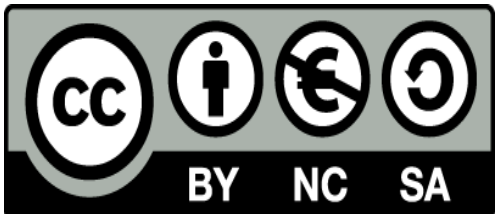
uc3m | Universidad Carlos III de Madrid




# OPENCOURSEWARE
# ADVANCED PROGRAMMING
# STATISTICS FOR DATA SCIENCE

Ricardo Aler

MASTER IN STATISTICS FOR DATA SCIENCE. UC3M.

ADVANCED PROGRAMMING.

2 HOURS

---

1. What is “Rcpp sugar” and why it is useful?

   Rcpp sugar allows to use some R functions within Rcpp and makes programming in Rcpp easier.

2. Explain the difference between dynamic and static typing.

   Dynamic typing allows variables to contain values that may belong to any type (this is what happens in Python or R). Static typing forces variables to contain just one type (like integer), which must be specified in advance (this is what happens in Rcpp).

3. Explain what *data-leakage* is. Give an actual example of *data-leakage*.

   Data leakage happens when information from the test data is (wrongly) used to train the model. For instance, this might happen if we do feature selection with the whole dataset, before partitioning the data into training and test. That would mean that information belonging to test would have been used to select the features, which is wrong. Pipelines help in avoiding data leakage.

4. What is the difference betwen a shallow copy and a deep copy? Explain with an example (code not required, you can use a picture if you prefer).

   Answer: For instance, in the following example:

   ```
   xs = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
   ys = xs
   ```

   ys would not be a copy of xs, but a reference to xs. If we modify xs, we also modify ys. If we want to avoid this, we would have to do a “Deep copy” of xs (i.e. a complete copy of xs).

5. What is the main difference between an iterator and a list?

   A list is a collection of elements. An iterator is not an actual list, but it can be used to iterate over a collection of elements (for instance, in a for loop).

6. What is meant by “*broadcasting*” in *Numpy*? Write an example where *broadcasting* is used and explain why it is useful (correct syntax is not important).

   Broadcasting is used to make operations between vectors (or matrices) of different sizes. For instance, it can be useful to do (3,2,1) + (7), because it is transformed into

this operation (3,2,1) + (7,7,7), which can now be performed because both vectors have the same number of elements.

7. What are universal functions? Is *np.sum()* a universal function (*np.sum*([1,2,3])=6)? Why?

   Universal functions allow to make element-wise operations on vectors and matrices. For instance (1,2,3) + (4,5,6) = (1+4, 2+5, 3+6), where + is applied to each of the corresponding elements of the two vectors. np.sum is not an universal function but a reducing function (it reduces a vector to a number, in this case).

8. Both Numpy 2-dimensional arrays and Pandas dataframes are 2-dimensional structures. What is the main difference between them?

   Numpy arrays must contain elements that belong to the same type (integers, etc.). In Dataframes, each column can belong to a different type (a column of integers, another column of categorical values, etc.). Dataframe columns can also have names, while numpy arrays cannot.

9. If you want to do boolean selection in Pandas, what could be used, *.loc* or *.iloc*? Why?

   In Pandas there are three ways of doing selection: by label (.loc), by position/integer (.iloc) and boolean. Boolean selection can actually be done using both .loc and .iloc.

10. If the following C++ function is compiled:

```
NumericVector f(NumericVector x)
{
  NumericVector out = x;
  out[0] = 3;
  return(out);
}
```

    and then it is called from R, as follows, what would be printed by *print(x)* and *print(y)*? Why?

```
x <- c(1,2,3)
y = f(x)
print(x)
print(y)
```

    Answer: the underlying concept here is cloning (copying) vs. References. In Rcpp, references are always used, except when explicitely cloning/copying. In the example above *out* would be a reference to *x*, so when modifying *out (out[0]=3))*, *x* would also be modified. Therefore, both print(x) and print(y) would print (3,2,3)

11. Is the following code sensible? Why?

```
param_grid = {'max_depth': [1,2,3]}
rs = RandomizedSearchCV(DecisionTreeRegressor(),param_grid,n_iter=10)
rs.fit(X, y)
```

    Answer: RandomizedSearch is used for hyper-parameter tuning. It randomly chooses values from a list of possible values. The number of times this random choosing is carried out is specified by n_iter. Given that there are only 3 possible values ([1,2,3]), it

would not make sense to randomly choose values 10 times, because some of them would be repeated. In this case, gridsearch would be more efficient.

12. What would be the contents of variable *result* after executing this piece of code? Why?

```
import numpy as np
result = np.array(['1',2,3])
```

Answer: the underlying concept here is that in Numpy, all elements in a vector must belong to the same type. Here, '1' is a string, but 2 and 3 are integers, therefore Numpy would transform all of them to the same type. In this particular case, result would end up containing ['1', '2', '3'] (all elements would be strings), but the concept described in the previous sentence would be enough.

13. What would be the contents of variable *result* after executing this code? Why?

```
d = {'a': 3}      # Note: d is a standard dictionary
result = d['b']
```

Answer: in a standard dictionary, when trying to access a key not present in the dictionary ('b' is not present in the dictionary in this case), would return an error (in a default dictionary, it would return the default value, typically 0).

14. What would be the contents of variable *result* after executing this code?

```
result = 'abcdef'[2::-1]
```

Answer: 2::-1 is equivalent to 2:end:-1, with negative steps, and excluding the end. Therefore, it would return positions 2, 1, and 0: "cba"

15. What would be the the main difference between code A and code B below, after executing them?

```
# CODE A                    # CODE B
x = [1,2,3]                 x = [1,2,3]
x+[4,5,6]                   x.extend([4,5,6])
```

Answer: `x+[4,5,6]` would return [1,2,3,4,5,6] but would not modify x. x.extend([4,5,6]) is a method, and would modify x. After the comand, x would contain [1,2,3,4,5,6].

16. What would be the contents of variable *result* after executing this code?

```
result = [2*a for a in [b+5 for b in range(10) if b%2==0]  if a>=2 ]
```

Answer: list comprehension [b+5 for b in range(10) if b%2==0] would be all the even numbers between 0 and 9, plus five. That is: [0+5, 2+5, 4+5, 6+5, 8+5]. The other list comprehension would take all numbers larger than 2 (all of them, actually), and multiply them by 2. The result would be: [2*(0+5), 2*(2+5), 2*(4+5), 2*(6+5), 2*(8+5)].

17. Let's suppose that we want to use:
    a. 3-fold crossvalidation for model evaluation

b. 5-fold crossvalidation for hyper-parameter tuning with grid-search. There are two hyper-parameters: *C* with possible values 0.01, 0.1, and 1; and *gamma* with possible values “linear” and “rbf”.

Then, how many models will be trained? Explain why.

Answer: For every hyper-parameter value combination, a different model must be trained (and evaluated). There are 3*2=6 such combinations. If 5-fold crossvalidation is used, this will be done 5 times, hence 5*3*2. If 3-fold crossvalidation is used for model evaluation, the whole process will be repeated 3 times, therefore 3*5*3*2 models will be trained.

18. Why is caching useful in a Pipeline? Explain an example of a pipeline where it is useful (with words, code is not necessary).

Answer: sometimes, when doing hyper-parameter tuning, a step in a pipeline should be done just once. For instance, when determining the optimal number of features to be selected, features must be ranked, and then the best k ones are selected. However, ranking must be done just once, even if in some cases 3 features are selected, or in other cases 7 features are selected, the ranking itself remains the same. Caching would compute the ranking just once and use the same ranking from which different number of features would be selected.

19. Why are priors useful in *Stan*?

Answer: they provide information. For instance, saying that a parameter belongs to the [0, +infinity] interval, gives less information than saying that it follows a normal centered around 1.6. In the latter case, Stan knows that the parameter is in a smaller region in parameter space than in the former case.

20. Give an example of a dataframe in wide format, and its translation to long format.

Wide:

| | Iteration | Method1 | Method2 | Method3 |
|---|---|---|---|---|
| **0** | 0 | 0.978094 | 0.580535 | 0.267976 |
| **1** | 1 | 0.078923 | 0.287646 | 0.282177 |
| **2** | 2 | 0.774219 | 0.908143 | 0.414082 |
| **3** | 3 | 0.427651 | 0.720817 | 0.305179 |
| **4** | 4 | 0.462860 | 0.051145 | 0.798201 |
| **5** | 5 | 0.582603 | 0.065501 | 0.453163 |
| **6** | 6 | 0.744782 | 0.448524 | 0.499329 |
| **7** | 7 | 0.620550 | 0.993869 | 0.267858 |
| **8** | 8 | 0.308720 | 0.855421 | 0.037727 |
| **9** | 9 | 0.981836 | 0.568189 | 0.238975 |

Long:

:

| | Iteration | Method_ID | Error |
|---|---|---|---|
| **0** | 0 | Method1 | 0.978094 |
| **1** | 1 | Method1 | 0.078923 |
| **2** | 2 | Method1 | 0.774219 |
| **3** | 3 | Method1 | 0.427651 |
| **4** | 4 | Method1 | 0.462860 |
| **5** | 5 | Method1 | 0.582603 |
| **6** | 6 | Method1 | 0.744782 |
| **7** | 7 | Method1 | 0.620550 |
| **8** | 8 | Method1 | 0.308720 |
| **9** | 9 | Method1 | 0.981836 |
| **10** | 0 | Method2 | 0.580535 |
| **11** | 1 | Method2 | 0.287646 |
| **12** | 2 | Method2 | 0.908143 |
| **13** | 3 | Method2 | 0.720817 |
| **14** | 4 | Method2 | 0.051145 |
| **15** | 5 | Method2 | 0.065501 |
| **16** | 6 | Method2 | 0.448524 |
| **17** | 7 | Method2 | 0.993869 |
| **18** | 8 | Method2 | 0.855421 |
| **19** | 9 | Method2 | 0.568189 |
| **20** | 0 | Method3 | 0.267976 |
| **21** | 1 | Method3 | 0.282177 |
| **22** | 2 | Method3 | 0.414082 |
| **23** | 3 | Method3 | 0.305179 |
| **24** | 4 | Method3 | 0.798201 |
| **25** | 5 | Method3 | 0.453163 |
| **26** | 6 | Method3 | 0.499329 |
| **27** | 7 | Method3 | 0.267858 |
| **28** | 8 | Method3 | 0.037727 |
| **29** | 9 | Method3 | 0.238975 |