dbGaP2x

October 19, 2018

1 Using dbGaP2x, R package to explore and sort phenotypics data from dbGap

1.1 Introduction

1.1.1 Load the package

```
In [1]: #devtools::install_github("gversmee/dbGaP2x", force = TRUE)
        library(dbGaP2x)
```

1.1.2 Get the list of the function for this new package

```
In [2]: lsf.str("package:dbGaP2x")
```

```
browse.dbgap : function (phs, jupyter = FALSE)
browse.study : function (phs, jupyter = FALSE)
consent.groups : function (phs)
datatables.dict : function (phs)
dbgap.data_dict : function (xml, dest)
dbgap.decrypt : function (files, key = FALSE)
dbgap.download : function (krt, key = FALSE)
is.parent : function (phs)
n.pop : function (phs, consentgroups = TRUE, gender = TRUE)
n.tables : function (phs)
n.variables : function (phs)
parent.study : function (phs)
phs.version : function (phs)
search.dbgap : function (term, jupyter = FALSE)
study.name : function (phs)
sub.study : function (phs)
variables.dict : function (phs)
```

1.2 Search for dbGap studies

1.2.1 Let's try to explore the "Jackson Heart Study" cohort that exists on dbGap.

The dbGap search engine can be tricky, that's why we created the function "browse.dbgap", who helps you find the studies related to the term that you search on your web browser.

Note that if you run this function in a jupyterhub environment, it will return a url since jupyterhub doesn't have access to your local browser.

```
In [3]: search.dbgap("Jackson", jupyter = TRUE)
```

'https://www.ncbi.nlm.nih.gov/gap/?term=Jackson%5BStudy+Name%5D'

dbGap returns the list of the studies related to your term. As you see, there are 6 studies associated with the "Jackson Heart Study" (JHS). One of these study is the main one aka the "parent study", whereas the other ones are substudies. In this case, phs000286.v5.p1 is the parent study. Firslty, we can use the phs.version() function in order to be sure that this is the latest version of the study. We can abbreviate the phs name by giving just the digit, or we can use the full dbGap id.

```
In [4]: phs.version("286")
```

'phs000286.v5.p1'

The is.parent() function is usefull to test if a study is a parent study or a substudy

```
In [5]: is.parent("000286") # JHS main cohort
        is.parent("phs499") # substudy "CARe" for JHS
```

TRUE
FALSE

If you don't know the parent study of a substudy, try parent.study()

```
In [6]: parent.study("phs000499")
```

1. 'phs000286.v5.p1' 2. 'Jackson Heart Study (JHS) Cohort'

On the other side, use sub.study() to get the name and IDs of the substudies from a parent one

```
In [7]: sub.study("286")
```

phs	name
phs000499.v3.p1	NHLBI Jackson Heart Study Candidate Gene Association Resource (CARe)
phs000498.v3.p1	Jackson Heart Study Allelic Spectrum Project
phs000402.v3.p1	NHLBI GO-ESP: Heart Cohorts Exome Sequencing Project (JHS)
phs001098.v1.p1	T2D-GENES Multi-Ethnic Exome Sequencing Study: Jackson Heart Study

If you want to get the name of a study from its dbGap id, use study.name()

```
In [8]: study.name("286")
```

'Jackson Heart Study (JHS) Cohort'

Finally, you can watch your study on dbGap with browse.dbgap().

If a website exists for this study, you can browse it using browse.study()

```
In [9]: browse.dbgap("286", jupyter = TRUE)
        browse.study("286", jupyter = TRUE)
```

'https://www.ncbi.nlm.nih.gov/projects/gap/cgi-bin/study.cgi?study_id=phs000286.v5.p1'
'https://www.jacksonheartstudy.org'

1.3 Explore the characteristics of your study

For each dbGap study, there can be multiple consent groups that will have there specificities. Use consent.groups to know the number and the name of the consent groups in the study that you are exploring. Let's keep focusing on JHS.

```
In [10]: JHS <- "phs000286"
         consent.groups(JHS)
```

	shortName	longName
0	NRUP	Subjects did not participate in the study, did not complete a consent document and
1	HMB-IRB-NPU	Health/Medical/Biomedical (IRB, NPU)
2	DS-FDO-IRB-NPU	Disease-Specific (Focused Disease Only, IRB, NPU)
3	HMB-IRB	Health/Medical/Biomedical (IRB)
4	DS-FDO-IRB	Disease-Specific (Focused Disease Only, IRB)

Use n.pop() to know the number of patient included in each groups

```
In [11]: n.pop(JHS)
         n.pop(JHS, consentgroups = FALSE)
```

consent_group	male	female	total
HMB-IRB	1860	2504	4549
HMB-IRB-NPU	264	505	802
DS-FDO-IRB-NPU	63	107	180
HMB-IRB	784	1232	2131
DS-FDO-IRB	173	289	489
TOTAL	3144	4637	8151

8151

Use n.tables() and n.variables() to get the number of datatables in your study and the total number of variables (n.variables goes into the study files to count the actual number of variables)

```
In [12]: n.tables(JHS)
         n.variables(JHS)
```

66
4326

datatables.dict() will return a data frame with the datatables IDs (phtxxxxxx) and description of your study

```
In [13]: tablesdict <- datatables.dict(JHS)
         head(tablesdict)
```

pht	dt_study_name	dt_label
pht002539.v2	ESP_HeartGO_JHS_Subject_Phenotypes	Subject ID, ESP cohort, target capture used in seq
pht001948.v1	CSTA	Agatston score of all coronary section among par
pht001947.v1	CSIA	Approach to life B. Life style among participants
pht001968.v1	PPAA	Post physical activity monitoring among particip
pht001955.v1	ECHA	Echocardiographic abnormalities among particip
pht001952.v1	DPASS_DIET1	Dietary data (DPASS) among participants of the J

variables.dict() will return a data frame with the variables IDs (phvxxxxxx), their name in the study, the datatable where they come from and their description

```
In [14]: vardict <- variables.dict(JHS)
         head(vardict)
```

dt_study_name	phv	var_name	var_desc
ESP_HeartGO_JHS_Subject_Phenotypes	phv00165323.v2	SUBJID	Subject ID
ESP_HeartGO_JHS_Subject_Phenotypes	phv00165322.v2	ESP_Cohort	Cohort name [JHS]
ESP_HeartGO_JHS_Subject_Phenotypes	phv00165324.v2	ESP_phenotype	ESP Phenotype group (j
ESP_HeartGO_JHS_Subject_Phenotypes	phv00181282.v1	Sequence_center	Indicates where the sam
ESP_HeartGO_JHS_Subject_Phenotypes	phv00181283.v1	Target	Indicates target capture
ESP_HeartGO_JHS_Subject_Phenotypes	phv00181284.v1	ESP_race_selfreport	Self report race [Africar

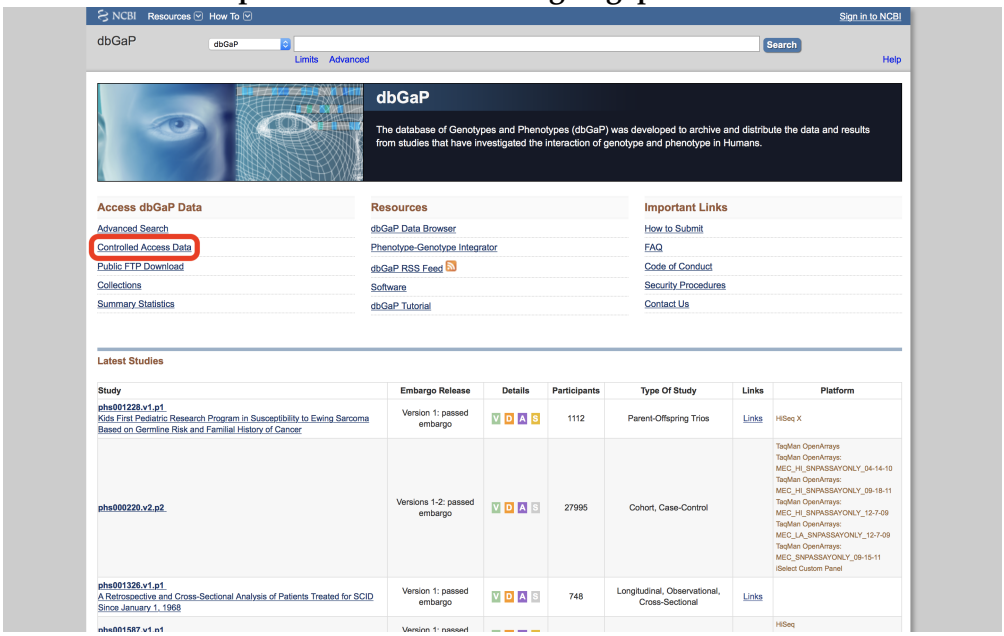
Now that we have explore our datasets, let's use sandboxR in order to clean our variables, and to gather them into a tree that will be easier to use for researchers. Note that for chapter 3, we will need to move and create a lot of files on your environment. It will be easier to use on your local computer than in the Jupyterhub environment.

1.4 Extract your study

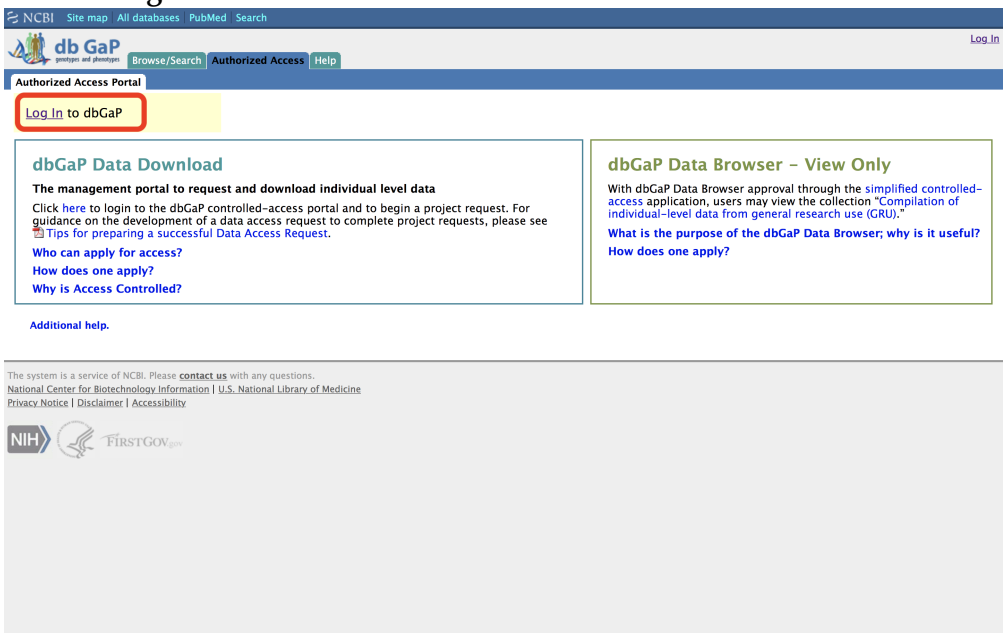
1.4.1 Get your dbGaP repository key

In order to download or decrypt your data from dbGap, you will need to request an access and to get a decryption key. Follow those steps to access your dbGaP repository key:

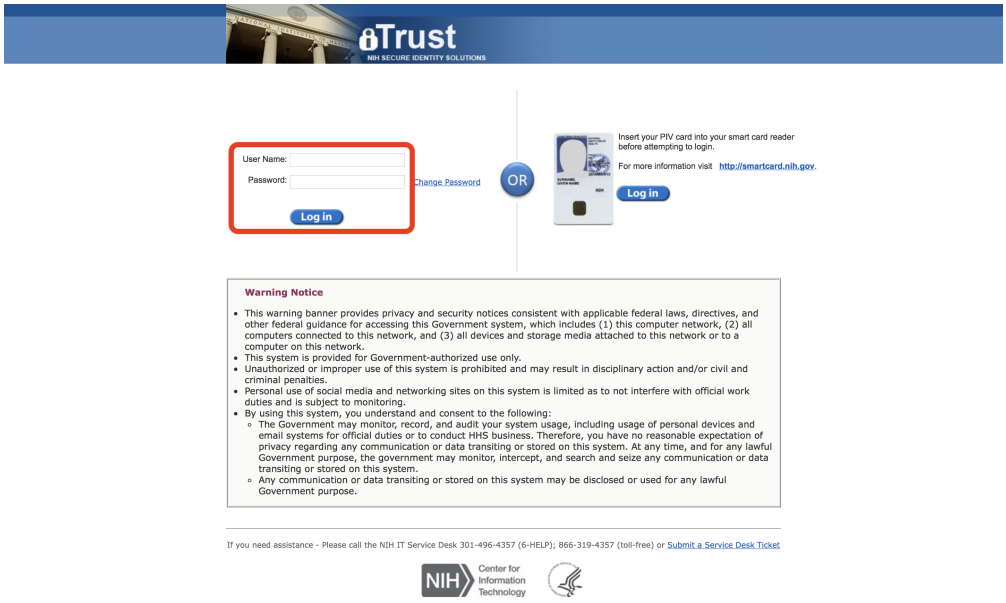
a. Go to https://www.ncbi.nlm.nih.gov/gap and click on “controlled access data”



b. Click on Log in to dbGaP

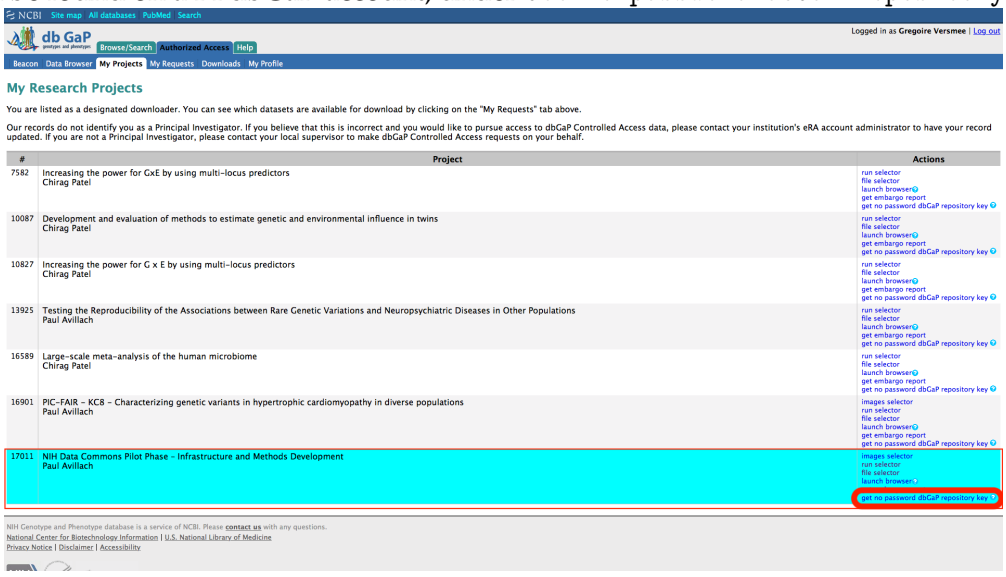


c. Identify yourself with your era common ID and password



d. Get a PI dbGaP repository key

In order to download the files and to decrypt them, you will need a decryption key. This key can be found on a PI dbGaP account, under `Get no password dbGaP repository key`



1.4.2 Decrypt the .ncbi_enc files

On dbGaP, the phenotypic files are encrypted. We created a decryption function that uses a dockerized version on sratoolkit. To use that function, you need to have docker installed on your device (www.docker.com). If you are using the dockerized version of this software (available at hub.docker.com/r/gversmee/dbgap2x), docker is already pre-installed, but you'll need to upload your key on the jupyter working directory. To try the function, we put some pre-encrypted files on the repo

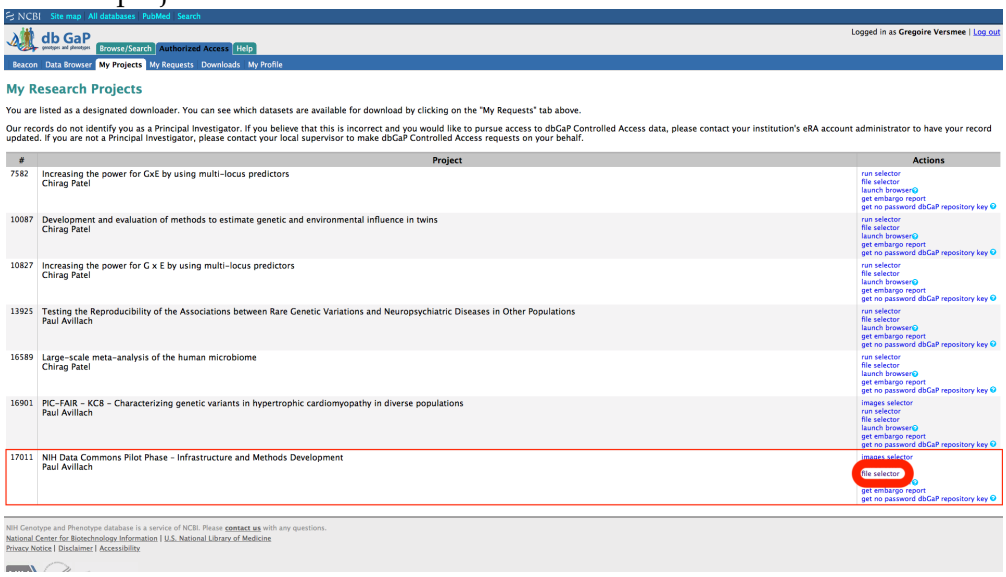
```
In [15]: key <- "prj_17011.ngc"
         files <- "encrypted_files"
         dbgap.decrypt(files, key)
```

You should see a "decrypted_files" directory in the directory where your encrypted files are located

1.4.3 3.1. Download dbGaP files

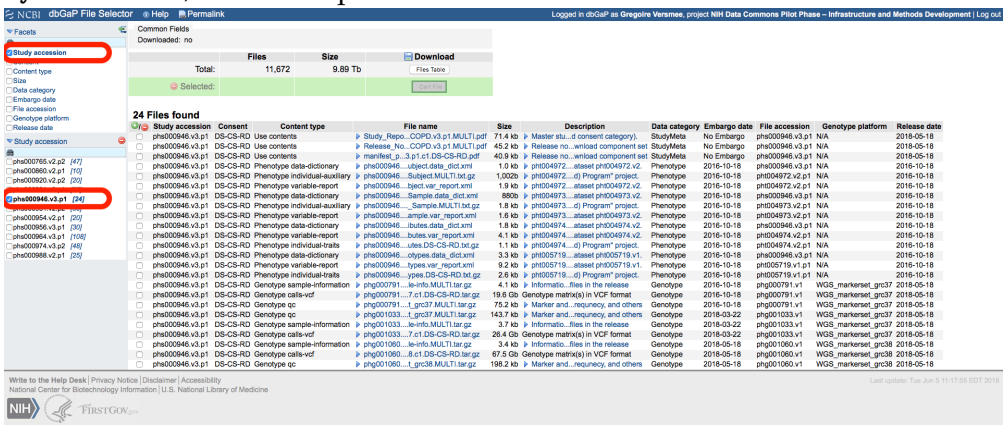
a. Click on "file selector"

This gives you access to the dbGaP file selector where you can find all the files available for the selected project.



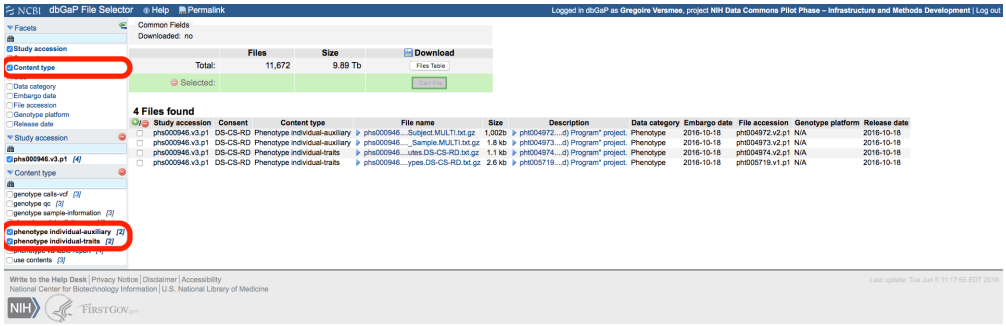
b. Filter by study accession

Here, we want to get the phenotypic data for the study "Early onset COPD", so after checking `Study accession`, we select "phs000946".



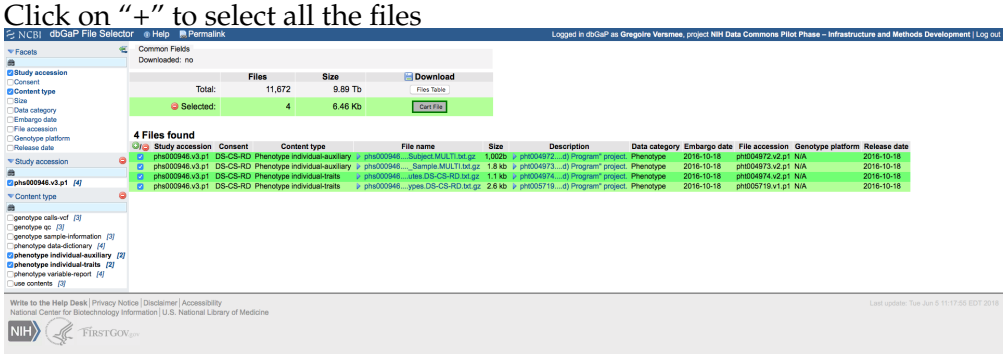
c.Filter again

Since we are only interested in getting the phenotypic data, let's filter by `Content type` and select `phenotype individual-auxiliary` and `phenotype individual-traits`



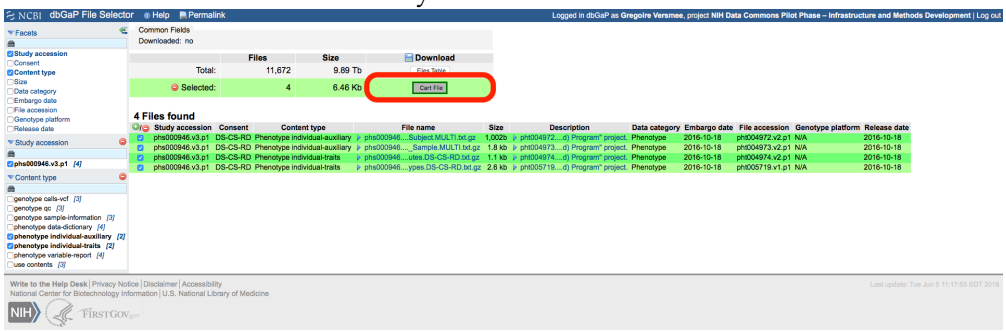
d. Select the files

Click on "+" to select all the files



e. Click on "Cart file"

This will downlaod a .krt file in your download folder



f. Download and decrypt the files with a simple command

```
In [16]: key <- "prj_17011.ngc"
         cart <- "cart_prj17011_201810151143.krt"
         dbgap.download(cart, key)
```

You should see in your working directory a new one name dbGaP-*** that contains your files