Relax with CouchDB — Into the non-relational DBMS era of bioinformatics

Ganiraju Manyama,1, Michelle A. Paytona,b,1, Jack A. Rothc, Lynne V. Abruzzob, Kevin R. Coombesa,⁎

a Department of Bioinformatics and Computational Biology, The University of Texas MD Anderson Cancer Center, Houston, TX 77030, USA
b Department of Hematopathology, The University of Texas MD Anderson Cancer Center, Houston, TX 77030, USA
c Department of Thoracic and Cardiovascular Surgery, The University of Texas MD Anderson Cancer Center, Houston, TX 77030, USA

Available online 17 May 2012

Article info
Article history:
Received 2 March 2012
Accepted 10 May 2012
© 2012 Elsevier Inc. All rights reserved.

Abstract

With the proliferation of high-throughput technologies, genome-level data analysis has become common in molecular biology. Bioinformaticians are developing extensive resources to annotate and mine biological features from high-throughput data. The underlying database management systems for most bioinformatics software are based on a relational model. Modern non-relational databases offer an alternative that has flexibility, scalability, and a non-rigid design schema. Moreover, with an accelerated development pace, non-relational databases like CouchDB can be ideal tools to construct bioinformatics utilities. We describe CouchDB by presenting three new bioinformatics resources: (a) geneSmash, which collates data from bioinformatics resources and provides automated gene-centric annotations, (b) drugBase, a database of drug-target interactions with a web interface powered by geneSmash, and (c) HapMap-CN, which provides a web interface to query copy number variations from three SNP-chip HapMap datasets. In addition to the web sites, all three systems can be accessed programmatically via web services.

1. Introduction

With the proliferation of high-throughput technologies that provide genome-wide overviews of the molecular landscape within cells, researchers can quickly profile the whole genome using multiple modalities. For example, The Cancer Genome Atlas (TCGA) is a comprehensive effort to accelerate our understanding of cancer by profiling hundreds of samples of different types of cancer using many technologies, including gene expression microarrays, methylation arrays, microRNA arrays, array CGH, SNP chips, and large scale genomic sequencing [1]. The bioinformatics challenge of integrating these disparate sources of information remains difficult. The bioinformatics community continues to develop extensive resources to annotate and mine biological and genomic features from these high-throughput analyses. In this context, database management systems (DBMSs) are required to store the source data and the analysis results, and to build tools that annotate and mine the data.

The underlying DBMS for most current bioinformatics utilities are based on a relational model [2]. Relational database management systems (RDBMSs) provide excellent data integrity at the cost of fragmenting data into multiple tables. There might be issues associated with scalability and performance, especially during data retrieval, although these issues rarely come into play unless the database expands across multiple server nodes. Non-relational or NoSQL (not only SQL) database systems that do not require specific design schemas are used in such cases for high-throughput. Although NoSQL systems may not provide strict data consistency like RDBMS, they are highly scalable [3]. They are increasingly used for giant cloud computing services by commercial establishments like Google and Amazon [4]. Their close integration with web servers and standard protocols can, under the right circumstances, facilitate rapid development of interactive database applications; the absence of a rigid design schema also makes it easier to change designs while continuing to use and access the existing data. Some bioinformatics systems have already been developed using non-relational database systems like Hbase [5] and Persevere [6].

CouchDB is an open-source non-relational database system being developed by the Apache Software Foundation (http://couchdb.apache.org). Here, we empirically demonstrate the utility of CouchDB as a database management system for bioinformatics by building several web-based database utilities, namely (a) geneSmash, (b) drugBase, and (c) HapMap-CN. The geneSmash database collates data from various bioinformatics resources and provides automated gene-centric annotations. The drugBase system is a CouchDB database of drug-target interactions with a web interface powered by geneSmash. The HapMap-CN database provides an interface to query the copy number variations identified using the HapMap [7,8] SNP chip dataset using gene information or genomic location. In addition to the web sites, all three systems can be accessed programmatically via web services for high-throughput analyses.
A

CouchDB database

Documents

_key

_pre-computed_view(s)

key

CouchDB web server

R client

HTTP

JSON

Browser client

AJAX

JavaScript

B

Source data files

perl scripts

Subversion Repository

_attachments

(HTML, CSS, JavaScript, etc)

_views

(JavaScript source)

CouchApp

CouchDB database

CouchApp

Perl scripts

Source data files

JSON documents

CouchDB Database

Design document

C

Web Services

drugBase database

geneSmash database

HapMap-CN database

Web Sites

drugBase web site

geneSmash web site

HapMap-CN web site

User Actions

Query drugs by targeted gene

Query gene information

Query gene location by symbol

Query copy number data by location
2. Results

Fig. 1A provides an overview of the CouchDB architecture. From the perspective of client applications, a CouchDB database looks like a web server. Clients make HTTP calls and get responses formatted as HTML or JSON documents. Internally, the database contains data documents, design documents, and pre-computed query results that can be accessed using URLs. Fig. 1B illustrates a typical development environment for CouchDB databases and applications. Although CouchDB allows views (or queries) to be defined in various languages, JavaScript is the most common choice. The source code for views is contained in the design document, and can therefore be extracted from the database for closer study.

2.1. geneSmash

As noted above, large-scale collaborative projects like TCGA are generating genome-wide data from multiple platforms. A fundamental need is to integrate these data, which always requires matching probes across platforms, either by the genes they target or by their genomic coordinates. Moreover, it would be extremely useful if the genomic coordinates were directly available to the programming language or scripts used to analyze the data. A number of attempts to address this problem have already been implemented. We start by reviewing the important tools previously developed in this domain.

BioMart provides general tools for serving biological data on the Internet. Their solution is based on a federated architecture, where each data supplier is responsible for exposing their own relational database through the BioMart interface [5]. Programmatic access to BioMart requires the user to learn a new application programming interface; usage in this form is limited to a small number of programming languages that have implemented the API. Alternatively, access to BioMart can occur through HTTP calls using the Mart Query Language (MQL), which is inspired by the Structured Query Language (SQL) used to query relational databases [10]. MQL supports extremely general queries, but requires the user to understand the XML format of queries as well as the table structure of individual marts. BioMart does support more complex queries than geneSmash, but exploiting that power requires learning a new vocabulary, the MQL. For researchers who want to integrate disparate kinds of data and annotations, the BioMart approach puts the burden of integration on the client, with a difficult learning curve.

GeneCards is a comprehensive compilation of annotative information about human genes [11]. The gene-centric content is integrated from over 80 digital sources resulting in extensive genomic, proteomic, transcriptomic, disease, and functional data. The integrated, relational database is available as plain text, XML files, and MySQL dumps. GeneCards provides a sophisticated search engine and two tools for working with sets of genes: GeneAlaCart and GeneDecks [11,12]. GeneAlaCart accepts a list of gene identifiers along with the user’s selected data fields and produces a file of annotations for the gene set. GeneDecks allows users to search for functional paralogs and to find annotations shared by a group of genes. Overall, GeneCards restricts the types of queries users can conduct and makes it difficult for users to integrate their own data with the GeneCards database.

BioDAS (Distributed Annotation System) is a communication protocol used to exchange annotations. It is based on the concept that annotations should not be provided by centralized databases, but should be spread over multiple sites [13]. BioDAS is a client-server system where a single client can integrate annotation data from multiple distant web sites and display it to the user in a single view. The communication between client and servers is defined by DAS XML specification. The system has its advantages: data providers maintain control over data, data is free from problems associated with release cycles, and data duplication is avoided. BioDAS also has some disadvantages: a lack of enforced semantics limits applications to visual displays and data sources must be DAS-enabled [14].

2.1.1. geneSmash application details

The geneSmash database integrates data from the NCBI Entrez Gene database [15], the UCSC genome site [16], miRBase [17-20], and multiple microarray manufacturer web sites [21-23]. The NCBI Entrez Gene identifier is used as the primary ID; any record without this identifier is discarded while populating the geneSmash database.

At present, geneSmash contains data only on the Homo sapiens genome. However, nothing about the design requires this restriction. Supplementary Table S1 explains the structure of the fields in a typical geneSmash document. The source of each field is recorded, along with an indication of which fields can be queried.

Users can query geneSmash with a variety of gene identifiers through their corresponding views. As described in Materials and methods, the names and the source code for the views are contained in the design document, available at http://app1.bioinformatics.mdanderson.org/genesmash/_design/basic. As of this writing, geneSmash annotations can be queried by Entrez Gene identifier, HGNC symbol (by_symbol), Ensembl identifier (by_ensembl), UniGene cluster (by_unigene), cytotband location (by_cytoband), by probe (by Probe and get_probes) and genomic coordinates (by_location and gene_location). Probe annotation information is available for common Affymetrix, Illumina, and Agilent gene expression microarray platforms. Genomic coordinates are available for three different builds of the human genome (NCBI v3.5 through v.37, or UCSC hg17 through hg19). Locations of microRNAs are included for four releases of miRBase (from v.13 through v.16).

Because geneSmash provides a web service, and not simply a web site displaying the underlying database, it can serve as a gene-centric platform on which other applications are constructed. We have already developed several such applications including drugBase and HapMap-CN. Fig. 1C explains how the web site interfaces to drugBase or HapMap-CN use geneSmash as a web service to produce a better user experience.

2.2. drugBase

When conducting high-throughput studies of disease, biologists and clinicians frequently find themselves confronted by long lists of “interesting” genes; the canonical example is a set of differentially
expressed genes between cohorts who respond differently to a certain treatment. When this happens, a frequent goal is to identify other drugs that might benefit the patients who respond poorly to the initial treatment or who have an adverse reaction to a drug or a negative drug–drug interaction. Exploratory molecular analysis on the mechanism of action of drugs with biomolecules was initiated in the final decades of the 20th century. In the past, the drug interaction information summarized from these scientific studies was limited to journals and commercial catalogs. Online public depository of existing drugs and their target information were initiated by the Therapeutic Target Database (TTD) in 2002 [24,25]. Over the decade, many other resources like DrugBank [26,27], PDTD [28], STITCH [29,30] and SuperTarget [31] were developed to store information about drug–target interactions.

The web portals supported by the existing drug–target databases provide graphical web interfaces to manually access and search the data. However, for a comprehensive high-throughput analysis one usually needs to download the entire database, since the existing interfaces do not support batch queries of the drug–target interaction data. It would be useful to have a tool that supports batch queries using a list of genes to retrieve drug–target information that could be integrated into analyses of high-throughput genomic data.

2.2.1. drugBase application details

The drugBase database contains drug–target interactions, which can be batch-queried by gene target. The primary source of information for drugBase is Matador, a manually curated version of the SuperTarget database of drug–target interactions [31]. Users can query drugBase with a gene identifier to extract all the drugs targeting the protein associated with that specific gene. Sample code using perl or R to access drugBase for batch queries is provided in the online documentation. Supplementary Table S2 explains the structure of fields in a typical drugBase document.

The database can be manually queried with a variety of gene identifiers from the web site. The web site uses HTML and JavaScript based on the CouchDB JSON API. Behind the scenes, the web interface uses geneSmash to make the application more user-friendly (Fig. 1C). The chromosomal location is then determined to the gene location is performed using the geneSmash web service (Fig. 1C). The chromosomal location is then used to query the HapMap-CN database in order to identify all segments that overlap that location. Results are returned in an HTML table that can be sorted by sample name, copy number, or the start or end of the segment in the individual sample.

Our copy number analysis of the HapMap also identified numerous regions where the copy number varies from individual-to-individual; the frequency with which these changes occurred often differed between ethnic groups. One example of a highly variable region occurs on chromosome 22 in the vicinity of the PRAME gene, whose loss has previously been noticed in both chronic lymphocytic leukemia [33] and mantle cell lymphoma [34]. Using the genomic location of PRAME to query the 225 individuals in the HapMap-CN database, we found 3 individuals with homozygous deletions, 15 with heterozygous deletions, and 9 with gains in this region.

2.3.2. Database size

Table 1 provides an overview of the size of these three databases. Each database contains only one kind of document, whose structure is explained in Supplementary Tables S1–3. More complicated designs can contain many different kinds of documents; the JavaScript code to define a view typically relies on either the existence or the value of a particular field to determine the document type and to decide if that document is an appropriate response to a query. At present, the drugBase (by EntrezGene) and HapMap-CN (by location) databases each contain only one pre-defined view. The geneSmash database, by contrast, has defined 11 views; since the values are precomputed for every view, geneSmash occupies much more disk space. At present, all three databases are synchronized quarterly with updates from their external sources.

3. Discussion

CouchDB is a non-relational, document-oriented, schema-free database with a built-in web server. Database queries are processed through HTTP requests, which are handled by the RESTful JSON API. This feature provides universal accessibility to any modern programming language without any customized API. CouchDB also provides native support for incremental database replication, which makes it a very convenient tool to maintain mirror copies of data. The reliance of CouchDB on standard protocols (HTTP and JSON) that are fully integrated into web

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size of CouchDB databases.</strong></td>
</tr>
<tr>
<td>Database</td>
</tr>
<tr>
<td>geneSmash</td>
</tr>
<tr>
<td>drugBase</td>
</tr>
<tr>
<td>HapMap-CN</td>
</tr>
</tbody>
</table>
browsers promotes rapid, incremental development of database applications. We have illustrated the power of CouchDB by presenting three specific bioinformatics applications: geneSmash, drugBase and HapMap-CN.

CouchDB makes it easy to replicate databases. We have taken advantage of replication during database development and maintenance. The development version of the database runs on the developer's machine; because CouchDB is fairly lightweight, this can be a fairly modest laptop. The development version of the database is then replicated to a test environment that matches the intended production environment. After testing is completed, the database is replicated to the production server. For small databases like the ones described here, it is feasible to replicate the entire database onto the computers used by statistical and bioinformatics analysts, giving them immediate local access to the data.

We must point out that the HapMap–CN application illustrates one potential challenge when using CouchDB for genomic data. CouchDB stores all documents in a B-tree structure that mathematically represents a complete (linear) ordering of the stored documents. Because it uses this structure, the CouchDB API is optimized for queries that return a consecutive range of documents (i.e., those between startkey and endkey; see Materials and methods). In the HapMap-CN database, each document represents a genomic segment. The fundamental key that describes a segment consists of a chromosomal starting and ending location. Queries into HapMap-CN database also take the form of segments, and the goal is to retrieve all segments in the database that overlap the query segment. One can show mathematically that it is impossible to impose a complete ordering on the set of all sub-segments that will ensure that the response to every query is a consecutive range of segments (or documents). Because the DBMS is currently not able to provide such functionality, HapMap-CN solves the problem by embedding logic in the front-end JavaScript application to restrict the results of a query to the desired segments. The open source CouchDB development community has faced similar challenges when trying to handle geographic data, which is also intrinsically two-dimensional. They responded by developing a modified server, GeoCouch [35], which uses a different underlying data store optimized for spatial queries. If CouchDB is adopted by enough genomics researchers, a similar alteration could be made to the open source code to develop a server that is more highly optimized for segment-based queries.

Relational (RDBMS) databases and NoSQL databases have their own strengths and weaknesses, and are best suited for different kinds of applications. RDBMS represents a mature technology that has been optimized over several decades. RDBMS systems have extensive support for transactions, which they use to ensure data consistency and data integrity. With properly normalized tables, there is a unique source for each piece of data stored in the database. RDBMSs also have a flexible and powerful query language (SQL) that supports ad hoc queries. By contrast, CouchDB is much less mature, with limited support for transactions. The same piece of information may be repeated across many documents, which presents potential difficulties with maintenance if one of those pieces of information must be updated. Finally, CouchDB does not support ad hoc queries; all views into the database must be predefined in the design documents. If any of these features are critical to an application, then an RDBMS should be preferred over CouchDB.

There are, however, many potential database applications in genomics that do not require transactions, normalized tables, or ad hoc queries; the three applications we present here provide examples. In general, the tradeoff for normalization is a lack of flexibility: it is hard to change things when a piece of data comes along that does not fit into the schema. Our earliest version of geneSmash simply re-capitulated the human gene data from Entrez Gene. While developing HapMap-CN, we recognized that it would be convenient for users to search for genes in a genomic region of interest. We then added the genomic mapping data from the UCSC Genome Browser into geneSmash, along with some new views. We later decided that it would be useful when analyzing microarray data to be able to query genes by the probe identifiers defined by different microarray manufacturers. So, we also added these data to the existing gene documents and defined additional views. In both cases, none of the existing views had to change; and none of our editing code broke.

We believe that the tight integration of CouchDB with web standards provides two advantages. First, client-side web applications talk directly to Couch without the need for a server-side middle layer, significantly reducing development time. These applications rely on Asynchronous JavaScript and XML (AJAX) methods. AJAX plays a central role in much current web development, and, as a result, there are extensive open source libraries that make it easy to develop web sites that allow users to interact directly with the data. We use the implementation of AJAX provided by the jQuery JavaScript library (http://jquery.com/). The sortable table in the HapMap-CN application is implemented using a jQuery plug-in called TableSorter (http://tablesor.com/docs/).

We are currently developing additional applications that rely on a jQuery plug-in called flot (http://code.google.com/p/flot/) to create interactive graphics in a web browser using data stored in CouchDB.

The second advantage arising from CouchDB’s use of web standards is that every CouchDB application provides a web service with a RESTful interface, and not just a web site. Now, it is undoubtedly true that there are many experienced database programmers and database administrators who know how to use SQL to get data out of RDBMS databases and convert it into the format needed to perform analyses. Many bioinformatics and statistical analysts, however, do not know how to do this. In our experience, we have found that they can easily and quickly learn to use the HTTP/JSON API of CouchDB to get data into the systems that they use to perform their analyses. For example, once they know the URL for a relevant query, they can get the results of that query into the R statistical programming environment in only three lines of code:

```r
library(RJSONIO)
tempstring <-- paste(readLines(URL), collapse=")
results <- fromJSON(tempstring)
```

The “results” object that this code produces is a native R object that statistical analysts can examine and then use in their analyses directly.

One potential advantage that is often promoted for NoSQL databases is “horizontal scalability”. The performance of RDBMS databases scales well as long as all of the data can be stored on a single server. Performance can be improved by using multiple servers, provided each one has access to a complete copy of the data. However, if the data grows large enough that it needs to be split over multiple server nodes, RDBMS performance can degrade quickly; relational table joins across servers are complicated and expensive. The NoSQL (and CouchDB) solution is to replace table joins with queries that use the map/reduce paradigm, which is an inherently parallel procedure. Major web presences like Google, Amazon, Facebook, and Twitter use NoSQL databases to provide some of their services. The three databases described in this manuscript are far too small to benefit from this ability. We speculate, however, that the large datasets being generated by next generation sequencing technologies could potentially benefit from the horizontal scalability of NoSQL databases.

4. Materials and methods

4.1. CouchDB

CouchDB is a document-oriented, schema-free database with a built-in web server. Database queries are processed through HTTP requests, which are handled by a RESTful application programming interface (API) [36]. All interaction with the database uses only RESTful verbs (GET, PUT, POST, DELETE) that are defined by the HTTP standard.
Database responses are returned in JavaScript Object Notation (JSON). It is possible to define specialized queries in a CouchDB application that return alternative formats in addition to JSON; the geneSmash, drugBase, and HapMap-CN web sites use this facility to return HTML-formatted responses for display in a web page. The CouchDB API uses only protocols that are defined by open web standards. Since the systems we developed use the same protocols, their web services are immediately accessible from any modern programming language without having to learn a new specialized query language.

The possible queries that can be made to these web services are self-documenting; this feature is a direct consequence of the design of CouchDB. The developer of a CouchDB database application defines the named views (the CouchDB term for queries) that can be applied to the data. Each named view produces key-value pairs from appropriate documents. The generic CouchDB API defines parameters _key, startkey, and endkey that allow the user to restrict the set of documents returned when querying a view. Moreover, each named view must be defined as part of a design document that is stored in the database. Every CouchDB document has a unique identifier; design documents are characterized by an identifier that starts with the reserved word _design. Thus, you can obtain a list of all design documents in a Couch database called "db" by sending an HTTP GET request of the form GET/db/_all_docs?startkey="_design"&endkey="_design0".

You can then send another GET request for each design document and read the resulting JSON value to obtain a complete list of the views that it defines.

CouchDB relies on a map/reduce paradigm that allows it to precompute the responses to all views defined in all design documents. Technically, it computes all responses when the first query is made after a view is defined or modified. When new data documents are inserted, it only computes the responses for the new or updated documents. The initial indexing step can be quite time-consuming, depending on the number of documents in the database and the number and complexity of the views that are defined in the design document. After indexing has been completed, responses are extremely fast. For example, retrieving a complete list of the 50,000 symbols known to geneSmash takes about 10 s; retrieving the complete gene documents for all 50,000 symbols takes about 30 s. How quickly you can use this information in a program, however, depends on the quality of the implementation of the programming library that converts from the JSON format of the response into the native format for that language. The (highly optimized) RJSONIO package [37] for the R statistical software environment converts all 50,000 gene documents into an R object in less than 30 s.

CouchDB provides native support for database replication. The replication function allows any database to be copied from one CouchDB server to another. Replication works on the granularity of individual documents and is incremental. That is, only documents that have been inserted or modified since the previous replication are replicated on a later request. As a consequence, creation and maintenance of mirror copies of databases as CouchDB instances is easy, reliable, and robust.

4.2. geneSmash

geneSmash integrates data from existing biological annotation data repositories. The basic gene and cross-database identifier information was obtained from the NCBI Entrez Gene database [15]. The gene locus information for multiple human genome builds was acquired from the UCSC genome browser web site [16]. miRNA information was obtained from multiple versions of miRBase [17-20]. Probe annotation information for a repertoire of gene expression arrays was obtained from the corresponding manufacturer's web sites [21-23]. The geneSmash web site can be accessed at http://app1.bioinformatics.mdanderson.org/genesmash/_design/basic/index.html or the web service can be queried directly at http://app1.bioinformatics.mdanderson.org/genesmash using the full CouchDB API.

4.3. drugBase

drugBase is a database of drug–target interactions, which can be batch-query by gene target. The primary source of information for drugBase is Matador, a manually curated version of the SuperTarget database of drug–target interactions [31]. Potential drug–target relationships in SuperTarget were extracted from the literature using PubMed, Medline, and Medical Subject Heading (MeSH) terms [38]. Drug–target interactions with supporting PubMed literature in the existing databases DrugBank [26,39], TTD [24,25], KEGG [40], PDB [41,42] and SuperLigands [43] were also present in SuperTarget. The protein identifiers used in Matador were defined based on an older version of the String (v7.1) protein interaction database [44]. We mapped these identifiers onto the corresponding Entrez Gene identifiers. In instances where they could not be mapped directly, we used HGNC gene symbols and NCBI RefSeq identifiers. Because drugBase is designed as a gene-centered drug–target interaction database, the drug–target interactions defined solely based on MeSH terms without any specific gene or protein in Matador were not included in drugBase. The web site can be accessed at http://app1.bioinformatics.mdanderson.org/drugbase/_design/basic/index.html or the web service can be queried directly at http://app1.bioinformatics.mdanderson.org/drugbase/ using the full CouchDB API.

4.4. HapMap-CN database

The HapMap-CN database provides a tool to query the results of a copy number analysis of SNP chip data of 225 HapMap samples. The raw data were downloaded as three data sets (GSE17205, GSE17206, and GSE17207) from the Gene Expression Omnibus [32]. These datasets from the HapMap project were acquired using Illumina 610K Quad v1 chips containing measurements on 225 normal samples from different ethnic groups. We processed the raw data in Illumina GenomStudio to compute genotype calls, log R ratios (LRR), and B allele frequencies (BAF) for each SNP in each sample. We applied the circular binary segmentation (CBS) algorithm to LRR and BAF values for each chromosome of every sample to derive copy-number segments [45]. We used the CBS implementation from the R package DNAcopy to record the segmentation results [46]. Loss of heterozygosity (LOH) was also estimated for every chromosome in all the samples. The algorithm computes the odds ratio of LOH vs. no LOH by considering the probability of seeing (almost) all consecutive homozygous SNPs in a string of 40 consecutive informative SNPs. The segment data from multiple algorithms was then summarized for each sample along each segment. Complete Sweave source code detailing the copy number analysis of the HapMap samples is available online at http://bioinformatics.mdanderson.org/HapMap/docs/index.html.

The web site can be accessed at http://app1.bioinformatics.mdanderson.org/hapmap/_design/basic/index.html or the web service can be queried directly at http://app1.bioinformatics.mdanderson.org/hapmap/ using the full CouchDB API.

Acknowledgments

This work was supported in part by grants R01 CA123252, P30 CA016672, and P50 CA070907 from the National Cancer Institute of the National Institutes of Health.

Appendix A. Supplementary data

Supplementary data to this article can be found online at http://dx.doi.org/10.1016/j.ygeno.2012.05.006.