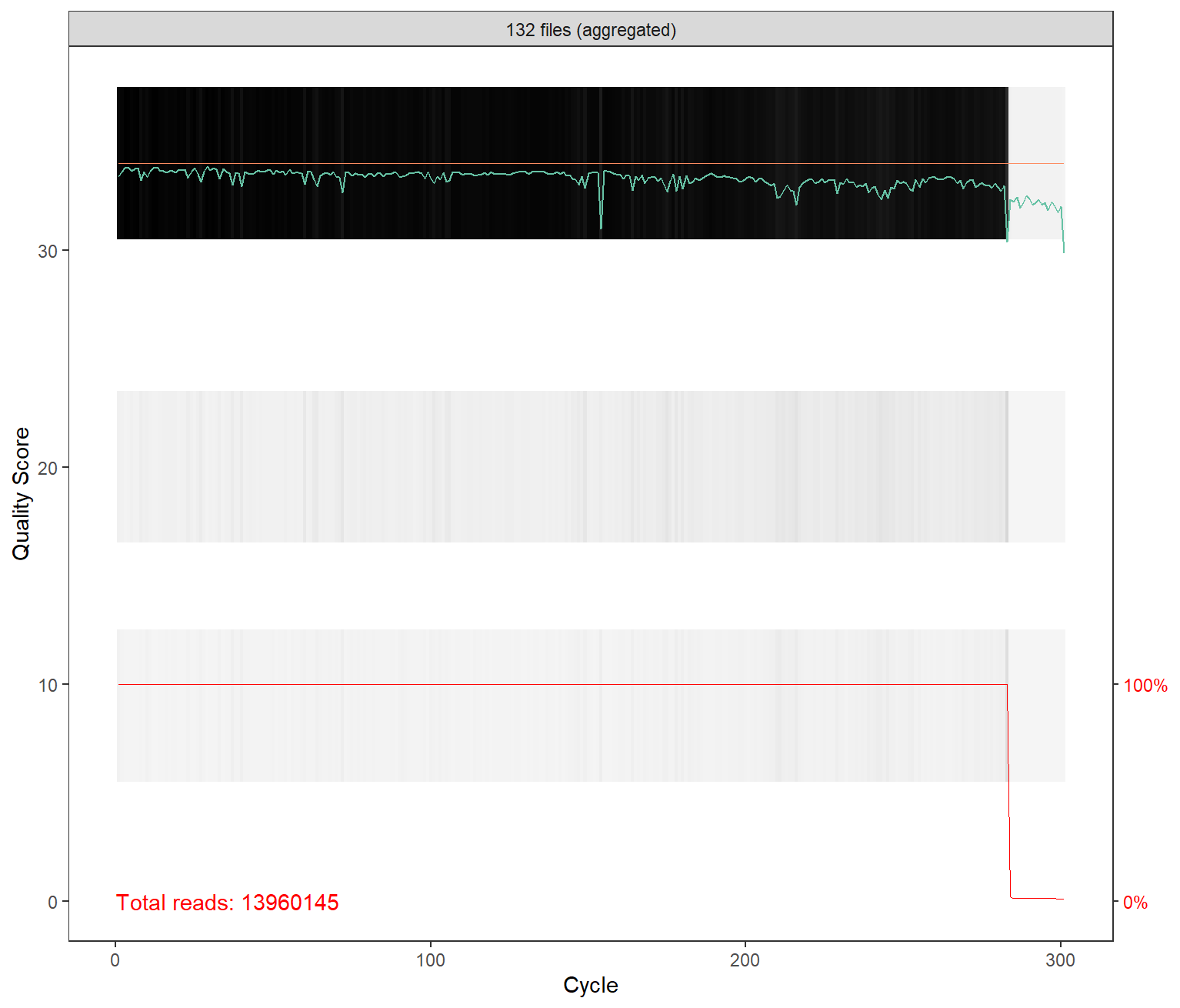
Let’s start from the beginning :

Cutadapt :

26/08/2024 : I realized something really dumb, but checking the raw 16S files, the same sequences always reappears, and it’s when you substract the adapter sequencer from the “amorce”  
It gives you a 18 nucleotide sequence (appropriate size for a primer)  
Might be this problem since the beginning…  
I ran again cutAdapt with my “new” primers



They are still some issues

Back from death week: week of ultra \*\*\*death

It was fixed later, they’re all the same size now, though they’re still issues when estimating the error rates. This is due to the binned nature of our quality scores.

|  |  |  |
| --- | --- | --- |
| **Qscore Bin** | **50, 100, 200 and 300 cycle kits** | **600 cycle kits** |
| Q0 | 0 | 0 |
| ~Q15 | 12 | 9 |
| ~Q20 | 26 | 20 |
| >Q30 | 34 | 34 |

We have 600 cycles kits

A graph with red text

Description automatically generatedhh

Going to try to adjust for that by testing on a subset of the data.

Error rates: A way to ensure an appropriate learning of the error rates is to ensure monotonicity or in other words: as the quality score increases the error rate always decreases or stabilizes but does not increase as the dips that we find sometimes in the weird error plots

A graph with a red line

Description automatically generated

A graph with black dots and red line

Description automatically generated

This is not acceptable This is more acceptable

Ways to ensure monotonicity consist of modifying the parameters of the loess function, the function that is used to estimate the build a error rates model (modeling probability of observing sequencing errors).

Loess = non-parametric regression which is going to smooth the relationship between observed error rates and quality score. It fits simple models at localized subsets of data, instead of applying one global model across all obersvations.

When quality scores are binned, it’s kinda fucked cause it works well with continuous data, not discrete data which is what we have with bins. As the binned nature of our data is part of the sequencing technology itself (NOVASeq) we don’t have the choice to deal with it.

Therefore, we can adjust the parameters of the loess function to ensure monotonicity. There are two parameters:

-span: how much of the data is used to fit each local regression model = how local the smoothing is.  
Between 0 and 1, and higher means more data is used.  
Too low = model might overfit, error rate estimation is less robust  
Too high = model might smooth out important variations, underfitting and missing of key error patterns

-degree: degree of polynomial used in local regression (1= linear and 2 = quadratic)

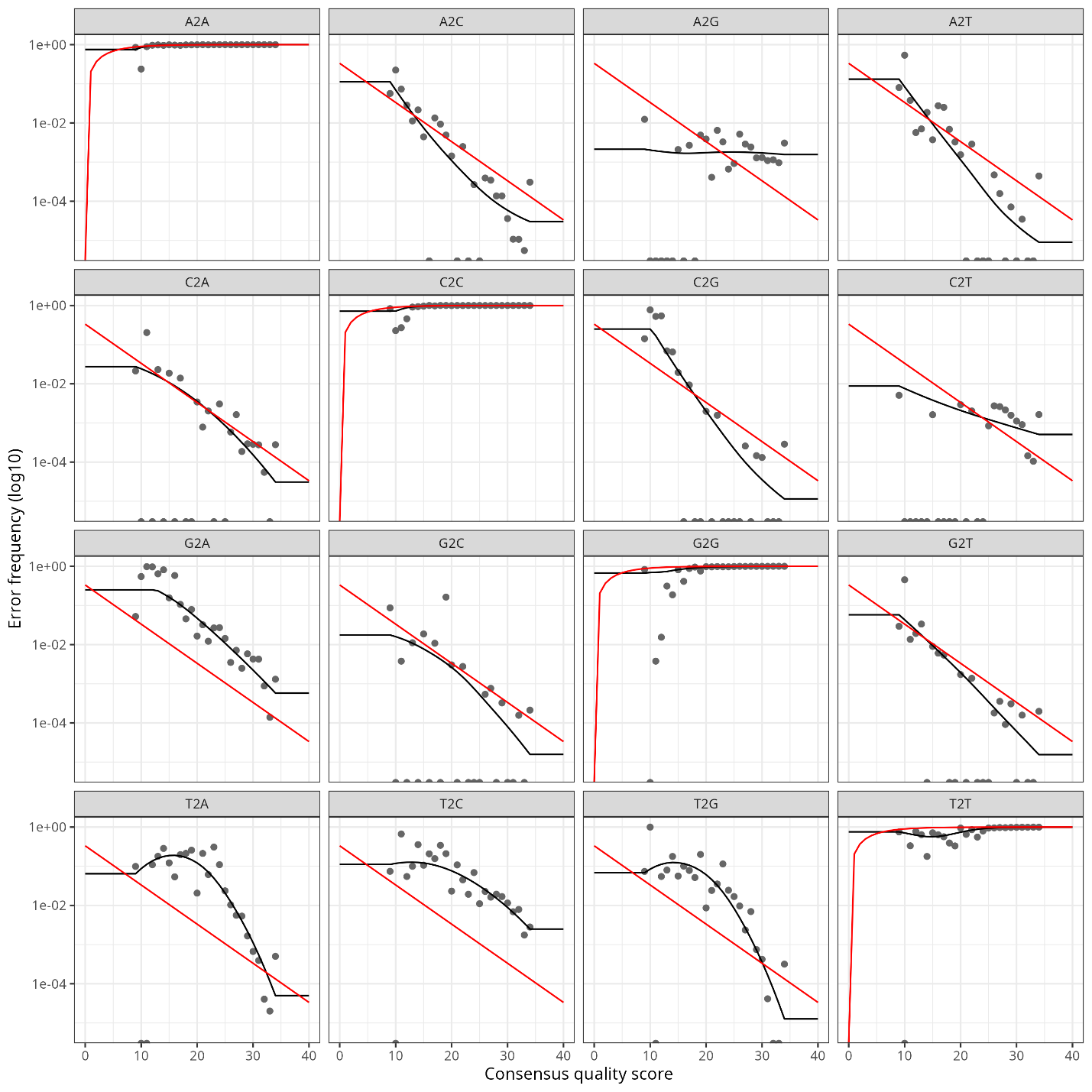
-weights: giving more or less influence to specific data points  
By default, each point contributes to the local regression based on its distance, the further it is, the less it contributes.  
As binning produces data aggregation

ChatGPT suggestions:  
0.8 span and degree 1. Now, finding only 4 forms of adapted loess function let’s test each and see how things go.

A graph of a graph of a function

Description automatically generated with medium confidenceBase error estimation not changing the loess function:

Model 1 (altering weights, span and enforcing monotonicity):



A graph of a graph of a number of numbers

Description automatically generated with medium confidenceModel 2 (enforcing monotonocity only):

Model 3 (alter weights and enforcing monotonicity):

A graph of different types of lines

Description automatically generated with medium confidence

Model 4 (altering weights, span and degree and enforcing monotonicity):

A graph of different types of graphs

Description automatically generated with medium confidence

Model 1 and 4 seem to the most appropriate, only difference is that in model 4 we play the with the degree, and I guess in 4 we use a linear model which would explain why some of the curves are very.. linear.

Let’s proceed with model 4 error rates estimation while also using model 4 in the dada function.