2.0 - Resampling

Resampling techniques facilitate inference about a population, and allow to quantify bias, variance and other statistical characteristics of interest (of an estimator, a prediction error, etc.) e.g. by:

- simulating data from a theoretic model in simulations;
- simulating copies of a real dataset;
- randomizing training and validation steps in predictive modelling.

Main methodologies:

- Monte Carlo
- Bootstrap and jackknife
- Cross-validation
- Permutation tests (not covered here)

Simulating data

- **Monte-Carlo** methods are generally used for simulation of data from a statistical model (rather than observed). Repetitions of the randomized statistical experiment are performed; the experiment outputs are collected and then analysed as outcomes of a random variable. Their statistical distribution (and therefore statistical performance of the technique) can then be characterized.
- In situations where assuming a theoretic statistical model is not appropriate or desired (e.g. when the Central Limit Theory does not apply), **Bootstrapping** is a way of simulating new samples from just one sample of observations. Statistical properties of interest can be assessed from this set of bootstrap resamples.

Validating predictive models

- When evaluating the performance of a predictive model, we are interested in measuring prediction accuracy (and related metrics) on unseen data, which are usually not available. **Cross-validation** frameworks allow to generate a number of training subsets (for model fitting) and distinct validation subsets (treated as new data). Prediction performance is then simply assessed on average on the validation samples.

2.1 - Monte-Carlo sampling

Monte-Carlo methods

Monte Carlo methods use resampling techniques to estimate the distribution of a population.

They rely on the law of large numbers, to allow us to approximate an expectation by the sample mean of a random variable (or a function thereof).

For example, risk analysis may be performed by MC repetitions of a simulated model outcome. Each outcome is generated for a different set of random values from the model distribution. The (sample) distribution of outcome values can then be used as a basis for assessing (i.e. approximating) the theoretic risk and/or functionals of it.

Monte-Carlo integration

Let our objective be to compute

$$\theta = \int_a^b g(x)dx$$

for a given function g, assuming this integral exists. Recall that if X is a r.v. with density $f(x)$, then

$$E\left[g(X)\right] = \int_{-\infty}^{\infty} g(x)f(x)dx$$

Then if a random sample is available from $F(X)$, an unbiased estimator of $E[g(X)]$ is the sample mean

$$\overline{g_M(X)} = \frac{1}{M}\sum_{i=1}^{M} g(X_i)$$

This provides an algorithm for the evaluation of integrals.

Example: we wish to evaluate

$$\theta = \int_0^1 g(x)dx$$

If $X_1, \ldots, X_M$ is a random $\mathcal{U}(0,1)$ sample, then by the strong LLN,

$$\hat{\theta} = \overline{g_M(X)} = \frac{1}{M}\sum_{i=1}^{M} g(X_i)$$

converges to $\theta = E[g(X)]$ (since $X \sim \mathcal{U}(0,1)$) with probability 1.

Generalisation to any interval: in order to evaluate

$$\theta = \int_a^b g(x)dx$$

we can perform a change of variables to fall back to the $\mathcal{U}(0,1)$ case, OR ELSE simply sample from a $\mathcal{U}(a,b)$:

$$\hat{\theta} = (b-a)\overline{g_M(X)} = \frac{(b-a)}{M}\sum_{i=1}^{M} g(X_i)$$

Exercise: evaluate e^{-x} over $(2,4)$ and check with the theoretic value.

Estimation variance and computation calibration

Variance: we have

$$\mathrm{Var}\left(\overline{g_M(X)}\right) = \frac{1}{M}\mathrm{Var}\left(g(X)\right)$$

and thus

$$\mathrm{Var}\left(\hat{\theta}\right) = (b-a)^2\mathrm{Var}\left(\overline{g_M(X)}\right) = \frac{(b-a)^2}{M}\mathrm{Var}\left(g(X)\right)$$

For M large, the Central Limit Theorem (generally) applies and one may assume that $\hat{\theta}$ is approximately Normally distributed. This is useful for deriving confidence intervals and for selection of M.

Efficiency: given two estimators, $\hat{\theta}_1$ is more efficient than $\hat{\theta}_2$ if $\mathrm{Var}(\hat{\theta}_1) < \mathrm{Var}(\hat{\theta}_2)$

Generalization of the principle

Now suppose we want to evaluate the unknown functional g of X, a r.v. with any density $f(x)$,

$$E\left[g(X)\right] = \int_{-\infty}^{\infty} g(x)f(x)dx$$

The same approach is applicable, using a random sample from the distribution of X:

$$\hat{\theta} = \frac{1}{M}\sum_{i=1}^{M} g(X_i)$$

Then with probability 1,

$$\hat{\theta} \underset{M\to\infty}{\longrightarrow} E[\hat{\theta}] = \theta$$

Note: **Importance sampling** is an extension of the principle of Monte Carlo simulations. It consists in choosing an adequate sampling distribution (from which random realizations are generated) in order to approximate the expected value with "good" control on the estimation variance.

Monte-Carlo repetitions of a statistical experiment

Suppose we need to evaluate a functional from observations

$$Y_i = \eta(\beta, X_i) + \varepsilon_i, \qquad i = 1, \dots, N$$

where:

- η and $X_i \in \mathcal{X}$ are known
- β describes a finite number of parameters to be estimated
- $\varepsilon_i \overset{iid}{\sim} f_\varepsilon(0, \sigma^2)$

The Monte Carlo principle can be applied to estimate

$$E\left[Y\right] = \eta(\beta, X)$$

Example: consider estimating β in a linear regression model

$$Y_i = \beta X_i + \varepsilon_i, \qquad i = 1, \ldots, N$$

with deterministic $X_i \in \mathcal{X}$ and $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0,1)$... Then

$$f_{Y|\beta,X} = \mathcal{N}(\beta X, 1)$$

1. Analyse a synthetic statistical problem with Monte-Carlo repetitions of the experiment...
2. This implies generating M random samples of size N for the same model
3. The statistical solution is derived for each repetition and stored in an adequate R object
4. This yields an **empirical distribution** for the estimator $\hat{\beta}$

1. Assume a true value for β, e.g. $\beta = 3$:

   $$Y_i = 3X_i + \varepsilon_i, \qquad i = 1, \ldots, N$$

   and regressors X_i, e.g. $X = (1, 2, 3, 4, 5)$
2. Set experiment parameters, e.g. $N = 50$ and $M = 100$
3. Implement repetitions of the set experiment using a loop:

```
x    = rep(c(1:5), 10);    # initialisation
ests = matrix(0, M, 1)     # initialisation
for(i in 1:M){
      y = 3*x + rnorm(N)
      ests[i] = lm(y ~ x+0)$coef[1]
}
```

4. Analyse the distribution of estimates (stored in `ests`):

```
summary(ests)
hist(ests)
boxplot(ests)
mean(ests)
sd(ests)
...
```

→ This allows comparing several estimators in terms of their distributions

→ One could for example compare the distributions of the LS estimator and of a robust alternative in a linear regression problem with non-Gaussian noise
(hint: `?MASS::rlm` and `?rexp`)

2.2 - Bootstrapping

Boostrapping

One often refers to Monte Carlo repetitions as *parametric bootstrapping*, since in that case the distribution is known and parametrized.

The Bootstrap has been introduced by Bradley Efron in the late 1970's.

Bootstrap methods are often used when the target population is not specified and the sample is the only available information to us. The distribution of the finite population represented by the sample can be regarded as a pseudo-population with similar characteristics as the true population.

Bootstrap estimates of a sampling distribution are analogous to the idea of density estimation. We construct a histogram of a sample to obtain an estimate of the shape of the density function. The histogram is not the density but can be viewed as a reasonable estimate of it.

By repeatedly generating random samples from the pseudo-population (resampling) the sampling distribution of a statistic can be estimated. Properties of an estimator such as bias, standard error etc can then be estimated.

General idea of the Bootstrap procedure

1. Create lots of new samples (Bootstrap samples) by sampling from the original random sample with replacement. All samples have same size N as original.

2. Calculate the statistic in question for each one of the Bootstrap samples.

→ The distribution of these statistics is called a Bootstrap distribution.

→ It will give information about the shape, centre, and spread of the sampling distribution of the statistic.

General algorithm

Given an initial sample $X_1, \ldots, X_N$, assuming we are interested in estimating parameter θ,

1. Repeat the following steps for $b = 1, \ldots, B$:
    1. draw a sample $X^{*(b)}$ from X with replacement
    2. evaluate the estimate $\hat{\theta}^{(b)}$ from $X^{*(b)}$
2. Deduce the bootstrap estimate of $F_{\hat{\theta}}$ as the empirical distribution of replicates $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$

Example: Bootstrap standard error estimate

Bootstrap SE estimate for estimator $\hat{\theta}$ (e.g., $\hat{\theta}_b = \bar{x}_b$):

$$SE_B = \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} \left(\hat{\theta}_b - \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}_b \right)^2}$$

Common bootstrap statistics

Given:
- an initial sample $\{X_1, \ldots, X_N\}$ with initial statistic $\hat{\theta}$,
- B bootstrap estimates $\{\hat{\theta}_b\}_{b=1}^{B}$ of parameter of interest θ
(e.g. $\hat{\theta} = \bar{X}$ and $\hat{\theta}_b = \bar{X}_b^*$):

- Bias: $B\hat{i}as(\hat{\theta}) = \frac{1}{B}\sum_{b=1}^{B}(\hat{\theta}_b - \hat{\theta}) = \bar{\hat{\theta}} - \hat{\theta}$
- Variance: $V\hat{a}r(\hat{\theta}) = \frac{1}{B}\sum_{b=1}^{B}(\hat{\theta}_b - \bar{\hat{\theta}})^2$
- MSE: $M\hat{S}E(\hat{\theta}) = B\hat{i}as(\hat{\theta})^2 + V\hat{a}r(\hat{\theta})$
- Standard error:

$$SE_B = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\theta}_b - \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_b\right)^2}$$

Accolade: International prize for the Bootstrap

“With the bootstrap [...], scientists are able to learn from limited data in a simple way that enables them to assess the uncertainty of their findings. In essence, it is possible to simulate a potentially infinite number of data sets from an original data set and–in looking at the differences–measure the uncertainty of the result from the original data analysis.”

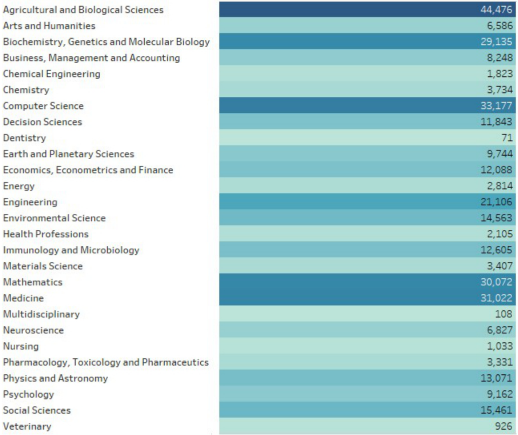
“”*While statistics offers no magic pill for quantitative scientific investigations, the bootstrap is the best statistical pain reliever ever produced*”, says Xiao-Li Meng, Whipple V. N. Jones Professor of Statistics at Harvard University. *”It has saved countless scientists and researchers the headache of finding a way to assess uncertainty in complex problems by providing a simple and practical way to do so in many seemingly hopeless situations.”*”

“”*Because the bootstrap is easy for a computer to calculate and is applicable in an exceptionally wide range of situations, the method has found use in many fields of science, technology, medicine and public affairs*”, says Sir David Cox [...].”

Accolade: International prize for the Bootstrap

The impact of the bootstrap across research fields as measured by citation

The dataset contains over 200,000 articles from over 200 journals between 1980 and 2018



Agricultural and Biological Sciences	44,476
Arts and Humanities	6,586
Biochemistry, Genetics and Molecular Biology	29,135
Business, Management and Accounting	8,248
Chemical Engineering	1,823
Chemistry	3,734
Computer Science	33,177
Decision Sciences	11,843
Dentistry	71
Earth and Planetary Sciences	9,744
Economics, Econometrics and Finance	12,088
Energy	2,814
Engineering	21,106
Environmental Science	14,563
Health Professions	2,105
Immunology and Microbiology	12,605
Materials Science	3,407
Mathematics	30,072
Medicine	31,022
Multidisciplinary	108
Neuroscience	6,827
Nursing	1,033
Pharmacology, Toxicology and Pharmaceutics	3,331
Physics and Astronomy	13,071
Psychology	9,162
Social Sciences	15,461
Veterinary	926

Courtesy of Cornell University and EPAM Systems Inc.

2.3 - Cross-validation

Cross-validation (CV)

- A single error/performance measurement is not reliable
- A model fitted (trained) to a sample will describe that sample's pattern very well, maybe even too well (overfitting)
- Cross-validation is used specifically to evaluate model performance on 'new' data
- Main CV frameworks:
  - Leave-one-out CV (LOO CV)
  - Leave-p-out CV (LpO CV)
  - k-fold CV
  - Monte Carlo CV
  - Nested k-fold CV

LOO CV (source: ISLR)

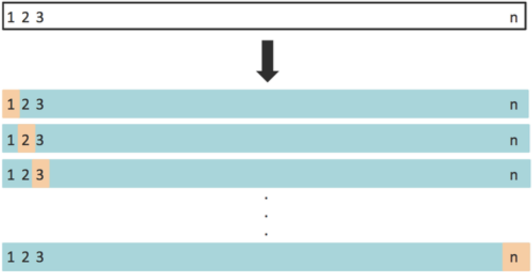


FIGURE 5.3. *A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.*

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

(figure from ISLR)

LOO CV and LpO CV

- Cross-validated error is the average of all LOO CV sample MSE's (or any other performance criterion):

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

- LOO CV uses all possible combinations of $n-1$ data points as training sets
- Leave-p-out CV generalizes this principle by using all possible combinations of $n-p$ data points as training sets
- LpO CV thus requires calculating C_p^n model fits and corresponding error estimates
- Even for moderately large n, this can quickly become infeasible when $p > 1$

k-fold CV (source: ISLR)

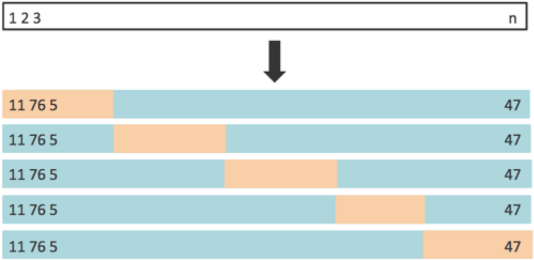


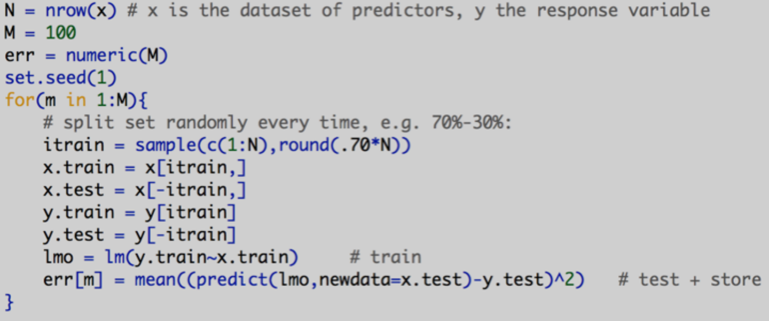
FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

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

(figure from ISLR)

Monte Carlo CV

Instead of splitting into distinct folds (as in k-fold CV), split dataset randomly at every iteration (using same training and test sample sizes every time)



```
N = nrow(x) # x is the dataset of predictors, y the response variable
M = 100
err = numeric(M)
set.seed(1)
for(m in 1:M){
    # split set randomly every time, e.g. 70%-30%:
    itrain = sample(c(1:N),round(.70*N))
    x.train = x[itrain,]
    x.test = x[-itrain,]
    y.train = y[itrain]
    y.test = y[-itrain]
    lmo = lm(y.train~x.train)      # train
    err[m] = mean((predict(lmo,newdata=x.test)-y.test)^2)    # test + store
}
```

Nested k-fold CV

- When training elaborate models, tuning of some model hyper-parameters may be required
- Ex: the degree p of a polynomial in polynomial regression

$$Y = \theta_0 + \theta_1 X + \cdots + \theta_p X^p + \varepsilon$$

- Nested CV consists in applying an inner CV loop on each CV training sample for such model tuning
- Usually k-fold CV is used for both outer and inner CV
- This will be covered more extensively in the follow-on course