4 Data Splitting

Contents

- Simple Splitting Based on the Outcome
- Splitting Based on the Predictors
- Data Splitting for Time Series
- Data Splitting with Important Groups

4.1 Simple Splitting Based on the Outcome

The function `createDataPartition` can be used to create balanced splits of the data. If the `y` argument to this function is a factor, the random sampling occurs within each class and should preserve the overall class distribution of the data. For example, to create a single 80/20% split of the iris data:

```r
library(caret)
set.seed(3456)
trainIndex <- createDataPartition(iris$Species, p = .8,
                                  list = FALSE,
                                  times = 1)
head(trainIndex)
```

createDataPartition

y	a vector of outcomes. For createTimeSlices, these should be in chronological order.
times	the number of partitions to create
p	the percentage of data that goes to training
list	logical - should the results be in a list (TRUE) or a matrix with the number of rows equal to floor(p * length(y)) and times columns.

```
##      Resample1
## [1,]         1
## [2,]         2
## [3,]         4
## [4,]         5
## [5,]         6
## [6,]         8
```

```r
irisTrain <- iris[ trainIndex,]
irisTest  <- iris[-trainIndex,]
```

The `list = FALSE` avoids returning the data as a list. This function also has an argument, `times`, that can create multiple splits at once; the data indices are returned in a list of integer vectors. Similarly, `createResample` can be used to make simple bootstrap samples and `createFolds` can be used to generate balanced cross–validation groupings from a set of data.

createFolds(y, k = 10, list = TRUE, returnTrain = FALSE)
createResample(y, times = 10, list = TRUE)

times	the number of partitions to create
k	an integer for the number of folds.

4.2 Splitting Based on the Predictors

Also, the function `maxDissim` can be used to create sub–samples using a maximum dissimilarity approach (Willett, 1999). Suppose there is a data set *A* with *m* samples and a larger data set *B* with *n* samples. We may want to create a sub–sample from *B* that is diverse when compared to *A*. To do this, for each sample in *B*, the function calculates the *m* dissimilarities between each point in *A*. The most dissimilar point in *B* is added to *A* and the process continues. There are many methods in R to calculate dissimilarity. `caret` uses the `proxy` package. See the manual for that package for a list of available measures. Also, there are many ways to calculate which sample is "most dissimilar". The argument `obj` can be used to specify any function that returns a scalar measure. `caret` includes two functions, `minDiss` and `sumDiss`, that can be used to maximize the minimum and total dissimilarities, respectfully.

As an example, the figure below shows a scatter plot of two chemical descriptors for the Cox2 data. Using an initial random sample of 5 compounds, we can select 20 more compounds from the data so that the new compounds are most dissimilar from the initial 5 that were specified. The panels in the figure show the results using several combinations of distance metrics and scoring functions. For these data, the distance measure has less of an impact than the scoring method for determining which compounds are most dissimilar.

```r
library(mlbench)
data(BostonHousing)

testing <- scale(BostonHousing[, c("age", "nox")])
set.seed(5)
## A random sample of 5 data points
startSet <- sample(1:dim(testing)[1], 5)
samplePool <- testing[-startSet,]
start <- testing[startSet,]
newSamp <- maxDissim(start, samplePool, n = 20)
head(newSamp)
```

在1：506的数里随机选5 个数

maxDissim(a, b, n = 2, obj = minDiss, useNames = FALSE, randomFrac = 1, verbose = FALSE, ...)

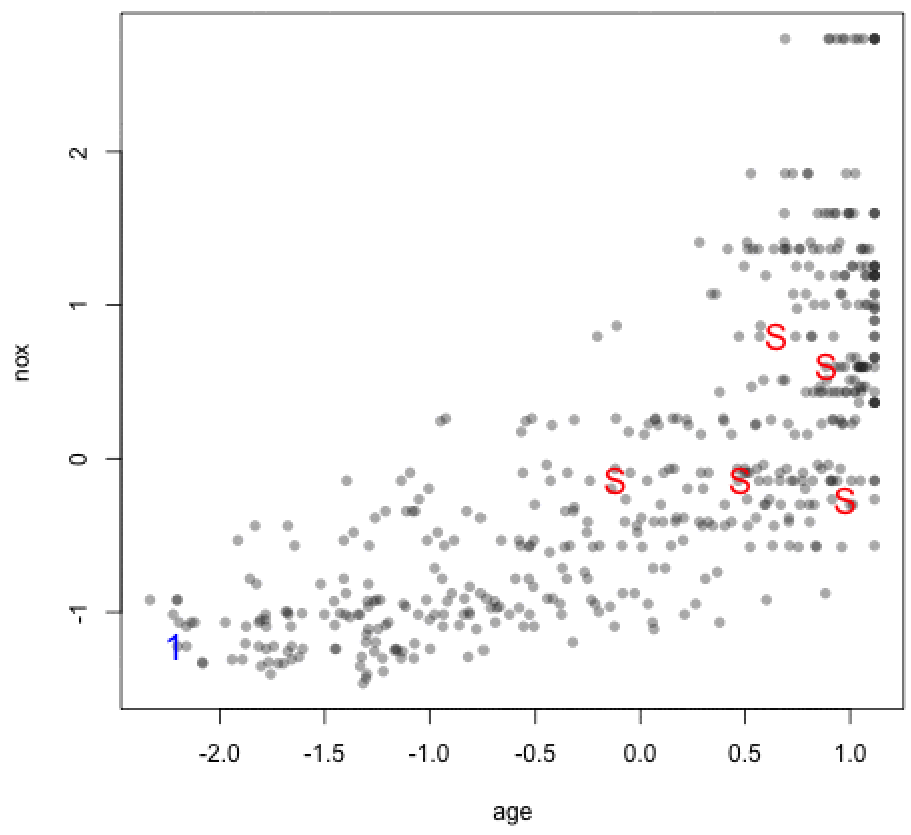
a	a matrix or data frame of samples to start
b	a matrix or data frame of samples to sample from
n	the size of the sub-sample

A：start: 小数据集，size=5
B：samplePool: 大数据集，size=506-5=501
希望从B中抽取和A不同的子样本。为了这样做，对于B中的每一个样本，函数计算与A中每一样本点的相异性。把B中最大相异性的点加到A并继续。
从B中抽取n=20个与A不同的子样本。

```
## [1] 461 406  49 308 469  76
```

得到的为什么是6个数而不是20个？
newSamp指的是什么？

The visualization below shows the data set (small points), the starting samples (larger blue points) and the order in which the other 20 samples are added.



4.3 Data Splitting for Time Series

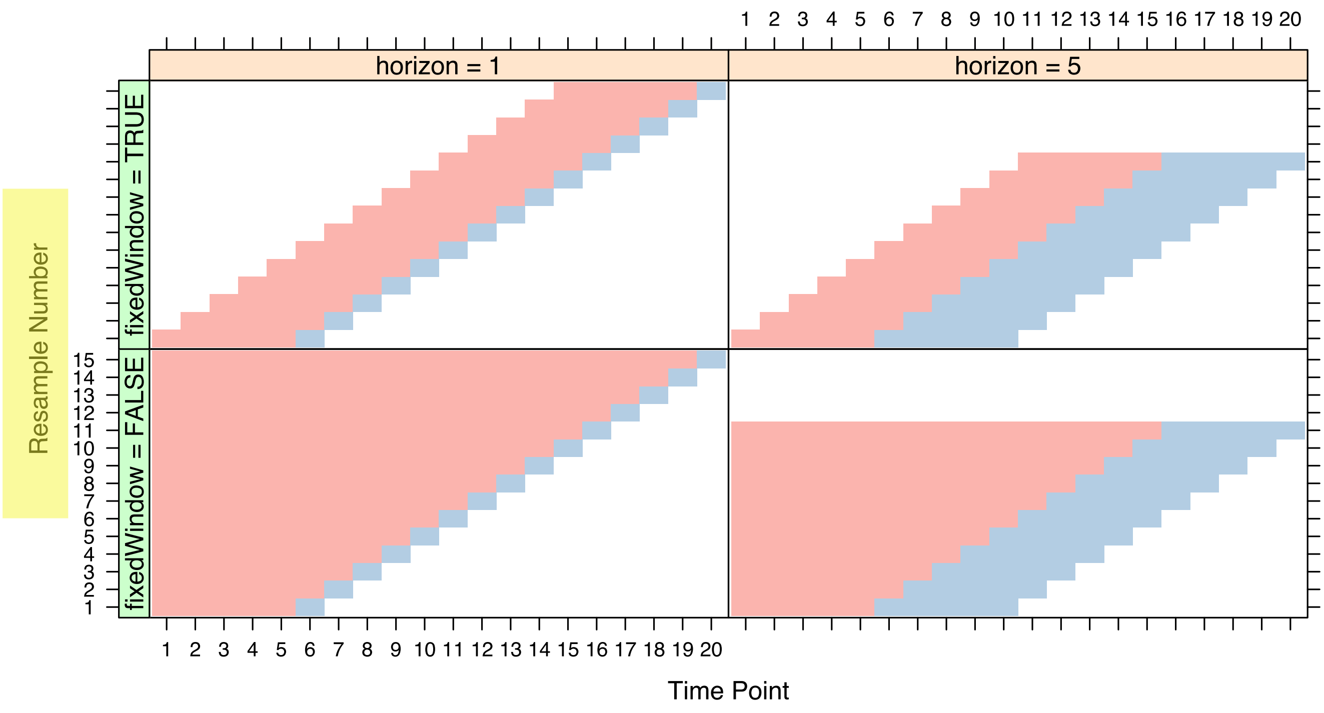
Simple random sampling of time series is probably not the best way to resample times series data. Hyndman and Athanasopoulos (2013) discuss *rolling forecasting origin* techniques that move the training and test sets in time. caret contains a function called `createTimeSlices` that can create the indices for this type of splitting.

The three parameters for this type of splitting are:

- `initialWindow` : the initial number of consecutive values in each training set sample
- `horizon` : The number of consecutive values in test set sample

- `fixedWindow` : A logical: if `FALSE` , the training set always start at the first sample and the training set size will vary over data splits.

As an example, suppose we have a time series with 20 data points. We can fix `initialWindow = 5` and look at different settings of the other two arguments. In the plot below, rows in each panel correspond to different data splits (i.e. resamples) and the columns correspond to different data points. Also, red indicates samples that are in included in the training set and the blue indicates samples in the test set.



是因为自变量是时间所以用creatTimeSlices吗？
看不懂这图。

4.4 Simple Splitting with Important Groups

In some cases there is an important qualitative factor in the data that should be considered during (re)sampling. For example:

- in clinical trials, there may be hospital-to-hospital differences
- with longitudinal or repeated measures data, subjects (or general independent experimental unit) may have multiple rows in the data set, etc.

There may be an interest in making sure that these groups are not contained in the training and testing set since this may bias the test set performance to be more optimistic. Also, when one or more specific groups are held out, the resampling might capture the “ruggedness” of the model. In the example where clinical data is recorded over multiple sites, the resampling performance estimates partly measure how extensible the model is across sites.

To split the data based on groups, `groupKFold` can be used:

```
set.seed(3527)
subjects <- sample(1:20, size = 80, replace = TRUE)
table(subjects)
```

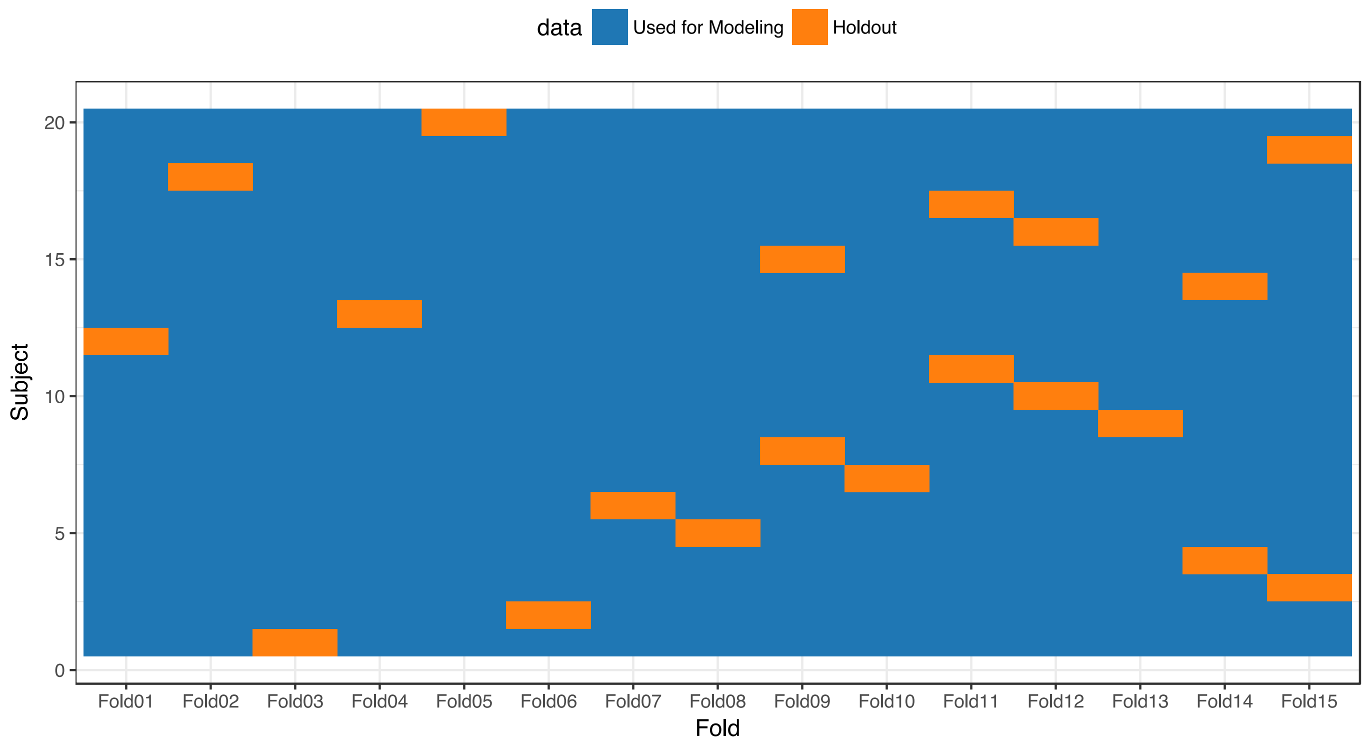
```
## subjects
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  3  5  4  3  2  6  4  6  3  5  5  3  5  6  4  4  7  1  3  1
```

```
folds <- groupKFold(subjects, k = 15)
```

15 组数，每组80个

The results in `folds` can be used as inputs into the `index` argument of the `trainControl` function.

This plot shows how each subject is partitioned between the modeling and holdout sets. Note that since `k` was less than 20 when `folds` was created, there are some holdouts with model than one subject.



K=15组数据。橙色代表测试集，蓝色代表训练集。

Ch4
splitting method

1.createDataPartition(p=)
2.maxDissim(n=)
3.CreatTimeSlices()
4.groupKFold(k=)