
DeepGS Manual

An R toolkit based on deep learning for genomic selection

Version 1.2

June 5th, 2018

Authors: Wenlong Ma, Zhixu Qiu, Jie Song, Qian Cheng, Chuang Ma

Contact:

Wenlong Ma: mawenlong_nwsuaf@163.com

Dr. Chuang Ma: chuangma2006@gmail.com

Table of Contents

Brief introduction

DeepGS Manual	1
1. DeepGS installation	3
1.1 DeepGS-CPU Docker installation.....	3
1.1.1 Verify if Docker is installed correctly	4
1.1.2 DeepGS installation from Docker Hub	4
1.2 DeepGS-GPU Docker installation	5
1.3 Quickly start	5
2. GS cross-validation (CV) experiment design	5
3. GS evaluation	6
3.1 Train DeepGS model.....	6
3.2 Phenotype prediction based on trained GS model	7
3.3 GS prediction performance evaluation.....	7
4. Ensemble GS model training	7
5. Source codes availability	8
6. How to access help	8
7. References.....	8

Brief introduction:

DeepGS is a docker-based R toolkit that aims to predict phenotypes using genome-wide genotypic markers with an advanced machine learning technique (deep learning). This toolkit contains a series of functions required for genomic selection (GS) cross-validation experiment design, GS evaluation, linearly combining the predictions from different GS models. DeepGS also takes advantage of machine learning technologies for GS prediction, with high prediction performance, using deep convolutional neural network. Hence DeepGS is a versatile GS analysis pipeline covering GS experiment design, evaluation, and combination of different GS predictions.

1. DeepGS installation

1.1 DeepGS-CPU Docker installation

- For Windows (Test on Windows 10 Enterprise version):
 - a) Download the installer from following URL:
<https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe>;
 - b) Double click the EXE file to open it;
 - c) Follow the wizard instruction and complete installation;
 - d) Search docker, select **Docker for Windows** in the search results and click it.
- On Mac OS X (Test on macOS Sierra version 10.12.6 and macOS High Sierra version 10.13.3):
 - a) Download the installer from following URL:
<https://download.docker.com/mac/stable/Docker.dmg>;
 - b) Double click the DMG file to open it;
 - c) Drag the docker into Applications and complete installation;
 - d) Start docker from Launchpad by click it.
- On Ubuntu (Test on Ubuntu 14.04 LTS and Ubuntu 16.04 LTS):
 - a) Go to <https://download.docker.com/linux/ubuntu/dists/>, choose your Ubuntu version, browse to [pool/stable](#) and choose [amd64](#), [armhf](#), [ppc64el](#) or [s390x](#). Download the DEB file for the Docker version you want to install;
 - b) Install Docker, supposing that the DEB file is download into following path:

```
/home/docker-ce_<version-XXX>~ubuntu_amd64.deb
```

1. `sudo dpkg -i /home/docker-ce-<version-XXX>~ubuntu_amd64.deb`
2. `sudo apt-get install -f`

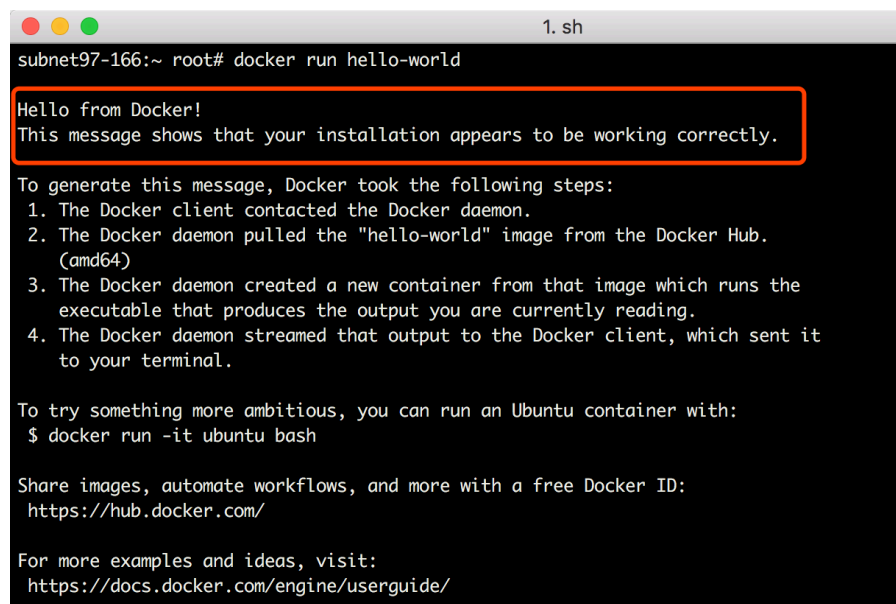
1.1.1 Verify if Docker is installed correctly

Once Docker installation is completed, we can run **hello-world** image to verify if Docker is installed correctly. Open terminal in Mac OS X and Linux operating system and open CMD for Windows operating system, then type the following command:

```
$ docker run hello-world
```

Note: root permission is required for Linux operating system.

If docker is installed successfully, you can see the following messages.



```
subnet97-166:~ root# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
```

- **Note:** considering that differences between different computers may exist, please refer to official Docker installation manual (<https://docs.docker.com/install>) if instructions above don't work.

1.1.2 DeepGS installation from Docker Hub

For Mac OS X and Linux operating systems, open the terminal, for Windows operating system, open CMD. Typing the following command:

1. `# Pulling DeepGS from Docker DeepGS`

```
2. $ docker pull malab/deepgs_cpu
```

If DeepGS is installed correctly, type “docker images”, you will see the following messages:

```
subnet97-166:~ root# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
malab/deepgs_cpu    latest      f2b869a87327     2 hours ago     3.14GB
```

1.2 DeepGS-GPU Docker installation

We also provided DeepGS GPU version for a better computation efficiency, which needs a series of operating environment configuration, including CUDA, nvidia-docker and so on. Please refer to

[https://github.com/cma2015/DeepGS/blob/master/DeepGS_GPU_installation.m](https://github.com/cma2015/DeepGS/blob/master/DeepGS_GPU_installation.md)
[d](#).

1.3 Quickly start

Once DeepGS is installed successfully, type the following command to start DeepGS:

```
1. $ docker run -it -v /host_directory_of_dataset:/home/data
    malab/deepgs_cpu R
2. # Supposing that users' private dataset is located in the directory "/home/test", then change the red colored words above (/host_directory_of_dataset) to host directory (/home/test)
3. library(DeepGS)
4. setwd("/home/data/")
```

Important: the directory ("/home/data/") is a virtual directory in DeepGS Docker image. In order to use private dataset more easily, the parameter “-v” is strongly recommended to mount host directory of dataset to DeepGS image.

2. GS cross-validation (CV) experiment design

DeepGS provides t-fold cross-validation procedure to generate CV samples index for a GS training experiment.

```
1. ## 5-fold cross validation
2. b <- cvSampleIndex(sampleNum = 2000, cross = 5, seed = 1)
```

3. GS evaluation

DeepGS provides a flexible framework for GS model design. The default parameters' effectiveness has been demonstrated in predicting eight phenotypic traits on a population of 2,000 Iranian bread wheat (*Triticum aestivum*) lines from the wheat gene bank of the International Maize and Wheat Improvement Center (CIMMYT). It also provides a performance evaluation for high phenotypic individuals based on mean normalized discounted cumulative gain value (MNV) (Blondel, et al., 2015).

3.1 Train DeepGS model

```
1. # load example data
2. data(wheat_example)
3. Markers <- wheat_example$Markers
4. y <- wheat_example$y
5. cvSampleList <- cvSampleIndex(length(y),10,1)
6. # cross validation set
7. cvIdx <- 1
8. trainIdx <- cvSampleList[[cvIdx]]$trainIdx
9. testIdx <- cvSampleList[[cvIdx]]$testIdx
10. trainMat <- Markers[trainIdx,]
11. trainPheno <- y[trainIdx]
12. validIdx <- sample(1:length(trainIdx),floor(length(trainIdx)*0.1))
13. validMat <- trainMat[validIdx,]
14. validPheno <- trainPheno[validIdx]
15. trainMat <- trainMat[-validIdx,]
16. trainPheno <- trainPheno[-validIdx]
17. conv_kernel <- c("1*18") ## convolution kernels (fileter shape)
18. conv_stride <- c("1*1")
19. conv_num_filter <- c(8) ## number of filters
20. pool_act_type <- c("relu") ## active function for next pool
21. pool_type <- c("max") ## max pooling shape
22. pool_kernel <- c("1*4") ## pooling shape
23. pool_stride <- c("1*4") ## number of pool kernerls
```

```

24. fullayer_num_hidden <- c(32,1)
25. fullayer_act_type <- c("sigmoid")
26. drop_float <- c(0.2,0.1,0.05)
27. cnnFrame <- list(conv_kernel =conv_kernel,conv_num_filter = conv_num_filter,
  conv_stride = conv_stride,pool_act_type = pool_act_type, pool_type = pool_t
  ype,pool_kernel =pool_kernel, pool_stride = pool_stride,fullayer_num_hidden
  = fullayer_num_hidden, fullayer_act_type = fullayer_act_type,drop_float = d
  rop_float)
28. markerImage = paste0("1*",ncol(trainMat))
29. trainGSmodel <- train_deepGSModel(trainMat = trainMat,trainPheno = trainPhen
  o, validMat = validMat,validPheno = validPheno, markerImage = markerImage,
  cnnFrame = cnnFrame,device_type = "cpu",gpuNum = 1, eval_metric = "mae", n
  um_round = 6000,array_batch_size= 30,learning_rate = 0.01, momentum = 0.5,w
  d = 0.00001, randomseeds = 0,initializer_idx = 0.01)

```

Note: the parameter `device_type = "cpu"` can be modified to `device_type = "gpu"` if DeepGS-GPU image (see **Section 1.2** for details) has been installed to gain a better computation efficiency.

3.2 Phenotype prediction based on trained GS model

```

1. predscores <- predict_GSModel(GSModel = trainGSmodel,testMat = Markers[testI
  dx,], markerImage = markerImage )

```

3.3 GS prediction performance evaluation

```

1. meanNDCG(realScores, predScores, topAlpha = c(10))

```

4. Ensemble GS model training

DeepGS provides an ensemble learning approach based on particle swarm optimization (ELBPSO) (Kennedy and Eberhart, 1995) for linearly combining the predictions of different GS models.

```

1. test_datapath <- system.file("exdata", "test_ELBPSO.RData",

```

```

2. package = "DeepGS")
3. load(test_datapath)
4. weight <- ELBPSO(rep_times = 100, interation_times = 25, weight_dimension = 2,
5. weight_min = 0, weight_max=1, rate_min = -0.01, rate_max = 0.01,
6. paticle_number = 10, pred_matrix = train_predMat, IW = 1,
7. AF1 = 2, AF2 = 2)
8. new_pre <- (test_predMat %*% weight)/sum(weight)

```

5. Source codes availability

The source codes of DeepGS are freely available at

<https://github.com/cma2015/DeepGS>

6. How to access help

- If users encounter any bugs or issues, feel free to leave a message at Github issues: <https://github.com/cma2015/DeepGS/issues>. We will try our best to deal with all issues as soon as possible.
- In addition, if any suggestions are available, feel free to contact: mawenlong_nwsuaf@163.com or chuangma2006@gmail.com

7. References

- Blondel M, Onogi A, Iwata H, Ueda N (2015) A ranking approach to genomic selection. PLoS One 10 (6):e0128570. doi:10.1371/journal.pone.0128570
- Kennedy J, Eberhart R (1995) Particle swarm optimization. ICNN 4:1942-1948. doi:10.1109/icnn.1995.488968