DotPlot3D

Manual and User Guide

Developed by Samiksha Babbar under the supervision of Dr. Jonathon Stone

Introduction

The following document serves as a summary of the usage and applications of DotPlot3D. This includes a description of the tool itself, examples of analysis, and overall next steps. DotPlot3D is a tool that expands on existing dot plot methodology and creates an output for three sequences in an interactive, user-friendly interface. In comparison to other methods, this allows for expansion beyond pairwise alignment, provides different colour and sizing for the generated points to reduce noise, and easily identifies regions of interest and synteny between sequences. Rapid visualisation with an existing set of pre-inputted sequences, or FASTA sequences that the user inputs is implemented into the software. A working version can be downloaded and accessed through:

https://github.com/samibabbar/DotPlot3D

There are two main tools provided within the working version:

1. **AnalysisTable3D**: An additional analysis tool useful for determining regions of interest and using along with the main dot plot
2. **DotPlot3D**: An novel, expanded version of original methodology of dot plots proposed by Gibbs and McIntyre that provides 3D analysis of nucleotide sequences

These were created on Wolfram Mathematica v.14, a symbolic language that uses original algorithms from Wolfram Research and integrates them into an easy-to-use platform for data visualisation.

AnalysisTable3D: For Conducting Supplementary Analyses

During the development of DotPlot3D, there was one clear issue for the limits in analysis - determining whether to plot the entire sequence or specific segments. When looking at a randomly generated set of sequences, there is a specified default length of 20 bp and has no issues in generating the plot. However when inputting custom FASTA sequences and determining similarities beyond approximately 100 bp, the tool becomes difficult to interpret and needs specifications of specific regions. Therefore, AnalysisTable3D acts as a supplementary tool that specifies and identifies regions of interest prior to the plotting tool. For example, Figure 1 depicts an analysis of the first 15 positions of three influenza A sequences by region in Canada.

Position	sq1	sq2	sq3	Seq 1 + Seq 2	Seq 2 + Seq 3	Seq 1 + Seq 3	All three
1	A	T	A	False	False	True	False
2	T	A	T	False	False	True	False
3	G	C	G	False	False	True	False
4	A	A	A	True	True	True	True
5	A	T	A	False	False	True	False
6	G	T	G	False	False	True	False
7	A	T	G	False	False	False	False
8	C	A	C	False	False	True	False
9	T	C	A	False	False	False	False
10	A	A	A	True	True	True	True
11	T	A	T	False	False	True	False
12	C	C	A	True	False	False	False
13	A	C	C	False	True	False	False
14	T	G	T	False	False	True	False
15	T	C	A	False	False	False	False

Seq1	>MN538623.1 Influenza A virus (A/Ontario/244/2018(H3N2)) segment 4 hemagglutinin (HA) gene, complete cds
Seq2	>MN538799.1 Influenza A virus (A/British Columbia/080/2018(H1N1)) segment 4 hemagglutinin (HA) gene, partial cds
Seq3	>MN538830.1 Influenza A virus (A/Quebec/197/2018(H1N1)) segment 4 hemagglutinin (HA) gene, complete cds

Figure 1: Example output of AnalysisTable3D, showing similarities at each position of three different Influenza A strains. Details of each sequence can be seen below the figure, representing samples collected in different regions of Canada in 2018. The first 15 positions are shown out of the total, identifying regions of all three similarities.

With identifications of regions that have similarity within the sequence, users are able to easily spot specific ranges using the green highlights, looking further into specific regions. In the example of analysing strains of influenza based on geographical location, it is clear that sequences 1 and 3 (Canada and Quebec) have a lot more geographical similarities than comparisons with British Columbia. In this case, the user may want to alter their analysis and create a dot plot with sequences collected in closer proximity together. This shows that the tool is a useful supplement to analysis with DotPlot3D and can act as a tool for initial comparison. Users can easily identify similarities and decide which sequences to choose for analysis.

DotPlot3D: For Nucleotide Sequence Analysis

After identifying sequences and regions of interest, the plotting is complete using the DotPlot3D tool. Using varying colour options, similarities are shown between sequence 1 and 2, 2 and 3, 1 and 3, and all three sequences. The plot is a straight, 3D line that differs based on the data provided. The following table summarises how dot colour is determined:

Table 1: How colour corresponds to similarity in DotPlot3D

Colour	Similarity Shown
White	Similarities between all three
Red	Sequence 1 and 2 similarities
Green	Sequence 1 and 3 similarities
Blue	Sequence 2 and 3 similarities

Similarly, another feature is varying point size based on the presence of transitions and transversions. The current working version will identify these mutations based on the first sequence, and whether there is deviation in both sequence 2 and 3. The following table shows the type of change and whether there is a site of prominent mutation:

Table 2: Detecting the presence of transitions and transversions using DotPlot3D

Point Size	Similarity Shown
0.01	Null point size
0.02	Transition
0.03	Transversion

With these attributes in mind, the next pages will walk through the methodology using a breakdown of the code and how to generate the final plot. This includes generating or inputing the sequences, the process of testing, how similarities are identified, and the output

USING DOTPLOT3D

A Visual Guide

```
Clear[sequenceGenerator];
sequenceGenerator[n_] := RandomChoice[{a, c, g, t}, n]
sm1 = sequenceGenerator[20]
sm2 = sequenceGenerator[20]
sm3 = sequenceGenerator[20]
```

RANDOM SEQUENCE GENERATOR

The DotPlot3D tool is used on a set of 3 nucleotide sequences generated at length 20. This is done using the sequenceGenerator function, which uses a random choice between A,T,G, or C at a specified length, n. This function is called to three variables: sm1, sm2, sm3.

```
Clear[PositionTest];
PositionTest[{x1_, x2_, x3_}] := MapThread[{Boole@SameQ[#1, #2],
      Boole@SameQ[#1, #3],
      Boole@SameQ[#2, #3]} &, {x1, x2, x3}];

tfsm = PositionTest[{sm1, sm2, sm3}]
```

TESTING AT EACH POSITION

PositionTest takes 3 positions in a list and compared based on its similarities. The “Boole@” changes them from true or false statements, and returns similarity. This is completed with the variable tfsm. which is a position test between sm1, sm2, and sm3. The variable is called and parted later on.

{1,0,0} ->. Sequence 1 and 2 are the same
{0,1,0} -> Sequence 1 and 3 are the same
{0,0,1} -> Sequence 2 and 3 are the same
{1,1,1} -> All three are the same
{0,0,0} -> None are similar

```
ppt = Transpose[{sm1, sm2, sm3}]
customPointSize[list_List] := Module[{size}, size[{A, G, G}] := PointSize[0.05];
  size[{g, a, a}] := PointSize[0.02];
  size[{a, g, g}] := PointSize[0.02];
  size[{t, c, c}] := PointSize[0.02];
  size[{c, t, t}] := PointSize[0.02];
  size[{t, a, a}] := PointSize[0.03];
  size[{a, t, t}] := PointSize[0.03];
  size[{g, c, c}] := PointSize[0.03];
  size[{c, g, g}] := PointSize[0.03];
  size[_] := PointSize[0.01];
  size /@ list]
```

SETTING POINT SIZE

The function customPointSize assigns the module “size” based on size 0.02 for transitions, 0.03 for transversions, and a default of point size 0.01. This transforms a list of the transposed sequences into point sizes based on what they are labelled as.

```
colourSpecify[list_List] := Module[{size},
  size[{1, 0, 0}] := RGBColor[1, 0, 0];
  size[{0, 1, 0}] := RGBColor[0, 1, 0];
  size[{0, 0, 1}] := RGBColor[0, 0, 1];
  size[{0, 0, 0}] := RGBColor[1, 1, 1];
  size[{1, 1, 1}] := RGBColor[0, 0, 0];
  size /@ list]
colourSpecify[tfsm]
```

SPECIFYING COLOUR

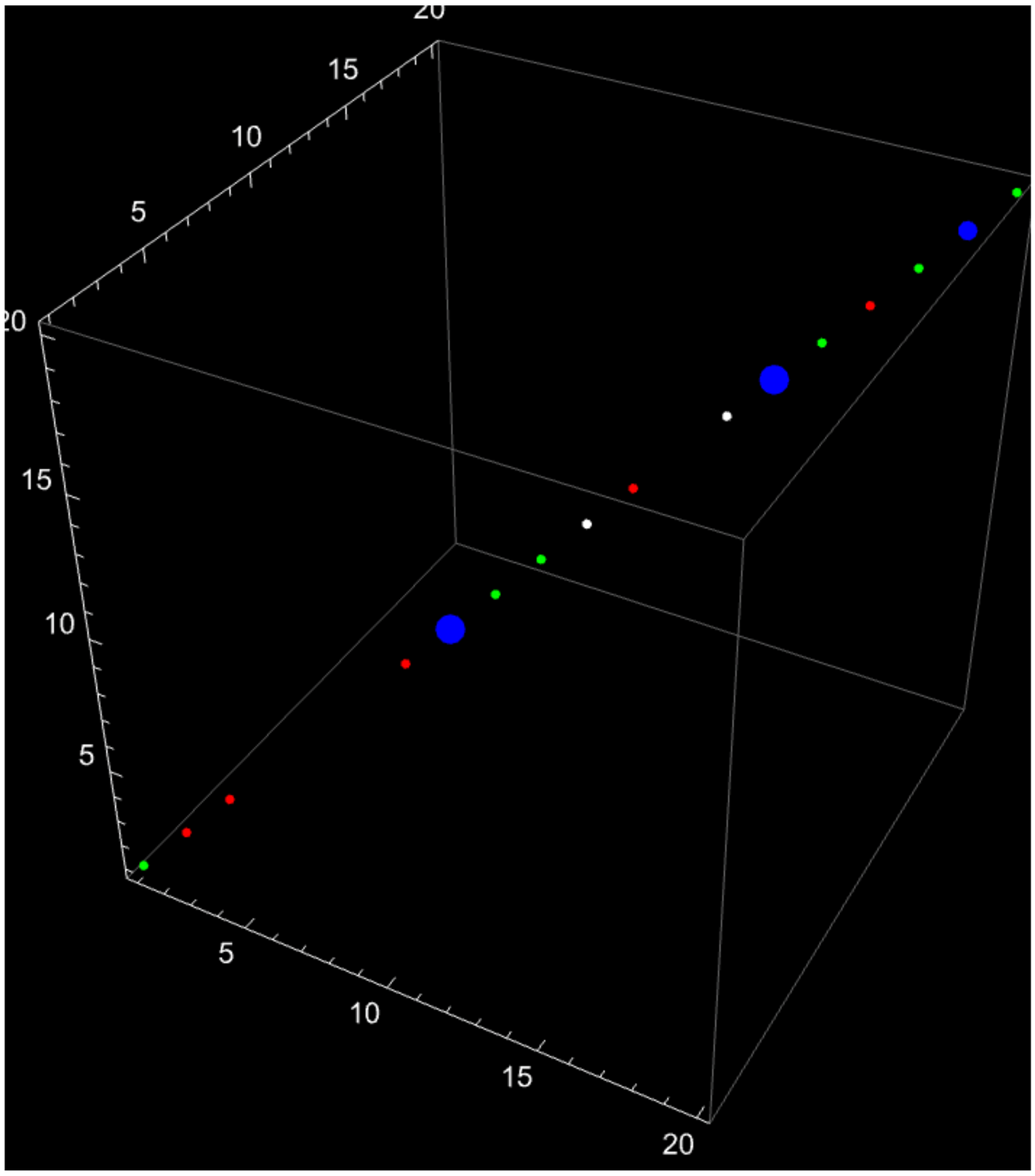
Assigns colours based on what similarity is at each position. Referring back to the PositionTest function for example, if all three sequences are the same, it assigns an RGB colour to that position. Note that RGBColor{0,0,0} can be switched around based on whether the plot is on a black or white background.

```
finalpoint =
      Table[
       {
        Part[customPointSize[ppt], i],
        Part[colourSpecify[tfsm], i],
        Point[{i, i, i}]
        },
       {i, 1, Length[tfsm]}
      ]
```

CREATING PLOTTING POINT

The final point consists of all the steps mentioned before. It parts customPointSize and colourSpecify parted at every position, as well as the 3D point for that position. The table is specified for the length of the sequence, given by the variable tfsm.

THE FINAL PLOT



GENERATED BASED ON THE FOLLOWING REFERENCE SEQUENCES:

1) {c, g, a, t, g, a, c, g, g, c, t, a, a, a, g, g, a, t, g, t}

2) {t, g, a, g, t, g, c, c, t, a, t, a, g, a, c, c, a, c, a, a}

3) {c, t, g, c, c, t, g, c, g, c, t, t, c, a, c, g, t, t, a, t}