**Readme**

**Prerequisites:**

Diversity Browser is distributed as a iPython Jupyter notebook, to run it locally two software need to be installed.

**Install Python Jupyter Notebook**

Anaconda is the recommended software, please follow your platform documentation: <https://www.anaconda.com/distribution/>. There is alternative software available, more details could be found at <https://jupyter.org/install.html>.

**Install PLINK**

PLINK is a free, open-source whole genome association analysis toolset which is picked to do all analyses for the Diversity Browser. Please follow the documentation: <https://www.cog-genomics.org/plink/> to install the toolset

**Get Diversity Browser**

The most convenient way to get the tool is to go to the repository <https://github.com/cnr-ibba/IMAGE-DiversityBrowser>, click the green button, and choose Download Zip option. After unzipping the downloaded file, the tool is there for use.

Along with the tool, there are four data files:

1. IMAGE001\_23\_01\_BPW\_PIG.vcf: the pig reference dataset which contains 149 pigs that were genotyped with the IMAGE ‘*multispecies SNP-chip*’. The ‘*multispecies SNP-chip*’ was specifically designed to handle multiple species, in other words the chip contains around 10,000 SNPs per species, including pig, cattle, chicken, goat, and sheep.
2. SoI.vcf: the dataset which will be used as the example in the following demonstration
3. legend\_metadata.txt: the dataset which contains the phenotypic (origin and breed) data for the demonstration
4. pigsDataref.vcf: the middle file generated by PLINK from the pig reference dataset

**How to run Diversity Browser**

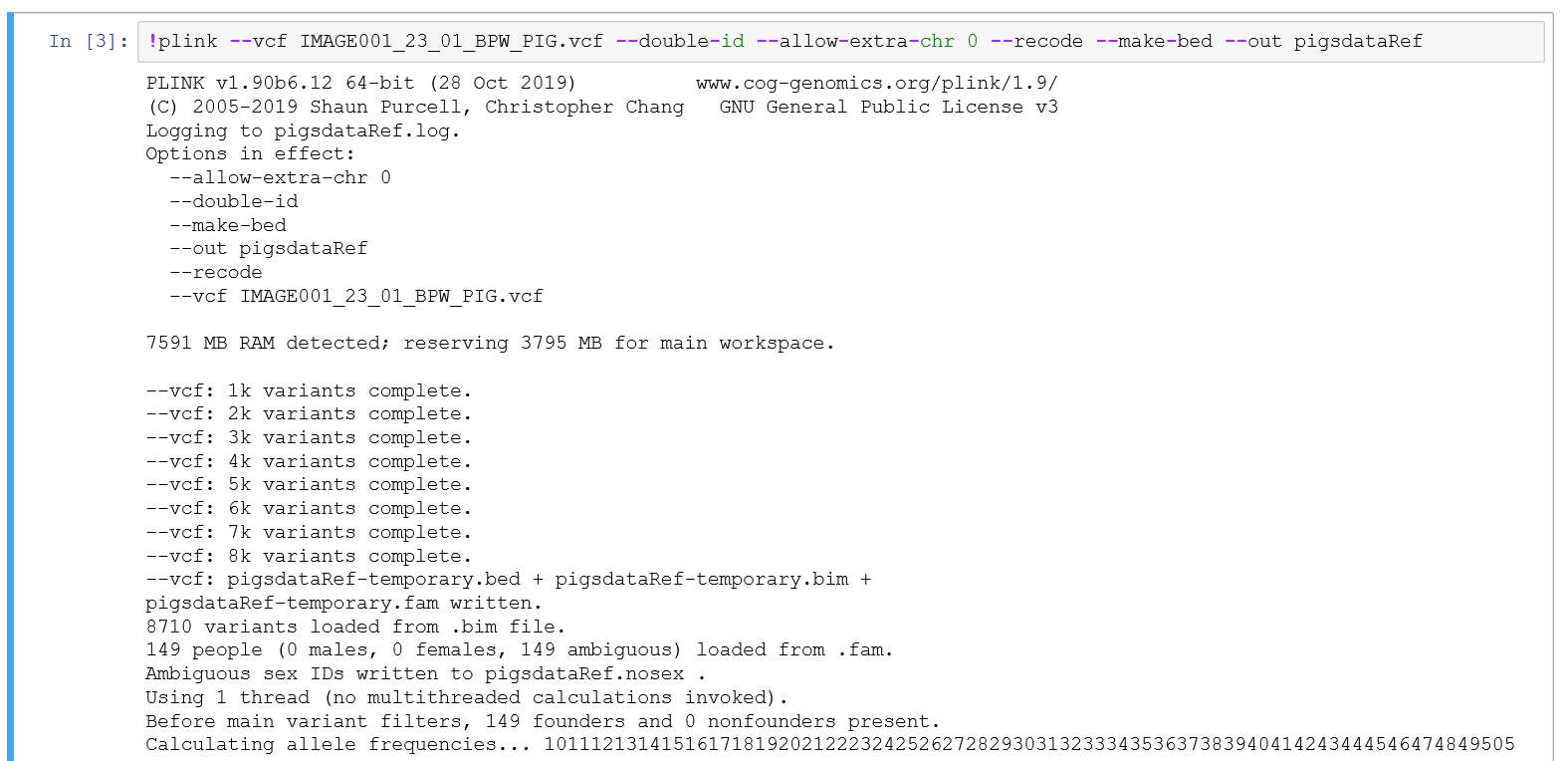
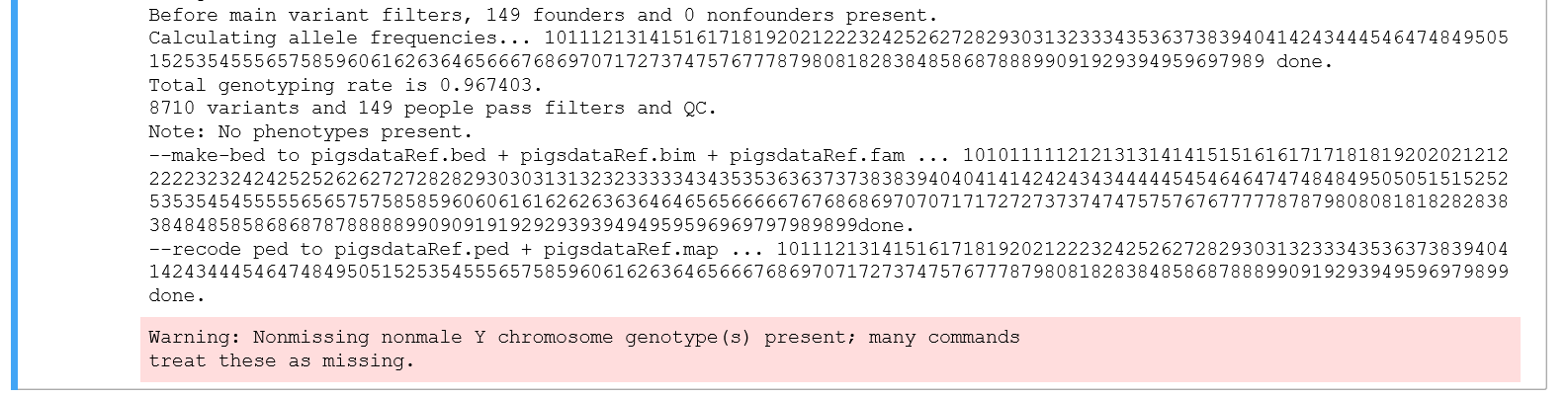
1. Start the notebook: open a terminal, enter the folder where the Diversity Browser is and type the command ‘jupyter notebook’. The notebook page will be opened in your default web browser and there is no notebook running. Click the New button to start a new notebook or double click one existing notebook in the file tab.
2. The working folder should be IMAGE-DiversityBrowser. To check, please type the following statement  
   import os  
   os.getcwd()  
   then click the run button, which will provide the output like this:  
   C:/Users/schokker/IMAGE-DiversityBrowser/  
   If not, please use command  
   os.chdir(“C:/Users/schokker/IMAGE-DiversityBrowser/”)  
   to the repository folder.
3. Save your samples data typed by the ‘*multispecies SNP-chip*’ in the same folder. Below we will use the example dataset soI.vcf to do the demonstration.
4. Create the reference dataset (after this step pigsDataref.vcf will be generated in the current folder)

*!plink --vcf IMAGE001\_23\_01\_BPW\_PIG.vcf --double-id --allow-extra-chr 0 --recode --make-bed --out pigsdataRef  
The used options are:*

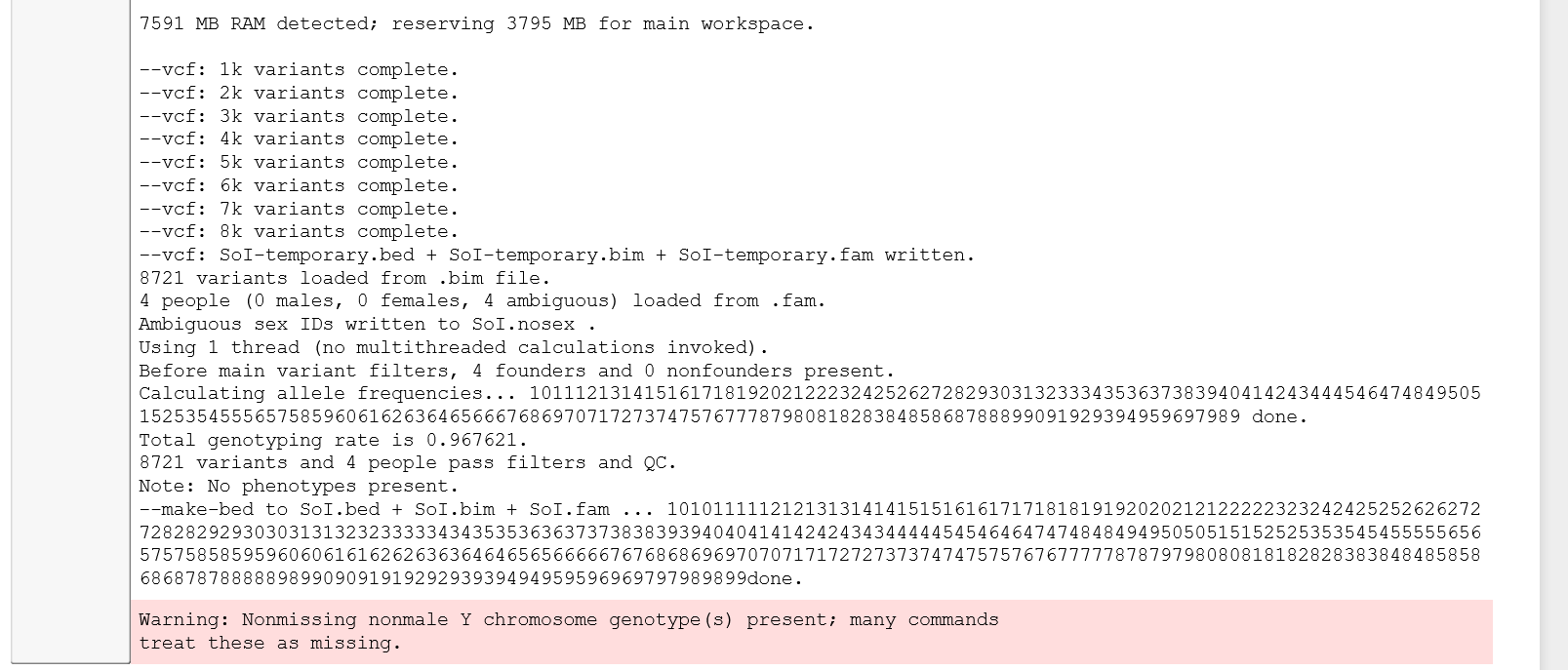
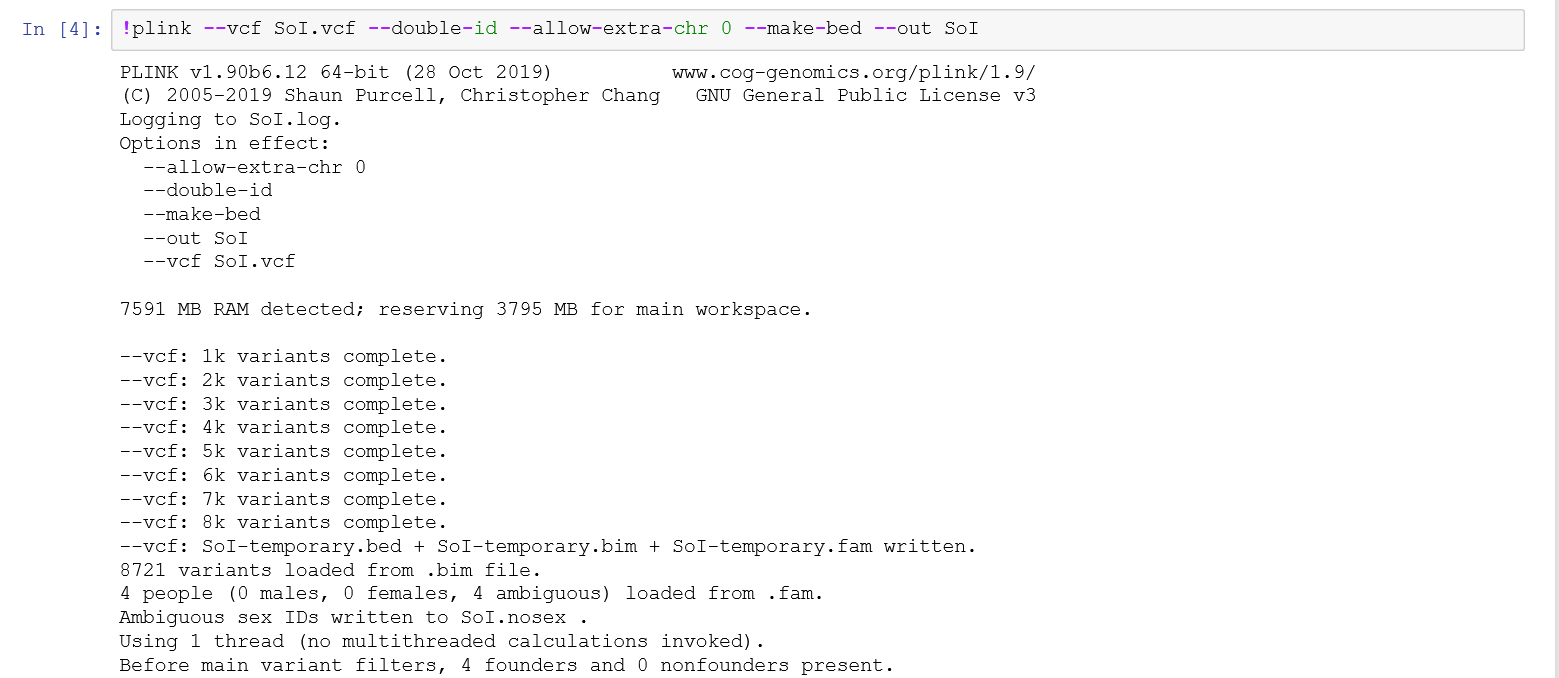
1. *--vcf:* a variant call file is used as input, here ‘*IMAGE001\_23\_01\_BPW\_PIG.vcf*’
2. *--double-id：*this means that two identifiers are allowed
3. *--allow-extra-chr*: this allows an extra chromosome, here chromosome zero
4. *–recode:* which will output the allele labels as they appear in the original; also, the missing genotype code is preserved if this is different from 0.
5. *--make-bed:* This option does the same as --recode but creates binary files
6. *--out:* This command created the output file, here ‘pigsdataRef’

*More details for the option usages please check the PLINK documentation.*

Below is the screenshot of messages generated by the PLINK.



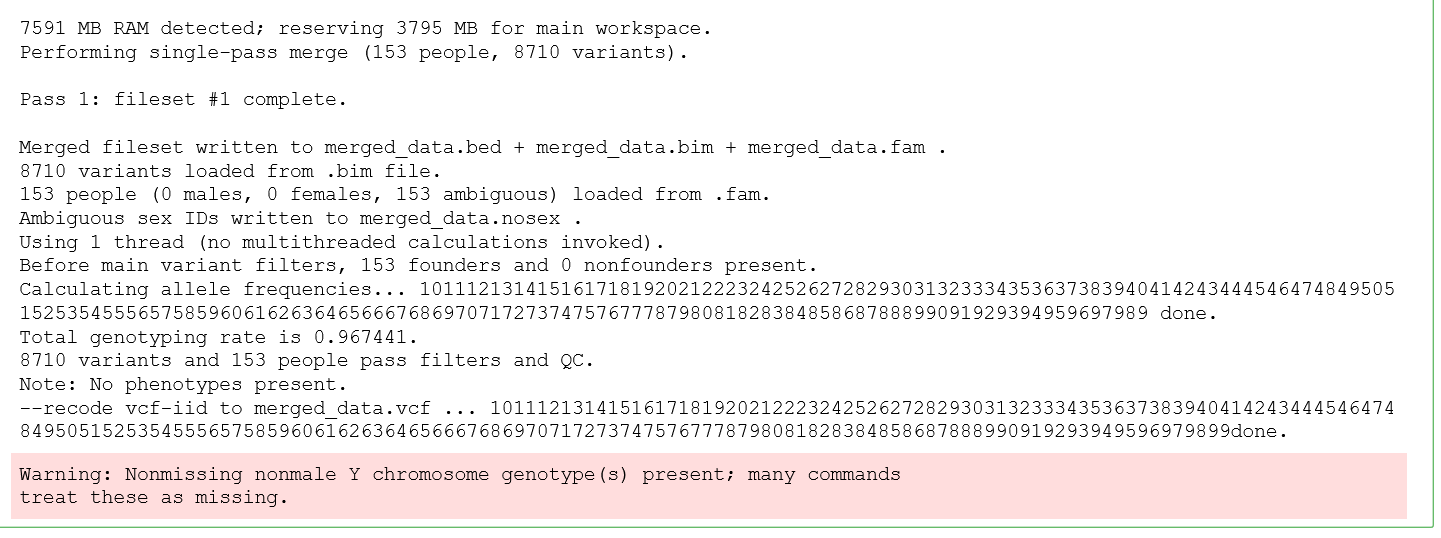
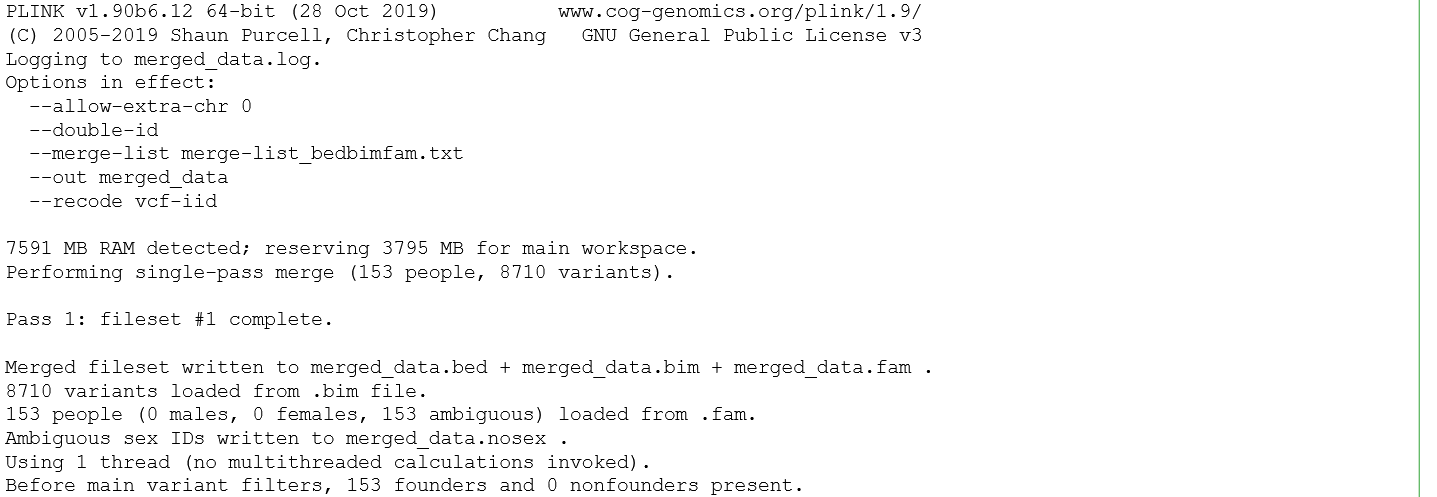
1. Similar to step 4, it is time to generate a file for the samples/dataset of interest. This is done by the next line of code, see below for the command and the output generated by the Notebook.



1. When both reference and sample of interest files are generated, it is time to do the merging which is done by the following code within PLINK:

*!plink --merge-list merge-list\_bedbimfam.txt --double-id --allow-extra-chr 0 --recode vcf-iid --out merged\_data*

This will generate the following screen output as well as the file *merged\_data.vcf*.



--merge: allows you to merge exactly one text fileset with the reference fileset.

The latter file will serve as input for the Principal Component Analysis (PCA).

1. Within PLINK it is possible to employ a PCA (code below), here we have set the number of principal components to 25, and generate a new CSV (comma separated value) output file named *PCA\_pigsdataRef,* whereone of the default files has the extension ‘eigenvec’.

*!plink --vcf merged\_data.vcf --double-id --allow-extra-chr 0 --pca 25 --out PCA\_pigsdataRef*

--pca: performs a principal components analysis (PCA) based on the variance-standardized relationship matrix

The output file, *PCA\_pigsdataRef*, contains the eigenvectors needed to create the ‘easy-to-interpret’ scatterplot figure.

1. The legend\_metadata.txt file is a tabular file which contains the metadata of each sample. There are four columns in the file separated by tab: Sample\_ID\_for\_Batchregistration, Plate\_Position\_, breed and origin, which contains the information for \*\*\*, location on the chip, breed and country of origin correspondingly. The file is loaded into Diversity Browser by the following command:

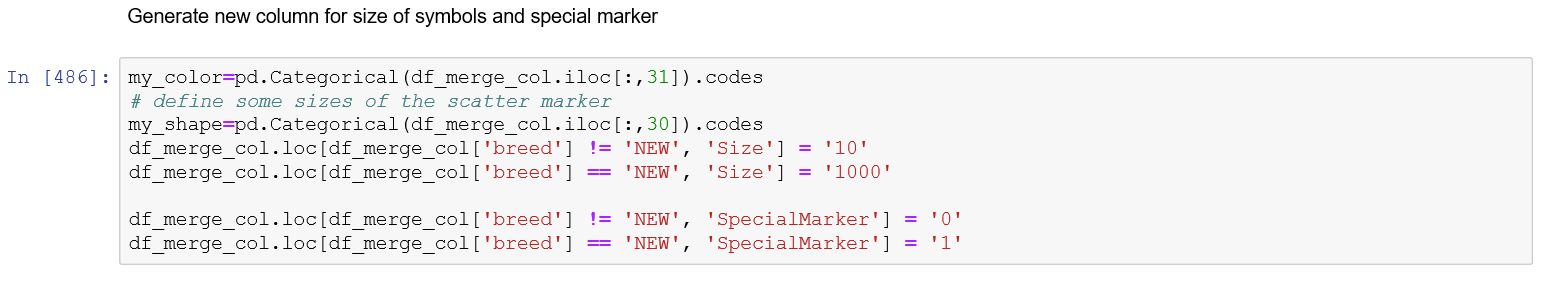
pd.read\_csv(), with separator (sep) set to a space by “ ” and explicitly setting the header to ‘*None’.*

1. The PCA result file, i.e. PCA\_pigsdataRef.eigenvec, is processed by pandas - Python Data Analysis Library (for convenience pandas is abbreviated to ‘pd’). In this PCA result file, each row contains a pig sample, the first 2 columns refer to the identifier, in our case they are identical identifiers, the other 25 columns refer to a coordinate on a principal component, where column 3 is principal component 1 and the next is one higher in ascending order up to principal component 25.

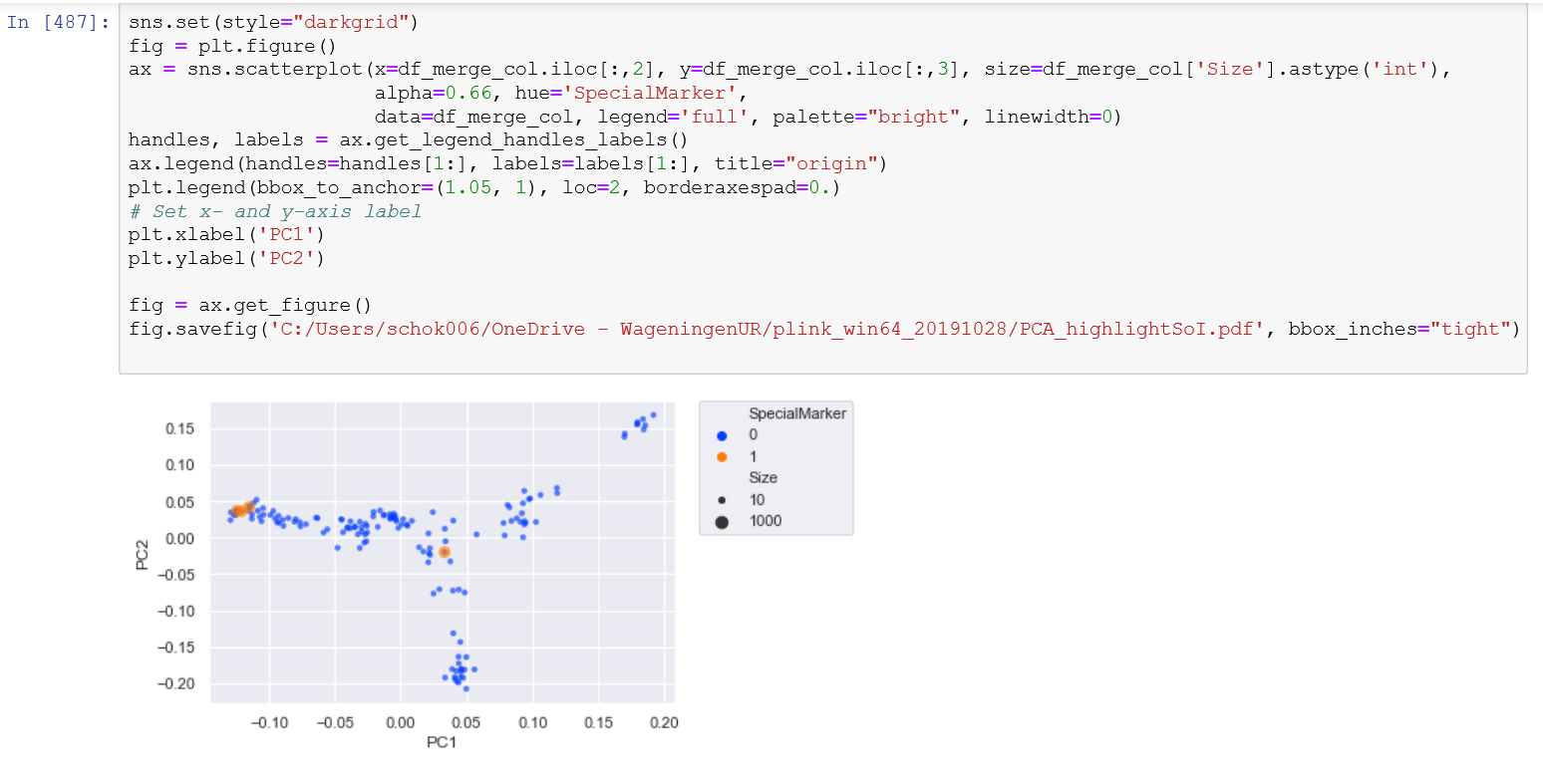
From this new data frame called [data], the identifier [IID] column is transformed by removing the “.CEL” text, to generate an identical format as the metadata file (legend\_metadata.txt). This transformation of the identifiers, i.e. column names, is necessary, because an exact match is required for the merge of the PCA result file and the metadata file to rule out loss of data. Below the screenshot of the Notebook with the actual code.

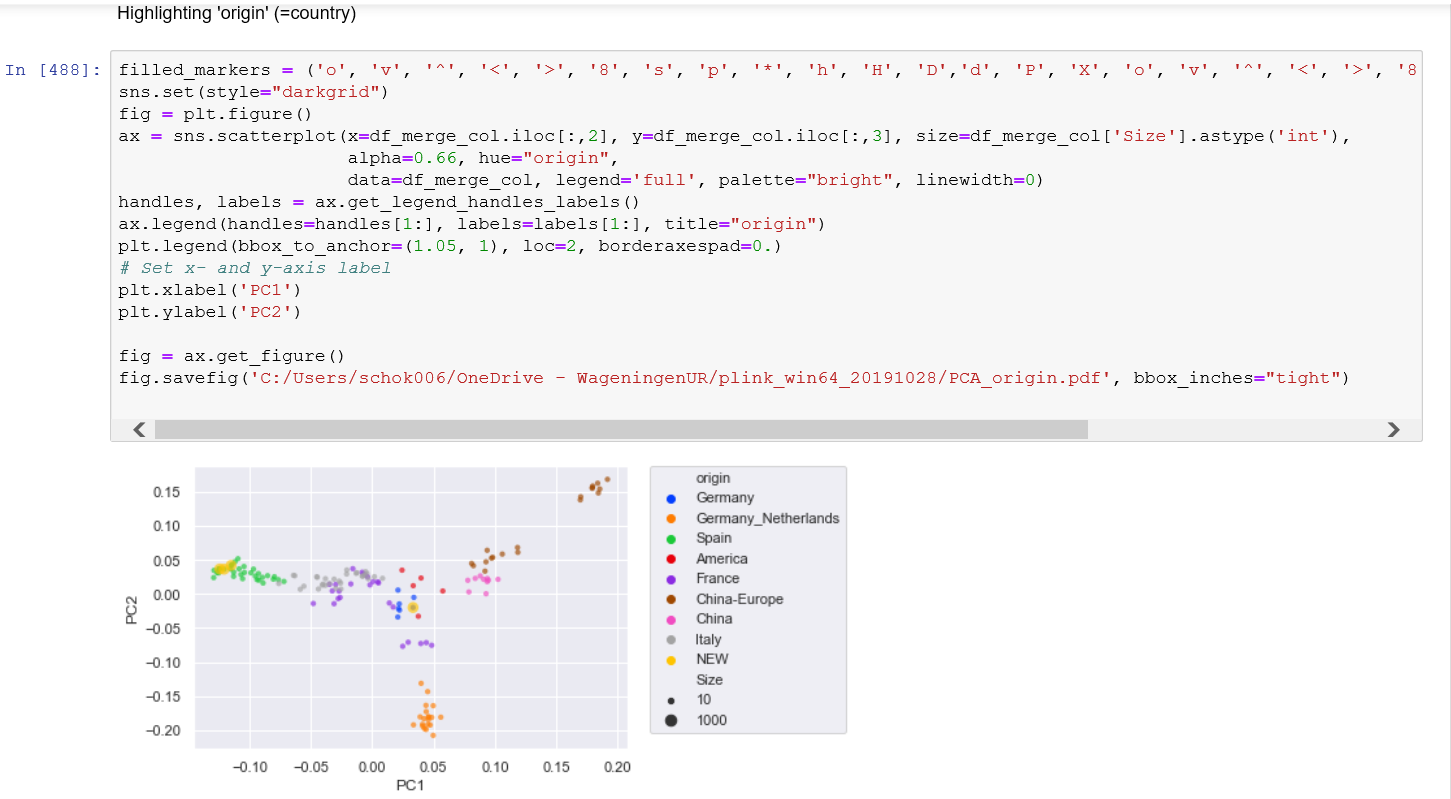


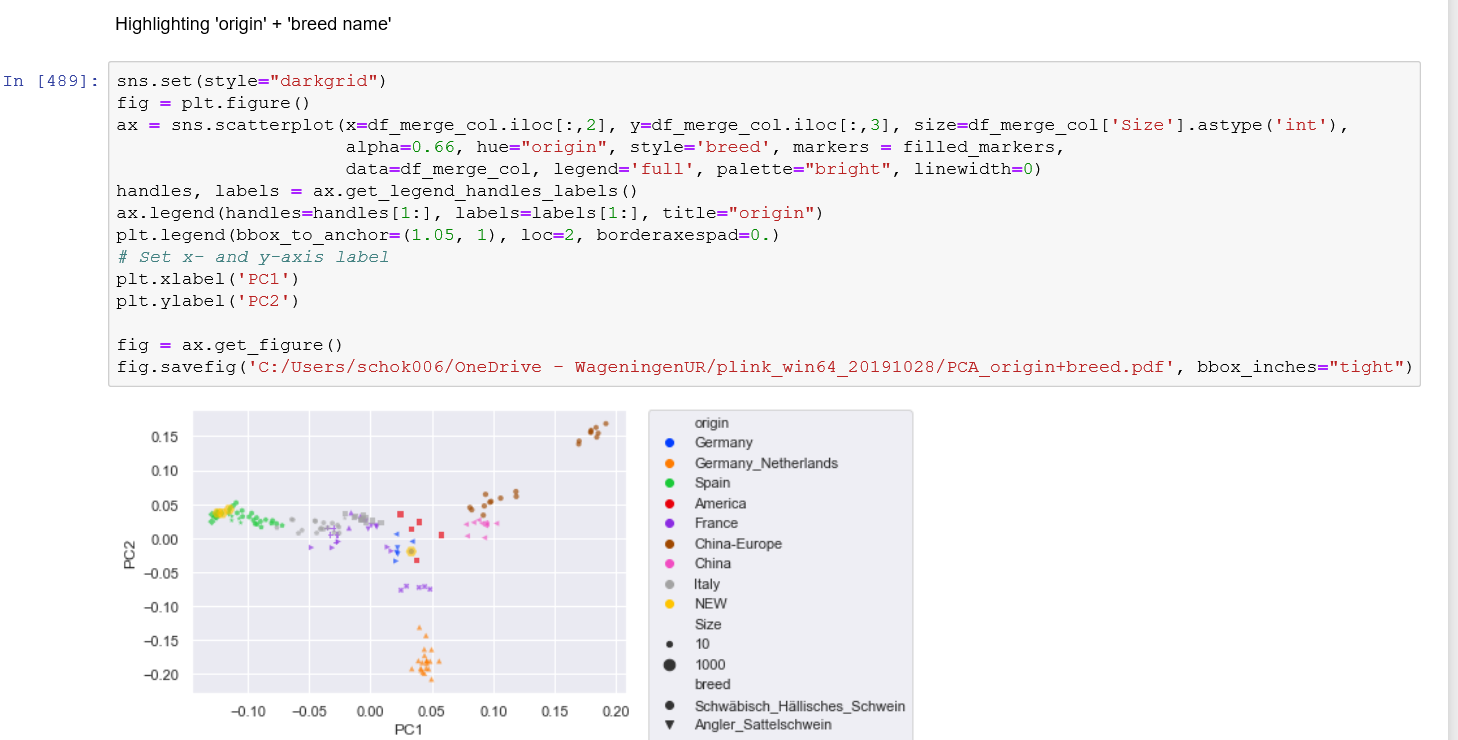
1. Next step is to generate code that can easily transform data to different features in the scatterplots. A variable [my\_color] is generated that contains different colors based upon column 31. The *.iloc* function refers to column 31 and the .*codes* function renames the categories. Column 31 of the **‘merged’** dataset contains the metadata information of **origin**. A similar approach is taken to generate a variable [my\_shape], which contains the different shapes, based upon the **breed names** which are in column 30 of the **‘merged’** dataset. And lastly, we generate two new columns *Size* and *SpecialMarker* based upon the column with the **breed** name, namely *Size* and *SpecialMarker*, both to highlight our sample(s) of interest.



The last three blocks of code are generating different flavors of highlighting the sample(s) of interest with the corresponding metadata (origin and/or breed). Below all three block of code with their corresponding output.







The last piece of code generates a three-dimensional (3D) representation of the scatterplot. This may be used when the first two axes do not give a clear separation of the different origins and/or breeds.

