Desenvolupament d'eines de recerca genòmica i la seva integració a l'entorn Galaxy

Títol: **Desenvolupament d'eines de recerca genòmica i la seva integració a l'entorn Galaxy**

Autor: Ferran Lloret Yusta

Data: Febrer - Juny 2013

Director: Xavier Messeguer Peypoch

Departament del director: Llenguatges i Sistemes Informàtics (LSI)

Co-director: Eduardo Eyras Jiménez

Departament del co-director: Grup de recerca en genòmica computacional (UPF)

Titulació: Enginyeria Informàtica (Pla 2003)

Centre: Facultat d'Informàtica de Barcelona (FIB)

Universitat: Universitat Politècnica de Catalunya (UPC). BarcelonaTech

Índex

Índex de figures

1. INTRODUCCIÓ

Gràcies a la informàtica, durant les darreres dècades la recerca científica ha pogut gaudir de grans millores en mètodes de recerca. A més, han aparegut nous camps d'investigació, un dels quals és la bioinformàtica. Aquest projecte s'ha realitzat en el marc de la recerca genòmica i les tècniques *ChIP-sequencing*.

ChIP-sequencing, també anomenat *ChIP-seq,* és un mètode per analitzar les interaccions entre ADN i proteïnes. S'utilitza per detectar factors de transcripció, que regulen els processos de transcripció genètica. D'aquesta manera, es pot conèixer millor el funcionament de diversos processos biològics, com l'aparició de malalties.

> L'ADN (àcid desoxiribonucleic) és una cadena de molècules que codifica el funcionament de tot organisme viu. Dins l'ADN hi trobem els *gens*, les unitats bàsiques en què s'emmagatzema aquesta informació. El procés d'interpretació de l'ADN s'anomena *transcripció genètica*, i es basa en la còpia de diversos segments dels gens (anomenats *exons*) en una altra cadena de molècules: l'ARN missatger.
>
> Els factors de transcripció són proteïnes que s'uneixen a l'ADN i regulen l'activitat dels gens. D'aquesta manera es controla el procés de transcripció genètica i la funció que desenvolupen les cèl·lules.

El procés amb què es detecten els factors de transcripció s'anomena *peak-calling*. Aquest es basa en trobar regions genòmiques amb un nombre elevat de lectures (blocs d'informació genòmica procedents de la seqüenciació de l'ADN).

Amb aquest projecte es pretén desenvolupar una sèrie de funcionalitats que ajudin a l'anàlisi de dades d'experiments *ChIP-seq*, millorant així les eines de recerca existents.

Es pot trobar més informació sobre ChIP-seq a l'article *Computation for ChIP-seq and RNA-seq studies*, de Shirley Pepke, Barbara Wold i Ali Mortazavi.

1.1. Objectius

Els dos objectius principals del projecte són l'ampliació de l'eina *Pyicos* i l'adaptació d'aquesta a l'entorn *Galaxy*. Com a objectiu secundari, aquesta memòria pretén ser una

eina de consulta per part dels membres del grup de recerca.

L'ampliació de Pyicos es basa en el disseny i implementació de noves funcionalitats per donar suport a la recerca genòmica. Algunes de les funcionalitats es van proposar a l'inici del projecte, tot i que algunes altres s'han anat proposant al llarg del seu desenvolupament.

Amb el segon objectiu esmentat es pretén integrar el conjunt d'eines de Pyicos (les implementades i les ja existents) a la plataforma *Galaxy*, que en simplifica l'ús mitjançant una interfície web.

1.2. Tecnologies utilitzades

Python: Aquest projecte s'ha implementat amb Python, un llenguatge interpretat i d'alt nivell dissenyat per a un desenvolupament ràpid i eficient d'aplicacions. A causa dels freqüents canvis en metodologies d'investigació genòmica i l'aparició de nous mètodes d'anàlisi, les tecnologies que s'utilitzen han de permetre realitzar modificacions al codi de forma ràpida i fàcil. És per això que es prefereixen llenguatges de programació *Scripting* com Perl o Python, en comptes de llenguatges compilats com C o Java.

Git: Per la gestió del codi s'ha utilitzat l'eina de control de versions *git*, amb un repositori privat a *BitBucket* (http://bitbucket.org). Això ha facilitat l'edició del codi per part d'altres contribuents i la immediata disponibilitat de les funcionalitats noves per part del grup de recerca.

Sphinx: Per generar el manual d'ús i documentació del programa s'ha utilitzat l'eina lliure *Sphinx*. Aquesta ha permès la generació de documents HTML a partir de fitxers en format reStructuredText.

XML: Per integrar Pyicoteo amb Galaxy s'ha utilitzat XML, segons la sintaxi descrita a http://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax

1.3. Organització de la memòria

La memòria s'ha estructurat en els següents apartats:

1. Introducció. Es fa una breu introducció al tema del projecte i se n'exposen els objectius.
2. Planificació. Es presenta la forma en què s'ha organitzat la feina i s'estima el cost econòmic del projecte.
3. Disseny i implementació. S'explica com s'ha desenvolupat el projecte: quines decisions s'han pres i a quines solucions s'ha arribat.
4. Conclusions. Reflexions finals sobre el projecte.
5. Bibliografia. Conté les fonts d'informació que s'han utilitzat durant el desenvolupament del projecte.
6. Annexos. Material complementari.

2. PLANIFICACIÓ

Per organitzar la feina, els objectius del projecte es van dividir en tasques i se'n va estimar el temps de desenvolupament.

- Procés d'aprenentatge (1 mes)
- Ampliació de Pyicos (2 mesos)
  - GFFReader
  - Cerca d'exons, introns, regions inter-/intragèniques i TSS
  - Marcatge de regions a pyicoenrich
  - CustomReader
  - UCSCUpload
  - Reestructuració de Pyicos a Pyicoteo
  - Documentació del programa
- Integració de Pyicos a Galaxy (1 mes)
- Elaboració de la memòria i presentació (1 mes)

Cost econòmic

El desenvolupament del projecte va durar un total de 82 dies. El nombre d'hores dedicades és de 82*8 = 656 h. Considerant un sou de becari 6 € l'hora, el cost total del desenvolupament és de 656*6 = 3936 €.

No s'han produït costos externs, gràcies què totes les tecnologies utilitzades són obertes i de lliure distribució.

Recompte d'esdeveniments

Per desenvolupar el projecte em vaig incorporar al grup de recerca en genòmica computacional dirigit per l'Eduardo Eyras. al Parc de Recerca Biomèdica de Barcelona (PRBB).

Vaig començar el projecte el 4 de febrer de 2013. Durant les primeres quatre setmanes vaig estar aprenent el funcionament del software i l'entorn de treball. Vaig instal·lar l'última versió de Pyicos i vaig fer execucions amb dades de mostra, per a observar-ne el funcionament. També vaig fer diverses modificacions al manual d'ús del programa i vaig llegir-me articles relacionats amb la feina d'investigació que fa el grup de recerca.

A la quarta setmana, mentre encara estava en procés d'aprenentatge, se'm va proposar que implementés un conjunt de classes i funcions per a llegir i interpretar fitxers de regions GFF.

Amb aquestes classes implementades, se'm va proposar dissenyar i implementar diversos algorismes per a la cerca de regions. També se'm va demanar trobar alguna forma de mostrar les dades des de Pyicos al UCSC Browser.

A principis d'abril vaig implementar el marcatge de regions seleccionades com a “interessants” a les gràfiques generades per Pyicoenrich. Durant aquestes dates també vaig contribuir a la reestructuració del codi de Pyicos a Pyicoteo.

El dia 15 d'abril se'm va donar la oportunitat de presentar la feina feta fins llavors en la reunió setmanal del grup de recerca. Per preparar la presentació, durant la setmana anterior vaig haver de deixar de banda el desenvolupament.

Durant la reunió se'm va proposar la implementació d'un altre algorisme per a la cerca de regions: els TSS, el qual vaig implementar durant l'última setmana d'abril. Abans, però, vaig dissenyar i implementar el lector de fitxers propis CustomReader.

La major part del mes de maig la vaig dedicar a integrar les eines de Pyicoteo a l'entorn Galaxy i provar-ne el funcionament.

Per acabar, les últimes setmanes de desenvolupament del projecte les vaig dedicar a investigar quines altres eines hi ha amb funcionalitat semblant a Pyicoteo.

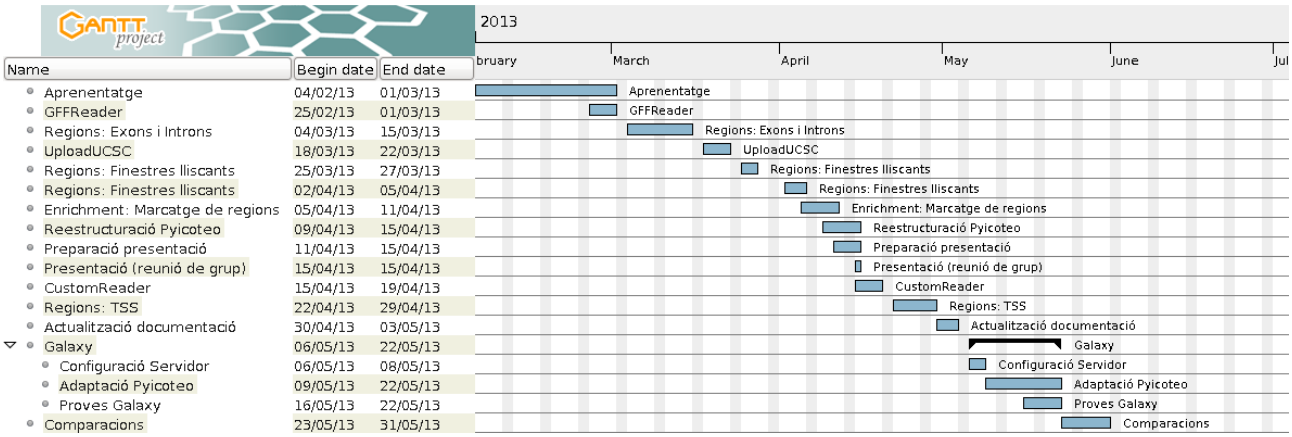

Figura 1: Diagrama de Gantt de la feina realitzada

3. DISSENY I IMPLEMENTACIÓ

3.1. Pyicos

Pyicos (anomenat Pyicoteo a partir de la versió 2.0) és un conjunt d'eines per a l'anàlisi de dades genòmiques. Proporciona comandes executables i una biblioteca per a qui vulgui desenvolupar noves eines.

Les seves funcionalitats són diverses, entre elles trobem:

- Manipulació de dades: Permeten fer modificacions bàsiques als fitxers de dades, com *extend* (incrementa la mida de les lectures), *push* (canvia la posició de les lectures), *subtract* (elimina les lectures que coincideixen amb les d'un fitxer proporcionat) i *convert* (converteix entre diferents formats de fitxer).
- Enriquiment: Donats els resultats de dos experiments, realitza un anàlisi estadístic per a trobar similaritats i diferències entre ells.
- *Peak-caller*: Realitza anàlisis estadístics per trobar regions significants en dades de ChIP-seq.

Un dels objectius principals de Pyicos és que sigui fàcil d'executar amb qualsevol ordinador personal modern. Això s'aconsegueix amb l'optimització dels algorismes per a què utilitzin poca memòria, encara que s'afecti negativament el temps d'execució. També s'intenta reduir al mínim el nombre de dependències amb biblioteques externes, per a què l'únic requeriment de software sigui un intèrpret de Python. Actualment les úniques biblioteques externes de les que depèn són *Matplotlib*, per generar gràfiques, i *Jinja2*, per mostrar les gràfiques generades en un document HTML. Les dues són opcionals i només s'utilitzen en cas que estiguin instal·lades. També pot fer ús de *samtools* per llegir fitxers de format BAM, tot i que Pyicos proporciona la seva pròpia implementació.

La seva pàgina web es troba a http://regulatorygenomics.upf.edu/pyicos

Per a més informació, es pot consultar l'article *Pyicos: A versatile toolkit for the analysis of high-throughput sequencing data*, de Sonja Althammer, Juan González-Vallinas, Cecilia Ballaré, Miguel Beato i Eduardo Eyras.

3.1.1. GFFReader

La primera feina de desenvolupament proposada va ser un lector de fitxers en format GFF (*General Feature Format)*, també anomenat GTF (*General Transfer Format*). Aquest format s'utilitza per anotar informació sobre regions i característiques del genoma. La motivació per al seu desenvolupament és la manca d'eines que permetin filtrar regions del genoma segons criteris determinats per l'usuari.

Cada línia d'un fitxer GFF representa una característica del genoma i consta de deu camps separats per tabulador, dels quals vuit són obligatoris. Aquests camps són:

1. Nom de seqüència. Generalment és el cromosoma al qual pertany la característica.
2. Font de la característica. Especifica d'on prové la informació presentada.
3. Tipus de característica. Descriu quin tipus d'element genòmic representa la regió.
4. Posició d'inici. Coordenada dins la seqüència en què comença la regió.
5. Posició de final. Coordenada dins la seqüència en què finalitza la regió.
6. Puntuació. Valor en coma flotant o '.', en cas que no s'especifiqui.
7. *Strand*. Pot ser '+', '-' o, en cas que no estigui especificat, '.'
8. *Frame*. Especifica que la primera base de la característica correspon a la primera base d'un codó. Pot ser '0', '1', '2' o, en cas que no estigui especificat, '.'
9. (Opcional) Atributs. Conté informació diversa relacionada amb la regió, amb una estructura “clau-valor”, i amb els elements separats per punt i coma. Entre els atributs destaquen l'identificador de la característica, l'identificador del node superior a la jerarquia, i el tipus de trànscript.
10. (Opcional) Comentaris. Altra informació proporcionada pel proveïdor del fitxer.

A l'annex A es pot veure un fragment de fitxer GFF. L'especificació oficial es pot trobar a http://www.sanger.ac.uk/resources/software/gff/spec.html

GFFReader va començar com un conjunt de classes independents de Pyicos, que s'utilitzava per llegir un fitxer d'entrada i generar una estructura de dades amb el seu contingut.

Durant el desenvolupament de GFFReader es van trobar una sèrie d'inconvenients del format, que dificulten la seva lectura i anàlisi:

- Tot i que representa una jerarquia, l'especificació no força a mantenir les línies en un ordre concret. A causa d'això, no es pot fer cap suposició sobre la jerarquia entre línies properes.
- El novè camp, que conté atributs com l'identificador de regió, el tipus i la relació amb altres regions, és opcional. En cas que no estigui especificat, no es pot saber a quina característica pertany cada regió (per exemple, a quin gen pertany un exó).
- No existeix una forma estàndard de donar nom als tipus de regions, ja que els autors del format només proporcionen algunes recomanacions. Així doncs, el proveïdor del fitxer pot anomenar-los com vulgui.

La causa d'aquests inconvenients és l'aparició d'estàndards *de facto*, sense cap conveni entre les organitzacions de recerca. Això es podria solucionar si s'utilitzés algun format jeràrquic i estricte, com l'XML.

Disseny i implementació

En primer lloc, es va dissenyar la classe *AnnotationRegion*, per representar tota la informació d'una regió. És una classe abstracta (tot i que Python no implementa explícitament aquest tipus de classes, mai se'n crea una instància) que s'utilitza com a base de tres classes concretes: *AnnotationGene*, *AnnotationTranscript* i *AnnotationExon*. Aquest disseny facilita la incorporació de nous tipus de regions, en cas que s'hagués de fer una futura ampliació. Els seus atributs són els mateixos que els camps d'una línia del fitxer, a més d'una llista amb les regions que conté dins. També implementa mètodes per a comparar objectes del seu tipus i per a buscar regions internes segons el seu identificador.

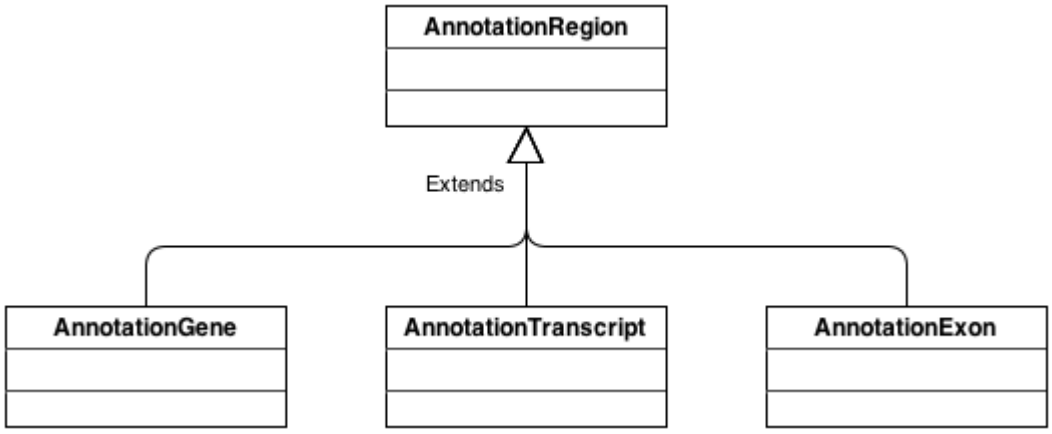

Figura 2: Classes de GFFReader

La primera solució que es va desenvolupar per llegir el fitxer carregava totes les regions a memòria (mitjançant una llista amb els gens) i buscava les relacions entre elles durant l'execució. Així es podia trobar la relació entre dues regions independentment de la zona del fitxer en què es trobessin. Aquesta solució no va ser considerada adequada, ja que requeria d'una gran quantitat de memòria disponible (entre 1 i 2 GB per al genoma sencer) i els temps d'execució del lector eren superiors a una hora. Per intentar millorar aquests factors, es va afegir una llista per emmagatzemar les referències a tots els trànscripts trobats, però el temps d'execució no va millorar. Degut a això i a l'excessiu ús de memòria, es va haver de descartar la solució.

Seguidament es va adoptar una nova estratègia. En primer lloc, s'ordena acumulativament el fitxer segons els camps *nom_seqüència, posició_inici, posició_fi* (de forma descendent) i *tipus_característica*. D'aquesta manera, queda el fitxer ordenat primer segons el nom del cromosoma, després segons posició (cada posició d'inici és igual o més gran que l'anterior i el mateix amb les posicions de fi, així les regions solapades s'ordenen de petita a gran) i finalment per tipus de seqüència (amb l'ajut d'una funció que assigna valors als gens, trànscripts i exons segons l'ordre jeràrquic). Posteriorment es llegeixen les regions assumint que mai es trobarà una característica amb posició d'inici inferior a les llegides i que tots els trànscripts estaran relacionats amb un gen que ja s'ha trobat (i el mateix amb exons i trànscripts). Gràcies a aquesta propietat i a la operació *yield* de Python (implementa un generador, que permet retornar valors a cada iteració d'un bucle), es poden eliminar de la memòria les característiques que ja s'han trobat completes, cosa que millora considerablement el temps d'execució.

Un altre aspecte que es va haver de tenir en compte és el tipus de característica.

Un fitxer GFF pot contenir informació especulativa o no confirmada sobre el genoma, que generalment no interessa utilitzar en anàlisis. Per a resoldre-ho, es va afegir un paràmetre de tipus llista *trànscripts_acceptats*, que permet a l'usuari seleccionar els tipus d'informació que llegirà el programa (és a dir, totes les regions amb un tipus que no aparegui a la llista seran descartades). Per defecte, la llista només conté *protein_coding*, que indica que el gen codifica una proteïna i s'ha confirmat experimentalment.

A continuació podem veure el pseudocodi de l'estructura principal de l'algorisme, que retorna cada gen amb els seus trànscripts i exons:

```
func llegir_gff(fitxer, trànscripts_acceptats):
    fitxer := ordenar_fitxer(nom_seq, inici, -fi, tipus)
    regions := []
    per cada línia del fitxer:
        camps := analitzar_línia(línia)
        si camps['tipus_trànscript'] és dins trànscripts_acceptats:
            si camps['tipus_característica'] == “GEN”:
                si gen no solapa amb gen anterior:
                    retornar cada gen de regions
                    regions := []
                regions := regions + Gen(camps)
            si no, si camps['tipus_característica'] == “TRÀNSCRIPT”:
                gen := busca_regió(regions, id=camps['id_gen'])
                gen[trànscripts] := gen[trànscripts] + Trànscript(camps)
            si no, si camps['tipus_característica'] == “EXÓ”:
                trànscripts := busca_regió(regions, id=camps['id_trànscript'])
                trànscripts[exons] := trànscripts[exons] + Exó(camps)
    retornar cada gen de regions
```

La funció *analitzar_línia* comprova que els camps d'una línia són correctes i retorna els seus valors en una tupla.

Durant el desenvolupament es van trobar complicacions amb el tipus de trànscript d'algunes regions. Alguns dels gens són de tipus *polymorphic_pseudogene*, que significa que el gen codifica proteïna en alguns individus però no en altres. Això provocava que el programa no els considerés com a codificadors de proteïnes i els eliminés, tot i que alguns dels seus trànscripts sí que eren de tipus *protein_coding*. Finalment, degut a la baixa

quantitat de gens de tipus *polymorphic_pseudogene* (26 sobre un total de 56.563, segons la versió 16 de Gencode), es va decidir descartar-los per defecte.

3.1.2. Cerca de regions

Amb el lector de GFF ja funcionant, es va demanar la implementació de diversos algorismes per a la cerca de regions segons les seves característiques. Aquestes poden ser *exons*, *introns*, *regions intergèniques*, *regions intragèniques* i *TSS*.

Totes les funcions han estat implementades com a generadors de Python, mitjançant la instrucció *yield* en comptes de *return*. Això ha permès millorar l'eficiència en memòria del programa, perquè es retorna cada regió quan s'ha trobat completa.

Cerca d'exons

Dins d'un gen, els exons són les regions que s'acaben codificant en una proteïna, mitjançant el procés de *splicing*.

En un fitxer GFF, els exons es representen explícitament i es poden trobar interpretant el tercer camp de cada línia. Gràcies a aquest fet, el disseny de l'algorisme no va portar complicacions.

A la imatge següent es mostra de forma esquemàtica la detecció d'exons dins el genoma. A la part inferior de la imatge es representa una regió amb dos gens, els exons dels quals són marcats en negre. Les zones de color blau corresponen als exons retornats.

Figura 3: Cerca d'exons

Pseudocodi de l'algorisme:

```
func llegir_exons():
    gens_actuals := []
    per cada gen de llegir_gff():
        si gen solapa amb gens_actuals:
            gens_actuals := gens_actuals + gen
        si no:
            exons := []
            per cada gen2 de gens_actuals:
                tràns cripts := gen2 -> tràns cripts
                per cada tràns cript de tràns cripts:
                    exons := exons + (tràns cript -> exons)
            ordena(exons)
            retorna cada exó de exons
            gens_actuals := [gen]
    ordena i retorna exons de gens_actuals
```

S'ha de considerar que quan acaba el bucle principal encara poden quedar-hi gens per processar. Per fer-ho, cal ordenar els seus exons i retornar-los, seguint la mateixa estructura que dins el bucle.

Cerca d'introns

Els introns són els elements oposats als exons, és a dir, les regions pertanyents a un gen que no s'acaben codificant en una proteïna.

A la imatge següent es mostra l'efecte de la cerca d'introns dins el genoma. Les

zones de color blau representen els introns retornats, que corresponen a les regions intragèniques en què no hi ha exons.

Figura 4: Cerca d'introns

Degut a què els introns no apareixen explícitament al fitxer GFF, la seva cerca es basa en analitzar les regions internes a cada gen que no pertanyin a cap exó. També cal tenir en compte els gens que puguin estar total o parcialment solapats en una regió; en aquest cas, l'algorisme dissenyat els tracta com si fossin un de sol.

Pseudocodi de l'algorisme:

```
func llegir_introns():
    gens_actuals := []
    per cada gen de llegir_gff():
        si gen solapa amb gens_actuals:
            gens_actuals := gens_actuals + gen
        si no:
            introns := busca_introns(gens_actuals)
            retorna cada intró de introns
            gens_actuals := [gen]
    introns := busca_introns(gens_actuals)
    retorna cada intró de introns

func busca_introns(gens):
    exons := []
    per cada gen de gens:
        exons := exons + (gen -> exons)
    ordena(exons)
    per i dins [0 .. mida(exons) – 1]:
        si (exons[i] -> posició_inici) < (exons[i+1] -> posició_final):
            retorna Intró(exons[i] -> posició_inici, exons[i] -> posició_final)
```

Com es pot veure, la seva estructura és similar a la de `llegir_exons`, tot i que utilitza la funció auxiliar `busca_introns`, que busca i retorna tots els introns a partir d'una llista de gens.

Cerca de regions amb finestra lliscant

Una forma de buscar regions dins el genoma és utilitzant finestres lliscants. Amb aquesta tècnica es recorre el genoma de forma seqüencial tot llegint-ne segments d'una mida fixa i amb una separació determinada entre ells. Aquesta separació ha de ser inferior o igual a la mida de la finestra, per a què el recorregut sigui continu.

Per al disseny d'aquest algorisme es van prendre les següents decisions:

- En el cas que una regió sencera sigui més petita que la mida de la finestra, no serà retornada. S'ha optat per aquesta solució perquè es considera que l'investigador vol descartar les zones de mida inferior a la que ha especificat.
- En el cas que la posició final de l'últim segment superi la de la regió, el desplaçament serà inferior a l'especificat, de manera que les dues posicions finals coincideixin en el mateix punt. En altres paraules, es dóna més importància a què tots els segments siguin de la mateixa mida.

A les següents imatges es mostra l'efecte de la cerca amb finestra lliscant. En algun dels casos es pot notar que el desplaçament de l'últim segment de la regió és més curt.

Figura 5: Cerca amb finestra: regions intergèniques

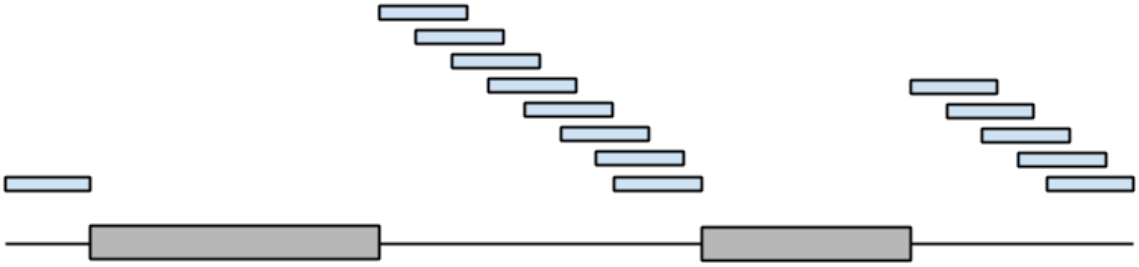

Figura 6: Cerca amb finestra: regions intragèniques

Pseudocodi de l'algorisme per regions intergèniques:

```
func finestres_intergèniques(mida, distància):
    gens_actuals := []
    per cada gen de llegir_gff():
        si gen solapa amb gens_actuals:
            gens_actuals := gens_actuals + gen
        si no:
            punt_inici := maxim_final(gens_actuals)
            punt_final := gen -> posicio_inici
            genera_finestres(punt_inici, punt_final, mida, distància)
            gens_actuals := [gen]
    punt_inici := màxim_final(gens_actuals)
    punt_final := mida_cromosoma(gen -> nom_seqüència)
    genera_finestres(punt_inici, punt_final, mida, distància)
```

Pseudocodi de l'algorisme per regions intragèniques:

```
func finestres_intragèniques(mida, distància):
    per cada gen de llegir_gff():
        gens_actuals := []
        si gen solapa amb gens_actuals:
            gens_actuals := gens_actuals + gen
        si no:
            punt_inici := minim_inici(gens_actuals)
            punt_final := maxim_final(gens_actuals)
            genera_finestres(punt_inici, punt_final, mida, distància)
            gens_actuals := [gen]
    punt_inici := mínim_inici(gens_actuals)
    punt_final := màxim_final(gens_actuals)
    genera_finestres(punt_inici, punt_final, mida, distància)
```

Els dos algorismes utilitzen la funció `genera_finestres`, que retorna tots els segments entre `punt_inici` i `punt_final`, amb la mida i distància determinades.

Cerca de TSS

Els TSS (*Transcript Start Site*) són les regions del genoma en què s'inicia el procés de transcripció. Degut a què consten d'un sol punt a l'inici de trànscript, resulta útil introduir-hi un cert nombre de bases a l'inici i final, per a retornar intervals de mida superior a 1. A més, la seva localització depèn de l'*strand*, de forma que els trànscripts positius tenen el TSS a la posició d'inici, i els negatius a la posició de final.

A la imatge següent es pot veure com Pyicoteo retorna els TSS. El punt d'inici de cada trànscript es troba representat de color vermell i els gens tenen *strands* diferents (el de l'esquerra és positiu i el de la dreta, negatiu). Les zones retornades són intervals al voltant del punt.

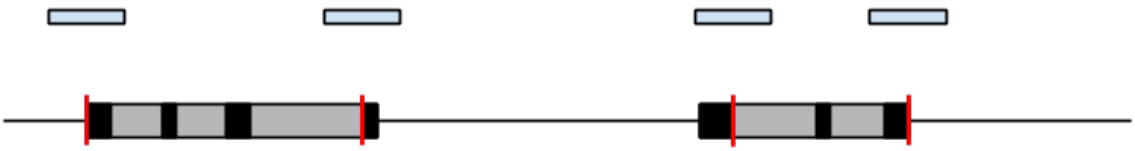

Figura 7: Cerca de TSS

Pseudocodi de l'algorisme:

```
func llegir_tss(suma_inici, suma_final):
    gens_actuals := []
    per cada gen de llegir_gff():
        si gen solapa amb gens_actuals:
            gens_actuals := gens_actuals + gen
        si no:
            per cada trànscript de gen -> trànscripts:
                si strand == '+' o strand == '.':
                    tss = trànscript -> posició_inici
                    retorna TSS(tss – suma_inici, tss + suma_final)
                si no:
                    tss = trànscript -> posició_final
                    retorna TSS(tss + suma_inici, tss – suma_final)
```

```
gens_actuals := [gen]
```

Integració a Pyicoteo

Per integrar els algorismes de cerca de regions a Pyicoteo es va afegir un nou paràmetre `--region-magic` a la línia de comandes. Aquest rep el tipus de regió que es vol generar i la resta de paràmetres específics a cadascuna.

Per cada tipus de regió possible, el paràmetre pot tenir les següents formes:

- `--region-magic exons`
- `--region-magic introns`
- `--region-magic slide [window_size] [window_step] [inter/intra]`
- `--region-magic tss [add_start] [add_end]`

Finalment, per generar els fitxers amb les regions trobades es va dissenyar la classe RegionWriter. Aquesta s'encarrega d'interpretar els paràmetres rebuts amb `--region-magic` i escriure les regions generades a un fitxer de format BED.

3.1.3. Unit testing

Per completar el desenvolupament de la cerca de regions es van dissenyar diversos tests unitaris. Gràcies a aquests tests es pot comprovar si el programa segueix funcionant bé en cas que en el futur es faci algun canvi en el codi.

Per implementar-los, es va crear un nou mòdul *TestRegions* dins el conjunt de tests que ja existien a Pyicoteo. Aquesta classe conté les operacions per a la comprovació de pyicoregion, descrites a continuació:

- `setUp()`: Escriu un fitxer GFF temporal amb regions de prova arbitràries. En total, conté dos cromosomes diferents: *chr1*, amb dos gens, i *chr2*, amb un gen.
- `tearDown()`: Elimina tots els fitxers temporals creats per les proves. S'utilitza la

llista *tempfiles* per a emmagatzemar-ne les rutes.

- `test_get_exons()`: Comprova que la funció de cerca d'exons retorna els 7 exons del fitxer temporal.
- `test_get_introns()`: Comprova que la funció de cerca d'introns retorna els 3 introns del fitxer temporal.
- `test_generate_windows_inter()`: Comprova que la cerca de regions intergèniques amb finestra de 1000 bases i separació de 500 funciona correctament. També genera un nou fitxer temporal amb mides dels cromosomes.
- `test_generate_windows_intra()`: Comprova que la cerca de regions intragèniques amb finestra de 1000 bases i separació de 500 funciona correctament. A diferència de la cerca de regions intergèniques, no genera cap fitxer amb les mides dels cromosomes, ja que la funció no el necessita.
- `test_get_tss()`: Comprova que la funció de cerca de TSS retorna els 3 TSS del fitxer temporal de forma correcta, amb la suma de 100 bases a l'inici i 50 al final.

3.1.4. Marcatge de regions

L'eina pyicoenrich permet a l'investigador fer comparacions entre dos conjunts de dades. El resultat que retorna és un fitxer de text amb els valors dels càlculs i una gràfica de la comparació, generada amb la biblioteca *Matplotlib*.

Una de les funcionalitats que es va demanar implementar va ser el marcatge de regions seleccionades per l'usuari a la gràfica. Per fer-ho, l'investigador proporciona un conjunt de regions que considera interessants i el programa les marca a la gràfica, en cas que hi apareguin.

Per introduir les regions que es vol marcar, es va afegir un nou paràmetre a la línia de comandes: `--interesting-regions`. Ha d'anar acompanyat per la ruta a un fitxer de text amb els identificadors de les regions, separats per salts de línia.

Seguidament, es va modificar la funció de dibuix de la gràfica. S'hi va afegir una llista per emmagatzemar les coordenades de les regions amb identificador a *interesting-*

regions. Aquestes coordenades es dibuixen després d'haver representat la resta de regions, per a què quedin els punts en un primer pla.

A la següent figura es mostra la gràfica resultant de pyicoenrich amb marcatge de regions. La part superior representa una distribució de punts aleatòria, per calcular els marges que separen les regions significants (de color vermell a la gràfica inferior) de les no significants (de color negre a la gràfica inferior). Els punts verds representen les regions que l'investigador ha desitjat marcar.

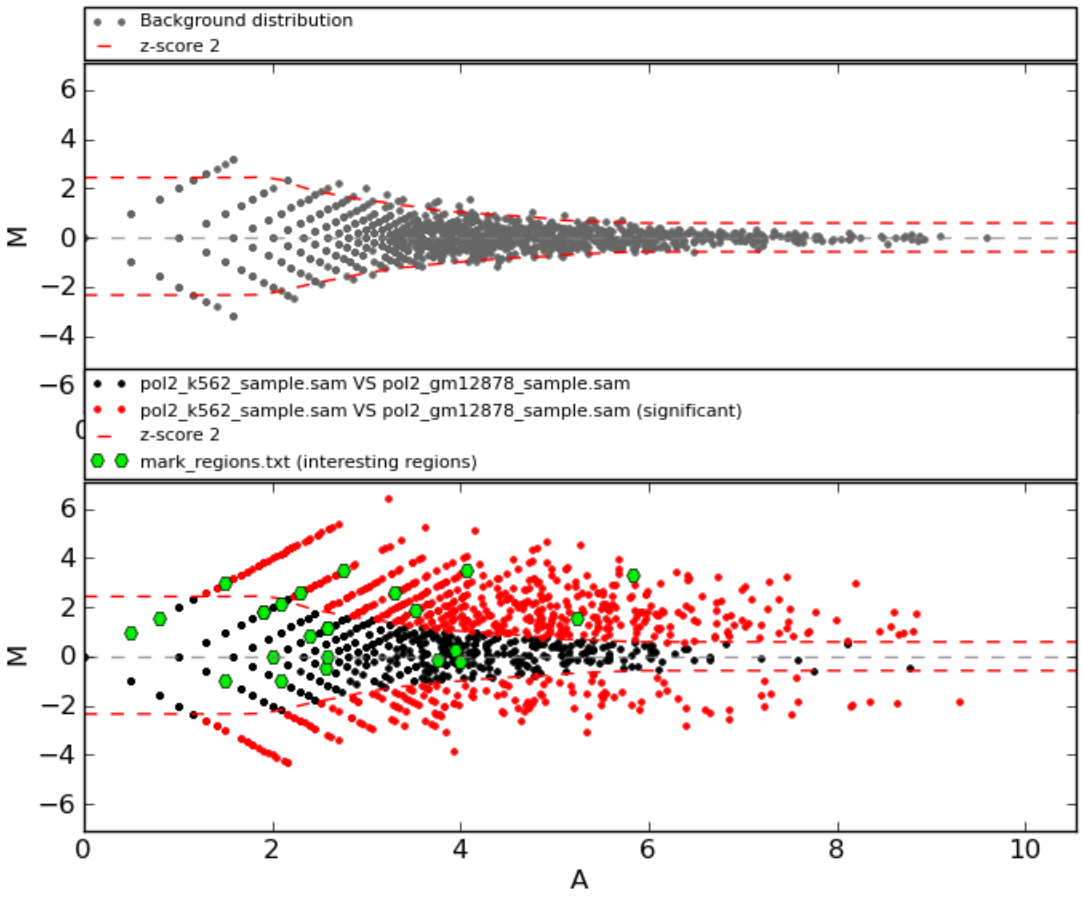

Figura 8: Pyicoenrich: marcatge de regions

També relacionat amb el marcatge de regions, es va demanar l'addició d'un nou paràmetre `--disable-significant-color`, per a què tots els punts de la gràfica tinguin el mateix color. Aquesta funcionalitat és útil en els casos que l'investigador vulgui mostrar la gràfica sense remarcar les regions significants de color vermell.

3.1.5. Lector flexible

Per representar dades genòmiques existeixen diversos formats, com BED, SAM, WIGGLE o ELAND. Tots ells s'emmagatzemen com a fitxers de text pla i la informació que representen és semblant. Això és degut a què cadascun és dissenyat per una sola organització, sense que hi hagi cap acord per crear un estàndard.

De la informació que contenen, els camps considerats importants són el nom de seqüència, les posicions d'inici i final i l'*strand*. De vegades els investigadors utilitzen programes d'edició de text per generar fitxers que no s'ajusten a cap format concret. A causa d'això, es va demanar implementar un nou lector per a Pyicoteo, que pogués llegir fitxers d'aquest tipus i convertir-los a altres formats.

En primer lloc, es van afegir els següents paràmetres:

- `--f-custom-in`: Rep quatre enters, que representen la columna en què es troben els camps, per ordre: *seqname*, *start*, *end* i *strand*. L'especificació de *strand* és opcional.
- `--custom-in-sep`: Caràcter o cadena de caràcters que separen les columnes del fitxer d'entrada.
- `--f-custom-out`: De la mateixa manera que `--f-custom-in`, especifica en quina columna s'escriuran els camps, per ordre: *seqname*, *start*, *end* i *strand*.
- `--custom-out-sep`: Caràcter o cadena de caràcters amb què se separaran les columnes al fitxer de sortida.

Seguidament, es va dissenyar la classe *CustomReader*. És una especialització de la classe abstracta *Reader*, que proporciona operacions bàsiques per a la lectura d'objectes *Cluster* de Pyicoteo. Anàlogament, es va dissenyar la classe *CustomWriter*, com a especialització de la classe abstracta *Writer*.

CustomReader implementa la funció `read_line`, que converteix una línia llegida en un objecte Cluster. Utilitzant els paràmetres `f-custom-in` i `custom-in-sep` és capaç d'extreure la informació necessària per crear-lo. A partir d'aquí, l'objecte creat es pot utilitzar en els experiments de Pyicoteo, com si provingués de qualsevol format.

CustomWriter implementa la funció `write_line`, que converteix un objecte Cluster

en una cadena de text. De manera semblant a *CustomReader*, utilitza els paràmetres `f-custom-out` i `custom-out-sep` per generar el fitxer de sortida.

3.1.6. Pyicoteo

Durant la realització del projecte es va decidir reestructurar el codi de Pyicos, que va passar a dir-se Pyicoteo. L'objectiu d'aquest canvi és millorar el software des d'un punt de vista arquitectònic i facilitar la creació de noves eines a la suite.

Un dels canvis més importants va ser la divisió de l'executable *pyicos* en mòduls i comandes múltiples:

- El conjunt d'operacions de manipulació de dades genòmiques (*extend*, *push*, *subtract*, *convert*...) segueix anomenant-se *pyicos*.
- Les operacions d'enriquiment es mouen a *pyicoenrich*.
- Els *peak-callers* passen a anomenar-se *pyicoclip* i *pyicoller*.
- La part de lectura i cerca de regions passa a dir-se *pyicoregion*, com a comanda independent (tot i que les operacions segueixen utilitzant-se com a complement a *pyicoenrich*).
- L'execució de fitxers de protocol propis de Pyicoteo es realitza amb el mòdul *pyicotrocol*.

Finalment es va haver d'actualitzar el manual d'ús del programa, per reflectir els canvis efectuats.

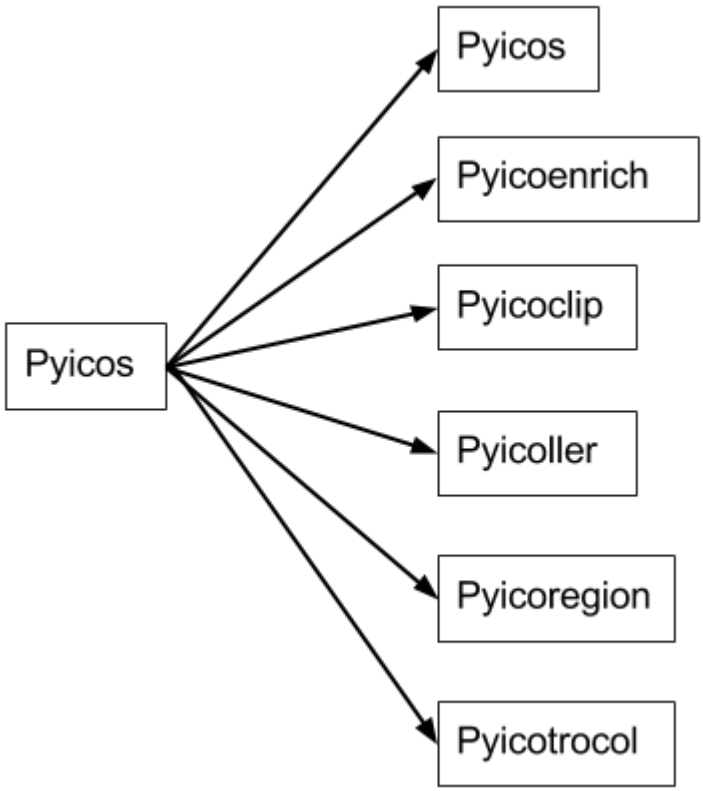

Figura 9: Pyicos: Separació en mòduls

Aquests canvis també van suposar la separació del *parser* de la línia de comandes en diversos *subparsers*, un per a cada nova comanda executable. D'aquesta manera el codi queda més net i es facilita la incorporació de noves funcionalitats.

3.1.7. UCSCUpload

Una altra funcionalitat que es va proposar implementar a Pyicoteo va ser la de carregar dades al *UCSC Browser* (http://genome.ucsc.edu/). És una aplicació web que proporciona un interfície per a visualitzar característiques del genoma des del navegador. Permet carregar fitxers de dades i veure'ls representats de forma gràfica, fet que ajuda a la comprensió de les dades per part dels investigadors.

La motivació d'aquesta funcionalitat és proporcionar una manera ràpida de carregar dades al *Browser*, sense haver de fer-ho amb un navegador web.

Després de la lectura de la documentació i diversos manuals d'ajuda del Browser, es va concloure que la implementació d'aquesta funcionalitat no es pot realitzar, ja que l'aplicació no permet la càrrega de fitxers des de fora del navegador. Això és per a evitar l'abús (voluntari o involuntari) de l'aplicació, ja que les dades carregades acostumen a tenir una mida considerable i els servidors podrien col·lapsar-se en el cas que s'enviés una gran quantitat de fitxers.

Una alternativa considerada va ser l'ús de peticions HTTP de tipus GET i POST, per simular l'ús d'un navegador. Aquesta solució es va descartar per la dificultat d'implementació i els possibles canvis interns que es produeixin al *Browser*.

A la següent imatge es pot veure una captura de pantalla de la interfície del *UCSC Browser*. Aquesta mostra un fragment del cromosoma 1, amb la informació d'alguns gens i els exons detectats per pyicoregion.

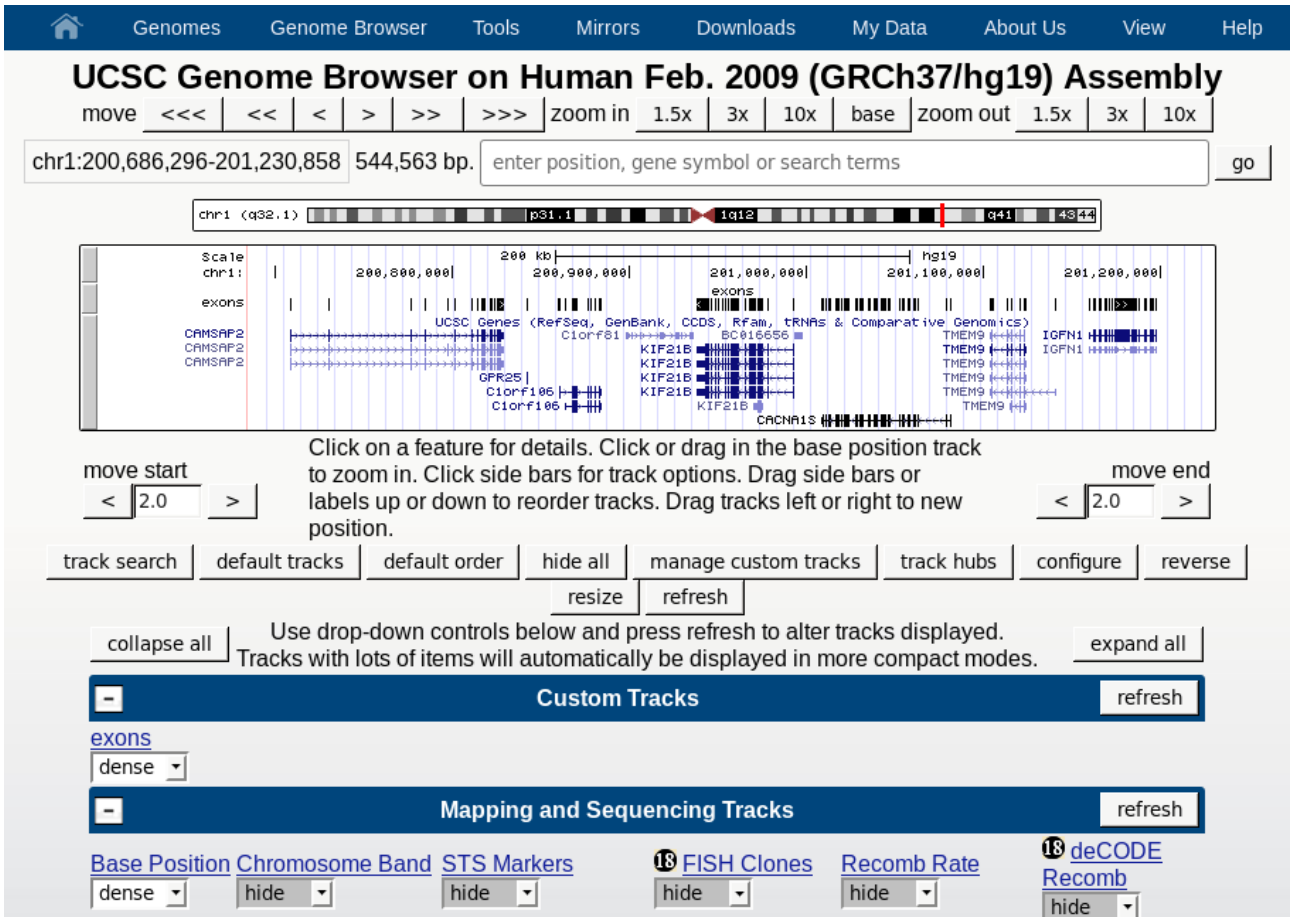

Figura 10: Interfície del UCSC Browser

3.1.8. Manual d'ús i documentació

Per donar a conèixer el funcionament de Pyicoteo als usuaris que vulguin utilitzar-lo, es proporciona un manual d'ús en format HTML. Aquest es genera amb Sphinx, el generador de documentació de Python. És un software que converteix reStructuredText a HTML i altres formats, generalment utilitzat en pàgines web.

Com a exemple, a l'annex B es pot trobar el fitxer reStructuredText amb el manual d'ús de pyicoregion.

3.1.9. Comparació de Pyicoteo amb altres eines

La fase final del projecte va ser dedicada a investigar eines amb funcionalitats semblants a Pyicoteo. No hi ha cap programa amb les mateixes funcionalitats, ja que cadascun s'adapta a les necessitats dels investigadors que les desenvolupen. A més, els mètodes estadístics que implementen per fer anàlisis són diferents i no existeixen mètodes fiables per comparar el seu funcionament.

BEDTools

Igual que Pyicoteo, BEDTools és una suite que proporciona diverses eines per a la manipulació i anàlisi de dades genòmiques. Està programat en C++ i es publica sota llicència GPLv2.

La seva semblança amb Pyicoteo és la implementació d'eines de manipulació de dades. Entre elles hi trobem:

- Cerca i agrupament de lectures solapades
- Extensió, intersecció, subtracció i ordenació de lectures
- Conversió entre formats de fitxers

- Generació de finestres lliscants (implementat a pyicoregion)

El programa es pot descarregar a http://code.google.com/p/bedtools/ i la seva documentació es pot consultar a http://bedtools.readthedocs.org/

Bioconductor

Bioconductor és una col·lecció de paquets per al software estadístic R. Agrupa una gran quantitat de mòduls (més de 600), desenvolupats per diferents autors.

Les funcionalitats que implementa són molt diverses, ja que pretén recollir un ampli espectre d'eines d'anàlisi genòmica. Entre les eines que proporciona, les més semblants a Pyicoteo s'agrupen en els paquets:

- *Clustering*: Algorismes per a l'agrupament de lectures.
- *Enrichment*: Conjunt d'eines per a la comparació entre experiments.

La seva pàgina web és http://www.bioconductor.org/

Vancouver Short Reads

Vancouver Short Reads és un paquet centrat en l'anàlisi de dades ChIP-seq mitjançant *peak-calling*. Està desenvolupat amb Java per Anthony Fejes, de la Universitat de British Columbia.

Les seves funcionalitats s'agrupen en tres categories:

- *FindPeaks*. Implementa mètodes estadístics per a la cerca de pics en dades genòmiques. Pyicoteo implementa eines semblants, com *pyicoller*, *pyicoclip* i *pyicoenrich*.
- Base de dades de variacions. Proporciona eines per crear bases de dades amb informació genòmica. Pyicoteo no implementa cap funcionalitat semblant.
- Altres eines. De la mateixa manera que Pyicoteo, Vancouver Short Reads conté eines per manipular dades genòmiques, com la conversió entre formats i l'ordenació de fitxers.

La seva pàgina web es troba a http://vancouvershortr.sourceforge.net/

3.2. Galaxy

El projecte Galaxy (http://galaxyproject.org) és una plataforma web per a la realització d'anàlisis bioinformàtics. L'equip principal de desenvolupadors està format per membres de la Universitat de Penn State i la Universitat d'Emory. A més, al ser un projecte de codi obert (amb llicència *Academic Free License 3.0*), permet que altres desenvolupadors que ho desitgin puguin aportar-hi el seu codi.

Els seus objectius són:

- Accessibilitat: Els usuaris no cal que tinguin experiència en programació. Gràcies a això, els investigadors amb pocs coneixements informàtics poden fer servir les eines sense haver d'aprendre el funcionament de la línia de comandes.
- Reproductibilitat: Els experiments realitzats poden ser repetits per qualsevol usuari, ja que el programa emmagatzema els valors amb què s'ha executat cada experiment.
- Transparència: Els usuaris poden publicar els seus experiments i compartir-los entre ells.

Per a què tothom pugui provar el seu funcionament, els desenvolupadors ofereixen un servidor públic amb eines de mostra: http://usegalaxy.org

Està programat en Python, cosa que facilita la modificació dels seus components. Tot i això, aquest llenguatge té els seus inconvenients, especialment en concurrència, ja que l'intèrpret només permet mantenir un fil d'execució al processador.

El programa funciona en sistemes *Unix-like* i conté tots els components necessaris per a la seva execució (excepte l'intèrpret de Python, que l'ha de proveir l'usuari). A més, facilita la substitució d'algun dels seus components per a millorar-ne el rendiment:

- En comptes de tenir-lo instal·lat en un servidor propi, es pot desplegar en un servidor de computació *cloud* com Amazon EC2, per a beneficiar-se de la computació distribuïda. Amb això es poden reduir les limitacions de Python amb la

concurrència, esmentades anteriorment. També es pot utilitzar *Apache* o *nginx* com a distribuïdors de la càrrega, en cas que es disposi de múltiples servidors. Aquestes opcions es van descartar pel cost econòmic que implica contractar el servei d'Amazon i per la disponibilitat d'un servidor propi.

- Un altre dels components que es pot reemplaçar és la base de dades. Per defecte, *Galaxy* utilitza *SQLite* per emmagatzemar tota la informació dels usuaris i experiments (excepte les pròpies dades i resultats). En el cas que tingués una gran quantitat d'usuaris i es requerís un alt nivell de concurrència, es podria canviar per *PostgreSQL* o *MySQL*, que ofereixen un alt rendiment. En aquest projecte no s'ha considerat necessari degut a la complexitat que introdueix l'administració d'un sistema gestor de bases de dades.

La motivació d'utilitzar Galaxy és donar a conèixer l'eina Pyicoteo, desenvolupada dins el grup de recerca, a la resta d'investigadors bioinformàtics.

3.2.1. Instal·lació del servidor

En primer lloc, es va descarregar el servidor *Galaxy*. La forma recomanada de fer-ho és des del repositori oficial, utilitzant el software de gestió de versions *Mercurial*, ja que facilita la seva actualització.

El repositori de Galaxy es troba a l'adreça https://bitbucket.org/galaxy/galaxy-dist/

Per arrencar el servidor només cal executar el fitxer run.sh, localitzat al directori arrel de Galaxy. Si tot és correcte, accedint a http://localhost:8080 des d'un navegador es mostra la pantalla principal del programa.

3.2.2. Configuració del servidor

El servidor Galaxy es configura mitjançant fitxers INI i XML localitzats al directori d'instal·lació. Els més importants són universe_wsgi.ini i tool_conf.xml, que permeten

configurar els paràmetres del servidor i la localització de les eines que s'hi integren, respectivament.

La configuració per defecte de universe_wsgi.ini ja conté la majoria d'opcions necessàries per a què funcioni correctament. Tot i això, cal afegir l'adreça electrònica de l'usuari administrador, mitjançant l'opció *admin_users*.

A l'hora d'instal·lar Galaxy en un servidor de producció, cal eliminar les funcionalitats de depuració per evitar l'accés a dades restringides (per part dels usuaris:

```
debug = False
use_interactive = False
```

Quan el servidor està en funcionament, es poden crear els usuaris. Com a mínim s'ha de registrar l'administrador, amb l'adreça especificada a universe_wsgi.ini. Aquest usuari tindrà accés al panell d'administració, que permet realitzar tasques de manteniment del servidor. Entre aquestes tasques destaquen l'assignació de permisos als usuaris registrats i la gestió de quotes de disc.

Quotes de disc

Les dades que s'utilitzen en anàlisis acostumen a tenir una mida considerable, amb fitxers que poden superar 1 GB. A causa d'aquest fet, és desitjable poder controlar l'espai del servidor que ocupen els fitxers dels usuaris. Per a fer-ho, s'utilitzen les quotes.

A la configuració per defecte, les quotes estan desactivades. Per a evitar que els fitxers de dades enviats pels usuaris col·lapsin el servidor, es van activar. Per a fer-ho, en primer lloc es va modificar el fitxer de configuració universe_wsgi.ini amb les opcions:

```
enable_quotas = True
allow_user_dataset_purge = True
```

Això permet que els administradors puguin aplicar quotes als usuaris, des de la interfície de gestió del servidor. Així doncs, accedint a l'apartat *admin* de la interfície web com a usuari administrador es poden modificar els paràmetres de les quotes.

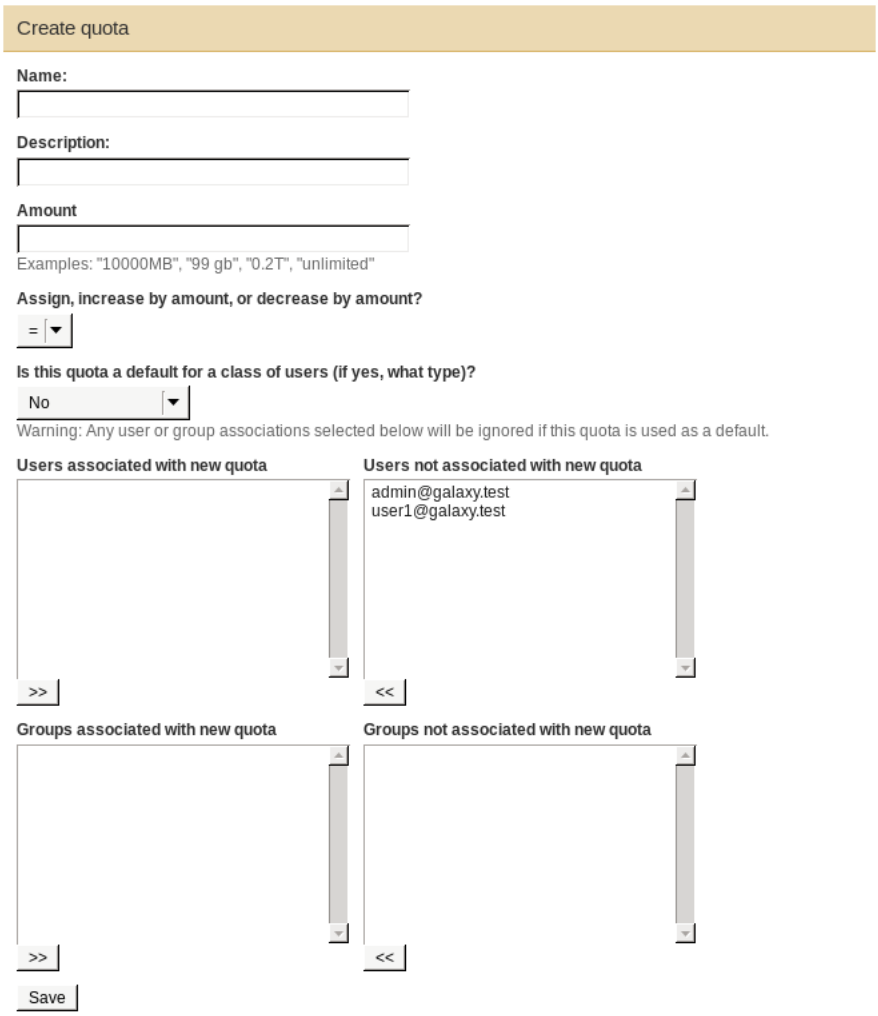

Figura 11: Galaxy: Interfície per crear quotes

3.2.3. Integració de Pyicoteo

La integració de noves eines a l'entorn Galaxy es realitza amb fitxers XML, que descriuen com s'executa el programa, quins paràmetres accepta i de quin tipus són i què retorna com a resultat. L'especificació de totes les etiquetes que suporta es pot trobar a http://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax

L'estructura bàsica dels fitxers de configuració és la següent:

- `<tool>`: Node principal que descriu l'eina. Requereix els atributs *id* (identificador intern de l'eina), *name* (nom que es mostrarà a la interfície de Galaxy) i *version* (versió de l'eina, per evitar conflictes en cas que s'actualitzi).

- `<description>`: Proporciona una curta descripció de l'eina a la barra lateral de Galaxy.
- `<command>`: Conté la comanda que s'ha d'executar per realitzar l'experiment. Utilitza una sintaxi que permet l'ús d'estructures condicionals simples i la lectura dels paràmetres introduïts per l'usuari.
- `<inputs>`: Secció en què s'especifiquen tots els paràmetres d'entrada de l'eina.
  - `<param>`: Cada paràmetre es defineix amb un node d'aquest tipus. Els seus atributs són *name* (nom del paràmetre), *type* (tipus del paràmetre: *integer*, *float*, *text*, *select*...), *label* (nom que apareixerà a la interfície gràfica) i *format* (forma en què es representen les dades, generalment *tabular* o *text*, per acceptar la majoria de formats).
- `<outputs>`: Especificació dels paràmetres de sortida (és a dir, els fitxers que retorna l'eina).
- `<help>`: Permet proporcionar un manual d'ajuda, que es mostrarà al final de la pàgina.

L'especificació completa de la sintaxi XML es pot trobar a http://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax

Comanda d'execució

Mitjançant el node `<command>` s'especifica quina comanda s'ha d'executar per realitzar un experiment. El seu contingut és l'executable del programa i tots els seus paràmetres.

Els valors dels paràmetres d'entrada es llegeixen mitjançant el símbol '$' seguit del nom del paràmetre. En cas que es trobi dins un node condicional, cal especificar el nom del condicional abans del paràmetre, separats per un punt.

Alguns dels paràmetres són opcionals, o la seva existència depèn del valor d'altres paràmetres. Per tenir en compte aquests casos, Galaxy permet l'especificació d'expressions condicionals simples mitjançant expressions `#if`, `#elif`, `#else` i `#end if`.

Al següent exemple es pot veure la comanda utilitzada per pyicoregion, amb lectura

de paràmetres i expressions condicionals:

```
  <command interpreter="python">
      pyicoregion -output $output
            -F $output_format_cond.output_format
            $is_output_open
            --galaxy-workarounds
            --gff-file $gff_file
            --region-magic $region_magic_cond.region_magic
            #if $region_magic_cond.region_magic == "slide"
                  $region_magic_cond.window_size $region_magic_cond.window_step
$region_magic_cond.region_slide_cond.slide_type
                #if $region_magic_cond.region_slide_cond == "inter"
                    $region_magic_cond.region_slide_cond.chromlen_file
                #end if
            #elif $region_magic_cond.region_magic == "tss"
                $region_magic_cond.tss_add_start $region_magic_cond.tss_add_end
            #end if
  </command>
```

Paràmetres d'entrada

Els paràmetres d'entrada que es passen al programa s'especifiquen dins el node `<inputs>`. En cas que l'ús d'un paràmetre depengui del valor d'un altre, es poden utilitzar condicionals. La seva estructura és la següent:

- `<conditional>`: Conté un node paràmetre i un o més nodes de control de la condició. Requereix un atribut *name*, amb el nom que identifica el condicional.
    - `<param>`: Paràmetre el valor del qual serà avaluat en la condició.
    - `<when>`: Accepta un atribut *value*. Quan aquest coincideix amb el valor del paràmetre del mateix condicional, s'interpreten els nodes fills.

Paràmetres de sortida

La forma en què Galaxy emmagatzema les dades no permet especificar múltiples fitxers de sortida per una mateixa eina. Això va dificultar la integració de pyicoenrich, ja

que aquesta retorna dos fitxers: les dades en format de text pla i una imatge amb la gràfica. La solució a aquest problema va constar en modificar Pyicoteo per a què escrigui els resultats conjuntament a un fitxer HTML. Per a implementar-ho, es va utilitzar el motor de plantilles *Jinja2* i es va afegir el paràmetre `--html-output` al *parser*, per a què la funcionalitat fos opcional.

La plantilla dissenyada conté tres elements:

- Imatge: Mostra la gràfica retornada per pyicoenrich, en format PNG.
- Enllaç al fitxer: Permet descarregar el fitxer de text amb els resultats, ja que des de la interfície Galaxy es descarregaria tot l'HTML generat.
- Contingut del fitxer: Mostra els resultats en format de text pla.

La plantilla completa es pot trobar a l'annex D.

En el cas que l'usuari desitgi obtenir únicament el fitxer de text amb els resultats, pot desactivar l'ús de HTML. Galaxy permet canviar el tipus del fitxer de sortida mitjançant el node `<filter>`. S'afegeix dins un node `<data>` i conté la condició que s'ha de complir per a què s'apliqui aquell format de sortida.

També es va implementar una funció JavaScript per a mostrar i ocultar el text amb els resultats, però no va funcionar: per motius de seguretat Galaxy només permet un subconjunt de les etiquetes de l'estàndard HTML, generalment les de format de text i imatge. Aquest filtre es pot desactivar amb l'opció `sanitize_all_html` del fitxer de configuració universe_wsgi.ini, però finalment es va decidir no fer-ho, per les potencials vulnerabilitats que pot introduir.

Per adaptar la resta d'eines només es va especificar un sol paràmetre, amb format *tabular* i nom *output*.

Es pot trobar un exemple complet de descripció d'eina a l'annex C.

tool_conf.xml

Tenint implementats els fitxers de descripció de les eines, es van afegir al fitxer tool_conf.xml les rutes relatives als fitxers de configuració creats.

L'estructura de tool_conf.xml és la següent:

- `<toolbox>`: Node principal. Conté totes les eines.
  - `<section>`: Defineix un conjunt d'eines, generalment relacionades. Cal especificar els atributs *name* (nom que es mostrarà a la interfície Galaxy) i *id* (identificador de les eines)
    - `<tool>`: Cada node d'aquest tipus representa una de les comandes implementades. Només s'hi especifica la ruta (relativa des del directori *tools* de Galaxy) al fitxer de descripció de la comanda, amb l'atribut *file*.

A l'annex E es mostra el fitxer de configuració tool_conf.xml resultant de la integració de Pyicoteo.

Dificultats amb la integració

A l'hora d'executar les eines van sorgir diverses complicacions, causades per la forma en què Galaxy executa les comandes.

Una d'elles va ser la interpretació del resultat que es retorna a la línia de comandes. Per mostrar informació sobre l'execució del programa, Pyicoteo utilitza un *logger* que escriu els missatges al canal d'error estàndard del sistema. D'altra banda, Galaxy interpreta l'escriptura en aquest canal per part del programa com a execució incorrecta. Per solucionar-ho, es va fer ús dels nodes `<stdio>` i `<exit_code>`, que canvien el comportament de Galaxy a l'hora d'interpretar el resultat del programa. Imposant que només interpreti com a error les execucions que retornin un valor superior a 0 a la línia de comandes es va poder resoldre aquest inconvenient. El codi resultant és:

```
<stdio>
  <exit_code range="1:" level="error" />
</stdio>
```

També es van trobar problemes amb l'execució de Pyicoteo:

1. Galaxy filtra els valors introduïts als camps de text. Com a conseqüència, els paràmetres d'entrada que contenen caràcters especials no es reben bé a Pyicoteo.

2. Les execucions de pyicoenrich es realitzen en un subdirectori diferent al que utilitza per defecte. Això provoca que Pyicoteo no trobi alguns fitxers especificats com a rutes relatives.
3. La ruta relativa als fitxers de longitud de cromosomes és diferent a la de Pyicoteo.

Per resoldre'ls, es va afegir el paràmetre `--galaxy-workarounds` a la línia de comandes, que aplica les següents solucions:

1. Tots els paràmetres rebuts per línia de comandes s'analitzen i se substitueixen les cadenes de text modificades per Galaxy pels caràcters especials que representen.
2. Es canvia la ruta relativa al directori en què pyicoenrich escriu els resultats.
3. Es busca la ruta absoluta del fitxer amb longituds de cromosomes. Per defecte, s'utilitza l'última versió del genoma humà, hg19.

4. CONCLUSIONS

La bioinformàtica és un camp de la ciència que encara es troba en fase de creixement. Degut al reduït nombre d'entitats que hi participen, no s'han establert estàndards per la representació de dades. Això dificulta la creació d'eines d'anàlisi i l'intercanvi d'informació entre investigadors. A més, el desenvolupament de software és considerat de baixa prioritat: es dóna més importància a la publicació d'articles i als resultats dels anàlisis.

La plataforma Galaxy és una bona forma de mostrar el funcionament de paquets de software bioinformàtics com Pyicoteo, per la seva facilitat d'ús. També és una bona manera de compartir dades i resultats entre investigadors. D'altra banda, no hauria de ser considerada com a substitució de la línia de comandes, per diversos motius:

- Les dades s'emmagatzemen en un servidor remot. Si contenen informació confidencial, l'investigador pot tenir problemes legals.
- La disponibilitat del servei no està assegurada. La connexió a Internet pot patir interrupcions puntuals, tant per part del servidor com de l'usuari, que impedeixin l'ús de Galaxy.
- La línia de comandes dóna més control i flexibilitat a l'usuari. La interfície de Galaxy només permet especificar un conjunt de paràmetres definits pel desenvolupador, amb els valors possibles limitats.

Continuació del projecte

Com tots els projectes de software, el desenvolupament de Pyicoteo no finalitza quan les funcionalitats desitjades s'han implementat. El programa requereix feines de manteniment i de depuració d'errors.

D'altra banda, es podrien realitzar ampliacions a la *suite*, amb noves funcionalitats d'ajut a la recerca genòmica. A més, es podrien implementar els nous mètodes d'anàlisi que apareguessin en el futur.

5. BIBLIOGRAFIA

Articles

Althammer S., González-Vallinas J., Ballaré C., Beato M., Eyras E.

Pyicos: A versatile toolkit for the analysis of high-throughput sequencing data.

Bioinformatics. 2011 Dec 15;27(24):3333-40. doi: 10.1093/bioinformatics/btr570. Epub 2011 Oct 12.

Juan González-Vallinas, Sonja Althammer, Eduardo Eyras

Pyicos: A Flexible Tool Library for Analyzing Protein-Nucleotide Interactions with Mapped Reads from Deep Sequencing

Lecture Notes in Computer Science Volume 6620, 2012, pp 83-88

Pepke S, Wold B, Mortazavi A.

Computation for ChIP-seq and RNA-seq studies.

Nat Methods. 2009 Nov;6(11 Suppl):S22-32. doi: 10.1038/nmeth.1371.

Goecks, J, Nekrutenko, A, Taylor, J and The Galaxy Team.

Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences.

Genome Biol. 2010 Aug 25;11(8):R86.

Blankenberg D, Von Kuster G, Coraor N, Ananda G, Lazarus R, Mangan M, Nekrutenko A, Taylor J.

"Galaxy: a web-based genome analysis tool for experimentalists".

Current Protocols in Molecular Biology. 2010 Jan; Chapter 19:Unit 19.10.1-21.

Giardine B, Riemer C, Hardison RC, Burhans R, Elnitski L, Shah P, Zhang Y, Blankenberg D, Albert I, Taylor J, Miller W, Kent WJ, Nekrutenko A.

"Galaxy: a platform for interactive large-scale genome analysis."

Genome Research. 2005 Oct; 15(10):1451-5.

Pàgines web

The Galaxy Project
http://galaxyproject.org/

Galaxy Wiki
http://wiki.galaxyproject.org/

Python v2 Documentation
http://docs.python.org/2/

The Python Wiki
http://wiki.python.org/

Sphinx: Python Documentation Generator
http://sphinx-doc.org/

Jinja2 Template Engine
http://jinja.pocoo.org/docs/

W3Schools (HTML Reference)
http://www.w3schools.com/

Matplotlib API
http://matplotlib.org/api/

ENCODE: Encyclopedia of DNA Elements
http://genome.ucsc.edu/ENCODE/

UCSC Genome Browser
http://genome.ucsc.edu/

Wellcome Trust Sanger Institute – GFF Specifications Document
http://www.sanger.ac.uk/resources/software/gff/spec.html

ENSEMBL – GFF/GTF File Format
http://www.ensembl.org/info/website/upload/gff.html

ENSEMBL – The Bed File Format
http://www.ensembl.org/info/website/upload/bed.html

BEDTools – A flexible suite of utilities for comparing genomic features
http://code.google.com/p/bedtools/

BEDTools Documentation
http://bedtools.readthedocs.org/en/latest/

Vancouver Short Read Analysis Package
http://vancouvershortr.sourceforge.net/

Bioconductor
http://www.bioconductor.org/

Git – Version control system
http://git-scm.com/

Mercurial SCM
http://mercurial.selenic.com/

Annex A. Exemple de fitxer GFF

A continuació podem veure un fragment d'un fitxer GFF, extret de la versió 14 anotada per GENCODE del genoma humà. Aquest fragment conté un gen i diverses de les característiques que inclou.

El fitxer sencer es pot trobar a http://www.gencodegenes.org/releases/14.html

```
chr1 HAVANA gene 34554 36081 . - . gene_id "ENSG00000237613.2"; transcript_id "ENSG00000237613.2"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A"; level 2; havana_gene "OTTHUMG00000000960.1";
chr1 HAVANA transcript 34554 36081 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA exon 35721 36081 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA CDS 35721 35736 . - 0 gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA start_codon 35734 35736 . - 0 gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA exon 35277 35481 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA CDS 35277 35481 . - 2 gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA exon 34554 35174 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001";
```

```
level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA CDS 35141 35174 . - 1 gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA stop_codon 35138 35140 . - 0 gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA UTR 35737 36081 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA UTR 34554 35140 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000417324.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "protein_coding"; transcript_status "KNOWN"; transcript_name "FAM138A-001"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002842.1";
chr1 HAVANA transcript 35245 36073 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000461467.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "processed_transcript"; transcript_status "KNOWN"; transcript_name "FAM138A-002"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002843.1";
chr1 HAVANA exon 35721 36073 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000461467.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "processed_transcript"; transcript_status "KNOWN"; transcript_name "FAM138A-002"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002843.1";
chr1 HAVANA exon 35245 35481 . - . gene_id "ENSG00000237613.2"; transcript_id "ENST00000461467.1"; gene_type "protein_coding"; gene_status "KNOWN"; gene_name "FAM138A"; transcript_type "processed_transcript"; transcript_status "KNOWN"; transcript_name "FAM138A-002"; level 2; tag "basic"; havana_gene "OTTHUMG00000000960.1"; havana_transcript "OTTHUMT00000002843.1";
```

Annex B. Exemple de fitxer de documentació Sphinx

A continuació es mostra el manual d'ús de la comanda p*yicoregion*, en format reStructuredText.

```
.. _pyicoregiondocs:

Pyicoregion
===========

Pyicoregion is a Pyicoteo module for processing region files.

It uses GFF files as specified in
 http://www.sanger.ac.uk/resources/software/gff/spec.html


Pyicoregion arguments
---------------------

.. option:: --region-magic

    .. describe:: exons [position]

        Returns all the exons in the region file.

         If the optional argument ``[position]`` is specified (possible values: ``first``, ``last``), it will only return the first or last exon of every gene.


    .. describe:: introns [position]

        Returns all the introns in the region file.

         If the optional argument ``[position]`` is specified (possible values: ``first``, ``last``), it will only return the first or last intron of every gene.


          .. describe:: slide <window_size> <window_step> <region_type> [chromlen_file_path]

        Searches for intergenic and intragenic regions using sliding windows.
```

Mandatory arguments are ``<window_size>`` (the size of the sliding window), ``<window_step>`` (the distance between the start position of every consecutive window. It must be lower than or equal to the window size) and ``<region_type>`` (must be ``inter``, for intergenic, or ``intra``, for intragenic regions).

The optional argument ``[chromlen_file_path]`` is used to specify the path to the file containing the chromosome lengths (Pyicoteo's own chromlen files can be found in pyicoteolib/chromlen/). If it is not specified for intergenic regions, the results for the last regions of the chromosomes might be wrong.

Note: if the last segment of a region is shorter than the window size, the step distance is decreased by the difference (the window size stays the same).

Note: regions shorter than the window size are ignored.

.. describe:: tss <add_start> <add_end>

Returns the TSS for every transcript in the region file.

Due to a TSS being a single point, the arguments ``<add_start>`` and ``<add_end>`` specify the values added to the start and end of every TSS (taking into consideration the strand). For pyicoregion to work correctly, they must be non-negative integers. Also, if the strand is not specified, the regions will be treated as if they were positive.

Annex C. Integració Galaxy: Exemple XML

A continuació es mostra el contingut del fitxer XML per a la integració de la comanda Pyicoenrich amb Galaxy. S'ha escollit aquesta perquè és la més complexa i implementa totes les funcionalitats explicades.

```
<?xml version="1.0"?>
<tool id="pyicoteo_pyicoenrich" name="Pyicoenrich" version="2.0">
  <description>Pyicoteo enrichment operations.</description>
  <command interpreter="python">
          pyicoenrich -reads $experiment_a $experiment_b
                -f $experiment_format_cond.experiment_format
                $is_experiment_open
                $is_output_open
                --galaxy-workarounds

                --proximity $proximity
                $length_norm
                $number_reg_norm
                --binsize $bin_size
                $pseudocounts
                --zscore $z_threshold

          #if $replica_file
                --replica $replica_file
          #end if

          #if $tmm_values.tmm_norm
                $tmm_values.tmm_norm
                --a-trim $tmm_values.tmm_trim_a
                --b-trim $tmm_values.tmm_trim_b
          #end if

          #if $interesting_regions
                --interesting-regions $interesting_regions
          #end if

          #if $experiment_format_cond.experiment_format == "custom"
                --f-custom-in $experiment_format_cond.f_custom_in
                --custom-in-sep $experiment_format_cond.custom_in_sep
```

```
          #end if

          #if $region_file_cond.region_file_select == "region_file_option"
                --region-file $region_file_cond.region_file
                --region-format $region_file_cond.region_file_format
                $region_file_cond.region_file_open
          #elif $region_file_cond.region_file_select == "gff_file_option"
                --gff-file $region_file_cond.gff_file
                --region-magic $region_file_cond.region_magic_cond.region_magic
                #if $region_file_cond.region_magic_cond.region_magic == "slide"
                                  $region_file_cond.region_magic_cond.window_size
$region_file_cond.region_magic_cond.window_step
$region_file_cond.region_magic_cond.region_slide_cond.slide_type
                                                                                #if
$region_file_cond.region_magic_cond.region_slide_cond.slide_type == "inter" and
$region_file_cond.region_magic_cond.region_slide_cond.chromlen_file
                        $region_file_cond.region_magic_cond.region_slide_cond.ch
romlen_file
                    #end if
                #elif $region_file_cond.region_magic_cond.region_magic == "tss"
                                 $region_file_cond.region_magic_cond.tss_add_start
$region_file_cond.region_magic_cond.tss_add_end
                #end if
          #end if

          #if $html_output
              -output $html_file.files_path/enrich_output.bed
              --html-output $html_file
          #else
              -output $output
          #end if


  </command>
  <inputs>


      <param name="html_output" type="boolean" label="HTML output" help="Output
the results as HTML (with plot). WARNING: if enabled, the resulting output can
not be used as input to another tool (the BED file must be downloaded from 'View
data' and uploaded again)" />


     <!--<param format="tabular" name="experiment" type="data" label="Experiment
file" />-->
    <param name="experiment_a" type="data" label="Experiment A file" />
    <param name="experiment_b" type="data" label="Experiment B file" />


    <conditional name="experiment_format_cond">
         <param name="experiment_format" type="select" label="Experiment files
format" format="text">
```

```
        <option value="bed" selected="true">Bed</option>
        <option value="sam">Sam</option>
        <option value="eland">Eland</option>
        <option value="bed_wig">Wiggle (Bed)</option>
        <option value="bed_pk">Bed pk</option>
        <option value="custom">Custom</option>
      </param>

      <when value="custom">
        <param name="f_custom_in" type="text" size="10" label="Custom experiment format fields" />
         <param name="custom_in_sep" type="text" label="Custom experiment format separator" />
      </when>
   </conditional>

     <param name="is_experiment_open" type="boolean" label="Are the experiments half-open?" truevalue="--open-experiment" falsevalue="" />
    <param name="is_output_open" type="boolean" label="Is the output half-open?" truevalue="--open-output" falsevalue="" />


    <!-- Region files -->
    <conditional name="region_file_cond">
      <param name="region_file_select" type="select" label="Region file type">
        <option value="autogenerate">Auto-generate</option>
        <option value="region_file_option">Region file</option>
        <option value="gff_file_option">GFF file</option>
      </param>

      <when value="region_file_option">
        <param name="region_file" type="data" label="Region file" />
             <param name="region_file_format" type="select" label="Region file format">
          <option value="bed">Bed</option>
          <option value="sam">Sam</option>
          <option value="eland">Eland</option>
          <option value="bed_wig" selected="true">Wiggle (Bed)</option>
          <option value="variable_wig">Wiggle (Variable)</option>
          <option value="bed_pk">Bed pk</option>
        </param>
         <param name="region_file_open" type="boolean" label="Is the region file half-open?" truevalue="--open-region" falsevalue="" />
      </when>

      <when value="gff_file_option">
```

```
          <param name="gff_file" type="data" label="GFF file with regions" />
          <conditional name="region_magic_cond">
            <param name="region_magic" type="select" label="Region magic">
              <option value="exons">Exons</option>
              <option value="introns">Introns</option>
              <option value="slide">Sliding window</option>
              <option value="tss">TSS</option>
            </param>

            <when value="slide">
                  <param name="window_size" type="integer" label="Window size"
size="6" value="" />
                  <param name="window_step" type="integer" label="Window step"
size="6" value="" />
              <conditional name="region_slide_cond">
                   <param name="slide_type" type="select" label="Sliding window
type">
                <option value="inter">Intergenic</option>
                <option value="intra">Intragenic</option>
              </param>
              <when value="inter">
                 <param name="chromlen_file" type="data" label="Chromlen file"
optional="true"  help="If  not  specified,  hg19  chromlen  will  be  used  by
default." />
              </when>
            </conditional>
          </when>

          <when value="tss">
              <param name="tss_add_start" type="integer" label="Add to start"
size="6" value="0" />
                  <param name="tss_add_end" type="integer" label="Add to end"
size="6" value="0" />
          </when>
        </conditional>
      </when>
    </conditional>


     <param name="replica_file" type="data" label="Experiment A replica file"
optional="true" />
    <param name="proximity" type="integer" label="Proximity" size="6" value="50"
/>
    <param name="length_norm" type="boolean" label="Region length normalization"
truevalue="--len-norm" falsevalue="" />
       <param  name="number_reg_norm"  type="boolean"  label="Number  of  regions
normalization" truevalue="--n-norm" falsevalue="" />
```

```
    <conditional name="tmm_values">
            <param name="tmm_norm" type="boolean" label="TMM normalization" truevalue="--tmm-norm" falsevalue="" />
      <when value="--tmm-norm">
          <param name="tmm_trim_a" type="float" label="Trimming of the A values (for the TMM normalization)" size="6" value="0.05" />
          <param name="tmm_trim_b" type="float" label="Trimming of the B values (for the TMM normalization)" size="6" value="0.25" />
      </when>
    </conditional>


    <param name="bin_size" type="float" label="Bin size" size="6" value="0.3" />
    <param name="pseudocounts" type="boolean" label="Use pseudocounts to include in the analysis regions with 0 reads in one of the samples" truevalue="--pseudocount" falsevalue="" />
     <param name="z_threshold" type="float" label="z-score threshold for calling significant regions" value="2.0" />


      <param name="interesting_regions" type="data" label="Interesting regions file" optional="true" />


  </inputs>
  <outputs>
    <data format="tabular" name="output">
      <filter>not html_output</filter>
    </data>
    <data format="html" name="html_file">
      <filter>html_output</filter>
    </data>
  </outputs>
  <stdio>
    <exit_code range="1:" level="error" />
  </stdio>
  <help>
        More information at: http://regulatorygenomics.upf.edu/pyicos
  </help>
</tool>
```

Annex D. Integració Galaxy: Plantilla per Pyicoenrich

Aquesta és la plantilla que s'utilitza per mostrar els resultats de pyicoenrich conjuntament. Es pot notar l'ús de JavaScript per mostrar i amagar el text, que al final no s'utilitza a causa del filtratge fet per Galaxy.

```
<html>
  <head>
    <title>{{ page_title }}</title>
  </head>

  <body>
    <!--<script language="javascript">
        function toggleHidden(text) {
            var div = document.getElementById(text);
            var a = document.getElementById(text + "_a");
            if (div.style.display == "none") {
                div.style.display = "block";
                a.innerHTML = "Hide text";
            } else {
                div.style.display = "none";
                a.innerHTML = "Show text";
            }
        }
    </script> -->
    <div id="img_div">
      <h3>Enrichment plot</h3>
      <img src="{{ plot_path }}" />
    </div>
      <!--<a      href="javascript:void(0);"      onclick      =      "javascript:
toggleHidden('output_contents')" id="output_contents_a">Show text</a>-->
    <div id="output_contents" style="display: none;">
      <h3><a href="{{ bed_path }}">Download Results (BED)</a></h3>
      <pre>{{ results_output }}</pre>
    </div>
  </body>
</html>
```

Annex E. Integració Galaxy: tool_conf.xml

A continuació es presenta el contingut del fitxer tool_conf.xml, que descriu les rutes als fitxers de les eines. Com es pot veure, consta de dues seccions:

- Get Data: Ja inclosa per defecte a Galaxy, implementa eines per a carregar dades a l'espai de treball. Entre elles destaca data_source/upload.xml, per carregar fitxers des del navegador. Les altres no s'han eliminat perquè es considera que poden ser d'utilitat als usuaris.
- Pyicoteo: Secció implementada en aquest projecte. Conté les eines de Pyicoteo integrades a Galaxy.

```
<?xml version="1.0"?>
<toolbox>
  <section name="Get Data" id="getext">
    <tool file="data_source/upload.xml"/>
    <tool file="data_source/ucsc_tablebrowser.xml" />
    <tool file="data_source/ucsc_tablebrowser_test.xml" />
    <tool file="data_source/ucsc_tablebrowser_archaea.xml" />
    <tool file="data_source/bx_browser.xml" />
    <tool file="data_source/ebi_sra.xml"/>
    <tool file="data_source/microbial_import.xml" />
    <tool file="data_source/biomart.xml" />
    <tool file="data_source/biomart_test.xml" />
    <tool file="data_source/cbi_rice_mart.xml" />
    <tool file="data_source/gramene_mart.xml" />
    <tool file="data_source/fly_modencode.xml" />
    <tool file="data_source/flymine.xml" />
    <tool file="data_source/flymine_test.xml" />
    <tool file="data_source/modmine.xml" />
    <tool file="data_source/mousemine.xml" />
    <tool file="data_source/ratmine.xml" />
    <tool file="data_source/yeastmine.xml" />
    <tool file="data_source/metabolicmine.xml" />
    <tool file="data_source/worm_modencode.xml" />
    <tool file="data_source/wormbase.xml" />
    <tool file="data_source/wormbase_test.xml" />
    <tool file="data_source/eupathdb.xml" />
    <tool file="data_source/encode_db.xml" />
```

```
    <tool file="data_source/epigraph_import.xml" />
    <tool file="data_source/epigraph_import_test.xml" />
    <tool file="data_source/hbvar.xml" />
    <tool file="genomespace/genomespace_file_browser_prod.xml" />
    <tool file="genomespace/genomespace_importer.xml" />
    <tool file="validation/fix_errors.xml" />
  </section>

  <section name="NGS:Pyicoteo" id="Pyicoteo">
    <tool file="pyicoteo/pyicoteo_convert.xml" />
    <tool file="pyicoteo/pyicoteo_remregions.xml" />
    <tool file="pyicoteo/pyicoteo_remduplicates.xml" />
    <tool file="pyicoteo/pyicoteo_strcorr.xml" />
    <tool file="pyicoteo/pyicoteo_extend.xml" />
    <tool file="pyicoteo/pyicoteo_subtract.xml" />
    <tool file="pyicoteo/pyicoteo_split.xml" />
    <tool file="pyicoteo/pyicoteo_discard.xml" />
    <tool file="pyicoteo/pyicoteo_push.xml" />
    <tool file="pyicoteo/pyicoteo_filter.xml" />
    <tool file="pyicoteo/pyicoteo_trim.xml" />
    <tool file="pyicoteo/pyicoteo_pyicoller.xml" />
    <tool file="pyicoteo/pyicoteo_pyicoenrich.xml" />
    <tool file="pyicoteo/pyicoteo_pyicoclip.xml" />
    <tool file="pyicoteo/pyicoteo_pyicoregion.xml" />
  </section>
</toolbox>
```