Benchmark de algoritmos para la detección de CNVs con resolución de exón a partir de datos NGS de panel

José Marcos Moreno Cabrera
Máster universitario en Bioinformática y bioestadística UOC-UB
Bioinformática traslacional

Bernat Gel Moreno
Carles Ventura Royo

2 de enero de 2018

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Benchmark de algoritmos para la detección de CNVs con resolución de exón a partir de datos NGS de panel*
Nombre del autor:	*José Marcos Moreno Cabrera*
Nombre del consultor/a:	*Bernat Gel Moreno*
Nombre del PRA:	*Carles Ventura Royo*
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	*Máster universitario en Bioinformática y bioestadística UOC-UB*
Área del Trabajo Final:	*Bioinformática traslacional*
Idioma del trabajo:	*Castellano*
Palabras clave	*CNV, NGS, benchmark*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Next Generation Sequencing (NGS) es una tecnología fundamental en el diagnóstico genético de enfermedades hereditarias, ampliamente utilizada para la detección de pequeñas variantes. Sin embargo, la detección de variantes más grandes como las variaciones en el número de copias (*copy number variants*, CNVs) sigue siendo un reto. La técnica más empleada para la detección de las CNVs es *multiplex ligation-dependent probe amplification* (MLPA), método costoso y que exige tiempo para llevarlo a cabo. Disponer de una herramienta de screening que funcionara a partir de datos NGS permitiría reducir el número de MLPAs necesarios para detectar CNVs, con el consiguiente ahorro de recursos.

La mayoría de los algoritmos de detección de CNVs funcionan bien detectando variaciones grandes (del orden de megabases) pero no con variaciones pequeñas que afecten a uno o pocos exones. Además, estos algoritmos suelen estar diseñados para trabajar con todo el genoma o exoma, no para realizar la detección a partir de datos de panel (subexoma).

En este trabajo se ha realizado un benchmark para algoritmos CNV que han mostrado un buen funcionamiento con datos NGS de panel a resolución de exón: DECoN, CoNVaDING y panelcn.MOPS. Estos algoritmos se han evaluado sobre un dataset público y validado de 96 muestras. Además, cada algoritmo ha sido optimizado mediante un algoritmo de optimización para mejorar la sensibilidad y especificidad que ofrecen los valores por defecto.

Los resultados mostraron que panelcn.MOPS alcanzó una sensibilidad

suficiente para poder ser usado como técnica de screening previa a la validación por MLPA.

Abstract (in English, 250 words or less):

Next Generation Sequencing (NGS) is a key technology for detecting small variants in the genetic diagnostics of hereditary diseases. However, detection of larger variants as copy number variants (CNVs) from NGS data remains a challenge. The most used technique for CNV detection is multiplex ligation-dependent probe amplification (MLPA), which implies important costs and time. Finding a screening technique would allow us to decrease the number of necessary MLPAs, and therefore, resources may be saved.

Most CNV calling algorithms perform well when calling large CNVs (in the order of megabases) but are not able to reliably detect small CNVs affecting only one or few exons. In addition, most of them are designed to work with whole exome or whole genome data and have problems with the more sparsed data generated by NGS panels.

In this work, it has been perfomed a benchmark of CNV calling algortihms which have shown to perform well with NGS data at single exon resolution: DECoN, CoNVaDING and panelcn.MOPS. These algorithms have been evaluated over a publicy available and validated dataset of 96 samples. Moreover, each algorithm has been optimized using an optimization algorithm to improve sensitivity and specificity offered by default parameter values.

Results revealed panelcn.MOPS reached sufficient sensitivity in the dataset to be used for CNV screening prior to MLPA validation.

Índice

1. INTRODUCCIÓN 5

1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO 5
1.2 OBJETIVOS DEL TRABAJO 7
1.2.1 Objetivo general 7
1.2.2 Objetivos específicos 7
1.3 ENFOQUE Y MÉTODO SEGUIDO 7
1.4 PLANIFICACIÓN DEL TRABAJO 8
1.4.1 Recursos empleados 8
1.4.2 Planificación temporal 8
1.5 BREVE SUMARIO DE PRODUCTOS OBTENIDOS 10
1.6 BREVE DESCRIPCIÓN DE LOS OTROS CAPÍTULOS DE LA MEMORIA 11
1.7 AGRADECIMIENTOS 11

2. BENCHMARK, LA ESTRATEGIA PARA DETERMINAR EL MEJOR ALGORITMO 12

2.1 ESTRUCTURA DEL CÓDIGO 13
2.1.1 Ficheros datasets.yaml y algorithms.yaml 14
2.1.2 Fichero runBenchmark.sh 14
2.1.3 Ficheros .yaml y .r de cada algoritmo 15
2.1.4 Ficheros summary.r y cnvStats.r 15

3. ALGORITMOS A COMPARAR 16

3.1 DECON 16
3.2 CONVADING 17
3.3 PANELCN.MOPS 18

4. DATATET VALIDADO 20

4.1 DESCARGA Y PUESTA A PUNTO DEL DATASET 20
4.1.1 Descarga del dataset 20
4.1.2 Desencriptado de los fastqs 21
4.1.3 Alineamiento: de fastqs a ficheros BAM 21
4.1.4 BED file 22

5. OPTIMIZACIÓN 23

5.1 ESTRATEGIA 23
5.2 OPTIMIZACIÓN SOBRE UN SUBCONJUNTO 24
5.3 ALGORITMO GREEDY 24
5.3.1 Greedy básico 24
5.3.2 Greedy con diferentes puntos de inicio 26
5.3.3 Greedy con diferentes puntos de inicio y heurística modificada 28

6. RESULTADOS 30

6.1 RESULTADOS CON VALORES POR DEFECTO 30
6.2 RESULTADOS CON VALORES OPTIMIZADOS 30
6.2.1 Heurística agresiva 31
6.2.2 Heurística balanceada 32
6.2.3 Heurística conservadora 34

7. CONCLUSIONES 36

7.1 CONCLUSIONES GENERALES 36

7.2 OBJETIVOS PLANTEADOS Y PLANIFICACIÓN 37
7.3 LÍNEAS DE TRABAJO FUTURO 37

8. GLOSARIO 39

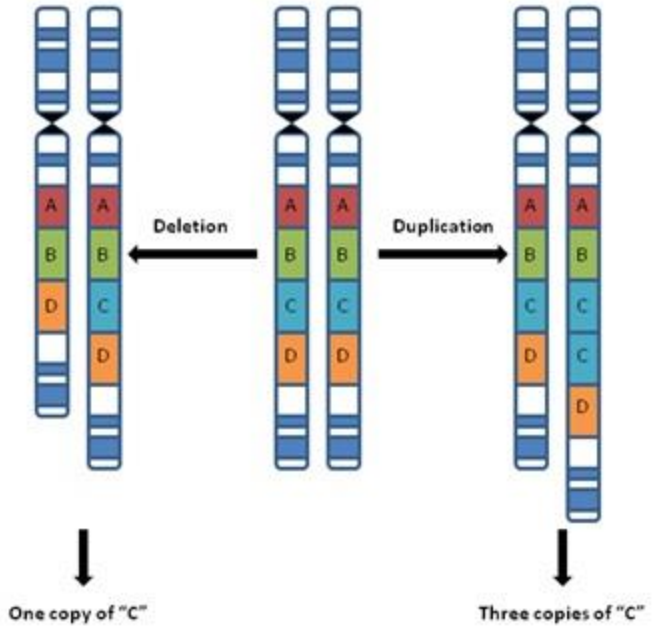
9. BIBLIOGRAFÍA 40

1. Introducción

1.1 Contexto y justificación del Trabajo

La tecnología *next generation sequencing* (NGS) ha significado un extraordinario incremento en la capacidad de secuenciación y mejora de los costes asociados respecto a Sanger, cambiando el paradigma del diagnóstico genético de enfermedades hereditarias [1]. De los aproximadamente 20000 genes que conforman el genoma humano, es necesario el estudio de un pequeño grupo de ellos para establecer el diagnóstico de una determinada enfermedad hereditaria. En concreto, resulta de especial interés el análisis de determinados genes cuya mutación o variación pueda significar un impacto funcional en la proteína codificada que guarda relación con la enfermedad que se quiere diagnosticar.

Estos genes pueden presentar dicha variación a pequeña escala, como la variación de un nucleótido, ya sea en forma de deleción, inserción o cambio de una base. Y las mencionadas variaciones también pueden producirse a una escala superior, como es la variación en el número de copias (*copy number variation*, CNV). Esta variación en el número de copias suele definirse como una deleción o duplicación de regiones mayores de 50 pares de bases [2]. Con frecuencia las CNVs a los que nos referimos comprenden al menos uno o más exones, pudiendo llegar a un gen entero o una porción significativa del cromosoma.



Representación de un evento CNV por deleción o duplicación [6]

Aproximadamente el 12% del genoma humano posee alguna CNV, la cual puede ser o no patogénica [3]. Para realizar el diagnóstico genético de una

enfermedad hereditaria es necesario aplicar alguna técnica fiable de detección de CNV. Las dos tecnologías más empleadas son *array comparative genomic hybridization* (aCGH) y *multiplex ligation-dependent probe amplification* (MLPA). Si bien son opciones fiables, ambas exigen una cantidad significativa de recursos económicos y tiempo para llevarlas a cabo. **Disponer de una alternativa a estas técnicas sería,** por tanto, **una oportunidad de reducir costes y simplificar el proceso** de diagnóstico en el laboratorio.

En los últimos años han surgido herramientas para la detección de CNVs a partir de los datos secuenciados mediante NGS. Existe un amplio espectro de algoritmos para detectar CNVs, conformado por aproximadamente 40 herramientas con diferente alcance. La mayoría de los algoritmos de detección de CNVs funcionan bien detectando variaciones grandes (del orden de megabases) pero no con variaciones pequeñas que afecten a uno o pocos exones. Además, estos algoritmos suelen estar diseñados para trabajar con todo el genoma (*whole genome*) o exoma (*whole exome*), no para realizar la detección a partir de datos de panel (subexoma), mucho más esparcidos.

El reto es, por tanto, **determinar un algoritmo que sea capaz de detectar CNVs a nivel de exón y a partir de datos NGS de panel**. De esta manera, si pudiésemos optimizar algún algoritmo para que en el contexto explicado alcanzara una sensibilidad del 100%, podríamos emplearlo como técnica de screening en la rutina de diagnóstico, previa a la posterior validación mediante aCGH o MLPA. Una cierta cantidad de falsos positivos sería tolerable dado que a posteriori es posible confirmar los positivos mediante MLPA.

El objetivo mencionado en el apartado anterior no está planteado sobre un escenario imaginario. Al contrario, **este reto ha nacido con la intención de ser aplicado a un escenario real: el diagnóstico genético de cáncer hereditario que actualmente se realiza en el Institut Catalá de Oncologia (ICO) y en el Institut Germans Trias i Pujol (IGTP)** - en el Laboratori de Recerca Traslacional 2 y Unidad de Diagnóstico Genético del Cáncer Hereditario respectivamente -. De obtener resultados esperados, y tras posteriores trabajos que confirmen dichos resultados, **la buscada técnica de screening podría ser implementada para el diagnóstico real de pacientes**, con un ahorro significativo en tiempo y coste.

En definitiva, se ha elegido este tema por su utilidad clínica y traslacional. Se trata de un campo donde existe margen para obtener conclusiones que sean útiles a la comunidad científica y médica que trabaja en el campo del diagnóstico genético de enfermedades hereditarias.

1.2 Objetivos del Trabajo

1.2.1 Objetivo general

El objetivo del presente TFM es comparar algoritmos de detección de CNVs sobre un dataset validado de muestras secuenciadas mediante NGS, y evaluar el rendimiento de cada algoritmo en cuanto a sensibilidad y especificidad con el objetivo de determinar el algoritmo más adecuado, si lo hubiera, para ser aplicado a la rutina de diagnóstico en el laboratorio.

1.2.2 Objetivos específicos

En concreto, el anterior objetivo general se desarrolla en los siguientes objetivos específicos:

1. Seleccionar y obtener acceso a un conjunto de datos de panel NGS, accesible públicamente, para el cual se tengan los resultados CNV validados.
2. Determinar qué algoritmos, de toda la bibliografía publicada, se ajustan al contexto explicitado en este TFM: detección de CNV a nivel de exón a partir de datos de panel (NGS). Seleccionar al menos tres candidatos para su posterior evaluación.
3. Encontrar los valores de los parámetros que maximizan la sensibilidad (primer criterio) y especificidad (segundo criterio) de cada algoritmo.
4. Determinar el algoritmo que muestra un mejor rendimiento en el dataset analizado y evaluar la conveniencia de aplicarlo en la rutina del laboratorio.

1.3 Enfoque y método seguido

Como se ha comentado anteriormente, la necesidad que pretende cubrir el presente trabajo es la de disponer de una técnica que permita hacer un screening previo a la validación por MLPA para determinar la existencia de CNVs en un contexto de diagnóstico genético. Por tanto, tenemos dos vías principales para enfocar una solución:

1. Desarrollar una algoritmo que tenga en cuenta el contexto específico, con el objetivo de alcanzar una sensibilidad de 100% para datos NGS de panel y llegando a resolución de exón. Este desarrollo podría realizarse desde cero o bien partiendo de los múltiples algoritmos de software libre existentes en la literatura científica.
2. Hacer un análisis preciso a partir de un benchmark de los algoritmos que hayan mostrado comportarse mejor en las condiciones descritas. Disponiendo de un dataset validado por una técnica fiable como MLPA, se puede comparar la sensibilidad y especificidad alcanzada por los algoritmos estudiados contra este dataset.

En toda la literatura científica existe un solo benchmark que compara algoritmos CNV a partir de datos de NGS de panel y llegando a resolución de

exón. Esta comparativa se encuentra en el propio artículo donde se presenta panelcn.MOPS [2]. Teniendo en cuenta la escasez de trabajos que comparan diferentes algoritmos en el citado contexto, encontramos de interés optar por la segunda alternativa de enfoque, y por tanto realizar un benchmark propio sobre un dataset diferente al empleado por panelcn.MOPS, y con una selección de algoritmos propia.

1.4 Planificación del Trabajo

1.4.1 Recursos empleados

En este apartado cabe reseñar el uso de recursos de almacenamiento y computación facilitados por el Institut Germans Trias i Pujol (IGTP). En especial, ha sido clave para el desarrollo de este proyecto disponer de un clúster computación con 240 cores. Su uso ha sido imprescindible para los procesos de optimización de cada algoritmo donde cientos de ejecuciones han sido necesarias.

1.4.2 Planificación temporal

Los cuatro objetivos detallados en el apartado 1.2 se han llevado a cabo siguiendo la siguiente estimación de **tareas** descrita en la PEC1:

0. Lectura y formación sobre CNV y datos de panel NGS. (30 horas)
1. Selección de un dataset.
   a) Búsqueda de un dataset sobre el que se realizará el benchmark. El dataset debe ser de al menos 30 muestras, con datos NGS de panel y con un fichero donde se identifiquen los CNV encontrados por una técnica ortogonal. (20 horas)
   b) Acceso al dataset. El ADN secuenciado es información sensible, por lo que con frecuencia estos conjuntos de datos requieren una solicitud de acceso justificando su uso. (5 horas)
   c) Puesta a punto del dataset. Adaptar los datos al formato exacto que requieren los algoritmos a evaluar. (25 horas)
2. Selección de los 3 algoritmos CNV.
   a) Búsqueda y revisión de papers existentes en los que se exponga el uso de algoritmos CNV para datos de panel en un contexto de diagnóstico. (40 horas)
   b) Selección de 3 algoritmos que puedan ser buenos candidatos para la comparativa a realizar. (10 horas)
   c) Descarga y puesta a punto de los 3 algoritmos seleccionados. Probar su funcionamiento para algún ejemplo simple. (40 horas)
3. Construir un benchmark que compare los 3 algoritmos sobre el dataset.
   a) Construir los scripts que automaticen la ejecución de los 3 algoritmos sobre el dataset. (40 horas)

b) Ejecutar cada algoritmo con diferentes valores de parámetros para encontrar cuáles maximizan la sensibilidad de dicho algoritmo. (20 horas)

4. Analizar los resultados obtenidos para determinar el algoritmo óptimo para el diagnóstico en el dataset analizado. (10 horas)

Además, una vez finalizadas las fases de desarrollo, también se programó la realización de las siguientes tareas:

- Redacción de la memoria del TFM. (45 horas)
- Elaboración de la presentación. (15 horas)

La siguiente tabla muestra el **calendario** con el desglose de tareas programadas con las fechas que comprenden y la correspondiente fase de desarrollo.

Tarea	Fechas	Fase desarrollo
0	17/10 al 24/10	1
1a	25/10 al 29/10	1
1b	30/10 al 1/11	1
1c	2/11 al 7/11	1
2a	8/11 al 17/11	1
2b	18/11 al 20/11	1
2c	21/11 al 30/11	2
3a	1/12 al 10/12	2
3b	10/12 al 18/12	2
4	19/12 al 21/12	
Redacción memoria del TFM	7/11 al 2/01	
Elaboración de la presentación	3/01 al 10/01	

Y de manera esquemática, a continuación se muestra el un **diagrama de Gantt** con la relación de dependencia entre las tareas.

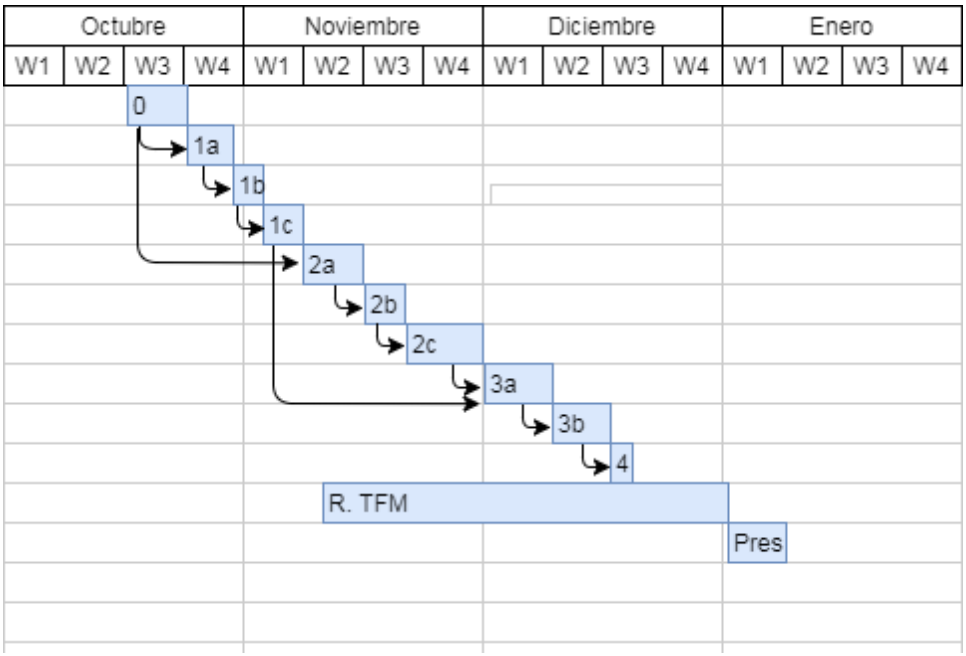


Diagrama de Gantt con la planificación temporal

De acuerdo a los objetivos específicos planteados en el apartado 1.2.2, destacamos los siguientes **hitos** programados:

- 1/11: Se ha obtenido acceso y se ha descargado un dataset público.
- 7/11: Se ha adaptado el formato del dataset para su posterior uso.
- 20/11: Se dispone de una lista de 3 algoritmos para probar su rendimiento.
- 30/11: Se ha instalado y configurado los 3 algoritmos para su ejecución.
- 10/12: Se tiene el código capaz de lanzar cada algoritmo contra el dataset.
- 18/12: Se tienen los resultados de la ejecución de los algoritmos con diferentes valores de parámetros.

1.5 Breve sumario de productos obtenidos

El producto obtenido se puede desglosar en las siguientes partes:

- La presente memoria del Trabajo Final de Máster, que incluye los resultados de los algoritmos contra el dataset validado, tanto con los valores por defecto para los parámetros como con los valores optimizados.
- El código que descarga y adapta el dataset validado:
  - Script que descargan el dataset y adapta el formato a .bam (descrypt.sh y convertToBam.sh).
- El código que automatiza el benchmark de los algoritmos y genera los resultados de la comparación entre algoritmos:
  - runBenchmark.sh: lanza la ejecución de los algoritmos contra los datasets.
  - Ficheros dataset.yaml y algorithms.yaml.

    - Carpeta algorithms con el script R que ejecuta cada algoritmo, junto con el yaml de configuración.
    - Carpeta utils con un conjunto de utilidades R empleadas en distintas partes del código.
- El código que optimiza los valores de los parámetros de cada algoritmo en función de su sensibilidad y especificidad:
    - Fichero de configuración yaml con los parámetros disponibles.
    - Fichero optimizador en R que gestiona la ejecución del algoritmo con diferentes valores de parámetros para encontrar el óptimo.
    - Fichero R que calcula los CNV para unos valores de parámetros dados.
    - Script .sh que lanza el job contra el clúster.
    - En la carpeta utils, fichero optimizeUtils.r con herramientas para la optimización.

Todo el código generado se encuentra adjunto con la entrega de esta memoria.

1.6 Breve descripción de los otros capítulos de la memoria

En el capítulo 2 se expone la estrategia general empleada a través de un benchmark como mecanismo de evaluación de los algoritmos CNV. El capítulo 3 explica la selección de los algoritmos CNV que han sido escogidos para realizar el benchmark. En el cuarto se detalla el dataset validado de 96 muestras sobre el cual se ejecutan los algoritmos mencionados. En el capítulo 5 se abarca el proceso de optimización que tiene como objetivo la búsqueda de los mejores valores de parámetros que maximicen la sensibilidad y especificidad (segundo criterio). En el capítulo sexto se exponen los resultados obtenidos tras la ejecución del benchmark y, finalmente, en el capítulo 7 se desarrollan las conclusiones del trabajo obtenido.

1.7 Agradecimientos

Este trabajo no hubiera sido posible sin los recursos de computación y almacenamiento facilitados por el Institut Germans Trias i Pujol (IGTP) y el Instituto de Investigación Biomédica del Hospital de Bellvitge (IDIBELL). Además, ha sido especialmente valiosa la colaboración proporcionada por Eduard Serra y Bernat Gel para encauzar este trabajo, siempre buscando utilidad desde el punto de vista clínico y traslacional. Cabe destacar que esta colaboración se ha producido gracias al Programa Conjunto ICO-PMPPC en Cáncer Hereditario.

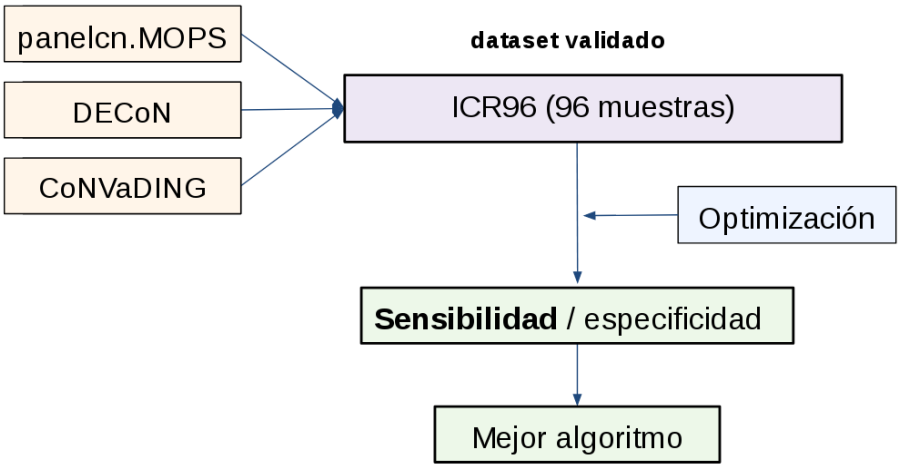
Así mismo, agradecer a Nazneen Rahman y todos los responsables del dataset ICR96 [4], por permitir el acceso al mismo para la realización de la presente investigación, así como de los futuros progresos que se lleven a cabo.

2. Benchmark, la estrategia para determinar el mejor algoritmo

En el apartado 1.3 se ha justificado la estrategia a emplear en el presente trabajo: la realización de un benchmark de partir de los algoritmos CNV existentes sobre un dataset validado por una técnica fiable.

Un benchmark es una comparativa en la que se evalúa bajo el mismo criterio el rendimiento de diferentes sistemas que comparten una misma función. En nuestro caso, los elementos a estudiar son **tres algoritmos** que han mostrado en publicaciones un buen rendimiento para detectan CNVs a partir de datos NGS de panel y sobre células constitucionales: panelcn.MOPS, DECoN y CoNVaDING. En el apartado 3 se desarrolla cómo han sido seleccionados estos 3 algoritmos de los más de 40 existentes.

Los tres algoritmos se evalúan contra un **dataset validado** por una técnica fiable (MLPA): el dataset ICR96 formado por 96 muestras. En el apartado 4 se amplía información sobre este dataset.



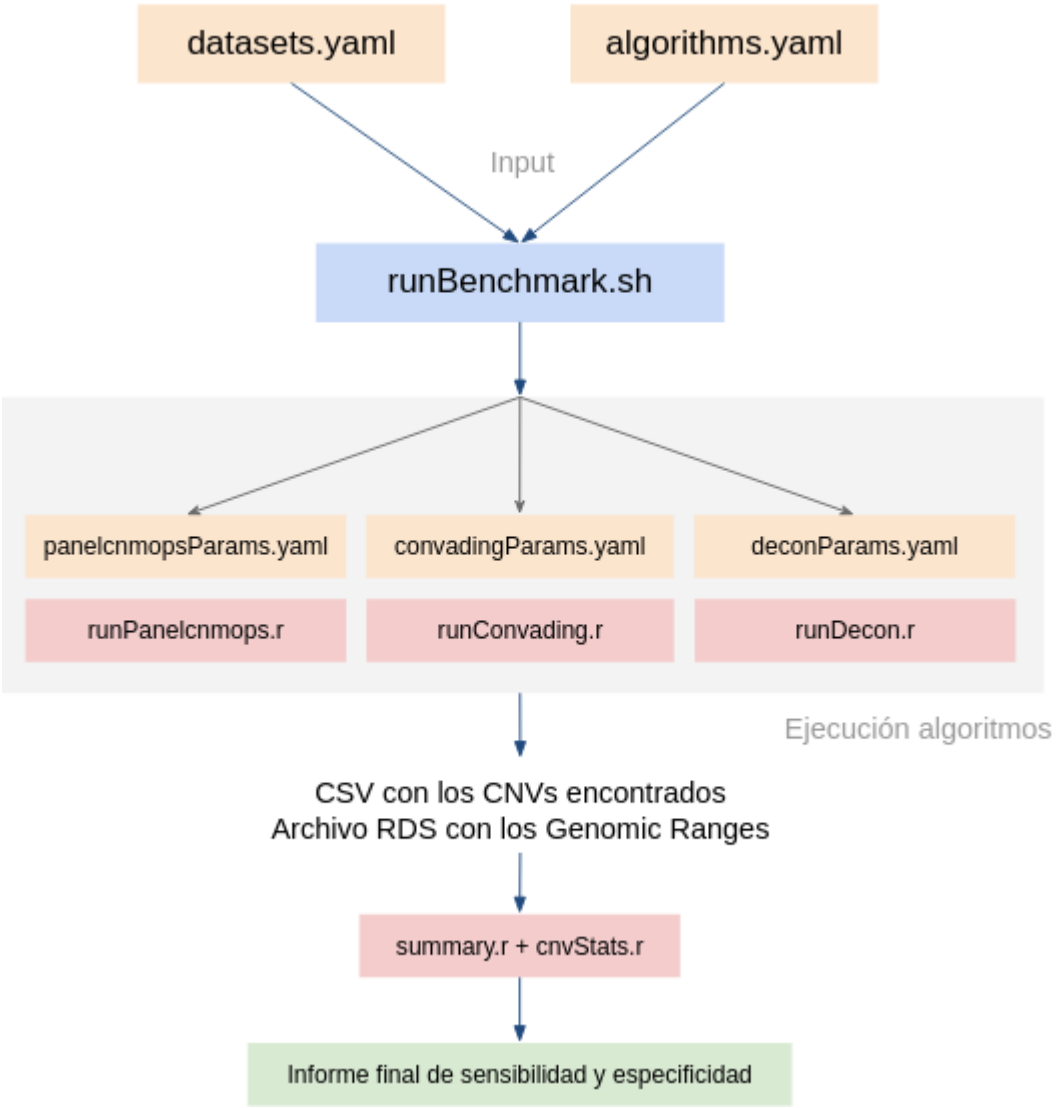
Esquema del benchmark realizado

Los algoritmos se ejecutan contra el dataset tanto con sus **valores por defecto** como con sus valores optimizados. El proceso de **optimización** consiste en la búsqueda de los mejores valores para los parámetros que maximizan la sensibilidad – primer criterio - y especificidad – segundo criterio -. En el apartado 5 se describe cómo se lleva a cabo la estrategia de optimización mediante un algoritmo Greedy y el empleo de un subconjunto del dataset.

Con lo descrito se obtienen resultados para los valores por defecto y para los valores optimizados. Estos resultados están descritos en el apartado 6.

2.1 Estructura del código

Se ha implementado un benchmark buscando un **diseño desacoplado, reutilizable y escalable** para permitir ampliar el benchmark a más datasets y algoritmos. El siguiente esquema muestra la estructura de ficheros que automatizan la ejecución del benchmark:



Se ha hecho uso de tres lenguajes: R, yaml y *shell command language* (sh):

- Se ha elegido R por ser un lenguaje de alto nivel, muy empleado en entornos bioinformáticos y para el cual existen una gran cantidad de utilidades estadísticas, genómicas, etc.
- El lenguaje yaml es un formato inspirado en XML y definido como un lenguaje de marcado ligero, muy útil para definir configuraciones.
- El lenguaje sh es el medio natural para realizar scripts en el entorno del sistema operativo Linux. Dado que el presente trabajo se ha realizado

con la distribución Debian 8, se ha podido aprovechar el potencial de este lenguaje de scripting.

2.1.1 Ficheros datasets.yaml y algorithms.yaml

Estos dos ficheros definen los algoritmos a ejecutar y sobre qué datasets lo harán. Están pensados para ser ampliados en el futuro. De hecho, en el fichero algorithms.yaml podemos ver más algoritmos de los evaluados en este trabajo:

```yaml
# contains params to define which algorithms and datasets will be run in the benchmark
algorithms:
  decon: false
  panelcn: false
  convading: true
  exomedepth: false
  viscap: false
```

Por su parte, el fichero datasets.yaml no sólo establece si un dataset ha de ser incluido en el benchmark (parámetro include), sino los parámetros que definen dicho dataset: directorio con los ficheros bams, fichero bed, fichero fasta, fichero de validación, read length y el formato del dataset (utilizado por otros ficheros del benchmark):

```yaml
icr96:
  include: false
  bams_dir: "/imppc/labs/eslab/jmmoreno/datasets/icr96/bams-bwa/"
  bed_file: "/imppc/labs/eslab/jmmoreno/datasets/icr96/bed4.bed"
  fasta_file: "/imppc/labs/eslab/jmmoreno/human_g1k/human_g1k_v37.fasta"
  validated_results_file: "/imppc/labs/eslab/jmmoreno/datasets/icr96/MLPAresults.txt"
  read_length: 101
  format: "icr"
```

2.1.2 Fichero runBenchmark.sh

El fichero runBenchmark es un script shell con dos partes diferenciadas. En la primera automatiza la ejecución de los algoritmos:

```shell
# Execute algorithms over selected datasets
if [ "$pars_algorithms_panelcn" == "true" ]; then
    echo "[$(date)] Executing panelcn.MOPS"
  Rscript-3.4-bioc-3.5 ./algorithms/panelcnmops/runPanelcnmops.r ./algorithms/panelcnmops/panelcnm
fi

if [ "$pars_algorithms_decon" == "true" ]; then
    echo "[$(date)] Executing DECoN"
  Rscript-bioc-3.6 ./algorithms/decon/runDecon.r ./algorithms/decon/deconParams.yaml datasets.yaml
fi

if [ "$pars_algorithms_convading" == "true" ]; then
    echo "[$(date)] Executing CoNVaDING"
  Rscript-bioc-3.6 ./algorithms/convading/runConvading.r ./algorithms/convading/convadingParams.ya
fi
```

En la segunda, se genera el informe llamando al fichero summary.r:

```
# Generate summary file
echo "[$(date)] Generating summary file"
Rscript-bioc-3.6 ./utils/summary.r algorithms.yaml datasets.yaml  > logs/summary.log 2>&1
```

2.1.3 Ficheros .yaml y .r de cada algoritmo

Para que cada algoritmo pueda ser ejecutado, se requieren dos ficheros:

- Un fichero yaml de configuración que define los valores de todos los parámetros que admite el algoritmo.
- Un fichero R cuyo seudocódigo a grandes rasgos sería el siguiente:

  Leer parámetros del algoritmo
  Leer parámetros del dataset
  Para cada dataset
    crear carpetas input/output
    ejecutar algoritmo
    transformar resultados a txt y GenomicRanges

2.1.4 Ficheros summary.r y cnvStats.r

El fichero summary.r genera un archivo txt con el informe de sensibilidad y especificidad para cada algoritmo y dataset ejecutado. Para procesar los resultados hace uso del fichero cnvStats.r que define varias clases que permiten calcular estadísticas:

- SampleAlgStats. Representa las estadísticas para una muestra concreta obtenida por un determinado algoritmo contra un dataset. Posee los campos TP, TN, FP y FN entre otros.
- SampleStats. Representa todas las estadísticas para una muestra concreta obtenida por uno o más algoritmos sobre un determinado dataset. Almacena entre otros los negativos y positivos validados para el dataset, así como una lista de SampleAlgStats (una para cada algoritmo).
- SummaryStats. Es la clase empleada directamente por el fichero summary.r y representa todas las estadísticas de los datasets y algoritmos ejecutados. Permite cargar resultados validados, generar informes o imprimir falsos negativos/positivos, entre otros.

Todas las anteriores clases están definidas bajo el sistema *Reference Classes* de R también conocido como R5 [13].

3. Algoritmos a comparar

En la literatura científica existe un gran número de algoritmos que detectan CNVs a partir de datos NGS con diferente alcance. La mayoría de ellos se comportan bien cuando trabajan con todo el genoma o el exoma, pero tienen problemas de rendimiento al trabajar con datos más esparcidos como son los **datos de panel** (habitualmente un subexoma conformado por un número de genes en torno a 100). Por otra parte, buena parte de los algoritmos CNVs son capaces de encontrar grandes CNVs (del orden de megabases) pero no lo son para pequeños CNVs que con frecuencia involucran uno o pocos exones, es decir, no llegan a **resolución de exón**. Por último, una parte de los algoritmos están enfocados a trabajar con células somáticas en lugar de **células constitucionales**, lo cual tampoco se ajusta a nuestro objetivo.

En consecuencia, se ha llevado a cabo un proceso de revisión del estado del arte referente a los algoritmos CNVs existentes. A falta de que la lista final pueda ser ampliada en futuras revisiones, hemos observado 43 algoritmos CNV que trabajan con datos NGS:

- 38 algoritmos poseen nombre propio: ExCopyDepth, CNVPanelizer, ControlFreeC, ExomeDepth, ExCopyDepth, CNV-RF, CLAMMS, FishingCNV, ExomeCopy, CODEX, EXCAVATOR, ADTEx, CONTRA, XHMM, CNVer, eXome, CANOES, Conifer, Capture-based high-coverage NGS, Invitae, Scalpel, ExomeCNV, panelcn.MOPS, cn.MOPS, SAAS-CNV, cnvOffSeq, ExoCNVTest, VisCap, CoNVaDING, VarScan2, cnvKit, CNVitae, SeqCNV, DECoN, PatternCNV, CONVector, SeqNext, NextGENe.
- 5 algoritmos aparecen en trabajos o artículos donde se propone una herramienta sin bautizarla con ningún nombre [8-12].

De toda la lista anterior, hemos observado hasta 6 algoritmos (descartando las licencias comerciales) que han mostrado en publicaciones tener un buen rendimiento en el citado contexto: DECoN, CoNVaDING, panelcn.MOPS, VisCap, ExomeDepth y cnvKit. Para acotar el ámbito abarcado por el presente trabajo, el benchmark a realizar se centrará en los 3 algoritmos que muestran resultados más prometedores (**DECoN, CoNVaDING y panelcn.MOPS**), sin perjuicio de que en el futuro esta lista pueda ser ampliada.

3.1 DECoN

DECoN (*Detection of Exon Copy number*) es un programa escrito en R que está optimizado para detectar CNVs a resolución de exón a partir de datos NGS de panel [5]. Ha sido desarrollado gracias a la colaboración del *Rahman Group* en el *Institute of Cancer Research* en Londres y el *Lunter Group* en el *Wellcome Trust Centre for Human Genetics* de Oxford. En el artículo donde se publica la herramienta muestran resultados con alta sensibilidad y especificidad para 2 datasets: un dataset validado de 96 muestras y un dataset simulado de 48 muestras con múltiples variaciones.

DECoN es una leve evolución de otro algoritmo CNV: ExomeDepth. Este algoritmo selecciona las muestras que correlacionan bien con la muestra evaluada y ajusta un modelo binomial para detectar posibles CNVs tanto por deleción como por inserción [14]. DECoN integra ExomeDepth (es un paquete de R), ajusta algunos valores de parámetros, y aporta una GUI interactiva para facilitar su uso.

Como inputs DECoN admite los siguientes elementos:

- BAM files (incluyendo .bai)
- Target regions (BED file)
- FASTA file con genoma de referencia
- 3 parámetros para configurar el algoritmo:
  - Minimum correlation threshold (mincorr)
  - Minimum coverage threshold (mincov)
  - Transition probability (transProb)

Como salida genera:

- Fichero tabulado con información sobre los CNVs encontrados: CNV.ID, Sample, Correlation, N. comp, Start.b, End.b, CNV.type, N.exons, Start, End, Chromosome, Genomic.ID, BF, Reads.expected, Reads.observed Reads.ratio, Gene.
- PDF con un informe
- Gráficos con los CNVs encontrados.

3.2 CoNVaDING

CoNVaDING (*Copy Number Variation Detection In Next-generation sequencing Gene panels*) es un programa escrito en Perl y diseñado para detectar pequeños CNVs a parir de datos NGS de panel con un alto coverage [15]. Ha sido desarrollado por el *University Medical Center Groningen*. En el artículo donde se publica CoNVaDING, mostraron alcanzar una sensibilidad del 100% sobre un dataset de 320 muestras (no incluyeron los positivos no encontrados para muestras etiquetadas como de baja calidad).

Este algoritmo hace uso de un subgrupo de muestras de control que presentan un patrón más similar [16]. Después, realiza una normalización del coverage y estima los CNVs existentes a partir de una combinación de ratio scores y z-scores de la muestra de interés comparada con las otras muestras.

Como inputs CoNVaDING admite los siguientes elementos:

- Directorio BAM files.
- Target regions BED file (targeted panel).
- 8 parámetros para configurar el algoritmo:
  - Número de muestras de control (controlSamples)
  - Porcentaje de todas las muestras que difieren más de 3 desviaciones estándar de la media de coverage para una región especificada (regionThreshold)
  - Umbral de correlación para determinar una deleción (ratioCutOffLow)

    - Umbral de correlación para determinar una duplicación (ratioCutOffHigh)
    - Umbral z-score para determinar una deleción (zScoreCutOffLow)
    - Umbral z-score para determinar una duplicación (zScoreCutOffHigh)
    - Umbral para excluir una muestra a partir de su z-score (sampleRatioScore)
    - Porcentaje de targets etiquetadas como *less reliable* (percentageLessReliableTargets).

Como salida, se generan tres niveles de archivos:

- Una longlist: contiene las CNVs detectados sin tener en cuenta la calidad de la muestra.
- Una shortlist: es un subconjunto de la longlist, filtrando por criterios de control de calidad obtenidos de la propia muestra.
- Una finallist: es un subconjunto de la shortlist, filtrando por criterios de control de calidad obtenidos de otras muestras.

Dado que el objetivo es obtener una buena sensiblidad, se ha optado por trabajar con la **lista longlist** que es la que a priori contará con mayor números de positivos, aunque el número de falsos positivos pueda ser mayor.

3.3 panelcn.MOPS

panelcn.MOPS es un paquete implementado en R que detecta CNVs a partir de datos NGS de panel. Se trata de una evolución del paquete de R cn.mops (C*opy Number estimation by a Mixture Of Poissons*) y que toma también parte del trabajo del paquete de R exomeCopy. En el paper donde se publica panelcn.MOPS afirman obtener una sensibilidad del 100% para un dataset validado de 170 muestras [2].

Este algoritmo implementa un pipeline compuesto por un control de calidad de la muestra, un proceso de selección de muestras de control más correlacionadas, una normalización por el tercer cuartil y un segundo control de calidad.

Como entrada panelcn.MOPS admite los siguientes parámetros:

- Directorio BAM files.
- Target regions BED file (targeted panel).
- Read length.
- 9 parámetros para configurar el algoritmo:
    - Tipo de normalización aplicada a los *read counts* de las muestras (normType).
    - Vector de 5 valores que determinar el *fold change* que define las CNVs (I).
    - Tipo de cálculo para los *size factors* de la normalización (sizeFactor).
    - Quantil del normType (qu).
    - Quantil del sizeFactor (quSizeFactor).
    - Estrategia general de normalización (norm).

- Umbral mínimo para las medias de de read counts de cada muestra (minMedianRC)
- Número máximo de muestras de control a usar (maxControls)

Como salida, genera una lista de objetos CNVDetectionResult, para los cuales podemos obtener la lista de GRanges o una tabla resumen. Además, también se pueden generar boxplots de los read counts normalizados para cada gen especificado por el usuario.

4. Datatet validado

Para evaluar el rendimiento de cada algoritmo es necesario disponer de un dataset validado mediante una técnica fiable. El dataset escogido para realizar el benchmark es el *ICR exon CNV validation series,* y se encuentra en el *European-Genome phemome Archive* (EGA) con el *accession number EGAS00001002428* [4]. Es necesario redactar un documento justificando la motivación de su uso para que los gestores del repositorio den finalmente acceso a los fastqs del dataset.

La siguiente tabla resume algunos datos descriptivos del dataset:

Nº muestras	96
Positivos	68 positivos: • 25 single-exon • 43 multi-exon o whole gene 2 muestras presentan dos positivos
Genes	15 BRCA1, 10 BRCA2, 1 NF1, 1 MLH1, 8 MSH2, 2 MSH6, 5 CHEK2, 1 EPCAM, 1 EZH2, 1 FH, 2 ATM, 6 NSD1, 1 PALB2, 5 PMS2, 1 PTEN, 1 RAD51C, 1 RB1, 2 SDHB, 3 TP53, 1 WT1
Secuenciación	TruSight Cancer Panel v2 (100 genes) en HiSeq 2500

Este dataset posee una gran representación de genes relacionados con el cáncer de mama (25 positivos para los genes BRCA1 y BRCA2). También incluye otros genes que se estudian habitualmente para realizar el diagnóstico de cáncer hereditario: MLH1, MSH2, MSH6, PMS2, EPCAM, PTEN y TP53 [4].

4.1 Descarga y puesta a punto del dataset

Una vez obtenido el permiso de acceso al dataset desde el repositorio EGA, la descarga y puesta a punto requiere de tres pasos para obtener los ficheros BAM que sirve de entrada para los algoritmos del benchmark.

4.1.1 Descarga del dataset

En el repositorio, el dataset se compone de 192 ficheros fastqs encriptados, dos para cada muestra. Para la descarga de todos los fastqs ha sido necesario descargar el cliente java ofrecido por EGA y seguir estos pasos:

- Lanzar el cliente con*: java -jar EgaDemoClient.jar*
- login nombreUsuario

- Determinar el path de descarga con: *path path-donde-quiero-descargar*
- Crear la solicitud de descarga: *request dataset EGAD00001003400 any request_EGAD00001003400*
- Iniciar la descarga*: download request_EGAD00001003400*
- Comprobar cuántas muestras se han bajado exitosamente del total: repetir los pasos anteriores hasta que indique que están todas descargadas. Nota: En las dos veces que he descargado el dataset han sido necesarias 3-4 órdenes de descarga del dataset.

Sin embargo, aquí no termina el proceso. Se ha de comprobar que el número de ficheros descargados en el directorio es el esperado. En las dos descargas del dataset, ocurrió que faltaba algún fichero. La solución de socorro empleada para este bug del cliente EGA ha sido volver a repetir todo el proceso de descarga y, una vez obtenido un nuevo directorio con los ficheros de las muestras, realizar la unión de ambos directorios. Así finalmente se ha obtenido el directorio completo con tofod los fastqs encriptados, dos para cada muestra.

4.1.2 Desencriptado de los fastqs

Los 192 fastqs descargados se encuentran encriptados con la clave indicada en el paso anterior. De nuevo es necesario emplear el cliente java EGA para proceder a su desencriptado. Para hacerlo de forma automática hemos construido un script shell que lanza el cliente para cada cip file del directorio:

```
for f in *.cip
do
        java -jar EgaDemoClient.jar -p user@mail.cat password -dc $f -dck abc
done
```

4.1.3 Alineamiento: de fastqs a ficheros BAM

Una vez desencriptados los ficheros fastqs podemos proceder al alineamiento de las secuencias contra el genoma de referencia para obtener los ficheros BAM (la versión binaria del formato SAM). El genoma necesario para realizar el alineamiento (el fichero human_g1k_v37.fasta y los ficheros .fai) ha sido obtenido del *1000 Genomes Proyect* [17].

El alineamiento (con el posterior ordenamiento, compresión a binario y generación de índices) se ha realizado mediante los programas *Burrows-Wheeler Aligner* y *SAMtools*. El proceso es muy exigente computacionalmente, por lo que para llevarlo a cabo se ha hecho uso de los recursos de computación del clúster disponible en el Institut Germans Trias I Pujol. Esto implica dividir el código en dos ficheros. El primero es un script shell (convertToBam-launcher.sh) que lanza contra clúster los jobs necesarios que ejecutarán el script convertToBam.sh:

```
7 for f in fastq/*.gz
8 do
9         if [[ "$f" == *R1.fastq.gz ]]
10        then
11                # obtain bam name
12                IFS='.|/' read -ra array_1 <<< "$f"
13
14                qsub -v sample=${array_1[1]::-2} -e output/${array_1[1]::-2}.log
15                         -o output/${array_1[1]::-2}.log convertToBam.sh
16        fi
17 done
```

El segundo es el propio script que realiza las operaciones de alineamiento, ordenamiento y generación de índices (convertToBam.sh). Obsérvese que como entrada recoge el nombre de la muestra:

```
34 samplefastq1=fastq/"$sample"R1.fastq.gz
35 samplefastq2=fastq/"$sample"R2.fastq.gz
36 bamname=bams-bwa/"$sample".bam
37
38 cd /icr96/
39
40 bwa mem -t 8 human_g1k_v37.fasta $samplefastq1 $samplefastq2
41         | samtools sort -@8 -O BAM -o $bamname
42 samtools index $bamname
```

4.1.4 BED file

Además de los ficheros BAM, los algoritmos a evaluar requieren un fichero BED que indique las ROI (*region of interest*) a evaluar. En concreto se necesita un fichero BED que cubra todas los exones de los genes del panel TruSight Cancer Panel v2 (100 genes). El fichero BED descargado ha sido el indicado en el propio paper donde se publica el dataset [4]. Una vez descargado, es necesario adaptar el formato para que sea compatible con todos los algoritmos:

- Eliminar el texto 'chr' de la columna del cromosoma.
- Ordenar por cromosoma y columna start.

5. Optimización

En la introducción del capítulo 1.1 se expuso que la motivación última en este trabajo es la de encontrar un algoritmo que permita realizar un screening previo a la validación por aCGH o MLPA. Es decir, el objetivo es determinar un algoritmo que sea capaz de encontrar todos los eventos CNV. Este algoritmo debería por tanto poseer una sensibilidad del 100% si bien la especificidad no es un requisito alto, siendo admisibles un cierto número de falsos positivos que serían descartados por la posterior validación.

Los algoritmos evaluados en el presente benchmark han sido diseñados con diferentes propósitos. En consecuencia, es necesario modificar los valores por defecto de los parámetros de un algoritmo dado para maximizar la sensibilidad que pueda ofrecer dicho algoritmo, pudiéndose maximizar además la especificidad una vez la sensibilidad no sea mejorable.

5.1 Estrategia

La búsqueda de los mejores valores de parámetros se realiza mediante un proceso de **optimización**, es decir, explorando el espacio de soluciones posible para maximizar la sensibilidad y especificidad (en segundo término). Este espacio de soluciones puede ser, sin embargo, de una dimensión inalcanzable computacionalmente si se pretende recorrer en su totalidad. Haciendo una estimación naif, si un algoritmo de los estudiados posee 8 parámetros (panelcn.MOPS posee 12) y evaluáramos todas las combinaciones de valores partiendo de que tenemos 10 valores posibles para cada parámetro, resultaríamos en 8^10 ejecuciones = $1,07 * 10^9$ ejecuciones. El tiempo de cada ejecución oscila entre los 20 min de DECoN y las 8 horas de CoNVaDING; en cualquier caso estamos ante un problema de optimización que requiere el empleo de **dos estrategias**: hacer uso de un **algoritmo de optimización** y realizarla sobre un **subconjunto** (*subset*) en lugar de sobre el total, además de hacer uso de un clúster que permita realizar muchas ejecuciones en paralelo.

Cabe remarcar que un rasgo necesario de la estrategia de optimización a realizar es que ha de seguir **un proceso sistemático donde se trate cada algoritmo y parámetro de igual manera**, sin ningún juicio precio. Así, se propone evaluar de la siguiente manera:

- Para cada **parámetro numérico**, evaluar 9 valores: el valor por defecto y 4 valores por encima y por debajo del valor por defecto. Estos valores se obtendrán haciendo variar el valor por defecto hasta un 30% sobre el valor por defecto.
- Para cada **parámetro categórico**, evaluar todos los valores o categorías que integren dicho parámetro.

5.2 Optimización sobre un subconjunto

En el apartado anterior se ha observado que el tiempo de ejecución de cada algoritmo unido al inmenso espacio de soluciones posibles hacen necesario el uso de estrategias para llevar a cabo el proceso de optimización en un tiempo razonable. Optimizar sobre un subconjunto en lugar de todo el dataset responde a dos motivaciones:

- Reducir el tiempo de ejecución del algoritmo dado que ha de procesar muchas menos muestras.
- Evitar problemas de *overfitting:* el subconjunto funciona como set de *trainning*, y se dispone del resto de muestras para validar de forma independiente los resultados obtenidos en dicho subset.

El subset que se ha definido para realizar el proceso de optimización del benchmark se compone de 30 muestras aleatorias (31% del total) que incluye entre ellas las muestras que poseen los positivos no encontrados por los algoritmos con los valores por defecto (4 muestras en total).

5.3 Algoritmo Greedy

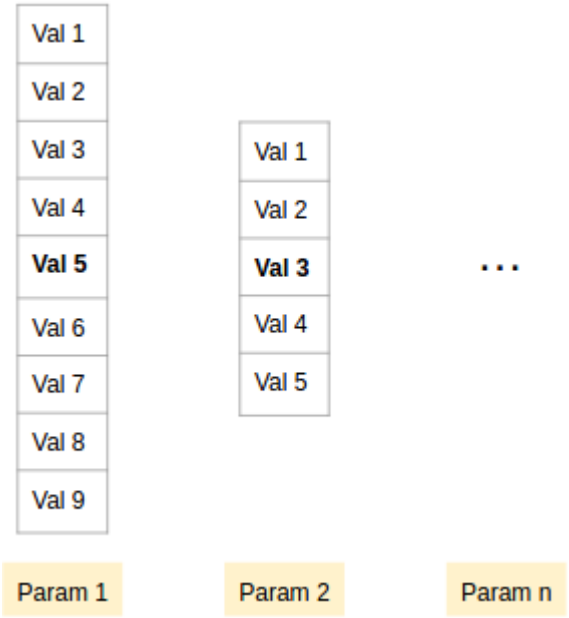
En informática, los problemas de optimización pueden ser resueltos por un vastísima variedad de algoritmos: desde un algoritmo de fuerza bruta hasta algoritmos genéticos, pasando por técnicas como backtracing o branch & bound. En este trabajo hemos optado por la elección de un algoritmo capaz de obtener resultados notables en problemas no muy complejos, a la vez que exige un tiempo de implementación moderado: el algoritmo Greedy.

Un algoritmo voraz o Greedy es aquel que lleva a cabo el proceso de optimización eligiendo la opción óptima en cada paso local. En algorítmica, existen ejemplos de algoritmos voraces muy estudiados, como es el caso del algoritmo de Kruskal o el algoritmo de Dijkstra.

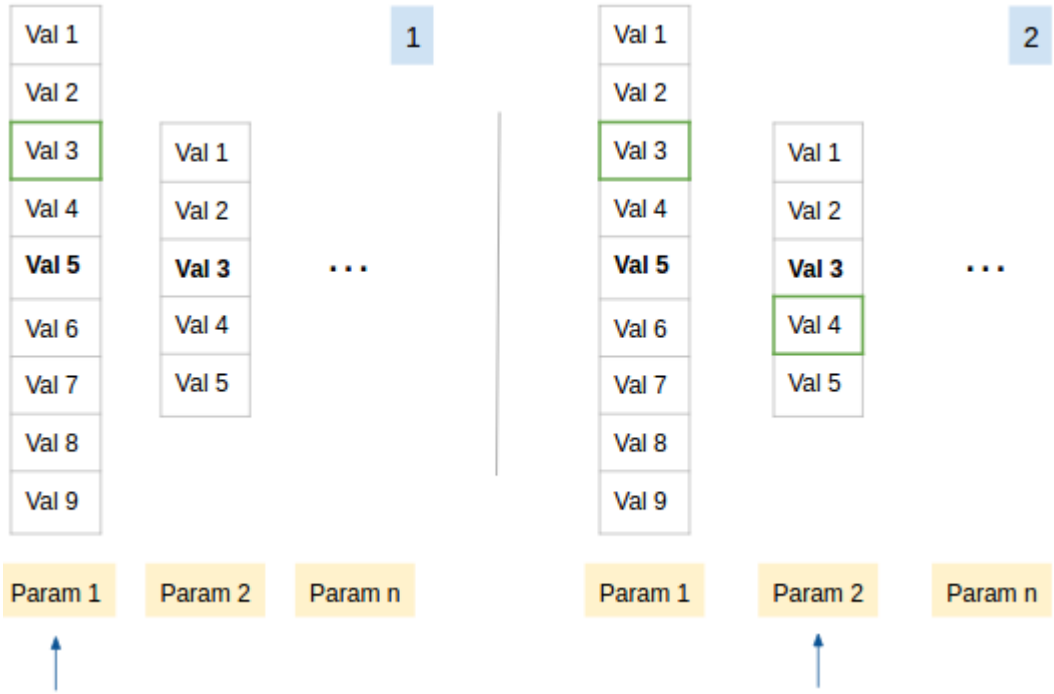
A continuación se exponen las tres iteraciones por las que ha pasado el diseño del algoritmo Greedy implementado.

5.3.1 Greedy básico

La primera aproximación del algoritmo Greedy es la más evidente. El siguiente esquema muestra el punto de partida, en el que tenemos n parámetros con una cierta cantidad de valores a evaluar para cada uno (ver apartado 5.1). En negrita se han resaltado los valores por defecto de cada parámetro:



En el primer paso del algoritmo se abarca el parámetro 1: se ejecuta tantas veces el algoritmo CNV como valores tome el parámetro (9 en el ejemplo), tomando el resto de parámetros los valores por defecto. Se elige como valor "ganador" aquel que obtuvo la mayor sensibilidad. En caso de empate, se elige la ejecución que obtuvo una mayor especificidad. Y si de nuevo ocurre un empate, se toma el valor por defecto bajo la premisa de no alterar si no es necesario los valores por defecto establecidos por los creadores del algoritmo CNV.



Esquema de funcionamiento del algoritmo Greedy básico, pasos 1 y 2

En un segundo paso, el algoritmo pasa a repetir el proceso con el parámetro 2. Se vuelven a hacer tantas ejecuciones como valores tenga el parámetro 2, pero esta vez tomando para el parámetro 1 el valor que resultara ganador en el anterior paso.

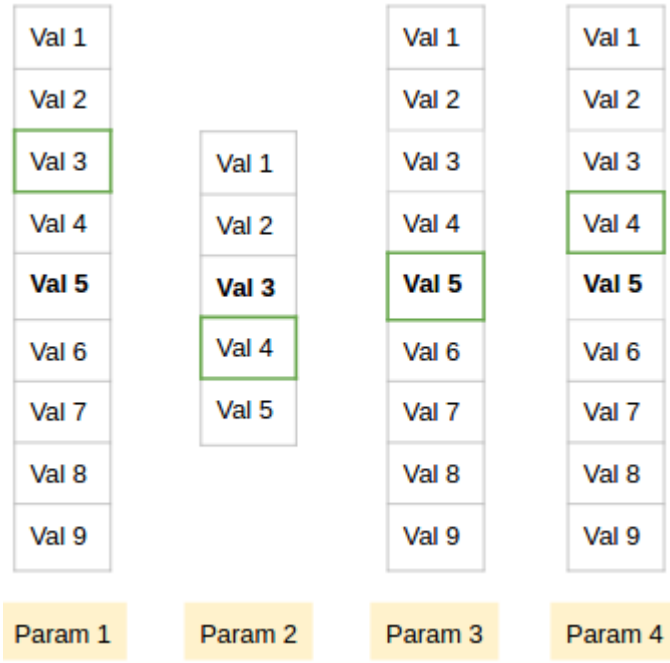
El esquema de funcionamiento se repita hasta completar todos los parámetros que posea el algoritmo CNV.

Esta primera aproximación del algoritmo Greedy posee una dificultad evidente para alcanzar el óptimo global: la elección de cuál es el primer parámetro condiciona de manera necesaria el resultado final. Dado que este algoritmo fija las decisiones tomadas haciendo imposible deshacerlas, sólo con la elección de cuál es el primer parámetro podemos estar condicionando la capacidad que tengan otros parámetros para influir en la decisión final. Teniendo en cuenta lo explicado, en el siguiente apartado se expone la segunda aproximación al algoritmo Greedy.

5.3.2 Greedy con diferentes puntos de inicio

En la primera aproximación hemos observado el sesgo que venía determinado por la elección de cuál era el primer parámetro a optimizar. Para evitar en buena medida este sesgo y explorar con mayor riqueza al espacio de soluciones, en la segunda aproximación se ha diseñado un algoritmo Greedy que realiza la aproximación básica para **tantos puntos de inicio como parámetros** tenga el algoritmo. Si tenemos 8 parámetros, lanzamos 8 veces el algoritmo Greedy. A modo de ejemplo: el algoritmo Greedy que tenga como punto de inicio el parámetro 2, comenzará por este parámetro y acabará en el parámetro 1.

Un riesgo al implementar esta estrategia es no hacer un control para ejecuciones idénticas que cada algoritmo con cada punto de inicio puede requerir. Para evitarlo, en el fichero optimizeUtils.r se ha codificado cada ejecución con el número de secuencia que la representa: cada dígito designa la posición que ocupa el valor en el espacio de valores del parámetro. Obsérvese la secuencia numérica en el ejemplo de un algoritmo que posea 4 parámetros:



Ejecución de un algoritmo de 4 parámetros, representada por la secuencia 3454

Esta codificación permite, entre otros propósitos, controlar que no se vuelva a realizar una ejecución que ha sido previamente realizada (o está pendiente de serlo) por otro algoritmo.

Aunque en los resultados iniciales el algoritmo Greedy con diferentes puntos de inicio estaba aportando más diversidad al espacio de soluciones y mejor rendimiento **sobre el subset de 30 muestras**, observamos que este buen rendimiento **no se extrapolaba al total del dataset**. Es decir, el rendimiento en sensibilidad para todo el dataset era peor incluso que el rendimiento ofrecido por la solución por defecto.

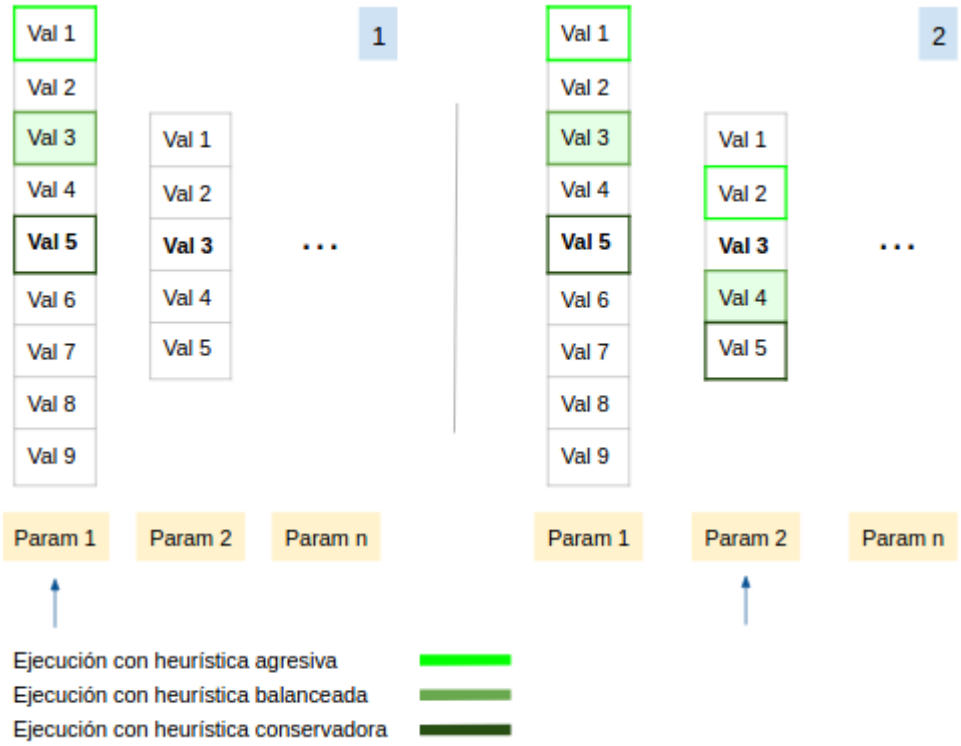
En los logs generados por la ejecución de los optimizadores se puede observar el comportamiento de los algoritmos: se alcanza la sensibilidad máxima que puede alcanzar el algoritmo rápidamente. En este punto, nos encontraríamos en una **zona "meseta"** sin margen de mejora para la sensibilidad, y sólo con margen para la mejora en especificidad. Este último criterio sí tiende a mejorar a cada paso del algoritmo: la especificidad queda optimizada. Sin embargo, acabamos obteniendo lo que podemos presumir como un "punto precipicio": un punto que aún mantiene sensibilidad máxima (está dentro de la meseta) pero al límite de caer en sensibilidad. Es decir, teniendo en cuenta el **trade-off existente entre sensibilidad y especificidad**, el algoritmo obtiene finalmente un set de parámetros que si bien mantiene el máximo de sensibilidad para el subset, queda significativamente volcado hacia criterios de especificidad, generando un resultado que al extrapolarlo al dataset completo pierde sensibilidad. En otras, palabras, **el algoritmo de optimización acaba siendo extremadamente agresivo al optimizar la especificidad en condiciones donde la sensibilidad no mejora.**

En el siguiente apartado se propone la tercera y definitiva versión del algoritmo Greedy para solucionar la problemática explicada.

5.3.3 Greedy con diferentes puntos de inicio y heurística modificada

La tercera versión del algoritmo Greedy incorpora una heurística que trata de encontrar soluciones menos volcadas hacia la especificidad. La idea clave de la heurística desarrollada es que para cada punto de inicio (cada parámetro) se tienen tres flujos de ejecución que se comportan de diferente manera cuando se tiene un conjunto de ejecuciones con la misma sensibilidad:

- En el primero se elegirá siempre la ejecución que presente la especificidad mejor, lo que denominamos una **heurística agresiva**.
- En el segundo se elegirá siempre la ejecución cuya especificidad sea próxima a la especificidad media de las ejecuciones con mejor y peor especificidad; la denominamos **heurística balanceada**.
- En el tercero se elegirá siempre la ejecución que presente la especificidad peor, lo que denominamos una **heurística conservadora**.



Los tres flujos de ejecución para cada punto de partida del algoritmo generarán como resultado un conjunto de soluciones con idéntica sensibilidad y diferente especificidad. Siguiendo el mismo criterio heurístico, el algoritmo ofrece finalmente **tres soluciones finales** de optimización:

- Una **solución con heurística agresiva**: la que presente mejor especificidad.

- Una **solución con heurística balanceada**: la que presente una especificidad cercana a la media de la mejor y peor de las soluciones encontradas.
- Una solución con **heurística conservadora**: la que presente peor especificidad.

6. Resultados

6.1 Resultados con valores por defecto

El siguiente gráfico muestra los resultados obtenidos al ejecutar los tres algoritmos con los parámetros por defecto sobre el dataset ICR96:

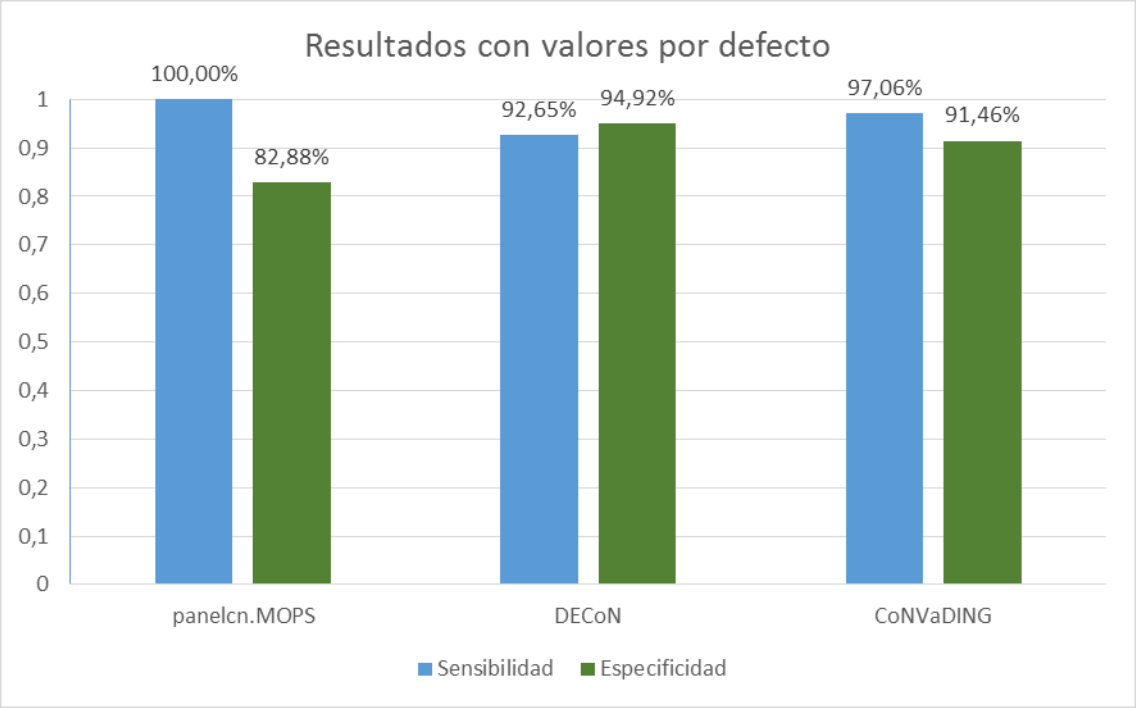


Figura de resultados 1

El algoritmo **panelcn.MOPS** es el único que no presenta ningún falso negativo: **alcanza una sensibilidad del 100%**. El resto de algoritmos presentan una sensibilidad alta sin obtener el mismo rendimiento: 92,65% para DECoN (4 falsos negativos) y 97,06% para CoNVaDING (2 falsos negativos).

En cuanto a la especificidad, DECoN (94,92%) y CoNVaDING (91,46%) obtienen los valores más altos, mientras que panelcn.MOPS consigue un valor sensiblemente menor (82,88%).

6.2 Resultados con valores optimizados

La optimización se ha realizado sobre un subconjunto de 30 muestras al que hemos denominado **subset30**. Los siguientes apartados muestran el rendimiento obtenido por el optimizador para dicho subset30, así como el rendimiento al evaluar los valores de parámetros obtenidos por el optimizador sobre todo el dataset ICR96.

6.2.1 Heurística agresiva

El output del optimizador recoge la mejor solución generada con la heurística agresiva, con los parámetros utilizados para cada algoritmo:

panelcn.MOPS
Best solution found: Sequence 555532455553 with 1 sensitivity and 0.785472972972973 specificity
Params: CN0=0.025 -v CN1=0.57 -v CN3=1.46 -v CN4=2 -v sizeFactor=quant -v norm=1 -v normType=poisson -v qu=0.25 -v quSizeFactor=0.75 -v priorImpact=1 -v minMedianRC=30 -v maxControls=25

DECoN
Best solution found: Sequence 551 with 0.884615384615385 sensitivity and 0.940038684719536 specificity
Params: -v mincorr=0.98 -v mincov=100 -v transProb=0.014 -v sequence=551

CoNVaDING
Best solution found: Sequence 25555155 with 0.961538461538462 sensitivity and 0.802218114602588 specificity
Params: qsub -v controlSamples=30 -v regionThreshold=20 -v ratioCutOffLow=0.65 -v ratioCutOffHigh=1.4 -v zScoreCutOffLow=-3 -v zScoreCutOffHigh=0 -v sampleRatioScore=0.09 -v percentageLessReliableTargets=20

Con estos resultados de sensibilidad y especificidad generamos el siguiente gráfico de resultados sobre el subset30:

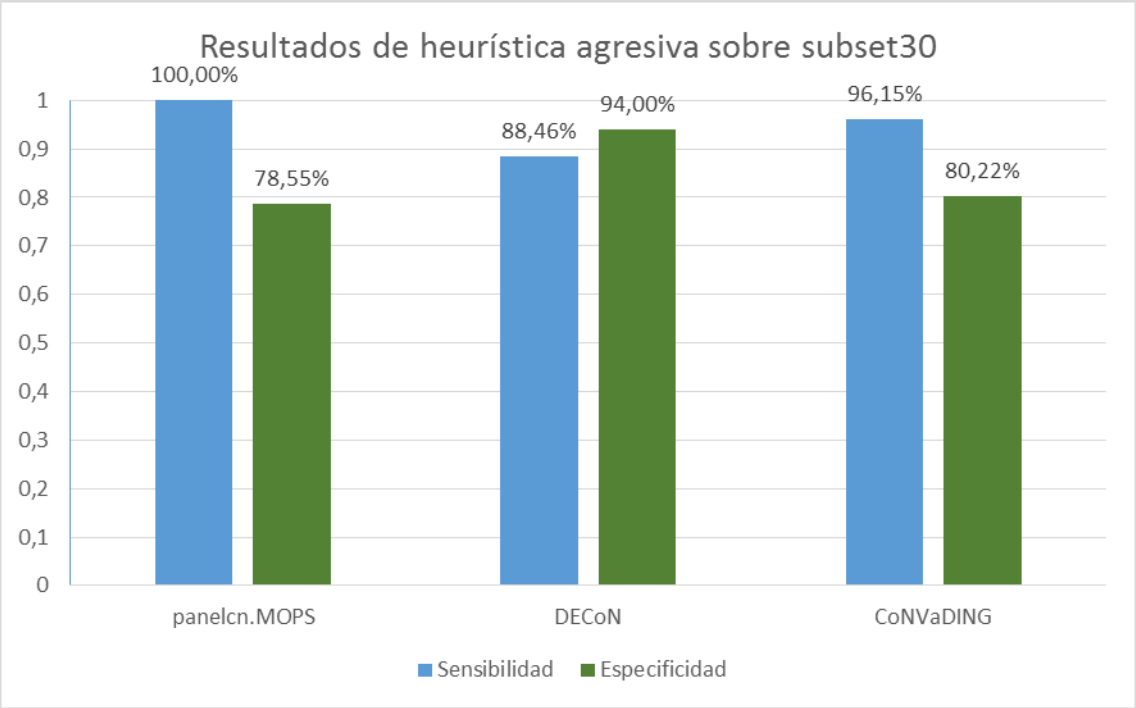


Figura de resultados 2

Y si extrapolamos los parámetros encontrados a todo el dataset ICR96:

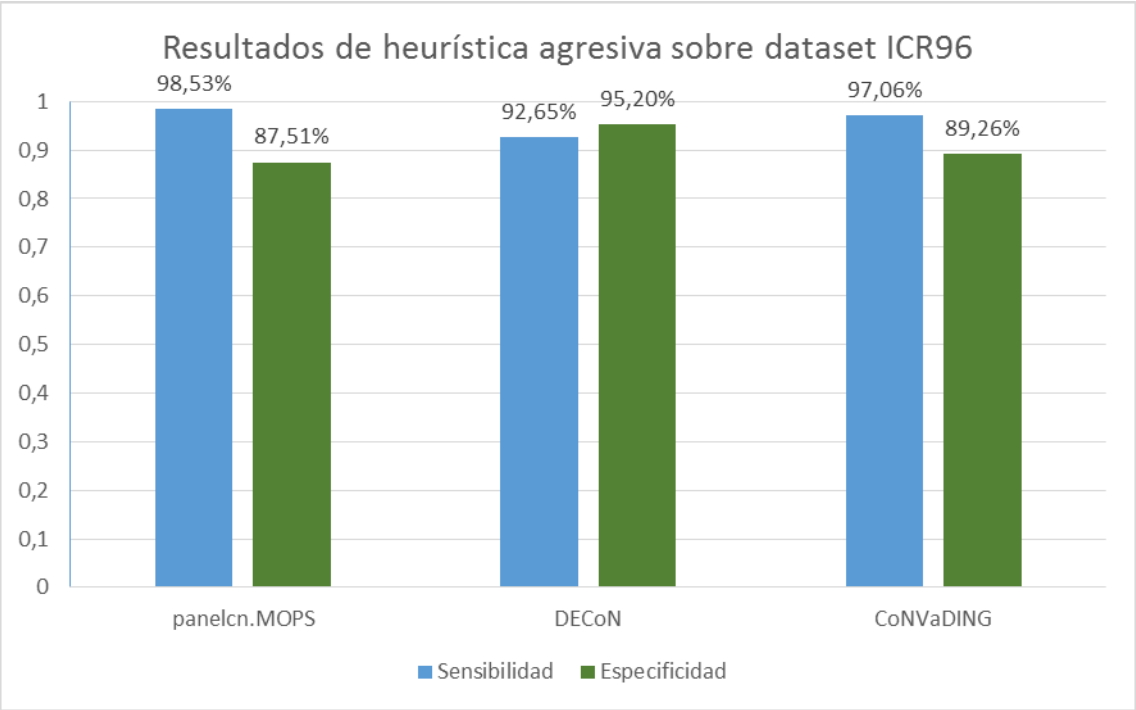


Figura de resultados 3

Para panelcn.MOPS observamos una sensibilidad peor que la obtenida por los valores por defecto (98,53% frente a 100%). La especificidad, sin embargo, si mejora con el proceso de optimización llegando a un 87,51% frente al 82,88% original. El resto de algoritmos mantienen la misma sensibilidad que en la ejecución con valores por defecto y mejoran levemente la especificidad.

6.2.2 Heurística balanceada

El output del optimizador recoge la mejor solución obtenida con la heurística balanceada, con los parámetros utilizados para cada algoritmo:

panelcn.MOPS
Best solution found (balanced): Sequence 555532555553 with 1 sensitivity and 0.752827140549273 specificity
Params: CN0=0.025 -v CN1=0.57 -v CN3=1.46 -v CN4=2 -v sizeFactor=quant -v norm=1 -v normType=mode -v qu=0.25 -v quSizeFactor=0.75 -v priorImpact=1 -v minMedianRC=30 -v maxControls=25

DECoN:
Best solution found (balanced): Sequence 554 with 0.884615384615385 sensitivity and 0.936293436293436 specificity
Params: mincorr=0.98 -v mincov=100 -v transProb=0.0185

CoNVaDING
Best solution found: Sequence 25555155 with 0.961538461538462 sensitivity and 0.802218114602588 specificity

Params: qsub -v controlSamples=30 -v regionThreshold=20 -v ratioCutOffLow=0.65 -v ratioCutOffHigh=1.4 -v zScoreCutOffLow=-3 -v zScoreCutOffHigh=0 -v sampleRatioScore=0.09 -v percentageLessReliableTargets=20

Con estos resultados de sensibilidad y especificidad generamos el siguiente gráfico de resultados sobre el subset30:

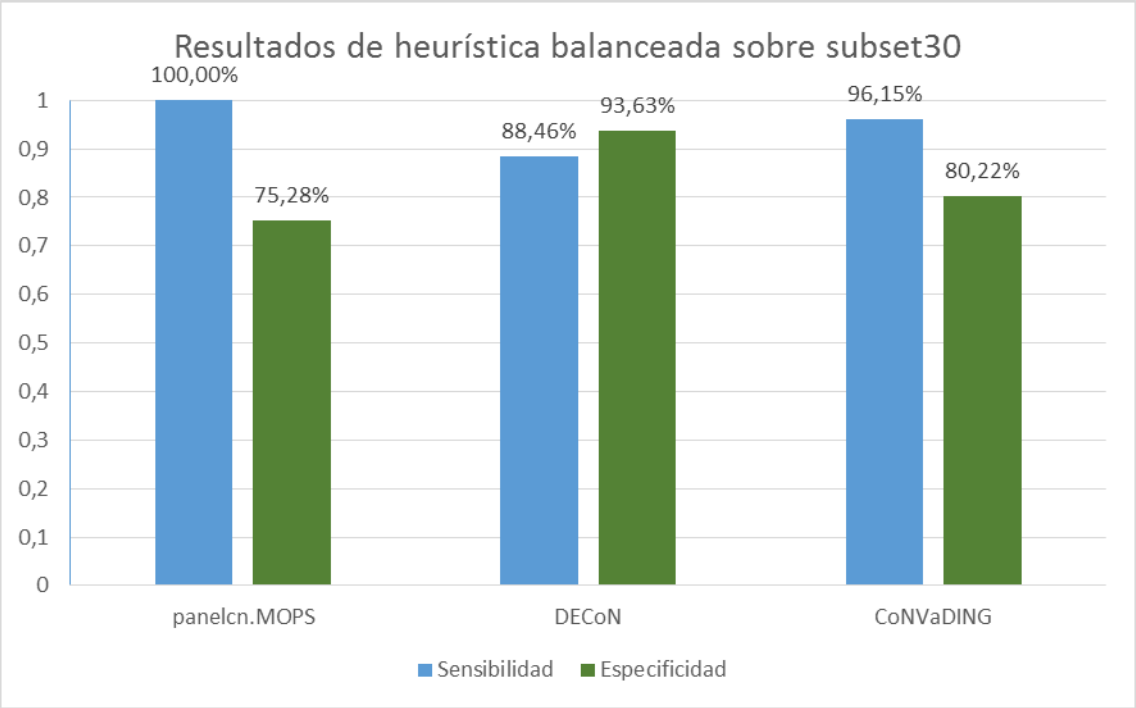


Figura de resultados 4

Y si extrapolamos los parámetros encontrados a todo el dataset ICR96:

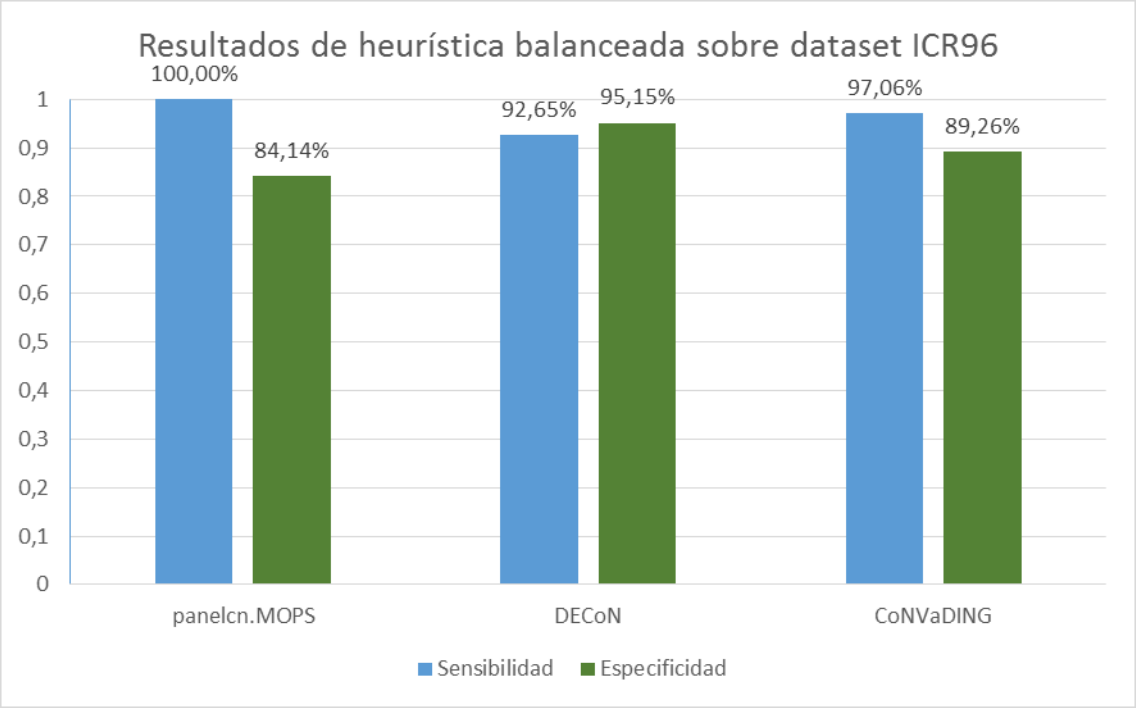


Figura de resultados 5

Observamos que los tres algoritmos mantienen la misma sensibilidad con respecto a la obtenida con el set de parámetros original. Sin embargo, dos de ellos mejoran la especificidad: panelcn.MOPS (84,14% frente a 82,88%) y DECoN (95,15% frente a 94,92). Por último, CoNVaDING pierde algo más de 2 puntos respecto al porcentaje original (89,26% frente a 91,46%).

6.2.3 Heurística conservadora

El output del optimizador recoge la mejor solución obtenida con la heurística conservadora generada, con los parámetros utilizados para cada algoritmo:

panelcn.MOPS:
Best solution found (conservative): Sequence 155232315112 with 1 sensitivity and 0.72827804107425 specificity
Params: CN0=0.025 -v CN1=0.57 -v CN3=1.46 -v CN4=1.55 -v sizeFactor=quant -v norm=1 -v normType=quant -v qu=0.25 -v quSizeFactor=0.75 -v priorImpact=1 -v minMedianRC=30 -v maxControls=25

DECoN
Best solution found (conservative): Sequence 115 with 0.884615384615385 sensitivity and 0.934362934362934 specificity
Params: mincorr=0.98 -v mincov=70 -v transProb=0.02

CoNVaDING
Best solution found: Sequence 25555155 with 0.961538461538462 sensitivity and 0.802218114602588 specificity
Params: qsub -v controlSamples=30 -v regionThreshold=20 -v ratioCutOffLow=0.65 -v ratioCutOffHigh=1.4 -v zScoreCutOffLow=-3 -v zScoreCutOffHigh=0 -v sampleRatioScore=0.09 -v percentageLessReliableTargets=20

Con estos resultados de sensibilidad y especificidad generamos el gráfico de la figura de resultados 7.

Extrapolando los parámetros encontrados a todo el dataset ICR96 generamos el gráfico de la figura de resultados 8. Observamos que los tres algoritmos vuelven a mantener la sensibilidad obtenida con los parámetros no optimizados. Respecto a la especificidad, panelcn.MOPS obtiene una leve mejoría (82,96% frente al 82,88%), DECoN la mantiene en el 94,92% y CoNVaDING vuelve a empeorarla (89,26% frente a 91,46%).

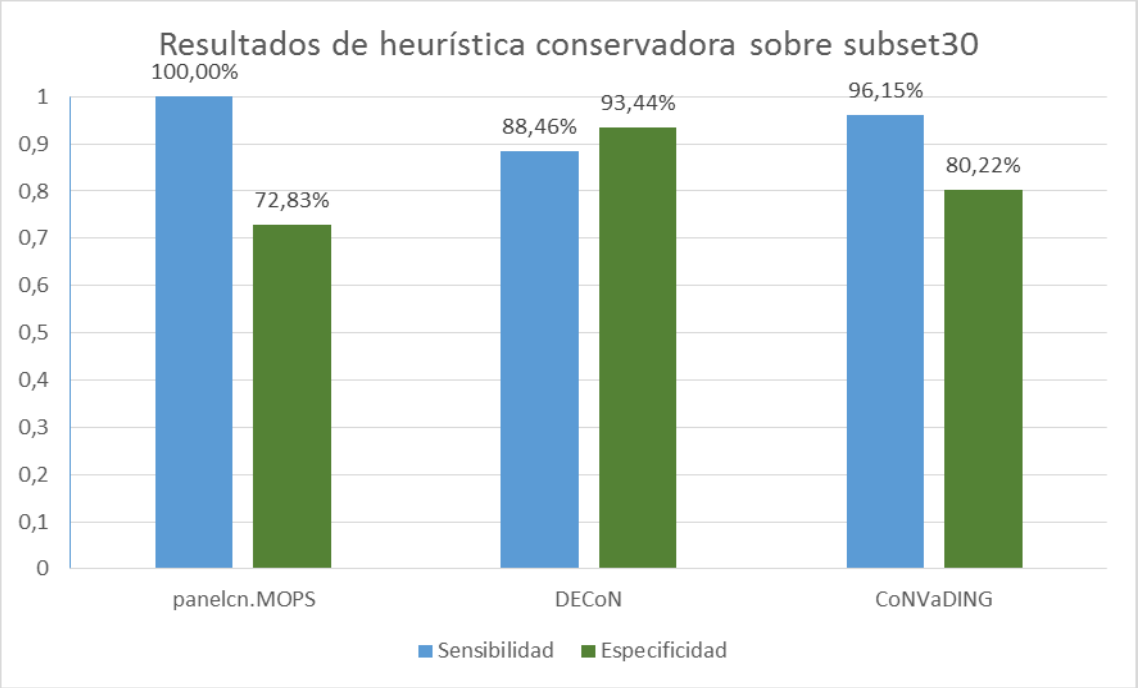


Figura de resultados 6

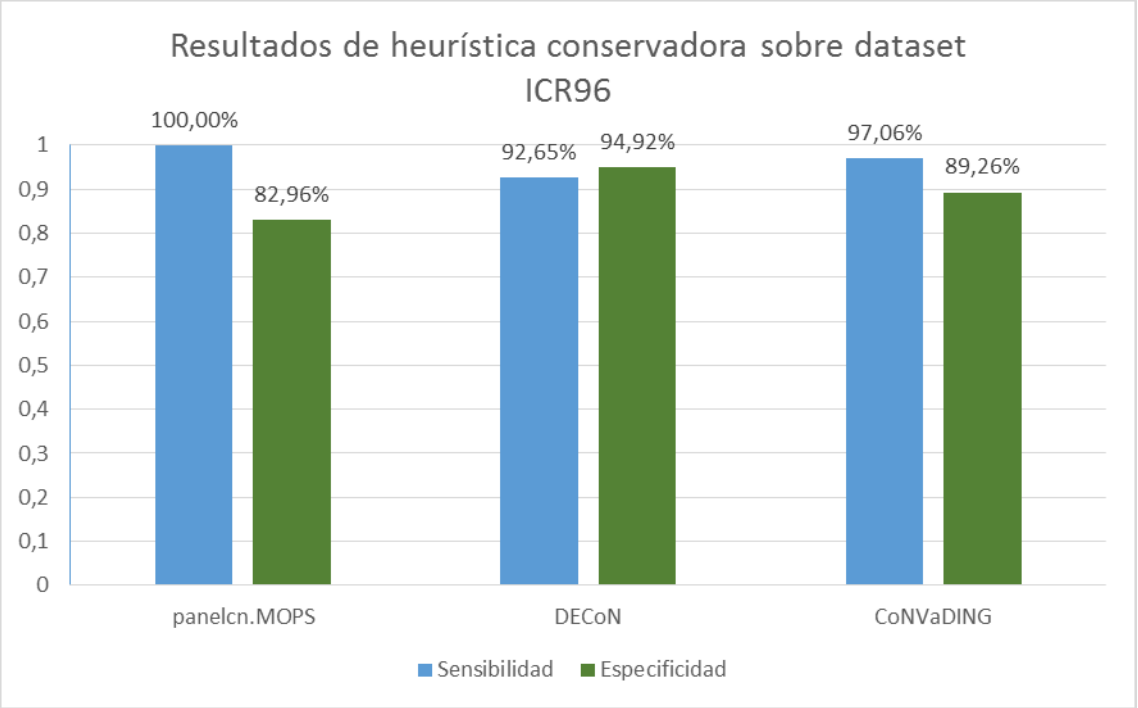


Figura de resultados 7

7. Conclusiones

7.1 Conclusiones generales

Observando los resultados de los valores por defecto (figura de resultados 1), podemos afirmar que el algoritmo panelcn.MOPS es el único que cumple el objetivo de sensibilidad 100%, es decir, detecta todos los positivos existentes en el dataset validado. CoNVaDING y DECoN muestran una sensibilidad alta pero insuficiente para los criterios de diagnóstico genético explicados en este trabajo.

El hecho de que panelcn.MOPS no tenga ningún falso negativo lo perfila como mejor candidato para su posible implementación en el laboratorio. Sin embargo, emplearlo sobre pacientes reales en la rutina del laboratorio exige confirmar que este mismo rendimiento se alcanza para otros datasets e idealmente para un dataset compuesto por muestras analizadas en el propio laboratorio donde se desea implementar el screening. En definitiva, panelcn.MOPS es un buen candidato pera será necesario más trabajo que confirme su rendimiento.

Respecto a la optimización con el algoritmo Greedy, podemos extraer conclusiones para cada una de las heurísticas:

- La heurística agresiva ha mostrado ser efectiva para mejorar significativamente la especificidad, pero a costa de obtener una peor sensibilidad cuando los parámetros optimizados se evalúan sobre todo el dataset (para el caso de panelcn.MOPS). Por tanto, no cumpliría con el objetivo planteado en este trabajo.
- La heurística balanceada ha obtenido unos resultados que mejoran la especificidad significativamente pero no mejoran la sensibilidad obtenida por los valores por defecto. Se comporta mejor que la heurística agresiva pero sin conseguir el objetivo de mejorar la sensibilidad.
- La heurística conservadora ha mostrado unos resultados similares a los de la heurística balanceada, pero con una mejora de la especificidad más pobre.

Cabe reseñar que si bien la sensibilidad obtenida al extrapolar los parámetros optimizados no mejora bajo ninguna de las heurísticas, la sensibilidad del algoritmo CoNVaDING sí mejora para el subset30. El optimizador es capaz de mejorar el algoritmo (bajo las 3 heurísticas) para detectar el positivo existente en la muestra 17301, pero al extrapolar al dataset con todas las muestras CoNVaDING no es capaz de detectar el positivo de dicha muestra. Esto sugiere la necesidad de estudiar por qué el algoritmo no es capaz de volver a encontrar un positivo en el dataset general, al mismo tiempo que parece indicar que el algoritmo podría estar optimizando en términos de sensibilidad:

  - En el caso de CoNVaDING, por alguna razón el algoritmo se comporta diferente para ambos datasets.
  - En el caso de panelcn.MOPS, no existe margen de mejora.

7.2 Objetivos planteados y planificación

Respecto a los **objetivos planteados** en la PEC 1 (ver apartado 1.2.2), podemos afirmar que en términos generales se han cumplido, salvo los resultados intermedios obtenidos para el tercer objetivo.

Respecto al primero, se ha seleccionado y obtenido acceso a un dataset con los requisitos planteados: el dataset ICR 96 explicado en el apartado 4. Respecto al segundo, también se han seleccionado tres algoritmos de toda la literatura científica, justificando su elección para cada uno de ellos. En cuanto al tercer objetivo, podemos afirmar que se ha hecho una búsqueda de los valores que maximizan la sensibilidad y especificidad para cada algoritmo, si bien conviene observar con visión crítica que el resultado de la optimización no ha mostrado resultados positivos en términos de sensibilidad, aunque sí lo haya hecho en términos de especificidad. Y respecto al cuarto objetivo y más importante, en párrafos anteriores hemos determinado panelcn.MOPS como el algoritmo que muestra mejor rendimiento al tiempo que se ha evaluado la conveniencia de aplicarlo en la rutina del laboratorio.

En cuanto a la **planificación**, durante las diversas etapas del presente TFM se han cumplido convenientemente los hitos, si bien la duración de las tareas ha sido en algunos casos levemente diferente (a veces por exceso y otras por defecto). Las tareas de optimización han resultado ser más complejas de lo inicialmente previsto, en parte debido a que exigimos sensibilidad 100% sobre todo el dataset pero entrenamos el algoritmo sobre un subset de 30 muestras. Esta dificultad fue solventada en parte añadiendo un tercer diseño del algoritmo de optimización que incorporaba diversas heurísticas.

Respecto a la **metodología empleada**, la decisión de realizar un benchmark para buscar una herramienta adecuada que sirva de screening previo a MLPA ha sido fructífera. Hemos podido evaluar los 3 algoritmos y observar el mejor rendimiento de panelcn.MOPS, no dejando ningún falso negativo en su proceso de detección de CNVs.

7.3 Líneas de trabajo futuro

Varios son los campos en los que este trabajo debe ser ampliado para terminar de confirmar al algoritmo panelcn.MOPS, u otros si los hubiera, como adecuado para ser empleado en la rutina del laboratorio.

Por una parte, es necesario extender el número de muestras sobre las que estamos evaluando los algoritmos. Sería conveniente, por tanto, encontrar más datasets accesibles públicamente que dispongan de muestras con CNVs que hayan sido previamente validadas por una técnica fiable. Sería de especial interés que uno de esos datasets esté compuesto por muestras secuenciadas en el propio laboratorio donde se pretende aplicar la técnica de cribado. Obtener sensibilidad 100% para también este último dataset nos permitiría determinar que un algoritmo puede ser empleado en la rutina del laboratorio

con un riesgo de no detectar positivos minimizado - con las consecuencias que ello tendría para el diagnóstico genético para los pacientes -.

Por otra parte, sería idóneo ampliar el benchmark a otros algoritmos que también mostraron un buen rendimiento pero que no han sido incluidos en este trabajo. Algunos de esos algoritmos ya fueron citados en el capítulo 3: VisCap, ExomeDepth y cnvKit.

Por último, el algoritmo de optimización implementado ha mostrado obtener una mejora en términos de especificidad pero no así en términos de sensibilidad. Si bien el algoritmo panelcn.MOPS no era mejorable en este sentido, el hecho de que CoNVaDING haya mejorado su sensibilidad notablemente sobre el subset30 pero no sobre todo el dataset, exige un mayor estudio para comprender mejor el problema y proponer cambios en el proceso de optimización que hagan más extrapolables los resultados obtenidos. Además, el algoritmo de optimización empleado es Greedy, pero existe una cantidad enorme de herramientas algorítmicas, susceptibles de poder obtener mejores resultados que la aproximación realizada en el presente trabajo.

8. Glosario

aCGH. Arrays de hibridación genómica comparativa (a*rray comparative genomic hybridization*), técnica de detección de alteraciones en el número de copias.

Benchmark. Comparativa en la que se evalúa bajo el mismo criterio el rendimiento de diferentes sistemas que comparten una misma función.

BAM. Versión binaria de un fichero SAM.

CNV. Variación en el número de copias (c*opy number variation)* consistente en la deleción o inserción de una o varias regiones de al menos 50 pares de bases.

Coverage. Número de reads existentes para una nucleótido en la secuencia obtenida tras el alineamiento.

Fastqs. Formato de texto que almacena secuencias de nucleótidos con su calidad de lectura.

Fold change. Medida que expresa cuánto cambia un valor inicial para convertirse en un cierto valor final, y se calcula como la ratio de la diferencia entre el valor final e inicial.

Heurística. Criterio que modifica la lógica de un algoritmo para dotarlo de mayor velocidad o para alcanzar una solución no óptima.

MLPA*. Multiplex ligation-dependent probe amplification* es una técnica basada en la reacción en cadena de la polimerasa (RCP), de cuantificación relativa del número de copias normales y anormales de ADN de hasta 40 secuencias genómicas diferentes [7].

NGS. Tecnología de secuenciación en paralelo que ha cambiado el paradigma de la secuenciación genómica, también conocida como *high-throughput sequencing*, *massively parallel* o *deep sequencing*.

Overfitting. Es el sobreajuste que se produce cuando un algoritmo o modelo se comporta mejor para el subconjunto de entrenamiento que para el subconjunto de validación.

SAM. Formato de texto tabulado que almacena secuencias de bases alineadas respecto al genoma.

9. Bibliografía

[1] Wong, L. C. (2017). *Next Generation Sequencing Based Clinical Molecular Diagnosis of Human Genetic Disorders*.

[2] Povysil, G., Tzika, A., Vogt, J., Haunschmid, V., Messiaen, L., Zschocke, J., … Wimmer, K. (2017). panelcn.MOPS: Copy number detection in targeted NGS panel data for clinical diagnostics. *Human Mutation*, (April). https://doi.org/10.1002/humu.23237

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

[4] Mahamdallie, S., Ruark, E., Yost, S., Ramsay, E., Uddin, I., Wylie, H., … Rahman, N. (2017). The ICR96 exon CNV validation series: a resource for orthogonal assessment of exon CNV calling in NGS data. Wellcome Open Research, 2, 35. https://doi.org/10.12688/wellcomeopenres.11689.1

[5] Fowler, A., Mahamdallie, S., Ruark, E., Seal, S., Ramsay, E., Clarke, M., … Rahman, N. (2016). Accurate clinical detection of exon copy number variants in a targeted NGS panel using DECoN. Wellcome Open Research, 1, 20. https://doi.org/10.12688/wellcomeopenres.10069.1

[6] "What is a copy number variant, and why are they important risk factors for ASD?", Autism Reading Room. http://readingroom.mindspec.org/?page_id=8221

[7] Estrada-Juárez, Higinio & Fernández-Hernández, Liliana & Rivera-Pedroza, Carlos & Grether-González, Patricia. (2012). MLPA (multiplex ligation-dependent probe amplification) in perinatal rapid diagnosis of major aneuploidies. Perinatología y reproducción humana / INPer. 26. 172-179.

[8] Wei, X., Dai, Y., Yu, P., Qu, N., Lan, Z., Hong, X., … Yi, X. (2014). Targeted next-generation sequencing as a comprehensive test for patients with and female carriers of DMD/BMD: a multi-population diagnostic study. European Journal of Human Genetics, 22(1), 110–118. https://doi.org/10.1038/ejhg.2013.82

[9] Ji, H., Lu, J., Wang, J., Li, H., & Lin, X. (2014). Combined examination of sequence and copy number variations in human deafness genes improves diagnosis for cases of genetic deafness. BMC Ear, Nose and Throat Disorders, 14(1), 9. https://doi.org/10.1186/1472-6815-14-9

[10] Moens, L. N., Falk-Sörqvist, E., Asplund, A. C., Bernatowska, E., Smith, C. I. E., & Nilsson, M. (2014). Diagnostics of Primary Immunodeficiency Diseases: A Sequencing Capture Approach. PLoS ONE, 9(12), e114901. https://doi.org/10.1371/journal.pone.0114901

[11] Wong, L. C. (2017). Next Generation Sequencing Based Clinical Molecular Diagnosis of Human Genetic Disorders.

[12] Shen, W., Szankasi, P., Sederberg, M., Schumacher, J., Frizzell, K. A., Gee, E. P., … Kelley, T. W. (2016). Concurrent detection of targeted copy number variants and mutations using a myeloid malignancy next generation sequencing panel allows comprehensive genetic analysis using a single testing strategy. British Journal of Haematology, 173(1), 49–58. https://doi.org/10.1111/bjh.13921

[13] “ReferenceClasses”, R Documentation, v3.4.3.
https://www.rdocumentation.org/packages/methods/versions/3.4.3/topics/ReferenceClasses

[14] Plagnol, V., Curtis, J., Epstein, M., Mok, K. Y., Stebbings, E., Grigoriadou, S., … Nejentsev, S. (2012). A robust model for read count data in exome sequencing experiments and implications for copy number variant calling. Bioinformatics (Oxford, England), 28(21), 2747–54. https://doi.org/10.1093/bioinformatics/bts526

[15] Johansson, L. F., van Dijk, F., de Boer, E. N., van Dijk-Bos, K. K., Jongbloed, J. D. H., van der Hout, A. H., … Sikkema-Raddatz, B. (2016). CoNVaDING: Single Exon Variation Detection in Targeted NGS Data. Human Mutation, 37(5), 457–464. https://doi.org/10.1002/humu.22969

[16] “CoNVaDING User Guide”, CoNVaDING, v1.2.1.
https://molgenis.gitbooks.io/convading/

[17] FTP of 1000 Genomes Project
http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/

[18] PubMed
https://www.ncbi.nlm.nih.gov/pubmed/

[19] Google Scholar
https://scholar.google.es/

[20] Biostars
https://www.biostars.org/