國立臺灣大學生物資源暨農學院農藝學研究所

碩士論文

Graduate Institute of Agronomy

College of Bioresources and Agriculture

National Taiwan University

Master Thesis

基因體選拔中兩種訓練集最佳化準則之比較

A Comparison of two criteria for training set optimization in genomic selection

廖柏鈞

Po-Chun Liao

指導教授: 廖振鐸 博士

Advisor: Chen-Tuo Liao Ph.D.

中華民國 110 年 7 月

July, 2021

國立臺灣大學碩士學位論文

口試委員會審定書

基因體選拔中兩種訓練集最佳化準則之比較

A Comparison of two criteria for training set optimization in genomic selection

本論文係廖柏鈞君（R07621207）在國立臺灣大學農藝學研究所生統組完成之碩士學位論文，於民國 110 年 7 月 15 日承下列考試委員審查通過及口試及格，特此證明

口試委員：　高振宏博士　高振宏

蔡欣甫博士　蔡欣甫

廖振鐸博士　廖振鐸

（指導教授）

Acknowledgements

這篇論文之所以能順利完成，最要感謝的是我的指導老師廖振鐸教授，感謝老師這兩年來悉心指導，無論是學術上的關懷，還是課業上的指導，都讓大學時期非農藝背景的我也能慢慢熟悉並了解統計學於生物學上的應用，在研究態度與方法上獲益良多，並給予我的論文細心指正。轉眼之間，兩年的研究所生活已經邁入了尾聲，感謝研究所時期所有系上生統組老師們孜孜不倦地教導及提攜及研究所同儕與實驗室學長和學妹們生活上的照顧，在此向大家致上誠摯的感激與謝意。

我還要另外感謝在新冠肺炎疫情嚴峻期間願意以線上會議的方式撥冗參加我論文口試的口委們，高振宏老師、蔡欣甫老師與廖振鐸老師，感謝教授們的建議，促使本篇論文能更臻完備，未來我將以我系上所學貢獻於職場上。此外，我更要感謝我的家人，因為有你們一路上的支持及鼓勵，才讓我得以在人生的旅途中努力向目標前進。謹以此向所有鼓勵過我的人致上深深的謝意，並將這篇論文的成果獻給你們，若研究所兩年生活中沒有你們的支持，這篇論文無法順利完成。

摘要

雖然次世代定序 (Next Generation Sequencing) 技術目前可協助降低基因型獲得 (genotyping) 的成本，但表現型獲得 (phenotyping) 於育種領域的執行成本上仍是一大考驗。因此，基因體選拔 (genomic selection) 可藉由篩選出使預測準確率最大化的特定訓練集 (training set) 資料來降低該訓練集於表現型獲得所需的成本並建立預測模型。在基因體選拔的過程中，較佳的訓練集能協助我們建立預測測試集數量性狀較為精準的模型。而從候選集 (candidate set) 篩選對應每組測試集 (testing set) 的最佳訓練集過程中，本論文各採用以 r-score 和 mspe-score 作為目標函數 (objective function) 的基因演算法 (genomic algorithms, GA) 來求之。透過基因演算法選出的訓練集在表現型獲得後，將可用來估計測試集個體的育種價 (genomic estimated breeding values, GEBVs)。基因演算法中採用的目標函數 r-score 和 mspe-score 可分別由測試集的表現型值與育種價間的皮爾森相關係數 (Pearson's correlation) 與均方預測誤差 (mean squared prediction error) 推導而得。此外，本論文以 Tropical rice 和 rice44k 兩組資料作為範例，並採用一般常見的皮爾森相關係數及均方根誤差來評估預測模型的準確度；其中，由於 rice44k 資料的水稻個體共含六種次族群 (subpopulations) 結構，在建模過程除了比較測試集已知及未知外，還需考量次族群的影響。

關鍵字：基因體選拔、基因演算法、全基因組迴歸模式、線性混合模型、限制最大概似估值

Abstract

While genotyping has become more cost-effective due to next-generation sequencing technique, the cost of phenotyping is still an obstacle in plant breeding. Therefore, the determination of a training set plays an important role to the success of a genomic selection (GS) program. An appropriate training set can be employed to reduce the phenotyping cost and maximize the prediction accuracy of the breeding program simultaneously. In this study, two optimality criteria derived from Pearson's correlation and the mean squared prediction error between the phenotypic values and their genomic estimated breeding values (GEBVs) of a testing set are proposed for the training set optimization. A genetic algorithm implementing the two optimality criteria is used to generate the desired training set. The chosen optimal training set is phenotyped, and the resulting phenotypic values together with the genotypic values are used to build a GS prediction model, which is then applied to estimate the GEBVs of the testing set. Pearson's correlation and root-mean-square error (RMSE) are further used as the measures to compare the performance between the two optimality criteria. Real data analysis and simulation studies based on two rice genome datasets are carried out, and the results show that the two optimality criteria have almost the same performance.

Keywords: genomic selection; genetic algorithm; whole-genome regression; linear mixed effects model; restricted maximum likelihood estimate.

Contents

List of Figures

List of Tables

Chapter 1 INTRODUCTION

Unlike conventional plant breeding which relies on phenotypes, the concept of GS is to improve quantitative traits through genome-wide SNP markers (Meuwissen et al. (2001)). Therefore, the composition of the training set, e.g., the heritability, the number of SNP markers and the sample size, should have a significant impact on the prediction accuracy (Heffner et al. (2009); Lorenz et al. (2012); Wimmer et al. (2013)). Moreover, the genomic relationship between the genotypic values of training and testing sets also plays a key role in constructing an accurate GS prediction model. According to Akdemir et al. (2015), optimizing the training set provides an economical and efficient way to the success of a GS breeding program. Whole-genome regression (WGR) and linear mixed effects model (LMM) are two commonly used GS prediction models. We first review some preliminaries as follows.

1.1 Whole genome regression

Suppose there are n individuals and p SNPs involving the experiment, and the additive effects model can be cast as:

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + e_i, \quad e_i \overset{i.i.d.}{\sim} N(0, \sigma_e^2)$$

where y_i denotes the phenotypes of the i-th individual; β_0 the constant term; $\boldsymbol{x}_{ij}$ the marker score of the i-th individual at the j-th locus, coded as -1 for M_1M_1, 0 for M_1M_2 and 1 for M_2M_2; β_j the marker effect of the j-th locus and e_i the error term of the i-th individual.

Different from ordinary least squares regression (OLS), WGR includes ridge, LASSO and elastic net regressions (Xavier et al.(2016)) which allow tackling 'large-p-with-small-n' problems with multicollinearity by adding penalized shrinkage estimators or adopting Bayesian estimation such as Bayes A, Bayes B (Meuwissen et al. (2001)) and Bayes C (Shariati et al. (2012)) with Gibbs sampling.

Among the shrinkage methods, ridge regression includes an L2 penalty, which is a popular type of regularized linear regression. By minimizing the loss function of $\boldsymbol{e}^T\boldsymbol{e} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$, we can derive the solution of $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e}$ for ridge regression:

$$\hat{\boldsymbol{\beta}}_{ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_p)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

1.2 Linear mixed effects model

Linear mixed effects model treats additive marker effects as random effects and can be cast as:

$$\boldsymbol{y} = \boldsymbol{1}_n\mu + \boldsymbol{Z}\boldsymbol{g} + \boldsymbol{e}$$

where $\boldsymbol{y}$ is the vector of phenotypes, $\boldsymbol{\mu}$ the vector of overall mean considered as a fixed effect, whereas $\boldsymbol{g}$ and $\boldsymbol{e}$ the vectors of random additive and residual effects. $\boldsymbol{Z}$ is the incident matrix of vector $\boldsymbol{g}$. In LMM studies, two statistical methods are commonly used for predicting the genomic estimated breeding values (GEBVs) inclusive of ridge regression best linear unbiased predictor (rrBLUP) and genomic best linear unbiased predictor (GBLUP).

For the rrBLUP method, $\boldsymbol{Z}$ is referred to as a design matrix of dimensions $n \times p$ for SNP effects and the vector of random additive effects $\boldsymbol{g}$ and random residual effects $\boldsymbol{e}$ are assumed to follow normal distributions $\boldsymbol{g} \sim N(0, \boldsymbol{I}_n\sigma_m^2)$ and $\boldsymbol{e} \sim N(0, \boldsymbol{I}_n\sigma_e^2)$, where σ_m^2 is the proportion of the genetic variance

given by each SNP and σ_e^2 is the residual variance. Hence, we can write the rrBLUP model as:

$$\boldsymbol{y} = \mathbf{1}_n\mu + \boldsymbol{Z}\boldsymbol{g} + \boldsymbol{e}$$

where $\boldsymbol{y}$ denotes the phenotypic values vector, $\boldsymbol{Z}$ the design matrix for SNP effects, the vector of random additive genetic effects $\boldsymbol{g} \sim N(0, \boldsymbol{I}_n\sigma_m^2)$ and the vector of random residuals $\boldsymbol{e} \sim N(0, \boldsymbol{I}_n\sigma_e^2)$.

As for the GBLUP method, the vector of random additive effects and residual effects are assumed to follow normal distributions $\boldsymbol{g} \sim N(0, \boldsymbol{K}\sigma_a^2)$ and $\boldsymbol{e} \sim N(0, \boldsymbol{I}_n\sigma_e^2)$, where σ_a^2 and σ_e^2 denote additive genetic variance and residual variance, $\boldsymbol{K}_{n\times n}$ is the genomic relationship matrix or the kinship matrix and $\boldsymbol{Z}$ is defined as $\boldsymbol{I}_n$ so that the GBLUP model can be written as:

$$\begin{aligned}\boldsymbol{y} &= \mathbf{1}_n\mu + \boldsymbol{I}_n\boldsymbol{g} + \boldsymbol{e} \\ &= \mathbf{1}_n\mu + \boldsymbol{g} + \boldsymbol{e}\end{aligned}$$

where $\boldsymbol{y}$ denotes the vector of phenotypic values, the vector of random additive genetic effects $\boldsymbol{g} \sim N(0, \boldsymbol{K}\sigma_a^2)$ and the vector of random residuals $\boldsymbol{e} \sim N(0, \boldsymbol{I}_n\sigma_e^2)$. In addition, the genomic relationship matrix $\boldsymbol{K}_{n\times n}$ is defined by the PC-based matrix $\boldsymbol{P}_{n\times n}$. The i-th column of $\boldsymbol{P}_{n\times n}$ denotes the i-th PC score of the scaled genotypic matrix $\boldsymbol{X}_{n\times p}$ and thus $\boldsymbol{K}_{n\times n} = \frac{1}{p}\boldsymbol{P}\boldsymbol{P}^T$. The PC-based matrix $\boldsymbol{P}_{n\times n}$ can also help saving computational cost during the process of calculating fitness values in genetic algorithms. In this thesis, we adopt the GBLUP model for predicting GEBVs.

Note that the shrinkage parameter λ is also required to regularize regression coefficients. This shrinkage parameter can be defined as $\lambda = \frac{\sigma_e^2}{\sigma_a^2}$ so that the effect of regularization shows direct proportion to σ_e^2 and inverse proportion to σ_a^2. The heritability, a statistic measuring the ratio of phenotypic difference due to genetic variation, can be expressed as the formulation of $h^2 = \frac{\sigma_a^2}{\sigma_a^2+\sigma_e^2} = (1+\lambda)^{-1}$.

Facilitating the solution of the GBLUP model gives the simplified mixed model equation (Hen-

derson et al. (1950)) formulated as $\boldsymbol{Cb} = \boldsymbol{r}$ as follows.

$$\boldsymbol{Cb} = \begin{bmatrix} n & \mathbf{1}_n^T \\ \mathbf{1}_n & \boldsymbol{I}_n + \frac{\sigma_e^2}{\sigma_a^2}\boldsymbol{K}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\mu} \\ \hat{\boldsymbol{g}} \end{bmatrix} = \begin{bmatrix} \mathbf{1}_n^T \boldsymbol{y} \\ \boldsymbol{y} \end{bmatrix} = \boldsymbol{r}$$

$$\boldsymbol{b} = \begin{bmatrix} \hat{\mu} \\ \hat{\boldsymbol{g}} \end{bmatrix} = \boldsymbol{C}^{-1}\boldsymbol{r} = \begin{bmatrix} n & \mathbf{1}_n^T \\ \mathbf{1}_n & \boldsymbol{I}_n + \frac{\sigma_e^2}{\sigma_a^2}\boldsymbol{K}^{-1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{1}_n^T \boldsymbol{y} \\ \boldsymbol{y} \end{bmatrix}$$

However, σ_a^2 and σ_e^2 are still unknown. Therefore, to obtain the two variance components, one can estimate by maximizing the restricted marginal likelihood through expectation maximization (EM), an algorithm to iteratively update estimates until they converge (Dempster et al.(1977)).

The restricted marginal likelihood can be cast as:

$$L(\mu, \sigma_a^2, \sigma_e^2) = -\frac{1}{2}|\boldsymbol{V}| - \frac{1}{2}|\mathbf{1}_n^T \boldsymbol{V}^{-1}\mathbf{1}_n| - \frac{1}{2}(\boldsymbol{y} - \mathbf{1}_n\mu)^T \boldsymbol{V}^{-1}(\boldsymbol{y} - \mathbf{1}_n\mu)$$

where

$$\boldsymbol{V} = \sigma_a^2 \boldsymbol{K} + \sigma_e^2 \boldsymbol{I}_n$$

Baysian Gibbs sampling (BGS) is another approach to obtain estimates of variance components (Stuart German and Donald German (1984)). It updates each estimate sampled from the posterior distribution every iteration and the estimates are stored in each cycle, discarded prior to the stationary state (burn-in) and averaged out. The posterior distribution of coefficients follows a normal distribution and that of variance components follows a scaled inverse Chi squared distribution $\chi_{\nu,S}^{-2}$, where ν denotes the degrees of freedom of the variance components which is set as 5 and S the priors' shape which is set as $0.5 \times var(\boldsymbol{y})$ (Morota et al.(2014)). We decide to choose the GBLUP method with Gibbs sampler running for 6,000 iterations with a burn-in of 1,000 iterations and a 5-iteration thin-

ning intervals and implement an RKHS regression, a convenient method by providing the eigenvalue decomposition of $\boldsymbol{K}$, to build a GS model with R package BGLR in this thesis (Pérez et al. (2014)).

The BGS algorithm is as follows.

1. Propose the initial values for $\boldsymbol{b}$, σ_e^2 and σ_a^2.
2. Sample each coefficient b_i from $N(mean = (r_i - \boldsymbol{C}_{i,-i}\boldsymbol{b}_{-i})\boldsymbol{C}_{ii}^{-1}, variance = \sigma_e^2\boldsymbol{C}_{ii}^{-1})$
3. Compute residuals $\boldsymbol{e} = \boldsymbol{y} - [\boldsymbol{1}_n, \boldsymbol{Z}]\boldsymbol{b}$
4. Update the residual variance σ_e^2 as $\frac{\boldsymbol{e}^T\boldsymbol{e}+S\nu}{\chi^2_{n+\nu}}$ and the random variance σ_a^2 as $\frac{\boldsymbol{g}^T\boldsymbol{K}^{-1}\boldsymbol{g}+S\nu}{\chi^2_{p+\nu}}$
5. Update $\boldsymbol{C}$ with the new value of $\lambda = \frac{\sigma_e^2}{\sigma_a^2}$
6. Repeat step 2 to 5 for a fixed number of iterations, discard the cycles prior to entropy and average the values for $\hat{\boldsymbol{g}}$, $\hat{\sigma}_e^2$ and $\hat{\sigma}_a^2$ at regular intervals from iterations.

When the $\hat{\mu}$ and $\hat{\boldsymbol{g}}$ estimates are given through Henderson's mixed model equation, the estimated genotypic values for the testing set can be written as:

$$\hat{\boldsymbol{g}}_0 = \boldsymbol{K}_{01}K^{-1}\hat{\boldsymbol{g}}$$

where $\hat{\boldsymbol{g}}_0$ and $\hat{\boldsymbol{g}}$ denote the estimated genotypic values of testing and training sets, $\boldsymbol{K}_{01}$ the genomic relationship matrix between testing and training sets and $\boldsymbol{K}$ the genomic relationship matrix among the training set. Therefore, the GEBVs for the testing set can be predicted as:

$$\hat{\boldsymbol{y}}_0 = \boldsymbol{1}_n\hat{\mu} + \boldsymbol{K}_{01}\boldsymbol{K}^{-1}\hat{\boldsymbol{g}}$$

Chapter 2 Methods

Because phenotyping is the limiting factor in genomic prediction, we should take the case where all the individuals are genotyped but only a proportion will be phenotyped into account. To determine the optimal training set from a candidate set, the r-score criteria proposed by Ou et al. (2019) has been verified to be better than CD and PEV^{Ridge} which are suggested by Rincent et al.(2012) and Akdemir et al.(2015) respectively because the derivation of r-score considers both the variance and bias while PEV^{Ridge} only takes the prediction error variance into account. As a result, we decide to make a comparison between the prediction accuracy obtained by r-score and the new criteria, mspe-score.

Two optimality criteria, r-score and mspe-score, are derived from Pearson's correlation and mean squared prediction error to describe genetic relationship between training and testing sets. To compare the prediction accuracy of GS models, we apply the Pearson's correlation and the root mean square error between phenotypic values and GEBVs of the testing set as performance measures for our GS models. The following introduces how r-score and mspe-score are derived.

In ridge regression, $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_p)^{-1}\boldsymbol{X}^T\boldsymbol{y}$ can be written equivalently as $\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{y}$, which is derived from the identity (Searle et al.(1982),p.261)

$$(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{F}^{-1}\boldsymbol{B})^{-1} = \boldsymbol{D}^{-1} - \boldsymbol{D}^{-1}\boldsymbol{C}(\boldsymbol{F} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}$$

The detailed derivation of the identity is as follows.

$$
\begin{aligned}
(\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I}_p)^{-1}\boldsymbol{X}^T\boldsymbol{y} &= (\lambda\boldsymbol{I}_p+\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \\
&= \{(\lambda\boldsymbol{I}_p)^{-1}-(\lambda\boldsymbol{I}_p)^{-1}\boldsymbol{X}^T[\boldsymbol{I}_n+\boldsymbol{X}(\lambda\boldsymbol{I}_p)^{-1}\boldsymbol{X}^T]^{-1}\boldsymbol{X}(\lambda\boldsymbol{I}_p)^{-1}\}\boldsymbol{X}^T\boldsymbol{y} \\
&= [\frac{1}{\lambda}\boldsymbol{X}^T-\frac{1}{\lambda}\boldsymbol{X}^T(\boldsymbol{I}_n+\frac{1}{\lambda}\boldsymbol{X}\boldsymbol{X}^T)^{-1}(\frac{1}{\lambda}\boldsymbol{X}\boldsymbol{X}^T)]\boldsymbol{y} \\
&= [\frac{1}{\lambda}\boldsymbol{X}^T-\frac{1}{\lambda}\boldsymbol{X}^T(\boldsymbol{I}_n+\frac{1}{\lambda}\boldsymbol{X}\boldsymbol{X}^T)^{-1}(\frac{1}{\lambda}\boldsymbol{X}\boldsymbol{X}^T+\boldsymbol{I}_n-\boldsymbol{I}_n)]\boldsymbol{y} \\
&= [\frac{1}{\lambda}\boldsymbol{X}^T-\frac{1}{\lambda}\boldsymbol{X}^T+\frac{1}{\lambda}\boldsymbol{X}^T(\boldsymbol{I}_n+\frac{1}{\lambda}\boldsymbol{X}\boldsymbol{X}^T)^{-1}]\boldsymbol{y} \\
&= \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T+\lambda\boldsymbol{I}_n)^{-1}\boldsymbol{y}
\end{aligned}
$$

Let $\hat{\boldsymbol{y}}_0$, $\boldsymbol{y}_0$ and $\boldsymbol{X}_0$ denote GEBVs, phenotypic values and genetic matrix of the testing set individually while $\boldsymbol{y}$ and $\boldsymbol{X}$ the phenotypic values and genetic matrix of the training set. Hence, $\boldsymbol{y}_0$ can be denoted exactly by $\boldsymbol{y}_0=\boldsymbol{X}_0\boldsymbol{\beta}+\boldsymbol{\epsilon}_0$, where $\boldsymbol{\epsilon}_0\sim N(0,\boldsymbol{I}_{n_0}\sigma_e^2)$, n_0 is the sample size of testing set and $\boldsymbol{\beta}$ is an unknown parameter vector so that $Var(\boldsymbol{y}_0)=\boldsymbol{I}_{n_0}\sigma_e^2$ and $Var(\bar{\boldsymbol{y}}_0)=\bar{\boldsymbol{J}}_{n_0}\sigma_e^2$.

Besides, $\hat{\boldsymbol{y}}_0$ can be estimated as $\hat{\boldsymbol{y}}_0=\boldsymbol{X}_0\hat{\boldsymbol{\beta}}=\boldsymbol{X}_0\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T+\lambda\boldsymbol{I}_n)^{-1}\boldsymbol{y}=\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{y}$ where $\boldsymbol{A}=\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T+\lambda\boldsymbol{I}_n)^{-1}$ in ridge regression so that $Var(\hat{\boldsymbol{y}}_0)=\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}_0^T\sigma_e^2$ and $Var(\bar{\hat{\boldsymbol{y}}}_0)=\bar{\boldsymbol{J}}_{n_0}\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}_0^T\bar{\boldsymbol{J}}_{n_0}\sigma_e^2$. Hence, we apply the knowledge above to derive r-score and mspe-score.

2.1 r-score

Define the Pearson's correlation between $\boldsymbol{y}_0$ and $\hat{\boldsymbol{y}}_0$ as

$$
r=\frac{\sum_{i=1}^{n_0}(y_{0i}-\bar{y}_0)(\hat{y}_{0i}-\bar{\hat{y}}_0)}{\sqrt{\sum_{i=1}^{n_0}(y_{0i}-\bar{y}_0)^2\sum_{i=1}^{n_0}(\hat{y}_{0i}-\bar{\hat{y}}_0)^2}} \quad ,
$$

where y_{0i} and $\hat{y}_{0i}$ are the i-th element of $\boldsymbol{y}_0$ and $\hat{\boldsymbol{y}}_0$ while $\bar{y}_0$ and $\bar{\hat{y}}_0$ the i-th sample mean of $\boldsymbol{y}_0$ and $\hat{\boldsymbol{y}}_0$.

Let $\boldsymbol{w}_1 = \boldsymbol{y}_0 - \bar{\boldsymbol{y}}_0 \mathbf{1}_{n_0}$ and $\boldsymbol{w}_2 = \hat{\boldsymbol{y}}_0 - \bar{\hat{\boldsymbol{y}}}_0 \mathbf{1}_{n_0}$ so that r can be written as

$$r = \frac{\boldsymbol{w}_1^T \boldsymbol{w}_2}{\sqrt{(\boldsymbol{w}_1^T \boldsymbol{w}_1)(\boldsymbol{w}_2^T \boldsymbol{w}_2)}}$$

Then, we substitute surrogate expectation $E(\tilde{r})$ for $E(r)$ by calculating Pearson's correlation in which $\boldsymbol{w}_1^T \boldsymbol{w}_2$, $\boldsymbol{w}_1^T \boldsymbol{w}_1$ and $\boldsymbol{w}_2^T \boldsymbol{w}_2$ are replaced with $E(\boldsymbol{w}_1^T \boldsymbol{w}_2)$, $E(\boldsymbol{w}_1^T \boldsymbol{w}_1)$ and $E(\boldsymbol{w}_2^T \boldsymbol{w}_2)$ so that $E(\tilde{r}) = \frac{E(\boldsymbol{w}_1^T \boldsymbol{w}_2)}{\sqrt{E(\boldsymbol{w}_1^T \boldsymbol{w}_1) E(\boldsymbol{w}_2^T \boldsymbol{w}_2)}}$.

To take a step further, $\boldsymbol{M}$ is a non-negative definite matrix if it can be decomposed into $\boldsymbol{R}^T \boldsymbol{R}$. Then, the quadratic form $\boldsymbol{\beta}^T \boldsymbol{M} \boldsymbol{\beta}$ is proportional to $\sum m_{ii} \beta_i^2$,where m_{ii} is the i-th diagonal element of $\boldsymbol{M}$ so that $\boldsymbol{\beta}^T \boldsymbol{M} \boldsymbol{\beta}$ is also proportional to $Tr(\boldsymbol{M}) = \sum m_{ii}$.

Hence, based on the identity of the non-negative definite matrix above and the formula (Searle et al.(1982), p.355) for vectors x and y :

$$E(\boldsymbol{x}^T \boldsymbol{y}) = Tr[Cov(\boldsymbol{x}, \boldsymbol{y})] + E(\boldsymbol{x}^T) E(\boldsymbol{y}) \quad ,$$

we will obtain:

1. $E(\boldsymbol{w}_1^T \boldsymbol{w}_1) = Tr[(\boldsymbol{I}_{n_0} + \bar{\boldsymbol{J}}_{n_0})] \sigma_e^2 + \boldsymbol{\beta}^T \boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} - \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0 \boldsymbol{\beta}$

   $\propto (n_0 + 1) + Tr[\boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} - \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0] \stackrel{\star}{=} q_{11}$

2. $E(\boldsymbol{w}_2^T \boldsymbol{w}_2) = Tr[\boldsymbol{A}^T \boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} + \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0 \boldsymbol{A}] \sigma_e^2 + \boldsymbol{\beta}^T \boldsymbol{X}^T \boldsymbol{A}^T \boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} - \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0 \boldsymbol{A} \boldsymbol{X} \boldsymbol{\beta}$

   $\propto Tr[\boldsymbol{A}^T \boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} + \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0 \boldsymbol{A}] + Tr[\boldsymbol{X}^T \boldsymbol{A}^T \boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} - \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0 \boldsymbol{A} \boldsymbol{X}] \stackrel{\star}{=} q_{22}$

3. $E(\boldsymbol{w}_1^T \boldsymbol{w}_2) = \boldsymbol{\beta}^T \boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} - \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0 \boldsymbol{A} \boldsymbol{X} \boldsymbol{\beta}$

   $\propto Tr[\boldsymbol{X}_0^T (\boldsymbol{I}_{n_0} - \bar{\boldsymbol{J}}_{n_0}) \boldsymbol{X}_0 \boldsymbol{A} \boldsymbol{X}] \stackrel{\star}{=} \quad q_{12}$

Then, we successfully derive r-score= $\frac{q_{12}}{\sqrt{q_{11} q_{22}}}$

2.2 mspe-score

Define the mean of squared prediction errors between $\boldsymbol{y}_0$ and $\hat{\boldsymbol{y}}_0$ as

$$\begin{aligned} MSPE &= \frac{1}{n_0}\sum_{i=1}^{n_0}(y_{0i}-\hat{y}_{0i})^2 \\ &= \frac{1}{n_0}(\boldsymbol{y}_0-\hat{\boldsymbol{y}}_0)^T(\boldsymbol{y}_0-\hat{\boldsymbol{y}}_0) \\ &= \frac{1}{n_0}(\boldsymbol{y}_0-\boldsymbol{X}_0\hat{\boldsymbol{\beta}})^T(\boldsymbol{y}_0-\boldsymbol{X}_0\hat{\boldsymbol{\beta}}) \\ &= \frac{1}{n_0}(\boldsymbol{y}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{y})^T(\boldsymbol{y}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{y}) \end{aligned}$$

Then, the expectation of mean squared prediction error will be given by

$$\begin{aligned} E(MSPE) &= \frac{1}{n_0}E[(\boldsymbol{y}_0-\boldsymbol{X}_0\hat{\boldsymbol{\beta}})^T(\boldsymbol{y}_0-\boldsymbol{X}_0\hat{\boldsymbol{\beta}})] \\ &= \frac{1}{n_0}\{Tr[\boldsymbol{I}_{n_0}Var(\boldsymbol{y}_0-\boldsymbol{X}_0\hat{\boldsymbol{\beta}})]+E(\boldsymbol{y}_0-\boldsymbol{X}_0\hat{\boldsymbol{\beta}})^TE(\boldsymbol{y}_0-\boldsymbol{X}_0\hat{\boldsymbol{\beta}})\} \\ &= \frac{1}{n_0}[Tr(\boldsymbol{I}_{n_0}+\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}_0^T)\sigma_e^2+\boldsymbol{\beta}^T(\boldsymbol{X}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{X})^T(\boldsymbol{X}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{X})\boldsymbol{\beta}] \end{aligned}$$

Therefore, mspe-score is defined below, which is proportional to E(MSPE)

$$\begin{aligned} mspe-score &= \frac{1}{n_0}\{Tr(\boldsymbol{I}_{n_0}+\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}_0^T)+Tr[(\boldsymbol{X}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{X})^T(\boldsymbol{X}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{X})]\} \\ &= 1+\frac{1}{n_0}\{Tr(\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}_0^T)+Tr[(\boldsymbol{X}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{X})^T(\boldsymbol{X}_0-\boldsymbol{X}_0\boldsymbol{A}\boldsymbol{X})]\} \end{aligned}$$

Note that either r-score or mspe-score requires only the genotypic information of the training set and testing set.

2.3 The choice of λ

The r-score and mspe-score depend on the value of λ, which can be obtained from phenotypes. However, the phenotypic values are assumed to be unknown during the derivation of r-score and mspe-score. To check whether the choice of λ could influence the training set determination, we randomly choose a testing set of $n_0 = 50$ and a training set of $n = 100$ from the candidate set of $n_c = 250$ when we serve the rice44k genome as our template and calculate the r-score and mspe-score at $\lambda = 0.001, 0.01, 0.1, 1, 10, 100, 1000$ with 20 repetitions. We report the ranking of resulting r-score and mspe-score for 20 repetitions on the next page. We can find that the r-scores and mspe-scores for each repetition are almost the same and their rankings are identical for all λ values. According to this property, we fix $\lambda = 1$ in the calculation of r-score and mspe-score throughout this article.

Table 2.1: The r-score and rankings of 20 repetitions at various values of λ

repetition	0.001	0.01	0.1	1	10	100	1000
12	0.7708(1)	0.7708(1)	0.7708(1)	0.7708(1)	0.7708(1)	0.7708(1)	0.7707(1)
3	0.7747(2)	0.7747(2)	0.7747(2)	0.7747(2)	0.7747(2)	0.7747(2)	0.7745(2)
13	0.7747(3)	0.7747(3)	0.7747(3)	0.7747(3)	0.7747(3)	0.7747(3)	0.7746(3)
20	0.7756(4)	0.7756(4)	0.7756(4)	0.7756(4)	0.7756(4)	0.7756(4)	0.7754(4)
5	0.7768(5)	0.7768(5)	0.7768(5)	0.7768(5)	0.7768(5)	0.7767(5)	0.7766(5)
11	0.7769(6)	0.7769(6)	0.7769(6)	0.7769(6)	0.7769(6)	0.7769(6)	0.7767(6)
2	0.7786(7)	0.7786(7)	0.7786(7)	0.7786(7)	0.7786(7)	0.7786(7)	0.7784(7)
4	0.7809(8)	0.7809(8)	0.7809(8)	0.7809(8)	0.7809(8)	0.7809(8)	0.7808(8)
17	0.7818(9)	0.7818(9)	0.7818(9)	0.7818(9)	0.7818(9)	0.7818(9)	0.7816(9)
16	0.7820(10)	0.7820(10)	0.7820(10)	0.7820(10)	0.7820(10)	0.7820(10)	0.7818(10)
10	0.7832(11)	0.7832(11)	0.7832(11)	0.7832(11)	0.7832(11)	0.7832(11)	0.7830(11)
18	0.7832(12)	0.7832(12)	0.7832(12)	0.7832(12)	0.7832(12)	0.7832(12)	0.7830(12)
6	0.7837(13)	0.7837(13)	0.7837(13)	0.7837(13)	0.7837(13)	0.7837(13)	0.7836(13)
8	0.7841(14)	0.7841(14)	0.7841(14)	0.7841(14)	0.7841(14)	0.7841(14)	0.7839(14)
1	0.7854(15)	0.7854(15)	0.7854(15)	0.7854(15)	0.7854(15)	0.7854(15)	0.7852(15)
15	0.7855(16)	0.7855(16)	0.7855(16)	0.7855(16)	0.7855(16)	0.7855(16)	0.7853(16)
7	0.7857(17)	0.7857(17)	0.7857(17)	0.7857(17)	0.7857(17)	0.7857(17)	0.7855(17)
14	0.7858(18)	0.7858(18)	0.7858(18)	0.7858(18)	0.7858(18)	0.7858(18)	0.7856(18)
19	0.7864(19)	0.7864(19)	0.7864(19)	0.7864(19)	0.7864(19)	0.7864(19)	0.7862(19)
9	0.7868(20)	0.7868(20)	0.7868(20)	0.7868(20)	0.7868(20)	0.7868(20)	0.7866(20)

Table 2.2: The mspe-score and ranking of 20 repetitions at various values of λ

repetition	0.001	0.01	0.1	1	10	100	1000
9	12814(1)	12814(1)	12814(1)	12814(1)	12814(1)	12814(1)	12827(1)
19	12835(2)	12835(2)	12835(2)	12835(2)	12835(2)	12835(2)	12846(2)
1	12862(3)	12862(3)	12862(3)	12862(3)	12862(3)	12862(3)	12874(3)
14	12868(4)	12868(4)	12868(4)	12868(4)	12868(4)	12868(4)	12879(4)
7	12873(5)	12873(5)	12873(5)	12873(5)	12873(5)	12873(5)	12885(5)
15	12900(6)	12900(6)	12900(6)	12900(6)	12900(6)	12900(6)	12913(6)
8	12939(7)	12939(7)	12939(7)	12939(7)	12939(7)	12939(7)	12951(7)
6	12969(8)	12969(8)	12969(8)	12969(8)	12969(8)	12969(8)	12979(8)
18	13003(9)	13003(9)	13003(9)	13003(9)	13003(9)	13003(9)	13015(9)
10	13020(10)	13020(10)	13020(10)	13020(10)	13020(10)	13021(10)	13032(10)
16	13068(11)	13068(11)	13068(11)	13068(11)	13068(11)	13068(11)	13079(11)
17	13083(12)	13083(12)	13083(12)	13083(12)	13083(12)	13083(12)	13093(12)
4	13121(13)	13121(13)	13121(13)	13121(13)	13121(13)	13121(13)	13132(13)
2	13246(14)	13246(14)	13246(14)	13246(14)	13246(14)	13247(14)	13256(14)
11	13339(15)	13339(15)	13339(15)	13339(15)	13339(15)	13339(15)	13350(15)
5	13340(16)	13340(16)	13340(16)	13340(16)	13340(16)	13340(16)	13351(16)
20	13405(17)	13405(17)	13405(17)	13405(17)	13405(17)	13405(17)	13415(17)
13	13456(18)	13456(18)	13456(18)	13456(18)	13456(18)	13456(18)	13466(18)
3	13463(19)	13463(19)	13463(19)	13463(19)	13463(19)	13463(19)	13473(19)
12	13661(20)	13661(20)	13661(20)	13661(20)	13661(20)	13662(20)	13672(20)

2.4 Genetic algorithm

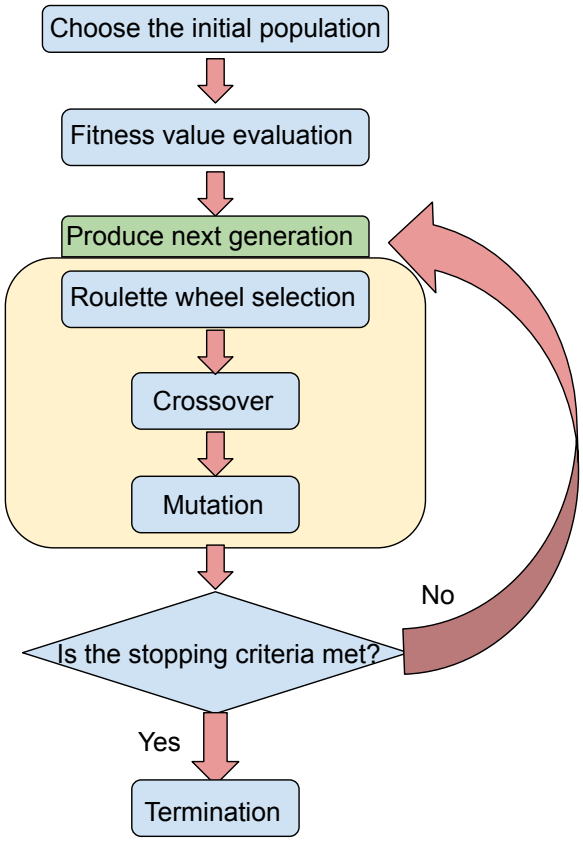

Figure 2.1: The procedure of genomic algorithms

Genetic algorithm (GA) is a search-based optimization technique based on rules of Genetics and Darwin's theory of evolution by natural selection. In this study, we implement GA to find the optimal training set that maximizes r-score or minimizes mspe-score from a candidate set. The concept of GA was first proposed by John Holland (1975), which attempts to mimic natural selection by replenishing an older population to a newer one accompanied by hopefully more advantageous fitness values (e.g., higher r-scores and lower mspe-scores) in order to solve complicated optimization problems.

The key steps of genetic algorithms are as follows.

1.Population initialization

Creating the initial population is the first step of GA. Each individual from the population has a chromosome composed of binary gene encoding, which indicates the existence of corresponding samples. In this step, we need to set several parameters including population size (Nc), length of 1 in chromosomes (n), fitness function (r-score & mspe-score), stopping criteria, crossover rate and mutation rate in advance to proceed to the next step.

2.Fitness value evaluation

The second step of GA is fitness value evaluation. This step calculates fitness values of each individual so that we can eliminate some with poorer fitness values and save hopefully more advantageous individuals which might cause an ideal target value to pass on successive generations.

3.Produce next generation

Three operators in GA, selection, crossover and mutation (Whitley (1994)), are applied to replenish an older population with a newer one prone to retain finer genes.

a.Selection operator

In a selection operator, roulette wheel selection is often used to make sure individuals tend to get higher r-scores or lower mspe-scores of having a high possibility to be selected.

b.Crossover and mutation operators

As learned from Genetics, the more genetic diversity a population has, the more likely the population is to survive and keep evolving. Therefore, crossover and mutation are two operators to maintain genetic diversity and prevent GA from being stuck at a local minimum or maximum after a few generations.

4.Determine whether the termination condition is met.

In general, we keep one of the following rules as our stopping criteria in GA.

- When there has been no improvement for numerous iterations.
- When we attain a fixed number of generations.
- When the objective function value has arrived at a pre-defined value.

GA is set to stop when there is no improvement over $(Nc - n) \times n$ iterations in this study, where Nc is the sample size of the candidate set and n is that of the training set.

5.Termination

If not, repeat step 2 to step 4 until GA terminates. Note that the fitness value of the final solution might still be a local optimum, not an extreme value.

Chapter 3 Results

Tropical rice and rice44k genome datasets are analyzed here to demonstrate our methods. The tropical rice dataset has mild population structures and the rice44k dataset has strong population structures with six categories. Here we use gBLUP model with Bayesian Gibbs sampling (BGS) combined with RKHS for the estimation of GEBVs.

In data processing, we first eliminate SNP locus with more than 10% NAs and replace the remaining NAs with 1, which is considered dominant alleles. Afterwards, we convert the marker score matrices into PC score matrices before performing the training set optimization.

Then, we determine 30 testing sets with each sample size of 50 randomly and apply genetic algorithms to choose the corresponding training sets which give the higher r-scores or lower mspe-scores by targeted and untargeted methods (Akdemir and Isidro Sanches (2019)). The targeted method considers genotypes of the testing set to be defined while the untargeted one serves the individuals in the candidates as the testing set. Afterwards, we construct a GS model to evaluate Pearson's correlation and root of mean squared prediction error (RMSE) between GEBVs and phenotypic values of each testing set. As for the case of the rice44k dataset, stratified sampling should be taken into consideration.

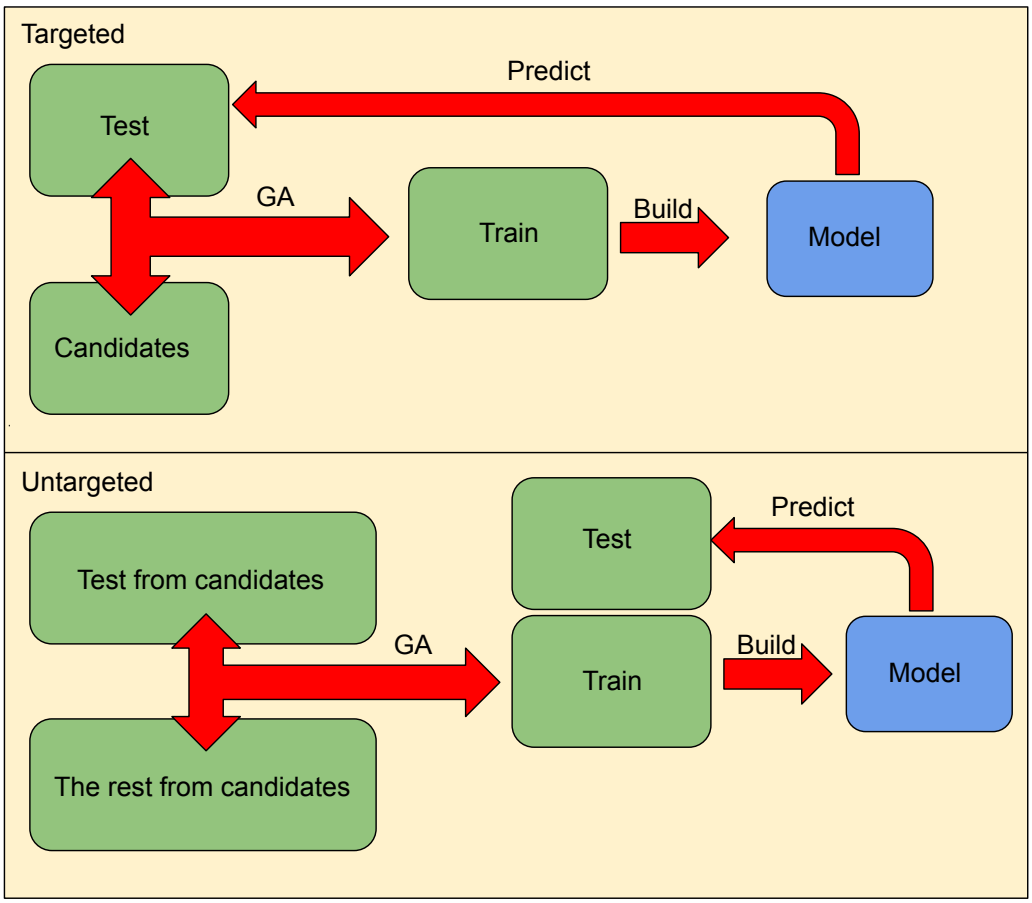

Figure 3.1: The schemes applied to GA

3.1 Real Data Analysis

3.1.1 Tropical Rice data

The tropical rice dataset is composed of 328 individuals with 73137 SNPs after data processing and each individual has three quantitative traits, yield, plant height and flowering time. The size of the testing set is set to be 50 and that of the training set selected at random or genetic algorithms is set to be 50, 100, 150 and 200 respectively. The final results are brought forth as follows.

Root Mean Square Error

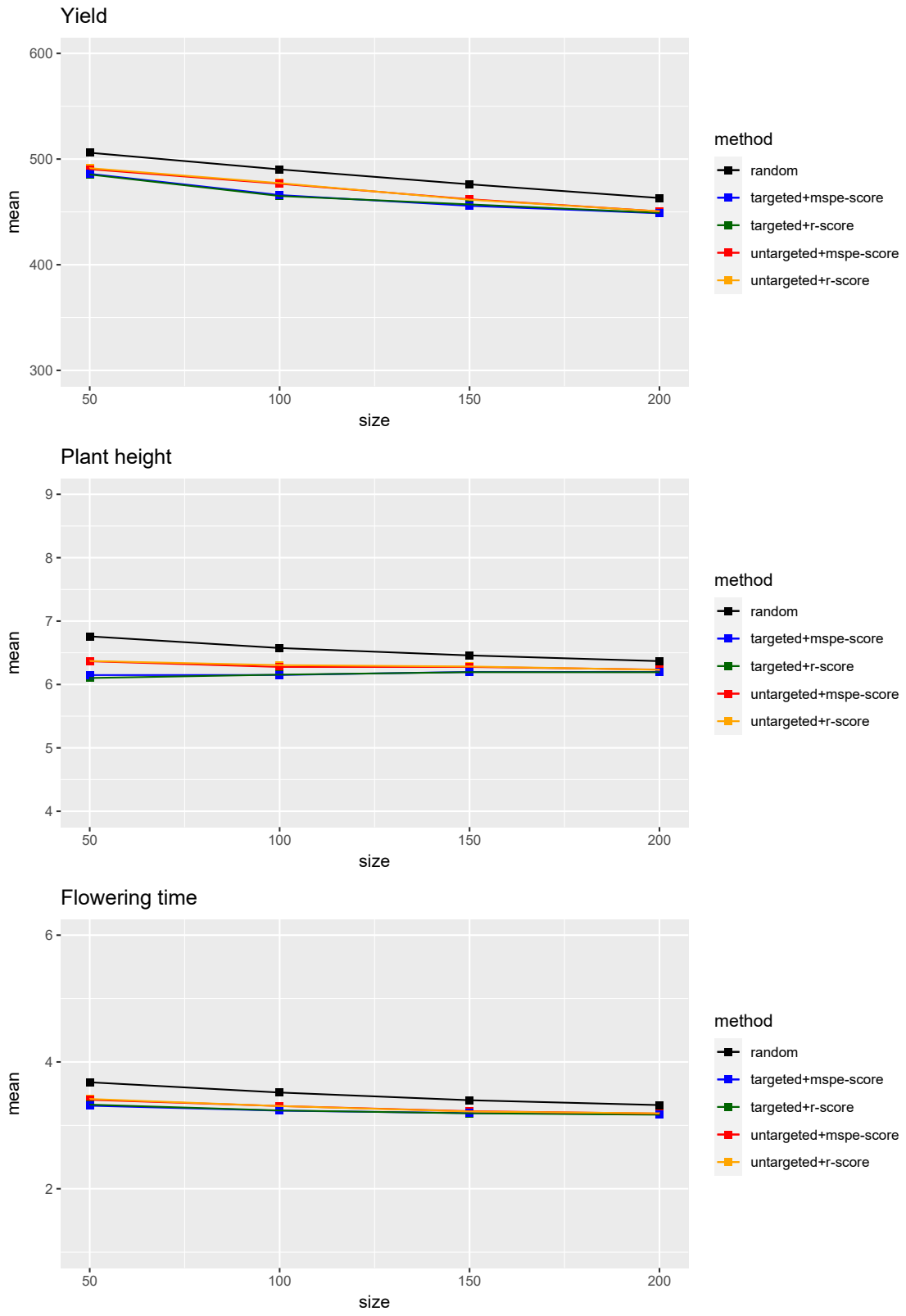

Figure 3.2: Mean of RMSEs for the three traits in the tropical rice dataset

Pearson's Correlation

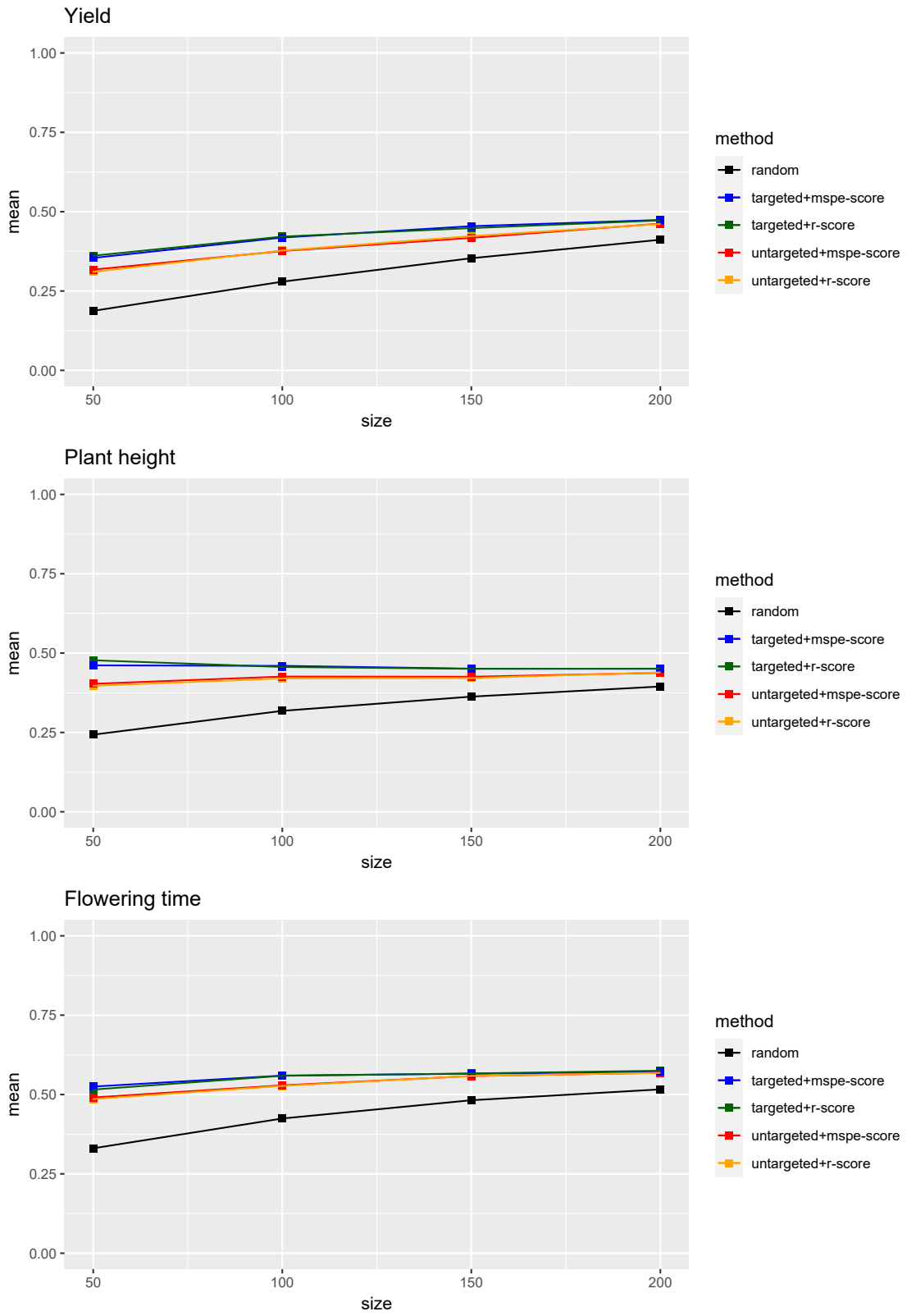

Figure 3.3: Mean of Pearson's correlations for the three traits in the tropical rice dataset

According to the results, we can find that the prediction accuracy increases with the size of the training set on average regardless of phenotypes. Besides, the targeted scheme has higher prediction accuracy than the untargeted and random ones but the untargeted scheme still has higher prediction accuracy than the random scheme especially when the sample size of the training set is small. The difference between the methods gradually vanishes as the training set size increases.

3.1.2 Rice44K data

Unlike the tropical rice dataset, the rice-44k dataset presented by Zhao et al. (2011) contains 300 individuals with six subpopulations (ADMIX, AROMATIC, AUS, IND, TEJ and TRJ) and 36901 SNPs after data processing. Therefore, the structure of the training set (Isidro et al. (2015)) should be considered in addition to the knowledge of genotypes of the testing set. In the same manner, the size of the testing set is set to be 50 and that of the training set selected by chance or GA is 50, 75, 100, 125 or 150. Subsequently, we choose four quantitative traits inclusive of flowering time at Arkansas, Faridpur and Aberdeen respectively and plant height to show our results of prediction accuracy as follows.

FT at Arkansas

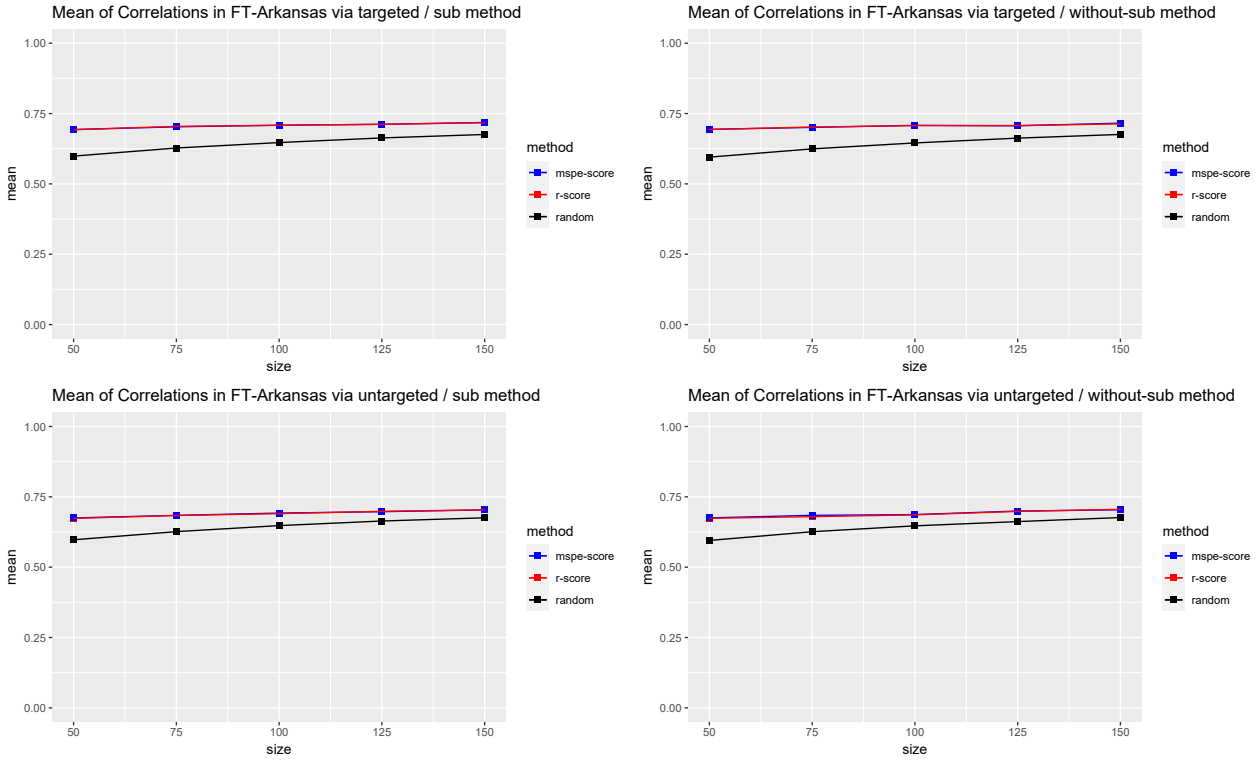

Figure 3.4: Mean of Pearson's correlations on FT-Arkansas

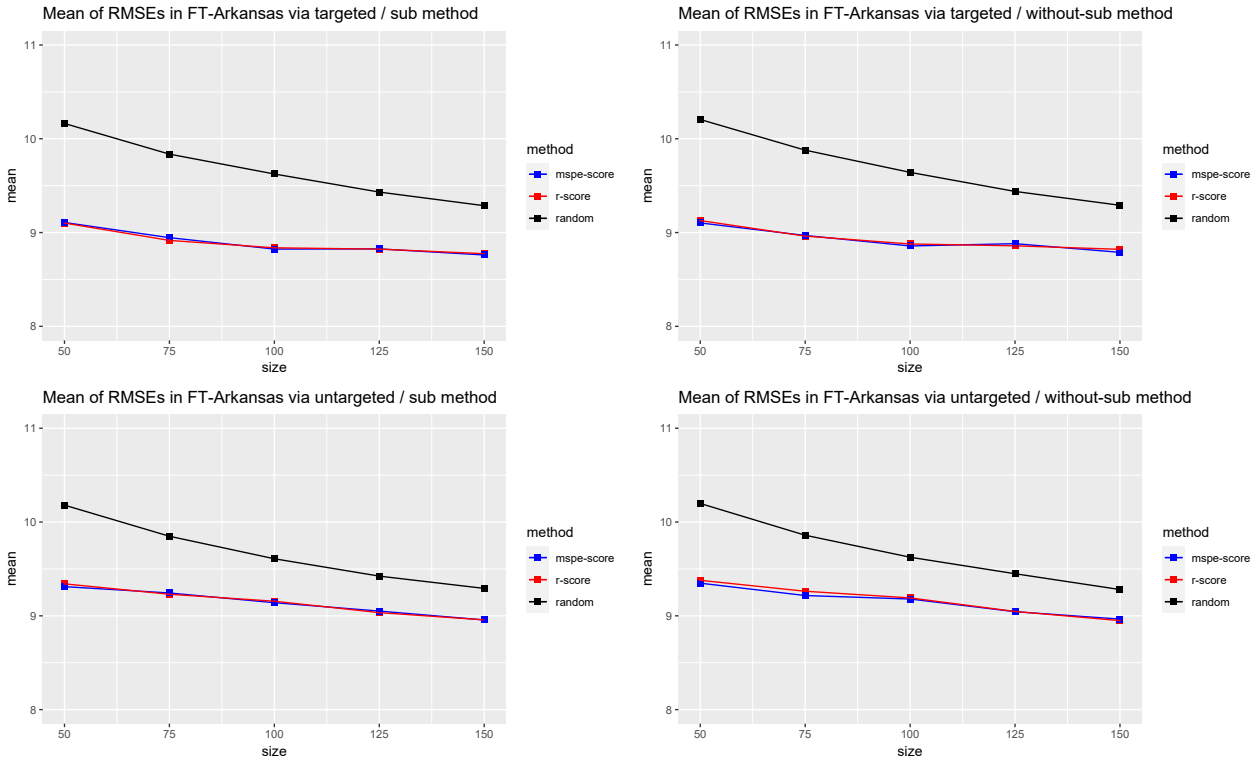

Figure 3.5: Mean of RMSEs on FT-Arkansas

FT at Faridpur

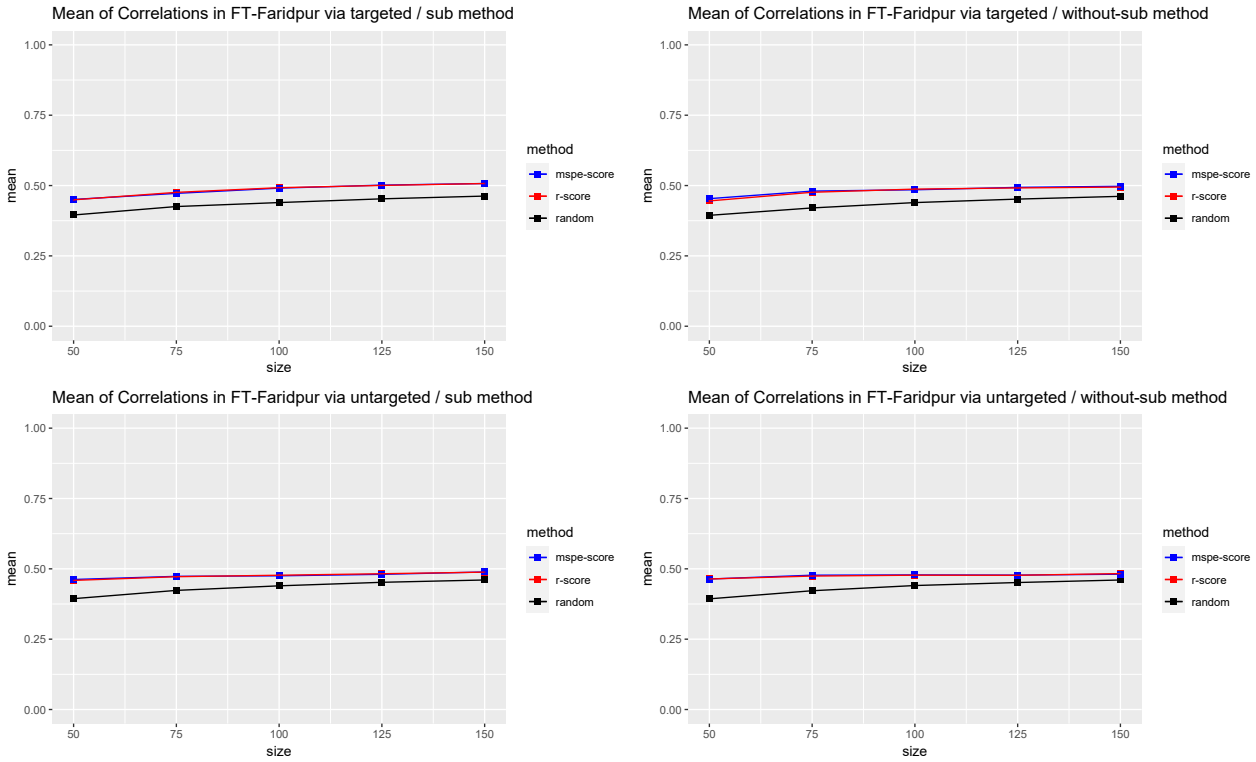

Figure 3.6: Mean of Pearson's correlations on FT-Faridpur

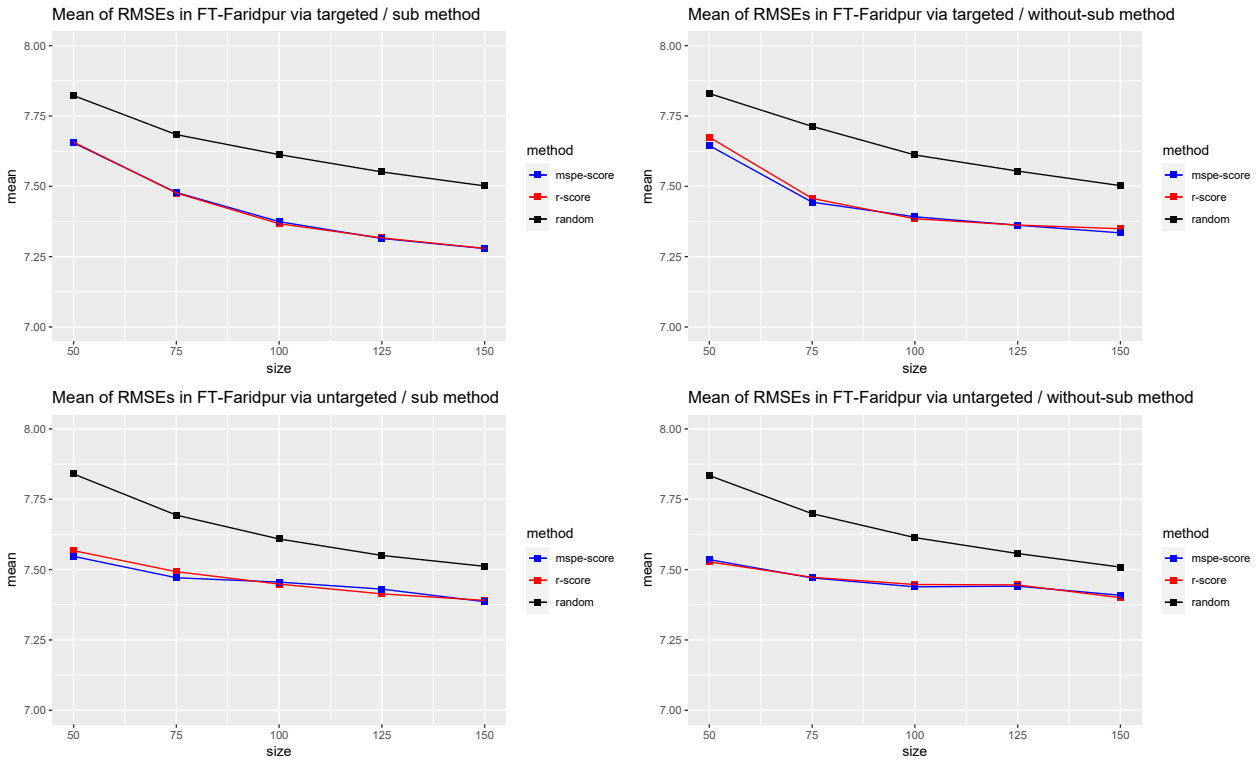

Figure 3.7: Mean of RMSEs on FT-Faridpur

FT at Aberdeen

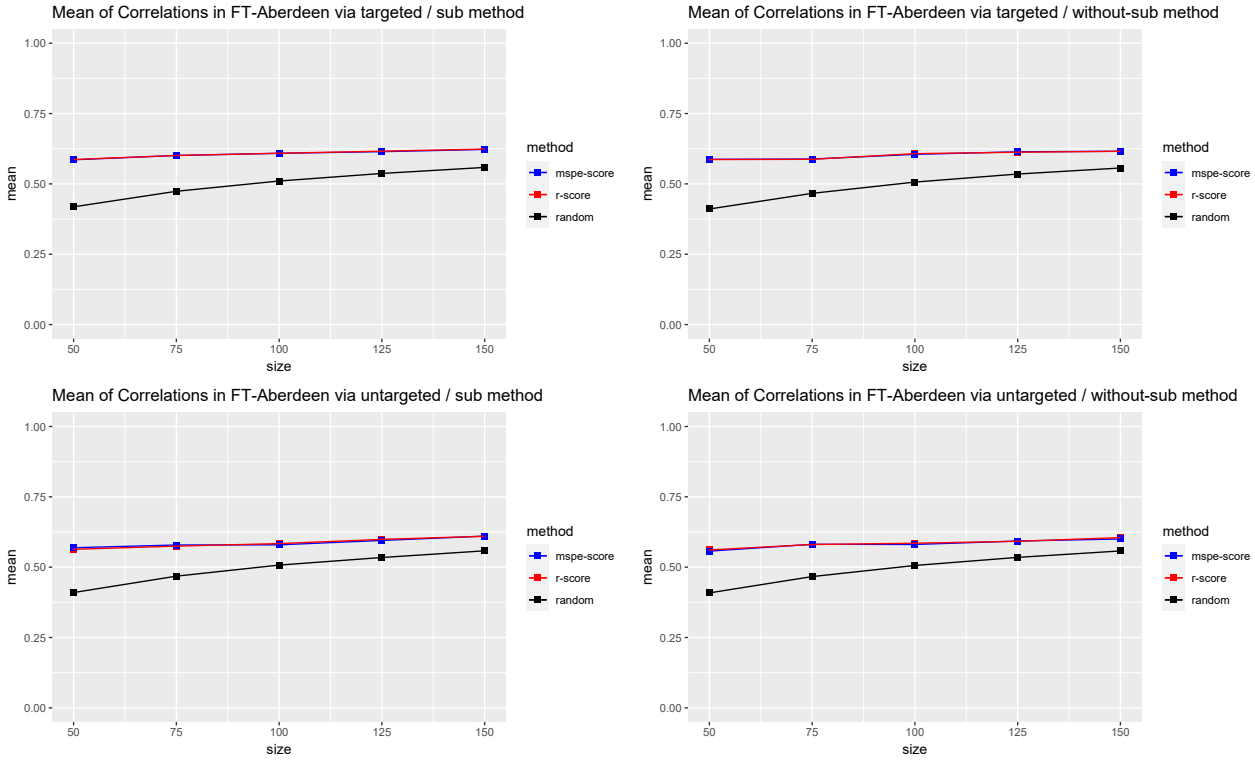

Figure 3.8: Mean of Pearson's correlations on FT-Aberdeen

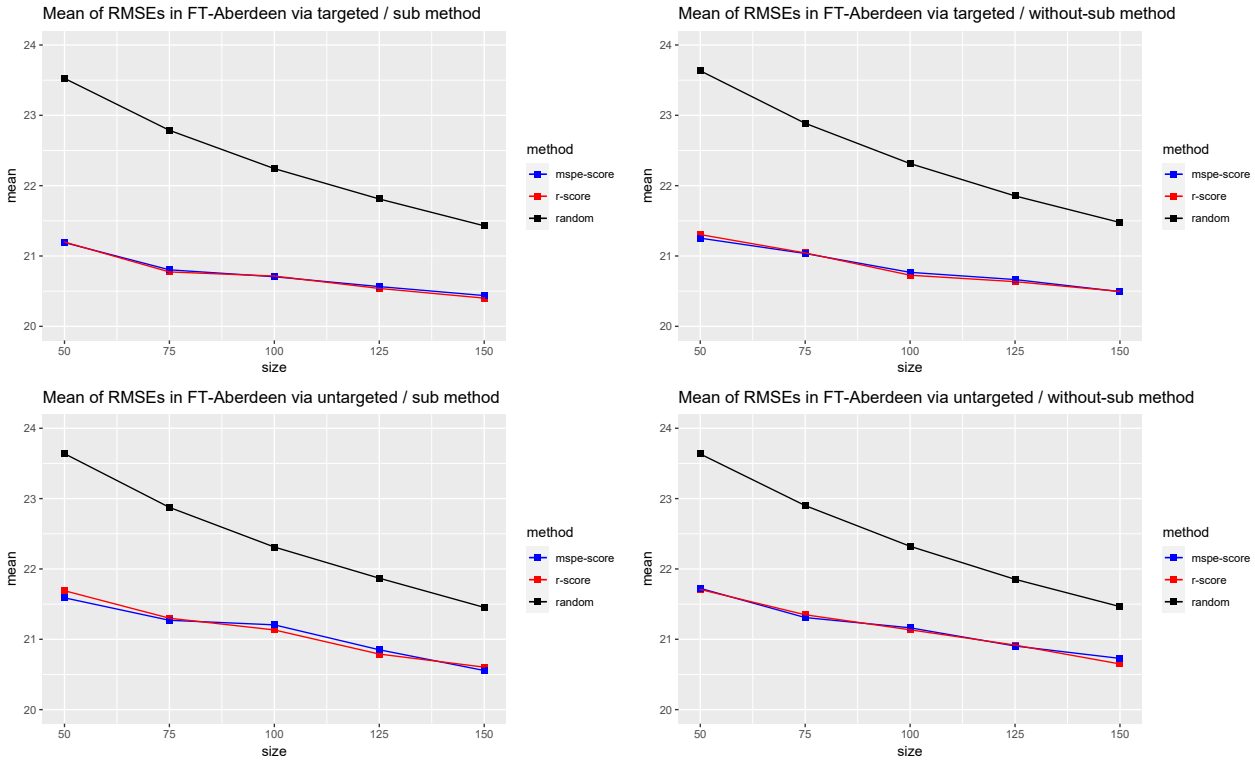

Figure 3.9: Mean of RMSEs on FT-Aberdeen

Plant height

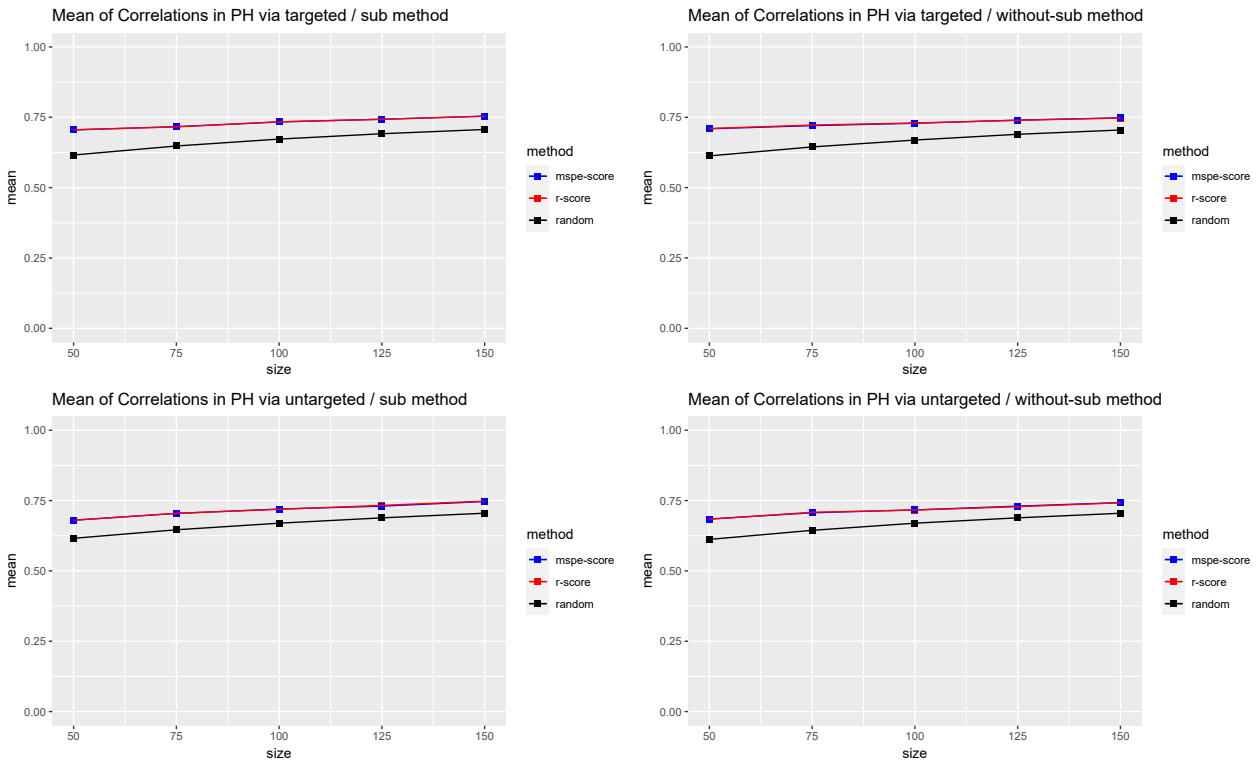

Figure 3.10: Mean of Pearson's correlations on plant heights

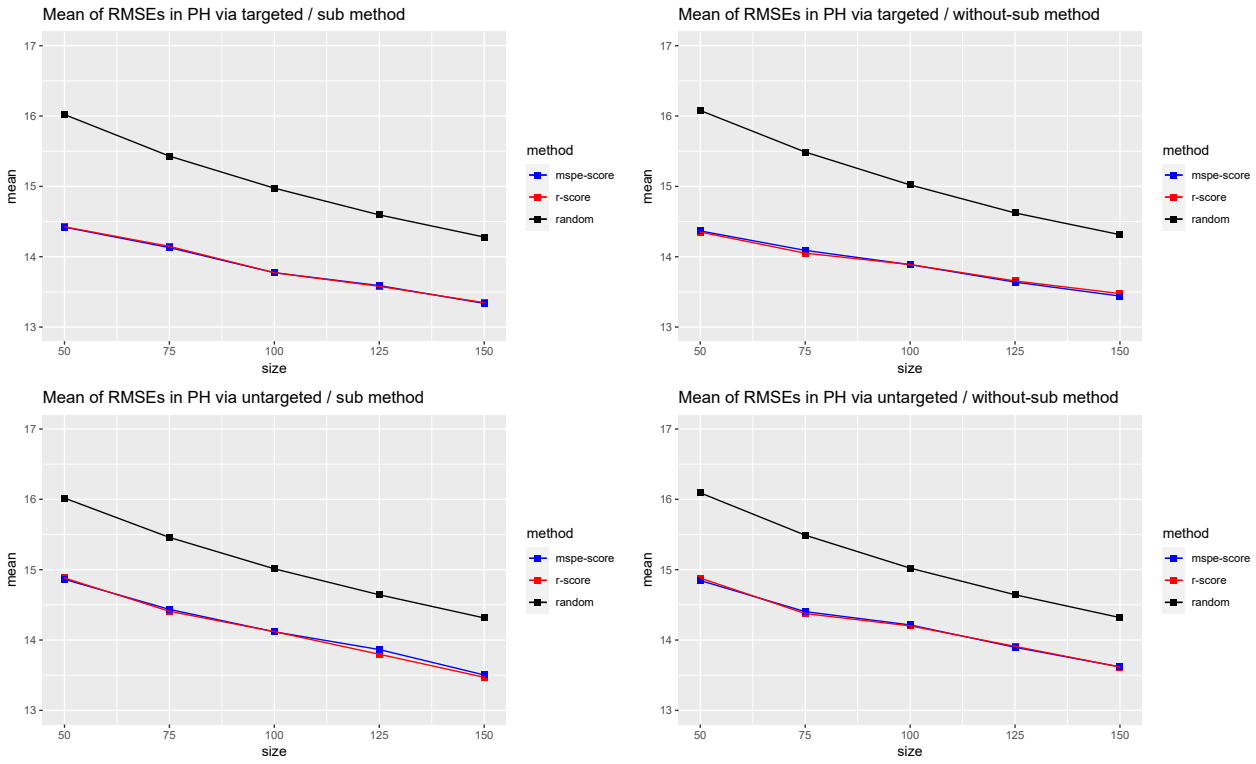

Figure 3.11: Mean of RMSEs on plant heights

In line with the consequences, we can come to the same conclusion that the targeted, untargeted and random schemes give the most, second most and least accurate indicators regardless of stratified sampling, which also give rise to higher Pearson's correlations and lower RMSEs as the sample size increases. However, there is little difference in prediction accuracy between stratified and random selection of training sets.

3.2 Simulation Study

In addition to the sample size and genetic relationship between genome data of training and testing sets, the heritability (h^2) of phenotypes is one of the main factors that also influences the prediction accuracy.

Hence, we simulate 5000 populations with $\mu = 100$, $\sigma_g^2 = 25$, h^2=0.2 , 0.5 , 0.8 respectively by using tropical rice and rice44k genome datasets as our templates. The first 10 testing and training sets of the 30 repetitions in the real data analysis are used to measure the prediction accuracy. Thus, the simulated phenotypic values for various scenarios will be sampled from MVN($100 \times \mathbf{1}_n, 25 \times \boldsymbol{K} + 25 \times \frac{1-h^2}{h^2} \times \boldsymbol{I}_n$), where $\boldsymbol{K}$ is the kinship matrix of the genome dataset.

3.2.1 Tropical rice data

The results for the tropical rice template are given as follows.

RMSE

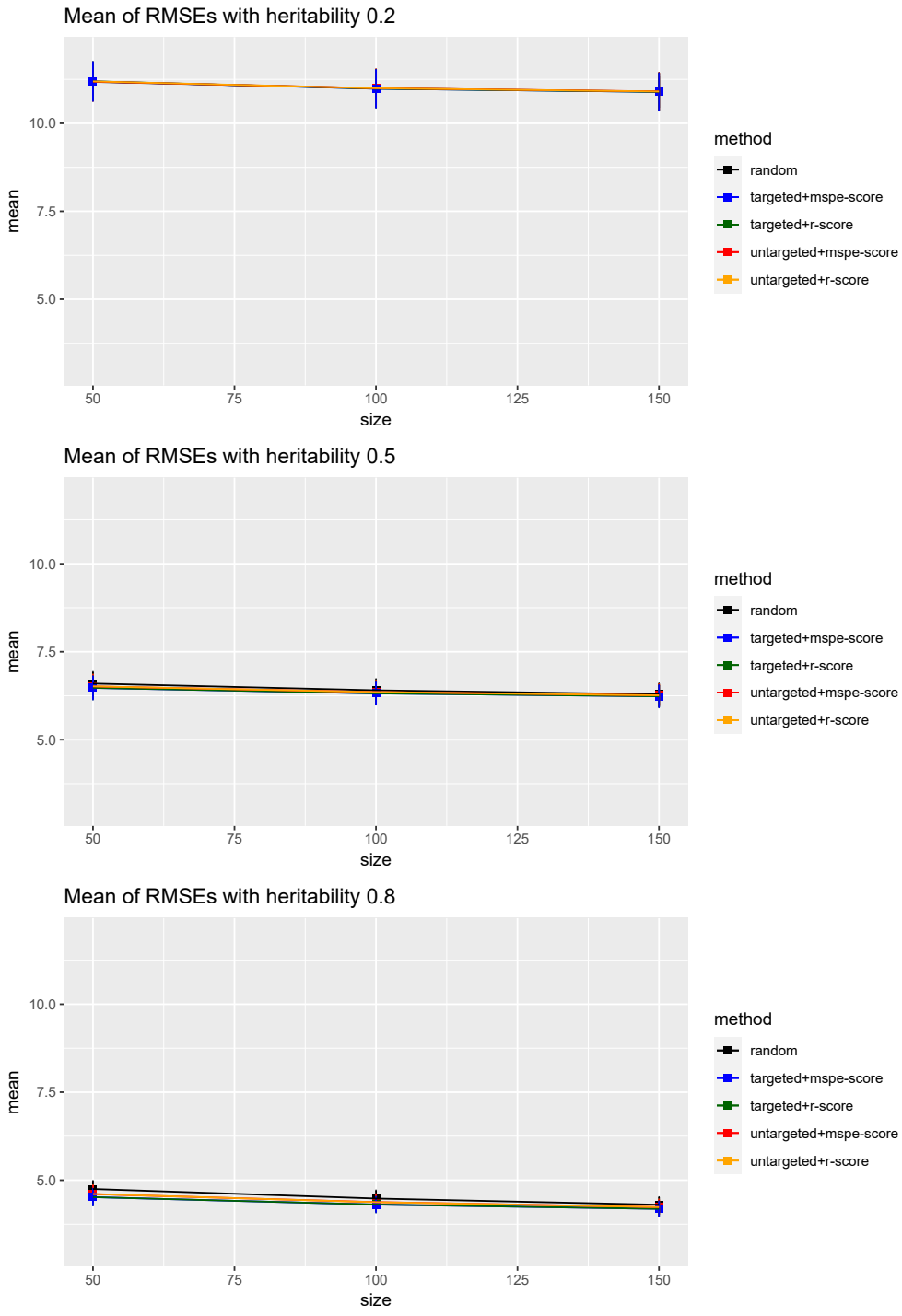

Figure 3.12: Mean of RMSEs in simulation studies based on the tropical rice data

Table 3.1: Mean and SD of RMSEs in simulation studies based on the tropical rice data

Criteria	Method	h^2	n_t=50	n_t=100	n_t=150
r-score	targeted	0.2	11.194(0.552)	10.984(0.537)	10.894(0.531)
		0.5	6.474(0.330)	6.317(0.319)	6.234(0.316)
		0.8	4.518(0.239)	4.307(0.221)	4.185(0.213)
	untargeted	0.2	11.191(0.477)	10.998(0.538)	10.905(0.532)
		0.5	6.517(0.332)	6.354(0.319)	6.261(0.315)
		0.8	4.607(0.248)	4.373(0.224)	4.233(0.214)
mspe-score	targeted	0.2	11.191(0.554)	10.983(0.539)	10.894(0.531)
		0.5	6.473(0.330)	6.317(0.318)	6.235(0.314)
		0.8	4.518(0.240)	4.306(0.221)	4.184(0.214)
	untargeted	0.2	11.185(0.550)	10.998(0.537)	10.907(0.533)
		0.5	6.515(0.332)	6.356(0.319)	6.263(0.314)
		0.8	4.604(0.249)	4.376(0.224)	4.234(0.214)
random	targeted	0.2	11.175(0.469)	10.997(0.461)	10.902(0.457)
		0.5	6.597(0.331)	6.404(0.320)	6.290(0.312)
		0.8	4.758(0.263)	4.478(0.232)	4.301(0.218)
	untargeted	0.2	11.175(0.469)	10.997(0.461)	10.902(0.457)
		0.5	6.597(0.331)	6.404(0.320)	6.290(0.312)
		0.8	4.758(0.263)	4.478(0.232)	4.301(0.218)

Correlation

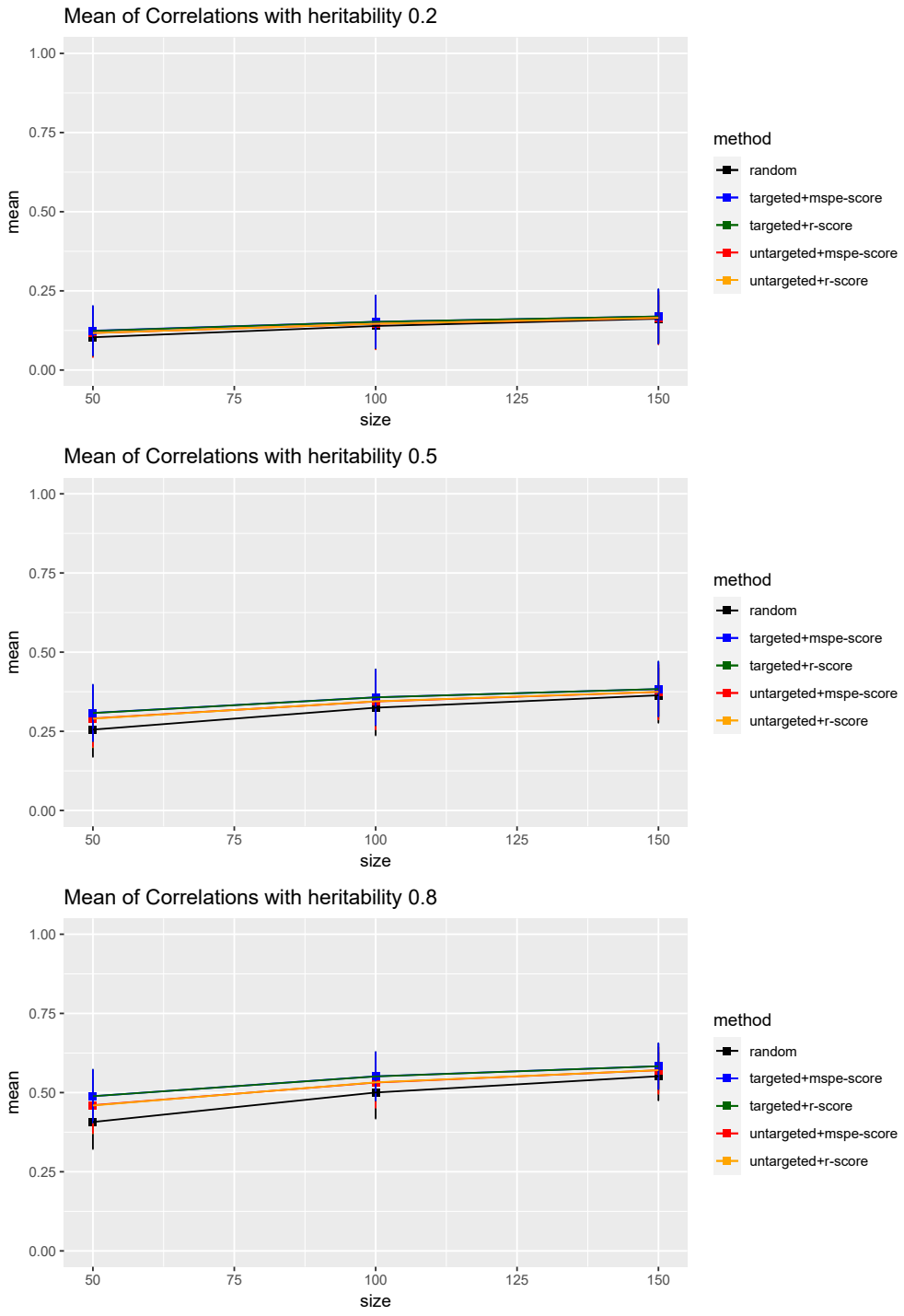

Figure 3.13: Mean of ρ in simulation studies based on the tropical rice data

Table 3.2: Mean and SD of ρ in simulation studies based on the tropical rice data

Criteria	Method	h^2	n_t=50	n_t=100	n_t=150
r-score	targeted	0.2	0.123(0.078)	0.152(0.083)	0.169(0.085)
		0.5	0.308(0.089)	0.357(0.088)	0.383(0.086)
		0.8	0.489(0.083)	0.551(0.076)	0.583(0.072)
	untargeted	0.2	0.117(0.068)	0.146(0.080)	0.165(0.083)
		0.5	0.290(0.090)	0.345(0.088)	0.375(0.087)
		0.8	0.459(0.088)	0.532(0.079)	0.571(0.074)
mspe-score	targeted	0.2	0.124(0.078)	0.152(0.083)	0.169(0.086)
		0.5	0.308(0.089)	0.357(0.087)	0.383(0.086)
		0.8	0.489(0.083)	0.551(0.076)	0.583(0.072)
	untargeted	0.2	0.117(0.076)	0.146(0.080)	0.164(0.083)
		0.5	0.291(0.089)	0.344(0.088)	0.374(0.087)
		0.8	0.460(0.088)	0.532(0.079)	0.571(0.074)
random	targeted	0.2	0.104(0.059)	0.139(0.070)	0.162(0.075)
		0.5	0.255(0.086)	0.325(0.088)	0.364(0.087)
		0.8	0.404(0.090)	0.500(0.082)	0.552(0.076)
	untargeted	0.2	0.104(0.059)	0.139(0.070)	0.162(0.075)
		0.5	0.255(0.086)	0.325(0.088)	0.364(0.087)
		0.8	0.404(0.090)	0.500(0.082)	0.552(0.076)

3.2.2 Rice44K data

As for the template of rice44k genome, the results of prediction accuracy including Pearson's correlation and Root Mean Square error within 3 methods (random, r-score and mspe-score) and 3 different kinds of heritabilities are illustrated among 3 schemes (targeted and stratified, targeted and unstratified design, untargeted and unstratified design) in the simulation studies shown on the next page.

RMSE

Targeted and stratified

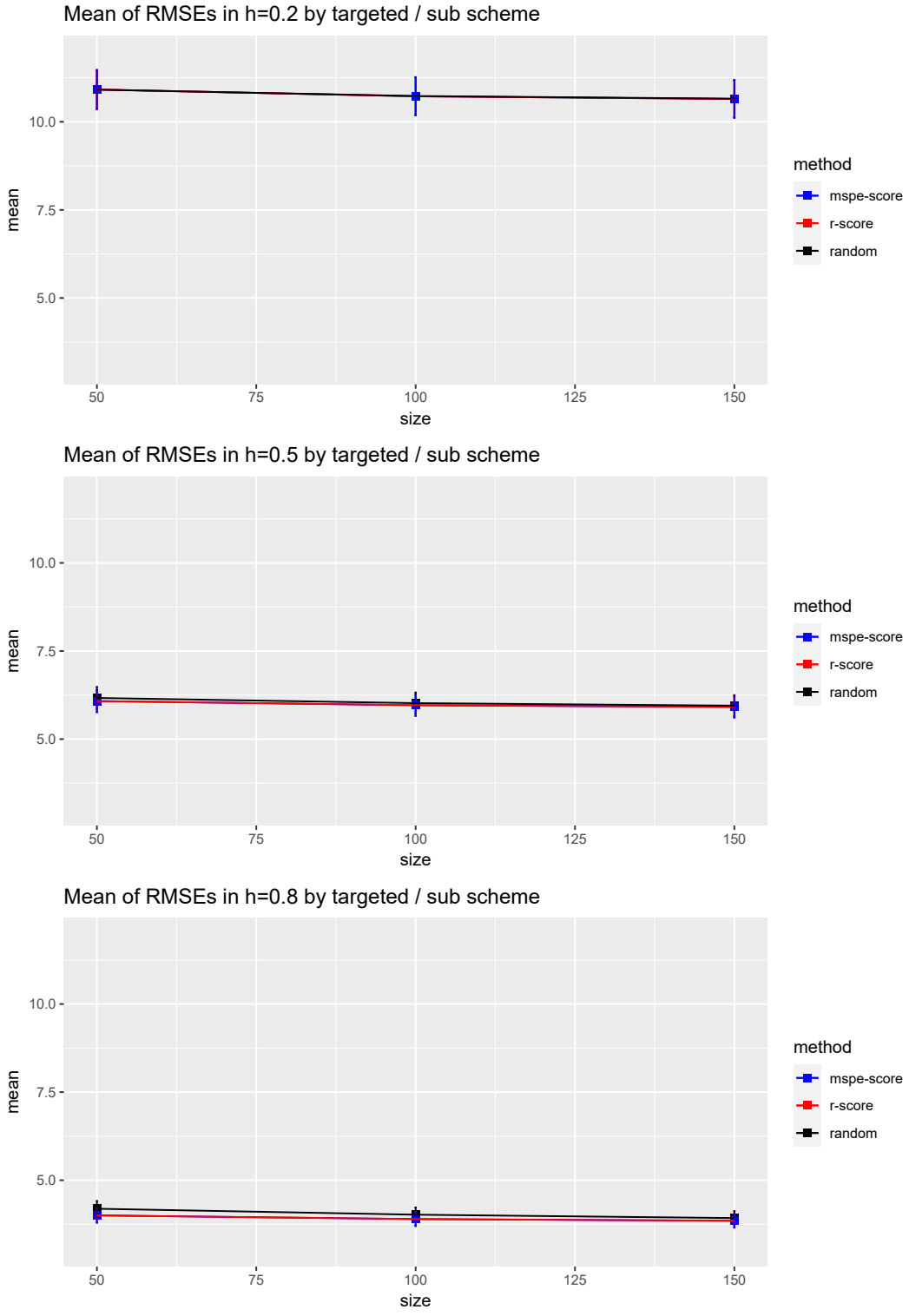

Figure 3.14: Mean of RMSEs in simulation studies based on targeted and stratified schemes of rice44k data

Targeted and unstratified

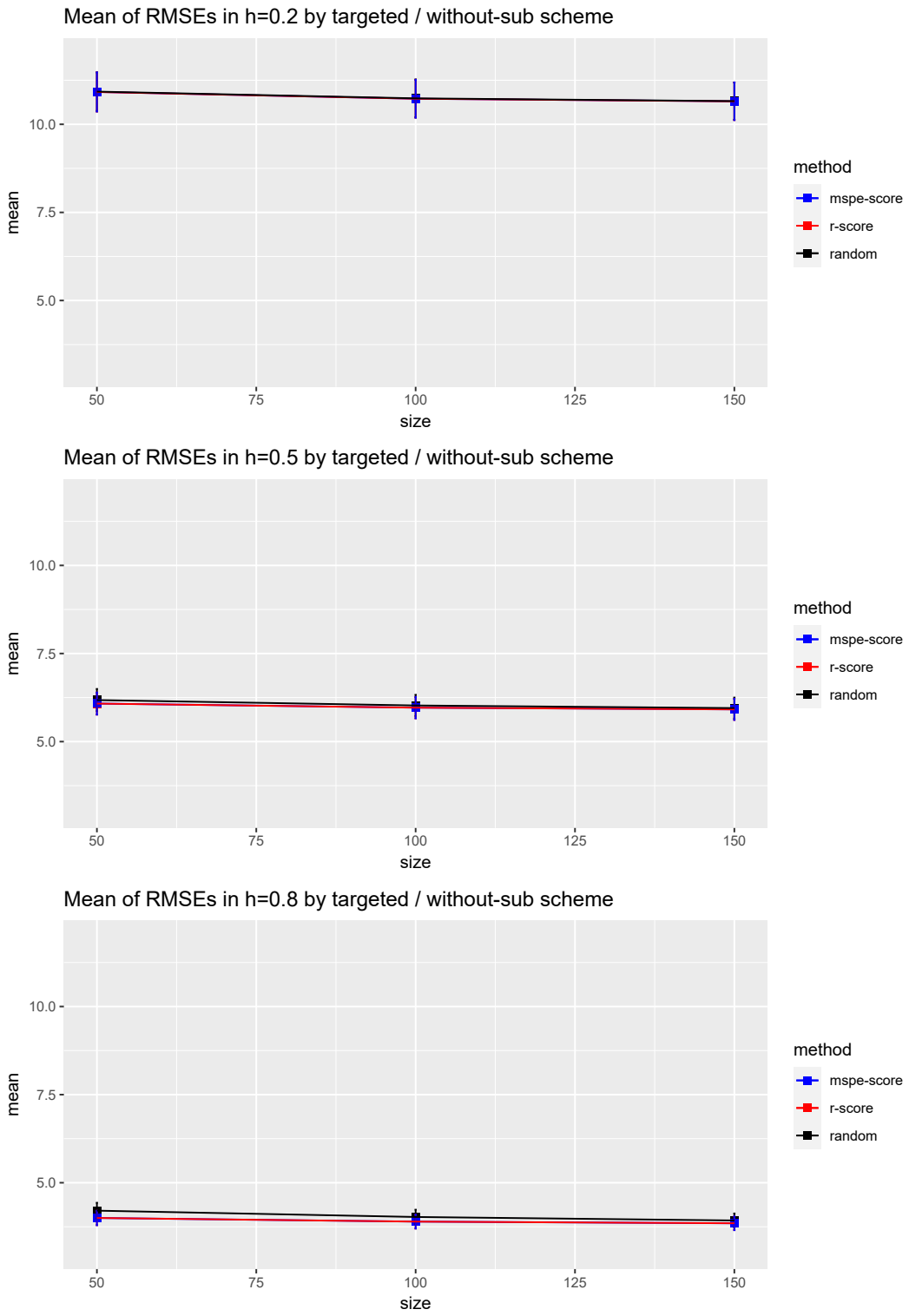

Figure 3.15: Mean of RMSEs in simulation studies based on targeted and unstratified schemes of rice44k data

Untargeted and unstratified

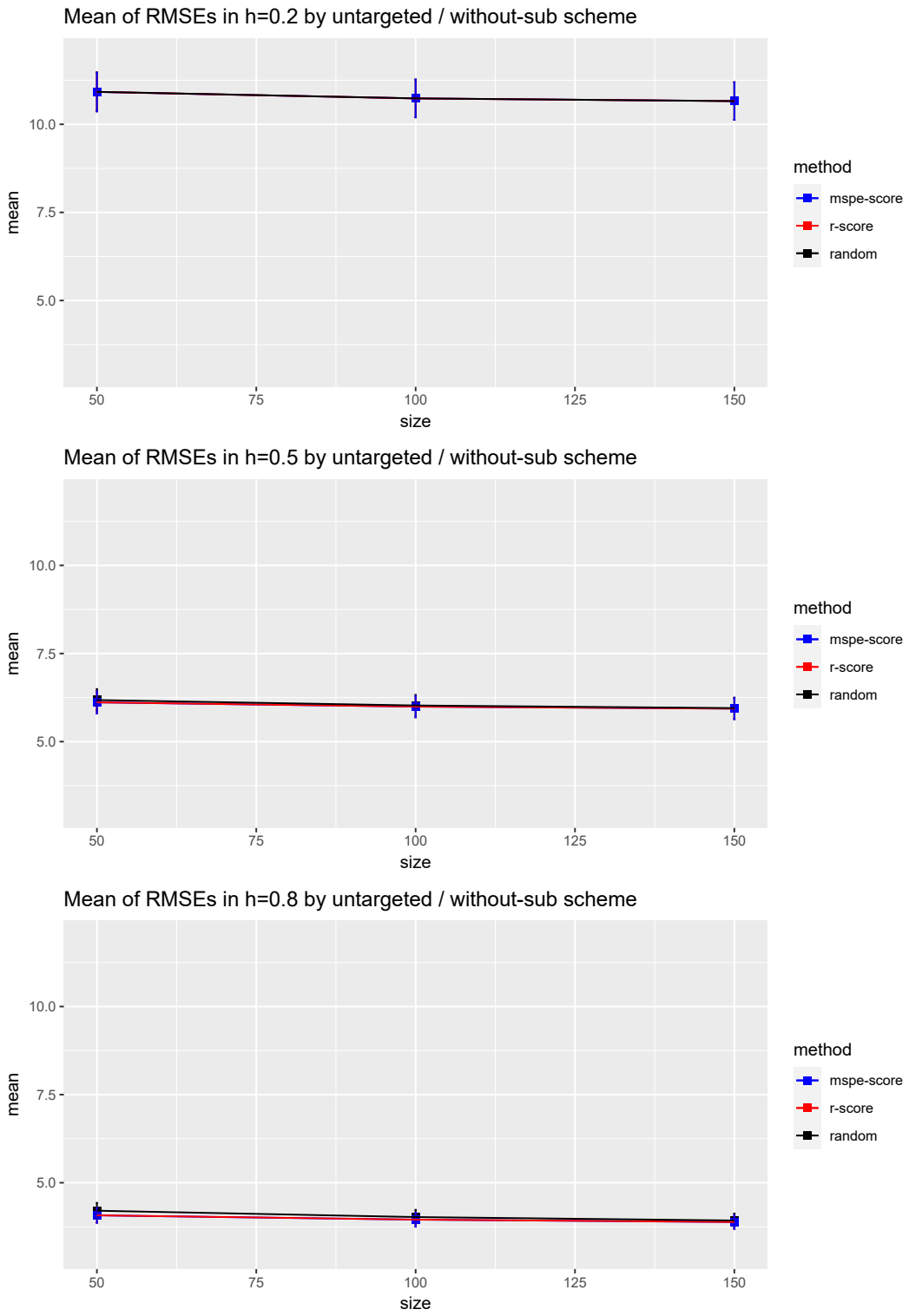

Figure 3.16: Mean of RMSEs in simulation studies based on untargeted and unstratified schemes of rice44k data

Table 3.3: Mean and SD of RMSEs in the simulation study based on the rice44K data

Criteria	Method	stratified	h^2	n_t=50	n_t=100	n_t=150
r-score	targeted	yes	0.2	10.917(0.556)	10.724(0.540)	10.648(0.534)
			0.5	6.079(0.314)	5.962(0.304)	5.913(0.300)
			0.8	4.002(0.206)	3.898(0.198)	3.850(0.195)
		no	0.2	10.917(0.558)	10.726(0.539)	10.652(0.533)
			0.5	6.079(0.314)	5.963(0.304)	5.914(0.300)
			0.8	4.000(0.205)	3.900(0.197)	3.850(0.195)
	untargeted	no	0.2	10.920(0.551)	10.734(0.536)	10.660(0.532)
			0.5	6.114(0.314)	5.993(0.303)	5.936(0.300)
			0.8	4.075(0.211)	3.953(0.200)	3.885(0.196)
mspe-score	targeted	yes	0.2	10.915(0.555)	10.724(0.539)	10.648(0.534)
			0.5	6.079(0.314)	5.962(0.304)	5.913(0.301)
			0.8	4.001(0.205)	3.898(0.198)	3.850(0.195)
		no	0.2	10.918(0.557)	10.726(0.539)	10.653(0.535)
			0.5	6.080(0.314)	5.963(0.304)	5.914(0.299)
			0.8	4.000(0.206)	3.897(0.198)	3.849(0.196)
	untargeted	no	0.2	10.918(0.552)	10.735(0.536)	10.660(0.534)
			0.5	6.113(0.315)	5.993(0.303)	5.936(0.300)
			0.8	4.074(0.212)	3.952(0.200)	3.885(0.196)
random	targeted	yes	0.2	10.912(0.547)	10.732(0.536)	10.656(0.533)
			0.5	6.167(0.314)	6.021(0.301)	5.949(0.299)
			0.8	4.191(0.221)	4.022(0.205)	3.926(0.199)
		no	0.2	10.928(0.548)	10.737(0.538)	10.658(0.531)
			0.5	6.179(0.316)	6.026(0.303)	5.952(0.298)
			0.8	4.207(0.224)	4.028(0.206)	3.929(0.198)
	untargeted	no	0.2	10.928(0.548)	10.737(0.538)	10.658(0.531)
			0.5	6.179(0.316)	6.026(0.303)	5.952(0.298)
			0.8	4.207(0.224)	4.028(0.206)	3.929(0.198)

Correlation

Targeted and stratified

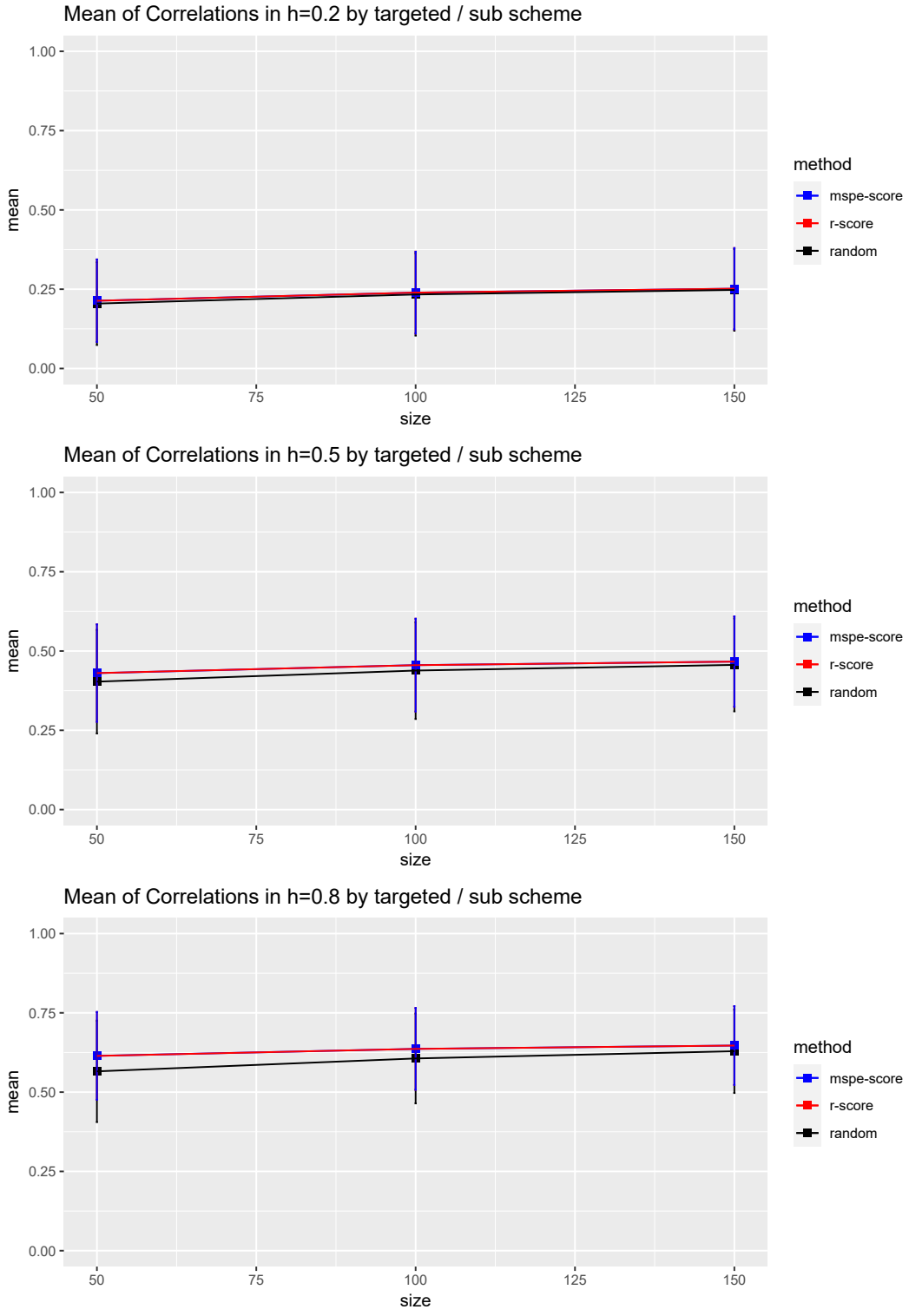

Figure 3.17: Mean of ρ in simulation studies based on targeted and stratified schemes of rice44k data

Targeted and unstratified

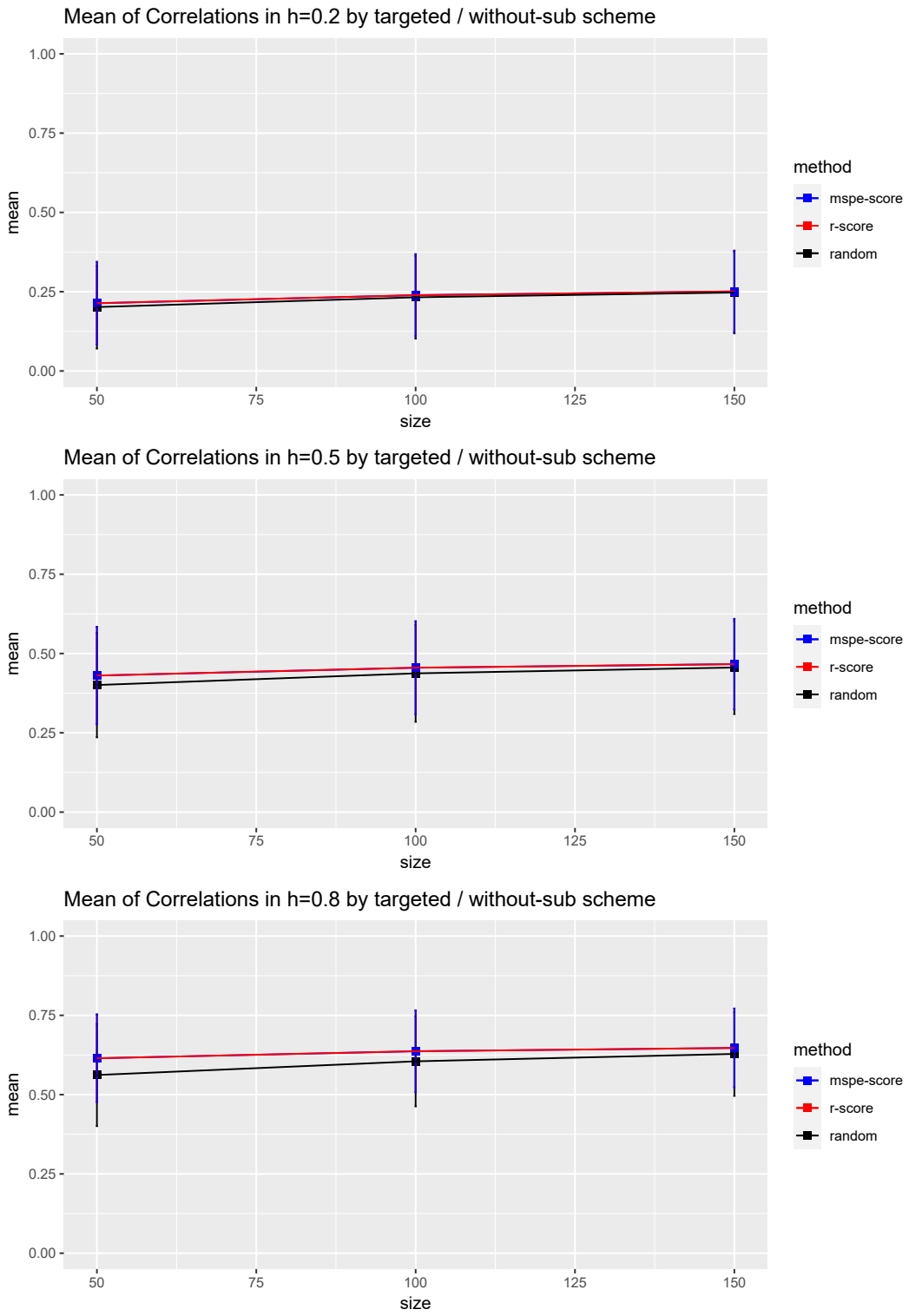

Figure 3.18: Mean of ρ in simulation studies based on targeted and unstratified schemes of rice44k data

Untargeted and unstratified

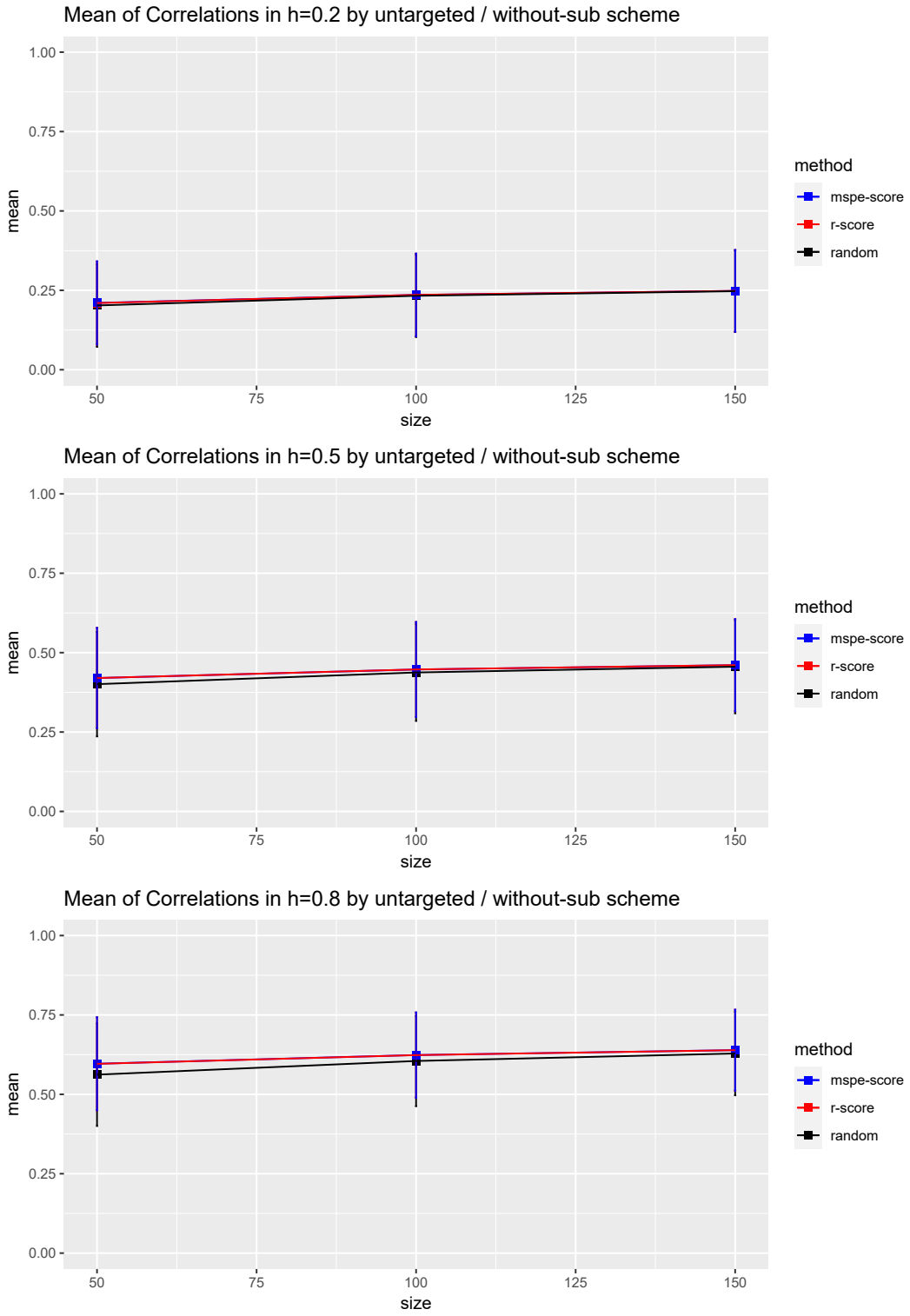

Figure 3.19: Mean of ρ in simulation studies based on untargeted and unstratified schemes of rice44k data

Table 3.4: Mean and SD of Pearson's correlations in the simulation study based on the rice44K data

Criteria	Method	stratified	h^2	n_t=50	n_t=100	n_t=150
r-score	targeted	yes	0.2	0.213(0.130)	0.239(0.129)	0.252(0.128)
			0.5	0.430(0.154)	0.455(0.147)	0.467(0.143)
			0.8	0.614(0.138)	0.636(0.129)	0.647(0.124)
		no	0.2	0.214(0.130)	0.239(0.129)	0.251(0.129)
			0.5	0.431(0.154)	0.455(0.146)	0.467(0.143)
			0.8	0.615(0.138)	0.637(0.129)	0.647(0.124)
	untargeted	no	0.2	0.210(0.131)	0.236(0.130)	0.248(0.129)
			0.5	0.420(0.158)	0.447(0.150)	0.461(0.145)
			0.8	0.596(0.147)	0.623(0.134)	0.639(0.128)
mspe-score	targeted	yes	0.2	0.214(0.130)	0.239(0.129)	0.252(0.128)
			0.5	0.430(0.154)	0.455(0.146)	0.467(0.143)
			0.8	0.614(0.139)	0.636(0.129)	0.647(0.124)
		no	0.2	0.213(0.130)	0.239(0.129)	0.251(0.128)
			0.5	0.430(0.154)	0.455(0.147)	0.466(0.143)
			0.8	0.615(0.138)	0.637(0.129)	0.647(0.124)
	untargeted	no	0.2	0.210(0.131)	0.236(0.130)	0.248(0.129)
			0.5	0.420(0.158)	0.447(0.150)	0.461(0.145)
			0.8	0.596(0.147)	0.624(0.134)	0.639(0.128)
random	targeted	yes	0.2	0.205(0.131)	0.233(0.130)	0.248(0.129)
			0.5	0.403(0.163)	0.438(0.153)	0.456(0.147)
			0.8	0.566(0.160)	0.606(0.142)	0.629(0.132)
		no	0.2	0.202(0.131)	0.232(0.130)	0.247(0.129)
			0.5	0.401(0.165)	0.437(0.153)	0.456(0.147)
			0.8	0.562(0.161)	0.605(0.142)	0.628(0.132)
	untargeted	no	0.2	0.202(0.131)	0.232(0.130)	0.247(0.129)
			0.5	0.401(0.165)	0.437(0.153)	0.456(0.147)
			0.8	0.562(0.161)	0.605(0.142)	0.628(0.132)

Based on the results of simulations from both the templates of tropical rice and rice44k genomes, a GS model with the targeted method is more accurate than those with the untargeted and random ones, which can be observed from the results of Pearson's correlations or root mean square errors. Besides, stratified sampling also plays a crucial role in improving the prediction accuracy according to the template of rice44k genome, because it takes advantage of the information about subpopulation structure of the candidate and testing sets.

As for the heritability, the difference of prediction accuracy among random, targeted and untargeted methods becomes more significant as the heritability increases. The higher the heritability, the more the prediction accuracy. Also, Pearson's correlation is recommended as a prediction index when the heritability is small while there seems to be little difference in RMSEs among the simulation scenarios.

Chapter 4 Discussion

Owing to the fact that we cannot jump to conclusions only by the real data analysis without considering whether or not stratified sampling, knowledge of testing sets and heritability that may have influences on prediction accuracy. Therefore, simulation studies are usually required to investigate the performance of the training set optimization criteria.

Different from the objective function, the trace of PEV^{Ridge}, proposed by Akdemir et al.(2015) considering only the prediction error variance, r-score and mspe-score take both variance and bias into account. Hence, r-score and mspe-score are expected to lead to higher accuracy. From the simulation study, the prediction performance of GS using r-score and mspe-score criteria are more accurate than random sampling and it is obvious that stratified sampling gives the better prediction accuracy than random sampling. The box-plots of the proportions that the two criteria lead to the identical candidates in various training set sizes are displayed in Figures 22 and 24. Moreover, the scatter plots of r-scores against their corresponding mspe-scores are displayed in Figures 23 and 25. From these figures, we can see why the r-score and mspe-score criteria have almost same prediction performance.

Nonetheless, mspe-score may be suggested in practical use. One of the reasons is that the derivation of r-score consists 3 expectation approximations while that of mspe-score has only one. Another reason is that the computation of mspe-score is much more time-saving. For instance, assume an extreme situation that the RMSE equals zero, i.e., $\boldsymbol{y}_i = \hat{\boldsymbol{y}}_i$ for all i, the correlation between $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$

is equal to 1. Conversely, when the correlation between $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ is equal to 1, it only means that $\boldsymbol{y}_i = a \times \hat{\boldsymbol{y}}_i + b$, and RMSE is $\sqrt{\frac{1}{n} \times \sum_{i=1}^{n} [(a-1) \times \hat{\boldsymbol{y}}_i + b]^2}$ for this case, which is not necessarily equal to 0. Hence, it can be seen that the lower RMSE implies the higher correlation while the higher correlation does not certainly imply the lower RMSE. That is why the mspe-score is advised to serve as the criterion for training set optimization.

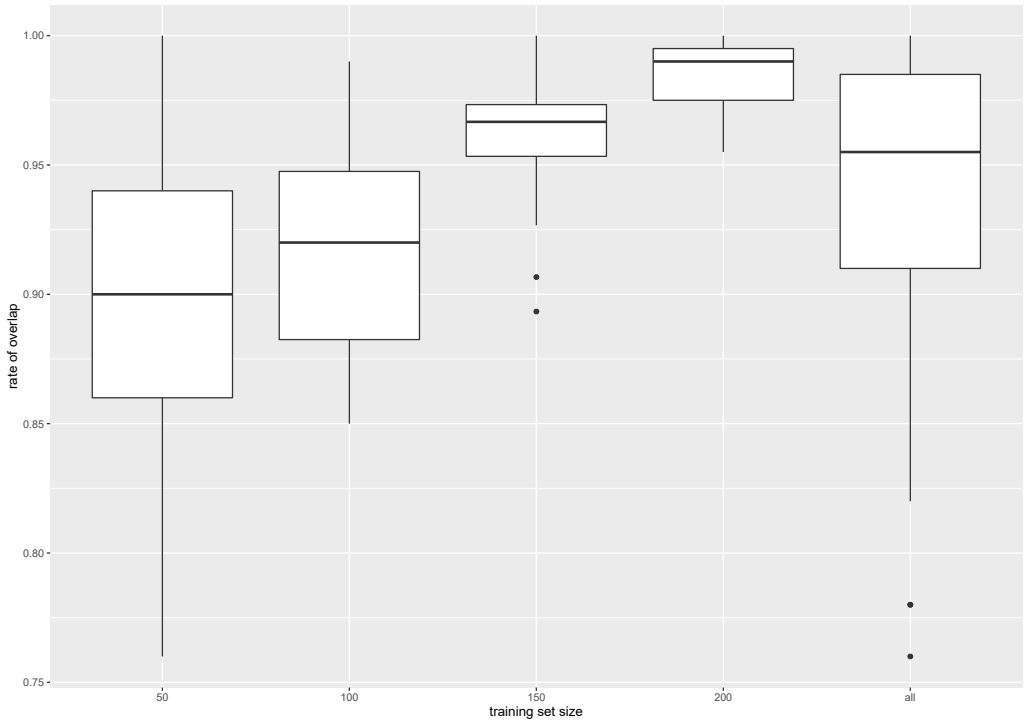

Figure 4.1: Proportions of overlap in training sets through r-score and mspe-score criteria in a targeted and unstratified scheme of the tropical rice dataset.

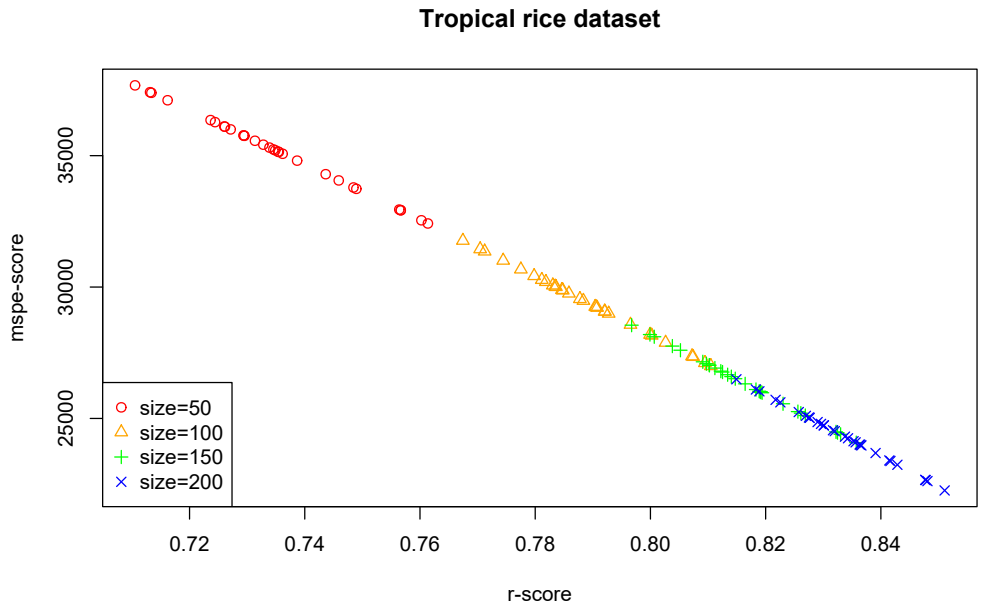

Figure 4.2: The scatter plot of r-scores against mspe-scores in a targeted and unstratified scheme of the tropical rice dataset.

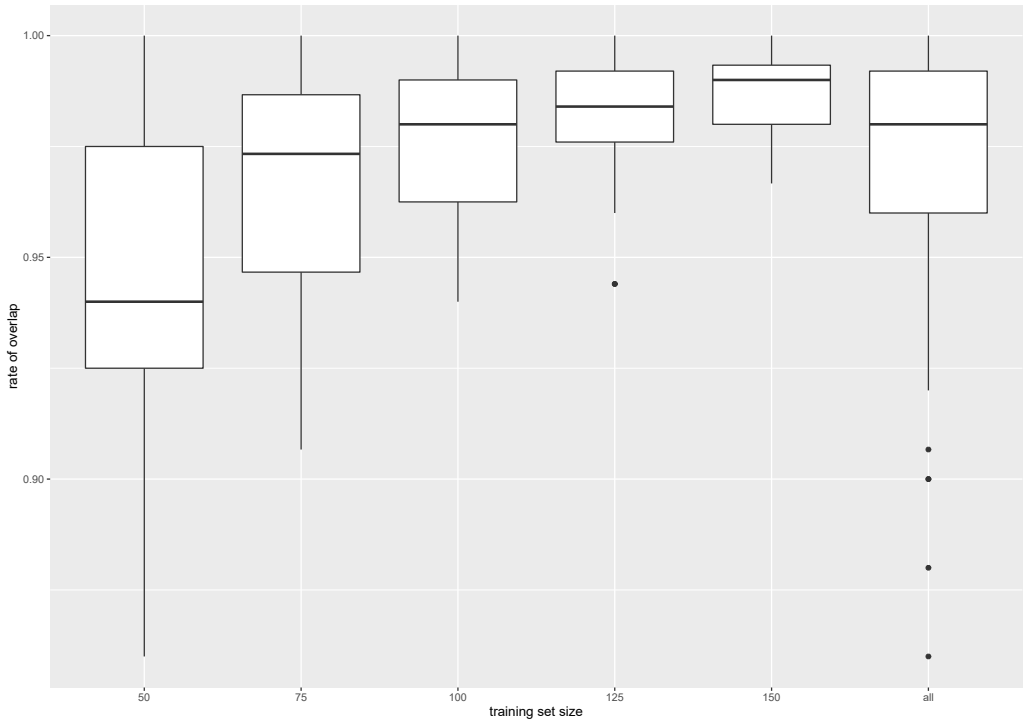

Figure 4.3: Proportions of overlap in training sets through r-score and mspe-score criteria in a targeted and unstratified scheme of the rice44k dataset.

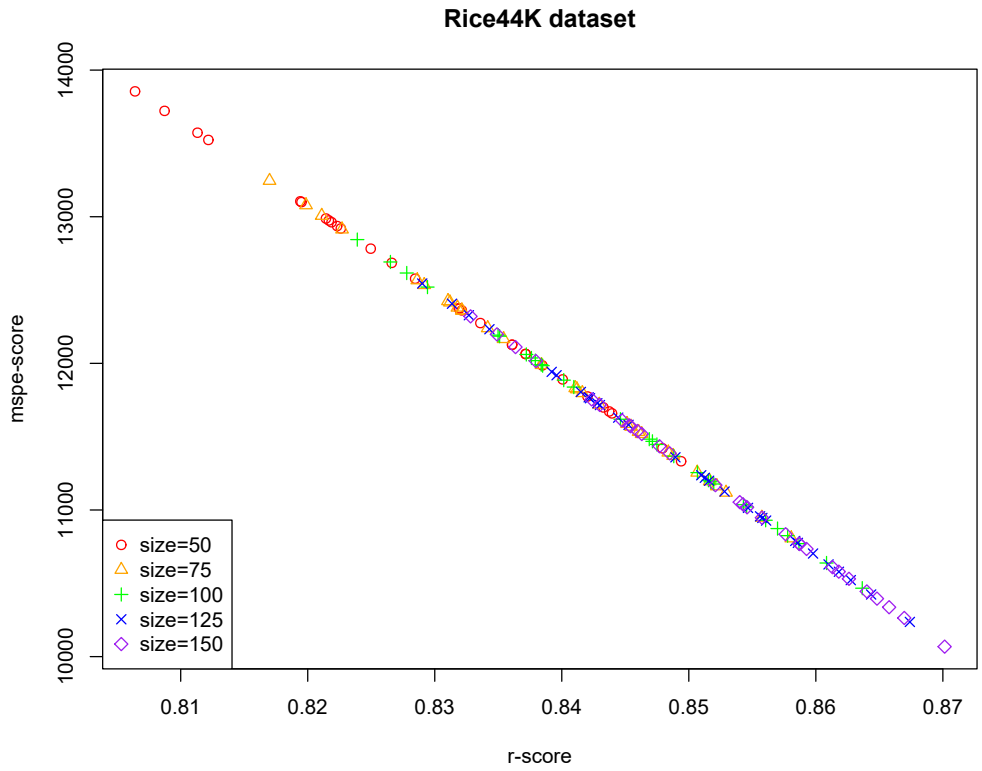

Figure 4.4: The scatter plot of r-scores against mspe-scores in a targeted and unstratified scheme of the rice44K dataset.

Chapter 5 Bibliography

Bibliography

[1] T. H. Meuwissen, B. J. Hayes, and M. E. Goddard. Prediction of total genetic value using genome wide dense marker maps.GENETICS, 157(4):1819–1829, 2001.

[2] S. Maenhout, B. De Baets, and G. Haesaert. Graph-based data selection for the construction of genomic prediction models.Genetics, 185(4):1463-75, 2010.

[3] E. L. Heffner, M. E. Sorrells, and J. L. Jannink. Genomic selection for crop improvement, Crop-Science, 49(1):1–12, 2009.

[4] A. J. Lorenz, K. Smith, and J. L. Jannink. Potential and optimization of genomic selection for fusarium head blight resistance in six-row barley, Crop Science,52(4):1609–1621, 2012.

[5] V. Wimmer, C. Lehermeier, T. Albrecht, H.J. Auinger, Y. Wang, and C.C. Schön. Genome wide prediction of traits with different genetic architecture through efficient variable selection.GENETICS, 195(2):573–587, 2013.

[6] R. Rincent, D. Laloë, S. Nicolas, T. Altmann, D. Brunel, P. Revilla, V.M. Rodríguez, J. Moreno-Gonzalez, A. Melchinger, E. Bauer, C-C. Schoen, N. Meyer, C. Giauffret, C. Bauland, P. Jamin, J. Laborde, H. Monod, P. Flament, A. Charcosset, and L. Moreau. Maximizing the Reliability of Genomic Selection by Optimizing the Calibration Set of Reference Individuals: Comparison of Methods in Two Diverse Groups of Maize Inbreds (Zea mays L.), Genetics, 192(2):715–728, 2012

[7] D.Akdemir, J. I. Sanchez, and J. L. Jannink. Optimization of genomic selection training populations with a genetic algorithm.Genetics Selection Evolution, 47(1):38,2015.

[8] A. Xavier, W. M. Muir, B. Craig, and K. M. Rainey. Walking through the statistical black boxes of plant breeding.Theoretical and Applied Genetics, 129(10):1933–1949, 2016.

[9] M. M. Shariati, P. Sørensen, and L. Janss. A two step bayesian approach for genomic prediction of breeding values.BMC Proceedings, 6(2):S12, 2012.

[10] C. R. Henderson. Estimation of genetic parameters.Annals of Mathematical Statistics, 21(3):309–310, 1950.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm.Journal of the Royal Statistical Society.Series B(Methodology), 39(1):1–38, 1977.

[12] S. German and D. German. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images.IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(6):721–741, 1984.

[13] S. R. Searle and A. I. Khuri. Matrix Algebra Useful for Statistics, chapter 10 and chapter 13,pages 261,355. Wiley, 1982.

[14] J. H. Holland. Genetic algorithms and adaptation, NATO Conference Series, 16(1):317–333, 1975.

[15] D. Whitley. A genetic algorithm tutorial, Statistics and Computing, 4(2):65–85,1994.

[16] D. Akdemir and J. I. Sánchez. Design of training populations for selective phenotyping in genomic prediction, Scientific Reports, 9(1):1446, 2019.

[17] R. Rincent, A. Charcosset, and L. Moreau. Predicting genomic selection efficiency to optimize calibration set and to assess prediction accuracy in highly structured populations, Theoretical and Applied Genetics, 130(11):2231–2247, 2017

[18] J. Isidro, J. L. Jannink, D. Akdemir, J. Poland, N. Heslot, and M. E. Sorrells. Training set optimization under population structure in genomic selection, Theoretical and Applied Genetics, 128(1):145–158, 2015.

[19] K. Zhao, C. W. Tung, G. C. Eizenga, M. H. Wright, M. L. Ali, A. H. Price, G. J.Norton, M. R. Islam, A. Reynolds, J. Mezey, A. M. McClung, C. D. Bustamante,and S. R. McCouch. Genomewide association mapping reveals a rich genetic architecture of complex traits in oryza sativa, Nature Communications, 2(467), 2011

[20] J.H. Ou and C.T. Liao. Training set determination for genomic selection, Theoretical and Applied Genetics, 132(10):2781–2792, 2019

[21] P. Perez and G. Campos. Genomewide regression and prediction with the bglr statistical package, Genetics, 198(2):483–495, 2014

[22] G. Morota, P. Boddhireddy, N. Vukasinovic, D. Gianola, and S. DeNise. Kernel-based variance component estimation and whole-genome prediction of pre-corrected phenotypes and progeny tests for dairy cow health traits. Front Genet, 5(56):10–3389, 2014

Appendix A — Source code

A.1 Genetic algorithm

```
optTrain = function(geno, cand, n.train, subpop=NULL, test=NULL,target=TRUE,
   ↪ method="rScore", min.iter=NULL){
  if(!method%in%c("rScore","MSPE")){stop("Method not found. Please choose one
     ↪ from (rScore,MSPE)")}
  n=n.train; N=nrow(geno); Nc=length(cand)
  geno = as.matrix(geno)
  if(target==TRUE){
    subpop=as.character(subpop)
    if(method=="MSPE"){
      ## pop target MSPE
      if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}
      sol=matrix(NA, n, 30);
      for(i in 1:30){
        pops = names(table(subpop))
        pop.ratio = round(n*as.numeric(table(subpop[test]))/sum(as.numeric(
           ↪ table(subpop[test]))))
```

```
  stop=0
  if(pop.ratio[which.min(pop.ratio)]==1){pop.ratio[which.min(pop.ratio)
    ↪ ]=2}
  while(stop==0){
    if(sum(pop.ratio)>n){
      pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
        ↪ ratio==max(pop.ratio))[1]]-1
    }else if(sum(pop.ratio)<n){
      pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
        ↪ ratio==max(pop.ratio))[1]]+1
    }else{
      stop=1
    }
  }
  a=c()
  for(j in 1:length(pops)){
    a = c(a, sample(cand[subpop[cand]==pops[j]],pop.ratio[j]))
  }
  sol[,i] = sort(a)}
score=rep(NA,30); for(i in 1:30){score[i]=mspe(geno[sol[,i],], geno[test
  ↪ ,])}
iter.score=score; top.score=min(score)
stop=0; iter=1
while(stop==0){
```

```
elite = which(rank(1/score)>27)
del = sample(seq(30)[-elite], 3, prob=((score[-elite])/sum(score[-elite
    ↪ ])))
for(i in 1:length(del)){
  par = sample(seq(30)[-del],2)
  b=c()
  for(j in 1:length(pops)){
    b=c(b,sample(unique(c(sol[,par[1]],sol[,par[2]]))[subpop[unique(c(
        ↪ sol[,par[1]],sol[,par[2]]))]==pops[j]],pop.ratio[j]))
    sol[,del[i]] = sort(b)}
}
for(i in 1:30){
  new.sol=sol[,i]
  p = sample(length(pops),1,prob=pop.ratio)
  if(i %in% del){
    sol[,i][sample(which(subpop[sol[,i]]==pops[p]),1)]=cand[!cand%in%
        ↪ sol[,i]][sample(which(subpop[cand[!cand%in%sol[,i]]]==pops[p
        ↪ ]),1)]
    score[i] = mspe(geno[sol[,i],], geno[test,])
  }else{
    new.sol[sample(which(subpop[new.sol]==pops[p]),1)]=cand[!cand%in%
        ↪ new.sol][sample(which(subpop[cand[!cand%in%new.sol]]==pops[p
        ↪ ]),1)]
    new.score = mspe(geno[new.sol,], geno[test,])
```

```
        if(new.score < score[i]){sol[,i] = new.sol; score[i]=new.score}
      }
    }
    iter.score=c(iter.score,mean(score)); top.score=c(top.score, min(score)
        ↪ )
    cat(iter, "..", sep=""); iter = iter + 1
    if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
        ↪ ==0){stop=1}}
  }
  sol = sol[,which(score==min(score))[1]]
}else if(method=="rScore"){
  ## pop target r-score
  if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}
  sol=matrix(NA, n, 30);
  for(i in 1:30){
    pops = names(table(subpop))
    pop.ratio = round(n*as.numeric(table(subpop[test]))/sum(as.numeric(
        ↪ table(subpop[test]))))
    stop=0
    if(pop.ratio[which.min(pop.ratio)]==1){pop.ratio[which.min(pop.ratio)
        ↪ ]=2}
    while(stop==0){
      if(sum(pop.ratio)>n){
        pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
```

```
        ↪ ratio==max(pop.ratio))[1]]-1
    }else if(sum(pop.ratio)<n){
      pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
        ↪ ratio==max(pop.ratio))[1]]+1
    }else{
      stop=1
    }
  }
  a=c()
  for(j in 1:length(pops)){
    a = c(a, sample(cand[subpop[cand]==pops[j]],pop.ratio[j]))
  }
  sol[,i] = sort(a)}
score=rep(NA,30); for(i in 1:30){score[i]=r_score(geno[sol[,i],], geno[
  ↪ test,])}
iter.score=score; top.score=max(score)
stop=0; iter=1
while(stop==0){
  elite = which(rank(score)>27)
  del = sample(seq(30)[-elite], 3, prob=((1/score[-elite])/sum(1/score[-
    ↪ elite])))
  for(i in 1:length(del)){
    par = sample(seq(30)[-del],2)
    b=c()
```

```
  for(j in 1:length(pops)){
    b=c(b,sample(unique(c(sol[,par[1]],sol[,par[2]]))[subpop[unique(c(
        ↪ sol[,par[1]],sol[,par[2]]))]==pops[j]],pop.ratio[j]))
    sol[,del[i]] = sort(b)}
}
for(i in 1:30){
  new.sol=sol[,i]
  p = sample(length(pops),1,prob=pop.ratio)
  if(i %in% del){
    sol[,i][sample(which(subpop[sol[,i]]==pops[p]),1)]=cand[!cand%in%
        ↪ sol[,i]][sample(which(subpop[cand[!cand%in%sol[,i]]]==pops[p
        ↪ ]),1)]
    score[i] = r_score(geno[sol[,i],], geno[test,])
  }else{
    new.sol[sample(which(subpop[sol[,i]]==pops[p]),1)]=cand[!cand%in%
        ↪ new.sol][sample(which(subpop[cand[!cand%in%new.sol]]==pops[p
        ↪ ]),1)]
    new.score = r_score(geno[new.sol,], geno[test,])
    if(new.score > score[i]){sol[,i] = new.sol; score[i]=new.score}
  }
}
iter.score=c(iter.score,mean(score)); top.score=c(top.score, max(score)
    ↪ )
cat(iter, "..", sep=""); iter = iter + 1
```

```
    if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
        ↪ ==0){stop=1}}
  }
  sol = sol[,which(score==max(score))[1]]
}}else{
  subpop=as.character(subpop)
  if(method=="MSPE"){
    ## pop untarget MSPE
    if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}
    sol=matrix(NA, n, 30);
    for(i in 1:30){
      pops = names(table(subpop))
      pop.ratio = round(n*as.numeric(table(subpop[cand]))/sum(as.numeric(
          ↪ table(subpop[cand]))))
      stop=0
      if(pop.ratio[which.min(pop.ratio)]==1){pop.ratio[which.min(pop.ratio)
          ↪ ]=2}
      while(stop==0){
        if(sum(pop.ratio)>n){
          pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(
              ↪ pop.ratio==max(pop.ratio))[1]]-1
        }else if(sum(pop.ratio)<n){
          pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(
              ↪ pop.ratio==max(pop.ratio))[1]]+1
```

```
    }else{
      stop=1
    }
  }
  a=c()
  for(j in 1:length(pops)){
    a = c(a, sample(cand[subpop[cand]==pops[j]],pop.ratio[j]))
  }
  sol[,i] = sort(a)}
score=rep(NA,30); for(i in 1:30){score[i]=mspe(geno[sol[,i],], geno[
   ↪ cand[!cand%in%sol[,i]],])}
iter.score=score; top.score=min(score)
stop=0; iter=1
while(stop==0){
  elite = which(rank(1/score)>27)
  del = sample(seq(30)[-elite], 3, prob=((score[-elite])/sum(score[-
     ↪ elite])))
  for(i in 1:length(del)){
    par = sample(seq(30)[-del],2)
    b=c()
    for(j in 1:length(pops)){
      b=c(b,sample(unique(c(sol[,par[1]],sol[,par[2]]))[subpop[unique(c
         ↪ (sol[,par[1]],sol[,par[2]]))]==pops[j]],pop.ratio[j]))
      sol[,del[i]] = sort(b)}
```

```
  }
  for(i in 1:30){
    new.sol=sol[,i]
    p = sample(length(pops),1,prob=pop.ratio)
    if(i %in% del){
      sol[,i][sample(which(subpop[sol[,i]]==pops[p]),1)]=cand[!cand%in%
          ↪ sol[,i]][sample(which(subpop[cand[!cand%in%sol[,i]]]==pops[
          ↪ p]),1)]
      score[i] = mspe(geno[sol[,i],], geno[cand[!cand%in%sol[,i]],])
    }else{
      new.sol[sample(which(subpop[new.sol]==pops[p]),1)]=cand[!cand%in%
          ↪ new.sol][sample(which(subpop[cand[!cand%in%new.sol]]==pops[
          ↪ p]),1)]
      new.score = mspe(geno[new.sol,], geno[cand[!cand%in%new.sol],])
      if(new.score < score[i]){sol[,i] = new.sol; score[i]=new.score}
    }
  }
  iter.score=c(iter.score,mean(score)); top.score=c(top.score, min(
      ↪ score))
  cat(iter, "..", sep=""); iter = iter + 1
  if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
      ↪ ==0){stop=1}}
}
sol = sol[,which(score==min(score))[1]]
```

```
}else if(method=="rScore"){

  ## pop untarget r-score

  if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}

  sol=matrix(NA, n, 30);

  for(i in 1:30){

    pops = names(table(subpop))

    pop.ratio = round(n*as.numeric(table(subpop[cand]))/sum(as.numeric(

        ↪ table(subpop[cand]))))

    stop=0

    if(pop.ratio[which.min(pop.ratio)]==1){pop.ratio[which.min(pop.ratio)

        ↪ ]=2}

    while(stop==0){

      if(sum(pop.ratio)>n){

        pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(

            ↪ pop.ratio==max(pop.ratio))[1]]-1

      }else if(sum(pop.ratio)<n){

        pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(

            ↪ pop.ratio==max(pop.ratio))[1]]+1

      }else{

        stop=1

      }

    }

    a=c()

    for(j in 1:length(pops)){
```

```
    a = c(a, sample(cand[subpop[cand]==pops[j]],pop.ratio[j]))
  }
  sol[,i] = sort(a)}
score=rep(NA,30); for(i in 1:30){score[i]=r_score(geno[sol[,i],], geno[
    ↪ cand[!cand%in%sol[,i]],])}
iter.score=score; top.score=max(score)
stop=0; iter=1
while(stop==0){
  elite = which(rank(score)>27)
  del = sample(seq(30)[-elite], 3, prob=((1/score[-elite])/sum(1/score
      ↪ [-elite])))
  for(i in 1:length(del)){
    par = sample(seq(30)[-del],2)
    b=c()
    for(j in 1:length(pops)){
      b=c(b,sample(unique(c(sol[,par[1]],sol[,par[2]]))[subpop[unique(c
          ↪ (sol[,par[1]],sol[,par[2]]))]==pops[j]],pop.ratio[j]))
      sol[,del[i]] = sort(b)}
  }
  for(i in 1:30){
    new.sol=sol[,i]
    p = sample(length(pops),1,prob=pop.ratio)
    if(i %in% del){
      sol[,i][sample(which(subpop[sol[,i]]==pops[p]),1)]=cand[!cand%in%
```

```
            ↪ sol[,i]][sample(which(subpop[cand[!cand%in%sol[,i]]]==pops[
            ↪ p]),1)]
          score[i] = r_score(geno[sol[,i],], geno[cand[!cand%in%sol[,i]],])
        }else{
          new.sol[sample(which(subpop[sol[,i]]==pops[p]),1)]=cand[!cand%in%
            ↪ new.sol][sample(which(subpop[cand[!cand%in%new.sol]]==pops[
            ↪ p]),1)]
          new.score = r_score(geno[new.sol,], geno[cand[!cand%in%new.sol
            ↪ ],])
          if(new.score > score[i]){sol[,i] = new.sol; score[i]=new.score}
        }
      }
      iter.score=c(iter.score,mean(score)); top.score=c(top.score, max(
        ↪ score))
      cat(iter, "..", sep=""); iter = iter + 1
      if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
        ↪ ==0){stop=1}}
    }
    sol = sol[,which(score==max(score))[1]]
  }}
ret = list(
  OPTtrain=as.numeric(sort(sol)),
  TOPscore=as.numeric(top.score[-1]),
  ITERscore=as.numeric(iter.score[-1])
```

```
  )

  return(ret)

}
```

A.2 Simple exchange algorithm

```
optTrain = function(geno, cand, n.train, subpop=NULL, test=NULL,target=TRUE,
   ↪ method="rScore", min.iter=NULL){
  if(!method%in%c("rScore","MSPE")){stop("Method not found. Please choose one
     ↪ from (rScore,MSPE)")}
  n=n.train; N=nrow(geno); Nc=length(cand)
  geno = as.matrix(geno)
  if(target==TRUE){
    subpop=as.character(subpop)
    if(method=="MSPE"){
      ## pop target MSPE
      if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}
      pops = names(table(subpop))
      pop.ratio = round(n*as.numeric(table(subpop[test]))/sum(as.numeric(table
         ↪ (subpop[test]))))
      stop=0
      while(stop==0){
        if(sum(pop.ratio)>n){
          pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
             ↪ ratio==max(pop.ratio))[1]]-1
        }else if(sum(pop.ratio)<n){
          pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
             ↪ ratio==max(pop.ratio))[1]]+1
        }else{
```

```
    stop=1
  }
}
sol=c()
for(i in 1:length(pops)){
  sol = c(sol, sample(cand[subpop[cand]==pops[i]],pop.ratio[i]))
}
score=mspe(geno[sol,], geno[test,])
iter.score=score; top.score=score
stop=0; iter=1
while(stop==0){
  new.sol=sol
  p = sample(length(pops),1,prob=pop.ratio)
  new.sol[sample(which(subpop[sol]==pops[p]),1)]=cand[!cand%in%new.sol][
    ↪ sample(which(subpop[cand[!cand%in%new.sol]]==pops[p]),1)]
  new.score=mspe(geno[new.sol,],geno[test,])
  if(new.score<score){
    sol=new.sol; score=new.score
    iter.score=c(iter.score,score)
    top.score=c(top.score,score)
  }else{
    iter.score=c(iter.score,new.score)
    top.score=c(top.score,score)
  }
```

```
    cat(iter, "..", sep=""); iter = iter + 1
    if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
       ↪ ==0){stop=1}}


  }
}else if(method=="rScore"){
  ## pop target r-score
  if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}
  pops = names(table(subpop))
  pop.ratio = round(n*as.numeric(table(subpop[test]))/sum(as.numeric(table
     ↪ (subpop[test]))))
  stop=0
  while(stop==0){
    if(sum(pop.ratio)>n){
      pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
         ↪ ratio==max(pop.ratio))[1]]-1
    }else if(sum(pop.ratio)<n){
      pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
         ↪ ratio==max(pop.ratio))[1]]+1
    }else{
      stop=1
    }
  }
  sol=c()
```

```
for(i in 1:length(pops)){
  sol = c(sol, sample(cand[subpop[cand]==pops[i]],pop.ratio[i]))
}
score=r_score(geno[sol,], geno[test,])
iter.score=score; top.score=score
stop=0; iter=1
while(stop==0){
  new.sol=sol
  p = sample(length(pops),1,prob=pop.ratio)
  new.sol[sample(which(subpop[sol]==pops[p]),1)]=cand[!cand%in%new.sol][
     ↪ sample(which(subpop[cand[!cand%in%new.sol]]==pops[p]),1)]
  new.score=r_score(geno[new.sol,],geno[test,])
  if(new.score>score){
    sol=new.sol; score=new.score
    iter.score=c(iter.score,score)
    top.score=c(top.score,score)
  }else{
    iter.score=c(iter.score,new.score)
    top.score=c(top.score,score)
  }
  cat(iter, "..", sep=""); iter = iter + 1
  if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
     ↪ ==0){stop=1}}
```

```
    }
  }
}else{
  subpop=as.character(subpop)
  if(method=="MSPE"){
    ## pop untarget MSPE
    if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}
    pops = names(table(subpop))
    pop.ratio = round(n*as.numeric(table(subpop[cand]))/sum(as.numeric(table
        ↪ (subpop[cand]))))
    stop=0
    while(stop==0){
      if(sum(pop.ratio)>n){
        pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
            ↪ ratio==max(pop.ratio))[1]]-1
      }else if(sum(pop.ratio)<n){
        pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
            ↪ ratio==max(pop.ratio))[1]]+1
      }else{
        stop=1
      }
    }
    sol=c()
    for(i in 1:length(pops)){
```

```
  sol = c(sol, sample(cand[subpop[cand]==pops[i]],pop.ratio[i]))
}
score=mspe(geno[sol,], geno[cand[!cand%in%sol],])
iter.score=score; top.score=score
stop=0; iter=1
while(stop==0){
  new.sol=sol
  p = sample(length(pops),1,prob=pop.ratio)
  new.sol[sample(which(subpop[sol]==pops[p]),1)]=cand[!cand%in%new.sol][
     ↪ sample(which(subpop[cand[!cand%in%new.sol]]==pops[p]),1)]
  new.score=mspe(geno[new.sol,],geno[cand[!cand%in%new.sol],])
  if(new.score<score){
    sol=new.sol; score=new.score
    iter.score=c(iter.score,score)
    top.score=c(top.score,score)
  }else{
    iter.score=c(iter.score,new.score)
    top.score=c(top.score,score)
  }
  cat(iter, "..", sep=""); iter = iter + 1
  if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
     ↪ ==0){stop=1}}

}
```

```
}else if(method=="rScore"){

  ## pop untarget r-score

  if(is.null(min.iter)){min.iter=round((Nc-n)*n+1)}

  pops = names(table(subpop))

  pop.ratio = round(n*as.numeric(table(subpop[cand]))/sum(as.numeric(table
     ↪ (subpop[cand]))))

  stop=0

  while(stop==0){

    if(sum(pop.ratio)>n){

      pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
         ↪ ratio==max(pop.ratio))[1]]-1

    }else if(sum(pop.ratio)<n){

      pop.ratio[which(pop.ratio==max(pop.ratio))[1]]=pop.ratio[which(pop.
         ↪ ratio==max(pop.ratio))[1]]+1

    }else{

      stop=1

    }

  }

  sol=c()

  for(i in 1:length(pops)){

    sol = c(sol, sample(cand[subpop[cand]==pops[i]],pop.ratio[i]))

  }

  score=r_score(geno[sol,], geno[cand[!cand%in%sol],])

  iter.score=score; top.score=score
```

```
    stop=0; iter=1
    while(stop==0){
      new.sol=sol
      p = sample(length(pops),1,prob=pop.ratio)
      new.sol[sample(which(subpop[sol]==pops[p]),1)]=cand[!cand%in%new.sol][
          ↪ sample(which(subpop[cand[!cand%in%new.sol]]==pops[p]),1)]
      new.score=r_score(geno[new.sol,],geno[cand[!cand%in%new.sol],])
      if(new.score>score){
        sol=new.sol; score=new.score
        iter.score=c(iter.score,score)
        top.score=c(top.score,score)
      }else{
        iter.score=c(iter.score,new.score)
        top.score=c(top.score,score)
      }
      cat(iter, "..", sep=""); iter = iter + 1
      if(iter > min.iter){if(abs(top.score[iter]-top.score[iter-(Nc-n)*n])
          ↪ ==0){stop=1}}
    }
  }
}
ret = list(
  OPTtrain=as.numeric(sort(sol)),
  TOPscore=as.numeric(top.score[-1]),
```

```
    ITERscore=as.numeric(iter.score[-1])
  )
  return(ret)
}
```

A.3 r-score and mspe-score

```
library(compiler)

r_score<-function(x,x0){

  n0=nrow(x0);n=nrow(x);I=diag(1,n0,n0);J=matrix(1/n0,n0,n0);

  A=t(x)%*%solve(x%*%t(x)+diag(1,n,n));X0AX=x0%*%A%*%x

  q1=(n0+1)+sum(diag(t(x0)%*%(I-J)%*%x0))

  q2=sum(diag(t(A)%*%t(x0)%*%(I+J)%*%x0%*%A))+sum(diag(t(X0AX)%*%(I-J)%*%X0AX
     ↪ ))

  q12=sum(diag(t(x0)%*%(I-J)%*%X0AX))

  return(q12/sqrt(q1*q2))

}

mspe<-function(x,x0){

  n0=nrow(x0);n=nrow(x);I=diag(1,n0,n0);J=matrix(1/n0,n0,n0);

  A=t(x)%*%solve(x%*%t(x)+diag(1,n,n));X0AX=x0%*%A%*%x

  return(1+(1/n0)*(sum(diag(x0%*%A%*%t(A)%*%t(x0)))+sum(diag(t(x0-X0AX)%*%(x0
     ↪ -X0AX)))))

}

r_score<-cmpfun(r_score);mspe<-cmpfun(mspe)

library(foreach);library(doParallel)

cpu.cores<-detectCores()

cl=makeCluster(cpu.cores)

registerDoParallel(cl)
```