# Análisis Avanzado de Datos.

Nicolás López

Primer semestre de 2023

# Los problemas de la linealidad

# Los problemas de la linealidad

Considerar que el proceso generador de datos $P_\theta$ sigue una línea recta es a lo menos, muy optimista:

```
library('ISLR2')
sample_size <- nrow(Auto)
set.seed(456)
train       <- sample(sample_size, 0.5*sample_size)
test        <- seq(sample_size)[!seq(sample_size) %in% train]

mod1 = lm(mpg ~ horsepower,data=Auto[train,])
mse1 = mean((Auto$mpg[test] - predict(mod1,Auto)[test])**2)
mod2 = lm(mpg ~  poly(horsepower,2),data=Auto[train,])
mse2 = mean((Auto$mpg[test] - predict(mod2, Auto)[test])**2)

print(mse1)
```

```
## [1] 22.27728
```

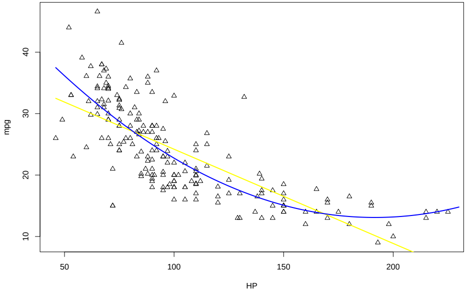
```
print(mse2)
```

```
## [1] 16.38132
```

Tan sólo considerar la **regresión polinomial** decrece de manera importante el ECM de prueba.

Visualmente es claro que el proceso no sigue RLS:

```
xvals = data.frame(horsepower = seq(min(Auto$horsepower),
                                    max(Auto$horsepower,by=1)))
plot(Auto$horsepower[train],Auto$mpg[train],xlab="HP",ylab="mpg",pch=2)
lines(xvals$horsepower,predict(mod1,xvals),lwd=2,col="yellow")
lines(xvals$horsepower,predict(mod2,xvals),lwd=2,col="blue")
```

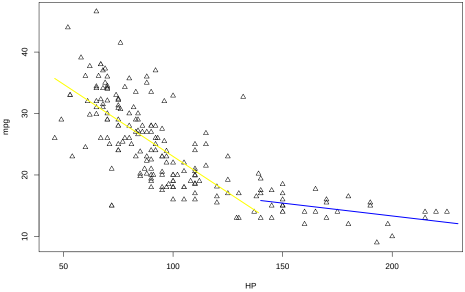



Sin embargo podríamos “forzar” el modelo RLS separando el dominio de $x$.

# Podemos ajustar regresiónes **lineales a trozos** para procurar resolver el problema

```
mod11 = lm(mpg ~ horsepower,data=Auto[train,] %>% filter(horsepower< 140))
mod12 = lm(mpg ~ horsepower,data=Auto[train,] %>% filter(horsepower>=140))

plot(Auto$horsepower[train],Auto$mpg[train],xlab="HP",ylab="mpg",pch=2)
lines(xvals$horsepower[xvals$horsepower<140] ,predict(mod11,xvals %>% filter(horsepower< 140)),lwd=2,col="yellow")
lines(xvals$horsepower[xvals$horsepower>=140],predict(mod12,xvals %>% filter(horsepower>=140)),lwd=2,col="blue")
```



## Y calcular el ECM de prueba

```
mse11 = ((Auto[test,] %>% filter(horsepower< 140))$mpg -
              predict(mod11,Auto[test,] %>% filter(horsepower< 140)))**2

mse12 = ((Auto[test,] %>% filter(horsepower>=140))$mpg -
              predict(mod12,Auto[test,] %>% filter(horsepower>=140)))**2

mean(c(mse11,mse12))
```

```
## [1] 17.25907
```

## Obteniendo un valor relativamente cercano al polinomio de grado dos. ¿Ve algún problema en este acercamiento?

Hay por lo menos dos problemas:

1. El punto en el eje $x$ es arbitrariamente definido.
2. Nuevamente, ¿es realista pensar que el proceso que genera los datos tiene esta forma lineal?

Note además que para el polinomio de grado 2 hacen falta 3 parámetros a ser estimados, mientras que para las líneas rectas se requieren un total de 4.

Es un problema que aún mediante técnicas sofisticasdes de penalización (ridge, lasso), se presenta. Recuerde que el problema subyacente que solucionamos con dichos métodos fue reducir la varianza de los estimadores, pero **seguimos bajo un modelo lineal**.

# Regersión polinomial

# Regersión polinomial

La generalización simple del modelo lineal simple dada por polinomios en la covariable no lineal:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_p x_i^p + \epsilon_i$$

Puede escribirse como

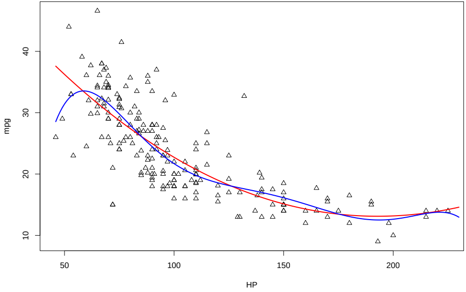
$$y_i = \beta_0 f^0(x_i) + \beta_1 f^1(x_i) + \beta_2 f^2(x_i) + ... + \beta_p f^p(x_i) + \epsilon_i$$

Con $f^h(x) = x^h$,y así, es claro que resulta en un modelo de RLM con covariables dadas por los polinomios de $x_i$, por lo cual su estimación puede darse bajo mínimos cuadrados.

A medida que incremento $p$, la curva va a sobreajustarse a los datos, y particularmente en los extremos va a tender a mostrar un comportamiento errático (poco natural).

```
mod_lm3 = lm(mpg ~  poly(horsepower,3),data=Auto[train,])
mod_lm6 = lm(mpg ~  poly(horsepower,6),data=Auto[train,])

plot(Auto$horsepower[train],Auto$mpg[train],xlab="HP",ylab="mpg",pch=2)
lines(xvals$horsepower,predict(mod_lm3,xvals),lwd=2,col="red")
lines(xvals$horsepower,predict(mod_lm6,xvals),lwd=2,col="blue")
```

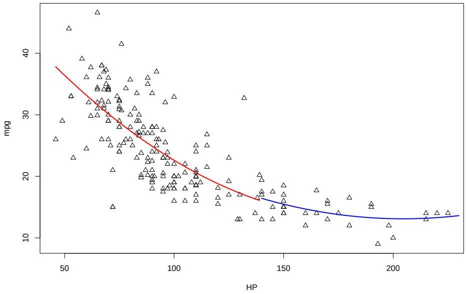



Usualmente $p < 4$.

Sin embargo, el ajuste con un polinomio de bajo grado a trozos (o **ajustado localmente**) podría darnos un modelo menos errático en los extremos y aún más preciso:

```
mod11_2 = lm(mpg ~ poly(horsepower,2),data=Auto[train,] %>% filter(horsepower< 140))
mod12_2 = lm(mpg ~ poly(horsepower,2),data=Auto[train,] %>% filter(horsepower>=140))

plot(Auto$horsepower[train],Auto$mpg[train],xlab="HP",ylab="mpg",pch=2)
lines(xvals$horsepower[xvals$horsepower<140] ,predict(mod11_2,xvals %>% filter(horsepower< 140)),lwd=2,col="red")
lines(xvals$horsepower[xvals$horsepower>=140],predict(mod12_2,xvals %>% filter(horsepower>=140)),lwd=2,col="blue")
```




Los puntos donde los coeficientes cambian son llamados *knots*.

## Y nuevamente calcular el ECM de prueba

```
mse11_2 = ((Auto[test,] %>% filter(horsepower< 140))$mpg -
              predict(mod11_2,Auto[test,] %>% filter(horsepower< 140)))**2

mse12_2 = ((Auto[test,] %>% filter(horsepower>=140))$mpg -
              predict(mod12_2,Auto[test,] %>% filter(horsepower>=140)))**2

mean(c(mse11_2,mse12_2))
```

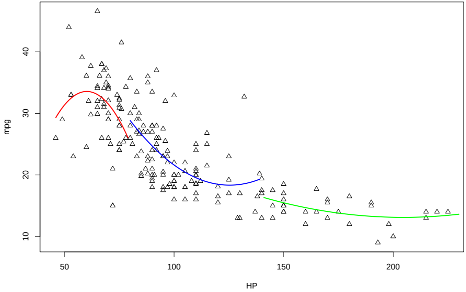
```
## [1] 16.35463
```

Obteniendo un valor aún más bajo que el del polinomio de grado dos **ajustado globalmente**. En esta caso fueron 6 parámetros estimados en la estimación de la variable respuesta. Note que los problemas de los trozos lineales se mantienen, pero ahora con un mejor ajuste.

Quizá un tercer modelo permita un mejor ajuste:

```
mod11_3 = lm(mpg ~ poly(horsepower,2),data=Auto[train,] %>% filter(horsepower< 80))
mod12_3 = lm(mpg ~ poly(horsepower,2),data=Auto[train,] %>% filter(horsepower>= 80 & horsepower<140))
mod13_3 = lm(mpg ~ poly(horsepower,2),data=Auto[train,] %>% filter(horsepower>=140))

plot(Auto$horsepower[train],Auto$mpg[train],xlab="HP",ylab="mpg",pch=2)
lines(xvals$horsepower[xvals$horsepower<80] ,
      predict(mod11_3,xvals %>% filter(horsepower< 80)),lwd=2,col="red")
lines(xvals$horsepower[xvals$horsepower>=80 & xvals$horsepower<140],
      predict(mod12_3,xvals %>% filter(horsepower>=80 & horsepower<140)),lwd=2,col="blue")
lines(xvals$horsepower[xvals$horsepower>140] ,
      predict(mod13_3,xvals %>% filter(horsepower> 140)),lwd=2,col="green")
```




Obteniendo un resultado **ajustado localmente** que pareciera no tener mucho sentido como $P_\theta$: hay demasiada flexibilidad, un total de 9 parámetros fueron estimados en dicho ajuste.

# Nuevamente podemos calcular el ECM de prueba

```
mse11_3 = ((Auto[test,] %>% filter(horsepower< 80))$mpg -
              predict(mod11_3,Auto[test,] %>% filter(horsepower < 80)))**2

mse12_3 = ((Auto[test,] %>% filter(horsepower>=80 & horsepower < 140))$mpg -
              predict(mod12_3,Auto[test,] %>% filter(horsepower>=80 & horsepower < 140)))**2

mse13_3 = ((Auto[test,] %>% filter(horsepower>=140))$mpg -
              predict(mod13_3,Auto[test,] %>% filter(horsepower>=140)))**2

mean(c(mse11_3,mse12_3,mse13_3))
```

```
## [1] 16.83014
```

Problemas:

1. ¿Cómo evitar esos saltos de discontinuidad en los *knots*?
2. ¿Cómo determinar qué tanta flexibilidad (grado) debe terner el polinomio en un ajuste local?

# Polinomios truncados y base de funciones

## Polinomios truncados y base de funciones

El primer problema en el $k$-ésimo *knot* se puede solucionar al **adicionar polinomios truncados** de grado $h$ a la regresión polinomial usual, los cuales están dados por:

$$g^h(x,k) = \begin{cases} (x-k)^h & x \geq k \\ 0 & x < k \end{cases}$$

El valor de $h$ determina la continuidad en las $h-1$ derivadas en $k$. Es decir, si por ejemplo $h=3$, y $k=80hp$, la función politómica estimada es continua en $k$ en su primera y segunda derivada, dando mayor suavidad a la función resultante.

Antes de ajustar el modelo correspondiente, note que para $h=1$ con un *knot* $k$ se ajusta un modelo lineal continuo en $k$ dado por:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 g^h(x_i, k) + \epsilon_i = \beta_0 + \beta_1 x_i + \beta_2 (x_i - k) I(x_i \geq k) + \epsilon_i$$

Es igual a

$$y_i = \begin{cases} \beta_0 + \beta_1 x_i & x < k \\ \beta_0 + \beta_1 x_i + \beta_2 (x_i - k) & x \geq k \end{cases}$$

Con lo cual, en $k$, la función es continua.

```
mod1_knot = lm(mpg ~ horsepower + I((horsepower-80)*(horsepower>=80))
                                + I((horsepower-140)*(horsepower>=140)) ,
                                data=Auto[train,])
mse1_knot = mean((Auto$mpg[test] - predict(mod1_knot,Auto)[test])**2)
print(mse1_knot)
```

```
## [1] 16.68567
```

Obtieniendo, con los mismos 4 parámetros que usa el modelo lineal a trozos, un menor error de predicción. Note además que el error obtenido es cercano al encontrado bajo el modelo polinomial global de grado 2.

## Ahora el modelo polinomial a trozos

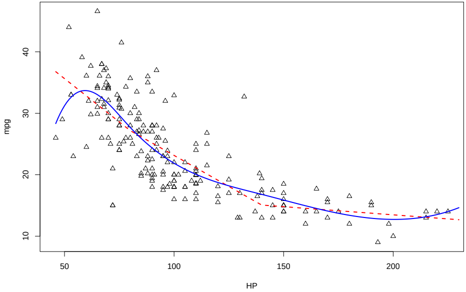
```
mod2_knot = lm(mpg ~ horsepower + I(horsepower^2)
                                + I(horsepower^3)
                                + I(((horsepower-80)*(horsepower>=80))^3)
                                + I(((horsepower-140)*(horsepower>=140))^3) ,
                                data=Auto[train,])
mse2_knot = mean((Auto$mpg[test] - predict(mod2_knot,Auto)[test])**2)
print(mse2_knot)
```

```
## [1] 16.92826
```

Con uno de los menores errores de predicción vistos en los modelos hasta ahora ajustados.

## Gráficamente se obtiene

```
xvals = data.frame(horsepower = seq(min(Auto$horsepower),
                                    max(Auto$horsepower,by=1)))
plot(Auto$horsepower[train],Auto$mpg[train],xlab="HP",ylab="mpg",pch=2)
lines(xvals$horsepower,predict(mod1_knot,xvals),lwd=2,col="red",lty=2)
lines(xvals$horsepower,predict(mod2_knot,xvals),lwd=2,col="blue",lty=1)
```




1. Siempre se prefiere el modelo más **parsimonioso**, con lo cual, el polinomio grado dos resume las características no lineales globales. Sin embargo, esto sucede con poca frecuencia en la práctica. El ejemplo permitió ejemplificar el **spline lineal** y el **spline cúbico**.
2. En general, un spline de grado $d$ es un polinomio de grado $d$ definido a trozos con continuidad hasta la $d-1$ derivada en cada *knot*.

Hay múltiples formas de representar un spline de grado $d$. La adición mediante polinomios truncados es una de múltiples posibles elecciónes de **bases de funciones**:

$$\{\phi_i\} = \{\phi_1 = 1, \phi_2 = x, \phi_3 = x^2, \phi_4 = x^3, \phi_5 = g^3(x, k)\}$$

Similar a la base de un espacio vectorial. Se pueden encontrar bases $B$-spline, bases de fourier y Wavelets, entre otras bases de $f$.

# Splines de regresión

## Splines de regresión

Independiente de la base $\{\phi\}_i$ que seleccione se tiene un problema de RLM subyacente que se puede resolver mediante MCO, ya que:

$$y_i = \sum_j \beta_j \phi_j(x_i) + \epsilon_i = f(x|\beta_1, ..., \beta_p, \phi_1, ..., \phi_p) + \epsilon_i = f(x) + \epsilon_i$$

Con lo que la SCE esta dada por

$$SC_E = SC_E(\beta_1, ..., \beta_p) = \sum_i (y_i - \sum_j \beta_j \phi_j(x_i))$$

Dado lo último, se le llama **splines de regresión** al proceso de **suavizamiento** de la curva esperada de $y$ mediante una base de funciones en $x$ definida a través de uno o varios *knots*.

Como vimos anteriormente, la flexibilidad de la regresión depende de dos factores:

1. Ubicación de los *knots*.
2. Grado del polinomio.

Los *knots* dependen de la **suavidad esperada** y el polinomio del **interés en las derivadas**. Usualmente $d = 3$.

Note además que ya solucionamos el primer problema de los polinomios locales (continuidad en *knots*) sin embargo, el segundo problema sigue abierto (flexibilidad de la función resultante).

1. Una solución está dada por la correcta ubicación de *knots* y grado del polinomio.
2. Otra solución está dada mediante el **suavizamiento spline**.

# Suavizamiento spline

# Suavizamiento spline

La elección de los *knots* en la regresión spline (independiente de la base), resulta en una elección subjetiva y difícil, ya que se desconoce el proceso (funcional) subyacente que genera la relación entre $x$ y $y$. Esto puede ser solucionando nuevamente al penalizar la función resultante, sea $\lambda \geq 0$:

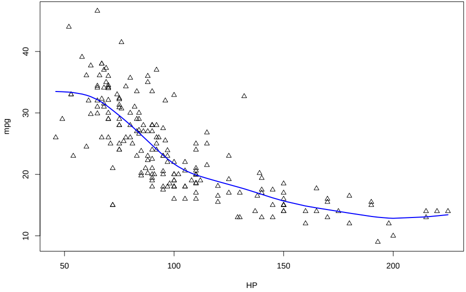
$$SC_{E_S} = \sum_i (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx = SC_E + \lambda \int f''(x)^2 dx$$

Que al ser minimizado, busca reducir el error y el *ruido*/*rugosidad* de la función $f$ en su dominio.

1. Si $\lambda$ es muy pequeño, $f$ interpola los puntos: mucha *libertad* (muchos parámetros estimados, en el caso de regresión spline).
2. Si $\lambda$ es muy grande, $f$ es muy suave (línea recta): poca *libertad* (pocos parámetros estimados, en el caso de regersión spline).

Nuevamente, $\lambda$ es determinado a partir de valudación cruzada. Y este caso permite LOOCV con el costo computacional de una sola regresión.

```
plot(Auto$horsepower[train],Auto$mpg[train],xlab="HP",ylab="mpg",pch=2)
mod_sspline = smooth.spline(Auto$horsepower[train], Auto$mpg[train], cv = TRUE)
lines(mod_sspline, col = " blue ", lwd = 2)
```



## Y el error de prueba para los datso trabajados hasta ahora

```
mse2_knot = mean((Auto$mpg[test] - predict(mod_sspline,Auto[test,]$horsepower)$y)**2)
print(mse2_knot)
```

```
## [1] 16.35966
```

## Resulta en el menor error de predicción visto en los modelos ajustados.

# Referencias

# Referencias

1. Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical Learning. Springer.
2. Garet, Witten, Hastie, Tibshirani. Introduction to Statistical Learning with R.