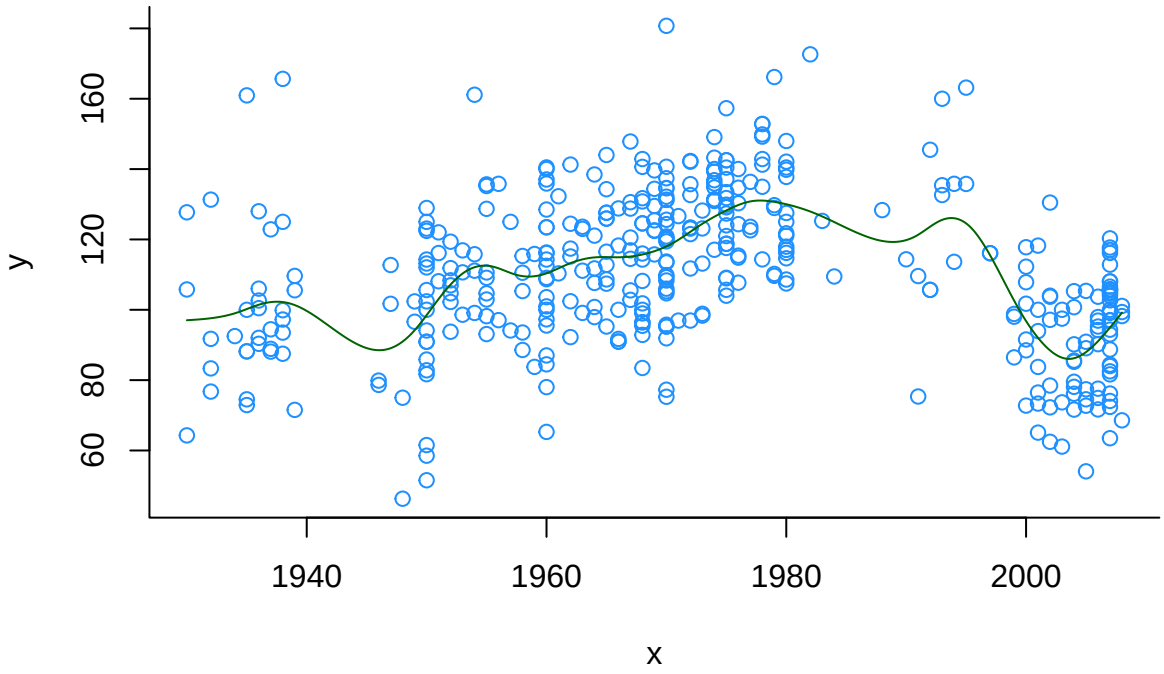
Semiparametric regression - Homework 4

Klaudia Weigel

1 Exercise 1

We will fit a penalized spline model to the `WarsawApts` data. Then we will plot the datapoints together with the fitted line to see how our model performs. In the first fit we use 30 cubic spline basis.

```
library(HRW)
library(mgcv)
data(WarsawApts)
x <- WarsawApts$construction.date
y <- WarsawApts$areaPerMzloty
plot(x, y, bty = 'l', col = 'dodgerblue')
fitGAMcr <- gam(y~s(x, bs = 'cr', k = 30))
xg <- seq(min(x), max(x), length = 1001)
fHatgGAMcr <- predict(fitGAMcr, newdata = data.frame(x = xg))
lines(xg, fHatgGAMcr, col = 'darkgreen')
```



The line follows the trends in the data.

1.1 (a)

We will now obtain fits with different basis functions, namely:

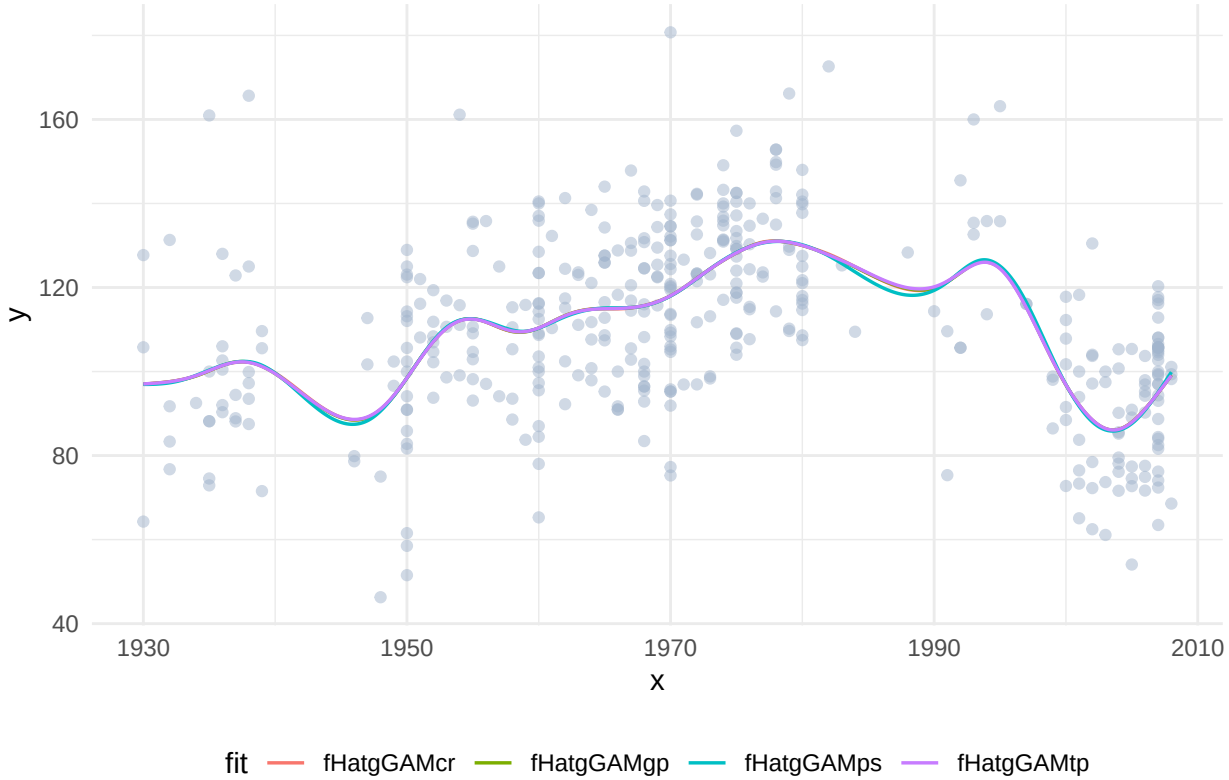
- Gaussian process basis functions (`bs = 'gp'`),
- P-splines (`bs = 'ps'`),
- thin plate regression splines (`bs = 'tp'`).

```
library(ggplot2); library(tidyr)

fitGAMgp <- gam(y~s(x, bs = 'gp', k = 30))
fitGAMps <- gam(y~s(x, bs = 'ps', k = 30))
fitGAMtp <- gam(y~s(x, bs = 'tp', k = 30))

fHatgGAMgp <- predict(fitGAMgp, newdata = data.frame(x = xg))
fHatgGAMps <- predict(fitGAMps, newdata = data.frame(x = xg))
fHatgGAMtp <- predict(fitGAMtp, newdata = data.frame(x = xg))

data_plot <- data.frame(cbind(xg, fHatgGAMcr, fHatgGAMgp, fHatgGAMps, fHatgGAMtp))
data_plot %>%
  gather('fit','value', -xg) %>%
  ggplot() +
  geom_point(data = data.frame(cbind(x,y)), aes(x = x, y = y),
             color = 'lightsteelblue3', alpha=0.5) +
  geom_line(aes(x = xg, y = value, color = fit), size = 0.6) +
  theme_minimal() +
  theme(legend.position = 'bottom')
```



As we can see the fits are similar regardless of the choice of the basis functions.

1.2 (b)

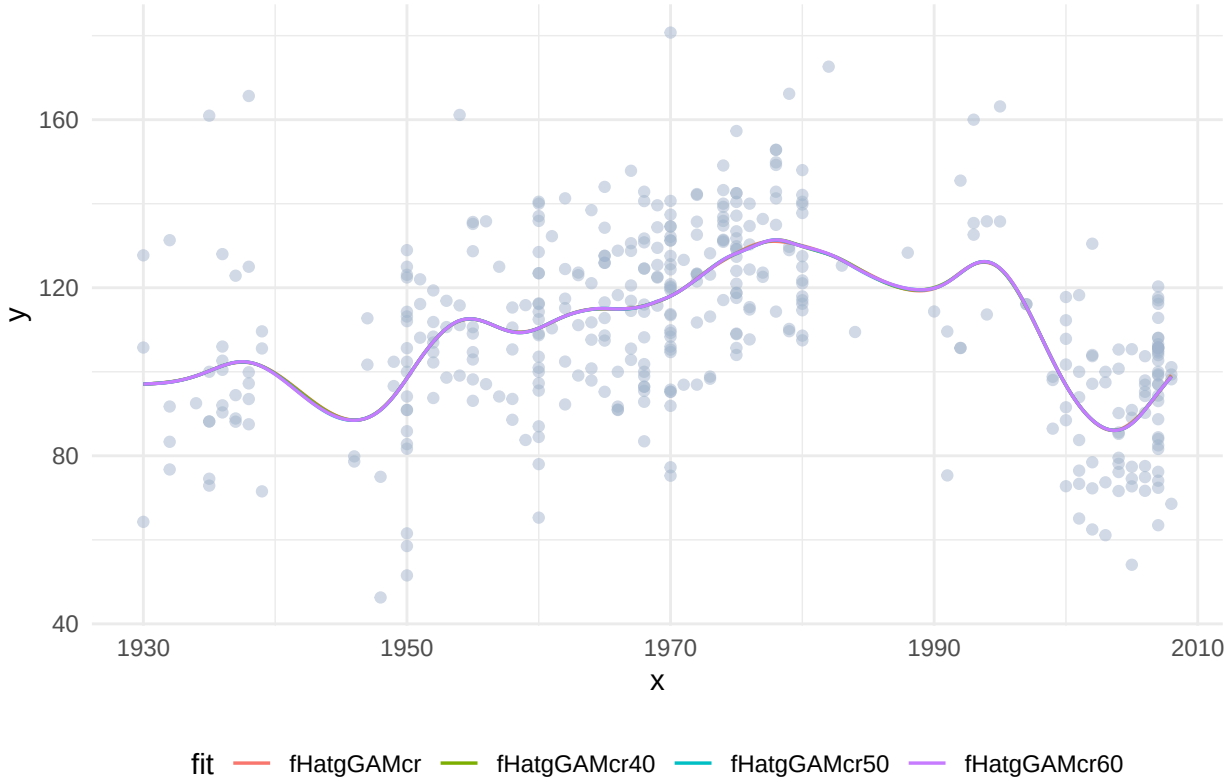
We will obtain fits with other numbers of basis functions:

- `k = 40`,
- `k = 50`,
- `k = 60`.

```
fitGAMcr40 <- gam(y~s(x, bs = 'cr', k = 40))
fitGAMcr50 <- gam(y~s(x, bs = 'cr', k = 50))
fitGAMcr60 <- gam(y~s(x, bs = 'cr', k = 60))

fHatgGAMcr40 <- predict(fitGAMcr40, newdata = data.frame(x = xg))
fHatgGAMcr50 <- predict(fitGAMcr50, newdata = data.frame(x = xg))
fHatgGAMcr60 <- predict(fitGAMcr60, newdata = data.frame(x = xg))

data_plot <- data.frame(cbind(xg, fHatgGAMcr, fHatgGAMcr40, fHatgGAMcr50, fHatgGAMcr60))
data_plot %>%
  gather('fit','value', -xg) %>%
  ggplot() +
  geom_point(data = data.frame(cbind(x,y)), aes(x = x, y = y),
             color = 'lightsteelblue3', alpha=0.5) +
  geom_line(aes(x = xg, y = value, color = fit), size = 0.6) +
  theme_minimal() +
  theme(legend.position = 'bottom')
```



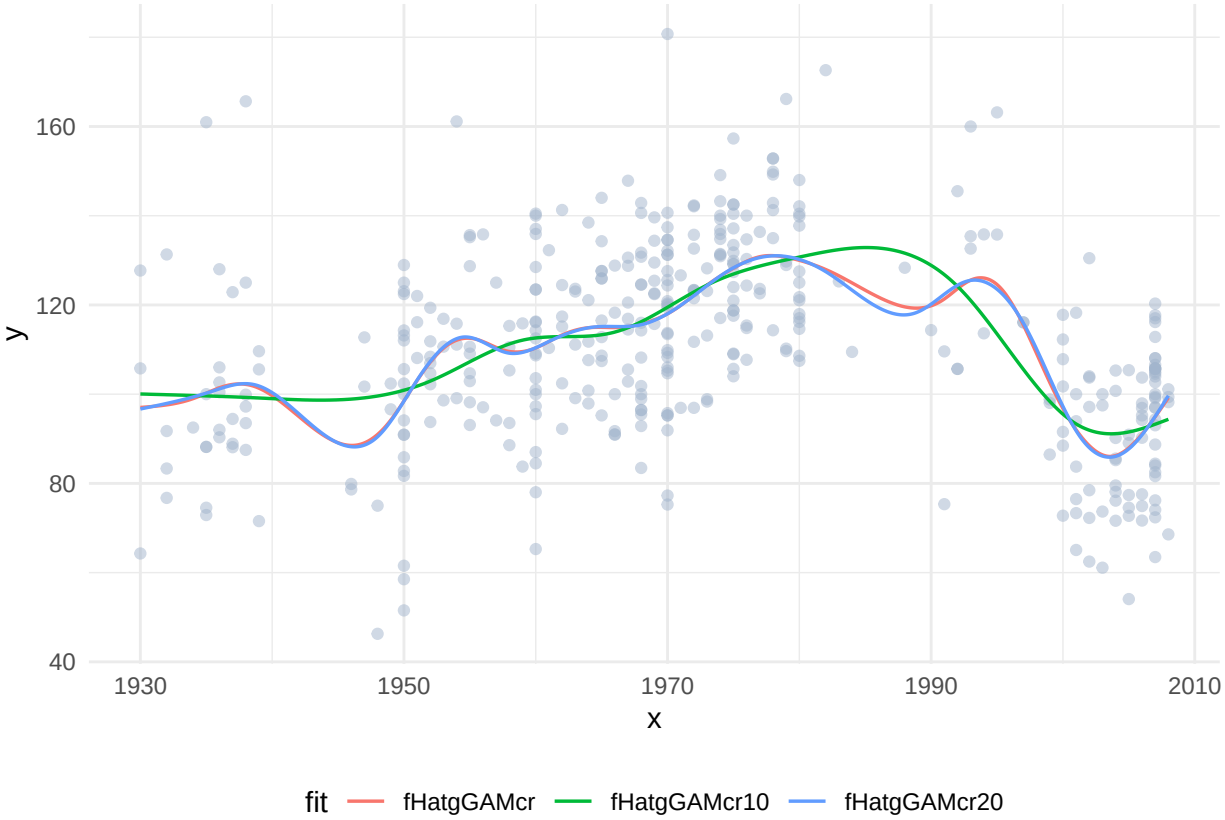
There is no difference between the fitted lines. This indicates that the model does not significantly benefit from added dimensions and considering increased computational complexity we should choose as small a `k` as possible.

To get a further insight we can also check what happens if we decrease the number of basis functions.

```
fitGAMcr10 <- gam(y~s(x, bs = 'cr', k = 10))
fitGAMcr20 <- gam(y~s(x, bs = 'cr', k = 20))

fHatgGAMcr10 <- predict(fitGAMcr10, newdata = data.frame(x = xg))
fHatgGAMcr20 <- predict(fitGAMcr20, newdata = data.frame(x = xg))

data_plot <- data.frame(cbind(xg, fHatgGAMcr10, fHatgGAMcr20, fHatgGAMcr))
data_plot %>%
  gather('fit','value', -xg) %>%
  ggplot() +
  geom_point(data = data.frame(cbind(x,y)), aes(x = x, y = y),
             color = 'lightsteelblue3', alpha=0.5) +
  geom_line(aes(x = xg, y = value, color = fit), size = 0.6) +
  theme_minimal() +
  theme(legend.position = 'bottom')
```



The fit with 10 basis functions is more general than the other two. The fits with both 20 and 30 functions are quite similar.

1.3 (c)

Based on point (a) and (b) we can conclude that for this particular data the choice of the penalized smoothing basis does not matter greatly. Also the optimal number of basis functions seems to be around 30. Adding more functions does not significantly change the result, as can be seen from the plots. Also by adding more redundant functions we unnecessarily increase the complexity of our model. Regarding the choice of k we may also use the function `k.check`:

```
res_kcheck = rbind( k.check(fitGAMcr10),
                    k.check(fitGAMcr20),
                    k.check(fitGAMcr),
                    k.check(fitGAMcr40),
                    k.check(fitGAMcr50),
                    k.check(fitGAMcr60))
rownames(res_kcheck) = NULL
```

Table 1: Type I error.

k'	edf	k-index	p-value
9	8.049593	0.9508549	0.1875
19	13.522284	1.0045830	0.5275
29	14.129568	1.0056251	0.5275
39	14.400567	1.0068828	0.5325
49	14.563729	1.0073933	0.5425
59	14.611683	1.0075586	0.5525

Low p-values may indicate that the basis dimension, k, has been set too low. Here we see that the p-values do not differ by much between k=20, k=30, k=40, k=50 and k=60. We should therefore choose a model with the smallest feasible k, and as has been already mentioned for this particular data that k would be around 30. However, it might be worth pointing out that in some time periods there is very little data provided and we also seem to be dealing with some outliers. If given more information or a training set it might turn out that we are overfitting to the current shape of the signal. It is not out of the question therefore to actually consider a smaller k and the corresponding fit which captures a more general relationship.