QAA_Lab_Report

Julia Jones

2024-09-08

File assignments

Under the path: `projects/bgmp/shared/2017_sequencing/demultiplexed/`

File	Alias
34_4H_both_S24_L008_R1_001.fastq.gz	34_4H_R1
34_4H_both_S24_L008_R2_001.fastq.gz	34_4H_R2
21_3G_both_S15_L008_R1_001.fastq.gz	21_3G_R1
21_3G_both_S15_L008_R1_001.fastq.gz	21_3G_R2

Table 1: My file assignments

File	Size	Read Count	Read Length
21_3G_R1	480M	9,237,299	101
21_3G_R2	535M	9,237,299	101
34_4H_R1	469M	9,040,597	101
34_4H_R2	525M	9,040,597	101

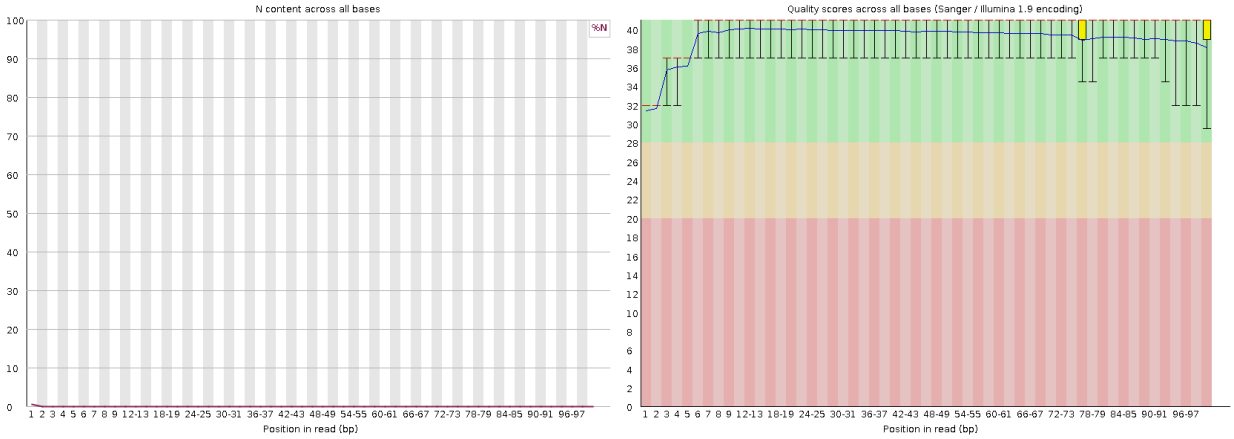
Table 2: Relevant information about my files

FastQC

21_3G

Score	On	Original File
PASS	Per base sequence quality	21_3G_R1
PASS	Per base N content	21_3G_R1

Table 3: FASTQC 21_3G_R1 output overview



(a) % N content by position

(b) Median and IQR of quality score by position

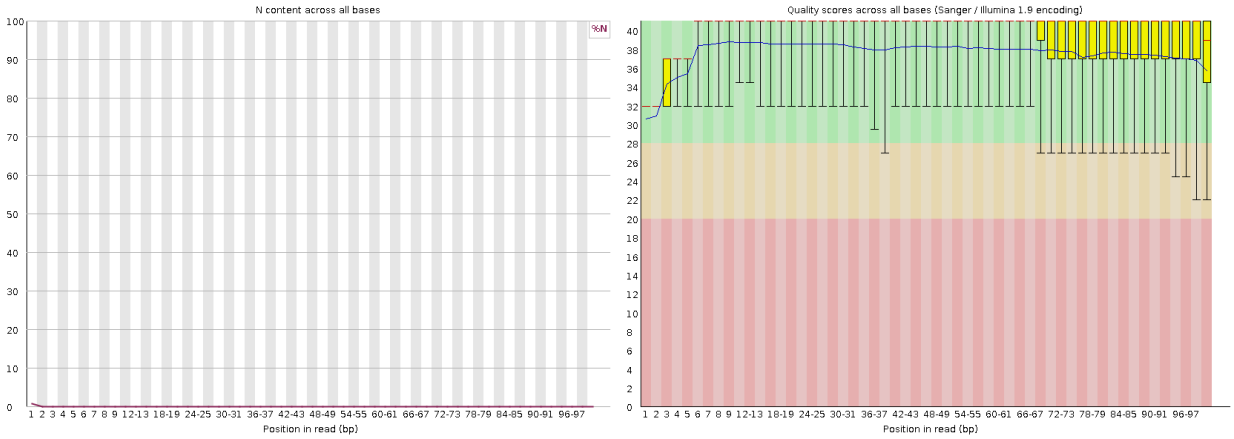
Figure 1: FASTQC output for 21_3G_R1

N-content stays extremely low throughout the reads, even at the susceptible early positions. As there are no quality data outliers in the red zone of the quality score distribution plot, this lack of N-content makes sense

Quality is worse at the lower positions, while quality score variation increases at the highest positions. Despite this all qualities lie within the acceptable region.

Score	On	Original File
PASS	Per base sequence quality	21_3G_R2
PASS	Per base N content	21_3G_R2

Table 4: FASTQC 21_3G_R2 output overview



(a) % N content by position

(b) Median and IQR of quality score by position

Figure 2: FASTQC output for 21_3G_R2

N-content stays extremely low throughout the reads, even at the susceptible early positions. As there are no quality data outliers in the red zone of the quality score distribution plot, this lack of N-content makes sense

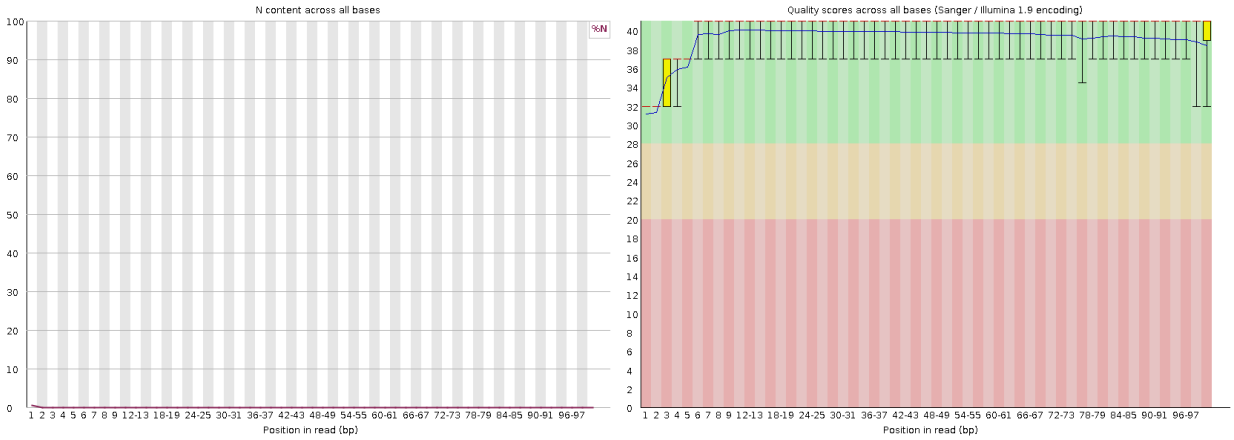
Average quality is worse at the lower positions, while quality score variation increases at the highest positions.

Despite this all median qualities lie within the acceptable region. The outlier warning qualities at positions 66-97 are something to look out for when examining other metrics.

34_4H

Score	On	Original File
PASS	Per base sequence quality	34_4H_R1
PASS	Per base N content	34_4H_R1

Table 5: FASTQC 34_4H_R1 output overview



(a) % N content by position

(b) Median and IQR of quality score by position

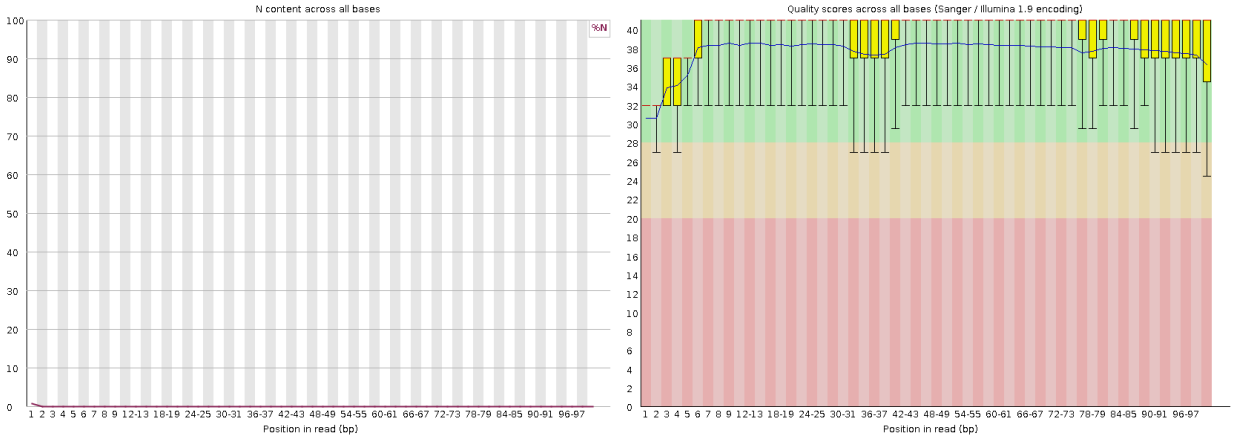
Figure 3: FASTQC output for 34_4H_R1

N-content stays extremely low throughout the reads, even at the susceptible early positions. As there are no quality data outliers in the red zone of the quality score distribution plot, this lack of N-content makes sense

Quality is worse at the lower positions, while quality score variation increases at the highest positions. Despite this all qualities lie within the acceptable region.

Score	On	Original File
PASS	Per base sequence quality	34_4H_R2
PASS	Per base N content	34_4H_R2

Table 6: FASTQC 34_4H_R2 output overview



(a) % N content by position

(b) Median and IQR of quality score by position

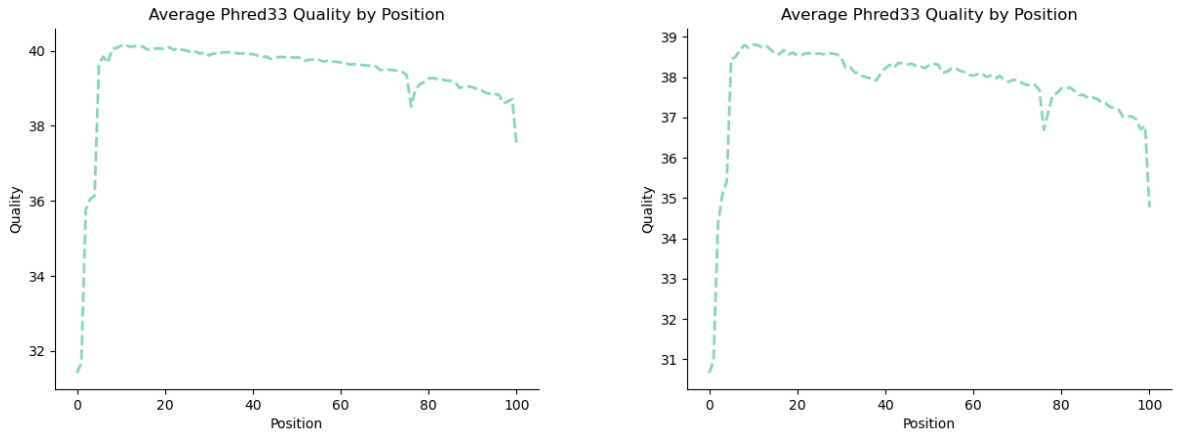
Figure 4: FASTQC output for 34_4H_R2

N-content stays extremely low throughout the reads, even at the susceptible early positions. As there is no quality data outliers in the red zone, this lack of N-content makes sense

Average quality is worse at the lower positions, while quality score variation increases at 31-32 and 72-101. All median qualities lie within the acceptable region. The outlier qualities at positions 31-32 and 72-101 are something to look out for when examining other metrics.

My Quality Score Script

21_3G



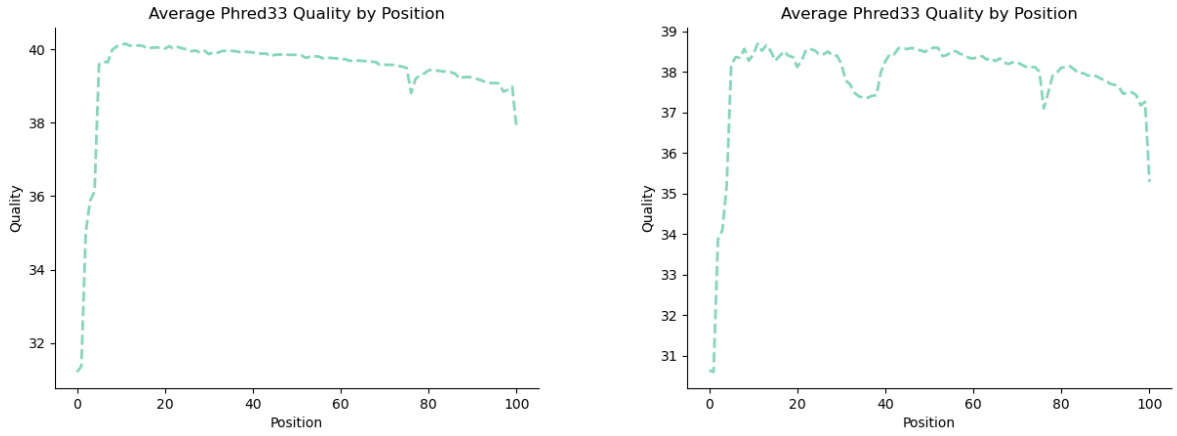
(a) R1

(b) R2

Figure 5: Average quality score by position for 21_3G

R2 means seems to be consistently lower than R1 means by 1 point. This slight difference is probably insignificant.

34_4H



(a) R1

(b) R2

Figure 6: Average quality score by position for 34_4H

R2 means seems to be consistently lower than R1 means by 1 point (plus a small region at 35-40 that is lower by 3-4 points). This slight difference should be insignificant.

My script vs. FASTQC

Quality of quality score distribution plots

The FASTQC data plots impart much more information than my own. My plots solely map the mean quality distribution by position. Thus, I cannot see when my quality scores are lowered due to outliers or from low quality as a whole. In contrast, FASTQC calculates the mean, median, and IQR. And while my mean line and FASTQC's mean line look identical, the FASTQC's multi-metric methodology handles outliers much better.

FASTQC also incorporates great visual cues into their plots: a hard ylim of 40 and quality color coding in the figure's background. These additions make interpreting quality scores much simpler at a glance. In my code, I did not set a hard ylim. So, if a whole run is relatively worse than normal, my y-axis maximum will be lower than 40 (i.e., Figure 6b) this coupled with the lack of background color coding can mask the low-quality nature of the run at first glance.

Runtime difference

File	Script	Runtime (seconds)
21_3G_R1	mine	304.88
21_3G_R1	FASTQC	38.27*
21_3G_R2	mine	308.32
21_3G_R2	FASTQC	38.27*
34_4H_R1	mine	307.19
34_4H_R1	FASTQC	38.27*
34_4H_R2	mine	317.67
34_4H_R2	FASTQC	38.27*
average	mine	309.52
average	FASTQC	38.27

File	Script	Runtime (seconds)

Table 7: FASTQC vs. my run time

∗Ran all files at once and divided total run time by 4.

On average, FASTQC is `(309.52/38.27)=8.1` times **faster** than my code!

This large disparity in speed makes sense. I wrote my De-multiplexing code in one week in Python, while FASTQC was built by a team in 2010 in Java (a generally faster language) and continually improved over the past decade. Even in the one week I spent on De-multiplexing I was able to greatly speed up my code. This is a case of disparity in time and resources.

Memory/CPU usuage

File	Script	% CPU used
21_3G_R1	mine	99
21_3G_R1	FASTQC	97
21_3G_R2	mine	99
21_3G_R2	FASTQC	97
34_4H_R1	mine	99
34_4H_R1	FASTQC	99
34_4H_R2	mine	99
34_4H_R2	FASTQC	97

Table 8: FASTQC vs. my data usage

CPU usage was consistent across runs, with FASTQC using slightly less CPU than my own code. This difference is slight enough to be insignificant.

Comment on overall data quality

21_3G Overall quality

Recommendation	Metric	File
PASS	Per base sequence quality	R1, R2
PASS	Per base N content	R1, R2
PASS	Sequence Length Distribution	R1, R2
WARN	Sequence Duplication Levels	R1, R2
FAIL	Per tile sequence quality	R1, R2
FAIL	Per base sequence content	R1
WARN	Per base sequence content	R2
PASS	Per sequence GC content	R1, R2
WARN	Overrepresented sequences	R1, R2
PASS	Adapter Content	R1, R2

Table 9: FASTQC Calls on all metrics

Per base sequence quality, Per base N content

This metric is not a concern with this data. See **FASTQC** section above for more information.

Sequence Length Distribution

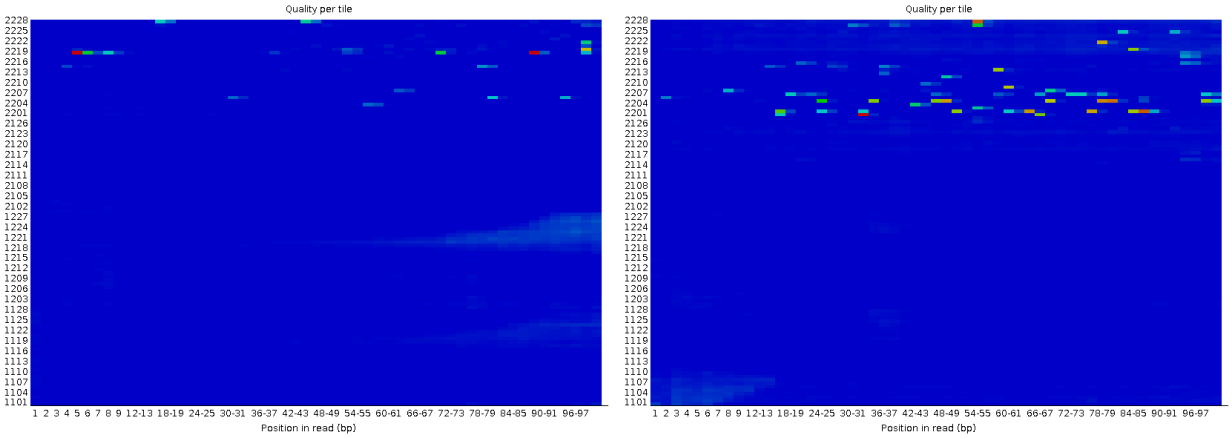
All sequences were length 101. See **File assignments** section above for more information.

Sequence duplication levels

Because this is an RNA-seq experiment, we don't care about sequence duplication levels. In RNA-seq, sequence duplication imparts differential expression.

Per tile sequence quality

Both R1 and R2 failed this metric, lets examine the tile results in more detail.



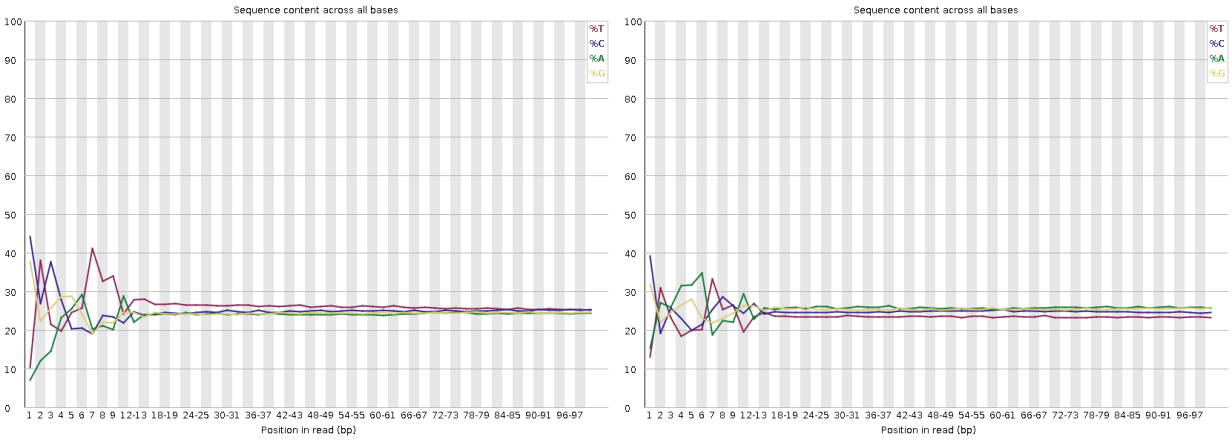
(a) R1: Fail

(b) R2: Fail

Figure 7: Per tile sequence quality for 21_3G

An ideal per-tile quality plot should be completely blue (high quality). Here our R1 file has some low quality hotspots from the 3' and 5' end of reads constructed in tile 2219. This tile constructed a small proportion of our total R1 reads. Though R2 has more middling quality tiles, its number of consistently low-quality tiles is small (2201, 2204). Thus, I would not consider tile-quality to be a concern for our data.

Per base sequence content

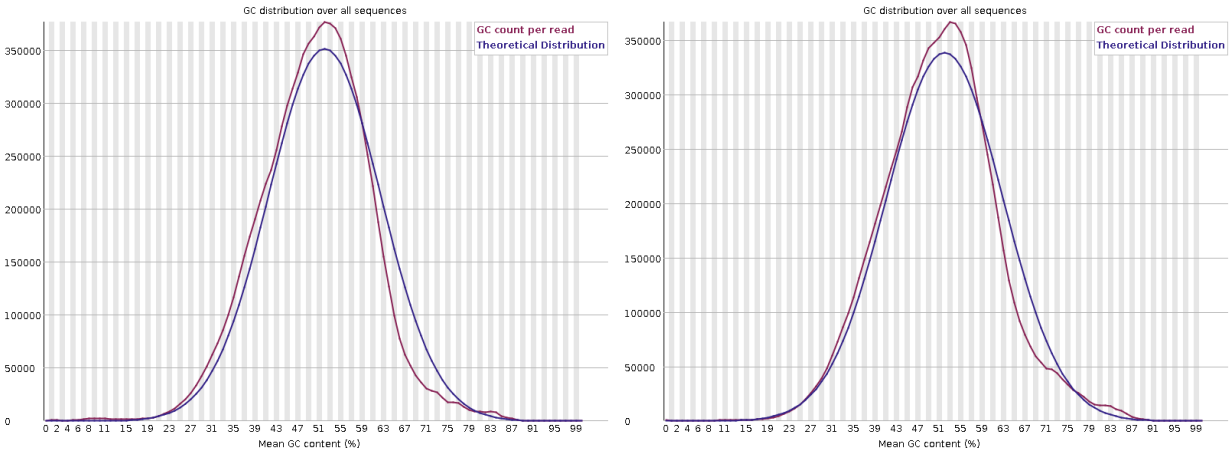


(a) R1: Fail (b) R2: Warn

Figure 8: Per base sequence content for 21_3G

In an ideal per-base content graphs the A,T, C, and G y-values would remain consistent as you move along the x-axis. In our data, the y-values (sequence contents) spike and drop until position 13 where they level out. Thus, R1 is called a fail, and R2 is a warning. R1 fails because its T sequence content spike from position 5-9 is higher than anything in our R2 plots. But, because this y-value volatility is localized to the start of the reads, where we have low-quality data to be trimmed downstream by `Trimmomatic`, I would not be concerned by R1's or R2's per-base sequence content.

Per base GC content



(a) R1: Pass (b) R2: Pass

Figure 9: Per base GC content for 21_3G

For both R1 and R2, our GC content closely follows the theoretical distribution provided by FASTQC. The center of our distribution is a bit higher than the theoretical distribution, but as this is the value where our

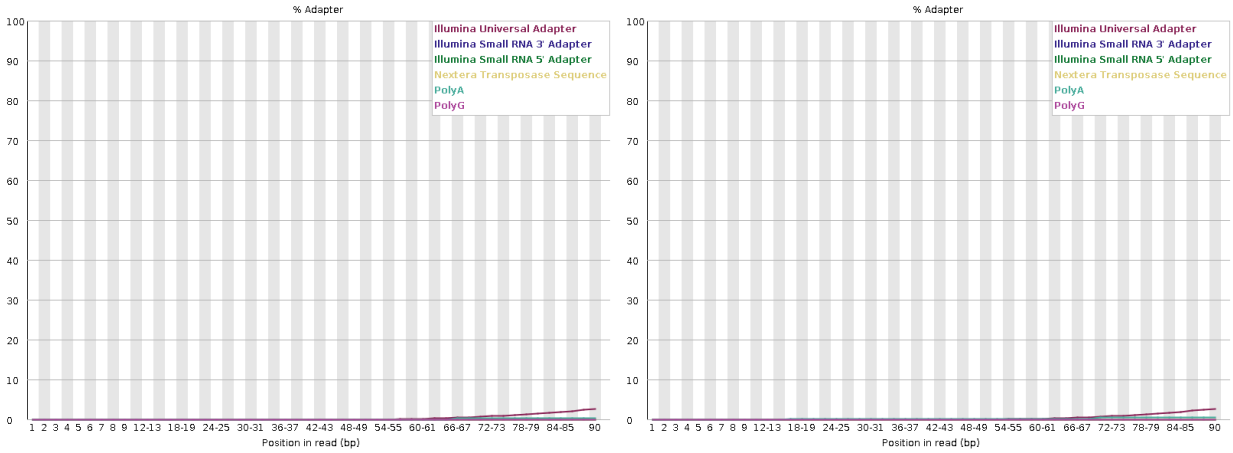
nucleotides are balanced (50%) this is not problematic. Thus, GC content is likely not an area of concern here.

Overrepresented sequences

File	Source	Count	Percentage
R1	TruSeq Adapter, Index 18	29130	0.32
R2	Illumina Single End Sequencing Primer	30251	0.33

Though we got a warning for R1 and R2, I would not call this data problematic (even though `21_3G` has more overrepresented sequences than `34_4H`). Both of our over represented sequences are known adapters, and thus will be captured and trimmed downstream with `cutadapt`. Thus over represented sequences are not an area of concern for our data.

Adapter Content



(a) R1 (b) R2

Figure 10: Adapter content 21_3G

As you approach the 3' end of our read, adapter content does increase. In both our R1 and R2 files this difference is slight. Furthermore, the vast majority of this adapter content is the Illumina universal adapter which will be trimmed with `cutadapt`. Thus, adapter content is not a concern for this dataset.

21_3G Final reccomendation

With the metrics discussed above in mind, it's is safe to say that both `21_3G` files are high enough quality to proceed with further analysis. Most metrics passed, and those that produced a warning or failure make sense (sequence duplication) or can be fixed downstream (per-base sequence content).

34_4H Overall quality

Recommendation	Metric	File
PASS	Per base sequence quality	R1, R2
PASS	Per base N content	R1, R2
PASS	Sequence Length Distribution	R1, R2
WARN	Sequence Duplication Levels	R1, R2
FAIL	Per tile sequence quality	R1, R2
FAIL	Per base sequence content	R1
WARN	Per base sequence content	R2
PASS	Per sequence GC content	R1, R2
WARN	Overrepresented sequences	R1, R2
PASS	Adapter Content	R1, R2

Table 9: FASTQC Calls on all metrics

Per base sequence quality, Per base N content

Overall, this metric is not a concern with this data. See **FASTQC** section above for more information.

Sequence Length Distribution

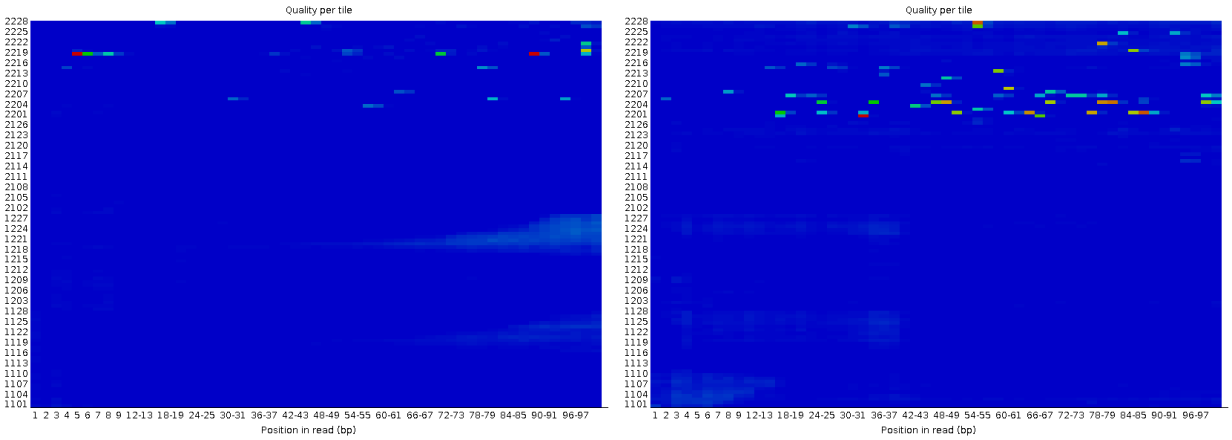
All sequences were length 101. See **File assignments** section above for more information.

Sequence duplication levels

Because this is an RNA-seq experiment, we don't care about sequence duplication levels. In RNA-seq, sequence duplication imparts differential expression.

Per tile sequence quality

Both R1 and R2 failed this metric, lets examine the tile results in more detail.



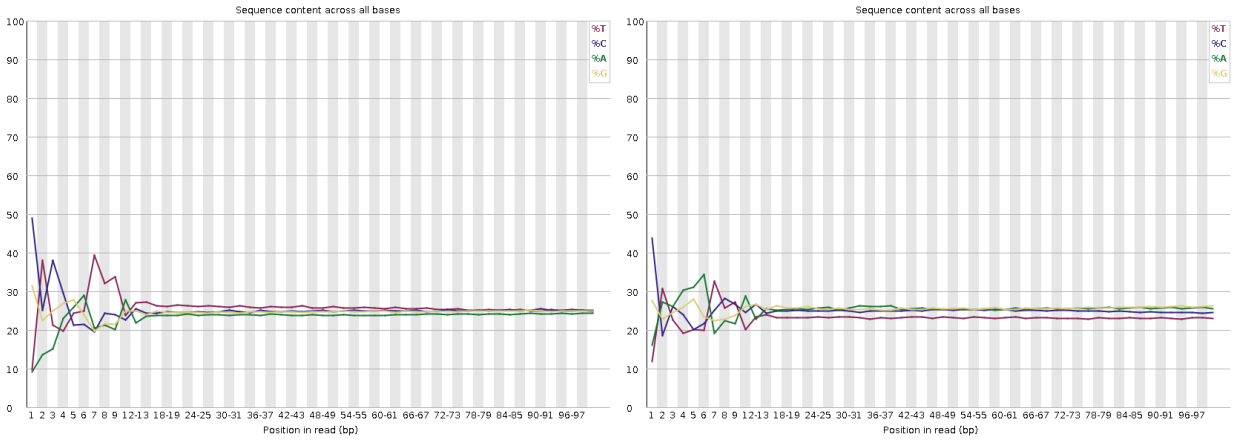
(a) R1: Fail

(b) R2: Fail

Figure 11: Per tile sequence quality for 34_4H

An ideal per-tile quality plot should be completely blue (high quality). Here our R1 file has some low quality hotspots from the 3' and 5' end of reads constructed in tile 2219 and 22222. These tiles constructed a small proportion of our total R1 reads. Though R2 has more middling quality tiles, its number of low-quality tiles is small. Thus, I would not consider tile-quality to be a concern for our data here.

Per base sequence content

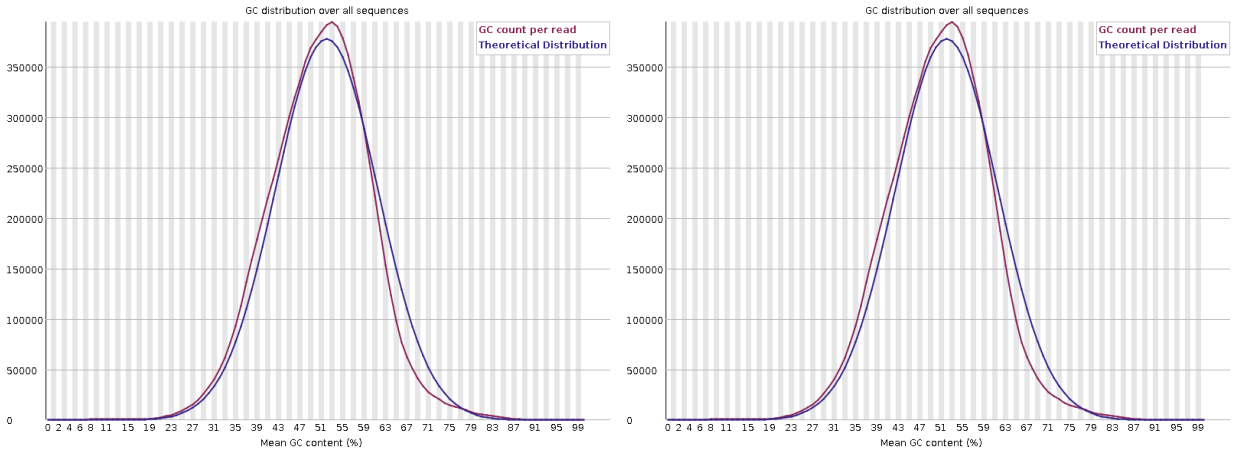


(a) R1: Fail (b) R2: Warn

Figure 12: Per base sequence content for 34_4H

These R1 and R2 graphs are almost identical to the `21_3G` graphs, with slightly more volatility around the first base. Recall, in an ideal per-base content graphs the A, T, C, and G y-values would remain consistent as you move along the x-axis. In our data, the y-values (sequence contents) spike and drop until position 13 where they level out. Thus, R1 is called a fail, and R2 is a warning. R1 fails because its T sequence content spike from position 6-12 is higher than any spike in our R2 plots. But, because y-value volatility is localized to the start of the reads, where our low-quality data will be trimmed downstream, I would not be concerned by R1's or R2's per-base sequence content.

Per base GC content



(a) R1: Pass (b) R2: Pass

Figure 13: Per base GC content for 34_4H

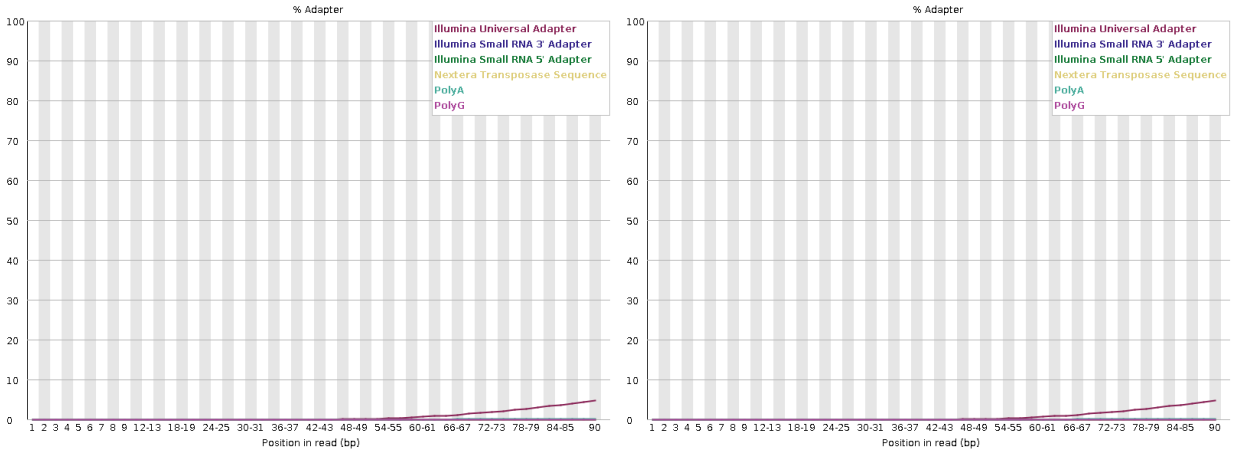
For both R1 and R2, our GC content closely follows the theoretical distribution provided by FASTQC. Thus, GC content is likely not an area of concern here.

Overrepresented sequences

File	Source	Count	Percentage
R1	TruSeq Adapter, Index 15	9502	0.11
R2	Illumina Single End Sequencing Primer	9611	0.11

Though we got a warning for R1 and R2, I would not call this data problematic. Both of our over represented sequences are known adapters, and will be captured and trimmed downstream with `cutadapt`. Thus overrepresented sequences are not an area of concern for our data.

Adapter Content



(a) R1: Pass

(b) R2: Pass

Figure 14: Adapter content 34_4H

Adapter content appears to be identical between R1 and R2. As you approach the 3' end of our read, adapter content does increase. Adapter content at the 3' end is higher in `34_4H` than `21_3G`, though for both replicates the level of adapter content is not a concern. This is because the vast majority of the adapter content is the Illumina universal adapter which will be trimmed with `cutadapt`. Thus, adapter content is not a concern for this dataset.

34_4H final quality determination

With the metrics discussed above in mind, it's is safe to say that both `34_4H` files are high enough quality to proceed with further downstream analysis. Though my `21_3G` data is higher quality than my `34_4H` most metrics passed, and those that produced a warning or failure make sense (sequence duplication) or can be fixed downstream (per-base sequence content).

Part 2

Cutadapt

Determine adapters
Mechanistically in one run, adapters won't be present in every sequence. They may not even be the top hit. But because Illumina uses universal adapters, we can compare across disparate runs (`34_4H_RN` and `21_3G_RN`). Comparing across files, the universal adapter is the unifying sequence.

File name	Adapter
21_3G_R1	AGATCGGAAGAGCACACGTCTGAACTCCAGTCA
21_3G_R2	AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
34_4H_R1	AGATCGGAAGAGCACACGTCTGAACTCCAGTCA
34_4H_R2	AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

Table 9: Universal adapter by file

With my adapters determined, I could proceed to actually running `cutadapt`

Cutadapt results

File name	Percent adapter trimmed
21_3G_R1	6.6
21_3G_R2	7.4
34_4H_R1	9.1
34_4H_R2	9.8

Table 10: Percentage of file requiring adapter trimming

Cutadapt interpretation
As discussed in the **34_4H Overall quality** section, my `34_4H` data had higher adapter content (and were trimmed more) than my `21_3G` data. Thus, it is not surprising that my `34_4H` files were more adapter trimmed than my `21_3G` files. Nevertheless, in all my assigned files, the percent that needed to be adapter trimmed were relatively low. In both `21_3G` and `34_4H` our R2 files were trimmed more than our R1 files. This makes sense because R2 runs last on the sequencer and is typically lower quality than R1. For more information on why R2 files were more trimmed than R1 files see the section **R1 and R2 adapter trimmed at different rates?** below.

Trimmomatic

With our adapters removed, our data is now ready for `Trimmomatic`

File	Percent trimmed
21_3G_R1	10.8
21_3G_R2	17.3
34_4H_R2	13.1
34_4H_R2	18.6

Table 11: Percentage of reads that have been trimmed (including adapter trimming)

R1 files were trimmed more than R2 files and that `21_3G` had lower trimming levels than `34_4H`. This difference in trimming is unsurprising given our `FASTQC` quality assessment above.

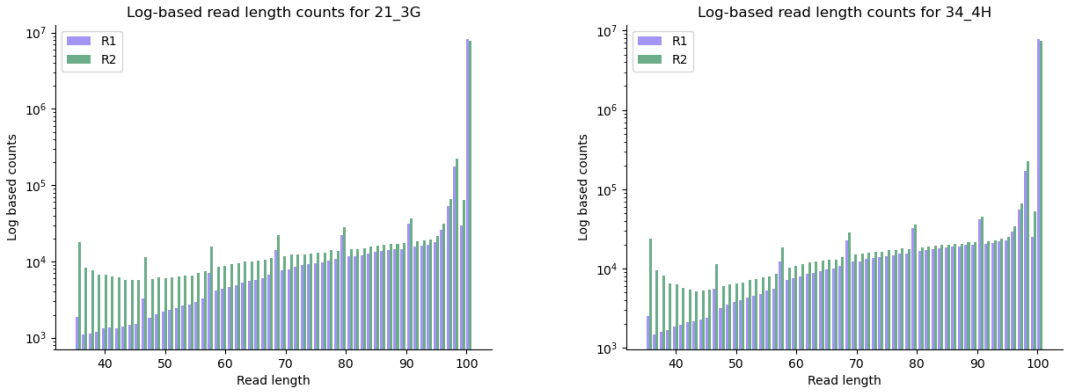


Figure 15: Read length distribution by file

The summary statistics in Table 11 translate over into Figure 15. The R2 bins are consistently taller than our R1 bins (save for when no trimming was performed), while the `21_3G` graph has more reads at the no trimming performed bin than `34_4H`.

R1 and R2 adapter trimmed at different rates?

R2 files should be adapter trimmed at higher rates than R1 files because R2 files are completed at the end of the run. As the run continues reagents and the signal degrade. Signal degradation causes adapter bases to be called over the insert bases, causing adapters to be over represented. This contributes to the overall lowering of run quality, which we can see from our FASTQ output above.

Part 3

Genome generate

Align reads

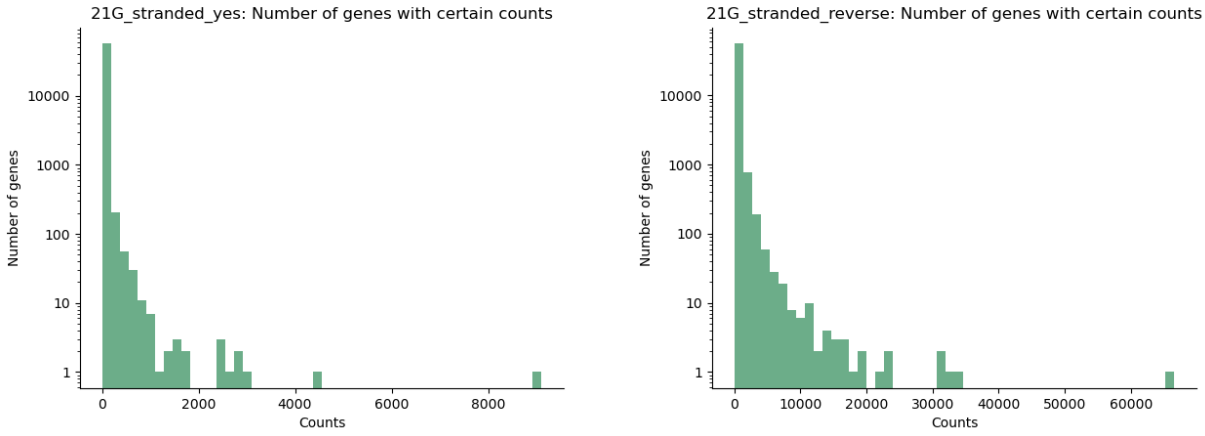
Replicate	Mapped reads	Unmapped reads	% mapped	% unmapped
21_3G	15,125,868	2,580,756	85.4	14.6
34_4H	15,631,049	1,675,233	90.3	9.7

Table 12: Percentage of mapped reads by replicate

Determining standedness

I propose that both my `21_3G` and `34_4H` libraries are stranded. The reasoning behind this determination is clear once one examines the `htseq-count` output in detail.

21_3G

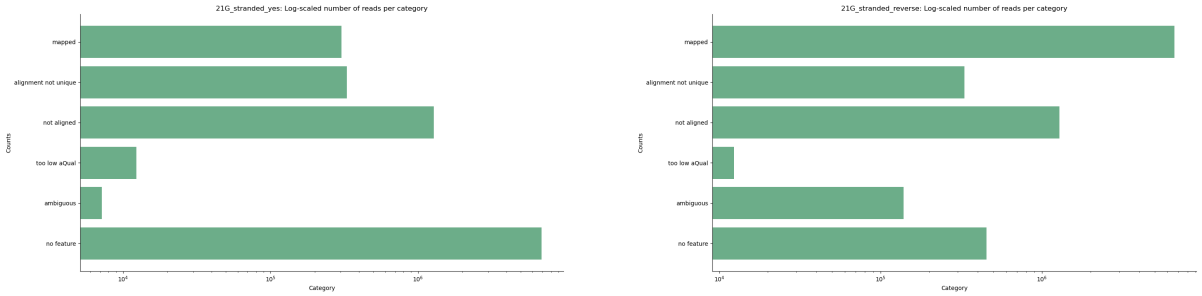


(a) Forward strand

(b) Reverse strand

Figure 16: Number of genes mapped X times by strand

If our data was unstranded, the number of genes mapped to specific runs would be the same whether the strand direction was set to forward or reverse. This is not the case: visually the reverse strand plot has much more data mapped than the yes (forward) strand plot. Numerically 303,726 reads mapped on the forward strand and 6,636,009 reads mapped on the reverse strand. This 2084.87% increase could not happen by chance. Still, lets zoom out on our `htseq-count` data and view our output categorically by strand.



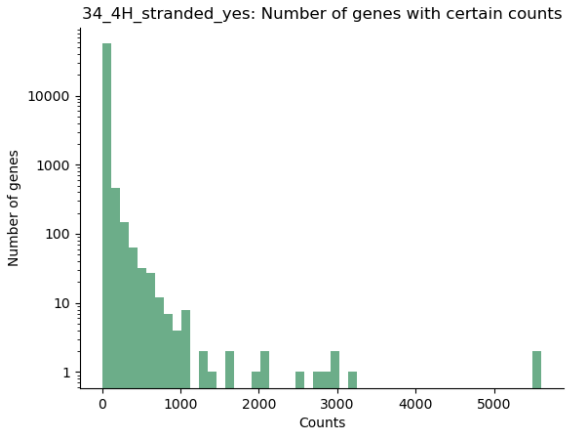
(a) Forward strand

(b) Reverse strand

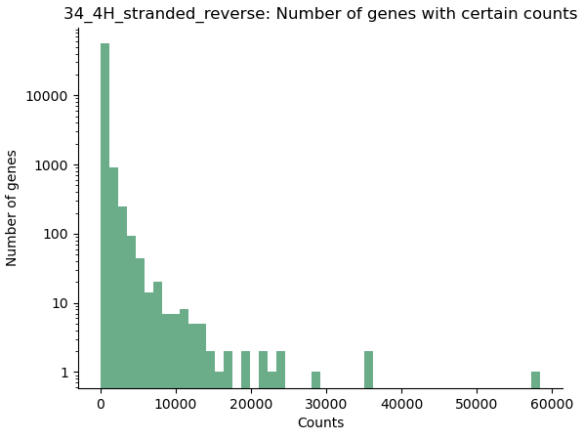
Figure 17: Number of reads within each category

This graph demonstrates how many reads mapped to each possible `htseq-count` category. In the yes (forward) strand data, *no feature* dominates our read categories. In the reverse strand data, *mapped* dominates our read categories. This categorical examination again clearly indicates that our data is stranded in the **reverse** direction.

34_4H



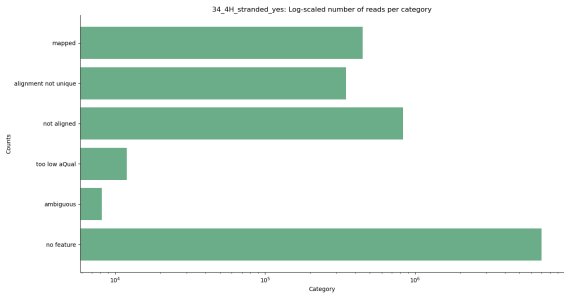
(a) Forward strand



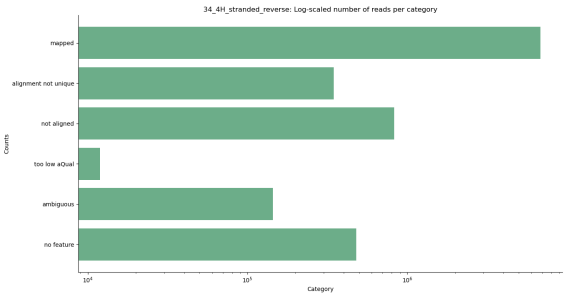
(b) Reverse strand

Figure 18: Number of genes mapped X times by strand

The same as was mentioned for `21_3G` holds true for `34_4H`. If our data was unstranded, the number of genes mapped to specific runs would be the same whether the strand direction was set to yes (forward) or reverse. This is not the case: visually the reverse strand plot has much more data mapped than the forward strand plot. Numerically, 449,409 reads mapped on the forward strand while 6,837,824 reads mapped on the reverse strand. This is a 1421.51% increase in read mapping, which could simply not occur by chance.



(a) Forward strand



(b) Reverse strand

Figure 19: Number of reads within each category

As mentioned in `21_3G`, this graph demonstrates how many reads mapped to each possible `htseq-count` category. In the yes (forward) strand data, *no feature* dominates our read categories. In the reverse strand data, *mapped* dominates our read categories. Though this trend is slighter here in the `34_4H` data, this is still conclusive evidence that our data is stranded in the **reverse** direction.