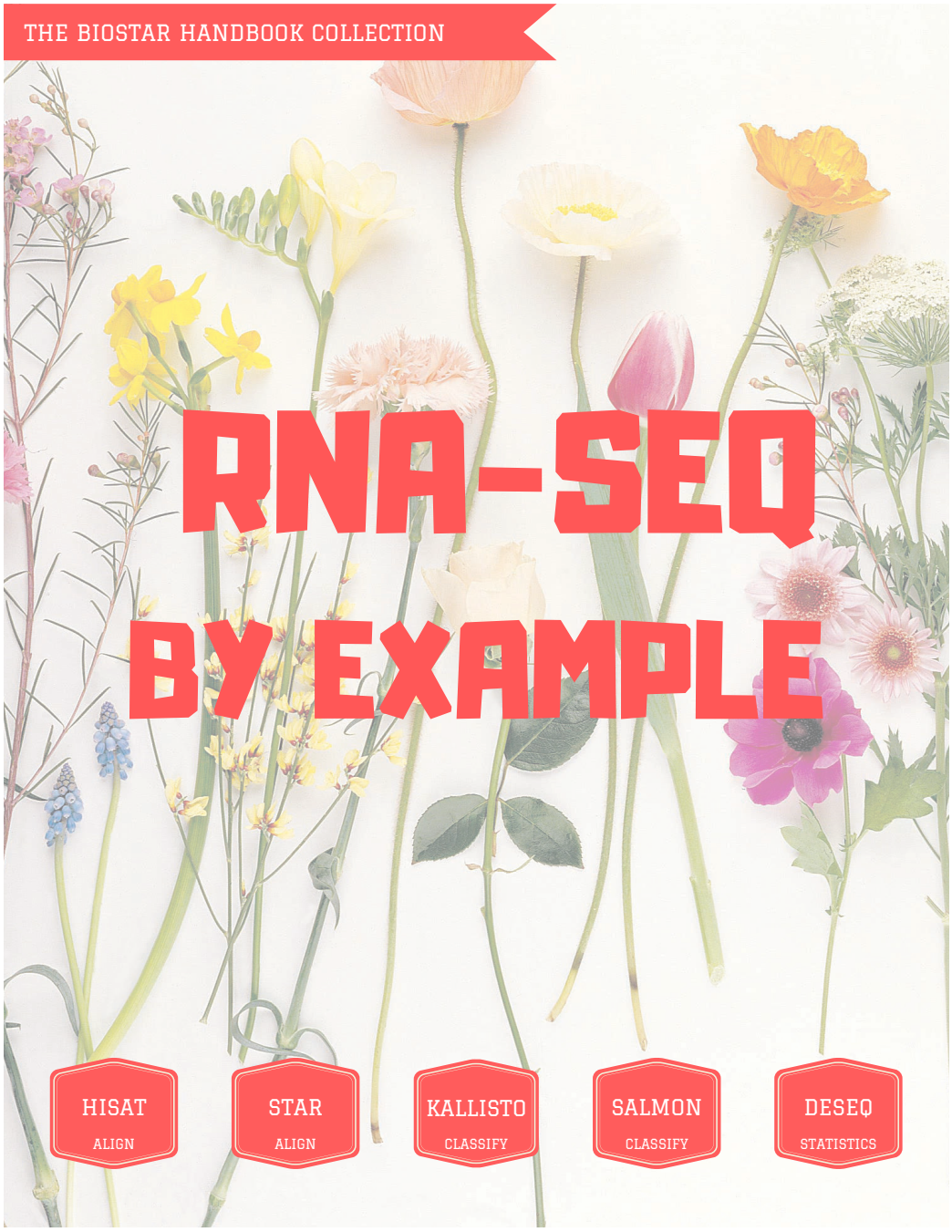


Book updated on February 7, 2020

Contents

Chapter 1

Welcome to the world of RNA-Seq

Last updated on **February 07, 2020**

> **Note: Work in progress. Expected completion February 2020.**

This book is the volume from the Biostar Handbook Collection[1] that introduces readers to RNA-Seq data analysis. The content of the book is regularly updated. Sign up below to be notified of new additions.

1.1 The Biostar Handbook Collection

The Biostar Handbook is being reworked into smaller more manageable units of study.

- **The Biostar Handbook**[2] - The main introduction to Bioinformatics.
- **The Art of Bioinformatics Scripting**[3] - Learn Unix and Bash scripting.

[1] `https://www.biostarhandbook.com`
[2] `https://www.biostarhandbook.com/`
[3] `https://www.biostarhandbook.com/books/scripting/index.html`

- **RNA-Seq by Example**[4] - Learn RNA-Seq data analysis.

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

1.2 Get notified

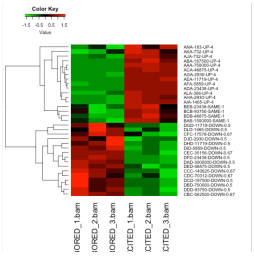
Want to know when new content is published? Subscribe below:

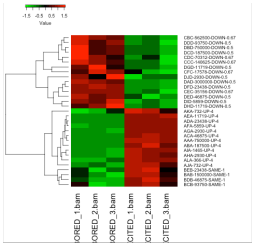
Access to new content is always included in your subscription!

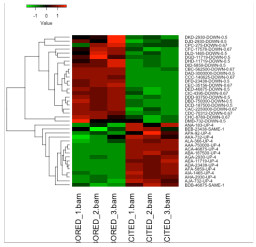
1.3 Mission Impossible RNA-Seq

Not all heroes wear capes! One day you may be called upon to save your world (or your thesis) with a masterful data analysis. Fear not, you came to the right place, we will teach you how to do it, perhaps in a minute:

The above analysis produces a differential expression study with three different methods:







Read on to learn how you can do it as well.

1.4 Typesetting conventions

In the current book long lines of code are **NOT** wrapped. Here is an example of two long lines of code.

```
cat foo.fa | parallel --round-robin --pipe --recstart '>' 'blat -noHead genome1.fa stdin
cat foo.fa | parallel --round-robin --pipe --recstart '>' 'blat -noHead genome2.fa stdin
```

[4] https://www.biostarhandbook.com/books/rnaseq/index.html

Hover over the box and scroll (or use the scroll bar) to see the rest of the line. We are torn between wrapping and not wrapping information. Try as we might, some lines of code will be too long. But if we were to wrap these long lines, the resulting output may look even more confusing. It is best to recognize and deal with the fact that some lines may end up too long to fit.

The PDF and eBook formats cannot be scrolled. We apologize for this limitation and, in general when copying code from the materials we recommend that you use the web version of the book. Unfortunately the PDF and eBook formatters can insert various invisible characters into the output of the code, that, in some circumstances can cause problems.

1.5 How to download the book

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

- RNA-Seq by Example, PDF[5]
- RNA-Seq by Example, eBook[6]

Our books are updated frequently. We recommend accessing each book via the website as the web version will always contain the most recent and up-to-date content. A few times a year we send out emails that describe the new additions.

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 Answer[7] website, the leading resource 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.

[5] `rnaseq-by-example.pdf`
[6] `rnaseq-by-example.epub`
[7] `https://www.biostars.org`

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 do I access my account?

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

Access Your Account

You may change your email or log out via your account page above.

Chapter 2

Basic RNA-Seq concepts

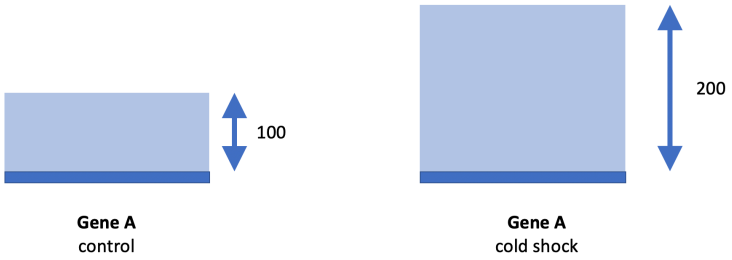
2.1 What is RNA-Seq when simplified to its essence?

Conceptually an RNA-Seq analysis is quite straightforward process. It goes like this:

1. We produce sequencing data from a transcriptome in a "control" state
2. We match sequencing reads to the genome or the transcriptome.
3. We count how many reads align to a 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. We perform the same measurements as before:

1. We produce sequencing data from the transcriptome in the "perturbed" state
2. We match sequencing reads to the genome or the transcriptome.
3. We count how many reads align to the same region (feature), gene `A`. Suppose `200` reads overlap with `Gene A`.



We see that `200` is twice as big as `100`. In an ideal world, we could 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 in the second experiment. We call this doubling the *differential expression*.

In a perfect world we would be done with the "informatics" section of our RNA-Seq analysis.

Now the process of biological interpretation would start. 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 not perhaps we discovered a new role for the gene. And so on.

That's RNA-Seq in a nutshell.

2.2 What makes RNA-Seq challenging?

Of course, the reality is not as simple as described above - the 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 two numbers; we can't even trust the numbers to mean what they are supposed to. 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 that task itself is quite straightforward: compare one number to another and estimate whether their difference is big enough to be considered a valid change.

Take solace and strength from recognizing that the complexity lies in the data

and information management - the analysis process itself does not call upon particularly abstract, mathematical or analytical problem-solving skills.

On the other side, you best understand early on that there is no silver bullet that would make everything always fall into place. There is no magic, always correct approach that will always guarantee a correct data analysis. Your job will always be to respond to what the data tells you. For that, you will need to understand the methods in a little more depth.

2.3 How does RNA sequencing 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 higher levels. In reality, the problems of counting and comparing are more complicated and confounded by several factors - the most important being that the mRNA levels in the cell are almost never static. There is a continuous ebb and flow, each in response to particular stimuli. We need to isolate the signal that corresponds to the function of interest from all the other unrelated changes that might be present in the mRNA concentrations.

2.4 What is the final result of an RNA-Seq analysis?

The result of an RNA-Seq analysis is a *quantification matrix*. For our toy example the file might look like this:

```
 name       control     cold-shock
Gene A        100          200
Gene B        300          120
Gene C        400          530
...
```

This quantification file when processed by a statistical method will be augmented with statistical measures that characterize the changes over the comparisons that we might choose to make. For example, if we were to perform a pairwise comparison between cold-shock and control, the file above will gain more information, expressed as additional columns:

name	control	cold-shock	fold_change	pvalue
Gene A	100	200	2.0	0.000035
Gene B	80	60	0.75	0.234
Gene C	120	180	1.5	0.013
...				

Interpreting this file at the 0.05 statistical significance level would then tell us that `Gene A` and `Gene C` are differentially expressed between `control` and `cold-shock`, whereas `Gene B` is not.

2.5 How many replicates do I need?

While we can make some inferences from a single measurements, the inherent variability of biological phenomena typically requires that we repeat our measurements several times. The process of repeating the entire experimental process, from sample extraction to sequencing is called replication.

A typical recommendation is making minimally three measurements (replicates), the word on the street is that five replicates might be optimal. With replicates, our quantification matrix will look more like this:

name	control1	control2	control3	cold-shock1	cold-shock2	cold-shock3
Gene A	100	103	88	200	188	199
Gene B	300	276	310	120	121	120
Gene C	400	389	459	530	600	700
...						

Above we measured each gene three times in each state `control` and `cold-shock`.

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 5 x 2 x 2 = 20 FASTQ files coming off the sequencing instrument. Your ability to automate processes wil become more important than ever.

2.6 Can one have too many replicates?

Superficially it might seem that the more replicates the better. From a purely mathematical perspective the statement is valid.

Yet there is a human factor to replication: with increasing number of samples the chances of introducing a human error increases. Errors such as mislabeling samples, using protocols incorrectly can have immensely detrimental effects on our ability to interpret data. A single incorrect replicate may severely impact the statistical power of the experiment. Therefore the potential for human error needs to be factored into the cost/benefit evaluation.

Empirically we've observed an inverse correlation between experiment size and data quality. The larger the experiment, the worse the data is. In general we recommend focused studies that strike a balance between the statistical power requirements and human factors.

2.7 What are the main methods for quantifying RNA-Seq data?

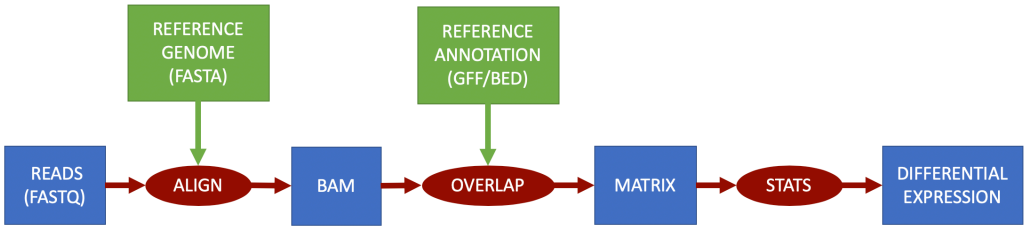
The very first task of an RNA-Seq data analysis is to select the "reference" that your study will use to quantify the changes in gene expression. Two main approaches are in use:

1. Quantify against a genome.
2. Classifying against a transcriptome.

2.8 How does quantifying against a genome work?

When quantifying against a genome case the reference data will be the genome sequence and the genome annotations. You will need a sequence file that contains chromosomal coordinates (FASTA) and an annotation (interval) file (GFF) that describes coordinates on the genome.

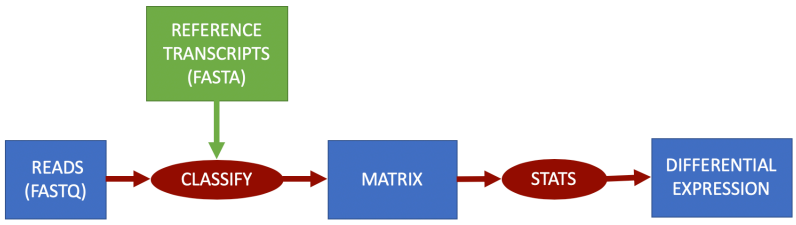
The approach will intersect the alignment files with annotations to produce "counts" that are then filtered to retain statistically significant results. The quantification matrix represents how many times did alignments overlap (intersect) with an interval labeled as part of a transcript.



- The green boxes represent the reference data you need have access to before starting the analysis.
- The blue boxes represent data that you start out with or you will need to create along the way.
- The red ovals represent methods/algorithms that transform the data. These are the software tools you need to employ.

2.9 How does classifying against a transcriptome work?

When quantifying against a transcriptome the reference data will be the FASTA file of each individual transcript. The classification will assign each read to a transcript, resolving ambiguities along the way. The quantification matrix for each transcript will represent the number of times reads were classified as that particular transcript.



- The green boxes represent the reference data you need have access to before starting the analysis.
- The blue boxes represent data that you start out with or you will need to create along the way.
- The red ovals represent methods/algorithms that transform the data. These are the software tools you need to employ.

2.10 What are the advantages of either method?

1. Genome based methods allow us to visualize the data in the context of the entire genome.
2. Genome based methods allow us to discover/validate new transcripts.

On the other hand:

1. Transcriptome based methods are more accurate.
2. Transcriptome based methods have much faster runtimes.

2.11 Which method should I use?

You should perform BOTH types of analyses!

Each method will provide you with a different perspective of the same phenomena. Only by seeing the data in the context of genome as well as transcriptome will you be able to fully appreciate the complexity of the task at hand.

The only reason NOT to do both is when the known information is not available. For example if you have no genome, no annotations or the transcriptome has not yet been built.

2.12 Will there ever be an optimal RNA-Seq analysis method?

As you will see, we present several alternative analysis protocols. You may ask yourself, why isn't there a "best" method?

There is an analogy to a mathematical phenomena called the voting (Condorcet) paradox[1]. This paradox states that when tallying votes for more than two options we may end up with final rankins that are in conflict with the individual wishes of a majority of voters! Moreover the Condorcet paradox also states there is no optimal voting strategy that would avoid all possible conflicts. Basically, another way to say this is that, every voting scheme can, in some (and potentially rare) situations, produce wildly unrealistic results that do not capture the actual intent of the voters.

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 the measurement to its most likely origin ("candidate") transcript. Unsurprisingly the Condorcet paradox will apply, and no matter what method we choose, under some conditions, methods may to evaluate the gene expression incorrectly.

What is the take-home message here? 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 in time. "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.

[1] `https://en.wikipedia.org/wiki/Voting_paradox`

There is always a chance that you'll have a unusual phenomena to quantify for which your initial choice of methods is not optimal. The most crucial skill then is to recognize this situation.

> You will need not just to learn how to perform and 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 that you "just run" is a recipe if not for disaster then, at best for superficial and uninformative conclusions.

2.13 When will I know that I understand RNA-Seq?

Here is a simple guideline:

> You know RNA-Seq when you can readily analyze a dataset with three different methods!

What is essential here is to not be a "one shot wonder". There is this one thing that seems to do RNA-Seq and you can only do that. Life sciences is always a lot more complicated than that. Having a clear understanding of the processes will allow you to make informed decisions. You will see that it is not that difficult at all to use entirely different methods to analyze the same data.

The challenge making an informed decision when biologically interpreting the results.

Part I

RNA-SEQ STEP-BY-STEP

In biology a "gold standard" data is one with known properties. By reproducing the known properties during an analysis, we validate the methods that we use and we demonstrate our skills in applying the methods in the proper way.

In this section of the book we will demonstrate several RNA-Seq data analysis methods through the lens of a "gold standard" data.

Chapter 3

1. Introducing the Golden Snitch

The data that we will analyze was originally created as so-called *spike-in control*, where known abundances of mRNA were added to mRNA exctracted from a cell; then both the artificial spike-in and the real data then sequenced on an Illumina sequencer.

The study was originally published as Informatics for RNA-seq: A web resource for analysis on the cloud[1]. in *PLoS Computational Biology (2015) by Malachi Griffith et al.* While working through the example data I found the re-analysis and especially the validation of the control data to be unexpectedly tedious.

In a moment of inspiration I came up with the idea of transforming the data to represent a more interesting and more realistic scenario. Basically I have "created" a hypothetical organism that corresponds to the control data. Please allow me the following bold statement about the `Golden Snitch`:

Once you understand how the `Golden Snitch` regulation manifests itself in the RNA-Seq data, and your results can prove that behavior as well, you have become an advanced RNA-Seq analyst.

[1] `http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004393`

The `Golden Snitch` is not a toy model! You will see that the same code and approach used to analyze the Golden Snitch data can be used almost unchanged in many other realistic data analysis scenarios.

3.1 What do we know about the Golden Snitch?

Partly mechanical and partly fictitious, the `Golden Snitch` turns out to be the perfect candidate for RNA-Seq data analysis.

What makes it so well suited? Two peculiar features. First, the `Golden Snitch` can only be in two moods:

1. `BORED` when it flies nice and straight.
2. `EXCITED` when it constantly changes direction.

Additionally, the `Golden Snitch` is so magically regulated that, in each state, we know exactly how many transcripts are expressed. The organism is so perfect controlled that even genes could be named in a way that describes their changes for both states. For example a gene might be called:

`ABC-1000-UP-4`

The gene was named as such because in the `BORED` state, each cell has 1000 copies of the transcript corresponding to gene `ABC`. Moreover the name also conveys that in the `EXCITED` state, this same gene is be up-regulated `4` fold compared to the `BORED` state.

Another way to exaplain the naming scheme is that, in the `BORED` state there will be 1000 copies of the `ABC` transcript, whereas in the `EXCITED` state each cell will contain `4000` copies of the `ABC` transcript.

Thus, just by looking at a gene name, we already know how much of it is *inside the cells*. If we were to build a quantification matrix for the *real* mRNA abundance inside the cell our data would look like this:

```
    name            BORED      EXCITED    foldChange
ABC-1000-UP-4       1000        4000          4
ABD-40-DOWN-2         40          20          2
ABE-90-SAME-1         90          90          1
...
```

Across the two states each gene can only be either `UP`, `DOWN` regulated or stay the `SAME`. Note how the quantification matrix above represents what "really" happens in the cell.

It remains to be seen how well results from an RNA-Seq data can reproduce the expected numbers or ratios. The gene naming convention will allow you to spot check any intermediate result and evaluate how well the analysis is going.

3.2 What is the objective of this section?

Your job will be to demonstrate how well, and under what circumstances, can you reproduce the known regulatory behavior when starting with RNA-Seq data.

As you study the `Golden Snitch` and unlock its secrets, you will begin to understand the bigger picture of how to think about RNA-Seq in general.

So let's get the data, and see what it is all about.

Chapter 4

2. Understand your reference

The very first task of an RNA-Seq data analysis is to select the "reference" that your study will use to quantify the gene expression. As we described in the previous chapter there are two analyis paradigms:

1. Quantify against a genome
2. Classify against a transcriptome

As you recall we also recommended that you **do both**.

Your first step will be to *identify* and obtain the reference files for your study. Typically there are data distribution sites distributing reference files. There in only one location (for now ;-) for the `Golden Snitch`, the reference is available from:

- `http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar.gz`

4.1 What is my very first step?

Understand your reference data! What properties does be the "reference" that you will use to quantify your data have?

1. How big is the reference?
2. How many chromosomes does it have?
3. What are these chromosomes called? How long is each?
4. How many features do the annotation have?

5. What types of features are listed? …

etc. Typically, for people working on the same organism, they would only need to investigate their genome once, the refer to this information later when needed.

The process of understanding your genome is the first step in the confidence building process that moves you from being a mere passenger tagging along and observing, to becoming the driver.

4.2 What is the Golden Snitch's genome like?

> First, do yourself a favor, create a subdirectory and work in that. I have seen countless well intentioned individuals struggle to make sense of their methods because their home directories were overrun with hundreds of files of various origins. Make a new separate directory for every new project!

I recommend making a work directory, then creating a new subdirectory in that, like so:

```
# Activate your environment
conda activate bioinfo

# Make a directory for this analysis
mkdir -p work/golden

# Switch to the directory
cd work/golden
```

We assume that all commands forward on are run in the `work/golden` directory that you have just created and that the `bioinfo` environment is activated. Now let's get the reference data and investigate it.

```
# Download the reference genome.
wget http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar.gz

# Unpack the reference genome.
tar xzvf golden.genome.tar.gz
```

At this point you will see a `refs` folder that has been created. List the contents of it:

```
$ ls -l refs/
```

the command prints:

```
-rw-r--r--  1 ialbert  staff    12K Feb  5 12:51 features.gff
-rw-r--r--  1 ialbert  staff   127K Feb  5 12:51 genome.fa
-rw-r--r--  1 ialbert  staff    83K Jan 24 14:06 transcripts.fa
```

We can see that we have a genome, features and transcripts. For other organisms the naming of files might not be as obvious and clear cut.

4.3 How many sequences in the genome?

Evaluate the FASTA files:

```
seqkit stats refs/*.fa
```

will print:

```
file            format  type  num_seqs  sum_len  min_len  avg_len  max_len
refs/genome.fa     FASTA  DNA        1  128,756  128,756  128,756  128,756
refs/transcripts.fa FASTA  DNA       92  82,756    273   899.5   2,022
```

The genome has a single chromosome of size of `128,765 bp`. There are `92` transcripts. The shortest transcript is `273bp` the longest is `2022bp`. The total transcript size is `92 x 899 = 82,708` basepairs.

4.4 What does the genome file look like?

```
cat refs/genome.fa | head -5
```

prints:

```
>Golden full genome, version 2020, Yo Ho Ho!
GCATTTTGAAAATTCTATGGAAGAGCTAGCATCTCTGACGAAAACAGCAGACGGAAAAGTACTGACCAGCGTCACACAAA
AACGGAACAGGGCTGACGCCGCTACATATATAGGAAAAGGGAAGGTAGAAGAGCTGAAGGCACTCGTGGAAGAGCTTGAA
GCTGATCTCCTCATCTTTAATGATGAACTGTCGCCAAGTCAGCTGAAGTCATTGGCAACAGCAATTGAAGTGAAGATGAT
TGACCGCACGCAATTGATATTAGATATTTTTGCAAAGCGGGCGAGAACGAGAGAAGGCAAACTTCAAATTGAGCTGGCTC
```

The genome is a `FASTA` file, the chromosomal sequence is named `Golden`

4.5 What is the annotation file?

```
cat refs/features.gff | head -5
```

prints:

```
##gff-version 3
Golden ERCC   exon   1  1060   0  +  .  gene_name=AAA-750000-UP-4; gene_id=AAA-
Golden ERCC   exon   1560   2083   0  +  .  gene_name=ABA-187500-UP-4; gene_id=
Golden ERCC   exon   2583   3616   0  +  .  gene_name=ACA-46875-UP-4; gene_id=A
Golden ERCC   exon   4116   5138   0  +  .  gene_name=ADA-23438-UP-4; gene_id=A
```

how many exons (this is an approximate count):

```
cat refs/features.gff | grep exon | wc -l
```

prints:

```
92
```

We note that the IDs in the feature file have the same name: `Golden` as in the genome. So the two names will match.

4.6 What do the transcripts look like?

```
cat refs/transcripts.fa | head -5
```

prints:

```
>AAA-750000-UP-4
GCATTTTGAAAATTCTATGGAAGAGCTAGCATCTCTGACGAAAACAGCAGACGGAAAAGTACTGACCAGCGTCACACA
AACGGAACAGGGCTGACGCCGCTACATATATAGGAAAAGGGAAGGTAGAAGAGCTGAAGGCACTCGTGGAAGAGCTTG
GCTGATCTCCTCATCTTTAATGATGAACTGTCGCCAAGTCAGCTGAAGTCATTGGCAACAGCAATTGAAGTGAAGATG
TGACCGCACGCAATTGATATTAGATATTTTTGCAAAGCGGGCGAGAACGAGAGAAGGCAAACTTCAAATTGAGCTGGC
```

We can see the naming schemes. Print out more gene names:

```
cat refs/transcripts.fa | grep ">" | head
```

prints:

```
>AAA-750000-UP-4
>ABA-187500-UP-4
>ACA-46875-UP-4
>ADA-23438-UP-4
>AEA-11719-UP-4
>AFA-5859-UP-4
>AGA-2930-UP-4
>AHA-2930-UP-4
>AIA-1465-UP-4
>AJA-732-UP-4
```

From here we can see that in this genome gene `AAA` expresses with `750,000` copies in the `BORED` state, whereas gene `AJA` expresses with 732 copies. Basically, at any time there will `1000x` more transcripts for the `AAA` than for `AJA`. Make a mental note of that.

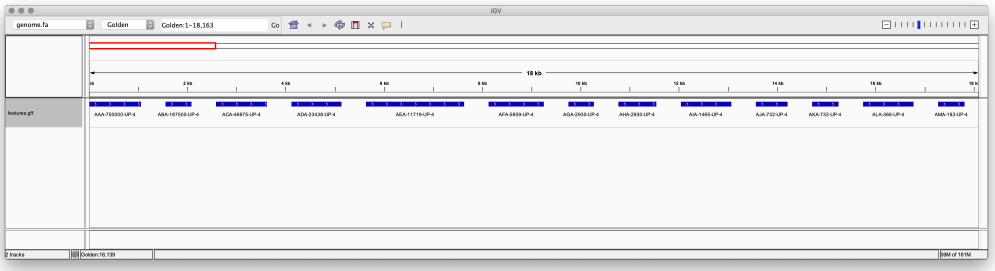
Do also note how both the `AAA` and the `AJA` genes change the same amount 4 fold! The magnitude of the change (the ratio) across the states will be the same, the absolute value of the concentration of each transcript will, however be massively different.

It makes sense that it will be more challenging to properly quantify `AJA` than `AAA`. As a matter of fact it is quite possible that there is a limit under which we cannot detect changes, perhaps we can't detect anything at levels of `AFA` and below.

In a realistic scenario we would not know beforehand which transcript might be lowly expressed. But we need to approach the analysis with the expectation that we will only be able detect transcripts above a certain expression abundance threshold.

4.7 Visualize your reference file

Load both the genome and annotations into IGV.



4.8 Optional step: align your transcripts against the reference

You may want to validate your transcripts. Will they match the genome as you assume they would?

```
# The index to the genome
IDX=refs/genome.fa

# Build the index
hisat2-build $IDX $IDX

# Create the transcripts.
hisat2 -x $IDX -f -U refs/transcripts.fa | samtools sort > refs/transcripts.bam

# Index the BAM file
samtools index refs/transcripts.bam
```

Visualizing the generated alignment file makes our hearts overflow with joy: that the sequences do indeed correspond to the features in the GFF file.

4.9 What is the next step?

We have a good graps of what our references contain. Now we'll need to get same level of confidence regarding the RNA-Seq data. Lets Understand the sequencing reads.

Chapter 5

3. Understand the sequencing reads

The next step is to get a solid grasp of the properties and layout of your sequencing reads. If the data is stored in SRA you can use `fastq-dump` for other cases your data will be stored in files distributed from various locations. In our case the data is located at:

- `http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz`

5.1 What information do I need to know?

You need to know how many reads were measured, what the experimental layout is and get a sense of how well the experiment worked. You may have access to various sources of information that describe the experiment. In general we found that documentation often lacks details and describes data incompletely. As a rule, you can only continue on by figuring out by yourself, from the files themselves of what they contain. Thus we will follow that strategy:

```
# Download the data
wget http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz

# Unpack the data
```

```
tar zxvf golden.reads.tar.gz

# List the content of the current directory.
ls -l

# List the contents of the reads directory
ls -l reads
```

on my system it prints:

```
-rw-r--r--  1 ialbert  staff    26M Feb  5 11:56 BORED_1_R1.fq
-rw-r--r--  1 ialbert  staff    26M Feb  5 11:56 BORED_1_R2.fq
-rw-r--r--  1 ialbert  staff    32M Feb  5 11:56 BORED_2_R1.fq
-rw-r--r--  1 ialbert  staff    32M Feb  5 11:56 BORED_2_R2.fq
-rw-r--r--  1 ialbert  staff    29M Feb  5 11:56 BORED_3_R1.fq
-rw-r--r--  1 ialbert  staff    29M Feb  5 11:56 BORED_3_R2.fq
-rw-r--r--  1 ialbert  staff    55M Feb  5 11:56 EXCITED_1_R1.fq
-rw-r--r--  1 ialbert  staff    55M Feb  5 11:56 EXCITED_1_R2.fq
-rw-r--r--  1 ialbert  staff    37M Feb  5 11:56 EXCITED_2_R1.fq
-rw-r--r--  1 ialbert  staff    37M Feb  5 11:56 EXCITED_2_R2.fq
-rw-r--r--  1 ialbert  staff    46M Feb  5 11:56 EXCITED_3_R1.fq
-rw-r--r--  1 ialbert  staff    46M Feb  5 11:56 EXCITED_3_R2.fq
```

Run a statistics on the reads:

```
seqkit stats reads/*.fq
```

the command prints:

```
file           format  type  num_seqs    sum_len  min_len  avg_len  max_len
reads/BORED_1_R1.fq   FASTQ  DNA   112,193  11,219,300    100    100    100
reads/BORED_1_R2.fq   FASTQ  DNA   112,193  11,219,300    100    100    100
reads/BORED_2_R1.fq   FASTQ  DNA   137,581  13,758,100    100    100    100
reads/BORED_2_R2.fq   FASTQ  DNA   137,581  13,758,100    100    100    100
reads/BORED_3_R1.fq   FASTQ  DNA   123,093  12,309,300    100    100    100
reads/BORED_3_R2.fq   FASTQ  DNA   123,093  12,309,300    100    100    100
reads/EXCITED_1_R1.fq FASTQ  DNA   237,018  23,701,800    100    100    100
reads/EXCITED_1_R2.fq FASTQ  DNA   237,018  23,701,800    100    100    100
reads/EXCITED_2_R1.fq FASTQ  DNA   158,009  15,800,900    100    100    100
reads/EXCITED_2_R2.fq FASTQ  DNA   158,009  15,800,900    100    100    100
reads/EXCITED_3_R1.fq FASTQ  DNA   196,673  19,667,300    100    100    100
```

```
reads/EXCITED_3_R2.fq FASTQ  DNA   196,673 19,667,300    100    100    100
```

5.2 What is the summary so far?

1. The experiment captures data from two states: `BORED` and `EXCITED`.
2. Three replicates were measured for each state: `1`, `2`, `3`
3. The sequencing run is paired end designated by `R1` and `R2`. The read length is 100bp.
4. The measurements are distributed somewhat unevenly over the samples.
5. Twice as much data was collected for sample `EXCITED_1` than for `BORED_1`

We have previously identified the genome and transcriptome sizes to be 128,765 and 82,708 respectively. If the coverage were uniform then the expected coverage of the worst covered sample `BORED_1` would be:

```
112,193 * 100 / 128,765 = 87x
```

or, we consider the transcriptome alone as target:

```
112,193 * 100 / 82,708 = 135x
```

Thus if the expressions were uniform (every gene expressed at the same level), each transcript would be covered at `135x`.

Now we do not expect that gene expression is uniform, still the question is how much will the expression deviate from the `135x`. As a matter of fact, is the `135x` even good as a first approximation?

5.3 Generate the root names

As we pointed out in the Art of Bioinformatics Scripting[1] every analysis needs to operate on so called "root" names rather than "patterns" matching files. Never run any method on patterns like `*_R1.fq` even if initially it seems to work. Don't allow the computer to "match" and discover files on its own by matching patterns. Pain and suffering lies that way - all it takes is one

[1] `https://www.biostarhandbook.com/books/scripting/index.html`

extra, or out of order or missing file to potentially render the entire analysis incorrect, often without even a hint that something went amiss. The most serious analysis errors are caused not by applying the wrong methods, but by plugging the wrong data in the wrong place.

What are the root ids for our problem at hand?

We see that the reads pairs `R1`, `R2` will need to be handled together. Beyond that we have the states and replicate. Appropriate root ids would be `HAPPY_1`, `HAPPY_2` etc. We can write out these root ids by hand, or generate then with `parallel` like so:

```
parallel -j 1 echo {1}_{2} ::: BORED EXCITED ::: 1 2 3
```

will print:

```
BORED_1
BORED_2
BORED_3
EXCITED_1
EXCITED_2
EXCITED_3
```

We set the `-j 1` flag above (number of jobs run at the same time) to limit `parallel` to executing only one job at a time (in sequential order). Otherwise, some jobs might finish a faster, and might produce ids in a different order. The order would not make the ids incorrect, just potentially confusing.

Put these ids into a file:

```
parallel -j 1 echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids
```

We can now continue on with analyzing the data.

Chapter 6

4. Alignment based RNA-Seq

6.1 Index your reference genome

First we need ot index the reference genome. Needs to be done only once and the index can be reused for the same organism.

```
# The indexed reference genome.
IDX=refs/genome.fa

# Build the genome index.
hisat2-build $IDX $IDX

# Index the reference genome with samtools.
samtools faidx $IDX
```

If you haven't done already create the root ids BORED_1, BORED_2 … by hand or with:

```
parallel -j 1 echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids
```

Does it matter what you list first? Yes.

When we make pairwise comparison we compare the second category to the first. Our differential expressions will be expressed SECOND/FIRST. While you can of course change the order later, it is best if you initial listing already captures the correct order.

In this project we want to know the find the changes in the `HAPPY` state relative to the `BORED` state.

6.2 Generate the alignments

Since we will generate a `BAM` alignment for each sample, we'll make a directory to store these alignments and other related files:

```
# Create the BAM folder.
mkdir -p bam

# Align the FASTQ files to the reference genome.
cat ids | parallel "hisat2 -x $IDX -1 reads/{}_R1.fq -2 reads/{}_R2.fq -S bam/{}.sam"

# Sort each SAm into a BAM file.
cat ids | parallel "samtools sort bam/{}.sam > bam/{}.bam"

# Index each BAM file.
cat ids | parallel  "samtools index bam/{}.bam"
```

The alignments have all been created, we can see the resulting files with:

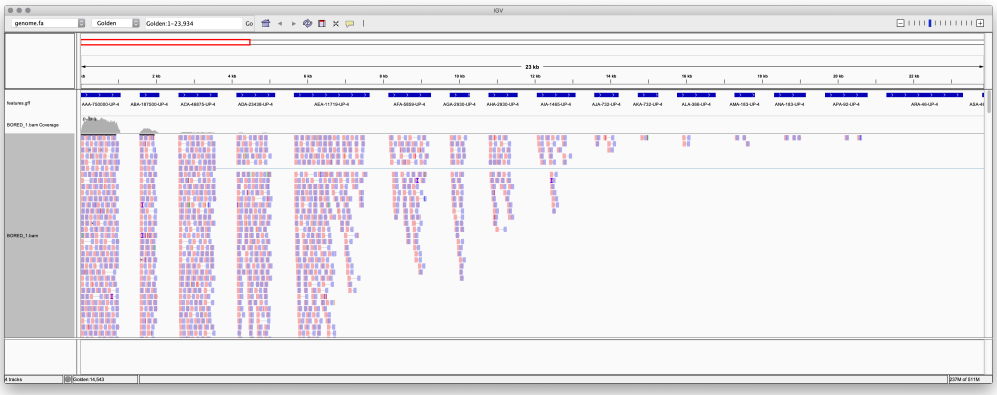
```
ls -l bam
```

it will print:

```
-rw-r--r--  1 ialbert  staff    12M Feb  5 13:06 BORED_1.bam
-rw-r--r--  1 ialbert  staff   352B Feb  5 13:06 BORED_1.bam.bai
-rw-r--r--  1 ialbert  staff    75M Feb  5 13:06 BORED_1.sam
-rw-r--r--  1 ialbert  staff    16M Feb  5 13:06 BORED_2.bam
-rw-r--r--  1 ialbert  staff   352B Feb  5 13:06 BORED_2.bam.bai
-rw-r--r--  1 ialbert  staff    92M Feb  5 13:06 BORED_2.sam
...
```

6.3 Visualize the alignments

Let's visualize the file in IGV:



6.4 Create coverage data

Now, if you do load the data, you will notice a problem. Even if you only load in a single BAM file it takes quite a bit of time to have it appear in IGV. Moreover the same loading process need to take place after each zooming or panning operation.

It is super tedious to use, frankly somewhat of an embarrassment for the Broad Institute[1] the maintainers of the software. Really? Is this how you are curing cancer? Is this the best tool that hundreds of millions of dollars in public and private funding is able to provide? A genomic browser that chokes on the Gold Snitch data?

Of course there is a workaround. There always is. Sometimes, I feel bioinformatics itself is the journey of working around problems that shouldn't exist in the first place. Since what we primarily care about is the "coverage" rather than individual alignments we can turn the BAM file into a so called `bigWig` file - ah yes, I wish I was joking.

The `bigWig` format is the brain child of a celebrated bioinformatician Dr. Jim Kent[2] of the Human Genome fame. Among the many notable accomplishments Dr. Kent is also the creator of the UCSC genome browser a tool of high utility. Alas he is also the "inventor" of ridiculous data formats such as `BED` or `bigWig`, formats that are little more than band-aid solutions to much

[1] `https://www.broadinstitute.org/`
[2] `https://en.wikipedia.org/wiki/Jim_Kent`

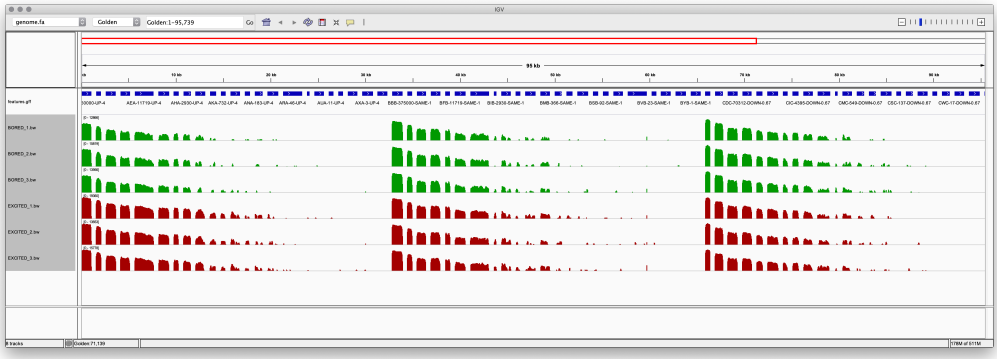
larger, more fundamental and still unaddressed problems of biological data representation.

Ludicrous or not, `bigWig` is what must use to work around the limitations of IGV. Oh well, let's make a `bigWig`:

```
# First we turn the each BAM file into BedGraph coverage.
cat ids | parallel "bedtools genomecov -ibam  bam/{}.bam -split -bg  > bam/{}.bg"

# Then convert the BedGraph coverage into BigWig coverage.
cat ids | parallel "bedGraphToBigWig bam/{}.bg  ${IDX}.fai bam/{}.bw"
```

The resulting `bigWig` files will have `*.bw` extension and ar placed in the `bam`. BigWig files will load up much faster. Below I have loaded all samples, resized the tracks, colored them by samples. I have also turned on logarithmic and automatic scaling. The resulting browser track is quite informative:



Note that once you make a useful visualization with IGV you can save the IGV session into a file, that can be reloaded later.

6.5 What can you learn from the visualization?

Try to answer the following questions from the visualization

1. What are the minimal gene expression levels that can be detected at all?

2. What are the minimal gene expression levels that could be used to detect a 4 fold change?
3. Is the `135x` average coverage computed in the previous chapter a reasonable approximation for average gene expression?
4. Does the data support the expected gene regulation?

6.6 What is our quantification matrix?

The goal of a RNA-Seq method is to produce the quantification matrix. In an alignment based RNA-Seq the process requires intersecting the BAM alignment file with the intervals listed in an annotation file.

The intersection process, often called "feature counting" produces a count of how many alignments overlap with each feature of the file. For example, take `FEATURE A` as pictured below. Assume that each dashed line is an alignment. How many features overlap with `FEATURE A`?

```
-------    -------  -------
   ------  ---------  ---------
     -------  ---------
 ------- --------- ---------
      |<-- FEATURE A -->|
```

As you can probably noted yourself, the answer depends what the word "overlap" means precisely. Should any amount of overlap suffice? Then the `11` alignments overlap with the feature. Should the alignment be completely inside the feature? Then the count is only `4`. Should at least 50% of the alignment cover the feature? Now the count is another value. There is no universally correct count, depending on the situation one or the other count would be more appropriate. Most tools can be configured to apply the various strategies. Investigate your data to identify best strategies. By default we will use the "any amount of overlap" counting strategy.

6.7 How to count the features?

We recommend using the tool featureCounts[3], many scientists seem to swear by htseq-counts[4][htseq-counts[5]. A typical invocation of `featureCounts` would be:

```
# Run the featureCounts program to summarize the number of reads that overlap with feature
featureCounts -p -a refs/features.gff -o counts.txt bam/BORED_1.bam

# Show the first five lines of the counts.txt file.
cat counts.txt | head -5
```

It will produce the file counts with the following content:

```
# Program:featureCounts v1.6.4; Command:"featureCounts" "-p" "-a" "refs/features.gff" "-
Geneid  Chr Start   End Strand  Length  bam/BORED_1.bam
AAA-750000-UP-4 Golden  1   1060    +   1060    8103
ABA-187500-UP-4 Golden  1560    2083    +   524 979
ACA-46875-UP-4  Golden  2583    3616    +   1034    505
```

A nice feature of featureCounts is that we can list multiple BAM files

```
featureCounts -p -a refs/features.gff -o counts.txt bam/BORED_1.bam bam/BORED_2.bam bam/
```

While in general we try to avoid pattern matching, in this case we can a save a lot of typing by letting the shell fill in the file names as long as we are quite certain these are in the correct order:

```
featureCounts -p -a refs/features.gff -o counts.txt bam/BORED_?.bam bam/EXCITED_?.bam
```

The question mark will expand to all possible combinations, the simplest way to test out what a shell metacharacter will do is to use it in different contexts:

```
ls bam/BORED_?.bam
```

prints:

```
bam/BORED_1.bam bam/BORED_2.bam bam/BORED_3.bam
```

running the command mentioned before:

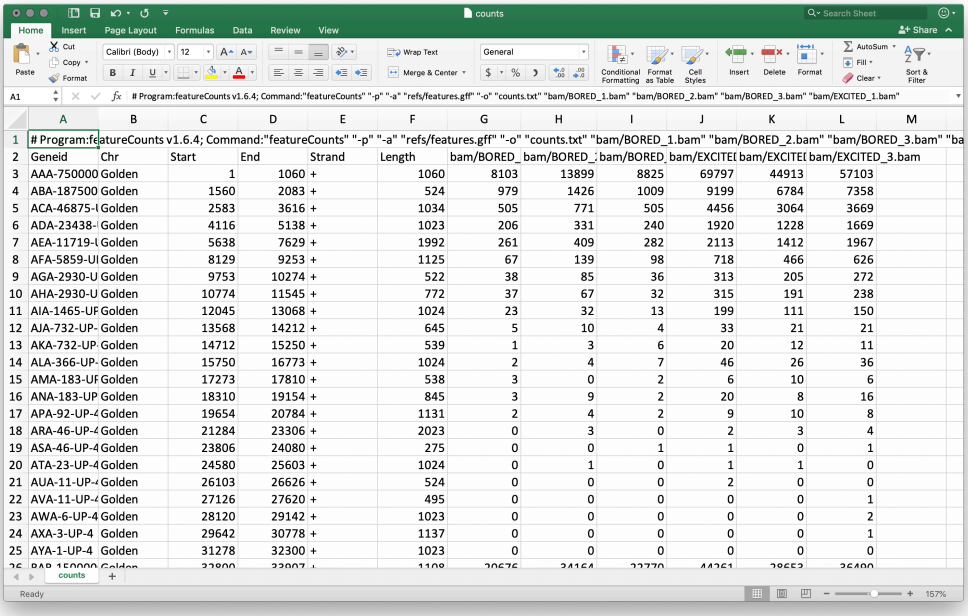
```
featureCounts -p -a refs/features.gff -o counts.txt bam/BORED_?.bam bam/EXCITED_?.bam
```

[3] http://bioinf.wehi.edu.au/featureCounts/

[4] https://htseq.readthedocs.io/en/release_0.11.1/count.html

[5] https://htseq.readthedocs.io/en/release_0.11.1/count.html

will create the file text file `counts.txt` that when opened in Excel will contain:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	# Program:featureCounts v1.6.4; Command:"featureCounts" "-p" "-a" "refs/features.gff" "-o" "counts.txt" "bam/BORED_1.bam" "bam/BORED_2.bam" "bam/BORED_3.bam" "ba												
2	Geneid	Chr	Start	End	Strand	Length	bam/BORED_	bam/BORED_	bam/BORED_	bam/EXCITE	bam/EXCITE	bam/EXCITED_3.bam	
3	AAA-750000	Golden	1	1060	+	1060	8103	13899	8825	69797	44913	57103	
4	ABA-187500	Golden	1560	2083	+	524	979	1426	1009	9199	6784	7358	
5	ACA-46875-(	Golden	2583	3616	+	1034	505	771	505	4456	3064	3669	
6	ADA-23438-	Golden	4116	5138	+	1023	206	331	240	1920	1228	1669	
7	AEA-11719-(	Golden	5638	7629	+	1992	261	409	282	2113	1412	1967	
8	AFA-5859-UI	Golden	8129	9253	+	1125	67	139	98	718	466	626	
9	AGA-2930-U	Golden	9753	10274	+	522	38	85	36	313	205	272	
10	AHA-2930-U	Golden	10774	11545	+	772	37	67	32	315	191	238	
11	AIA-1465-UF	Golden	12045	13068	+	1024	23	32	13	199	111	150	
12	AJA-732-UP-	Golden	13568	14212	+	645	5	10	4	33	21	21	
13	AKA-732-UP	Golden	14712	15250	+	539	1	3	6	20	12	11	
14	ALA-366-UP-	Golden	15750	16773	+	1024	2	4	7	46	26	36	
15	AMA-183-UF	Golden	17273	17810	+	538	3	0	2	6	10	6	
16	ANA-183-UP	Golden	18310	19154	+	845	3	9	2	20	8	16	
17	APA-92-UP-4	Golden	19654	20784	+	1131	2	4	2	9	10	8	
18	ARA-46-UP-4	Golden	21284	23306	+	2023	0	3	0	2	3	4	
19	ASA-46-UP-4	Golden	23806	24080	+	275	0	0	1	1	0	1	
20	ATA-23-UP-4	Golden	24580	25603	+	1024	0	1	0	1	1	0	
21	AUA-11-UP-4	Golden	26103	26626	+	524	0	0	0	2	0	0	
22	AVA-11-UP-4	Golden	27126	27620	+	495	0	0	0	0	0	1	
23	AWA-6-UP-4	Golden	28120	29142	+	1023	0	0	0	0	0	2	
24	AXA-3-UP-4	Golden	29642	30778	+	1137	0	0	0	0	0	1	
25	AYA-1-UP-4	Golden	31278	32300	+	1023	0	0	0	0	0	0	

What does this file tell us? Let's read out the first line:

> The feature named `AAA-750000-UP-4` located on the chromosome named `Golden` between positions `1` and `1060` on the `+` strand having a length of `1060` had `8103` overlapping alignments in the file `BORED_1.bam`, `13899` overlapping alignments in the file `BORED_2.bam` … and so on.

This file is the quantification matrix that will be processed further with statistical methods. You can consider it as the primary output of the qunatification method itself. Different statistical methods may be run on the matrix.

6.8 There is even more to counting

We you run `featureCounts` with default parameters, numerous decisions are made on your behalf:

1. Which types of features are used to count ovlaps (default=exons)
2. How are counted features grouped into the same (default=gene_name attribute)
3. How much overlap is required to consider an alignment overlapping with a feature (default=1)
4. Should individual read alignments or read-pairs be counted(default=no, setting the `-p` will count read pairs) 1 … and so on

Depending on the use case you may need to override one or more of these parameters. See all possible options on the documentation page[6] or by invoking the `-h` flag:

```
featureCounts -h
```

If you do override any parameters make a note of that in your script, like so:

```
# Count read pairs that overlap on the same strand.
featureCounts -p -S 1 ...
```

6.9 What is the next step

You can now either visit the page on how to compute the same quantification using classification or read how to perform differential expression on the quantification matrix.

[6] `http://bioinf.wehi.edu.au/featureCounts/`

Chapter 7

5. Classification based RNA-Seq

```
# Set the shortcuts to reference and index.
REF=refs/transcripts.fa
IDX=refs/transcript.idx

# Index the genome with kallisto.
kallisto index -i $IDX $REF
```

If you haven't done already create the root ids BORED_1, BORED_2 … by hand or with:

```
parallel -j 1 echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids
```

Run the kallisto classification:

```
# Make a directory for the results
mkdir -p output

# Run Kallisto to classify the reads.
cat ids | parallel kallisto quant -i $IDX -o output/{} reads/{}_R1.fq reads/{}_R2
```

Infuriatingly the `kallisto` developers nonchalantly commit quite a major data naming sin. The files that kallisto produces are all named the same `abundance.tsv`! We can only keep them separate by putting them into different folders! Thus when you run a sample called `foo` instead of creating an output called `foo.txt` you will be given a directory called `foo` in which the

result will be called `abundance.tsv`. Look at this abomination that kallisto has just created for us:

```
find . -name 'abundance.tsv'
```

The same file name in differently named directories:

```
./output/BORED_2/abundance.tsv
./output/BORED_3/abundance.tsv
./output/EXCITED_3/abundance.tsv
./output/EXCITED_2/abundance.tsv
./output/BORED_1/abundance.tsv
./output/EXCITED_1/abundance.tsv
```

If you really want to know who to blame, it was a tool called `Tophat` that first introduced this dumb behavior … every alignment file produced by `Tophat` was called `accepted.bam`, irritating many generations of scientists. `Tophat` as an aligner has be long superseded by more accurate and more performant algorithms, but the bad habits introduced with TopHat endure far longer.

Encoding data identity information in the file path is a terrible idea, one we dearly hope won't catch on even more. Above we now have 6 files, all called 'abundances.tsv' in different folders. It is exceedinly easy to using the wrong file without realizing it. Each abundance file has the following structure:

```
cat output/BORED_1/abundance.tsv | head -5
```

prints:

```
target_id       length   eff_length   est_counts    tpm
AAA-750000-UP-4  1059     887.222         8103      51802.2
ABA-187500-UP-4  523      351.222          979      15810.2
ACA-46875-UP-4   1033     861.222          505      3325.92
ADA-23438-UP-4   1022     850.222          206      1374.26
```

you can read more about `effective lengths` and `tpm` on the RNA-Seq terminology page. To make all our statistical methods work the same way we want to turn the files above into a single counts file similar to what we obtain when we perform an Alignment based RNA-Seq approach.

Turning the six files into a single count file, where each column is named by sample is surprisingly challenging to do at the command line. Of course we could always copy paste by hand but that breaks the automation. Copy

pasting is an error prone and tedious process, it takes away our ability to quickly redo an analysis.

It goes to show how ill an advised approach compounds the problems and make analyses unnecessarily difficult. When the kallisto developers chose their myopic approach, they have also taken away the ability of people to analyze the data without making use of yet another piece of code.

We wrote a Python script that we believe is both simple and generic enough for all use-cases:

```
# Download the custom script to combine kallisto outputs.
curl http://data.biostarhandbook.com/books/rnaseq/code/combine.py > ~/bin/combi

# Make the script executable.
chmod +x ~/bin/combine
```

We can now combine the abundances into a single count matrix with:

```
cat ids | combine output > counts.txt
```

it produces an output such as:

Chapter 8

6. The differential expression

In this chapter we will focus on the practical aspects of using statistical methods. We do also have a more expansive discussion on the role of statistics when processing RNA-Seq data.

8.1 Prepare the environment

If you haven't already done so install the statistical packages into into your environment with the command:

```
# Activate the bioinformatics environment.
conda activate bioinfo

# Install the statistical packages.
URL=http://data.biostarhandbook.com/books/rnaseq/code/install-conda.txt
curl $URL | xargs conda install -y
```

For this book we have written scripts that make use of several, different R based methods. We have devoted a substantial effort to standardizing both the usage and the results produced by these different methods.

To make these scripts universally accessible and simpler to use, just put them into the '`~/bin` folder and make them executable like so:

```
# Download the scripts into the ~/bin folder.
curl http://data.biostarhandbook.com/books/rnaseq/code/deseq1.r > ~/bin/deseq1.r
```

```
curl http://data.biostarhandbook.com/books/rnaseq/code/deseq2.r > ~/bin/deseq2.
curl http://data.biostarhandbook.com/books/rnaseq/code/edger.r > ~/bin/edger.r
curl http://data.biostarhandbook.com/books/rnaseq/code/heatmap.r > ~/bin/heatma

# Make them executable
chmod +x ~/bin/*.r
```

Test that the code works by typing:

```
deseq1.r
```

it should print something similar:

```
Error: The experimental design must be specified as NxM
Execution halted
```

You may also download each script individually from the following locations:

- `http://data.biostarhandbook.com/books/rnaseq/code/deseq1.r`
- `http://data.biostarhandbook.com/books/rnaseq/code/deseq2.r`
- `http://data.biostarhandbook.com/books/rnaseq/code/edger.r`
- `http://data.biostarhandbook.com/books/rnaseq/code/heatmap.r`

If you do download the script separately then you may run each with R using:

```
Rscript deseq1.r
```

Note: the installation described above needs to be done only once!

8.2 Which data will be analyzed?

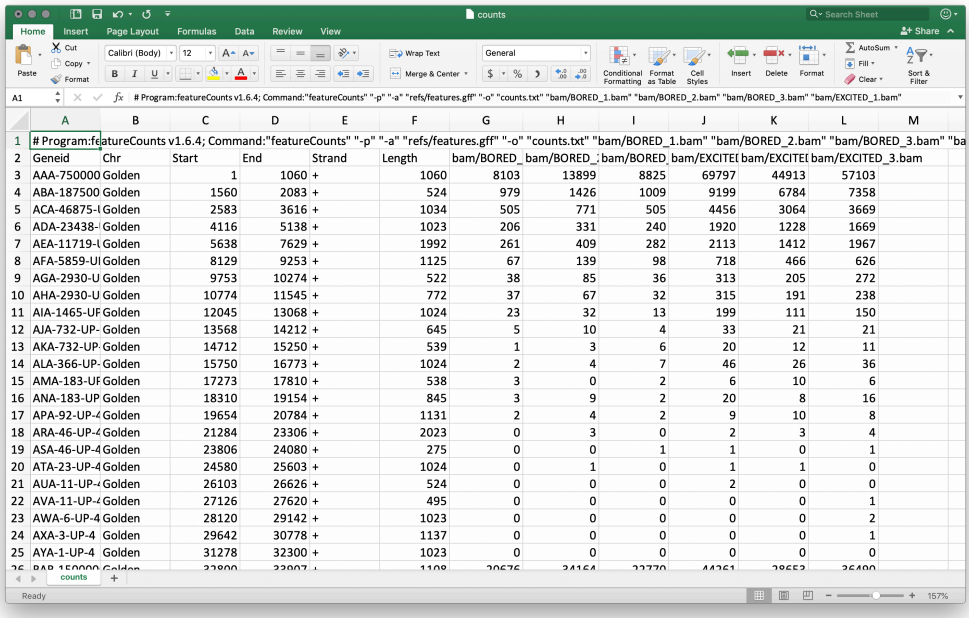
The previous chapters on Alignment based RNA-Seq or Classification based RNA-Seq ended with creating a count matrix. If you don't have that data download it from:

- `http://data.biostarhandbook.com/books/rnaseq/data/golden-counts.txt`

and name it `counts.txt`. From command line it would look like so:

```
curl http://data.biostarhandbook.com/books/rnaseq/data/golden-counts.txt > coun
```

the file contains the counts for alignments that overlap with a each feature:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	# Program:featureCounts v1.6.4; Command:"featureCounts" "-p" "-a" "refs/features.gff" "-o" "counts.txt" "bam/BORED_1.bam" "bam/BORED_2.bam" "bam/BORED_3.bam" "ba												
2	Geneid	Chr	Start	End	Strand	Length	bam/BORED_	bam/BORED_	bam/BORED_	bam/EXCITE	bam/EXCITE	bam/EXCITED_3.bam	
3	AAA-750000	Golden	1	1060	+	1060	8103	13899	8825	69797	44913	57103	
4	ABA-187500	Golden	1560	2083	+	524	979	1426	1009	9199	6784	7358	
5	ACA-46875-I	Golden	2583	3616	+	1034	505	771	505	4456	3064	3669	
6	ADA-23438-I	Golden	4116	5138	+	1023	206	331	240	1920	1228	1669	
7	AEA-11719-I	Golden	5638	7629	+	1992	261	409	282	2113	1412	1967	
8	AFA-5859-UI	Golden	8129	9253	+	1125	67	139	98	718	466	626	
9	AGA-2930-U	Golden	9753	10274	+	522	38	85	36	313	205	272	
10	AHA-2930-U	Golden	10774	11545	+	772	37	67	32	315	191	238	
11	AIA-1465-UI	Golden	12045	13068	+	1024	23	32	13	199	111	150	
12	AJA-732-UP-	Golden	13568	14212	+	645	5	10	4	33	21	21	
13	AKA-732-UP-	Golden	14712	15250	+	539	1	3	6	20	12	11	
14	ALA-366-UP-	Golden	15750	16773	+	1024	2	4	7	46	26	36	
15	AMA-183-UI	Golden	17273	17810	+	538	3	0	2	6	10	6	
16	ANA-183-UP	Golden	18310	19154	+	845	3	9	2	20	8	16	
17	APA-92-UP-4	Golden	19654	20784	+	1131	2	4	2	9	10	8	
18	ARA-46-UP-4	Golden	21284	23306	+	2023	0	3	0	2	3	4	
19	ASA-46-UP-4	Golden	23806	24080	+	275	0	0	1	1	0	1	
20	ATA-23-UP-4	Golden	24580	25603	+	1024	0	1	0	1	1	0	
21	AUA-11-UP-4	Golden	26103	26626	+	524	0	0	0	2	0	0	
22	AVA-11-UP-4	Golden	27126	27620	+	495	0	0	0	0	0	1	
23	AWA-6-UP-4	Golden	28120	29142	+	1023	0	0	0	0	0	2	
24	AXA-3-UP-4	Golden	29642	30778	+	1137	0	0	0	0	0	1	
25	AYA-1-UP-4	Golden	31278	32300	+	1023	0	0	0	0	0	0	

We will process this file with a pairwise comparison to determine which features show differential expression between the `EXCITED` and `BORED` states.

8.3 How do I find differential expression?

All statistical methods rely on various 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 frankly it not their fault. It always takes a surprising amount of effort to understand applicability of each method. You see scientists are typically not all that eager to talk about the limitations and would rather not even mention them.

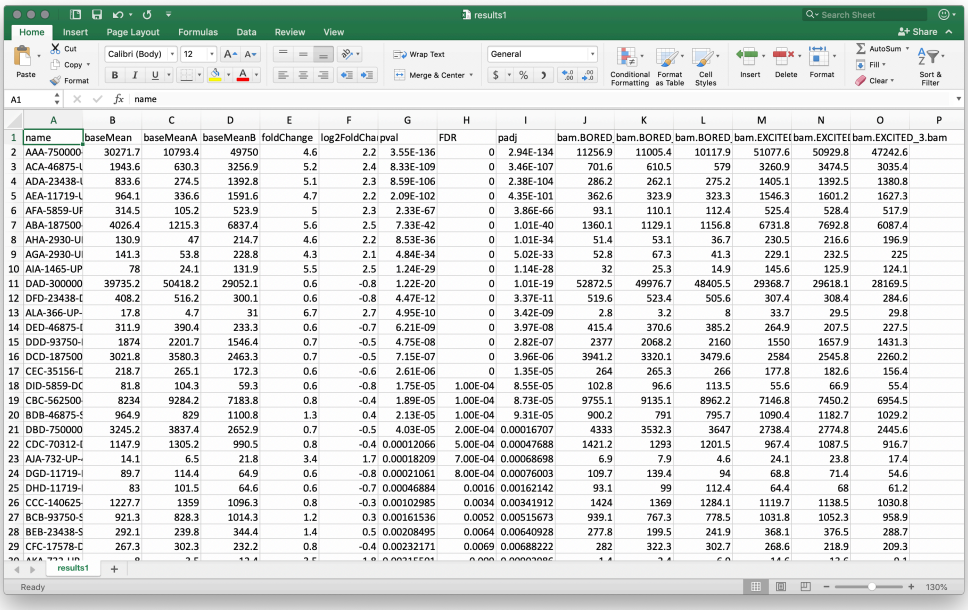
We will be making pairwise comparisons between two samples, each with multiple replicates. The statistical methods need to be informed which columns belong to the same sample. In our statistical scripts we chose to specify the experimental design by entering two numbers separated by an `x`. For example, in the current case we have counts for `3 BORED` and `3 EXCITED` samples. Thus the experimental design will be:

3x3

To perform a statistical analysis with the method published as Differential gene expression analysis based on the negative binomial distribution (deseq)[1] we execute:

```
cat counts.txt | deseq1.r 3x3 > results1.csv
```

The command above will create the `results1.csv` file that, when viewed in Excel, looks like this:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	name	baseMean	baseMeanA	baseMeanB	foldChange	log2FoldCha	pval	FDR	padj	bam.BORED_	bam.BORED_	bam.BORED_	bam.EXCITE(	bam.EXCITE(	bam.EXCITED_3.bam	
2	AAA-750000	30271.7	10793.4	49750	4.6	2.2	3.55E-136	0	2.94E-134	11256.9	11005.4	10117.9	51077.6	50929.8	47242.6	
3	ACA-46875-(	1943.6	630.3	3256.9	5.2	2.4	8.33E-109	0	3.46E-107	701.6	610.5	579	3260.9	3474.5	3035.4	
4	ADA-23438-(	833.6	274.5	1392.8	5.1	2.3	8.59E-106	0	2.38E-104	286.2	262.1	275.2	1405.1	1392.5	1380.8	
5	AEA-11719-(	964.1	336.6	1591.6	4.7	2.2	2.09E-102	0	4.35E-101	362.6	323.9	323.3	1546.3	1601.2	1627.3	
6	AFA-5859-UF	314.5	105.2	523.9	5	2.3	2.33E-67	0	3.86E-66	93.1	110.1	112.4	525.4	528.4	517.9	
7	ABA-187500	4026.4	1215.3	6837.4	5.6	2.5	7.33E-42	0	1.01E-40	1360.1	1129.1	1156.8	6731.8	7692.8	6087.4	
8	AHA-2930-UI	130.9	47	214.7	4.6	2.2	8.53E-36	0	1.01E-34	51.4	53.1	36.7	230.5	216.6	196.9	
9	AGA-2930-UI	141.3	53.8	228.8	4.3	2.1	4.84E-34	0	5.02E-33	52.8	67.3	41.3	229.1	232.5	225	
10	AIA-1465-UP	78	24.1	131.9	5.5	2.5	1.24E-29	0	1.14E-28	32	25.3	14.9	145.6	125.9	124.1	
11	DAD-300000	39735.2	50418.2	29052.1	0.6	-0.8	1.22E-20	0	1.01E-19	52872.5	49976.7	48405.5	29368.7	29618.1	28169.5	
12	DFD-23438-(	408.2	516.2	300.1	0.6	-0.8	4.47E-12	0	3.37E-11	519.6	523.4	505.6	307.4	308.4	284.6	
13	ALA-366-UP-	17.8	4.7	31	6.7	2.7	4.95E-10	0	3.42E-09	2.8	3.2	8	33.7	29.5	29.8	
14	DED-46875-(	311.9	390.4	233.3	0.6	-0.7	6.21E-09	0	3.97E-08	415.4	370.6	385.2	264.9	207.5	227.5	
15	DDD-93750-(	1874	2201.7	1546.4	0.7	-0.5	4.75E-08	0	2.82E-07	2377	2068.2	2160	1550	1657.9	1431.3	
16	DCD-187500	3021.8	3580.3	2463.3	0.7	-0.5	7.15E-07	0	3.96E-06	3941.2	3320.1	3479.6	2584	2545.8	2260.2	
17	CEC-35156-(	218.7	265.1	172.3	0.6	-0.6	2.61E-06	0	1.35E-05	264	265.3	266	177.8	182.6	156.4	
18	DID-5859-DC	81.8	104.3	59.3	0.6	-0.8	1.75E-05	1.00E-04	8.55E-05	102.8	96.6	113.5	55.6	66.9	55.4	
19	CBC-562500	8234	9284.2	7183.8	0.8	-0.4	1.89E-05	1.00E-04	8.73E-05	9755.1	9135.1	8962.2	7146.8	7450.2	6954.5	
20	BDB-46875-(	964.9	829	1100.8	1.3	0.4	2.13E-05	1.00E-04	9.31E-05	900.2	791	795.7	1090.4	1182.7	1029.2	
21	DBD-750000	3245.2	3837.4	2652.9	0.7	-0.5	4.03E-05	2.00E-04	0.00016707	4333	3532.3	3647	2738.4	2774.8	2445.6	
22	CDC-70312-(	1147.9	1305.2	990.5	0.8	-0.4	0.00012066	5.00E-04	0.00047688	1421.2	1293	1201.5	967.4	1087.5	916.7	
23	AJA-732-UP-	14.1	6.5	21.8	3.4	1.7	0.00018209	7.00E-04	0.00068698	6.9	7.9	4.6	24.1	23.8	17.4	
24	DGD-11719-(	89.7	114.4	64.9	0.6	-0.8	0.00021061	8.00E-04	0.00076003	109.7	139.4	94	68.8	71.4	54.6	
25	DHD-11719-(	83	101.5	64.6	0.6	-0.7	0.00046884	0.0016	0.00162142	93.1	99	112.4	64.4	68	61.2	
26	CCC-140625-	1227.7	1359	1096.3	0.8	-0.3	0.00102985	0.0034	0.00341912	1424	1369	1284.1	1119.7	1138.5	1030.8	
27	BCB-93750-5	921.3	828.3	1014.3	1.2	0.3	0.00161536	0.0052	0.00515673	939.1	767.3	778.5	1031.8	1052.3	958.9	
28	BEB-23438-5	292.1	239.8	344.4	1.4	0.5	0.00208495	0.0064	0.00640928	277.8	199.5	241.9	368.1	376.5	288.7	
29	CFC-17578-(	267.3	302.3	232.2	0.8	-0.4	0.00232171	0.0069	0.00688222	282	322.3	302.7	268.6	218.9	209.3	

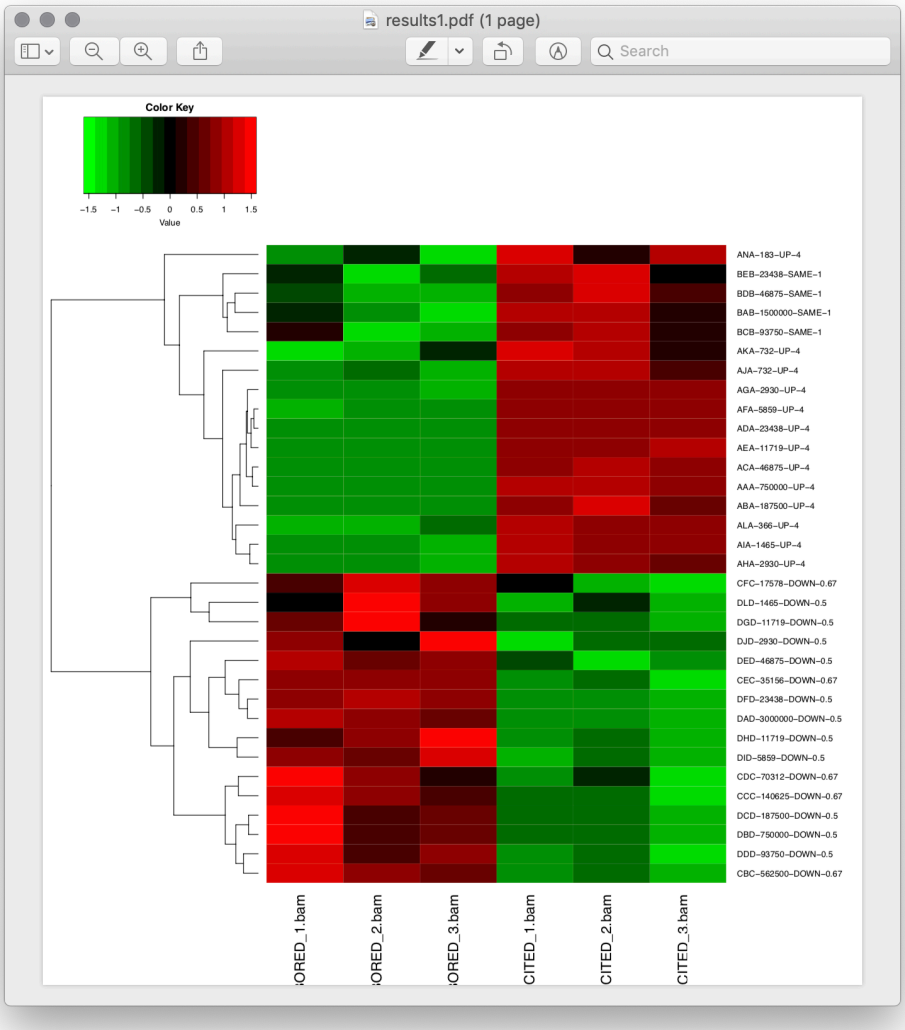
And that's it! **Your statistical analysis is done**. The remaining steps are not a computational anymore but rely on the interpretation of the results.

We will explain below what the `results1.csv` file contains, but for now we note that you can also create a heatmap of the differentially expressed features stored in the `results1.csv` file with the following instruction:

```
cat results1.csv | heatmap.r > results1.pdf
```

The code above produces the visualization:

[1] `https://bioconductor.org/packages/release/bioc/html/DESeq.html`



Pretty neat!

8.4 But wait, there is more!

We have implemented not just one, but three methods. All work the same way.

Perhaps you'd want to perform the analysis with a Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2 (deseq2)[2]. This is how you do it:

```
cat counts.txt | deseq2.r 3x3 > results2.csv
```

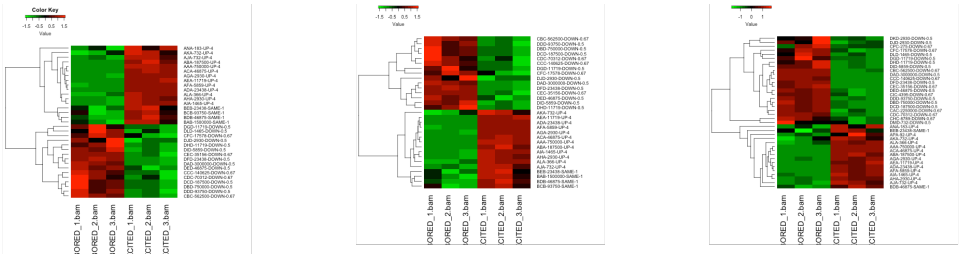
What if you wish to do an Empirical Analysis of Digital Gene Expression Data in R (edger)[3]. We made that work with:

```
cat counts.txt | edger.r 3x3 > results3.csv
```

It gets better! The heat maps can be also generated the same way:

```
cat results1.csv | heatmap.r > results1.pdf
cat results2.csv | heatmap.r > results2.pdf
cat results3.csv | heatmap.r > results3.pdf
```

You can quickly generate a result file and heatmap for each method:



Three complete statistical analyses, all generated in fewer than 20 seconds. It is a magical world, now isn't it?

Now ask yourself, why is it that if you look at the original documentation for each method above you will find sprawling instructions that would take you hours-upon-hours of trial and error, until you finally give up, many never succeeding doing what they wanted. What in this book looks like this:

```
cat counts.txt | method1.r 3x3 > results1.csv
cat counts.txt | method2.r 3x3 > results2.csv
cat counts.txt | method3.r 3x3 > results3.csv
```

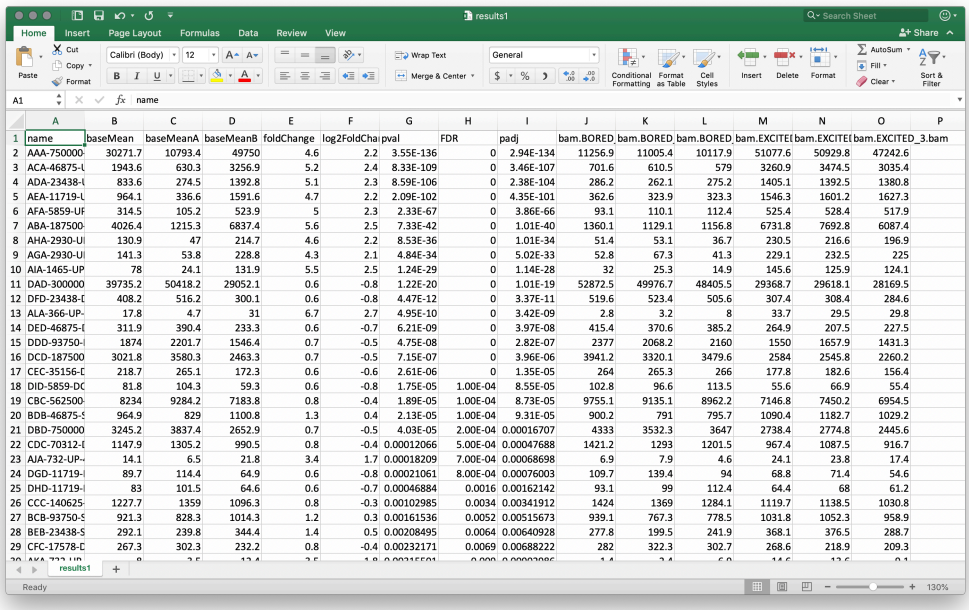
is far more difficult to accomplish in the world envisioned by the creators of each package. Your job as a future scientist, as one who will shape science for the decades to come is to **demand** that tools are not just sophisticated mathematically, but also straightforward to use!

[2] `http://bioconductor.org/packages/release/bioc/html/DESeq2.html`
[3] `https://bioconductor.org/packages/release/bioc/html/edgeR.html`

8.5 What do the results mean?

We have come up with what we believe is the minimally required data to make informed decisions about gene or transcript expressions. As it happens, none of the statistical packages we know of do automatically produce this information! As a matter of fact we had to make a concerted effort to both standardize the results across all methods, and to ensure that each method includes all fields. T

The result file is a column oriented comma separated (CSV) file that can be readily opened in Excel. Each row in the file represents information on a single feature across two conditions (A and B). As we shown it before it looks like this:



The required columns are the following:

1. `name` - the feature identity. Must be unique within the column. It may be a gene name, a transcript name, an exon - whatever the feature that we chose to quanitfy.
2. `baseMean` - the average normalized expression level across samples. It measure how much total information is there.
3. `baseMeanA` - the average normalized expression level across the first

condition. It measure how much total information is there for condition A.

4. `baseMeanB` - the average normalized expression level across the first condition. It measure how much total information is there for condition B.
5. `foldChange` - the ratio of `baseMeanB`/`baseMeanA`. Very important to always be aware that in the fold change means B/A (second condition/first condition)
6. `log2FoldChange` - the second logarithm of `foldChange`. Log 2 transformations are convenient as they transform the changes onto a uniform scale. A four fold increase after transformation is `2` . A four fold decrease (1/4) after log 2 transform is `-2`. Much easier to compare the magnitude of up/down changes.
7. `pval` - the uncorrected p-value of the likelihood of observing the effect of the size `foldChange` (or larger) by chance alone. This p-value is not corrected for multiple comparisons.
8. `FDR` - the False Discovery Rate - this column is increasing, and represents the fraction of false discoveries in all the rows above the row where the value is listed. If, in row 100 the FDR is 0.01, it means that if you were to take all rows above and including the current row then 100 * 0.01 = 1 result out of the 100 will be false.
9. `padj` - adjusted p-value. The multiple comparison adjusted p-value from the column `pval`. Different methods apply different adjusments. `padj` will always larger than `pval`. As an approximation, this value is usually as much smaller as many features in the whole table. For example if `pval` is 1E-6 and you have 100 (1E2) features then `padj` will be around 1E-4.
10. The next columns represent the normalized matrix for your count data.

8.6 What is the normalized matrix?

In the chapter Statistical analysis for RNA-Seq we describe the concept in more detail, in a nutshell it is a transformation that makes numbers comparable across samples. For example compaaring earnings of say `$25` vs `$50`only makes sense if we know that both refer to the same time period.

The original (raw) counts that we load into the statistical method will first

get transformed into a normalized counts - then the statistical method is run on these normalized counts. We believe that is exceedingly important to be able to investigate this data to fully understand what the statistical method does. For example, here are the original counts for our 6 samples:

```
cat counts.txt | grep AAA | cut -f 7-12
```

it prints:

```
8103    13899   8825    69797   44913   57103
```

The normalized counts however are:

```
# Also replace the comma with tab for easier comparisons
cat results1.csv | grep AAA | cut -d , -f 10-16 | tr ',' '\t'
```

the code prints:

```
11256.9  11005.4  10117.9  51077.6  50929.8  47242.6
```

Let's write the numbers under one another (I will also drop the digit for clarity):

```
Original:     8103    13899     8825    69797    44913    57103
Normalized:  11256    11005    10117    51077    50929    47242
```

Let's recap: the normalization method of the `deseq1` method rescales the numbers to make them comparable. The algorithm that the method uses rescales the numbers. For example it turns 8103`into`11256`, then in another column transforms`13899`into`11005‘, an so on. Note how some columns are adjusted up, others are adjusted down. Do also note how after normalization the numbers make more sense, the replicates are a lot more consistent.

Statistical packages may apply a different normalization processes, and that's in addition to applying a different statistical method on evaluating the differences. No wonder there are disagreements on which one method is better.

We include the normalized matrix because we believe that uou need to understand what the statistical method "thinks" about your data. All the methods developed in the book will provide you with the normalized matrix.

8.7 How do I visualize the normalized matrix?

We believe this matrix is essential and should be visualized and clustered to understand the inter-replicate and inter-sample variability. To that end, any result file created with our tools above can be immediately visualized with:

```
cat results1.csv | heatmap.r > results1.pdf
```

8.8 Where do go next?

Now that we run two algorithms and three methods we are all dygin to know: So which method is best?

Chapter 9

7. Which method is the best?

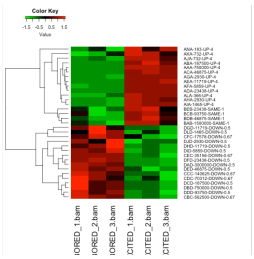
In the previous chapter we've applied different statistical methods to our results:

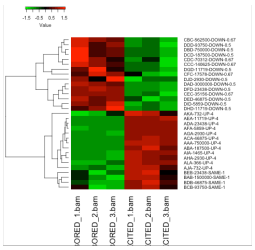
```
cat counts.txt | deseq1.r 3x3 > results1.csv
cat counts.txt | deseq2.r 3x3 > results2.csv
cat counts.txt | edger.r  3x3 > results3.csv
```

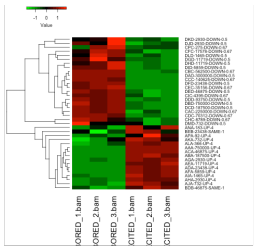
we've even generated three different heatmaps:

```
cat results1.csv | heatmap.r > results1.pdf
cat results2.csv | heatmap.r > results2.pdf
cat results3.csv | heatmap.r > results3.pdf
```

that looked like this:







Which method is the best? In what context is it the best? Should we always use that method?

9.1 What kinds of errors do we expect to see?

When reporting differentially expressed genes several kinds of errors may occur:

1. Failing to report genes that did change - **false negatives**
2. Reporting genes that did not actually change - **false positives**
3. Effect size inaccuracies. A fold change of 4 could be reported as a foldchange of 2.

Now you can see what makes the `Gold Snitch` naming convention so convenient. It allows us identify the errors by inspecting the gene names. Recall that our genes are named as:

```
ADA-23438-UP-4
DLD-1465-DOWN-0.5
BEB-23438-SAME-1
```

Where the first number is the concentration, the second is the direction of change, and the last is the magnitude of change. How will our errors look?

- The genes with the word `SAME` in the name reported as differentially expressed are the false positives.
- Genes with the word `UP` or `DOWN` in the name reported as not being differentially expressed are all false negatives.
- The difference in the fold change relative to the number in the name is the accuracy of our method.

9.2 How many genes should be detected as differentially expressed?

```
cat results1.csv | egrep "UP|DOWN" |  wc -l
```

prints:

```
69
```

Thus we know 69 genes do change (have `UP` of `DOWN` in their name). But we also should expect that some genes express at such low levels, that there might not be be enough data to detect the change.

9.3 How many genes are actually detected as differentially expressed?

We will call a gene *differentially expressed* if its adjusted p-value is less than `0.05`. While commonly used, the choice is not the only possible option. We could also use the FDR column or come up with a different definition altogether.

We will use a tool called `csvcut` to operate on our CSV files. The tool allows us to cut columns by name, rather than having to remember the column index. Some of our results files might have different orders of columns, thus this approach helps avoid making mistakes. If you don't have `csvcut` installed already do a `conda install csvkit -y`.

First we extract the names of the differentially expressed genes from method into a separate file. For the simplicity of subsequent commands we'll call the files `A` (deseq1), `B` (deseq2) and `C` (edger)

```
cat results1.csv | csvcut -c name,padj | awk -F , '$2 < 0.05 { print $1 }' | sort > A
cat results2.csv | csvcut -c name,padj | awk -F , '$2 < 0.05 { print $1 }' | sort > B
cat results3.csv | csvcut -c name,padj | awk -F , '$2 < 0.05 { print $1 }' | sort > C
```

The files `A`, `B` and `C` contain the names of the differentially expressed genes detected by the corresponding method:

```
cat A | head -5
```

prints:

```
AAA-750000-UP-4
ACA-46875-UP-4
ADA-23438-UP-4
...
```

The above are the genes detected with `deseq1`.

9.4 How to find common elements?

We will use tool called `comm` to find common or distinct elements in sorted files. When used properly `comm` allows you to answer a wide variety of interesting questions with relative ease.

Running:

```
man comm
```

describes the tool for us:

```
The comm utility reads file1 and file2, which should be sorted lexically, and
produces three text columns as output:

  - lines only in file1;
  - lines only in file2;
  - lines in both files.

The following options are available:

  -1      Suppress printing of column 1.
  -2      Suppress printing of column 2.
  -3      Suppress printing of column 3.
```

There is only one recurring difficulty when using `comm`. We have to remember that the flags remove columns from the output. When we select `-1` we *remove* the lines present only in file 1.

9.5 How many features detected by each method?

```
wc -l A B C
```

prints:

```
33 A
31 B
38 C
```

Looks like `edger` produces most features that pass the `0.05` significance level. We now that there are `69` changed genes in total, thus even the best, if none of the results were false positives, even the best method would only find about 55% of the changed genes. Clearly there are limitations to the RNA-Seq method when it comes to comprehensively identify all genes that change.

9.5.1 How many false positives in each method?

In our results a false positive is a differentially expressed gene that has the word `SAME` in the name:

```
cat A | grep SAME
```

We'll use a neat trick called anonymous pipe (`<()`), to print and paste the output of all three files A,B and C at the same time:

```
paste <(cat A | grep SAME) <(cat B | grep SAME) <(cat C | grep SAME)
```

Will print:

```
BAB-1500000-SAME-1      BAB-1500000-SAME-1      BDB-46875-SAME-1
BCB-93750-SAME-1        BCB-93750-SAME-1        BEB-23438-SAME-1
BDB-46875-SAME-1        BDB-46875-SAME-1
BEB-23438-SAME-1        BEB-23438-SAME-1
```

Looks like `deseq1` and `deseq2` produce the same four false positives, whereas `edger` produces just two. Two false positives are the same across all three methods!

9.5.2 How many genes are present in both A and B?

```
comm -1 -2 A B  | wc -l
```

Prints:

```
31
```

The results from `deseq2` are fully contained within the results of `deseq1`.

9.5.3 Which genes are found only with `deseq2`?

```
comm -1 -3 A B
```

Nothing gets printed! Looks like `deseq1` contains all results of `deseq2`.

9.5.4 Which genes are found only with `deseq1`?

```
comm -2 -3 A B
```

prints:

```
ANA-183-UP-4
DLD-1465-DOWN-0.5
```

9.5.5 Which genes are found only with `deseq1` and not `edger`?

```
comm -2 -3 A C
```

prints:

```
BAB-1500000-SAME-1
BCB-93750-SAME-1
```

Looks like `edger` contains all the genes from `deseq1` minus to false positives.

9.5.6 Which genes are found only with C (edger) and not A (deseq1)

```
comm -1 -3 A C
```

prints:

```
APA-92-UP-4
CAC-2250000-DOWN-0.67
CHC-8789-DOWN-0.67
CIC-4395-DOWN-0.67
CPC-275-DOWN-0.67
DKD-2930-DOWN-0.5
DMD-732-DOWN-0.5
```

All the additional genes found with `edger` are correct! Nice job `edger`

9.6 And the winner is `edger`

For this particular example `edger` is the clear-cut winner. The `edger` results contain 2 fewer false positives and 36 genes that do indeed change. Out of the 69 the method correctly identified 34 valid genes.

That being said, it does not mean the same result will hold across all types of data.

9.7 Which genes were missed?

```
# Get all gene names that do not stay the same.
cat counts.txt | cut -f 1 | egrep "UP|DOWN"  | sort > CHANGED

# Which genes are changed but not in the edger results?
comm -2 -3 CHANGED C
```

Below is the full list of genes that the methods were not able to detect as differentially expressed. What is common about them? The concentrations are either very low, or the change is not that substantial. For example for fold changes over 4 the minimum detection level concentration is at 183. Whereas for a fold change of 0.67 the minimum concentration levels is 8700!

The full list of false negatives:

```
AMA-183-UP-4
ARA-46-UP-4
ASA-46-UP-4
ATA-23-UP-4
AUA-11-UP-4
AVA-11-UP-4
AWA-6-UP-4
AXA-3-UP-4
AYA-1-UP-4
CGC-8789-DOWN-0.67
CJC-2197-DOWN-0.67
CKC-2197-DOWN-0.67
CLC-1099-DOWN-0.67
CMC-549-DOWN-0.67
CNC-549-DOWN-0.67
CRC-137-DOWN-0.67
CSC-137-DOWN-0.67
CTC-69-DOWN-0.67
CUC-34-DOWN-0.67
```

```
CVC-34-DOWN-0.67
CWC-17-DOWN-0.67
CXC-9-DOWN-0.67
CYC-2-DOWN-0.67
DND-732-DOWN-0.5
DPD-366-DOWN-0.5
DRD-183-DOWN-0.5
DSD-183-DOWN-0.5
DTD-92-DOWN-0.5
DUD-46-DOWN-0.5
DVD-46-DOWN-0.5
DWD-23-DOWN-0.5
DXD-11-DOWN-0.5
DYD-3-DOWN-0.5
```

Part II

RNA-SEQ CONCEPTS

Chapter 10

RNA-Seq terminology

In the section, we'll try to clarify terms that you may encounter when reading RNA-Seq related documentation.

10.1 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 x 5=10

The "RNA-Seq" sample should not be confused with "statistical sample" – in statistics "sample" refers to selecting a subset of a population.

10.2 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.

10.3 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.

10.4 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 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.

10.5 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.

10.6 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`) that 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:

```
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:

```
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 group[1] of scientists that invented this measure would have just called this quantity a *pachter*, as a hat tip to Lior Pachter[2], 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.

10.7 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 units[3].

[1] `http://www.nature.com/nmeth/journal/v5/n7/abs/nmeth.1226.html`
[2] `https://math.berkeley.edu/~lpachter/`
[3] `https://haroldpimentel.wordpress.com/2014/05/08/`

10.8 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.

10.9 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`.

```
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 group[4] of scientists that invented TPM would have just called this quantity a *salzberg*,as a hat tip to Steven Salzberg[5] 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).

10.10 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.

10.11 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.

10.12 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/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.

10.13 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

10.14 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' Retracted, Replaced[7]

[7] `https://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!

`newsalrt_190327_MSCPEDIT&uac=267659MN&impID=1919559&faf=1`

Chapter 11

Statistical analysis for RNA-Seq

Our favorite summary for statistics comes from the blog post The four aspects of statistics[1] 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.

[1] `http://madhadron.com/posts/2016-01-15-aspects_of_statistics.html`

read more on each point on the blog[2].

11.1 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.

11.2 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 statistican, 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

[2] http://madhadron.com/posts/2016-01-15-aspects_of_statistics.html

```
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.

11.3 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 routinel make mistakes when summarizing and reformulating statements derived from statistical tests. See the section on p-values later on this page.

11.4 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.

11.5 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.

11.6 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 Computing[3]

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`.

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 are not that well versed in proper software engineering practice. As a result typical R code is affected by far many more issues that 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.

11.7 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 Page[4].

Once installed this way the program will be universally available on your computer from the command line as well.

11.8 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:

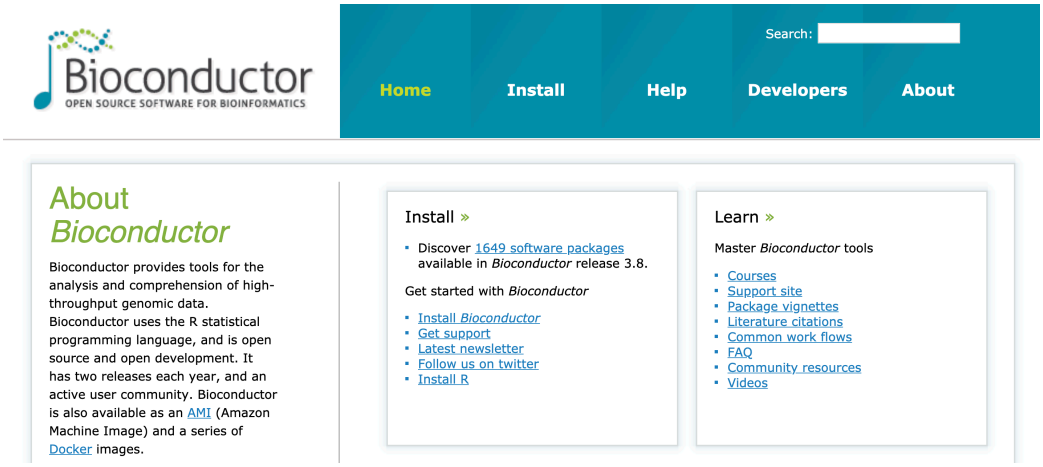
[3] `https://www.r-project.org/`
[4] `https://www.r-project.org/`

```
conda install r
```

If you work in R, a good option (especially when learning) is to run it via RStudio[5], a graphical user interface to R.

11.9 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.

[5] `https://www.rstudio.com/`

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
```

11.10 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 misunderstood[6]
- Scientists rise up against statistical significance[7] in Nature, 2019
- Statisticians issue warning over misuse of P values[8]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

[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`

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 somewhat 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 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 "expert statistican" goes out of their ways in taking any

responsibility for making a decisions - all the responsibility is punted back to the biologist basically saying to them, 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…*, That entire advice for us feels nothing more than p-hacking and data dredging[9], 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!

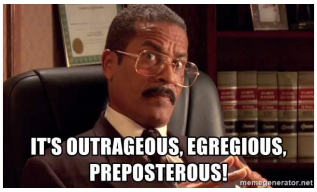
Perhaps our explanation will feel like splitting hairs but it is not - it cuts to the very essence of p-hacking. Filtering data before an analysis is an acceptable practice. You may remove lowly expressed genes, or highly expressed genes, genes that rhyme with `foo`, or genes that *from far away look like flies*, you may do whatever you wish to do as long as you have a scientifically acceptable rationale for doing so. State and explain that rationale, then document it - all fine. What is however *not acceptable* is the reasoning that the letter above recommends: to filter data once we saw the results, for the *sole purpose* of making more results pass a threshold. That, in our opinion, is the definition of p-hacking[10]. Yet a famous statistician is the author of the advice above. What gives?

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 incredible 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. Gee, thanks (sarcasm)!

11.11 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

[9] `https://en.wikipedia.org/wiki/Data_dredging`
[10] `https://en.wikipedia.org/wiki/Data_dredging`



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"*
- *"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 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 change could be small or large. The pvalue does not capture the actual size of the effect. 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.

But then, 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 even understand the concept or that it is even worth using p-values *at all*:

- Interpreting P values[11], Nature Methods 2017
- P values and the search for significance[12]
- … and so on …

11.12 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 noise as seen in Genomic organization of human transcription initiation complexes, Nature, 2013[13].

[11] `https://www.nature.com/articles/nmeth.4210`
[12] `https://www.nature.com/articles/nmeth.4120`
[13] `https://www.nature.com/articles/nature12535`

Part III

FURTHER EXPLORATIONS

Chapter 12

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.

12.1 How would I get started solving this puzzle?

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 how 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.

12.2 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`.

12.3 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	?	?

12.4 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”?

12.5 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.

Chapter 13

The Bear Paradox

Park rangers hike through forests until they identify 100 animals:

1. In the first forest, rangers counted 90 bears, 1 rabbit and 9 squirrels.
2. In the second forest, rangers counted 30 bears, 20 rabbits and 50 squirrels

Answer the following question:

- Are there more bears in the first forest than in the second?

Explore the following ideas:

- List what you think are valid interpretations of the counts above.
- List what you think might be common, yet invalid interpretation of the counts above.

Draw a parallel to sequencing instrument that works by randomly measuring a predetermined number of DNA fragments from a sample.

- Can we tell how many DNA molecules were there in total?
- What if the sample contains DNA from large number of organisms each with a different length of DNA?
- Would there be a change if a sequencer sequenced all DNA that was placed in the instrument?

What if park rangers counted patches of fur. Instead of seeing 30 bears they found 30 patches of bear fur. Etc.

How would your interpretation change.

13.1 Write your own data here

Imagine that you have an instrument that can sequence 1000 reads. An organism can be observed in two states:

1. Wild type (`WT`): transcript A expresses with 800 copies, transcript B expresses with 200 copies
2. Cold Shock (`SHOCK`): transcript A expresses with 800 copies, transcript B expresses with 200,000 copies

Note how transcript A is unaffected, it expresses at same level in both states.

Imagine that you ran an RNA seq experiment what would be a quantification matrix that would capture the expected changes above

```
      WT       SHOCK    FOLD_CHANGE
A
B
```

Could you get another correct matrix that, even though (technically) correct, does not capture the expected changes?

Part IV

RECIPES

Chapter 14

Recipe: Alignment based RNA-SEQ

14.1 Download

- http://data.biostarhandbook.com/books/rnaseq/code/rnaseq-alignment-recipe.sh

14.2 Command line use

```
# Get the recipe.
wget -nc http://data.biostarhandbook.com/books/rnaseq/code/rnaseq-alignment-recipe.sh

# Run the recipe.
bash rnaseq-alignment-recipe.sh
```

14.3 Code listing

```
#!/usr/bin/env bash
```

```
#
# This program performs an alignment based RNA-Seq analysis.
#
# Details at: http://www.biostarhandbook.com
#

# Stop on errors. Print the commands.
set -uex

# Download the reference genome.
wget -nc http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar.g

# Unpack the reference genome.
tar xzvf golden.genome.tar.gz

# Download the data
wget -nc http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz

# Unpack the data
tar zxvf golden.reads.tar.gz

# The indexed reference genome.
IDX=refs/genome.fa

# Build the hisat2 genome index.
hisat2-build $IDX $IDX

# Index the reference genome with samtools.
samtools faidx $IDX

# Create the root ids of the data layout.
parallel -j 1 echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids

# Create the BAM folder.
mkdir -p bam
```

```
# Align the FASTQ files to the reference genome.
cat ids | parallel "hisat2 -x $IDX -1 reads/{}_R1.fq -2 reads/{}_R2.fq -S bam/{}.sam"

# Sort each SAm into a BAM file.
cat ids | parallel "samtools sort bam/{}.sam > bam/{}.bam"

# Index each BAM file.
cat ids | parallel  "samtools index bam/{}.bam"

# First we turn the each BAM file into BedGraph coverage.
cat ids | parallel "bedtools genomecov -ibam  bam/{}.bam -split -bg  > bam/{}.bg"

# Then convert the BedGraph coverage into BigWig coverage.
cat ids | parallel "bedGraphToBigWig bam/{}.bg  ${IDX}.fai bam/{}.bw"

# Run featureCounts on BAM files in the right order.
featureCounts -p -a refs/features.gff -o counts.txt bam/BORED_?.bam bam/EXCITED_?.bam

# Download the edger R script.
curl http://data.biostarhandbook.com/books/rnaseq/code/edger.r > edger.r

# Download the heatmap R script.
curl http://data.biostarhandbook.com/books/rnaseq/code/heatmap.r > heatmap.r

# Perform the differential expression detection with edger.
cat counts.txt | Rscript edger.r 3x3 > results.csv

# Draw the heatmap from the results.
cat results.csv | Rscript heatmap.r > results.pdf
```

Chapter 15

Recipe: Classification based RNA-Seq

15.1 Download

- http://data.biostarhandbook.com/books/rnaseq/code/rnaseq-classification-r
  sh

15.2 Command line use

```
# Get the recipe.
wget -nc http://data.biostarhandbook.com/books/rnaseq/code/rnaseq-classificatio

# Run the recipe.
bash rnaseq-classification-recipe.sh
```

15.3 Code listing

```
#!/usr/bin/env bash
```

```
#
# This program performs a classification based RNA-Seq analysis.
#
# Details at: http://www.biostarhandbook.com
#

# Stop on errors. Print the commands.
set -uex

# Download the reference genome.
wget -nc http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar.gz

# Unpack the reference genome.
tar xzvf golden.genome.tar.gz

# Download the data
wget -nc http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz

# Unpack the data
tar zxvf golden.reads.tar.gz

# Set the name for the reference.
REF=refs/transcripts.fa

# Set the name of the index.
IDX=refs/transcript.idx

# Index the genome with kallisto.
kallisto index -i $IDX $REF

# Index the reference genome with samtools
samtools faidx $REF

# Create the root ids of the data layout.
parallel -j 1 echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids
```

```
# Make a directory for the results
mkdir -p output

# Run Kallisto to classify the reads.
cat ids | parallel kallisto quant -i $IDX -o output/{} reads/{}_R1.fq reads/{}

# Download the custom script to combine kallisto outputs.
curl http://data.biostarhandbook.com/books/rnaseq/code/combine.py > combine.py

# Combine the outputs created by kallisto.
cat ids | python combine.py output > counts.txt

# Download the edger R script.
curl http://data.biostarhandbook.com/books/rnaseq/code/edger.r > edger.r

# Perform the differential expression detection with edger.
cat counts.txt | Rscript edger.r 3x3 > results.csv

# Draw the heatmap from the results.
cat results.csv | Rscript heatmap.r > results.pdf
```

Chapter 16

Recipe: Mission Impossible RNA-Seq

16.1 Download

- http://data.biostarhandbook.com/books/rnaseq/code/rnaseq-mission-impossible.sh

16.2 Command line use

```
# Get the recipe.
wget -nc http://data.biostarhandbook.com/books/rnaseq/code/rnaseq-mission-impossible.s

# Run the recipe.
bash rnaseq-mission-impossible.sh
```

16.3 Code listing

```
#
# Your mission, should you choose to accept it,
```

```
# is to run a complete RNA-Seq analysis in a minute.
#
# Good luck, commander! The fate of your world depends on you!
#
# Usage:
#
#   time bash mission-impossible-rnaseq.sh
#
# Stop on any error.
#
set -uex

# The location of the reference genome.
GENOME_URL=http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar

# The location of the sequencing reads.
DATA_URL=http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz

# Download and unpack both data
curl -f $GENOME_URL | tar xz
curl -f $DATA_URL | tar xz

# Run sequence statistics on the sequencing reads.
# seqkit stats reads/*.fq

#  Run sequence statistics on the genome and transcripts.
# seqkit stats refs/*.fa

# Create the sample names.
parallel echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids

# The indexed reference genome.
IDX=refs/genome.fa

# The messages.log contains information that would be printed to the screen.
# Consult the content of the messages.log file for information if you have t
```

```
# Index the reference genome. Needs to be done only once and can be reused.
hisat2-build $IDX $IDX  > messages.log 2> messages.log

# Index the reference genome with samtools. Needs to be done only once and can be r
samtools faidx $IDX

# Create the folder that will store the alignment files.
mkdir -p bam

# Align the FASTQ files to the reference genome.
cat ids | parallel "hisat2 -x $IDX -1 reads/{}_R1.fq -2 reads/{}_R2.fq -S bam/{}.sam

# Sort each SAm into a BAM file.
cat ids | parallel "samtools sort bam/{}.sam > bam/{}.bam"

# Index each BAM file.
cat ids | parallel  "samtools index bam/{}.bam"

# Run the featureCounts program to summarize the number of reads that overlap with
featureCounts -p -a refs/features.gff -o counts.txt bam/BORED_?.bam bam/EXCITED_?.bam

#
# We will run a differential analysis with  different methods.
#

# Store the analysis scripts separately.
mkdir -p code

# Get the data analysis scripts.
wget -P code -qnc  http://data.biostarhandbook.com/books/rnaseq/code/deseq1.r
wget -P code -qnc  http://data.biostarhandbook.com/books/rnaseq/code/deseq2.r
wget -P code -qnc  http://data.biostarhandbook.com/books/rnaseq/code/edger.r
wget -P code -qnc  http://data.biostarhandbook.com/books/rnaseq/code/heatmap.r

# Generate the results with method 1.
cat counts.txt | Rscript code/deseq1.r 3x3 > results1.csv
```

```
cat results1.csv | Rscript code/heatmap.r > results1.pdf

# Generate the results with method 2.
cat counts.txt | Rscript code/deseq2.r 3x3 > results2.csv 2>> messages.log
cat results2.csv | Rscript code/heatmap.r > results2.pdf

# Generate the results with method 3.
cat counts.txt | Rscript code/edger.r 3x3 > results3.csv
cat results3.csv | Rscript code/heatmap.r > results3.pdf
```