# P1 Research

Austin Spencer
09/29/2020

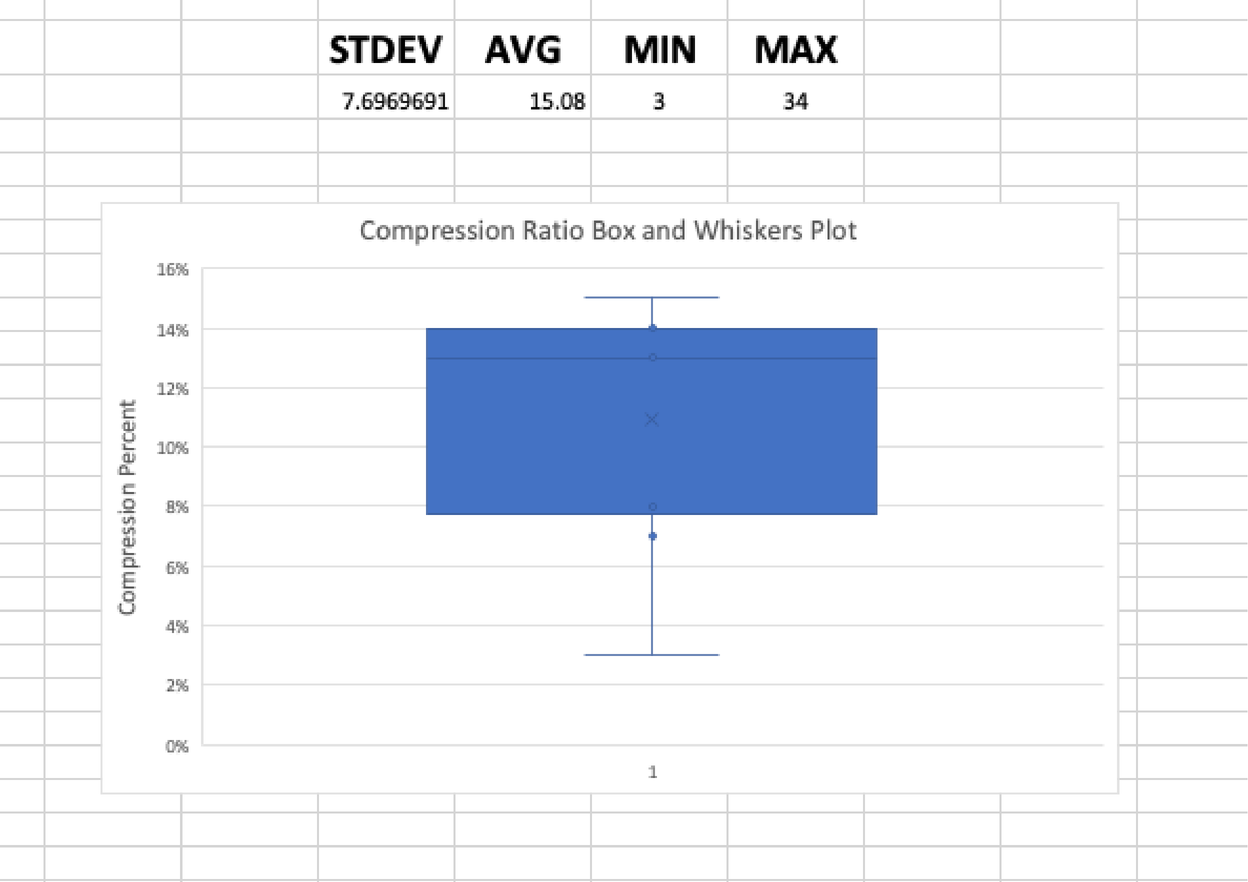
Below is a summary of the research I, Austin Spencer, did for the CSE 310 encoding/decoding project.

## COMPRESSION RATIO:

**Hypothesis for compression ratio:**
My hypothesis for the compression ratio is that multiple lines encoded at once will increase the compression ratio. I believe this will happen since with multiple lines there will inevitably be more characters per encoded line. With more characters there will also, more than likely, be more repeat characters allowing more clusters.

This first plot is of all example inputs given of compression ratio of encoding 1 line at a time.

| **Small Input** | 1 line |
|---|---|
| Haiku1 | 8% |
| Haiku2 | 8% |
| Haiku3 | 7% |
| Haiku4 | 3% |
| **Medium Input** | |
| Medium1 | 14% |
| Medium2 | 14% |
| Medium3 | 15% |
| **Large Input** | |
| Large1 | 13% |
| Large2 | 13% |
| Large3 | 14% |

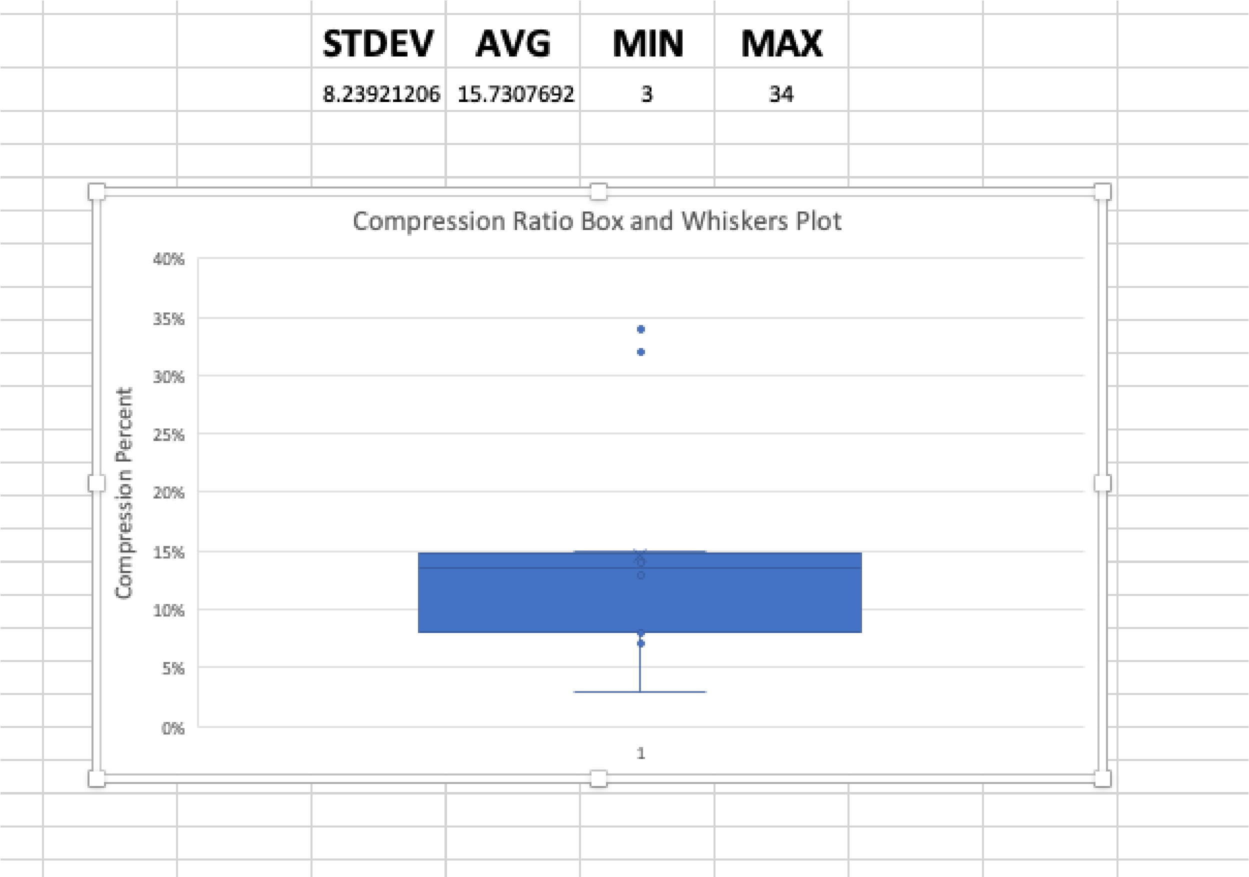


| STDEV | AVG | MIN | MAX |
|---|---|---|---|
| 7.6969691 | 15.08 | 3 | 34 |

Next plot also includes these

## Large Input

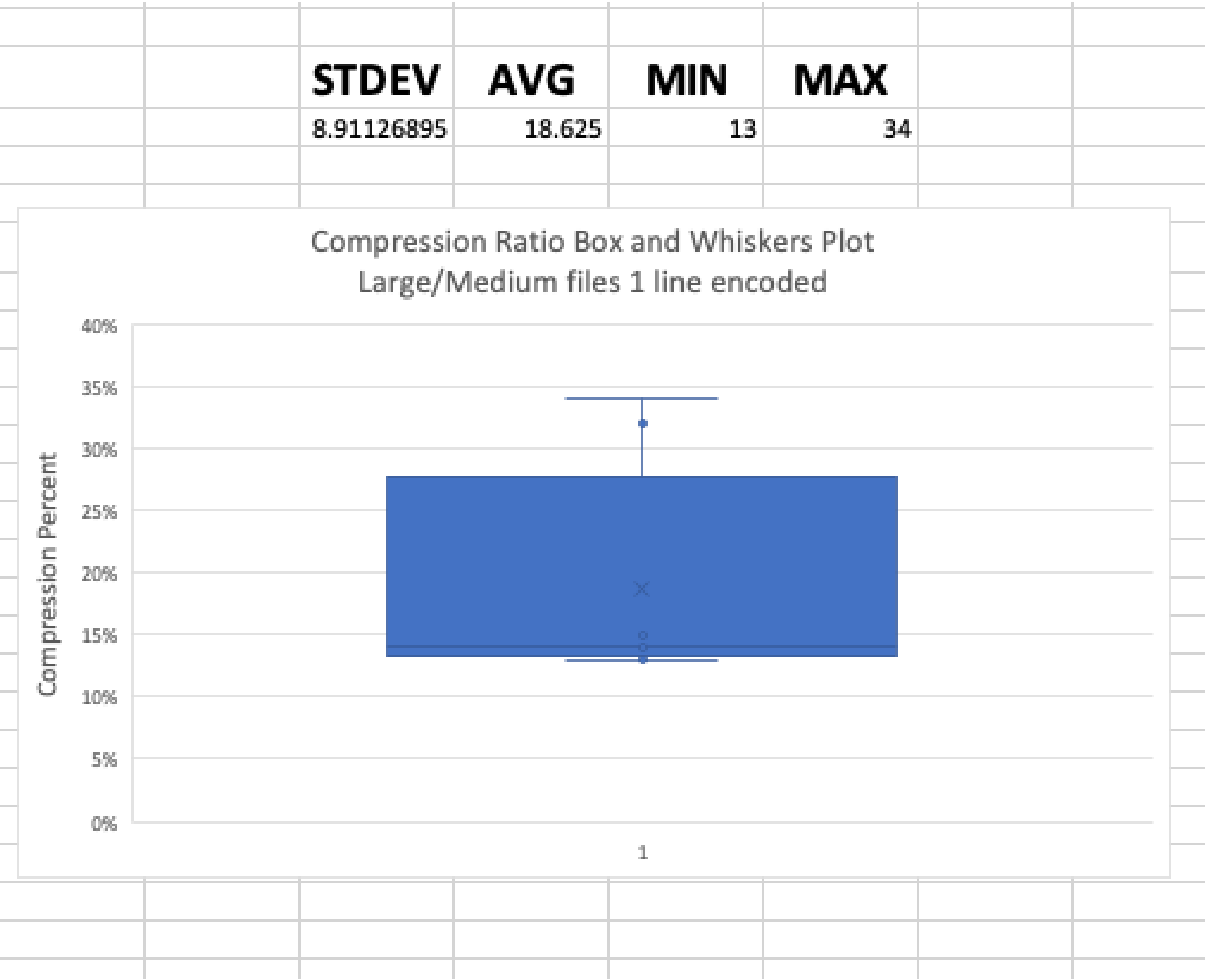
| | |
|---|---|
| Massive1 | 34% |
| Tongue Twisters | 32% |



| STDEV | AVG | MIN | MAX |
|---|---|---|---|
| 8.23921206 | 15.7307692 | 3 | 34 |

Compression Ratio encoding 1 line at a time.

| Medium Input | |
|---|---|
| Medium1 | 14% |
| Medium2 | 14% |
| Medium3 | 15% |
| **Large Input** | |
| Large1 | 13% |
| Large2 | 13% |
| Large3 | 14% |
| Massive1 | 34% |
| Tongue Twisters | 32% |

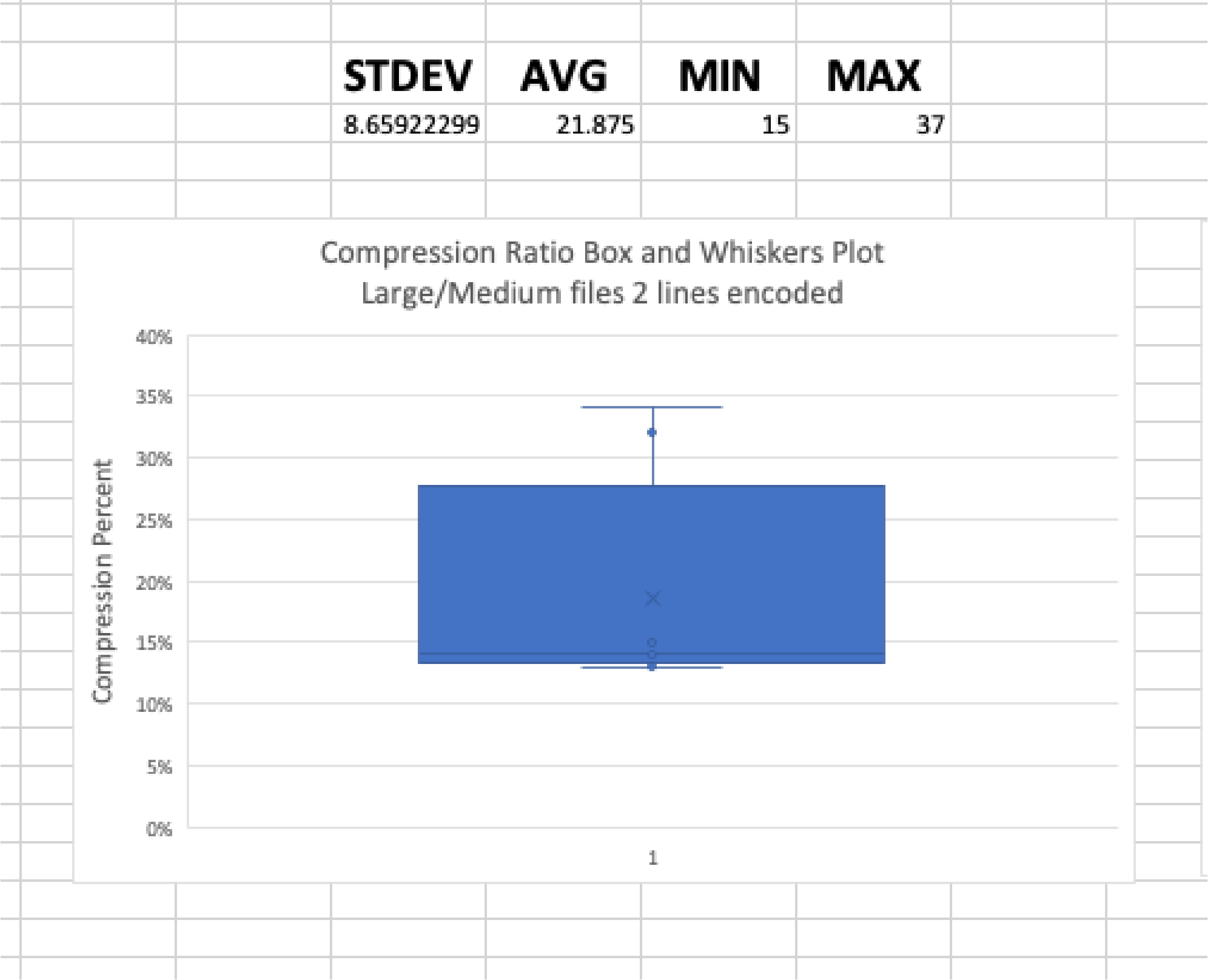


| STDEV | AVG | MIN | MAX |
|---|---|---|---|
| 8.91126895 | 18.625 | 13 | 34 |

Compression Ratio Box and Whiskers Plot
Large/Medium files 1 line encoded
Compression Percent
40%
35%
30%
25%
20%
15%
10%
5%
0%
1

Compression ratio with 2 lines encoded at once

| | |
|---|---|
| **Medium Input** | |
| Medium1 | 16% |
| Medium2 | 21% |
| Medium3 | 19% |
| **Large Input** | |
| Large1 | 15% |
| Large2 | 16% |
| Large3 | 17% |
| Massive1 | 34% |
| Tongue Twisters | 37% |

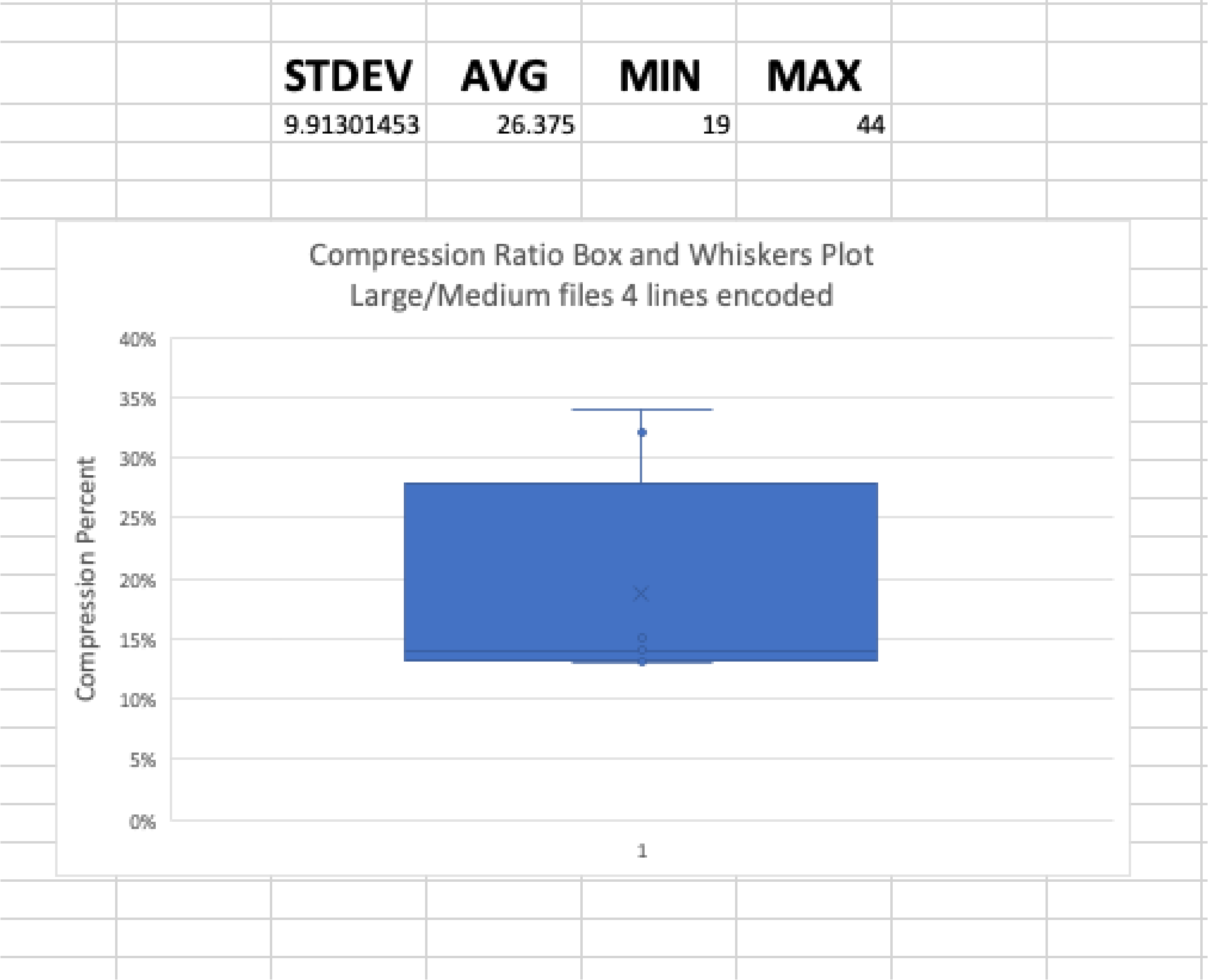


| STDEV | AVG | MIN | MAX |
|---|---|---|---|
| 8.65922299 | 21.875 | 15 | 37 |

Compression Ratio Box and Whiskers Plot
Large/Medium files 2 lines encoded
Compression Percent
40%
35%
30%
25%
20%
15%
10%
5%
0%
1

Compression Ratio 4 lines encoded at once

| | |
|---|---|
| **Medium Input** | |
| Medium1 | 19% |
| Medium2 | 25% |
| Medium3 | 23% |
| **Large Input** | |
| Large1 | 19% |
| Large2 | 20% |
| Large3 | 21% |
| Massive1 | 44% |
| Tongue Twisters | 40% |

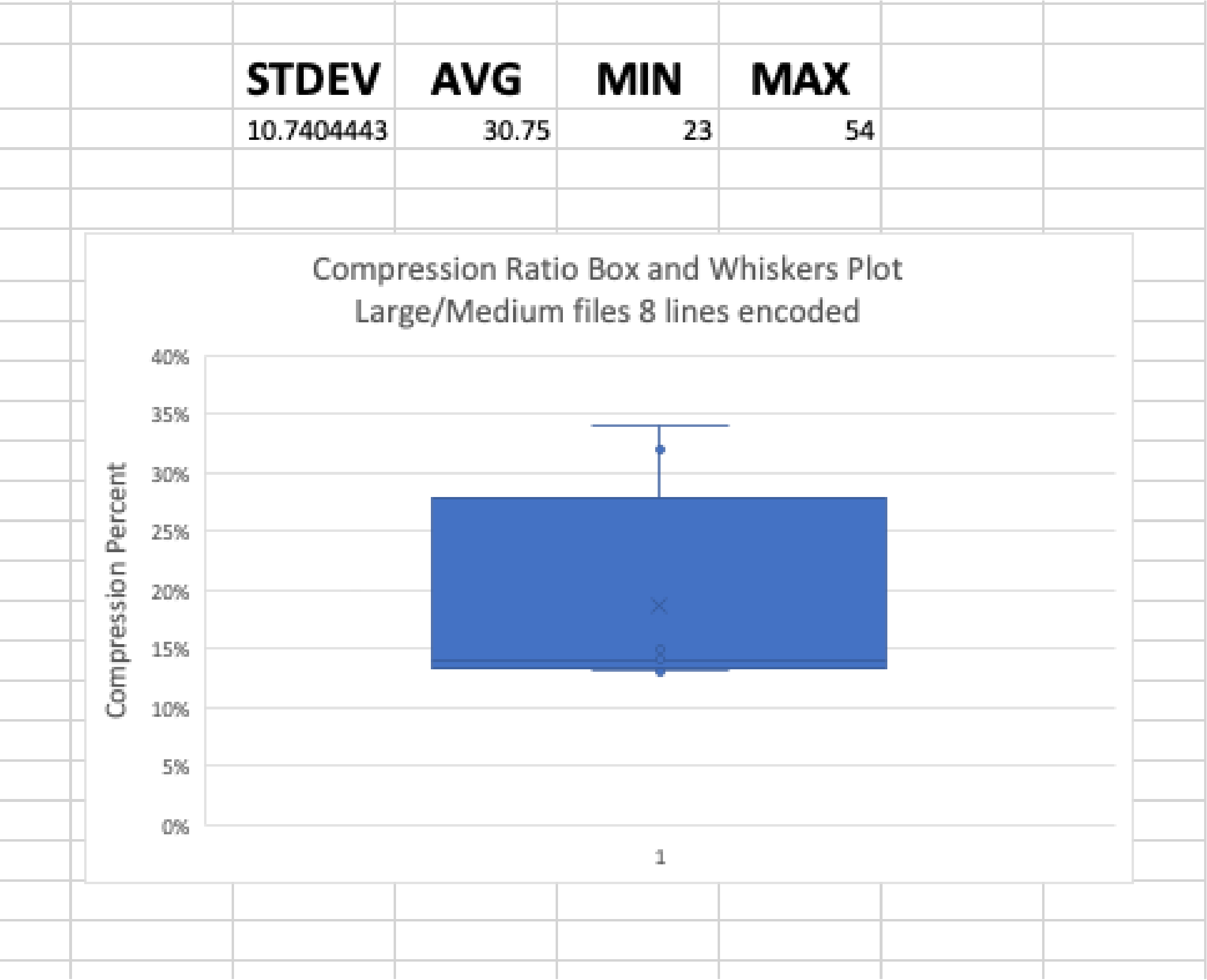


| STDEV | AVG | MIN | MAX |
|---|---|---|---|
| 9.91301453 | 26.375 | 19 | 44 |

Compression Ratio Box and Whiskers Plot
Large/Medium files 4 lines encoded
Compression Percent
40%
35%
30%
25%
20%
15%
10%
5%
0%
1

Compression Ratio 8 lines encoded at once.

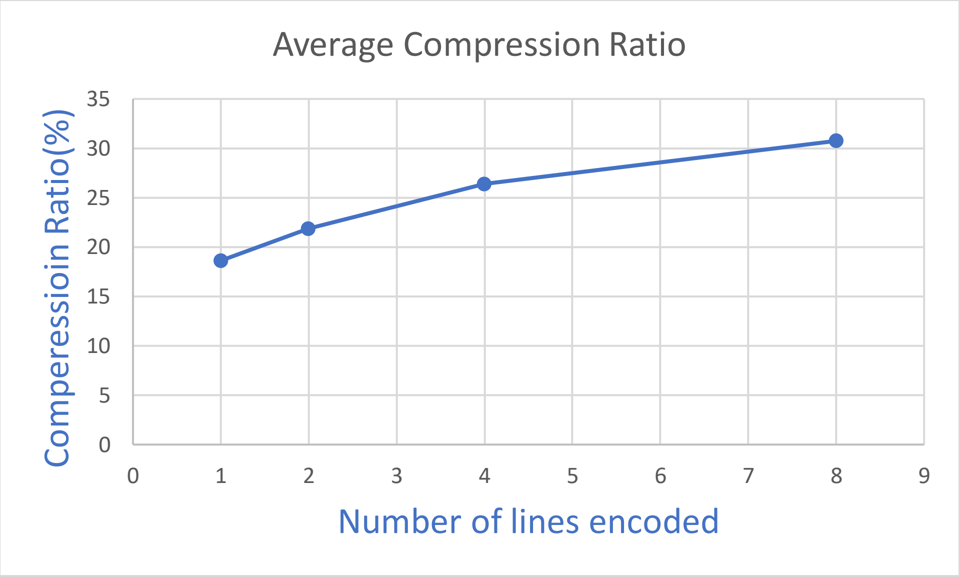
| **Medium Input** | |
|---|---|
| Medium1 | 23% |
| Medium2 | 29% |
| Medium3 | 28% |
| **Large Input** | |
| Large1 | 23% |
| Large2 | 24% |
| Large3 | 26% |
| Massive1 | 54% |
| Tongue Twisters | 39% |



| STDEV | AVG | MIN | MAX |
|---|---|---|---|
| 10.7404443 | 30.75 | 23 | 54 |

Compression Ratio Box and Whiskers Plot
Large/Medium files 8 lines encoded
Compression Percent
40%
35%
30%
25%
20%
15%
10%
5%
0%
1

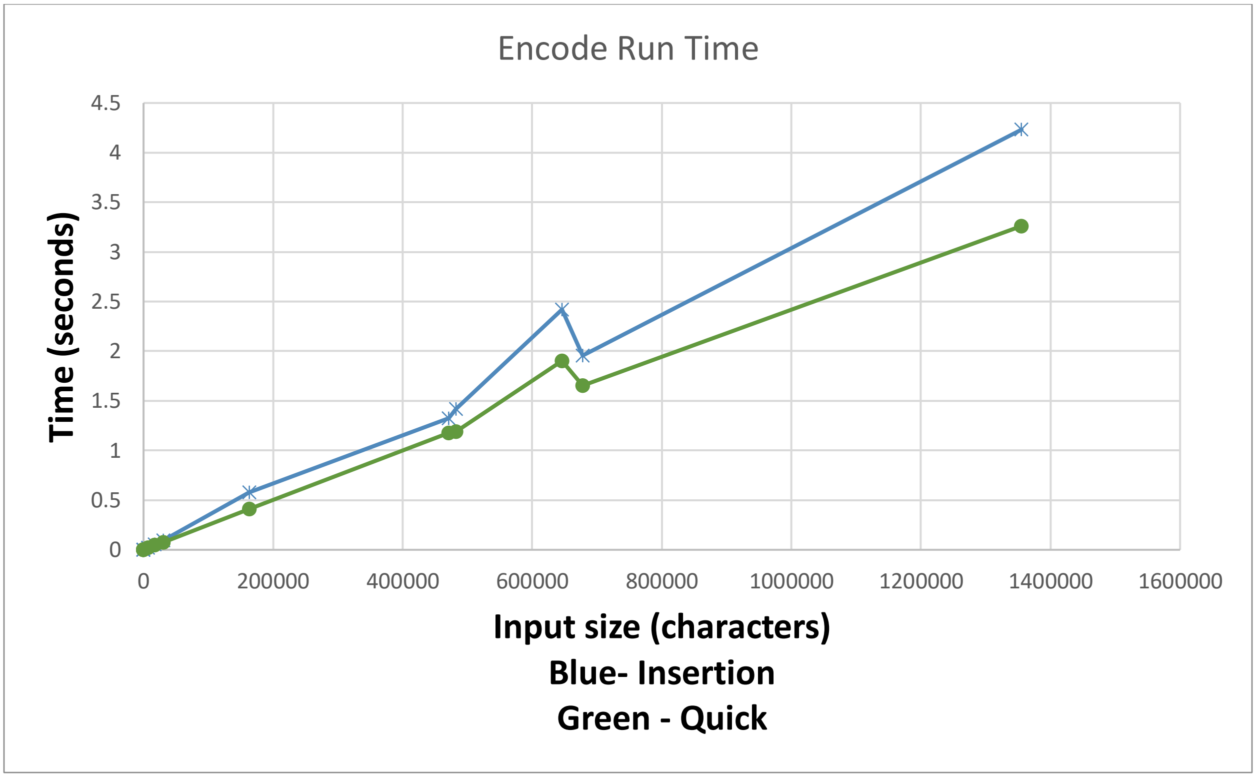
Here is a graph of the average compression ratio per line encoded.



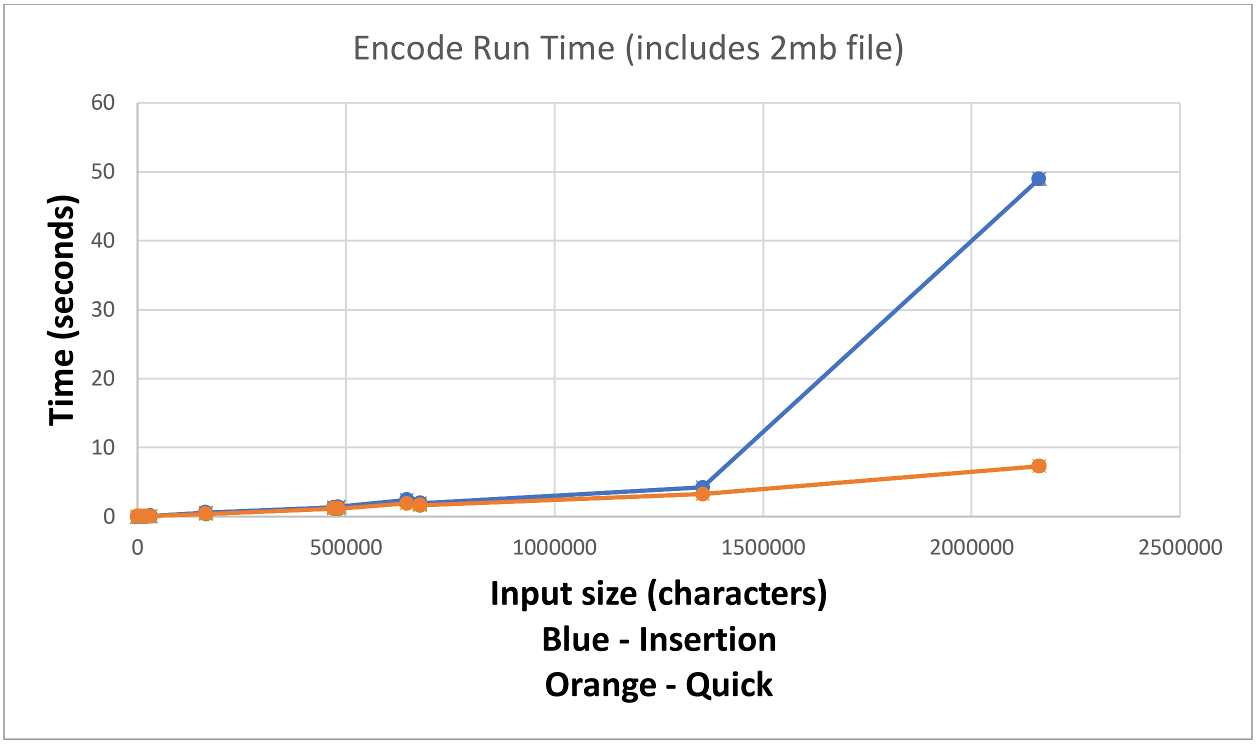
**Conclusion for compression ratio:**
I expected there to be an increase in the compression ratio with each increase in the number of lines encoded. This is clearly the case for the input I used. I do think that this change is rather significant and might be worth encoding multiple lines for this reason.

### Time to encode each input for each type of sort:

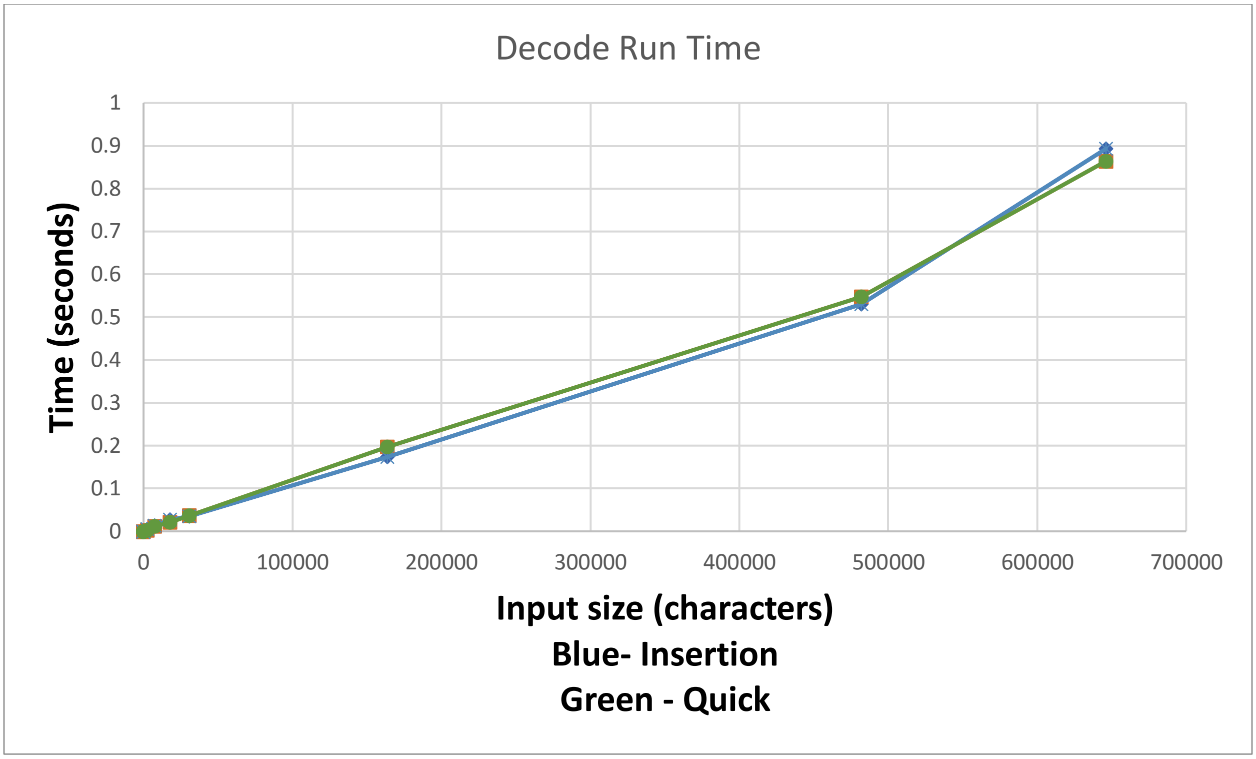
Time to encode all sample inputs using insertion/quick sort.



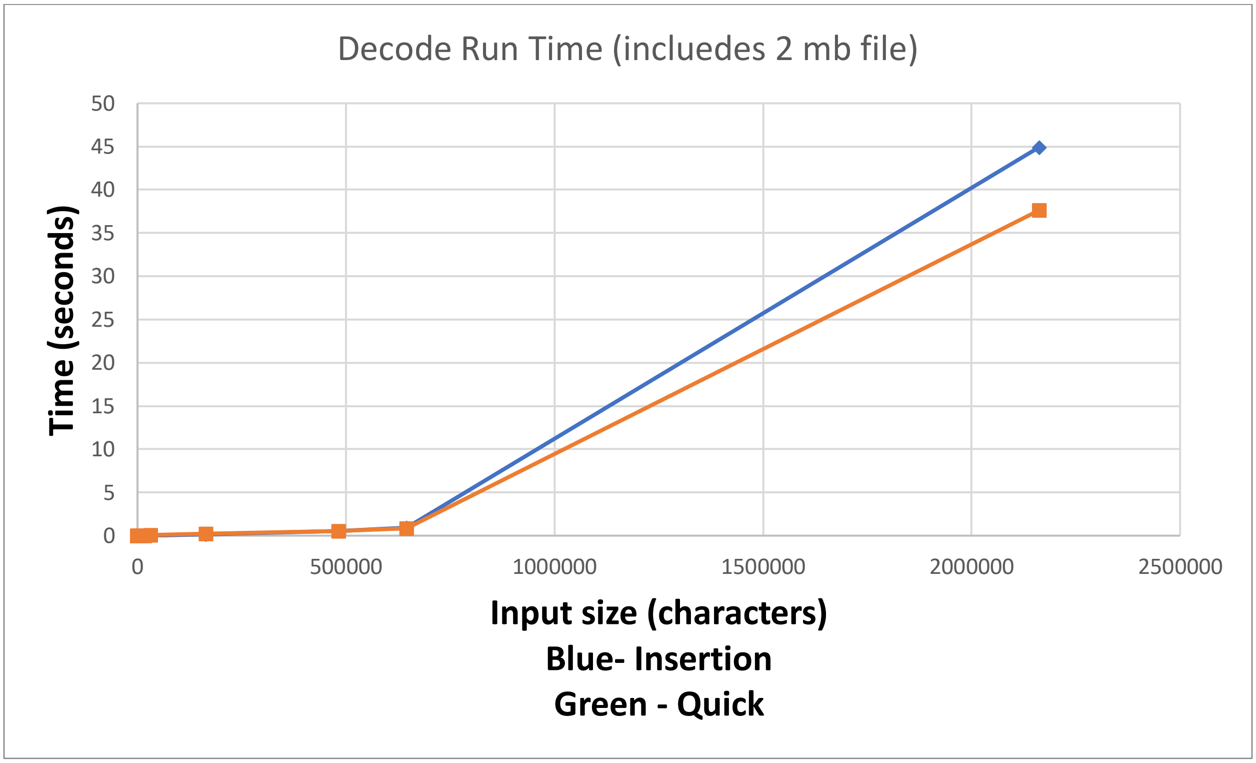
Time to encode using quick/insertion sort. However, this includes a 2mb file to show the difference quick sort can make the bigger the file is.



Now for the time to decode using quick/insertion sort. Something interesting I noticed here. Using the same quick sort method we don't see nearly the same difference running quick vs insertion sort that we do with the encode program. My guess is that this is because there are other methods in my decode function that aren't as efficient as my encode.



Here is where we can really see there isn't the same drastic drop with quick sort vs insertion sort.


Decode Run Time (incluedes 2 mb file)
Time (seconds)
Input size (characters)
Blue- Insertion
Green - Quick