STAT5900F

Assigment 2

Eralda Gjika Student ID:101248793

Due 17 November 2022

Introduction

We will be using the dataset from the publication below: Zhao, K., Tung, C. W., Eizenga, G. C., Wright, M. H., Ali, M. L., Price, A. H., ... & McCouch, S. R. (2011). Genome-wide association mapping reveals a rich genetic architecture of complex traits in Oryza sativa. Nature communications, 2(1), 1-10.

Question 1

Read the genotype data contained in the file "Genotype.csv". The data is coded using an additive model (i.e., Xi = 0, 1, 2). Remove any SNP that has more than 20% missing values. For the SNPs with less than 20% missing values, replace the missing values with the genotype of the heterozygous individual (i.e., replace missing data with 1).

Solution 1

At these part we are importing and observing the dimensions of our three datasets.

Importing the Phenodata

```
library(readr) # read csv and text file
# Import Pheno_data
Pheno_data <- read.delim("D:/ALDA 2021/CARLETON 2022/STAT5900-Genomic/Assignments STAT5900/Assignment 2
# [1] 413  38
```

Import Chromosome data

```
chromosome <- read_csv("D:/ALDA 2021/CARLETON 2022/STAT5900-Genomic/Assignments STAT5900/Assignment 2 S
```

```
## Rows: 36901 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): Chromosome, SNP_id, Genetic_distance, Position
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dim(chromosome)
```

```
## [1] 36901     4
```

```
# [1] 36901     4
```

Import Genotype data

```
Genotype <- read_csv("D:/ALDA 2021/CARLETON 2022/STAT5900-Genomic/Assignments STAT5900/Assignment 2 STA
```

```
## New names:
## Rows: 413 Columns: 36902
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): ...1 dbl (36901): SNP1, SNP2, SNP3, SNP4, SNP5, SNP6, SNP7, SNP8, SNP9,
## SNP10, SN...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
dim(Genotype)
```

```
## [1]   413 36902
```

```
# [1] 413 36902
```

Remove any SNP that has more than 20% missing values.

The function below will delete columns with missing values greater or equal than n% (n should take values like: 0.2 which corresponds to 20%)

```
delete.col_na <- function(D, n) {
  D[-which(sapply(D, function(x) sum(is.na(x)))>=n*nrow(D))]
}

Genotype_20<-delete.col_na(Genotype,0.2)# after removing columns with 20% missing values or higher
dim(Genotype_20)
```

```
## [1]   413 34923
```

```
# [1] 413 34923

Removed=ncol(Genotype)-ncol(Genotype_20)
Removed # number of SNPs removed
```

```
## [1] 1979
```

Replace the missing values

Replace NAs with the genotype of the heterozygous individual (i.e., replace missing data with 1).

```
Genotype_20[is.na(Genotype_20)]=1
dim(Genotype_20)
```

```
## [1]   413 34923
```

```
# [1]   413 34923
```

Question 2

The three phenotype traits "Seed.length", "Seed.width", and "Seed.volume" also consists of missing data. You may remove the observations with missing data or use any approach to estimate these missing data.

Solution 2

First we read the data and select from the dataset our interested variables (Seed.length, Seed.width and Seed.volume).

```
# The dataset with the phenotype:
Pheno_data <- read.delim("D:/ALDA 2021/CARLETON 2022/STAT5900-Genomic/Assignments STAT5900/Assignment 2
# Select the three variables from this dataset
Pheno_data3<-Pheno_data %>% select(Seed.length,Seed.width, Seed.volume)

dim(Pheno_data3)
```

```
## [1] 413   3
```

```
# [1] 413   3
head(Pheno_data3,4)
```

```
##   Seed.length Seed.width Seed.volume
## 1    8.064117   3.685183    2.587448
## 2    7.705383   2.951458    2.111145
## 3    8.237575   2.926367    2.154759
## 4    9.709375   2.382100    1.940221
```

Remove the observations with missing data from this dataset

Below is created a function which **delete NAs based on a threshold** which is by default zero (it means all rows which have NA's).

```
delete.row_na <- function(D, n=0) {
  D[rowSums(is.na(D)) <= n,]
}

Pheno_data4<-delete.row_na(Pheno_data3)
dim(Pheno_data4)
```

```
## [1] 377   3
```

```
# [1] 377   3
```

From 413 only 36 observations were removed which are approximately 8.7% of total observations. We will continue with this number 377 of observations for the steps below.

```
id=complete.cases(Pheno_data3) # returns a vector of row index which have NA's values
index<-which(id==FALSE) # save the index for those rows with NA's values
# index
# [1]  10  13  29  32  48  50  69  88  90 102 111 121 123
# [14] 145 164 166 307 332 333 334 335 336 337 338 339 340
# [27] 341 342 343 344 345 346 353 354 355 393
```

Now we should remove also the observations from **Genotype_20** which were removed from Phenotype (index).

```
Genotype_4<-Genotype_20[-index,]
Genotype_4<-Genotype_4[,-1] # removing the first column on scanID
dim(Genotype_4)
```

```
## [1]   377 34922
```

```
# [1] 377 34922
```

We should also remove from **chromosome** the same observations removed above (index) get column names from **Genotype_4** and extract from Chromosome data only those remained after transformations.

```
names_chro<-names(Genotype_4)# SNPs names after removing and substitute missing data with 1
chromosome_1<-data.frame(SNPs_name=names(Genotype[-1]),chromosome) # attaching the SNPs names to chromo

chromosome_4<-chromosome_1 %>% filter(SNPs_name %in% names_chro)
dim(chromosome_4)
```

```
## [1] 34922     5
```

```
# [1] 34922     5
# At this point the dimensions of our datasets are:
# Genotype_4    377x34922
# Pheno_data_4  377x4
# Chromosome_4  34922 x5
```

Question 3

Conduct an appropriate single SNP analysis to first screen SNPs that are associated with: **(i) "Seed.Length"**, **(ii) "Seed.width"**, and **(iii) "Seed.volume"**.

You may use the **assocRegression()** in the R package GWASTools or just appropriate base R function with loops. Use an FDR of 0.001 as the threshold. How many SNPs were associated with each of the three phenotype traits?

Solution 3

First we create a dataset with all information.

There are two main types of study designs:

Case Control Study Design: The phenotype is a binary outcome.

Quantitative Study Design: The phenotype is a continuous outcome.

Our variables: Seed.length, Seed.width and Seed.volume are continuous variables. This will be taken into consideration at this step when modeling (type of model linear will be selected and also family gaussian).

```
library(GWASTools)
```

```
## Loading required package: Biobase

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
###Creating MatrixGenotypeReader class object
genoData<-MatrixGenotypeReader(genotype=t(Genotype_4),snpID=as.integer(chromosome_4$SNP_id),chromosome=a
```

```
snp_info<-SnpAnnotationDataFrame(data=data.frame(snpID=as.integer(chromosome_4$SNP_id),chromosome=as.in
```

Removing the first column since it is the scanID for SNPs name An additional column names is created that contains the SNP names.

```
pheno_data<-ScanAnnotationDataFrame(data.frame(scanID=1:377,Pheno_data4))
#Phenotype used here is the one after transformations we created before (Pheno_data4).
```

```
#Combining all three datatypes.
all<-GenotypeData(genoData,scanAnnot=pheno_data,snpAnnot=snp_info)
```

FDR and threshold

The FDR is the rate that features called significant are truly null. An FDR of 5% means that, among all features called significant, 5% of these are truly null. Below a threshold 0.001 for FDR test is used.

Seed.length FDR

After running the code below we obtain **5509 SNPs** were associated with **Seed.Length** phenotype traits.

```
# Seed.Length
res1 <- assocRegression(all,outcome="Seed.length",model.type="linear")
```

```
## Reading in Phenotype and Covariate Data...

## Running analysis with 377 Samples

## Beginning Calculations...

## Block 1 of 7 Completed - 4.022 secs

## Block 2 of 7 Completed - 3.781 secs

## Block 3 of 7 Completed - 3.756 secs

## Block 4 of 7 Completed - 3.824 secs

## Block 5 of 7 Completed - 3.972 secs

## Block 6 of 7 Completed - 3.78 secs

## Block 7 of 7 Completed - 3.678 secs
```

```
head(res1,n=5)
```

```
##   snpID chr effect.allele       EAF       MAF   n         Est         SE
## 1     1   1             A 0.3037135 0.3037135 377 -0.09496525 0.05316348
## 2     2   1             A 0.3063660 0.3063660 377 -0.09149153 0.05304872
## 3     3   1             A 0.1604775 0.1604775 377 -0.02743348 0.06693424
## 4     4   1             A 0.1379310 0.1379310 377 -0.12004558 0.07081703
## 5     5   1             A 0.3023873 0.3023873 377 -0.09425101 0.05331826
##           LL         UL Wald.Stat  Wald.pval
## 1 -0.1991638 0.00923326  3.190822 0.07405276
## 2 -0.1954651 0.01248204  2.974487 0.08458696
## 3 -0.1586222 0.10375521  0.167983 0.68191067
## 4 -0.2588444 0.01875324  2.873538 0.09004671
## 5 -0.1987529 0.01025086  3.124785 0.07711006
```

```
head(res1[,c(1,3,9,10,11,12)],n=5)### Extracting wald's statistics and p-value
```

```
##   snpID effect.allele         LL         UL Wald.Stat  Wald.pval
## 1     1             A -0.1991638 0.00923326  3.190822 0.07405276
## 2     2             A -0.1954651 0.01248204  2.974487 0.08458696
## 3     3             A -0.1586222 0.10375521  0.167983 0.68191067
## 4     4             A -0.2588444 0.01875324  2.873538 0.09004671
## 5     5             A -0.1987529 0.01025086  3.124785 0.07711006
```

```
p_adj1<-p.adjust(res1[,12], method = "fdr") ###multiple testing correction using fdr
head(data.frame(res1[,c(1,9,10,11,12)],p_adj1<=0.001),n=5)
```

```
##   snpID         LL         UL Wald.Stat  Wald.pval p_adj1....0.001
## 1     1 -0.1991638 0.00923326  3.190822 0.07405276           FALSE
## 2     2 -0.1954651 0.01248204  2.974487 0.08458696           FALSE
## 3     3 -0.1586222 0.10375521  0.167983 0.68191067           FALSE
## 4     4 -0.2588444 0.01875324  2.873538 0.09004671           FALSE
## 5     5 -0.1987529 0.01025086  3.124785 0.07711006           FALSE
```

```
P_adj1<-data.frame(res1[,c(1,9,10,11,12)],p_adj1<=0.001) # control if p_adj<=0.001
sum(P_adj1$p_adj1....0.001==TRUE)# counting how many observations SPNs are significant using threshold=
```

```
## [1] 5509
```

```
# [1] 5509 SNPs were associated with Seed.Length phenotype traits

i_1<-which(P_adj1$p_adj1....0.001==TRUE)# saves the rows of SNPs which are associated with Seed.length
#we will use this index to create a set for question 5, and split it to training and testing. The same
new_data<-data.frame(Pheno_data4,Genotype_4)
set_1<-new_data[,i_1]
dim(set_1)
```

```
## [1]  377 5509
```

```
# Use this set_1 at question 5 for modelling Seed.length and split it in train and test
```

Seed.width FDR

After running the code below we obtain **25299 SNPs** were associated with **Seed.width** phenotype traits.

```
# Seed.width
res2 <- assocRegression(all,outcome="Seed.width",model.type="linear")
```

```
## Reading in Phenotype and Covariate Data...

## Running analysis with 377 Samples

## Beginning Calculations...

## Block 1 of 7 Completed - 3.749 secs

## Block 2 of 7 Completed - 3.759 secs

## Block 3 of 7 Completed - 3.727 secs

## Block 4 of 7 Completed - 3.757 secs

## Block 5 of 7 Completed - 3.746 secs

## Block 6 of 7 Completed - 3.735 secs

## Block 7 of 7 Completed - 3.682 secs
```

```
head(res2,n=10)
```

```
##    snpID chr effect.allele       EAF       MAF   n         Est         SE
## 1      1   1             A 0.3037135 0.3037135 377 -0.11658444 0.02106844
## 2      2   1             A 0.3063660 0.3063660 377 -0.11766855 0.02099661
## 3      3   1             A 0.1604775 0.1604775 377 -0.13255926 0.02661064
## 4      4   1             A 0.1379310 0.1379310 377 -0.05879996 0.02901710
## 5      5   1             A 0.3023873 0.3023873 377 -0.11463519 0.02116120
## 6      6   1             A 0.2015915 0.2015915 377 -0.11917198 0.02607033
## 7      7   1             A 0.1604775 0.1604775 377 -0.13395931 0.02659194
## 8      8   1             A 0.3037135 0.3037135 377 -0.11815092 0.02104515
## 9      9   1             A 0.2970822 0.2970822 377 -0.11596323 0.02148162
## 10    10   1             A 0.1405836 0.1405836 377 -0.05832833 0.02894744
##            LL           UL Wald.Stat    Wald.pval
## 1  -0.1578778 -0.075291063 30.620804 3.137119e-08
## 2  -0.1588212 -0.076515952 31.406709 2.092566e-08
## 3  -0.1847152 -0.080403359 24.814725 6.311349e-07
## 4  -0.1156724 -0.001927498  4.106258 4.272481e-02
## 5  -0.1561104 -0.073160011 29.346445 6.052744e-08
## 6  -0.1702689 -0.068075074 20.895622 4.850019e-06
## 7  -0.1860786 -0.081840073 25.377326 4.714267e-07
## 8  -0.1593987 -0.076903179 31.518830 1.975156e-08
## 9  -0.1580664 -0.073860019 29.141132 6.729306e-08
## 10 -0.1150643 -0.001592391  4.060119 4.390740e-02
```

```
head(res2[,c(1,3,9,10,11,12)],n=3)# Extracting wald's statistics and p-value
```

```
##   snpID effect.allele         LL          UL Wald.Stat    Wald.pval
## 1     1             A -0.1578778 -0.07529106  30.62080 3.137119e-08
## 2     2             A -0.1588212 -0.07651595  31.40671 2.092566e-08
## 3     3             A -0.1847152 -0.08040336  24.81472 6.311349e-07
```

```
p_adj2<-p.adjust(res2[,12], method = "fdr") #multiple testing correction using fdr
head(data.frame(res2[,c(1,9,10,11,12)],p_adj2),n=3)
```

```
##   snpID         LL          UL Wald.Stat    Wald.pval       p_adj2
## 1     1 -0.1578778 -0.07529106  30.62080 3.137119e-08 7.132451e-08
## 2     2 -0.1588212 -0.07651595  31.40671 2.092566e-08 4.837267e-08
## 3     3 -0.1847152 -0.08040336  24.81472 6.311349e-07 1.252301e-06
```

```
P_adj2<-data.frame(res2[,c(1,9,10,11,12)],p_adj2<=0.001) # control if p_adj<=0.001
sum(P_adj2$p_adj2....0.001==TRUE)# counting how many observations SPNs are significant using threshold=
```

```
## [1] 25299
```

```
# [1] 25299 SNPs were associated with Seed.width phenotype traits

i_2<-which(P_adj2$p_adj2....0.001==TRUE)# saves the rows of SNPs which are associated with Seed.width
#we will use this index to create a set for question 5, and split it to training and testing. Same for
new_data<-data.frame(Pheno_data4,Genotype_4)

set_2<-new_data[,i_2]
dim(set_2)
```

```
## [1]   377 25299
```

```
# Use this set_2 at question 5 for modelling Seed.width and split it in train and test
```

Seed.volume FDR

After running the code below we obtain **26088 SNPs** were associated with **Seed.volume** phenotype traits.

```
# Seed.volume
res3 <- assocRegression(all,outcome="Seed.volume",model.type="linear")
```

```
## Reading in Phenotype and Covariate Data...

## Running analysis with 377 Samples

## Beginning Calculations...

## Block 1 of 7 Completed - 3.949 secs
```

```
## Block 2 of 7 Completed - 3.705 secs

## Block 3 of 7 Completed - 3.735 secs

## Block 4 of 7 Completed - 3.734 secs

## Block 5 of 7 Completed - 3.71 secs

## Block 6 of 7 Completed - 3.713 secs

## Block 7 of 7 Completed - 3.698 secs
```

```
head(res3,n=10)
```

```
##    snpID chr effect.allele       EAF       MAF   n         Est         SE
## 1      1   1             A 0.3037135 0.3037135 377 -0.08453868 0.01277524
## 2      2   1             A 0.3063660 0.3063660 377 -0.08453832 0.01274037
## 3      3   1             A 0.1604775 0.1604775 377 -0.09154618 0.01625604
## 4      4   1             A 0.1379310 0.1379310 377 -0.04502567 0.01782501
## 5      5   1             A 0.3023873 0.3023873 377 -0.08322037 0.01283851
## 6      6   1             A 0.2015915 0.2015915 377 -0.09112489 0.01581920
## 7      7   1             A 0.1604775 0.1604775 377 -0.09269467 0.01623868
## 8      8   1             A 0.3037135 0.3037135 377 -0.08507693 0.01276570
## 9      9   1             A 0.2970822 0.2970822 377 -0.08364323 0.01304363
## 10    10   1             A 0.1405836 0.1405836 377 -0.04575967 0.01777539
##              LL          UL Wald.Stat    Wald.pval
## 1  -0.10957768 -0.05949967 43.789804 3.656046e-11
## 2  -0.10950898 -0.05956766 44.029453 3.234716e-11
## 3  -0.12340743 -0.05968493 31.713993 1.786305e-08
## 4  -0.07996205 -0.01008930  6.380592 1.153750e-02
## 5  -0.10838338 -0.05805737 42.017515 9.045953e-11
## 6  -0.12212995 -0.06011983 33.182201 8.391514e-09
## 7  -0.12452189 -0.06086745 32.584297 1.141332e-08
## 8  -0.11009725 -0.06005661 44.415446 2.655834e-11
## 9  -0.10920827 -0.05807820 41.121123 1.430817e-10
## 10 -0.08059879 -0.01092054  6.627159 1.004353e-02
```

```
head(res3[,c(1,3,9,10,11,12)],n=3)### Extracting wald's statistics and p-value
```

```
##   snpID effect.allele         LL          UL Wald.Stat    Wald.pval
## 1     1             A -0.1095777 -0.05949967  43.78980 3.656046e-11
## 2     2             A -0.1095090 -0.05956766  44.02945 3.234716e-11
## 3     3             A -0.1234074 -0.05968493  31.71399 1.786305e-08
```

```
p_adj3<-p.adjust(res3[,12], method = "fdr") ###multiple testing correction using fdr
head(data.frame(res3[,c(1,9,10,11,12)],p_adj3),n=3)
```

```
##   snpID         LL          UL Wald.Stat    Wald.pval       p_adj3
## 1     1 -0.1095777 -0.05949967  43.78980 3.656046e-11 9.903541e-11
## 2     2 -0.1095090 -0.05956766  44.02945 3.234716e-11 8.798408e-11
## 3     3 -0.1234074 -0.05968493  31.71399 1.786305e-08 3.803972e-08
```

```
P_adj3<-data.frame(res3[,c(1,9,10,11,12)],p_adj3<=0.001) # control if p_adj<=0.001
sum(P_adj3$p_adj3....0.001==TRUE)# counting how many observations SPNs are significant using threshold=
```

```
## [1] 26088
```

```
# [1] 26088 SNPs were associated with Seed.volume phenotype traits

i_3<-which(P_adj3$p_adj3....0.001==TRUE)# saves the rows of SNPs which are associated with Seed.volume
#we will use this index to create a set for question 5, and split it to training and testing. Same for
new_data<-data.frame(Pheno_data4,Genotype_4)

set_3<-new_data[,i_3]
dim(set_3)
```

```
## [1]   377 26088
```

```
# Use this set_3 at question 5 for modelling Seed.volume and split it in train and test
```

Reference:

https://cran.r-project.org/web/packages/glmnet/vignettes/glmnetFamily.pdf; https://glmnet.stanford.edu/articles/glmnet.html

Question 4

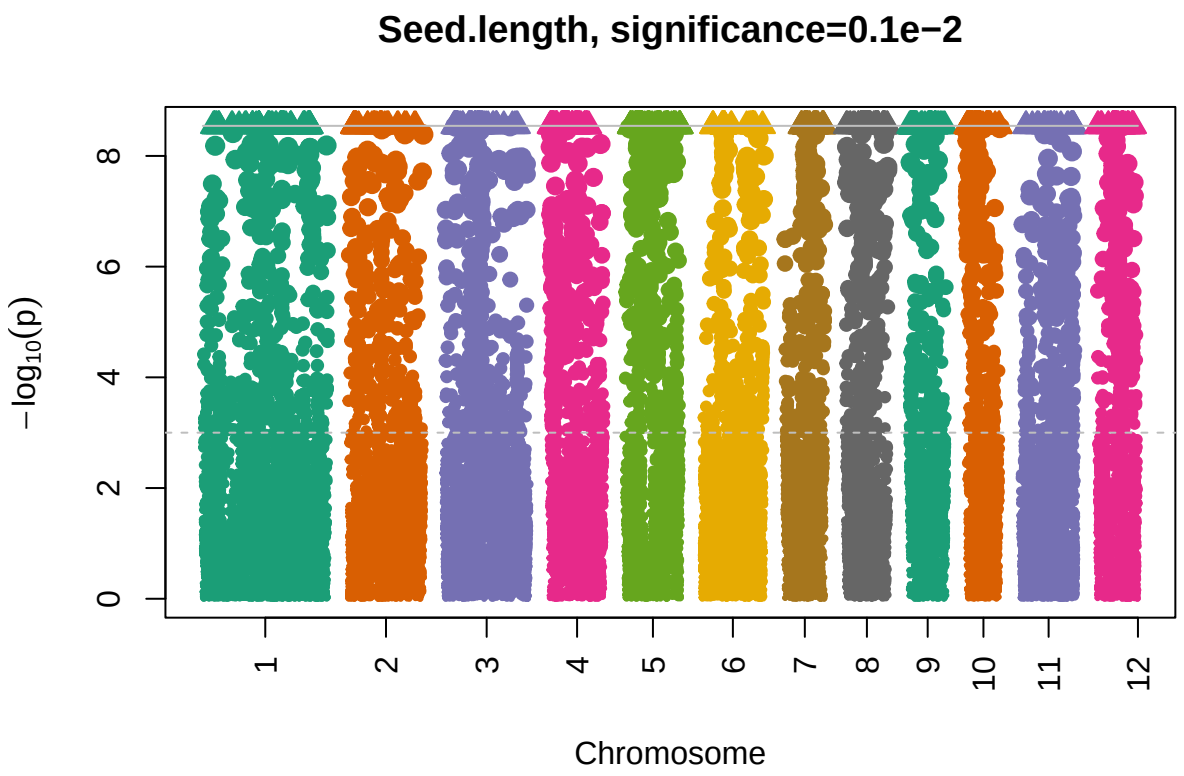
Create a Manhattan plot for all three phenotypes separately. ##Solution 4

A Manhattan plot is a type of plot, usually used to display data with a large number of data-points, many of non-zero amplitude, and with a distribution of higher-magnitude values. The plot is commonly used in genome-wide association studies (GWAS) to display significant SNPs. Below are the Manhattan plot for each of the three variables: seed.length, seed.width and seed.volume with respect to the transformed dataset (after removing 20% of NAs and rows with missing values as well). For each case two significance levels are used for comparison purposes. In both cases the differences are smaller.

Manhattan plot Seed.Length

Code for the dataset obtained at Step 3.

```
# Manhattan plot

# Seed.Length
pvals1<-p_adj1
chromosome <- chromosome_4$Chromosome # length 34922
manhattanPlot(pvals1, chromosome, signif=0.001,main="Seed.length, significance=0.1e-2")
```

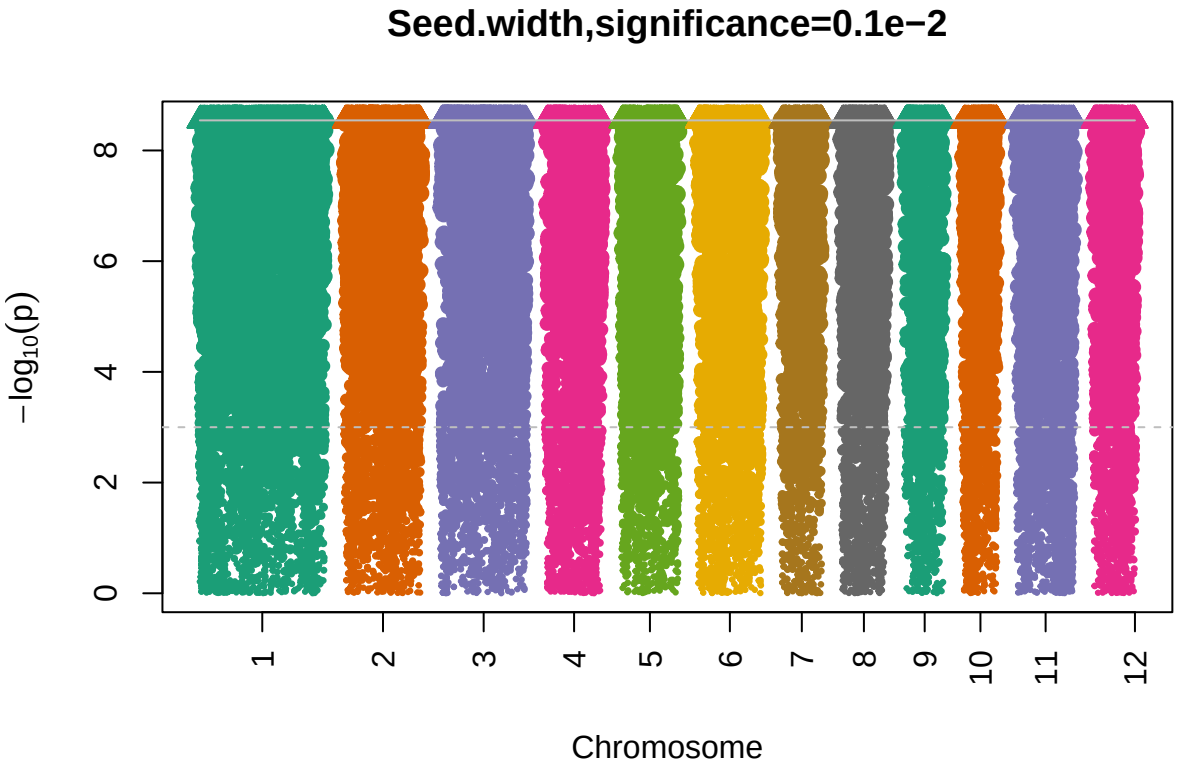


```
log10(0.001)
```

```
## [1] -3
```

Manhattan plot Seed.width

```
# Manhattan plot
# Seed.width
pvals2<-p_adj2
chromosome <- chromosome_4$Chromosome # length 34922
manhattanPlot(pvals2, chromosome, signif=0.001,main="Seed.width,significance=0.1e-2")
```

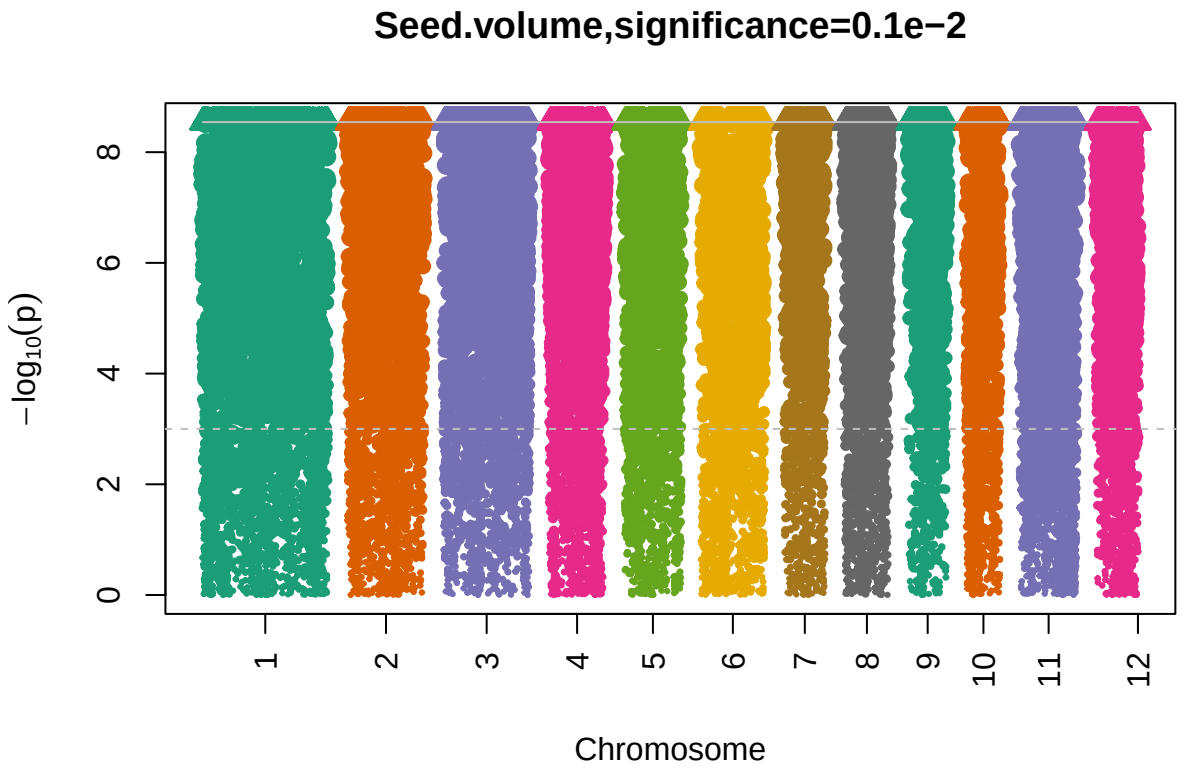



```
log10(0.001)
```

```
## [1] -3
```

Manhattan plot Seed.volume

```r
# Manhattan plot
# Seed.Length
pvals3<-p_adj3
chromosome <- chromosome_4$Chromosome # length 34922
manhattanPlot(pvals3, chromosome, signif=0.001,main="Seed.volume,significance=0.1e-2")
```




```r
log10(0.001)
```

```
## [1] -3
```

Question 5

Split the data into training and test set such that about 80% of your data is in the training set. Using the training set, build a multivariate model using LASSO or SPLS for each of the three traits separately (using univariate Y as the response). Make sure you choose the tuning parameter appropriately. For each of the trait, provide a summary of the number of SNPs with non-zero coefficients, and the performance on the training and test set. You may use root mean square error or the square of the correlation of the predicted and observed value (i.e. R2) as a measure for your model performance.

Solution 5

Training and Testing data

Create a new dataset including all information from **Genotype** and **pheno data**. This will be our data from which we will create a **train** and **test** dataset. The training set will be obtained from the subset created at question 3 for each traits (Seed.length, Seed.width and Seed.volume).

```
# set_1 is our subset of SNPs for seed.length obtained after question 3 analysis
# create a train and test dataset
sample_size <- floor(0.8 * nrow(set_1))
sample_size
```

```
## [1] 301
```

```
## set the seed to make your partition reproducible
set.seed(512)
train_ind<- sample(seq_len(nrow(set_1)), size = sample_size)

# now our train and testing data are
train_1 <- set_1[train_ind, ]
test_1 <- set_1[-train_ind, ]

dim(train_1)
```

```
## [1]  301 5509
```

```
dim(test_1)
```

```
## [1]   76 5509
```

The graphical results obtained below show: Each curve which corresponds to a variable. It shows the path of its coefficient against the L1-norm of the whole coefficient vector as lambda varies. The axis above indicates the number of nonzero coefficients at the current lambda, which is the effective degrees of freedom (df) for the lasso.

Model 1- Seed.Width LASSO model evaluation

Trying Poisson and gaussian gives some visible differences on the nonzero coefficients. Below we will continue with **gaussian** family considering our data are continuous data. For the **Seed.length** we observe from the graphical output of the model that the number of nonzero coefficients at the current log(lambda) approximately -0.5 is one.

```
library(glmnet)
```

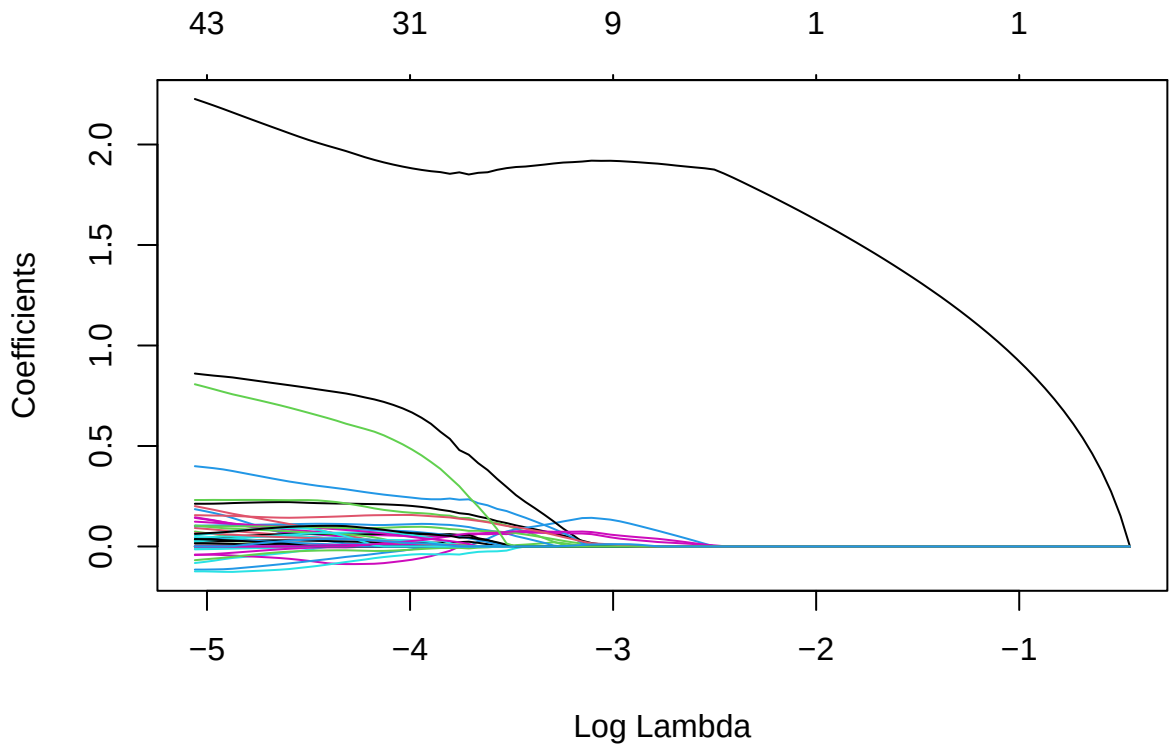
```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

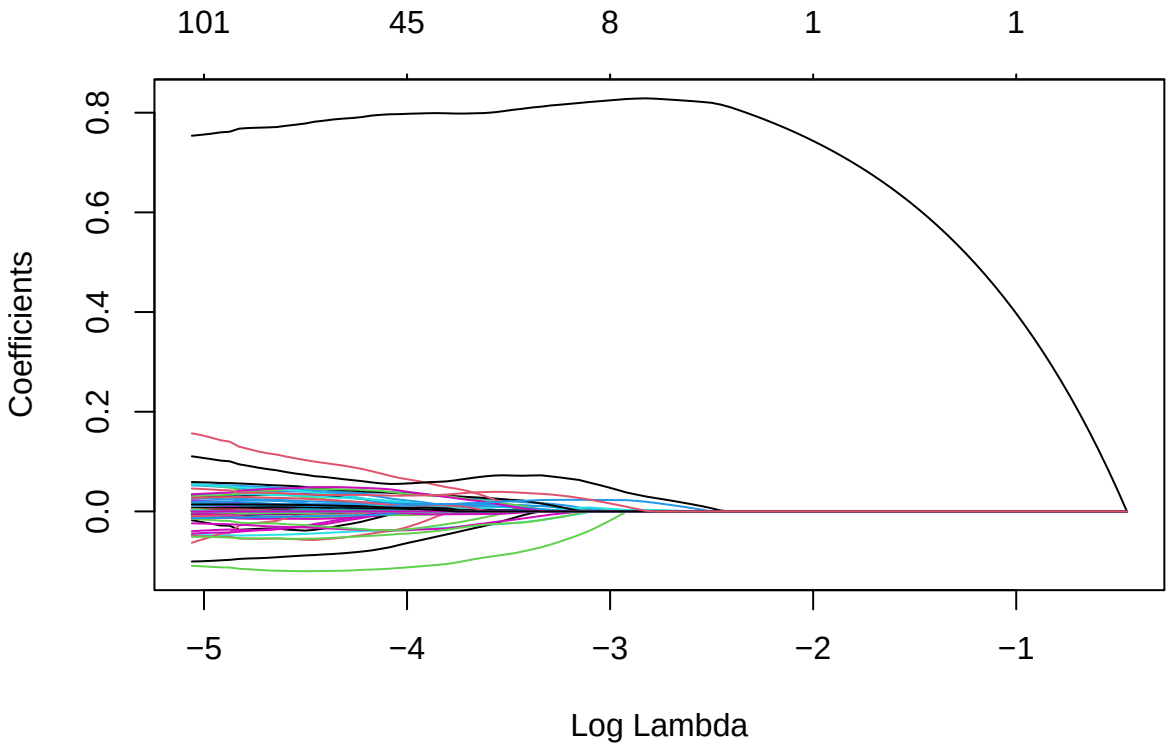
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-4
```

```
mod1<-glmnet(y=train_1[,1],x=as.matrix(train_1[,-c(1,2,3)]),family="poisson")
plot(mod1,xvar="lambda")
```




```
mod1<-glmnet(y=train_1[,1],x=as.matrix(train_1[,-c(1,2,3)]),family="gaussian")
plot(mod1,xvar="lambda")
```



We may observe that most of the variables follow typical regularization paths and shrink to zero eventually. Trying cv.glmnet() function below:

```
set.seed(12345)
cv_mod1<-cv.glmnet(x=as.matrix(train_1[,-c(1,2,3)]),y=train_1[,1],family="gaussian")
#plot(cv_mod1)
cv_mod1$lambda.min ## Lambda that gives minimum prediction error
```

```
## [1] 0.04477315
```

```
mod1<-glmnet(y=train_1[,1],x=as.matrix(train_1[,-c(1,2,3)]),lambda=cv_mod1$lambda.min,family="poisson")
head(coef(mod1),10)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) -3.734645
## SNP54        .
## SNP55        1.919810
## SNP175       .
## SNP196       .
## SNP198       .
## SNP200       .
## SNP205       .
## SNP225       .
## SNP227       .
```

The table output of the model gives information about: nonzero coefficients (Df), the percent (of null) deviance explained (%dev) and the value of (Lambda)

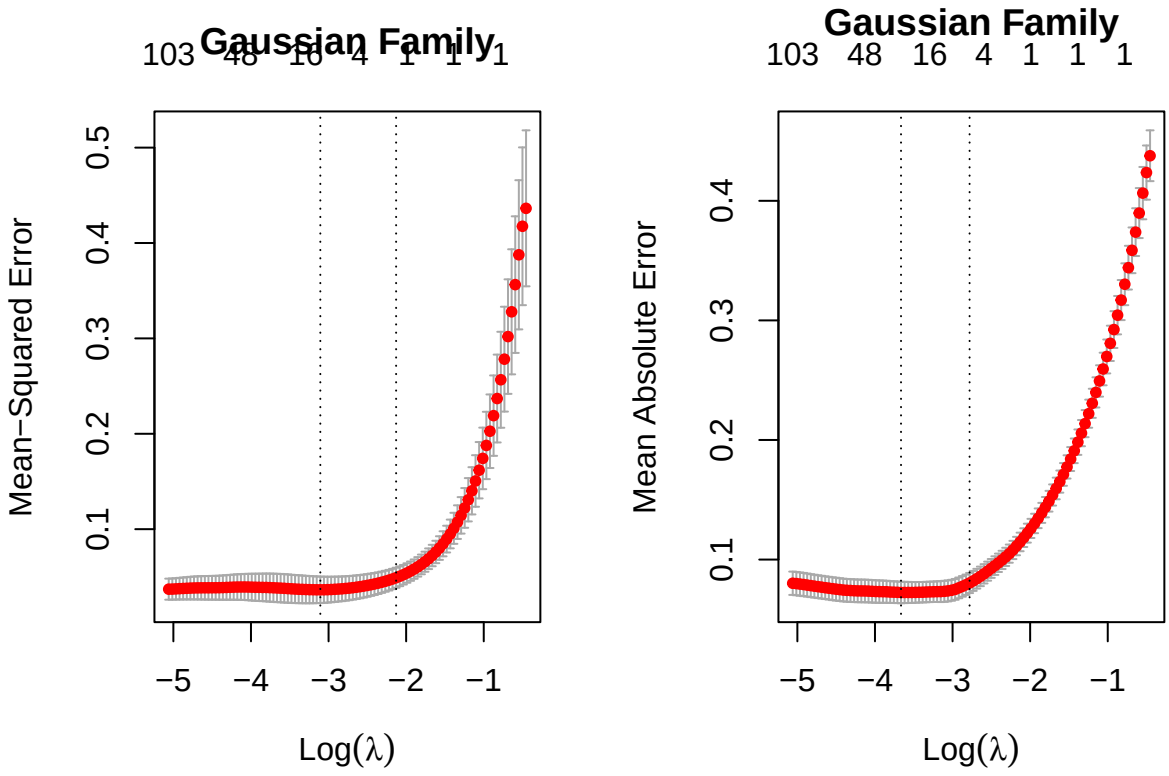
```
mod_1_fit<-data.frame(DF=cv_mod1$glmnet.fit$df, Dev_perc=cv_mod1$glmnet.fit$dev.ratio, Lambda=cv_mod1$gl
head(mod_1_fit)
```

```
##   DF   Dev_perc    Lambda
## 1  0 0.00000000 0.6346479
## 2  1 0.08233494 0.6058021
## 3  1 0.15735548 0.5782675
## 4  1 0.22571139 0.5519844
## 5  1 0.28799476 0.5268958
## 6  1 0.34474504 0.5029476
```

```
# cv_mod1$glmnet.fit #for all info just use the direct call
```

We may obtain the plot of the cross-validation curve (red dotted line) along with upper and lower standard deviation curves along the lambda sequence (error bars) as shown below. Graphs below are the same but in respect to **MAE-Mean Absolute Error** and **MSE-Mean Squared error**. Depending on the error used for the Seed.length the number of coefficients nonzero **varies from 2 to 11 based on MSE, and from 11 to 23 based on MAE.** based on the visualization but also observe the cross-validation results below for an accurate result in numbers.

```
par(mfrow=c(1,2))
plot(cv_mod1)
title("Gaussian Family", line = 1.5)
#
set.seed(1011)
mod1.1 = cv.glmnet(x=as.matrix(train_1[,-c(1,2,3)]),y=train_1[,1], type.measure = "mae")
plot(mod1.1)
title("Gaussian Family", line = 2)
```



lambda.min is the value of lambda that gives minimum mean cross-validated error. We can use the following code to get the value of *lambda.min* and the model coefficients at that value of lambda:

```
cv_mod1$lambda.min
```

```
## [1] 0.04477315
```

```
log(cv_mod1$lambda.min)# observe the first bar to the left this is  the value of log() for the minimum
```

```
## [1] -3.106147
```

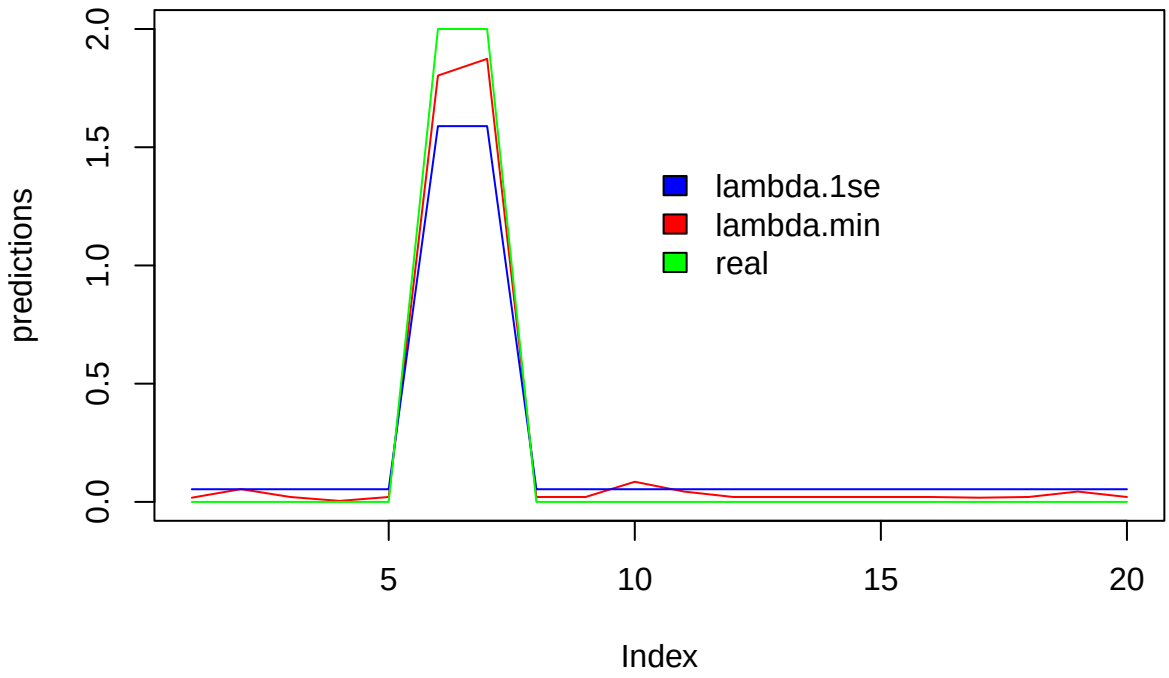
While *lambda.1se* is the value of lambda that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.

```
Pred_1<-predict(cv_mod1, newx = as.matrix(test_1[1:20,-c(1,2,3)]), type="response",s =c(cv_mod1$lambda.m
Pred_1
```

```
##             s1         s2
## 2   0.017744026 0.05351553
## 3   0.054017084 0.05351553
## 8   0.020757140 0.05351553
## 16  0.004130259 0.05351553
## 18  0.020683748 0.05351553
## 30  1.802960883 1.58920339
```

```
## 44  1.873739980 1.58920339
## 45  0.020683748 0.05351553
## 58  0.020683748 0.05351553
## 64  0.085112968 0.05351553
## 67  0.043758251 0.05351553
## 70  0.020683748 0.05351553
## 80  0.020683748 0.05351553
## 82  0.020683748 0.05351553
## 87  0.020683748 0.05351553
## 94  0.020683748 0.05351553
## 95  0.017744026 0.05351553
## 98  0.020683748 0.05351553
## 106 0.043758251 0.05351553
## 115 0.020683748 0.05351553
```

```
# To compare the predictions graphically we may also plot them
plot(Pred_1[,1],col="red",type="l",ylab="predictions",ylim=c(0,2))
lines(Pred_1[,2],col="blue")
lines(test_1[1:20,1],col="green")
legend(10,1.5,c("lambda.1se","lambda.min","real"),fill=c("blue","red","green"),box.col="white")
```



A summary may be also obtained if we decide the number of cross-validation mean squared error criterion (mse) or mean absolute error (mae).

```
cv.glmnet(x=as.matrix(train_1[,-c(1,2,3)]),y=train_1[,1], type.measure = "mse",nfolds = 20)
```

```
##
## Call:  cv.glmnet(x = as.matrix(train_1[, -c(1, 2, 3)]), y = train_1[,      1], type.measure = "mse",
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.00635   100 0.03540 0.01242     103
## 1se 0.11892    37 0.04748 0.01296       1
```

Model performance functions RMSE and RSquare

```
# Root Mean Squared Error (RMSE)
RMSE = function(x,y) {
  sqrt(mean((x-y)^2))
}
# R SQUARED error metric -- Coefficient of Determination
R_square = function(y_actual,y_predict){
  cor(y_actual,y_predict)^2
}
```

Model 1 performance functions RMSE and RSquare

```
# Training Accuracy
Train_pred<-predict(cv_mod1, newx = as.matrix(train_1[1:301,-c(1,2,3)]), type="response",s =cv_mod1$lam

RMSE(train_1[1:301,1],Train_pred)
```

```
## [1] 0.1699992
```

```
R_square(train_1[1:301,1],Train_pred)
```

```
##             s1
## [1,] 0.9388146
```

```
# Testing Accuracy
Pred_1_all<-predict(cv_mod1, newx = as.matrix(test_1[,-c(1,2,3)]), type="response",s =cv_mod1$lambda.mi
RMSE(test_1[,1],Pred_1_all)
```

```
## [1] 0.3288517
```

```
R_square(test_1[,1],Pred_1_all)
```

```
##             s1
## [1,] 0.8343956
```

```
# A list of outputs for training and testing
list(RMSE=c(Train=RMSE(train_1[,1],Train_pred),Test=RMSE(test_1[,1],Pred_1_all)),R_square=c(Train=R_squa
```

```
## $RMSE
##     Train      Test
## 0.1699992 0.3288517
##
## $R_square
##     Train      Test
## 0.9388146 0.8343956
```

Provide a summary of the number of SNPs with non-zero coefficients. 1 out the 5509 SNPs had non-zero coefficients and magnitude and sign of the coefficient will provide a measure of its effect and direction to **Seed.length**.

```
length(which(as.matrix(coef(cv_mod1))!=0))-1 ##Subtracting 1 for intercept
```

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

Model 2- Seed.Width LASSO model evaluation

Again here our Seed.Width is a continuous variable so we use "gaussian".

```
# set_2 is our subset of SNPs for seed.width obtained after question 3 analysis
# create a train and test dataset
sample_size <- floor(0.8 * nrow(set_2))
sample_size
```

```
## [1] 301
```

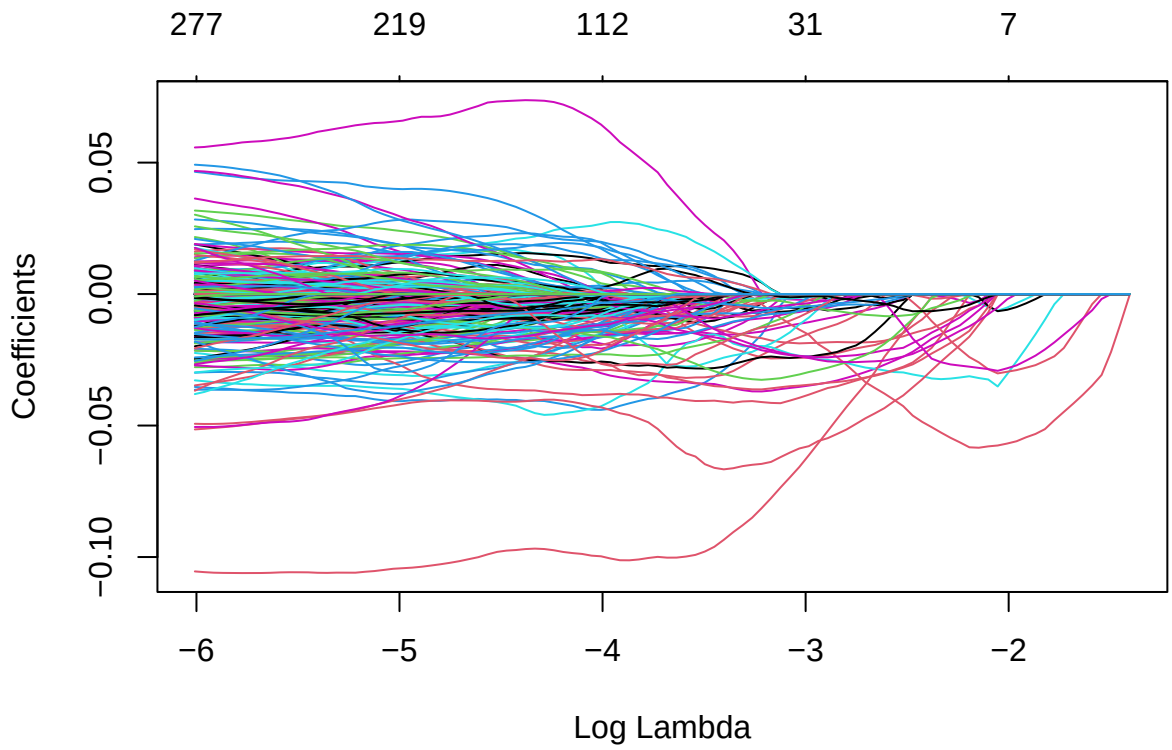
```
## set the seed to make your partition reproducible
set.seed(1234)
train_ind<- sample(seq_len(nrow(set_2)), size = sample_size)

# now our train and testing data are
train_2 <- set_2[train_ind, ]
test_2 <- set_2[-train_ind, ]

dim(train_2)
```

```
## [1]   301 25299
```

```
dim(test_2)
```

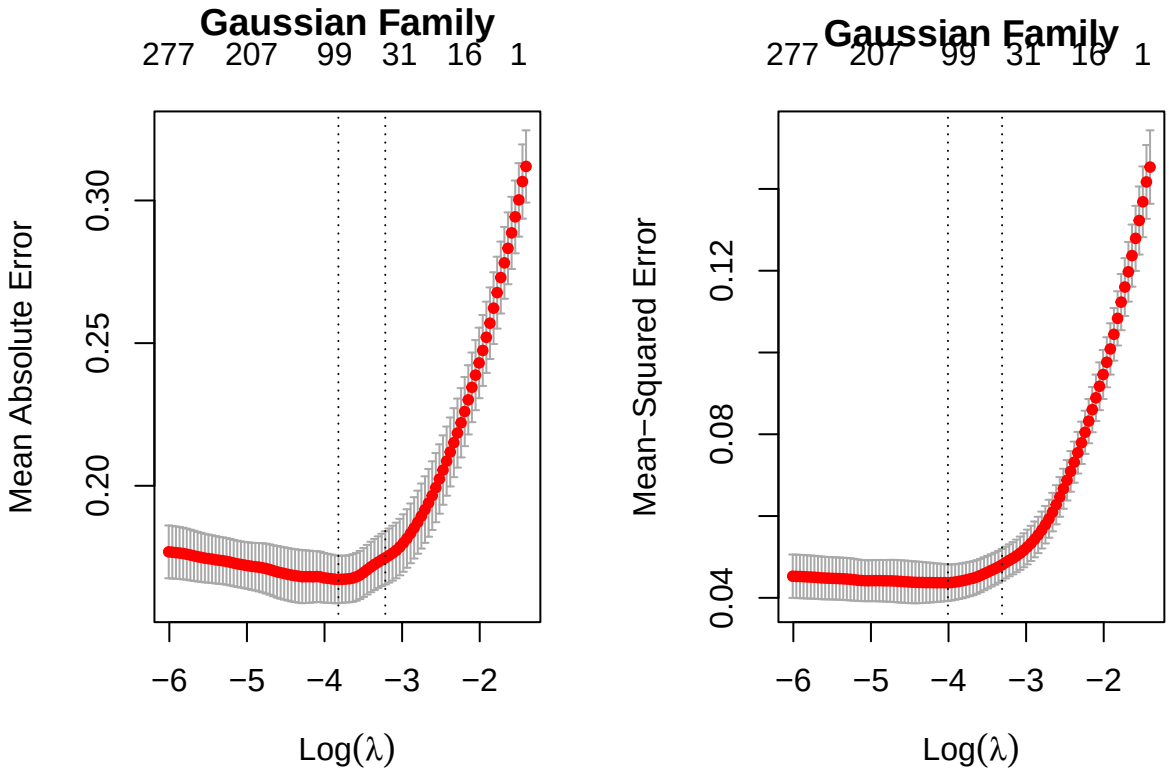
```
## [1]    76 25299
```

```
library(glmnet)


mod2<-glmnet(y=train_2[,2],x=as.matrix(train_2[,-c(1,2,3)]),family="gaussian")
plot(mod2,xvar="lambda")

# we may also organize the limits for y axis
mod2<-glmnet(y=train_2[,2],x=as.matrix(train_2[,-c(1,2,3)]),family="gaussian",lower.limits = -0.3, uppe
plot(mod2,xvar="lambda")
```



Same for model 2, we may observe that most of the variables follow typical regularization paths and shrink to zero eventually. We may obtain the plot of the cross-validation curve (red dotted line) along with upper and lower standard deviation curves along the lambda sequence (error bars) as shown below. The number of **SNPs with coefficients nonzero vary from 43 up to 115** from the visualization but also observe the cross-validation results below for an accurate result in numbers.

```
par(mfrow=c(1,2))
set.seed(1011)
cv_mod2 = cv.glmnet(x=as.matrix(train_2[,-c(1,2,3)]),y=train_2[,2], type.measure = "mae")
plot(cv_mod2)
title("Gaussian Family", line = 2)

mod2.1 = cv.glmnet(x=as.matrix(train_2[,-c(1,2,3)]),y=train_2[,2], type.measure = "mse")
plot(mod2.1)
title("Gaussian Family", line = 1.7)
```



The table output of the model gives information about: nonzero coefficients (Df), the percent (of null) deviance explained (%dev) and the value of lambda (Lambda)

```
mod_2_fit<-data.frame(DF=cv_mod2$glmnet.fit$df, Dev_perc=cv_mod2$glmnet.fit$dev.ratio, Lambda=cv_mod2$gl
head(mod_2_fit)
```

```
##   DF   Dev_perc    Lambda
## 1  0 0.00000000 0.2459280
## 2  1 0.03698982 0.2347502
## 3  1 0.07069357 0.2240805
## 4  2 0.10246271 0.2138957
## 5  3 0.13626674 0.2041738
## 6  3 0.16911555 0.1948938
```

```
# cv_mod2$glmnet.fit #for all info just use the direct call
```

lambda.min is the value of lambda that gives minimum mean cross-validated error. We can use the following code to get the value of *lambda.min* and the model coefficients at that value of lambda:

```
cv_mod2$lambda.min
```

```
## [1] 0.02189288
```

```
log(cv_mod2$lambda.min)# observe the first bar to the left this is  the value of log() for the minimum
```

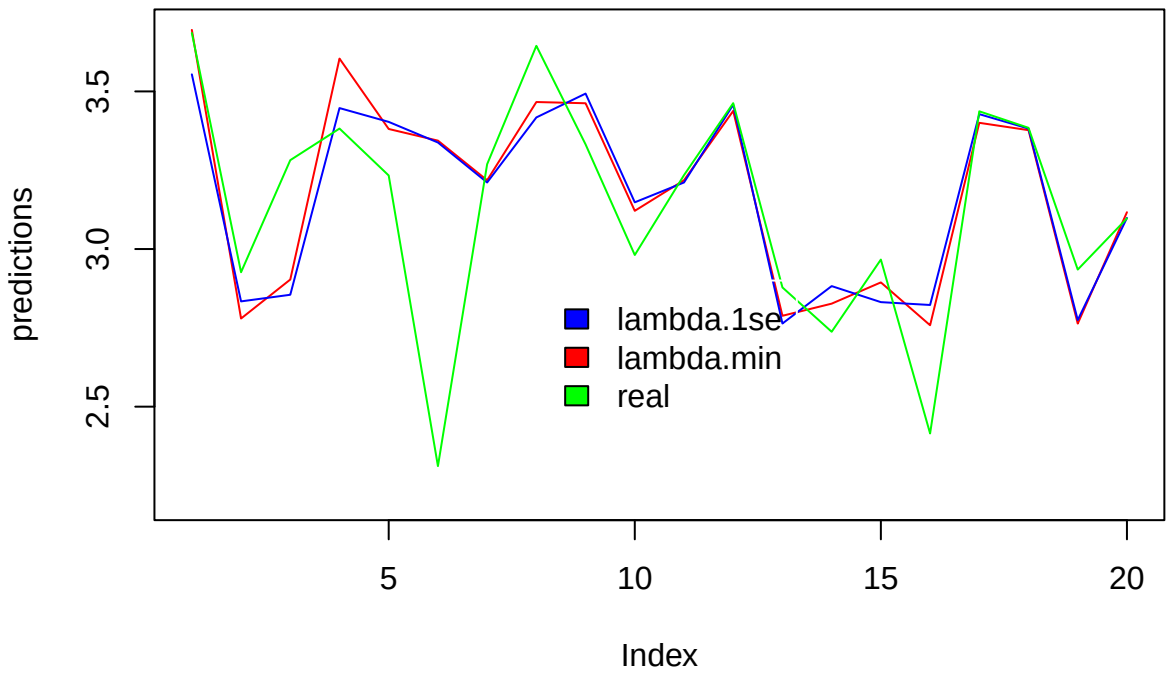
```
## [1] -3.821594
```

While *lambda.1se* is the value of lambda that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.

```
Pred_2<-predict(cv_mod2, newx = as.matrix(test_2[1:20,-c(1,2,3)]), type="response",s =c(cv_mod2$lambda.m
Pred_2
```

```
##           s1       s2
## 1   3.695124 3.554147
## 3   2.779546 2.833908
## 5   2.903199 2.854961
## 8   3.604132 3.447085
## 9   3.380776 3.403265
## 12  3.343808 3.337915
## 20  3.218111 3.211146
## 23  3.466300 3.417435
## 25  3.462454 3.493040
## 36  3.121209 3.148196
## 49  3.218111 3.211146
## 51  3.438158 3.457870
## 56  2.788271 2.763765
## 62  2.826781 2.882389
## 71  2.894173 2.831563
## 74  2.758251 2.822517
## 75  3.400325 3.427823
## 76  3.377113 3.381969
## 85  2.763556 2.774779
## 101 3.116708 3.098683
```

```
# To compare the predictions graphically we may also plot them
plot(Pred_2[,1],col="red",type="l",ylab="predictions",ylim=c(2.2,3.7))
lines(Pred_2[,2],col="blue")
lines(test_2[1:20,2],col="green")
legend(8,2.9,c("lambda.1se","lambda.min","real"),fill=c("blue","red","green"),box.col="white")
```



A summary may be also obtained if we decide the number of cross-validation mean squared error criterion (mse) or mean absolute error (mae).

```
cv.glmnet(x=as.matrix(train_2[,-c(1,2,3)]),y=train_2[,2], type.measure = "mse",nfolds = 20)
```

```
##
## Call:  cv.glmnet(x = as.matrix(train_2[, -c(1, 2, 3)]), y = train_2[,      2], type.measure = "mse",
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.01995    55 0.04488 0.004533     101
## 1se 0.03652    42 0.04937 0.004076      49
```

Model 2-Seed.Width performance:

```
# Training Accuracy
Train_pred<-predict(cv_mod2, newx = as.matrix(train_2[1:301,-c(1,2,3)]), type="response",s =cv_mod2$lamb

RMSE(train_2[,2],Train_pred)
```

```
## [1] 0.1455934
```

```
R_square(train_2[,2],Train_pred)
```

```
##             s1
## [1,] 0.8747202
```

```
# Testing Accuracy
Pred_2_all<-predict(cv_mod2, newx = as.matrix(test_2[1:76,-c(1,2,3)]), type="response",s =cv_mod2$lambda
RMSE(test_2[,2],Pred_2_all)
```

```
## [1] 0.2314261
```

```
R_square(test_2[,2],Pred_2_all)
```

```
##             s1
## [1,] 0.6985782
```

```
# A list of outputs for training and tetsing
list(RMSE=c(Train=RMSE(train_2[,2],Train_pred),Test=RMSE(test_2[,2],Pred_2_all)),R_square=c(Train=R_squa
```

```
## $RMSE
##     Train      Test
## 0.1455934 0.2314261
##
## $R_square
##     Train      Test
## 0.8747202 0.6985782
```

Provide a summary of the number of SNPs with non-zero coefficients. 43 out the 25296 SNPs had non-zero coefficients and magnitude and sign of the coefficient will provide a measure of its effect and direction to Seed.length.

```
length(which(as.matrix(coef(cv_mod2))!=0))-1 ##Subtracting 1 for intercept
```

```
## [1] 43
```

Model 3- Seed.Volume LASSO model evaluation

Trying Poisson and gaussian does not give a clear difference. So, we continue with gaussian.

```
# set_3 is our subset of SNPs for seed.volume obtained after question 3 analysis
# create a train and test dataset
sample_size <- floor(0.8 * nrow(set_3))
sample_size
```

```
## [1] 301
```

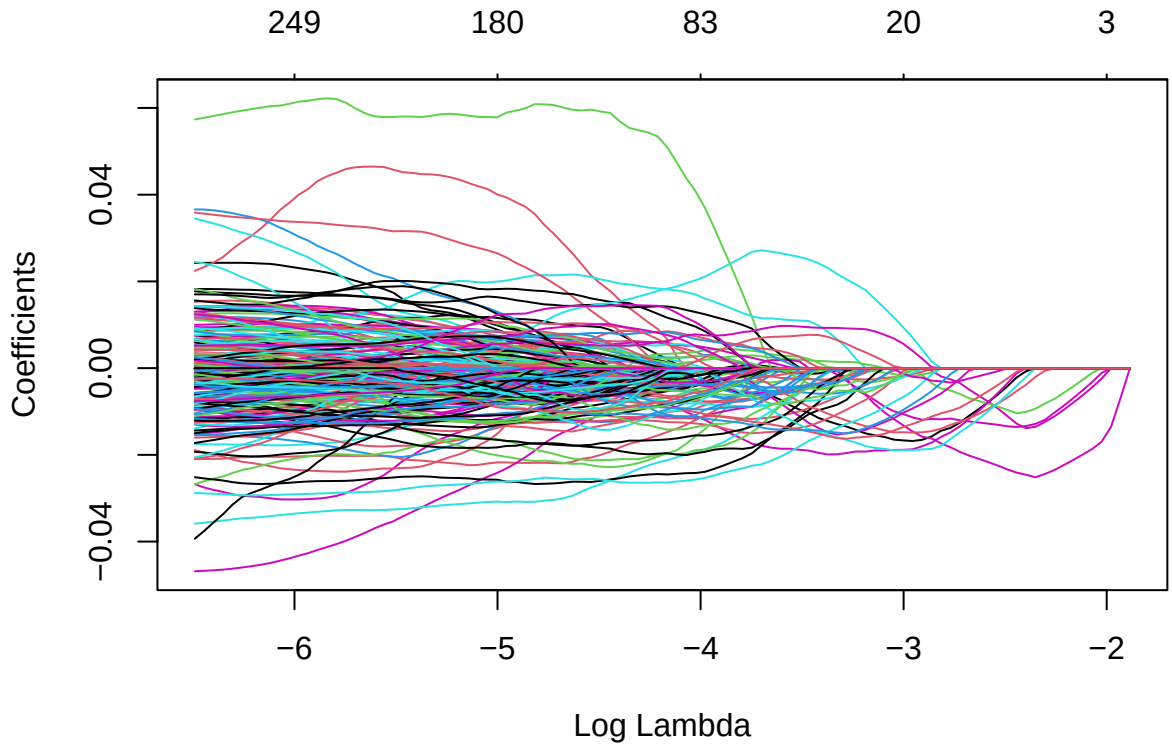
```
## set the seed to make your partition reproducible
set.seed(1234)
train_ind<- sample(seq_len(nrow(set_3)), size = sample_size)

# now our train and testing data are
train_3 <- set_3[train_ind, ]
test_3 <- set_3[-train_ind, ]

dim(train_3)
```

```
## [1]   301 26088
```

```
dim(test_3)
```

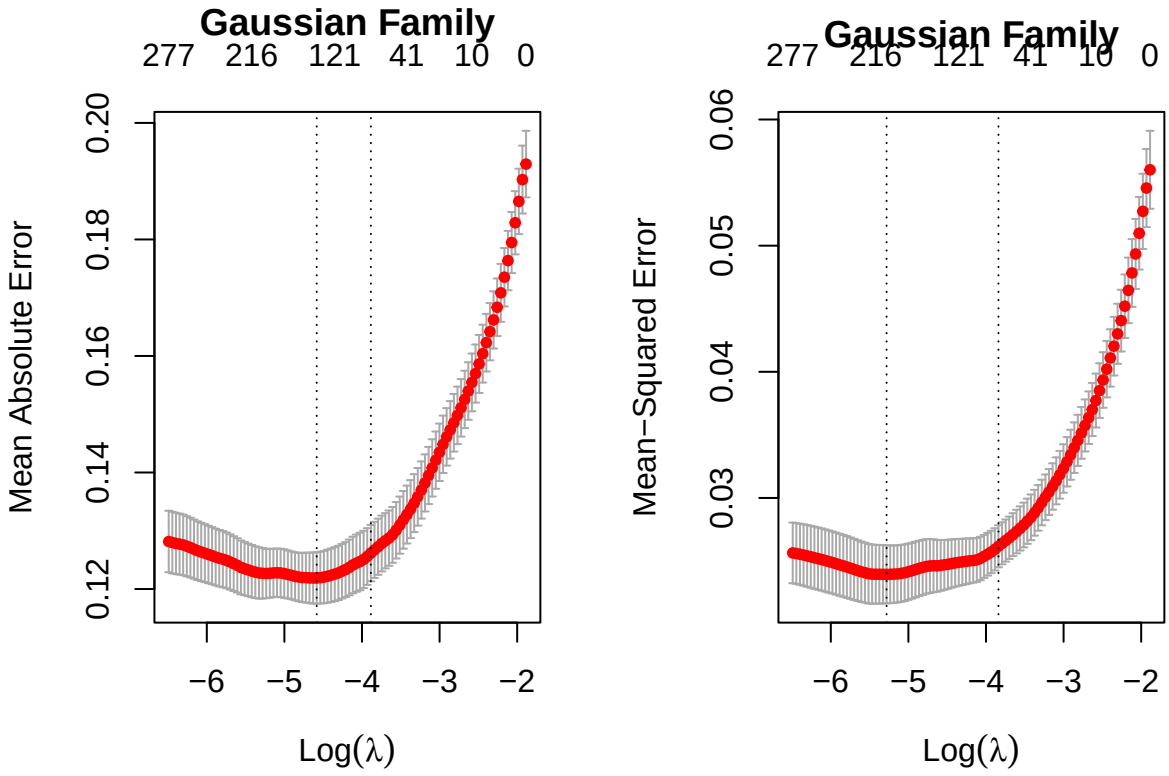
```
## [1]    76 26088
```

```
library(glmnet)

mod3<-glmnet(y=train_3[,3],x=as.matrix(train_3[,-c(1,2,3)]),family="gaussian")
plot(mod3,xvar="lambda")

# we may also organize the limits for y axis
mod3<-glmnet(y=train_3[,3],x=as.matrix(train_3[,-c(1,2,3)]),family="gaussian",lower.limits = -0.3, uppe
plot(mod3,xvar="lambda")
```



Same for model 3, we may observe that most of the variables follow typical regularization paths and shrink to zero eventually. We may obtain the plot of the cross-validation curve (red dotted line) along with upper and lower standard deviation curves along the lambda sequence (error bars) as shown below. The number of **SNPs with coefficients nonzero vary from 79 up to 142** from the visualization but also observe the cross-validation results below for an accurate result in numbers.

```
par(mfrow=c(1,2))
set.seed(1011)
cv_mod3 = cv.glmnet(x=as.matrix(train_3[,-c(1,2,3)]),y=train_3[,3], type.measure = "mae")
plot(cv_mod3)
title("Gaussian Family", line = 2)

mod3.1 = cv.glmnet(x=as.matrix(train_3[,-c(1,2,3)]),y=train_3[,3], type.measure = "mse")
plot(mod3.1)
title("Gaussian Family", line = 1.7)
```



The table output of the model gives information about: nonzero coefficients (Df), the percent (of null) deviance explained (%dev) and the value of (Lambda)

```
mod_3_fit<-data.frame(DF=cv_mod3$glmnet.fit$df, Dev_perc=cv_mod3$glmnet.fit$dev.ratio, Lambda=cv_mod3$gl
head(mod_3_fit)
```

```
##   DF   Dev_perc    Lambda
## 1  0 0.00000000 0.1517616
## 2  1 0.03662021 0.1448638
```

```
## 3  1 0.06998719 0.1382795
## 4  3 0.10334627 0.1319945
## 5  4 0.13484759 0.1259951
## 6  4 0.16370019 0.1202684
```

```
# cv_mod3$glmnet.fit #for all info just use the direct call
```

lambda.min is the value of lambda that gives minimum mean cross-validated error. We can use the following code to get the value of *lambda.min* and the model coefficients at that value of lambda:

```
cv_mod3$lambda.min
```

```
## [1] 0.01021985
```

```
log(cv_mod3$lambda.min)# observe the first bar to the left this is  the value of log() for the minimum
```

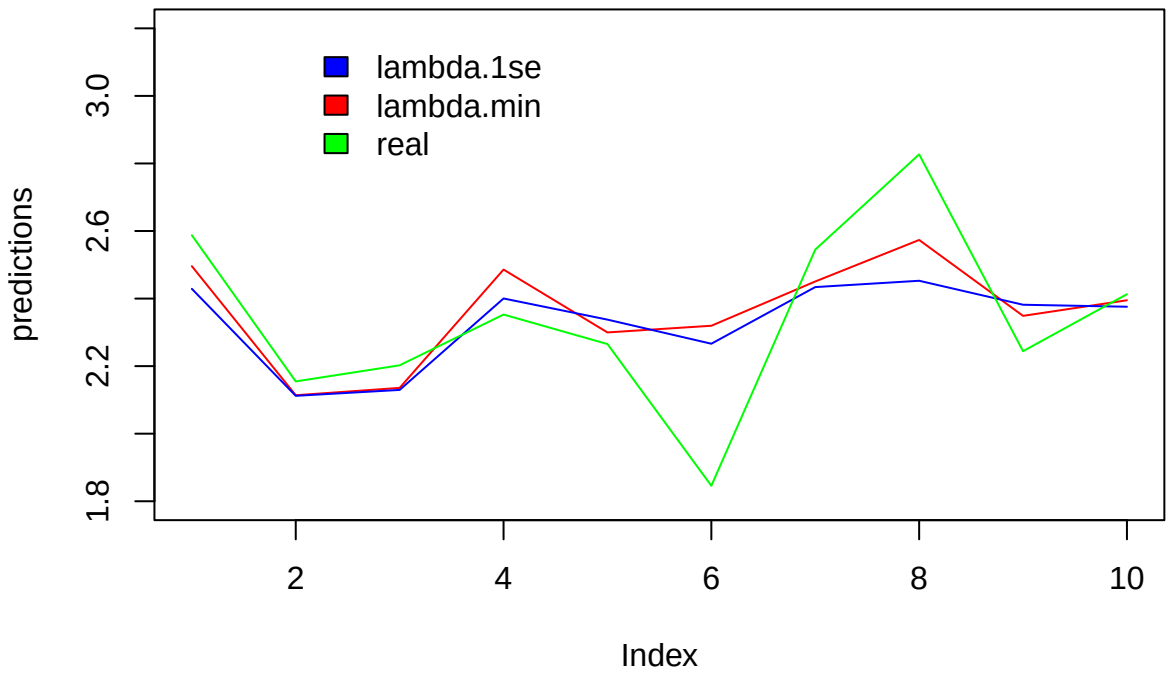
```
## [1] -4.583423
```

While *lambda.1se* is the value of lambda that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.

```
Pred_3<-predict(cv_mod3, newx = as.matrix(test_3[1:10,-c(1,2,3)]), type="response",s =c(cv_mod2$lambda.
Pred_3
```

```
##          s1       s2
## 1  2.495919 2.428831
## 3  2.113765 2.112049
## 5  2.135575 2.129547
## 8  2.485895 2.400290
## 9  2.299763 2.337674
## 12 2.319555 2.266306
## 20 2.450606 2.433994
## 23 2.573517 2.452727
## 25 2.348786 2.381728
## 36 2.394970 2.375812
```

```
# To compare the predictions graphically we may also plot them
plot(Pred_3[,1],col="red",type="l",ylab="predictions",ylim=c(1.8,3.2))
lines(Pred_3[,2],col="blue")
lines(test_3[1:10,3],col="green")
legend(2,3.2,c("lambda.1se","lambda.min","real"),fill=c("blue","red","green"),box.col="white")
```



A summary may be also obtained if we decide the number of cross-validation mean squared error criterion (mse) or mean absolute error (mae).

```
cv.glmnet(x=as.matrix(train_3[,-c(1,2,3)]),y=train_3[,3], type.measure = "mse",nfolds = 20)
```

```
##
## Call:  cv.glmnet(x = as.matrix(train_3[, -c(1, 2, 3)]), y = train_3[,      3], type.measure = "mse",
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.01022    59 0.02314 0.002205     142
## 1se 0.01960    45 0.02529 0.002171      79
```

Model 3-Seed.Width performance:

```
# Training Accuracy
Train_pred<-predict(cv_mod3, newx = as.matrix(train_3[1:301,-c(1,2,3)]), type="response",s =cv_mod3$lam

RMSE(train_3[,3],Train_pred)
```

```
## [1] 0.08125303
```

```
R_square(train_3[,3],Train_pred)
```

```
##             s1
## [1,] 0.9034213
```

```
# Testing Accuracy
Pred_3_all<-predict(cv_mod3, newx = as.matrix(test_3[1:76,-c(1,2,3)]), type="response",s =cv_mod3$lambda
RMSE(test_3[,3],Pred_3_all)
```

```
## [1] 0.1468543
```

```
R_square(test_3[,3],Pred_2_all)
```

```
##             s1
## [1,] 0.5030563
```

```
# A list of outputs for training and tetsing
list(RMSE=c(Train=RMSE(train_3[,3],Train_pred),Test=RMSE(test_3[,3],Pred_3_all)),R_square=c(Train=R_squa
```

```
## $RMSE
##      Train       Test
## 0.08125303 0.14685427
##
## $R_square
##     Train      Test
## 0.9034213 0.6673507
```

Provide a summary of the number of SNPs with non-zero coefficients. **78 out the 34922 SNPs had non-zero coefficients** and magnitude and sign of the coefficient will provide a measure of its effect and direction to Seed.volume.

```
length(which(as.matrix(coef(cv_mod3))!=0))-1 ##Subtracting 1 for intercept
```

```
## [1] 78
```

Final Comments question 5: What it is observed (which is already known) is the fact that for training set the errors are smaller than for testing set. Also the R-squared is higher for training and less for testing. We also observe differences in the number of SNPs with nonzero coefficients if MAE or MSE is used. But, numbers are in most cases approximate.

As a result of the above analysis: **For seed.length:1 out the 5509 SNPs had non-zero coefficients For Seed.width: 43 out the 26088 SNPs had non-zero coefficients For Seed.volume: 78 out the 34922 SNPs had non-zero coefficients**

Question 6

You may use the complete data for this part. Conduct M-SPLS analysis using all three covariates together (multivariate response) and create bootstrap confidence intervals to identify SNPs that are: • associated with all three traits. • associated with each trait individually. What proportions of SNPs identified here are in agreement with your analysis in Step 5

Solution 6

The high dimenssion of the dataset SNPs the **cv.spls()** function will not work (Output:Error Some of the columns of the predictor matrix have zero variance.)

SPLS for all three traits : Seed.length, Seed.width and Seed.volume

First let's create a dataset with the SNPs which **intersect** in all three sets for Seed.length, Seed.width and Seed.volume.

```
intersect_1<-intersect(colnames(data.frame(set_3,set_2)),colnames(set_1))
length(intersect_1)
```

```
## [1] 4338
```

```
Genotype_all<-Genotype_4[,intersect_1]# call only those SNPs extracted above
dim(Genotype_all)
```

```
## [1]  377 4338
```

```
# [1] 4338 SNPS
reduced_data<-data.frame(Pheno_data4, Genotype_all)
reduced_data[1:5,1:5]
```

```
##   Seed.length Seed.width Seed.volume SNP50 SNP51
## 1    8.064117   3.685183    2.587448     0     0
## 2    7.705383   2.951458    2.111145     0     2
## 3    8.237575   2.926367    2.154759     0     2
## 4    9.709375   2.382100    1.940221     2     2
## 5    7.118183   3.281892    2.202360     0     2
```

```
dim(reduced_data)
```

```
## [1]  377 4341
```

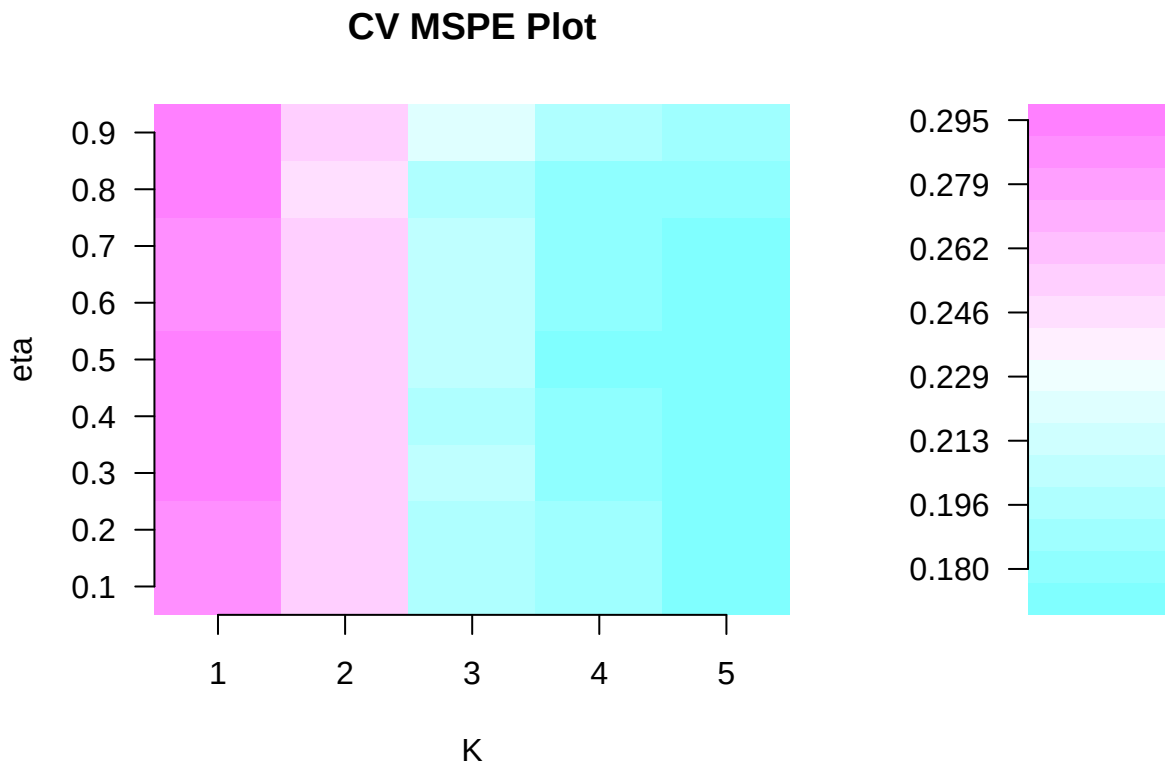
From the results below: SPLS chose **2552 variables among 4338 variables** as important for all three traits.

```
library(spls)
```

```
## Sparse Partial Least Squares (SPLS) Regression and
## Classification (version 2.2-3)
```

```r
set.seed(123)

x.seed<-reduced_data[,-c(1,2,3)]# all SNPs associated with all three traits
y.seed<-reduced_data[,c(1,2,3)]# three traits
cv<-cv.spls( x.seed, y.seed, eta = seq(0.1,0.9,0.1), K = c(1:5))
```

```
## eta = 0.1
## eta = 0.2
## eta = 0.3
## eta = 0.4
## eta = 0.5
## eta = 0.6
## eta = 0.7
## eta = 0.8
## eta = 0.9
##
## Optimal parameters: eta = 0.5, K = 5
```




```r
# Optimal parameters: eta = 0.5, K = 5
cv$eta.opt # optimal value of eta among 0.1 to 0.9 by step 0.1
```

```
## [1] 0.5
```

```
cv$K.opt # optimal K
```

```
## [1] 5
```

```
model_spls<-spls(x.seed,y.seed, eta = cv$eta.opt, K = cv$K.opt)
coef_spls<-coef(model_spls)
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP51"  "SNP53"  "SNP54"  "SNP205" "SNP482" "SNP560"
```

```
coef_spls<-coef(model_spls)
head(coef_spls[which(coef_spls!=0)])
```

```
## [1] -0.001916044 -0.005752665 -0.003370001 -0.001549683 -0.006902733
## [6] -0.006180468
```

```
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP51"  "SNP53"  "SNP54"  "SNP205" "SNP482" "SNP560"
```

```
#model_spls
# Sparse Partial Least Squares for multivariate responses
# ----
# Parameters: eta = 0.5, K = 5, kappa = 0.5
# PLS algorithm:
# pls2 for variable selection, simpls for model fitting
#
# SPLS chose 2552 variables among 4338 variables
#
# Selected variables:
# SNP51 SNP53   SNP54   SNP205  SNP482
# SNP560    SNP720  SNP723  SNP725  SNP782
# SNP828    SNP830  SNP832  SNP844  SNP1549
# SNP1885   SNP1943 SNP1977 SNP1979 SNP2021
```

```
ci.f<-ci.spls(model_spls)# create the bootstrap confidence intervals
```

```
## 10 % completed...
## 20 % completed...
## 30 % completed...
## 40 % completed...
## 50 % completed...
## 60 % completed...
## 70 % completed...
## 80 % completed...
## 90 % completed...
## 100 % completed...
```

```
head(ci.f$cibeta$Seed.length)# bootstrap intervals for Seed.length
```

```
##                2.5%         97.5%
## SNP51  -0.003864646  0.0003196013
## SNP53  -0.007501749 -0.0020062589
## SNP54  -0.005127798 -0.0009346581
## SNP205 -0.003813556  0.0010607555
## SNP482 -0.009960900 -0.0018083485
## SNP560 -0.008350204 -0.0019402224
```

```
head(ci.f$cibeta$Seed.width) #bootstrap intervals for Seed.width
```

```
##                 2.5%        97.5%
## SNP51  -0.0007437650 0.0003756098
## SNP53   0.0000661966 0.0013435045
## SNP54  -0.0003230052 0.0007682253
## SNP205 -0.0004630963 0.0007870435
## SNP482  0.0008262767 0.0025633756
## SNP560 -0.0001521762 0.0013025534
```

```
head(ci.f$cibeta$Seed.volume) #bootstrap intervals for Seed.volume
```

```
##                 2.5%        97.5%
## SNP51  -0.0007664483 3.128688e-05
## SNP53  -0.0005791743 3.177670e-04
## SNP54  -0.0006701223 1.021928e-04
## SNP205 -0.0004188768 2.965197e-04
## SNP482 -0.0002830578 8.980880e-04
## SNP560 -0.0007393003 1.262906e-04
```

```
CI_corrected<-correct.spls(ci.f)# visualization of original and corrected coefficients to zero

head(CI_corrected[model_spls$A,])# table of corrected coefficients for SNPs and traits
```

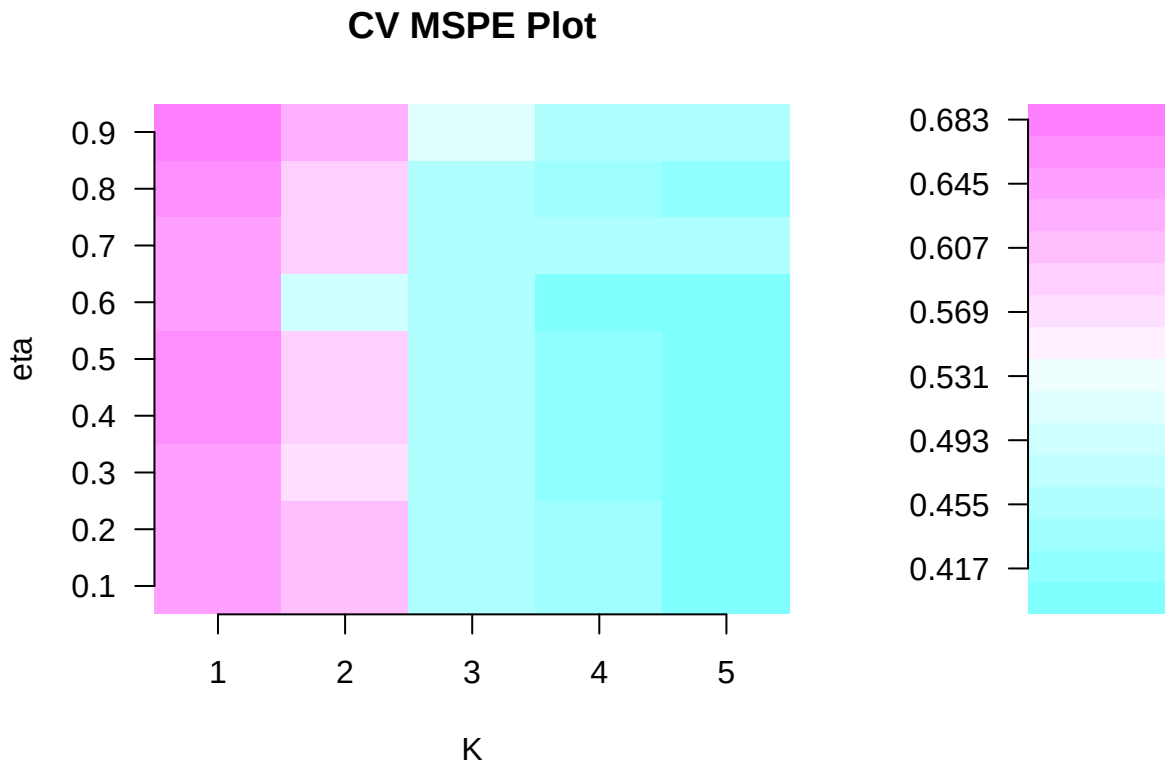
```
##         Seed.length  Seed.width Seed.volume
## SNP51   0.000000000 0.000000000           0
## SNP53  -0.005752665 0.000907451           0
## SNP54  -0.003370001 0.000000000           0
## SNP205  0.000000000 0.000000000           0
## SNP482 -0.006902733 0.001987834           0
## SNP560 -0.006180468 0.000000000           0
```

SPLS for Seed.length

From the results below: SPLS chose **3380 variables among 5509 variables** as important for Seed.length

```
set.seed(123)
x.seed<-set_1 # seed.length considered
y.seed<-Pheno_data4# all three traits considered
cv<-cv.spls( x.seed, y.seed[,1], eta = seq(0.1,0.9,0.1), K = c(1:5))
```

```
## eta = 0.1
## eta = 0.2
## eta = 0.3
## eta = 0.4
## eta = 0.5
## eta = 0.6
## eta = 0.7
## eta = 0.8
## eta = 0.9
##
## Optimal parameters: eta = 0.5, K = 5
```




```
# Optimal parameters: eta = 0.5, K = 5
cv$eta.opt # optimal value of eta among 0.1 to 0.9 by step 0.1
```

```
## [1] 0.5
```

```
cv$K.opt # optimal K
```

```
## [1] 5
```

```
model_spls<-spls(x.seed,y.seed[,1], eta = cv$eta.opt, K = cv$K.opt)

coef_spls<-coef(model_spls)
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP50"  "SNP53"  "SNP55"  "SNP175" "SNP198" "SNP200"
```

```
coef_spls<-coef(model_spls)
head(coef_spls[which(coef_spls!=0)])
```

```
## [1]  0.0034432048 -0.0042593340  0.0043825969  0.0026003517  0.0006457178
## [6]  0.0021729088
```

```
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP50"  "SNP53"  "SNP55"  "SNP175" "SNP198" "SNP200"
```

```
# model_spls # run this to obtain the follow output

#
# Sparse Partial Least Squares for an univariate response
# ----
# Parameters: eta = 0.5, K = 5
# PLS algorithm:
# pls2 for variable selection, simpls for model fitting
#
# SPLS chose 3380 variables among 5509 variables
#
# Selected variables:
# SNP50 SNP53   SNP55   SNP175  SNP198
# SNP200    SNP225  SNP227  SNP228  SNP234
# SNP235    SNP240  SNP306  SNP313  SNP330
# SNP344    SNP371  SNP379  SNP388  SNP392
# SNP393    SNP397  SNP402  SNP404  SNP420
# SNP431    SNP433  SNP437  SNP439  SNP449
# SNP452    SNP454  SNP482  SNP488  SNP523
```

```
ci.f<-ci.spls(model_spls)
```

```
## 10 % completed...
## 20 % completed...
## 30 % completed...
## 40 % completed...
## 50 % completed...
## 60 % completed...
## 70 % completed...
## 80 % completed...
## 90 % completed...
## 100 % completed...
```

```
head(ci.f$cibeta$Seed.length)# bootstrap intervals for Seed.length
```

```
## NULL
```

```
CI_corrected<-correct.spls(ci.f)# visualization original and corrected coefficients to zero

head(CI_corrected[model_spls$A,])# table of corrected coefficients for SNPs and seed.length
```

```
##        SNP50        SNP53        SNP55       SNP175       SNP198       SNP200
##  0.003443205 -0.004259334  0.004382597  0.002600352  0.000000000  0.002172909
```

SPLS for Seed.width

From the results below: SPLS chose **1651 variables among 2361 variables** as important for Seed.width First we have taken into consideration those SNPs which adj_pvalue<1e-20 (since for the whole set from Step 3 the code has given error output). From the dataset the reduced set is compound of 2364 SNPs.

```
set.seed(123)
P_adj22<-data.frame(res2[,c(1,9,10,11,12)],p_adj2<=1e-20)
sum(P_adj22$p_adj2....1e.20==TRUE)
```

```
## [1] 2364
```

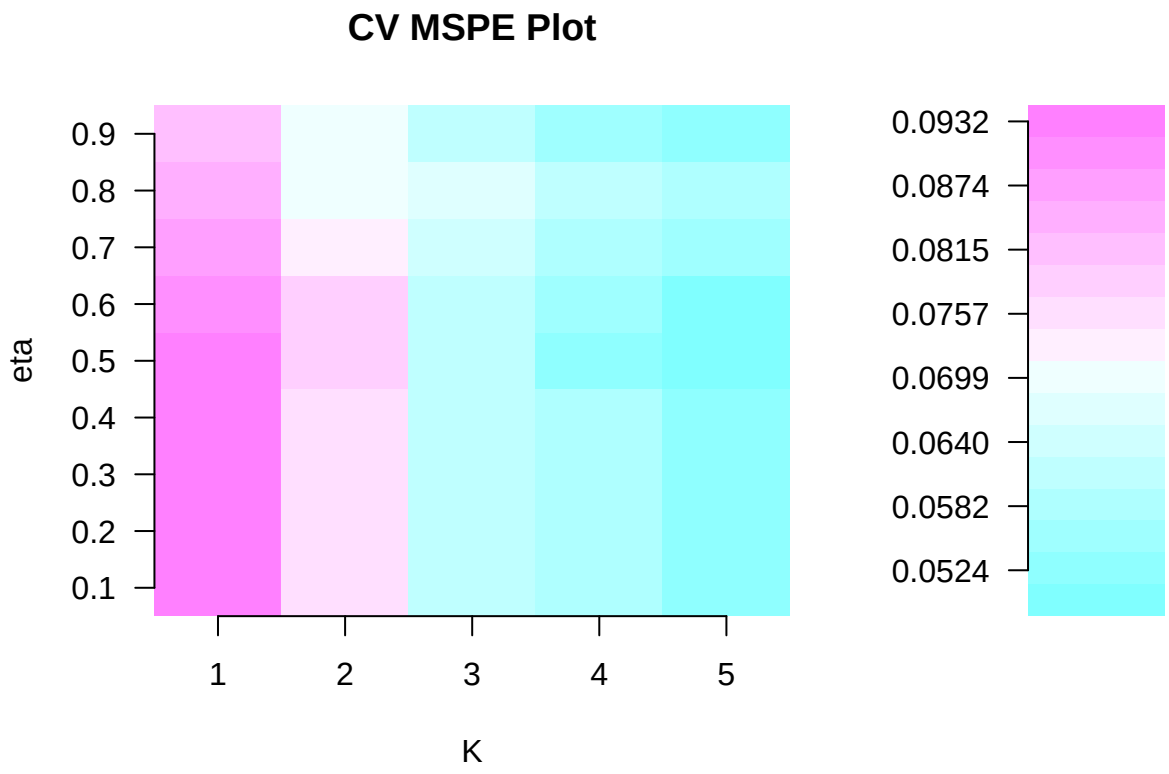
```
i_22<-which(P_adj22$p_adj2....1e.20==TRUE)# saves the rows of SNPs which are associated with Seed.width
set_22<-new_data[,i_22]
dim(set_22)
```

```
## [1]  377 2364
```

```
# [1] 2364 SNPs
```

```
x.seed<-set_22[,-c(1,2,3)] # seed.width considered 2364 SNPs
y.seed<-Pheno_data4# all three traits considered
cv<-cv.spls( x.seed, y.seed[,2], eta = seq(0.1,0.9,0.1), K = c(1:5))
```

```
## eta = 0.1
## eta = 0.2
## eta = 0.3
## eta = 0.4
## eta = 0.5
## eta = 0.6
## eta = 0.7
## eta = 0.8
## eta = 0.9
##
## Optimal parameters: eta = 0.5, K = 5
```




```
cv$eta.opt # optimal value of eta among 0.1 to 0.9 by step 0.1
```

```
## [1] 0.5
```

```
cv$K.opt # optimal K
```

```
## [1] 5
```

```
model_spls<-spls(x.seed,y.seed[,2], eta = cv$eta.opt, K = cv$K.opt)
coef_spls<-coef(model_spls)
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP582"  "SNP585"  "SNP592"  "SNP639"  "SNP954"  "SNP1031"
```

```
coef_spls<-coef(model_spls)
head(coef_spls[which(coef_spls!=0)])
```

```
## [1]  1.577442e-03 -6.103958e-04  3.139635e-04  1.304135e-03  9.417653e-06
## [6] -4.090682e-04
```

```
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP582"  "SNP585"  "SNP592"  "SNP639"  "SNP954"  "SNP1031"
```

```
# model_spls
# Sparse Partial Least Squares for an univariate response
# ----
# Parameters: eta = 0.5, K = 5
# PLS algorithm:
# pls2 for variable selection, simpls for model fitting
#
# SPLS chose 1651 variables among 2361 variables
#
# Selected variables:
# SNP582    SNP585  SNP592  SNP639  SNP954
# SNP1031   SNP1072 SNP1093 SNP1095 SNP1098
# SNP1106   SNP1107 SNP1136 SNP1160 SNP1167
# SNP1934   SNP1969 SNP2061 SNP2082 SNP2085
```

```
ci.f<-ci.spls(model_spls)
```

```
## 10 % completed...
## 20 % completed...
## 30 % completed...
## 40 % completed...
## 50 % completed...
## 60 % completed...
## 70 % completed...
## 80 % completed...
## 90 % completed...
## 100 % completed...
```

```
#head(ci.f$cibeta)# bootstrap intervals for Seed.width
```

```
CI_corrected<-correct.spls(ci.f)# visualization original and corrected coefficients to zero

head(CI_corrected[model_spls$A,])# table of corrected coefficients for SNPs and seed.width
```

```
##  SNP582  SNP585  SNP592  SNP639  SNP954 SNP1031
##       0       0       0       0       0       0
```

SPLS for Seed.volume

From the results below: SPLS chose **3001 variables among 3865 variables** as important for Seed.width Same for seed.volume we created a reduced dataset taking into consideration those SNPs which adj_pvalue was less than 1e-20.

```
P_adj33<-data.frame(res3[,c(1,9,10,11,12)],p_adj3<=1e-20)
sum(P_adj33$p_adj3....1e.20==TRUE)
```

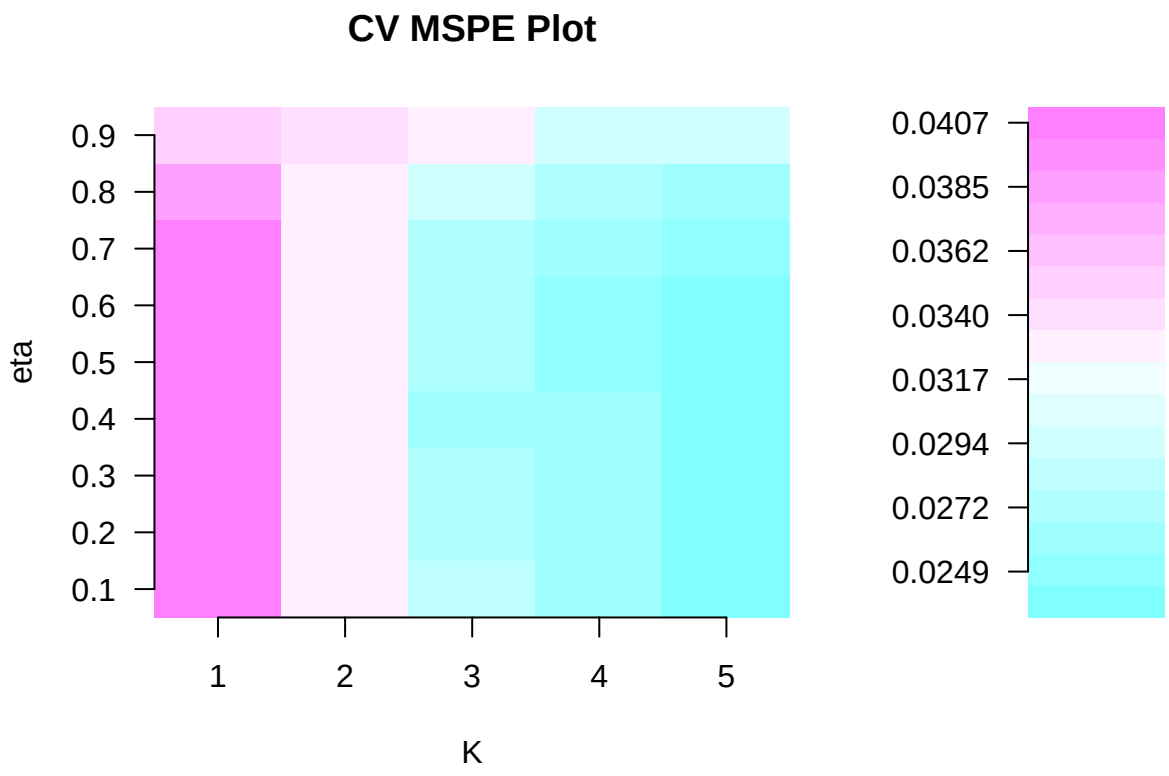
```
## [1] 3868
```

```
i_33<-which(P_adj33$p_adj3....1e.20==TRUE)# saves the rows of SNPs which are associated with Seed.volum
set_33<-new_data[,i_33]
dim(set_33)
```

```
## [1]  377 3868
```

```
set.seed(123)
x.seed<-set_33[,-c(1,2,3)] # seed.width considered 2364 SNPs
y.seed<-Pheno_data4# all three traits considered
cv<-cv.spls( x.seed, y.seed[,3], eta = seq(0.1,0.9,0.1), K = c(1:5))
```

```
## eta = 0.1
## eta = 0.2
## eta = 0.3
## eta = 0.4
## eta = 0.5
## eta = 0.6
## eta = 0.7
## eta = 0.8
## eta = 0.9
##
## Optimal parameters: eta = 0.5, K = 5
```



```
cv$eta.opt # optimal value of eta among 0.1 to 0.9 by step 0.1
```

```
## [1] 0.5
```

```
cv$K.opt # optimal K
```

```
## [1] 5
```

```
model_spls<-spls(x.seed,y.seed[,3], eta = cv$eta.opt, K = cv$K.opt)
coef_spls<-coef(model_spls)
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP618"  "SNP624"  "SNP715"  "SNP1701" "SNP1702" "SNP1705"
```

```
coef_spls<-coef(model_spls)
head(coef_spls[which(coef_spls!=0)])
```

```
## [1] 0.0006566456 0.0001875638 0.0003676546 0.0004543181 0.0009581650
## [6] 0.0002791937
```

```
head(rownames(coef_spls)[which(coef_spls!=0)])
```

```
## [1] "SNP618"  "SNP624"  "SNP715"  "SNP1701" "SNP1702" "SNP1705"
```

```
#model_spls
# Sparse Partial Least Squares for an univariate response
# ----
# Parameters: eta = 0.5, K = 5
# PLS algorithm:
# pls2 for variable selection, simpls for model fitting
#
# SPLS chose 3001 variables among 3865 variables
#
# Selected variables:
# SNP618    SNP624  SNP715  SNP1701 SNP1702
# SNP1705   SNP1718 SNP1719 SNP1720 SNP1721
# SNP1739   SNP1742 SNP1743 SNP1761 SNP1773
# SNP1777   SNP1781 SNP1785 SNP1786 SNP1789
```

```
ci.f<-ci.spls(model_spls)
```

```
## 10 % completed...
## 20 % completed...
## 30 % completed...
## 40 % completed...
## 50 % completed...
## 60 % completed...
## 70 % completed...
## 80 % completed...
## 90 % completed...
## 100 % completed...
```

```
#head(ci.f$cibeta)# bootstrap intervals for Seed.volume
#              2.5%         97.5%
# SNP1    -8.648783e-04  2.162893e-03
# SNP2    -8.504813e-04  2.255298e-03
# SNP3    -2.062148e-03  1.015296e-03
# SNP5    -6.396920e-04  2.405865e-03
# SNP7    -2.399910e-03  7.530565e-04
# SNP8    -1.142958e-03  2.054155e-03
```

```
CI_corrected<-correct.spls(ci.f)# visualization original and corrected coefficients to zero

head(CI_corrected[model_spls$A,])# table of corrected coefficients for SNPs and seed.volume
```

```
##  SNP618  SNP624  SNP715 SNP1701 SNP1702 SNP1705
##       0       0       0       0       0       0
```

Step 6 final comments :

For this step we have considered reduced sets from those obtained at STEP 3. The reason was that the cross validation process in R was not performing due to an error. The selection of the SNPs considered at each case was: **Case 1- all traits**: The reduced set was obtainet from the intersection of the SNPs which were associated with all thre traits. **Case 2- Seed.length**: the same set as the one obtained at step 3 was used. **Case 3 (seed.width) and Case 4 (Seed.volume)**: The reduced set of SNPs was obtained based on a threshold of adj_pvalue <= 1e-20.

Overall, compared with the results from Step 5 we got a higher number of SNPs associated with the traits but the ratio among the number of SNPs associated and those considered was around the same logic. Such as: for seed.length a lower number among considered was found in both STEP 5 and STEP 6. And for seed.width and seed.volume this ratio was higher in both steps (5 and 6).

The approach used in STEP 5 was more strict, removing a significant number of SNPs from the consideration.

##Appendix The Rmarkdown code for this project may be found here:

https://github.com/EGjika/Genome-wide-association-mapping-/blob/main/Genome%20wide%20association%20mapping%20in%20R.Rmd

Reference: http://mixomics.org/case-studies/spls-liver-toxicity-case-study/ https://www.bioconductor.org/packages/devel/bioc/vignettes/mixOmics/inst/doc/vignette.html#principle-of-pls