Universidad Complutense de Madrid

Facultad de Informática

Departamento de Ingeniería del Software e Inteligencia Artificial


# Métodos y Aplicaciones para el Análisis Funcional en Bioinformática

(Functional Analysis Methods and Applications in Bioinformatics)

*Memoria para optar al grado de doctor presentada por:*

Miguel Vázquez García

*Dirigida por los doctores:*

Alberto Pascual Montano y
Juan Pavón Mestras

Mayo 2010

# Acknowledgements

I would like to start by thanking my PhD. advisors Alberto Pascual Montano and Juan Pavón Mestras.

Alberto introduced me to this topic of bioinformatics, which gave me the perfect subject to apply machine learning, my long-time passion. In very little time bioinformatics and molecular biology became a major interest for me, so I owe him the chance of having a job I really love.

None of this would have happened, though, if it wasn't for Juan, who offered me a grant just as I got back from Texas a long time ago, and helped me tremendously in my academic career. For his advice, but most importantly, for his concern, I will always be grateful.

He does not figure as an advisor, for unfortunate administrative reasons, but Pedro Carmona Saez paid a very important role in this thesis. Working with him in MARQ was a great experience. I had the chance to learn a lot from him, and his down-to-earth advice kept avoiding my getting distracted. Creating MARQ was one of the highlights of my career, which I will always remember very fondly. Thank you very much, friend.

I would also like to thank a great deal of people with whom I have worked over the years. Of course, Rubén; we had a lot of fun working together, and things always seemed to work! I hope we can do that again often in the future. With the rest of the bioinfo group, Mariana and Edgardo in particular, I have always gotten along very well –It has been a pleasure working with you guys!

I also have a sweet spot for my friends in the department. Our academic interests are apart, but not so our life interests; I feel I share a lot with you. I will always remember the years we shared office 411.

There are some other people that I consider no less than landmarks in my academic career. From my time in Texas: Inderjit Dhillon, who showed me

just how incredibly cool data mining is, and Robert van de Geijn who was always a good friend, and perhaps one of the best teachers I've ever had. If nothing else, I would have learnt from Robert how to use a real text editor; he used Emacs, I use Vim now, and I would not be the programmer I am without it. I would like to thank both, and also Dan Boley, who invited me to Minnesota a few years ago; likewise an interesting experience.

Last but not least, my family. My mother Mariluz always had the energy to support me in all my endeavors, she was comforting when I was stressed, and encouraging when I most needed it. With her help, anyone can do a thesis. My father Juan Luis has always been a role model, in his personal life, but specially in his professional life. Actually, I learned from him that science is not only a job, it is a passion, the passion of learning. I would count myself happy if I achieved half of what he has in life. And finally my sister Isabel, one of the most unique and fascinating people I know. She is so well read! Thank you for all the proof reading and the fascinating conversations on style. To all of you, my dear family, I love you very much.

# About this document

## Summary

Computational methods play a critical role in modern molecular biology. One reason for that is the vast amount of data being produced by high throughput experiments, which requires statistical and machine learning techniques to highlight interesting patterns. Another reason is the diversity of areas of study covered by molecular biology, all of which must be integrated to form a holistic understanding. It is very hard for a researcher to be up to date in every aspect that might relate to his line of work without powerful knowledge management tools. Bioinformatics thus emerges as a fundamental discipline to support molecular biology research.

In this work we present a series of contributions to the field of bioinformatics that concentrate on producing methods and applications to help researchers analyze data produced by high throughput technologies. We have also defined some software development practices for bioinformatics and produced a repository of software packages and development frameworks tailored to the needs of this particular field. The applications include data analysis methods for a variety of applications, such as information extraction from biomedical text, factorization methods for matrix analysis, content based retrieval of microarray experiments, or statistical enrichment analysis of concurrent annotations.

The work in the thesis is composed of several independent but closely related contributions, and presented as a binding of edited publications under section 4.4 of the *Normativa de desarrollo de los artículos 11, 12, 13 y 14 del Real Decreto 65/2005, del 21 de Enero, por el que se regulan los*

*estudios universitarios oficiales de postgrado de la Universidad Complutense*, approved by the Consejo de Gobierno as of June 13th 2005 and published in the BOUC as of July 5th 2005. Each article is relatively self contained, and presents either an application or a software package intended for application development.

The products of the thesis presented in this document are: six peer reviewed articles, four of them published in a journal of significant relevance (one of them still in press), and 11 open source projects hosted on a public source code repository, most of them for bioinformatics applications or projects directly concerning development of such applications.

The following is a summary of my personal contributions. I was the main developer of two applications: SENT (Semantic Features in Text), a literature mining tool for functional analysis; and MARQ (Microarray Rank Query), a microarray retrieval and meta-analysis tool. In particular, I was in charge of designing and implementing the statistical inference and learning methods, and of the development of the applications. I also contributed to the development of two other applications: GeneCodis, a co-annotation enrichment analysis tool, and BioNMF, a web tool for non-negative matrix factorization (NMF). For these two applications, I was in charge of the web service infrastructure and contributed to defining the applications overall design. I was also the main developer of two development frameworks for bioinformatics, Rbbt (Ruby bioinformatics toolkit), for developing reusable components of complex functionalities, and SimpleWS, a tool that simplifies the development of SOAP web services.

## Document Structure

This documents presents the work for this PhD. through a collection of edited publications which appeared in peer reviewed journals and conferences, accompanied by several chapters that serve as introduction to the topics and discussion of the work.

Chapter 2 offers a very general introduction to molecular biology. It reviews some of the concepts that would help put the work in context such as the gene expression microarray technology, and the state of the field of

bioinformatics.

The objectives for this thesis are introduced in chapter 3, while chapter 4 presents the relevant publications in their edited format. Each publication is preceded by a brief summary that tries to capture its specific motivation and achievements.

Chapter 5, elaborates on the work of the thesis and elicits some general threads that serve to connect the technical aspects of the different contributions.

Chapter 6 presents some conclusions along with the current lines of work and future plans.

Appendices A and B include a technical description of SimpleWS and a review of relevant open source projects produced during this thesis.

Chapter 1 includes a fairly literal translation to Spanish of chapters 2, 3, 5 and 6.

The bibliography at the end of this document contains the full collection of references from the papers and for the addendum chapters.

# Contents

# Chapter 1

# Resumen En Español

## 1.1 Introducción

### 1.1.1 Introducción a la Biología Molecular

#### Breve repaso histórico

Los mecanismos de la herencia, descritos en primera instancia por Mendel a mediados del siglo XIX, han sido explotados desde entonces en el cultivo de plantas y la cría de animales. Pero ha sido durante el siglo XX cuando la tecnología se desarrolla lo suficiente para empezar a descifrar como funcionan realmente estos mecanismos. Algunos de los hitos más importantes han sido el descubrimiento de la estructura de doble hélice de las moléculas de ADN, el ADN recombinante, el código genético y el Dogma Central de la biología molecular.

En particular, las importantes consecuencias del Dogma Central de la biología molecular tienen una especial relevancia en esta tesis. Tal y como describía Francis Crick en un artículo seminal (Crick 1970), el ADN se *transcribe* a ARN (denominado *ARN mensajero*), y el ARN se *traduce* en proteínas, que son las moléculas que desempeñan la gran mayoría de las funciones biológicas de la célula. A pesar de que pronto se descubrió que este esquema tenía ciertas excepciones, una consecuencia fundamental que se deriva de él es que es posible estimar los mecanismos biológicos que se están desarrollando en una célula a partir de la simple presencia de estas moléculas

Las últimas dos décadas han sido testigo del desarrollo de diversas técnicas para medir la cantidad de estas entidades (su *nivel de expresión*). Estos avances permiten a los investigadores realizar conexiones entre fenotipos y los mecanismos biológicos que los caracterizan, con inmediatas aplicaciones en medicina y agricultura. Durante la última década estas tecnologías han sido objeto de otra profunda revolución: los *métodos de alto rendimiento*, revolución facilitada por la axiomatización de los protocolos y la reducción de los costes. Algunos ejemplos importantes son los *microarrays de expresión génica* y las técnicas de *proteómica*.

Aún sabiendo de la existencia de gran parte de los genes del genoma humano (Venter et al. 2001*b*), todavía no conocemos la función que muchos de ellos desempeñan, así como las complejas interacciones que presentan. Técnicas como los microarrays de expresión génica o la *secuenciación de nueva generación* prometen grandes descubrimientos en los años venideros.

### Biología computacional y bioinformática

Los métodos de alto rendimiento han destacado la importancia de la biología computacional; la extremada complejidad de los sistemas que permiten la vida sólo puede ser afrontada con sofisticados mecanismos de gestión del conocimiento que sean capaces de lidiar con la ingente cantidad de información que está siendo producida constantemente por los grupos experimentales, que sufre además de un gran nivel de heterogeneidad. En consecuencia, la biología computacional se ha convertido en una disciplina de gran importancia, atrayendo el interés de áreas como la informática, la matemática aplicada, la estadística o el aprendizaje máquina. En particular, varios avances recientes en el área de estadística se han visto motivados por problemas de la biología molecular, que plantea retos a los métodos clásicos en términos de escala y complejidad de los datos; retos a menudo mayores que los que la estadística había enfrentado en el pasado.

La biología de sistemas es un término que ha cobrado gran importancia en los últimos años; se refiere al estudio de la biología de la célula desde una perspectiva holística, integrando información sobre los distintos niveles y facetas que forman parte de un sistema biológico: evolución, regulación génica, rutas

metabólicas, etc. La bioinformática, por lo tanto, juega un papel crucial en la biología de sistemas, ya que se ha convertido en el principal catalizador de esta integración; todas estas disciplinas solamente pueden verse conectadas a través de protocolos computacionales de tratamiento masivo de información como el que suponen las técnicas de alto rendimiento junto con los sistemas de tratamiento y gestión de información.

La definición del término bioinformática varía entre los distintos investigadores, generalmente sobre lo que debe o no incluirse bajo dicho término. En este trabajo consideraremos bajo este título todas las herramientas software que se usan para la investigación en biología molecular, el análisis y modelado de datos, el desarrollo de bases de datos y de formatos de codificación de la información, o cualquier otro desarrollo técnico que esté orientado a simplificar el examen de los datos biológicos.

El campo de la bioinformática ha recibido un gran impulso, con numerosas aplicaciones nuevas surgiendo año tras año. Aunque este ritmo de producción puede parecer satisfactorio, con los problemas siendo resueltos en plazos razonables, la disciplina aún carece de sofisticados protocolos y prácticas de desarrollo. La informática se ha involucrado en el área recientemente, y hay muchos aspectos del oficio que se beneficiarían de prácticas y armazones para el desarrollo de software; tal y como se ha visto en el campo del desarrollo web, con numerosos avances en lo que se ha dado en llamar *web 2.0*. De hecho, la Internet es una parte fundamental de cualquier disciplina de investigación, incluyendo la bioinformática, dado que es un elemento fundamental para facilitar los flujos de información entre los distintos repositorios de datos. Las tecnologías de Internet y de bases de datos están siendo objeto de una profunda revolución, y existe un gran potencial que aún debe ser aprovechado en bioinformática. La bioinformática, por lo tanto, tiene mucho que ganar de los avances en protocolos de comunicación, prácticas y armazones de desarrollo software, y estándares de gestión de la información.

### 1.1.2 Microarrays de Expresión Génica

Esta sección repasará algunos datos importantes sobre la tecnología de microarrays de expresión génica, que representa un gran ejemplo de una técnica

de alto rendimiento, y que muestra una estrecha relación con gran parte del trabajo realizado para esta tesis. Los microarrays de expresión génica se usan para estudiar muy diversas condiciones biológicas: mutaciones, enfermedades, efectos de drogas, etc.

Los microarrays de expresión génica son capaces de estimar los niveles de expresión de miles de genes simultáneamente. Se dice que un gen está expresado si se ve transcrito a mARN (ARN mensajero), la molécula que se encarga de transmitir la información genética contenida en el ADN hasta la maquinaria ribosomal que se encarga de fabricar las proteínas. De esta forma, la cantidad de mARN correspondiente a un gen proporciona una estimación aproximada de la cantidad de proteína que está siendo producida. Esta estimación cuantitativa, como se mencionó con anterioridad, nos permite obtener una "instantánea" de los mecanismos biológicos que están tomando parte en las muestras sujeto de estudio. Es importante, en cualquier caso, tener presente que el uso del mARN como indicativo de la actividad a nivel del proteoma no es sino una aproximación, dado que no es excepcional que estas moléculas no desemboquen en la fabricación de las correspondientes proteínas, o que sean ellas mismas capaces de algún tipo de función biológica por sí solas.

El resto de esta sección ofrecerá una breve revisión de las características fundamentales de la fabricación y análisis de microarrays de expresión génica. Una revisión más completa se puede encontrar en Schulze & Downward (2001), Allison et al. (2006) o Slonim & Yanai (2009).

### Tecnología de microarrays

La idea fundamental de la tecnología de microarrays de expresión génica es hacer uso de las propiedades de hibridación de las moléculas de ácido nucleico. Las bases de ácido nucleico, también conocidas como nucleótidos, tienden a formar enlaces covalentes con sus complementarias bajo las condiciones adecuadas. Al situar secuencias particulares de estos nucleótidos en posiciones concretas de un soporte sólido, estas secuencias, denominadas *sondas*, son capaces de reconocer la presencia de secuencias complementarias suspendidas en una solución. Estas secuencias complementarias se dice que hibridan con

las sondas. Dado que los enlaces covalentes que se forman entre las parejas de nucleótidos son de carácter débil, sólo cuando se forma una secuencia relativamente larga de enlaces la fuerza de unión es suficiente como para resistir el subsiguiente lavado. Para estimar la cantidad de sondas que hibridan con sus secuencias complementarias se añade una molécula fluorescente a estas últimas y se mide la emisión desde cada posición en el soporte físico.

Las dos tecnologías más populares para la fabricación de estos microarrays son las de cDNA y las de oligonucleótidos. Los primeros, los de cDNA, se basan en una colección de secuencias de ADN complementarias a genes conocidos. Estas secuencias son amplificadas mediante la técnica de PCR (del inglés *polymerase chain reaction*), que utiliza la maquinaria de replicación de una bacteria resistente a la temperatura para duplicar cadenas de ADN, y posteriormente situadas en posiciones concretas del soporte físico, generalmente cristal. Estas sondas se adhieren al cristal utilizando un brazo robot y un sistema similar al usado por las impresoras de inyección. Las sondas en cDNA son de entre 20 y 1000 bases de longitud, aunque lo habitual es que estén en torno a las 400 ó 500 bases. Este tipo de microarrays se suele diseñar por el propio equipo de investigación que desarrolla el experimento.

Los microarrays de oligonucleótidos, popularizados por la empresa Affymetrix, tienen las sondas sintetizadas directamente en el soporte utilizando una tecnología similar a la usada al imprimir circuitos. Mediante el uso de un reactivo sensible a la luz y una serie de máscaras, las bases se van añadiendo secuencialmente a todas las sondas hasta que alcanzan la longitud final. Las sondas de estos microarrays son generalmente más cortas que las de cDNA, en torno a las 125 bases. Los arrays de oligonucleótidos se distribuyen por la empresa fabricante (ie. Affymetrix), a diferencia de los de cDNA; estos contienen distribuciones de sondas estándar, y vienen acompañadas de las herramientas software necesarias para leer y analizar los datos. Dalma-Weiszhausz et al. (2006) contiene una revisión más completa de la tecnología de Affymetrix.

Mediante el uso de los microarrays de expresión génica se quiere medir la concentración que presentan las moléculas de mARN, el intermediario entre el ADN y la maquinaria que crea las proteínas. Dado que las moléculas de ARN no presentan las mismas propiedades de hibridación de las moléculas de

ADN, una vez extraído de las muestras el mARN, debe ser transcrito de nuevo a ADN utilizando la maquinaria de transcripción inversa, de ahí el nombre de *ADN complementario* (cDNA). Tras la elaboración de las moléculas de cDNA, el siguiente paso consiste en marcar esas moléculas para su posterior identificación. En el caso de los microarrays de cDNA, dichas moléculas se marcan con fluorescentes, típicamente Cy5 (rojo) y Cy3 (verde) para distinguir las procedentes de cada clase de muestras a comparar. Las muestras marcadas se lavan sobre la superficie del microarray para permitir la hibridación con las sondas.

La tecnología de microarrays se utiliza a menudo para comparar los niveles de expresión génica entre dos tejidos distintos, como pueden ser un tumor y un tejido sano. Los ratios entre los dos tejidos se conocen como los *ratios de expresión diferencial*. En cDNA se suelen lavar las muestras de interés y de referencia sobre el mismo chip, así que el ratio de emisión entre el Cy5 y el Cy3 para cada sonda representa el ratio de expresión diferencial. En los de oligonucleótidos en cambio cada muestra se lava contra un chip distinto.

### Análisis estadístico

Usamos el término *análisis estadístico* para referirnos a todo aquello relacionado con extraer los ratios de expresión diferencial y establecer su significancia. Este proceso suele involucrar normalizar los valores crudos leídos en el microarray, para corregir sesgos en los datos de carácter técnico, y realizar inferencias. Para aliviar el efecto de la variabilidad en los datos, es necesario usar replicados, ya sean técnicos (sondas repetidas en un microarray) o de muestrales (chips repetidos). Para la normalización existen diversas alternativas, véase Lim et al. (2007) para una comparativa. Los chips de Affymetrix, por ejemplo, ya vienen acompañados del software necesario para realizar este procesado.

Una vez que los valores han sido normalizados, se hace uso de una estadística para inferir patrones interesantes de expresión. Como ya mencionamos con anterioridad, algunas de las estadísticas más comunes son para medir los ratios de expresión diferencial. Uno de los primeros métodos usados consistía en considerar directamente el ratio entre los valores de expresión,

una vez hecho un promedio entre los distintos replicados. Un gen se consideraría significativamente sobre-expresado si tenia un ratio de 2 o más o inhibido si era de 0.5 o menos, por ejemplo.

Usar directamente los ratios sufre de una gran limitación al no tener en cuenta la variabilidad de los valores. Un gen en particular puede tener un ratio importante, pero si los replicados muestran una gran variabilidad, es posible que ese ratio no sea mas que fruto de azar. Este problema ha hecho que los investigadores ahora tiendan a preferir métodos con más solidez estadística que tengan en cuenta la variabilidad. El *test de Student* sería un ejemplo de dichos métodos, aunque las características del problema, la dimensión de los datos en particular, hacen que esta alternativa no sea la mejor opción; las mismas ideas han sido adaptadas en otros métodos como SAN (Significance Analysis of Microarrays, Tusher et al. (2001)) y eBayes (empirical Bayes, Efron & Tibshirani (2007)), aunque existen numerosas opciones.

Los métodos basados en estas pruebas estadísticas también tienen inconvenientes. Dada una muestra suficientemente amplia, una pequeña variación en los niveles de expresión puede resultar significativa en términos estadísticos, aunque no lo sea en términos biológicos.

Otro aspecto importante surge del hecho de que, al estar examinando miles de genes a la vez, donde para cada uno se realiza una prueba estadística, un porcentaje de ellos resultará significativo simplemente por puro azar. Éste es un problema denominado de *hipótesis múltiples*; problemas como los planteados por los microarrays de expresión génica han promovido el desarrollo de protocolos como el del *Ratio de Falsos Descubrimientos* (FDR) (Benjamini & Hochberg 1995, Verhoeven et al. 2005).

### Interpretación funcional

El análisis estadístico descrito anteriormente, cualquiera que sea el método concreto que se use, acaba por identificar una lista de genes que potencialmente están asociados a las condiciones bajo escrutinio. Hay una importante distinción entre los métodos de *descubrimiento*, como los que representa el análisis de expresión diferencial, y los de prueba de hipótesis predefinidas. La

distinción es sutil pero importante, ya que en este caso no se trata de probar si una hipótesis concreta es cierta o no, sino de descubrir patrones interesantes que puedan proponer nuevas hipótesis de trabajo. En consecuencia, el análisis de expresión suele ir seguido de un proceso de interpretación que tiene el objetivo de plantear una serie de conclusiones basadas en las particularidades de los genes que se han encontrado significativamente desregulados. Este paso se conoce como el *análisis funcional* o la *interpretación funcional*.

Una de las alternativas más comunes para la interpretación funcional incluye el uso de bases de datos de anotaciones, que asocian genes con términos de un vocabulario controlado. The Gene Ontology (Ashburner et al. 2000, Hill et al. 2008) incluye anotaciones como la función biológica que desempeña el gen, o su localización celular; KEGG (Kanehisa et al. 2008) indica las rutas metabólicas en las que se cree que participa el gen. Estas anotaciones tienen la ventaja de que han sido introducidas en las bases de datos mediante un proceso de revisión de la literatura, así que gozan de una cierta fiabilidad.

Adicionalmente, el hecho de que dichas anotaciones provengan de un vocabulario controlado y de que sean introducidas supervisadamente permite realizar pruebas estadísticas sobre su propia distribución. En particular, un método de análisis consiste en ver si en nuestra lista de genes en particular existe alguna anotación que esté especialmente representada, es decir, que sea más frecuente en nuestra lista de genes de lo que cabría esperar por azar. Este tipo de análisis recibe el nombre de *análisis de enriquecimiento de anotaciones*.

El uso de anotaciones también tiene sus inconvenientes. El proceso de lectura y revisión de la literatura necesario para realizar estas anotaciones es lento y costoso. Al ritmo en que los artículos son publicados es muy difícil que estas bases de datos se mantengan actualizadas con los nuevos descubrimientos. De hecho, estos procesos de revisión tienen tendencia a cubrir algunos organismos o procesos biológicos más que otros, introduciendo sesgos. Así mismo, el hecho de que las anotaciones se hagan con vocabularios controlados limita la expresividad que pueden alcanzar; si una característica de unos genes no se encuentra reflejada en el vocabulario, no podrá ser detectada por el análisis.

Una alternativa a las anotaciones supervisadas es volver a la fuente de la

información. En particular, la minería de textos trata de extraer conocimiento biológico directamente de los artículos de biomedicina, dado que su texto contiene toda la información que puede ser potencialmente relevante para el investigador. Las aplicaciones de minería de textos en bioinformática incluyen tareas como reconocimiento de entidades nombradas, clustering, anotación semántica, resumen automático de textos, etc (ver Shatkay & Feldman (2003) para una revisión más completa del tema). Diversas competiciones de carácter académico han tenido como objetivo la minería de textos; un buen ejemplo de esto es BioCreative (Hirschman, Colosimo, Morgan & Yeh 2005).

Numerosas aplicaciones llevan a cabo tareas de minería de textos, pero los resultados parecen indicar que su alcance no llega a explotar más que superficialmente esta información. Comprender el lenguaje es una tarea difícil para una computadora, y pueden esperarse grandes descubrimientos con los avances en el campo de la lingüística computacional.

Otra alternativa que va aún más atrás en la fuente de la información reutiliza los propios datos de los experimentos de microarrays. Estos métodos son capaces de combinar los datos de diversos experimentos en un mismo análisis, y reciben el nombre de *meta-análisis*. Repositorios de datos experimentales como The Gene Expression Omnibus (GEO, Barrett et al. (2005)), guardan grandes cantidades de estos datos, y ofrecen un valioso recurso para las herramientas de meta-análisis. Aún así, estas herramientas pueden considerarse relativamente inmaduras, ya que la información en estos repositorios aparenta ser mucho más rica de lo que las aplicaciones modernas son capaces de extraer. El campo del meta-análisis constituye un línea de investigación con un gran potencial para el descubrimiento.

### 1.1.3 Aplicaciones de Bioinformática

Hemos repasado el rol tan importante que la bioinformática juega hoy en día en la biología molecular. Desafortunadamente, hay muchos retos todavía, gran parte de ellos debido a la complejidad de los problemas que se quieren resolver. La vida opera mediante una elaborada interacción de numerosos sistemas, cada uno a su vez constituido por numerosas entidades. La regulación de estos sistemas se ha vuelto intrincada y compleja en el curso de

la evolución; para descubrir sus secretos es necesario explorar un espacio de posibilidades enorme usando técnicas que son capaces de iluminar, cada una, sólo un aspecto en particular. No existe ningún método computacional capaz de tomar todos los datos y resolver el problema; la solución sólo puede materializarse mediante una inteligente combinación de diversas fuentes de información y herramientas de análisis.

### Scripts y librerías de funciones

Para satisfacer las necesidades de un método de trabajo tan flexible, los investigadores requieren de una larga colección de herramientas a su disposición que puedan usar e interconectar de varias maneras. La heterogeneidad de los datos y de las herramientas de análisis dificulta esta interconexión. De hecho, se puede decir que la bioinformática surgió cuando los investigadores comenzaron a usar lenguajes de programación para realizar estas tareas de formateo y procesado de datos (Stajich et al. 2002).

La forma en que la bioinformática surgió sirve para entender su situación actual. Al principio, los bioinformáticos eran básicamente investigadores que no tenían por qué tener una gran formación en programación. Tendían a preferir lenguajes de script como PERL porque eran fáciles de aprender, flexibles y con grandes capacidades de análisis y procesado de textos, y los usaban sobre todo para tareas de acceso y formateo de datos. Pronto empezó a ser evidente que muchas de estas tareas estaban siendo repetidas una y otra vez, y eso motivó el desarrollo de los primeros repositorios de funciones. Algunos ejemplos son Bioperl (Stajich et al. 2002), una librería de módulos en PERL, y Bioconductor (Gentleman et al. 2004), una colección de librerías para el lenguaje de programación de estadística R (Gentleman et al. 1997).

### Aplicaciones autónomas y servicios web

Las librerías de funciones sólo son capaces de cubrir ciertas necesidades; cuando las funcionalidades resultan demasiado complejas para ser exportadas en librerías de funciones, toman la forma de aplicaciones, ya sean locales u online. Las aplicaciones online tienen la ventaja de que no necesitan ningún tipo de instalación o mantenimiento por parte de usuario. Las aplicaciones

locales solían tener acceso a un interfaz de usuario más rico que las que se usaban a través del navegador, pero ésto está dejando de ser una consideración importante gracias a los avances de las tecnologías web modernas, en particular las posibilidades del lenguaje javascript.

Los protocolos de análisis de bioinformática empezaron a incluir pasos de copiado, formateado, y pegado entre diversas aplicaciones; esto supone un gran problema para la estandarización y el reuso de estos protocolos. La solución a este problema parece pasar por la implementación de servicios web, por ejemplo los basados en protocolos de comunicación, como SOAP o REST. Estos servicios web dan a los desarrolladores la posibilidad de acceder a las funcionalidades de manera programática, permitiendo así incluir éstas funcionalidades en análisis más largos (Stein 2002).

El éxito de las tecnologías web está convirtiendo en una práctica estándar el ofrecer interfaces basados en servicios web junto con las aplicaciones online. Una prueba de esta tendencia la encontramos en el éxito del número especial que la prestigiosa revista *Nucleic Acids Research* dedica a esta tecnología.

Una de las alternativas más populares para desarrollar estos servicios web es el protocolo SOAP (Box et al. 2000), aunque REST está ganando adeptos, tanto en el campo de la bioinformática como en el desarrollo web en general. SOAP ofrece una especificación más completa y más compleja que REST, y da soporte a características más avanzadas, como la validación de tipos. El principal inconveniente de SOAP viene precisamente de esta complejidad, que ha motivado largos debates sobre detalles altamente técnicos como el formato de codificación de los mensajes. A pesar de las ventajas frente a REST, en concreto su facilidad de interacción con el servidor desde el código cliente, las dificultades técnicas de SOAP complican al desarrollador de los servicios con consideraciones irrelevantes.

En cualquier caso, las tecnologías de servicios web no carecen de limitaciones con respecto a las librerías de funciones o APIs (del Inglés Application Programmer Interface). Quizás la más obvia son los problemas de latencia y conectividad, aunque esto no debería ser una limitación con los avances en la infraestructura de Internet. Por otra parte, los APIs tienen un mayor grado de granularidad en el acceso a las funcionalidades, aunque los servicios web, en teoría, podrían ser diseñados también a ese nivel. La limitación más

importante es frente a las librerías de funciones de código abierto, ya que estas pueden tener su código examinado, copiado y adaptado, mientras que los servicios web tienen una característica de "caja negra"

## 1.2 Objetivos

El principal objetivo de esta tesis ha sido desarrollar métodos y aplicaciones para ayudar en la interpretación funcional de experimentos de alto rendimiento. La sección de introducción identifica tres alternativas para la interpretación funciona: basada en anotaciones, en minería de textos y en el análisis de datos experimentales anteriores. Nuestra intención ha sido hacer contribuciones sustanciales a las tres alternativas en términos de aplicaciones utilizables.

Las bases de datos de anotaciones son numerosas y diversas. Algunos ejemplos son The Gene Ontology, que proporciona información acerca de la función biológica de los genes, por ejemplo, o KEGG, que informa sobre las rutas metabólicas en las que se cree que un determinado gen participa. Nuestra contribución en este caso ha sido un método que es capaz de encontrar patrones de sobre-representación, no sólo para anotaciones simples, sino para combinaciones de anotaciones, incluso de recursos diferentes. Esta alternativa proporciona unos resultados más exhaustivos, ya que distintas anotaciones sirven para expandir o delimitar el significado de las otras. Nuestro objetivo ha sido producir una aplicación que ofreciera esa funcionalidad para una colección de anotaciones de distintos recursos.

Los problemas con las anotaciones supervisadas que fueron expuestos en la sección anterior nos motivaron para desarrollar un método de interpretación funcional basado en minería de textos. El objetivo es crear un agrupamiento de genes basados en las similitudes en la literatura asociada a cada uno de ellos, y describir estos grupos basándonos en aquellas características mencionadas en la literatura que justifican dicho agrupamiento. Para ello se hizo uso de una factorización no negativa de matrices con unas propiedades muy prometedoras.

El paso final de la interpretación funcional suele ser un proceso de revisión de la literatura, de manera que las aplicaciones que asisten en la

interpretación funcional se pueden entender como guiando este proceso de revisión de la literatura. Nuestra aplicación debía, por lo tanto, proveer una herramienta que permitiese acceder a la colección de artículos usados para describir los genes, pero ordenados de acuerdo a su relevancia en relación con los resultados del análisis.

En cuanto al meta-análisis, nos dispusimos a crear una herramienta que permitiera aprovecharse del potencial de los datos en los repositorios de microarrays de manera sencilla. A este respecto, hay varios niveles de funcionalidad que debe cubrir. Por una parte, debe permitir al investigador descubrir aquellos experimentos relevantes en el repositorio de manera que puedan ser usados en una siguiente fase de meta-análisis. Por otra parte, la aplicación debe proporcionar un entendimiento sobre aquellas características comunes entre los experimentos que se hayan determinado relevantes. Finalmente, la aplicación debe ofrecer un entorno que permita examinar los datos y profundizar en los experimentos, así como ofrecer algunas tareas de meta-análisis por sí misma.

Un objetivo paralelo de esta tesis ha sido definir un conjunto de prácticas y armazones para el desarrollo de software en bioinformática. En este sentido, nuestras aplicaciones debían ser desarrolladas como proyectos de código abierto que fueran fáciles de usar por otros grupos de trabajo. Los proyectos en sí debían estar diseñados para maximizar la reusabilidad y la interoperatividad.

Con respecto a la reusabilidad, el objetivo era desarrollar un API que diese acceso a todas las funcionalidades, independientemente de su complejidad. Para este fin fue necesario desarrollar un sistema para empaquetar en un API funcionalidades con gran dependencia de datos.

Para obtener un buen nivel de interoperatividad, se requería que las aplicaciones fueran accesibles a 4 niveles: a través del API, localmente por línea de comandos, utilizando servicios web, y finalmente, a través de una aplicación online con un interfaz que permitiera una mejor interpretación de los datos. De esta manera, investigadores y desarrolladores podrían hacer uso de las funcionalidades en cualquier punto del gradiente que equilibra flexibilidad con facilidad de uso.

Esto requirió del desarrollo de servicios web para todas las aplicaciones.

Dado que escogimos usar el protocolo SOAP, famoso por su compleja especificación, nos propusimos desarrollar un armazón que facilitase su desarrollo basado en las particularidades de las aplicaciones de bioinformática; esto liberaría a los desarrolladores de complicaciones innecesarias.

## 1.3 Discusión

### 1.3.1 Visión General

Esta tesis realiza diversas contribuciones al análisis funcional de experimentos de alto rendimiento. Estas contribuciones cubren las tres alternativas propuestas en la sección de introducción: enriquecimiento de anotaciones, minería de textos y meta-análisis. Para cada una de estas aproximaciones se ha descrito un método de análisis y se ha acompañado de una aplicación completamente funcional.

Los tres métodos propuestos ofrecen alternativas complementarias para el análisis funcional. El método de análisis de anotaciones de GeneCodis ofrece la posibilidad de juntar anotaciones de diferentes bases de datos en el mismo análisis, dando así un nivel adicional de especificidad al análisis de anotaciones simples, que también se proporciona.

El método de análisis de literatura implementado en SENT se puede utilizar para detectar temas en la literatura cuya esencia no sería fácilmente capturada por anotaciones provenientes de vocabularios controlados, o que puede incluir información que no esté aún reflejada en las bases de datos. Adicionalmente, construye un índice con los artículos relevantes para permitir explorar la colección de artículos de acuerdo a su coincidencia con los descubrimientos del propio análisis.

La aplicación MARQ ofrece una alternativa muy distinta al usar directamente los datos de los experimentos de microarrays que se encuentran alojados en repositorios online. Este método no sólo indica la relevancia de los experimentos anteriores, también permite acceso a los datos procesados de miles de muestras mediante un interfaz que facilita su exploración en profundidad.

En paralelo a estas aplicaciones, este trabajo incluye contribuciones en

dos problemas relacionados. En BioNMF se ofrece un interfaz web para realizar factorización no-negativa de matrices: un método de factorización cuya alta interpretatividad favorece su amplia aplicabilidad a diversos problemas. BioNMF está de hecho en el corazón del análisis de minería de textos implementado en SENT. La otra contribución está también relacionada con la minería de textos y consiste en la completa implementación de un sistema de reconocimiento y normalización de nombres de genes en texto; este sistema también ha sido utilizado para el método de análisis implementado en SENT.

A pesar de que los métodos desarrollados en esta tesis cubren un amplio espectro de técnicas de análisis estadístico y de aprendizaje máquina, análisis de factores, *conditional random fields*, minería de conjuntos de items frecuentes, pruebas de hipótesis o estadísticas basadas en rangos, hay numerosas similitudes entre las aplicaciones que justifican la creación de armazones para favorecer la productividad en su desarrollo.

Todas las aplicaciones desarrolladas para esta tesis tienen un diseño similar basado en tres capas (ver la figura 5.1 y la sección 5.2). El grueso de las funcionalidades se ofrece a través de un API, que forma la primera capa. La segunda capa la constituye un servicio web que ofrece las funcionalidades online. La capa final consiste en una aplicación web con un interfaz amigable para lanzar trabajos en el servidor web y examinar sus resultados.

El Armazón Rbbt (Ruby Bioinformatics Toolkit) resuelve el problema de empaquetar funcionalidades complejas con gran dependencia de datos en un API. Rbbt ofrece un acceso unificado a todas estas funcionalidades que permite acceder a ellas a scripts u otras aplicaciones de manera programática y mediante un interfaz muy simple. BioNMF es la excepción en este caso, dado que las funcionalidades básicas no se encuentran implementadas dentro del entorno de Rbbt, y son un proyecto separado.

Hemos mencionado que la segunda capa de todas estas aplicaciones es un servidor web. Esta tesis presenta un armazón para simplificar el desarrollo encapsulando todas las consideraciones mas técnicas del protocolo y ofreciendo soporte para trabajos asíncronos. Este armazón, SimpleWS, es usado en todas las aplicaciones.

Teniendo en cuenta lo anterior, los objetivos de la tesis se ven cumplidos en la forma de:

1. Tres métodos de análisis funcional y sus tres aplicaciones correspondientes, cada uno ofreciendo una alternativa distinta: GeneCodis, SENT y MARQ. Estas aplicaciones implementan métodos de análisis estadístico y de aprendizaje máquina nuevos.

2. Una aplicación, BioNMF, para realizar factorización no-negativa de matrices, que tiene una amplia aplicabilidad para análisis de datos, y que ha sido usada para el método de minería de datos implementado por SENT.

3. Un sistema general de reconocimiento de entidades nombradas, con una configuración particular para reconocimiento de nombres de genes. Este sistema incluye un módulo que se encarga de la normalización de los nombres reconocidos. Ha sido usado también en SENT.

4. Dos armazones para el desarrollo de software: Rbbt, que empaqueta un amplio conjunto de funcionalidades de alto nivel, así como de acceso y análisis de datos, y SimpleWS, que ofrece un sencillo interfaz para desarrollar servicios web basados en SOAP.

Acompañando a las publicaciones científicas producidas (seis de ellas incluídas en este documento) se ha creado un repositorio de código en código abierto que cubre la mayor parte de las aplicaciones aquí presentadas. Las excepciones son BioNMF, que está mantenida por otra persona, y GeneCodis, que tiene su código portado a Rbbt, pero que todavía no ha sido liberado al dominio público. El resto de código de las aplicaciones y armazones está disponible y es de acceso libre; desde los métodos de análisis a las aplicaciones online.

Dado que los métodos de análisis presentados en esta tesis están descritos en las publicaciones y son relativamente independientes, el resto de esta sección se encargará de detallar algunos de los aspectos técnicos comunes. Esta discusión presentará algunas consideraciones de carácter técnico relacionadas con los lenguajes y prácticas de programación, que no han sido cubiertos en las publicaciones correspondientes.

### 1.3.2 Interoperatividad y Reusabilidad

Hemos discutido ya la importancia de hacer las aplicaciones accesibles a varios niveles: desde aplicaciones online amigables, a servicios web programáticos, a librerías de funciones y APIs. Para poder ofrecer estos distintos puntos de acceso sin tener que duplicar el código, los hemos organizado jerárquicamente, de manera que cada nivel delega el análisis en el siguiente.

La figura 5.1 ilustra el diseño de la aplicación. Existen tres niveles de acceso: uno a través de HTTP, el siguiente utilizando SOAP [1] y, finalmente, uno al que se accede utilizando el API. Se espera que los usuarios finales accedan a a través de un navegador a la primera capa, mientras que los scripts y workflows, como aquellos instanciados con Taverna (Hull et al. 2006), pueden acceder a las funcionalidades a través de SOAP o usando el API.

Las aplicaciones descritas en esta tesis interaccionan con las demás de diversas maneras, y, dependiendo del propósito, esta interacción ocurre a diferentes niveles. En algunos casos la funcionalidad de una aplicación se usa para mejorar la experiencia del usuario al examinar los resultados de otra. Este es el caso de SENT y GeneCodis, donde la funcionalidad de análisis de enriquecimiento de anotaciones proporcionada por GeneCodis se usa en la aplicación online de SENT para ayudar a examinar los grupos de genes. Dado que GeneCodis es usado desde el interfaz online, esta interacción se realiza desde la capa HTTP de SENT usando el servidor web de GeneCodis en SOAP. Otro caso distinto es el de la interacción entre SENT y BioNMF, en la que la factorización no-negativa de matrices se realiza en el corazón del propio análisis de SENT; en este caso la interacción es entre el API de SENT y el servidor web de BioNMF.

MARQ, por otra parte, utiliza GeneCodis para anotar las firmas de expresión con términos de GO, pero, a diferencia de SENT, utiliza la funcionalidad off-line, es decir, durante el proceso de preparación e instalación de los datos de la aplicación. En este caso MARQ accede a las funcionalidades de GeneCodis a través del API directamente; ésto es para no cargar excesivamente el servidor de GeneCodis. Este tipo de interacción está posibilitado

[1] SOAP puede utilizar HTTP para la comunicación de paquetes, pero no requiere un navegador

por el hecho de que la funcionalidad de GeneCodis está contenida en un API que puede ser instalado y configurado en la misma máquina en la que corre MARQ. Por otra parte, este tipo de interacción no sería factible entre SENT y BioNMF, ya que al no tener acceso a la infraestructura de alto rendimiento sobre la que funciona BioNMF, el análisis tardaría considerablemente más.

La figura 5.2 muestra nuestro ecosistema de aplicaciones con las interacciones arriba descritas. Nótese también como los APIs de GeneCodis, SENT y MARQ están incluídos dentro del armazón Rbbt. De esta manera, Rbbt no solo facilita exportar funcionalidades de alto nivel de complejidad mediante un API, sino que, al ofrecer un acceso integral a éstas, facilita que las aplicaciones reusen las funcionalidades de las demás. Nótese también como el API de BioNMF no está incluído en Rbbt y permanece como proyecto separado.

### 1.3.3 Servicios Web en SOAP

Como ya mencionamos con anterioridad, todas las aplicaciones ofrecen sus funcionalidades a través de un interfaz de servicio web. Esta tarea es extremadamente simple gracias al uso de nuestro armazón SimpleWS.

El protocolo de comunicación SOAP (Box et al. 2000) tiene una compleja especificación. El ámbito completo de la especificación incluye un nivel de interoperatividad que no es generalmente necesario en las aplicaciones de bioinformática: SOAP soporta la especificación de tipos de datos de complejidad arbitraria como parámetros de entrada y valor de salida; algo que no es realmente necesario para la mayoría de las aplicaciones, donde los componentes son tan diferentes. Se puede argumentar que los tipos complejos de datos no hacen sino complicar la interoperatividad en sistemas tan heterogéneos como los que se presentan en bioinformática, ya que fuerzan a los clientes del servicio a definir objetos de una jerarquía específica al servicio web simplemente para almacenar la información que se desea transmitir. A menudo es más simple transmitir la información en texto plano usando algún tipo sencillo de formato (como listas de valores separados por tabulador) y parsearlo en el cliente para introducirlo en la estructura de datos que resulte más conveniente. Esto es especialmente cierto al usar lenguajes de programación como PERL o Ruby, que son muy populares en parte gracias a su

habilidad para procesar texto con gran facilidad (Stajich et al. 2002).

Otra consideración importante sobre SOAP es la producción del fichero de WSDL (Web Server Description Language). Éste contiene la información que requiere el cliente para preparar el driver de comunicación. Como ocurre con la especificación de SOAP en general, el fichero de WSDL permite más funcionalidad de la que es realmente requerida por una aplicación de bioinformática, lo que hace que escribirlos sea innecesariamente complicado. SimpleWS ofrece un lenguaje específico de dominio (DSL) para escribir los servicios web que se encarga de producir un fichero WSDL actualizado automáticamente. EL DSL de SimpleWS es lo más escueto posible, sólo requiere el nombre del servicio, los nombre y tipos de los parámetros y el valor de retorno (necesarios para el WSDL), y un bloque de código que será el que se encargue de servir la funcionalidad. Opcionalmente, el DSL tiene primitivas para añadir documentación a las operaciones, a los parámetros y al servidor web en general; documentación que se añade automáticamente al WSDL y a una tabla HTML con numerosos tags CSS para incrustar en la sección de documentación de la aplicación online.

Una importante característica de los procesos de análisis en bioinformática que no se contempla directamente en la especificación de SOAP es la de dar soporte a trabajos asíncronos. A menudo, una funcionalidad requiere para ser completada un lapso de tiempo superior al que tarda una conexión HTTP en dar un "time out", dejando sólo la alternativa de utilizar procesos asíncronos. Dado que ésto no está soportado por SOAP directamente, debe de implementarse *ad-hoc* sobre las funcionalidades estándar. En SimpleWS, usando una sintaxis muy similar a la de los servicios normales, se pueden definir servicios asíncronos para los que SimpleWS proporciona automáticamente todos los métodos de gestión del ciclo de vida. Las tareas asíncronas se implementan como procesos en segundo plano del sistema operativo, y el soporte por defecto para su ciclo de vida incluye operaciones para lanzarlos, abortarlos, comprobar su estatus y recuperar los resultados, ya sean ficheros o parejas arbitrarias de clave-valor.

Con el armazón SimpleWS, implementar servicios web se convierte en una tarea trivial. Adicionalmente, encapsular los detalles de la producción del WSDL ayuda a controlar los errores de programación. Este framework se ha

usado con éxito en SENT, MARQ, GeneCodis, BioNMF y otras aplicaciones en desarrollo con una buena estabilidad. Dado que esta tesis no incluye un artículo específico sobre SimpleWS, el apéndice A cubre los detalles de diseño e implementación en más detalle.

### 1.3.4 Rbbt: Un Armazón Para Funcionalidades Reusables

Implementar las aplicaciones de GeneCodis, SENT y MARQ requirió de múltiples funcionalidades. SENT, por ejemplo, incluye funcionalidades relacionadas con el análisis de la matriz gen-palabra, pero también funcionalidades para acceder y preparar los datos, en particular, para entrenar los modelos de aprendizaje del sistema de reconocimiento y normalización de menciones a genes en texto. Algunas de estas funcionalidades son complicadas de implementar, especialmente las que requieren la presencia de determinados ficheros de datos. La propuesta de Rbbt es la de integrar el API con un repositorio local de datos. De esta manera, estos datos se sitúan en localizaciones concretas del repositorio de datos donde el API los puede encontrar. Para preparar este repositorio se incluyen herramientas de configuración que se encargan de descargar y procesar los datos necesarios.

Todas las funcionalidad de carácter general que son susceptibles de ser reusadas en otras aplicaciones son extraídas a módulos y empaquetadas en el proyecto de Rbbt: `rbbt`. Las funcionalidades más específicas de cada aplicación se empaquetan en proyectos separados, como `rbbt-sent` para SENT, mientras que el interfaz de servicios web y la aplicación online se incluyen en el paquete `rbbt-sent-www`. Todos estos paquetes se pueden instalar con el comando `gem`, un sistema de gestión de paquetes para Ruby similar a `CPAN` para PERL. Tras instalar cada "gema", una herramienta de configuración se encarga de realizar todas las tareas de instalación necesarias para preparar el entorno de trabajo que dará soporte a las funcionalidades de más alto nivel. Algunos ejemplos de funcionalidades reusadas en varias aplicaciones son: la traducción automática de identificadores de genes, la generación de vectores de palabras para fragmentos de texto, o la computación de la estadística hypergeométrica.

El paquete Rbbt incluye funcionalidades para una gran variedad de tar-

eas. Como no todas las tareas son necesarias para cada aplicación en particular, las herramientas de configuración pueden instalarlas selectivamente. Adicionalmente, para reducir la complejidad del API, se usan preferentemente tipos sencillos de datos frente a complejas jerarquías de objetos. El uso de tipos sencillos facilita el uso de las funcionalidades desde pequeños scripts, ya que no requiere producir código que se encargue de definir y manipular esos objetos. Adicionalmente, los tipos sencillos de datos reducen el acoplamiento entre distintos fragmentos de código, y permiten incluirlos por separado, o incluso, cortar, pegar y modificar fragmentos para adaptarlos a otros proyectos.

El lenguaje escogido para implementar el API de alto nivel de Rbbt es Ruby. Ruby es un lenguaje de script similar a PERL, pero con una fuerte orientación a objetos. Tiene la mayoría de las mejores características de PERL, como buenas capacidades de procesamiento de texto y una sintaxis escueta, pero se puede argumentar que tiene un aspecto más limpio y es más fácil de aprender. Además, dispone de algunos de los armazones para el desarrollo web más sofisticados. Esta última característica es de importancia crítica, ya que en las aplicaciones online es uno de los métodos preferidos para interaccionar con las aplicaciones de bioinformática, especialmente para los investigadores con menos formación en el uso ordenadores.

Algunas de las funcionalidades de Rbbt se delegan en paquetes mantenidos por terceras personas. En este caso, las herramientas de configuración se encargan de descargar, compilar e instalar estos paquetes, y Rbbt ofrece un acceso unificado y coherente a través de su API. De esta manera, el complicado proceso de facilitar la interoperatividad entre herramientas sólo debe resolverse una vez, y después puede usarse sin complicaciones.

### 1.3.5 Lenguajes Específicos de Dominio

Hemos discutido ya algunas de las ventajas de utilizar Ruby como lenguaje de alto nivel para el API, como son las capacidades de procesado de texto, o buena oferta de armazones para el desarrollo web. Otra ventaja importante de este lenguaje emana de sus capacidades de meta-programación, que simplifican el desarrollo de lenguajes específicos de dominio (DSL) basados en

código completamente ejecutable. Esto, junto con la idea de usar *convenio antes que configuración*, un principio básico de las metodologías ágiles de desarrollo, popularizadas por el famoso armazón de desarrollo web Ruby-on-Rails, simplifica considerablemente tareas tediosas o enrevesadas.

Las DSLs ofrecen varias ventajas frente al estándar de facto XML. Los ficheros XML son verbosos y difíciles de leer y editar manualmente. De hecho, muchas aplicaciones ofrecen herramientas de autoría con la finalidad de facilitar su manipulación. Los DSL están diseñados para eliminar la mayor parte del "ruido sintáctico" y así hacer las tareas de lectura y edición más sencillas, sin requerir más herramienta de autoría que cualquier editor de textos. Adicionalmente, los ficheros XML están pensados para almacenar valores de variables de configuración o datos, ya sean valores simples o de complejidad arbitraria, pero no para almacenar información de proceso, es decir, ejecutable. Por otra parte, los DSLs como los de nuestras aplicaciones son 100% ejecutables, lo que significa que la gramática usada para especificar la configuración contempla especificar cualquier tipo de bloque de código ejecutable. Este nivel de flexibilidad hace sencillo realizar tareas simples, pero permite también solucionar problemas más complejos o excepcionales.

En este trabajo, el DSL se usó primero para la tarea de reconocimiento y normalización de nombres de genes en texto, para lo que toda la información relacionada con el dominio se separó de los detalles de funcionamiento del algoritmo moviéndola a unos ficheros de configuración usando DSL. En el caso del paso de reconocimiento de genes, por ejemplo, el algoritmo de aprendizaje, conditional random fields, es una aproximación relativamente general que se puede aplicar a multitud de problemas; toda la información del dominio está contenida en la definición de las "features", que se realiza utilizando un sencillo fichero de configuración en DSL.

Otro claro ejemplo es SimpleWS, que usa una DSL para ofrecer un entorno sencillo y escueto, pero flexible, para definir servicios web. SimpleWS es uno de los elementos más importantes de nuestra infraestructura de aplicaciones, y su simplicidad y expresividad juegan un papel importante en el buen rendimiento de nuestro grupo produciendo aplicaciones. El apéndice A, sección A.4 incluye algunos ejemplos de uso que pueden servir para ilustrar el aspecto de esta DSL en particular. Otros ejemplos pueden verse en el

artículo presentado en la sección 4.5.

## 1.4 Conclusiones y Trabajo Futuro

### 1.4.1 Conclusiones

El trabajo presentado en esta tesis proporciona diversos nuevos métodos de análisis para bioinformática. Los artículos que los describen demuestran su aplicabilidad a problemas relevantes utilizando datos reales o "gold-standards" populares. Las aplicaciones resultantes ofrecen tres maneras complementarias para examinar los resultados de experimentos de alto rendimiento. De hecho, sus funcionalidades son a menudo reusadas para mejorar los resultados unas de otras.

Los métodos propuestos en estas aplicaciones cubren una amplia variedad de técnicas de análisis. GeneCodis usas herramientas estadísticas como las distribuciones Chi-cuadrado o hipergeométrica, corrección de hipótesis múltiples por control de ratio de falsos descubrimientos, y pruebas de permutaciones. SENT usa diversas tareas de minería de textos como el reconocimiento y normalización de nombres de genes utilizando conditional random fields, y análisis de factores usando factorización no-negativa de matrices. Finalmente, MARQ utiliza una estadística basada en rangos sobre firmas de expresión de microarrays analizadas con modelos lineales y Bayes empírico.

Las aplicaciones han sido diseñadas para favorecer su reutilización, permitiendo el acceso a través de distintos interfaces, como navegadores, servicios web en SOAP y APIs, y este diseño ha seguido una serie de buenas prácticas de programación. Es más, la colección completa de código para la mayor parte de estas aplicaciones está disponible a través de un repositorio de control de versiones. De hecho, el código ha sido empaquetado en distintos proyectos para favorecer su reutilización en otras aplicaciones y por otros desarrolladores.

Uno de estos paquetes, SimpleWS, ofrece un lenguaje específico de dominio para la definición de servicios web que simplifica enormemente su desarrollo. Este armazón ha sido usado exitosamente en las cuatro aplicaciones

presentadas en esta tesis: GeneCodis, SENT, BioNMF y MARQ, así como en diversas otras aplicaciones en desarrollo.

Otro de los paquetes, Rbbt, ofrece una infraestructura para implementar funcionalidades complejas a través de un sencillo API. Esta infraestructura es usada para envolver otros sistemas y herramientas mantenidas por terceras personas, así como scripts y librerías de funciones en R. A todas estas funcionalidades se accede a través del mismo API.

Es más, los pipelines de análisis de datos y de preparación e instalación de la aplicación están diseñadas también para favorecer la reutilización, permitiéndola por lo tanto a varios niveles: el API de alto nivel, los scripts de bajo nivel (como librerías de funciones en R), los pipelines de procesamiento y finalmente, los datos en sí, que están almacenados en un formato que favorece ser parseados.

El armazón de Rbbt se compone de un paquete con las funcionalidades básicas y una serie de paquetes plug-in para cada una de las aplicaciones, que incluyen sus interfaces online. De esta manera, cualquiera puede desplegar cualquiera de estas aplicaciones simplemente instalando los paquetes con la herramienta `gem` de Ruby, y corriendo los scripts de configuración.

### 1.4.2 Lineas de Trabajo Futuro

Esta tesis abre diversas líneas de trabajo futuro. Una de las más evidentes es, quizás, la de realizar más aplicaciones de bioinformática reutilizando la infraestructura disponible. Hemos sido capaces de tomar una metodología existente y convertirla en una aplicación completa, con API, servicios web y interfaz online, en un periodo de un par de semanas por un programador.

Actualmente estamos trabajando en proporcionar interfaces para dos aplicaciones: TaLasso y XMIPP. TaLasso es una herramienta de predicción de "targets" para miARN usa microarrays de expresion génica y de miARN. En particular, utiliza un método novedoso utilizando lasso (Tibshirani 1996). Por otra parte, XMIPP (Sorzano et al. 2004) es un paquete con funcionalidades de análisis para imágenes de microscopia; la aplicación que estamos desarrollando ofrecerá un interfaz web a las funcionalidades mas importantes así como otras provenientes de otros paquetes como detección de *normal-*

*modes*, *docking* o *fitting*.

Algunas líneas de trabajo bastante interesantes surgen directamente de la aplicación MARQ. Por una parte, la información proporcionada por MARQ ofrece un entendimiento de los datos experimentales que puede ayudar a plantear nuevas hipótesis de trabajo y sugerir nuevos métodos de análisis; todo esto favorecido por un gran repositorio de datos procesados y fácilmente accesibles. En esta misma linea estamos trabajando en investigar el efecto de *gastrin* usando datos de microarrays, con posibles aplicaciones en terapias para cáncer gástrico.

Por otra parte, la disponibilidad de tales repositorios de datos procesados y accesibles permite otros tipos de análisis de amplio espectro. Dada la matriz de firmas, el análisis de MARQ se puede entender como mirando los vectores de firmas de experimentos. Otras aplicaciones como SPELL (Hibbs et al. 2007) hacen lo opuesto: comparan la expresion de los genes como perfiles de expresión en cada una de las firmas y buscan relaciones entre ellos. Este tipo de análisis seria muy fácil de implementar a partir de la infraestructura disponible en MARQ.

Los productos de esta tesis abren líneas de trabajo también en minería de textos, en particular, en clasificación de documentos y reconocimiento de entidades nombradas; el desarrollo de métodos para estas tareas se verá favorecido por la presencia de una infraestructura en Rbbt para el entrenamiento y la validación de modelos y métodos de análisis. En particular, para el reconocimiento y la normalización de nombres de genes, Rbbt utiliza los datos y los scripts de evaluación de BioCreative.

En el caso de SENT, las características semánticas resultantes del análisis consisten en una colección de términos relevantes, que no tienen un orden o estructura particular. La interpretación de estas características semánticas se puede ver favorecida si se equiparan con términos de vocabularios controlados como pueden ser GO o UMLS (Bodenreider 2004). De hecho, esta idea ya está siendo desarrollada.

En términos más generales, el diseño del API de Rbbt, que dispone de varios pipelines de análisis y acceso a datos implementados con poco acoplamiento y con gran generalidad permite adaptarlos para nuevos datos o nuevas herramientas de análisis. El análisis de la literatura usando minería

de textos continúa siendo una alternativa muy prometedora para el análisis funcional; añadiendo más herramientas de análisis y fuentes de datos a Rbbt los investigadores serán capaces de implementar scripts más versátiles, y realizar así una exploración de la literatura más centrada y dirigida.

# Chapter 2

# Background

## 2.1 Introduction to Molecular Biology

### 2.1.1 Brief historical review

The mechanisms of heredity, although first described by Mendel in the mid nineteenth century, have been exploited for many centuries for animal and plant breeding. It has been only recently, during the twentieth century, that the technology become advanced enough to start deciphering how these mechanisms actually work. Some landmarks of these advances are the discovery of DNA double helix structure, recombinant DNA, the genetic code, and the Central Dogma.

The Central Dogma of molecular biology, in particular, is of special relevance to this thesis, as it has critical implications. As described by Francis Crick in a seminar paper (Crick 1970), DNA gets *transcribed* to RNA, and RNA is *translated* to proteins, which are the molecules in charge of most of the biological functions. Although it was soon discovered that this general scheme had exceptions, the implications were that one could asses to a good extent the biological mechanisms taking part in a cell solely by measuring the presence of these molecules (Alberts et al. 2002).

The last two decades have witnessed the development of numerous techniques for measuring the quantity (*expression level*) of these entities. These advances allow researchers to connect particular phenotypes to the biological

mechanisms that characterize them, with immediate applications in agriculture and health. During the last decade, these technologies have undergone another profound revolution: high throughput techniques, facilitated by the reduction in costs and the advances in axiomatization of the protocols. Techniques like gene expression microarrays and proteomics are chief examples.

Although we know about the existence of most of the genes in the human genome (Venter et al. 2001*a*), we still have not discovered the function of a great portion of them, let alone their complex interactions. Current techniques like *gene expression microarrays* or the more recent *next generation sequencing* (Schuster 2008), hold the promise of great discoveries in the years to come.

### 2.1.2 Computational biology and bioinformatics

High throughput technologies have highlighted the importance of computational techniques; the extremely complex system of life can only be tackled with powerful knowledge management systems capable of dealing with vasts amounts of information very heterogeneous in nature and constantly being produced by experimental groups. Computational biology, in consequence, has emerged as a very active field of research, and attracted attention from neighbouring fields such as computer science, applied mathematics, machine learning, and statistics. In particular, many advances in statistics have been motivated by molecular biology problems that challenge classical approaches: a challenge often brought on by their very scale and a higher complexity than has typically been tackled by statistics in the past.

Systems biology has become a central concept in recent years; it refers to the study of cellular biology from a holistic perspective, by integrating information from the different levels and facets of a biological system: Evolution, regulation of gene expression, metabolic pathways, etc. Bioinformatics thus plays a critical role in systems biology, as it has become the main enabler of this integration; all these different disciplines can only be connected through information-heavy and computer driven protocols, such as those using high throughput experiments coupled with information management systems.

Definitions of bioinformatics may vary between individual researchers,

generally regarding what should or should not be included under the bioinformatics umbrella. As for the present work, we will consider under this heading software tools used in biological research, data analysis and modeling techniques applied to the field, development of databases and information encoding standards for biological data, and any other technical development aimed at helping researchers examine biological data.

A great amount of work has been done in the field of bioinformatics, and every year yields many new applications and data analysis techniques. Even though this rate of production may seem satisfactory, with problems being addressed at a timely pace, the field does lack sophisticated software development practices. Computer Scientists have only recently gotten involved in Bioinformatics, and there are many aspects of the craft that could benefit from software development methodologies and frameworks; just like web development has seen a great deal of advances with the advent of the so called *web 2.0*. In fact, the internet is now ubiquitous to most research fields, including bioinformatics, and essential in allowing for an adequate information flow between the different data sources. Internet technologies and databases are undergoing profound changes, and great potential remains to be tapped by bioinformatics. Bioinformatics has thus much to gain from advances in communication protocols, software development practices and frameworks, and information management standards.

## 2.2 Gene Expression Microarrays

This section will review some important facts about the gene expression microarray technology; in it we find a great example of a high throughput technology, and a great portion of the work in this thesis is closely connected to this particular topic.

Gene expression microarrays can assay the expression levels of thousands of genes at a time. A gene is said to be expressed if it becomes transcribed to mRNA (messenger RNA), the molecule that transmits the genetic information from the DNA to the ribosomes that use it to build the corresponding protein. The amount of mRNA for a given gene is, thus, a rough estimate of the amount of its protein being produced. This quantitative assessment, as

mentioned in the beginning of this chapter, provides a 'snapshot' of the underlaying biological mechanisms involved in the samples under study. However, the use of mRNA as a proxy for the proteome can only be an approximation, since it is not too uncommon for mRNA to not be transcribed to a protein due to other regulatory events, or even to have a function that does not require this transcription at all. Gene expression microarrays are used to study many different conditions, among which diseased tissues or drug effects.

The fact that these techniques measure expression levels for thousands of genes at the same time highlights the importance of computational techniques to process these data, infer interesting patterns, and derive some practical understanding.

The reminder of this section will provide a brief review of the basic facts behind the manufacture and analysis of microarray experiments. More complete reviews of gene expression microarrays technology and data analysis methods can be found in Schulze & Downward (2001), Allison et al. (2006) and Slonim & Yanai (2009).

### 2.2.1 Microarray Technology

The basic idea behind microarray technology is using the hybridization properties of nucleic acid molecules; specifically, that nucleic acid bases, also known as nucleotides, will tend to form non-covalent bonds with their complementary bases under the right experimental conditions. By laying out specific sequences of nucleotides, known as probes, on predefined locations on a physical support, these can be used to monitor the presence of complementary strings of nucleic acids suspended in a buffer solution. These complementary sequences are said to 'hybridize' with the probes. Since nucleotides form rather weak non-covalent bonds, only by having many base matches can the bonding energy be enough to avoid being washed away. The extent to which the probes on a particular position in the microarray have found matching sequences to hybridize to can be measured by labeling the latter with fluorescent dyes and measuring the emission from each probe location.

Two technologies are the most popular for producing these microarrays,

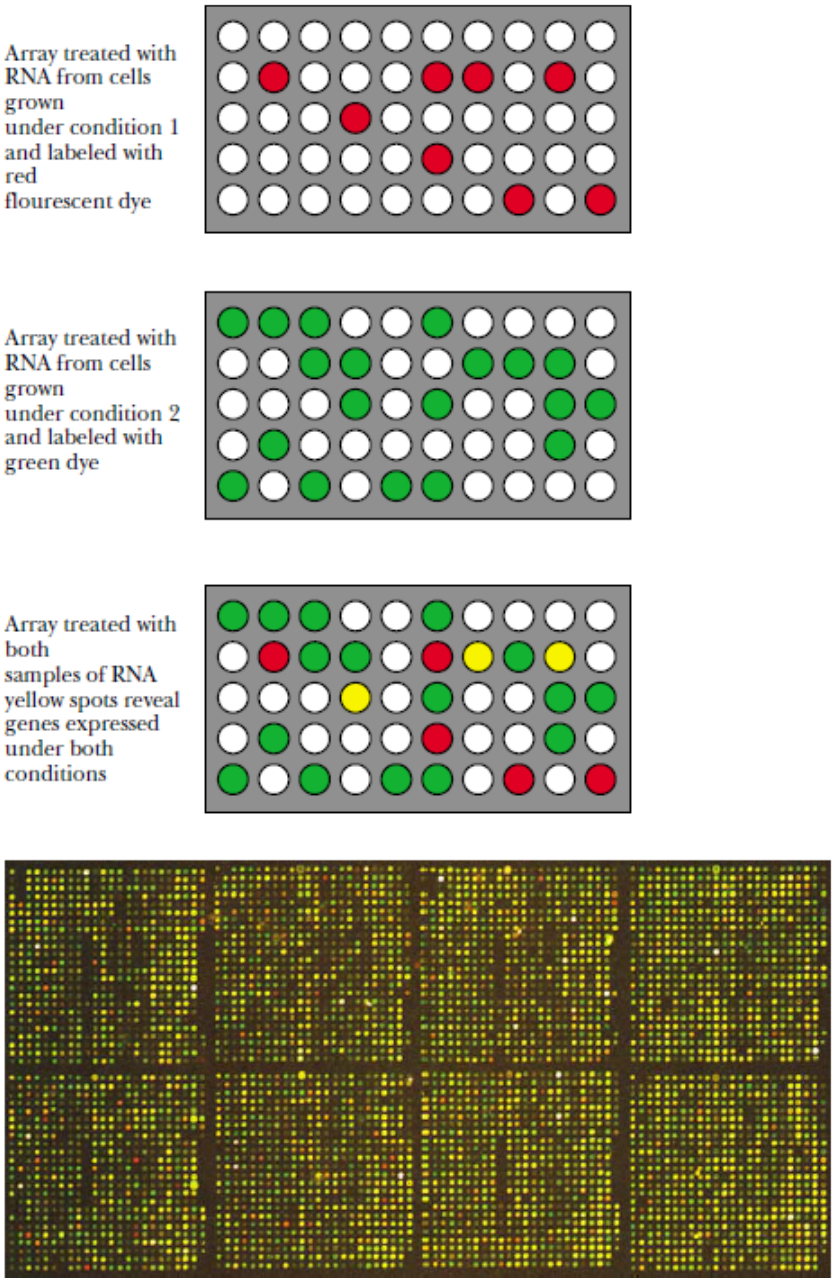


Figure 2.1: Schematic of a cDNA microarray with red and green fluorescent dyes, and picture of a real microarray image. Source: *MOLECULAR BIOLOGY. David Clark. Elsevier Academic Press. 2005. ISBN: 0-12-175551-7. page 710.*

cDNA and oligonucleotide arrays. The first kind, cDNA, is based on a collection of complementary DNA strings for known genes. These strings are amplified using Polymerase Chain Reaction (PCR), in which technique the replication machinery of a particular high temperature resistant bacteria is used to duplicate DNA strings, and placed on the physical support, generally glass. These probes are attached by means of a robotic arm using a printing system much like ink injection printers. The probes on cDNA typically range from 20 to 1000 bases long, with an average of four or five hundred bases. This kind of microarrays are habitually designed by the research group conducting the analysis.

Oligonucleotides, popularized by Affymetrix, synthesize the probes directly on the support using a photolithography technique similar to the one used in printing circuits. Using a light sensible reactive and light masks, the bases can be added to the sequences to extend the probe sequences one base at a time until completion. Oligonucleotide probes are smaller than cDNA, typically around 125 bases long. Oligonucleotide arrays are produced by the manufacturer (eg. Affymetrix); each contain standard probe layouts, and are shipped with standard reading and analysis software. See Dalma-Weiszhausz et al. (2006) for a more complete review of the Affymetrix technology.

In the case of microarray for gene expression, the molecules we wish to quantify are messenger RNA (mRNA), the intermediate courier between DNA and the ribosome transcription machinery that builds the proteins. In particular, for cDNA arrays, mRNA must be extracted from the chosen sample, reverse-transcribed to DNA molecules (called cDNA for complimentary DNA), labeled with fluorescent dyes, typically Cy5 (red) and Cy3 (green), hybridized to the microarray, and finally washed off to remove weak bonds.

Microarray techniques are commonly used in a comparative setting where the expression levels of genes in a particular sample of interest are compared against the expression levels of a reference sample. The ratios of gene expression between the two types of samples are referred to as *differential gene expression* ratios. In cDNA, treated and reference samples are usually placed in the same chip, so the emission ratio between the Cy5 and Cy3 dyes already represents the differential gene expression ratio between the two conditions. In oligonucleotide arrays, each sample is placed on a different chip and then

they are compared.

### 2.2.2 Statistical analysis

By statistical analysis we refer to everything related to extracting the differential gene expression ratios and assessing their importance. This typically involves normalizing the raw values read from the microarray to correct technical biases and performing inferences over the resulting values. In order to make the results from the analysis more meaningful, the inherent variability of microarray measurements must be alleviated by the use of technical replicates (multiple gene probes in the array) and sample replicates (multiple chips for the same sample). There are several ways to perform this normalization, see Lim et al. (2007) for a comparative. Affymetrix chips, for example, are shipped with the necessary tools to perform this step.

After the values have been normalized, a test statistic is required to infer interesting patters of gene expression. The most common statistics, as above mentioned, are for differential gene expression. One of the first approaches was based on measuring the ratio of gene expression values between samples, having previously averaged technical and sample replicates. A gene was considered to be differentially expressed (over expressed or under expressed compared to the reference) if its ratio was larger than 2 fold changes, for example.

The fold change approach has one important drawback: since it uses only the mean values of the gene expression levels for each sample type, it does not take into account the variability in the actual values. A particular gene may have a large fold change, but if the values for the replicates show a strong variability, its particular fold change value is likely only due to chance. This fact has encouraged researchers to favor more statistically sound approaches that take variability into account. Typical approaches are based on sample mean comparison tests like the student's t-test. The dimension of the problem actually makes the student's t-test a poor choice; the same ideas have been adapted to this problem in approaches like SAM (Significant analysis of Microarrays, Tusher et al. (2001)) and eBayes (empirical Bayes, Efron et al. (2001)), but many other alternatives exist.

Statistical analysis of differential expression using t-test based statistics have also been criticized because, given a large enough sample, even very small changes in gene expression can become significant. These very small changes in gene expression, although significant in a statistical sense, may not indicate any meaningful biological change.

Another important consideration comes from the size of the hypothesis space. Each gene in the analysis is subject to a hypothesis test, where the alternative hypothesis is that the gene is regulated by the biological condition under examination. This poses a multiple hypothesis problem, since, with such a large hypothesis space, many of the 'interesting' findings may actually be fortuitous. Problems of this sort have led to the development of new multiple hypothesis approaches like the False Discovery Rate (Benjamini & Hochberg 1995, Verhoeven et al. 2005).

### 2.2.3 Functional interpretation

The statistical analysis, whichever the method used to perform it, identifies genes that are likely to be associated with the contested conditions. It's important to note that gene expression analysis is a discovery technique, rather than a hypothesis testing technique. This is a subtle consideration: the experiment is not performed to prove a particular hypothesis, but to search for interesting patterns, which may in turn suggest interesting hypotheses. In consequence, statistical analyses of DNA microarrays are typically followed by interpretation in the hopes that interesting conclusions might arise based on the particularities of the genes found to be significantly deregulated. This step is often referred to as *functional analysis* or *functional interpretation.*

One of the most common approaches for functional interpretation includes the use of curated annotation databases, where genes are associated with terms from a controlled vocabulary, like in the Gene Ontology (Ashburner et al. 2000, Hill et al. 2008), or organized into biological pathways, like in the KEGG pathway database (Kanehisa et al. 2004). These annotations have the benefit of being curated, that is, manually extracted from the literature and introduced into a database, which to an extent guarantees their correctness.

Additionally, the fact that the terms used in the annotation come from

a controlled vocabulary makes performing statistical analyses over their distribution easier. One particularly popular approach consists in performing annotation enrichment analysis over the terms associated to a list of genes, for example, the list of genes found to be over-expressed under certain conditions. This kind of analysis could determine if certain annotation terms are more frequent in the query list of genes than they would be by mere chance.

Curated annotations have their drawbacks as well. The curation process is expensive and, since articles are constantly being published, it is seldom up to date. Also, curation efforts may favor certain topics, genes, or organisms, potentially introducing biases, or too sparsely covering certain aspects. Finally, the expressiveness of annotations from a controlled vocabulary is limited precisely because the predicates that are used in the results are predefined by the vocabulary itself.

The alternative to curated annotations is returning to the information source. In particular, literature mining tools try to extract biological knowledge directly from biomedical articles, since their free text may contain all information relevant to the researcher. Biomedical text mining encompasses different tasks such as entity extraction, clustering, semantic annotation of biological entities, or automatic summarization of articles (see Shatkay & Feldman (2003) for a review in literature mining methods). Biomedical text mining has recently been the subject of several competitions, a chief example of which is the BioCreative competition (Hirschman, Yeh, Blaschke & Valencia 2005).

Numerous applications perform text-mining, but results seem to indicate that current approaches are unable to fully exploit this information. Understanding language remains a very hard task, and great advances may be expected as the field of computational linguistics continues to make progress.

An approach delving even further back into the source of the information reuses experimental data from previous experiments. This approach, combining data from several experiments, is commonly referred to as meta-analysis. Data repositories, such as the Gene Expression Omnibus (Barrett et al. 2009), hold large amounts of experimental data, with great potential for meta-analysis. This kind of tool can still be considered in its infancy, since the data on the repositories greatly supersedes the ability of contem-

porary tools to exploit it. Meta-analysis constitutes a great line of research and harbors the promise of a vast and untapped potential for discovery.

## 2.3 Bioinformatics Applications

We have discussed the critical role that bioinformatics plays in modern molecular biology. Unfortunately, many great challenges are yet to be met, most of them having to do with the complexity of the problems themselves. Biology operates in an elaborate interplay of various systems, each in turn composed of numerous entities. The regulation of these systems has become increasingly intricate and complex throughout the history of evolution; in unraveling its secrets, researchers must explore a vast space of possibilities using techniques that can only each illuminate some particular aspect. There is no computational method that can gather all the data and just solve the problem; the solution can only be slowly envisioned by the intelligent combination of numerous data sources and analysis tools.

This section briefly reviews the history and current status of bioinformatics software development tools and practices.

### 2.3.1 Scripts and function libraries

To satisfactorily fulfill the requirements of such a flexible exploration method, researchers need a large collection of tools at their disposal that they can apply and interconnect in various ways. The heterogeneity of both data and analysis tools complicates this interconnection. In fact, it could be said that bioinformatics emerged when researchers began to use programming languages to script some of these complex data formatting and processing steps (Stajich et al. 2002).

The emergence of the bioinformatics field can help understand its current situation. At first bioinformaticians were basically researchers and need not have a strong computer programming background. They tended to favor scripting languages such as PERL because they found them easy to learn, flexible, and with powerful text processing capabilities; which they used mostly for data gathering and formatting. It soon became apparent

that many tasks were constantly being redone and thus, the first repositories of functionalities started to emerge. Two good examples are Bioperl (Stajich et al. 2002), a collection of modules for the PERL language, and Bioconductor (Gentleman et al. 2004), a collection of function libraries for the R statistical programming environment (Gentleman et al. 1997).

### 2.3.2 Standalone applications and web services

Function libraries can only go so far; many functionalities are complex enough to be exported only as standalone applications, which can take the form of local or online applications. To their advantage, online applications simplify by not requiring any type of installation or maintenance on the part of the user. Standalone local applications traditionally allowed access to richer user interfaces, although this is progressively becoming less of an issue since browser applications started exploiting the capabilities of javascript to improve user experience.

Bioinformatics protocols began to include copying, reformatting and pasting information across the different applications; this strongly hindered standardization and reuse. The solution to this problem seemed to be implementing online servers to offer interfaces using web service communication protocols such as SOAP or REST. These web service technologies gave software developers and script writers access to the functionalities in a programmatic way, furthermore making it possible to include their functionalities in larger analyses (Stein 2002).

The success of web service technology is making it a standard practice for bioinformatics applications to offer web service interfaces along with standard online ones. Proof of it is the annual special issue that the journal *Nucleic Acids Research* has on web services. One of the most popular options for implementing these web services is the SOAP protocol (Box et al. 2000), although REST is gaining adepts, both in bioinformatics and in general web development. SOAP offers a far more complex and complete specification for these web services than REST and consequently supports more advanced features such as type validation in messages. The downside to SOAP springs from its very complexity, which in fact spurred long debates over very tech-

nical details such as message encoding formats. In spite of advantages over the REST protocol, namely in that it allows for a cleaner programmatic interface for service client developers, the technical nuisance of the SOAP protocol burdens server side developers with unnecessary complications.

In any case, web services are not without limitations with respect to function libraries and APIs. Particularly obvious are the problems with connectivity and latency, but these expect to be overcome with advances in internet technologies. Also, APIs typically offer a finer level of granularity in accessing their functionalities, although web services could, in theory, be designed at that level of granularity. Finally, a more pervasive drawback comes from the fact that open source APIs can have their code examined, modified and copied, whereas web services have a 'black box' property.

# Chapter 3

# Objectives

The main objective for this thesis was developing methods and applications to aid functional interpretation of high throughput experiments. The background chapter (chapter 2) identifies three approaches to functional interpretation: based on curated annotations, on literature mining, and on meta-analysis of previous experimental data. Our aim was to make substantial contributions to each approach in terms of usable applications.

Annotation databases are numerous and diverse. Some examples are the Gene Ontology, which provides information about the biological function of genes, or the KEGG pathway database, which indicates the biological pathways in which genes are thought to take part. Our contribution was to propose an approach that could find patterns of over-representation not only for each of the single terms in each database, as most applications already did, but also for combinations of such terms, even from different databases. This approach provides more comprehensive results, since different annotations would qualify and expand each other's meaning. Our objective was to produce a practical application that offered such functionality for a collection of different annotation resources.

The problems of curated annotations outlined in the background chapter enticed us to develop a literature mining approach to functional interpretation. Our objective was twofold: group related genes by means of the similarity in their associated literature, and have an outline of the characteristics described in their literature that justify grouping them that particular way.

To this end we were to use a very promising matrix factorization technique called non-negative matrix factorization.

The final step of any functional analysis typically consists of a review of the relevant literature; most applications for functional interpretations can be seen as guiding the researchers in their literature examination step. We wanted our literature mining application to also be a literature examination tool that could be used to retrieve articles by relevance to the actual conclusions of the analysis.

With regards to meta-analysis, we wanted a tool that could easily harness the potential of the data in microarray repositories. There are several levels of functionality that the application should cover. First, it should allow the researcher to find relevant experiments in the repository so that they can be used in for meta-analysis. The application should also provide insights into the common characteristics of the signatures found to be most relevant; this way, the actual analysis could already offer interesting insights into the data. Finally, the application interface should provide sufficient exploratory tools to be used effectively as a meta-analysis application in itself.

A parallel objective of the thesis was to define a set of software development practices and frameworks that could fill some of the current gaps in software crafting for bioinformatics. In this sense, we wanted all our applications to be developed as open-source projects that could easily be deployed by other groups. The projects themselves should be coded in a way that could maximize reusability, and interoperability.

Reusability was to be achieved by providing an API access for all functionalities, regardless of how complex they where. To this end we had to address the issue of how to package highly data-dependant functionalities into an API.

As to interoperability, we wanted to make sure that our applications where accessible at these four levels: Through an API, locally by using the command line, using web service interfaces in SOAP, and through friendly online interfaces (browser based). This way, each type of user, from application developers to researchers whom the applications are ultimately thought for, could all access their application at whichever point seemed more comfortable to them in terms of a gradient balancing flexibility and easy use.

This, of course, implied that all our applications should export their functionalities using web services. Since we opted for the SOAP communication protocol, famous for its complex specification, we wanted to develop a framework for developing such web services that was tailored to the particularities of bioinformatics application; this would free developers from unnecessary complexity.

# Chapter 4

# Relevant Publications

The six publications contained in this section are the ones that best reflect the work of this thesis. Four of them represent the four ready to use applications: GeneCodis, SENT, BioNMF and MARQ. These articles where published in the journal *Nucleic Acids Research*, which edits an annual special issue on web services for bioinformatics that constitutes the ideal forum for these applications.

I was the main author of two of these four projects: SENT and MARQ. In both cases, other than the manuscript, I was in charge of implementing the data processing and learning methods, as well as the web service and online interfaces. For the other two applications, GeneCodis and BioNMF, I collaborated in the overall design of the application and in the writing of the manuscript, and was in charge of implementing their web service infrastructure. Also, for GeneCodis, I ported the application to the Rbbt framework.

The other two publications where presented at conferences and describe some of the more technical aspect of the work. The development frameworks, such as Rbbt and SimpleWS, still improve significantly with each application that uses them; so far, only the parts that are start to look relatively mature have been published. One of the papers describes the gene mention and normalization system developed for the data preprocessing of SENT, and included in the Rbbt framework. The other, the Rbbt framework itself: its design and most relevant features. I was also the main developer behind these projects.

## 4.1 GeneCodis: interpreting gene lists through enrichment analysis and integration of diverse biological information

**Reference:** Nogales-Cadenas, R., Carmona-Saez, P., **Vazquez, M.**, Vicente, C., Yang, X., Tirado, F., Carazo, J. M. & Pascual-Montano, A. D. (2009), 'GeneCodis: interpreting gene lists through enrichment analysis and integration of diverse biological information', *Nucleic Acids Research* **37**(Web Server issue), W317–W322. (JCR 2008 impact factor: **6.878**).

**Summary:** Interpreting results from high throughput experiments requires taking into account relevant information for a great number of entities. To aid accessing and reviewing this information, curated resources such as the Gene Ontology (GO) annotate genes with terms from controlled vocabularies that describe their biological function, the process their are involved with, or the cellular location of their protein products. These annotations not only work as succinct descriptions of the relevant information for the gene, but are also susceptible for statistical analysis. Several tools provide statistical analysis of the annotations for a given list of relevant genes. These statistics are mostly concerned with the over-representation of some of these annotations in the query list compared with a reference list of genes, such as the complete genome.

The main contribution of this work is a method for determining over-represented combinations of terms. By combining these terms, the semantics of the applications is expanded, specially when these terms happen to come from different resources which may provide complementary information. This approach, co-annotation enrichment analysis, provides an alternative to the classical one of examining the over-representation of one term at a time, and has been very well received by the community.

This work implements an online tool that performs co-annotation enrichment analysis, with the characteristic of being able to integrate annotations from different sources like as GO terms and KEGG pathways, among many

others, or even annotations provided by the user.

The application is accessible via SOAP web services and online interfaces. The source code for the application has been ported to the Rbbt framework; it is yet not publicly available, but, can be provided upon request.

*Published online 22 May 2009* *Nucleic Acids Research, 2009, Vol. 37, Web Server issue* ***W317–W322***
*doi:10.1093/nar/gkp416*

# GeneCodis: interpreting gene lists through enrichment analysis and integration of diverse biological information

**Ruben Nogales-Cadenas[1], Pedro Carmona-Saez[1], Miguel Vazquez[2], Cesar Vicente[1], Xiaoyuan Yang[1], Francisco Tirado[1], Jose María Carazo[3] and Alberto Pascual-Montano[1,*]**

[1]Computer Architecture Department, [2]Software Engineering Department, Complutense University of Madrid and [3]Biocomputing Unit, National Center for Biotechnology, CNB-CSIC, Madrid, Spain

Received January 31, 2009; Revised April 24, 2009; Accepted May 6, 2009

## ABSTRACT

**GeneCodis is a web server application for functional analysis of gene lists that integrates different sources of information and finds modular patterns of interrelated annotations. This integrative approach has proved to be useful for the interpretation of high-throughput experiments and therefore a new version of the system has been developed to expand its functionality and scope. GeneCodis now expands the functional information with regulatory patterns and user-defined annotations, offering the possibility of integrating all sources of information in the same analysis. Traditional singular enrichment is now permitted and more organisms and gene identifiers have been added to the database. The application has been re-engineered to improve performance, accessibility and scalability. In addition, GeneCodis can now be accessed through a public SOAP web services interface, enabling users to perform analysis from their own scripts and workflows. The application is freely available at http://genecodis.dacya.ucm.es**

## INTRODUCTION

High-throughput experiments such as DNA microarrays or proteomics techniques have been widely used during the last decade. Currently they are standard technologies in many research centers and facilities. Although these methodologies generate huge amounts of information, the challenge lies not only in the data processing area, in which the bioinformatics community has made significant advances, but also in the interpretation of the information. In this context, new data mining techniques able to extract interpretable facts and biological knowledge are needed to understand the biological meaning of experiments.

An essential task in this analysis is to translate gene signatures into information that can assist in understanding the underlying biological mechanisms. In the last few years several methods and tools have been developed to interpret large lists of genes or proteins using information available in biological databases. The common idea used in most of these methods is to find functional descriptors that are significantly enriched in the gene signature.

The first type of techniques that emerged in this field were focused on evaluating the frequency of individual annotations and apply an statistical test to determine which annotations are significantly enriched in an input list with respect to a reference list, usually the whole genome or all genes in a microarray. Annotations from different sources like Gene Ontology (GO) (1) or KEGG (2) are commonly used in this context. Several tools have been developed following this approach, and although each application introduces its own variations such as different statistical tests, sources of annotations or supported organisms, they all carry out the same type of analysis, producing only slightly differences in the results. Good reviews of such methods can be found in (3,4).

A fresh line of research appeared with the observation that the use of thresholds to select the significant genes could underestimate the effect of significant biological effects during the functional analysis. This idea derived in a new and different analytical concept in which the distribution of annotations is evaluated over the whole list of genes, sorted by their correlation with the phenotype. In this context, the gene set enrichment analysis (GSEA) was introduced by Subramanian and Tamayo (5) and since then different methods have followed this approach.

Nevertheless both approaches, the standard enrichment analysis and GSEA methods evaluate each annotation

*To whom correspondence should be addressed. Tel: +34 913944420; Fax: +34 913944687; Email: pascual@fis.ucm.es

The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

independently from the others without taking into account the potential relationships among them. However, most of the annotations in biological databases are interconnected because they are associated to common genes. Patterns that contain these relationships can provide invaluable information and extend our understanding of biological events associated to the experimental system. It is therefore highly desirable to evaluate these associations in the functional analysis of gene lists. New tools that attempt to extract this type of information have been proposed [see a review in (6)].

In 2007, we introduced GeneCodis (7), a tool for modular enrichment analysis oriented to integrate information from different sources and find enriched combinations of annotations in large lists of genes or proteins. Modular enrichment represents a step forward in the functional analysis because of its capacity to integrate heterogeneous annotations and to discover significant combinations among them. Since its original publication, this tool has achieved more than 25 000 submissions from all over the world and has been referenced in different works. A mirror has even been recently set up at the Center for Bioinformatics in Peking University to facilitate the access in the Asian region.

In this work, we present a new version of the software with improved functionality, performance, accessibility and extended scope. First, we have expanded the type of information available in the application by incorporating new types of annotations such as microRNAs and transcription factors. In this way, the new version of GeneCodis offers now the possibility to mine not only functional information but also regulatory patterns with the potential to integrate both sources of annotations in the same analysis. Moreover, the application allows researchers to submit their own annotations and perform a joint analysis with the rest of available information. The new version also includes the analysis of individual annotations (singular enrichment analysis) and new type of gene identifiers and organisms were included to cover more possible analytical scenarios.

From a technical point of view, GeneCodis has been completely reengineered making it faster and more flexible. The algorithm that retrieves sets of concurrent annotations has been improved and runs in a multi-grid environment to better handle large number of jobs simultaneously and thus improving the performance and the throughput of the system. The application can now be accessed in a programmatic way using SOAP Web Services. This allows researchers to include GeneCodis functionalities in their own data analysis pipelines. Finally, the new friendly interface design facilitates the use of the tool, and new graphs and file formats have been added in the results to facilitate its processing and interpretation.

## FEATURES AND FUNCTIONALITY

GeneCodis has suffered plenty of changes since its first publication, but the tool still works in a similar way. Users have to select the organism and the biological annotations to be considered in the analysis and then load the gene list. As advanced options the reference set of genes can be specified; otherwise the whole genome will be used by default. It is also possible to select the preferred statistical test among three possible alternatives: hypergeometric test (default), chi-square test or both. Annotations will be considered in the concurrent analysis only if they appear in at least a minimum number of genes, known as minimum support, which is set to three by default. Finally, users can select the multiple hypothesis correction method: false discovery rate method (default), permutation-based correction or none. As new feature, we allow the submission of a file with a list of user-defined annotations that can be considered in the analysis together with the rest of selected annotations.

### Functional analysis in GeneCodis

As stated previously, the enrichment analysis of individual annotations was the first method introduced for the functional interpretation of large lists of genes or proteins, and still remains the most popular one. There is a large collection of tools that implement this methodology, most of them focused on the evaluation of GO annotations. The arrival of GSEA methods turned the enrichment analysis of individual annotations from a gene-centered to a gene-set based analysis. These methods, although extremely useful for the interpretation of gene lists, do not exploit the inter-relationships that exist among gene annotations. In this context, tools such as GeneCodis provide a new way to analyze functional information by taking into account these relationships among annotations associated to common genes in the list. This analysis offers different advantages with respect to singular enrichment methods. Combined terms may contain unique biological meaning for a given study, not held by individual terms. For example, the combination of two terms such as *Apoptosis* and *Mitochondria* may be enriched in a gene list while the individual annotations are not significant if evaluated independently in the same list of genes. But probably more interesting is the potential of this approach to integrate and jointly analyze information that covers different aspects of the biology of the genes. Although there are several applications for modular analysis [see (6) for a review], GeneCodis is one of the few tools that tackle this problem by offering enrichment analysis of concurrent annotations using an algorithm based on the extraction of frequent itemsets [see (8) for details].

In this new version of GeneCodis, we have also included the singular enrichment. That is, in addition to the analysis of combinations of annotations, analysis of individual annotations is also performed and included in the results. In this way, if two categories are selected, results will include three different lists: one with the concurrent analysis results and two other lists with singular enrichment information, one for each category. Results are provided in different formats: html tables, tabulated text files, xml files, pie charts and bar graphs.

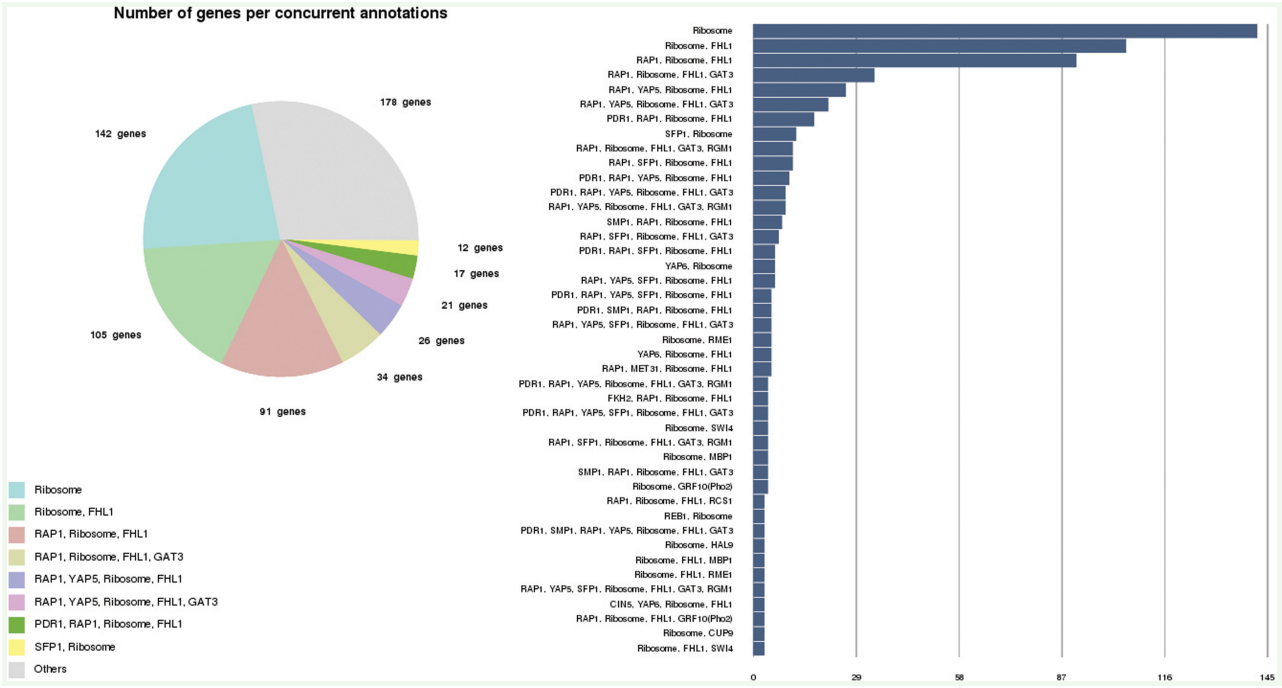


**Figure 1.** Example of the graphics generated in a typical GeneCodis analysis. Enriched combinations of significant annotations are represented in a pie and bar graphs, where the length of the bars and size of the slices are proportional to the number of genes supporting the significant combination of annotations.

## New sources of annotations: integrating functional and regulatory information

Among the multiple resources of information, GO is by far the most popular one for functional analysis of gene lists. This is reasonable due to the rich content of GO in terms that describe the functional role of genes at a molecular level, and the initiatives of different consortiums to annotate complete genomes with GO terms. Earlier enrichment tools were mainly based on the analysis of GO terms, although information from other sources such as KEGG or Biocarta (www.biocarta.org) has been incorporated in more recent applications. Even though GO covers three aspects of the biology of genes: Biological Process, Cellular Component and Molecular Function, the former is the information in which researchers have been mainly interested for the functional characterization of gene lists. This is in part because biological process terms provide explicit information to interpret the biological mechanisms that may be associated to the experimental system.

Nevertheless, beyond functional information there are other properties of genes and proteins that can also be very useful to interpret biological systems, such as information related to gene expression regulation. There are different sources of regulatory information that have been incorporated in enrichment tools (9–11). In GeneCodis, we now include annotations related to regulatory information from different places. For example, miRBase (12) that contains putative targets of microRNAs, molecules that in the last few years have been shown as key regulators in many biological systems. From this database we have extracted the microRNAs associated to genes in different organisms: *B. taurus, C. elegans, D. rerio, D. melanogaster, G. gallus, H. sapiens, M. musculus* and *R. norvegicus*. Another source is transcription factors, which are key regulators that provide significant information in enrichment analysis [e.g. (5,13,14)]. Like in the case of GSEA (5) or FactorY (13), we have annotated genes from *H. sapiens, M. musculus* and *R. norvegicus* based on the information contained in TransFac database. In addition, we have used results from chip-on-chip experiments (15) to annotate genes in *S. cerevisiae* with transcription factors that bind to their promoter regions. These new annotations allow users to perform enrichment analysis with regulatory information in addition to functional information.

Besides the independent analysis of different properties of genes, the integration of heterogeneous sources of information can provide a more complete picture of the system under analysis. In this context, the new sources of information offer the possibility to mine gene lists to discover associations among regulatory and functional information. This integration is probably one of its most useful features. This is illustrated in Figure 1 with a very simple example. It shows a screenshot of the GeneCodis results from the analysis of the ribosomal gene set in KEGG. Using the information of transcription factors in the analysis we can see how co-annotations of functional and regulatory information are evaluated. For example enriched annotations contain transcription factors such as RAP1, FHL1 or SFP1 that are well known to play important roles in the regulation of ribosomal genes.

### Organisms and identifiers

To extend the scope of GeneCodis we have also increased the number of supported organisms and gene identifiers. GeneCodis works with most of the model organisms in biological research. The whole list includes *Arabidopsis thaliana, Bos taurus, Caenorhabditis elegans, Candida albicans, Danio rerio, Drosophila melanogaster, Escherichia coli, Gallus gallus, Homo sapiens, Leishmania major, Mus musculus, Rattus norvegicus, Saccharomyces cerevisiae, Schizosaccharomyces pombe* and *Vibrio cholerae*. Alternatively, to facilitate the usability of the application we have extended the type of gene identifiers that are supported, including propietary identifiers from commercial platforms such as Affymetrix, Agilent, Codelink and Illumina. The backbone of the system is now based on Ensembl and gene ids have been obtained using biomart (http://www.biomart.org). A new menu is now available for the selection of the reference list that includes arrays from the most popular manufacturers: Affymetrix, Agilent and Illumina.

### Interface

The new friendly interface has been designed to facilitate usability. Results can be explored more dynamically because they are presented in different formats, including pie charts and bar graphs images that allow users to interpret the data with a quick look (see Figure 1 for example). XML files with structured results are also provided permitting its use in data processing pipelines. Tabulated text files and html tables are also available. These different file formats allow researchers to use GeneCodis results in other applications.

Big efforts have been done to facilitate the access of GeneCodis functionality by different ways. In addition to the classical web-server access, the tool can now be used programmatically trough the SOAP Web Services technology. Using this technology, researchers can insert functional analysis in their data mining pipelines or in other bioinformatics systems in a very straightforward manner (see Figure 2 for a very simple code example). There are many advantages in the use of Web Services. It is a platform-independent and language-independent technology which makes it very adequate for loosely coupled systems working in different architectures. A complete tutorial can be found in the web site including example scripts in Ruby, Perl and PHP.

## IMPLEMENTATION

The algorithmic core of GeneCodis has been reviewed and drastically improved. There is a new implementation of the method to extract closed itemsets based on a modification of previously reported efficient techniques (16). The new algorithm can deal with large sets of annotations in a faster and more efficient way than the methodology implemented in the previous version. This is especially evident if multiple annotations are included in the same analysis or when the permutation-based test is used for the multiple hypothesis correction. In both cases the computing time has decreased drastically with respect to the previous version.

The throughput and performance of the entire system has also been improved by implementing the new algorithm and the Web Service technology in the context of a multi-grid computational environment. The new system is able to handle all submitted jobs simultaneously without queuing any of them for long periods of time. The current implementation takes advantage of one cluster with two Quad-Core Intel Xeon processors of 64 bits and two independent grid infrastructures (CyTED, http://www.cyted.org and EELA2, http://www.eu-eela.org/) integrated by the GridWay metascheduler (17).

When users submit a query through the web site, the system launches one job for the concurrent analysis and one job for each selected annotation category in the singular enrichment. All jobs will be executed in parallel. Users can also submit a query directly from a script using the Web Services. In all cases the workflow of the system is the same: a meta-scheduler determines, depending on the computational load of the cluster and the grids, in which computational environment the jobs will be executed. The jobs are queued and executed in the cluster if it is free. Otherwise, the jobs are sent to the less work-loaded grid resource. This approach represents a cost-effective alternative to improve the throughput of the application, and to guarantee its real-time performance, a critical aspect of any popular web-server application.

## CONCLUSIONS AND DISCUSSION

High-throughput experimental techniques have demonstrated to be very useful for the study of biological systems from a global perspective. In many cases, these techniques generate huge amounts of data in the form of large gene or protein lists. An essential task in this context is to translate these lists to functional information that aids researchers in the interpretation of the underlying biological processes. Although this interpretation is not always trivial, methods based on the enrichment analysis have been successfully used for this purpose.

GeneCodis is a tool designed to expand the traditional enrichment analysis of annotations by adding the possibility of evaluating not only individual terms, but also their significant combinations. Since its creation, this application has become a useful resource for the research activity. This encourages us to improve it by adding more functionality to extend its scope, performance and accessibility.

In summary, the new version of GeneCodis finds combinations of annotations and also includes the traditional singular enrichment analysis. New annotations are now available, such as microRNAs or transcription factors which help to expand the functionality of GeneCodis towards the analysis of regulatory information. Moreover, GeneCodis allows researchers to jointly analyze regulatory and functional information and to extract association patterns between both data sources. The algorithm that retrieve sets of concurrent annotations has been improved and implemented for running in a multi-grid

```ruby
require 'soap/wsdlDriver'

#### Test arguments ###
#
org = "Sc" #Saccharomyces Cerevisiae
algorithm = 1 # Concurrent analysis
test = 0 # Hypergeometric p values
correction = 1 # FDR method
minsupport = 3 # Minimum support of 3 is considered in the analysis
annotations = ["GO_Biological_Process","GO_Molecular_Function"]
reference_list = [] # whole genome
input_list = ["S000004295", "S000005284", "S000000598", "S000006205",
              "S000006183", "S000004982", "S000005662", "S000001387",
              "S000002555", "S000005668", "S000002585", "S000003736"] #input list


#### Connecting to server ###
#
WSDL_URL = "http://genecodis.dacya.ucm.es/static/wsdl/genecodisWS.wsdl"
driver = SOAP::WSDLDriverFactory.new(WSDL_URL).create_rpc_driver

### Submit tha analysis ###
#
job_id = driver.analyze(org,algorithm,test,correction,minsupport,input_list,annotations,reference_list)

### Check job state ###
#
status = driver.status(job_id)
while status == 1
  puts "Waiting ..."
  sleep 2
  status = driver.status(job_id)
end

# If error
if status < 0
  error_message = driver.info(job_id)
  raise "Finished with error #{ error_message }"
end

# Get results
results = driver.results(job_id)
puts results
```

**Figure 2.** Example of a ruby client code to invoke the GeneCodis Web Service. The access only needs three main steps: submit the analysis, ask for the job status and get the results.

environment that is able to handle all submitted jobs simultaneously. In this way the performance and the throughput of the system is enhanced. Finally, GeneCodis can be accessed programmatically by a web services interface; allowing its inclusion in data analysis pipelines.

One of the disadvantages of the modular enrichment analysis included in GeneCodis is the intrinsic redundancy of the combined functional annotations. This is caused by the unavoidable natural redundancy of the information across biological databases and by the nature of the data mining algorithm that extract all possible combinations. A method to filter out and group annotations that are related to the same biological topic is still needed. Going in the direction of creating a more self-contained application, new future releases will also include GSEA methods to allow users the selection of all possible flavours of functional analysis in the same environment.

The new version has been running since August 2008. Extensive tests have been carried out using synthetic and real datasets for which the outcome of the software is known. The diverse functionalities supported by this tool have been also fully tested by real users who have provided feedback on issues that have helped in improving the application. We hope the renewed GeneCodis will be of interest to the scientific community.

## GENECODIS AVAILABILITY

This application can be freely accessed through its main site at http://genecodis.dacya.ucm.es. A mirror has also been recently set up at the Center for Bioinformatics

in Peking University. The mirror is available at http://genecodis.cbi.pku.edu.cn/.

## ACKNOWLEDGEMENTS

The authors thank the support of Integromics, S.L. This work also makes use of results produced by the EELA-2 project (www.eu-eela.eu), co-funded by the European Commission within its Seventh Framework Programme. The authors like to acknowledge Luis Canet for his technical help. Special thanks also to Prof. Jinchu Luo from the Center for Bioinformatics at Peking University for his help in setting the Asian GeneCodis mirror. A.P.M. acknowledges the support of the Spanish Ramón y Cajal program.

## FUNDING

Spanish grants BIO2007-67150-C03-02, S-Gen-0166/2006, CYTED-505PI0058, TIN2005-5619 and PS-010000-2008-1 and European Union Grant FP7-HEALTH-F4-2008-202047. Funding for open access charge: Spanish Grant BIO2007-67150-C03-02.

*Conflict of interest statement*. None declared.

## REFERENCES

1. Ashburner,M., Ball,C.A., Blake,J.A., Botstein,D., Butler,H., Cherry,J.M., Davis,A.P., Dolinski,K., Dwight,S.S., Eppig,J.T. *et al.* (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.*, **25**, 25–29.
2. Kanehisa,M., Araki,M., Goto,S., Hattori,M., Hirakawa,M., Itoh,M., Katayama,T., Kawashima,S., Okuda,S., Tokimatsu,T. *et al.* (2008) KEGG for linking genomes to life and the environment. *Nucleic Acids Res.*, **36**, D480–D484.
3. Khatri,P. and Draghici,S. (2005) Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics,*, **21**, 3587–3595.
4. Dopazo,J. (2006) Functional interpretation of microarray experiments. *OMICS*, **10**, 398–410.
5. Subramanian,A., Tamayo,P., Mootha,V.K., Mukherjee,S., Ebert,B.L., Gillette,M.A., Paulovich,A., Pomeroy,S.L., Golub,T.R., Lander,E.S. *et al.* (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl Acad. Sci. USA*, **102**, 15545–15550.
6. Huang da,W., Sherman,B.T. and Lempicki,R.A. (2009) Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.*, **37**, 1–13.
7. Carmona-Saez,P., Chagoyen,M., Tirado,F., Carazo,J.M. and Pascual-Montano,A. (2007) GENECODIS: a web-based tool for finding significant concurrent annotations in gene lists. *Genome Biol.*, **8**, R3.
8. Carmona-Saez,P., Chagoyen,M., Rodriguez,A., Trelles,O., Carazo,J.M. and Pascual-Montano,A. (2006) Integrated analysis of gene expression by Association Rules Discovery. *BMC Bioinformatics*, **7**, 54.
9. Abascal,F., Carmona-Saez,P., Carazo,J.M. and Pascual-Montano,A. (2008) ChIPCodis: mining complex regulatory systems in yeast by concurrent enrichment analysis of chip-on-chip data. *Bioinformatics,*, **24**, 1208–1209.
10. Dennis,G. Jr, Sherman,B.T., Hosack,D.A., Yang,J., Gao,W., Lane,H.C. and Lempicki,R.A. (2003) DAVID: database for annotation, visualization, and integrated discovery. *Genome Biol.*, **4**, P3.
11. Al-Shahrour,F., Minguez,P., Tarraga,J., Montaner,D., Alloza,E., Vaquerizas,J.M., Conde,L., Blaschke,C., Vera,J. and Dopazo,J. (2006) BABELOMICS: a systems biology perspective in the functional annotation of genome-scale experiments. *Nucleic Acids Res.*, **34**, W472–W476.
12. Griffiths-Jones,S., Saini,H.K., van Dongen,S. and Enright,A.J. (2008) miRBase: tools for microRNA genomics. *Nucleic Acids Res.*, **36**, D154–D158.
13. Guruceaga,E., Segura,V., Corrales,F.J. and Rubio,A. (2009) FactorY, a bioinformatic resource for genome-wide promoter analysis. *Comput. Biol. Med.*, **39**, 385–387.
14. Xie,X., Lu,J., Kulbokas,E.J., Golub,T.R., Mootha,V., Lindblad-Toh,K., Lander,E.S. and Kellis,M. (2005) Systematic discovery of regulatory motifs in human promoters and 3′ UTRs by comparison of several mammals. *Nature*, **434**, 338–345.
15. Lee,T.I., Rinaldi,N.J., Robert,F., Odom,D.T., Bar-Joseph,Z., Gerber,G.K., Hannett,N.M., Harbison,C.T., Thompson,C.M., Simon,I. *et al.* (2002) Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science*, **298**, 799–804.
16. Zaki,M.J. and Hsiao,C.-J. (2005) Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure. *IEEE Trans. Knowledge Data Eng.*, **17**, 12.
17. Huedo,E., Montero,R.S. and Llorente,I.M. (2005) The GridWay framework for adaptive scheduling and execution on grids. *SCPE*, **6**, 8.

## 4.2 SENT: semantic features in text

**Reference:** **Vazquez, M.**, Carmona-Saez, P., Nogales-Cadenas, R., Chagoyen, M., Tirado, F., Carazo, J. M. & Pascual-Montano, A. D. (2009), 'SENT: semantic features in text', *Nucleic Acids Research* **37**(Web Server issue), W153–W159. (JCR 2008 impact factor: **6.878**).

**Summary:** Some of the most common functional analysis methods make use of curated resources, such as GO terms or KEGG pathways, as a source for information. Literature mining offers an interesting alternative to curated resources in that it goes directly to the information source skipping the curation process, which can be slow, expensive and even biased. Additionally, annotation enrichment analysis often serves only as a guide for a literature exploration process where the researcher synthesises the relevant information from previously published articles, so an analysis that mines the literature should also be able to aid its examination by the researcher.

Dimensionality reduction is a very popular approach for feature extraction in text mining. Here we use non-negative matrix factorization (NMF), which we find to be a very interesting alternative to other dimensionality reduction methods like principal component analysis. In particular, the non-negativity of feature and profile vectors, and the sparsity it induces, allows for a much a better interpretability and insight.

In this work we present a literature mining tool that uses NMF to cluster and summarize an input list of genes, supporting input lists for eight different model organisms. Additionally the application can be used as a literature exploration tool, where relevant articles are sorted according to their specificity to the semantic features detected in the analysis.

The tool offers a programmatic interface based on SOAP web services and a rich and user friendly online interface. Additionally, the applications source code is integrated in the Rbbt development framework.

**Nucleic Acids Research Advance Access published May 20, 2009**

*Nucleic Acids Research, 2009, 1–7*
*doi:10.1093/nar/gkp392*

# SENT: semantic features in text

**Miguel Vazquez[1], Pedro Carmona-Saez[2], Ruben Nogales-Cadenas[2], Monica Chagoyen[3], Francisco Tirado[2], Jose Maria Carazo[3] and Alberto Pascual-Montano[2,*]**

[1]Software Engineering Department, [2]Computer Architecture Department, Complutense University and [3]Biocomputing Unit, National Center for Biotechnology, CNB-CSIC, Madrid, Spain

Received January 31, 2009; Revised April 20, 2009; Accepted April 30, 2009

## ABSTRACT

**We present SENT (semantic features in text), a functional interpretation tool based on literature analysis. SENT uses Non-negative Matrix Factorization to identify topics in the scientific articles related to a collection of genes or their products, and use them to group and summarize these genes. In addition, the application allows users to rank and explore the articles that best relate to the topics found, helping put the analysis results into context. This approach is useful as an exploratory step in the workflow of interpreting and understanding experimental data, shedding some light into the complex underlying biological mechanisms. This tool provides a user-friendly interface via a web site, and a programmatic access via a SOAP web server. SENT is freely accessible at http:\\sent.dacya.ucm.es.**

## INTRODUCTION

Advances in biomedical technologies such as DNA microarrays have enabled researchers to identify a large number of molecules simultaneously, opening the path to study biological systems from a global perspective. These techniques have been routinely used in research labs all around the world, generating huge amounts of data. The methods used to analyze and process this data have evolved significantly in the recent years, to the point that they can be considered mature. The interpretation of the results of the analysis, however, still remains one of the main challenges in bioinformatics, mainly due to the inherent complexity of biological systems.

One of the most notable initiatives to help the interpretation of a list of genes is the Gene Ontology (GO) (1). Several approaches use GO annotations to discover what biological terms are significantly enriched in a list of genes. This is an example of an annotation based approach to functional interpretations, a good review of the topic can be found in (2). Annotation based approaches provide a fast, easy and statistically sound interpretation of a list of genes. Although this information is extremely useful for the analysis of gene sets, its scope is limited by structured vocabularies and curated annotations.

Literature mining offers an interesting alternative to annotation based methods. The rationale behind it is that it contains much richer information about the function of genes that can be captured in structured vocabularies. Biomedical literature covers almost all aspects of biology and biochemistry, and with almost no limit to the types of information that may be recovered through careful and exhaustive mining (3). Many researchers have focused their attention in the use of text mining, with methodologies that go from determining protein–protein interactions from biomedical texts (4–7), to providing summary descriptions for genes or determining their similarities (8–12). Even though lots of works in this area have been reported, the practical use by the scientific community is hindered by the lack of efficient and easy to use software.

In a previous work we introduced a technique, based on Non-negative Matrix Factorization (NMF), to extract semantic features from the biomedical literature associated to a list of genes (13). The use of the term 'semantic features' was first introduced by Lee and Seung (14) to describe the NMF factors that group semantically related words, and has been used in this work to follow this nomenclature. These semantic features were able to characterize the biological meaning of the gene list by capturing be main biological topics that where discussed in the articles. Relationships between the genes could be established on the basis of their relationship to these semantic features. The technique has shown a great potential to analyze large literature collections, and has centered the attention of several works in the field (15–17).

This contribution presents a working, usable implementation of a methodology based on (13). SENT (semantic features in text) allows users to explore the biomedical literature associated to a list of genes by summarizing its contents in semantic features, and allowing the user to browse intelligently the relevant articles. It also includes several assisting functionalities like GO enrichment analysis, provided by the GENECODIS web server (18). SENT offers its services through an easy to use web site, and

*To whom correspondence should be addressed. Tel: +34 913 944 420; Fax: +34 913 944 687; Email: pascual@fis.ucm.es

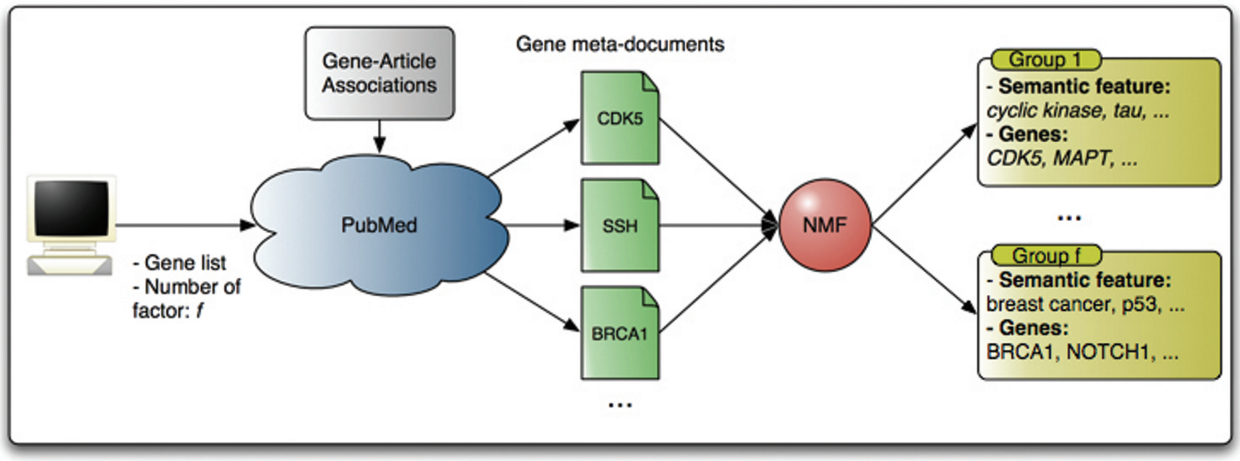


**Figure 1.** General schematic view of SENT. A set of meta-documents (merged documents associated to each gene) are decomposed by the NMF algorithm to produce groups of semantic features (sets of semantically related words) with their associated genes.

through an SOAP API that allows researchers to use it inside their own scripts and workflows.

## METHODS

A general overview of the data analysis workflow implemented in SENT is presented in Figure 1. The input of the system is a set of gene identifiers and the number of semantic features (factors) to use in the NMF analysis. Titles and abstracts from articles associated to each gene are used to produce a meta-document and from all gene meta-documents a term frequency matrix is created. This matrix is then analyzed by means of the NMF algorithm yielding a set of semantic features and a way to associate genes to these semantic features.

The collection of articles used in the analysis is built into an index. This index can be queried to retrieve articles that mention certain terms. In particular, it can be used to find the articles that are most relevant to each semantic feature and, by extension, most relevant to understand the list of genes. This way the user can clarify and ground his interpretation of the semantic features by contrasting the literature. Coupled with the GO enrichment analysis, also provided in the web site, SENT serves as a guide in the examination of the literature.

Therefore, the methodology in SENT can be divided in three steps: Finding the articles to associate to each gene, processing the text into a vector representation that can be analyzed, and finally, analyzing that data to find the semantic features. The next sections describe these three steps in more detail, plus the indexing of documents for the literature examination.

### Literature retrieval

To determine the set of articles to associate to each gene, SENT uses two sources:

(i) Curated resources. Databases such as GeneRIF (19) and the GO (1) already provide gene-articles associations. GeneRIF (Gene Reference Into Function) links any article about the basic biology of a gene or protein to the corresponding entry in Entrez Gene, while GO includes references to articles to support the associations of genes to GO annotations.

(ii) Associations automatically derived from the literature. SENT uses PubMed to find articles in which a given gene is explicitly mentioned in the abstract. To this end, a PubMed query is executed containing the organism name and, optionally, to narrow down the search, the words 'gene' or 'protein'. This retrieves a broad corpus of literature in which Named Entity Recognition (NER) and Normalization methodologies are used to find explicit mentions to genes in the article abstracts. NER is the process of finding mentions of entities in text, which in the particular context of gene names is called Gene Mention Recognition. Finding mentions to genes is only the first part of the problem, once the mentions are found the system needs to determine the specific gene they refer to. This problem is known as Gene Mention Normalization, and is aggravated by the often ambiguous ways in which genes are mentioned in free text. Both gene mention and normalization have received a considerable attention by the bioinformatics text mining community, and are a central task on several competitions, such as TREC and, specially, BioCreative (20,21). In SENT we have implemented solutions to both these problems, following many of the central ideas on the state of the art methodologies. Since this is a computationally time consuming process, the collection of articles examined for each organism is limited to the 30 000 most recently published.

### Data processing

We construct a meta-document for each gene merging the text from the titles and abstracts of all its

associated articles. A vector representation (22) is generated from these meta-documents following a standard text-mining procedure: We remove words appearing in a list of very common English words, called a stop-word list. The rest of the words are reduced to their stem using the Porter stemmer (23), to group related words with a common stem, like 'telomer' and 'telomeric'. The stems resulting from the previous step are collected individually (unigrams) and in all overlapping pairs (bigrams) to form the bag-of-words representation of the document, called that way because the order of appearance the terms is no longer considered.

The number of individual terms that can appear in the bag-of-words is very large, and this would cause a problem in most text-mining applications. To select the most useful subset, we apply a filter to remove the terms that appear in too many documents ($\geq$80%), as they can be seen as a background, non-informative signal, or in to few documents ($\leq$0.5%), as they are to rare to be useful. We score each of the remaining terms using the Term Frequency, Inverse Document Frequency (TF–IDF) measure (24). This measure balances favoring a words frequency (TF, if it appears many times) and its specificity (IDF, if it does so in only a few documents). With this score we do another filtering to select only the top 3000 best terms ($w$). The genes are represented as $w$-dimensional vectors where each coefficient is the frequency of a particular term in the genes meta-document, multiplied by the IDF value for that term.

The above filtering process was done by looking at the frequency of appearance of each term in a collection of documents. SENT supports two options to define what this collection of document should be:

- the meta-documents for the complete list of genes in the organisms' genome. This will select broad and general terms for the word vectors.
- The meta-documents for just the genes in the input query. The terms selected are those important to the specific query, and thus, may be of a higher level of detail.

The first option is the default way the analysis is done, what we call a *standard analysis*. The second option is used in what we call the *fine-grained analysis*. There is an additional type of analysis, called the *custom analysis*, in which the user may provide a list of entities with their associated articles. The custom analysis allows the user to explore genes from unsupported organisms or even entities of other nature, like diseases, authors, journals, etc. The custom, like the fine-grained, also uses the collection of articles in the actual query to perform the filtering.

Note that in fine-grained and the custom analysis the computations must be done on-line, as opposed to the standard analysis in which the vectors may be precomputed. These computations are actually the most resource consuming part of the process, and may delay the jobs considerably.

## Analysis

The previous step left us with a collection of $n$ $w$-dimensional word-vectors representing the $n$ genes in the input list query. The $w$ dimensions represent the $w$ selected terms, ideally 3000. These vectors are arranged as columns of a matrix M of dimensions $wxn$. We use NMF to factor the **M** matrix into two non-negative lower rank ($f$) matrices

$$\mathbf{M} = \mathbf{WH}$$

where $f$ is the number of factors or semantic features. **W** is a $w$x$f$ projection matrix and **H** is the coefficient matrix of dimension $f$x$n$. The column vectors of the **W** projection matrix are called semantic features, due to the fact that they are collections of semantically related terms. The columns of **H** project the original gene vectors in this new low rank space spanned by the **W** matrix. These vectors are known as the gene profiles, since they can be seen as expressing the genes as combinations of the semantic features. To calculate the NMF model we use the bioNMF web-server application reported in (25).

The NMF algorithm does not necessarily find the best solution. To cope with this situation several initializations strategies have been proposed to improve the convergence rate and eventually find better solutions under certain conditions. The NNDSVD (26) and CENTROID (27) methods are good examples that have proved to be suitable for this problem and will deserve our attention in future versions of this application.

In SENT we decided to take advantage of the non-deterministic nature of NMF to assess the stability of the factorizations at different ranks, using the approach introduced in (13). This approach is based on the collection of a series of results from repeated executions with random initializations. The rationale behind this is that strong signals that are present in the data will be captured by some factor and maintained from execution to execution. Weak signals, on the other hand, or inappropriate choice of the number of factors, will result in noisy factors across executions and won't be maintained. The extent to which factors are reproduced between executions can be used as a measure of appropriateness of the factorizations. We use the cophenetic correlation coefficient as well as a clustering heat-map as assessment of this appropriateness. The actual factors reported by the application are the results of clustering together the factors resulting from 10 separate runs, attending to their semantic features, and averaging both the semantic features and the correspondent gene semantic profiles for each cluster. We use as many clusters as factors originally specified. We will use the terms 'factors' and 'semantic features' to refer to the correspondent cluster averages for simplicity.

We select the most representative terms for each semantic feature as its description. A more sensible selection of terms picks the ones that are both important and exclusive for each factor. We use the following score function for the score of a term $t$ in the $i$ semantic feature

$$S_{i,j} = W_{t,i} / \left\{ average\left(W_{t,j}\right) | \forall_{j \neq i} \right\}$$

The representative terms are those with the 15 best scoring.

In this context, the well-known singular value decomposition (SVD) can also be used to find a low rank approximation of a data matrix. This technique and its application in texts is know as latent semantic indexing (LSI) (28), and show some similarity with the methodology described here.

The main benefit of NMF over LSI is interpretability. The non-negativity of the factors and the projection of the rows in the factor space make them easier to interpret as opposed to when there are negative coefficients as it happens with SVD-based approaches. In addition; the factors found by SVD are designed to incrementally capture all the variance in the data and thus larger factors explain most of data while the rest of factors explain the residuals. On the contrary NMF factors try to capture local signals, and thus they might have equivalent importance. Furthermore, NMF factors show certain degree of overlap as they are not required to be orthogonal. These characteristics make NMF factors more attractive than SVD for interpretability

### Literature indexing

SENT builds an index with the titles and abstracts from all the articles associated to the genes in the input list using Ferret, a version of the popular indexing engine Lucene for the Ruby programming language. The index can be queried and will assign scores to the articles. These scores are used, for example, to sort the articles for relevance to a semantic feature.

## SOFTWARE USAGE

The input for the application consists in a list of gene identifiers for a given organism. SENT supports ids from several databases, which are listed under the 'Supported ids' in the help section. Currently SENT supports the following organisms: *Candida albicans*, *Caernorhabditis elegans*, *Homo sapiens*, *Arabidopsis thaliana*, *Rattus norvegicus*, *Mus musculus*, *Saccharomyces cerevisiae* and *Saccharomyces pombe*.

SENT allows users to simultaneously explore the data at different resolutions determined by different ranks in the factorization, which must be between 2 and 32. The factorizations at the selected ranks will be produced sequentially in increasing order and made available as they are finished. Once all the analysis are completed another batch of factorizations can be scheduled, either to try with other ranks or to recalculate the results for a given value.

The creation of the literature index can be also scheduled to build it in the same analysis or to leave it for latter. In addition, the application offers two types of analysis: standard and fine grained analysis, being the former the default option. The 'fine grained' analysis produces semantic features whose terms are more specific, but it may take some time (average estimations are from 15 to 30 min). The job name can be used to access the results page at any latter time. Also, if an email is specified it will be used to notify the completion of the first factorization, this is useful for lengthy fine grained jobs.

Once the results are available users may explore the different resolutions to find which of them show a better stability. This can be done looking for high cophenetic correlation values for each factorization or by looking at the heat-maps. At this point one can detect that certain genes are sparsely annotated and produce clear blocks of useless terms relating to analysis methodologies, for example gels, microscopy, spectrometry, etc. If this is the case the genes can be removed and the analysis can be redone. Also, interesting genes may bundle together in large general groups; instead of increasing the resolution of the analysis an interesting option would be to use these genes in a separate analysis. Because the gene list may use some cleaning, it is advised to leave the fine grained analysis and build the literature index (both costly operations) for a second analysis job, after these issues have been assessed.

Once a factorization is found, the different factors in the results provide a general idea of the topics latent in the literature. From this point there are several alternatives to further interpret the groups of genes associated to each factor. One option is to examine the 'Gene Details' page to view results from a GO term enrichment analysis of the genes in the group. Another option is to look at the literature explorer for that group. The literature explorer will show the articles related to the genes in the group sorted for relevance to the terms in the semantic feature. To determine the role of a certain gene in the group we may visit the outside description page (e.g. Entrez Gene for the human), or use the literature explorer to review the articles related to that gene. It is also worth noting that the literature explorer also supports custom queries to the index, which will be used to rank the collection of articles according to that particular query. This is useful to find how a gene relates to each particular term in the feature for example.

With these functionalities the user should be able to find interesting nuggets of knowledge in the data and retrieve the relevant articles that support the findings.

## USE CASE

To exemplify the type of results that can be obtained we present the analysis of 50 genes reported by Homayouni *et al.* (29). The 50-gene set is based on the manual selection of genes related to cancer biology, Alzheimer's disease and development, and includes five genes that are involved in the Reelin pathway (RELN, VLDLR, LRP8, DAB1 and FYN). Reelin is an extracellular protein that controls neuronal positioning, formation of laminated structures and synapse structure in the developing central nervous system. In addition some components in the Reelin signaling pathway are associated with Alzheimer disease.

SENT offers this list of genes as the *H. sapiens* example dataset, and can be loaded from the main form. Table 1 shows the semantic features and associated genes for reelin dataset from one analysis in which four groups were formed using the fine-grained analysis.

**Table 1.** Reelin dataset summarized into four groups using fine-grained analysis

| | |
|---|---|
| 1 | **Terms:** cdk5, tau, cyclin depend, gsk, calpain, gsk 3beta, depend kinas, 3beta, microtubul, cyclin, ser, cdc2, microtubul associ, cdk2, kinas activ<br>**Genes:** CDK5, CDK5R1, CDK5R2, FYN, MAPT |
| 2 | **Terms:** hedgehog, brca1, wnt, kit, breast, egfr, sonic, myc, breast cancer, basal cell, ptc, p53, notch, patch, renal<br>**Genes:** ABL1, ATOH1, BRCA1, BRCA2, DLL1, DNMT1, EGFR, ERBB2, ETS1, FOS, GLI1, GLI2, GLI3, JAG1, KIT, MYC, NOTCH1, NRAS, PAX2, PAX3, PTCH1, ROBO1, SHH, SMO, SRC, TGFB1, TP53, WNT1, WNT2, WNT3 |
| 3 | **Terms:** reelin, dab1, apo, lrp, lipoprotein, apolipoprotein, densiti lipoprotein, lipoprotein receptor, low densiti, ldl, macroglobulin, ldl receptor, schizophrenia, receptor relat, apolipoprotein apo<br>**Genes:** A2M, APOE, DAB1, LRP1, LRP8, RELN, VLDLR |
| 4 | **Terms:** app, amyloid, presenilin, fe65, precursor protein, amyloid precursor, abeta, secretas, gamma secretas, alzheim, alzheim diseas, beta amyloid, protein app, ptb, amyloid beta<br>**Genes:** APBA1, APBB1, APLP1, APLP2, APP, PSEN1, PSEN2, SHC1 |

The first group contains the cyclin-dependent kinase 5 and receptors and two additional genes, FYN and MAPT. These genes were also grouped together in the original work of Homayouni *et al.* (29). The semantic feature contains terms related to kinases and terms such as 'tau' and 'microtubl associ'. Indeed, Cdk5 is one of the major kinases that phosphorylate the microtubule-associated protein tau, which is encoded by the MAPT gene. Exploring the literature associated to this group we found articles that discus the role of the cyclic dependent kinases in the phosphorylation of the tau protein among the first ranked abstracts.

The second group is related with cancer and development while the third group contains genes from the Reelin pathway (except FYN) and some genes related with Alzheimer. The fyn kinase has been largely associated with cancer and its association to Reelin has only recently been demonstrated. The terms in this semantic feature also provide insights into the role of reelin in 'binding to lipoprotein receptors and especially low density lipoprotein receptors' as claimed in the highest ranked article (30) in the literature associated to this feature. In addition, there is apolipoprotein E blocks the interaction of Reelin with its receptors and is also considered a risk factor for late-onset Alzheimer disease.

Finally, the fourth group is associated to a semantic feature that clearly captured terms related with Alzheimer disease and most of the genes originally included in this category.

### Comparison with similar tools

Other tools that enable literature-based knowledge discovery include GenCLiP (31) and FAUN (17). GenCLiP is a Windows application that offers clustering of genes and terms based on the literature. It also generates gene networks based on co-occurrence of genes in the literature associated to certain keywords. The clustering is done to identify sets of terms related to sets of genes as described in Chaussabel and Sher (8). GenClip allows users to find specific information based on interesting keywords. The clustering step is comparable in principle to the analysis in SENT, while the network generation based on co-occurrence could be considered a different application. Compared with SENT we think both applications focus on different goals and could complement each other. One of the main benefits of SENT over GenCLiP is processing time and simplicity in the interaction with the user.

FAUN on the other hand is a NMF-based text mining tool similar to SENT. The FAUN application is available at http://grits.eecs.utk.edu/faun (17) and at the time of this writing it only contains models for the same 50 gene dataset from Homayouni *et al.* (29) that was used in our use case. They provide visualizations at three resolutions; high, low and medium, meaning 10, 15 and 20 factors respectively. However, FAUN lacks the possibility to explore new datasets in almost real-time manner. With regards to the actual features extracted, the comparison of both applications seems to show the most relevant information at the first ranked items (results not shown).

Another important difference among both tools is the way in which genes are associated to features. In FAUN this is done in a fuzzy way, where each gene can belong to several groups at the same time, while in SENT genes are assigned to one and only one group. Both applications provide with tools to investigate the results further. In this area FAUN stands out showing the relation of each gene to each of the terms of the feature and also shows a gene-gene correlation matrix. In SENT we can use the literature explorer to find articles associated to the genes containing the terms of interest. In addition, SENT provides a collection of result files for downloading, in particular the semantic profiles for each gene, which can be used to calculate the gene–gene correlation matrix.

One of the most interesting features in FAUN is the sentence highlighting, which marks relevant sentences for each gene in relation to each of the features. SENT offers a similar functionality that highlights, not sentences, but complete abstracts. Both alternatives are based on the same idea but work at different resolutions.

## SOAP WEB SERVER

All the jobs that are issued from the web site are forwarded to a SOAP server, so the web site can be seen as a front-end to this server. The SOAP server offers an API that can be used to access the functionality programmatically from other work-flows or scripts. Any job issued in the SOAP web server can be examined in the web site using its unique identifier. The web site help section includes an API description, the WSDL file, which is an XML file that most SOAP libraries can use to automatically set up a client for the server, and an example script, that can be used as a command line tool to launch jobs.

## CONCLUSIONS

We have developed SENT, a web-based tool for the functional analysis of gene lists extracted from the biomedical literature. The main motivation to construct this tool is the lack of available user-friendly software to automatically analyze large amounts of documents related to genes or proteins. This is a very complex research area that is still in its infancy and we are all aware of the fact the methodologies to solve the full-text mining problematic are still under development. However, precisely because of this, any contribution in this area is more than welcome.

This tool offers several advantages in the area of biomedical text mining: first, SENT is oriented to give researchers a global functional picture of their genes of interest by summarizing the associated literature content in a small set of semantic topics. Second, SENT is able to categorize the list of genes or proteins according to these topics and also associated to Biological Processes terms in GO. Finally this functionality, and the way it is implemented using web-services technology, allows researchers to easily include this analysis into their workflows, providing their research with one more piece of information to be taken into account.

As any other system SENT is not without limitations and we are working in improving both the results and their interpretability. Several ideas gathered from FAUN and GenCLiP are been considered, as well as the possibility of automatically mapping semantic features to biomedical dictionaries or ontologies. We will work to have future versions of SENT updated with these enhancements.

We hope this application is useful to the biomedical community.

## ACKNOWLEDGEMENTS

The authors thanks the support of Integromics, S.L. A.P.M. acknowledges the support of the Spanish Ramón y Cajal program.

## FUNDING

Spanish grants [BIO2007-67150-C03-02, S-Gen-0166/2006, TIN2005-5619, PS-010000-2008-1]; European Union Grant [FP7-HEALTH-F4-2008-202047]. Funding for open access charge: Spanish Grant number BIO2007-67150-C03-02.

*Conflict of interest statement*. None declared.

## REFERENCES

1. Ashburner,M., Ball,C., Blake,J., Botstein,D., Butler,H., Cherry,J., Davis,A., Dolinski,K., Dwight,S., Eppig,J. *et al.* (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat. Genet., **25**, 25–29.
2. Huang da,W., Sherman,B.T. and Lempicki,R.A. (2009) Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.*, **37**, 1–13.
3. Shatkay,H. and Feldman,R. (2003) Mining the biomedical literature in the genomic era: an overview. *J. Comput. Biol.*, **10**, 821–855.
4. Blaschke,C., Andrade,M.A., Ouzounis,C. and Valencia,A. (1999) Automatic extraction of biological information from scientific text: protein-protein interactions. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **1999**, 60–67.
5. Jenssen,T.K., Laegreid,A., Komorowski,J. and Hovig,E. (2001) A literature network of human genes for high-throughput analysis of gene expression. *Nat. Genet.*, **28**, 21–28.
6. Wren,J.D. and Garner,H.R. (2004) Shared relationship analysis: ranking set cohesion and commonalities within a literature-derived relationship network. *Bioinformatics*, **20**, 191–198.
7. Hoffmann,R. and Valencia,A. (2005) Implementing the iHOP concept for navigation of biomedical literature. *Bioinformatics*, **21**, 252–258.
8. Chaussabel,D. and Sher,A. (2002) Mining microarray expression data by literature profiling. *Genome Biol.*, **3**, 1–16.
9. Jelier,R., Jenster,G., Dorssers,L.C.J., Wouters,B.J., Hendriksen,P.J.M., Mons,B., Delwel,R. and Kors,J.A. (2007) Text-derived concept profiles support assessment of DNA microarray data for acute myeloid leukemia and for androgen receptor stimulation. *BMC Bioinformatics*, **8**, 14.
10. Raychaudhuri,S., Schütze,H. and Altman,R.B. (2002) Using text analysis to identify functionally coherent gene groups. *Genome Res.*, **12**, 1582–1590.
11. Huang,Z.X., Tian,H.Y., Hu,Z.F., Zhou,Y.B., Zhao,J. and Yao,K.T. (2008) GenCLiP: a software program for clustering gene lists by literature profiling and constructing gene co-occurrence networks related to custom keywords. *BMC Bioinformatics*, **9**, 308.
12. Frijters,R., Heupers,B., van Beek,P., Bouwhuis,M., van Schaik,R., de Vlieg,J., Polman,J. and Alkema,W. (2008) CoPub: a literature-based keyword enrichment tool for microarray data analysis. *Nucleic Acids Res.*, **36**, W406.
13. Chagoyen,M., Carmona-Saez,P., Shatkay,H., Carazo,J.M. and Pascual-Montano,A. (2006) Discovering semantic features in the literature: a foundation for building functional associations. *BMC Bioinformatics*, **7**, 41.
14. Lee,D.D. and Seung,H.S. (1999) Learning the parts of objects by non-negative matrix factorization. *Nature*, **401**, 788–791.
15. Pehkonen,P., Wong,G. and Toronen,P. (2005) Theme discovery from gene lists for identification and viewing of multiple functional groups. *BMC Bioinformatics*, **6**, 16.
16. Heinrich,K.E., Berry,M.W. and Homayouni,R. (2008) Gene tree labeling using nonnegative matrix factorization on biomedical literature. *Comput. Intelligence and Neuroscience*, **2008**, 12.
17. Tjioe,E., Berry,M. and Homayouni,R. (2008) *First Workshop on Data Mining in Functional Genomics, IEEE International Conference on Bioinformatics and Biomedicine*, November 3–5, 2008, Philadelphia, pp. 185–192.
18. Carmona-Saez,P., Chagoyen,M., Tirado,F., Carazo,J.M. and Pascual-Montano,A. (2007) GENECODIS: a web-based tool for finding significant concurrent annotations in gene lists. *Genome Biol.*, **8**, R3.
19. Mitchell,J.A., Aronson,A.R., Mork,J.G., Folk,L.C., Humphrey,S.M. and Ward,J.M. (2003) Gene indexing: characterization and analysis of NLM's GeneRIFs. *AMIA\ldots Annu. Symp. Proc. [electronic resource]*, **2003**, 460.
20. Yeh,A., Morgan,A., Colosimo,M. and Hirschman,L. (2005) BioCreAtIvE task 1A: gene mention finding evaluation. *BMC Bioinformatics*, **6**, 1.
21. Wilbur,J., Smith,L. and Tanabe,L. (2007) Biocreative 2. gene mention task. *Proc. Second BioCreative Challenge Eval. Workshop*, **1**, 7–16.
22. Salton,G., Wong,A. and Yang,C. (1975) A vector space model for automatic indexing. *Commun. ACM*, **18**, 613–620.
23. Porter,M. (1980) An algorithm for suffix stripping. *Program*, **14**, 130–137.
24. Sparck,J.K. (1988). In Willett,P. (ed.) A statistical interpretation of term specificity and its application in retrieval. *Document Retrieval Systems, Taylor Graham Series in Foundations of Information Science*, Vol. 3, Taylor Graham Publishing, London, UK, pp. 132–142.

25. Mejia-Roa,E., Carmona-Saez,P., Nogales,R., Vicente,C., Vazquez,M., Yang,X.Y., Garcia,C., Tirado,F. and Pascual-Montano,A. (2008) bioNMF: a web-based tool for non-negative matrix factorization in biology. *Nucleic Acids Res.*, **36**, W523–W528.
26. Boutsidis,C. and Gallopoulos,E. (2008) SVD based initialization: a head start for nonnegative matrix factorization. *Pattern Recogn.*, **41**, 1350–1362.
27. Wild,S., Curry,J. and Dougherty,A. (2004) Improving non-negative matrix factorizations through structured initialization. *Pattern Recogn.*, **37**, 2217–2232.
28. Deerwester,S., Dumais,S., Furnas,G.W., Landauer,T.K. and Harshman,R. (1990) Indexing by latent semantic analysis. *J. Am. Soc. Inform. Sci.*, **41**, 391–407.
29. Homayouni,R., Heinrich,K., Wei,L. and Berry,M.W. (2005) Gene clustering by latent semantic indexing of MEDLINE abstracts. *Bioinformatics*, **21**, 104–115.
30. D'Arcangelo,G., Homayouni,R., Keshvara,L., Rice,D., Sheldon,M. and Curran,T. (1999) Reelin is a ligand for lipoprotein receptors. *Neuron*, **24**, 471–479.
31. Huang,Z.X., Tian,H.Y., Hu,Z.F., Zhou,Y.B., Zhao,J. and Yao,K.T. (2008) GenCLiP: a software program for clustering gene lists by literature profiling and constructing gene co-occurrence networks related to custom keywords. *BMC Bioinformatics*, **9**, 308.

## 4.3 BioNMF: a web-based tool for nonnegative matrix factorization in biology

**Reference:** Mejía-Roa, E., Carmona-Saez, P., Nogales, R., Vicente, C., **Vazquez, M.**, Yang, X. Y., García, C., Tirado, F. & Pascual-Montano, A. D. (2008), 'bioNMF: a web-based tool for nonnegative matrix factorization in biology', *Nucleic Acids Research* **36**(Web Server issue), W523–W528. (JCR 2008 impact factor: **6.878**).

**Summary:** Non-negative matrix factorization (NMF) is a dimensionality reduction strategy that has several advantages over other alternatives such as principal component analysis (PCA). Although the approximation would not be as close as PCA, the non-negative property of the factors and the factor profiles, and the sparseness it induces on these vectors, provide better interpretability. This interpretability is due to the possibility of factors being seen as a relatively sparse and positive combination of elements, as opposed to principal components, which are dense combinations of elements with positive and negative coefficients. This degree of interpretability is very appropriate for applications such as text mining or gene expression, and was, in fact, used in the SENT literature mining tool.

The NMF algorithm is very computationally intensive, particularly the task of finding the appropriate number of features to use in the analysis by means of comparing the cophenetic correlation coefficient for different values. The wide applicability of this algorithm called for an application to perform these analyses that offered speed and a friendly interface.

This work offers an implementation of several alternative versions of the NMF algorithm that benefits from the grid or computer clusters.

The functionality is offered via SOAP web service and online interfaces.

**Nucleic Acids Research Advance Access published May 30, 2008**

*Nucleic Acids Research, 2008, 1–6*
*doi:10.1093/nar/gkn335*

# bioNMF: a web-based tool for nonnegative matrix factorization in biology

**E. Mejía-Roa[1], P. Carmona-Saez[2], R. Nogales[1], C. Vicente[1], M. Vázquez[3], X. Y. Yang[1], C. García[1], F. Tirado[1] and A. Pascual-Montano[1,*]**

[1]Computer Architecture Department, Complutense University, 28040, [2]Integromics, S.L. Calle Darwin 3. 28049 and [3]Software Engineering Department, Complutense University, 28040 Madrid, Spain

Received January 30, 2008; Revised April 18, 2008; Accepted May 10, 2008

## ABSTRACT

**In the last few years, advances in high-throughput technologies are generating large amounts of biological data that require analysis and interpretation. Nonnegative matrix factorization (NMF) has been established as a very effective method to reveal information about the complex latent relationships in experimental data sets. Using this method as part of the exploratory data analysis, workflow would certainly help in the process of interpreting and understanding the complex biology mechanisms that are underlying experimental data. We have developed bioNMF, a web-based tool that implements the NMF methodology in different analysis contexts to support some of the most important reported applications in biology. This online tool provides a user-friendly interface, combined with a computational efficient parallel implementation of the NMF methods to explore the data in different analysis scenarios. In addition to the online access, bioNMF also provides the same functionality included in the website as a public web services interface, enabling users with more computer expertise to launch jobs into bioNMF server from their own scripts and workflows. bioNMF application is freely available at http://bionmf.dacya.ucm.es.**

## INTRODUCTION

The analysis of complex data sets generated by *–omics* technologies requires the use of statistical and data mining techniques able to find natural group structures in the data. Different data mining methods have shown to be very useful in providing significant information for hypothesis formulation and discovery of biological patterns. Clustering algorithms or matrix factorization techniques, such as PCA or SVD, are among the most popular tools for the exploratory analysis of high-dimensional biological datasets.

*Nonnegative matrix factorization* (NMF) (1) is one of such techniques that, although relatively new, is increasingly used in biomedical sciences. It has gained a lot of popularity in the scientific community due to its capability of providing new insights and relevant information about the complex latent relationships in high-dimensional biological data sets. In the particular case of biomedical sciences, several successful studies have been conducted using this method and some of its variants. For example, NMF has been successfully applied to gene-expression analysis (2–5), scientific literature mining (6,7), proteomics, metabolomics (8,9), sequence analysis (10) or neurosciences (11), among others.

Due to the increasing interest on this technique by the bioinformatics community, several standalone applications and code in different programming languages have been developed to support NMF analysis and related alternatives, in special for the biomedical field. In 2006, we introduce one of such standalone applications, *bioNMF* (12), which implements the NMF methodology in different analysis contexts to support some of the most popular applications of this new methodology. This includes clustering and biclustering of gene-expression data and sample classification.

Even if standalone applications play their role in the research process, online web tools are clearly the preferred option for most of the users, because no resources and computational expertise are required to run a scientific analysis.

In this work, we propose a user-friendly, web-based tool that implements the same methodologies present on the previous standalone application (12). In addition, this web tool offers new improvements to process big data sets in a distributed computing environment without the usage complexity present on this kind of systems. It also provides an automated access to external applications through a web services interface.

*To whom correspondence should be addressed. Tel: +34 913944420; Fax: +34 913944687; Email: pascual@fis.ucm.es

To the best of our knowledge, this is the first web-based dedicated application for NMF. This new tool is freely available at http://bionmf.dacya.ucm.es/.

## FEATURES AND FUNCTIONALITY

Experimental biological information, like for example gene expression, is usually represented and stored as a numerical data matrix, where observations or genes are stored in rows and conditions, experiments or samples are represented in columns. In this case, each cell corresponds to the expression value of a gene in a specific experimental condition.

Formally, the nonnegative matrix decomposition can be described as $\mathbf{V} \approx \mathbf{WH}$, where $\mathbf{V} \in \mathbb{R}^{m \times n}$ is a positive data matrix with $m$ variables and $n$ objects, $\mathbf{W} \in \mathbb{R}^{m \times k}$ are the reduced $k$ basis vectors or factors and $\mathbf{H} \in \mathbb{R}^{k \times n}$ contains the coefficients of the linear combinations of the basis vectors needed to reconstruct the original data. The number of factors ($k$) is generally chosen so that it takes a value less than $n$ and $m$. The distinctive attributes of NMF with respect to other factorization models are the nonnegativity constraints imposed on **V**, **W** and **H**. In this way, only additive combinations of **W** and **H** are possible, which induces not only an effective dimensionality reduction but also a more interpretable information (1). Figure 1 shows a graphic representation of the model in the case of gene-expression data.

The bioNMF online tool provides a functionality to cover some of the most important applications of the NMF algorithm. More particularly in biology (2,3,6,7,10,13). This is achieved through three different modules: *Sample Classification*, *Standard NMF* and *Biclustering Analysis*.

The *Sample Classification* module implements the method proposed by Brunet *et al.* (2) to determine the most suitable number of sample clusters in a given data set and to group the data samples into $k$ clusters, being $k$ the best factorization rank within a given input range. This method is probably one of the most used methods in the field to estimate the best factorization rank.

### Standard NMF

This module performs the classical NMF factorization using the algorithm proposed by Lee and Seung in 1999 (1). This wide-ranging module is not specifically focused to any particular analysis, but more generally oriented to any potential application that might use this factorization method. As a new feature with respect to the previous standalone application (12), this module now integrates the consensus clustering methodology described above to determine the best rank of factorization in a given range. This saves the need of launching the *Standard NMF* analysis (and therefore, uploading the data matrix) several times, or running the *Sample Classification* process as a previous step.

Finally, the *Biclustering Analysis* module implements a two-way clustering method to identify gene–experiment relationships. bioNMF estimates biclusters using a method based on a modified variant of the NMF algorithm, which produces a suitable decomposition as a product of three matrices that are constrained to have nonnegative elements. This variant, denoted as *Nonsmooth Nonnegative Matrix Factorization* (nsNMF) (14), produces a sparse representation of the gene-expression data matrix, making possible the simultaneous clustering of genes and conditions that are highly related in subportions of the data (3). This module also incorporates the consensus clustering methodology to determine the best rank of factorization.

## SOFTWARE USAGE

bioNMF has been designed as a web-based tool that mimics its standalone predecessor application (12). In all analysis methods, the original matrix is decomposed in two new nonnegative matrices that encode the latent information embedded in the original input data. In addition, visualizations used in this application also help in the interpretation of the results.

The full process is carried out in three very simple steps:

### Data set selection

The input data is a single standard tab-delimited text file that contains the data matrix with, or without labels, for example, a gene-expression matrix. In addition, if an email address is provided, the user will be notified when the analysis is finished. This feature is very useful when submitting large data sets or analysis that might take long time to process.

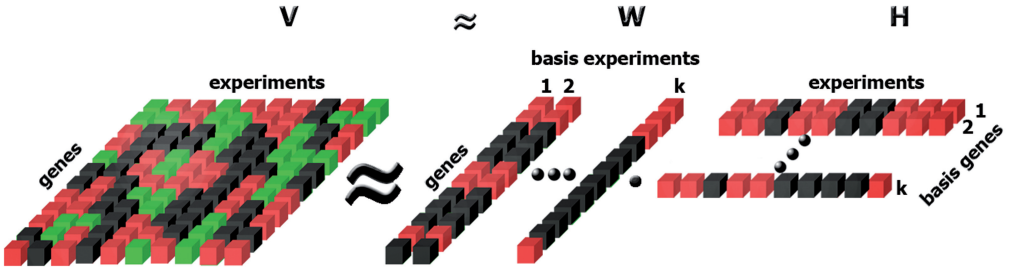


**Figure 1.** Schematic representation of the NMF model applied to gene-expression data. Input data matrix **V** is represented as a gene–experiment matrix and it is decomposed by the product of two new nonnegative matrices **W** and **H**. The $k$ columns of **W**, therefore, will have the dimension of a single array (genes) and are known as basis experiments or factors. The columns of **H** are known as encoding vectors and are in one-to-one correspondence with a single experiment of the gene-expression matrix. Consequently, each row of **H** has the dimension of a single gene and it is denoted as basis gene.

### Data preprocessing

Before the analysis, the data matrix can be transposed, normalized and/or transformed by several methods to satisfy the nonnegative constraints required by the NMF algorithm. Normalization methods include data centering, standardization of rows and columns (independently or simultaneously), mean subtraction by rows and columns and the normalization method proposed by Getz *et al.* (15) that first divides each column by its mean and then normalizes each row. On the other hand, transformation methods to make data positive include subtracting the absolute minimum, the exponential scaling and two data folding methods proposed by Kim and Tidor (13). These folding methods duplicate each row or column in which the first occurrence indicates positive expressions and the second indicates negative values.

### Data analysis

As described in the previous section, three different types of analysis are provided in *bioNMF* to cover some of the most important applications of this methodology: (i) *Sample Classification*; (ii) *Standard NMF* and (iii) *Biclustering Analysis*.

The sample classification module implements the methodology described by Brunet *et al.* (2). This methodology uses NMF and a model selection algorithm to determine the most suitable number of sample clusters in a given data set. This sample classification model is based on a reduced set of metagenes, and it has been proved to provide a more accurate and robust classification with respect to the classification based on the high-dimensional gene space. Results will be an estimation of the best number of clusters in the data set and the cluster assignments of each experimental condition. According to (2) the proper factorization, rank should be selected where the magnitude of the cophenetic correlation coefficient begins to fall. A graphical representation of the cophenetic correlation coefficient and the ordered consensus matrix as described in ref. (2) are also provided. Figure 2 provides a snapshot of the results of this step when applied to the acute myelogenous leukemia (AML) and acute lymphoblastic leukemia (ALL) data set (17).

The ordered consensus matrix, in conjunction with the cophenetic correlation coefficient provided in this step,

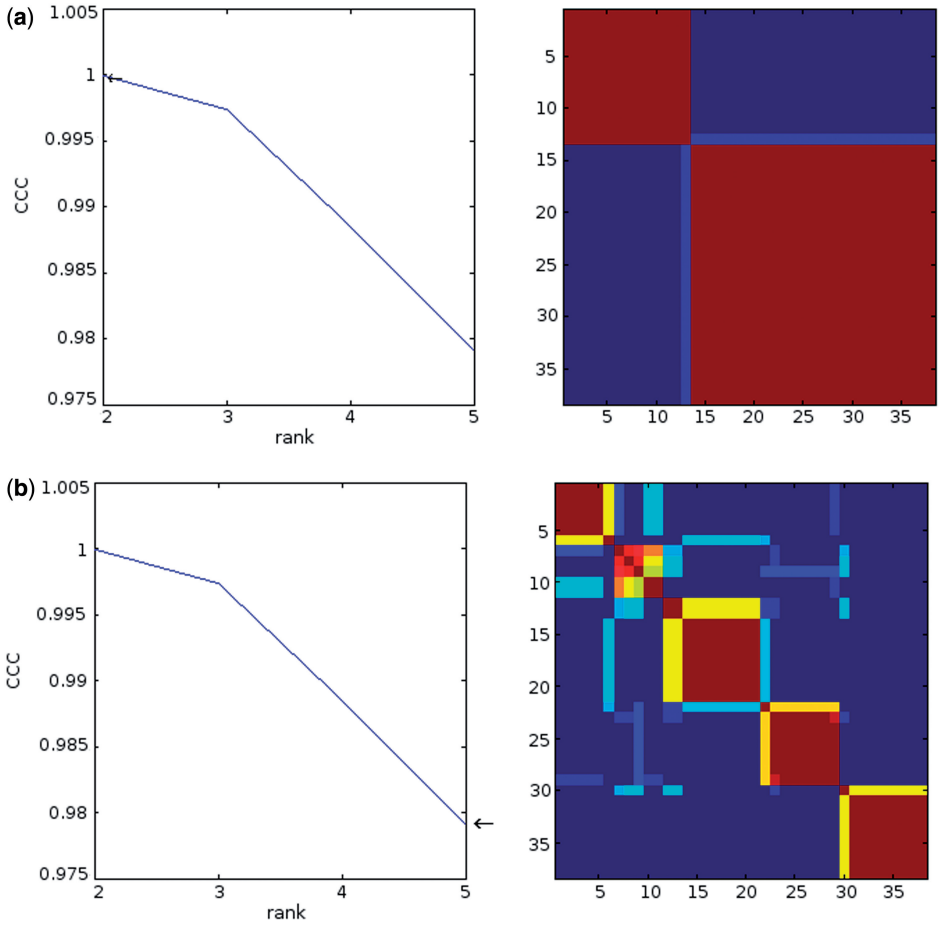


**Figure 2.** Snapshot of the output of the Sample Classification module. Results show the cophenetic correlation coefficient (left) for different values of $k$ and the reordered consensus matrices (right) calculated for the AML–ALL data set. (**A**) The consensus matrix pattern for $k = 2$ indicates a stable classification into two samples (most of the values are either 0 or 1 represented in red and blue colors in the picture). This is the expected clustering pattern in this two-class data set. (**B**) Consensus matrix for $k = 5$ showing a scattered pattern that indicates a more unstable classification in five classes.

gives an appropriate idea of the stability of the factorization for a given $k$. Since the NMF algorithm is nondeterministic, its solutions might vary from run to run when executed with different random initial values for **W** and **H**. The rationale behind the model selection approach proposed in ref. (2) is based on the fact that if the factorization is stable for a given value of $k$, we would expect that column's assignments to those $k$ factors would vary little from run to run.

For each run, the column assignment is defined by a connectivity matrix **C** of size $n \times n$ (where $n$ is the number of columns). Each entry $Cij$ in this matrix equals 1, if column $i$ and $j$ have their maximum for the same factor, and $Cij = 0$ if they do not. Consensus matrix is then defined as the average connectivity matrix over many factorization runs with different initial random conditions. The entries range from 0 to 1 and reflect the level of reproducibility of the columns' assignments. If a factorization is stable **C** will tend not to vary among runs, and thus entries will be close to 0 or 1. Dispersion between 0 and 1 will indicate a lack of reproducibility of the columns' assignments along the different runs. **C** matrix is then reordered to reflect the column's similarity. The more scattered the reordered matrix is, the less stable solution it reflects, since it will indicate that columns that were assigned to the same factor in one run, are probably assigned to different factors in another.

The standard NMF module performs the classical NMF factorization to the input data matrix. The tool returns the **W** and **H** matrices resulting in the factorization. It is also possible to run NMF different times using random initial conditions each time. Results can be independently saved or combined for further analysis as described in ref. (6). NMF is nondeterministic and therefore it may or may not converge to the same solution on each run depending on the random initial conditions. Therefore, executing the algorithm several times with different random initializations is a good approach for selecting the **W** and **H** that best approximates the input matrix **V**. Depending on the problem, less or more runs will be necessary to achieve an optimum solution. However, considering that the computational cost of this algorithm is very high a limited number of runs is recommended. On our own experience a value of 100 runs is normally enough to achieve reasonable results (3). This flexibility makes this unit a general instrument for any potential application in life sciences.

The biclustering algorithm module implements the methodology described in ref. (3). It is intended mainly for gene-expression analysis, although its applications can be extended to other type of data. Taking gene expression as a case study, this analysis group genes and samples based on local features generating sets of samples and genes that are locally related. Results are a set of biclusters (submatrices) encoding modular patterns. Each bicluster matrix contains the set of genes that are highly associated to a local pattern and samples sorted by its importance in this pattern. An image of the heatmap of each bicluster is generated. As an example, Figure 3 depicts a bicluster obtained from a data set containing the expression profiles of 46 soft-tissue tumor samples reported in (16).

## WEB SERVICES

In addition to the online access, bioNMF provides the same functionality included in the website as web services. Web services are a public programmatic API that enables users with more computer expertise to launch jobs into bioNMF server from their own programs, scripts and workflows.

The web services provided in bioNMF are built on open standards such as SOAP (*Simple Object Access Protocol*, a messaging protocol for transporting information; see http://www.w3.org/TR/soap/) and WSDL (*Web Services Description Language*, an XML format for describing web service capabilities and provided methods; see http://www.w3.org/TR/wsdl). The WDSL file describing bioNMF methods can be accessed at http://bionmf.dacya.ucm.es/WebService/BioNMFWS.wsdl.

The system allows the upload of matrices, and performs any of the three analyses described in the previous section. The web service works in a nonblocking way. The user launches the analysis and gets a job identifier as results. By using this job identifier, it is possible to poll the status

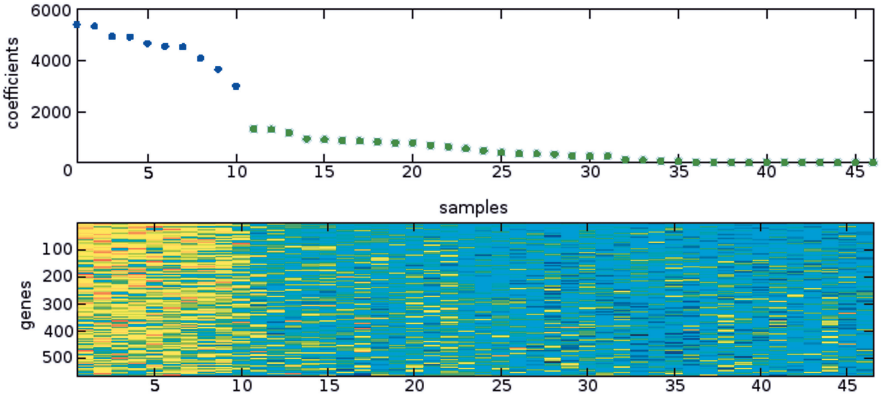


**Figure 3.** A Heatmap showing the subset of genes and samples in the bicluster. All samples are shown sorted by it association to the bicluster (local pattern). The plot on the upper part of the image represents the coefficients of all samples in the corresponding row of **H**. In blue are the samples that show the largest coefficient for that factor while in green are those samples associated to others.

of the job. Once the job is finished, the results can be retrieved using another web service function. The results for each operation include the essential information; however, all the analysis methods produce a set of files that provide other information, such as visualization images.

Information about all supported web services can be accessed from the web page at: http://bionmf.dacya.ucm.es/webservices.html. Full examples of a client program, implemented in Perl and Ruby are also provided.

## IMPLEMENTATION

This online tool has been designed to process big data sets in a multiprocessor environment. A batch-queuing manager controls most systems of this kind. In order to make this tool easy to use, bioNMF consists of two main components: the user interface (web or web services), that handles the batch-queuing system, and the analysis software executed by that system in the multiprocessor environment.

The web interface is implemented in PHP (*PHP: Hypertext Preprocessor*; see http://php.net/), a programming language for generating dynamic web pages. When an analysis is started, the web server submits a job to the batch-queuing system and shows a status page. This page is refreshed periodically until an independent process produces the results page when the job is finished. This system of two processes allows the user to close the browser at the status page stage without losing the results. An email with an URL to the results is optionally sent to the user when the process is finalized.

The analysis software is currently implemented in two layers. The external layer makes use of Matlab software (www.mathworks.com) to apply the preprocessing methods to the input data and to generate the graphical visualizations. On the other hand, the core of the system, explained below, is implemented in C language.

### NMF-parallelization

NMF is a very computing intensive technique. With this web application, we also provide an extremely efficient implementation of the NMF algorithm. All of the methods implemented in bioNMF use *parallel computing*. This technique is based on the principle that large problems can be divided into smaller ones, which may be solved in parallel (i.e. simultaneously) in multiple processors. This permits taking advantage of multicore CPUs and computing clusters environments.

The parallelization is focused on the most demanding part of the *bioNMF* analysis methods: the NMF algorithm. As it is based on matrix–matrix operations, our approach divides each matrix into a set of sub-matrices. Operations between these sub-matrices are computed in parallel. This is the case of **W** and **H** matrices, which are broken into smaller pieces and distributed among the processors. Each sub-matrix operation is then performed simultaneously. When necessary, the results are gathered in order to synchronize the updated matrices. This is done using MPI (*Message Passing Interface*; see http://www.mpi-forum.org/) that provides a low overhead communication mechanism.

This implementation represents a very cost-effective alternative to improve the throughput of this web-based application where simultaneous user's requests are going to be handled. Currently, a dedicated eight-node cluster is supporting this application.

## APPLICATION PERFORMANCE

As an example of performance, as well as of validation, a test of *Sample Classification* method was made by comparing the Matlab algorithm reported by Brunet *et al.* (2), available at www.broad.mit.edu/cancer/pub/nmf/with the online bioNMF analysis module. Both algorithms were tested with the AML and ALL data set (17), which is a 5000-gene by 38-sample data matrix. bioNMF's results are very close to those obtained in. Brunet *et al.*'s (2) algorithm took 2102 s (about 35 min) to complete the analysis in a single AMD Opteron processor, while bioNMF finished in 310 s (5 min, 10 s) in an Opteron eight-processor system. This effective implementation using parallel computing represents a suitable approach to reduce the long computing times required by bigger data sets. Better speedups can also be obtained if computer clusters with a larger number of nodes are used. The current implementation allows this upgrade in a transparent manner.

## DISCUSSION AND CONCLUSIONS

In the era of *–omics* technologies, the use of sophisticated statistical and data mining methods has become an essential task in many molecular biology laboratories. Web-based tools offer the opportunity of use complex data analysis methods in a friendly environment that quickly bring to many potential users new methodological developments. These types of applications are helping researchers in the exploration and analysis of large volume of data.

In this work, we present a web-based implementation of the NMF algorithm. This technique is a matrix factorization method that is increasingly used in many fields, such as image analysis, proteomics, metabolomics or genomics. This tool provides an efficient implementation of the NMF and different analysis pipelines based on this algorithm: standard NMF, biclustering analysis and sample classification. The architecture we used to implement this tool also allows the insertion of new variants of NMF or related methods in a straightforward manner. This is particularly important since many new factorization approaches are developing (18). A good example of this is the projected gradient NMF algorithms like ALS and HALS (19,20) which outperforms in many aspects the standard NMF models (see NMFLAB toolbox at http://www.bsp.brain.riken.jp/ICALAB/nmflab.html).

The design and implementation of bioNMF permit nonexpert users in exploring their data with NMF algorithm in an easy and transparent manner or even insert this analysis in their workflows using the provided

web services. Therefore, it is our hope that bioNMF will become an important tool to assist life-sciences researches in the exploratory data analysis cycle.

## ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish grants BIO2007-67150-C03-02, S-Gen-0166/2006, CYTED-505PI0058, CSD00C-07-20811 and TIN2005-5619. E.M.R. is supported by the grant *FPU* from the Spanish Ministry of Education. A.P.M. acknowledges the support of the Spanish *Ramón y Cajal* program. Funding to pay the Open Access publication charges for this article was provided by Spanish Grant. BIO2007-67150-C03-02.

*Conflict of interest statement*. None declared.

## REFERENCES

1. Lee,D.D. and Seung,H.S. (1999) Learning the parts of objects by non-negative matrix factorization. *Nature*, **401**, 788–791.
2. Brunet,J.P., Tamayo,P., Golub,T.R. and Mesirov,J.P. (2004) Metagenes and molecular pattern discovery using matrix factorization. *Proc. Natl Acad. Sci. USA*, **101**, 4164–4169.
3. Carmona-Saez,P., Pascual-Marqui,R.D., Tirado,F., Carazo,J.M. and Pascual-Montano,A. (2006) Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinform.*, **7**, 78.
4. Tamayo,P., Scanfeld,D., Ebert,B.L., Gillette,M.A., Roberts,C.W. and Mesirov,J.P. (2007) Metagene projection for cross-platform, cross-species characterization of global transcriptional states. *Proc. Natl Acad. Sci. USA*, **104**, 5959–5964.
5. Inamura,K., Fujiwara,T., Hoshida,Y., Isagawa,T., Jones,M.H., Virtanen,C., Shimane,M., Satoh,Y., Okumura,S., Nakagawa,K. *et al*. (2005) Two subclasses of lung squamous cell carcinoma with different gene expression profiles and prognosis identified by hierarchical clustering and non-negative matrix factorization. *Oncogene*, **24**, 7105–7113.
6. Chagoyen,M., Carmona-Saez,P., Shatkay,H., Carazo,J.M. and Pascual-Montano,A. (2006) Discovering semantic features in the literature: a foundation for building functional associations. *BMC Bioinform.*, **7**, 41.
7. Pehkonen,P., Wong,G. and Toronen,P. (2005) Theme discovery from gene lists for identification and viewing of multiple functional groups. *BMC Bioinform.*, **6**, 162.
8. Dueck,D., Morris,Q.D. and Frey,B.J. (2005) Multi-way clustering of microarray data using probabilistic sparse matrix factorization. *Bioinformatics*, **21 (Suppl. 1)**, i144–i151.
9. Venter,J.C., Adams,M.D., Myers,E.W., Li,P.W., Mural,R.J., Sutton,G.G., Smith,H.O., Yandell,M., Evans,C.A., Holt,R.A. *et al*. (2001) The sequence of the human genome. *Science*, **291**, 1304–1351.
10. Heger,A. and Holm,L. (2003) Sensitive pattern discovery with 'fuzzy' alignments of distantly related proteins. *Bioinformatics*, **19 (Suppl. 1)**, i130–i137.
11. Lohmann,G., Volz,K.G. and Ullsperger,M. (2007) Using non-negative matrix factorization for single-trial analysis of fMRI data. *Neuroimage*, **37**, 1148–1160.
12. Pascual-Montano,A., Carmona-Saez,P., Chagoyen,M., Tirado,F., Carazo,J.M. and Pascual-Marqui,R.D. (2006) bioNMF: a versatile tool for non-negative matrix factorization in biology. *BMC Bioinform.*, **7**, 366.
13. Kim,P.M. and Tidor,B. (2003) Subsystem identification through dimensionality reduction of large-scale gene expression data. *Genome Res.*, **13**, 1706–1718.
14. Pascual-Montano,A., Carazo,J.M., Kochi,K., Lehmann,D. and Pascual-Marqui,R.D. (2006) Nonsmooth nonnegative matrix factorization (nsNMF). *IEEE Trans. Pattern Anal. Mach. Intell.*, **28**, 403–415.
15. Getz,G., Levine,E. and Domany,E. (2000) Coupled two-way clustering analysis of gene microarray data. *Proc. Natl Acad. Sci. USA*, **97**, 12079–12084.
16. Nielsen,T.O., West,R.B., Linn,S.C., Alter,O., Knowling,M.A., O'Connell,J.X., Zhu,S., Fero,M., Sherlock,G., Pollack,J.R. *et al*. (2002) Molecular characterisation of soft tissue tumours: a gene expression study. *Lancet*, **359**, 1301–1307.
17. Golub,T.R., Slonim,D.K., Tamayo,P., Huard,C., Gaasenbeek,M., Mesirov,J.P., Coller,H., Loh,M.L., Downing,J.R., Caligiuri,M.A. *et al*. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.
18. Cichocki,A., Zdunek,R. and Amari,S. (2008) Nonnegative matrix and tensor factorization. *IEEE Signal Processing Magazine*, **25**, 142–145.
19. Cichocki,A. and Zdunek,R. (2007) Regularized alternating least squares algorithms for non-negative matrix/tensor factorizations. *Lect. Notes Comput. Sci.*, **4493**, 793–802.
20. Cichocki,A., Zdunek,R. and Amari,S. (2007) Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. *Lect. Notes Comput. Sci.*, **4666**, 169–176.

## 4.4 MARQ: an online tool to mine GEO

**Reference:** **Vazquez, M.**, Nogales-Cadenas, R., Arroyo, J., Botías, P., García, R., Carazo, J., Tirado, F., Pascual-Montano, A. & Carmona-Saez, P. (2010), 'MARQ: an online tool to mine GEO for experiments with similar or opposite gene expression signatures', *Nucleic Acids Research* (Web Server issue). To appear in volume 38. (JCR 2008 impact factor: **6.878**).

**Summary:** Gene expression data repositories, such as the Gene Expression Omnibus (GEO), collect vasts amounts of experimental data covering a large number of experimental conditions. In spite of their great potential for new discovery, these data have been sparsely exploited. A lack of meta-analysis tools may account for that, although in recent years several have emerged. These new tools try to integrate published data basically by implementing a manner of information retrieval scheme, where a query returns a list of relevant signatures (summaries of gene expression patterns) derived from these experiment repositories. The returned signatures are deemed relevant when they show similar or opposite patters of gene expression to the query.

After reviewing the available tools, we have implemented some of the most successful approaches in this work, in particular, the use of rank based statistics. Ranks have the benefit of being less influenced by heterogeneity between different experiments than the original expression values; yet still manage to retain more information than more naïve approaches that just look at overlaps between lists of differentially expressed genes.

The application has been enhanced with several exploration and meta-analysis features. Experimental use cases illustrate the great level of insight offered by the results. In our opinion, the generality of the application and the insight provided supersede those of applications published to this date.

The applications are offered through SOAP web services and online interfaces, and the source code has been included in the Rbbt project.

# MARQ: an online tool to mine GEO for experiments with similar or opposite gene expression signatures

Miguel Vazquez[1] (miguel.vazquez@fdi.ucm.es), Ruben Nogales-Cadenas[2] (ruben.nogales@fis.ucm.es), Javier Arroyo[3] (jarroyo@farm.ucm.es), Pedro Botías[4] (pbotias@bio.ucm.es), Raul García[3] (rgarcias@farm.ucm.es), Jose M. Carazo[5] (carazo@cnb.csic.es), Francisco Tirado[2] (ptirado@dacya.ucm.es), Alberto Pascual-Montano[5] (pascual@cnb.csic.es) and Pedro Carmona-Saez[2,*] (pcarmona05@gmail.com)

[1]Software Engineering Department, Facultad de Informatica, Universidad Complutense de Madrid. Spain, [2]Computer Architecture Department, Facultad de Fisicas, Universidad Complutense de Madrid. Spain, [3]Microbiology Department II, Facultad de Farmacia, Universidad Complutense de Madrid. Spain, [4]Genomics Unit, Parque Científico de Madrid-Universidad Complutense de Madrid. Spain, [5]National Center for Biotechnology. CNB-CSIC. Madrid. Spain

[*]To whom correspondence should be addressed. Email addresses: pcarmona05@gmail.com

MARQ: http://marq.dacya.ucm.es/

Keywords: DNA microarrays, gene expression, meta-analysis, gene signatures

## ABSTRACT

The enormous amount of data available in public gene expression repositories such as Gene Expression Omnibus (GEO) offers an inestimable resource to explore gene expression programs across several organisms and conditions. This information can be used to discover experiments that induce similar or opposite gene expression patterns to a given query, which in turn may lead to the discovery of new relationships among diseases, drugs or pathways, as well as the generation of new hypotheses.

In this work we present MARQ, a web-based application that allows researchers to compare a gene signature provided as a query in the form of a set of over- and under-expressed genes against a signature database built from GEO datasets for different organisms and platforms. MARQ offers an easy-to-use and integrated environment to mine GEO in order to identify conditions that induce similar or opposite gene expression patterns to a given experimental condition. MARQ also includes additional functionalities for the exploration of the results, including a meta-analysis pipeline to find genes that are differentially expressed across different experiments. The application is freely available at http://marq.dacya.ucm.es/.

## INTRODUCTION

DNA microarrays have become an extensively used technique to analyze global gene expression profiles. Its widespread usage quickly promoted the creation of gene expression data repositories such as the NCBI Gene Expression Omnibus (GEO) (1), which currently holds more than 10000 experiments for over 500 organisms. Nevertheless, in spite of this huge amount of information, in most cases researchers still focus their analysis on individual experiments, thus not making use of such valuable resources. In this way, gene expression analyses are usually performed from a gene

centered perspective with the goal of identifying relevant sets of genes, e.g. genes differentially expressed between different conditions; which are then used to understand the biological mechanisms relevant in that experimental setup. However, besides this gene-to-phenotype approach, gene signatures can also be used to establish connections among different physiological conditions (2). For example, we can discover connections among diseases, drugs or pathways by similarities in their gene expression signatures; conditions inducing similar signatures might be modulating the same pathways, while conditions with opposite signatures might be involved in reverting the original phenotype.

During the last few years, some previous works have shown the usefulness of mining large gene expression libraries to find similarities in gene expression signatures. Hughes et al. (3) showed that a compendium of gene expression data for diverse mutations and chemical treatments in yeast could be used for the functional annotation of small molecules and genes. Rhodes et al. developed Oncomine (4), which integrates gene expression signatures from manually selected cancer microarray experiments from several sources. More recently, Lamb et al. introduced the Connectivity Map (2), a searchable database that allows users to compare a query signature against a database populated with global signatures derived from human cell lines treated with different compounds. The Connectivity Map compares signatures using a statistic similar to the one in the Gene Set Enrichment Analysis (5), which is based on the ranks of the genes in the complete signature, as opposed to other applications such as L2L (6) or more recently Exalt (7) which use threshold-based criteria. The rank statistic alleviates the high variability that comes from sub-setting the list of genes using a threshold. A more extensive comparison with related applications can be found in the additional material.

In this work we present MARQ (Microarray Rank Query), a web-based application that allows researchers to query a signature database derived from GEO to find experiments that induce similar or opposite gene expression patterns than a given experiment. The signature database has been compiled from all GEO Datasets for five model organisms (*Homo sapiens, Mus musculus, Rattus norvegicus, Saccharomyces cerevisiae* and *Arabidopsis thaliana*). MARQ uses as input the standard output of most microarray studies, that is, a set of over- and/or a set under-expressed genes, and uses a rank based statistic to compare it with signatures in the database. MARQ also includes several functionalities that can be use for further exploration of results and to infer potential relationships among gene signatures in the database. The application is freely accessible at http://marq.dacya.ucm.es. The web portal includes tutorials of use and a detailed description of the web service interface and methods.

## FEATURES AND FUNCTIONALITIES

Figure 1 shows a general overview of the system. It has been organized in three main components: data input, data analysis and data visualization and exploration.

The input of the system consists of a query signature composed by a set of over-expressed and /or a set of under-expressed genes. This query is compared against all signatures in the database using a rank-based statistics, and the output lists all signatures sorted by their similarity to the query. In this way, signatures with the highest positive scores represent experimental conditions that show similar over- and under-expression patterns to the query, while signatures with the highest negative scores are those that reverse the expression pattern of the query. The application also provides several features to explore and assess the significance of the results. Each signature is linked to a set of terms such as GO terms and words derived from its GEO description. These

terms can be used to find datasets associated with the same terms and potentially sharing some biological characteristic, and to perform an enrichment analysis in order to determine which of these concepts are over-represented in the most significant signatures. These concepts may provide clues to understand the molecular processes that are relevant in the query experiment.
Finally, MARQ also offers a functionality to perform a rank based differential expression analysis to detect genes that are commonly over or under-expressed across different signatures in the database.

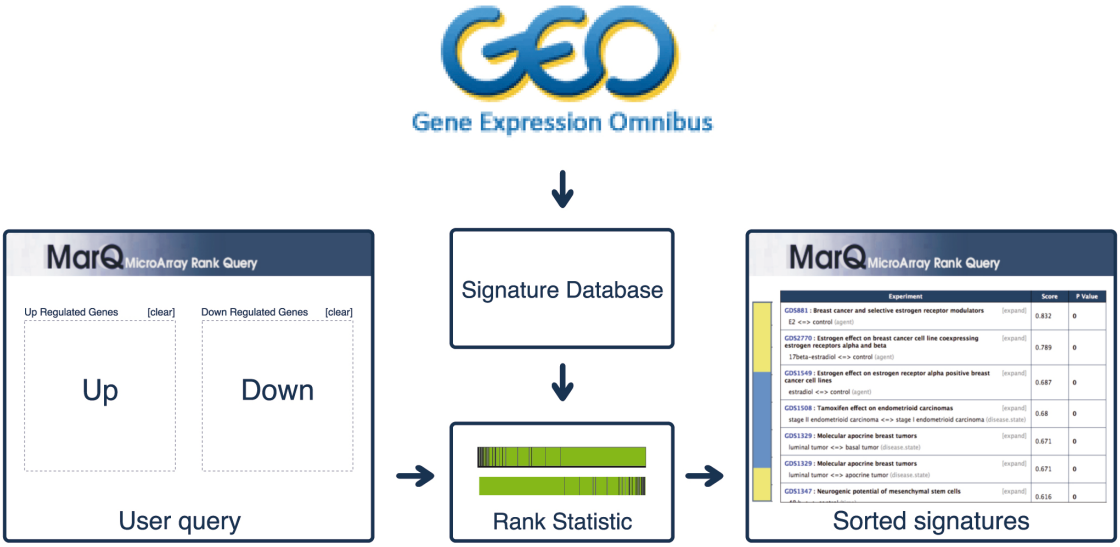


Figure 1: MARQ is composed by three layers, the data input form (on the left hand side) in which the user enter the query signature, the data analysis component that compares it against the database, and data visualization and exploration interface (on the right hand side of the figure)

**Signature Database Construction**

To build the gene signature database we processed GEO Datasets for five different organisms: *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Saccharomyces cerevisiae* and *Arabidopsis thaliana*. The current version of MARQ contains 11460 gene expression signatures from a total of 2050 GEO datasets and series from 355 platforms, including 708 datasets for human, 750 for mouse, 302 for yeast, 213 for rat and 77 for A. thaliana.
We retrieved from GEO all datasets associated to each organism using the NCBI E-utils and the GEOQuery package (8), Soft formatted gene expression data was loaded into the R environment, where the limma package was used to perform differential expression analysis among every pair of classes for each experimental factor. The gene signatures were then generated by sorting all genes in the array by their t-values, or fold-changes if the dataset did not have enough sample replicates to apply the statistical test. The direction of the comparison (A Vs B or B Vs A) was arbitrarily set, unless any of the classes were identified as a reference condition. Additionally, for two channel platforms we also performed one-class differential expression analysis, as each sample already represents a gene expression ratio.
For cross-platform comparisons we used a common identifier to which all platform Ids were converted. Probes that could not be translated were removed from the analysis, and expression

profiles were averaged for probes associated to the same gene. Using this framework we generated a new set of gene-centered signatures.

**Signature Comparison**

For a given query, we compute a similarity score for each signature in the database adapting the methodology reported in the Connectivity Map (2). This score reflects whether the up-regulated set of genes in the query signature tend to appear on the top of the database signature while down-regulated genes tend to appear on the bottom (positive score), or vice versa (negative score). Note that query signatures are composed of unordered lists of up and down regulated genes, while database signatures are composed of the complete list of probes in the platform sorted by differential expression. A high positive score indicates that the signature in the database has a similar pattern of expression and repression than the query. Alternatively, a high negative score indicates that the signature has an antagonistic pattern. The score is calculated for the up and down regulated genes using a weighted Kolmogorov–Smirnov like statistic.

The significance of each score is determined using random permutations: The p-value associated with the score is the fraction of a large number of random lists of genes of the same length as the query that produce a larger score. Since the number of signatures compared against with in each query can be of thousands a proportion of them could be false positives, so the MARQ web interface reports the p-values after making an FDR correction, as described in (9). The randomization scheme used has been reported to by slightly flawed (10), and over-estimate significance, but has been used successfully in similar settings, such as The Connectivity Map application (2). A complete description of the statistical tests and score computation are available at the application web site.

**Connecting datasets through common biomedical concepts**

In MARQ, each database signature is annotated with meta-information that allows us to connect them by common biological concepts, and to explore their relationships. These terms can be used for two purposes: to interactively find other signatures associated to the same biological concepts and explore their distribution in the sorted list, or to find which terms are over-represented among the most significant signatures.

Signatures are annotated with terms from different sources: terms contained in the description of experiments in GEO, and GO terms enriched in over-expressed and under-expressed genes in each signature. An enrichment score for each term associated to the significant signatures is calculated using the same rank based statistical test as described above. The result of this analysis is a list of biological concepts with their corresponding p-values. Those terms with statistically significant p-values are representative of the most related signatures, and can provide information about the underlying biological processes, or highlight potential connections among different experimental conditions, for example two different treatments that are activating the same pathway.

**Comparing signatures to find common differentially expressed genes**

One of the most powerful features of MARQ to explore the results is a meta-analysis tool that can be used to find genes that are commonly deregulated across different database signatures. This can analysis can be used, for example, to find out which genes that are over-expressed in a

disease signature are also under-expressed after a treatment signature. MARQ implements the Rank products method (11) to perform a differential expression analysis across user-selected gene signatures. The method is a non-parametric statistic that detects items that are consistently highly ranked in a number of lists, for example genes that are consistently found among the most strongly over-expressed and under-expressed in a number of replicates. It has the advantage that it overcomes the heterogeneity among multiple datasets and does not require normalization.
In MARQ, users can select a set of signatures and the application creates a rank matrix with common genes in the different signatures. This matrix is analyzed using Rank products to find genes that are over-expressed and under-expressed, considering the different signatures as biological replicates in the differential expression analysis. This approach can provide useful information about common biomarkers across different experimental conditions.

## MARQ APPLICATION

To run an analysis, users must input the query lists of genes and select the organism and the type of analysis (cross-platform or single platform). The gene id format depends on the type of analysis; for platform specific queries users must use platform specific probe ids (eg. Affymetrix probe ids), but for cross-platform queries, several id formats are supported, including Entrez, Ensemble, Gene symbols, most platform probe id formats, and most of the id formats listed in BioMart. Single platform analyses have a higher coverage on the number of probes but are limited to only signatures from the same platform, as opposed to cross-platform analyses, which allow users to query the all signatures in the database from an given organism. A single list can be also used (either considering it over- or under-expressed genes) which can be interesting in some situations, for example, to explore what GEO signatures modulate in a significant way genes from a KEGG pathway.

After a job is launched, the user is redirected to a job status page that automatically reloads reporting progress status, until the job is finished. Once the analysis is completed the results page is loaded, which includes a table where rows represent signatures, sorted for relevance to the query; signatures in the top are considered to be the most directly related to the query while signatures in the bottom are the most indirectly related. Signatures without enough replicates to perform the statistical test, and for which we sort genes by log-ratios instead of t-values, are marked with the '[ratio]' keyword in their name.

The textual description of the signatures is linked to the *Word* meta-information annotations described in a previous section, so clicking on a term highlights all signatures containing that word in their description. The *Annotations Menu* can be used to explore other types of meta-information annotations such as GO terms.

The results page also includes a hit bar for each signature that shows the position of the query genes in that signature. The gene distribution can be carefully examined in the hit exploration page linked to each hit-bar that includes exact positions in the signature, gene names and links to the corresponding GEO profiles, if available.

Finally, users can select a list of signatures to include in the meta-analysis pipeline and the signature comparison page shows the genes that are commonly over-expressed and the genes that are commonly under-expressed across all the selected signatures.

Beyond the web tool, the core functionalities of MARQ are offered via a SOAP web service, which is used by the web portal application as the computational back-end. Using this

technology, researchers can access to MARQ functionalities programmatically to insert them in their data mining pipelines or in other bioinformatics systems in a very straightforward manner. WSDL file and services description is available through the web portal help page. While the off-line processing involves using R, the on-line front end is coded entirely in Ruby, with some portions in in-lined C for efficiency. The complete source code for the applications back-end as well as for the web front end is available at http://github.com/mikisvaz, with instructions to configure and deploy a custom installation.

## USE CASE: EXPLORING CELL WALL STRESS RESPONSES IN YEAST

To exemplify the potentials of the application, we have used MARQ to analyze the transcriptional program of the budding yeast *Saccharomyces cerevisiae* to cell wall stress caused by zymolyase (12) which mainly affects the β-1,3glucan cell wall network. Yeast cells respond to cell wall damage by activating a transcriptional program that includes the up-regulation of genes mainly involved in cell wall remodeling, stress, metabolism, and signaling (13). In addition MAPK kinase signaling pathways play important roles in the regulation of the transcriptional responses necessary for adaptation of cells.

Analysis of the gene expression profile of a wild-type strain treated with zymolyase revealed that signatures derived from conditions such as exposure to Congo Red, long term exposure to DTT, hyperactivation of the cell wall integrity (CWI) pathway (GAL-PKC1-R398A), tunicamycin and hyper-osmotic conditions showed high positive scores with the zymolyase treatment. This is in agreement with the fact that all of these conditions affect cell wall integrity by different mechanisms. On the other hand, experiments related to hypo-osmotic conditions showed high negative scores.

The "compare signatures" tool allowed us the identification of genes commonly over-expressed among the cell wall stress conditions mentioned above. Signatures related to zymolyase exposure, Dithiothrietol exposure and over-expression of elements of MAPK pathways (14) were used determine common over-expressed genes. Using this approach we were able to identify the common cell wall compensatory "signature" developed by the yeast in response to cell wall stress. Common genes include encoding cell wall-remodeling enzymes like *CRH1* or *BGL2,* and other cell wall proteins such as *YLR1194C, SED1, CWP1, PST1, PIR3* or *YPS3,* which is in agreement with the fact that cell wall damage needs to be compensated by cell wall remodeling processes and cell wall-remodeling enzymes are the main components included in the common cell wall stress response (13) . In agreement with an increase in the chitin content, as part of this compensatory response, genes like *GFA1*, encoding for a protein involved in the biosynthesis of chitin, are included in the common over-expressed signature. Also in agreement with our previous findings (13), genes related to metabolism, stress and signaling were also identified. The last group is particularly interesting because two of the genes included in this group include *SLT2* and *MLP1*, both of them being signaling components of the CWI pathway, the main pathway involved in the regulation of the transcriptional responses to cell wall stress.

Additional MARQ analysis of the yeast transcriptional response to zymolyase in a *hog1* mutant allowed us to identify a crosstalk between the mating and HOG pathways. This result clearly illustrates the potential of this tool to identify novel connections between different signaling pathways. Full details of all these analysis are provided in the application web page. Another use case, for the human melanoma, has been included in the additional material.

## CONCLUSIONS

We present MARQ, a web based application that enables users to query signatures derived from GEO, and retrieve experimental conditions that may induce similar or opposite gene expression programs to a given query. The system is provided in an easy-to-use and integrated environment that offers additional functionalities such as the possibility of performing gene expression meta-analysis to find common over- and under-expressed genes across different conditions, or connect signatures through common words or GO. We hope the application will prove useful to the research community.

## FUNDING

This work has been partially funded by the Spanish grants BIO2007-67150-C03-02, S-Gen-0166/2006, TIN2005-5619, and PS-010000-2008-1.

## BIBLIOGRAPHY

1. Barrett, T., Troup, D.B., Wilhite, S.E., Ledoux, P., Rudnev, D., Evangelista, C., Kim, I.F., Soboleva, A., Tomashevsky, M., Marshall, K.A. *et al.* (2009) NCBI GEO: archive for high-throughput functional genomic data. *Nucleic Acids Res*, **37**, D885-890.
2. Lamb, J., Crawford, E.D., Peck, D., Modell, J.W., Blat, I.C., Wrobel, M.J., Lerner, J., Brunet, J.P., Subramanian, A., Ross, K.N. *et al.* (2006) The Connectivity Map: using gene-expression signatures to connect small molecules, genes, and disease. *Science*, **313**, 1929-1935.
3. Hughes, T.R., Marton, M.J., Jones, A.R., Roberts, C.J., Stoughton, R., Armour, C.D., Bennett, H.A., Coffey, E., Dai, H., He, Y.D. *et al.* (2000) Functional discovery via a compendium of expression profiles. *Cell*, **102**, 109-126.
4. Rhodes, D.R., Kalyana-Sundaram, S., Mahavisno, V., Varambally, R., Yu, J., Briggs, B.B., Barrette, T.R., Anstet, M.J., Kincead-Beal, C., Kulkarni, P. *et al.* (2007) Oncomine 3.0: genes, pathways, and networks in a collection of 18,000 cancer gene expression profiles. *Neoplasia*, **9**, 166-180.
5. Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S. *et al.* (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A*, **102**, 15545-15550.
6. Newman, J.C. and Weiner, A.M. (2005) L2L: a simple tool for discovering the hidden significance in microarray expression data. *Genome Biol*, **6**, R81.
7. Yi, Y., Li, C., Miller, C. and George, A.L., Jr. (2007) Strategy for encoding and comparison of gene expression signatures. *Genome Biol*, **8**, R133.
8. Sean, D. and Meltzer, P.S. (2007) GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. *Bioinformatics*, **23**, 1846.
9. Benjamini, Y. and Hochberg, Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 289-300.
10. Efron, B. and Tibshirani, R. (2007) On testing the significance of sets of genes. *Annals of Applied Statistics*, **1**, 107-129.
11. Breitling, R., Armengaud, P., Amtmann, A. and Herzyk, P. (2004) Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS letters*, **573**, 83-92.

12. Garcia, R., Rodriguez-Pena, J.M., Bermejo, C., Nombela, C. and Arroyo, J. (2009) The high osmotic response and cell wall integrity pathways cooperate to regulate transcriptional responses to zymolyase-induced cell wall stress in Saccharomyces cerevisiae. *J Biol Chem*, **284**, 10901-10911.
13. Arroyo, J., Bermejo, C., Garcia, R. and Rodriguez-Pena, J. (2009) Genomics in the detection of damage in microbial systems: cell wall stress in yeast. *Clinical Microbiology and Infection*, **15**, 44-46.
14. Roberts, C.J., Nelson, B., Marton, M.J., Stoughton, R., Meyer, M.R., Bennett, H.A., He, Y.D., Dai, H., Walker, W.L., Hughes, T.R. *et al.* (2000) Signaling and circuitry of multiple MAPK pathways revealed by a matrix of global gene expression profiles. *Science*, **287**, 873-880.

## 4.5 Named Entity Recognition and Normalization: A Domain-Specific Language Approach

**Reference:** **Vazquez, M.**, Chagoyen, M. & Pascual-Montano, A. D. (2008), Named entity recognition and normalization: A domain-specific language approach, *in* J. M. Corchado, J. F. de Paz, M. Rocha & F. F. Riverola, eds, 'IWPACBB', Vol. 49 of *Advances in Soft Computing*, Springer, pp. 147–155.

**Summary:** Gene mention recognition and normalization constitute important steps for many literature mining tools. Gene mention recognition refers to the task of finding portions of text that could make reference to a gene or gene product. Normalization consists of determining which particular gene is mentioned in each case. Both gene mention recognition and gene mention normalization have been the subject of several tasks in text mining competitions like BioCreative. Several applications are available for gene mention recognition; this is not the case for applications implementing a normalization phase.

This system implements both the recognition and normalization phases, achieving acceptable results in the BioCreative competition data while still retaining its generality. The system also achieves a great level of flexibility and reconfigurability owing to the use of state of the art meta-programming practices and domain specific languages.

The system is part of the base Rbbt framework, along with other less sophisticated named entity recognition strategies for simpler kinds of entities.

# Named Entity Recognition and Normalization. A Domain-Specific Language Approach

Miguel Vazquez[1], Monica Chagoyen[2,3], and Alberto Pascual-Montano[2]

[1] Departamento de Ingeniería del Software e Inteligencia Artificial.
Corresponding author: miguel.vazquez@fdi.ucm.es
[2] Dpto. Arquitectura de Computadores, Universidad Complutense de Madrid, Madrid, Spain.
[3] Biocomputing Unit, Centro Nacional de Biotecnología - CSIC, Madrid, Spain

**Summary.** We present a tool that performs Named Entity Recognition and Normalization of gene and protein mentions on biomedical text. Several freely available tools perform the recognition of named entities. None that we known of, however, performs the Normalization -matching these names with the actual entities they refer to. The tool we present not only offers a complete solution to the problem, but it does so by providing easily configurable framework, that abstracts the algorithmic details from the domain specific. Configuration and tuning for particular tasks is done using domain specific languages, clearer and more succinct, yet equally expressive that general purpose languages. An evaluation of the system is carried using the BioCreative datasets.

## 1 Introduction

Three factors have motivated the biomedical community to get interested in literature mining [9]. The first one is thanks to recent high throughput techniques, like DNA microarrays, that, by involving large collections of entities, place bigger demands in the researcher's use of previous knowledge. The second factor is the fast pace at which scientific articles are being published, making it hard for researchers to keep up to date. Coupled with these is the trend followed by many editorials of making the articles available on-line, thus, making them readily available for analysis by software tools.

Named entity recognition (NER) is an important tool for literature mining as it is found as an initial step in many information extraction applications [10]. Named entity recognition has been used in domains other than the biomedical, like news wire text. It is in the biomedical domain, however, where it presents the most challenges. Gene and protein names are the most common subjects for NER; they have several characteristics that make them challenging. They are very numerous, millions, the number grows continuously, and the names used to refer to them are not necessarily standardized

[6]. Furthermore, in many cases the names are the same across different genes or match some common English words, producing a significant amount of ambiguity [1]. These problems not only affect finding the mentions in the text, but also the subsequent task of identifying the actual entity they refer to. This identification step is usually referred to as normalization, and is as much a challenging task as NER itself [2].

Several initiatives have dealt with the problem of NER in the biomedical context. We have studied closely the BioCreative competition [10, 2], which had several tasks relating to NER and normalization. We used some of the resources they provided to evaluate our system. As for the availability of tools for NER and normalization, we have found several freely available tools for the NER step alone. Some of them, like Abner and Banner [8, 5] we have studied in detail. We did not, however, find any available tool that implemented the normalization step. This has motivated us to develop our own.

Examining the state of the art in NER reveals a strong trend towards the use of conditional random fields (CRF). These probabilistic models have proven to be appropriate for text labeling [4], which is how NER is usually approached. For NER we use CRF to predict labels for a sequence of words, given a set of features associated with each word. Most of the work on NER using CRF differs basically in what these features are. In fact, the nature of the CRF algorithm reduces the problem to determining a suitable set of features. Most common features are orthographic: Has the word any uppercase letter? Any digits? Is there a slash character in the word? Etc.

We developed a system for both NER and normalization that implemented a standard approach, but one that offers a practical way to be extended and configured. For NER we implemented the same ideas from Abner and Banner, abstracted the core working and committed all the details on creating the features to a domain specific language (DSL). For the normalization task we developed our own solution, based on many ideas from the BioCreative competition, and also separated all the configuration options to domain specific languages.

## 2 System Overview

This is an overview of the process we have implemented. Following sections will cover each aspect in more detail, but this section should be useful to gain perspective for further discussion. The CRF model for NER is trained off-line. The training text is chunked into words, each word is labeled and turned into features. This labels and features are used to train the model. The normalization also needs preparation beforehand. A lexicon is read by the system (a file containing all the genes of interest along with their most common synonyms. Usually for one particular organism) and genes are processed one at a time. For each gene the system takes each of its synonyms, runs them through several index cue generators, each turning the synonym into a set of

variations. It then maps each cue to the given gene in the appropriate index, one for each cue generator. There are several indexes so that some could be more restrictive than others and, thus, preferred. The same cue might get mapped to several genes.

When new text arrives, the NER system chunks it into words, turns them into features and uses the model trained before to predict the labels. Using the labels, the system figures out the mentions and their boundaries. These mentions are then feed to the normalizer system.

The normalizer takes each of the mentions and begins a process of identifying the gene it refers to. The first thing it does is to turn the mention into cues, using the same index cue generators as before, and uses those cues to query the indexes in order. As soon as a match is made the matching stops, since the sooner the match is made the better the quality, if the indexes are sorted appropriately. The results are submitted to the next step. If no match is made the system drops this mention and moves along.

The index match might have render one gene or several, in any case, for each gene the system finds the synonyms in the lexicon and compares them to the mention. This comparison is done by breaking both the mention and the synonym into tokens, identifying the nature of this tokens and evaluating their similarities and differences. For each gene the best matching synonym is considered, and the value of the comparison servers both as a filter and as a ranking. A threshold is established so that genes not having similar enough synonyms are dropped.

If any genes pass the threshold, the best ranking one is selected. If there is a draw between several genes, the system disambiguates them. This is done by taking each of the genes, translating them to Entrez Gene codes, and taking the text from their description there. This descriptive text is then compared to the text from where the mention came from, and the gene with the most similar text is selected, or both, if the draw still holds.

## 3 Named Entity Recognition

The approach we follow for NER is based on that of Abner. We will describe briefly how it works, a more detailed description can be found in [4] or [8].

Finding mentions in text means determining the words that are likely to constitute a gene name. This is commonly done by labeling the words with their position relative to the mentions. The `IOB` schema that we use [7], uses the label `B` to mark the initial word on a mention, `I` marks any other word in the mention, and `O` is used for words that are outside of any mention. Named Entity Recognition boils down to determining what these labels would be for a given sequence of words.

The labeling of a sequence of words using Conditional Random Fields is done by representing each word by a set of features, and using a probabilistic model to determine the sequence of labels that most likely would have produce

a sequence of features like the one been analyzed. This model is built in an off-line phase using example data, in our case, the one provided as training for the BioCreative competition.

Our system uses CRF++ [3] to build the model and produce the labels, as it provides bindings for Ruby. Abner and Banner use Mallet, a Java implementation of CRF of similar characteristics. The main difference between all three systems strides in the features used to represent each word. Abner defines a set of features based on morphological features, these features are further expanded in Banner to include semantic features as well. Our system extracts all the feature generation from the rest of the system, and provides a DSL language to specify them. The features used to evaluate our system include those in Abner and Banner, excluding the syntactic based ones, as they slowed the system down considerably and they did not seem to enhance performance in our application significantly.

We will now take a look at the DSL used to define the features. Figure 1 shows the definition of three of the features that exemplify the three ways of defining features. The first one defines the feature `hasDigits` with a regular expression, so that if the word has a digit the feature is true. In the second case, `prefix_3` is defined with a regular expression capture, in the case the features value is what ever is captured in by the parenthesis, the first three characters. The last example uses a code block, the results of which, the word in lower case, is assigned to the feature. These three features, along with 25 others, are described in the default configuration file, and can be easily over written, and extended. Regular expressions are used as they are themselves a DSL for string matching, arguably the best way to capture the requisites of this particular domain.

```
hasDigits     /\d/
prefix_3      /^(...)/
downcase      do |w| w.downcase end
```

**Fig. 1.** Feature declaration

The CRF tool that we use, CRF++ [3], has a particular characteristic of being able to consider a certain context around each word at any particular step, as opposed to just the current state and word features. This context may be defined in another section of our DSL, as shown in figure 2, which just states that features `special, token2, isPunctuation and isDelim` from words surrounding the current one in distance of up three should be considered. The features used here must have been defined in the previous DSL.

```
context_features   %w(special token2 isPunctuation isDelim)
context_window     %w(1 2 3 -1 -2 -3)
```

**Fig. 2.** CRF++ context

## 4 Normalization

The Named Entity Recognition described before can only identify what could constitute a reference to a gene in the text. The actual gene that the mention refers is not determined by this process. Matching gene mentions to the actual genes is known as Normalization, although some systems have been proposed by BioCreative competitors that merge NER and Normalization in one single process. Normalization usually uses a lexicon that specifies, for each gene in a particular genome, all the synonyms used to refer to that gene in the literature. Similarity between the mention and the gene names is the main way to stablish matches. Our approach is based on a three step pipeline, finding first candidates genes for the mention, then selecting only those viable and, finally, choosing the best match among those, if any remain.

To find the candidate genes we use the idea of a name index. An index is built where the keys are names of genes and the values are the gene identifiers for genes that have that name. We then use a mention to query this index. To be more permissive, we allow these index to work, not only with the actual names and mentions, but with simplifications of them designed to increase matching. This way we stablish a hierarchy of indexes from more strict to more permissive. Each mention is matches against these indexes in order, stopping at the first match. The actual simplifications for each index are specified using a DSL, as exemplified in figure 3. Each name or mention is turned to several sets of cues, where each set of cues is aimed at a different index.

```
equal    do |w| [w] end
standard do |w| [w.downcase.split(/\s+/).sort.join("")] end
words    do |w|  w.scan(/[a-z]+/i)  end
```

**Fig. 3.** Declaration of name indexes

In the example figure 3 the first index we define is composed of the name or mention as-is; the second takes each word in the mention, sorts them, and joins them together in lowercase –this is a more accommodating cue; whereas the third index makes a cue for each word in the name or mention.

The candidates for each mention can contain plenty of false negatives, especially if very accommodating cues are used, therefore, it is necessary to filter out the worst and to select amongst the rest. In order to do so we created a

method that compares two names and evaluates their similarity. This method consists in chunking the name or mention down into it different components: number, letters, Greek names, Roman numerals, etc. For example, the mention `ASD-2 promoter` becomes the components `ASD, 2` and `promoter`. Each component –token from now on– is then identified and named. This identification is again defined using a DSL:

```
roman        /^[IV]+$/
promoter
greek        do |w| $greek[w.downcase] != nil end
```

**Fig. 4.** Token types

The three examples illustrate the three ways to define the token types. The first one identifies a token as a roman numeral using a regular expression. If only the name of the type is given, a default regular expression is attached, matching the same name, case insensitive, and with a possible "`s`" character at the end, to account for possible plurals. The last example uses a code block to make this check, in this particular case, it makes use of a hash of Greek letter names to check if the token is one of them, again, this is done case insensitive.

We now have each mention or name broken down into tokens, and the token types identified. In order to evaluate how similar a mention is to a certain name we need to evaluate the differences and similarities. This is done, again, using a DSL.

```
same.greek     5
miss.greek     -3
diff.promoter  -10
```

**Fig. 5.** Token comparison weights

Example figure 5 specifies three rules. The first states that if the same Greek letter name is found in the two mentions, a value of 5 is added to the similarity measure. The second states that if the mention is missing a Greek letter, 3 is subtracted, and the last one states that if one of them has the token `promoter` and the other does not, 10 is subtracted. We have four operators: `same, miss, extra` and `diff`. Meaning that they have a token in common, the mention is missing one, the mention has an extra token or that one of them has one the other does not. For each operator we can associate any of the token types declared in the previous step.

In order to add more flexibility to this method, we have added two other operators: `transform`, used to change tokens from one type to another, and `compare`, that allows comparing a certain type of tokens using a code block, thus giving more flexibility.

```
transform.roman do |t| [t[0].arabic, 'number'] end
compare.number do |l1,l2|
val = 0
val -= 4 if (l1 - l2).length >0 || (l2 - l1).length >0
val -= 8 if l1[0]  != l2[0] &&  l1[0]
val += 3 if l1[0]  == l2[0]
val
end
```

**Fig. 6.** Advanced comparisons: Transformations and custom comparisons

Figure 6 holds an example of both. The first one turns a roman to a number, allowing the number `1` to match `I` in roman form, for example. The second compares the numbers of mention and name in a finner grained manner.

We now have the ability to assign a similarity score between a mention and each of its candidate genes. For each candidate gene the best score amongst those from all it's known synonyms is selected. We use these score to rule out those candidates that score to low, and also to establish a ranking amongst those. If there are several candidate genes containing the same synonym (as it often occurs) they will score the same. In order to resolve the draws, we have a final step of disambiguation.

The disambiguation step is done by comparing the context in which the mention occurs, the paragraph for example, with the description of that gene in the Entrez Gene database. A code block can be provided in order to translate gene identifiers to Entrez Gene format if needed. The comparison is made by finding the words in common between the text in the mentions context and the text in the gene description, and adding their word weights. The word weights are their inverse document frequency calculated from a corpus of example text drawn from PubMed article abstracts.

## 5 Evaluation

The NER system comes with a set of default configurations mostly copied from Abner, but with a few additions. The system performs fairly well with these defaults for gene mention recognition. We plugged our system into the BioCreative competition evaluation sets using these defaults. The results are shown in table 1, we downloaded Abner and Banner and performed the same tests. RNer, our system, seems to outperform Abner slightly. Banner performs

significantly better than both, possibly because it uses syntactic information as features, which our default configuration did not include.

| System | Precision | Recall | F-Measure |
|---|---|---|---|
| RNer | 0.819 | 0.780 | 0.799 |
| Abner | 0.789 | 0.741 | 0.764 |
| Banner | 0.836 | 0.828 | 0.832 |

**Table 1.** Results for the BioCreative I task 1-B

The Normalization step also comes with a default configuration. We show results for the BioCreative I task 1-B in tables 2 and 3. These results are obtained using only the default configuration, no specific tuning is performed for either organism, nor is the provided training data used in any way. We have performed the analysis using our NER system, as well as Abner and Banner. Our NER system outperforms Abner and Banner in the yeast dataset, and Banner significantly outperforms both in the mouse dataset.

| NER | Precision | Recall | F-Measure |
|---|---|---|---|
| RNer | 0.936 | 0.863 | 0.898 |
| Abner | 0.941 | 0.809 | 0.870 |
| Banner | 0.933 | 0.816 | 0.870 |

**Table 2.** Results for the yeast dataset of the BioCreative I task 1-B.

| NER | Precision | Recall | F-Measure |
|---|---|---|---|
| RNer | 0.695 | 0.645 | 0.669 |
| Abner | 0.680 | 0.649 | 0.664 |
| Banner | 0.666 | 0.686 | 0.675 |

**Table 3.** Results for the mouse dataset of the BioCreative I task 1-B.

## 6 Discussion

Ruby, our implementation language, proved to be very useful due to its clear syntax and meta-programming possibilities. Ruby can also be compiled into Java bytecode and used in any Java framework. The result is a simple to use system that works out of the box but at the same time is flexible and easy to configure and adapt to other domains. Domain specific languages allow us to describe the specifics of the system in a more clear and succinct way.

While the system does not, for the time being, beat any record in performance, it shows competitive and promising results, even though no organism based tuning is performed and the training data for normalization is not being used.

## 7 Availability

This system was developed as part of another system called SENT, available at `http://sent.dacya.ucm.es`. Very likely, this system will be made available to public domain when SENT is published, along with the rest of the code that composes the SENT. In the meantime it will be provided free upon request.

## 8 Acknowledgements

This work has been partially funded by the Spanish grants BIO2007-67150-C03-02, S-Gen-0166/2006, CYTED-505PI0058, TIN2005-5619. APM acknowledges the support of the Spanish Ramón y Cajal program.

## References

1. L. Chen, H. Liu, and C. Friedman. Gene name ambiguity of eukaryotic nomenclatures. *Bioinformatics*, 21(2):248–256, 2005.
2. L. Hirschman, M. Colosimo, A. Morgan, and A. Yeh. Overview of BioCreAtIvE task 1B: normalized gene lists. *BMC Bioinformatics*, 6(1):S11, 2005.
3. T. Kudo. Crf++: Yet another crf toolkit., 2005.
4. J.D. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the Eighteenth International Conference on Machine Learning table of contents*, pages 282–289, 2001.
5. R. Leaman and G. Gonzalez. Banner: An executable survey of advances in biomedical named entity recognition. In *Pacific Symposium on Biocomputing*, volume 13, pages 652–663. Citeseer, 2008.
6. U. Leser and J. Hakenberg. What makes a gene name? Named entity recognition in the biomedical literature. *Briefings in Bioinformatics*, 6(4):357–369, 2005.
7. B. Settles. ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191, 2005.
8. B. Settles, N. Collier, P. Ruch, and A. Nazarenko. Biomedical Named Entity Recognition using Conditional Random Fields and Rich Feature Sets. *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 107–110, 2004.
9. H. Shatkay and R. Feldman. Mining the Biomedical Literature in the Genomic Era: An Overview. *Journal of Computational Biology*, 10(6):821–855, 2003.
10. A. Yeh, A. Morgan, M. Colosimo, and L. Hirschman. BioCreAtIvE task 1A: gene mention finding evaluation. *BMC Bioinformatics*, 6:1, 2005.

## 4.6 Rbbt: A Framework for Fast Bioinformatics Development with Ruby

**Reference:** **Vazquez, M.**, Nogales-Cadenas, R., Carmona-Saez, P., Pascual-Montano, A. & Pavón, J. (2010), Rbbt: A framework for fast bioinformatics development with Ruby, *in* 'Proceedings', IWAPCBB'10: 3nd International Workshop on Practical Applications of Computational Biology & Bioinformatics. Accepted for publication.

**Summary:** Good software practices and development frameworks are still at an early stage in the bioinformatics field, far behind the current demands of molecular biology research. In particular, code reuse is hindered by the fact that many analysis functionalities are very data dependant; literature mining and other tasks that require data to train and evaluate models can serve as examples. Improving the development process of bioinformatics applications is crucial to upscale their turnout and consequently upgrade the quality, specificity, and depth of the analysis performed by molecular biologists.

In this work we present a development framework based on software practices that are tailored to the priorities of the bioinformatics field: flexibility, interoperability, and simplicity. The framework solves the problem of encapsulating complex reusable components with strong data dependencies by incorporating the data processing pipelines along with the functionality API.

Furthermore, interoperability between different tools is a common challenge in bioinformatics pipelines. Rbbt uses the versatility of the Ruby language to encapsulate third party software into the main functionality API, providing even simpler interfaces.

# Rbbt: A Framework for Fast Bioinformatics Development with Ruby

Miguel Vazquez, Rubén Nogales, Pedro Carmona, Alberto Pascual, and Juan Pavón

**Abstract** In a fast evolving field like molecular biology, which produces great amounts of data at an ever increasing pace, it becomes fundamental the development of analysis applications that can keep up with that pace. The Rbbt development framework intends to support the development of complex functionality with strong data processing dependencies, as reusable components, and serving them through a simple and consistent API. This way, the framework promotes reuse and accessibility, and complements other solutions like classic APIs and function libraries or web services. The Rbbt framework currently provides a wide range of functionality from text mining to microarray meta-analysis. The source code for the framework, as well as for several applications that use it, can be accessed at `http://github.com/mikisvaz`.

## 1 Background

Molecular biology produces data in many different fields, and in great amounts. There is a great potential for developing new analysis methods and applications by either merging data from different sources, or by taking advantage of the large scale of several data repositories. These applications are, however, hindered by the complications in developing such systems, as they may require complex data processing pipelines just to set up the application environment. The fact is that many application that implement similar or closely related functionality end up having to roll up their own processing back-end pipelines, which are often very similar to one another and

Miguel Vazquez · Juan Pavón
Dep. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense Madrid

Rubén Nogales · Pedro Carmona
Dep. Arquitectura de Computadores y Automática, Universidad Complutense Madrid

Alberto Pascual
Biocomputing Unit, National Center for Biotechnology-CSIC

may actually account for a great portion of the development time. It would increase the turnout on these applications if these processing pipelines where implemented in a clean and structured way such that they could be shared and reused between different applications.

In fact, this approach would not only allow for sharing just these pipelines, but would also allow for more sophisticated functionality to be exported as APIs, thanks to the possibility to easily install and process the supporting data. Things like identifier translation, gene mention recognition and normalization, or annotation enrichment analysis are good examples. The problem of providing APIs for these types of functionality, those that are very data dependant, is partially solved by allowing the necessary data files to be shipped along with the API. This solution, however, does not apply well when the data is to large, or is not static, either because it must be updated periodically, or because it must allow for some type of customization on how it is produced.

One solution to this problem is to build our own ad-hoc back-end systems, and export the functionality through the use of web services, such as those following the SOAP and REST communication protocols. This approach has several advantages, in particular, that the complexity of installing and administering the system falls only over the system developers and not over the API users. It does, however, have one important drawback with respect to classic APIs, other than possible reliability or latency problems, and that is that open source API can be modified and adapted to particular needs, while these web services are static, and their behaviour is entirely controlled by the developers and maintainers of the application.

The solution proposed in this work, and implemented by the Rbbt framework, is to develop the back-end processes in such a way that they can be shipped with the API so that they can install, automatically, the complete execution environment on the users system. This way, the users will have local access to the functionality while still maintaining the possibility of altering the code to fit their needs. Furthermore, by encapsulating all the complexity of the system behind a friendly to use top level API interface, the user can include this functionality, otherwise very complex, easily into their own scripts and applications, thus expanding the range of possibilities in their own analysis workflows.

Another important benefit of this approach is that the users will have direct access to the data used to provide the functionality, which can, in turn, allow for developing other functionality over this same data. Actually, by having access to the actual processing pipelines, these themselves can be adapted to process the data into any other format that suits better to the needs of the user.

By following a set of design practices, the process of implementing new analysis applications may leave behind a collection of such high-level functionality that can then be reused in other applications. In this fashion, the SENT and MARQ applications [10, 11] ended up producing the functionality in Rbbt, which where then expanded by including the functionality in Genecodis [6]. These functionality can now be used to develop new applications very fast, as much of the heavy lifting has already been done.

## 2 Implementation

Since one of the main objectives was to offer a cohesive top level API that provided access to all the functionality, we chose the Ruby language to implement that API. Ruby is a versatile language that can be used for small processing scripts, very much like Perl, but can also be used for web development using state-of-the-art frameworks such as the popular Ruby-on-Rails. This way, the usefulness of the API is maximized, since it can be used over the complete application stack, from small maintenance scripts to large web sites.

Ruby can also interface fairly easily with other tools such as the R environment or Java APIs. This allows the Rbbt API to wrap around these tools and offer their services through the main API, thus helping reduce the complexity of the user code by hiding it behind the API and data processing tools, which can take care of compiling and installing third party software.

Rbbt is designed to serve functionality that otherwise could not be included in classic API due to their strong data dependence. This means that the API must be able to find this data locally on the system, which in turn, requires a configuration tool that can take care of producing that data and installing it in the appropriate locations. We will refer to such locations as local data stores, and may include anything from data files, to third party software, or sandboxes with training and evaluation data to train models for machine learning methods. In fact, a few additional benefits arise by coupling the API functionality with these data stores, for instance, the possibility of implementing transparent caching strategies into the API.

The characteristics of data processing pipelines have similarities to compiling a software program, where certain files are produced by performing particular actions over their dependencies. So, for example, to produce a classifier for biomedical articles, one must build the model from the word vectors for positive and negative class articles, these word vectors are derived in turn from the text of these articles, which are themselves produced by downloading from PubMed the articles corresponding to the PubMed identifiers that are listed as positive and negative classes. Changing the way we compute the word vectors, for example, using a different statistic for feature selection, should only require updating the files downstream. The classic Make tool from UNIX offers a simple and clear way to define these dependencies and production steps. One drawback of Make is its rigid syntax; for this reason in Rbbt we use Rake, a Ruby clone of Make, that implements the same idea, but in 100% Ruby code, so that it can directly harness the full potential of the language, including, of course, the Rbbt libraries.

Data files in the local data stores are preferably saved in tab separated format, which is easy to parse, examine and modify using standard text editors. Only when performance, or some other factor, becomes an issue a database or a key value store should be used. This helps to maintain the interpretability and accessibility of the data.

## 3 Features

The Rbbt package includes a collection of **core functionality** to help accessing online data, and managing the local data stores. These include access to online repositories such as PubMed, Entrez, BioMart, The Gene Ontology, etc. Some of the resources accessed through the Rbbt API, such as PubMed articles, or gene records from Entrez, are cached to increase speed and reduce load on the remote servers. Access to general online resources which are likely to be updated periodically is also cached, but in a different type of non-persistent cache so that can be purged easily when performing system updates.

For managing the data stores, Rbbt includes functionality to work with tab-separated data files. These files often have an identifier in the first column and a list of properties as the successive columns. The Rbbt API includes functions to load this kind of files as a hash of arrays and merge several of such structures by matching arbitrary columns, which is used for example to build the identifier translation files from data in different sources such as biomart and bach download files from organisms specific databases.

**Organism specific information.** Rbbt uses a configuration file for each supported organism to list all the details for processing the basic information. One of such resources, for example, is the identifier file, which lists of the gene identifiers in some particular format, followed by other equivalent identifiers for that gene in other formats. In the case of yeast, for example, the native format is the SGD identifier, and other supported formats include Entrez Gene, Gene Symbol, or several Affymetrix probe id formats. These identifiers are retrieve from batch download files from organisms specific databases merged with informations gathered from BioMart and Entrez gene. Also, the Rbbt lists Gene Ontology associations for each gene, common synonyms for gene names as can appear on the literature, and PubMed articles associated to each gene derived from Entrez GeneRIF or GO association files. This files are the seed for the rest of Rbbt functionality for that organism, from simple identifier translation to automatic processing of GEO datasets, or models for gene mention and normalization, it all can be processed automatically once these files are in place, which is done by just complete these simple configuration files. Rbbt currently support 8 model organisms: *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Saccharomyces cerevisiae*, *Schizosaccharomyces pombe*, *Caenorhabditis elegans*, *Arabidopsis thaliana*, and *Candida albicans*. Other organisms can be easily included by just completing the information in their configurations files.

**Text mining** functionality in Rbbt includes building bag-of-words representations for documents, strategies for feature selection like TF-IDF or Kullback-Leibler divergence statistics, document classification, and several strategies for named entity recognition. In the last category are a general purpose named entity recognition system using regular expressions, which, coupled with the Polysearch thesaurus can be used to find mentions to diseases, cellular locations, and several other things. Rbbt implements a special purpose named entity recognition system for genes, which is also known as gene mention recognition [9], as well as a normalization [5] engine that can be used to map the mentions into the corresponding genes. Both the gene

mention recognition and the normalization tasks have specific sandboxes inside the data stores with training and evaluation data to build and asses the models. Additionally, Rbbt wraps the Abner and Banner software APIs [7, 4] for gene mention, with the same top level interface, so that they can be used interchangeably with its in-house developed system. Our own gene mention system borrows most of its ideas from Abner and Banner; it is based on conditional random fields [3, 8], trained using CRF++ [2], but has a much simpler and flexible configuration system to define the features.

**Microarray Analysis** in Rbbt includes automatic processing pipelines that can compile a database of gene expression patterns from most of the thousands of datasets available in GEO [1], as well as other GEO series or any other microarray repository with a little configuration. This data consists of gigabytes of expression data that can be used in meta analysis applications. It also supports automatic translation of probe ids from different platforms into a organism based common reference format that can be used to perform analysis across experiments using different technologies. The API includes functionality to manage this automatic processing pipelines, an R function library in charge of the actual automatic processing, so that the data is easily used from this environment, and a collection of features to perform rank based comparison for content based retrieval of similar signatures.

## 4 Examples of use

To illustrate the simplicity of using the Rbbt library we will use two example tasks, identifier translation of gene identifiers and gene mention recognition and normalization. For more complete examples check the applications sample directory, or the code for SENT and MARQ, which represent full blown applications using the complete Rbbt API.

In order to have access to this functionality the user must only retrieve the rbbt package, and run the processing pipelines to set up the necessary data and learning models, which is done using the configuration tool packaged in Rbbt.

The first example, in listing 1, is a command-line application that can translate gene identifiers for a given organism into Entrez Gene Ids. The input list of gene ids, which is provided through standard input one line each, may be specified in any of the supported gene id formats, which, for instance, in the case of *H. sapiens* includes more than 34 different formats such as Ensemble Gene Ids, RefSeq, or Affymetrix probes. The whole scripts boils down to loading the translation index for the organism and the selected target format (Entrez Gene Id in this case), and then using that index to resolve all our input ids.

**Listing 1** Translate Identifiers

```
require 'rbbt/sources/organism'
require 'rbbt/sources/pubmed'

usage =<<-EOT
```

```
  Usage: #{$0} organism

  organism = Sce, Rno, Mmu, etc. See 'rbbt_config organisms'

  You will need to have the organism installed. Example:
  'rbbt_config prepare organism -o Sce'.

  This scripts reads the identifiers from STDIN.

  Example:
  cat yeast_identifiers.txt | #{$0} Sce
EOT

organism = ARGV[0]

if organism.nil?
  puts usage
  exit
end

index =
  Organism.id_index(organism, :native => 'Entrez Gene Id')
STDIN.each_line{|l| puts "#{l.chomp} => #{index[l.chomp]}"}
```

The second example, in listing 2 (only relevant part is shown), is a command-line application that, given an organism, and a query string, performs the query in PubMed and finds mentions to genes in the abstracts of the matching articles. In this case the task boils down to loading the gene mention engine (*ner*), the normalization engine (*norm*), performing the query to find the list of articles (*pmid*), and then going through the articles performing the following two steps: use the gene mention engine to find potential gene mentions in the articles text, which is basically the title and abstract of the article, and then using the normalization engine to resolve the mentions into actual gene identifiers, using the article text to disambiguate between ties. The gene mention engine has several possible back-ends, Abner, Banner, and our own in-house development RNER, all based on conditional random fields. If RNER is used, the model for that specific organism must be trained, a process that, while been rather lengthy, is completely automated using the configuration tool, while still having room for customization, and needs only to be performed once. The normalization engine needs no training, and also has ample room for configuration.

**Listing 2** Gene Mention Recognition and Normalization

```
# Load data, this can take a few seconds
ner  = Organism.ner(organism, :rner) # Use RNER
norm = Organism.norm(organism)

# Query PubMed. Take only the last 'max' articles
pmids = PubMed.query(query, max.to_i)

# For each article:
PubMed.get_article(pmids).each{|pmid, article|
```

```
  # 1. Find mentions
  mentions = ner.extract(article.text)

  # 2. Normalize them
  codes = {}
  mentions.each{|mention|
    codes[mention] = norm.resolve(mention, article.text)
  }

  # 3. Print results
  puts pmid
  puts article.text
  puts "Mentions: "
  codes.each{|mention, list|
    puts "#{ mention } => #{list.join(", ")}"
  }
  puts
}
```

## 5 Conclusions

Open source APIs offer an inestimable resource for software developers. However, due to the complexity of bioinformatics analysis, this sharing was unfeasible for many functionality due to the strong dependence in elaborate data processing pipelines required to set up the applications environment. The use of web services opens the possibility of offering an API over this functionality while hiding these complex processes from the API user. However, web servers may suffer from unreliability or latency, and have much less room for adapting them to different circumstances. Rbbt has shown that the approach of constructing the processing pipelines themselves so that they can set up the data stores automatically not only allows to provide these same API locally, but has a number of additional benefits, in particular, more options in terms of modifying and adapting the code, reuse of the data files, and reusing the processing pipelines for other tasks.

Rbbt has been successfully used, in whole and in part, in developing several production ready applications. These applications where developed very fast with a limited development group; Rbbt and agile development practices played a fundamental role in the their fast turnout.

The source code for the framework, as well as for several applications that use it, can be accessed at `http://github.com/mikisvaz`.

## Acknowledgments

We acknowledge support from the project *Agent-based Modelling and Simulation of Complex Social Systems (SiCoSSys)*, which is financed by Spanish Council for Science and Innovation, with grant TIN2008-06464-C03-01. We also thank the support from the *Programa de Creación y Consolidación de Grupos de Investigación UCM-BSCH, GR58/08*

## References

1. Barrett, T., Suzek, T., Troup, D., Wilhite, S., Ngau, W., Ledoux, P., Rudnev, D., Lash, A., Fujibuchi, W., Edgar, R.: NCBI GEO: mining millions of expression profiles–database and tools. Nucleic acids research **33**(Database Issue), D562 (2005)
2. Kudo, T.: Crf++: Yet another crf toolkit. (2005). URL `http://chasen.org/taku/software`
3. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proceedings of the Eighteenth International Conference on Machine Learning table of contents pp. 282–289 (2001)
4. Leaman, R., Gonzalez, G.: Banner: An executable survey of advances in biomedical named entity recognition. In: Pacific Symposium on Biocomputing, vol. 13, pp. 652–663. Citeseer (2008)
5. Morgan, A., Lu, Z., Wang, X., Cohen, A., Fluck, J., Ruch, P., Divoli, A., Fundel, K., Leaman, R., Hakenberg, J., et al.: Overview of BioCreative II gene normalization. Genome Biology **9**(Suppl 2), S3 (2008)
6. Nogales-Cadenas, R., Carmona-Saez, P., Vazquez, M., Vicente, C., Yang, X., Tirado, F., Carazo, J., Pascual-Montano, A.: GeneCodis: interpreting gene lists through enrichment analysis and integration of diverse biological information. Nucleic Acids Research **37**(Web Server issue), W317–W322 (2009). URL `http://genecodis.dacya.ucm.es`
7. Settles, B.: ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text. Bioinformatics **21**(14), 3191 (2005)
8. Settles, B., Collier, N., Ruch, P., Nazarenko, A.: Biomedical Named Entity Recognition using Conditional Random Fields and Rich Feature Sets. COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004 pp. 107–110 (2004)
9. Smith, L., Tanabe, L., Ando, R., Kuo, C., Chung, I., Hsu, C., Lin, Y., Klinger, R., Friedrich, C., Ganchev, K., et al.: Overview of BioCreative II gene mention recognition. Genome Biology **9**(Suppl 2), S2 (2008)
10. Vazquez, M., Carmona-Saez, P., Nogales-Cadenas, R., Chagoyen, M., Tirado, F., Carazo, J., Pascual-Montano, A.: SENT: semantic features in text. Nucleic Acids Research **37**(Web Server issue), W153–W159 (2009). URL `http://sent.dacya.ucm.es`
11. Vazquez, M., Nogales-Cadenas, R., Arroyo, J., Botías, P., García, R., Carazo, J., Tirado, F., Pascual-Montano, A., Carmona-Saez, P.: MARQ: an online tool to mine GEO for experiments with similar or opposite gene expression signatures URL `http://marq.dacya.ucm.es`. Under submission

# Chapter 5

# Discussion of the Work

## 5.1 Overview

This thesis makes several contributions to functional analysis of high throughput experiments. These contributions cover all three approaches to functional analysis outlined in the background chapter: annotation enrichment analysis, literature mining, and meta-analysis. For each of these three approaches we have described an analysis method and accompanied it with a completely functional application.

All three applications offer complimentary approaches for functional interpretation. GeneCodis offers statistical analysis of annotations in biological databases. Its ability to merge different annotations, even from different resources, provides an additional level of specificity compared to that of singular annotations, which are also provided. The SENT application can be used to detect topics in the literature whose essence may not be easily captured by combinations of annotations from controlled vocabularies, or which may include facts not yet reflected on these databases. It also builds a literature index that can help explore the related literature by relevance to those particular topics. The MARQ application offers a completely different approach by directly using data from previous experiments, not focusing only on the relevance of previous research, but also giving access to the processed data from thousands of samples with an interface that enables in-depth exploration.

In addition to these applications, this thesis includes work on two related

problems. In BioNMF we have produced a web tool to perform non-negative matrix factorization: a factorization method whose high interpretability and insight guarantees wide applicability in data analysis. BioNMF has been used at the core of the literature mining strategy used in SENT. The other contribution made in this thesis is also related to literature mining : it is an implementation of a complete gene mention recognition and normalization system, which was, in fact, also used in the SENT methodology.

While the methods underlaying these applications cover a wide spectrum of statistical and machine learning approaches, matrix factorization, conditional random fields, frequent item set mining, hypothesis testing and rank based statistics, there are many similarities in the design for all applications. This thesis makes several contributions to exploiting these commonalties to generate high productivity frameworks for application development.

All applications share a common three layer design (see figure 5.1, and section 5.2). The core functionalities are provided through an API, which forms the first layer. The API is used by a web server to export a collection of data analysis methods; the second layer. The final layer is always a browser based application that offers a user friendly interface to the web server for launching analysis jobs and examining the results.

The Rbbt framework (Ruby bioinformatics toolkit) solves the problem of packaging complex and data-dependant functionalities in an API. Rbbt offers a unified access to all these functionalities that enables any script or application to access any of these functionalities programmatically with a simple interface. BioNMF is the exception here, as its core functionalities have not been ported to Rbbt and remain a separate project.

We have mentioned how the second layer of all these applications is a web server interface. This thesis presents a framework that conveniently encapsulates all the technical nuisance of providing SOAP based web service interfaces and implementing asynchronous jobs. This framework, SimpleWS, is used in all the applications.

The main objectives of this thesis are met in the form of:

- Three functional analysis applications that cover different approaches to the problem of functional analysis: GeneCodis, SENT and MARQ.

Each implementing some novel data analysis method based on statistics and machine learning techniques.

- One application. BioNMF, that performs non-negative matrix factorization, which has a wide applicability for data analysis and has been used in the SENT application.

- A general named entity recognition system, with a default configuration tailored for gene mention recognition. And a gene mention normalization system. Both used in the SENT application.

- Two frameworks for software development: Rbbt, implementing high level functionalities for several data retrieval and analysis tasks in bioinformatics, and SimpleWS, to ease implementation of SOAP based web services.

Accompanying the scientific publications, this thesis produced a source code repository covering a vast portion of the work. With the exception of BioNMF, which is not directly maintained by the doctorate, and GeneCodis, which has been ported to Rbbt but not yet released to public domain, the code for the rest of applications and frameworks is open and free to use; from the core analysis methods to the final browser based user interfaces.

Since the actual analysis methods for each of the products of this thesis are relatively independent, and the corresponding papers sufficiently self contained, the remainder of this section will detail the more common technical aspects, rather than the data analysis methods. This discussion will include some rather technical considerations with regards to programming language choices and software practices, which were not covered in the presented publications.

## 5.2 Interoperability and reuse

We have already discussed the importance of making applications accessible at various levels: From a user friendly application, a programmatic web server interface, and an API. In order to provide these different entry points without

having to duplicate code, we have set them up hierarchically to allow each level to relay the analysis on to the next one.

Figure 5.1 shows the proposed application design. There are three levels, one is accessed using the HTTP protocol, the next one is accessed using SOAP [1], and the third one is accessed directly via API method calls. Users are expected to access the applications through the HTTP protocol, via a user friendly browser applications, while scripts and workflows, such as those enacted using Taverna (Hull et al. 2006), can access the functionalities using the SOAP interface or directly using the API access.

The applications described in this thesis interact with one another in different ways, and, depending on the purpose, the interaction is done at different levels. For example, in some cases the functionality of one application is used to enhance the user experience when exploring the results of another. This is the case for SENT and GeneCodis, where the annotation enrichment analysis provided by GeneCodis is shown in the SENT browser interface when examining the results. This analysis can be used to determine the characteristics of each of the gene groups formed by SENT. Since the GeneCodis analysis is used only in the browser interface, the interaction is directly between the HTTP controller and the GeneCodis web server. A very different case would be the interaction of SENT and BioNMF, where the matrix factorization is performed in the core analysis, so the interaction must be between SENTs API and BioNMFs web service.

MARQ uses GeneCodis to annotate each signature in the database with GO terms, but unlike SENT it does so off-line. MARQ accesses the GeneCodis functionality directly through the API access; this is done because processing the complete list of signatures may take a long time and might block the access to the GeneCodis server. This alternative is possible thanks to the Rbbt API, which offers this kind of access to even complex functionalities, and can allow the GeneCodis functionality to be easily installed on any computer. On the other hand, local execution would not be feasible for the SENT-BioNMF interaction; analysis would take considerably longer if performed locally rather than being delegated to the high-performance

[1] note that the SOAP protocol may also use the HTTP protocol for package transfer, but does not require a browser

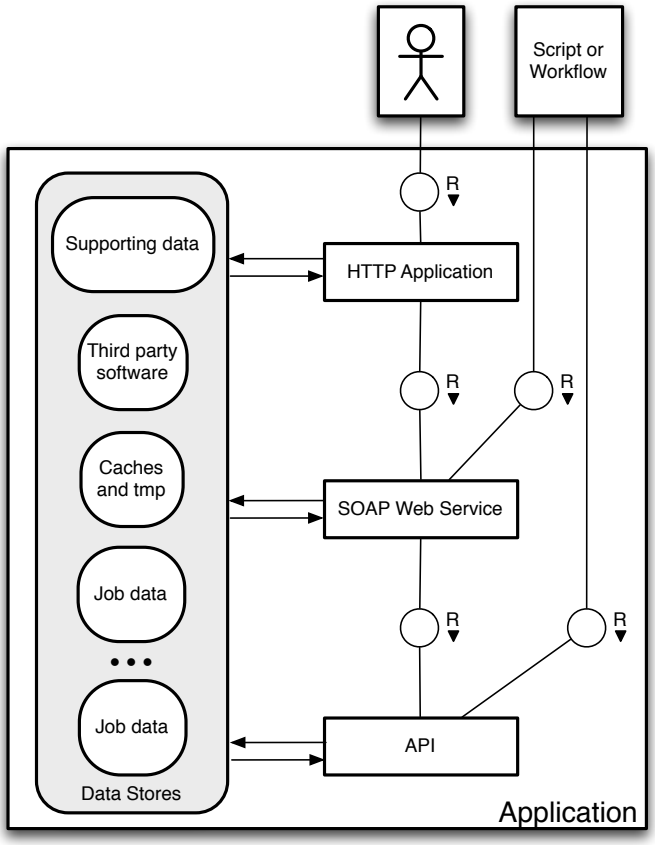


Figure 5.1: Application design. The functionality is ultimately served by the API, with the other two levels relaying the requests. All levels have access to the local data stores

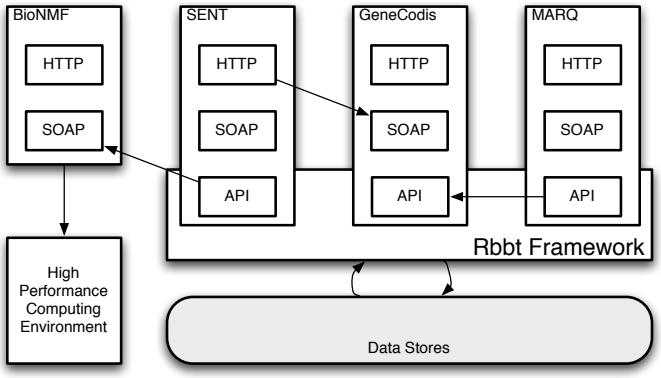


Figure 5.2: Interoperability within the application suite.

environment used by the BioNMF server.

Figure 5.2 shows a diagram of these interactions. Note how each may be at different levels for each of the actors. Also, note that the API for the GeneCodis, SENT and MARQ are all included in the Rbbt framework. Thus, Rbbt frameworks not only encourage sharing many core functionalities, for example those that deal with accessing the local data stores, but also allow for different applications to reuse each other's high level functionalities. Note also that the API for BioNMF has not yet been ported to Rbbt, and constitutes a separate project.

### 5.2.1 SOAP web services

All our applications offer their functionalities via web service interfaces. This tasks is made extremely simple thanks to the use of our SimpleWS framework.

The SOAP communication protocol (Box et al. 2000) has one very complex specification. The complete scope of the specification supports a level of interoperability not really required in the bioinformatics domain: SOAP supports specification of arbitrarily complex types such as function parameters and return values; this is not generally required for bioinformatics applications, where components are so different in nature. Arguably, complex types in this setting would do little but complicated interoperability, as it forces the client to define specific types in the object hierarchy just to hold the information used to connect with the web services. It is often simpler in

these cases to transmit the data in plain text using simple formatting (often tab separated values are enough) and parse the result in the client code into whichever data structure is deemed appropriate. This is specially true when using scripting languages such as PERL or Ruby, very popular among the bioinformatics community due specifically to their ability for processing text with great ease (Stajich et al. 2002).

Another important consideration regarding the use of SOAP technology is the production of the WSDL (Web Server Description Language) file. This file holds the necessary information for a client of the service to connect with the server. However, in a similar fashion to the whole SOAP specification, this file supports more functionality than is typically required in bioinformatics applications, which makes writing them unnecessarily complicated. SimpleWS offers a domain specific language (DSL) to write web services that takes care of producing the WSDL files. The SimpleWS DSL is as succinct as could be; only the name of the service, the names and types of the parameters and of the return value, and the block of code to execute are needed to produce a web service capable of exporting that operation and the WSDL files necessary to connect to it. Optionally, the DSL has primitives to add documentation to operations and their parameters, documentation that is automatically included in the WSDL file. For more information on DSLs see section 5.4 and appendix A.

There is one final characteristic of bioinformatics processes that is not contemplated directly in the SOAP specification: providing support for asynchronous jobs. Often a particular analysis cannot be completed within the life-span of a single HTTP connection, leaving no choice but to use asynchronous jobs. This is not supported by the SOAP specification directly, and so must be implemented ad hoc over the standard functionalities. In SimpleWS, using basically the same simple syntax, asynchronous jobs can be specified and have the web server include the necessary job management methods for their life cycle. Asynchronous jobs are implemented as background processes, and the job management methods included by default support for querying the status of the jobs, progress messages, gathering results or arbitrary key value information, or aborting a job.

With the SimpleWS framework producing web services becomes a trivial

task. Additionally, encapsulation of the technical details and automatic generation of WSDL help greatly to reduce errors. The framework has in fact been successfully used in SENT, MARQ, GeneCodis, BioNMF, and several other applications under development with a great record of stability. Since this thesis does not include a publication specific to SimpleWS, appendix A covers its design and implementation in more detail.

## 5.3 Rbbt: A framework for reusable functionalities

Implementing the GeneCodis, SENT and MARQ applications required a lot of functionalities. In SENT, for example, these include not only the functionalities related to performing the analysis over the gene-word matrix, but also those functionalities used in setting up the data, in particular the gene mention recognition and normalization functionalities. Some can be cumbersome to implement, in particular, those that require the presence of specific preprocessed data files. The approach we followed was to integrate the idea of a local data store into the API (which is the reason for choosing the term "framework" over the term "API"). That way, the relevant data files would be placed on these data stores where the API functionalities can find them. In order to set up these data stores, we include a configuration tool that manages the execution of a collection of data gathering and processing pipelines which perform all the necessary setup tasks.

All the functionalities susceptible to be reused on other applications are extracted into modules and packages in the base Rbbt package `rbbt`. Application specific functionalities are packaged into specific projects, such as `rbbt-sent` for SENT, while the web server and browser interface are packaged into a different packages, such as `rbbt-sent-www`. All these packages can be easily installed using the `gem` command, a Ruby packaging systems, much like `CPAN` for PERL. After installing each gem, a configuration tool can be used to launch the processing pipelines able to set up the necessary environment for the functionalities to work. Good examples of functionalities reused across applications are the automatic translation of gene identi-

fier across several database formats, building word vector representations for chunks of text, or computing the hypergeometric statistic.

The Rbbt package includes functionalities for a wide variety of tasks. Since not all the functionalities are always needed for a given application, the configuration tool can selectively launch each processing pipeline. Additionally, to reduce the complexity of the API, simple types, such as arrays and hash tables, are preferentially used over complex data structures. Using simple types facilitates using the functionalities for small processing scripts without complicating the code unnecessarily by instantiating and manipulating objects elsewhere defined. Simple types also reduce coupling between different parts of the API so they can be included separately, or even, portions can be copied and adapted for other projects.

The language used to implement the top level API in Rbbt is Ruby. Ruby is a scripting language very similar to PERL, but with a very strong object oriented design. It has most of PERL's best characteristics, like good text processing features and a succinct syntax, but is arguably cleaner and easier to learn, besides having some of the most sophisticated web development frameworks. This last feature is critical, since browser based applications are a preferred method of interfacing with an application for most researchers, specially those with less computer skills.

Some of the functionalities that are offered by third party packages are wrapped by the API. The data processing pipelines take care of downloading and compiling these packages, and the API offers a simple and cohesive top level interface for them. Following this approach, the cumbersome task of integrating different analysis tools need be resolved just once, and they can thereafter be used seamlessly.

## 5.4 Domain Specific Languages

We have already discussed some of the advantages of the Ruby language for the top level API, such as its text processing capabilities or its ample support for web development. Another important advantage of the language springs from its powerful meta-programming capabilities, which simplify defining Domain Specific Languages (DSL) of completely executable code. This, coupled

with the approach of using *convention over configuration*, a principle of agile methodologies popularized by the famous Ruby-on-Rails web development framework, can greatly simplify otherwise cumbersome and tedious tasks.

DSLs offer several advantages over the de-facto industry standard, XML files. XML files are verbose and hardly simple to read and edit manually. In fact, many applications have to provide specific authoring tools in order to allow users to edit them. DSL languages are designed to remove most of the syntactic noise and make the task of reading and editing configuration files easier, not requiring special authoring tools other than a simple text editor. Additionally, XML files are intended to provide configuration values, both simple values or structures of arbitrary complexity, but are not typically used to include any process information. On the other hand, DSL languages such as those used in our applications are 100% executable code, which means the grammar used to specify the configuration can also include any arbitrary code block. This kind of flexibility allows for the DSL to be as simple as possible for simple tasks, but still retain the ability to tackle more complex scenarios.

The approach of using DSL was first implemented for the tasks of gene mention recognition and normalizations, where all the domain specific options were separated from the actual workings of the algorithm by moving them to configuration files in DSL. In the case of the gene mention step, for example, the learning algorithm, conditional random fields, is a general approach that fits many different applications; all the domain specific configuration boils down to the definition of the features, which is done with very simple and expressive DSL.

Another great example is, of course, SimpleWS, which uses a DSL to offer an extremely simple and powerful interface to define web services. SimpleWS is one of the key players in all our applications, and its simplicity and expressiveness is an important reason behind our group's good application turnout. The appendix A section A.4 includes some examples of use that can serve as illustration for a particular DSL. Some more examples can be found the article presented in section 4.5.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

The work contributed in this thesis provides several novel methods for the analysis of bioinformatics data. The papers describing them demonstrate their applicability to relevant research problems using real datasets or popular gold-standards. The resulting applications offer three complementary ways to examine the results from high throughput experiments. In fact, their functionalities are often shared across these applications to enhance each other's results.

The methods proposed in these applications cover a wide range of data analysis techniques. GeneCodis uses statistical tools such as the chi-squared and the hypergeometric distributions, false discovery rates and permutation tests. SENT uses diverse literature mining techniques such as gene mention recognition and normalization using conditional random fields, and factor analysis using non-negative matrix factorization. BioNMF provides several alternatives for non-negative matrix factorization over a high-performance infrastructure. And finally, MARQ uses rank based statistics over signatures derived from gene expression microarrays analyzed using linear models and empirical Bayes.

The applications have been designed to favor their reuse, both by enabling access through the use of different interfaces, such as browsers, SOAP web services and APIs, and by following good software development practices.

Furthermore, the complete code for most of these applications is available through source control repositories. In fact, the code has been packaged into different projects to favor its reuse in other applications and by other developers.

One of this packages, SimpleWS, offers a simple domain specific language for defining web services which greatly simplifies their development. This framework has been successfully used in all four applications presented in this document, GeneCodis, SENT, BioNMF, and MARQ, as well as in several other projects currently under development.

Another of the packages, Rbbt, provides an infrastructure to implement complex functionalities offering a simple API access. This infrastructure is also used to wrap other systems, such as third party applications or R scripts, and provide access to their functionalities through the same coherent API.

Furthermore, the data analysis pipelines that prepare the data to support the functionalities are also designed to favor reuse, enabling it at several levels: Top level API, bottom level scripts (such as R scripts), processing pipelines, and finally, through the data itself, which is stored in a parser friendly format.

The Rbbt framework provides a core package with the basic framework functionalities and plug-in packages for each of the supported applications, including their online interfaces. This way anyone can completely deploy any of the applications by just installing the packages using the `gem` packaging tool for Ruby and running the configuration scripts.

## 6.2 Future lines of work

This thesis has set the basis for several lines of future work. One of the most obvious is, perhaps, the possibility of using all the application development infrastructure to increase the base of bioinformatics tools. We were able to take a method existing only as a proof of concept script and turn it into a full blown application with reusable API, web services and online interface in a period of a few weeks by a single programmer.

We are currently at work in developing such interfaces for two applications: TaLasso and XMIPP. TaLasso is a miRNA target prediction tool

for paired miRNA and gene expression arrays. This application implements several analysis methods to detect possible miRNA activity based on correlations of gene expression with miRNA concentrations. In particular, it implements a novel approach using lasso (Tibshirani 1996). On the other hand, XMIPP (Sorzano et al. 2004) is a package of functionalities regarding the analysis of microscopy image; the application we are developing will offer a unified web interface to some of its most important features, along with functionalities from other software packages such as normal-mode detection, docking or fitting.

Some other interesting lines of work follow directly from the MARQ application. On one hand, the information delivered by a MARQ analysis provides insights into experimental data that may suggest new working hypothesis which may in turn lead to other types of analysis; all of this is supported by the large repository of readily accessible data. Along this line, we are currently working on data from gastrin response gene expression microarrays with possible applications in colon cancer therapies.

On the other hand, the presence of such a data repository allows for a different type of general scope analysis. Given the gene signature matrix, the MARQ analysis can be seen as mining the signature vectors of gene expression. Other applications, such as SPELL (Hibbs et al. 2007), take the opposite approach; they compare gene expression profile under a collection of signatures to infer relationships between them. Analysis following this approach, based on gene profiles, can be easily implemented making use of the MARQ infrastructure.

The achievements reflected in this thesis open further lines of work in text mining, in particular in document classification and named entity recognitions; the development of new methods for performing these tasks will be aided by the previous infrastructure for gathering data and performing analyses. The Rbbt framework ships with sandboxes to test and evaluate classification, gene mention recognition and gene mention normalization. These sandboxes come equipped with test and evaluation data and scripts to build models and perform the evaluations. In particular, the gene mention recognition and normalization uses the BioCreative gold-standard datasets.

In the case of the SENT application, semantic features reported in the

results consist of a collection of relevant terms, having however no particular order or structure. Interpretation of these semantic features might be improved by mapping them back into controlled vocabularies such as GO or the more wide scoped UMLS (Bodenreider 2004). This approach is, in fact, under development.

In more general terms, the design of the Rbbt API, which has many analysis and data gathering tools implemented in a very general and loosely coupled way, will make much easier to try new analysis methods in new data sources. Literature mining remains one of the more promising approaches to enable the integration of very diverse information held in biomedical articles; by including more data gathering and analysis tools in Rbbt, the bioinformatics programmers will be able to implement more versatile scripts and analysis tools, in an effort to enable a more focused and controlled exploration of the literature under revision.

# Appendix A

# SimpleWS

The SimpleWS framework for SOAP web services is sufficiently developed, as proved by the applications that make use of it. It has not, however, been published yet as a technical article. This appendix explains its design and technical details. A user guide and examples of use can be found in the project sources `http://github.com/mikisvaz/simplews` and the help page `http://rbbt.dacya.ucm.es/simplews`.

The SimpleWS framework has two main levels. The first removes the technical difficulties of dealing with the SOAP protocol,such as producing the WSDL file. The second one builds upon the first to provide a simple way of implementing asynchronous jobs by taking care of all details in the job's life cycle.

## A.1 Basic SimpleWS

The basic SimpleWS uses the `SOAP4R` (`http://dev.ctor.org/soap4r`) package, which is a comprehensive implementation of the SOAP protocol for the Ruby language. Building upon this package, it develops a domain specific language (DSL) that allows the user to specify functionalities with just the necessary meta-data to produce the Web Server Description Language (WSDL) file. The basic SimpleWS is implemented by the class `SimpleWS`, which must be extended by the class that implements the actual web service.

The main method in the basic `SimpleWS` class is `serve`, which allows

exporting a currently defined method as a web service operation. The name of the method to export, which will also become the name of the operation, is specified as the first parameter to `serve`. The parameters for the exported method must receive explicit names and types in the web server operation.

The need to specify the types for the parameters comes from the fact that Ruby is dynamically typed while SOAP is not; this information must be made explicit to the `serve` method in order for the web service to be able to create the WSDL file. In fact, no other information is required, and the web server can take all the responsibility of producing these files. This is done by using a template and incrementing it with the necessary tags as operations are being served. The WSDL file can then be easily saved into a file and placed where a client can find it; the client program can then load it, and use it to automatically setup the driver to connect to the web service.

We have just discussed how any method defined for the web service class as an instance method can be provided through SOAP via `serve`. To avoid having to explicitly create such methods, `serve` accepts a block of code. The block of code will define a new method in the class, with the name of the operation (defined using the first parameter to `serve`), preceded by the `__inline_` prefix to avoid collisions with other method names. The method thus created at run-time will be the one exported as the operation. This alternative, while offering no practical advantage in terms of functionality, makes the source code cleaner and more succinct, and thus, less prone to errors.

In this same line, to further facilitate development and remove redundant code from the server, all the definitions of served methods can be extracted to a configuration file and loaded by a command called `start_ws`. This command creates a subclass of the `SimpleWS` class and instructs it to load the configuration file and has it evaluated at the class level as if the code in the configuration file was actually part of the class definition. The `start_ws` command also launches an instance of the web service to start listening for requests.

With all these enhancements, defining a web server boils down to defining just the strictly necessary elements: the code that implements the functionality, the name of the web service operation, and the parameters and their

types; this using a very clean and succinct syntax. The SimpleWS class actually has an embedded http server that requires no configuration and make deployment immediate.

## A.2 Asynchronous Job Support

Asynchronous jobs are implemented over the basic SimpleWS class, on a class called `SimpleWS::Jobs`. The life cycle of an asynchronous job is as follows.

1. The operation is called. A job identifier is returned. Job identifiers are unique. There could be several asynchronous operations, the job identifiers returned are used indifferently.

2. The job id is used to query the status of the job until its completion. During that time, the job can also be queried for its status or progress messages. Alternatively, the job can be aborted.

3. The status of a finished job, error or success, can be queried, and if it suffered an error, the error message can be retrieved.

4. If the job finished successfully the results can be gathered. These results may be in the form of files that can be downloaded, or in the form of a key-value store of arbitrary keys

All the steps in the life-cycle are automatically supported by the `SimpleWS::Jobs` class. The execution of the jobs is done as background processes following the sequence:

1. An object of the class `SimpleWS::Jobs::Scheduler::Job` is created, and a `run` method is added with the execution code serving the task. This class namespace comes equipped only with the necessary methods for the job to interact with the web service, status report, messages, producing result files, etc, but can be extended with user defined helper functions.

2. A new process is spawned, and the child process executes the `run` method of the `SimpleWS::Jobs::Scheduler::Job` object. After the

run method is finished the status is set to `done`. Exceptions are captured and result in setting the status to `error` and saving the exception message.

3. The `pid` of the child process is used to check termination periodically. If the job is found to have disappeared without reporting a `done` or `error` status, then the status is set to `aborted`.

4. The asynchronous job may indicate that it will be producing a collection of result files. When the status is `done` the web server allows the user to access these results, as it expects them to be placed in particular locations by the task. Consequently, background jobs are instructed about where to place the result files.

Communication between the web server and the job is done using a unique file for each job. This file contains the name of the job, the status, report messages, the key-value store used to compound arbitrary information, and the list of files the job is expected to have created upon successful termination.

The syntax to define asynchronous jobs is very similar to that of normal operations. Instead of using the `serve` method, `task` is used; it has an extra parameter, which is an array specifying the list of files that the job is programmed to create. As previously mentioned, each job receives a unique identifier. In order to facilitate job identification, an extra parameter is added automatically to each task; the parameter can then be used to specify a suggestion for the job id. If the job id suggestion matches an already existing job, then a consecutive number is appended. If the job id suggestion is an empty string, then a random sequence of letters and numbers is used as a job id.

## A.3 Documentation

As we mention in the first section of this appendix, the `SimpleWS` class is able to automatically build a WSDL file for the web service. Additionally, this WSDL can include documentation strings for the web service description, for each operation, and for the parameters of each operation, including

the return value. Optionally, this documentation can be specified using the `desc` command for operation descriptions, and the `param_desc` command for parameter descriptions; it accepts a hash with keys as parameter names and values as the description strings. The web service description is specified using a parameter of the construction method.

This documentation is not only included in the WSDL, but also in an HTML table that the web server can produce on request. The HTML table includes plenty of CSS tags designed to style the table accordingly, so it can be embedded in the online documentation of the applications. This approach greatly simplifies the production of completely documented web services with only very succinct meta-annotation.

## A.4 Examples

The first example shows the simplest possible web service. It implements the classic *Hello World!*. It does not specify the type for the return value, which is assumed to be a string.

Listing A.1: The *Hello World!* example

```
serve :hello do
  "Hello World!"
end
```

The next example (listing A.2) shows a similar function that receives a string parameter. It makes the types of the parameter and return value explicit, but this is not mandatory, since they would have been assumed strings.

Listing A.2: Another hello example

```
serve :hello, ['name'],
  :name => :string, :return => :string do |name|

  "Hello #{ name }"
end
```

Finally, listing A.3 shows a more complete example. It includes an integer parameter and returns an array of strings. Additionally, it also docu-

ments the operation and its parameters. As we mentioned, the WSDL file is generated automatically; a portion of that file is shown in A.4, notice the `documentation` tags. More examples, including for asynchronous jobs, and a complete example application with browser interface can be examined at `http://rbbt.dacya.ucm.es/simplews`.

Listing A.3: Example showing different parameter types and documentation

```
desc "Say hello to someone, several times."
param_desc :name   => "Person to say hello to",
           :times  => "How many times to say hello",
           :return => "Array of salutations"

serve :hello, ['name', 'times'], :name => :string,
  :times => :integer, :return => :array do |name, times|

  ["Hello #{ name }"] * times
end
```

Listing A.4: Portion of the WSDL file produced for server in listing A.3

```
<message name="helloRequest">
  <part type="xsd:string" name="name">
    <documentation>Person to say hello to</documentation>
  </part>
  <part type="xsd:integer" name="times">
    <documentation>How many times to say hello</documentation>
  </part>
</message>

<message name="helloResponse">
  <part type="tns:ArrayOfString" name="result">
    <documentation>Array of salutations</documentation>
  </part>
</message>

<portType name="SimpleWS">
  <operation name="hello">
    <documentation>Say hello to someone,
      several times.</documentation>
    <input message="tns:helloRequest"/>
```

```
    <output message="tns:helloResponse"/>
  </operation>
</portType>
```

# Appendix B

# Open Source Projects

This appendix briefly describes some relevant open source projects present in the `http://github.com/mikisvaz` repository. Most of these packages can be installed using the `gem` command.

`simplews` The package implementing the `SimpleWS` framework described in appendix A.

`rbbt` Contains the core functionalities for the Rbbt framework, such as organism support (identifier translation, gene synonym lexicon, etc), and text mining functionalities (vector space representation, gene mention and normalization, document classification, etc).

`rbbt-sent` Core functionalities for the SENT application, which are basically performing NMF factorization and analyzing the results.

`rbbt-sent-www` The web services for SENT and a complete online interface using Merb. This is the same interface used at `http://sent.dacya.ucm.es`.

`rbbt-marq` Core functionalities for MARQ. These include: automatic processing of GEO datasets, and semi-automatic processing of GEO series and arbitrary datasets (will a little configuration); rank based comparison of experiment signatures; Access to raw and processed data for GEO and arbitrary experimental data from R and Ruby; and annotat-

ing datasets with textual words, GO terms, UMLS concepts, Polysearch terms, or others.

`rbbt-marq-www` As with `rbbt-sent-www` this package contains the web service and a complete online interface in Merb, the same used at `http://marq.dacya.ucm.es`.

`DRbServe` This package can take any module and patch it run-time to export static methods through a `DRb` (Distributed Ruby) interface. The first instance run will create a new server, the following instances will patch their methods transparently to connect to the server. This way any module can be unobtrusively made distributed with no need to alter the original code.

`DBcache` This package offers functions to cache arbitrary hashes into a `MySQL` database.

`progress-monitor` This module can patch several types of loops down the execution call stack to monitor the progress. This way there is no need to include code for the actual progress report in the loop being monitored, as it can be unobtrusively patched run-time.

`TaLasso` **and** `xmipp-www` These two projects were created to implement web service (using SimpleWS) and online interfaces for XMIPP (Sorzano et al. 2004), and TaLasso, a work in progress for miRNA target prediction.

. .

.

# Bibliography

Abascal, F., Carmona-Saez, P., Carazo, J. & Pascual-Montano, A. (2008), 'ChIP-Codis: mining complex regulatory systems in yeast by concurrent enrichment analysis of chip-on-chip data', *Bioinformatics* **24**(9), 1208.

Al-Shahrour, F., Minguez, P., Tarraga, J., Montaner, D., Alloza, E., Vaquerizas, J., Conde, L., Blaschke, C., Vera, J. & Dopazo, J. (2006), 'BABELOMICS: a systems biology perspective in the functional annotation of genome-scale experiments', *Nucleic Acids Research* **34**(Web Server issue), W472.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. & Walter, P. (2002), *Molecular Biology of the Cell, Fourth Edition*, Garland.

Allison, D., Cui, X., Page, G. & Sabripour, M. (2006), 'Microarray data analysis: from disarray to consolidation and consensus', *Nature Reviews Genetics* **7**(1), 55–65.

Arroyo, J., Bermejo, C., García, R. & Rodríguez-Peña, J. M. (2009), 'Genomics in the detection of damage in microbial systems: cell wall stress in yeast.', *Clinical microbiology and infection : the official publication of the European Society of Clinical Microbiology and Infectious Diseases* **15 Suppl 1**, 44–6.

Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J. et al. (2000), 'Gene ontology: tool for the unification of biology. The Gene Ontology Consortium.', *Nat Genet* **25**(1), 25–9.

Barrett, T., Suzek, T., Troup, D., Wilhite, S., Ngau, W., Ledoux, P., Rudnev, D., Lash, A., Fujibuchi, W. & Edgar, R. (2005), 'NCBI GEO: mining millions of expression profiles–database and tools', *Nucleic acids research* **33**(Database Issue), D562.

Barrett, T., Troup, D., Wilhite, S., Ledoux, P., Rudnev, D., Evangelista, C., Kim, I., Soboleva, A., Tomashevsky, M., Marshall, K. et al. (2009), 'NCBI GEO: archive for high-throughput functional genomic data', *Nucleic acids research* **37**(Database issue), D885–D890.

Benjamini, Y. & Hochberg, Y. (1995), 'Controlling the false discovery rate: a practical and powerful approach to multiple testing', *JOURNAL-ROYAL STATISTICAL SOCIETY SERIES B* **57**, 289–289.

Blaschke, C., Andrade, M., Ouzounis, C. & Valencia, A. (1999), 'Automatic extraction of biological information from scientific text: protein-protein interactions', *Proc Int Conf Intell Syst Mol Biol* **1999**, 60–67.

Bodenreider, O. (2004), 'The unified medical language system (UMLS): integrating biomedical terminology.', *Nucleic acids research* **32**(Database issue), D267–70.

Boutsidis, C. & Gallopoulos, E. (2008), 'SVD based initialization: A head start for nonnegative matrix factorization', *Pattern Recognition* **41**(4), 1350–1362.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S. & Winer, D. (2000), 'Simple object access protocol (soap) 1.1', *W3C Note* .
**URL:** *http://www.w3.org/TR/soap/*

Breitling, R., Armengaud, P., Amtmann, A. & Herzyk, P. (2004), 'Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments.', *FEBS letters* **573**(1-3), 83–92.

Brunet, J., Tamayo, P., Golub, T. & Mesirov, J. (2004), 'Metagenes and molecular pattern discovery using matrix factorization', *Proceedings of the National Academy of Sciences of the United States of America* **101**(12), 4164.

Carmona-Saez, P., Chagoyen, M., Rodriguez, A., Trelles, O., Carazo, J. & Pascual-Montano, A. (2006), 'Integrated analysis of gene expression by association rules discovery', *BMC bioinformatics* **7**(1), 54.

Carmona-Saez, P., Chagoyen, M., Tirado, F., Carazo, J. & Pascual-Montano, A. (2007), 'GENECODIS: a web-based tool for finding significant concurrent annotations in gene lists', *Genome Biology* **8**(1), R3.

Carmona-Saez, P., Pascual-Marqui, R., Tirado, F., Carazo, J. & Pascual-Montano, A. (2006), 'Biclustering of gene expression data by non-smooth non-negative matrix factorization', *BMC bioinformatics* **7**(1), 78.

Chagoyen, M., Carmona-Saez, P., Shatkay, H., Carazo, J. & Pascual-Montano, A. (2006), 'Discovering semantic features in the literature: a foundation for building functional associations', *BMC Bioinformatics* **7**, 41.

Chaussabel, D. & Sher, A. (2002), 'Mining microarray expression data by literature profiling', *Genome Biology* **3**(10), 1–0055.

Chen, L., Liu, H. & Friedman, C. (2005), 'Gene name ambiguity of eukaryotic nomenclatures', *Bioinformatics* **21**(2), 248–256.

Cichocki, A. & Zdunek, R. (2007), 'Regularized alternating least squares algorithms for non-negative matrix/tensor factorization', *LECTURE NOTES IN COMPUTER SCIENCE* **4493**, 793.

Cichocki, A., Zdunek, R. & Amari, S. (2007), 'Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization', *Lecture Notes in Computer Science* **4666**, 169.

Cichocki, A., Zdunek, R. & Amari, S. (2008), 'Nonnegative matrix and tensor factorization', *IEEE Signal Processing Magazine* **25**(1), 142.

Crick, F. (1970), 'Central dogma of molecular biology', *Nature* **227**(5258), 561–563.

Dalma-Weiszhausz, D., Warrington, J., Tanimoto, E. & Miyada, C. (2006), 'The affymetrix GeneChip platform: an overview', *Methods in enzymology* **410**, 3.

D'Arcangelo, G., Homayouni, R., Keshvara, L., Rice, D., Sheldon, M. & Curran, T. (1999), 'Reelin is a ligand for lipoprotein receptors', *NEURON-CAMBRIDGE MA-* **24**, 471–479.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T. & Harshman, R. (1990), 'Indexing by latent semantic analysis', *Journal of the American Society for Information Science* **41**(6), 391–407.

Dennis Jr, G., Sherman, B., Hosack, D., Yang, J., Gao, W., Lane, H. & Lempicki, R. (2003), 'DAVID: database for annotation, visualization, and integrated discovery', *Genome Biol* **4**(5), P3.

Dopazo, J. (2006), 'Functional interpretation of microarray experiments', *OMICS: A Journal of Integrative Biology* **10**(3), 398–410.

Dueck, D., Morris, Q. & Frey, B. (2005), 'Multi-way clustering of microarray data using probabilistic sparse matrix factorization', *Bioinformatics* **21**(1), 144–151.

Efron, B. & Tibshirani, R. (2007), 'On testing the significance of sets of genes', *Annals of Applied Statistics* **1**(1), 107–129.

Efron, B., Tibshirani, R., Storey, J. & Tusher, V. (2001), 'Empirical Bayes analysis of a microarray experiment', *Journal of the American Statistical Association* **96**(456), 1151–1160.

Frijters, R., Heupers, B., van Beek, P., Bouwhuis, M., van Schaik, R., de Vlieg, J., Polman, J. & Alkema, W. (2008), 'CoPub: a literature-based keyword enrichment tool for microarray data analysis', *Nucleic Acids Research* **36**(Web Server issue), W406.

García, R., Rodríguez-Peña, J. M., Bermejo, C., Nombela, C. & Arroyo, J. (2009), 'The high osmotic response and cell wall integrity pathways cooperate to regulate transcriptional responses to zymolyase-induced cell wall stress in saccharomyces cerevisiae.', *The Journal of biological chemistry* **284**(16), 10901–11.

Gentleman, R., Carey, V., Bates, D., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J. et al. (2004), 'Bioconductor: open software development for computational biology and bioinformatics', *Genome biology* **5**(10), R80.

Gentleman, R., Ihaka, R. et al. (1997), 'The R project for statistical computing', *R home web site* .
**URL:** *http://www.r-project.org*

Getz, G., Levine, E. & Domany, E. (2000), 'Coupled two-way clustering analysis of gene microarray data', *Proceedings of the National Academy of Sciences* **97**(22), 12079.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M. et al. (1999), 'Molecular classification of cancer: class discovery and class prediction by gene expression monitoring', *Science* **286**(5439), 531.

Griffiths-Jones, S., Saini, H., van Dongen, S. & Enright, A. (2008), 'miRBase: tools for microRNA genomics', *Nucleic Acids Research* **36**(Database issue), D154.

Guruceaga, E., Segura, V., Corrales, F. & Rubio, A. (2009), 'FactorY, a bioinformatic resource for genome-wide promoter analysis', *Computers in Biology and Medicine* **39**(4), 385–387.

Heger, A. & Holm, L. (2003), 'Sensitive pattern discovery with'fuzzy'alignments of distantly related proteins', *Bioinformatics* **19**(1), 130–137.

Heinrich, K., Berry, M. & Homayouni, R. (2008), 'Gene tree labeling using nonnegative matrix factorization on biomedical literature', *Computational Intelligence and Neuroscience* **2008**, 276535.

Hibbs, M. A., Hess, D. C., Myers, C. L., Huttenhower, C., Li, K. & Troyanskaya, O. G. (2007), 'Exploring the functional landscape of gene expression: directed search of large microarray compendia.', *Bioinformatics (Oxford, England)* **23**(20), 2692–9.

Hill, D., Smith, B., McAndrews-Hill, M. & Blake, J. (2008), 'Gene Ontology annotations: what they mean and where they come from', *BMC bioinformatics* **9**(Suppl 5), S2.

Hirschman, L., Colosimo, M., Morgan, A. & Yeh, A. (2005), 'Overview of BioCreAtIvE task 1B: normalized gene lists', *BMC Bioinformatics* **6**(1), S11.

Hirschman, L., Yeh, A., Blaschke, C. & Valencia, A. (2005), 'Overview of BioCreAtIvE: critical assessment of information extraction for biology', *BMC bioinformatics* **6**(Suppl 1), S1.

Hoffmann, R. & Valencia, A. (2005), 'Implementing the iHOP concept for navigation of biomedical literature', *Bioinformatics* **21**(90002).

Homayouni, R., Heinrich, K., Wei, L. & Berry, M. (2005), 'Gene clustering by Latent Semantic Indexing of MEDLINE abstracts', *Bioinformatics* **21**(1), 104–115.

Huang, D., Sherman, B. & Lempicki, R. (2009), 'Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists', *Nucleic Acids Research* **37**(1), 1–13.

Huang, Z., Tian, H., Hu, Z., Zhou, Y., Zhao, J. & Yao, K. (2008), 'GenCLiP: a software program for clustering gene lists by literature profiling and constructing gene co-occurrence networks related to custom keywords', *BMC Bioinformatics* **9**(1), 308.

Huedo, E., Montero, R., Llorente, I., Thain, D., Livny, M., van Nieuwpoort, R., Maassen, J., Kielmann, T., Bal, H., Kola, G. et al. (2005), 'The GridWay framework for adaptive scheduling and execution on grids', *SCPE* **6**(8).

Hughes, T., Marton, M., Jones, A., Roberts, C., Stoughton, R., Armour, C., Bennett, H., Coffey, E., Dai, H., He, Y. et al. (2000), 'Functional discovery via a compendium of expression profiles', *Cell* **102**(1), 109–126.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M., Li, P. & Oinn, T. (2006), 'Taverna: a tool for building and running workflows of services', *Nucleic acids research* **34**(Web Server issue), W729.

Inamura, K., Fujiwara, T., Hoshida, Y., Isagawa, T., Jones, M., Virtanen, C., Shimane, M., Satoh, Y., Okumura, S., Nakagawa, K. et al. (2005), 'Two subclasses of lung squamous cell carcinoma with different gene expression profiles and prognosis identified by hierarchical clustering and non-negative matrix factorization', *Oncogene* **24**(47), 7105–7113.

Jelier, R., Jenster, G., Dorssers, L., Wouters, B., Hendriksen, P., Mons, B., Delwel, R. & Kors, J. (2007), 'Text-derived concept profiles support assessment of DNA microarray data for acute myeloid leukemia and for androgen receptor stimulation', *BMC Bioinformatics* **8**, 14.

Jenssen, T., Lægreid, A., Komorowski, J. & Hovig, E. (2001), 'A literature network of human genes for high-throughput analysis of gene expression', *Nature Genetics* **28**, 21–28.

Jones, K. et al. (2004), 'A statistical interpretation of term specificity and its application in retrieval', *Journal of documentation* **60**, 493–502.

Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., Tokimatsu, T. et al. (2008), 'KEGG for linking genomes to life and the environment', *Nucleic acids research* **36**(Database issue), D480.

Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y. & Hattori, M. (2004), 'The KEGG resource for deciphering the genome', *Nucleic acids research* **32**(Database Issue), D277.

Khatri, P. & Draghici, S. (2005), 'Ontological analysis of gene expression data: current tools, limitations, and open problems', *Bioinformatics* **21**(18), 3587–3595.

Kim, P. & Tidor, B. (2003), 'Subsystem identification through dimensionality reduction of large-scale gene expression data', *Genome Research* **13**(7), 1706.

Kochi, K., Lehmann, D., Pascual-Marqui, R. et al. (2006), 'Nonsmooth Nonnegative Matrix Factorization (nsNMF)', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(3), 415.

Kudo, T. (2005), 'Crf++: Yet another crf toolkit.'.
**URL:** *http://chasen.org/taku/software*

Lafferty, J., McCallum, A. & Pereira, F. (2001), 'Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data', *Proceedings of the Eighteenth International Conference on Machine Learning table of contents* pp. 282–289.

Lamb, J., Crawford, E., Peck, D., Modell, J., Blat, I., Wrobel, M., Lerner, J., Brunet, J., Subramanian, A., Ross, K. et al. (2006), 'The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease', *Science* **313**(5795), 1929–1935.

Leaman, R. & Gonzalez, G. (2008), Banner: An executable survey of advances in biomedical named entity recognition, *in* 'Pacific Symposium on Biocomputing', Vol. 13, Citeseer, pp. 652–663.

Lee, D. & Seung, H. (1999), 'Learning the parts of objects by non-negative matrix factorization', *Nature* **401**(6755), 788–791.

Lee, T., Rinaldi, N., Robert, F., Odom, D., Bar-Joseph, Z., Gerber, G., Hannett, N., Harbison, C., Thompson, C., Simon, I. et al. (2002), 'Transcriptional regulatory networks in Saccharomyces cerevisiae', *Science* **298**(5594), 799.

Leser, U. & Hakenberg, J. (2005), 'What makes a gene name? Named entity recognition in the biomedical literature', *Briefings in Bioinformatics* **6**(4), 357–369.

Lim, W., Wang, K., Lefebvre, C. & Califano, A. (2007), 'Comparative analysis of microarray normalization procedures: effects on reverse engineering gene networks', *Bioinformatics* **20**(13), i282–i288.

Lohmann, G., Volz, K. & Ullsperger, M. (2007), 'Using non-negative matrix factorization for single-trial analysis of fMRI data', *Neuroimage* **37**(4), 1148–1160.

Mejía-Roa, E., Carmona-Saez, P., Nogales, R., Vicente, C., Vazquez, M., Yang, X. Y., García, C., Tirado, F. & Pascual-Montano, A. D. (2008), 'bioNMF: a web-based tool for nonnegative matrix factorization in biology', *Nucleic Acids Research* **36**(Web Server issue), W523–W528.
**URL:** *http://bionmf.dacya.ucm.es*

Mitchell, J., Aronson, A., Mork, J., Folk, L., Humphrey, S. & Ward, J. (2003), Gene indexing: characterization and analysis of NLM's GeneRIFs, *in* 'AMIA... Annual Symposium proceedings [electronic resource]', Vol. 2003, American Medical Informatics Association, p. 460.

Morgan, A., Lu, Z., Wang, X., Cohen, A., Fluck, J., Ruch, P., Divoli, A., Fundel, K., Leaman, R., Hakenberg, J. et al. (2008), 'Overview of BioCreative II gene normalization', *Genome Biology* **9**(Suppl 2), S3.

Newman, J. C. & Weiner, A. M. (2005), 'L2l: a simple tool for discovering the hidden significance in microarray expression data.', *Genome biology* **6**(9), R81.

Nielsen, T., West, R., Linn, S., Alter, O., Knowling, M., O'Connell, J., Zhu, S., Fero, M., Sherlock, G., Pollack, J. et al. (2002), 'Molecular characterisation of soft tissue tumours: a gene expression study', *The Lancet* **359**(9314), 1301–1307.

Nogales-Cadenas, R., Carmona-Saez, P., Vazquez, M., Vicente, C., Yang, X., Tirado, F., Carazo, J. M. & Pascual-Montano, A. D. (2009), 'GeneCodis: interpreting gene lists through enrichment analysis and integration of diverse biolog-

ical information', *Nucleic Acids Research* **37**(Web Server issue), W317–W322. **URL:** *http://genecodis.dacya.ucm.es*

Pascual-Montano, A., Carmona-Saez, P., Chagoyen, M., Tirado, F., Carazo, J. & Pascual-Marqui, R. (2006), 'bioNMF: a versatile tool for non-negative matrix factorization in biology', *BMC bioinformatics* **7**(1), 366.

Pehkonen, P., Wong, G. & Törönen, P. (2005), 'Theme discovery from gene lists for identification and viewing of multiple functional groups', *BMC bioinformatics* **6**(1), 162.

Porter, M. (1997), 'An algorithm for suffix stripping (reprint)', *Readings in Information Retrieval* .

Raychaudhuri, S., Schütze, H. & Altman, R. (2002), 'Using Text Analysis to Identify Functionally Coherent Gene Groups', *Genome Research* **12**(10), 1582–1590.

Rhodes, D. R., Kalyana-Sundaram, S., Mahavisno, V., Varambally, R., Yu, J., Briggs, B. B., Barrette, T. R., Anstet, M. J., Kincead-Beal, C., Kulkarni, P., Varambally, S., Ghosh, D. & Chinnaiyan, A. M. (2007), 'Oncomine 3.0: genes, pathways, and networks in a collection of 18,000 cancer gene expression profiles.', *Neoplasia (New York, N.Y.)* **9**(2), 166–80.

Roberts, C. J., Nelson, B., Marton, M. J., Stoughton, R., Meyer, M. R., Bennett, H. A., He, Y. D., Dai, H., Walker, W. L., Hughes, T. R., Tyers, M., Boone, C. & Friend, S. H. (2000), 'Signaling and circuitry of multiple MAPK pathways revealed by a matrix of global gene expression profiles.', *Science* **287**(5454), 873–80.

Salton, G., Wong, A. & Yang, C. (1975), 'A vector space model for automatic indexing', *Communications of the ACM* **18**(11), 613–620.

Schulze, A. & Downward, J. (2001), 'Navigating gene expression using microarrays—a technology review', *Nature Cell Biology* **3**(8), E190–E195.

Schuster, S. C. (2008), 'Next-generation sequencing transforms today's biology.', *Nature methods* **5**(1), 16–8.

Sean, D. & Meltzer, P. S. (2007), 'GEOquery: a bridge between the gene expression omnibus (GEO) and BioConductor.', *Bioinformatics* **23**(14), 1846–7.

Settles, B. (2005), 'ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text', *Bioinformatics* **21**(14), 3191.

Settles, B., Collier, N., Ruch, P. & Nazarenko, A. (2004), 'Biomedical Named Entity Recognition using Conditional Random Fields and Rich Feature Sets', *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004* pp. 107–110.

Shatkay, H. & Feldman, R. (2003), 'Mining the Biomedical Literature in the Genomic Era: An Overview', *Journal of Computational Biology* **10**(6), 821–855.

Slonim, D. & Yanai, I. (2009), 'Getting started in gene expression microarray analysis', *PLoS computational biology* **5**(10), e1000543.

Smith, L., Tanabe, L., Ando, R., Kuo, C., Chung, I., Hsu, C., Lin, Y., Klinger, R., Friedrich, C., Ganchev, K. et al. (2008), 'Overview of BioCreative II gene mention recognition', *Genome Biology* **9**(Suppl 2), S2.

Sorzano, C., Marabini, R., Velázquez-Muriel, J., Bilbao-Castro, J., Scheres, S., Carazo, J. & Pascual-Montano, A. (2004), 'Xmipp: a new generation of an open-source image processing package for electron microscopy', *Journal of Structural Biology* **148**(2), 194–204.

Stajich, J., Block, D., Boulez, K., Brenner, S., Chervitz, S., Dagdigian, C., Fuellen, G., Gilbert, J., Korf, I., Lapp, H. et al. (2002), 'The Bioperl toolkit: Perl modules for the life sciences', *Genome research* **12**(10), 1611.

Stein, L. (2002), 'Creating a bioinformatics nation', *Nature* **417**(6885), 119–120.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005), 'Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles.', *Proceedings of the National Academy of Sciences of the United States of America* **102**(43), 15545–15550.

Tamayo, P., Scanfeld, D., Ebert, B., Gillette, M., Roberts, C. & Mesirov, J. (2007), 'Metagene projection for cross-platform, cross-species characterization of global transcriptional states', *Proceedings of the National Academy of Sciences* **104**(14), 5959.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society. Series B (Methodological)* **58**(1), 267–288.

Tjioe, E., Berry, M., Homayouni, R., Heinrich, K., Gu, W., Gerling, I., Jiao, Y., Gao, P., Xiong, Q., Ware, Z. et al. (2008), 'Using a literature-based NMF model for discovering gene functional relationships', *BMC Bioinformatics* **9**(Suppl 7), P1.

Tusher, V., Tibshirani, R. & Chu, G. (2001), 'Significance analysis of microarrays applied to the ionizing radiation response', *PNAS* **98**(9).

Vazquez, M., Carmona-Saez, P., Nogales-Cadenas, R., Chagoyen, M., Tirado, F., Carazo, J. M. & Pascual-Montano, A. D. (2009), 'SENT: semantic features in text', *Nucleic Acids Research* **37**(Web Server issue), W153–W159.
**URL:** *http://sent.dacya.ucm.es*

Vazquez, M., Nogales-Cadenas, R., Arroyo, J., Botías, P., García, R., Carazo, J., Tirado, F., Pascual-Montano, A. & Carmona-Saez, P. (2010), 'MARQ: an online tool to mine GEO for experiments with similar or opposite gene expression signatures', *Nucleic Acids Research* (Web Server issue). To appear in volume 38.
**URL:** *http://marq.dacya.ucm.es*

Venter, J., Adams, M., Myers, E., Li, P., Mural, R., Sutton, G., Smith, H., Yandell, M., Evans, C., Holt, R. et al. (2001*a*), 'The Sequence of the Human Genome'.

Venter, J., Adams, M., Myers, E., Li, P., Mural, R., Sutton, G., Smith, H., Yandell, M., Evans, C., Holt, R. et al. (2001*b*), 'The sequence of the human genome.', *Science* **291**(5507), 1304–1351.

Verhoeven, K., Simonsen, K. & McIntyre, L. (2005), 'Implementing false discovery rate control: increasing your power', *Oikos* **108**(3), 643–647.

Wilbur, J., Smith, L. & Tanabe, L. (2007), 'Biocreative 2. gene mention task', *Proceedings of the Second BioCreative Challenge Evaluation Workshop* pp. 7–16.

Wild, S., Curry, J. & Dougherty, A. (2004), 'Improving non-negative matrix factorizations through structured initialization', *Pattern Recognition* **37**(11), 2217–2232.

Wren, J. & Garner, H. (2004), 'Shared relationship analysis: ranking set cohesion and commonalities within a literature-derived relationship network', *Bioinformatics* **20**(2), 191–198.

Xie, X., Lu, J., Kulbokas, E., Golub, T., Mootha, V., Lindblad-Toh, K., Lander, E. & Kellis, M. (2005), 'Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals', *Nature* **434**(7031), 338–345.

Yeh, A., Morgan, A., Colosimo, M. & Hirschman, L. (2005), 'BioCreAtIvE task 1A: gene mention finding evaluation.', *BMC Bioinformatics* **6**, 1.

Yi, Y., Li, C., Miller, C. & George, A. L. (2007), 'Strategy for encoding and comparison of gene expression signatures.', *Genome biology* **8**(7), R133.

Zaki, M. & Hsiao, C. (2005), 'Efficient algorithms for mining closed itemsets and their lattice structure', *IEEE Transactions on Knowledge and Data Engineering* pp. 462–478.