SLLD - Module 1

Cross Validation

S.Tonini, F.Chiaromonte

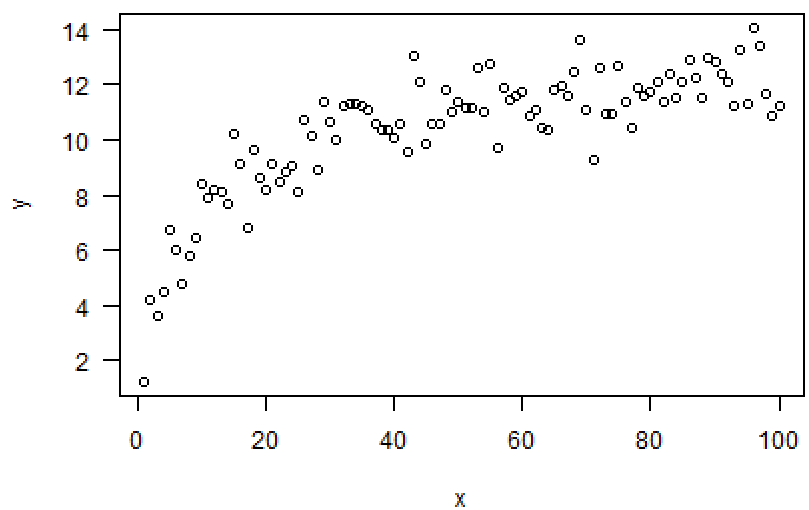
Sant'Anna School of Advanced Study - Pisa

21/2/2024

```r
library(tidyverse) # for data manipulation and visualization
library(ggplot2) # for plots
library(caret) # Classification and Regression Training
```

Data

```r
set.seed(123)
n <- 100
x <- seq(1, n, 1)
y <- ((runif(1, 10, 20) * x) / (runif(1, 0, 10) + x)) +
  rnorm(n, 0, 1)
plot(x, y, col = 9, las = 1)
```



Cross validation

We are going to perform CV by hand. Precisely we are going to perform:

- Leave-one-out cross validation (LOOCV)

- k-folds cross validation

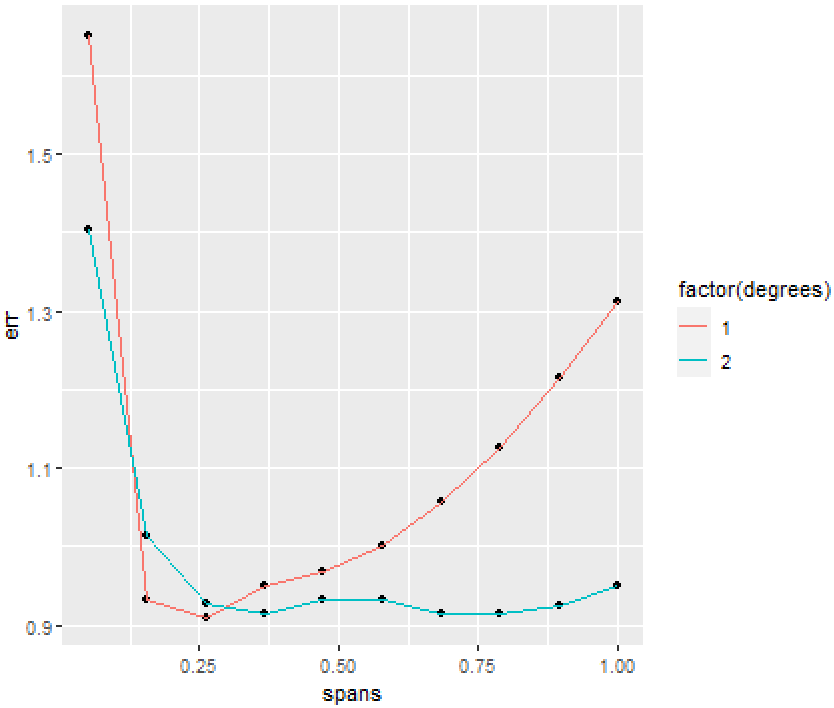
We use CV to evaluate the function **loess** at several **span** and **degree** values. Remember:

- **span:** the parameter which controls the degree of smoothing
- **degree:** the degree of the polynomials to be used, normally 1 or 2

LOOCV: in this approach, we reserve only one data point from the available dataset, and train the model on the rest of the data. This process iterates for each data point.

```
set.seed(2024)
df<-data.frame(cbind(y,x))
degree_list <- list()
span_values <- seq(0.05,1,length=10)
for(deg in 1:2){ #polynomial degree
  err <- list()
  for(k in 1:length(span_values)){ #smoothness
    score <- list()
    for(i in 1:(nrow(df))){
      training = df[-i,]
      model = loess(y ~ x, data = training,
                    span = span_values[k], degree=deg)
      validation = df[i,]
      pred = na.omit(predict(model, validation))
      # error of ith fold
      score[[i]] = (validation$y - pred)^2
      }
    # returns a vector with the average error for degree & span
    err[[k]] <- mean(unlist(score),na.rm=TRUE)
    }
  degree_list[[deg]] <- err
  }
spans <- rep(span_values,2)
degrees <- rep(c(1,2), each = length(span_values))
```

```
err <- unlist(degree_list)
df_toplot <- as.data.frame(cbind(spans,degrees,err))
p <- ggplot(df_toplot, aes(x=spans, y=err, group=factor(degrees))) +
geom_point() + geom_line(aes(col=factor(degrees)))
p
```



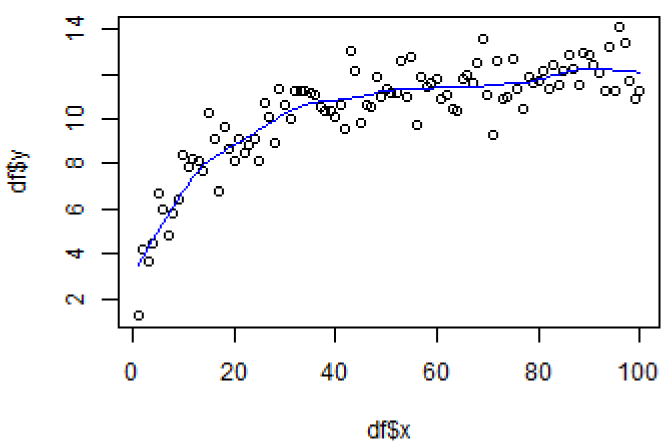
Let's find the parameters corresponding to the minimum error.

```
best <- df_toplot[which(df_toplot$err==min(df_toplot$err)),]
best
```

```
##       spans degrees       err
## 3 0.2611111       1 0.9095377
```

Let's plot the resulting smoothed curve.

```
res <- loess(y ~ x, data = df, span =best$spans,
             degree=best$degrees)
plot(df$x, df$y)
lines(predict(res), col='blue')
```



k-fold CV. Let's validate the parameter using the k-fold cross validation. These are the steps we need to implement:

- Randomly split your entire dataset into k folds;
- Iterate across each kth fold, which serves as a testing set, and train your model only on the remaining $k-1$ folds;
- Test model accuracy/effectiveness on the kth fold, and record the "error" you see on each of the k predictions;
- Repeat this until each of the k-folds has served as the test set;
- The average of your k recorded errors is called the **cross validation error** and will serve as a performance metric for the model.

Create the folds:

```r
options(width = 70)
flds <- caret::createFolds(1:nrow(df),
                           k = 5, list = TRUE,
                           returnTrain = FALSE)
flds
```

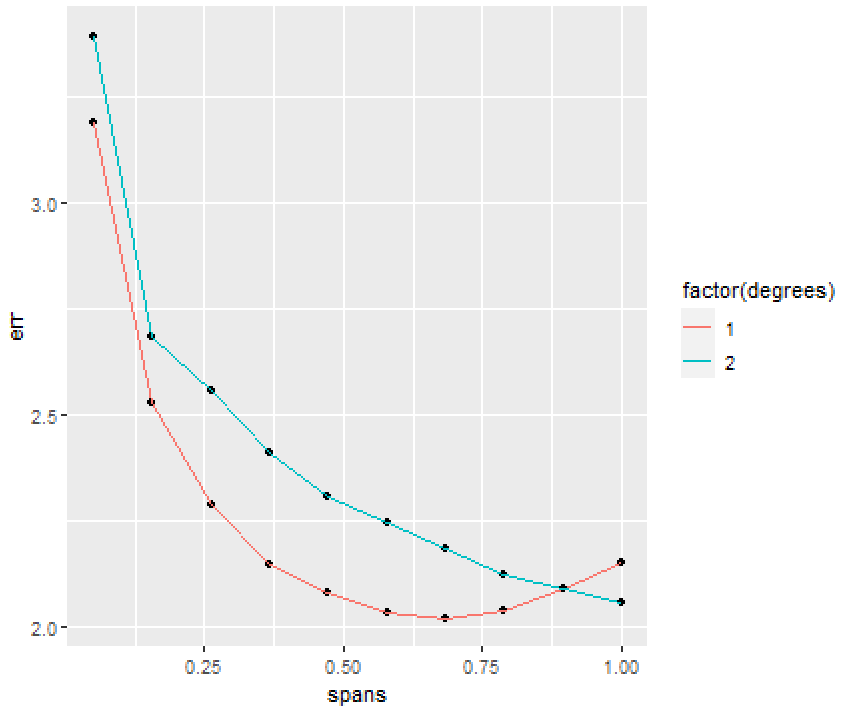
```
## $Fold1
##  [1]  3 10 11 14 16 33 34 35 43 50 54 56 63 68 70 76 81 82 91 93
##
## $Fold2
##  [1]  6 12 17 21 23 26 29 37 40 45 59 66 67 72 73 83 84 86 92 95
##
## $Fold3
##  [1]  7  9 15 22 25 28 30 44 46 47 55 60 64 69 74 80 85 90 98 99
##
## $Fold4
##  [1]   2   4   5   8  24  31  36  38  42  49  51  58  61  71  75  78
## [17]  89  96  97 100
##
## $Fold5
##  [1]  1 13 18 19 20 27 32 39 41 48 52 53 57 62 65 77 79 87 88 94
```

```r
# you can use [[k]] or [k] to access the k-th element
# flds[1]     # you need to "unlist" afterwards (see below)
# flds[[1]]   # you do not
```

Perform an iteration similarly to the one for LOOCV:

```
degree_list <- list()
for(deg in 1:2){ #polynomial degree
  err <- list()
  for(k in 1:length(span_values)){ #smoothness
    score <- list()
    for(i in 1:length(flds)){
      validation <- df[unlist(flds[i]),]
      training <- df[unlist(flds[-i]),]
      model = loess(y ~ x, data = training,
                    span = span_values[k], degree=deg)
      pred = na.omit(predict(model, validation))
      score[[i]] <- mean((pred - validation$y)^2, na.rm=TRUE)
    }
    err[[k]] <- mean(unlist(score))
  }
  degree_list[[deg]] <- unlist(err)
}
spans <- rep(span_values,2)
degrees <- rep(c(1,2), each = length(span_values))
err <- unlist(degree_list)
df_toplot <- as.data.frame(cbind(spans,degrees,err))
```

```
p <- ggplot(df_toplot, aes(x=spans, y=err, group=factor(degrees))) +
  geom_point() + geom_line(aes(col=factor(degrees)))
p
```



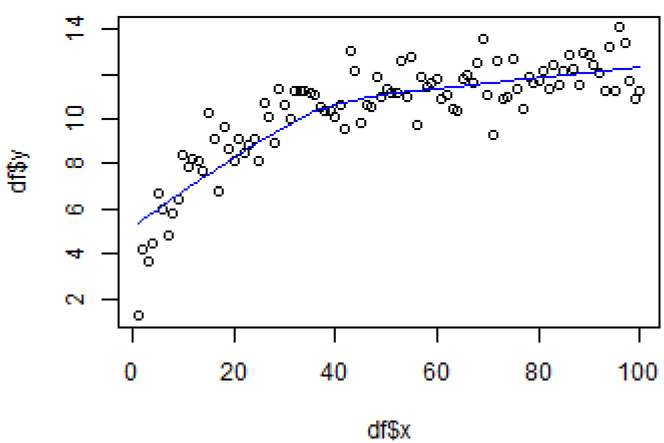
Let us find the parameters corresponding to the minimum error.

```
df_toplot[which(df_toplot$err==min(df_toplot$err)),]
```

```
##       spans degrees      err
## 7 0.6833333       1 2.017971
```

Let us plot the resulting regression line.

```
best <- df_toplot[which(df_toplot$err==min(df_toplot$err)),]
res <- loess(y ~ x, data = df, span =best$spans,
             degree=best$degrees)
plot(df$x, df$y)
lines(predict(res), col='blue')
```



Using CARET for CV

Note, cross validation is also implemented in the **train** function of the **caret** package. Here caret train function allows one to train different algorithms using the same syntax.

But, by default, caret used bootstrap resampling with 25 repetitions – this is the default resampling approach in caret.

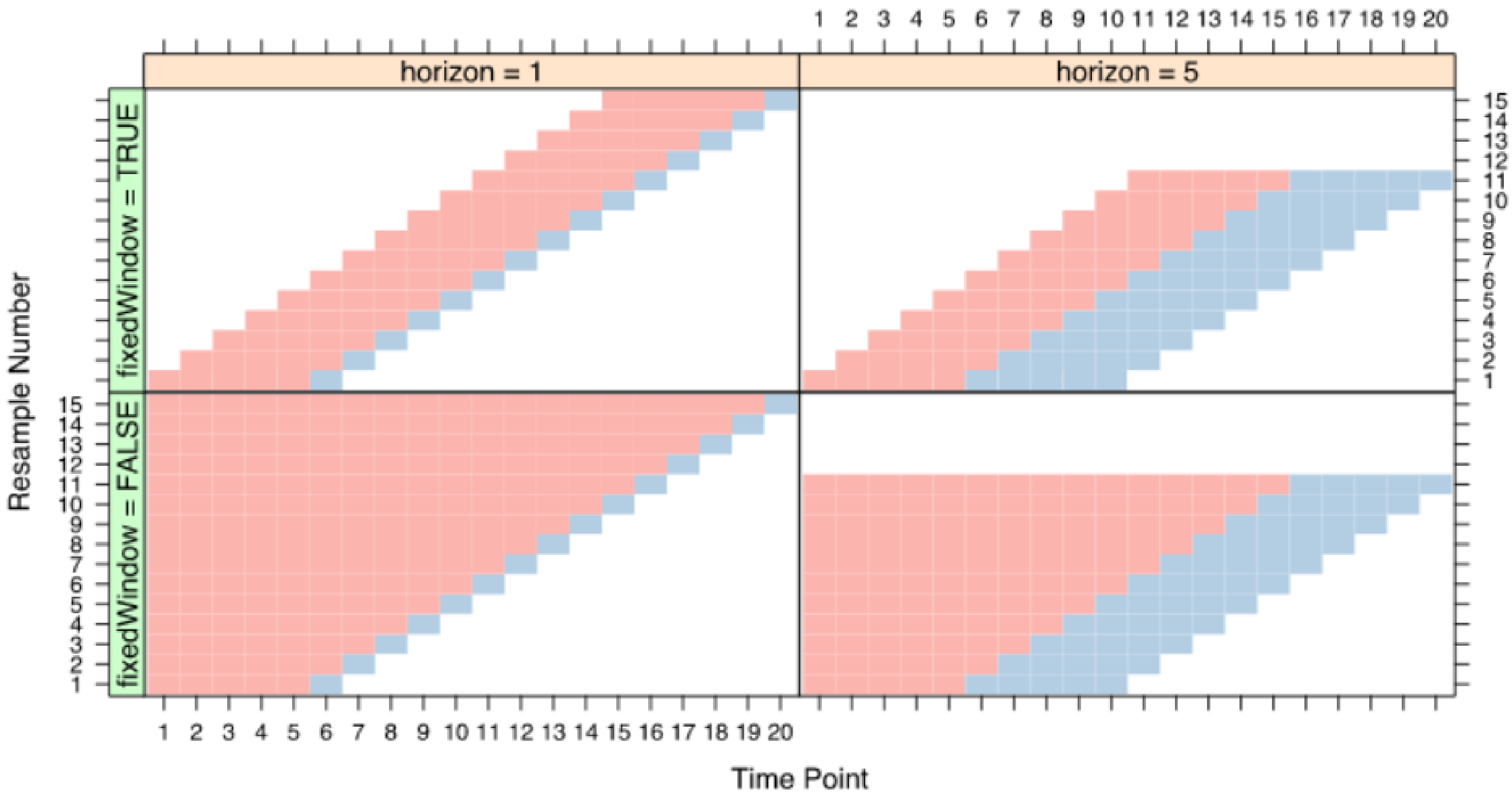
Check out the toolkit to see how it works!

Temporal Block "Cross" Validation

Simple random sampling is probably not the best way to resample time series data. **caret** contains a function called **createTimeSlices** that can create the indices for this type of splitting.

The function takes as input a vector and three parameters:

- **y:** a vector of outcomes. These should be in chronological order
- **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.


horizon = 1
horizon = 5
fixedWindow = TRUE
fixedWindow = FALSE
Resample Number
Time Point

Example 1: one training set and one test set

```
options(width = 60)
p <- 0.75
Ex1 <- createTimeSlices(y = 1:nrow(df),
                        initialWindow = round(p*nrow(df),0),
                        horizon = (nrow(df)-round(p*nrow(df),0)),
                        fixedWindow = TRUE)
# in this example fixedWindow doesn't matter!!
Ex1$train
```

```
## $Training75
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
## [19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
## [55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## [73] 73 74 75
```

```
Ex1$test
```

```
## $Testing75
##  [1]  76  77  78  79  80  81  82  83  84  85  86  87  88  89
## [15]  90  91  92  93  94  95  96  97  98  99 100
```

Example 2: rolling window

```r
options(width = 60)
Ex2 <- createTimeSlices(y = 1:nrow(df),
                 initialWindow = round(p*nrow(df),0),
                 horizon = ((nrow(df)-round(p*nrow(df),0))-2),
                 fixedWindow = TRUE)
Ex2$train[1]
```

```
## $Training75
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
## [19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
## [55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## [73] 73 74 75
```

```r
Ex2$test[1]
```

```
## $Testing75
##  [1] 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
## [19] 94 95 96 97 98
```

```r
options(width = 60)
Ex2$train[2]
```

```
## $Training76
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
## [19] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
## [37] 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
## [55] 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
## [73] 74 75 76
```

```r
Ex2$test[2]
```

```
## $Testing76
##  [1] 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94
## [19] 95 96 97 98 99
```

```r
options(width = 60)
Ex2$train[3]
```

```
## $Training77
##  [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## [19] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
## [37] 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
## [55] 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
## [73] 75 76 77
```

```r
Ex2$test[3]
```

```
## $Testing77
##  [1]  78  79  80  81  82  83  84  85  86  87  88  89  90  91
## [15]  92  93  94  95  96  97  98  99 100
```

Example 3: recursive window

```
options(width = 60)
Ex3 <- createTimeSlices(y = 1:nrow(df),
                 initialWindow = round(p*nrow(df),0),
                 horizon = ((nrow(df)-round(p*nrow(df),0))-2),
                 fixedWindow = FALSE)
Ex3$train[1]
```

```
## $Training75
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
## [19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
## [55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## [73] 73 74 75
```

```
Ex3$test[1]
```

```
## $Testing75
##  [1] 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
## [19] 94 95 96 97 98
```

```r
options(width = 60)
Ex3$train[2]
```

```
## $Training76
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
## [19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
## [55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## [73] 73 74 75 76
```

```r
Ex3$test[2]
```

```
## $Testing76
##  [1] 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94
## [19] 95 96 97 98 99
```

```r
options(width = 60)
Ex3$train[3]
```

```
## $Training77
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
## [19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
## [55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## [73] 73 74 75 76 77
```

```r
Ex3$test[3]
```

```
## $Testing77
##  [1]  78  79  80  81  82  83  84  85  86  87  88  89  90  91
## [15]  92  93  94  95  96  97  98  99 100
```

Now it's your turn!!!