STAT 260 - NONPARAMETRIC STATISTICS
Ying Sun
Statistics (STAT) Program
Computer, Electrical and Mathematical Sciences & Engineering (CEMSE) Division
King Abdullah University of Science and Technology (KAUST)

HOMEWORK
IV

Henrique Aparecido Laureano

Spring Semester 2018

Contents

Problem 1

This question is about illustrating the problems with polynomial bases. First run

```
# <r code> =================================================================== #
set.seed(1)                              # setting the "seed" to have always the same data
x <- sort(runif(40)*10)**.5                                             # generating x
y <- sort(runif(40))**.1                                                # generating y
# </r code> ================================================================== #
```

to simulate some apparently innocuous x, y data.

```
# <r code> =================================================================== #
par(mar = c(4, 4, 2, 2) + .1) ; plot(x, y)                              # plotting ...
# </r code> ================================================================== #
```

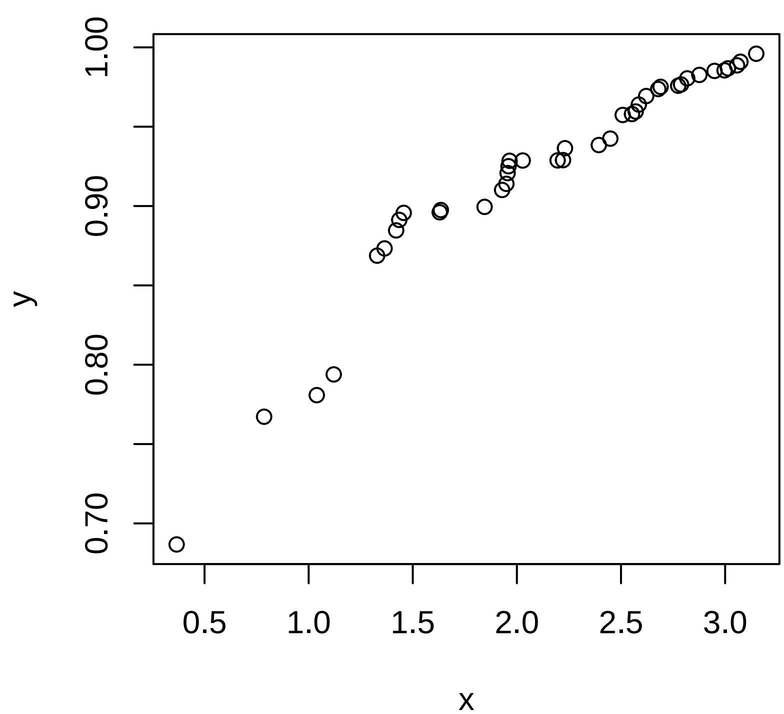


Figure 1: Some apparently innocuous x, y data.

(a)

Fit 5th and 10th order polynomials to the simulated data using, e.g., `lm(y ~ poly(x, 5))`**.**

Solution:

```
# <r code> ================================================================== #
poly5 <- lm(y ~ poly(x, 5)) ; poly10 <- lm(y ~ poly(x, 10))
# </r code> ================================================================= #
```

□

(b)

Plot the x, y data, and overlay the fitted polynomials. (Use the `predict` function to obtain predictions on a fine grid over the range of the x data: only predicting at the data fails to illustrate the polynomial behavior adequately).

Solution:

```
# <r code> ================================================================== #
grid <- seq(min(x), max(x), length = 200)      # 5 times more points (40 * 5 = 200)
par(mar = c(4, 4, 2, 2) + .1) ; plot(x, y)  # graphical definition & plotting data
                                      # overlaying the 5th order fitted polynomial
lines(grid, predict(poly5, data.frame(x = grid)), col = 2, lwd = 2)
                                     # overlaying the 10th order fitted polynomial
lines(grid, predict(poly10, data.frame(x = grid)), col = "#0080ff", lwd = 2)
# </r code> ================================================================= #
```

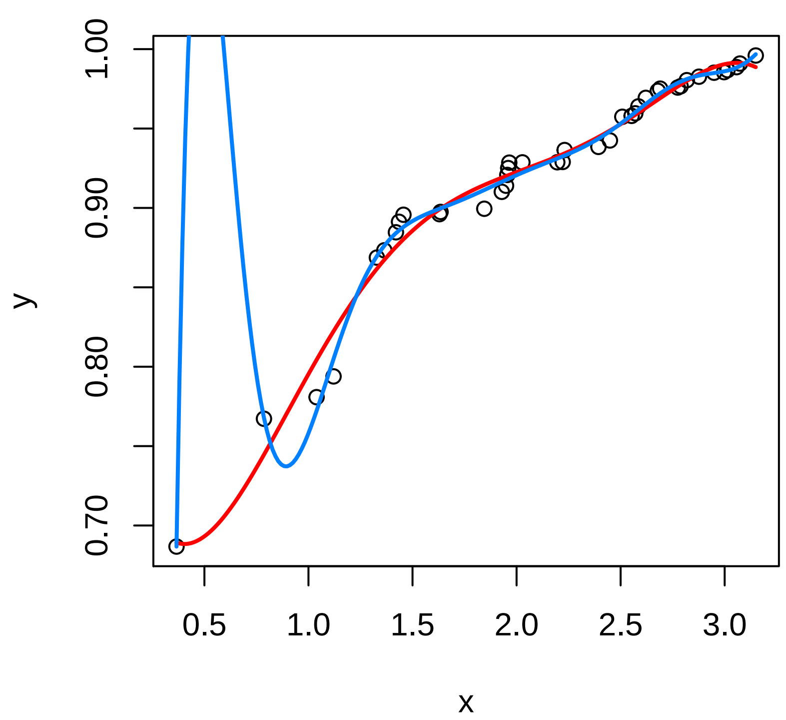


Figure 2: Data with overlay of the fitted polynomials (5th order in red and 10th order in blue).

We see how the 10th order fitted polynomial chase the data and how he lost himself between the initial points of x because we don't have data in that interval.

□

(c)

One particularly simple basis for a cubic regression spline is $b_2(x) = x$ and $b_{j+2}(x) = | x - x_j^* |^3$ for $j = 1, \dots, q-2$, where q is the basis dimension, and the x_j^* are knot locations. Use this basis to fit a rank 11 cubic regression spline to the x, y data (using `lm` and evenly spaced knots).

Solution:

```
# <r code> =================================================================== #
rank <- 11                                             # rank of the cubic spline
x_j <- ( ( 1:(rank - 2) / (rank - 1) )*10 )**.5                  # defining x_{j}
basis <- function(x, x_j) abs(x - x_j)**3                  # defining simple basis
           # constructing the formula for the basis of the cubic regression spline
fm <- paste0("basis(x, x_j[", 1:(rank - 2), "])", collapse = "+")
fm <- paste("y ~ x +", fm)
cs <- lm(formula(fm))        # fitting the model, cs: "c"ubic regression "s"pline
# </r code> ================================================================== #
```

□

(d)

Overlay the predicted curve according to the spline model, onto the existing x, y plot, and consider which basis you would rather use.

Solution:

```
# <r code> =================================================================== #
par(mar = c(4, 4, 2, 2) + .1) ; plot(x, y)  # graphical definition & plotting data
                                      # overlaying the 5th order fitted polynomial
lines(grid, predict(poly5, data.frame(x = grid)), col = 2, lwd = 2)
                                     # overlaying the 10th order fitted polynomial
lines(grid, predict(poly10, data.frame(x = grid)), col = "#0080ff", lwd = 2)
                                # overlaying the predicted cubic regression spline curve
lines(grid, predict(cs, data.frame(x = grid)), col = 3, lwd = 2)
# </r code> ================================================================== #
```

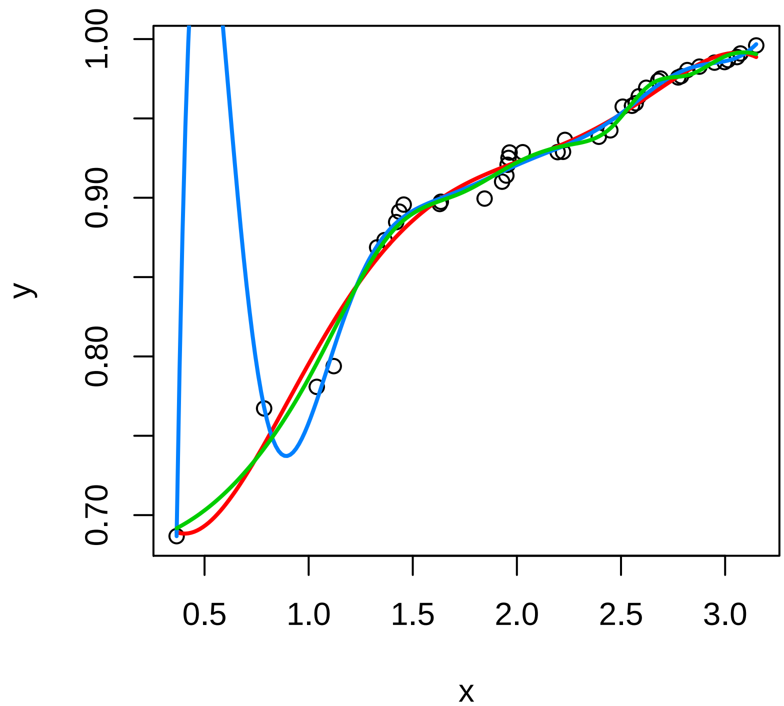


Figure 3: Data with overlay of the fitted polynomials (5th order in red and 10th order in blue) and the predicted cubic regression spline curve, in green.

The results with the 5th order polynomial and with the rank 11 cubic spline are quite similar, but with the cubic spline the fit looks more smooth.

$\square$

Problem 2

Show that the β minimizing $\parallel \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \parallel^2 + \boldsymbol{\lambda}\boldsymbol{\beta}^\top \mathbf{S}\boldsymbol{\beta}$ is given by $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X} + \boldsymbol{\lambda}\mathbf{S})^{-1}\mathbf{X}^\top\mathbf{y}$.

<u>Solution:</u>

$$\begin{aligned}
\parallel \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \parallel^2 + \boldsymbol{\lambda}\boldsymbol{\beta}^\top \mathbf{S}\boldsymbol{\beta} \\
(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \boldsymbol{\lambda}\boldsymbol{\beta}^\top \mathbf{S}\boldsymbol{\beta} \\
\mathbf{y}^\top\mathbf{y} - 2\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{y} + \boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\lambda}\boldsymbol{\beta}^\top \mathbf{S}\boldsymbol{\beta} \\
\mathbf{y}^\top\mathbf{y} - 2\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{y} + \boldsymbol{\beta}^\top(\mathbf{X}^\top\mathbf{X} + \boldsymbol{\lambda}\mathbf{S})\boldsymbol{\beta},
\end{aligned}$$

Taking the derivative with respect to $\boldsymbol{\beta}$ and setting to zero:

$$
\begin{aligned}
-2\mathbf{X}^\top\mathbf{y} + 2(\mathbf{X}^\top\mathbf{X} + \boldsymbol{\lambda}\mathbf{S})\hat{\boldsymbol{\beta}} &= \mathbf{0} \\
(\mathbf{X}^\top\mathbf{X} + \boldsymbol{\lambda}\mathbf{S})\hat{\boldsymbol{\beta}} &= \mathbf{X}^\top\mathbf{y} \\
\hat{\boldsymbol{\beta}} &= (\mathbf{X}^\top\mathbf{X} + \boldsymbol{\lambda}\mathbf{S})^{-1}\mathbf{X}^\top\mathbf{y}.
\end{aligned}
$$

□

Problem 3

Let X be an $n \times p$ model matrix, S a $p \times p$ penalty matrix, and B any matrix such that $\mathbf{B}^\top\mathbf{B} = \mathbf{S}$. If $\tilde{\mathbf{X}} = [\mathbf{X}^\top, \mathbf{B}^\top]^\top$ is an augmented model matrix, show that the sum of the first n elements on the leading diagonal of $\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top$ is $\mathrm{tr}\{\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}\mathbf{X}^\top\}$.

Solution:

$$
\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}} = [\mathbf{X}^\top\mathbf{B}^\top]\begin{bmatrix}\mathbf{X}\\ \mathbf{B}\end{bmatrix} = \mathbf{X}^\top\mathbf{X} + \mathbf{B}^\top\mathbf{B} = \mathbf{X}^\top\mathbf{X} + \mathbf{S}
$$

$$
\begin{aligned}
\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top &= \begin{bmatrix}\mathbf{X}\\ \mathbf{B}\end{bmatrix}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}[\mathbf{X}^\top\mathbf{B}^\top] \\
&= \begin{bmatrix}\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}\mathbf{X}^\top & \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}\mathbf{B}^\top \\ \mathbf{B}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}\mathbf{X}^\top & \mathbf{B}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}\mathbf{B}^\top\end{bmatrix}.
\end{aligned}
$$

The upper left $n \times n$ submatrix of $\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top$ is $\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}\mathbf{X}^\top$. Therefore, the sum of the first n elements on the leading diagonal is the $\mathrm{tr}\{\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \mathbf{S})^{-1}\mathbf{X}^\top\}$.

□

Problem 4

Read Section 4.2.4 and Section 4.3 of the textbook. The additive model of section 4.3 can equally well be estimated as a mixed model.

(a)

Write a function which converts the model matrix and penalty returned by `tf.XD` into mixed model form. Hint: because of the constraints the penalty null space is of dimension 1 now, leading to a slight modification of D_+.

Solution:

First taking the function `tf.XD` (and dependent functions) from the book (sections 4.2.1 and 4.3.1)

```
# <r code> =================================================================== #
                                # producing constrained versions of X_{j} and D_{J}
tf.XD <- function(x, xk, cmx = NULL, m = 2) {  # get X and D subject to constraint
  nk = length(xk)                                        # number of knots x_{k}
  X = tf.X(x,xk)[ , -nk]                                  # basis model matrix X
  D = diff(diag(nk), differences = m)[ , -nk]       # square root penalty matrix D
  if (is.null(cmx)) cmx = colMeans(X)   # values to subtract from the columns of X
  X = sweep(X, 2, cmx)                          # subtracting cmx from columns of X
  list(X = X, D = D, cmx = cmx)                               # returning objects
}
   # taking a sequence of knots and an array of x values to produce a model matrix
   #                                               for a piecewise linear function
tf.X <- function(x, xj) {   # tf basis matrix given data x and knot sequence x_{j}
  nk = length(xj) ; n = length(x)                                       # lengths
  X <- matrix(NA, n, nk)                          # creating empty model matrix X
  for (j in 1:nk) X[ , j] = tf(x, xj, j)                  # filling model matrix X
  X                                                     # returning model matrix X
}
                                            # defining the basis functions b_{j}(x)
tf <- function(x, xj, j) {                                      # tf: tent function
  dj = xj * 0 ; dj[j] = 1#      generating j-th tf from set defined by knots x_{j}
  approx(xj, dj, x)$y                               # performing linear interpolation
}
# </r code> ================================================================== #
```

Now, writing the function

```
# <r code> =================================================================== #
# converting returned constrained model and penalty matrices into mixed model form
mmform <- function(x, xk = NULL, k = 10, sep = TRUE) {
                                            # using default number of knots, k = 10
  if (is.null(xk))                                         #     if x_{k} is null,
    xk = seq(min(x), max(x), length = k)                   # build a grid of knots
  xd = tf.XD(x, xk)            # computing constrained versions of X_{j} and D_{j}
  D = rbind(0, xd$D) ; D[1, 1] = 1                            # doing modifications
  X = t(solve( t(D), t(xd$X) ))                         # computing model matrix X
  if (sep) list(X = X[ , 1, drop = FALSE], Z = X[ , -1], xk = xk)
  else list(X = X, xk = xk)
}
# </r code> ================================================================== #
```

□

(b)

Using your function from part (a) obtain the model matrices required to fit the two term additive tree model, and estimate it using `lme`. Because there are now two smooths, two `pdIdent` terms will be needed in the `random` list supplied to `lme`, which will involve two dummy grouping variables (which can just be differently named copies of the same variable).

Solution:

```
# <r code> =================================================================== #
   # generating constrained versions of X_{j} and D_{J} matrices for the variables
x_h <- mmform(trees$Height) ; x_g <- mmform(trees$Girth)

                 # putting together, building model matrix X with intercept column
X <- cbind(1, x_h$X, x_g$X)

Z_h <- x_h$Z ; Z_g <- x_g$Z                  # taking square root penalty matrices D

g1 <- g2 <- factor(rep(1, nrow(X)))                    # length of X, number of rows
library(nlme)                                                      # loading library
Y <- trees$Volume                                                  # response vector
                                  # fitting the mixed model with positive definite
                                  #             matrices structure of class pdIdent
model <- lme(Y ~ X - 1, random = list(g1 = pdIdent(~ Z_h - 1),
                                      g2 = pdIdent(~ Z_g - 1))
             )
# </r code> ================================================================== #
```

□

(c)

Produce residual versus fitted volume and raw volume against fitted volume plots.

Solution:

```
# <r code> =================================================================== #
rsd <- Y - fitted(model)                                      # computing residuals
Y_hat <- fitted(model)                                              # fitted values

par(mfrow = c(1, 2))                                        # graphical definitions
                                                                         # plotting
plot(Y_hat, rsd, xlab = "Fitted volume", ylab = "Residues", main = "(a)")
abline(h = 0, lty = 2)                                        # dashed line in zero
```

```r
plot(Y_hat, Y, xlab = "Fitted volume", ylab = "Raw volume", main = "(b)")
abline(a = 1, b = 1, lty = 2)                                # perfect dashed line
# </r code> =================================================================== #
```

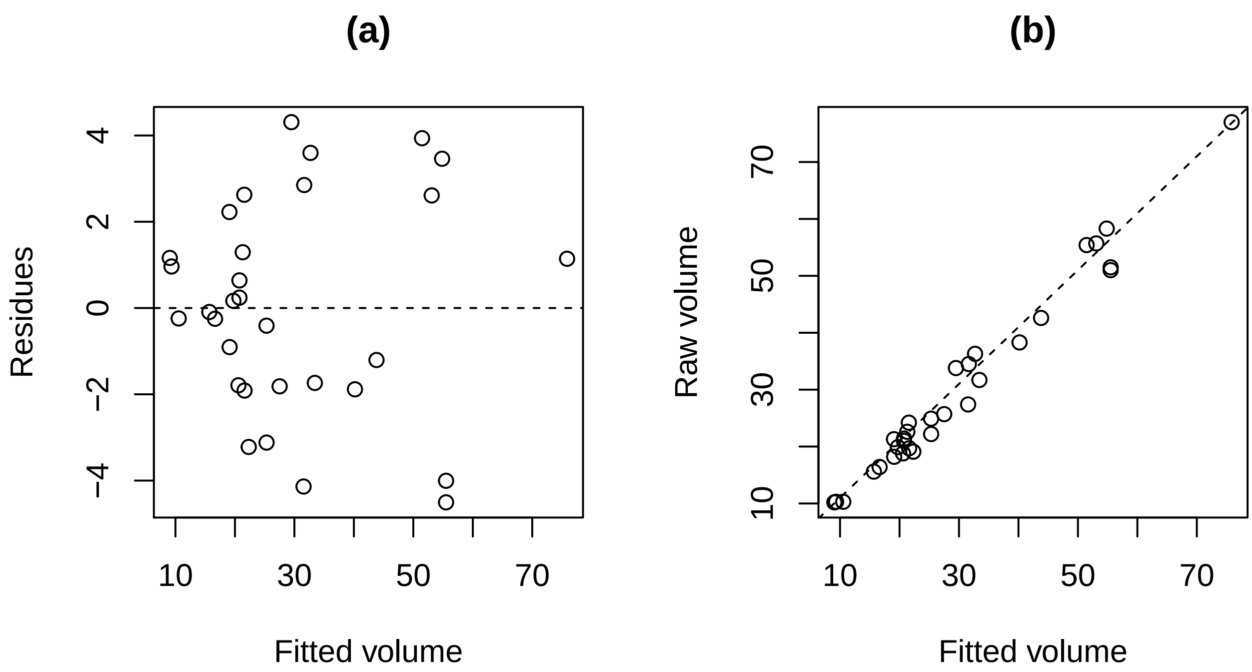


Figure 4: (a): fitted volume against residues; (b): fitted volume against raw volume.

In (a) we see the values spread between -4 and 4, what is good (but not so good); and in (b) we see that the values are quite similar, considering the sample size of 31.

□

(d)

Produce plots of the two smooth effect estimates with partial residuals.

Solution:

```r
# <r code> ==================================================================== #
                                           # getting prediction model matrices X's
X_h <- mmform(trees$Height, xk = x_h$xk, sep = FALSE)$X
X_g <- mmform(trees$Girth, xk = x_g$xk, sep = FALSE)$X

                                        # getting the coefficients for the smooths
coef_h <- as.numeric(coefficients(model)[c(2, 4:11)])                 # s(Height)
```

```r
coef_g <- as.numeric(coefficients(model)[c(3, 12:19)])              # s(Girth)

s_h <- X_h %*% coef_h           # doing the computations by the X's model matrices
s_g <- X_g %*% coef_g

par(mfrow = c(1, 2))                                      # graphical definitions
                                                                      # plotting
plot(trees$Height, s_h + rsd, xlab = "Height", ylab = "s(Height)", main = "(a)")
                                                             # addying fitted curve
lines(trees$Height, s_h, col = "#0080ff", lwd = 2)

plot(trees$Girth, s_g + rsd, xlab = "Girth", ylab = "s(Girth)", main = "(b)")
                                                             # addying fitted curve
lines(trees$Girth, s_g, col = "#0080ff", lwd = 2)
# </r code> ===================================================================== #
```

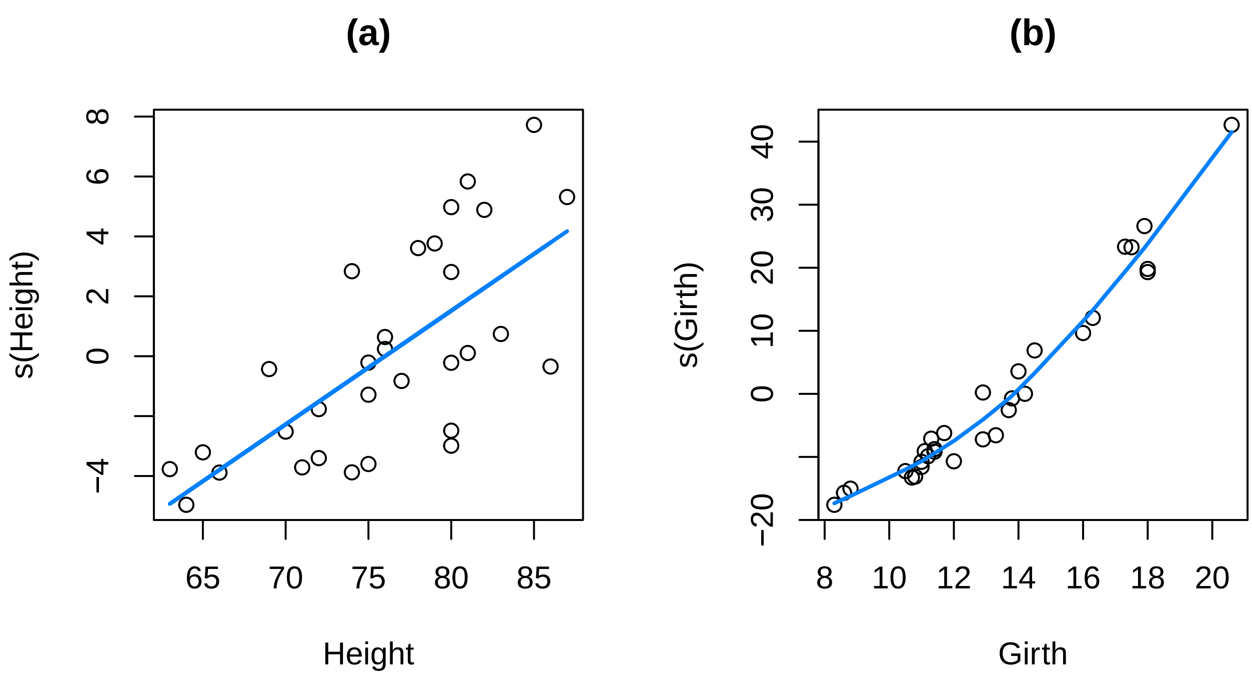


Figure 5: (a): height against s(height); (b): girth against s(girth).

□

Project proposal

Analyze some datasets using GAMs and GAMMs (when necessary), and with the bayesian framework.

"Which" bayesian framework?

- JAGS (`R` package `rjags`): Gibbs sampling;
- INLA (`R` package `INLA`): Integrated Nested Laplace Approximation.

To compare and to follow a reasoning starting with a more simple model, some non-bayesian models will be fitted using the `R` package `mgcv`.

Following the reasoning, some more simple models can also be fitted, as linear models and mixed linear models.

Datasets

1. Trade union data

Data on 534 U.S. workers with eleven variables (`SemiPar::trade.union`).

`R summary` output for the dataset:

```
   years.educ     south    female   years.experience union.member
 Min.   : 2.00   0:378   0:289   Min.   : 0.00    0:438
 1st Qu.:12.00   1:156   1:245   1st Qu.: 8.00    1: 96
 Median :12.00                   Median :15.00
 Mean   :13.02                   Mean   :17.82
 3rd Qu.:15.00                   3rd Qu.:26.00
 Max.   :18.00                   Max.   :55.00
      wage             age         race    occupation sector  married
 Min.   : 1.000   Min.   :18.00   1: 67   1: 55      0:411   0:184
 1st Qu.: 5.250   1st Qu.:28.00   2: 27   2: 38      1: 99   1:350
 Median : 7.780   Median :35.00   3:440   3: 97      2: 24
 Mean   : 9.024   Mean   :36.83           4: 83
 3rd Qu.:11.250   3rd Qu.:44.00           5:105
 Max.   :44.500   Max.   :64.00           6:156
```

Colors by `union.member` status:

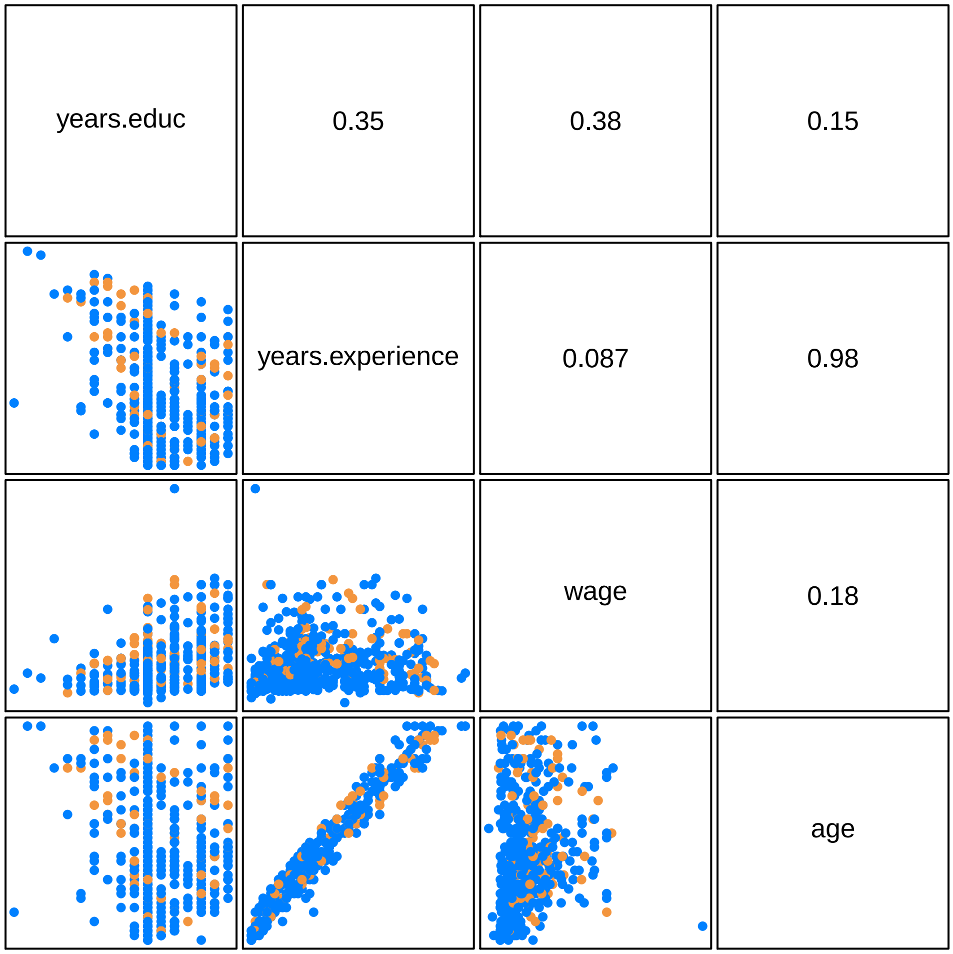


Figure 6: Scatter plots and correlations between the numerical variables of the dataset `trade.union`.

2. Sitka spruce data

13 measurements of log-size for 79 Sitka spruce trees grown in normal or ozone-enriched environments. The first 54 trees have an ozone-enriched atmosphere, the remaining 25 trees have a normal (control) atmosphere. (`SemiPar::sitka`).

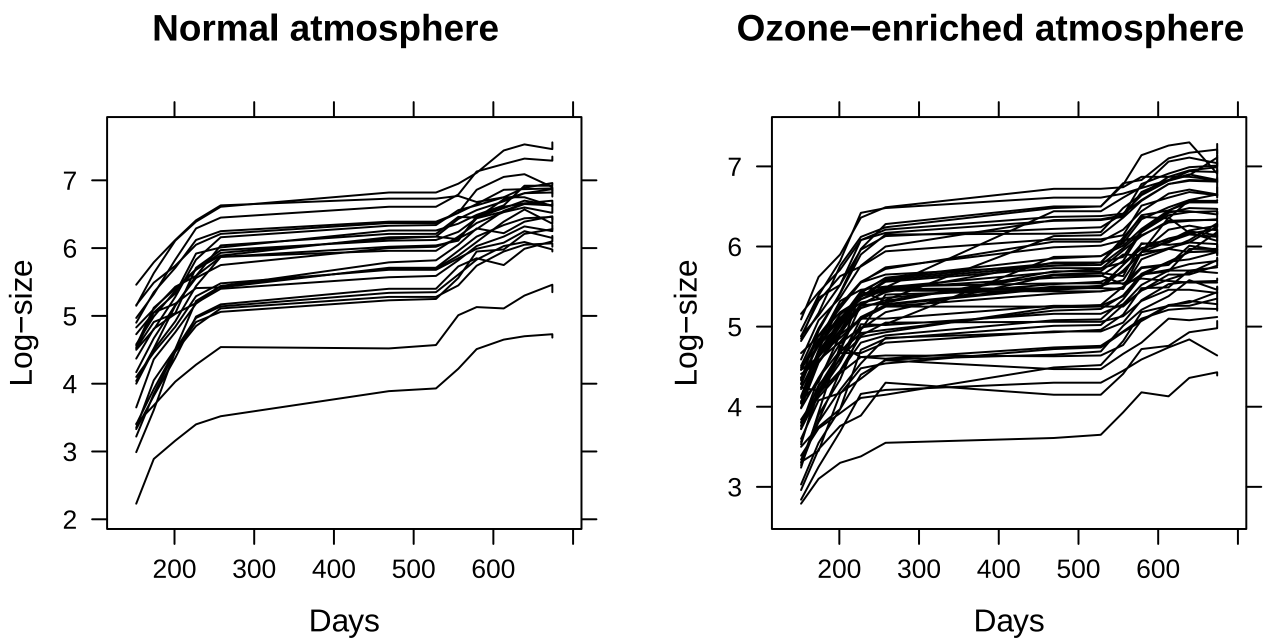
Figure 7: For illustration, Sitka spruce tree.

R `summary` output for the dataset:

```
    id.num        order          days          log.size      ozone  
 Min.   : 1   Min.   : 1   Min.   :152.0   Min.   :2.230   0:325  
 1st Qu.:20   1st Qu.: 4   1st Qu.:227.0   1st Qu.:4.945   1:702  
 Median :40   Median : 7   Median :528.0   Median :5.630          
 Mean   :40   Mean   : 7   Mean   :441.8   Mean   :5.547          
 3rd Qu.:60   3rd Qu.:10   3rd Qu.:613.0   3rd Qu.:6.250          
 Max.   :79   Max.   :13   Max.   :674.0   Max.   :7.560          
```

In the next par of graphs, each line correspond to a Sikta spruce tree along the evaluations, in days.

More comments are (will be) given in the project proposal presentation.


Normal atmosphere
Ozone−enriched atmosphere
Log−size
Days
200
300
400
500
600
2
3
4
5
6
7

■