<u>RESULTS FOR LinearRegressionMatrixInverse.ipynb Jupyter Notebook</u>

An attempt to determine the inverse of a matrix with Linear Regression for 3x3 invertible matrices

LinearRegressionMatrixInverse.ipynb

This notebook is basic for running initial tests to learn about how feasible it is to determine the inverse of a 3x3 matrix by linear regression. We will attempt to perform linear regression for each cell of the inverse matrix, we will try to predict the value of each matrix inverse cell. We will determine the mean of rsquareds - for the linear regression of all cells of the inverse matrix - as a metric for how good a model is.

Test 1: linear regression between matrix elements and inverse matrix elements

Test 2: linear regression with test 1 and all possible combinations of one matrix element multiplied by every other matrix element

Test 3: linear regression with test 2 and the square of each matrix element

Test 4: linear regression with test 1 and all possible combinations of one matrix element squared and multiplied by every other matrix element and the cube of each matrix element

Note how Test 4 does not even use the parts of test 3 that included all possible combinations of one matrix element multiplied by every other matrix element - these are the tests I am reporting to demonstrate the importance of the aspect of test 4

Initially I had more tests to determine what tests had the best terms for a fit I have condensed the list of tests

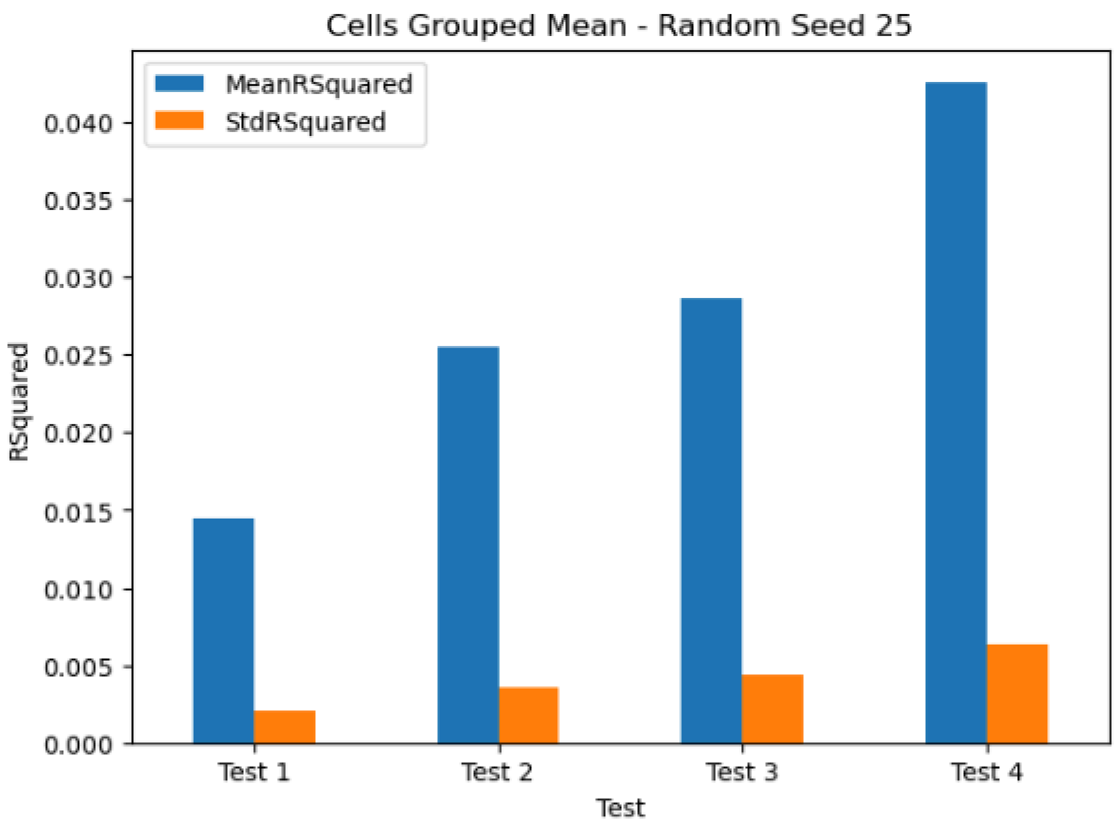
This analysis uses Ordinary Least Squares Linear Regression.

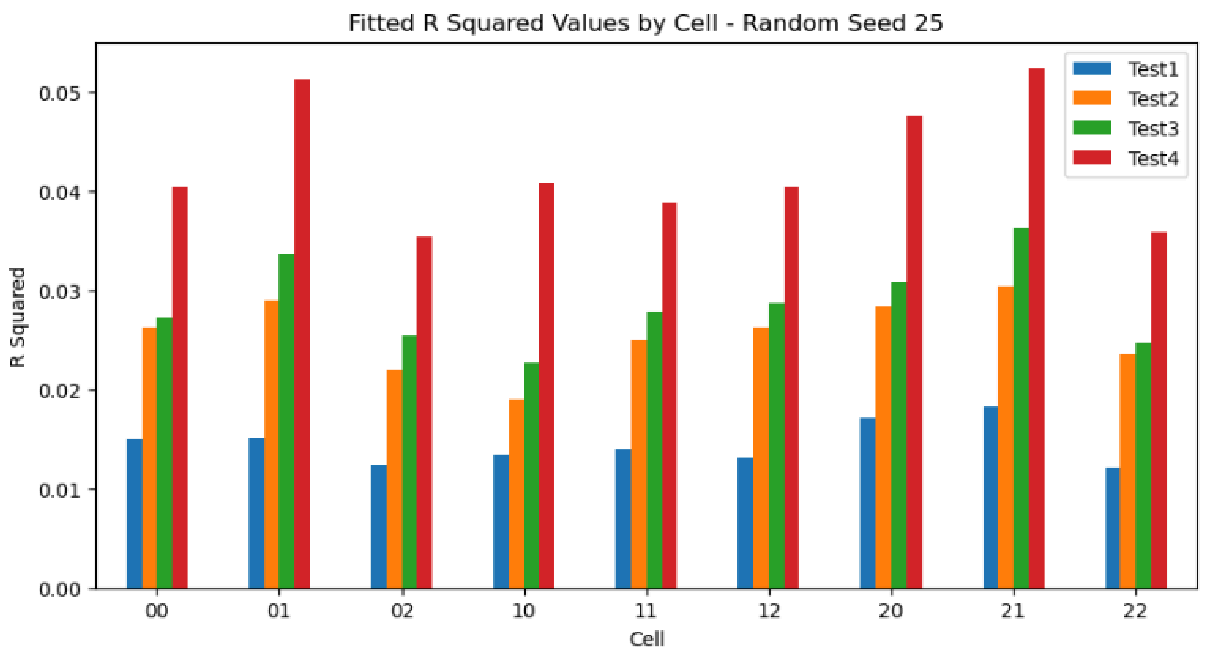
This document demonstrates the results from LinearRegressionMatrixInverse Jupyter Notebook.

It is very clear that randomly generated matrices produce different results. It would be expected that more matrices used would average out the pattern for the r squared values for the cells but the problem/solutions are highly non linear.

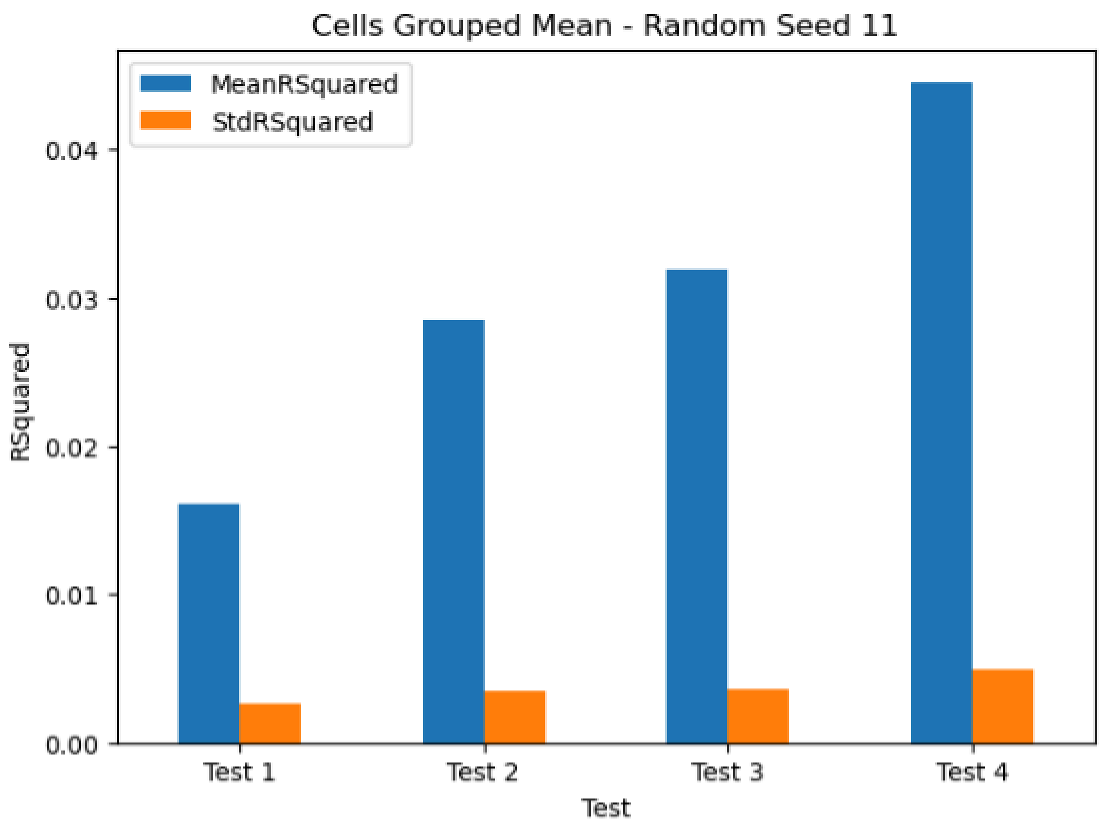
The Results will show the Cells Grouped mean results which is an average of the R Squared values for the linear regressions for all cells. As well as the a graph with the r squared values for all cells. Each set of matrices generated result is marked by Random Seed. This value can be inputted as a user defined variable at the top of the Jupyter notebook to reproduce these results.

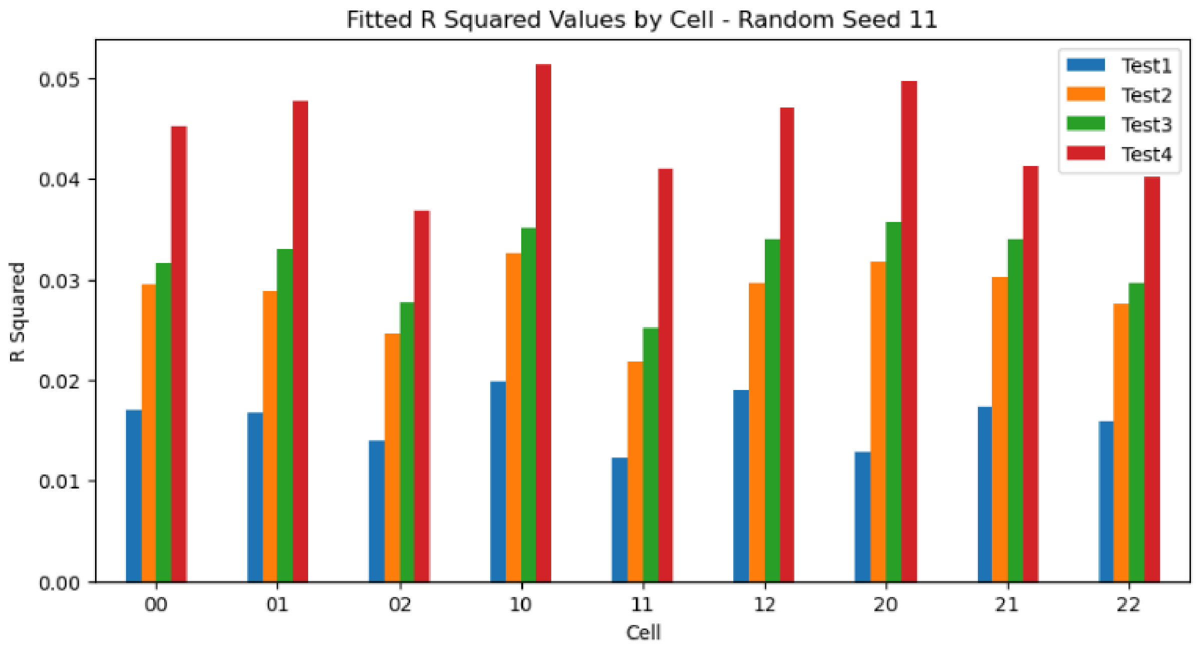
<u>RANDOM SEED 25</u>



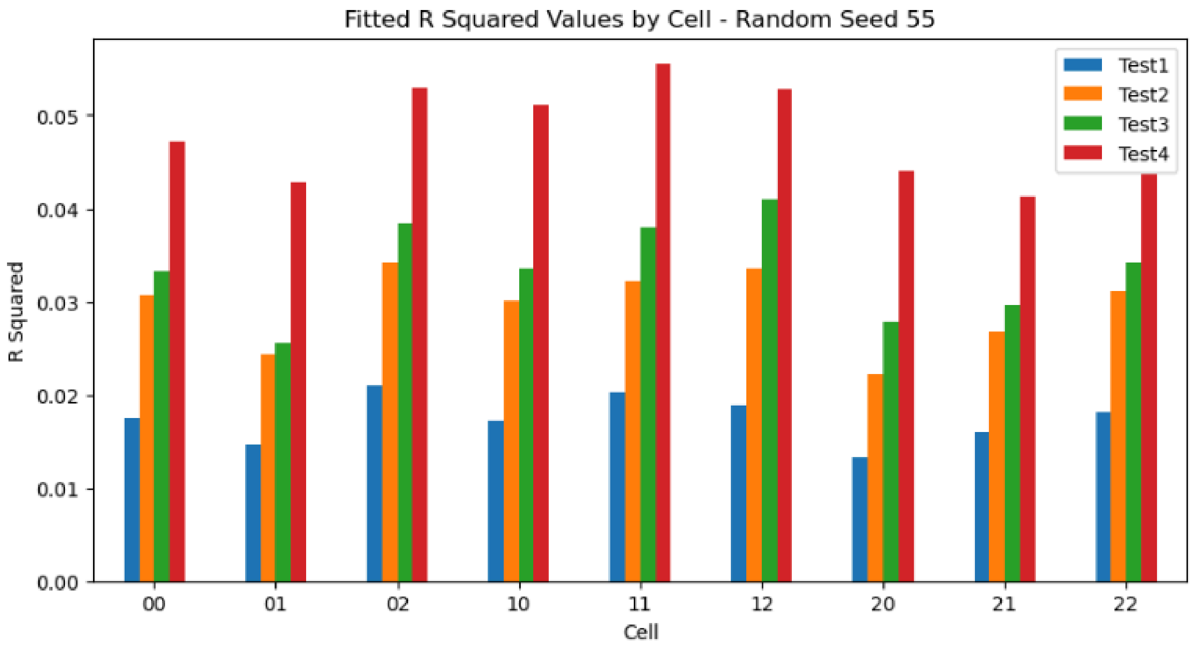


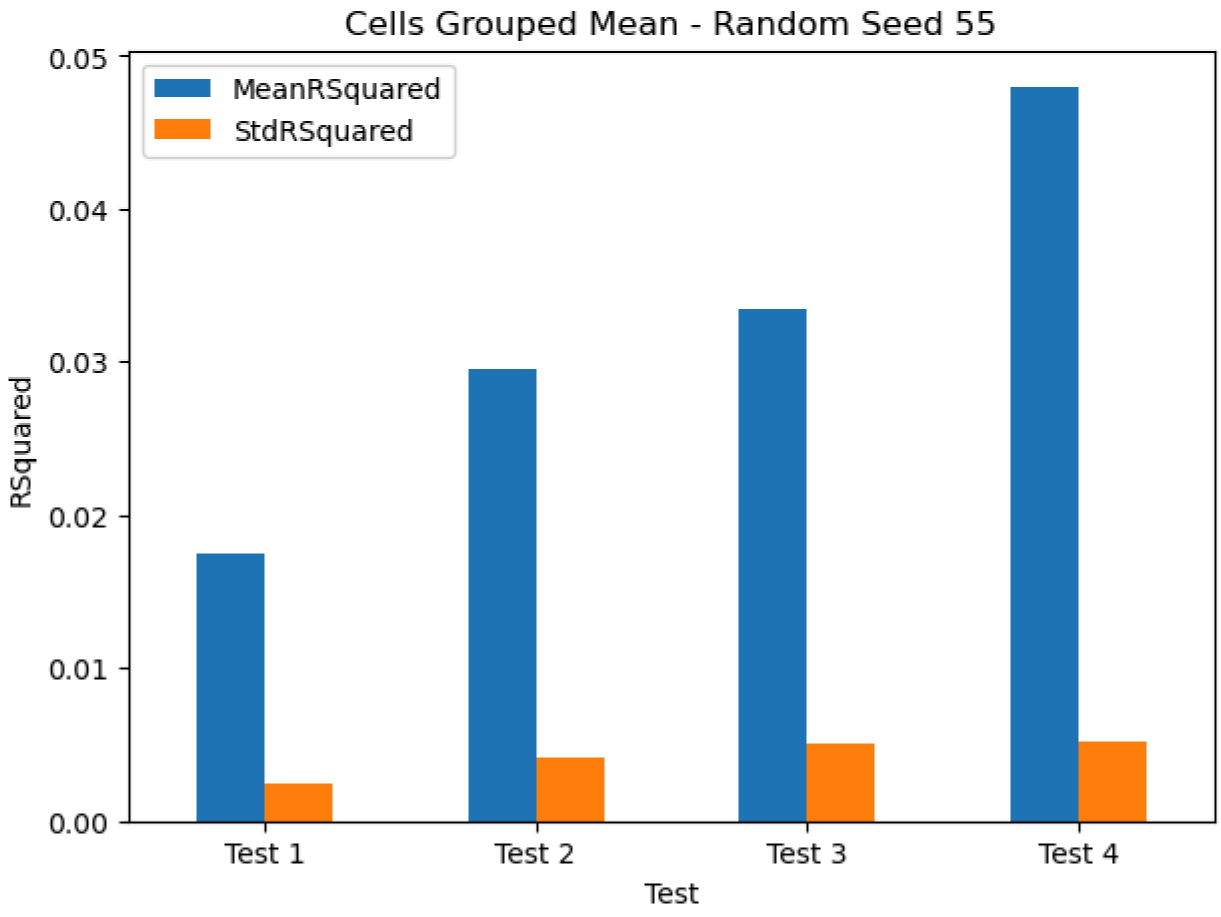
RANDOM SEED 11



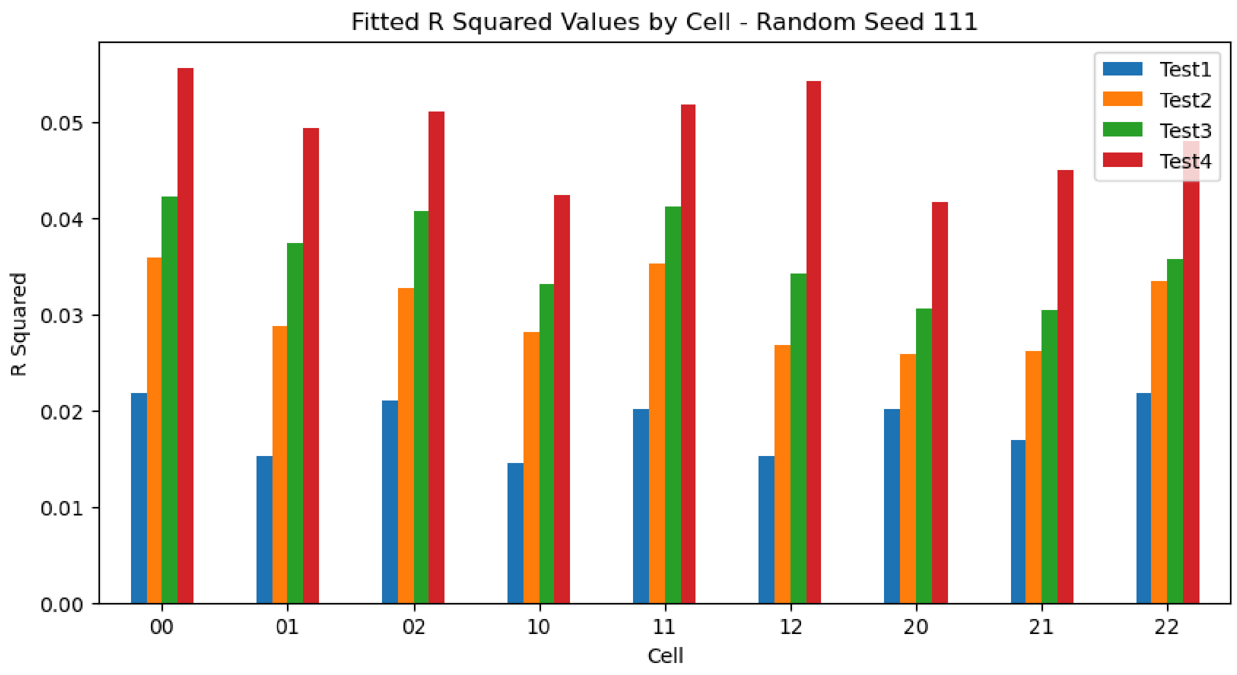


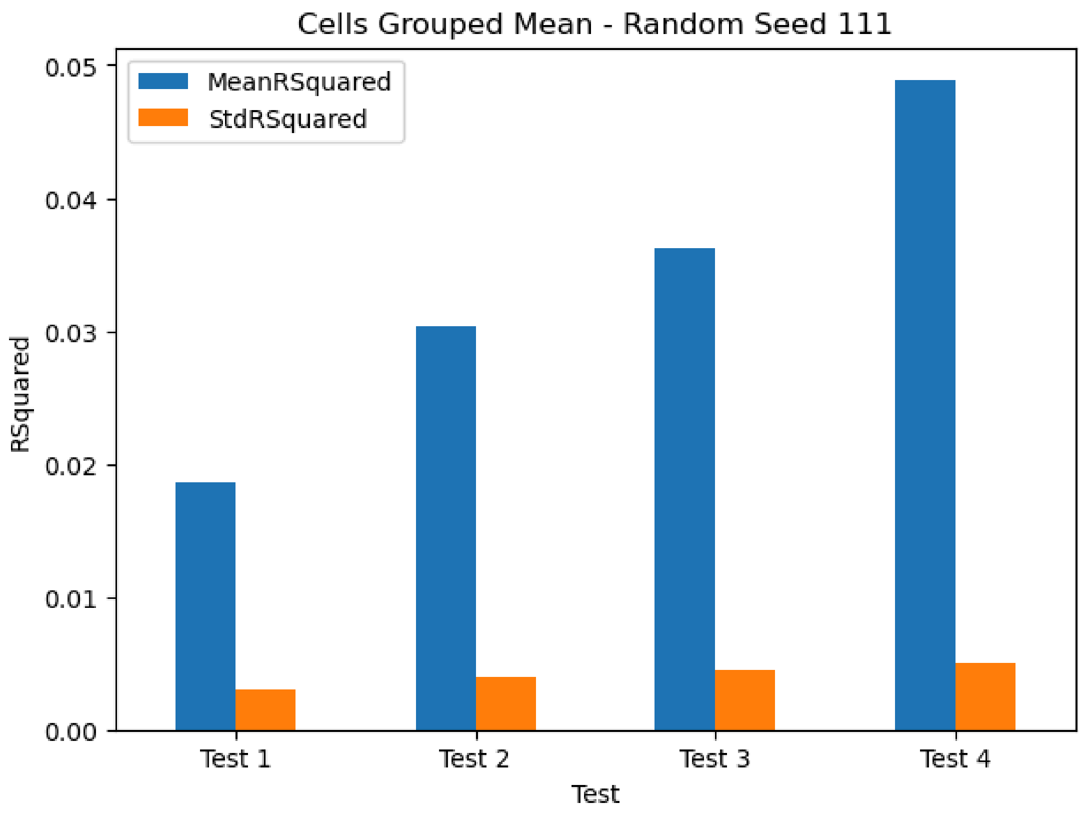
Random Seed 55



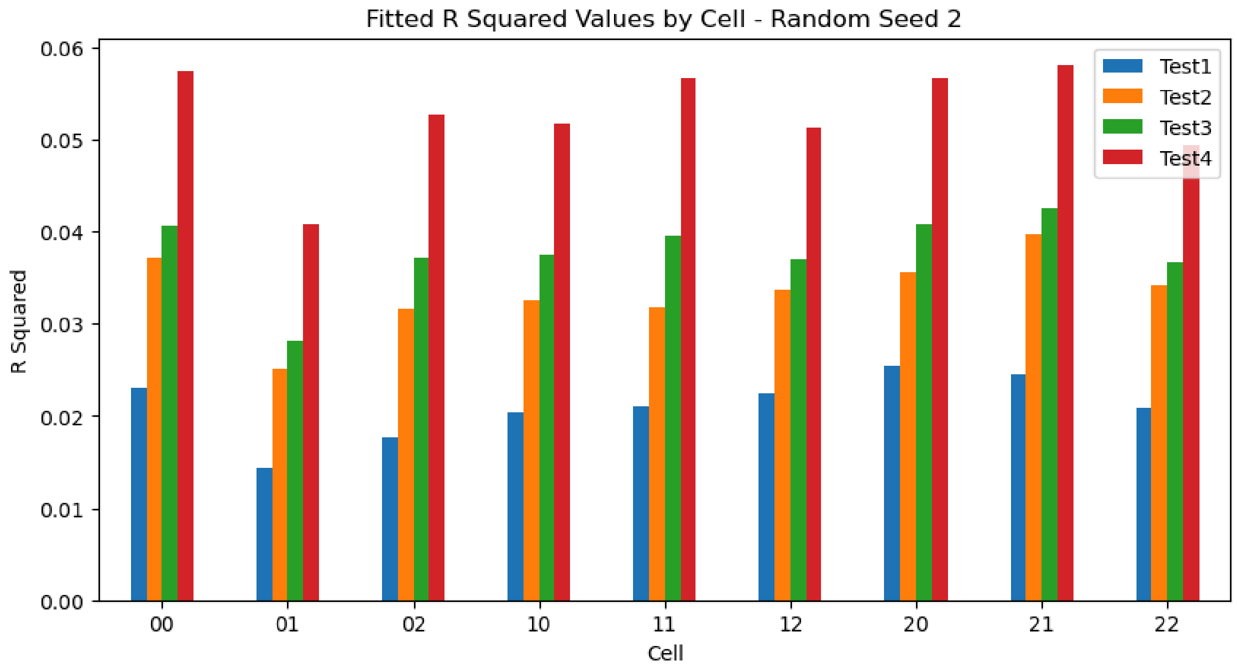


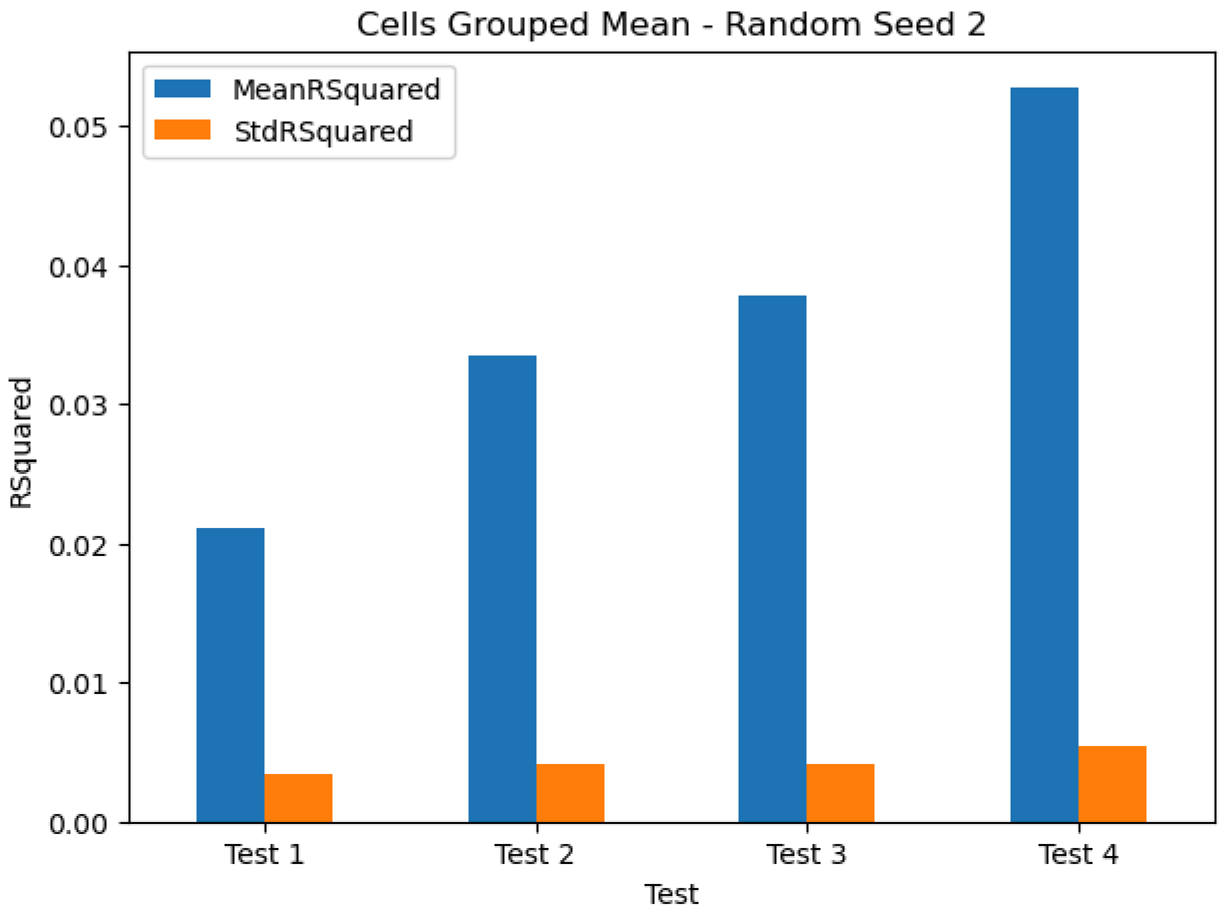
RANDOM SEED 111



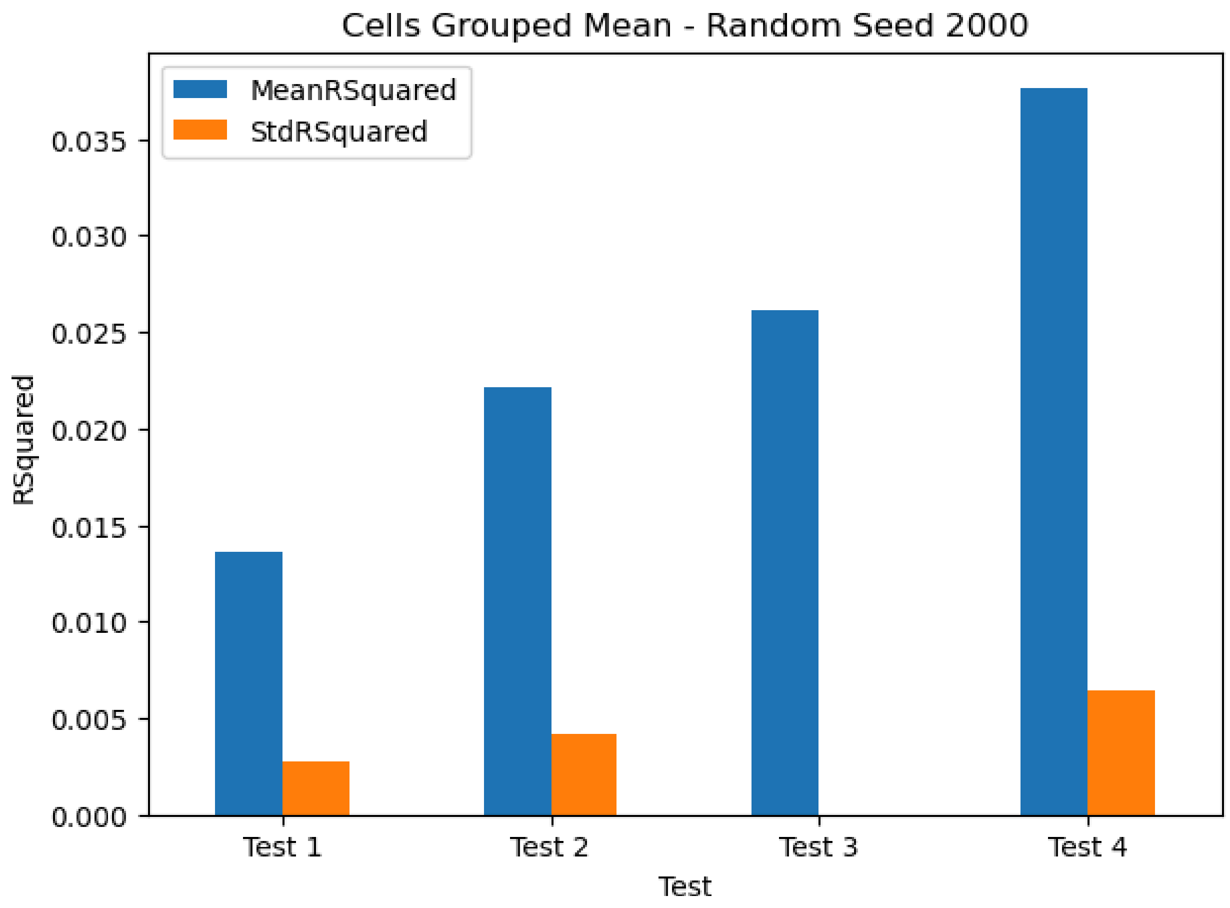


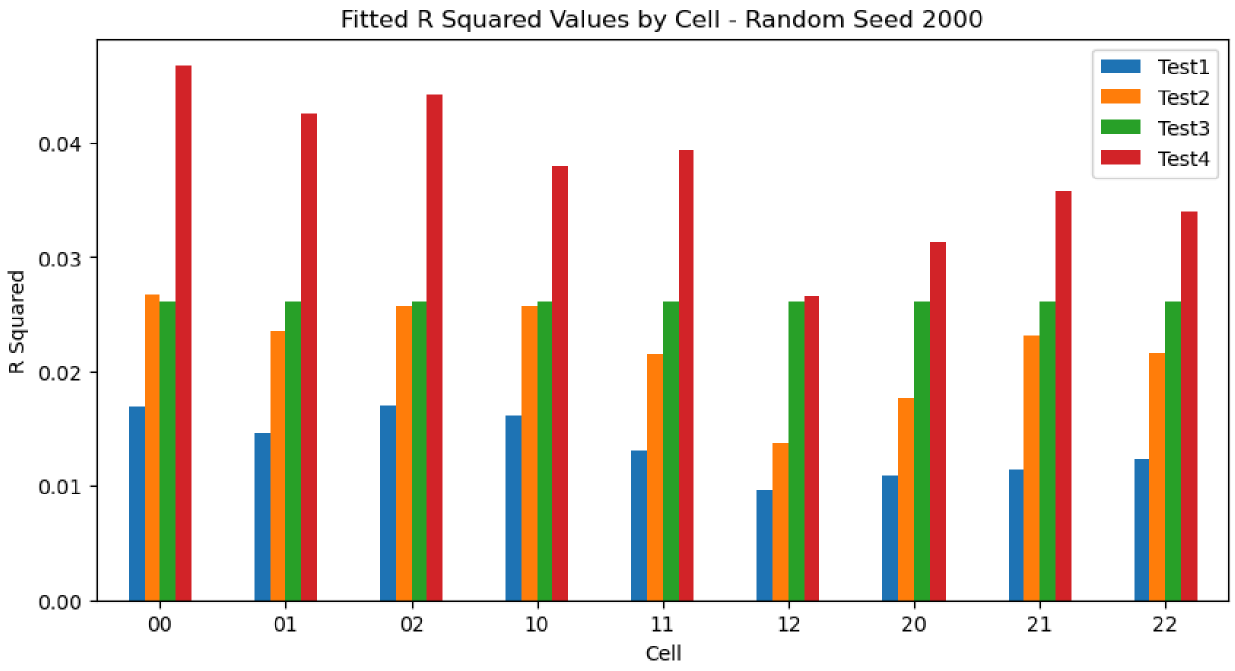
RANDOM SEED 2





RANDOM SEED 2000





Conclusion

We can see that the pattern across cells varies for each test. This indicates the highly nonlinear aspect of these results. Perhaps more matrices will level out or form a pattern in the by cell graph.

We can see that each test produces successively better r squared values. Test 4 is our best hope for a linear regression. It would be possible to add more complex terms but calculating these terms would be more computationally intensive and this is a problem as we are trying to create a faster way to compute the matrix inverse.

Next Steps: use AutomatedTesting.ipynb to determine an accurate rsquared value that we can expect for this linear regression. The terms are: every original matrix element and every element squared multiplied by every element (A cubed term is included).

Use automated testing to try to determine the r squared value for test 4 with enough tests to justify the rsquared value for each cell to be statistically different from every other cell