Trabajo funcionales: Modelado y Análisis de Espectros en Sistemas LHR con Penalización Adaptativa

Hugo Mugüerza Mendoza

2025-05-02

Introducción:

Los sistemas de **radiometría heterodina láser** (LHR, por sus siglas en inglés) representan una de las tecnologías más prometedoras para la **detección remota de gases de efecto invernadero (GEI)** en la atmósfera. En los últimos dos siglos, la actividad humana ha liberado cantidades masivas de GEI, como el dióxido de carbono (CO2) y el metano (CH4), provocando un **aumento significativo de la temperatura media del planeta**. Las consecuencias de este calentamiento global son múltiples y evidentes: desde la elevación del nivel del mar hasta fenómenos meteorológicos extremos cada vez más frecuentes. Frente a este escenario, es fundamental **identificar, caracterizar y monitorizar con precisión las fuentes y sumideros de GEI** a escala global. Para ello, se requiere no solo el uso de satélites y redes de observación global, sino también de **instrumentación terrestre precisa, portátil y sensible**, como el LHR.

El principio de funcionamiento de un sistema LHR se basa en la **interferencia óptica heterodina** entre la luz solar, que ha atravesado la atmósfera y ha sido parcialmente absorbida por los gases presentes, y un **láser local sintonizable**, cuya frecuencia se barre paso a paso. Esta interferencia genera una señal de radiofrecuencia (RF) que se detecta mediante un fotodetector. Cuando la frecuencia del láser coincide con una línea de absorción atmosférica, la intensidad de la señal RF disminuye, lo que permite reconstruir un **espectro de absorción** característico. En el presente estudio se utilizó este sistema para capturar y representar los espectros resultantes, tal como se muestra en la Figura 1. Cada caída (valle) en las curvas representa una **línea de absorción** asociada a la presencia de un gas específico en la atmósfera, y su análisis permite **inferir la composición del aire** en la trayectoria óptica medida.

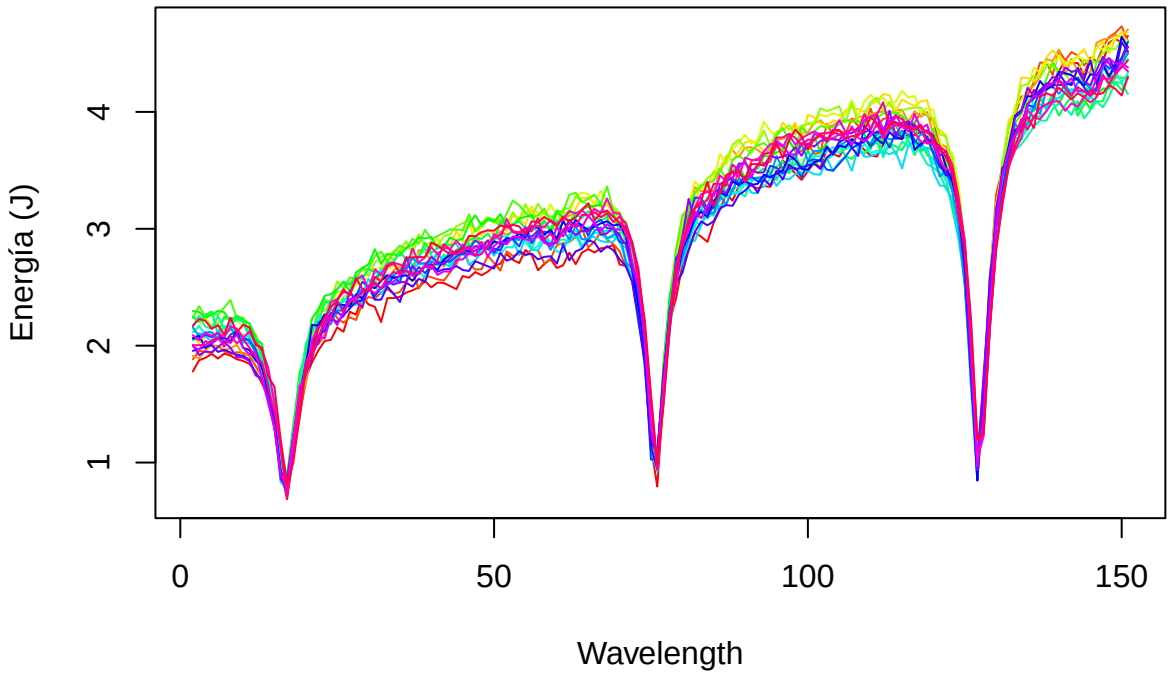
Los datos representados fueron obtenidos mediante **múltiples barridos del láser**, combinando su señal con la luz solar incidente en distintas condiciones. El resultado es un conjunto de espectros que muestran **consistencia y repetibilidad**, donde se identifican claramente varias líneas de absorción. Este enfoque, además de ser altamente sensible, permite realizar mediciones en ubicaciones puntuales, incluyendo entornos urbanos o industriales, donde otras tecnologías más voluminosas no serían operativas. Así, el sistema LHR se posiciona como una herramienta clave para el monitoreo de gases traza atmosféricos con aplicaciones científicas y ambientales de alto impacto.

```
source("Psplines.R")
source("GCV_Psplines.R")

#install.packages("C:/.../fda_2.4.8.1.tar", repos = NULL, type = "source", )
#install.packages("fds")
```

```
library(fda)
library(fds)
library(refund)
library(ggplot2)
```

```r
library(SOP)

# Cargar los datos
data <- read.csv("atmosphere2.csv", header = FALSE,row.names = NULL)
data <- as.matrix(data)

# Transponer para tener las curvas como columnas (formato estándar en fda/refund)
data_t <- t(data)

data_t[, 2:21] <- 10 * data_t[, 2:21]

data_t=data_t[2:151,]
#x_vals <- as.numeric(gsub("X", "", colnames(data)))

#order_index <- order(x_vals)
#x_vals <- x_vals[order_index]
#data_t <- data_t[order_index,]  # reordenar columnas del dataset también

#row.names(data_t)=x_vals

# Definir los puntos del dominio
argvals <- c(2:151)
matplot(argvals, data_t, type = "l", lty = 1, cex = 2,
        col = rainbow(ncol(data_t)), lwd = 1,
        xlab = "Wavelength", ylab = "Energía (J)",
        main = "Figura 1. Espectros de absorción")
```

Figura 1. Espectros de absorción



Se muestran los espectros de absorción obtenidos mediante LHR en distintos puntos de la periferia urbana de Madrid, bajo diversas condiciones ambientales y horarios. Se observa la presencia de tres líneas de absorción del C02 en torno a 1570.28 nm, 1570.54 nm y 1570.82 nm, medidas con una resolución óptica de 600 MHz (0.02 cm-1), estándar en redes de detección terrestre de GEI.

Como puede apreciarse, las curvas presentan una considerable variabilidad y ruido de alta frecuencia, lo que dificulta el análisis cuantitativo y la identificación precisa de características espectrales relevantes, como los picos de absorción.

Debido a esta variabilidad, surge la necesidad de aplicar un proceso de suavizado que permita mejorar la interpretabilidad de los datos sin perder las características estructurales clave. Para ello, se optó inicialmente por utilizar suavizado mediante P-splines adaptativos, una técnica flexible basada en modelos aditivos penalizados que permite controlar el grado de suavidad local mediante parámetros de penalización que varían a lo largo del dominio.

Suavizado adaptativo con penalización (P-splines)

Dada una base funcional Φ (por ejemplo, B-splines), cada curva $f_i(x)$ se aproxima como:

$$f_i(x) = \Phi a_i$$

donde a_i son los coeficientes a estimar para la curva i.

Estimación penalizada (no adaptativa)

Los coeficientes se obtienen minimizando:

$$(X_i - \Phi a_i)'(X_i - \Phi a_i) + \lambda a_i' P_d a_i$$

- X_i: observaciones de la curva i
- λ: parámetro de suavizado global
- P_d: matriz de penalización (por ejemplo, diferencias de orden $d = 2$)

Solución cerrada:

$$\hat{a}_i = (\Phi^\top \Phi + \lambda P_d)^{-1} \Phi^\top X_i$$

Selección del parámetro de suavizado λ en P-splines

El parámetro de suavizado λ controla el grado de suavidad de la curva ajustada:
- Si λ es grande, se penaliza mucho la rugosidad ,curva más suave.
- Si λ es pequeño, se penaliza poco ,curva más ajustada (más flexible).

Criterio de validación cruzada generalizada (GCV)

La selección óptima de λ se realiza minimizando el error de predicción esperado mediante el criterio de **validación cruzada generalizada** (GCV):

$$GCV(\lambda) = \frac{1}{N} \sum_{i=1}^{N} GCV_i(\lambda)$$

donde cada término $GCV_i(\lambda)$ se calcula para una curva i como:

$$GCV_i(\lambda) = \left(\frac{m_i + 1}{m_i + 1 - df(\lambda)} \right) \left(\frac{MSE_i}{m_i + 1 - df(\lambda)} \right)$$

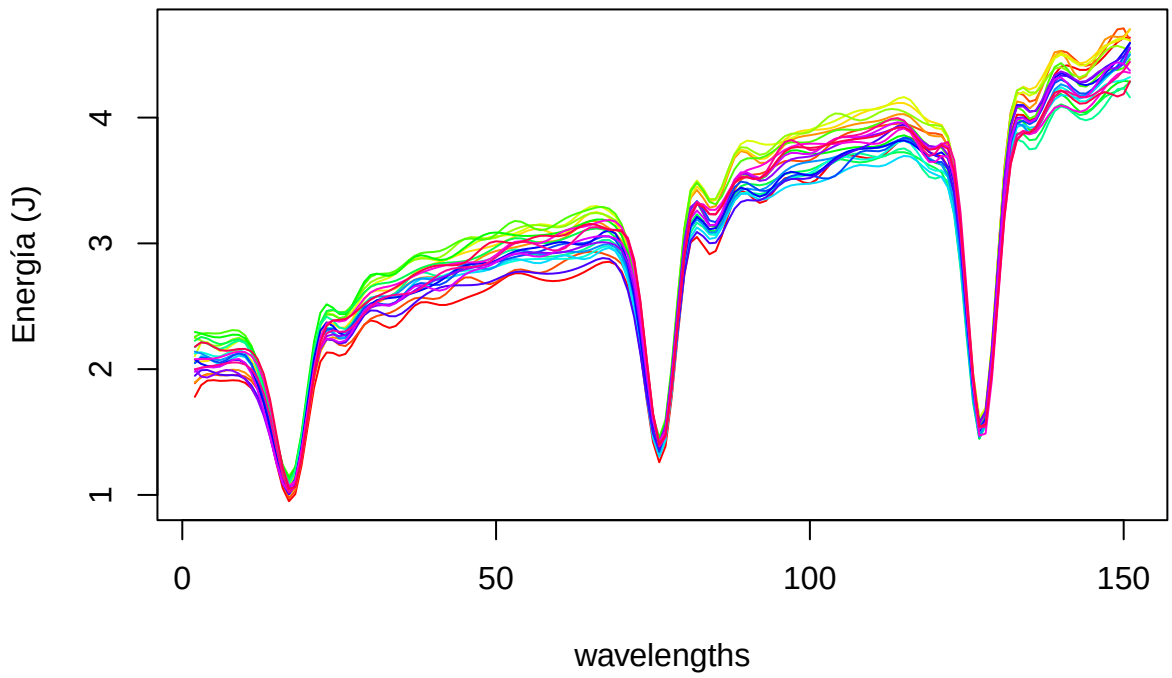
Definiciones:

- m_i: número de observaciones en la curva i
- $MSE_i = \frac{1}{m_i} \sum_{k=1}^{m_i} (X_{ik} - \hat{X}_{ik})^2$: error cuadrático medio de la curva
- $df(\lambda) = \text{trace}(H_i)$: grados de libertad efectivos
- $H_i = \Phi_i (\Phi_i^\top \Phi_i + \lambda P_d)^{-1} \Phi_i^\top$: matriz de proyección ("hat matrix")

El valor óptimo de λ es el que **minimiza** el GCV total. Es decir, la media de los GCV de cada curva. Este proceso se hace automáticamente en muchas implementaciones (como `mgcv` o `sop`)

```
# Define basis functions following Ruppert's rule
nbasis <- 41
# Define a range of lambda values
Vectlambda <- 10^seq(-6, 1, length.out = 100)

# Define the penalty order
dorder <- 3

# Compute optimal lambda using GCV_Psplines
result <- GCV_Psplines(argvals, data_t, nbasis, Vectlambda, dorder)

# Extract optimal lambda
optimal_lambda <- result$lambda

Pspline_fit<-Psplines(argvals=argvals, y=data_t, nbasis=nbasis,lambda=optimal_lambda,dorder=2)

basisobj <- Pspline_fit$Psplines$basis
coefs <- Pspline_fit$Psplines$coefs
eval_basis <- eval.basis(argvals, basisobj)
Pspline_estimated <- eval_basis %*% coefs

matplot (argvals, Pspline_estimated, type="l", lty=1, cex=2,
         col=rainbow(ncol(data_t)), lwd=1,  xlab = "wavelengths",
         ylab="Energía (J)",  main = "Figura 2. Espectros de absorción suavizados")
```

Figura 2. Espectros de absorción suavizados



En la Figura 2 se observan los espectros de absorción suavizados mediante P-splines con un parámetro de suavizado global. El valor óptimo de λ estimado es 2.9836472×10^{-4}. Aunque el resultado general es adecuado, se evidencia que un mismo nivel de suavizado no es suficiente para capturar correctamente la estructura local de las curvas: en regiones con transiciones rápidas (alrededor de los picos y valles principales), el suavizado tiende a sobreajustar o a suavizar en exceso, mientras que en zonas más planas se conserva un nivel de detalle innecesario. Esto pone de manifiesto la necesidad de aplicar un suavizado **adaptativo**, que permita variar la penalización localmente en función de la complejidad de la señal, mejorando la fidelidad de las estimaciones en todo el dominio.

Suavizado adaptativo (vector de λ)

En el enfoque clásico de P-splines, la función desconocida $f(x)$ se representa como una combinación lineal de funciones base B-spline:

$$f(x) = \sum_{j=1}^{c} \theta_j B_j(x)$$

La suavidad de la curva ajustada se controla penalizando las diferencias finitas de orden q entre los coeficientes θ_j:

$$P = \lambda \sum_{k=q+1}^{c} (\Delta^q \theta_k)^2 = \lambda \theta^\top D^\top D \theta$$

donde:

- λ es el parámetro de suavizado global
- D es la matriz de diferencias finitas de orden q
- $\Delta^q\theta_k$ representa la k-ésima diferencia entre coeficientes adyacentes. Al tratarse de diferencias de orden 2 ($q = 2$), el puntero k recorrerá el vector de coeficientes θ_j empezando por 3 hasta c, número de funciones básicas. Este al multiplicarse por Δ^q usará en la primera diferencia desde θ_1.

$$\theta_3\Delta^2 = \theta_3 \ -2\theta_2 \ +\theta_1.$$

Suavizado adaptativo

En lugar de usar un único valor de λ para todas las diferencias, se permite que **cada diferencia** tenga su propio parámetro de suavizado λ_k, lo que da lugar a una penalización más flexible:

$$P = \sum_{k=q+1}^{c} \lambda_k(\Delta^q\theta_k)^2 = \theta^\top D^\top \mathrm{diag}(\lambda) D\theta$$

donde $\lambda = (\lambda_{q+1}, \ldots, \lambda_c)^\top$ es ahora un vector con tantos parámetros como diferencias finitas.

Sin embargo, esto generaría demasiados parámetros y podría llevar a problemas de **sobreajuste** o **inestabilidad**. Para evitarlo, se modela λ como una función suave de la posición k, utilizando una base B-spline:

$$\lambda = \Psi\xi$$

- Ψ es una matriz de regresión B-spline de dimensión $(c-q)\times p$, siendo p el número de funciones Bspline usadas para estimar la curva de λ.
- $\xi_=(\xi_1, \ldots, \xi_p)^\top$ es el nuevo **vector reducido de parámetros de suavizado**
- El número de parámetros p se elige menor que $c - q$ para controlar la complejidad

Cada parámetro de suavización local λ_k se modela como una combinación lineal de funciones base B-spline:

$$\lambda_k = \sum_{p=1}^{P} \Psi_{kp} \cdot \xi_p$$

donde:

- Ψ_{kp}: valor de la p-ésima función base en la posición k
- ξ_p: coeficiente asociado a la función base p, que controla el nivel de suavizado

Esta formulación permite que **cada parámetro ξ_p actúe sobre múltiples diferencias** entre coeficientes originales. Gracias al solapamiento natural de las funciones B-spline, cada λ_k está influenciado por varias ξ_p, y a su vez cada ξ_p contribuye a suavizar **una región completa del dominio**. Esto introduce suavidad **no solo en la curva ajustada**, sino también en la **estructura del propio suavizado**, lo que evita sobreajuste local y garantiza una transición suave entre zonas del dominio con distinta complejidad.

Finalmente, al sustituir λ en la penalización adaptativa:

$$P = \theta^\top D^\top \mathrm{diag}(\Psi\xi) D\theta$$

Esto permite que el nivel de suavizado **varíe a lo largo del dominio**, adaptándose automáticamente a regiones más planas o con mayor variación.

Cuando se introduce suavizado adaptativo, el modelo penalizado ya no tiene una única penalización sencilla, sino que pasa a tener una penalización heterogénea. Esta penalización ya no es escalar ni fija, sino que depende de parámetros que también hay que estimar. Por eso se plantea reparametrizarse todo como un modelo mixto.

La estimación conjunta de los coeficientes θ y de los parámetros de suavizado ξ se realiza mediante técnicas de máxima verosimilitud restringida (REML).

Interpretación como modelo mixto

Se busca convertir esa penalización en una distribución normal sobre los efectos aleatorios:

$$P = \theta^\top D^\top \operatorname{diag}(\Psi\xi) D\theta$$

Esto da lugar a la formulación mixta del modelo:

$$\theta = X\beta + Z\alpha + \varepsilon, \quad \text{con } \alpha \sim \mathcal{N}(0, G), \quad \varepsilon \sim \mathcal{N}(0, \sigma_t^2 I), \quad \text{y } G^{-1} = \sum_{p=1}^{p} \sigma_p^{-2} \widetilde{\Lambda}_p$$

Los residuos ε representan la parte no explicada por el modelo (ruido) y deben seguir una distribución normal si los supuestos del modelo son válidos.

Construcción de las matrices X y Z

- La matriz D representa las diferencias finitas de orden q aplicadas a θ
- El **espacio nulo de** D (o núcleo) contiene las funciones no penalizadas: por ejemplo, constantes y términos lineales si $q = 2$
- Ese núcleo tiene dimensión q; se extraen q vectores ortogonales que forman la matriz X, correspondiente a la parte fija del modelo
- La matriz Z se construye como:

$$Z = BF, \quad \text{donde } F = D^\top (DD^\top)^{-1}$$

Esto proyecta las diferencias penalizadas en una base asociada a la curvatura.

Las matrices X y Z son ortogonales por construcción, lo que permite una separación clara entre las partes no penalizadas y penalizadas del modelo.

Estimación de la matriz de precisión G^{-1}

En el modelo mixto adaptativo, los coeficientes penalizados se consideran efectos aleatorios:

$$\alpha \sim \mathcal{N}(0, G)$$

La penalización se reexpresa como una distribución normal con matriz de precisión G^{-1}, que se define como:

$$G^{-1} = \frac{1}{\phi} F^\top P_{\text{Ad}} F = \sum_{p=1}^{p} \sigma_p^{-2} \widetilde{\Lambda}_p$$

donde:

- σ_t: parámetro global de varianza
- $\sigma_p^2 = \sigma_t/\xi_p$: varianza asociada al parámetro de suavizado local
- ξ_p: **parámetros a estimar** que determinan el grado de suavizado en distintas zonas del dominio
- $\widetilde{\Lambda}_p = \text{diag}(\psi_p)$: matrices conocidas que definen la estructura local de cada penalización. Se trata de una matriz diagonal con dimensiones $(c-q)\times(c-q)$, se trata de la función bspline p en formato matriz diagonal.
- $F = D_q^\top (D_q D_q^\top)^{-1}$: matriz de proyección de las diferencias penalizadas

Cada $\widetilde{\Lambda}_p$ define dónde actúa el parámetro ξ_p, y la estimación se centra en obtener los valores óptimos de estos ξ_p, que controlan la penalización adaptativa en cada región. La matriz G^{-1} total se puede ver como la suma de p matrices diagonales con dimensiones $(c-q)\times(c-q)$ en las que cada una de las matrices explica como actúa cada ξ_p en cada región de la función p.

La forma lineal de G^{-1} en función de los σ_l^{-2} permite aplicar el método SOP (Separation of Overlapping Penalties), lo que hace posible estimar los parámetros de suavizado locales de manera eficiente mediante **máxima verosimilitud restringida (REML)**. Esta función se encuentra en el paquete SOP y tiene como argumentos de entrada el número de nodos de las funciones básicas originales para hacer el suavizado ($nseg$) y el número de nodos de las funciones básicas para estimar la curva λ ($nseg.sp$). Con la notación explicada corresponde a $nseg$ = c - $ndegree$, siendo $ndegree$ el grado de los polinomios de las funciones básicas (en estos contextos siempre 3) y $nseg.sp$ = p - $ndegree$.

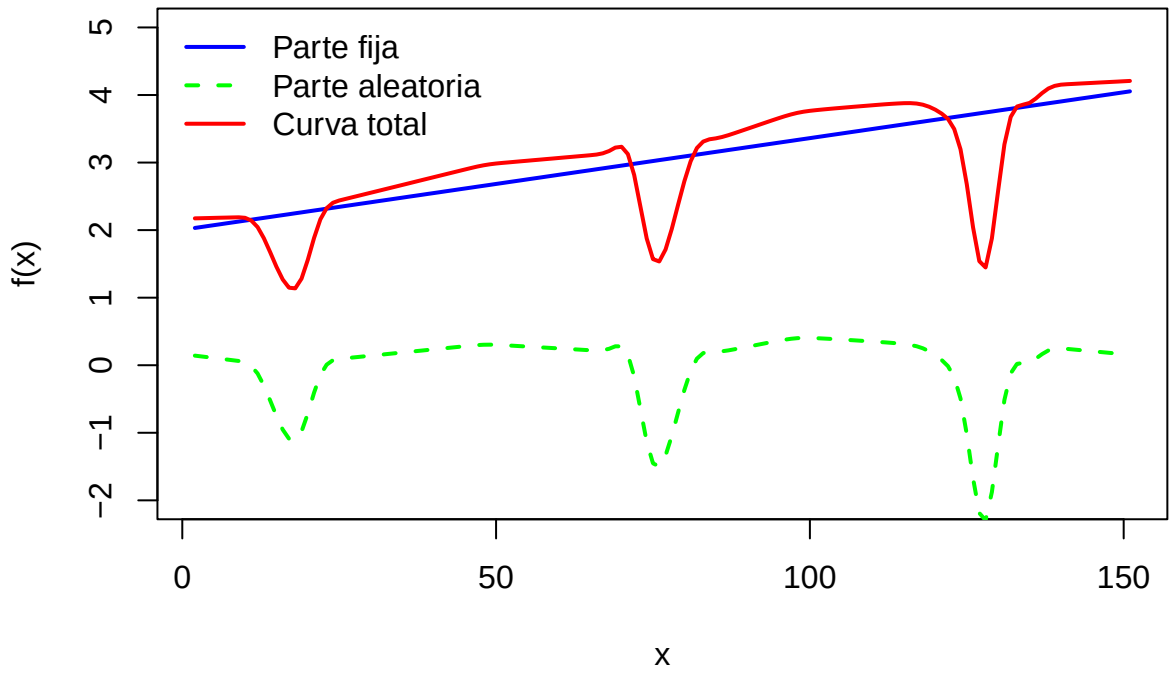
La definición de estos parámetros definirá la calidad del suavizado adaptativo. En situaciones estándar, el número de funciones base en un suavizado por P-splines suele determinarse siguiendo la conocida regla de Rupert, que recomienda utilizar aproximadamente una base B-spline por cada 4 o 5 puntos de observación. En el caso de las curvas analizadas, compuestas por 151 observaciones, esta regla sugiere emplear entre 30 y 40 funciones base. No obstante, tras realizar diversas pruebas empíricas y observar la presencia de picos marcados y variabilidad local en ciertas regiones del dominio, se optó por incrementar ligeramente la resolución del modelo. Concretamente, se ha fijado $nseg = 45$, lo que proporciona 48 funciones base cúbicas (orden 3), permitiendo capturar mejor los detalles sin comprometer la estabilidad del ajuste gracias a la penalización. Para la estimación de la función de suavizado adaptativo $\lambda(x)$, se ha empleado $nseg.sp = 35$, valor que permite que la penalización varíe suavemente a lo largo del dominio, adaptándose a las distintas regiones de curvatura de las curvas observadas.

El suavizado adaptativo ajusta localmente la suavidad de la curva y evita sobreajuste en regiones planas y permite flexibilidad donde hay cambios.

```r
# Crear matriz para guardar las curvas suavizadas
Númerodenodos=45
suavizadas <- matrix(NA, nrow = nrow(data_t), ncol = ncol(data_t))
funccoefs <- matrix(NA, nrow = Númerodenodos+3, ncol = ncol(data_t))
funcbasis <- vector("list", ncol(data_t))

# Hacer suavizado adaptativo por cada curva
for (i in 1:ncol(data_t)) {
  df <- data.frame(x = argvals, y = data_t[, i])
  fit <- sop(y ~ ad(x, nseg = Númerodenodos, nseg.sp = 35), data = df,
             control = list(trace = FALSE, epsilon = 1e-4))
  funccoefs[,i]=c(fit$b.fixed,fit$b.random)
  funcbasis[[i]] <- cbind(fit$X, fit$Z)  # Guarda la matriz completa por curva
  suavizadas[,i] <- fitted(fit)
}

# Parte fija
tendencia_general <- fit$X %*% fit$b.fixed
```

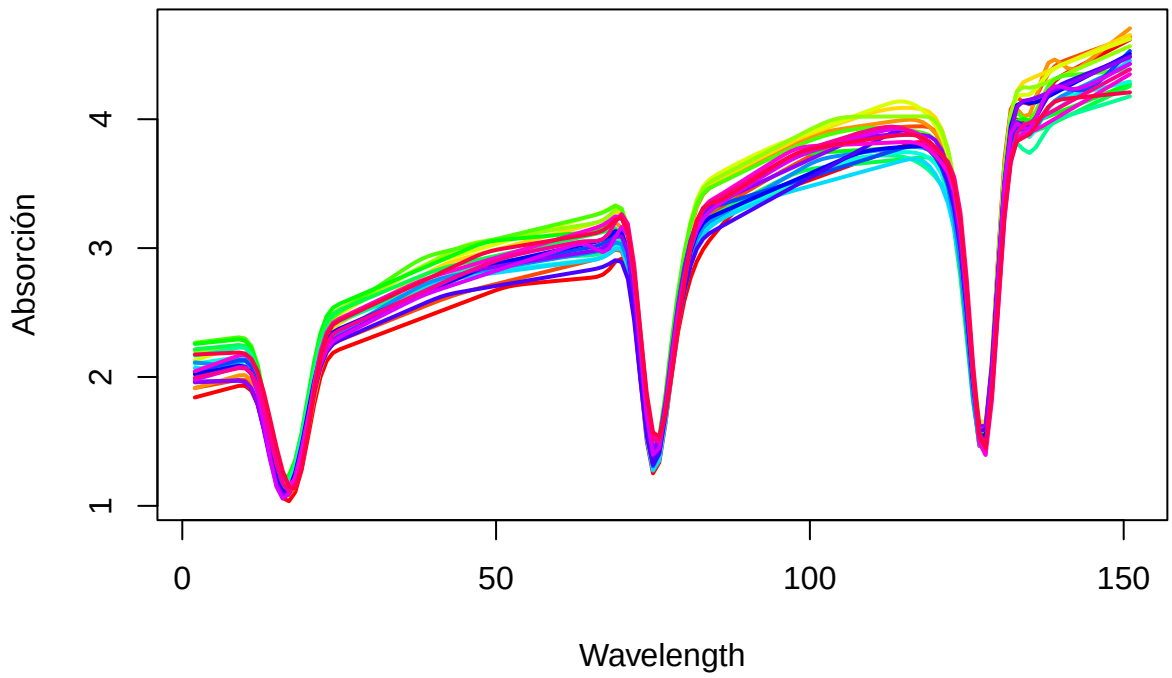
```r
# Parte aleatoria (curvatura adaptativa)
ondulacion <- fit$Z %*% fit$b.random

# Curva total
curva_total <- tendencia_general + ondulacion


# Graficar todo junto
plot(argvals, tendencia_general, type = "l", col = "blue", lwd = 2,
     ylab = "f(x)", xlab = "x", ylim = c(-2,5), main="Figura 3: Curvas expresadas como modelo mixto")
lines(argvals, ondulacion, col = "green", lwd = 2, lty = 2)
lines(argvals, curva_total, col = "red", lwd = 2)

legend("topleft", legend = c("Parte fija ", "Parte aleatoria ", "Curva total"),
       col = c("blue", "green", "red"), lty = c(1, 2, 1), lwd = 2, bty = "n")
```

Figura 3: Curvas expresadas como modelo mixto




```r
# Graficar resultados suavizados
matplot(argvals, suavizadas, type = "l", lty = 1, lwd = 2,
        col = rainbow(ncol(suavizadas)), xlab = "Wavelength", ylab = "Absorción",
        main = "Figura 4: Curvas suavizadas adaptativamente (SOP)")
```

Figura 4: Curvas suavizadas adaptativamente (SOP)



Las curvas suavizadas obtenidas mediante el modelo mixto adaptativo con el método SOP muestran una calidad significativamente superior respecto al ajuste clásico con P-splines. En la Figura 3 se representa la descomposición del modelo en sus dos componentes: la parte fija $X\beta$, que capta la tendencia global, y la parte aleatoria $Z\alpha$, que introduce flexibilidad local. Esta separación permite entender cómo el suavizado adaptativo actúa localmente sobre las curvaturas. En la Figura 4 se visualizan las curvas finales suavizadas, donde se observa un ajuste suave pero preciso, que se adapta correctamente a los cambios de forma sin sobreajustar en regiones planas.

Análisis de componentes principales funcional (FPCA)

Se aplica un análisis de componentes principales funcional con el objetivo de identificar los principales **modos de variación** entre las curvas suavizadas. Este análisis permite representar la mayoría de la variabilidad funcional en un espacio de dimensión reducida, facilitando la interpretación y el análisis posterior.

Las funciones base empleadas por el modelo SOP no son ortonormales, por lo que los coeficientes funcionales estimados no se pueden analizar directamente mediante un PCA clásico. Para corregir esto, es necesario aplicar una transformación basada en la matriz de productos interiores entre funciones base, tal como se ha explicado en clase.

Transformación para ortonormalizar

Si A es la matriz de coeficientes y Ψ la matriz de productos interiores entre funciones base ($\Psi_{ij} = \langle \phi_i, \phi_j \rangle$), se realiza la transformación:

$$\tilde{A} = A \cdot \Psi^{1/2}$$

Esta nueva matriz $\tilde{A}$ representa los coeficientes en una base ortonormalizada. A partir de aquí, se puede aplicar un PCA clásico de forma válida.

Representación funcional de las componentes principales

Una vez realizado el PCA clásico sobre la matriz ortonormalizada $\tilde{A}$, las componentes principales se expresan respecto a esta base transformada. Para poder interpretarlas correctamente en el espacio funcional original, es necesario **invertir la transformación** aplicada.

Transformación inversa

Si V^* son los autovectores obtenidos tras el PCA sobre $\tilde{A}$, su representación en la base original se obtiene mediante:

$$W = \Psi^{-1/2} \cdot V^*$$

donde:

- $\Psi^{-1/2}$ es la inversa de la raíz cuadrada matricial de Ψ
- Cada columna de W representa una componente principal funcional en la base original

---

Evaluación funcional

Para visualizar las componentes principales como funciones reales sobre el dominio, se combinan con las funciones base evaluadas en la malla x:

$$\mathrm{PC}_k(x) = \sum_{j=1}^{c} W_{jk} \cdot \phi_j(x)$$

o bien, en notación matricial:

$$\mathrm{PC}_k(x) = \texttt{phi\_eval} \cdot W_{:,k}$$

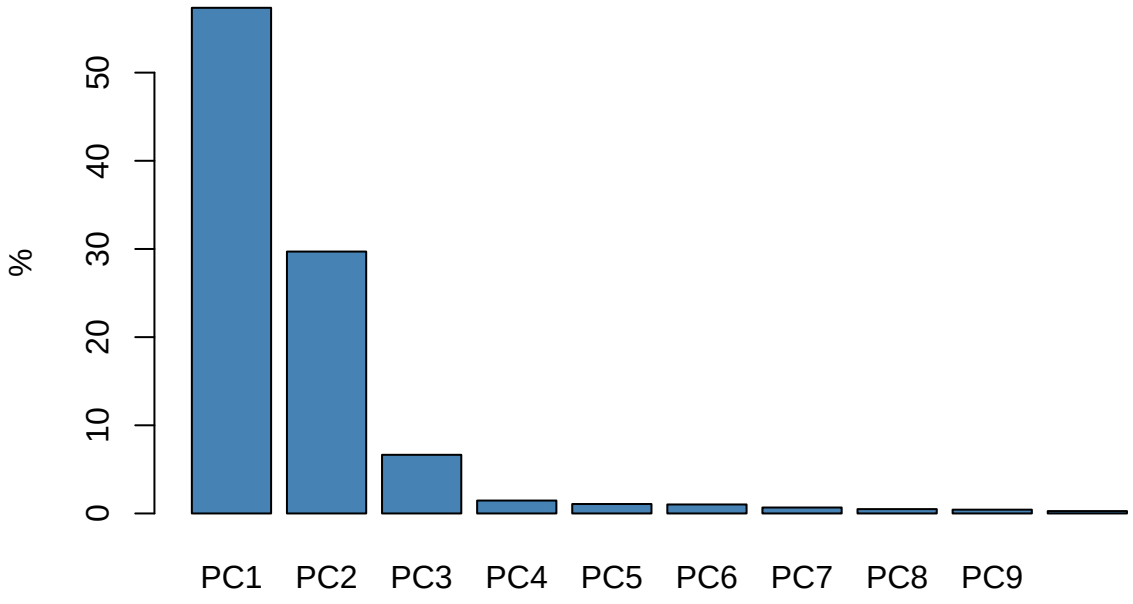
Esto permite representar correctamente los modos principales de variación funcional y analizarlos gráficamente.

Interpretación:

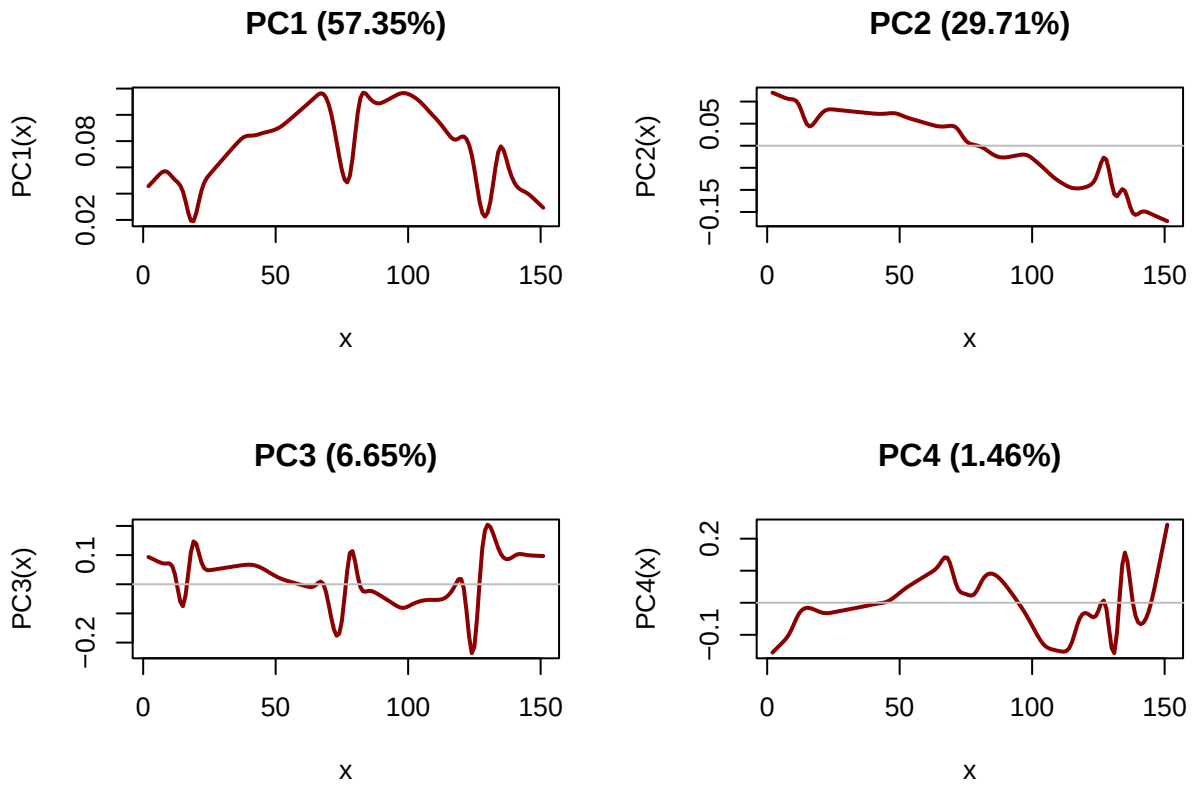
El PCA sobre $\tilde{A}$ permite obtener componentes principales funcionales que explican la mayor parte de la variabilidad de las curvas originales. Las componentes pueden reconstruirse y representarse gráficamente multiplicando los autovectores transformados por la base funcional evaluada. Este procedimiento garantiza que el análisis respete la métrica funcional adecuada y no dependa de la elección de una base no ortogonal.

```r
# 1. Obtener número de funciones base
nbasis <- nrow(funccoefs)

phi_eval <- funcbasis[[1]]

# Aproximamos los productos internos como sumas (trapezoidal)
dx <- diff(range(argvals)) / length(argvals)  # paso del eje x
PSI <- t(phi_eval) %*% phi_eval * dx  #

# 3. Calcular raíz cuadrada matricial de psi
eigen_psi <- eigen(PSI)
sqrt_PSI <- eigen_psi$vectors %*% diag(sqrt(eigen_psi$values)) %*% t(eigen_psi$vectors)

# 4. Transformar los coeficientes
A <- t(funccoefs)  # n_curves x nbasis
A_transf <- A %*% sqrt_PSI

# 5. Hacer PCA sobre los coeficientes transformados
fpca_metric <- prcomp(A_transf, center = TRUE, scale. = FALSE)
scores_2PC <- fpca_metric$x[, 1:3]

# 6. Mostrar % de varianza explicada
var_expl <- 100 * fpca_metric$sdev^2 / sum(fpca_metric$sdev^2)
barplot(var_expl[1:10], main = "Figura 5: % varianza explicada",
        ylab = "%", names.arg = paste0("PC", 1:10), col = "steelblue")
```

Figura 5: % varianza explicada



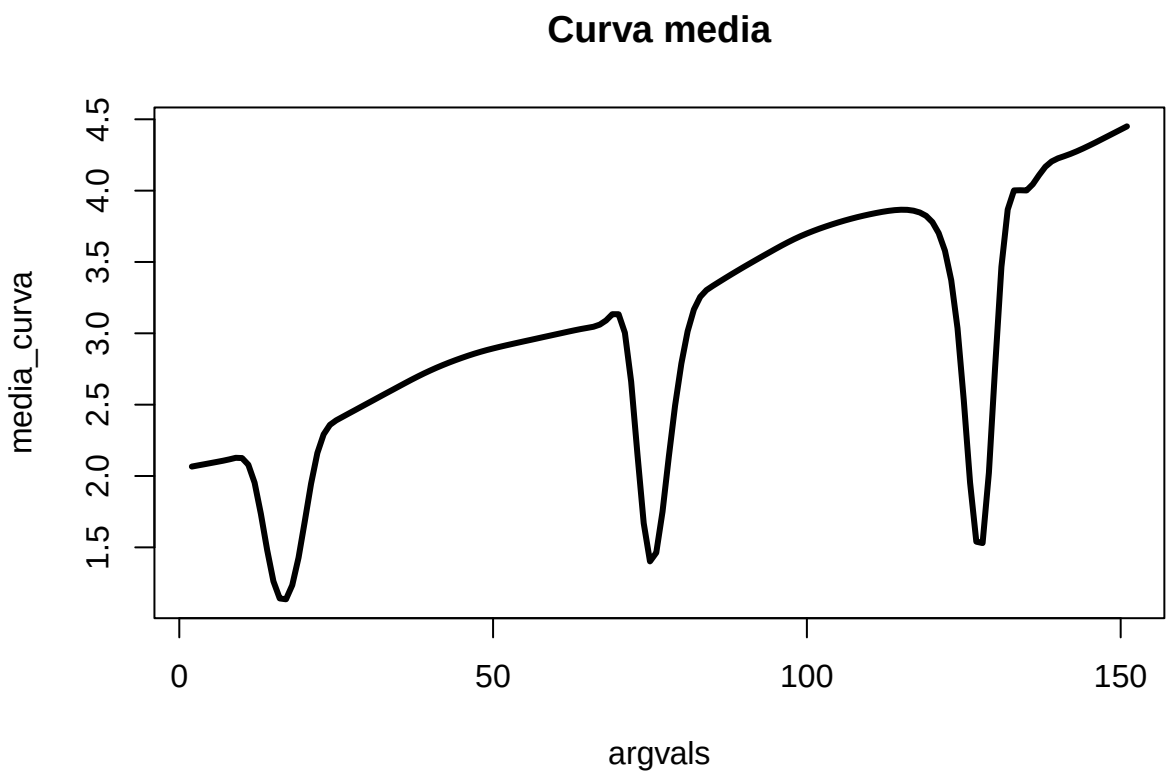
```r
# Volvemos a la base original con: v_k_original = psi^{-1/2} %*% PC_k_transf
inv_sqrt_PSI <- eigen_psi$vectors %*% diag(1 / sqrt(eigen_psi$values)) %*% t(eigen_psi$vectors)

# Base evaluada (funcbasis) - 151 x 44
phi_eval <- funcbasis[[1]]

# Graficamos las primeras 4 componentes funcionales corregidas
par(mfrow = c(2, 2))
for (i in 1:4) {
  pc_transf <- fpca_metric$rotation[, i]
  pc_original <- inv_sqrt_PSI %*% pc_transf  # volvemos al espacio original
  curva_pc <- phi_eval %*% pc_original       # evaluamos la función

  plot(argvals, curva_pc, type = "l", lwd = 2, col = "darkred",
       main = paste0("PC", i, " (", round(var_expl[i], 2), "%)"),
       xlab = "x", ylab = paste0("PC", i, "(x)"))
  abline(h = 0, col = "gray")}
```



Interpretación de la FPCA sobre curvas suavizadas

Tras aplicar el FPCA sobre las curvas suavizadas adaptativamente (Figura 4), obtenemos en la figura superior las **cuatro primeras componentes principales funcionales (PCs)**. Estas componentes representan **modos de variación típicos** respecto a la media.

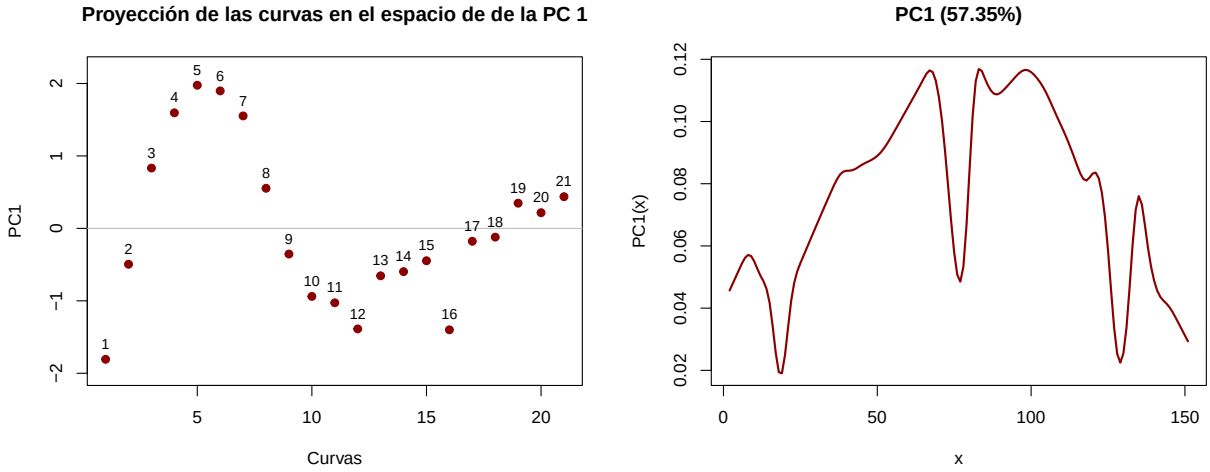
```r
# Evaluar la media funcional en los puntos de la malla
center_orig <- inv_sqrt_PSI %*% fpca_metric$center  # media funcional en espacio original
media_curva <- phi_eval %*% center_orig

# Graficar la curva media
plot(argvals, media_curva, type = "l", lwd = 3, col = "black",
     main = "Curva media")
```



- **PC1 (56.22%)**: Captura la mayor parte de la variabilidad. Representa una oscilación general en las regiones centrales de la curva (alrededor de los 3 picos de absorción, relacionada posiblemente con diferencias de amplitud o absorción media entre curvas.

```r
par(mfrow = c(1, 2))
plot( scores_2PC[,1],
     xlab = "Curvas", ylab = "PC1",
     main = "Proyección de las curvas en el espacio de de la PC 1",
     pch = 19, col = "darkred", ylim = c(-2,2.2))
abline(h = 0, v = 0, col = "gray")
text( scores_2PC[,1], labels = 1:nrow(scores_2PC), pos = 3, cex = 0.8)

pc_transf1 <- fpca_metric$rotation[, 1] #loadings primera componente principal
pc_original1 <- inv_sqrt_PSI %*% pc_transf1  # volvemos al espacio original
curva_pc1 <- phi_eval %*% pc_original1       # evaluamos la función

plot(argvals, curva_pc1, type = "l", lwd = 2, col = "darkred",
```

```
     main = paste0("PC", 1, " (", round(var_expl[1], 2), "%)"),
     xlab = "x", ylab = paste0("PC", 1, "(x)"))
abline(h = 0, col = "gray")
```

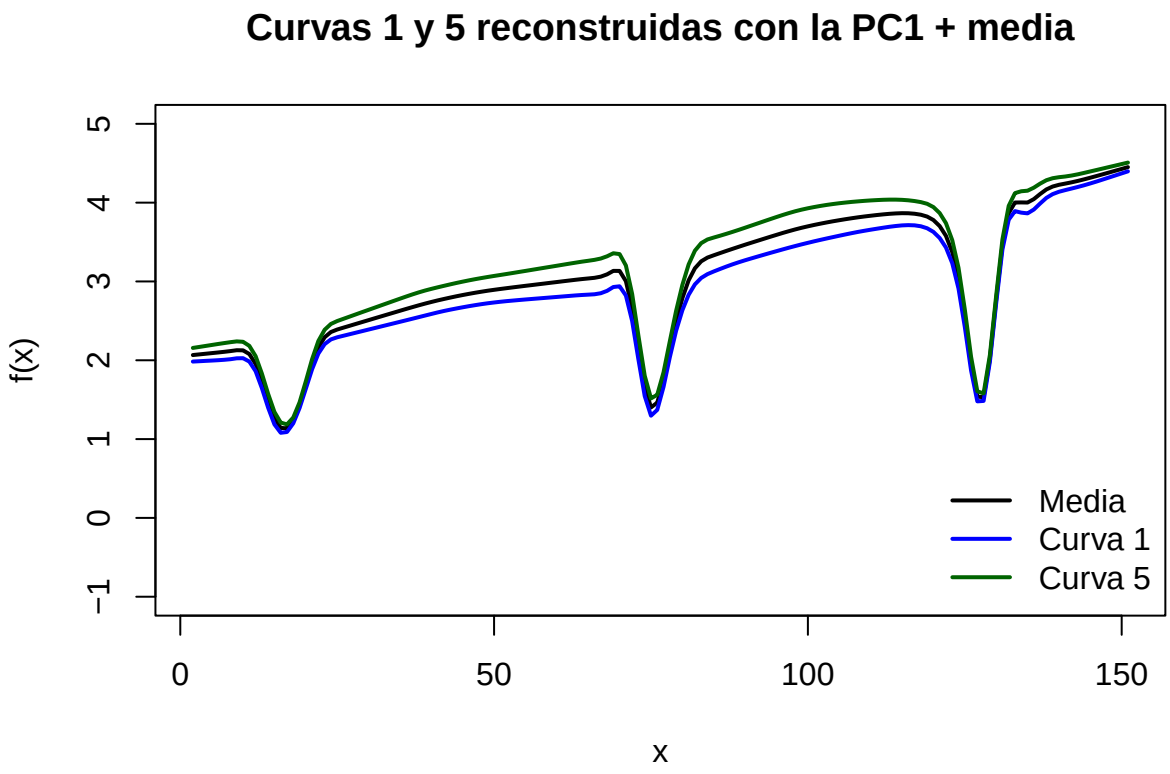


Para observar bien la naturaleza de estas componentes se reconstruyen las curvas 1 y 5 con la contribucion de la primera componente principal más la media. Esto es así debido a que estas dos curvas presentan scores en esta componente similares en pero con signo contrario. De esta forma se aprecia claramente la naturaleza de esta primera componente principal.

La primera componente principal presenta un perfil arqueado que alcanza sus valores máximos en la zona central del dominio. Esto sugiere que representa variaciones generales en la intensidad espectral, probablemente asociadas a momentos del día con mayor irradiación solar, como las horas centrales. Las curvas con proyección positiva en esta componente tenderían a tener mayor energía absorbida en todo el rango espectral.

```
# Curvas a reconstruir
i1 <- 1
i2 <- 5

scores_PC1 <- fpca_metric$x[, 1]

center_orig <- inv_sqrt_PSI %*% fpca_metric$center  # media funcional en espacio original
A_hat <- sweep(scores_PC1 %*% t(pc_original1), 2, center_orig, "+")

# Media funcional evaluada
media_curva <- phi_eval %*% center_orig  # vector 151 × 1

# Componentes principales en base original
PC1_func <- phi_eval %*% pc_original1[, 1]  # componente 1 evaluada en malla

# Reconstrucción con media + componente 1 (solo)
recon_i1 <- media_curva + scores_PC1[i1] * PC1_func
recon_i2 <- media_curva + scores_PC1[i2] * PC1_func

# Graficar
matplot(argvals, cbind(media_curva, recon_i1, recon_i2), type = "l", lty = 1, lwd = 2,
        col = c("black", "blue", "darkgreen"), ylim = c(-1, 5),
        main = "Curvas 1 y 5 reconstruidas con la PC1 + media", xlab = "x", ylab = "f(x)")
```

```r
legend("bottomright", legend = c("Media", "Curva 1", "Curva 5"),
       col = c("black", "blue", "darkgreen"), lwd = 2, lty = 1, bty = "n")
```

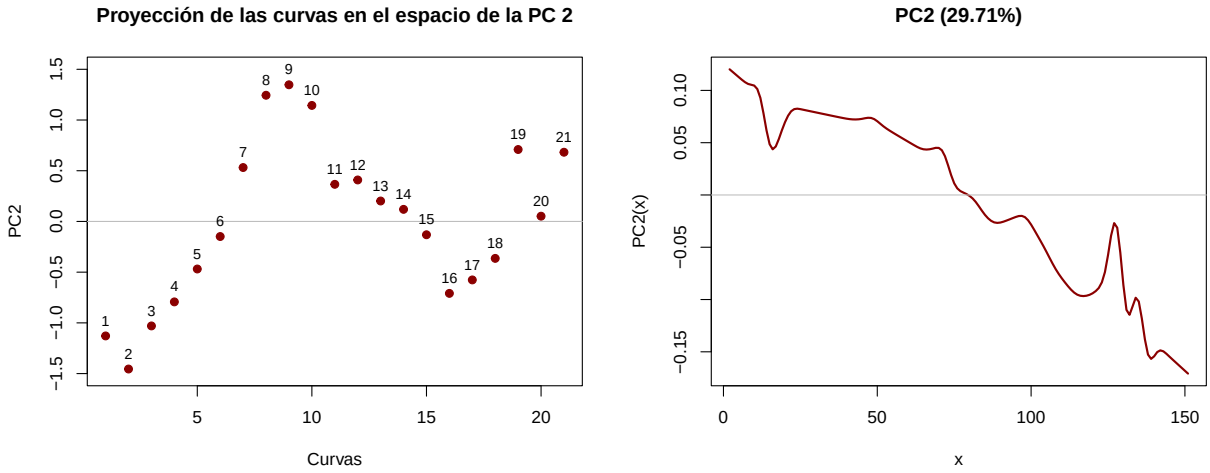


Se observa como la curva 5 se encuentra por encima de la media, al contrario de la curva 1 que está por debajo, así como cabe esperar atendiendo a sus scores. En los tramos cerca del segundo pico de absorción se aprecia como se alejan más de le media, respondiendo a la naturaleza creciente y decreciente a sendos lados del pico de absorción de la primera componente principal. Los picos de absorción en ambas curvas se encuentran superpuestos, comportamiento que se ve reflejado también en el diagrama de la componente principal. Esto refleja una variabilidad menor entre las curvas en las regiones de absorción en comparación con el resto de regiones.

- **PC2 (29.05%)**: Asocia una tendencia decreciente con oscilaciones suaves, lo que sugiere que explica diferencias más globales en la pendiente o nivel general de las curvas.

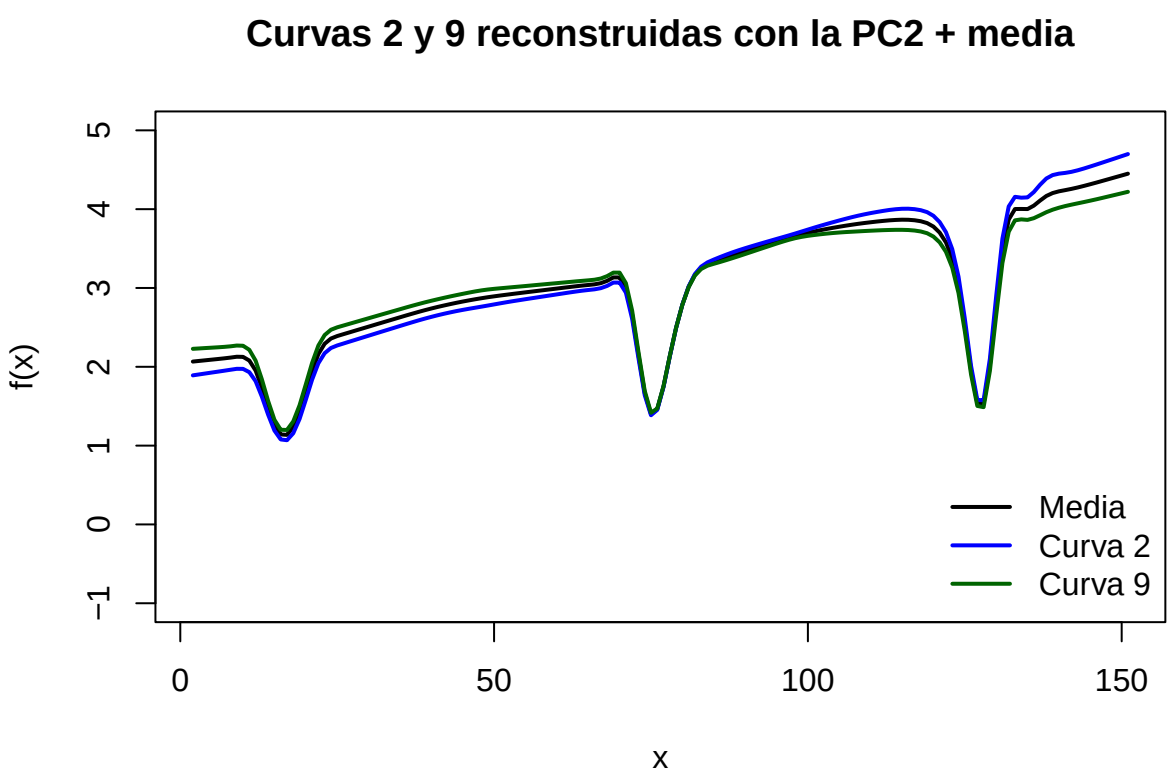
```r
par(mfrow = c(1, 2))
plot( scores_2PC[,2],
     xlab = "Curvas", ylab = "PC2",
     main = "Proyección de las curvas en el espacio de la PC 2",
     pch = 19, col = "darkred", ylim = c(-1.5,1.5))
abline(h = 0, v = 0, col = "gray")
text( scores_2PC[,2], labels = 1:nrow(scores_2PC), pos = 3, cex = 0.8)

pc_transf2 <- fpca_metric$rotation[, 2]
pc_original2 <- inv_sqrt_PSI %*% pc_transf2  # volvemos al espacio original
curva_pc2 <- phi_eval %*% pc_original2       # evaluamos la función
```

```r
plot(argvals, curva_pc2, type = "l", lwd = 2, col = "darkred",
     main = paste0("PC", 2, " (", round(var_expl[2], 2), "%)"),
     xlab = "x", ylab = paste0("PC", 2, "(x)"))
abline(h = 0, col = "gray")
```



Siguiendo el procedimiento anterior, se reconstruyen las curvas 2 y 9 con la contribucion de la segunda componente principal más la media.

```r
# Curvas a reconstruir
i1 <- 2
i2 <- 9

scores_PC2 <- fpca_metric$x[, 2]

center_orig <- inv_sqrt_PSI %*% fpca_metric$center  # media funcional en espacio original
A_hat <- sweep(scores_PC2 %*% t(pc_original2), 2, center_orig, "+")

# Media funcional evaluada
media_curva <- phi_eval %*% center_orig  # vector 151 × 1

# Componentes principales en base original
PC2_func <- phi_eval %*% pc_original2[, 1]  # componente 1 evaluada en malla

# Reconstrucción con media + componente 1 (solo)
recon_i1 <- media_curva + scores_PC2[i1] * PC2_func
recon_i2 <- media_curva + scores_PC2[i2] * PC2_func


# Graficar
matplot(argvals, cbind(media_curva, recon_i1, recon_i2), type = "l", lty = 1, lwd = 2,
        col = c("black", "blue", "darkgreen"), ylim = c(-1, 5),
        main = "Curvas 2 y 9 reconstruidas con la PC2 + media", xlab = "x", ylab = "f(x)")

legend("bottomright", legend = c("Media", "Curva 2", "Curva 9"),
       col = c("black", "blue", "darkgreen"), lwd = 2, lty = 1, bty = "n")
```



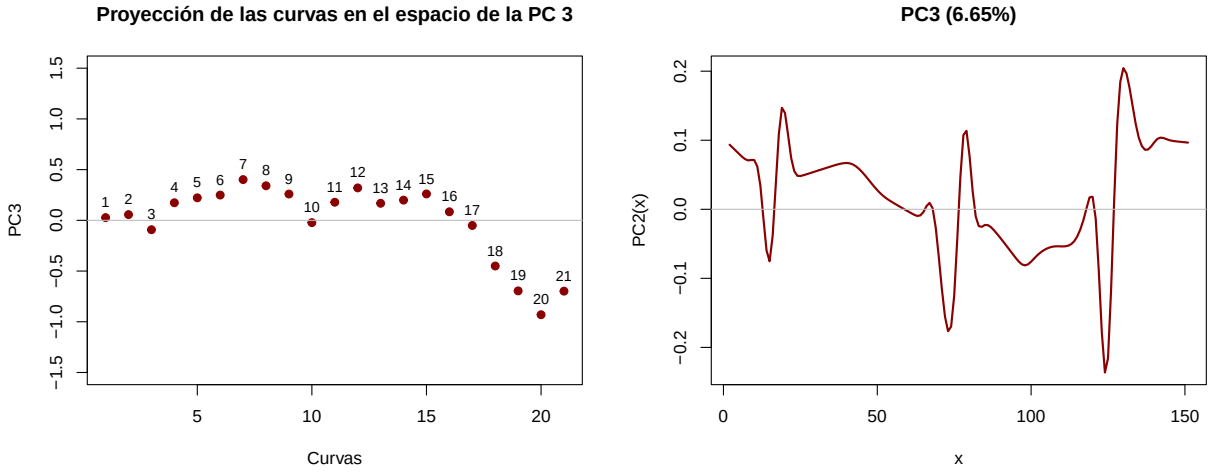
La curva 9 se observa como empieza por encima de la media y acaba por debajo con un cambio decreciente lineal. En las regiones centrales de la curva (alrededor del segundo pico de absorción), se puede apreciar como se superpone prácticamente con la media. Esto cabe esperar atendiendo al diagrama de la segunda componente principal, donde esta es positiva en los primeros tramos de la curva, cercano a 0 en los tramos medios y negativa en los tramos finales. Similarmente a la primera componente, en las zonas que corresponden a los picos de absorción la componente tiende a 0, lo que refleja la variabilidad más reducida de las curvas entre ellas en esas regiones en comparación con el resto. El comportamiento de la curva 2 es igual al de la curva anterior pero con signo inverso, tal y como cabe esperar por la naturaleza de los scores de las mismas.

La segunda componente principal refleja una variación compensada entre el inicio y el final del espectro, mostrando una tendencia decreciente. Este comportamiento sugiere que la PC2 representa cambios en la distribución relativa de energía espectral, posiblemente asociados a condiciones atmosféricas particulares o a momentos del día con ángulos solares más extremos, como el amanecer o el atardecer. Las curvas con puntuaciones positivas en esta componente tienden a tener mayor absorción en las longitudes de onda cortas y menor en las largas, y viceversa.

- **PC3 (8.17%)**: Explica pequeñas variaciones hacia abajo (al entrar en el pico de absorción) y hacia arriba (al salir de él) respecto de la media de las curvas.

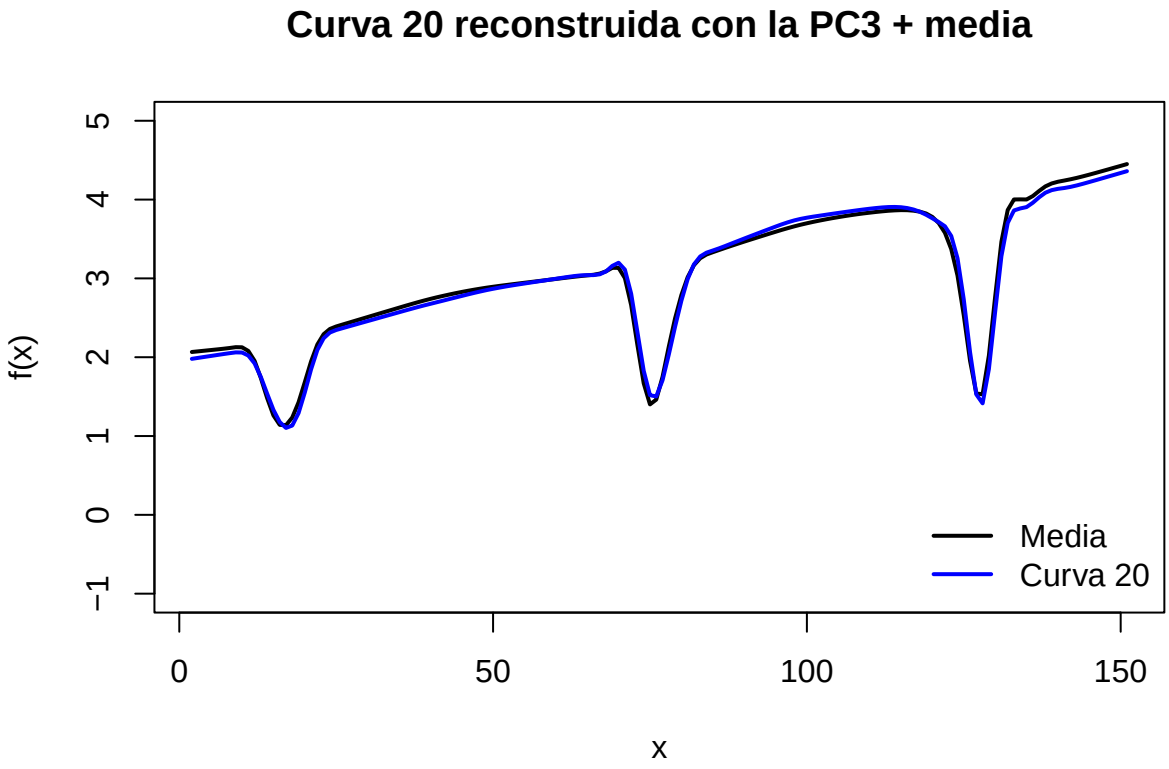
```
par(mfrow = c(1, 2))
plot( scores_2PC[,3],
     xlab = "Curvas", ylab = "PC3",
     main = "Proyección de las curvas en el espacio de la PC 3",
     pch = 19, col = "darkred", ylim = c(-1.5,1.5))
abline(h = 0, v = 0, col = "gray")
text( scores_2PC[,3], labels = 1:nrow(scores_2PC), pos = 3, cex = 0.8)
```

```r
pc_transf3 <- fpca_metric$rotation[, 3]
pc_original3 <- inv_sqrt_PSI %*% pc_transf3  # volvemos al espacio original
curva_pc3 <- phi_eval %*% pc_original3       # evaluamos la función

plot(argvals, curva_pc3, type = "l", lwd = 2, col = "darkred",
     main = paste0("PC", 3, " (", round(var_expl[3], 2), "%)"),
     xlab = "x", ylab = paste0("PC", 2, "(x)"))
abline(h = 0, col = "gray")
```



Se procede a reconstruir la curva 20 con la contribución de la componente 3 más la media debido a que esta curva presenta el score (en módulo) más elevado en esta componente.

```r
# Curva a reconstruir
i1 <- 20

scores_PC3 <- fpca_metric$x[, 3]

center_orig <- inv_sqrt_PSI %*% fpca_metric$center  # media funcional en espacio original
A_hat <- sweep(scores_PC3 %*% t(pc_original3), 2, center_orig, "+")

# Media funcional evaluada
media_curva <- phi_eval %*% center_orig  # vector 151 × 1

# Componentes principales en base original
PC3_func <- phi_eval %*% pc_original3[, 1]  # componente 1 evaluada en malla

# Reconstrucción con media + componente 1 (solo)
recon_i1 <- media_curva + scores_PC3[i1] * PC3_func


# Graficar
matplot(argvals, cbind(media_curva, recon_i1), type = "l", lty = 1, lwd = 2,
        col = c("black", "blue"), ylim = c(-1, 5),
        main = "Curva 20 reconstruida con la PC3 + media", xlab = "x", ylab = "f(x)")

legend("bottomright", legend = c("Media", "Curva 20"),
       col = c("black", "blue"), lwd = 2, lty = 1, bty = "n")
```



La tercera componente principal recoge una pequeña proporción de la variabilidad total (8.17%) y su forma indica que está asociada principalmente a las transiciones en las zonas de cambio abrupto del espectro, especialmente en los alrededores de los picos más pronunciados. Estas regiones son críticas, ya que es donde se concentra la mayor variación entre curvas. Por tanto, esta componente captura pequeñas diferencias en la forma o desplazamiento de los picos, lo que puede deberse a condiciones de medida específicas o a diferencias sutiles entre muestras. La contribución se puede observar alrededor del tercer pico de absorción de la curva 20, donde en los instantes anteriores se encuentra ligeramente por encima y ligeramente por debajo ya pasado el pico.

Conclusión:

Las dos primeras componentes explican más del 85% de la variabilidad funcional, lo cual indica que la mayoría de las diferencias entre curvas suavizadas pueden representarse en un espacio de dimensión muy reducida. Esto es útil para clasificación, reducción de dimensionalidad o modelado posterior.