## Supplementary File 1

## 10.1 R code for the BridgeDb R example

For more details check: <http://bioconductor.org/packages/release/bioc/html/BridgeDbR.html>

library(BridgeDbR)

mapper <- loadDatabase("SNP\_r91\_PolyPhen.bridge")

genes <- map(mapper, “Sn”, “rs5927022”, “En”)

# integration with other R packages, e.g. mygene

gene <- getGene(genes[2], fields="all")

gene\_name <- gene[[1]]$name

gene\_length <- gene[[1]]$genomic\_pos$end -

gene[[1]]$genomic\_pos$start

exonCount <- length(gene[[1]]$exons)

# it can also list SNPs for genes

length(map(mapper, "En", "ENSG00000198947", "Sn"))

## 10.2 Instructions to set up the BridgeDb IMS Docker service

This tutorial explains how to install and run a BridgeDb Identifier Mapping Service (IMS) with gene-to-variant and variant-to-gene functionality.

### Step 1. Downloading the gene-variant mappings

Download the linksets data you want to load into the IMS from<http://bridgedb.org/data/gene_database/linkset/>.

### Step 2. Install Docker

A prerequisite is a Docker installation. You can download Docker from various places and many GNU/Linux distributions ship a version as part of their distribution. Other options include:

* Docker Community Edition:<https://www.docker.com/get-docker>
* Docker Toolbox for older Mac and Windows versions:<https://docs.docker.com/toolbox/overview/>
* Docker for Windows:<https://docs.docker.com/docker-for-windows/>

### Step 3. Downloading the BridgeDb IMS Docker

Pull the IMS docker image:

docker pull openphacts/identitymappingservice

### Step 4. Set up the MySQL service

Create the mysql docker image to store the linksets:

docker run --name mysql-for-ims -e MYSQL\_ALLOW\_EMPTY\_PASSWORD=yes \  
-e MYSQL\_DATABASE=ims -e MYSQL\_USER=ims -e MYSQL\_PASSWORD=ims \  
-d mysql

### Step 5. Loading the gene-gene and gene-variant link sets

The next command line example assumes that the file load.xml is present the /home/johndoe/data5 directory.

Unix:

docker run --link mysql-for-ims:mysql -v /home/johndoe/data5:/staging \  
 openphacts/identitymappingservice loader file:///staging/load.xml

MINGW64 / Windows:

docker run --link mysql-for-ims:mysql -v /c/Users/johndoe/data5:/staging openphacts/identitymappingservice loader file:///staging/load.xml

An example of the load.xml file (more info):

<?xml version="1.0"?>  
 <loadSteps>  
 <clearAll/>  
 <void>file://staging/Ensembl\_SNP\_dataset.void.ttl</void>  
 <linkset>file:///staging/PolyPhen\_91.ttl</linkset>  
 </loadSteps>

### Step 6. Run the IMS docker image

docker run --name ims --link mysql-for-ims:mysql -p 8081:8080 -d openphacts/identitymappingservice

You can confirm the instance is running with:

docker ps

Both these commands will give you a long identifier which you should remember for later.

### Step 7. Check the QueryExpander expose at the local port 8081

Open the following webpage in a browser:

* Unix:<http://localhost:8081/QueryExpander/>
* Windows:<http://192.168.99.100:8081/QueryExpander/>

It should look like this:



### Step 8. Shutting down the Docker image

Shutting down the docker is done with the command docker stop and the identifier you got with the docker run or docker ps commands, for example:

docker stop b5e9715818f6

or

docker stop ims

Where b5e9715818f6 or ims happens to be the ID/name for the running image for the author at the time of writing.

You can delete the images with:

docker container rm b5e9715818f6 d989788c0b03

Where the two IDs are the one for the MySQL-IMS docker and the IMS docker containers.

## 10.3 R code for the IMS

library(jsonlite)

library("curl")

mapUriService = "http://localhost:8081/QueryExpander/mapUri"

# gene to variants

gene = "http://rdf.ebi.ac.uk/resource/ensembl/ENSG00000198947"

targetUriPattern = "http://identifiers.org/dbsnp"

h <- new\_handle()

handle\_setopt(h, customrequest = "POST")

handle\_setform(h, Uri=gene, format = "application/json", targetUriPattern = targetUriPattern)

r <- curl\_fetch\_memory(mapUriService, h)

jsonReply = rawToChar(r$content)

data = fromJSON(jsonReply)

variantCount = length(data$Mapping$targetUri)

head(data$Mapping$targetUri)

# variants to gene

gene = "http://identifiers.org/dbsnp/rs769658853"

targetUriPattern = "http://identifiers.org/ensembl"

h <- new\_handle()

handle\_setopt(h, customrequest = "POST")

handle\_setform(h, Uri=gene, format = "application/json", targetUriPattern = targetUriPattern)

r <- curl\_fetch\_memory(mapUriService, h)

jsonReply = rawToChar(r$content)

data = fromJSON(jsonReply)

gene = data$Mapping$targetUri