```
Script started on Sun Sep 25 09:24:57 2016
$ cat tally.pv
#!/usr/bin/python
# IN CLASS 8
 # Goal open a Illumina fastq file of and count the number of barcodes
 in_file = "/home/a-m/ib501_stud12/shel1/s_1_sequence.txt"
 # open file
 fh = open(in_file, 'r')
 # make barcodes dictionary
barcodes = {}
 count = 0
 # skip record name
next(fh)
 for line in fh:
       count = count + 1
        if count % 4 == 1:
               # make uniq barcode if it doesnt exist or increment by one if it does
               barcodes[line] = barcodes.get(line, 0) + 1
# print uniq barcodes and their counts
print barcodes.items()
 # Part 2
 # Goal:
# - Translate the quality scores for each FASTQ record
# - count the number of nucleotides below a quality score threshold.
# - Tally the number of reads that fall below the threshold
# - Print the number of sequences read and the number dropped.
fh2 = open(in_file, 'r')
# quality threshold parameters
phred_thresh = 50
perc highO nucs = 0.8
quality = {} count = 0
num_bad_reads = 0
 num_good_reads = 0
 # skill record name, seq, quality name
next(fh2)
next(fh2)
 next(fh2)
 for line in fh2:
        count = count + 1
        good nuc = 0
        bad nuc = 0
        qual_type = ""
       if count % 4 == 1:
               for i in line:
                      # convert ASCII to ints offset by 33
                      g score for nuc = ord(str(i)) - 33
                      # asign good gual threshold
                      if int(q_score_for_nuc) > phred_thresh:
                              good_nuc = good_nuc + 1
                      else:
                              bad_nuc = bad_nuc + 1
                 Determine whether read is high qual
               if float(good_nuc) / (float(bad_nuc) + float(good_nuc)) > perc_highQ_nucs:
                      qual_type = "High_qual"
num_good_reads = num_good_reads + 1
quality[count] = [good_nuc, bad_nuc, qual_type]
               else:
                      qual_type = "Bad_qual"
                      num_bad_reads = num_bad_reads + 1
                      quality[count] = [good nuc, bad nuc, qual type]
       else:
               continue
 fh2.close
 # print quality.items()
 total reads = num bad reads + num good reads
 print "\nThere were %s number of good reads out of %s total reads" % (num_good_reads, total_reads)
 print "* Good reads = %s percent of nucleotides in a read had a phred score > %s" % (perc highO nucs * 100, phred thresh
print num bad reads, "were bad reads and would be dropped "
 ^[]0;ib501_stud12@compute-0-19:~/shell^G^[[01;31m^B09:25:09 ^[[01;32m^Bib501_stud12 ^[[02;36m^Bcompute-0-19 ^[[01;34m^B/
home/a-m/ib501 stud12/shell ^[[00;33m^B^[[00m
('GCCT', 11), ('AAATG', 7), ('CATAA', 5), ('AAATC', 6), ('GCCCC', 1), ('AAATA', 7), ('CCTAT', 1), ('TTAGC', 2), ('GCCC T', 4), ('AAATT', 32), ('GCGTG', 3), ('GCGTA', 1), ('CCTAG', 4), ('CCTAA', 1), ('CCTAC', 2), ('CCTAG', 3), ('TGGTC', 57), ('GGCCG', 1), ('GGCTG', 9), ('TGGTC', 79), ('GGCTA', 2), ('TGGTC', 5), ('AGACA', 2), ('TTACT', 6), ('AGACG', 1), ('GCTG', 4), ('GCGG', 2), ('GATAC', 7), ('AGACG', 1), ('GATAA', 4), ('GTGGT', 4), ('TGGTT', 14508), ('CCCCA', 1), ('GATAA', 4), ('GTGT', 4), ('TGGTT', 1), ('GTGT', 1), ('GTGT', 1), ('GTGTT', 1), (
```

```
( 2), ('GATAT', 36), ('TIACA', 5), ('AGACT', 4), ('TIACC', 17), ('CCCAT', 4), ('AGCAA', 3), ('TCGCC', 2), ('AGCAT', 29), ('GCCC', 2), ('GATCC', 2), ('CATCC', 2), ('CATCC'
                                                   2). ('GATAT'.
                                                                                                                                                                                                                                             36), ('TTACA', 5), ('AGACT', 4), ('TTACC', 17), ('CGCAT', 4), ('AGCAA', 3), ('TCGCC', 2), ('AGCAC
("GARGT", 99) ("AGTCC", 2), ("TGARA", 12), ("ATTGT", 5), ("GARGG", 21), "GARGA", 58), ("AATCT", 32), ("GARGG", 1), ("ACGG", 1), ("ACGG", 1), ("GARGG", 2), ("ATTGG", 2), ("TCCAT", 16), ("GNAGC", 8), ("GARGG", 1), ("ACGG", 1), ("ACGG", 1), ("ACGG", 1), ("ACGG", 1), ("GARGG", 1), ("ACGG", 1), ("TGCT", 2), ("GARGG", 1), ("ACGG", 2), ("ANTTG", 3), ("ACGT", 1), ("ACGG", 1), ("ACGG", 1), ("ACGG", 2), ("ANTGG", 1), ("ACGG", 1), ("ACGG", 1), ("ACGG", 2), ("ANTGG", 1), ("ACGG", 2), ("ANTGG", 1), ("ACGG", 2), ("ANGGG", 2), ("ANGGG", 1), ("ACGGG", 1), ("ACGGG", 4), ("CAGG", 4), ("CAGG", 4), ("ANGGG", 4), ("CAGGG", 4), ("ANGGG", 4), ("ANGGG"
   [ 5], ('TGGAC', 17), ('ATGCT', 46), ('TGGAT', 23), ('TCTCT', 5), ('AGCTT', 2), ('GCAGT', 15), ('CGTT', 8), ('GTT', ('GGAGT', 11), ('GCGAT', 4), ('ANTGC', 5), ('CCATT', 26), ('ATATA', 'CGGAGY', 26), ('ACACT', 9), ('GGAGG', 34), ('CCCAT', 14), ('GGAGA', 32), ('GAGAC', 44), ('GTTGT', 5), ('ATATT', 'CGGAG', 26), ('ACACT', 9), ('GGAGA', 71), ('TGTGT', 12), ('GAGAG', 28), ('ACTCT', 13), ('GAATA', 11), ('ATTTA', 41), ('ATTTA', 41), ('GGGAA', 71), ('ATTTC', 81), ('GGGAG', 3), ('ATTTG', 9), ('GAATT', 51), ('ACTCA', 81), ('TGTC', 12), ('TGTGA', 2), ('TGTGC', 13), ('CCTGG', 3), ('ATTTG', 9), ('GAATT', 51), ('ACTCA', 81), ('TGTC', 12), ('CAGTC', 12), ('AGCTC', 12), ('CAGTC', 12), ('CAGTC', 12), ('AGCTC', 12), ('CAGTC', 12), ('CAGTC', 12), ('AGCTC', 12), ('AGCTC', 12), ('CAGTC', 12), ('CAGTC', 12), ('AGCTC', 12), ('AGGTC', 12), ('AG
```

', 12), ('AGCTC', 2), ('GCATC', 71), ('CTATT', 23), ('TGCAT', 15), ('GCAGA', 17), ('TACCA', 7), ('TACCC', 6), ('TAGTT', 34), ('GGCCT', 3), ('GGCAT', 3), ('CCCGT', 2), ('AGAGG', 1), ('AGGGGT', 6), ('CATCT', 19), ('AGAGC', 1), ('GGCGA', 1), ('AGGGT', 6), ('CAGGT', 1), ('CAGGC', 4), ('CAGGT', 1), ('CAGGT', 2), ('AGAGT', 22), ('AGAGT',