OXBench: A benchmark suite for evaluation of protein multiple sequence alignment accuracy

User and Developer Guide

Version 1.3 - June 2005

Tom Walsh and Geoffrey J. Barton

School of Life Sciences
University of Dundee
Dundee DD1 5EH
Scotland, UK

email: tom@compbio.dundee.ac.uk, geoff@compbio.dundee.ac.uk

If you use this suite, please cite:
Raghava, G.P.S., Searle, S.M.J., Audley, P.C., Barber, J.D. and Barton, G.J. OXBench: A benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics*, **4**: 4-47. 2003.

Contents

1 Introduction

The OXBench suite is a system for assessing the accuracy of multiple protein sequence alignments. For full details about the development of the suite, its evaluation and application to some popular alignment methods, please see Raghava et al (2003). Please see Section: 8 of this User Guide for instructions on how to install the OXBench suite.

OXBench includes a large set of reference alignments derived from 3D-structure comparison of protein domains as well as software to compare alignments, measure agreement by a variety of metrics and estimate the significance of any differences found. The essential steps in using OXBench are outlined here, but full step-by-step instructions are given in Section: 5. In outline:

1. Select the dataset on which you will test the method. This may be one of:

   master This is a set of 672 reference alignments of protein structural domains. Use this only if you are not intending to use OXBench to optimise parameters for the method.

   mref1/2 These are two subsets of the master dataset that are intended for training/test of new methods. i.e. you should use one of these sets with which to develop your method and optimise parameters,then test the method on the other set.

   small This dataset is the subset of the master dataset that contains alignments with 8 or fewer sequences. This can be useful for some computationally intensive algorithms.

   pair This is the subset of the master dataset that contains alignments of only two sequences.

   multi This is the subset that contains alignments with 3 or more sequences.

   full This is the full-length sequences of the domains contained in the master dataset. Use this to test if the method can align correctly a domain that is contained within a longer protein sequence.

   extended This is the master set of domains augmented by sequences of unknown structure. Use this to test the effect of having more sequences in an alignment on the alignment quality.

2. Run the method on the sequence files appropriate to the set you have selected.

3. Convert output of the method to BLC format

4. Choose either to run only on Structurally Conserved Regions (SCRs), or the full alignment. SCRs makes most sense since these regions are most likely to have a genuine alignment. Structurally Variable Regions (SVRs) may have no meaningful alignment.

5. Run the OXBench metric(s) on the results of the alignment method. In the paper (Raghava et al, 2003), a number of different metrics were investigated. Not all provided good discrimination between methods. For this reason, in Version 1.0 of the distributed suite, we only include the following metrics:

   column This counts the number of alignment columns in agreement between reference alignment and test alignment. In the paper, we found this to be less effective than the dep metric at discriminating small changes in alignment quality, but it is often used and so is included for the sake of completeness.

   dep The dep metric measures accuracy for each pair of sequences within a multiple alignment, and reports an average over all the pairs. The output files for this method record both the pairwise accuracy within the multiple alignment and the average of all the pairs.

   stamp The stamp metric does not compare to a reference alignment. Instead, it evaluates the "quality" of structural superposition obtained given the sequence alignment that is presented. "Quality" is reported as an "Sc score". Sc is a measure that takes into account both distance and conformational similarities between the superimposed protein structures. As with the dep metric, the stamp metric is calculated over all pairs within the multiple alignment and an average Sc reported.

6. Start and initialise the *R* package as explained in Section: 4 in order to summarise accuracies and report differences between methods.

2 Organisation of the Data files in the Distribution

Data files are found in the `data` subdirectory of the OXBench installation. Not all data files are needed to run evaluations in OXBench, but they are included in the distribution to allow you to characterise the data sets that are used in the benchmark.

2.1 Reference alignment set lists

Reference set lists contain lists of the reference alignments that comprise the test sets described in the OXBench paper. These lists of alignment IDs (aids) are key to using the OXBench metrics.

full.id The full-length sequence set (605 families)

mref.id Master data set (672 families)

mref1.id Subset of the master set for training and testing (334 families)

mref2.id Subset of the master set for training and testing (338 families)

mref_small.id Small families of 8 sequences or fewer (590 families)

mref_multi.id Alignments with 3 or more sequences (399 families)

mref_pair.id Alignments with 2 sequences in each (273 families)

2.2 Reference alignments and ancillary data

2.2.1 Directory structure

The reference alignments and other associated data are organised into subdirectories as follows:

align/ alignments of the master set families in BLOC format.

dom/ STAMP domain definitions for the domains in the master set famililes.

fasta/ Reference alignment sequences in FASTA format.

pairwise/ Pairwise alignments of sequences in the master set families.

pdb/ PDB files used by the STAMP metric.

pid/ Pairwise sequence identity data for the reference alignments.

scr/ Structural conservation data for the reference alignments.

seq/ Reference alignment sequences in PIR format.

2.2.2 Sequences

The `seq/` and `fasta` directories are subdivided into three directories:

master Sequences sets for the reference alignments (these correspond to the sequences in the master alignment files in the `align` directory.

extended Extended sequence sets for the reference alignments.

full Full-length sequence sets for the reference alignments.

The `master` and `full` directories contain further subdirectories corresponding to the test sets listed in the `$OXBENCH/data` directory.

2.2.3 Pairwise sequence identity data

The file `$OXBENCH/data/pid/all.pid` contains pairwise percentage identity (PID) values for the reference structural alignments. Pairwise values and average PID values for the full multiple alignment are included.

The contents of the first three fields are:

1. The alignment identifier (*aid*).
2. The identifier of the first domain in the pair (*id1*).
3. The identifier of the second domain in the pair (*id2*).

The next four fields (*pid1*,*pid2*,*pid3* and *pid4*) are percentage pairwise identities (PIDs). The value of the PID is given by:

$pid = \frac{n}{l}$.

where n is the number of identical positions and l is the number of positions over which identity is assessed. The PID values differ in the number of positions used to define l.

The four PIDs are:

1. PID calculated over the length of the aligned region, excluding positions where both sequences contain a gap, *i.e.*:

   $pid_1 = 100\frac{n_{ident}}{l_{ar}-n_{dg}}$.

   where n_{ident} is the number of identities, l_{ar} is the length of the aligned region and n_{dg} is the number double gaps.

   The aligned region for a pairwise comparison is defined by removing positions at the start and end in which both sequences contain a gap.

2. PID calculated over the length of the alignment, excluding positions at which there is a gap in one or both of the sequences.

   $pid_2 = 100\frac{n_{ident}}{a_{pos}}$

   where a_{pos} is the length of the alignment, excluding positions with gaps.

3. PID calculated over the number of residues in the shorter sequence.

   $pid_3 = 100\frac{n_{ident}}{min(len_1,len_2)}$

   where len_1 and len_2 are the lengths of the two sequences.

4. PID calculated over the length of the shorter sequence, including internal gaps in the shorter sequence.

   $pid_4 = 100\frac{n_{ident}}{l+n_g}$

   where l and n_g are the length of and number of gaps in the shorter of the two sequences.

The remaining fields are:

- Number of identical positions in the alignment (n_{ident})
- Number of aligned positions excluding positions where there is a gap in either or both sequences (a_{pos})
- Length of first sequence (len_1)
- Length of second sequence (len_2)
- Number of gaps in first sequence (n_{g1})
- Number of gaps in second sequence (n_{g2})
- Length of the aligned region, excluding double gaps (l_{ar})
- Number of alignment positions at which there is a double gap (n_{dg})

The penultimate line in the file contains the mean PIDs across the entire multiple alignment.

The last line contains the minumum PIDs in the multiple alignment.

2.2.4 Sequence identity calculated across multiple alignments

Percentage sequence identity values calculated across all sequences in an alignment for the reference alignments can be found in $OXBENCH/data/pidw.dat. The fields are:

- Alignment identifier
- Percentage sequence identity
- Number of identical positions
- Length of the alignment.

2.2.5 Structurally conserved regions

Structurally conserved regions (SCRs) are those in which one can have greater confidence that the reference alignment is meaningful when compared to other positions in the alignment. In the OXBench reference alignments, SCRs are defined as follows:

SCRs are stretches of 3 or more alignment positions in which the STAMP P'_{ij} score at each position is ≥ 6. The P'_{ij} score is column 7 in the vertical alignment section of the reference alignment file.

The file `$OXBENCH/data/scr/all.scr` contains data about the proportion of each reference alignment that is structurally conserved. The columns in the file are:

id The reference alignment identifier.

len The length of the alignment.

nscr The number of alignment positions that are structurally conserved.

pscr The percentage of alignment positions that are structurally conserved ($\frac{nscr}{len}$).

2.2.6 Sequence family sizes

The number of sequences in each reference alignments is listed in $OXBENCH/data/seq_count_master.dat. Corresponding data for the extended sequence families is in $OXBENCH/data/seq_count_extended.dat

3 Metrics

3.1 Introduction

There are three sequence alignment metrics provided in the package. This section summarises how they work and how to run them. See the Raghava et al(2003) for full details of how they are calculated. Run the metrics with the –help option to get a detailed list of the command-line options. The metric programs read alignments in BLOC format. Alignments in other formats can be converted to BLOC format by the aconvert program included in the OXBench package (see Section: 3.2 for details).

The run_metric.pl program (see below) is a convenient way to run a metric on a large number of test alignments at once.

3.2 Using `aconvert` to create alignments in BLOC format

The program `aconvert` can be used to generate alignments in BLOC format from other formats. The usage of `aconvert` is:

`aconvert [-in` *`type`* `-out` *`type`*`] < file1 > file2`

where *type* is one of:

c Clustal format

b BLOC format

f FASTA format

m MSF format

p PIR format

s PFAM (Sanger) format

If the input format is not specified, `aconvert` will attempt to identify it. If the output format is not specified, it is assumed to be BLOC format.

3.3 Running the programs in the package

The programs used in the package can be found in the `bin/` directory of the OXBench installation. The default behaviour of the metrics is to do their calculations only over the structurally-conserved regions (SCRs) in the alignments. To do the calculations over all positions, pass the –all option to the metric. If the metrics are run in default mode but there is no structural data in the reference alignment file, the metric will issue a warning and switch to assessing accuracy over the entire aligned region.

The metrics are run using commands of the form:

metric-name [options] -i TEST -r REFERENCE

where *TEST* is the file containing the test alignment of a given alignment from the reference set, and *REFERENCE* is the corresponding reference alignment. Both alignments must be in BLOC format.

3.3.1 Metric command line options

-r, –reference_alignment=STRING The reference alignment in BLOC format.

-i, –input=STRING Input (test) alignment in BLOC format.

-f, –full Input alignment is a full-length sequence alignment

-n, –alignment_id=STRING Alignment name to be included in the output, defaults to "default")

-a, –all Calculate accuracy over the full aligned region, not just the SCRs

-m, –allow_id_mismatch Don't abort if the test and reference seqs have different IDs

-h, –help Print help and exit

-V, –version Print version and exit

-v, –verbose display more information

-d, –debug display debug information

3.4 The dependent average accuracy metric - 'dep'

This metric compares a multiple alignment to a reference alignment (see below). The accuracy of each pairwise alignment in the multiple alignment is calculated by counting the number of positions that agree with the reference alignment. The overall agreement is the average of the pairwise alignment accuracies.

The metric is run using the `dep_acc_avg.metric` and is referred to as the 'dep' metric.

3.5 The dependent column score metric

This metric compares the columns of the test alignment with those of the reference alignment and calculates the percentage of the columns that are identical.

The metric is run using `column.metric`

3.6 The independent metric (STAMP metric) using structural comparison

This metric uses the alignment to create a structural superposition of the corresponding protein structures and calculates a structural similarity score based on the C_α atoms. The program for this metric is `stamp.metric`. Note that this metric requires a reference alignment only because the reference alignment contains structural information used to specify the portions of the PDB structures used in the structural comparison.

3.6.1 PDB files for the independent metric

The STAMP metric requires the PDB files corresponding to the sequences in the alignment. The metric finds the PDB files using a set of rules in the data/pdb.directories file. This file is created automatically during the installation process. There should be no need to edit this file unless the OXBench installation is moved to another location, in which case it can be recreated by running "make pdb.dir" in the OXBench directory. The entries in this file have the format:

Path Prefix Suffix

PDB files are assumed to have names of the form:

Path/PrefixPDBcodeSuffix

where *PDBcode* is a Brookhaven PDB code.For example, PDB files with conventional Brookhaven names, stored in /db/pdb/, can be found using the rule:

```
/site/packages/oxbench/data/pdb pdb .ent
```

A value of '_' for the path, prefix or suffix indicates that the corresponding element should be omitted from the filename.

3.7 The output format of the metrics

The metrics output data in comma-separated variable(CSV) format. The first line is a list of headers explaining what the corresponding fields mean.

3.7.1 Output format of the dependent average accuracy metric

The output format for the dependent average accuracy is:

```
aid,id1,id2,metric,pairwise,score,nmatch,ncomp
default,1bmda-1-GJB,1hlpa-1-AUTO.1,dep_acc_avg,1,86.62
default,1bmda-1-GJB,1hyha-1-AUTO.1,dep_acc_avg,1,67.30
...
default,total,total,dep_acc_avg,0,75.35
```

The first line lists the headers:

- aid: alignment identifier.
- id1 and id2: sequence identifiers. If the data are for the full multiple alignment, both identifiers will be 'total'.
- metric: the name of the metric
- pairwise: 1 if the data on this line are for a pairwise comparison, 0 if they are for a multiple alignment.
- score: the pairwise accuracy (%)
- nmatch: the number of positions at which the test alignment matches the reference alignment.
- ncomp: the number of positions at which the test and reference alignments were compared.

The subsequent lines list the data for each pairwise comparison. The final line contains the data for the overall multiple alignment. The score is the average pairwise score over the all of the pairs in the alignment. The *nmatch* field contains the sum of the pairwise scores and the *ncomp* contains the number of pairwise comparisons.

3.7.2 Output format of the column metric

The headers in the column metric output are:

- aid: alignment identifier.
- id1 and id2: sequence identifiers. If the data are for the full multiple alignment, both identifiers will be 'total'.
- metric: the name of the metric
- pairwise: 1 if the data are for a pairwise comparison, 0 if they are for a multiple alignment..
- score: the column accuracy (%)

- nmatch: the number of positions at which the test alignment matches the reference alignment.
- ncomp: the number of positions at which the test and reference alignments were compared.

3.7.3 Output format of the structural metric

The header line is:

```
aid,id1,id2,metric,pairwise,rmsd,sc,pfit,ntofit,ia,ib,lena,lenb
```

aid,*id1*,*id2*,*metric* and *pairwise* have the same meanings as for the dependent metric. The other fields are:

- *rmsd* is the RMSD for the pairwise comparison,
- *sc* is a structural similarity score (*Sc*; see the paper for details).
- *pfit* is the percentage of the alignment positions used to calculate the RMSD and *Sc* scores.
- *nfit* is the number of alignment positions.
- *ia* is the number of gaps in sequence *A* in the alignment.
- *ib* is the number of gaps in sequence *B* in the alignment.
- *lena* is the length of sequence *A*.
- *lenb* is the length of sequence *B*.

3.8 Running metrics on sets of alignments using `run_metric.pl`

The set of reference alignments is in the `ref/` directory. The metrics can be run for the one of the reference sets using the `run_metric.pl` program: To run this program, the OXBENCH environment variable must be set to the path of the OXBench installation directory.

`run_metric.pl` *`metric-name id-file`*

metric-name is the name of the metric to be run. Use `run_metric.pl --list` to get a list of the available metrics.

id-file is a file containing a list of the reference alignments on which the metric is to be run. If *id-file* does not contain a directory path, the program will look for the file firstly in the working directory and then in `$OXBENCH/data/`, which contains lists for the reference sets described in the OXBench paper. The `$OXBENCH/data/README` file contains descriptions of each of the subset files.

The output is a set of files, one for each alignment in the reference set, containing the metric data in CSV format (see the descriptions of the metrics for details of the output format). The data for the alignments are also aggregated

into a single file. This file will have a name that matches the pattern

`<id-file>-<metric-name>.csv`.

- `<id-file>` is the reference set used.
- `<metric-name>` is the metric name.

Reference alignment files are assumed to be in the `$OXBENCH/data/align` directory unless `--refdir` option is used to specify another directory. The default format for reference alignments is BLOC format. If you want to use an alternative set of reference alignments, the `--fasta` option can be used to specify that the reference alignments are in FASTA format.

Test alignment files are assumed to be in the current working directory unless the `--testdir` option is used to specify another directory. The default format for test alignments is FASTA (although the program also accepts alignments in BLOC format if the `--blc` option is used). By default, test alignments are assumed to have names identical to the corresponding reference alignments. For example, a test alignment for family 10t11 should be in a file named "10t11". The `--suffix` option can be used to specify extensions to the default name, e.g. `--suffix .fa` will make the program assume that the test alignment for 10t11 is named "10t11.fa".

Test alignments are assumed to be generated from the families of sequences in the master reference set or from extended sets containing additional sequences. Use `--full` to specify that the alignments contain full-length sequences; otherwise the metric will report errors because the test sequences do not match those in the reference alignemtns.

3.9 Running metrics on combinations of test sets and metrics

It usually the case that one wants to run all of the metrics on alignments of all of the test sets. Since this can be tedious using `run_metric.pl`, the `$OXBENCH/bin/oxbench.pl` script is provided to make benchmarking easier. See Example 4 in Section 5 of this manual for an explanation of how to use this script.

4 Analysis of metric results and comparison of methods

4.1 Introduction

The OXBench metrics produce results in a straightforward to parse comma-delimited format. Since analysis of these basic results requires combining the data in different ways (e.g. by percentage identity range) and the evaluation

of statistical significance when comparing different methods, we have employed the *R* language for all subsequent data analysis.

R allows interactive exploration of data, while including very powerful tools for subsetting and otherwise manipulating the data, plotting in a wide range of styles, and calculating statistics. Since *R* is a full programming language, we have written some functions to perform standard operations on a single OXBench metric result and to allow straighforward comparison of two metric results and assessment of significance of differences between the methods. The functions allow the alignments to be partitioned easily into different subsets based on the percentage identity of the reference alignments.

A step-by-step example of using the *R* functions is given in Section 5 but first we detail the basic steps in getting data into *R*, what each function does, and how to get hard copy.

4.2 Editing the oxbsetup.R script

In the *analyse/* subdirectory, edit the script *oxbsetup.R* and change the directory path to suit your installation.

4.3 Starting R and initialising reference data

Assuming *R* has been installed on your Linux workstation, just typing: *R* in the *analyse/* directory should produce something like:

```
R : Copyright 2004, The R Foundation for Statistical Computing
Version 2.0.1  (2004-11-15), ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for a HTML browser interface to help.
Type 'q()' to quit R.

>
```

The ">" is the prompt. To initialise the *R* data and load the *R* functions, type:

```
> source("oxbsetup.R")
Read 605 items
Read 523 items
Read 334 items
Read 338 items
Read 672 items
Read 399 items
Read 273 items
Read 590 items
Read 52514 records
Read 672 records
Read 672 records
Read 672 records
>
```

The numbers reported should be identical to the above for OXBench Version 1.0 data. The *R* system is now ready to analyse some alignment metric output. The functions available are given in the following sections.

4.4 Reading metric data

There are three functions that are defined as follows:

```
id.column= readmetric.column(resultsdir="",resultsfile="")
id.dep = readmetric.dep(resultsdir="",resultsfile="")
id.stamp = readmetric.stamp(resultsdir="",resultsfile="",score="sc")
```

Are designed to read the single file summary .csv file for a method/metric combination. You can supply either one or two arguments. If one, then it should be the full filename including the path. If two, then the first argument can be the directory and the second the filename.

`readmetric.stamp` has an extra argument to allow scores other than the default Sc score to be used. The default score is recommended since the others do not work as well (Raghava et al, 2003).

4.4.1 Example

```
id.column =
readmetric.column("/jobs/tom/oxbench/clustalw/master/column/mref-column.csv")
```

Will read the column metric output into an *R* data structure called "id.column". These data structures are the root for all subsequent data analysis in OXBench. The full description of each structure is given in the oxbsetup.R file, but essentially it follows the columns documented for each metric as explained in Section 3.7.

4.5 Plotting a boxplot for a method by percentage identity range

Boxplots allow multiple distributions of results to be plotted in a convenient way. The mid-line of the box shows the median of the data, the upper and lower extents of the box enclose 50% of the data, while the upper and lower whiskers, enclose 75% of the data.

Analysis of a single method/metric combination can be done by the following function:

```
oxb.bplot(meth,
          idint=c(0,10,20,30,50,100),    #OPTIONAL
          subtitle="",                   #SET TO CHOSEN TITLE
          addrnge=TRUE,
          metric="dep")
```

This plots a boxplot of accuracies from the output of a readmetric function A boxplot is produced where each box represents the accruracy of the method within the PID range. The mean accuracy in each range is shown on the plot with a triangle symbol The number of data points in each PID range is shown in line with the 5

Arguments to the function are:

meth the output of readmetric.dep, .column or .stamp.

idint breakpoints for percentage identity binning.

ilab character string for labelling bins (not needed as this is autogenerated)

subtitle arbitrary text that will be added to the bottom of the plot

addrnge if set to TRUE a box will be added to the plot for the complete data set

metric the metric type that we are plotting - default is "dep".

varwidth set this to TRUE to get boxes who's width is proportional to the number of counts in each range. This is misleading if addrnge is also TRUE.

A boxplot is produced on the current graphics device where each box represents the accruracy of the method within the PID range. The mean accuracy in each range is shown on the plot with a triangle symbol The number of data points in each PID range is shown in line with the 5accuracy.

See help(boxplot) to find out what the returned data structure contains.

4.6 Comparing methods and assessing the significance of differences

Given two different alignment methods that have been run on the benchmark, the task is to see which method performs best and whether any difference is significant.

The main function for doing comparison of methods is oxb.compare.methods

```
oxb.compare.methods(meth1,meth2,
        method1="",method2="",
        idint=c(0,10,20,30,50,100),   #OPTIONAL
        metric="dep",                 #REQUIRED
        npcols=2,nprows=3)            #OPTIONAL
```

This takes the output of readmetric.dep etc for two different alignment methods, or parameter combinations of alignment methods

meth1, meth2 output of readmetric.dep or similar

method1, method2 strings to identify the methods, e.g. "clustalw" "muscle"

idint vector of breakpoints for percentage identity binning. Default is shown as in the paper.

ilab character vector of labels that corresponds to the idint breakpoints. This is only here as the default. In fact, the character labels are auto-generated in the function by oxb.construct.pid.labels().

metric string to identify the type of metric used

npcols, nprows number of rows and columns of plots to put on each page.

oxb.compare.methods returns a table of results that includes the average metric score for each method, the difference between the average for each method in each PID range and the significance of the differences (estimated by the non-parametric Wilcoxon signed-rank test) for all the data and for subsets within percentage identity ranges. The function also produces sets of graphs of meth1 against meth2 in each PID range and for all the data. One or more pages are written depending on the setting of npcols and nprows.

On an X11() device, it is best to arrange npcols and nprows such that all plots will appear on one page. The default settings will do this. On other devices (e.g. pdf() or postscript()) multiple pages will be generated if the plots will not all fit on one page.

4.7 Comparing two result sets in detail

Having established the overall differences between methods by application of oxb.compare.methods, oxb.diffs allows a more detailed examination of differences for each multiple alignment and pairwise alignments within each multiple alignment.

```
oxb.diffs(oxb1,oxb2,decrease=FALSE,thresh=0,pairwise=0,report="all",dp=2)
```

Given results from two methods (or OXBench runs) returns a table that shows the value for the metric by each method on each alignment sorted by the absolute value of the accuracy difference.

Set "decrease" to TRUE to get differences sorted from largest to smallest

Only absolute differences greater than "thresh" are output To report all differences, set thresh to something very negative, e.g. -10000

If "pairwise" is set to 0, differences are calculated on the score for the complete alignments. If pairwise is set to 1, differences are calculated on the pairwise alignments that make up the metric (for stamp and dep metrics - column metric does not use pairwise comparisons)

if pairwise =1, six columns are output: aid, id1, id2, score1, score2 and score2-score1

if pairwise =0, four columns are output: aid, score1, score2 and score2-score1

set "report" to "all" to output both positive and negative differences (default) or "positive" or "negative" for respective output.

dp sets the number of decimal places to round to for reporting differences. Default 2.

Positive values of difference say that oxb2 is better than oxb1 dataset

This function is most useful when used in conjunction with subsetting by oxb.subset.data. oxb.subset.data allows individual alignments or small sets of alignments to be picked out for detailed scrutiny.

4.8 Utility Function: Subsetting data

Often one wishes to look at a subset of data rather than the complete master dataset. oxb.subset.data provides a convenient way to access subsets.

```
oxb.subset.data(oxbl,idlst)
```

This takes the data as read in by readmetric.column etc and creates an object that contains the subset of alignment results data for the Alignment IDs (aids) that are shown in idlst.

For example, assuming that id.column contains a full result set for the master dataset of alignments; doing the following would get the subset of alignments that are indicated in the character vector: c("22s39","12t116","34")

```
oxb.subset.data(id.dep,c("22s39","12t116","34"))
```

There are some pre-loaded subsets such as:

```
mref_pair.id   - the set of alignments that only contain two sequences.
mref_multi.id  - the set that contain at least 3 sequences.
```

For example, to get the results just for the alignments that have 3 or more sequences in them do:

```
oxb.subset.data(id.dep,mref_multi.id)
```

See oxbsetup.R for the complete list of id lists that are loaded by default.

4.9 Utility Function: Make PID labels

```
oxb.construct.pid.labels(idint,addrange=TRUE)
```

Takes a vector of breakpoints and makes a vector of ranges for annotating graphs etc

4.9.1 Example

```
oxb.construct.pid.labels(seq(0,100,10),addrange=FALSE)
```

gives:

```
[1] "0-10"   "10-20"  "20-30"  "30-40"  "40-50"  "50-60"  "60-70"  "70-80"
[9] "80-90"  "90-100
```

setting addrange to TRUE (default adds the full range of the idint as the last element of the vector e.g.:

```
oxb.construct.pid.labels(seq(0,100,10),addrange=T)
```

gives:

```
[1] "0-10"   "10-20"  "20-30"  "30-40"  "40-50"  "50-60"  "60-70"  "70-80"
[9] "80-90"  "90-100" "0-100"
```

4.10 Utility function: Check the order of two results sets

```
oxb.order.check(oxb1,oxb2)
```

Takes two sets of data as read by readmetric.column etc, and checks that the order of the "aids" is the same in both data sets. This should *not* be needed if the data sets have been generated by the OXBench Perl scripts since they will output the data sorted in the same way. However, it is worth using this function if, when comparing methods, the results look seriously weird.

4.11 Utility Function: Binning data by PID range

These three functions are used by other functions to create binned accuracy data in PID ranges. They can of course be used in isolation.

```
column.score2(id.column,idint=c(0,10,20,30,50,100))
dep.score2(id.dep,idint=c(0,10,20,30,50,100))
stamp.score2(id.dep,idint=c(0,10,20,30,50,100))
```

Given the result of using readmetric.column, dep or stamp, these functions bin the accuracy results according to the mean PID for each alignment. idint is a vector of breakpoints for the binning. The default shown is as was used in the paper, but you could select more break points, e.g.

```
column.score2(id.column,idint=seq(0,100,10))
```

for breakpoints at 0,10,20,30,40,50,60,70,80,90,100 PID.

These functions require the "pid" reference data to have been read into *R*. This should have happened already if you have sourced the oxbsetup.R script.

4.12 Combining R functions interactively

R functions may be arguments to *R* functions, so it is common to combine reading data from the .csv files with some analysis or subsetting. For example, for the master dataset to compare two results of running the "dep" metric that are stored in the files: "clustalw.csv" and "muscle.csv" you could do:

```
oxb.compare.methods(readmetric.dep(''clustalw.csv''),
                    readmetric.dep(''muscle.csv''),metric=''dep'')
```

If you only want to look at the subset of results for alignments that have two or more sequences in them, you can do:

```
oxb.compare.methods(oxb.subset.data(readmetric.dep(''clustalw.csv''),mref_multi.id),
    oxb.subset.data(readmetric.dep(''muscle.csv''),mref_multi.id),
                    metric=''dep'')
```

which will read the data from file, take the subset of alignments defined in "mref_multi.id", then compare the two methods, produce plots in PID ranges and statistics.

5 Worked examples - benchmarking clustalW and AMPS

5.1 Introduction

This section describes how to benchmark clustalW using the master sequence families and the corresponding full-length and extended sequence families. The data for this example can be found in `$OXBENCH/example/clustalw`.

This directory contains three subdirectories containing clustalW alignments and benchmark data for the master, full and extended test sets. The clustalW alignments were created using the default parameter settings for clustalW. The test alignments are in FASTA formats. The metrics accept only test alignments in BLOC format but we will use the `$OXBENCH/bin/run_metric.pl` program to run the metrics; this program accepts test alignments in FASTA format and automatically converts them to BLOC format for reading by the metrics.

5.2 Example 1: Benchmarking on the master reference set using the STAMP metric

This section describes an example of using the STAMP metric to benchmark clustalW alignments of the master test set. Data for the example is in `$OXBENCH/example/clustalw/master`. The directory contains the subdirectories:

- fasta - clustalW test alignments in FASTA format (required by the metric)
- column - results of running the column metric.
- dep - results of running the 'dep' metric.
- stamp - results of running the STAMP metric.

1. Align the master set families using clustalW

   The sequence families in the master reference set (listed in `$OXBENCH/data/mref.id`) were aligned using clustalW with the default parameter settings. The input sequence files were read from `$OXBENCH/data/fasta/master`. These alignments are in the `$OXBENCH/example/clustalw/master/fasta` directory and are in FASTA format. The metrics accept only test alignments in BLOC format but we will use the `run_metric.pl` program to run the metrics; this program accepts test alignments in FASTA format and automatically converts them to BLOC format for reading by the metrics. There are two important points to note about the naming of the test alignment files:

   - The names correspond to the reference alignments in $OXBENCH/data/align, *e.g.* the alignment of family 9t3 is in file `9t3.fa` and will be compared to `$OXBENCH/data/align/9t3`. This correspondence is essential to the running of `run_metric.pl`.
   - The '.fa' file suffix is arbitrary; we could just as easily have have used '.fasta' or any other suffix. We could in fact have used no suffix and simply named the test alignment of family 9t3 as '9t3'. By default, `run_metric.pl` assumes that the test alignments have no suffix. We will see below how to specify the suffix if, as in this case, one is used.

2. Run the 'dep' metric using `run_metric.pl`

   (Ensure that the $OXBENCH environment variable is set to point to the OXBench installation directory).

   Change into the `$OXBENCH/example/clustalw/master/stamp` directory and run the `run_metric.pl` program using:

   ```
   $OXBENCH/bin/run_metric.pl stamp mref.id --testdir ../fasta --suffix
   .fa
   ```

This produces a set of output files in CSV format, one for each family in the reference set list in `$OXBENCH/data/mref.id`. Note that:

(a) There is no need to specify the full path of the `mref.id` file; `run_metric.pl` will know where to find it

(b) The `--testdir` option is used to specify that the test alignments are in `../fasta` (relative to the working directory).

(c) The `--suffix` option to used to indicate that the test alignments have a '.fa' extension.

5.3 Example 2: Benchmarking full-length sequence alignments using the dep metric

Benchmarking alignments based on full-length sequences entails extracting the master sequences from the full-length sequences in order that they can be compared with the reference alignments. The data for this example is in $OXBENCH/example/clustalw/full.

1. Align the full-length sequence families using clustalW

   clustalW was run as described for the master set but using the set list in `$OXBENCH/data/full.id` and sequences read from `$OXBENCH/data/fasta/full`. The `full.id` list differs from `mref.id` because full-length sequences do not exist for all master alignments; see the OXBench paper for details).

   The data for this example is in `$OXBENCH/example/clustalw/full`.

2. Run the 'dep' benchmark

   Change into the `$OXBENCH/example/clustal/full/dep` directory and run `run_metric.pl`:

   ```
   run_metric.pl dep full.id --testdir ../fasta --suffix .fa --full
   ```

   Note that we use the `full.id` set list and that the `--full` option has been added to indicate that the test alignments contain full-length sequences from which the master sequences must be extracted for comparison to the reference alignment.

5.4 Example 3: Benchmarking using extended sequence families and the column metric

Extended sequence families contain additional sequences to those found in the reference alignments. The metrics automatically filter extended alignments to remove these additional sequences before comparing them to the reference alignments. The data for this example is in `$OXBENCH/example/clustalw/extended`.

1. Align the extended sequence families

The procedure here is the same as for the master set. The same reference set is used as in the example for the master sequences but the sequences are read from `$OXBENCH/data/fasta/extended`. The resulting alignments are in `$OXBENCH/example/clustalw/extended/fasta`.

2. Run the metric

   Change into the `$OXBENCH/example/clustalw/extended/column` directory and run the column metric using:

   ```
   run_metric.pl column mref.id --testdir ../fasta --suffix .fa
   ```

5.5 Example 4. Benchmarking clustalW using multiple combinations of test sets and metrics

Benchmarking using `run_metric.pl` can be somewhat tedious if one wishes to run all of the benchmarks on all of the test sets. For the sake of convenience, the OXBench distribution includes the `$OXBENCH/bin/oxbench.pl` script, which runs `run_metric.pl` using combinations of metrics and test sets specifed by the user. The documentation for this script can be viewed by running `$OXBENCH/bin/oxbench.pl --help` or `perldoc $OXBENCH/bin/oxbench.pl`. The script makes a number of assumptions about the layout of the directory containing the test data. This is most easily explained by reference to the clustalW example in the `example/` directory of the OXBench package. The starting point for running the benchmarks on this data is to have a directory containing the clustalW data with the layout below:

```
clustalw/
  extended/ Data for the extended set.
   fasta/
    Test alignments in FASTA format.
  full/     Data for the full set.
   [Subdirectory structure as for extended/]
  master/   Data for the master set
   [Subdirectory structure as for extended/]
```

The script would be run using:
`$OXBENCH/bin/oxbench.pl --methods=clustalw`
in the `$OXBENCH/example/` directory.

Within each set directory, the script creates subdirectories containing the benchmark data. For example, after running the script, the `clustalw/extended` directory will have this structure:

```
clustalw/
 extended/
```

```
fasta/
column/
column-all
dep/
dep-all/
stamp/
```

with each metric directory containing the corresponding CSV file.
The script assumes that:

1. The data for a given method is in a directory that is specified using the `--method` option, e.g. `--method=clustalw`. Multiple methods can be specified by separating the method names with commas, e.g. `--method=clustalw,muscle`.
2. The test alignments are in a subdirectory named `fasta`.
3. The test alignment files are in FASTA format.
4. The test alignment files have a `'.fa'` extension.

5.5.1 Specifying the metrics to be used

The default is to run all three metrics. Other combinations of metrics can be specified using the `--metrics` option, e.g. `--metrics=dep,stamp`. Note that the 'dep' and column metrics are automatically run in both SCR-only and full-alignment modes.

5.5.2 Specifying the test sets to be used

By default, the script assumes that test alignments exist for the master, full and extended sets. To explicitly define the sets that exist, use the `--sets` option, e.g. `--sets=master,full`.

5.6 Example 5. Benchmarking AMPS with multiple metrics and test sets

Benchmarking AMPS can be done as in the previous example. However, special options are needed beceausse AMPS output is in BLOC format rather than the default FASTA format. In the AMPS example, the alignments are in a directory named `'blc'` rather than `'fasta'` and the file suffix is `'.blc'` rather than `'.fa'`. To run the script on this data, do:
`oxbench.pl --method=amps --blc --suffix=.blc --testdir=../blc`.

5.7 Analysing the clustalW result in R

To generate a boxplot of the clustalW results for the dep metric on the master dataset, first cd to the `/analyse` directory, then the following commands should work after starting R.

```
> source("oxbsetup.R")
Read 605 items
Read 523 items
Read 334 items
Read 338 items
Read 672 items
Read 399 items
Read 273 items
Read 590 items
Read 52514 records
Read 672 records
Read 672 records
Read 672 records
> X11()
>
oxb.bplot(readmetric.dep("../example/clustalw/master/dep/mref-dep.csv"),
metric="dep",subtitle="clustalw, master dataset")
```

The above output has been truncated to delete the text output of oxb.bplot.

In order to plot a different metric, substitute the appropriate readmetric function and metric string to the oxb.bplot function. See Section: 4.5 above for details of plotting options and explanation of oxb.bplot output.

To get hard copy output use the pdf() or ps() functions in R. For example:

```
> pdf(file="clustalw_bplot.pdf")
> oxb.bplot(readmetric.dep("../example/clustalw/master/dep/mref-dep.csv"),
  metric="dep",subtitle="clustalw, master dataset")
> dev.off()
```

See help(pdf) for info on pdf plotting options.

5.8 Comparing the clustalw result to the AMPS result

Results for the AMPS multiple alignment method are included in the `example/amps` subdirectory. The directory structure is the same as that for clustalw.

To compare the results of clustalw and amps on the dep metric use the oxb.compare.methods function that is described in Section: 4.6. For example:

```
> oxb.compare.methods(readmetric.dep("../example/clustalw/master/dep/mref-dep.csv"),
  readmetric.dep("../example/amps/master/dep/mref-dep.csv"),
  metric="dep",method1="clustalw",method2="amps")
Read 51842 records
```

```
Read 51842 records
        clustalw     amps  Acc2-Acc1 Wilcoxon p
0-100   89.27628 89.44705  0.1707738  0.4563246
0-10    22.87333 22.09429 -0.7790476  0.6700765
10-20   58.29930 58.71404  0.4147368  0.9845966
20-30   79.10159 79.74000  0.6384127  0.7411265
30-50   91.61960 91.97794  0.3583429  0.5338825
50-100  98.80177 98.81455  0.0127809  0.3420469
```

You may see "Warning" messages from the wilcoxon function in R, but these can be safely ignored.

This will produce multiple plots on the current graphics device showing the relative accuracy of clustalw as measured by this metric in each of the PID ranges selected.

As for the oxb.bplot function, you can get hard-copy output by selecting the pdf() graphics device.

See the Section: 4.7 and Section: 4.8 for functions that allow more detailed comparison of methods and subsetting of the data.

6 Developing new metrics

6.1 Introduction

There are many possible ways of scoring agreement between multiple alignments and so OXBench allows for straightforward addition of further metrics within its framework. If you wish to try a new metric on the OXBench data, then the guidelines below should help you to implement your metric within the OXBench framework. We will be happy to work with you to do this. Please contact Geoff Barton to discuss.

New metrics can be added to the OXBench suite by adding a C++ program as described below. In principle, it would also be possible to write code in another language (e.g. Perl), but this would require some additional effort, in particular to deal with SCRs and file i/o.

The source code that implements the dependent metric can be found in metrics/dep_avg_acc/main.C. The code has extensive comments that explain how to use the C++ classes in the OXBench framework to implement the metric. This file should be easy to modify to implement other metrics. Source code documentation in HTML, LaTeX and as man pages is in the docs/ directory.

The supplied metrics use the Gengetopt library to create the command-line argument parsers. Further options can be added by modifying the .ggo files in the metric directories and regenerating the corresponding source code before rebuilding the executables. Options common to all metrics are defined in the global_options.ggo file in the src/ subdirectory.

The source code for every metric must include metric.h and use the following two macros defined in metric.h:

metric_premain() expands to set up global variables used by the metric. This macro must be included in the source code before the main() function.

metric_setup() calls the functions that parse the command-line arguments. This macro should be called in the main() function after the variable declarations.

7 New Reference Datasets

Version 1.0 of OXBench has 672 reference alignments. It is based on the 3Dee database of structural domains and was frozen in 1998. Given the expansion in the PDB since then, a new database of reference alignments is under development. This aims to provide more examples in the low PID ranges as well as include a greater variety of sequence alignment problems (e.g. simultaneous alignment of multiple dissimilar domains).

8 Installation

8.1 Introduction

The OXBench package has been successfully built and tested on the following Linux platforms:

- Red Hat 7.2 running on an Opteron processor (using GCC 3.3.3)
- Mandrake 10.1 running on a 32-bit Intel processor (using GCC 3.3.3 and 3.4.1)
- Red Hat Enterprise Edition 3AS running on an AMD64 processor (using GCC 3.2.3)

8.2 Requirements

The following packages are required to install OXBench:

- GCC v.3.0+ (3.3 preferred)
- bash
- GNU make
- GNU C Library
- PCRE (this can be download from http://www.pcre.org)
- Perl
- R (available from: http://www.r-project.org)

- GNU gengetopt (available from http://www.gnu.org/software/gengetopt/gengetopt.html)
- Doxygen (available from http://www.doxygen.org)

Gengetopt and Doxygen are needed only if you are doing development work on the OXBench source code. They are not required to run the metrics included in the package.

8.3 Building the installation

The installation is built using make.

1. Run `'make install'` to build and install the executables. Executables are installed into the bin directory of the distribution.

2. Run `'make subsets'` to create required symbolic links in the data directories.

3. Set the OXBENCH environment variable to the path of the OXBench installation and add the $OXBENCH/bin/ directory to your $PATH.

4. Run `'make test'` to test the executables. This runs each of the metrics on a sample alignment and compares the CSV output to a reference CSV file. If all is well, the output will look like this:

```
/sw/oxbench> make test
->   testing column... PASSED
->   testing dep_acc_avg... PASSED
->   testing stamp... PASSED
```

The STAMP metric test is known to give a fail message on the Red Hat Enterprise/AMD64 platform owing to a small difference in precision in the output on 64-bit machines and 32-bit machines. The test fails with the following output:

```
/sw/oxbench> make test
->   testing column... PASSED
->   testing dep_acc_avg... PASSED
->   testing stamp...122c122
< default,1ann-4-AUTO.1,1axn-2-AUTO.1,stamp,1,1.87,8.58,98.00,73,1,1,74,74
---
> default,1ann-4-AUTO.1,1axn-2-AUTO.1,stamp,1,1.87,8.59,98.00,73,1,1,74,74
 FAILED
make: *** [test] Error 1
```

5. (optional) Although not used in the standard OXBench metrics, a complete set of pairwise structural alignments may also be installed. Since the number of pairwise alignment files is very large, they are included as a tarball and can be installed as follows:

   ```
   make pairwise
   ```

The Makefile also includes the following targets that are useful for developers:

- make docs - updates the source code documentation generated using Doxygen. This is only required if the source code has been modified.
- make profile - generates a profile optimized version of the framework using the test suite as a training set.