Bon dia Alex, gràcies per les preguntes i els comentaris que fas.

Sí, penso també que integrar les dades d’exemple ajudaria l’usuari a situar-se més ràpidament a l’aplicació. De fet aquesta ha sigut també la meva idea, però desprès vaig deixar la integració de dades per al final, perquè encara no sabia on s’instal·laria l’aplicació i quina seria l’arquitectura de les carpetes. En el marc del paquet creat, si es tingués més temps i accés al servidor, sí que es podria fer especificant les rutes relatives que apuntarien al lloc on l’arxiu de les dades se situa (“./Data”, o “../Data”).

1. Has escollit el clusterprofiler com a base de les teves anàlisis però segurament has vist que hi ha moltes altres eines que fan coses semblants. Que penses d’aquesta diversitat?

Sí, hi ha molts més paquets que cobreixen gairebé tots els mètodes diferents d’anàlisi de les rutes. Com ho he descrit en la memòria hi ha tres línies generals d’anàlisi de les rutes: Overrepresentation analysis, Functional Class Scorring i mètodes PT (pathway topology). Mentre el mètode ORA és menys divers, el mètode GSEA sí que varia bastant. Tots els mètodes ORA es basen en la distribució hipergeomètrica. L’única cosa que es varia és el conjunt de gens de referència. Aquest conjunt de referència es pot basar en les bases de dades d’interès com ara GO, KEGG o Reactome implementades en l’aplicació. Els mètodes FCS tenen més espai per variar: aquí es poden variar els mètodes per calcular la puntuació, però també els mètodes per calcular la distribució teòrica per contrastar la hipòtesi que la ruta és diferencialment sobre o sota expressada. És aquí on la comunitat bioinformàtica ha treballat per oferir més diversitat. Un dels paquets que al principi vaig pensar d’implementar era GAGE. Aquí l’usuari pot adaptar millor l’anàlisi al seu cas particular (la mida de la mostra, mètodes paramètrics i no paramètrics etc.). En el context del TFM però aquesta diversitat va resultar ser un factor de distracció. Havia de prendre una decisió ràpida a l’hora de seleccionar els paquets que utilitzaré per a l’aplicació. Així m’assegurava que podria acabar l’aplicació a temps, si més no amb els mètodes bàsics. Crec també que la selecció del paquet clusterProfiler, ReactomePA i pathview era bona, perquè d’aquesta manera es donava la possibilitat de dur a terme l’anàlisi estàndard i integrada. Amb integrada vull dir l’anàlisi numèrica i l’anàlisi gràfica.

2. Què et va dur a adonar-te que calia modificar el paquet pathview? Com vas saber que calia fer? Sembla un tema bastant específic.

En Shiny el mètode per generar el gràfic és: primer definir un arxiu temporal on es guardarà el gràfic, generar el gràfic i eliminar posteriorment el gràfic del directori temporal. La funció per definir l’arxiu temporal és temfile() que genera una ruta més el nom temporal de l’arxiu. També tenim la funció tempdir() que dona només el directori temporal. D’altra banda la funció pathview() genera per defecte els tres gràfics amb un nom predeterminat en la ruta on es localitza l’arxiu app.R. Des de la funció renderImage podem eliminar el gràfic visualitzat amb deleteFile=TRUE, però no els dos arxius addicionals generats per pathview. Clar, això no era òptim perquè en el directori del codi s’acumularien els arxius innecessaris. El problema era per tant que la funció original de pathview() no permetia especificar el directori on els gràfics es guarden per poder especificar la ruta temporal. Per aquest motiu he decidit modificar la funció pathview::pathview. Utilitzant aquesta funció modificada, els tres gràfics es guarden en el directori temporal d’on s’eliminen automàticament. Segurament hi ha un altre “work around” més net, però en el moment del desenvolupament calia prendre decisions ràpides...

3. Saps si el gràfic “Enrichment Map” es basa en el programa del mateix nom (en Java) de Gary Bader?

No en tinc constància. Però he buscat el mètode utilitzat per generar els gràfics “enrichment maps” a Bader Labs i em sembla que en termes generals els enrichment maps de clusterPrifeler utilitzats en l’aplicació transmeten la mateixa idea:

Els conjunts de gens estan representats com a nodes els radis dels quals són proporcionalment relacionats amb el nombre de gens que formen part d’aquests conjunts. Els cantells indiquen els nodes que tenen gens compartits, on el seu gruix depèn del nombre de gens compartits. A més a més, es pot utilitzar el color dels nodes per representar una altra dimensió com ara el nivell de significació expressat pel valor de P.

4. Si haguessis de suggerir un pipeline per a l’anàlisi d’una llista de gens que proporciones una “Anàlisi de significació biològica completa” quin seria el teu consell?

El pipeline seria el següent:

1. Elegir com a espècie homo sapiens.
2. Pensar quina base de dades volem escollir per anotar les rutes.
3. Fer l’anàlisi ORA.
4. Visualitzar l’anàlisi ORA.
5. Especialment observar el grau de redundància mitjançant Enrichment-Plots i Category-Gen network.
6. Fer l’anàlisi GSEA.
7. Visualitzar l’anàlisi GSEA amb enrichment score gràfics.

5. En línia amb l’anterior quan recomenaries fer servir ORA o GSEA?

Jo recomanaria fer servir els dos mètodes. Primer, com s’indica en pipeline, jo faria l’anàlisi ORA. Aquesta anàlisi permet observar grans efectes, si n’hi ha. Aquesta anàlisi és intuïtiva i fàcil d’interpretar. Tot i així té el seu punt feble: Típicament ORA utilitza només els gens més significatius i descarta els altres. Els processos biològics però poden acumular efectes petits per produir un efecte conjunt clínicament significatiu. Són aquests efectes els que per a l’anàlisi ORA poden passar per alt, perquè ni sols tindran una oportunitat d’entrar al set dels gens diferencialment expressats. Per tant jo repetiria l’anàlisi GSEA primer per validar el resultat de l’ORA i completar-lo amb unes observacions addicionals.

6. En aquests moments l’aplicació funciona des de un servidor nostre, però hi ha algun motiu per a qeu no l’hgis mirat de posar al servidor public de Shiny? (Shiny.apps.io?)

De fet aquesta ha sigut sempre la meva opció preferida. I ho vaig intentar. Malauradament no era possible d’implementar l’aplicació en Shiny.io. El problema va sorgir bàsicament amb la instal·lació del paquet reactomdb. A l’hora de publicar l’aplicació al cap d’uns quants minuts la pujada resultava en un error:

Error: Unhandled Exception: Child Task ..... error: Unhandled Exception: HTTP 599:Timeout

Es veu que hi ha un timeout per construir (built) un paquet en Shiny.io. El paquet reactomdb mesura aprox. 2.5Gb en estat deompressat i necessita més de 15 min per ser construït, cosa que provoca un error.

Aquí es pot veure les discussions d’aquest problema:

<https://groups.google.com/forum/#!msg/shinyapps-users/QRydNubnMQA/kJBxwUIJCgAJ)>

<https://community.rstudio.com/t/http-599-timeout-during-request/15923>

<https://community.rstudio.com/t/error-599-timeout/9275/4>

7. Suposem que aquesta aplicació té molt d’èxit i comença a fer-la servir molta gent. Què creus que hauries de tenir en compte per evitar que “mori d’èxit”, es a dir per escalar-la adequadament?

Tant de bo que tingui tant d’èxit. De totes manera reconec que hi ha un problema d’escalar les aplicacions Shiny. El problema de R és que R és “single-threaded”. Això vol dir que els usuaris que es connecten a un procés de R es bloquejaran l’un a l’altre. Es pot millorar el performance de l’aplicació amb Shiny Server Pro. Aquí es pot tenir processos múltiples per a l’aplicació. Això significa que els usuaris múltiples poden ser distribuïts entre processos separats. No hi ha cap limitació en nombre de processos i es pot utilitzat tots els recursos de la màquina. Hi ha només una qüestió, Shiny pro val 9.995$/any. Hi ha però una alternativa que val la pena provar: ShinyProxy de OpenAnalytics. Aquí s’utilitza la tecnologia de contenidors:

<https://www.shinyproxy.io/shinyproxy-containers/>