

# Comparing genomic prediction methods on wheat data

*Xiang Zhu*

*September 11, 2015*

This vignette is designed to illustrate the application of **dscr** package in genomic prediction.

We will assess four methods for genotype-based prediction on a public wheat dataset. This dataset is available in **BLR** package. We will compare four methods: Bayes Lasso (from **BLR** package), BSLMM (from **GEMMA** software), rrBLUP and rrBLUP with a Gaussian kernel (from **rrBLUP** package).

## Quick start

1. Download the [repository](#) from GitHub.

```
wget https://github.com/xiangzhu/dscr_blm/archive/master.zip
unzip master.zip
rm master.zip
```

After this step, there is a local `dscr_blm` folder.

```
[xiangzhu@office] ls -l
drwxrwxr-x 3 xiangzhu xiangzhu 4096 Sep 10 17:39 datamakers
drwxrwxr-x 2 xiangzhu xiangzhu 4096 Sep 10 17:37 methods
-rw-rw-r-- 1 xiangzhu xiangzhu 505 Sep 11 14:29 methods.R
-rw-rw-r-- 1 xiangzhu xiangzhu 4798 Sep 10 13:47 README.md
-rw-rw-r-- 1 xiangzhu xiangzhu 151 Sep 11 10:38 run_dsc.R
-rw-rw-r-- 1 xiangzhu xiangzhu 160 Sep 11 14:30 scenarios.R
-rw-rw-r-- 1 xiangzhu xiangzhu 549 Sep 11 13:57 score.R
-rw-rw-r-- 1 xiangzhu xiangzhu 177728 Sep 11 15:16 vignette.pdf
-rw-rw-r-- 1 xiangzhu xiangzhu 6503 Sep 11 15:17 vignette.Rmd
```

2. Install the required R packages.

```
# for dsc
install.packages("devtools")
devtools::install_github("stephens999/ashr") # required by one vignette of dscr
devtools::install_github("stephens999/dscr", build_vignettes=TRUE)

# for genomic prediction
install.packages("BLR")
install.packages("rrBLUP")
install.packages("BGLR")
```

3. Go to the local `dscr_blm` directory. Start an R console and run `run_dsc.R`.

```
source('run_dsc.R')
```

After this step, there is a new folder `dsc_blm` containing all results of the comparisons.

```
[xiangzhu@office] ls -l
drwxrwxr-x 3 xiangzhu xiangzhu 4096 Sep 10 17:39 datamakers
drwxrwxr-x 6 xiangzhu xiangzhu 4096 Sep 11 14:31 dsc_blm
drwxrwxr-x 2 xiangzhu xiangzhu 4096 Sep 10 17:37 methods
-rw-rw-r-- 1 xiangzhu xiangzhu 505 Sep 11 14:29 methods.R
-rw-rw-r-- 1 xiangzhu xiangzhu 4798 Sep 10 13:47 README.md
-rw-rw-r-- 1 xiangzhu xiangzhu 151 Sep 11 10:38 run_dsc.R
-rw-rw-r-- 1 xiangzhu xiangzhu 160 Sep 11 14:30 scenarios.R
-rw-rw-r-- 1 xiangzhu xiangzhu 549 Sep 11 13:57 score.R
-rw-rw-r-- 1 xiangzhu xiangzhu 177728 Sep 11 15:16 vignette.pdf
-rw-rw-r-- 1 xiangzhu xiangzhu 6503 Sep 11 15:17 vignette.Rmd
```

## Step-by-step illustration

If `dscr` package is used for the first time, the following installation is necessary.

First load the library, and initialize a new `dsc` using `new_dsc`. This object will be used to store the details of the `dsc`.

```
library(dscr)
dsc_blm = new_dsc("genomic_prediction","dsc_blm")
```

Now define the function `datamaker` to create data, which consists of `input` and `meta` data. The `input` is what the methods will be given. The `meta` data will be used when scoring the methods. Here, the `input` is a list with three elements,

- `input$data.name`: the name of data set (character string)
- `input$phenotype.id`: the identifier of phenotype for multiple-phenotype dataset (integer)
- `input$test.subject`: the identifiers of individuals that are assigned to the test set (integer vector)

and the `meta` is a list with one element, `meta$true.value`, the observed phenotypes in the test set. The `datamaker` function takes a single parameter `args`, which is a list with three elements: `args$data.name`, `args$phenotype.id` (same as `input`) and `args$test.size` (the sample size of test set, integer). The function `datamaker.R` is in folder `datamaker/`.

Now use `datamaker` to define scenarios. Each scenario is determined by the `datamaker`, its arguments and the random seeds it uses. The name `wheat` is used for naming the scenario in results and also output directories.

```
source_dir("datamakers") # need datamaker to create scenarios

# use the first kind of phenotype in wheat data
# set the sample size of test set as 100
args_eg1 = list(data.name="wheat",test.size=100,phenotype.id=1)

# repeat the random partition of data for 20 times, using seed 1-20
add_scenario(dsc_blm,name="wheat",datamaker,args=args_eg1,seed=1:20)
```

Now define the methods. They share the same form where they take `input` data and produce `output` in a specified format. Here the `output` is a list with one component, `output$predict`, the predicted phenotypes in the test set. For each method, we have to write a “wrapper” function to guarantee the input-output requirement. We also allow for additional arguments of each method (e.g. the tuning/user-specified parameters of the method). The wrapper functions (`*.wrapper.R`) of all four methods are in folder `methods/`.

Now define a list of the methods that will run in our dsc. Each method is defined by its name (`name`), the function used to implement it (`fn`), and any additional arguments (`args`).

```
source_dir("methods") # need method wrappers

add_method(dsc_blm, name="bayes_lasso", fn=bayeslasso.wrapper, args=list(nIter=1.1e4, burnIn=1e3))
add_method(dsc_blm, name="gemma_bslmm", fn=gemma.bslmm.wrapper, args=list(w=1.1e4, s=1e3))
add_method(dsc_blm, name="rrblup", fn=rrblup.wrapper, args=list())
add_method(dsc_blm, name="rrblup_kernel", fn=rrblup.gkernel.wrapper, args=list())
```

To assess the prediction performance of each method, we define a score function; see `score.R`. The score function uses `data` (from `datamaker.R`) and `output` (from `*.wrapper.R`) to compute a variety of prediction quality metrics, and save them in a list `score`. Currently, there are four metrics used in our score function:

- `score$mse`: mean square error between predicted and observed phenotypes in the test set
- `score$rmse`: root mean square error between predicted and observed phenotypes in the test set
- `score$pcor`: Pearson correlation between predicted and observed phenotypes in the test set
- `score$slope`: slope of simple regression of observed on predicted phenotypes in the test set

We have completed defining `datamaker`, `methods` and `score` for our dsc. Now we will run all methods on all the scenarios.

```
res=run_dsc(dsc_blm)
```

This returns a data frame `res` with the results of running all the methods on all scenarios.

```
> head(res[sample(1:80, replace = FALSE), 1:8])
```

	.id	seed	scenario	method	mse	rmse	pcor	slope
49	wheat	9	wheat	rrblup	0.7254980	0.8517617	0.5583381	1.1970995
42	wheat	2	wheat	rrblup	0.8531229	0.9236465	0.5631131	1.2921754
34	wheat	14	wheat	gemma_bslmm	0.6087290	0.7802109	0.5353176	0.8697284
4	wheat	4	wheat	bayes_lasso	0.7785023	0.8823278	0.5645790	1.1769306
74	wheat	14	wheat	rrblup_kernel	0.8097204	0.8998447	0.6310907	0.9797745
56	wheat	16	wheat	rrblup	0.5883878	0.7670644	0.6604027	1.3740188

Now we can summarize the comparison results. For example, look at the median `rmse` and `pcor`.

```
> aggregate(rmse~method+scenario, res, median)
```

	method	scenario	rmse
1	bayes_lasso	wheat	0.8770236
2	gemma_bslmm	wheat	0.8881155
3	rrblup	wheat	0.8716839
4	rrblup_kernel	wheat	0.9682903

```
> aggregate(pcor~method+scenario, res, median)
```

	method	scenario	pcor
1	bayes_lasso	wheat	0.5267583
2	gemma_bslmm	wheat	0.5021476
3	rrblup	wheat	0.5337660
4	rrblup_kernel	wheat	0.5947725

## Update the comparison

We call this framework “dynamic comparison” because one can add new data, new methods and new score metrics to the existing comparisons.

### Add a dataset

In my current implementation, this part is not automatic. For each dataset, I first manually pre-process it such that the wrapper functions can use the data conveniently. For example, in the directory `/datamakers/wheat`, there are two R scripts to prepare the wheat data for comparison.

```
-rw-rw-r-- 1 xiangzhu xiangzhu   935 Sep 10 13:47 dataprep_gemma.R
-rw-rw-r-- 1 xiangzhu xiangzhu   356 Sep 10 13:47 dataprep_rtool.R
```

The function `dataprep_gemma.R` generates `*.txt` and `*.txt.gz` that are used by `gemma.bslmm.wrapper.R`. The function `dataprep_rtool.R` generates `*.RData` that are used by methods implemented as R packages. This one-time step requires manual scripting, so it might be sub-optimal when there are many datasets for comparison. After this step, the dataset is added and can be used directly in the future.

### Add a method

Write a new method wrapper, save it in the folder `methods`, add the new method to `dsc_blm`, and run `dsc`. For example, I add a G-BLUP method (available in [BGLR](#) package) in our comparison.

```
source_dir("methods")

library(BGLR)
add_method(dsc_blm, name="g_blup", fn=gblup.wrapper, args=list(nIter=1.1e4, burnIn=1e3))

res=run_dsc(dsc_blm)
```

Now the new method and its results will show up in the data frame `res`.

```
> aggregate(rmse~method+scenario, res, median)
  method scenario    rmse
1 bayes_lasso   wheat 0.8770236
2    g_blup   wheat 0.8600101
3 gemma_bslmm   wheat 0.8881155
4    rrblup   wheat 0.8716839
5 rrblup_kernel   wheat 0.9682903
```

### Add a score

Simply modify the script `score.R` by adding a new score metric as an element of the list.