Búsquedas programáticas en la base de datos ENA

07/02/2023

Introducción

La *European Nucleotide Archive* (ENA) es la contraparte europea del *International Nucleotide Sequence Database Collaboration* (INSDC). Además de poder consultar y descargar datos manualmente desde su página web, ENA nos ofrece diferentes herramientas para realizar consultas (incluso análisis) de forma **programática**, es decir, desde algún lenguaje de programación.

Los sistemas de envío de tareas a los servidores del ENA mediante un *script* se llaman SOAP y REST. Están documentados en el enlace siguiente: https://www.ebi.ac.uk/seqdb/confluence/display/JDSAT. Pero quedan fuera de los objetivos de este curso.

En esta práctica nos limitaremos a utilizar lo que llamamos una "interfaz de programación de aplicaciones" o API, por sus siglas en inglés: un conjunto de reglas que permiten la comunicación entre *programas*. En este caso, la comunicación se establece entre nuestro *script* (este mismo documento) y los servidores de ENA. La misma API se podría usar para desarrollar una aplicación web que sirviera de intermediaria para acceder a los datos de ENA.

La API en sí es solo la estructura que debe tener una dirección URL para que nos devuelva la información que queremos. En el enlace siguiente encontrarás documentación sobre cómo realizar búsquedas avanzadas en ENA programáticamente usando esta API:

https://ena-docs.readthedocs.io/en/latest/retrieval/programmatic-access/advanced-search.html

Podéis también descargar la documentación actualizada de la ENA Portal API, pinchando aquí. A través de esta API podemos obtener fundamentalmente *metadatos* sobre las secuencias nucleotídicas almacenadas en ENA. Es decir, tablas que describen los diferentes registros encontrados. Por ejemplo, copiando la cadena de caracteres siguiente y pegándola en la barra del navegador, accederéis a una tabla que describe los registros de tipo *read run* (lecturas cortas) procedentes del estómago de vacas del Reino Unido disponibles en ENA:

```
https://www.ebi.ac.uk/ena/portal/api/search?result=read_run&query=country= "United Kingdom"
AND host_tax_id=9913 AND host_body_site="rumen"
```

Podéis consultar la tabla también pinchando aquí.

Aunque esto es muy poca información, y no nos da acceso a los archivos o a las secuencias propiamente. En esta práctica vamos a ver cómo construir las URLs para extraer de ENA los metadatos que necesitemos. Para extraer directamente las secuencias, por ejemplo de genomas ensamblados y en formato fasta, ENA nos ofrece otro portal, con una API muy parecida: el ENA Browser Portal, cuya documentación podéis descargar pinchando aquí. Sin embargo, vamos a centrarnos en la API del portal de ENA.

La API del portal de ENA

Cualquier consulta dirigida al portal de ENA tomará la forma de una URL con la estructura siguiente:

- La dirección del portal: `https://www.ebi.ac.uk/ena/portal/api`.
- El **endpoint** `/search`, que especifica que realizamos una consulta o búsqueda. Existen otros *endpoints*, que podemos obviar de momento.
- Los parámetros de la búsqueda, que incluyen cinco campos separados por "&":

- `/result=<tipo de datos>`, donde "<tipo de datos" puede ser cualquiera de los *results* disponibles: *read_run*, *read_experiment*, *sample*, *study*, *sequence_release*, *sequence_update*, *wgs_set*, *tsa_set*, *assembly*, *coding_release*, *coding_update*, *noncoding_releaes*, *noncoding_update*, y *taxonomy*.
- Opcionalmente, `query=<filtros>`, donde "<filtros" debe sustituirse por una o más condiciones que deben cumplir los registros para ser presentados. Las diferentes condiciones o filtros irán separadas por "AND", "OR" o "NOT". Existen reglas sobre cómo especificar cada posible filtro para cada posible tipo de resultado. Estas reglas están especificadas en la documentación del portal que se puede descargar pinchando aquí.
- Opcionalmente, `fields=<campos>`, donde "" se debe sustituir por la lista, separada por comas, de los campos de información que deseamos extraer de cada registro. Es decir, las columnas que deseamos ver en la tabla de resultados. Cada tipo de `result` admite un conjunto de campos disponibles, que también están detallados en la documentación del portal.
- Opcionalmente, `limit=<límite>` donde "" representa el número máximo de registros (líneas) que queremos extraer en la tabla de resultados. Por defecto, el límite es 100000. Para no limitar la búsqueda, hay que especificar `limit=0`.
- Opcionalmente, `format=<formato>`, donde "" representa uno de dos valores posibles: `tsv` (*tab separated values*, usado por defecto) o `json`.

Ejemplos

```
https://www.ebi.ac.uk/ena/portal/api/search?result=sequence&query=(specimen_voucher="ZMB:Moll:*"
OR bio_material="ZMB:Moll:*")&fields=specimen_voucher,bio_material,scientific_name
```

```
https://www.ebi.ac.uk/ena/portal/api/search?result=analysis&query=country="United Kingdom"
AND host_tax_id=9913 AND host_body_site="rumen" AND analysis_type="SEQUENCE_ASSEMBLY" AND
assembly_type="primary metagenome"&fields=submitted_ftp
```

```
https://www.ebi.ac.uk/ena/portal/api/search?result=assembly&query=tax_tree(6157) AND
genome_representation="full"&fields=version,tax_id,scientific_name,last_updated,base_count&limit=0
```

Desde una sesión de R

Una opción es utilizar el paquete `curl` en R para enviar la consulta en forma de URL y descargar el resultado a un archivo, que posteriormente puede ser leído y analizado con R. Si no està instalado, habría que instalarlo mediante `install.packages('curl')`. Este paquete pone a nuestra disposición la función `curl_download()`, para descargar el contenido de una URL en un archivo.

```
library('curl')
```

```
## Using libcurl 7.74.0 with OpenSSL/1.1.1n
```

¿Qué problemas encuentras con cada uno de estos bloques?

Bloque 1

```
curl_download('https://www.ebi.ac.uk/ena/portal/api/search?result=assembly&query=tax_tree(6157) AND gen
              destfile = 'z1.txt')
```

Bloque 2

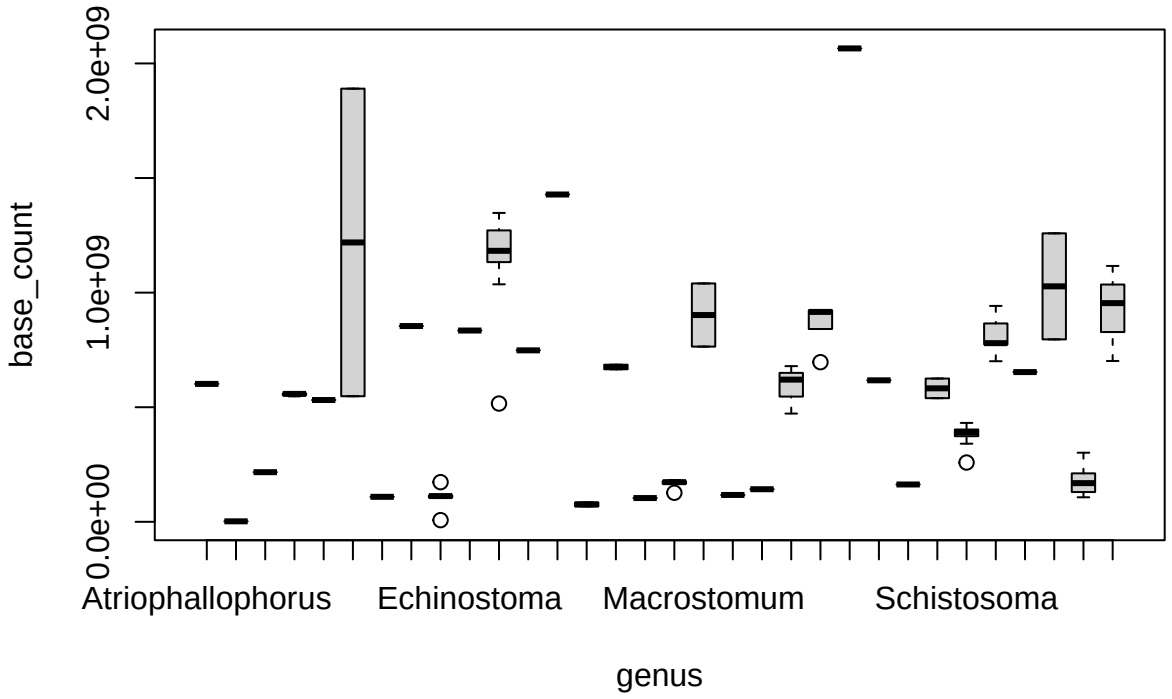
```
curl_download(
  URLencode('https://www.ebi.ac.uk/ena/portal/api/search?result=assembly&query=tax_tree(6157) AND genom
  destfile = 'z1.txt')
```

Bloque 3

```
portal <- 'https://www.ebi.ac.uk/ena/portal/api/search?'
result <- 'result=assembly&'
query  <- 'query=tax_tree(6157) AND genome_representation="full"&'
fields <- 'fields=version,tax_id,scientific_name,last_updated,base_count&'
limit  <- 'limit=0'
url <- paste0(portal, result, query, fields, limit)
curl_download(URLencode(url), destfile = 'z2.txt')
```

Trabajar con los resultados

```
platelmintos <- read.table('z2.txt',
                           header = TRUE,
                           colClasses = c('character', 'numeric', 'factor', 'factor', 'Date', 'numeric'
                           sep = '\t',
                           na.strings = '')
platelmintos$genus <- sapply(strsplit(as.character(platelmintos$scientific_name), ' '), '[', 1)
boxplot(base_count ~ genus, data = platelmintos)
```



Información sobre la sesión

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 11 (bullseye)
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
```

```
##
## locale:
##  [1] LC_CTYPE=ca_ES.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=ca_ES.UTF-8        LC_COLLATE=ca_ES.UTF-8
##  [5] LC_MONETARY=ca_ES.UTF-8    LC_MESSAGES=ca_ES.UTF-8
##  [7] LC_PAPER=ca_ES.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=ca_ES.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] curl_5.0.0
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.31   magrittr_2.0.3  evaluate_0.20   highr_0.9
##  [5] rlang_1.0.6     stringi_1.7.6   cli_3.6.0       rstudioapi_0.14
##  [9] rmarkdown_2.14  tools_4.2.2     stringr_1.4.0   xfun_0.30
## [13] yaml_2.3.5      fastmap_1.1.0   compiler_4.2.2  htmltools_0.5.4
## [17] knitr_1.38
```