This is a quick guide on how to use Usearch7 to go from fastq files all the way to a table of OTUs. More information can be found on the Usearch project page, http://www.drive5.com/usearch/manual/
Also refer to this page for download and installation instructions. I'm going to assume in this manual that you can call Usearch simply by typing ./usearch7, but this depends on the folder where you have installed it, how you have named it and the folder you're in.

This guide is based on Illumina MiSeq reads and Usearch 7.0.959. It will NOT work with older versions of Usearch. It hopefully works with later versions. Please let me know if it doesn't. My email is in the bottom of each page.

This guide assumes that you have overlapping reads, that is, that your forward and reverse reads overlap each other. If this is not the case, or if you're not sure, please refer to the Usearch_no-overlap.pdf manual.

PART I: FILTERING AND MERGING

STEP 1. Quality trimming

In this step, you cut off bases with low quality. MiSeq reads usually have good qualities. On the other hand, there are gonna be further quality filtering later, so there's no need to be too stringent.

The command:

```
./usearch7 -fastq_filter <infile> -fastq_truncqual <cutoff> -fastqout <outfile>
```

Example:

```
./usearch -fastq_filter reads_R1.fq -fastq_truncqual 20 -fastqout reads_R1.trim.fq
./usearch -fastq_filter reads_R2.fq -fastq_truncqual 20 -fastqout reads_R2.trim.fq
```

STEP 2: Synchronizing

Despite the fact that MiSeq reads usually have good quality, it can happen that in your files you had some bad reads, which were discarded on the previous step. If Usearch finds reads that are present in one file and not the other, it doesn't know how to proceed. Therefore, I've written a script that will throw away every read that isn't present in both the forward (R1) and the reverse (R2) files. You can download this script from www.biologic.se. This script will automatically choose the name for the output files by adding '.sync.fq' to the end.

The command:

```
perl resync --fwd=<forward_reads> –rev=<reverse_reads>
```

Example:

```
perl resync –fwd=reads_R1.trim.fq –rev=reads_R2.trim.fq
```

STEP 3: Merging

In this step we will merge the forward and reverse reads into a single amplicon. When the overlap is perfect, they will simply be combined; where there are differences, the base with the highest Phred-score will be chosen. A new probability score will also be calculated for the bases in the overlap region. You can set a maximum limit of how many bases can be different between the reads in the overlap region without discarding the read pair. Think about the size of your expected overlap and the percentage of errors you find acceptable. If you choose not to set this value, there will be no upper limit (every read will be kept, no matter the differences).

The command:

```
./usearch7 -fastq_mergepairs <forward_reads> -reverse <reverse_reads> -fastq_maxdiffs <maximum number of different bases> -fastqout <outfile>
```

Example

```
./usearch7 -fastq_mergepairs reads_R1.trim.sync.fq -reverse reads_R2.trim.sync.fq -fastq_maxdiffs 7 -fastqout reads.merge.fq
```

STEP 4: Quality filtering

When merging, new base qualities were calculated for the overlapping region. In this step, we discard low quality reads and reads that are too short. To set these parameters, consider the length of your amplicon, the expected error rate in the read and your downstream applications. We will also take this opportunity to convert our reads from fastq to fasta.

The comand

```
./usearch7 -fastq_filter <infile> -fastq_minlen <minimal length> -fastq_maxee <maximum number of errors> -fastaout <outfile>
```

Example:

```
./usearch7 -fastq_filter reads.merge.fq -fastq_minlen 150 -fastq_maxee 3 -fastaout reads.merge.fa
```

PART II: CLUSTERING

STEP 5: Sample concatenation

In most applications, you want to compare communities from different environments, conditions etc. For this, you have to have the same OTU defined for all samples. Therefore, at this point we concatenate all files.

The command

```
cat <all_fasta_files> > <outfile>
```

Example

```
cat reads1.fa reads2.fa reads3.fa > all.fa
```

STEP 6: Dereplication

Here we combine all reads that are identical into a single one, keeping track of how many copies there are of each one. This saves a lot of time down the road.

The command:
```
./usearch7 -derep_fulllength <infile> -output <outfile> -sizeout
```

Example
```
./usearch7 -derep_fulllength all.fa -output uniques.fa -sizeout
```

STEP 7: Sorting

Usearch reads the fasta file in order, assigning each sequence to a cluster as it finds it. Since a sequence that has been read several times is more likely to be right, starting by the most frequent sequences and working our way down increases the chances the OTU found are real. We will therefore sort sequences from most frequent to least. At this point you can also throw away sequences that haven't been found a minimal amount of times. Discarding singletons will increase your precision and speed up the process, with a small lost of sensitivity. If you choose not to throw away any sequences, don't write the parameter -minsize.

The command:
```
./usearch7 -sortbysize <infile> -output <outfile> -minsize <minimal cluster size>
```

Example:
```
./usearch7 -sortbysize uniques.fa -output uniques.sort.fa -minsize 2
```

STEP 8: Clustering

Usearch uses single-linkage clustering, that is, it checks whether each sequence is at no more than a certain distance from the centroid of its cluster. If you prefer to use complete-linkage, this can be approximated by specifying half the radius you want to have. For example, if you want 97% OTU with single-linkage, your radius is 3%; with complete-linkage, your radius is 1.5%.

The command:
```
./usearch7 -cluster_otus <infile> -otus <outfile> -otu_radius_pct <radius>
```

Example:
```
./usearch7 -cluster_otus uniques.sort.fa -otus otus97.fa -otu_radius_pct 3.0
```

STEP 9: Renaming OTU

Our OTU so far have the name of the read ID of their centroid, which is simply not pleasant. Therefore, we can change their names now to OTU_1, OTU_2 etc. This script can be downloaded from http://drive5.com/python/. You can choose any name for your OTUs, but please use OTU_ if you want to keep following this tutorial.

The command:

```
python fasta_number.py <infile> <prefix> > <outfile>
```

Example:

```
python fasta_number.py otus97.fa OTU_ > otus97num.fa
```

STEP 10: Assigning reads to OTU

We will now look at each of our original fasta files and assign them to OTU. At this point, take the opportunity to make a directory just for your new cluster files. This is important downstream. You're also requested to say how similar your sample must be to the centroid. This must be compatible with the radius you used for clustering. For example, if you used a radius of 3%, use now a similarity of 0.97.

In this step you may see that most reads are identified as chimera and just a small part are being recruited to OTU. That's a bug in the screen output that won't affect your data.

The command:

```
./usearch7 -usearch_global <sample file> -db <numbered out file> -strand
        <plus/minus/both> -id <similarity to the centroid> -uc <outfile>
```

Example:

```
./usearch7 -usearch_global reads1.merge.fa -db otus97.num.fa -strand plus -id 0.97 -uc
        clusters/reads1.uc
```

STEP 11: Classifying OTU

If you're working with 16S, I recommend using the online RDP classifier: http://rdp.cme.msu.edu/classifier/classifier.jsp. Download the fullrank result when you're done. If you're working with 18S, 23S or 28S, I recommend the SINA classifier. Its online version only accepts 1000 sequences at a time. You can choose to divide your file into chunks of 1000 sequences, and then concatenate the results, or you can download and run the SINA classifier locally: http://www.arb-silva.de/aligner/

STEP 12: Creating an OTU table

There are scripts on the Usearch homepage to do this. Here we use another alternative. It'll produce a table with OTUS on the lines, samples on the columns and the classification for each read and the sequence of the representative at the end of each line.

If you use the RDP classifier, you can choose a confidence cut-off – classification assignments with lower confidence will be disregarded. With either RDP or SINA you also have the choice of assigning a fixed depth of classification, and all finer classifications will be disregarded. If you want the whole classification without any cut-offs, choose 0 as minimal confidence and a large number as maximum depth. If you don't give any parameters, a cut-off of 50% confidence will be taken for RDP files and a depth of 5 for Silva files.

With online SINA you can choose different databases to use (EMBL, Greengenes, LTP, RDP and Silva, in this order). This script will only consider the last classification for each line, so consider that when choosing which databases to use.

Every classification file that you want included in your OTU table should be in the same folder, and no other files should be in it.

The command:

```
perl otu_tables --threshold=INTEGER --samples=<FOLDER>
        --classification=<RDP_FILE> --sequences=<FASTA>
```

or

```
perl otu_tables --depth=INTEGER] --samples=<FOLDER> --classification=<SINA_FILE>
        --sequences=<FASTA>
```

Example:

```
perl otu_tables --threshold=50 –samples=all_reads/
        –classification=otus97.num.fa_classified.txt –sequences=otus97.num.fa
```

or

```
perl otu_tables –depth=5 –samples=all_reads/
        --classification=otus97.csv –sequences=otus97.num.fa
```