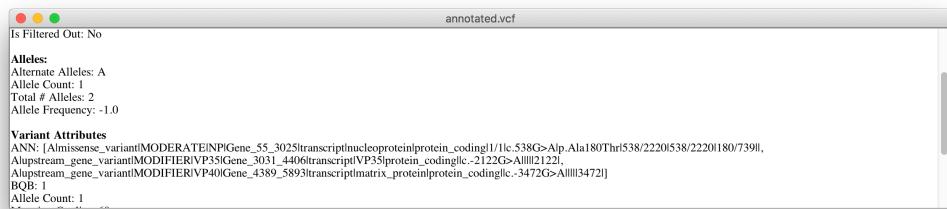


Figure 93.1

`snpEff` also generates report files `snpEff_summary.html`. The file is an HTML report with several entries. For example “Codon changes” shows:

93.6 Can I build custom annotations for snpEff?

Custom annotations can be built for `snpEff` from GeneBank and other files. See the Install `snpEff`⁶ page for details.

⁶<http://snpeff.sourceforge.net/>

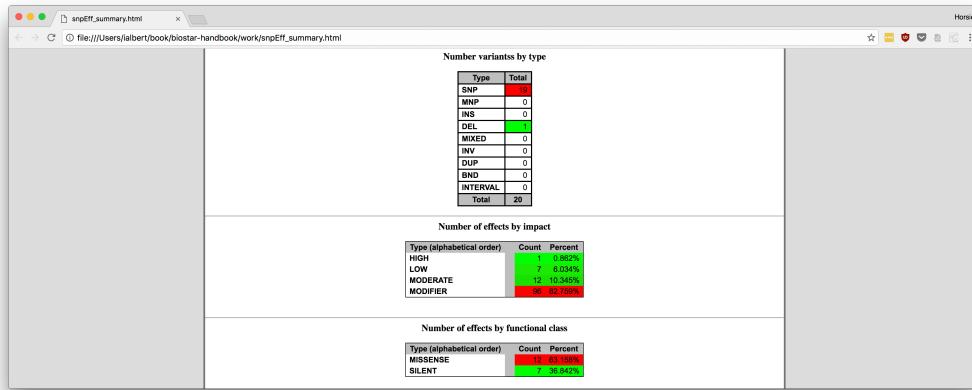


Figure 93.2

93.7 How do I use the Ensemble Variant Effect Predictor (VEP)

Get our subset of variants obtained from 1000 genomes project called `subset_hg19.vcf.gz` from <http://data.biostarhandbook.com/variant/> or from the command line as:

```
curl -O http://data.biostarhandbook.com/variant/subset_hg19.vcf.gz
```

Visit the VEP website at <http://useast.ensembl.org/Tools/VEP>.

Just as with the Ebola data, it is essential to use the same genome build for predictions as for alignment. For human genomes, the process is more error prone since, unlike the Ebola virus, the chromosome names stay the same across builds. Hence there is no way to tell from the name of the sequence what genomic build it came from.

Uploading the results and running VEP produces a report with quite a bit of detail.

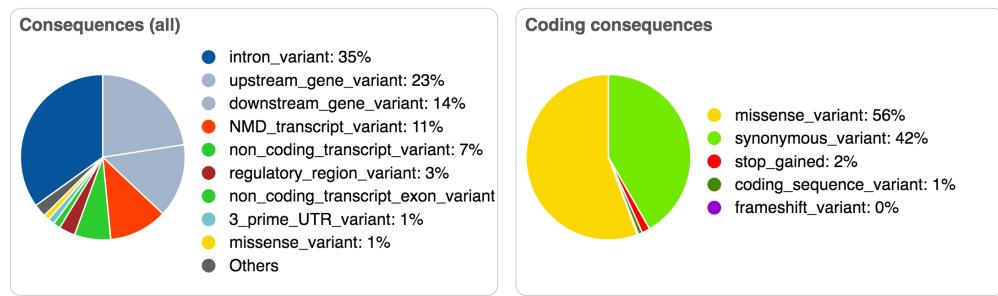


Figure 93.3

Chapter 94

Why is variant calling challenging?

As you will find there is no shortage of scientific papers comparing different variant callers. What is however never adequately explained is why is this task challenging and how are various tools trying to handle the problems.

94.1 Why is it so difficult to call variants?

Most genomics-related DNA sequences have several properties that confound intuition as they require thinking about context that human mind does not typically need to solve:

1. The DNA sequences can be exceedingly long (millions of bases)
2. Just four building blocks A/T/G/C make up any DNA sequence. These, even when distributed randomly could produce a variety of patterns just by sheer chance.
3. The DNA is only partially affected by randomness - usually, most of it has a function.
4. Depending on the organisms various segments of the DNA are affected by different rates of randomness.
5. Experimental protocols break long pieces of DNA into tiny ones that we then attempt to match back to a reference. Ideally, we hope that it still matches the most closely to the original location - because only

then will our match be a variant of the site of interest.

If our fragment were to match better somewhere else than the original location our sequence would align there instead of its “correct” position.

We call this a “misalignment,” though as we mentioned before the word is usually a misrepresentation of what happens. It is not actually “misaligning.” From algorithmic perspective the alignment is correct, and it does what we instructed it to do.

As you can see the deck is stacked quite a bit against us - it is surprising that variant calling works as well as it does!

94.2 How do variant calls overcome these challenges?

Nothing gets scientist’s minds going better than the need to tackle a seemingly impossible problem. Unexpectedly, the very fact that we chop sequences into little pieces has an unexpected benefit. If we only had a single sequence at hand then, once a “misalignment” took place there would be little extra information to recognize and correct it.

When the sequences are short, then each is affected slightly differently - and as it turns out - it is possible to attempt to recover the “correct” alignment from all the different types of errors that affect each shorter read differently. In essence, it is feasible to find another variation that would better explain and reconcile the differences present in each read.

This process has different implementations: it may be called *realignment* or *probabilistic alignment*.

94.3 How to explore the effect of mutations on variant callers?

We have written a script called compare-variant-callers.sh¹ that allows us to generate variants with multiple methods. The script assumes that you have

¹<http://data.biostarhandbook.com/variant/compare-variant-callers.sh>

installed `bcftools`, `freebayes` and `GATK` - if you only have a subset of these installed, you will need to edit the script and comment out the sections that run programs that your computer does not have installed.

Obtain the code:

```
curl -O http://data.biostarhandbook.com/variant/compare-variant-callers.sh
```

Now run it:

```
bash compare-variant-callers.sh
```

It should print:

```
*** Obtain data for AF086833 and save it as refs/REFERENCE.fa.
*** Index refs/REFERENCE.fa with several tools.
*** Generating simulated reads from: GENOME.fa
*** Aligning the reads into: align.bam
*** Genome alignment into: global.bam
*** Calling variants with samtools.
*** Calling variants with freebayes.
*** Calling variants with GATK.
*** Run completed successfully.
*** VCF files: samtools.vcf freebayes.vcf gatk.vcf
```

It ran three variant callers and created quite a few files, but the ones we are interested in are:

1. `samtools.vcf`
2. `freebayes.vcf`
3. `gatk.vcf`

Since we have not yet introduced any mutations, none of these files should contain any mutations.

Let's change that.

The file called `GENOME.fa` contains the genome from which the sequencing samples are generated. With a method of your choice modify and save this file. We will use the command line tool `biosed` to create a SNP replacing an A with a T. We have also selected this variation to be at position 2000 to help us more quickly identify whether the SNP caller indicates it correctly.

```
# A shortcut to the reference.
REF=refs/REFERENCE.fa
```

```
# This is the genome file that gets mutated from the reference.
GENOME=GENOME.fa

# Replace one sequence with another (the only difference is the A -> T at the end
cat $REF | biosed -filter -target AGGGTGGACAACAGAAGAAC -replace AGGGTGGACAACAGAA

now rerun the comparison script:
bash compare-variant-callers.sh
```

Here is what we get. All variant callers can reproduce and find this SNP:

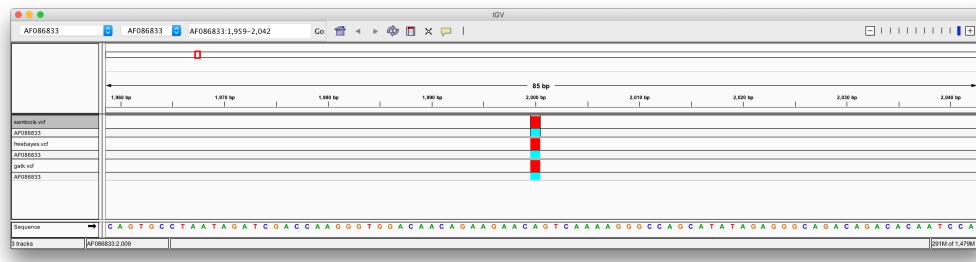


Figure 94.1

94.4 What happens when the variation is more complicated?

Let us cut two bases from the end of the sequence; this is now a two base INDEL

```
cat $REF | biosed -filter -target AGGGTGGACAACAGAAGAAC -replace AGGGTGGACAACAGAA

now rerun the comparison script:
bash compare-variant-callers.sh
```

What do we get now? Each of the variant callers produces a DIFFERENT variant call.

94.4. WHAT HAPPENS WHEN THE VARIATION IS MORE COMPLICATED? 735

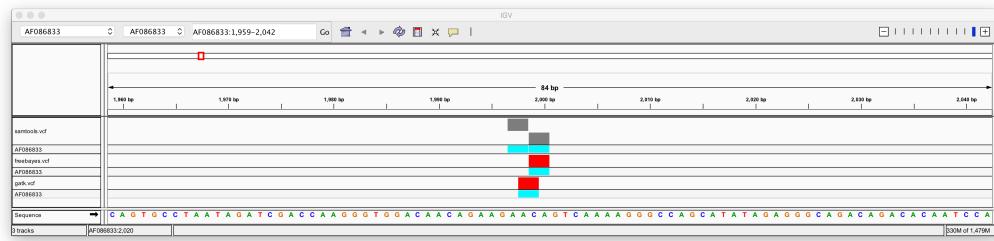


Figure 94.2

Part XXI

RNA-SEQ PRINCIPLES

Chapter 95

Introduction to RNA-Seq analysis

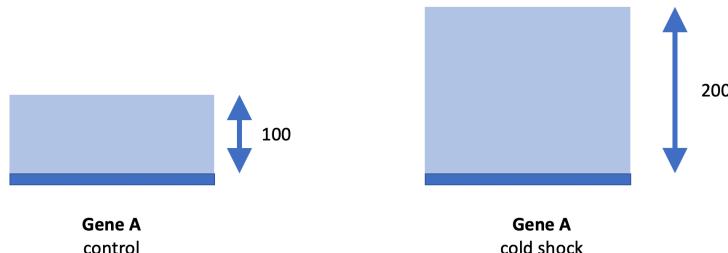
95.1 What is RNA-Seq when reduced to its essence?

Conceptually RNA-Seq analysis is quite straightforward. It goes like this:

1. Produce sequencing data from a transcriptome in a “normal” (control) state
2. Match sequencing reads to a genome or transcriptome.
3. Count how many reads align to the region (feature). Let’s say 100 reads overlap with **Gene A**.

Now say the cell is subjected to a different condition, for example, a cold shock is applied.

1. Produce sequencing data from the transcriptome in the “perturbed” state
2. Match sequencing reads to the same genome or transcriptome.
3. Count how many reads align to the same region (feature), gene A. Suppose there are 200 reads overlapping with **Gene A**.



200 is twice as big as 100. So we can state that there is a two-fold increase in the coverage. Since the coverage is proportional to the abundance of the transcript (expression level), we can report that the transcription level for gene A doubled under cold shock. We call the change as *differential expression*. We're done with the "informatics" section of our RNA-Seq analysis.

Now the process of biological interpretation starts. We need to investigate **Gene A**, what functions of it are known, is the response to cold shock one of them? If yes - what is the story, if no perhaps we discovered a new role for the gene. And so on.

That's RNA-Seq in a nutshell.

Of course, the reality is not that simple - the complexity, stochasticity, and imprecision of the many processes coupled to the scale of data conspire against determining fold change so readily. We can't just divide the numbers; we can't just trust the numbers. There is going to be a lot of bookkeeping, defensive measures taken to protect against you-know-who, The Dark Lord of False Discoveries. He grows stronger as the data grows; thus the defense is challenging even though what we need to do is simple.

Take solace and strength from recognizing that the complexity comes from data and information management - not some abstract, high-level problem-solving. The downside is that there is no secret trick that would make everything fall into place. The whole process is a data flow: from data, we make more data.

For an RNA-Seq analysis, the most important is to know which dataset that you produced is quantification matrix. That is the output of your analysis. For our toy example the file would look like this:

name	control	cold-shock
Gene A	100	200

This file contains your RNA-Seq results.

95.2 What is quantifying?

While the first applications of sequencing were designed to ascertain the nucleotide composition of a target DNA molecule, over time a radically different use emerged, one that has become increasingly more useful for life scientists. Collectively we call these type of applications as “quantifying via sequencing.”

In practice, this means that sequencing is used to determine not the composition of DNA fragments but their abundance. The usefulness of this lies in that when we have biological processes affecting the abundance of different DNA fragments; then by measuring these abundances, we estimate the natural process itself.

95.3 What is RNA-Seq analysis?

RNA-Seq (RNA sequencing) uses sequencing to identify and quantity of RNA in a biological sample at a given moment.

95.4 Why should I consider RNA-seq (over alternatives, e.g., microarray, qPCR)?

The invention of oligo-based microarrays allowed interrogation of multiple genes (and the entire “known” gene complement) from a genome at once. Microarrays opened the door for new opportunities of quantitative and diagnostic applications of gene expression. RNA-seq extends that functionality by allowing more accurate quantitation of gene expression compared to microarrays (think **digital** vs. **analog**). RNA-Seq has a much wider dynamic range compared to microarrays (10^5 vs. 10^3)¹.

¹<http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0078644>

RNA-seq is not immediately replacing microarrays as the method of choice for estimation of gene expression. In specific cases (e.g., a small research project restricted to well known/characterized genes), microarrays may still provide a useful alternative; they also have a low cost per sample and short turnaround time.

95.5 What are the main methods for quantifying in RNA-Seq reads?

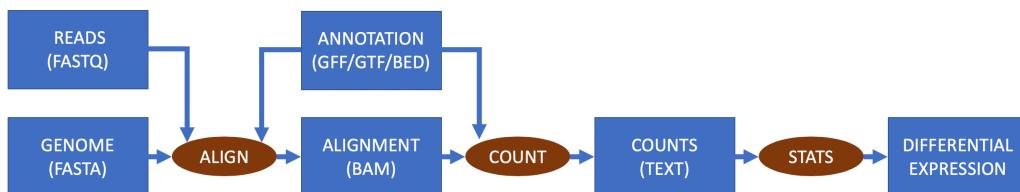
Two main approaches are currently in use. As always it is best to think in terms of what data they operate on and what data gets produced, rather than tool or algorithm names.

95.5.1 Quantifying against a genome

When quantifying against a genome case the input data will be:

1. Sequencing reads (FASTQ)
2. Genome including non-transcribed regions (FASTA)
3. Annotations that label the features on the genome (BED, GFF, GTF)

the approach will intersect the resulting alignment file with the use of annotations to produce abundances that are then filtered to retain statistically significant results.



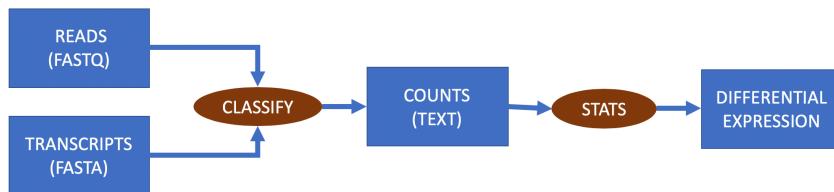
95.5.2 Classifying against a transcriptome

When classifying against a transcriptome the input data will be:

1. Sequencing reads (FASTQ)

2. Transcriptome files that contains all transcripts of the organism (FASTA)

the approach will directly produce abundances that are then filtered to produce statistically significant results.



95.6 What is the typical outcome of an RNA-Seq analysis?

The goal of most RNA-Seq analyses is to find genes or transcripts that change across experimental conditions. This change is called the differential expression. By observing these genes and transcripts, we can infer the functional characteristics of the different states.

95.7 What complicates RNA-Seq analysis?

First, what we measure via sequencing is almost always a relative measure. To use a term from statistics, we draw a “statistical sample” from the total amount of DNA. When we do that, we measure relative ratios. Relative ratios mean that the same absolute amount of one kind of RNA may appear at different abundances merely based on the presence and abundances of other types of RNAs that are present.

For example, say we have 100 As and 100 Bs in a population and we’ve estimated their abundance via sampling by looking at 20 randomly selected entries. In most cases, the relative abundance of A would then look like to be close to 50%. But if we had 100 As, 100 Bs, and 100 Cs in the next experiment, the abundance of A would turn out to be 33.3% of the total even though the same absolute amount of gene A was present. If we did not know how many categories there were, to begin with, we could be led to believe

that the abundance of A decreased in the second experiment whereas only the relative amount decreased.

Second, most current sequencing instruments measure only a short segment of each fragment. Transcription in higher eukaryotes makes use of splicing where introns get removed, and exons are joined in different configurations. Since the splicing of different exons will form similar transcripts, it can be difficult to match a sequencing read to its most likely originating transcript.

Third, unlike DNA, which is static, the mRNA abundances change over time. An experimental protocol will need to ensure not only that we observe a change but that this change correlates with the experimental conditions. Typically this is achieved by measuring the same state multiple times.

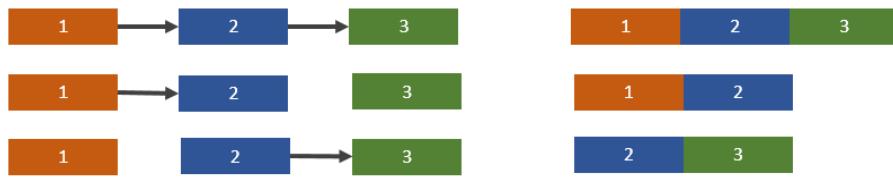
The process of repeating the same measurement is called replication. In general, the differential expression is considered informative when the changes between replicates are smaller than the differences between replicates across conditions. There are several methods to compare replicates and estimate the magnitude of changes.

Several competing methods have been proposed to account for the problems that we have enumerated above. Besides, there are several (too many perhaps) scientific publications that attempt to finally put these questions to rest and come up with a recommendation for the “best” method. Alas, the results of these papers are often underwhelming and end up with “it depends” and hand-waving statements that defer responsibility back to the reader.

Just as with variant calling, you have to recognize that the information encoded in the data significantly affects the performance of a given method. For some types of data, it does not matter **at all** what plan you pick. For other datasets, each process produces radically different results - and you have to decide which one to trust.

95.8 What are gene isoforms?

Gene isoforms are mRNAs that are produced from the same locus but are different in their transcription start sites (TSSs), protein-coding DNA sequences (CDSs) and untranslated regions (UTRs), potentially altering gene function.

**Figure 95.1**

Above we show three possible isoforms of a gene composed of three exons. Think about how reads that are fully contained in exon 2 would not be informative as to which transcript is expressed.

95.9 What kind of splicing events exist?

Five primary modes of alternative splicing are recognized:

1. Exon skipping or cassette exon.
2. Mutually exclusive exons.
3. Alternative donor (5') site.
4. Alternative acceptor (3') site.
5. Intron retention.

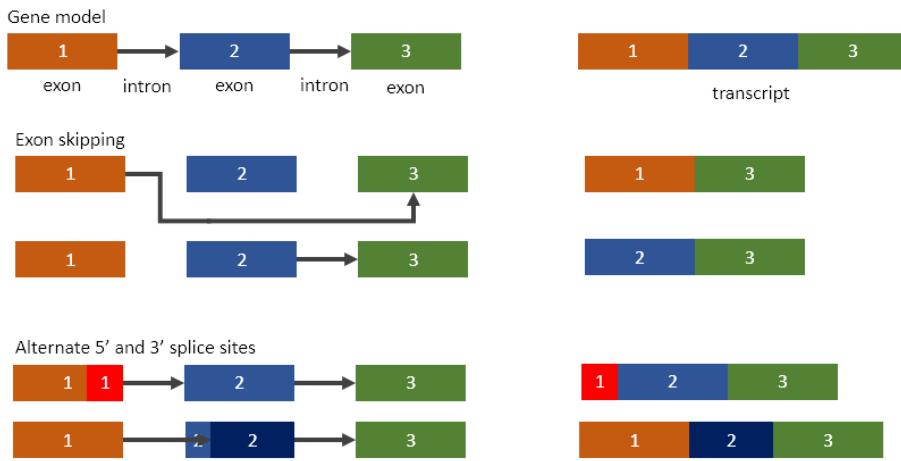
A few of these examples shown below:

When viewing RNA-Seq data in IGV you can recognize and identify these by eye.

95.10 How does RNA-seq analysis work?

The RNA-seq protocol turns the RNA produced by a cell into DNA (cDNA, complementary DNA) via a process known as reverse transcription. The resulting DNA is then sequenced, and from the observed abundances of DNA, we attempt to infer the original amounts of RNA in the cell.

Conceptually the task looks simple. It seems that all we need to do is count is how many DNA fragments are there. If we get higher numbers under a particular condition, it means that the cell must be producing RNA at

**Figure 95.2**

higher levels. In reality, the problems of counting and comparing are more complicated.

95.11 How many replicates do I need?

While we can make some inferences from a single measurement, in the current state of the field the minimal number of replicates typically recommended is three, and five are even better. As a general rule of thumb when you expect the effects to be subtle and biological variation significant (e.g., experiments with live animals from different populations) more replicates are better. Note that by generating five replicates across two conditions, each producing two files you are already looking at 20 FASTQ files coming off the sequencing instrument. The ability to automate processes is more important than ever.

95.12 Will there ever be an optimal RNA-Seq method?

There is an analogy to a mathematical paradox called the voting (Condorcet) paradox². This paradox states that when tallying votes for more than two options, we could end up conflicting with the individual wishes of a majority of voters, and there is no optimal way to count the votes that would avoid all possible conflicts.

When RNA-Seq methods assign measurements to genes, a sort of voting scheme applies. Because often there are ambiguities on how to resolve a measure, decisions have to be made to allocate it to its most likely origin (“candidate”) transcript. Unsurprisingly the Condorcet paradox will apply, and no matter what method we choose it is possible to count reads in ways that do not correspond to reality.

What is the take-home message here though? We believe that the explanation is as simple as it is counter-intuitive:

There is no best method for RNA-Seq. There are only methods that are *good enough*. The meaning of *good enough* evolves. “Good enough” means that the method will likely be able to identify some of the gene expression changes if they exist and are relevant to the study.

But there is always a chance that you’ll have unusual phenomena to quantify for which your initial choice of method is not optimal. The most crucial skill then is to recognize this situation.

You will need not just to learn how to perform an RNA-Seq analysis but to understand when a process does not seem to produce reasonable or correct results.

In our opinion, the usability and documentation of a method are among its most essential ingredients. Treating an RNA-Seq analysis as a black box is a recipe if not for disaster then for superficial and uninformative conclusions.

²https://en.wikipedia.org/wiki/Voting_paradox

Note: You know RNA-Seq when you can quickly analyze a dataset with at least three different methods.

Having a clear understanding of the processes will allow you to make informed decisions. You will see that it is not that difficult to use entirely different methods to analyze the same data. What is more challenging is to make an informed decision.

95.13 What is the first decision when performing an RNA-Seq analysis?

In general, the most important distinguishing factor between methods is the choice of the reference frame:

1. You may want to quantify against a **genome**, a database of all the DNA of the organism
2. You may choose to compare against a **transcriptome**, a database of all known transcripts for the organism.

Using a genome allows for the discovery of potentially new transcripts and gene isoforms. Using a transcriptome usually means more accurate quantification but only against a predetermined “ground” truth.

95.14 Should I quantify against a genome or a transcriptome?

YES

As of 2019, our recommendation is that whenever possible, that is both transcripts and genome are known, you should do both :-!)

95.15 What are the steps of an RNA-Seq analysis?

An RNA-Seq analysis is composed of three separate stages:

1. Assign reads to transcripts (alignments, mapping or classification).
2. Estimate abundances (read counting).
3. Compare abundances (differential expression).

There is quite a bit of disagreement regarding the “optimal” way to do each of these steps is, leading to numerous alternatives, each with its benefits and trade-offs. We sometimes mix and match different methods if it appears appropriate for a given data set.

Depending on the implementation details, some of these stages may be combined into what seems to be a single action. For example, `kallisto` will perform assignment and abundance estimation in a single step, or `cuffdiff` will carry out the read counting and differential expression computation as a single operation.

95.16 What is a splice-aware aligner?

A coding region of the DNA may produce multiple gene isoforms. Since we sequence DNA fragments by breaking the transcripts into smaller fragments, matching reads to exons alone is usually insufficient to establish which transcripts were present. For example, if our gene model supports the following three transcripts:

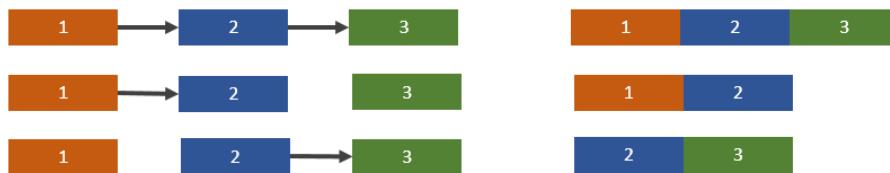


Figure 95.3

Then matching to exon one would only inform us that we have a transcript from one of the first two pairs, whereas matching to exon two would not help

at all to resolve the question of which transcripts are present.

The reads coming from the locations that join two exons (these are the so-called **splice sites**) will align to the genome with a large gap (the intron) between them. These are so-called spliced alignments. Thus, to align our RNA-Seq data to a genome, we need to use aligners that are “splice-aware”. These aligners can align reads with considerable gaps in between without penalizing the alignment.

It is only the spliced reads, a tiny subset of all reads, that carry the information of which neighboring exons were joined. But even these joins will resolve immediate neighbors, and can’t connect second or third neighbors. Hence, when aligning short reads, we will be unable to fully and reliably determine the entire transcript from measurements alone. All we can see are junctions that tell us partial information.

Tools may assist in estimating which transcripts exist by building a mathematical or probabilistic model to form the most straightforward possible arrangement that could explain all the observed junctions. As you might suspect, life rarely works by formulas - an organism wants to survive - and that may be at odds with “simple explanations”. While there are forces that drive so-called “minimal parsimony”, this property is not required for the functioning of a cell.

Recall how aligners differ in far more than just producing the alignment. The magnitude of differences is even more substantial across splice-aware aligners because there are even more parameters to consider. Thus different methods and software often show surprising and unexpected differences when paired up with one another.

95.17 Which splice-aware aligner should I use?

Our recommendation for splice aware aligners are:

- HiSat2: similar algorithms as Bowtie2/Tophat2 but performs at much faster speeds
- Subjunc: explicitly designed for gene expression analysis.
- BWA: the MEM algorithm of bwa supports spliced alignments.

- BBMap: Is capable of detecting very long splicing.
- STAR: fast, produces counts as well.

As you can see, we are in just the first phase of describing methods, yet the number of alternatives is already increasing. Later sections of the book will provide examples for the same analysis with different approaches.

95.18 How do I quantify mRNA abundances?

The history of estimating the magnitude of gene expression is short and convoluted. You may be surprised to learn that even today there is quite a bit of disagreement regarding what value we should compute, how to quantify, and how to compare the amount of mRNA in different samples.

The most common measures used currently are:

1. Counts: The number of reads overlapping with a transcript.
2. RPKM/FPKM: Reads/Fragments per kilobase of transcript per millions of reads mapped.
3. TPM: Transcripts per million

Again the rule of **good enough** applies. You need to choose a method that is good enough for your purpose. The challenge, of course, is how do you know when a technique is good enough?

Understandably most scientists define “good enough” to be the method “that seems to work.” While such an “after the fact” determination sounds unscientific and ludicrous, there is some validity to it. Remember, life does not work - believing that one particular way to align reads is universally optimal is no less ludicrous.

95.19 How do I compare mRNA abundances?

The troubled history of quantifying mRNA abundance goes hand in hand with the efforts to come up with a measure that can also be used to compare the levels of expression of transcripts between different conditions. Two types of comparisons may be necessary, and different criteria should be used for each:

1. *Within-sample* comparisons. In this case, we compare the expression of genes within the same experiment. For example: in this experiment does gene A express at a higher level than gene B?
2. *Between-sample* comparisons. In this case, we compare the expression of genes across experimental conditions. For example, has the gene expression for gene A changed across different experimental conditions?

Chapter 96

RNA-Seq terminology

In the section, we'll try to clarify terms that you may encounter when reading RNA-Seq related documentation.

96.1 What is a sample?

The commonly used word “sample” in sequencing lingo means a biological sample (extraction) that was subjected to sequencing. When we talk about samples, we assume that these are grouped into conditions via a so-called experimental design. For example, we could say we have 10 samples, arranged into two conditions with five replicates per condition: $2 \times 5 = 10$

The “RNA-Seq” sample should not be confused with “statistical sample” – in statistics “sample” refers to selecting a subset of a population.

96.2 What is normalization?

When we assign values to the same labels in different samples, it becomes essential that these values are comparable across the samples. The process of ensuring that these values are expressed on the same scale is called *normalization*. Several different normalization methods are in use, and each operates with different types of assumptions.

96.3 What is a library size normalization?

The term “library size” is a frequently used (but improper term) that usually means the sequencing coverage (depth) of a sample. For example, it might be that for experiment A we have ended up with twice as much material being placed into the instrument than for experiment B. Our quantification methods need to be able to detect and account for the difference that occurs merely due to the differences of the amount of initial material.

96.4 What is the effective length?

The sequencing coverage drops towards the ends of DNA molecules due to the lower chances of producing fragments that include the end of the molecule. For genomes, this loss of coverage is usually less of a problem since it only affects the ends of very long chromosomes - and typically we have only a few (dozens) of locations. When working with RNA transcripts that number in the ten or hundred thousands, the edge effects will affect each transcript. Besides, shorter transcripts will be more impacted (the edge effect is a larger fraction of the transcript) than longer ones.

A correction is usually applied to account for this edge effect - a so-called “effective length” will replace the true length of each feature. For example, one common correction is to shorten each transcript end by half of the read length. Only the values calculated over this shortened length will be used.

96.5 What gets counted in a gene level analysis?

The gene level analysis treats every gene as a single transcript that contains all exons of the gene. In some cases, this approximation is sufficient to get meaningful results. In other cases, a more fine-grained analysis will be necessary wherein abundances are quantified for each transcript level.

Gene level analyses will collapse the transcripts into a single “representative” sequence - that may not be a valid transcript - it is a sum of all transcripts. It

is clear that a gene level analysis will not be able to identify those mechanisms that rely on changes of the abundance of iso-forms of a single gene.

96.6 What is the RPKM?

If we wanted to compare the number of reads mapped to one given transcript to another transcript of the same sample, we have to account for the fact that longer transcripts will produce more DNA fragments merely because they are longer.

If N were the total number of reads mapped to a transcript, and C was the total number of reads mapped for the sample, we cannot just take N / C as our measure of gene expression. A single copy of a longer transcript will produce more fragments (larger N) than a single copy of a shorter transcript.

Instead, we may choose to divide the fraction of the reads mapped to the transcript by the effective length of the transcript:

$$\text{gene expression} = N / C * 1 / L$$

Basically, we first compute the fraction of the total reads that map to a transcript then divide that by the length of the transcript.

This number would typically be tiny since just a small subset of reads of the total will align to any gene, then we also dividing by the transcript length that again may be large number in the thousands. Other sciences solve the problem of reporting small numbers by using words such as *milli*, *micro*, *nano*, *pico*, etc. to indicate the scale.

Those who came up with the concept of RPKM yearned to create a more “user friendly” representation to avoid “confusing the stupid biologists” (they did not actually mention the word stupid, but it is so clearly implied in there) and decided that the best way to get there will be to express L in kilobases (10^3) and C in terms of millions of reads (10^6) in essence adding a factor of a billion to the formula above:

$$\text{RPKM} = 10^9 * N / L * 1 / C$$

Hence a weird unit was born, the RPKM, that, in our opinion only ends up being a lot more confusing than it needs to be. Those with training in sciences where formulas are not just pulled out of thin air, like say physics,

will note immediately that the RPKM as defined above has a “dimension” to it.

Whereas the N and C are integer numbers, the $1/L$ is an inverse of a distance. Which should give everyone a pause. Why is it appropriate to measure gene expression by a quantity that is an inverse of a distance? Transcripts either exist or not. The unit of the RPKM as defined above is a measure of the speed of transcription (how many units are produced per length) and not how many transcripts exist.

Oh, how life would have been a whole lot simpler if the original group¹ of scientists that invented this measure would have just called this quantity a *pachter*, as a hat tip to Lior Pachter², one of our favorite academic-rebel trolls. Then RPKM would have been just called a *pico-pachter* and may have ended up being a better understood quantity.

As we learn more about the limitations of RPKM, the consensus appears to be that RPKM is an inappropriate measure of gene expression, its use should be curbed and eliminated. Yet, as you will see, some of the most commonly used software continues to produce results in RPKM and is a zombie concept that refuses to die.

96.7 What the FPKM is that?

FPKM is an extension of the already flawed concept of RPKM to paired-end reads. Whereas RPKM refers to reads, FPKM computes the same values over read pair fragments. Conceptually is even worse as the word “fragment” only adds another level of ambiguity to an already questionable concept. Which fragments will be considered: The apparent fragment size (aka TLEN from SAM?) The sum of read lengths for the pair? The alignment lengths? The sum of aligned regions for each read? ... Alas this is not defined, you are left at the mercy of the tool implementer.

For further reading we recommend a useful blog post by Harold Pimentel, it is the post that inspired the title of this question: What the FPKM? A review of RNA-Seq expression units³.

¹<http://www.nature.com/nmeth/journal/v5/n7/abs/nmeth.1226.html>

²<https://liorpachter.wordpress.com/>

³<https://haroldpimentel.wordpress.com/2014/05/08/>

96.8 Why are RPKM and FPKM still used?

If the requirement for accurate is sufficiently low RPKM, FPKM can produce “useful” results. For example, they most certainly better than a naive estimate of read counts. Our measures are approximations, the method itself is an approximation. On the other hand several known factors severely affect RPKM and FPKM studies - and in our opinion, most results relying on these values are scientifically less sound.

96.9 What is TPM?

One serious limitation of RPKM is that ignores the possibility that new and different transcripts may be present when experimental conditions change. The RPKM dimension of 1/distance also indicates that instead of being a quantity that indicates amounts, it is a quantity that characterizes the change over distance.

Values can be only compared when that “distance” is the same. As it turns out that “distance” that RPKM tacitly assumes to be the same is the total transcript length. It assumes the reads are distributed over the same “range” of DNA.

A more appropriate distance normalization should divide with a value that accounts for the potential changes of the total transcript length T .

$$\text{gene expression} = N / L * 1 / T$$

A way to incorporate both the number of counts and the length into T is to sum the rates:

$$T = \sum Ni/Li$$

where i goes over all observed transcripts and Ni are the reads mapped to a transcript of length Li .

Not to be outdone by **RPKM** in the department of protecting biologists from confusion, a new measure was born, this time called the **TPM** where we multiply the above by a million to save biologists* from the unbearable mental overload of having to deal with small numbers:

*what-the-fpkm-a-review-rna-seq-expression-units/

```
TMP = 10^6 N / L * 1 / sum(Ni/Li)
```

Life would have been a whole lot simpler if the original group⁴ of scientists that invented TPM would have just called this quantity a *salzberg*, as a hat tip to Steven Salzberg⁵ one of the Greatest Bioinformaticians Of All Time (GBOAT). The TPM would have been called a *milli-salzberg* and may have turned out to be a better-understood quantity.

Since there is a distance dimension both in the numerator and denominator, the TPM is dimensionless (unlike RPKM).

96.10 What is TMM (edgeR) normalization?

Trimmed mean of M values (TMM) normalization estimates sequencing depth after excluding genes for which the ratio of counts between a pair of experiments is too extreme or for which the average expression is too extreme. The edgeR software implements a TMM normalization.

96.11 What is DESeq normalization?

The DESeq normalization method (implemented in the DEseq R package) estimates sequencing depth based on the count of the gene with the median count ratio across all genes.

96.12 Do I always need an advanced statistical analysis?

Surprisingly, the answer is no. The methods that need to be employed depend on the goals of your experiment.

⁴[https://academic.oup.com/bioinformatics/article/26/4/493/243395/](https://academic.oup.com/bioinformatics/article/26/4/493/243395/RNA-Seq-gene-expression-estimation-with-read)
RNA-Seq-gene-expression-estimation-with-read

⁵<https://salzberg-lab.org/>

If, before starting the experiment, you knew which gene you wanted to know more about and you care only about this gene, then the law of large numbers works in your favor.

This is to say that it is very unlikely that your gene of interest was singled out by chance and was affected in a way that misleads you. This is to say that if you use your RNA-Seq data to verify a statement then the simplest of statistical tests and common sense suffice.

But if you did not know which transcripts might change and you wanted to reliably determine that out of tens of thousands of alternatives and their combinations then more sophisticated methods are necessary to ensure that whatever change you observe was not caused by natural variation in the data.

96.13 What is a “spike-in” control?

The goal of the spike-in control is to determine how well we can measure and reproduce data with known (expected) properties. A commercial product such as the “ERCC ExFold RNA Spike-In Control Mix”⁶ can be added in different mixtures. This spike-in consists of 92 transcripts that are present in known concentrations across a wide abundance range (from very few copies to many copies).

You may use spike controls to validate that a protocol operates as expected. Of course challenges still remain, the spiked protocol

96.14 How should I name samples?

With RNA-seq analysis you may need to work with many dozens of samples. One of the skills that you have to develop is to parse file names and connect them to known sample information. File naming practices vary immensely but having an odd naming scheme can be the source of the most devious catastrophes! We suggest the following practices:

1. Each attribute of the data should be captured by a single section of the name.

⁶<https://www.thermofisher.com/order/catalog/product/4456739>

2. If there is a hierarchy to the information then start with the MOST GENERIC piece of information and work your way back from there, end the file with the MOST SPECIFIC bit of information.

For example, this is an appropriate naming scheme.

```
HBR_1_R1.fq  
HBR_2_R1.fq  
UHR_1_R2.fq  
UHR_2_R2.fq
```

The first unit indicates samples: HBR and UHR, then comes the replicate number 1, 2 and 3, then the paired files R1 and R2.

A bad naming scheme would be one such encodes additional sample specific information into the sample without grouping them properly:

```
HBR_1_Boston_R1_.fq  
HBR_2_Boston_R1_.fq  
UHR_1_Atlanta_R2_.fq  
UHR_2_Atlanta_R2_.fq
```

Above both HBR and Boston represent sample specific information whereas 1 and 2 are replicate specific information at a different level of granularity. The names are less well suited to automation and you'll have to work around this limitation under the most unexpected circumstances.

In a nutshell, it is much easier to automate and summarize processes when the sample information is properly structured. You'd be surprised how few data analysts understand this - only to end up with programs that are a lot more complicated than need to be.

The most dangerous mistake you will ever make is one where you mix up your samples! Whereas other errors will manifest themselves in various obvious ways that allow you to recognize them, mislabeling data will silently produce incorrect results.

Examples of the errors cause by mixed up samples abound in science, here is one we saw last week:

- Study Linking Autism to ‘Male Brain’ Retracted⁷

⁷https://www.medscape.com/viewarticle/910982?nlid=129068_3901&src=wnl_

where the authors accidentally flipped the labels and managed to publish a paper with exact opposite results than what the data indicated. Of all errors that you may make, this one, and its variations: what is divided by what? are the ones that will cause the most lasting devastations. The computational tasks are so complex, the error so simple and paradoxically that makes it a lot harder to identify!

Simplicity is key to success!

Chapter 97

Statistical analysis in RNA-Seq

Our favorite summary for statistics comes from the blog post [The four aspects of statistics¹](#) where Frederick J. Ross writes:

Statistics resembles the apocryphal elephant being examined by blind men. Each person uses, and often only knows, a particular set of statistical tools, and, when passing on their knowledge, does not have a picture to impart of the general structure of statistics. Yet that structure consists of only four pieces: planning experiments and trials; exploring patterns and structures in the resulting data; making reproducible inferences from that data; and designing the experience of interacting with the results of an analysis. These parts are known in the field as

- design of experiments
- exploratory data analysis
- inference
- visualization

Sadly, this basic structure of statistics doesn't seem to be written down anywhere, particularly not in books accessible to the beginner.

read more on each point on the blog².

¹http://madhadron.com/posts/2016-01-15-aspects_of_statistics.html

²http://madhadron.com/posts/2016-01-15-aspects_of_statistics.html

97.1 Why do statistics play a role in RNA-Seq?

Whereas alignments or counting overlaps are mathematically well-defined concepts, the goals of a typical experiment are more complicated. The data itself may be affected by several competing factors as well as random and systematic errors. Statistics gives us tools that can help us extract more information from our data and can help us assign a level of confidence or a degree of uncertainty to each estimate that we make.

97.2 When do I need to make use of statistics?

You will need to think about statistics first when designing the experiment.

In this stage you have to enumerate the goals and parameters of the experiment. Consulting with a statistician, if you that is option is available, is obviously a good choice. The most important advice I would give is to not be too ambitious (greedy?) with the experiment. In my experience projects that try too cover too many ideas at once, multiple genotypes, multiple conditions, various interactions, multiple time points etc. end up with less reliable results than focused experiments.

It is akin to the joke: *If you have one clock you know what the time is, if you have ten clocks you never know which one is right.*

The second stage for statistics comes into play when you collect and process the data into a matrix. Then, in most cases, interpreting the information in either a row, a column or a cell needs to be done in the context of those other numbers in the table.

ID	Condition 1	Condition 2	Condition3
SEPT3	1887.75036923533	81.1993358490033	2399.647399233
SEZ6L	1053.93741152703	530.9988446730548	211.73983343458
MICALL1	136.421402611593	197.470430842325	120.9483772358

A statistical test is a process by which you make quantitative or qualitative decisions about the numbers.

97.3 What kind of questions can we answer with a statistical test?

Here is a selection:

- How accurate (close to reality) are these results?
- How precise (how many digits are meaningful) are the values?
- For which observation do values change between conditions?
- For which observation is there at least one changed condition?
- Are there genes for which there is a trend to the data?
- What kinds of patterns are there?
- Which observations vary the same way?

Statistical tests operate with principles such as margins of error, probabilities of observing outcomes and other somewhat indirect measures. These measures can easily be misinterpreted and scientists routine make mistakes when summarizing and reformulating statements derived from statistical tests. See the section on p-values later on this page.

97.4 What types of statistical tests are common?

The pairwise comparison is one of the most common and conceptually most straightforward tests. For example, a pairwise comparison would compare the expressions of a gene between two conditions. A gene that is found to have changed its expression is called differentially expressed. The set of all genes with modified expression forms what is called the differential expression (DE).

Here, it is essential to understand the type of results that pairwise comparisons usually produce. Almost always we want to answer the question of whether a gene's expression level has changed. But instead what we will typically obtain is the probability that there was no difference between conditions (i.e., the probability of the null hypothesis).

When this probability is low (small) we can reject the null hypothesis, and we conclude that there is a change. The expression "reject the null hypothesis" may seem like mincing words or trying to sound clever. But when further

investigations are necessary it is essential to use these results in their proper context. It is exceedingly common to formulate conclusions in a manner that imply more than what the tests support.

97.5 Do I need to involve a statistician?

Ideally, of course, the answer is yes.

But we're in the practical advice business here and, in reality, it is not always all that easy to find a collaborator well versed in statics. Besides, just as with bioinformatics it would be a mistake to oversimplify statistics into a purely procedural skill: "any statistician can do it." You would need to find a collaborator who understands the characteristics of biological data as well as the challenges of working with it.

We believe that understanding and performing simple statistical tests like pairwise comparisons, making sound and reliable statistical decisions are well within anyone's reach and in this book, we will provide you with alternative ways for doing it.

For more sophisticated data modeling, for example, time course analysis or comparing several samples at once, you would need a statistical collaborator, or you will need to spend some time and effort understanding the statistics behind it.

Statistics is not as complicated as it looks – so don't be discouraged. There is a wealth of information available on more advanced statistical modeling.

97.6 What is R?

Unlike most other approaches, where we install and run command line tools, statistical analysis in general and the differential expression detection, in particular, are typically performed using packages that run within the R programming environment: The R Project for Statistical Computing³

Learning how to program in R is not so simple. Those who claim otherwise are probably the lucky ones whose minds happen to fit R.

³<https://www.r-project.org/>

You see, the R language was designed before computer scientists understood how a programming language should work, what features it should have, and what data types are useful. So don't kick yourself if you can't seem to quickly learn R, it is without a doubt harder to learn than many other computational languages. In addition most people that write code in are not that well versed in proper software engineering practice. As a result typical R code is affected by far many more issues than code written in other domains of science.

That being said, thankfully, less complicated and manageable to learn how to run tools written by others. As with other programming languages, an R script is simply a list of commands instructing R to perform certain actions.

97.7 How do I install R?

While there are R versions packaged with `conda` we recommend that you install R with a downloadable installer, see the R Home Page⁴.

Once installed this way the program will be universally available on your computer from the command line as well.

97.8 Can I also install R from command line?

Yes, but then you will may end up with two versions of R installed. Both need to be set up the same way. To install R with `conda` do:

```
conda install r
```

If you work in R, a good option (especially when learning) is to run it via RStudio⁵, a graphical user interface to R.

⁴<https://www.r-project.org/>

⁵<https://www.rstudio.com/>

97.9 What is Bioconductor?

Bioconductor <https://www.bioconductor.org/> is a project that collects R-based code for genomic data analysis.

If you have R installed separately, please visit the URL for each tool for the installation instructions. The installation instructions are simple but may change in time:

- <https://www.bioconductor.org/packages/release/bioc/html/DESeq.html>
- <https://www.bioconductor.org/packages/release/bioc/html/DESeq2.html>
- <https://bioconductor.org/packages/release/bioc/html/edgeR.html>

The commands above will churn for a while and will install all the required packages necessary for the subsequent commands. To run a differential expression study we need data and an R program that does the job.

If you installed R with conda you will need to install the following from command line:

```
conda install -y bioconductor-deseq bioconductor-deseq2 bioconductor-edger r-gp
```

97.10 What does a p-value mean?

You probably don't know what a p-value means especially when it comes to interpreting biological data. Don't sweat it. We have yet to meet someone that does.

We have of course met overconfident people who think they understand p-values and most of them are quite convinced that they got it right. In our personal experience and observation, even statisticians giving a talk on p-values, and even statistical textbooks routinely misuse the term.

Postulate: In our opinion nobody fully understands p-values and their proper use in biology. There are only people that misunderstand and misuse the term less egregiously.

For every precise definition there is an expert opinion that can pull that apart and prove that it does always mean what it is supposed to.

- Give p a chance: significance testing is misunderstood⁶
- Scientists rise up against statistical significance⁷ in Nature, 2019
- Statisticians issue warning over misuse of P values⁸ in Nature, 2016
- ...and so on...

Note how there is an ever increasing number of publications describing the misuse of p-values; publications that are akin to old men ranting when trying to get kids get off their lawn. I know what you're thinking, I am going to ask a super-duper-expert statistician. Well, here are excerpts from an email that one of our biologists collaborators received when they asked one of the leading statistical experts (with multiple Nature level "rant" papers) on advice and recommendation on choosing the "proper" RNA-Seq method. We include it here as we found it quite representative of what you might expect:

[...]

...I need to caution you that high throughput data analysis is some-

⁶<https://theconversation.com/give-p-a-chance-significance-testing-is-misunderstood-20207>

⁷<https://www.nature.com/articles/d41586-019-00857-9>

⁸<http://www.nature.com/news/statisticians-issue-warning-over-misuse-of-p-values-1.19503>

what fragile. Almost any change in the analysis such as different options for the mappings or the normalization or the optimization algorithm in the statistical routines will change the gene list - sometimes dramatically. This is one reason that biologists use supplemental methods - from wet lab experiments to high throughput statistical methods such as clustering and gene set analysis - to assist in the interpretation of the results.

Another idea is to use somewhat different options in the differential expression analysis and then take the genes in the intersection or union of the significant genes from both analyses. Using the intersection is likely to be quite conservative, selecting the genes that have the highest signal to noise ratios, while the union is likely to be anti-conservative, having a higher FDR than the adjusted P-values or q-values suggest.

Omitting some low-expressing genes from the analysis has a very simple effect on the significant gene list - it will get bigger, but no significant genes should be dropped. The reason for this is that the unadjusted P-values for the remaining genes usually do not change, and the adjusted P-values are a simple function of the unadjusted values and the number of tests, which decreases when the number of tests is smaller. I say "usually do not change" because the estimated dispersion is a function of all the genes, and will change a bit, not usually not much, when some genes are omitted from the analysis...

[...]

A short summary of the above: **P-values! Sixty percent of the time they work every time.**

A noteworthy feature of the letter is its evasiveness and non-committal nature of it. The author does not wish to take any responsibility for making the hard decisions - all the responsibility is punted back to the biologist: a.k.a. do it many ways and see what works.

In addition we can't help but point out that the recommendation that starts with: *Omitting some low-expressing genes from the analysis...* feels very much

like p-hacking and data dredging⁹, a misuse of statistics, where, after knowing a partial answer (the genes that are lowly expressed) we tweak the input data (omit genes) for the sole purpose of getting more results presented as statistically significant.

Let's clarify something here. Perhaps our explanation will feel like splitting hairs but it is not - it cuts to the very essence of p-hacking. Filtering data before an analysis is an acceptable practice. You may remove lowly expressed genes, or highly expressed genes, genes that rhyme with foo, or genes that from far away look like flies, whatever you wish to do as long as you have an scientifically acceptable rationale for doing so. State and explain that rationale, then document it - all fine. What is *not acceptable* is the reasoning that the letter above recommends: to filter data for the *sole purpose* of making more results pass a threshold. That is the definition of p-hacking.

Do you now see the origins of our postulate: *Nobody fully understands p-values and their proper use in biology?* When under the pressure to deliver results, and, when faced with the complexity and messiness of real biological data, statistics was, is and will be misused and misinterpreted even by very guardians and the experts that later chide us for not using the concepts in the proper context. S

97.11 So how do I deal with p-values?

Your primary responsibility is to avoid outrageous, preposterous and egregious errors. Use statistics to avoid radical mistakes instead of relying it to be the mechanism that leads you to truth and meaningful discoveries. The bar seems awfully low, but dont' let that trick you into a false sense of security

A common misuse of a p-value is formulating a stronger statement than what it was created for. Perhaps the most common misuse of a p-value is expressed in the following way: “*our small p-values show that the our results are not due to random chance*” or “*our small p-values show that the value increased two-fold*” As it turns out that is not at all what p-values indicate.

Another very common misuse of a p-value is to consider a smaller p-value to be a stronger indication that an effect exists. Sorting by p-value is not the

⁹https://en.wikipedia.org/wiki/Data_dredging

right mechanism to put the “better” results first. We sort by p-value to have “some” ordering in our data and to apply a cutoff more easily.

To avoid misleading yourself and others here is what we recommend on p-values:

1. Think of the p-value as the probability of obtaining an effect of the size that you observe due to random chance. Note how the “size” itself is not a factor here, the size could be small or large. The pvalue does not capture the actual size. Just that chance of observing that particular size (or larger).
2. Again remember the p-value does not care about how “big” the effect is.
3. Think of p-values as selection cutoffs. Use them to reduce a list for further study. But not because 0.05 (or whatever the cutoff) is good and 0.06 is not, it is because you have cut the list somewhere. There is nothing inherently right about 0.05 - or any other cutoff. The cutoff is arbitrary - beggars cannot be choosers - when you got noisy data with few results, you’ll have to be more generous with the cutoff. Then with noisy data expect the burden of proof to be higher, you will need additional evidence to back up your findings.
4. Do NOT use p-values as an indicator for more reliable or less reliable results nor as indicators of stronger or weaker effects.

Now, according to my theorem the sentences above most certainly contain inconsistencies and allow for invalid interpretations of p-values. My apologies.

The problems caused by misusing p-values are well documented, unfortunately the more papers you read, the less certain you’ll become that you understand the concept or that it is even worth using p-values *at all*:

- Interpreting P values¹⁰, Nature Methods 2017
- P values and the search for significance¹¹

¹⁰<https://www.nature.com/articles/nmeth.4210>

¹¹<https://www.nature.com/articles/nmeth.4120>

- ... and so on ...

97.12 Do I need to compute and discuss p-values?

Yes, I use them because there is no better alternative.

Theorem: The most cynical interpretation of p-values is that their serve as the mechanism to filter out studies created by people that were so clueless with respect to large scale analysis that they couldn't even produce small p-values. From that point of view p-values do correlate with the quality and validity of the research.

Corollary: An expert scientist can unwittingly publish a Nature publication with tiny p-values, impressive and compelling visualizations even when the underlying data is not different from random noise¹².

¹²<https://www.nature.com/articles/nature12535>

Chapter 98

Useful R scripts

For this book, we have developed R scripts to assist you with running statistical analyses. Use these scripts as starting points, and note how there is a wealth of information on the web for alternative approaches. Do also consult the recipes for examples.

If you installed R with conda you will need to install the following from command line:

```
conda install -y bioconductor-deseq bioconductor-deseq2 bioconductor-edger r-gp
```

If you installed R globally for your system visit the package installation:

- <https://www.bioconductor.org/packages/release/bioc/html/DESeq.html>
- <https://www.bioconductor.org/packages/release/bioc/html/DESeq2.html>
- <https://bioconductor.org/packages/release/bioc/html/edgeR.html>

98.1 How can I run RNA-Seq differential expression scripts from the command line?

Each script may be installed from the command line:

- `deseq1.r` that makes use of the DESeq method¹.
- `deseq2.r` that makes use of the DESeq2 method²
- `edger.r` that makes use of the EdgeR method³

Each of these scripts can be obtained and run from the command line. To obtain the scripts execute:

```
curl -O http://data.biostarhandbook.com/rnaseq/code/deseq1.r
curl -O http://data.biostarhandbook.com/rnaseq/code/deseq2.r
curl -O http://data.biostarhandbook.com/rnaseq/code/edger.r
```

98.2 How to the helper scripts work?

To see how these work we need a file with sample counts of known proportions, for example as those described in RNA-Seq: Analyzing control samples. We provide you with the file that contains the read counts at:

```
curl -O http://data.biostarhandbook.com/rnaseq/code/counts.txt
```

This file has the following content:

ERCC-00002	37892	47258	42234	39986	25978	33998
ERCC-00003	2904	3170	3038	3488	2202	2680
ERCC-00004	910	1078	996	9200	6678	7396
...						

Our scripts read their input from the standard input and write to the standard output in a form like this:

```
cat mydata.txt | Rscript do-something.r > output.txt
```

The differential expression scripts also take a so-called design parameter that describes how many columns correspond to each condition.

For example, the data above was summarized over 6 count columns, where the first 3 columns correspond to the first condition, and the second 3 columns correspond to the second condition. We will call this a 3x3 design and indicate that as such. Each script can be run with:

¹<http://bioconductor.org/packages/release/bioc/html/DESeq.html>

²<http://bioconductor.org/packages/release/bioc/html/DESeq2.html>

³<http://bioconductor.org/packages/release/bioc/html/edgeR.html>

```
# Analyze the counts with DESeq1.
cat counts.txt | Rscript deseq1.r 3x3 > results_deseq1.txt

# Analyze the counts with DESeq2.
cat counts.txt | Rscript deseq2.r 3x3 > results_deseq2.txt

# Analyze the counts with EdgeR.
cat counts.txt | Rscript edger.r 3x3 > results_edger.txt
```

98.3 Do I need to run all three scripts?

No, we usually stick with one. Each approach has some strengths and weaknesses. We'll talk more about these choices in the How to choose an RNA-Seq analysis chapter.

98.4 What is the `norm-matrix` file?

When you run say `deseq1` script, you will note that it also produces a file called `norm-matrix-deseq1.txt`. It will be named similarly for other methods.

This data is typically not shown to the end user when you run the method, we present and save in a file as we believe it carries essential information that you may need during your analysis. For example, the file might look like so:

ERCC-00002	52501.03	47122.57	45819.00	27185.80	29896.75	28304.06
ERCC-00003	4023.62	3160.91	3295.87	2371.43	2534.16	2231.15
ERCC-00004	1260.84	1074.91	1080.54	6254.92	7685.36	6157.32

The counts in the file above are the “normalized.” counts that the method operates on. As we mentioned before, “normalization” means that the numbers in this matrix were brought to be on the same scale and comparable to one another. It is worth looking at the original matrix to see what the initial values were.

ERCC-00002	37892	47258	42234	39986	25978	33998
ERCC-00003	2904	3170	3038	3488	2202	2680
ERCC-00004	910	1078	996	9200	6678	7396

98.5 How can I plot my normalized matrix?

First of all, plotting is the act of doing science - so no one can provide a generic, universally useful plot for you. You have to find the charting tool and technique that works for you and learn to use that — some people plot in R, others in Python, and so on. As you always need to tweak the plot, change colors, labels, etc., we can't give you a script that does it all.

But one particular plot, the clustered heatmap is so commonly used and useful, yet so challenging to achieve, that we included a script that does it for you. To run this script you will need to install the so called `gplots` package into the 'R' environment. Once that is done you can run the following:

```
curl -O http://data.biostarhandbook.com/rnaseq/code/draw-heatmap.r
```

This script produces a PDF output on the standard output. To plot your normalized matrix run it as:

```
cat norm-matrix-deseq1.txt | Rscript draw-heatmap.r > output.pdf
```

Note how this script also writes to standard out, though this time it writes a PDF file. What is in this PDF file? It is a hierarchically clustered heatmap image of your samples and differentially expressed genes:

Use this script to very quickly visualize what your data looks like.

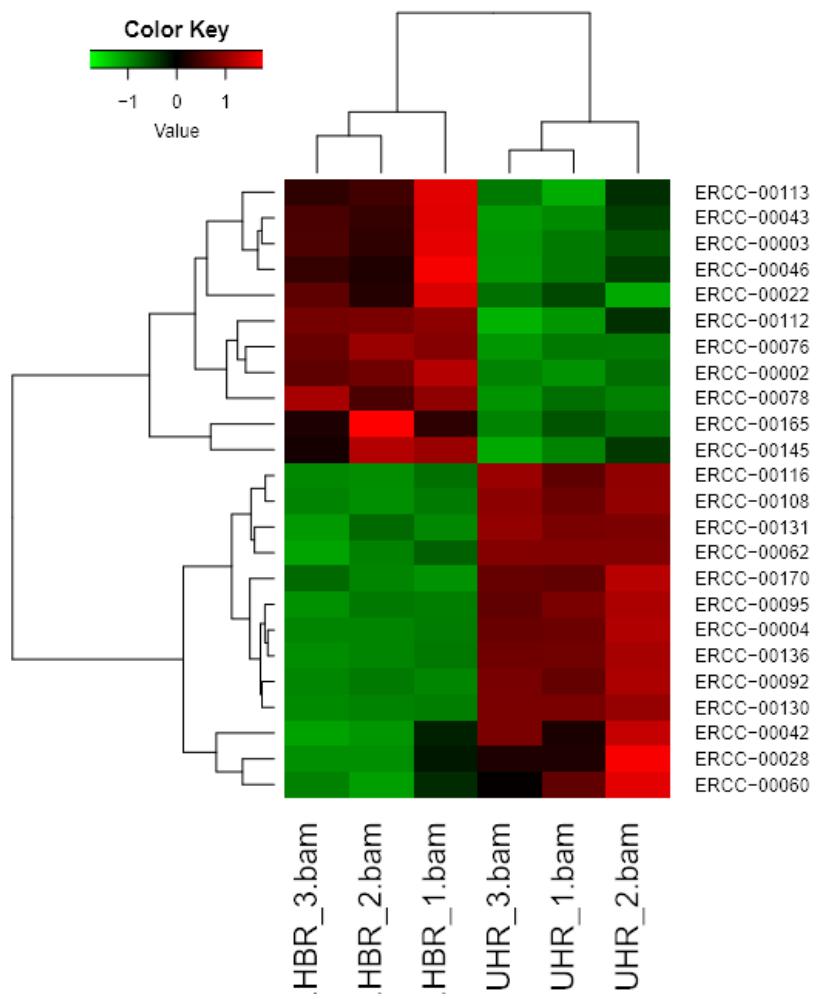


Figure 98.1

Chapter 99

The RNA-Seq puzzle

I thought up this problem while preparing a lecture on RNA-Seq data analysis. In the spur of the moment, I assigned it as a homework problem. Most students enjoyed solving it - and I received quite a bit of positive feedback. I think that it reads almost like a puzzle in the Sunday paper.

Later I came to believe that this puzzle provides the means to assess how well one understands RNA-Seq analysis.

When you can solve the puzzle it means that you know how RNA-Seq analysis works behind the scenes - what assumptions it makes and how various effects manifest themselves in the data.

99.1 How would I even solve this puzzle?

It is ok if you can't solve this puzzle yet. Read the next chapters, then come back here. We've list it early on to show you where we are going.

The purpose of this “puzzle” is to get you to think about what the numbers mean, and how they are linked together – and what it's like to obtain consistent and real data.

But, we admit it is also an unusual approach because it reverses the thought process. In a typical RNA-Seq analysis, you are given data, and you are asked to determine the gene expression from it. In this puzzle, you will be told what the reality is, what genes express relative to one another,

then you have to *make the data* that supports those statements. There are a number of constraints that need to be juggled. It is not unlike a sudoku puzzle actually.

Note – you don't need a computer to solve this problem: just paper and pencil or write the numbers into a text file.

99.2 The Pledge

Imagine that you have an organism that only has three distinct transcripts A, B, and, C.

- A, with a length of 10bp
- B, with a length of 100bp
- C, with a length of 1000bp

You want to study this organism under two conditions:

- Wild type: WT
- Heat shock: HEAT

You know from other sources that, within the WT condition, gene A expresses at levels that are twice as high as gene B.

You also know that only one transcript's expression level (but you don't know which) changes between WT and HEAT conditions. Assume that the change is substantial enough to be detectable. The other transcripts express at the same level in WT and HEAT.

Imagine that you have performed an RNA-Seq experiment to compare the wild-type WT and treatment HEAT - with just one replicate per experiment. Thus you end up with two experiments.

You have made one mistake, however. You have mixed the samples incorrectly, and you ended up placing twice as much DNA for the WT condition than for the treatment HEAT. You can still tell the samples apart since they are barcoded. You just mixed and sequenced twice as much DNA (mRNA) for WT as HEAT. There are twice as many reads sequenced for WT than for HEAT.

99.3 The Turn

Come up with the numbers for read coverage that represent the situation explained above. You can make up the numbers - the goal is to make them express what you know based on the description above. Create a 3x2 count table that shows the read counts. Each ? will need to have a number. Thus you have to come up with six numbers. That is the puzzle. Fill in the matrix below:

ID	WT	HEAT
A	?	?
B	?	?
C	?	?

99.4 The Prestige

Show that your numbers work. When you can answer them all, you understand how RNA-Seq works.

- How can you tell from your data that you placed twice as much WT material in the instrument?
- What is the CPM for each gene under each condition?
- What is the RPKM for each gene under each condition?
- What is the TPM for each gene under each condition?
- How can you tell that gene A expresses at twice the level of gene B within the WT sample?
- Can you tell which gene's expression level changes between WT and HEAT?
- Is the puzzle always solvable when correct values are specified in the “Turn”?

Now think about this:

- How many reads would you need to sequence for the CPM to be a “nice” number.
- How many reads would you need to sequence for the RPKM to be a “nice” number.
- How many reads would you need to sequence for the TPM to be a “nice” number.

- Does it make any sense to introduce measures like these above, that have arbitrary scaling factors, to make numbers look “nice”?

99.5 How to solve it (a hint)

As with a sudoku puzzle start filling in one number at a time and see if you can generate all the others accordingly.

Part XXII

RNA-SEQ EXAMPLE

Chapter 100

Understand your data

100.1 Which publication is reanalyzed?

This section will use data from the publication:

- Informatics for RNA-seq: A web resource for analysis on the cloud¹. 11(8):e1004393. PLoS Computational Biology (2015) by Malachi Griffith, Jason R. Walker, Nicholas C. Spies, Benjamin J. Ainscough, Obi L. Griffith.

An alternative tutorial is available online at https://github.com/griffithlab/rnaseq_tutorial/wiki

Note: We have greatly simplified the data naming and organization and processing. We recommend the original resource as an alternative tutorial and source of information.

100.2 What type of data is included?

The data consists of two commercially available RNA samples:

¹<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004393>

- Universal Human Reference (UHR) is total RNA isolated from a diverse set of 10 cancer cell lines.
- Human Brain Reference (HBR) is total RNA isolated from the brains of 23 Caucasians, male and female, of varying age but mostly 60-80 years old.

The data was produced in three replicates for each condition.

In addition to the biological samples the *also* contains two so called spike-in mixes: ERCC Mix 1 and ERCC Mix2. The spike-in consists of 92 transcripts that are present in known concentrations across a wide abundance range (from very few copies to many copies).

So to summarize this data consists of the following:

1. UHR + ERCC Mix1, Replicate 1, UHR_1
2. UHR + ERCC Mix1, Replicate 2, UHR_2
3. UHR + ERCC Mix1, Replicate 3, UHR_3
4. HBR + ERCC Mix2, Replicate 1, HBR_1
5. HBR + ERCC Mix2, Replicate 2, HBR_2
6. HBR + ERCC Mix2, Replicate 3, HBR_3

100.3 How do I download the example data?

A discussed above; this is a paired-end dataset with the experimental design of $2 \times 3 = 6$ within a paired sequencing format, we will have 12 files in total. Let's get the data (145MB):

```
# The URL the data is located at.
URL=http://data.biostarhandbook.com/rnaseq/projects/griffith/griffith-data.tar
```

```
# Downloading and unpack the data.
curl -s $URL | tar zxv
```

Once the process completes you will have three directories:

- **reads** containing the sequencing reads.
- **refs** containing genome and annotation information.

The data is ready for analysis. Read on to understand what information the data contains and what this script does. To display the data do

```
ls -1 reads
```

That produces:

```
HBR_1_R1.fq
HBR_1_R2.fq
HBR_2_R1.fq
HBR_2_R2.fq
HBR_3_R1.fq
HBR_3_R2.fq
UHR_1_R1.fq
UHR_1_R2.fq
UHR_2_R1.fq
UHR_2_R2.fq
UHR_3_R1.fq
UHR_3_R2.fq
```

The folder `refs` contains the annotations for the ERCC mixtures:

```
ls -1 refs
```

For this data, we will use a subset of the human genome as a reference. We will only use chromosome 22 to have examples complete much faster. Once the process appears to work correctly and we worked out the protocols we easily substitute chromosome 22 to the full human genome:

```
# The sequence information for human chromosome 22.
22.fa
```

```
# The genome annotations for chromosome 22.
22.gtf
```

```
# The sequence information for the ERCC mixture.
ERCC92.fa
```

```
# The genome annotation for the ERCC mixture.
ERCC92.gtf
```

Once you have the data, you may proceed to analyze it.

Chapter 101

Alignment based RNA-Seq of control samples

In this section, we will analyze only the “spike control” data of the research paper described before where the authors added so-called “spike-in” controls to each of their experimental conditions.

The three stages of RNA-Seq will be performed with:

1. Alignment: `hisat2`
2. Quantification: `featureCounts`
3. Differential expression: `DESeq`

101.1 What is a spike-in control?

The goal of the spike-in is to determine just how well can we measure and reproduce data with known (expected) properties. A standard product called the ERCC ExFold RNA Spike-In Control Mix¹ can be added in different mixtures.

This spike-in consists of 92 transcripts that are present in known concentrations across a wide abundance range (from very few copies to many copies). Here is an example:

¹<http://data.biostarhandbook.com/rnaseq/ERCC/ERCC-information.pdf>

Name	Mix1	Mix2	M1/M2	log2(M1/M2)
ERCC-00130	30000	7500	4	2
ERCC-00092	234	58	4	2
ERCC-00083	0.028	0.007	4	2
ERCC-00096	15000	15000	1	0
ERCC-00060	234	234	1	0
ERCC-00117	0.057	0.057	1	0
ERCC-00002	15000	30000	0.5	-1
ERCC-00079	58	117	0.5	-1
ERCC-00061	0.057	0.114	0.5	-1

If we were to add Mix 1 as condition 1 and Mix 2 as condition 2, then we would expect that our RNA-Seq experiment recovers the fold changes listed above.

For example for product ERCC-00130 we should observe a four fold change across the two mixes. In our case UHR had Mix 1, whereas HBR had Mix 2; thus UHR/HBR should show a four-fold increase if all is going well.

A	B	C	D	E	F
ERCC ID	subgroup	concentration in Mix 1	concentration in Mix 2 (expected fold-change ratio)	log2(Mix 1/Mix 2)	
ERCC-00130	A	30000.00	7500.00	4	2
ERCC-0004	A	7500.00	1875.00	4	2
ERCC-00136	A	1875.00	468.75	4	2
ERCC-00108	A	937.50	234.38	4	2
ERCC-00116	A	468.75	117.19	4	2
ERCC-00092	A	234.38	58.59	4	2
ERCC-00095	A	117.19	29.30	4	2
ERCC-00131	A	117.19	29.30	4	2
ERCC-00062	A	58.59	14.65	4	2
ERCC-00019	A	29.30	7.32	4	2
ERCC-00144	A	29.30	7.32	4	2
ERCC-00170	A	14.65	3.66	4	2
ERCC-00154	A	7.32	1.83	4	2

Figure 101.1

IMPORTANT: Note how in theory a 4-fold change could be observed for both highly expressed genes 30000/7500 or low expression genes 0.028/0.07. On the other hand, it should be clear that the latter will produce much far-far fewer measurements (reads); hence we expect that the method will deteriorate for transcripts of low expression levels. The question is at what concentration do we lose the ability to detect a fold change reliably.

Download and view the expected differential expressions from ERCC-datasheet.csv²

```
wget http://data.biostarhandbook.com/rnaseq/ERCC/ERCC-datasheet.csv
```

101.2 How do I align an RNA-seq sample?

We select a “splice aware” aligner, in this case, `hisat2` and build an index from our control sequences.

```
# This is the reference genome.
REF=refs/ERCC92.fa

# Name the prefix for the index the same as the REF.
IDX=refs/ERCC92.fa

# Build a hisat2 index for the genome.
hisat2-build $REF $IDX
```

For every script, our goal is to create it using “reusable” components. Reusability means that the resulting code can be easily adapted to another dataset. In this case, we want to set up our script in a way that we can run it on the spiked data and later on the real RNA-Seq dataset as well. To do that we need to consolidate the variable elements of our script into variables that we can set. Let’s also set up a few more shortcuts to simplify our commands:

```
# The sequencing data that contain the first in pair
R1=reads/UHR_1_R1.fq

# The sequencing data that contain the second in pair
R2=reads/UHR_1_R2.fq

# The name under which we store the BAM file.
BAM=bam/UHR_1.bam
```

To run our spliced aligner, we invoke it as:

```
# Make sure we have this folder.
```

²<http://data.biostarhandbook.com/rnaseq/ERCC/ERCC-datasheet.csv>

```

mkdir -p bam

# Align then sort and convert to BAM file.
hisat2 $IDX -1 $R1 -2 $R2 | samtools sort > $BAM

# Index the bam file.
samtools index $BAM

```

Note how generic and re-usable our construct has become:

```
hisat2 $IDX -1 $R1 -2 $R2 | samtools sort > $BAM
```

This command will run on any index and read pair, and we avoided the so-called “hard-coding” of information into the sections of the script that do the work. We can replace the read names, the bam file, yet the command stays the same. This is an approach is one that you should always strive for.

101.3 How do I automate my code for all samples?

The simplest possible solution could be just to list every command separately and put that information in a file that can be executed like so. It is a bit tedious and error-prone, but could be your first step:

```

# The name of the index
IDX=refs/ERCC92.fa

# Make the directory if it does not exist.
mkdir -p bam

R1=reads/UHR_1_R1.fq
R2=reads/UHR_1_R2.fq
BAM=bam/UHR_1.bam
hisat2 $IDX -1 $R1 -2 $R2 | samtools sort > $BAM

R1=reads/UHR_2_R1.fq
R2=reads/UHR_2_R2.fq
BAM=bam/UHR_2.bam

```

```

hisat2 $IDX -1 $R1 -2 $R2 | samtools sort > $BAM

R1=reads/UHR_3_R1.fq
R2=reads/UHR_2_R2.fq
BAM=bam/UHR_3.bam
hisat2 $IDX -1 $R1 -2 $R2 | samtools sort > $BAM
...

```

and so on. It is redundant, but at least it is very explicit in what it does. The command itself does not change, only the data names that it gets executed upon.

The primary danger with the approach is that you make an error, for example, you forget to change one or more of the file names then you end up using the wrong data. *Hey, did you notice the error that we made above? Look again!*

Since you've stored it in a file, you can inspect it, double-check for errors, and re-run at any time. While it is a simplistic approach, it is an excellent start to creating a reusable workflow. You don't have to write a more complex script unless you want to.

101.4 How to better automate the process?

Our first choice in automation is the `parallel` program. It has been designed for the exact use cases we have in mind.

Here is also where [proper sample naming](#howtoname) help greatly. Note the naming `UHR_1_R1.fq` we have UHR and HBR then replicates and then read pairs. Lets look at the file listing again:

```
ls -1 reads/UHR*.fq
```

Above the '`-1`' is minus one not little el and will list:

```

reads/UHR_1_R1.fq
reads/UHR_1_R2.fq
reads/UHR_2_R1.fq
reads/UHR_2_R2.fq
reads/UHR_3_R1.fq
reads/UHR_3_R2.fq

```

Other tutorials you may read will sometimes recommend that you run the commands on file listings for example:

```
ls -1 reads/*.fq | parallel echo sometool {}
```

then if necessary modify the paths. We strongly disagree with this approach. What you will usually end up with is an overcomplicated script that is “brittle” and error-prone because as it critically depends on the order of the file listing.

To solve the problem elegantly, we want just the “roots” for each name, from which we can then form whichever name we want.

```
UHR_1  
UHR_2  
UHR_3
```

You can generate these roots by hand (sometimes that’s the quickest), or you can generate them with `parallel` like so:

```
parallel -j 1 echo {1}_{2} :::: UHR HBR :::: 1 2 3 > names.txt
```

above we instruct `parallel` to process the parameters we gave it, to produce a file that contain:

```
UHR_1  
UHR_2  
UHR_3  
HBR_1  
HBR_2  
HBR_3
```

Using these root names, we can now automate the alignment into one single elegant command:

```
# Make a directory for BAM files.  
mkdir -p bam
```

```
# Run all samples in parallel.  
cat names.txt | parallel "hisat2 $IDX -1 reads/{}_R1.fq -2 reads/{}_R2.fq | samtools sort
```

With that, we have just created a pipeline that churns through all 12 files in a repeatable manner. We can modify the alignment command if we wish so and re-run everything with ease. The result will be six bam files in

the `bam` folder named `UHR_1.bam`, `UHR_2.bam`, `UHR_3.bam` and `HBR_1.bam`, `HBR_2.bam`, `HBR_3.bam`.

101.5 How do I estimate the abundance for a single sample?

We evaluate the abundance by counting the number of alignments that fall over a specified interval. Over the years a great many tools have been proposed, but our all-time favorite is `featureCounts`, a program that takes as input a gene feature file and one (or many) `bam` files.

```
# These are the coordinates of the genes.
GTF=refs/ERCC92.gtf
```

```
featureCounts -a $GTF -o counts.txt bam/HBR_1.bam
```

By default the `featureCounts` program uses the `gene_id` attribute in the GTF file. We can override that and instruct `featureCounts` to use the `gene_name` attribute instead with:

```
featureCounts -a $GTF -g gene_name -o counts.txt bam/HBR_1.bam
```

The resulting file `counts.txt` is a tab-delimited file where the first six columns contain feature specific information, and the rest of the columns hold the read counts that overlap with that feature.

Geneid	Chr	Start	End	Strand	Length	bam/HBR_1.bam
ERCC-00002	ERCC-00002	1	1061	+	1061	37892
ERCC-00003	ERCC-00003	1	1023	+	1023	2904
ERCC-00004	ERCC-00004	1	523	+	523	910
ERCC-00009	ERCC-00009	1	984	+	984	638

To find the sequences with most hits, we can sort by column 7:

```
cat counts.txt | sort -rn -k 7 | head
```

This produces:

ERCC-00002	ERCC-00002	1	1061	+	1061	37892
ERCC-00074	ERCC-00074	1	522	+	522	20982
ERCC-00096	ERCC-00096	1	1107	+	1107	20708
ERCC-00130	ERCC-00130	1	1059	+	1059	8088

101.6. ARE THERE DIFFERENT WAYS TO COUNT OVERLAPS? 793

```
ERCC-00113    ERCC-00113    1     840      +     840      7096
```

We can see how the counts in the last column correlate very closely with the expected abundances for Mix2 in the ECC Mix file.

101.6 Are there different ways to count overlaps?

Yes, there are.

As with many other tools counting how many reads overlap with a feature is also fraught with subjectivity. We may or may not include reads that overlap only partially or reads that align to more than one feature. We may also choose to count both pairs as one count or as two counts and so on. In some cases, these choices don't make any difference; in others they do.

101.7 How do I estimate abundances for all samples?

We can list more than one bam file for `featureCounts`. For example, using the shell metacharacter '*' we can list all HBR and all UHR samples in a compact form:

```
featureCounts -a $GTF -g gene_name -o counts.txt bam/HBR*.bam bam/UHR*.bam
```

In this case, the `counts.txt` file will contain a column for each of the samples, for a total of 13 columns. Note how each sample will have the read counts listed in each row that corresponds to each feature of the file.

Geneid	Chr	Start	End	Str	Len	HBR_1	HBR_2	HBR_3	UHR_1	UHR_2	UHR_3
ERCC-00002	ERCC-00002	1	1061	+	1061	37892	47258	42234	39986	25978	33998
ERCC-00003	ERCC-00003	1	1023	+	1023	2904	3170	3038	3488	2202	2680
ERCC-00004	ERCC-00004	1	523	+	523	910	1078	996	9200	6678	7396
...											

All that is left is to compare the replicates for HBR to the replicates in UHR to find those features for which the replicates within one condition are consistently different from those in the other state.

101.8 How do I find differential expression?

NOTE: All statistical methods rely on multiple assumptions regarding the characteristics of the data.

In general, in the vast majority of the users of statistical methods are unaware of these assumptions - and not always by their fault. It often takes a surprising amount of effort to locate, identify and understand the information describing the limitation of each method. You see scientists are typically not all that eager to talk about the restrictions. In this particular case, the number of features is very low, and most of the transcript do exhibit changes. These properties violate many of the assumptions that statistical methods rely on - hence the results of the statistical test will be far less reliable. But it is never clear how less reliable are these results, a little bit? A lot? all nonsense? More than ever before a critical eye and common sense are needed.

We have set up several R scripts that can produce differential expression computation for you using count information similar to the one that the `featureCounts` program produces. Get our R script as described on RNA-Seq with Bioconductor page:

```
wget -q -nc http://data.biostarhandbook.com/rnaseq/code/deseq1.r
```

The script expects a file that contains only gene names and counts. We need to remove the intermediate columns.

```
cat counts.txt | cut -f 1,7-12 > simple_counts.txt
```

So that our file now is just:

ERCC-00002	37892	47258	42234	39986	25978	33998
ERCC-00003	2904	3170	3038	3488	2202	2680
ERCC-00004	910	1078	996	9200	6678	7396
ERCC-00009	638	778	708	1384	954	1108
...						

Then pass this file through the script by specifying the design of the experiment in this case Three replicates for each of the two conditions so 3x3.

```
cat simple_counts.txt | Rscript deseq1.r 3x3 > results.txt
```

The `Rscript` command launches R with our `deseq1.r` script that takes its

input from the left and writes into `results.txt`.

101.9 What does a differential expression file look like?

A differential expression file describes the changes in gene expression across two conditions. It will be similar to:

<code>id</code>	<code>baseMean</code>	<code>baseMeanA</code>	<code>baseMeanB</code>	<code>foldChange</code>	<code>log2FoldChange</code>	<code>pval</code>	<code>padj</code>
ERCC-00130	29681	10455	48907	4.67	2.22	1.16e-88	9.10e-87
ERCC-00108	808	264	1352	5.10	2.35	2.40e-62	9.39e-61
ERCC-00136	1898	615	3180	5.16	2.36	2.80e-58	7.30e-57

Typically you get a column (though the naming will depend on the tool you use) for each of the following:

- `id`: Gene or transcript name that the differential expression is computed for,
- `baseMean`: The average normalized value across all samples,
- `baseMeanA`, `baseMeanB`: The average normalized gene expression for each condition,
- `foldChange`: The ratio `baseMeanB/baseMeanA`,
- `log2FoldChange`: \log_2 transform of `foldChange`. When we apply a 2-based logarithm the values become symmetrical around 0. A \log_2 fold change of 1 means a doubling of the expression level, a \log_2 fold change of -1 shows a halving of the expression level.
- `pval`: The probability that this effect is observed by chance,
- `padj`: The adjusted probability that this effect is observed by chance.

You may use `pval` if you already selected your target gene before evaluating these results.

You have to use `padj` in all other cases as this adjusted value corrects for the so-called multiple testing error - it accounts for the many alternatives and their chances of influencing the results that we see.

101.10 Did our RNA-Seq analysis reproduce the expected outcomes?

Let's look at the first few rows. ERCC-00130 was generated at the highest concentration (highest gene expression) at a fold change of 4. We obtain a fold change of 4.67 so we overshot it, though not by much.

ERCC-00108 was also deposited at a fold change of 4 but since it has 30 times smaller concentration, we expect less accuracy. Indeed we observe a fold change of 5.10 for it. We can line up the expected table with the obtained one like so:

```
# We need to sort data by ID so that can be pasted in columns.
curl http://data.biostarhandbook.com/rnaseq/ERCC/ERCC-datasheet.csv | grep ERCC-
cat results.txt | grep ERCC- | sort | cut -f 1,5,6 > table2

# Joining the tables that contain both datasets for comparison
paste table1 table2 > compare.txt
```

The results are summarized in an Excel table that you too can download from ERCC-results.csv³

In summary, for a fold change of 4 or higher, transcripts expressing over as much as 1000x coverage differences were reliably detected. Note how transcripts with 3000 copies were detected as reliably as transcripts with seven reads, though clearly, the accuracy of fold change detection varies.

Note how for data with insufficient measures we get infinite or NaN (Not a Number) values are indicating a division by zero or other numeric overflows.

³<http://data.biostarhandbook.com/rnaseq/ERCC/ERCC-results.csv>

ERCC ID	Mix 1	Mix 2	Expected Fold Change	Measured Fold Change	Error
ERCC-00130	30000	7500	4	4.7	0.7
ERCC-00004	7500	1875	4	5.9	1.9
ERCC-00136	1875	468.75	4	5.2	1.2
ERCC-00108	937.5	234.375	4	5.1	1.1
ERCC-00116	468.75	117.1875	4	4.6	0.6
ERCC-00092	234.375	58.59375	4	5.4	1.4
ERCC-00095	117.1875	29.296875	4	4.4	0.4
ERCC-00131	117.1875	29.296875	4	3.8	-0.2
ERCC-00062	58.59375	14.6484375	4	5.8	1.8
ERCC-00019	29.296875	7.32421875	4	2.8	-1.2
ERCC-00144	29.296875	7.32421875	4	2.4	-1.6
ERCC-00170	14.6484375	3.66210938	4	6.2	2.2
ERCC-00085	7.32421875	1.83105469	4	4.1	0.1
ERCC-00154	7.32421875	1.83105469	4	4.9	0.9
ERCC-00028	3.66210938	0.91552734	4	6.3	2.3
ERCC-00033	1.83105469	0.45776367	4	Inf	NaN
ERCC-00134	1.83105469	0.45776367	4	0	-4
ERCC-00147	0.91552734	0.22888184	4	NaN	NaN
ERCC-00097	0.45776367	0.11444092	4	Inf	NaN
ERCC-00156	0.45776367	0.11444092	4	NaN	NaN

Figure 101.2

Chapter 102

Alignment based RNA-Seq of biological data

In this section, we will analyze the full dataset of the research paper described before. We also assume that you have studied the Analyzing RNA-Seq control samples section as we will build upon on the code presented therein.

102.1 How do I adapt scripts to new data?

The most important feature of any automation is to isolate the variable parts of the analysis from the non changing segments. Move the variables of the analysis to the beginning of the script where you can quickly and reliably adapt the script to new data.

Note how easy it is to adapt a script that operates on control data to the real data. All we needed to do is edit these two lines to read:

```
# This is the new name of the index  
IDX=refs/22.fa  
  
# These are the new genomic genes.  
GTF=refs/22.gtf
```

And the same code from the previous section would run for a different genome and annotation file.

102.2 How do I generate the list of differentially expressed features?

The same steps need to be followed as in the previous example.

```
# Perform the feature counts in paired end mode.
featureCounts -p -a $GTF -g gene_name -o counts.txt bam/U*.bam bam/H*.bam

# Simplify the counts
cat counts.txt | cut -f 1,7-12 > simple_counts.txt

# Perform the differential expression counting.
cat simple_counts.txt | Rscript deseq1.r 3x3 > results.txt
```

Visualizing chromosome 22 in IGV, we can see the following for gene MAPK8IP2 located chr22:50600685–50613981 (type the gene name MAPK8IP2 into the IGV coordinate box):



Figure 102.1

From the image, it is clear that this gene has systematically more data in the HBR samples than the UHR samples. The next sections explain how we can find these regions automatically.

102.3 What does the differential expression file look like?

A differential expression file describes the changes in gene expression across two conditions. It will be similar to:

```
id      baseMean  baseMeanA  baseMeanB  foldChange  log2FoldChange  pval      padj
SYNGR1  526.8    1012.4    41.2       0.04        -4.61        1.92e-277  1.86e-274
...

```

I have rounded up the numbers above to make them more readable. The table columns mean the following:

- Gene or transcript name that the differential expression is computed for `id`.
- The average normalized value across all samples `baseMean`
- The average normalized gene expression for each condition `baseMeanA`, `baseMeanB`.
- The ratio of values called `foldChange` computed as `baseMeanB/baseMeanA`.
- A log2 transform of `foldChange` called `log2FoldChange`. When we apply a 2 based logarithm then the values become symmetrical around 0. A log2 fold change shown of 1 and -1 will show a doubling or halving of the expression levels.
- A probability that this effect is observed by chance: `pval`.
- An adjusted probability that this effect is observed by chance: `padj`.

You may use `pval` if you already had selected your target gene before evaluating these results.

You have to use `padj` in all other cases as this adjusted value corrects for the so-called multiple testing error - it accounts for the many alternatives and their chances of influencing the results that we see.

The genes MAPK8IP2, SYNGR1, SEPT3 and 293 more appear to be expressed at different levels across our experimental conditions.

102.4 How do I interpret the results?

The `results.txt` contains the genes sorted by their adjusted p-values (last column). Imposing a filter to this column, for example selecting genes that

have changed at 0.05 level:

```
cat results.txt | awk '$8 < 0.05 { print $0 }' > diffgenes.txt

# How many differentially expressed genes do we have?
cat diffgenes.txt | wc -l
# 293
```

At this step, you go ahead and perform **Gene Enrichment** study as shown in the previous chapters.

Chapter 103

Classification based RNA-Seq of control samples

Pseudoalignments are a concept introduced to the world by the work of **Rob Patro** and **Carl Kingsford** at the Lane Center for Computational Biology at Carnegie Mellon University. In a nutshell, pseudo-alignment-based methods identify locations in the genome using patterns rather than via alignment type algorithms. The advantage of pattern matching is that it operates many orders of magnitude faster than alignment and makes it possible to analyze massive datasets even with much lower resources.

After the introduction of the concept, the method was extended to RNA-Seq data analysis. Different implementations of the technique exist, Kallisto and Salmon, with others probably in development.

103.1 What are Kallisto and Salmon?

Kallisto¹ and Salmon² are software packages from different authors for quantifying transcript abundances. The tools perform a *pseudo-alignment* of reads against a transcriptome. In pseudo-alignment, the program tries to identify for each read the target that it originates from.

¹<https://pachterlab.github.io/kallisto/>

²<https://combine-lab.github.io/salmon/>

103.2. WHAT IS THE MAIN DIFFERENCE BETWEEN ALIGNMENTS AND CLASSIFICATION BASED RNA-SEQ?

103.2 What is the main difference between alignments and classification based RNA-Seq?

To perform a classification based RNA-Seq, we need to know the transcriptome of the organism. A classification will only produce information on the known entries. Classification based methods are typically much faster than alignment methods.

Alignment-based methods work on both genome or transcriptome. Alignment-based methods may be used to discover novel (unannotated) transcripts.

103.3 Where do we start?

First, obtain the data as described in the section First understand your data. Once the information is downloaded proceed with steps below.

103.4 How do I build a Kallisto index?

You may need to first install `kallisto` with:

```
conda install kallisto
```

Kallisto uses a transcriptome as the reference. We can build a Kallisto index for ERCC control sequence as shown below.

```
# This is the reference.  
REF=refs/ERCC92.fa  
  
# This is the name of the index  
IDX=refs/ERCC92.idx  
  
# Build kallisto index  
kallisto index -i $IDX $REF
```

103.5 How do I quantify transcripts with Kallisto?

Kallisto assigns reads to transcripts and calculates their abundance in one step. The algorithm can be invoked using `quant` command.

To quantify the paired-end data in our example

```
R1=reads/HBR_1_R1.fq
R2=reads/HBR_1_R2.fq

# Kallisto will generate multiple outputs.
OUTDIR=results

# Run kallisto quantification.
kallisto quant -i $IDX -o $OUTDIR $R1 $R2
```

The above command produces output files in the output directory called `out` specified by the `-o` option.

103.6 What are the files that `kallisto` produces?

A `kallisto` run produces three files:

1. `abundance.tsv`
2. `abundance.h5`
3. `run_info.json`

The `abundance.tsv` is of central interest, it is a tab delimited file that, for each transcript lists the target, the length, the corrected effective length, the counts and the TPM value.

target_id	length	eff_length	est_counts	tpm
ERCC-00002	1061	891.47	18946	243187
ERCC-00003	1023	853.47	1452	19467.4
ERCC-00004	523	353.47	455	14729.5
ERCC-00009	984	814.47	319	4481.72

Annoyingly and shortsightedly each run of kallisto will produce identically named files; thus these files will need to be placed in separate directories. Therefore more bookkeeping will be necessary to manage and rename the files.

103.7 How do I quantify all samples with Kallisto?

You can automate running kallisto with parallel:

```
# Make a directory for BAM files.
mkdir -p results

# Create the root names for each file.
parallel -j 1 echo {1}_{2} :::: UHR HBR :::: 1 2 3 > names.txt

# Run all samples in parallel.
cat names.txt | parallel "kallisto quant -i $IDX -o results/{} reads/{}_R1.fq reads/{}_R2.fq"
```

The commands above generate a directory called **results** that in turn, contains other directories, each named by the samples.

As we noted elsewhere, this is an improper way to create data as critical information ends up stored in the directory rather than the file name.. A file named **results/HBR_1/abundance.tsv** and one **results/HBR_1/abundance.tsv** contains information different samples, but note how you could not tell that from the file name alone. In both cases, the file is called **abundance.tsv**.

It is a terrible (yet common) practice that we dearly wish would go away. To fix this, right away we'll copy the abundance file into an unambiguous name **HBR_1.tsv**:

```
cat names.txt | parallel cp results/{}/abundance.tsv results/{}.abundance.tsv
```

to see what files the command above produces type:

```
ls -1 results/*.tsv
```

it prints:

```
results/HBR_1.abundance.tsv
```

```
results/HBR_2.abundance.tsv
results/HBR_3.abundance.tsv
results/UHR_1.abundance.tsv
results/UHR_2.abundance.tsv
results/UHR_3.abundance.tsv
```

Finally, we have one abundance file per sample.

103.8 How to interpret the files?

There are various alternatives to processing the abundance files. The approach we follow below is to merge all counts into a single count file, then cut the columns that correspond to estimated counts (we had to investigate the merged file to identify the column numbers manually):

```
paste results/H*.tsv results/U*.tsv | cut -f 1,4,9,14,19,24,29 > counts.txt
```

The count file `counts.txt` contains the information listed below. The file can be used as input to downstream DEseq differential expression analysis.

target_id	est_counts	est_counts	est_counts	est_counts	est_counts	est_counts
ERCC-00002	18946	23629	21117	19993	12989	16999
ERCC-00003	1452	1585	1519	1744	1101	1340
ERCC-00004	455	539	498	4600	3339	3698
ERCC-00009	319	389	354	692	477	554

You may change the headers to include the sample names.

103.9 How do I run a differential expression study?

Note that once you have a file that contains counts for each transcript, you may reuse the same methods developed to analyze counts derived with alignments.

For example:

```
# Get the deseq1.r script.
wget -q -nc http://data.biostarhandbook.com/rnaseq/code/deseq1.r
```

103.9. HOW DO I RUN A DIFFERENTIAL EXPRESSION STUDY? 807

```
# Generate differential expression from the counts.  
cat counts.txt | Rscript deseq1.r 3x3 > results.txt
```

Part XXIII

RNA-SEQ ZIKA

Chapter 104

Understand the Zika data

We have discussed the original publication in the chapter titled Zika virus targets human cortical neural precursors. As a reminder the original paper is:

- Zika Virus Targets Human Cortical Neural Precursors and Attenuates Their Growth¹ in Cell Stem Cell. 2016 May 5
- NCBI BioProject: PRJNA313294² and GEO: GSE78711³

The data from this publication was later re-analyzed by two other research groups:

- An open RNA-Seq data analysis pipeline tutorial with an example of reprocessing data from a recent Zika virus study⁴ F1000 Research, 2016
- Zika infection of neural progenitor cells perturbs transcription in neurodevelopmental pathways⁵ PLoS One, 2017

104.1 What data does the project contain?

First obtain the project runinfo:

¹<https://www.ncbi.nlm.nih.gov/pubmed/26952870>

²<https://www.ncbi.nlm.nih.gov/bioproject/PRJNA313294>

³<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE78711>

⁴<https://f1000research.com/articles/5-1574/v1>

⁵<http://biorxiv.org/content/early/2016/10/14/072439>

```
esearch -db sra -query PRJNA313294 | efetch -format runinfo > runinfo.csv
```

From the publication, we know that the authors use a control (mock) and an infected sample and compared the two. Annoyingly enough essential information regarding which sample is infected and which one is control was not made available in the run info file! It is only available in the so-called docsum format (or by wading through the supplementary information)

```
esearch -db sra -query PRJNA313294 | efetch -format docsum > runinfo.xml
```

The docsum format is complicated and convoluted making it challenging to interpret. It may take a few hours of additional work to figure out which sample is which. Now you see how the shortsightedness of the scientific process makes us “bleed time”, our most valuable resource. Instead of going ahead with analyzing the data we have to spend precious mental resources on untangling a data annotation mess. Hint: view the XML file in a browser to see its structure, alternatively you can run:

```
cat runinfo.xml | xtract -outline
```

to see how the document elements are nested. After fiddling with the format for some time, we found that running:

```
cat runinfo.xml | xtract -Pattern DocumentSummary -element Run@acc,Title
```

gives us:

```
SRR3194431  GSM2075588: ZIKV2-2; Homo sapiens; RNA-Seq  
SRR3194430  GSM2075587: ZIKV1-2; Homo sapiens; RNA-Seq  
SRR3194429  GSM2075586: Mock2-2; Homo sapiens; RNA-Seq  
SRR3194428  GSM2075585: Mock1-2; Homo sapiens; RNA-Seq  
SRR3191545  GSM2073124: ZIKV2-1; Homo sapiens; RNA-Seq  
SRR3191544  GSM2073123: ZIKV1-1; Homo sapiens; RNA-Seq  
SRR3191543  GSM2073122: Mock2-1; Homo sapiens; RNA-Seq  
SRR3191542  GSM2073121: Mock1-1; Homo sapiens; RNA-Seq
```

Finally some light at the end of the tunnel though, incredibly, as it turns out not even the docsum file has a field indicating an infection or a control (mock) status. The sample status is embedded into the title. We'll count our losses and move on.

After more investigation of the `runinfo.csv` and `runinfo.xml`, we finally concluded that the dataset consists of two separate instrumental runs, RNA-Seq libraries of zika infected (treatment) and mock infected (control) hu-

man neural progenitor cells (hNPCs). 75 base pair paired-end reads were sequenced using both an Illumina MiSeq and an Illumina NextSeq. Besides, we have two replicates for both mock and zika samples.

Thus we have a paired end run:

Sample	Accession	Condition	Library-type	Seq-machine
Mock1-1	SRR3191542	mock	paired-end	MiSeq
Mock2-1	SRR3191543	mock	paired-end	MiSeq
ZIKV1-1	SRR3191544	zika	paired-end	MiSeq
ZIKV2-1	SRR3191545	zika	paired-end	MiSeq

Also, the same data was sequenced again using an Illumina NextSeq in 75 bp single-end mode.

Sample	Accession	Condition	Library-type	Seq-machine
Mock1-2	SRR3194428	mock	single-end	NextSeq
Mock2-2	SRR3194429	mock	single-end	NextSeq
ZIKV1-2	SRR3194430	zika	single-end	NextSeq
ZIKV2-2	SRR3194431	zika	single-end	NextSeq

An unexpectedly complicated picture emerged right from the get-go. Save this information in a document that you can refer to later.

104.2 How to obtain the data?

The data for this paper is relatively large and may need a few hours to download. The analysis, however, can be performed much faster.

We have placed the commands needed to obtain the data into the script called [zika-data.sh[zika-data]] that you can download and run yourself:

```
wget http://data.biostarhandbook.com/redo/zika/zika-data.sh
```

then run it with:

```
bash zika-getdata.sh
```

Investigate the file yourself, the only complexity there is that we need to unpack the single and paired-end reads separately.

The full data is of the size of about 20GB. While the download only needs to take place once, it may take some time depending on the internet bandwidth that you have access to.

Note: By default, only 1 million reads are extracted. Edit the `zika-getdata.sh` script to change the limit. You may raise the limit to 100 million or remove the limit from `fastq-dump` altogether. When downloading the full dataset running the —at this point the chapter actually ends in the pdf!!

Chapter 105

What will the accession number files contain?

Two files created by the `zika-getdata.sh` script will be used frequently. Let us list their contents here to help with the process of understanding them:

The `runinfo.single.csv` that will be stored in the `SINGLE` variable:

```
cat runinfo.single.csv
```

will contain those SRR numbers that correspond to single end runs (two mock and two infected samples):

```
SRR3194428  
SRR3194429  
SRR3194430  
SRR3194431
```

The `runinfo.paired.csv` that will be stored in the `PAIRED` variable file contains:

```
cat runinfo.paired.csv
```

will contain those SRR numbers that correspond to paired end end runs (two mock and two infected samples):

```
SRR3191542  
SRR3191543  
SRR3191544  
SRR3191545
```

105.1 How much data is there in total?

When the data is not limited the command:

```
seqkit stat reads/*
```

produces the following output.

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
reads/SRR3191542_1.fastq	FASTQ	DNA	7,927,777	598,459,167	35	75.5	76
reads/SRR3191542_2.fastq	FASTQ	DNA	7,927,777	598,471,205	35	75.5	76
reads/SRR3191543_1.fastq	FASTQ	DNA	7,391,076	557,684,818	35	75.5	76
reads/SRR3191543_2.fastq	FASTQ	DNA	7,391,076	557,625,834	35	75.4	76
reads/SRR3191544_1.fastq	FASTQ	DNA	7,361,527	555,374,256	35	75.4	76
reads/SRR3191544_2.fastq	FASTQ	DNA	7,361,527	555,388,034	35	75.4	76
reads/SRR3191545_1.fastq	FASTQ	DNA	7,621,347	574,157,249	35	75.3	76
reads/SRR3191545_2.fastq	FASTQ	DNA	7,621,347	574,145,168	35	75.3	76
reads/SRR3194428.fastq	FASTQ	DNA	72,983,243	5,503,017,518	35	75.4	76
reads/SRR3194429.fastq	FASTQ	DNA	94,729,809	7,137,391,714	35	75.3	76
reads/SRR3194430.fastq	FASTQ	DNA	76,299,868	5,747,471,557	35	75.3	76
reads/SRR3194431.fastq	FASTQ	DNA	66,528,035	5,008,831,278	35	75.3	76

We can see that the paired-end files contain far fewer reads than single-end reads (7 million vs 70 million). We can also see that the read lengths appear to vary – meaning some length trimming was applied to these datasets.

105.2 How do I analyze data generated on different sequencing platforms?

Having two rounds of sequencing in the dataset leads to questions about how to analyze such a dataset - Should we pool the data from the two runs or should we keep them separate and treat them as another replicate? Another difficulty here – since the dataset consists of both paired-end and single-end samples, commands need to be run separately for each.

Since the question of combining data can be postponed to the interpretation phase we'll postpone this decision for later.

Chapter 106

Alignment based RNA-Seq of Zika data

106.1 How to get the reference genomes?

We want to align the Zika sequencing data to the entire human genome and transcriptome. Since these reference files are large and can be used in various projects the typical choice is to store them in a directory outside of the current project's location. This way we separate project specific data from data of generic use.

Besides, whereas for small genomes we can create indices within minutes, for large human size genomes the process may take days. Since the genome indices are the same regardless of who builds them, it makes sense to download these indices if these are available.

For this analysis, we chose the HiSat2¹ splice-aware aligner for mapping next-generation sequence data against a reference genome. It uses similar algorithms as Bowtie2 but has much-improved speed. The authors of HiSat2 do provide prebuilt indices for several genomes.

We have combined the commands to download the reference genomes into the zika-references.sh² a script that you would need to run once.

¹<https://ccb.jhu.edu/software/hisat2/manual.shtml>

²<http://data.biostarhandbook.com/redo/zika/zika-references.sh>

```
wget http://data.biostarhandbook.com/redo/zika/zika-references.sh
```

Investigate the file yourself, and you might want to save it for further use. For any project that you undertake, you should ensure that you keep the script that prepares the reference data. Very frequently you will read scientists stating: “we have downloaded the transcriptome data from Ensembl” - an insufficiently specific statement. Having a script that lists the precise and exact resources helps both you now, the future you, and your readers.

The download script contains simple downloads from various URLs, then it unpacks the data that was downloaded.

```
bash zika-references.sh
```

Further steps of this and future chapters assume that you have run the script and obtained the reference datasets.

106.2 What will be done in this section?

The three stages of our RNA-Seq will be:

1. Alignment: HISAT2 to produce a BAM file per each sequencing run
2. Quantification: `featureCounts` to count how many reads in each BAM file overlap with a genomic feature
3. Differential expression: `DESeq` to process the count file to select rows where the differences between conditions appear to be statistically significant.

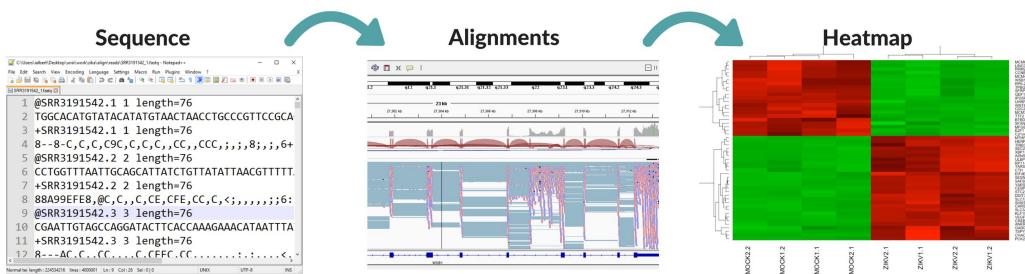


Figure 106.1

106.3 What are the steps to align the data?

Next, make sure that you have obtained and understood the structure of the Zika data as described in the chapter Understand the Zika data. Recall that you can obtain the data by running the data collection script zika-data.sh³. Since the data unpacking may also be time-consuming it is also best to keep it in a separate script. Thus to perform the analysis, you already have two scripts:

- zika-reference.hs⁴ to download the references
- zika-data.hs⁵ to download the sequencing data from SRA

We reuse the variable naming of that script and continue with the alignment phases.

```
# Full run information.
RUNINFO=runinfo.all.csv

# Run info for single end reads.
SINGLE=runinfo.single.csv

# Run information for paired end reads.
PAIRED=runinfo.paired.csv

# Store bam files here.
mkdir -p bam

# How many CPU cores on the system.
CPUS=4

# The hisat index name.
IDX=~/refs/grch38/genome

# Run hisat2 on the paired data.
cat $PAIRED | parallel "hisat2 -p $CPUS -x $IDX -1 reads/{}_1.fastq -2 reads/{}_2.
```

³<http://data.biostarhandbook.com/redo/zika/zika-data.sh>

⁴<http://data.biostarhandbook.com/redo/zika/zika-references.sh>

⁵<http://data.biostarhandbook.com/redo/zika/zika-data.sh>

```
# Run hisat2 on the single end data.
cat $SINGLE | parallel "hisat2 -p $CPUS -x $IDX -U reads/{}.fastq | samtools sort > bam/{}"

# Run a samtools index on all BAM files
ls -1 bam/*.bam | parallel samtools index {}
```

The bam files will be collected in the folder called `bam`. You can visualize these alignments with IGV choosing the build 38.



Figure 106.2

If the above statements confuse you, or you get errors stop using the script! Don't try to edit here and there, try to fix it up until it runs unless you have identified the problem. Otherwise, you will only end up with what is called "*programming by coincidence*" a situation where you may stumble upon a seemingly working code without understanding neither the error nor the solution. That won't help you long term.

Instead, if the above does code not make sense, study the chapter Alignment based RNA-Seq of control samples)(#rnaseq-align-control) and write out each command fully yourself. Do it without using `parallel` or even any variable name.

Another strategy is to break the commands into sections and place an `echo` before the command to see what it will attempt to execute. For example writing:

```
cat runinfo.paired.csv | parallel echo hisat2 -p $CPUS -x $IDX -1 reads/{}_1.fastq -2 reads/{}_2.fastq
```

will now print the commands instead of executing them:

```
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191542_1.fastq -2 r
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191543_1.fastq -2 r
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191544_1.fastq -2 r
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191545_1.fastq -2 r
```

Now troubleshoot the lines, assess any error that might occur. When you have special symbols like | or > the echo will be redirected, so when echoing you would also need to escape these with backslash \. For example, you would need to write:

```
cat runinfo.paired.csv | parallel echo "hisat2 -p $CPUS -x $IDX -1 reads/{}_1.fastq -2 r
```

Doing so will print the entire command that the tool is attempting to execute:

```
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191542_1.fastq -2 r
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191543_1.fastq -2 r
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191544_1.fastq -2 r
hisat2 -p 4 -x /Users/ialbert/refs/grch38/genome -1 reads/SRR3191545_1.fastq -2 r
```

106.4 How do I generate feature counts?

As you recall from the data section, the mock samples are SRR3191542, SRR3191543, SRR3194428 and SRR3194429 whereas the infected samples are SRR3191544, SRR3191545 SRR3194430 and SRR3194431. You may list the bam files at the command line like so:

```
# Shortcut to annotations
GTF=~/refs/Homo_sapiens.GRCh38.96.chr.gtf

# Run feature counts and count overlaps for gene names.
featureCounts -g gene_name -a $GTF -o counts.txt bam/SRR3191542.bam bam/SRR3191543.bam
```

Listing all bam files like that can be error-prone, but would be okay to do that, since you'd only need to do it once. You can also automate the process to generate the names at the command line. It is not clear if it is worth going through the trouble, and there may be simpler ways to achieve the same effect than this below:

```
# These are the MOCK numbers
MOCK=(SRR3191542 SRR3191543 SRR3194428 SRR3194429)
```

106.5. HOW DO I COMPUTE DIFFERENTIALLY EXPRESSED GENES?821

```
# These are the ZIKV numbers.  
ZIKV=(SRR3191544 SRR3191545 SRR3194430 SRR3194431)  
  
for SRR in ${MOCK[@]}; do  
    MOCK_BAMS+="bam/${SRR}.bam "  
done  
  
for SRR in ${ZIKV[@]}; do  
    ZIKV_BAMS+="bam/${SRR}.bam "  
done  
  
# The annotation file.  
GTF=~/refs/Homo_sapiens.GRCh38.96.chr.gtf  
  
# Count the features over each gene name.  
featureCounts -g gene_name -a $GTF -o counts.txt $MOCK_BAMS $ZIKV_BAMS  
  
# Simplify the counts.  
cat counts.txt | cut -f 1,7-14 > simple_counts.txt
```

106.5 How do I compute differentially expressed genes?

Just as in Alignment based RNA-Seq of control samples the steps would be:

```
# Obtain the deseq1 code  
wget -q -nc http://data.biostarhandbook.com/rnaseq/code/deseq1.r  
  
# Run the statistical model 4x4 matrix.  
cat simple_counts.txt | Rscript deseq1.r 4x4 > results_deseq1.txt
```

106.6 How do I visualize the differentially expressed genes?

A simplified method for visualizing differentially expressed genes is a heatmap. A particular variant of it, the clustered heatmap is so commonly used and so useful (yet non-trivial) to obtain that we've included a script that will generate one for you.

```
curl -O http://data.biostarhandbook.com/rnaseq/code/draw-heatmap.r
```

This script produces a PDF output on the standard output. When you ran your DE script it also created an additional file called a “normalized matrix”. You can generate your heatmap from this matrix with:

```
cat norm-matrix-deseq1.txt | Rscript draw-heatmap.r > zika-output.pdf
```

Note how this script also writes to standard out, though this time it writes a PDF file. What is in this PDF file? It is a hierarchically clustered heatmap image of your samples and differentially expressed genes:

The plot looks pretty messy at this point and demonstrates some of the problems with automatically generating results. To fix it, you can edit the file called `norm-matrix-deseq1.txt`, relabel it, and filter it in some way.

That being said, the plot clearly demonstrates that there are genes that are up/down regulated.

106.7 How do I make sense of this data?

This is where we go back to one of the first chapters in this book. The sections describing what words mean, how functions are assigned to genes and what gene enrichment analysis is.

The differential expression file that you get above and the normalized matrix can be entered into various tools to investigate the biology behind the measurements.

Here is what we find with the data that we ran above:

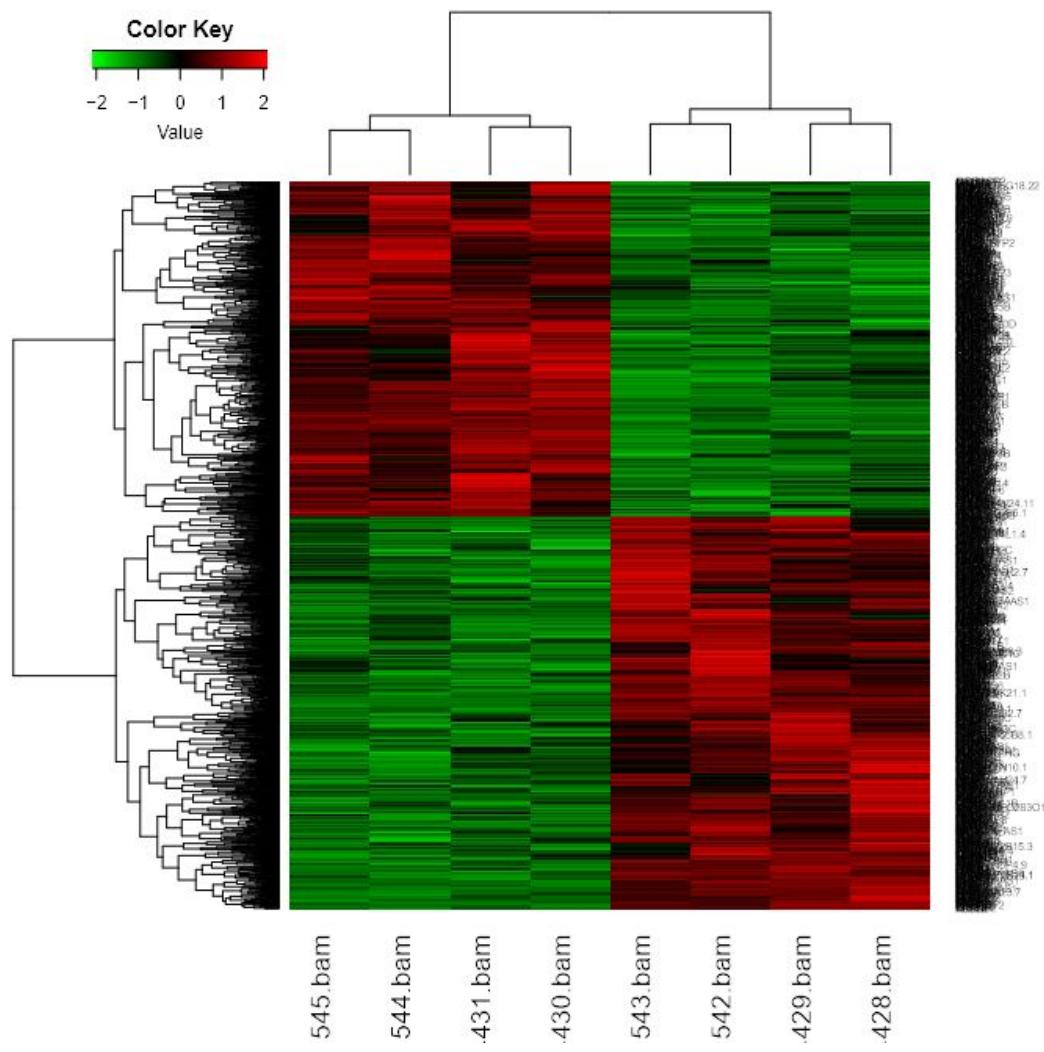


Figure 106.3: Zika Results

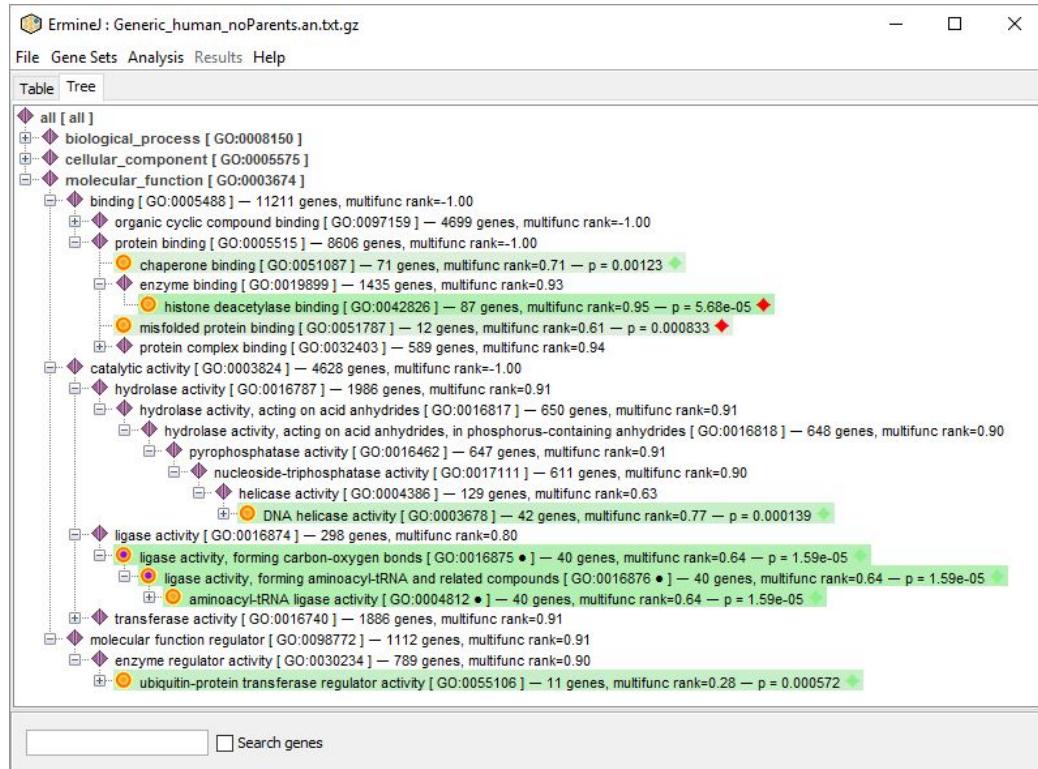


Figure 106.4: Zika Results

106.8 What is the result of the differential expression analysis?

The differential expression analysis script produces a table with changes in gene expression across the two conditions. The difference in gene expression is shown as fold change. The p-value and adjusted p-values generated by DESeq are also given.

id	baseMean	baseMeanA	baseMeanB	foldChange	log2FoldChange	pval
SEC24D	613.732089	129.834166	1097.630012	8.454092196	3.079649846	7.09E-10
SLC7A5	5451.691973	1661.110476	9242.273469	5.563912576	2.476099751	2.13E-10
XBP1	2481.903296	746.6520304	4217.154561	5.648085573	2.497761947	2.43E-10
HERPUD1	2328.813848	741.1048934	3916.522803	5.28470779	2.401823702	1.36E-09

- **id** - Gene or transcript name that the differential expression is com-

106.9. HOW DO THE RESULTS RELATE TO EXPECTED RESULTS?825

puted for

- `baseMean` - The average normalized value across all samples
- `baseMeanA` and `baseMeanB` - The average normalized gene expression for each condition
- `foldChange` - The ratio of values, computed as `baseMeanB/baseMeanA`.
- `log2FoldChange` - A log₂ transform of `foldChange`. When we apply a base 2 logarithm, the values become symmetrical around 0. A log₂ fold change of 1 and -1 indicate a doubling or halving of the expression level, respectively.
- `pval` - The p-value generated by DESeq.
- `padj` - The p-value corrected by multiple hypothesis testing.

106.9 How do the results relate to expected results?

```
cat results_deseq1.txt | awk '$8 <0.05 {print $0}' | wc -l  
4451
```

```
cat results_deseq1.txt | awk '$7 <0.05 {print $0}' | wc -l  
7131
```

We have 4451 differentially expressed genes with `padjusted <0.05`. If we consider `p-value <0.05`, then there are 7131 significant genes. You may use `p-value` if you already had selected your target gene before evaluating these results. In all other cases, the adjusted p-value should be used as this corrects for multiple testing errors.

If we had run `DESeq2`, we would have 9497 differentially expressed genes with `p-adjusted <0.05`.

From the `GSE78711_gene.exp.all.txt` file that you've already downloaded (see zika-data chapter) there are 6864 differentially expressed genes with `p-value < 0.05`. Adjusted p-values are not given.

```
cat GSE78711_gene.exp.all.txt | grep "yes" | wc -l  
6864
```

Now, let's take a look at some individual genes and see how the fold change varies between the methods.

The fold change for geneIFIT2 from the downloaded results is 5.23. Deseq1 reports 4.45 and DESeq2 reports 3.4 fold change for the same gene. But for geneCEBPB downloaded results and both DESeq1 and DESeq2 results are in agreement. (3.08,3.07,2.99)

Please note that since different methods use different statistics and have different sensitivities to false positives, there will always be some variation in the results.

Keep on investigating the results, place the genes in a gene set ontology tool and see how well your findings reproduce those that were published.

Chapter 107

Classification based RNA-Seq of Zika data

107.1 What are the steps of a classification based RNA-Seq?

The three stages of RNA-Seq will be performed with:

1. Classification with `kallisto` to produce a count file.
2. Differential expression: `DESeq`

Kallisto is a tool for quantifying transcript abundances. It performs a *pseudoalignment* of reads against a transcriptome. In a pseudoalignment, the software attempts to identify the best match to each read using techniques other than alignments - but then from those matches, it generates an alignment-like arrangement.

Typing `kallisto` on the command line will show all the available commands that can be used. The `index` command builds the kallisto index and the `quant` command runs the transcript abundance quantification algorithm.

107.2 How do I build a Kallisto index?

Kallisto uses a transcriptome as a reference. The first step to run Kallisto is to get the transcriptome and build a Kallisto specific index. Since `kallisto` does not provide any pre-built indices, we'll have to build our own.

We have discussed the step to downloading the reference files at the beginning of the chapter Alignment based RNA-Seq of Zika data. In a nutshell, you would need to run (only once) the script `zika-references.sh`:

```
# Get the script that downloads reference files.
wget -nc http://data.biostarhandbook.com/redo/zika/zika-references.sh

# Run the script that downloads the reference files.
bash zika-references.sh
```

The shortcuts to reference and index file names are then:

```
REF=~/refs/Homo_sapiens.GRCh38.cdna.all.fa
IDX=~/refs/Homo_sapiens.GRCh38.cdna.all.idx
```

```
# Build the kallisto index
kallisto index -i $IDX $REF
```

This too only needs to be run once per reference file, the index may be reused for other studies for the same transcriptome. On our system (iMac) the indexing takes about 5 minutes.

107.3 How do I quantify transcripts with Kallisto?

Kallisto assigns reads to transcripts and calculates the abundance in one step. The algorithm can be invoked using `quant` command.

We are using the zika-infected samples that we have already downloaded in the ‘data’ directory.

Paired-end data can be quantified like this:

```
# Create an output directory.
mkdir -p results
```

```

# The run id.
SRR=SRR3191542

# Shortcut to read names.
R1=reads/${SRR}_1.fastq
R2=reads/${SRR}_2.fastq

# Run the kallisto quant command for paired end mode.
kallisto quant -i $IDX -o results/$SAMPLE $R1 $R2

```

The above command creates a directory specified with the `-o` flag, in this case `SRR3191542`, and places the abundance files there.

To quantify single end data, the `--single` option needs to be specified. Fragment length (`--l`) and standard deviation (`--s`) of the fragment length are also mandatory parameters. According to kallisto manual “Illumina libraries produce fragment lengths ranging from 180–200 bp but it’s best to determine this from a library quantification with an instrument such as an Agilent Bio-analyzer.”

The Zika dataset was reanalyzed by the authors of `kallisto` in the paper titled *Zika infection of neural progenitor cells perturbs transcription in neurodevelopmental pathways*¹ where they state that

For single-end read quantification, we used default parameters (kmer size = 31, fragment length = 187 and sd = 70). For each of the eight samples, kallisto quantified transcript abundances and performed 100 bootstraps.

Thus to follow the recommendation most precisely the command should be:

```

# Run the kallisto quant command in single-end mode.
kallisto quant -i $IDX -o results/$SAMPLE -l 187 -s 70 $R1 $R2

```

¹<http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0175744>

107.4 What are the outputs of Kallisto?

`kallisto` produced output files will be placed in the directory specified via the `-o` flag. Transcript abundance estimate results are in a tab delimited file named `abundance.tsv`. `abundance.tsv` file looks like this:

target_id	length	eff_length	est_counts	tpm
ENST00000628275.2	453	275.603	2.76617	2.97517
ENST00000628424.1	322	148.792	1.94712	3.87909
ENST00000630311.1	270	102.019	0.675219	1.96191
ENST00000628947.1	535	357.24	6.5	5.39349
ENST00000628339.2	2953	2775.11	1.87061	0.199811

Effective length (the third column) scales the transcript length by the fragment length distribution. It takes into account the part of the transcript where fragments would overlap fully in any location. It is usually calculated as `transcript length - mean(fragment length) + 1`. Abundances are reported in “estimated counts” (`est_counts`) and in Transcripts Per Million (TPM).

107.5 How do I run kallisto for all the samples?

The command below can be used to automate `kallisto` runs for all samples. First, recall the accession number grouping as described in Understand the Zika data

```
# SRR numbers for single-end reads.
SINGLE=runinfo.single.csv

# SRR numbers for paired-end reads.
PAIRED=runinfo.paired.csv

# Run kallisto on single end reads.
cat $SINGLE | parallel kallisto quant -i $IDX -o results/{} --single -l 187 -s 70 reads/{}_1.fq.gz

Run kallisto on single end reads.
cat $PAIRED | parallel kallisto quant -i $IDX -o results/{} -l 187 -s 70 reads/{}_1.fq.gz
```

107.6 How do I quantify the results?

By the end of the run above, in the `results` folder, we end up with an output directory named after each sample. We need to rename the files by the samples so that we can handle them better. Let the tedium begin. We need to rename and combine many data files into a single one.

Note: You may ask yourself, why are these steps so clunky? Why is the process so inefficient? Why can't the tool produce an output that does not need line after line of massaging into the proper form? All good questions, all come back to the same answer. There is no reward for making non-clunky tools.

Ok, let's get to it. We want to end up with a single file that contains all estimated counts.

```
# Extract the counts from each file.
cat $SINGLE $PAIRED | parallel "cut -f 1,4 results/{}/abundance.tsv > results/{}.txt"

# These are the control samples.
#MOCK samples are SRR3191542 SRR3191543 SRR3194428 SRR3194429
paste results/SRR3191542.txt results/SRR3191543.txt results/SRR3194428.txt results/SRR3194429.txt

# The infected ZIKV samples.
# ZIKV SRR3191544 SRR3191545 SRR3194430 SRR3194431
paste results/SRR3191544.txt results/SRR3191545.txt results/SRR3194430.txt results/SRR3194431.txt

# Combine the mock and zikv.
# The header needs to be edited by hand to reflect the true sample names.
paste mock.txt zikv.txt | cut -f 1-5,7-11 > counts.txt
```

Counts file `counts.txt` will be as shown below. This count file can be used as input to downstream DEseq differential expression analysis.

target_id	est_counts						
ENST00000472225.6	6.11331	1.28737	10.0567	2.23768	2.1543	0	
ENST00000496578.3	3.12574	3.97029	8.18468	0	1.6547	1.49019	
ENST00000624726.1	6.73189	3.0668	1.16443	2.39545	4.31444	1.26448	

You may change the header to include the sample names.

107.7 How do I find differentially expressed transcripts?

Whereas the previous analyses were gene-based (though we could have used transcripts there as well), kallisto works on transcript data. You could, of course, emulate the “gene” level analysis by creating “fake” transcripts that are made up of concatenated exons (but there is no reason to do so).

Let's get our differential expression script and run it:

```
curl -O http://data.biostarhandbook.com/rnaseq/code/deseq1.r
```

But, surprise! The script won't run. We have a bit of a problem here. At the transcript level, your data may have many transcripts that are not detected at all. These produce zeros for all samples. In many cases, we prefer not to include these in the statistical analysis. For example if the file has too many rows, the input may fail. There may also be an error associated with the classification process, fewer measurements are in general far less reliable. So there is a reason to believe that we should apply a cutoff.

For example, we might choose only to keep rows where the sum of all counts is larger than 25. Or we may choose to restrict the list to the 20 thousand most expressed transcripts. It is never clear (at least not to us) how to find this threshold optimally. If you wanted to filter the file to keep only transcripts that at least 25 reads covered overall we could do it with awk (or many other methods):

```
cat counts.txt | awk '$2+$3+$4+$5+$6+$7+$8+$9 > 25 { print $0 }' > over25.txt
```

The code above drops the number of rows from 178,126 to 26,759.

We now have another problem here as well. The `kallisto` estimator uses effective lengths and produces floating point numbers rather than integers for counts. Some tools (like DESeq2) will balk at that, and you need to convert them to integers instead. You can do this inside R or with another awk script by applying the formatting operator `%3.0f` on each element like so:

We also admit the above construct looks pretty absurd – but hey, sometimes awk can save you a lot of time and effort. Just be sure to document the rationale for each step. Finally, your chest might swell with pride. Look at this pretty data!

107.7. HOW DO I FIND DIFFERENTIALLY EXPRESSED TRANSCRIPTS?833

ENST00000419783.3	47	52	40	38	59	52	44	47
ENST00000651274.1	1	7	2	6	12	2	0	1
ENST00000361390.2	1767	1706	1199	1247	1145	1226	815	919
ENST00000361453.3	2097	1945	1770	1738	1175	1324	1013	1179
ENST00000361624.2	10594	10350	9275	9373	8228	9083	7204	7917
ENST00000361739.1	4433	4292	3184	3023	3121	3212	2244	2388
ENST00000361851.1	270	239	94	70	171	215	59	58
ENST00000361899.2	2582	2181	2111	1863	1454	1701	1173	1375

Let's run our differential expression estimator:

```
cat valid.txt | Rscript deseq1.r 4x4 > zika-results.txt
```

The file `zika-results.txt` is your answer as to what changes occur in the gene expression of Zika infected brain samples.

Part XXIV

CHIP-SEQ Analysis

Chapter 108

Introduction to ChIP-Seq analysis

108.1 What is ChIP-Seq?

ChIP-Seq stands for chromatin immunoprecipitation followed by sequencing. In a nutshell, the process consists of a laboratory protocol (abbreviated as ChIP) by the end of which the full DNA content of a cell is reduced to a much smaller subset of it. This subset of DNA is then sequenced (abbreviated as Seq) and is mapped against a known reference genome. Since you are now obtaining data from a potentially variable subset of the entire genome, it poses unique challenges to data interpretation, hence a whole subfield of data analytics has grown out around it.

Whereas the name appears to imply that the analysis methods “require” that the data was obtained via chromatin immunoprecipitation (ChIP) this is not necessarily so. Many other laboratory protocols where the net effect is that only a subset of the total DNA is selected for can be processed with ChIP-Seq methods.

108.2 What are the challenges of ChIP-Seq?

Perhaps one of the most significant problem of this entire subfield is the insufficiently well-defined terminology. See for example the section “**What the heck is a peak**” later on. Also, more so than in other analysis methods, there a “subjectivity creep.” Seemingly simple decisions are made all the time - that in the end seem to accumulate and slowly grow out of control as the analysis proceeds.

Examples of these insufficiently defined terms are for instance the concepts of “upstream” and “downstream” regions which typically refer to the genomic intervals that lie ahead of and after a particular genomic coordinate.

When referring to these ranges, you will need to associate a finite length to them: how far out do you still call a region to be “upstream.” The choice of this distance may, in turn, significantly alter the outcome of the analysis. You will likely obtain different results when the upstream region is defined to be 500bp, 1000bp or 5000 kb long.

Further complicating the picture is that different genomes have varying complexities; in some genomes, the genes are densely packed, other vast tracts are seemingly nonfunctional. Hence there is no “objective” measure by which a “reliable” length for an upstream or downstream region could be determined from. The size of the “upstream” region is little more than a handwaving approximation.

108.3 What the heck is a peak?

You will see how ChIP-Seq analysis is all about peaks - a term, that in our opinion is adopted widely yet lacks any exact definition. Somewhat ironically speaking, perhaps the best description of a peak is the following:

A “peak” is what a “peak-finder” finds.

What makes the problem complicated is that we all think that we know what a “peak” is (or should be). A peak is a “shape” that rises sharply to a maximum then quickly drops back to zero. In that ideal case then the

peak consists of the **coordinate** of the maxima on the genome (an integer number) and a **peak height**.

As it happens, in reality, most of your signals may not follow such well-behaved patterns - the peaks may be a lot broader,

Now if the peak is wide you may need to compute the **peak width** that will indicate the range where the signal is present. Then once you have a broader peak representing it with its center may be counterproductive as it is not clear that the summit will be symmetrical around it. Instead, you'll have to keep track of the **right** and **left** borders of it.

Just having a signal, of course, is insufficient, we may want to ensure that the signal rises **above the background** of the noise levels. Also, most studies are interested in how cells respond to stimuli that in turn will alter the signal. So the changes may occur by the peak rising in a **different location** or via **changed peak height** often denoted with the term **differential peak expression**.

You can see how a seemingly innocuous word: “peak” may require managing more information than you’d expect.

108.4 How are RNA-Seq studies different from ChIP-Seq studies?

Whereas both approaches appear to operate under similar constraints and count reads over intervals, there are several significant differences:

1. The DNA fragments coming from a ChIP-Seq study are much shorter than a transcript studied in RNA-Seq analysis. Instead of counting measurements distributed over transcripts a single read may cover most of the fragment (or may even be longer than the original DNA fragment)
2. The DNA fragments of a ChIP-Seq experiment are less localized than transcripts. The cell machinery accurately locates and transcribes DNA always starting from the same start locations. The ChIP-Seq protocol is a whole lot more inaccurate. You need to apply a peak-calling process to establish the origin of the fragments from incorrect data.

3. ChIP-Seq data produces a more significant number of false positives. In a typical RNA-Seq experiment, only transcripts are isolated identically, and we can be fairly sure that an observed sequence did exist as RNA. In contrast, ChIP-Seq protocols strongly depend on the properties of the reagents, selectivity, and specificity of the binding process, and these properties will vary across different proteins and across the genome.

As a matter of fact, for ChIP seq results we need to compare to control (baseline) to verify that the protocol worked **at all**. Whereas in RNA-Seq we can immediately see the success of it by having reads cover transcript, in ChIP-Seq study no such certainty exists. Also when comparing peaks across conditions each peak has to first pass the comparison to a background, and then only those peaks that pass the first stage are compared to one another in a second stage. Since each statistical test is fraught with uncertainties, differential peak expression studies will often exhibit compounding errors.

108.5 What will the data look like?

For the record let's enumerate the possible observations for a given genomic location.

1. No signal of any kind.
2. A signal that shows no peak structure and appears to be noise.
3. A "peak like" signal that is reported to be under a background level.
4. A "peak like" signal that is reported to be over a background level.
5. A "peak like" signal that exhibits differential expression across samples.

Many (most) peak finders will start reporting data only from level 4 on - leaving you with many head-scratching moments of the form: "Why is this perfectly looking peak not reported?". It all makes sense once you recognize that "peak finding" as the field currently defines it, combines finding the peaks and comparing them to a background. In addition, the statistical methods employed to determine these background levels have very high levels of uncertainty.

108.6 What does a “peak” represent?

Most ChIP-Seq guides start with explaining how ChIP-seq works, and how it allows us to identify where proteins were bound to DNA. We believe that is far more important to understand what your data stands for in a more straightforward way first, then extend that to the laboratory process.

Imagine that we have DNA from a cell that looks like this:

```
AGTGATACCGGTCTAACGCCTCGCCTTCCACAGGGAAACTAGGTGGCC
```

Now let's say some locations are “covered,” say “protected” from an interaction. Let's visualize those locations via the = symbols.

```
===== =====
AGTGATACCGGTCTAACGCCTCGCCTTCCACAGGGAAACTAGGTGGCC
===== =====
```

Suppose that can wash away all unprotected regions. If we did that then sequenced the remaining DNA we'd be left with:

```
1 GGTCTAAGC
1 GGGGAAACTA
```

The first number indicates how many of those sequences were found. So far so good. Now let's take another cell from the same population. Suppose that the DNA for this new cell is also now covered, but in only one location:

```
=====
AGTGATACCGGTCTAACGCCTCGCCTTCCACAGGGAAACTAGGTGGCC
=====
```

After washing away the uncovered regions we add the outcome to the prior results, and we have:

```
2 GGTCTAAGC
1 GGGGAAACTA
```

Finally, let's assume we have DNA from a third cell, this one has a protein bound in the second location.

```
=====
AGTGATACCGGTCTAACGCCTCGCCTTCCACAGGGAAACTAGGTGGCC
```

=====

Thus our cumulative data that we obtain after washing away unprotected regions from three cells would be.

2 GGTCTAAGC
2 GGGGAAACTA

In a ChIP-Seq experiment, instead of sequencing the entire genomes from three different cells we only sequence relatively short pieces of DNA (in reality the sequences are longer than these example sequences above but still radically shorter than say a transcript). The alignments that we generate will pass through a “peak finding” program that will attempt to reconstruct which locations had sequences mapping to and how many from each of the small sequences we have observed. If everything goes well, the ideal outcome of our analysis for the data above will produce:

- A peak of height 2 that starts at coordinate 10 and ends at coordinate 19
- A peak of height 2 that starts at coordinate 34 and ends at coordinate 43

Do note however that the DNA from one cell had two regions covered, whereas the two other cells had only one zone each. Our peaks above do not indicate this. They represent a superposition of configurations across all cells. New configurations leading to the same picture may be possible; there is no way to know that the covered regions appear all together or separately in the original DNA molecule.

Essential to remember:

The peaks are cumulative, superimposed measures of the observed configurations across millions of different cells.

108.7 Do peaks represent binding?

Now, up to this point, we avoided the word “binding” altogether - yet the purpose of ChIP-Seq is to measure exactly that. A protein bound to a DNA molecule can “protect” it from being washed away - then in a second

“pulldown” stage, only the DNA bound to a particular protein is kept. Hence under the ideal condition, the result of a ChIP-Seq experiment contains DNA that was protected by one specific type of protein.

The reality is a bit different. Remember your ChIP-Seq data does not detect binding. It recognizes the abundance of DNA fragments in a pool of DNA molecules. Each measure indicated that a given DNA fragment was present when all the DNA was combined. As it turns out in practice, there may be several other reasons for DNA to be present in your library. Control samples are almost always required (see next question)

108.8 How do we determine peaks from sequencing data?

There is a tendency to gloss over this step, relying on the “peak finder” then using the results that it produces. Once you think about the problem, you’ll see how peak finding process is more counterintuitive than you’d imagine.

Remember how sequencing instrument produces a linear sub-measurement of DNA fragments? This measured “read” has a length associated with it, but this length has nothing to do with the original fragment length.

If we mark the first fragment as = then we can see that depending on the DNA fragment lengths and our measurement (read) lengths the measured reads could fall into three different orientations.

```
===== DNA FRAGMENT =====

# Reads much shorter than fragment.
-----> <-----

# Reads as long as the fragment.
----->
<-----

# Reads longer than the fragment
----->
<-----
```

Three different situations occur.

1. The read length is much smaller than the fragment.
2. The read length is about the same size as the DNA fragment
3. The read length is larger than the DNA fragment and runs past the fragment ends.

As a result, the analysis methods fall into two categories: 1 and 3 as the second case occurs very rarely. Often these are called “narrow” peak and “broad” peak methods.

Be advised that a different mindset and approach is required for each situation - methods that fit broad peaks are likely to work sub-optimally in for narrow ones and vice versa.

108.9 What does a ChIP-Seq data measure?

If you refer to the sketch above only one thing is common across all data sets. The start of the reads that align on the forward strand indicates the left border of the DNA fragment. The start of the reads that align on the reverse strand all indicates the right border of the DNA fragment.

Hence we can state the fundamental “law” of ChIP-Seq data:

ChIP-Seq data measure the outer borders, the “edges” of the DNA fragments.

Of course, this is not all that great of news. It makes the picture a whole lot more complicated. Let me show you what this means. Imagine that you see coverage that looks like this image below. it seems so clear cut, there is one binding event, in the middle, indicated by the highest point of the peak.

Sadly this is misleading. Where the fragments appear to overlap to most may be a superposition of many different, alternative positions. In this particular case if you were to break down the data by strand then plot only the coverages for the 5' ends for each strand separately you would see the following coverages:

It is the superposition of these two lower tracks that gives the top coverage. But when plotted this way you can clearly see at least two distinct peaks.

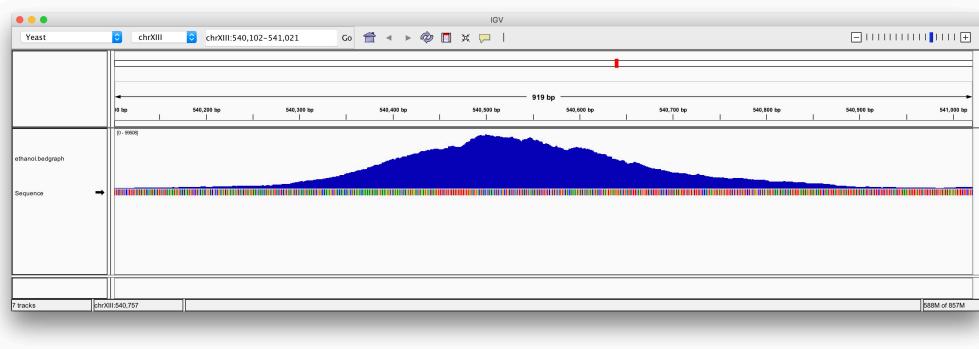


Figure 108.1

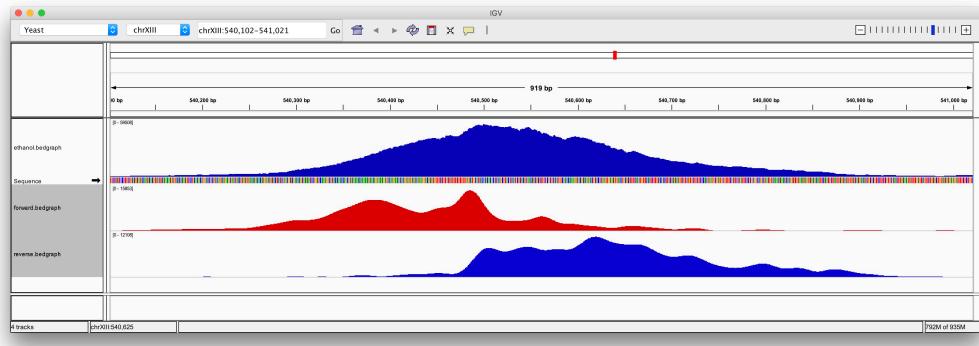


Figure 108.2

Remember these now track what the instrument measures: the ends of the DNA fragments. It appears that there are a few distinct fragments (position) on the DNA. Let me mark these positions.

If we trust that our ChIP-Seq method measures bound locations, then these are the locations occupied by our protein. But note how none of them are bound in the location that the peak maxima indicate. If location accuracy is critical, the peaks must be found by pairing up the peaks called separately from the forward and reverse strands to attempt to reconstitute the original DNA fragments, rather than the center of each peak.

If peak location is not all that important, you may be able to take the midpoint of the original peak.

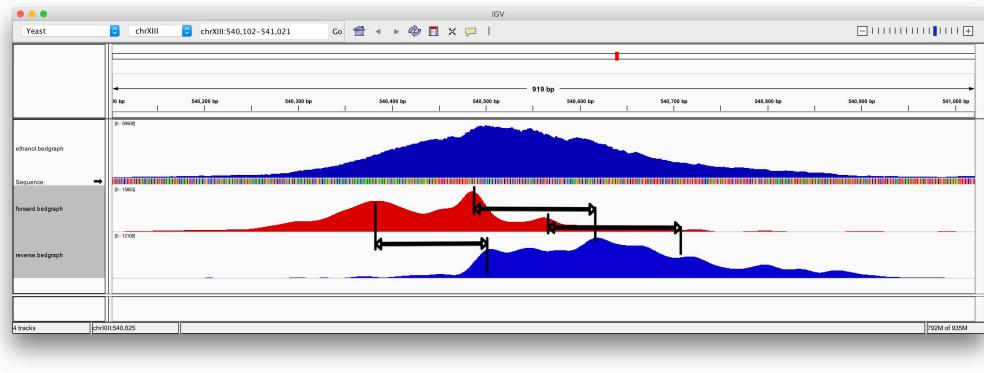


Figure 108.3

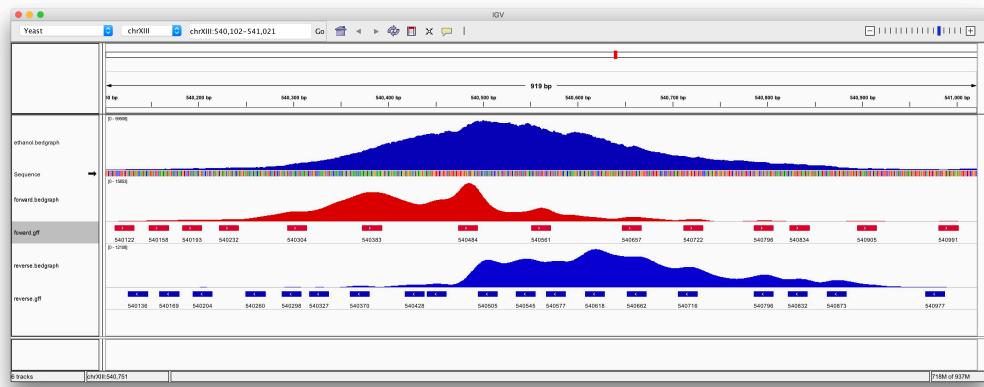


Figure 108.4

Much algorithmic effort is spent to create software that automatically builds the “consensus” location out of data. Some work better than others - none of the tools can handle all the situations, and much subjectivity remains. The type of the data and its properties, whether or not a superposition of locations are present will all substantially alter the performance of the method you choose.

108.10 What type of ChIP-Seq studies are performed?

Speaking by the extremes, and your study will likely fall somewhere in between but closer to one of these extremes, ChIP-Seq studies fall into two major categories:

1. The **Theory of Everything**: these are extensive scale studies, usually studying millions of locations, then often make sweeping, large-scale generalizations on where certain positions are about others. The principle “gain-some lose-some drives the methods”. The data analysis typically processes all peaks with the same parameters producing very high rates of false positives and false negatives. The goal of these studies is to beat random chance - which is a very low bar to pass.
2. The **Theory of One Peculiar Thing**: these are focused studies that use ChIP-Seq surveys to attempt to explain one particular phenomenon. The data often gets reduced to just a few locations where the very high accuracy of the methods is required.

The needs of the two approaches are radically different. As a rule, if you need to accurately determine the “peak” at just a few locations the best tool that you can use is YOUR EYE! We make use of “peak callers” primarily because we can’t afford to investigate every single location.

108.11 Does the data need to be compared to control?

Yes. The ChIP-Seq protocols have turned out to be affected by many more systematic errors than initially thought. Hence control samples are needed to identify these errors.

Typically two kinds of controls samples are used: **IgG** control and **input** control. Each control can be thought of as an incomplete ChIP-Seq protocol, where the process intentionally skips a step of the process. These controls attempt to identify regions that are enriched by other processes beyond the protein being bound to the DNA.

108.12. SHOULD MY CHIP-SEQ ALIGNER BE ABLE TO DETECT INDELS?847

- The **IgG** control is DNA resulting from a “mock” ChIP with Immunoglobulin G (IgG) antibody, which binds to the non-nuclear antigen.
- **Input** control is DNA purified from cells that are cross-linked, fragmented, but without adding any antibody for enrichment.

One problem with IgG control is that if too little DNA is recovered after immunoprecipitation(IP), sequencing library will be of low complexity (less diverse) and binding sites identified using this control could be biased. I

Input DNA control is thought to be ideal in most of the cases. It represents all the chromatin that was available for IP. More information on choosing controls can be seen in:

- Biostars post: What Control For Chip-Seq: Input, IgG Or Untagged Strain?¹ for more details.
- Nature, 2015: ChIP-Seq: technical considerations for obtaining high-quality data²

108.12 Should my ChIP-Seq aligner be able to detect INDELS?

Absolutely YES!

One of the most misleading advice - one that is reiterated even in peer-reviewed publications like Practical Guidelines for the Comprehensive Analysis of ChIP-Seq Data³ PLoS Comp. Biology 2014, stems from a fundamental misunderstanding of scientists when it comes to defining how “alignment” is different from “mapping.”

We described the duality before in the alignment section stating that “mapping” refers to finding the genomic location of where a read originates from, whereas “alignment” refers to finding the place and arrangement about the genome. Since ChIP-Seq methods need to identify genomic locations many wrongly assume that the alignment methods that cannot handle insertions

¹<https://www.biostars.org/p/15817/>

²<http://www.nature.com/ni/journal/v12/n10/abs/ni.2117.html>

³<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003326>

and deletion are still acceptable since we don't care about the exact alignment anyway.

It is time to put this misconception out of its misery as it has already caused and continues to produce substantial wasted efforts. So let's make it abundantly clear:

An aligner that cannot handle insertions and deletions will not be able to find the correct coordinates when a measurement contains such differences. The "mapping" produced by such aligners over the reads in question will be severely and systematically affected.

As a rule, aligners that cannot detect insertions and deletions should **NEVER** be used under **ANY** circumstances - unless no other aligner can be utilized (very rare cases where other aligners would not work at all).

108.13 What are the processing steps for ChIP-Seq data?

The general processing steps are as follows:

1. Visualize and, in necessary, correct the quality of the sequencing data.
2. Align sequencing reads to a reference genome. The most popular read aligners are `bwa` and `bowtie`.
3. Call peaks from the alignment bam files.
4. Visualize the resulting peak files and signal files.
5. Find the biological interpretation of the positions where the peaks are observed.

Chapter 109

Aligning ChIP-Seq data

As we explained before chromatin immunoprecipitation (ChIP) followed by high-throughput DNA sequencing (ChIP-Seq) is a technique that is used to identify locations in the genome, for example, the footprints of genome-wide transcription factor binding sites and histone modification enriched regions.

We will demonstrate the steps used to process ChIP-Seq data through a re-analysis of a published scientific work.

109.1 Which paper is going to be re-analyzed?

To illustrate all the analysis steps, we are going to use data from the following publication:

- Genome-Wide Mapping of Binding Sites Reveals Multiple Biological Functions of the Transcription Factor Cst6p in *Saccharomyces cerevisiae*.¹ published in MBio in 2016

The data corresponding to this paper can be found in the SRA Bioproject:

- <https://www.ncbi.nlm.nih.gov/bioproject/PRJNA306490>
- The script that will reproduce this paper is located in <http://data.biostarhandbook.com/chipseq/>

¹<https://www.ncbi.nlm.nih.gov/pubmed/27143390>



Figure 109.1

This section provides an annotated guide to the commands in the `chip-exo-redo.sh` script. This script makes use of GNU Parallel to execute multiple commands at a time (up to the number of CPU on the system).

109.2 How do I obtain the data for project PRJNA306490?

Make a few directoried to store data in:

```
mkdir -p data
mkdir -p refs
mkdir -p bam
```

Then perform the search for the run information:

```
esearch -db sra -query PRJNA306490 | efetech -format runinfo > runinfo.csv
```

The file contains information for four SRR runs

```
runinfo.csv | cut -d , -f 1,2,3 | head
```

will produce

```
Run,ReleaseDate,LoadDate
SRR3033157,2016-04-08,2015-12-18
SRR3033156,2016-04-08,2015-12-18
SRR3033155,2016-04-08,2015-12-18
SRR3033154,2016-04-08,2015-12-18
```

You could run the following command on each SRR run:

```
fastq-dump -O data --split-files -F SRR3033154
```

Or you could automate that into a single line, by cutting the first column for SRR ids then piping the result into `fastq-dump`

```
# Isolate just the run ids for the run.
cat runinfo.csv | cut -f 1 -d , | grep SRR > runids.txt

# Download the fastq files.
cat runids.txt | parallel --eta --verbose "fastq-dump -O data --split-files -F {}"
```

The data download may chug along for a bit; the files are a few gigabytes each. You could use the `-X` flag to limit the dataset sizes.

109.3 How do I find out more information each sample?

While the `runinfo.csv` contains quite a bit of information, maddeningly it lacks what arguably would be the most important detail: which SRA number corresponds to the samples mentioned in the paper. We need to run the Entrez Direct tool called `esummary` to connect run ids to sample names:

```
esearch -db sra -query PRJNA306490 | esummary > summary.xml
```

The XML file returned by this query has a nested structure with a wealth of information that can be difficult to extract. Sometime the easiest is to read it by eye and manually match the information. This might work here

since there are only four samples. But there are ways to automate this data extraction.

The tool called **xtract** that is also part of EDirect: **xtract**² is meant to facilitate turning XML files into tabular files. The **xtract** tool operates by parsing an XML file, matching and extracting and printing elements from this file.

Now XML as an industry standard with methods to do just this, there is a so-called **XPath** query language and the **XSL** styling language that could be used to turn XML documents into any other format. **xtract** has nothing to do with either of these - yet it is used to achieve the same. It is, in a way a re-imagining of what **XPath** and **XSL** do - but this time filtered through a bioinformaticians mind.

For example, the following command generates a new row on each element called named **DocumentSummary** then extracts the **acc** attribute of the **Run** element and prints that with the title. Investigate the XML file to see where this information is present.

```
cat summary.xml | xtract -pattern DocumentSummary -element Run@acc Title
```

will transform the **summary.xml** file into :

```
SRR3033157 GSM1975226: Cst6p_ethanol_rep2; Saccharomyces cerevisiae; ChIP-Seq  
SRR3033156 GSM1975225: Cst6p_ethanol_rep1; Saccharomyces cerevisiae; ChIP-Seq  
SRR3033155 GSM1975224: Cst6p_glucose_rep2; Saccharomyces cerevisiae; ChIP-Seq  
SRR3033154 GSM1975223: Cst6p_glucose_rep1; Saccharomyces cerevisiae; ChIP-Seq
```

From this output we can see here that each sample has two replicates:

- SRR3033157 and SRR3033156 correspond to Cst6p_ethanol
- SRR3033155 and SRR3033154 correspond to Cst6p_glucose

109.4 What other information do I need to start a ChIP-Seq analysis?

You will now need a genome sequence and an annotation file that lists the genes.

²<https://dataguide.nlm.nih.gov/edirect/xtract.html>

Sometimes (often?) it is surprisingly complicated to get the latest information even from websites dedicated to the distribution of data. As you will note for yourself, the data always seems to be either in the wrong format or combined in ways that are inappropriate to your current needs.

The data sources claim that it is a matter of “just downloading it” yet soon you’ll be knee-deep in elbow-grease as you wrestle in the mud with uncooperative file formats. For example, SGD claims that the latest genome releases can be downloaded from http://downloads.yeastgenome.org/sequence/S288C_reference/genome_releases/. As you happily and cheerfully download that data, you realize that the chromosomal naming coordinates are different across the sequence and the annotations. For example, the FASTA file will have the sequence names listed as:

```
> ref|NC_001133| [org=Saccharomyces cerevisiae] [strain=S288C] [moltype=genomic] [chrom
```

whereas the features the GFF files in the *very same* downloaded data will list the chromosomal coordinates as:

```
> chrI
```

Suffice to say no bioinformatics tool will be able to match up these two. To use this data you would need postprocess the file to rewrite `chromosome=I` to be `chrI`. A data janitor job if there ever was one.

109.5 How do I get the “standard” yeast genome data and annotation?

The paper that we are reanalyzing makes use of the most up-to-date UCSC yeast genome release code named `sacCer3` from the UCSC download site³ some might balk at knowing that this release dates back to 2011. Quite a bit of knowledge could have been accumulated since then. One advantage is though that many tools, like IGV, already know of this genomic build and no custom genome building is necessary.

Let’s get the genome:

```
# Reference genome.  
REF=refs/saccer3.fa
```

³<http://hgdownload.cse.ucsc.edu/goldenPath/sacCer3/>

Download the chromosomes and build a genome. The sequences are stored by chromosome:

```
URL=http://hgdownload.cse.ucsc.edu/goldenPath/sacCer3/bigZips/chromFa.tar.gz
curl $URL | tar zxv

# Get the chromosome sizes. Will be used later.
curl http://hgdownload.cse.ucsc.edu/goldenPath/sacCer3/bigZips/sacCer3.chrom.sizes

# Move the files
mv *.fa refs

# Create genome.
cat refs/chr*.fa > $REF
```

Remember to index the reference.

```
bwa index $REF
samtools faidx $REF
```

We now have the sacer3 genome on our local filesystem.

109.6 How do I align the ChIP-Seq data?

We assume that by this time your `fastq-dump` programs above have finished. The alignment commands will all be of the form:

```
bwa mem $REF data/SRR3033154_1.fastq | samtools sort > SRR3033154.bam
```

Note how this data is aligned in single end mode and only file 1 is used. We too were surprised to learn this from the authors, their rationale is that they are using a ChIP-Exo method that sequences only a short segment of the fragment. We don't quite believe this rationale to be correct - suffice to say only half of the data was used and half of the data was tossed out.

We can automate the alignments as just one single line.

```
cat runids.txt | parallel --eta --verbose "bwa mem -t 4 $REF data/{}_1.fastq | samtools sort > {}_aligned.bam"
```

Once the step above completes the computationally most demanding process of ChIP-Seq has now been completed - but the analysis itself is only starting in its earnest.

109.7 Do I need to trim data to borders?

As we mentioned this before ChIP-Seq data in general and ChIP-Exo in particular measures the left-right borders of the binding sites rather than the center of them. Even though tools should be able to reconstruct these borders algorithmically from the entire alignments, sometimes (often) it is best to not even “show” this spurious data to the algorithm, and to keep only the essential sections of the data, in this case, the borders only.

That is what the authors chose here, trimming 70 bp from the ends of each alignment using the bamUtil tool:

```
# Trim each bam file.
cat runids.txt | parallel --eta --verbose "bam trimBam bam/{}.bam bam/temp-{}.bam -R 70 -"

# We also need to re-sort the alignments.
cat runids.txt | parallel --eta --verbose "samtools sort -@ 8 bam/temp-{}.bam > bam/trimm

# Get rid of temporary BAM files.
rm -f bam/temp*

# Reindex trimmed bam files.
cat runids.txt | parallel --eta --verbose "samtools index bam/trimmed-{}.bam"
```

The same process may work without this step.

109.8 How do I visualize the alignments?

ChIP-Seq data is all about coverages and locations. As such the details of alignments are usually less important. Visualizing BAM files directly puts too much strain on visualizers as often many samples need to be viewed simultaneously.

A bed graph⁴ file is a variant of a BED file that is meant to display coverages over intervals. It is a simple format consisting of a tab separated format of this form:

```
chrom start end value
```

⁴<https://genome.ucsc.edu/goldenpath/help/bedgraph.html>

Where `value` is the coverage over the interval between `start` and `end`. To create a coverage bedgraph file, we can use `bedtools`:

```
# Create an genome file that bedtools will require.
samtools faidx $REF
```

```
# Create a bedgraph file out of the BAM file.
```

```
bedtools genomecov -ibam bam/SRR3033154.bam -g $REF.fai -bg > bam/SRR3033154.bed
```

The bed graph⁵ file are smaller than the BAM file and may be visualized and processed with many other tools

Below is a figure of visualizing bedgraph and BAM files simultaneously. The upper track is bedgraph, the lower two panels are the coverage and alignment sections of a BAM file.



Figure 109.2

As it happens the bedgraph is not quite efficient either, IGV has to load the entire file into memory to visualize it. Yet another file format had to be invented (and we'll spare your the gory details) one that is called bigwig⁶. A bigwig is a binary format that contains the same information as a bedgraph but allows for efficient access, hence allows software tools to jump to various location in the file more quickly. This format, by the way, was invented by Jim Kent, the “benevolent dictator” of the UCSC genome browser.

⁵<https://genome.ucsc.edu/goldenpath/help/bedgraph.html>

⁶<https://genome.ucsc.edu/goldenpath/help/bigWig.html>

109.9. ARE THERE OTHER WAYS TO GENERATE BIGWIG FILES?857

The format itself is not documented or described properly - a single program code, also written by Jim Kent was the sole source of understanding how the format works. We invite the readers to reflect on this a bit.

The tool called `bedGraphToBigWig` in Jim Kent's utilities can perform this transformation from bedgraph to bigwig:

```
bedGraphToBigWig bam/SRR3033154.bam $REF.fai bam/SRR3033154.bw
```

We can automate these steps as well:

```
# Create the coverage files for all BAM files.  
ls bam/*.bam | parallel --eta --verbose "bedtools genomecov -ibam {} -g $REF.fai -bg | sort"
```

```
# Generate all bigwig coverages from bedgraphs.  
ls bam/*.bedgraph | parallel --eta --verbose "bedGraphToBigWig {} $REF.fai {}.bw"
```

Let's recapitulate what each file contains and what their sizes are:

- `SRR3033154.sam` (if generated) would be 5.3Gb, plain text, has alignment details, no random access.
- `SRR3033154.bam` is 423Mb, binary compressed format, has alignment details, allows random access.
- `SRR3033154.bedgraph` is 34Mb, plain text, coverage info only, loads fully into memory.
- `SRR3033154.bw` is 8.1Mb, binary format, coverage info only, allows random access.

We went from 5.3Gb to 8.1Mb, an approximately 1000 fold data size reduction. Hence, in general, when we intend to visualize these ChIP-Seq alignment files we transform them from BAM files into a **bigwig** files and subsequently, we only display these. Note that this same approach should be used when visualizing RNA-Seq coverage data as well.

109.9 Are there other ways to generate bigwig files?

There are other tools that can generate bigwig files directly from BAM files. The package that contains them is called `deeptools` and may be installed with:



Figure 109.3

```
# Create a new environment.
conda create --name bioinfo

# Activate the environment.
source activate bioinfo

# Install the deeptools.
conda install deeptools
```

One specific program of this package called `bamCoverage` offers a large number of functions to generate coverages. It is also able to produce outputs with various smoothing, averaging and normalizing parameters:

```
# Counts within a window of 50bp.
bamCoverage -b bam/SRR3033154.bam -o bam/SRR3033154-digitized.bw

# Smooth the signal in a window.
bamCoverage -b bam/SRR3033154.bam --smoothLength 300 -o bam/SRR3033154-smooth.bw
```

We recommend a lot of caution when applying any of these data altering, methods. Use them only when there are good reasons and a well-defined rationale for using them. Never “borrow” settings from another analysis or publication. In general, the best bet is to use the original coverage and resort to smoothing and averaging methods only if these appear to be necessary.

109.10. WHAT IS THE NEXT STEP OF ANALYZING THE DATA? 859

As an example here is the comparison between the original coverage, on the top and a smoothed data based on the coverage at the bottom. As you can see for yourself, the data characteristics change quite a bit - sometimes this is desirable but not always.

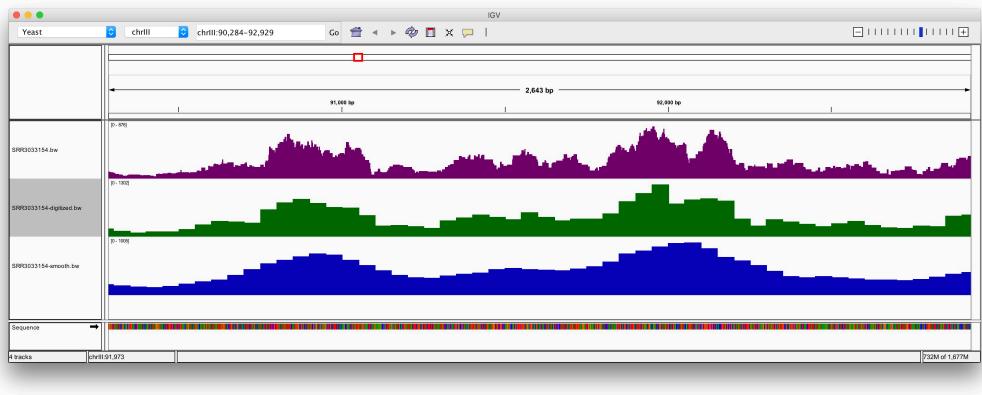


Figure 109.4

109.10 What is the next step of analyzing the data?

Once you have the alignments and coverages you can move onto the peak calling processes.

Chapter 110

Chip-Seq peak calling

This chapter continues with the data processing started in

- ChIP-Seq alignments.

We assume that you have the BAM alignment and coverage files.

110.1 How do I reanalyze a ChIP-Seq experiment?

As you recall we are trying to reanalyze data from the following publication:

- Genome-Wide Mapping of Binding Sites Reveals Multiple Biological Functions of the Transcription Factor Cst6p in *Saccharomyces cerevisiae*.¹ published in MBio in 2016

Whereas we can download the data from SRA the publication offers substantially less information on how the analysis itself was performed. It briefly states that:

To identify peaks and compare biological duplicates, the program GEM (39) was used.

¹<https://www.ncbi.nlm.nih.gov/pubmed/27143390>

Now if you were to visit the GEM homepage² you'll see that the tool takes a wide variety of parameters - each of which has consequences. Another line states:

The noise level was calculated from averaged noise throughout each replicate.

You may not fully understand what this means in practice- we don't either. Finally, you'll find the following statement:

"Binding events were manually curated in the Integrative Genomics Viewer to remove redundant events representing the same binding and false-positive events with poor peak shape or poor duplication"

I think we can all agree that this is not an objective process that would allow you to repeat the selection for yourself. What you will find that a typical ChIP-Seq data analysis is characterized by surprisingly subjective decision making. You will have quite a hard time following the analysis as the processes used to generate the results will often rely on seemingly subjective decisions.

110.2 Can/should I contact the author of a study?

Yes. We too have contacted the authors of the study and they, in turn, have been very responsive to my questions. I would like to thank both Dr. Jens Nielsen and David Bergenholm for being most speedy and accommodating with their time.

Yet even so, as you yourself will experience it, if you ever undertake a reanalysis of this type, the process of moving forward with reproducing the results may still proceed at a surprisingly slow pace.

Somehow every single question/response pair missed an important detail, an

²<http://groups.csail.mit.edu/cgs/gem/>

essential yet perhaps tacitly assumed decision, one that required yet another followup email.

After twenty (!) or so email exchanges with the author I was finally able to simplify the process to the following script `chip-exo-redo.sh`³ that in turn, I believe to contain everything one needs to redo the analysis.

Notably, while the script contains most reproducible commands that I believe were used in the paper, the results do not exactly match (but are similar to) the published ones.

110.3 Should I first summarize my data ?

Some like to simplify and summarize their data by merging all replicates into a single file and comparing them. By doing this merging, you will, of course, lose the independent measurements. On the other hand, it may allow you to gain a higher level understanding of all of its properties.

```
samtools merge -r bam/glucose.bam bam/SRR3033154.bam bam/SRR3033155.bam
samtools merge -r bam/ethanol.bam bam/SRR3033156.bam bam/SRR3033157.bam
samtools index bam/glucose.bam
samtools index bam/ethanol.bam
```

Generate the coverages for each, here you may skip making bigwig files since these files are not as large and IGV will handle them as is:

```
bedtools genomecov -ibam bam/glucose.bam -g $REF.fai -bg > bam/glucose.bedgraph
bedtools genomecov -ibam bam/ethanol.bam -g $REF.fai -bg > bam/ethanol.bedgraph
```

We can clearly see that in the region that we have visualized, the glucose sample has a more uniform coverage throughout whereas the ethanol affected sample shows a differential binding event. Auto-scaling, in this case, may be misleading, it appears as if the upper track has more data yet both have a similar number of reads 48 million versus 39 million.

The main difference is that the scale on the upper track goes to 1436 whereas the scale of the lower track goes to 5642.

³<http://data.biostarhandbook.com/chipseq/chip-exo-redo.sh>

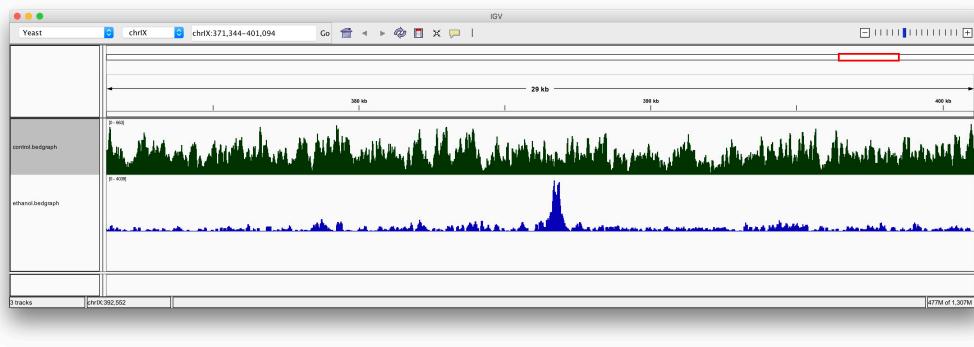


Figure 110.1

110.4 What tools can I use to predict peaks?

There is no shortage of tools that one can invoke to predict peaks. The most well-known is called MACS: Model-based Analysis for ChIP-Seq. Even for this tool there are two versions of it `macs` and `macs2` and each can be installed with conda or home brew.

Since these are python based tools they are best installed into a separate environment:

```
# Create a namespace for the tool
conda create --name macs python=2.7

# Activate the new environment.
source activate macs

# Install the tools.
conda install numpy
conda install macs2
```

You can also install the GEM peak predictor designed specifically for ChIP-Exo data - this is the tool used in the paper, and we'll use it as well later. Let's call peaks with `macs2`:

```
# Glucose samples.
GLU1=bam/trimmed-SRR3033154.bam
GLU2=bam/trimmed-SRR3033155.bam
```

```
# Ethanol samples.
ETH1=bam/trimmed-SRR3033156.bam
ETH2=bam/trimmed-SRR3033157.bam

macs2 callpeak -t $ETH1 $ETH2 -c $GLU1 $GLU2 --gsize 1E7 --name ethanol --outdir ethanol
```

The results of this peak calling process will go into the directory `ethanol` where a number of files are created. The file called `ethanol_peaks.bed` contains the regions that form peaks.

Visually inspecting the results show the regions that MACS believes to form “peaks”. As a comparison we also plot in a track the final results of the paper that would indicate the binding locations. As we can see `macs2` does reasonably good job in finding the locations, but at the same time the peaks don’t quite seem to be well localized, they spread over larger regions.

110.5 How do I refine the peaks?

The `ethanol_summits.bed` file contains the most likely maxima of each peak. These are single base pair wide location computed at the same time as the rest of the peaks:

Subsequent analysis typically consists of processing these summits into one place (if there are multiple of them overlapping in one region) and filtering for the most likely binding location. Alas neither process is entirely objective, much effort and custom coding goes into refining the final positions

As we mentioned before the parameter settings for filtering are almost always “reverse engineered.” There are expectations, and then in turn the parameters are tuned in the direction that strengthens these expectations. For example, we could keep only peaks with a score above 10 with:

```
cat ethanol/ethanol_summits.bed | awk '$5 > 10 { print $0 }' > highscore.bed
```

Of course, here we treading into the dangerous territory of data manipulation - that is artificially selecting data with some properties. Unfortunately, this is a risk we must take in this case - as objective measures are few and far between.

110.6 What else can I do to refine the peaks?

Due to the biased nature of the sequencing protocols ChIP-Seq data is typically affected by a very higher number of false positives. Thus the most common necessary step is that of removing some (or many) of the resulting peaks.

One such example is that of filtering out repetitive or low complexity regions of the genome. Data from these areas almost always indicate false signals as, if there is a measurement that matches equally

The `sdust` program of the [minimap][<https://github.com/lh3/minimap>] can be used to create an interval file with the regions of low complexity:

```
sdust refs/sc.fa > lowcomplexity.bed
```

then you could keep the high scoring values that do not fall on low complexity regions with:

```
bedtools intersect -v -a highscore.bed -b lowcomplexity.bed > highcomplexity.bed
```

You could also filter to keep only peaks that fall upstream of gene starts. Again bedtools could help:

```
bedtools flank -g refs/sc.fa.fai -l 1000 -r 0 -s -i refs/genes.gff > flanked.gff
```

Then intersect the peaks with that

```
bedtools intersect -u -a ethanol_summits.bed -b flanked.gff > upstream-peaks.bed
```

These types of tweaking and filtering are perhaps the most time-consuming aspects of a ChIP-Seq analysis - they are part of the quest to refine the very high false positive rates to a more reliable subset.

In the end the success of the ChIP-Seq analysis gets decided right here at peak filtering. Every single subsequent process operates on the position and sequence information of these peaks hence these peak filtering have the most effect on the results.

At the same time, and we reiterate, there is the danger that you may create an artificial, non-representative smaller subset of data.

110.7 How did the paper call peaks?

The paper used the GEM peak caller in the following way:

```
# Shortcuts.  
GEM=~/src/gem/gem.jar  
DIST=~/src/gem/Read_Distribution_ChIP-exo.txt  
  
# Run GEM on ethanol samples.  
java -Xmx3G -jar $GEM --d $DIST --g refs/sacCer3.chrom.sizes --genome refs --expt
```

and these will generate the peaks that most closely reproduce the published results.

Chapter 111

Chip-Seq motifs

This chapter continues with the data processing starting in both

- ChIP-Seq alignments.
- ChIP-Seq peak-calling
- ChIP-Seq data¹

Recall the section on Sequence Patterns, see Pattern matching where a sequence pattern (a motif) was defined as a sequence of bases described by certain rules. These rules could vary from requiring exact matches to being more elaborate requirements such as:

All locations where ATA is followed by one or more GCs and no more than five Ts then ending with GTA

111.1 Why do scientists consider sequence motifs to be important?

Sequence motifs are of particular interest in ChIP-Seq data as it is believed that these motifs control (or at least influence) many biological processes: for example the binding of transcription factors to DNA.

¹<http://data.biostarhandbook.com/chipseq/>

111.2 What makes motifs difficult to evaluate?

The most common challenge when dealing with motifs stems from not having a proper picture of just how often a given motif might be present in the genome, to begin with. Finding a motif in a binding location would be interesting only if seeing that motif has predictive value. Since the content of genome itself is not random the only way to properly assess the motif's abundance is to match it back to the genome.

Then, of course, a motif might turn out to be “statistically significant” in that it may be observed at a rate that is different from random chance. That alone is a weak observation. At the same time, you should also be aware of and take into account the “effect size”: just how big this deviation from random chance is.

111.3 How do I find known motifs?

The paper that we are re-analyzing indicates that the motif GTGACGT was found in the location of the peaks. This pattern is then represented by a so-called sequence logo and described as

A binding site motif of 5 GTGACGT 3 was enriched from the bound region sequences (discovered in 32 of the 40 regions)

A sequence (consensus) logo consist of a stack of letters where the size of each letter corresponds to the frequency of observation of each letter.

How do you find this motif in the genome? In the chapters Pattern matching we describe a few tools, below we'll use `seqkit` as an example:

```
REF=refs/sc.fa
```

```
# Turn the output into a GFF file:  
cat $REF | seqkit locate -p GTGACGT | head
```

will produce the output:

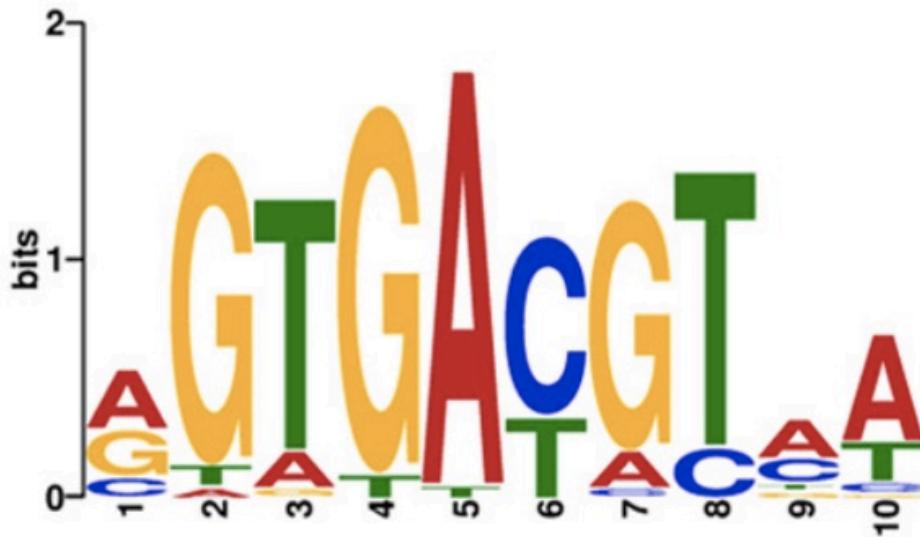


Figure 111.1: o

```

seqID patternName pattern strand start end matched
chrI GTGACGT GTGACGT + 10705 10711 GTGACGT
chrI GTGACGT GTGACGT + 19820 19826 GTGACGT
chrI GTGACGT GTGACGT + 84191 84197 GTGACGT
chrI GTGACGT GTGACGT + 101042 101048 GTGACGT
chrI GTGACGT GTGACGT + 120400 120406 GTGACGT
...

```

To visualize this result in IGV you can turn it into a 9 column GFF file with an `awk` incanation (add `.` for unknown columns):

```
cat $REF | seqkit locate -p GTGACGT | grep -v seqID | awk '{ OFS="\t"; print $1, ".", ".",
```

Visualizing the file you can see that there are 716 occurrences of the GTGACGT motif throughout the genome. The tracks below show the peaks found in the paper and the locations of the exact motif matches.

The published data finds 40 binding locations. Since there are 716 motif locations, it seems that even if all peaks fell on motifs only 5% of motif locations were bound by factors. Clearly, the presence of motif does not indicate binding. But as it turns out even that is not quite right. Let's find



Figure 111.2

out how many of the published peaks fall on motifs:

```
curl -O http://data.biostarhandbook.com/chipseq/peaks-published.bed
bedtools intersect -u -a peaks-published.bed -b motifs.gff | wc -l
# 4
```

It turns out only 4 of the reported peaks intersect exactly with this motif locations. So the wording of 32 of 40 regions containing the motif is probably interpreted differently - though again we don't know what it means. Most likely the word "region" means a wider interval, not just the reported peak coordinate. Let's extend these peaks:

```
bedtools slop -b 50 -g refs/sacker3.fa.fai -i peaks-published.bed > wider.bed
bedtools intersect -u -a wider.bed -b motifs.gff | wc -l
# 11
```

Extending peaks by 50 bases left and right produced 11 more hits. We're getting there but not quite there yet.

Of course, as you may have already noticed the motif above does not need to be exact - it allows for mismatches. On the other hand of course when we allow a motif to be inexact the count for its locations rises, often dramatically.

111.4 How many “degenerate” motifs in the genome?

The sequence logo as drawn in the paper allows for some variations, but just how many it covers is difficult to visually assess. For all their visual flair and seeming “logic” sequence logos can often be misleading. They show the most common letter, but we don’t see the individual motifs - only a superposition of all states. Let’s run our pattern locator by allowing two alternatives at three locations:

```
cat $REF | seqkit locate -p 'G(T|A)GA(C|T)(G|A)T' | wc -l | grep -v seqID | awk '{ OFS="\t"'
```

It now produces 11349 possible matches! That is a tenfold increase over the exact motif! Note just how dense the motif coverage over the genome has become.

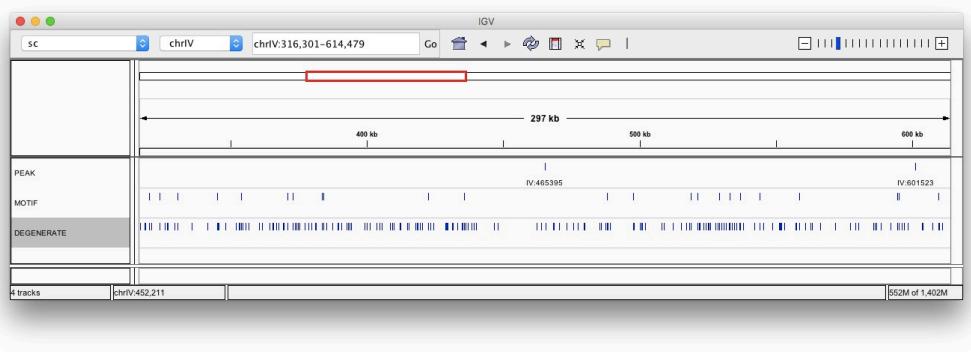


Figure 111.3

Even so, this pattern has picked up just two new peaks, while raising the total non-relevant locations for the motif 15 fold. Not a good tradeoff! That still does not reproduce all the motifs as described in the paper.

111.5 How do I call motifs?

The GEM peak predictor that besides calling peaks also calls motifs, though the authors appear to have not used those results, instead chose to use the MEME suite.

For example, if you were to attempt to find motifs for the peaks called with methods shown in the previous chapter you would need to do the following:

1. take the peak locations,
2. extend them left and right by a fixed amount,
3. extract the sequences that these intervals cover then
4. run a motif search program on these sequences

The script might look like this:

```
BED=ethanol/ethanol_summits.bed
REF=refs/saccer3.fa
N=50
```

```
# Extend the peak locations by 50 basepairs left and right.
bedtools slop -g $REF.fai -i $BED -b $N > extended.bed

# Extract the sequences corresponding to these locations.
bedtools getfasta -bed extended.bed -fi $REF -fo sequences.fa

# Run meme-chip on the sequences.
rm -rf results
meme-chip -meme-maxw 10 -meme-minw 5 -o results sequences.fa
```

These commands will generate a directory called **results** that among other information will contain the logo. In this case the logo looks like:



Figure 111.4

Obviously, this motif is not similar to the published motif at all. It appears to have a low complexity to it, indicating that our sequences represent too many repetitive regions. A good reminder that many/most peak prediction

methods still produce a high number of false positives and that repetitive regions can influence results. Replace the BED file above with the different filtered files that we also suggested before: the `highscoring.bed` then the `highcomplexity.bed` file and rerun the script to see how the motif evolves. In the latter case we get the following:



Figure 111.5

This motif now matches the published result, though we can recognize the relative frequencies of the letters is different. If anything this motif may look a little cleaner than the published one. In addition we have filtered the data no by eye but with an objective measure of taking into account the genome complexity.

111.6 What is the conclusion on finding motifs?

When it comes to motifs the results appear to be more subjective than the scientific field currently cares to admit. As much as we would all like to find a simple explanation of why a factor binds to a certain location, why a phenomenon is localized in the genome the evidence shows that the influence of the sequences and motifs are less well defined than we hoped for.

In our opinion motifs are pursued primarily because they appear to bring objectivity and determinism as to the why a factor binds to a given location - in reality, this effect is much weaker than many assume.

111.7 Have motifs ever misled scientists?

The most educational cautionary tale of motif finding comes from the now retracted paper titled Genomic organization of human transcription initiation

complexes² published in Nature in 2013 (retracted in 2014 by the authors). In it the authors report sequence patterns of the form:

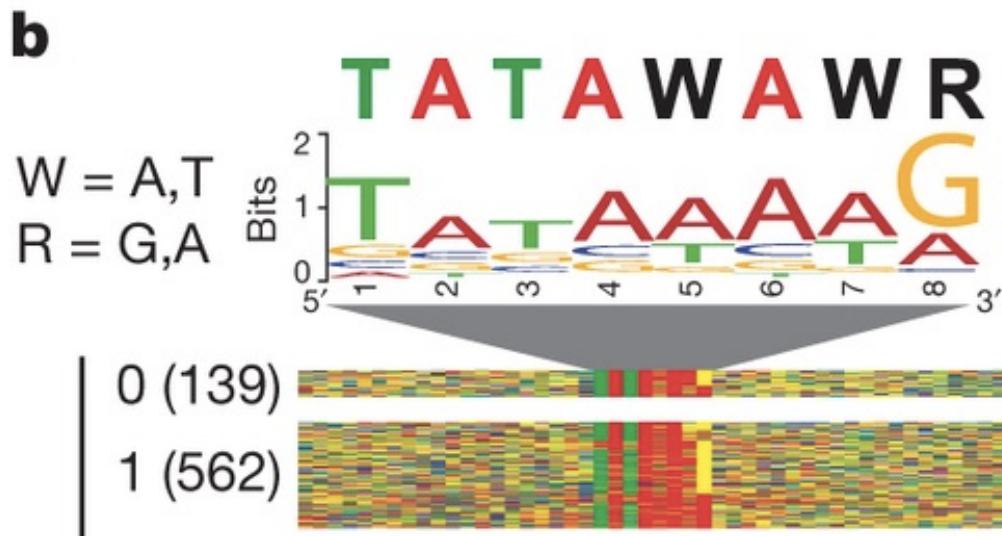


Figure 111.6

As it turns, it has been found that due to the degeneracy of the patterns the resulting motifs could have been produced just by chance in any preselected region of the genome. While the authors have attempted to correct and verify that the motif appears at rates above this chance they performed the comparison incorrectly and hence believed their results as being novel.

It does not help that the motif itself is so “obviously” clear cut. It is quite hard to believe that this pattern could be due to random chance. The goal here not to criticize the authors but to learn from them - you’ll have to protect yourself from errors like with ongoing vigilance.

The lesson here is to understand and remember to treat motifs with a healthy dose of skepticism and to recognize their valuable yet also limited information content. It is very unlikely that whatever mechanisms you are after will be explained by the presence/absence of motifs alone. The DNA is composed of only four bases A,T,G and C. This low variety of choices coupled to a large number of measurements and samples can easily produce results that seem impossible to occur by chance.

²<http://www.nature.com/nature/journal/v502/n7469/full/nature12535.html>

Part XXV

Ming Tang's ChIP Seq

Chapter 112

Ming Tang's guide to ChIP-Seq analysis

The author of this guide is Ming Tang¹. The material was developed for the Biostar Handbook.

- GitHub profile²
- Diving into Genetics and Genomics³

112.1 What is ChIP-seq?

Chromatin immunoprecipitation (ChIP) followed by high-throughput DNA sequencing (ChIP-seq) is a technique to map genome-wide transcription factor binding sites and histone-modification enriched regions.

Briefly, DNA bounding proteins and DNA (Chromatin) are cross-linked by formaldehyde and the chromatin is sheared by sonication into small fragments (typically 200 ~ 600 bp). The protein-DNA complex is immnuoprecipitated by an antibody specific to the protein of interest. Then the DNA is purified and made to a library for sequencing. After aligning the sequencing reads to a reference genome, the genomic regions with many reads enriched are where

¹<https://github.com/crazyhottommy>

²<https://github.com/crazyhottommy>

³<http://crazyhottommy.blogspot.hu/>

the protein of interest bounds. ChIP-seq is a critical method for dissecting the regulatory mechanisms of gene expression.

112.2 What are the processing steps for ChIP-seq data?

The general processing steps are as following:

1. Quality control of your `fastq` read files using tools such as FASTQC⁴. Read our previous section: Visualizing sequencing data quality⁵.
2. Aligning `fastq` reads to reference genome. The most popular read aligners are `bwa` and `bowtie`. Read section Short Read Aligners⁶. `bowtie` has two versions: `bowtie1` and `bowtie2`. `bowtie2` is better for reads length greater than 50bp. It is still very common to use 36bp single-end sequencing library for ChIP-seq, so `bowtie1` is preferred for ChIP-seq with short reads.
3. Calling peaks from the alignment `bam` files.
4. Visualizing the resulting peak files and raw signal files.

112.3 What are IgG control and input control for ChIP-seq?

Like any other experiments, a control is needed for ChIP-seq. There are two kinds of controls for ChIP-seq: IgG control and input control. IgG control is DNA resulting from a “mock” ChIP with Immunoglobulin G (IgG) antibody, which binds to non-nuclear antigen; input control is DNA purified from cells that are cross-linked, fragmented, but without adding any antibody for enrichment.

One problem with IgG control is that if too little DNA is recovered after immunoprecipitation(IP), sequencing library will be of low complexity and

⁴<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

⁵<https://read.biostarhandbook.com/data/fastq-quality-visualization.html>

⁶<https://read.biostarhandbook.com/align/short-read-aligners.html>

binding sites identified using this control could be biased. Input DNA control is ideal in most of the cases. It represents all the chromatin that was available for IP. Read this biostars post⁷ for discussion.

112.4 Data sets

To illustrate all the analysis steps, we are going to use a public data set from two papers:

- Genome-wide association between YAP/TAZ/TEAD and AP-1 at enhancers drives oncogenic growth⁸. Francesca Zanconato et.al. 2014. *Nature Cell Biology*. We will use transcription factor YAP1 ChIP-seq data in MDA-MB-231 breast cancer cells from this paper.
- Nucleosome positioning and histone modifications define relationships between regulatory elements and nearby gene expression in breast epithelial cells⁹. Suhn Kyong Rhie et.al. 2014. *BMC Genomics*. We will use H3K27ac ChIP-seq data in MDA-MB-231 cells from this paper.
- The data sets can be found with accession number GSE66081¹⁰ and GSE49651¹¹.

112.5 How to obtain the data?

Read our previous section: Accessing the Short Read Archive (SRA).

Prepare folders and download the data.

```
# go to your home directory
cd ~
# make folders for this example
mkdir -p ChIP-seq/{data,results,scripts,software}
```

⁷<https://www.biostars.org/p/15817/>

⁸<https://www.ncbi.nlm.nih.gov/pubmed/26258633>

⁹<https://www.ncbi.nlm.nih.gov/pubmed/24885402>

¹⁰<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE66081>

¹¹<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE49651>

```

mkdir -p ChIP-seq/results/{bams,peaks,fastqc,motifs}
cd ChIP-seq/data

# for YAP1. It takes time, alternatively download fastq directly from EGA ht
fastq-dump SRR1810900

# for H3K27ac
fastq-dump SRR949140

# for input DNA
fastq-dump SRR949142

```

In general, do not change the names of files. This can be error prone, especially when you have many samples. Instead, prepare a name mapping file, and interpret the final results with the mapped names. For demonstrating purpose, I will re-name the files to more meaningful names.

```

mv SRR1810900.fastq YAP1.fastq
mv SRR949140.fastq H3K27ac.fastq
mv SRR949142.fastq Input.fastq

```

Now, run FASTQC on all the samples. Read previous section on looping over files¹². or use GNU parallel:

```

# always specify -j to not abuse the computing cluster
find *fastq | parallel -j 4 fastqc {} -o ../results/fastqc

```

Overall, the sequencing qualities are good for all three samples, I will skip trimming low quality bases, and go ahead with the original fastq files for aligning.

¹²<https://read.biostarhandbook.com/unix/loops.html>

112.6 How to align ChIP-seq reads?

once you have finished QC confirming the sequencing reads are of high quality and with no adaptor contamination, you can do:

```
# use 10 threads to speed up. should finish within 10 mins
# change the path to the bowtie1 index in your own machine.
bowtie -p 10 --best --chunkmbs 320 /risapps/reference/bowtie1/hg19 -q YAP1.fastq -S .

# reads processed: 24549590
# reads with at least one reported alignment: 19664247 (80.10%)
# reads that failed to align: 4885343 (19.90%)
Reported 19664247 alignments to 1 output stream(s)

bowtie -p 10 --best --chunkmbs 320 /risapps/reference/bowtie1/hg19 -q H3K27ac.fastq -S .

# reads processed: 29759754
# reads with at least one reported alignment: 29100444 (97.78%)
# reads that failed to align: 659310 (2.22%)
Reported 29100444 alignments to 1 output stream(s)

bowtie -p 10 --best --chunkmbs 320 /risapps/reference/bowtie1/hg19 -q Input.fastq -S .

# reads processed: 18811563
# reads with at least one reported alignment: 18127743 (96.36%)
# reads that failed to align: 683820 (3.64%)
Reported 18127743 alignments to 1 output stream(s)
```

For demonstrating purpose, I will write down explicitly all the commands, but the GNU parallel trick can be used to save you some typing. In the end, I will introduce a shell script to chain all the steps together and show you how to make reusable codes.

Convert **sam** to sorted **bam**:

```
# avoid writing unsorted bam to disk
# note that samtools sort command changed invoke pattern after version 1.3 https://
```

```

# samtools view and index will be multi-thread soon https://github.com/samtools/samtools
cd ../results/bams
samtools view -bS YAP1.sam | samtools sort -@ 4 - -T YAP1 -o YAP1.sorted.bam
# index the sorted bam
samtools index YAP1.sorted.bam

samtools view -bS H3K27ac.sam | samtools sort -@ 4 - -T H3K27ac -o H3K27ac.sorted.bam
samtools index H3K27ac.sorted.bam

samtools view -bS Input.sam | samtools sort -@ 4 - -T Input -o Input.sorted.bam
samtools index Input.sorted.bam

# remove sam files to save space , rm can be dangerous, use rm -i if necessary
rm *sam

```

112.7 How do I call peaks?

MACS¹³ is short for Model Based Analysis for ChIP-Seq data, which was developed in Sheirly Liu's lab at Harvard by Tao Liu. It is one of the most popular peak-calling algorithms used in literatures. First, you need to install MACS.

```

## install through pip
pip install macs

# peak calling for human samples, with default p value 1e-5
macs -t YAP1.sorted.bam -c Input.sorted.bam -n YAP1 -g hs -p 1e-5 -o ../peaks/YAP1_peaks

## peak calling for H3K27ac
macs -t H3K27ac.sorted.bam -c Input.sorted.bam -n H3K27ac -g hs -p 1e-5 -o ../peaks/H3K27ac_peaks

```

In addition to `bam` files, `macs` can take in other input file formats such as `bed` and `sam`. type `macs` on your terminal and press `Enter` to see full list of helps.

¹³<http://liulab.dfci.harvard.edu/MACS/00README.html>

Parameter settings can be different depending on your own data sets. There is no one-fit-all settings. You may want to load the called peaks in to IGV and visually inspect them.

Note: Tao Liu is developing MACS2¹⁴. there is a `-broad` option to call broad peaks for histone modification ChIP-seq(e.g. H3K36me3). However, in my experience, `macs14` is still widely used and performs very well.

Take a look at How to set MACS2 peak calling parameters¹⁵.

112.8 How do I call super-enhancers?

The fancy “super-enhancer” term was first introduced by Richard Young¹⁶’s lab in Whitehead Institute. Basically, super-enhancers are clusters of enhancers (most commonly defined by H3K27ac peaks) stitched together if they are within 12.5kb apart. The concept of super-enhancer is NOT new. One of the most famous example is the Locus Control Region (LCR) that controls the globin gene expression, and this has been known for decades. If you are interested in the controversy, please read:

A review in Nature Genetics What are super-enhancers?¹⁷

Another comment in Nature Genetics Is a super-enhancer greater than the sum of its parts?¹⁸

From the HOMER page¹⁹

How finding super enhancers works:

Super enhancer discovery in HOMER emulates the original strategy used by the Young lab. First, peaks are found just like any other ChIP-Seq data set. Then, peaks found within a given distance are ‘stitched’ together into larger regions (by default this is

¹⁴<https://github.com/taoliu/MACS>

¹⁵https://github.com/crazyhottommy/ChIP-seq-analysis/blob/master/part1.3_MACS2_peak_calling_details.md

¹⁶<http://younglab.wi.mit.edu/publications.htm>

¹⁷<http://www.nature.com/ng/journal/v47/n1/full/ng.3167.html>

¹⁸http://www.nature.com/ng/journal/v49/n1/full/ng.3759.html?WT.feed_name=subjects_molecular-biology

¹⁹<http://homer.salk.edu/homer/ngs/peaks.html>

set at 12.5 kb). The super enhancer signal of each of these regions is then determined by the total normalized number reads minus the number of normalized reads in the input. These regions are then sorted by their score, normalized to the highest score and the number of putative enhancer regions, and then super enhancers are identified as regions past the point where the slope is greater than 1.

HOMER²⁰ is a very nice tool for finding enriched peaks, doing motif analysis and many more. It requires making a tag directory first. ROSE²¹ is the tool from Richard Young's lab. Since it works directly with `bam` files, we will use it for demonstration.

Download ROSE to the `software` folder.

```
cd ../../software/
wget https://bitbucket.org/young_computation/rose/get/1a9bb86b5464.zip
unzip 1a9bb86b5464.zip
cd young_computation-rose-1a9bb86b5464/

# there are multiple python scripts in the folder.
# one has to run ROSE inside the ROSE folder.
python ROSE_main.py -g hg19 -i ../../results/peaks/H3K27ac_peaks.bed -r ../../
```

112.9 How do I get a normalized bigWig track for visualizing the raw signal?

After you get the peak calls, we want to inspect the quality of the peaks by visualizing them in a browser such as IGV along with the raw signal track. We will use `bigWig` track for this purpose.

From the UCSC help²²:

²⁰<http://homer.salk.edu/homer/index.html>

²¹http://younglab.wi.mit.edu/super_enhancer_code.html

²²<http://genome.ucsc.edu/goldenpath/help/bigWig.html>

112.9. HOW DO I GET A NORMALIZED BIGWIG TRACK FOR VISUALIZING THE RAW SIGNAL?

The bigWig format is for display of dense, continuous data that will be displayed in the Genome Browser as a graph. BigWig files are created initially from wiggle (wig) type files, using the program wigToBigWig. The resulting bigWig files are in an indexed binary format. The main advantage of the bigWig files is that only the portions of the files needed to display a particular region are transferred to UCSC, so for large data sets bigWig is considerably faster than regular wiggle files

Make sure you understand the other two closely related file formats: bedgraph²³ and wig²⁴ file.

macs14 can generate a bedgraph file by specifying --bdg, but the resulting bedgraph is **NOT** normalized to sequencing depth²⁵. Instead, we are going to use another nice tool deeptools²⁶ for this task. It is a very versatile tool and can do many other things.

Make a bigWig file:

```
# install deeptools
conda install -c bioconda deeptools

# normalized bigWig with Reads Per Kilobase per Million mapped reads (RPKM)
bamCoverage -b YAP1.sorted.bam --normalizeUsingRPKM --binSize 30 --smoothLength 300 -

bamCoverage -b H3K27ac.sorted.bam --normalizeUsingRPKM --binSize 30 --smoothLength 300

bamCoverage -b Input.sorted.bam --normalizeUsingRPKM --binSize 30 --smoothLength 300
```

I set --extendReads to 200bp which is the fragment length. Why should we extend the reads? Because in a real ChIP-seq experiment, we fragment the genome into small fragments of ~200bp, and pull down the protein bound DNA with antibodies. However, we only sequence the first 36bp(50bp, or

²³<http://genome.ucsc.edu/goldenpath/help/bedgraph.html>

²⁴<http://genome.ucsc.edu/goldenpath/help/wiggle.html>

²⁵https://github.com/crazyhottommy/ChIP-seq-analysis/blob/master/part1.2_convert_bam2_bigwig.md#how-macs12-output-wigbedgraph-files-are-produced

²⁶<https://github.com/fidelram/deepTools>

100bp depending on your library) of the pull-down DNA. To recapitulate the real experiment, we need to extend it to the fragment size.

You can read more details Why do we need to extend the reads to fragment length/200bp?²⁷

112.10 How do I put all the steps together?

112.10.1 Shell script comes to rescue

Up until now, we have finished the basic analyzing steps for ChIP-seq. We aligned the reads to get `bam` files; we called peaks using `macs`; we generated `bigWig` tracks as well. This is great, but one of the stumbling blocks for beginners is learning how to reuse the commands we have typed. We do not want to type the same command time and time again changing the file names only when new data come in.

The strategy is to write a shell script to chain all the steps together. see the section writing scripts²⁸ in the handbook.

First, you need to think about how the workflow should be. We will need to download the `fastq` files and convert them to `bam` files; next, we will call peaks for a pair of IP and Input. Sometimes, you may get sorted `bam` files, which can be used directly to call peaks. It is reasonable that we have two shell scripts for each task, respectively.

```
#!/bin/bash

set -euo pipefail

# the SRA number to work with
SRA=$1

# the reference file path, change to where you put the reference
```

²⁷https://github.com/crazyhottommy/ChIP-seq-analysis/blob/master/part1.2_convert_bam2_bigwig.md#why-do-we-need-to-extend-the-reads-to-fragment-length200bp

²⁸<https://read.biostarhandbook.com/unix/writing-scripts.html>

```

REF="/risapps/reference/bowtie1/hg19"

##extract the fastq files
fastq-dump $SRA

## step1, quality control of the fastq files
fastqc "${SRA}".fastq

## step2, align the reads with bowtie
bowtie -p 10 --best --chunkmbs 320 $REF -q "${SRA}".fastq -S "${SRA}".sam

## step3, convert sam to bam, and index the bam
samtools view -bS "${SRA}".sam | samtools sort -@ 4 - -T "${SRA}" -o "${SRA}".sorted.
samtools index "${SRA}".sorted.bam

# remove sam to save space
rm "${SRA}".sam

```

For more on `set -euo pipefail`, read this post Use the Unofficial Bash Strict Mode (Unless You Looove Debugging)²⁹.

Save it to `sra2bam.sh`. Make it executable `chmod u+x sra2bam.sh`. execute:

```

# YAP1
./sra2bam.sh SRR1810900

# H3K27ac
./sra2bam.sh SRR949140

# Input
./sra2bam.sh SRR949142

```

Now, with the `sra2bam.sh` script, it saves you from typing different file names for aligning `fastqs` and converting `sam` files. Moreover, it can be used

²⁹<http://redsymbol.net/articles/unofficial-bash-strict-mode/>

to process any number of `sra` files.

If you have a `sra_id.txt` file with one `sra` id on each line, you can:

```
## run 6 jobs in parallel
cat sra_ids.txt | parallel -j 6 ./sra2bam {}
```

to process all of them.

second shell script `bam2peaks.sh`:

```
#!/bin/bash

set -euo pipefail

IP_bam=$1
Input_bam=$2
oprefix=$(basename "${IP_bam}" .sorted.bam)

macs -t "${IP_bam}" -c "${Input_bam}" -n "${oprefix}" -g hs -p 1e-5 -o ${opref
```

The `sra2bam.sh` script will generate the indexed `bam` files, those `bam` files can be fed into `bam2peaks.sh` to call peaks. In our example data set, we only have two IPs and one Input. We can call peaks by:

```
# call peaks for H3K27ac
./bam2peaks.sh SRR949140.sorted.bam SRR949142.sorted.bam

# call peaks for YAP1
./bam2peaks.sh SRR1810900.sorted.bam SRR949142.sorted.bam
```

Imagine we have a lot of `bam` files generated by `sra2bam.sh`, and we want to call peaks for them, how should we stream the workflow?

Because it involves a pair of files (IP and Input control) for calling peaks, we need to make a tab delimited `txt` file with pairs of file names on each line.

```
cat bam_names.txt
SRR949140.sorted.bam    SRR949142.sorted.bam
SRR1810900.sorted.bam   SRR949142.sorted.bam
```

Now, we can loop over the `bam_names.txt` file one by one and call peaks:

```
cat bam_names.txt | while read -r IP Input
do
    ./bam2peaks.sh "${IP}" "${Input}"
done
```

112.10.2 Arguments handling for shell script

What we have so far is all good, but what if you want to make your script more flexible? e.g. you want to specify mouse genome for mapping reads. You can add arguments for your script.

```
#!/bin/bash

set -euo pipefail

# show help
show_help(){
cat << EOF
This is a wrapper to align fastq reads to bam files for ChIP-seq experiments.
usage: ${0##*/} -d < a directory path containing the fastq.gz files > -r < h or m>
       -h display this help and exit
       -f the full path name of the fastq file
       -r reference genome to be used. m for mouse; h for human
EOF
}

## if there are no arguments provided, show help
if [[ $# == 0 ]]; then show_help; exit 1; fi
```

```

## parsing arguments
while getopts ":hf:r:" opt; do
    case "$opt" in
        h) show_help;exit 0;;
        f) fq=$OPTARG;;
        r) REF=$OPTARG;;
        '?') echo "Invalid option $OPTARG"; show_help >&2; exit 1;;
    esac
done

## set up some defaults
REF=${REF:-"h"}

## check if the fastq file exist
if [ ! -f "$fq" ]; then
    echo "file ${fq} does not exist"
    exit 1
fi

## reference genome path for mouse and human
human_ref="/risapps/reference/bowtie1/hg19"
mouse_ref="/risapps/reference/bowtie1/mm9"

## which species? human or mouse?
if [[ $REF == "m" ]]; then
    ref_genome="${mouse_ref}"
elif [[ $REF == "h" ]]; then
    ref_genome="${human_ref}"
else
    echo "please only specify m or h for the reference genome"
    exit 1
fi

## extract output name
oprefix=$(basename "${fq}" .fastq)

```

```

## mapping
bowtie -p 10 --best --chunkmbs 320 ${ref_genome} -q "${fq}" -S "${oprefix}".sam

## convert sam to sorted bam
samtools view -bS "${oprefix}.sam" | samtools sort -@ 4 - -T "${oprefix}" -o "${opref

# index sorted bam
samtools index "${oprefix}.sorted.bam

rm "${oprefix}.sam"

```

With this script, you can map not only ChIP-seq data for human but also for mouse. You can surely add more arguments to set `bowtie` mapping thread numbers and `samtools sort` thread numbers (the above script uses 10 and 4 respectively). You can read more on arguments handling for shell scripts: small getopt tutorial³⁰.

Congratulations! You have come this far with me. If you want to take the next level of making your analysis reproducible and flexible, read our Using Makefiles section.

112.11 What are the black-listed regions?

For ChIP-seq, it is important to filter artifact regions that has abnormal high signals. From the website of Anshul Kundaje³¹ in Stanford University:

These regions are often found at specific types of repeats such as centromeres, telomeres and satellite repeats. It is especially important to remove these regions that computing measures of similarity such as Pearson correlation between genome-wide tracks that are especially affected by outliers

One can download the hg19 blacklist:

³⁰http://wiki.bash-hackers.org/howto/getopts_tutorial

³¹<https://sites.google.com/site/anshulkundaje/projects/blacklists>

```
wget https://www.encodeproject.org/files/ENCFF001TDO/@download/ENCFF001TDO.bed.gz

# 411 regions are black-listed
zless ENCFF001TDO.bed.gz | wc -l
411
```

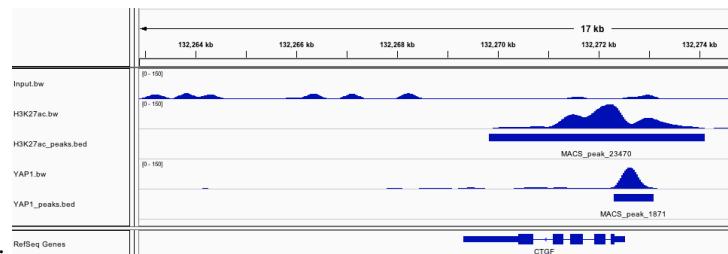
Note that more and more sequencing projects are moving their reference genome to the latest version GRCh38. GRCh38 blacklist was uploaded by Anshul Kundaje on 2016-10-16. The file can be downloaded by:

```
wget https://www.encodeproject.org/files/ENCFF419RSJ/@download/ENCFF419RSJ.bed.gz
```

112.12 How do I visualize the peaks?

We will use IGV to visualize the peaks and raw signal. check our previous section on using IGV³².

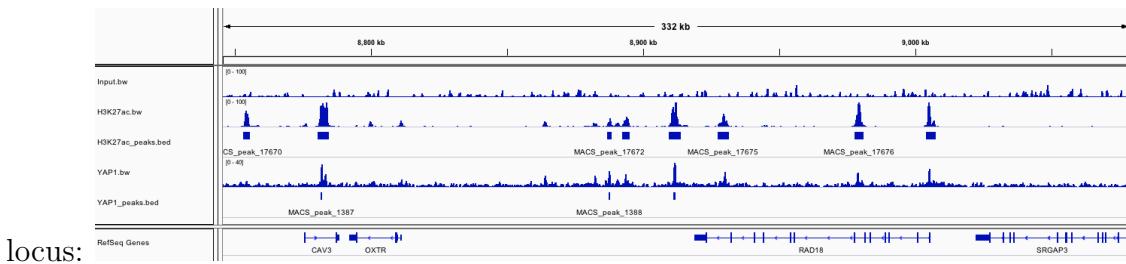
Open IGV, click File, then Load From File, choose the peak files end with **bed** and the raw signal files end with **bw**. you can change the scale of the **bigWig** tracks by right click the tracks and choose **Set Data Range**. Now, you can browser through the genome or go to your favorite genes.



YAP1 binds to known target gene *CTGF*:

one other example from the original *Nature Cell Biology* paper at the *RAD18*

³²<https://read.biostarhandbook.com/visualize/igv.html>



We see that `macs` did a pretty good job in identifying enriched regions for histone modifications and transcription factor (TF) binding peaks. However, we do also see some potential YAP1 binding sites are missed. Fine tuning the peak calling parameters may improve the results.

Chapter 113

ChIP-Seq Downstream Analysis 1

The author of this guide is Ming Tang¹. The material was developed for the Biostar Handbook.

- GitHub profile²
- Diving into Genetics and Genomics³

113.1 How do I compare the peak sets?

Now, we have the peaks called for H3K27ac and YAP1 and we want to exclude the peaks that overlap with the blacklisted regions. How should you do it? We will use a very popular tool `bedtools`⁴ developed by Aaron Quinlan lab⁵ for this purpose. Documentation of `bedtools` is of top niche. Also read Interval analysis tools⁶ section in the handbook.

There are many sub-commands for `bedtools`, but the most commonly used one is `bedtools intersect`:

¹<https://github.com/crazyhottommy>

²<https://github.com/crazyhottommy>

³<http://crazyhottommy.blogspot.hu/>

⁴<http://bedtools.readthedocs.io/en/latest/>

⁵<http://quinlanlab.org/>

⁶<https://read.biostarhandbook.com/tools/install.html#interval>

```
# make sure you are inside the folder containing the peak files and the black-listed regions
# unzip the file
gunzip ENCFF001TDO.bed.gz

# How many H3K27ac peaks overlap with black-listed regions?
bedtools intersect -a H3K27ac_peaks.bed -b ENCFF001TDO.bed -wa | wc -l
#14

# exclude those peaks
bedtools intersect -a H3K27ac_peaks.bed -b ENCFF001TDO.bed -v > H3K27ac_filtered_peaks.bed

# do the same for YAP1
bedtools intersect -a YAP1_peaks.bed -b ENCFF001TDO.bed -v > YAP1_filtered_peaks.bed
```

How many YAP1 peaks overlap with H3K27ac peaks?

```
bedtools intersect -a YAP1_filtered_peaks.bed -b H3K27ac_filtered_peaks.bed -wa | wc -l
#1882

bedtools intersect -a YAP1_filtered_peaks.bed -b H3K27ac_filtered_peaks.bed -wa | sort
#1882

bedtools intersect -a H3K27ac_filtered_peaks.bed -b YAP1_filtered_peaks.bed -wa | wc -l
#1882

bedtools intersect -a H3K27ac_filtered_peaks.bed -b YAP1_filtered_peaks.bed -wa | sort
#1772
```

what's happening here? why there are only 1772 unique H3K27ac peaks overlap with YAP1 peaks?

It turns out that bedtools will output the overlapping peaks of H3K27ac whenever there is an overlapping with YAP1 peaks, and it is possible that the same H3K27ac peak (tends to be really broad peaks) overlaps with multiple YAP1 peaks. With that in mind, I always do `sort | uniq` following `bedtools` command.

You can identify those H3K27ac peaks by:

```
bedtools intersect -a H3K27ac_filtered_peaks.bed -b YAP1_filtered_peaks.bed -w
  4 chr14  93495168  93513756  MACS_peak_8569  935.17
  4 chr20  30276949  30312747  MACS_peak_16217  3100.00
  3 chr10  95216260  95244053  MACS_peak_3871  3100.00
  3 chr11  65654426  65689186  MACS_peak_5045  3100.00
  3 chr14  23441596  23453313  MACS_peak_7876  3100.00
  3 chr1   86040764  86052311  MACS_peak_1241  3100.00
  3 chr1   86070388  86080738  MACS_peak_1243  2714.64
  3 chr19  13943280  13955945  MACS_peak_13069  3100.00
  3 chr19  13956264  13965827  MACS_peak_13070  3100.00
  3 chr1   95080161  95091780  MACS_peak_1347  3038.66
```

We see some H3K27ac peaks can overlap with up to 4 peaks of YAP1.

If you choose to believe your eyes, let's visualize it in IGV:

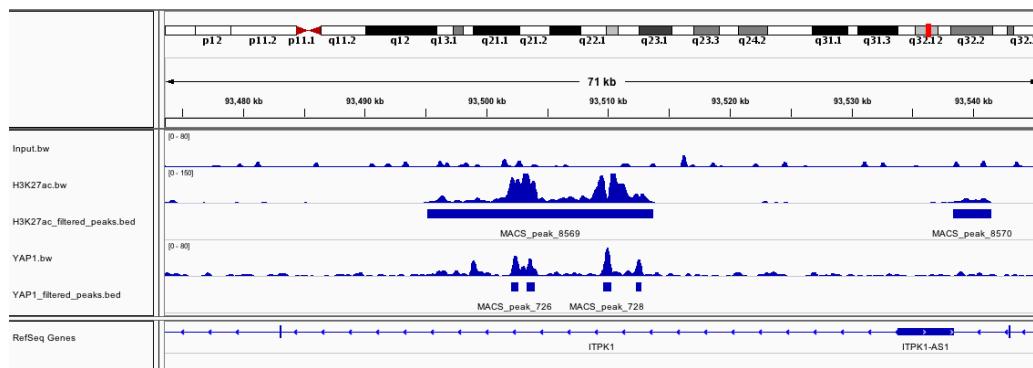


Figure 113.1

One of the under-appreciated tools is BEDOPS⁷. It has many nice features as well and the documentation is also very good. Read this biostar post: Question: Bedtools Compare Multiple Bed Files?⁸

⁷<https://bedops.readthedocs.io/en/latest/>

⁸<https://www.biostars.org/p/13516/>

113.2 How do I annotate my peaks?

The next obvious question is where are the peaks in terms of their genomic context. e.g. What genes are the peaks associated with? Are some of the peaks located in intergenic region? You need to **annotate** your peaks. There are many tools you can use. You can find a list here⁹.

I will show you how to do peak annotation using an R bioconductor¹⁰ package ChIPseeker¹¹ developed by Guangchuan Yu¹².

```
> # load the packages, install them following the instructions in the links above
> library(ChIPseeker)
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> library(rtracklayer)
> library("org.Hs.eg.db")
>
> txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

> # read in the peaks for YAP1
> YAP1<- import("YAP1_peaks.bed", format = "BED")
> YAP1
GRanges object with 2150 ranges and 2 metadata columns:
      seqnames          ranges strand |      name      score
      <Rle>          <IRanges>  <Rle> |      <character> <numeric>
 [1]   chr1  [ 959961,  960483]    * | MACS_peak_1  132.10
 [2]   chr1  [1295901, 1296269]    * | MACS_peak_2  59.44
 [3]   chr1  [1992556, 1993176]    * | MACS_peak_3  98.23
 [4]   chr1  [ 8963802, 8964418]    * | MACS_peak_4 126.02
 [5]   chr1  [16499970, 16500516]    * | MACS_peak_5 142.50
 ...
[2146]   chrX  [102911475, 102912040]    * | MACS_peak_2146  73.47
[2147]   chrX  [103168254, 103168801]    * | MACS_peak_2147  58.64
[2148]   chrX  [115403519, 115404015]    * | MACS_peak_2148  50.08
```

⁹<https://github.com/crazyhottommy/ChIP-seq-analysis#peak-annotation>

¹⁰<https://bioconductor.org/>

¹¹<http://bioconductor.org/packages/release/bioc/html/ChIPseeker.html>

¹²<https://github.com/GuangchuangYu>

```

[2149]      chrX [149252969, 149253470]      * | MACS_peak_2149 61.44
[2150]      chrX [153147396, 153148074]      * | MACS_peak_2150 61.75
-----
seqinfo: 23 sequences from an unspecified genome; no seqlengths

> YAP1_anno<- annotatePeak(YAP1, tssRegion=c(-3000, 3000),
                               TxDb=txdb, level = "gene", annoDb="org.Hs.eg.db",
                               sameStrand = FALSE, ignoreOverlap = FALSE,
                               overlap = "TSS")
> # some nice visualization you can do
> plotAnnoPie(YAP1_anno)
> upsetplot(YAP1_anno, vennpie=TRUE)

> # check the annotation
> head(as.data.frame(YAP1_anno))
  seqnames     start       end width strand      name   score
1   chr1  959961  960483    523      * MACS_peak_1 132.10
2   chr1 1295901 1296269    369      * MACS_peak_2  59.44
3   chr1 1992556 1993176    621      * MACS_peak_3  98.23
4   chr1 8963802 8964418    617      * MACS_peak_4 126.02
5   chr1 16499970 16500516    547      * MACS_peak_5 142.50
6   chr1 17454948 17455379    432      * MACS_peak_6  59.60
                                             annotation geneChr geneStart geneEnd
1 Intron (uc001ack.2/375790, intron 2 of 35)          1  955503  991499
2                               Promoter (<=1kb)          1 1288071 1297157
3 Intron (uc001aiq.3/5590, intron 4 of 17)          1 1981909 2116834
4                               Distal Intergenic        1  8921059  8939151
5                               Distal Intergenic        1 16450832 16482582
6                               Distal Intergenic        1 17393256 17445948
  geneLength geneStrand geneId distanceToTSS           ENSEMBL SYMBOL
1      35997          1  375790       4458 ENSG00000188157   AGRN
2       9087          2   54587        888 ENSG00000162576   MXRA8
3     134926          1   5590       10647 ENSG00000067606   PRKCZ
4     18093           2   2023      -24651 ENSG00000074800   EN01
5     31751           2   1969      -17388 ENSG00000142627   EPHA2
6     52693           2   11240      -9000 ENSG00000117115   PADI2

```

```

          GENENAME
1               agrin
2 matrix-remodelling associated 8
3           protein kinase C zeta
4           enolase 1, (alpha)
5           EPH receptor A2
6 peptidyl arginine deiminase, type II

> # you can save it to a txt file to your computer
> write.table(as.data.frame(YAP1_anno), "YAP1_peaks_anno.txt", row.names =F, col.names =T)

```

113.3 How do I do pathway enrichment analysis for the peaks?

Now, you have the genes that are associated with the peaks, you can do pathway enrichment analysis using the genes. I will use the `clusterProfiler` package developed by Guangchuan Yu as well:

```

library(clusterProfiler)

## GO term enrichment
ego <- enrichGO(gene      = as.data.frame(YAP1_anno)$SYMBOL,
                 OrgDb     = org.Hs.eg.db,
                 keytype   = "SYMBOL",
                 ont       = "BP",
                 pAdjustMethod = "BH",
                 pvalueCutoff = 0.01,
                 qvalueCutoff = 0.05)

# visualization
dotplot(ego, showCategory = 20)

## Kegg pathway enrichment, need the Entrez ID

```

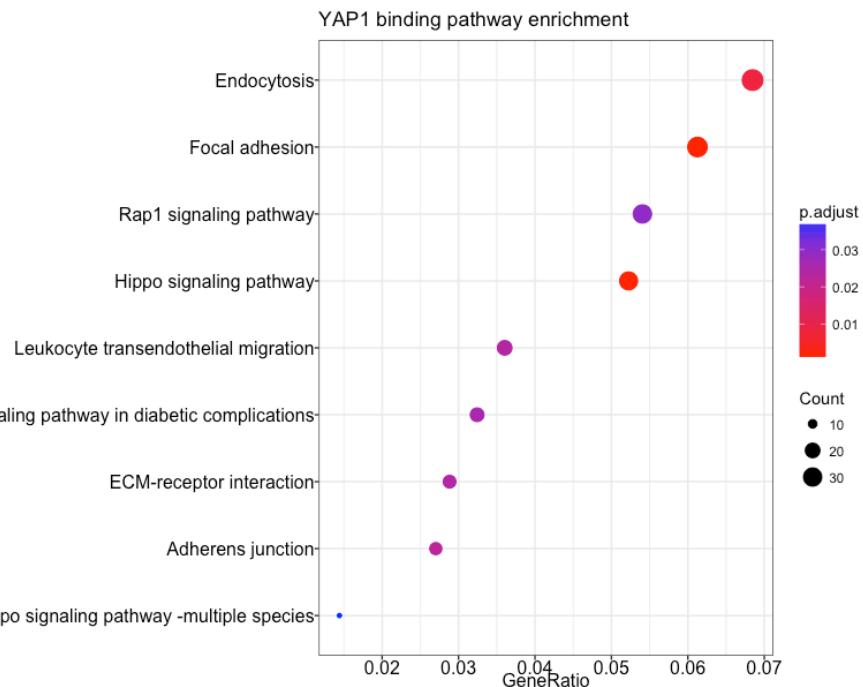
```

kk<- enrichKEGG(gene      = as.data.frame(YAP1_anno)$geneId,
                  organism    = 'hsa',
                  pvalueCutoff = 0.05)

dotplot(kk, showCategory = 20, title = "YAP1 binding pathway enrichment")

## you can write the result to a tsv file
write.table(kk@result, "YAP_KEGG_pathway_genes.txt", sep = "\t", col.names = T)

```



e.g.

```

library(dplyr)
## only consider genes within 5000 bp of the peaks

```

```
as.data.frame(YAP1_anno) %>% dplyr::filter(abs(distanceToTSS) < 5000)

## or you can rank the peaks by the Pvalue and choose the top 1000 peaks (this number
# score column is the -10*log10 Pvalue
as.data.frame(YAP1_anno) %>% dplyr::arrange(desc(score)) %>% head(n =1000)
```

The underlying mechanism is to compare the peaks associated gene set with the various annotated pathway gene sets to see whether the peak associated genes are overpresented in any of the known gene sets using hypergeometric test. Of course, many other complex algorithms have been developed for this type of analysis. Further reading: A Comparison of Gene Set Analysis Methods in Terms of Sensitivity, Prioritization and Specificity¹³

- GREAT¹⁴ predicts functions of cis-regulatory regions

113.4 How do I do motif analysis with the peaks?

One of the most popular tools is MEME-suite¹⁵ for motif enrichment analysis. I will demonstrate how to use MEME-ChIP¹⁶ for YAP1 peaks.

MEME-ChIP needs 500bp DNA sequences around the YAP1 binding summits (with the highest signal of binding), which is output by `macs14` when you call `peaks`.

Fetching the 500 bp DNA sequence around the YAP1 summits:

```
# get the coordinates of 500bp centered on the summit of the YAP1 peaks
cat YAP1_summits.bed | awk '$2=$2-249, $3=$3+250' OFS="\t" > YAP1_500bp_summits.bed

# you need a fasta file of the whole genome, and a bed file containing the coordinates
```

¹³<http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0079217>#pone-0079217-t001

¹⁴<http://bejerano.stanford.edu/great/public/html/>

¹⁵<http://meme-suite.org/>

¹⁶<http://meme-suite.org/tools/meme-chip>

```
# the whole genome fasta file can be gotten by:  
rsync -avzP rsync://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/ .  
cat *fa.gz > UCSC_hg19_genome.fa.gz  
gunzip UCSC_hg19_genome.fa.gz  
  
# Use bedtools getfasta http://bedtools.readthedocs.org/en/latest/content/tools/getfasta.html  
bedtools getfasta -fi UCSC_hg19_genome.fa -bed YAP1_500bp_summits.bed -fo YAP1_500bp.fasta
```

Now, upload it to MEME-ChIP:

MEME-ChIP
Motif Analysis of Large Nucleotide Datasets

Version 4.11.2

MIME-ChIP performs **comprehensive motif analysis** (including motif discovery) on **LARGE** (50MB maximum) sets of sequences (typically **nucleotide**) such as those identified by ChIP-seq or CLIP-seq experiments (**sample output** from **sequences**). See this **Manual** for more information.

Data Submission Form

Perform motif discovery, motif enrichment analysis and clustering on large nucleotide datasets.

Select the motif discovery and enrichment mode

Normal mode Discriminative mode [?](#)

Select the sequence alphabet

Use sequences with a standard alphabet or specify a custom alphabet.

DNA, RNA or Protein Custom [Choose File](#) No file chosen

Input the primary sequences

Enter the (equal-length) nucleotide sequences to be analyzed. [?](#)

Upload sequences [Choose File](#) YAP1_500bp.fa [DNA](#) [?](#)

Input the motifs

Select, upload or enter a set of known motifs. [?](#)

Eukaryote DNA [DNA](#) [?](#)
 Vertebrates (In vivo and in silico) [?](#)

Input job details

(Optional) Enter your email address. [?](#)

(Optional) Enter a job description. [?](#)

► Universal options
 ► MEME options
 ► DREME options
 ► CentriMo options

I will just keep the defaults for all the options. It takes a while to finish.

Output from **MEME-ChIP**:

YAP1 is not a DNA binding transcription factor itself, rather it is in a complex with TEAD, which is a DNA binding transcription factor. The first motif is a TEAD binding motif in tandem, and the second motif is AP-1 binding motif.

One downside of **MEME-ChIP** is that you can only input the DNA sequences

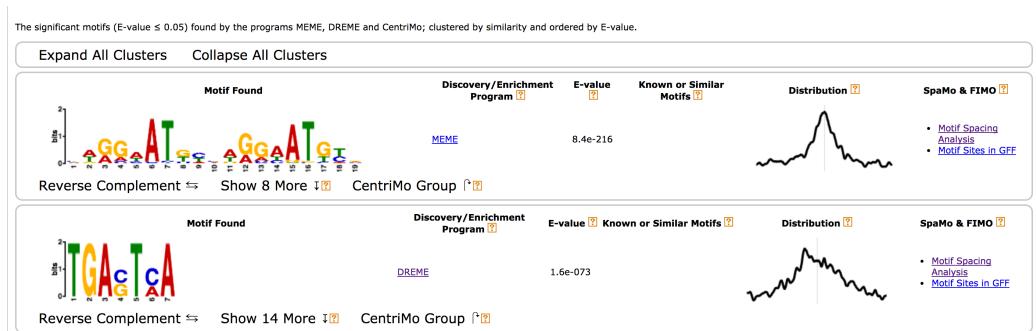


Figure 113.2

with the same length. Instead, you can use Homer `findMotifsGenome.pl`¹⁷ with the length of the peaks for finding motifs:

```
findMotifsGenome.pl YAP1_filtered_peaks.bed hg19 YAP1_motifs -size given
```

Rank	Motif	Name	P-value	log P-value	q-value (Benjamini)	# Target Sequences with Motif	% of Targets Sequences with Motif
1	CCAGGAATGC	TEAD4(TEA)/Tropoblast-Tead4-ChIP-Seq(GSE37350)/Homer	1e-401	-9.255e+02	0.0000	1430.0	66.54%
2	CCTAGGAATGC	TEAD(TEA)/Fibroblast-PU.1-ChIP-Seq(Unpublished)/Homer	1e-350	-8.066e+02	0.0000	1242.0	57.79%
3	CCAGGAATGT	TEAD2(TEA)/Py2T-Tead2-ChIP-Seq(GSE55709)/Homer	1e-348	-8.035e+02	0.0000	1116.0	51.93%
4	CGATGACTCATC	Fra1(bZIP)/HTS49-Fra1-ChIP-Seq(GSE46166)/Homer	1e-297	-6.841e+02	0.0000	957.0	44.53%
5	CGATGACTCATC	Atf3(bZIP)/GBM-ATF3-ChIP-Seq(GSE33912)/Homer	1e-287	-6.631e+02	0.0000	1037.0	48.26%
6	CGATGACTCAT	BATF(bZIP)/Th17-BATF-ChIP-Seq(GSE39756)/Homer	1e-284	-6.552e+02	0.0000	1022.0	47.56%
7	ATGACTCATC	AP-1(bZIP)/ThioMac-PU.1-ChIP-Seq(GSE21512)/Homer	1e-269	-6.212e+02	0.0000	1083.0	50.40%

Figure 113.3

The Homer results are in consistence with the MEME-ChIP results.

¹⁷<http://homer.salk.edu/homer/ngs/peakMotifs.html>

113.5 How to call differential peaks?

One of the other frequent questions biologists ask is how the peaks change in different conditions. e.g. different treatment of the cells, different subtypes of the same cancer. To answer this question, you'd better have biological replicates of each condition.

There is a review paper on this topic: A comprehensive comparison of tools for differential ChIP-seq analysis¹⁸. You can choose tools based on various rules:

12 | Steinhauser et al.

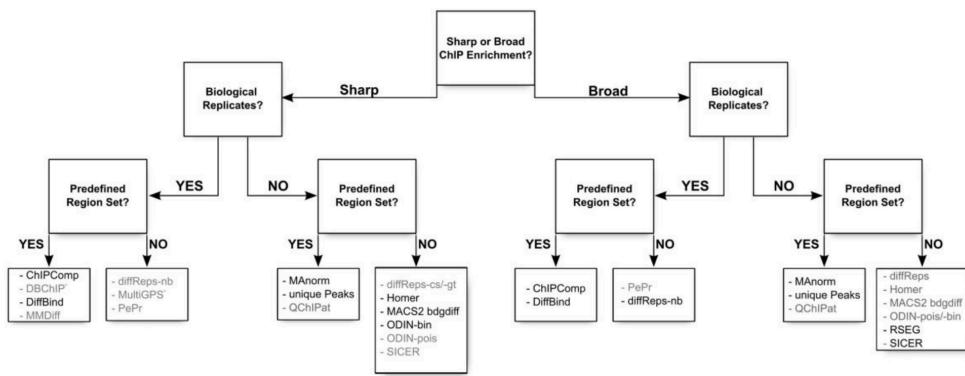


Figure 7. Decision tree indicating the proper choice of tool depending on the data set: shape of the signal (sharp peaks or broad enrichments), presence of replicates and presence of an external set of regions of interest. We have indicated in dark the name of the tools that give good results using default settings, and in gray the tools that would require parameter tuning to achieve optimal results: some tools suffer from an excessive number of DR (PePr, ODIN-pois), an insufficient number of DR (QChIPat, MMDiff, DBChIP) or from an imprecise definition of the DR for sharp signal (SICER, diffReps-nb). *MultiGPS has been explicitly developed for transcription factor ChIP-seq.

Figure 113.4

¹⁸<http://bib.oxfordjournals.org/content/early/2016/01/12/bib.bbv110.short?rss=1>

Chapter 114

ChIP-Seq Downstream Analysis 2

The author of this guide is Ming Tang¹. The material was developed for the Biostar Handbook.

- GitHub profile²
- Diving into Genetics and Genomics³

114.1 How do I generate a heatmap with ChIP-seq data?

Heatmap is of no mystery. It is a way to visualize the data a.k.a. using colors to represent values. However, one really needs to understand the details of heatmaps. I recommend you to read Mapping quantitative data to color⁴ and Heat maps⁵ from a series of articles from *Nature Methods*.

Usually one has a matrix and then plot the matrix using R functions such

¹<https://github.com/crazyhottommy>

²<https://github.com/crazyhottommy>

³<http://crazyhottommy.blogspot.hu/>

⁴<http://www.nature.com/nmeth/journal/v9/n8/full/nmeth.2134.html>

⁵<http://www.nature.com/nmeth/journal/v9/n3/full/nmeth.1902.html>

as `heatmap.2`⁶, `pheatmap`⁷ or `Heatmap`⁸. With those R packages, it is very easy to make heatmaps, but you do need to read the documentations of the packages carefully to understand the details of the arguments. Read a tale of two heatmap functions⁹.

For ChIP-seq data, one wants to plot ChIP-seq signal around the genomic features(promoters, CpG islands, enhancers, and gene bodies). To do this, you need to first take the regions of say 5kb upstream and downstream of all (around 30000) the transcription start sites (TSS), and then bin each 10kb region to 100 bins (100 bp per bin). Count the ChIP-seq signal (reads number) in each bin. Now, you have a data matrix of 30000 (promoters) x 100 (bin), and you can plot the matrix using a heatmap function mentioned above.

You can of course do it from scratch, but one of the taboos of bioinformatics is re-inventing the wheel. Most likely, someone has written a package for this type of analysis. Indeed, check this biostar post¹⁰ for all the available tools. Choose the right tool for yourself, if you are not that familiar with R, you can use some GUI tools. `EaSeq`¹¹ is a pretty powerful tool for windows users.

I personally like `EnrichmentHeatmap`¹² by Zuguang Gu the most because it gives me the most flexibility to control the looks of my heatmap. It is based on `ComplexHeatmap`¹³, which I highly recommend you to learn how to use. Below, I will walk you through how to make a heatmap with the `EnrichmentHeatmap` package.

First, let's read in the data:

```
library(EnrichedHeatmap) # for making heatmap
library(rtracklayer) # for reading bigwig and bed files
```

⁶<https://cran.r-project.org/web/packages/gplots/index.html>

⁷<https://cran.r-project.org/web/packages/pheatmap/index.html>

⁸<https://github.com/jokergoo/ComplexHeatmap>

⁹<https://rpubs.com/crazyhotommy/a-tale-of-two-heatmap-functions>

¹⁰<https://www.biostars.org/p/180314/>

¹¹<http://easeq.net/>

¹²<https://bioconductor.org/packages/release/bioc/html/EnrichedHeatmap.html>

¹³<https://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html>

```

library(GenomicRanges)
# read in the bed peak files to GRanges object
H3K27ac.bed<- import("~/ChIP-seq/results/peaks/H3K27ac_peaks.bed", format = "BED")
YAP1.bed<- import("~/ChIP-seq/results/peaks/YAP1_peaks.bed", format = "BED")

# read in bigwig files to GRanges object, it can be slow. bigwig is several times faster
# you can use which argument to restrict the data in certain regions.
H3K27ac.bw<- import("~/ChIP-seq/results/bams/H3K27ac.bw", format = "BigWig")
YAP1.bw<- import("~/ChIP-seq/results/YAP1.bw", format = "BigWig")

```

We want to plot the H3K27ac signal and YAP1 signal 5kb flanking the center of YAP1 peaks.

```

YAP1.10kb<- resize(YAP1.bed, width = 10000, fix = "center")

# take only the center of the peaks
YAP1.10kb.center<- resize(YAP1.10kb, width = 1, fix = "center")

YAP1.mat<- normalizeToMatrix(YAP1.bw, YAP1.10kb.center, value_column = "score",
                               mean_mode="w0", w=100, extend = 5000)

H3K27ac.mat<- normalizeToMatrix(H3K27ac.bw, YAP1.10kb.center, value_column = "score",
                                   mean_mode="w0", w=100, extend = 5000)

## check data range
quantile(YAP1.mat, probs = c(0.005, 0.5, 0.90))
quantile(H3K27ac.mat, probs = c(0.005, 0.5, 0.90))

## mapping colors
library(circlize)

## from the quantile, I choose the color mapping range
col_fun_YAP<- circlize::colorRamp2(c(0, 20), c("white", "red"))
col_fun_H3K27ac<- circlize::colorRamp2(c(0, 100), c("white", "red"))

```

```

EnrichedHeatmap(YAP1.mat, axis_name_rot = 0, name = "YAP1",
                 column_title = "YAP1", use_raster = TRUE, col = col_fun_YAP,
                 top_annotation = HeatmapAnnotation(lines = anno_enriched()),
                 top_annotation_height = unit(2, "cm")) +
EnrichedHeatmap(H3K27ac.mat, axis_name_rot = 0, name = "H3K27ac",
                 column_title = "H3K27ac", use_raster = TRUE, col = col_fun_H3K27ac,
                 top_annotation = HeatmapAnnotation(lines = anno_enriched()),
                 top_annotation_height = unit(2, "cm"))

```

114.2 How do I generate a meta profile plot with ChIP-seq data?

Although we used `HeatmapAnnotation(lines = anno_enriched())` to add a line plot on top of the heatmap, it may not easy to put the two line graph together. It is very common to compare signal strength for the same ChIPed factor in different conditions (treatment vs control, tumor vs normal etc). How can we do it?

The `YAP1_mat` and `H3K27ac_mat` are just like regular matrix, you can use `colMeans` to get the average and plot the average in the same figure with `ggplot2`. We can even add 95% confidence intervals onto the graph.

```

YAP1_mean<- data.frame(avg = colMeans(YAP1.mat),
                         CI_lower = apply(YAP1.mat, 2, Hmisc::smean.cl.normal)[2,],
                         CI_upper = apply(YAP1.mat, 2, Hmisc::smean.cl.normal)[3,])
                         mutate(factor = "YAP1", pos = colnames(YAP1.mat))

H3K27ac_mean<- data.frame(avg = colMeans(H3K27ac.mat),
                            CI_lower = apply(H3K27ac.mat, 2, Hmisc::smean.cl.normal)[2,],
                            CI_upper = apply(H3K27ac.mat, 2, Hmisc::smean.cl.normal)[3,])
                            mutate(factor = "H3K27ac", pos = colnames(H3K27ac.mat))

library(tidyverse)
combine_both<- bind_rows(YAP1_mean, H3K27ac_mean)

```

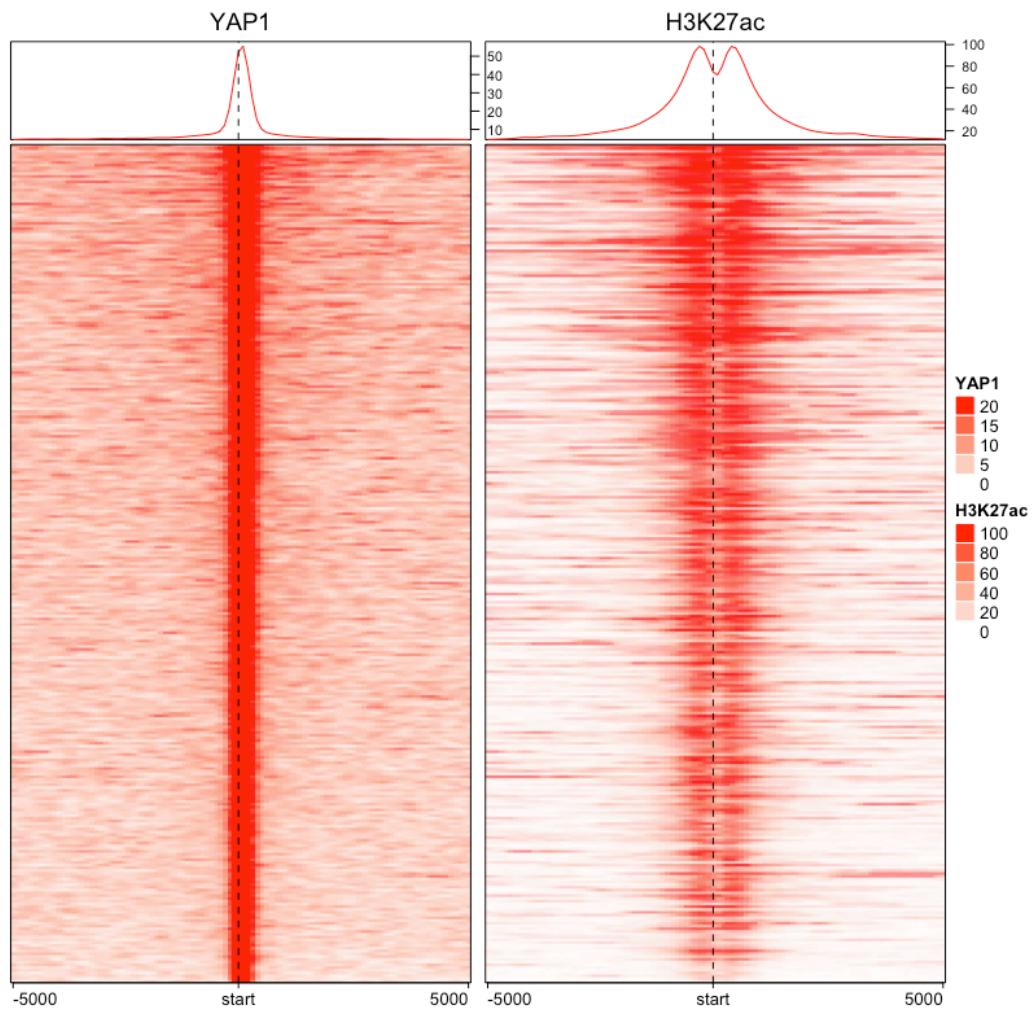


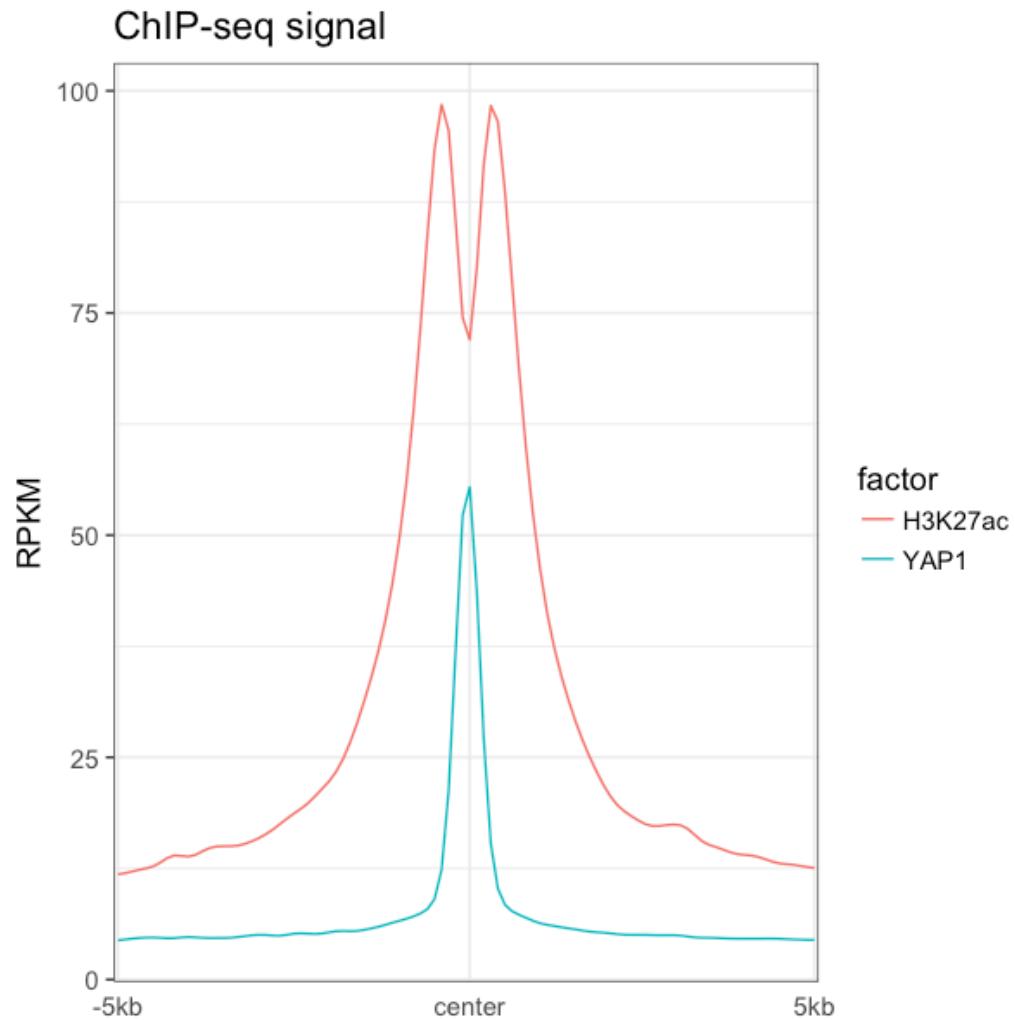
Figure 114.1

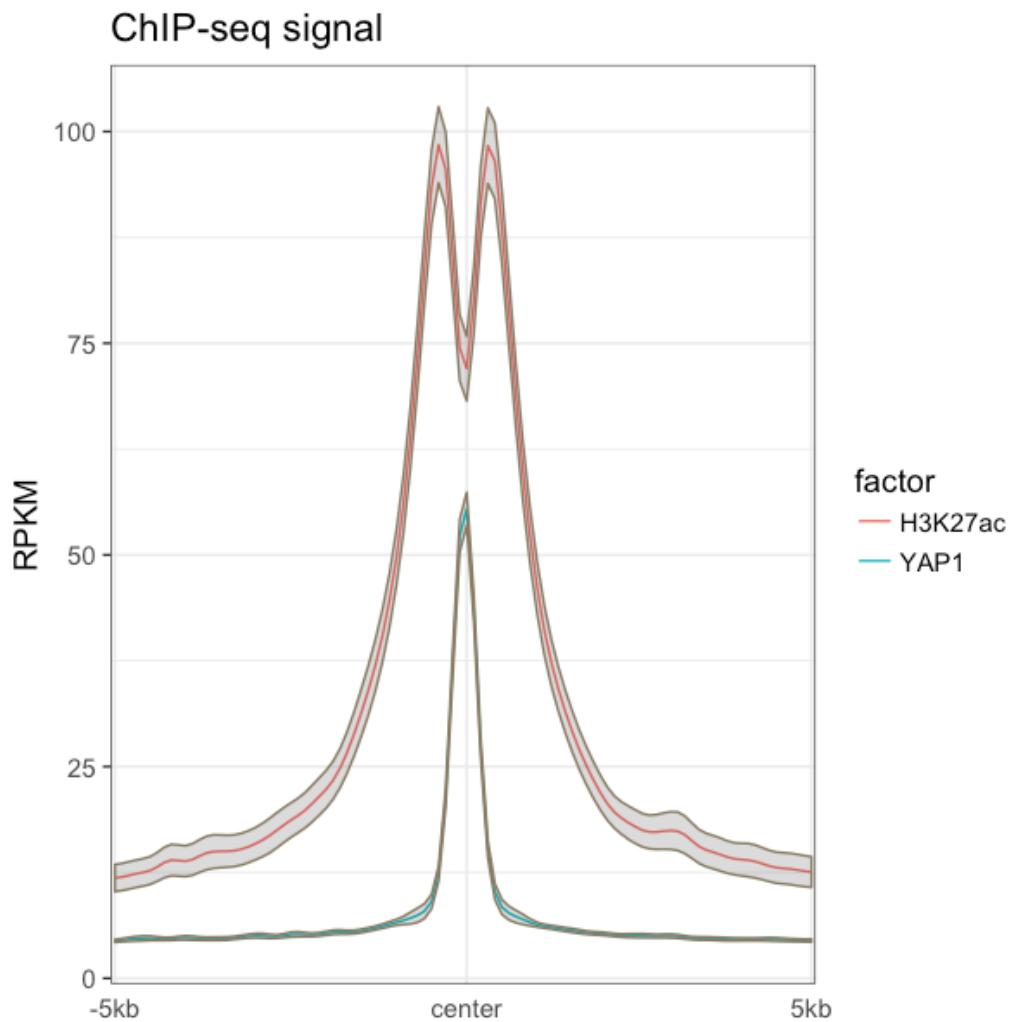
```
## change position to factor and order it
combine_both$pos<- factor(combine_both$pos, levels= YAP1_mean$pos)

## without confidence interval

ggplot(combine_both, aes(x = pos,y = avg, group = factor)) + geom_line(aes(color = fa
  theme_bw(base_size = 14) +
  theme(axis.ticks.x = element_blank()) +
  scale_x_discrete(breaks = c("u1", "d1", "d50"), labels =c ("‐5kb", "center",
  xlab(NULL) +
  ylab("RPKM")+
  ggttitle("ChIP-seq signal")

## take some touch up to finalize the figure
ggplot(combine_both, aes(x = pos,y = avg, group = factor)) + geom_line(aes(color = fa
  geom_ribbon(aes(ymin= CI_lower, ymax=CI_upper), alpha=0.2, col = "#8B7E66") +
  theme_bw(base_size = 14) +
  theme(axis.ticks.x = element_blank()) +
  scale_x_discrete(breaks = c("u1", "d1", "d50"), labels =c ("‐5kb", "center",
  xlab(NULL) +
  ylab("RPKM")+
  ggttitle("ChIP-seq signal")
```





As you can see, when you really know the power of R programming language, you do much more!

114.3 Where can I get public available ChIP-seq data sets?

114.3.1 ENCODE

It is not complete for a ChIP-seq chapter without mentioning the ENCODE project¹⁴. Many ChIP-seq data sets from various cell types can be found there. In addition, ENCODE has many other data sets such as ChIA-PET, Hi-C and DNase-seq.

114.3.2 UCSC genome browser

Every biologist need to know how to use UCSC genome browser¹⁵! It is one of the most popular web genome browsers.

If you scroll to the Regulation tracks (hg19 version). There are many ChIP-

seq data available for you to browse.

click ENC TF Binding:

click HAIB TFBS:

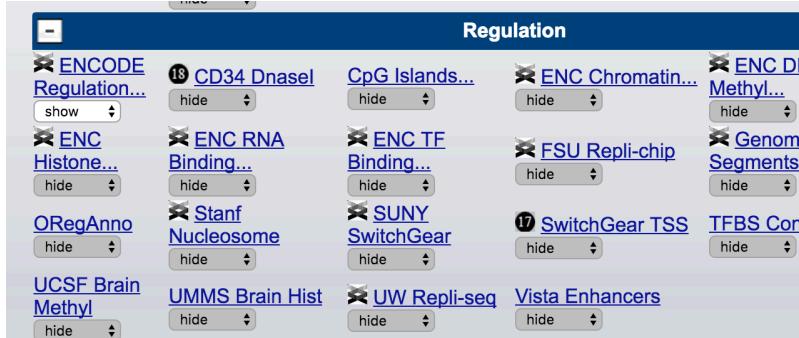
Now you can check various boxes to make it show up in the browser. quite useful! Many of the data sets are from ENCODE.

I recommend you to watch the tutorials from openhelix¹⁶ to take full advantage of this great resource.

¹⁴<https://www.encodeproject.org/>

¹⁵<https://genome.ucsc.edu/>

¹⁶<http://www.openhelix.com/ucsc>



ENC TF Binding Super-track Settings



ENCODE Transcription Factor Binding Tracks ([All Regulation tracks](#))

Display mode: [hide](#) [Submit](#)

[dense](#) [Uniform TFBS](#) Transcription Factor ChIP-seq Uniform Peaks from ENCODE/Analysis *ENCODE March 2012 Freeze*

[hide](#) [HAIB TFBS](#) Transcription Factor Binding Sites by ChIP-seq from ENCODE/HAIB

[hide](#) [SYDH TFBS](#) Transcription Factor Binding Sites by ChIP-seq from ENCODE/Stanford/Yale/USC/Harvard

[hide](#) [UChicago TFBS](#) Transcription Factor Binding Sites by Epitope-Tag from ENCODE/UChicago

[hide](#) [UTA TFBS](#) Open Chromatin TFBS by ChIP-seq from ENCODE/Open Chrom(UT Austin) *ENCODE July 2011 Freeze*

[hide](#) [UW CTCF Binding](#) CTCF Binding Sites by ChIP-seq from ENCODE/University of Washington

Figure 114.2

Figure 114.3

114.3.3 Cistrome

Cistrome¹⁷ is a project maintained by Sheirly Liu's lab in Harvard. It has a lot more data sets including ENCODE and other public data sets from GEO¹⁸. Some features are very friendly to wet biologists.

114.3.4 GEO and ENA

Of course, if you want to adventure yourself with the raw data in a publication, you can always go to GEO¹⁹ to get the SRA file and convert to fastq. Alternatively, you can directly download fastqs from European Nucleotide Archive (ENA)²⁰.

¹⁷http://www.cistrome.org/Cistrome/Cistrome_Project.html

¹⁸<https://www.ncbi.nlm.nih.gov/gds/>

¹⁹<https://www.ncbi.nlm.nih.gov/gds/>

²⁰<http://www.ebi.ac.uk/ena/browse/read-download>

Part XXVI

Sequence Assembly

Chapter 115

Introduction to sequence assembly

115.1 What does “sequence assembly” mean?

Assembly is a “catch-all” term used to describe methods where we combine shorter individual measurements called “reads” into longer contiguous sequences typically called “contigs.”

ASSEMBLY is the process of turning shorter READS into longer CONTIGS.

Because the sequencing process works by “breaking” the original DNA into smaller fragments, the assembly process is conceptually similar to putting together an image puzzle from its many pieces.

The software that performs the assembly is called the “assembler.”

115.2 What applications does assembly have?

First, there is the “obvious” use of identifying the base composition of longer DNA segments. Since the majority of organisms have not yet been sequenced, thus there is plenty to do there.

Interestingly enough there is a resurgence of interest in using assembly methods to identify variants of already known genomes. Alignment based variation calling is inadequate whenever the length of the genomic changes is comparable to measurement lengths (read lengths). The reads spanning variations of this size often fail to align altogether - or even worse; they align in another location. By assembling these reads into longer contigs, you can identify more substantial differences in the genomes.

115.3 What are the challenges of assembly?

Sequence assembly is perhaps the application domain of bioinformatics where “skill” and “expertise” are the most difficult to identify and define.

As you will see, it is a field where the procedural method descriptions “*this is the command we used to assemble the genome*” hide the staggering complexity and challenges of finding the particular command reported to work well. Assemblers are quite unlike any other software tool you will ever use. Most come with a bewildering array of parameters - the purpose of which are not explained, yet many will have profound effects on the results that they produce.

Trial and error are one of the most commonly used strategies - you will have to keep tuning the parameters and rerun the entire process hoping that the results improve - sometimes in vain. As it turns out, genome assembly is the most computational demanding bioinformatics method of them all. Assembling a large genome may take even weeks(!) and substantial computational resources. Thus any expertise built on trial and error will have to be accumulated over a much more extended period.

Finally, even when assembly appears to work, almost always it will contain several severe and substantial errors. That is where, in our opinion, bioinformatics expertise matters more. The ability to understand, visualize and correct the mistakes of an assembly has a utility that will outlast the present and is more valuable than knowing the exact invocation of a tool by heart.

115.4 What is the “white whale” of assembly?

In reference to Moby Dick, the “white whale” is something someone chases and chases then becomes obsessed with, but in the end, they can never get it.

The white whale of assembly is obtaining “single linear genome.” What it means is that you hope that even when starting with the many millions of short reads you will end up with one, unique, unambiguous, contiguous long sequence that corresponds to the truth, the “reference” genome that other scientists will now refer to in their work.

The reason that this is a white whale is that there is no such thing as a single linear best, representative genome. The better we understand large-scale genomic variation, the more apparent it becomes that no single reference genome could characterize each member of the same species. Instead of a single genome, the new school of thought is to approach the problem in terms of a network of genomes, and within that framework identify the path that describes our genome in this much larger graph space. Hence our goal should be to identify the segments and connectors in this graph space and worry less about the concept of single linear “reference”. For a higher level overview we recommend the following article that describes the upcoming shift in the way we think about genomes:

- As DNA reveals its secrets, scientists are assembling a new picture of humanity¹

115.5 What can be “assembled”?

For individual organisms scientist typically perform a:

- Genome assembly: establish the DNA composition of the organism
- Transcriptome assembly: build the transcript (RNA) expression for the organism

Applications to metagenomics:

¹<https://www.statnews.com/2016/10/07/dna-genome-sequencing-new-maps/>

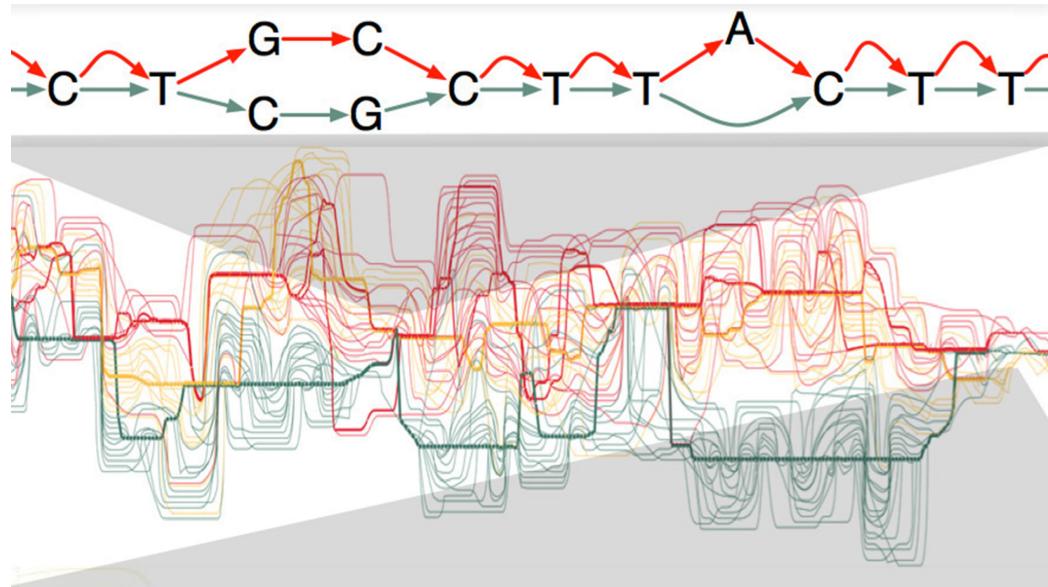


Figure 115.1

- Meta-genome assembly.
- Meta-transcriptome assembly.

Chapter 116

Concepts in sequence assembly

116.1 How do “assemblers” work?

The last decade has brought substantial innovation to this field, from so-called “overlap layout” algorithms that use multiple sequence alignments to k-mer and graph-based algorithms that attempt to find the shortest path that connects shorter subsequences.

At the same time, different instrumentation requires radically different approaches - it is unlikely that a tool designed for Illumina data would perform well on reads produced by a PacBio instrument.

116.2 What makes genome assembly difficult?

Since there are only four bases in the DNA and because genomes contain repetitive elements some regions may allow for many different yet seemingly equally good alternatives in which these shorter segments could be pieced back together. In essence, there could be multiple, similarly good solutions to the puzzle.

Assembly processes are more sensitive to the read lengths and the systematic errors of the sequencing process. Short reads may contain fewer identifiable regions. Systematic errors may accumulate and support spurious results.

Note: Assembly as an application domain of bioinformatics is immature. The success of an assembly depends on (potentially hidden) attributes of the data.

It is not uncommon for software to produce radically different outcomes when slightly modifying their settings and parameters. The algorithms require several levels of decision making, often with far-reaching consequences.

116.3 Beyond technical limitations what else makes genome assembly difficult?

Sequencing measurements originating from non-identical genomic sequences cause a variety of problems. Diploid organism already contains two slightly different copies of DNA. Various members of the same species may exhibit even more divergence. These difference make the job of an assembly program even harder as it tries to reconcile and consolidate these differences into a single consensus sequence. Also, certain regions produce much lower coverages than others leading to uneven coverage. Collecting more data may not be sufficient to address the systematic problems of this type.

116.4 How good is good enough?

Let us make something very clear from the beginning. No genome assembly is *entirely correct*. Even to this day, vast tracts of the human genome remain unknown; different genomic builds and releases are used to publish ever more accurate genomes of even model organisms.

Your goal is always to define “how good is good enough.” What quality of the assembly would you need to answer questions under study? That should be your driving motivation rather than producing a single, linear sequence that corresponds to a whole genome.

116.5 Is data quality control more important?

Data consistency has a more profound effect on assembly than for resequencing. When aligning against a reference, your aligner has additional information that can assist in making the right decisions. In contrast when assembling from scratch errors and mistakes compound and may lead to substantially lower performance.

116.6 Does the sequencing platform matter?

Assembly as a field of science got jump-started by the Illumina sequencers. Most of the published results, algorithms and evaluations refer to assembling data from Illumina instruments.

Long read technologies such as PacBio and MinION produce reads that are substantially longer. In a manner of speaking, they get a “head start” on assembly. At the same time since the error rate on these platforms are considerably higher, radically different algorithmic approaches are required to overlap and extend these measurements. “Classical” methods may not be appropriate at all.

116.7 What is a hybrid assembly?

A hybrid assembly is a process that uses data from different platforms and tries to make use of the strengths of each. For example, the Illumina platform provides higher coverage data whereas the PacBio platforms produce longer reads that span over longer intervals. Combining these data can significantly improve the assembly.

116.8 Does more data lead to better assembly?

While intuitively we may assume that more data gives better chances to find overlaps in practice more data does not guarantee better assembly. The coverage of the genomes will eventually yield diminishing returns.

In general assembly, processes will perform better with long and paired-end reads. The data quality (having DNA from a single genome) and genome' complexity have the most pronounced effects. The less complex (more unique) a region of a genome is the more difficult will be to assemble it. We mentioned before how “complexity” is used backward in bioinformatics. Scientists in this field call repetitive and redundant information as being more complex ... go figure ... not the first and not last of the oddities.

116.9 What does a genome assembly rely on?

The current experimental procedures for genome assembly demand that some of our measurements (reads) overlap with each other. The assembler attempts to detect the overlaps and will try to piece and extend the sequences by overlapping the identical (or similar) regions.

For example, suppose that the following were your “sequencing reads”:

```
ATATCGAAGAA  
GGAAATACATTTATTA  
GTACAAACATA  
AGAATAAGGAAAG  
CATAGAAGGAGGAAA  
AGGAAAGATGGCATT
```

Then your assembler may find what it believes to be the most “optimal” way to overlap this data and may consider the following as the best arrangement:

ATATCGAAGAA	GTACAAACATA
AGAATAAGGAAAG	CATAGAAGGAGGAAA
AGGAAAGATGGCATT	GGAAATACATTTATTA

And from the overlaps above it forms two “contigs”:

116.10. WHY ARE REPETITIVE REGIONS CHALLENGING TO ASSEMBLE?927

ATATCGAAGAATAAGGAAAGATGGCATTAGTACAAACATAGAAGGAGGAAATACATTTATT

Thus the six reads have been assembled into two contigs. Essential to be aware of the following limitations:

1. Contigs may get assembled on different strands.
2. Contigs do not indicate the order in which they follow each other in the genome.

Then note that it instead of overlapping reads 1,4 and 6 our assembler could have overlapped just 1 and 6 like so:

ATATCGAAGAA
AGGAAAGATGGCATT

To produce a shorter but different contig:

ATATCGAAGAAAAGATGGCATT

This contig was built at the cost of discarding read 2. The series of choices that the “assembler” makes along the process will always have substantial implications on the final result.

116.10 Why are repetitive regions challenging to assemble?

Sequencing libraries are built by fragmenting the original DNA into smaller pieces. When regions are repetitive, it becomes difficult to tell if a repeated part comes from the same or a different location. If your reads looked like so:

TROLLOLLOOTROLLOL
LOOTROLLOLLOTROLLOLL
ROLLOLLOTROLLOLLO
OLLOLLOOTROLLOLLOTROLLO

Each of the following contigs below could be a valid assembly:

TROLLOLLOOTROLLOLLOOTROLO

or

ROLLOLLOOTROLLOLLOTROLLOLLOOTROLLOLLOOTROLLOLLO

or

OLLOLLOOTROLLLOLOOTROLLLOLOTROLLLOOTROLLLOOTROLLLO

Plus many other valid alternatives may exist. Note how the length of the read becomes essential in identifying the minimal “repeat” structure. The read length has to be comparable to the repeat length to be able to resolve the repeats reliably.

Chapter 117

Redo: GAGE A critical evaluation of genome assemblies and assembly algorithms.

We are interested in accessing the data for the study titled:

- GAGE: A critical evaluation of genome assemblies and assembly algorithms¹

The paper was published in 2012 in the Genome Research journal. Specifically, we will discuss the results for the sequence data deposited for the bacteria *Staphylococcus Aureus* known for its potential to cause various skin and soft tissue infections.

We do like this paper and do no doubt that the authors of it were (and probably still are) the most skilled “assemblers” on the planet. That being said we have to point out what we think are fundamental weaknesses of the whole approach. The whole paper suggests that assembly is a “procedural” task. They publish scripts with hard-coded and parameters that are not explained. With they imply that you could download these scripts, use the same parameters on your data to get results of comparable quality.

Alas, that is not how assembly works. The recipes that they come up with

¹<https://www.ncbi.nlm.nih.gov/pubmed/22147368>

Resource

GAGE: A critical evaluation of genome assemblies and assembly algorithms

Steven L. Salzberg,^{1,7} Adam M. Phillippy,² Aleksey Zimin,³ Daniela Puiu,¹ Tanja Magoc,¹ Sergey Koren,^{2,4} Todd J. Treangen,¹ Michael C. Schatz,⁵ Arthur L. Delcher,⁶ Michael Roberts,³ Guillaume Marçais,³ Mihai Pop,⁴ and James A. Yorke³

¹McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University School of Medicine, Baltimore, Maryland 21205, USA;

²National Biodefense Analysis and Countermeasures Center, Battelle National Biodefense Institute, Frederick, Maryland 21702, USA;

³Institute for Physical Sciences and Technology, University of Maryland, College Park, Maryland 20742, USA; ⁴Center for Bioinformatics and Computational Biology, University of Maryland, College Park, Maryland 20742, USA; ⁵Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, Cold Spring Harbor, New York 11724, USA; ⁶Institute for Genome Sciences, University of Maryland School of Medicine, Baltimore, Maryland 21201, USA

New sequencing technology has dramatically altered the landscape of whole-genome sequencing, allowing scientists to initiate numerous projects to decode the genomes of previously unsequenced organisms. The lowest-cost technology can generate deep coverage of most species, including mammals, in just a few days. The sequence data generated by one of these projects consist of millions or billions of short DNA sequences (reads) that range from 50 to 150 nt in length. These sequences must then be assembled de novo before most genome analyses can begin. Unfortunately, genome assembly remains a very difficult problem, made more difficult by shorter reads and unreliable long-range linking information. In this study, we evaluated several of the leading de novo assembly algorithms on four different short-read data sets, all generated by Illumina sequencers. Our results describe the relative performance of the different assemblers as well as other significant differences in assembly difficulty that appear to be inherent in the genomes themselves. Three overarching conclusions are apparent: first, that data quality, rather than the assembler itself, has a dramatic effect on the quality of an assembled genome; second, that the degree of contiguity of an assembly varies enormously among different assemblers and different genomes; and third, that the correctness of an assembly also varies widely and is not well correlated with statistics on contiguity. To enable others to replicate our results, all of our data and methods are freely available, as are all assemblers used in this study.

Figure 117.1

match the “data” - they have most likely obtained these parameters after a lengthy process that itself is not adequately discussed - the rationale for picking each parameter is not mentioned. We believe that hidden properties of their data are “baked into” the parameters themselves. We believe that it is unlikely that these same parameters would produce optimal or even acceptable results on your data. Moreover, we want to note how there are no scripts that would help readers properly evaluate and correct the assemblies - those steps are left mostly unexplained.

That being said we consider this paper as a recommended reading to anyone interested in deepening their understanding of this field.

117.1 Where is the supporting material?

Commendably the authors of the publication maintain a website to distribute their results:

- <http://gage.cbcn.umd.edu/index.html>

117.2 How does GAGE differ from the other assembly bake-offs?

Both the question and the answers are lifted right from their web page and are reproduced verbatim because we find them amusingly combative and “in-your-face”:

- “*GAGE is being run by assembly experts.* Our team has assembled hundreds of genomes and has written some of the leading genome assembly software. We have been evaluating assemblers for more than 10 years. All of the assemblies and the comparisons among them will be conducted by experts.”
- “*All natural ingredients.* GAGE will use FOUR different whole-genome shotgun data sets, all from recent sequencing projects. Assemblathon and dnGASP will both use simulated data. Who can say how simulated data relates to real results? We prefer the real thing.”
- “*Completely open protocols.* We will compare multiple genome assembly programs, and we will describe all the parameters used to run them. We will also explain all the steps we take in cleaning up the data (pre-processing) and scrubbing the output of the assembly programs (post-processing). All of our results will thus be reproducible by anyone with sufficient computing resources.”

117.3 How do I get the data for the paper?

As always any analysis reproducibility starts with obtaining the data itself. Helpfully the GAGE paper lists the accession numbers for the data. In this case, let’s focus on *Staphylococcus aureus* datasets listed as shown below:

Methods

Data for *S. aureus* were downloaded from the Sequence Read Archive (SRA) at NCBI, accession numbers SRX007714 and SRX016063. The *R. sphaeroides* data have SRA accessions SRX033397 and SRX016063. The SRA libraries downloaded had higher coverage than was needed for most experiments. Each library was therefore randomly sampled to create a data set with 45× genome coverage, giving a total of 90× coverage for each genome.

Figure 117.2

Except the information above is not correct! Oh yes, the curse of reproducibility strikes again.

There is a typo in the paper. The number SRX016063 was entered for two entries, for two different organisms! See it for yourself:

Methods

Data for *S. aureus* were downloaded from the Sequence Read Archive (SRA) at NCBI, accession numbers SRX007714 and SRX016063. The *R. sphaeroides* data have SRA accessions SRX033397 and SRX016063. The SRA libraries downloaded had higher coverage than was needed for most experiments. Each library was therefore randomly sampled to create a data set with 45× genome coverage, giving a total of 90× coverage for each genome.

Figure 117.3

Typos are nothing new - what is unique here is that the paper offers no other indication as to what the “correct” accession number is. Hence from the publication alone, it is impossible to identify what data has been analyzed!

Well, well, one more indication (if you needed more) on just how complicated data reproducibility is. Even authors who are acutely aware of its importance, who make a concerted effort to publish everything that they do and create detailed descriptions of their methods may commit errors that render the

entire process *impossible from the very beginning*.

If anything this is a flaw of the old publishing process itself. Essential numbers should not need to be manually be embedded in a paper. Data should be cross-referenced, labeled automatically to make errors like this standout.

Did we find the correct SRR number? In this case, yes. Thanks to the author's foresight to create another, standalone site to distribute their data we were able to figure the accession number eventually — yet another example of how bioinformatics requires problem-solving of unexpected nature.

Visit the GAGE data site <http://gage.cbcn.umd.edu/data/>

```
#  
# Download the short jump file:  
wget http://gage.cbcn.umd.edu/data/Staphylococcus_aureus/Data.original/shortjump_1.fastq.gz  
  
# Look at the first read name  
gzcat shortjump_1.fastq.gz | head -1
```

It will print:

```
@SRR022865.7
```

From this SRR run number SRR022865 we can find the real experiment id SRX007711:

- <https://www.ncbi.nlm.nih.gov/sra/SRX007711>

Finally, we find that:

- <https://www.ncbi.nlm.nih.gov/sra/?term=SRX007714> that corresponds to run id SRR022868
- <https://www.ncbi.nlm.nih.gov/sra/?term=SRX016063> that corresponds to run id SRR034528
- <https://www.ncbi.nlm.nih.gov/sra/?term=SRX033397> that corresponds to run id SRR081522
- <https://www.ncbi.nlm.nih.gov/sra/?term=SRX007711> that corresponds to run id SRR022865

Chapter 118

Whole genome assembly

118.1 What are the steps of Genome Assembly?

The typical genome assembly process is composed of three different stages:

1. **Assembling contigs:** Merging reads into longer contigs.
2. **Scaffolding the contigs:** Spatially orienting contigs relative to one another.
3. **Finishing the genome:** Bridging over the spaces between the spatially oriented contigs and connecting them.

While possible it is infrequent that a single assembly process would start with sequencing reads and would produce a finished genome. Perhaps if your coverage is high (over 50x), the data is of very high quality and the genome does not contain repetitive regions.

The steps above require increasing human intervention, especially when finishing genomes may need another laboratory protocol where specially targeted regions of the genome that are thought to be close to one another are re-sequenced with a low-throughput method that attempts to bridge the gaps.

118.2 How do we quantify assembly quality?

Assembly processes are challenging to evaluate because we lack simple terms to quantify when one outcome is superior to another. There is a shorthand notation called the 3Cs (CCC): contiguity, correctness, and completeness. Ideally, we expect assemblies to be:

1. Contiguity. Produce the longest possible contigs.
2. Correctness. Assemble contigs with few/no errors.
3. Completeness. Cover the entire original sequence and minimize missing regions.

Balancing these three requires trade-offs based on the requirement of the problem that is being solved. Expect to have to make these tradeoffs, and you should identify your priorities before you start the process: e.g. “I would rather be more accurate even if it means the assembly will be less complete or contiguous.”

Scientists working in the field have long been searching for various measures that would incorporate all the three Cs above. In practice measuring the first ‘C’ “contiguity” appears to be the dominant way to quantify assembly quality. And within that realm, a measure called the N50 statistic is used as the primary differentiator. The N50 statistic was adopted as one that most closely captures the quality of the first step of assembly: extending reads to form longer contigs.

118.3 What type of assembly errors will I get?

Chaff contigs that are comparable to a read size. These are indicative of dead ends in the assembly process.

Misjoins: joining two sequences that should not be together.

Coverage misses: regions of the genome that are not assembled at all.

Repeat compression: a common error that makes consecutive repeats appear as a single sequence. ZZZ ABC ABC ABC ZZZ may look like as ZZZ ABC ZZZ

118.4 What is the N50 statistic?

Even though the N50 statistic is the most commonly used measure of the quality of a genome assembly, it might come as a surprise that its meaning is not rigorously defined. As a matter of fact, no definition for N50 that we've seen appears to be entirely correct. Take a moment to reflect on that a bit. Like p-values, that I describe later - everyone is using p-values, but practically nobody is using the concepts quite right.

The authors of GAGE: A critical evaluation of genome assemblies and assembly algorithms¹ Genome Research (2012) states that:

The N50 value is the size of the smallest contig (or scaffold) such that 50% of the *genome* is contained in contigs of size N50 or larger.

Let us note first that "GAGE" stands for "Genome Assembly Gold-Standard Evaluation." Calling something "gold standard" from the start is a common and annoying habit of some scientists. In our (though admittedly limited) experience papers declaring themselves as gold standard from the beginning are rarely that.

Then in the definition above you might immediately raise the question of how would you know what the *genome* size is when you are assembling it for the first time?. Now while there are experimental and comparative methods to estimating genome sizes, these are a different domain of science.

Many other scientists use different designations N50 and NG50 where N50 refers to the sum of all contigs lengths of the assembly whereas NG50 relates to using the length of the genome (if known or estimated). When the assembly length is reasonably close to the genome length, the two measurements are very similar. But note that there are legitimate reasons (not necessarily errors) that may cause an assembly to end up much longer or much shorter than the target genome.

Confusingly, several alternatives and slightly different definitions of N50 exist. Sometimes the median of contig lengths is used. For example in the Assemblathon 1: A competitive assessment of de novo short read assembly

¹<https://www.ncbi.nlm.nih.gov/pubmed/22147368>

methods² Genome Research, 2011 the authors overcomplicate the definition and state:

The N50 of an assembly is a weighted median of the lengths of the sequences it contains, equal to the length of the longest sequence s , such that the sum of the lengths of sequences greater than or equal in length to s is greater than or equal to half the length of the genome being assembled.

Frankly, I don't understand this definition - especially when it comes to weighted medians. I guess the authors agreed with this sentiment since in the Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species³ Genome Research, 2013 the authors write:

N50 is calculated by summing all sequence lengths, starting with the longest, and observing the length that takes the sum length past 50% of the total assembly length.

This definition is almost right, except it does not explicitly state that the summing should proceed by sorted lengths. It is noteworthy to observe just how difficult 'b]r93rg2ris to even state the definition for N50 perhaps foreshadowing that this domain of application is far more complicated than bioinformaticians care to admit.

118.5 So what is the N50 statistic?

A simpler explanation of N50, (in our own interpretation, that of course might not be quite correct), starts by ordering contigs by length. Suppose we have 10 different contigs (designated by XXXXXX) and we ordered these by their decreasing sizes:

Contig	Length	Sum
XXXXXXXXXX	10	10
XXXXXXXXX	9	19

²<http://genome.cshlp.org/content/21/12/2224.full>

³<https://gigascience.biomedcentral.com/articles/10.1186/2047-217X-2-10>

XXXXXXX	8	27
XXXXXX	7	34
XXXXXX	6	40
XXXXX	5	45
XXXX	4	49
XXX	3	52
XX	2	54
X	1	55

The sum of these lengths starting with the longest is 55. Half of that is 27.5.

Go down on this list and add up the lengths to find the contig where the cumulative length exceeds this half value. When we hit contig number 7 we have $10 + 9 + 8 + 7 = 34$, this value is larger than 27.5 so it is at this point at least half of the genome is stored in contigs of size 7 or greater.

Our N50 is then 7. Another simple way to state this (following the wording of this blogpost⁴) is that

“At least half of the nucleotides in this assembly belong to contigs of size 8bp or longer.”

See how we gave some many definitions for N50. Pick the one you like :-)

118.6 What are the problems with N50?

One of the biggest problems using the N50 metric as the primary means of evaluating assembly quality is that it rewards “misjoins.” A “misjoin” is the error of concatenating separate contigs into a single segment. Such mistakes can happen when the evidence is insufficient, but the algorithm is tuned to be overly generous in accepting these pieces of evidence.

Also applying cutoffs can lead to odd situations when selecting contigs would lead us to be either well under or well over the 50% see the section below?

⁴<http://jermdemo.blogspot.com/2008/11/calculating-n50-from-velvet-output.html>

118.7 So what is your N50?

While building artificial corner cases to demonstrate a weakness of a method or measure is akin to building straw men we could not resist it this time around. We invite you to do the math below and with that demonstrate to yourself that you do indeed understand how the N50 computation works.

Imagine that you have a genome of a size 1 million bp with and you ran an assembler that created one large contig of the length of 500,000bp plus another 500,000 contigs of sizes of 1bp each.

Contig	Length
XXXXXXXXXXXXXXXXXXXX...XXXXXXXXXXXX	500,000
X	1
X	1
X	1
X	1
X	1
... 500,000 1bp long sequences ...	

What is your N50?

Now take the same example and make your largest contig just a single base-pair shorter 499,999bp:

Contig	Length
XXXXXXXXXXXXXXXXXXXX...XXXXXXXXXXXX	499,999
X	1
X	1
X	1
X	1
X	1
... 500,000 1bp long sequences ...	

So what is your N50 now?

Well, what do you think?

Chapter 119

How to improve the assembly process

When you start out in the field of assembly you will note with surprise how much of the advice can be summarized as:

Keep running your assembler with many different parameters until you get a proper assembly.

There is a “handwaving,” and almost “unscientific” feel to the recommended processes. In general, should we ever tweak and run a tool until it gives you what you want? Besides, you will get somewhat ad-hoc guidance as to what a “good” assembly even means.

119.1 Why are assemblers so sensitive to parameters?

We believe that assemblers are sensitive to parameters because we as scientists are not quite sure when they work correctly - hence the rationale is to allow for extensive customization. Then, the information content and structure of sequences will vary to a great extent, both within a genome and more across species. This variation poses severe challenges to any algorithmic approach - as principles and optimizations that work well in one region will not

work for others.

Also, it is hard to identify with clarity how the assembler works internally and what decision it makes - hence it hard to foresee the effect of each parameter value on the result. Thus we are left with trial and error.

119.2 Should I error correct my data?

As you recall error correction means correcting for sequencing errors using the data itself. There are many logical explanations of why this should work. For example, error corrections will significantly reduce the number of k-mers in the data - and as such will make k-mer based algorithms job easier.

At the same time error correction is a new application domain that we have not had sufficient experience with. Later examples in this book may explore the effect of error correction and it, in general, it should be a choice that you evaluate yourself. Many authors, with experience, believe that error correction is a requirement.

119.3 What are the k-mer sizes?

Refer to the chapter title Sequence K-Mers¹ for a definition of k-mers. In nutshell, a k-mer are all possible subsequences of size k . The rationale for breaking our reads into even smaller pieces, **k-mers** is that we want to identify “correct” k-mers, those that originate from the real data. The assumption is that whatever errors the reads may have these are distributed randomly - hence will produce different erroneous k-mers. The correct k-mers will always be the most abundant.

In general, it is true that the longer a k-mer is, the fewer identical k-mers to it exist. At the same time it is also true the longer a k-mer is, the more likely is that it will contain an error. When it comes to assembly processes the rule of reasoning is that:

- A larger k value allows resolving more repetitions.
- A smaller k increases the chances of seeing a given k-mer.

¹<https://read.biostarhandbook.com/pattern/kmers.html>

Hence the selection of a **k-mer** is a tradeoff between longer repeats and more reliable measures. In general, you should assemble sequences using the largest k-mer size possible, such that the k-mer coverage is sufficient.

You may estimate these via trial and error (as stated above) but there are also tools like **kmergenie** that will assist you in determining the most likely k-mer sizes and coverage cutoffs. These tools sometimes work well, other times no so much.

119.4 How to tune the expected coverages?

From experience, we found that parameters indicating the “expected coverage” have a pronounce and substantial effect on the resulting assembly. This parameter may take different names for different tools:

`-abundance-min`
`-cov_cutoff`

each indicates the minimal number of observations necessary to “trust” that a k-mer sequence is correct. Again it is never clear what the value should be from the start. Some tools allow for an automatic inference.

Chapter 120

The lucky bioinformatician’s guide to genome assembly

Assembling a genome is a challenging process. Except when you get lucky. Then “it just works”. But even then you need a little more than luck. You have to make your own luck.

In this section, we will be using the Minia assembler¹ to demonstrate the typical usage patterns during an assembly process. We like `minia` because it is easy to run, is super fast and has other tools that integrate with it.

120.1 How do I run an assembly?

Genome assembly is an iterative process where you will:

1. Apply quality control measures
2. Estimate initial parameters (k-mers-sizes, coverages)
3. Run the assembly
4. Evaluate assembly (then go to 2 if necessary)

¹<http://minia.genouest.org/>

120.2 Is quality control necessary?

Yes. Absolutely. Every single guide will tell you so. In fact, quality control for assembly is a more complicated process. You will need to correct not just “visible” errors that the sequencing instrument marks with its quality values but even the invisible ones that need to be fixed with the most sophisticated methods the human mind has ever invented.

Unless you get lucky. In which case you don’t need to correct data at all. You “just carry on” and it will “just work”.

120.3 Do I need a lot of data?

Yes. Absolutely. Not only that, the consensus is that you will need the longest reads the instrument can generate, preferably 250bp or longer. Plus you’ll need paired-end reads that help scaffold the resulting contigs. Also, you better collect more data than you think you need.

Unless of course, you are lucky. In which case you may be able to get by using a small fraction of the sequenced data, say a hundred thousand, single end, short reads.

120.4 What does it look like when I am getting lucky?

Remember the study that we mentioned in the Redo: Genomic surveillance elucidates Ebola virus origin”

- Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak²

published in 2014 in the Science research journal. Let’s obtain a sequencing run from it. It has hundreds of sequencing runs, Hmmm, where to even start... well, I don’t know ... well let’s pick a run ... how about run id SRR1553425³:

²<http://www.sciencemag.org/content/early/2014/08/27/science.1259657.full>

³<https://www.ncbi.nlm.nih.gov/sra/?term=SRR1553425>

120.4. WHAT DOES IT LOOK LIKE WHEN I AM GETTING LUCKY?945

Get the data, it has 1.8 million paired-end reads (3.6 million total reads) but for now, for the sake of this example let's take a subset of 100,000 reads. In fact, we did not even expect it to work, we only want to demonstrate the process:

```
fastq-dump SRR1553425 -X 100000 --split-files
```

The data is paired-end, but for, and for the sake of simplicity we'll focus on first pairs, ignoring that we have extra information and even more data in the other pairs. Notably, when we are taking 100K reads from a single file, hence we are using just 3% (!) of the total sequencing data:

```
READS=SRR1553425_1.fastq
```

Off we go with the assembly. Let's run `minia` with no other parameters set:

```
minia -in $READS -out genome
```

First of all, hats off to Dr. Rayan Chikhi⁴. The process took only 5 seconds on a iMac! It is mind boggling that it only takes 5 seconds to build an internal graph structure out of millions of k-mers, that were spread over 10 million basepairs of 100,000 reads, then to churn through this network with a graph algorithm to find the shortest paths in them. We can unequivocally state that bioinformatics software developers are the best the world has ever seen!

The results of the process are stored in a file called `genome.contigs.fa` that we can run a quick stats on:

```
seqkit stat genome.contigs.fa
```

to produce:

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
genome.contigs.fa	FASTA	DNA	190	16,812	63	88.5	277

Okay, so we obtained:

- 190 contigs
- the average length is 88.5bp
- the longest is 270bp

This result seems useless, all *chaff*. Our read lengths were 101 long. How does one even “assemble” 101 long reads into an average of 88.5? ... Upon

⁴<http://cristal.univ-lille.fr/~chikhi/>

closer inspection, the numbers are more encouraging. See the `sum_len` field? It says 16,812. We do know that the Ebola genomes are of 18,000bp size.

Now consider that we started with 100,000 reads where each read was 101bp long. So the assembler was able to collapse 10,000,000 sequenced bases into a total length of 16,000! This assembly tells us that the algorithm has a good “sense” of what the individual pieces are, but it has a hard time connecting these parts. Things may not be all that grim.

120.5 How do I get “luckier”?

You saw how the assembly above looked “wrong” from contig length perspective but promising when evaluating the total genome length. This information tells you it might be worth poking around a little more, perhaps rerun the assembly with other parameters, such as different k-mers. When the assembly runs in seconds like the example above this is feasible. For now, we continue, hoping to get even more “lucky.”

The default k-mer size is 31 and we mentioned before how usually we want to raise this as high as possible, but still keeping sufficient coverages. Let’s go higher to 100.

```
minia -in $READS -out genome -kmer-size 100
```

Now the assembly reports a total assembly size of 233bp - still well off the mark.

When you are searching for a parameter range you shouldn’t change it small steps. First find the limits, then perform what is called a “binary-search” in between. Take the half between the explored intervals and chose the direction towards the better results. We saw how 31 seemed ok, but 100 was not. Half between is about 75

```
minia -in $READS -out genome -kmer-size 75
```

this improved our assembly quite a bit:

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
genome.contigs.fa	FASTA	DNA	21	19,896	156	947.4	3,539

We got 19 contigs, the longest is 3.5Kb. Sounds good. Let’s keep going higher with the k-mer sizes:

```
minia -in $READS -out genome -kmer-size 90
```

What? No way!

```
file      format type num_seqs sum_len min_len avg_len max_len
genome.contigs.fa FASTA DNA      1 18,820 18,820 18,820 18,820
```

Minia has now assembled our data into a single sequence. Do you recall that for `-kmer-size 100` the assembly was practically useless? And for `-kmer-size 90` we get complete genome in a single contig. Is this skill or luck?

120.6 How do I evaluate the assembly?

Let's align the assembled contig against the Zaire Ebola build of 1976.

```
# Make a folder for the references.
mkdir -p refs

# Accesion number for the 1976 outbreak.
ACC=AF086833

# This is the reference name.
REF=refs/${ACC}.fa

# Get the reference sequence.
efetch -db=nuccore -format=fasta -id=$ACC | seqret -filter -sid $ACC > $REF

# Index reference for the aligner.
bwa index $REF
```

120.7 What do I see in this assembly that I wouldn't otherwise?

Then align the assembled contigs against the 1976 strain:

```
CONTIGS=genome.contigs.fa
bwa mem $REF $CONTIGS | samtools sort > genome.bam
```

```
samtools index genome.bam
```

Visualizing the resulting `genome.bam` file in IGV we would see:

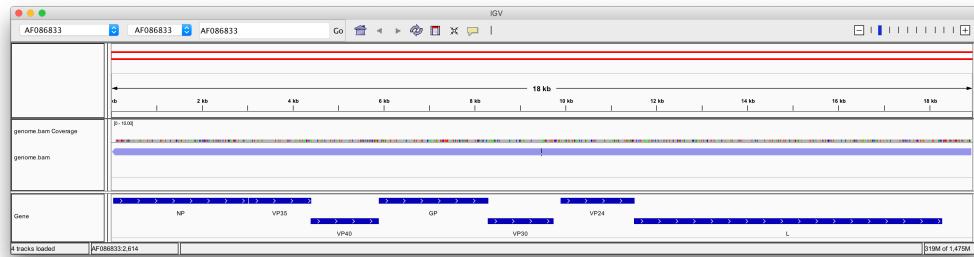


Figure 120.1

Note how the alignment is blue and not red. This alignment indicates that we have assembled the reverse strand! Since our data comes from a single stranded RNA virus that was transcribed into DNA for the purpose of sequencing you would need to reverse complement your assembly to find the real viral sequence.

```
cat genome.contigs.fa | seqkit seq -p -r > ebola.fa
```

Now align this reverse complemented sequence. It now aligns in the proper direction. When zoomed in you will see that genomic variants show up directly from the alignments and no SNP calling is necessary.

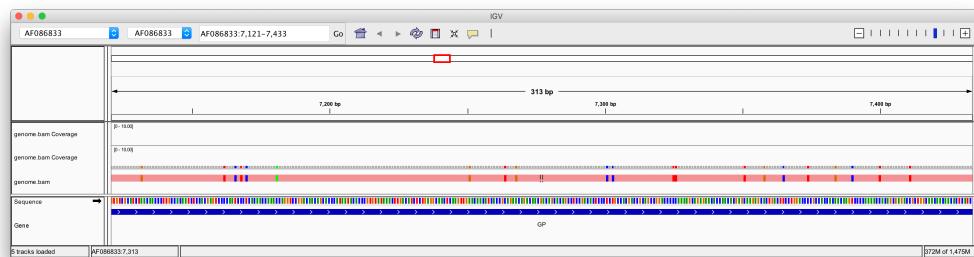


Figure 120.2

So there. You can produce and almost complete assembly and find all variants in an Ebola genome in four tries, totaling about 20 seconds. And you can do

that by using only 3% of the collected measurements, using paired end reads in single end mode, without any error correction. Well this is how good is to be lucky.

120.8 How do I get lucky?

The history of this chapter is rather amusing. We did not cherry pick this example out of the many hundreds that come with this publications. We randomly picked an SRR run and geared up to demonstrate the challenges and frustrations that one feels when attempting to assemble even a short and straightforward genome.

Much to our surprise the assembly “just worked.” It also helped crystallize in our mind a long-standing suspicion that we’ve always had:

Assembly is all about the data and not the method/protocol.

Clean and well-defined data can be easily assembled.

“Mosaic” data will give your troubles. What is “mosaic” data?

- data obtained from the superposition of multiple genomes
or
- data containing regions with high self-similarity

High quality, simple data does not need much filtering or effort.

Difficult data may be impossible to assemble. It is very unfair!

You get lucky by making your own luck. Choose the problem wisely, recognize problems early, then don’t blame yourself if the assembly does not work.

Chapter 121

How to perform a genome assembly

121.1 How do I run an assembly?

Genome assembly is an iterative process where you will:

1. Apply quality control measures
2. Estimate initial parameters (k-mers-sizes, coverages)
3. Run the assembly
4. Evaluate assembly (then go to 2 if necessary)

121.2 How to evaluate a genome assembly

The term “assembly evaluation” has two distinct and unrelated meanings:

1. Characterize the assembled contigs by their observed properties.
2. Characterize the assembled contigs relative to the “reality,” typically a known, similar genome.

The two definitions often get confounded into a single term.

121.3 What are the steps of genome assembly?

This section will work through genome assembly in the context of the assembly presented in the paper: GAGE: A critical evaluation of genome assemblies and assembly algorithms¹ Genome Research (2012). We start with the sequence data deposited for the bacteria *Staphylococcus aureus* known for its potential to cause various skin and soft tissue infections.

```
# Make a directory for the reference.  
mkdir -p refs  
  
# Accession number for the 1976 outbreak.  
ACC=NC_010079  
  
# This is the reference name.  
REF=refs/${ACC}.fa  
  
# Get the reference sequence.  
efetch -db=nuccore -format=fasta -id=$ACC | seqret -filter -sid $ACC > $REF
```

The data for a sequencing run for this bacteria is deposited in the Bioproject: PRJNA40073² The project describes 56 sequencing runs, out of which we start with SRR022868 as the GAGE paper does. This sequencing run produces reads of length 101.

SRA=SRR022868

The genome of this bacteria is known and deposited in GenBank under the accession number NC_010079. Its size is reported to be 2.8 million. Now as you recall the coverage C of a genome is:

$$C = N * L / G$$

Where N are the number of reads, L is the length of a read, and G is the size of the genome. As we are interested in the number of reads of length 101 that

$$N = C * G / L$$

¹<https://www.ncbi.nlm.nih.gov/pubmed/22147368>

²<https://www.ncbi.nlm.nih.gov/bioproject/PRJNA40073>

If we were aiming for say a 45x coverage, then for an L-101 we would need around 1.2 million reads. Since the data is paired-end, we need to select half as many reads. Let's obtain those data:

```
# Make a directory to hold the original data
mkdir -p data
```

We could extract the reads.

```
SRA=SRR022868
fastq-dump $SRA -X 600000 --split-files -O data
```

When a sequencing run is affected by many errors (as this one is, as we shall see) it is inadvisable to build a smaller dataset by taking the first N reads of the file, as we did in other examples. Errors may manifest themselves more systematically with respect of the physical layout of the flowcell. The order of the reads matches the flowcell; hence the first N reads are not a proper statistical sample. The proper way to randomly sample the data would be to get all the data:

```
SRA=SRR022868
fastq-dump $SRA --split-files -O data
```

Then randomly extract reads from each dataset.

```
seqkit sample -2 -n 600000 data/SRR022868_1.fastq > data/F1.fq
seqkit sample -2 -n 600000 data/SRR022868_2.fastq > data/F2.fq
```

Set up variables to make the commands easier to rerun on other input:

```
READ1=data/F1.fq
READ2=data/F2.fq
```

The paper states that the dataset has an insert size of 180 with a standard deviation of 20. If we did not know this, we could find out this information by aligning the sequences then computing the average template length. Below we filter (S=see the SAM filtering chapter³) for paired, correctly mapped reads on the forward strand.

```
bwa index $REF
bwa mem $REF $READ1 $READ2 | samtools view -F 4 -f 3 -F 16 | datamash mean 9
```

This command gives us 193 as the fragment (insert size).

³<https://read.biostarhandbook.com/sam/sam-filtering.html>

The FastQC report shows data of surprisingly deficient quality (Left is SRR022868; the right plot is for run SRR022865).

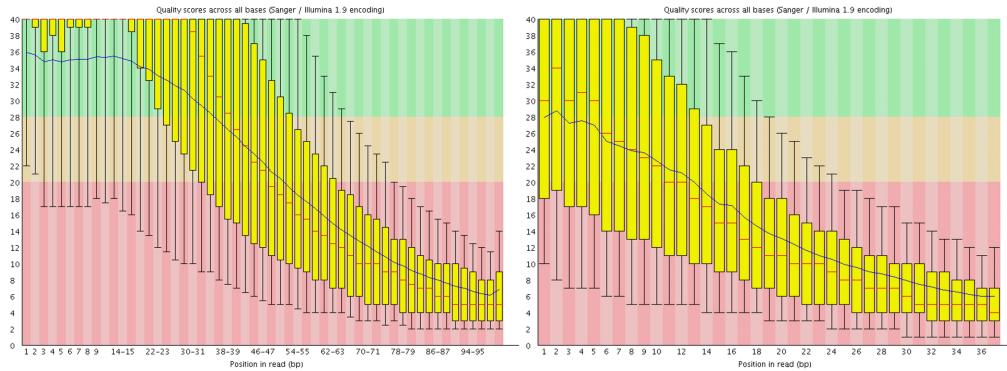


Figure 121.1

Looking at this data our first instinct is that the data is useless in reality that is not the case. As scientists, we are trained to trust measurements and instruments, and we give credibility to quality values. The more I learn about the sequencing instruments, the more I come to believe that most FASTQ sequencing qualities are wildly inaccurate. Moreover, the data above cannot be corrected in the “traditional” way. Try it, and you will see why. Filtering for say the average quality of 30 would remove just about all the data. It looks like a total loss. In fact, it is not.

First, let’s see what can we do if we took the data as is with if we were to put it through an easy to run assembler like Minia.

```
# Concatenate both paired file into one.
cat data/F1.fq data/F2.fq > data/minia.fq

# Assembly with Minia.
minia -in data/minia.fq -out minia
```

Minia completes in about 27 seconds but it is not a “pair-end aware” assembler, and with that, we are losing information. It does finish though and then again, just as in the previous example to total assembled genome size is surprisingly accurate.

121.4 How do I visualize a resulting assembly?

Thanks to the leadership of Pavel Pevzner⁴ The Center for Algorithmic Biotechnology⁵ St. Petersburg State University in Russia has become a powerhouse of assembly software. Among the many tools they have created, QUAST, Quality Assessment Tool for Genome Assemblies stands out as one that produces one of the most straightforward reports that characterize an assembly.

Using `quast` is quite straightforward; it requires a reference and a file with contigs:

```
quast -R $REF minia.contigs.fa
```

The result is an interactive web page that visualizes the assembly that you have created.

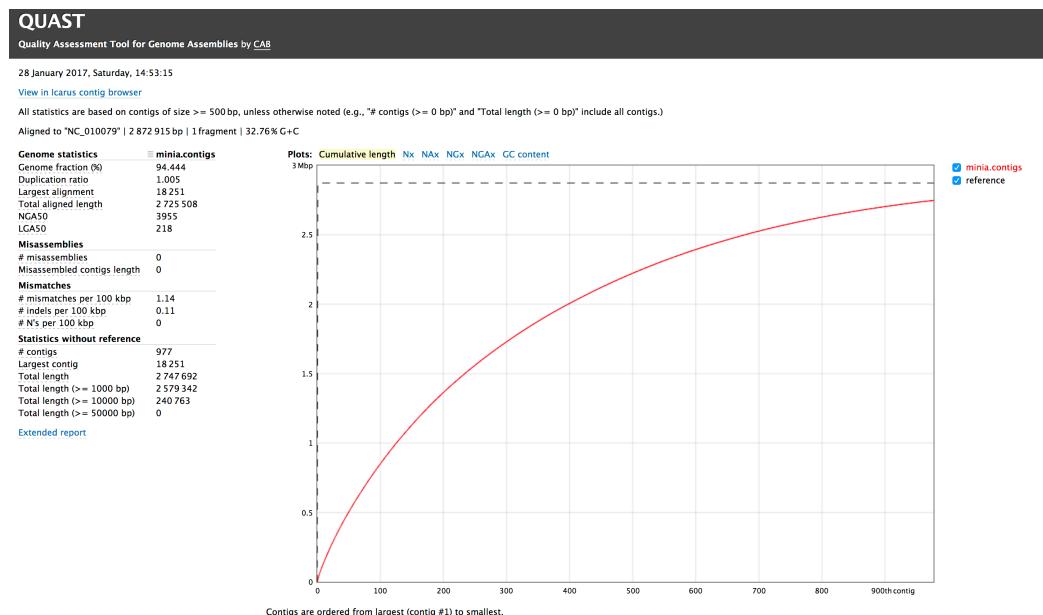


Figure 121.2

⁴<http://cseweb.ucsd.edu/~ppevzner/>

⁵<http://cab.spbu.ru/>

121.5 Can I improve the assembly?

Let's give a different assembler a chance. A perennial favorite, `velvet` is run in two steps with two subcommands. First, you will need to compute subsequences of a given k-mer (31) size and as shown below store them in a folder, called here `velvet31`:

```
velveth velvet31 31 -fastq -separate -shortPaired $READ1 $READ2
```

You can use the `velvetg` subcommand to assemble the genome from these precomputed data:

```
velvetg velvet31 -exp_cov auto -ins_length 190
```

Again the process is very speedy, taking no more than about 2 minutes. The resulting assembly will be located in the file `out31` called `contigs.fa`. We can now plot both assemblies on the same QUAST report.

```
quast -R $REF minia.contigs.fa velvet31/contigs.fa
```

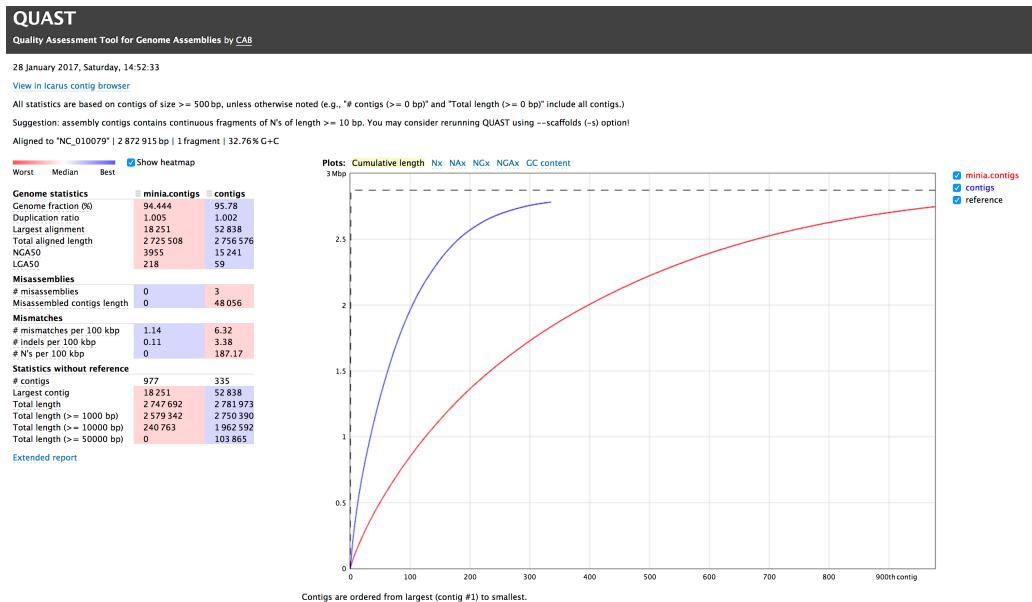


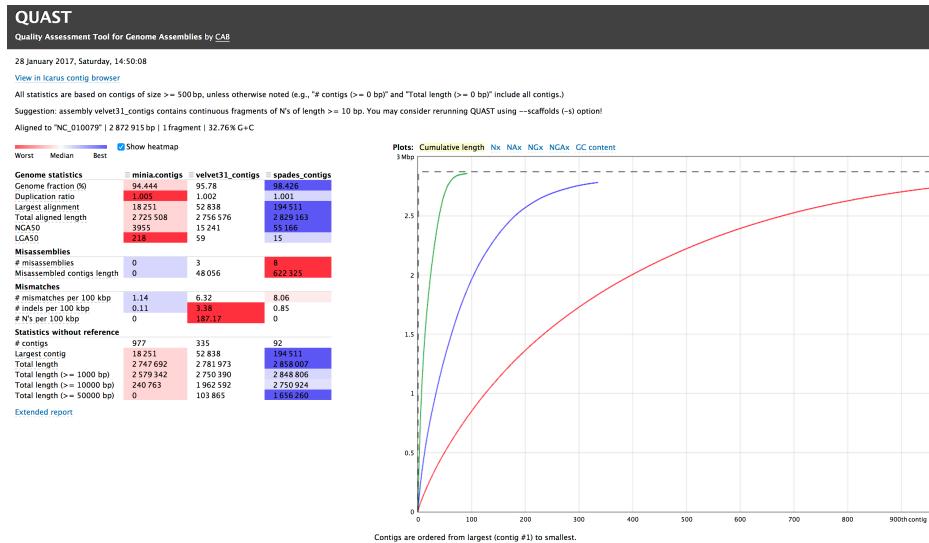
Figure 121.3

The resulting image shows the new assembly is shown as a blue line. Note how it rises much quicker to 100%; hence it contains longer segments.

121.6 What assembler should I use?

There are many choices you can choose from. See the GAGE paper for more options and inspiration if you need to. We like `velvet` because it is easy to run and allows you to set a baseline for your assembly. One of our other favorite assemblers is `spades`, another tool made at The Center for Algorithmic Biotechnology⁶.

```
spades.py -1 $READ1 -2 $READ2 -o spades
```



Over the years `spades` has

The assembly with `spades` appears the best by large, yet that information alone may not tell you the whole story. QUAST comes with the so-called Icarus contig browser, another visual aid that helps assess how well the assembly performs:

Better than any other explanation this browser shows you the tradeoffs. The Minia assembly is built out of many more pieces, and none are incorrect or in the wrong orientation, and it does cover the genome correctly. On the other end, the spades assembly contains the largest of the pieces - but it also includes misjoins, large misassembled regions. The velvet assembler produces output that is in between them both.

⁶<http://cab.spbu.ru/>

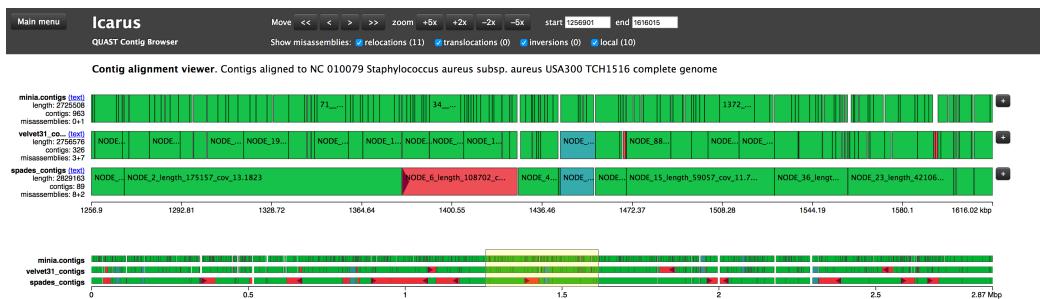


Figure 121.4

121.7 What do I do next?

At this point is where the “white whale” chase starts in earnest. It is easy to get excited and start tuning and running different methods. Progress is initially quick and fun, but then it can drag on. You can error correct, filter the data, make use of extra information, continuously evaluate the effects of each.

You may be able to use the reference genome to provide additional information to the assembler. Other data, preferably from a different sequencing platform helps even more.

Slowly you can build out better and better assemblies, stumble upon parameter setting, read up on other experiences, find and tune your results. It won’t be pretty, and it won’t be quick. There won’t be a recipe that you could make use of right away the next time around - but you will do the second assembly much better as you have learned what works and what does not for your data.

The GAGE paper lists many other choices and strategies.

Part XXVII

METAGENOMICS

Chapter 122

Introduction to metagenomics

Microbial communities cover the entire surface of Earth, including the inside of our bodies. Life of all forms coexists with and rely on a vast variety of microorganisms. Understanding the role, function and purpose of microbial species are essential to understanding higher order organisms.

Through environmental sequencing, scientists have come to realize that the level of functional and taxonomic diversity in the earth's microbial communities is much greater than previously anticipated.

Metagenomics analyses are more challenging than other sequencing data analysis because the skills required to succeed will swing wildly across the different domains of science even more abruptly than you might expect. Do you need to be a programmer, a biologist, a statistician, an ecologist, a microbiologist to succeed? Perhaps all of them at the same time. And while wearing many hats is not an unexpected requirement in bioinformatics where metagenomics will surprise you is that decision making is exceedingly subjective and choices made along the way have a pronounced impact on the results.

122.1 What is metagenomics?

Taking the definition from Wikipedia:

Metagenomics is the study of genetic material recovered directly from environmental samples. The broad field may also be referred to as environmental genomics, ecogenomics or community genomic

From a bioinformatician's point of view, metagenomics is the application of high-throughput genomic techniques to the study of communities of microbial organisms.

122.2 What are the challenges of metagenomics?

Traditionally genetics has focused on *vertical transfer* of genetic material: from parent to offspring. There is another mechanism of evolution, the so called *horizontal gene transfer* that allows some types of organisms to share genetic material *across* different species. Among single-celled organisms, it is perhaps the dominant form of genetic transfer.

Consequently, the variety and diversity of the genomic content of microorganisms is incomparably larger than that of higher order organisms, and cannot easily be modeled in the traditional top-down, across generation models. Also whereas in higher order organisms the phenotypes are well defined within a taxonomic level, bacteria from the same genera can manifest widely different functions.

122.3 Do humans host ten times as many bacteria as their cells?

Crack open any book on metagenomics there is a good chance that a statement of the following nature will greet you:

... humans have more bacterial cells inhabiting our body than our own cells ...

For example as seen in A Primer on Metagenomics, PLoS Comp Biology, 2010¹.

profoundly affected by microbes, from the scourges of human, farm animal, and crop pandemics, to the benefits in agriculture, food industry, and medicine to name a few. We humans have more bacterial cells (10^{14}) inhabiting our body than our own cells (10^{13}) [2],[3]. It has been stated that the key to understanding the human condition lies in understanding the human genome [4],[5]. But given our intimate relationship with microbes [6], researching the

Figure 122.1

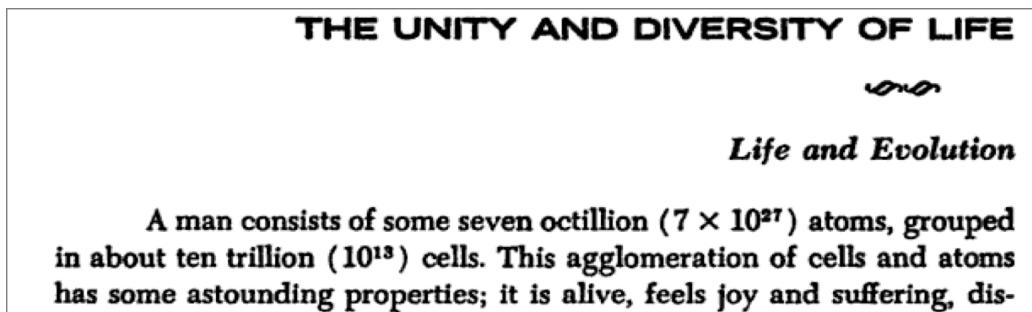
This claim often surfaces as a favorite introductory quote to presentations on metagenomics.

Amusingly, tracking down the origin of the numbers in this statement (10^{13} = trillion, 10^4 = ten trillion) is a mixture of amusement and incredulity. It appears that most scientists keep borrowing this statement from one another, repeating it without making any effort to ensure that it is true. Each paper cites another paper, stretching back to seemingly the beginning of modern biology. For example, the A Primer on Metagenomics, PLoS Comp Biology, 2010² cites a paper from 1996 in “Trends Microbiol” titled “The indigenous gastrointestinal microflora” as the origin of the statement above. But this cited article is not the source of this number. Instead, it repeats the claim and cites another source for from 1977 titled “Microbial ecology of the gastrointestinal tract.” But that paper is not the actual source either; it refers the reader to the origin of the number to another publication from 1972 in “American Journal of Clinical Nutrition” titled “Introduction to intestinal microecology.” Unsurprisingly this paper, in turn, cites a book from 1971 titled “Genetics of the evolutionary process” by Theodosius Grigorievich Dobzhansky. It is in the introduction of that book, in the very first sentence actually where Dobzhansky states that the human body contains ten trillion cells. There is no indication as to where he got that number from, this estimate (an opinion really) has been cited as a fact ever since.

Jordan Anaya has series of blog posts that demonstrate the absurdity of this scientific “habit” that is unfortunately not as rare as one might hope. He

¹<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000667>

²<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000667>

**Figure 122.2**

calls the practice russia-n-doll citations³ that often, like above culminate in a so called “rick-roll” citation⁴. The origin for the estimate of the bacterial numbers is similarly lengthy and equally unreliable. A more recent estimate states that Body’s bacteria don’t outnumber human cells so much after all⁵

Why even mention this here? It is an example and a reminder that you will need to approach every statement and discovery that has to do with metagenomics with a very healthy dose of skepticism. Yes, the field is fascinating, yes the field has a lot of potential - but at the same time it is also a domain that is rife with incorrect assumptions, flawed analyses, hand-waving arguments, grandiose claims, intriguingly sounding yet meaningless “discoveries.”

122.4 What type of answers does metagenomics produce?

Metagenomics is the science of answering questions that fall into the following categories:

1. Which microorganisms are present in a sample?
2. What abundance (quantity) is each microorganism present in?
3. How do the observed abundances change across conditions?
4. What are the functions of the microorganisms that are observed?

³<https://medium.com/@0mnesRes/rickrolls-and-russian-dolls-the-world-of-mindless-citing-fda5d0b0a5>

⁴<https://medium.com/@0mnesRes/rickrolls-and-russian-dolls-the-world-of-mindless-citing-fda5d0b0a5>

⁵<https://www.sciencenews.org/article/body%20%99s-bacteria-don%20%99t-outnumber-human-cells-so-much-after-all>

Important: It is essential to recognize from the very beginning that most microorganisms cannot exist independently and their survival requires the presence of diverse communities of other microbes. Thus we need to keep in mind that organisms need to be treated in the context of their communities rather than individual species

122.5 What are the metagenomics approaches?

Current practices in metagenomics fall into the categories of:

1. 16S rRNA sequencing
2. Whole-metagenome sequencing

122.6 What is 16S sequencing?

From the Wikipedia page on 16S Sequencing⁶

16S ribosomal RNA (or 16S rRNA) is the component of the 30S small subunit of a prokaryotic ribosome [1]. The genes coding for it are referred to as 16S rRNA gene and are used in reconstructing phylogenies, due to the slow rates of evolution of this region of the gene.

In a nutshell the 16S gene forms a peculiar part of a genome that has both very highly conserved regions (identical across just about all bacteria) as well as highly variable regions (hypervariable) that may be specific to a taxonomical level such as genus or species. You can use the conserved regions to isolate the variable regions that may be used to identify the bacteria. Since only a tiny section of the entire bacteria is genome will be sequenced, the 16S method is very “economical” requires a fraction of the coverage and produces far lower quantities of data. The analysis methods are simpler, more efficient and more focused.

The 16S rRNA gene is about 1500bp long the nine hypervariable regions labeled as V1-V9 range from about 30-100 base pairs long. Due to sequencing

⁶https://en.wikipedia.org/wiki/16S_ribosomal_RNA

technology read length limitations, most 16S methods operate on just a single one of these regions at any given time. For example V4. Picking the “best” region is a source of lots of (somewhat pointless) discussions.

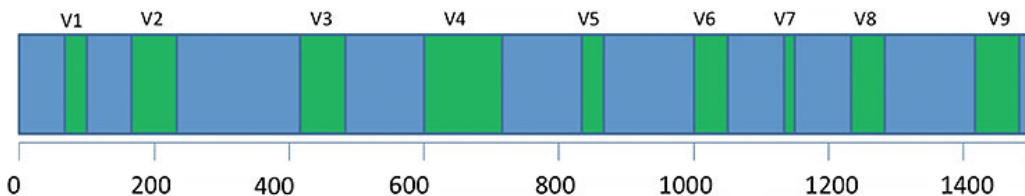


Figure 122.3

122.7 What is whole-genome metagenomics?

This method is identical to conventional whole genome sequencing. The mixture of DNA that originates from many different organisms is randomly sheared into smaller fragments (100-600bp) that are then sequenced. Of course unlike the situations where the origin of the DNA is from a single organism a metagenomic will represent a mixture of DNA. The main complication arises from following factors:

1. Each organism may be present in wildly different abundances.
2. The sequencing represents random sample.
3. All measures are relative to the total number of DNA fragments.

122.8 How many bacteria have been fully sequenced?

In a subsequent chapter, we will see how to get this data. For now, we’ll say that NCBI stores 7318 fully sequenced bacterial genomes. There are much more sequences of course that are known to correspond to bacteria but have not yet been assembled into a single genome.

The NCBI bacterial taxonomy currently labels 65,231 species under the superkingdom of bacteria.

122.9 How many bacteria are unknown?

Scientists believe that the vast majority of bacteria are yet unknown. Depending on who you ask large numbers potential species are considered to exist, sometimes ranging into millions. Notably when the scientist providing the estimate is involved in so called “big data analysis” their assessment tends to be very large (billions). You may even notice an interesting pattern, The publication from 2004: Status of the Microbial Census, Microbiology and Molecular Biology Reviews, 2014⁷ states that:

Estimating the diversity of life is a persistent challenge in biology. In microbiology, the task is complicated by the fact that the subjects of the census are not visible to the naked eye or easily differentiated morphologically, and they are estimated to number over 10^{30} individual bacteria worldwide (30). The properties of microorganisms necessitate the use of indirect

Figure 122.4

and it cites a paper from 1998 titled “Prokaryotes: the unseen majority, PNAS, 1998” that in turn states that:

The subsurface is defined here as terrestrial habitats below 8 m and marine sediments below 10 cm. Few direct enumerations of subsurface prokaryotes have been made, largely because of the difficulty in obtaining uncontaminated samples. Nevertheless, circumstantial evidence suggests that the subsurface biomass of prokaryotes is enormous (20). For instance, groundwater from deep aquifers and formation water from petroleum deposits contain 10^3 – 10^6 prokaryotic cells/ml (21, 22).

Figure 122.5

and cites a paper from 1994 titled “The deep hot biosphere, PNAS, 2014” and so on. The pattern might be familiar now, as the citations go back in time each publication is a little less rigorous and a little more speculative. The problem here is that what was acceptable as speculation in the early 1990s has, by the virtue of being cited over and over, become “a fact”.

Now, we do need to recognize that bacteria are subject to more diverse mechanisms for evolution. It makes sense to assume that there would be more variety to them. The confounding factor is that there is no definition of when two bacteria should be labeled as different species. Discussing this distinction

⁷<http://mmbr.asm.org/content/68/4/686.full>

is beyond the scope of this book. Interested readers consult chapters from: Chapter 3: Classification in Medical Microbiology. 4th edition.⁸ or the many other resources.

In summary, there are many inconsistencies when trying to place labels on organisms as diverse as bacteria.

⁸<https://www.ncbi.nlm.nih.gov/books/NBK8406/>

Chapter 123

What is a metagenomics analysis?

123.1 What type of analyses will I need to perform?

Metagenomics analyses span the whole gamut from well-defined tasks of the classification category i.e. “what is in my pot” to the highly speculative tasks where you may be called upon to evaluate the “metabolic potential” of metagenomes. Whereas the former job can be objectively evaluated, the success of the second types of analyses will strongly depend not just on your results but perhaps more so on your persuasion skills.

123.2 What online tools can I use?

Metagenomics is a unique field in that most of the time its results need to characterize hundreds if not thousands of species. For that reason, visualization takes a central role. Unlike other fields of bioinformatics, there are several alternatives for online analysis and visualization can be performed. All these choices can also be confusing. It is almost never clear exactly what type of studies a given site offers - they all seem to promise “everything” yet the implementation details vary considerably. Only after you upload

the data do you get to see the limitations of the tools. For that reason, we firmly prefer command line tools though these add substantial overhead and complexity to the analysis.

Another factor to consider is that most online tools are developed to operate on a single dataset. Most scientists collect dozens if not more samples for each experiment. Processing the multitude of samples efficiently poses radically different challenges that online services may not be able to handle.

We will mention a few online tools, then in next chapters, we will perform operations with command line tools.

- MGRAST: Metagenomics Analysis Server¹
- VIROME²
- QIITA³
- EBI Metagenomics⁴

123.3 How can I quantify and compare results?

Terms such as alpha, beta diversity were introduced to describe biodiversity. The alpha diversity is the species diversity at a local scale (same site). The beta diversity represents differences in species composition among different sites.

Do note however that the mathematical definitions behind these words are not uniquely defined. There are several ways to calculate each. You should think in terms of categories such as:

1. Community richness
2. Community evenness
3. Community diversity
4. Shared community richness
5. Similarity in community membership
6. Similarity in community structure

¹<http://metagenomics.anl.gov/>

²<http://virome.dbi.udel.edu/>

³<https://qiita.ucsd.edu/>

⁴<https://www.ebi.ac.uk/metagenomics/>

7. ...

Each category, in turn, can have one or more formulas that capture one dimension of the concept. See for example the [mothur calculators][calculator] that the above listing was taken from. As you can see there, for example, the **Berker Parker Index** is calculated as the ratio of the abundance of the dominant OTU divided by the total number of individuals in the sample. It is a fraction that represents how much of the total does the most abundant species correspond to.

The BP index is a single number and you should think of it as one, particular characteristic of the data. There are several more that you could (and should) compute. A Simpson index, a Jaccard similarity, etc. Picking the right choice depends a great deal on what your data looks like and what changes take place. Therefore you need to pick the right “calculator” for the job.

123.4 What command line tools may be used to analyze metagenomics data?

The metagenomics command line analysis communities are split into two categories:

1. QIIME⁵
2. mothur⁶

See the Biostar question of the day Qiime Vs Mothur : Why Use One Over The Other?⁷. Here is what I wrote as an answer a few years ago about them, this distinction still holds well:

There is a fundamental almost philosophical difference in how the tools are developed

Mothur is a single program that re-implements a large number of very useful algorithms into a single, high performance stand-alone executable program for each platform: linux, mac and

⁵<http://qiime.org/index.html>

⁶<http://www.mothur.org/>

⁷<https://www.biostars.org/p/94869/>

windows. In my opinion, it is one of the most amazing feats of bioinformatics software engineering especially considering that is being developed by only two people.

QIIME is a python interface (glue) that connects a very large number of disparate programs and what QIIME really does is transforms the input/outputs of these and allows you to feed one program into the other. You will need to install a very large number of dependent programs to use QIIME. In fact it is not even correct to say that you used QIIME to compute something, you are using the programs that QIIME runs for you: `pynast`, `uclust` etc.

Now because in QIIME each program is developed independently some by entire groups some of these may offer higher performance than what is implemented in mothur. On the other hand the unified interfaces of mothur make it far more consistent and better documented not to mention easier to run.

123.5 What are typical steps of an analysis?

Tools like Mothur and QIIME offer detailed instructions that you can follow. For example the Mothur SOP (Standard Operating Protocol)⁸ for MiSeq data lists and describes in great detail the following steps:

1. Reducing sequencing and PCR errors
2. Processing improved sequences
3. Assessing error rates
4. Processing OTUs
 - Calling OTUs
 - Phylotypes
 - Phylogenetic Analysis
5. Diversity
 - Alpha diversity

⁸https://www.mothur.org/wiki/MiSeq_SOP

- Beta diversity measurements
 - Population-level analysis
6. Phylotype-based analysis
 7. Phylogeny-based analysis

We recommend following the instructions as described there.

Chapter 124

Introduction to taxonomies

124.1 What is a taxonomy?

The “taxonomy” is the science of defining and naming groups of biological organisms by shared characteristics. Swedish botanist Carl Linnaeus ushered a new era of taxonomy in the 18th century when he proposed to characterize organisms into a hierarchical scheme:

Domain -> Kingdom -> Phylum -> Class -> Order -> Family -> Genus -> Species

But then what was an excellent idea in the 18th century for classifying higher order organisms is less suited to the needs of the 21st century when we need to describe microorganisms. The Linnean scheme is not appropriate for characterizing bacterial species where the spectra of differences go beyond the top-down, vertical, hierarchical representation. For that reason, the classification of a bacteria into any given taxonomical rank is far less informative when it comes to the function and phenotype.

The flaws of the taxonomical classification have led to creating several alternatives versions to them - for an outsider this only adds to the confusion.

124.2 What the heck is an OTU?

As soon as you get into the field, you’ll soon realize that the word OTU is deeply embedded into the vocabulary of meta-genomicists - yet many of

them will struggle to explain what an OTU is or is not. Other words are built by using the word OTU:

- OTU calling
- OTU clustering

etc. many suffering from the same level ambiguity. Common misunderstanding includes believing that the term has a more substantial meaning than what it stands for, as well as believing that finding an OTU is the result of a well determined, unique process.

The word OTU stands for “Operational Taxonomic Unit.” The term itself is almost meaningless: it was originally defined as the taxonomic unit that was being studied, “operated” with. Obviously, this is a very loose definition. The meaning got narrowed when the OTU started to represent “things” that were also “similar.” Later the concept of “similarity” grew to represent similarity over a marker gene, such as the 16S gene.

Today in most parlance the OTU accounts for a grouping of sequences based on their sequence similarity. An OTU group does imply that the sequences have identical (or even similar) taxonomical assignments, though there may be a representative taxonomical classification that gets assigned to represent an OTU. Further complications arise as well when sequences could belong equally well in more than one OTU or many very different clades.

124.3 How many taxonomies are available?

Too many. Picking and sticking to a taxonomy appears to have an almost tribal quality to it. For a more detailed discussion see SILVA, RDP, Greengenes, NCBI and OTT — how do these taxonomies compare?¹ in BMC Genomics 2017 where the authors conclude that:

While we find that SILVA, RDP and Greengenes map well into NCBI, and all four map well into the OTT, mapping the two larger taxonomies on to the smaller ones is problematic.

¹<https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12864-017-3501-4>

124.4 Which taxonomy should I use?

We typically use the NCBI taxonomy as it makes it a lot easier to connect to other data sources. There are also command line tools from `edirect` to `taxonkit` that were designed to interact with the taxonomy files developed by NCBI.

Scientists (while few admit that the problem exists) can be biased when it comes to their pet projects. In those cases, it is best to go with the flow. Go ahead and read a few papers and see what taxonomy is commonly used in the field that you need to publish in and choose that.

124.5 How to view the NCBI taxonomy?

The NCBI web site at <https://www.ncbi.nlm.nih.gov/taxonomy> provides access to a visual display of the taxonomy.

124.6 What is the story with the NCBI disclaimer?

Most pages on the NCBI taxonomy will show you the following “disclaimer”:

Disclaimer: The NCBI taxonomy database is not an authoritative source for nomenclature or classification - please consult the relevant scientific literature for the most reliable information.

Thanks for clearing that up NCBI (note: sarcasm). May we humbly ask though, if you are *not authoritative* then why are you storing and distributing this data at all? And if you know so well that you are not the *authoritative source* then what is the *authoritative* resource? Why are you punting the responsibility back to us, the readers?

Mocking aside the way we read this is that making taxonomies turned out to be somewhat subjective and controversial. NCBI tries to fend off any criticism by pointing to this disclaimer.

124.7 How to get the NCBI taxonomy file?

NCBI states that it has phased out the `gi` numbers even though some tools rely on this information. The “old-style” tools require `gi` to `taxid` type of information whereas the “new-style” tools work on `accession` to `taxid` mapping.

```
URL=https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/accession2taxid/nucl_gb.accession2taxid.gz
curl -k $URL | gunzip -c > acc2taxid.txt
```

It contains a mapping of `gi` number to taxonomy id.

```
cat acc2taxid.txt | head
```

It prints:

```
accession accession.version taxid gi
A00002 A00002.1 9913 2
A00003 A00003.1 9913 3
X17276 X17276.1 9646 4
X60065 X60065.1 9913 5
```

The “old-style” files are still available and approximately 1.5Gb in size and it was last updated:

```
URL=https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/gi_taxid_nucl.dmp.gz
curl -O $URL
```

It contains a mapping of `gi` number to taxonomy id.

```
gzcat gi_taxid_nucl.dmp.gz | head
```

like so:

```
2 9913
3 9913
4 9646
5 9913
7 9913
...
```

The old style files may cease to be updated at some point (we are not even sure if that has already come). Still, as of now, there may be tools that need the information in this old format. Thankfully we can “convert” a new style file to the old-style with:

```
cat acc2taxid.txt | grep -v taxid | awk '{ print $4, "\t", $3 }' | head
to produce:
```

```
2    9913
3    9913
4    9646
5    9913
11   9913
```

124.8 How many taxonomical ranks are there?

Whereas the Linnean taxonomy prescribed nine ranks for the taxonomy the NCBI data has a lot many more:

```
URL=https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar.gz
wget $URL
tar zxvf taxdump.tar.gz
```

```
cat nodes.dmp | cut -f 3 -d '||' | uniq -c | sort -rn
```

that produces:

```
1281994  species
217328   no rank
82878    genus
20883    subspecies
8594     family
6826     varietas
2775     subfamily
1990     tribe
1420     order
1314     subgenus
807      superfamily
480      subtribe
474      forma
438      species group
331      suborder
```

309	class
220	phylum
133	species subgroup
132	subclass
104	infraorder
49	superorder
28	subphylum
15	infraclass
10	parvorder
5	superkingdom
4	superclass
3	kingdom
3	cohort
2	superphylum
1	subkingdom

Quite a few many more ranks than you might expect.

124.9 How do I search the NCBI taxonomy from the command line?

124.9.1 Access taxonomy with EDirect

For some use cases, searching NCBI with Entrez Direct may be sufficient. For other more advanced functionality we recommend the `taxonkit` command line program.

124.10 How do I search the taxonomies?

What does the taxid 9913 stand for?

```
fetch -db taxonomy -id 9913
```

it will print:

1. *Bos taurus*
(cattle), species, even-toed ungulates

124.10.1 Access taxonomy with TaxonKit

```
taxonkit list --show-rank --show-name --ids 9913  
prints:  
9913 [species] Bos taurus
```

124.11 How do I find the taxid of a taxonomy name?

124.11.1 Taxonomy name with EDirect

For this, to work you'd have to (reasonably) correctly spell the scientific name then `esearch` will work:

```
esearch -query "E. coli[txn]" -db taxonomy | efetech  
prints:  
562
```

124.11.2 Taxonomy name with TaxonKit

You have to specify the exact name with `taxonkit`:

```
echo Escherichia coli | taxonkit name2taxid  
prints:  
Escherichia coli      562
```

As you can see above `taxonkit` is designed to work with streams. If you had a file that contains:

```
Archaea  
Bacteria  
Eukaryota  
Viroids  
Viruses  
other sequences  
unclassified sequences
```

you could get the taxid of each term with:

```
cat names.txt | taxonkit name2taxid
```

to obtain:

Archaea	2157
Bacteria	2
Eukaryota	2759
Viroids	12884
Viruses	10239
other sequences	28384
unclassified sequences	12908

124.12 How do I get the taxid for an accession number?

```
esearch -db nuccore -query NC_009565,NC_002570,NC_008358 | elink -target taxonomy
# 336982
# 272558
# 228405
```

```
efetch -db taxonomy -id 336982,272558,228405
# 1. Mycobacterium tuberculosis F11
#   high GC Gram+
# 2. Bacillus halodurans C-125
#   firmicutes
# 3. Hyphomonas neptunium ATCC 15444
#   a-proteobacteria
```

124.13 How do I get the lineage of a taxid?

124.13.1 Lineage with EDirect

```
efetch -db taxonomy -id 562 --format xml > out.xml
```

will produce quite the output. You can process that XML output with `xtract` or other tools:

```
cat output.xml | xtract -pattern LineageEx/Taxon -element TaxId,ScientificName
prints:
```

```
131567 cellular organisms
2 Bacteria
1224 Proteobacteria
1236 Gammaproteobacteria
91347 Enterobacteriales
543 Enterobacteriaceae
561 Escherichia
```

you can also format it in a row oriented way with:

```
cat output.xml | xtract -pattern TaxaSet/Taxon -element TaxId,ScientificName
that prints:
```

```
562 131567 2 1224 1236 91347 543 561 Escherichia coli  cellular organisms
```

124.13.2 Lineage with TaxonKit

```
echo 9913 | taxonkit lineage
```

prints:

```
9913 cellular organisms;Eukaryota;Opisthokonta;Metazoa;Eumetazoa;Bilateria;Deuterost
```

124.14 How do I list all taxids of a given rank?

This command lists all the taxonomies that correspond to the viruses (taxid 10239)

```
taxonkit list --ids 10239 --indent "" | head
```

Will print:

```
10239
12333
```

12340

12347

12366

12371

...

If you were to store the above query into a file called `virus.taxid.txt` you would see how many taxonomical ranks do virii have:

```
wc -l virus.taxid.txt
```

```
5285 virus.taxid.txt
```

124.15 How many species of viruses are in the current taxonomy?

```
taxonkit list --ids 10239 --indent "" --show-rank --show-name | grep species | head -n 10
```

prints:

12340 [species] Enterobacteria phage 933J

12347 [species] Actinophage JHJ-1

12366 [species] Streptococcus pyogenes phage H4489A

12371 [species] Phage h30

12374 [species] Lactococcus phage

12375 [species] Lactococcus phage (ISOLATE 7-9)

12386 [species] Lactococcus phage mi7-9

12388 [species] Mycobacterium phage FRAT1

12392 [species] Lactobacillus phage mv4

12403 [species] Leuconostoc phage P32

or if you want to count them:

```
taxonkit list --ids 10239 --indent "" --show-rank --show-name | grep species | wc -l
```

prints:

4824

The answer is that NCBI knows of 4824 virus species distributed over 5285 ranks.

Chapter 125

How do I obtain sequences based on taxonomies?

125.1 How do I get all known bacterial genomes?

It used to be that obtaining all known bacterial genomes from NCBI was as easy as downloading a prebuilt FASTA file that contained these genomes. Alas this practice has been discontinued and replaced with a somewhat convoluted series of actions¹:

```
# Get the summary as a tabular text file.  
curl -O ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/assembly_summary.txt  
  
# Filter for complete genomes.  
awk -F "\t" '$12=="Complete Genome" && $11=="latest"{print $20}' assembly_summary.txt > ...  
  
# Identify the FASTA files (.fna.) other files may also be downloaded here.  
awk 'BEGIN{FS="/";filesuffix="genomic.fna.gz"}{ftpdir=$0;asm=$10;file=asm"_"file...'
```

Moreover at this point the official NCBI instructions basically punt the task right back to you the reader and tell you to “use a script” to download the data for the URLs collected into this file. Here is a way that we can do:

¹<https://www.ncbi.nlm.nih.gov/genome/doc/ftpfaq/#allcomplete>

```
mkdir all
cat ftpfilepaths | parallel -j 20 --verbose --progress "cd all && curl -O {}"
```

The above will put all the genomes into the `all` directory that now contains.
Do note that the process may take many hours.

```
GCF_000005825.2_ASM582v2_genomic.fna.gz
GCF_000005845.2_ASM584v2_genomic.fna.gz
GCF_000006605.1_ASM660v1_genomic.fna.gz
...
```

At this point the folder contains sufficiently large number of files that shell meta-character expansion won't work anymore and you will get errors like the following:

```
ls -l all/*.gz
-bash: /bin/ls: Argument list too long
```

This is because the shell attempts to expand the pattern and list all the matching files *before* executing the command. You can still list all files:

```
ls -1 | wc -l
7182
```

Due to the limitation mentioned before if you wanted to sub-select files with a certain meta-character you would need to use `find` command and pipe the results into the next action:

```
find all -name '*fna.gz' | wc -l
7182
```

125.2 Are there other ways to get all bacterial/viral genomes?

Probably, though you should not get your hopes too high. On this topic there are many tutorials and guides yet it appears that most advice becomes obsoleted at a rapid pace. NCBI appears to move around the files quite a bit while also changing the data naming and formatting guidelines. In just 4 month the advice in the Biostar Question of the Day: Extract all bacteria sequences from the nr database² has become invalid.

²<https://www.biostars.org/p/236690/>

125.2. ARE THERE OTHER WAYS TO GET ALL BACTERIAL/VIRAL GENOMES?985

NCBI offers advice and support services that may be of some use - though we can't help but wonder of why the processes need to be so complicated. Below is an "official" answer for the question on how to obtain all viral genomes from NCBI:

Dear Colleague,

NCBI processes submitted (GenBank/INSDC) viral genomes differently than prokaryotic/eukaryotic genomes: - In general, we do not create Assembly records for the primary (submitted) genomes. As you have already established there is /genomes/refseq/viral path on the Genomes FTP site: <ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/viral/> that is populated with assemblies, while the genomes/genbank/viral path has a negligible content (that is coming from viral metagenomes, an oddball of environmental data that can't be made into RefSeq by RefSeq's standards): <ftp://ftp.ncbi.nlm.nih.gov/genomes/genbank/viral/> - NCBI generates RefSeqs only for selected GenBank records, so to get coverage for each viral species: at least one assembly, sometimes more. The good thing is that part of the RefSeq processing is to calculate (find) GenBank neighbors for the RefSeq entries. The total of the neighbors approximates the number of complete GenBank/INSDC genomes. - You can retrieve genome neighbors on the web by using one of the two methods that are described in this viral genomes resource FAQ: https://www.ncbi.nlm.nih.gov/genome/viruses/about/HowTo/#retrieve_neighbors Note that the number of displayed records will be larger than the number of the actual genomes, because the genomes for some organisms are multi-segmented and there is an individual record for each segment. There is another exception that I have to mention: RefSeq does not calculate neighbors for the flu virus. You will have to obtain these sequences separately as described below. Since you are after all of the GenBank genomes, the download from the web may time out for those formats that generate larger files. In such case use Batch Entrez or eutils as described in this

article: <https://support.ncbi.nlm.nih.gov/link/portal/28045/28049/Article/1928> For the influenza sequences, go to the Influenza virus resource: <https://www.ncbi.nlm.nih.gov/genomes/FLU/Database/nph-select.cgi?go=database> Click on the FTP link at the top of the page, that will take you to: <ftp://ftp.ncbi.nih.gov/genomes/INFLUENZA/>

Refer to the README file in the directory.

125.3 How do I set up the BLAST for taxonomy operations?

To enable blast searches that recognize taxonomies you have to place the contents of the `taxdb.tar.gz` file in the location where blast looks for databases.

```
wget ftp://ftp.ncbi.nlm.nih.gov/blast/db/taxdb.tar.gz
```

Then move this file to the blast database directory. Once this is set up the `-outfmt` parameter to `blast` may contain taxonomy related placeholders `%T`, `%t` etc.

```
blastdbcmd -db 16SMicrobial -entry all -outfmt "%a %T %t" | head
```

produces:

```
NR_118889.1 36819 Amycolatopsis azurea strain NRRL 11412 16S ribosomal RNA gene, p
NR_118899.1 1658 Actinomyces bovis strain DSM 43014 16S ribosomal RNA gene, partial
NR_074334.1 224325 Archaeoglobus fulgidus strain DSM 4304 16S ribosomal RNA gene,
NR_118873.1 224325 Archaeoglobus fulgidus strain VC-16 16S ribosomal RNA gene, co
NR_119237.1 224325 Archaeoglobus fulgidus strain VC-16 16S ribosomal RNA gene, co
NR_118890.1 1816 Actinokineospora fastidiosa strain ATCC 31181 16S ribosomal RNA
NR_044838.1 1381 Atopobium minutum strain NCFB 2751 16S ribosomal RNA gene, comple
NR_118908.1 1068978 Amycolatopsis methanolica strain 239 16S ribosomal RNA gene,
NR_118900.1 1655 Actinomyces naeslundii strain DSM 43013 16S ribosomal RNA gene,
NR_044822.1 2075 Pseudonocardia nitrificans strain IFAM 379 16S ribosomal RNA gen
```

Note how the results include the taxid 36819 and scientific name `Amycolatopsis azurea`.

125.4 How do I extract sequences by taxonomy from a BLAST database?

Suppose we wanted all sequences deposited under the taxonomy id 1392

```
taxonkit list --show-name --show-rank --ids 1392
```

that corresponds to *Bacillus anthracis* species:

```
1392 [species] Bacillus anthracis
  191218 [no rank] Bacillus anthracis str. A2012
  198094 [no rank] Bacillus anthracis str. Ames
  205919 [no rank] Bacillus anthracis str. Kruger B
  212045 [no rank] Bacillus anthracis str. Western North America USA6153
  260799 [no rank] Bacillus anthracis str. Sterne
  261591 [no rank] Bacillus anthracis str. Vollum
```

The following steps would achieve that

```
URL=https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/accession2taxid/nucl_gb.accession2taxid
curl $URL | gunzip -c > acc2taxid.txt
```

```
# Filter accession numbers by the known taxonomy 9606.
cat acc2taxid.txt | awk '$3 == 1392 {print $1}' > acc.txt
```

```
# Extract sequences that belong to that taxonomy.
cat acc.txt | blastdbcmd -db nt -entry_batch - > sequences.fa
```

125.5 What does the env_nt blast database contain?

NCBI states:

Sequences from environmental samples, such as uncultured bacterial samples isolated from soil or marine samples. The largest single source is Sagarss (Would this be Sargasso ?) Sea project. This does NOT overlap with nucleotide nr

```
mkdir -p ~/blastdb  
(cd ~/blastdb && update_blastdb.pl --decompress env_nt)  
blastdbcmd -db env_nt -entry all -outfmt "%a %T %S" | head  
prints:
```

```
AAFX01139340.1 256318 metagenome  
AAFX01139339.1 256318 metagenome  
AAFX01139338.1 256318 metagenome  
AAFX01139324.1 256318 metagenome  
AAFX01139323.1 256318 metagenome  
AAFX01139322.1 256318 metagenome
```

most of this data consists of short, random, unlocalized and unidentified sequences obtained during large scale sequencing.

Chapter 126

Classifying 16S sequences {#16S-classification}

126.1 What data will be analyzed?

For this chapter, we will use data from the Human Microbiome Project (HMP) Demonstration Projects that contain 1065 sequencing runs that attempt to connect the disease of Necrotizing enterocolitis a devastating complication of prematurity to microbial composition of the gut.

- <https://www.ncbi.nlm.nih.gov/bioproject/46337>

It is important to note that often data with clinical relevance has many limitations imposed due to patient privacy practices. To fully analyze this data in its full context you would need permission to access phenotype (disease) related information on each patient. In the following, we limit ourselves to publicly available information.

Get all the SRR runs for the project (1065 runs in total):

```
# Search the runs for this project.  
esearch -query PRJNA46337 -db sra | efetch --format runinfo > runinfo.csv
```

Investigating the file (open in Excel as it is comma separated) we can see that the first run id is SRR1614899. Get the data for it:

```
# Obtain the data for the run.  
fastq-dump --split-files SRR1614899
```

This produces three files, with no explanation of what these are. A little detective work is necessary. Depending on the instrumentation related metadata makes it into the repository. For example the barcode by which the instrument tagged the samples or the primer that was used to isolate the sequences.

In this case the first file `SRR1614899_1.fastq` contains the sequence `TCAGACGGCTC` and second file `SRR1614899_2.fastq` contains the 16S rRNA primer that was used to select the 16S rRNA region `CCGTCAATTCAATTGAGT`. Checking the 16S rRNA wiki page¹ we can see how this sequence corresponds to primer `907R`.

Finally, the file called `SRR1614899_3.fastq` contains the primary dataset and will be used for data analysis.

126.2 Can I classify 16S sequences via alignment?

The most “logical” and “straightforward” approach to classification - though as we’ll see it is neither the most accurate nor efficient is to align the sequences to their closest match. Since we do expect that many of our sequences will only be partially similar but not necessarily identical to known sequences a local alignment tool, like BLAST might be the most appropriate:

```
# Get the 16S database and place it into a newly made folder called blastdb.
mkdir -p ~/blastdb && cd ~/blastdb && update_blastdb.pl --decompress 16SMicrobial

# Convert FASTQ file to FASTA
seqtk seq -A SRR1614899_3.fastq > SRR1614899.fa

# Align converted sequence with blastn
blastn -db ~/blastdb/16SMicrobial -query SRR1614899.fa > SRR1614899.blastn
```

The input file has 7280 sequences. Guess how many lines does the `SRR1614899.blastn` contain? Done? There are probably more than you think, a lot more. Perhaps add a few zeroes to estimate. In the case above the total number of lines in the file was 88,073,057, and the data was

¹https://en.wikipedia.org/wiki/16S_ribosomal_RNA

generated in about 30 minutes. Apparently, this process would not scale well as the amount of data rises.

In the resulting alignment file, each sequence may contain hundreds if not thousands of hits to all other similar 16S genes. Of course, since all 16S genes are similar to one another the resulting file is humongous. Obviously, the method is extremely redundant. You'd be surprised however just how many publications use blast to classify sequences.

126.3 How do I visualize the 16S classification?

The Metagenome Analyzer (MEGAN)² is a software can be used to load up the data. There are different versions of the tool, and as it happens the latest version, MEGAN 6, appears to have some data loading bugs. Only certain types of data can be visualized with it. In the following images use MEGAN version 5. Explaining what MEGAN does and how it works is beyond the scope of this book. The tool has a detailed manual and several publications that describe the use cases.

What we will do is demonstrate a few of the visualizations. For example, MEGAN can draw a taxonomical tree with the number of reads assigned to a rank:

Our `SRR1614899.blastn` classification will look like the image above. Note how the classification stops at the `Family` rank.

126.4 Are there better ways to classify 16S sequences?

Oh, indeed there are. Blast alignment “kind-of” work but are an incredibly redundant and wasteful choice for this process. For the 16S gene, in particular, far more efficient classifiers have been built that rely on the patterns in the sequences rather than alignments.

²<https://ab.inf.uni-tuebingen.de/software/megan5>

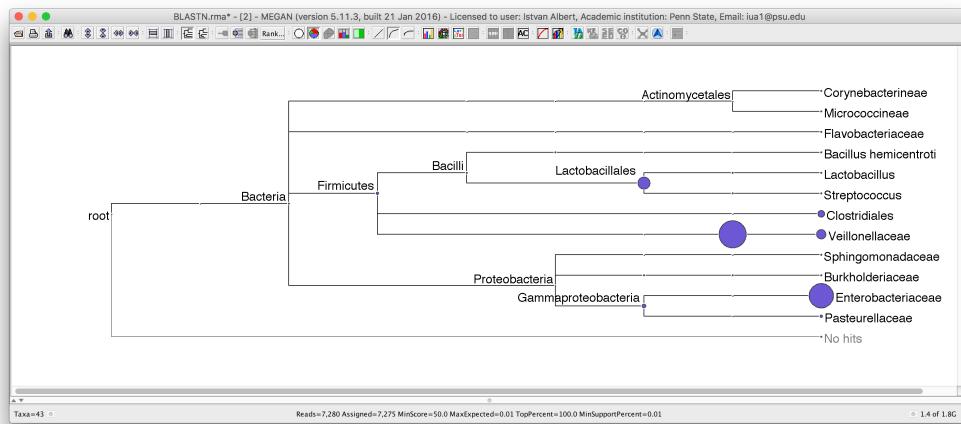


Figure 126.1

We prefer the RPD classifier as a command line choice. It used to be that it was distributed as a simple command line tool that required only Java to be installed. Today is a more complicated as described in RDPTools GitHub page³. Once installed you can run the classifier as:

```
java -jar ~/src/RDPTools/classifier.jar classify SRR1614899_3.fastq -o assign.txt
```

This process classifies the same number of sequences as before but does so in just 50 seconds and produces much smaller files. The `assign.txt` file contains a taxonomical assignment for each input read and can be loaded up into Megan to visualize the results. The `ranks.txt` file contains the number of reads assigned to a taxonomical rank and the number of reads (count) assigned to that rank.

Note how this classification has now assigned reads up to the genus level though still not at species level that many might have expected.

³<https://github.com/rdpstaff/RDPTools>

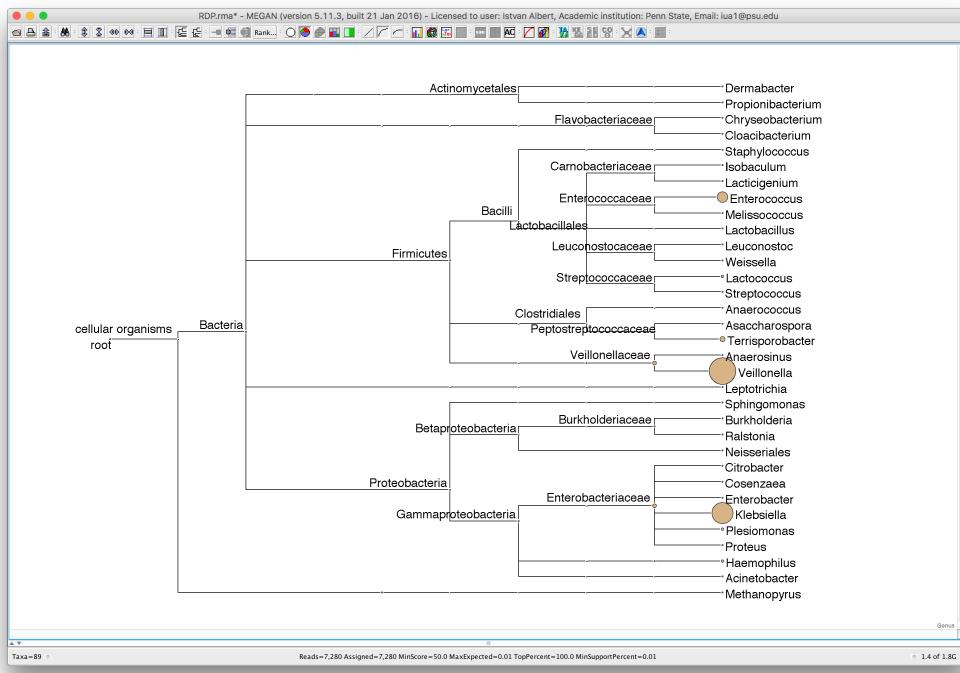


Figure 126.2

126.5 How can I visually compare classifications?

MEGAN allows us to compare classifications. For example, we may compare the BLAST and RDP results to draw the image below. Note just how much more detailed the blue assignments are. RDP (blue) can classify the sequences faster and at a lower rank than BLAST (light green). Unfortunately, the limitations of BLAST for classification are not well understood and large numbers of scientific publications still make use of it.

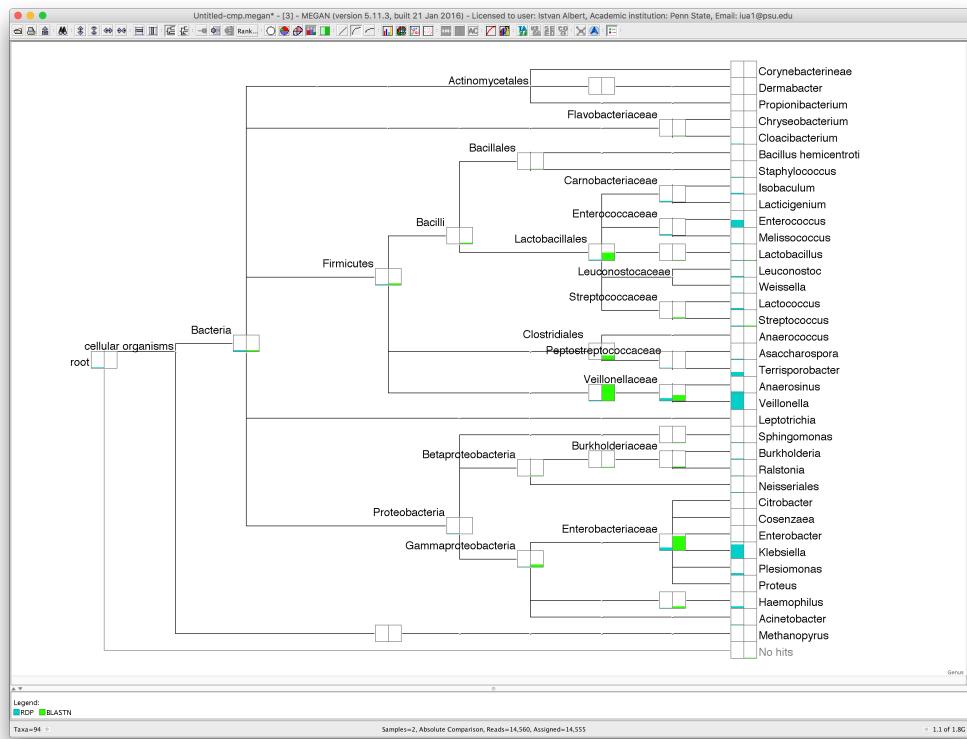


Figure 126.3

126.6 Are there other 16S classification methods?

Yes, there is a wide variety of tools and techniques, often integrated into automated pipelines. See the chapter on Metagenomics Analysis

126.7 How do I classify multiple samples?

The RPD classifier can take multiple datasets and can classify them simultaneously. Let's get data for four files and classify all four:

```
# Obtain four runs.
```

```
cat runinfo.txt | cut -f 1 -d , | grep SRR | head -4 | xargs -n 1 fastq-dump --split
```

```
# Classify all four datasets.
```

```
java -jar ~/src/RDPTools/classifier.jar classify *_3.fastq -f allrank -o assign.txt -h ra
```

the resulting file contains a tabular ranking that :

```
cat ranks.txt | cut -f 1,3-10 | head
```

of the form:

			taxid	name	rank	SRR1614899_3.fastq	SRR1614900_3.fastq	SRR1614901_3.fastq	SRR1
0	Root				rootrank	7280	3590	8891	11472
1	Bacteria				domain	7256	3572	8850	11393
841	"Proteobacteria"				phylum	2674	3055	933	8000
1501	Gammaproteobacteria				class	2653	3039	933	7993
1616	"Enterobacteriales"				order	2589	3030	933	7975
1617	Enterobacteriaceae				family	2589	3030	933	7975
1649	Plesiomonas				genus	2	2	0	3
1637	Klebsiella				genus	2100	2450	598	6413
1631	Enterobacter				genus	2	1	0	1

You can subselect the classification by different taxonomical levels:

```
cat ranks.txt | cut -f 1,3-10 | awk '$3=="class" { print $0}' | sort -rn -k 4
```

that produces:

					2722 Negativicutes	class	3497	3	41	2
1501	Gammaproteobacteria					class	2653	3039	933	7993
2260	Bacilli					class	803	428	5831	3079
2375	Clostridia					class	220	6	492	19
1175	Betaproteobacteria					class	14	6	0	0
842	Alphaproteobacteria					class	5	8	0	0
458	Flavobacteriia					class	3	13	0	0
3	Actinobacteria					class	2	18	160	143
790	Fusobacteriia					class	1	0	0	0
423	"Bacteroidia"					class	0	0	1135	0

A lot can be learned from a classification as simple as this. While we are at it note how widely different the counts are across samples.

126.8 How do I determine statistically relevant differences between the classification counts?

There is a debate on whether the count data obtained via classification is conceptually similar to the counts obtained when assigning reads to transcripts. If the statistical assumptions used to model data originating from RNA-Seq experiments still apply to metagenomics data, then we could use the wide variety of methods designed for RNA-Seq to analyze classification counts as well.

One such assumption, for example, is that the total number of non-changing elements is substantially larger than the number of differentially expressed entries (basically that only a small subset changes). If this requirement is not met, then methods originally developed for RNA-Seq should not be used unless there is a reason to believe that this requirement may be ignored.

Much has been written about the subject of comparison and normalization; a recent overview can be read in the paper Normalization and microbial differential abundance strategies depend upon data characteristics⁴ the authors summarize their findings in what we interpreted as: “Well.. it depends on a lot of factors. Good luck my friend!”

Let’s give it a shot here. Suppose we have two groups. The first three runs belong to one group, the other to another group. Put these runs into a file called `groups.txt`.

```
SRR1614918
SRR1614919
SRR1614920
SRR1614917
SRR1614922
SRR1614924
```

obtain the data and classify it:

```
mkdir -p sra
cat groups.txt | xargs -n 1 fastq-dump --outdir sra --split-files --skip-technical
```

⁴<https://microbiomejournal.biomedcentral.com/articles/10.1186/s40168-017-0237-y>

126.8. HOW DO I DETERMINE STATISTICALLY RELEVANT DIFFERENCES BETWEEN THE C

```
# Ensure to list the SRR numbers by groups. First three, second three.  
java -jar ~/src/RDPTools/classifier.jar classify SRR1614918_3.fastq SRR1614919_3.fastq  
  
# Filter the data for only the columns of interest.  
# Add a header as well.  
cat ranks.txt | head -1 | cut -f 3,5-11 > counts.txt  
cat ranks.txt | awk '$4 == "genus" { print $0 }' | cut -f 3,5-11 >> counts.txt  
  
# Obtain the deseq2 script from the Biostar Handbook.  
curl -O http://data.biostarhandbook.com/rnaseq/code/deseq2.r > deseq2.r  
  
# Run the deseq2  
cat counts.txt | Rscript deseq2.r 3x3 > results_deseq2.txt  
  
# Draw the heatmap  
cat norm-matrix-deseq2.txt | Rscript draw-heatmap.r > output.pdf
```

That will, in turn, generate a heat map at genus level for the different patients and the genus level abundance of the various bacteria.

One thing is clear - there is a lot of variation across the different groups. Every individual in every group shows radical changes in their metagenomic composition. On the other hand, we don't know if the groups were built correctly - the grouping data was not available due to privacy reasons.

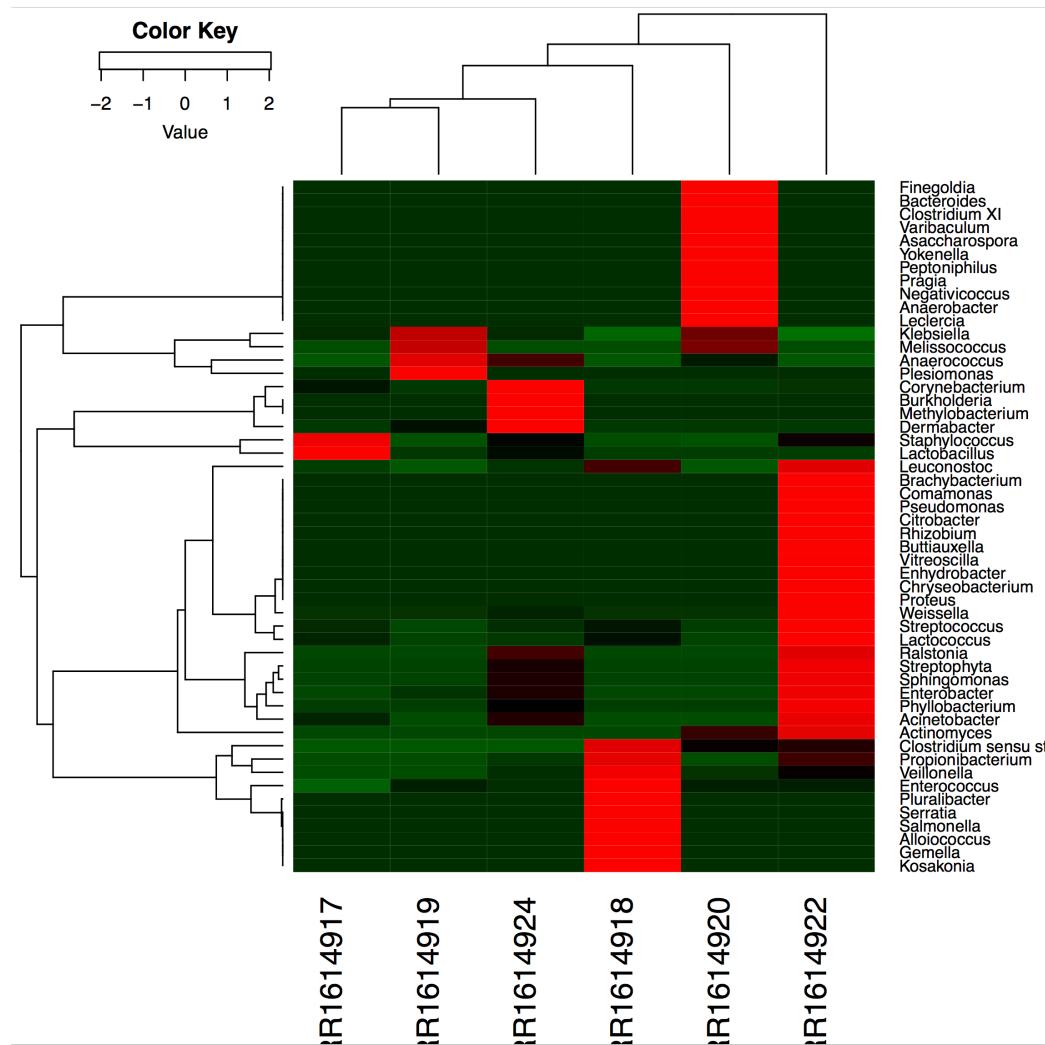


Figure 126.4

Chapter 127

Classifying whole genome data

Shotgun metagenomic sequencing allows scientists to study the full DNA of organisms present in a given “biological sample.” Whereas 16S sequencing allows you to group bacteria into OTUs and different taxonomical ranks, whole shotgun sequencing allows for a more detailed study of organisms from classifying into subspecies to strains to investigating functional roles as well.

Of course, there is the added complexity of having to deal with massively increased amounts of data that often are of many orders of magnitude larger than a 16S type study.

127.1 How do I evaluate how well a given method works?

Since most bacteria are currently unknown, it is somewhat difficult to assess methods as we have to evaluate success relative to a small known subset of reality. Also, as we found out, there is a bit of wild west mentality in the field, as you will see it is surprisingly difficult to evaluate methods as we continuously struggle with data janitorial tasks: the naming of bacteria can vary wildly, taxonomies not used consistently, etc.

Several publications have attempted to objectively quantify the quality of metagenomic techniques, among them the paper titled Assessment of Metagenomic Assembly Using Simulated Next Generation Sequencing Data, PLoS

One, 2012¹. We will use a simulated data set from this paper to get a practical feel for the strengths and weaknesses of the various methods.

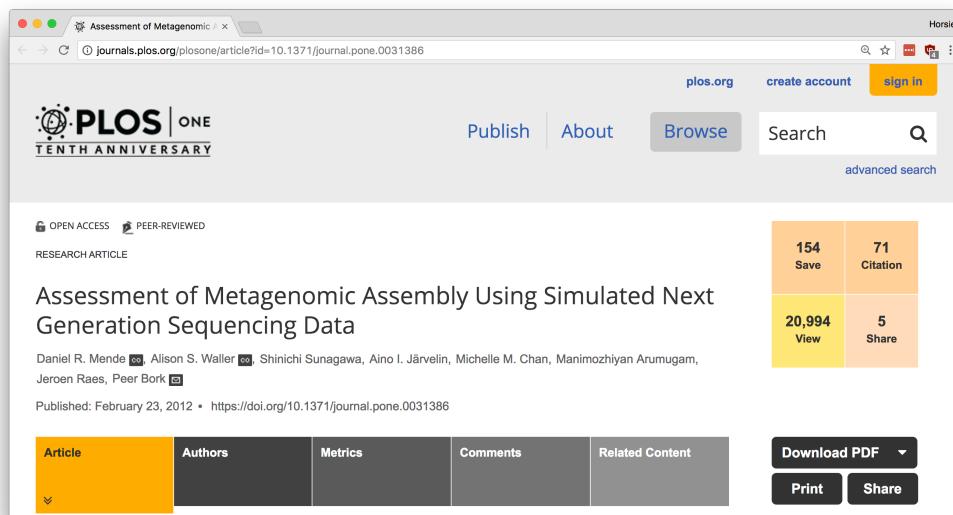


Figure 127.1

The data is distributed from:

- http://www.bork.embl.de/~mende/simulated_data
- http://www.bork.embl.de/~mende/simulated_data/README.txt
- Supporting Table 2: <https://doi.org/10.1371/journal.pone.0031386.s002>

127.2 How do I obtain the data?

If your computer can handle the computational load you could download the original data from the site above with

```
curl http://www.bork.embl.de/~mende/simulated_data/illumina_100species.1.fq.gz
curl http://www.bork.embl.de/~mende/simulated_data/illumina_100species.2.fq.gz
```

¹<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0031386>

127.3. WHAT ARE THE EXPECTED ABUNDANCES OF THE DATA?1001

And run your analyses on `read1.fq` and `read2.fq` respectively. That being said these datasets are a wee bit too large for exploratory analyses. We have down-sampled the first-pair file into 1 million random reads with:

```
seqtk sample -2 -s 2 read1.fq 1000000 > subset.fq
```

and made it available from the Biostar Handbook data site. You can download this data with:

```
curl http://data.biostarhandbook.com/meta/data/subset.fq.gz | gunzip -c > subset.fq
```

```
# See what the file contains.
seqkit stat subset
#file      format type  num_seqs   sum_len min_len avg_len max_len
#subset.fq FASTQ  DNA   1,000,000 75,000,000    75     75     75
```

This subsampled data can help us in evaluating classification methods but would not be sufficient to say assemble new genomes.

127.3 What are the expected abundances of the data?

The paper comes with some supplementary files among them:

```
# This will download the supplemental info in XLS format.
curl -OJL https://doi.org/10.1371/journal.pone.0031386.s002
```

In this case before we could evaluate a classification method we ought to understand what we expect to see - what kind of properties do the artificially generated data have? Look at the Excel file that we downloaded above.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Supplemental Table 2 - Genomes Used in the Medium Complexity Metagenome and Estimated Coverage (100 genomes)													
Rank	Chromosome	Chromosome	Name	phylogenetic group	genome size (Mb) ¹	GC %	Percent abundance ²	Est_proportion of sequences	Estimated coverage for 454	Estimated coverage for Illumina			
1 NC_007969	0	Psychrobacter cryohalolentis K5		Gammaproteobacteria	3.1	49.2	1000	2.27683986	1.865791273	1.50224206	1.20179	17.84686	
2 NC_009077	0	Mycobacterium sp. JLS		Actinobacteria	6	68.4	794.3106671	1.769473519	2.863794984	1.19324791	0.9546	14.17579	
3 NC_007604	0	Synechococcus elongatus PCC 7942		Cyanobacteria	2.75	55.4	707.1067812	1.575210382	1.168470804	1.06224619	0.8498	12.61948	
4 NC_004722	0	Bacillus cereus ATCC 14579		Firmicutes	5.42	35.3	656.2595204	1.461938759	2.137347181	0.96586125	0.78689	11.71203	
5 NC_002662	0	Lactococcus lactis subsp. lactis II1403		Firmicutes	2.4	35.3	621.9749249	1.385653518	0.896983155	0.93435745	0.74749	11.10017	
6 NC_004344	0	Wigglesworthia glossinidiae endosymbiont of Glossina brevipalpis		Gammaproteobacteria	0.7	22.5	596.8306535	1.329550698	0.251043828	0.8965851	0.71727	10.65143	
7 NC_000091	0	Escherichia coli str. K12 substr. W3110		Gammaproteobacteria	4.65	50.8	577.3502692	1.286153891	1.613215904	0.86732038	0.69386	10.30377	

Figure 127.2

Take for example the first entry (table sorted by coverage) states that `Psychrobacter cryohalolentis K5` corresponds to accession numbers

NC_007969 and will be present at an expected coverage of 17.84. Let's see how well the data supports this information.

```
# This took
cat subset.fq | egrep '^@NC_007969|^@NC_007968' | wc -l
# 15902

# Get the size of NC_007969 + NC_007968
efetch -db=nuccore -id=NC_007969,NC_007968 -format=fasta | seqkit stat
# file format type num_seqs sum_len min_len avg_len max_len
# - FASTA DNA 2 3,101,097 41,221 1,550,548.5 3,059,876
```

Above we see that a total of 15902 reads were found as being generated from *Psychrobacter cryohalolentis* K5. The original data contained about 53 million reads and we used a subset of 1 million of it, we can extrapolate (estimate) the total number of reads in the original data as being about $15902 * 53 \sim 842,000$. Since we did download the full data it is worth checking out how well the estimated number matches reality:

```
cat read1.fq read2.fq | egrep '^@NC_007969|^@NC_007968' | wc -l
# 849442
```

The result is very close indeed. The accuracy of our estimate gives us confidence that we'll be able to estimate other quantities of interest just as well even from a sample of 1 million reads.

127.4 Are there different kinds of abundances?

The word abundance can carry different meanings. Whereas during 16S sequencing each read corresponds to a bacterial cell, in the case of whole genome sequencing the length of the genome also matters. Longer bacteria will produce more DNA and will be seen in more sequencing reads. Another way to say this is that ten read counts of one bacteria vs. one read count of another bacteria does not mean that there were more of the first than the second.

127.5 What percent of a measurements originate from a given species?

This measure depends on the length of the DNA of the organism multiplied by the number of copies of that organism. It can be found by dividing the number of reads that are classified at that rank with the total number of reads in the sample:

15902/1000000

This value does not vary when we subsample the data. In both cases 1.5% of the reads in our file were quantified as belonging to *Psychrobacter cryohalolentis* K5. This number does not appear in the supporting materials of this research paper. Some classifiers will only report this value.

Again it is important to remember that these numbers do not indicate the counts of the bacteria in the sample. They indicate what percent of measurements were “spent” identifying one particular member of the community.

127.6 What is the coverage?

As you recall the coverage represents the number of times each base of genome is expected to be sequenced. The genome size for NC_007969 can be found with:

```
efetch -db=nuccore -id=NC_007969,NC_007968 -format=fasta | seqkit stat
#file format type num_seqs sum_len min_len avg_len max_len
#- FASTA DNA 2 3,101,097 41,221 1,550,548.5 3,059,876
```

Since each read is 75 bp long and you have sequenced 849442 reads the coverage of this bacteria will be $849442 * 75 / 3101097 = 20.5x$. Notably, this does not match the coverage in the Excel sheet that is reported to be 17.4x. The difference is not substantial, but it is a bit discomforting that we are unable to reproduce it from the data. It is not clear why this discrepancy exists - unfortunately it is not uncommon to run into such issues.

127.7 How can I tell how many bacteria are in the sample?

First, you need to recognize that sequencing does not produce absolute numbers. From sequencing data alone we cannot tell if a bacteria was present in one, ten, one million or one billion copies. What we can identify are the so called “relative abundances” how many more of one bacteria are there compared to another.

When a coverage tells you that an organism was covered at 10x versus another one covered at 1x it directly means that the first must have had ten times as many more copies present as the latter. To figure out the relative abundance percentages sum up all coverages then express each coverage as a percentage of the total.

For example, if you had bacteria A, B, and C covered at 10x, 20x and 30x coverages ($10+20+30=60$) then their relative abundances will be $10/60=17$, $20/60=33$ and $30/60=50$. The fractions help to express the composition as percentages. Your sample is then composed of 17% bacteria A, 33% bacteria B and 50% bacteria C. These numbers represent abundances based on the number of cells.

127.8 How do I match the results of a classifier?

As we prepared this chapter, we realized that it is surprisingly difficult to verify and validate the results produced by classifiers. Several inconsistencies may exist in both the expected data and the results generated by our software. Sometimes the taxonomy id is missing, in other cases, the names may be spelled slightly differently, for each organism, or perhaps the accession numbers are not annotated with a taxonomy etc.. This makes it surprisingly difficult to match results to expectations.

127.9 How does Kraken classifier work?

The classifier has quite high performance and can be run with:

```
kraken -db ~/refs/kraken/minikraken_20141208 -fastq-input subset.fq > results.out
```

that produces the output:

```
26667004 sequences (2000.03 Mbp) processed in 401.844s (3981.7 Kseq/m, 298.63 Mbp/m).
 16202163 sequences classified (60.76%)
 10464841 sequences unclassified (39.24%)
```

The above was generated with the “prebuilt” mini-kraken database distributed by the authors. The `results.out` file tracks each input read and lists the classification for each measurement. The information typically needs to be consolidated into a more informative report with:

```
kraken-report -db ~/refs/kraken/minikraken_20141208 results.out > results.txt
```

Where the `results.txt` generated above is a file that contains the sample abundances at each taxonomical rank (check the Kraken Manual² for details on what each column means):

39.38	393820	393820	U	0	unclassified
60.62	606180	3126	-	1	root
60.31	603053	9	-	131567	cellular organisms
59.29	592937	930	D	2	Bacteria
25.12	251162	491	P	1224	Proteobacteria
13.69	136877	323	C	1236	Gammaproteobacteria
6.69	66892	0	O	91347	Enterobacteriales
6.69	66892	24149	F	543	Enterobacteriaceae
1.00	10042	426	G	561	Escherichia
0.96	9611	8721	S	562	Escherichia coli
0.06	602	602	-	331111	Escherichia coli E24377A
0.02	243	243	-	405955	Escherichia coli APEC O1
0.00	21	19	-	83333	Escherichia coli K-12
0.00	2	2	-	316407	Escherichia coli str. K-12 substr.
...					

²<http://ccb.jhu.edu/software/kraken/MANUAL.html>

Each row indicates the taxonomical rank (for example S means Species). To select the rows at species rank with at least 100 reads supporting that rank

```
cat results.txt | awk '$2>100 && $4=="S" { print $0}' | wc -l
# 83
```

Hence the classifier predicts the presence of 83 species. It remains to be seen if these are indeed matching the 100 species that we know were present. It is clear though that we have missed out on some 20% of the species. To compare to the expected data reformat the `results.txt` file to select as species level and keep the taxid and name of the bacteria (we do also sort it so that we can later join this file to another):

```
# Put the results with their taxid into a file
cat results.txt | awk '$2>100 && $4=="S" { OFS="\t"; print $5 }' | sort > kraken.txt
```

This file called `kraken.taxid` now contains just the taxonomic ids:

```
101534
106590
1076
1100
...
...
```

We will now attempt match this to the expected data. Because the tax ids were not specified for the data in the paper it is surprisingly challenging to compare the expected results to the obtained ones.

127.10 How do I match back the bacteria?

As you'll experience yourself often the greatest of challenges occur when trying to do something really simple, that has seemingly nothing to do with bioinformatics all. In this case we want to figure out how well the classification worked, we want to match the file `kraken.txt` to the contents of the XLS sheet distributed as Supplementary information. It is a surprisingly tough job to get right.

The taxid is present in one file but os not there in the second, hence we first need to generate a taxid for each entry of the XLS sheet. Lets give it a go. First create a `tmp` file by pasting the bacteria names from the D column of

the Excel sheet into it. Then cut the first two words and place them into the file called `names.txt`:

```
cat tmp | cut -f 1,2 -d ' '
```

So that stream will give you this:

```
Psychrobacter cryohalolentis
Mycobacterium sp.
Synechococcus elongatus
Bacillus cereus
...
```

Now pipe this output into the taxonkit:

```
cat tmp | cut -f 1,2 -d ' ' | taxonkit name2taxid
```

that will give you:

Psychrobacter cryohalolentis	330922
Mycobacterium sp.	1785
Synechococcus elongatus	32046
Bacillus cereus	1396

We notice that some bacterial names did not produce a taxid and there are duplicates there as well. Two species may correspond to same taxid.

```
cat tmp | cut -f 1,2 -d ' ' | taxonkit name2taxid | cut -f 2 | sort | uniq | egrep "[0-9]+>
```

When all said and done the `expected.taxids` file contains 76 distinct taxonomic ids. That is quite the drop from the original 100.

The `comm` tool allows us to select lines that are common (or different in files).

```
# Taxids in both file.
comm -1 -2 expected.taxid kraken.taxid | wc -l
# 70
```

```
# Taxids missed in classification.
comm -2 -3 expected.taxid kraken.taxid
# 6
```

```
# Taxids present in the kraken classification only.
comm -1 -3 expected.taxid kraken.taxid | wc -l# 78
#13
```

So what was missed

```
comm -2 -3 expected.taxid kraken.taxid | taxonkit lineage
```

Expected species not found:

```
1765  cellular organisms;Bacteria;Terrabacteria group;Actinobacteria;Actinobac
1773  cellular organisms;Bacteria;Terrabacteria group;Actinobacteria;Actinobac
1785  cellular organisms;Bacteria;Terrabacteria group;Actinobacteria;Actinobac
1831  cellular organisms;Bacteria;Terrabacteria group;Actinobacteria;Actinobac
203804 cellular organisms;Bacteria;Proteobacteria;Gammaproteobacteria;Enteroba
50422  cellular organisms;Bacteria;Proteobacteria;Gammaproteobacteria;Alteromo
```

127.11 Would the full kraken database be more accurate?

If we were to build the most up-to-date full database as instructed in the Kraken manual (though the process is a bit tedious) the rate of classification ratio rises substantially:

```
kraken --db /export/refs/kraken_db/kraken_std_db/ --fastq-input subset.fq > resu
that in turn produces:
```

```
1000000 sequences (75.00 Mbp) processed in 25.376s (2364.5 Kseq/m, 177.33 Mbp/m).
 826977 sequences classified (82.70%)
 173023 sequences unclassified (17.30%)
```

So even in the “best-case scenario” of a controlled experiment with 100 known species 17% of our sequences will be unclassified even though they match known genomes.

127.12 How to use sourmash?

According to their documentation:

sourmash is a command-line tool and Python library for computing MinHash sketches from DNA sequences, comparing them to

each other, and plotting the results. This allows you to estimate sequence similarity between even very large data sets quickly and accurately.

```
pip install https://github.com/dib-lab/sourmash/archive/2017-ucsc-metagenome.zip
pip install khmer
```

Download and install the sourmash signature database:

```
mkdir -p ~/refs/sourmash
URL=https://s3-us-west-1.amazonaws.com/spacegraphcats.ucdavis.edu/microbe-refseq-sbt-
(cd ~/refs/sourmash && curl -k $URL | tar xzv)
```

Compute a signature on your subset of data:

```
sourmash compute --scaled 10000 -k 21 subset.fq
```

Characterize the signature:

```
sourmash sbt_gather -k 21 ~/refs/sourmash/refseq-k21.sbt.json subset.fq.sig -o sourmash
```

The tool will produce somewhat differnt information on the screen versus the results file `sourmash.csv`

overlap	p_query	p_genome	
1.4 Mbp	2.7%	24.5%	NZ_CP009464.1 Bacillus anthracis strain
1.1 Mbp	2.2%	40.1%	NZ_JOPX01000001.1 Staphylococcus aureus
1.1 Mbp	2.1%	26.0%	NZ_CP009685.1 Escherichia coli str. K-12
1.0 Mbp	1.9%	15.9%	NC_009077.1 Mycobacterium sp. JLS, compl
0.8 Mbp	1.6%	9.0%	NC_010628.1 Nostoc punctiforme PCC 73102
0.8 Mbp	1.5%	8.0%	NC_008268.1 Rhodococcus jostii RHA1, com
...			

whereas the `sourmash.csv` file contains:

```
intersect_bp,f_orig_query,f_found_genome,name
1410000.0,0.2447916666666666,0.02744258466329311,"NZ_CP009464.1 Bacillus anthracis st
1140000.0,0.40070921985815605,0.022187621642662515,"NZ_JOPX01000001.1 Staphylococcus a
1100000.0,0.26004728132387706,0.02140910860256909,"NZ_CP009685.1 Escherichia coli str.
970000.0,0.15901639344262294,0.018878941222265473,"NC_009077.1 Mycobacterium sp. JLS, o
800000.0,0.08958566629339305,0.015570260801868432,"NC_010628.1 Nostoc punctiforme PCC 7
...
```

How well did the method work? First `sourmash` runs substantially slower than `kraken`, at least 10x slower. Then the accession numbers are not useful for matching purposes since the same sequence may be present under different accession numbers. Let us test how well the results match. Note the command line contraptions that we had to come up with:

```
cat sourmash.csv | cut -d ',' -f 4 | cut -d ' ' -f 2,3 | taxonkit name2taxid | cut -
```

Now let's compare the outputs:

```
# Taxids in both file.
comm -1 -2 expected.taxid sourmash.taxid | wc -l
# 69

# Taxids missed in the sourmash classification.
comm -2 -3 expected.taxid sourmash.taxid | wc -l
# 7

# Taxids present in the sourmash classification only.
comm -1 -3 expected.taxid sourmash.taxid
#6
```

what has been missed by `sourmash`:

```
comm -2 -3 expected.taxid sourmash.taxid | taxonkit lineage
```

```
1100  cellular organisms;Bacteria;FCB group;Bacteroidetes/Chlorobi group;Chlor
1172  cellular organisms;Bacteria;Terrabacteria group;Cyanobacteria/Melainabac
1765  cellular organisms;Bacteria;Terrabacteria group;Actinobacteria;Actinobac
1831  cellular organisms;Bacteria;Terrabacteria group;Actinobacteria;Actinobac
203804 cellular organisms;Bacteria;Proteobacteria;Gammaproteobacteria;Enteroba
65   cellular organisms;Bacteria;Terrabacteria group;Chloroflexi;Chloroflexia;He
9    cellular organisms;Bacteria;Proteobacteria;Gammaproteobacteria;Enterobacter
```

Quite a few are the same as before.

127.13 What is binning?

Binning is the process of grouping reads or contigs and assigning them to “bins” that will represent taxonomical ranks or taxonomical units. Typically, binning methods are based on sequence compositional features.

The incomplete nature of the sequencing coupled to ambiguity introduced by the method itself (breaking long pieces of DNA into smaller fragments) makes it hard to assemble individual genes, much less recovering the full genomes of each organism. Binning techniques represent a “best effort” to identify reads or contigs with certain groups of organism.

Typically reads are first assembled into longer contigs, then these contigs are binned into categories.

127.14 How do I assemble metagenomes?

This topic is so early in the development of the field that we can't, in good faith cover it here. There are many tools that can be run just as other assembler are run, but assessing their results and understanding tuning their quality, interpreting the assemblies are still in our opinion no more than just guesstimates.

Most practical advice follows the rule: “Find an assembler that others say is good, keep running it with different parameters until the results seem to make sense”. It is weak sauce - but the pretty much the only advice we can share as well.

Chapter 128

Alaska Oil Reservoir Metagenome study

Note: this chapter is unfinished.

The reason we kept it is to show you what happens when you go to remote part of Alaska and sequence samples from the ice. Unsurprisingly only 10% of your samples can be assigned to known species. But then you have to write a paper from that. Quite the conundrum of course.

128.1 Which data are we going to re-analyze?

We will obtain data for a terrestrial metagenome project with id PRJNA278302

The URL for the bioproject is at:

- <https://www.ncbi.nlm.nih.gov/bioproject/PRJNA278302>

128.2 How do I get more information on the project?

The `runinfo` and `summary` formats contain information on the runs:

```
esearch -db sra -query PRJNA278302 | efetch -format runinfo > runinfo.txt
```

128.2. HOW DO I GET MORE INFORMATION ON THE PROJECT?1013

The screenshot shows the NCBI BioProject page for project ID PRJNA278302. The main content area is titled "terrestrial metagenome" and describes the "Alaska oil reservoir Metagenome". It details hydrocarbon reservoirs as major sites of methane production and carbon turnover, mentioning significant impacts on energy resources and global biogeochemical cycles. Samples were collected from six locations between 1,128 m (24-27°C) and 2,743 m (80 - 83°C) below the surface. The text notes that sample complexity decreased with increasing temperature. Bacteria from candidate phyla (CP) were highly represented in the lower temperature zones and to some extent in the intermediate temperature zones. Most microorganisms in the intermediate and higher temperature samples were related to previously studied methanogenic and non-methanogenic archaea and thermophilic bacteria. The reconstruction of a near-complete genome for an organism sampled from the lower temperature reservoir SB1 revealed an OD1 population. The genome is small and metabolic predictions support an obligate anaerobic, fermentation-based lifestyle. At moderate abundance in samples were members of other CP bacteria, including TA06, OP11, OP9, Marinimicrobia (SAR406) and OP1. For OP1, a reconstructed first near-complete genome based on the inventory of glycosyl hydrolase genes and the absence of respiratory capacity, it was inferred that the predominant roles for the OP1 bacterium are fermentation of mannose, cellulose or cell wall carbon polymers and hydrogen cycling. Overall, the results elucidate roles of uncultivated organisms in hydrocarbon reservoir biological processes. A "Less..." link is present.

On the right side, there is a sidebar titled "Related information" which includes links to BioSample, Full text in PMC, Genome, Nucleotide, Protein, PubMed, SRA, Taxonomy, and WGS master. Below this is a section for "LinkOut to external resources" with a link to GOLD: Gp015470 (Genomes On Line Database).

Figure 128.1

```
esearch -db sra -query PRJNA278302 | efetech -format summary > summary.xml
```

In the latter case the result is an XML file that you could view in a browser:

```
<><EXPERIMENT_PACKAGE_SET>
<><EXPERIMENT_PACKAGE>
<><EXPERIMENT xmlns="" alias="S122_K3" accession="SRX997729" center_name="Lawrence Berkeley National Lab">
<><IDENTIFIERS>
  <PRIMARY_ID>SRX997729</PRIMARY_ID>
  <SUBMITTER_ID namespace="Lawrence Berkeley National Lab">S122_K3</SUBMITTER_ID>
</IDENTIFIERS>
<><TITLE>
  metagenome K3(S122) from petroleum reservoir (this well has been treated with materials containing sulfide),
  Kuparuk formation, Alaska North Slope
</TITLE>
<><STUDY_REF accession="SRP057267">
<><IDENTIFIERS>
  <PRIMARY_ID>SRP057267</PRIMARY_ID>
</IDENTIFIERS>
</STUDY_REF>
<><DESIGN>
<><DESIGN_DESCRIPTION>
  DNA was extracted from filters having produced water passed through them using a bead-beating and organic solvent
  extraction protocol. DNA was purified using a MoBio UltraSoil DNA extraction kit starting by adding Solution S3 to
  the extract and continuing per the manufacturer's protocol. DNA was prepared for sequencing using an Illumina
  sequencing kit (paired-end, 150 bp reads) for HiSeq instrument. Library preparation was performed by the Yale
  Center for Genome Analysis.
</DESIGN_DESCRIPTION>
```

Figure 128.2

Visualizing this file will allow you to understand its (convoluted) structure. Then you can use the **xtract** tool may be used to parse this XML file and extract various components of it and format those as a tabular file.

```
cat summary.xml | xtract -pattern "EXPERIMENT_PACKAGE" -element PRIMARY_ID, TITLE
will connect the SRR run numbers to the title of each section.
```

```
SRR1977365 metagenome K3(S122) from petroleum reservoir (this well has been treated)
SRR1977357 metagenome K2 (MPF) from not soured petroleum reservoir, Kuparuk formation
SRR1977304 metagenome I2(PW) from soured petroleum reservoir, Ivishak formation,
SRR1977296 metagenome I1 (DS1215) from not soured petroleum reservoir , Ivishak f
SRR1977249 metagenome SB2 from not soured petroleum reservoir, Schrader bluffer f
SRR1976948 metagenome SB1 from not soured petroleum reservoir, Schrader bluffer r
```

128.3 How do I classify this data?

We can employ several strategies discussed in the previous chapters, though in general you should not be surprised if these don't quite work at the level of accuracy of precision that you were expecting:

```
# Get the data for one run. Add the -X 100000 flag to limit the data to a subset.
fastq-dump --split-files -skip-technical SRR1976948
```

Now it is worth checking out just how many of our sequences can be classified before and after quality control:

```
kraken -db /export/refs/kraken_db/kraken_std_db/ --threads 16 -fastq-input SRR1976948
in just 3 minutes the tool completes with the following result:
```

```
34456404 sequences (8648.56 Mbp) processed in 175.190s (11800.8 Kseq/m, 2962.00 M
```

- 3720501 sequences classified (10.80%)
- 30735903 sequences unclassified (89.20%)

It reports that that majority of the reads could not be classified.

128.4 How many species were present?

```
kraken-report -db /export/refs/kraken_db/kraken_std_db/ results.out > results.txt
cat results.txt | awk '$2>10 && $4=="S" { print $0}' | wc -l
1230
```

that starts with:

3.47	1196351	0	S	301375	Methanosaeta harundinacea
0.06	20863	0	S	2223	Methanosaeta concili
0.01	4390	0	S	420950	Methanolobus psychrophilus
0.00	1136	0	S	2209	Methanosarcina mazei
0.00	571	0	S	2214	Methanosarcina acetivorans
0.00	377	0	S	2208	Methanosarcina barkeri
0.00	734	0	S	101192	Methanomethylovorans hollandica
0.00	619	0	S	2176	Methanohalophilus mahii
0.00	170	0	S	29291	Methanococcoides burtonii
0.00	73	0	S	2322	Methanohalobium evestigatum

128.5 Does quality control improve the classification?

Let us apply quality control and see how the classification performance may change with that.

```
# Make the illumina adapter file.
echo ">illumina" > adapter.fa
echo "AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC" >> adapter.fa

# Let apply adapter trimming and quality correction.
trimmmomatic PE -baseout trimmed.fq SRR1976948_1.fastq SRR1976948_2.fastq ILLUMINACLIP:"

# Error correction with tadpole
tadpole.sh in=trimmed_1P.fq in2=trimmed_2P.fq out=corrected_1P.fq out2=corrected_2P.fq
```

Classify after error correction:

```
# Classify with kraken
kraken -db /export/refs/kraken_db/kraken_std_db/ --threads 16 -fastq-input corrected_1P.fq
```

the result is unexpected! The classification produces fewer results overall only 2424 sequences were classified instead of 7424:

```
78272 sequences (14.88 Mbp) processed in 5.743s (817.7 Kseq/m, 155.45 Mbp/m).
2424 sequences classified (3.10%)
75848 sequences unclassified (96.90%)
```

1016CHAPTER 128. ALASKA OIL RESERVOIR METAGENOME STUDY

In all honesty we don't know how to explain this. Chalk it up to one of the many mysteries you will undoubtedly run into in your career. The first rule of the "Bioinformatics Fight Club" is that there are no rules.

We can format the result differently, though that alone won't help with the results:

```
kraken-report -db ~/refs/kraken/minikraken_20141208 results.out > results.txt
```

to filter at species level with at least 10 hits we can do:

```
cat results.txt | awk '$2>10 && $4=="S" { print $0}'
```

that will produce:

1.44	1442	0	S	28108	Alteromonas macleodii
0.49	494	0	S	85698	Achromobacter xylosoxidans
0.01	11	0	S	994695	Candidatus Kinetoplastibacterium galatii
0.07	66	0	S	316277	Syntrophus aciditrophicus
0.02	17	0	S	181663	Desulfococcus oleovorans
0.35	349	0	S	1184387	Mesotoga prima
0.10	105	0	S	69499	Petrotoga mobilis
0.04	44	0	S	102134	Desulfotomaculum gibsoniae
0.01	11	0	S	58135	Desulfotomaculum kuznetsovii
0.06	58	0	S	110500	Pelotomaculum thermopropionicum
0.01	12	0	S	233055	Desulfitobacterium dichloroeliminans
0.02	19	0	S	35701	Helio bacterium modesticaldum
0.02	21	0	S	336261	Thermovirga lienii
1.98	1980	0	S	301375	Methanosaeta harundinacea
0.02	18	0	S	2223	Methanosaeta concilii
0.83	827	0	S	2198	Methanoculleus marisnigri
0.45	445	0	S	83986	Methanoculleus bourgensis
0.14	140	140	S	1379702	Methanobacterium sp. MB1
0.41	411	411	S	374840	Enterobacteria phage phiX174 sensu lato

128.6 Are there other methods to classify samples?

There is no shortage of approaches that claim to do better. In reality the results are typically decidedly less reliable.

128.7 How do I use “sourmash” to understand my data?

Based on the tool description

sourmash can also be used to quickly search large databases of genomes for matches to query genomes and metagenomes;

First install the database:

```
mkdir -p ~/ref/sourmash
cd ~/ref/sourmash
wget https://s3-us-west-1.amazonaws.com/spacegraphcats.ucdavis.edu/microbe-refseq-sbt-
tar xzvf microbe-refseq-sbt-k21-2017.05.09.tar.gz
```

You can now use sourmash to classify:

```
# Compute a signature.
sourmash compute --scaled 10000 -k 21 SRR1976948_1.fastq

# Classify the signature.
sourmash sbt_gather -k 21 ~/refs/sourmash/refseq-k21.sbt.json SRR1976948_1.fastq.sig
```

it will produce:

overlap	p_query	p_genome	
130.0 kbp	0.9%	4.7%	NZ_LN515531.1 Methanobacterium formicicu
120.0 kbp	0.8%	4.9%	NC_009051.1 Methanoculleus marisnigri JR
60.0 kbp	0.4%	2.3%	NC_017527.1 Methanosaeta harundinacea 6A
50.0 kbp	0.3%	1.6%	NC_017934.1 Mesotoga prima MesG1.Ag.4.2,

so out of 100,000 reads it was able to classify just 3 percent of them into 4 categories.

Chapter 129

Analyzing the neonatal microbiome

The following study contains analysis of full datasets. You may complete the analysis even if your computational resources by limiting the data to a smaller subset.

129.1 How to classify all the data for the project PRJNA46337?

Suppose you'd want to classify all the data in project PRJNA46337? How much of computational resource would it take? The answer depends on a number of factors data download speed as well as number of computer cores.

The project contains two kinds of data:

1. 1407 samples of 16S sequencing data producing on average 10000 reads per run
2. 6 runs of whole genome sequencing (WGS) on a Illumina Hiseq sequencer producing 70 million reads per sample.

Here we demonstrate the analysis of the 1407 samples

```
# Get the runinfo. Took 40 seconds.  
esearch -query PRJNA46337 -db sra | efetch --format runinfo > runinfo.csv
```

129.1. HOW TO CLASSIFY ALL THE DATA FOR THE PROJECT PRJNA46337?1019

```
# Keep the runs performed on a 454 sequencer.  
cat runinfo.csv | grep "454 GS FLX Titanium" | cut -f 1 -d , > srr-16s.txt  
  
# Store the 16S data here.  
mkdir -p data  
  
# Download the SRR files 10 at a time. On our system this took about 60 minutes.  
cat srr-16s.txt | parallel -j 10 --verbose --progress --eta fastq-dump --outdir data --sp
```

Annoyingly some SRR data was split into 3 pieces other into 4, hence the files that contain our biological data are sometimes called `..._3.fastq` versus `..._4.fastq`. These type of inconsistencies are very annoying and slow down the automated processing of data. We do also note that some runs failed to download for mysterious reasons. We have only 1338 datasets whereas the file lists 1397 oh well - we just carry on.

We'd like to rename each file in such a way to split off the `_3` and `_4` and keep just the SRA number itself turning `SRR1038144_3.fastq` into `SRR1038144.fastq` and `SRR052332_4.fastq` into `SRR052332.fastq` respectively. But guess what, it is time to get in touch with that data janitor slice of your soul because there is another complication that you probably miss the first time around and create the wrong solution first (like we did). The number after the `SRR` varies too, it may be 6 or 7 digits long so it renaming won't be as simple as slicing of the string at a position (an easy operation in bash). Instead we have to find and remove either `_3.fastq` or `_4.fastq` from the end of a string. Thankfully there is an operator to do that, and you find it by googling it, the `%` operator.

The sloppiness of the data modeling at SRA causes all kinds of unexpected complications right from the beginning.

So we need a renaming script. Call that `rename.sh` and have it contain:

```
for file in *.fastq; do  
    name=$file  
  
    # Chop off certain types of ends if they exists.  
    name=${name%_3.fastq}  
    name=${name%_4.fastq}  
  
    # Perform the renaming.
```

```
echo mv $file "${name}.fq"
done
```

Running this script in the data folder:

```
bash rename.sh
```

will show you the operations that it wants to perform. To have the actions perform instead of showing them to you pipe the output again into bash:

```
bash rename.sh | bash
```

We now have nicely named file. Let's classify them all:

```
# This directory will store the results of classification.
mkdir -p results

# Classify in parallel 10 at a time.
cat srr-16s.txt | parallel -j 10 --verbose --progress --eta java -jar ~/src/RDPToo...
```



```
# Classify all 1381 columns.
time java -jar ~/src/RDPTools/classifier.jar classify data/*.fq -f allrank -o res...
```

On our system this took about 7 hours and you end up with a file with over 1000 columns where each sample is fully classified.

The datamash utility can help us probe the data to compute various values:

```
cat counts.txt | datamash -H sum 2-8
```

produces:

```
sum(SRR052150.fq)  sum(SRR052151.fq)  sum(SRR052152.fq)  sum(SRR052153.fq)  su...
2509      3346      3379      3302      2988      2905      3350
```

The above shows the number of classified sequences per sample. Pretty low actually. Let us count how many bacteria of the Streptococcus genus each column has

```
cat counts.txt | grep Streptococcus | cut -f 1-20 | head
```

it produces:

```
Streptococcus 191 290 27 2536 1 0 0 309 1 0 10 370 86 1 392 166 37 0
```

It is pretty clear that the variability across samples is gigantic. Some samples indicate no such bacteria is present (SRR052155.fq) whereas others (example SRR052153.fq) show that over 75% of the bacteria (2536/3302) of

129.2. WHAT IS THE MOST COMMON MISUNDERSTANDING (PARADOX) OF METAGENOMICS?

the sequences assigned to this genus. This genus include over 50 species of bacteria and while many have quite the bad rep for causing pink-eye, meningitis, pneumonia many more are not pathogenic at all and are often present in healthy guts as well. In a manner of speaking this makes all classifications at higher levels of taxonomy less informative than initially assumed.

Now due to privacy considerations scientists were not allowed to distribute phenotype data of clinical relevance. For example you can't immediately access information that would connect samples to a disease phenotype. You would need to apply for permission to obtain that. The typical process here would create subgroups of our columns then attempt to discover trends in data for these.

129.2 What is the most common misunderstanding (paradox) of metagenomics?

Look at the results above. Note how `SRR052155.fq` has no reads assigned to the `Streptococcus` genus whereas `SRR052153.fq` has 75% of its reads assigned to it. As yourself the following

Would you conclude that the first sample contains fewer bacteria at the `Streptococcus` genus than the second?

As surprising as it sounds the answer is: NO. The only thing you may conclude is that, when compared to all the other existing bacteria, relatively speaking there are fewer `Streptococcus` bacteria in the first sample than the other.

Here is a better explanation. Imagine that both samples have the same number of `Streptococcus` bacteria in them makes that 10,000. But the first sample also has many millions of other bacteria and our random sampling of 2000 never had the chance to pick them up, they always hit something else. In the second sample there were very few other bacteria, our sampling mostly hit the `Streptococcus` variety. Hence turned out at 75%.

See how the counts reflect the relative abundances and not absolute ones. And of course that also provides us with very valuable information. We only

wish to caution against interpreting the classification rates incorrectly.

129.3 How reliable are these results?

You may wonder, just how reliable is the observation that 75% of the bacteria in a certain gut sample belongs to the *Streptococcus* genus? The short answer is that probably not that reliable.

The entire field of metagenomics is at its infancy. It is not entirely clear how to sample the same gut correctly after all there are many imhomogenities in different areas of the gut. There is variability and errors in sample extraction, handling and of the lenghty process of analysis.

A reliable method would be one that replicated each measurement multiple times and had the means to validate the accuracy and precision of each obtained value. The field is extremely far from doing that.

Instead of proper experimental design and validation scientists perfer to rely on “fancy” statistical methods that few (usually not even the people that use them) understand. There is a long way to go yet in obtaining reliabke results.

Metagenomics must still be considered as an early, exploratory science where we still bumbling around.

Part XXVIII

Appendix

129.4 Do I need to compute and discuss p-values?

Absolutely! Forget about publishing if you don't have small p-values.

Theorem: The most cynical interpretation of p-values is that their serve as the mechanism to filter out studies created by people that were so clueless with respect to large scale analysis that they couldn't even produce small p-values. From that point of view p-values do correlate with the quality and validity of the research.

Corollary: An expert scientist can unwittingly publish a Nature publication with tiny p-values, impressive and compelling visualizations even when the underlying data is not different from random noise¹.

¹<https://www.nature.com/articles/nature12535>

Chapter 130

Setting up MacOS

130.1 How do I get started?

First of all, ensure that your Mac runs the latest version of MacOS. Go to the

"Apple Icon" -> "About This Mac" -> "Software Update".

Verify that your Mac is updated to the latest version of MacOS.

130.2 Change the shell to BASH!

Starting with Mac OSX version 10.14 the default shell has been changed to zsh

All our instructions use the bash shell, thus you will need to configure the Mac OSX to use bash by default as well. Type the following into a terminal:

```
chsh -s /bin/bash
```

It will ask for your password then set the default shell.

130.3 What other App Store software is needed?

A compiler is a software that turns a “text” based program into one that the computer can execute. You will need to install the “XCode” compiler. Go to the “App Store” and find the “XCode” application. Install it, it is a free program.



You will also need to install the additional command line tools to XCode by opening the “Terminal” application, then writing the following into it:

```
xcode-select --install
```

This will trigger an installation procedure.

You may occasionally need to re-launch the XCode application whenever the license agreement created by Apple changes. You will get messages of the sort:

Agreeing to the Xcode/iOS license requires admin privileges, please run “sudo xcodebuild -license” and then retry this command.

You can accept a license agreement change from the command line by typing (requires your password):

```
sudo xcodebuild -license accept
```

The automatic software update will update XCode as well. In those cases, you may need to re-accept the license.

130.4 What is Homebrew?

HomeBrew¹ is a MacOS specific package manager that we use to facilitate the installation of various software.

130.5 Do I need Homebrew?

We typically recommend that readers using MacOS install Homebrew, though it is not strictly required. In the book we will use another package manager called `conda` that operates identically across Linux, OSX and Windows Bash, whereas Homebrew is specific to MacOS.

In addition as of June 10, 2019 readers reported problems installing Homebrew. For more details (and a potential solution) see:

- Homebrew and Xcode Command Line Tools #74²

We expect that the problems (that appear to be Homebrew specific) will get resolved in a few weeks, in the meantime, if you are running into issues you may skip the Homebrew installation and continue on with customizing the Finder. Revisit the Homebrew installation later. You will be able to follow along the book with `conda` alone.

130.6 How do I install Homebrew?

While most of the tools in the book will use the `conda` package manager, there are software packages that are best installed with Homebrew. Visit the

¹<http://brew.sh/>

²<https://github.com/biostars/biostar-handbook-issues/issues/74>

HomeBrew³ page for the installation script that at the time of writing this book was to execute:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Once installed, a program named:

```
brew
```

will be available at the command line. This program can be used to install other programs like so:

```
brew install <program name goes here>
```

130.7 What software are required?

There are quite a few libraries that tools may require. It is best if these are installed right away, sooner or later you will run a tool that needs them:

```
brew install gd libharu git imagemagick lzo hdf5 bison wget findutils coreutils
```

The commands above may take a while to process.

130.8 How do I install Java?

The Java programming language is essential for many bioinformatics tools. You can type

```
java
```

in your terminal then follow the “More info” button to get to the download page.

Alternatively, you may visit the Java JDK for MacOS⁴ page. Test your installation by typing:

```
java -version
```

³<http://brew.sh/>

⁴<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

130.9 How do I upgrade software?

Libraries may become outdated in time in which case they need to be upgraded. Periodically, you may need to run the `brew upgrade` update these:

```
brew upgrade gd libharu git imagemagick lzo hdf5 bison wget findutils coreutils
```

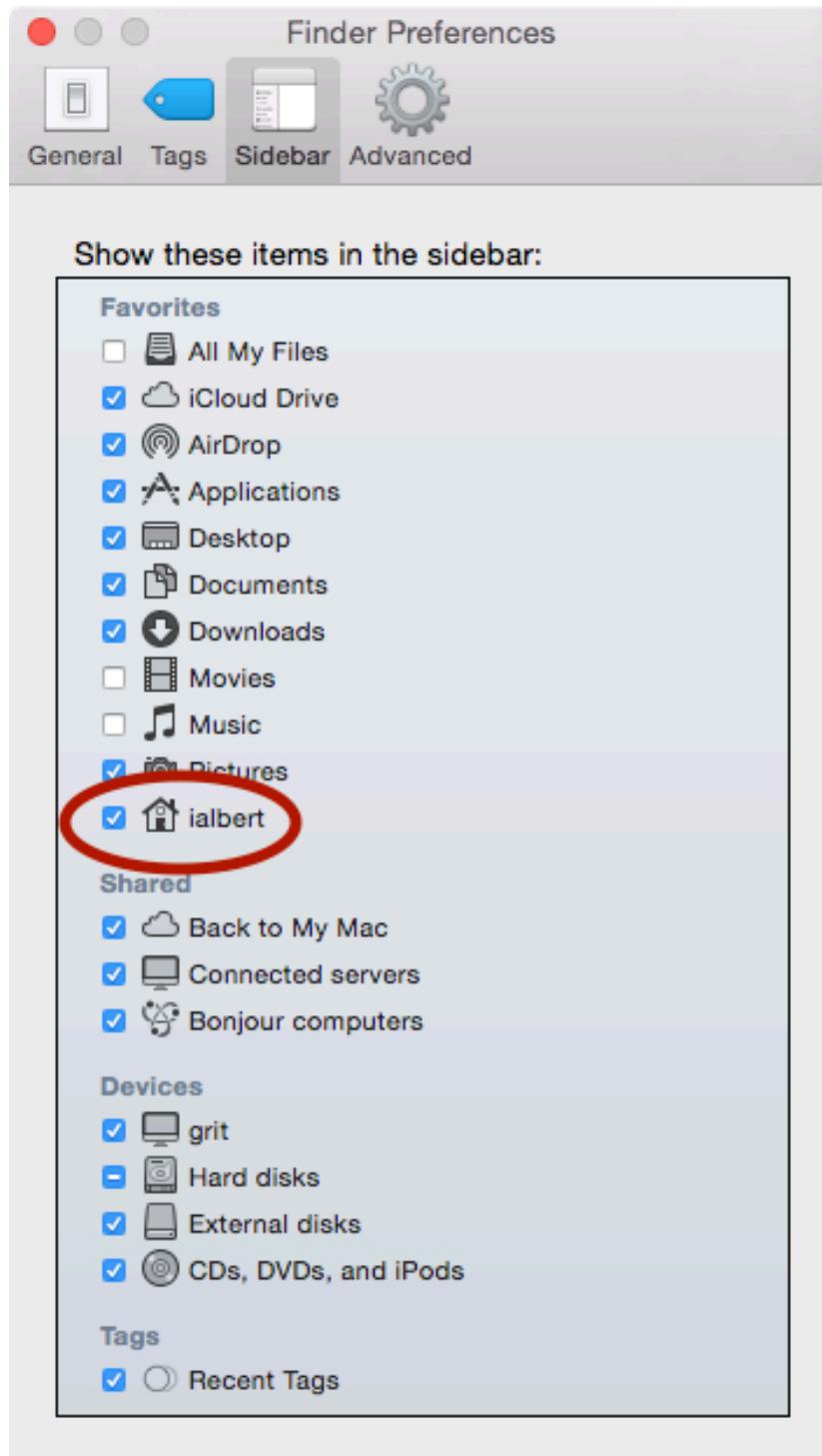
A warning: because Apple occasionally makes substantial changes to the MacOS operating system, the upgrade command may run for an extended period. Let it run in a command window and don't interrupt the process.

130.10 How do I view my home directory in the Finder?

By default, the Finder will now show the home directory that your Terminal opens to.

To be able to view your home directory within the Mac Finder you need to tick a checkbox next to it. Go to Finder->Preferences. Your home directory is named as your username. My username is `ialbert`, yours will be different. See below:

130.10. HOW DO I VIEW MY HOME DIRECTORY IN THE FINDER?1031



130.11 Will the Finder show all files?

No, some files won't be visible via the Finder. For example, the Finder will not show so-called "hidden" files, these are files whose name starts with a period: .

There is nothing really hidden about these files, it is just a convention that Unix adopted. By default, the system avoids showing file names that are typically used for configuration.

130.12 What is the next step?

Continue of with the terminal setup

Chapter 131

Setting up Linux

131.1 Which distributions of Linux does this book support?

There are numerous Linux distributions with various naming schemes.

To set up their system users will need to install packages via so-called package managers: `apt-get` or `yum` on Fedora based systems.

Our commands assume a Ubuntu/Debian based systems.

131.2 What is the first step for Linux?

1. Update your Linux system.
2. Install the required Linux libraries and Java.

131.3 How do I update a Linux based system?

On Ubuntu Linux start a “Terminal” then run the following:

```
sudo apt-get update && sudo apt-get upgrade -y
```

These commands will update your distribution (while printing copious amounts of information on the screen) then upgrade all installed packages.

131.4 What are the required libraries for Linux?

Depending on the Linux distribution's initial configuration one may need to install the package shown below. On Ubuntu the following was necessary:

```
sudo apt-get install -y curl unzip build-essential ncurses-dev  
sudo apt-get install -y byacc zlib1g-dev python-dev git cmake  
sudo apt-get install -y default-jdk ant
```

Wait for these to complete.

131.5 What is the next step?

Continue of with the terminal setup

Chapter 132

Setting up Windows

Starting with the August 2016 Anniversary Edition the Windows 10 operating system supports running Linux based software.



The Linux subsystem for Windows is a fully configurable Linux environment.

132.1 How do I install Ubuntu Linux on Windows?

In a nutshell:

1. Use the Start menu to launch the **Microsoft Store** application.
2. Search for **Ubuntu** and select the first result, ‘Ubuntu’, published by Canonical Group Limited.
3. Click on the **Install** button.

132.2 How do I start Ubuntu Linux?

Double-click the icon for the Ubuntu subsystem.

The first time this starts you it will ask you for a username and password.

Make the username simple one word: “biouser” or something similar as your home directory path will include this word. You don’t want it lengthy and complicated.

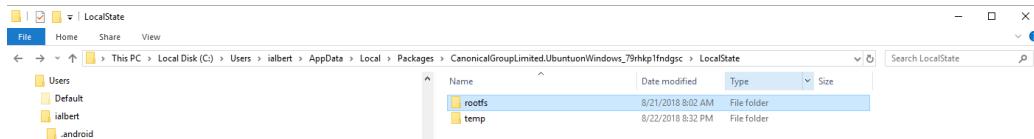
132.3 How do I see the Unix filesystem from Windows?

Typically you would want to be able to access the files from both Linux and with Windows Explorer. Setting this up will be somewhat circuitous as the Unix filesystem is tucked away in some spurious corners of your Windows machine.

Specifically when installed as an Ubuntu App your filesystem will be located at (the path below is so long that you will need to scroll to see it all:

```
C:\Users\<Windows UserName>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu
```

Here is how it may look when using the File Explorer on Windows



Where <Windows UserName> and <Ubuntu UserName> are the usernames that are set for Windows and Ubuntu respectively.



For easy access to this monstrous path, locate it, then right click to create a shortcut to it on your Desktop. You will frequently need to open this shortcut when accessing files that you create in Linux.

132.4 Install Java for Windows

Some tools, like IGV will require running Java as a Windows application. Ensure that you have Java installed as described in the official documentation at

- <https://www.java.com/en/>

132.5 What does not work on Windows?

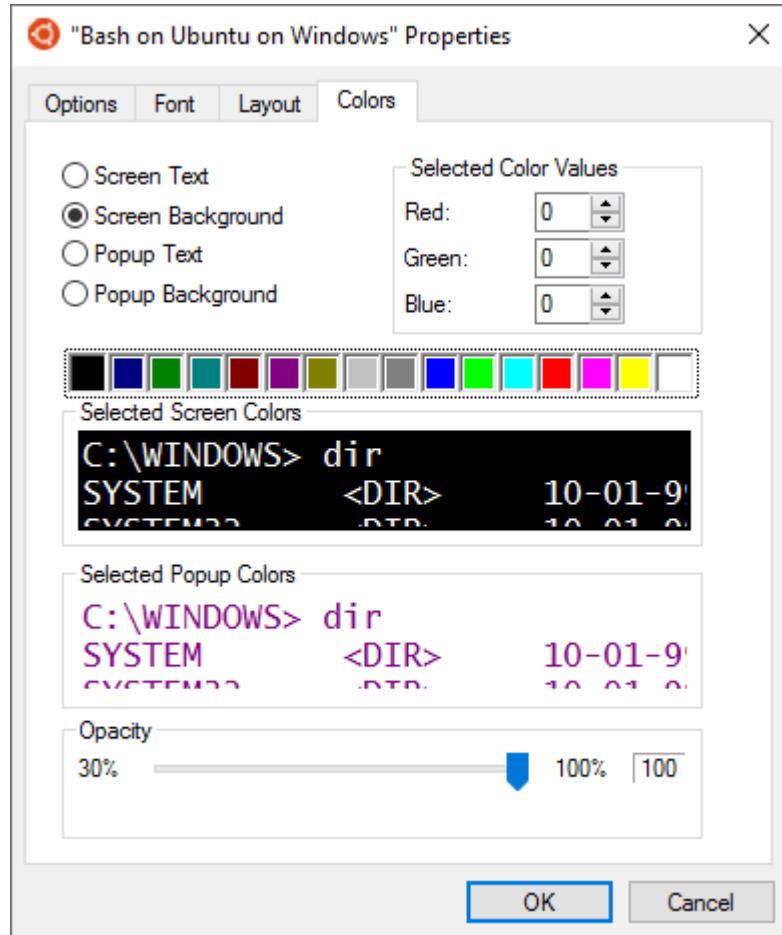
Tools that generate graphical user interfaces need to be installed in Windows, run in Windows.

For example, the Integrative Genome Viewer has a graphical interface. To run IGV, download the Windows version of it, then you would need to locate your files under windows (see above).

132.6 How do I customize the Windows Bash terminal?

The default settings for the terminal can be hard on the eyes. Right-click the terminal bar to customize the colors and fonts sizes.

Here is how it may look when using the File Explorer on Windows



132.7 How do I finish setting up Ubuntu on Windows?

Once the Ubuntu subsystem is installed as above you can treat the terminal as a standalone Linux computer.

Follow the Setup Linux page and type these commands into your Ubuntu terminal.

Chapter 133

Setting up your Bash profile

You may want to apply certain settings automatically when a terminal is opened.

133.1 What is a “shell”?

As the Wikipedia page on Shell(computing)¹ states:

In computing, a shell is a user interface for access to an operating system's services. In general, operating system shells use either a command-line interface (CLI) or graphical user interface (GUI), depending on a computer's role and particular operation. It is named a shell because it is a layer around the operating system kernel.

133.2 What is Bash?

Bash is a Unix shell and command language written for the GNU project as a free software replacement for the Bourne shell. The name is an acronym for “Bourne-again shell”.

¹[https://en.wikipedia.org/wiki/Shell_\(computing\)](https://en.wikipedia.org/wiki/Shell_(computing))

133.3 What are shell profiles?

A file that is processed when a shell is opened is called a *shell profile*.

This file has to have a special name and has to be located in the home directory. All the commands listed in this shell profile file will be applied when a new shell is initialized.

133.4 Can we have multiple shell profiles?

Yes. Due to historical and other usage considerations, multiple files may contain shell configurations. There are rules regarding which of these settings are applied when more than one shell profile file is present.

Josh Staiger in .bash_profile vs .bashrc² writes:

When you login (type username and password) via console, either sitting at the machine, or remotely via ssh: `.bash_profile` is executed to configure your shell before the initial command prompt. But, if you've already logged into your machine and open a new terminal window (xterm) inside Gnome or KDE, then `.bashrc` is executed before the window command prompt. `.bashrc` is also run when you start a new bash instance by typing `/bin/bash` in a terminal.

An exception to the terminal window guidelines is Mac OS X's Terminal.app, which runs a login shell by default for each new terminal window, calling `.bash_profile` instead of `.bashrc`. Other GUI terminal emulators may do the same, but most tend not to.”

²http://www.joshstaiger.org/archives/2005/07/bash_profile_vs.html

133.5 What's the best setup for multiple shell profiles?

You should handle this in the following way:

1. The file called `.bash_profile` should be set up to automatically load `.bashrc`.
2. The main shell profile file that you use should be `.bashrc` and it should contain all shell related settings.

The easiest way to set this up is to add the following lines to `.bash_profile`:

```
if [ -f ~/.bashrc ]; then
    source ~/.bashrc
fi
```

This command loads `.bashrc` when loading the `.bash_profile`. You will never need to look at the `.bash_profile` file again. From now on you need to only change the `.bashrc` file.

Both of these files probably already exist on your system, if not you can create them as below.

133.6 How can I create the minimal settings?

You may also apply our recommended settings with the following (apply only once!):

```
curl http://data.biostarhandbook.com/install/bash_profile.txt >> ~/.bash_profile
curl http://data.biostarhandbook.com/install/bashrc.txt >> ~/.bashrc
```

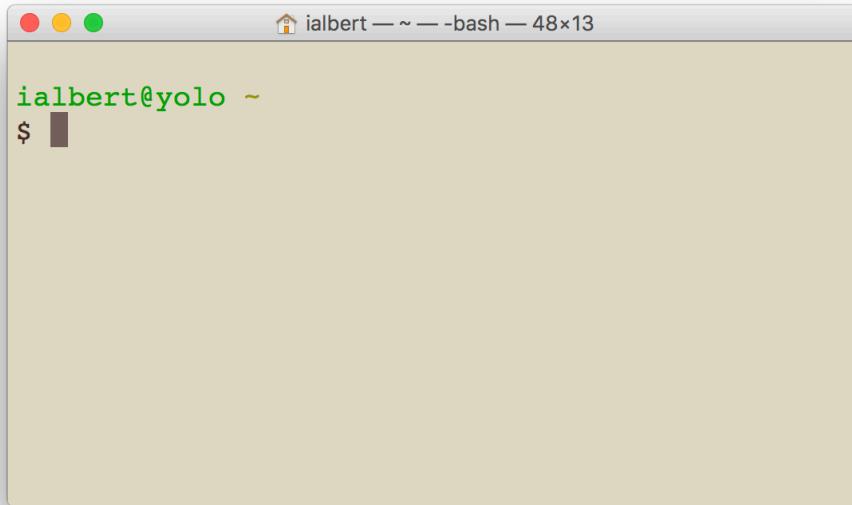
133.7 What should my `.bashrc` file contain?

Minimally we recommend the following:

```
#  
# A minimal BASH profile.  
#  
  
# ON Mac OS uncomment the line below.  
# alias ls='ls -hG'  
  
# On Linux use the following:  
# alias ls='ls -h --color'  
  
# Safe versions of the default commands.  
# Will ask permissions before overwriting files.  
alias rm='rm -i'  
alias mv='mv -i'  
alias cp='cp -i'  
  
# Extend the program search PATH and add the ~/bin folder.  
export PATH=~/bin:$PATH  
  
# Makes the prompt much more user friendly.  
# But I do agree that the command looks a bit crazy.  
export PS1='[\e]0;\w\a]\n[\e[32m]\u@\h \[\e[33m\]\w\[\e[0m\]\n$ '  
  
# This is required for Entrez Direct to work.  
# Disables strict checking of an encryption page.  
export PERL_LWP_SSL_VERIFY_HOSTNAME=0
```

133.8 Can I customize the terminal colors and fonts?

Yes, and please do so. You can vary font sizes, background colors etc. It makes a tremendous difference when reading it for extended periods of time. Here is an easy to read terminal:



133.9 How do I edit my shell profile?

You can use the `nano` editor

```
nano ~/.bash_profile
```

then:

```
nano ~/.bashrc
```

133.10 How do I activate a shell profile?

Changing a shell profile file will not automatically apply the changed settings to terminals that are already open.

New terminals will use the new settings but existing terminal windows need to be instructed to apply the new settings by sourcing the file in question:

```
source ~/.bashrc
```

133.11 Troubleshooting the PATH

It is surprisingly easy to accidentally break everything so thoroughly that not even basic UNIX commands work anymore.

```
$ ls  
-bash: ls: command not found
```

Sooner or later we all do this. Wear it as a badge of honor.

133.11.1 How did I nuke my path?

If you mistype the PATH variable while adding it into the `.bashrc` shell profile you may end up with the unenviable situation of applying an incorrect setting every time your terminal initializes. For example, on a beautiful and sunny morning you'll do a:

```
echo 'export PATH=~/bin:PATH' >> ~/.bashrc
```

It all looks great, but there is a fatal typo there. The net effect of it all will be that even basic system programs that used to work before won't do so anymore.”

```
$ ls  
-bash: ls: command not found
```

Congratulations! You've just **nuked** your PATH. You forgot the \$ sign on the right hand side of the PATH variable. Instead of using the value of variable

named \$PATH you are setting it to be the actual word PATH. Your command should have looked like this:

```
echo 'export PATH=~/bin:$PATH' >> ~/.bashrc
```

First let's look at the bright side – there is probably no command line guru out there that did not do this at one point in their career. Obviously you are well on your way towards “guru-ness”. Rock on!

Jules Winnfield of Pulp Fiction³ explains it best:

The PATH to any righteous tool is beset on all sides by the inequities of UNIX and the tyranny of Environment Variables. Blessed is he who, in the name of charity and good will, shepherds the weak through the valley of darkness, for he is truly his brother's keeper and the finder of lost programs.

133.11.2 How do I fix my nuked path?

Access your terminal. Oh yes, it is still broken. But you can temporarily override the PATH within this terminal with:

```
PATH=/bin:/usr/bin
```

Now edit your .bashrc with nano and remove the lines that cause the trouble:

```
nano ~/.bashrc
```

Now nano is a bit quirky for a text editor. Take it as punishment meted out for clobbering your path. It could be worse. At least you don't have to use vi (But some of us love vi!).

Edit the file, correct the offending line(s), then save the file with **ctrl + O** (that is a capital letter O), then exit the editor with **ctrl + X**.

³https://en.wikipedia.org/wiki/Pulp_Fiction

Open a NEW terminal window and verify that your “corrections” work. In times like these it is very easy to make another error while “fixing” the first one. Rinse and repeat until everything checks out. It will eventually.

Good luck, comrade!

Chapter 134

How do I set up my PATH?

One of the most frustrating issues faced by newcomers to the Unix command line is how to run a program they have just installed. This typically happens if they install a program from source instead of via a global installer.

Suppose you install the program `fastq-dump` but then as you try to run it you could get this:

```
$ fastq-dump  
fastq-dump: command not found
```

The main concept to remember here is that our computers look at only a few locations when trying to find a program. When bash cannot run a program it is because it cannot “see” the program. When we install a new program ourselves, we need to tell our computer where to look for it. There are different ways to do this:

134.1 Solution 1: Use the full program path

Run the program by its fully qualified path (location). First you need to figure out where that is. Go to the location where you installed the software. From that software directory, type `pwd` and that will tell you the full path that you can then use:

```
/Users/ialbert/src/sratoolkit.2.5.2-mac64/bin/fastq-dump
```

We may use a shorter form when the program directory opens from our home directory:

```
~/src/srato toolkit.2.5.2-mac64/bin/fastq-dump
```

If that works, then we can always invoke the program by this full name. It is a bit hard on eyes. For a simpler way, see the next solution.

134.2 Solution 2: Extend the search PATH to include the installation directory.

If we wanted to be able to run the program just by typing its name (say `fastq-dump`) we need to instruct the computer to also look for programs to run in the `/Users/ialbert/src/srato toolkit.2.5.2-mac64/bin/` directory. To do so, we need add this location to the so called *search path* of the shell.

The shell looks at the variable called `PATH` and searches all locations listed in the `PATH` for names of programs to run. What is the current value of the `PATH`? Type the following:

```
echo $PATH
```

This will produce a colon : separated list of all the locations that bash will look in when searching for a program. We can extend this by adding the new location via the following construct:

```
export PATH=~/src/srato toolkit.2.5.2-mac64/bin:$PATH
```

The new `PATH` will be constructed as the new directory + : + old value of the `PATH`. Now we should be able to run the program as

```
fastq-dump
```

Now, the above is a solution, but typing this (and all other) `export` commands we might add every time we open a new terminal is too tedious. We would be better off applying the `PATH` and other settings automatically. To do this, we must take one more step down the system admin rabbit hole. In this case, you need to append this information to the so-called *shell profile* as described in How to set the profile.

134.3 Solution 3: Create shortcuts

There are many cases when we only need to access a single program, and adding the whole directory to the path is tedious and error prone. In this case, the solution is to first create one directory that is being searched by our bash shell, then

create symbolic links (shortcuts) from the programs we want to run in this `~/bin` directory. Since the directory is already on the search path, any new programs added there will also become accessible.

```
# Creates the ~/bin directory
mkdir -p ~/bin

# Add this directory to the search path.
export PATH=~/bin:$PATH

#
# Link fastq-dump program into the ~/bin folder
#
# Syntax: ln -s source destination
#
ln -s ~/src/sratoolkit.2.5.2-mac64/bin/fastq-dump ~/bin/fastq-dump
```

Of course we still need to add `~/bin` to our path. But now we only have to do it once. After that, linking the program into `~/bin` will make it available everywhere. The export command should be:

```
export PATH=~/bin:$PATH
```

To load these settings automatically see the page How to set the profile

Chapter 135

Installing `wgsim` and `dwgsim`

Wgsim (whole genome simulator) is a tool for simulating sequence reads from a reference genome. It is able to simulate diploid genomes with SNPs and insertion/deletion (INDEL) polymorphisms, and simulate reads with uniform substitution sequencing errors. It does not generate INDEL sequencing errors, but this can be partly compensated by simulating INDEL polymorphisms.

Wgsim outputs the simulated polymorphisms, and writes the true read coordinates as well as the number of polymorphisms and sequencing errors in read names. One can evaluate the accuracy of a mapper or a SNP caller with `wgsim_eval.pl` that comes with the package.

Webpage: <https://github.com/lh3/wgsim>

DWGsim is similarly named tool written by Nils Homer inspired by `wgsim`. This tool offers more options and it can generate reads corresponding to different sequencing platforms and produces its results in VCF format. `dwgsim` also has a more streamlined installation instructions.

Webpage: <https://github.com/nh13/DWGSIM>

135.1 Installing DWGSim

On Mac type:

```
brew install dwgsim
```

The conda version of `dwgsim` will roll back samtools to a previous version so on Linux you need to install from source.

135.2 Testing

`dwgsim`

135.3 Information on mutations

The information on the mutations that a read was subjected to is embedded into the read name in the form: `chr1_6181150_6181637_2:0:0_2:0:0_2/1:`

By default, `wgsim` simulates a diploid genome with indels and then simulates reads from the simulated genome without indels. In the example above, the first `2:0:0` indicates that the first read has two sequencing errors and that no substitution and no gaps are involved in the simulated genome. The second `2:0:0` is for the second read. As sequencing errors and substitutions are added separately, they may overlap. You may apply “`-e 0.0`” to disable sequencing errors.

Both the `wgswim` and `dwgsim` distributions come with evaluation scripts `wgsim_eval.pl` and `dwgsim_eval` that can be used evaluate mapping qualities of SAM files where the reads names are encoded as above. The scripts will parse the read name and compare the results to the alignment and, from that establish mapping accuracy.

135.4 Source code installation

Source code install for `wgsim`:

```
cd ~/src
git clone https://github.com/lh3/wgsim.git
cd wgsim
```

```
gcc -g -O2 -Wall -o wgsim wgsim.c -lz -lm
```

```
# Link the program to your bin folder  
ln -s ~/src/wgsim/wgsim ~/bin/wgsim
```

A nicely formatted manual can be accessed via:

```
more ~/src/wgsim/README
```

Source install for dwgsim:

```
cd ~/src  
git clone --recursive https://github.com/nh13/DWGSIM  
cd DWGSIM  
make
```

```
# Link the program to your bin folder  
ln -s ~/src/DWGSIM/dwgsim ~/bin/dwgsim
```

Chapter 136

Jim Kent's bioinformatics utilities

The page <http://hgdownload.cse.ucsc.edu/admin/exe/> location contains a number of utility programs developed by Jim Kent.

Some of these tools are the only “reference” for formats such as bigWig, bigBed etc.

136.1 Installation

You may download the programs directly from the website and place them in your `~/bin` folder (or any other folder in your bash search path). Alternatively you may install tools individually once you find the name that they are distributed under with conda.

Search for all UCSC tools:

```
conda search ucsc
```

will produce a lengthy list of tool. To find the `bedgraph` to `bigwig` converter do a:

```
conda search ucsc | grep bedgraph
```

this will produce the following:

```
ucsc-bedgraphpack
```

```
324
```

```
0 bioconda
```

```
ucsc-bedgraphtobigwig          308          0  bioconda
ucsc-bedgraphtobigwig          377  h8b15c51_0  bioconda
ucsc-bigwigtobedgraph          324          0  bioconda
```

now install the tool of choice:

```
conda install ucsc-bedgraphtobigwig
```

test that the converter was installed correctly:

```
bedGraphToBigWig
```

will produce:

```
bedGraphToBigWig v 4 - Convert a bedGraph file to bigWig format.
usage:
```

```
bedGraphToBigWig in.bedGraph chrom.sizes out.bw
```

```
...
```

136.2 Manual Installation

The general way to install the tools is to navigate to the directory that contains binaries¹ for your platform (Linux/Mac OSX/) then download the tool.

For example `bedGraphToBigWig` on MacOS X:

```
# Change this for Linux
URL=http://hgdownload.cse.ucsc.edu/admin/exe/macOSX.x86_64

# Get the files
curl $URL/bedGraphToBigWig > ~/bin/bedGraphToBigWig
chmod +x ~/bin/bedGraphToBigWig

curl $URL/faToTwoBit > ~/bin/faToTwoBit
chmod +x ~/bin/faToTwoBit
```

¹<http://hgdownload.cse.ucsc.edu/admin/exe/>