MTHM506 - Statistical Data Modelling

Topic 3 - Interactions in GAMs

Preliminaries

In the last session we showed that in practice, the fitting and inference for GAMs is similar to that of GLMs. This session will show another example of GAMs, and in particular how we can model interactions and why sometimes we may need to apply subjective reasoning rather than relying on model inference.

These notes refer to Topics 3.1-3.8 from the lecture notes. All practical aspects will be done using the `munichrent` dataframe. This can be found in `datasets.RData` on the course ELE page and can be loaded into `R` using the `load()` function.

```
# Loading datasets required
load('datasets.RData')
```

We also need the `mgcv`, `ggplot` and `plot3D` packages:

```
# Load required packages into the library
library(mgcv)
library(ggplot2)
library(plot3D)
```

Example: `munichrent` dataframe

The `munichrent` dataframe, contains data on the price of rent per week (in Euro) collected for 3082 apartments in Munich. The dataset also contains covariates:

- `yearc` – the year of construction for the apartments
- `location` – location index (1 indicating a deprived neighbourhood, 2 for average neighbourhood and 3 indicating a affluent neighbourhood)
- `district` – an identifier of the district the flat is located in
- `area` – in square meters

```
# Fist 6 rows of munichrent
head(munichrent)
```

```
      rent   rentsqm area yearc location bath kitchen cheating district
1 120.9744  3.456410   35  1939        1    0       0        0     1112
2 436.9743  4.201676  104  1939        1    1       0        1     1112
```

```
3 355.7436 12.267021   29  1971        2    0       0        1     2114
4 282.9231  7.254436   39  1972        2    0       0        1     2148
5 807.2308  8.321964   97  1985        1    0       0        1     2222
6 482.8205  7.787426   62  1962        1    0       0        1     2222
```

Rent as a function of year and location

One question of interest is in how rent prices vary with the year of construction and the location index.

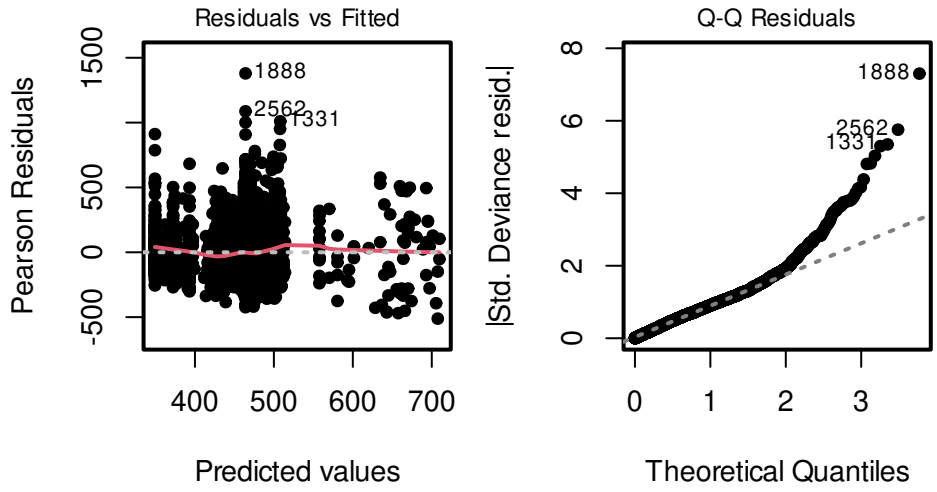
Let's start by fitting a Normal GLM to model rent prices in terms of year of construction and location index and an interaction between them. (We won't write out for simplicity)

```r
# Fit a linear model
model1 <- glm(rent ~ yearc + location + yearc:location,
              data = munichrent,
              family = gaussian(link = 'identity'))

# Model summary
summary(model1)
```

```
Call:
glm(formula = rent ~ yearc + location + yearc:location, family = gaussian(link = "identi
    data = munichrent)

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -3.649e+03  4.215e+02  -8.658  < 2e-16 ***
yearc            2.085e+00  2.151e-01   9.692  < 2e-16 ***
location2        3.031e+03  6.174e+02   4.909 9.65e-07 ***
location3        2.120e+02  1.799e+03   0.118    0.906
yearc:location2 -1.520e+00  3.158e-01  -4.815 1.54e-06 ***
yearc:location3 -1.922e-03  9.199e-01  -0.002    0.998
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 35786.03)

    Null deviance: 117945363  on 3081  degrees of freedom
Residual deviance: 110077841  on 3076  degrees of freedom
AIC: 41070

Number of Fisher Scoring iterations: 2
```

```r
# Check residuals
```

```
par(mfrow=c(1,2),mar = c(4, 4, 1, 1),cex=1.2,lwd=2)
plot(model1,1,pch=20)
plot(model1,2,pch=20)
```

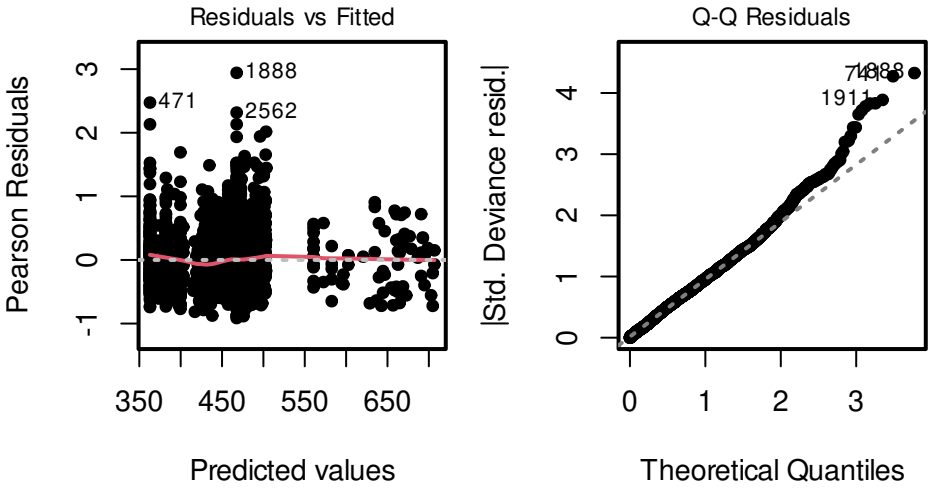


The model fit is quite poor. The QQ plot shows severe deviation from the line which may be due to using the Normal distribution for the strictly positive response `rent`. So let's fit a Gamma GLM with an identity-link

```
model2 <- glm(rent ~ yearc + location + yearc:location,
              data = munichrent,
              family = Gamma(link = 'identity'))

# Model summary
summary(model2)

Call:
glm(formula = rent ~ yearc + location + yearc:location, family = Gamma(link = "identity"
    data = munichrent)

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -2997.4969   374.3483  -8.007 1.65e-15 ***
yearc               1.7520     0.1914   9.154  < 2e-16 ***
location2        2591.1048   601.6008   4.307 1.71e-05 ***
location3        -281.6856  2364.7076  -0.119    0.905
yearc:location2    -1.2963     0.3081  -4.208 2.66e-05 ***
yearc:location3     0.2500     1.2114   0.206    0.837
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Gamma family taken to be 0.1682382)
```

```
    Null deviance: 540.48  on 3081  degrees of freedom
Residual deviance: 506.61  on 3076  degrees of freedom
AIC: 40525

Number of Fisher Scoring iterations: 8
```

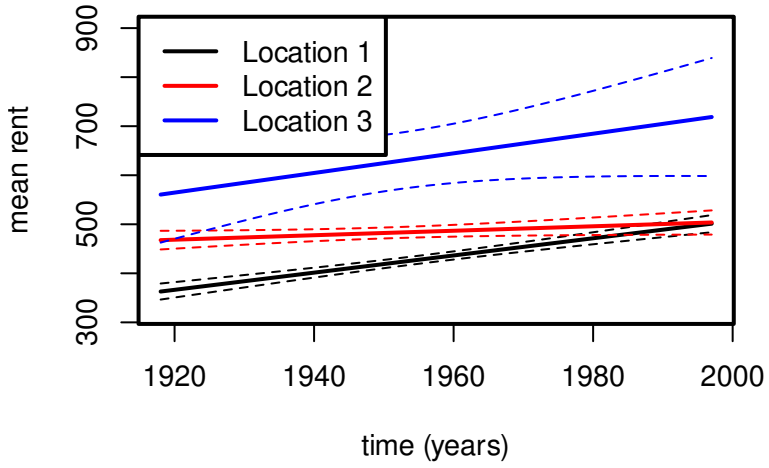
```
# Check residuals
par(mfrow=c(1,2),mar = c(4, 4, 1, 1),cex=1.2,lwd=2)
plot(model2,1,pch=20)
plot(model2,2,pch=20)
```



Unfortunately, this also fits badly. Let's produce some predictions from this model to see what is happening

```
# Look at predicted values for each location
years <- 1918:1997

# Predictions for our three types of location
loc1 <- predict(model2,newdata=data.frame(yearc=years,location=as.factor(1)),se.fit=T)
loc2 <- predict(model2,newdata=data.frame(yearc=years,location=as.factor(2)),se.fit=T)
loc3 <- predict(model2,newdata=data.frame(yearc=years,location=as.factor(3)),se.fit=T)

# Plot margins
par(mar = c(4, 4, 1, 1),cex=1.2,lwd=2)

# Predictions for location 1
plot(years,loc1$fit,type="l",ylim=c(320,900),xlab="time (years)",ylab="mean rent")
lines(years,loc1$fit+1.96*loc1$se.fit,lty=2,lwd=1)
lines(years,loc1$fit-1.96*loc1$se.fit,lty=2,lwd=1)

# Predictions for location 3
lines(years,loc2$fit,col="red")
```

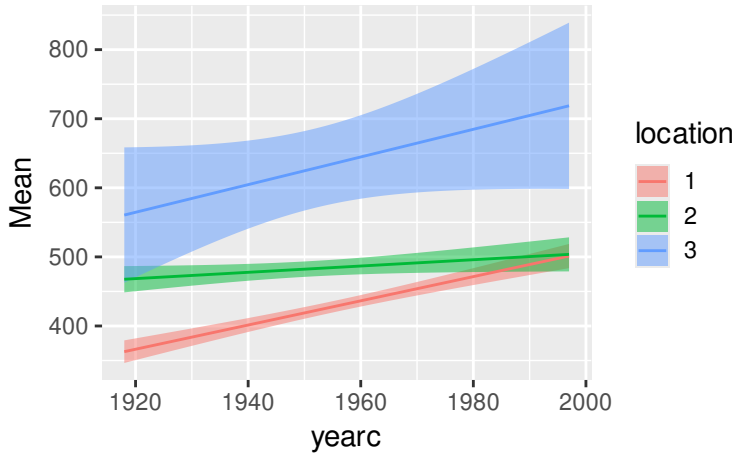
```
lines(years,loc2$fit+1.96*loc2$se.fit,lty=2,lwd=1,col="red")
lines(years,loc2$fit-1.96*loc2$se.fit,lty=2,lwd=1,col="red")

# Predictions for location 3
lines(years,loc3$fit,col="blue")
lines(years,loc3$fit+1.96*loc3$se.fit,lty=2,lwd=1,col="blue")
lines(years,loc3$fit-1.96*loc3$se.fit,lty=2,lwd=1,col="blue")

# add a legend
legend("topleft",c("Location 1","Location 2","Location 3"),col=c("black","red","blue"),l
```



Much easier in ggplot:

```
# create a grid of years and locations
plotDat <- expand.grid(years,1:3)
plotDat <- data.frame(plotDat)
names(plotDat) <- c("yearc","location")
plotDat$location <- factor(plotDat$location)
head(plotDat)
  yearc location
1  1918        1
2  1919        1
3  1920        1
4  1921        1
5  1922        1
6  1923        1

# predict mean and standard errors
preds <- predict(model2,newdata=plotDat,se.fit=T)

# put in the dataframe
plotDat$Mean <- preds$fit
```

```
plotDat$Upper <- preds$fit + 1.96*preds$se.fit
plotDat$Lower <- preds$fit - 1.96*preds$se.fit

# and plot
myPlot <- ggplot(data=plotDat) +
  geom_ribbon(aes(x=yearc,ymin=Lower,ymax=Upper,fill=location),alpha=0.5) +
  geom_line(aes(x=yearc,y=Mean,col=location))
myPlot
```



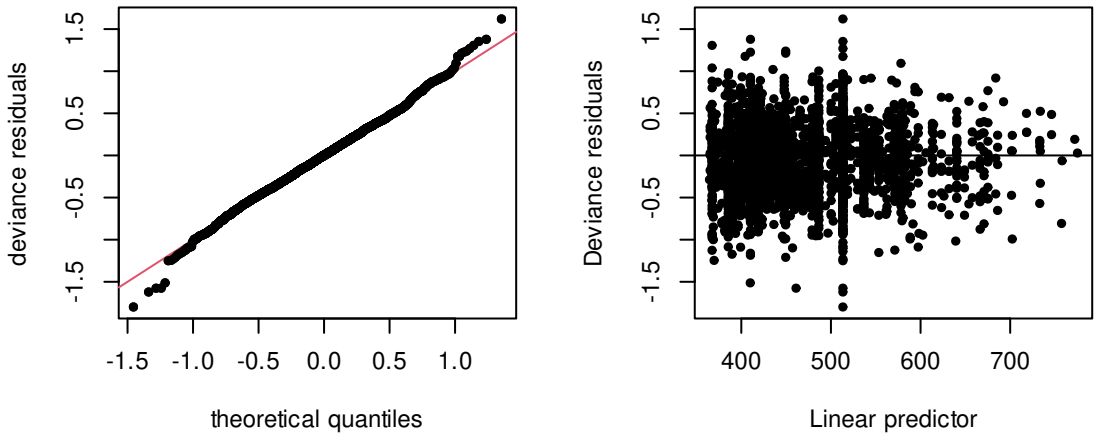
So, a different linear relationship between rent price and year of construction for each location. The question then becomes, **are these relationships linear**? It's convenient to model them as linear but they may be non-linear.

Interaction between a smooth function and a factor

Let's investigate the non-linearity using GAMs, where we would need a model that involves an interaction between a smooth function for `yearc` and factor `location`. In `mgcv`, such an interaction can be achieved via the `by=` argument within the function `s()`.

```
# Fit as a GAM
model3 <- gam(rent ~ location + s(yearc, by = location, k = 10),
              data = munichrent,
              family = Gamma(link="identity"))

# check rank
k.check(model3)
                    k'      edf   k-index p-value
s(yearc):location1  9 5.774746 0.9512249  0.0200
s(yearc):location2  9 3.616925 0.9512249  0.0300
s(yearc):location3  9 4.795705 0.9512249  0.0275

## Check residuals
```

```
par(mfrow=c(1,2))
# QQ plot
qq.gam(model3,pch=20)
# deviance residuals vs linear predictor
xx <- model3$linear.predictors
yy <- residuals(model3,type="deviance")
plot(xx,yy,pch=20,xlab="Linear predictor",ylab="Deviance residuals")
abline(h=0)
```



The `by=` argument means that we have one smooth function for each level of the factor `location`. (Be sure that the covariate is designated as a factor in `R` otherwise the `by=` argument will do something else.) Let

$$z_{1,i} = \begin{cases} 1 & \text{location 1} \\ 0 & \text{otherwise} \end{cases} \qquad z_{2,i} = \begin{cases} 1 & \text{location 2} \\ 0 & \text{otherwise} \end{cases} \qquad z_{3,i} = \begin{cases} 1 & \text{location 3} \\ 0 & \text{otherwise} \end{cases}$$

so the model we are fitting is mathematically:

$$\mu_i = \beta_0 + \beta_2 z_{2,i} + \beta_3 z_{3,i} + z_{1,i} f_1(x_i) + z_{2,i} f_2(x_i) + z_{3,i} f_3(x_i)$$

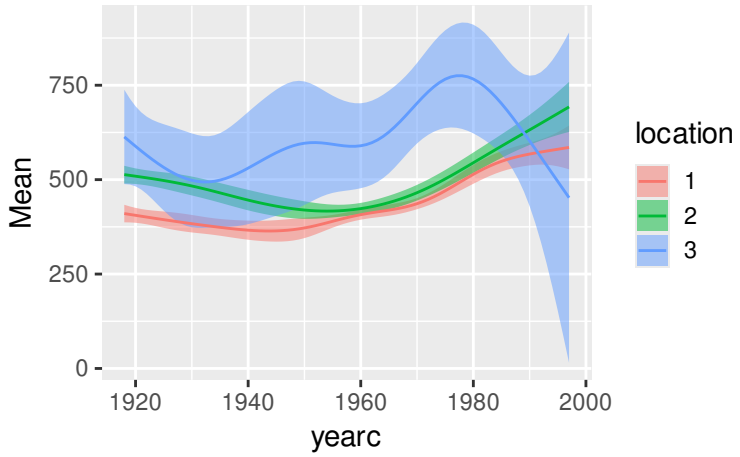
In other words,

$$\mu_i = \begin{cases} \beta_0 + f_1(x_i) & \text{location 1} \\ \beta_0 + \beta_2 + f_2(x_i) & \text{location 2} \\ \beta_0 + \beta_3 + f_3(x_i) & \text{location 3} \end{cases}$$

Note that to save space we have used `k.check` and manually created the two residual plots, rather than use `gam.check`. The residual plots are looking much better, in particular the QQ plot – although some deviation evident at both extreme ends. The other plot is also fine, although a little funneling is evident. We could try adding more covariates to fix some of these issues, but for now let's proceed with visualising the estimates.

There are 3 smooth functions so we get 3 rows in `k.check`, one for each location. The first two of these functions look like they have enough flexibility as the EDF is not close to k', but the third function looks like it may need more flexibility as the EDF is close to k'. Let's look at the estimates:

```r
# predict mean and standard errors
preds <- predict(model3,newdata=plotDat,se.fit=T)

# put in the dataframe
plotDat$Mean <- preds$fit
plotDat$Upper <- preds$fit + 1.96*preds$se.fit
plotDat$Lower <- preds$fit - 1.96*preds$se.fit

# and plot
myPlot <- ggplot(data=plotDat) +
  geom_ribbon(aes(x=yearc,ymin=Lower,ymax=Upper,fill=location),alpha=0.5) +
  geom_line(aes(x=yearc,y=Mean,col=location))
myPlot
```



Relationships are quite non-linear especially location 3. Best to not increase the rank as the function is probably too "wiggly" as is. Despite penalisation, too much flexibility might mean that the functions pick up **noise** rather than **trend**, and given the confidence intervals, this may be the case in location 3.

Interaction between two (or more) quantitative covariates.

One key piece of information in the `munichrent` dataframe is the area of each property in m^2. This of course plays an important role in the amount of rent. Suppose that we are specifically interested in whether

- how the rent amount relates to year of construction and how does this relationship vary with the area of each property?

In other words, we want to see if there is an interaction between `yearc` and `area` in a model for `rent`. Let's stick with the Gamma distribution and formulate a GAM to that effect. However, let's also use the log-link, which is actually the more sensible link to choose.

In GAMs, we can construct a function of more than one variable, say $f(x_1, x_2)$ as a **tensor product interaction**. Fortunately, this is mathematically quite straightforward. We saw how splines can

be used to construct smooth functions:

$$f(x) = \sum_{k=1}^{K} \beta_k b_k(x)$$

If we were to make the coefficients of $f(x)$ themselves smooth functions of another covariate, say z, then:

$$\beta_k(z) = \sum_{l=1}^{L} \gamma_{k,l} b_l(z)$$

then we well get a 2D amooth function:

$$\begin{aligned} f(x,z) &= \sum_{k=1}^{K} \beta_k(z) b_k(x) \\ &= \sum_{k=1}^{K} \sum_{l=1}^{L} \gamma_{k,l} b_l(z) b_k(x) \end{aligned}$$

which is just another **linear predictor** of coefficients $\gamma_{k,l}$ and splines $b_l(z)b_k(x)$.

Let's see this in action for our rent model. Our mean is now formulated as

$$\log(\mu_i) = \beta_0 + f(x_{1,i}, x_{2,i})$$

where $x_{1,i}$ is `yearc` and $x_{2,i}$ is `area`. Let's fit this using `gam()`:

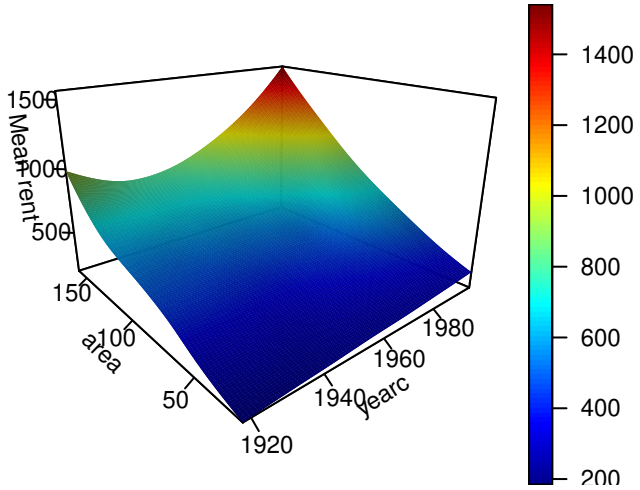
```
# Fit as a GAM
model4 <- gam(rent ~ te(yearc,area,k=c(10,10),bs=c("cs","cs")),
              data = munichrent,
              family = Gamma(link="log"))

# Model summary
summary(model4)

Family: Gamma
Link function: log

Formula:
rent ~ te(yearc, area, k = c(10, 10), bs = c("cs", "cs"))

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.090745   0.005385    1131   <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
```

```
te(yearc,area) 22.54     98 26.94  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.461   Deviance explained = 44.8%
GCV = 0.098349  Scale est. = 0.089385  n = 3082

# check rank
k.check(model4)
               k'      edf   k-index p-value
te(yearc,area) 99 22.54458 0.9756435   0.175
```

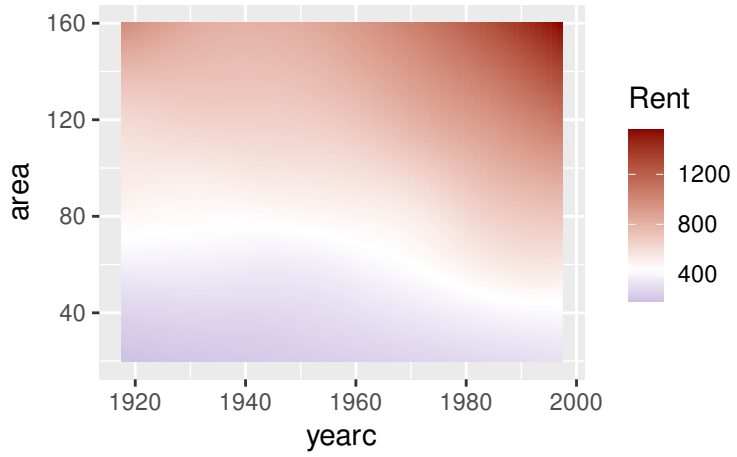
With `bs=c("cs","cs")` we specified a cubic spline basis for each covariate. We have $K = 10$ for `yearc` and another `k=10` for `area` so that's $10 \times 10 = 100$ coefficients minus 1 for the sum-to-zero constraint given `k'` equal to 99. Seems like we have a lot more than we need so let's proceed to understand what this model has estimated.

```
# create a grid area values
range(munichrent$area)
[1]  20 160
areas <- 20:160
# grid of years
range(munichrent$yearc)
[1] 1918 1997
years <- 1918:1997
# grid of area and years
plotDat <- expand.grid(years,areas)
plotDat <- data.frame(plotDat)
names(plotDat) <- c("yearc","area")
head(plotDat)
  yearc area
1  1918   20
2  1919   20
3  1920   20
4  1921   20
5  1922   20
6  1923   20

# predict mean and standard errors
preds <- predict(model4,newdata=plotDat,se.fit=T,type="link")

# put in the dataframe
plotDat$Mean <- exp(preds$fit)
plotDat$Upper <- exp( preds$fit + 1.96*preds$se.fit )
plotDat$Lower <- exp( preds$fit - 1.96*preds$se.fit )
```

```
# and plot as a 3D surface
par(mar=c(0,2,0,6))
persp3D(years, areas, matrix(plotDat$Mean, length(years), length(areas)),
        theta = 320, phi = 20,ticktype = "detailed", xlab = "yearc", ylab = "area",
        zlab = "Mean rent", expand = 2/3, shade = 0.5,main="")
```




```
# Easier in ggplot as a 2D raster plot
myPlot <- ggplot(data=plotDat) +
  geom_raster(aes(x=yearc,y=area,fill=Mean)) +
  scale_fill_gradient2(midpoint=exp(model4$coefficients[1]),high="darkred",
    low="darkblue",name="Rent")
myPlot
```



Plots indicate a non-linear interaction between area and year. The white line in the 2D plot relates to the overall mean rent, which is $\exp\{\beta_0\}$ since $f(x_1, x_2)$ is centered on zero. This should of course be close to the sample mean rent:

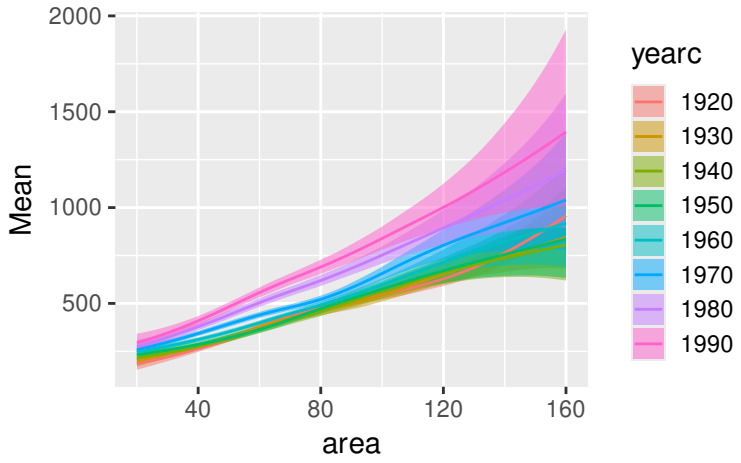
```
mean(munichrent$rent)
[1] 459.4372
```

```
exp( model4$coefficients[1] )
(Intercept)
   441.7503
```

but not necessarily the same.

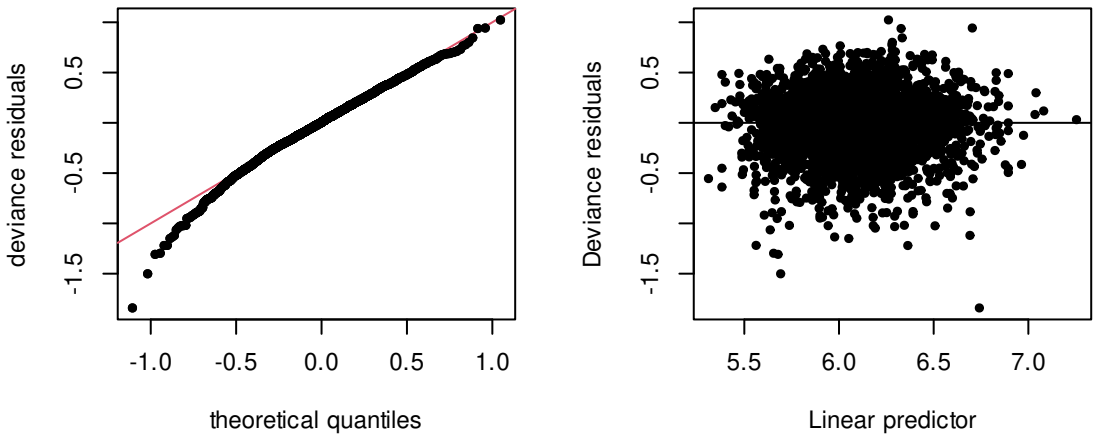
To include uncertainty, we could take "slices" of the surface for specific years, and plot the estimated mean rent against area:

```
# index to pick up only the decades in plotDat
index <- plotDat$yearc%%10==0

# create a new dataframe
plotDatDecades <- plotDat[index,]

# make yearc a factor for ggplot
plotDatDecades$yearc <- factor(plotDatDecades$yearc)

# Plot mean rent vs area for each area
ggplot(data=plotDatDecades) +
  geom_ribbon(aes(x=area,ymax=Upper,ymin=Lower,fill=yearc),alpha=0.5) + #
  geom_line(aes(x=area,y=Mean,col=yearc))
```



All this is nice but does the model fit well?

```
## Check residuals
par(mfrow=c(1,2))
# QQ plot
qq.gam(model4,pch=20)
# deviance residuals vs linear predictor
xx <- model4$linear.predictors
yy <- residuals(model4,type="deviance")
plot(xx,yy,pch=20,xlab="Linear predictor",ylab="Deviance residuals")
abline(h=0)
```



Not looking good. QQ plot indicates problems for very low values, while the residuals vs linear predictor plot indicates some funnelling at high values. Again, we can fix this by including some of the other covariates, but this is left as an exercise.