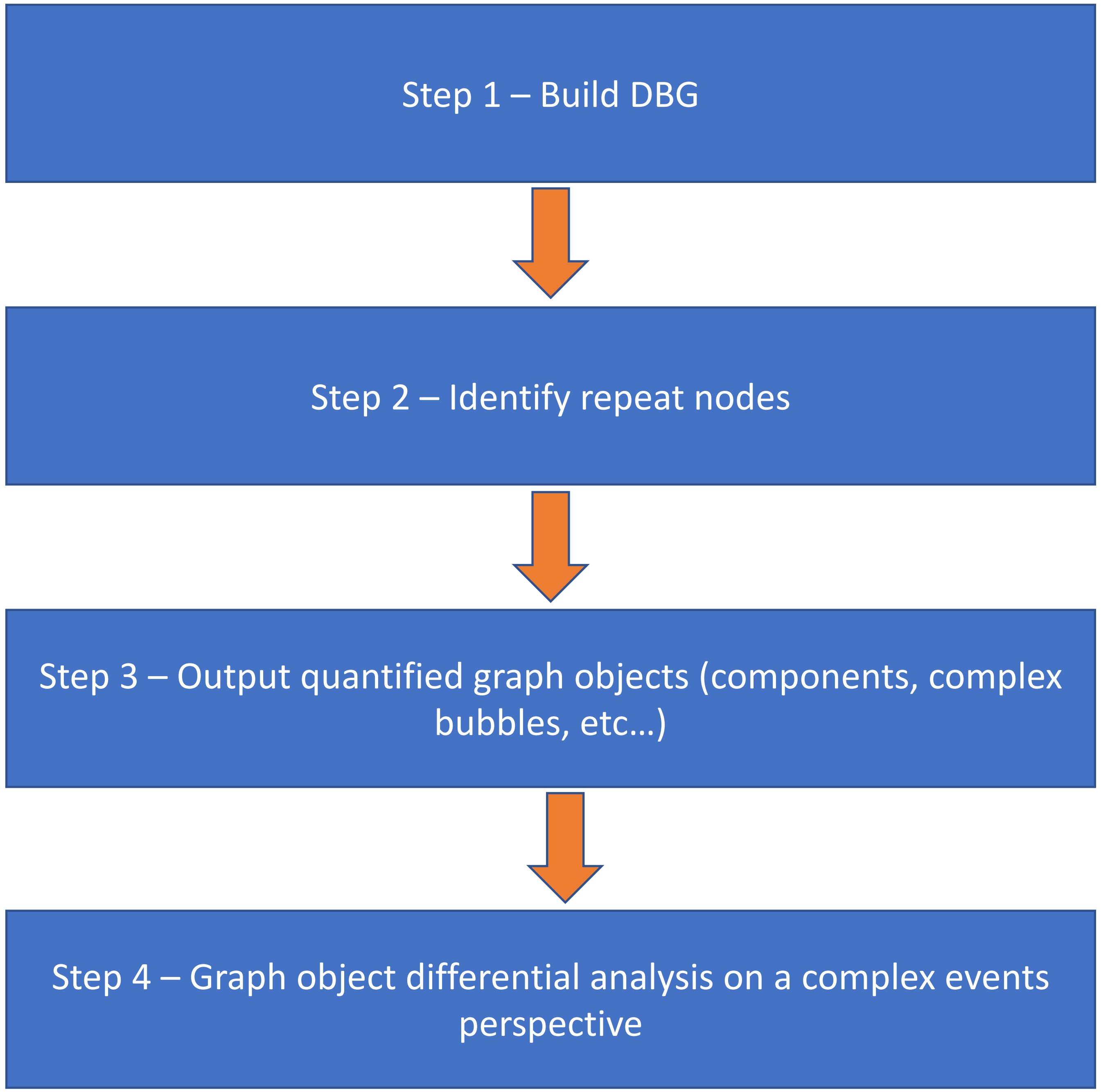
Step 1 – Build DBG
Step 2 – Identify repeat nodes
Step 3 – Output quantified graph objects (components, complex bubbles, etc…)
Step 4 – Graph object differential analysis on a complex events perspective

Step 1 – Build DBG

1/ DBG construction;
2/ Get (k+1)-mers (edges) counts per read file;

Step 2 – Identify repeat nodes

-Currently done reference-based;

-Identifies the unitigs that are due to **genomic** repeats;
    -Not ideal, we want to find unitigs corresponding to transcriptomic repeats;
    -Or even further: **repeats that are problematic to the sequenced transcriptome only**;

-Pipeline:
    1/ Align unitigs to the genome using STAR;
    2/ Intersect the .bam with UCSC's repeat masker track;
    3/ Unitigs having overlaps with repeats of at least 20 bases (by default) are identified as repeat nodes;
    4/ Unitigs mapped to too many loci and unmapped unitigs are also identified as repeat nodes;

-TODO:
    -Do this in a de-novo way;
    -Use machine learning with several features including beta-value, branching concentration, coverage, sequence entropy (what else?) to help on classifying if a node is repeat-induced or not;
    -Features extraction (for the current ones) is almost done;

Step 3 – Output quantified graph objects (components, complex bubbles, etc…)

1/ Output the full repeat-free graph (nodes and quantified edges);

2/ Print all the uncompressed components of the graph (nodes and quantified edges);

- This was based on the hypothesis that removing repeat-induced untigis would disconnect the graph into several components, one for each gene, or gene family;

3/ Compress the components of the graph in a sequence-based way;

- Pretty bad algorithm for now:
    1/ List all paths from any source to any target in the component;
    2/ Transform all paths in sequences;
    3/ Cluster the sequences (sequences with edit distance <1% are clustered together);
    4/ For each cluster, get the most k-mer covered sequence as representative;
    5/ Build the DBG of the representative sequences;
    6/ Output the compressed components;
    - Edge requantification not yet done;
    - Bad algorithm because in some components with some hundreds nodes, we can have millions paths and this would take too, too much time…
    - Fall back to simpler methods like 4 nodes compression? Iterative approach? Is this really needed?

4/ We used to also find and output complex bubbles in each component... But not anymore, I don't remember why...

Step 4 – Graph object differential analysis on a complex events perspective

???