Series 2: Applied Machine Learning and Predictive Modelling 1

Dr. Luisa Barbanti and Dr. Matteo Tanadini

Fall Semester 2025 (HSLU)

*** Solutions ***

Exercise

In this exercise we are going to use the "Ozone" data set from the `gss` (type `?gss::ozone` for more information about the data).

```
## if not yet there, install the package first:
# install.packages("gss")
data(ozone, package = "gss")
##
head(ozone)
```

```
  upo3 vdht wdsp hmdt sbtp ibht dgpg ibtp vsty day
1    3 5710    4   28   40 2693  -25   87  250   3
2    5 5700    3   37   45  590  -24  128  100   4
3    5 5760    3   51   54 1450   25  139   60   5
4    6 5720    4   69   35 1568   15  121   60   6
5    4 5790    6   19   45 2631  -33  123  100   7
6    4 5790    3   25   55  554  -28  182  250   8
```

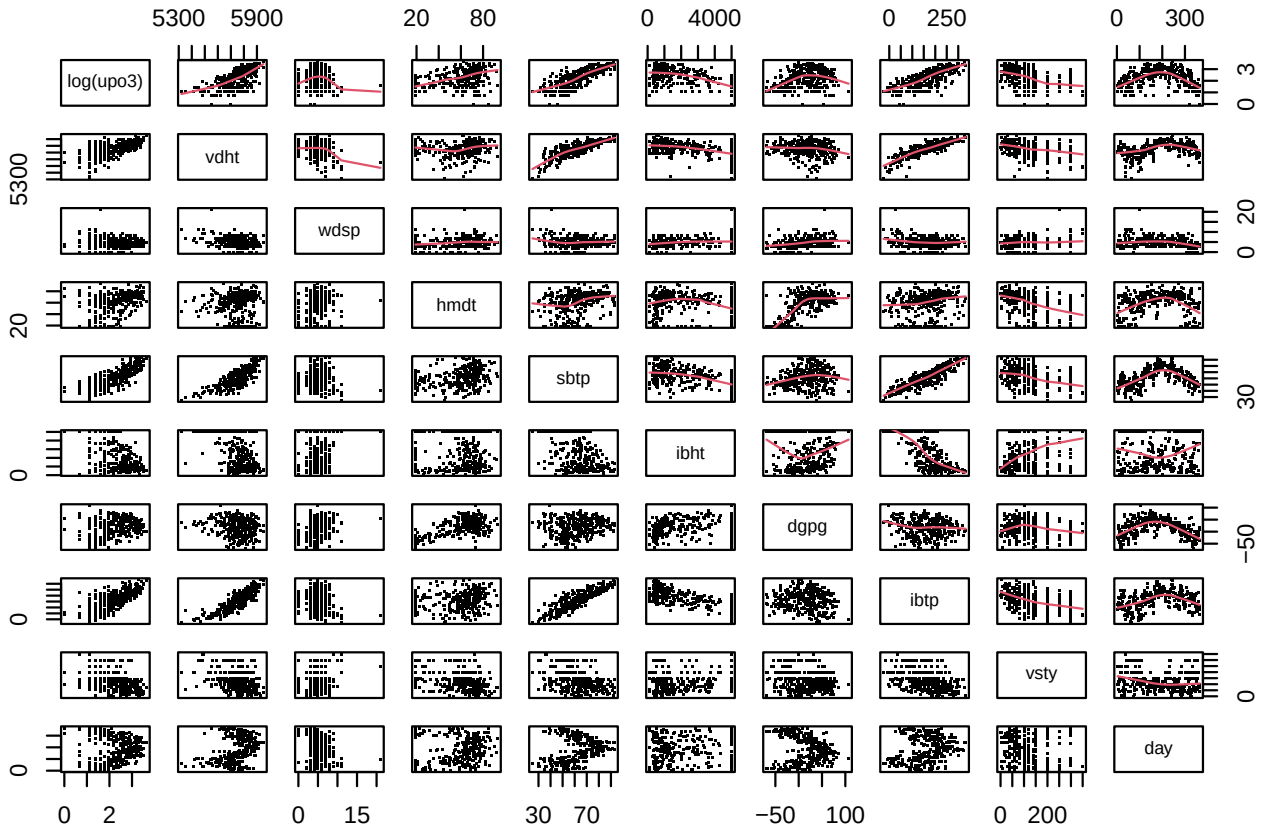
```
# ?gss::ozone
```

Our aim here is to fit a GAM model to the `ozone` data set.

As usual we start by visualising the response variable against each predictor. As the response variable can only take positive values (i.e. is an "amount") we log-transform it. This is often the standard procedure with concentrations.

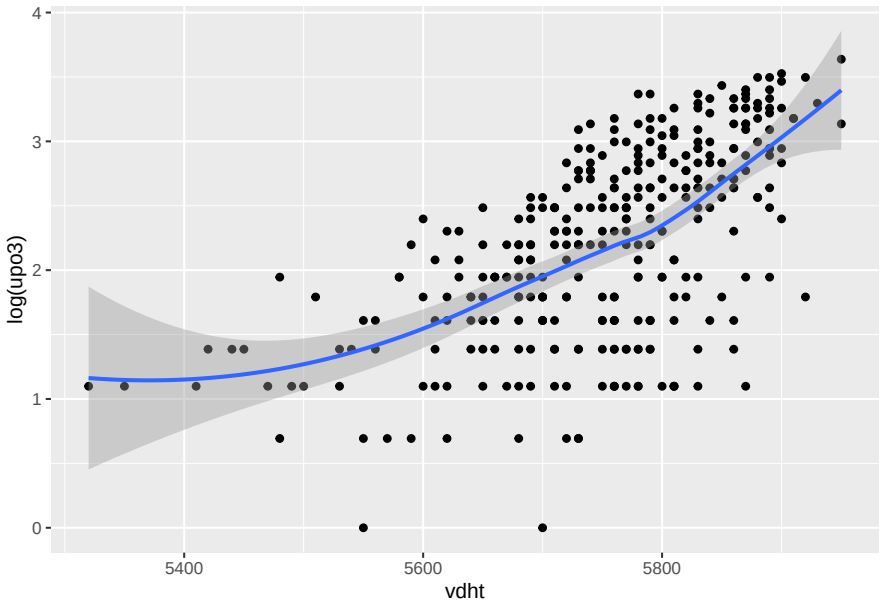
To start "quick and dirty" we can create a graph with `pairs()`.

```
pairs(log(upo3) ~ . ,
      data = ozone,
      pch = ".",
      upper.panel = panel.smooth)
```



As there are many predictors, the single graphs are too small. Therefore, we prefer to create each graph separately. Let's start with `vdht`.

```
library(ggplot2)
ggplot(data = ozone,
       mapping = aes(y = log(upo3),
                     x = vdht)) +
  geom_point() +
  geom_smooth()
```



This relationship appears to be non-linear. It is possible that a simple quadratic term is enough to model it. Nevertheless, we will fit a GAM model and then look at the estimated smooth function and at its "edf" (i.e. estimated degrees of freedom) that quantify the complexity of a smooth term.

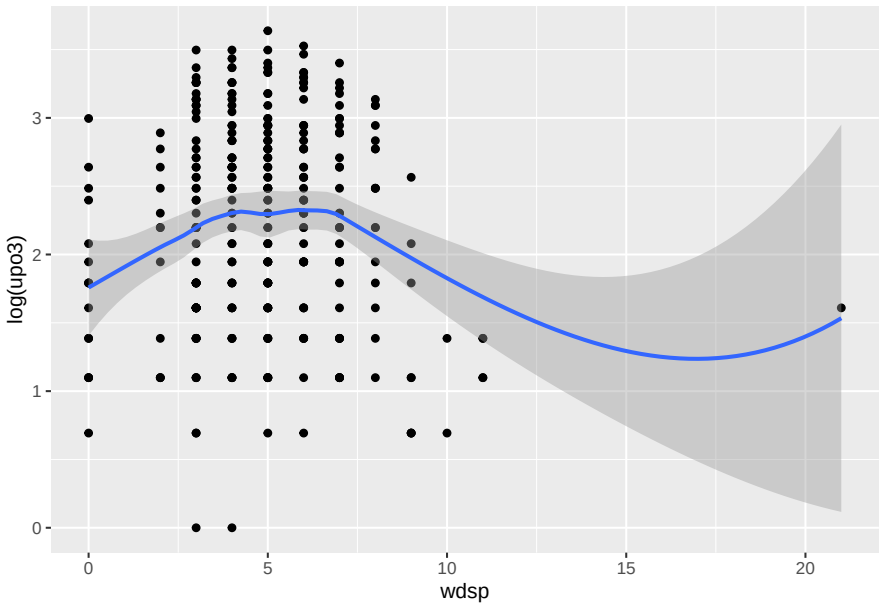
Question

Go through all the other predictors, produce the corresponding graph and comment on the relationship. You may want to say whether the relationship can be assumed to be linear, quadratic or more complex. If you prefer, you may look at a couple of predictors only (as there are many).

Answers

We now turn our attention to the next predictor: Wind speed (i.e. `wdsp`).

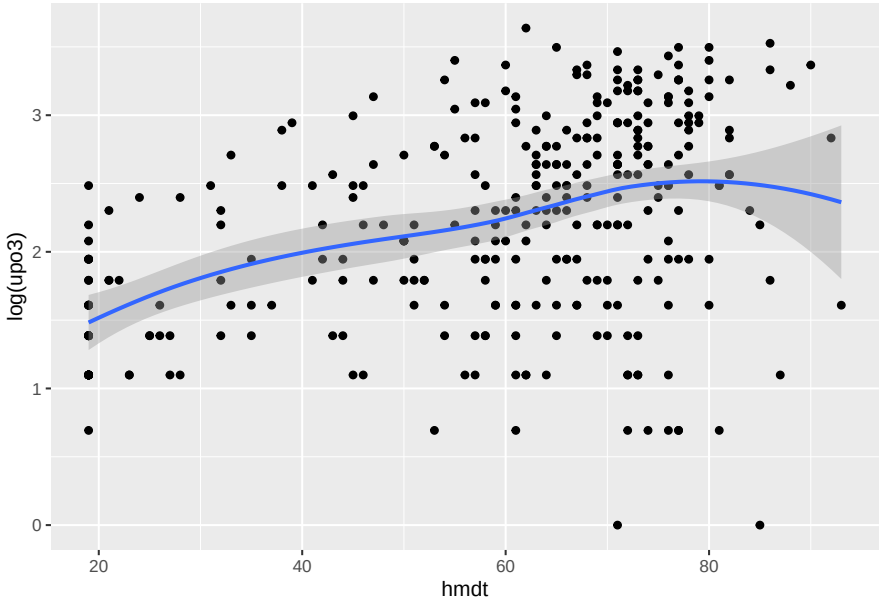
```
ggplot(data = ozone,
       mapping = aes(y = log(upo3),
                     x = wdsp)) +
  geom_point() +
  geom_smooth()
```



This relationship does not seem to be linear. Although this is not the focus of this series, we not that we have an influential observation here. Indeed, there a an extreme x-values that completely drives the smoother on the right-hand side of the graph. For now, we don't take any action.

We now turn our attention to humidity (i.e. `hmdt`).

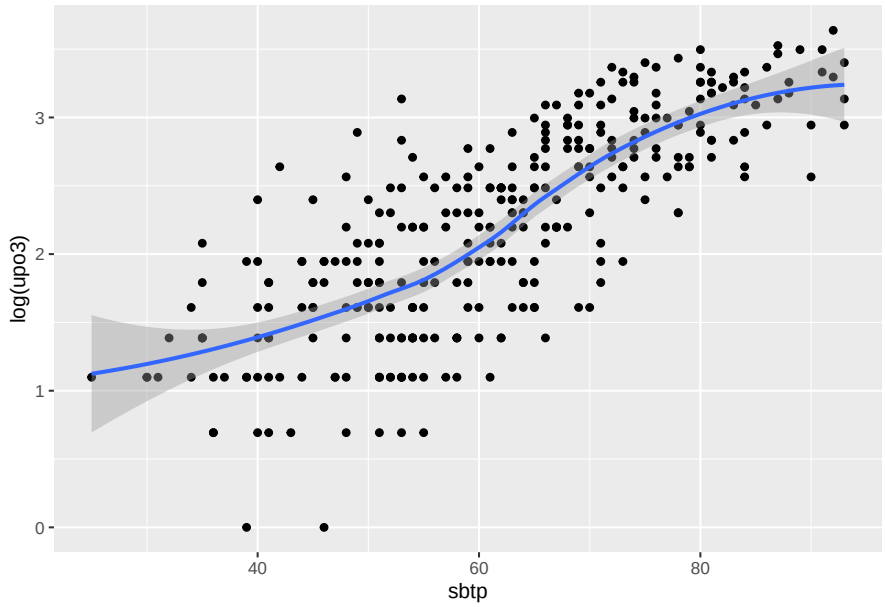
```
ggplot(data = ozone,
       mapping = aes(y = log(upo3),
                     x = hmdt)) +
  geom_point() +
  geom_smooth()
```



This relationship seems to be linear. In particular, humidity seems to have a positive effect.

We now turn our attention to air temperature (i.e. `sbtp`).

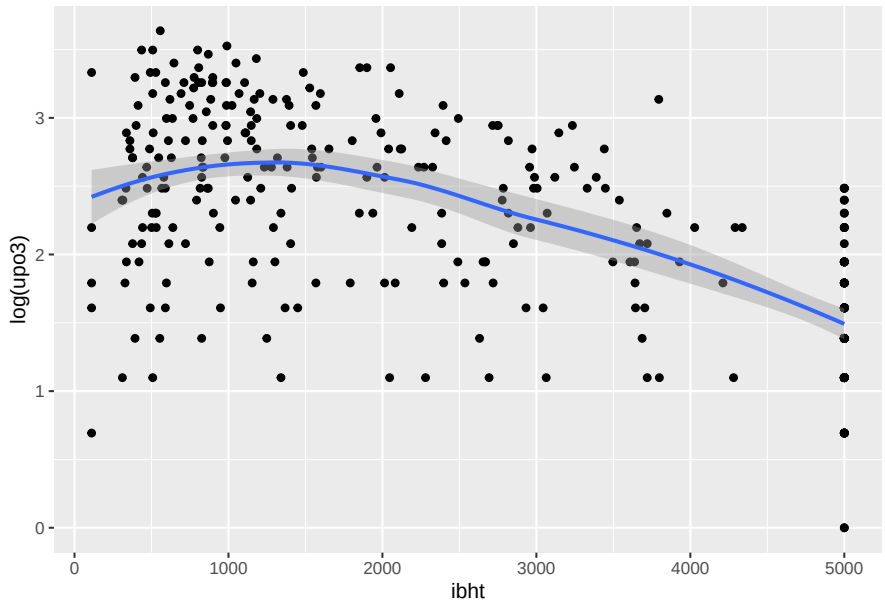
```
ggplot(data = ozone,
       mapping = aes(y = log(upo3),
                     x = sbtp)) +
  geom_point() +
  geom_smooth()
```



This relationship seems to be more complex than quadratic. The smoother shows indeed two bumps.

We now turn our attention to inversion height (i.e. `ibht`).

```
ggplot(data = ozone,
       mapping = aes(y = log(upo3),
                     x = ibht)) +
  geom_point() +
  geom_smooth()
```



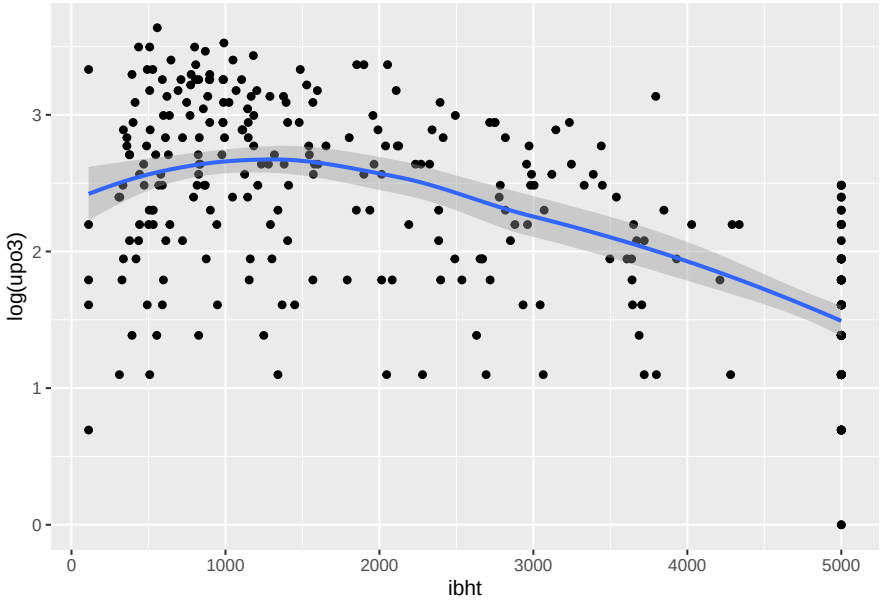
This relationship seems nice and quadratic. Note that there are many observations at a value of 5'000. Our guess is that these observations refer to measurements that were made at 5'000 metres or more. At this kind of issues are not the focus of this series, we do not take any action here.

Note that every time we produce a new graph very many lines of the code are the same. In practice, we only replace the name of the predictor and keep the rest unchanged. In order to shorten our code and to avoid copy and paste errors we can create a "template graph" that we then reuse for each predictor.

```
template.graph.ozone <- ggplot(data = ozone,
                               mapping = aes(y = log(upo3))) +
  geom_point() +
  geom_smooth()
```

Note that in this template graph we did not specify what the x-variable is. Indeed, we want to use this template for different x-variables. To make an example, we can reproduce the plot for `ibht`.

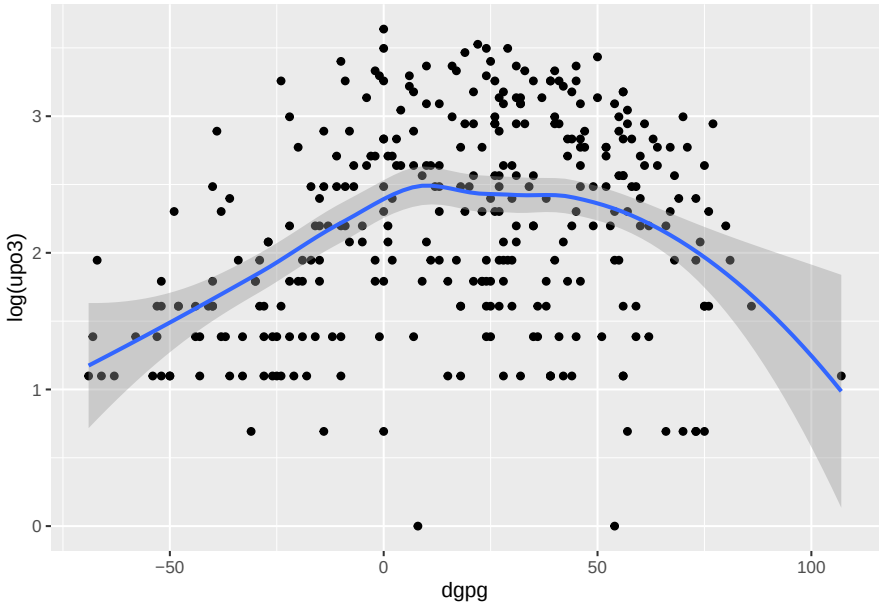
```
template.graph.ozone + aes(x = ibht)
```



This is indeed identical to the graph we obtained above with the "long" call.

We now turn our attention to air pressure (i.e. `dgpg`).

```
template.graph.ozone + aes(x = dgpg)
```

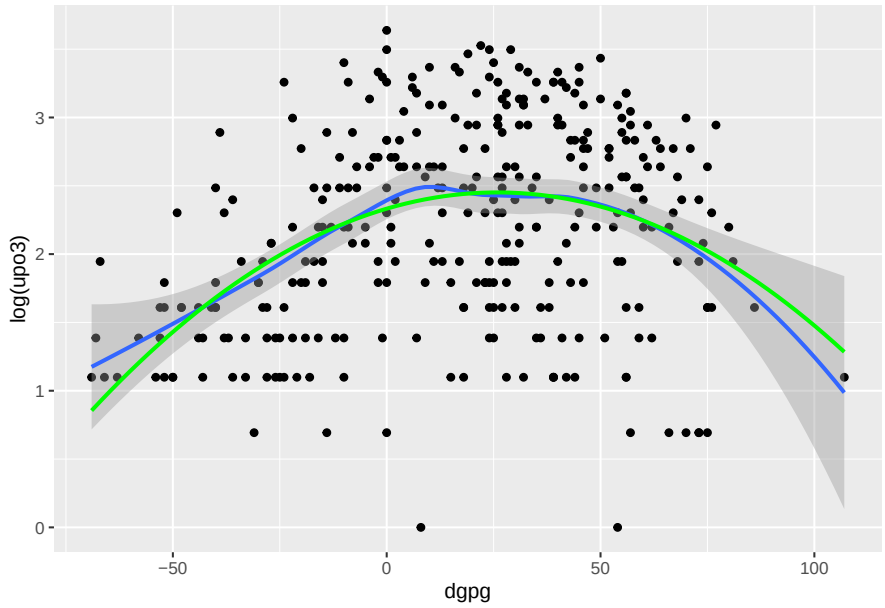


The relationship between the response and `dgpg` seems to increase linearly till the value 0. Then it stays relatively constant and the after 50 decreases again.

This is typical case, where using a quadratic polynomial may lead to a too crude approximation[1]. Indeed, in practice it may be important to note that the relationship seems to be somehow piecewise linear. For example it may be important in practice to highlight that there is clear change at around zero. If we were to use a quadratic polynomial we would miss this pattern (see graph below).

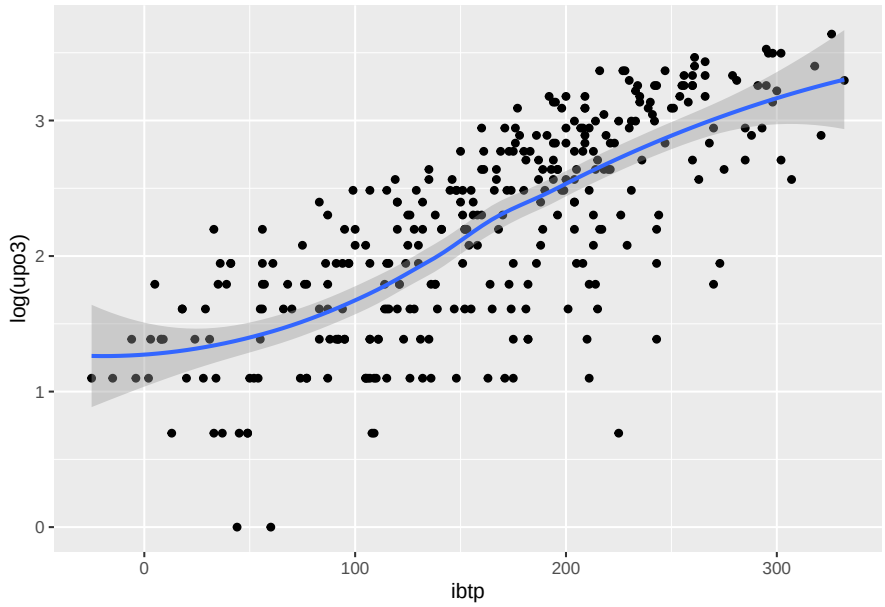
[1]This is not something you are expected to know. Some experience is needed to make this kind of comments.

```
template.graph.ozone + aes(x = dgpg) +
  geom_smooth(method = "lm", formula = y ~ poly(x, degree = 2),
              se = FALSE, colour = "green")
```



We now turn our attention to inversion temperature (i.e. `ibtp`).

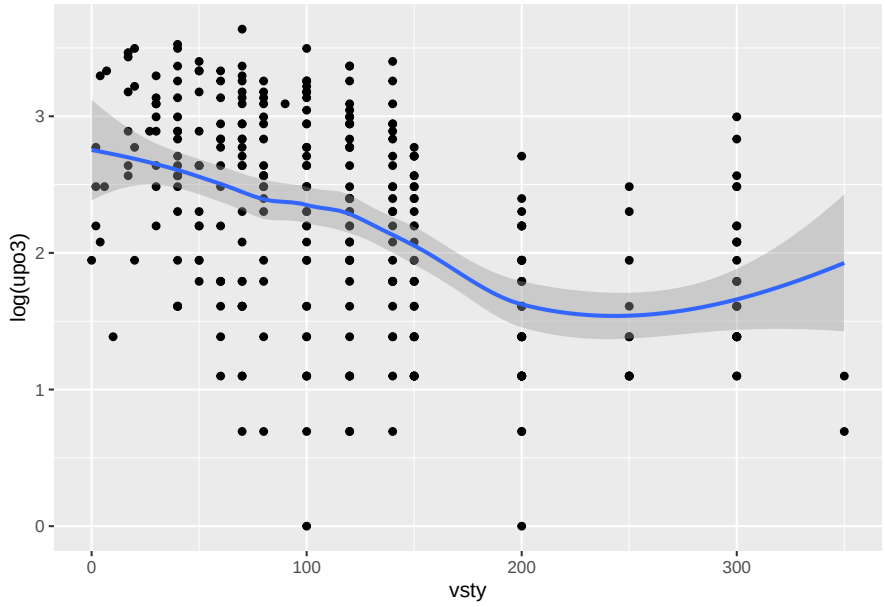
```
template.graph.ozone + aes(x = ibtp)
```



This relationship may be approximated with a cubic polynomials. Nevertheless, we will use a GAMs model and see what the estimated complexity of the smooth term is (i.e. we will look at the "edf" value).

We now turn our attention to visibility (i.e. `vsty`).

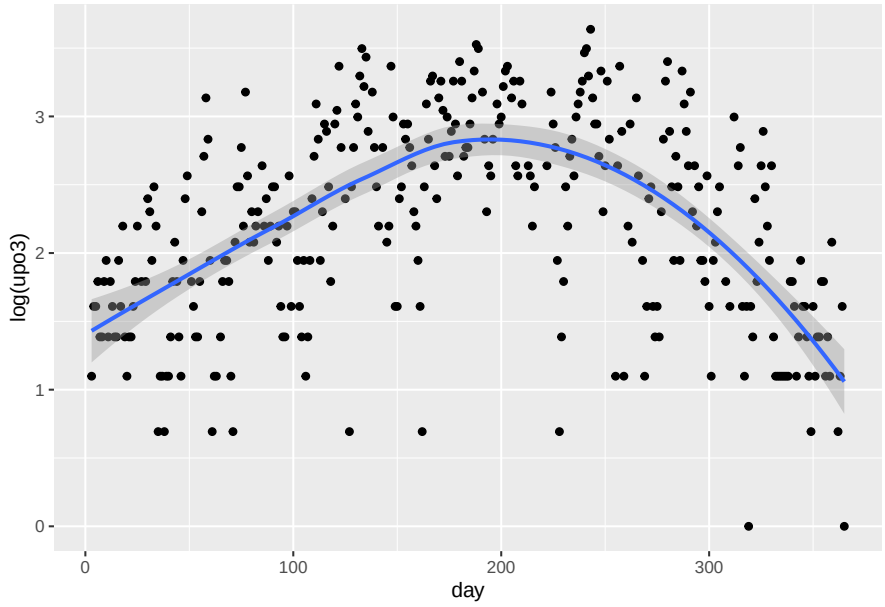
```
template.graph.ozone + aes(x = vsty)
```



This relationship does not seem to fit into a quadratic or cubic polynomial. Again, we estimate it via a smooth term.

Finally, we turn our attention to day of the year (i.e. `day`).

```
template.graph.ozone + aes(x = day)
```



This relationship it likely to be well approximated with a quadratic polynomial.

% % scale_y_log10() + % sarebbe interessante, ma dopo devo spiegare perché faccio log(y) nel modello % # scale_y_continuous(trans = "log") + % bello ma la y-scale e in veramente nel log-space, non belllo. . . % %

Question

Fit a Generalised Additive Model with the `gam()` function from {mgcv} package. Remember to fit the model to the log-transformed response variable.

Answers

Let's fit a model where all these continuous predictors are taken as smooth terms.

```
library(mgcv)
gam.03 <- gam(log(upo3) ~ s(vdht) + s(wdsp) + s(hmdt) +
                s(sbtp) + s(ibht) + s(dgpg) + s(ibtp) +
                s(vsty) + s(day),
              data = ozone)
```

Question

Look at the summary of the model you just fitted and answer the following questions: i) How many of the smooth terms appear to have a significant effect if we were to use a strict 5% threshold? ii) Which smooth terms are estimated to have linear effect? iii) Which one is the term that is estimated to be the most complex? iv) Are there any "parametric" terms in this model?

Answers

Let's print the summary of this model.

```
summary(gam.03)
```

```
Family: gaussian
Link function: identity

Formula:
log(upo3) ~ s(vdht) + s(wdsp) + s(hmdt) + s(sbtp) + s(ibht) +
    s(dgpg) + s(ibtp) + s(vsty) + s(day)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.2130     0.0172     129   <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(vdht) 1.00   1.00 10.97  0.0010 **
s(wdsp) 2.53   3.19  2.84  0.0411 *
s(hmdt) 2.35   2.96  2.55  0.0482 *
s(sbtp) 3.77   4.70  4.21  0.0015 **
s(ibht) 2.80   3.42  5.19  0.0013 **
s(dgpg) 3.25   4.13 14.17 < 2e-16 ***
s(ibtp) 1.00   1.00  0.45  0.5037
s(vsty) 5.73   6.88  5.91 2.9e-06 ***
s(day)  4.61   5.73 24.27 < 2e-16 ***
---
```

```
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.826   Deviance explained = 84.1%
GCV = 0.10629  Scale est. = 0.097256  n = 330
```

i) All the smooth terms apart from `s(ibtp)` have a p-value below the 5% threshold. Note, however, that using the 5% threshold in a black or white manner is bad practice. Indeed, we would rather say that for `s(vdht)`, `s(sbtp)`, `s(ibht)`, `s(dgpg)`, `s(vsty)` and `s(day)` there is strong evidence that they play a role. In addition, there is some weak evidence that `s(wdsp)` and `s(hmdt)` may play a role. There is no evidence that `s(iibtp)` plays a role.

ii) Two smooth terms `s(vdht)` and `s(ibtp)` have an estimated degree of freedom of 1. Their effect is thus estimated to be linear. Note that `s(ibtp)` having a non-significant p-values is considered to have no effect at all (i.e. a straight line that can be considered flat).

iii) The smooth term `s(vsty)` has the largest `edf` value (i.e. 5.7).

iv) Yes, there is an intercept.

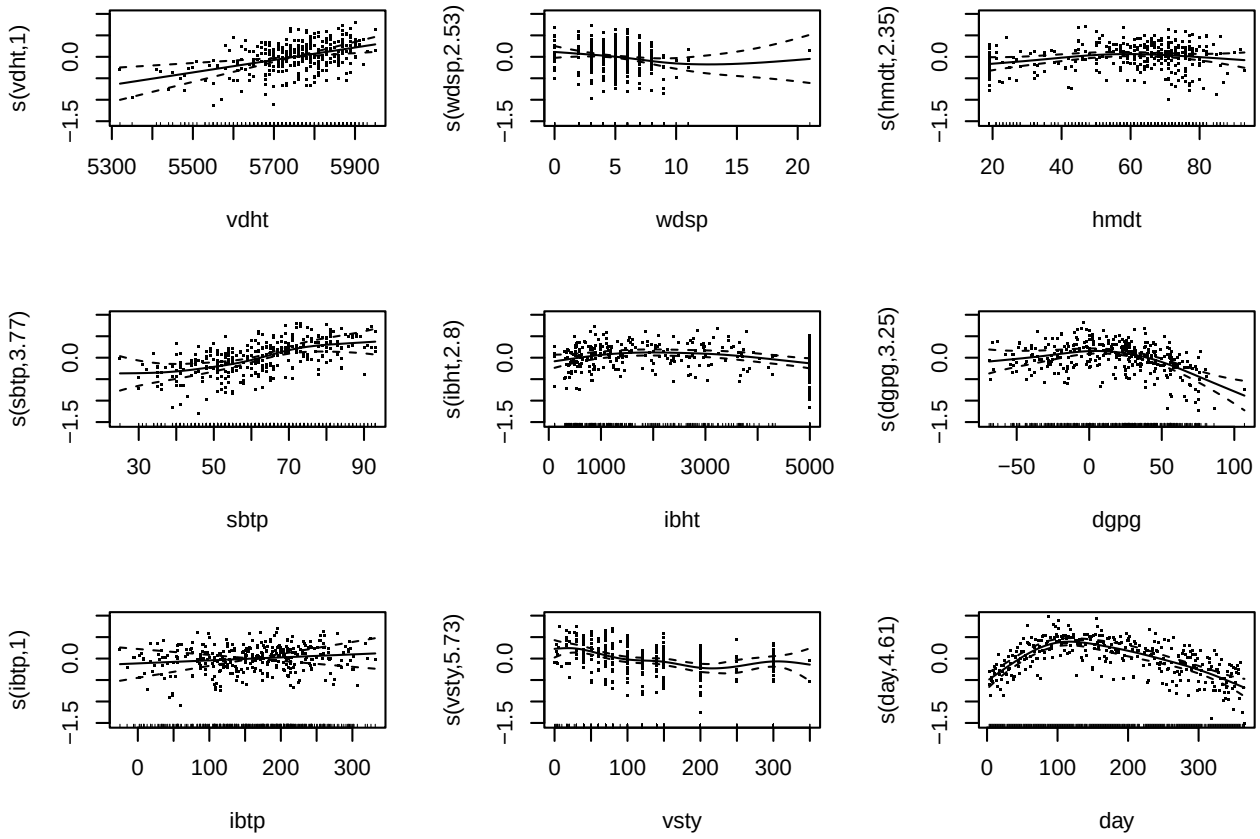
Question

Use the `plot()` function to visualise the estimated smooth terms.

Answers

We visualise all the estimated smooth functions on one page.

```
plot(gam.03, residuals = TRUE, pages = 1, shade = TRUE)
```



Question

Ask generative AI to

- *provide an example or an explanation when it is important to include polynomial effects in a linear model*
- *explain the concept of collinearity*
- *explain when it is appropriate to log-transform a variable before putting it in a model*

and compare it with the definitions you find in the lecture materials. Do you think they are different in any way? Which one is easier for you to understand?