UNIVERSIDAD DE COSTA RICA
SISTEMA DE ESTUDIOS DE POSGRADO

MODELADO HÍBRIDO DE DATOS GENÓMICOS DE ALTA DIMENSIONALIDAD PARA EL RECONOCIMIENTO DE PATRONES EN QUIMIOSENSIBILIDAD DEL CÁNCER

Tesis sometida a la consideración de la Comisión del Programa de Estudios de Posgrado en Computación e Informática para optar al grado y título de Maestría Académica en Computación e Informática

JUAN CARLOS COTO ULATE

Ciudad Universitaria Rodrigo Facio, Costa Rica

2017

Dedicatoria

A Anacris,

porque los sueños son más lindos

cuando soñamos juntos.

Agradecimientos

A Dios, por su amor; por enseñarme que el fracaso es un maestro que da lecciones invaluables.

A Anacris, por llenar mi vida de amor, risas, gatos y todo lo demás. Gracias por tu apoyo y motivación - especialmente cuando más cuesta. Te amo.

A Papi y a Mami por su apoyo y cariño, que en tantas maneras me ha impulsado a crecer.

A Francisco, Rodrigo y Álvaro, por haberme dado tantas oportunidades y por haber sido guías y amigos durante estos últimos años. Gracias también al PRIS-Lab por toda su colaboración.

Este trabajo se ha realizado como parte del proyecto «*Plataforma biocomputacional de análisis de datos genómicos para superar la resistencia a la terapia contra el cáncer y las infecciones microbianas multiresistentes*», el cual tiene como unidad base al Centro de Investigación en Tecnologías de Información y Comunicación (CITIC), inscrito bajo el código *834-B4-504* en la Vicerrectoría de Investigación de la Universidad de Costa Rica.

Esta tesis fue aceptada por la Comisión del Programa de Estudios de Posgrado en Computación e Informática de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Académica en Computación e Informática.

M.Sc. Alan Calderón Castro
Representante del Decano
Sistema de Estudios de Posgrado

Dr. rer. nat. Francisco Siles Canales
Director de Tesis

Dr. Rodrigo Mora Rodríguez, Ph.D.
Asesor

Dr. Álvaro de la Ossa Osegueda, Ph.D.
Asesor

Dr. Vladimir Lara Villagrán, Ph.D.
Director
Programa de Posgrado en Computación e Informática

Juan Carlos Coto Ulate
Candidato

Resumen

La resistencia del cáncer a las quimioterapias presenta uno de los principales problemas en los que se intenta aplicar la computación en la actualidad, debido a la cantidad de dimensiones que involucra su análisis. La ingeniería reversa basada en datos es una técnica computacional cuya aplicación permite identificar patrones de comportamiento a partir de un conjunto de observaciones de un fenómeno. Este proyecto consiste en el desarrollo de un sistema de reconocimiento de patrones para la representación y predicción del comportamiento de un conjunto de líneas celulares de cáncer de mama. Como fuente de datos se utiliza la *Cancer Cell Line Encyclopedia* (*CCLE*), un repositorio de dátos genómicos y transcriptómicos de líneas celulares de cáncer.

Para este fin se construye una base de datos genómicos del cáncer que permite realizar consultas y análisis genómicos relacionados con la quimiosensibilidad. Sobre estos datos, se plantean modelos híbridos, es decir, aquellos compuestos por modelos de clasificación y regresión, del nivel de expresión como función del número de copias de sus genes, que luego son evaluados con base en diversas medidas de bondad de ajuste. Además, se construyen herramientas de análisis y visualización exploratorios, que permiten la extensión de la plataforma por parte de investigadores con conocimientos básicos de programación estadística. Finalmente, se construye una infraestructura de procesamiento en paralelo que permite realizar análisis de modelos híbridos sobre todo el conjunto de datos utilizando más los recursos computacionales disponibles de forma más eficiente que mediante ejecución secuencial.

Abstract

Cancer cell resistance to chemotherapy is one of the principal problems where computational are currently being applied, due to the high number of dimensions that must be taken into account to analyze it. Data-based reverse engineering is a computational technique whose application enables the identification of behavior patterns from a set of observations of a given phenomenon. This project consists in the development of a computing system based on hybrid classification and regression models for the representation and prediction of the behavior of a set of breast cancer cell lines as a function of gene copy numbers and expression levels. The *Cancer Cell Line Encyclopedia* (*CCLE*), a repository of genomic and trascriptomic cancer cell line data, is used as the principal data source.

In order to construct this system, an integrated cancer genomic database was constructed. This database allows for efficient data querying, essential for chemosensitivity-related analyses. Hybrid models, those composed of classification and regression models, of gene expression as a function of gene copy number, are proposed and evaluated through different goodness of fit metrics. Additionally, exploratory analysis and visualization tools were created in order to allow for simple access to the genomic data, which require only basic knowledge of statistical programming languages in order to extend the system's functionality. Finally, a parallel processing infrastructure was also built, which allows for both more efficient use of computing and faster response time when executing batched modeling when compared to sequential execution.

Índice de Contenidos

Índice de Figuras

Índice de Cuadros

Índice de Fragmentos de Código

Glosario

Amplificación Aumento en el número de copias de un fragmento de ADN particular.

Aneuploidía Cambio en el número cromosómico, donde el número de cromosomas difiere del tipo silvestre.

Anotaciones El proceso de marcado de los genes y otras características biológicas de una secuencia de ADN, generalmente el genoma.

ARN mensajero Ácido ribonucleico que transporta la información genética procedente del ADN del núcleo celular al ribosoma, y que determina el orden en que se ensamblarán los aminoácidos de la proteína y actúa como plantilla o patrón para su síntesis.

Cancer Cell Line Encyclopedia Un repositorio de datos de caracterización genómica de un conjunto de líneas celulares de cáncer humanas, construido por el Broad Institute en colaboración con Novartis.

Clasificación El problema de la identificación de la categoría, dentro de un conjunto de categorías, a la cual pertenece una observación.

Cromosoma Una estructura, altamente organizada, formada por ADN y proteínas, que contiene la mayor parte de la información genética de un individuo.

Gen Una unidad de información en un locus de ácido desoxirribonucleico que codifica un producto funcional; es la unidad de herencia molecular.

Genoma El conjunto de genes contenidos en los cromosomas, que puede interpretarse como la totalidad de la información genética de una célula, tejido, órgano u organismo.

Genómica El conjunto de ciencias y técnicas dedicadas al estudio integral del funcionamiento, contenido, evolución y origen de los genomas.

Línea celular Un linaje de células de un tipo específico que son inmortalizadas y que se cultivan bajo condiciones controladas.

Microarreglo También conocido como chip de ADN, es una superficie sólida al a cual se adhieren fragmentos de ADN con la finalidad de analizar la expresión diferencial de genes.

Número de copias de ADN
La cantidad de copias de un gen particular en el genoma de un individuo.

Oncología de precisión
La práctica de utilizar terapias anticáncer específicamente dirigidas a un individuo particular - más allá de una población o segmento. Generalmente, esto requiere una personalización en la *combinación* de terapias existentes, en lugar de en la fabricación de agentes novedosos ajustados al caso particular.

Perfil génico Una representación matemática de las características genómicas de las células, en este caso dada como una función del número de copias y nivel de expresión de los genes.

Quimiosensibilidad Susceptibilidad de las células (cancerígenas en este caso) a los efectos de agentes terapéuticos que.

Regresión lineal Una técnica para modelar la relación entre una variable dependiente escalar y una o más variables independientes.

Resistencia a la quimioterapia
La capacidad de células cancerosas para adaptarse ante la acción de un agente quimioterapéutico y así evitar la muerte.

Transcrito Un producto de hebra sencilla de ARN sintetizado por la transcripción del ADN, que luego puede ser procesado para rendir diversos productos maduros de ARN como ARN mensajero, ARN de transferencia o ARN ribosomal.

Transcritos alternos Uno de los posibles productos sintetizados a partir de un fragmento de ADN que puede contener diversas porciones del mismo, seleccionadas mediante el proceso de splicing alternativo.

Transcriptómica El estudio de la totalidad del ARN que existe en una célula, tejido, órgano u organismo.

Capítulo 1

Introducción

En la actualidad, la cantidad de información generada y almacenada producto de la investigación científica, particularmente en áreas como la física o biología molecular, está en una fase de crecimiento acelerado. En los próximos 10 años, se espera que la biología molecular (y en particular la genómica) sea una de las principales áreas en cuanto a la producción de información y en cuanto a requerimientos de procesamiento sobre la misma (Stephens et al. 2015). En la actualidad, la necesidad siempre creciente de sistemas computacionales para la interpretación y procesamiento de los datos disponibles en el campo de la genómica no está siendo suplida. En general, en las diversas áreas de la investigación científica la cantidad de datos generados está en constante aumento - sin embargo, sin los sistemas computacionales adecuados no serán aprovechados adecuadamente.

El advenimiento de tecnologías de secuenciación de próxima generación (que reducen el costo de la secuenciación y aumentan la resolución) continuará expandiendo la brecha entre datos y capacidad computacional (Marx, 2013). Es importante notar que, para el procesamiento de datos de tal volumen, la implementación de nuevas estrategias de reducción de dimensionalidad es de particular interés, pues permitiría resolver problemas cada vez más complejos. La existencia de proyectos que avancen el estado del arte en la resolución de problemas de índole computacional relacionados con el procesamiento de datos genómicos podría facilitar la realización de nuevas investigaciones por parte de personal con niveles técnicos computacionales reducidos pero con amplia experiencia en sus campos respectivos (Stephens et al. 2015).

Una plataforma computacional que permita construir modelos simples y poderosos para la es-

timación del comportamiento del cáncer de mama podría avanzar significativamente el proceso de implementación de terapia personalizada. Mediante el uso de plataformas novedosas de secuenciación, es probable que en el futuro próximo sea posible contar con datos genómicos y proteómicos para cada paciente. Esto a su vez podría permitir la confección de programas terapéuticos específicos para cada persona, ajustados al perfil génico particular de su enfermedad (Ramsey et al. 2011; Yuan, Paskov, Paskov, González & Leslie, 2016).

Cabe destacar que un proyecto de este tipo se encuentra alineado con las metas de nuestro país, pues fomenta la interrelación de la computación con diferentes dominios, como la biología o la medicina (Kopper Arguedas & Vásquez Soto, 2015). Es, en resumidas cuentas, un esfuerzo transdisciplinario. Esta investigación puede considerarse como parte de la iniciativa de incorporación de temáticas de bioinformática y biología de sistemas al currículo de investigación de Costa Rica y Centro América, dentro de la cual la Universidad de Costa Rica juega un papel de liderazgo en la región (Orozco, Morera, Jiménez & Boza, 2013).

1.1. Problema

La interrogante principal que se busca responder con este proyecto es:

> *Es posible modelar matemáticamente la relación entre el perfil genético dado por el número de copias, el nivel de expresión génico y la respuesta a ciertas quimioterapias a partir de datos extraídos de un conjunto de líneas celulares de cáncer de mama. Dado este modelado, ¿es posible construir una plataforma biocomputacional para el reconocimiento de patrones en dichos datos de alta dimensionalidad?*

Definición formal

Sea L_k la matriz de datos de número de copias y expresión génica recopilados para todos los genes en la línea celular k. L_k tiene dimensión $g \times 2$, donde g equivale al número total de genes analizados. El valor L_{ki0} corresponde al número de copias génicas para el gen i representado como una función de $\log_2(a/b)$ donde a es el número de copias del gen i observadas en la muestra y b es el número de

copias del gen i observado en el tejido normal (generalmente 2). Finalmente, el valor L_{ki1} corresponde al nivel de expresión de ese gen en la muestra.

Sea G_j la matriz de datos de número de copias y expresión génica recopilada a través de todas las líneas celulares para el gen j. G_j tiene dimensión $l \times 2$, donde l equivale al número total de líneas celulares estudiadas. Similarmente a la definición anterior, G_{ji0} corresponde al número de copias del gen j en la línea i, mientras que G_{ji1} corresponde al nivel de expresión del gen j en la línea i.

Sea además T_j la matriz de datos de respuesta a tratamientos recopilados para todas las quimioterapiass disponibles en la línea celular j. T_j tiene dimensión $d \times 4$, donde d equivale al número total de quimioterapias probadas para la línea celular j. T_{ji0} corresponde al área sobre la curva de inhibición de la quimioterapia i para la línea j. T_{ji1} corresponde al valor de inhibición máxima lograda por la quimioterapia i en la línea j, como porcentaje. T_{ji2} corresponde al valor de la IC_{50} (dosis de quimioterapia a la cual se logró la inhibición del 50 % del crecimiento) de la quimioterapia i para la línea j. Finalmente, T_{ji3} corresponde a la EC_{50} (dosis a la cual se llegó al 50 % de la inhibición total lograda por la quimioterapia) de la quimioterapia i para la línea j.

Con estas definiciones, se propone la ecuación:

$$\mathcal{F}_k = \mathcal{H}_k(\mathcal{M}_k(G_{\{1,\ldots,g\}}, L_k), T_k) \tag{1.1}$$

Donde $\mathcal{M}_k$ es un modelo de clasificación no supervisada que toma cada fila de L_j y la asocia con un valor $m \in \mathbb{N}$, donde $m < g$. Por ende, realiza una reducción de dimensionalidad de g a m. Luego, $\mathcal{F}_k$ es un modelo de regresión que asocia una función $\mathcal{H}_k$ del modelo de clasificación para la línea celular k a la respuesta a tratamientos de la línea celular k.

1.2. Hipótesis

Es posible representar el perfil genético de las líneas celulares de cáncer de mama de la *CCLE* en un espacio dimensional reducido con respecto a aquel de los datos originales mediante modelos híbridos de número de copias y expresión

génica, que permita realizar asociaciones con la respuesta a quimioterapia, como se muestra en la ecuación 1.1.

1.3. Propuesta

A partir de la ecuación 1.1, entonces, es posible concretar qué es necesario para evaluar la hipótesis: proponer y construir una plataforma computacional que facilite el procesamiento de la información necesario para encontrar los $\mathcal{F}_k, \mathcal{M}_k, \mathcal{H}_k$ para todos los valores de k correspondientes a las líneas celulares estudiadas. Esto, a su vez, implica una simplificación de la representación del sistema y por ende una reducción de dimensionalidad del problema. Idealmente, se reducirá también el consumo de recursos computacionales necesario para realizar estimaciones sobre su comportamiento.

La plataforma planteada debe seguir un flujo de selección de características, plateamiento de modelos de caracterización y eventualmente predicción (De Niz, Rahman, Zhao & Pal, 2016). Entonces, como primer paso, se requiere la capacidad de realizar análisis exploratorios para guiar el proceso de modelado. Luego, se ha debe poder construir un *perfil génico*, es decir, una representación matemática de las características genómicas de las células. Finalmente, se busca encontrar asociaciones entre este perfil y las observaciones de respuesta a quimioterapias disponibles, para así tener como producto final un conjunto de reglas que permitan identificar nuevas y posibles estrategias de tratamiento para realizar oncología de precisión. Si bien estas actividades son los componentes principales de la plaforma computacional que aquí se detalla, la misma ha sido diseñada para permitir nuevos y diferentes análisis en la medida en que el los datos o conocimientos acerca del cáncer aumenten.

1.4. Objetivos

Objetivo General

Construir una plataforma computacional para el modelado de la relación entre número de copias y expresión génica de las líneas celulares representadas en la *CCLE*.

Objetivos Específicos

1. Integrar los datos de la *CCLE* en una base de datos que permita realizar consultas de dichos datos filtrados por gen, línea y respuesta a quimioterapias.
2. Caracterizar los datos de la *CCLE* mediante herramientas de visualización y análisis exploratorio.
3. Definir modelos de regresión y agrupación para el modelado del perfil genético.
4. Definir criterios de bondad de ajuste para la validación de la representación de los datos.
5. Construir una infraestructura de procesamiento de datos que permita ejecutar dichos modelos sobre el conjunto de datos de la *CCLE* .

En resumen, la solución que se plantea en este trabajo consiste de una plataforma computacional para la construcción de modelos híbridos (es decir, que pueden combinar tareas de agrupación, clasificación y regresión), del perfil genético para la eventual predicción de la respuesta a quimioterapias. Los modelos de regresión se utilizan para construir una representación simplificada del comportamiento de una línea celular con base en el número de copias y el nivel de expresión de sus genes. Por otra parte, los modelos de clasificación y agrupación buscan asociar las respuestas de las distintas líneas celulares a ciertas quimioterapias. Sintéticamente, este enfoque busca proyectar una representación de gran dimensionalidad (en donde cada gen y cada línea constituye una dimensión) a una de menor dimensionalidad (pertenencia a uno de varios grupos de genes o líneas).

Capítulo 2

Antecedentes

El cáncer es uno de los problemas más importantes a los que se enfrenta la ciencia en la actualidad. Por ende, existe un enorme cuerpo de investigaciones que pueden relacionarse con este tema. Seguidamente, se presentará una síntesis de trabajos recientes destacados en esta área. Finalmente, se resaltan los cuatro trabajos seminales sobre los cuales se inspiró esta plataforma.

2.1. Panorama general

Herramientas de visualizacion y consulta

Los trabajos clasificados en esta categoría presentan herramientas para realizar consultas, visualizaciones o comparaciones puntuales sobre diferentes tipos de datos relacionados con el cáncer. Se mencionan como antecedentes importantes para este trabajo porque ejemplifican componentes computacionales diseñados para complementar la investigación en laboratorio, tal y como lo hace la plataforma biocomputacional presentada en este trabajo.

Primeramente, Gao et al. (2013) muestran *cBioPortal* como una herramienta que facilita la consulta de datos de diversos grados de complejidad para realizar análisis de perfil genómico, posiblemente muy complejos Fig. 2.1. *cBioPortal*, a diferencia de este proyecto, se ubica en un nivel más bajo de consulta de datos y análisis preliminar - es decir, es un componente que facilita el acceso a datos pero no está diseñado para realizar análsis más extensos. Por otra parte, Varma, Pommier, Sunshine, Weinstein y Reinhold (2014) formularon un método para estimación de alta fidelidad del número de copias génicas a partir de diversas plataformas de medición. Además, publicaron esta información en

CellMiner, una herramienta Web en línea. El trabajo de Varma et al. (2014) ejemplifica la necesidad de poseer herramientas computacionales de alta calidad para el análisis de datos genómicos. La escala a la cual se trabaja en este tipo de proyectos requiere esfuerzos significativos para acceder, consultar y visualizar los datos. A diferencia del trabajo de Varma et al. (2014), esta plataforma permite a cualquier investigador con el conocimiento básico necesario incluir análisis y visualizaciones simples, o bien realizar procesos de modelado y validación más complejos.

Figura 2.1: Pantalla de consulta de cBioPortal. El sistema cBioPortal permite consultar información genómica del cáncer. **Fuente:** Gao et al. (2013)

Figura 2.2: CellMiner permite consultar datos de número de copias de líneas celulares del cáncer. *CellMiner* facilita el acceso a algunos análisis comunes predeterminados, además de datos crudos de origen genómico del cáncer. **Fuente:** Varma, Pommier, Sunshine, Weinstein y Reinhold (2014)

Con un enfoque mayor en la biología del cáncer, es importante destacar el trabajo de Sinha,

Schultz y Sander (2015), Gupta et al. (2016) y Nuzzo et al. (2016). Sinha et al. (2015) desarrollaron una herramienta de comparación genómica para identificar la validez de diferentes líneas celulares como modelos tumorales, *TumorComparer*. Esta herramienta asigna un peso a cada línea celular de acuerdo con la similaridad con respecto a líneas tumorales observadas en diferentes bases de datos. Este tipo de análisis es de suma importancia en etapas preliminares a un análisis más robusto como el que permite la plataforma biocomputacional aquí presentada. Previo a construir modelos de predicción del comportamiento de líneas celulares como modelos tumorales, es necesario valorar qué tan cercanas realmente son a los tumores. De esta manera, este trabajo es complementario al de la plataforma biocomputacional. Luego, Gupta et al. (2016) construyeron una herramienta para la predicción de la efectividad de terapias de vacunación a partir de diversas bases de datos de características de epitopos de cáncer Fig. 2.3. Esta herramienta busca apoyar esfuerzos para el diseño de vacunas semi-personalizadas, adecuadas a cada paciente a partir del perfil genómico y proteómico. Esta herramienta tiene un enfoque fundamental distinto a la expuesta en este trabajo, pues se encuentra orientada hacia inmunoterapia y es de interés principalmente para la vacunación anticancer. Es una herramienta para el análisis bioquímico primordialmente. Sin embargo, el flujo de trabajo que se encuentra ejemplificado en ese trabajo tiene un diseño similar al de esta tesis, pues permite tomar datos genómicos y a partir de ellos formular modelos que finalmente se podrán utilizar para tener estrategias de tratamiento concretas. Nuzzo et al. (2016) desarrollaron la herramienta *KAOS*, para identificar y visualizar valores extremos de expresión en genes de interés. Esta herramienta permite identificar sobreexpresiones en muestras pequeñas y sin requerir una segunda muestra como control.

Modelos basados en aprendizaje de perfil genómico

Los trabajos clasificados en esta categoría buscan resolver el problema de la sensibilidad a quimioterapias mediante aprendizaje computacional. Las estrategias que adoptaron son más complejas, y por ende imponen restricciones más fuertes sobre los datos que pueden utilizar (es decir, se debe contar con datos más complejos) que la plataforma biocomputacional de este trabajo. Este tipo de enfoque es sumamente útil para evaluar hipótesis muy particulares bajo condiciones específicas. La plataforma expuesta en esta tesis, sin embargo, busca encontrar asociaciones entre genómica y quimioterapia a partir de datos y modelos simples, cuya obtención es considerablemente más sencilla y económica.

Chen, Litvin, Ungar y Pe'er (2015) construyeron modelos predictivos basados en diversas fuentes

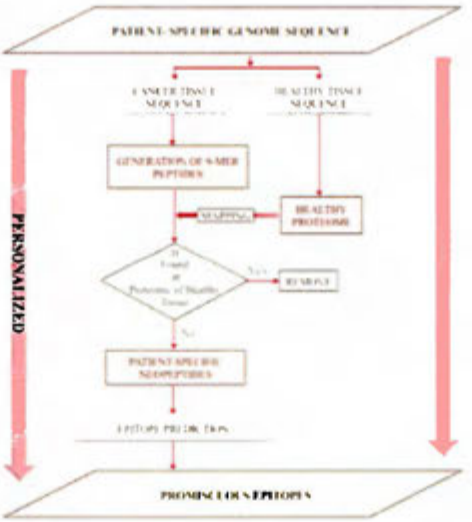

Figura 2.3: Flujo de información de CancerTope. El sistema CancerTope busca identificar posibles blancos terapéuticos para inmunoterapia por medio de vacunación. Para esto, compara el genoma de pacientes sanos con el de pacientes afectados por cáncer, e intenta identificar posibles terapias entre las secuencias afectadas que no tienen una secuencia sana correspondiente. **Fuente:** Gupta et al. (2016)

de datos, incluyendo la *CCLE*, además de información proveniente de la literatura, farmacología y genómica. El método que proponen, *CHER*, utiliza técnicas avanzadas de transferencia de conocimiento a través de diversos modelos para mejorar la predicción - relaciona el desempeño del modelo en una línea con el desempeño en otras de acuerdo con su similaridad. Su objetivo final es lograr obtener mayor conocimiento sobre las vías biológicas que son similares y divergentes entre los diferentes linajes del cáncer. El enfoque de nuestra plataforma es considerablemente más sencillo que el propuesto por Chen et al. (2015).

Dorman et al. (2016) proponen un modelo farmacológico basado en número de copias génicas, mutaciones, y expresión génica, tanto en líneas celulares como en pacientes. Se concentraron en una única droga, paclitaxel, y construyeron modelos supervisados (máquinas de soporte vectorial). Finalmente, formularon un modelo mediante aprendizaje estadístico y de máquina que fue capaz de clasificar acertadamente al 82 % de los casos Fig. 2.4.

Vougas et al. (2016) formularon un modelo reglas de asociación mediante Deep Learning con redes neuronales, que busca predecir la respuesta a 139 drogas anticancer con base en un conjunto de datos

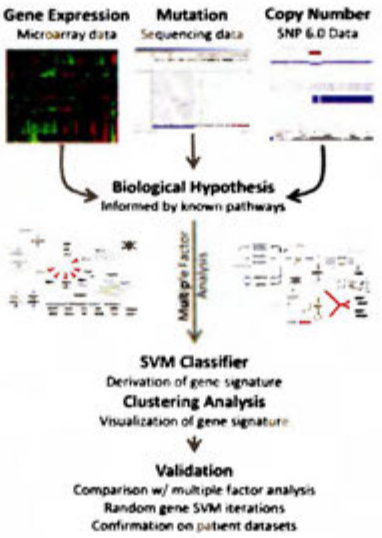

Figura 2.4: Flujo de trabajo para la predicción de efectividad de terapias sobre cancer de mama. Mediante un análisis de múltiples factores, Dorman et al. (2016) construyeron un modelo de clasificación de perfiles genómicos entre sensibles y resistentes para paclitaxel y gemcitabina. **Fuente:** Dorman et al. (2016)

compilado de diversas bases de datos como la *CCLE*, *The Cancer Genome Atlas*, *Genomics of Drug Sensitivity in Cancer* y *NCI-60*. El enfoque de este trabajo se basa en la construcción de un modelo predictivo que permita asociar qué tanto influye un gen en la respuesta de una línea celular a una quimioterapia.

Yuan et al. (2016) presentan, de manera similar a Chen et al. (2015), proponen un modelo basado en aprendizaje que busca aprender modelos de respuesta a quimioterapias, en una estrategia denominada *trace norm multitask learning*. La ventaja de este enfoque es que permite trasladar conocimiento entre diversos modelos para eventualmente obtener un modelo más efectivo en relación con modelos de regresión simples.

Modelos de simulación

Finalmente, esta sección presenta dos trabajos interesantes que tienen un enfoque distinto al de esta plataforma, a saber Liu, Kong, Gao, Zuliani y Clarke (2015) y Almendro et al. (2014). Ambos ilustran modelos detallados de procesos biológicos, los cuales dependen de un conocimiento profundo del comportamiento del perfil humoral y la dinámica evolutiva de los tumores, respectivamente.

Liu et al. (2015) formularon modelos de ecuaciones diferenciales para predecir la evolución del perfil hormonal de pacientes con cáncer de próstata. Debido a que este modelo se basa en la optimización de sistemas dinámicos de alto nivel de complejidad, su validación experimental es difícil. Sin embargo, provee un primer inicio en la aplicación de técnicas formales a problemas de farmacología de sistemas.

Almendro et al. (2014) desarrollaron un sistema de simulación de la dinámica evolutiva de células tumorales durante un programa de quimioterapia. Se basa modelos construidos sobre datos tomados antes y después de la aplicación de quimioterapia. Con estos modelos, puede formular una simulación *in silico* de los cambios en heterogeneidad tumoral provocados por la aplicación de los agentes quimioterapéuticos.

2.2. Investigaciones seminales relacionadas

Como investigaciones seminales para este trabajo, se puede destacar primeramente el trabajo realizado por Barretina et al. (2012), quienes recopilaron y construyeron la *Cancer Cell Line Encyclopedia*, fundamental para el desarrollo de este proyecto. Con esto, generaron un conjunto de datos rico en cuanto a variedad de genes muestreados, líneas celulares estudiadas y quimioterapias anticancer administradas. Adicionalmente, emplearon modelos Bayesianos simples para el reconocimiento de patrones de interés para el pronóstico de la malignidad del cáncer. Sin embargo, los modelos que construyeron no son muy sofisticados, por lo que existe interés en este proyecto de mejorar la calidad de las predicciones efectuadas sobre el conjunto de datos.

Luego, Solvang, Lingjærde, Frigessi, Børresen-Dale y Kristensen (2011) exploraron la importancia de la relación entre el número de copias de un gen y su nivel de expresión para identificar comportamientos aberrantes en el cáncer. En este proyecto, se utiliza esta relación como base para un método de caracterización del perfil génico.

Además, Thompson, Duarte, Marks y Congdon (2014) formularon un flujo de trabajo automatizado en forma de «pipeline» para el procesamiento de datos de expresión génica y la posterior agrupación de pacientes con base en su perfil, que se utilizó como inspiración para el enfoque de esta plataforma.

Adicionalmente, realizaron análisis de supervivencia basados en esta agrupación.

Finalmente, Dong et al. (2015) construyeron un modelo de clasificación basado en la *Cancer Cell Line Encyclopedia* para la quimiosensibilidad. Mediante su trabajo,establecieron la validez de la *Cancer Cell Line Encyclopedia* como base para análisis predictivos sobre quimiosensibilidad.

En el presente proyecto se combinan resultados y planteamientos de estas investigaciones junto con el concepto novedoso de una plataforma diseñada para permitir la construcción de modelos poco complejos (basados en número de copias y expresión génica). El fin último de esta plataforma es proveer herramientas computacionales para mejorar significativamente la calidad de la predicción de quimioterapia, al incorporar técnicas de selección de características y reducción de dimensionalidad más sofisticadas. En síntesis, este proyecto se basa en que la relación entre el número de copias de ADN de un gen y su nivel de expresión contiene información intrínseca pero decodificable de las redes de regulación génica que controlan el comportamiento de cada tipo de cáncer, lo cual debería permitir obtener conocimiento suficiente para predecir su respuesta a diferentes quimioterapias.

Capítulo 3

Marco Teórico

3.1. El cáncer y la genómica

La gran mayoría de las actividades celulares son llevadas a cabo por proteínas. Las proteínas son ensamblajes de aminoácidos, compuestos orgánicos que fungen como bloques fundamentales, sintetizados en el ribosoma de las células eucariotas (dentro de las cuales están incluidas las humanas). La manera en que los aminoácidos son ensamblados depende de secuencias de ácidos ribonucléicos, o ARN. Aunque existen diferentes tipos de ARN que afectan el proceso de síntesis de proteínas, el ARN mensajero (ARNm) está encargado de especificar el orden de los aminoácidos - por lo que es sumamente importante para determinar la estructura final de las proteínas. A través de un proceso complejo, el ARNm es sintetizado (o expresado) a partir del ácido deoxiribonucleico, o ADN. El ADN es también un polímero, y está encargado de acarrear información genética, en forma del código formado por sus subunidades. Una secuencia de ADN que codifica una proteína se llama gen, considerado como la unidad básica de almacenamiento de información genética, fundamental para la herencia. Al conjunto de todos los genes de una célula se le denomina genoma. En los eucariotas, los genes están localizados en unidades estructurales enrolladas en forma de hélice llamadas cromosomas. Los seres humanos tenemos 23 pares de cromosomas; así, en condiciones normales, tenemos dos copias de cada gen (una copia de cada gen por cada uno de los dos miembros de cada par de cromosomas) (Lodish et al. 2000).

Con base en lo anterior, es claro que el genoma es un componente fundamental del comportamiento de la célula. En él, se encuentran codificados todos los genes que son utilizados en los procesos celulares - efectivamente, es la materia prima de la actividad celular. De ahí que las alteraciones en el genoma

sean capaces de producir desórdenes catastróficos en la célula, tal y como se evidencia en el cáncer. El cáncer es una enfermedad compleja, caracterizada por un conjunto de desórdenes y comportamientos anormales que ocurren a nivel celular, los cuales finalmente provocan la pérdida de función normal en células, tejidos y órganos. Las alteraciones ocurren en procesos celulares fundamentales, tales como la proliferación, crecimiento y muerte, debido a complejas interacciones en el organismo, las cuales son alteradas por diversas fuerzas, provenientes de la genética, el ambiente, el estilo de vida o una combinación de ellas (Hajiloo et al. 2013).

El principal comportamiento aberrante de las células cancerosas consiste en su capacidad de proliferar de forma descontrolada, lo cual altera fundamentalmente las estructuras de los tejidos y conlleva a un compromiso de su función. Existen diversas maneras por las cuales se ha determinado que sucede esta alteración en la proliferación, como evasión de la muerte programada, invasión de otros tejidos (metástasis) y generación de vasos sanguíneos, entre otros (Hanahan & Weinberg, 2011).

Aunque no se conoce específicamente los mecanismos por los cuales se desarrolla el cáncer, como se menciona anteriormente, están basados en un desorden en el proceso de la síntesis de proteínas, lo cual desregula el funcionamento normal de las células. En efecto, se ha visto que ciertos genes están sobrexpresados (es decir, las proteínas para las cuales codifican se encuentran en mayor cantidad), mientras que otros sufren el proceso inverso. Además, los procesos cancerígenos producen aberraciones en la síntesis del ADN, de manera que frecuentemente existen más (o menos) de dos copias de un gen en una célula cancerígena (lo cual se conoce como aneuploidía, que a su vez es una situación que provoca alteraciones significativas en el comportamiento celular) (Stingele et al. 2012). Así, ocurren dos situaciones anormales concurrentemente: cambios en el nivel de expresión y en el número de copias de los genes. Actualmente, no se conoce con exactitud la naturaleza de la relación entre ambos eventos, o cómo su interacción afecta el destino de la célula, ya que la relación entre ellos es no lineal debido a la presencia de complejas redes de regulación génica que presentan propiedades de sistemas complejos, tales como compensación de dosis génica o biestabilidad (Vera, Lai, Schmitz & Wolkenhauer, 2013). Sin embargo, esta interacción provee una ventana al proceso de desregulación celular debido al cáncer. Finalmente, es importante notar que estos cambios son específicos a cada célula, por lo que hay una enorme heterogeneidad en las alteraciones de comportamiento que ocurren dentro de un mismo proceso canceroso (Solvang et al. 2011; Barretina et al. 2012).

3.2. *Cancer Cell Line Encyclopedia*

Una línea celular es una población de células cultivadas que, mediante un cambio transformativo, es capaz de crecer indefinidamente (Lodish et al. 2000); dado que muchos comportamientos de la célula están dados por su perfil génico, las líneas celulares son herramientas útiles para el análisis de un tipo particular de célula - específicamente tipos de tumores de cáncer. A partir de una línea celular, es posible realizar análisis que aproximen con un buen grado de certeza el comportamiento de un tumor primario - es decir, aquél que ha sido extraído de un paciente. En esencia, una línea celular permite simular, *in vivo*, un tipo particular de cáncer. La *Cancer Cell Line Encyclopedia*es un compendio de datos genómicos de 947 líneas celulares de cáncer y datos de respuesta a 24 fármacos para cerca de 500 de esas líneas. Esta enciclopedia de datos de cáncer comprende 36 tipos de tumores diferentes, e incluye estudios de mutaciones, comparaciones de similitud y otros (ver Fig. 3.1). Esta enciclopedia de datos genómicos y farmacológicos se encuentra disponible para la descarga gratuita (Barretina et al. 2012).

Dentro de los estudios realizados en la *Cancer Cell Line Encyclopedia*, se incluyen estudios de respuesta a agentes quimioterapéuticos. Estos estudios se representan mediante curvas de inhibición de la actividad celular, tal como la que se muestra en la Fig. 3.2. En esta curva, se identifican cuatro mediciones importantes, que pueden utilizarse para analizar cuantitativamente, qué tan efectivo es un agente quimioterapéutico para inhibir la actividad celular. La curva representa el porcentaje de inhibición relativa del crecimiento de una población celular, con un mínimo de 0% (no hay inhibición) y máximo de 100% (inhibición total del crecimiento), como función de la concentración del agente químico. Se espera que una curva de una droga efectiva inicie con 0% de inhibición con una dosis de 0, y logre una inhibición creciente a medida que la dosis incremente, posiblemente llegando al 100%. **Amax** representa la inhibición de actividad máxima lograda por el agente. **Activity area**, el área sobre la curva, es una medida continua que estima la efectividad total del agente; un agente muy efectivo tendrá un área sobre la curva mayor (dado que su actividad inicia a dosis menores y alcanza inhibiciones mayores, por lo tanto maximizando el área sobre la curva) mientras que un agente poco efectivo tendrá un área menor sobre la curva (pues la inhibición iniciaría a dosis superiores y no alcanzaría porcentajes de inhibición altos, lo cual resultaría en un área menor sobre la curva). $\mathbf{IC_{50}}$ cuantifica la dosis a la cual se logró inhibir al 50% la población de células estudiada, mientras que $\mathbf{EC_{50}}$ mide la dosis a la cual se logró la inhibición equivalente al 50% de la inhibición maximal del

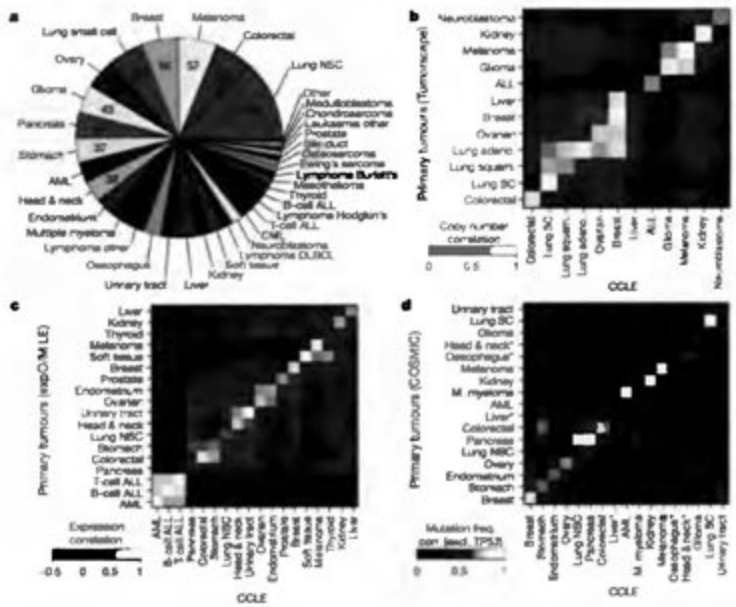

Figura 3.1: Clasificación de datos celulares disponibles como parte de la *Cancer Cell Line Encyclopedia*. Clasificación de líneas celulares por tejido que se incluidos a la Encyclopedia, además de la cantidad de diferentes líneas celulares disponibles para ellos **(a)**. Los mapas de calor, de menor a mayor (claro y oscuro, respectivamente) indican correlación entre las líneas celulares y tumores primarios (extraídos de pacientes) en cuanto a número de copias **(b)**, expresión génica **(c)** y frecuencia de mutaciones **(d)**. La alta correlación sobre la diagonal indica que las líneas celulares son una buena aproximación a tumores primarios (Barretina et al. 2012).

experimento (Barretina et al. 2012).

Como se menciona en la sección 2.2, Barretina et al. (2012) y Dong et al. (2015) han construido modelos predictivos para la estimación del comportamiento del cáncer con base en la *Cancer Cell Line Encyclopedia*, con buenos resultados; sin embargo, sus modelos dependen de muchos parámetros (mutaciones específicas, nivel de metilación de ADN, entre otros) lo cual compromete su simplicidad y limita su aplicabilidad a casos en los cuales se cuente con la información requerida. A su vez, para obtener esta información es necesario recopilar grandes cantidades de datos a partir de diversos tipos de ensayos clínicos, lo cual implica altos costos económicos y de material tumoral (el cual generalmente proviene de una muestra tomada por biopsia y por ende no se encuentra en grandes cantidades). Adicionalmente, no es posible obtener esta información detallada de células individuales con las tecnologías de análisis molecular disponibles actualmente; sin embargo, la obtención del número de copias y expresión génica es factible en células individuales, lo cual permite una granularidad mayor de los datos.

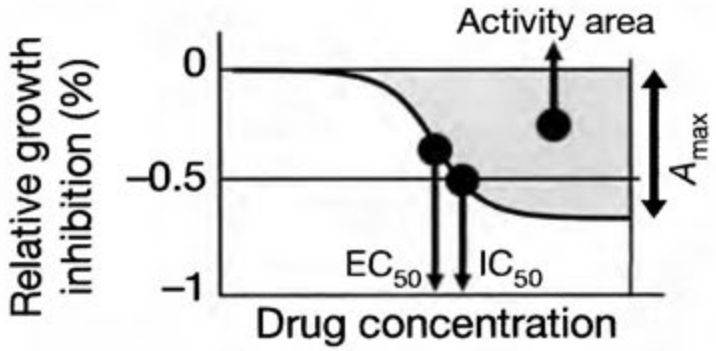

Figura 3.2: Ejemplo de curva de actividad de agente quimioterapéutico de la *Cancer Cell Line Encyclopedia*. Curva de inhibición de actividad celular producto de un agente quimioterapéutico, representada como porcentaje de inhibición en función de la concentración del agente. **Amax**: inhibición de actividad máxima lograda por el agente. **Activity area**: área sobre la curva de inhibición, efectivo como modelo representativo de la efectividad del agente. $\mathbf{IC_{50}}$: dosis de inhibición del 50 % de la población celular. $\mathbf{EC_{50}}$: dosis de inhibición del 50 % de la inhibición maximal (Barretina et al. 2012).

3.3. Ingeniería reversa de sistemas biológicos

El proceso mediante el cual se infieren las propiedades de un sistema biológico a partir de la observación de su comportamiento se denomina «ingeniería reversa de sistemas biológicos». Básicamente, puede llevarse a cabo de dos maneras: con base en diseño y con base en datos. La metodología basada en diseño consiste en modelar las propiedades conocidas de los sistemas biológicos y simular o abstraer propiedades emergentes, de manera que el comportamiento del modelo se asemeje a la realidad. El principal limitante de este acercamiento se encuentra en la necesidad de conocer de antemano partes fundamentales del funcionamiento del sistema, lo cual no siempre es posible (y, además, dependiendo de la cantidad de componentes de sus mecanismos, podría no ser factible en cuanto a procesamiento computacional). Por otro lado, la metodología de ingenería reversa basada en datos consiste en la observación y extracción de patrones a partir de observaciones registradas de sistemas biológicos reales en funcionamiento. Con los recientes avances en tecnología de recopilación de datos *-ómicos*, se ha hecho factible contar con suficientes datos como para extraer propiedades interesantes de los sistemas (Quo et al. 2012). El segundo tipo de metodología es de interés para este proyecto.

Al construir un marco de trabajo para la resolución de problemas de ingeniería reversa de sistemas biológicos basada en datos, es necesario enfrentarse a varios retos. Primero, se debe realizar un análisis exploratorio de los datos, con el fin de encontrar relaciones interesantes. Luego, se debe seleccionar

las características que describan mejor el fenómeno, de manera que puedan ser identificadas en las muestras que van a ser analizadas posteriormente (tanto en el proceso de entrenamiento del modelo subsecuente como durante su uso). Finalmente, una vez determinadas las características importantes, se procede a construir, entrenar y validar modelos estadísticos que permitan realizar predicciones (diagnósticos, pronósticos o estimación del comportamiento) (Quo et al. 2012). Esta estructura general es adoptada dentro de los procedimientos de este proyecto.

Durante el análisis de datos biológicos para el reconocimiento de patrones, sin embargo, se tiende a presentar un problema de resolución de datos, denominado «la maldición de la dimensionalidad». Esta situación se presenta cuando la cantidad de dimensiones de un problema (en este caso, la cantidad de genes observados) supera la cantidad de muestras disponibles, pues el espacio de soluciones crece exponencialmente debido a la cantidad de posibles combinaciones de valores de las distintas dimensiones. En ese caso, es necesario aplicar métodos de reducción de dimensionalidad o selección de características, que pueden incluir métodos de agrupación (Phan et al. 2009; Quo et al. 2012).

3.4. Reconocimiento de patrones basado en datos

El reconocimiento de patrones basado en datos consiste en la construcción de modelos, o representaciones abstractas, de alguna parte de la realidad instanciada mediante datos (Flach, 2012). La actividad de reconocimiento de patrones puede ser supervisada o no supervisada, de acuerdo con la disponibilidad de los datos existentes con respecto a la variable de respuesta que se desea evaluar (cuando se conoce el valor de la variable respuesta dado un objeto muestreado y cuando no se conoce, respectivamente). Un modelo generalmente toma un subconjunto de las propiedades descriptivas de los objetos muestreados por los datos, denominadas características, y las utiliza para de alguna manera exponer los principios fundamentales del objeto en observación (Hastie, Tibshirani & Friedman, 2009).

Los dos principales tipos de modelos que se formulan en el reconocimiento de patrones son de clasificación y de regresión. Los modelos de clasificación permiten identificar a un objeto como miembro de una clase o tipo, y realizan una asociación entre un objeto definido por una cantidad n de características y una clase discreta. Así, puede reducirse la dimensión de la muestra de n a 1. Los modelos de regresión toman las características del objeto y las asocian con un valor de la variable de respuesta,

por lo que únicamente pueden formularse en casos de reconocimiento de patrones supervisada (Flach, 2012; Hastie et al. 2009).

3.5. Datos de alta dimensionalidad

En problemas de análisis de espacios de alta dimensionalidad (donde el número de dimensiones es significativamente mayor a la cantidad de muestras evaluadas), como el que se describe en este documento, se ha descrito una característica denominada la «maldición de la dimensionalidad». Básicamente, a medida que aumenta la cantidad de dimensiones del espacio Euclideano de soluciones de un problema particular, se produce un aumento exponencial en el volumen de ese espacio. Por ende, la cardinalidad de los datos que se utilicen para muestrear el espacio debe ser mucho mayor para poder realizar inferecias fidedignas (Keogh & Mueen, 2011). Por esta razón, para este proyecto se presenta la necesidad de formular modelos que permitan reducir le dimensionalidad del problema de manera suficiente para representar adecuadamente el espacio de soluciones sin requerir un incremento exponencial en la cantidad de puntos de datos.

Capítulo 4

Metodología

En vista de lo expuesto en los capítulos 1 y 3, se ha propuesto entonces que existe una necesidad de construir una plataforma computacional para enfrentar el problema de la asociación entre el perfil genético y la respuesta a quimioterapia. Antes de exponer con detalle la metodología que se desarrolló para ese fin, cabe mencionar que se ha decidido conscientemente tomar el perfil genético únicamente como la relación entre el número de copias y expresión génica. La razón para esto es que, si fuera posible encontrar en un futuro próximo una manera de concretar esta asociación, sería posible implementar esta plataforma en casos de diagnóstico más personalizado, por ejemplo, para oncología de precisión. Sería posible, por ejemplo, utilizar tecnologías de secuenciación de células independientes para construir un perfil de una granularidad menor y por lo tanto más ajustado a la condición real del paciente, como lo planteado en Baslan et al. (2012). El contribuir al cuerpo de conocimiento existente para promover el avance de la terapia pesonalizada del cáncer es un objetivo fundamental de este proyecto.

4.1. Materiales

La arquitectura de la plataforma computacional presentada en este documento se divide en los siguientes componentes:

1. **Base de datos:** Base de datos relacional, montada sobre un servidor `MySQL 5.7.17`.
2. **Análisis y visualización exploratorios:** Servidor Web a la medida, códigos de análisis y visualización desarrollados en `RStudio`, y `Shiny`. Estos dos paquetes utilizan el lenguaje `R`, como base, y permiten obtener gráficos y resultados analíticos con un costo de desarrollo relativamente bajo, lo que los hace ideales para el análisis exploratorio.

3. **Plataforma de análisis por lotes:** Compuesta por varios módulos desarrollados en `Python 3.5.1`. Utiliza `SQLAlchemy` para conexiones a la base de datos, `Numpy` y `Scikit-Learn` para procesamiento vectorial y funciones de modelado, `Pandas` para las funciones de carga y transformación de datos y `Matplotlib` para graficación.

4.2. Integración de datos

El proceso de integración de base de datos siguió un flujo de trabajo de retroalimentación con usuarios expertos del grupo de trabajo, y partió siempre del supuesto de que el objetivo final de dicha base de datos era triple: suplir las necesidades de consulta para el problema de quimiosensibilidad en cáncer de mama; funcionar como una fuente única de verdad para el proyecto; y permitir combinar los datos de manera correcta y de nuevas maneras a medida que se conociera más acerca de ellos o que el proyecto fuera aumentando su alcance. Este tipo de proceso buscó adherirse en la medida de lo posible a las recomendaciones del diseño de bases de datos (Teorey, Yang & Fry, 1986; Devlin, 1996).

El diseño final de la base de datos sigue un modelo de almacén de datos, como otros sistemas con objetivos similares (Park et al. 2008; Bannasch et al. 2004; Kasprzyk, 2011; Lyne et al. 2013). Las tablas relacionadas con información general (genes, líneas celulares, experimentos, tipos de tejido) se organizaron como dimensiones, y se definieron dos tablas de hechos, `GeneProfileMatView` y `DrugResponseMatView` para recopilar los resultados de los diferentes experimentos de perfil genético y de respuesta a quimioterapia, respectivamente. La versión final de este diseño permite realizar consultas muy sencillas (e incluso exportar los datos a formatos aptos para la colaboración entre diferentes equipos de trabajo, como `CSV`) que otorgan toda la información necesaria. La rutina de acceso de datos que se utiliza para obtener toda la información disponible de perfil genético se encuentra en el Fragmento 4.1. Se requieren únicamente los nombres de las columnas adecuadas para realizar la consulta, y estas consultas son configuradas dentro de la sección de configuración del proyecto.

```
1    def get_iterator(initialized_funcs, limit=None, streamed=False):
2        fun = _result_set_iterator if not streamed else _cursor_iterator
3        return partial(fun, initialized_funcs, limit)
4
5    def _cursor_iterator(initialized_funcs, limit):
6        _, fetch_all_genes, fetch_gene = initialized_funcs
7        for gene in fetch_all_genes()[:limit]:
8            gene_data = fetch_gene(gene)
9            yield gene_data_as_profile(gene, gene_data)
```

Fragmento de Código 4.1: Modelos de regresión lineal y cuadrático

4.3. Módulo de análisis exploratorio y visualización

Este módulo consiste de funcionalidad para consulta a la base de datos (Fragmento 4.2), un servidor Web para visualización de los análisis realizados (Fragmento 4.3), y análisis exploratorios (Fragmento 4.4). La decisión de utiizar R como plataforma para desarrollar este módulo se basa en la rápida velocidad de desarrollo para modelos exploratorios, lo cual fue de gran ayuda para la iteración rápida durante las fases iniciales del proyecto.

```
1   GetAll <- function() {
2       conn <- GetConnection()
3       result <- dbGetQuery(
4           conn,
5           "select * from GeneProfileMatView;"
6       )
7       CloseConnection(conn)
8       result
9   }
10
11  GetDataByGeneSymbol <- function(symbol) {
12      conn <- GetConnection()
13      statement <- paste0(
14          "select * from GeneProfileMatView where symbol = '",
15          symbol,
16          "';"
17      )
18      result <- dbGetQuery(conn, statement)
19      CloseConnection(conn)
20      result
21  }
```

Fragmento de Código 4.2: Consulta a la base de datos

```
shinyServer(function(input, output) {
    output$visualization.output <- renderUI({
        computed.visualizations <- ExecuteVisualization(
            input$visualization.select
        )
        lapply(computed.visualizations, function(visualization) {
            render.function <- SelectRenderingFunction(visualization$graph.type)
            render.function(visualization$visualization())
        })
    })
})

SelectRenderingFunction <- function(visualization.function) {
    switch(visualization.function,
        plot = renderPlot,
        hist = renderPlot,
        plot3d = renderWebGL,
        h3 = h3
    )
}

ExecuteVisualization <- function(visualization) {
    visualization.index <- as.integer(visualization)
    if (visualization.index > 1) {
        visualization.to.execute <- visualization.list[[visualization.index]]
        visualization.to.execute()
    }
}
```

Fragmento de Código 4.3: Servidor Web de visualización. Muestra los resultados de los análisis como gráficos en una página Web.

```
RunGaussianModel <- function() {
    data <- GetAll()
    mix <- normalmixEM(data$quantileNormalizedRMAExpression)
    plot(mix, density=TRUE, cex.axis=1.4, cex.lab=1.4, cex.main=1.8, which=2)
    mix
}
```

Fragmento de Código 4.4: Análisis de ejemplo. Se agrupan los niveles de expresión en la cantidad de grupos que resulte en un valor óptimo de una métrica de entropía o distancia.

4.4. Módulo de construcción de perfiles génicos

```
def linear(x, y):
    regression = LinearRegression(fit_intercept=True)
    regression.fit(x, y)
    return (
        regression.predict(x),
        _get_linear_regression_model_description(regression)
    )

def quadratic(x, y):
    polinomial_x = PolynomialFeatures(degree=2).fit_transform(x)
    regression = LinearRegression(fit_intercept=True)
    regression.fit(polinomial_x, y)
    return (
        regression.predict(polinomial_x),
        _get_linear_regression_model_description(regression)
    )
```

Fragmento de Código 4.5: Modelos de regresión lineal y cuadrático

Con el fin de construir los perfiles génicos de las líneas celulares, se siguió un enfoque similar al expuesto por Solvang et al. (2011): una línea se puede caracterizar mediante qué tan bien es modelada por un modelo lineal o cuadrático. Aquellas líneas que tengan una bondad de ajuste mayor a un modelo lineal se consideran como lineales - y en otro caso como cuadráticas, si se ajustan a ese modelo, o anormales. Esto permite, de manera muy elegante y eficiente, tener unmodelo sencillo de cada gen a través de las líneas.

Como se muestra en el Fragmento 4.5, los modelos son fuciones muy simples - reciben un paráme-

tro x y un parámetro y y retornan una estimación y una lista de los coeficientes del modelo resultante luego del ajuste. Esto permite dos objetivos muy interesantes para el proyecto hoy y a futuro: facilidad de extensión (es sencillo incorporar nuevas características al sistema) y flexibilidad (los modelos pueden aplicarse a un amplio rango de problemas).

Para determinar qué tan bien se ajustan los datos a cada uno de los modelos de regresión evaluados, se definieron criterios de bondad de ajuste: coeficiente de correspondencia y suma de los cuadrados de las diferencias (ver Fragmento 4.6). Esta plataforma fue diseñada para poder implementar una cantidad arbitraria de métricas de bondad de ajuste, lo cual permite probar y desarrollar nuevos algoritmos en el transcurso de la vida futura del proyecto.

```
1    def residual_sum_of_squares(observed, predicted):
2        return np.sum(np.power((observed - predicted), 2))
3
4    def coefficient_of_determination(observed, predicted):
5        return _r2_score(observed, predicted)
```

Fragmento de Código 4.6: Funciones de cálculo de métricas de bondad de ajuste.

4.5. Módulo de agrupación de respuesta a drogas

Como parte del trabajo de agrupación, se realizaron modelos de agrupación de respuesta a drogas con base en el área sobre la curva de actividad, se utilizó como estimador de la efectividad de la droga para inhibir la actividad de las células de las diferentes líneas (ver sección 3.2). El Fragmento 4.7 muestra cómo se agrupan las diferentes líneas con base en el área de actividad.

```
class DrugResponseByCellLineModeler(object):

    def _train(self, model_function, training_data):
        ccle_name, act_area = training_data
        return model_function(ccle_name, act_area)

    def run(self):
        pipeline = Pipeline(
            self._train,
            [linear, quadratic]
        )
        return pipeline.run()

    def plot_drug_response_graphs(self, results):
        for data_set_result in results:
            name = data_set_result.data_set.name
            obs_ccle_name, obs_act_area = data_set_result.data_set.training_data
            lin_model, _ = data_set_result.get_model_results(linear)
            quad_model, _ = data_set_result.get_model_results(quadratic)

            scatter_plot = BidimensionalGraph(ScatterDataFormatter(), LineTrendFormatter())
            self._write_plot_to_path("~/Desktop/ddumps/{name}.png".format(name=name),
                scatter_plot.set_title("Drug Response Model for {name}".format(name=name)) \
                    .add_data(obs_ccle_name, obs_act_area) \
                    .add_trend(obs_ccle_name, lin_model) \
                    .add_trend(obs_ccle_name, quad_model, line_color="green")
            )
```

Fragmento de Código 4.7: Agrupación de datos de quimioterapia con respecto al área de actividad.

4.6. Módulo de procesamiento de modelos en paralelo

El reto más significativo en cuanto a recursos computacionales se refiere es el volumen de datos (5115 genes a través de 59 líneas celulares - además de al menos dos modelos y dos métricas de bondad de ajuste para cada gen). Para mitigar este problema, se construyó un módulo para la ejecución en paralelo de los modelos. Este diseño explota la ortogonalidad de los datos de cada gen - cuando se construye el perfil genético, cada gen es modelado de forma independiente. De esta forma, es posible ejecutar varios modelos a la vez. Con esta finalidad, se utiliza un diseño de recuperación de datos que

permite que los datos estén disponibles antes de comenzar a ejecutar los modelos, de manera que el acceso a la base de datos no se convierta en un cuello de botella. El Fragmento 4.9 muestra como permite ejecutar un flujo de trabajo a través de varios procesos. Por otro lado, el Fragmento 4.8 muestra la estructura básica de un flujo de trabajo: se itera a través de los datos, y para cada conjunto se ejecuta el modelo, y se captura el resultado. El resultado final de cada flujo de trabajo es una lista de modelos junto con sus resultados.

```
1   def pipeline(runner, analysis_functions):
2       assert runner is not None, "Must provide a training runner function"
3       assert analysis_functions is not None and len(analysis_functions) >= 1, \
4           "Must specify a list of analysis functions with at least one element"
5       return partial(_pipeline, runner, analysis_functions)
6
7   def _pipeline(runner, analysis_functions, data):
8       analysis_results = list()
9       for analysis_function in analysis_functions:
10          t_res = _analyze(runner, analysis_function, data)
11          analysis_results.append((analysis_function, t_res))
12      return data, analysis_results
13
14  def _analyze(runner, analysis_function, data):
15      return runner(analysis_function, data)
```

Fragmento de Código 4.8: Definición de un flujo de trabajo para ejecución de modelos sobre un conjunto de datos.

```
1  def parallel_runner(number_of_workers=cpu_count(),
2                      max_tasks_per_worker=50):
3      process_pool = _create_process_pool(number_of_workers, max_tasks_per_worker)
4      return partial(_run_in_parallel, process_pool)
5
6  def _create_process_pool(number_of_workers, max_tasks_per_worker):
7      return Pool(number_of_workers, maxtasksperchild=max_tasks_per_worker)
8
9  def _run_in_parallel(process_pool, pipeline, data_iterator):
10     result = process_pool.map(pipeline, data_iterator)
11     return result
```

Fragmento de Código 4.9: Módulo para ejecución en paralelo de los flujos de trabajo.

4.7. Módulo de graficación en lotes

Como se mencionó en la sección 4.6, el volumen de datos procesado por la plataforma es considerable. Se decidió utlizar la generación masiva de imágenes de los modelos como método para la verificación del funcionamiento de la plataforma. Para esto, se construyó infraestructura que utiliza `Matplotlib` para generar y almacenar imágenes de los modelos. El Fragmento 4.10, Fragmento 4.11 y Fragmento 4.12 muestran el código básico de la graficación. Fue desarrollado de manera que sea posible combinar diferentes métodos de graficación de datos con diferentes métodos de generación de líneas de tendencia, lo que ha permitido construir diversos tipos de gráficos con mucha facilidad. Adicionalmente, al ser diseñado mediante composición, es posible extender su funcionalidad sin requerir gran trabajo.

```
class BidimensionalGraph(object):

    def __init__(self, data_formatter, trend_formatter):
        self._data_formatter = data_formatter
        self._trend_formatter = trend_formatter

    def set_title(self, title):
        self._assert_graphics_is_initialized()
        self._plot.set_title(title)
        return self

    def set_xlabel(self, label):
        self._assert_graphics_is_initialized()
        self._plot.set_xlabel(label)
        return self

    def set_ylabel(self, label):
        self._assert_graphics_is_initialized()
        self._plot.set_ylabel(label)
        return self

    def add_data(self, x, y, **kwargs):
        self._data_formatter.format(x, y, self._plot, **kwargs)
        return self

    def add_trend(self, x, trend, **kwargs):
        self._trend_formatter.format(x, trend, self._plot, **kwargs)
        return self

    def as_raw_png_binary_string(self):
        output_string = BytesIO()
        self._canvas.print_png(
            output_string, dpi=self.__DEFAULT_RESOLUTION_IN_DPI__
        )
        return output_string
```

Fragmento de Código 4.10: Agrupación de datos de quimioterapia con respecto al área de actividad.

```
class ScatterDataFormatter(object):


    def format(self, x, y, plot, **kwargs):
        color = kwargs.get("c" or self._dot_color)
        return plot.scatter(x, y, c=color)



class HistogramDataFormatter(object):

    def format(self, x, y, plot, **kwargs):
        plot.grid(self._grid)
        return plot.hist(x, y, color=self._face_color)
```

Fragmento de Código 4.11: Objetos para la graficación de los datos.

```
class NoTrendFormatter(object):


    def format(self, x, y, plot, **kwargs):
        return None



class LineTrendFormatter(object):

    def format(self, x, y, plot, line_color='blue', **kwargs):
        return plot.plot(x, y, color=line_color, linewidth=self._line_width)
```

Fragmento de Código 4.12: Objetos para la graficación de los las líneas de tendencia.

Capítulo 5

Resultados y análisis

5.1. Base de datos integrada para análisis

Como se mencionó anteriormente, para la construcción del perfil génico, se utilizan datos de la *CCLE*. En su versión original, estos datos se encuentran organizados en archivos de texto separados por comas. Si bien este formato tiene utilidad práctica en cuanto a facilidad interpretación en diferentes contextos, rápidamente se determinó que no es apto para el tipo de consultas requeridas para este trabajo, en donde se requiere filtrar por diferentes campos (gen, línea celular) y procesar diversos subconjuntos de los datos. Con el fin de permitir los casos de uso proyectados para la consulta de datos, se normalizó en un esquema de base de datos relacional, montado en MySQL (ver 4.1). Esta base de datos comprende 5115 genes a través de 59 líneas de cáncer de mama. Este esquema de base de datos, si bien se ha limitado para este proyecto a incluir solamente líneas de cáncer de mama, está diseñado para permitir consultar datos de otros tipos de cáncer sin modificaciones adicionales, de acuerdo con las metas futuras del proyecto superior al cual éste se encuentra asociado. De todas maneras, se ha observado que el linaje celular es un buen predictor para la ortogonalidad de comportamiento de las células, por lo que se ha sugerido que los análisis por tipo de tejido específico son más acertados para modelos tumorales (Barretina et al. 2012).

La tabla de resumen de perfil genético permite capturar el comportamiento de una línea celular de acuerdo con cada uno de los genes evaluados. Por ende, es posible construir un perfil de cómo se comporta cada línea con respecto a las otras, lo cual es particularmente útil cuando es posible definir grupos naturales de los datos que permitan reducir la dimensionalidad.

Campo	Tipo	Descripción
cellLineName	varchar(100)	Nombre de la línea celular
cellLineId	int(11)	Identificador interno de la línea celular
gene	varchar(50)	Nombre del gen
geneId	int(11)	Identificador interno del gen
expression	decimal(8,6)	Nivel de expresión génica
copyNumber	decimal(7,5)	Número de copias
relativeGenomeLocation	int(11)	Localización relativa en el genoma
cellLineType	varchar(100)	Tipo de línea celular

Cuadro 5.1: Diccionario de datos de la tabla de perfil genético.

Campo	Tipo	Descripción
ccleName	varchar(100)	Nombre de la línea celular
ec50UM	decimal(10,8)	Dosis de respuesta al 50 % del efecto máximo
ic50UM	decimal(10,8)	Dosis de respuesta del 50 % del total
aMax	decimal(9,6)	Efecto máximo
actArea	decimal(6,4)	Área sobre la curva de actividad
fitType	enum	Tipo de ajuste de la curva de efecto
name	varchar(100)	Nombre del tratamiento
class	enum	Clase de tratamiento
mechanismOfAction	varchar(150)	Mecanismo de acción

Cuadro 5.2: Diccionario de datos de la tabla de respuesta a quimioterapias.

Por otra parte, la tabla de respuesta a quimioterapias permite representar ensayos clínicos sobre cultivos celulares, lo cual otorga información de granularidad gruesa sobre cómo fue afectada cada línea probada con los tratamientos que se le aplicaron. Es posible obtener datos de relación perfil genético - respuesta a quimioterapia si se utilizan las columnas `ccleName` y `cellLineName` como llaves para una operación de `JOIN`.

Como se puede notar, no existe granularidad suficiente para determinar, a partir de los datos, cómo afecta cada uno de los genes a la respuesta al agente químico. De ahí proviene la necesidad de modelar este efecto. Este hecho refleja, más que un problema en el diseño de la base de datos, una restricción inherente al problema que se desea resolver.

5.2. Visualización exploratoria y modelado inicial

Los análisis exploratorios se realizaron como paso inicial en el proyecto. Se calcularon índices de correlación de Spearman, como primer modelo de linealidad (de manera similar a lo realizado por Zhang et al. (2015)). Este primer modelo reveló distribuciones diversas de linealidad de los genes a través del conjunto de células de mama. Se observaron múltiples casos en que el coeficiente de linealidad era cercano a 0 (es decir, que el modelo lineal no es un buen ajuste) y algunos en que era cercano a 1 o -1 (al contrario, con un comportamiento cercano al lineal, y una pendiente cuyo signo corresponde con el signo del índice de correlación de Spearman) - ver Fig. 5.1. Adicionalmente, se verificó si existen casos en los que a un mayor número de copias hay menor expresión, o viceversa, mediante la graficación de los valore de número de copias y expresión con respecto a su media, para diversos genes (Fig. 5.2). Esta exploración indicó que este caso es común en los genes estudiados, por lo que sería importante modelar este tipo de casos (genes con comportamiento aberrante o no lineal) de una manera más integral.

5.3. Perfil genómico basado en localización relativa

Se calculó la localización relativa de cada uno de los genes con base en la posición de inicio de su código en el genoma. Este rankin permite ordenar todos los genes estudiados, con el fin de determinar si existe algún tipo de relación entre su localización en relación con otros genes y sus niveles de expresión o número de copias; se ha visto en estudios previos que afecta la transcripción y síntesis de proteínas (Redon et al. 2006). Cuando se calculó para todo el conjunto de líneas celulares (Fig. 5.3), no se encontró una tendencia discernible, tanto para la variación en el número de copias como el nivel de expresión. Luego, al realizar el mismo proceso para líneas celulares independientes, se notaron tendencias de similaridad en cuanto a número de copias, mas no para niveles de expresión (Fig. 5.4).

Si bien no se ha podido encontrar una razón clara de por qué se observa este comportamiento, es de interés para futuros estudios de la genómica del cáncer. Una posible explicación sería que ocurran eventos de copias de bloques ADN, de manera que el material duplicado exceda aquel definido para un gen. Existe la posibilidad, igualmente, de que las claras tendencias de número de copias constante se deba a algún artefacto durante la captura de datos.

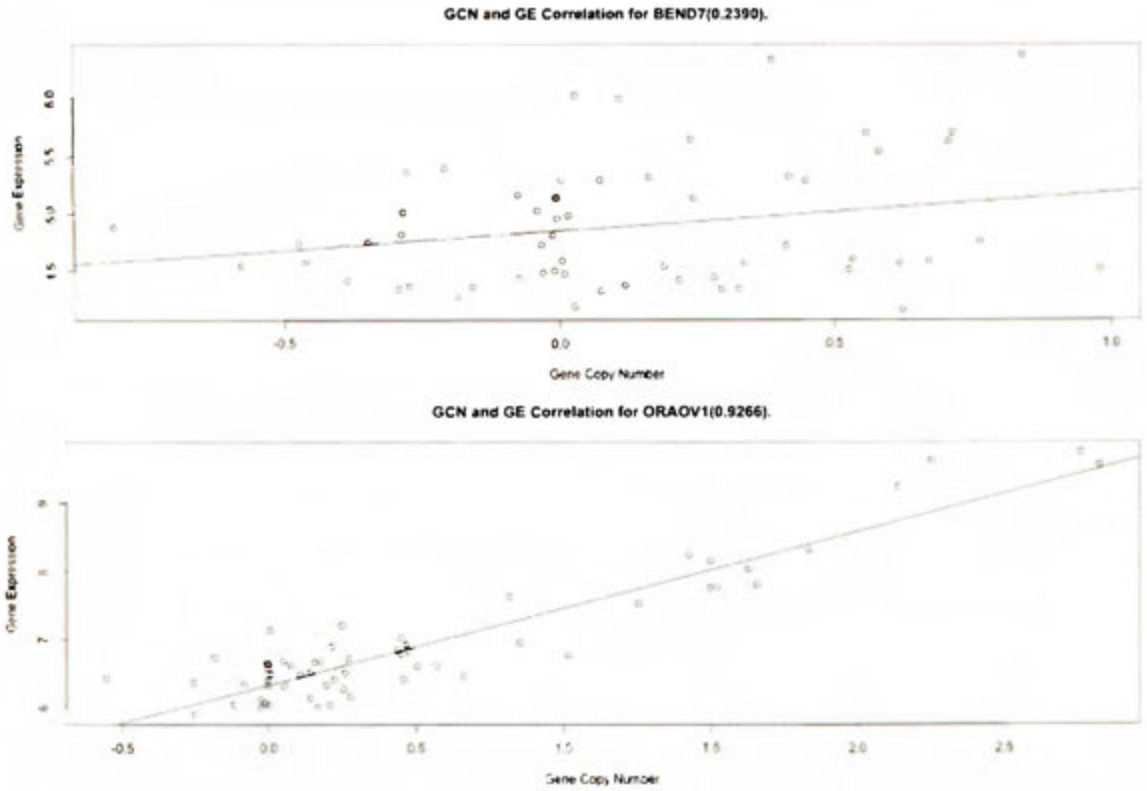

Figura 5.1: Índice de Correlación de Spearman para los genes *BEND7* y *ORAOV1*. Se consideró inicalmente que la correlación lineal podría ser un buen estimador del comportamiento celular para un gen. El gen *BEND7* (arriba, 0.2390, n=59) claramente demuestra que los datos no pueden ser bien descritos por medio del índice de Spearman. Si bien se nota una pendiente positiva, hay múltiples puntos y subconjuntos de datos que no se ajustan bien al modelo. El gen *ORAOV1*, por otra parte, es un buen ejemplo en donde el modelo lineal se ajusta bien a los datos, y en el que el índice de correlación de Spearman podría ser suficienemente descriptivo (0.9266, n=59). **Fuente:** Coto, Siles y Mora-Rodríguez (in press).

5.4. Agrupación no supervisada de expresión génica basada en modelos de combinación de Gaussianas

Se realizó agrupación de modelos de mezcla de Gaussianas sobre los datos de expresión génica, lo que reveló al menos dos poblaciones distintas de genes (Fig. 5.5). Esta observación coincide con estudios previos del comportamiento de expresión génica, en donde algunos genes codifican para vías comunes y otros para vías aberrantes dependientes del linaje celular, y sus niveles de expresión difieren

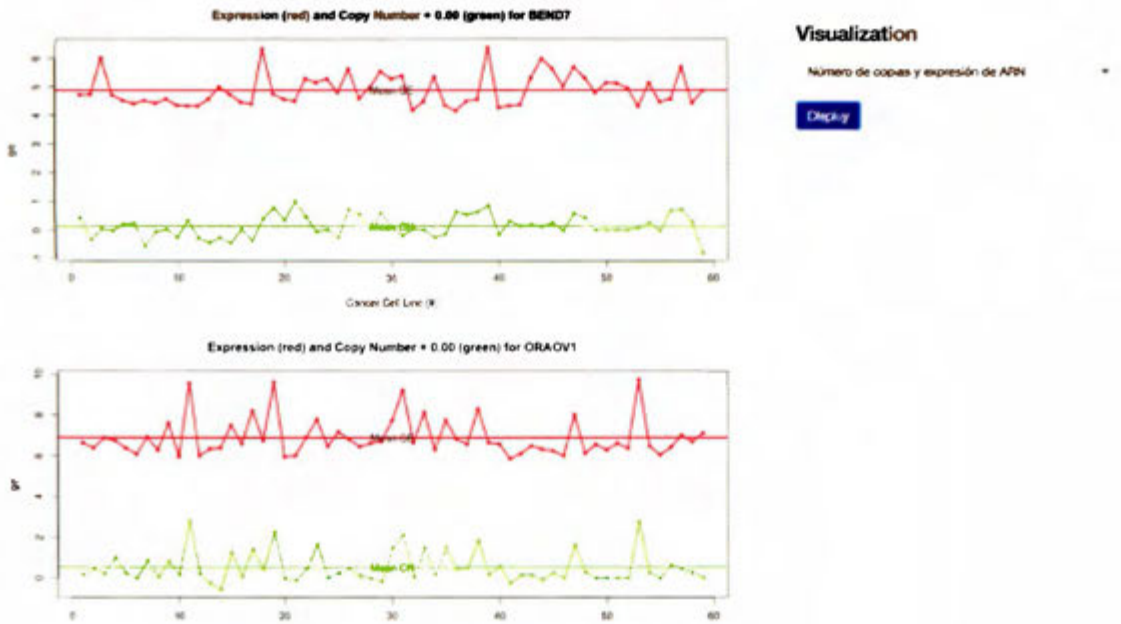

Figura 5.2: Relación entre incrementos y decrementos de número de copias y nivel de expresión con respecto a la media *BEND7* y *ORAOV1*. Se calculó la media aritmética del número de copias y nivel de expresión para ambos genes. Luego, se graficaron las magnitudes de los valores correspondientes para cada línea, de manera que fuera posible observar un incremento o decremento de alguna con respecto a la media. Se observaron comúnmente casos en los que un aumento en el número de copias no correspondió con un aumento en el nivel de expresión, lo cual podría señalar a algun mecanismo del cáncer que permita identificar posibles blancos terapéuticos relacionados con ese gen.

significativamente Solvang et al. (2011).

Esta observación es valiosa al indicar que los datos de la *CCLE*, de manera incipiente, pueden presentar subpoblaciones identificables por métodos de agrupación. Si bien el modelo Gaussiano no fue suficiente para realizar análisis más complejos, fue una interesante indicación de que los efuerzos de modelado podrían eventualmente lograr una capacidad predictiva.

5.5. Modelado de perfil génico cuadrático y lineal

Modelos lineales y cuadráticos se utilizaron para determinar si ciertos genes contaban con un comportamiento de expresión génica con respecto a número de copias lineal o no lineal. Como se menciona anteriormente, la linealidad en esta relación es importante, pues puede indicar procesos celulares abe-

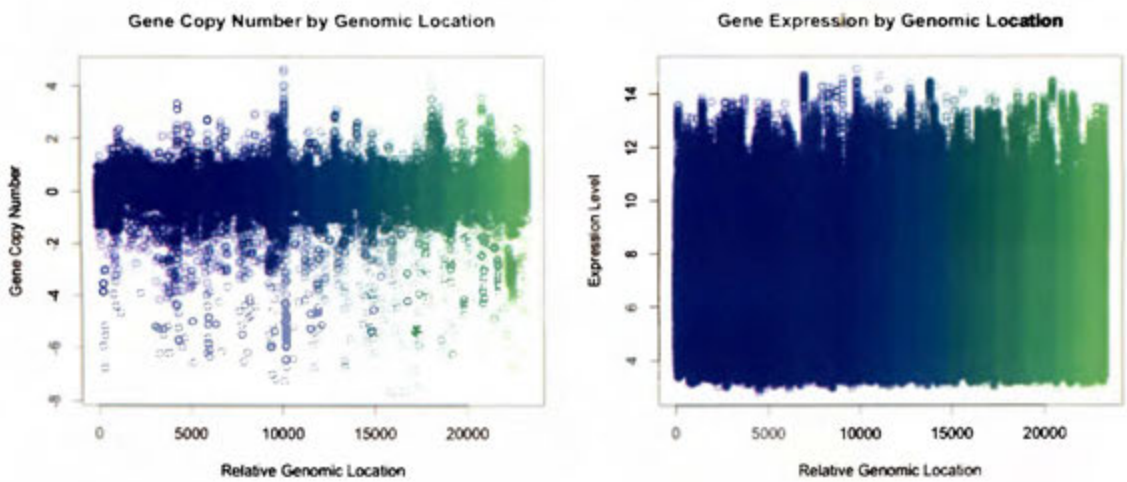

Figura 5.3: Número de copias y nivel de expresión génica ordenados por localización genómica relativa. Se ordenaron los genes de acuerdo con la localización en el genoma de donde fueron secuenciados. Con base en este orden, se graficaron número de copias (izquierda) y nivel de expresión (derecha). **Fuente:** Coto, Siles y Mora-Rodríguez (in press).

rrantes y por lo tanto funcionar como un criterio de distinción entre linajes sanos y cancerosos desde un punto de vista genómico (Solvang et al. 2011). Para determinar si los datos para un gen dado se comportan de manera más similar al modelo lineal o cuadrático, se calcularon medidas de bondad de ajuste para cada modelo (ver Fig. 5.6).

5.6. Agrupación no supervisada de expresión génica contra número de copias

Se realizaron experimentos de agrupación no supervisada entre los datos de número de copia y expresión génica. Se consideró como una alternativa que podría rendir buenos resultados como un método de reducción de dimensionalidad. Para este caso en particular, no se obtuvo resultados suficientemente promisorios para continuar con dicho método. La principal razón por lo cual se cree que este método no fue efectivo es por el relativamente bajo número de muestras por gen - 59líneas posiblemente no permita obtener grupos suficientemente representativos.

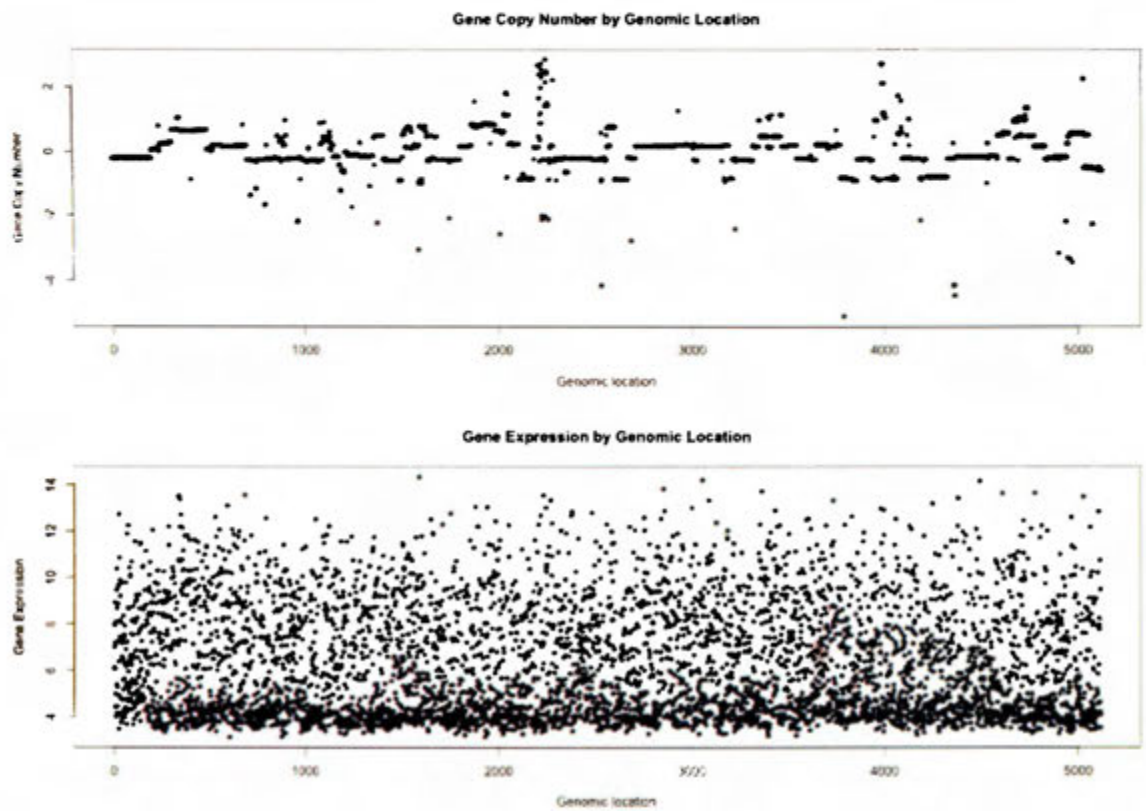

Figura 5.4: Número de copias y nivel de expresión génica ordenados por localización genómica relativa para la línea celular *AU565_BREAST*. Se ordenaron los genes de acuerdo con la localización en el genoma de donde fueron secuenciados. Con base en este orden, se graficaron número de copias (arriba) y nivel de expresión (abajo).
Fuente: Coto, Siles y Mora-Rodríguez (in press).

5.7. Agrupación no supervisada de coeficientes de los modelos cuadráticos

Finalmente, se realizaron modelos de agrupación sobre los coeficientes de los modelos cuadráticos mencionados en la sección 5.5. Esta agrupación, como el último modelo realizado hasta el momento, parece dar una perspectiva interesante. Al agrupar los coefientes de los modelos cuadráticos, con el coeficiente de x en el eje x y de x^2 en el eje y, es posible visualizar cuáles líneas tienen un comportamiento lineal y cuáles no - ya sea cuadrático u otro. Como se muestra en la Fig. 5.8, hay un grupo importante que incluye dos subconjuntos: aquellos cuyo comportamiento es relativamente constante (agrupados alrededor del punto `(0,0)`) y aquellos que tienen un comportamiento lineal (agrupados cerca del punto `(1,0)`). Podrían ser interesantes para estudios más avanzados de blancos terapéuticos

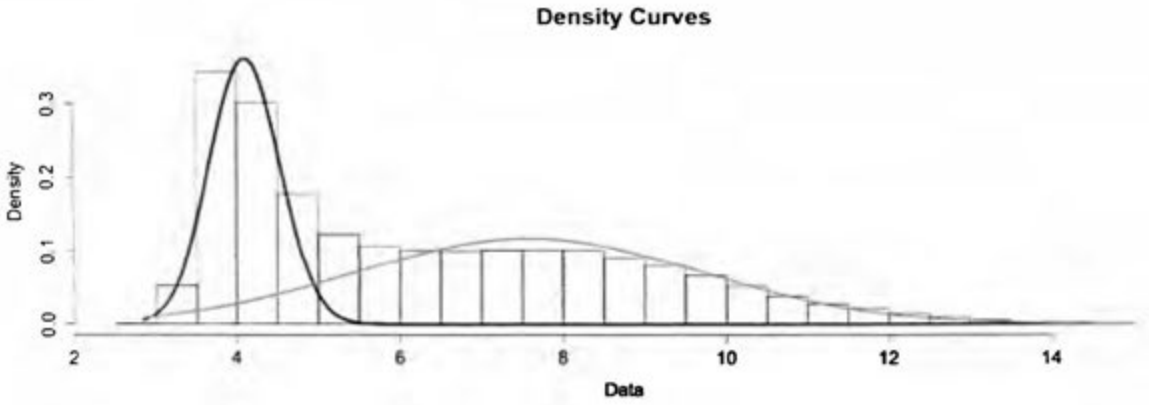

Figura 5.5: Modelos de mezcla de Gaussianas para niveles de expresión génica. Un procedimiento de modelado de mezclas de Gaussianas se ejecutó sobre el conjunto de datos de expresión génica. Este proceso encuentra la menor cantidad de subpoblaciones que minimice una medida de entropía o distancia de los datos con respecto al modelo. En este caso, un arreglo de dos subpoblaciones fue el que resultó más estable. Lo que indica este resultado es que existe una distinción significativa entre los niveles de expresión de genes que pertenecen a estos grupos. **Fuente:** Coto, Siles y Mora-Rodríguez (in press).

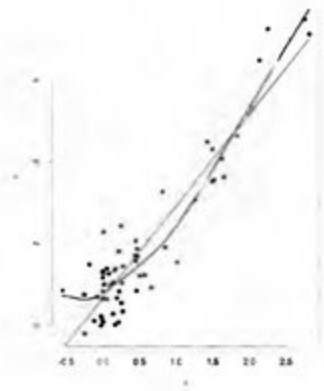

Figura 5.6: Modelos lineales, cuadráticos y de Splines para el gen ORAOV1. Se calcularon tres modelos para el gen *ORAOV1*: lineal (azul), cuadrático (verde) y Spline (rojo). **Fuente:** Coto, Siles y Mora-Rodríguez (in press).

todos aquellos genes que no se encuentren en el grupo lineal.

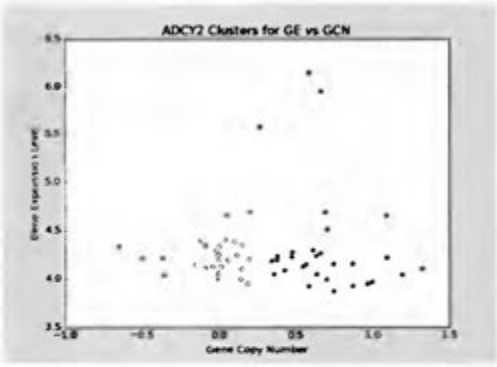

Figura 5.7: Agrupación no supervisada de la expresión génica como función del número de copias en el gen *ADCY2*. Mediante el algoritmo de *K-Means* se hallaron tres grupos en el perfil génico del gen *ADCY2*. Estos grupos pueden indicar comportamientos significativamente distintos en el subconjunto de las diferentes líneas celulares. **Fuente:** Coto, Siles y Mora-Rodríguez (in press).

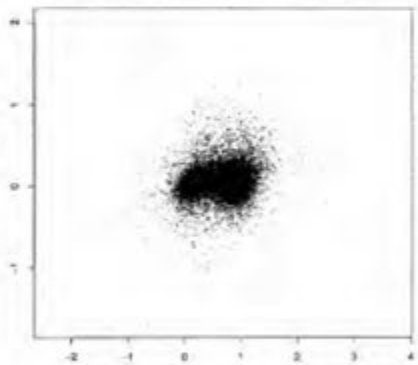

Figura 5.8: Agrupación no supervisada de los coeficientes de los modelos cuadráticos. Se identificaron tres grupos de interés. Particularmente los puntos en el grupo de mayor tamaño, y que no se encuentren el en grupo alrededor del punto `(1,0)` podrían ser interesantes como posibles blancos terapéuticos. **Fuente:** Esteban Zamora Alvarado, PRIS-Lab.

Capítulo 6

Conclusiones y Recomendaciones

6.1. Conclusiones

Como una plataforma principalmente de modelado, el enfoque de este proyecto ha sino la exploración y análisis de datos en un campo que constituye uno de los principales retos de la sociedad moderna.

El contar con un diseño de base de datos sobre el cual puedan construirse nuevos métodos de análisis es una ventaja para el futuro de proyectos relacionados con quimiosensibilidad. Esta infraestructura de datos ha sido aplicada en múltiples casos de uso relacionados con quimiosensibilidad y cáncer. Además, ha facilitado la comunicación del conocimiento generado durante el desarrollo del proyecto.

Las herramientas extensibles de visualización y análisis exploratorio han permitido la evaluación de hipótesis de manera eficiente. Cuando antes era necesario tomar varios días o semanas para construir, probar y presentar algún modelo, esta plataforma computacional permite construir modelos exploratorios en horas o incluso minutos. Basta con codificar el modelo en un lenguaje ampliamente utilizado en entornos científicos y de procesamiento de datos como lo es R, incluirlo a la herramienta, y es posible evaluar los resultados, generar gráficos y comunicarlos.

La definición de diversos modelos y su evaluación mediante métricas de bondad de ajuste ha sido un acercamiento novedoso, cuya aplicación fue efectiva en este proyecto. Al contar con la capacidad de ajustar diferentes modelos mediante métricas distintas, se ha logrado proponer un enfoque de análisis

novedoso. Debido a que es posible combinar diversos modelos como parte de un proceso de análisis, este tipo de modelado puede ser aplicado en diferentes tipos de problemas, además del cáncer.

El diseño en paralelo de la infraestructura de ejecución de modelos permite maximizar el aprovechamiento de los recursos de procesamiento disponibles para la plataforma. Principalmente mediante la utilización de múltiples procesos simultáneamente, se han obtenido dos beneficios clave. Primero, utilizar más eficientemente la infraestructura de procesamiento computacional. Y segundo, reducir significativamente el tiempo requerido para obtener resultados de modelos sobre los datos analizados.

Finalmente, en cuanto al desarrollo de esta plataforma, es importante destacar el trabajo transdisciplinario como un enfoque clave para enfrentar los retos complejos de la sociedad actual, como lo es el cáncer. La transdisciplinariedad, como se practicó en el PRIS-Lab durante el desarrollo de este proyecto y se continúa practicando, se basa en la colaboración de profesionales de distintas áreas que buscan aprender mutuamente sobre sus particulares especialidades, de manera que sea posible trascender sus límites para resolver problemas que se encuentran en los nexos de diversos dominios científicos. Debido a que el problema de reconocimiento de patrones en el cáncer involucra conceptos de computación, biología, medicina, microbiología y otros, fue indispensable para este proyecto

6.2. Recomendaciones

Para el futuro próximo, es necesario atacar primeramente el problema, todavía no resuelto, de la reducción de dimensionalidad del espacio en que se define la relación entre perfil génico y quimioterapia. Es decir: se requiere completar un proceso de selección de características que permita simplificar el modelo - idealmente a un conjunto de dimensiones considerablemente menor a las más de 5115 presentes actualmente. Se han realizado importantes esfuerzos en ese proceso: según el trabajo aquí expuesto, un acercamiento que parece ser prometedor radica en agrupar los genes de acuerdo con los valores de los coeficientes del modelo de regresión cuadrático.

Una vez definidos bajo un criterio formal los subconjuntos de ese modelo, deberá implementarse un algoritmo que permita relacionarlos a las líneas celulares. Por ejemplo, podría considerarse un algoritmo de reducción de dimensionalidad que discrimine aquellos genes que no se encuentran en el grupo de comportamiento no lineal mencionado en la sección 5.7. Dentro de los posibles candidatos se ha discutido el algoritmo t-SNE, normalmente usado para la reducción de la dimensionalidad en aplicaciones relacionadas con la visualización. Este algoritmo permite maximizar la similaridad entre subconjuntos de datos al formar grupos que minimicen la suma de las distancias de los puntos que los conforman (van der Maaten & Hinton, 2008). Según nuestras observaciones, este tipo de enfoque podría prestarse para encontrar grupos de genes relacionados entre sí y que puedan de alguna manera correlacionarse con la respuesta a quimioterapia.

Como recomendación final, sería útil incorporar conceptos relacionados al análisis de datos al diseño experimental de proyectos como la *CCLE*. Utilizar una escala de dosificación lineal, no logarítmica como se practica actualmente, por ejemplo, podría facilitar el modelado de la respuesta a las drogas. Además, sería de gran utilidad contar con valores de referencia normales o control, además de aquellos que han sido afectados por cáncer, de manera que sea posible comparar las observaciones entre sí, en vez de requerir fuentes externas de validación o validación cruzada.

Capítulo 7

Bibliografía

Almendro, V., Cheng, Y.-K., Randles, A., Itzkovitz, S., Marusyk, A., Ametller, E., ... Polyak, K. (2014). Inference of Tumor Evolution during Chemotherapy by Computational Modeling and In Situ Analysis of Genetic and Phenotypic Cellular Diversity. *Cell Reports*, *6*(3), 514-527. doi:10.1016/j.celrep.2013.12.041

Bannasch, D., Mehrle, A., Glatting, K.-H., Pepperkok, R., Poustka, A. & Wiemann, S. (2004). LIFEdb: a database for functional genomics experiments integrating information from external sources, and serving as a sample tracking system. *Nucleic Acids Research*, *32*(suppl_1), D505-D508. doi:10.1093/nar/gkh022

Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A. A., Kim, S., ... Garraway, L. A. (2012). The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, *483*(7391), 603-607. doi:10.1038/nature11003

Baslan, T., Kendall, J., Rodgers, L., Cox, H., Riggs, M., Stepansky, A., ... Hicks, J. (2012). Genome-wide copy number analysis of single cells. *Nature Protocols*, *7*(6), 1024-1041. doi:10.1038/nprot.2012.039

Chen, B.-J., Litvin, O., Ungar, L. & Pe'er, D. (2015). Context Sensitive Modeling of Cancer Drug Sensitivity. *PLOS ONE*, *10*(8), e0133850. doi:10.1371/journal.pone.0133850

Coto, J. C., Siles, F. & Mora-Rodríguez, R. (in press). Biocomputing Platform Module for Cancer Genomics and Chemotherapy. *IEEE CONCAPAN XXXVI.*

De Niz, C., Rahman, R., Zhao, X. & Pal, R. (2016). Algorithms for Drug Sensitivity Prediction. *Algorithms*, *9*(4), 77. doi:10.3390/a9040077

Devlin, B. (1996). *Data Warehouse: From Architecture to Implementation* (L. D. Cote, Ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Dong, Z., Zhang, N., Li, C., Wang, H., Fang, Y., Wang, J. & Zheng, X. (2015). Anticancer drug sensitivity prediction in cell lines from baseline gene expression through recursive feature selection. *BMC Cancer, 15*, 489. doi:10.1186/s12885-015-1492-6

Dorman, S. N., Baranova, K., Knoll, J. H., Urquhart, B. L., Mariani, G., Carcangiu, M. L. & Rogan, P. K. (2016). Genomic signatures for paclitaxel and gemcitabine resistance in breast cancer derived by machine learning. *Molecular Oncology, 10*(1), 85-100. doi:10.1016/j.molonc.2015.07.006

Flach, P. A. (2012). *Machine learning: the art and science of algorithms that make sense of data.* Cambridge ; New York: Cambridge University Press.

Gao, J., Aksoy, B. A., Dogrusoz, U., Dresdner, G., Gross, B., Sumer, S. O., ... Schultz, N. (2013). Integrative Analysis of Complex Cancer Genomics and Clinical Profiles Using the cBioPortal. *Science signaling, 6*(269), pl1. doi:10.1126/scisignal.2004088

Gupta, S., Chaudhary, K., Dhanda, S. K., Kumar, R., Kumar, S., Sehgal, M., ... Raghava, G. P. S. (2016). A Platform for Designing Genome-Based Personalized Immunotherapy or Vaccine against Cancer. *PLOS ONE, 11*(11), e0166372. doi:10.1371/journal.pone.0166372

Hajiloo, M., Damavandi, B., HooshSadat, M., Sangi, F., Mackey, J. R., Cass, C. E., ... Damaraju, S. (2013). Breast cancer prediction using genome wide single nucleotide polymorphism data. *BMC Bioinformatics, 14*(Suppl 13), S3. doi:10.1186/1471-2105-14-S13-S3

Hanahan, D. & Weinberg, R. A. (2011). Hallmarks of Cancer: The Next Generation. *Cell, 144*(5), 646-674. doi:10.1016/j.cell.2011.02.013

Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning.* Springer Series in Statistics. New York, NY: Springer New York. Recuperado el 29 de enero de 2016, desde http://link.springer.com/10.1007/978-0-387-84858-7

Kasprzyk, A. (2011). BioMart: driving a paradigm change in biological data management. *Database, 2011.* doi:10.1093/database/bar049

Keogh, E. & Mueen, A. (2011). Curse of Dimensionality. En C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 257-258). DOI: 10.1007/978-0-387-30164-8_192. Springer US. Recuperado el 22 de febrero de 2016, desde http://link.springer.com.ezproxy.sibdi.ucr.ac.cr:2048/referenceworkentry/10.1007/978-0-387-30164-8_192

Kopper Arguedas, G. & Vásquez Soto, C. a. (2015). *Plan Nacional de Ciencia, Tecnología e Innovación 2015-2021* (S. Núñez Corrales & P. Loría Herrera, Eds.). San José, Costa Rica: Unidad de Planificación Institucional, Ministerio de Ciencia, Tecnología y Telecomunicaciones de Costa Rica.

Liu, B., Kong, S., Gao, S., Zuliani, P. & Clarke, E. M. (2015). Towards Personalized Prostate Cancer Therapy Using Delta-reachability Analysis. En *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control* (pp. 227-232). HSCC '15. New York, NY, USA: ACM. doi:10.1145/2728606.2728634

Lodish, H., Berk, A., Zipursky, S. L., Matsudaira, P., Baltimore, D. & Darnell, J. (2000). *Molecular Cell Biology* (4th). W. H. Freeman.

Lyne, M., Smith, R. N., Lyne, R., Aleksic, J., Hu, F., Kalderimis, A., ... Micklem, G. (2013). metabolicMine: an integrated genomics, genetics and proteomics data warehouse for common metabolic disease research. *Database, 2013*. doi:10.1093/database/bat060

Marx, V. (2013). Biology: The big challenges of big data. *Nature, 498*(7453), 255-260. doi:10.1038/498255a

Nuzzo, A., Carapezza, G., Di Bella, S., Pulvirenti, A., Isacchi, A. & Bosotti, R. (2016). KAOS: a new automated computational method for the identification of overexpressed genes. *BMC Bioinformatics, 17*(12), 340. doi:10.1186/s12859-016-1188-1

Orozco, A., Morera, J., Jiménez, S. & Boza, R. (2013). A review of Bioinformatics training applied to research in Molecular Medicine, Agriculture and Biodiversity in Costa Rica and Central America. *Briefings in Bioinformatics, 14*(5), 661-670. doi:10.1093/bib/bbt033

Park, J., Park, B., Jung, K., Jang, S., Yu, K., Choi, J., ... Lee, Y.-H. (2008). CFGP: a web-based, comparative fungal genomics platform. *Nucleic Acids Research, 36*(suppl_1), D562-D571. doi:10.1093/nar/gkm758

Phan, J. H., Moffitt, R. A., Stokes, T. H., Liu, J., Young, A. N., Nie, S. & Wang, M. D. (2009). Convergence of biomarkers, bioinformatics and nanotechnology for individualized cancer treatment. *Trends in biotechnology, 27*(6), 350-358. doi:10.1016/j.tibtech.2009.02.010

Quo, C. F., Kaddi, C., Phan, J. H., Zollanvari, A., Xu, M., Wang, M. D. & Alterovitz, G. (2012). Reverse engineering biomolecular systems using -omic data: challenges, progress and opportunities. *Briefings in Bioinformatics, 13*(4), 430-445. doi:10.1093/bib/bbs026

Ramsey, S. D., Veenstra, D. L., Tunis, S. R., Garrison, L. P., Crowley, J. J. & Baker, L. H. (2011). How Comparative Effectiveness Research Can Help To Advance 'Personalized Medicine' In Cancer Treatment. *Health affairs (Project Hope)*, *30*(12), 2259-2268. doi:10.1377/hlthaff.2010.0637

Redon, R., Ishikawa, S., Fitch, K. R., Feuk, L., Perry, G. H., Andrews, T. D., ... Hurles, M. E. (2006). Global variation in copy number in the human genome. *Nature*, *444*(7118), 444-454. doi:10.1038/nature05329

Sinha, R., Schultz, N. & Sander, C. (2015). Comparing cancer cell lines and tumor samples by genomic profiles. *bioRxiv*, 028159. doi:10.1101/028159

Solvang, H. K., Lingjærde, O., Frigessi, A., Børresen-Dale, A.-L. & Kristensen, V. N. (2011). Linear and non-linear dependencies between copy number aberrations and mRNA expression reveal distinct molecular pathways in breast cancer. *BMC Bioinformatics*, *12*(1), 197. doi:10.1186/1471-2105-12-197

Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., ... Robinson, G. E. (2015). Big Data: Astronomical or Genomical? *PLoS Biol*, *13*(7), e1002195. doi:10.1371/journal.pbio.1002195

Stingele, S., Stoehr, G., Peplowska, K., Cox, J., Mann, M. & Storchova, Z. (2012). Global analysis of genome, transcriptome and proteome reveals the response to aneuploidy in human cells. *Molecular Systems Biology*, *8*. doi:10.1038/msb.2012.40

Teorey, T. J., Yang, D. & Fry, J. P. (1986). A Logical Design Methodology for Relational Databases Using the Extended Entity-relationship Model. *ACM Comput. Surv.* *18*(2), 197-222. doi:10.1145/7474.7475

Thompson, J. A., Duarte, C., Marks, P. & Congdon, C. B. (2014). An Automated Pipeline for Discovering Gene Expression Patterns Associated with Increased Cancer Survival Time. En *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 627-628). BCB '14. New York, NY, USA: ACM. doi:10.1145/2649387.2660807

van der Maaten, L. & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579-2605. Recuperado el 27 de febrero de 2017, desde http://jmlr.org/papers/v9/vandermaaten08a.html

Varma, S., Pommier, Y., Sunshine, M., Weinstein, J. N. & Reinhold, W. C. (2014). High Resolution Copy Number Variation Data in the NCI-60 Cancer Cell Lines from Whole Genome Microarrays Accessible through CellMiner. *PLOS ONE*, *9*(3), e92047. doi:10.1371/journal.pone.0092047

Vera, J., Lai, X., Schmitz, U. & Wolkenhauer, O. (2013). MicroRNA-regulated networks: the perfect storm for classical molecular biology, the ideal scenario for systems biology. *Advances in Experimental Medicine and Biology, 774*, 55-76. doi:10.1007/978-94-007-5590-1_4

Vougas, K. N., Jackson, T., Polyzos, A., Liontos, M., Johnson, E. O., Georgoulias, V., ... Gorgoulis, V. G. (2016). Deep Learning and Association Rule Mining for Predicting Drug Response in Cancer. *bioRxiv*, 070490. doi:10.1101/070490

Yuan, H., Paskov, I., Paskov, H., González, A. J. & Leslie, C. S. (2016). Multitask learning improves prediction of cancer drug sensitivity. *Scientific Reports, 6*. doi:10.1038/srep31619

Zhang, N., Wang, H., Fang, Y., Wang, J., Zheng, X. & Liu, X. S. (2015). Predicting Anticancer Drug Responses Using a Dual-Layer Integrated Cell Line-Drug Network Model. *PLoS computational biology, 11*(9), e1004498. doi:10.1371/journal.pcbi.1004498

Apéndices

Apéndice A

Artículo: Biocomputing Platform Module for Cancer Genomics and Chemotherapy

Biocomputing Platform Module for Cancer Genomics and Chemotherapy

J. C. Coto, F. Siles
Pattern Recognition and Intelligent Systems Laboratory
Department of Electrical Engineering,
University of Costa Rica (UCR)
San José, Costa Rica
Information and Communication Technology Research Center (CITIC)
School of Computer Science and Informatics,
University of Costa Rica (UCR)
San José, Costa Rica
Email: {juan.cotoulate, francisco.siles}@ucr.ac.cr

R. Mora-Rodríguez
LQT, Research Center on Tropical Diseases (CIET)
Faculty of Microbiology,
University of Costa Rica (UCR)
San José, Costa Rica
Email: rodrigo.morarodriguez@ucr.ac.cr

***Abstract*—In this paper, we present a biocomputational platform module for the representation and analysis of the response of cancer cell lines to diverse chemotherapeutic compounds based on their gene profile. This system, currently in active development, involves integration of gene profile data into a data repository, a broad array of visualizations based on a widely-used data analysis tool set and batched analysis of the entirety of gene profile and chemotherapeutic data using several regression and unsupervised clustering models. A description of the processing carried out by the system and results from exploratory data analysis, simple modeling and more complex clustering are presented.**

***Index Terms*—Biocomputing, Pattern Recognition, Data Processing, Learning Systems, Cancer, Genomics**

I. Introduction

Cancer is one of the most pressing health issues we face as a society. As of 2012, it accounted for approximately 8 million deaths worldwide, affecting over 14 million people. By 2008, an estimated 24.6 million people around the world were living with the disease, clearly making it a significant burden on universal health care [1], [2]. Particularly, in Costa Rica, a 31% increase in incidence was observed between 1995 and 2004, as well as 39.3% higher mortality rate 2007 in comparison to that observed in 1972. Currently, approximately 8000 cases are diagnosed yearly, leading to an estimated 3500 deaths; by 2025, it is expected to account for 50% of all deaths in the country, as opposed to 20% in 2007 [3]. Consequently, devising novel ways to challenge the problems presented by cancer is one of the principal goals of scientific and technological development today.

A close yet obscure relationship exists between modifications to the gene profile of cancer-affected cells and aberrations in their behavior. Therefore, the prospect of identifying patterns in cancer cell gene profiles which may allow estimation with a significant degree of certainty of their response to specific compounds is intensely interesting. Doing so would bring the possibility of crafting custom treatment programs based on the particular characteristics of each patient and tumor much closer to feasible fruition. If such a methodology were effective, it might result in a significant physical and economical cost reduction to patients and health care systems, as well as considerably improved outcomes [4].

One promising research strategy is modeling cancer cell behaviour through biocomputing. The United States National Institutes of Health (NIH) defines biocomputing, or computational biology, as "the development and application of data analytical and theoretical methods, mathematical models and simulation techniques for the study of biological, behavioral and social systems" [5]. This innovative approach seeks to understand the fundamental design principles of biological systems [6]. By applying biocomputational methods to previous, observed, gene profile and chemotherapy response data, it is possible to reverse engineer and describe specific mechanisms of cancer systemics [7], which has been done with success in several occasions, such as [8], [9] and [10].

In the current paper, we describe the cancer genomics and chemotherapy module of a biocomputational module currently being built as the foundation for a larger system directed towards eventual support of personalized therapy. Its core purpose is to provide efficient access to cancer cell related genomic and chemotherapy response data and provide a common framework on which mathematical models may be defined, trained and validated on said data. Populated in its foundational stages with data from the *Cancer Cell Line Encyclopedia* (*CCLE*) [8], this work seeks to empower researchers from areas such as Microbiology, Medicine, Electrical Engineering and Computer Science to represent and validate the feasibility of constructing *in silico* models of cancer cell line response to chemotherapeutic agents based on the relationship between gene copy number and expression level.

II. State of the Art

As the basis of the cellular machinery that is in charge of the majority of cell functions, genetic material is a fundamental component of cell behavior. Through the processes of transcription into RNA and subsequent translation into aminoacid polymers, DNA-coded genes are responsible for the majority of the functional units that carry out cell processes. Furthermore, these processes include the modification of cell behavior through adjustments in transcription, translation and protein synthesis, which introduces complex feedback mechanisms into the cellular system. These feedback mechanisms may be dependent on elements intrinsic or extrinsic to the cell and are therefore accountable for a significant subset of the dynamic behaviors typically observed in normally functioning biological systems. Because of this, genomic analysis may provide an illuminating peek at the underpinnings of cell behavior, allowing for the elucidation of the different mechanisms of which it is comprised [11].

Cancer, it is now known, encompasses transformational alterations to normal cellular mechanisms, particularly in processes related to critical functions, such as proliferation, growth and programmable cell death. These changes might be triggered by genetic, environmental or lifestyle pressures, or a combination thereof [4]. The principal aberration in cancer cell behavior lies in their ability to proliferate in an uncontrolled fashion, fundamentally altering tissue structure and eventually compromising its function. There are several ways in which this proliferation may occur, for example programmed cell death evasion, foreign tissue invasion (metastasis) and capillary vessel generation [12]. Although the complete mechanisms through which these alterations occur have not yet been described, irregularities in the DNA - protein synthesis pipeline have been shown to play a critical role in cancer processes. Healthy cells possess complex mechanisms of gene regulation, such as gene dosage compensation and bistability, which maintain homeostasis, effectively keeping cell behavior within expected parameters; however, this homeostatic equilibrium is disturbed during cancerous processes [13].

Certain gene products, which will eventually be translated into proteins, have been observed to be abnormally under- or over-expressed during cancer. Additionally, the number of gene copies in the cellular genome have been shown to vary from reduction or complete absence to several times over the normal and expected number of two: since chromosomes are arranged in pairs, every gene should have exactly two copies in the genome, one for each member of the pair [14]. Finally, a lack of correspondence between the number of copies of a certain gene and the magnitude of its representation in the cell transcriptome has been discovered, and it has been linked to changes related to cancerous processes [15]. Moreover, developmental changes in cancer cell lineage generally result in a wide array of cancer cell types within a single tumor, which in turn produce even greater variety in genomic alterations. Cell lines, cell cultures grown from single cancerous cells isolated from primary tumors which are immortalized and preserved [11], may be used to represent specific geno- and phenotypes of cancer tissue [8].

Taken in conjunction, these alterations result in a profound deviation in cancer cell gene profile from the norm, where instead of behaving as most healthy cells do, cancerous cells demonstrate a severe deregulation of the genetic code apparatuses. It is our belief that the nature of this deregulation may be accurately modeled through the identification of patterns of correspondence between gene copy number and gene expression. If true, then it should be possible to construct a model of gene copy number and expression relationships for each specific cancer cell type and associate it with that cell type's ability to tolerate treatment with specific chemotherapeutic agents. The ultimate purpose of the module here described is to evaluate this hypothesis via machine learning techniques [16], [17].

III. Methodology

A. Data Acquisition, Curation and Integration

Data for the initial stages of the module here described were taken from the *CCLE*. This corpus of data encompasses studies performed at the Broad Institute of MIT and Harvard universities (Massachusetts, U.S.A.) where extensive gene profiling and chemotherapeutic assays were performed. These data, available freely from the *CCLE*'s website (http://www.broadinstitute.org/ccle), are organized in plain text files in tabular format. Gene copy number data was organized by Entrez Gene identifier, common gene name and copy number per line. Expression data followed the GCT format, which is a tabular organization of microarray probe identifier, gene name, and expression per line. Data from chemotherapeutic agent assays was organized in a comma-separated-value table, with response data summarized per cell line per drug. In order to integrate data into a normalized database, all gene names had to be consolidated by gene identifier. Also, drug response data by dosage had to be de-normalized from a text representation of an array-like structure into several rows. Finally, database tables were designed to store data in a mostly denormalized fashion with a central fact table for simplified analysis, as depicted in Fig. 1.

B. Data Visualization and Simple Modeling

For visualization purposes, a Web application system was added to the architecture of the module. This visualization application is built with R. It provides an interface to the main data fact tables, making it trivial to access the integrated database for analysis. Furthermore, it provides functionality to extend the repertoire of visualization modules, with a

Figure 1: Logical Design of Integrated Relational Database. Database tables were designed and created to store gene profile and chemotherapy response data of breast cancer cell lines. The data to populate these tables was taken from the *CCLE*, available at http://www.broadinstitute.org/ccle.

low barrier of entry: all that is required is for a basic understanding of how the infrastructure of the application works, and enough knowledge of the R language to generate the new required visualizations. The primary objective for this module, however, is not data-intensive modeling over the entire data set but quick confection of visualizations, especially by subject matter experts who do not have much experience with programming. This allows for rapid prototyping and evaluation of intuitive hypotheses by the researchers, without requiring them to learn unnecessarily complex software.

C. Batch Processing and Per-Gene Modeling

Finally, the principal method of data analysis requires traversal of the entire data set, evaluating one or more models, which may be of different types and complexities. When dealing with a data set of a non trivial cardinality, it is not practical to require online processing. To process the data set on a gene-by-gene or line-by-line basis, a toolset was built in Python which accesses the integrated database and provides batch job capability. As is the case for the Web component of the module, one of the main design objectives for this module was to have a low barrier of entry, which would permit subject matter experts with minimal programming experience to utilize the system in order to test their hypotheses. Modeling capabilities include both supervised and unsupervised learning, graphics plotting and storage, as well as output to several data formats, such as JSON or CSV, which may substantially aid in the dissemination of results. Using this module, hybrid regression/clustering models are being developed to represent the relationship between copy number, gene expression and chemotherapy response.

IV. Results and Analysis

A. Integrated Analytical Database

Data from the *CCLE* were downloaded, analyzed and integrated in a relational database, hosted as a MySQL schema. This database describes gene copy number and expression levels for a subset of 5115 genes across 59 breast cancer cell lines. The data set was limited to include data only from breast cancer for evaluation of the viability of the module's approach; moreover, it is expected to eventually include data from other tissue sources. It should be noted, however, that cell lineage has been observed to be a strong predictor for line behavior orthogonality, which suggests that lineage-dependent analyses are better suited to obtaining accurate model representation of tumors based on cancer cell lines over lineage-independent ones, as seen by [8].

B. Exploratory Visualizations and Initial Modeling

As an initial step in the data processing pipeline, exploratory analyses were performed. Spearman Correlation Indexes were computed for gene expression versus copy number, as a first point of reference. This analysis step revealed a wide variety of data point distributions across all breast cancer cell lines. Cases where the linear correlation index between copy number and expression was high and low were observed (Fig. 2).

These modules were implemented in a composable manner, which allows for the rearrangement and extension of exploratory analyses in posterior experiments.

C. Genome Location-Dependent Gene Profile

Gene copy number and expression level were organized according to the relative location of genes on the genome, which has been shown to affect gene transcription and synthesis [18]. When computed over the entire set of cell lines Fig. 3, variation in gene copy number was found to be substantial, leading to no clear tendency. Regarding gene expression level, the variation was even greater. Based on this initial exploration, it was clear evaluation over the entire data set was not a promising alternative. When plotting the data of a single cell line as an exploratory assay, clear tendencies in gene copy number were observed, contrary to a less cohesive behavior for gene expression Fig. 4.

D. Gene Expression Unsupervised Clustering with a Gaussian Mixture Model

Gaussian Mixture Model clustering was performed on gene expression data, which revealed at least two distinct populations of genes (Fig. 5). This is in agreement with observed gene profile data, where genes which encode more common, canonical pathways have a similar pattern of

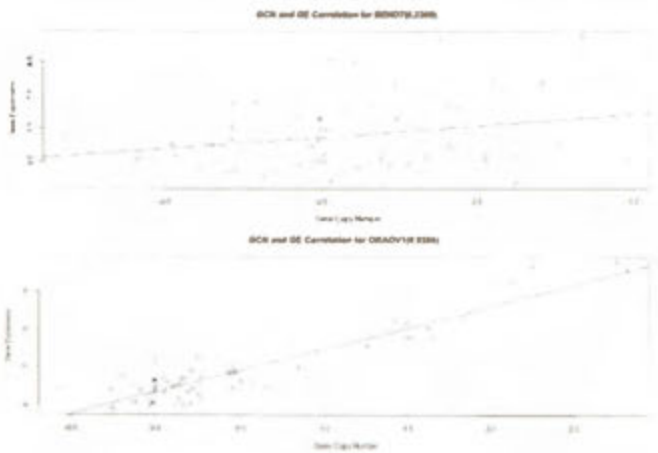

Figure 2: Spearman Correlation Index for the *BEND7* and *ORAOV1* Genes. Linear correlation was initially thought to be a good classification parameter for gene expression versus copy number. The *BEND7* gene (top, 0.2390, n=59) clearly demonstrates that the complexity of the arrangement of the data is not sufficiently described by a Spearman Correlation Index. While there is a definite positive slope to the linear model, there are multiple subsets of the data that do not fit the model well. *ORAOV1*, however, is a good example where the data fits a linear model, and when Spearman Correlation may be a sufficiently descriptive model (0.9266, n=59).

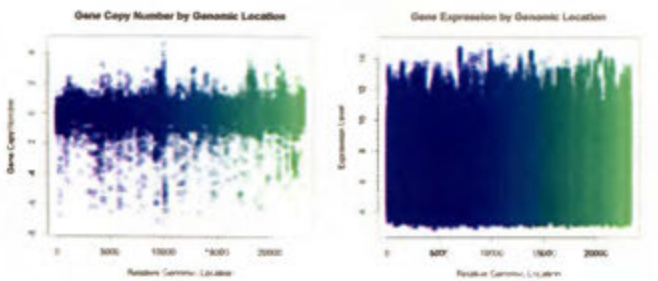

Figure 3: Gene Copy Number and Expression Sorted by Relative Genome Location. Genes were ordered according to the location in the genome from which they were sequenced. Based on this ordering, gene copy number (top) and expression level (bottom) were plotted.

expression as opposed to genes that encode for more lineage-dependent pathways - which might be a good indicator for aberrant behavior due to cancer [15].

E. Linear and Quadratic Gene Profile Modeling

Linear and quadratic models were used to determine whether specific genes followed a linear or non-linear gene expression versus gene copy number behavior. As mentioned above, linearity in this relationship is an important feature of cell behavior which allows for distinction between cell

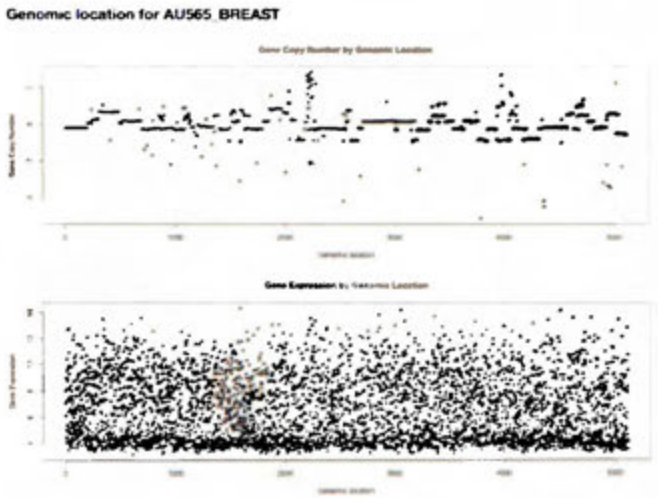

Figure 4: Gene Copy Number and Expression Sorted by Relative Genome Location for the *AU565_BREAST* Cell Line. Genes were ordered according to the chromosome location where they were sequenced. Based on this ordering, gene copy number (top) and expression level (bottom) were plotted.

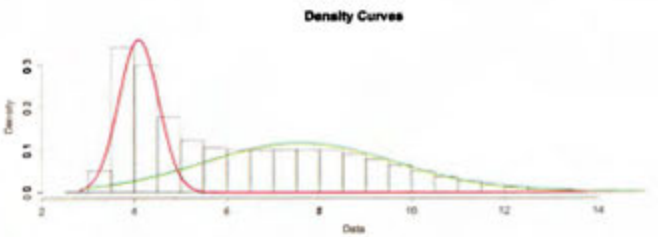

Figure 5: Gaussian Mixture Models for Gene Expression Levels. A Gaussian Mixture Model procedure was run over the gene expression data set. An arrangement of two main subpopulations was found to be the most stable. This indicates a clear distinction between the expression levels of both groups of genes.

lineages from a genetic standpoint [15]. In order to determine whether data for a given gene behaves either linearly or quadratically, coefficients of determination were evaluated for every gene and model (Fig. 6).

F. Unsupervised Clustering of Gene Expression Versus Copy Number

Unsupervised clustering was performed on every gene expression versus copy number data set. As a first step towards associating chemotherapy response with this relationship, clustering may improve the power of subsequent models by providing a richer set of significant features when compared to the raw data. This method of dimensionality reduction should prove to be highly valuable for either further classification or

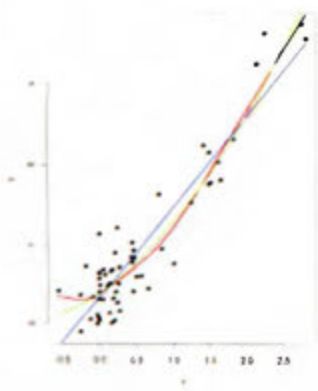

Figure 6: Linear, Quadratic and Spline Models for the ORAOV1 Gene. Three models were computed for the *ORAOV1* gene: linear (blue), quadratic (green) and spline (red).

regression models of chemotherapy effectiveness.

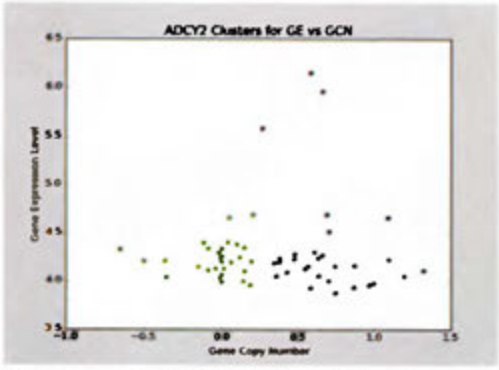

Figure 7: Unsupervised Clustering of Gene Expression versus Gene Copy Number for the *ADCY2* Gene. Using the *K-Means* algorithm, three clusters were found in the *ADCY2* gene profile. These clusters identify significantly distinct behaviors observed for this gene across all evaluated cancer cell lines.

This system aims to substantially aid in the evaluation of the hypothesis that the relationship between gene expression and copy number is related to cancer cell line response to chemotherapy. In order to do so, it requires data from multiple sources, generated from a variety of experiments. The *CCLE* proved to be an excellent data source for this end. Nevertheless, the challenge of curating and integrating data in order to construct more complex models was substantial.

In terms of the modeling of cancer behavior, it has proved to be an exceedingly complex task, with highly clustered data that is difficult to classify. As such, the strategies which we had initially planned were not sufficient to render a satisfactory model of the system. As it stands, major strides have been made towards associating chemotherapy response with gene profile. We have found, for instance, that there are distinct subpopulations of gene expression within our data set; we have also determined that linear/non-linear classification of gene expression and copy number functions is not powerful enough for the problem we intend to solve. Moreover, we have found that the location of a gene in the genome seems to have no significant bearing on its expression profile, while it does affect gene copy number to a certain extent. This effect, nevertheless, is not generalizable to all genes, which limits its usefulness to per-gene evaluation.

One of the most important features of our system is that, by design, it requires minimal, relatively easy to obtain, input data to provide a good estimate of therapy response. By requiring only gene copy number and expression data, contrary to models such as those in [8], [9] and [10], which are dependent on data such as oncogene variety and mutations. Our system has a very real possibility of being viable for clinical use once it has been completed and experimentally verified, at some point in the future. In the coming years, as gene sequencing becomes progressively less expensive, powerful models which require only sequencing data should be excellent tools for health care applications, especially in contexts with limited resources such as the Costa Rican public health system. We hope that our experience will pave the way for more research to be done both locally and abroad for the application of biocomputational tools to health issues.

V. Conclusions and Future Work

Transdisciplinary Work. Collaboration with experts in the field of cellular and cancer biology has been a cornerstone of the development of this module and proved to be absolutely invaluable in terms of providing overall project guidance and direction. In fact, one of the primary advantages of this system is that it has been designed to be accessible to experts with less software-technical training. During hypothesis evaluation, it became readily apparent that the capacity of having a system in which experiments could be run online during a meeting provided feedback which helped direct research in an extremely agile fashion.

Data-centric Experiments. Due to the intricacies of data generation, collection and processing, as well as the elevated cost of biological experiments, data availability and design is perhaps the most significant problem faced when modeling cancer biology systems. However, disparity between the ideal data configuration for processing and that generated by current standard biological experiments is a clear issue. For instance, chemotherapy survival rate assays would be much better suited for data analysis if linear dosage sampling (i.e. testing doses incremented linearly) were used instead of the standard, logarithmic scale serial dosage dilution. Linear dosage sampling would result in a more uniform distribution of data points along the dosage axis, instead of the closely

grouped points near the origin and much sparser distribution of points as the dosage increases that would be manifest on a logarithmic dosage scale. It is our belief that experiments designed with at least some notion of issues like this will allow for much more efficient data analysis during later stages.

What Comes Next. As this module matures, it is our intention to incorporate more types of models, which might perhaps lend themselves to better represent gene expression and copy number relationships. In any case, we believe having a richer battery of models to generate cancer cell line profiles will result in a more comprehensive analysis. We also intend, in the near future, to combine clustering and regression methods in order to bridge the gap between gene profile and chemotherapy response. This, without a doubt, is the most complex issue still to be resolved in this system. In fact, additional pattern recognition work will have to be carried out, since due to the number of genes that must be considered in such a model, its dimensionality is an issue that must be resolved, as mentioned previously and in [19] and [20].

ACKNOWLEDGEMENTS

This work is being partially funded through the Costa Rican Ministry of Science, Technology and Telecommunications (MICITT) and by the Vicepresidency of Research of the UCR, inscribed as *Biocomputational platform for the analysis of genomic data to overcome cancer and microbial infection resistance to therapy* under the CITIC. It's also part of the Translational Cancer and Biocomputing Research Network (ITCB), which includes researchers from several institutes from the UCR, including CITIC, PRIS-Lab, and CIET. Finally, the authors thank both the UCR's Computer Science and Informatics, and Electrical Engineering graduate programs for their support.

REFERENCES

[1] C. B.W. Stewart, "World cancer report 2014."

[2] B. P. S. D. D. J. B. K. Curado, M., "Burden of cancer in low- and middle-income countries." *cancer epidemiology: Low- and middle-income countries and special populations*, pp. 3–24, 2013.

[3] *Plan Nacional para la Prevención y Control del Cáncer 2011 - 2017*, 1st ed. San José, Costa Rica: Ministerio de Salud, 2012.

[4] M. Hajiloo, B. Damavandi, M. HooshSadat, F. Sangi, J. R. Mackey, C. E. Cass, R. Greiner, and S. Damaraju, "Breast cancer prediction using genome wide single nucleotide polymorphism data," *BMC Bioinformatics*, vol. 14, no. Suppl 13, p. S3, 2013. [Online]. Available: http://www.biomedcentral.com/1471-2105/14/S13/S3

[5] C. Wanjek, "Systems Biology as Defined by NIH," Nov. 2011. [Online]. Available: http://irp.nih.gov/catalyst/v19i6/systems-biology-as-defined-by-nih

[6] J. E. Ferrell, "Q&A: Systems biology," *Journal of Biology*, vol. 8, no. 1, p. 2, 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2656213/

[7] C. F. Quo, C. Kaddi, J. H. Phan, A. Zollanvari, M. Xu, M. D. Wang, and G. Alterovitz, "Reverse engineering biomolecular systems using -omic data: challenges, progress and opportunities," *Briefings in Bioinformatics*, vol. 13, no. 4, pp. 430–445, Jul. 2012. [Online]. Available: http://bib.oxfordjournals.org/cgi/doi/10.1093/bib/bbs026

[8] J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, S. Kim, C. J. Wilson, J. Lehár, G. V. Kryukov, D. Sonkin, A. Reddy, M. Liu, L. Murray, M. F. Berger, J. E. Monahan, P. Morais, J. Meltzer, A. Korejwa, J. Jané-Valbuena, F. A. Mapa, J. Thibault, E. Bric-Furlong, P. Raman, A. Shipway, I. H. Engels, J. Cheng, G. K. Yu, J. Yu, P. Aspesi, M. de Silva, K. Jagtap, M. D. Jones, L. Wang, C. Hatton, E. Palescandolo, S. Gupta, S. Mahan, C. Sougnez, R. C. Onofrio, T. Liefeld, L. MacConaill, W. Winckler, M. Reich, N. Li, J. P. Mesirov, S. B. Gabriel, G. Getz, K. Ardlie, V. Chan, V. E. Myer, B. L. Weber, J. Porter, M. Warmuth, P. Finan, J. L. Harris, M. Meyerson, T. R. Golub, M. P. Morrissey, W. R. Sellers, R. Schlegel, and L. A. Garraway, "The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity," *Nature*, vol. 483, no. 7391, pp. 603–607, Mar. 2012. [Online]. Available: http://www.nature.com/nature/journal/v483/n7391/full/nature11003.html

[9] Z. Dong, N. Zhang, C. Li, H. Wang, Y. Fang, J. Wang, and X. Zheng, "Anticancer drug sensitivity prediction in cell lines from baseline gene expression through recursive feature selection," *BMC Cancer*, vol. 15, no. 1, Dec. 2015. [Online]. Available: http://www.biomedcentral.com/1471-2407/15/489

[10] Y. Ma, Z. Ding, Y. Qian, Y.-W. Wan, K. Tosun, X. Shi, V. Castranova, E. J. Harner, and N. I. Guo, "An integrative genomic and proteomic approach to chemosensitivity prediction," *International journal of oncology*, vol. 34, no. 1, pp. 107–115, Jan. 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2667126/

[11] H. Lodish, A. Berk, S. L. Zipursky, P. Matsudaira, D. Baltimore, and J. Darnell, *Molecular Cell Biology*, 4th ed. W. H. Freeman, 2000.

[12] D. Hanahan and R. Weinberg, "Hallmarks of Cancer: The Next Generation," *Cell*, vol. 144, no. 5, pp. 646–674, Mar. 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0092867411001279

[13] J. Vera, X. Lai, U. Schmitz, and O. Wolkenhauer, "MicroRNA-regulated networks: the perfect storm for classical molecular biology, the ideal scenario for systems biology," *Advances in Experimental Medicine and Biology*, vol. 774, pp. 55–76, 2013.

[14] S. Stingele, G. Stoehr, K. Peplowska, J. Cox, M. Mann, and Z. Storchova, "Global analysis of genome, transcriptome and proteome reveals the response to aneuploidy in human cells," *Molecular Systems Biology*, vol. 8, Sep. 2012. [Online]. Available: http://msb.embopress.org/cgi/doi/10.1038/msb.2012.40

[15] H. K. Solvang, O. Lingjærde, A. Frigessi, A.-L. Børresen-Dale, and V. N. Kristensen, "Linear and non-linear dependencies between copy number aberrations and mRNA expression reveal distinct molecular pathways in breast cancer," *BMC Bioinformatics*, vol. 12, no. 1, p. 197, 2011. [Online]. Available: http://www.biomedcentral.com/1471-2105/12/197

[16] P. A. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge ; New York: Cambridge University Press, 2012.

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009. [Online]. Available: http://link.springer.com/10.1007/978-0-387-84858-7

[18] R. Redon, S. Ishikawa, K. R. Fitch, L. Feuk, G. H. Perry, T. D. Andrews, H. Fiegler, M. H. Shapero, A. R. Carson, W. Chen, E. K. Cho, S. Dallaire, J. L. Freeman, J. R. González, M. Gratacòs, J. Huang, D. Kalaitzopoulos, D. Komura, J. R. MacDonald, C. R. Marshall, R. Mei, L. Montgomery, K. Nishimura, K. Okamura, F. Shen, M. J. Somerville, J. Tchinda, A. Valsesia, C. Woodwark, F. Yang, J. Zhang, T. Zerjal, J. Zhang, L. Armengol, D. F. Conrad, X. Estivill, C. Tyler-Smith, N. P. Carter, H. Aburatani, C. Lee, K. W. Jones, S. W. Scherer, and M. E. Hurles, "Global variation in copy number in the human genome," *Nature*, vol. 444, no. 7118, pp. 444–454, Nov. 2006. [Online]. Available: http://www.nature.com/nature/journal/v444/n7118/full/nature05329.html

[19] E. Keogh and A. Mueen, "Curse of Dimensionality," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer US, 2011, pp. 257–258, dOI: 10.1007/978-0-387-30164-8_192. [Online]. Available: http://link.springer.com.ezproxy.sibdi.ucr.ac.cr:2048/referenceworkentry/10.1007/978-0-387-30164-8_192

[20] J. H. Phan, R. A. Moffitt, T. H. Stokes, J. Liu, A. N. Young, S. Nie, and M. D. Wang, "Convergence of biomarkers, bioinformatics and nanotechnology for individualized cancer treatment," *Trends in biotechnology*, vol. 27, no. 6, pp. 350–358, Jun. 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3779321/

Apéndice B

Código fuente

```
import sys
sys.path.append("../")
sys.path.append("../method")

from os.path import expanduser
import records
from sklearn.cluster import KMeans
from numpy import array
from method.helpers.plotting.bidimensional_graph import BidimensionalGraph
from method.helpers.plotting.data_formatters import ScatterDataFormatter
from method.helpers.plotting.trend_formatters import NoTrendFormatter


conn = "mysql://jc:280487@localhost:3306/ccle"

GENE_NAME = "ADCY2"


def get_data(database):
    return database.query(
        "select * from GeneticProfileMatView where symbol = :s",
        s=GENE_NAME
    )

def process(data):
    data = list(data)
    model = KMeans(n_clusters=3)

    points = list()
    for r in data:
        points.append(r["snpCopyNumber2Log2"])
        points.append(r["quantileNormalizedRMAExpression"])

    points = array(points).reshape(len(points) / 2, 2)

    fitted = model.fit(points)
    labels = model.labels_

    plot = BidimensionalGraph(ScatterDataFormatter(), NoTrendFormatter()) \
        .set_title("{g} Clusters for GE vs GCN".format(g=GENE_NAME)) \
        .set_xlabel("Gene Copy Number") \
        .set_ylabel("Gene Expression Level") \
        .add_data(points[:,0], points[:,1], c=labels)

    _write_plot_to_path("~/Desktop/Cluster.png", plot)

def _write_plot_to_path(path, plot):
    with open(expanduser(path), "wb") as f:
        png_data = plot.as_raw_png_binary_string()
        f.write(png_data.getvalue())
        png_data.close()


def main():
    db = records.Database(conn)
    data = get_data(db)
    result = process(data)

if __name__ == "__main__":
    main()
```

```
source("src/data/db.R")
source("src/modeling/models.R")
source("src/modeling/data.R")

RunModelsOnGeneData <- function(gene.symbol, goodness.of.fit.function, ...) {
    models <- list(...)
    number.of.models <- length(models)
    result <- rep.int(0, number.of.models)
    data <-
}
```

```
# Author and copyright: Juan Carlos Coto, 2015.

from matplotlib.backends.backend_agg import FigureCanvasAgg
from matplotlib.figure import Figure
from matplotlib.axes import Axes
from io import BytesIO

class BidimensionalGraph(object):
    # This class creates, populates and outputs bidimensional graphs.
    #
    # To use it:
    # 1. Create a new instance with a data formatter and a trend formatter.
    # 2. Set the title and/or axis labels.
    # 3. Add the data as x, y vectors (pretty much just iterables).
    # 4. Add a trendline as x, y vectors (will be used as (xi, yi) pairs from x and y to crea
    #      points for the line.
    # 5. Output as raw PNG (BytesIO), which you may save, print or whatever you like.
    # 6. (Optional) reset and go back to 2.

    __DEFAULT_RESOLUTION_IN_DPI__ = 150

    def __init__(self, data_formatter, trend_formatter):
        # Create a new plotter with the provided data and trend formatters.

        self._data_formatter = data_formatter
        self._trend_formatter = trend_formatter

        self._figure = None
        self._plot = None
        self._canvas = None
        self._initialize_graphics()

    def set_title(self, title):
        self._assert_graphics_is_initialized()
        self._plot.set_title(title)
        return self

    def set_xlabel(self, label):
        self._assert_graphics_is_initialized()
        self._plot.set_xlabel(label)
        return self

    def set_ylabel(self, label):
        self._assert_graphics_is_initialized()
        self._plot.set_ylabel(label)
        return self

    def add_data(self, x, y, **kwargs):
        self._data_formatter.format(x, y, self._plot, **kwargs)
        return self

    def add_trend(self, x, trend, **kwargs):
        self._trend_formatter.format(x, trend, self._plot, **kwargs)
        return self

    def as_raw_png_binary_string(self):
        output_string = BytesIO()
        self._canvas.print_png(output_string, \
                               dpi=self.__DEFAULT_RESOLUTION_IN_DPI__)
        return output_string

    def reset(self):
        self._initialize_graphics()
        return self
```

```
5
6       def _initialize_graphics(self):
7           self._figure = Figure()
8           self._plot = self._figure.add_subplot(1, 1, 1)
9           self._canvas = FigureCanvasAgg(self._figure)
0
1       def _assert_graphics_is_initialized(self):
2           assert isinstance(self._figure, Figure)
3           assert isinstance(self._plot, Axes)
4           assert isinstance(self._canvas, FigureCanvasAgg)
5
```

```python
from ccle.lib.pipeline import pipeline as pl
from ccle.lib.models import linear, quadratic
from ccle.lib.utils.vectors import vectorize


def main():
    return pl(_analyze, [linear, quadratic])


def _analyze(model_function, data):
    gene_name, data_tuples = data
    copy_number = vectorize([item[1] for item in data_tuples]).reshape(-1, 1)
    exp = vectorize([item[2] for item in data_tuples]).reshape(-1, 1)
    return model_function(copy_number, exp)
```

```r
# Correlation visualizations.  These correlate gene copy number with gene
# expression.

source("src/data/db.R")
library(shiny)

# Calculate all the correlations, segmenting the data by gene.
CorrelationsByGene <- function() {
  ApplyByGene(function(data, gene.label) {
    cor(data$quantileNormalizedRMAExpression, data$snpCopyNumber2Log2)
  })
}

# Get the correlation for a specific gene.
CorrelationForGene <- function(symbol) {
  ApplyForGene(symbol, function(data, gene.label) {
    cor(data$quantileNormalizedRMAExpression, data$snpCopyNumber2Log2)
  })[[symbol]]
}

# Generate a histogram of the correlation values between copy number and
# gene expression over all genes, over all cell lines.
CorrelationHistogram <- function() {
  correlations.by.gene <- CorrelationsByGene()
  list(
    list(
      graph.type = "hist",
      visualization = function() {
        hist(x = unlist(correlations.by.gene),
             xlab = "Correlation",
             ylab = "Frequency",
             main = "Copy Number, Expression Correlation")
      }
    )
  )
}

# Generate linear fit graphs for a selection of genes, based on the symbol.
CorrelationFits <- function() {
  selected.gene.symbols <- c("BEND7", "ORAOV1", "PLD5")
  lapply(selected.gene.symbols, function(gene.symbol) {
    force(gene.symbol)
    list(
      graph.type = "plot",
      visualization = function() {
        data <- GetDataByGeneSymbol(gene.symbol)
        x <- data$snpCopyNumber2Log2
        y <- data$quantileNormalizedRMAExpression
        corr <- CorrelationForGene(gene.symbol)
        print(corr)
        plot(x = x,
             y = y,
             xlab = "Gene Copy Number",
             ylab = "Gene Expression",
             main = paste0(
               "GCN and GE Correlation for ",
               gene.symbol,
               " (",
               format(round(corr, 4), nsmall = 4),
               ")."))
        model <- lm(y ~ x)
        abline(model)
      }
    )
```

```
  })
}

# Generate a dual plot of CopyNumber and GeneExpression data for selected
# genes, in order to analyze whether changes in the first are reflected in the
# second according to the Central Dogma of Biology.
SideBySideCNAndGE <- function() {
  selected.gene.symbols <- c("BEND7", "ORAOV1", "PLD5")
  lapply(selected.gene.symbols, function(gene.symbol) {
    force(gene.symbol)
    list(
      graph.type = "plot",
      visualization = function() {
        data <- GetDataByGeneSymbol(gene.symbol)
        cn <- data$snpCopyNumber2Log2
        ge <- data$quantileNormalizedRMAExpression
#         cn.normalization.value <- mean(ge) / 2
        cn.normalization.value <- 0
        normalized.cn <- cn + cn.normalization.value

        plot.title <- paste0(
          "Expression (red) and Copy Number + ",
          format(round(cn.normalization.value, 2), nsmall = 2),
          " (green) for ",
          gene.symbol)

        x <- 1:length(cn)
        ylim <- c(min(c(min(normalized.cn), min(ge))),
                  max(c(max(normalized.cn), max(ge))))

        plot(x = x,
             y = ge,
             col="red",
             xlab = "Cancer Cell Line (#)",
             type = "p",
             main = plot.title,
             ylim = ylim,
             pch = 20)
        lines(x = x, y = ge, col = "red", lwd = 2)
        abline(mean(ge), 0, col = "red", lwd = 2)
        text(x[ceiling(length(x) / 2)], mean(ge), label = "Mean GE" )
        lines(x = x, y = normalized.cn, col="green", lwd = 2)
        points(x = x, y = normalized.cn, col="green", pch = 20)
        abline(mean(normalized.cn), 0, col = "green", lwd = 2)
        text(x[ceiling(length(x) / 2)], mean(normalized.cn), label = "Mean CN" )
      }
    )
  })
}
```

```
from functools import partial
from os.path import expanduser

from pandas import read_csv

from ccle.data.profile import gene_data_as_profile


# API: data source

def get_iterator(initialized_funcs, limit=None, streamed=False):
    fun = _get_file_iterator
    return partial(fun, initialized_funcs, limit)


def _get_file_iterator(initialized_funcs, limit):
    read_data, _, fetch_gene_list, fetch_gene = initialized_funcs
    data = read_data()
    for gene in fetch_gene_list(data)[:limit]:
        gene_data = fetch_gene(gene)
        yield gene_data_as_profile(gene, gene_data)


def initialize(gene_col, line_col, copy_number_col, expression_col, path):
    return (
        partial(_read_data, path),
        partial(_fetch_complete_dataset, gene_col,
                line_col, copy_number_col, expression_col),
        partial(_fetch_gene_list, gene_col),
        partial(_fetch_gene, gene_col, line_col,
                copy_number_col, expression_col)
    )


def _read_data(path):
    full_path = expanduser(path)
    return read_csv(full_path)


def _fetch_complete_dataset(gc, lc, cc, ec, data_stream):
    raise NotImplementedError


def _fetch_gene_list(gene_col, data_stream):
    the_list = list(set(data_stream[gene_col].tolist()))
    return the_list


def _fetch_gene(gene_col, lc, cc, ec, data_stream, gene):
    data_grouped_by_gene = data_stream.groupby(gene_col).get_group(gene)
    filtered_data = data_grouped_by_gene[[cc, ec]]
    sorted_data = filtered_data.sort_values(by=cc)
    return [(row[lc], row[cc], row[ec]) for row in sorted_data]

```

```python
import csv
from numpy import mean, std
from method.model.pipeline import Pipeline

from ccle.data.data_source import DataSource
from ccle.data.data_set_generator import DataSetGenerator

class CVCalculator(object):

    def __init__(self):
        self._data_source = DataSource()
        self._data_set_generator = DataSetGenerator(self._data_source)

    def _train(self, model_function, training_data):
        copy_number, expression = training_data
        return (model_function(copy_number), model_function(expression))

    def _validate(self, model_function, validation_data, training_result):
        return None

    @staticmethod
    def _calculate_cv(x):
        return std(x) / mean(x)

    def run(self):
        pipeline = Pipeline(
            self._train,
            self._validate,
            self._data_set_generator,
            [self._calculate_cv]
        )
        return pipeline.run()

    def write_results_to_csv(self, results, path):
        with open(path, "w") as f:
            writer = csv.writer(f, delimiter=",")
            writer.writerow(["Gen", "CV Copy Number", "CV Expresion"])
            for dataset_result in results:
                _training, _validation = dataset_result.get_model_results(self._calculate_cv)
                cv_cn, cv_exp = _training
                writer.writerow([dataset_result.data_set.name, cv_cn, cv_exp])

```

```
import sys
sys.path.append("../")
sys.path.append("../method")

from ccle.cv_calculator import CVCalculator

def main():
    calculator = CVCalculator()
    print("Comenzando calculo de CV....")
    results = calculator.run()
    calculator.write_results_to_csv(results, "/Users/jc/Desktop/coeficiente_variacion.csv")
    # for result in results:
    #     print("For result %s:" % result.data_set.name)
    #     for model, model_result in result.items():
    #         print("\t %s: %s" % (model.__name__, model_result))
    #     print("\n")
    print("Finalizado.")

main()
```

```python
# All classes must implement the format(self, x, y, plot) method

class ScatterDataFormatter(object):
    def __init__(self, dot_color='black'):
        self._dot_color = dot_color

    def format(self, x, y, plot, **kwargs):
        # Assumes plot is a matplotlib plot object.

        color = kwargs.get("c" or self._dot_color)
        return plot.scatter(x, y, c=color)

class HistogramDataFormatter(object):
    def __init__(self, face_color='white', grid=True):
        self._face_color = face_color
        self._grid = grid

    def format(self, x, y, plot, **kwargs):
        # Assumes plot is a matplotlib plot object.

        plot.grid(self._grid)
        return plot.hist(x, y, color=self._face_color)
```

```
import ccle.data.csv_data_source as csv_ds
import ccle.data.sql_data_source as sql


def get_data_source(options):
    limit = options.LIMIT or None
    has_conn_string = options.get("SQL_SOURCE_URL") is not None
    data_source = \
        _get_sql_data_source(limit, options) if  has_conn_string \
        else _get_csv_data_source(limit, options)
    return data_source()


def _get_sql_data_source(limit, options):
    connection_string = options.SQL_SOURCE_URL
    i_funcs = sql.initialize(connection_string, options.TABLE_NAME,
                             options.GENE_COLUMN, options.CELL_LINE_COLUMN,
                             options.COPY_NUMBER_COLUMN, options.EXPRESSION_COLUMN)
    return sql.get_iterator(i_funcs, limit)


def _get_csv_data_source(limit, options):
    path = options.CSV_SOURCE_PATH
    i_funcs = csv_ds.initialize("gene", "cell_line_name",
                                "gene_copy_number", "gene_expression_level", path)
    return csv_ds.get_iterator(i_funcs, limit)
```

```
GetCopyNumberAndGeneExpressionAsXY <- function(data) {
    database.data <- data.frame(
        x=data$snpCopyNumber2Log2, y=data$quantileNormalizedRMAExpression
    )

    database.data <- database.data[with(database.data, order(x)), ]
    x <- database.data$x
    y <- database.data$y
    list(x=x, y=y)
}

GetDrugResponseData <- function(db.data) {
  result.data <- data.frame(
    line=db.data$ccleName, drug=db.data$name, actArea=db.data$actArea,
    aMax=db.data$aMax, ic50=db.data$ic50UM, ec50=db.data$ec50UM
  )
}
```

```r
# This is the main database connection file, which allows for data.frame access
# to the cancer cell line data.
library(RMySQL)

# Return a database connection with the default parameters.
GetConnection <- function() {
  dbConnect(MySQL(), user="jc", password="280487",
                  dbname="ccle", host="localhost")
}

# Close the provided connection.
CloseConnection <- function(connection) {
  dbDisconnect(connection)
}

# Return all the data in the Genetic Profile materialized view.
GetAll <- function() {
  conn <- GetConnection()
  result <- dbGetQuery(conn, "select * from GeneticProfileMatView;")
  CloseConnection(conn)
  result
}

# Select data for a specific gene symbol.  This is probably not needed as
# ApplyByGene allows for cursor iteration with a callback, and that"s usually
# what is needed.
GetDataByGeneSymbol <- function(symbol) {
  conn <- GetConnection()
  statement <- paste0("select * from GeneticProfileMatView where symbol = '", symbol, "';")
  result <- dbGetQuery(conn, statement)
  CloseConnection(conn)
  result
}

# Iterate, cursor-like, through the data by Gene symbol.  The function parameter
# must be of the signature (data.frame, group.label)
ApplyByGene <- function(fun) {
  # fun: function(data.frame, group.label, ...)
  conn <- GetConnection()
  result <- dbSendQuery(conn,
                        "select * from GeneticProfileMatView order by symbol;")
  output <- dbApply(result, INDEX = "symbol", fun)
  CloseConnection(conn)
  output
}

# Similarly to the function above, execute function for a gene.  However, this
# only executes the function on data with the matching gene symbol, instead of
# iterating through the data set.
ApplyForGene <- function(symbol, fun) {
  # fun: function(data.frame, group.label, ...)
  conn <- GetConnection()
  statement <- paste0(
    "select * from GeneticProfileMatView where symbol = '",
    symbol,
    "' order by symbol;"
    )
  result <- dbSendQuery(conn, statement)
  output <- dbApply(result, INDEX = "symbol", fun)
  CloseConnection(conn)
  output
}

GetAllDrugData <- function() {
```

```
    conn <- GetConnection()
    result <- dbGetQuery(conn,
                         "select * from DrugResponseMatView order by ccleName;")
    CloseConnection(conn)
    result
}
```

```python
from click import File, Path

DEFAULT_OPTIONS = {
    "limit": 1,
    "output_directory": "~/Workspace/ccle_data/results/parallel_test",
    "sql_source_url": "mysql://jc:280487@localhost:3306/hybrid_models_ccle",
}

UNSET = object()
```

```python
from copy import copy

def update(dict1, dict2):
    dict1.update(dict2)
    return dict1

def merge(dict1, dict2, exclude_values=None):
    if not exclude_values:
        exclude_values = set()
    exclude_values = set(exclude_values)
    dict1.update({k: v for k, v in dict2.items() if v not in exclude_values})
```

```python
from method.model.pipeline import Pipeline
from method.model.models import linear, quadratic
from method.helpers.plotting.bidimensional_graph import BidimensionalGraph
from method.helpers.plotting.data_formatters import ScatterDataFormatter
from method.helpers.plotting.trend_formatters import LineTrendFormatter
from ccle.drug_response.data import DataSource, DataSetGenerator


class DrugResponseByCellLineModeler(object):

    def __init__(self):
        self._data_source = DataSource()
        self._data_set_generator = DataSetGenerator(self._data_source)

    def _train(self, model_function, training_data):
        ccle_name, act_area = training_data
        return model_function(ccle_name, act_area)

    def _validate(self, model_function, validation_data, training_result):
        return None

    def run(self):
        pipeline = Pipeline(
            self._train,
            self._validate,
            self._data_set_generator,
            [linear, quadratic]
        )
        return pipeline.run()

    def plot_drug_response_graphs(self, results):
        for data_set_result in results:
            name = data_set_result.data_set.name
            obs_ccle_name, obs_act_area = data_set_result.data_set.training_data
            lin_model, _ = data_set_result.get_model_results(linear)
            quad_model, _ = data_set_result.get_model_results(quadratic)

            scatter_plot = BidimensionalGraph(ScatterDataFormatter(), LineTrendFormatter())
            self._write_plot_to_path("/Users/jc/Desktop/ddumps/{name}.png".format(name=name)
                scatter_plot.set_title("Drug Response Model for {name}".format(name=name)) \
                    .add_data(obs_ccle_name, obs_act_area) \
                    .add_trend(obs_ccle_name, lin_model) \
                    .add_trend(obs_ccle_name, quad_model, line_color="green")
            )

    @staticmethod
    def _write_plot_to_path(path, plot):
        with open(path, "wb") as f:
            png_data = plot.as_raw_png_binary_string()
            f.write(png_data.getvalue())
            png_data.close()
```

```r
# Compute mixed gaussian models for gene expression data.

library(mixtools)
source("src/data/db.R")

# Create a mixed gaussian model (i.e. find different gaussian distributions) for
# the gene expression profile of all genes.
ExpressionMixedGaussianModel <- function() {
  data <- GetAll()
  mix <- normalmixEM(data$quantileNormalizedRMAExpression)
  list(
    list(
      graph.type = "plot",
      visualization = function() {
        plot(mix, density=TRUE, cex.axis=1.4, cex.lab=1.4, cex.main=1.8, which=2,
             axes=FALSE, ann=FALSE)
        title(main="Subpoblaciones de expresion de ARN")
        box()
        axis(side=1)
        mtext(side = 1, "Expresion ARN", line = 2)
        axis(side=2)
        mtext(side = 2, "Densidad", line = 2)
      }
    )
  )
}

RunGaussianModel <- function() {
  data <- GetAll()
  mix <- normalmixEM(data$quantileNormalizedRMAExpression)
  plot(mix, density=TRUE, cex.axis=1.4, cex.lab=1.4, cex.main=1.8, which=2
       # ,xlab="Expresion ARN", ylab="Densidad", main="Subpoblaciones de expresion de ARN"
  )
  mix
}

```

```r
# Generate plots based on the genomic location of the genes, which might change
# the copy number and possibly expression values.
library(shiny)
source("src/data/db.R")

# Get a genetic profile dump ordered by genomic location (chromosome, start,
# end)
GetGenomicLocationOrderedGeneticProfile <- function() {
  data <- GetAll()

  # Replace X and Y chromosome entries with "23" so that conversion to numeric
  # places them at the end of the chromosome order
  data$chromosome[data$chromosome %in% c("X", "Y")] <- "23"
  data$numeric.chromosome <- as.numeric(data$chromosome)

  ordered <- with(data, order(numeric.chromosome,
                              chromosomeLocationStart,
                              chromosomeLocationEnd))
  data[ordered, ]
}

# Plot both the Copy Number and the Gene Expression of the provided data ordered
# by genomic location.
GetGenomicLocationPlots <- function(data, title = NULL) {
  x <- 1:nrow(data)
  result <- list(
    list(
      graph.type = "plot",
      visualization = function() {
        y <- data$snpCopyNumber2Log2
        ylim <- c(min(y), max(y))

        plot(x = x, y = y,
             xlab = "Localización genómica", ylab = "Número de copias de genes",
             main = "Número de copias de genes por Localización genómica",
             type = "p", ylim = ylim, pch = 20, col="green")
      }
    ),
    list(
      graph.type = "plot",
      visualization = function() {
        y <- data$quantileNormalizedRMAExpression
        ylim <- c(min(y), max(y))

        plot(x = x, y = y,
             xlab = "Localización genómica", ylab = "Expresión génica",
             main = "Expresión génica por Localización genómica",
             type = "p", ylim = ylim, pch = 20, col="red")
      }
    )
  )

  if (!is.null(title)) {
    result <- append(result, list(list(
      graph.type = "h3",
      visualization = function() {
        title
      })), 0)
  }
  result
}

# Plot all the data according to genomic location (this is a very messy
# visualization, but it may provide insight or really strange results)
```

```r
GenomicLocations <- function() {
  data <- GetGenomicLocationOrderedGeneticProfile()
  GetGenomicLocationPlots(data)
}

# Plot the genomic location ordered data for selected cell lines.
GenomicLocationsForSelectedCellLines <- function() {
  lines <- c("AU565_BREAST")#, "BT474_BREAST", "BT483_BREAST")
  data <- GetGenomicLocationOrderedGeneticProfile()

  # Lapply returns a list of list items, each of which is a plot definition;
  # it needs to be unlisted one level so that it is list of plot definitions,
  # as is expected by calling functions.
  # This is due to GetGenomicLocationPlots, called in GenomicLocationsByCellLine,
  # returning a list of plot definitions.
  all.plots <- lapply(lines, function(line) {
    force(line)
    GenomicLocationsByCellLine(line, data)
  })
  unlist(all.plots, recursive = FALSE)
}

# Get the genomic location ordered data for a specific cell line.
GenomicLocationsByCellLine <- function(line, data) {
  line.data <- data[with(data, ccleName == line), ]
  GetGenomicLocationPlots(line.data, title = paste("Localización genómica para ", line))
}
```

```python
# Measurements of goodness of fit.  These functions calculate a value that indicates a measur
# distance between vectors.
import numpy as np
from sklearn.metrics import r2_score as _r2_score


def residual_sum_of_squares(observed, predicted):
    # Calculate the difference of the squares of both vectors.

    _validate_arrays(observed, predicted)
    return np.sum(np.power((observed - predicted), 2))


def coefficient_of_determination(observed, predicted):
    # Calculate the coefficient of determination (a.k.a. R^2) between both arrays (observed a
    # predicted as usually called).

    _validate_arrays(observed, predicted)
    return _r2_score(observed, predicted)


def _validate_arrays(x, y):
    """
    Assert that input arrays are ndarrays and their dimensions are equal (to support subtract
    """
    _assert_array_is_ndarray(x)
    _assert_array_is_ndarray(y)
    # _assert_array_is_double_precision(x)
    # _assert_array_is_double_precision(y)
    _assert_array_shapes_are_equivalent(x, y)


def _assert_array_is_ndarray(a):
    """
    Assert if an array is of datatype ndarray (numpy n-dimensional array).
    """
    assert isinstance(a, np.ndarray), "The input array is not ndarray."


def _assert_array_shapes_are_equivalent(a, b):
    """
    Assert that the shapes of both provided arrays are equivalent.
    """
    assert a.shape == b.shape, "The shapes of the input arrays are not equivalent."


def _assert_array_is_double_precision(a):
    assert a.dtype == np.float64, "The input array's data type is different from float64."
```

```
# Fitness functions MUST adapt to the spec :: f(x, y) -> number, where x and y are numeric vec
# of equal length, and number is a single-element numeric vector.

# Compute the square difference between x and y.
Square.Difference <- function(x, y) {
    sum((x - y) ^ 2)
}

# Compute the coefficient of determination between x and y.
Coefficient.Of.Determination <- function(x, y) {
    m <- mean(x)
    sum((y - m) ^ 2) / sum((x - m) ^ 2)
}

# Return the goodness of fit for a given model, based on the real y values, according to a gi
# goodness of fit function.
GetModelGoodnessOfFit <- function(real.data.y, modelled.y, fitness.function) {
    fitness.function(real.data.y, modelled.y)
}
```

```python
from pathlib import Path
from pandas import read_sql_query, HDFStore
from sqlalchemy import create_engine

__GENE_QUERY = "SELECT * FROM GeneProfileMatView where cellLineType = 'breast';"
__DRUG_QUERY = "SELECT * FROM DrugResponseMatView;"
__OUT_FILE = "~/Workspace/ccle_data/ccle_breast_data.h5"

def _get_out_file_path():
    return str(Path(__OUT_FILE).expanduser())


def dump_from_database():

    conn_string = "mysql://jc:280487@localhost:3306/hybrid_models_ccle?unix_socket=/tmp/mysql
    db = create_engine(conn_string)
    out_file_path = _get_out_file_path()

    gene_data = read_sql_query(__GENE_QUERY, db)
    drug_data = read_sql_query(__DRUG_QUERY, db)

    gene_data.to_hdf(out_file_path, "gene_profile", complevel=9, complib='bzip2')
    drug_data.to_hdf(out_file_path, "drug_response", complevel=9, complib='bzip2')

    print("File output done.")


def read():
    out_file_path = _get_out_file_path()
    print("Opening and printing query result...")
    with HDFStore(out_file_path) as store:
        print(len(store.get("gene_profile")))
        print(len(store.get("drug_response")))
    print("Done.")


if __name__ == "__main__":
    read()
```

```python
# Copyright: Juan Carlos Coto, 2016.


import csv
import os
from os.path import expanduser
import numpy as np

from ccle.scripts.data_source import get_data_source


from ccle.lib.parallel_runner import parallel_runner
from ccle.lib.plotting.bidimensional_graph import BidimensionalGraph
from ccle.lib.plotting.data_formatters import HistogramDataFormatter
from ccle.lib.plotting.trend_formatters import NoTrendFormatter
from ccle.lib.goodness_of_fit import (
    residual_sum_of_squares, coefficient_of_determination
)
from ccle.lib.utils.vectors import vectorize
from ccle.pipelines.copy_number_expression import main as pipeline


def hybrid_models(options):
    runner = parallel_runner()
    modeling = pipeline()
    data_iterator = get_data_source(options)
    print("Comenzando calculo de modelos...")
    result = runner(modeling, data_iterator)
    print("Calculando bondades de ajuste...")
    goodness_of_fit = calculate_goodness_of_fit_for_results(result)
    print("Escribiendo resultados en csv...")
    write_goodness_of_fit_to_path(goodness_of_fit, options)
    print("Graficando resultados...")
    plot_goodness_of_fit_result_histograms(goodness_of_fit, options)
    print("Finalizado.")


# Calcular bondad de ajuste de los resultados
def calculate_goodness_of_fit_for_results(results):
    goodness_of_fit = list()
    for res in results:

        (data, [lin_result, quad_result]) = res
        gene_name, observations = data
        expression = vectorize(
            [item[2] for item in observations]).reshape(-1, 1).astype(np.float64)

        # Resultados de los modelos
        (lin_model, lin_params) = lin_result[1]
        (quad_model, quad_params) = quad_result[1]

        # Metricas de evaluacion
        lin_rss = residual_sum_of_squares(expression, lin_model)
        lin_r2 = coefficient_of_determination(expression, lin_model)
        quad_rss = residual_sum_of_squares(expression, quad_model)
        quad_r2 = coefficient_of_determination(expression, quad_model)

        goodness_of_fit.append((gene_name,
                                lin_rss,
                                lin_r2,
                                quad_rss,
                                quad_r2,
                                ",".join([str(x) for x in lin_params]),
                                ",".join([str(x) for x in quad_params])))
```

```
    return goodness_of_fit

# Escribir resultados de bondad de ajuste de los modelos en un csv
def write_goodness_of_fit_to_path(results, options):
    out_dir = expanduser(options.OUTPUT_DIRECTORY)
    os.makedirs(out_dir, exist_ok=True)

    with open("{}/goodness_of_fit.csv".format(out_dir), "w") as csv_file:
        writer = csv.writer(csv_file, delimiter=",")
        writer.writerow(["Gen",
                         "RSS lineal",
                         "R2 lineal",
                         "RSS Cuadratico",
                         "R2 Cuadratico",
                         "Coef_lin",
                         "Coef_cuad"])

        for gof_result in results:
            writer.writerow(gof_result)


def plot_goodness_of_fit_result_histograms(gof, options, number_of_bins=100):
    out_dir = expanduser(options.OUTPUT_DIRECTORY)
    os.makedirs("{}/plots".format(out_dir), exist_ok=True)

    linear_residuals = [result[2] for result in gof]
    quadratic_residuals = [result[4] for result in gof]
    linear_squares = [result[1] for result in gof]
    quadratic_squares = [result[3] for result in gof]

    histogram = BidimensionalGraph(
        HistogramDataFormatter(), NoTrendFormatter())

    write_plot_to_path("{}/plots/linear_residuals.png".format(out_dir),
                       histogram.set_title(
                           "Linear Copy Number and Expression R^2")
                       .add_data(linear_residuals, number_of_bins))

    # Note the following methods call histogram.reset() before setting properties, which cle
    # the histogram before updating its contents

    write_plot_to_path("{}/plots/linear_squares.png".format(out_dir),
                       histogram.reset().set_title("Linear Copy Number and Expression SSD")
                       .add_data(linear_squares, number_of_bins))

    write_plot_to_path("{}/plots/quadratic_residuals.png".format(out_dir),
                       histogram.reset().set_title("Quadratic Copy Number and Expression R^2
                       .add_data(quadratic_residuals, number_of_bins))

    write_plot_to_path("{}/plots/quadratic_squares.png".format(out_dir),
                       histogram.reset().set_title("Quadratic Copy Number and Expression SSD
                       .add_data(quadratic_squares, number_of_bins))


def write_plot_to_path(path, plot):
    with open(path, "wb") as f:
        png_data = plot.as_raw_png_binary_string()
        f.write(png_data.getvalue())
        png_data.close()
```

```
import collections

def flatten(l):
    for el in l:
        if isinstance(el, collections.Iterable) and not isinstance(el, (str, bytes)):
            yield from flatten(el)
        else:
            yield el
```

```
from attrdict import AttrDict
import click

from ccle.scripts.hybrid_models import hybrid_models
from ccle.scripts.defaults import DEFAULT_OPTIONS, UNSET

pass_config = click.make_pass_decorator(AttrDict, ensure=True)


@click.group()
@click.option("--output-directory", default=UNSET, #type=click.Path(), default=UNSET[click.Pa
              help="The path of the directory where results will be stored.")
@pass_config
def ccle(config, **kwargs):
    _merge_options(config, DEFAULT_OPTIONS)
    _merge_options(config, kwargs)


@ccle.command()
@click.option("--csv-source-path", default=UNSET, #type=click.File('r'), default=UNSET[click
              help="The path of a csv file to be used as gene profile source.")
@click.option("--sql-source-url", default=UNSET,#type=str, default=UNSET[str],
              help="<protocol>://<user>:<password>@server:port/<db>")
@click.option("--table-name", default="GeneProfileMatView",
              help="The name of the gene profile table to be used as source")
@click.option("--gene-column", default="gene",
              help="The name of the gene column on the source table")
@click.option("--cell-line-column", default="cellLineName",
              help="The name of the cell line name column on the source table")
@click.option("--copy-number-column", default="copyNumber",
              help="The name of the copy number column on the source table")
@click.option("--expression-column", default="expression",
              help="The name of the expression column on the source table")
@click.option("--limit", default=UNSET,
              help="The maximum number of data items to be retrieved from the source")
@pass_config
def copy_number_expression_models(config, **kwargs):
    _merge_options(config, kwargs)
    options = _get_options_dict(config)
    _echo_options_to_execute(options)
    hybrid_models(options)


def _merge_options(base, diff):
    set_in_diff = {k: v for k, v in diff.items() if v is not UNSET}
    base.update(set_in_diff)
    return base


def _get_options_dict(opts):
    _remove_unset_options(opts)
    return AttrDict({key.upper(): value for key, value in opts.items()})


def _remove_unset_options(opts):
    keys = [k for k, v in opts.items() if v is UNSET]
    for k in keys:
        del opts[k]


def _echo_options_to_execute(opts):
    click.echo("Using options:")
    for k, v in opts.items():
        click.echo("{}={}".format(k, v))
```

```
# Execute a model function over input data, and calculate the cost according to a given fitnes
# function.
RunModel <- function(x, y, model.function) {
    model.function(x, y)
}
```

```
source("app/src/data/db.R")
source("app/src/modeling/goodness_of_fit.R")
source("app/src/modeling/model_runner.R")
source("app/src/modeling/models.R")
source("app/src/modeling/data.R")

DEFAULT_LINE_WIDTH <- 2

# Plot a summary of the models that have been calculated for given x,y vectors
PlotSummary <- function(x, y, loess.fit, linear, quadratic) {
    the.plot <- plot(x, y, pch=16)
    lines(x, loess.fit, col="red", lwd=DEFAULT_LINE_WIDTH)
    lines(x, linear, col="blue", lwd=DEFAULT_LINE_WIDTH)
    lines(x, quadratic, col="green", lwd=DEFAULT_LINE_WIDTH)
    mtext("Resumen de modelos: Loess en rojo, lineal azul, cuadratico verde", 3)
}

# Print a summary of differences
PrintSummary <- function(loess.diff, loess.coef,
                         linear.diff, linear.coef,
                         quad.diff, quad.coef) {
    print("Medidas de ajuste:")
    print("Cuadrado de la diferencia [menor es mejor]:")
    print(paste("LOESS", loess.diff))
    print(paste("Lineal", linear.diff))
    print(paste("Cuadratico", quad.diff))
    print("**********")

    print("Coeficiente de determinacion (R^2) [0 - 1]:")
    print(paste("LOESS", loess.coef))
    print(paste("Lineal", linear.coef))
    print(paste("Cuadratico", quad.coef))
    print("**********")
}

# Run functions
RunModelsOnData <- function(data, title_prefix="") {
    formatted.data <- GetCopyNumberAndGeneExpressionAsXY(data)
    x <- formatted.data$x
    y <- formatted.data$y

    loess.model <- RunModel(x, y, GetLoessModel)
    linear.model <- RunModel(x, y, GetLinearModel)
    quad.model <- RunModel(x, y, GetQuadraticModel)

    loess.square <- GetModelGoodnessOfFit(y, loess.model, Square.Difference)
    loess.coef <- GetModelGoodnessOfFit(y, loess.model, Coefficient.Of.Determination)

    linear.square <- GetModelGoodnessOfFit(y, linear.model, Square.Difference)
    linear.coef <- GetModelGoodnessOfFit(y, linear.model, Coefficient.Of.Determination)

    quad.square <- GetModelGoodnessOfFit(y, quad.model, Square.Difference)
    quad.coef <- GetModelGoodnessOfFit(y, quad.model, Coefficient.Of.Determination)


    PlotSummary(x, y, loess.model, linear.model, quad.model)
    PrintSummary(loess.square, loess.coef,
                 linear.square, linear.coef,
                 quad.square, quad.coef)
}

RunModelsOnGene <- function(gene.symbol) {
    data <- GetDataByGeneSymbol(gene.symbol)
    RunModelsOnData(data, gene.symbol)
```

```
}

RunOnSpecificGene <- function() {
    RunModelsOnGene("ORAOV1")
}

RunOnAllData <- function() {
    data <- GetAll()
    RunModelsOnData(data)
}
```

```python
# Author and copyright: Juan Carlos Coto, 2015.  Usage under explicit instruction only.
# Basic model functions.  Should be applicable to any case where there is f(x) = y.
# As of September 2016, model functions are required to return some model object that
# can be used to retrieve model properties or parameters.

import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

from ccle.lib.utils.lists import flatten

def linear(x, y):
    regression = LinearRegression(fit_intercept=True)
    regression.fit(x, y)
    return (
        regression.predict(x),
        _get_linear_regression_model_description(regression)
    )

def quadratic(x, y):
    polinomial_x = PolynomialFeatures(degree=2).fit_transform(x)
    regression = LinearRegression(fit_intercept=True)
    regression.fit(polinomial_x, y)
    return (
        regression.predict(polinomial_x),
        _get_linear_regression_model_description(regression)
    )

def _get_linear_regression_model_description(model):
    return list(flatten(
        [model.intercept_.tolist(),
        model.coef_.tolist()]
    ))
```

```
GetLoessModel <- function(x, y) {
  loess(y ~ x, span=0.75)$fit
}

GetLinearModel <- function(x, y) {
  lm(y ~ x)$fitted.values
}

GetQuadraticModel <- function(x, y) {
  lm(y ~ poly(x, 2))$fitted.values
}
```

```python
from attrdict import AttrDict
from functools import partial
from multiprocessing import Pool, cpu_count


def parallel_runner(number_of_workers=cpu_count(),
                    max_tasks_per_worker=50):
    # Create a parallel running function with the
    # given number of workers and max tasks per worker.

    # Returns a function(pipeline, data_iterator)
    # that runs the given pipeline over each item of the
    # iterator. The items in the iterator MUST be independent
    # from each other, as results will be calculated
    # independently and finally aggregated in a list.

    process_pool = _create_process_pool(number_of_workers, max_tasks_per_worker)
    return partial(_run_in_parallel, process_pool)


def _create_process_pool(number_of_workers, max_tasks_per_worker):
    # Create a multiprocessing.Pool with the given
    # number of workers and max tasks per worker.

    # When a worker performs max_tasks_per_worker tasks,
    # it is killed and restarted (cleanup)

    return Pool(number_of_workers, maxtasksperchild=max_tasks_per_worker)


def _run_in_parallel(process_pool, pipeline, data_iterator):
    # Run a pipeline over a data set in parallel
    # with the provided process pool.

    # data_set MUST be an iterable, where each
    # of the items is provided to the pipeline.

    result = process_pool.map(pipeline, data_iterator)
    return result
```

```python
from attrdict import AttrDict
from functools import partial


def pipeline(runner, analysis_functions):
    # Create a modeling pipeline with the configured runners and data.
    #
    # runner: function(model_function, training_data)
    # analysis_functions: list(function(params)), where the params must be provided by the
    #                     _analyze function (i.e. that one converts from the data to
    #                     the correct input variables)

    assert runner is not None, "Must provide a training runner function"
    assert analysis_functions is not None and len(analysis_functions) >= 1, \
        "Must specify a list of analysis functions with at least one element"

    return partial(_pipeline, runner, analysis_functions)


def _pipeline(runner, analysis_functions, data):
    # (data, [(func, t_res),...])
    analysis_results = list()
    for analysis_function in analysis_functions:
        t_res = _analyze(runner, analysis_function, data)
        analysis_results.append((analysis_function, t_res))
    return data, analysis_results


def _analyze(runner, analysis_function, data):
    # Run training function for a model using training set data.

    return runner(analysis_function, data)
```

```
from attrdict import AttrDict


def gene_data_as_profile(gene_name, line_gc_ec_tuples):
    return (gene_name, list(line_gc_ec_tuples))
```

```
# Este es un script sencillo para instalar la aplicación (instala bibliotecas requeridas)

package.list <- c(
  'reshape2',
  'rgl',
  'RMySQL',
  'shiny',
  'shinyRGL'
  )
install.packages(package.list)

print('En un intérprete de R, asegúrese de que el directorio de trabajo esté apuntando al dire
print('Luego, ejecute `source("server.R")` seguido por `shiny::runApp()`.')
```

```r
# Main server script.  This script starts the shiny web server, configures the
# visualization options so that the appropriate rendering functions can be
# selected for each visualization and waits for user input.

library(shiny)
library(shinyRGL)
source("visualization_config.R")

# Start the main Web server loop, and render the UI components where the
# visualizations will be rendered.
shinyServer(function(input, output) {

  output$visualization.output <- renderUI({

    computed.visualizations <- ExecuteVisualization(
      input$visualization.select
    )

    lapply(computed.visualizations, function(visualization) {
      render.function <- SelectRenderingFunction(visualization$graph.type)
      render.function(visualization$visualization())
    })
  })
})

# Return a rendering function for each of the visualization functions, in order
# to render the correct content depending on the required visualization.
SelectRenderingFunction <- function(visualization.function) {
  switch(visualization.function,
    plot = renderPlot,
    hist = renderPlot,
    plot3d = renderWebGL,
    h3 = h3
  )
}

# Run the selected visualization function and return its results, so they can
# be rendered.
ExecuteVisualization <- function(visualization) {
  visualization.index <- as.integer(visualization)
  if (visualization.index > 1) {
    visualization.to.execute <- visualization.list[[visualization.index]]
    visualization.to.execute()
  }
}
```

```
from setuptools import setup, find_packages

setup(
    name="ccle",
    version="3.0",
    description="CCLE Machine Learning modeling program",
    license="GPL3",
    packages=find_packages(),
    install_requires=[
        "autopep8",
        "attrdict",
        "click",
        "matplotlib",
        "mysqlclient",
        "numpy",
        "pandas",
        "pep8",
        "scikit-learn",
        "scipy",
        "SQLAlchemy",
    ],
    entry_points={
        'console_scripts': [
            'ccle=main:ccle'
        ]
    },
)
```

```python
from functools import partial
from itertools import groupby
from sqlalchemy import create_engine, MetaData

from ccle.data.profile import gene_data_as_profile


# API: data source


def get_iterator(initialized_funcs, limit=None, streamed=False):
    """
    Return a partial function that, upon execution, iterates over the dataset.

    *WARNING*: the streamed parameter affects the method of data retrieval: when set to true,
    will be streamed from a database cursor; otherwise, it will be fetched as a whole block a
    iterated over using a generator.

    For cases where the data set is not MASSIVE, fetching the whole dataset and then iteratir
    it will probably work much faster. Use streaming only when fetching the whole thing is nc
    practical.
    """
    fun = _result_set_iterator if not streamed else _cursor_iterator
    return partial(fun, initialized_funcs, limit)


def _cursor_iterator(initialized_funcs, limit):
    _, fetch_all_genes, fetch_gene = initialized_funcs

    for gene in fetch_all_genes()[:limit]:
        gene_data = fetch_gene(gene)
        yield gene_data_as_profile(gene, gene_data)


def _result_set_iterator(initialized_funcs, limit):
    fetch_complete_dataset, _, _ = initialized_funcs

    for gene, gene_data in fetch_complete_dataset()[:limit]:
        yield gene_data_as_profile(gene, gene_data)


# API: sql data source

# For every function, gc is gene column, lc is line column, cc is copy number column and ec i
# expression column.


def initialize(conn_string, table_name, gene_col, line_col, cn_col, exp_col):
    """
    Get a tuple of initialized partial functions with the provided table information to use f
    querying.
    """
    metadata = _get_metadata(conn_string)
    table = metadata.tables[table_name]
    gc, lc, cc, ec = (table.c[col]
                      for col in (gene_col, line_col, cn_col, exp_col))
    return (
        partial(_fetch_complete_dataset, table, gc, lc, cc, ec),
        partial(_fetch_gene_names, table, gc),
        partial(_fetch_gene, table, gc, lc, cc, ec),
    )
```

```
def _get_metadata(conn_string):
    engine = create_engine(conn_string)
    metadata = MetaData(bind=engine)
    metadata.reflect()
    return metadata


# Internal

def _fetch_complete_dataset(table, gc, lc, cc, ec):
    result_set = table.select().order_by(gc).execute()
    all_rows = result_set.fetchall()
    grouped_rows = groupby(all_rows, key=lambda r: r[gc])
    return[
        (gene, _gene_result_set_to_dict(lc, cc, ec, gene_data))
        for (gene, gene_data) in grouped_rows
    ]


def _fetch_gene_names(table, gc):
    result_set = table.select(gc).order_by(gc).execute()
    return [row[0] for row in result_set.fetchall()]


def _fetch_gene(table, gc, lc, cc, ec, gene):
    result = table.select().where(gc == gene).order_by(cc).execute()
    gene_data = result.fetchall()
    return _gene_result_set_to_dict(lc, cc, ec, gene_data)


def _gene_result_set_to_dict(lc, cc, ec, result_set):
    return [(row[lc], row[cc], row[ec]) for row in result_set]
```

```
# All classes must implement the format(self, x, y, plot) method

class NoTrendFormatter(object):

    def __init__(self):
        pass

    def format(self, x, y, plot, **kwargs):
        return None

class LineTrendFormatter(object):

    def __init__(self, line_width=2):
        self._line_width = line_width

    def format(self, x, y, plot, line_color='blue', **kwargs):
        # Assumes plot is a matplotlib plot object.

        return plot.plot(x, y, color=line_color, linewidth=self._line_width)
```

```
# The main UI file.  This sets up the HTML structure where the content will be
# generated via the Shiny HTML-generating library.

library(shiny)
library(shinyRGL)
source("visualization_config.R")

# Generate a simple UI where there is a sidebar and a main content area.  The
# sidebar contains a select component that lets the user choose which
# visualization to display and a submission button to execute the selected
# choice. Note that select is dinamically populated with the result of calling
# `GetVisualizationFunctionChoices` in visualization_config.R
shinyUI(fluidPage(

  titlePanel("Visualizador de datos de cáncer"),

  sidebarLayout(
    sidebarPanel(
      selectInput("visualization.select",
                  label = h3("Visualization"),
                  choices = GetVisualizationFunctionChoices(),
                  selected = 1),
      submitButton("Display")
    ),

    mainPanel(
      uiOutput("visualization.output")
    ),
    position = "right"
  )
))
```

```
import numpy as np

def vectorize(iterable):
    return np.array(iterable)
```

```
# This file allows for configuring visualizations, basically matching names to
# functions which can display graphs on the front end.

source('src/analysis/correlation.R')
source('src/analysis/genomic_location.R')
source('src/analysis/expression_mixed_modeling.R')
#source('src/analysis/models.R')

# This list / hash table maps names to visualization functions and allows one of
# the visualizations to be selected via an index.
visualization.list <- list(
  '--' = '',
  'Histograma de correlaciones' = CorrelationHistogram,
  'Ejemplos de modelos lineales' = CorrelationFits,
  'Número de copias y expresión de ARN' = SideBySideCNAndGE,
  'Localización genómica (TODOS)' = GenomicLocations,
  'Localización genómica' = GenomicLocationsForSelectedCellLines,
  'Modelos mixtos de expresión' = ExpressionMixedGaussianModel
)

# Return a list of items contained in the visualization list above (note R
# lists work like hash tables, so it is basically an index for which
# visualization function to run based on a selection).
GetVisualizationFunctionChoices <- function() {
  vis.names <- names(visualization.list)
  result <- list()
  for (i in 1:length(vis.names)) {
    name <- vis.names[[i]]
    result[[name]] <- i
  }
  result
}
```