# Progress report: Detecting interaction with unknown environmental covariate

Ziang Zhang

15/10/2020

## Contents



# 1 The Underlying Model:

For binary response variable, it is often assumed that the response variable $y$ conditioning on the regressors $G, Z$ come from a latent model such that:

$$\begin{aligned} Y^* &= \beta_0 + \beta_G G + \beta_Z Z + \epsilon \\ Y &= I\{Y^* > 0\} \end{aligned} \tag{1}$$

The unobserved latent variable $Y^*$ determines whether the observed response variable $Y$ is 0 or 1. The error term $\epsilon$ in $Y^*$ needs to have a completely known distribution, which can be $\mathrm{N}(0,1)$ for the model of $Y|G,Z$ to become a probit model, or a logistic distribution with mean 0 and variance 3.28 for the model of $Y|G,Z$ to become a logistic regression model.

Here the regressor $G$ represents the allele of interest, and the regressor $Z$ is any regressor that can be non-genetic. For now on, we will assume the model is probit for simplicity, unless otherwise indicated. For probit regression model, we can express $\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big)$ as:

$$\begin{aligned}\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big) &= \frac{\mathrm{E}(Y^*|G,Z)}{\sqrt{\mathrm{Var}(Y^*|G,Z)}} \\ &= \beta_0 + \beta_G G + \beta_Z Z\end{aligned} \tag{2}$$

Where $\Phi(.)$ denote the CDF function of standard normal distribution.

Similarly, we can have a Genotypic Model defined as:

$$\begin{aligned}Y^* &= \beta_0 + \beta_{G1} I(G=1) + \beta_{G2} I(G=2) + \beta_Z Z + \epsilon \\ Y &= I\{Y^* > 0\}\end{aligned} \tag{3}$$

Therefore, the regression model can be written as:

$$\begin{aligned}\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big) &= \frac{\mathrm{E}(Y^*|G,Z)}{\sqrt{\mathrm{Var}(Y^*|G,Z)}} \\ &= \beta_0 + \beta_{G1} I(G=1) + +\beta_{G2} I(G=2) + \beta_Z Z\end{aligned} \tag{4}$$

The Genotypic Model has higher degree of freedom than the additive model due to the extra regression parameter.

It can be noticed that the variance parameter of $\epsilon$ is fixed to a specific constant to avoid the problem of identifiability, since we cannot identify both $\beta$ and $\mathrm{Var}(\epsilon)$ in $\frac{\mathrm{E}(Y^*|G,Z)}{\sqrt{\mathrm{Var}(Y^*|G,Z)}}$.

## 1.1 When the true model does contain gene-environment interaction

Assume for simplicity that $E_i$ the environmental variable has a normal distribution with mean $\mu_E$ and variance $\sigma_E^2$, and suppose that the true underlying model is:

$$\begin{aligned}Y^* &= \beta_0 + \beta_G G + \beta_Z Z + \beta_E E + \beta_{G\times E} G \times E + \epsilon \\ Y &= I\{Y^* > 0\} \\ \epsilon &\sim \mathrm{N}(0,1)\end{aligned} \tag{5}$$

Furthermore, we can compute that:

$$\begin{aligned}\mathrm{E}(Y^*|G,Z) &= \beta_0 + \beta_E \mu_E + (\beta_G + \beta_{G\times E}\mu_E)G + \beta_Z Z \\ \mathrm{Var}(Y^*|G,Z) &= (\beta_{G\times E} G)^2 \sigma_E^2 + \beta_E^2 \sigma_E^2 + 1 \\ Y^*|G,Z &\sim \mathrm{N}\Big(\beta_0 + \beta_E\mu_E + (\beta_G + \beta_{G\times E}\mu_E)G + \beta_Z Z, (\beta_{G\times E}G)^2\sigma_E^2 + \beta_E^2\sigma_E^2 + 1\Big)\end{aligned} \tag{6}$$

That implies that the probability we get a case for different levels of $G$ and $Z$ will be:

$$\begin{aligned}\mathrm{P}(Y=1|G,Z) &= \mathrm{P}(Y^*>0|G,Z) \\ &= \mathrm{P}\Big(\frac{Y^* - \mathrm{E}(Y^*|G,Z)}{\sqrt{\mathrm{Var}(Y^*|G,Z)}} > \frac{-\mathrm{E}(Y^*|G,Z)}{\sqrt{\mathrm{Var}(Y^*|G,Z)}}\Big) \\ &= \Phi\Big(\frac{\mathrm{E}(Y^*|G,Z)}{\sqrt{\mathrm{Var}(Y^*|G,Z)}}\Big)\end{aligned} \tag{7}$$

Therefore, applying the inverse CDF on both sides, we get

$$\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big) = \frac{\beta_0 + \beta_E\mu_E + (\beta_G + \beta_{G\times E}\mu_E)G + \beta_Z Z}{\sqrt{(\beta^2_{G\times E}G^2\sigma^2_E + \beta^2_E\sigma^2_E + 1)}}$$

This is only a linear function of $G$ when the interaction parameter $\beta_{G\times E} = 0$, and the slope of Z is constant across different genes only when $\beta_{G\times E}$ is zero.

1. If the true underlying model also contains another regressor $W$ but $W$ is uncorrelated with $G$ for example. Then even though ignoring that regressor scales all the regression parameters by an unknown constant (since now $\epsilon$ does not follow standard normal), but $\Phi^{-1}(\mathrm{P}(Y=1|G,Z))$ will still be a linear function of $G$. So detecting based on the linearity of $\Phi^{-1}\mathrm{P}$ will not be affected by omitted exogenous regressors.

2. Even though $\Phi^{-1}\mathrm{P}$ is not linear in G as shown above, this model can still be rewritten as a valid genotypic probit regression model by regressing on $I(G=1)$ and $I(G=2)$. The genotypic model in this case will have six regression parameters in this case (two for effects of G, three for slopes of Z, and one intercept).

3. The reason we used probit model instead of logistic model here is that assuming $E$ follows normal distribution, $Y^*|G,Z$ will still be normal if we omit the interaction term, since linear combination of normal is normal. But assuming $E$ follows logistic distribution does not imply that $Y^*|G,Z$ will be logistically distributed as logistic distribution is not closed under linear combination. However, logistic regression model for $Y|G,Z$ implies that $Y^*|G,Z$ must follow logistic distribution. In other words, probit regression model with omitted covariate will still be a probit regression model, just with different regression parameters. Based on the literature, it seems like probit model and logistic model have really closed results in real applications.

If the model is Genotypic instead:

$$Y^*_i = \beta_0 + \beta_{G1}I(G_i=1) + \beta_{G2}I(G_i=2) + \beta_Z Z_i + \beta_E E_i + \beta_{G1E}I(G_i=1)\times E_i + \beta_{G2E}I(G_i=2)\times E_i + \epsilon_i \tag{8}$$

then we can derive the following:

$$\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big) = \frac{\beta_0 + \beta_E\mu_E + (\beta_{G1} + \beta_{G1E}\mu_E)I(G=1) + (\beta_{G2} + \beta_{G2E}\mu_E)I(G=2) + \beta_Z Z}{\sqrt{(\beta^2_{G1E}I(G=1)\sigma^2_E + \beta^2_{G2E}I(G=2)\sigma^2_E + \beta^2_E\sigma^2_E + 1)}}$$

In this case, the model will still be linear in $I(G=1), I(G=2)$ because of the extra parameter, just with different regression parameters. But the slope of Z will continue to differ between different Genetic types unless the there are no interaction effects $\beta_{G1E}$ and $\beta_{G2E}$.

# 2 Compare with the traditional method for normal data:

When the response variable $Y$ is normally distributed, the traditional method to test for potential GxE interaction is based on the test of heteroskedasticity of $Y$ for different genotypes. The idea is that if:

$$Y = \beta_0 + \beta_1 G + \gamma_1 G\times E + \epsilon, \text{ where } \epsilon \sim N(0,\sigma^2)$$

then we can see that $Var(Y|G) = \gamma_1^2 G^2 Var(E) + \epsilon$ if we assume that $E$ is uncorrelated with $G$. In other words, the conditional variance of $Y$ is a quadratic function of $G$.

If the response data is binary, and we assume that the true underlying model is a probit model as in the previous section, then the situation for the latent variable $Y^*$ is basically the same, except that our $\epsilon$ follows

a specific standard normal distribution. It would suffice if we can detect the heteroskedasticity of our latent variable $Y^*$, but unlike the case of normal data, these normally distributed $Y^*$ are not directly observable. For binary data, the only observable quantities related to $Y^*$ is the binary variable $Y = I(Y^* > 0)$.

Therefore, as described in the previous section, since $Y|G \sim Ber(p = \Phi(\frac{E(Y^*|G)}{\sqrt{Var(Y^*|G)}}))$, the conditional variance of $Y^*$ is not identifiable from the data. However, this ratio of $\frac{E(Y^*|G)}{\sqrt{Var(Y^*|G)}}$ can still be identified, and the numerator is always a linear function of $G$ regardless of whether there is GxE interaction.

It worth notice that we cannot use infer the GxE interaction from $Var(Y|G)$ if $Y$ is binary data, since unlike normal variable, Bernoulli variable's variance is directly specified through their means. In this case, each $Y$ will be Bernoulli variable with fixed variance being $\Phi(\frac{E(Y^*|G)}{\sqrt{Var(Y^*|G)}})\Phi(-\frac{E(Y^*|G)}{\sqrt{Var(Y^*|G)}})$. This also implies that we cannot identify any overdispersion parameter if we are using individual-level probit model.

If we choose to work with the group-level data (i.e. binomial data for number of cases of each genotype), then we can estimate the overdispersion parameter. However, that overdispersion parameter quantifies how individuals in the dataset are correlated (i.e. how is the distribution of number of cases of each genotype different from a binomial distribution), which will not be our primary interest here.

# 3 Method for Additive Model:

In this section, I will present a method for the detection of interaction effect when the true model is additive.

## 3.1 Testing of Linearity:

Recall that when the model is additive, then:

$$\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big) = \frac{\beta_0+\beta_E\mu_E+(\beta_G+\beta_{G\times E}\mu_E)G+\beta_Z Z}{\sqrt{(\beta^2_{G\times E}G^2\sigma^2_E+\beta^2_E\sigma^2_E+1)}}$$

This method relies on checking linearity of $\Phi^{-1}(P)$, so the test statistics will also focus on the detection of potential deviation from linearity (or additivity). Note that the above 1 degree of freedom regression model, although it is not linear in $G$, it can be rewritten as a valid 2 degree of freedom genotypic model that is linear in $I(G=1), I(G=2)$:

$$\begin{aligned}\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big) &= \frac{\beta_0+\beta_E\mu_E+(\beta_G+\beta_{G\times E}\mu_E)G+\beta_Z Z}{\sqrt{(\beta^2_{G\times E}G^2\sigma^2_E+\beta^2_E\sigma^2_E+1)}}\\ &= \gamma_0+\gamma_1 I(G=1)+\gamma_2 I(G=2)+\gamma_{Z1G}I(G=1)*Z+\gamma_{Z2G}I(G=2)*Z+\gamma_Z Z\end{aligned} \tag{9}$$

Here, we can compute that:

$$\begin{aligned}\gamma_0 &= \frac{\beta_0+\beta_E\mu_E}{\sqrt{\beta^2_E\sigma^2_E+1}}\\ \gamma_1 &= \frac{\beta_0+\beta_E\mu_E+\beta_G+\beta_{G\times E}\mu_E}{\sqrt{\beta^2_{G\times E}\sigma^2_E+\beta^2_E\sigma^2_E+1}}-\gamma_0\\ \gamma_2 &= \frac{\beta_0+\beta_E\mu_E+2(\beta_G+\beta_{G\times E}\mu_E)}{\sqrt{4\beta^2_{G\times E}\sigma^2_E+\beta^2_E\sigma^2_E+1}}-\gamma_0\end{aligned} \tag{10}$$

Therefore, we can conclude that if $\beta_{G\times E}=0$, then $\beta_G = 2\gamma_1 = \gamma_2$ must hold. If we further have the information on the covariate $Z$, then we can increase the power of our detection by also testing on $\gamma_{Z1G} = \gamma_{Z2G} = 0$ (equal slopes of Z across genotypes).

### 3.1.1 Wald Test Statistics:

To test the null hypothesis of $\beta_{G\times E}=0$, we can use the following steps:

**1. Rewrite the additive regression model:** We first rewrite the additive regression model as a genotypic model, i.e:

$$\Phi^{-1}\Big(\mathrm{P}(Y=1|G,Z)\Big) = \gamma_0+\gamma_1 I(G=1)+\gamma_2 I(G=2)+\gamma_{Z1G}I(G=1)*Z+\gamma_{Z2G}I(G=2)*Z+\gamma_Z Z$$

Now, the regression parameters in this model may not carry very meaningful interpretations due to the potential presence of $\beta_{G\times E}$. However, we can use them to detect the presence of missing interaction effect by either testing $H_0: 2\gamma_1=\gamma_2$ or $H_0: 2\gamma_1=\gamma_2,\ \gamma_{Z1G}=\gamma_{Z2G}=0$, depending on whether the information of $Z$ is available, and whether $Z$ is ordinal.

**2. Wald Test on linearity:** To test $H_0: 2\gamma_1=\gamma_2$ or $H_0: 2\gamma_1=\gamma_2,\ \gamma_{Z1G}=\gamma_{Z2G}=0$, we can simply do a Wald test on the genotypic working model that we considered above. It turns out that this Wald test

statistic $T$ can also be equivalently derived through an application of Delta method on sample proportion, or as a two-stage linear regression.

If we have information about the covariate $Z$ in the regression model, and $Z$ is an ordinal variable with additive effect, then the power of our test will be augmented if we test $H_0 : 2\gamma_1 = \gamma_2,\ \gamma_{Z1G} = \gamma_{Z2G} = 0$ in our Wald test. We can also only test $H_0 : \gamma_{Z1G} = \gamma_{Z2G} = 0$, this makes our test robust to the case that G's effect is actually non-additive. However, we then need to be more careful to make sure that $Z$ indeed has no interaction with G or E in the **true** model.

# 4 Method for Genotypic Model:

## 4.1 Auxiliary variable method:

Recall for a Genotypic Model with interaction like below:

$$Y_i^* = \beta_0 + \beta_{G1} I(G_i = 1) + \beta_{G2} I(G_i = 2) + \beta_Z Z_i + \beta_E E_i + \beta_{G1E} I(G_i = 1) \times E_i + \beta_{G2E} I(G_i = 2) \times E_i + \epsilon_i \tag{11}$$

We can derive that:

$$\begin{aligned} \Phi^{-1}\Big(\mathrm{P}(Y = 1|G, Z)\Big) &= \frac{\beta_0 + \beta_E\mu_E + (\beta_{G1} + \beta_{G1E}\mu_E)I(G = 1) + (\beta_{G2} + \beta_{G2E}\mu_E)I(G = 2) + \beta_Z Z}{\sqrt{(\beta_{G1E}^2 I(G = 1)\sigma_E^2 + \beta_{G2E}^2 I(G = 2)\sigma_E^2 + \beta_E^2\sigma_E^2 + 1)}} \\ &= \gamma_0 + \gamma_1 I(G = 1) + \gamma_2 I(G = 2) + \gamma_Z Z + \gamma_{Z1G} I(G = 1) \times Z + \gamma_{Z2G} I(G = 2) \times Z \end{aligned} \tag{12}$$

where the new parameters $\gamma_{Z1G}$ and $\gamma_{Z2G}$ will be defined as:

$$\gamma_{Z1G} = \frac{\beta_Z}{\sqrt{\beta_E^2\sigma_E^2 + \beta_{G1E}^2\sigma_E^2 + 1}} - \frac{\beta_Z}{\sqrt{\beta_E^2\sigma_E^2 + 1}}$$

and:

$$\gamma_{Z2G} = \frac{\beta_Z}{\sqrt{\beta_E^2\sigma_E^2 + \beta_{G2E}^2\sigma_E^2 + 1}} - \frac{\beta_Z}{\sqrt{\beta_E^2\sigma_E^2 + 1}}$$

In other words, a Genotypic Model with an missing interaction can still be written as a linear function of these two indicator functions of G because of the extra regression parameter. However, ignoring this environment to gene interaction will create an artificial interaction between gene and the covariate Z. Since the interaction effects are zero if and only if the covariate Z has constant slopes across different genotypes, we can test the environmental interaction by testing the null hypothesis $H_0 : \gamma_{Z1G} = \gamma_{Z2G} = 0$, using either wald test, likelihood ratio test or score test.

The key in this method is to test the equal slopes of the auxiliary variable $Z$. In order for this method to work, we need the following assumption:

1. The auxiliary variable $Z_i$ is assumed to have no interaction effect with G in the model conditional on G, Z and E.
2. The auxiliary variable $Z_i$ is also assumed to have no interaction effect with E in the model conditional on G, Z and E.

**Main concern about genotypic model:** When we use the above method to test for the presence of GE interaction in a genotypic model, the power of our test is highly dependent on the true values for $\beta_Z$ and $\beta_{G\times E}$. Since $\gamma_{Z1G}$ and $\gamma_{Z2G}$ will eventually converge to 0 as $\beta_Z$ gets closer to 0, or as $\beta_{G\times E}$ gets closer to 0. In other words, our method has very low power when:

$$\frac{\beta_Z}{\sqrt{\beta_E^2\sigma_E^2 + \beta_{G1E}^2\sigma_E^2 + 1}} \approx \frac{\beta_Z}{\sqrt{\beta_E^2\sigma_E^2 + \beta_{G2E}^2\sigma_E^2 + 1}} \approx \frac{\beta_Z}{\sqrt{\beta_E^2\sigma_E^2 + 1}}$$

# 5 GWAS Implementation on the 1000 Genome Project Data:

## 5.1 Simulation to check type I error rate:

### Classical Testing for Main effect

In this section, we will conduct a GWAS on the 1kGP dataset. The cleaned set of data has 1736 independent individuals and around 2 millions SNPs. However, since our analysis will be more sensitive to the low genotypic frequency of each gene, we will further have some additional quality control to make sure the genotypic frequency is high enough for each category. We further filtered SNPs in this dataset with threshold on MAF being 0.05, and keep only the SNPs on autosomes. After this additional QC procedure, there are 1749 individuals with 231879 SNPs in our dataset.

```
### Read in data:
path <- "D:/gwas-practice/indep_QC.bed"
tmpfile  <- tempfile()
snp_readBed(path, backingfile = tmpfile)
```

```
## [1] "C:\\Users\\aguer\\AppData\\Local\\Temp\\RtmpU32FqQ\\file24a077d35c45.rds"
```

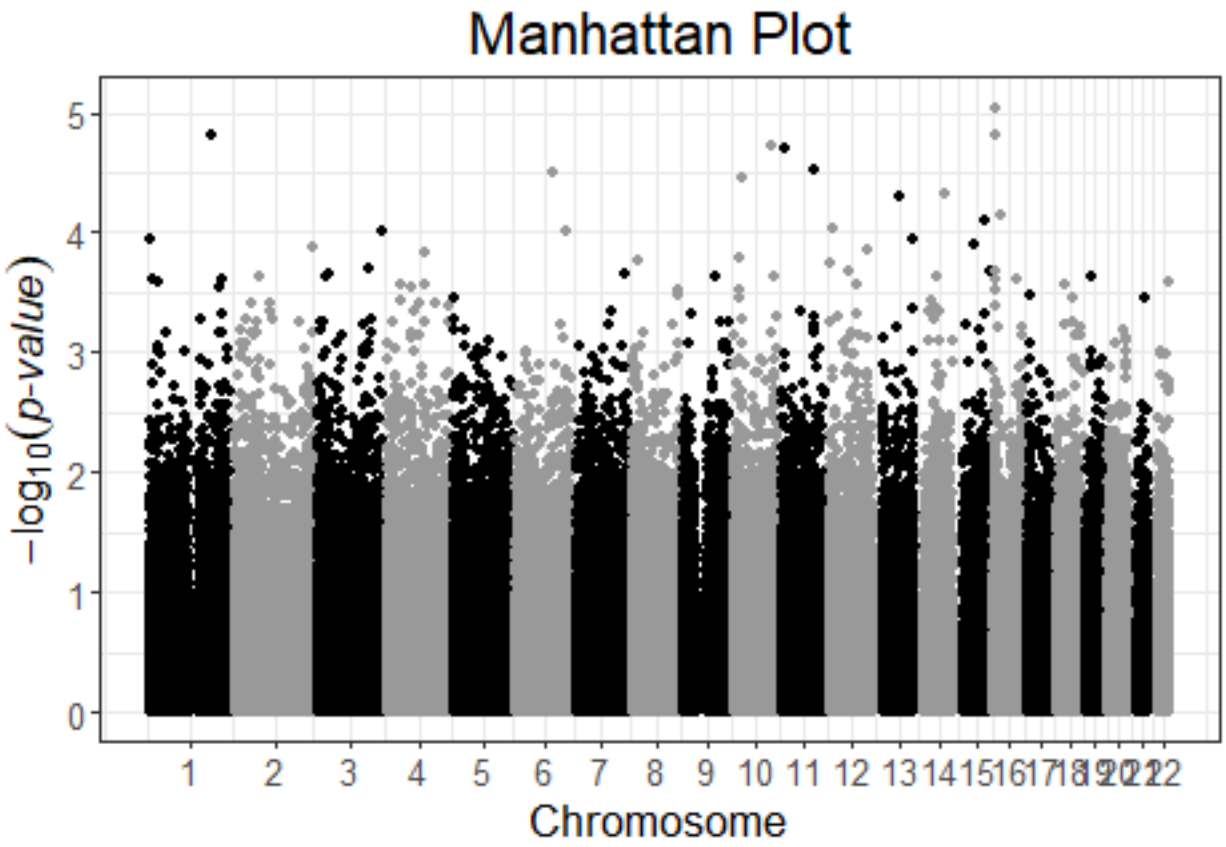
```
obj.bigSNP <- snp_attach(paste0(tmpfile , ".rds"))

G   <- obj.bigSNP$genotypes
CHR <- obj.bigSNP$map$chromosome
POS <- obj.bigSNP$map$physical.pos

# Check some counts for the 10 first SNPs
big_counts(G, ind.col = 1:10)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## 0     1507  635 1046 1399 1556 1410 1488 1394 1426  1314
## 1      229  774  572  308  172  293  256  337  289   394
## 2       13  340  131   42   21   46    5   18   34    41
## <NA>     0    0    0    0    0    0    0    0    0     0
```

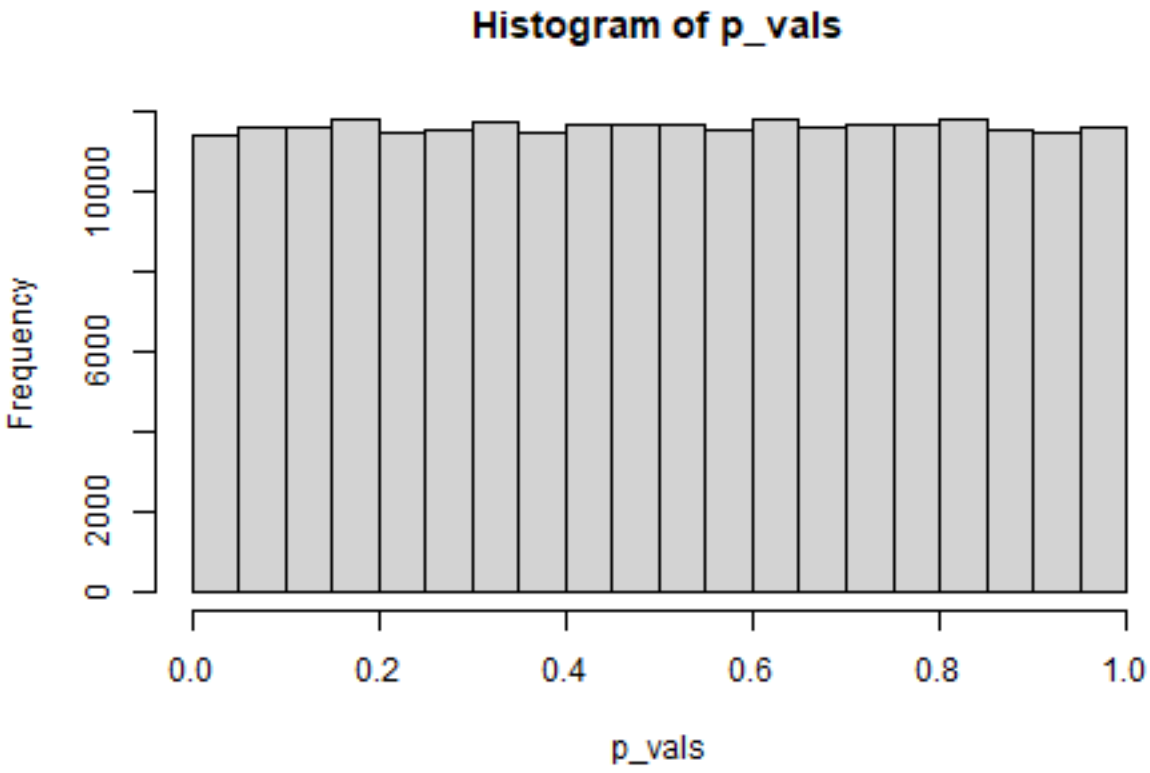
We randomly assigned an individual to case or control, and assume the disease prevalence is 0.3. Firstly, we conduct the classical logistic regression to test the main effects. Because of the random assignment, the p-values should have similar behavior as Unif$[0, 1]$. The results are summarized into histogram, QQ plot and Manhattan plot at below:

```
### Randomly generate case/control data under the null hypothesis
set.seed(12345,sample.kind="Rounding")
case <- rbinom(nrow(G),size = 1,prob = 0.3)
obj.bigSNP$fam$case <- case


### Testing for main effects using Logistic regression:
obj.gwas <- big_univLogReg(G, y01.train = case,
                           ncores = 6L, maxiter = 100)


### QQ plot, Manhattan plot and Genomic Control
snp_qq(gwas = obj.gwas)
```


```
snp_manhattan(gwas = obj.gwas, infos.chr = CHR, infos.pos = POS)
```




```
### Histogram of p-values:
p_vals <- 2*pnorm(-abs(obj.gwas$score))
hist(p_vals, breaks = 30)
```




Based on the plots above, we can conclude that p-value's distribution is very close to the uniform distribution, which is what we expect.

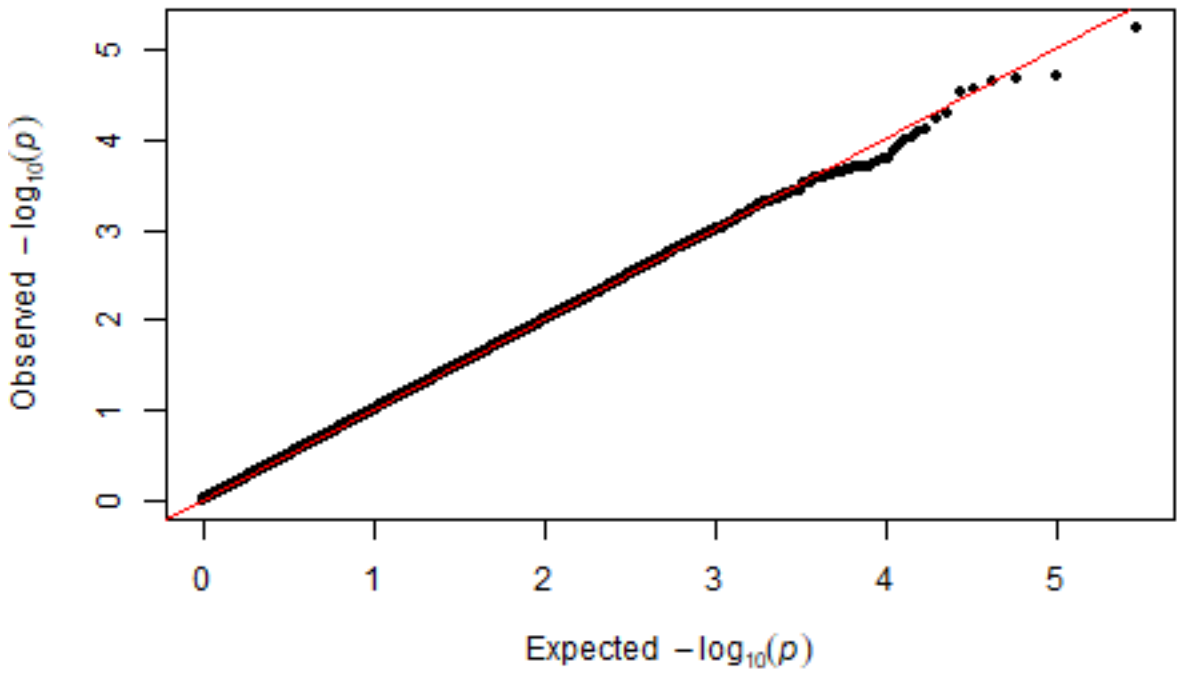
**Proposed Method: Assuming additive effect**

Then, we will apply our proposed methodology for detection of missing interaction, assuming the true effect of gene is additive. Again, we expect to see the distribution of p-values to be close to uniform.

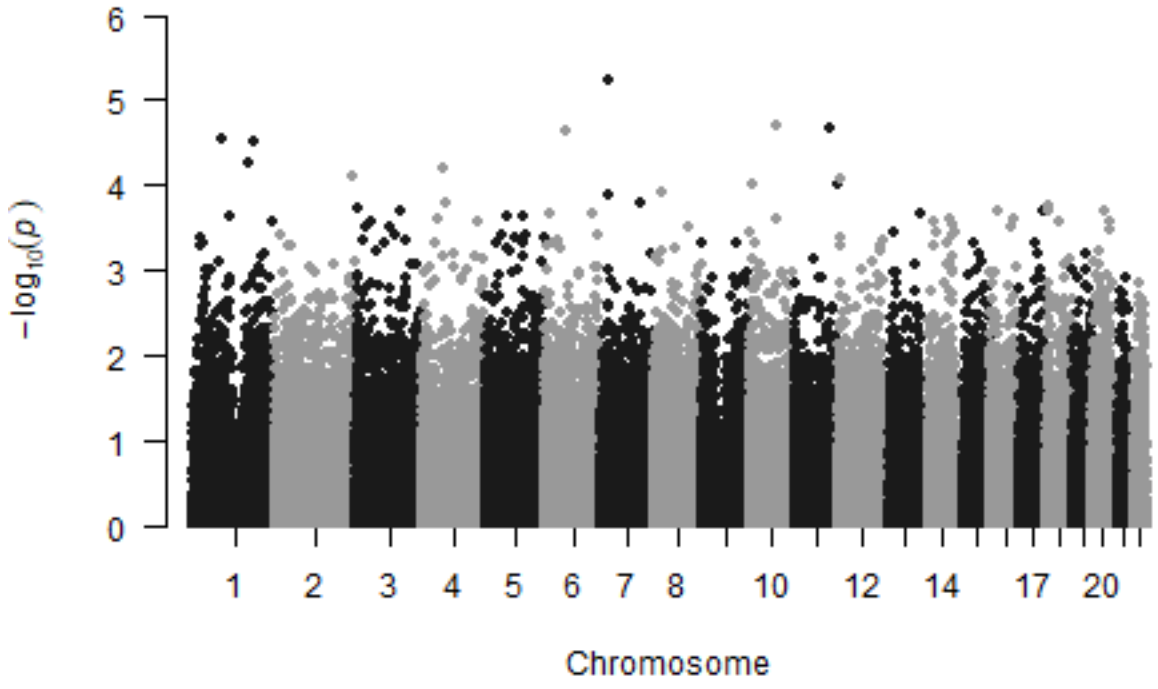
```
load("D:/gwas-practice/additive_testing.Rdata")

## View of the result
head(result_P1)
```

```
## # A tibble: 6 x 5
##   SNP            CHR      BP    stats      P
##   <chr>        <int>   <int>    <dbl>  <dbl>
## 1 SNP1-840643      1  850780 0.000836 0.977
## 2 SNP1-842827      1  852964 0.0175   0.895
## 3 SNP1-908480      1  918617 2.39     0.122
## 4 rs3766192        1 1017197 0.421    0.516
## 5 rs3766191        1 1017587 4.33     0.0375
## 6 SNP1-1013008     1 1023145 1.41     0.236
```

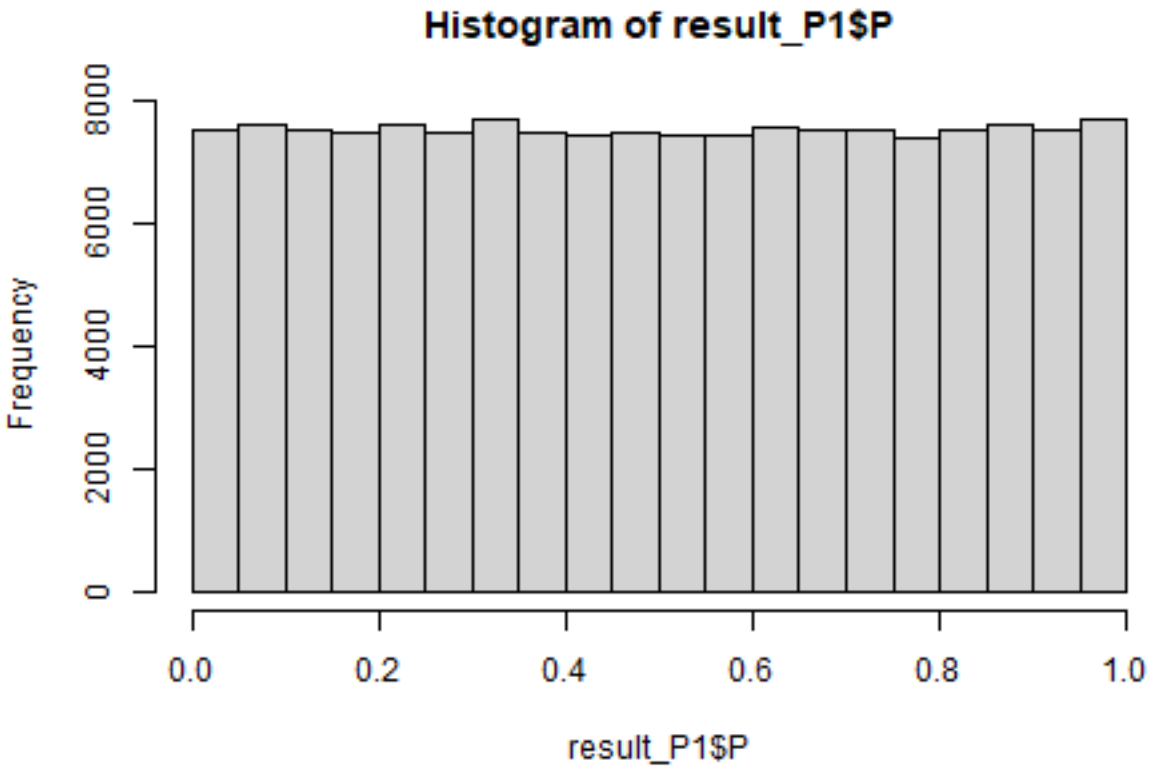
```
## QQ plot of p-values
qq(na.omit(result_P1$P))
```




```
## Manhattan plot of p-values:
manhattan(na.omit(result_P1), suggestiveline = F,genomewideline = -log(0.05/nrow(G)))
```




```
## Histogram of p-values:
hist(result_P1$P, breaks = 30)
```


Histogram of result_P1$P
Frequency
0
2000
4000
6000
8000
0.0
0.2
0.4
0.6
0.8
1.0
result_P1$P

## 5.2 Simulation to check power:

In this section, we consider to check the power of our proposed approach for detecting GxE interaction via two different simulation settings. We first compare the power of the proposed method with the power of the oracle method when information of $E$ can be collected accurately, and then compare their powers when there is a measurement error in $E$. For all of these settings, we assume that there is only one casual gene in the model, and that true casual gene has both main effect and interaction effect with $E$.

### 5.2.1 Power of the proposed method:

In this case, we consider the following model:

$$Y^* = \beta_0 + \beta_G G_k + \beta_E E + \gamma G_k E + \epsilon$$

Here $G_k$ is the casual gene we randomly sampled from the genome, with the constraint that $G_k$ should have MAF at least 0.3. The values of our parameters are:

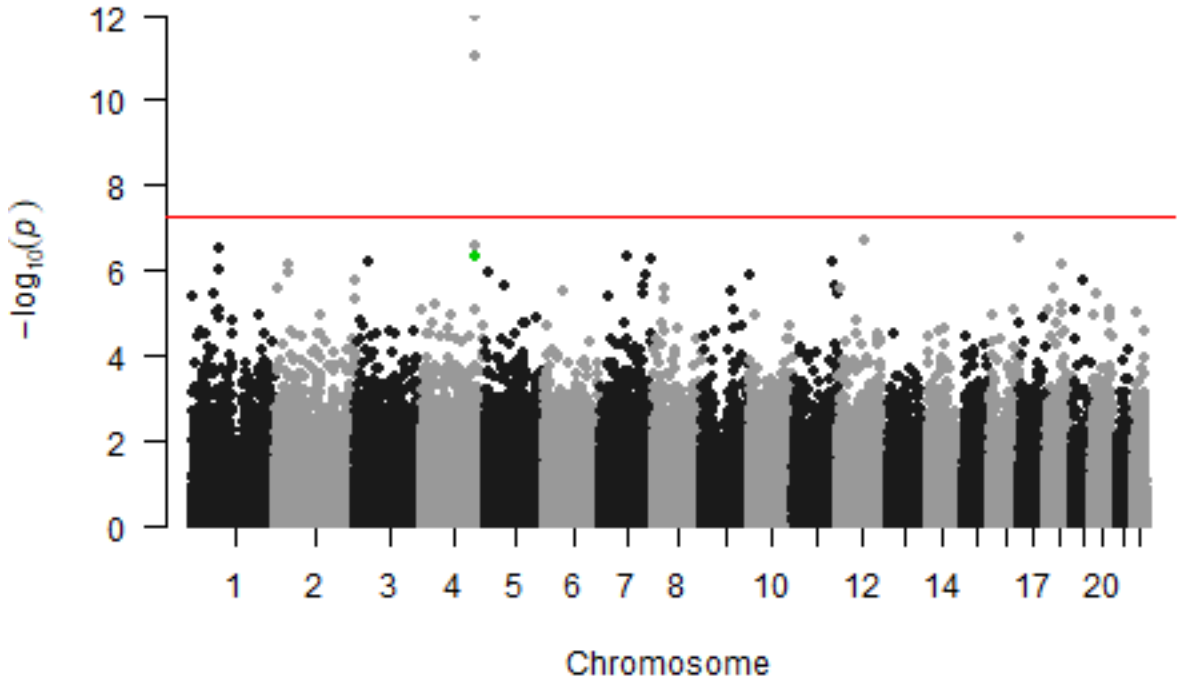
- $\beta_0 = -1, \beta_G = 0.8, \beta_E = 0.3, \gamma = 0.5$
- $E \sim N(0, 1)$
- $\epsilon \sim N(0, 0.5)$

```
################ Power of the proposed method for interaction: With one casual genes

### Again, first take account of the linkage disequilibrium problem:
set.seed(123,sample.kind="Rounding")
indx <- 1:ncol(G)
G_using <- G[,indx]
CHR_using <- CHR[indx]
POS_using <- POS[indx]


### Randomly sample 1 genes:
p <- 1
### Need to make sure that all genotypes have enough frequencies in the selected genes:
freq_counts <- big_counts(G,ind.col = indx)
MAF <- snp_MAF(G,ind.col = indx)
Qualified <- freq_counts[3,] >= 200 & MAF>=0.3
POS_EFF <- sample(POS_using[Qualified], size = 1)


### Randomly sample 1/2 genes with strong effects and 1/2 genes with weak effects
b0 <- -1
bG <- 0.8
bint <- 0.8
bE <- 0.3


### Generate the underlying environment variable:
E <- rnorm(n = length(G_using[,1]), sd = 1)

### Generate the latent variable:
y_lat <- b0 + bG*G_using[,POS_using == POS_EFF] + bE*E + bint*E*G_using[,POS_using == POS_EFF] + rnorm(
case_new <- ifelse(y_lat > 0, 1, 0)
```

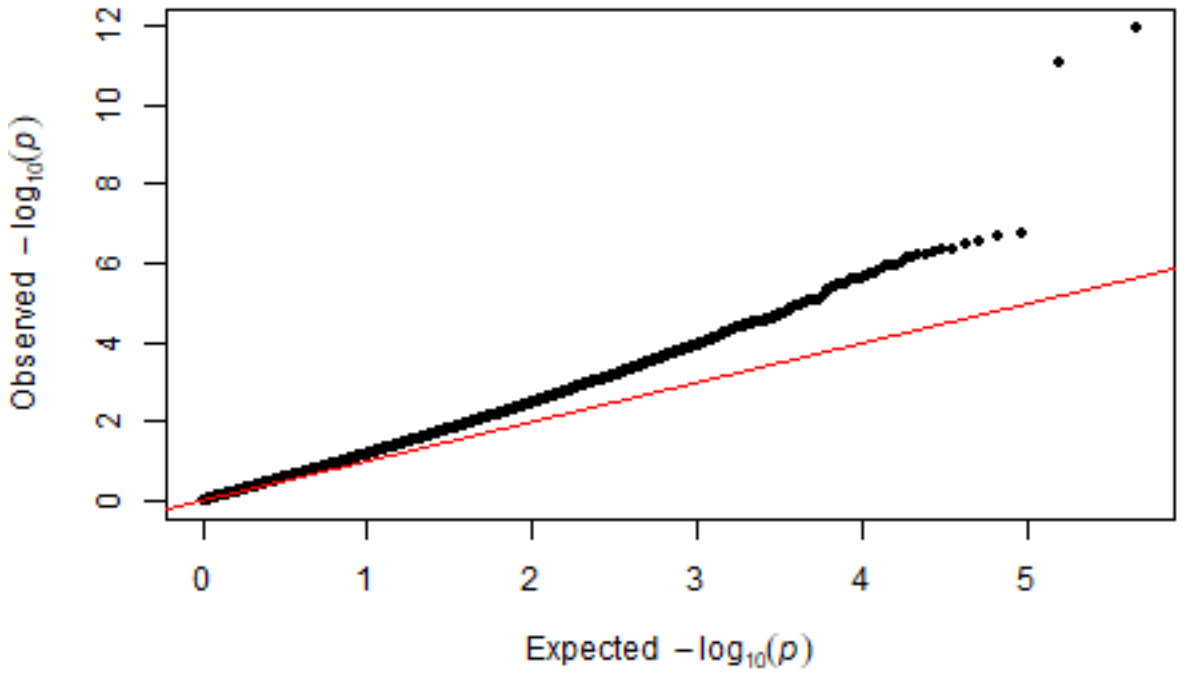
```r
### case_control ratio:
table(case_new)
```

```
## case_new
##    0    1
## 1268  481
```

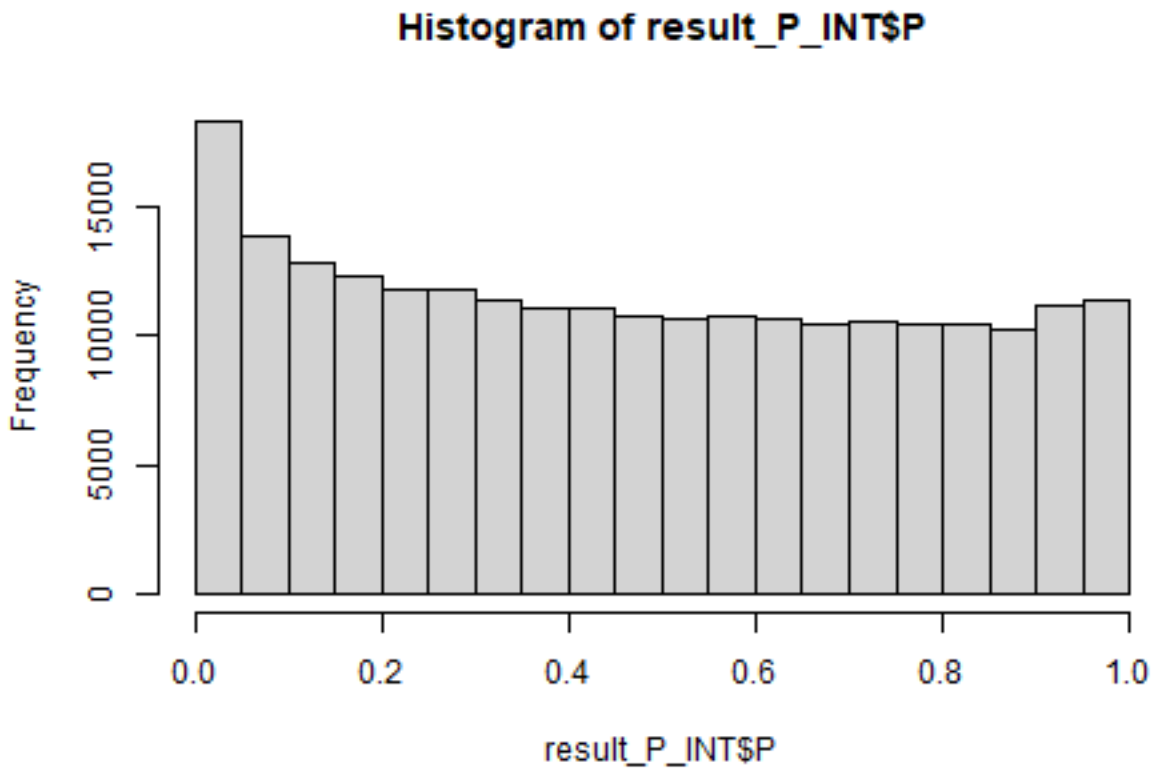
```r
#### Testing for interaction effect
### Assuming additive model: Testing for non-linearity
waldstats_P1 <- c()
p_vals_P1 <- c()

for (i in 1:ncol(G_using)) {
  Gi <- G_using[,i]

  if(length(unique(Gi)) != 3){
    waldstats_P1[i] <- 0
    p_vals_P1[i] <- -1
  }
  else{
    modi <- glm(case_new ~ factor(Gi), family = binomial(link = "probit"))
    waldstats_P1[i] <- as.numeric(wald.test(vcov(modi)[-1,-1], b = modi$coefficients[-1], H0 = matrix(0
    p_vals_P1[i] <- as.numeric(wald.test(vcov(modi)[-1,-1], b = modi$coefficients[-1], H0 = matrix(0,nr
  }
}

result_P_INT <- tibble(SNP = obj.bigSNP$map$marker.ID[indx], CHR = CHR_using, BP = POS_using , stats = w



### diagnostic plots:
manhattan(result_P_INT, highlight = result_P_INT$SNP[which(POS_using %in% POS_EFF)], suggestiveline = FA
```




```
qq(na.omit(result_P_INT)$P)
```




```
hist(result_P_INT$P, breaks = 30)
```




```
result_P_INT[result_P_INT$BP %in% POS_EFF, ]
```

```
## # A tibble: 1 x 5
##   SNP              CHR        BP stats        P
##   <chr>          <int>     <int> <dbl>    <dbl>
## 1 SNP4-164081321     4 163861871  50.7 1.07e-12
```

Here we can see that our proposed method works well to identify the true casual gene in this simulation example. We assumed a quite large interaction effect compared to the main effect of $E$, and we assume a very small variance for $\sigma_\epsilon^2$ to make sure that the casual gene will be easy to be found by our method, since our sample size is very small in this example.

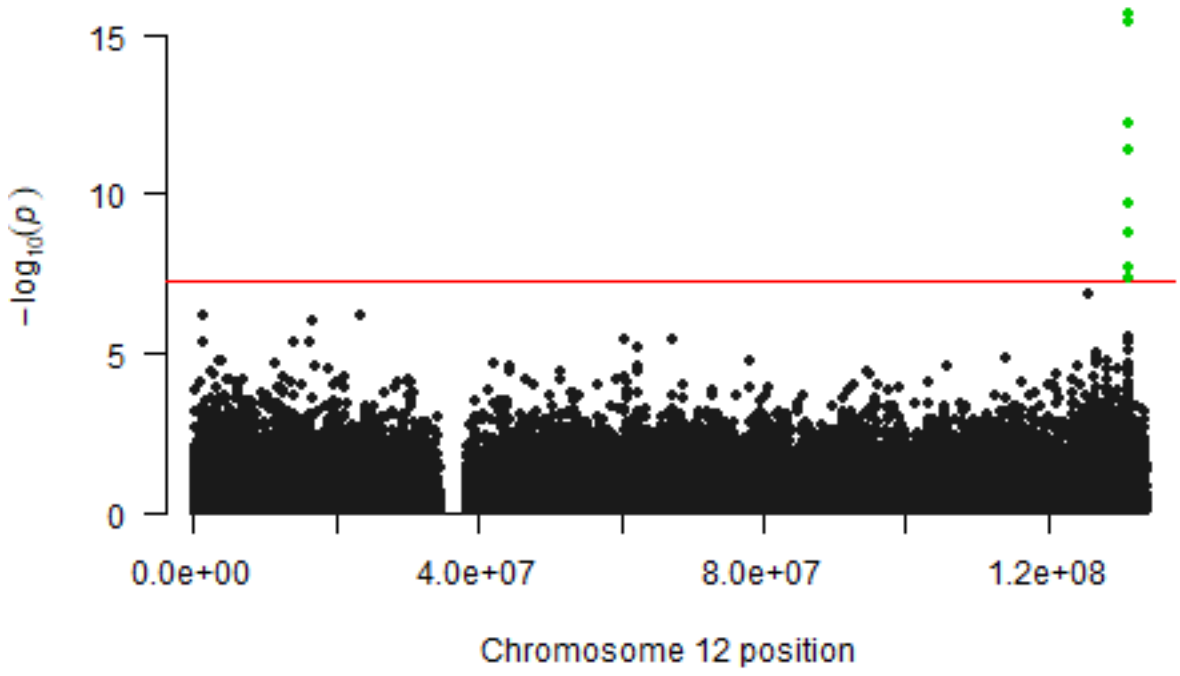
### 5.2.2 Aggregating p-values through k repetitions:

In order to better understand the power of our proposed method, we will perform the simulation study with only one casual gene and aggregate the result for K = 50 times:

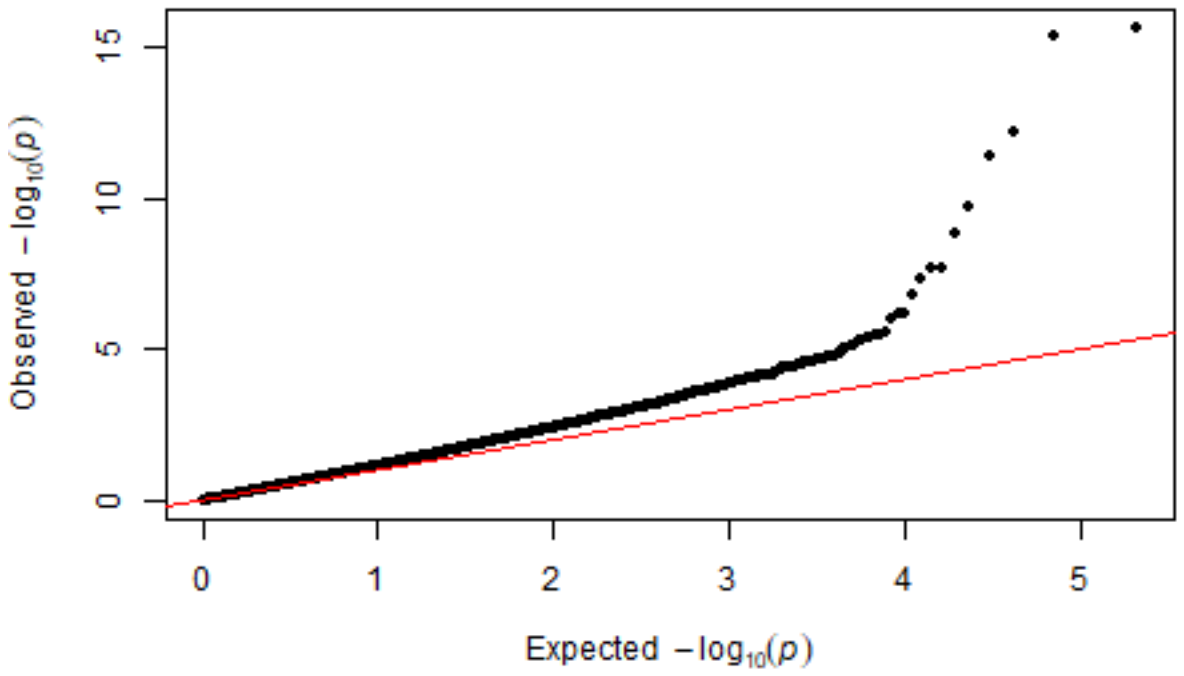
Here $G_k$ is the casual gene we randomly sampled from the genome, with the constraint that $G_k$ should have MAF at least 0.25. The values of our parameters are:

- $\beta_0 = -1, \beta_G = 0.8, \beta_E = 0.3, \gamma = 0.3$
- $E \sim N(0, 1)$
- $\epsilon \sim N(0, 0.5)$

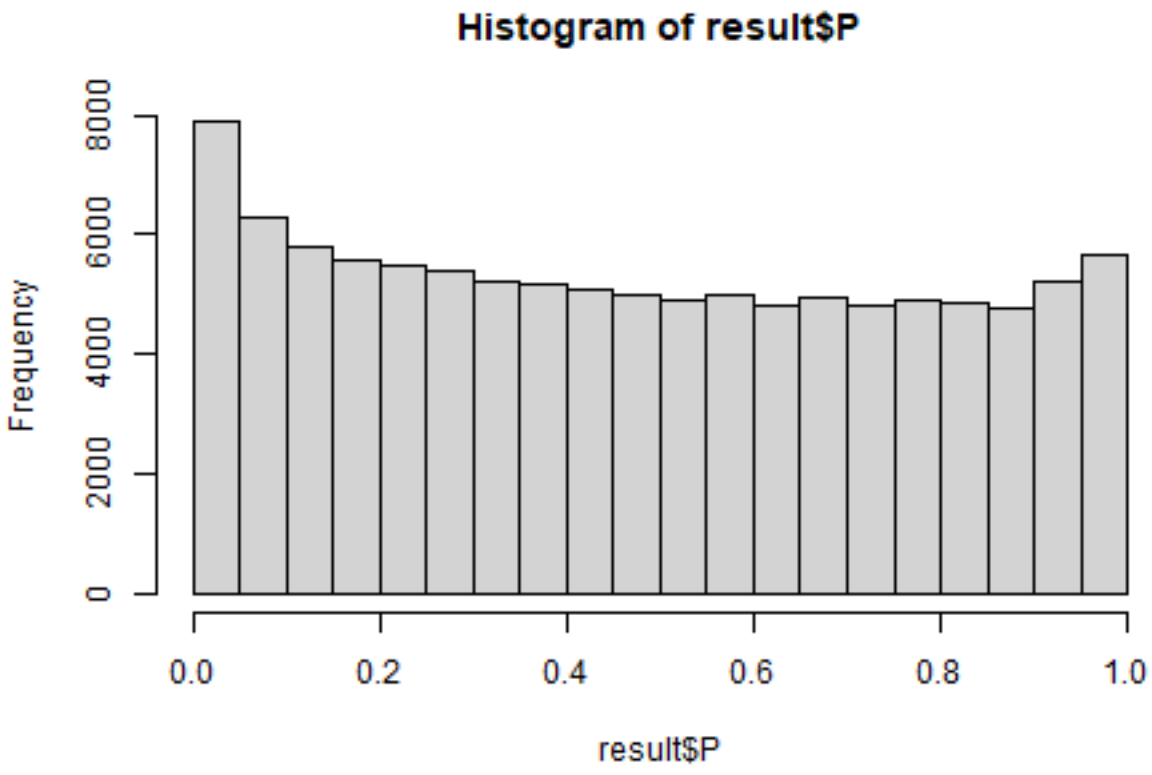
```
### First randomly draw a position:
### Again, first take account of the linkage disequilibrium problem:
set.seed(12345,sample.kind="Rounding")
p <- 1
freq_counts <- big_counts(G)
MAF <- snp_MAF(G)
Qualified <- freq_counts[3,] >= 200 & MAF>=0.25
POS_EFF <- sample(POS[Qualified], size = 1)
indx <- which(CHR == CHR[which(POS == POS_EFF)])
G_using <- G[,indx]
CHR_using <- CHR[indx]
POS_using <- POS[indx]

### Let's repeat for 30 times
result <- Proposed_Aggreg(k = 10, SNP_names = obj.bigSNP$map$marker.ID[indx], Gusing = G_using, POS_usi
result <- result %>% filter(P > 0)
manhattan(result, highlight = obj.bigSNP$map$marker.ID[indx][which(POS_using == POS_EFF)], suggestiveli
```




```
qq(na.omit(result)$P)
```




```
hist(result$P, breaks = 30)
```




```
### Check for power:
classifications <- result %>% filter(BP == POS_EFF) %>% mutate(classified = ifelse(P <= (5 * (10 ^ -8))
sum(classifications)/length(classifications)
```

```
## [1] 1
```

```
### Check for type I error rate:
error <- result %>% filter(BP != POS_EFF) %>% mutate(classified = ifelse(P <= (5 * (10 ^ -8)), 1, 0)) %
sum(error)/length(error)
```

```
## [1] 0
```

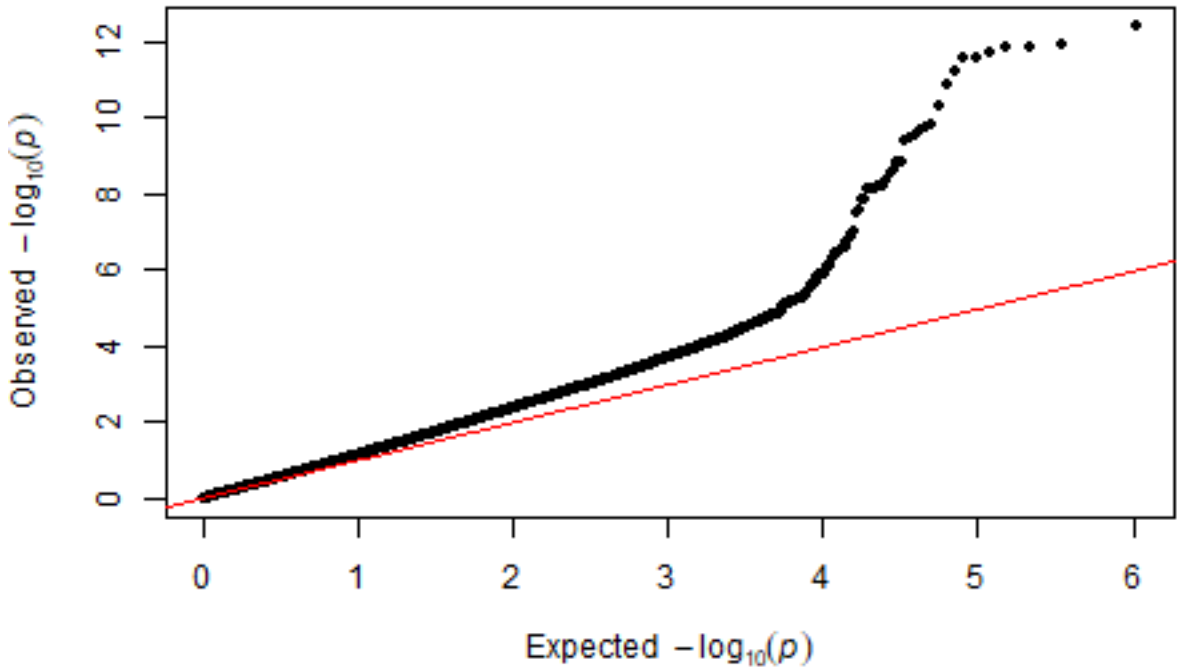
Based on the result above, with 50 times aggregations, our power is estimated to be around 10%, while the type I error rate is estimated to be zero in these simulations.

Then we changed the setting of our simulation. The values of our parameters now are:

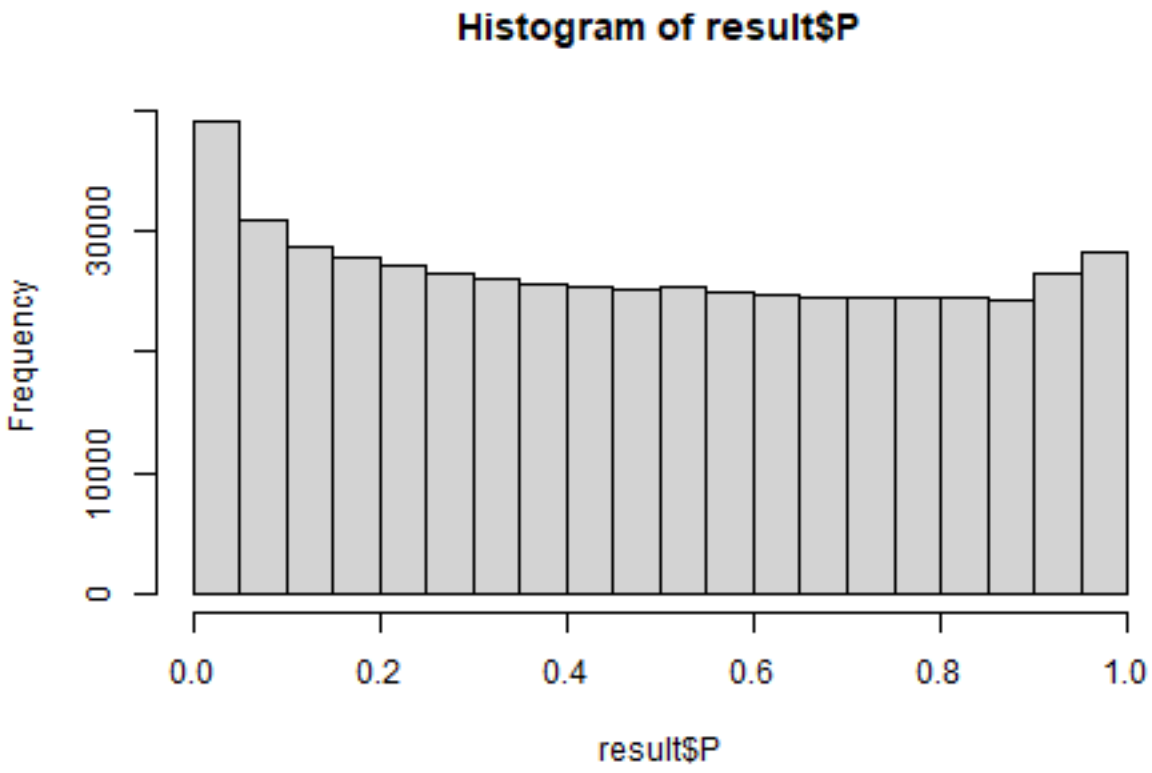
- $\beta_0 = -1, \beta_G = 0.8, \beta_E = 0.3, \gamma = 0.5$
- $E \sim N(0, 1)$
- $\epsilon \sim N(0, 0.5)$

```
### First randomly draw a position:
### Again, first take account of the linkage disequilibrium problem:
set.seed(123,sample.kind="Rounding")
```

```r
### Let's repeat for 30 times
result <- Proposed_Aggreg(k = 50, SNP_names = obj.bigSNP$map$marker.ID[indx], Gusing = G_using, POS_usi
result <- result %>% filter(P > 0)

qq(na.omit(result)$P)
```




```r
hist(result$P, breaks = 30)
```




```
### Check for power:
classifications <- result %>% filter(BP == POS_EFF) %>% mutate(classified = ifelse(P <= (5 * (10 ^ -8))
sum(classifications)/length(classifications)
```

```
## [1] 0.66
```

```
### Check for type I error rate:
error <- result %>% filter(BP != POS_EFF) %>% mutate(classified = ifelse(P <= (5 * (10 ^ -8)), 1, 0)) %
sum(error)/length(error)
```

```
## [1] 0
```

By increasing $\gamma$ from 0.3 to 0.5, we can see that the power of our proposed approach increases significantly.

**Summary:** Through simulation study above, we found that the performance of our proposed approach is highly dependent on the following:

- How large is $\gamma$ compared to $\beta_E$: Increasing $\beta_E$ will significantly decrease the power.
- How large is $Var(\epsilon) = \sigma^2_\epsilon$: Decreasing this will increase the power of our procedure, sometimes by a lot.
- How large is $|\beta_G|$: Higher $|\beta_G|$ is also associated with higher power.

## 5.3 Comparison between the Proposed method with oracle method:

Here we still consider the 1kGP dataset which we used in the last section. The true underlying model of our simulation will be:

$$Y^* = \beta_0 + \beta_G G_k + \beta_E E + \gamma G_k E + \epsilon$$

with the following model specifications:

- $\beta_0 = -1, \beta_G = 0.8, \beta_E = 0.3, \gamma = 0.5$
- $E \sim N(0, 1)$
- $\epsilon \sim N(0, 0.5)$ The $G_k$ will be sampled for six times, with different levels of MAF in each time.

We will compare our proposed method with the oracle method, which assumes that the information about $E$ can be observed (either with measurement error or not), so the direct test of $G \times E$ is possible. When there exists measurement error in $E$, we assume that the observed $\tilde{E} = E + W$ where $W \sim N(0, \sigma_w^2 = 0.25\sigma_E^2)$ is independent of $E$.

### 5.3.1 Power Comparison:

```
set.seed(123,sample.kind="Rounding")

### Sample three casual genes, corresponding to different MAF
freq_counts <- big_counts(G,ind.col = indx)
MAF <- snp_MAF(G,ind.col = indx)
Qualified <- freq_counts[3,] >= 30 & MAF>=0.05 & MAF<=0.1
POS_EFF1 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 30 & MAF>=0.1 & MAF<=0.15
POS_EFF2 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 80 & MAF>=0.15 & MAF<=0.2
POS_EFF3 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 30 & MAF>=0.2 & MAF<=0.3
POS_EFF4 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.3 & MAF<=0.4
POS_EFF5 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 30 & MAF>=0.4
POS_EFF6 <- sample(POS_using[Qualified], size = 1)


### show their corresponding MAF:
sort(MAF[POS_using %in% c(POS_EFF1,POS_EFF2,POS_EFF3, POS_EFF4, POS_EFF5, POS_EFF6)])
```

```
## [1] 0.0703259 0.1143511 0.1795312 0.2307033 0.3807890 0.4885649
```

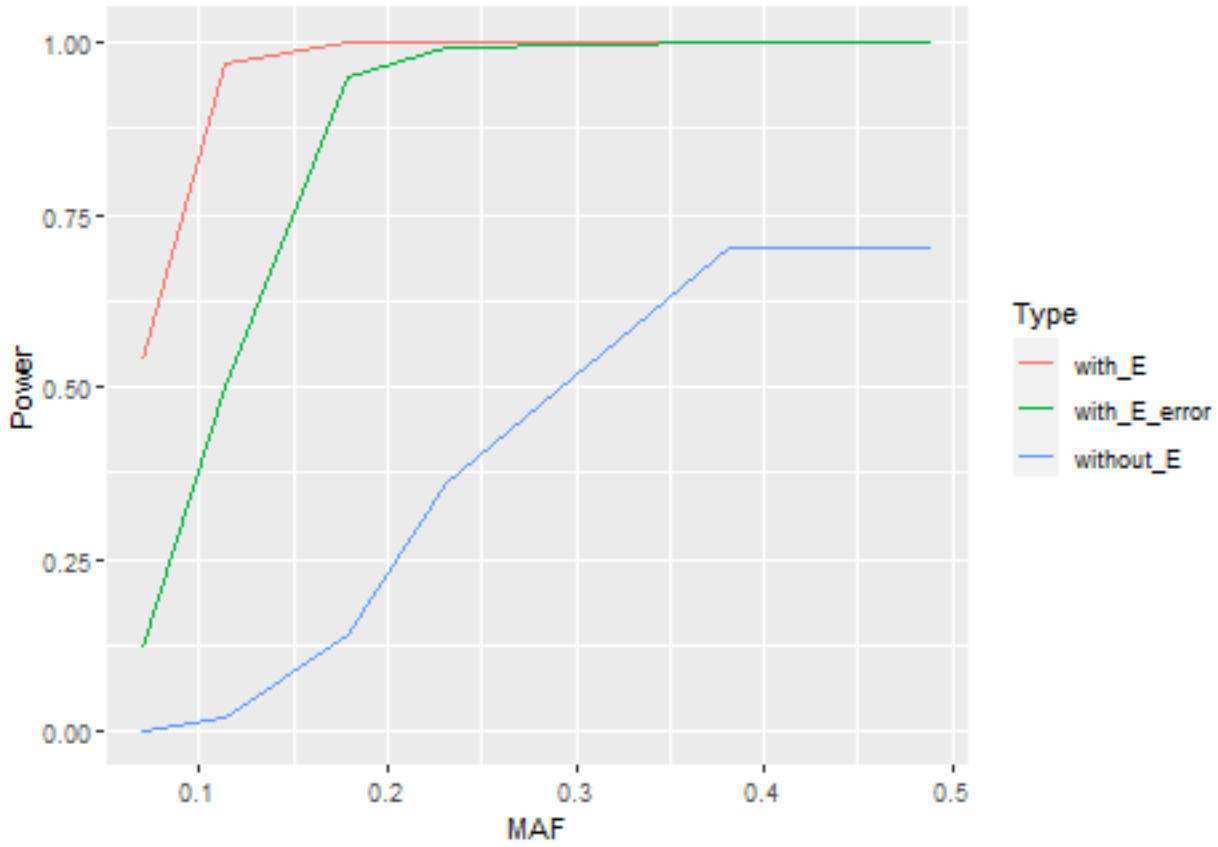
```
### Compute power of each case:
suppressWarnings(power1 <- Compare_Aggreg_power(k = 100, G_using = G_using, POS_using = POS_using, effe
power1 <- power1 <= 5 * (10 ^ -8)
power1 <- apply(power1, MARGIN = 2, FUN = mean)

suppressWarnings(power2 <- Compare_Aggreg_power(k = 100, G_using = G_using, POS_using = POS_using, effe
power2 <- power2 <= 5 * (10 ^ -8)
power2 <- apply(power2, MARGIN = 2, FUN = mean)
```

Table 1: Comparison of Power

| proposed | oracle | oracle_error | MAF |
|---|---|---|---|
| 0.00 | 0.54 | 0.12 | 0.0703259 |
| 0.02 | 0.97 | 0.50 | 0.1143511 |
| 0.14 | 1.00 | 0.95 | 0.1795312 |
| 0.36 | 1.00 | 0.99 | 0.2307033 |
| 0.70 | 1.00 | 1.00 | 0.3807890 |
| 0.70 | 1.00 | 1.00 | 0.4885649 |

```
suppressWarnings(power3 <- Compare_Aggreg_power(k = 100, G_using = G_using, POS_using = POS_using, effe
power3 <- power3 <= 5 * (10 ^ -8)
power3 <- apply(power3, MARGIN = 2, FUN = mean)

suppressWarnings(power4 <- Compare_Aggreg_power(k = 100, G_using = G_using, POS_using = POS_using, effe
power4 <- power4 <= 5 * (10 ^ -8)
power4 <- apply(power4, MARGIN = 2, FUN = mean)

suppressWarnings(power5 <- Compare_Aggreg_power(k = 100, G_using = G_using, POS_using = POS_using, effe
power5 <- power5 <= 5 * (10 ^ -8)
power5 <- apply(power5, MARGIN = 2, FUN = mean)

suppressWarnings(power6 <- Compare_Aggreg_power(k = 100, G_using = G_using, POS_using = POS_using, effe
power6 <- power6 <= 5 * (10 ^ -8)
power6 <- apply(power6, MARGIN = 2, FUN = mean)
## Table:
power <- as_tibble(as.matrix(rbind(power1,power2,power3, power4,power5,power6))) %>% mutate(MAF = sort(
kableExtra::kable(power,caption = "Comparison of Power")
```

```
## Figure:
power_plot <- power %>% pivot_longer(proposed:oracle_error, "Type", values_to = "Power") %>% mutate(Type
levels(power_plot$Type) <- c("with_E","with_E_error", "without_E")
power_plot %>% ggplot(aes(MAF,Power,color = Type)) + geom_line()
```




Seems like the oracle method with information of `E` will have better power than the proposed method, regardless whether there is measurement error. This difference gets smaller as `MAF` increases.

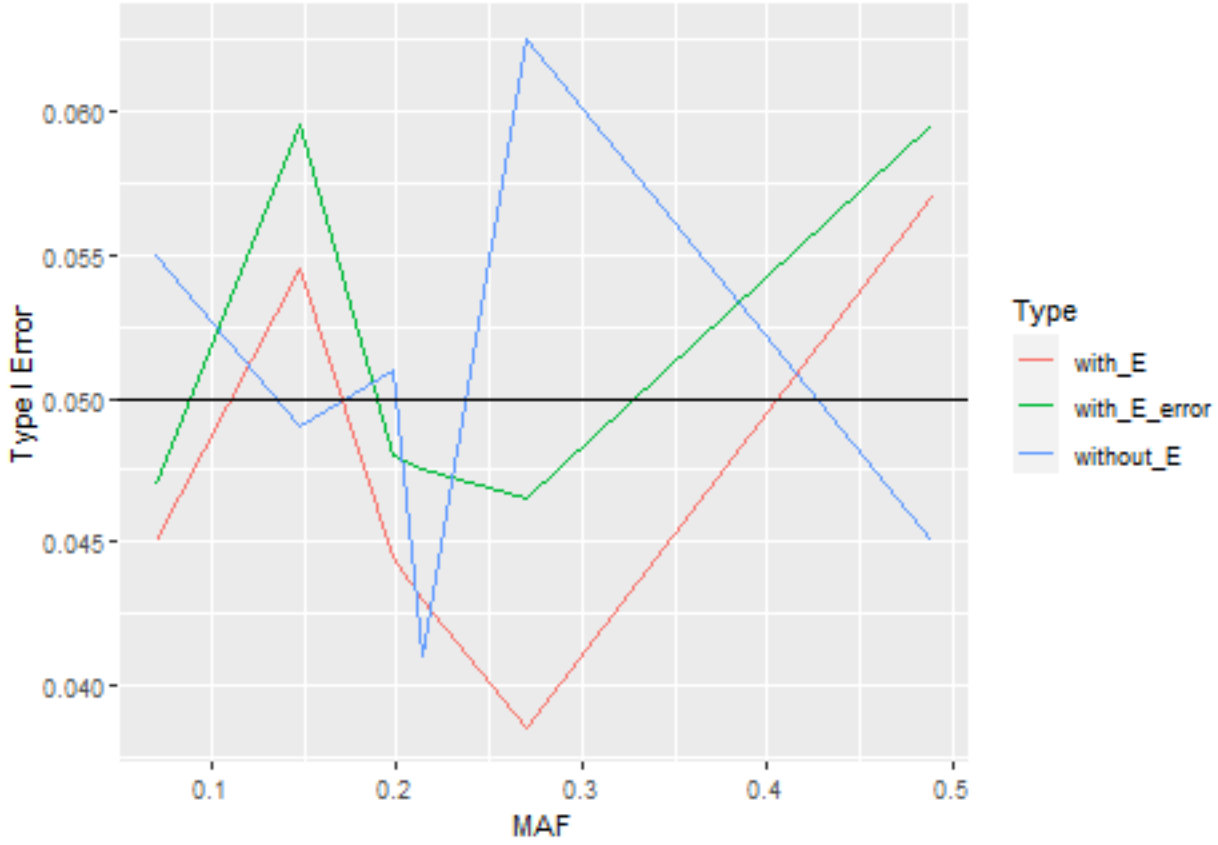
### 5.3.2 Type I error Comparison:

```
set.seed(123,sample.kind="Rounding")

### Sample three casual genes, corresponding to different MAF
freq_counts <- big_counts(G,ind.col = indx)
MAF <- snp_MAF(G,ind.col = indx)
Qualified <- freq_counts[3,] >= 30 & MAF>=0.05 & MAF<=0.1
POS_EFF1 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 80 & MAF>=0.1 & MAF<=0.15
POS_EFF2 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.15 & MAF<=0.2
POS_EFF3 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.2 & MAF<=0.25
POS_EFF4 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.25 & MAF<=0.3
POS_EFF5 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.3
POS_EFF6 <- sample(POS_using[Qualified], size = 1)
### show their corresponding MAF:
sort(MAF[POS_using %in% c(POS_EFF1,POS_EFF2,POS_EFF3,POS_EFF4,POS_EFF5,POS_EFF6)])
```

```
## [1] 0.0703259 0.1477987 0.1975415 0.2135506 0.2695826 0.4885649
```

```
### Compute type I error rate of each case:
#1:
error1 <- Compare_Aggreg_power(k = 2000, G_using = G_using, POS_using = POS_using, effective_gene = POS
r1 <- error1 %>% summarise(proposed = mean(proposed <=0.05), oracle = mean(oracle <= 0.05), oracle_erro



#2:
error2 <- Compare_Aggreg_power(k = 2000, G_using = G_using, POS_using = POS_using, effective_gene = POS
r2 <- error2 %>% summarise(proposed = mean(proposed <=0.05), oracle = mean(oracle <= 0.05), oracle_erro


#3:
error3 <- Compare_Aggreg_power(k = 2000, G_using = G_using, POS_using = POS_using, effective_gene = POS
r3 <- error3 %>% summarise(proposed = mean(proposed <=0.05), oracle = mean(oracle <= 0.05), oracle_erro

#4:
error4 <- Compare_Aggreg_power(k = 2000, G_using = G_using, POS_using = POS_using, effective_gene = POS
r4 <- error4 %>% summarise(proposed = mean(proposed <=0.05), oracle = mean(oracle <= 0.05), oracle_erro



#5:
error5 <- Compare_Aggreg_power(k = 2000, G_using = G_using, POS_using = POS_using, effective_gene = POS
r5 <- error5 %>% summarise(proposed = mean(proposed <=0.05), oracle = mean(oracle <= 0.05), oracle_erro


#6:
```

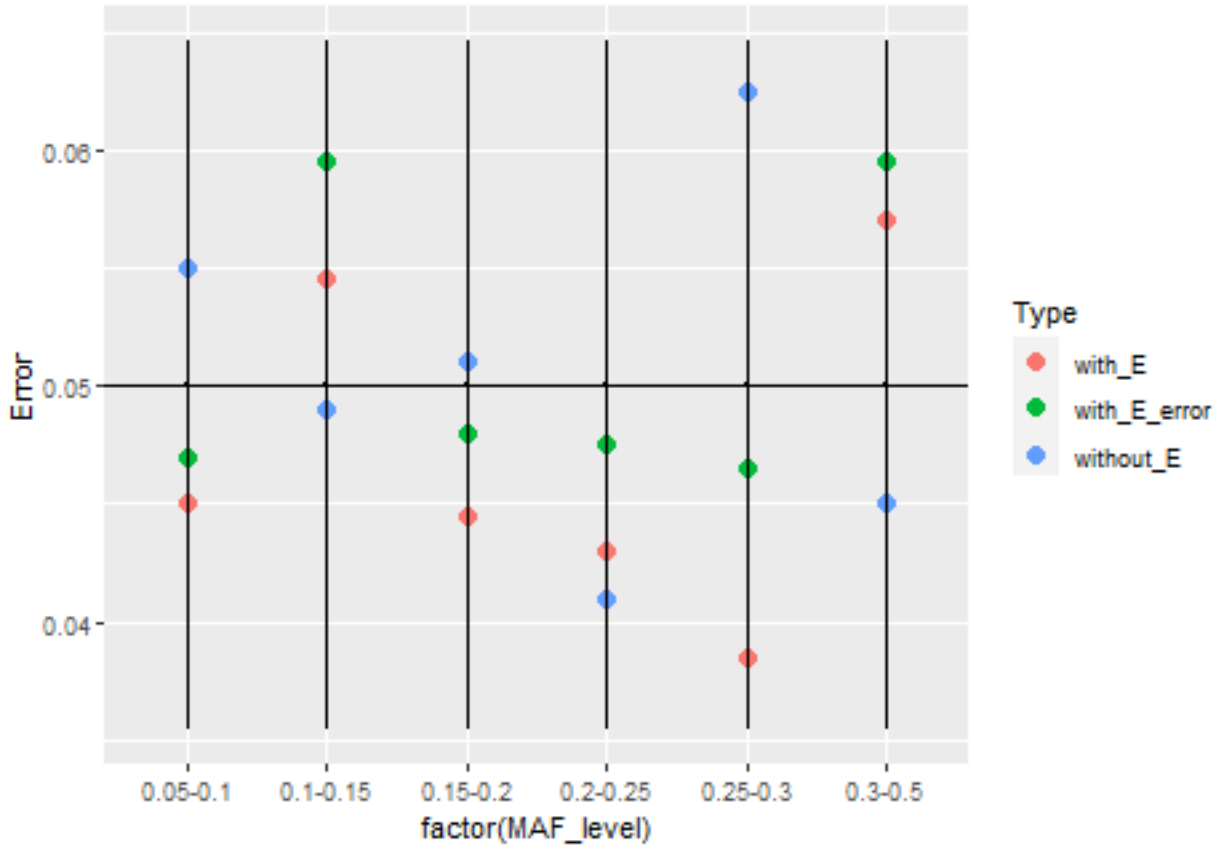
```
error6 <- Compare_Aggreg_power(k = 2000, G_using = G_using, POS_using = POS_using, effective_gene = POS
r6 <- error6 %>% summarise(proposed = mean(proposed <=0.05), oracle = mean(oracle <= 0.05), oracle_erro



r <- as_tibble(rbind(r1,r2,r3,r4,r5,r6))
r$MAF <- sort(MAF[POS_using %in% c(POS_EFF1,POS_EFF2,POS_EFF3,POS_EFF4,POS_EFF5,POS_EFF6)])
r_plot <- r %>% rename(c("proposed" = "proposed","oracle" = "oracle")) %>% pivot_longer(proposed:oracle
levels(r_plot$Type) <- c("with_E","with_E_error", "without_E")
r_plot %>% ggplot(aes(MAF,Error,color = Type)) + geom_line() + ylab("Type I Error") + geom_hline(aes(yi
```




The above plot shows the relationship between the observed type I error rate (assuming significance level of 5 percent) with the MAF of the `casual` gene in the underlying model. Next, we can take a look into the relationship between MAF (of non-casual genes) and the observed type I error rate:

```
all_error <- r
all_error$MAF_level <- ifelse(all_error$MAF >= 0.1, "0.1-0.15","0.05-0.1")
all_error$MAF_level <- ifelse(all_error$MAF >= 0.15, "0.15-0.2", all_error$MAF_level)
all_error$MAF_level <- ifelse(all_error$MAF >= 0.2, "0.2-0.25", all_error$MAF_level)
all_error$MAF_level <- ifelse(all_error$MAF >= 0.25, "0.25-0.3", all_error$MAF_level)
all_error$MAF_level <- ifelse(all_error$MAF >= 0.3, "0.3-0.5", all_error$MAF_level)



all_error <- all_error %>% group_by(MAF_level) %>% summarise(proposed = mean(proposed), oracle = mean(o

all_error <- all_error %>% mutate(upper = 0.05 + 3*sqrt(0.05*0.95/number), lower = 0.05 - 3*sqrt(0.05*0
```

```
all_error_plot <- all_error %>% pivot_longer(proposed:oracle_error, "Type", values_to = "Error") %>% mu
levels(all_error_plot$Type) <- c("with_E","with_E_error", "without_E")

all_error_plot %>% ggplot(aes(factor(MAF_level))) + geom_point(aes(y = Error, color = Type), size = 3) +
```




The range in each category is computed using $\alpha \pm 3\sqrt{\alpha \times (1-\alpha)/\text{num.rep}}$. Note that for very low MAF level (0.05-0.1), the type I error rate of our proposed method seems invalid. However, as MAF level increases, all the methods seem to have correct type I error behaviors.

## 5.4 Simulation with Auxiliary Variable:

Now, assumes that the true underlying model has the form:

$$Y^* = \beta_0 + \beta_G G_k + \beta_E E + \gamma G_k E + \beta_Z Z + \epsilon$$

where $Z$ is an auxiliary variable which is known to have no interaction with environmental variable $E$.

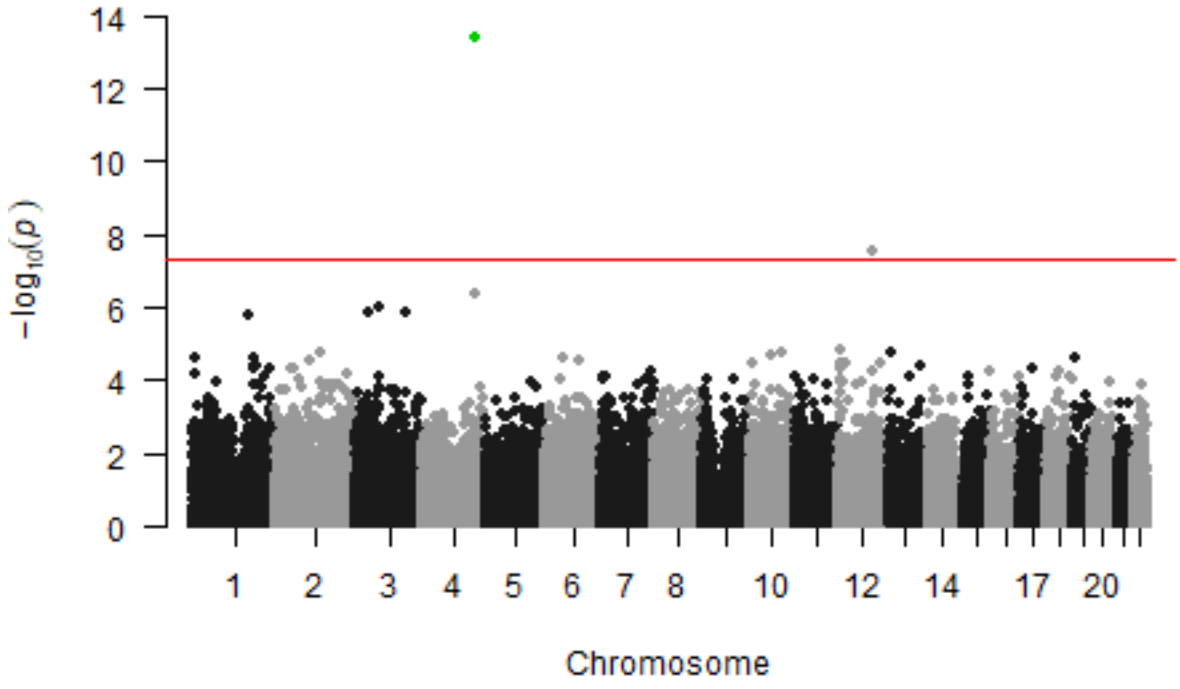
### 5.4.1 Analysis of 1KGP data with one casual gene:

Here we repeat the same procedure on the 1kGP data set, assuming that there is one casual gene in the model

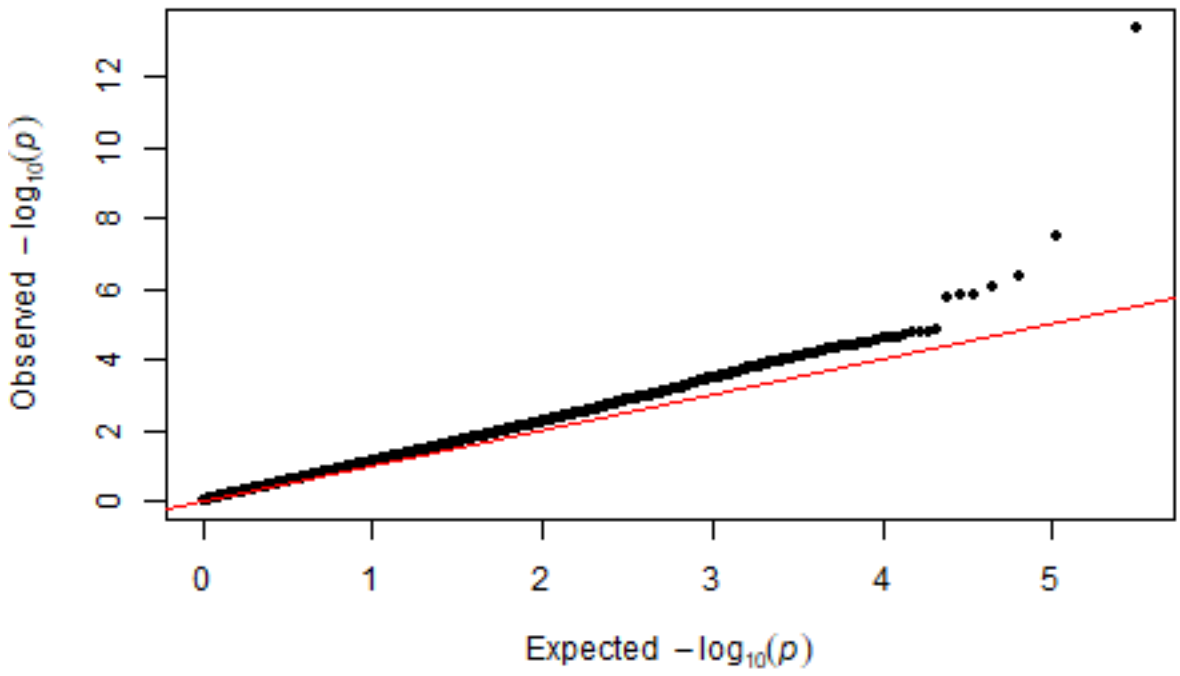
```
set.seed(123,sample.kind="Rounding")
indx <- 1:ncol(G)

G_using <- G[,indx]
CHR_using <- CHR[indx]
POS_using <- POS[indx]


### Randomly sample 1 genes:
p <- 1
### Need to make sure that all genotypes have enough frequencies in the selected genes:
freq_counts <- big_counts(G,ind.col = indx)
MAF <- snp_MAF(G,ind.col = indx)
Qualified <- freq_counts[3,] >= 50 & MAF>=0.3
POS_EFF <- sample(POS_using[Qualified], size = 1)
Qualified_2 <- freq_counts[3,] >= 50
G_using <- G[,Qualified_2]
CHR_using <- CHR[Qualified_2]
POS_using <- POS[Qualified_2]


### Randomly sample 1/2 genes with strong effects and 1/2 genes with weak effects
b0 <- -1
bG <- 0.8
bint <- 0.8
bE <- 0.1
bZ <- 0.3

### Generate the underlying environment variable:
E <- rnorm(n = length(G_using[,1]), sd = 1)

### Generate the auxiliary variable:
Z <- rnorm(n = length(G_using[,1]), sd = 1)

### Generate the latent variable:
y_lat <- b0 + bG*G_using[,POS_using == POS_EFF] + bE*E + bZ*Z + bint*E*G_using[,POS_using == POS_EFF] +
case_new <- ifelse(y_lat > 0, 1, 0)

### case_control ratio:
table(case_new)
```

```
## case_new
##    0    1
## 1175  574
```

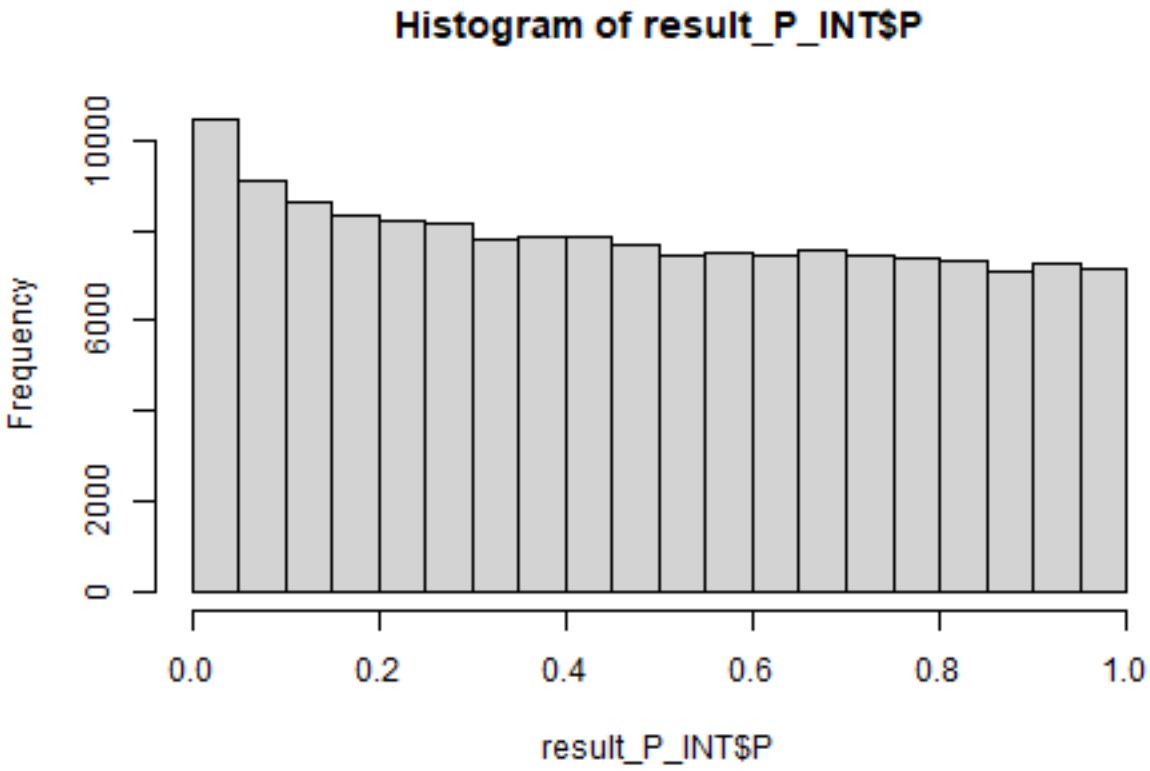
```
#### Testing for interaction effect
### Assuming additive model: Testing for non-linearity
waldstats_P1 <- c()
p_vals_P1 <- c()

for (i in 1:ncol(G_using)) {
  Gi <- G_using[,i]

  if(length(unique(Gi)) != 3){
    waldstats_P1[i] <- 0
    p_vals_P1[i] <- -1
  }
  else{
    modi <- glm(case_new ~ factor(Gi)*Z, family = binomial(link = "probit"))
    if(any(is.na(modi$coefficients))){
      waldstats_P1[i] <- 0
      p_vals_P1[i] <- -1

    }
    else{
      waldstats_P1[i] <- as.numeric(aod::wald.test(vcov(modi)[-c(1,4),-c(1,4)], b = modi$coefficients[-
      p_vals_P1[i] <- as.numeric(aod::wald.test(vcov(modi)[-c(1,4),-c(1,4)], b = modi$coefficients[-c(1
    }
  }
}

result_P_INT <- tibble(SNP = obj.bigSNP$map$marker.ID[Qualified_2], CHR = CHR_using, BP = POS_using , s
manhattan(result_P_INT, highlight = result_P_INT$SNP[which(POS_using %in% POS_EFF)], suggestiveline = F
```




```
qq(na.omit(result_P_INT)$P)
```




```
hist(result_P_INT$P, breaks = 30)
```


Histogram of result_P_INT$P
Frequency
0
2000
6000
10000
0.0
0.2
0.4
0.6
0.8
1.0
result_P_INT$P

### 5.4.2 Power Comparison:

Here we consider the following parameter specification:

- $\beta_0 = -1, \beta_G = 0.8, \beta_E = 0.2, \gamma = 0.4, \beta_Z = 0.8$
- $E \sim N(0,1)$
- $\epsilon \sim N(0, 0.5)$ The $G_k$ will be sampled for six times, with different levels of MAF in each time.

```
set.seed(123,sample.kind="Rounding")
indx <- 1:ncol(G)

G_using <- G[,indx]
CHR_using <- CHR[indx]
POS_using <- POS[indx]

### Sample three casual genes, corresponding to different MAF
freq_counts <- big_counts(G,ind.col = indx)
MAF <- snp_MAF(G,ind.col = indx)
Qualified <- freq_counts[3,] >= 30 & MAF>=0.05 & MAF<=0.1
POS_EFF1 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 50 & MAF>=0.1 & MAF<=0.15
POS_EFF2 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 80 & MAF>=0.15 & MAF<=0.2
POS_EFF3 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 80 & MAF>=0.2 & MAF<=0.3
POS_EFF4 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.3 & MAF<=0.4
POS_EFF5 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.4
POS_EFF6 <- sample(POS_using[Qualified], size = 1)


### Computation:
### Compute power of each case:
suppressWarnings(power1 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power1 <- power1 <= 5 * (10 ^ -8)
power1 <- apply(power1, MARGIN = 2, FUN = mean)

suppressWarnings(power2 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power2 <- power2 <= 5 * (10 ^ -8)
power2 <- apply(power2, MARGIN = 2, FUN = mean)

suppressWarnings(power3 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power3 <- power3 <= 5 * (10 ^ -8)
power3 <- apply(power3, MARGIN = 2, FUN = mean)

suppressWarnings(power4 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power4 <- power4 <= 5 * (10 ^ -8)
power4 <- apply(power4, MARGIN = 2, FUN = mean)

suppressWarnings(power5 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power5 <- power5 <= 5 * (10 ^ -8)
power5 <- apply(power5, MARGIN = 2, FUN = mean)
```

Table 2: Comparison of Power

| proposed_1 | proposed_2 | oracle | oracle_error | MAF |
|---|---|---|---|---|
| 0.005 | 0.070 | 0.680 | 0.310 | 0.0737564 |
| 0.020 | 0.110 | 0.980 | 0.855 | 0.1463694 |
| 0.055 | 0.205 | 0.995 | 0.960 | 0.1803888 |
| 0.050 | 0.170 | 1.000 | 1.000 | 0.2172670 |
| 0.070 | 0.250 | 1.000 | 1.000 | 0.3070326 |
| 0.040 | 0.175 | 1.000 | 1.000 | 0.4568325 |

```
suppressWarnings(power6 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power6 <- power6 <= 5 * (10 ^ -8)
power6 <- apply(power6, MARGIN = 2, FUN = mean)

## Table:
power <- as_tibble(as.matrix(rbind(power1,power2,power3, power4,power5,power6))) %>% mutate(MAF = sort(
kableExtra::kable(power,caption = "Comparison of Power")
```

```
## Figure:

power_plot <- power %>% pivot_longer(proposed_1:oracle_error, "Type", values_to = "Power") %>% mutate(Ty
levels(power_plot$Type) <- c("with_E","with_E_error", "without_E_1df", "without_E_3df")
power_plot %>% ggplot(aes(MAF,Power,color = Type)) + geom_line()
```

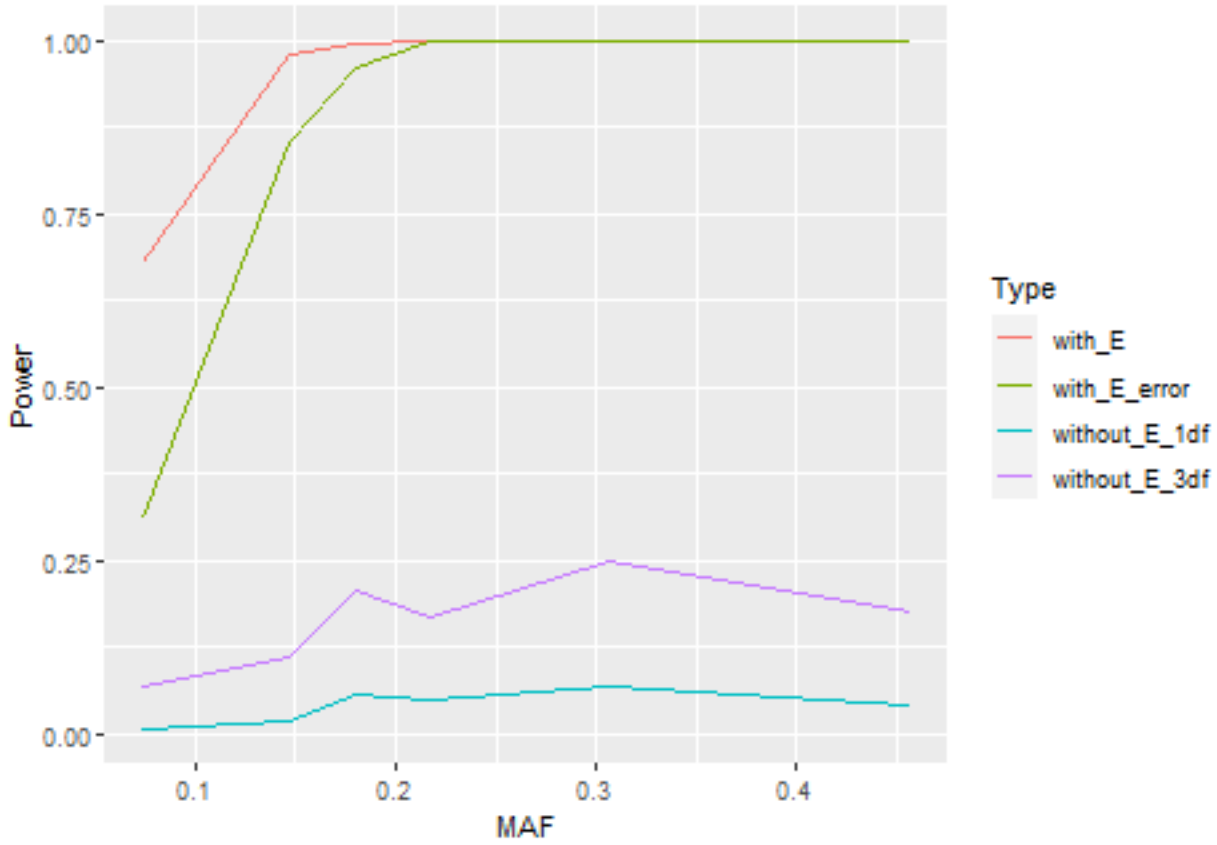


Table 3: Comparison of Power

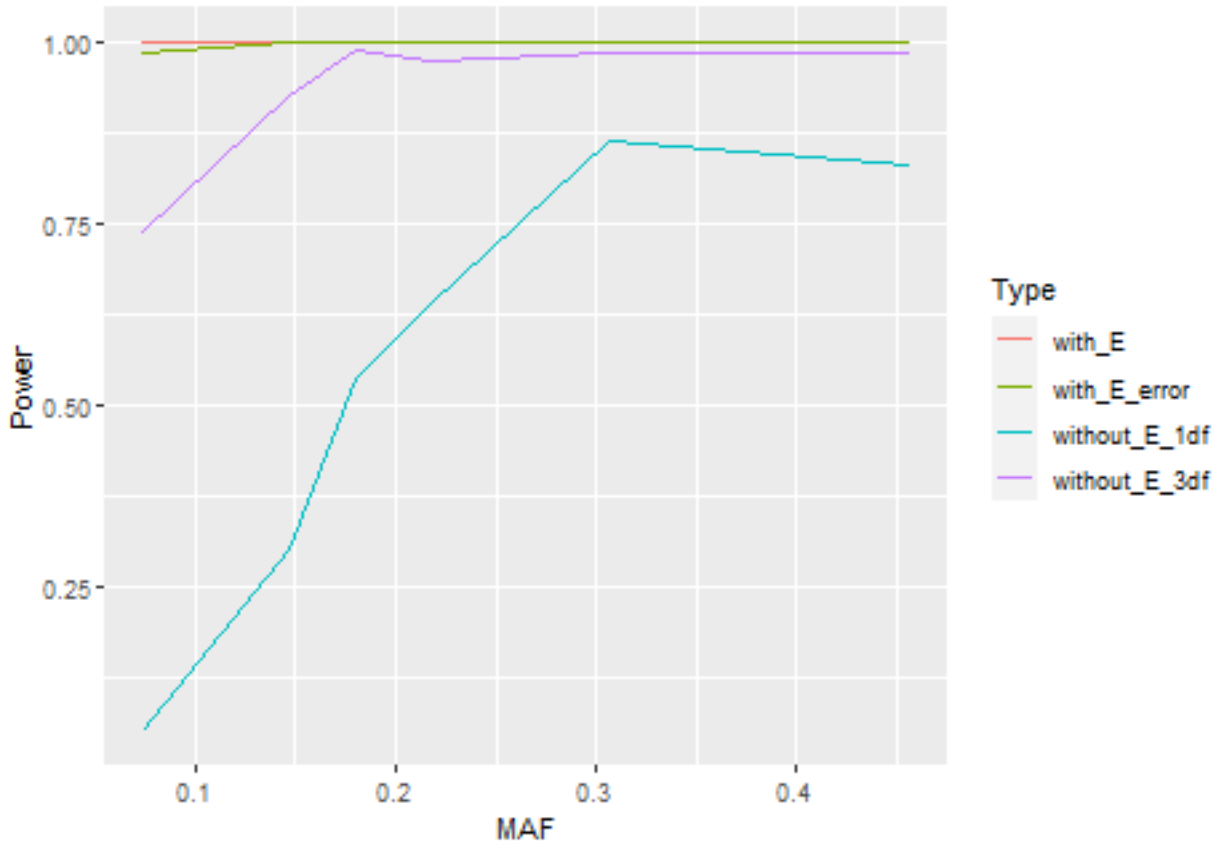
| proposed_1 | proposed_2 | oracle | oracle_error | MAF |
|---|---|---|---|---|
| 0.050 | 0.740 | 1 | 0.985 | 0.0737564 |
| 0.300 | 0.925 | 1 | 1.000 | 0.1463694 |
| 0.535 | 0.990 | 1 | 1.000 | 0.1803888 |
| 0.640 | 0.975 | 1 | 1.000 | 0.2172670 |
| 0.865 | 0.985 | 1 | 1.000 | 0.3070326 |
| 0.830 | 0.985 | 1 | 1.000 | 0.4568325 |

In this parameter setting, we can see that even though with the help from the auxiliary variable `z`, the power is increased significantly. The proposed methods still in generally have much lower power than the oracle method using information of `E`, regardless whether there is measurement error in `E`.

Next, we will make a more optimistic parameter setting, by increasing $\gamma$ from 0.4 to 0.8, and do the power comparison again:

```
set.seed(123,sample.kind="Rounding")

### Computation:
### Compute power of each case:
suppressWarnings(power1 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power1 <- power1 <= 5 * (10 ^ -8)
power1 <- apply(power1, MARGIN = 2, FUN = mean)

suppressWarnings(power2 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power2 <- power2 <= 5 * (10 ^ -8)
power2 <- apply(power2, MARGIN = 2, FUN = mean)

suppressWarnings(power3 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power3 <- power3 <= 5 * (10 ^ -8)
power3 <- apply(power3, MARGIN = 2, FUN = mean)

suppressWarnings(power4 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power4 <- power4 <= 5 * (10 ^ -8)
power4 <- apply(power4, MARGIN = 2, FUN = mean)

suppressWarnings(power5 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power5 <- power5 <= 5 * (10 ^ -8)
power5 <- apply(power5, MARGIN = 2, FUN = mean)

suppressWarnings(power6 <- Compare_Aggreg_power_auxiliary(k = 200, G_using = G_using, POS_using = POS_us
power6 <- power6 <= 5 * (10 ^ -8)
power6 <- apply(power6, MARGIN = 2, FUN = mean)
## Table:
power <- as_tibble(as.matrix(rbind(power1,power2,power3, power4,power5,power6))) %>% mutate(MAF = sort(
kableExtra::kable(power,caption = "Comparison of Power")
```

```
## Figure:
power_plot <- power %>% pivot_longer(proposed_1:oracle_error, "Type", values_to = "Power") %>% mutate(Ty
levels(power_plot$Type) <- c("with_E","with_E_error", "without_E_1df", "without_E_3df")
power_plot %>% ggplot(aes(MAF,Power,color = Type)) + geom_line()
```




From the plot above, we can notice that the difference becomes smaller as $\gamma$ gets larger.

### 5.4.3 Type I error Comparison:

Still consider the setting:

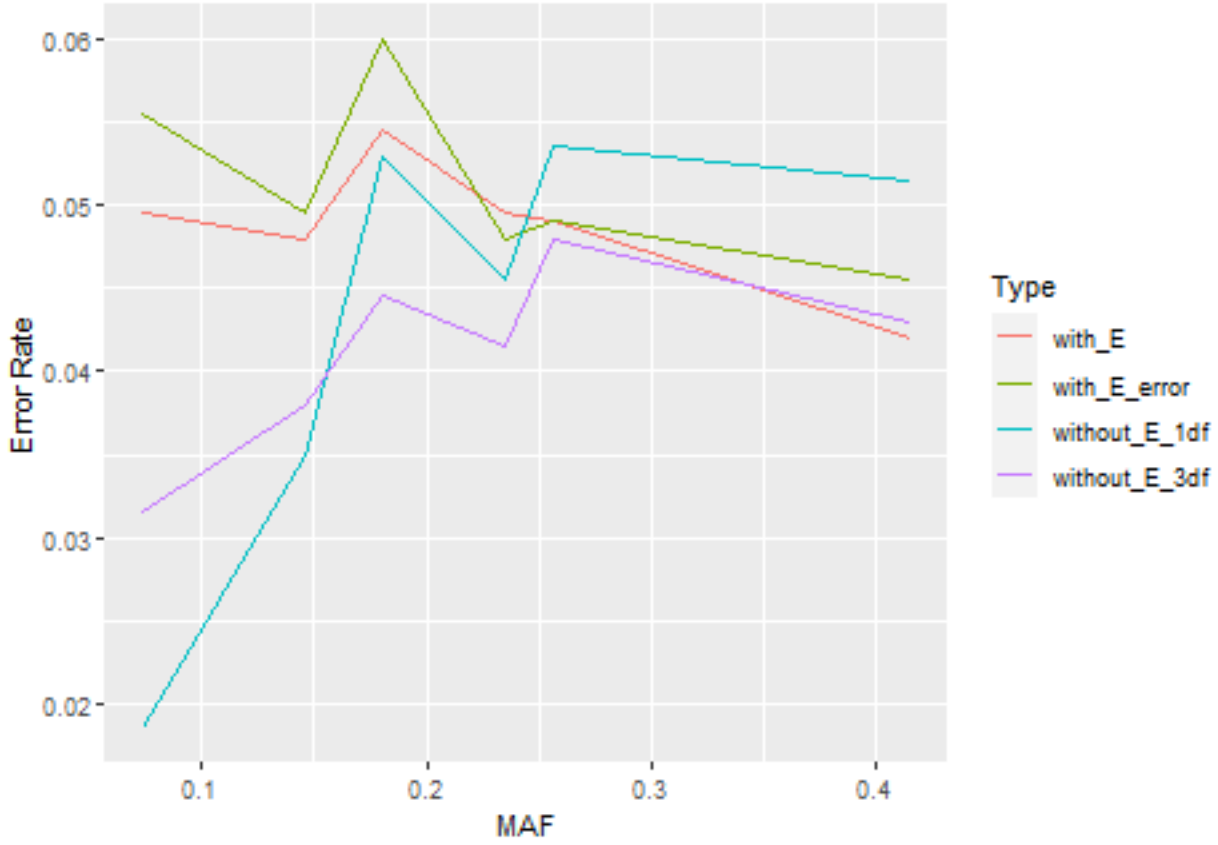
- $\beta_0 = -1, \beta_G = 0.8, \beta_E = 0.2, \gamma = 0.8, \beta_Z = 0.8$
- $E \sim N(0,1)$
- $\epsilon \sim N(0, 0.5)$ Now, we aim to compare the type I error rate of these procedures at different MAF levels:

```
set.seed(123,sample.kind="Rounding")

### Sample three casual genes, corresponding to different MAF
### Sample three casual genes, corresponding to different MAF
freq_counts <- big_counts(G,ind.col = indx)
MAF <- snp_MAF(G,ind.col = indx)
Qualified <- freq_counts[3,] >= 30 & MAF>=0.05 & MAF<=0.1
POS_EFF1 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 50 & MAF>=0.1 & MAF<=0.15
POS_EFF2 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 80 & MAF>=0.15 & MAF<=0.2
POS_EFF3 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 80 & MAF>=0.2 & MAF<=0.25
POS_EFF4 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.25 & MAF<=0.3
POS_EFF5 <- sample(POS_using[Qualified], size = 1)
Qualified <- freq_counts[3,] >= 100 & MAF>=0.3
POS_EFF6 <- sample(POS_using[Qualified], size = 1)

### show their corresponding MAF:
sort(MAF[POS_using %in% c(POS_EFF1,POS_EFF2,POS_EFF3)])
```

```
## [1] 0.07375643 0.14636935 0.18038879
```

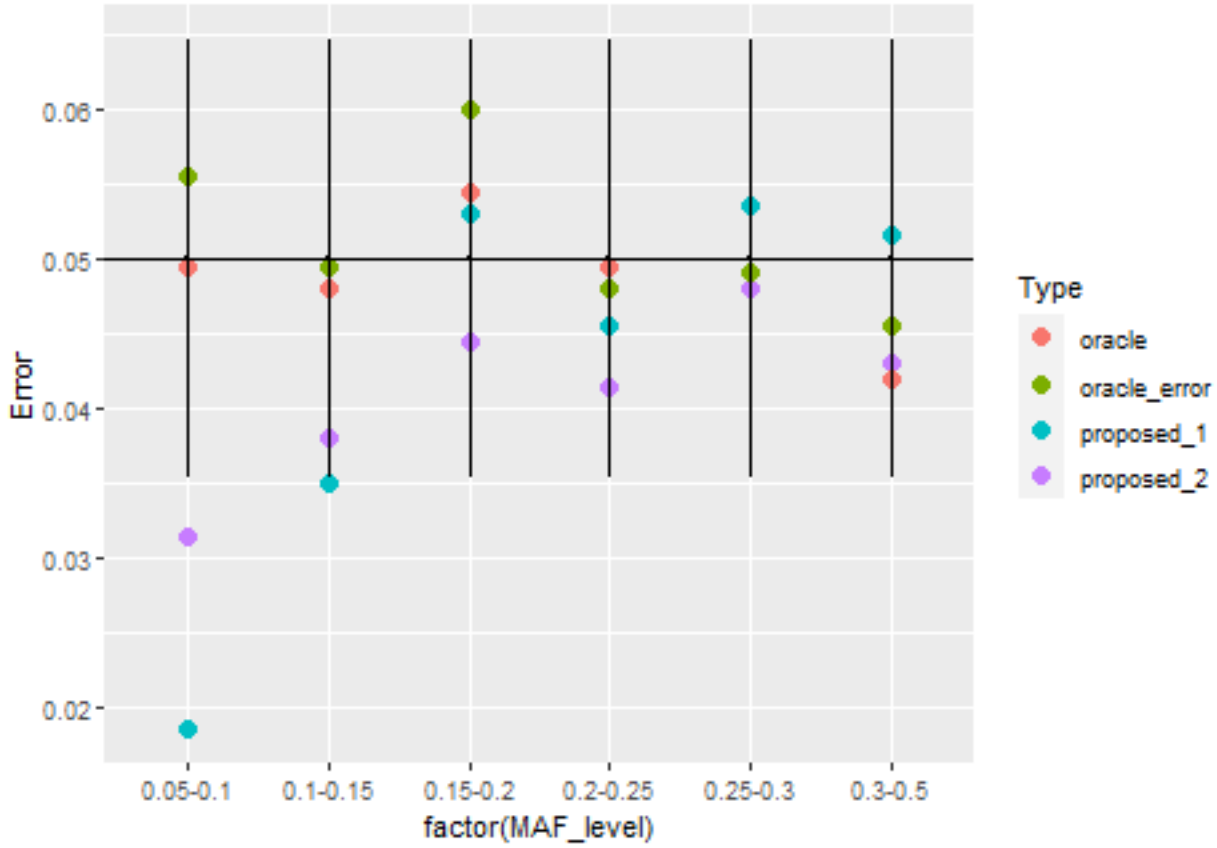
```
suppressWarnings(power1 <- Compare_Aggreg_power_auxiliary(k = 2000, G_using = G_using, POS_using = POS_u
power1 <- power1 <= 5 * (10 ^ -2)
power1 <- apply(power1, MARGIN = 2, FUN = mean)

suppressWarnings(power2 <- Compare_Aggreg_power_auxiliary(k = 2000, G_using = G_using, POS_using = POS_u
power2 <- power2 <= 5 * (10 ^ -2)
power2 <- apply(power2, MARGIN = 2, FUN = mean)

suppressWarnings(power3 <- Compare_Aggreg_power_auxiliary(k = 2000, G_using = G_using, POS_using = POS_u
power3 <- power3 <= 5 * (10 ^ -2)
power3 <- apply(power3, MARGIN = 2, FUN = mean)

suppressWarnings(power4 <- Compare_Aggreg_power_auxiliary(k = 2000, G_using = G_using, POS_using = POS_u
power4 <- power4 <= 5 * (10 ^ -2)
power4 <- apply(power4, MARGIN = 2, FUN = mean)

suppressWarnings(power5 <- Compare_Aggreg_power_auxiliary(k = 2000, G_using = G_using, POS_using = POS_u
power5 <- power5 <= 5 * (10 ^ -2)
power5 <- apply(power5, MARGIN = 2, FUN = mean)

suppressWarnings(power6 <- Compare_Aggreg_power_auxiliary(k = 2000, G_using = G_using, POS_using = POS_u
```

```r
power6 <- power6 <= 5 * (10 ^ -2)
power6 <- apply(power6, MARGIN = 2, FUN = mean)
## Table:
power <- as_tibble(as.matrix(rbind(power1,power2,power3, power4,power5,power6))) %>% mutate(MAF = sort(


power_plot <- power %>% pivot_longer(proposed_1:oracle_error, "Type", values_to = "Power") %>% mutate(Ty
levels(power_plot$Type) <- c("with_E","with_E_error", "without_E_1df", "without_E_3df")
power_plot %>% ggplot(aes(MAF,Power,color = Type)) + geom_line() + ylab("Error Rate")
```

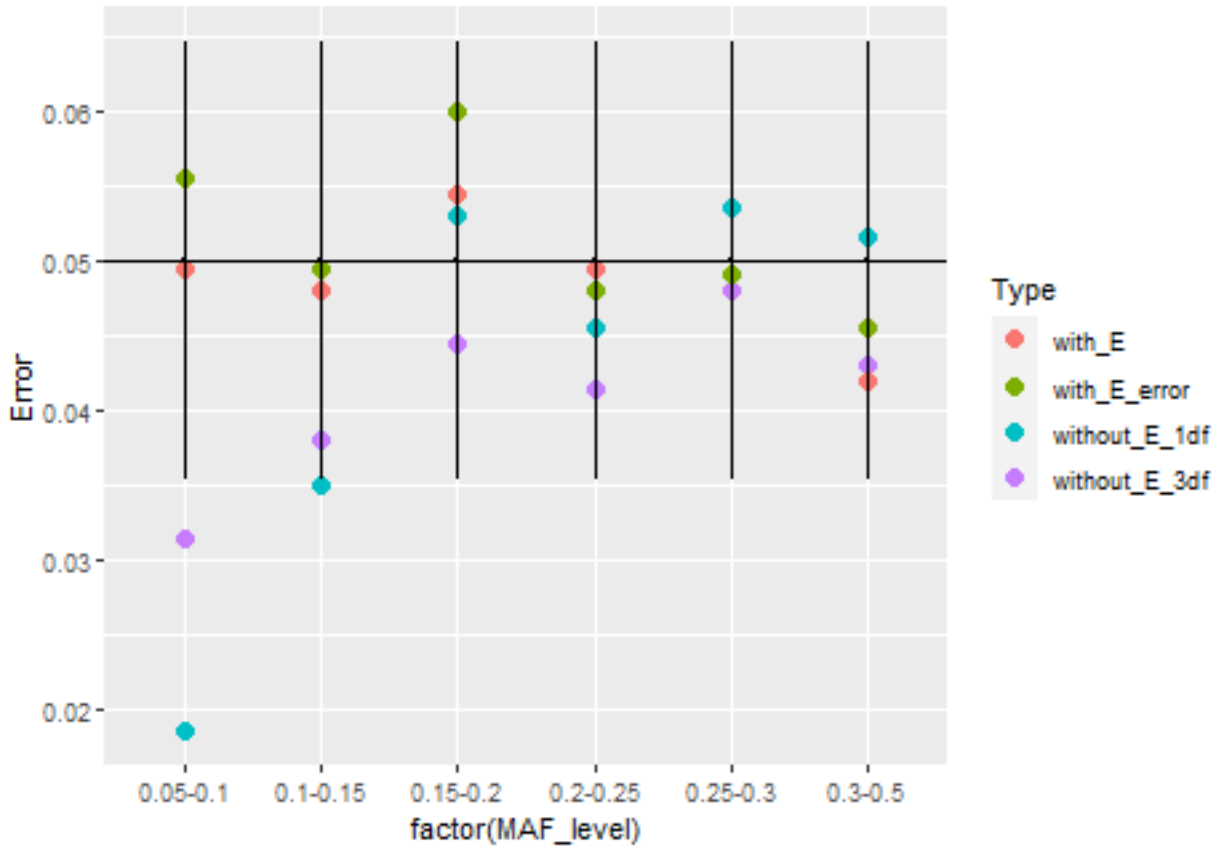



Here we pick the last set of simulation (with the MAF of casual gene being 0.42), and study the relationship between MAF and type I error more closely:

```r
all_error <- power
all_error$MAF_level <- ifelse(all_error$MAF >= 0.1, "0.1-0.15","0.05-0.1")
all_error$MAF_level <- ifelse(all_error$MAF >= 0.15, "0.15-0.2", all_error$MAF_level)
all_error$MAF_level <- ifelse(all_error$MAF >= 0.2, "0.2-0.25", all_error$MAF_level)
all_error$MAF_level <- ifelse(all_error$MAF >= 0.25, "0.25-0.3", all_error$MAF_level)
all_error$MAF_level <- ifelse(all_error$MAF >= 0.3, "0.3-0.5", all_error$MAF_level)


all_error <- all_error %>% group_by(MAF_level) %>% summarise(proposed_1 = mean(proposed_1),proposed_2 =

all_error <- all_error %>% mutate(upper = 0.05 + 3*sqrt(0.05*0.95/number), lower = 0.05 - 3*sqrt(0.05*0

all_error %>% pivot_longer(proposed_1:oracle_error, "Type", values_to = "Error") %>% ggplot(aes(factor(
```




```
all_error_plot <- all_error %>% pivot_longer(proposed_1:oracle_error, "Type", values_to = "Error") %>% r
levels(all_error_plot$Type) <- c("with_E","with_E_error", "without_E_1df", "without_E_3df")

all_error_plot %>% ggplot(aes(factor(MAF_level))) + geom_point(aes(y = Error, color = Type), size = 3) +
```




Based on the above plot, all the procedures have type I error rate in the expected range, when the MAF level is larger than 0.1.