Statistical Testing and Model Selection

Daniel Lawson University of Bristol

Lecture 4, Week 3, 2019-2020

Signposting

Last sesson we covered **regression**. This is something of a pre-requisite for a useful analysis of **testing**.
We'll cover testing in three sections:

1. Traditional testing (recap)
2. Resampling methods
3. Model selection

Intended Learning Outcomes

- ILO2 Be able to **use and apply basic machine learning** tools
- ILO3 Be able to make and report appropriate inferences from the results of applying basic tools to data

Specifically:

- Be able to define and use a null hypothesis significance test;
- Contrast classical and resampling tests, and judge appropriate uses;
- Be able to perform basic calculations with Leave-one-out cross validation (CV) and to make judgement calls about the appropriate use of k-fold CV.

Null hypothesis test

Given some data y about which is known x:

H0: A statement is true about y

H1: The statement is not true.

We then compute the p-value $p(y)$, the probability of observing y given that H0 is true.

Example: H0: $\mathbb{E}(y) = \mu$ with $\mu = 0$. H1: $\mu \neq 0$.

This is **not model selection**. We favour H0 and must find evidence against it to accept H1.

Null hypothesis significance testing

Hypothesis testing is asking: are my data consistent **with this hypothesis** when **using this measure**?

- If you choose a silly hypothesis, testing will dutifully say "no"
- If you use a weak measure, testing will dutifully say "yes"
- Nothing is learned by this!

The correct use of statistical testing is where:

1. the **null hypothesis might plausibly be true**, or
2. it might not be true, but you care how much **power the data has to reject the null**

When to use hypothesis testing

Some valid use cases include:

- To **rank hypotheses** by how much evidence there is against them
- To obtain a **standardized scale** (0-1) for combining evidence
- When **validating simulations**
- When **data are scarce**

Types of error

The **p-value** defines the probabilty that H0 is true, but is rejected.
The **power of the test** is the probability that H0 is false but is accepted anyway. Low power situations are to be avoided: see e.g. Andrew Gelman's blog[1].
Power is a surprisingly important problem because there are many *researcher degrees of freedom* so if power is low, we tend to find significant results anyway, through the (often unintentional) use of the data to choose the test.

[1] https://andrewgelman.com/2018/02/18/low-power-replication-crisis-learned-since-2004-1984-1964/

Types of error

Error notation

.	H0 true	H0 false
H0 accepted	Correct	Type II error
H0 rejected	Type I error	Correct

Alternative notation

.	H0 true	H0 false
H0 accepted	True Positive	False Positive
H0 rejected	False Negative	True Negative

Example: T-test for a difference in mean

To test if the mean of x is μ_0, we calculate the **statistic**:

$$t = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}},$$

where s is the standard deviation and n the sample size. Under $H0$:

$$t \sim \mathrm{t}(t; \nu = n - 1)$$

where ν is the degrees of freedom and which has the density

$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})} \left(1 + \frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

Example: T-test for a difference in mean

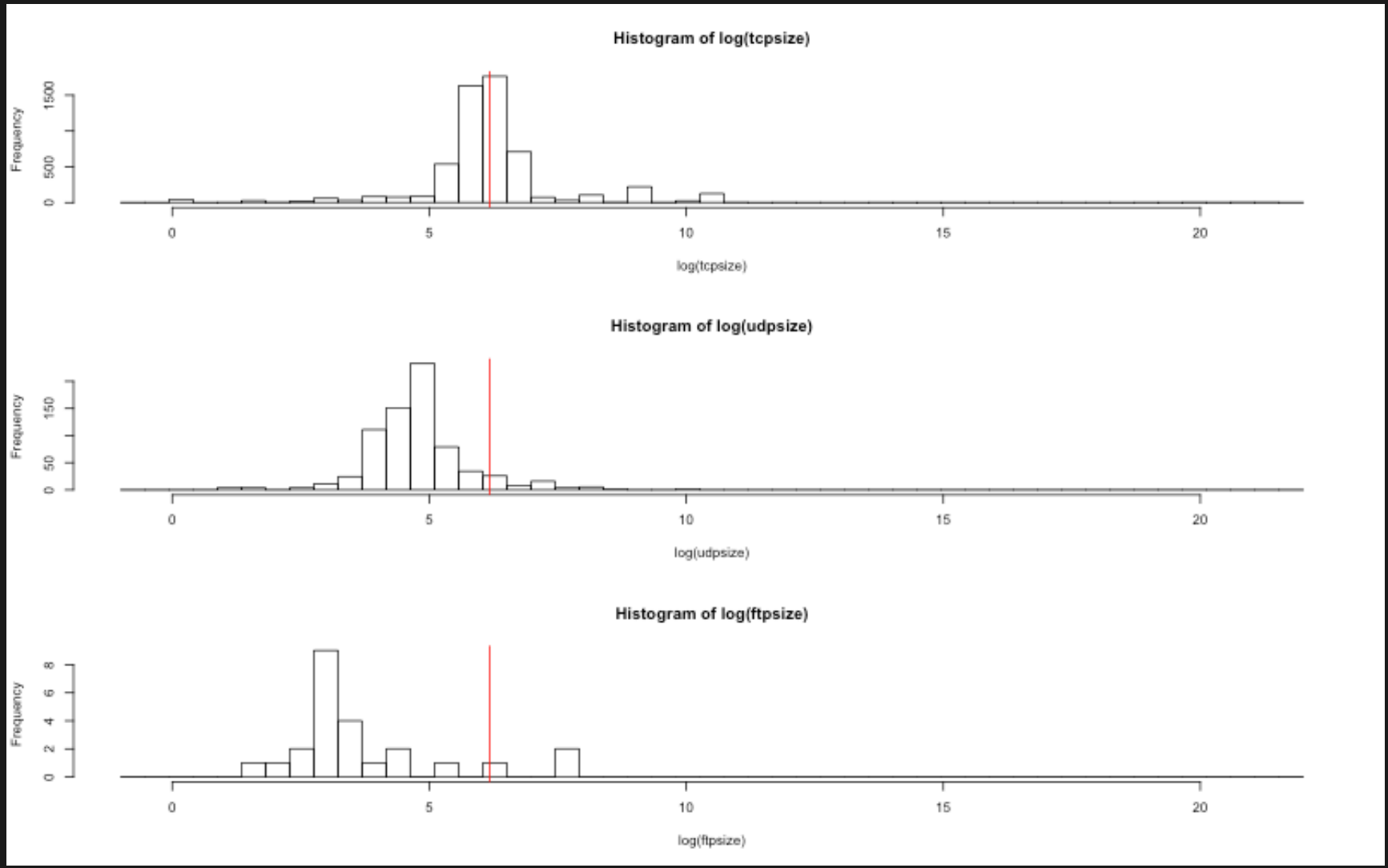
```
## Extract TCP and UDP packet sizes
tcpsize=conndata[conndata[,"proto"]=="tcp","orig_bytes"]
udpsize=conndata[conndata[,"proto"]=="udp","orig_bytes"]
ftpsize=conndata[conndata[,"service"]=="ftp","orig_bytes"]

## Convert and omit missing data
tcpsize=as.numeric(tcpsize[tcpsize!="-"])
udpsize=as.numeric(udpsize[udpsize!="-"])
ftpsize=as.numeric(ftpsize[ftpsize!="-"])
tcpsize=tcpsize[tcpsize>0]
udpsize=udpsize[udpsize>0]
ftpsize=ftpsize[ftpsize>0]
```

Example: T-test for a difference in mean

```
mu=mean(log(tcpsize))
t.test(log(udpsize),mu=mu)$p.value
t.test(log(ftpsize),mu=mu)$p.value
```

```
> t.test(log(udpsize),mu=mu)$p.value
[1] 2.733874e-182
> t.test(log(ftpsize),mu=mu)$p.value
[1] 8.334782e-08
```

Example: T-test for a difference in mean



t-tests

Can be one-tailed (H0: $\mu \leq \mu_0$) or two-tailed (H0: $\mu = \mu_0$)
Assumes that the **data are Normal** and the standard deviation is either known (t is then Normal) or estimated from the data (t is then t distributed).
Used in regression, paired tests, etc.
NB Incomplete notes as this is a prerequisite!

Chi squared test

The χ^2 test is for categorical data comparing two variables. $H0$: No relationship between the variables; $H1$: Some relationship between them. The **test statistic** for N datapoints from k classes, with x_i observations of type i, with expected value $m_i = Np_i$ where p_i is the expected probabilities, is (under the null):

$$X^2 = \sum_{i=1}^{k} \frac{(x_i - m_i)^2}{m_i} \sim \chi^2(k-1)$$

This is most often used for **contingency tables** though appears elsewhere. See also **Fishers exact test** for small samples.
NB Incomplete notes as this is a prerequisite!

Other important tests

Nonparametric tests:

- **Mann-Whitney U or Wilcoxon rank sum** test: are two samples are drawn from the same distribution? by comparing their ranks.
- **Wilcoxon signed-rank** test - as rank sum test, for paired data.
- **Kolmogorov-Smirnov** test - are two samples from the same distribution? by comparing the empirical cumulative distribution function.

There are many online cookbooks which state exactly which circumstances each test should be used in. You should be able to use them.
NB Incomplete notes as this is a prerequisite!

Statistical testing overview

The tests we have discussed are **classic statistics**, that is, before computers (indeed pre-1950s). The most important types of test for data science are yet to come.

Resampling

The main types of resampling tests include:

- **jacknifing**, which is analysing subsets of data to estimate (variance of) parameter estimates
- **bootstrapping**, which is resampling with replacement, to estimate (variance of) parameter estimates
- **permutation**, which is resampling without replacement, to test a null hypothesis
- **cross-validation**, which is analysing subsets of data to estimate out-of-sample prediction, for model performance

Each of these methods can be applied to a wide variety of problems, and often requires thought to use appropriately.

Permutations

All permutations of three colors (each column is a permutation):

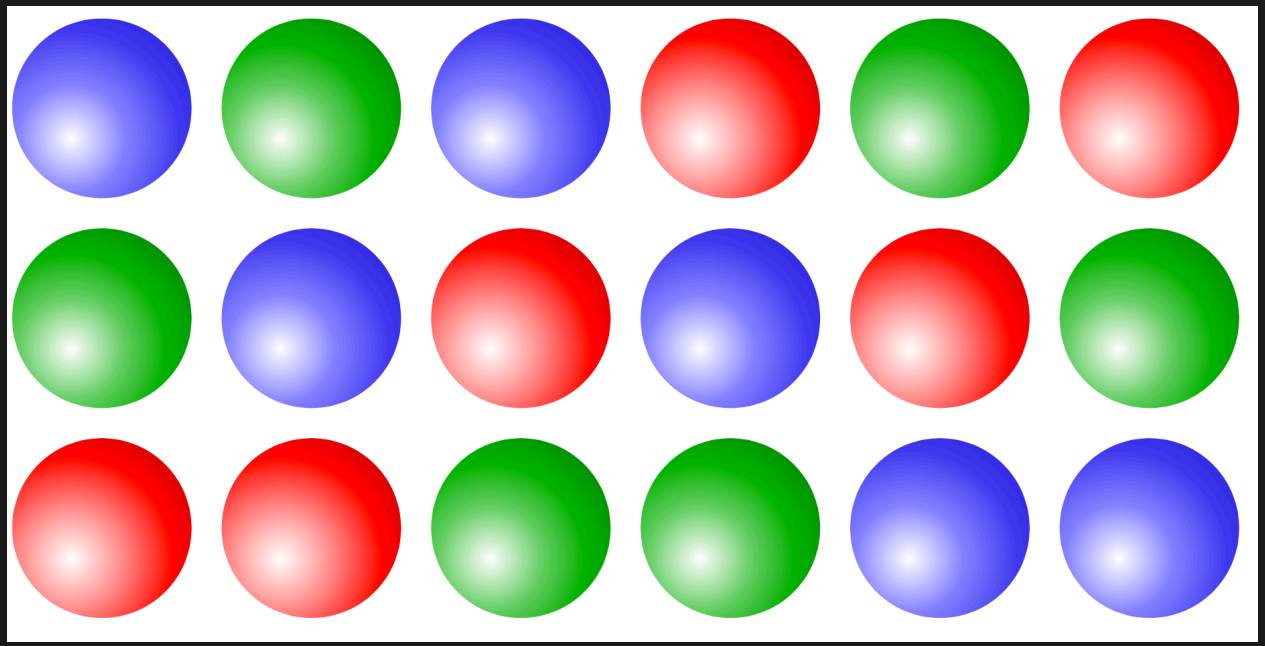


Figure from Wikipedia[2]. There are in general $n!$ permutations.

[2] https://upload.wikimedia.org/wikipedia/commons/4/4c/Permutations_RGB.svg

Generating permutations

```
> set.seed(1)
> n = 5
> x = seq(0,20,length=n)
> x
[1]  0  5 10 15 20
> x[sample.int(n)]
[1]  5 20 15 10  0
> x[sample.int(n)]
[1] 20 15  5 10  0
```

Use of permutations in testing

Consider the following general class of problem:
H0: y is independent of x.
H1: y is dependent on x.
x may be continuous, categorical, etc and y may depend on a number of other things. A **permutation test** says: lets resample x, y pairs **under H0** and see if a test statistic T is extreme in the real data, compared to the permutations.

Why permutations

The main advantage is that the test is asymptotically correct and distribution free. We only have to assume **exchangability**.
Exchangability of what?

- what would be **equal if the null hypothesis is true**, and
- would be **differerent if the alternative hypothesis is true**?

It is essential to **maintain any true correlation structure** when performing the test, otherwise the test is not correct. For example, if the indices were originally correlated as from e.g. a time-series, permutation will fail.

Some main types of test

- Permutation of indices
- Permutation of signs, retaining magnitudes
- Permutation of groups
- Permutation within groups

Monte-Carlo testing

There are in general $n!$ permutations. This is typically too many for $n > 20$.

We instead choose N **random permutations** from all the possible ones.

Monte-Carlo testing is an important subject in its own right. Its often possible to place guarantees on the p value from very few samples.

Monte-Carlo test

To conduct a Monte-Carlo test, we construct N random datasets and add our real dataset. We then ask, is our real dataset strange compared to the random datasets?
Specifically, the p-value for a test T applied to X (where large values are considered strange) is:

$$\frac{\mathrm{Rank}(T(X); T(\{x_i\}))}{N+1}$$

where Rank simply counts the number of cases as large or larger.

Heuristics for how many permutations to use

- The **smallest possible p-value** with N permuations is $1/(N+1)$. So 999 permutations gives a minimum of 0.001.
- The **variance** around a chosen threshold, say $p = 0.05$, is determined by the sampling distribution of the Binomial:

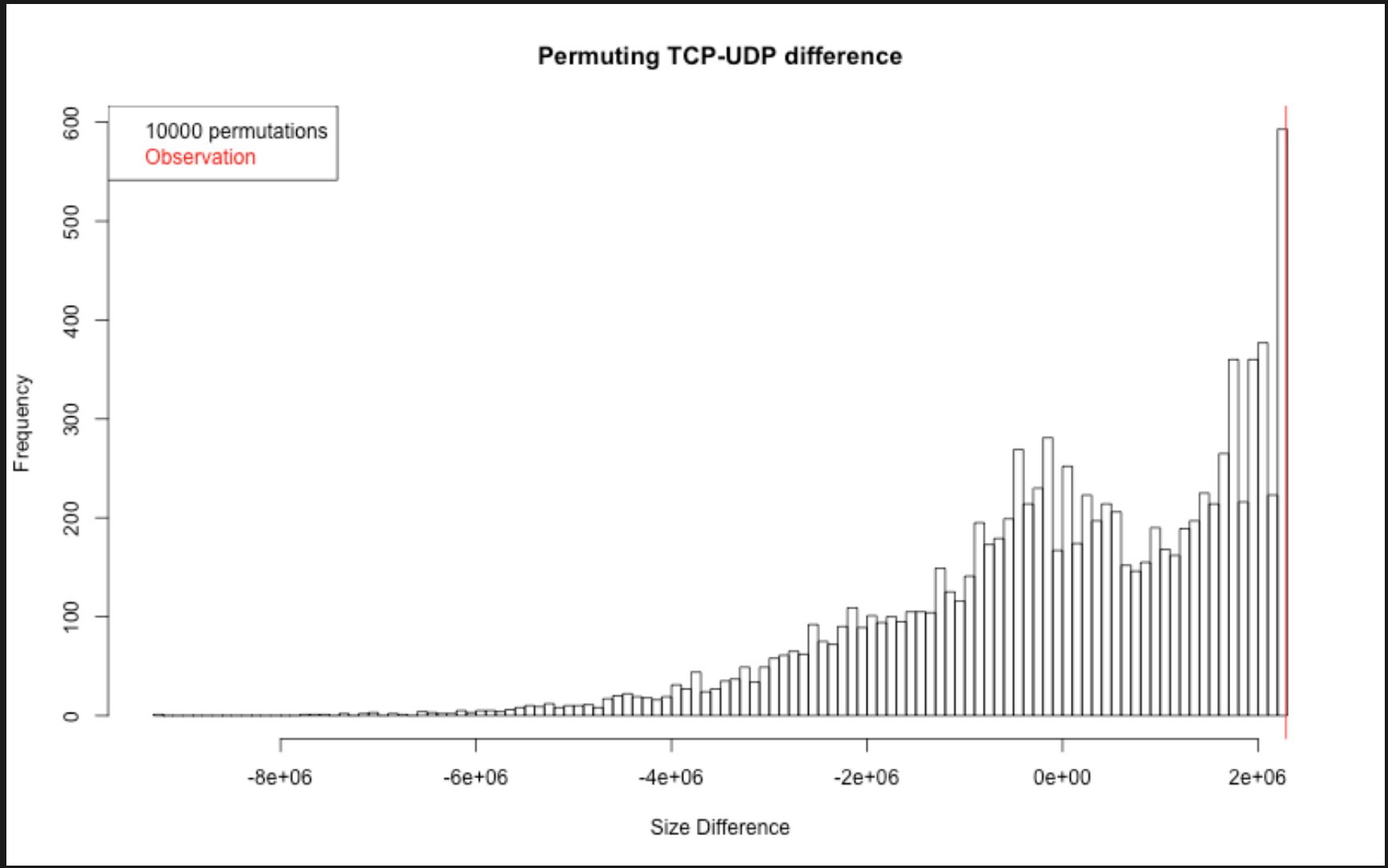
$$\mathrm{sd}(p) = \mathrm{sd}\left(\mathrm{Bin}(N, p)\right) = \sqrt{\frac{p(1-p)}{n}}$$

- So to be 95% confident that $p \leq t$ we need the empirical p-value to be less that $t - 1.96\mathrm{sd}(p = t)$
- For $N = 999$ and $t = 0.05$, $\mathrm{sd}(p = t) = 0.0135$ and therefore $p < 0.036$
- A similar calculation shows $N = 999$ wouldn't be enough to be sure we were less than 0.005.
- This is a heuristic. The distribution may not be Normal. Plot it. Often the empirical value is much less than the permutations can generate, or located in a mode.

Permutation example: TCP vs UDP size

```
tcpudp=c(tcpsize,udpsize)
n1=length(tcpsize)
n2=length(udpsize)
myteststatistic=function(x,n1,n2){
    mean(x[1:n1]) - mean(x[n1+(1:n2)])}
tobs=myteststatistic(tcpudp,n1,n2)
trep=sapply(1:10000,function(i){
    xrep=sample(tcpudp)
    myteststatistic(xrep,n1,n2)
})
mean(tobs<=trep)
# 0
```

Permutation example: TCP vs UDP size



Permutation example: FTP vs UDP size

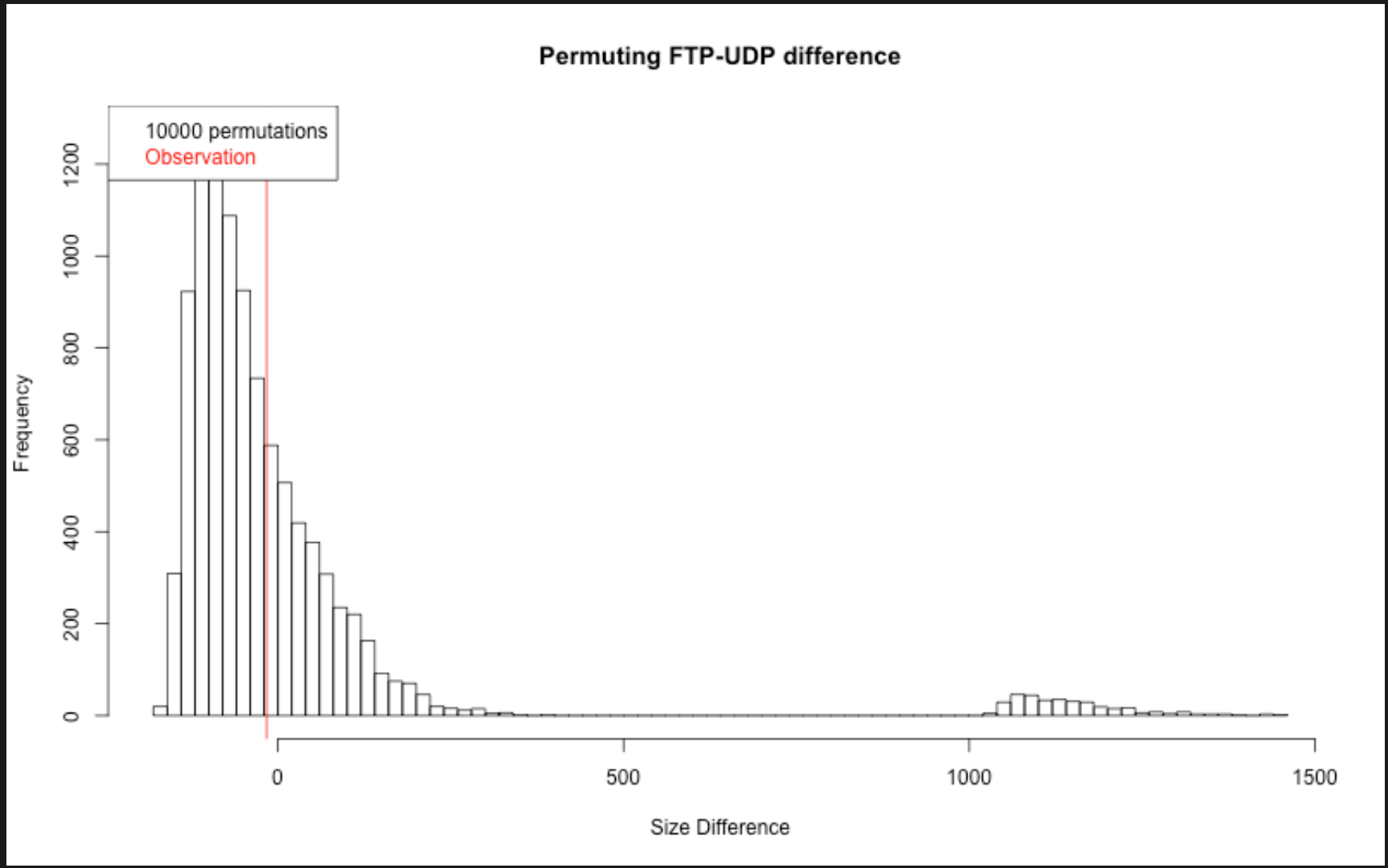
T-test suggests that FTP and UDP are different sizes

```
muudp=mean(log(udpsize))
t.test(log(ftpsize),mu=muudp)$p.value
## 0.003375621
```

Permutation example: FTP vs UDP size

```
ftpudp=c(ftpsize,udpsize)
n1=length(ftpsize)
n2=length(udpsize)
ftpudpobs=myteststatistic(ftpudp,n1,n2)
ftpudprep=sapply(1:10000,function(i){
    xrep=sample(ftpudp)
    myteststatistic(xrep,n1,n2)
})
mean(ftpudpobs<=ftpudprep)
## 0.3403
```

Permutation example: FTP vs UDP size



Permutation testing summary

- **Distributional assumptions** are often invalid (regular tests)
- **Exchangability assumptions** are often plausible (permutation tests)
- It is possible to get misleading inference if the assumptions of a test don't hold
- Permutation tests are really important for generating **plausible null hypotheses**, especially in cyber security

Model Selection

Imagine that we have run two different inference procedures (models) on our data, and we want to decide which of these gives the **best** description of the data.
(For the moment we will pretend we want to know which one is **right**…)
Model selection formalises how to make this assessment.

Residuals

The **residual sum of squares** for n predictions of a univariate Y with mean $\hat{\mu}$:

$$R^2 = \sum_{i=1}^{n} (Y_i - \hat{\mu})^2$$

The expected value of the prediction error $e^2 = R^2/n$.
What happens if **compare two models** M_1 and M_2, where M_1 is a subset of M_2?

Linear Models - Model selection

For illustration, consider

$$Y = \mathbf{x}_1 A_1 + \epsilon_1$$

and

$$Y = \mathbf{x}_1 A_1 + \mathbf{x}_2 A_2 + \epsilon_2.$$

Unless $\mathbf{x}_2 = 0$ or $\mathbf{x}_2 \equiv \mathbf{x}_1$, then ϵ_2^2 will be smaller than ϵ_1^2. This is an example of a more general rule: **larger models always have better predictions**.

So prediction error is OK to use to fit models with the same dimension, but it is of course useless for **model selection**.

Exercise: Show that $\mathbb{E}(\epsilon_2^2) < \mathbb{E}(\epsilon_1^2)$.

Cross-Validation Motivation

Usually we are not interested in properties of **our sample**. We instead wish to know how our inference will generalise to **new samples**.
The most straight forward way to predict how a model generalises is to test in **held-out data**. Cross Validation is a procedure to leave-out some data for testing.
How much data?

- **Leave-one-out Cross-Validation** (LOOCV) leaves out one datapoint at a time for testing.
- **k-Fold Cross Validation** (k-fold CV) keeps a fraction $(k-1)/k$ of the data for learning parameters and $1/k$ for testing.

General considerations

To make Cross-Validation work, we need to be able to define our inference goal cleanly. Some scenarios:

- **Same source, single datapoint**: Within a single datastream, how well can we predict the **next** point?
- **Same source, segment of data**: Within a single datastream, how well could we predict everything that happens within an hour?
- **New but understood source**: We have multiple datastreams, each of which might be different but all are generated by a similar process. How well can we predict a new such datasource?
- **Unexpected source**: We have many classes of datastream. How well can we predict what would happen on a new class of datastream?

Prediction accuracy in linear regression

In linear regression, the errors are

$$e = y - AX = y - \mathrm{H}y = y - \hat{y}$$

The expected MSE for the i-th datapoint is

$$\begin{aligned}
\mathbb{E}(e_i^2) = & \quad \mathbb{E}\left[(y_i - \hat{y}_i)^T (y_i - \hat{y}_i)\right] && (1)\\
= & \quad \mathrm{Var}[y_i - \hat{y}_i] + [\mathbb{E}(y_i - \hat{y}_i)]^2 && (2)\\
= & \quad \mathrm{Var}[y_i] + \mathrm{Var}[\hat{y}_i] - 2\mathrm{Cov}[y_i, \hat{y}_i] + [\mathbb{E}(y_i) - \mathbb{E}(\hat{y}_i)]^2 && (3)
\end{aligned}$$

Out-of-sample prediction accuracy in linear regression

We can write the same thing when predicting an **out-of-sample** y'_i:

$$\mathbb{E}({e'}_i^2) = \mathbb{E}\left[(y'_i - \hat{y}_i)^T (y'_i - \hat{y}_i)\right] \quad (4)$$

$$= \mathrm{Var}[y'_i] + \mathrm{Var}[\hat{y}_i] - 2\mathrm{Cov}[y'_i, \hat{y}_i] + [\mathbb{E}(y'_i) - \mathbb{E}(\hat{y}_i)]^2 \quad (5)$$

y'_i and y_i are independent with the same distribution (same expectation and variance). However, $\mathrm{Cov}[y'_i, \hat{y}_i] = 0$ wheras $\mathrm{Cov}[y_i, \hat{y}_i] \neq 0$.
Therefore:

$$\mathbb{E}({e'}_i^2) = \mathbb{E}(e_i^2) + 2\mathrm{Cov}[y_i, \hat{y}_i]$$

Quantifying Out-of-sample prediction accuracy

Fortunately we already did the work required to describe this:

$$\mathrm{Cov}[y_i, \hat{y}_i] = \sigma^2 \mathrm{H}_{ii}$$

The mean out-of-sample prediction error is

$$n^{-1} \sum_{i=1}^{n} e_i'^2 = n^{-1} \sum_{i=1}^{n} e_i^2 + 2n^{-1} \mathrm{tr}(\mathrm{H})$$

We know $\mathrm{tr}(\mathrm{H}) = \sigma^2 p$ where p=number of predictors. (Exercise: show this). So we can write the **optimism** as $2n^{-1}\sigma^2 p$.
The optimism grows with σ^2 and p but shrinks with n. It is used to define the **model selection criteria** ΔC_p which is minimised:

$$\Delta C_p = MSE_1 - MSE_2 + \frac{2}{n}\hat{\sigma}^2(p_1 - p_2)$$

Linear model optimism and AIC

Minimising **Akaike's Information Criterion**:

$$AIC = -2\mathbb{L}(\hat{\theta}) + 2\mathrm{Dim}(\theta)$$

reduces to the ΔC_p method when the Likelihood $\mathbb{L}$ is a Normal distribution.
Exercise: Show this.

LOOCV

We write a statistic $\hat{s}$ based on all data $\{y\}$ except i as $\hat{s}^{(-i)}$ and the data is $\{y\}^{(-i)}$. For a general **loss function** we can write:

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}\left(y_i; \hat{\theta}|y^{(-i)}\right)$$

i.e. we evaluate the loss function for each datapoint using the estimate from the remaining data.

NB A loss function is something that we choose the parameters θ to minimise. It can be the MSE, the (negative log) likelihood, a penalised version of these, or any other convenient quantity.

LOOCV for linear models

For the MSE of a linear model we can write:

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i^{(-i)}\right)^2$$

It is bookwork (Exercise!) to show that

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - H_{ii}}\right)^2$$

i.e. the LOOCV can be directly computed from a regression containing all data, by "downweighting" low-leverage data and upwighting high-leverage (hard to predict) data.

Leave-one-out Cross-Validation

Leaving out a single datapoint is going to be insufficient for testing real world out-of-sample performance, unless the **data are independent**. However, there is often a computationally convenient way to compute LOOCV, and it is still better than leaving nothing out. It converges to C_p for large n. Analogous tricks work for:

- **Linear models** including **Best Linear Unbiased Predictors** (BLUPs)
- **Kernel methods**
- **Nearest neighbour** methods
- And others

Asymptotics

Here are some facts about the asymptotic behaviour of LOOCV:

- As $n \to \infty$, the expected out-of-sample MSE of the model picked by LOOCV cross-validation is **close to that of the best model** considered.
- As $n \to \infty$, if the true model is among those being compared, LOOCV tends to pick a **strictly larger model** than the truth.

LOOCV is not the right tool for choosing the **right model**. It is however an excellent tool for choosing the model with the best out-of-sample **predictive power**.
…when the data to be predicted come from the **same distribution as the data**!

Problems with LOOCV

We might worry that leaving out one datapoint at a time isn't enough:

- **Cost**. It is straightforward to apply LOOCV to an arbitrary loss function, including a Likelihood. However, it can be costly.
- **Quality**. LOOCV estimates of out-of-sample loss has high variance because each test datapoint using $n - 2$ of the **same training datapoints**…
  - Empirically, we often choose a different model on different data generated under the same distribution!
- **Correlation**. Any correlation breaks LOOCV.

Naive **k-fold CV** addresses the first issue by creating a **bias-variance tradeoff**: we introduce a bias (towards simpler models) but also significantly reduce the variance of the MSE estimation.
More complicated sampling in k-fold settings can also address correlation.

K-fold CV

Split the data into k "folds" $f(i)$, that is, **random non-overlapping samples** of the data of size n/k. Then:

- **For each fold** i:
  - Call $X^{-(f(i))}$ the "training" dataset and $X^{(f(i))}$ the "test" dataset
  - Learn parameters $\hat{\theta}_i$ with data $X^{-(f(i))}$
  - Evaluate $l_i = \text{Loss}(X^{(f(i))}|\hat{\theta}_i)$
- And report $\sum_{i=1}^{k} l_i$

How many folds?

k-fold CV loses a fraction of the data, whereas LOOCV only loses a constant. This means that (under the assumption that the **true model is not in the model space**) k-fold CV will choose a **simpler model** with less predictive power than was possible. However, smaller k can make the inference more consistent across different data.
For **small data**, LOOCV is recommended. For **larger data**, $k = 10$ is often chosen:

- **cost**. k defines the minimum number of times you need to run the models. If you can afford to run a model once, you can probably afford 10 times.
- **practicality**. If you had only 10% more data you might expect to get the same performance as LOOCV. We frequently lose this amount of data to quality control etc.

Handling correlation

Correlation structures can be handled in k-fold CV by **careful sampling**:

- a-priori there is a correlation in time or space expected. we can therefore **remove windows**.
- the data have some associated covariate, which can be removed en-masse.
- empirical correlation structures can be used to select a point i and all points correlated with it above some **correlation threshold**.

Some of these can be used in other contexts - for examples see the **block bootstrap**, or simply using a different definition of a "datapoint" in a leave-one-out context.

Signposting

Cross Validation is extremely poplular **because it works**. It is probably the most important component of machine learning.
We'll take a look at an example with the remaining time.
Next time: **Latent Structures and Clustering**