Session 4.
Penalised methods for genetic data analysis

Florian Privé

King's College London -- May 20, 2019

Slides: https://privefl.github.io/R-presentation/penalised-genetics.html

Introduction to penalized models

Multiple linear regression

We want to solve

$$y = \beta_0 + \beta_1 G_1 + \cdots + \beta_p G_p + \gamma_1 COV_1 + \cdots + \gamma_q COV_q + \epsilon \, .$$

Let $\beta = (\beta_0, \beta_1, \ldots, \beta_p, \gamma_1, \ldots, \gamma_q)$ and
$X = [1; G_1; \ldots; G_p; COV_1; \ldots; COV_q]$, then

$$y = X\beta + \epsilon \, .$$

This is equivalent to minimizing

$$||y - X\beta||_2^2 = ||\epsilon||_2^2 \, ,$$

whose solution is

$$\beta = (X^T X)^{-1} X^T y \, .$$

What is the problem when analyzing genotype data?

$$n < p$$

Penalization term -- L_2 regularization

Instead, we can minimize

$$||y - X\beta||_2^2 + \lambda||\beta||_2^2 \,,$$

whose solution is

$$\beta = (X^T X + \lambda I)^{-1} X^T y \,.$$

This is the L2-regularization ("**ridge**", Hoerl and Kennard, 1970); **it shrinks coefficients β towards 0**.

https://doi.org/10.1080/00401706.1970.10488634

Penalization term -- L_1 regularization

Instead, we can minimize

$$||y - X\beta||_2^2 + \lambda||\beta||_1 \ ,$$

which does not have any closed form but can be solved using iterative algorithms.

This is the L1-regularization ("**lasso**", Tibshirani, 1996); **it forces some of the coefficients to be equal to 0** and can be used as a means of variable selection, leading to sparse models.

https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Penalization term -- L_1 and L_2 regularization

Instead, we can minimize

$$||y - X\beta||_2^2 + \lambda(\alpha||\beta||_1 + (1-\alpha)||\beta||_2^2) \ ,$$

which does not have any closed form but can be solved using iterative algorithms ($0 \leq \alpha \leq 1$).

This is the L1- and L2-regularization ("**elastic-net**", Zou and Hastie, 2005); it is a compromise between the two previous penalties.

https://doi.org/10.1111/j.1467-9868.2005.00503.x

Advantages and drawbacks of penalization

Advantages

- Make it possible to solve the linear problem
- Generally prevents overfitting (because of smaller effects)

Drawback

- Add at least one hyper-parameter (λ) that needs to be chosen and another one if using the elastic-net regularization (α)

Alternative

- Select a few variables before fitting the linear model (e.g. using marginal significance/p-values); heuristic: $p = n/10$

Binary outcome (case-control)

Penalized logistic regression: minimize

$$L(\lambda, \alpha) = \underbrace{-\sum_{i=1}^{n} \left(y_i \log(z_i) + (1 - y_i) \log(1 - z_i) \right)}_{\text{Loss function}}$$

$$\underbrace{+\lambda \left((1-\alpha) \|\beta\|_2^2 + \alpha \|\beta\|_1 \right)}_{\text{Penalization}},$$

where $z_i = 1 / \left(1 + \exp\left(-(\beta_0 + X_{(i)}^T \beta)\right)\right)$, X_i denotes the genotypes and covariates (e.g. principal components) for individual i, and y_i is the disease status for individual i.

Code in practice

Data

Download data and unzip files.

I store those files in a directory called "tmp-data" here.

```
# Convert the bed/bim/fam data to the format used
# by packages {bigstatsr} and {bigsnpr}.
bigsnpr::snp_readBed("tmp-data/public-data.bed")
```

```
# Access the genotype matrix and the phenotype
data <- bigsnpr::snp_attach("tmp-data/public-data.rds")
G <- data$genotypes
X <- G[]  ## 560 MB
y <- data$fam$affection - 1
```

```
# Divide the indices in training/test sets
set.seed(1)
n <- nrow(X)
ind.train <- sample(n, 400)
ind.test <- setdiff(1:n, ind.train)
```

Multiple logistic model

```
mod <- glm(y[ind.train] ~ X[ind.train, ], family = "binomial")
```

```
Error: cannot allocate vector of size 128.4 Gb
In addition: Warning message:
glm.fit: algorithm did not converge
```

Prioritizing on marginal p-values

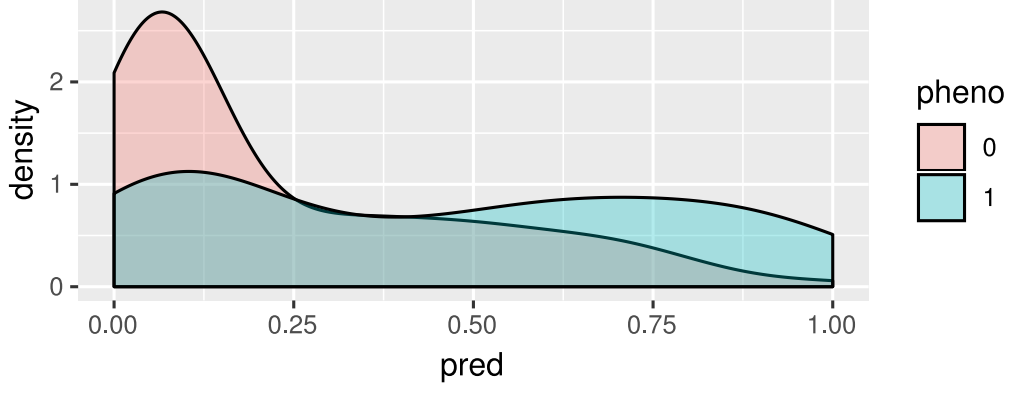
```
library(bigstatsr)
gwas <- big_univLogReg(G, y[ind.train], ind.train = ind.train,
                       ncores = nb_cores())
pval <- predict(gwas, log10 = FALSE)
ind <- order(pval)
```

Multiple logistic model after selection

```
df <- data.frame(y = y, X.sub = I(X[, ind[1:40]]))
mod <- glm(y ~ X.sub, data = df, subset = ind.train,
           family = "binomial")
pred <- predict(mod, df[ind.test, ], type = "response")
AUCBoot(pred, y[ind.test])
```

```
      Mean        2.5%       97.5%          Sd
0.68731950 0.59291869 0.77324362 0.04612472
```

```
library(ggplot2)
ggplot(data.frame(pheno = as.factor(y[ind.test]), pred = pred)) +
  geom_density(aes(pred, fill = pheno), alpha = 0.3)
```



Penalized models using {glmnet}

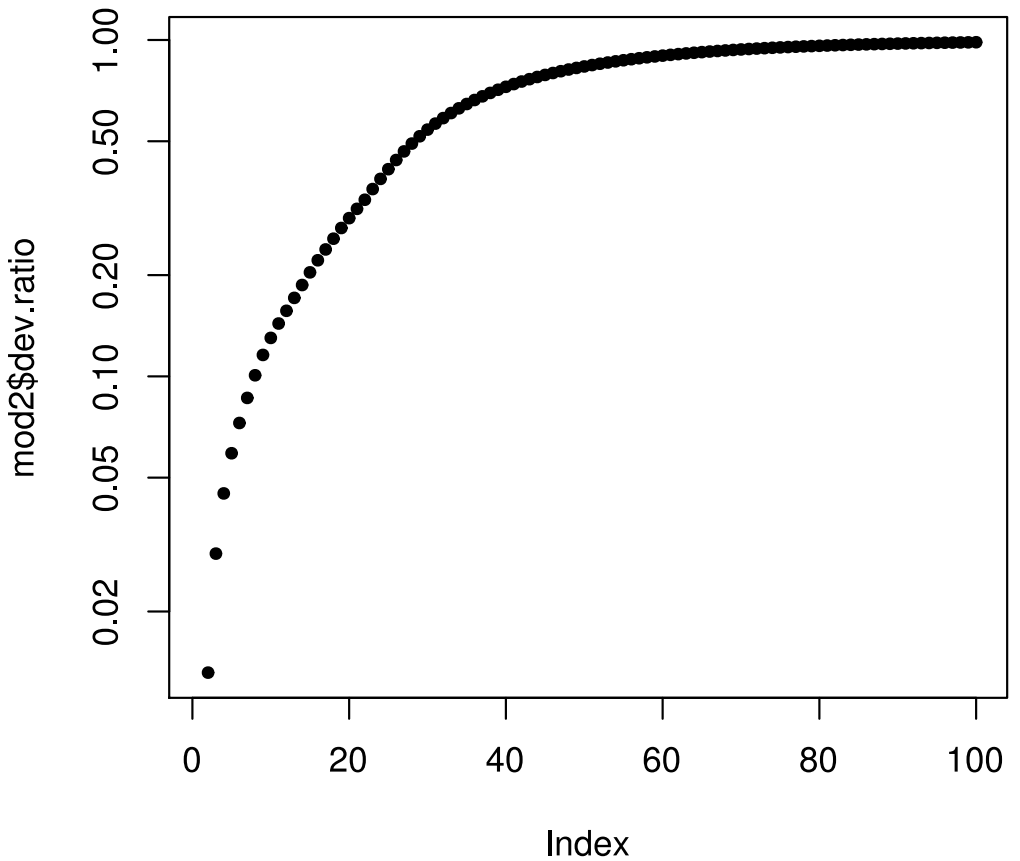
```
library(glmnet)
```

```
mod2 <- glmnet(X[ind.train, ], y[ind.train], family = "binomial")
pred2 <- predict(mod2, X[ind.test, ], type = "response")
```

```
plot(mod2$lambda[-1], apply(pred2, 2, AUC, target = y[ind.test])[-1],
     xlim = rev(range(mod2$lambda)), log = "x", pch = 20,
     xlab = expression(lambda~~(log-scale)), ylab = "AUC (on test set
```

From underfitting to overfitting

```
plot(mod2$dev.ratio, pch = 20, log = "y")
```



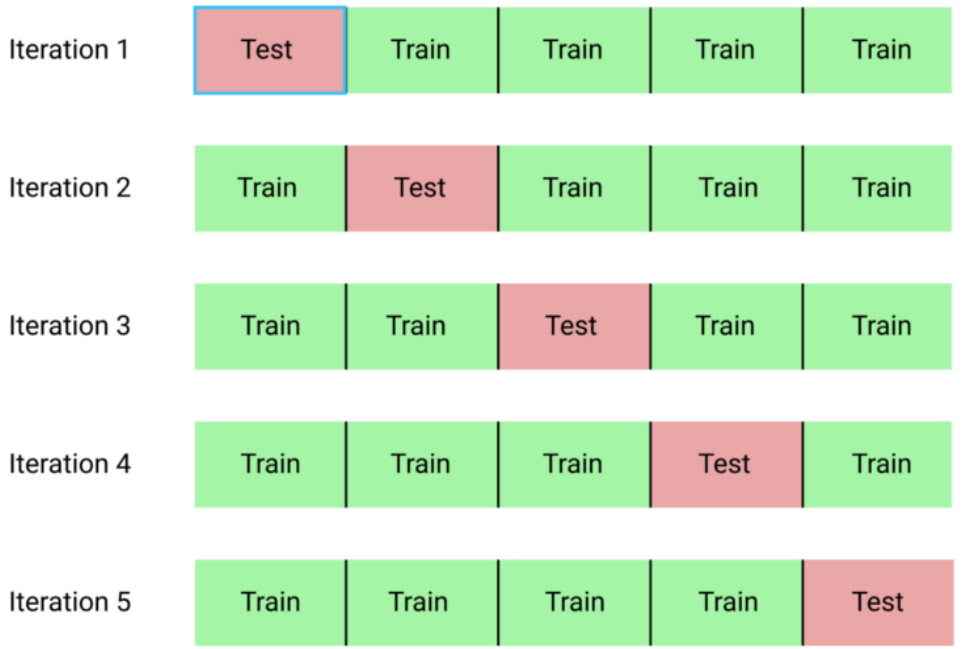
Evaluating models

Dividing in training / test sets

What is the issue with this?

What would be a better solution?

K-fold cross-validation (here, K = 5):



A possible implementation

```
set.seed(1)
K <- 5
test_grp <- sample(rep_len(1:K, n))
head(test_grp, 20)
```

```
 [1] 4 3 5 5 2 3 3 5 2 4 4 2 1 5 5 1 5 3 1 5
```

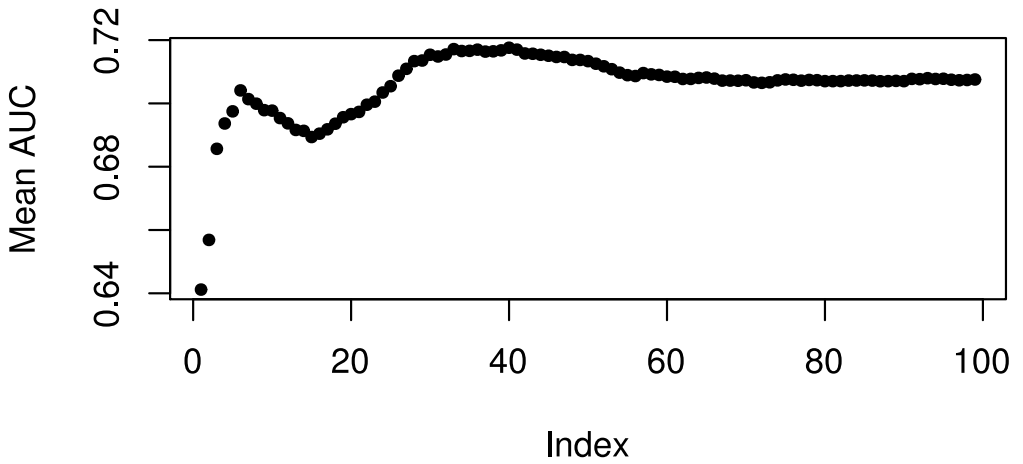
```
sapply(1:K, function(k) {
  ind.test  <- which(test_grp == k)
  ind.train <- which(test_grp != k)
  ## Replace with your preferred model
  df <- data.frame(y = y, x = X[, 1])
  mod <- glm(y ~ x, data = df, subset = ind.train, family = "binomial
  pred <- predict(mod, df[ind.test, ], type = "response")
  AUC(pred, y[ind.test])
})
```

```
[1] 0.4917044 0.4717244 0.4361781 0.5601521 0.4949495
```

Using cross-validation for parameter(s) selection

```
aucs <- sapply(1:K, function(k) {
  ind.test  <- which(test_grp == k)
  ind.train <- which(test_grp != k)
  ## Replace with your preferred model
  mod <- glmnet(X[ind.train, ], y[ind.train], family = "binomial")
  pred <- predict(mod, X[ind.test, ], type = "response")
  apply(pred, 2, AUC, target = y[ind.test])
})
```

```
plot(rowMeans(aucs)[-1], pch = 20, ylab = "Mean AUC")
```

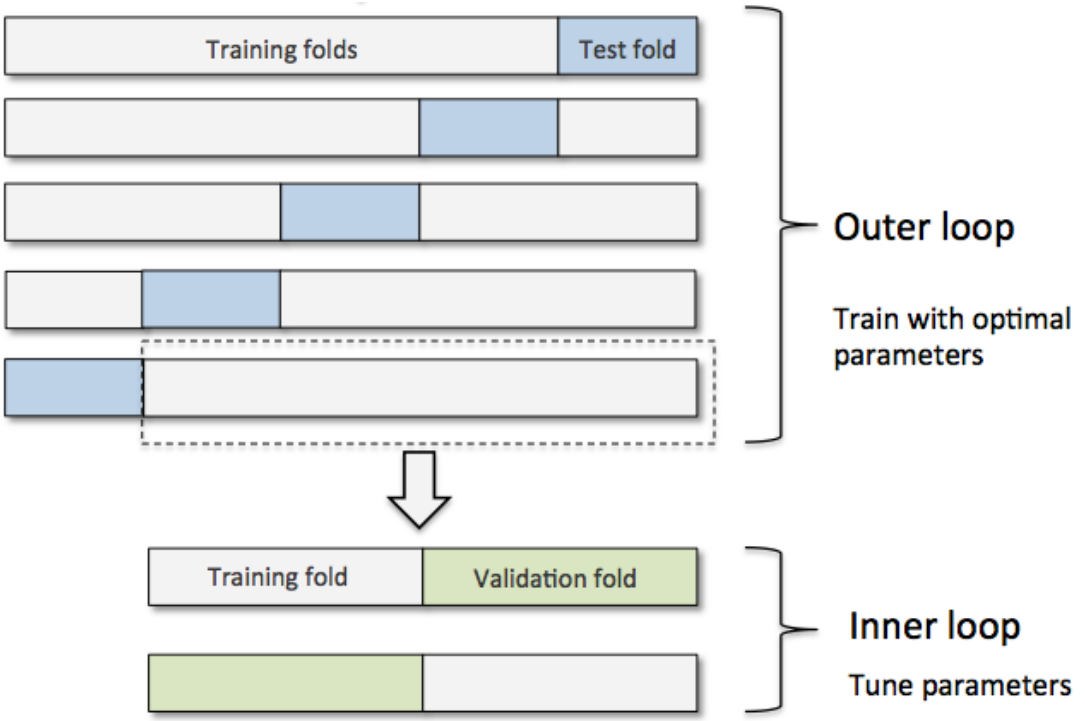


Both parameter selection and evaluation

How to do this?

We already used all data to find best parameter λ..

Nested cross-validation (here 5x2 nested CV):



Using the internal cross-validation of {glmnet}

Nested 5x3 cross-validation:

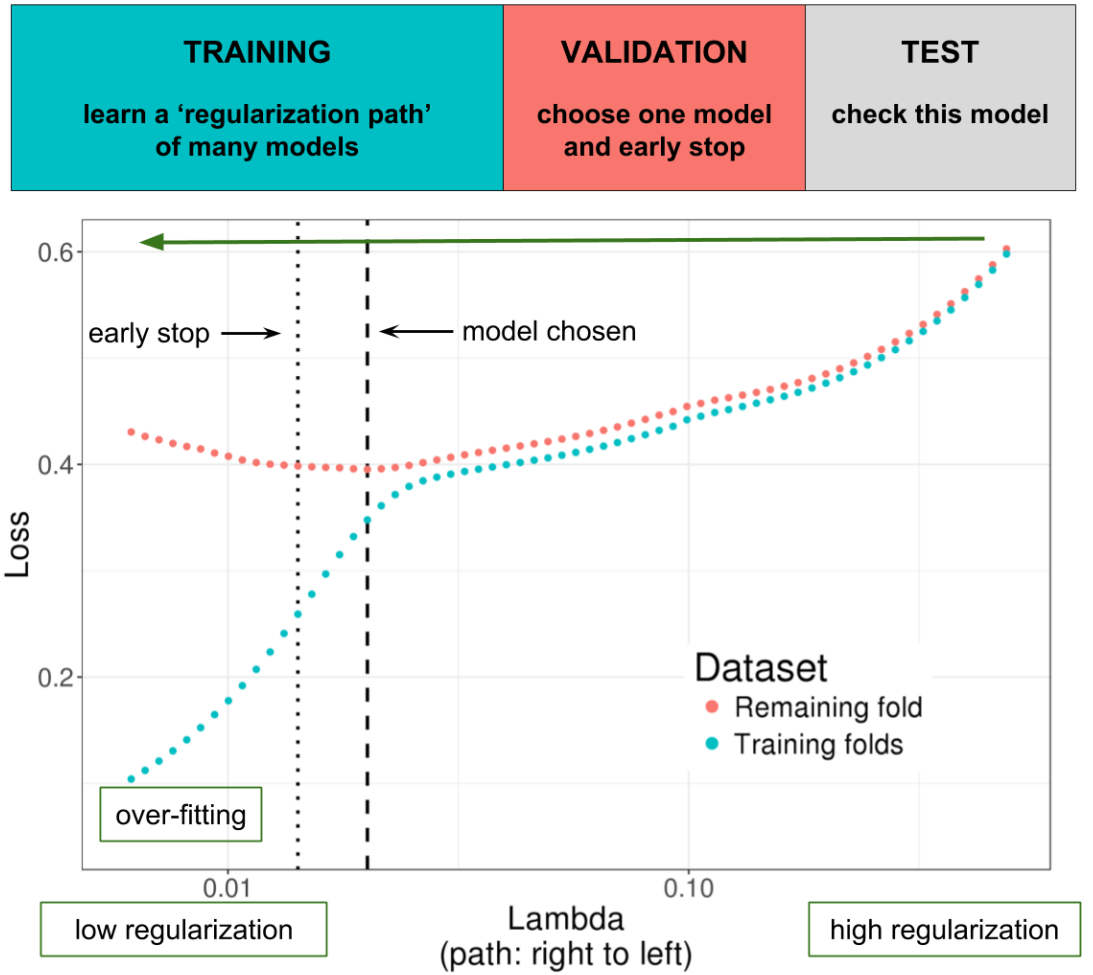
```
sapply(1:K, function(k) {
  ind.test  <- which(test_grp == k)
  ind.train <- which(test_grp != k)
  ## Penalized logistic regression with cross-validation
  mod_cv <- cv.glmnet(X[ind.train, ], y[ind.train],
                      family = "binomial", nfolds = 3)
  pred <- predict(mod_cv, X[ind.test, ], type = "response",
                  s = "lambda.1se")
  AUC(pred, y[ind.test])
})
```

```
[1] 0.6602564 0.6798088 0.7269017 0.6788909 0.7082362
```

However, it is starting to take some time to run.

A slightly different approach in {bigstatsr}

Cross-Model Selection and Averaging (CMSA)



CMSA in practice

```
sapply(1:K, function(k) {
  ind.test  <- which(test_grp == k)
  ind.train <- which(test_grp != k)
  ## Penalized logistic regression with CMSA
  mod_cmsa <- big_spLogReg(G, y[ind.train], ind.train = ind.train,
                           K = 3)
  pred <- predict(mod_cmsa, G, ind.test)
  AUC(pred, y[ind.test])
})
```

```
[1] 0.7405732 0.7009160 0.6909091 0.6900716 0.6915307
```

Advantages:

- faster (mainly because of early-stopping criterion)
- memory efficient (because data is stored on disk)

So, can be applied to huge genotype data.

Time to practice yourself

Data

Download data and unzip files.

I store those files in a directory called "tmp-data" here.

```
# Convert the bed/bim/fam data to the format used
# by packages {bigstatsr} and {bigsnpr}.
bigsnpr::snp_readBed("tmp-data/public-data.bed")
```

```
# Access the genotype matrix and the phenotype
data <- bigsnpr::snp_attach("tmp-data/public-data.rds")
G <- data$genotypes
X <- G[]  ## Access as standard R matrix of 560 MB
y <- data$fam$affection - 1
```

```
# Let us use the same folds for evaluation in cross-validation
set.seed(1)
n <- nrow(G)
K <- 5
test_grp <- sample(rep_len(1:K, n))
```

Make the best prediction of disease

You can use any model you like.

In addition to the genotype data we already used, you can use external summary statistics provided in `tmp-data/public-data-sumstats.txt`.

For example,

```
library(bigstatsr)
sapply(1:K, function(k) {
  ind.test  <- which(test_grp == k)
  ind.train <- which(test_grp != k)
  ## Penalized logistic regression with CMSA
  mod_cmsa <- big_spLogReg(G, y[ind.train], ind.train = ind.train,
                           K = 4, ncores = nb_cores())
  pred <- predict(mod_cmsa, G, ind.test)
  AUC(pred, y[ind.test])
})
```

```
[1] 0.7232278 0.6933493 0.7213358 0.7110912 0.7482517
```

Correction

Some possible solutions

Train elastic-net instead of only lasso

```
sapply(1:K, function(k) {
  ind.test  <- which(test_grp == k)
  ind.train <- which(test_grp != k)
  ## Penalized logistic regression with CMSA
  mod_cmsa <- big_spLogReg(G, y[ind.train], ind.train = ind.train,
                           K = 4, ncores = nb_cores(),
                           alphas = 10^(-(0:4)))
  pred <- predict(mod_cmsa, G, ind.test)
  AUC(pred, y[ind.test])
})
```

```
[1] 0.7586727 0.7120669 0.7187384 0.7204830 0.7288267
```

Use summary statistics to prioritize variables

```
sumstats <- bigreadr::fread2("tmp-data/public-data-sumstats.txt")
pval <- sumstats$p
conf <- pmax(-log10(pval), 1)
```

Apply a different penalization factor to each variable:

```
sapply(1:K, function(k) {
  ind.test  <- which(test_grp == k)
  ind.train <- which(test_grp != k)
  ## Penalized logistic regression with CMSA
  mod_cmsa <- big_spLogReg(G, y[ind.train], ind.train = ind.train,
                           K = 4, ncores = nb_cores(),
                           alphas = 10^(-(0:4)), pf.X = 1 / conf)
  pred <- predict(mod_cmsa, G, ind.test)
  AUC(pred, y[ind.test])
})
```

```
[1] 0.7914781 0.7224213 0.7239332 0.7218247 0.7482517
```

Thanks!

privefl privefl F. Privé

Slides created via the R package **xaringan**.