Lectures for statistical genetics in PNU-winter-workshop-2019

Kipoong Kim, Hansong Lee

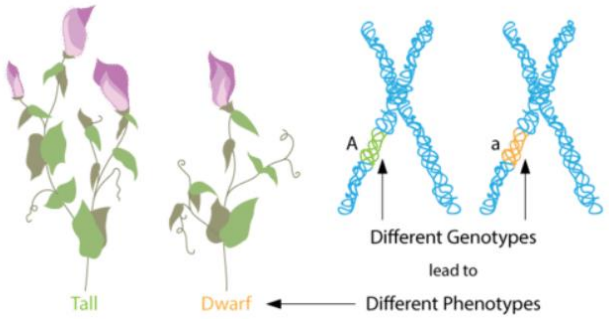
January 2019

Part 1. GAPIT

Contents

1. Introduction to Genome-wide Association Studies (GWAS)

- Observational study of a genome-wide set of genetic variants in different individuals to see if any variant is associated with a trait.
- Compare the DNA(ex.SNP) of samples having varying phenotypes for a particular trait or disease.
- If one type of the variant (one allele) is more frequent in sample, the variant is said to be associated with the disease. The associated SNPs are then considered to mark a region of the human genome that may influence the risk of disease.



Association Study

1) Population based association study

- A common design for association studies is population-based, where unrelated subjects are collected.
- when samples are of different ethnic ancestries, population-based association studies may produce spurious associations due to population stratification.

2) Family based association study

- An alternative design uses family-based studies, where family members are collected for association analyses.
- This methods protect against population stratification, but costly.

Univaraite Analysis vs Multivariate Analysis

1) Univariate

- One variable is analyzed at a time. Objective is to describe the variable.

$$\mathrm{Y} = b_1 X_1 + e$$

$$where \quad Y: phenotype,\ X_1: genotype\ of\ SNP1,\ b_1: effect\ of\ SNP1,\ e: error\ term$$

2) Multivariate

- More than two variables are analyzed together for any possible association or interactions.

$$\mathrm{Y} = b_1 X_1 + b_2 X_2 + b_3 X_3 + \cdots + e$$

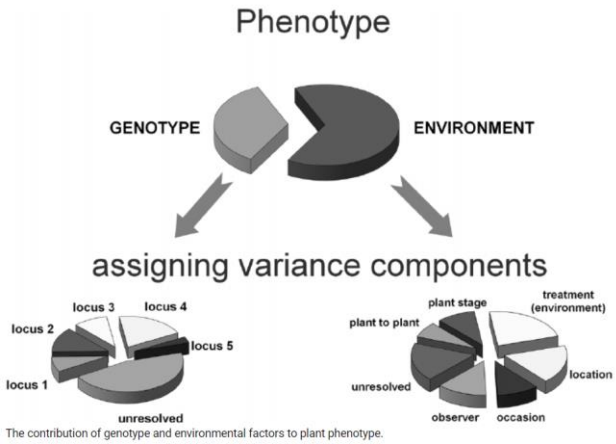
$$where \quad Y: phenotype,\ X_i: genotype\ of\ SNPi,\ b_i: effect\ of\ SNPi,\ e: error\ term$$

- Multivariate analysis can also improve the ability to detect susceptible genetic variants whose effects are too small to be detected in univariate analysis

GWAS with covariate

- As in main-effects GWAS that search for the effects of genotype alone, differences(termed population substructure) can be mistaken for true genetic effects, and is therefore a serious concern.
- The extent of the substructure problem can be inspected by using

- Inflation factor λ: Inflation factor λ > 1 indicates population structure and/or genotyping error
- QQ plot



The contribution of genotype and environmental factors to plant phenotype.

1 Hunter DJ (2005) Gene-environment interactions in human diseases. Nat Rev Genet 6: 287–298
2 Behavior of QQ-Plots and Genomic Control in Studies of Gene-Environment Interaction

2. Statistical Models of GAPIT

- **Concept of fixed & random effect**

fixed effect

A fixed effect is a factor whose level is selected by a procedure that is not random, or is made up of possible levels in the entire population.

Researchers are interested in comparing the effects of factors only for the levels of response variables included in the study.

"FIXED EFFECT MODEL"

random effect

Factor levels are often various and researchers or data analysts choose subset levels to include in the study.

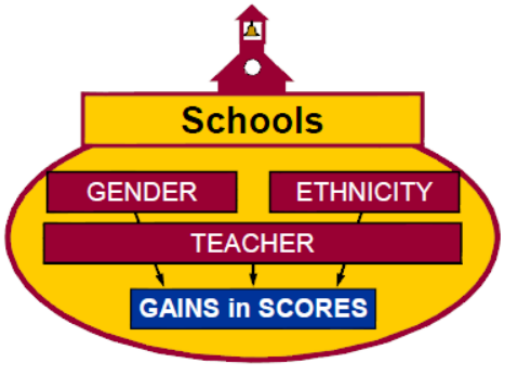
This effect is called "random effect".

"RANDOM EFFECT MODEL"

Concept of fixed & random effect

ex) In all schools in a school district, test scores, gender and ethnicities of 4-th grade students, and teacher identification were recorded. The main purpose of this data set was

1. to assess the school with a test score
2. to assess gender and ethnicity.



Schools, Gender, Ethnicity : fixed effect

Teacher : random effect

MLM(mixed linear model)

Among available statistical methods, the mixed linear model (MLM) has been the most flexible and powerful for controlling population structure and individual unequal relatedness (kinship), the two common causes of spurious associations.

$$\mathbf{Y} = \mathbf{X\beta} + \mathbf{Zu} + \mathbf{e}$$

- assumption

$Var\begin{pmatrix} u \\ e \end{pmatrix} = \begin{pmatrix} G & 0 \\ 0 & R \end{pmatrix}$ where G=$\sigma_a{}^2$K with $\sigma_a{}^2$ as additive genetic variance and K as the kinship matrix

- notation

Y: observed phenotype (Y: input)

β: fixed effects, including the genetic marker, population structure(Q), and the intercept

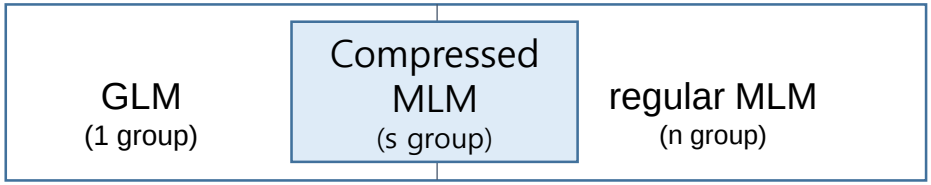
u: random additive genetic effect

X, Z: design matrices (X: input)

Population stratification (or population structure)

- the presence of a systematic difference in allele frequencies between subpopulations in a population, possibly due to different ancestry, especially in the context of association studies.
- population stratification can result in excess false positive or false negative.
- QQ plot, genetic inflation factor, PCoA plot can be used.

Models in GAPIT



• CMLM(compressed MLM)

Regular MLM is an extreme case of CMLM where each individual is considered as a group.

CMLM methods reduces the size of the random genetic effect in the absence of pedigree information and eliminate iterations.

• ECMLM(enriched CMLM)

The ECMLM calculates kinship using several different algorithms and then chooses the best combination between kinship algorithms and grouping algorithms.

The model fit was reflected by twice negative log likelihood (−2LL).

3. Quality Control(QC) before Analysis

• **call rate**

Markers with low call rate(poor quality DNA sample) should be eliminated. Classically, markers with a call rate less than 95% are removed from further study.

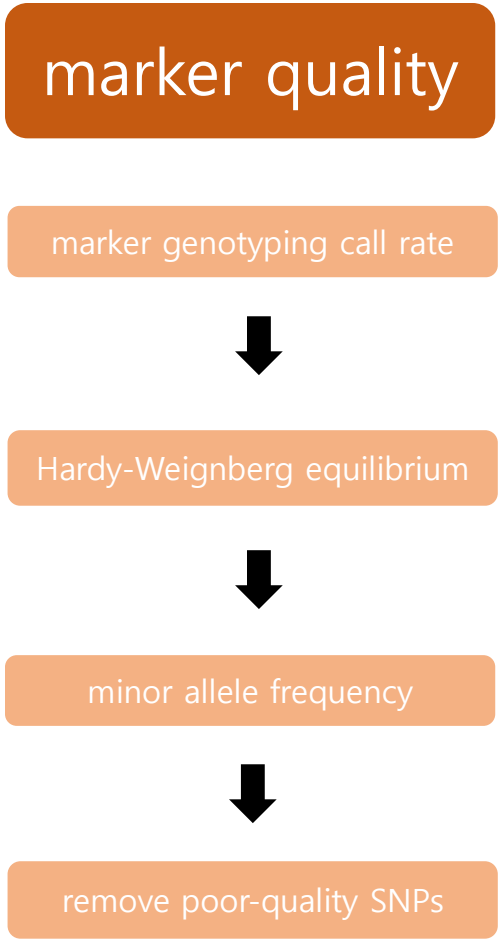
• **Hardy-Weinberg equilibrium, HWE**

- Departure from this equilibrium can be indicative of potential genotyping errors, population stratification, or even actual association(evolutionary selection) to the trait under study.
- In case-control study, just use control data for HWE test.

Quality Control(QC) before Analysis

- **Minor allele frequency, MAF**

It is important to filter SNPs base on MAF, because statistical power is extremely low for rare SNPs. However, many causal genetic variants are rare. A recommended practice is to not remove them, but interpret them with caution.



4. Analysis using GAPIT

INPUT

- Genotypic Data: HapMap format or in numeric format

1) HapMap

- SNP information in row and SNP information in first 11 columns.

GAPIT uses only 3 of these: (8 columns can be “NA”s)

“rs” column: the SNP name;

“chrom” column: which is the SNP’s chromosome;

“pos” column: SNP’s base pair (bp) position

- Missing genotypic data are indicated by either “NN” (double bit) or “N” (single bit).

- The first row contains the header labels (head=F).

PRACTICE_read data

```
library(multtest); library(gplots); library(LDheatmap); library(genetics)
library(ape); library(EMMREML); library(compiler)
library(scatterplot3d)
source("http://zzlab.net/GAPIT/gapit_functions.txt")
source("http://zzlab.net/GAPIT/emma.txt")
setwd("C:/Users/pc/Desktop/gapitex")

library(data.table)
raw_myY <- read.table(file="filename.csv", sep=",", header=T)
raw_myX <- as.data.frame(fread(file="filename.csv", sep=",", header=F))
# load("data.RData")
head(raw_myY); raw_myX[1:5,1:20]
raw_myY <- raw_myY[,1:2]
dim(raw_myY); dim(raw_myX)
```

INPUT

2) Numeric

- Columns are used for SNPs and rows are used for taxa.
- Two separate files must be provided to GAPIT.

One(called "GD" file): numeric genotypic data

the other(called "GM" file): contains the position of each SNP along the genome with same order of "GD" file's SNPs.

<GD file>

taxa	PZB00859.1	PZA01271.1	PZA03613.2	PZA03613.1
33-16	2	0	0	2
38-11	2	2	0	2
4226	2	0	0	2
4722	2	2	0	2
A188	0	0	0	2

<GM file>

Name	Chromosome	Position
PZB00859.1	1	157104
PZA01271.1	1	1947984
PZA03613.2	1	2914066
PZA03613.1	1	2914171
PZA03614.2	1	2915078

INPUT

- **Phenotypic Data**
  - Taxa names should be in the first column.
  - The remaining columns should contain the observed phenotype from each individual.
  - Missing phenotypic data should be indicated by either “NaN” or “NA”.
- **Kinship**
  - n by n+1 matrix
  - It is an option to put Kinship matrix, because GAPIT calculate kinship matrix automatically.

33-16	2	0.228837	0.229322	0.268842	0.237145	0.0781	0.347107
38-11	0.228837	2	0.244965	0.293708	0.175211	0.079276	0.295606
4226	0.229322	0.244965	2	0.214859	0.236153	0.082693	0.283713
4722	0.268842	0.293708	0.214859	2	0.25935	0.061573	0.160104
A188	0.237145	0.175211	0.236153	0.25935	2	0.061469	0.232799
A214N	0.0781	0.079276	0.082693	0.061573	0.061469	2	0.110364
A239	0.347107	0.295606	0.283713	0.160104	0.232799	0.110364	2

kinship matrix

Using kinship information, GAPIT makes group and

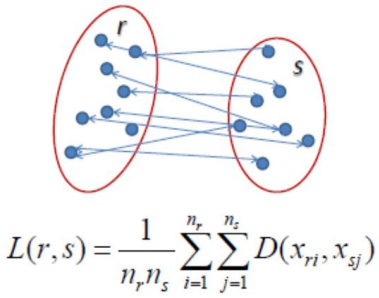
Depending on the options you specify, the objects that belong to each group are different.

kinship.algorithm : Algorithm to Derive Kinship from Genotype

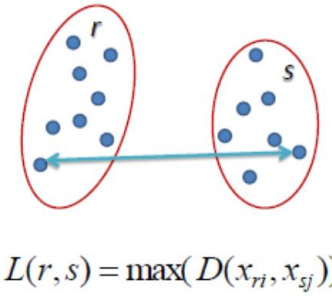
1. VanRaden(default) : more similar minor allele, higher kinship value
2. Separation : make kinship matrix using princal component of var-covariance
3. Zhang : farther form average allele, higher kinship value
4. EMMA : make kinship matrix using heterozygosity

kinship matrix

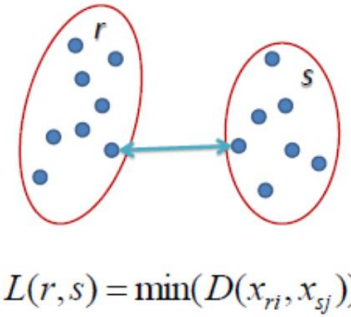
kinship.cluster – average(default), complete, single, ward.D, mcquitty, median, and centroid
: Clustering algorithm to group individuals based on their kinship



<average>



<complete>



<single>

kinship.group – Mean(default), Max, Min, Median
: Method to derive kinship among groups

INPUT

- **Covariate variables**

A file containing covariates (called “CV” in GAPIT) can include information such as population structure, which are fitted into the GWAS and GS models as fixed effects.

Taxa	Q1	Q2	Q3
33-16	0.014	0.972	0.014
38-11	0.003	0.993	0.004
4226	0.071	0.917	0.012
4722	0.035	0.854	0.111
A188	0.013	0.982	0.005
A214N	0.762	0.017	0.221
A239	0.035	0.963	0.002
A272	0.019	0.122	0.859
A441-5	0.005	0.531	0.464

DATA

	Taxa	Aspartic
1	WC2-001	4636.43
2	WC2-002	4341.99
3	WC2-003	5041.24
4	WC2-004	4991.21
5	WC2-005	5084.88

	V1	V2	V3	V4	V5	V6	...	V12	V13	V14	V15
1	rs	alleles	chrom	pos	strand	assembly		WC2-046	WC2-110	WC2-358	WC2-340
2	AX-90327988	<NA>	1	31321	<NA>	<NA>		TT	TT	TT	TT
3	AX-90510676	<NA>	1	34212	<NA>	<NA>		CC	TT	CC	CC
4	AX-90347236	<NA>	1	46222	<NA>	<NA>		GG	GG	GG	GG
5	AX-90488775	<NA>	1	56126	<NA>	<NA>		CC	CC	TT	CC
								GG	GG	GG	GG

- 439 samples, 169028 SNPs.
- 11 columns in X data are SNP information which is essential to use GAPIT.

PRACTICE_pre-process data

```
#remove NA Y
Yna <- is.na(raw_myY[,2])
myY <- raw_myY[!Yna,]
myX <- raw_myX[,c(rep(T,11),!Yna)]
dim(myY);dim(myX)

# check proprotion of NA
genedata <-  myX[-1,-c(1:11)]
na <- apply(genedata, 1, function(x) sum(x=='NN'))
length(na)
max(na); max(na)/ncol(genedata)
```

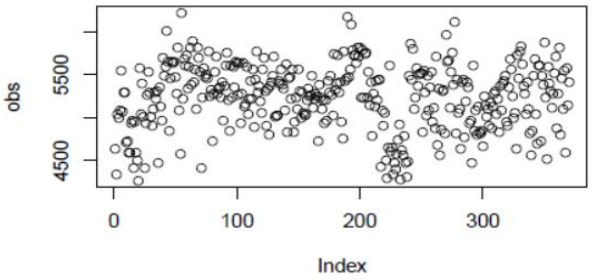
PRACTICE_pre-process data

```
# check one genotype per SNP
one <- apply(genedata, 1, function(x) length(unique(x[x!="NN"]))==1)
sum(one)
wone <- which(one) + 1
myX <- myX[-wone,]
dim(myX)  #161798    381
```

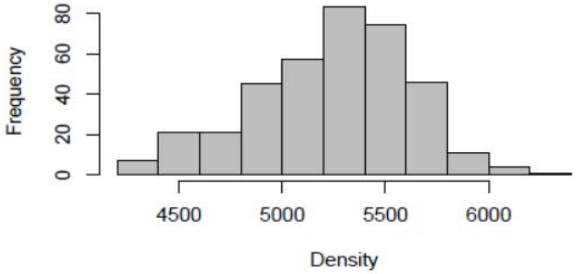
PRACTICE_ code 1: treating heterozygosity

```
setwd("C:/Users/pc/Desktop/gapitex/result1")
result1 <- GAPIT(Y=myY, G=myX)
summary(myY)
```

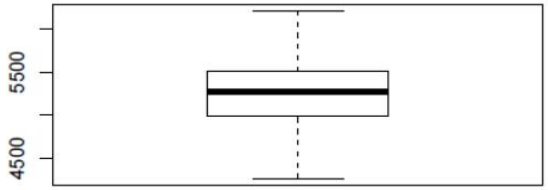
① Phenotype diagnosis



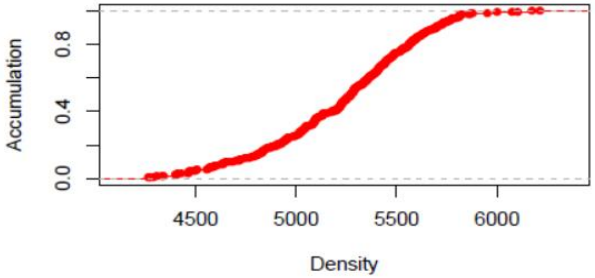
scatter plot



histogram



boxplot



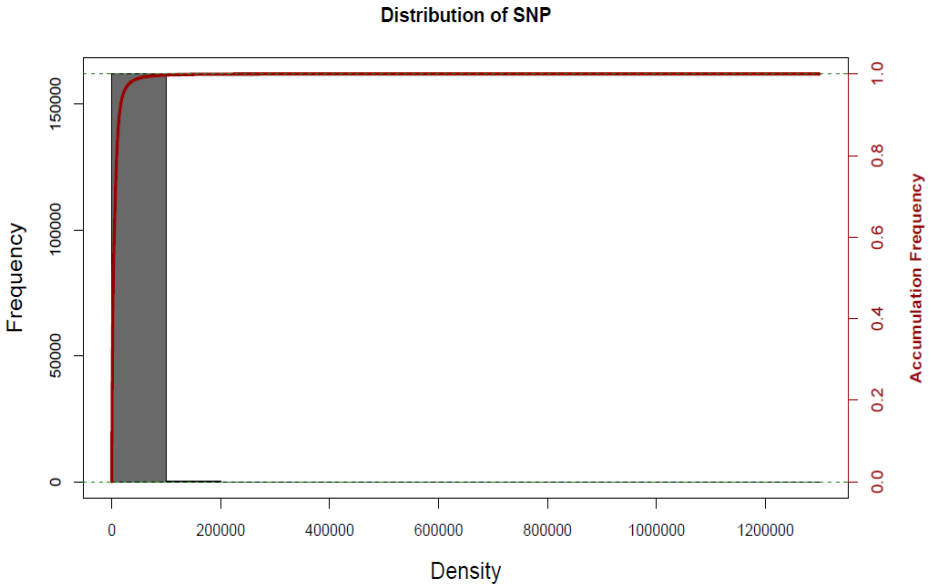
accumulative distribution

RESULT

② Marker density

Previous studies suggested selecting SNPs with large effects can produce almost as effective evaluation for GEBVs as HD panels.

$$\text{marker density} = \frac{number\ of\ markers}{length\ of\ the\ used\ parts\ of\ the\ genome}$$



RESULT

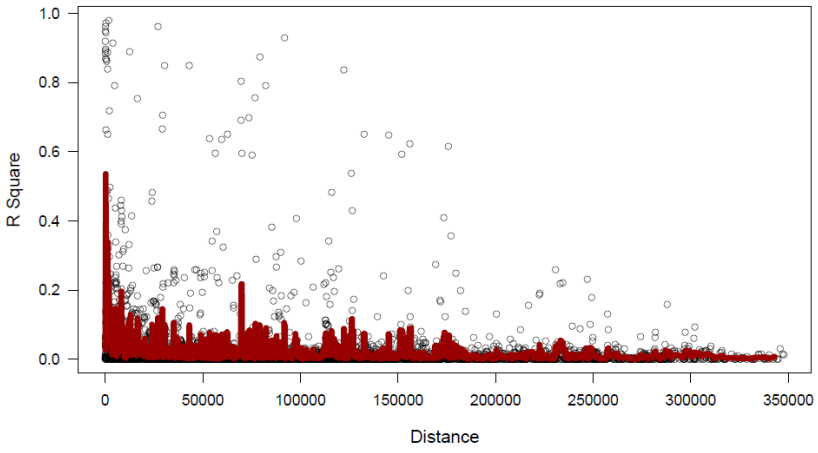
③ Linkage Disequilibrium Decade

LD is important in GWAS because it allows us to get good coverage of the entire genome by genotyping only a small fraction of markers.

Each dot : a pair of distances between two markers on and their squared correlation coefficient

red line : moving average of the 10 adjacent markers.

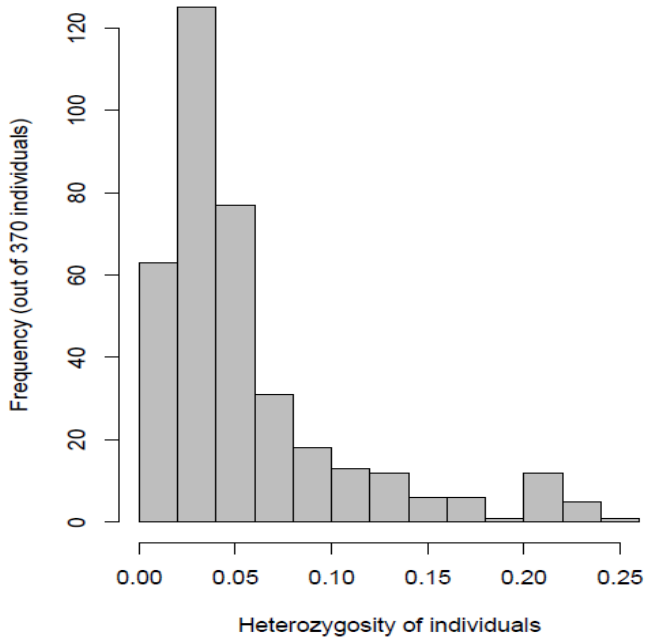
“PLINK”

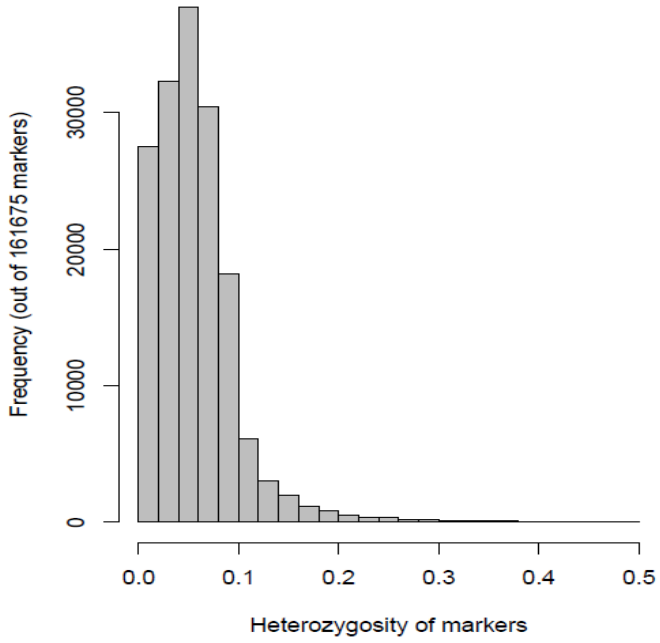


RESULT

④ Heterozygosis

High level of heterozygosis indicated low quality. For example, over 50% of heterozygosis on inbred lines for some of markers suggested they problematic.





PRACTICE_heterozygosity

```
myGD= apply(myX[-1,-(1:11)], 1,
          function(one) GAPIT.Numericalization(one, bit=2, impute="Middle",
           Major.allele.zero=F))
H=1-abs(myGD-1)
het.ind=apply(H,1,mean)
het.snp=apply(H,2,mean)
hist(het.ind,col="gray", main="", xlab="Heterozygosity of individuals")
hist(het.snp,col="gray", main="", xlab="Heterozygosity of markers")


sum(het.ind > 0.5)
sum(het.snp > 0.5)
w1 <- which(het.snp > 0.5) + 1
myX <- myX[-w1,]
dim(myX)  #161677     381
```

genotyping byte 1 or 2

imputation method

T: 0 means Major.allele
F: 0 means Minor.allele

PRACTICE_PCoA(principal coordinates analysis)

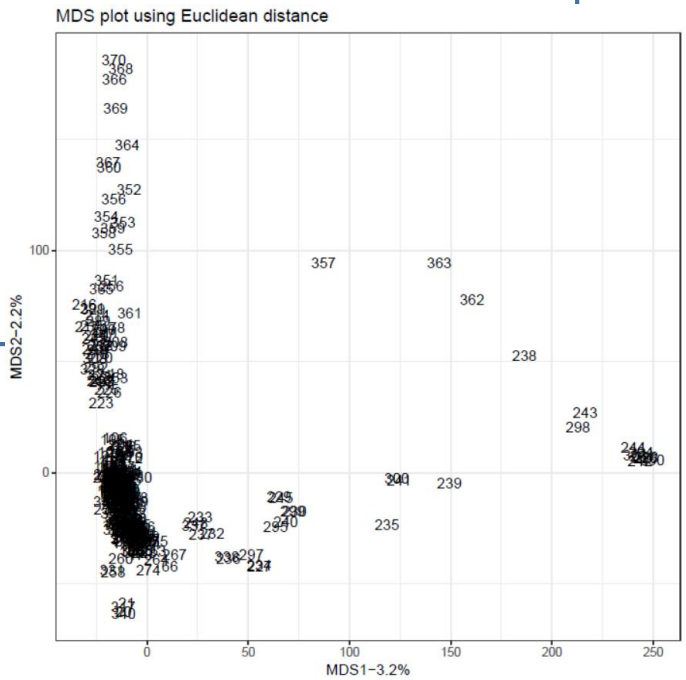
Principal Coordinates Analysis (PCoA, = Multidimensional scaling, MDS) is a method to explore and to visualize similarities or dissimilarities of data. It starts with a similarity matrix or dissimilarity matrix (= distance matrix) and assigns for each item a location in a low-dimensional space.

By using PCoA we can visualize individual and/or group differences. Individual differences can be used to show outliers.

```
library(ggplot2)
dim(myGD)
distance.matrix <- dist(myGD)
mds.stuff <- cmdscale(distance.matrix, eig=T, x.ret=T)
mds.var.per <- round(mds.stuff$eig/sum(mds.stuff$eig)*100,1)
mds.var.per[1:5]
mds.values <- mds.stuff$points
mds.data <- data.frame(Sample=1:nrow(myY),  X=mds.values[,1], Y=mds.values[,2])
```

PRACTICE_PCoA(principal coordinates analysis)

```
pdf("PCoA.pdf")
gg <- ggplot(data=mds.data, aes(x=X, y=Y, label=Sample)) +
  geom_text() +
  theme_bw() +
  xlab(paste("MDS1-", mds.var.per[1],"%", sep="")) +
  ylab(paste("MDS2-", mds.var.per[2],"%", sep="")) +
  ggtitle("MDS plot using Euclidean distance")
print(gg)
dev.off()
```



PRACTICE_code 2: imputation and controlling MAF

1. SNP imputation: use “SNP.impute” option

   ‘Major’: Major allele

   ‘Middle’: heterozygote(default)

   ’Minor’: Minor allele

2. Minor Allele Frequency(MAF) : use “SNP.MAF” option

   MAF to filter SNPs in GWAS Reports, >0 and <1

```
setwd("C:/Users/pc/Desktop/gapitex/result2")
result2 <- GAPIT(Y=myY, G=myX, SNP.impute="Middle", SNP.MAF=0.02)
```

```
maf_index
 FALSE   TRUE
 22947 138729
[1] "Calculating kinship..."
[1] "Number of individuals and SNPs are  370  and  138729"
```

PRACTICE_code 3 : optimal PC number

```
setwd("C:/Users/pc/Desktop/gapitex/result3")
result3 <- GAPIT(Y=myY, G=myX,  SNP.MAF=0.02,  PCA.total=100,  Model.selection=TRUE)
# optimal number of PC using BIC
BICdata <-
read.csv("C:/Users/pc/Desktop/gapitex/result3/GAPIT.MLM.Aspartic.BIC.Model.Selection.Results.csv",
        header=T)
head(BICdata)
BIC <- BICdata[,2]
BICdata[which.max(BIC),1]
```

genomic inflation factor, lambda & QQplot

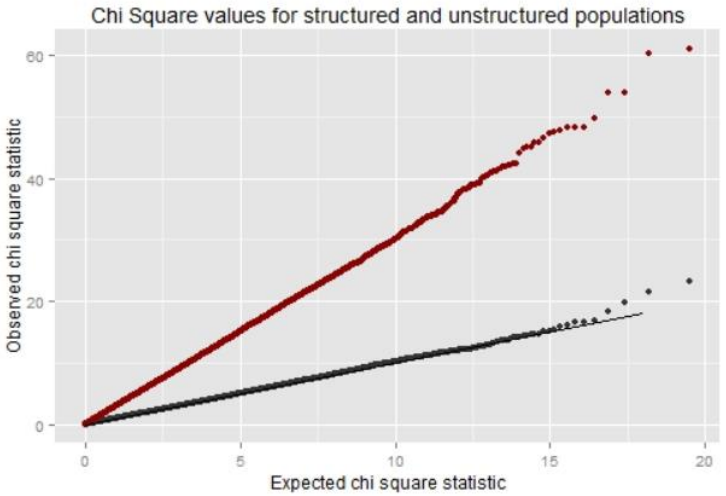
Population structure can lead to false positive associations, unless properly controlled for.

A comparison of observed test statistics (or P values) against an expected distribution, shows when datasets are prone to false positives.

The slope gives an inflation factor (lambda) which is a measure of the degree of the problem.

$$\text{inflation factor } \lambda = \frac{\text{median}(Xi^2)}{0.456}$$

λ > 1 indicates population structure and/or genotyping error.



PRACTICE_population structure using genomic inflation factor

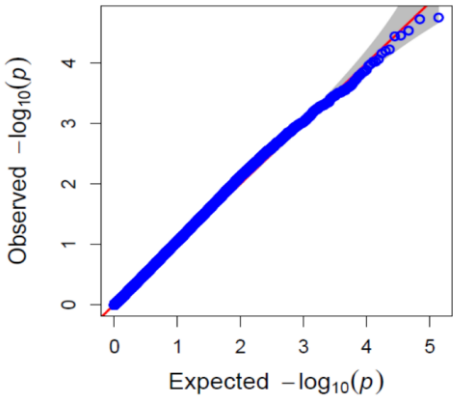
```
pvaldata <-
read.csv("C:/Users/pc/Desktop/gapitex/result3/GAPIT.MLM.Aspartic.GWAS.Results.csv", header=T)
p <- pvaldata[,4]
if (stat_type == "Z")  z = Z[, 1]
if (stat_type == "CHISQ") z = sqrt(CHI[, 1])
if (stat_type == "PVAL")  z = qnorm(p / 2)
lambda = round(median(z^2) / 0.454, 3)
lambda
```

RESULT

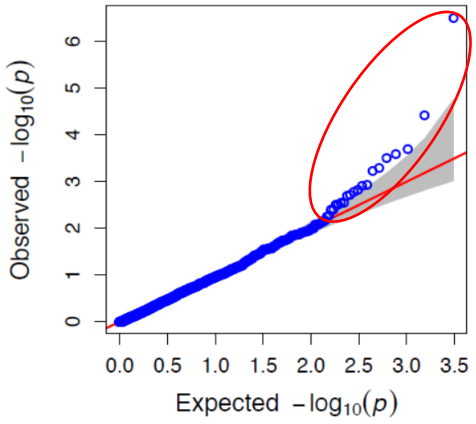
⑤ QQplot

The Y-axis is the observed $-log_{10}(pvalue)$ and the X-axis is the expected $-log_{10}(pvalue)$ under the assumption that the *P*-values follow a uniform[0,1] distribution.

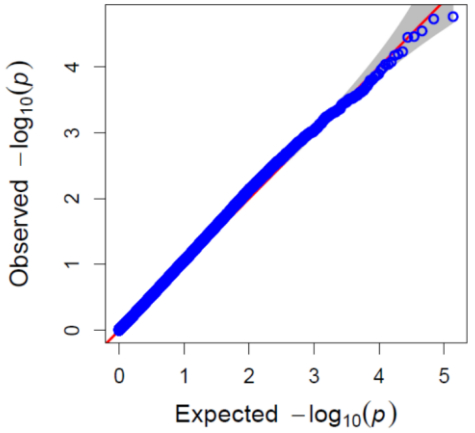
The dotted lines show the 95% confidence interval for the QQ-plot under the null hypothesis of no association between the SNP and the trait.



no association



with association

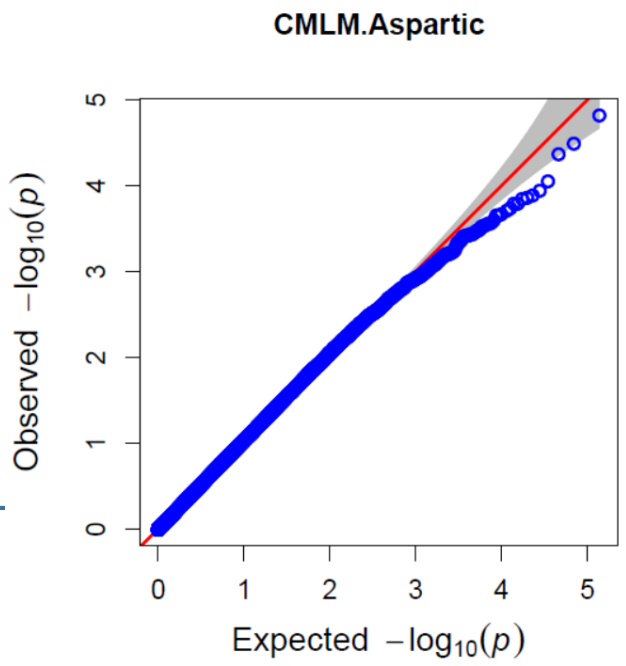


our result

PRACTICE_code4 : Compressed MLM

group.from(start group number), group.to(final group number), group.by(gap of group)

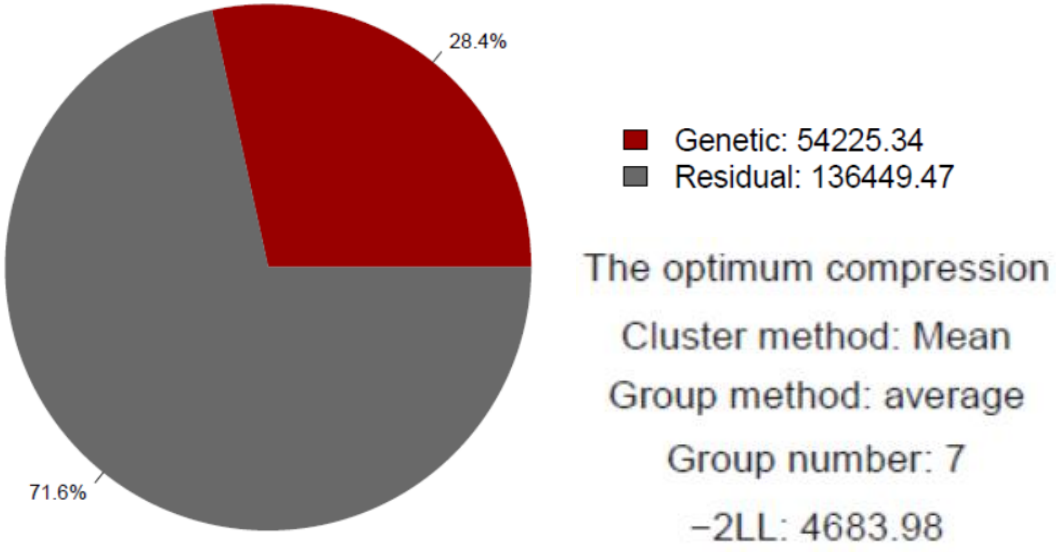
```
setwd("C:/Users/pc/Desktop/gapitex/result4")
result4 <- GAPIT(Y=myY, G=myX,
                 SNP.MAF=0.02,
                 group.to=370,
                 group.from=1,
                 group.by=1,
                 model="CMLM"
                       )
```



RESULT

⑥ Optimum Compression

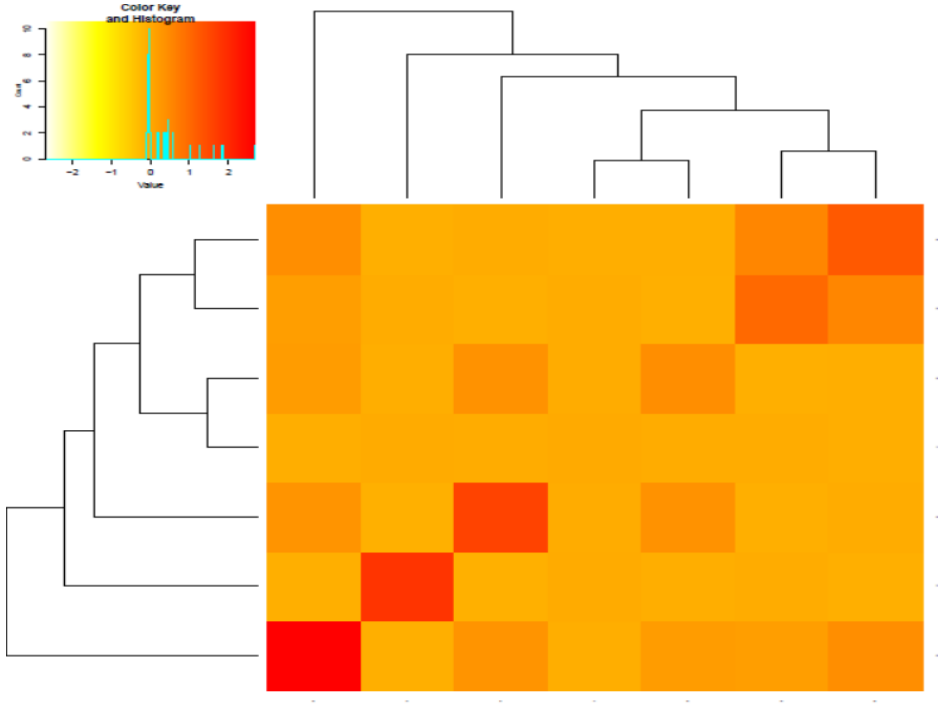
GAPIT select optimal group number which maximize Log Likelihood.



RESULT

⑦ Kinship plot

A heat map of the values in the values in the kinship matrix is created.



RESULT

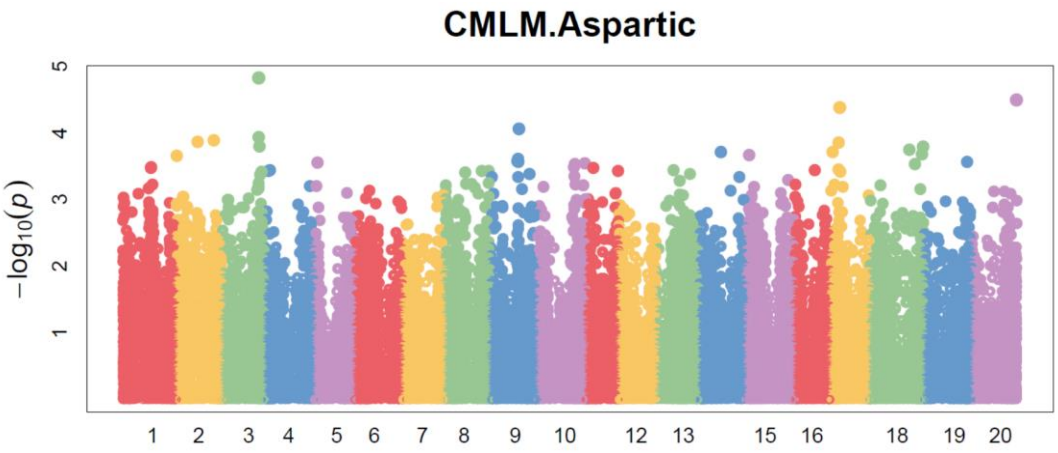
⑧ Manhattan plot

- The Manhattan plot is a scatter plot that summarizes GWAS results.

- The X-axis is the genomic position of each SNP, and the Y-axis is the negative logarithm of the *P*-value obtained from the GWAS model.

- *F*-test for testing H0: No association between the SNP and trait.

- GAPIT's default cutoff value is 0.05



PRACTICE_ code5 : Enriched CMLM

kinship.cluster: average(default), complete, single, ward.D, mcquitty, median, and centroid

kinship.group: Mean, Max, Min, Median (Method to derive kinship among groups)

Optimal setting is considered to be maximizing LL(log likelihood).

```
setwd("C:/Users/pc/Desktop/gapitex/result5")
result5 <- GAPIT(Y=myY, G=myX,
                 SNP.MAF=0.02,
                 kinship.cluster=c("average", "complete", "ward.D"),
                 kinship.group=c("Mean", "Max"),
                 group.from=1,
                 group.to=370,
                 group.by=1,
                 memo="ECMLM"
)
```

In our case, number of Setting:

3("average", "complete", "ward.D") *
2("Mean", "Max") *
370(possible group number)

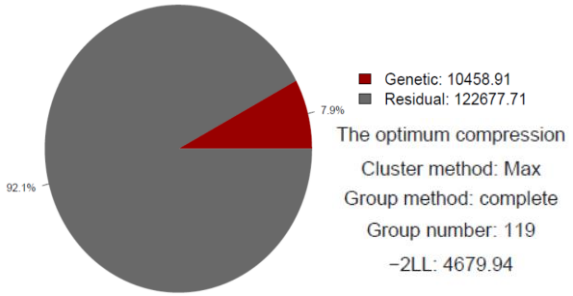
= 2220

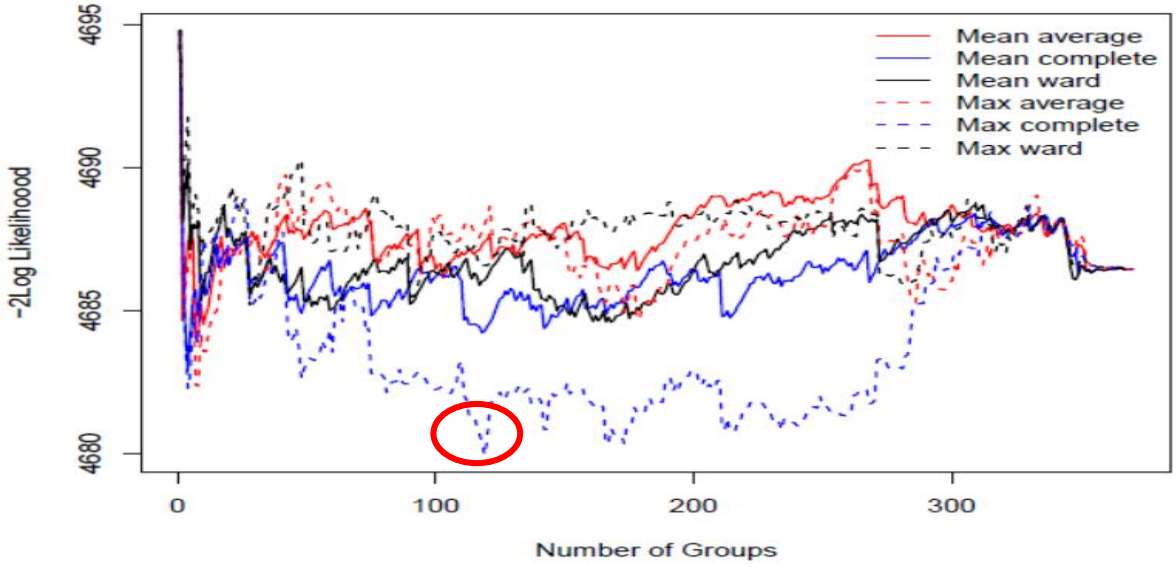
RESULT

⑨ Compression profile over multiple groups

When a range of groups is specified, a different series of graphs are created.

- -2*log likelihood function
- the estimated genetic variance components
- the estimated residual variance component
- the estimated total variance
- the heritability estimate.





PRACTICE_small p-value

```
# small p-value
pvaldata <-
read.csv("C:/Users/pc/Desktop/gapitex/result5/GAPIT.CMLM.Aspartic.GWAS.Results.csv", header=T)
head(pvaldata)
o <- order(pvaldata$P.value)
opvaldata <- pvaldata[o,]
head(opvaldata)
```

Part 2. Regularization

Data description

- Traits (Y)

  - Total sum of 17 essential amino acids

    Continuous variable

- Genomic data (X)

  - 370 samples and 15,000 variants were extracted from the SNP data with 439 samples and 157,762 variants for convenience.

Goals of genome-wide association studies

1. **(Variable selection)**

   - Identify the genetic variants or genomic region associated with a trait or a disease to better understand the biology of disease.

2. **(Prediction)**

   - Make prediction susceptibilities for disease and who is at risk for prevention or better treatment

Variable selection in high-dimensional data

- "Small n, large p" problem

  → Subset selection with "least squares" cannot be used, because the covariance matrix is not invertible.

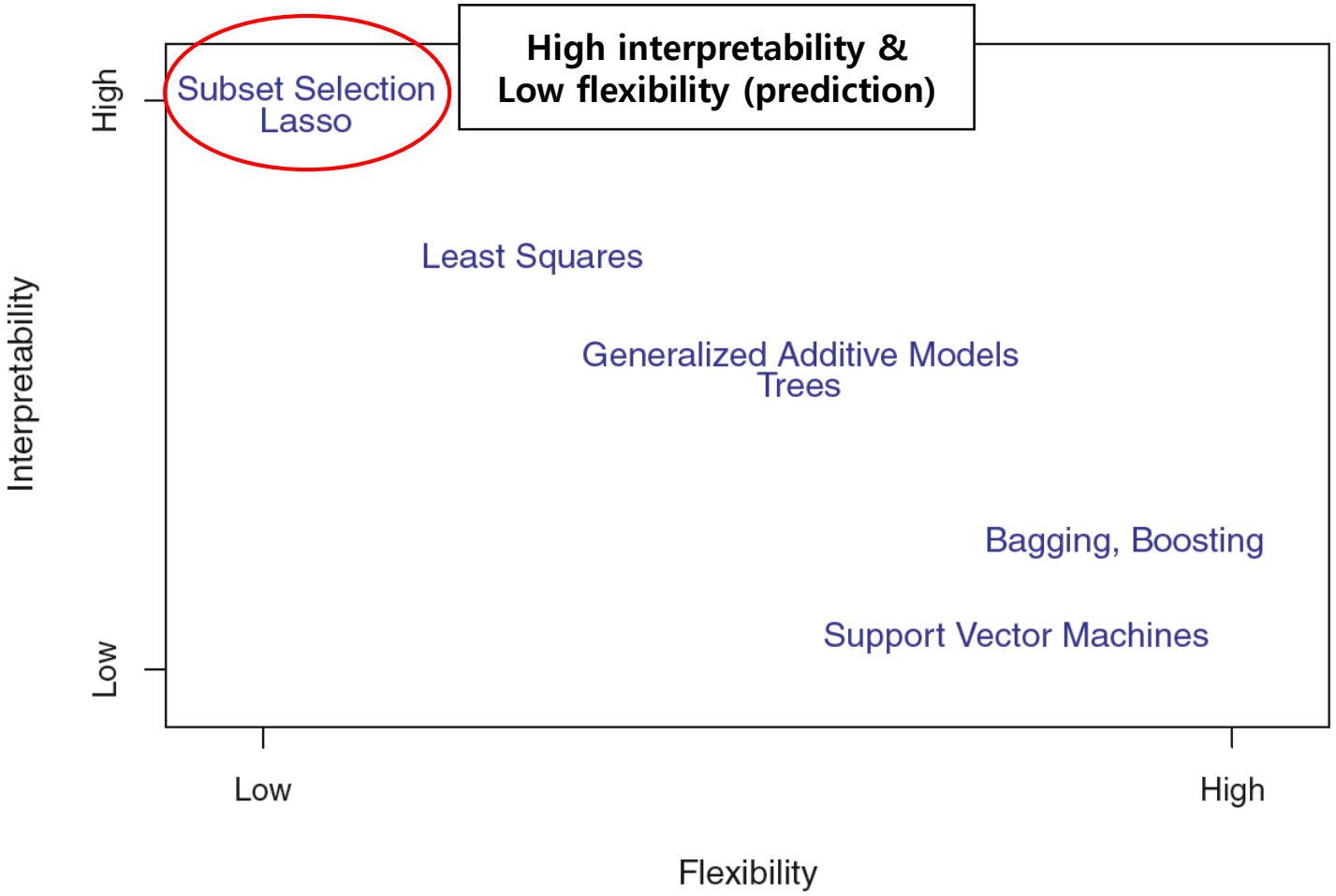
- The answer is regularization using a penalty function

  → A penalty can shrink coefficients which are less associated with response

  ※ Goals of regularization

  - Variable selection
  - Interpretation

The tradeoff between interpretability and flexibility



James, G., et al. (2013) “An Introduction to Statistical Learning with applications in R”

Regularization procedures

$$\hat{\beta} = \arg\min_{\beta} \| Y - X\beta \|_2^2 + P_\lambda(\beta)$$

Least squares

Penalty function

- Ridge
- The Lasso
- Elastic-net
- Group lasso
- Sparse group lasso
- etc…

Ridge

- Ridge

$$\hat{\beta} = \arg\min_{\beta} \left(\frac{1}{2} \| Y - X\beta \|_2^2 + \lambda \| \beta \|_2^2 \right)$$

- (Pros) In the case of highly correlated variables, it prevents inflation of variance of an estimator (by "Multicollinearity").
- [Cons] No matter how high the penalty is, the regression coefficient does not exactly equal to zero. (Variable selection is not possible)

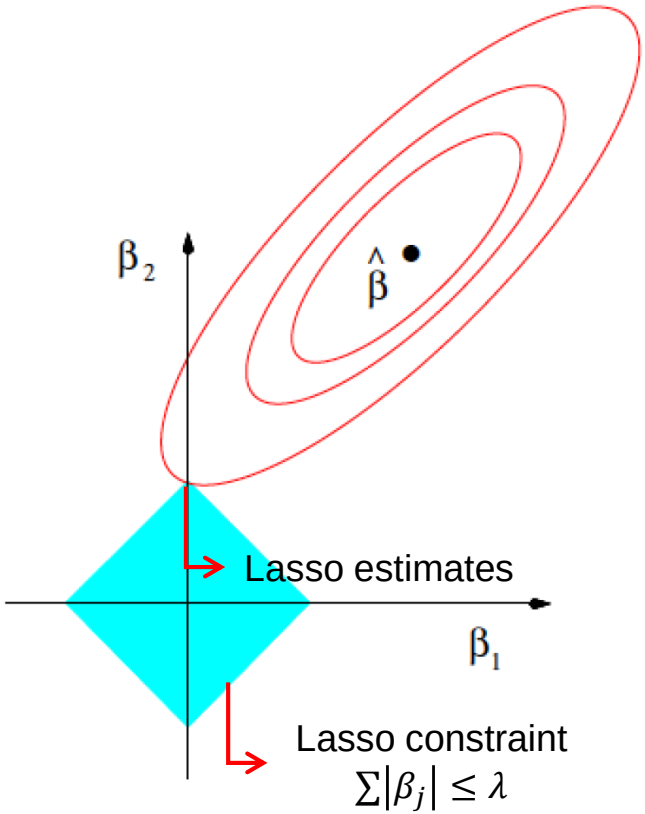
The Lasso

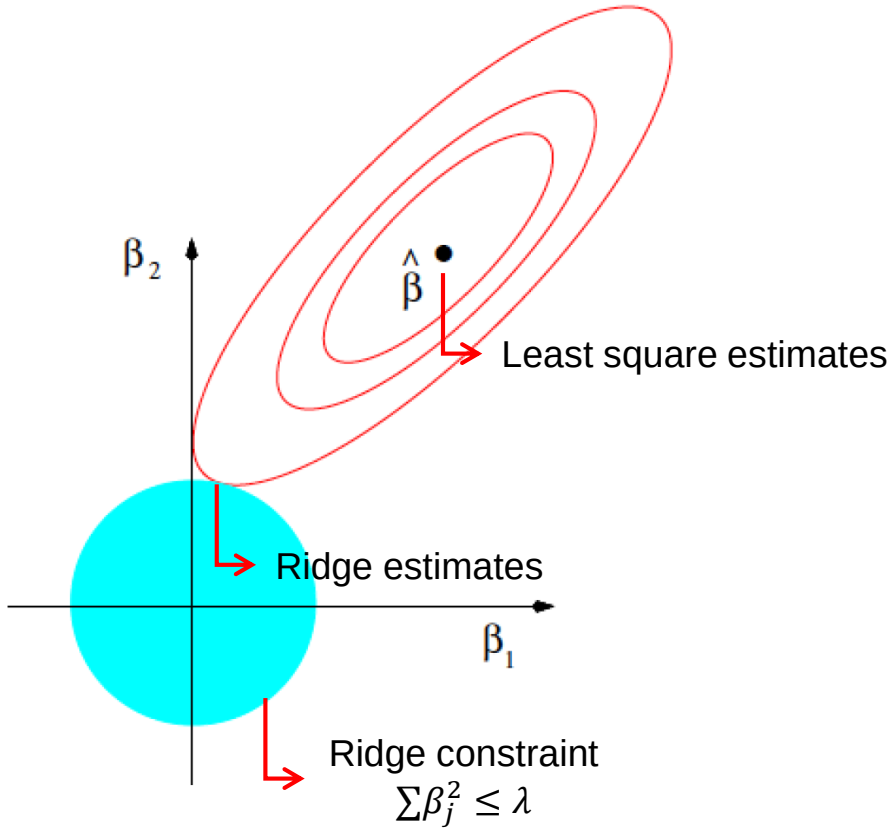
- The Lasso

$$\hat{\beta} = \arg\min_{\beta}(\| Y - X\beta \|_2^2 + \lambda \| \beta \|_1)$$

- (Pros) The coefficients for variables, which is less associated with response, can be exactly zero. (Variable selection is possible)
- [Cons] It selects only one variable among a set of highly correlated variables. (Group selection is not possible)
- [Cons] In the case of n≪ p, it selects at most n variables.

A difference between lasso and ridge





James, G., et al. (2013) “An Introduction to Statistical Learning with applications in R”

Elastic-net

- Elastic-net

$$\hat{\beta} = \arg\min_{\beta} \left(\frac{1}{2} \| Y - X\beta \|_2^2 + \lambda \left[\frac{(1-\alpha)}{2} \| \beta \|_2^2 + \alpha \| \beta \|_1 \right] \right)$$

- (Pros) Even if independent variables are highly correlated, variable selection(group selection) is possible.
- (Pros) In the case of n≪p, it can select at most p variables.
- [Cons] It is important to select the optimal value of two tuning parameters (α, λ).

Others

- Group lasso (Group selection)

$$\hat{\beta} = \arg\min_{\beta} \left(\frac{1}{2} \parallel Y - \sum_{\ell=1}^{L} X_\ell \beta_\ell \parallel_2^2 + \lambda \sum_{\ell=1}^{L} \sqrt{p_\ell} \parallel \beta_\ell \parallel_2 \right)$$

- Sparse group lasso (Selection of variables within a group)

$$\hat{\beta} = \arg\min_{\beta} \left(\frac{1}{2} \parallel Y - \sum_{\ell=1}^{L} X_\ell \beta_\ell \parallel_2^2 + \lambda \left[(1-\alpha) \sum_{\ell=1}^{L} \sqrt{p_\ell} \parallel \beta_\ell \parallel_2 + \alpha \parallel \beta \parallel_1 \right] \right)$$

- etc…

"glmnet"

- Generalized linear model (GLM) regularized with lasso and elastic-NET

- GLMs are commonly used to model various types of response variable, for instance, continuous, binary, count, survival time, and etc.

- In "glmnet", there are also many types of regression, such as **linear**, **logistic**, poisson, and cox regression.

Elements of "glmnet"

```
ls("package:glmnet")
```

```
 [1] "auc"                "auc.mat"            "check_dots"         "coef.cv.glmnet"
 [5] "coef.glmnet"        "coefnorm"           "coxnet"             "coxnet.deviance"
 [9] "cv.coxnet"          "cv.elnet"           "cv.fishnet"         "cv.glmnet"
[13] "cv.lognet"          "cv.mrelnet"         "cv.multnet"         "cvcompute"
[17] "cvtype"             "deviance.glmnet"    "elnet"              "fishnet"
[21] "getcoef"            "getcoef.multinomial" "getmin"            "glmnet"
[25] "glmnet.control"     "glmnet_softmax"     "jerr"               "jerr.coxnet"
[29] "jerr.elnet"         "jerr.fishnet"       "jerr.lognet"        "jerr.mrelnet"
[33] "lambda.interp"      "lognet"             "mrelnet"            "na.mean"
[37] "nonzeroCoef"        "plot.cv.glmnet"     "plot.glmnet"        "plot.mrelnet"
[41] "plot.multnet"       "plotCoef"           "predict.coxnet"     "predict.cv.glmnet"
[45] "predict.elnet"      "predict.fishnet"    "predict.glmnet"     "predict.lognet"
[49] "predict.mrelnet"    "predict.multnet"    "print.glmnet"       "response.coxnet"
[53] "rmult"
```

Main functions of "glmnet"

- glmnet

```
glmnet(x, y,
   family=c("gaussian", "binomial", "poisson",
            "multinomial", "cox", "mgaussian"),
   alpha,
   nlambda,
   labmda,
   standardize=TRUE,
   penalty.factor,
   weights )
```

- cv.glmnet

```
cv.glmnet(x, y,
 family=c("gaussian", "binomial", "poisson", "multinomial", "cox", "mgaussian"),
 alpha,
 type.measure=c("deviance", "mse", "auc"),
 nfolds=10,
 ...
 )
```

The Lasso

The Lasso

the number of lambda value which will be fitted, recommended to be 100 or more.

```
fit.lasso <- glmnet(x, y, alpha=1, nlambda=10, family="gaussian")
fit.lasso
```

α=0 (ridge), α=1 (lasso), and 0<α<1 (elastic-net)

```
Call:  glmnet(x = x, y = y, family = "gaussian", alpha = 1, nlambda = 10)

      Df   %Dev  Lambda
 [1,]  0 0.0000 903.900
 [2,] 13 0.1047 541.800
 [3,] 84 0.4047 324.800
 [4,] 180 0.6906 194.700
 [5,] 252 0.8522 116.700
 [6,] 310 0.9313  69.980
 [7,] 349 0.9704  41.950
 [8,] 386 0.9884  25.150
 [9,] 407 0.9955  15.080
[10,] 439 0.9983   9.039
```

- Lambda : the lambda value determining the strength of penalty
- Df : # of the nonzero coefficients
- %Dev : the percentage of deviance explained (for gaussian, R-square)

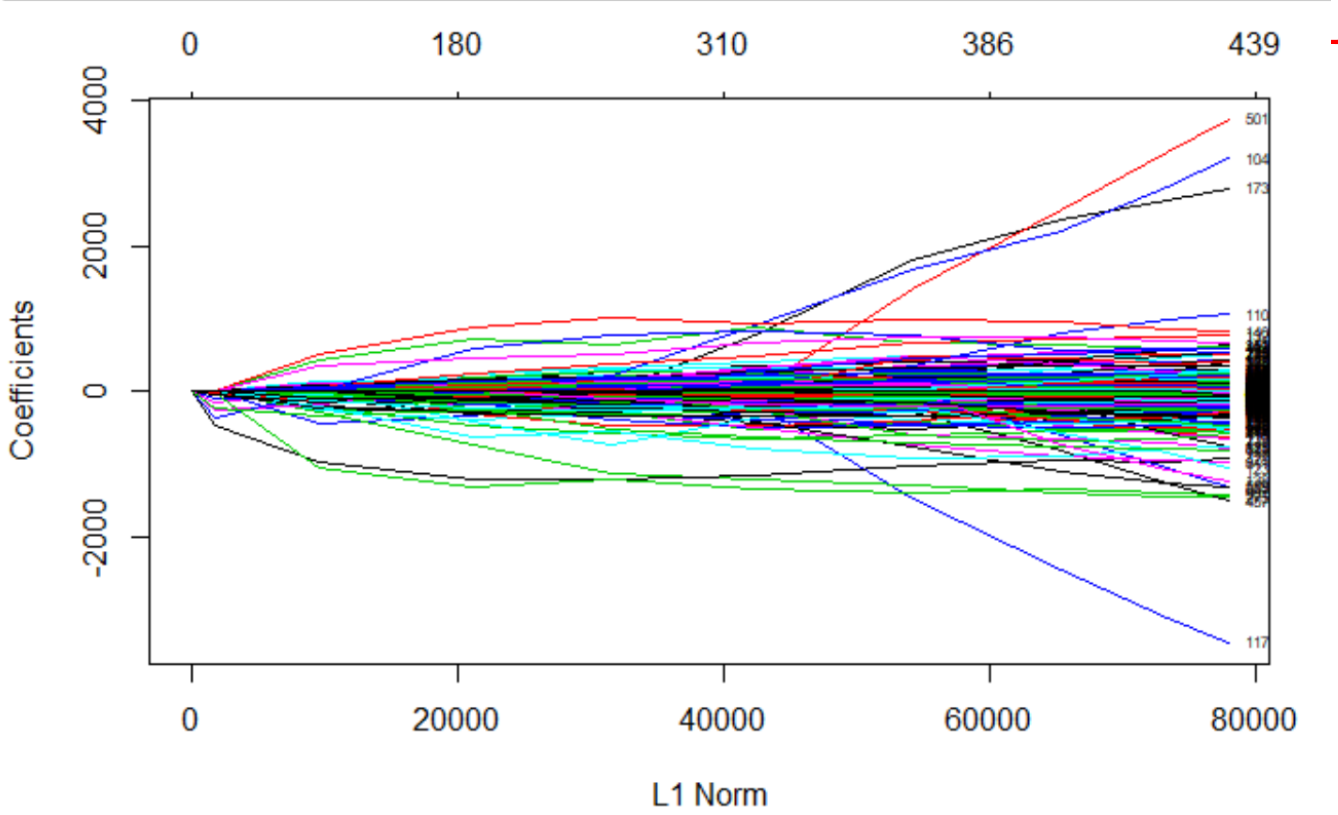
This is not an absolute criterion, because %Dev increases as variables are added.

Solution path for lasso

If TRUE, labels the curves with variable numbers.

```
plot( fit.lasso, xvar="norm", label=TRUE )
```

Measurement on the X-axis includes “norm”, “lambda”, and “dev”.



The number of nonzero coefficients

Sum of absolute value of regression coefficients

How to get coefficients

```
dim( fit.lasso$beta )
```

```
[1] 15000    10
```

Hide

```
head( fit.lasso$beta )
```

```
6 x 10 sparse Matrix of class "dgCMatrix"
```

```
   [[ suppressing 10 column names 慎牦慎狂s0慎牦慎貊, 慎牦慎狂s1慎牦慎貊, 慎牦慎狂s2慎牦慎貊 ... ]]
```

```
                 λ1        ⋯        λ10
AX.90435719 . . . . . . . . . .
AX.90432064 . . . . . . . . . .
AX.90454689 . . . . . . . . . .
AX.90332566 . . . . . . . . . .
AX.90496155 . . . . . . . . . .
AX.90443918 . . . . . . . . . .
```

→ Sparse matrix to reduce memory usage

Others to retrieve coefficients for a specific λ

- For the 5th lambda,

  - "coef"

Set a specific λ value

```
beta_coef <- coef( fit.lasso, s = fit.lasso$lambda[5] )
```

  - "predict"

```
beta_pred <- predict( fit.lasso, type="coef", s = fit.lasso$lambda[5] )
```

To get coefficients

Set a specific λ value

The results of three functions are same

```
tmp <- c(33, 54, 192, 350, 361, 435)
```

For the 5th λ value

```
fit.lasso$beta[tmp, 5]
```

```
AX.90373344 AX.90316606 AX.90507036 AX.90361043 AX.90447851 AX.90486253
-144.513722  -19.347762   -1.024381  -21.043062  -60.090700  -60.462772
```

```
beta_coef[tmp+1,]
```

→ "coef" and "predict" have an intercept ($\hat{\beta}_0$)

```
AX.90373344 AX.90316606 AX.90507036 AX.90361043 AX.90447851 AX.90486253
-144.513722  -19.347762   -1.024381  -21.043062  -60.090700  -60.462772
```

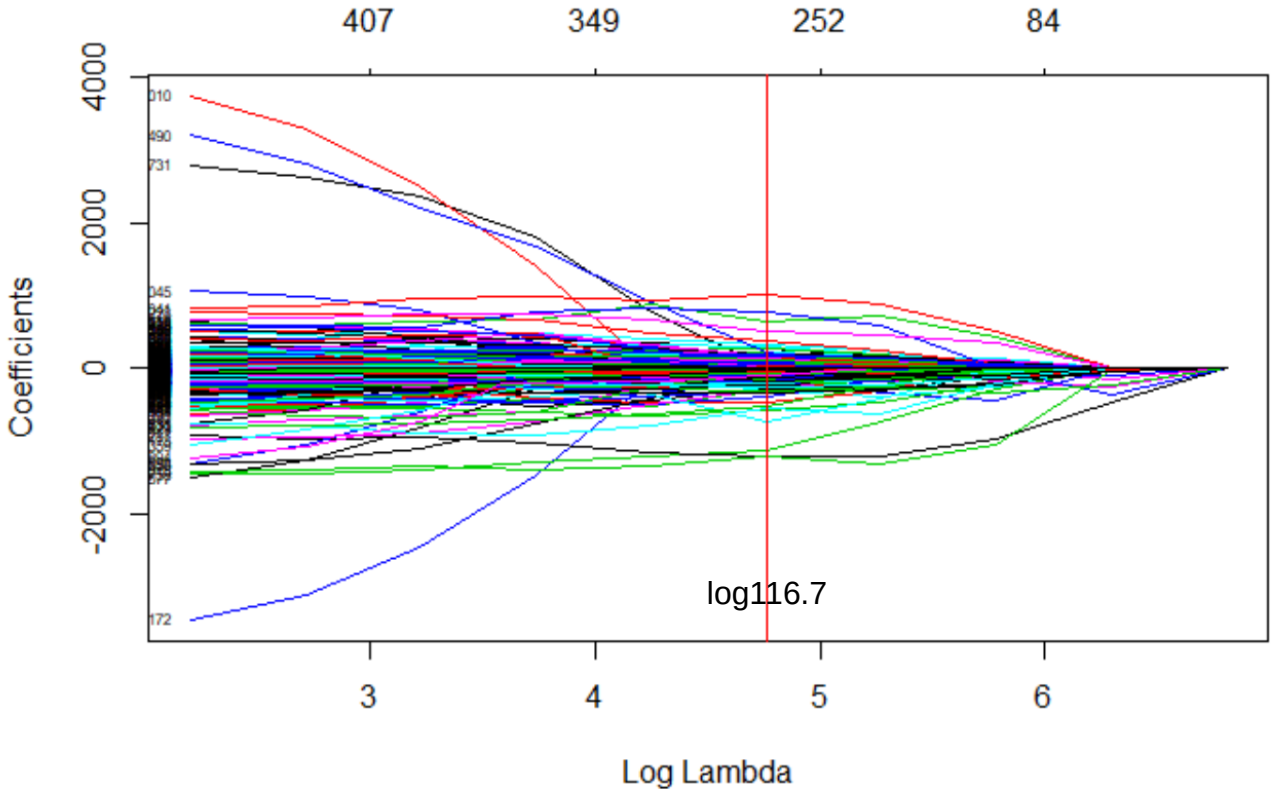
```
beta_pred[tmp+1,]
```

```
AX.90373344 AX.90316606 AX.90507036 AX.90361043 AX.90447851 AX.90486253
-144.513722  -19.347762   -1.024381  -21.043062  -60.090700  -60.462772
```

Coefficients in solution path

```
plot(fit.lasso, xvar="lambda", label=TRUE)
abline(v=log(116.7), cex=2, col="red")
```

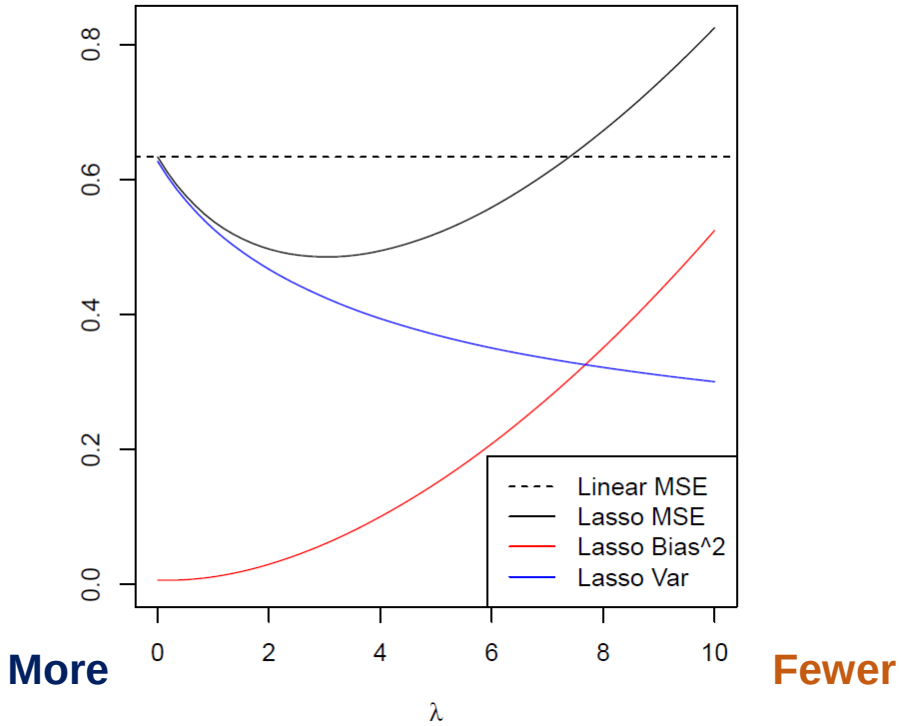
↳ The 5th λ value of "fit.lasso" = 116.7



Which is better, more and fewer?

- As shown above, the coefficients depend on lambda.
- Which is the best, having **more** variables or **fewer** variables?
- (Example) n=50, p=30; true coefficients: 10 large, 20 small

  <Goal> Lambda value which has the smallest MSE (black curve)



James, G., et al. (2013) "An Introduction to Statistical Learning with applications in R"

Model selection

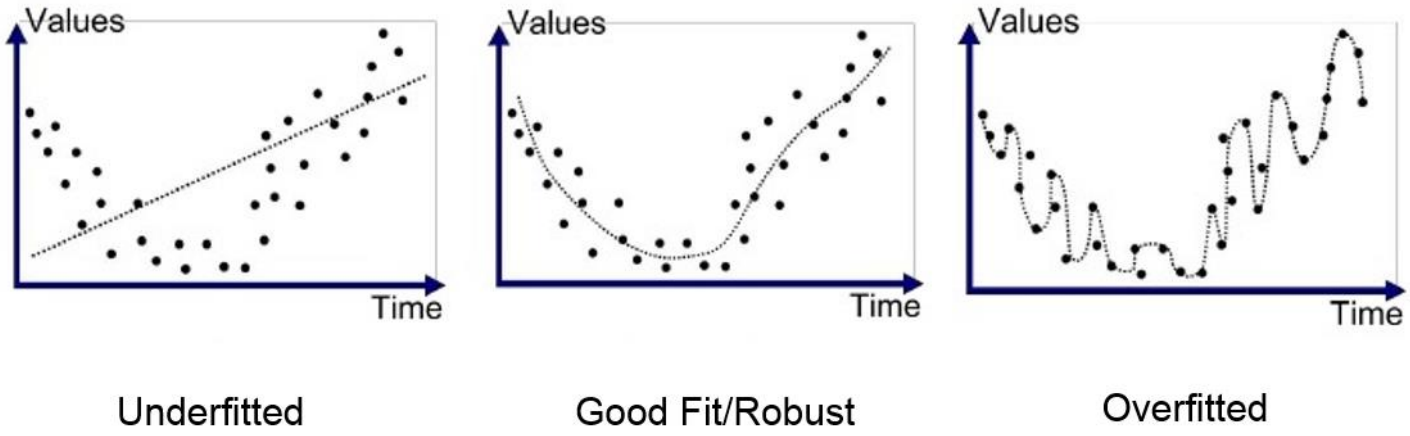
- How to choose the optimal lambda value?
- Model selection
  - (Goal) To select a model which has the best model criterion.
  - λ sequence : $\lambda_1, \lambda_2, \cdots, \lambda_{100}$
  - Model criteria : (1) Mean squared error
    (2) Deviance
    Binary response { (3) Misclassification error
    (4) AUC (Area Under the Curve)

fit.lasso

	Df	%Dev	Lambda
[1,]	0	0.0000	903.900
[2,]	13	0.1047	541.800
[3,]	84	0.4047	324.800
[4,]	180	0.6906	194.700
[5,]	252	0.8522	116.700
[6,]	310	0.9313	69.980
[7,]	349	0.9704	41.950
[8,]	386	0.9884	25.150
[9,]	407	0.9955	15.080
[10,]	439	0.9983	9.039

Overfitting

- Overfitting : construct a complicated model which do not explain new datasets.



Cross-validation to prevent overfitting

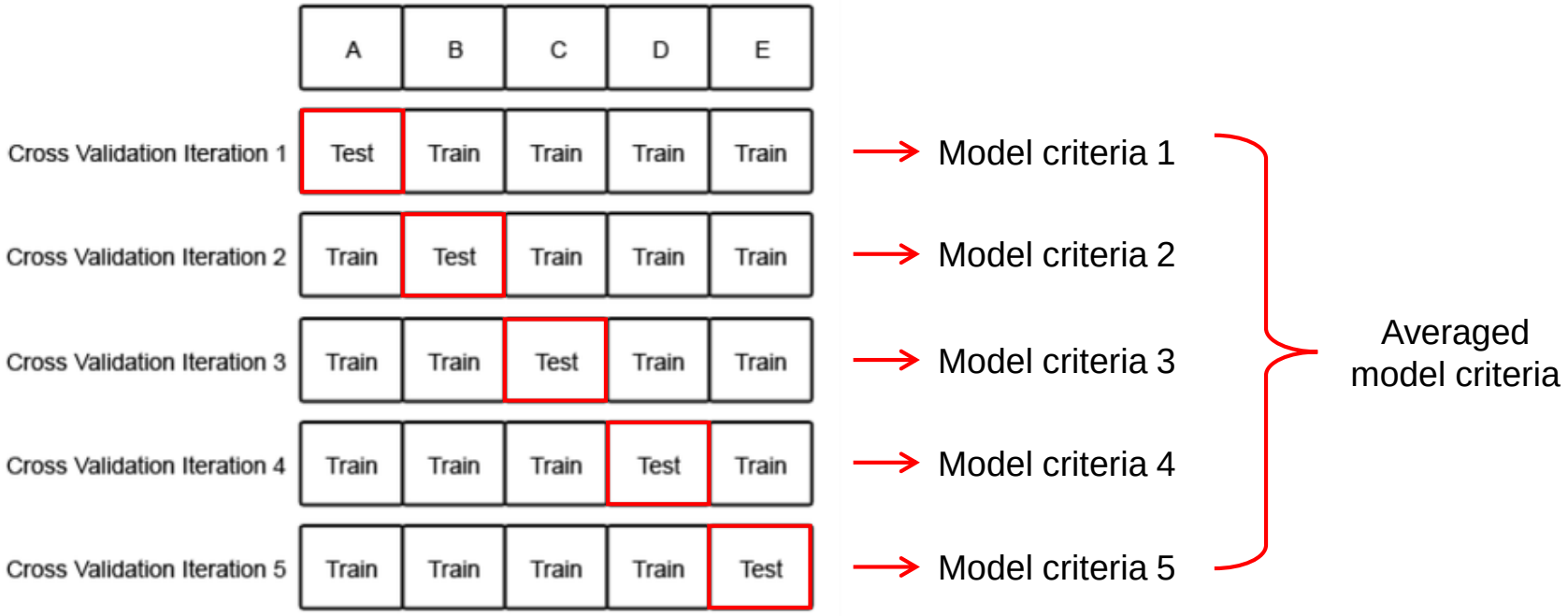
- How to avoid the overfitting problem?

  - The answer is K-fold Cross-validation

- Cross-validation is to apply model selection for training set and test set to avoid overfitting

Cross-validation

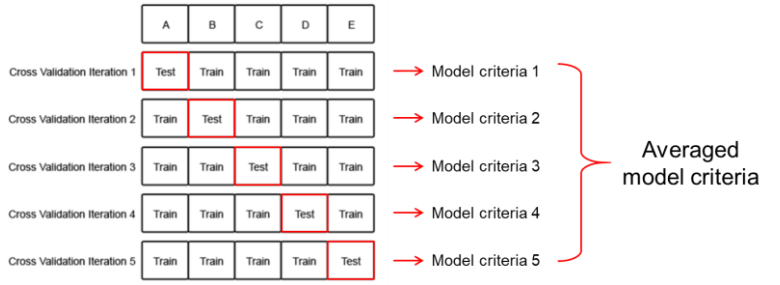
- Example) 5-fold cross-validation



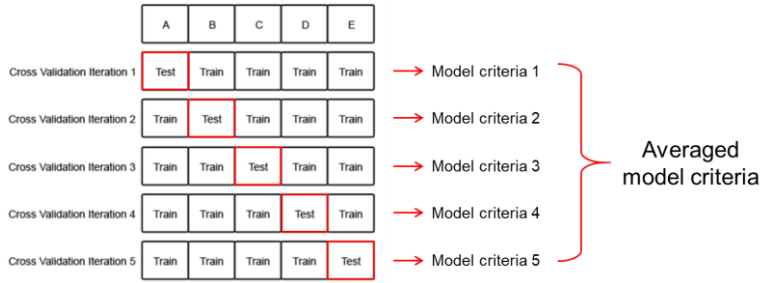
Procedure of selecting tuning parameter

- λ sequence : $\lambda_1, \lambda_2, \cdots, \lambda_{100}$

λ_1



λ_{100}



Select the optimal λ,
which has the smallest
the averaged model criteria

Cross-validation in "glmnet"

- cv.glmnet

Model criterion

```
cv.lasso <- cv.glmnet(x, y, alpha=1, nlambda=100, family="gaussian", type.measure = "mse")
str(cv.lasso, max.level = 1)
```

```
List of 10
 $ lambda    : num [1:98] 904 863 824 786 750 ...
 $ cvm       : num [1:98] 9985323 9969938 9938710 9884819 9837412 ...
 $ cvsd      : num [1:98] 765746 769471 769759 773887 778668 ...
 $ cvup      : num [1:98] 10751069 10739408 10708469 10658706 10616080 ...
 $ cvlo      : num [1:98] 9219576 9200467 9168951 9110932 9058744 ...
 $ nzero     : Named int [1:98] 0 2 2 3 3 3 5 7 8 9 ...
  ..- attr(*, "names")= chr [1:98] "s0" "s1" "s2" "s3" ...
 $ name      : Named chr "Mean-Squared Error"
  ..- attr(*, "names")= chr "mse"
 $ glmnet.fit:List of 12
  ..- attr(*, "class")= chr [1:2] "elnet" "glmnet"
 $ lambda.min: num 623
 $ lambda.1se: num 904
 - attr(*, "class")= chr "cv.glmnet"
```

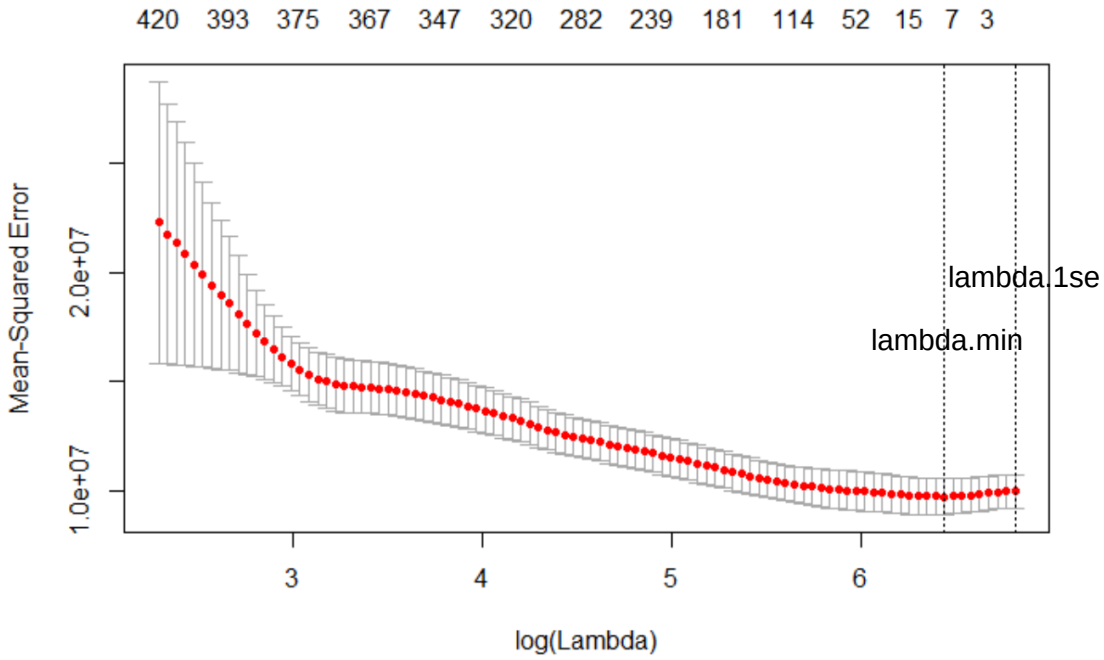
The optimal lambda values

cvm : averaged mse
cvsd : standard deviation of mse
cvup : cvm + cvsd
cvlo : cvm - cvsd
nzero : # of nozero coefficients
name : name of model criterion

Cross-validation in "glmnet"

- Cross-validation curve with error bars of SE

```
plot(cv.lasso)
```



- lambda.min: lambda value which has the smallest mean-squared error
- lambda.1se: lambda value which gives the most regularized model such that error is within one standard error of minimum

Optimal tuning parameter

```
fit.lasso.min <- glmnet(x, y, alpha=1, family="gaussian", lambda=cv.lasso$lambda.min)
fit.lasso.1se <- glmnet(x, y, alpha=1, family="gaussian", lambda=cv.lasso$lambda.1se)
fit.lasso.min
fit.lasso.1se
```

```
Call:  glmnet(x = x, y = y, family = "gaussian", alpha = 1, lambda = cv.lasso$lambda.min)

     Df    %Dev Lambda
[1,]  8 0.06801    623
```

→ Recommend !

```
Call:  glmnet(x = x, y = y, family = "gaussian", alpha = 1, lambda = cv.lasso$lambda.1se)

     Df      %Dev Lambda
[1,]  1 9.567e-17  903.9
```

→ Too few

```
lasso.nonzero.min <- which( fit.lasso.min$beta != 0 )
lasso.nonzero.min
```

```
[1]  1163  1366  4287  6456  7340  8494  9796 12294
```

Comparison with univariate analysis

Comparison I

- Univariate analysis for each variable

  (example) Linear regression of the first variant on a trait

```
summary( lm(y~x[, 1]) )
```

```
Call:
lm(formula = y ~ x[, 1])

Residuals:
    Min      1Q  Median      3Q     Max
-8183.5 -1930.5   405.2  2282.6  7763.5

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 43055.04     224.18 192.052   <2e-16 ***
x[, 1]         74.88     175.47   0.427     0.67
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3154 on 368 degrees of freedom
Multiple R-squared:  0.0004946, Adjusted R-squared:  -0.002221
F-statistic: 0.1821 on 1 and 368 DF,  p-value: 0.6698
```

Comparison II

- For-loop over all variables to get p-values of univariate analysis

```
lm.pvalue <- NULL
for( j in 1:ncol(x) ){
  lm.coef <- summary( lm(y~x[, j]) )$coef
  if( nrow(lm.coef)==2 ){
    lm.pvalue[j] <- lm.coef[2, 4]
  } else if ( nrow(lm.coef)==1 ){
    lm.pvalue[j] <- NA
  }
}
```

```
str( lm.pvalue )
```

```
 num [1:15000] 0.67 0.588 0.688 0.203 0.262 ...
```

```
summary( lm.pvalue )
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 0.0000  0.0951  0.3462  0.3933  0.6619  1.0000     633
```

→ Variables which have only one value

Comparison III

- Struct a data.frame with a reference

```
df.lm.pvalue <- data.frame( ref, p=lm.pvalue)
colnames(df.lm.pvalue) <- c("SNP", "CHR", "BP", "P")
(list(ref=head(ref), df.lm.pvalue=head(df.lm.pvalue)))
```

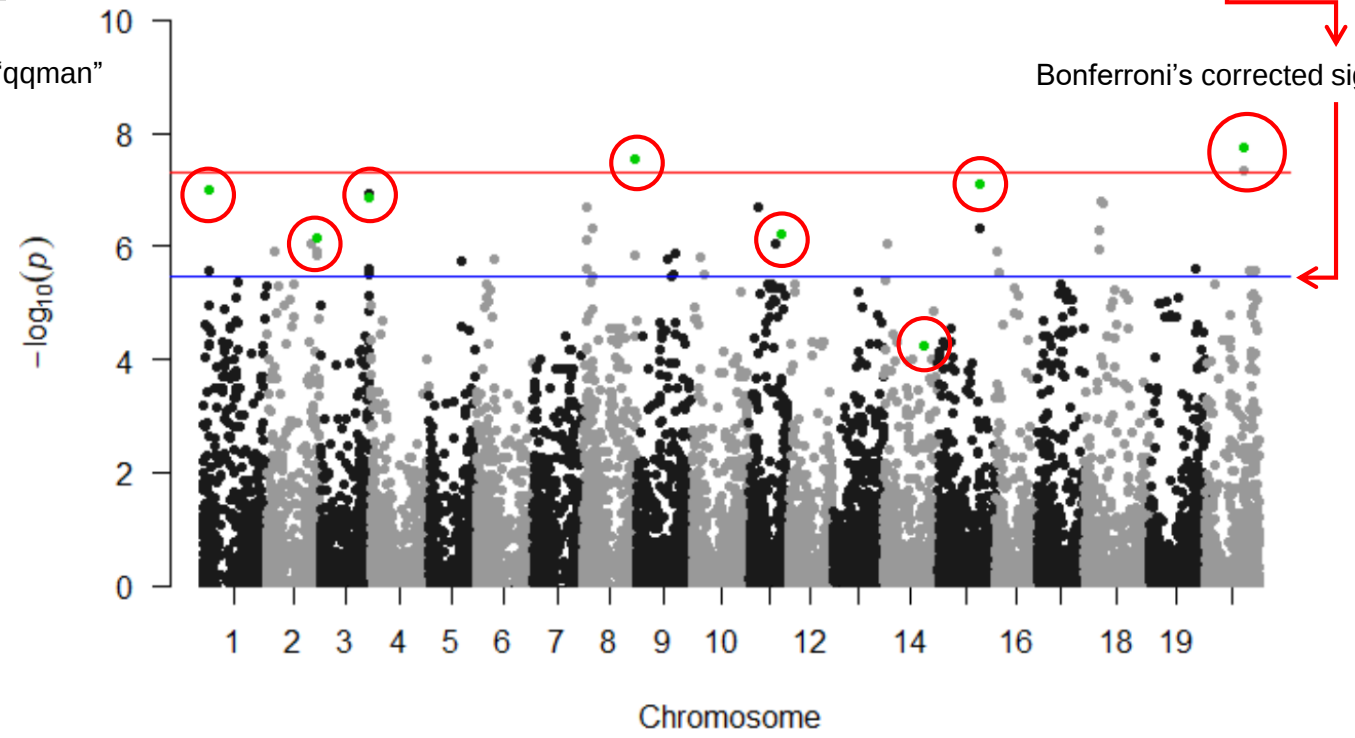
```
$ref
                rs chrom      pos
47207   AX-90435719     6 19444456
84      AX-90432064     1   342857
86513   AX-90454689    10 50203642
2388    AX-90332566     1 13870972
10966   AX-90496155     2  8404704
161363 AX-90443918    20   482799

$df.lm.pvalue
                SNP CHR       BP          P
47207   AX-90435719   6 19444456 0.66981825
84      AX-90432064   1   342857 0.58762581
86513   AX-90454689  10 50203642 0.68812025
2388    AX-90332566   1 13870972 0.20327856
10966   AX-90496155   2  8404704 0.26238478
161363 AX-90443918  20   482799 0.05218934
```

Manhattan plot

```
bonferroni.cutoff <- 0.05/ncol(x)
qqman::manhattan(df.lm.pvalue, chr="CHR", bp="BP", p="P", snp="SNP", suggestiveline=-log10(bonferroni.cutoff),
                 highlight=df.lm.pvalue[lasso.nonzero.min,"SNP"], ylim=c(0,10))
```

R package “qqman”

Bonferroni’s corrected significance level = α/p



- Green dots: variables selected by lasso
- Red line: genome-wide significant line [=-log10(5.0e-08)]
- Blue line: Bonferroni significant line [=-log10(3.3e-06)]

Prediction I

- Splitting training and test set

```
set.seed(20190130)
idx.train <- sample(nrow(x), 0.9*nrow(x))
idx.test <- (1:nrow(x))[-idx.train]
```

- Cross-validation

```
cv.lasso.train <- cv.glmnet(x[idx.train,], y[idx.train], alpha=1, family="gaussian")
```

- Fitting lasso with optimal lambda (min & 1se)

```
fit.lasso.train.min <- glmnet(x[idx.train,], y[idx.train], alpha=1, family="gaussian", lambda=cv.lasso.train$lambda.min)
fit.lasso.train.1se <- glmnet(x[idx.train,], y[idx.train], alpha=1, family="gaussian", lambda=cv.lasso.train$lambda.1se)
```

Prediction II

- Results of the lasso with the optimal lambda values

```
fit.lasso.train.min
```

```
Call:  glmnet(x = x[idx.train, ], y = y[idx.train], family = "gaussian",      alpha = 1, lambda = cv.lasso.train$lambda.min)

     Df    %Dev Lambda
[1,]  3 0.06137  702.4
```

```
fit.lasso.train.1se
```

```
Call:  glmnet(x = x[idx.train, ], y = y[idx.train], family = "gaussian",      alpha = 1, lambda = cv.lasso.train$lambda.1se)

     Df     %Dev Lambda
[1,]  1 0.009172  972.8
```

Prediction III

- Predicting a new dataset with the lasso-trained model.

```
newx <- x[idx.test, ]
newy.fit.min <- predict( fit.lasso.train.min, newx, type="response" )
head(
  data.frame(
    test.y=y[idx.test],
    fit.newy.min=as.numeric(newy.fit.min)
  )
)
```

```
    test.y fit.newy.min
1 42908.20     43131.13
2 42897.82     42600.05
3 36763.64     43131.13
4 43530.82     43131.13
5 39341.26     43131.13
6 43512.29     43131.13
```

※ Notification
Since the lasso is not a prediction model, so the prediction results is not good

Elastic-net

Elastic-net for binary response

- Transform “y” into binary data based on median value

```
y2 <- ifelse( y > median(y), 1, 0 )
table(y2)
```

```
y2
  0   1
185 185
```

```
fit.enet <- glmnet(x, y2, alpha=0.5, nlambda=10, family="binomial")
fit.enet
```

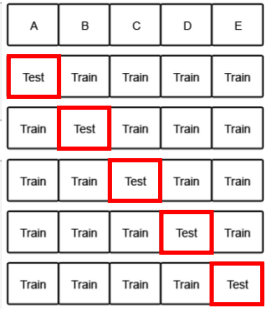
In the case of binary response, family=“binomial”
(logistic regression model)

```
Call:  glmnet(x = x, y = y2, family = "binomial", alpha = 0.5, nlambda = 10)

       Df      %Dev   Lambda
 [1,]   0 7.528e-15 0.268100
 [2,]  27 7.385e-02 0.160700
 [3,] 119 3.403e-01 0.096360
 [4,] 230 5.844e-01 0.057760
 [5,] 300 7.473e-01 0.034630
 [6,] 358 8.468e-01 0.020760
 [7,] 411 9.076e-01 0.012440
 [8,] 447 9.443e-01 0.007460
 [9,] 476 9.664e-01 0.004472
[10,] 512 9.797e-01 0.002681
```

Selecting the optimal tuning parameters

```
set.seed(1234)
foldid <- sample(rep(1:10, length=length(y)))
```

→ To compare the measurement for each alpha with the same dataset.




```
alpha.grid <- seq(0, 1, 0.1)
cv.enet <- as.list( 1:length(alpha.grid) )
for( h in 1:length(alpha.grid) ){
  cv.enet[[h]] <- cv.glmnet(x, y2, family="binomial", type.measure="dev",
                            alpha=alpha.grid[h], nlambda=15,
                            foldid=foldid)
}
```

```
load("[object]cv.enet.RData")
cv.enet.cvm <- NULL
for( h in 1:length(cv.enet)){
  cv.enet.cvm <- cbind(cv.enet.cvm, cv.enet[[h]]$cvm)
}
dimnames(cv.enet.cvm) <- list( paste0("lambda.",1:14), paste0("alpha.",1:11) )

opt.tune <- which( cv.enet.cvm == min(cv.enet.cvm), arr.ind=TRUE )
opt.alpha <- alpha.grid[ opt.tune[2] ]
opt.lambda.min <- cv.enet[[ opt.tune[2] ]]$lambda.min
opt.lambda.1se <- cv.enet[[ opt.tune[2] ]]$lambda.1se

c(opt.alpha=opt.alpha, opt.lambda.min=opt.lambda.min, opt.lambda.1se=opt.lambda.1se)
```

```
##      opt.alpha opt.lambda.min opt.lambda.1se
##      0.1000000      0.6943505      0.9647969
```

Additional grid search for alpha

```
alpha.grid2 <- seq(0.01, 0.1, 0.02)
cv.enet2 <- as.list( 1:length(alpha.grid2) )
for( h in 1:length(alpha.grid2) ){
  cv.enet2[[h]] <- cv.glmnet(x, y2, family="binomial", type.measure="dev",
                             alpha=alpha.grid2[h], nlambda=15,
                             foldid=foldid)
}
# save(cv.enet2, alpha.grid2, file="[object]cv.enet2.RData")
```

```
load("[object]cv.enet2.RData")
cv.enet.cvm2 <- NULL
for( h in 1:length(cv.enet2)){
  cv.enet.cvm2 <- cbind(cv.enet.cvm2, cv.enet2[[h]]$cvm)
}
dimnames(cv.enet.cvm2) <- list( paste0("lambda.",1:14), paste0("alpha.",1:5) )

opt.tune2 <- which( cv.enet.cvm2 == min(cv.enet.cvm2), arr.ind=TRUE )
opt.alpha2 <- alpha.grid2[ opt.tune2[2] ]
opt.lambda.min2 <- cv.enet2[[ opt.tune2[2] ]]$lambda.min
opt.lambda.1se2 <- cv.enet2[[ opt.tune2[2] ]]$lambda.1se

c(opt.alpha2=opt.alpha2, opt.lambda.min2=opt.lambda.min2, opt.lambda.1se2=opt.lambda.1se2)
```

```
##      opt.alpha2 opt.lambda.min2 opt.lambda.1se2
##        0.010000        3.596371        4.997141
```

→ The optimal alpha may be in [0, 0.01),
but we do not investigate the alpha range, in which
the models have too many variables.

Our final model selection

```
fit.enet.opt.min <- glmnet(x, y2, family="binomial", alpha=opt.alpha, lambda=opt.lambda.min)
fit.enet.opt.1se <- glmnet(x, y2, family="binomial", alpha=opt.alpha, lambda=opt.lambda.1se)
fit.enet.opt.min2 <- glmnet(x, y2, family="binomial", alpha=opt.alpha2, lambda=opt.lambda.min2)
fit.enet.opt.1se2 <- glmnet(x, y2, family="binomial", alpha=opt.alpha2, lambda=opt.lambda.1se2)
```

```
fit.enet.opt.min
```

```
##
## Call:  glmnet(x = x, y = y2, family = "binomial", alpha = opt.alpha,       lambda = opt.lambda.min)
##
##       Df    %Dev Lambda
## [1,] 105 0.09748 0.6944
```

→ We select this as our final model.

```
fit.enet.opt.min2
```

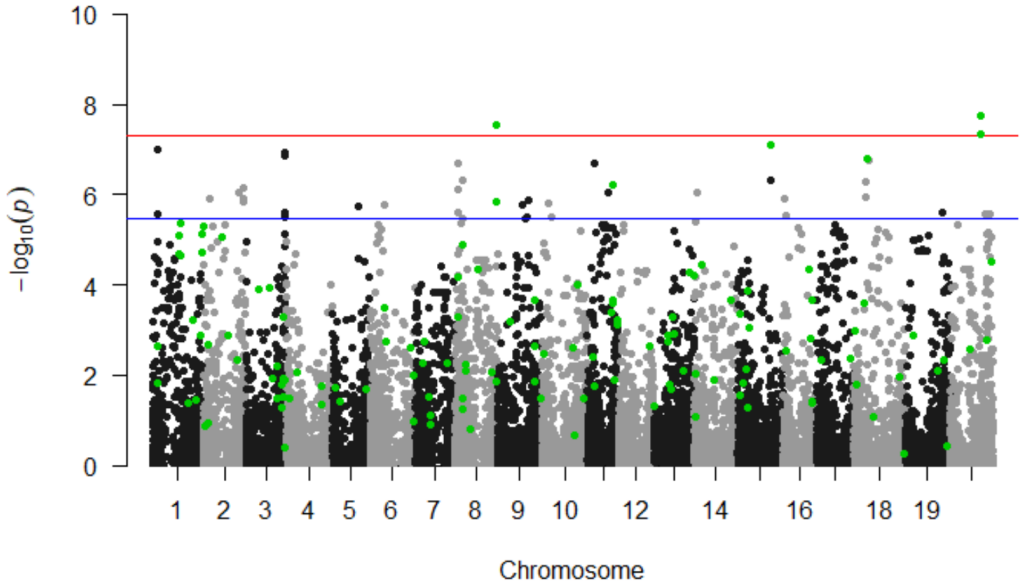
```
##
## Call:  glmnet(x = x, y = y2, family = "binomial", alpha = opt.alpha2,       lambda = opt.lambda.min2)
##
##       Df    %Dev Lambda
## [1,] 1269 0.2406  3.596
```

→ Too many.

Remember that
choosing lambda for the goal of reducing MSE, for the sake of variable selection, is somewhat of a different task.

Comparison in manhattan plot

```
nonzero.enet.opt <- which( fit.enet.opt.min$beta != 0 )
bonferroni.cutoff <- 0.05/ncol(x)
qqman::manhattan(df.lm.pvalue,
                 chr="CHR", bp="BP", p="P", snp="SNP",
                 suggestiveline=-log10(bonferroni.cutoff),
                 highlight=df.lm.pvalue[nonzero.enet.opt,"SNP"],
                 ylim=c(0,10))
```



- Green dots: variables selected by elastic-net
- Red line: genome-wide significant line [=-log10(5.0e-08)]
- Blue line: Bonferroni significant line [=-log10(3.3e-06)]

Covariate-adjusted model

Arbitrary covariates related with “y2”

- Generating arbitrary covariates from probability distribution

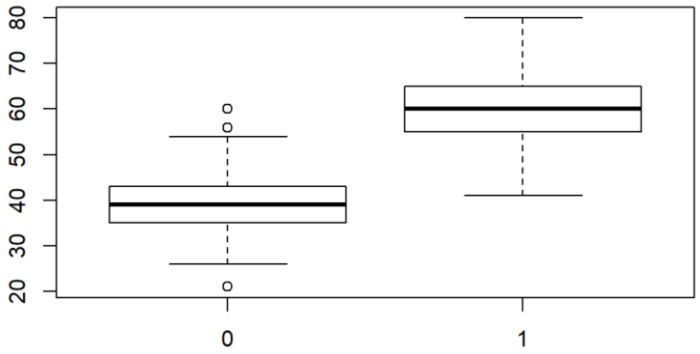
```
cov1 <- NULL
cov1[y2==0] <- rbinom(sum(y2==0), 1, prob=0.8)
cov1[y2==1] <- rbinom(sum(y2==1), 1, prob=0.4)
table(y2, cov1)
```

→ probability of getting “1” = 0.8

```
##     cov1
## y2     0   1
##    0  38 147
##    1 109  76
```

```
cov2 <- NULL
cov2[y2==0] <- rpois(sum(y2==0), lambda=40)
cov2[y2==1] <- rpois(sum(y2==1), lambda=60)
boxplot(cov2~y2)
```

→ Poisson distribution with mean of 40



Including covariates in "glmnet"

- The variables with penalty.factor=0 are always included.

```
x.cov <- cbind(cov1, cov2, x)

pf <- c(0, 0, rep(1,ncol(x)) )
table(pf)
```

→ Set the first two elements of penalty.factor to 0

```
## pf
##     0     1
##     2 15000
```

```
pf.fit <- glmnet(x.cov, y2, family="binomial", penalty.factor = pf, nlambda=10 )
pf.fit
```

```
##
## Call:  glmnet(x = x.cov, y = y2, family = "binomial", nlambda = 10,      penalty.factor = pf)
##
##       Df   %Dev    Lambda
##  [1,]  2 0.7570 0.0532100
##  [2,] 23 0.8201 0.0319000
##  [3,] 54 0.9024 0.0191200
##  [4,] 62 0.9452 0.0114600
##  [5,] 67 0.9682 0.0068720
##  [6,] 75 0.9813 0.0041200
##  [7,] 82 0.9889 0.0024700
##  [8,] 82 0.9934 0.0014800
##  [9,] 84 0.9960 0.0008875
## [10,] 85 0.9976 0.0005321
```

→ Two variables are always included.

What are benefits of covariate?

- By adding the covariates in a model,
  we adjust the effects of the covariates on the phenotype of interest.
- We can also adjust the effects of stratification such as race or species using the eigen vectors or the principal components

- Therefore, we can only focus on the effects of genetic variants.

Part 3. Stability selection

Stability selection

- Regularization procedures such as elastic-net outperform univariate analysis in terms of variable selection in many studies.
- However, its results depend on how to select the optimal tuning parameters.
- To address this issue, stability selection (Meinshausen and Buhlmann, 2010) was proposed.

Definition (Selection probabilities)

Algorithm Selection Probability with Elastic-net

0: Let us assume that SNP data has n samples and p variables.

A grid of tuning parameters

1: For all $\Lambda = (\alpha, \lambda)$, where $\alpha \in [0,1]$, $\lambda > 0$

2: **for** k=1 to K **do**

Subsampling ("subagging")

3: Subsample I_k with size [n/2]

4: Compute $\hat{\beta}_j^{\Lambda}(I_k)$ with regularization model

5: **end for**

Selection probabilities

6: $SP_j^{\Lambda} = \frac{1}{K}\#\{k \le K : \hat{\beta}_j^{\Lambda}(I_k) \neq 0\}$

Stability

7: $SP_j = \max_{\Lambda} SP_j^{\Lambda}$, $j = 1, \cdots, p$

8: **return** $\boldsymbol{SP} = (SP_1, \cdots, SP_p)$

Setting a grid of tuning parameters

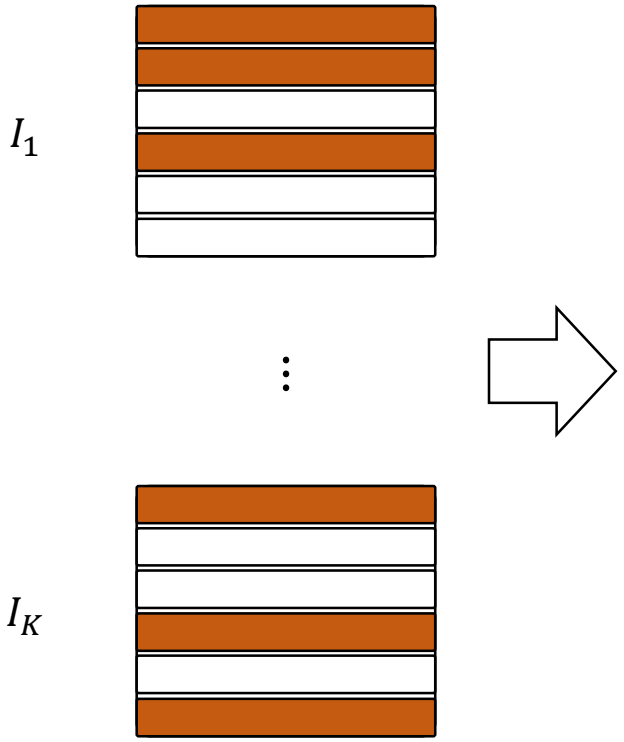
```
library(glmnet)
alpha.grid <- seq(0.1, 0.9, 0.1)
lambda.mat <- array( NA, dim = c(length(alpha.grid), 10, 10),
                 dimnames=list(paste0("alpha=", alpha.grid),
                               paste0("lambda", 1:10),
                               paste0("REP=", 1:10)) )
for( REP in 1:10 ){
  for( i in 1:length(alpha.grid) ){
    set.seed( 1000*REP + i )
    sp.subset <- sample( nrow(x), floor(nrow(x)/2) )
    lambda.mat[i,,REP] <- glmnet(x=x[sp.subset,], y=y[sp.subset], family="gaussian",
                            alpha=alpha.grid[i], nlambda=10)$lambda
  }
}
```

```
lambda.grid <- seq( summary(lambda.mat)[2], summary(lambda.mat)[4], length.out = 100 )
range(lambda.grid)
```

```
[1]  69.30751 831.11629
```

Applying regularization for the same subsamples

Subsampling



With the same subsamples,
applying regularization for all tuning parameters

	λ_1	λ_2	$\cdots$	λ_{100}
$\alpha = 0.1$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.1, \lambda_1)$ $j = 1, \cdots, p$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.1, \lambda_2)$ $j = 1, \cdots, p$		$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.1, \lambda_{10})$ $j = 1, \cdots, p$
$\alpha = 0.2$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.2, \lambda_1)$ $j = 1, \cdots, p$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.2, \lambda_2)$ $j = 1, \cdots, p$	$\cdots$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.2, \lambda_{10})$ $j = 1, \cdots, p$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha = 0.9$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.9, \lambda_1)$ $j = 1, \cdots, p$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.9, \lambda_2)$ $j = 1, \cdots, p$	$\cdots$	$\widehat{\beta}^{\Lambda}{}_j(I_k)$ $\Lambda = (0.9, \lambda_{10})$ $j = 1, \cdots, p$

Calculating the selection probabilities

Subsampling

Applying regularization

```
sp.array <- array(0, dim=c(ncol(x), length(alpha.grid), length(lambda.grid)) )
K <- 50 ## recommended to be 100 or more
for( k in 1:K ){
  cat("A current iteration is ", k, "₩n")
  set.seed(123*k)
  sp.subset <- sample( nrow(x), floor(nrow(x)/2) )

  for( i in 1:length(alpha.grid) ){
    for( j in 1:length(lambda.grid) ){
      sp.fit <- glmnet(x=x[sp.subset,], y=y[sp.subset], family="gaussian",
                       alpha=alpha.grid[i], lambda=lambda.grid[j])
      sp.array[,i,j] <- sp.array[,i,j] + as.numeric(sp.fit$beta!=0)/K
    }
  }
}
```

Selection probabilities

```
sp <- apply( sp.array, 1, max )
head( data.frame(sp = sort( sp, decreasing=TRUE ), variable = order( sp, decreasing=TRUE ) ) )
```

```
    sp variable
1 0.86    11561
2 0.78     9015
3 0.70     4287
4 0.68     2758
5 0.66     8903
6 0.66    10095
```

Why split data with 0.5 proportion

- The answer is sub-bootstrap aggregating (subagging)

- Although the accuracy with 0.5 subsampling proportion is same as with whole samples, the computational time is much cheaper.

Buhlmann and Yu (2002). ANALYZING BAGGING. The anals of statistics. Vol 30(4): 927-961

Subagging

- Subagging reduces the variance of estimator with the same accuracy as "bagging"

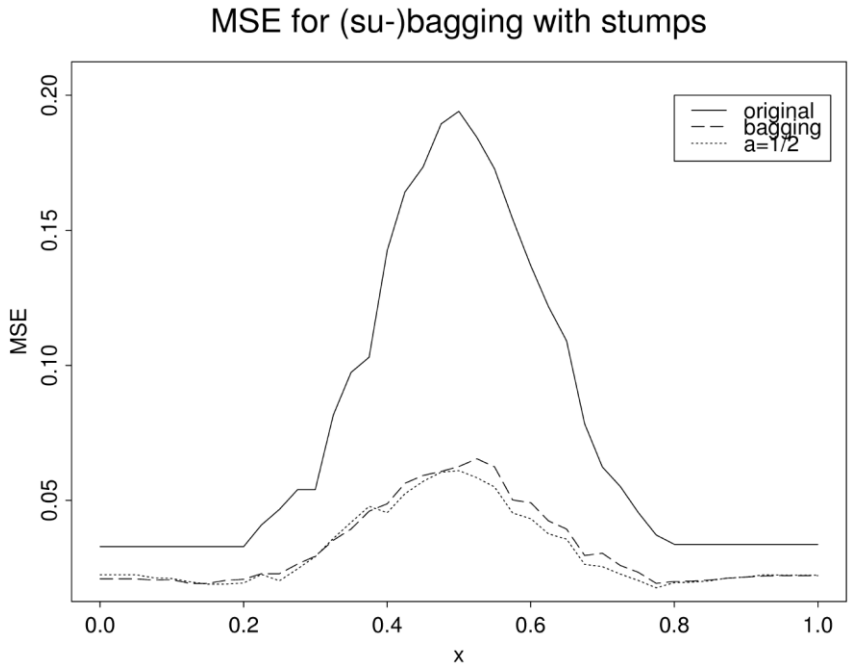


FIG. 8. *Mean squared error of stumps* $\hat{\theta}_n(x)$ *in* (3.1) (*solid line*) *and its* (*su-*)*bagged version* $\hat{\theta}_{n;SB(m)}(x)$ *for* $x \in [0, 1]$. *Sample size* $n = 100$ *and subsampling size* $m = [an]$. *Everything multiplied by the factor* $1/(\beta_u^0 - \beta_\ell^0)^2 = 1/2.25$ *to obtain* (*asymptotically*) *the scale from Figure* 7.

Buhlmann and Yu (2002). "ANALYZING BAGGING". The anals of statistics, 30(4): 927-961

Summary of calculating sel.prob

Alpha grid

10 is enough

Lambda grid

Subsampling with size $\lfloor n/2 \rfloor$

Applying regularization for each value of a grid

Selection probabilities

```
library(glmnet)
alpha.grid <- seq(0.1, 0.9, 0.1)

lambda.mat <- array( NA, dim = c(length(alpha.grid), 10, 10),
                dimnames=list(paste0("alpha=", alpha.grid), paste0("lambda", 1:10), paste0("REP=", 1:10)) )
for( REP in 1:10 ){
  for( i in 1:length(alpha.grid) ){
    set.seed( 1000*REP + i )
    sp.subset <- sample( nrow(x), floor(nrow(x)/2) )
    lambda.mat[i,,REP] <- glmnet(x=x[sp.subset,], y=y[sp.subset], family="gaussian", alpha=alpha.grid[i], nlambda=10)$lambda
  }
}

lambda.grid <- seq( summary(lambda.mat)[2], summary(lambda.mat)[4], length.out = 100 )
lambda.grid

sp.array <- array(0, dim=c(ncol(x), length(alpha.grid), length(lambda.grid)) )
K <- 50 ## recommended to be 100 or more
for( k in 1:K ){
  set.seed(123*k)
  sp.subset <- sample( nrow(x), floor(nrow(x)/2) )

  for( i in 1:length(alpha.grid) ){
    for( j in 1:length(lambda.grid) ){
      sp.fit <- glmnet(x=x[sp.subset,], y=y[sp.subset], family="gaussian", alpha=alpha.grid[i], lambda=lambda.grid[j])
      sp.array[,i,j] <- sp.array[,i,j] + as.numeric(sp.fit$beta!=0)/K
    }
  }
}

sp <- apply( sp.array, 1, max )
```

Threshold to control the false positive

- Error control with threshold π_{thr},

$$E(V) \leq \frac{1}{2\pi_{thr} - 1} \frac{q_\Lambda^2}{p},$$

where E(V) is the expected number of falsely selected variables,

q_Λ is the averaged number of selected variables for the domain Λ.

- From the above inequality, threshold can be defined as

$$\pi_{thr} = \frac{1}{E(V)} \frac{q_\Lambda^2}{2p} + \frac{1}{2}$$

Calculation of the threshold (π_{thr})

- "sp" is an array of sel.prob with (variables, alpha, lambda)

Calculation of q_Λ

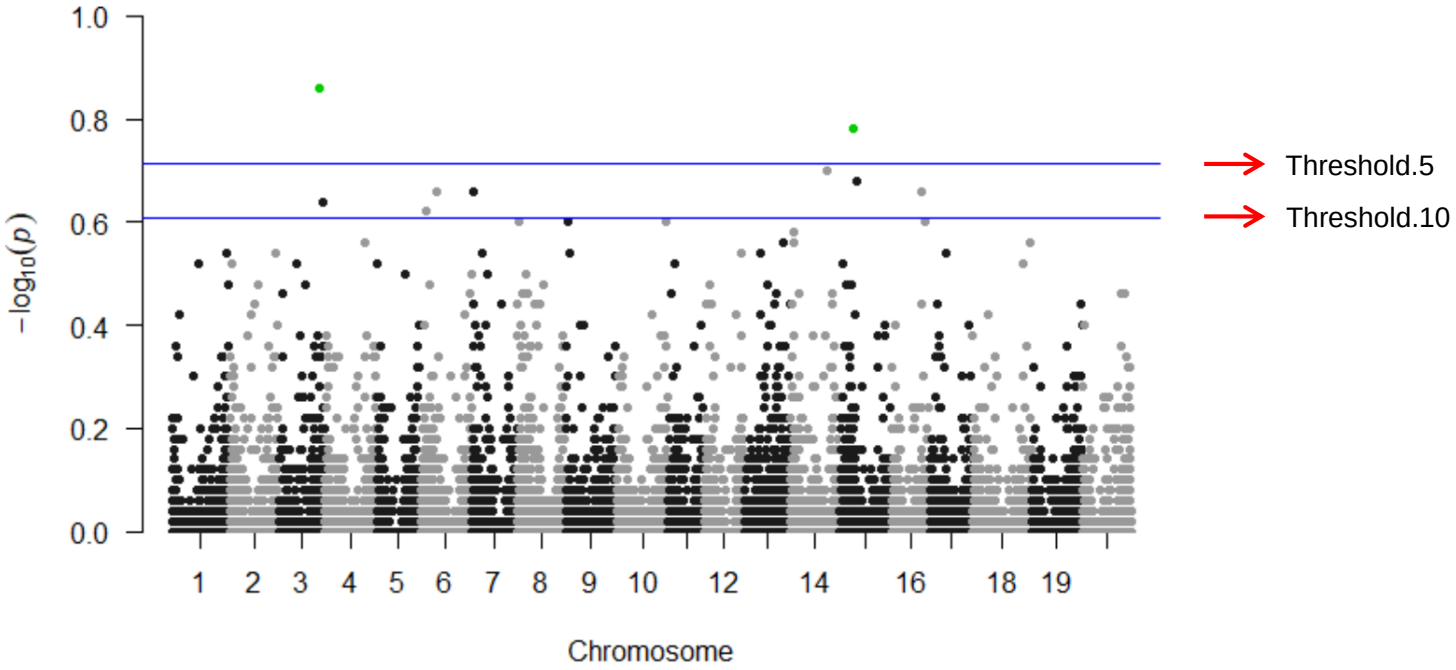
Calculation of π_{thr}

```
sp.threshold <- function(sp, FP=5){
  if(max(sp)>1 | min(sp)<0) stop("'Sel.prob' should be in [0,1].")
  if(length( dim(sp) ) < 3) stop("'sp' should be a 3-dimensional array.")
  qhat <- sum(sp)/(dim(sp)[2]*dim(sp)[3])
  p <- dim(sp)[1]
  threshold <- qhat^2/(2*FP*p) + 0.5

  if(threshold>1) threshold <- 1
  return( threshold )
}
theoretical.threshold5 <- sp.threshold(sp.array, FP=5)
theoretical.threshold10 <- sp.threshold(sp.array, FP=10)
c(theoretical.threshold5, theoretical.threshold10)
```

```
[1] 0.7129721 0.6064860
```

Manhattan plot with sel.prob ≥ π_{thr}

```
df.sp <- data.frame( ref, p=sp)
colnames(df.sp) <- c("SNP", "CHR", "BP", "P")
var.theoretical.threshold5 <- which( sp > theoretical.threshold5 )
qqman::manhattan(df.sp,
                 chr="CHR", bp="BP", p="P", snp="SNP", logp=FALSE,
                 suggestiveline=c(theoretical.threshold5, theoretical.threshold10),
                 highlight=df.sp[var.theoretical.threshold5, "SNP"],
                 ylim=c(0,1))
```



- Green dots: variables whose "sel.prob" is greater than or equal to π_{thr}.

Stability selection <R code>

Since this section only shows the stability of selection probabilities, this is not necessary in real data analysis

```
## Stability selection
stable.lambda <- glmnet(x=x, y=y, family="gaussian", alpha=0.1, nlambda=10)$lambda

stable.sp.out <- array(0, dim=c(ncol(x), length(stable.lambda)) )
K=50
for( k in 1:K ){
  if(k%%10==0) cat("A current iteration is ", k, "₩n")
  set.seed(123*k)
  sp.subset <- sample( nrow(x), floor(nrow(x)/2) )
  for( j in 1:length(stable.lambda) ){
    sp.fit <- glmnet(x=x[sp.subset,], y=y[sp.subset], family="gaussian",
                     alpha=0.1, lambda=stable.lambda[j])
    stable.sp.out[,j] <- stable.sp.out[,j] + as.numeric(sp.fit$beta!=0)/K
  }
}
# save(stable.sp.out, file="stable.sp.out.RData")
```

```
load("stable.sp.out.RData")
var.threshold10 <- which( sp > theoretical.threshold10 )
str(var.threshold10)
```

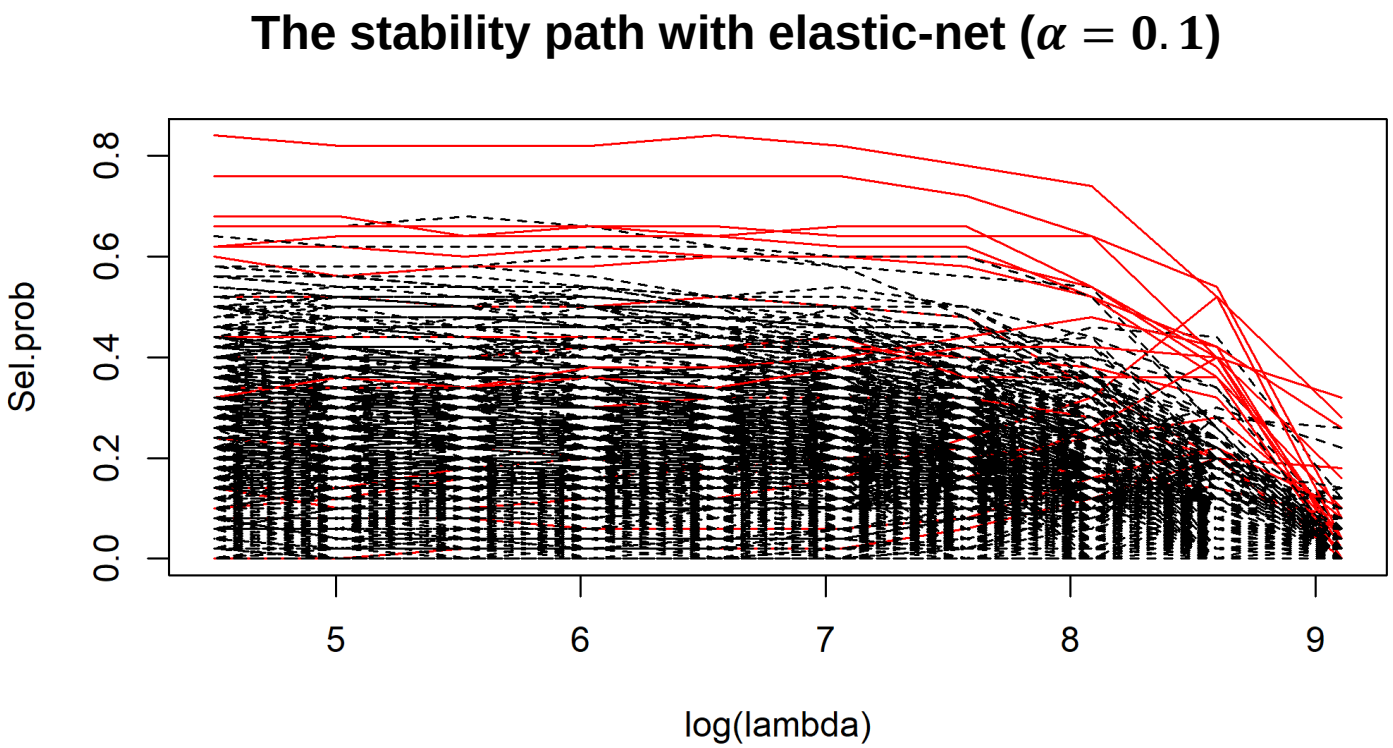
Significant genes from theoretical.threshold10

```
 int [1:9] 2758 4287 8903 9015 10095 10361 11313 11561 11985
```

```
matplot( log(stable.lambda), t(stable.sp.out), type="l",
         lty=ifelse(1:ncol(x) %in% var.threshold10, 1, 2),
         col=ifelse(1:ncol(x) %in% var.threshold10, 2, 1),
         xlab="log(lambda)", ylab="Selection probabilities" )
```

Plotting

Stability selection <figure>



The end