Pydpiper Virtual Machine

Downloading the Virtual Machine

Download a tarball of the Virtual Machine (VM) here (last updated: June 24, 2015, **PydPiper version 1.14-beta, https://github.com/Mouse-Imaging-Centre/pydpiper/releases**). Alternatively, you can download it using wget:

```
wget --tries=3 --retry-connrefused -c http://repo.phenogenomics.ca/repo/MICe-software/MICe-Pydpiper-v1.14-beta.
tar.gz -O MICe-Pydpiper-v1.14-beta.tar.gz

# unpack the tarball
tar xzvf MICe-Pydpiper-v1.14-beta.tar.gz

# using Oracle VirtualBox you can now add the machine using the .vbox file (MICe-Pydpiper.vbox)
```

⚠ The Virtual Machine is originally set to use 1 processor. Prior to running the VM, you can increase the number of processors in order to speed up the test cases below.

Test data

The virtual machine contains a folder on the Desktop with test data for the 4 core applications: MAGet.py, MBM.py, registration_chain.py and twolevel_model_building.py. The following sections describe how to run each of these and provides information to verify that the tests ran successfully.

Testing the MBM.py application

Open a shell, and run the following:

```
cd /home/mice-pydpiper/Desktop/Pydpiper_test_data/
mkdir test_MBM
cd test_MBM/
#
# The following command will run MBM.py on a single processor using 2GB of memory. It will sequentially
# perform a linear 6 parameter, a linear 12 parameter and a non-linear registration stage. Finally a set
# of files will be created that can be used for statistical analysis
#
MBM.py  \
--num-executors=1 \
--proc=1 \
--mem=2 \
--pipeline-name=striatum_mutants \
--registration-method=mincANTS \
--nlin-protocol=/home/mice-pydpiper/Desktop/Pydpiper_test_data/MBM/mincANTS_protocol_224_micron_pydpiper.csv \
--init-model=/home/mice-pydpiper/Desktop/Pydpiper_test_data/MBM/initial-model/test-data-atlas.mnc \
--lsq6-large-rotations \
--lsq6-large-rotations-parameters=mousebrain \
--lsq12-subject-matter=mousebrain \
--calc-stats \
/home/mice-pydpiper/Desktop/Pydpiper_test_data/MBM/test_files_striatum_mutants/mouse_*mnc
```

When the application starts it will provide information about the total amount of stages in the pipeline, and how many of these stages are duplicates (redundant). Then it will launch the server and the executor. The output should be similar to this:

```
816 stages skipped as redundant. 1209 stages to run.
Daemon is running on port: 7766
The object's uri is: PYRO://10.0.2.15:7766/7f000101088120c0e3d65ae2750c770e58
Client registered (banzai!): PYRO://10.0.2.15:7767/7f000101088720c0e3d733000000a76047
```

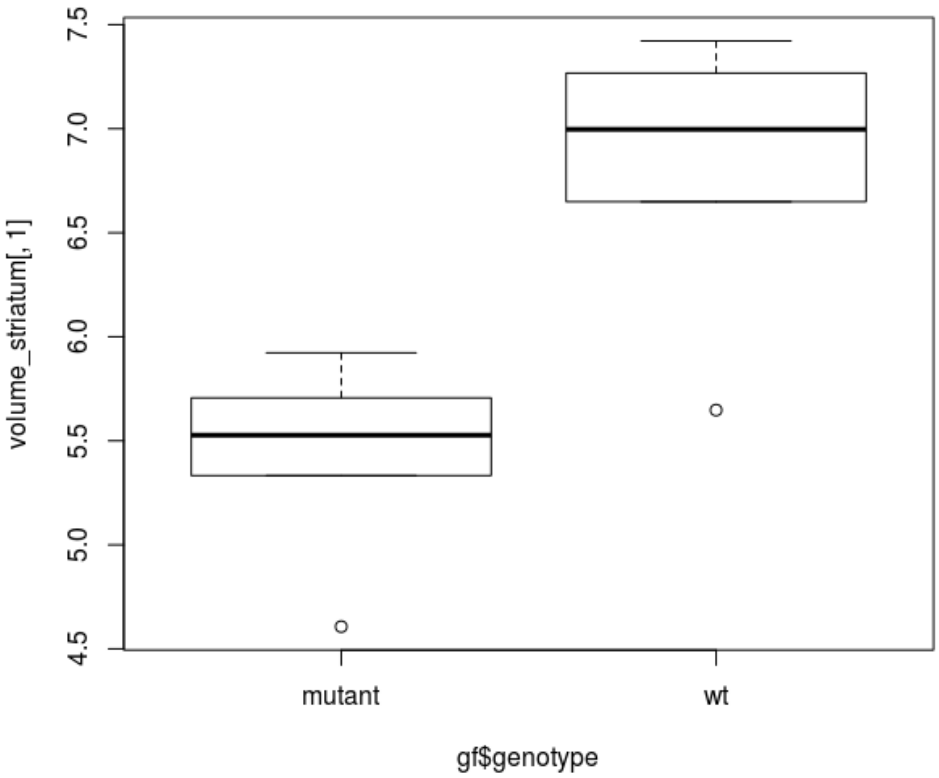
Using a single core (processor), this will take about 10 hours (speed up the computation time significantly by adding either additional executors, or by increasing the number of processors per executor!)

Next, use R and RMINC to determine the difference between the two input data sets. First, create a csv file that points to the absolute Jacobian determinants.

```
echo absolute_jacobians,genotype > absolute_jacobians_and_genotypes.csv; \
for file in striatum_mutants_processed/*; \
do base=`basename $file`; if [[ $base =~ "mutant" ]]; \
then type="mutant"; else type="wt"; fi; pwd=`pwd`; \
echo ${pwd}/${file}/stats-volumes/${base}-final-nlin_with_additional_inverted_absolute_log_determinant.
mnc,$type; \
done >> absolute_jacobians_and_genotypes.csv

#
# Run R, and load RMINC
#

R
# from here on, the character ">" indicates the command prompt in R
> library(RMINC)
# now load the data mapping
> gf <- read.csv("absolute_jacobians_and_genotypes.csv")
# extract volume information for all structures
> volume_striatum <- anatGetAll(gf$absolute_jacobians, atlas="/home/mice-pydpiper/Desktop/Pydpiper_test_data/MBM
/rough_striatum_segmentation.mnc", defs="/home/mice-pydpiper/Desktop/Pydpiper_test_data/MBM
/mapping_for_striatum.csv")
# the volumes for the striatum
> volume_striatum
       striatum
 [1,]  7.254975
 [2,]  5.922732
 [3,]  7.267388
 [4,]  5.706052
 [5,]  7.421839
 [6,]  5.565661
 [7,]  5.647604
 [8,]  4.606988
 [9,]  6.649315
[10,]  5.332636
[11,]  6.738569
[12,]  5.487864
attr(,"atlas")
[1] "/home/mice-pydpiper/Desktop/Pydpiper_test_data/MBM/rough_striatum_segmentation.mnc"
attr(,"class")
[1] "anatUnilateral" "anatMatrix"     "matrix"
attr(,"anatIDs")
[1] "1"
# plot the difference between the two genotypes:
> plot(volume_striatum[,1] ~ gf$genotype)
```




```
# finally we can look at the percentage difference:
> tapply(volume_striatum[,1], gf$genotype, mean)
  mutant       wt
5.436989 6.829948
> (tapply(volume_striatum[,1], gf$genotype, mean)[1] - tapply(volume_striatum[,1], gf$genotype, mean)[2] ) /
tapply(volume_striatum[,1], gf$genotype, mean)[2] * 100
   mutant
-20.39488
```

Testing the MAGeT.py application

Open a shell, and run the following:

```
cd /home/mice-pydpiper/Desktop/Pydpiper_test_data/
mkdir test_MAGeT
cd test_MAGeT/

MAGeT.py  \
--create-graph \
--num-executors=1 \
--proc=1 \
--mem=2 \
--registration-method=minctracc \
--atlas-library=/home/mice-pydpiper/Desktop/Pydpiper_test_data/MAGeT/library/ \
--lsq12-protocol=/home/mice-pydpiper/Desktop/Pydpiper_test_data/MAGeT/default_linear_MAGeT_prot.csv \
--nlin-protocol=/home/mice-pydpiper/Desktop/Pydpiper_test_data/MAGeT/default_nlin_MAGeT_minctracc_prot.csv  \
/home/mice-pydpiper/Desktop/Pydpiper_test_data/MAGeT/input-files/input_*mnc
```

When the application starts it will provide information about the total amount of stages in the pipeline, and how many of these stages are duplicates (redundant). Then it will launch the server and the executor. The output should be similar to this:

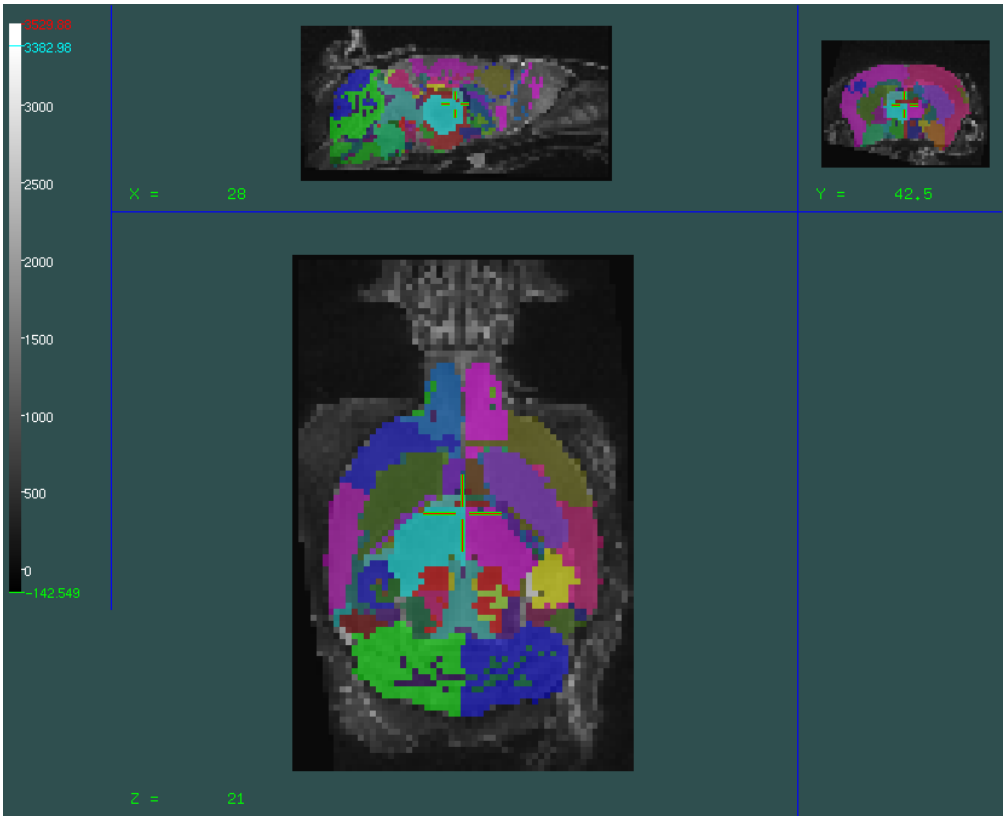
```
80 stages skipped as redundant.  68 stages to run.
Daemon is running on port: 7766
The object's uri is: PYRO://10.0.2.15:7766/7f00010151b920c0e6f789e2e537423cf3
Client   registered   (banzai!): PYRO://10.0.2.15:7767/7f00010151bf20c0e6f7a7000000b19955
```


Using a single core (processor), this will take about 20 minutes (time for a snack!)

Verify the label files using Display:

```
Display  /home/mice-pydpiper/Desktop/Pydpiper_test_data/MAGeT/input-files/input_1.mnc -label /home/mice-pydpiper
/Desktop/Pydpiper_test_data/test_MAGeT/input_1/labels/input_1_votedlabels.mnc
```

After changing the background colour to gray, press space bar, then d, and then d again (see more details here: http://en.wikibooks.org/wiki/MINC/VisualTools/Display), the labels will look something like this:




```
# the same thing can be done for the other file
Display  /home/mice-pydpiper/Desktop/Pydpiper_test_data/MAGeT/input-files/input_5.mnc -label /home/mice-pydpiper
/Desktop/Pydpiper_test_data/test_MAGeT/input_5/labels/input_5_votedlabels.mnc
```

Testing the twolevel_model_building.pl application

Open a shell, and run the following:

```
cd /home/mice-pydpiper/Desktop/Pydpiper_test_data/
mkdir test_twolevel_model_building
cd test_twolevel_model_building/

twolevel_model_building.py \
--num-executors=1 \
--proc=1 \
--mem=2 \
--init-model=/home/mice-pydpiper/Desktop/Pydpiper_test_data/twolevel_model_building/init-model
/file_1_timepoint_1.mnc \
--nlin-protocol=/home/mice-pydpiper/Desktop/Pydpiper_test_data/twolevel_model_building
/mincANTS_example_nlin_protocol.csv  \
--stats-kernels=0.4,0.3 \
--registration-method=mincANTS \
--pipeline-name=test-two-level \
--create-graph \
/home/mice-pydpiper/Desktop/Pydpiper_test_data/twolevel_model_building/filelist.csv
```

When the application starts it will provide information about the total amount of stages in the pipeline, and how many of these stages are duplicates (redundant). Then it will launch the server and the executor. The output should be similar to this:

```
65 stages skipped as redundant.  334 stages to run.
Daemon is running on port: 7766
The object's uri is: PYRO://10.0.2.15:7766/7f000101525120c0ea68f609fe483a2a49
Client   registered   (banzai!): PYRO://10.0.2.15:7767/7f000101525720c0ea6934000000850a1a
```

Using a single core (processor), this will take about 12 hours.

Testing the registration_chain.py application

Open a shell, and run the following:

```
cd /home/mice-pydpiper/Desktop/Pydpiper_test_data/
mkdir test_registration_chain
cd test_registration_chain/

registration_chain.py \
--avg-time-point=2 \
--init-model=/home/mice-pydpiper/Desktop/Pydpiper_test_data/twolevel_model_building/init-model
/file_1_timepoint_1.mnc \
--nlin-protocol=/home/mice-pydpiper/Desktop/Pydpiper_test_data/twolevel_model_building
/mincANTS_example_nlin_protocol.csv \
--registration-method=mincANTS \
--stats-kernels=0.5,1.0 \
--num-executors=1 --proc=1 \
/home/mice-pydpiper/Desktop/Pydpiper_test_data/twolevel_model_building/filelist.csv
```

When the application starts it will provide information about the total amount of stages in the pipeline, and how many of these stages are duplicates (redundant). Then it will launch the server and the executor. The output should be similar to this:

```
254 stages skipped as redundant.  279 stages to run.
Daemon is running on port: 7766
The object's uri is: PYRO://10.0.2.15:7766/7f000101712d20c0eab2310a07ced16572
Client   registered   (banzai!): PYRO://10.0.2.15:7767/7f000101713320c0eab25a0000001d0f53
```

Using a single core (processor), this will take about 2 hours (more than enough time for a special lunch, and some good coffee!)

Accessing the Virtual Machine through ssh

The virtual machine has been set up such that you can ssh into it from the parent operating system:

```
# ssh using the VM's username: "mice-pydpiper", and credentials: "MICe":
ssh -p 3022 mice-pydpiper@127.0.01

# an example of copying a file into the VM's Desktop (notice here the capital P!):
scp -P 3022 file.mnc mice-pydpiper@127.0.01:/home/mice-pydpiper/Desktop
# or with rsync
rsync -cav -e 'ssh -p 3022' --progress file.mnc mice-pydpiper@127.0.01:/home/mice-pydpiper/Desktop/
```

Information on how this was setup:

http://stackoverflow.com/questions/5906441/how-to-ssh-to-a-virtualbox-guest-externally-through-a-host