SEGUID and GCG Checksum: New checksum algorithms for Biopython

Sebastián Bassi
sbassi@clubdelarazon.org
Virginia Gonzalez
vgonzalez@unq.edu.ar

Universidad Nacional de Quilmes, Argentina

What is a checksum?

"A checksum is a form of redundancy check, a simple way to protect the integrity of data by detecting errors in data that are sent through space (telecommunications) or time (storage). It works by adding up the basic components of a message, typically the asserted bits, and storing the resulting value. Anyone can later perform the same operation on the data, compare the result to the authentic checksum, and (assuming that the sums match) conclude that the message was probably not corrupted."

Source: Wikipedia contributors. Checksum. Wikipedia, The Free Encyclopedia. June 7, 2007, 20:30 UTC. Available at: http://en.wikipedia.org/w/index.php?title=Checksum&oldid=136678031. Accessed June 24, 2007.

Why using a checksum for biological sequences?

There are different reasons:

- Data integrity validation: To be sure you are dealing with the same sequence after extensive manipulation or retrieving from multiples sources.
- ID genes, proteins and splicing variants: To give an unique ID of each sequence.

Most used checksum algorithms in Biology

CRC64: Proteins in Uniprot.
GCG-Checksum: DNA and Protein sequences in the file format of GCG and compatible programs.
SEGUID: "A SEquence Globally Unique IDentifier" Proteome Database

CRC64

Used by Uniprot.
Provides "only" 1.8×10^{19} different possibilities.
There are collisions:

>immunoglobulin lambda light chain variable region [Homo sapiens]
QSALTQPASVSGSPGQSITISCTGTSSDVGSYNLVSWYQQHPGKAPKLMIYEGSKRPSGVSNRF
SGSKSGNTASLTISGLQAEDEADYYCSSYAGSSTLVFGGGTKLTVL

>immunoglobulin lambda light chain variable region [Homo sapiens]
QSALTQPASVSGSPGQSITISCTGTSSDVGSYNLVSWYQQHPGKAPKLMIYEGSKRPSGVSNRF
SGSKSGNTASLTISGLQAEDEADYYCCSYAGSSTWVFGGGTKLTVL

Both sequences share the same CRC64 checksum: 44CAAD88706CC153

Already implented in BioPerl and BioPython

GCG-Checksum

Used by GCG software suite (and compatible programs).
Very low number of combinations: 10^4.
Used by several sequence file formats.
BioPerl implementation available.

Introducing BioPython implementation (based on BioPerl version).

GCG-Checksum: Python Code

Main program

```
def gcg(seq):
    import sys
    if sys.version_info >= (2,4):
        import gcg24
        return gcg24.gcg(seq)
    else:
        #slower version for Python 2.3
        index = checksum = 0
        if type(seq)!=type("aa"):
            seq=seq.tostring().upper()
        else:
            seq=seq.upper()
        for char in seq.upper():
            index += 1
            checksum += index * ord(char)
            if index == 57: index = 0
        return checksum % 10000
```

gcg24.py

```
def gcg(seq):
    from itertools import cycle, izip
    return sum(n*ord(c.upper()) for (n,c) in\
            izip(cycle(range(1,58)),seq)) % 10000
```

gcg24.py is a module that is imported from the main program when Python version is ≥2.4. It can't be included in main program because this code can't be parsed under Python 2.3

SEGUID: The problem

There are many sequence databases. Each one has its own ID and is not easy to make cross-references between them.

Database Name	Identifier Syntax
NCBI: (Release 152)	
GenBank	gb|accession|locus
EMBL Data Library	emb|accession|locus
DDBJ, DNA Database of Japan	dbj|accession|locus
NBRF PIR	pir||entry
Protein Research Foundation	prf||name
SWISS-PROT	sp|accession|entry name
Brookhaven Protein Data Bank	pdb|entry|chain
Patents	pat|country|number
GenInfo Backbone Id	bbs|number
General database identifier	gnl|database|identifier
NCBI Reference Sequence	ref|accession|locus
Local Sequence identifier	lcl|identifier

SEGUID: Proposed solution

"We propose the use of a unique sequence identifier (SEGUID) that is **derived from the primary sequence itself and easily generated by any user**. SEGUIDs are resilient to changes in public and private databases as they remain constant throughout the lifetime of a given protein sequence. The SEGUID Proteome Database (http://bioinformatics.anl.gov/seguid/) provides aliases for the annotated entries available from several public databases and can be downloaded or generated easily at remote sites. SEGUIDs have been used in our proteomics laboratory for years and proved to be useful integrating mass spectrometry results, two-dimensional gel electrophoresis data, and bioinformatics information"

Source: SEGUID: Overview.
http://bioinformatics.anl.gov/seguid/overview.aspx
For more information: http://dx.doi.org/10.1002/pmic.200600032

SEGUID: Code

```
def seguid(seq):
    try:
        import hashlib
        m = hashlib.sha1()
    except:
        import sha
        m = sha.new()
    import base64
    if type(seq)!=type("aa"):
        seq=seq.tostring().upper()
    else:
        seq=seq.upper()
    m.update(seq)
    try:
        return base64.b64encode(m.digest()).rstrip("=")
    except:
        import os
        return base64.encodestring(m.digest())\
               .replace(os.linesep,"").rstrip("=")
```

SEGUID Application: Check for matchs from two FASTA files.


```
from Bio import SeqIO
seq1=set()
handle=open("FILE1","r")
for record in SeqIO.parse(handle,"fasta"):
    seq1.add(seguid(record.seq))
handle.close()
seq2=set()
handle=open("FILE2","r")
for record in SeqIO.parse(handle,"fasta"):
    seq2.add(seguid(record.seq))
handle.close()
shared_elements=seq1.intersection(seq2)
handle=open("FILE1","r")
for record in SeqIO.parse(handle,"fasta"):
    if seguid(record.seq) in shared_elements:
        print record.id
handle.close()
```

Code availability

All code in this presentation was submitted to Biopython project (Bug #2323), and may be available in next Biopython version (1.44?)

Biopython is freely available from Biopython.org under a very permissive license.