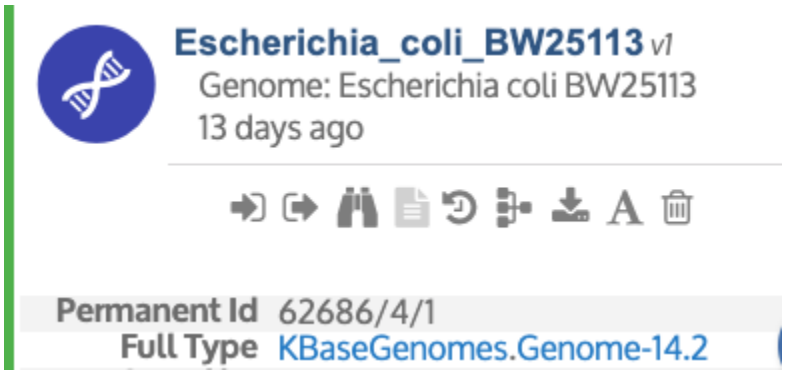
RBTnSeq KBase Apps How-To

The RBTnSeq process is divided into two parts, 'TnSeq' and 'BarSeq'. Use the TnSeq manual if you are running the programs for the mapping step, after which you will have a TnSeq library. Read the BarSeq manual if you already have a TnSeq library and are ready to analyze gene fitness in various conditions. The main TnSeq apps are called "RBTnSeq Maps To Pool" and "RBTnSeq MapTables", with the auxiliary app "RBTnSeq Find TnSeq Model". The main BarSeq apps are called "RBTnSeq Reads to Pool Counts" and "RBTnSeq BarSeqR" with auxiliary apps "RBTnSeq Import Files (TSV) from Staging Area" and "RBTnSeq Download Tables".

TnSeq (Part 1/2)

Within RBTnSeq you choose an organism on which to perform the experiment. In the KBase narrative this means you create a genome object in KBase, which will looks like this in your Data tab:



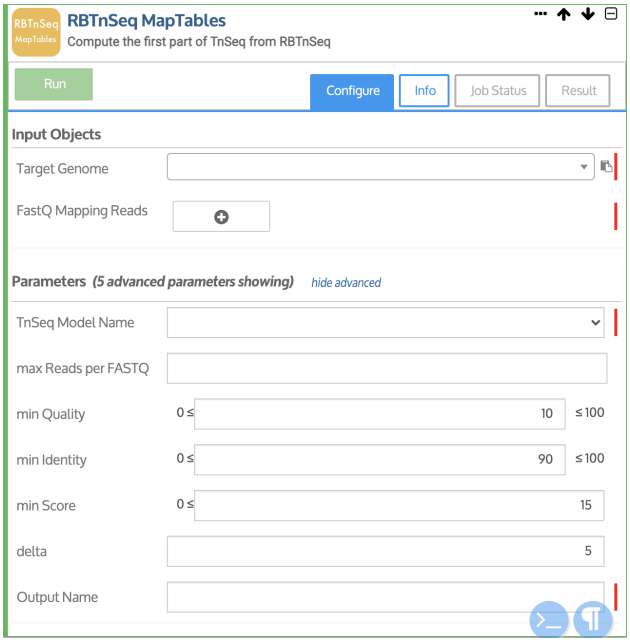
Follow these instructions for assistance in creating a genome object. (Note that the Permanent Id value and the number '14.2' at the end of 'Full Type' are bound to change and the apps do not depend on these).

The FASTQ files you get as a result of TnSeq sequencing (called the TnSeq Reads) are to be used along with the Genome Object to map the locations of the insertions within the genome. Make sure these are single-end reads, if you have paired-end reads, merge the two reads before using the mapping app. Within your KBase narrative, add the FASTQ reads to your data as SingleEndLibrary objects. For a tutorial on how to do that, follow the instructions here.

Another primary step in the RBTnSeq process is to insert barcoded transposons into a cell's genome in random locations using a vector. This transposon is represented by the "**TnSeq model**", which contains two short DNA sequences that represent the vector: the '**model**' sequence and the '**pastEnd**' sequence. The '**model**' sequence is the nucleotide sequences surrounding the barcode (labelled "U1" and "U2" within the RBTnSeq paper (Wetmore et al. 2015)) and with the barcode in the middle (represented by a string of 'N's). The '**pastEnd**' sequence is the nucleotide sequence in the vector that comes *right after* the transposon junction, and is included in *only some* of the standard TnSeq models (the **pastEnd** sequence is used to find the reads that represent intact vectors, i.e. vectors that weren't integrated into the genome). In order to run the mapping program, you must have the name of the **TnSeq model** used to insert the barcodes. A list of the standard **TnSeq models** and their sequences is provided further in the document under the section "Standard TnSeq models". If you used a TnSeq model that is not one of the given models, contact the KBase team to add that model to the list of standard names. If you don't know which TnSeq model was used to insert the barcodes, you can use the app "**RBTnSeq Find TnSeq Model**" to get how many hits each model gets (against your TnSeq Reads FASTQ file).

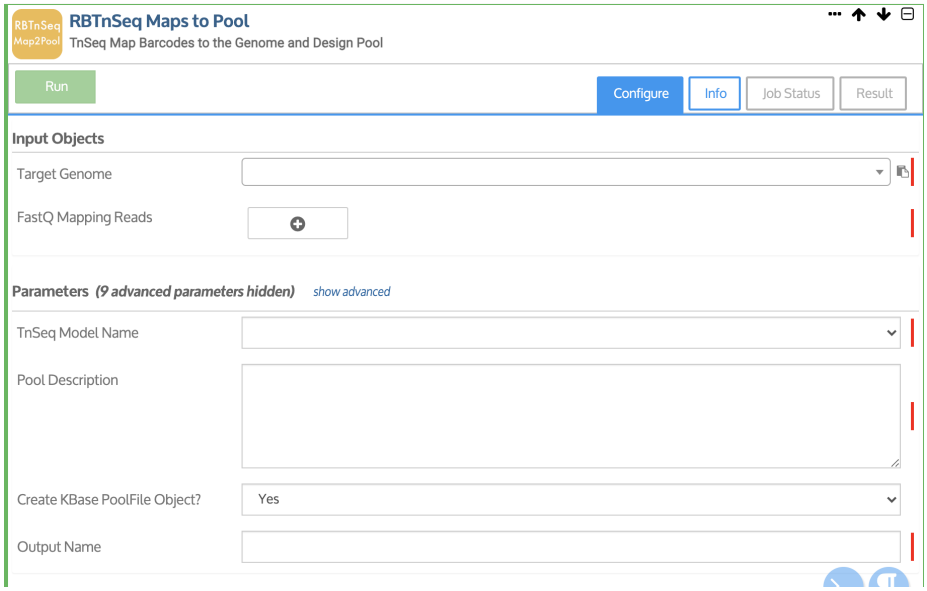
There are two apps you can choose from when it comes to mapping insertions - the first only maps insertions and returns tables which list where insertions occurred for each FASTQ file, and the second maps insertions and generates a mutant pool given constraints. The first of these two apps is called **"RBTnSeq MapTables**", the second is called "**RBTnSeq Maps to Pool**".

First, we explain the app **"RBTnSeq MapTables**", the application which takes in a genome, FASTQ files, a TnSeq model, and certain parameters to generate a single table per FASTQ file and returns it to the user for the user to discern whether to use a batch of organisms to generate a pool.



Referring to above inputs, for the parameter “Target Genome”, choose the genome object of your organism. For the parameter “TnSeq Model Name”, choose the name of the TnSeq model you used as explained above. Under FASTQ mapping reads, you can select multiple FASTQ reads objects (**Note:** again, make sure these are single-end reads, if you have paired-end reads, merge the two reads before using the mapping app). The advanced parameters are described below as they are the same advanced parameters as for the app “**RBTnSeq Maps to Pool**”.

Second, we explain the app “**RBTnSeq Maps to Pool”**. This composite of **RBTnSeq Maps To Pool** and the script **Design Random Pool** encompasses the first major analysis for the RBTnSeq process: it provides you with a mutant pool with which to do further experiments. (*To find the app, you can use the search feature under the “APPS” section in your narrative*)



For the parameter “Target Genome”, choose the genome object of your organism. For the parameter “TnSeq Model Name”, choose the name of the TnSeq model you used as explained above. Under FASTQ mapping reads, you can select multiple FASTQ reads objects (**Note:** again, make sure these are single-end reads, if you have paired-end reads, merge the two reads before using the mapping app). For “Pool Description”, write a brief description of the experiment (there are no hard restrictions on this). For “Create KBase RBTS object”, select “Yes” if you would like to continue the analysis using this data - it may be better to run the app once and consider the results before selecting ‘Yes’ on this parameter (in order to check that the insertion went well enough for you to continue the process to counting barcodes in the future). Selecting ‘No’ means the app is run and you get the results but the KBase pool data type isn’t created. If you run the app while selecting ‘No’ and it ran well and you want to create the KBase pool data type, then you can run it again with the same configuration again except for this time, select ‘Yes’ for this parameter. Finally, the parameter “Output Name” follows the same rules as all other Output Names - don’t use spaces or odd characters.

There are also the following “Advanced Parameters” to consider (although you can ignore them at first): “max Reads per FASTQ” which limits the number of reads the program will process for every FASTQ file. Suppose your FASTQ file has a million reads. If you set this parameter to ten thousand, then the program will only analyze the first ten thousand reads from the file. “Min Quality” is the minimum quality within a barcode for it to be considered an actual barcode. “Min

Identity" is a *blat* parameter, the minimum *percent* identity mapped to a segment of the genome for a barcode mapping to be considered. Can also be thought of as, for a hit, the minimum number of matching bases divided by the number of alignment columns times 100. "Min Score" is the minimum length of sequence after the transposon junction found in the genome for the insertion to be considered to have hit that location. "Delta" - the minimum difference in length from the top hit to the next for considering a mapping unique, "minN" - the minimum number of 'good' reads for a barcode supporting its mapping, where 'good' means a unique hit to the genome and the hit to the genome starts at the first nucleotide after the transposon junction. "minFrac" is the minimum fraction of reads for a barcode that agree with the most common mapping. "minRatio" is the minimum ratio for a barcode between the amount of reads for the most common mapping to the second most common mapping. "max Query Beginning" - the maximum amount of nucleotides after the transposon junction and before the hit to the genome for a hit to be considered.

At this point you will have run "**RBTnSeq Maps to Pool**" successfully and are analyzing the results in order to discern whether your insertion experiment was good enough to move forward with the next steps of the process. The HTML Report gives you three sections to analyze the data. The simplest and most important one is the Statistics Report. The Statistics report gives you some important numbers which are explained under the section "Explanation of MapTnSeq Statistics Report". For an example of what good results for TnSeq look like, refer to these tables.

An explanation of what computations happen in the app can be found under the section "Explanation of the MapTnSeq Program".

Explanation of MapTnSeq Statistics Report:

The statistics report is divided into 3 sections: Model information, FASTQ Reads report, and the Mutant Pool Report.

The **Model Information** simply contains the name of the returned file in which the model text is found and the model string itself (from the TnSeq model).

The **FASTQ reads report** contains information about each FASTQ file in sequential order as inputted in the app configuration.

a. '# Reads Processed' is the total number of reads on this file that were processed. If the value 'maxReads' was not given, then this is equal to the total number of reads in the fastq file. Otherwise, if 'maxReads' was given then the number of reads processed will either be equal to the number 'maxReads' or the total number of reads in the file if it is less than 'maxReads'.
b. '# Reads that are longer than 100 bp' - How many reads in the FASTQ file were longer than 100 bp.
c. '# Reads with a barcode' - How many reads have a barcode found in them (the flanking sequences and a barcode of length ~20 are all found within the read).
d. '# Mappings Attempted' - How many reads had a transposon junction that had at least 'minScore' nucleotides after it. In other words, how many reads had a barcode and a transposon junction, and had 'minScore' (e.g. 5) nucleotides left before the end of the read.
e. '# Reads with Mapped Insertion' is the number of reads that have a barcode, a transposon junction, and, the sequence after the junction matched the genome (there was enough sequence after the insertion to find a location). This value should be less than '# Reads with barcode'. If not, contact the developer responsible for the program.
f. '# Reads that map uniquely' - Is the number of reads that map to the genome and to just one location in the genome (and not to a repetitive element in the genome).
g. '# Reads that map to intact vector' is the number of reads for which the sequence after the transposon junction matches the vector. So the read arose from the intact vector, not from a mutant.
h. All the percentages are the above numbers divided by '# Reads Processed'.

The **Mutant Pool Report** contains information about the output, the Pool File.

a. '# Usable Barcodes' - The total number of barcodes that are usable (across all input FASTQ files).
b. '% Coverage of Mapped Reads ' - Out of the reads that are mapped transposon insertions, what percentage contain usable barcodes.
c. '# Protein-Coding Genes with Central Insertions' - Central insertions are where the transposon was inserted within the central 80% of the gene, i.e. the transposon was not inserted into the first 10% or the last 10% of the gene.
d. '# Central Insertions in Protein Coding Genes' - Central insertions are defined above. How many usable barcodes were inserted in the central 80% of genes.
e. 'Fraction of putatively essential genes with good hits' - Making a crude estimate of what the essential genes are, we divide the number of them with good barcode insertions by their total number (where a good barcode insertion is a central one (10-90%) and has enough reads supporting it). E.g, there are 50 essential genes, and 10 with good insertions, then this value would be .2.
f. 'Fraction of non-essential (putative) genes with good hits' - Similar to above, but using the complement of the above genes. Suppose there are 200 genes, 50 of which are essential, so we count the number of genes out of the 150 non-essential ones that have good insertions and divide by 150. This rate should be higher than the rate at 'e'. This is the value that is listed in the Mutant Pool MetaData as "gene_hit_frac".
g. '# Putatively essential genes with Central Insertions' - putatively essential genes are estimated very crudely. This number should be relatively small.
h. 'Reads per protein: Mean/Median': Taking into account only central insertions (middle 80%), the mean/median number of insertions per gene.
i. '% Of insertions in Protein-Coding genes on the Coding strand': Taking into account only central insertions (middle 80%), the percentage that are on the coding strand (# on coding strand/ total number).

Explanation of MapTnSeq Program

The MapTnSeq program is comprised of two main scripts: one called "MapTnSeq", and one called "Design Random Pool". "MapTnSeq" is a script that runs once per FASTQ file. "Design Random Pool" runs just once, combining the outputs from the multiple MapTnSeq runs. Since MapTnSeq parses a single FASTQ file each time, if there are x FASTQ files, MapTnSeq will run x times, each time producing a table that contains the following columns (in sequential order):

read_name (from FASTQ)

barcode (nt string (usually having length 20))

scaffold (from genome (there may be as few as one scaffold in a genome and as many as over 100))

position (within scaffold (int))

strand (+/-)

unique (1 if yes, 0 if no, uniqueness refers to whether genome sequence is unique).

query beginning loc (the sequence of the transposon junction, when does it start in the genome?)

query end loc (the sequence of the transposon junction, where does its match to the genome end?)

bit score (Score of matching according to blat)

percent identity (Percent match)

The Map Tn Seq tables have one row per read (with a barcode and transposon junction found). So within the Map Tn Seq tables you might have multiple different rows with the same barcode, but you'll never have two rows with the same read name. How the script gets above table is that it looks for the TnSeq model sequence (**TnSeq model**) flanking the barcode within each read from the FASTQ file, making sure the reads pass the quality thresholds such as the **minimum quality** and length of sequence after the transposon junction. Then, for every read name that labels a good read, it stores the barcode associated with it and it stores the sequence after the transposon junction. Then it takes all those sequences after the transposon junctions and runs **BLAT** against the entire genome FNA sequence (all the scaffolds), looking for matches of those sequences to the genome.

BLAT gives information such as bit score and percent identity for each of the matches. Now the program looks at the matches given by BLAT and stores the

matches that pass the thresholds, and for those good ones it marks the read name, the barcode, the location, the strand and whether it mapped uniquely to the genome (some of the sequences after the transposon junction might appear multiple times in the genome).
Now on to the second major script, “Design Random Pool”. It runs a single time, taking as an input all the tables from MapTnSeq as well as the gene table and the genome fna. It parses the tables from MapTnSeq and counts the number of times the barcodes occur and the ratios between their various mapped locations within the genome; if the number of times a barcode was read was good enough, and the ratio between its most common mapping and the second most common mapping (location with the most hits vs the location with the second most hits) is good enough, then the barcode and its location is added to the mutant pool file, so that the Mutant Pool only contains the 'good' barcodes that pass the thresholds. Then it computes the statistics for that pool using the script “PoolStats.R”. The statistics it computes are explained under the section “**Explanation of MapTnSeq Statistics Report”.**