OligoCounter Manual
Version 0.51

Colin Davenport
May 12, 2009

Any queries and comments - please contact the author.
davenport <dot> colin <at> mh-hannover.de

Contents

1 Introduction

OligoCounter is a Java command line program which counts overrepresented 8-14bp oligonucleotides in DNA sequences. These oligos can be useful in visualising small (i.e. less than 10mb) genomes on a global basis, investigating repeats at certain regions, and for attributing short reads (for example from metagenomics) to a sequenced genome.

Plain text output is produced which can be visualised with a suite of associated Java applications, for example OligoViz initially or JCircleGraph. Essentially, this a pipeline of separate programs.

You can get OligoCounter at the following website
http://webhost1.mh-hannover.de/davenport/oligocounter/index.html

If you are a windows user not accustomed to the very limited command line on that system, try the following documentation as a brief primer
http://webhost1.mh-hannover.de/davenport/oligocounter/docs_cmdline.html

2 Quickstart commands

(requires Java, OligoWords requires python and perl. Windows users should change all copy commands (cp) to copy or use a file manager)

```
> cd OligoCounter
> java -Xmx1548m -jar OligoCounter0_64.jar
        (Press 9 to start)
> cp resultsPosit* ../OligoViz/
> cp resultsPosit* ../JCircleGraph/
> cd ../OligoViz
> java -Xmx1000m -jar OligoViz0_50.jar
> cd ../OligoWords
> perl runOligoWords.pl
> cp -R gc ../JCircleGraph
> cp -R d  ../JCircleGraph
> cp -R ps ../JCircleGraph
> cp -R ouv ../JCircleGraph
> cd ../JCircleGraph
> java -Xmx1000m -jar JCircleGraph0_51.jar
```

3 Source files

Source files
OligoCounter has been developed and used extensively within our group with NCBI RefSeq files. These use the RefSeq (NC_000000) part of the fasta header to name output files, since these are unique numbers which are useful for web databases and referencing other NCBI RefSeq files. OligoCounter can be used at present with fasta files from other sources, but the RefSeq header should be faked -- see below. We will provide a more elegant solution for this problem using GI numbers (Genbank identifier) in the future, but the problem is these cannot be as easily referenced to GFF files containing coding data (which can be used for OligoViz) from the RefSeq collection.

Using non-NCBI RefSeq source files
RefSeq accession numbers are parsed from the fasta file and used. The file must contain a RefSeq header even if it is a non-RefSeq genome. These can simply be faked by copying a RefSeq complete fasta header and modifying it. Alternatively, if OligoCounter does not find a RefSeq accession it will generate a fake one.

For example:

RefSeq header:

```
>gi|26986745|ref|NC_002947.3| Pseudomonas putida KT2440, complete genome
```

Make following changes - the NC_XXXXXX is the key part, but also the | characters need to be left in place

```
>gi|0000000|ref|NC_111111.1| My genome description
```

Using NCBI RefSeq files

Get source files from the NCBI ftp site at the following link

ftp://ftp.ncbi.nih.gov/genomes/Bacteria/

OligoCounter requires at least one fasta file in .fna format (fasta). We have widely tested the program on prokaryotic genomes less than 10MB in size but larger genomes can be analysed if more RAM is attributed (we suggest about 2000MB RAM). We have successfully analysed 20MB of human DNA.

Necessary input files:

.fna (pure fasta)

Recommended extra files:

GFF files provide coding coordinates required to create multi colour images from results with OligoViz.

.gff (coding coordinates)

If you want to run analyses for all genomes, it is advisable to download zip files (tar.gz) of all currently publicly available genomes from the NCBI ftp site.

4 OligoCounter

OligoCounter is a Java program that analyse chromosomes or DNA sequences in pure unannotated fasta format for overrepresented oligonucleotides that are 8-14bp in length. Results are supplied as two tab-delimited text files with statistics and genomic positions of the located oligonucleotides.

Requirements:

Java

2 GB RAM

Parameters:

Supplied in the initial program menu.

To use default parameters a -nomenu switch can be used at the end of the command.

To run OligoCounter:

```
java -Xmx1548m -jar OligoCounter<version>.jar
```

<version> should be currently replaced by 0_64

```
java -Xmx1548m -jar OligoCounter0_64.jar
```

Full example:

Take this RefSeq header, and modify your own fasta input file header to this format:

```
>gi|26986745|ref|NC_002947.3| Pseudomonas putida KT2440, complete genome
```

For example:

```
>gi|0000000|ref|NC_111111.1| My genome description
```

1# Make sure Java 1.5 or above is installed and available on the command line Type java at the command shell

```
java
```

If feedback on options is shown Java should be set up correctly!

2# Put all input fna (fasta files) in a clean working directory.

3# Place OligoCounter.jar in the same directory as the fna files

4# Open a shell and change to the working directory

5# Run OligoCounter

```
java -Xmx1548m -jar OligoCounter<version>.jar
```

The -Xmx tag allows java to use a larger amount of memory, eg 1548MB

<version> should be currently replaced by 0_64

6# You are confronted with the OligoCounter text menu:

```
Enter a number to change an option
Options:
0: Lower oligo frequency threshold: 70
1: Chi squared significance threshold: 3000
2: Second Chi squared significance threshold: 0
3: Heuristic version, saves memory, 1 indicates on, 0 off: 0
4: Heuristic frequently removes oligos with counts below: 2
8: Help
9 to exit this menu and start the program
```

Enter the relevant number to change an option, then press enter, then enter a new value for that option See the help for more details on the options. A chi squared value of 500 might be a good place to start for most genomes.

7# For example, enter 0 to change the Lower oligo frequency threshold

```
0
```

```
Enter a new value for the lower threshold: oligos which occur less frequently than this value in the genome
will be ignored (after all oligos in the whole genome have been counted) in subsequent analysis steps:
```

```
20
```

Explanation of all OligoCounter parameters

8# When all values are set to your satisfaction enter 9 to start the program

```
9
```

9# OligoCounter will now run through all fasta files located in the same directory with .fna file extensions.

10# Output files (tab-delimited text) are created in the same directory

Files include a prefix:

resultsStats + extension
resultsPositions + extension

Stats files contain the oligo, number of times it is present and statistics

Positions files contain the oligo, and the position where each oligo instance begins

Fasta ouput files simply contain the oligo in fasta format - these can be useful for multiple sequence alignments

Performance:

On a Pentium 4 3600 this program requires about 3 minutes per megabase to read in, count, sort and output the common oligonucleotides.

Troubleshooting:

- If no more hard disk space exists OligoCounter will crash with a null pointer exception error. Make sure at least 1 GB of free disk space exists before running the program as this is needed for the temporary files it generates.
- RefSeq accession numbers are parsed from the fasta file and used. The file **must** contain a RefSeq header even if it is a non-RefSeq genome. These can simply be faked by copying a RefSeq complete fasta header and modifying it.

5 OligoViz

OligoViz was created to visualise the genomic position of the oligomers in the resultsPositions files created by OligoCounter.

Requirements:

To run OligoViz:

```
java -Xmx1000m -jar OligoViz0_50.jar
```

A drop down list selects the genome by its RefSeq identifier.

Press “Get data” to read the data from this genome.

A dot is printed in each 10kbp region if an oligo is present there. A black dot indicates the instance of the oligo was located in a non-coding region, while a green dot is painted for coding regions.

The genome position to the nearest 10kbp is output to the top right dialog box if the figure is clicked.

Divergent regions tend to be occupied by fewer or no overrepresented oligos and appear as horizontal lines.

Two different tab-separated files can be created. These are best opened with WordPad (not Notepad!) under Windows or any Linux text editor.

The "Save distances" button saves a data level summary to the working directory, with the oligos and the average distance between each instance i.e. position of the oligo. Pushing this button creates a text file like this

```
distsPositionsNC_002947Pseudo_putid_3000.txt
```

in the current directory. This file contains all overrepresented oligos, the log (base10) average distance between all instances of the oligo and the average distance in nucleotides between all instances. These distances allow estimation of how much space occurs between instances of an oligo in the genome - repeats tend to have large average distances while coding oligos tend to be more evenly distributed, i.e. with a smaller average distance, as in the images.

```
>gi|18311643|ref|NC_003364.1| Pyrobaculum aerophilum str. IM2, complete genome
Genome size: 2222430
Oligo      LogAverageDist  AverageDist
GAATCTCAAAAAGA      1.93      27102.0
AATCTCAAAAAGAG      1.93      27102.0
```

The "Save coding" button works only if a .GFF file containing information on coding regions is in the same directory as the current files. Pushing this button creates a text file called

```
codingNC_003364.txt
```

or similar. Parameters in this file include the oligo, the total number of this oligo found in the genome (instances), and the instances in coding regions. From this the percent coding is calculated, and of the coding oligos the percent which are coding but out of frame are also derived using coding region start codon positions. Reading frames are thus all relative to the respective start codon. The number of coding oligos found in the three forward reading frames are also presented. OligoCounter only deals with the direct DNA strand. Frame1 indicates "in frame", i.e. the oligos which appear to be directly translated.

```
>gi|20088899|ref|NC_003552.1| Methanosarcina acetivorans C2A, complete genome
Genome size: 5751492
Oligo     totalOligos       codingOligos      pcCoding          pcCodingOutOfFrame      Frame1  Frame2  Frame3
TTCTTTTT        1241    713     57.5    45.7    387     204     122
AAAAAGAA        1134    623     54.9    68.9    194     297     132
```

6 OligoWords

This program takes a .fna or .fst fasta file in the working directory and calculates genomic statistical parameters from it. Output is a series of plain text files. Full details are published in Ganesan et al. 2008, BMC Bioinformatics.

6.1 **The easy way** (requires perl, python, and linux for the copy command “cp”)

Type

```
perl runOligoWords.pl
```

6.2 **The normal way** (requires python)

Type

```
python
```

Feedback should indicate python is installed correctly.

Add all files to the working directory

Run OligoWords from the command line four times with the four necessary parameters as below.

```
python OligoWords1.2.exe.py task=n0_4mer:XXX, frame=10000, step=5000
```

Where XXX = GC, D, PS, V for each respective run

After each run manually add the parameter to the end of the filename, else it will be overwritten by the next run.

```
NC_006156.out -> NC_006156gc.out
NC_006156.out -> NC_006156d.out
...
NC_006156ps.out,
NC_006156ouv.out.
```

Move all these created files into the JCircleGraph directory to be visualised.

6.3 High throughput analysis

Alternatively, if you are analysing many genomes and do not want to have to rename each file individually:

Create the following directories in the directory where you plan to use JCircleGraph, and insert the raw output files from OligoWords into the relevant directory (take care not to mix them up though !).

```
gc, d, ps, ouv
```

7 JCircleGraph

JCircleGraph visualises whole genome data from OligoWords (required data) and OligoCounter (optional) in an attractive genome atlas format.

See the website and reference for full details of JCircleGraph:
http://webhost1.mh-hannover.de/davenport/oligocounter/docs_jcirclegraph.html

Requires:
Java, six datafiles: four files from OligoViz, two from OligoCounter

To run JCircleGraph:

```
java -Xmx1000m -jar JCircleGraph0_51.jar
```

You can run the jar file by double clicking on it on some Windows PCs, however the memory assigned by default is not sufficient to run the program properly. Therefore it is best to run it from the command line with the -Xmx1000m switch to make 1000 megabytes of memory available.

JCircleGraph requires all 4 tetranucleotide parameters from OligoWords, else will refuse to work. A dropdown list of available genome RefSeqs are currently derived from the resultsPositions files from OligoCounter, so you need to have these files in the same directory.

7.1 Interpreting an image

The dark grey inner ring is the scale, with ticks every 0.25MB or 0.5MB depending on genome size.

The four innermost rings are mononucleotide or tetranucleotide (4mer) parameters derived from a 10kb sliding window using the Python program OligoWords (see the downloads page on this server, or Reva and Tümmler 2004 for the program and a full description of parameters).

1: GC content (proportion of G and C in one window)
2: Distance (of a local 10kb pattern relative to the global genomic pattern)
3: Pattern skew (distance between patterns on the leading and lagging strands in one window)
4: Oligonucleotide variance (variation of word deviations)

The next two rings plot, if information is available, the presence of overrepresented 8-14mers. This is taken as the number of bases in a 5kbp region occupied by an overrepresented 8-14mer, divided by 5000 and multiplied by 100. Hence the results are a percentage occupancy of the 5kbp regions bases by overrepresented 8-14bp oligos.

Example: a single 14mer in a 5000bp region. (14 / 5000) * 100 = 0.28% occupancy.

Example: 2 half-overlapping 8mers in a 5000bp region. (12 / 5000) * 100 = 0.24% occupancy. Percentage occupancy reduces the amount of redundancy in a dataset with overlapping repeats. One characteristic of OligoCounter is overlapping repeats since it uses a window size of 1bp.

5: Percentage occupancy of the 5kbp regions bases by overrepresented 8-14bp oligos at default chi-sq. level 500
6: Percentage occupancy of the 5kbp regions bases by overrepresented 8-14bp oligos at default chi-sq. level 1200

Filtering: Should an inappropriately high chi squared value have been selected for rings 5 or 6, few or no data will be available resulting in a dark blue ring which dominates the rest of the graph. To avoid this, where the for the ring the average minus one standard deviation is less than zero (which is the case when few data are available), the whole ring is filtered and left grey. The solution is to use a lower chi squared value (i.e. rerun OligoCounter).

The correlation class circle, ring 7, indicates the differences between tetranucleotides - in this case oligonucleotide variance in ring 4 - and the innermost 8-14mer percentage occupancy in ring 5.

7: 4mer-8mer correlation class derived from rings 4 (OUV, 4mer) and 5(% occupancy, 8-14mers)

Colours

Colours range from dark blue (below average) through light grey(average) to dark red (above average). These colours cover 3 standard deviations above and 3 below the average, and thus over 99% of normally distributed data. Extremes that do not lie within 3 standard deviations so are coloured more emphatically. By using this colour dimension, regions of the genome which are divergent in various parameters from average can be clearly seen. These may be genome islands, integrated phages, horizontally transferred genomic elements, rDNA or repeat regions.

8 Reference

Davenport, C. F., Wiehlmann, L., Reva, O. N. & Tümmler, B. Visualization of Pseudomonas genomic structure by abundant 8-14mer oligonucleotides. Environmental Microbiology, Epub Jan21 (2009).

9 Credits

Thanks to Lutz Wiehlmann for assistance with program design. Many others from the Tümmler lab at Hannover Medical School contributed with beta testing or comments.