Advanced Regression: 3c Prediction accuracy and cross-validation

Verena Zuber

Epidemiology and Biostatistics, Imperial College London

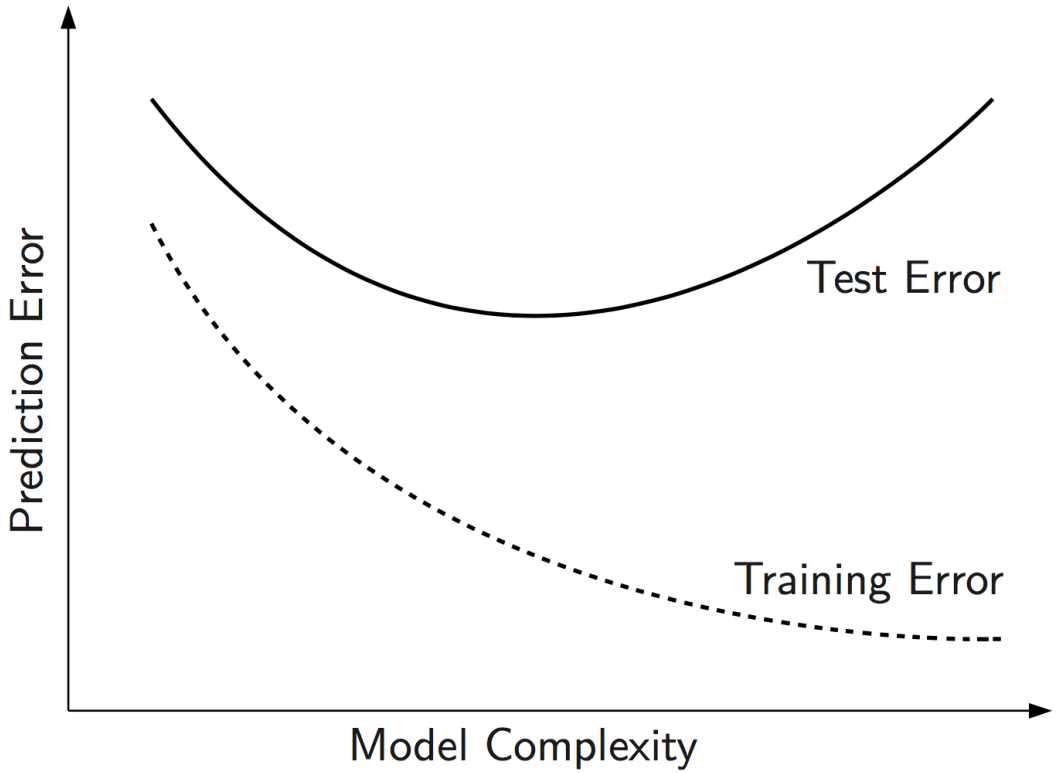
31st January 2019

Prediction and Overfitting

Cross-validation (CV)

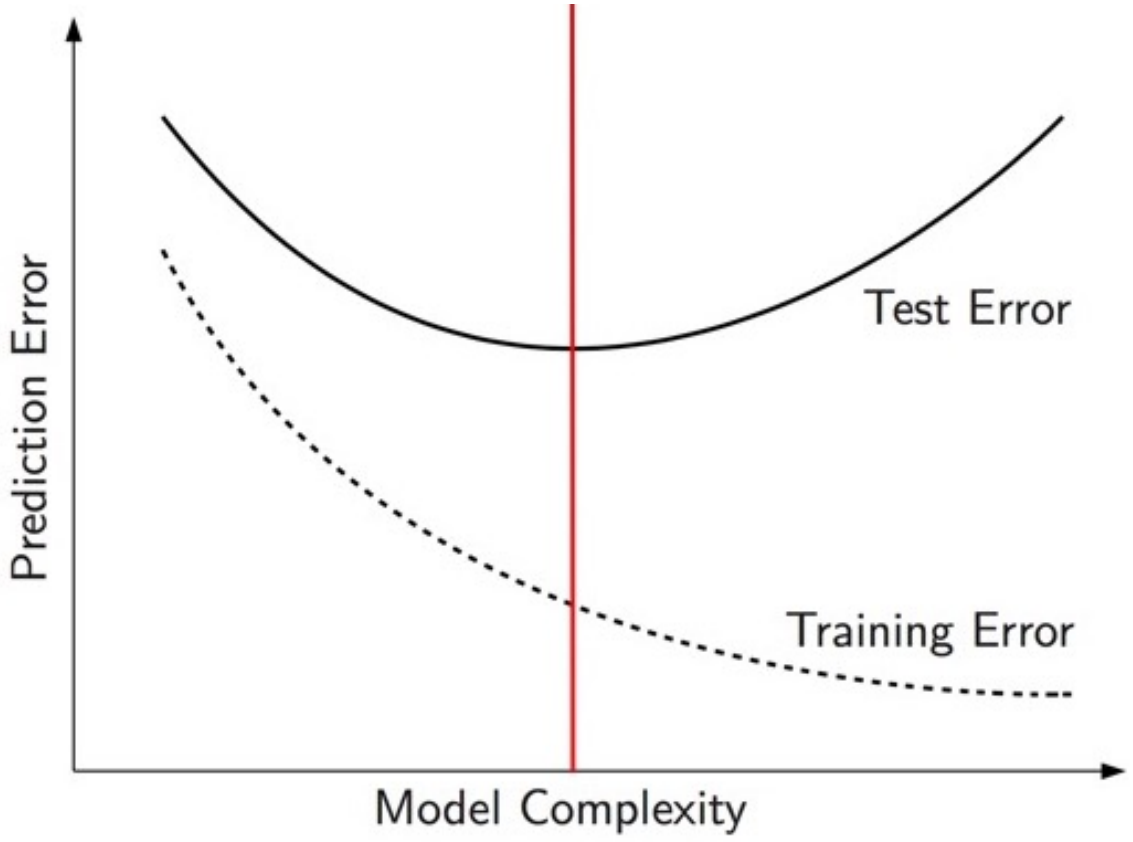
- Exhaustive cross-validation
- Non-exhaustive cross-validation
- Cross-validation to evaluate prediction performance
- Cross-validation in R: `crossval`
- Cross-validation to fix open parameters
- Cross-validation in R: `cv.glmnet`

Prediction and overfitting



- Training MSE: We can always reduce the MSE by adding more variables (higher complexity).
- Test MSE: After the model is saturated, we will increase the MSE by adding more variables.

Prediction and overfitting



- Cross-validation can be used to define the optimal model complexity for prediction.

Prediction and overfitting

When performing prediction we split the data into the following three subsets:

- **Training data** to fit the models.
- (Validation data to estimate extra parameters of the prediction rule.)
- **Test data** to assess the generalization properties.

But often we only have **one** dataset.

Cross-validation (CV)

provides a means to estimate prediction error from training data alone.

CV as a re-sampling technique

Tools that involve repeatedly drawing samples from a training set and refitting a model on each sample. In each draw we obtain more information about the fitted model.

Aims

1. To evaluate prediction rules and compare different models with respect to their predictive performance.
2. To fix open parameters and set model complexity, e.g. λ the regularisation parameter in regularised regression.

CV approaches

1. Exhaustive cross-validation
   - Leave-one-out cross-validation
   - Leave-p-out cross-validation
2. Non-exhaustive cross-validation
   - k-fold cross-validation
   - Repeated random sub-sampling validation

Leave-one-out cross-validation (LOOCV)

- Split the data containing n observations into
  1. Training data of size $n - 1$
  2. Test data of size 1
- In each split, we leave out **one** observation.
- We fit the prediction rule $\hat{f}(x)$ on the training data without observation i.
- We evaluate the MSE_i of $\hat{f}(x_i)$ on the single observation i.
- Repeat n-times for $i \in 1, ..., n$.
- Overall mean CV test error is defined as

$$CV_{\{n\}} = \frac{1}{n} \sum_{i=1}^{n} MSE_i$$

Leave-one-out cross-validation (LOOCV)

Leave-p-out cross-validation (LOOCV)

- Split the data containing n observations into
  1. Training data of size $n - p$
  2. Test data of size p
- In each split, we leave out p observations.
- We fit the prediction rule $\hat{f}(x)$ on the training data without the p observations.
- We evaluate the MSE_i of $\hat{f}(x_i)$ on the test data $i \in p$.
- Repeat for all possible combinations $comb = \binom{n}{p}$ of how to select p elements from a set of n.
- Overall mean CV test error is defined as

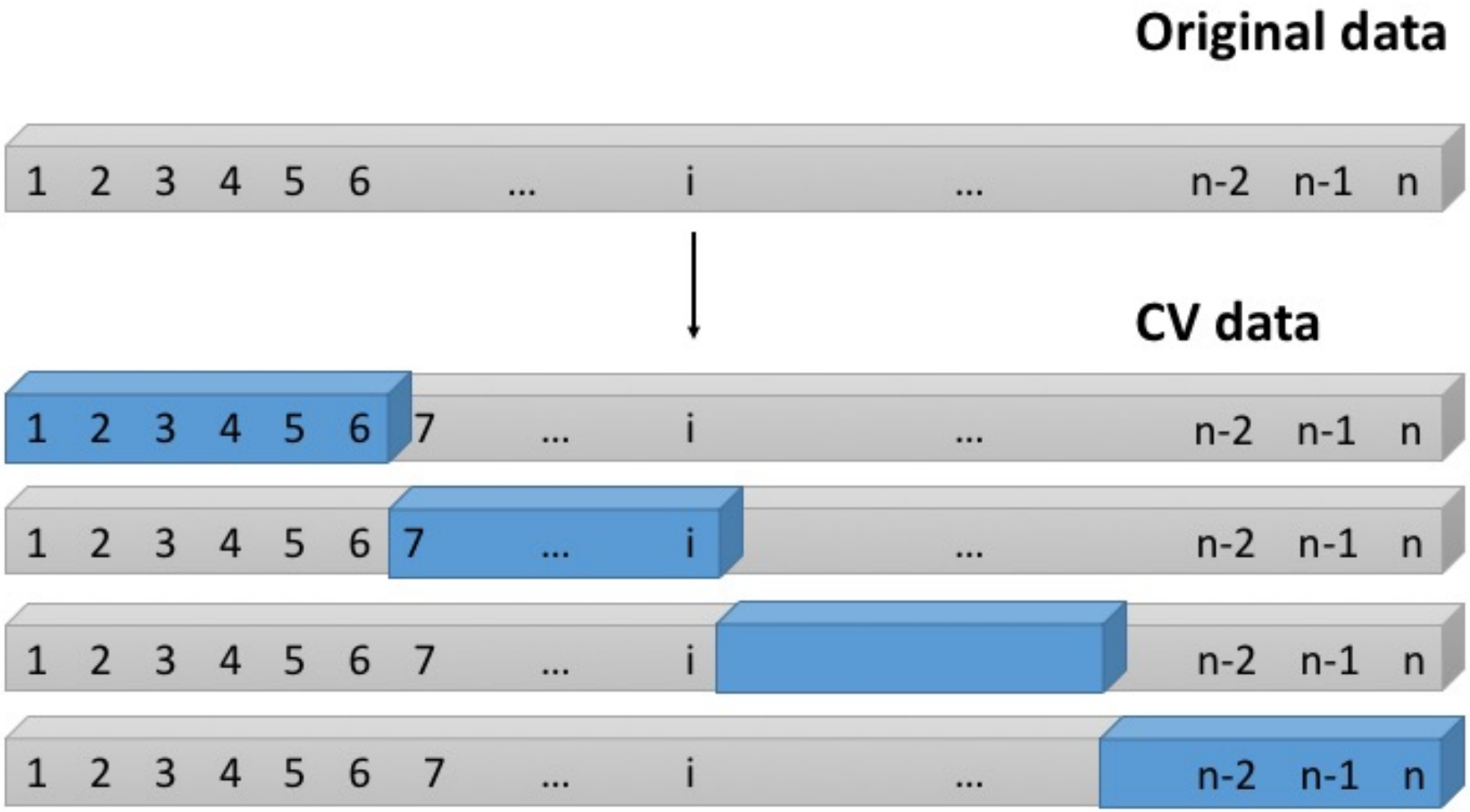
$$CV_{\{comb\}} = \frac{1}{comb} \sum_{i=1}^{comb} MSE_i$$

k-fold cross-validation

- With k-fold CV, we divide the data set into k different subsets, each of the same length.
- Recommended are $k = 5$ or $k = 10$.
- We fit the prediction rule $\hat{f}(x)$ on the training data including $k - 1$ subsets.
- We evaluate the MSE_g of $\hat{f}(x_g)$ on all observations g in subset k.
- Repeat k-times for $g \in 1, ..., k$.
- Overall CV test error rate is defined as

$$CV_{k-fold} = \frac{1}{k} \sum_{g=1}^{k} MSE_g$$

k-fold cross-validation



Repeated random sub-sampling validation

- Also known as Monte Carlo CV.
- Randomly splits the dataset into training and test data.
- Advantage: The proportion of the training/test split is not dependent on the folds.
- No guarantee that the samples are evenly distributed among training and test data, e.g. some samples might only ever be in the training data and never used to test the prediction.

Comparison LOOCV and k-fold CV

- LOOCV is a special case of k-fold CV, when $n = k$.
- LOOCV has less bias than k-fold CV.
- LOOCV has higher variance than k-fold CV.
- LOOCV is deterministic, while k-fold CV depends on the random draw of the folds.
- LOOCV (n iterations) is more computationally intensive than k-fold CV (k iterations).
- L-p-OCV ($\binom{n}{p}$ iterations) is computationally infeasible for medium sample size. For example $p = 10$ out of n has over $e + 13$ possible combinations (`R: choose(100,10)`).

Cross-validation to evaluate prediction rules

We can use CV:

- To evaluate the prediction performance of different methods, e.g. Ridge regression against Elastic Net.
- To compare different models with different predictors (not necessarily nested) and decide which model has the better prediction performance.
- To examine how well a prediction rule generalises to the population (given that our data was representative).

CV is build to evaluate prediction performance.

It does not help us to understand how the individual components of a model work.

Cross-validation in R: crossval

Example: Prediction of Diabetes disease progression

- Does the prediction improve when we add blood lipid measurements to our model?

```
> x = as.matrix(diabetes$x)
> colnames(x) = c("age","sex","bmi","map","tc","ldl","hdl",
"tch","ltg","glu")
>
> x1 = x[,c(1,2,3,4,10)]
> colnames(x1) = c("age","sex","bmi","map","glu")
>
```

Cross-validation in R: crossval

1. Write a prediction function.

```
> predfun.lm = function(train.x, train.y, test.x, test.y){
+ lm.fit = lm(train.y ~ ., data=train.x)
+ ynew = predict(lm.fit, test.x )
+
+ # compute squared error risk (MSE)
+ out = mean( (ynew - test.y)^2 )
+ return( out )
+
+ }
```

Cross-validation in R: crossval

2. Load the crossval package and perform the CV for model x (including blood lipids) and x1 (excluding blood lipids).

```
> set.seed(12345)
> cv.out = crossval(predfun.lm, x, y, K=5, verbose=FALSE)
> cv.out1 = crossval(predfun.lm, x1, y, K=5, verbose=FALSE)
```

3. Evaluate the CV test error rate for x and x1.

```
> cv.compare = rbind(c(cv.out$stat, cv.out$stat.se),
c(cv.out1$stat, cv.out1$stat.se))
> colnames(cv.compare) = c("CV","se")
> rownames(cv.compare) = c("x","x1")
> cv.compare
          CV       se
x   3015.351 38.43239
x1  3589.367 43.81240
```

4. Model x has a lower CV test error than x1. → Lipid measurements improve the prediction of disease progression.

Cross-validation to fix open parameters

- Many algorithms have open parameters.
- Example: Penalisation parameter λ in regularised regression

$$\underset{\alpha,\beta}{argmin} = RSS(\alpha, \beta) + \lambda f(\beta)$$

- It is not recommended to fix those parameter arbitrary since we might not understand what the consequences are.
- Cross-validation can be used to fix these parameters with respect to the prediction performance of a model.
- Again, cross-validation tunes the parameter to optimise prediction performance, this is not to help us understand a model.

Cross-validation to fix open parameters

- Both the MSE in LOOCV and the MSE in k-fold CV are random variables.
- The mean CV test error is an estimate for the expected CV test error.
  - LOOCV

$$CV_{\{n\}} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

  - k-fold CV

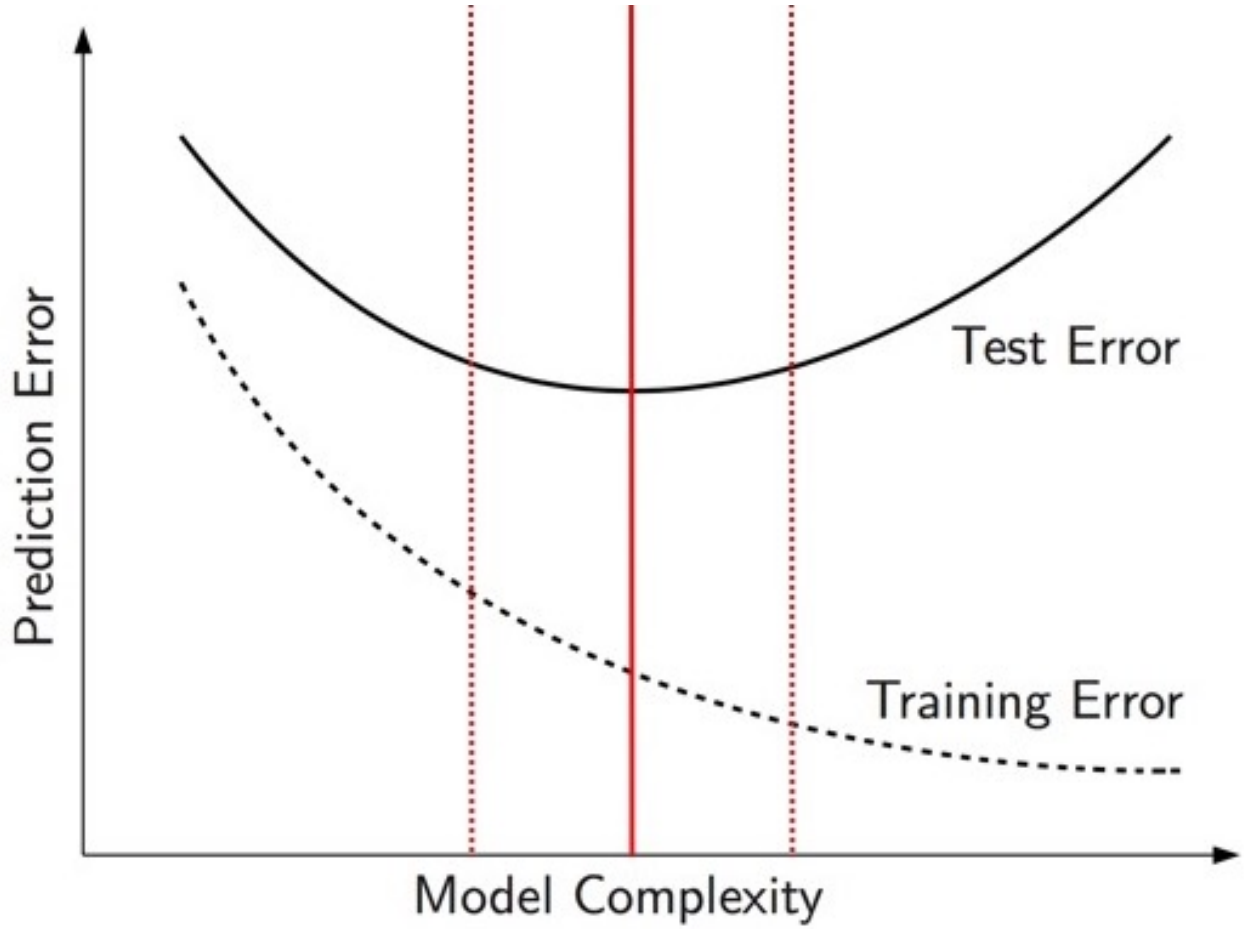
$$CV_{k-fold} = \frac{1}{k}\sum_{g=1}^{k} MSE_g$$

- Each mean CV test error has a variance and a standard error.

Occam's razor

Select the largest value of λ (smallest model) such that error is within 1 standard error of the minimum CV error.

Cross-validation to fix open parameters



Cross-validation in R: cv.glmnet

- The R package glmnet has its own inbuilt function to perform CV: cv.glmnet.
- When computing a glmnet object without pre-specified λ glmnet computes the model over a grid of lambdas and returns a matrix of regression coefficients.

```
> glmnet.all = glmnet(x,y,family="gaussian",alpha=1)
>
>
> dim(glmnet.all$beta)
[1] 10 88
> length(glmnet.all$lambda)
[1] 88
```

Cross-validation in R: cv.glmnet

- glmnet defines the optimal grid of lambda values for you.

```
> glmnet.all$lambda
 [1] 45.16003002 41.14813742 37.49265031 34.16190659 31.12705697 28.36181502
 [7] 25.84222954 23.54647710 21.45467297 19.54869896 17.81204642 16.22967331
[13] 14.78787387 13.47415991 12.27715268 11.18648427 10.19270784  9.28721577
[19]  8.46216512  7.71040969  7.02543815  6.40131759  5.83264218  5.31448632
[25]  4.84236200  4.41217991  4.02021401  3.66306928  3.33765230  3.04114447
[31]  2.77097757  2.52481156  2.30051426  2.09614292  1.90992736  1.74025468
[37]  1.58565525  1.44479000  1.31643884  1.19949004  1.09293065  0.99583771
[43]  0.90737023  0.82676196  0.75331471  0.68639230  0.62541510  0.56985495
[49]  0.51923061  0.47310359  0.43107437  0.39277891  0.35788552  0.32609195
[55]  0.29712284  0.27072727  0.24667660  0.22476253  0.20479525  0.18660180
[61]  0.17002461  0.15492010  0.14115742  0.12861739  0.11719137  0.10678041
[67]  0.09729434  0.08865098  0.08077547  0.07359960  0.06706121  0.06110368
[73]  0.05567540  0.05072935  0.04622269  0.04211640  0.03837489  0.03496577
[79]  0.03185951  0.02902920  0.02645032  0.02410055  0.02195952  0.02000870
[85]  0.01823118  0.01661157  0.01513585  0.01379122
```

Cross-validation in R: cv.glmnet

cv.glmnet performs the CV and returns two values:

1. $lambda.min: The λ with the minimum CV error
2. $lambda.1se: The λ within 1 standard error of the minimum CV error, that provides the sparsest model

```
> glmnet.cv = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
> glmnet.cv$lambda.min
[1] 0.08077547
> glmnet.cv$lambda.1se
[1] 4.842362
```

Cross-validation in R: cv.glmnet

- Note: CV is not deterministic.
- When you repeat the CV, you can see different CV mean errors and the respective optimal lambdas.

```
> glmnet.cv = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
> glmnet.cv$lambda.min
[1] 0.08077547
> glmnet.cv$lambda.1se
[1] 4.842362
>
>
> glmnet.cv = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
> glmnet.cv$lambda.min
[1] 0.9073702
> glmnet.cv$lambda.1se
[1] 7.025438
```

Cross-validation in R: cv.glmnet

- In order to synchronise the random number generator in R and reproduce your results use set.seed().

```
> set.seed(1234)
> glmnet.cv = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
> glmnet.cv$lambda.min
[1] 1.19949
> glmnet.cv$lambda.1se
[1] 5.832642
>
>
> set.seed(1234)
> glmnet.cv = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
> glmnet.cv$lambda.min
[1] 1.19949
> glmnet.cv$lambda.1se
[1] 5.832642
```

Cross-validation in R: cv.glmnet

- To extract the lasso model with cross-validated lambda, run first the CV and then extract the model with the specific lambda.

```
> set.seed(1234)
> glmnet.cv = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
> glmnet.out = glmnet(x,y,family="gaussian",alpha=1, lambda= glmnet.cv$lambda.1se)
> coef(glmnet.out)
11 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept)  152.13348
age            .
sex          -13.58817
bmi          506.75582
map          199.08174
tc             .
ldl            .
hdl         -124.20902
tch            .
ltg          441.61689
glu            .
```

- Shortcut:

```
> cvfit = glmnet::cv.glmnet(x, y, type.measure="mse")
> coef(cvfit, s = "lambda.1se")
```

Cross-validation in R: cv.glmnet

- Elastic net regression has two parameters to optimise λ_1 (lambda) and λ_2 (alpha).

```
> a = seq(0.05, 0.95, 0.05)
> search = foreach(i = a, .combine = rbind) %dopar% {
+   cv = cv.glmnet(x,y,family = "gaussian", type.measure = "mse", alpha = i)
+   data.frame(cvm = cv$cvm[cv$lambda == cv$lambda.1se], lambda.1se = cv$lambda.1se,
 alpha = i)
+ }
>
> cv = search[search$cvm == min(search$cvm), ]
> cv
      cvm lambda.1se alpha
3 3108.28   18.47318  0.15
>
>
>
> enet.out = glmnet(x, y, family = "gaussian", lambda = cv$lambda.1se, alpha = cv$al
pha)
```

Take away: Cross-validation

- Cross-validation is a powerful tool to evaluate the prediction performance of a model.
- Cross-validation is computer-intensive, but that is not a problem anymore.
- It can be used
  - To compare different models or different sets of predictor variables.
  - To compare different algorithms or methods.
  - To fix open parameters in complex methods.
- Cross-validation prevents overfitting.
- It is absolutely essential to do cross-validation when performing prediction.

Outlook: Next week

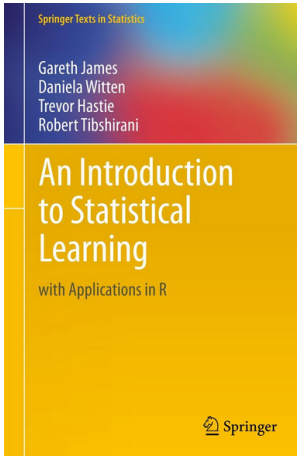
Practical: The epigentic clock

- Perform model selection using regularised regression models.
- Compare different models using cross-validation.

Lectures:

- Hierarchical models
- Non-linear models (lowess, spline, GAM)
- Non-parametric models (decision trees and random forests)

Further reading



- Chapter 5 Resampling Methods in 'An Introduction to Statistical Learning'
  Free pdf avaialable from
  `http://www-bcf.usc.edu/~gareth/ISL/index.html`
- 'A cross-validation of risk-scores for coronary heart disease mortality based on data from the Glostrup Population Studies and Framingham Heart Study' `https://academic.oup.com/ije/article/31/4/817/630270`