SPARTA Documentation

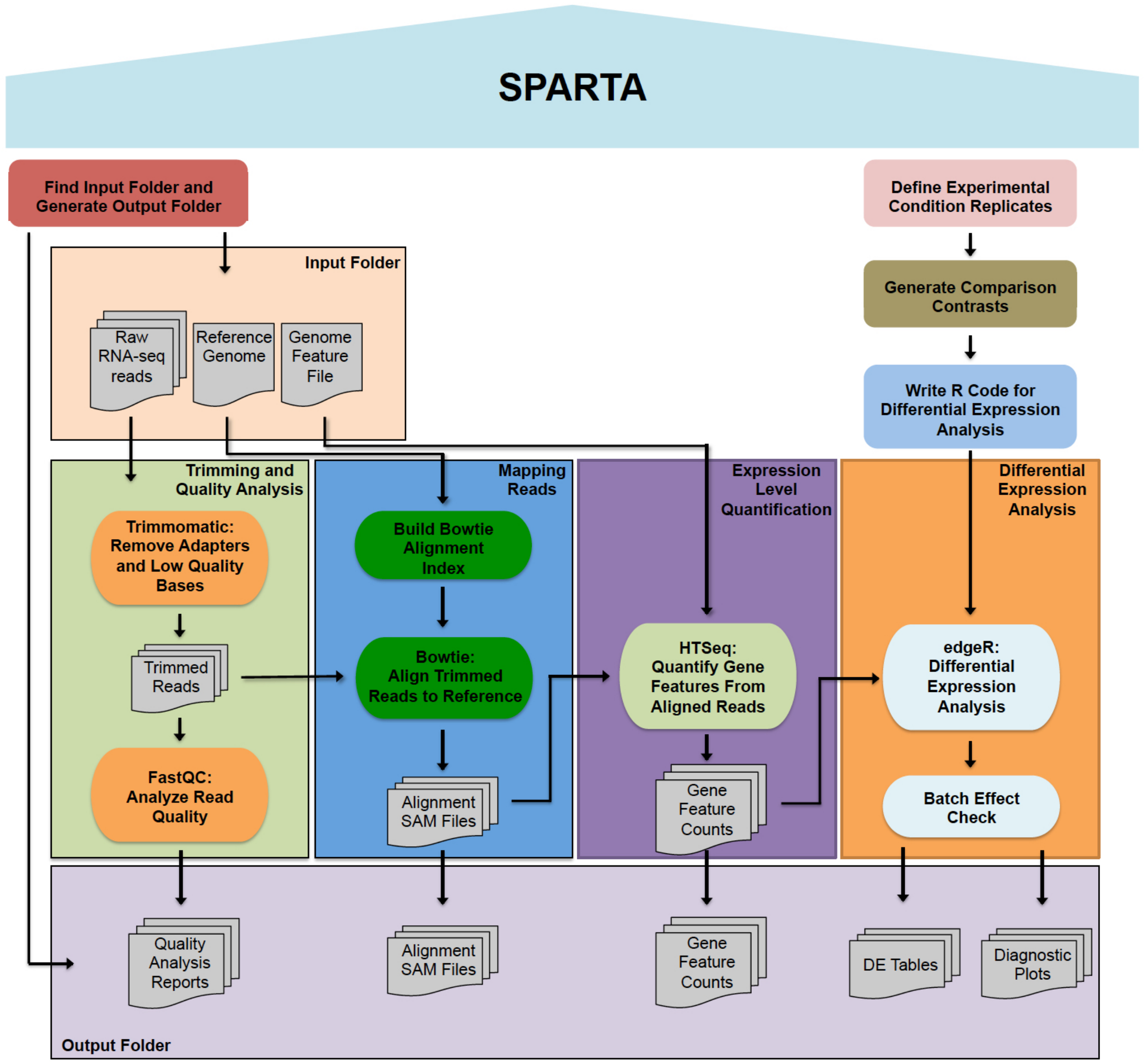
Release 1.0

Benjamin K Johnson

August 04, 2016

Contents

SPARTA is a workflow aimed at analyzing single-end Illumina RNA-seq data. The software is supported on Windows, Mac OS X, and Linux platforms. The workflow combines several tools: Trimmomatic (read trimming/adapter removal), FastQC (read quality analysis), Bowtie (mapping reads to the reference genome), HTSeq (transcript/gene feature abundance counting), and edgeR (differential gene expression analysis). Within the differential gene expression analysis step, batch effects can be detected and the user is warned of the potential, unintended additional variable. The analysis procedure is outlined below.


SPARTA
Find Input Folder and Generate Output Folder
Input Folder
Raw RNA-seq reads
Reference Genome
Genome Feature File
Define Experimental Condition Replicates
Generate Comparison Contrasts
Write R Code for Differential Expression Analysis
Trimming and Quality Analysis
Trimmomatic: Remove Adapters and Low Quality Bases
Trimmed Reads
FastQC: Analyze Read Quality
Mapping Reads
Build Bowtie Alignment Index
Bowtie: Align Trimmed Reads to Reference
Alignment SAM Files
Expression Level Quantification
HTSeq: Quantify Gene Features From Aligned Reads
Gene Feature Counts
Differential Expression Analysis
edgeR: Differential Expression Analysis
Batch Effect Check
Quality Analysis Reports
Alignment SAM Files
Gene Feature Counts
DE Tables
Diagnostic Plots
Output Folder

CHAPTER 1

How to get and use SPARTA:

Mac Users - Mac OS X tutorial

Windows Users - Windows tutorial

Linux Users - Linux tutorial

Cloud computing tutorial - Cloud computing with SPARTA on Amazon EC2

1.1 Contents:

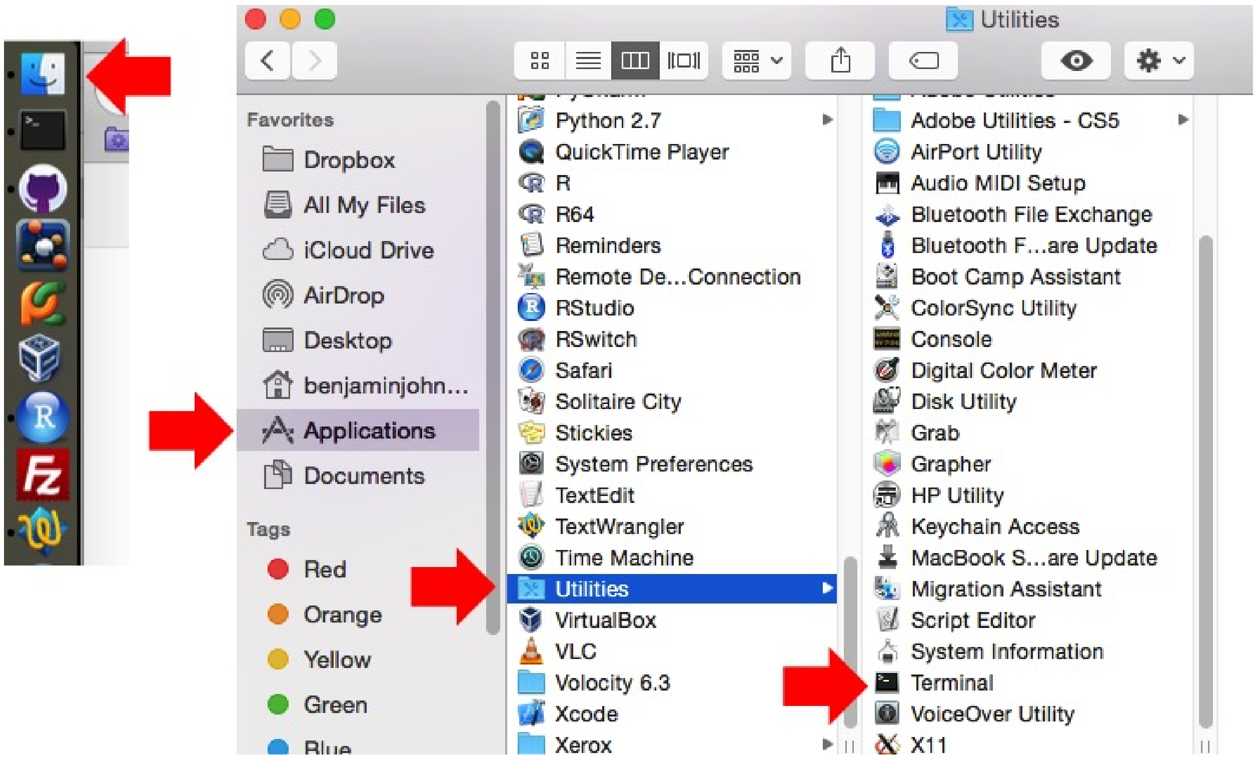
1.1.1 Mac OS X tutorial

Important: There is a known issue introduced by Apple in the newer operating system (El Capitan) that does affect SPARTA. You will need to install the command line tools. To initiate that process, type 'gcc' into the terminal (without the quotes) and hit enter. From here it will ask you if you want to install the command line tools. Click 'Install' or 'Agree'. Close and re-open the terminal and proceed with the subsequent installation steps.

Download the workflow: SPARTA for Mac

Introduction

Many bioinformatics software packages and workflows require the user to utilize them from the command line or terminal. SPARTA is no different. The reason the command line interface is utilized is that a great deal of power and flexibility can be gained without the use of a graphical user interface (GUI). Further, a GUI can be difficult to implement across various platforms. To find the command line interface/Terminal on Mac OS X, go to Finder -> Applications -> Utilities -> Terminal (might just be worth dragging it onto your dock).



Decompress the SPARTA_Mac-master.zip file by double-clicking on it. Now, drag and drop the decompressed folder onto your desktop.

SPARTA expects either compressed (.gz) or uncompressed FASTQ files (.fq or .fastq) as input, with a reference genome file in FASTA format and a genome feature file (.gtf) within the folder that contains the input data. To see an example of appropriate input data, look inside the ExampleData folder within the SPARTA_Mac-master folder.

To download a reference genome and genome feature file for your favorite bacteria, go to the Ensembl website. The reference genome and feature file are already present for the ExampleData.

Basic Terminal Commands

Let's have a look at some basic Terminal commands, we will cover the commands necessary to:

1. Move through folders

2. List the contents of a folder

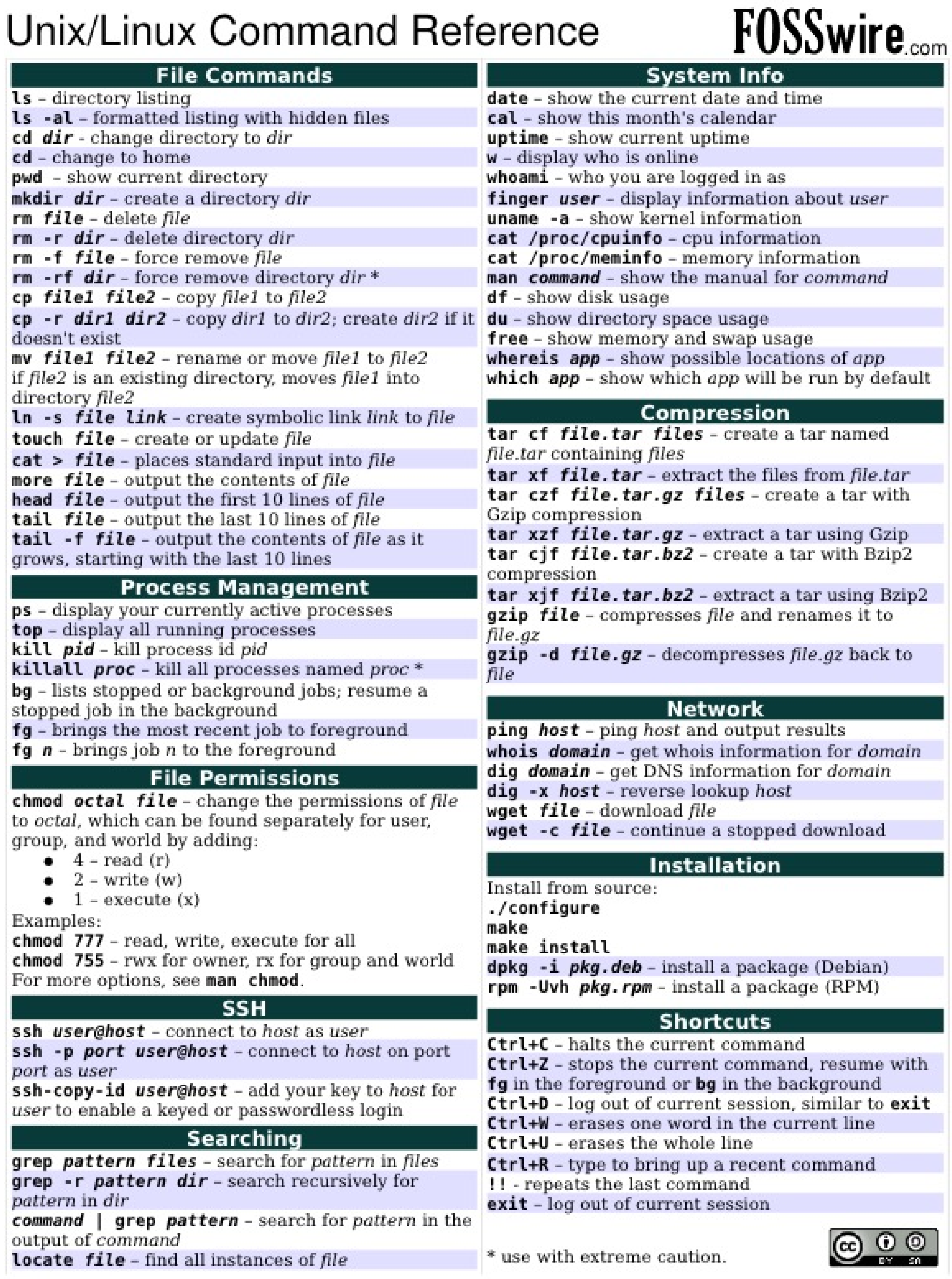
3. Make new folders

4. Rename files/folders

5. Delete files/folders

	Command	What it does	Examples
1.	cd	Change directory/folder	cd ~ (this changes to your home directory); cd .. (this goes back one folder)
2.	ls	List the contents of a folder	ls
3.	mkdir	Make a new directory/folder	mkdir NewFolder (this will make a new folder called 'NewFolder' in your current directory)
4.	mv	Rename or move a file from one name to another	mv file1 file2 (this will rename/move file1 to file2)
5.	rm	Remove a file (add the -r flag to remove a folder)	rm file1 (remove file1); rm -r folder1 (remove folder1)

Command reference sheet



Unix/Linux Command Reference

FOSSwire.com

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2* if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

chmod *octal file* – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
For more options, see **man chmod**.

SSH

ssh *user@host* – connect to *host* as *user*
ssh -p *port user@host* – connect to *host* on port *port* as *user*
ssh-copy-id *user@host* – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* – search for *pattern* in *files*
grep -r *pattern dir* – search recursively for *pattern* in *dir*
command* | grep *pattern – search for *pattern* in the output of *command*
locate *file* – find all instances of *file*

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*
wget *file* – download *file*
wget -c *file* – continue a stopped download

Installation

Install from source:
```
./configure
make
make install
```
dpkg -i *pkg.deb* – install a package (Debian)
rpm -Uvh *pkg.rpm* – install a package (RPM)

Shortcuts

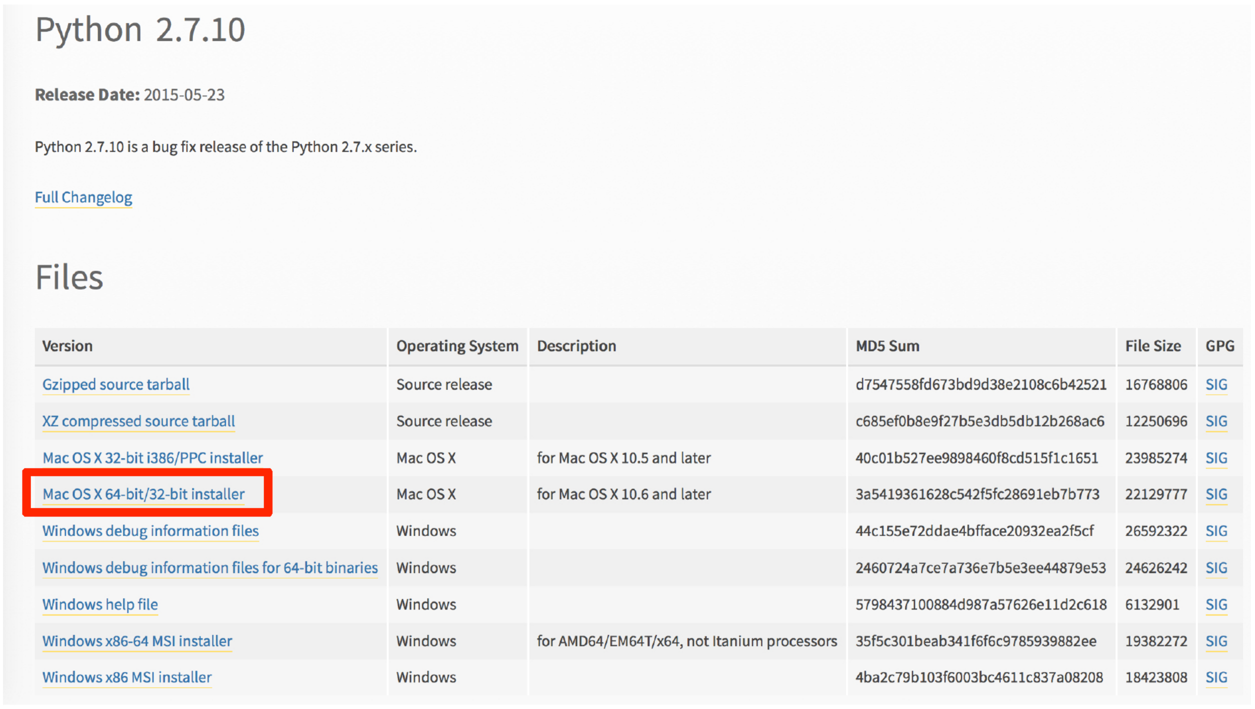
Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

* use with extreme caution.

Ref. sheet from: http://files.fosswire.com/2007/08/fwunixref.pdf

Install Dependencies

The SPARTA workflow requires a few things in order to run: Python, Java, NumPy, and R. If you already have these installed, great! If you don't, let's start by downloading the latest version of Python 2 (see image below). You will want to download and install the red boxed version of Python 2. Follow the prompts to install Python with the default values.



Python 2.7.10

Release Date: 2015-05-23

Python 2.7.10 is a bug fix release of the Python 2.7.x series.

Full Changelog

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		d7547558fd673bd9d38e2108c6b42521	16768806	SIG
XZ compressed source tarball	Source release		c685ef0b8e9f27b5e3db5db12b268ac6	12250696	SIG
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later	40c01b527ee9898460f8cd515f1c1651	23985274	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	3a5419361628c542f5fc28691eb7b773	22129777	SIG
Windows debug information files	Windows		44c155e72ddae4bfface20932ea2f5cf	26592322	SIG
Windows debug information files for 64-bit binaries	Windows		2460724a7ce7a736e7b5e3ee44879e53	24626242	SIG
Windows help file	Windows		5798437100884d987a57626e11d2c618	6132901	SIG
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64, not Itanium processors	35f5c301beab341f6f6c9785939882ee	19382272	SIG
Windows x86 MSI installer	Windows		4ba2c79b103f6003bc4611c837a08208	18423808	SIG

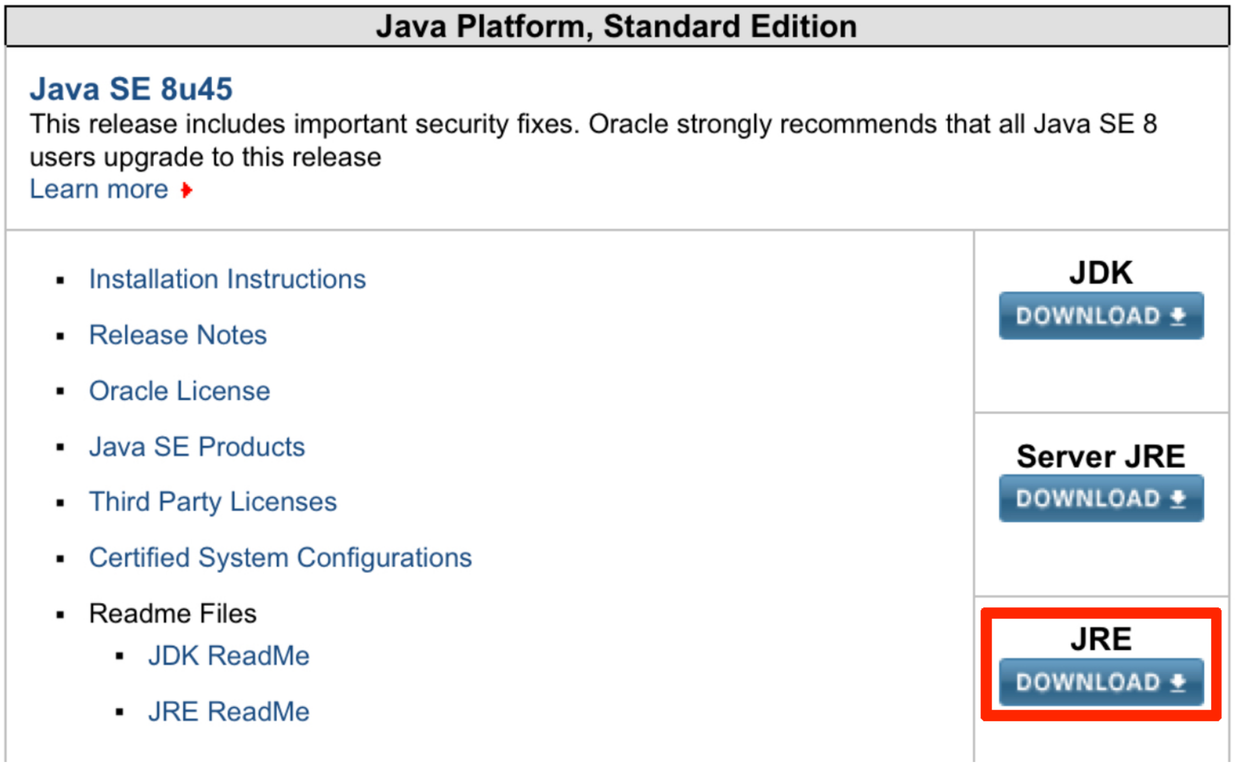
Great! Let's check and see if Java is already installed on your system. Open up the terminal, (if you don't remember how to do this, head back to the *Introduction*) and type:

```
java -version
```

If Java is already installed, it will produce some output that looks like this:

```
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
```

If the output does *not* look something like this, Java is likely not installed and two of the tools require Java to function (Trimmomatic and FastQC). Let's download and install a suitable version of Java (see image below). You will want to download and install the red boxed version of Java JRE. You will also need to click on the button (red arrow) to accept the terms and conditions of using Java JRE. Follow the prompts to install Java.



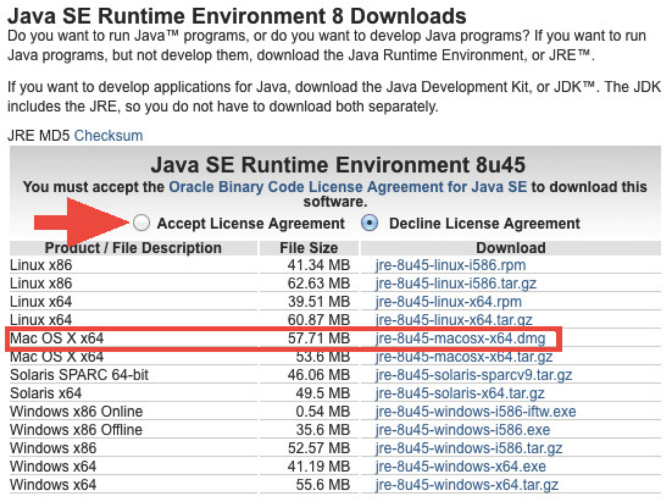
Java Platform, Standard Edition

Java SE 8u45

This release includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release

Learn more

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations
- Readme Files
    - JDK ReadMe
    - JRE ReadMe



Java SE Runtime Environment 8 Downloads

Do you want to run Java™ programs, or do you want to develop Java programs? If you want to run Java programs, but not develop them, download the Java Runtime Environment, or JRE™.

If you want to develop applications for Java, download the Java Development Kit, or JDK™. The JDK includes the JRE, so you do not have to download both separately.

JRE MD5 Checksum

Java SE Runtime Environment 8u45

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux x86	41.34 MB	jre-8u45-linux-i586.rpm
Linux x86	62.63 MB	jre-8u45-linux-i586.tar.gz
Linux x64	39.51 MB	jre-8u45-linux-x64.rpm
Linux x64	60.87 MB	jre-8u45-linux-x64.tar.gz
Mac OS X x64	57.71 MB	jre-8u45-macosx-x64.dmg
Mac OS X x64	53.6 MB	jre-8u45-macosx-x64.tar.gz
Solaris SPARC 64-bit	46.06 MB	jre-8u45-solaris-sparcv9.tar.gz
Solaris x64	49.5 MB	jre-8u45-solaris-x64.tar.gz
Windows x86 Online	0.54 MB	jre-8u45-windows-i586-iftw.exe
Windows x86 Offline	35.6 MB	jre-8u45-windows-i586.exe
Windows x86	52.57 MB	jre-8u45-windows-i586.tar.gz
Windows x64	41.19 MB	jre-8u45-windows-x64.exe
Windows x64	55.6 MB	jre-8u45-windows-x64.tar.gz

To install NumPy, go back to or open the Terminal and type:

```
sudo pip install numpy
```

This will prompt you for your password. Enter your password and hit Enter/Return.

Note: As you type in your password, **no characters will appear** but you *are* entering characters.

Once you have entered your password and hit Enter/Return, NumPy will be downloaded and installed on your system.

Finally, let's install R. Navigate to the SPARTA_Mac folder and go to the folder labeled "Install_R". Within this folder is an R installer. Double-click on the installer and follow the prompts to install R.

Note: If you have OSX 10.9 (Mavericks) or higher, you will want to use version 3.2.3. If you have OSX 10.6 to 10.8, you want to use the version 3.2.1. To check which version you have, click on the Apple logo in the upper left hand corner of your screen and then click on "About This Mac". A window will appear telling you which version of OSX you have.

Congratulations! You've installed the necessary dependencies to run SPARTA!

Initializing SPARTA

Once SPARTA is initialized, the workflow will seek to identify that all of the necessary dependencies are met. If they are not satisfied, a message specific to what is not installed will appear as output in the terminal window.

To initialize SPARTA, go to the Terminal and navigate to the SPARTA_Mac-master folder on your desktop by typing:

```
cd ~/Desktop/SPARTA_Mac-master
```

To start the workflow, type:

```
python SPARTA.py
```

This will start the software and check for dependencies.

Analyzing Example Data

SPARTA is distributed with some example data. Specifically, it is the first 100,000 reads of each sample from Baker et al..

To begin the analysis, navigate into the SPARTA_Mac-master folder and drag and drop the folder called "Example-Data" out onto the desktop.

If you haven't already, *initialize SPARTA* from the Terminal.

If all the *dependencies* are met, SPARTA will pause and prompt the user:

```
Is the RNAseq data in a folder on the Desktop? (Y or N):
```

Type:

```
Y
```

Hit Enter/Return

Note: SPARTA assumes the data is located in a folder on the desktop by default. It is easiest if all future analyses have the data in a folder (WITHOUT SPACES IN THE NAME) on the desktop.

Now it will prompt the user for the name of the folder:

```
What is the name of the folder on the Desktop containing the RNAseq data?:
```

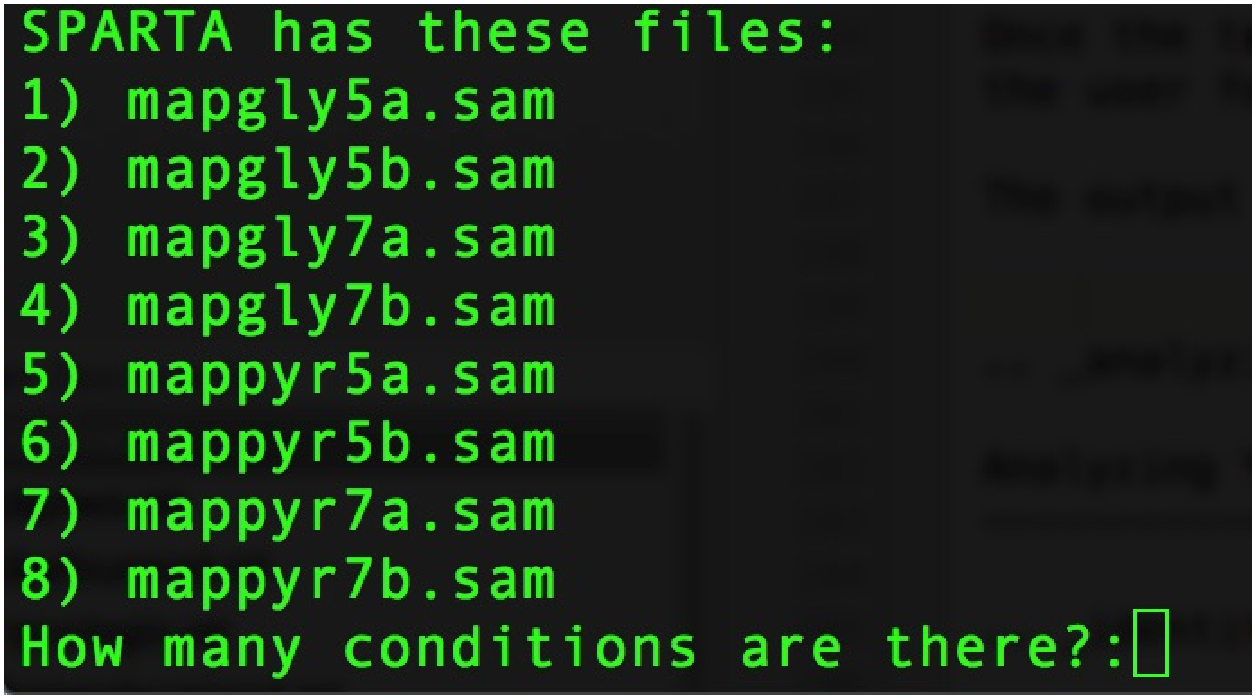
Type:

```
ExampleData
```

This is the name of the folder on the desktop that contains the input example data. Hit Enter/Return. From here, the software will trim, QC, align, and count transcript abundance for each sample. All output/analyses are put in a folder that SPARTA generates on the desktop called "RNAseq_Data". Within this folder are separate folders for each SPARTA run that are denoted by the date (e.g. 2015-06-04). Within these folders are four more folders that separate each step of the analysis and are called: 1) QC, 2) Bowtie, 3) HTSeq, and 4) DEanalysis.

Once the trimming, QC, alignment, and counting are complete, SPARTA will again pause and prompt the user for how many experimental conditions exist within the analysis.

The output at this point will look like this:



```
SPARTA has these files:
1) mapgly5a.sam
2) mapgly5b.sam
3) mapgly7a.sam
4) mapgly7b.sam
5) mappyr5a.sam
6) mappyr5b.sam
7) mappyr7a.sam
8) mappyr7b.sam
How many conditions are there?:
```

At the prompt that says:

```
How many conditions are there?:
```

Type:

```
4
```

Hit Enter/Return. There are 4 experimental conditions that we are considering:

1. Glycerol pH 7.0
2. Glycerol pH 5.7

3. Pyruvate pH 7.0
4. Pyruvate pH 5.7

Each condition has 2 replicates. The next prompt will read:

```
Enter the relevant file names, based on the names given in 'SPARTA has these files', with the replica
As an example, please see the 'conditions_input_example.txt' in the DEanalysis folder.
Once you have entered the file names, hit Enter/Return:
```

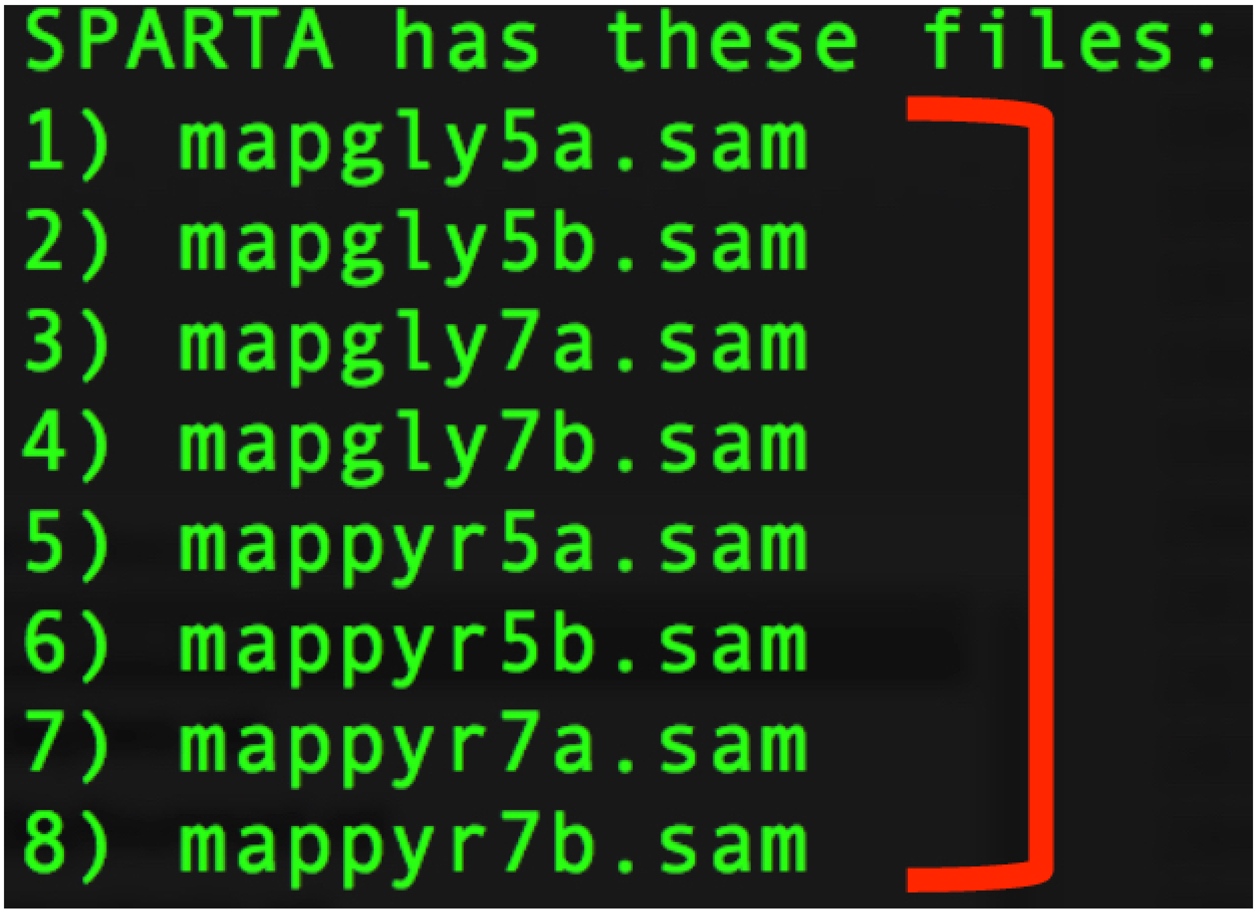
At this point, we need to edit a text file (conditions_input.txt) to tell SPARTA which file belongs to a given condition. To do this:

1. Navigate to the SPARTA output folder called RNAseq_Data located on the desktop
2. Go to the current run folder (will be the last folder listed if sorted by name)
3. Go into the DEanalysis folder
4. Open the conditions_input.txt file in a text editor (NOT MICROSOFT WORD) such as TextEdit

The number of experimental conditions listed are based on the number entered at the prompt asking "How many conditions are there?:". Thus, in our case, there are 4. The contents of the file will look like:

```
Reference_Condition_Files:
Experimental_Condition_2_Files:
Experimental_Condition_3_Files:
Experimental_Condition_4_Files:
```

We now need to enter the file names of the replicates in each condition. These are comma-separated file names that correspond to the output given by SPARTA (denoted with red bracket)



```
SPARTA has these files:
1) mapgly5a.sam
2) mapgly5b.sam
3) mapgly7a.sam
4) mapgly7b.sam
5) mappyr5a.sam
6) mappyr5b.sam
7) mappyr7a.sam
8) mappyr7b.sam
```

Note: The file names are case-sensitive and must be spelled *exactly* as listed in the output given by SPARTA

Thus, when all the file names are inputed, the conditions_input.txt file should look like this:

```
Reference_Condition_Files: mapgly7a.sam, mapgly7b.sam
Experimental_Condition_2_Files:mapgly5a.sam, mapgly5b.sam
Experimental_Condition_3_Files:mappyr7a.sam, mappyr7b.sam
Experimental_Condition_4_Files:mappyr5a.sam, mappyr5b.sam
```

Now, save the changes by going to File -> Save. Go back to the terminal and hit Enter/Return. From here, the workflow will perform the differential gene expression analysis through edgeR. If a batch effect may be present, the output will attempt to warn the user of the potential, unintended variable that *must* be accounted for before drawing experimental conclusions.

All the differential gene expression output is located in the RNAseq_Data -> date of your current run -> DEanalysis folder. The file output includes:

1. Differential gene expression tables
2. MDS plot (somewhat analogous to a principle component analysis plot) which will show whether your replicates group together and treatment groups separate based on the treatment
3. BCV plot (biological coefficient of variation) to look at gene level variation between samples

Congratulations! You've analyzed RNA-seq data from raw reads to differential gene expression!

Analyzing Your Data

If you haven't already, we recommend working through the *example data analysis* first before attempting to work through your own data set to familiarize yourself with the workflow.

As stated in the *Introduction*, SPARTA expects either compressed (.gz) or uncompressed FASTQ files (.fq or .fastq) as input, with a reference genome file in FASTA format and a genome feature file (.gtf) within the folder that contains the input data on your desktop. To see an example of appropriate input data, look inside the ExampleData folder within the SPARTA_Mac-master folder.

Now, to analyze your own data, follow the steps to *initialize SPARTA*, and start the analysis!

If you would like to tweak the analysis options for a given step/tool, have a look at the *Altering Workflow Execution Options*.

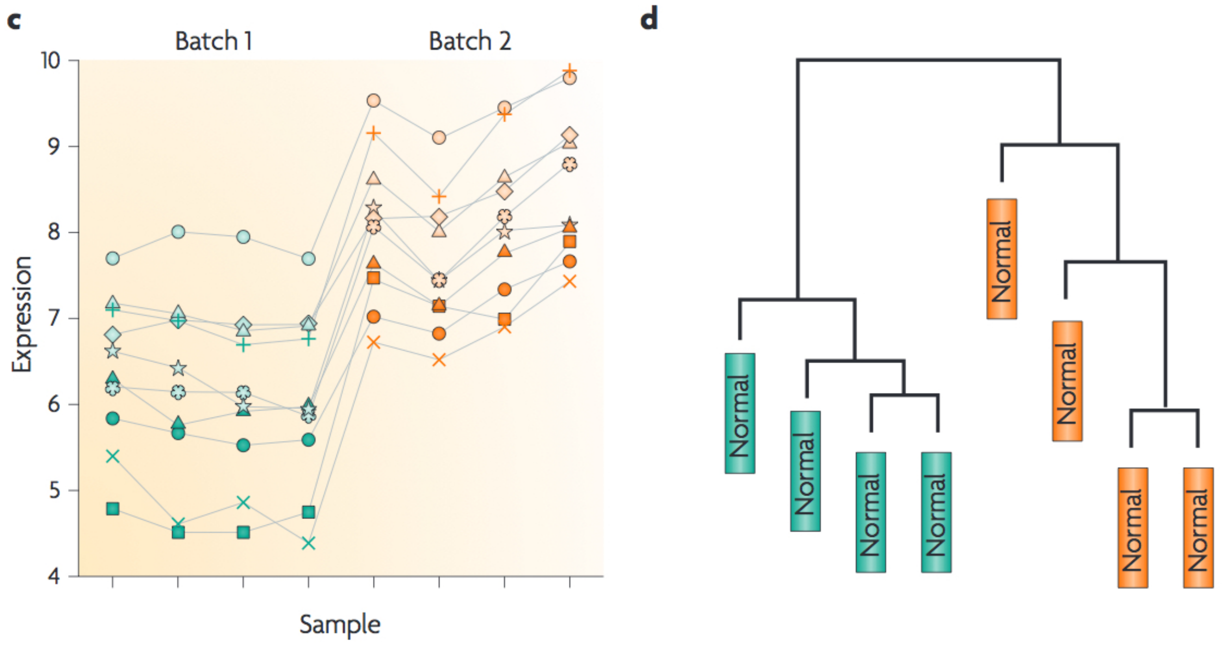
Identifying Potential Batch Effects

Batch effects can be a source of variation in RNA-seq data that can confound biological conclusions. In fact, there have been documented cases of batch effects present in published studies that led readers to be concerned for the validity of the results.

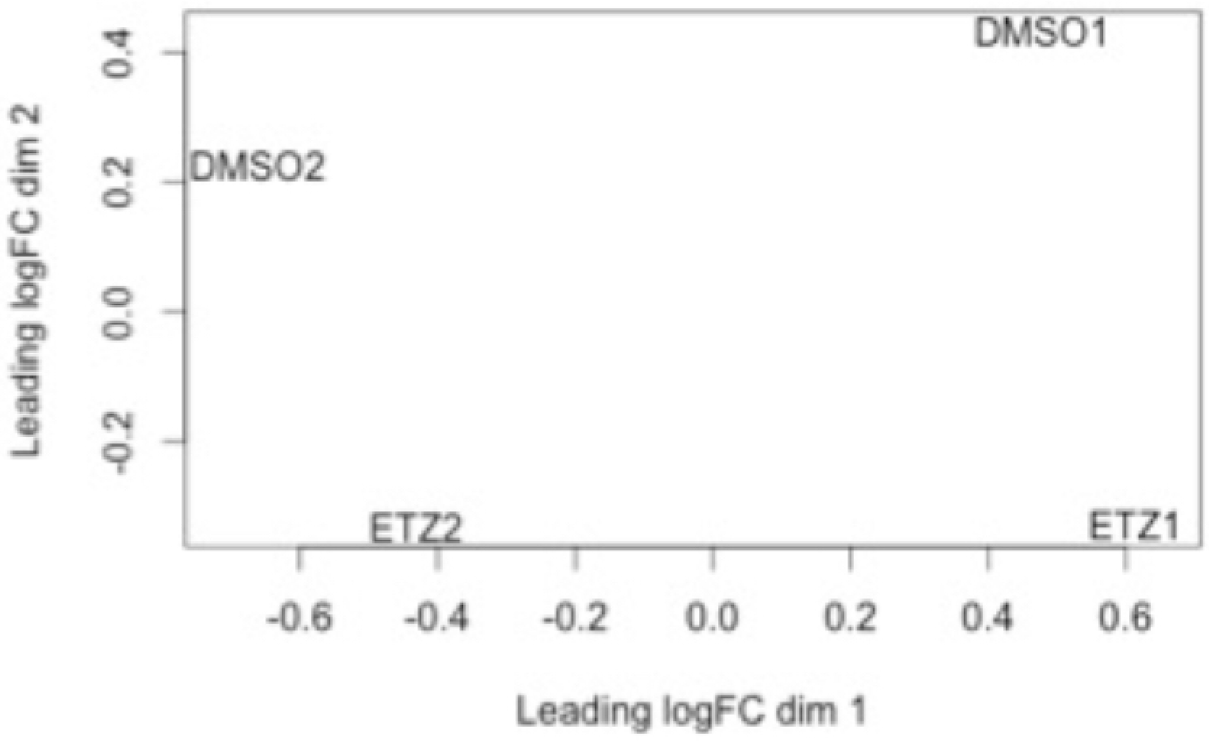
To quote a previously published paper in Nature Reviews Genetics, "Batch effects are sub-groups of measurements that have qualitatively different behaviour across conditions and are unrelated to the biological or scientific variables in a study. For example, batch effects may occur if a subset of experiments was run on Monday and another set on Tuesday, if two technicians were responsible for different subsets of the experiments or if two different lots of reagents, chips or instruments were used."

Thus, it is paramount that one address batch effects within their data before drawing biological conclusions from a specific RNA-seq experiment. To illustrate what a batch effect may look like within the data, we will utilize several different plots.

This first plot comes from the Nature Reviews Genetics paper where they examine Affymetrix data from a published bladder cancer study. You can quickly see that panels C and D from Figure 1 show that samples from batch 1 (blue) cluster together based on gene expression and samples from batch 2 (orange) cluster together.



Within RNA-seq data, using SPARTA and the MDS plot generated by edgeR, another example of batch effects within a study comparing *Mycobacterium tuberculosis* treated with a compound, we can clearly see that the mock-treated samples (DMSO) and compound-treated samples (ETZ) separate based on batch (A vs B) instead of by treatment. Ideally, we would have the samples group together based on treatment as opposed to batch.



If a potential batch effect is detected in the data set, SPARTA will output a message into the terminal that says:

```
IMPORTANT! YOU MAY HAVE A BATCH EFFECT! PLEASE LOOK AT THE MDS PLOT!
```

If this occurs, have a look at the MDS plot in the RNAseq_Data folder -> date of current run -> DEanalysis folder -> MDSplot.png

From here, you will want to adjust your model to account for the batch effect. Within edgeR, this can be accomplished through an additive linear model. The documentation for edgeR contains a tutorial on how to deal with batch effects that can be found here.

Future implementations of SPARTA will include the ability to adjust for batch effects.

Altering Workflow Execution Options

SPARTA is capable of allowing the user to alter the parameters associated with each analysis step to be tailored to specific use cases. Below are the different parameters that can be altered and their usage.

Options:

```
Usage: python SPARTA.py [options]

      Simple Program for Automated reference-based bacterial RNA-seq Transcriptome
      Analysis (SPARTA)
```

```
-h, --help            show this help message and exit
--cleanup             Clean up the intermediate files to save space. Default
                      action is to retain the intermediate files.
--verbose             Display more output for each step of the analysis.
--noninteractive      Non-interactive mode. This is for running SPARTA
                      without any user input. Assumes data is on the
                      desktop. If this option is specified, you must fill
                      out the configuration file (ConfigFile.txt) with the
                      appropriate experimental conditions in the SPARTA
                      folder.
--threads=THREADS     Define the number of threads that SPARTA should run
                      with. This will enable some speed-up on multi-
                      processor machines. As a generality, define the number
                      of threads as the same number of cores in your
                      computer. Default is 2.

Trimmomatic options:
  The order the options will be run are: ILLUMINACLIP, LEADING,
  TRAILING, SLIDINGWINDOW, MINLEN

  --clip=ILLUMINACLIP
                      ILLUMINACLIP options. MiSeq & HiSeq usually
                      TruSeq3.fa; GAII usually TruSeq2.fa. Default is
                      ILLUMINACLIP:TruSeq3-SE.fa:2:30:10. Usage:
                      --clip=<adapterseqs>:<seed mismatches>:<palindrome
                      clip threshold>:<simple clip threshold>
  --lead=LEADING      Set the minimun quality required to keep a base.
                      Default is LEADING=3. Usage: --lead=<quality>
  --trail=TRAILING    Set the minimum quality required to keep a base.
                      Default is TRAILING=3. Usage: --trail=<quality>
  --slidewin=SLIDINGWINDOW
                      SLIDINGWINDOW options. Default is SLIDINGWINDOW:4:15.
                      Usage: --slidewin=<window_size>:<required_quality>
  --minlentrim=MINLENTRIM
                      Set the minimum read length to keep in base pairs.
                      Default is 36. Usage: --minlentrim=<readlength>

Bowtie options:
  --mismatch=MISMATCH
                      Output alignments with at most a defined number of
                      mismatches. Usage: --mismatch=<integer_value>
  --otherbowtieoptions=OTHERBOWTIEOPTIONS
                      Bowtie has so many options that it is not worth
                      listing them here. Go to http://bowtie-
                      bio.sourceforge.net/manual.shtml#command-line for the
                      manual and all available options. Usage:
                      --otherbowtieoptions='all options inputed as a string
                      (note the quotes!)'

HTSeq options:
  --stranded=STRANDED
                      Stranded options: yes, no, reverse. Default is
                      --stranded=reverse. Usage: --stranded=yes/no/reverse
  --order=ORDER       Order options: name, pos. Usage: --order=name/pos.
  --minqual=MINQUAL   Skip all reads with quality lower than the given
                      value. Default is --minqual=10. Usage:
                      --minqual=<value>
  --type=TYPE         The feature type (3rd column in GTF file) to be used.
```

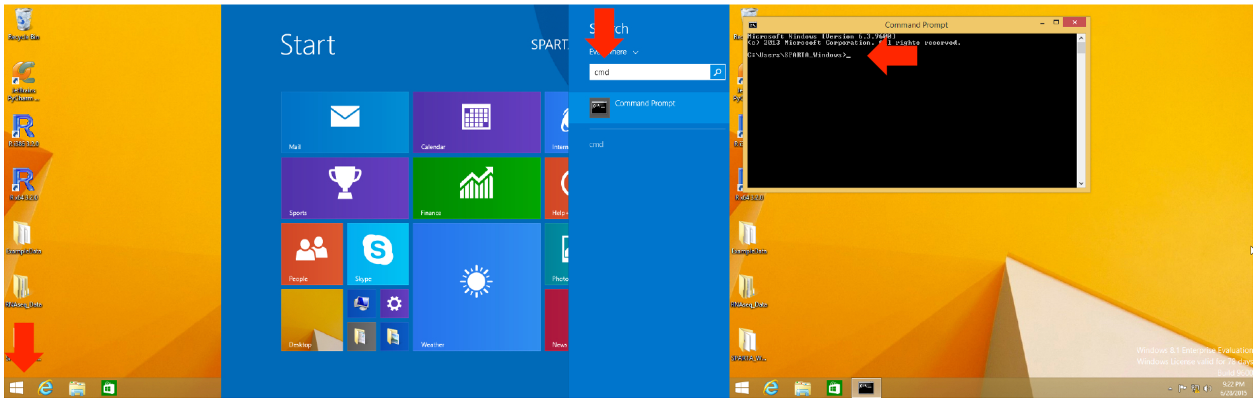
```
                      Default is --type=exon (suitable for RNA-seq analysis)
  --idattr=IDATTR     Feature ID from the GTF file to identify counts in the
                      output table Default is --idattr=gene_id. Usage:
                      --idattr=<id attribute>
  --mode=MODE         Mode to handle reads overlapping more than one
                      feature. Default is --mode=union. Usage: --mode=union
                      /intersection-strict/intersection-nonempty
```

1.1.2 Windows tutorial

Download the workflow: SPARTA for Windows

Introduction

Many bioinformatics software packages and workflows require the user to utilize them from the command line or terminal. SPARTA is no different. The reason the command line interface is utilized is that a great deal of power and flexibility can be gained without the use of a graphical user interface (GUI). Further, a GUI can be difficult to implement across various platforms. To find the command line interface/Terminal on Windows, go to Windows start button -> Search -> Type in: cmd -> Terminal is now open to enter commands.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\SPARTA_Windows>_
```

Decompress the SPARTA_Windows-master.zip file by double-clicking on it. Now, drag and drop the decompressed folder onto your desktop.

SPARTA expects either compressed (.gz) or uncompressed FASTQ files (.fq or .fastq) as input, with a reference genome file in FASTA format and a genome feature file (.gtf) within the folder that contains the input data. To see an example of appropriate input data, look inside the ExampleData folder within the SPARTA_Windows-master folder.

To download a reference genome and genome feature file for your favorite bacteria, go to the Ensembl website. The reference genome and feature file are already present for the ExampleData.

Basic Terminal Commands

Let’s have a look at some basic Terminal commands, we will cover the commands necessary to:

1. Move through folders

2. List the contents of a folder

3. Make new folders

4. Rename files/folders

5. Delete files/folders

	Command	What it does	Examples
1.	cd	Change directory/folder	cd ~ (this changes to your home directory); cd .. (this goes back one folder)
2.	dir	List the contents of a folder	dir
3.	mkdir	Make a new directory/folder	mkdir NewFolder (this will make a new folder called ‘NewFolder’ in your current directory)
4.	move	Rename or move a file from one name to another	move file1 file2 (this will rename/move file1 to file2)
5.	rm	Remove a file (rmdir is the command to remove a folder)	rm file1 (remove file1); rmdir folder1 (remove folder1)

Basic Command Prompt Commands:

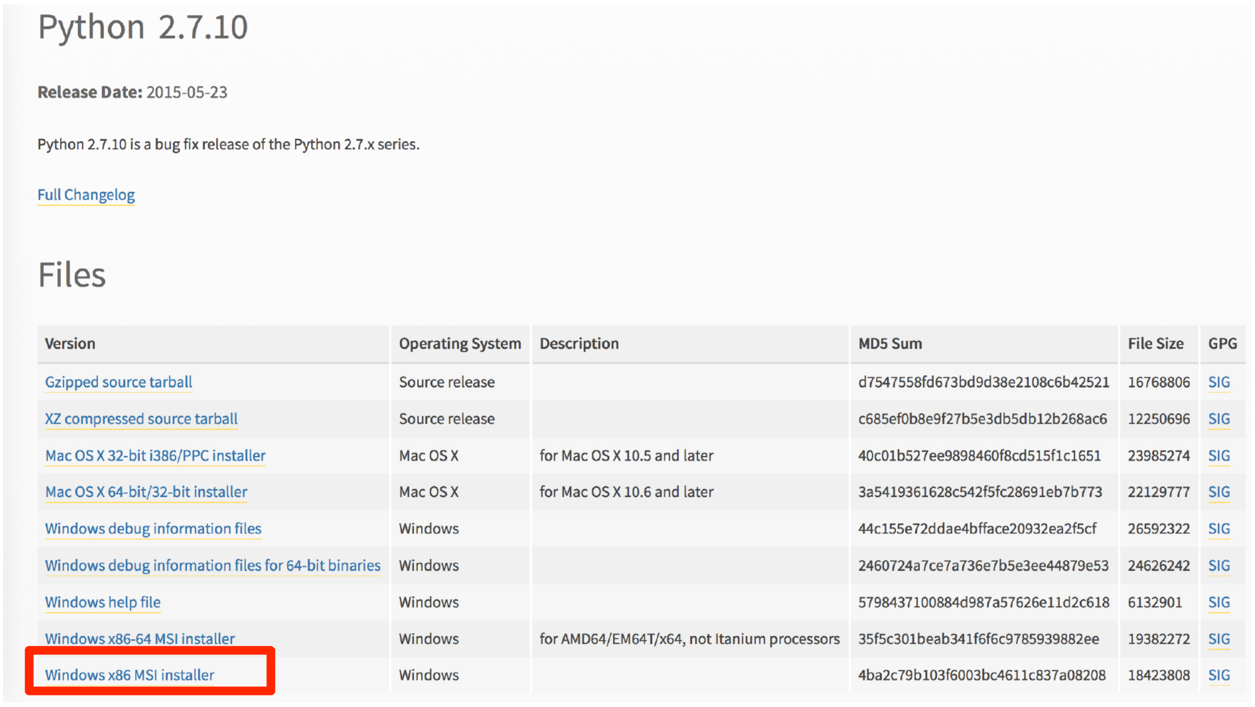
```
x /? = provides syntax info and complete list of all parameters for x (a command, like “cd”)
cd = change directory
cd .. = move to the parent directory
cd\ = move to the root of current drive
cd x = move to the current\x directory
cd z: = change to the z root directory (as opposed to c:\)
copy x y = copy file x to directory y (Ex: D:\games\galaga.exe C:\programs[\awesome.exe]), [] = optio
copy file con = display file contents in console
copy con file.txt = create text file in the console window, end with ctrl+z (^z or F6)
date = change the date
del = delete/erase
del x = deletes all files/folders fitting x
del . = deletes all files within current directory
del *.* = deletes all files within current directory
dir = display contents of current directory (Ex: dir [c:][\programs]), [] = optional
dir *.txt = list all .txt files in current directory
dir *.? = list all files with extensions one character in length in current directory
dir /w /p *.* = display all contents one screen at a time
dir | more = display all contents one line at a time
dir /? = provides syntax info and complete list of all dir parameters
echo = send command line input to display (by default)
echo sometext >> somefile.txt = append line(s) of text to any file
echo sometext > somefile.txt = overwrites file with sometext
erase = delete/erase
exit = exit the command prompt
filename.txt = opens filename.txt in current directory in Notepad (or default .txt program)
format z: = format z drive [Ex: use to format a disc or flash drive]
mkdir x = make directory x in current directory
move x y = more or rename x to y
q = escapes sequential display of contents (i.e. the more parameter)
rd x = remove/delete directory x if it’s empty
ren x y = rename file x to y
```

```
time = change the time
type file = display the contents of the file 'file' (displays file contents in console)
type file |more = display the contents one line at a time
```

Ref. sheet from: http://blog.simplyadvanced.net/cheat-sheet-for-windows-command-prompt/

Install Dependencies

The SPARTA workflow requires a few things in order to run: Python, Java, NumPy, and R. If you already have these installed, great! If you don't, let's start by downloading the latest version of Python 2 (see image below). You will want to download and install the red boxed version of Python 2. Follow the prompts to install Python with the default values.



Python 2.7.10

Release Date: 2015-05-23

Python 2.7.10 is a bug fix release of the Python 2.7.x series.

Full Changelog

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		d7547558fd673bd9d38e2108c6b42521	16768806	SIG
XZ compressed source tarball	Source release		c685ef0b8e9f27b5e3db5db12b268ac6	12250696	SIG
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later	40c01b527ee9898460f8cd515f1c1651	23985274	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	3a5419361628c542f5fc28691eb7b773	22129777	SIG
Windows debug information files	Windows		44c155e72ddae4bfface20932ea2f5cf	26592322	SIG
Windows debug information files for 64-bit binaries	Windows		2460724a7ce7a736e7b5e3ee44879e53	24626242	SIG
Windows help file	Windows		5798437100884d987a57626e11d2c618	6132901	SIG
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64, not Itanium processors	35f5c301beab341f6f6c9785939882ee	19382272	SIG
Windows x86 MSI installer	Windows		4ba2c79b103f6003bc4611c837a08208	18423808	SIG

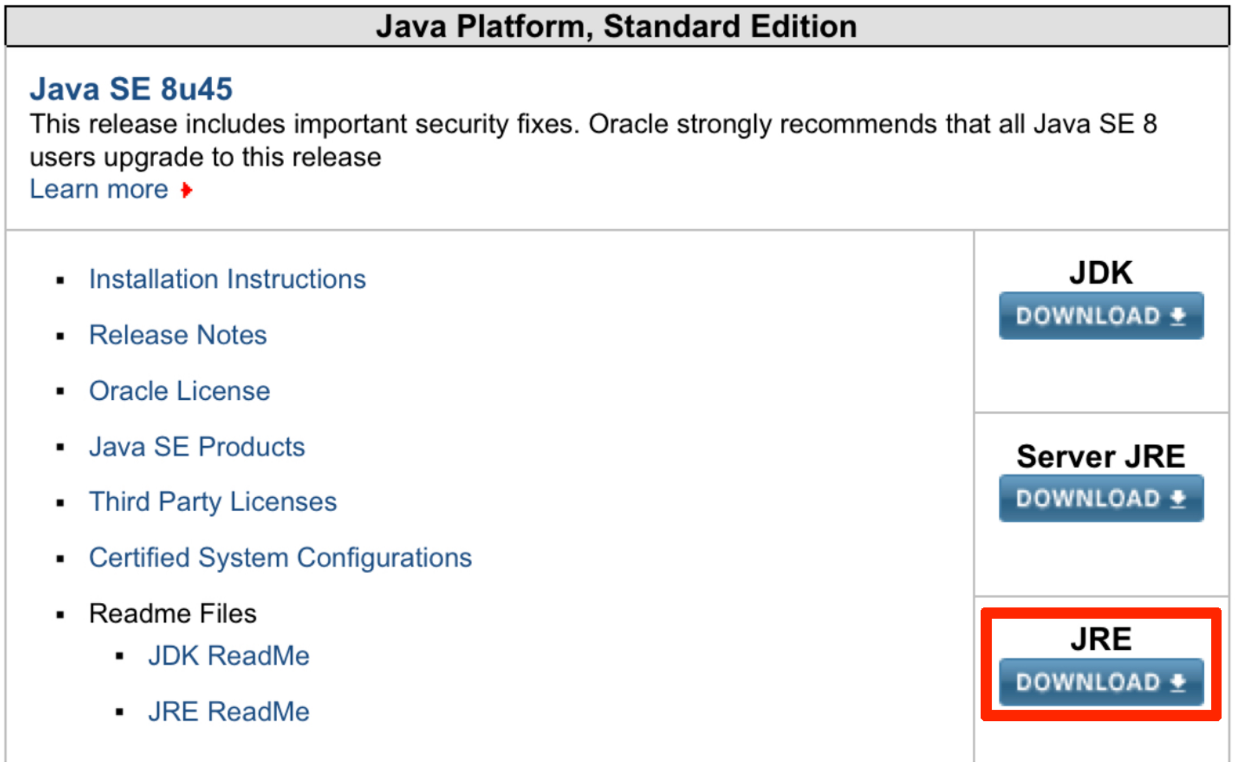
Great! Let's check and see if Java is already installed on your system. Open up the terminal, (if you don't remember how to do this, head back to the *Introduction*) and type:

```
java -version
```

If Java is already installed, it will produce some output that looks like this:

```
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
```

If the output does *not* look something like this, Java is likely not installed and two of the tools require Java to function (Trimmomatic and FastQC). Let's download and install a suitable version of Java (see image below). You will want to download and install the red boxed version of Java JRE. You will also need to click on the button (red arrow) to accept the terms and conditions of using Java JRE. Follow the prompts to install Java.



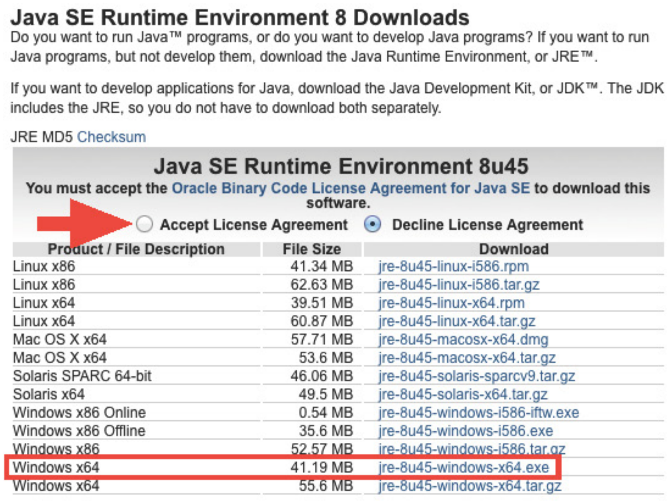
Java Platform, Standard Edition

Java SE 8u45

This release includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release

Learn more

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations
- Readme Files
  - JDK ReadMe
  - JRE ReadMe



Java SE Runtime Environment 8 Downloads

Do you want to run Java™ programs, or do you want to develop Java programs? If you want to run Java programs, but not develop them, download the Java Runtime Environment, or JRE™.

If you want to develop applications for Java, download the Java Development Kit, or JDK™. The JDK includes the JRE, so you do not have to download both separately.

JRE MD5 Checksum

Java SE Runtime Environment 8u45

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux x86	41.34 MB	jre-8u45-linux-i586.rpm
Linux x86	62.63 MB	jre-8u45-linux-i586.tar.gz
Linux x64	39.51 MB	jre-8u45-linux-x64.rpm
Linux x64	60.87 MB	jre-8u45-linux-x64.tar.gz
Mac OS X x64	57.71 MB	jre-8u45-macosx-x64.dmg
Mac OS X x64	53.6 MB	jre-8u45-macosx-x64.tar.gz
Solaris SPARC 64-bit	46.06 MB	jre-8u45-solaris-sparcv9.tar.gz
Solaris x64	49.5 MB	jre-8u45-solaris-x64.tar.gz
Windows x86 Online	0.54 MB	jre-8u45-windows-i586-iftw.exe
Windows x86 Offline	35.6 MB	jre-8u45-windows-i586.exe
Windows x86	52.57 MB	jre-8u45-windows-i586.tar.gz
Windows x64	41.19 MB	jre-8u45-windows-x64.exe
Windows x64	55.6 MB	jre-8u45-windows-x64.tar.gz

To install the remaining dependencies, SPARTA is distributed with installers for each remaining piece of software, however, there is an ideal order with which to install them.

Navigate to the SPARTA_Windows-master folder and then into the "Software_To_Install" folder. Inside this folder is a series of executable installers. Double-click and install them in the following order:

1. numpy
2. vcredist
3. HTSeq
4. R (see the "Important" below before installing)
5. gzip

Important: When installing R, **make sure that the 32-bit files are also installed**. You may have to check the box when the installer presents you with what files to install.

Now, there is one remaining batch file called "add_python_and_R_to_path.bat". This will add the Python, R, and gzip executables to your path so you can run them from the terminal. To execute this script, right-click on the file and then click on the option called "Run as administrator". Windows may warn you that this script is unsafe because it is from an unknown developer. Click on the "Details" button and then click on "Run anyway".

Note: If this script is not run, SPARTA will not function properly.

Congratulations! You've installed the necessary dependencies to run SPARTA!

Initializing SPARTA

Once SPARTA is initialized, the workflow will seek to identify that all of the necessary dependencies are met. If they are not satisfied, a message specific to what is not installed will appear as output in the terminal window.

To initialize SPARTA, go to the Terminal and navigate to the SPARTA_Windows-master folder on your desktop by typing:

```
cd Desktop\SPARTA_Windows-master
```

To start the workflow, type:

```
python SPARTA.py
```

This will start the software and check for dependencies.

Analyzing Example Data

SPARTA is distributed with some example data. Specifically, it is the first 100,000 reads of each sample from Baker et al..

To begin the analysis, navigate into the SPARTA_Mac-master folder and drag and drop the folder called "Example-Data" out onto the desktop.

If you haven't already, *initialize SPARTA* from the Terminal.

If all the *dependencies* are met, SPARTA will pause and prompt the user:

```
Is the RNAseq data in a folder on the Desktop? (Y or N):
```

Type:

```
Y
```

Hit Enter/Return

Note: SPARTA assumes the data is located in a folder on the desktop by default. It is easiest if all future analyses have the data in a folder (WITHOUT SPACES IN THE NAME) on the desktop.

Now it will prompt the user for the name of the folder:

```
What is the name of the folder on the Desktop containing the RNAseq data?:
```

Type:

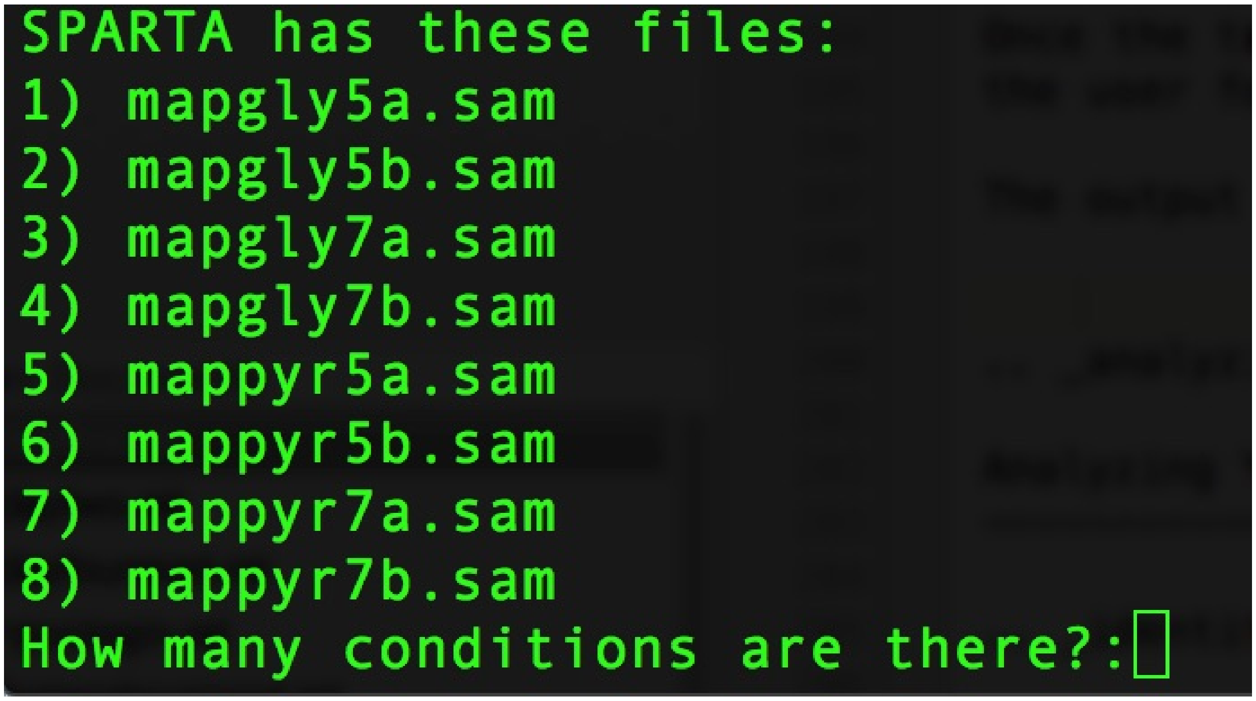
```
ExampleData
```

This is the name of the folder on the desktop that contains the input example data. Hit Enter/Return. From here, the software will trim, align, and count transcript abundance for each sample. All output/analyses are put in a folder that SPARTA generates on the desktop called "RNAseq_Data". Within this folder are separate folders for each SPARTA run that are denoted by the date (e.g. 2015-06-04). Within these folders are four more folders that separate each step of the analysis and are called: 1) QC, 2) Bowtie, 3) HTSeq, and 4) DEanalysis.

Note: There is a known issue here. FastQC will *not* run non-interactively on Windows (but feel free to contribute to the project and fix this issue!). It is important to QC your data and FastQC can be run interactively by navigating to the FastQC folder: SPARTA_Windows-master -> QC_analysis -> FastQC -> run_fastqc.bat. FastQC should now start and to analyze your trimmed files within FastQC: File -> Open -> RNAseq_Data -> dateofyourrun -> QC -> yourtrimmedfiles.

Once the trimming, alignment, and counting are complete, SPARTA will again pause and prompt the user for how many experimental conditions exist within the analysis.

The output at this point will look like this:



```
SPARTA has these files:
1) mapgly5a.sam
2) mapgly5b.sam
3) mapgly7a.sam
4) mapgly7b.sam
5) mappyr5a.sam
6) mappyr5b.sam
7) mappyr7a.sam
8) mappyr7b.sam
How many conditions are there?:
```

At the prompt that says:

```
How many conditions are there?:
```

Type:

```
4
```

Hit Enter/Return. There are 4 experimental conditions that we are considering:

1. Glycerol pH 7.0
2. Glycerol pH 5.7
3. Pyruvate pH 7.0
4. Pyruvate pH 5.7

Each condition has 2 replicates. The next prompt will read:

```
Enter the relevant file names, based on the names given in 'SPARTA has these files', with the replica
As an example, please see the 'conditions_input_example.txt' in the DEanalysis folder.
Once you have entered the file names, hit Enter/Return:
```

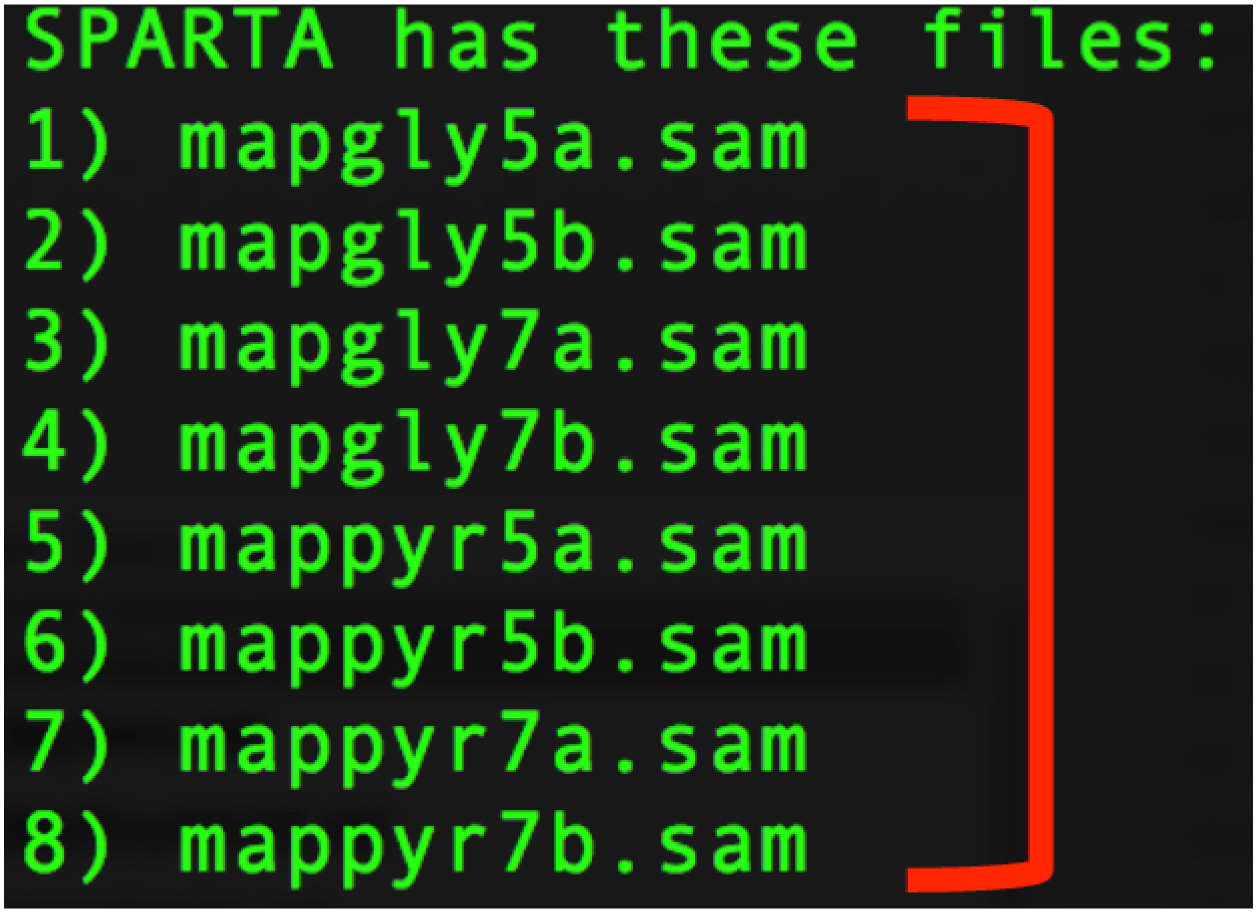
At this point, we need to edit a text file (conditions_input.txt) to tell SPARTA which file belongs to a given condition. To do this:

1. Navigate to the SPARTA output folder called RNAseq_Data located on the desktop
2. Go to the current run folder (will be the last folder listed if sorted by name)
3. Go into the DEanalysis folder
4. Open the conditions_input.txt file in a text editor (NOT MICROSOFT WORD) such as Notepad

The number of experimental conditions listed are based on the number entered at the prompt asking "How many conditions are there?:". Thus, in our case, there are 4. The contents of the file will look like:

```
Reference_Condition_Files:
Experimental_Condition_2_Files:
Experimental_Condition_3_Files:
Experimental_Condition_4_Files:
```

We now need to enter the file names of the replicates in each condition. These are comma-separated file names that correspond to the output given by SPARTA (denoted with red bracket)



```
SPARTA has these files:
1) mapgly5a.sam
2) mapgly5b.sam
3) mapgly7a.sam
4) mapgly7b.sam
5) mappyr5a.sam
6) mappyr5b.sam
7) mappyr7a.sam
8) mappyr7b.sam
```

Note: The file names are case-sensitive and must be spelled *exactly* as listed in the output given by SPARTA

Thus, when all the file names are inputed, the conditions_input.txt file should look like this:

```
Reference_Condition_Files: mapgly7a.sam, mapgly7b.sam
Experimental_Condition_2_Files:mapgly5a.sam, mapgly5b.sam
Experimental_Condition_3_Files:mappyr7a.sam, mappyr7b.sam
Experimental_Condition_4_Files:mappyr5a.sam, mappyr5b.sam
```

Now, save the changes by going to File -> Save. Go back to the terminal and hit Enter/Return. From here, the workflow will perform the differential gene expression analysis through edgeR. If a batch effect may be present, the output will attempt to warn the user of the potential, unintended variable that *must* be accounted for before drawing experimental conclusions.

All the differential gene expression output is located in the RNAseq_Data -> date of your current run -> DEanalysis folder. The file output includes:

1. Differential gene expression tables

2. MDS plot (somewhat analogous to a principle component analysis plot) which will show whether your replicates group together and treatment groups separate based on the treatment
3. BCV plot (biological coefficient of variation) to look at gene level variation between samples

Congratulations! You've analyzed RNA-seq data from raw reads to differential gene expression!

Analyzing Your Data

If you haven't already, we recommend working through the *example data analysis* first before attempting to work through your own data set to familiarize yourself with the workflow.

As stated in the *Introduction*, SPARTA expects either compressed (.gz) or uncompressed FASTQ files (.fq or .fastq) as input, with a reference genome file in FASTA format and a genome feature file (.gtf) within the folder that contains the input data on your desktop. To see an example of appropriate input data, look inside the ExampleData folder within the SPARTA_Windows-master folder.

Now, to analyze your own data, follow the steps to *initialize SPARTA*, and start the analysis!

If you would like to tweak the analysis options for a given step/tool, have a look at the *Altering Workflow Execution Options*.

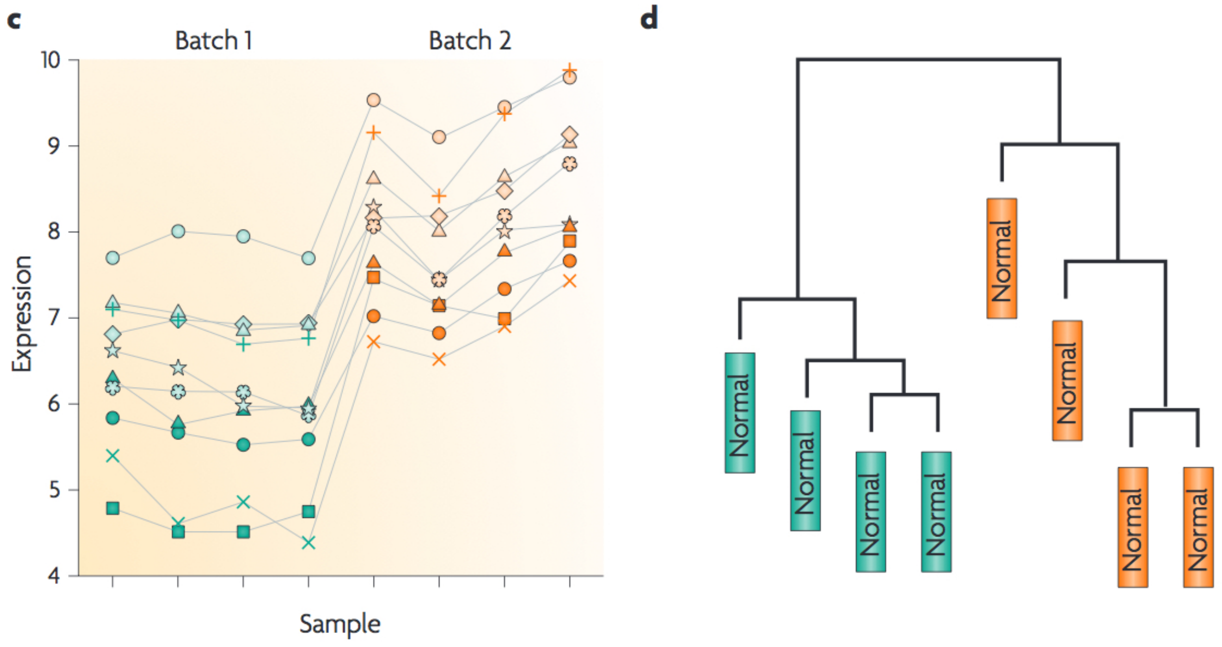
Identifying Potential Batch Effects

Batch effects can be a source of variation in RNA-seq data that can confound biological conclusions. In fact, there have been documented cases of batch effects present in published studies that led readers to be concerned for the validity of the results.

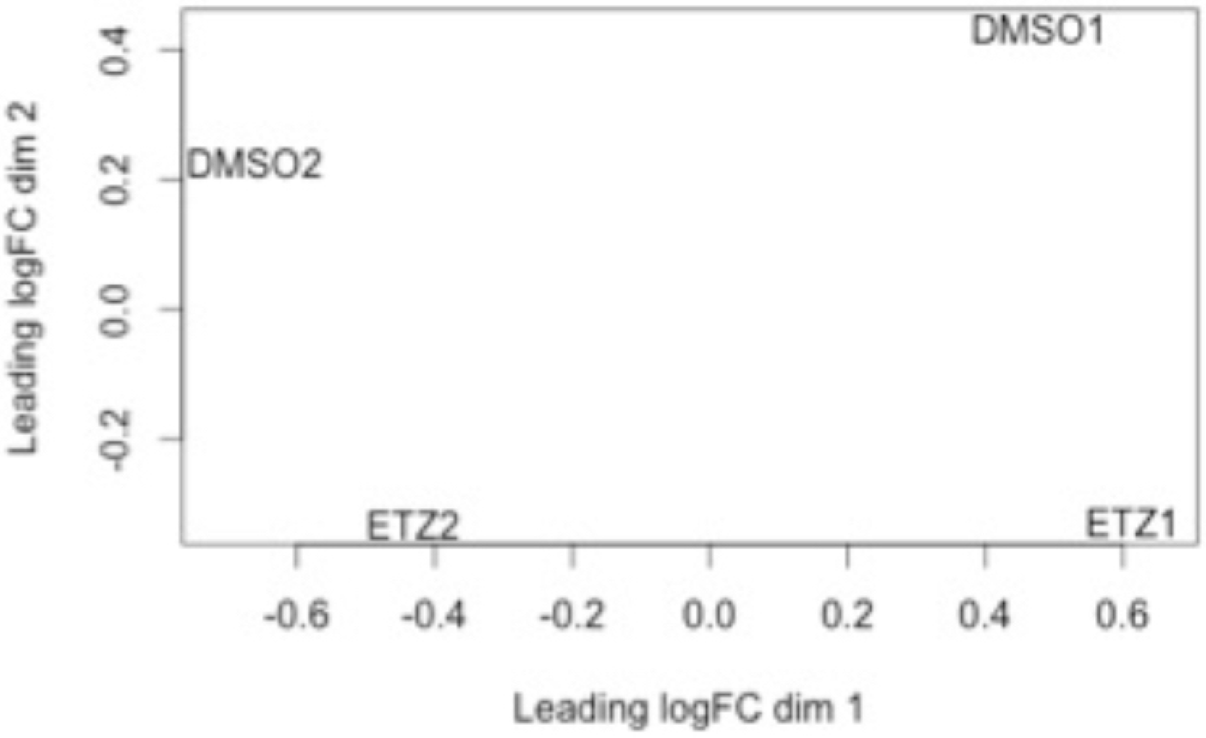
To quote a previously published paper in Nature Reviews Genetics, "Batch effects are sub-groups of measurements that have qualitatively different behaviour across conditions and are unrelated to the biological or scientific variables in a study. For example, batch effects may occur if a subset of experiments was run on Monday and another set on Tuesday, if two technicians were responsible for different subsets of the experiments or if two different lots of reagents, chips or instruments were used."

Thus, it is paramount that one address batch effects within their data before drawing biological conclusions from a specific RNA-seq experiment. To illustrate what a batch effect may look like within the data, we will utilize several different plots.

This first plot comes from the Nature Reviews Genetics paper where they examine Affymetrix data from a published bladder cancer study. You can quickly see that panels C and D from Figure 1 show that samples from batch 1 (blue) cluster together based on gene expression and samples from batch 2 (orange) cluster together.



Within RNA-seq data, using SPARTA and the MDS plot generated by edgeR, another example of batch effects within a study comparing *Mycobacterium tuberculosis* treated with a compound, we can clearly see that the mock-treated samples (DMSO) and compound-treated samples (ETZ) separate based on batch (A vs B) instead of by treatment. Ideally, we would have the samples group together based on treatment as opposed to batch.



If a potential batch effect is detected in the data set, SPARTA will output a message into the terminal that says:

```
IMPORTANT! YOU MAY HAVE A BATCH EFFECT! PLEASE LOOK AT THE MDS PLOT!
```

If this occurs, have a look at the MDS plot in the RNAseq_Data folder -> date of current run -> DEanalysis folder -> MDSplot.png

From here, you will want to adjust your model to account for the batch effect. Within edgeR, this can be accomplished through an additive linear model. The documentation for edgeR contains a tutorial on how to deal with batch effects that can be found here.

Future implementations of SPARTA will include the ability to adjust for batch effects.

Altering Workflow Execution Options

SPARTA is capable of allowing the user to alter the parameters associated with each analysis step to be tailored to specific use cases. Below are the different parameters that can be altered and their usage.

Options:

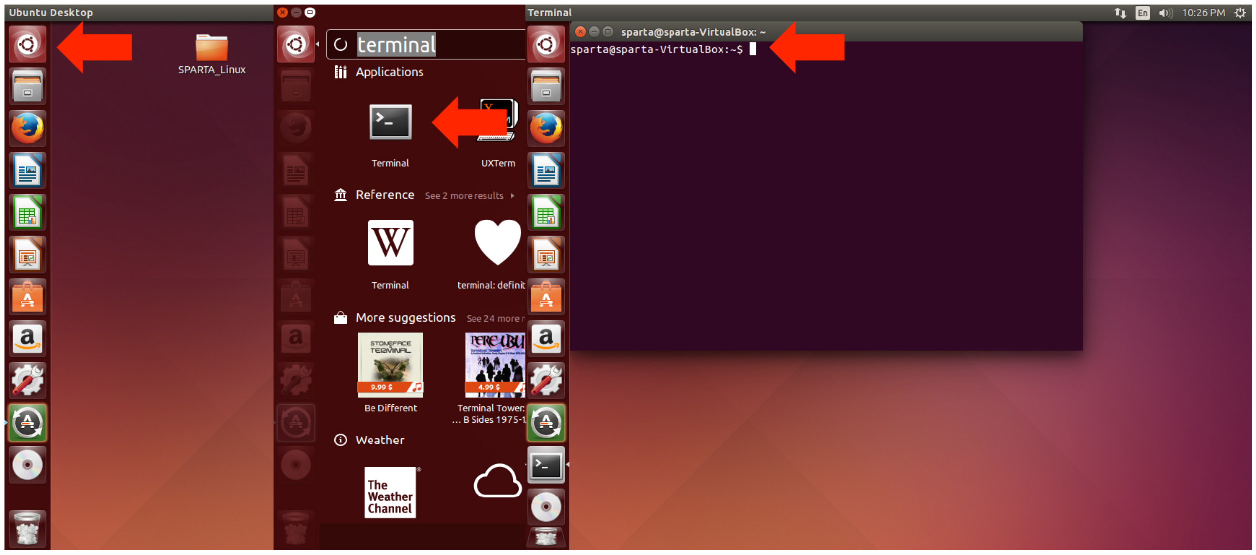
```
Usage: python SPARTA.py [options]

      Simple Program for Automated reference-based bacterial RNA-seq Transcriptome
      Analysis (SPARTA)

-h, --help            show this help message and exit
--cleanup             Clean up the intermediate files to save space. Default
                      action is to retain the intermediate files.
--verbose             Display more output for each step of the analysis.
--noninteractive      Non-interactive mode. This is for running SPARTA
                      without any user input. Assumes data is on the
                      desktop. If this option is specified, you must fill
                      out the configuration file (ConfigFile.txt) with the
                      appropriate experimental conditions in the SPARTA
                      folder.
--threads=THREADS     Define the number of threads that SPARTA should run
                      with. This will enable some speed-up on multi-
                      processor machines. As a generality, define the number
                      of threads as the same number of cores in your
                      computer. Default is 2.

Trimmomatic options:
  The order the options will be run are: ILLUMINACLIP, LEADING,
  TRAILING, SLIDINGWINDOW, MINLEN

  --clip=ILLUMINACLIP
                      ILLUMINACLIP options. MiSeq & HiSeq usually
                      TruSeq3.fa; GAII usually TruSeq2.fa. Default is
                      ILLUMINACLIP:TruSeq3-SE.fa:2:30:10. Usage:
                      --clip=<adapterseqs>:<seed mismatches>:<palindrome
                      clip threshold>:<simple clip threshold>
  --lead=LEADING      Set the minimun quality required to keep a base.
                      Default is LEADING=3. Usage: --lead=<quality>
  --trail=TRAILING    Set the minimum quality required to keep a base.
                      Default is TRAILING=3. Usage: --trail=<quality>
  --slidewin=SLIDINGWINDOW
                      SLIDINGWINDOW options. Default is SLIDINGWINDOW:4:15.
                      Usage: --slidewin=<window_size>:<required_quality>
  --minlentrim=MINLENTRIM
                      Set the minimum read length to keep in base pairs.
                      Default is 36. Usage: --minlentrim=<readlength>

Bowtie options:
  --mismatch=MISMATCH
                      Output alignments with at most a defined number of
                      mismatches. Usage: --mismatch=<integer_value>
  --otherbowtieoptions=OTHERBOWTIEOPTIONS
                      Bowtie has so many options that it is not worth
                      listing them here. Go to http://bowtie-
                      bio.sourceforge.net/manual.shtml#command-line for the
                      manual and all available options. Usage:
                      --otherbowtieoptions='all options inputed as a string
                      (note the quotes!)'

HTSeq options:
  --stranded=STRANDED
                      Stranded options: yes, no, reverse. Default is
                      --stranded=reverse. Usage: --stranded=yes/no/reverse
  --order=ORDER       Order options: name, pos. Usage: --order=name/pos.
  --minqual=MINQUAL   Skip all reads with quality lower than the given
                      value. Default is --minqual=10. Usage:
                      --minqual=<value>
  --type=TYPE         The feature type (3rd column in GTF file) to be used.
                      Default is --type=exon (suitable for RNA-seq analysis)
  --idattr=IDATTR     Feature ID from the GTF file to identify counts in the
                      output table Default is --idattr=gene_id. Usage:
                      --idattr=<id attribute>
  --mode=MODE         Mode to handle reads overlapping more than one
                      feature. Default is --mode=union. Usage: --mode=union
                      /intersection-strict/intersection-nonempty
```

1.1.3 Linux tutorial

Download the workflow: SPARTA for Linux

Introduction

Many bioinformatics software packages and workflows require the user to utilize them from the command line or terminal. SPARTA is no different. The reason the command line interface is utilized is that a great deal of power and flexibility can be gained without the use of a graphical user interface (GUI). Further, a GUI can be difficult to implement across various platforms. To find the command line interface/Terminal on Linux (shown in Ubuntu with red arrows), go to "Search your computer and online sources" button -> Search for "terminal" -> Click on Terminal -> Terminal is now open and ready to enter commands (might just be worth dragging it onto your dock).



```
sparta@sparta-VirtualBox:~$
```

Decompress the SPARTA_Linux-master.zip file by clicking on it and extracting all the files to the desktop.

SPARTA expects either compressed (.gz) or uncompressed FASTQ files (.fq or .fastq) as input, with a reference genome file in FASTA format and a genome feature file (.gtf) within the folder that contains the input data. To see an example of appropriate input data, look inside the ExampleData folder within the SPARTA_Linux-master folder.

To download a reference genome and genome feature file for your favorite bacteria, go to the Ensembl website. The reference genome and feature file are already present for the ExampleData.

Basic Terminal Commands

Let's have a look at some basic Terminal commands, we will cover the commands necessary to:

1. Move through folders

2. List the contents of a folder

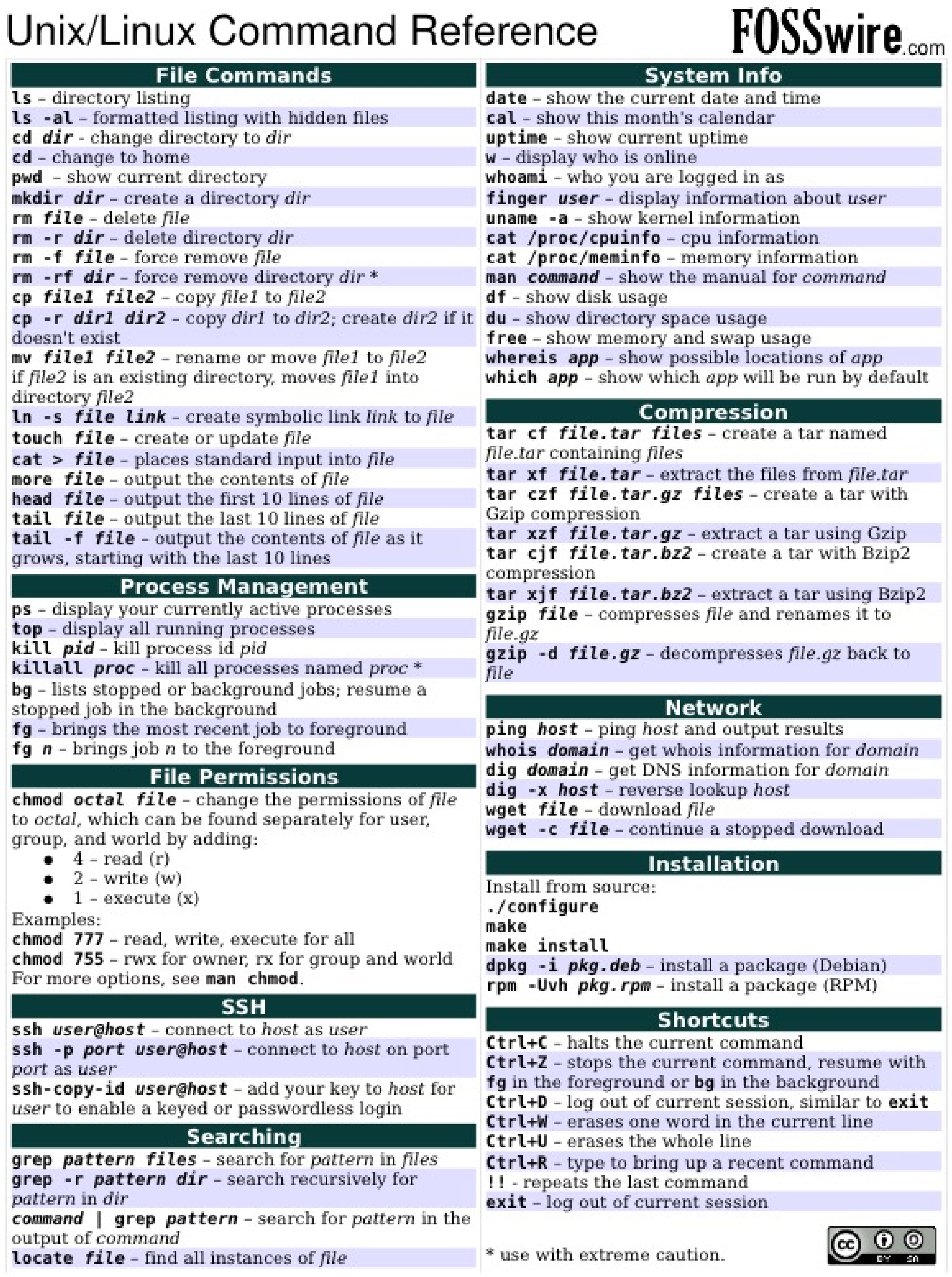
3. Make new folders

4. Rename files/folders

5. Delete files/folders

	Command	What it does	Examples
1.	cd	Change directory/folder	cd ~ (this changes to your home directory); cd .. (this goes back one folder)
2.	ls	List the contents of a folder	ls
3.	mkdir	Make a new directory/folder	mkdir NewFolder (this will make a new folder called 'NewFolder' in your current directory)
4.	mv	Rename or move a file from one name to another	mv file1 file2 (this will rename/move file1 to file2)
5.	rm	Remove a file (add the -r flag to remove a folder)	rm file1 (remove file1); rm -r folder1 (remove folder1)

Command reference sheet



Unix/Linux Command Reference

FOSSwire.com

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2*
if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

chmod *octal file* – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
For more options, see **man chmod**.

SSH

ssh *user@host* – connect to *host* as *user*
ssh -p *port user@host* – connect to *host* on port *port* as *user*
ssh-copy-id *user@host* – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* – search for *pattern* in *files*
grep -r *pattern dir* – search recursively for *pattern* in *dir*
command* | grep *pattern – search for *pattern* in the output of *command*
locate *file* – find all instances of *file*

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*
wget *file* – download *file*
wget -c *file* – continue a stopped download

Installation

Install from source:

```
./configure
make
make install
```

dpkg -i *pkg.deb* – install a package (Debian)
rpm -Uvh *pkg.rpm* – install a package (RPM)

Shortcuts

Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

* use with extreme caution.

Ref. sheet from: http://files.fosswire.com/2007/08/fwunixref.pdf

Install Dependencies

The SPARTA workflow requires a few things in order to run: Python, Java, NumPy, and R. If you already have these installed, great! If you don't, let's start by downloading and installing the dependencies by running the bash script called "install_dependencies.sh".

To run this script, navigate to the SPARTA_Linux-master folder on the desktop:

```
cd ~/Desktop/SPARTA_Linux-master
```

Now, type:

```
bash install_dependencies.sh
```

This will update, download, and install the necessary dependencies to run SPARTA.

Congratulations! You've installed the necessary dependencies to run SPARTA!

Initializing SPARTA

Once SPARTA is initialized, the workflow will seek to identify that all of the necessary dependencies are met. If they are not satisfied, a message specific to what is not installed will appear as output in the terminal window.

To initialize SPARTA, go to the Terminal and navigate to the SPARTA_Linux-master folder on your desktop by typing:

```
cd ~/Desktop/SPARTA_Linux-master
```

To start the workflow, type:

```
python SPARTA.py
```

This will start the software and check for dependencies.

Analyzing Example Data

SPARTA is distributed with some example data. Specifically, it is the first 100,000 reads of each sample from Baker et al..

To begin the analysis, navigate into the SPARTA_Linux-master folder and drag and drop the folder called "Example-Data" out onto the desktop.

If you haven't already, *initialize SPARTA* from the Terminal.

If all the *dependencies* are met, SPARTA will pause and prompt the user:

```
Is the RNAseq data in a folder on the Desktop? (Y or N):
```

Type:

```
Y
```

Hit Enter/Return

Note: SPARTA assumes the data is located in a folder on the desktop by default. It is easiest if all future analyses have the data in a folder (WITHOUT SPACES IN THE NAME) on the desktop.

Now it will prompt the user for the name of the folder:

```
What is the name of the folder on the Desktop containing the RNAseq data?:
```

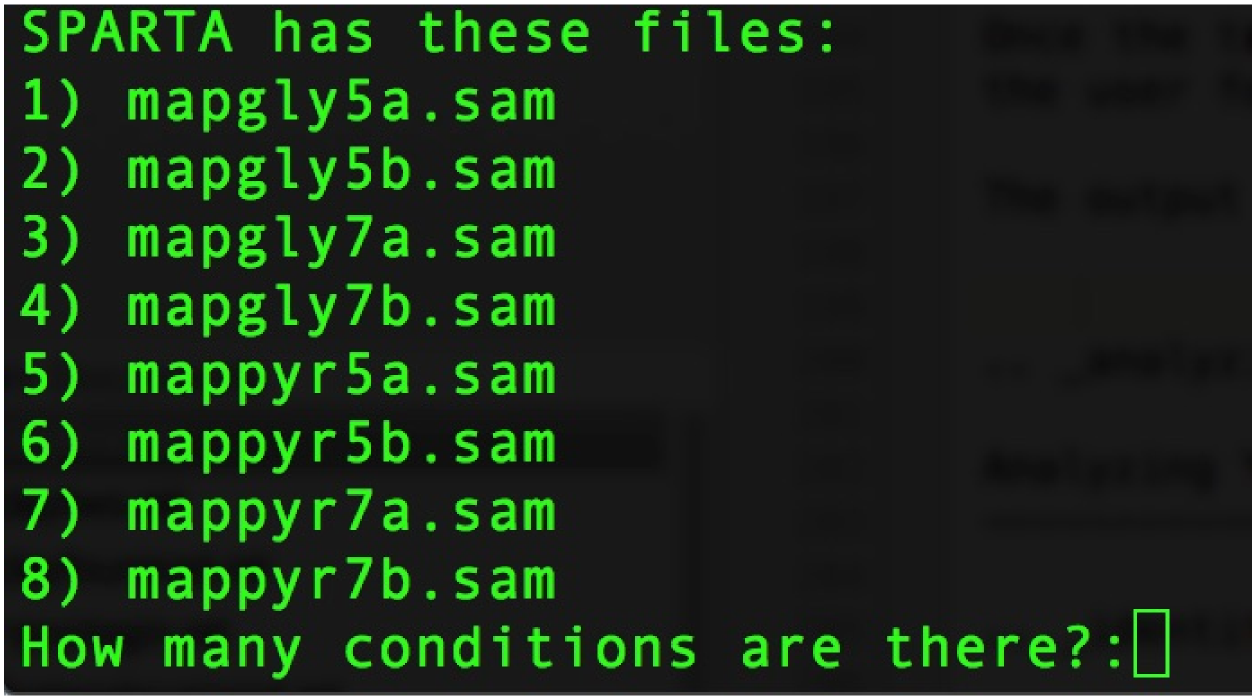
Type:

```
ExampleData
```

This is the name of the folder on the desktop that contains the input example data. Hit Enter/Return. From here, the software will trim, QC, align, and count transcript abundance for each sample. All output/analyses are put in a folder that SPARTA generates on the desktop called "RNAseq_Data". Within this folder are separate folders for each SPARTA run that are denoted by the date (e.g. 2015-06-04). Within these folders are four more folders that separate each step of the analysis and are called: 1) QC, 2) Bowtie, 3) HTSeq, and 4) DEanalysis.

Once the trimming, QC, alignment, and counting are complete, SPARTA will again pause and prompt the user for how many experimental conditions exist within the analysis.

The output at this point will look like this:



```
SPARTA has these files:
1) mapgly5a.sam
2) mapgly5b.sam
3) mapgly7a.sam
4) mapgly7b.sam
5) mappyr5a.sam
6) mappyr5b.sam
7) mappyr7a.sam
8) mappyr7b.sam
How many conditions are there?:
```

At the prompt that says:

```
How many conditions are there?:
```

Type:

```
4
```

Hit Enter/Return. There are 4 experimental conditions that we are considering:

1. Glycerol pH 7.0
2. Glycerol pH 5.7
3. Pyruvate pH 7.0
4. Pyruvate pH 5.7

Each condition has 2 replicates. The next prompt will read:

```
Enter the relevant file names, based on the names given in 'SPARTA has these files', with the replica
As an example, please see the 'conditions_input_example.txt' in the DEanalysis folder.
Once you have entered the file names, hit Enter/Return:
```

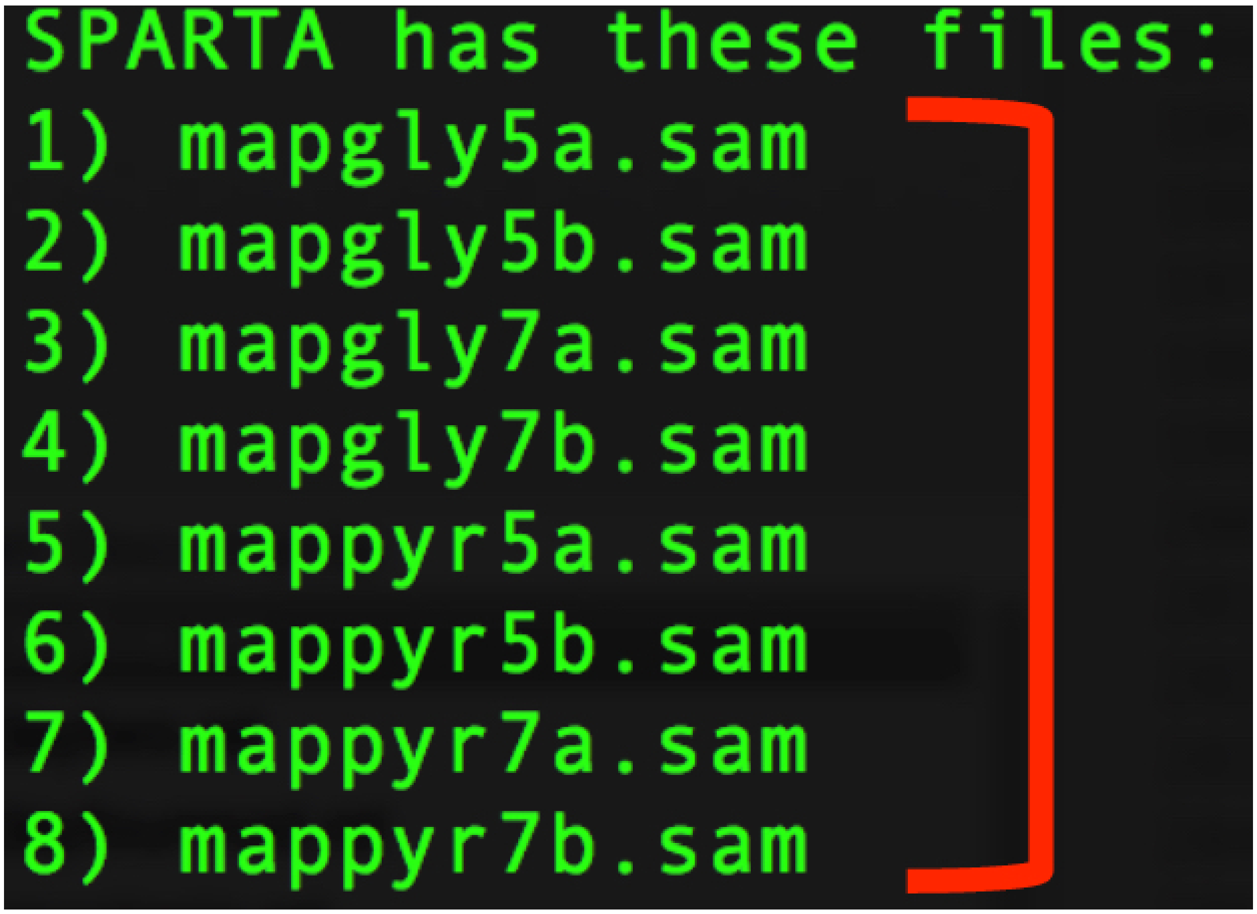
At this point, we need to edit a text file (conditions_input.txt) to tell SPARTA which file belongs to a given condition. To do this:

1. Navigate to the SPARTA output folder called RNAseq_Data located on the desktop
2. Go to the current run folder (will be the last folder listed if sorted by name)
3. Go into the DEanalysis folder
4. Open the conditions_input.txt file in a text editor (NOT MICROSOFT WORD) such as gedit

The number of experimental conditions listed are based on the number entered at the prompt asking "How many conditions are there?:". Thus, in our case, there are 4. The contents of the file will look like:

```
Reference_Condition_Files:
Experimental_Condition_2_Files:
Experimental_Condition_3_Files:
Experimental_Condition_4_Files:
```

We now need to enter the file names of the replicates in each condition. These are comma-separated file names that correspond to the output given by SPARTA (denoted with red bracket)



```
SPARTA has these files:
1) mapgly5a.sam
2) mapgly5b.sam
3) mapgly7a.sam
4) mapgly7b.sam
5) mappyr5a.sam
6) mappyr5b.sam
7) mappyr7a.sam
8) mappyr7b.sam
```

Note: The file names are case-sensitive and must be spelled *exactly* as listed in the output given by SPARTA

Thus, when all the file names are inputed, the conditions_input.txt file should look like this:

```
Reference_Condition_Files: mapgly7a.sam, mapgly7b.sam
Experimental_Condition_2_Files:mapgly5a.sam, mapgly5b.sam
Experimental_Condition_3_Files:mappyr7a.sam, mappyr7b.sam
Experimental_Condition_4_Files:mappyr5a.sam, mappyr5b.sam
```

Now, save the changes by going to File -> Save. Go back to the terminal and hit Enter/Return. From here, the workflow will perform the differential gene expression analysis through edgeR. If a batch effect may be present, the output will attempt to warn the user of the potential, unintended variable that *must* be accounted for before drawing experimental conclusions.

All the differential gene expression output is located in the RNAseq_Data -> date of your current run -> DEanalysis folder. The file output includes:

1. Differential gene expression tables
2. MDS plot (somewhat analogous to a principle component analysis plot) which will show whether your replicates group together and treatment groups separate based on the treatment
3. BCV plot (biological coefficient of variation) to look at gene level variation between samples

Congratulations! You've analyzed RNA-seq data from raw reads to differential gene expression!

Analyzing Your Data

If you haven't already, we recommend working through the *example data analysis* first before attempting to work through your own data set to familiarize yourself with the workflow.

As stated in the *Introduction*, SPARTA expects either compressed (.gz) or uncompressed FASTQ files (.fq or .fastq) as input, with a reference genome file in FASTA format and a genome feature file (.gtf) within the folder that contains the input data on your desktop. To see an example of appropriate input data, look inside the ExampleData folder within the SPARTA_Mac-master folder.

Now, to analyze your own data, follow the steps to *initialize SPARTA*, and start the analysis!

If you would like to tweak the analysis options for a given step/tool, have a look at the *Altering Workflow Execution Options*.

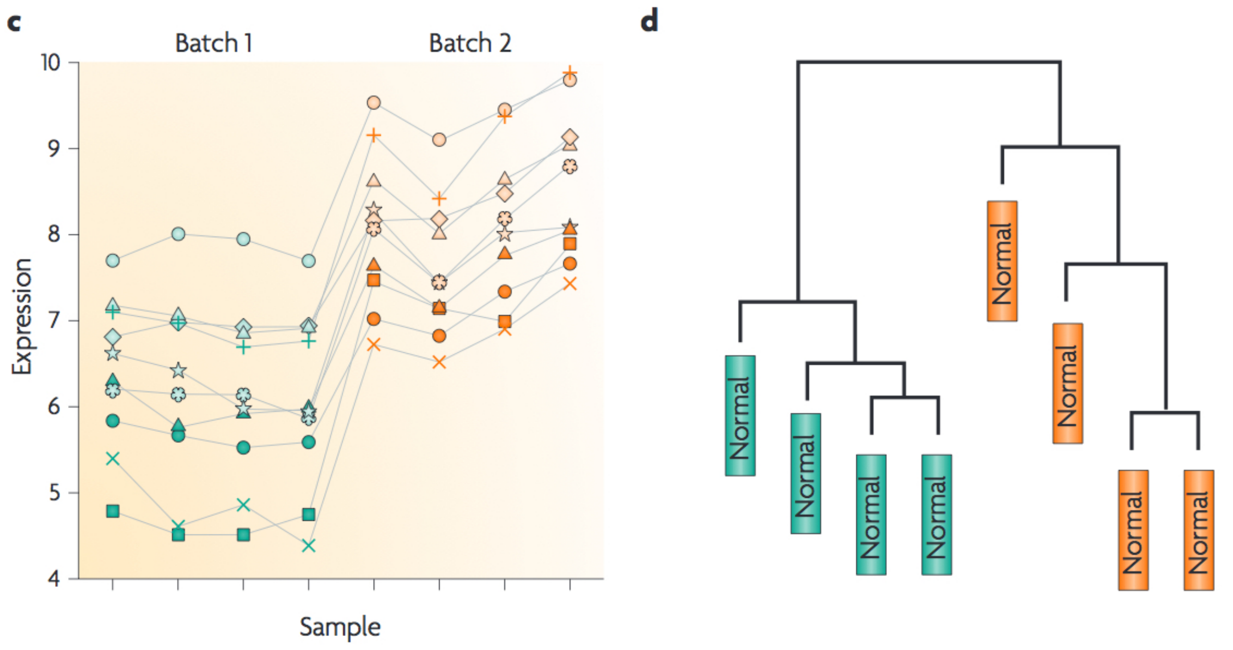
Identifying Potential Batch Effects

Batch effects can be a source of variation in RNA-seq data that can confound biological conclusions. In fact, there have been documented cases of batch effects present in published studies that led readers to be concerned for the validity of the results.

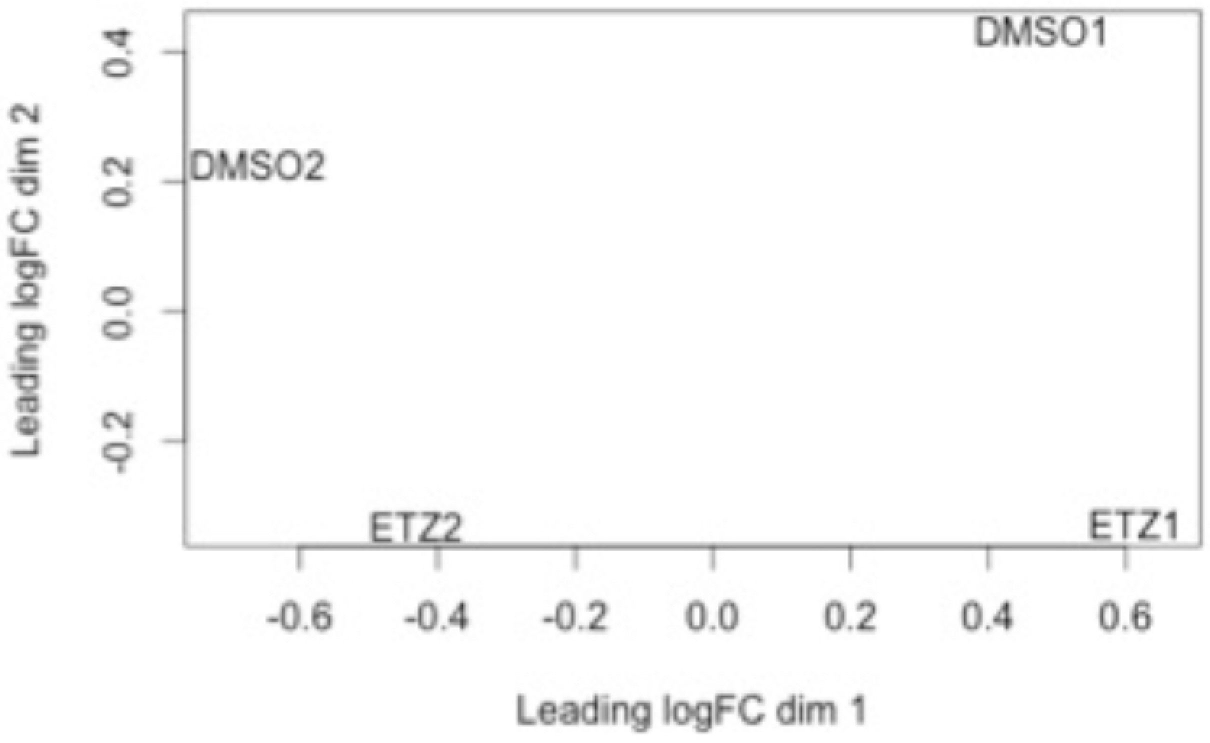
To quote a previously published paper in Nature Reviews Genetics, "Batch effects are sub-groups of measurements that have qualitatively different behaviour across conditions and are unrelated to the biological or scientific variables in a study. For example, batch effects may occur if a subset of experiments was run on Monday and another set on Tuesday, if two technicians were responsible for different subsets of the experiments or if two different lots of reagents, chips or instruments were used."

Thus, it is paramount that one address batch effects within their data before drawing biological conclusions from a specific RNA-seq experiment. To illustrate what a batch effect may look like within the data, we will utilize several different plots.

This first plot comes from the Nature Reviews Genetics paper where they examine Affymetrix data from a published bladder cancer study. You can quickly see that panels C and D from Figure 1 show that samples from batch 1 (blue) cluster together based on gene expression and samples from batch 2 (orange) cluster together.



Within RNA-seq data, using SPARTA and the MDS plot generated by edgeR, another example of batch effects within a study comparing *Mycobacterium tuberculosis* treated with a compound, we can clearly see that the mock-treated samples (DMSO) and compound-treated samples (ETZ) separate based on batch (A vs B) instead of by treatment. Ideally, we would have the samples group together based on treatment as opposed to batch.



If a potential batch effect is detected in the data set, SPARTA will output a message into the terminal that says:

```
IMPORTANT! YOU MAY HAVE A BATCH EFFECT! PLEASE LOOK AT THE MDS PLOT!
```

If this occurs, have a look at the MDS plot in the RNAseq_Data folder -> date of current run -> DEanalysis folder -> MDSplot.png

From here, you will want to adjust your model to account for the batch effect. Within edgeR, this can be accomplished through an additive linear model. The documentation for edgeR contains a tutorial on how to deal with batch effects that can be found here.

Future implementations of SPARTA will include the ability to adjust for batch effects.

Altering Workflow Execution Options

SPARTA is capable of allowing the user to alter the parameters associated with each analysis step to be tailored to specific use cases. Below are the different parameters that can be altered and their usage.

Options:

```
Usage: python SPARTA.py [options]

      Simple Program for Automated reference-based bacterial RNA-seq Transcriptome
      Analysis (SPARTA)

-h, --help            show this help message and exit
--cleanup             Clean up the intermediate files to save space. Default
                      action is to retain the intermediate files.
--verbose             Display more output for each step of the analysis.
--noninteractive      Non-interactive mode. This is for running SPARTA
                      without any user input. Assumes data is on the
                      desktop. If this option is specified, you must fill
                      out the configuration file (ConfigFile.txt) with the
                      appropriate experimental conditions in the SPARTA
                      folder.
--threads=THREADS     Define the number of threads that SPARTA should run
                      with. This will enable some speed-up on multi-
                      processor machines. As a generality, define the number
                      of threads as the same number of cores in your
                      computer. Default is 2.

Trimmomatic options:
  The order the options will be run are: ILLUMINACLIP, LEADING,
  TRAILING, SLIDINGWINDOW, MINLEN

  --clip=ILLUMINACLIP
                      ILLUMINACLIP options. MiSeq & HiSeq usually
                      TruSeq3.fa; GAII usually TruSeq2.fa. Default is
                      ILLUMINACLIP:TruSeq3-SE.fa:2:30:10. Usage:
                      --clip=<adapterseqs>:<seed mismatches>:<palindrome
                      clip threshold>:<simple clip threshold>
  --lead=LEADING      Set the minimun quality required to keep a base.
                      Default is LEADING=3. Usage: --lead=<quality>
  --trail=TRAILING    Set the minimum quality required to keep a base.
                      Default is TRAILING=3. Usage: --trail=<quality>
  --slidewin=SLIDINGWINDOW
                      SLIDINGWINDOW options. Default is SLIDINGWINDOW:4:15.
                      Usage: --slidewin=<window_size>:<required_quality>
  --minlentrim=MINLENTRIM
                      Set the minimum read length to keep in base pairs.
                      Default is 36. Usage: --minlentrim=<readlength>

Bowtie options:
  --mismatch=MISMATCH
                      Output alignments with at most a defined number of
                      mismatches. Usage: --mismatch=<integer_value>
  --otherbowtieoptions=OTHERBOWTIEOPTIONS
                      Bowtie has so many options that it is not worth
                      listing them here. Go to http://bowtie-
                      bio.sourceforge.net/manual.shtml#command-line for the
                      manual and all available options. Usage:
                      --otherbowtieoptions='all options inputed as a string
                      (note the quotes!)'

HTSeq options:
  --stranded=STRANDED
                      Stranded options: yes, no, reverse. Default is
                      --stranded=reverse. Usage: --stranded=yes/no/reverse
  --order=ORDER       Order options: name, pos. Usage: --order=name/pos.
  --minqual=MINQUAL   Skip all reads with quality lower than the given
                      value. Default is --minqual=10. Usage:
                      --minqual=<value>
  --type=TYPE         The feature type (3rd column in GTF file) to be used.
                      Default is --type=exon (suitable for RNA-seq analysis)
  --idattr=IDATTR     Feature ID from the GTF file to identify counts in the
                      output table Default is --idattr=gene_id. Usage:
                      --idattr=<id attribute>
  --mode=MODE         Mode to handle reads overlapping more than one
                      feature. Default is --mode=union. Usage: --mode=union
                      /intersection-strict/intersection-nonempty
```

1.1.4 Cloud computing with SPARTA on Amazon EC2

The ability to perform large scale data analysis may require computational capacity not found on a personal computing environment. Thus, SPARTA is capable of running in the cloud or on high performance computing environments. In the subsequent tutorial, we describe the analysis process of computing differentially expressed genes using SPARTA and the provided ExampleData in the cloud with Amazon EC2.

Contents:

Create an Amazon Web Services Account

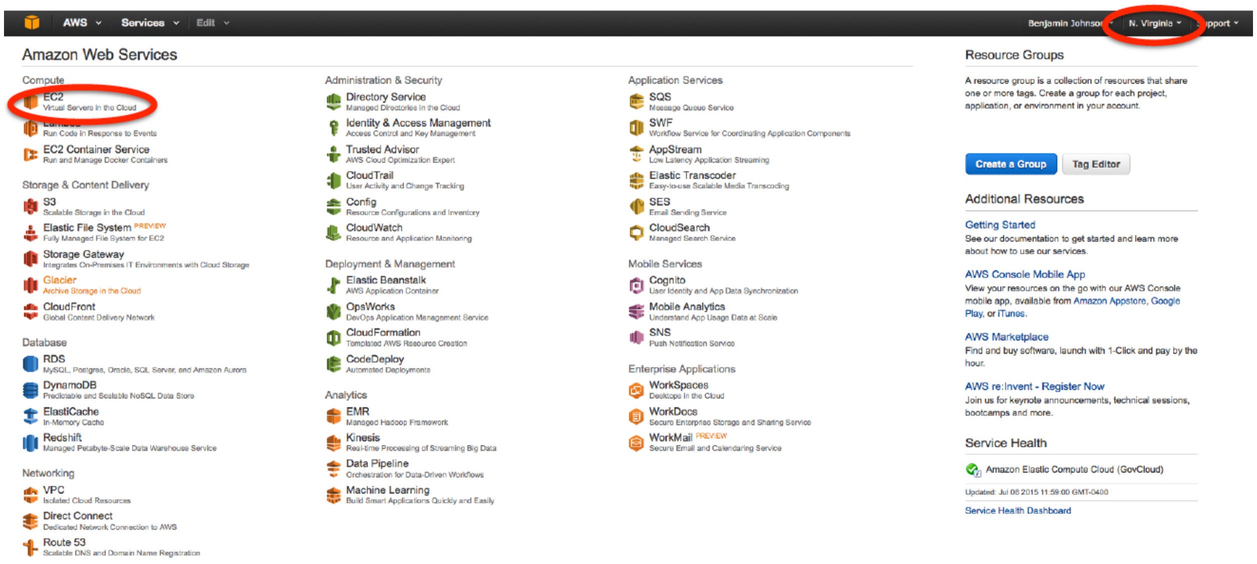
First we need to create an Amazon Web Services (AWS) account. To do this:

1. Navigate to the AWS site
2. In the upper right corner, click on the “Sign In to the Console” button
3. Enter your e-mail and make sure the “I am a new user” is selected

4. Click on the "Sign in using our secure server" button to get started creating your account
5. Enter your information and password into the relevant fields and click "Create account"
6. Continue to input your necessary information as denoted by the fields with an asterisk (*)
7. Read and click on the "I agree to the AWS terms" so that it is checked
8. Click on the "Create account and continue" button
9. From here you will need to enter your credit card information so that if you decide to go beyond the "Free tier" machines, they can charge you (computing time, even on large machines is pretty cheap). Also, you will need to verify your information and select a support plan that suites you.

Now that you have created your account, we can log into the AWS console if you aren't already. To do this:

1. Navigate to the AWS site.
2. Click on the "My Account" in the upper right and select "AWS Management Console" from the menu options
3. Log in with your user name and password
4. Set your location to "N. Virginia" if you are in the midwest or another relevant location (upper right red circle) and click on EC2 (red circle on the left side of the page)



Amazon Web Services

Compute
EC2
Virtual Servers in the Cloud
Lambda
Run Code in Response to Events
EC2 Container Service
Run and Manage Docker Containers

Storage & Content Delivery
S3
Scalable Storage in the Cloud
Elastic File System PREVIEW
Fully Managed File System for EC2
Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage
Glacier
Archive Storage in the Cloud
CloudFront
Global Content Delivery Network

Database
RDS
MySQL, Postgres, Oracle, SQL Server, and Amazon Aurora
DynamoDB
Predictable and Scalable NoSQL Data Store
ElastiCache
In-Memory Cache
Redshift
Managed Petabyte-Scale Data Warehouse Service

Networking
VPC
Isolated Cloud Resources
Direct Connect
Dedicated Network Connection to AWS
Route 53
Scalable DNS and Domain Name Registration

Administration & Security
Directory Service
Managed Directories in the Cloud
Identity & Access Management
Access Control and Key Management
Trusted Advisor
AWS Cloud Optimization Expert
CloudTrail
User Activity and Change Tracking
Config
Resource Configurations and Inventory
CloudWatch
Resource and Application Monitoring

Deployment & Management
Elastic Beanstalk
AWS Application Container
OpsWorks
DevOps Application Management Service
CloudFormation
Templated AWS Resource Creation
CodeDeploy
Automated Deployments

Analytics
EMR
Managed Hadoop Framework
Kinesis
Real-time Processing of Streaming Big Data
Data Pipeline
Orchestration for Data-Driven Workflows
Machine Learning
Build Smart Applications Quickly and Easily

Application Services
SQS
Message Queue Service
SWF
Workflow Service for Coordinating Application Components
AppStream
Low Latency Application Streaming
Elastic Transcoder
Easy-to-use Scalable Media Transcoding
SES
Email Sending Service
CloudSearch
Managed Search Service

Mobile Services
Cognito
User Identity and App Data Synchronization
Mobile Analytics
Understand App Usage Data at Scale
SNS
Push Notification Service

Enterprise Applications
WorkSpaces
Desktops in the Cloud
WorkDocs
Secure Enterprise Storage and Sharing Service
WorkMail PREVIEW
Secure Email and Calendaring Service

Resource Groups

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

Additional Resources

Getting Started
See our documentation to get started and learn more about how to use our services.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent - Register Now
Join us for keynote announcements, technical sessions, bootcamps and more.

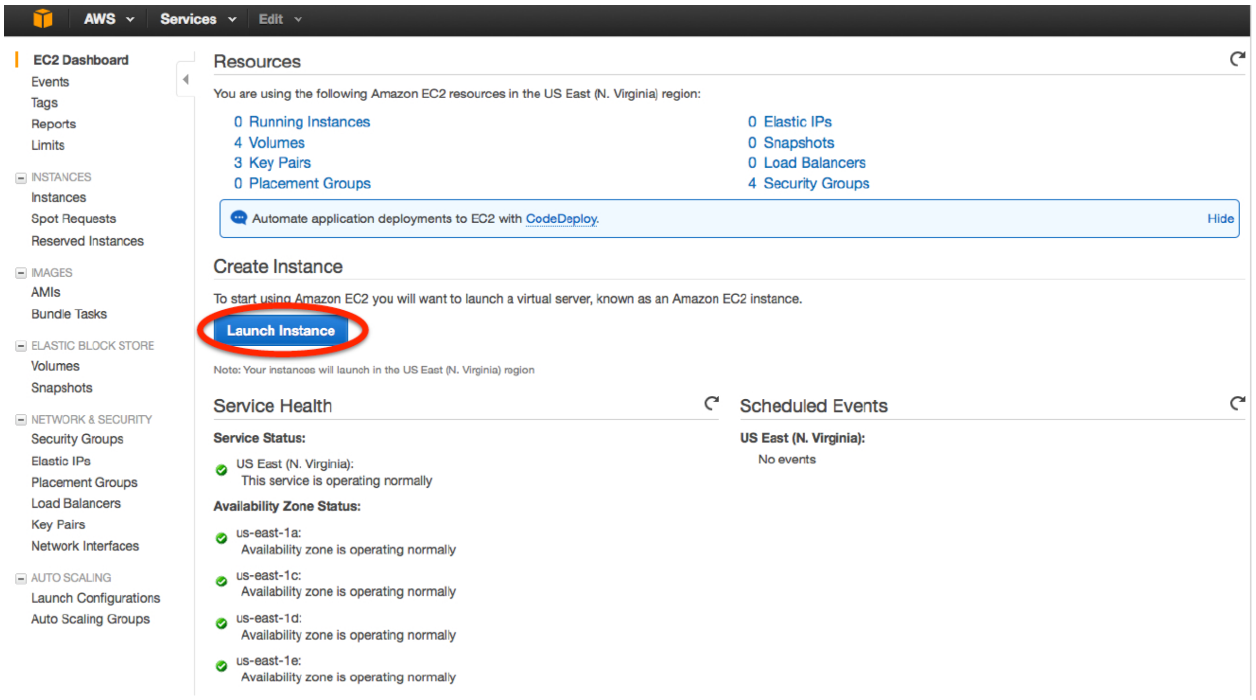
Service Health

Amazon Elastic Compute Cloud (GovCloud)

Updated: Jul 08 2015 11:59:00 GMT-0400

Service Health Dashboard

5. Select "Launch Instance"



Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

0 Running Instances	0 Elastic IPs
4 Volumes	0 Snapshots
3 Key Pairs	0 Load Balancers
0 Placement Groups	4 Security Groups

Automate application deployments to EC2 with CodeDeploy.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (N. Virginia) region

Service Health

Service Status:

US East (N. Virginia):
This service is operating normally

Availability Zone Status:

us-east-1a:
Availability zone is operating normally

us-east-1c:
Availability zone is operating normally

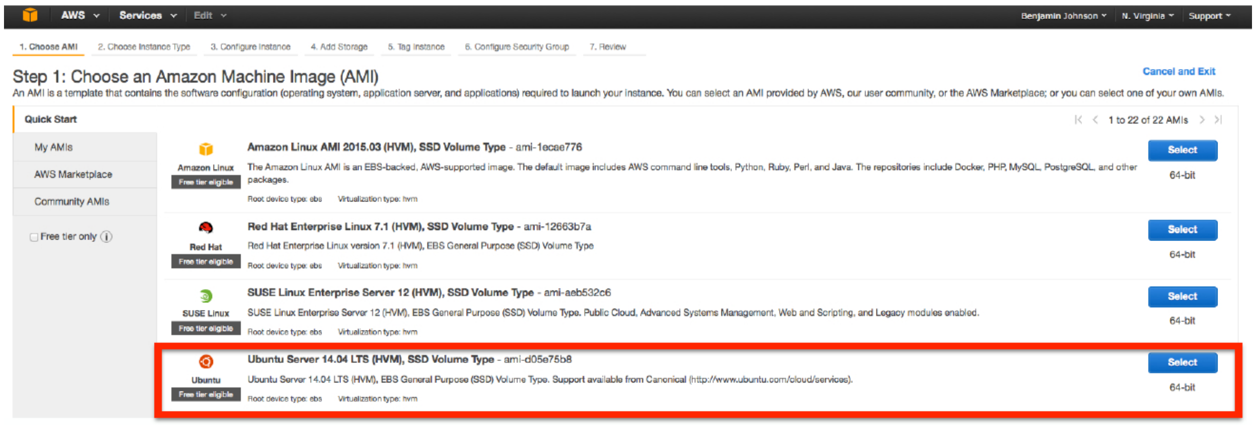
us-east-1d:
Availability zone is operating normally

us-east-1e:
Availability zone is operating normally

Scheduled Events

US East (N. Virginia):
No events

6. Select the "Ubuntu Server 14.04 LTS (HVM), SSD Volume Type" machine image by clicking on "Select"



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

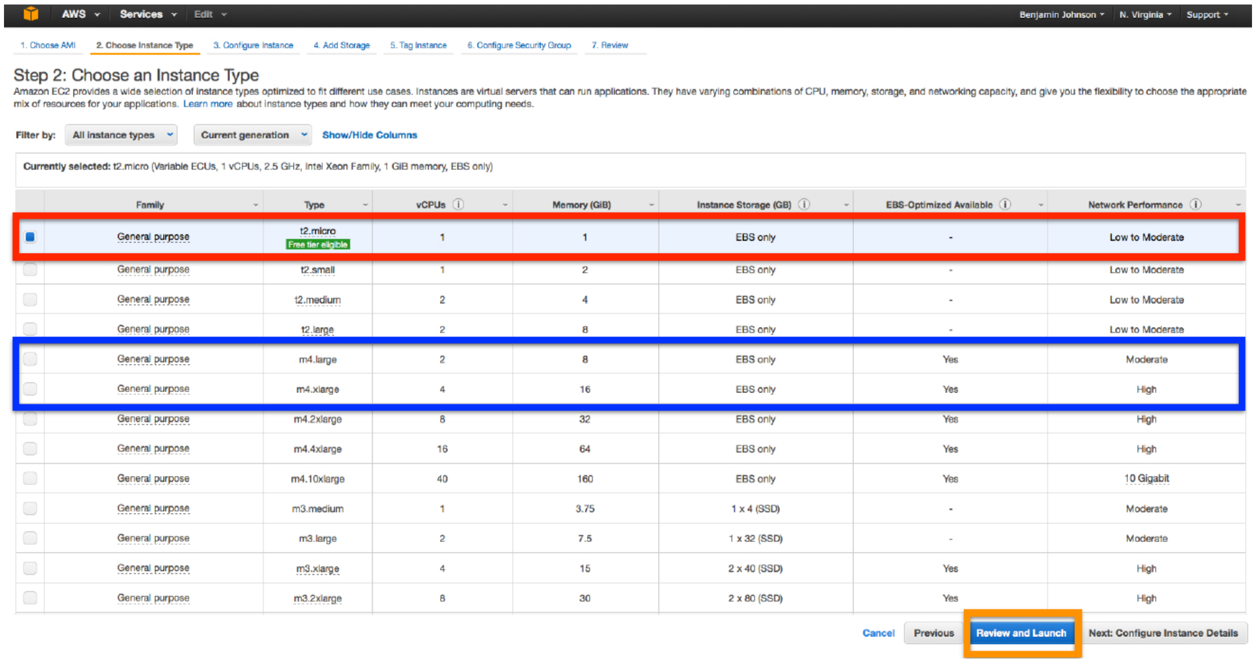
Amazon Linux AMI 2015.03 (HVM), SSD Volume Type - ami-1ecae776
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm

Red Hat Enterprise Linux 7.1 (HVM), SSD Volume Type - ami-12663b7a
Red Hat Enterprise Linux version 7.1 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm

SUSE Linux Enterprise Server 12 (HVM), SSD Volume Type - ami-aeb532c6
SUSE Linux Enterprise Server 12 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8
Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm

7. For working with the ExampleData we do not need significant hardware capacity, so for now, select the t2.micro instance type (red box). However, if you would like to analyze your own data, either the m4.large or m4.xlarge instance types are reasonable (blue box; these instances will charge you per hour, though are quite cheap). Then, click on "Review and Launch" (orange box).



Step 2: Choose an Instance Type

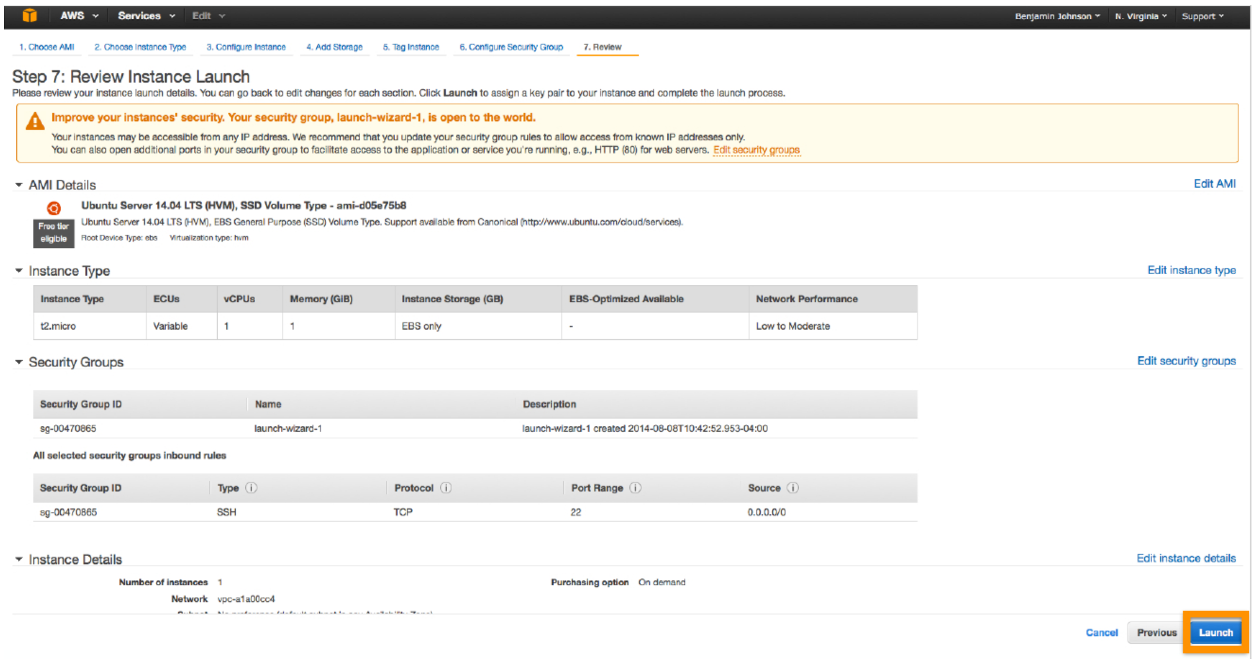
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
●	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
	General purpose	m4.large	2	8	EBS only	Yes	Moderate
	General purpose	m4.xlarge	4	16	EBS only	Yes	High
	General purpose	m4.2xlarge	8	32	EBS only	Yes	High
	General purpose	m4.4xlarge	16	64	EBS only	Yes	High
	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit
	General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
	General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
	General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
	General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High

8. Ignore the warning and click "Launch" (orange box)



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, launch-wizard-1, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

AMI Details

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8
Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root Device Type: ebs Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups

Security Group ID	Name	Description
sg-00470865	launch-wizard-1	launch-wizard-1 created 2014-08-08T10:42:52.953-04:00

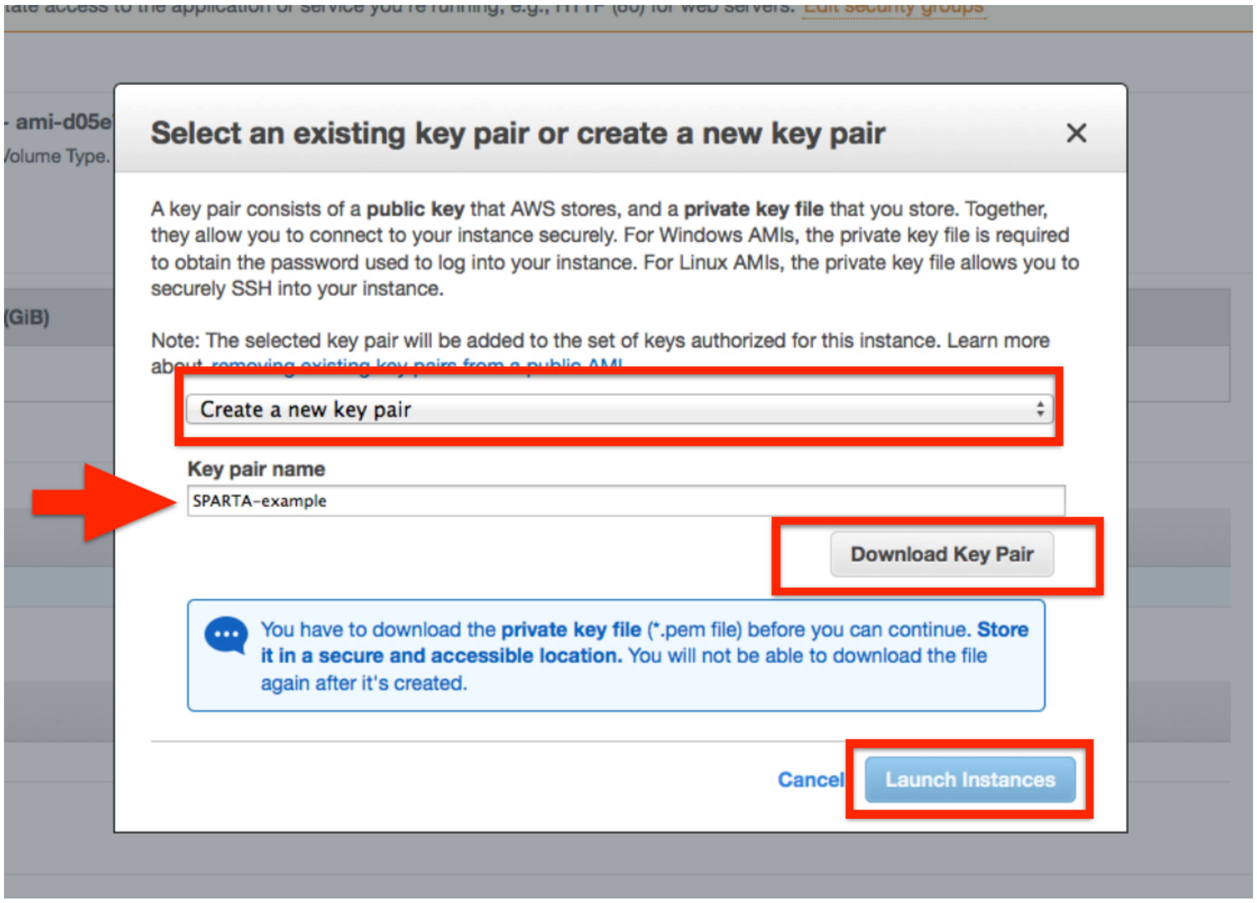
All selected security groups inbound rules

Security Group ID	Type	Protocol	Port Range	Source
sg-00470865	SSH	TCP	22	0.0.0.0/0

Instance Details

Number of instances	1
Network	vpc-a1a00cc4
Purchasing option	On demand

9. Create a new key pair and name it "SPARTA-example" (no quotes; red arrow). Then click on "Download Key Pair". Save this .pem file. After you download and save your .pem file, click on the "Launch Instances" button.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

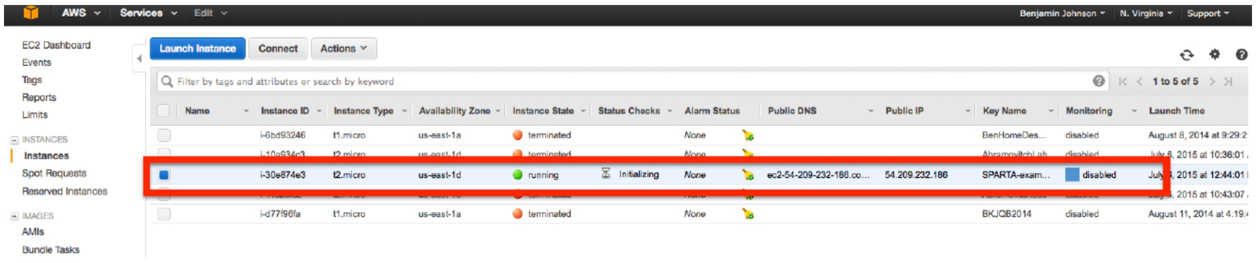
Key pair name: SPARTA-example

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

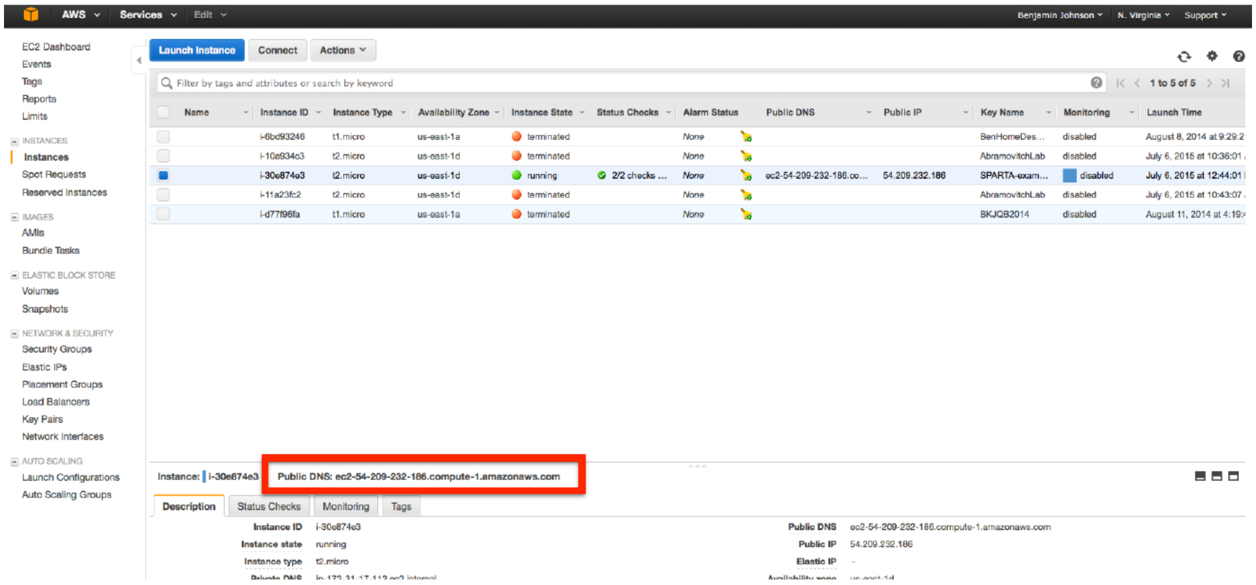
Cancel | Launch Instances

10. Select the "View Instances" button and wait until your "Instance State" turns green.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time
	i-6bd93246	t1.micro	us-east-1a	terminated		None			BenHomeDes...	disabled	August 8, 2014 at 9:29:2
	i-10a934c3	t2.micro	us-east-1d	terminated		None			AbramovitchLab...	disabled	July 6, 2015 at 10:36:01
	i-30e874e3	t2.micro	us-east-1d	running	Initializing	None	ec2-54-209-232-186.co...	54.209.232.186	SPARTA-exam...	disabled	July 6, 2015 at 12:44:01
	[illegible]	[illegible]	[illegible]	[illegible]		[illegible]			[illegible]	[illegible]	[illegible], 2015 at 10:43:07
	i-d77f96fa	t1.micro	us-east-1a	terminated		None			BKJQB2014	disabled	August 11, 2014 at 4:19:

11. Copy and paste the "Public DNS" into a text document (e.g. TextEdit or Notepad) and save it. This is your Amazon EC2 machine and you will need this to log into it.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time
	i-6bd93246	t1.micro	us-east-1a	terminated		None			BenHomeDes...	disabled	August 8, 2014 at 9:29:2
	i-10a934c3	t2.micro	us-east-1d	terminated		None			AbramovitchLab	disabled	July 6, 2015 at 10:36:01
	i-30e874e3	t2.micro	us-east-1d	running	2/2 checks ...	None	ec2-54-209-232-186.co...	54.209.232.186	SPARTA-exam...	disabled	July 6, 2015 at 12:44:01
	i-11a23fc2	t2.micro	us-east-1d	terminated		None			AbramovitchLab	disabled	July 6, 2015 at 10:43:07
	i-d77f96fa	t1.micro	us-east-1a	terminated		None			BKJQB2014	disabled	August 11, 2014 at 4:19:

Instance: i-30e874e3 Public DNS: ec2-54-209-232-186.compute-1.amazonaws.com

Instance ID	i-30e874e3	Public DNS	ec2-54-209-232-186.compute-1.amazonaws.com
Instance state	running	Public IP	54.209.232.186
Instance type	t2.micro	Elastic IP	-

To log into the machine, follow either the *Mac/Linux version* or the *Windows version*

Mac/Linux Login Procedure

To log into the machine you just created, we need to use the .pem file and the Terminal. If you don't remember how to get to the terminal, see the image in the Introduction in either the Mac OS X tutorial or Linux tutorial.

Move the .pem file from your Downloads folder to your Desktop for the time being.

Start up your terminal and type:

```
cd ~/Desktop
```

This will navigate to your Desktop. We will change the permissions to read only for you, the user:

```
chmod 400 SPARTA-example.pem
```

Now, let's log into our machine!

To do this we will type something like this (NOTE THE DNS ADDRESS AFTER THE 'ubuntu@' IS NOT REAL. THIS IS WHERE YOU SHOULD PUT YOUR PUBLIC DNS FROM EARLIER):

```
ssh -i ~/Desktop/SPARTA-example.pem ubuntu@ec2-your-public-dns-goes-here.compute-1.amazonaws.com
```

What you are doing is logging in using the secure shell (ssh) command with your credentials in the .pem file as the user 'ubuntu' to the machine 'ec2-...-compute-1.amazonaws.com'.

You should now see something like:

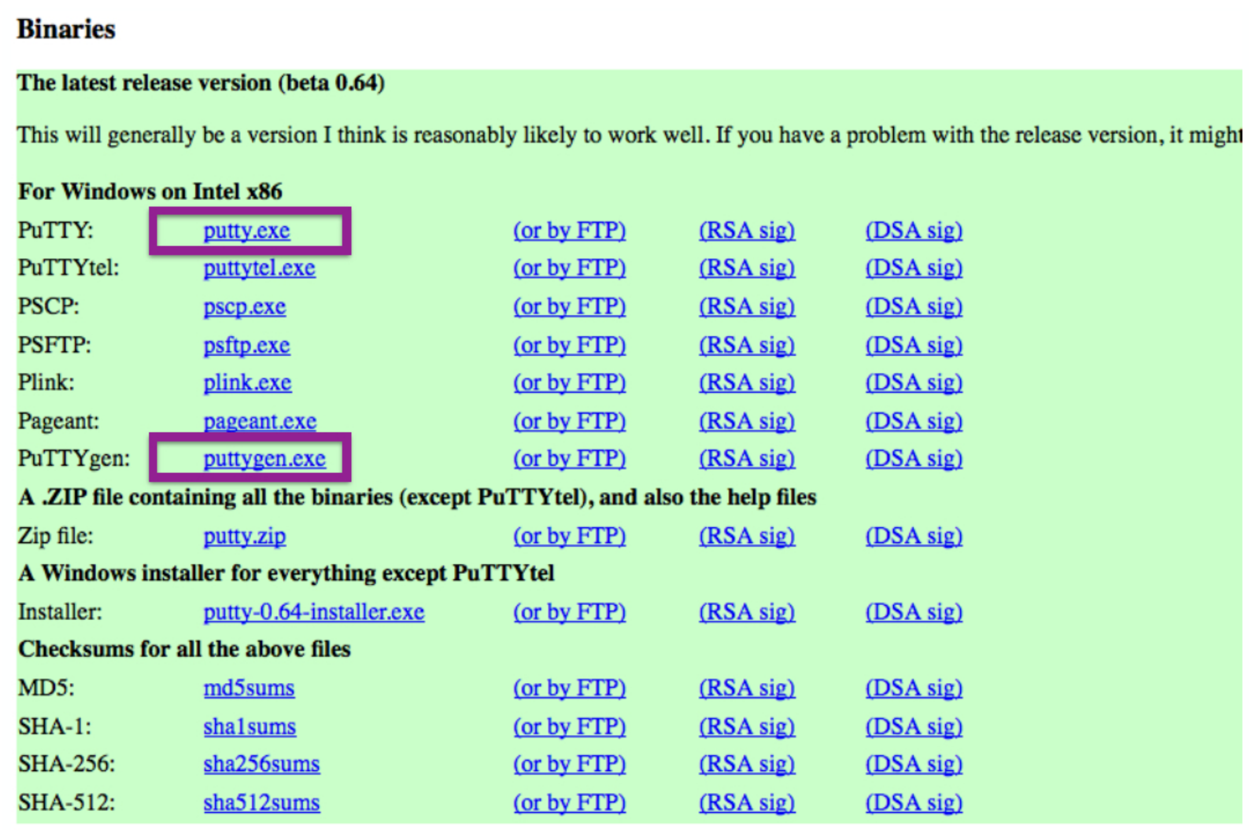
```
ubuntu@ip-345-67-89-10:
```

Congratulations! You're now on the cloud computer that you launched earlier!

Windows Login Procedure

To log into the machine you just created, we need to use the .pem file, a key generator called PuTTYgen, and a secure shell (ssh) client called PuTTY.

Download PuTTY and PuTTYgen from here.



Binaries

The latest release version (beta 0.64)

This will generally be a version I think is reasonably likely to work well. If you have a problem with the release version, it might

For Windows on Intel x86

PuTTY:	putty.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYtel:	puttytel.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSCP:	pscp.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSFTP:	psftp.exe	(or by FTP)	(RSA sig)	(DSA sig)
Plink:	plink.exe	(or by FTP)	(RSA sig)	(DSA sig)
Pageant:	pageant.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYgen:	puttygen.exe	(or by FTP)	(RSA sig)	(DSA sig)

A .ZIP file containing all the binaries (except PuTTYtel), and also the help files

Zip file:	putty.zip	(or by FTP)	(RSA sig)	(DSA sig)

A Windows installer for everything except PuTTYtel

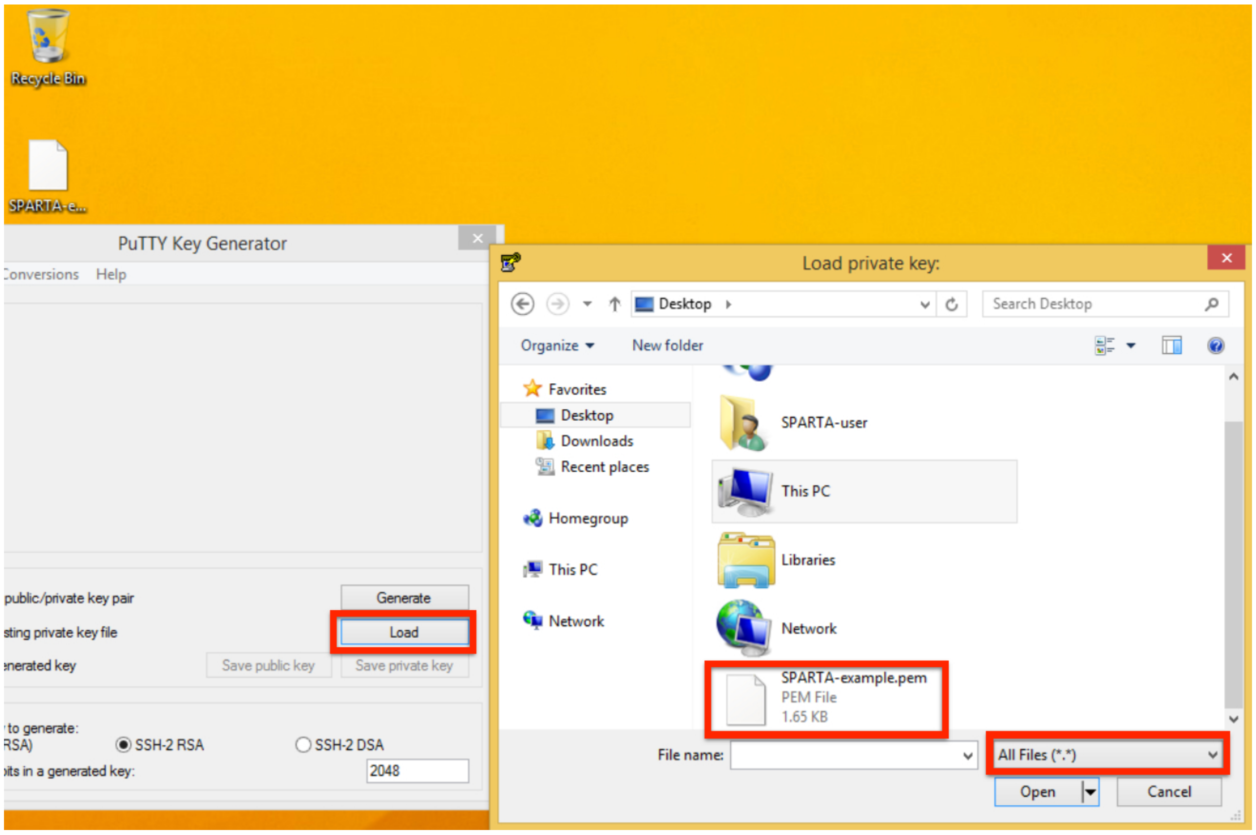
Installer:	putty-0.64-installer.exe	(or by FTP)	(RSA sig)	(DSA sig)

Checksums for all the above files

MD5:	md5sums	(or by FTP)	(RSA sig)	(DSA sig)
SHA-1:	sha1sums	(or by FTP)	(RSA sig)	(DSA sig)
SHA-256:	sha256sums	(or by FTP)	(RSA sig)	(DSA sig)
SHA-512:	sha512sums	(or by FTP)	(RSA sig)	(DSA sig)

Move the .pem file from your Downloads folder to your Desktop for the time being.

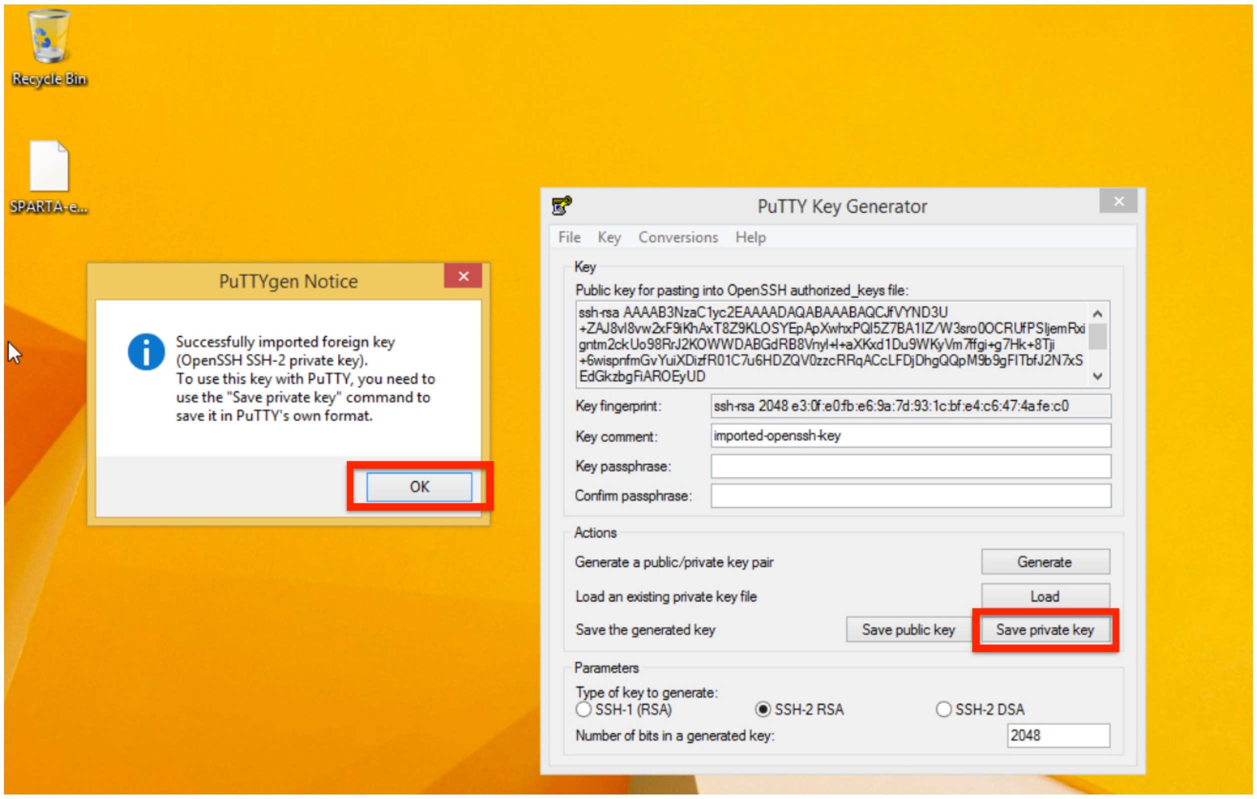
Open up PuTTYgen and click on “Load”. Navigate to your Desktop and select the SPARTA-example.pem, click “Open”.



PuTTY should present you with a window that says something like "Successfully imported private key..." It also states in the window that we need to use the "Save private key" command. So, let's do just that.

Click "OK"

Click on "Save private key". Save it somewhere you know where it is (reasonably easy to find is always a good idea) It may ask you if you want to save it without a passphrase. Click "Yes". Save it as "SPARTA-example" on the Desktop (no quotes).



Successfully imported foreign key (OpenSSH SSH-2 private key).
To use this key with PuTTY, you need to use the "Save private key" command to save it in PuTTY's own format.

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCJfVYND3U
+ZAJ8vl8vw2xF9iKhAxT8Z9KLOSYEpApXwhxPQI5Z7BA1IZ/W3sro0OCRUfPSljemRxi
gntm2ckUo98RrJ2KOWWDABGdRB8Vnyl+l+aXKxd1Du9WKyVm7ffgi+g7Hk+8Tji
+6wispnfmGvYuiXDizfR01C7u6HDZQV0zzcRRqACcLFDjDhgQQpM9b9gFITbfJ2N7xS
EdGkzbgFiAROEyUD
```

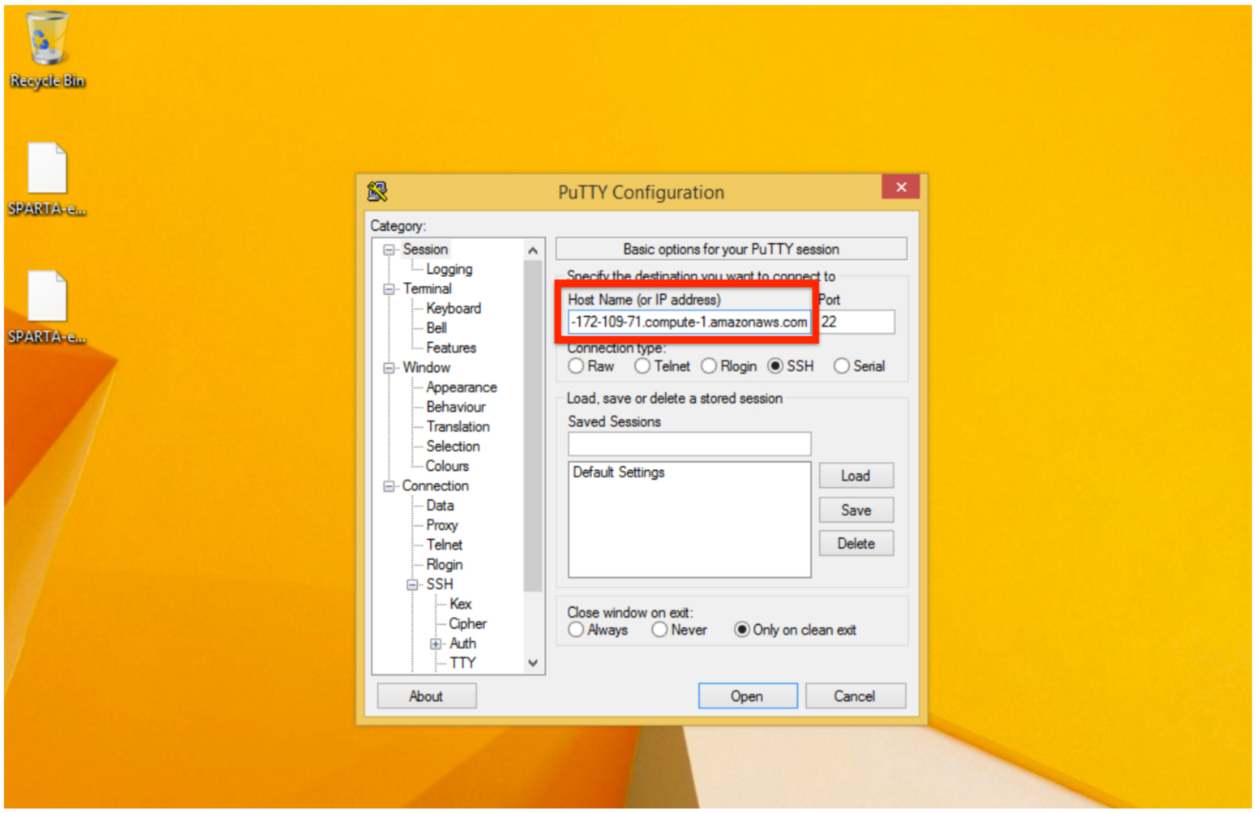
Key fingerprint: ssh-rsa 2048 e3:0f:e0:fb:e6:9a:7d:93:1c:bf:e4:c6:47:4a:fe:c0
Key comment: imported-openssh-key
Key passphrase:
Confirm passphrase:

Generate a public/private key pair
Load an existing private key file
Save the generated key

Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA
Number of bits in a generated key: 2048

Now, let's log into our machine!

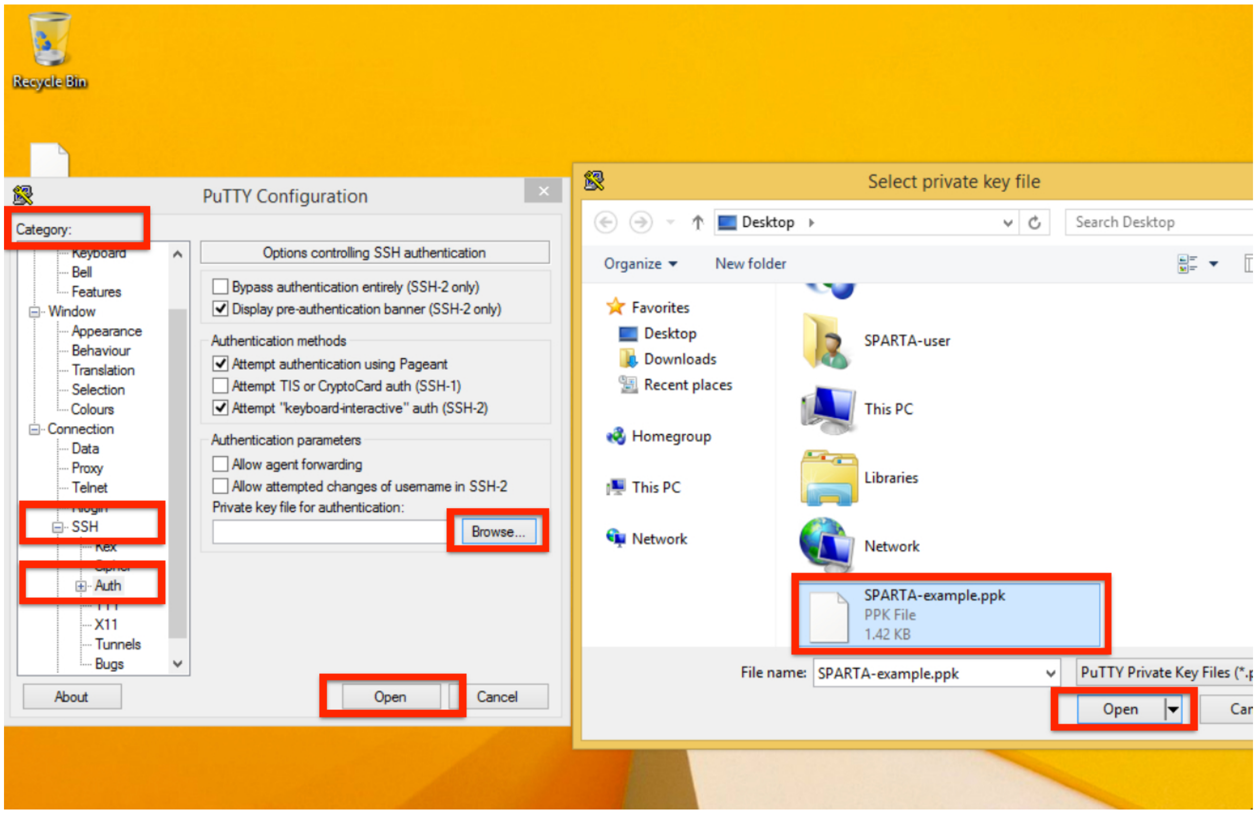
To do this, we will need to open up PuTTY (not PuTTYgen, which is what we were just using). Enter the Host Name (public DNS from earlier) into the "Host Name" box.



Host Name (or IP address): -172-109-71.compute-1.amazonaws.com
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

Look in the Category section (left-hand side of the window) and navigate to the SSH section (about halfway down the list).

Click on “Auth” in the SSH category and add your PuTTYgen key (ppk) you just made by clicking on “Browse” and selecting the ppk file. Now click “Open”.



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

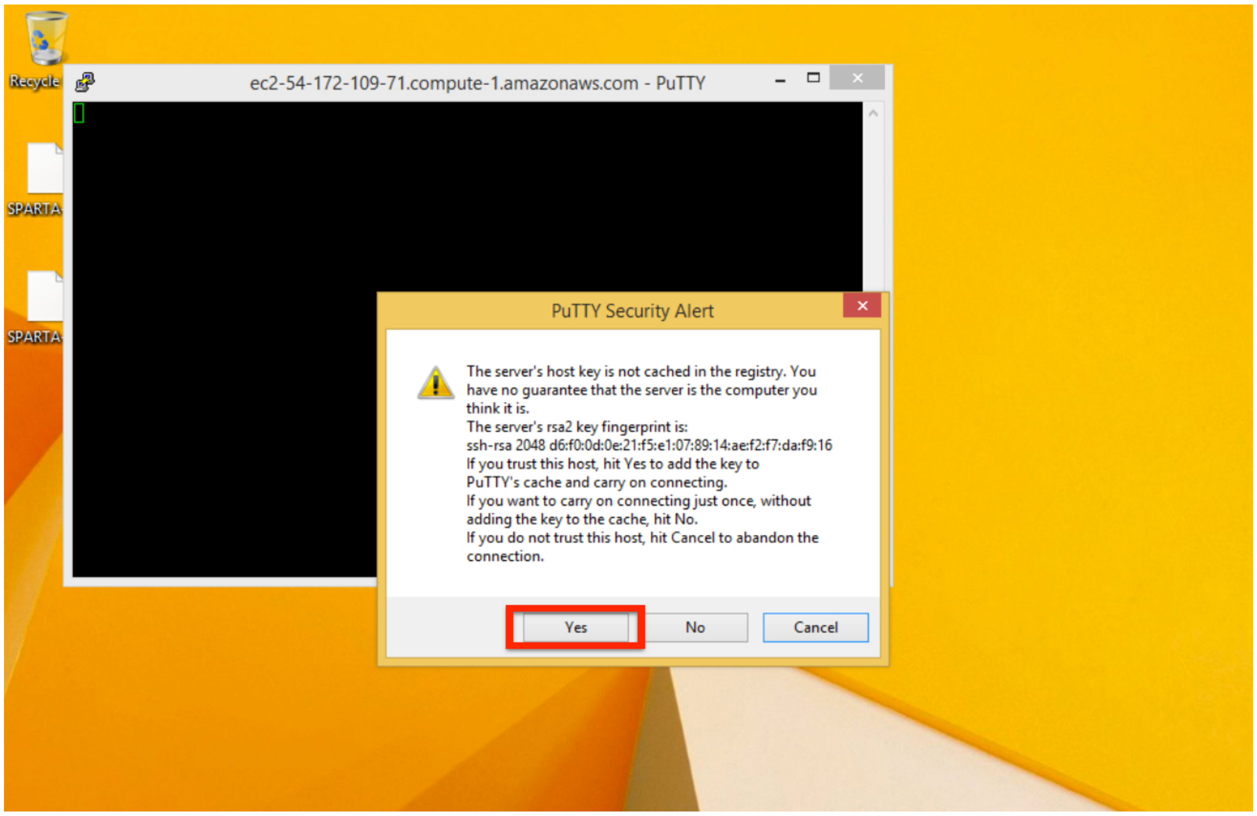
Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

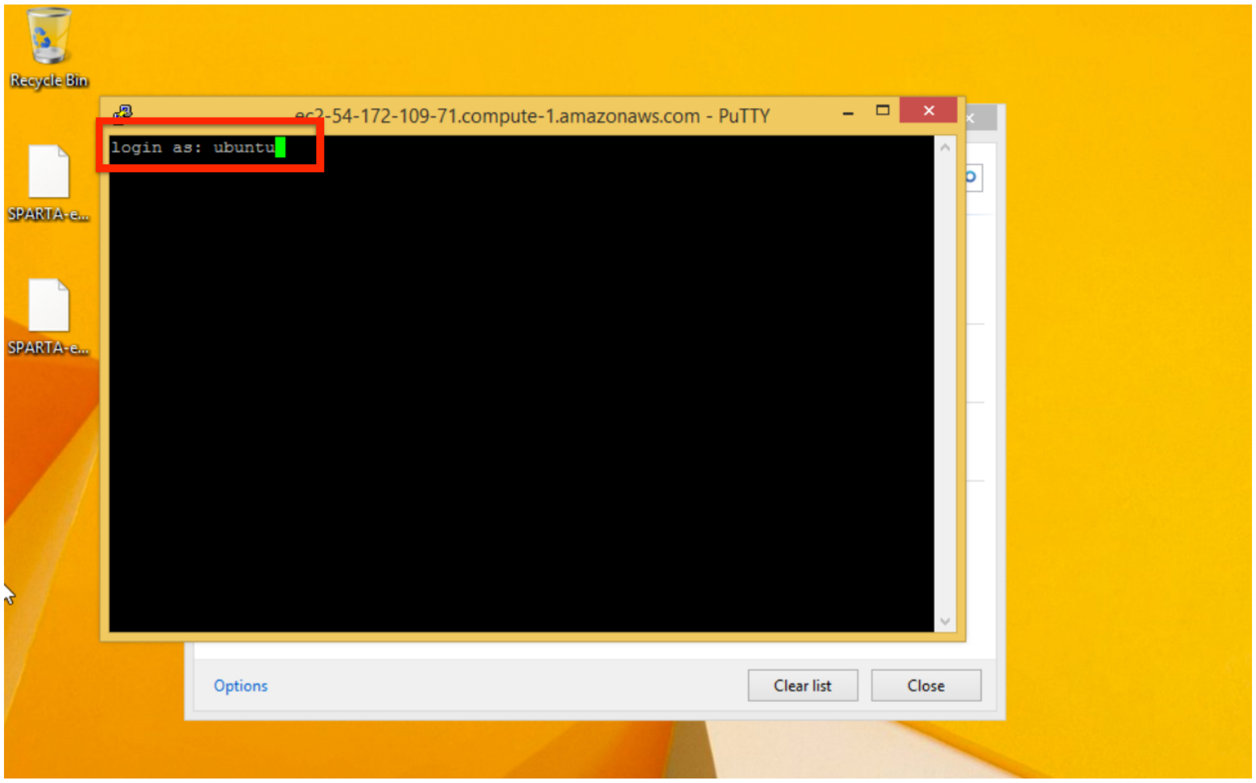
File name: SPARTA-example.ppk

The first time you log into a new machine, it may prompt you with a window similar to the one below. This is verifying the RSA fingerprint of the machine. Click "Yes".



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 d6:f0:0d:0e:21:f5:e1:07:89:14:ae:f2:f7:da:f9:16
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Login as "ubuntu"



```
login as: ubuntu
```

You should now see something like:

```
ubuntu@ip-345-67-89-10:
```

Congratulations! You're now on the cloud computer that you launched earlier!

Analyzing the RNA-seq ExampleData with SPARTA

Now that we are logged into our Amazon EC2 machine, let's start analyzing the ExampleData that comes with SPARTA.

To get SPARTA onto our EC2 machine, we need a tool called git. To download and install this tool, type:

```
sudo apt-get install git
```

This will begin downloading the necessary files. It will likely prompt you with a yes/no (Y/n) question about proceeding with the install. Type:

```
Y
```

Before we download SPARTA, let's make and navigate into a folder to put everything in and let's call it Desktop. Type:

```
mkdir Desktop
cd Desktop
```

Now, we can download SPARTA_Linux from the GitHub repository. Type:

```
git clone https://github.com/biobenkj/SPARTA_Linux
```

This may take a minute or two to download the necessary files. Once they are downloaded, we can move the ExampleData folder out of the SPARTA_Linux folder and into Desktop. To do this, type:

```
cd SPARTA_Linux
mv ExampleData ..
```

To install the dependencies, type:

```
bash install_dependencies.sh
```

This will collect and install all of the dependencies necessary to run SPARTA (it will take a couple minutes). It will likely prompt you with a yes/no (Y/n) question about proceeding. Type:

```
Y
```

Now, we need to edit the ConfigFile.txt within SPARTA_Linux to run the workflow in non-interactive mode.

The ConfigFile.txt should be displayed before you. You cannot use your mouse to move the cursor around, but you can use the arrow keys. Navigate down to the bottom where the experimental conditions input is. To compare all four of the experimental conditions in the ExampleData, we need to add two more lines below "Experimental_condition_2_files:".

Before any of the files are entered, it should look like this:

```
Reference_condition_files:
Experimental_condition_2_files:
Experimental_condition_3_files:
Experimental_condition_4_files:
```

This is for *4* conditions.

Now, we need to add in the file names. At this point in the analysis, the file names will have a prefix called "map" and an extension called ".sam". So, based on the names of our input data, we can type in the file names with the appropriate prefix and extension.

So if our input data looks like this:

```
gly7a.fq.gz
gly7b.fq.gz
gly5a.fq.gz
gly5b.fq.gz
pyr7a.fq.gz
pyr7b.fq.gz
pyr5a.fq.gz
pyr5b.fq.gz
```

Our files at this point in the analysis will look like this:

```
mapgly7a.sam
mapgly7b.sam
mapgly5a.sam
mapgly5b.sam
mappyr7a.sam
mappyr7b.sam
mappyr5a.sam
mappyr5b.sam
```

Thus, once we have added these files to the appropriate experimental conditions, it will look like this:

```
Reference_condition_files: mapgly7a.sam, mapgly7b.sam
Experimental_condition_2_files: mapgly5a.sam, mapgly5b.sam
Experimental_condition_3_files: mappyr7a.sam, mappyr7b.sam
Experimental_condition_4_files: mappyr5a.sam, mappyr5b.sam
```

To save the file, hit the Control key and "O" (not the number zero). Hit enter/return. To exit the editor, hit the Control key and "X".

Now we can run the analysis non-interactively! Type:

```
python SPARTA.py --noninteractive
```

From here, the analysis will proceed from QC, aligning, counting, and differential gene expression.

Congratulations! You've analyzed the ExampleData in the cloud!

Transferring files to and from Amazon EC2 computers

Mac/Linux users:

You can use a command line tool called "scp"

An example usage to transfer the file "YourFile.txt" to your home (~) directory on an Amazon EC2 computer from your Desktop:

```
scp -i ~/Desktop/SPARTA-example.pem ~/Desktop/YourFile.txt ubuntu@ec2-...-.compute-1.amazonaws.com:~
```

An example usage to transfer the file "YourFile.txt" from your home (~) directory on an Amazon EC2 computer to your Desktop:

```
scp -i ~/Desktop/SPARTA-example.pem ubuntu@ec2-...-.compute-1.amazonaws.com:~ ~/Desktop/YourFile.txt
```

If you would like to transfer an entire folder/directory, add the "-r" option. Thus, to transfer YourFolder from your Desktop to the home (~) directory on an Amazon EC2 computer:

```
scp -i ~/Desktop/SPARTA-example.pem -r ~/Desktop/YourFolder ubuntu@ec2-...-.compute-1.amazonaws.com:
```

Windows users:

You can use a client called WinSCP. Click on the "Installation package" under "Download WinSCP" to initiate the download.

Follow the installer and just use the default settings.

Once the client is open:

- Host name - Your Public DNS to your EC2 machine
- User name - ubuntu
- Advanced -> SSH -> Authentication -> Private key file (click on the "..." button) -> select the PuTTYgen (.ppk) file generated earlier in the tutorial

Click "Login" to connect.

Now, you can transfer files, to and from your local machine and the EC2 machine!

1.1.5 Frequently Asked Questions

1. Does SPARTA support paired-end reads?

Not yet. Currently, SPARTA only supports single-end reads as we have found it is the most common/inexpensive approach for differential gene expression analysis. Paired-end read support will be incorporated in future releases of SPARTA. If you have paired-end reads and would like to use SPARTA, as a workaround, you can run just the forward reads.

2. What if I only have a GFF file and not a GTF file for my organism?

A GTF file is a more stringent version of a GFF file. Thus, your GFF file *may* work with HTSeq for counting transcript abundance. However, GFF file formating is more relaxed and thus, it may not work. As a potential workaround, you can open the GFF file in a plain text editor like TextEdit (Mac) or Notepad (Windows). Look at each line and see if the beginning of each line in the GFF file begins with the same phrase. In the example below the GTF line begins with *Chromosome* and the reference genome FASTA file begins with the same phrase *Chromosome*. Next, examine each line for a phrase that relates specifying a region for a gene. In the example below, HTSeq by default looks for the phrase **exon**. If your file **does not** have **exon** as the phrase, you can specify to SPARTA/HTSeq which phrase to look for through the option `--type=your_gene_region_name` where `your_gene_region_name` is the phrase specific to your file.

Note: The preferred location for downloading a reference genome file and GTF file is through Ensembl (http://bacteria.ensembl.org/info/website/ftp/index.html). This list is fairly comprehensive though not exhaustive (especially if there is no reference and you've had to assemble your own/annotate it).

GTF example:

Chromosome protein_coding **exon** 1 1524 . + . gene_id “MT0001”; transcript_id “AAK44224”; exon_number “1”; gene_name “dnaA”; transcript_name “dnaA/AAK44224”; seqedit “false”; Chromosome protein_coding CDS 1 1521 . + 0 gene_id “MT0001”; transcript_id “AAK44224”; exon_number “1”; gene_name “dnaA”; transcript_name “dnaA/AAK44224”; protein_id “AAK44224”; Chromosome protein_coding stop_codon 1522 1524 . + 0 gene_id “MT0001”; transcript_id “AAK44224”; exon_number “1”; gene_name “dnaA”; transcript_name “dnaA/AAK44224”;

Reference genome example (FASTA):

>*Chromosome* dna:chromosome chromosome:GCA_000008585.1:Chromosome:1:4403837:1 TTGACCGATGACCCCGGTTCAGGCTTCACCACAGTGTGGAACGCGGTCGTCTCCGAACTT AACGGCGACCCTAAGGTTGACGACGGACCCAGCAGTGATGCTAATCTCAGCGCTCCGCTG ACCCCTCAGCAAAGGGCTTGGCTCAATCTCGTCCAGCCATTGACCATCGTCGAGGGGTTT GCTCTGTTATCCGTGCCGAGCAGCTTTGTCCAAAACGAAATCGAGCGCCATCTGCGGGCC CCGATTACCGACGCTCTCAGCCGCCGACTCGGACATCAGATCCAACTCGGGGTCCGCATC...

3. I keep getting an error at the differential gene expression stage stating ‘‘Error: unexpected symbol in “name_of_your_file” Execution halted‘‘

This error will occur if you have file names that begin with a number instead of a letter. R (the language used to do the DE analysis) doesn't like having variable names that begin with a number instead of a letter. Thus, the remedy is to ensure all of your sample files begin with a letter instead of a number.

4. My sample files are split between multiple .fastq/.fq files. How can I put them into a single file?

If you have sequenced many samples across several lanes of an Illumina flowcell (as an example), you can concatenate all of them into one file per sample using the following commands (though you will need to alter the file names to fit your needs).

1. Make a copy of your files in a different folder so that if something goes wrong, you still have the raw data.
2. Open the terminal and navigate to the folder containing your copied sample files. As an example, if they are in a folder on the Desktop and you're on a Mac/Linux machine, you can type `cd ~/Desktop/your_folder_with_copied_sample_files`. This is changing directories/folders to the one containing your sample files on the Desktop.
3. To combine the files, ensure they are unzipped or decompressed to .fastq or .fq files (e.g. NOT .fastq.gz or .fq.gz or .fastq.zip or .fq.zip, etc).
4. Performing the concatenation can be accomplished as follows with an example for Mac/Linux machines.

```
cat samplefile1.fastq samplefile2.fastq samplefileN.fastq >>
new_combined_sample_file.fastq
```

1.1.6 License

1.1.7 Release notes

Version 1.0

1.1.8 Citation

Johnson BK, Scholz MB, Teal TK, Abramovitch RB: SPARTA: Simple Program for Automated reference-based bacterial RNA-seq Transcriptome Analysis. *BMC Bioinformatics* 2016, 17(1):1-4.

1.1.9 Acknowledgements

We would like to thank the members of the Abramovitch Lab for helpful discussions and critical assessment/bug identification within the workflow. We would also like to thank the developers and contributors of Python, Trimmomatic, FastQC, Bowtie, HTSeq, and edgeR; without these individuals, SPARTA would not be possible. Finally, we would like to thank you, the user, for utilizing the workflow and making it better.

1.1.10 Functionality wishlist

1. Add paired-end support for SPARTA
2. Add more modular approach to implementing different tools (perhaps through option specification?)
3. Include the ability to deal with batch effects in an efficient manner, requiring minimal user input
4. Support for other sequencing platforms as input (through adding support for SAM, BAM, FASTA, etc.)
5. Operon analysis
6. Definition of UTRs
7. Output read mapping files with normalized expression values
8. Non-reference based analysis

- **Contribute:** If you would like to contribute to the project, the source code for each platform can be found in the GitHub repository.
- **Bugs: If you found a bug, please have a look at the issues page and add a description (please be explicit and include error**
  - Mac OS X issues
  - Windows issues
  - Linux issues
- Frequently Asked Questions
- License
- Release notes
- Citation and Acknowledgements

- Functionality wishlist