Laboratory 5 - Statistical Methods

N Torelli, G Di Credico, V Gioia

15/12/2024

Contents

Regularisation techniques

The idea of the regularisation techniques (shrinkage methods) is to consider a *penalised likelihood framework*.

Here, we will see an example of ridge and LASSO regression. The shrinked coefficient can be obtained as

$$\tilde{\beta} = \text{argmin}_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} -l(\eta_i; y_i) + \lambda[(1-\alpha)\frac{1}{2}\sum_{j=1}^{p} \beta_j^2 + \alpha \sum_{j=1}^{p} |\beta_j|]$$

where

- $\eta_i = \beta_0 + \beta_1 x_{i1} + \ldots \beta_p x_{ip}$ is the i-th linear predictor
- $-l(\eta_i; y_i)$: negative log-likelihood contribution, up to an additive constant which does not depend on β (in the Gaussian case $-l(\eta_i; y_i) = \frac{1}{2\sigma^2}(y_i - \eta_i)^2)$, with $\eta_i = \mu_i$; while in the binomial case $-l(\eta_i; y_i) = -y_i\eta_i + \log(1 + \exp(\eta_i)))$, con $\eta_i = \text{logit}(p_i)$
- $\alpha = 0 \implies$ **Ridge regression**
- $\alpha = 1 \implies$ **Lasso regression**
- $0 < \alpha < 1$ different regularisation technique
- λ: tuning parameter, which controls the strength of penalisation. It could be fixed or selected by using cross-validation techniques.
  - $\lambda \to 0 \quad \implies \quad \hat{\boldsymbol{\beta}}_{LASSO} \to \hat{\boldsymbol{\beta}}_{LS}$ and $\hat{\beta}_{RIDGE} \to \hat{\boldsymbol{\beta}}_{LS}$, where $\hat{\beta}_{LS}$ refers to the least square.
  - For large λ the estimates are shrunk toward zero

The regularisation techniques allows to overcome possible problems of multicollinearity, and for the LASSO also to reduce the complexity of the models by selecting a model with a good trade-off between parsimony (it allows variable selection) and goodness of fit.

There is also a nice Bayesian interpretation for both Ridge and LASSO. See *An Introduction to Statistical Learning: with Applications in R* by James, G., Witten D., Hastie, T., Tibshirani, T. (2013)

In the following we will use the **glmnet** package routines to fit ridge and LASSO regression. Ridge regression in R can be also obtained with the function `lm.ridge()` in `MASS` package. See also https://glmnet.stanford.edu/articles/glmnet.html for an exhaustive introduction to the main functionality of the **glmnet** package and to explore several applications moving beyond the classical normal linear model.

Prostate data example

Prostate dataset (from the deprecated R package **lasso2**) aims to explore the relationship between the prostate specific antigen (PSA) and a number of clinical measures in men who were about to receive a radical prostatectomy.

- **lcavol**: log(cancer volume)
- **lweight**: log(prostate weight)
- **age**: age
- **lbph**: log(benign prostatic hyperplasia amount)
- **svi**: seminal vesicle invasion
- **lcp**: log(capsular penetration)
- **gleason**: Gleason score
- **pgg45**: percentage Gleason scores 4 or 5
- **lpsa**: log(prostate specific antigen) ← **outcome**

```
load("Prostate.rda")

str(Prostate)
```

```
## 'data.frame':    97 obs. of  9 variables:
##  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
##  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
##  $ age    : num  50 58 74 58 62 50 64 58 47 63 ...
##  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ svi    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ gleason: num  6 6 7 6 6 6 6 6 6 6 ...
##  $ pgg45  : num  0 0 20 0 0 0 0 0 0 0 ...
##  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

```
summary(Prostate[, -5])
```

```
##      lcavol           lweight           age             lbph
##  Min.   :-1.3471   Min.   :2.375   Min.   :41.00   Min.   :-1.3863
##  1st Qu.: 0.5128   1st Qu.:3.376   1st Qu.:60.00   1st Qu.:-1.3863
##  Median : 1.4469   Median :3.623   Median :65.00   Median : 0.3001
##  Mean   : 1.3500   Mean   :3.653   Mean   :63.87   Mean   : 0.1004
##  3rd Qu.: 2.1270   3rd Qu.:3.878   3rd Qu.:68.00   3rd Qu.: 1.5581
##  Max.   : 3.8210   Max.   :6.108   Max.   :79.00   Max.   : 2.3263
```

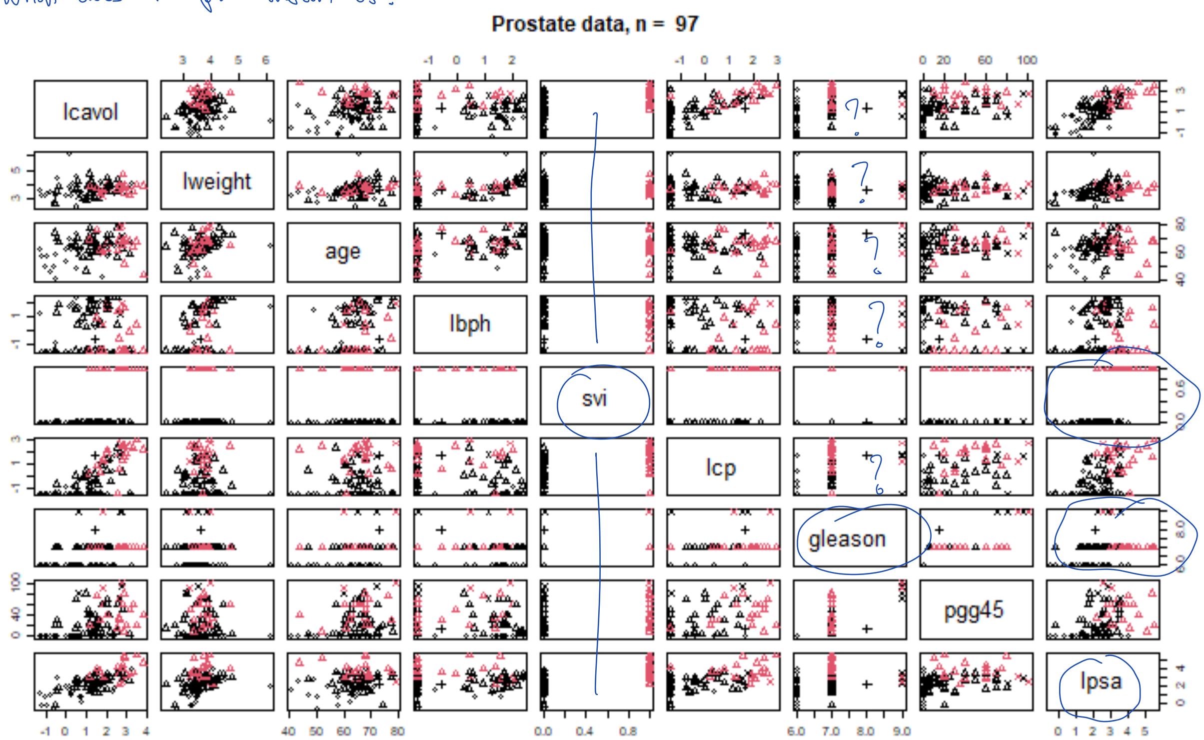
```
##       lcp              gleason           pgg45              lpsa
##  Min.   :-1.3863   Min.   :6.000   Min.   :  0.00   Min.   :-0.4308
##  1st Qu.:-1.3863   1st Qu.:6.000   1st Qu.:  0.00   1st Qu.: 1.7317
##  Median :-0.7985   Median :7.000   Median : 15.00   Median : 2.5915
##  Mean   :-0.1794   Mean   :6.753   Mean   : 24.38   Mean   : 2.4784
##  3rd Qu.: 1.1787   3rd Qu.:7.000   3rd Qu.: 40.00   3rd Qu.: 3.0564
##  Max.   : 2.9042   Max.   :9.000   Max.   :100.00   Max.   : 5.5829
```

```r
table(Prostate$svi)/nrow(Prostate)
```

```
##
##         0         1
## 0.7835052 0.2164948
```

```r
pairs(Prostate, col = 1 + Prostate$svi,
      pch = Prostate$gleason - 5,
      main = paste("Prostate data, n = ", nrow(Prostate)))
```

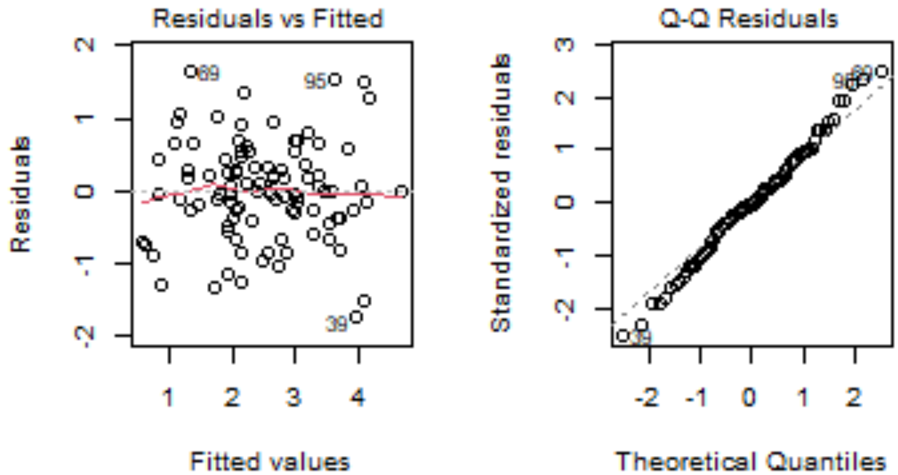
What does this plot inform us?



Let's assume a linear model for the antigene level (*lpsa*) including all the available covariates

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

```r
fit.lin <- lm(lpsa ~ ., data = Prostate)
par(mfrow = c(1,2))
plot(fit.lin, which = c(1, 2))
```




```r
fit1s <- summary(fit.lin)
fit1s
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = Prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.73316 -0.37133 -0.01702  0.41414  1.63811
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.669399   1.296381   0.516  0.60690
## lcavol       0.587023   0.087920   6.677 2.11e-09 ***
## lweight      0.454461   0.170012   2.673  0.00896 **
## age         -0.019637   0.011173  -1.758  0.08229 .
## lbph         0.107054   0.058449   1.832  0.07040 .
## svi          0.766156   0.244309   3.136  0.00233 **
## lcp         -0.105474   0.091013  -1.159  0.24964
## gleason      0.045136   0.157464   0.287  0.77506
## pgg45        0.004525   0.004421   1.024  0.30885
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7084 on 88 degrees of freedom
## Multiple R-squared:  0.6548, Adjusted R-squared:  0.6234
## F-statistic: 20.86 on 8 and 88 DF,  p-value: < 2.2e-16
```

Let's explore the VIF index

```r
library(car)
vif(fit.lin)
```

```
##   lcavol  lweight      age     lbph      svi      lcp  gleason    pgg45
## 2.054113 1.363706 1.323600 1.375537 1.956882 3.097954 2.473403 2.974362
```

< 10 ⇒ OK

Ridge regression

In order to use the capability of the **glmnet** we must organise the data to be passed to the **glmnet** function, which requires to provide seprately the outcome vector and the model matrix (without intercept). Note that the estimator takes the form $\hat{\boldsymbol{\beta}}_{RIDGE} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{K})\mathbf{X}^t \mathbf{y}$ where $K = \text{diag}(0, 1, \ldots, 1)$ as the penalization of the intercepts is undesired. Note that the ridge estimator is biased but has smaller variance than the LS one.

```
library(glmnet)
n <- dim(Prostate)[1]
p <- dim(Prostate)[2]
X <- as.matrix(Prostate[, 1 : 8])
y <- Prostate[, p]
fit_ridge <- glmnet(X, y, alpha = 0)
```

Then we can print the percent (of null) deviance explained (%dev) and the value of λ

```
print(fit_ridge)
```

→ Df %Dev Lambda

Might be important

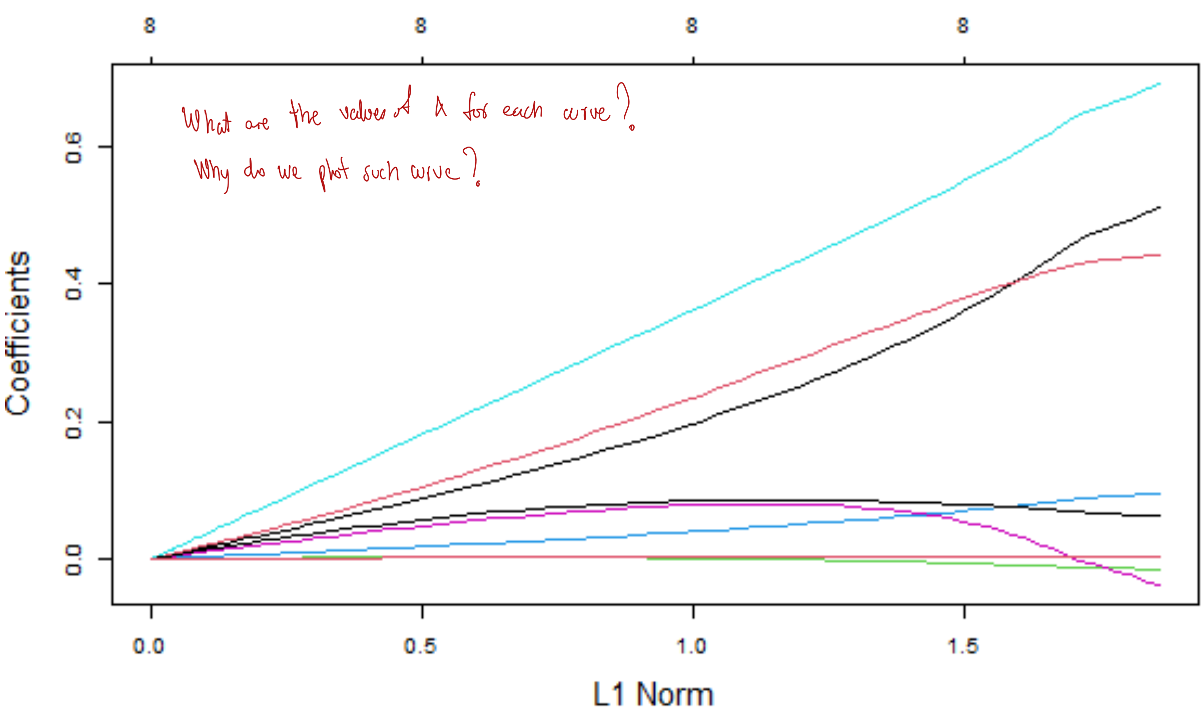
From the **fit_ridge** model we can select a specific model according to a value of λ (denoted as the argument s). For instance, one can select the model corresponding to $\lambda = 0.1$ and explore the coefficients.

```
round(t(coef(fit_ridge, s = 0.1)), 4)
```

```
## 1 x 9 sparse Matrix of class "dgCMatrix"
##    (Intercept) lcavol lweight     age   lbph    svi     lcp gleason  pgg45
## s1      0.4553 0.5007  0.4397 -0.0146 0.0934 0.6801 -0.0295  0.0633 0.0033
```

However, it is better to visualise the path of its coefficient against the ℓ_1-norm of the whole coefficient vector as λ varies.

```
plot(fit_ridge, cex.lab = 1.5)
```



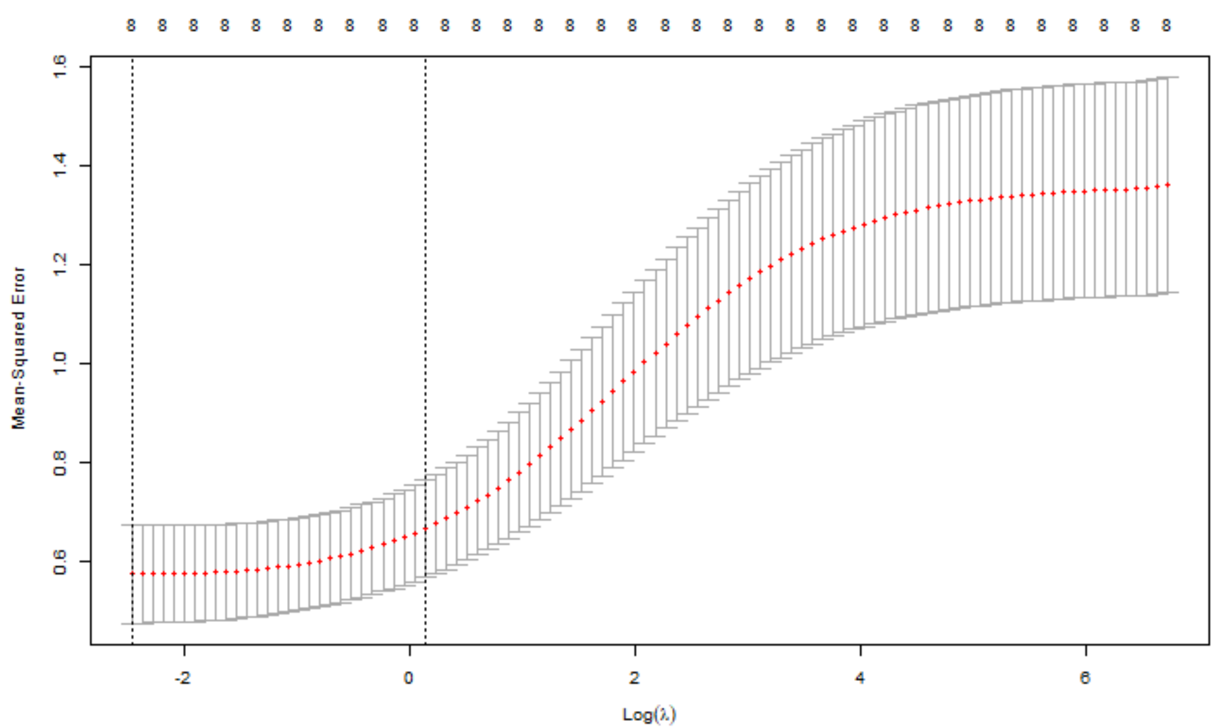
In order to select λ, **glmnet** provides routine performing cross-validatation.

```
cvfit_ridge <- cv.glmnet(X, y, alpha = 0)
cvfit_ridge
```

```
##
## Call:  cv.glmnet(x = X, y = y, alpha = 0)
##
## Measure: Mean-Squared Error
##     Which one do we choose? → The minimun value : to avoid overpenalization
##     Lambda Index Measure      SE Nonzero
## min 0.0843   100  0.5743 0.09977       8
## 1se 1.1412    72  0.6661 0.09889       8
```

Thus, we can visualise the results of the cross-validation plotting the MSE along the $\log(\lambda)$ values, including upper and lower standard deviation curves. The two marked values for λ correspond to the minimum value of the cross-validated MSE, and the λ value for which the cross-validated error is within one standard error of the minimum.

```
plot(cvfit_ridge)
```



Then, the ridge regression coefficients can be obtained for such values. For instance.

```
cvfit_ridge$lambda.min
```

```
## [1] 0.08434274
```

```
cvfit_ridge$lambda.1se
```

```
## [1] 1.141198
```

```r
coef(cvfit_ridge, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.477219879
## lcavol       0.511620675
## lweight      0.442406565
## age         -0.015205261   ( negative and small ? )
## lbph         0.095156600
## svi          0.690577706
## lcp         -0.038131799
## gleason      0.061518053
## pgg45        0.003457535   small. → 0
```

```r
coef(cvfit_ridge, s = "lambda.1se")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.405877986
## lcavol       0.250437271
## lweight      0.290478414
## age         -0.000877059
## lbph         0.049942613
## svi          0.432438952
## lcp          0.079357403
## gleason      0.085912851
## pgg45        0.002660706
```

LASSO regression

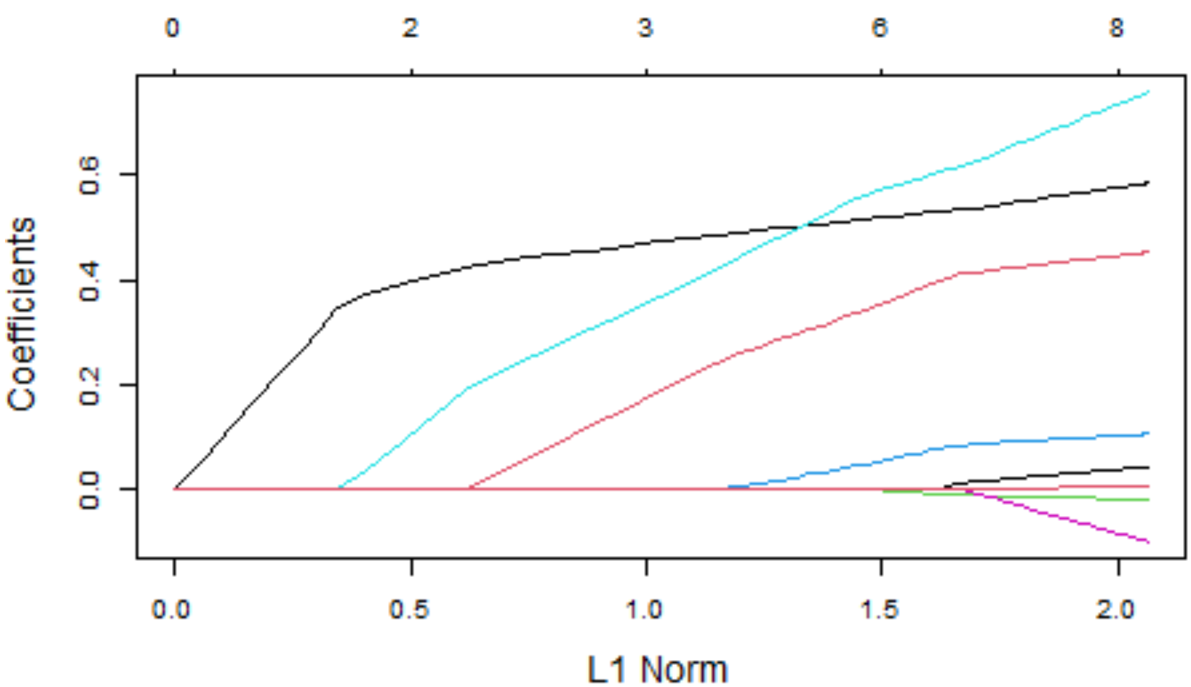
The LASSO regression allows to shrink a group of regression coefficients weakly linked to the outcome. In addition, due to the use of the ℓ_1 penalty it allows to perform variable selection. Note that in this case we can not express the estimator as a linear estimator since $2\mathbf{X}^\top\mathbf{X}\beta + 2\mathbf{X}^\top\mathbf{y} + \lambda\sum_{j=1}^p \text{sign}(\beta_j) = 0$ does not have explicit solution and it must be optimized numerically. In addition, it retains the same property of the ridge estimator: it is biased but has smaller variance than the LS estimator.

```
fit_lasso <- glmnet(X, y, alpha = 1)
```

As above we can inspect the percentage (of null) deviance explained (%dev) and the value of λ, as well as plotting the path of its coefficient against the ℓ_1-norm of the whole coefficient vector as λ varies.

```
print(fit_lasso)
```

```
plot(fit_lasso, cex.lab = 1.4)
```

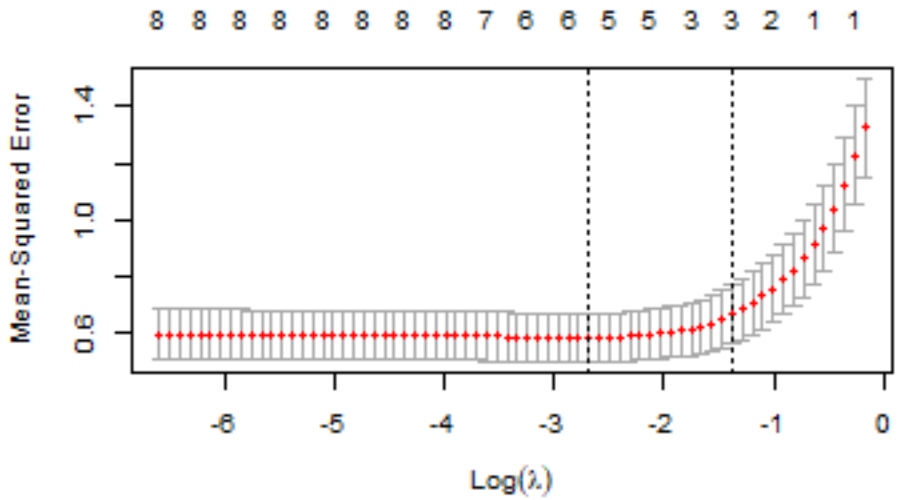


So, we can leverage on the cross-validation procedure in order to select λ.

```
cvfit_lasso <- cv.glmnet(X,y, alpha = 1)
cvfit_lasso
```

```
##
## Call:  cv.glmnet(x = X, y = y, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.06841    28  0.5835 0.08477       5
## 1se 0.25165    14  0.6666 0.10585       3
```

```r
plot(cvfit_lasso)
```




```r
round(t(coef(cvfit_lasso, s = "lambda.min")), 4)
```

```
## 1 x 9 sparse Matrix of class "dgCMatrix"
##    (Intercept) lcavol lweight age   lbph    svi lcp gleason  pgg45
## s1      0.4012 0.5133  0.3359   . 0.0458 0.5533   .       . 0.0013
```

```r
round(t(coef(cvfit_lasso, s = "lambda.1se")), 4)
```

```
## 1 x 9 sparse Matrix of class "dgCMatrix"
##    (Intercept) lcavol lweight age lbph    svi lcp gleason pgg45
## s1      1.5121 0.4464  0.0834   .    . 0.2721   .       .     .
```

In the following table, the estimated coefficients obtained via LS, Ridge and LASSO are reported

```r
res <- cbind(coef(fit.lin), coef(cvfit_ridge, s = "lambda.min"),
             coef(cvfit_lasso, s = "lambda.min"), coef(cvfit_lasso, s = "lambda.1se"))
colnames(res) <- c("LS", "RIDGE", "LASSOmin", "LASSO1se")
res
```

```
## 9 x 4 sparse Matrix of class "dgCMatrix"
##                       LS        RIDGE    LASSOmin   LASSO1se
## (Intercept)  0.669399027  0.477219879 0.401248829 1.51214665
## lcavol       0.587022881  0.511620675 0.513293944 0.44639279
## lweight      0.454460641  0.442406565 0.335940521 0.08341893
## age         -0.019637208 -0.015205261 .           .
## lbph         0.107054351  0.095156600 0.045754719 .
## svi          0.766155885  0.690577706 0.553349460 0.27207350
## lcp         -0.105473570 -0.038131799 .           .
## gleason      0.045135964  0.061518053 .           .
## pgg45        0.004525324  0.003457535 0.001341672 .
```

Extra: Evaluate model performance

```
set.seed(23)
idx_train <- sample(1 : nrow(Prostate), 0.90 * nrow(Prostate), replace = FALSE)
y_train <- y[idx_train]
y_test <- y[-idx_train]
X_train <- X[idx_train,]
X_test <- X[-idx_train,]
fit.lin <- lm(lpsa ~ ., data = Prostate[idx_train,])
pred_lm <- as.numeric(predict(fit.lin, newdata = Prostate[-idx_train,]))
cvfit_ridge <- cv.glmnet(X_train, y_train, alpha = 0)
cvfit_ridge
```

```
##
## Call:  cv.glmnet(x = X_train, y = y_train, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure      SE Nonzero
## min 0.0796   100  0.5429 0.09245       8
## 1se 1.1813    71  0.6340 0.08116       8
```

```
pred_ridge1se <- as.numeric(predict(cvfit_ridge, newx = X_test, s = "lambda.1se"))
pred_ridgemin <- as.numeric(predict(cvfit_ridge, newx = X_test, s = "lambda.min"))
cvfit_lasso <- cv.glmnet(X_train, y_train, alpha = 1)
cvfit_lasso
```

```
##
## Call:  cv.glmnet(x = X_train, y = y_train, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.00574    54  0.5392 0.08695       8
## 1se 0.19705    16  0.6194 0.07597       3
```

```
pred_lasso1se <- as.numeric(predict(cvfit_lasso, newx = X_test, s = "lambda.1se"))
pred_lassomin <- as.numeric(predict(cvfit_lasso, newx = X_test, s = "lambda.min"))
MSE <- function(y, pred_y){
  return(mean((y-pred_y)^2))
}
MSE_lm <- MSE(y_test, pred_lm)
MSE_ridge1se <- MSE(y_test, pred_ridge1se)
MSE_lasso1se <- MSE(y_test, pred_lasso1se)
MSE_ridgemin <- MSE(y_test, pred_ridgemin)
MSE_lassomin <- MSE(y_test, pred_lassomin)
c(MSE_lm, MSE_ridge1se, MSE_lasso1se)
```

```
## [1] 0.6470641 0.9206499 0.9218953
```

```
c(MSE_lm, MSE_ridgemin, MSE_lassomin)
```

```
## [1] 0.6470641 0.6395095 0.6350075
```

Exercise: Try to implement a one-fold cross validation and evaluate the model performances using LS, ridge and LASSO.

Spline functions

Splines are functions defined as piecewise polynomials of fixed degree. The points of connections of the polynomials are called knots. The idea is that, between two knots, the function $f(x)$ is a a said polynomial and such polynomials meets at the knots, with further constraints to achieve regularity constraints. For instance, given K knots $\zeta_1, \dots, \zeta_K$, a cubic (degree = 3) regression spline is expressed as

$$f(x; \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{k=1}^{K} \beta_{k+3}(x - \zeta_k)_+^3$$

with $(x - \zeta_k)_+^3 = \max(0, (x - \zeta_k)^3)$. We call spline basis functions the $h_j(x) = x^{j-1}$, $j = 1, \dots, 4$ and $h_j(x) = (x - \zeta_j)_+^3$, $j = 1, \dots, k$. However, different basis expansions can be used to define a spline function and in general

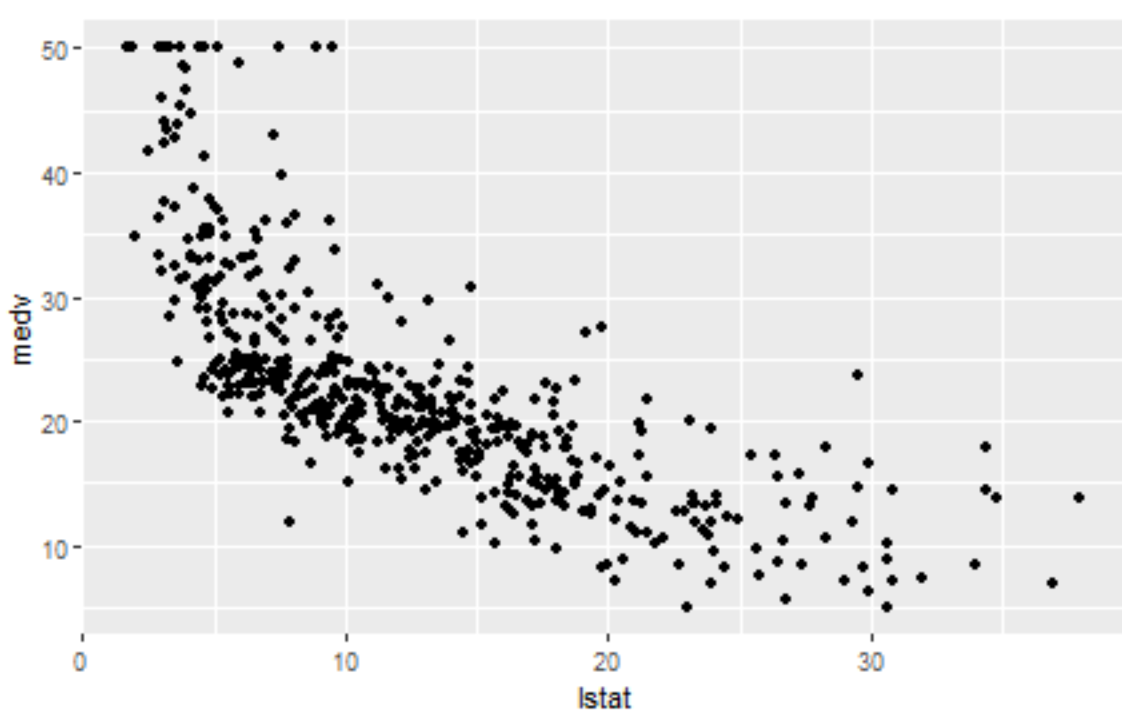
$$f(x; \boldsymbol{\beta}) = \sum_{k=0}^{K+d} \beta_k h_k(x)$$

We will adopt the B-spline which are built by means of the Cox–de Boor recursion formula.

Boston data example

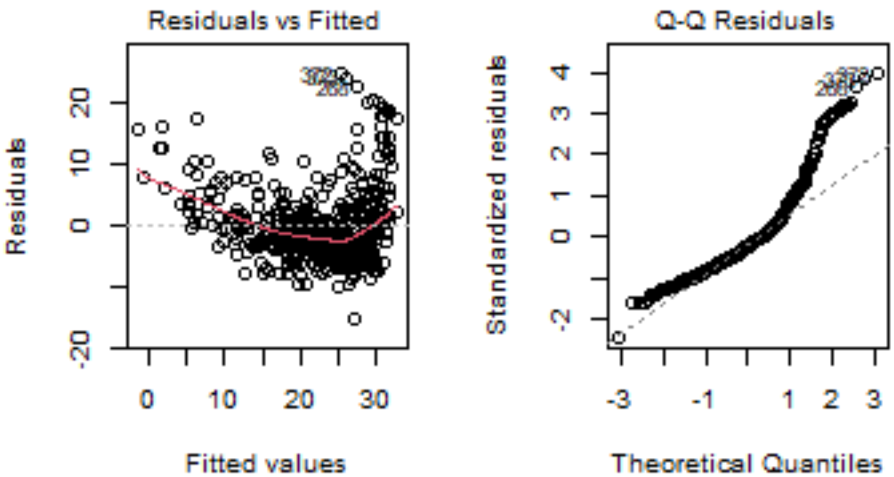
We use a naive application on `Boston` dataset available in the `MASS` package to compare the use of polynomials and splines. The dataset collects information on housing values in suburbs of Boston and in particular we are going to model the continuous variable `medv` (median value of owner-occupied homes in $1000s) using a simple linear model including the continuous variable `lstat` (lower status of the population) as a predictor.

```r
library(splines)
library(ggplot2)
library(MASS)
data(Boston) # From MASS package
ggplot(Boston, aes(lstat, medv)) + geom_point()
```



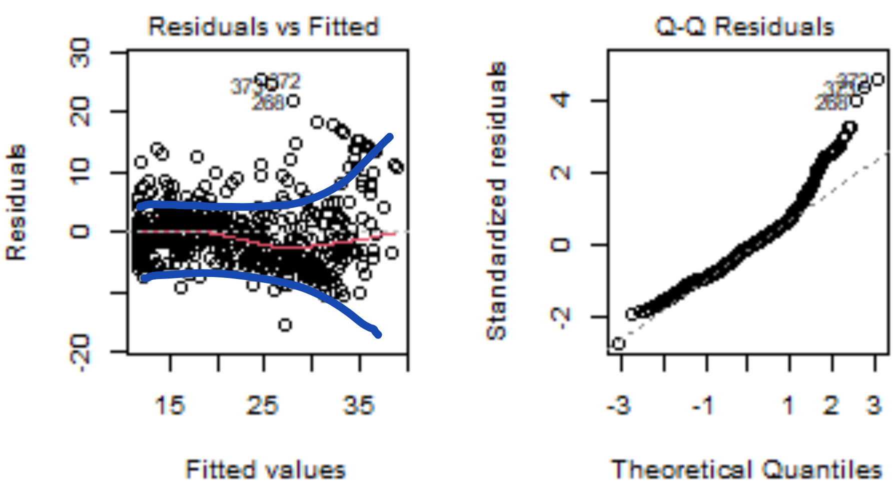
As first step, we fit a linear model and then we analyse the residuals.

```r
fit <- lm(medv ~ lstat, data = Boston)
par(mfrow = c(1, 2))
plot(fit, which = c(1, 2))
```



Then, we consider a quadratic term for lstat. There are two ways to achieve this (using or not orthogonal polynomials; see the help of *poly*)

```r
fit.poly2 <- lm(medv ~ lstat + I(lstat^2), data = Boston)
fit.poly2 <- lm(medv ~ poly(lstat, 2, raw = TRUE), data = Boston)
par(mfrow = c(1,2))
plot(fit.poly2,  which = c(1,2))
```

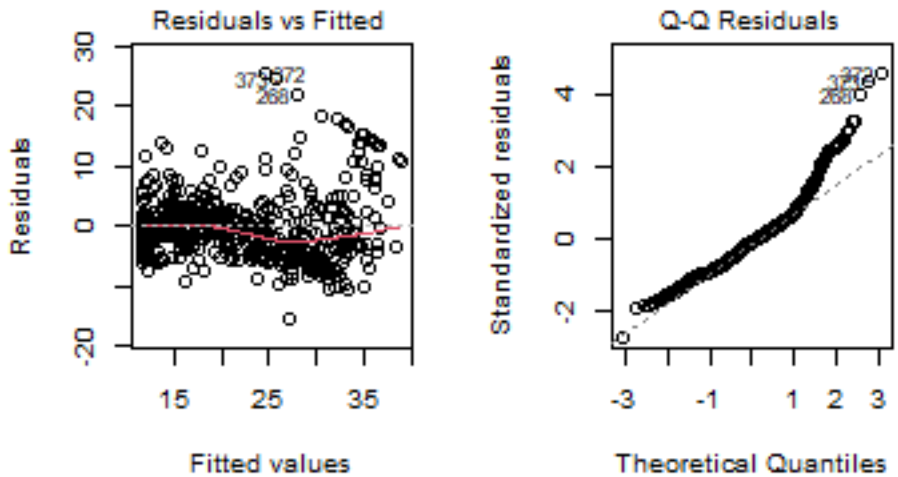


Then, we consider a a polynomial of degree 5 for lstat.

```
fit.poly5 <- lm(medv ~ poly(lstat, 5, raw = TRUE), data = Boston)
par(mfrow=c(1,2))
plot(fit.poly2, which = c(1,2))
```

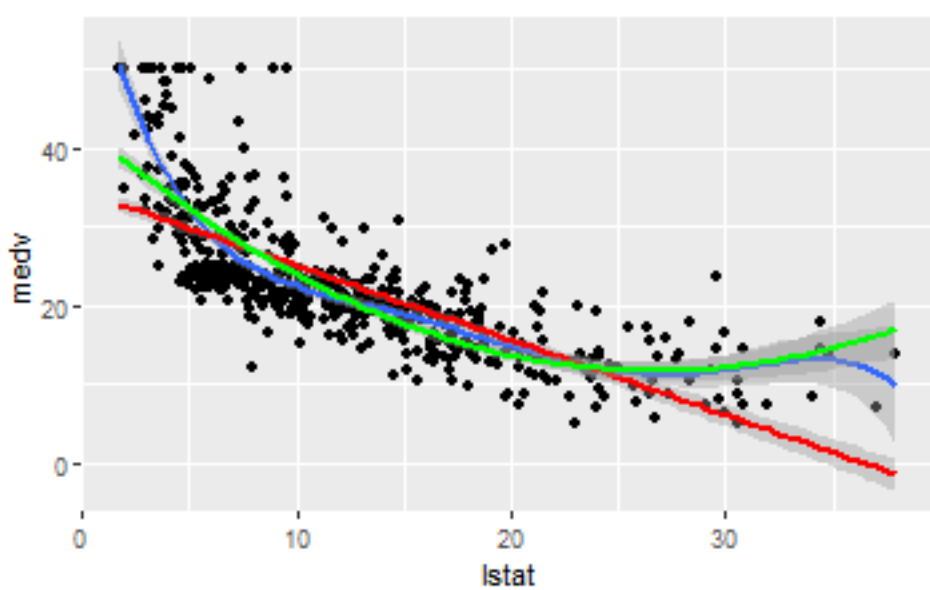
What?

Exactly. Run code and look at the plot.



Compare our fitted curve.

```
ggplot(Boston, aes(lstat, medv)) +
 geom_point()+
 stat_smooth(method = lm, formula = y ~ poly(x, 5, raw = TRUE))+
 stat_smooth(method = lm, formula = y ~ x, col = "red") +
 stat_smooth(method = lm, formula = y ~ poly(x, 2, raw = TRUE), col = "green")
```



Now we consider the cubic regression splines, namely we consider B-splines by means of **bs()**. We fix the (internal) knots at first, second and third quartile. However, consider that a similar specification could be obtained by specifying the degrees of freedom equal to 6 as an alternative to the specification of the knots

```
knots <- quantile(Boston$lstat)[2 : 4]
fit.spline <- lm(medv ~ bs(lstat, knots=knots), data = Boston)
summary(fit.spline)
```

0% 25% 50% 75% 100%

What?

This 3 make sense to be knots

df = K + 4 = 7 = 6 + 1

intercept

```
##
## Call:
## lm(formula = medv ~ bs(lstat, knots = knots), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.8071  -3.1502  -0.7389   2.1076  26.9529
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   50.628      2.582  19.608  < 2e-16 ***
## bs(lstat, knots = knots)1  -13.682      3.886  -3.521  0.00047 ***
## bs(lstat, knots = knots)2  -26.684      2.449 -10.894  < 2e-16 ***
## bs(lstat, knots = knots)3  -28.416      2.917  -9.743  < 2e-16 ***
## bs(lstat, knots = knots)4  -40.092      3.050 -13.144  < 2e-16 ***
## bs(lstat, knots = knots)5  -39.718      4.212  -9.431  < 2e-16 ***
## bs(lstat, knots = knots)6  -38.484      4.134  -9.308  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.206 on 499 degrees of freedom
## Multiple R-squared:  0.6833, Adjusted R-squared:  0.6795
## F-statistic: 179.5 on 6 and 499 DF,  p-value: < 2.2e-16
```

```
fit.spline <- lm(medv ~ bs(lstat, df = 6), data = Boston)
summary(fit.spline)
```

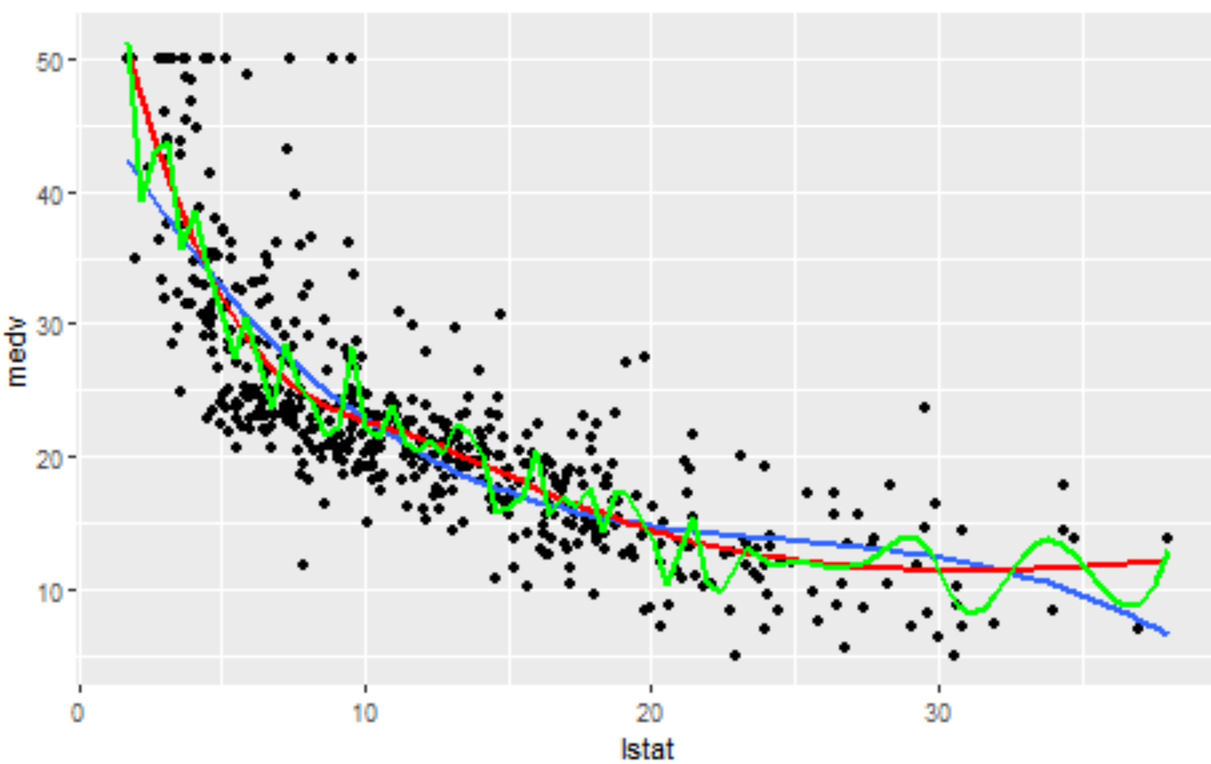
```
##
## Call:
## lm(formula = medv ~ bs(lstat, df = 6), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.8071  -3.1502  -0.7389   2.1076  26.9529
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            50.628      2.582  19.608  < 2e-16 ***
## bs(lstat, df = 6)1  -13.682      3.886  -3.521  0.00047 ***
## bs(lstat, df = 6)2  -26.684      2.449 -10.894  < 2e-16 ***
## bs(lstat, df = 6)3  -28.416      2.917  -9.743  < 2e-16 ***
## bs(lstat, df = 6)4  -40.092      3.050 -13.144  < 2e-16 ***
## bs(lstat, df = 6)5  -39.718      4.212  -9.431  < 2e-16 ***
## bs(lstat, df = 6)6  -38.484      4.134  -9.308  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.206 on 499 degrees of freedom
## Multiple R-squared:  0.6833, Adjusted R-squared:  0.6795
## F-statistic: 179.5 on 6 and 499 DF,  p-value: < 2.2e-16
```

$p \to 0 \ \forall$ params.

A graphical comparison is useful. Thus, we can plot the fitted lines comparing

- Model 1: polynomial of degree = 3 (BLUE) just 1 poly
- Model 2: cubic B-splines with df = 6 (RED) 4 polys of degree 3.
- Model 3: cubic B-splines with df = 100 (GREEN)

```r
ggplot(Boston, aes(lstat, medv)) +
 geom_point()+
 stat_smooth(method = lm, formula = y ~ poly(x, 3, raw = TRUE) , se = FALSE)+
 stat_smooth(method = lm, formula = y ~ bs(x, df = 6), col = "red", se = FALSE) +
 stat_smooth(method = lm, formula = y ~ bs(x, df = 100), col = "green", se = FALSE )
```



We can analyse the AIC

```r
c(AIC(fit.poly2), AIC(fit.spline), AIC(lm(medv ~ bs(lstat, df = 100), data = Boston)))
```

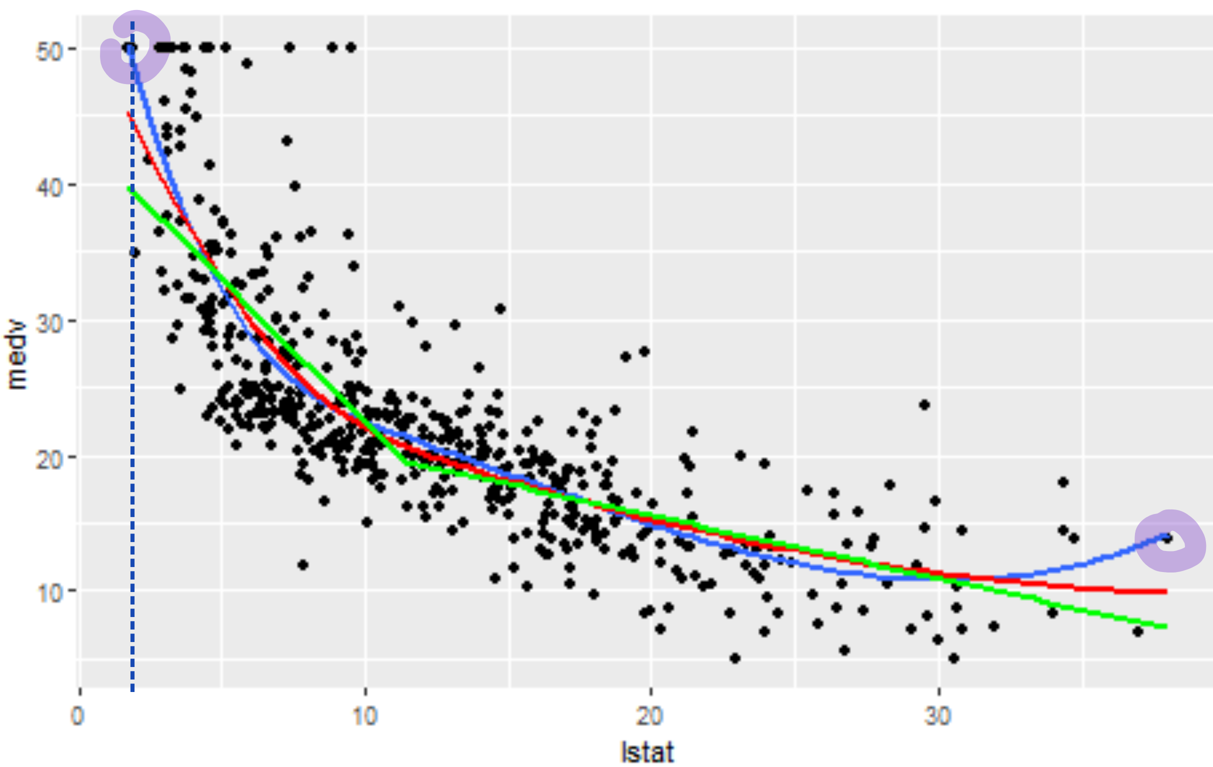
```
## [1] 3170.516 3114.610 3204.101
```

Clearly the benefit of using regression splines (w.r.t. selecting a suitable polynomial) vanishes as we are overparametrising the model.

To conclude, let us consider one knot on the median and plot the fitted lines comparing three different specification of spline degrees:

- Model 1: cubic B-splines (BLUE)
- Model 2: quadratic B-splines (RED)
- Model 3: linear B-splines (GREEN)

```
knot <- knots[2]
# knot <- median(Boston$lstat)
par(mfrow=c(1,2))
ggplot(Boston, aes(lstat, medv)) +
 geom_point()+
 stat_smooth(method = lm, formula = y ~ bs(x, knots = knot), se = FALSE)+
 stat_smooth(method = lm, formula = y ~ bs(x, knots = knot, degree = 2), col = "red", se = FALSE) +
 stat_smooth(method = lm, formula = y ~ bs(x, knots = knot, degree = 1), col = "green", se = FALSE )
```



Generalized additive models (GAM)

A generalized additive model (GAM) is a generalized linear model (GLM) in which the linear predictor is given by a user specified sum of smooth functions of the covariates plus a conventional parametric component of the linear predictor. GAMs can be stated

$$\eta_i = g(\mu_i) = f(x_{i1}, x_{i2}, \ldots, x_{ip}) = \sum_j f_j(x_i^j) + \sum_k s_k(x_i^k)$$

where the $f_j(\cdot)$'s denotes linear effects of x_i^j , while smooth effects of x_i^j are denoted with $s_k(\cdot)$. Here, we restrict the formulation to univariate effects, although the r.h.s. of the formula above could account for bivariate smooth effects and smooth factor interactions.

- η_i: i-th linear predictor
- $Y_i \sim \mathrm{EF}(\mu_i, \phi)$, with $\mu_i = EY_i$ and ϕ a scale parameter.

By using the (linear) basis spline expansion the smooth effects can be represented as

$$s_j(x_i^j) = \sum_k b_k^j(x_i^j)\alpha_k^j \ ,$$

where b_k^j are spline basis functions of a suitable dimension, while α_k^j are regression coefficients. Since also the linear components can be expressed in the usual way

$$\sum_j f_j(x_i^j) = \mathbf{x}_i^\top \boldsymbol{\gamma},$$

where $\mathbf{x}_i^\top$ is the i-th row of the model matrix the parametric components. Further a sum-to-zero constraint is generally adopted, since the effects are identifiable to within an intercept term. Thus, the linear predictor $\boldsymbol{\eta} = (\eta_i, \ldots, \eta_n)^\top$ collapses into the following specification

$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$$

where $\boldsymbol{\beta} = (\boldsymbol{\gamma}^\top, \boldsymbol{\alpha}^\top)^\top$ and X is the model matrix.

GAM models consider a penalized likelihood approach

$$\ell(\boldsymbol{\beta}) - \lambda R(\boldsymbol{s}) = \ell(\boldsymbol{\beta}) - \frac{1}{2}\boldsymbol{\beta}^\top \tilde{\mathbf{S}}^{\boldsymbol{\lambda}} \boldsymbol{\beta},$$

where $\boldsymbol{\lambda}$ vector of smoothing parameters, $R(\boldsymbol{s})$ is a measure of roughness, and $\tilde{\mathbf{S}}^{\boldsymbol{\lambda}}$ are positive semi-definite (padded with zeros). The complexity of the smooth effects is controlled during the fitting considering a quadratic penalty on $\boldsymbol{\alpha}$ (which corresponds to give a prior on $\boldsymbol{\alpha}$).

- if $\lambda_j \to \infty \Rightarrow$ **maximal smoothness**. The fitted curve $s(\cdot)$ is a straight line. The *effective number of parameters* associated to the predictor x_j is 1. No need for a smooth function.
- if $\lambda_j \to 0 \Rightarrow$ **no smoothness**. No penalty is considered and the *effective number of parameters* associated to the predictor x_j is greater than 1.

Fitting the model:

- Original backfitting algorithm by Hastie and Tibshirani (1986 https://www.jstor.org/stable/pdf/2245459.pdf) for additive models and a weighted version of it (penalized iteratively re-weighted least squares) for GAMs
- Nested iteration scheme (Wood, 2016 https://www.tandfonline.com/doi/full/10.1080/01621459.2016.1180986) : model fitting procedure alternates
    1. Regression coefficient estimation by Newton's optimisation of the penalised log-likelihood
    2. Smoothing parameter optimisation by maximising the Laplace approximate marginal likelihood (LAML). The latter is usually done done via Newton's methods

Additive model for modelling Ozone data

Let us consider the **ozone** dataset in the **faraway** package. This study aims to explore the relationship between atmospheric ozone concentration and meteorology in the Los Angeles Basin in 1976. A number of cases with missing variables have been removed for simplicity. The data frame reports 330 observations on the following 10 variables.

- **O3**: Ozone concentration (ppm), at Sandbug AFB.
- **vh**: Vandenburg 500 millibar height (in) visibility (miles)
- **wind**: wind speed (mph)
- **humidity**: humidity (percent)
- **temp**: Sandburg AFB temperature (Farheneit) (note that in the original paper are reported in Celsius)
- **ibh**: inversion base height (ft.)
- **dpg** Daggett pressure gradient (mmhg)
- **ibt**: inversion base temperature (Farheneit)
- **vis**: visibility (miles)
- **doy**: day of the year

According to the original study (https://www.jstor.org/stable/pdf/2288473.pdf) we consider the log transformation of the outcome.

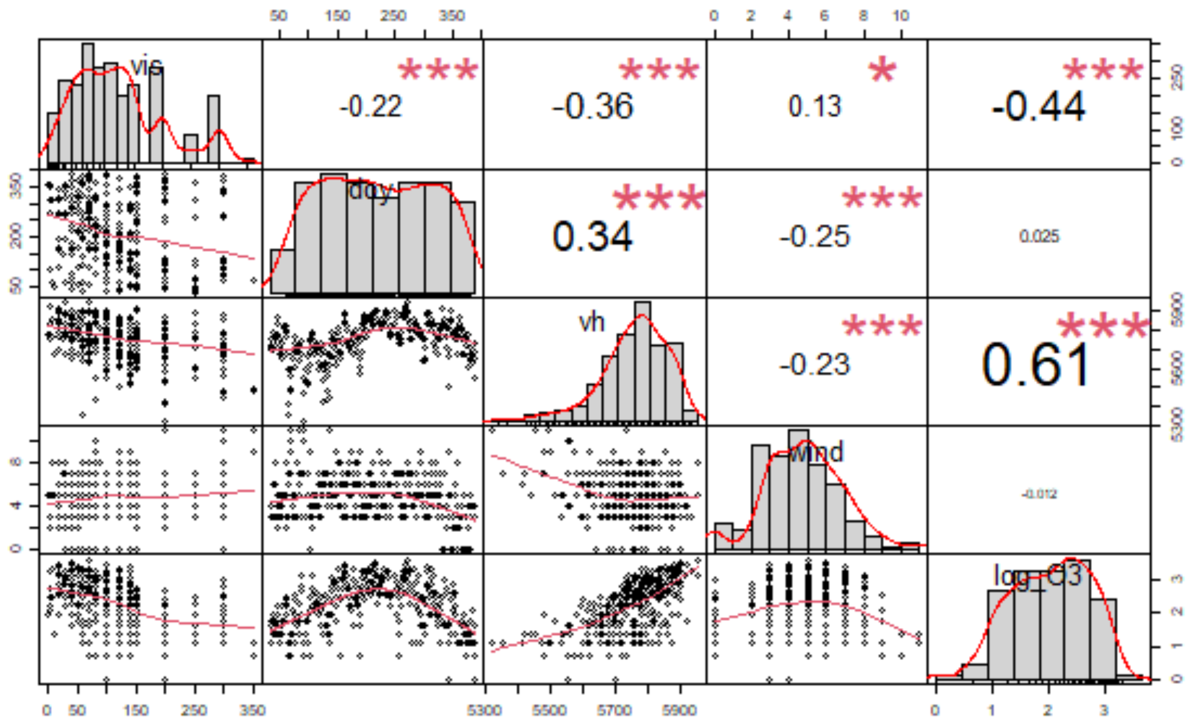
```
library(mgcv)
library(PerformanceAnalytics)
library(faraway)
data(ozone)
ozone$log_O3 <- log(ozone$O3)
summary(ozone)
```

```
##        O3              vh            wind           humidity
##  Min.   : 1.00   Min.   :5320   Min.   : 0.000   Min.   :19.00
##  1st Qu.: 5.00   1st Qu.:5690   1st Qu.: 3.000   1st Qu.:47.00
##  Median :10.00   Median :5760   Median : 5.000   Median :64.00
##  Mean   :11.78   Mean   :5750   Mean   : 4.848   Mean   :58.13
##  3rd Qu.:17.00   3rd Qu.:5830   3rd Qu.: 6.000   3rd Qu.:73.00
##  Max.   :38.00   Max.   :5950   Max.   :11.000   Max.   :93.00
##       temp            ibh             dpg              ibt
##  Min.   :25.00   Min.   : 111.0   Min.   :-69.00   Min.   :-25.0
##  1st Qu.:51.00   1st Qu.: 877.5   1st Qu.: -9.00   1st Qu.:107.0
##  Median :62.00   Median :2112.5   Median : 24.00   Median :167.5
##  Mean   :61.75   Mean   :2572.9   Mean   : 17.37   Mean   :161.2
##  3rd Qu.:72.00   3rd Qu.:5000.0   3rd Qu.: 44.75   3rd Qu.:214.0
##  Max.   :93.00   Max.   :5000.0   Max.   :107.00   Max.   :332.0
##       vis             doy             log_O3
##  Min.   :  0.0   Min.   : 33.0   Min.   :0.000
##  1st Qu.: 70.0   1st Qu.:120.2   1st Qu.:1.609
##  Median :120.0   Median :205.5   Median :2.303
##  Mean   :124.5   Mean   :209.4   Mean   :2.213
##  3rd Qu.:150.0   3rd Qu.:301.8   3rd Qu.:2.833
##  Max.   :350.0   Max.   :390.0   Max.   :3.638
```

✓ not skewed

In a first attempt, let us consider only the explanatory variables: temperature, the visibility, wind speed and the day of the year.

```
chart.Correlation(ozone[, c("vis", "doy", "vh", "wind", "log_O3")])
```



Model 1

Given $\mu_i = E[Y_i]$, a simple example is:

$$\mu_i = \beta_0 + \sum_j s_j(x_i^j),\ i = 1, \dots, n,$$

where the dependent variable $Y_i \sim \mathsf{Normal}(\mu_i, \sigma^2)$ and s_j are smooth functions of the covariates x^j.

Let us start fitting the model including the temperature, the visibility, wind speed and the day of the year. By default the family function is **family = gaussian(link = identity)**

```
gamfit <- gam(log_O3 ~ s(temp) + s(vis) + s(doy) + s(wind), data = ozone)
summary(gamfit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log_O3 ~ s(temp) + s(vis) + s(doy) + s(wind)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.21297    0.02042   108.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df      F  p-value
## s(temp) 4.338  5.365 36.216  < 2e-16 ***
## s(vis)  6.439  7.565  8.192  < 2e-16 ***
## s(doy)  5.629  6.814 15.965  < 2e-16 ***
## s(wind) 2.024  2.587  6.906 0.000515 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.754   Deviance explained = 76.8%
## GCV = 0.14614  Scale est. = 0.13754   n = 330
```

λ

From the summary we can see the distinction between parametric coefficients table and a summary for the smooth terms. By default we built smooth effects with ten basis functions and accounting for the common intercept the number of parameters in the model are $1 + 9 + 9 + 9 + 9 = 37$

Analysing the first column of the table it seems that smooth effects are sensible for all the variables included (moderately for the wind effect). Looking at the smoothing parameters we can see the λs for temperature, vis, doy are close to zero, while the λ_j associated with the wind speed is moderately large, suggesting that likely such an effect could be modelled using a linear effect.

```
length(gamfit$coef)
```

```
## [1] 37
```

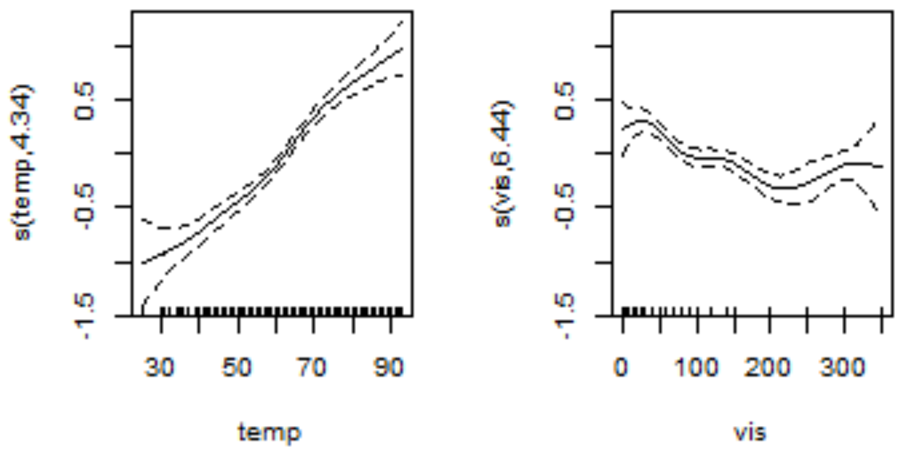
```
gamfit$sp
```

```
##     s(temp)      s(vis)      s(doy)     s(wind)
##  0.18486375  0.06189805  0.07602479 18.02601191
```

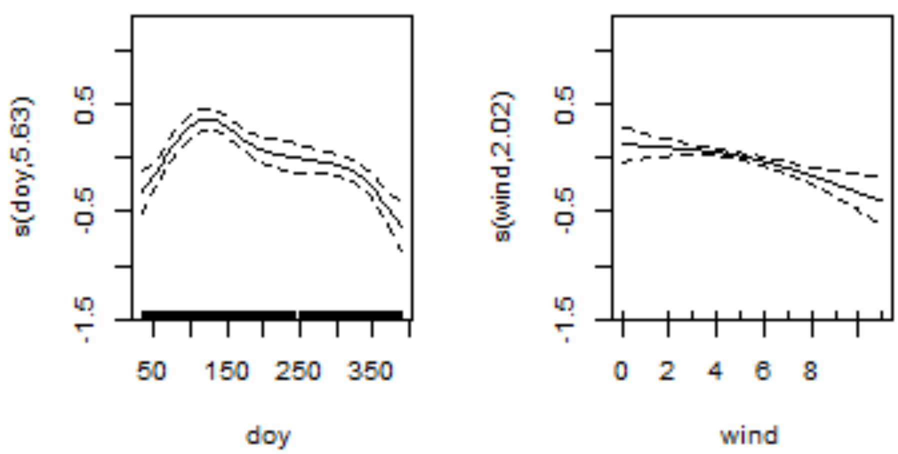
Visualisation and diagnostics of a gam model

Visualizing the fitted smooth effects $\hat{s}(x^j)$.

```r
par(mfrow=c(1,2))
plot(gamfit, select = 1)
plot(gamfit, select = 2)
```




```r
par(mfrow=c(1,2))
plot(gamfit, select = 3)
plot(gamfit, select = 4)
```

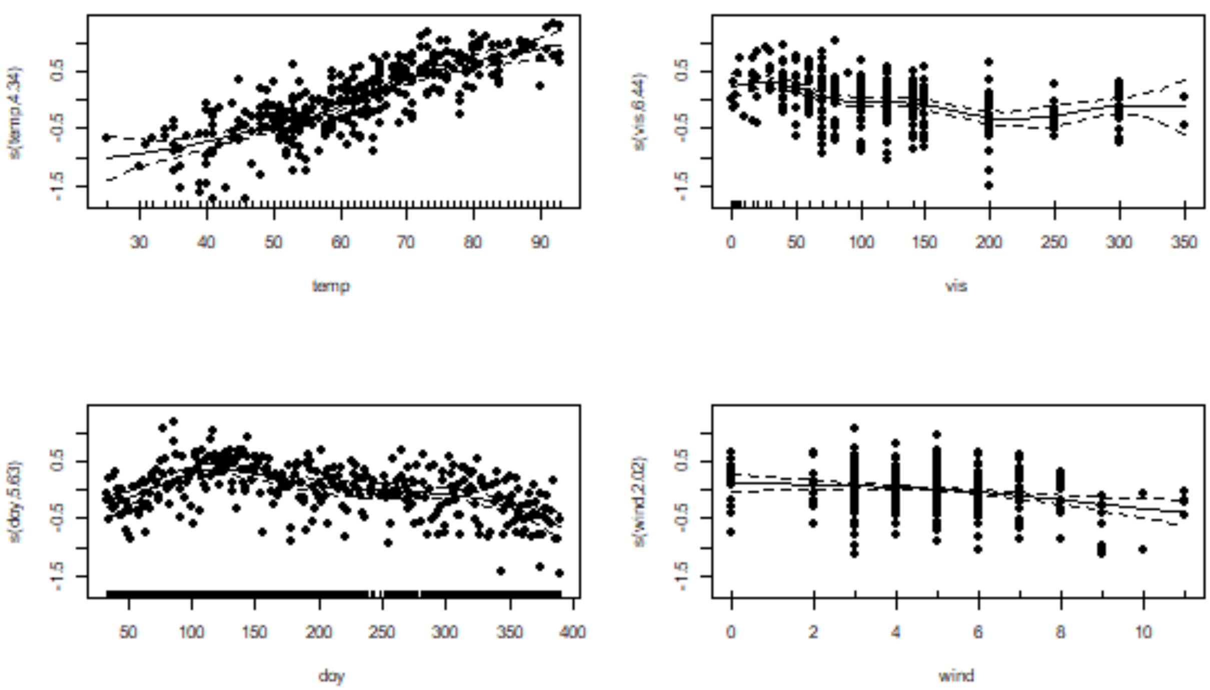


Diagnostic plot involve the representation of the smooth function and the partial residuals defined as:

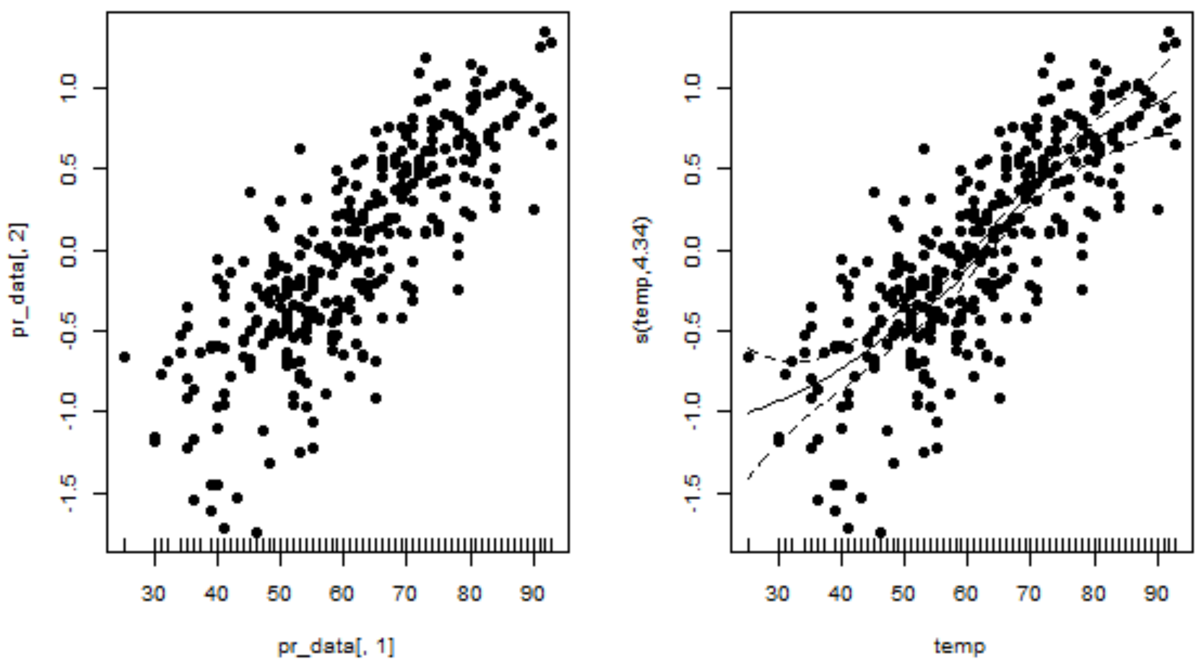
$$\hat{\epsilon}_{ij}^{part} = \hat{s}_j(x_{ij}) + \hat{\epsilon}_i^P$$

where $\hat{\epsilon}^P$ are the Pearson residuals of the model. Looking at this plot we are interested in noticing non linearity or wiggle behavior of the smooth function and if the partial residuals are evenly distributed around the function. To justify use of splines

```
plot(gamfit, residuals = TRUE, pch = 19, pages = 1)
```




```
# Part of the plot obtained via gam routines
pr_data <- cbind(ozone$temp, residuals(gamfit, type="pearson") +
                gam(log_O3 ~ s(temp) + s(vis) + s(doy) + s(wind), data = ozone,
                    fit=FALSE)$X[,2:10] %*%coef(gamfit)[2:10])
par(mfrow = c(1,2))
plot(pr_data[,1],pr_data[,2], pch = 19)
rug(pr_data[,1])
plot(gamfit, residuals = TRUE, pch = 19, select = 1)
```



Exercise: Try to add the missing information into the left plot, that is align with the right plot.

Including smooth and linear effects

From the previous results we can compare some models. The following models consider

- Model 2: linear effect for wind and smooth effects for the others
- Model 3: linear effect for all the covariates (fitted using the gam function)
- Model 4 = Model 3, albeit it is fitted using the glm function

```r
# Model 2
gamfit2 <- gam(log(O3)~s(temp) + s(vis) + s(doy) + wind, data = ozone)
summary(gamfit2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ s(temp) + s(vis) + s(doy) + wind
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.42921    0.05981  40.617  < 2e-16 ***
## wind        -0.04460    0.01159  -3.849 0.000144 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df      F p-value
## s(temp) 4.451  5.497 36.059  <2e-16 ***
## s(vis)  6.206  7.354  8.042  <2e-16 ***
## s(doy)  5.716  6.910 17.019  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##                         < before
## R-sq.(adj) =  0.752   Deviance explained = 76.5%   < before
## GCV = 0.14722  Scale est. = 0.13902   n = 330
```

```r
# Model 3
glmfit_viagam <- gam(log(O3) ~ temp + vis + doy + wind, data = ozone)
summary(glmfit_viagam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ temp + vis + doy + wind
##
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.3876365  0.1490553   2.601  0.00973 **
## temp         0.0391508  0.0018246  21.457  < 2e-16 ***
## vis         -0.0017879  0.0003311  -5.399 1.29e-07 ***
## doy         -0.0014802  0.0002450  -6.041 4.20e-09 ***
## wind        -0.0123246  0.0117254  -1.051  0.29399   → not significant
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.666   Deviance explained =   67%   < Model 1 , 2
## GCV = 0.1901  Scale est. = 0.18722   n = 330
```

glm and gam are equivalent for lineal models

```r
# Model 4
glmfit <- glm(log(O3) ~ temp + vis + doy + wind, data = ozone)
summary(glmfit)
```

```
##
## Call:
## glm(formula = log(O3) ~ temp + vis + doy + wind, data = ozone)
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.3876365  0.1490553   2.601  0.00973 **
## temp         0.0391508  0.0018246  21.457  < 2e-16 ***
## vis         -0.0017879  0.0003311  -5.399 1.29e-07 ***
## doy         -0.0014802  0.0002450  -6.041 4.20e-09 ***
## wind        -0.0123246  0.0117254  -1.051  0.29399      -> not significant
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1872198)   < 1
##
##     Null deviance: 184.252  on 329  degrees of freedom
## Residual deviance:  60.846  on 325  degrees of freedom
## AIC: 390.56
##
## Number of Fisher Scoring iterations: 2
```

Finally we can compare the models by using the AIC metric. Obviously the last two models provide the same AIC, while it is clear that considering the smooth effects for temperature, visibility and the day of the year allows to obtain a better model. Considering a smooth effect for the wind effect provides benefit of marginal significance.

```r
cbind(AIC(gamfit, gamfit2, glmfit_viagam, glmfit))
```

```
##                     df      AIC
## gamfit        20.43023 302.6597
## gamfit2       19.37318 305.2065   <- Maybe this one, is simpler.
## glmfit_viagam  6.00000 390.5554
## glmfit         6.00000 390.5554
```

Extending the model considering all the available covariates

Let us include all the available information in the linear predictor and let us start considering smooth effects for all the covariates.

```
gamfit_ext <- gam(log(O3)~s(vh)+s(wind)+s(humidity)+s(temp)+s(ibh) +
                    s(dpg)+s(ibt)+s(vis)+s(doy),data=ozone)
summary(gamfit_ext)
```

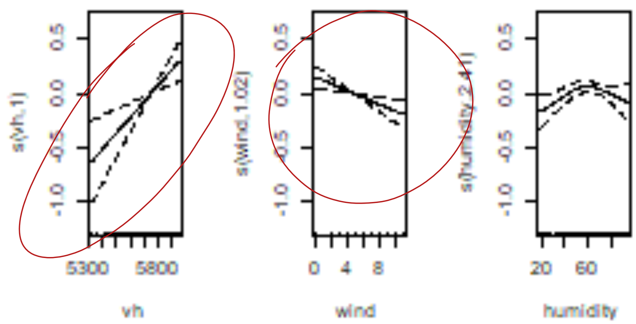
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ s(vh) + s(wind) + s(humidity) + s(temp) + s(ibh) +
##     s(dpg) + s(ibt) + s(vis) + s(doy)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.21297    0.01717   128.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df      F  p-value
## s(vh)       1.000  1.000 10.780  0.00115 **  ?
## s(wind)     1.021  1.040  8.980  0.00292 **  ?
## s(humidity) 2.406  3.025  2.567  0.05192 .   ?
## s(temp)     3.801  4.740  4.161  0.00155 **
## s(ibh)      2.774  3.393  5.341  0.00107 **
## s(dpg)      3.285  4.176 14.268  < 2e-16 ***
## s(ibt)      1.000  1.000  0.495  0.48225      x
## s(vis)      5.477  6.635  6.054 3.35e-06 ***
## s(doy)      4.612  5.738 25.200  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.826   Deviance explained =   84%
## GCV = 0.10569  Scale est. = 0.097247  n = 330
```

```
gamfit_ext$sp
```

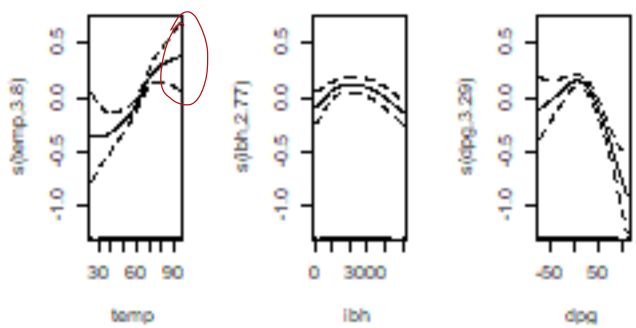
```
##        s(vh)      s(wind)  s(humidity)      s(temp)       s(ibh)       s(dpg)
## 4.669441e+08 1.759526e+03 1.666873e+00 3.008595e-01 1.148304e+00 7.540753e-01
##       s(ibt)       s(vis)       s(doy)
## 6.936647e+08 1.372183e-01 1.618457e-01
```

From the above results, it seems that the variables **vh, wind, ibt** can be included in the model as linear effects. So let's check the fitted smooth effects.

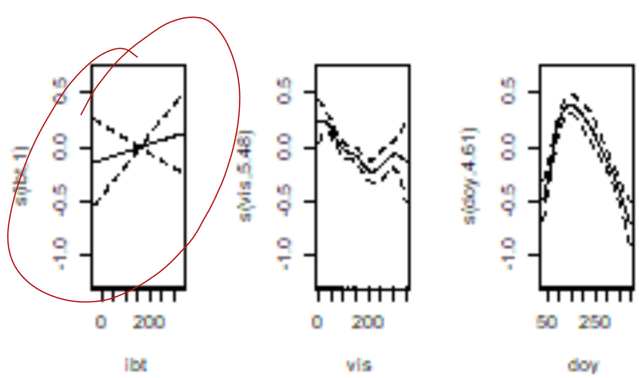
```r
par(mfrow=c(1,3))
for(j in 1:3) plot(gamfit_ext, select=j)
```




```r
for(j in 1:3) plot(gamfit_ext, select=j+3)
```




```r
for(j in 1:3) plot(gamfit_ext, select=j+6)
```



Combining the summary output and the plots above, we can consider linear effects for wind, vh and ibt.

Further, take a look to the residuals plot (not reported here)

```r
plot(gamfit_ext, residuals = TRUE, pch = 19)
```

Then, we fit the model considering

- Extended model 2: Linear effects for wind, vh and ibt and smooth effects for the remaining ones.
- Extended model 3: Linear effects for wind, vh and ibt, ibh, humidity, and smooth effects for the remaining ones.
- Extended model 4: Linear effects all the predictors

```r
# Extended model 2
gamfit_ext2 <- gam(log(O3) ~ vh + wind + s(humidity) + s(temp) + s(ibh) +
                     s(dpg) + ibt + s(vis) + s(doy), data=ozone)
summary(gamfit_ext2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ vh + wind + s(humidity) + s(temp) + s(ibh) + s(dpg) +
##     ibt + s(vis) + s(doy)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.0874893  2.4699664  -2.465  0.01427 *
## vh           0.0014501  0.0004384   3.308  0.00105 **
## wind        -0.0311454  0.0101294  -3.075  0.00230 **
## ibt          0.0006986  0.0010388   0.673  0.50177
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df      F  p-value
## s(humidity) 2.398  3.017  2.476 0.059093 .
## s(temp)     3.811  4.753  4.081 0.001778 **
## s(ibh)      2.761  3.378  5.431 0.000953 ***
## s(dpg)      3.354  4.259 14.320  < 2e-16 ***
## s(vis)      4.559  5.627  6.572 3.91e-06 ***
## s(doy)      4.571  5.692 25.618  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.826   Deviance explained = 83.9%
## GCV = 0.10575  Scale est. = 0.097591  n = 330
```

< Model 1

```r
# Extended model 3
gamfit_ext3 <- gam(log(O3) ~ vh + wind + humidity + s(temp) + ibh +
                     s(dpg) + ibt + s(vis) + s(doy), data=ozone)
summary(gamfit_ext3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ vh + wind + humidity + s(temp) + ibh + s(dpg) + ibt +
##     s(vis) + s(doy)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.504e+00  2.531e+00  -2.569 0.010658 *
## vh           1.530e-03  4.489e-04   3.409 0.000739 ***
## wind        -2.954e-02  1.038e-02  -2.846 0.004730 **
## humidity     1.000e-03  1.640e-03   0.610 0.542337
## ibh         -3.813e-05  2.331e-05  -1.636 0.102958
## ibt          6.146e-04  1.060e-03   0.580 0.562526
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df      F  p-value
## s(temp) 4.962  6.068  3.792  0.00124 **
## s(dpg)  3.904  4.923 13.126  < 2e-16 ***
## s(vis)  4.578  5.650  6.681 2.89e-06 ***
## s(doy)  4.227  5.309 23.786  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.813   Deviance explained = 82.6%      < Model 4, 2
## GCV = 0.11268  Scale est. = 0.1046    n = 330
```

```r
# Extended model 4 (GLM)
gamfit_ext4 <- gam(log(O3) ~ vh + wind + humidity + temp + ibh +
                     dpg + ibt + vis + doy, data=ozone)
summary(gamfit_ext4)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ vh + wind + humidity + temp + ibh + dpg + ibt + vis +
##     doy
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.301e-01  2.684e+00   0.235  0.81453
## vh          -1.244e-05  4.908e-04  -0.025  0.97980
## wind        -8.403e-03  1.226e-02  -0.685  0.49357
## humidity     5.084e-03  1.713e-03   2.968  0.00322 **
## temp         2.993e-02  4.527e-03   6.611 1.60e-10 ***
## ibh         -8.495e-05  2.687e-05  -3.161  0.00172 **
## dpg          5.470e-04  1.026e-03   0.533  0.59414
## ibt          4.902e-04  1.237e-03   0.396  0.69221
## vis         -8.395e-04  3.410e-04  -2.462  0.01434 *
## doy         -1.021e-03  2.522e-04  -4.049 6.47e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.709   Deviance explained = 71.7%
## GCV = 0.16808  Scale est. = 0.16298   n = 330
```

< Previous Models.

Let's compare the models by means of AIC.

```r
cbind(AIC(gamfit_ext,gamfit_ext2,gamfit_ext3,gamfit_ext4))
```

```
##                   df      AIC
## gamfit_ext  27.37511 194.6941
## gamfit_ext2 26.45411 195.0168
## gamfit_ext3 24.67099 216.2586
## gamfit_ext4 11.00000 349.6878
```

← select

Model 2

From the AIC table above, we can see the importance of including smooth effects for temp, dpg, vis, doy. Further, also including the smooth effects for humidity and ibh achieves better performances, while considering a smooth effect for vh, wind and ibt does not provide any benefit.

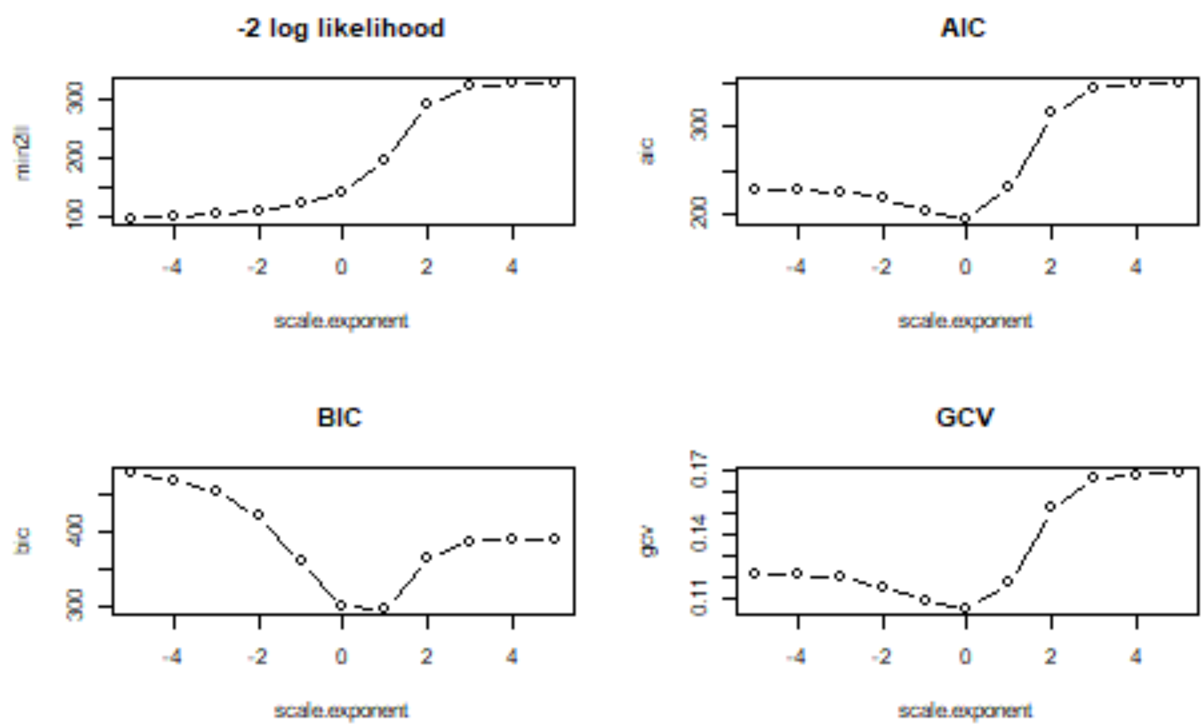
Selecting the smoothing parameter

The smoothing parameter (vector) $\boldsymbol{\lambda}$ may be estimated by several methods, such as CV, AIC, BIC. . . By default, gam function consider the method = "GCV.Cp" and scale = 0, meaning that the λs are estimated via GCV, or via AIC if the family is Binomial or Poisson. To undertand the machinery consider the following lines of code

```r
sp <- gamfit_ext$sp # smoothing parameter obtained for the extended model 1

tuning.scale<-c(1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4, 1e5)
scale.exponent <- log10(tuning.scale)
n.tuning <- length(tuning.scale)
min2ll <- aic <- bic <- gcv <- rep(NA, n.tuning)
for (i in 1:n.tuning) {
  gamfit_opt<-gam(log(O3) ~ s(vh) + s(wind) + s(humidity) + s(temp) + s(ibh)+
                  s(dpg) + s(ibt) + s(vis) + s(doy), data = ozone, sp = tuning.scale[i] * sp)
  min2ll[i]<- -2 * logLik(gamfit_opt)
  aic[i] <- AIC(gamfit_opt)
  bic[i] <- BIC(gamfit_opt)
  gcv[i] <- gamfit_opt$gcv.ubre
}

par(mfrow = c(2, 2))
plot(scale.exponent, min2ll, type = "b",main = "-2 log likelihood")
plot(scale.exponent, aic, type = "b", main = "AIC")
plot(scale.exponent, bic, type = "b", main = "BIC")
plot(scale.exponent, gcv, type = "b", main = "GCV")
```



Why do we use BIC? Is it because it is a convex function?

```r
min.bic <- 1e100
opt.tuning.scale <- NULL
for (i in 1 :n.tuning) {
  if (bic[i] < min.bic) {
    min.bic <- bic[i]
    opt.tuning.scale <- tuning.scale[i]
  }
}
opt.sp <- opt.tuning.scale * sp  # Smoothing parameter selected via BIC

gamobj <- gam(log(O3) ~ s(vh) + s(wind) + s(humidity) + s(temp) + s(ibh) +
                s(dpg) + s(ibt) + s(vis) + s(doy),
                data = ozone, sp = opt.sp)
summary(gamobj)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ s(vh) + s(wind) + s(humidity) + s(temp) + s(ibh) +
##     s(dpg) + s(ibt) + s(vis) + s(doy)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.21297    0.01838   120.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df      F  p-value
## s(vh)       1.000  1.000  3.650   0.0570 .
## s(wind)     1.002  1.004  5.862   0.0161 *
## s(humidity) 1.369  1.646  2.349   0.0574 .
## s(temp)     2.098  2.649  9.676 2.43e-05 ***
## s(ibh)      1.657  2.030  6.658   0.0012 **
## s(dpg)      1.727  2.192 13.232 1.47e-06 ***
## s(ibt)      1.000  1.000  0.037   0.8472
## s(vis)      3.171  3.965  7.033 2.51e-05 ***
## s(doy)      2.462  3.196 27.265  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.801   Deviance explained =   81%
## GCV = 0.11734  Scale est. = 0.11147    n = 330
```

```r
summary(gamfit_ext)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(O3) ~ s(vh) + s(wind) + s(humidity) + s(temp) + s(ibh) +
##     s(dpg) + s(ibt) + s(vis) + s(doy)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.21297    0.01717   128.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df      F  p-value
## s(vh)       1.000  1.000 10.780  0.00115 **
## s(wind)     1.021  1.040  8.980  0.00292 **
## s(humidity) 2.406  3.025  2.567  0.05192 .
## s(temp)     3.801  4.740  4.161  0.00155 **
## s(ibh)      2.774  3.393  5.341  0.00107 **
## s(dpg)      3.285  4.176 14.268  < 2e-16 ***
## s(ibt)      1.000  1.000  0.495  0.48225
## s(vis)      5.477  6.635  6.054 3.35e-06 ***
## s(doy)      4.612  5.738 25.200  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.826   Deviance explained =   84%
## GCV = 0.10569  Scale est. = 0.097247  n = 330
```

```r
AIC(gamobj, gamfit_ext)
```

```
##                  df      AIC
## gamobj     17.48649 230.5532
## gamfit_ext 27.37511 194.6941
```

The results show clearly a better smoothing parameter selection by considering the GCV score than the BIC criterion.

What model should we choose?

Hourly electricity demand modelling

The following dataset is extracted from the load forecasting track of the GefCom2014 competition (Hong et al., 2016, https://www.sciencedirect.com/science/article/pii/S0169207016000133), a global energy forecasting competition. For illustrative purposes, we only consider the hourly loads (in MW) at 5 p.m. (load_h17) on a daily basis as the outcome variable. The dataset, spanning the period from 2005/01/02 to 2011/11/30, includes

- **doy**: the day of the year
- **dow**: the day of the week
- **temp95_h**: the exponentially smoothed temperature
- **load24_h**: the loads of the previous day.

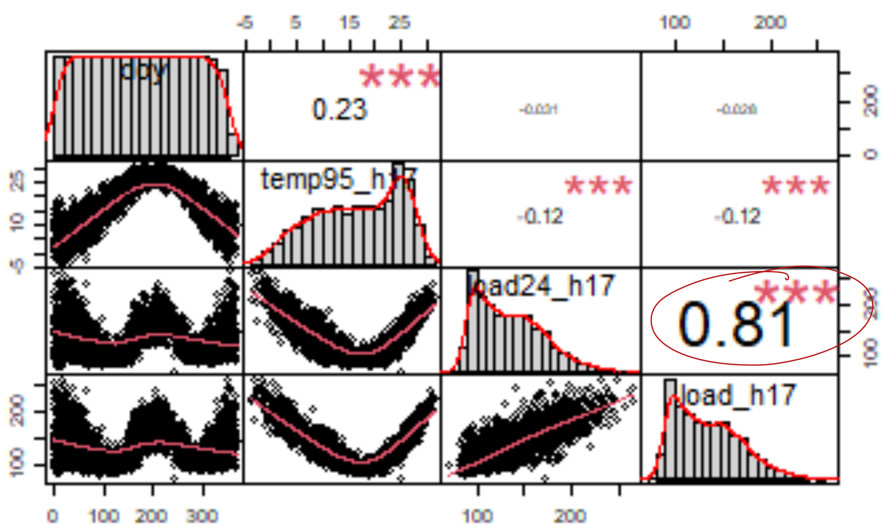
```
load("GEF14_d4.RData")
str(GEF14_d4[, c("dow", "doy", "temp95_h17", "load24_h17", "load_h17")])
```

```
## 'data.frame':    2520 obs. of  5 variables:
##  $ dow       : Factor w/ 7 levels "0","1","2","3",..: 2 3 4 5 6 7 1 2 3 4 ...
##  $ doy       : num  3 4 5 6 7 8 9 10 11 12 ...
##  $ temp95_h17: num  11.4 13.4 16.4 16.4 18.1 ...
##  $ load24_h17: num  107.9 102.2 93.9 84.1 85.7 ...
##  $ load_h17  : num  102.2 93.9 84.1 85.7 81.8 ...
```

```
summary(GEF14_d4[, c("temp95_h17", "load24_h17", "load_h17")])
```

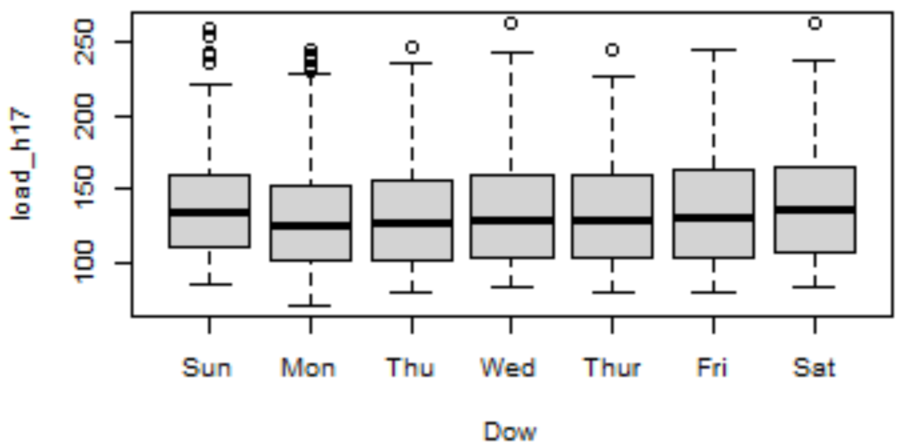
```
##    temp95_h17       load24_h17       load_h17
##  Min.   :-3.719   Min.   : 71.2   Min.   : 71.2
##  1st Qu.:10.022   1st Qu.:104.6   1st Qu.:104.6
##  Median :17.620   Median :130.7   Median :130.7
##  Mean   :16.819   Mean   :135.6   Mean   :135.6
##  3rd Qu.:24.505   3rd Qu.:159.8   3rd Qu.:159.8
##  Max.   :31.486   Max.   :262.8   Max.   :262.8
```

```
chart.Correlation(GEF14_d4[, c("doy", "temp95_h17", "load24_h17", "load_h17")])
```



What does this mean?

```r
plot(load_h17 ~ factor(dow, labels= c("Sun", "Mon", "Thu", "Wed", "Thur", "Fri", "Sat")),
     xlab = "Dow", data = GEF14_d4)
```



Let us consider the following linear predictor specification, including smooth effects for the day of the year and the (exponentially smoothed) temperature. We also consider a linear effect for the lagged demand. Obviously **the dow is a categorical variable, so it must be included as linear (in a dummy fashion)**.

```r
gam1_gef14 <- gam(load_h17 ~ dow + load24_h17 + s(temp95_h17) + s(doy), data = GEF14_d4)
summary(gam1_gef14)$p.table
```

```
##                Estimate Std. Error     t value     Pr(>|t|)
## (Intercept) 176.7903748 2.26511179  78.0492936 0.000000e+00
## dow1         -5.7026426 0.84130924  -6.7782955 1.512145e-11
## dow2         -5.6563975 0.84955865  -6.6580424 3.400501e-11
## dow3         -4.7450294 0.84685319  -5.6031310 2.336075e-08
## dow4         -3.8717857 0.84488968  -4.5825932 4.818505e-06
## dow5         -2.7380652 0.84606355  -3.2362405 1.226996e-03
## dow6          0.4536524 0.84396404   0.5375258 5.909524e-01
## load24_h17   -0.2800522 0.01568926 -17.8499285 3.639116e-67
```

dowo

numerical v.

```r
summary(gam1_gef14)$s.table
```

```
##                    edf   Ref.df         F p-value
## s(temp95_h17) 7.893269 8.693837 591.94429       0
## s(doy)        8.518643 8.937398  60.89799       0
```

D = 89,9%

Due to the high edf in the table for the smooth effects (close to 9, the maximum achievable number considering a spline with ten basis) we can try to increase the number of basis functions. Finally, we compare all the models as usually, also including the model without smooth effects.

```r
gam2_gef14 <- gam(load_h17 ~ dow + load24_h17 +
                    s(temp95_h17, k = 20) + s(doy, k =20), data = GEF14_d4)
summary(gam2_gef14)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## load_h17 ~ dow + load24_h17 + s(temp95_h17, k = 20) + s(doy,
##     k = 20)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 177.07759    2.25727  78.448  < 2e-16 ***
## dow1         -5.69543    0.83779  -6.798 1.32e-11 ***
## dow2         -5.66833    0.84585  -6.701 2.55e-11 ***
## dow3         -4.71939    0.84323  -5.597 2.42e-08 ***
## dow4         -3.85598    0.84134  -4.583 4.81e-06 ***
## dow5         -2.71837    0.84251  -3.227  0.00127 **
## dow6          0.44803    0.84021   0.533  0.59392
## load24_h17   -0.28222    0.01564 -18.050  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                  edf Ref.df      F p-value
## s(temp95_h17)  8.962  11.12 465.13  <2e-16 ***
## s(doy)        15.430  17.55  33.08  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.899   Deviance explained = 90.1%
## GCV = 127.54  Scale est. = 125.9     n = 2520
```

```r
glm_gef14 <- gam(load_h17 ~ dow + load24_h17 +temp95_h17 + doy, data = GEF14_d4)
summary(glm_gef14)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## load_h17 ~ dow + load24_h17 + temp95_h17 + doy
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.051096   2.267669  11.929  < 2e-16 ***
## dow1        -5.228808   1.534654  -3.407 0.000667 ***
## dow2         1.715175   1.538261   1.115 0.264953
## dow3         1.581850   1.536189   1.030 0.303239
## dow4         0.426131   1.536660   0.277 0.781565
## dow5         2.166677   1.536702   1.410 0.158678
## dow6         4.182448   1.537194   2.721 0.006557 **
## load24_h17   0.808800   0.011711  69.061  < 2e-16 ***
## temp95_h17  -0.121417   0.050515  -2.404 0.016307 *
## doy          0.001329   0.004043   0.329 0.742431
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.661   Deviance explained = 66.2%
## GCV = 425.56  Scale est. = 423.87    n = 2520
```

```r
AIC(gam1_gef14, gam2_gef14, glm_gef14)
```

```
##                  df      AIC
## gam1_gef14 25.41191 19386.12
## gam2_gef14 33.39245 19371.04
## glm_gef14  11.00000 22408.01
```

Due to the time series nature of the data and the forecasting problem, in order to evaluate model performance in a validation set it is better to consider a strategy which is known as rolling origin forecasting procedure. See https://otexts.com/fpp3/tscv.html for further details.

GAM for diabetic retinopathy

The **wesdr** data frame contains 669 observations on the following variables.

- **dur**: Duration of diabetes at baseline, in years.
- **gly**: Percent of glycosylated hemoglobin at baseline.
- **bmi** Body mass index at baseline.
- **ret** Binary indicator of retinopathy progression at first follow-up.

It is of interest consider possible non-linear relationship between the covariates and the logit probability of retinopathy progression.

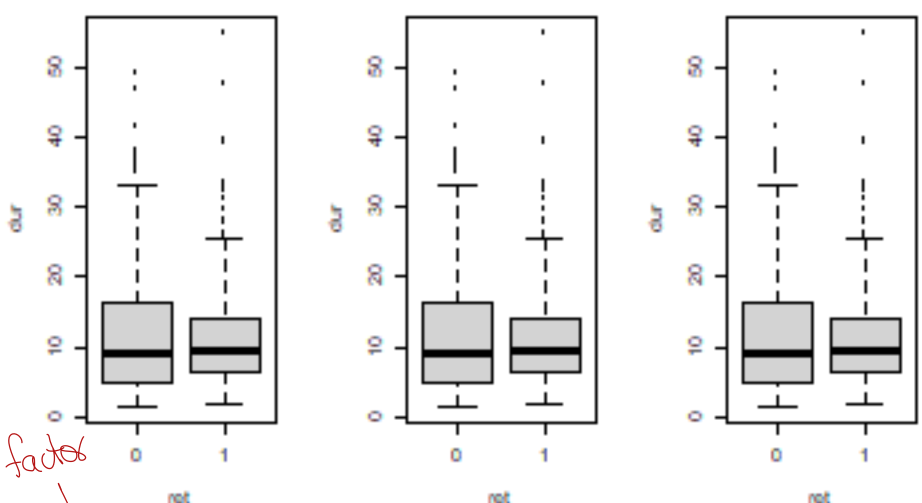
```
library(gss)
data(wesdr)
str(wesdr)
```

```
## 'data.frame':    669 obs. of  4 variables:
##  $ dur: num  10.3 9.9 15.6 26 13.8 31.1 2.6 39.6 7.3 8.3 ...
##  $ gly: num  13.7 13.5 13.8 13 11.1 11.3 15.1 13.4 14.6 15.4 ...
##  $ bmi: num  23.8 23.5 24.8 21.6 24.6 24.6 19.3 30.4 24.8 20.7 ...
##  $ ret: num  0 0 0 1 1 1 0 1 1 1 ...
```

```
wesdr$ret <- as.factor(wesdr$ret)
par(mfrow = c(1,3))
plot(dur ~ ret, data = wesdr)
plot(dur ~ ret, data = wesdr)
plot(dur ~ ret, data = wesdr)
```

→ classification

same plot.



factor

```
gam_log <- gam(ret ~ s(dur) + s(gly) + s(bmi), family = binomial(), data = wesdr)
summary(gam_log)
```

```
##
## Family: binomial
## Link function: logit
##
```

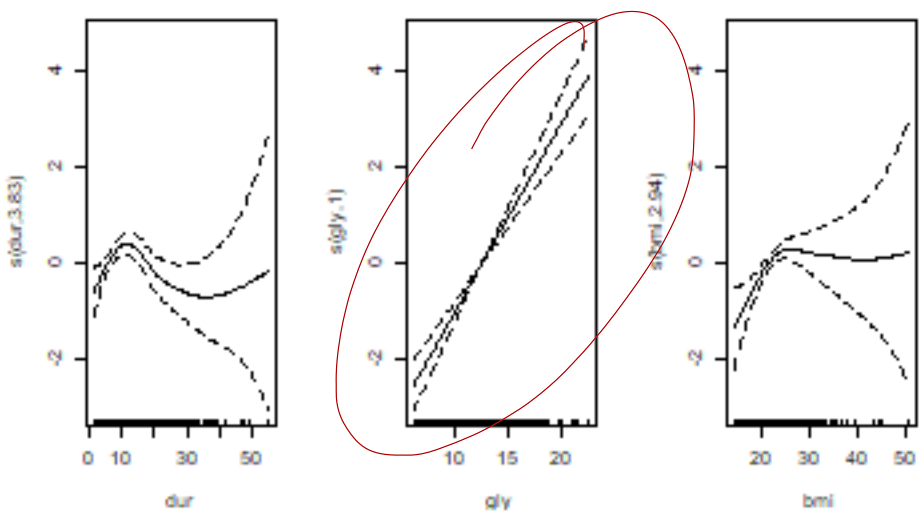
```
## Formula:
## ret ~ s(dur) + s(gly) + s(bmi)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.41911    0.08925  -4.696 2.65e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df Chi.sq p-value
## s(dur) 3.827  4.764  16.12 0.00585 **
## s(gly) 1.000  1.000  89.97 < 2e-16 ***
## s(bmi) 2.937  3.724  14.22 0.00536 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.216   Deviance explained = 17.8%
## UBRE = 0.14207  Scale est. = 1         n = 669
```

?

?

poor model.

```r
par(mfrow = c(1,3))
for(j in 1 : 3) {
  plot(gam_log, select = j)
}
```



```
gam_log2 <- gam(ret ~ s(dur) + gly + s(bmi), family = binomial(), data = wesdr)
summary(gam_log2)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## ret ~ s(dur) + gly + s(bmi)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.27670    0.52848  -9.985   <2e-16 ***
## gly          0.38880    0.04099   9.485   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df Chi.sq p-value
## s(dur) 3.827  4.764  16.12 0.00585 **
## s(bmi) 2.937  3.724  14.22 0.00536 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.216   Deviance explained = 17.8%
## UBRE = 0.14207  Scale est. = 1         n = 669
```

Since edf of gly = 1 the models are practically the same.

```
AIC(gam_log, gam_log2)
```

```
##                df      AIC
## gam_log  8.763676 764.0425
## gam_log2 8.763549 764.0423
```

Clearly the effect of **gly** can be considered as linear.

Exercise

1- Simulate some Poisson data with the `gamSim` function contained in the `mgcv` package (Use the default arguments `eg=1` for the Gu and Wahba 4 univariate term example and `scale=0.1`).

2- Fit a Gam and print the results. Interpret the fit.

3- Produce some plots for each x_j plotted against $s(x_j)$, and comment. Do you maybe drop some covariates?

4- Compute the AIC between your final gam and a glm.