Tutorial 6

Epidemiology

Step 0: If you have not done so already: *claim a virtual machine from infomdssLabA*

Login to the azure portal (https://portal.azure.com/),

Go to **Dashboard → infomdssLabA → infomdssLabA** and claim a virtual machine.

Login to your lab B vm (the default password is the same as in tutorial 1) and **change your password**.

```
$ passwd # choose your new password wisely
```

These actions are explained in tutorial 1.

Step 1: Download Epidemiology Data

The data for this tutorial is prepared on `transfer.sh`, which will be available for 14 days. The data contains two zip files and one .gz file (an open alternative for zip).

```
$ mkdir /home/labuser/epidemiology && cd "$_" # mkdir and cd
$ wget https://transfer.sh/F2bXi/CHD.zip
$ wget https://transfer.sh/10u2wK/MENO.zip
$ wget https://transfer.sh/mrAq9/T2D.zip
```

Step 2: Get Checksum Files

In order to check if all files have downloaded well, we use checksums, a checksum generates a unique string for an input. If the file would only change one bit, the checksum will be completely different. Which makes it a perfect way to check if all files are identical to the ones stored on the server.

Se first we need to download the checksums of the existing files:

```
$ wget https://raw.githubusercontent.com/Infomdss2018/infomdss/master/tutorial_6/check_sum_files/CHD.zip.sha1
$ wget https://raw.githubusercontent.com/Infomdss2018/infomdss/master/tutorial_6/check_sum_files/MENO.zip.sha1
$ wget https://raw.githubusercontent.com/Infomdss2018/infomdss/master/tutorial_6/check_sum_files/T2D.zip.sha1
```

Step 3: Execute Checksum

Now we can generate checksum of the data files we just downloaded and check if the generated check sums equal the expected check sums (downloaded in the previous step).

```
$ sha1sum -c CHD.zip.sha1
$ sha1sum -c MENO.zip.sha1
$ sha1sum -c T2D.zip.sha1
```

Checksum Errors

If all works well, you can expect the following message for each checksum:

```
sha1sum: cardio_c4d.zip.sha1: ok
```

If there problems with the checksum files, then you can expect an error:

```
sha1sum: <FILE>: no properly formatted sha1 checksum lines found
```

If the checksums do not match (which means the data-files do not correspond with the data on the server), you see a warning like this:

```
sha1sum: warning: 1 computed checksum did not match
```

In case of an error or warning, you can try to download the data-files and checksum-files.

Step 4: Unzip Epidemiology Data

In order to unzip the `.zip` files, you can use the `unzip`-command from previous tutorial. As you will see, there are two files in every `.zip`, one is the data and the other is a sample for testing purposes.

```
$ unzip CHD.zip -d data/
$ unzip MENO.zip -d data/
$ unzip T2D.zip -d data/
```

Step 5: Give Permissions

In order to work with the data, spark needs to have access to the directories and files, so we can give permission with the `chmod`-command:

```
$ chmod 777 data/*
```

Step 6: Open Notebook

Now all data is downloaded and prepared, we can start with the notebooks. You can open ayour notebook by navigating to https://vmlabaXXX.westeurope.cloudapp.azure.com:8000, where you have to replace XXX with the number of your VM.

You should get a login screen where you can fill-in the username and password of your VM (e.g. “labuser” and your updated password).

After login you can choose if you want to continue this assignment in R or python, this tutorial is written in python, so you can select **new → python 3 spark-local** and start with the assignments.

Assignments: Preprocess data

These data are not standard data per individual, but on the marker (SNPid). Usually this is the first column in the dataset (you can see things as rs1233445). Thus, you can merge the datasets based on this rs number.

For each of the following questions, create a small script below to preprocess the data files.

TIP: It is good practice to develop and test the scripts using only a small subset of the data. Only when you are pretty sure that the script works as intended, you let it process the entire data file. Saves time!

1. Load the three datasets within your Spark environment.

This script can help you getting started, a new context is created just like in the Spark tutorial from last week.

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("epidemiology").getOrCreate()
sc = spark.sparkContext

meno_RDD = sc.textFile("<server path to your file>")
card_RDD = sc.textFile("<server path to your file>")
trans_RDD = sc.textFile("<server path to your file>")
```

Note that a good way of manipulating data is by using data frames, which can be created from your RDD data, for example we can create a dataframe, `meno_df`, from `meno_RDD`:

```
meno_header = meno_RDD.first()

# split data set
temp_var = meno_RDD.filter(lambda line: line != meno_header).map(lambda k: k.split("\t"))
meno_df = temp_var.toDF(header.split("\t"))
```

Now we have a dataframe, we can for example show the first twenty rows:

```
meno_df.show()
```

```
+----------+-------+-------+----------+------+------+-------+
|MarkerName|allele1|allele2|HapMap_eaf|effect|stderr|      p|
+----------+-------+-------+----------+------+------+-------+
|      rs10|      a|      c|      0.03| -0.01|  0.05|9.0e-01|
| rs1000000|      a|      g|      0.37| -0.01|  0.02|5.8e-01|
|rs10000010|      t|      c|      0.57|  0.01|  0.02|6.7e-01|
[...]
```

Or we can look-up the types of data in every column:

```
meno_df.schema
```

```
StructType(List(StructField(MarkerName,StringType,true),StructFiel
d(allele1,StringType,true),StructField(allele2,StringType,true)[..
.]
```

Or we can show all column names:

```
meno_df.columns
```

```
['MarkerName', 'allele1', 'allele2', 'HapMap_eaf', [...]
```

Or we can create a new matrices with a selection of available columns:

```
meno_df.select(["MarkerName", "allele1", "allele2"]).show()
```

```
+----------+-------+-------+
|MarkerName|allele1|allele2|
+----------+-------+-------+
|      rs10|      a|      c|
| rs1000000|      a|      g|
|rs10000010|      t|      c|
[...]
```

2. Merge the datasets on markername, into one integrated raw data file titled “`epi.csv`” (hereafter: “data file”).

Tip: You will notice that the diabetes set does not have markername. However, it has a column named `chr:Position`. This is a combined column, which is the same as the combination of two columns in the CAD file (`chr` and `bp_hg19`).

3. Record how many SNPs (markers) in each dataset cannot be merged.

4. Further sanitise the data file by removing any empty lines, unused columns. Identical rows?

Stop your virtual machine at the end of the workshop