Análisis farmacogenómico de paneles celulares, drug screening

Joaquín Amat Rodrigo

Septiembre, 2018

Tabla de contenido

Versión PDF: Github

Introducción

Durante los últimos años, la investigación en los laboratorios dedicados al descubrimiento de nuevos fármacos, industria y academia, ha experimentado un cambio notable gracias al avance en las tecnologías de robótica y automatización video. Con los nuevos dispositivos, se ha pasado de generar unas pocas decenas de resultados semanales (low-medium throughput) a cientos o miles (high throughput), lo que ha hecho posible un amplio abanico de nuevas líneas de investigación. Uno de los campos en los que más ha cambiado el paradigma de investigación es en el de la oncología, donde los paneles de líneas celulares, combinados con la genómica, han permitido dar un paso sustancial hacia la medicina de precisión (personalizada).

La enfermedad del cáncer surge como resultado de la aparición y acumulación de alteraciones en el ADN (mutaciones) que, en última instancia, acaban confiriendo a las células la capacidad de crecer y multiplicarse de forma descontrolada, provocando daños al organismo. Las modificaciones del genoma pueden ser muy variadas (sustituciones de nucleótidos, delecciones, reordenamientos…), de ahí la amplia diversidad de tipos de cáncer, cada uno con unas características propias. Han sido muchos los esfuerzos de investigación llevados a cabo para tratar de identificar qué genes y qué mutaciones están causalmente implicados en la oncogénesis (oncogenes). El conocimiento obtenido a partir de estos estudios ha resultado clave para entender la biología del cáncer y encontrar nuevas terapias que logren combatirlo.

Una de las estrategias más extendidas en la industria farmacéutica para el desarrollo de fármacos antitumorales son las plataformas de *phenotypic screening* basadas en paneles celulares. En términos generales, el proceso seguido consiste en cuantificar la actividad antitumoral (muerte celular, antiproliferación) de una determinada molécula (potencial fármaco) en diferentes tipos de células cancerígenas extraídas de pacientes y cultivadas fuera del cuerpo humano (in vitro). Combinando los resultados de su actividad con información biológica de los tumores y de los pacientes de los que proceden (mutaciones, proteómica, histología, información clínica) se consigue entender cómo y por qué funciona la molécula (modo de acción) así como la identificación de biomarcadores, en concreto, estos últimos se han convertido en la piedra angular para conseguir fármacos de medicina de precisión.

El reto actual es ser capaces de encontrar patrones moleculares dentro de la gigantesca y compleja estructura que correlaciona la biología con el efecto de los fármacos. Aportar soluciones a este problema mediante técnicas de *Data Mining* es clave para acelerar el largo camino que conlleva crear nuevas medicinas.

Necesidad

Aunque la implementación y ejecución de los experimentos de paneles celulares está ampliamente validada y automatizada, los centros de investigación se encuentran ahora con limitaciones muy importantes a la hora de aplicar análisis que permitan la identificación de biomarcadores. Las principales razones son:

- Integración de grandes volúmenes de datos: una de las características de las disciplinas Ómicas es que trabajan con grandes volúmenes de datos, por ejemplo, el genoma de un solo paciente tiene aproximadamente 20,000 genes y su proteoma unos 111,451 transcritos. Integrar todos estos datos con los generados en el laboratorio sin comprometer su integridad requiere conocimiento sobre cómo extraer información de bases de datos.
- Complejidad del análisis: identificar de forma veraz relaciones de causalidad entre eventos requiere del uso de métodos estadísticos y de Data Mining avanzados que deben de ser adaptados de forma correcta en cada uno de los análisis. Este hecho se vuelve todavía más crítico si tiene en cuenta la extrema complejidad de los datos biológicos (miles de variables, información no relevante, comparaciones múltiples…).
- Visualización de datos multidimensionales: la complejidad intrínseca de los datos biológicos requiere de métodos de visualización específicos, generalmente interactivos, que no suelen estar disponibles en herramientas gráficas básicas.
- Software: las herramientas disponibles para los análisis bioinformáticos son muy variadas, existen herramientas comerciales muy optimizadas pero con un alto coste por licencia (*GeneData*, *Ingenuity-Pathway-Analysis QiaGen*), o herramientas de open source de acceso gratuito pero poco unificadas y cuya curva de aprendizaje suele ser más compleja.

Todos los conocimientos necesarios para solventar con éxito los problemas descritos no suelen formar parte de las habilidades de los investigadores del ámbito de la biomedicina (biólogos, biotecnólogos, médicos), lo que, en la práctica, suele conllevar que el análisis sea delegado a expertos en otras áreas (bioinformáticos y estadísticos en su mayoría). Esto supone un riesgo para el avance de los proyectos de ámbito biomédico. En primer lugar, el análisis de la información se convierte en un cuello de botella, los datos generados por muchos investigadores deben de ser analizados por unos pocos, lo que ralentiza la obtención de resultados. En segundo lugar, y más importante, si la trasferencia de conocimiento entre profesionales de las distintas áreas no es suficiente o el analista no tiene formación sobre el ámbito del que proceden los datos, difícilmente se podrá extraer la información adecuada.

El objetivo de este documento es hacer accesibles a través del lenguaje de programación **R** algunos los análisis necesarios para la identificación de biomarcadores:

- Carga de datos
- Integración con datos Ómicos (información genómica y transcriptómica)
- Contrastes de hipótesis
- Análisis de correlación
- Análisis de pathways
- Clustering de compuestos
- Visualizar los resultados.

Biomarcadores

Varias décadas de tratamientos contra el cáncer han dejado patente que existe una alta heterogeneidad en cuanto a la efectividad que tiene un determinado fármaco oncológico en la población de pacientes. Los avances en el estudio de la biología molecular del cáncer, han sacado a la luz claras evidencias de que la respuesta de un fármaco depende en gran medida de factores genómicos propios del paciente y del tumor.

Un biomarcador se define como una característica biológica que puede cuantificarse y emplearse como indicador de la presencia o ausencia de una patología, o de la respuesta que se tendrá frente a un determinado tratamiento. Algunos ejemplos incluyen: patrones de expresión genética, niveles de proteína en sangre, mutaciones, etc. Son, por lo tanto, una herramienta para caracterizar y segmentar la población de pacientes de cara a posibles tratamientos (Figura 1).

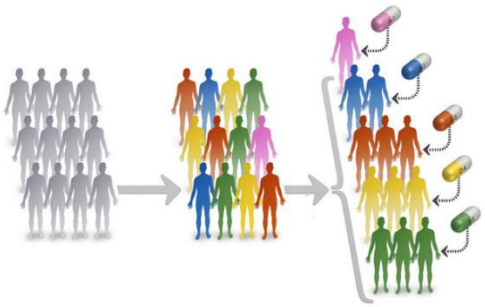


Fig.1 Esquema sobre la estratificación de los pacientes. Imagen obtenida de 2017 The University of Texas MD Anderson Cancer Center.

Dentro del ámbito de los fármacos de oncología, son muchas las ventajas que implica disponer de un biomarcador, algunas de las principales son: identificación certera de los pacientes que pueden beneficiarse de un tratamiento, reducción de efectos secundarios y aumento del éxito en las fases clínicas. Un claro ejemplo es el fármaco *Imatinib*, que consigue un porcentaje de

supervivencia en los 5 primeros años del 90% en aquellos enfermos de Leucemia Milode Crónica que tienen una aberración en los genes *BCR-ABL* (*Druker BJ, et al. Five-year follow-up of patients receiving imatinib for chronic myeloid leukemia*).

Aunque el anterior es un caso muy prometedor y que pone de manifiesto el potencial médico que tiene la identificación de biomarcadores asociados con enfermedades, son pocos los casos descubiertos. De hecho, la gran mayoría de tratamientos oncológicos capaces de combatir con notable eficacia el cáncer, no han sido asociados a una alteración biológica concreta que pueda ser empleada para seleccionar a aquellos pacientes en los que el tratamiento resulta efectivo. La combinación de la información generada en las diferentes disciplinas Ómicas (genómica, trascriptómica, proteómica, etc) junto con las metodologías de *Data Mining* se han convertido en la principal estrategia para la identificación de biomarcadores.

Cultivo celular

El término cultivo celular hace referencia al proceso por el cual células vivas se mantienen y se reproducen bajo condiciones controladas fuera de su ambiente natural. Dada la característica de las células cancerígenas para multiplicarse de forma descontrolada, es posible, a partir de unas pocas células extraídas en una biopsia médica, mantenerlas y expandirlas de forma ilimitada dentro de dispositivos diseñados para reproducir las condiciones del cuerpo humano (temperatura, humedad, nutrientes, gases…) (Figura 2).

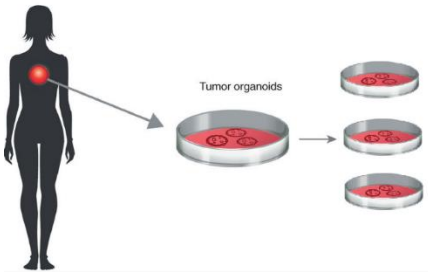


Fig.2. Esquema de la obtención de cultivos celulares a partir de biopsias de pacientes. Imagen obtenida de Personalized In Vitro and In Vivo Cancer Models to Guide Precision Medicine, 2017 American Association for Cancer Research.

Esta forma de proceder ha permitido crear bancos de células en los que se recogen multitud tipos de tumorales (pulmón, intestino, colon, etc), haciendo posible que los investigadores dispongan de material biológico con el que experimentar independientemente de los pacientes. Un ejemplo de ello es la American Type Culture Collection (ATCC), una organización sin ánimo de lucro que reúne, almacena y distribuye hasta 3000 líneas celulares de origen animal.

Actividad in vitro

Estudiar la actividad in vitro de un fármaco consiste en cuantificar su actividad a través de experimentos empíricos con el objetivo de estimar el efecto que tiene sobre un determinado sistema. Estos estudios se engloban dentro de las fases pre-clínicas del desarrollo y su éxito es clave para maximizar los resultados que consiguen los fármacos cuando llegan a los pacientes (fases clínicas).

En el ámbito de los fármacos antitumorales, uno de los estudios más frecuentes de actividad *in vitro* consiste en medir la relación que existe entre la dosis del fármaco y la respuesta celular que provoca, lo que se conoce como *cell-based drug rensponse*. Para ello, las celular tumorales se exponen a distintas concentraciones del fármaco en estudio y, tras un determinado tiempo de exposición, se cuenta el número de células vivas. A continuación, se normaliza el número de células vivas respecto a un control máximo (células no expuestas al fármaco, señal máxima) y un control mínimo (células expuestas a un fármaco de referencia que las mata, señal mínima) convirtiendo así las cuentas en porcentajes de actividad.

$$\%\ \text{actividad} = \frac{\text{señal} - \text{señal}_{\text{mínima}}}{\text{señal}_{\text{máxima}} - \text{señal}_{\text{mínima}}}$$

Finalmente, empleando los % de actividad, se ajusta una curva sigmoidea que representa la relación entre la dosis del fármaco y la respuesta conseguida.

$$\text{respuesta} = \text{límite inferior} + \frac{\text{límite superior} - \text{límite inferior}}{1 + (\frac{X}{IC50})^{p}}$$

donde los límites inferior y superior son las asíntotas de la curva, X la concentración del fármaco, p la pendiente de la curva e $IC50$ la concentración del fármaco con el que se consigue un 50% de la actividad máxima.

A partir de este modelo se pueden obtener múltiples métricas que describen la actividad del fármaco, algunas de las más empleadas son:

- *IC50*: concentración del fármaco con el que se consigue un 50% de la actividad máxima observada.
- *Emax*: actividad máxima observada, sea o no a la concentración máxima estudiada.
- *AUC*: área bajo la curva dosis-respuesta.

Curva Dosis Respuesta con R

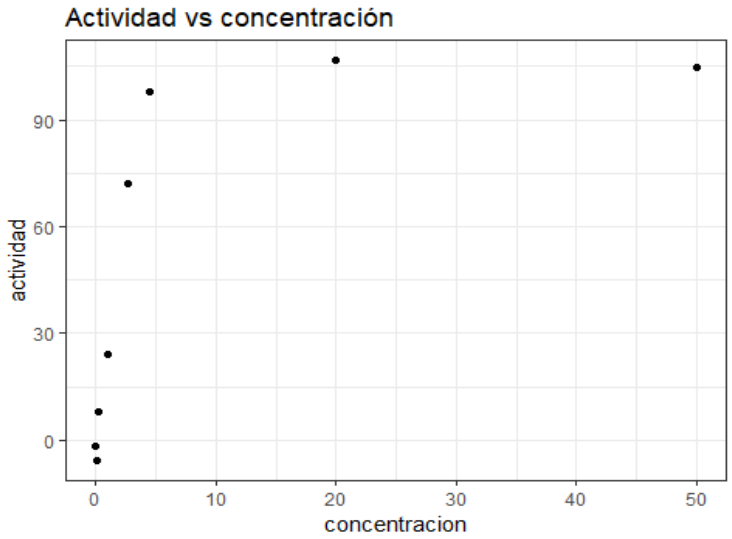
Desde el punto de vista matemático/estadístico, una curva dosis respuesta es un modelo de regresión en el que la variable independiente (predictor) es la concentración y la variable dependiente (variable respuesta) es la actividad o efecto del fármaco. Existen varios paquetes en `R` que permiten ajustar curvas dosis respuesta, uno de ellos en `drc`, cuyos autores, además de crear el paquete, han publicado un artículo donde se pueden encontrar información muy detallada sobre el análisis de curvas dosis respuesta. A continuación, se muestran algunos ejemplos.

Supóngase un experimento de laboratorio en el que se exponen células cancerígenas a distintas concentraciones de un fármaco y se registra la actividad antiproliferativa. Se asume que los valores de actividad han sido normalizados respecto a una referencia máxima y una mínima.

```
library(ggplot2)

concentracion <- c(0.0108, 0.0488, 0.2195, 0.9877, 2.643, 4.4444, 20.0, 50.0)
actividad     <- c(-1.6941, -5.6772, 8.2225, 24.2046, 72.145, 98.0494, 106.7267,
104.568)
datos_actividad <- data.frame(concentracion, actividad)
head(datos_actividad)
```

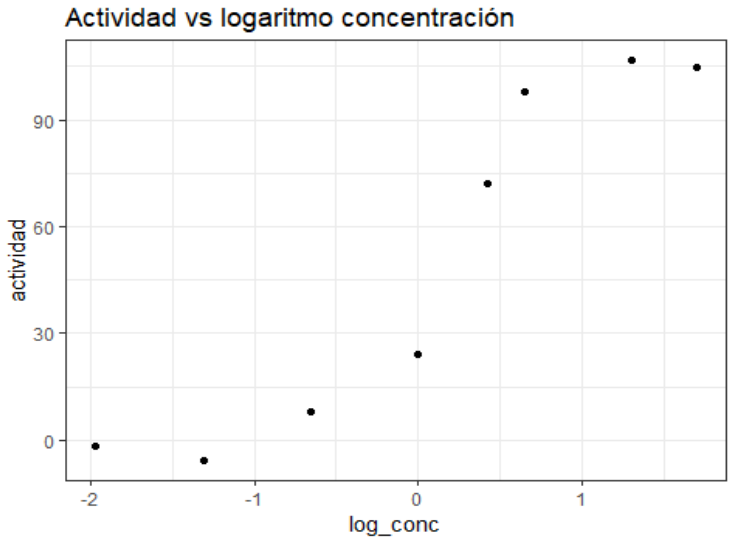
```
##   concentracion actividad
## 1        0.0108   -1.6941
## 2        0.0488   -5.6772
## 3        0.2195    8.2225
## 4        0.9877   24.2046
## 5        2.6430   72.1450
## 6        4.4444   98.0494
```

```
ggplot(data = datos_actividad, aes(x = concentracion, y = actividad)) +
  geom_point() +
  labs(title = "Actividad vs concentración") +
  theme_bw()
```



Con frecuencia, las concentraciones empleadas en los experimentos de *screening* son diluciones 1:10, por lo que la representación gráfica mejora si se emplea el logaritmo.

```
datos_actividad$log_conc <- log10(concentracion)
ggplot(data = datos_actividad, aes(x = log_conc, y = actividad)) +
  geom_point() +
  labs(title = "Actividad vs logaritmo concentración") +
  theme_bw()
```



Con la función `drm()` se obtiene el ajuste de la curva, entre sus argumentos destacan:

- `formula`: descripción de las variables que forman el modelo en forma de *'response ~ dose'*.
- `curveid`: vector numérico o factor que actúa como identificador para diferenciar entre varias curvas (en caso de que las haya).

- `data`: dataframe que contiene los datos.
- `fc`: tipo de función empleada para crear el modelo (curva). La función `drm()` permite generar curvas empleando varios tipos de modelos, LL.4 y LL.5 para regresión logística de 4 y 5 parámetros respectivamente, y W1.4 para Weibull. Cada una de estas funciones recibe como argumento una lista con el valor de los parámetros. Si se desea que el modelo encuentre el valor óptimo de los parámetros se les da el valor `NA`.
- `na.action`: función que trate los valores ausentes, por defecto se emplea `na.omit`.

```
library(drc)
# Ajuste de la curva dosis respuesta con un modelo logístico de 4 parámetros en
# el que no se fija ninguno de ellos.
curve_fit <- drm(formula = actividad ~ concentracion, data = datos_actividad,
                 na.action = na.omit,
                 fct = LL.4(fixed = c(NA,NA,NA,NA),
                            names = c("Hill","Bottom","Top","IC50")))
```

El `summary` del objeto devuelto por `drc` muestra el valor estimado de cada uno de los parámetros.

```
summary(curve_fit)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) (4 parms)
##
## Parameter estimates:
##
##                     Estimate Std. Error t-value   p-value
## Hill:(Intercept)    -2.12297    0.36954 -5.7449 0.0045501 **
## Bottom:(Intercept)  -0.21912    3.30744 -0.0662 0.9503582
## Top:(Intercept)    106.96927    3.82516 27.9646 9.728e-06 ***
## IC50:(Intercept)     1.76658    0.18704  9.4447 0.0007008 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  5.395822 (4 degrees of freedom)
```

Una vez obtenido el ajuste, se puede predecir la actividad esperada para nuevas concentraciones no probadas en el laboratorio, junto con su intervalo de confianza.

```
predict(object = curve_fit,
        newdata = data.frame(concentracion = 5),
        interval = "confidence")
```

```
## Prediction      Lower      Upper
##   96.36089   86.94579  105.77599
```

Para extraer el valor de la IC50 relativa (concentración a la que el compuesto alcanza el 50% de su actividad máxima), o absoluta (concentración a la que el fármaco consigue una actividad del 50%) se emplea la función `ED()`.

```
ED(object = curve_fit, respLev = 50, type = "relative")
```

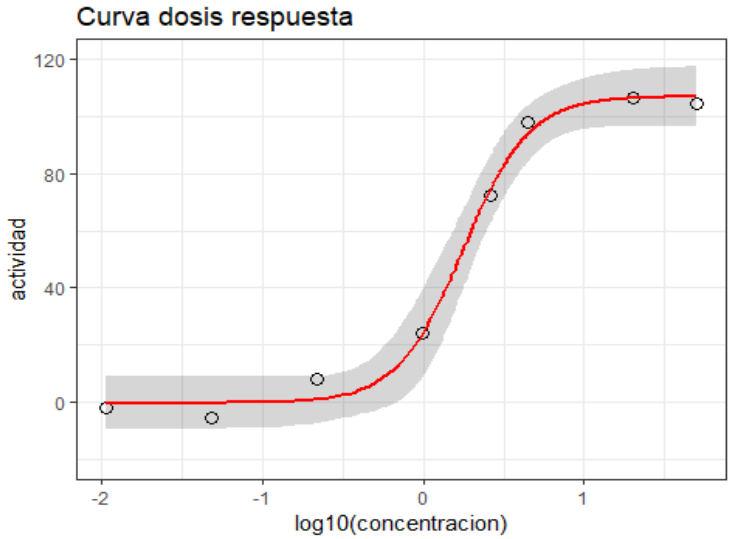
```
##
## Estimated effective doses
##
##        Estimate Std. Error
## e:1:50  1.76658    0.18704
```

Aunque se pude aplicar la función `plot()` a un objeto `drc`, suele ser preferible recurrir a la librería `ggplot2`.

```
# Se predice un grid de concentraciones entre la concentración mínima y máxima.
# Con estos valores se representa la curva del modelo. Cuantos más puntos se
# interpolen, mejor resolución tiene la curva.

grid_concentraciones <- seq(from = min(datos_actividad$concentracion),
                            to = max(datos_actividad$concentracion),
                            length.out = 1000)
grid_concentraciones <- data.frame(grid_concentraciones)

predicciones <- predict(object = curve_fit,
                        newdata = grid_concentraciones,
                        interval = "confidence")

# Se añade al dataframe predicciones el valor de las concentraciones.
predicciones <- cbind(grid_concentraciones, predicciones)
colnames(predicciones)[colnames(predicciones) == "grid_concentraciones"] <-

                                                    "concentracion"
head(predicciones)
```

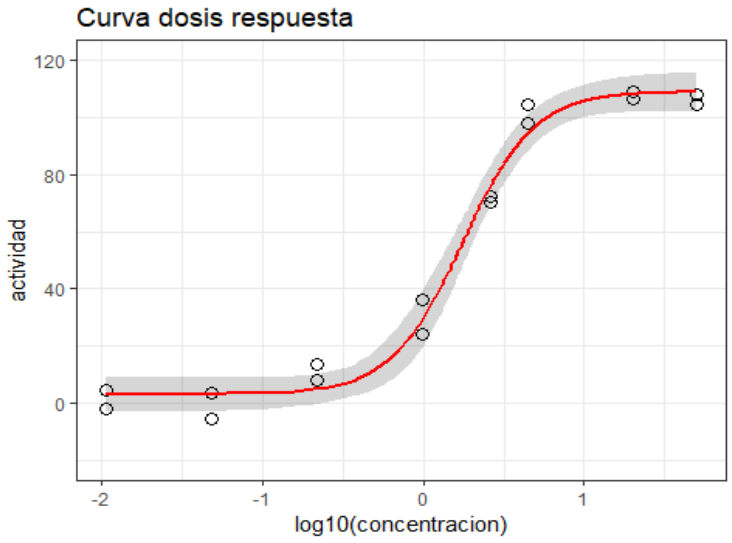
```
##   concentracion  Prediction     Lower    Upper
## 1    0.01080000 -0.21697605 -9.395023 8.961071
## 2    0.06083924 -0.13516962 -9.185813 8.915474
## 3    0.11087848  0.08046217 -8.722471 8.883395
## 4    0.16091772  0.43923185 -8.066327 8.944791
## 5    0.21095696  0.94508745 -7.288161 9.178335
## 6    0.26099620  1.59887209 -6.460533 9.658277
```

```r
# Combinando las observaciones iniciales con los datos predichos, se representan
# los puntos y la curva.

ggplot(data = datos_actividad,
       aes(x = log10(concentracion), y = actividad)) +
    geom_point(size = 3, shape = 1) +
    # Se añade la curva
    geom_ribbon(data = predicciones,
                aes(x = log10(concentracion), y=Prediction, ymin=Lower,
ymax=Upper),
                alpha = 0.2) +
    geom_line(data = predicciones,
              aes(x=log10(concentracion), y= Prediction),
              colour = "red", size=0.8) +
    coord_cartesian(ylim = c(-20, 120)) +
    labs(title = "Curva dosis respuesta") +
    theme_bw()
```



Si existen varios valores de actividad para una misma concentración, por ejemplo, porque el experimento tiene varias repeticiones, la función `drc` las tiene en cuenta automáticamente.

```
concentracion <- c(0.0108, 0.0488, 0.2195, 0.9877, 2.643, 4.4444, 20.0, 50.0,
                   0.0108, 0.0488, 0.2195, 0.9877, 2.643, 4.4444, 20.0, 50.0)
actividad     <- c(-1.6941, -5.6772, 8.2225, 24.2046, 72.145, 98.0494, 106.7267,
                   104.568, 4.771, 3.834, 13.686, 36.412, 70.215, 104.320,
                   109.035, 108.163)
datos_actividad <- data.frame(concentracion, actividad)
datos_actividad$log_conc <- log10(concentracion)

curve_fit <- drm(formula = actividad ~ concentracion, data = datos_actividad,
                 na.action = na.omit,
                 fct = LL.4(fixed = c(NA,NA,NA,NA),
                            names = c("Hill","Bottom","Top","IC50")))

grid_concentraciones <- seq(from = min(datos_actividad$concentracion),
                            to = max(datos_actividad$concentracion),
                            length.out = 1000)
grid_concentraciones <- data.frame(grid_concentraciones)

predicciones <- predict(object = curve_fit,
                        newdata = grid_concentraciones,
                        interval = "confidence")
predicciones <- cbind(grid_concentraciones, predicciones)
colnames(predicciones)[colnames(predicciones) == "grid_concentraciones"] <-
                                                              "concentracion"

ggplot(data = datos_actividad,
       aes(x = log10(concentracion), y = actividad)) +
    geom_point(size = 3, shape = 1) +
    geom_ribbon(data = predicciones,
                aes(x = log10(concentracion), y=Prediction, ymin=Lower,
ymax=Upper),
                alpha = 0.2) +
    geom_line(data = predicciones,
              aes(x=log10(concentracion), y= Prediction),
              colour = "red", size=0.8) +
    coord_cartesian(ylim = c(-20, 120)) +
    labs(title = "Curva dosis respuesta") +
    theme_bw()
```

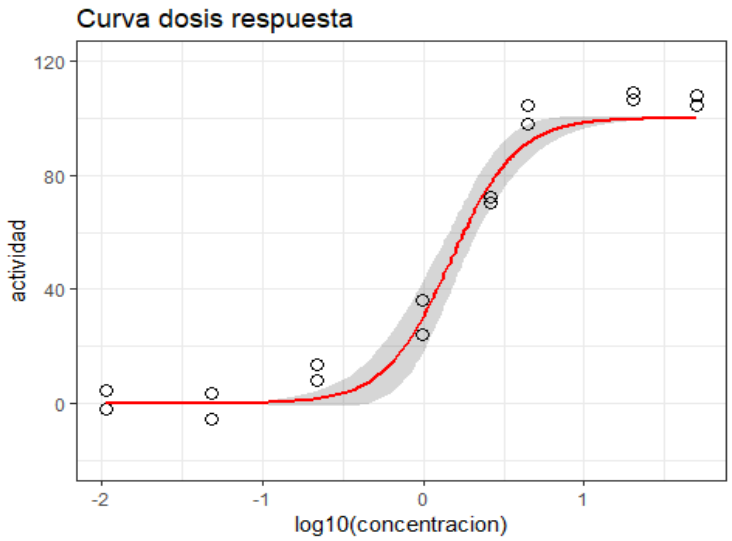


En la práctica, los resultados obtenidos en los experimentos suelen contener bastante ruido, lo que hace que los datos no se ajusten bien a una curva sigmoide, produciendo valores para algunos de los parámetros que no tienen sentido desde el punto de vista biológico, por ejemplo, que el *top* de la curva sea muy superior al 100% o que el *bottom* sea muy inferior a 0%. En estos casos se pueden fijar algunos de los parámetros de la curva.

```
# Ajuste de una curva dosis respuesta fijando el bottom a cero y el top a 100.
curve_fit <- drm(formula = actividad ~ concentracion, data = datos_actividad,
                 na.action = na.omit,
                 fct = LL.4(fixed = c(NA,0,100,NA),
                            names = c("Hill","Bottom","Top","IC50")))

# Los parámetros fijados no aparecen en el summary
summary(curve_fit)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) (2 parms)
##
## Parameter estimates:
##
##                  Estimate Std. Error t-value   p-value
## Hill:(Intercept) -2.12637    0.33865  -6.279 2.025e-05 ***
## IC50:(Intercept)  1.47841    0.14288  10.347 6.112e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  7.846383 (14 degrees of freedom)
```

```
grid_concentraciones <- seq(from = min(datos_actividad$concentracion),
                            to = max(datos_actividad$concentracion),
                            length.out = 1000)
grid_concentraciones <- data.frame(grid_concentraciones)

predicciones <- predict(object = curve_fit,
                        newdata = grid_concentraciones,
                        interval = "confidence")
predicciones <- cbind(grid_concentraciones, predicciones)
colnames(predicciones)[colnames(predicciones) == "grid_concentraciones"] <-
                                                              "concentracion"

ggplot(data = datos_actividad,
       aes(x = log10(concentracion), y = actividad)) +
    geom_point(size = 3, shape = 1) +
    geom_ribbon(data = predicciones,
                aes(x = log10(concentracion), y=Prediction, ymin=Lower,
ymax=Upper),
                alpha = 0.2) +
    geom_line(data = predicciones,
              aes(x=log10(concentracion), y= Prediction),
              colour = "red", size=0.8) +
    coord_cartesian(ylim = c(-20, 120)) +
    labs(title = "Curva dosis respuesta") +
    theme_bw()
```




```
# Se hace un detach del paquete drc y MASS para que no haya conflicto de funciones.
detach("package:drc", unload=TRUE)
```

Datos

Actividad de fármacos

El proyecto *Genomics of Drug Sensitivity in Cancer* se creó como una colaboración entre los centros de investigación *Wellcome Sanger Institute* y el *Center for Molecular Therapeutics, Massachusetts General Hospital Cancer Center*. El objetivo de este proyecto fue lanzar un plataforma a gran escala que permitiera analizar la actividad antitumoral de moléculas en una cantidad de líneas celulares lo suficientemente grande para que representaran a la población de pacientes de cáncer. Como resultado, se analizó un total de 266 moléculas en más de 1000 líneas celulares cancerígenas. En el momento de redactar este documento, la versión disponible es (*v17.3*).

- **PANCANCER_IC.csv** (28.4Mb): archivo coma delimitado con información detallada la actividad de cada uno de los fármacos en cada una de las líneas celulares. Las variables empleadas en el desarrollo de este documento son:
  - Drug name: identificador único (nombre) del fármaco.
  - Cell line name: identificador único (nombre) de la línea celular en la que se ha registrado la actividad.
  - IC50: valor de actividad.

Si bien el objetivo es que los investigadores empleen sus propios datos de actividad, entre las moléculas analizadas en el proyecto *Genomics of Drug Sensitivity in Cancer*, se incluyen fármacos que llevan años en el mercado y cuyas asociaciones con los patrones moleculares están bien descritos. Esto supone una fuente de información muy útil para validar nuevas estrategias de análisis.

En este documento, se emplean como ejemplo los datos del fármaco *AZ628*, un inhibidor *Raf kinase* desarrollado por *AstraZeneca*. Su eficacia ha sido probada para la inhibición de la proliferación tumoral especialmente en líneas con mutaciones *BRAF V600E*. Produce arresto del ciclo celular, lo que induce la apoptosis y, finalmente, la muerte celular.

Mutaciones y expresión genética

Para la identificación de biomarcadores, es necesario disponer, además de la información sobre la actividad, información sobre las mutaciones y los niveles de expresión genética de cada una de las líneas cancerígenas en las que se ha evaluado el fármaco.

Son varios los proyectos europeos y americanos que han puesto a disposición pública este tipo de información. *COSMIC* (*Catalogue Of Somatic Mutations In Cancer*) es una base de datos que recopila información molecular sobre 1015 líneas celulares y que se ha convertido en el mayor referente en cuanto a información de esta naturaleza. *COSMIC* ofrece acceso a toda su información sin costes de licencia a toda la comunidad de investigadores siempre y cuando no tengan fines comerciales. En el momento de redactar este documento, la versión disponible es (*release v85, 8th May 2018*). Los dos data sets empleados en este proyecto son:

- **Complete mutation data** (408.4Mb): archivo tab delimitado con información detallada sobre las mutaciones de cada una de las líneas celulares incluidas en el proyecto *Cosmic Cell Lines*. De entre las múltiples columnas que contiene el data set, para los análisis que se realizan en el programa aquí presentado se emplean:
  - Gene name: identificador único (nombre) del gen.
  - Sample name: identificador único de cada una de las líneas celulares.
  - Mutation Description: tipo de mutación a nivel de aminoácidos (substitution, deletion, insertion, complex, fusion etc.)
  - Primary site: tejido del que se ha extraído el tumor.
- **Gene expression** (650.1Mb): archivo tab delimitado con los niveles de expresión de todos los genes en cada una de las líneas celulares incluidas en el proyecto *Cosmic Cell Lines*. Los niveles de expresión se han obtenido con la tecnología * Affymetrix Human Genome U219 Array*.
  - Sample name: identificador único de cada una de las líneas celulares.
  - Gene Name: identificador único (nombre) del gen.
  - Z-score: nivel de expresión normalizado.

Pathways

ConsensusPathDB-human es una base de datos creada por el *Max Planck Institute for Molecular Genetics* que integra información sobre los genes que codifican las proteínas que participan en cada una de las rutas de señalización (*pathways*). Esta información es muy importante cuando se desea conocer si las alteraciones genéticas presentes un determinado cáncer están localizadas en una ruta de señalización concreta. EL contenido de la base de datos puede descargarse libremente en un archivo texto llamado *CPDB_pathways_genes.tab*. En su interior se encuentra la siguiente información:

- Pathway: nombre del *pathway*.
- Source: fuente de original de la que se ha obtenido la información.
- hgnc_symbol_ids: nombre de los genes que participan en el *pathway*.

Carga de datos

Se cargan los 3 sets de datos (mutaciones, expresión y *pathways*) empleados a lo largo del documento. Hay que tener en cuenta que todos ellos suman aproximadamente 1 GB, por lo que pueden saturar la memoria RAM. De ser así, es aconsejable cargarlos individualmente cuando sean necesarios y eliminarlos cuando no.

```
library(tidyverse)

# EXPRESIÓN GENÉTICA
# ==============================================================================
# Se cargan únicamente las columnas de interés que sea más rápido.
datos_expresion <- read_delim(file = "CosmicCLP_CompleteGeneExpression.txt",
                              col_names = TRUE,
                              col_types = cols_only(SAMPLE_NAME = "c",
                                                    GENE_NAME   = "c",
                                                    Z_SCORE     = "n"),
                              delim = "\t")
datos_expresion <- datos_expresion %>% rename(sample_name = SAMPLE_NAME,
                                              gene_name   = GENE_NAME,
                                              z_score     = Z_SCORE)
head(datos_expresion)
```

```
## # A tibble: 6 x 3
##   sample_name gene_name z_score
##   <chr>       <chr>       <dbl>
## 1 MC-CAR      NXNL2       -1.71
## 2 MC-CAR      CCR1        -1.03
## 3 MC-CAR      SH3KBP1      1.35
## 4 MC-CAR      GIMAP5       2.17
## 5 MC-CAR      FAM46D       1.23
## 6 MC-CAR      NR2E3       -0.92
```

Al tratarse de valores normalizados, aquellos valores mayores que 2 se consideran atípicamente alto (gen con alta expresión) y aquellos con un valor menor que -2 se consideran atípicamente bajos (genes con baja expresión).

```
# MUTACIONES
# ==============================================================================
# Se cargan únicamente las columnas de interés que sea más rápido.
datos_mutaciones <- read_delim(file = "CosmicCLP_MutantExport.txt",
                               col_names = TRUE,
                               col_types = cols_only(
                                             `Gene name`    = "c",
                                             `Sample name`  = "c",
                                             `Primary site` = "c",
                                             `Mutation Description` = "c"),
                               delim = "\t")

datos_mutaciones <- datos_mutaciones %>% rename(
                                          gene_name    = `Gene name`,
                                          sample_name  = `Sample name`,
                                          primary_site = `Primary site`,
                                          mutation     = `Mutation Description`
                                         )
# Se filtran los transcritos alternativos: sus nombres siguen el siguiente patrón.
# Ejemplo: ASTN1 -> ASTN1_ENST00000367654, todos tienen el separador "_"o ".".
library(stringr)
datos_mutaciones <- datos_mutaciones %>%
                    filter(!(str_detect(string = gene_name, pattern = "_"))) %>%
                    filter(!(str_detect(string = gene_name, pattern = "[.]")))
head(datos_mutaciones)
```

```
## # A tibble: 6 x 4
##   gene_name sample_name primary_site                mutation
##   <chr>     <chr>       <chr>                       <chr>
## 1 KRAS      PL-21       haematopoietic_and_lymphoid~ Substitution - Misse~
## 2 P2RY2     A375        skin                        Substitution - Misse~
## 3 SALL4     MCC26       skin                        Substitution - Misse~
## 4 SLC35F2   LS-411N     large_intestine             Substitution - codin~
```

```
# PATHWAYS
# ================================================================================
datos_pathways <- read_delim(file = "CPDB_pathways_genes.txt",
                             col_names = TRUE,
                             col_types = cols_only(
                                                   pathway = "c",
                                                   source  = "c",
                                                   hgnc_symbol_ids = "c"
                                                   ),
                               delim = "\t")
head(datos_pathways)
```

```
## # A tibble: 6 x 3
##   pathway                       source hgnc_symbol_ids
##   <chr>                         <chr>  <chr>
## 1 PI3K-Akt signaling pathway -~ KEGG   BCL2L1,PCK2,PCK1,GHR,PPP2R3B,PPP2R~
## 2 Alanine, aspartate and gluta~ KEGG   NIT2,ADSS,ASNS,GLUL,ABAT,ADSL,GLS,~
## 3 Folate biosynthesis - Homo s~ KEGG   ALPP,ALPPL2,DHFR,QDPR,SPR,GCH1,MOC~
## 4 Complement and coagulation c~ KEGG   KNG1,F11,CD55,C1QC,VSIG4,CD59,PROS~
## 5 Citrate cycle (TCA cycle) - ~ KEGG   PCK2,PCK1,CS,MDH2,MDH1,FH,IDH3A,OG~
## 6 Antigen processing and prese~ KEGG   CD74,HSPA2,TAP2,KLRD1,HSPA5,HSPA4,~
```

```
# ACTIVIDAD DE AZ628
# ================================================================================
datos_actividad <- read_csv(file = "compound_activity_palbociclib_cosmic_lung.csv")
head(datos_actividad)
```

```
## # A tibble: 6 x 7
##   cell_line id_compuesto Tissue `Tissue sub-typ~ actividad abs_IC50_uM
##   <chr>     <chr>        <chr>  <chr>                <dbl>       <dbl>
## 1 A549      palbociclib  lung   lung_NSCLC_aden~      0.68         648
## 2 SK-LU-1   palbociclib  lung   lung_NSCLC_aden~    709            646
## 3 NCI-H838  palbociclib  lung   lung_NSCLC_aden~    788            675
## 4 NCI-H3122 palbociclib  lung   lung_NSCLC_aden~    831            684
## 5 NCI-H1355 palbociclib  lung   lung_NSCLC_aden~      1.16         773
## 6 NCI-H1703 palbociclib  lung   lung_NSCLC_aden~      1.19         716
## # ... with 1 more variable: max_activity <dbl>
```

Exploración de datos

Antes de empezar con los análisis estadísticos, conviene realizar una exploración de los datos disponibles.

Líneas celulares

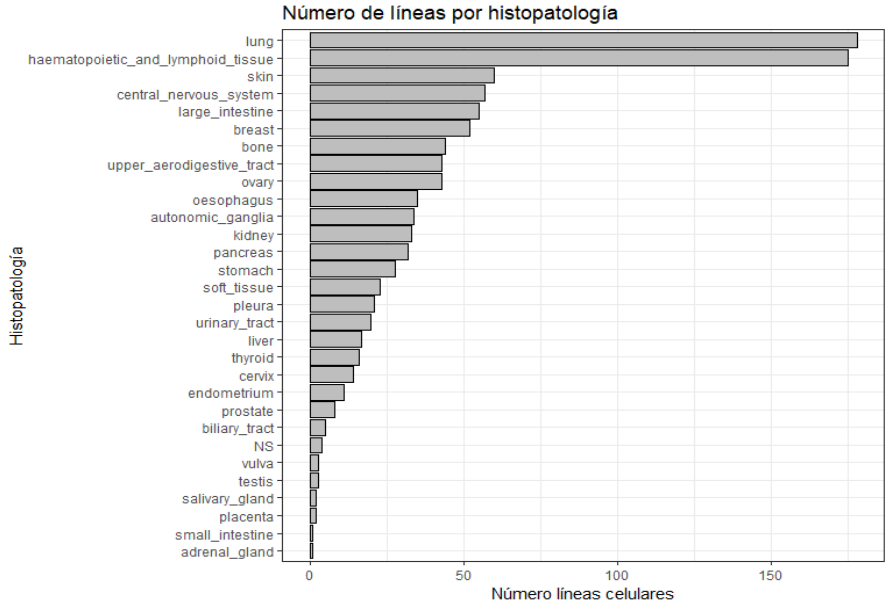
Número total de líneas celulares para las que se dispone de información genética:

```
datos_mutaciones %>% pull(sample_name) %>% unique() %>% length()
```

```
## [1] 1020
```

Número total de líneas celulares por histopatología (tejido de origen):

```
datos_mutaciones %>% select(sample_name, primary_site) %>% unique() %>%
                     group_by(primary_site) %>% count() %>%
                     ggplot(aes(x = reorder(primary_site, n), y = n)) +
                      geom_col(fill = "gray", color = "black") +
                      coord_flip() +
                      theme_bw() +
                      labs(title = "Número de líneas por histopatología",
                           x = "Histopatología",
                           y = "Número líneas celulares")
```

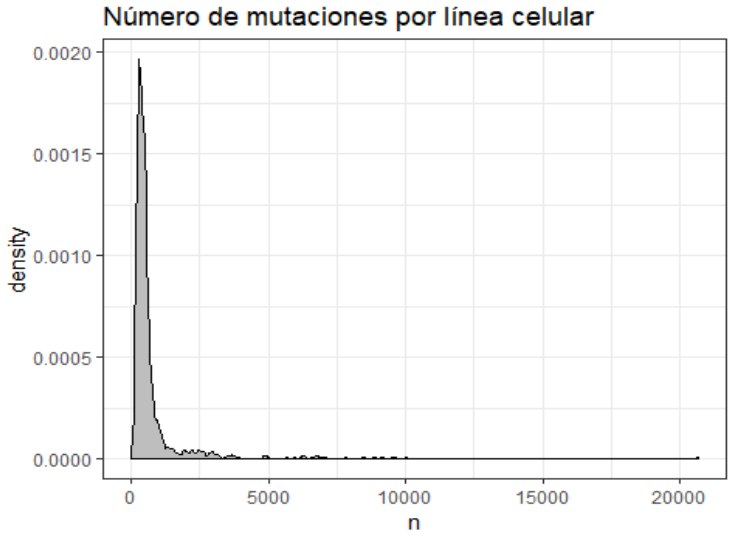


Número de mutaciones

```
mutaciones_por_linea <- datos_mutaciones %>%
                        select(sample_name, gene_name) %>%
                        group_by(sample_name) %>%
                        count() %>%
                        arrange(desc(n))
head(mutaciones_por_linea)
```

```
## # A tibble: 6 x 2
## # Groups:   sample_name [6]
##   sample_name     n
##   <chr>       <int>
## 1 SNU-1040    20675
## 2 Daudi       10046
## 3 CW-2         9567
## 4 SNU-81       9171
## 5 HCC2998      8882
## 6 KARPAS-45    8480
```

```
ggplot(data = mutaciones_por_linea, aes(x = n)) +
  geom_density(fill = "gray") +
  labs(title = "Número de mutaciones por línea celular") +
  theme_bw()
```




```
summary(mutaciones_por_linea$n)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     5.0   285.0   424.5   709.5   600.5 20675.0
```

```
rm(mutaciones_por_linea)
```

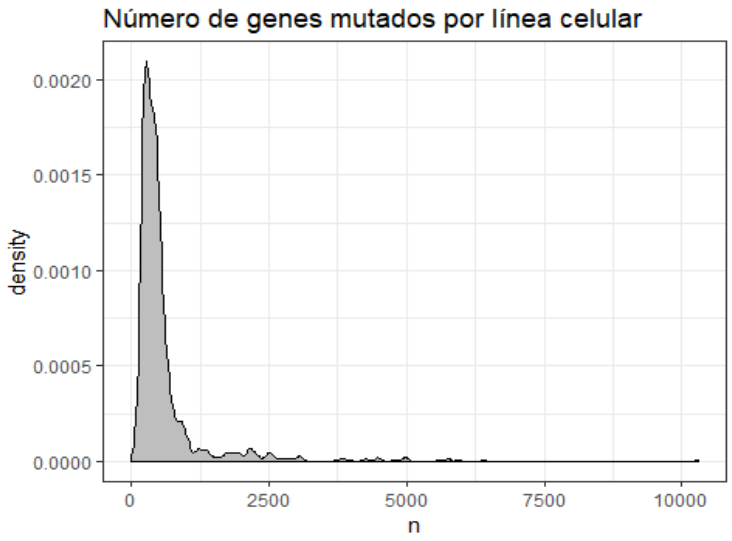
Puede observarse que, la distribución del número de mutaciones por línea celular, es muy asimétrica. La gran mayoría de líneas acumulan en torno a 600-700 mutaciones, con algunas excepciones en las que se dispara esta cantidad.

Los valores anteriores pueden resultar sospechosos puesto que, algunas líneas, tienen un número de mutaciones superior al número de genes. Esto es posible ya que, un mismo gen, puede contener múltiples mutaciones. Véase ahora el número de genes mutados por línea celular.

```
genes_mut_por_linea <- datos_mutaciones %>%
                       select(sample_name, gene_name) %>%
                       unique() %>%
                       group_by(sample_name) %>%
                       count() %>%
                       arrange(desc(n))
head(genes_mut_por_linea)
```

```
## # A tibble: 6 x 2
## # Groups:   sample_name [6]
##   sample_name     n
##   <chr>       <int>
## 1 SNU-1040    10295
## 2 CW-2         6403
## 3 SNU-81       5943
## 4 HCC2998      5756
## 5 KARPAS-45    5739
## 6 HCT-15       5592
```

```
ggplot(data = genes_mut_por_linea, aes(x = n)) +
  geom_density(fill = "gray") +
  labs(title = "Número de genes mutados por línea celular") +
  theme_bw()
```



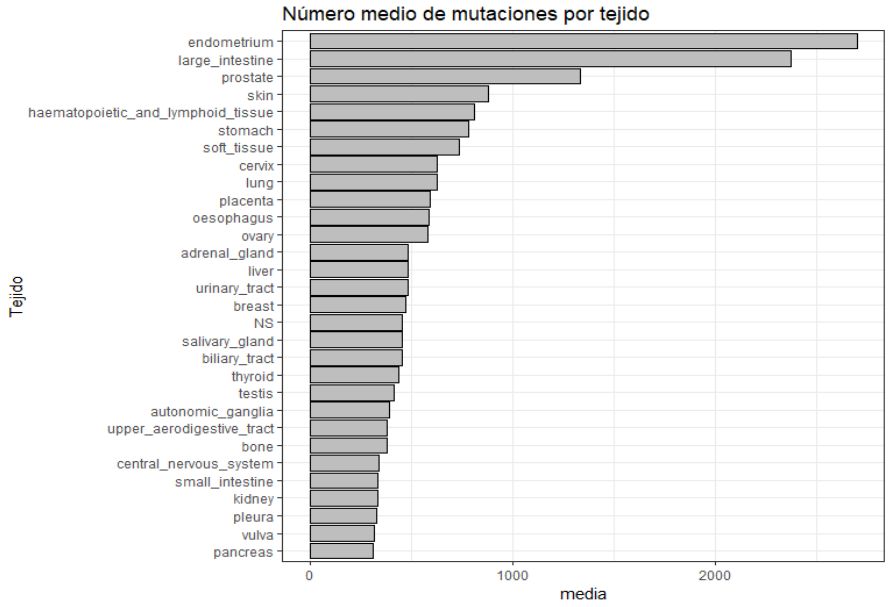
```
summary(genes_mut_por_linea$n)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     5.0   274.5   405.5   612.1   565.0 10295.0
```

```
rm(genes_mut_por_linea)
```

Número medio de mutaciones por tejido (un gen mismo gen puede tener múltiples mutaciones):

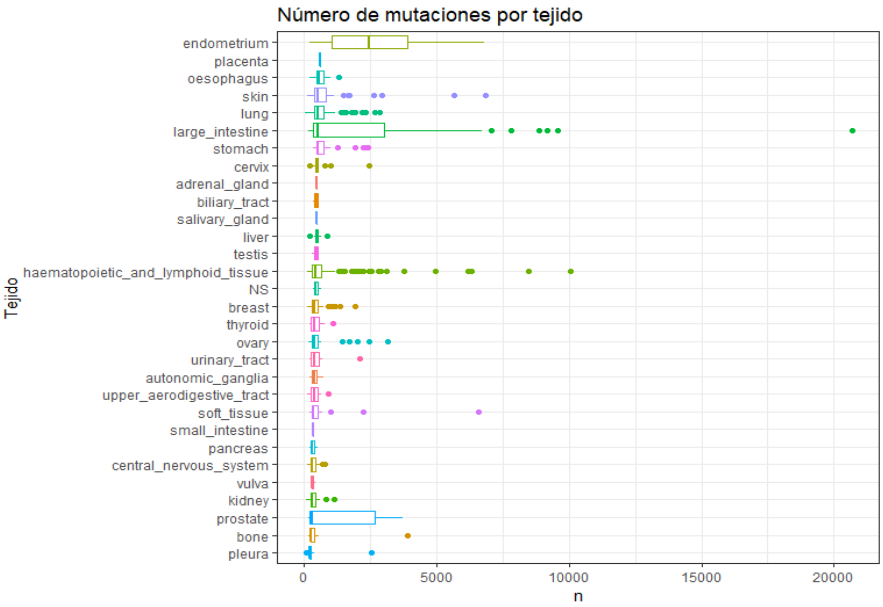
```
mutaciones_por_tejido <- datos_mutaciones %>%
                         group_by(primary_site, sample_name) %>%
                         count() %>%
                         ungroup() %>%
                         group_by(primary_site) %>%
                         summarise(media = mean(n)) %>%
                         arrange(desc(media))

ggplot(data = mutaciones_por_tejido,
       aes(x = reorder(primary_site, media), y = media)) +
  geom_col(fill = "gray", color = "black") +
  coord_flip() +
  theme_bw() +
  labs(title = "Número medio de mutaciones por tejido",
       x = "Tejido")
```




```
summary(mutaciones_por_tejido$media)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   311.7   386.1   479.3   664.1   629.7  2697.9
```

```
rm(mutaciones_por_tejido)
datos_mutaciones %>%
  group_by(primary_site, sample_name) %>%
  count() %>%
  group_by(primary_site) %>%
  mutate(mediana = median(n)) %>%
  ungroup() %>%
  ggplot(aes(x = reorder(primary_site, mediana),  y = n, color = primary_site)) +
    coord_flip() +
    geom_boxplot() +
    labs(title = "Número de mutaciones por tejido",
         x = "Tejido") +
    theme_bw() +
    theme(legend.position = "none")
```



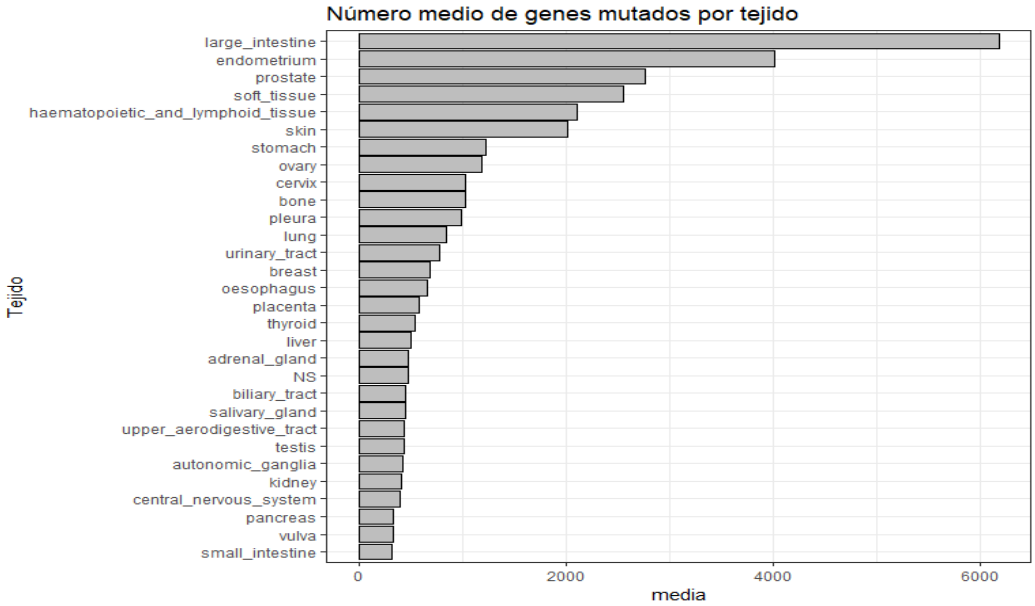
Número medio de genes mutados por tejido:

```
genes_mut_por_tejido <- datos_mutaciones %>%
                        group_by(sample_name, gene_name) %>%
                        unique() %>%
                        select_all() %>%
                        ungroup() %>%
                        group_by(sample_name) %>%
                        count() %>%
                        ungroup() %>%
                        left_join(y = datos_mutaciones %>%
                                      select(sample_name, primary_site)
                                  ) %>%
                        group_by(primary_site) %>%
```

```
                          summarise(media = mean(n)) %>%
                          arrange(desc(media))
head(genes_mut_por_tejido)
```

```
## # A tibble: 6 x 2
##   primary_site                        media
##   <chr>                               <dbl>
## 1 large_intestine                     6178.
## 2 endometrium                         4016.
## 3 prostate                            2759.
## 4 soft_tissue                         2550.
## 5 haematopoietic_and_lymphoid_tissue 2109.
## 6 skin                                2020.
```

```
ggplot(data = genes_mut_por_tejido,
       aes(x = reorder(primary_site, media), y = media)) +
  geom_col(fill = "gray", color = "black") +
  coord_flip() +
  theme_bw() +
  labs(title = "Número medio de genes mutados por tejido",
       x = "Tejido")
```

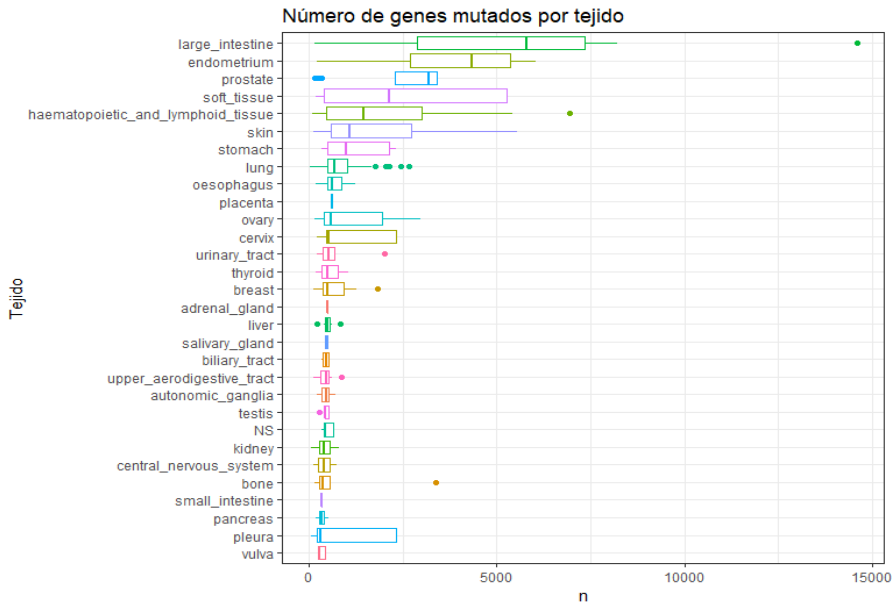



```
summary(genes_mut_por_tejido$media)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   325.0   440.3   623.0  1155.8  1150.9  6177.8
```

```
rm(genes_mut_por_tejido)
```

```
datos_mutaciones %>%
   group_by(sample_name, gene_name) %>%
   unique() %>%
   select_all() %>%
   group_by(sample_name) %>%
   count() %>%
   left_join(datos_mutaciones %>%
             select(sample_name, primary_site)
            ) %>%
  group_by(primary_site) %>%
  mutate(mediana = median(n)) %>%
  ungroup() %>%
  ggplot(aes(x = reorder(primary_site, mediana),  y = n, color = primary_site)) +
    coord_flip() +
    geom_boxplot() +
    labs(title = "Número de genes mutados por tejido",
         x = "Tejido") +
    theme_bw() +
    theme(legend.position = "none")
```



Los tejidos que más mutaciones y genes mutados acumulan son: *large instestine*, *endometrium* y *prostate*.

Número promedio de genes por pathway

```
# Función para contar cuantos genes hay en cada patway.
contar_elementos <- function(x){
  x_split <- str_split(string = x, pattern = ",") %>% unlist()
  return(length(x_split))
}

datos_pathways <- datos_pathways %>%
                  mutate(n_genes = map_int(.x = hgnc_symbol_ids,
                                           .f = contar_elementos
                                           )
                         )
summary(datos_pathways$n_genes)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    8.00   20.00   42.26   43.00 2524.00
```

La gran mayoría de *pathways* (75%) no superan los 43 genes, pero existen algunos casos extremos con hasta 2524 genes. En la siguiente tabla puede verse que estos últimos se corresponden con *pathways* muy genéricos, por ejemplo, metabolismo.

```
datos_pathways %>%
  select(pathway, n_genes) %>%
  arrange(desc(n_genes)) %>% head(10)
```

```
## # A tibble: 10 x 2
##    pathway                                      n_genes
##    <chr>                                          <int>
##  1 Signal Transduction                             2524
##  2 Metabolism                                      2021
##  3 Immune System                                   1936
##  4 Gene Expression                                 1745
##  5 Metabolism of proteins                          1503
##  6 Signaling by GPCR                               1303
##  7 Innate Immune System                            1297
##  8 Post-translational protein modification         1026
##  9 GPCR downstream signaling                       1023
## 10 Generic Transcription Pathway                    856
```

Similitud entre líneas celulares

Desde que en 1951 se generara la primera línea celular inmortalizada *Hela* a partir de una muestra de cáncer cérvico-uterino, le han sucedido cientos de nuevos linajes. Sin embargo, la anotación e identificación de las líneas celulares no ha estado libre de errores y de contaminaciones cruzadas. Por esta razón, es conveniente identificar el grado de similitud que hay entre las líneas celulares que forman un panel de *screening*, puesto que se podrían estar considerando como distintas líneas celulares que son realmente la misma.

Como toda comparación, ha de estar basada en alguna característica que se considere propia de cada línea celular, algunas de las más empleadas son: similitud basada en el perfil de mutaciones, en los niveles de expresión genética y niveles de expresión de proteínas.

Similitud según perfil de expresión genética

```
# Se reestructuran los datos en formato de tabla ancha.
datos_expresion_tabla <- datos_expresion %>%
  spread(key = sample_name, value = z_score)

# Se almacena el nombre de las líneas en el nombre de las filas.
datos_expresion_tabla <- datos_expresion_tabla %>%
  as.data.frame() %>%
  column_to_rownames(var = "gene_name")

# Distancias basadas en correlación pearson (valor absoluto).
mat_distancia <- 1 - abs(cor(x = datos_expresion_tabla, method = "pearson"))

# Listado de líneas por orden de descendiente de distancia
# (menor distancia = mayor similitud).
mat_distancia <- as.matrix(mat_distancia)
mat_distancia[upper.tri(mat_distancia)] <- NA
similitud_tidy <- as.data.frame(mat_distancia) %>%
                  rownames_to_column(var = "linea1") %>%
                  gather(key = "linea2", value = "distancia", -linea1) %>%
                  unique() %>%
                  filter(linea1 != linea2 & !is.na(distancia)) %>%
                  arrange(distancia)

# Se muestran los 20 pares de líneas más similares.
head(similitud_tidy, n = 20)
```

```
##            linea1     linea2 distancia
## 1       LB771-HNC      FLO-1 0.2500546
## 2           EW-22      EW-11 0.3676458
## 3       RPMI-6666       EHEB 0.3678125
## 4       RPMI-6666       CESS 0.3747457
## 5        BB30-HNC       A388 0.3854252
## 6            EHEB       CESS 0.3927492
## 7         HCC1806    HCC1143 0.3986833
## 8      MHH-CALL-4 MHH-CALL-2 0.4030271
## 9          MOLT-4   CCRF-CEM 0.4030643
## 10         DND-41   CCRF-CEM 0.4145064
## 11         SUP-B8    P32-ISH 0.4185976
## 12          JVM-2       HC-1 0.4237229
## 13         SUP-B8    OCI-LY7 0.4276465
## 14          KY821      HL-60 0.4278928
## 15      NCI-H2126       EHEB 0.4282145
## 16         MC-CAR       HC-1 0.4286647
## 17       SK-NEP-1      EW-22 0.4326555
## 18         MOLT-4      KE-37 0.4369276
## 19 Ramos-2G6-4C10      BL-41 0.4379790
## 20         MOLT-4    MOLT-13 0.4402484
```

```
rm(list = c("datos_expresion_tabla", "mat_distancia", "similitud_tidy"))
```

Similitud según perfil de mutaciones

En este segundo caso, la similitud entre líneas se cuantifica en función de los genes mutados, sin tener en cuenta el tipo de mutación. Esto significa que, si dos líneas tienen exactamente los mismos genes mutados, se consideran iguales aun cuando las mutaciones no sean las mismas.

Nota: este cálculo de computación pude tardar más de una hora. Puede cargarse directamente el resultado del archivo *similitud_jaccard_lineas.csv* disponible en Github.

```
datos_mutaciones_tabla <- datos_mutaciones %>%
                          select(gene_name, sample_name) %>%
                          mutate(valor = 1) %>%
                          unique() %>%
                          spread(key = sample_name, value = valor, fill = 0)
head(datos_mutaciones_tabla)

# Distancias basadas en índice Jaccard
#==============================================================================

# Se genera un grid con todas las comparaciones que se tienen que realizar
comparaciones <- expand.grid(unique(datos_mutaciones$sample_name),
                             unique(datos_mutaciones$sample_name),
                             stringsAsFactors = FALSE)
```

```r
# Esto genera combinaciones redundantes
comparaciones %>% filter(Var1 == "NCI-H630" & Var2 == "HT-115" |
                         Var2 == "NCI-H630" & Var1 == "HT-115")

# Se eliminan las redundancias
comparaciones <- comparaciones %>%
                 mutate(Var1_sorted = if_else(Var1 < Var2, Var1, Var2),
                        Var2_sorted = if_else(Var1 < Var2, Var2, Var1)) %>%
                 select(Var1_sorted, Var2_sorted) %>%
                 unique() %>%
                 rename(Var1 = Var1_sorted, Var2 = Var2_sorted)

# Se comprueba que no hay redundancias
comparaciones %>% filter(Var1 == "NCI-H630" & Var2 == "HT-115" |
                         Var2 == "NCI-H630" & Var1 == "HT-115")

# Se eliminan pares iguales
comparaciones <- comparaciones %>% filter(!(Var1 == Var2))

# Se define la función que calcula la similitud
indice_jaccard <- function(col1, col2, datos) {
  # Esta función calcula el índice jaccard entre dos columnas de un dataframe.
  # El valor 1 indica presencia y el valor 0 ausencia.
  m11 <- sum(datos[, col1] == 1 & datos[, col2] == 1)
  m10 <- sum(datos[, col1] == 1 & datos[, col2] == 0)
  m01 <- sum(datos[, col1] == 0 & datos[, col2] == 1)
  indice <- m11 / sum(m01 + m10 + m11)
  return(indice)
}


# Con la función map2 del paquete purrr, se aplica la función indice_jaccard
# empleando las columnas del grid comparaciones como valores de los argumentos.
comparaciones <- comparaciones %>%
                 mutate(similitud = map2_dbl(.x = Var1,
                                             .y = Var2,
                                             .f = indice_jaccard,
                                             datos = datos_mutaciones_tabla))
write_csv(x = comparaciones, path = "similitud_jaccard_lineas.csv")
rm(datos_mutaciones_tabla)
```

Se muestran los 20 pares de líneas celulares más similares en función de las mutaciones:

```
comparaciones %>% arrange(desc(similitud)) %>% head(20)
```

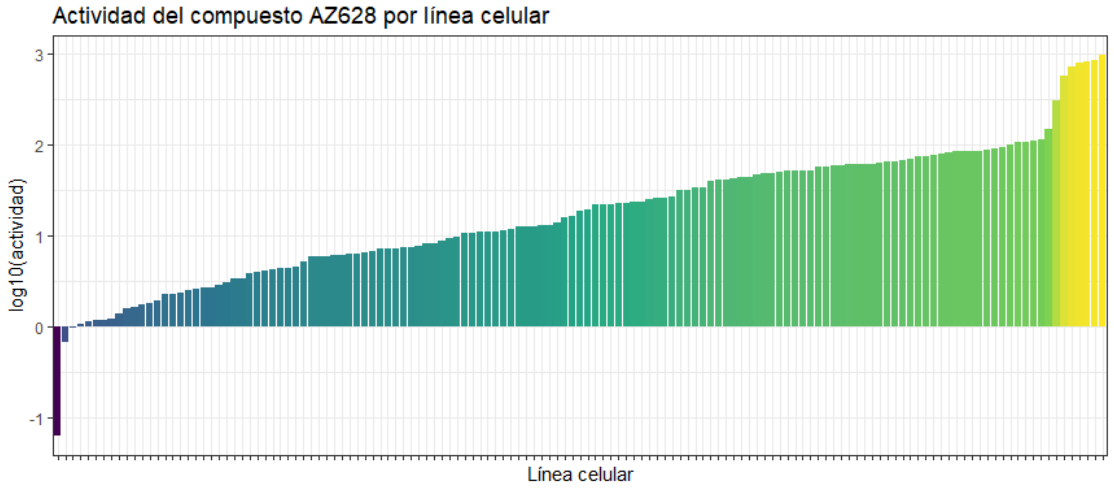
```
## # A tibble: 20 x 3
##    Var1      Var2      similitud
##    <chr>     <chr>         <dbl>
##  1 CW-2      SNU-1040      0.357
##  2 HCT-15    SNU-1040      0.326
##  3 KARPAS-45 SNU-1040      0.323
##  4 SNU-1040  SNU-81        0.321
##  5 HCC2998   SNU-1040      0.318
##  6 HCC2998   SNU-81        0.302
##  7 SNU-1040  SNU-175       0.297
##  8 EN        SNU-1040      0.291
##  9 HT-115    SNU-1040      0.287
## 10 CW-2      SNU-81        0.287
## 11 HT-115    SNU-81        0.285
## 12 CW-2      HCC2998       0.282
## 13 GP5d      SNU-1040      0.282
## 14 CW-2      HCT-15        0.277
## 15 MFE-319   SNU-1040      0.271
## 16 HCC2998   HT-115        0.269
## 17 HCT-15    SNU-81        0.269
## 18 Mewo      SNU-1040      0.265
## 19 CW-2      KARPAS-45     0.265
## 20 CW-2      EN            0.264
```

```
rm(comparaciones)
```

Actividad del compuesto

El siguiente gráfico muestra la actividad registrada para el fármaco *AZ628* en cada una de las líneas celulares.

```
p_barplot <- ggplot(data = datos_actividad,
                    aes(x = reorder(cell_line, log10(actividad)),
                     y = log10(actividad),
                     text = cell_line,
                     fill = log10(actividad))) +
                  geom_col() +
                  scale_fill_viridis_c() +
                  labs(title = "Actividad del compuesto AZ628 por línea celular",
                       x = "Línea celular") +
                  theme_bw() +
                  theme(legend.position = "none",
                        #axis.text.x = element_text(angle = 90, size = 3),
                        axis.text.x = element_blank())
p_barplot
```




```
# Puede obtenerse un gráfico interactivo con plotly.
# Para muchos datos (como en este caso) puede bloquear el explorador.
# library(plotly)
# ggplotly(p = p_barplot, tooltip = "text")
```

Procesado de datos

Reestructuración

Los dos sets de datos distribuidos por COSMIC se encuentran en formato de tabla larga, es decir, cada variable forma una única columna. Para facilitar posteriores cálculos, los datos se reestructuran en formato tabla ancha, permitiendo así trabajar como si se tratase de una matriz.

Los sets de datos contienen información de más de 1000 líneas celulares, para agilizar el proceso, antes de reestructurar los datos, se filtra únicamente información sobre las líneas celulares para las que el analista ha cargado información de actividad (el filtrado en formato tabla larga es más eficiente).

Recodificación

Tal y como se describe más adelante, la identificación de mutaciones asociadas con la actividad del fármaco se realiza mediante contrastes de hipótesis que comparan la respuesta observada en las líneas celulares mutadas y no mutadas. Para poder realizar este contraste, es necesario convertir la matriz que contiene la información de las mutaciones en una matriz binaria. Si la línea celular tiene mutado el gen, independientemente del tipo de mutación, se codifica como "Y" (Yes), y si no está mutada, campo vacío, se codifica como "N" (No).

Existe un caso particular de mutación cuyas alteraciones en el DNA no repercuten en la proteína codificada, a este tipo de mutaciones se les conoce como *Silent*. Si así lo especifica el usuario, se codifican estas mutaciones como si no existiesen.

Código

```
# TRATAMIENTO DE LAS MUTACIONES TIPO SILENT
# ==============================================================================
# Si ignorar_silent == TRUE, se ignoran las mutaciones de tipo silent.
ignorar_silent <- TRUE

if(isTRUE(ignorar_silent)){
  datos_mutaciones <- datos_mutaciones %>%
                      filter(mutation != "Substitution - coding silent")
}

# FILTRADO DE LA INFORMACIÓN DE LAS LÍNEAS DE INTERÉS
# ==============================================================================
# Se selecciona solo la información de las líneas de interés.
lineas <- unique(datos_actividad$cell_line)
datos_mutaciones <- datos_mutaciones %>%
                    filter(sample_name %in% lineas)

# ESTRUCTURA DE TABLA ANCHA
# ==============================================================================
# Se reestructura en formato tabla ancha los datos de mutaciones
datos_mutaciones <- datos_mutaciones %>%
                    select(sample_name, gene_name) %>%
                    distinct() %>%
                    spread(key = gene_name,
                           value = gene_name,
                           fill = NA)

# RECODIFICACIÓN BINARIA DEL ESTADO DE CADA GEN
# ==============================================================================
# Se recodifica el estado de los genes en binario Y/N
custom_replacer <- function(x){
  x <- as.character(x)
  x[!is.na(x)] <- "Y"
  x[is.na(x)] <-  "N"
  return(x)
}

datos_mutaciones <- datos_mutaciones %>%
                    map_at(.at = 2:ncol(datos_mutaciones),
                           .f  = custom_replacer) %>%
                    as.data.frame()
```

```
# JOIN CON LA ACTIVIDAD DEL FÁRMACO
# ============================================================================
# Se añade una nueva columna con la información de la actividad del fármaco en
# cada una de las líneas celulares.
datos_mutaciones <- datos_mutaciones %>%
                    left_join(y = select(datos_actividad, actividad, cell_line),
                              by = c("sample_name" = "cell_line"))
```

Se repite el mismo proceso pero esta vez con los datos de expresión.

```
# FILTRADO DE LA INFORMACIÓN DE LAS LÍNEAS DE INTERÉS
# ============================================================================
# Se selecciona solo la información de las líneas de interes.
lineas <- unique(datos_actividad$cell_line)
datos_expresion <- datos_expresion %>%
                   filter(sample_name %in% lineas)

# ESTRUCTURA DE TABLA ANCHA
# ============================================================================
# Se reestructura en formato tabla ancha los datos de mutaciones
datos_expresion <- datos_expresion %>%
                   spread(key = gene_name,
                          value = z_score,
                          fill = NA)
head(datos_expresion)
```

```
## # A tibble: 6 x 16,375
##   sample_name   A1BG   A1CF    A2M  A2ML1 A3GALT2P A4GALT  A4GNT   AAAS
##   <chr>        <dbl>  <dbl>  <dbl>  <dbl>    <dbl>  <dbl>  <dbl>  <dbl>
## 1 201T        -1.21  -1.2   -0.263 -1.84    -0.988  0.721 -0.62  -0.084
## 2 A427         1.57  -0.447 -3.27   1.15     0.285  1.01  -0.062  0.806
## 3 A549         1.01   1.83  -0.233  0.043   -0.205  0.204 -0.594  1.48
## 4 CAL-12T     -0.197 -0.052 -0.768  1.90    -2.02   1.44   0.293  0.933
## 5 Calu-3      -0.714 -0.679 -1.61  -0.512    0.426  0.204 -0.637 -2.14
## 6 Calu-6      -1.04   1.32  -0.203  1.78     0.668 -2.47  -0.26  -0.005
## # ... with 16,366 more variables: AACS <dbl>, AADAC <dbl>, AADACL2 <dbl
```

```
# JOIN CON LA ACTIVIDAD DEL FÁRMACO
# ============================================================================
# Se añade una nueva columna con la información de la actividad del fármaco en
# cada una de las líneas celulares.
datos_expresion <- datos_expresion %>%
                   left_join(y = select(datos_actividad, actividad, cell_line),
                             by = c("sample_name" = "cell_line"))
```

Identificación de mutaciones

Que una mutación esté asociada (que no necesariamente causa) con la respuesta que un tumor muestra ante un fármaco significa que la actividad observada en líneas celulares que poseen dicha alteración es distinta a la respuesta observada en líneas celulares que no tienen la mutación. El análisis de estos experimentos requiere de métodos que comparen cuantitativamente las muestras con la finalidad de determinar si las diferencias observadas suponen una evidencia suficiente de que las poblaciones son distintas, es decir, si existen diferencias entre el grupo mutado y no mutado más allá de lo que cabría esperar por azar. Dos test estadísticos se han implementado para dar respuesta a esta pregunta: uno paramétrico T-test y otro no paramétrico Mann–Whitney–Wilcoxon. El proceso seguido en ambos es el mismo (Figura 3): para cada uno de los genes, se identifican qué líneas celulares están mutadas y cuáles no, se calcula un estadístico que agregue la actividad en cada grupo (media, mediana…) y se aplica el test con el que conocer si la diferencia observada es significativa (*p-value*).

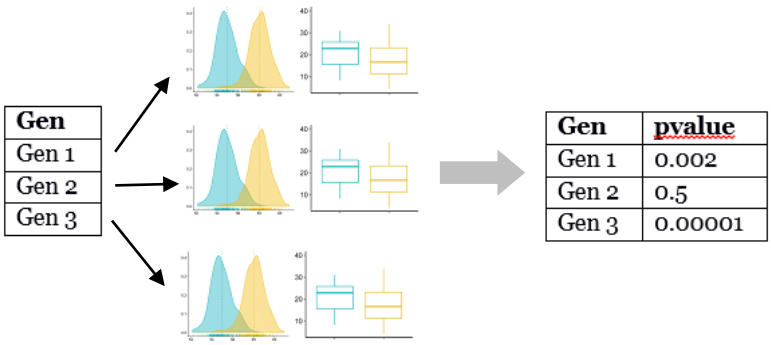


Fig.3. Esquema de las comparaciones múltiples con las que se calcula la significancia asociada a cada uno de los genes.

T-test

El término T-test engloba al conjunto de test en los que el estadístico estudiado sigue una distribución *t-student* acorde a la hipótesis nula. Una de sus principales aplicaciones es estudiar si la media de dos poblaciones es distinta a partir de la comparación de las medias de muestras obtenidas de cada una de ellas. En concreto, las hipótesis contrastadas son:

$$H_0: \mu_A = \mu_B$$

$$H_A: \mu_A \neq \mu_B$$

El valor devuelto por el test se conoce como *p-value* y se corresponde como la probabilidad de obtener una diferencia igual o superior a la observada si se cumple la hipótesis nula. En este caso, puede interpretarse como la probabilidad de observar una diferencia en las medias igual o mayor si realmente no hay ninguna relación entre la mutación del gen estudiado y la respuesta del fármaco.

A pesar de la sencillez y utilidad del T-test, para que sus resultados sean válidos es necesario que se cumplan una serie de condiciones, entre las que se encuentran:

- Independencia: Las observaciones tienen que ser independientes unas de las otras.

- Normalidad: Las poblaciones que se comparan tienen que distribuirse de forma normal. A pesar de que la condición de normalidad recae sobre las poblaciones, normalmente no se dispone de información sobre ellas, por lo que las muestras (dado que son reflejo de la población) tienen que distribuirse de forma aproximadamente normal. En caso de cierta desviación de la normalidad, los T-test son considerablemente robustos cuando el tamaño de las muestras de ambos grupos es mayor o igual a 30.

- Igualdad de varianza (homocedasticidad): la varianza de las poblaciones comparadas debe de ser igual. Tal como ocurre con la condición de normalidad, si no se dispone de información de las poblaciones, esta condición se ha de asumir a partir de las muestras. En caso de no cumplirse esta condición, se puede emplear un *Welch Two Sample t-test*. Esta corrección se incorpora a través de los grados de libertad permitiendo compensar la diferencia de varianzas pero con el inconveniente de que pierde poder estadístico.

Mann–Whitney–Wilcoxon

El test de *Mann–Whitney–Wilcoxon (WMW)*, también conocido como *Wilcoxon rank-sum test* o *u-test*, es un test no paramétrico que contrasta si dos muestras proceden de poblaciones equidistribuidas. La idea en la que se fundamenta este test es la siguiente: si las dos muestras comparadas proceden de la misma población, al juntar todas las observaciones y ordenarlas de menor a mayor, cabe esperar que las observaciones de una y otra muestra estén intercaladas aleatoriamente. Por lo contrario, si una de las muestras pertenece a una población con valores mayores o menores que la otra población, al ordenar las observaciones, estas tenderán a agruparse de modo que, las de una muestra, queden por encima de las de la otra.

Acorde a esta idea, el test de *Mann–Whitney–Wilcoxon* contrasta que la probabilidad de que una observación de la población X supere a una observación de la población Y es igual a la

probabilidad de que una observación de la población Y supere a una de la población X. Es decir, que los valores de una población no tienden a ser mayores que los de otra.

$$H_0: P(X > Y) = P(Y > X)$$

$$H_A: P(X > Y) \neq P(Y > X)$$

Para que sus resultados del test de *Mann–Whitney–Wilcoxon* sean válidos, es necesario que se cumplan una serie de condiciones, entre las que se encuentran:

- Los datos tienen que ser independientes.
- Los datos tienen que ser ordinales o bien se tienen que poder ordenar de menor a mayor.
- No es necesario asumir que las muestras se distribuyen de forma normal o que proceden de poblaciones normales pero sí que ambas distribuciones son iguales.
- Igualdad de varianza entre grupos (homocedasticidad).

Tamaño del efecto D de Cohen

Los test estadísticos descritos anteriormente tienen como objetivo determinar si las poblaciones de las que proceden las muestras son distintas. El *p-value* generado, aunque fundamental, no aporta información sobre la magnitud de la diferencia, es en esto último donde el tamaño del efecto entra en juego.

La *d de Cohen* es una medida del tamaño del efecto basada en la diferencia de medias estandarizada. Informa de cuántas desviaciones típicas de diferencia hay entre los resultados de los dos grupos que se comparan. Su ecuación es:

$$d = \frac{|\text{diferencia de medias entre los grupos}|}{sd}$$

Aunque no existe una escala única, algunos valores de referencia empleados son:

- d ≤ 0.2 pequeño
- d ≥ 0.5 mediano
- d ≥ 0.8 grande

Si bien esta definición *d de Cohen* está implementada en varios paquetes de *R*, al emplear el valor absoluto de la diferencia, no aporta información sobre la dirección de la diferencia. Para este estudio, es interesante conocer qué grupo (líneas mutadas o no mutadas) es más sensible al fármaco, por lo que se ha implementado una versión que mantiene el signo de la diferencia.

Análisis de normalidad

Una de las aplicaciones más frecuentes del test de *Mann–Whitney–Wilcoxon* es su uso como alternativa al T-test cuando las muestras no proceden de poblaciones con distribución normal (asimetría o colas) o porque tienen un tamaño demasiado reducido para poder afirmarlo. Sin embargo, al ser un test no paramétrico, tiene la desventaja frente al T-test de poseer menor poder estadístico, es decir, menor capacidad para identificar diferencias reales.

Con la finalidad de ayudar al analista a decidir qué test es el adecuado, se incorpora un análisis de normalidad. Los análisis de normalidad, también llamados contrastes de normalidad, tienen como objetivo determinar cuánto difiere la distribución de los datos observados respecto a lo esperado si procediesen de una distribución normal con la misma media y desviación típica. Dos aproximaciones están disponibles:

- **Gráfico de cuantiles teóricos**: Consiste en comparar los cuantiles de la distribución observada con los cuantiles teóricos de una distribución normal con la misma media y desviación estándar que los datos. Cuanto más se aproximen los datos a una normal, más alineados están los puntos entorno a la recta.

- **Test de Shapiro-Wilk y Test de Kolmogorov-Smirnov**: Ambos test consideran como hipótesis nula que los datos sí proceden de una distribución normal y como hipótesis alternativa que no lo hacen. El *p-value* de estos test indica la probabilidad de obtener una distribución como la observada si los datos proceden realmente de una población con una distribución normal. Por defecto, se ha configurado para que se aplique el test de *Shapiro-Wilk* si el número de observaciones es menor o igual a 50 y el test de *Kolmogorov-Smirnov* si es mayor.

Es importante tener en cuenta que, al tratarse de *p-values*, cuanto mayor sea el tamaño de la muestra, más poder estadístico tienen y más fácil es encontrar evidencias en contra de la normalidad. Al mismo tiempo, cuanto mayor sea el tamaño de la muestra, menos sensibles son los métodos paramétricos a la falta de normalidad. Por esta razón, es importante no basar las conclusiones únicamente en el *p-value* del test, sino también considerar la representación gráfica y el tamaño de la muestra.

Tamaño mínimo de la muestra

Por defecto, la herramienta solo incorpora en el análisis genes para los que hay un mínimo de 5 líneas celulares mutadas y 5 líneas no mutadas. El objetivo de esta condición es conseguir un poder estadístico mínimo y así reducir el número de falsos positivos y negativos. Aun así, el usuario tiene libertad de modificar este valor en base a sus criterios.

Cluster de genes

Ocurre con frecuencia que, mutaciones que afectan a distintos genes que participan en una misma ruta de señalización, tienen el mismo resultado en la biología de la célula. Por ejemplo, véase el siguiente esquema (Figura 4) en el que 3 genes transmiten de forma secuencial una señal que termina llegando al núcleo de la célula haciendo que esta se divida.

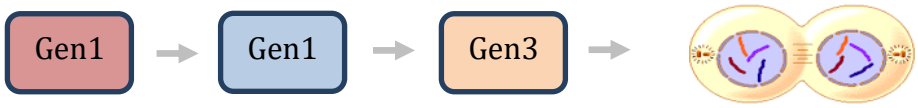


Fig.4. Representación de una ruta de señalización secuencial en la que participan 3 proteínas codificadas por 3 genes.

Si se detiene la actividad de cualquiera de los 3 genes, el resultado es el mismo, la señal deja de llegar al núcleo. En un caso de como este, el investigador podría estar interesado en contrastar las líneas celulares que tengan mutado alguno de estos genes frente a aquellas que no tienen mutado ninguno. Para proporcionar esta capacidad, se incorpora el concepto de *cluster*. El usuario puede especificar un listado de genes con el que se crea una nueva variable. Para cada una de las líneas celulares, el valor de esta variable es "Y", si al menos uno de los genes está mutado, y "N", si ninguno de los genes está mutado.

Corrección de error por comparaciones múltiples

Considerar el resultado de un test estadístico como significativo depende de si el *p-value* obtenido está por debajo de un límite conocido como nivel de significancia α que estable por el investigador. Si se considera un nivel de significancia α=0.05, existe un 5% de probabilidad de rechazar la hipótesis nula siendo esta verdadera, a esto se le conoce como error tipo I, *false positive rate* o probabilidad de falso positivo. Si se realiza un único test, esta probabilidad de falso positivo es relativamente baja. Sin embargo, si se realizan 100 test, se esperan en promedio 5 falsos positivos a pesar de que todas las hipótesis nulas fuesen ciertas. Si los test son independientes, la probabilidad de que al menos 1 de los 100 test rechace la hipótesis nula de forma incorrecta es del 99.4%.

El problema de las comparaciones múltiples se debe al incremento en el error de tipo I como consecuencia del uso repetido de test estadísticos. Si se realizan k comparaciones independientes, la probabilidad de que al menos ocurra un falso positivo, también conocida como *family-wise error rate* (*FWER*) viene dada por la ecuación:

$$FWER = 1 - (1 - \alpha)^k$$

Existen métodos que consiguen corregir la inflación del error de tipo I (*false positive rate*), es decir, la probabilidad de rechazar la hipótesis nula siendo esta cierta. Esta aproximación es útil cuando se emplea un número limitado de comparaciones. Para escenarios de *large-scale multiple testing* como son los estudios genómicos, en los que se realizan miles de test de forma simultánea, el resultado de estos métodos es demasiado conservador e impide que se detecten diferencias reales. Una alternativa es controlar el *false discovery rate*.

El f*alse discovery rat*e (*FDR*) se define como la probabilidad de que una hipótesis nula sea cierta habiendo sido rechazada por el test estadístico. El objetivo de controlar el *false discovery rate* es establecer un límite de significancia para un conjunto de test tal que, de entre todos los test considerados como significativos, la proporción de hipótesis nulas verdaderas (falsos positivos) no supere un determinado valor. Otra ventaja añadida es su fácil interpretación, por ejemplo, si un estudio publica resultados estadísticamente significativos para un *FDR* del 10%, el lector tiene la seguridad de que, como máximo, un 10% de los resultados considerados como significativos son realmente falsos positivos. En este documento se emplea el método *q-value* desarrollado por John D. Storey y Robert Tibshirani en 2003 para controlar el *FDR*.

Código

```
# CREACIÓN DEL CLUSTER DE GENES
# ================================================================================
# Si no se quiere incluir ningun cluster de genes, se le asigna el valor NA.
cluster_genes <- c("ABL1", "CCND1", "CREB1")

if(!is.na(cluster_genes)){
  nombre_cluster <- paste(c("KRAS", "RAF", "NRAS"), collapse = "_")
    temp <- datos_mutaciones %>% select(one_of(cluster_genes))
    cluster_status <- apply(X = temp,
                            MARGIN = 1,
                            FUN = function(x){ifelse(any(x == "Y"), "Y", "N")})

    datos_mutaciones[[nombre_cluster]] <- cluster_status
    print("Se ha añadido un cluster de genes en el análisis")
}
```

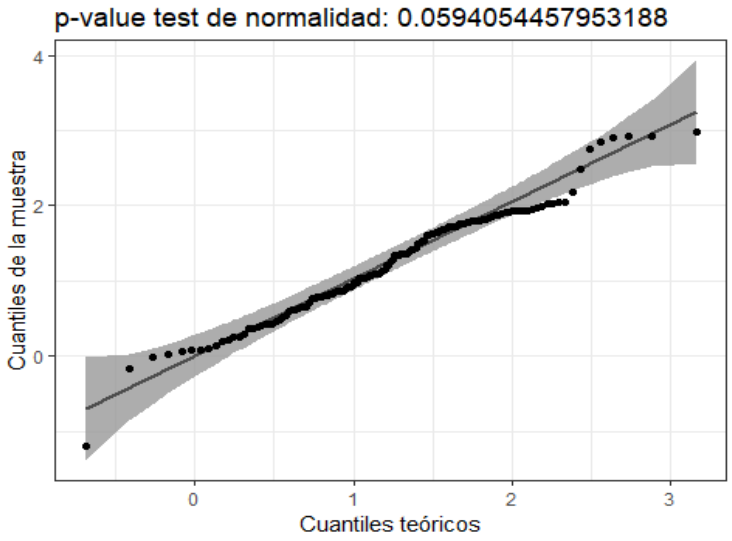
```
## [1] "Se ha añadido un cluster de genes en el análisis"
```

```
# FILTRAR GENES CON UN MINIMO DE MUESTRAS MUTADAS Y NO MUTADAS
# ================================================================================
n <- 5
paste("Solo genes con al menos", n, "líneas celulares mutadas y", n,
      "no mutadas, se incluyen en el análisis")
```

```
## [1] "Solo genes con al menos 5 líneas celulares mutadas y 5 no mutadas, se
incluyen en el análisis"
```

```
filtrar_por_n <- function(x, minimo){
  if (sum(x == "Y") >= minimo & sum(x == "N") >= minimo) {
    TRUE
  } else{
    FALSE
  }
}

cols <- datos_mutaciones %>%
        map_lgl(.f = filtrar_por_n, minimo = n)

# Las columnas sample_name y actividad no deben excluirse.
cols[c("sample_name", "actividad")] <- TRUE
```

```
# Se filtran las columnas en función del vector lógico cols. La tabla resultante
# contiene únicamente los genes para los que, al menos, hay n líneas mutadas y no
# mutadas.
datos_mutaciones <- datos_mutaciones[ ,cols]

# ANÁLISIS DE NORMALIDAD
#======================================================================================
library(nortest)
library(qqplotr)

if(nrow(datos_actividad) <= 50){
    test <- shapiro.test(x = log10(datos_actividad$actividad))
  p_value_test <- test$p.value
}else{
    test <- lillie.test(x = log10(datos_actividad$actividad))
  p_value_test <- test$p.value
}

ggplot(data = datos_actividad, mapping = aes(sample = log10(actividad))) +
 stat_qq_band() +
 stat_qq_line() +
 stat_qq_point() +
 labs(x = "Cuantiles teóricos", y = "Cuantiles de la muestra",
     title = paste("p-value test de normalidad:", p_value_test)) +
 theme_bw()
```



```r
# TEST ESTADÍSTICO
#==============================================================================

# Los dos test estadísticos implementados son ttest y utest.
test_estadistico <- "utest"

if(!(test_estadistico %in% c("ttest", "utest"))){
  stop("El test estádistico debe ser ttest o utest")
}

# Funciones para cada uno de los test y para el tamaño del efecto.
signed_effect_size_custom <- function(data, formula = log10(actividad)~status){
  # Esta función calcula el effect size con signo entre las medias de dos grupos.
  data <- model.frame(formula = formula, data = data)
  data_splited <- split(data[[1]], data[[2]])
  if(length(data_splited) != 2){
    stop("El número de grupos debe de ser 2")
  }
  a <- data_splited[[1]]
  b <- data_splited[[2]]
  delta <- mean(a) - mean(b)
  n_a <- length(a)
  n_b <- length(b)
  pooled_sd <- sqrt( ((n_a - 1)*(sd(a)^2) + (n_b - 1)*(sd(b)^2)) / (n_a + n_b -2 ))
  return(delta / pooled_sd)
}

ttest_custom <- function(df, formula = log10(actividad)~status){
  # Esta función devuelve el p_value de un t.test independiente.
  p_value <- t.test(formula = formula, data = df, paired = FALSE)$p.value
  return(p_value)
}

wilcoxtest_custom <- function(df, formula = log10(actividad)~status){
  # Esta función devuelve el p_value de un u-test.
  p_value <- wilcox.test(formula = formula, data = df, paired = FALSE)$p.value
  return(p_value)
}

# Para poder agrupar los datos por gen, se transforma la tabla ancha
# "datos_mutaciones" en formato largo (tidy).
datos_mutaciones_long <- gather(data = datos_mutaciones,
                                key = gene_name,
                                value = status, - c(sample_name, actividad))

# Se aplica el test estadístico seleccionado.
if(test_estadistico == "ttest"){
  resultados_test <- datos_mutaciones_long %>%
                     group_by(gene_name) %>%
```

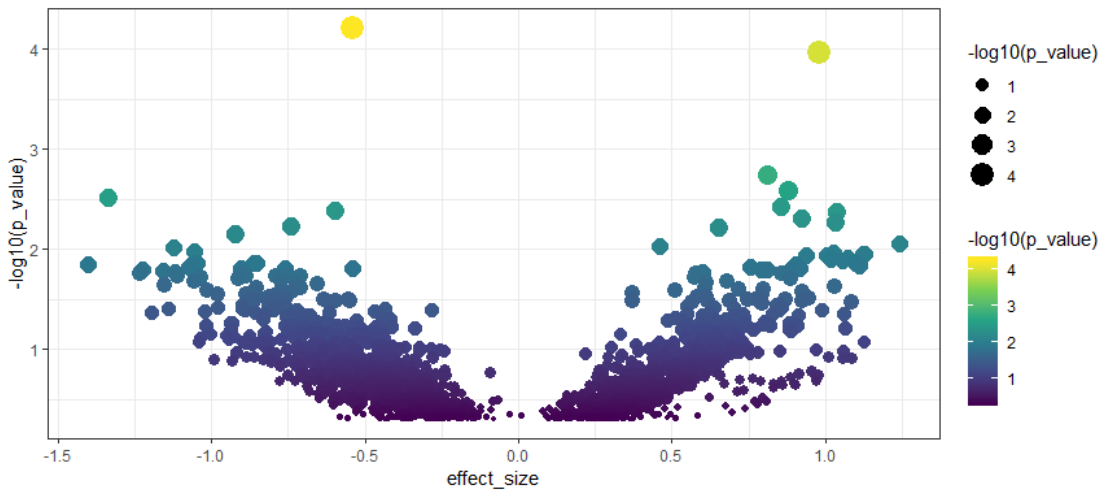
```
                    nest() %>%
                    mutate(p_value = map_dbl(.x = data,
                                             .f = ttest_custom),
                           effect_size = map_dbl(.x = data,
                                                 signed_effect_size_custom)) %>%
                    select(-data) %>%
                    arrange(p_value)
}

if(test_estadistico == "utest"){
  resultados_test <- datos_mutaciones_long %>%
                    group_by(gene_name) %>%
                    nest() %>%
                    mutate(p_value = map_dbl(.x = data,
                                             .f = wilcoxtest_custom),
                           effect_size = map_dbl(.x = data,
                                                 signed_effect_size_custom)) %>%
                    select(-data) %>%
                    arrange(p_value)
}

# CÁLCULO DEL Q-VALUE PARA CONTROLAR EL FDR
#==============================================================================
library(qvalue)
resultados_test$q_value <- qvalue(p = resultados_test$p_value)$qvalues
#resultados_test$fdr     <- p.adjust(p = resultados$p_value, method = "BH")

resultados_test %>% head()
```

```
## # A tibble: 6 x 4
##   gene_name   p_value effect_size q_value
##   <chr>         <dbl>       <dbl>   <dbl>
## 1 RB1       0.0000614      -0.538   0.248
## 2 SPTBN5    0.000110        0.979   0.248
## 3 KIAA1462  0.00179         0.812   1
## 4 DOT1L     0.00259         0.881   1
## 5 LAP3      0.00313        -1.33    1
## 6 RGPD3     0.00378         0.858   1
```

Resultados

Se muestran los resultados obtenidos en los test estadísticos mediante un gráfico de tipo *volcano*. El eje horizontal muestra el tamaño de efecto con signo y el eje vertical el logaritmo negativo del *p-value* (cuanto más negativo el logaritmo menor el *p-value*).

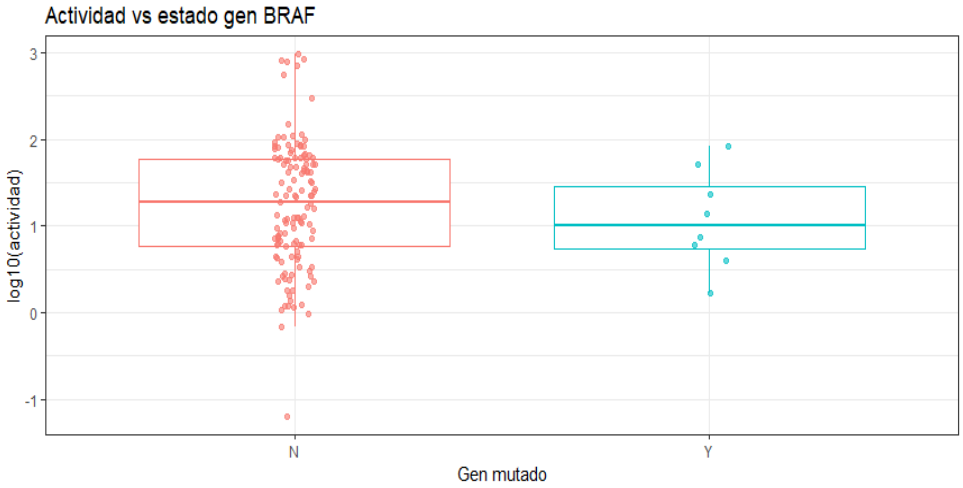
```
# library(plotly)
# library(DT)
# plot_ly(
#   # Se excluyen aquellos genes que cuyo p-value es > 0.5 para no saturar el
gráfico.
#   data = resultados_test[resultados_test$p_value < 0.5, ],
#   x = ~effect_size,
#   y = ~-log10(p_value),
#   size = ~-log10(p_value),
#   color = ~-log10(p_value),
#   key = ~gene_name,
#   #alpha = ~-log10(p_value)
#   text = ~gene_name,
#   hoverinfo = "text"
#   )
ggplot(
  # Se excluyen aquellos genes que cuyo p-value es > 0.5 para no saturar el
gráfico.
  data = resultados_test[resultados_test$p_value < 0.5, ],
  aes(x = effect_size, y = -log10(p_value), size = -log10(p_value),
      color = -log10(p_value))
    ) +
  geom_point() +
  scale_color_viridis_c() +
  theme_bw()
```



```r
head(resultados_test)
```

```
## # A tibble: 6 x 4
##   gene_name   p_value effect_size q_value
##   <chr>         <dbl>       <dbl>   <dbl>
## 1 RB1       0.0000614      -0.538   0.248
## 2 SPTBN5    0.000110        0.979   0.248
## 3 KIAA1462  0.00179         0.812   1
## 4 DOT1L     0.00259         0.881   1
## 5 LAP3      0.00313        -1.33    1
## 6 RGPD3     0.00378         0.858   1
```

```r
datos_mutaciones_long %>%
  filter(gene_name == "BRAF") %>%
  ggplot(aes(x = status, y = log10(actividad), color = status)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.05, alpha = 0.6) +
  labs(title = "Actividad vs estado gen BRAF",
       x = "Gen mutado") +
  theme_bw() +
  theme(legend.position = "none")
```



Empleando el test no paramétrico se ha comparado la actividad observada entre líneas mutadas y no mutadas para cada uno de los genes en los que se dispone, como mínimo, de 5 líneas por grupo. El gen con menor *p-value* de entre todos los analizados es *BRAF*, y su tamaño de efecto está por encima de 1. Tal y como cabía esperar acorde a los estudios publicados con este fármaco, las líneas procedentes de tumores con mutaciones en *BRAF* son más sensibles al tratamiento con *AZ628*, se requieren dosis más bajas para conseguir el efecto antiproliferativo.

Niveles de expresión

Que la expresión de un gen esté correlacionada con la respuesta ante la exposición de un fármaco significa que, la actividad observada, es mayor cuanto mayor es la expresión de dicho gen en el tumor (o lo opuesto, si la correlación es negativa). El análisis de esta hipótesis requiere de métodos estadísticos que cuantifiquen las relaciones observadas entre ambas variables (expresión y actividad) y permitan determinar si existe evidencia suficiente para afirmar que la relación es real. Dos test estadísticos se emplean a continuación para dar respuesta a esta pregunta, uno paramétrico (correlación de Pearson) y otro no paramétrico (correlación de Spearman). El proceso seguido en ambos es el mismo (Figura 5): para cada uno de los genes, se aplica un el test de correlación entre la actividad y la expresión en el conjunto de las líneas celulares y se obtiene un valor de *p-value*.

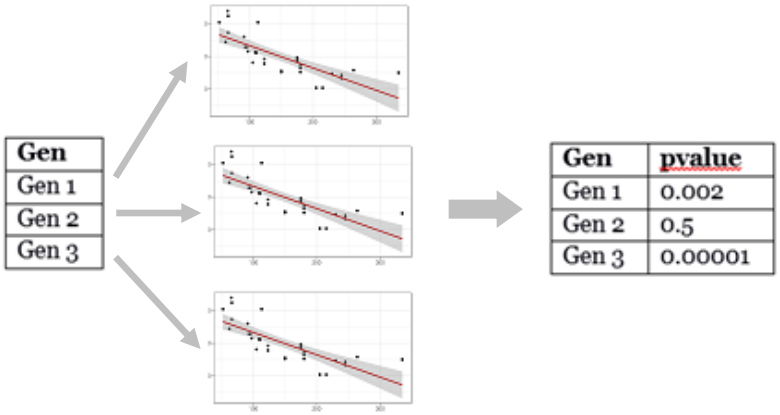


Fig.5. Esquema de las comparaciones múltiples con las que se calcula la significancia asociada a cada uno de los genes.

Correlación de Pearson

El coeficiente de correlación de *Pearson* es la covarianza estandarizada. Sus valores están siempre en el rango [-1, +1]. Siendo +1 una correlación positiva perfecta y -1 una correlación negativa perfecta. Para que el valor de la correlación de *Pearson* sea válido, es necesario que se cumplan una serie de condiciones, entre las que se encuentran:

- La relación que se quiere estudiar entre ambas variables es lineal (de lo contrario, el coeficiente de *Pearson* no la puede detectar).

- Las dos variables deben de ser cuantitativas.
- Normalidad: ambas variables se tienen que distribuir de forma normal.

Correlación de Spearman

El coeficiente de *Spearman* es el equivalente al coeficiente de *Pearson* pero con una previa transformación de los datos a rangos. Se emplea como alternativa cuando los valores son ordinales, o bien, cuando los valores son continuos pero no satisfacen la condición de normalidad requerida por el coeficiente de *Pearson* y se pueden ordenar transformándolos en rangos. Al trabajar con rangos, es menos sensible que *Pearson* a valores extremos. Existe una diferencia adicional con respecto a *Pearson*. El coeficiente de *Spearman* requiere que la relación entre las variables sea monótona, es decir, que cuando una variable crece la otra también lo hace o cuando una crece la otra decrece (que la tendencia sea constante). Este concepto no es exactamente el mismo que linealidad.

Además del valor obtenido para el coeficiente de correlación en cuestión, es necesario calcular su significancia. Solo si el *p-value* es significativo se puede aceptar que existe correlación y esta será de la magnitud que indique el coeficiente. Por muy cercano que sea el valor del coeficiente de correlación a +1 o -1, si no es significativo, se ha de interpretar que la correlación de ambas variables es nula ya que el valor observado se puede deber al azar.

Código

```
# TEST ESTADÍSTICO
#==============================================================================
# Los dos tipos de correlación disponibles son pearson y spearman.
test_estadistico <- "spearman"

if(!(test_estadistico %in% c("pearson", "spearman"))){
  stop("El test estádistico debe ser pearson o spearman")
}

# Funciones para cada uno de los test y para el tamaño del efecto.
cor_pearson <- function(df){
  # Esta función devuelve el p_value y r de un test de correlación de pearson.
  test <- cor.test(x = df[["actividad"]],
                   y = df[["expresion"]],
                   alternative = "two.sided",
```

```
                    method = "pearson",
                    conf.level = 0.95)
  r <- test$estimate
  p_value <- test$p.value
  return(tibble(r = r, p_value = p_value))
}

cor_spearman <- function(df){
  # Esta función devuelve el p_value y r de un test de correlación de spearman.
  test <- cor.test(x = df[["actividad"]],
                    y = df[["expresion"]],
                    alternative = "two.sided",
                    method = "spearman",
                    conf.level = 0.95)
  r <- test$estimate
  p_value <- test$p.value
  return(tibble(r = r, p_value = p_value))
}

# Para porder agrupar los datos por gen, se transforma la tabla ancha
# "datos_mutaciones" en formato largo (tidy).
datos_expresion_long <- gather(data = datos_expresion,
                               key  = gene_name,
                               value = expresion,
                               -c(sample_name, actividad))
datos_expresion_long <- na.omit(datos_expresion_long)

# Se aplica el test estadístico seleccionado.
if(test_estadistico == "pearson"){
  resultados_cor <-  datos_expresion_long %>%
                     group_by(gene_name) %>%
                     nest() %>%
                     mutate(p_value = map(.x = data, .f = cor_pearson)) %>%
                     select(-data) %>%
                     unnest()
}

if(test_estadistico == "spearman"){
  resultados_cor <-  datos_expresion_long %>%
                     group_by(gene_name) %>%
                     nest() %>%
                     mutate(p_value = map(.x = data, .f = cor_pearson)) %>%
                     select(-data) %>%
                     unnest()
}

resultados_cor <- resultados_cor %>% arrange(p_value)
```

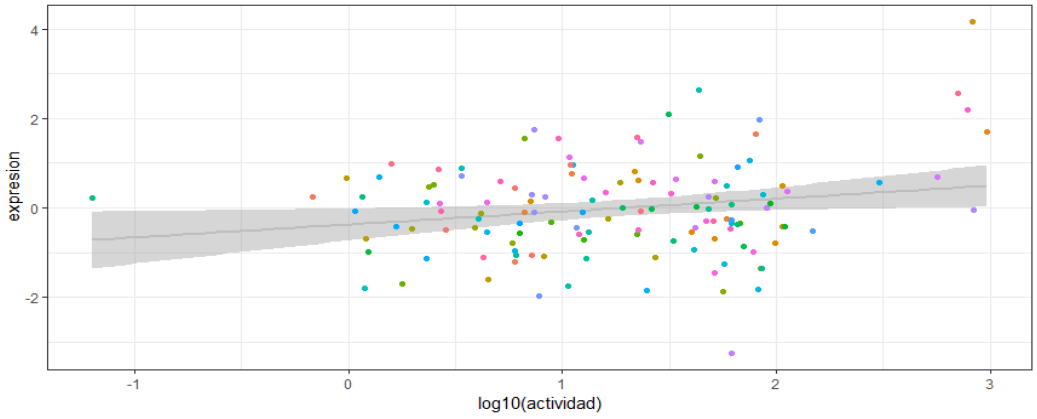
```
# CÁLCULO DEL Q-VALUE PARA CONTROLAR EL FDR
#==============================================================================
library(qvalue)
resultados_cor$q_value <- qvalue(p = resultados_cor$p_value)$qvalues
#resultados_test$fdr     <- p.adjust(p = resultados$p_value, method = "BH")
resultados_cor %>% head()
```

```
## # A tibble: 6 x 4
##   gene_name      r    p_value q_value
##   <chr>      <dbl>      <dbl>   <dbl>
## 1 LRRC29     0.385 0.00000528  0.0865
## 2 TRPT1      0.340 0.0000679   0.298
## 3 KRT33B     0.334 0.0000901   0.298
## 4 ZFPL1      0.333 0.0000952   0.298
## 5 ABCB10    -0.331 0.000107    0.298
## 6 DPP9       0.330 0.000109    0.298
```

Resultados

Se muestran los resultados obtenidos para el gen con menor *p-value* mediante un gráfico de dispersión.

```
#library(plotly)
top_gen <- resultados_cor$gene_name[1]
p_correlacion <-  datos_expresion_long %>% filter(gene_name == top_gen) %>%
                  ggplot(aes(x = log10(actividad), y = expresion,
                             text = sample_name, group = 1)) +
                             geom_smooth(method = "lm", color = "gray") +
                             geom_point(aes(color = sample_name)) +
                             theme_bw() + theme(legend.position = "none")
p_correlacion
```




```
#ggplotly(p = p_correlacion, tooltip = "text")
```

Gene set enrichment analysis

En el ámbito de biología, el término *pathway* hace referencia al conjunto de genes que están relacionados con una función biológica específica, por ejemplo, la división celular. Una forma de identificar qué procesos del funcionamiento de una célula están asociados con la respuesta a un fármaco consiste en estudiar si los genes identificados de forma individual (mutaciones o correlación) forman parte de un mismo *pathway*.

Dado que existen cientos de *pathways* distintos y cada uno puede estar formado por varias decenas de genes, este análisis requiere de métodos estadísticos que calculen la probabilidad (*p-value*) de que la asociación entre un conjunto de genes y un determinado *pathway* se deba únicamente al azar. Una forma sencilla de hacerlo es mediante un test exacto de Fisher que tenga en cuenta el número de genes candidatos, el número de genes que forman cada *pathway*, el número de genes coincidentes y el número total de posibles genes. Al tratarse de múltiples test, los *p-values* se corrigen para controlar el *FDR*.

Como ejemplo ilustrativo se emplean 6 genes que participan en el *pathway mTOR* (*AKT1*, *CCNE1*, *DEPTOR*, *MAP2K1*, *PLD1*, *RPTOR*).

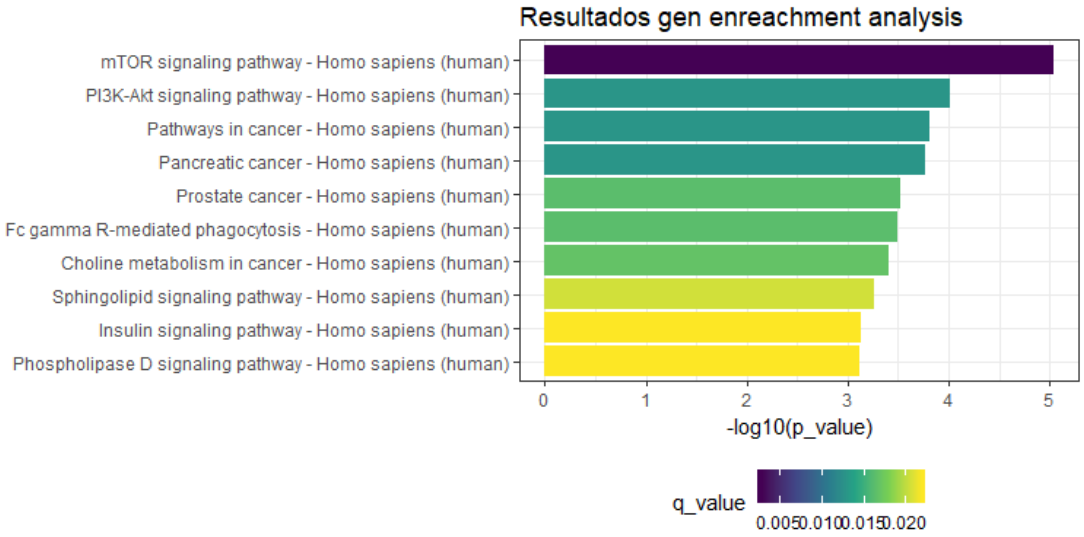
Código

```
# TEST ESTADÍSTICO
#==============================================================================
total_genes <- 20000 # Aproximadamente todo el genoma humano.

# Ejemplo de set de genes para el análisis.
genes <- c("AKT1", "CCNE1", "DEPTOR", "MAP2K1", "PLD1", "RPTOR")

# Se emplean únicamente pathways obtenidos de KEGG.
datos_pathways <- datos_pathways %>% filter(source == "KEGG")

# Función para calcular el test de Fisher.
fisher_test_ease <- function(pathway, genes_interes, total_genes) {
  # Esta función devuelve el p-value de un test de fisher con corrección de EASE.
  pathway <- str_split(string = pathway, pattern = ",")[[1]]
  n_genes_interes        <- length(genes_interes)
  n_genes_pathway        <- length(pathway)
  n_genes_comunes        <- length(intersect(pathway, genes_interes))

  # Corrección de EASE: La corrección de EASE le resta 1 al número de eventos
  # comunes entre el pathway y el listado de genes de interés.
```

```
  if (n_genes_comunes > 0) {
    n_genes_comunes <- n_genes_comunes - 1
  }

  n_genes_no_comunes     <- n_genes_interes - n_genes_comunes
  n_genes_no_en_pathway  <- total_genes - n_genes_pathway
  tabla <- matrix(data = c(n_genes_comunes, n_genes_no_comunes, n_genes_pathway,
                           n_genes_no_en_pathway), nrow = 2)

  return(fisher.test(tabla, alternative = "greater")$p.value)
}

# Se calcula el resultado del test estadístico para cada pathway.
resultados_pathways <- datos_pathways %>%
                       mutate(p_value = map_dbl(.x = hgnc_symbol_ids,
                                                .f = fisher_test_ease,
                                                genes_interes = genes,
                                                total_genes = total_genes
                                               )
                       )
resultados_pathways <- resultados_pathways %>% arrange(p_value)
resultados_pathways$q_value <- qvalue(p = resultados_pathways$p_value)$qvalues
resultados_pathways %>% head()
```

```
## # A tibble: 6 x 6
##   pathway         source hgnc_symbol_ids           n_genes p_value q_value
##   <chr>           <chr>  <chr>                       <int>   <dbl>   <dbl>
## 1 mTOR signaling~ KEGG   STK11,IKBKB,AKT1,AKT2,AK~     153 9.14e-6 0.00275
## 2 PI3K-Akt signa~ KEGG   BCL2L1,PCK2,PCK1,GHR,PPP~     340 9.62e-5 0.0127
## 3 Pathways in ca~ KEGG   CRKL,PTGER4,PIK3CA,PIK3C~     397 1.52e-4 0.0127
## 4 Pancreatic can~ KEGG   BCL2L1,TGFB1,TGFB2,TGFB3~      66 1.69e-4 0.0127
## 5 Prostate cance~ KEGG   IKBKB,AKT1,AKT2,AKT3,ARA~      89 3.03e-4 0.0162
## 6 Fc gamma R-med~ KEGG   CRKL,AKT1,AKT2,AKT3,FCGR~      92 3.24e-4 0.0162
```

```
ggplot(data = resultados_pathways %>% head(10),
       aes(x = reorder(pathway,-log10(p_value)),
           y = -log10(p_value),
           fill = q_value)) +
       geom_col() +
       scale_fill_viridis_c() +
       labs(title = "Resultados gen enreachment analysis",
            x = "") +
       coord_flip() +
       theme_bw() +
       theme(legend.position = "bottom")
```



Los resultados del análisis muestran que sí se ha detectado correctamente el *pathway* en el que participan mayoritariamente el listado de genes. Aunque este es listado de prueba, podría emplearse cualquier otro que sea de interés para el investigador.

Observaciones

- A la hora de interpretar los resultados es importante tener en cuenta que, al tratarse de contrastes estadísticos, el tamaño de las muestras es crítico para alcanzar un poder estadístico mínimo que permita detectar diferencias. Además, los resultados obtenidos en este tipo de análisis no deben interpretarse como una prueba total de relación causa efecto, más bien deben de utilizarse como guía para decidir qué experimentos o líneas de investigación vale la pena continuar.

- La idea de "un biomarcador un fármaco" es una simplificación con la que se intenta estratificar a la población de pacientes como sensibles o resistentes a un tratamiento, sin embargo, no hay que olvidar que, incluso dentro de cada grupo, siempre existe variabilidad.

- Aunque las mutaciones y expresión genética son claras fuentes de biomarcadores, existen otras modificaciones que ocurren a lo largo de la proliferación tumoral, como por ejemplo las modificaciones post-translacionales. Si se dispone de este tipo de información, es sencillo incorporarla.

- Los test estadísticos mostrados en este documento analizan el impacto de cada gen de forma individual. En un sistema tan complejo como es la biología celular, cabe esperar que muchos procesos sean el resultado de interacciones múltiples. El desarrollo de métodos que contemplen interacciones (múltiples mutaciones, mutaciones y expresión, etc) puede suponer un avance muy importante para una mejor estratificación de los pacientes.

Bibliografía

Liquid handling devices in drug discovery: When, what, why? Article in European Pharmaceutical Review December 2013

Genomics-Driven Oncology: Framework for an Emerging Paradigm, Levi A. Garraway, Clin Oncol 31:1806-1814. © 2013 by American Society of Clinical Oncology

Lessons learned from the application of whole-genome analysis to the treatment of patients with advanced cancers, Laskin et al. 2015 Cold Spring Harb Mol Case Stud 1: a000570

A census human cancer genes Futreal et al, 2004 Nat Rev Cancer

The COSMIC (Catalogue of Somatic Mutations in Cancer) database and website British Journal of Cancer (2004) 91, 355 – 358

A comprehensive transcriptional portrait of human cancer cell lines Nature Biotechnology volume 33, pages 306–312 (2015)

Pharmacogenomic agreement between two cancer cell line data sets Nature volume 528, pages 84–87 (03 December 2015)

The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity Nature. 2012 Dec 13;492(7428):290

Cell Panel Profiling Reveals Conserved Therapeutic Clusters and Differentiates the Mechanism of Action of Different PI3K/mTOR, Aurora Kinase and EZH2 Inhibitors Mol Cancer Ther. 2016 Dec.

Covell DG (2012) Integrating Constitutive Gene Expression and Chemoactivity: Mining the NCI60 Anticancer Screen. PLoS ONE 7(10): e44631.

Systematic identification of genomic markers of drug sensitivity in cancer cells. Nature. 2012 Mar 28;483(7391):570-5.

Machine Learning in Genomic Medicine: A Review of Computational Problems and Data Sets, Proceedings of the IEEE |Vol.104,No.1,January2016

Overview of biomarkers in disease, drug discovery and development, Drug Discovery World Spring 2005

https://www.nature.com/subjects/biomarkers

Precision medicine needs pioneering clinical bioinformaticians, Gonzalo Gómez-López, Joaquín Dopazo, Juan C. Cigudosa, Alfonso Valencia and Fátima Al-Shahrour Briefings in Bioinformatics, 2017, 1–15

Cancer Cell Line Encyclopedia (CCLE)

COSMIC Cell Lines Project

The Genomics of Drug Sensitivity in Cancer Project, Cancer Genome Project at the Wellcome Sanger Institute (UK) and the Center for Molecular Therapeutics, Massachusetts General Hospital Cancer Center (USA).

An Introduction to Statistical Learning with Applications in R, Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani

Applied Predictive Modeling By Max Kuhn and Kjell Johnson

Data Mining for Bioinformatics, Sumeet Dua, Pradeep Chowriappa

Druker BJ, et al. Five-year follow-up of patients receiving imatinib for chronic myeloid leukemia. N Engl J Med. 2006;355:2408–2417.

Dose-Response Analysis Using R Ritz C, Baty F, Streibig JC, Gerhard D (2015)

Dose-Response Analysis Using R. PLOS ONE 10(12): e0146021.

Points of significance: Significance, P values and t-tests, Martin Krzywinski & Naomi Altman, Nature Methods volume 10, pages 1041–1042 (2013)

Points of significance: Comparing samples—part I, Martin Krzywinski & Naomi Altman, Nature Methods volume 11, pages 215–216 (2014)

John D. Storey with contributions from Andrew J. Bass, Alan Dabney and David Robinson (2015). qvalue: Q-value estimation for false discovery rate control. R package version 2.10.0. http://github.com/jdstorey/qvalue.

Khazak V, Astsaturov I, Serebriiskii IG, Golemis EA. Selective Raf Inhibition in Cancer Therapy. Expert opinion on therapeutic targets. 2007;11(12):1587-1609. doi:10.1517/14728222.11.12.1587.

Bollard J, Miguela V, Ruiz de Galarreta M, et al. Palbociclib (PD-0332991), a selective CDK4/6 inhibitor, restricts tumour growth in preclinical models of hepatocellular carcinoma. Gut. 2017;66(7):1286-1296. doi:10.1136/gutjnl-2016-312268.