Simítás, spline-regresszió, additív modellek

Ferenci Tamás, tamas.ferenci@medstat.hu

2020. június 16.

Tartalom

Előszó

Ez a jegyzet simítóeljárásokkal (pl. LOESS), a spline-okkal, és azok regresszióban történő felhasználásával, valamint általában az additív modellekkel foglalkozik.

Téma: simítás, spline-regresszió, additív modellek

Ajánlott irodalom: Simon N. Wood: Generalized Additive Models: an introduction with R (Chapman & Hall/CRC, Texts in Statistical Science sorozat, 2. kiadás, ISBN 9781498728331, 2017).

Az olvasó a kísérlet eredményét a jegyzet végére megítélheti. Én mindenesetre – különösen azért, mert ez egy kísérlet – minden visszajelzést, véleményt, kritikát a lehető legnagyobb örömmel veszek a tamas.ferenci@medstat.hu email-címen!

Minden visszajelzést örömmel veszek a tamas.ferenci@medstat.hu email-címen

A jegyzet weboldala (oktatási segédanyagokkal, technikai információkkal) a https://github.com/tamas-ferenci/FerenciTamas_SmoothingSplinesGAM címen érhető el.

1. fejezet

A LOESS simító

A LOESS simítóról lesz szó

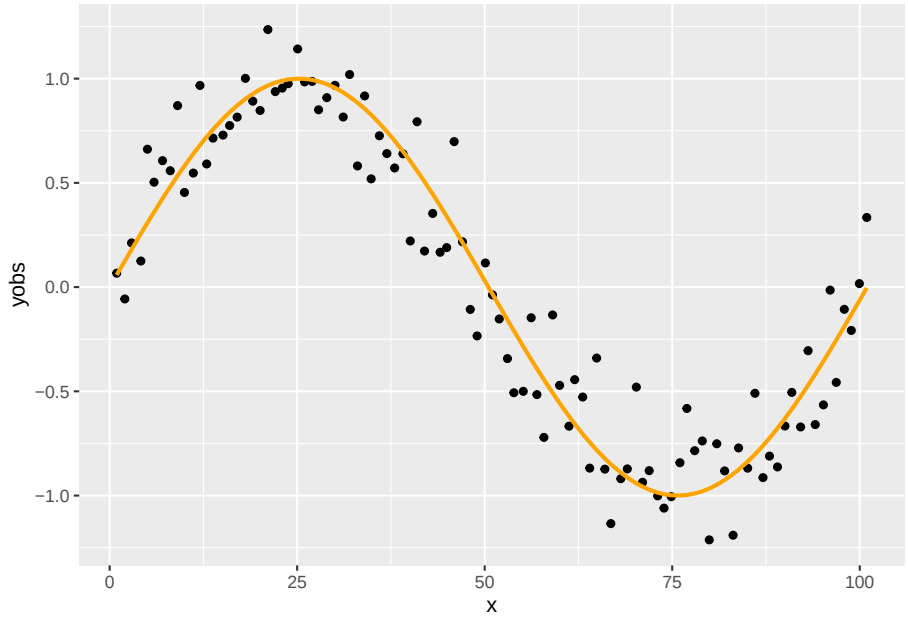
Ehhez az anyaghoz egyedül a `ggplot2` könyvtárra lesz szükségünk (illetve beállítjuk a véletlenszám-generátor seed-jét a reprodukálhatóság kedvéért):

```
library(ggplot2)
set.seed(1)
```

1.1. Motiváció

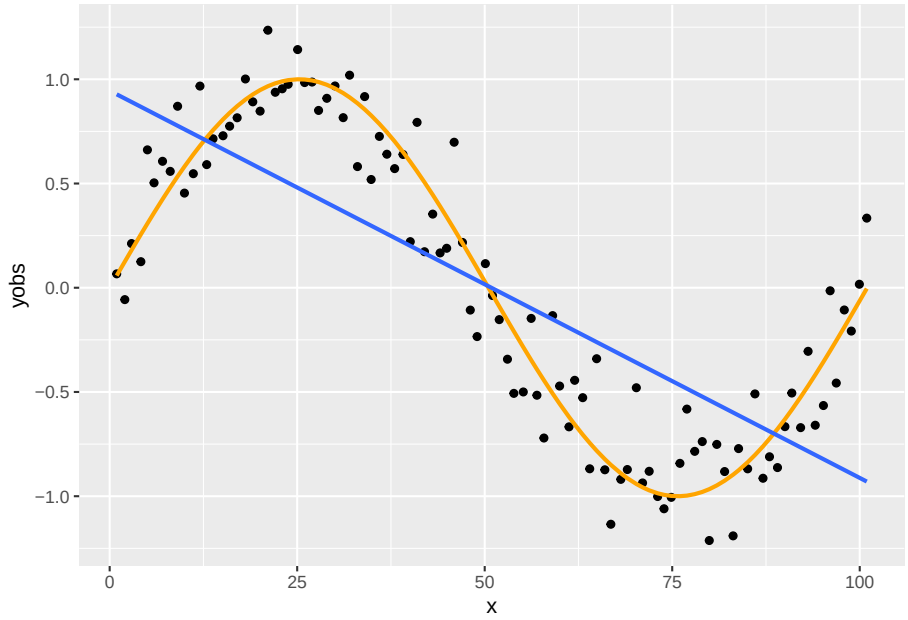
Első lépésben előkészítünk egy demonstrációs adatbázist. Szimulált adatokat fogunk használni (zajos szinusz), így mi is tudni fogjuk, hogy mi az „igazság”, a valódi függvény amiből a pontok jöttek:

```
n <- 101
x <- (1:n) + rnorm(n, 0, 0.1)
y <- sin(x/n*(2*pi))
yobs <- y + rnorm(n, 0, 0.2)
SimData <- data.frame(x, y, yobs)
p <- ggplot(SimData, aes(x = x, y = yobs)) + geom_point() +
  geom_line(aes(y = y), color = "orange", lwd = 1)
p
```



Paraméteres görbeillesztésnél fel kell tételeznünk egy függvényformát (ti. ami a pontok mögött van a valóságban). Például, hogy lineáris:

```
p + geom_smooth(formula = y~x, method = "lm", se = FALSE)
```



Ez az ábra jól mutatja ennek a fő problémáját: hogy ezt a feltételezést el is ronthatjuk! Természetesen vannak diagnosztikai eszközök ennek felderítésére, és ez alapján kereshetünk jobb függvényformát, de gyökerestül csak az oldja meg a problémát, ha olyan módszert találnánk, ami a nélkül működik, hogy egyáltalán fel kelljen tételeznie (bármilyen) függvényformát. Ezt oldják meg a simítási eljárások. (Lényegében nemparaméteres regresszióról van szó.)

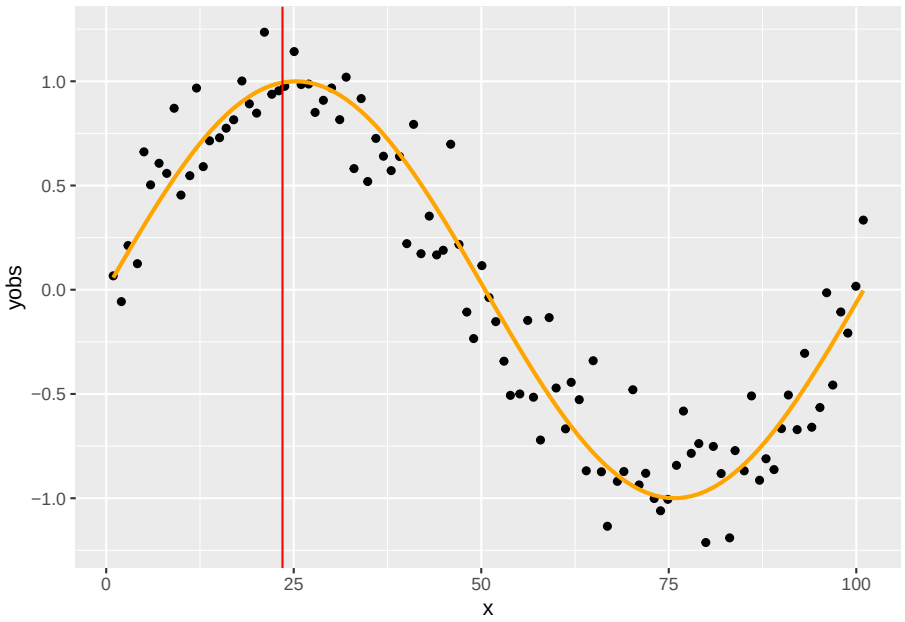
Annak az előnynek, hogy nem kell ilyen feltételezéssel élnünk (és így azt el sem ronthatjuk), természetesen ára van: kevésbé hatásosan becsülhető, mint a paraméteres illesztés, nincsenek számszerű paramétereink (aminek esetleg tárgyterületi interpretációt lehet adni), csak „ábrát tudunk rajzolni", és végezetül extrapoláció sem lehetséges, legalábbis nem triviálisan.

1.2. A LOESS simító alapgondolata

Az egyik legnépszerűbb megoldás a LOESS (locally weighted scatterplot smoothing, néha LOWESS, vagy Savitzky–Golay szűrő), melynek alapgondolata, hogy végigmegy az x-változó releváns tartományán, és minden értékre meghatározza a pontfelhő ottani, tehát lokális közelítését, egy polinomális regresszióból. Utána az egész simítást ezekből a darabkákból építi fel.

Legyen például a vizsgált érték a 23.5:

```
p + geom_vline(xintercept = 23.5, color = "red")
```



1.3. Lokalitás

A lokalitást két eszközzel érjük el. Az egyik, hogy nem használjuk az összes pontot, csak a vizsgált értékhez legközelebb eső α hányadát (ha ez nem egész lenne, akkor felső egészrészt veszünk); ezt a paramétert szokták `span`-nek nevezni. Például, ha ez 75%, akkor a távolság ameddig figyelembe vesszük a pontokat:

```
span <- 0.75
n*span
```

```
## [1] 75.75
```

```
ceiling(n*span)
```

```
## [1] 76
```

```
sort(abs(x-23.5))
```

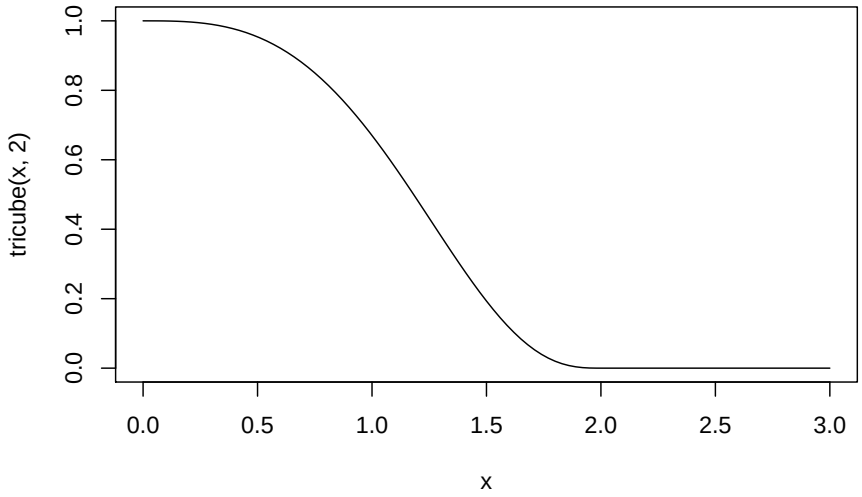
```
##   [1]  0.3010648  0.4925435  1.4217864  1.5619826  2.4081023  2.4943871  3.4406099
##  [13]  6.5016190  6.5417942  7.5044934  7.6358680  8.3875069  8.4897212  9.5387672
##  [25] 12.3488219 12.4585005 13.4605710 13.5305388 14.4424219 14.4940687 15.4261675
##  [37] 18.4670492 18.4746638 19.3404719 19.5696963 20.5556663 20.5835629 21.4311244
##  [49] 25.4887654 26.5881108 27.5398106 28.4387974 29.5341120 30.3870637 31.6433024
##  [61] 37.7401618 38.4960760 39.5689739 40.5028002 41.4256727 42.5188792 43.3195041
##  [73] 49.5610726 50.4065902 51.3746367 52.5291446 53.4556708 54.5001105 55.5074341
##  [85] 61.5593946 62.5332950 63.6063100 64.4695816 65.5370019 66.5267099 67.4457480
##  [97] 73.3723408 74.4426735 75.3775387 76.4526599 77.4379633
```

```
sort(abs(x-23.5))[ceiling(n*span)]
```

```
## [1] 52.52914
```

A másik eszköz, hogy még a megtartott pontokon belül is súlyozunk: minél távolabb esik egy pont a vizsgált értéktől annál kisebb lesz a súlya. Általában a trikubikus súlyfüggvényt használjuk:

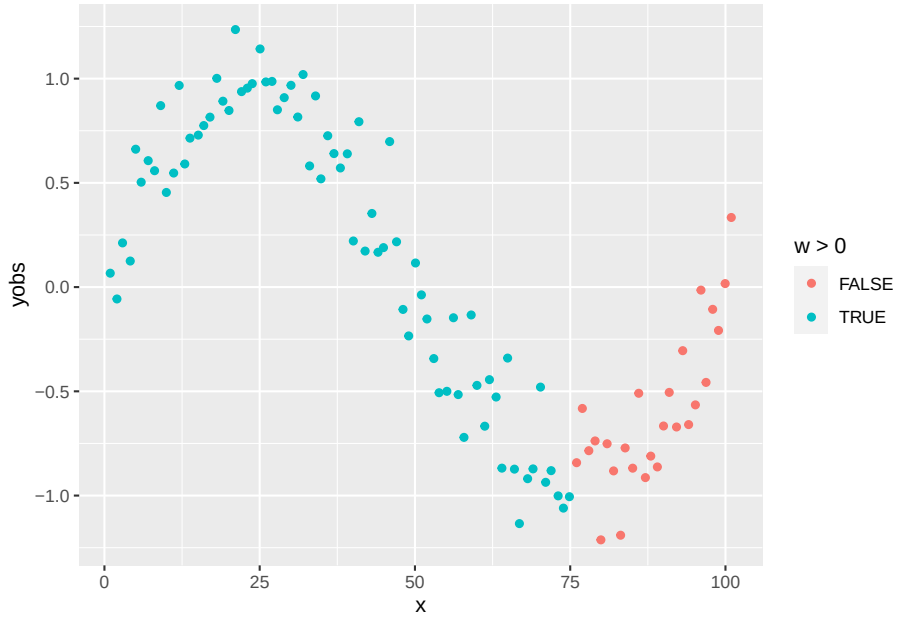
```
tricube <- function(u, t) ifelse(u<t, (1-(u/t)^3)^3, 0)
curve(tricube(x, 2), to = 3)
```



(A fenti definícióval természetesen a kétféle lépés együtt van benne a `tricube` függvényben.)

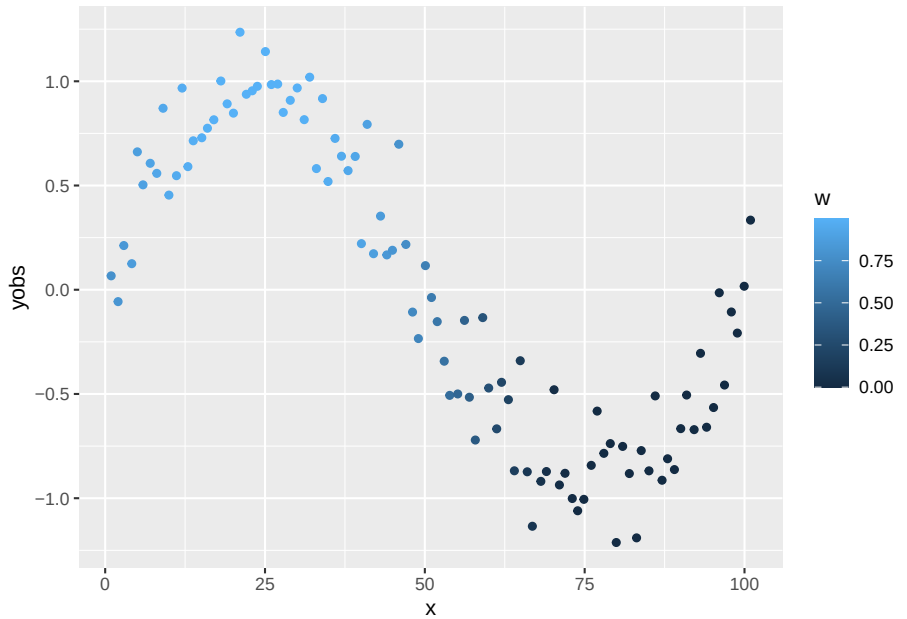
A felhasznált pontok:

```
SimData$w <- tricube(abs(x-23.5), sort(abs(x-23.5))[ceiling(n*span)])
ggplot(SimData, aes(x = x, y = yobs, color = w>0)) + geom_point()
```



A súlyozás:

```
ggplot(SimData, aes(x = x, y = yobs, color = w)) + geom_point()
```

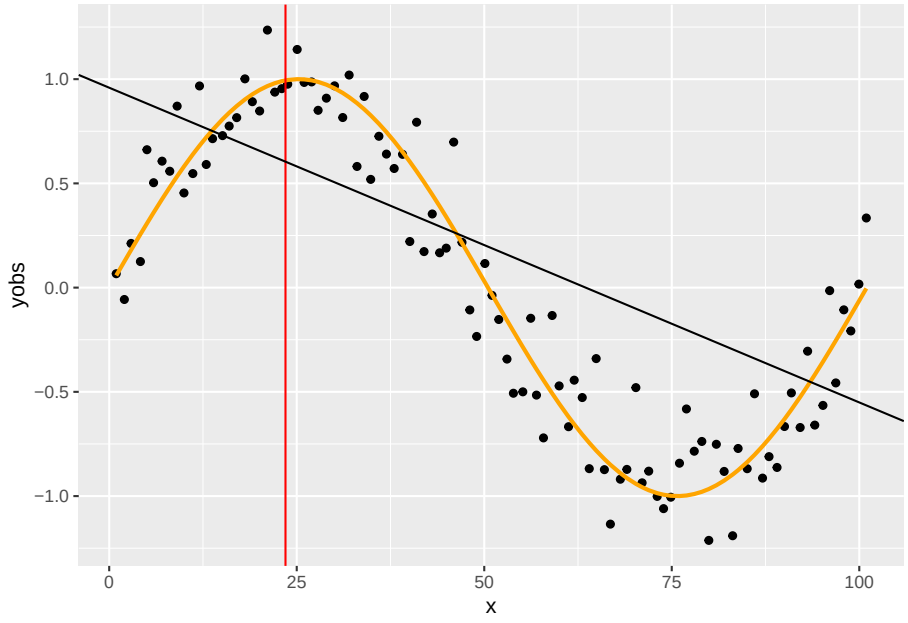


1.4. Polinomiális regresszió

A leszűkített és átsúlyozott ponthalmazra – ezt most tehát egyben tartalmazza a `w` – egy polinomális regressziót illesztünk.

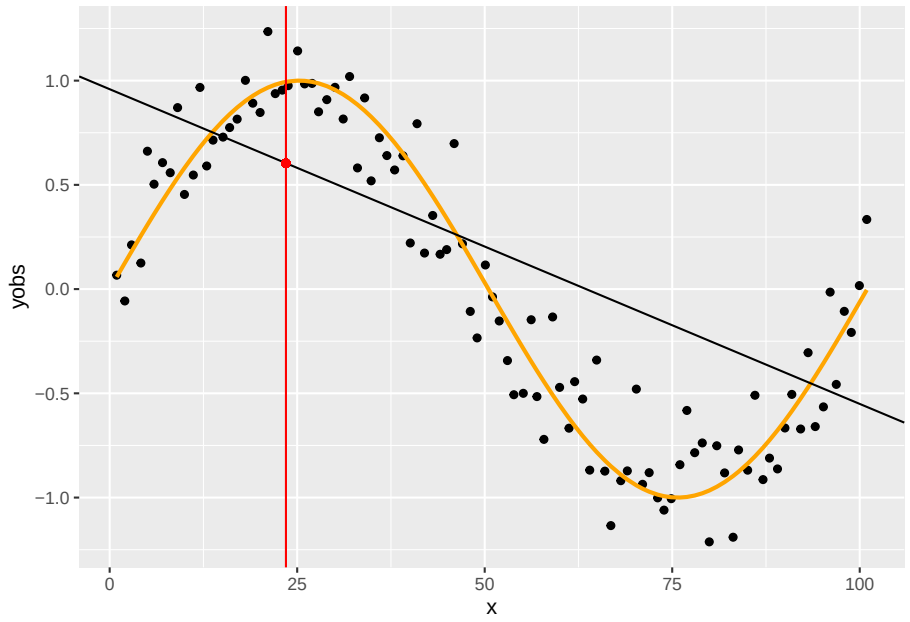
Legegyszerűbb esetben ez lineáris regresszió:

```
fit <- lm(yobs ~ x, weights = w, data = SimData)
p + geom_vline(xintercept = 23.5, color = "red") +
  geom_abline(intercept = coef(fit)[1], slope = coef(fit)[2])
```



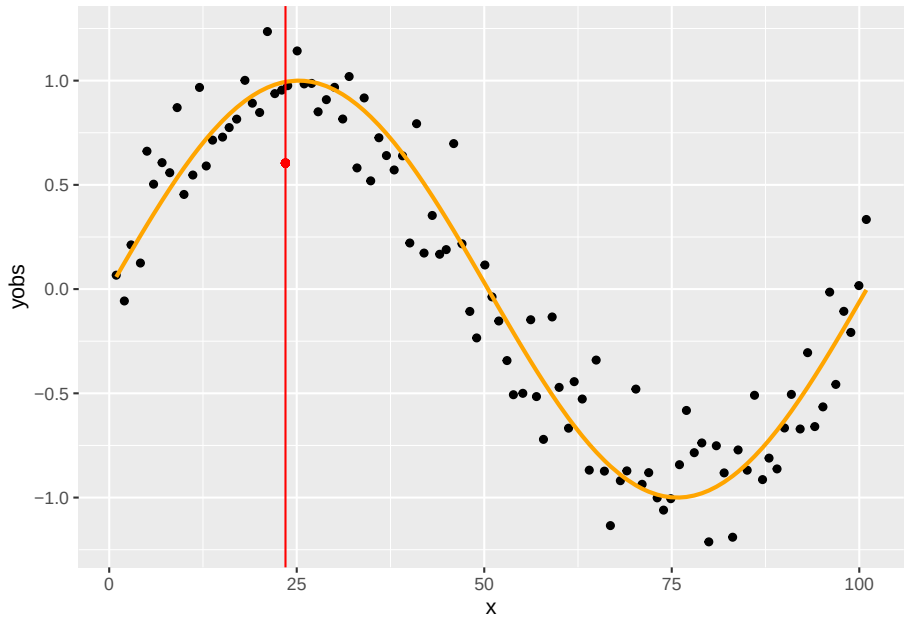
Az illesztett regressziónak azt a pontját vesszük ki, ami a vizsgált érték volt! Az előbbi példát folytatva:

```
p + geom_abline(intercept = coef(fit)[1], slope = coef(fit)[2]) +
  geom_vline(xintercept = 23.5, color = "red") +
  geom_point(x = 23.5, y = predict(fit, data.frame(x = 23.5)), color="red")
```



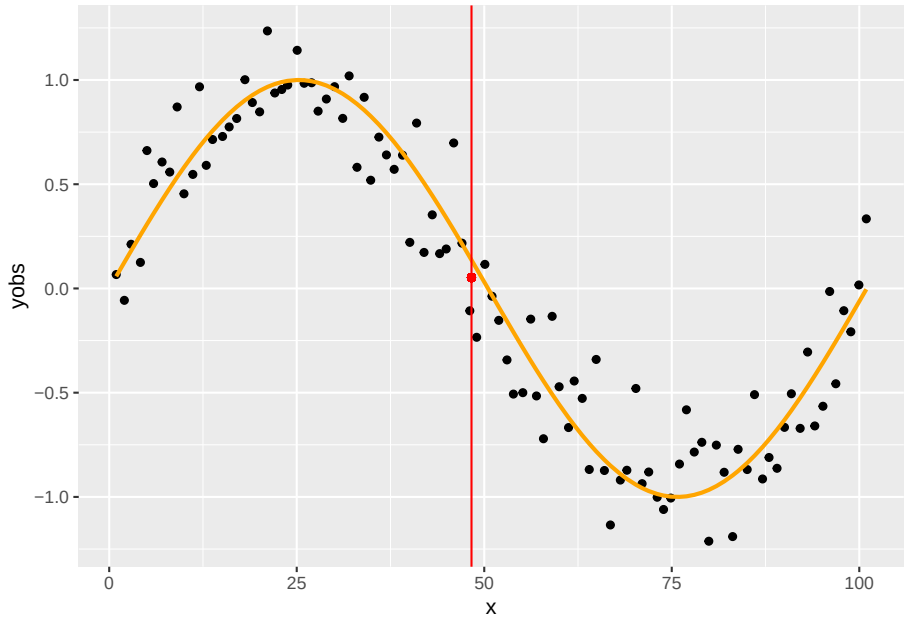
A dolgot automatizálhatjuk is:

```
loessfun <- function(xin, x, yobs, span) {
  n <- length(x)
  w <- tricube(abs(x-xin), sort(abs(x-xin))[ceiling(n*span)])
  fit <- lm(yobs ~ x, weights = w)
  predict(fit, data.frame(x = xin))
}
p + geom_vline(xintercept = 23.5, color = "red") +
  geom_point(x = 23.5, y = loessfun(23.5, SimData$x, SimData$yobs, 0.75), color="red")
```

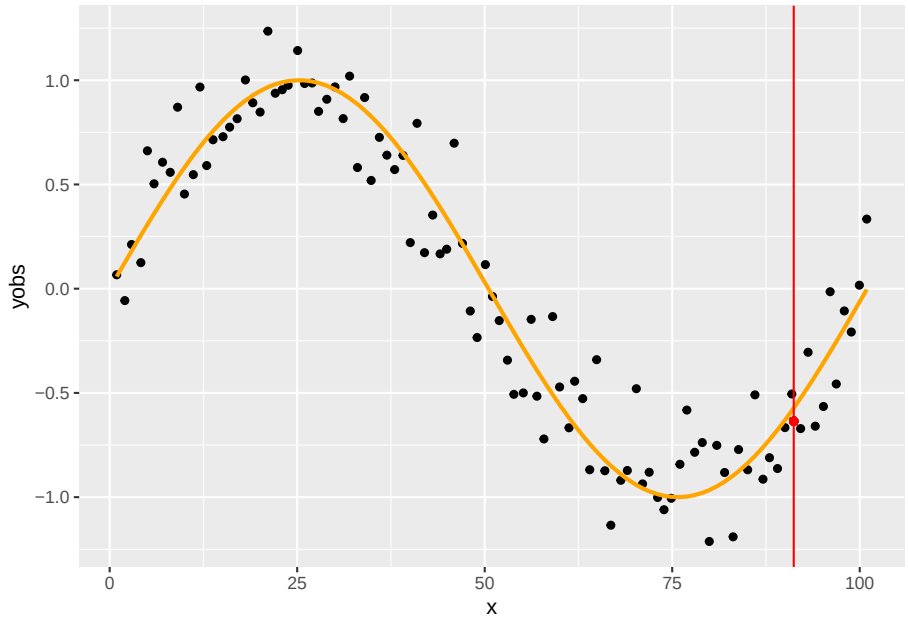


Ezt használva természetesen kényelmesen kiszámíthatjuk ezt bármely más értékre is:

```
p + geom_vline(xintercept = 48.3, color = "red") +
  geom_point(x = 48.3, y = loessfun(48.3, SimData$x, SimData$yobs, 0.75), color="red")
```



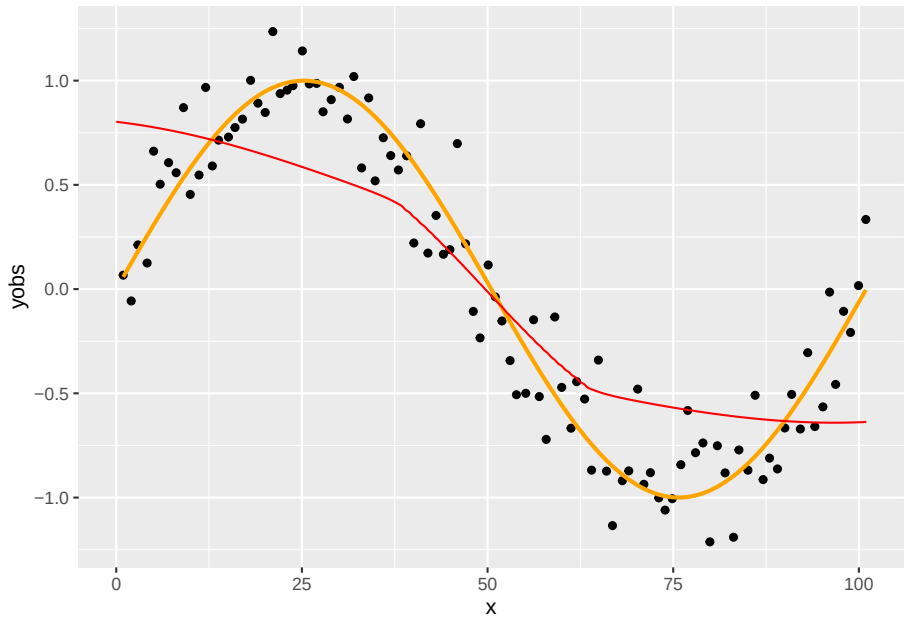
```
p + geom_vline(xintercept = 91.2, color = "red") +
  geom_point(x = 91.2, y = loessfun(91.2, SimData$x, SimData$yobs, 0.75), color="red")
```



1.5. Összerakva az építőelemeket: lokális polinomiális regressziókkal közelítés

Innen már értelemszerű a következő lépés, számítsuk ki ezeket a simított értékeket az x releváns tartományának minden pontjára:

```
SmoothData <- expand.grid(grid = seq(0, 101, 0.1))
SmoothData$value <- apply(SmoothData, 1, function(x)
  loessfun(x["grid"], SimData$x, SimData$yobs, 0.75))
p + geom_line(data = SmoothData, aes(x = grid, y = value), color = "red")
```



Ez lesz a LOESS simítás!

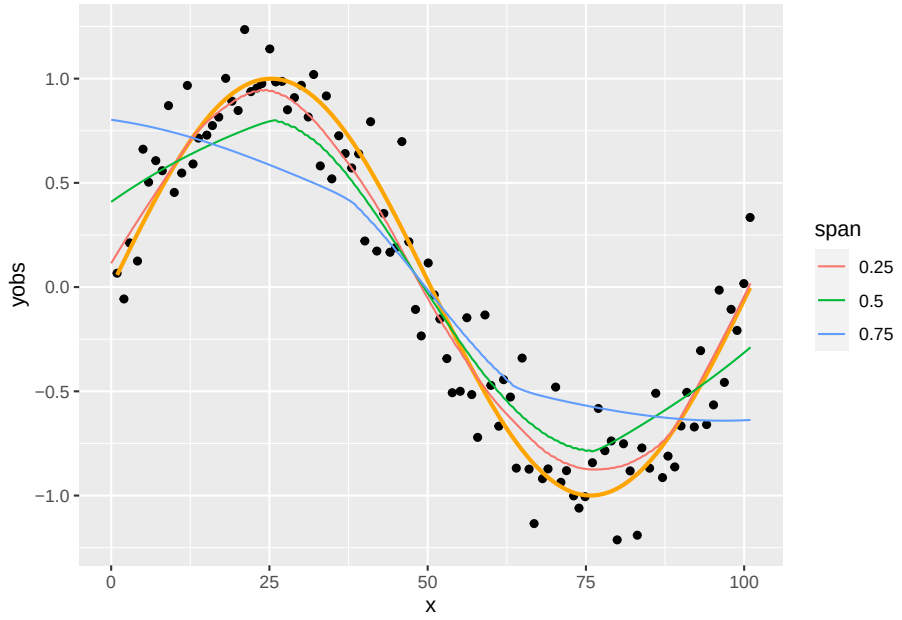
Min múlott az eredmény? Két paramétert használtunk: azt, hogy a pontok mekkora hányadát tartjuk meg, és azt, hogy hányadfokú polinomot illesztettünk. A fenti példában ezek $\alpha = 0{,}75$ és $p = 1$. (Természetesen a súlyozófüggvény a harmadik paraméter, de azt most végig rögzítettnek fogjuk tekinteni.)

1.6. A paraméterek megválasztásának hatása: lokalitás

Kézenfekvő a kérdés, hogy vajon a simításra hogyan hatnak ezek a paraméterek (annál is inkább, mert a fenti simítás nem néz ki túl bíztatóan!). Kezdjük a lokalitást szabályzó α paraméter hatásával:

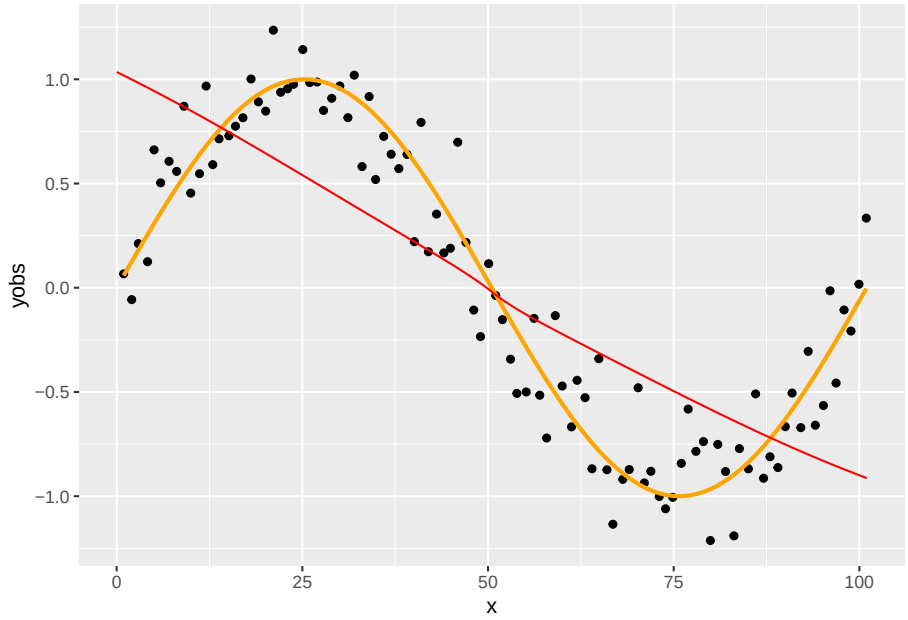
```
SmoothData <- expand.grid(grid = seq(0, 101, 0.1),
                          span = c(2/n+1e-10, 0.25, 0.5, 0.75, 1))
SmoothData$value <- apply(SmoothData, 1, function(x)
  loessfun(x["grid"], SimData$x, SimData$yobs, x["span"]))
SmoothData$span <- as.factor(SmoothData$span)

p + geom_line(data = SmoothData[SmoothData$span%in%c(0.25, 0.5, 0.75), ],
              aes(x = grid, y = value, color = span))
```



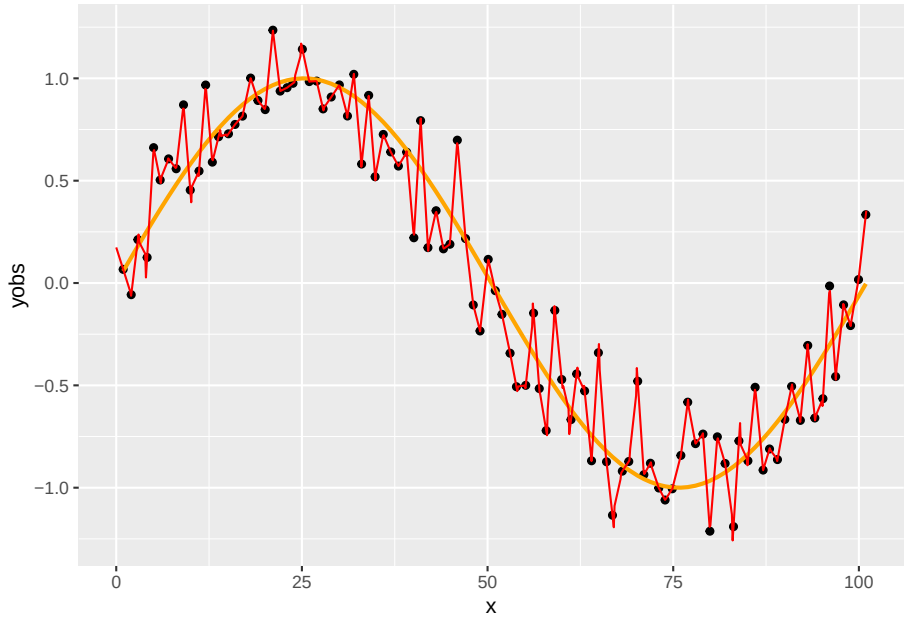
Még szemléletesebb, ha megnézzük a két szélső értéket is. Ha minden pontot figyelembe veszünk (nincs lokalitás):

```
p + geom_line(data = SmoothData[SmoothData$span==1, ], aes(x = grid, y = value),
              color = "red")
```



Ha semennyi pontot nem veszünk figyelembe, a legközelebbi kettő kivételével értelemszerűen, hogy legyen mire illeszteni a görbét (teljes lokalitás):

```
p + geom_line(data = SmoothData[SmoothData$span==2/n+1e-10, ], aes(x = grid, y = value),
              color = "red")
```



Az első esetet szokták úgy hívni, hogy túlsimítás, a másodikat úgy, hogy alulsimítás. Vajon hogyan tudjuk a simítási paraméter (ennél a módszernél az α) értékét optimálisan megválasztani?

1.7. A paraméterek megválasztásának hatása: a polinom fokszáma

Mielőtt az előbbi kérdésre válaszolunk, meg kell nézni még egy kérdést, mert vissza fog hatni a válaszra: az illesztett polinom p fokszámát. Vajon mi történik, ha lineáris regresszió helyett magasabb fokszámú polinomot használunk?

1.7.1. Kitérő: polinomiális regresszió illesztésének szintaktikája R alatt

Érdemes kitérni arra a kérdésre, hogy a polinomiális regressziót hogyan kell R alatt specifikálni (az `lm`-nek megadni).

A dolognak van ugyanis egy szintaktikai trükkje. Az ugyanis, ami a legkézenfekvőbbnek tűnne, nem működik:

```
lm(y~x+x^2, data = SimData)
```

```
##
## Call:
## lm(formula = y ~ x + x^2, data = SimData)
##
## Coefficients:
## (Intercept)            x
##     0.96485     -0.01892
```

A probléma oka, hogy az `lm` formula interfészében a műveleti jelek speciálisan viselkednek. A `^` *nem* a hatványozás jele, hanem interakciót specifikál, azaz az `(x+y)^2` ugyanaz mint az `x+y+x:y`, viszont egy tagnál nincs mivel interakciót képezni, így az `x^2` ugyanaz lesz mint az `x`.

A megoldást az `I()` függvény jelenti, ami azt mondja az R-nek, hogy a beleírt kifejezésben szereplő operátorokat a szokásos aritmetikai értelemmel értékelje ki:

```
lm(y~x+I(x^2), data = SimData)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2), data = SimData)
##
## Coefficients:
## (Intercept)            x       I(x^2)
##   1.014e+00   -2.178e-02    2.806e-05
```

Ez már működik, de eljárhatunk egyszerűbben is, a `poly` függvény ugyanis pont erre szolgál:

```
lm(y~poly(x,2), data = SimData)
```

```
##
## Call:
## lm(formula = y ~ poly(x, 2), data = SimData)
##
## Coefficients:
## (Intercept)  poly(x, 2)1  poly(x, 2)2
##  -0.0001279   -5.5423654    0.2142741
```

Látszólag mást kaptunk, de valójában csak a parametrizálásban van eltérés, a predikciók azonosak:

```
predict(lm(y~x+I(x^2)), data.frame(x = 43.9))
```

```
##         1
## 0.1119441
```

```
predict(lm(y~poly(x,2)), data.frame(x = 43.9))
```

```
##         1
## 0.1119441
```

A magyarázat, hogy a `poly` alapjáraton ortogonalizálja a tagokat (azaz olyan másodfokú polinomot szolgáltat, melynek elemei korrelálatlanok egymással). Nézzük is meg, a kapott vektorok csakugyan ortogonálisak, sőt, sortonormáltak, egymásra és a csupa 1 vektorra nézve:

```
t(cbind(1, poly(x, 3)))%*%cbind(1, poly(x, 3))
```

```
##                              1             2             3
##    1.010000e+02  2.498002e-16  2.720046e-15 -2.775558e-17
## 1  2.498002e-16  1.000000e+00 -2.706169e-16 -5.551115e-17
## 2  2.720046e-15 -2.706169e-16  1.000000e+00 -1.457168e-16
## 3 -2.775558e-17 -5.551115e-17 -1.457168e-16  1.000000e+00
```

Ha szeretnénk, ezt kikapcsolhatjuk, és akkor visszakapjuk a kézel létrehozott eredményt:

```
lm(y~poly(x,2, raw = TRUE), data = SimData)
```

```
##
## Call:
## lm(formula = y ~ poly(x, 2, raw = TRUE), data = SimData)
##
## Coefficients:
##             (Intercept)  poly(x, 2, raw = TRUE)1  poly(x, 2, raw = TRUE)2
##               1.014e+00               -2.178e-02                 2.806e-
05
```

Az alapértelmezett persze nem véletlenül az, ami: az ortogonális polinomok becslése sokkal jobb numerikus szempontból. Például a modellmátrix kondíciószámát nézve:

```
kappa(cbind(1, poly(x, 3)), exact = TRUE)
```

```
## [1] 10.04988
```

```
kappa(cbind(1, poly(x, 3, raw = TRUE)), exact = TRUE)
```

```
## [1] 1651776
```

A `poly` használata nem csak elegánsabb és numerikusan szerencsésebb, de jóval kényelmesebb is (gondoljunk bele mi volna, ha véletlenül tizedfokú polinomot akarnánk specifikálni, vagy változó lenne, hogy hányadfokú polinomról van szó).

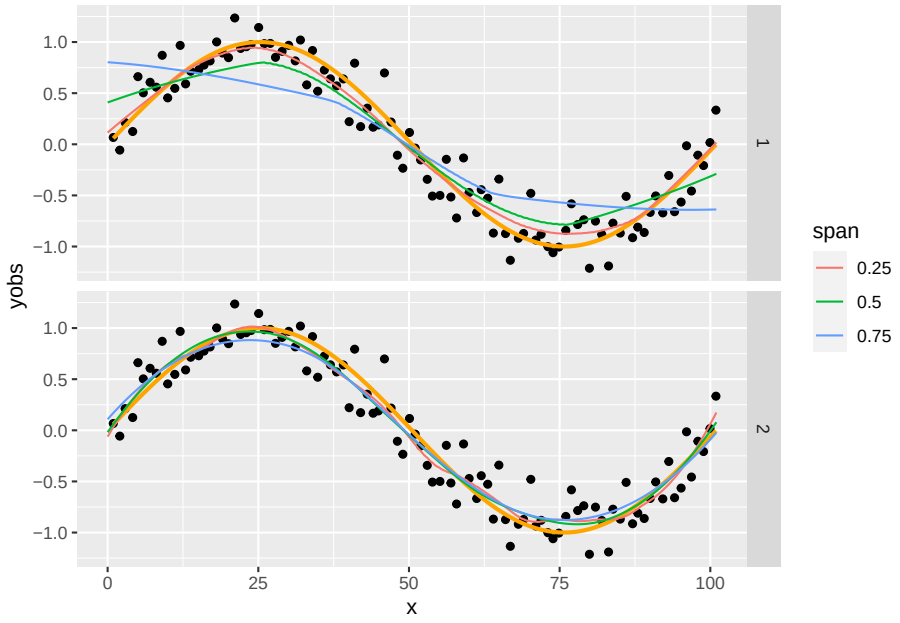
1.7.2. Polinom fokszámának változtatása

Most már könnyedén megoldhatjuk, hogy a fokszám is változtatható legyen:

```
loessfun <- function(xin, x, yobs, span, degree) {
  n <- length(x)
  w <- tricube(abs(x-xin), sort(abs(x-xin))[ceiling(n*span)])
  fit <- lm(yobs ~ poly(x, degree), weights = w)
  predict(fit, data.frame(x = xin))
}
```

Ezt használva immár különböző fokszámokkal és simítási paraméterrel is próbálkozhatunk:

```
SmoothData <- rbind(expand.grid(grid = seq(0, 101, 0.1),
                                span = c(2/n+1e-10, 0.25, 0.5, 0.75, 1), degree = 1),
                    expand.grid(grid = seq(0, 101, 0.1),
                                span = c(3/n+1e-10, 0.25, 0.5, 0.75, 1), degree = 2))
SmoothData$value <- apply(SmoothData, 1, function(x)
  loessfun(x["grid"], SimData$x, SimData$yobs, x["span"], x["degree"]))
SmoothData$span <- as.factor(SmoothData$span)
p + geom_line(data = SmoothData[SmoothData$span%in%c(0.25, 0.5, 0.75), ],
              aes(x = grid, y = value, color = span)) + facet_grid(rows = vars(degree))
```



Egy nagyon fontos dolgot látunk: ha áttérünk a másodfokú polinom használatára, akkor gyakorlatilag a simítási paramétertől függetlenül szinte tökéletes simítást kapunk!

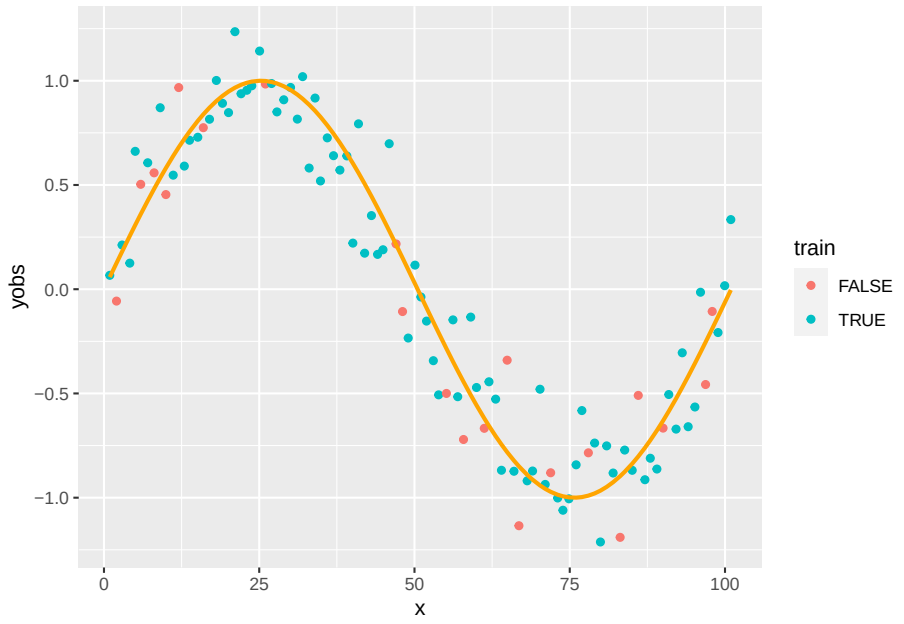
1.8. A paraméterek megválasztása

Adja magát a kérdés, hogy a paramétereket hogyan választhatjuk meg egy valódi helyzetben (értsd: ahol mi sem tudjuk mi az igazi függvény).

Itt most csak az érzékeltetés kedvéért mutatunk meg egy nagyon egyszerű módszert (megjegyezve, hogy ennél okosabban is el lehet járni, de ez is szemléltetni fogja, hogy a probléma kezelhető).

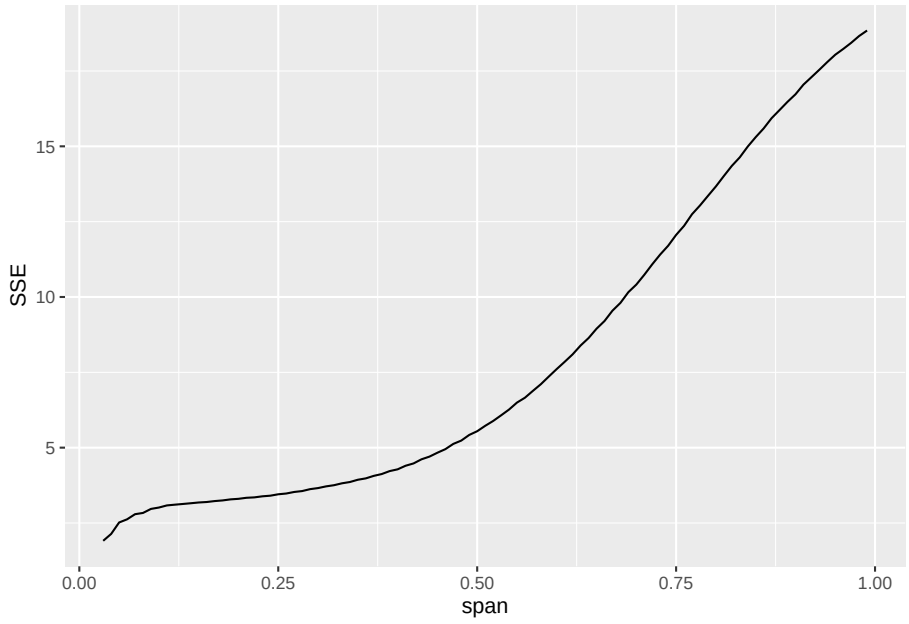
Amit meg fogunk nézni az lényegében egy hold-out set validáció. A simítás jóságát azzal fogjuk mérni, hogy a simítógörbe és a pontok között mekkora a négyzetes eltérésösszeg. Ennek minimalizálása természetesen mindig alulsimított megoldást eredményezne, hiszen ez a célfüggvény nullába is vihető. Éppen ezért cselesebben járunk el: a pontokat véletlenszerűen két részre osztjuk, az egyik alapján határozzuk meg a simítógörbét (tanítóhalmaz), de a hibát a másik halmazon (teszthalmaz) mérjük le! Így ha elkezdünk túlsimítani, akkor a tanítóhalmazon ugyan csökken a hiba, de a teszthalmazon elkezd nőni. Azt a simítást választjuk tehát, ami a teszthalmazon mért hibát minimalizálja.

```
SimData$train <- FALSE
SimData$train[sample(1:101, 80)] <- TRUE
ggplot(SimData, aes(x = x, y = yobs, color = train)) + geom_point() +
  geom_line(aes(y = y), color = "orange", lwd = 1)
```



Nézzük meg hogyan alakul a hiba a simítási paraméter változtatásával, ha az összes pontra illesztünk (az egyszerűség kedvéért a fokszám legyen fixen 1):

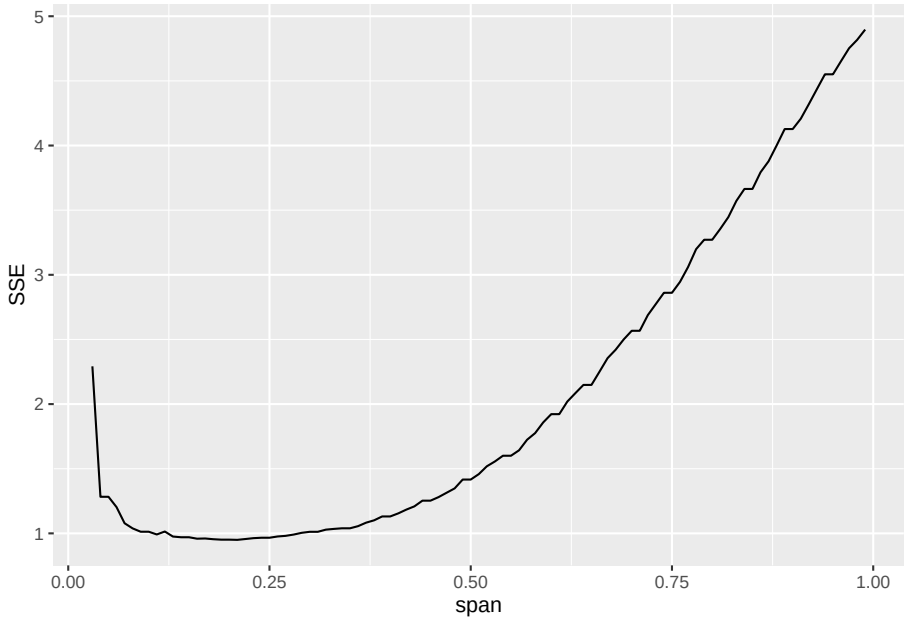
```
spans <- seq(0.03, 0.99, 0.01)
SSEfull <- sapply(spans, function(sp) sum((sapply(1:101, function(i)
  loessfun(SimData$x[i], SimData$x, SimData$yobs, sp, 1))-yobs)^2))
ggplot(data.frame(span = spans, SSE = SSEfull), aes(x = span, y = SSE)) + geom_line()
```



Ahogy vártuk, a hiba folyamatosan csökken, az alulsimított megoldás tűnik a legjobbnak. (Jól látható, hogy az alulsimítás a túlilleszkedés analóg fogalma).

Most vessük be a trükköt: csak a tanítóhalmazra illesztünk, miközben a teszthalmazon mérjük a hibát. Íme az eredmény:

```
SSEfull <- sapply(spans, function(sp) sum((sapply(which(!SimData$train), function(i)
  loessfun(SimData$x[i], SimData$x[SimData$train==TRUE],
           SimData$yobs[SimData$train==TRUE], sp, 1))-yobs[SimData$train==FALSE])^2))
ggplot(data.frame(span = spans, SSE = SSEfull), aes(x = span, y = SSE)) + geom_line()
```



Pontosan a várakozásainknak megfelelően így már szép, értelmes optimum van: mind a túl-, mind az alulsimítást észre tudjuk venni ezzel a validációval.

Az optimális simítási paraméter értéke számszerűen is meghatározható:

```
spans[which.min(SSEfull)]
```

```
## [1] 0.21
```

2. fejezet

Spline fogalma, lineáris regressziótól a spline-regresszióig

2.1. A regresszió

A regresszió legtöbb alkalmazott statisztikai terület talán legfontosabb eszköze

Regresszió: változók közti kapcsolat (illetve annak becslése minta alapján)

„Kapcsolat" formalizálása: függvény a matematikai fogalmával, tehát keressük az

$$Y = f\left(X_1, X_2, \dots, X_p\right) + \varepsilon = f(\mathbf{X})$$

függvényt

(Y eredményváltozó, X_i-k a magyarázó változók)

A regresszió legtöbb alkalmazott statisztikai terület talán legfontosabb eszköze

Regresszió: változók közti kapcsolat (illetve annak becslése minta alapján)

„Kapcsolat" formalizálása: függvény a matematikai fogalmával, tehát keressük az

$$Y = f\left(X_1, X_2, \dots, X_p\right) + \varepsilon = f(\mathbf{X})$$

függvényt

(Y eredményváltozó, X_i-k a magyarázó változók)

2.2. Regresszió becslése mintából

Paraméteres regresszió: ha *a priori* feltételezzük, hogy az f függvény valamilyen – paraméterek erejéig meghatározott – függvényformájú (az „alakja" ismert), és így a feladat e paraméterek becslésére redukálódik

Tipikus példa a **lineáris regresszió**: $f(\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = \mathbf{X}^T\boldsymbol{\beta}$, így $Y = \mathbf{X}^T\boldsymbol{\beta} + \varepsilon$

Ha rendelkezésre állnak az $\left\{y_i, \mathbf{x}_i\right\}_{i=1}^n$ megfigyeléseink a háttéreloszlásra, akkor e mintából megbecsülhetjük a paramétereket például **hagyományos legkisebb négyzetek**

Paraméteres regresszió: ha *a priori* feltételezzük, hogy az f függvény valamilyen – paraméterek erejéig meghatározott – függvényformájú (az „alakja" ismert), és így a feladat e paraméterek becslésére redukálódik

Tipikus példa a **lineáris regresszió**: $f(\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = \mathbf{X}^T\boldsymbol{\beta}$, így $Y = \mathbf{X}^T\boldsymbol{\beta} + \varepsilon$

Ha rendelkezésre állnak az $\left\{y_i, \mathbf{x}_i\right\}_{i=1}^n$ megfigyeléseink a háttéreloszlásra, akkor e mintából megbecsülhetjük a paramétereket például **hagyományos legkisebb négyzetek** (OLS) módszerével:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\mathbf{b}} \sum_{i=1}^{n} \left[Y_i - \mathbf{X}_i^T\mathbf{b}\right]^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{b}\|^2$$

Itt tehát $\mathbf{X}$ az a mátrix, amiben a

(OLS) módszerével:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\mathbf{b}} \sum_{i=1}^{n} \left[Y_i - \mathbf{X}_i^T \mathbf{b} \right]^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{b}\|^2$$

Itt tehát $\mathbf{X}$ az a mátrix, amiben a magyarázó változók elé egy csupa 1 oszlopot szúrtunk, a neve **modellmátrix** vagy design mátrix

2.3. Paraméteres és nem-paraméteres regresszió

De cserében mindig ott lebeg felettünk a kérdés, hogy a függvényformára *jó feltételezést* tettünk-e (hiszen ez nem az adatokból következik, ezt „ráerőszakoljuk" az adatokra)

(Persze ezért van a modelldiagnosztika)

A nem-paraméteres regresszió *flexibilis*, olyan értelemben, hogy minden a priori megkötés nélkül követi azt, ami az adatokból következik (a valóság ritkán lineáris?)

Cserében nehezebb becsülni, és nem kapunk analitikus – jó esetben valamire hasznosítható – regressziós függvényt, nem lehet értelmesen interpolálni és extrapolálni („fordul a kocka" a paraméteres esethez képest)

2.4. A lineáris regresszió kibővítése, nemlinearitások

Maradva a paraméteres keretben, arra azért mód van, hogy a függvényformát kibővítsük (és így flexibilisebbé tegyük)

Ezzel a különféle **nemlineáris regressziókhoz** jutunk el

E nemlinearitásoknak két alaptípusa van

- Változójában nemlineáris modell (pl. $\beta_0 + \beta_1 x + \beta_2 x^2$): csak a szó „matematikai értelmében" nemlineáris, ugyanúgy becsülhető OLS-sel
- Paraméterében nemlineáris modell (pl. $\beta_0 x_1^{\beta_1} x_2^{\beta_2}$): felrúgja a lineáris struktúrát, így érdemileg más, csak linearizálás után, vagy NLS-sel becsülhető

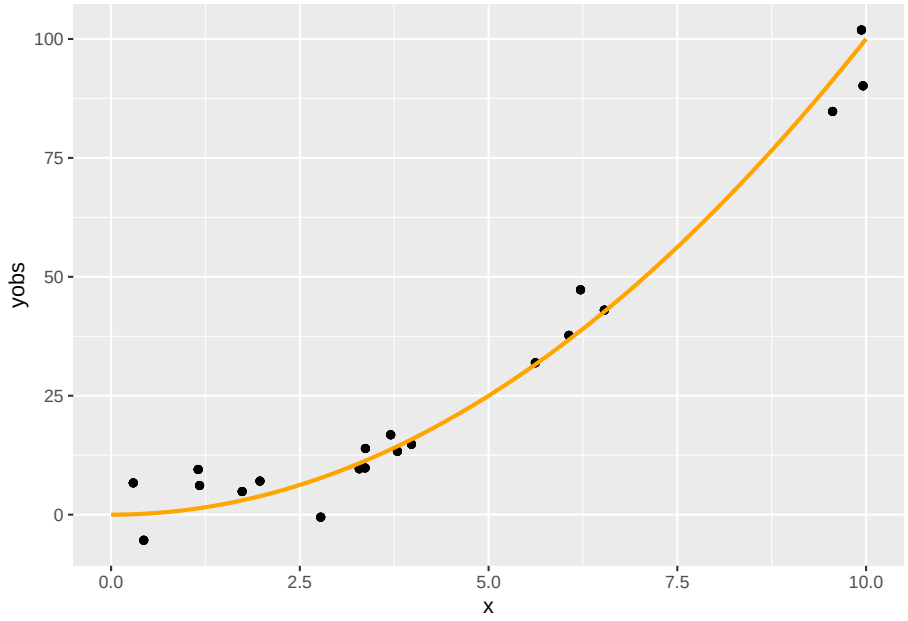
Mi most az első esettel fogunk foglalkozni

Az itt látott „polinomiális regresszió" valóban nagyon gyakori módszer a flexibilitás növelésére

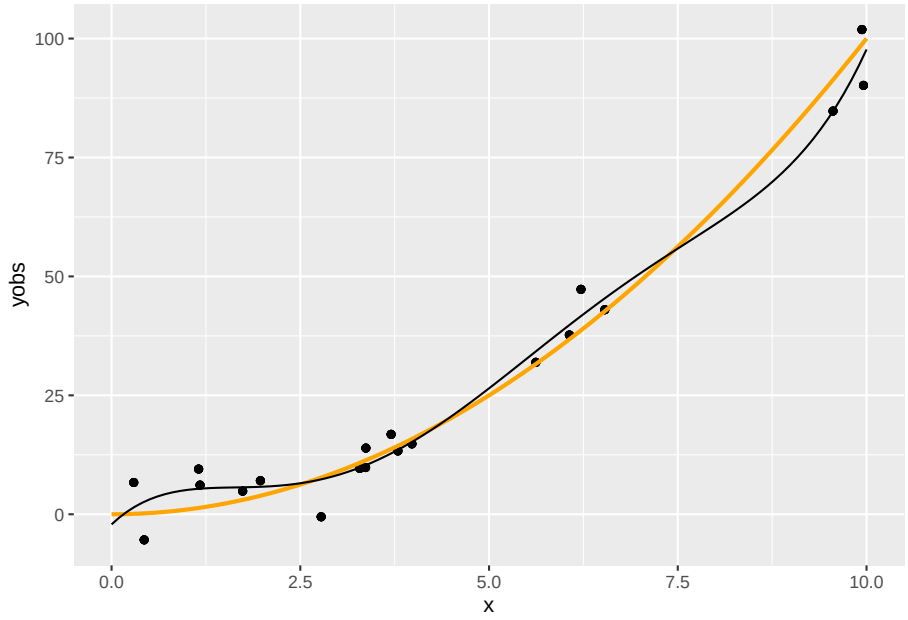
2.5. Egy példa

Tekintsünk most egy másik példát, egy zajos másodfokú függvényt, kevesebb pontból:

```
n <- 20
x <- runif(n, 0, 10)
xgrid <- seq(0, 10, length.out = 100)
ygrid <- xgrid^2
yobs <- x^2 + rnorm(n, 0, 5)
SimData <- data.frame(x, xgrid, ygrid, yobs)
p <- ggplot(SimData) + geom_point(aes(x = x, y = yobs)) +
  geom_line(aes(x = xgrid, y = ygrid), color = "orange", lwd = 1)
p
```




```
fit5 <- lm(yobs ~ poly(x, 5), data = SimData)
p + geom_line(data = data.frame(xgrid, pred = predict(fit5, data.frame(x = xgrid))),
              aes(x = xgrid, y = pred))
```



2.7. Módosítás

Mondjuk, hogy nagyobb flexibilitásra vágyunk

Mondjuk, hogy nagyobb flexibilitásra vágyunk

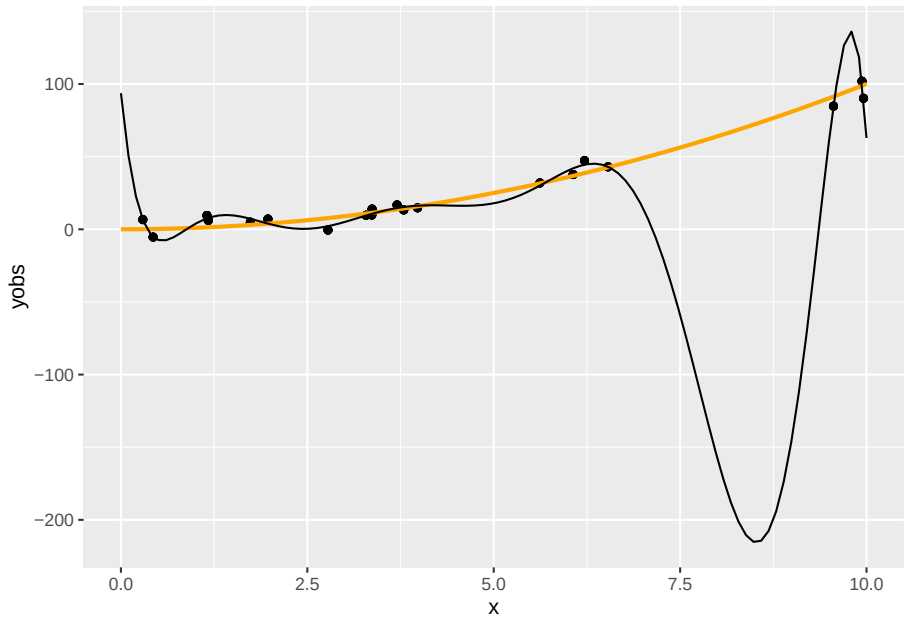
- Például figyelembe akarjuk venni, hogy ez nem tűnik teljesen lineárisnak, vagy meg akarjuk ragadni a finomabb tendenciákat is

Emeljük a polinom fokszámát (ez nyilván növeli a flexibilitást, hiszen a kisebb fokszám nyilván speciális eset lesz), például 10-re

Szokás azt mondani, hogy a rang 5 illetve 10 (a polinom fokszáma, a becsülendő paraméterek száma nyilván egyezik a modellmátrix rangjával, de ez a fogalom később, amikor nem is polinomunk van, akkor is használható)

2.8. Regresszió tizedfokú polinommal

```
fit10 <- lm(yobs ~ poly(x, 10), data = SimData)
p + geom_line(data = data.frame(xgrid, pred = predict(fit10, data.frame(x = xgrid))),
              aes(x = xgrid, y = pred))
```



2.9. Mi a jelenség oka?

Szokás azt mondani, hogy *túlilleszkedés*, ami persze igaz is, de itt többről van szó

A polinomok elsősorban *lokálisan* tudnak jól közelíteni (a Taylor-sorfejtéses érvelés miatt), de nekünk arra lenne szükségünk, hogy *globálisan* jól viselkedő függvényformát találjunk

Pedig a polinomokat amúgy szeretjük, többek között azért is, mert szép sima görbét írnak le (matematikai értelemben véve a simaságot: végtelenszer folytonosan deriválhatóak, C^∞-beliek)

Mi lehet akkor a megoldás?

2.10. Mi lehet a megoldás?

Egy lehetséges megközelítés: „összerakjuk a globálisat több lokálisból”

Azaz szakaszokra bontjuk a teljes intervallumot, és mindegyiket *külön-külön* polinommal igyekszünk modellezni

Így próbáljuk kombinálni a két módszer előnyeit

Persze a szakaszosan definiált polinomok önmagában még nem jók: a szakaszhatárokon találkozniuk kell (e találkozópontok neve: **knot**, „csomópont”, a számukat $q-2$-val jelöljük, a pozíciójukat x_i^*-vel)

Szokás azt mondani, hogy *túlilleszkedés*, ami persze igaz is, de itt többről van szó

A polinomok elsősorban *lokálisan* tudnak jól közelíteni (a Taylor-sorfejtéses érvelés miatt), de nekünk arra lenne szükségünk, hogy *globálisan* jól viselkedő függvényformát találjunk

Pedig a polinomokat amúgy szeretjük, többek között azért is, mert szép sima görbét írnak le (matematikai értelemben véve a simaságot: végtelenszer folytonosan deriválhatóak, C^∞-beliek)

Mi lehet akkor a megoldás?

Egy lehetséges megközelítés: „összerakjuk a globálisat több lokálisból”

Azaz szakaszokra bontjuk a teljes intervallumot, és mindegyiket *külön-külön* polinommal igyekszünk modellezni

Így próbáljuk kombinálni a két módszer előnyeit

Persze a szakaszosan definiált polinomok önmagában még nem jók: a szakaszhatárokon találkozniuk kell (e találkozópontok neve: **knot**, „csomópont”, a számukat $q-2$-val jelöljük, a pozíciójukat x_i^*-vel)

Sőt, ha a simasági tulajdonságokat is át akarjuk vinni, akkor az érintkezési pontokban a deriváltaknak (magasabbrendűeknek is) is egyezniük kell

Sőt, ha a simasági tulajdonságokat is át akarjuk vinni, akkor az érintkezési pontokban a deriváltaknak (magasabbrendűeknek is) is egyezniük kell

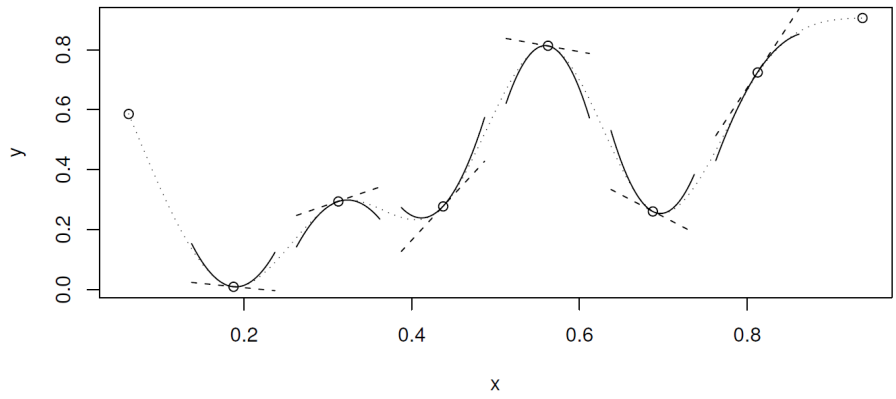
Ha p-edfokú polinomokat használunk, akkor az első $p-1$ derivált – és persze a függvényérték – egyezését kell kikötnünk a knot-okban (és esetleg még valamit a végpontokra)

Ez így már jó konstrukció lesz, a neve: **spline**

2.11. Természetes köbös spline

(Azért köbös, mert harmadfokúak a polinomok, és azért természetes, mert azt kötöttük ki, hogy a végpontokban nulla legyen a második derivált)

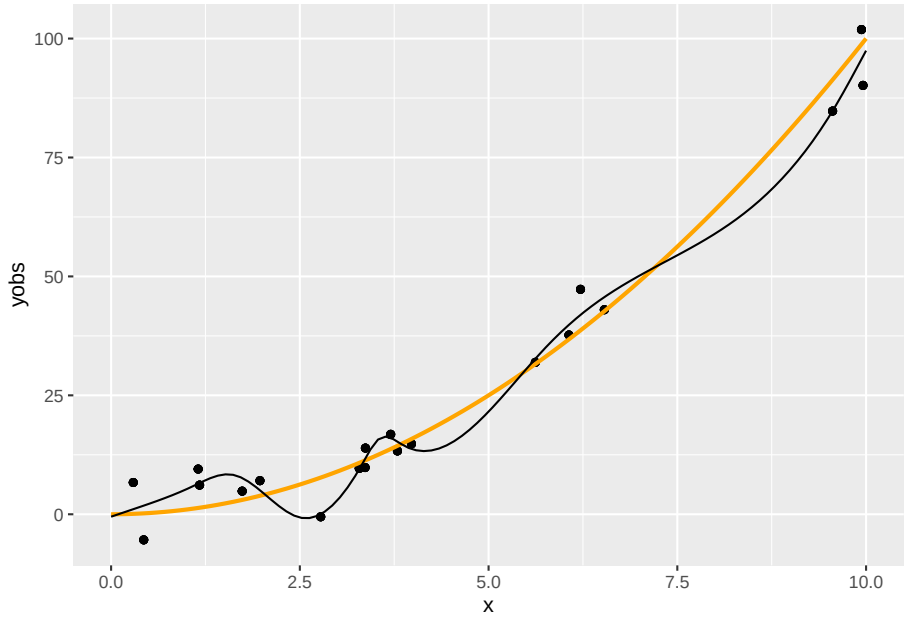
(Azért köbös, mert harmadfokúak a polinomok, és azért természetes, mert azt kötöttük ki, hogy a végpontokban nulla legyen a második derivált)



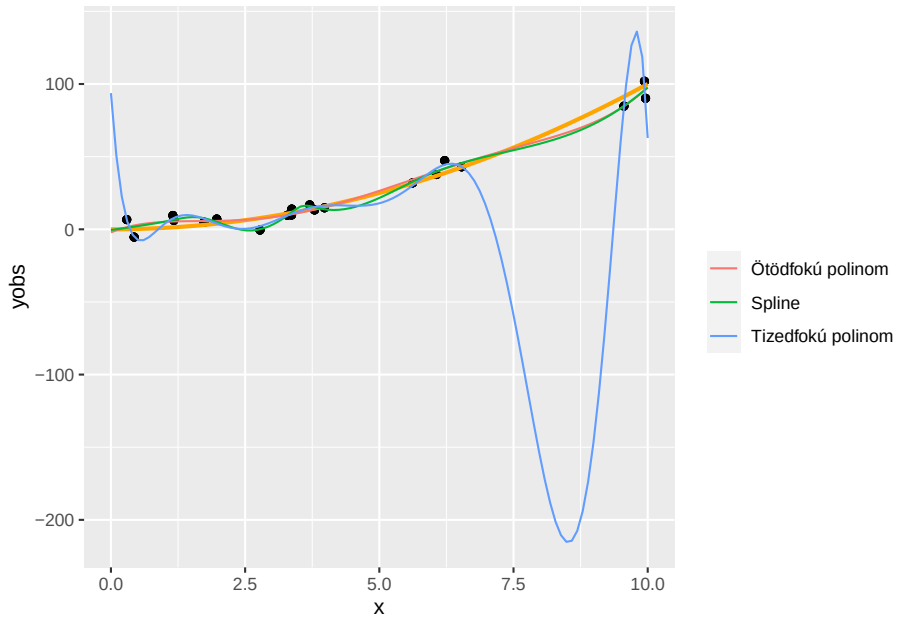
2.1. ábra. Természetes köbös spline

2.12. A példa regressziója természetes köbös spline-nal

```
fitSpline <- lm(yobs ~ splines::ns(x, 10), data = SimData)
p + geom_line(data = data.frame(xgrid, pred = predict(fitSpline, data.frame(x = xgrid)),
              aes(x = xgrid, y = pred))
```




```
p + geom_line(data = rbind(data.frame(type = "Ötödfokú polinom",
                                      pred = predict(fit5, data.frame(x = xgrid)), xgrid),
                           data.frame(type = "Tizedfokú polinom",
                                      pred = predict(fit10, data.frame(x = xgrid)), xgrid),
                           data.frame(type = "Spline",
                                      pred = predict(fitSpline, data.frame(x = xgrid)),
                                      xgrid)),
              aes(x = xgrid, y = pred, color = type)) + labs(color = "")
```



2.14. A spline-regresszió ereje

Nem csak az a jó, hogy szépen illeszkedik (tulajdonképpen még annál is jobban, mint a tizedfokú polinom, még ott is, ahol az jól illeszkedik amúgy)

…hanem, hogy – most már elárulhatom – *ez is ugyanúgy 10 rangú* mint a tizedfokú polinom!

Mégis: nyoma nincs túlilleszkedésnek

Nem csak az a jó, hogy szépen illeszkedik (tulajdonképpen még annál is jobban, mint a tizedfokú polinom, még ott is, ahol az jól illeszkedik amúgy)

…hanem, hogy – most már elárulhatom – *ez is ugyanúgy 10 rangú* mint a tizedfokú polinom!

Mégis: nyoma nincs túlilleszkedésnek

3. fejezet

Spline-regresszió becslése bázisfüggvényekkel, penalizáltan

3.1. Bázisfüggvényekkel felírás

3.1.1. Hogyan becsüljük meg a spline-regressziót?

Amiről nem beszéltünk eddig: ez mind szép, de hogyan tudunk ténylegesen is megbecsülni egy ilyen spline-regressziót?

Ehhez visszalépünk pár lépést, és bevezetünk egy első kicsit absztraktnak tűnő, de később rendkívül jó szolgálatot tevő megközelítést

Bár a célunk a spline-regresszió becslésének a megoldása, de a dolog – értelemszerűen – alkalmazható polinomiális regresszióra is (legfeljebb nincs sok értelme, mert az hagyományos módszerekkel is jól kézbentartható), úgyhogy először azon fogjuk illusztrálni

3.1.2. Polinomok tere mint függvénytér

A másodfokú polinomok – mint függvények – összessége **függvényteret** alkot

Ez egy olyan *vektortér*, aminek az elemei a függvények, a skalárok a valós számok, a két művelet pedig

- Skalárral szorzás: $\left(cf\right)(x) = cf(x)$
- Vektorok (azaz függvények) összeadása: $\left(f+g\right)(x) = f(x) + g(x)$, tehát pontonkénti összeadás

Amiről nem beszéltünk eddig: ez mind szép, de hogyan tudunk ténylegesen is megbecsülni egy ilyen spline-regressziót?

Ehhez visszalépünk pár lépést, és bevezetünk egy első kicsit absztraktnak tűnő, de később rendkívül jó szolgálatot tevő megközelítést

Bár a célunk a spline-regresszió becslésének a megoldása, de a dolog – értelemszerűen – alkalmazható polinomiális regresszióra is (legfeljebb nincs sok értelme, mert az hagyományos módszerekkel is jól kézbentartható), úgyhogy először azon fogjuk illusztrálni

A másodfokú polinomok – mint függvények – összessége **függvényteret** alkot

Ez egy olyan *vektortér*, aminek az elemei a függvények, a skalárok a valós számok, a két művelet pedig

- Skalárral szorzás: $\left(cf\right)(x) = cf(x)$
- Vektorok (azaz függvények) összeadása: $\left(f+g\right)(x) = f(x) + g(x)$, tehát pontonkénti összeadás

Belátható, hogy ez teljesíti a vektortéraxiómákat, mert zárt a két műveletre (másodfokú polinomok összege másodfokú polinom és másodfokú polinom konstansszorosa másodfokú polinom), és a többi követelményt is teljesíti

Belátható, hogy ez teljesíti a vektortéraxiómákat, mert zárt a két műveletre (másodfokú polinomok összege másodfokú polinom és másodfokú polinom konstansszorosa másodfokú polinom), illetve az összeadásra nézve kommutatív csoport, a szorzás és az összeadás mindkét irányból disztributív, van egységelem szorzásra nézve és a skalárszorzás valamint a valós számok szorzása kompatibilis

3.1.3. Polinomok terének bázisa

Szuper, de mindez mire jó?

Ha vektortér, akkor létezik **bázisa**, azaz olyan vektorok halmaza, melyekből lineáris kombinációval minden vektor – egyértelműen – előállítható (bázis: lineárisan független generátorrendszer)

A bázis nem feltétlenül egyértelmű, de az elemszáma igen, ez a vektortér **dimenziója**

Például a másodfokú polinomok jó bázisa $\left\{1, x, x^2\right\}$, nyilvánvaló, hogy ebből tényleg minden $ax^2 + bx + c$ másodfokú polinom előállítható lineáris kombinációval (triviálisan, a súlyok c, b és a)

Függvényterek esetében a bázis elemeit **bázisfüggvényeknek** is szokás nevezni, az $\left\{1, x, x^2\right\}$ tehát a másodfokú polinomok bázisfüggvényei

3.1.4. A polinomok terének dimenziója

Mivel mutattunk egy konkrét bázist, így a dimenzió nyilván 3, de a későbbiek szempontjából jól jön egy másik módszer is

Azzal, hogy az $ax^2 + bx + c$ polinomot megfeleltettük az (a, b, c) valós számhármasnak, a polinomok tere és a valós számhármasok tere (az $\mathbb{R}^3$) között létesítettünk egy izomorfizmust (a leképezés művelettartó és kölcsönösen egyértelmű)

Emiatt a polinomok terének ugyanaz a dimenziója, mint az $\mathbb{R}^3$-nak, ami viszont természetesen 3

Ez a módszer általában is használható: a dimenzió a felíráshoz szükséges paraméterek száma (feltéve, hogy ezek valós számok, valamint mindegyikhez tartozik egy polinom és viszont)

3.1.5. Spline-ok függvénytere

Mindez a spline-okra is igaz!

Érthető: minden pontban két polinomot adunk össze, vagy polinomot szorzunk skalárral, az eredmény polinom (már láttuk) – így tud spline adott pontja lenni!

Azaz: spline-okat is elő tudunk állítani bázisfüggvények lineáris kombinációjaként!

3.1.6. Hány dimenziós a spline-ok tere?

Mielőtt megkeressük a spline-ok terének egy bázisát (azaz a konkrét bázisfüggvényeket), tisztázni kellene, hogy hány bázisfüggvényt keresünk egyáltalán, azaz hány dimenziós a spline-ok függvénytere

Naiv ötlet (köbös spline-okat használva példaként): van $q-1$ szakasz ($q-2$ knot, ami meghatároz $q-3$ szakaszt meg a két vége; úgy is felfogható, hogy a két végével együtt q knot van, ami meghatároz $q-1$ szakaszt) és mindegyiken egy harmadfokú polinom (aminek 4 paramétere van), akkor az $4q-4$ paraméter

Igen ám, de vannak megkötések: a knotokban a függvényérték és az első két derivált egyezik

Minden megkötés minden pontban 1 egyenlet, az 1-gyel csökkenti a paraméterek számát: van $q-2$ knot és 3 megkötés, az $3q-6$ csökkentés, marad $q+2$ paraméter

De mivel természetes, így a végpontokban is van 1-1 megkötés: marad q paraméter, azaz q dimenziós a természetes köbös spline-ok tere (ezért neveztük a knot-ok számát $q-2$-nek!)

3.1.7. Mik a spline-ok bázisfüggvényei?

Természetesen itt is igaz, hogy adott, rögzített spline-osztályra (pl. természetes köbös) is végtelen sok bázis van

Köztük célszerűség alapján választhatunk

A részletek nélkül két példa:

- $b_1(x) = 1, b_2(x) = x, b_i(x) = \left|x - x^*_{i-2}\right|^3 \ (i = 3, 4, \dots, q)$
- $b_1(x) = 1, b_2(x) = x, b_i(x) = R\left(x, x^*_{i-2}\right) \ (i = 3, 4, \dots, q)$, ahol R egy nevezetes – elég hosszú, bár nem túl bonyolult – függvény (hamar látni fogjuk, hogy ez miért előnyös), annyi fontos, hogy x a $[0, 1]$ intervallumban essen (egyszerű átskálázással mindig elérhető)

Most már csak a regresszió kivitelezését kell kitalálnunk

3.2. Modellmátrix előállítása

3.2.1. A bázisfüggvények használatának ereje

A bázisfüggvények használatának két hatalmas előnye van:

- A probléma visszavezethető velük a sima lineáris regresszióra
- Sőt, ehhez a modellmátrix is könnyen előállítható

Mielőtt megkeressük a spline-ok terének egy bázisát (azaz a konkrét bázisfüggvényeket), tisztázni kellene, hogy hány bázisfüggvényt keresünk egyáltalán, azaz hány dimenziós a spline-ok függvénytere

Naiv ötlet (köbös spline-okat használva példaként): van $q-1$ szakasz ($q-2$ knot, ami meghatároz $q-3$ szakaszt meg a két vége; úgy is felfogható, hogy a két végével együtt q knot van, ami meghatároz $q-1$ szakaszt) és mindegyiken egy harmadfokú polinom (aminek 4 paramétere van), akkor az $4q-4$ paraméter

Igen ám, de vannak megkötések: a knotokban a függvényérték és az első két derivált egyezik

Minden megkötés minden pontban 1 egyenlet, az 1-gyel csökkenti a paraméterek számát: van $q-2$ knot és 3 megkötés, az $3q-6$ csökkentés, marad $q+2$ paraméter

De mivel természetes, így a végpontokban is van 1-1 megkötés: marad q paraméter, azaz q dimenziós a természetes köbös spline-ok tere (ezért neveztük a knot-ok számát $q-2$-nek!)

Természetesen itt is igaz, hogy adott, rögzített spline-osztályra (pl. természetes köbös) is végtelen sok bázis van

Köztük célszerűség alapján választhatunk

A részletek nélkül két példa:

- $b_1(x) = 1, b_2(x) = x, b_i(x) = \left|x - x^*_{i-2}\right|^3 \ (i = 3, 4, \dots, q)$
- $b_1(x) = 1, b_2(x) = x, b_i(x) = R\left(x, x^*_{i-2}\right) \ (i = 3, 4, \dots, q)$, ahol R egy nevezetes – elég hosszú, bár nem túl bonyolult – függvény (hamar látni fogjuk, hogy ez miért előnyös), annyi fontos, hogy x a $[0, 1]$ intervallumban essen (egyszerű átskálázással mindig elérhető)

Most már csak a regresszió kivitelezését kell kitalálnunk

A bázisfüggvények használatának két hatalmas előnye van:

- A probléma visszavezethető velük a sima lineáris regresszióra
- Sőt, ehhez a modellmátrix is könnyen előállítható

3.2.2. Bázisfüggvények használata másodfokú polinomnál

Legyen $b_1(x) = 1$, $b_2(x) = x$ és $b_3(x) = x^2$ a bázisunk

Legyen $b_1(x) = 1$, $b_2(x) = x$ és $b_3(x) = x^2$ a bázisunk

Az eredeti regresszió:

$$y_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2 + \varepsilon_i$$

Az eredeti regresszió:

$$y_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2 + \varepsilon_i$$

Átírva bázisokra (lényegében transzformált magyarázó változók):

$$y_i = \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \varepsilon_i$$

Átírva bázisokra (lényegében transzformált magyarázó változók):

$$y_i = \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \varepsilon_i$$

Ez már tiszta lineáris regresszió

Ez már tiszta lineáris regresszió

3.2.3. Bázisfüggvények használatának előnye

Ez úgy tűnik, hogy csak egy nagyon nyakatekert felírás egy amúgy egyszerű problémára

Ez úgy tűnik, hogy csak egy nagyon nyakatekert felírás egy amúgy egyszerű problémára

Valójában viszont egy elképesztően erőteljes dolgot nyertünk: *minden* olyan függvény, legyen bármilyen komplikált is, ami felírható bázisfüggvényekkel (azaz az osztálya függvényosztályt alkot), az berakható egy *kutyaközönséges* regresszióba (azaz lehet ő a regrssziós függvény) a fenti transzformációval, tehát

$$\sum_{i=1}^{q} \beta_i b_i(x)$$

alakban

Valójában viszont egy elképesztően erőteljes dolgot nyertünk: *minden* olyan függvény, legyen bármilyen komplikált is, ami felírható bázisfüggvényekkel (azaz az osztálya függvényosztályt alkot), az berakható egy *kutyaközönséges* regresszióba (azaz lehet ő a regrssziós függvény) a fenti transzformációval, tehát

$$\sum_{i=1}^{q} \beta_i b_i(x)$$

alakban

(Azaz minden függvény, ami egy függvénytér eleme)

(Azaz minden függvény, ami egy függvénytér eleme)

3.2.4. A bázisfüggvények ereje, 1. felvonás

Még egyszer: *minden* függvény, ami felírható bázisfüggvényekkel

Még egyszer: *minden* függvény, ami felírható bázisfüggvényekkel

Azaz: *minden*

Azaz: *minden*

…és az összesnek *pontosan ugyanúgy* az lesz az alakja, hogy

$$\sum_{i=1}^{q} \beta_i b_i(x),$$

…és az összesnek *pontosan ugyanúgy* az lesz az alakja, hogy

$$\sum_{i=1}^{q} \beta_i b_i(x),$$

egyedül a bázisfüggvényt kell az adott esetnek megfelelően megválasztani

egyedül a bázisfüggvényt kell az adott esetnek megfelelően megválasztani

Tehát a spline is mehet ugyanígy (csak megfelelő b_i-kkel)!

Tehát a spline is mehet ugyanígy (csak megfelelő b_i-kkel)!

És ha ez az alak megvan, akkor onnantól természetesen *sima lineáris regresszióval* elintézhető

És ha ez az alak megvan, akkor onnantól természetesen *sima lineáris regresszióval* elintézhető

3.2.5. A bázisfüggvények ereje, 2. felvonás

Ráadásul az $\mathbf{X}$ modellmátrix (design mátrix) előállítása is nagyon könnyű lesz: az i-edik sora

$$\left[b_1\left(x_i\right), b_2\left(x_i\right), \ldots, b_q\left(x_i\right)\right]$$

Így maga a mátrix az $\mathbf{x}$ és az $\left[1, 2, \ldots, q\right]$ vektor *külső szorzata* (tenzorszorzata), ha a művelet alatt az oszlopban szereplő érték által meghatározott bázisfüggvény sorbeli elemre történő alkalmazását értjük, tehát $i \otimes j = b_j\left(x_i\right)$, és így

$$\begin{array}{c} \begin{pmatrix} 1 & 2 & \cdots & q \end{pmatrix} \\ \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \begin{bmatrix} b_1\left(x_1\right) & b_2\left(x_1\right) & \cdots & b_q\left(x_1\right) \\ b_1\left(x_2\right) & b_2\left(x_2\right) & \cdots & b_q\left(x_2\right) \\ \vdots & \vdots & \ddots & \vdots \\ b_1\left(x_n\right) & b_2\left(x_n\right) & \cdots & b_q\left(x_n\right) \end{bmatrix} \end{array}$$

Így, a teljes modellmátrix egy lépésben megkapható…

… majd közvetlenül rakható is bele a sima lineáris regresszióba (ld. 1. előny):

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Ráadásul az $\mathbf{X}$ modellmátrix (design mátrix) előállítása is nagyon könnyű lesz: az i-edik sora

$$\left[b_1\left(x_i\right), b_2\left(x_i\right), \ldots, b_q\left(x_i\right)\right]$$

Így maga a mátrix az $\mathbf{x}$ és az $\left[1, 2, \ldots, q\right]$ vektor *külső szorzata* (tenzorszorzata), ha a művelet alatt az oszlopban szereplő érték által meghatározott bázisfüggvény sorbeli elemre történő alkalmazását értjük, tehát $i \otimes j = b_j\left(x_i\right)$, és így

$$\begin{array}{c} \begin{pmatrix} 1 & 2 & \cdots & q \end{pmatrix} \\ \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \begin{bmatrix} b_1\left(x_1\right) & b_2\left(x_1\right) & \cdots & b_q\left(x_1\right) \\ b_1\left(x_2\right) & b_2\left(x_2\right) & \cdots & b_q\left(x_2\right) \\ \vdots & \vdots & \ddots & \vdots \\ b_1\left(x_n\right) & b_2\left(x_n\right) & \cdots & b_q\left(x_n\right) \end{bmatrix} \end{array}$$

Így, a teljes modellmátrix egy lépésben megkapható…

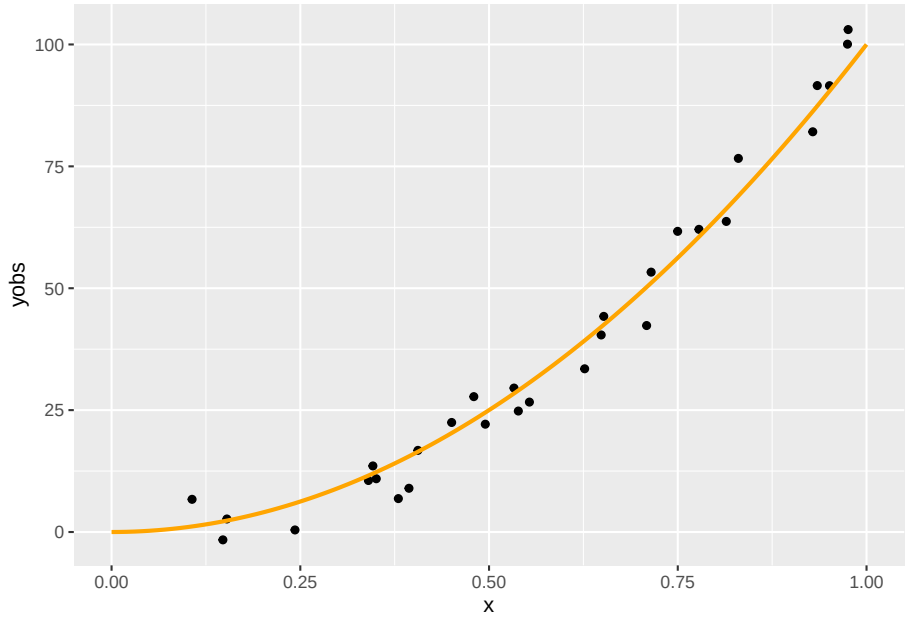
… majd közvetlenül rakható is bele a sima lineáris regresszióba (ld. 1. előny):

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

3.2.6. Megvalósítás R alatt

Folytassuk az előző fejezet példáját, csak az egyszerűség kedvéért a $\left[0, 1\right]$ intervallumon lévő x-szel (ha nem is így lenne, ez átskálázással mindig elérhető):

```
n <- 30
x <- runif(n, 0, 1)
xgrid <- seq(0, 1, length.out = 100)
ygrid <- 100*xgrid^2
yobs <- 100*x^2 + rnorm(n, 0, 5)
p <- ggplot(data.frame(x, yobs)) + geom_point(aes(x = x, y = yobs)) +
    geom_line(data = data.frame(xgrid, ygrid), aes(x = xgrid, y = ygrid),
              color = "orange", lwd = 1)
p
```



A csomópontokat egyenletesen vesszük fel, számuk $q - 2$:

```
xk <- 1:4/5
q <- length(xk) + 2
```

A bázisfüggvényeknél említett *R* függvény:

```
rk <- function( x, z ) {
    ((z-0.5)^2-1/12)*((x-0.5)^2-1/12)/4-((abs(x-z)-0.5)^4-(abs(x-z)-0.5)^2/2+7/240)/24
}
```

A modellmátrixot csupa 1-gyel inicializáljuk, így az első oszlop rendben is lesz:

```
X <- matrix(1, n, q)
```

Beállítjuk a második oszlopot is:

```
X[, 2] <- x
```

És most jön a trükk: az `outer` tetszőleges függvénnyel tud „külső szorzatot” képezni:

```
X[, 3:q] <- outer(x, xk, FUN = rk)
```

Mindezeket a későbbiekre tekintettel egy függvénybe is összefoghatjuk:

```
spl.X <- function(x, xk) {
    q <- length(xk) + 2
    n <- length(x)
    X <- matrix(1, n, q)
```

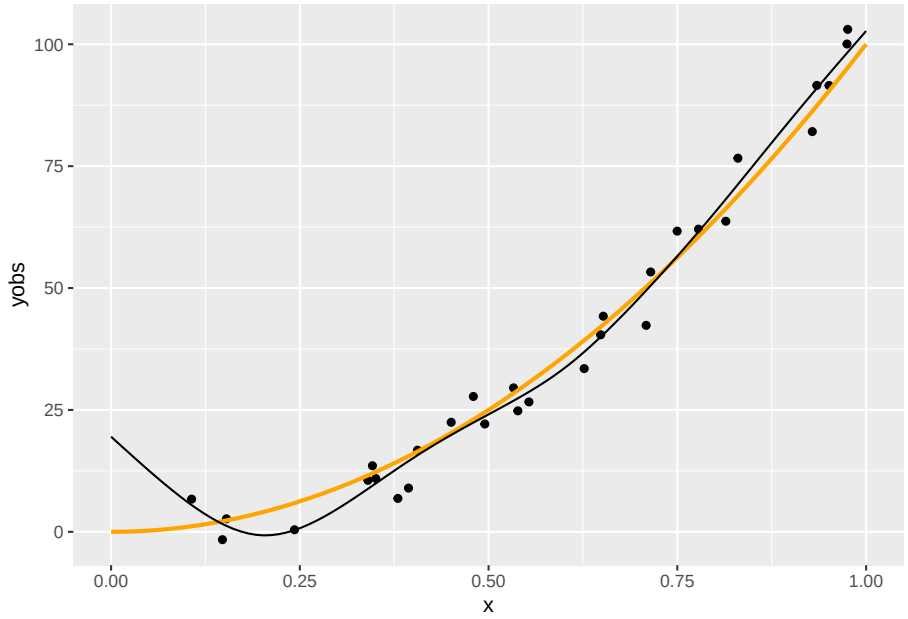
```
    X[, 2] <- x
    X[, 3:q] <- outer(x, xk, FUN = rk)
    X
}
```

Ezzel a modellmátrixszal végrehajthatjuk a regressziót (ne felejtsük, tengelymetszetre nincs szükség, pontosabban külön tengelymetszre nincs, hiszen az már benne van az így összerakott `X`-ben):

```
fit <- lm(yobs ~ X - 1 )
```

Az eredmény szemléltetéséhez az `xgrid` pontjait is kifejtjük a spline-nal:

```
Xp <- spl.X(xgrid, xk)
yp <- Xp%*%coef(fit)
p + geom_line(data = data.frame(xgrid, yp), aes(x = xgrid, y = yp))
```



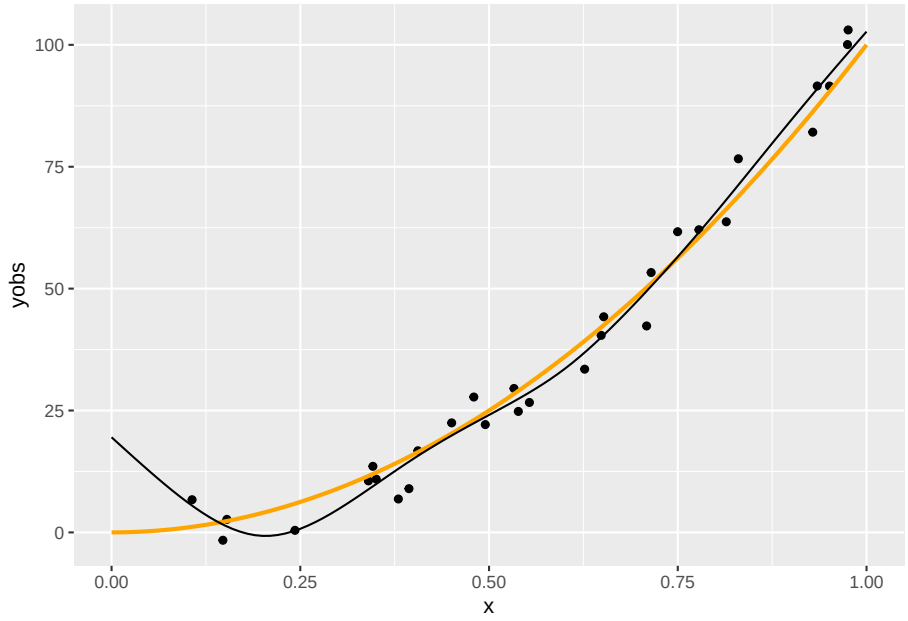
Még egy kicsit automatizáljunk:

```
predspline <- function(x, y, q) {
    xk <- (1:(q-2))/(q-1)
    X <- spl.X(x, xk)
    fit <- lm(y ~ X - 1)
    xp <- 0:100/100
    Xp <- spl.X(xp, xk)
    yp <- Xp%*%coef(fit)
    list(fit = fit, xp = xp, yp = yp)
```

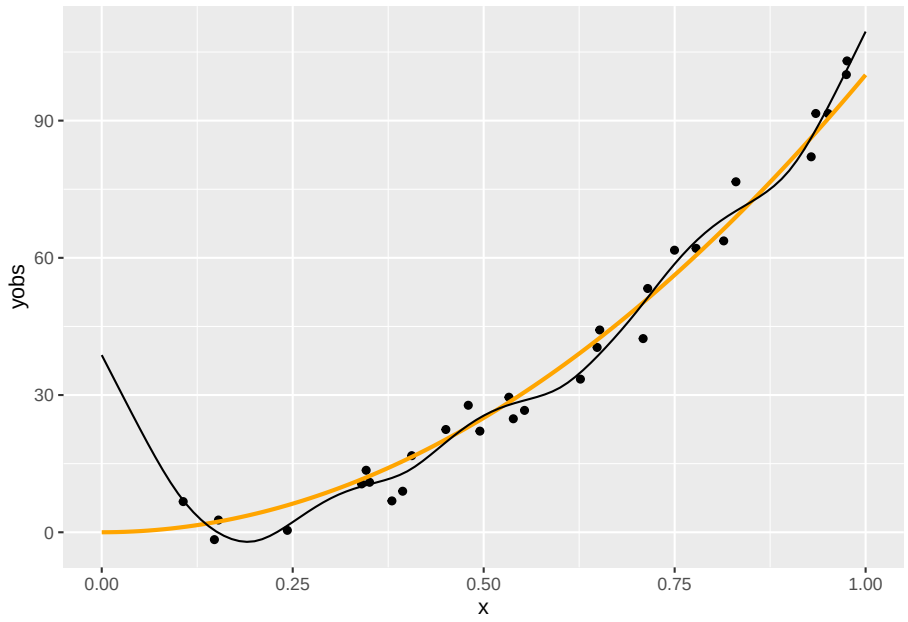
```
}
```

Így például könnyen megnézhetjük az eredményt különböző q-kkal:

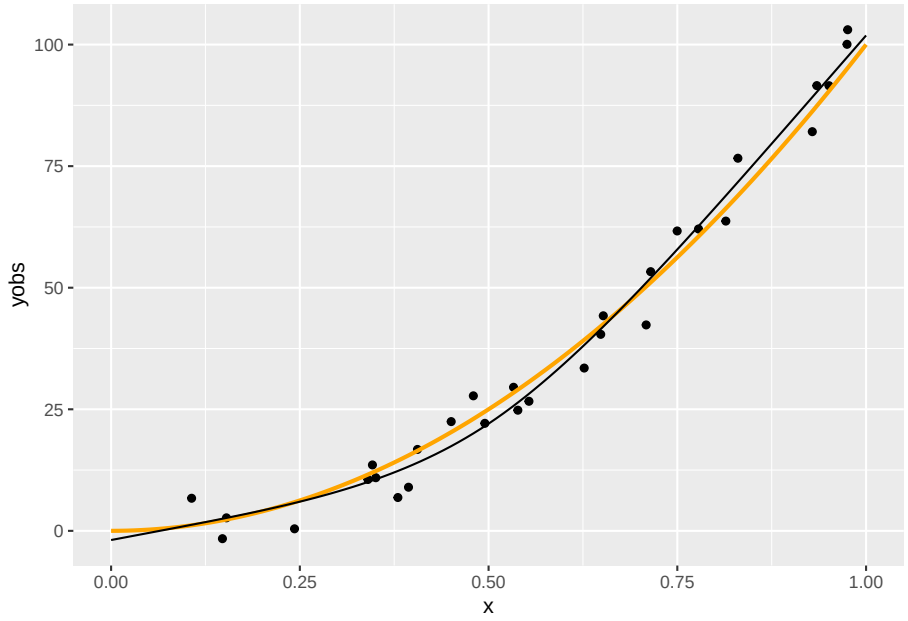
```
p + geom_line(data = with(predspline(x, yobs, 6), data.frame(xp, yp)),
              aes(x = xp, y = yp))
```




```
p + geom_line(data = with(predspline(x, yobs, 11), data.frame(xp, yp)),
              aes(x = xp, y = yp))
```




```
p + geom_line(data = with(predspline(x, yobs, 3), data.frame(xp, yp)),
              aes(x = xp, y = yp))
```



Látszik, hogy a $q = 6$ nagyjából megfelelő, a 11 kicsit sok, a 3 pedig egy leheletnyit mintha kevés lenne. (Most persze könnyű dolgunk van, hiszen tudjuk mi az igazság!)

Erre a kérdésre nemsokára visszatérünk.

3.3. Penalizálás

3.3.1. Dimenzió meghatározása

A q dimenzió tehát az illeszkedés szabadságát határozza meg

Valahogy ezt is meg kellene határozni

Jön a fő kérdéskör: a túlilleszkedés elleni védekezés

Milyen legyen a „simítás foka”?

3.3.2. Simítás fokának meghatározása

Tehát q-t kellene valahogy jól belőni

Egyszerű *modellszelekció*?

- Vagy nem beágyazott modellek szelekciója, vagy nem ekvidisztáns knot-ok, egyik sem túl szerencsés

Alternatív ötlet: q legyen inkább rögzített (elég nagy értéken, kicsit a várható fölé lőve), de a függvényformát nem engedjük teljesen szabadon alakulni

Hogyan? Büntetjük a túl „zizegős” függvényt!

Ez épp a **penalizált regresszió** alapötlete

És ami rendkívül fontos: így már jellemzően sem q pontos megválasztása, sem a knot-ok pontos helye nem bír nagy jelentőséggel (választhatjuk például egyenletesen)!

3.3.3. Penalizált regresszió

Klasszikus megoldás: a második derivált jelzi adott pontban a „zizegősséget”, ezt kiintegrálva kapunk egy összesített mértéket az egész függvényre

Valamilyen súllyal ezt vegyük figyelembe:

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \int_0^1 \left[f''(x)\right]^2 \mathrm{d}x$$

A λ a *simítási paraméter*, ez határozza meg a trade-off-ot a jó illeszkedés és a simaság között

- $\lambda = 0$: penalizálatlan becslés, $\lambda \to \infty$: egyenes regressziós függvény

3.3.4. A simasági büntetőtag meghatározása

A regressziós függvény alakja: $f(x) = \sum_{i=1}^{q} \beta_i b_i(x)$

Kétszer deriválva: $f''(x) = \sum_{i=1}^{q} \beta_i b_i''(x)$

Négyzetre emelve: $\left[f''(x)\right]^2 = \sum_{i=1}^{q}\sum_{j=1}^{q} \beta_i b_i''(x) b_j''(x) \beta_j$

Kiintegrálva: $\int_0^1 \left[f''(x)\right]^2 \mathrm{d}x = \sum_{i=1}^{q}\sum_{j=1}^{q} \beta_i \left(\int_0^1 b_i''(x) b_j''(x)\,\mathrm{d}x\right) \beta_j$

De hát ez épp egy *kvadratikus alak*! ($\sum_{i=1}^{q}\sum_{j=1}^{q} x_i a_{ij} x_j = \mathbf{x}^T\mathbf{A}\mathbf{x}$)

Legyen $S_{ij} = \int_0^1 b_i''(x) b_j''(x)\,\mathrm{d}x$ és $\mathbf{S}$ az ezekből alkotott mátrix, akkor tehát a simítási büntetőtag:

$$\lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta}$$

Az előbb definiált R-rel $\mathbf{S}$ alakja nagyon egyszerű lesz: $S_{i+2,j+2} = R\left(x_i^*, x_j^*\right)$, az első két oszlop és sor pedig csupa nulla

A regressziós függvény alakja: $f(x) = \sum_{i=1}^{q} \beta_i b_i(x)$

Kétszer deriválva: $f''(x) = \sum_{i=1}^{q} \beta_i b_i''(x)$

Négyzetre emelve: $\left[f''(x)\right]^2 = \sum_{i=1}^{q}\sum_{j=1}^{q} \beta_i b_i''(x) b_j''(x) \beta_j$

Kiintegrálva: $\int_0^1 \left[f''(x)\right]^2 \mathrm{d}x = \sum_{i=1}^{q}\sum_{j=1}^{q} \beta_i \left(\int_0^1 b_i''(x) b_j''(x)\,\mathrm{d}x\right) \beta_j$

De hát ez épp egy *kvadratikus alak*! ($\sum_{i=1}^{q}\sum_{j=1}^{q} x_i a_{ij} x_j = \mathbf{x}^T\mathbf{A}\mathbf{x}$)

Legyen $S_{ij} = \int_0^1 b_i''(x) b_j''(x)\,\mathrm{d}x$ és $\mathbf{S}$ az ezekből alkotott mátrix, akkor tehát a simítási büntetőtag:

$$\lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta}$$

Az előbb definiált R-rel $\mathbf{S}$ alakja nagyon egyszerű lesz: $S_{i+2,j+2} = R\left(x_i^*, x_j^*\right)$, az első két oszlop és sor pedig csupa nulla

3.3.5. Megvalósítás R alatt

Az `xk` szokásosan a knot-ok helye; a mátrixot pedig csupa nullával inicializáljuk, hogy az első két oszlop és sor egyből jó is legyen és csak a többit kelljen kitölteni:

```
spl.S <- function(xk) {
    q <- length(xk) + 2
    S <- matrix(0, q, q)
    S[3:q, 3:q] <- outer(xk, xk, FUN = rk)
    S
}
```

3.3.6. A simítási büntetőtag beépítése a regressziós célfüggvénybe

Kényelmes lenne, ha $\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta}$ helyett írhatnánk egyetlen normát célfüggvényként

Ez nem nehéz, ha a második tagot át tudjuk normává alakítani, hiszen (innentől némi blokkmátrix műveletekre szükség lesz)

$$\|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 = \left\|\begin{pmatrix}\mathbf{a}\\ \mathbf{b}\end{pmatrix}\right\|^2$$

Legyen $\mathbf{B}$ olyan, hogy $\mathbf{B}^T\mathbf{B} = \mathbf{S}$ (pl. spektrális dekompozícióval, vagy Cholesky-dekompozícióval megtalálható a mátrix ilyen „négyzetgyöke”), ekkor

$$\lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta} = \lambda\boldsymbol{\beta}^T\mathbf{B}^T\mathbf{B}\boldsymbol{\beta} = \lambda\left(\mathbf{B}\boldsymbol{\beta}\right)^T\mathbf{B}\boldsymbol{\beta} = \left(\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right)^T\left(\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right)$$

Kényelmes lenne, ha $\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta}$ helyett írhatnánk egyetlen normát célfüggvényként

Ez nem nehéz, ha a második tagot át tudjuk normává alakítani, hiszen (innentől némi blokkmátrix műveletekre szükség lesz)

$$\|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 = \left\|\begin{pmatrix}\mathbf{a}\\ \mathbf{b}\end{pmatrix}\right\|^2$$

Legyen $\mathbf{B}$ olyan, hogy $\mathbf{B}^T\mathbf{B} = \mathbf{S}$ (pl. spektrális dekompozícióval, vagy Cholesky-dekompozícióval megtalálható a mátrix ilyen „négyzetgyöke”), ekkor

$$\lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta} = \lambda\boldsymbol{\beta}^T\mathbf{B}^T\mathbf{B}\boldsymbol{\beta} = \lambda\left(\mathbf{B}\boldsymbol{\beta}\right)^T\mathbf{B}\boldsymbol{\beta} = \left(\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right)^T$$

Ezzel meg is vagyunk, hiszen a norma egyszerűen $\|\mathbf{a}\|^2 = \mathbf{a}^T\mathbf{a}$, így

$$\lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta} = \left\|\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right\|^2$$

ahonnan

$$\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta} = \left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \left\|\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right\|^2$$

Ezzel meg is vagyunk, hiszen a norma egyszerűen $\|\mathbf{a}\|^2 = \mathbf{a}^T\mathbf{a}$, így

$$\lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta} = \left\|\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right\|^2$$

ahonnan

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta} = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \left\|\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right\|^2$$

és így, az előzőek szerint

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \left\|\sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\right\|^2 = \left\|\begin{pmatrix}\mathbf{y} - \mathbf{X}\boldsymbol{\beta} \\ \sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\end{pmatrix}\right\|^2$$

Jó lenne $\boldsymbol{\beta}$-t kiemelni; ez nem is túl nehéz, hiszen $\mathbf{a}$ és $-\mathbf{a}$ normája ugyanaz:

$$\left\|\begin{pmatrix}\mathbf{y} - \mathbf{X}\boldsymbol{\beta} \\ \sqrt{\lambda}\mathbf{B}\boldsymbol{\beta}\end{pmatrix}\right\|^2 = \left\|\begin{pmatrix}\mathbf{y} \\ \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{X} \\ \sqrt{\lambda}\mathbf{B}\end{pmatrix}\boldsymbol{\beta}\right\|^2$$

3.3.7. Regresszió megoldása a penalizálással

Innentől a regresszió játszi könnyedséggel (értsd: a szokványos, nem is penalizált eszköztárral) megoldható, csak $\mathbf{X}$ szerepét $\begin{pmatrix}\mathbf{X} \\ \sqrt{\lambda}\mathbf{B}\end{pmatrix}$, $\mathbf{y}$ szerepét $\begin{pmatrix}\mathbf{y} \\ \mathbf{0}\end{pmatrix}$ játssza

Így az „$\mathbf{X}^T\mathbf{X}$” épp $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{B}^T\mathbf{B} = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{S}$ lesz

Az „$\mathbf{X}^T\mathbf{y}$” pedig $\mathbf{X}^T\mathbf{y}$ (a kiegészített eredményváltozóban lévő nullák épp a magyarázó változók kiegészítését ütik ki)

Így az OLS megoldás:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{S}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

(Persze a gyakorlatban ennek közvetlen számítása helyett célszerűbb az augmentált eredmény- és magyarázóváltozókat berakni egy hatékonyabb lineáris regressziót megoldó módszerbe)

3.3.8. Megvalósítás R alatt

Mátrix „gyökének” a számítása (spektrális felbontással):

```
mat.sqrt <- function(S) {
    d <- eigen(S, symmetric = TRUE)
    d$vectors%*%diag(d$values^0.5)%*%t(d$vectors)
}
```

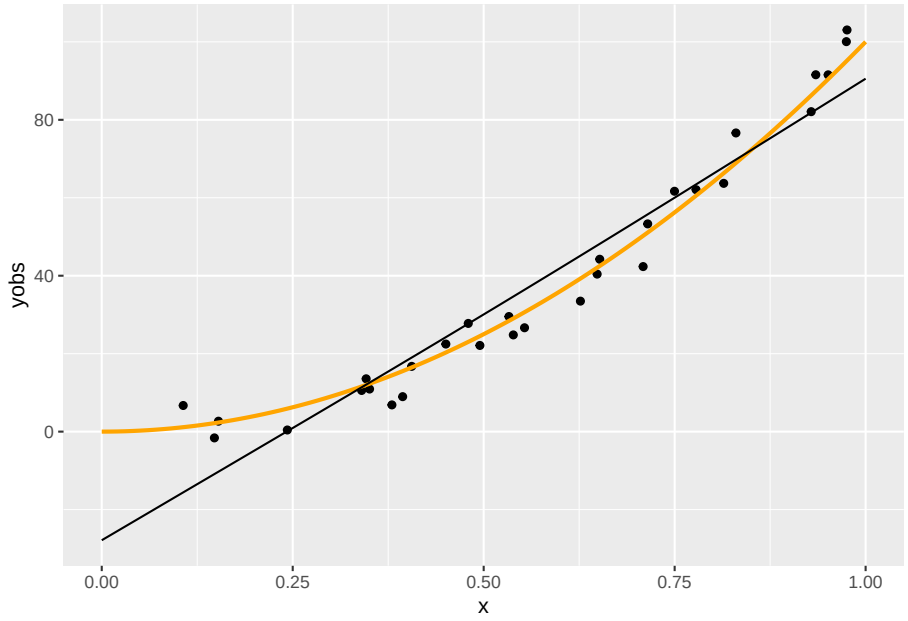
Ahogy volt róla, penalizálás mellett a q pontos értéke nem túl fontos, csak ne legyen túl kicsi, ezért használjunk most $q = 20$-at.

A penalizált becslés az augmentált modellmátrix használatával (kihasználjuk, hogy ha nem létező elemre hivatkozunk, az R automatikusan kiegészíti a vektort):

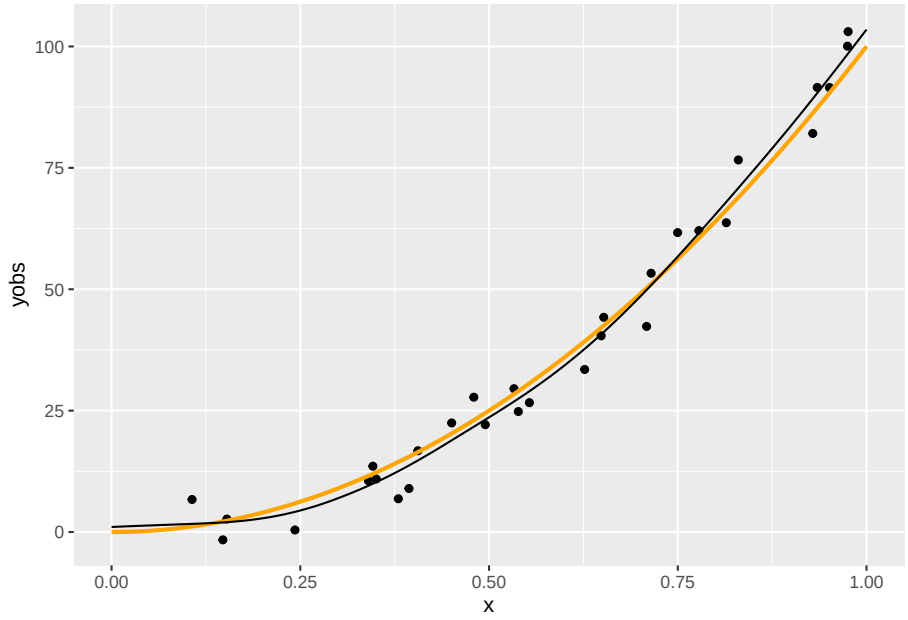
```
predsplinepen <- function(x, y, q, lambda) {
    xk <- (1:(q-2))/(q-1)
    Xa <- rbind(spl.X(x, xk), sqrt(lambda) * mat.sqrt(spl.S(xk)))
    ya <- c(y, rep(0, q))
    fit <- lm(ya ~ Xa - 1)
    xp <- 0:100/100
    Xp <- spl.X(xp, xk)
    yp <- Xp%*%coef(fit)
    list(fit = fit, xp = xp, yp = yp)
}
```

Ezzel könnyen meghatározhatjuk az eredményt különböző λ-kra:

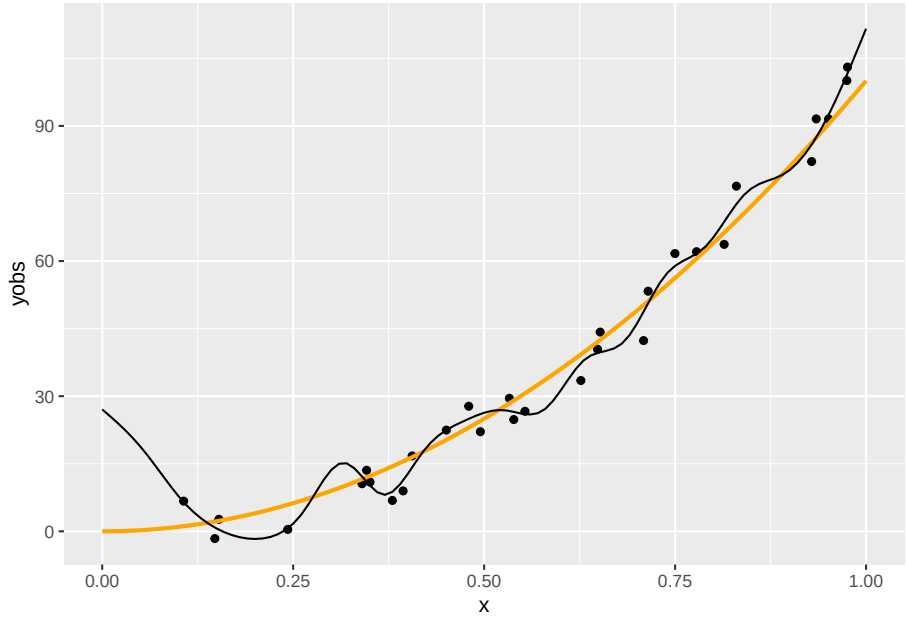
```
p + geom_line(data = with(predsplinepen(x, yobs, 20, 1), data.frame(xp, yp)),
              aes(x = xp, y = yp))
```




```
p + geom_line(data = with(predsplinepen(x, yobs, 20, 0.001), data.frame(xp, yp)),
              aes(x = xp, y = yp))
```




```
p + geom_line(data = with(predsplinepen(x, yobs, 20, 0.000001), data.frame(xp, yp)),
              aes(x = xp, y = yp))
```



Látható, hogy a $\lambda = 1$ túl nagy, a 0,001 jónak tűnik, a 0,000001 túl kicsi.

3.4. Simítási paraméter meghatározása

3.4.1. A simítási paraméter meghatározása

Kérdés még a λ értéke

Sima OLS-jellegű eljárással, tehát a reziduális négyzetösszeg minimalizálást tűzve ki célul nyilván nem határozható meg (hiszen az mindig 0-t adna)

Épp az a lényeg, hogy a túlilleszkedésre is tekintettel legyünk

Ötlet: keresztvalidáció

3.4.2. Keresztvalidációs módszerek: OCV

Mindig egy pontot hagyunk ki, és így számolunk hibát: OCV

(Szokták egy-kihagyásos keresztvalidációnak, LOOCV-nek is nevezni)

Tehát:

$$E_{OCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{f}_i^{[-i]} - y_i\right)^2$$

Szerencsére nem kell ténylegesen n-szer lefuttatni a regressziót mert belátható, hogy

$$E_{OCV} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}_i\right)^2 / \left(1 - A_{ii}\right)^2,$$

ahol **A** az influence mátrix

3.4.3. Keresztvalidációs módszerek: GCV

Ha az A_{ii}-ket az átlagukkal helyettesítjük, akkor az általánosított keresztvalidációhoz jutunk (GCV)

Tehát:

$$E_{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}_i\right)^2 / \left[\mathrm{tr}\left(\mathbf{I} - \mathbf{A}\right)\right]^2$$

3.4.4. Megvalósítás R alatt

```
predV <- 10^(seq(-8, 3, length.out = 100))
V <- sapply(predV, function(lambda) {
    fit <- predsplinepen(x, yobs, 20, lambda)$fit
    trA <- sum(influence(fit)$hat[1:n])
    rss <- sum((yobs - fitted(fit)[1:n])^2)
    n*rss/(n - trA)^2
} )
ggplot(data.frame(predV, V), aes(x = predV, y = V)) + geom_line() + scale_x_log10()
```

Kérdés még a λ értéke

Sima OLS-jellegű eljárással, tehát a reziduális négyzetösszeg minimalizálást tűzve ki célul nyilván nem határozható meg (hiszen az mindig 0-t adna)

Épp az a lényeg, hogy a túlilleszkedésre is tekintettel legyünk

Ötlet: keresztvalidáció

Mindig egy pontot hagyunk ki, és így számolunk hibát: OCV

(Szokták egy-kihagyásos keresztvalidációnak, LOOCV-nek is nevezni)

Tehát:

$$E_{OCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{f}_i^{[-i]} - y_i\right)^2$$

Szerencsére nem kell ténylegesen n-szer lefuttatni a regressziót mert belátható, hogy

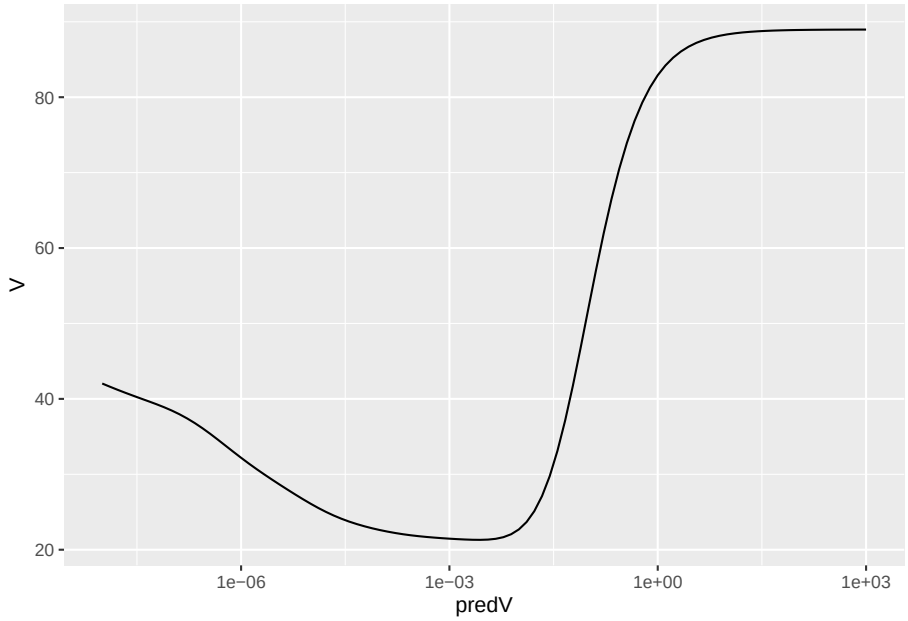
$$E_{OCV} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}_i\right)^2 / \left(1 - A_{ii}\right)^2,$$

ahol **A** az influence mátrix

Ha az A_{ii}-ket az átlagukkal helyettesítjük, akkor az általánosított keresztvalidációhoz jutunk (GCV)

Tehát:

$$E_{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}_i\right)^2 / \left[\mathrm{tr}\left(\mathbf{I} - \mathbf{A}\right)\right]^2$$



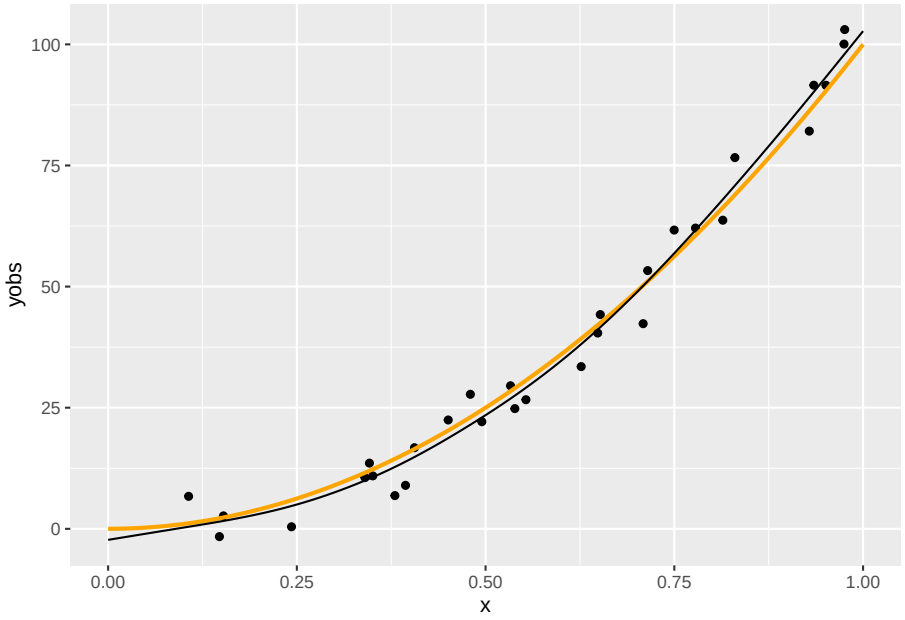
A legjobb λ konkrét érték:

```
predV[which.min(V)]
```

```
## [1] 0.002782559
```

És az – ilyen értelemben – optimális spline ezzel:

```
p + geom_line(data = with(predsplinepen(x, yobs, 20, predV[which.min(V)]),
                          data.frame(xp, yp)), aes(x = xp, y = yp))
```


100
75
50
25
0
yobs
0.00
0.25
0.50
0.75
1.00
x

4. fejezet

Additív modellek

4.1. Több magyarázó változó

Eddig egy magyarázó változó esetével foglalkoztunk

Eddig egy magyarázó változó esetével foglalkoztunk