Applied Machine Learning and Predictive Modelling 1 - Exercises

Nils Rechberger

2026-02-21

Series 1: Linear Models

In class we fitted a model to the "cats" dataset. You may remember that the interpretation of the intercept was somehow problematic. Let's get the data, visualise it and refit the model again.

```
d.cats <- read.csv(
    file = "/home/nils/dev/mscids-notes/fs26/mpm1/data/Cats.csv",
    header = TRUE,
    stringsAsFactors = TRUE
    )

str(d.cats)
```

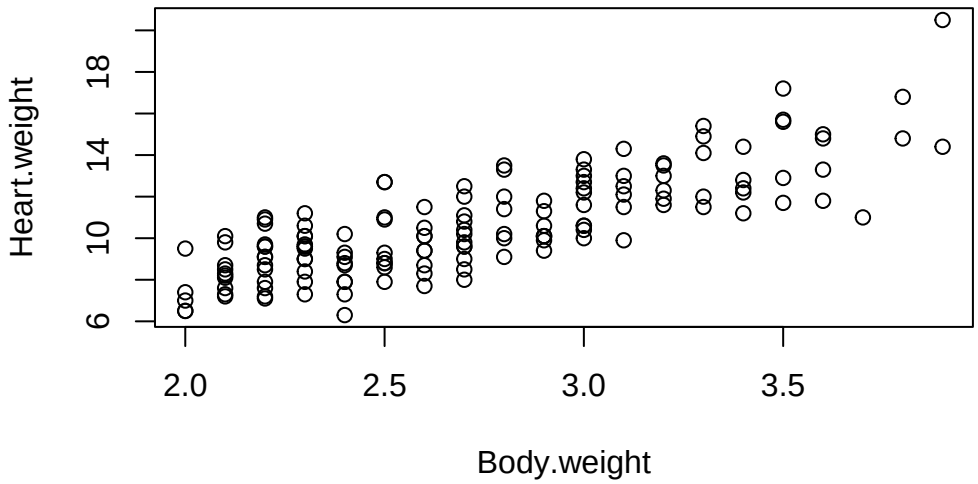
```
'data.frame':   144 obs. of  3 variables:
 $ Sex         : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
 $ Body.weight : num  2 2 2 2.1 2.1 2.1 2.1 2.1 2.1 2.1 ...
 $ Heart.weight: num  7 7.4 9.5 7.2 7.3 7.6 8.1 8.2 8.3 8.5 ...
```

```
head(d.cats)
```

```
  Sex Body.weight Heart.weight
1   F         2.0          7.0
2   F         2.0          7.4
3   F         2.0          9.5
4   F         2.1          7.2
5   F         2.1          7.3
6   F         2.1          7.6
```

Let's display the effect of Body.weight.

```
plot(Heart.weight ~ Body.weight, data = d.cats)
```



The first model we fitted was:

```
lm.cats <- lm(Heart.weight ~ Body.weight, data = d.cats)
```

The estimated coefficients of this model are:

```
coef(lm.cats)
```

```
(Intercept) Body.weight
 -0.3566624   4.0340627
```

As mentioned in class, the correct interpretation of the intercept is "a cat with zero body.weight, is expected to have a heart weight of -0.36". It is obviously nonsensical for two reasons:

1. There is no cat of zero body weight
2. A negative prediction for the response variable heart.weight is impossible in reality.

Questions

Question 1

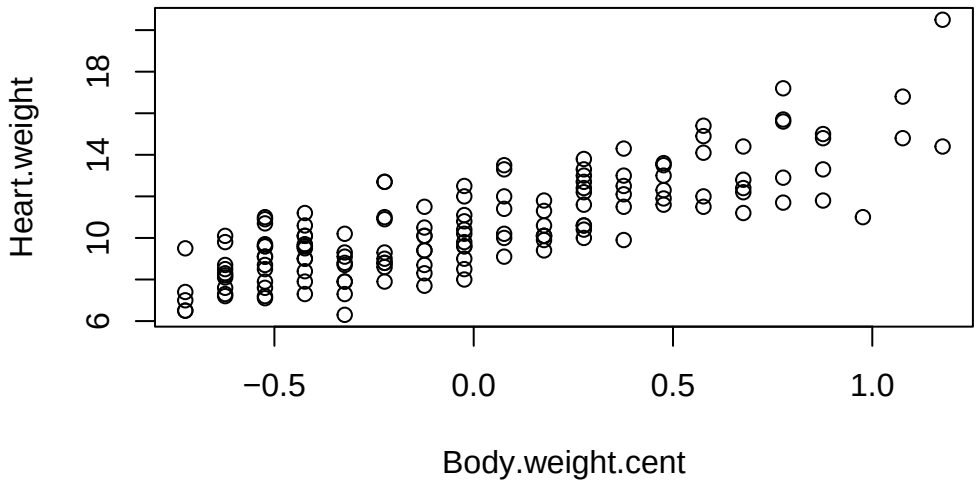
How would you proceed to simplify the interpretation of the intercept in this model? Hint: try to manipulate the predictor body.weight (e.g. by centering it).

Answer

By centering the variable `Body.weight`, we can get a interpretable intercept.

$$\text{Body.weight.cent} = \text{Body.weight} - \bar{x}$$

```
d.cats$Body.weight.cent <- d.cats$Body.weight - mean(d.cats$Body.weight)
plot(Heart.weight ~ Body.weight.cent, data = d.cats)
```



Question 2

Let's turn our attention to the model that contains sex too, but no interaction. Reparametrise this model such that "M" is the reference. Hint: use the relevel() function.

Answer

First we check the current level of `Sex`.

```
levels(d.cats$Sex)
```

```
[1] "F" "M"
```

We can see that `F` ist the reference level. To change that:

```
d.cats$Sex <- relevel(x = d.cats$Sex, ref = "M")
levels(d.cats$Sex)
```

```
[1] "M" "F"
```

Now we need to refit our model:

```
lm.cats.relevelled <- lm(Heart.weight ~ Body.weight + Sex, data = d.cats)
coef(lm.cats.relevelled)
```

```
(Intercept) Body.weight        SexF
-0.49704946  4.07576892  0.08209684
```

Question 3

When the predictor sex was added to the model, the estimated coefficient for 'body.weight' slightly changed. Refit both models, show the estimated coefficients and write a sentence that correctly describes their "biological interpretation" of the Body.weight predictor in each model.

Answer

```
cat("Ceoff for without sex:", coef(lm.cats), "\n")
```

```
Ceoff for without sex: -0.3566624 4.034063
```

```
cat("Ceoff for with sex:", coef(lm.cats.relevelled))
```

```
Ceoff for with sex: -0.4970495 4.075769 0.08209684
```

- First model: By increasing by one unit body weight, we expect an increase of 4.03 in the response variable.
- Second model: By increasing by one unit body weight, while keeping all the other predictors fixed, we expect an increase of 4.08 in the response variable.

Question 4

This time we assume that Body.weight was not provided as a continuous variable, but rather as a categorical one. We do this by creating four classes with similar size. With this purpose in mind, we use the `quantil()` and `cut()` functions.

```
quantiles.Body.weight <- quantile(d.cats$Body.weight)
quantiles.Body.weight
```

```
   0%   25%   50%   75%  100%
2.000 2.300 2.700 3.025 3.900
```

```
d.cats$Body.weight.Class <- cut(
    x = d.cats$Body.weight,
    breaks = quantiles.Body.weight,
    include.lowest = TRUE
    )
```

Let's check how many observations are present in each class.

```
table(d.cats$Body.weight.Class)
```

```
   [2,2.3]  (2.3,2.7] (2.7,3.02] (3.02,3.9]
        42         40         26         36
```

Fit a model with Sex and Body.weight.Class and compute a p-value for both predictors.

Answer

```r
lm.cats.bodyClass <- lm(
    Heart.weight ~ Sex + Body.weight.Class,
    data = d.cats)

drop1(lm.cats.bodyClass, test = "F")
```

```
Single term deletions

Model:
Heart.weight ~ Sex + Body.weight.Class
                  Df Sum of Sq    RSS    AIC F value Pr(>F)
<none>                         354.77 139.84
Sex                1      0.30 355.06 137.96  0.1166 0.7333
Body.weight.Class  3    350.49 705.26 232.78 45.7751 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Body.weight.Class seems to play a relevant role, while Sex does not. This is in full agreement with the model we have seen last week where Body.weight was taken as a continuous predictor.

Question 5

Now run some contrasts to see whether all pair of levels of the Body.weight.Class predictor differ from each other. Comment on the results.

Answer

```r
require(multcomp)
```

```
Loading required package: multcomp

Loading required package: mvtnorm

Loading required package: survival

Loading required package: TH.data
```

```
Loading required package: MASS


Attaching package: 'TH.data'

The following object is masked from 'package:MASS':

    geyser
```

```r
glht.1 <- glht(lm.cats.bodyClass, linfct = mcp(Body.weight.Class = "Tukey"))
##
summary(glht.1)
```

```
     Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: lm(formula = Heart.weight ~ Sex + Body.weight.Class, data = d.cats)

Linear Hypotheses:
                             Estimate Std. Error t value Pr(>|t|)
(2.3,2.7] - [2,2.3] == 0       0.7155     0.3813   1.876 0.241674
(2.7,3.02] - [2,2.3] == 0      2.4273     0.4382   5.539  < 1e-04 ***
(3.02,3.9] - [2,2.3] == 0      4.6086     0.4401  10.473  < 1e-04 ***
(2.7,3.02] - (2.3,2.7] == 0    1.7118     0.4042   4.235 0.000227 ***
(3.02,3.9] - (2.3,2.7] == 0    3.8932     0.3816  10.202  < 1e-04 ***
(3.02,3.9] - (2.7,3.02] == 0   2.1814     0.4166   5.236  < 1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

Question 6

Ask generative AI to provide the interpretation of - the coefficients from a linear model - the p-values from a linear 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?

Answer

Prompt:

```
Provide a interpretation of the coefficients and the p-values from a linear model
```

Note: Used Gemeni 3 Fast

Answer (excerpt) :

```
Interpreting a linear model is all about understanding the "story" the data is telling. When
```

The general statement is accurate, although dividing p-values into “significant” and “not significant” is very bad practice.

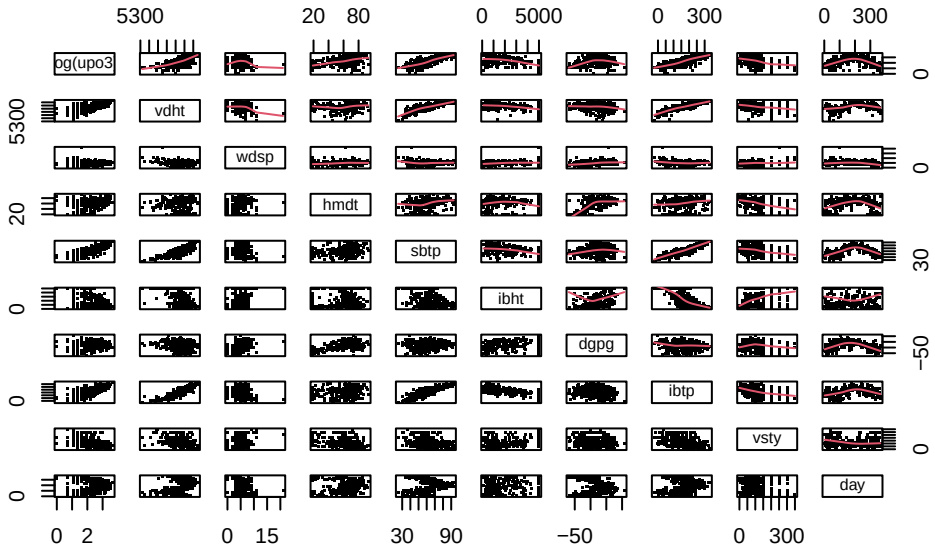
Series 2: Non-linearities

Exercise 1

```
# 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
```

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

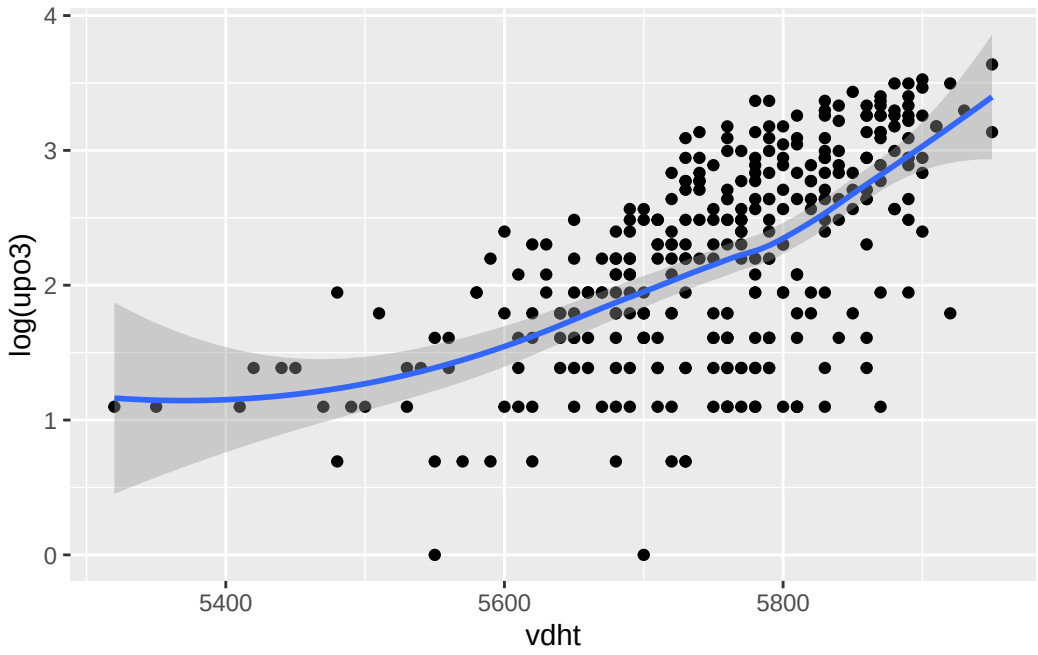


As there are many predictors, the single graphs are too small. Therefore, we prefer to create each graph separately. Let's start with `vdht`.

```
library(ggplot2)

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

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Question 1

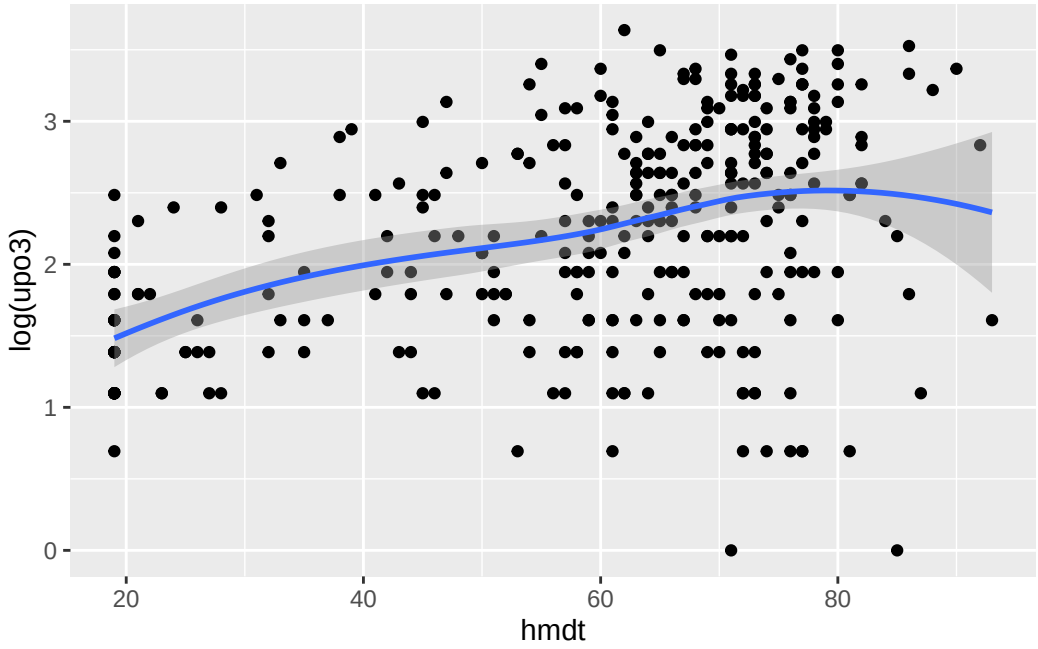
Go through all the other predictors, produce the corresponding graph and comment on the relationship. You may want to say whether the relationship can be assumed to be linear, quadratic or more complex. If you prefer, you may look at a couple of predictors only (as there are many).

Answer

```
library(ggplot2)

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

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

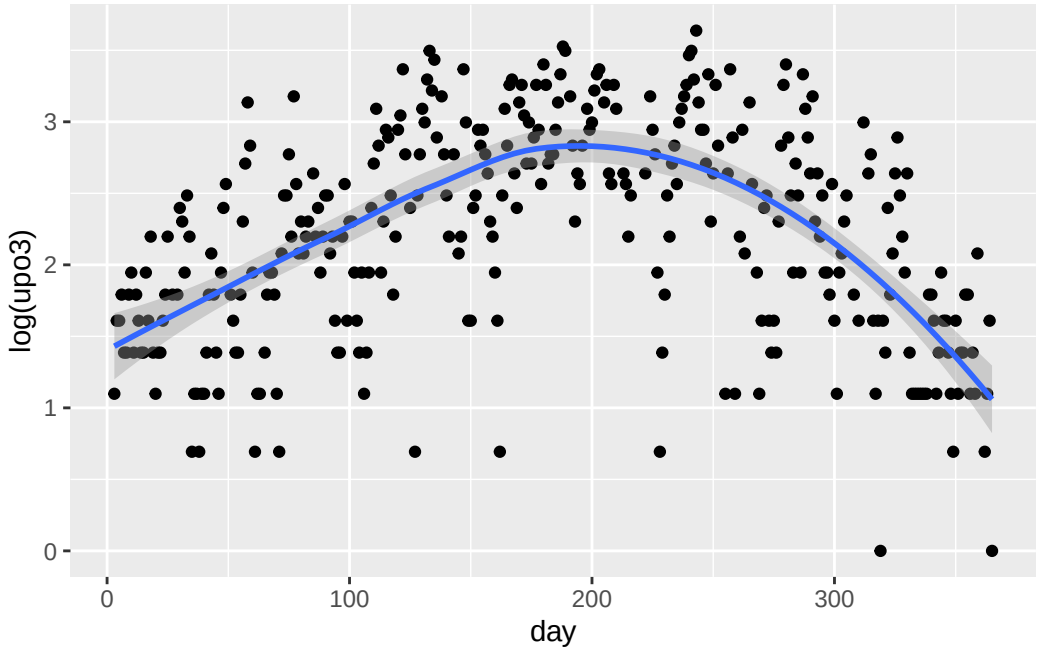


The relationship between `log(upo3)` and `ibtp` appears to be linear, except at the end of the plot.

```
library(ggplot2)

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

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

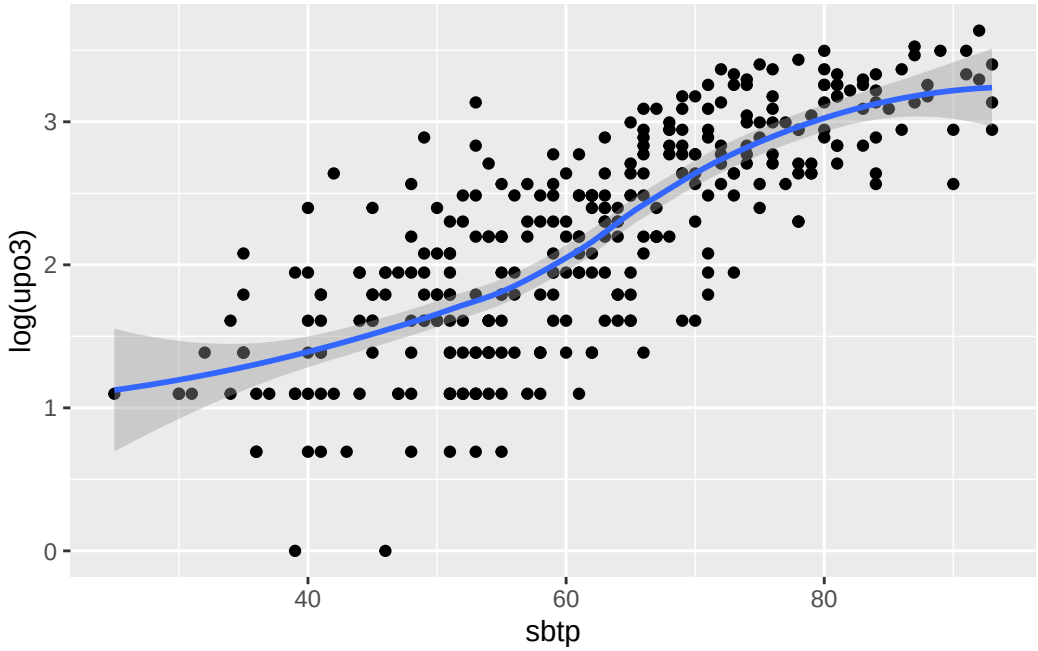


The relationship between `log(upo3)` and `day` appears to be quadratic or polynomial of order 2, 4, 6, etc.

```
library(ggplot2)

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

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



The relationship between `log(upo3)` and `sbpt` appears to be cubic or polynomial of order 3, 5, 7, etc.

Question 2

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

Answer

```
library(ggplot2)
library(mgcv)
```

```
Loading required package: nlme
```

```
This is mgcv 1.9-4. For overview type '?mgcv'.
```

```
gam_ozone <- gam(
    log(upo3) ~ s(vdht) + s(wdsp) + s(hmdt) + s(sbtp) + s(ibht) + s(dgpg) + s(ibtp) + s(vsty
    data = ozone
)
```

Question 3

Look at the summary of the model you just fitted and answer the following questions:

```
summary(gam_ozone)
```

```
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)  2.21297    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(vdht) 1.000  1.000 10.973  0.00104 **
s(wdsp) 2.526  3.194  2.842  0.04113 *
s(hmdt) 2.353  2.958  2.546  0.04818 *
s(sbtp) 3.772  4.703  4.214  0.00146 **
s(ibht) 2.797  3.421  5.185  0.00132 **
s(dgpg) 3.248  4.130 14.169  < 2e-16 ***
s(ibtp) 1.000  1.000  0.448  0.50366
s(vsty) 5.728  6.884  5.908 2.94e-06 ***
s(day)  4.609  5.729 24.269  < 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) 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?

I

All smooth terms except for `ibtp` seems to have a statistical impact on the model.

II

The two smooth terms`vdht` and `ibtp` have an estimated degree of freedom of 1. Their effect is thus estimated to be linear.

III

The smoth term `vsty` seems to bee the most complex one.

IV

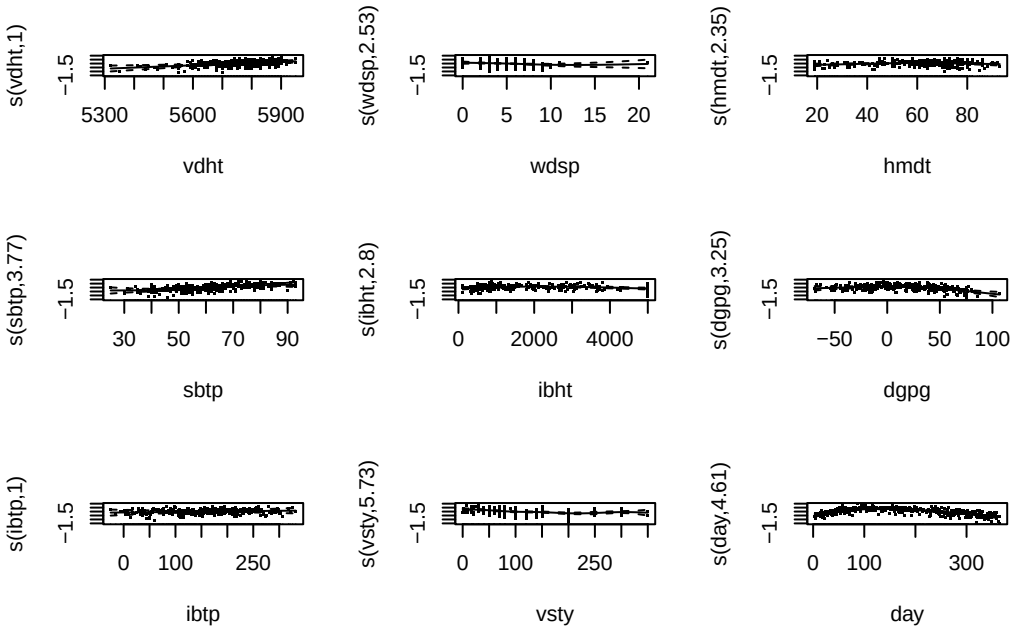
Yes, there is an intercept.

Question 4

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

Answer

```
plot(
    gam_ozone,
    residuals = TRUE,
    pages = 1,
    shade = TRUE
)
```



Series 3: GLMs - Binominal

```
d.xray <- readRDS("/home/nils/dev/mscids-notes/fs26/mpm1/data/Xray.RDS")
str(d.xray)
```

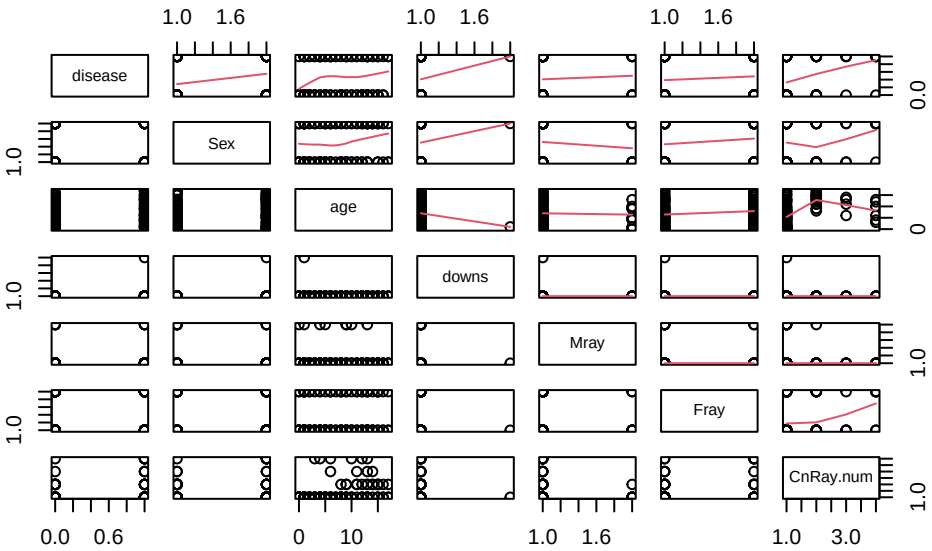
```
tibble [112 x 7] (S3: tbl_df/tbl/data.frame)
 $ Sex      : Factor w/ 2 levels "F","M": 1 2 1 2 2 1 1 2 1 2 ...
 $ disease  : num [1:112] 1 1 0 1 1 1 1 0 0 0 ...
 $ age      : int [1:112] 0 6 8 1 4 9 17 5 0 7 ...
 $ downs    : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
 $ Mray     : Factor w/ 2 levels "no","yes": 1 1 1 1 2 2 1 1 1 1 ...
 $ Fray     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ CnRay.num: num [1:112] 1 3 1 1 1 1 2 1 1 1 ...
```

Question 1

Visualise the data. You may want to take advantage of the `pairs()` function.

Answer

```
pairs(
    disease ~ . ,
    data = d.xray,
    upper.panel = panel.smooth
)
```



Question 2

Fit a starting model to the data. Start by assuming linearity for the continuous predictors and assume that no interaction is needed. Comment on the results obtained wit the summary function.

Answer

```
glm.xray <- glm(
    disease ~ Sex + downs + age + Mray + Fray + CnRay.num,
    data = d.xray,
```

```
    family = "binomial"
)

summary(glm.xray)
```

```
Call:
glm(formula = disease ~ Sex + downs + age + Mray + Fray + CnRay.num,
    family = "binomial", data = d.xray)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.08036    0.54863  -3.792 0.000149 ***
SexM           0.88102    0.41952   2.100 0.035723 *
downsyes      16.06989 1455.39759   0.011 0.991190
age            0.04494    0.04092   1.098 0.272151
Mrayyes        0.69668    0.77274   0.902 0.367290
Frayyes        0.14528    0.43966   0.330 0.741064
CnRay.num      0.65058    0.30944   2.102 0.035514 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 152.97  on 111  degrees of freedom
Residual deviance: 135.87  on 105  degrees of freedom
AIC: 149.87

Number of Fisher Scoring iterations: 14
```

Only two of total six model parameters seems to have a effect on the output (wee p-values).

Question 3

Interpret the coefficients of SexM and age.

Answer

While sex seems to have a statistical impact on the model, age does not.

Question 4

Look at the estimated effect of the variable downs and its p-value. Do the results make sense to you? Comment.

Answer

It's interesting to see that only the parameters `SexM` and `CnRay.num` have a statistical impact on the model.

Series 3: GLMs - Poisson

```
data(InsectSprays)
head(InsectSprays)
```

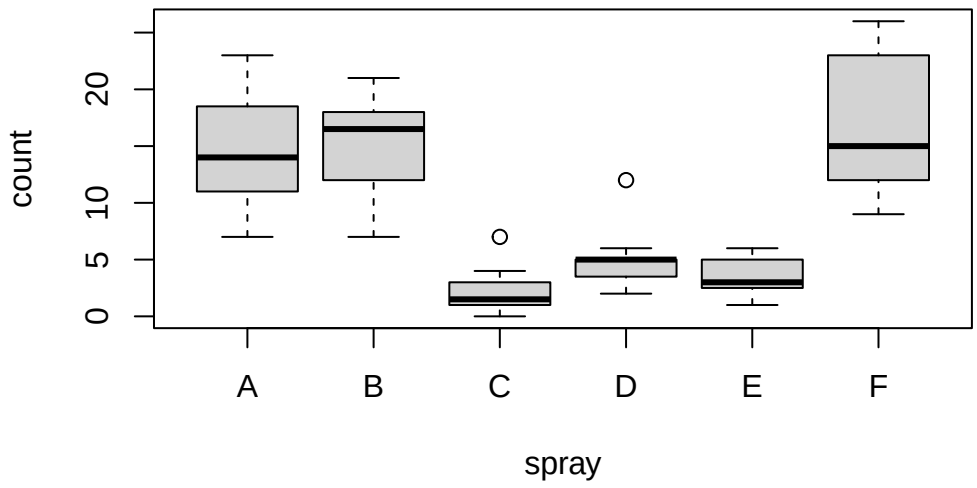
```
  count spray
1    10     A
2     7     A
3    20     A
4    14     A
5    14     A
6    12     A
```

Question 1

Use an appropriate function to visualise the data.

Answer

```
boxplot(count ~ spray, data = InsectSprays)
```



Question 2

What do you observe on the graph? Are there differences among groups? Is the variability within each group similar?

Answer

We can clearly see that some sprays have a better effect than others.

Question 3

Fit a Poisson model to the data and interpret the regression coefficients.

Answer

```
glm.InsectSprays <- glm(
    count ~ spray,
    data = InsectSprays,
    family = "poisson"
```

```
)

summary(glm.InsectSprays)
```

```
Call:
glm(formula = count ~ spray, family = "poisson", data = InsectSprays)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.67415    0.07581  35.274  < 2e-16 ***
sprayB       0.05588    0.10574   0.528    0.597
sprayC      -1.94018    0.21389  -9.071  < 2e-16 ***
sprayD      -1.08152    0.15065  -7.179 7.03e-13 ***
sprayE      -1.42139    0.17192  -8.268  < 2e-16 ***
sprayF       0.13926    0.10367   1.343    0.179
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 409.041  on 71  degrees of freedom
Residual deviance:  98.329  on 66  degrees of freedom
AIC: 376.59

Number of Fisher Scoring iterations: 5
```

Question 4

Check whether the model is overdispersed. If this is the case, fit a quasipoisson model and compare the estimated regression coefficients with those of the Poisson model.

Answer

The residual deviance is about 98 on 66 degrees of freedom. If there were no overdispersion, we would expect the residual deviance to be about 66. We can therefore conclude that the model is overdispersed.

Question 5

Formally test whether there is any evidence that the predictor spray plays a role in the "quasipoisson" model.

Answer

```
glm.InsectSprays <- glm(
    count ~ spray,
    data = InsectSprays,
    family = "quasipoisson"
)

summary(glm.InsectSprays)
```

```
Call:
glm(formula = count ~ spray, family = "quasipoisson", data = InsectSprays)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.67415    0.09309  28.728  < 2e-16 ***
sprayB       0.05588    0.12984   0.430    0.668
sprayC      -1.94018    0.26263  -7.388 3.30e-10 ***
sprayD      -1.08152    0.18499  -5.847 1.70e-07 ***
sprayE      -1.42139    0.21110  -6.733 4.82e-09 ***
sprayF       0.13926    0.12729   1.094    0.278
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 1.507713)

    Null deviance: 409.041  on 71  degrees of freedom
Residual deviance:  98.329  on 66  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

As expected, there is very strong evidence that spray plays a role.

Question 6

We previously tested whether the predictor spray plays a role. Given that this test was statistically significant, we can dig further and test hypotheses that involve some of the levels of this predictor. Use the glht() function in the `multcomp` package to test whether the conventional pesticides "C", "D" and "E" differ from the organic pesticides "A", "B" and "F". Test also whether the newly developed pesticides "A" and "B" differ from the old conventional pesticide "F". Use the "quasipoisson" model to test these hypotheses.

Answer

```
library(multcomp)

glm.insects.quasi <- glm(count ~ spray - 1, data = InsectSprays, family = quasipoisson)

matrix.of.contrasts <- rbind(
    "organic vs conv" = c(1/3, 1/3, -1/3, -1/3, -1/3, 1/3),
    "new vs old conv" = c(1/2, 1/2, 0, 0, 0, -1)
)
colnames(matrix.of.contrasts) <- levels(InsectSprays$spray)

glht.insects <- glht(
    glm.insects.quasi,
    linfct = mcp(spray = matrix.of.contrasts)
)

summary(glht.insects)
```

```
     Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: User-defined Contrasts


Fit: glm(formula = count ~ spray - 1, family = quasipoisson, data = InsectSprays)

Linear Hypotheses:
                       Estimate Std. Error z value Pr(>|z|)
organic vs conv == 0   1.5461     0.1274  12.132   <1e-10 ***
new vs old conv == 0  -0.1113     0.1084  -1.027    0.516
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

As expected (from the graphical analysis), we can confirm that there is a clear difference between organic and conventional pesticides. However, there is no statistical evidence that the new conventional sprays differ from the old conventional one.

Series 4: Cross Validation

```r
d.xray <- readRDS("/home/nils/dev/mscids-notes/fs26/mpm1/data/Xray.RDS")
str(d.xray)
```

```
tibble [112 x 7] (S3: tbl_df/tbl/data.frame)
 $ Sex      : Factor w/ 2 levels "F","M": 1 2 1 2 2 1 1 2 1 2 ...
 $ disease  : num [1:112] 1 1 0 1 1 1 1 0 0 0 ...
 $ age      : int [1:112] 0 6 8 1 4 9 17 5 0 7 ...
 $ downs    : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
 $ Mray     : Factor w/ 2 levels "no","yes": 1 1 1 1 2 2 1 1 1 1 ...
 $ Fray     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ CnRay.num: num [1:112] 1 3 1 1 1 1 2 1 1 1 ...
```

Question 1

Fit the following four models on the whole dataset. Take a look at the summary-output of the four models. Compute the in-sample AIC-values (= AIC-values of the models fitted on the whole dataset) of all four models. You can use the function AIC(). Based on these in-sample AIC-values, which model would you select as the best performing model?

```r
library(mgcv)

glm.xray.simple <- glm(
    formula = disease ~ age + Sex,
    data = d.xray,
    family = "binomial"
)

glm.xray.complex <- glm(
```

```
    formula = disease ~ age + Sex + Mray + Fray + CnRay.num,
    data = d.xray,
    family = "binomial"
)

glm.xray.very.complex <- glm(
    formula = disease ~ Sex * (poly(age, degree = 2) + Mray + Fray + CnRay.num),
    data = d.xray,
    family = "binomial"
)

gam.xray <- gam(
    formula = disease ~ s(age, by = Sex) + Sex + Mray + Fray + CnRay.num,
    data = d.xray,
    family = "binomial"
)
```

Answer

```
cat("AIC of glm.xray.simple:", AIC(glm.xray.simple),"\n")
```

```
AIC of glm.xray.simple: 149.63
```

```
cat("AIC of glm.xray.complex:", AIC(glm.xray.complex) ,"\n")
```

```
AIC of glm.xray.complex: 149.759
```

```
cat("AIC of glm.xray.very.complex:", AIC(glm.xray.very.complex),"\n")
```

```
AIC of glm.xray.very.complex: 158.6927
```

```
cat("AIC of gam.xray:", AIC(gam.xray),"\n")
```

```
AIC of gam.xray: 148.9194
```

Based on the AIC, we should go on with the model `gam.xray` with a value of 148.9.

Question 2

Calculate the proportion of correctly classified observations in-sample (that is on the whole data set as well) for all four models. Use a cutoff of 0.5 in the predicted probability for disease. This means that if a patient has a predicted probability of 0.5 or higher, we would predict disease = 1 for this patient. Based on the in-sample proportion of correctly classified observations, which model would you choose?

Answer

```
prop.correct.classified <- function(model, newdata){
    pred.probabilities <- predict(
        object = model,
        newdata = newdata,
        type = "response"
    )

    pred.disease <- ifelse(pred.probabilities >= 0.5, 1, 0)
    return(mean(pred.disease == newdata$disease))
}
```

```
cat("Prop. of correctly classified observations of glm.xray.simple:", prop.correct.classifie
```

```
Prop. of correctly classified observations of glm.xray.simple: 0.625
```

```
cat("Prop. of correctly classified observations of glm.xray.complex:", prop.correct.classifi
```

```
Prop. of correctly classified observations of glm.xray.complex: 0.6785714
```

```
cat("Prop. of correctly classified observations of glm.xray.very.complex:", prop.correct.cla
```

```
Prop. of correctly classified observations of glm.xray.very.complex: 0.6785714
```

```
cat("Prop. of correctly classified observations of gam.xray:", prop.correct.classified(model
```

```
Prop. of correctly classified observations of gam.xray: 0.75
```

Based on the proportion of correctly classified observations in-sample, we should go on with the model `gam.xray` with a value of 0.75.

Question 3

Now implement 5-fold CV to compare the estimated proportion of correctly classified observations on test data. If you want, you can implement repeated 5-fold CV to see how much the estimated proportion of correctly classified observations varies between different 5-fold CVs (different permutation of the rows of the data).

Answer

```
set.seed(1)

df <- d.xray[sample(nrow(d.xray)), ]

folds <- cut(seq(1, nrow(df)), breaks = 5, labels = FALSE)

test_indices <- which(folds == 1)
test_data  <- df[test_indices, ]
train_data <- df[-test_indices, ]
```

Question 4

Based on the results of the 5-fold CV, which of the four models would you choose and why?

```
library(mgcv)

glm.xray.simple <- glm(
    formula = disease ~ age + Sex,
    data = train_data,
    family = "binomial"
)

glm.xray.complex <- glm(
    formula = disease ~ age + Sex + Mray + Fray + CnRay.num,
    data = train_data,
    family = "binomial"
)

glm.xray.very.complex <- glm(
    formula = disease ~ Sex * (poly(age, degree = 2) + Mray + Fray + CnRay.num),
    data = train_data,
```

```
    family = "binomial"
)

gam.xray <- gam(
    formula = disease ~ s(age, by = Sex) + Sex + Mray + Fray + CnRay.num,
    data = train_data,
    family = "binomial"
)
```

```
cat("Prop. of correctly classified observations of glm.xray.simple:", prop.correct.classifie
```

```
Prop. of correctly classified observations of glm.xray.simple: 0.6521739
```

```
cat("Prop. of correctly classified observations of glm.xray.complex:", prop.correct.classifi
```

```
Prop. of correctly classified observations of glm.xray.complex: 0.6086957
```

```
cat("Prop. of correctly classified observations of glm.xray.very.complex:", prop.correct.cla
```

```
Prop. of correctly classified observations of glm.xray.very.complex: 0.5217391
```

```
cat("Prop. of correctly classified observations of gam.xray:", prop.correct.classified(model
```

```
Prop. of correctly classified observations of gam.xray: 0.5652174
```

Based on the new proportion of correctly classified observations in-sample, we should go on with the model `glm.simple` with a value of 0.65.