Enero 2020

Cristina Gil Martínez

https://rpubs.com/Cristina_Gil/RSM

(cristina_gil_m@hotmail.com)

METODOLOGÍA DE SUPERFICIE DE RESPUESTA (RSM)

Explicacion de conceptos teóricos con ejemplos en R

CONTENIDO

INTRODUCCIÓN

La **metodología de superficie de respuesta** (RSM) es una colección de técnicas matemáticas y estadísticas empleadas para desarrollar, mejorar y optimizar procesos. También es aplicable en el diseño, desarrollo y formulación de nuevos productos, así como la mejora de diseños de productos ya implantados.

Una de las aplicaciones más ampliamente extendidas de estas técnicas es la de modelar y analizar problemas en los cuales una respuesta de interés (pudiendo ser más de una) viene influenciada por varios factores de carácter cuantitativo ($\xi_1, \xi_2, \ldots, \xi_k$), siendo el objetivo el de optimizar dicha respuesta determinando los valores óptimos de los factores implicados. La relación vendrá dada por:

$$y = f(\xi_1, \xi_2, \ldots, \xi_k) + \epsilon$$

que se supone continua en ξ_i, $\forall$ i = 1, ..., k, donde ϵ representa el ruido o error observado en la respuesta, cuya distribución se asume normal con media cero.

Las variables $\xi_1, \xi_2, \ldots, \xi_k$ en la ecuación anterior son **variables naturales**, ya que se expresan en las unidades naturales de medida. Sin embargo, es común transformarlas a **variables codificadas** ($x_1, x_2, \ldots, x_k$): sin dimensiones, con media cero y la misma desviación estándar. Así pues, el valor real esperado que tome la variable respuesta

$$\eta = f(x_1, x_2, \ldots, x_k)$$

supone una relación que se puede representar mediante una hipersuperficie denominada **superficie de respuesta**.

En la realidad, la forma de la verdadera función respuesta f es desconocida, por lo que debemos aproximarla. La elección de los factores apropiados es, por tanto, importante. El éxito de aplicar la técnica MSR radica también en que la respuesta pueda ajustarse a un polinomio de primer o segundo grado.

Nota: Es poco probable que un modelo polinómico sea una aproximación razonable a lo largo de todo el espacio de los predictores, pero para regiones relativamente delimitadas estos modelos pueden ser adecuados.

El modelo de primer orden vendrá dado por

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \epsilon$$

pudiendo incluir también términos de interacción $\beta_{12} x_1 x_2$, los cuales introducen algo de curvatura en la función. Contando con $N \geq 3$ podremos obtener mediante mínimos cuadrados los estimadores de los coeficientes de regresión $b_0, b_1, b_2, \ldots, b_k$, además de la variación del error experimental. El análisis de superficie de respuesta se llevará a cabo, por tanto, con la

función o superficie ajustada obtenida. En la figura 1 puede verse un ejemplo de una superficie en tres dimensiones obtenida a partir de una función polinómica de primer grado, en la que se representa la evolución de la respuesta y en función de dos factores x_1 y x_2. El correspondiente gráfico de contornos ayuda a visualizar la forma de esta relación, donde cada línea de contorno representa una altura determinada de la superficie de respuesta, o lo que es lo mismo, distintas combinaciones de niveles de factores que provocan una misma respuesta. En este caso, el gráfico de contornos asociado está formado por una serie de líneas paralelas entre sí.

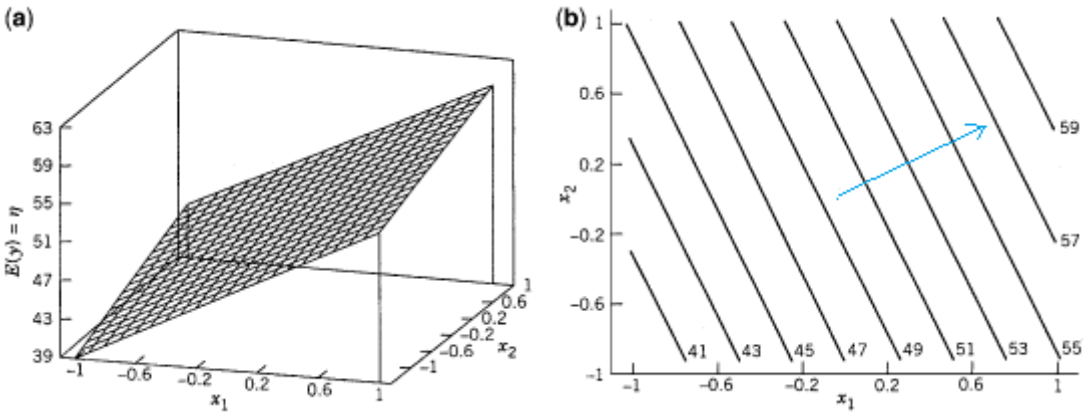


Figura 1. Superficie de respuesta (a) y gráfico de contorno (b) de un modelo de primer orden.

RESUMEN DE LOS PASOS A SEGUIR:

1. Definir los objetivos de la optimización: plantear el problema a resolver y seleccionar la variable respuesta a evaluar.
2. Escoger factores de interés y la región de operabilidad en función de la información o conocimiento sobre el proceso a estudiar y las posibilidades instrumentales.
3. Elaborar un diseño experimental de superficie de respuesta, llevarlo a cabo y obtener los datos.
4. Ajustar un modelo matemático de primer/segundo orden y evaluar si son adecuados.
5. En caso de que cierto modelo sea adecuado, localizar el óptimo buscado para la respuesta, utilizando herramientas gráficas y/o matemáticas.
6. Validar experimentalmente midiendo la respuesta utilizando los niveles óptimos de los factores obtenidos, y comprobar si se encuentra en consonancia con lo predicho por el modelo.

MÉTODO DE MÁXIMA PENDIENTE EN ASCENSO

De manera frecuente, la primera estimación de las condiciones de operación de un sistema o proceso estará alejada del óptimo real, en caso de tratarse de un proceso cuyo comportamiento es inicialmente desconocido. En este caso en el que nos encontremos alejados del óptimo, podemos asumir que un modelo de primer orden será una aproximación adecuada a la verdadera superficie de una pequeña región de los predictores.

La MSR es un **procedimiento secuencial**, podremos hacer uso del método de máxima pendiente en ascenso para movernos en la dirección de máximo incremento de la respuesta (la dirección en la que $\hat{y}$ aumenta más rápido), si el objetivo es maximizarla. Los incrementos en esta dirección serán proporcionales a los coeficientes $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$. Por ejemplo, en la figura 2, la dirección de máxima pendiente en ascenso sería, desde el centro de la región de interés ($x_1 = x_2 = 0$), perpendicular a las líneas de contorno y en sentido en que la variable respuesta aumenta. Esta dirección es paralela a la normal de la superficie ajustada. Tras cada aumento, se vuelven a ajustar modelos de primer orden y se determinan nuevas direcciones de ascenso en caso necesario. De la misma manera, puede seguirse el método de máxima pendiente en descenso si el objetivo es minimizar la respuesta. Este procedimiento secuencial se seguirá hasta que el aumento o disminución de la respuesta dejen de darse o sean muy pequeños. Cuando se llega a la cercanía del óptimo y existe falta de ajuste del modelo de primer orden, pasamos a ajustar un modelo polinómico de segundo orden.

ALGORITMO

Un algoritmo utilizado para determinar las coordenadas de la trayectoria de máxima pendiente en ascenso es el siguiente:

1. Se escoge un tamaño de incremento para una de las variables independientes Δx_j. Normalmente elegimos la variable con mayor coeficiente de regresión absoluto $|\beta_j|$ o que más conozcamos.

2. Aplicamos el incremento en el resto de variables:

$$\Delta x_i = \frac{\hat{\beta}_i}{\hat{\beta}_j / \Delta x_j}$$

   donde $i = 1, 2, \ldots, k$ $i \neq j$.

3. Convertimos Δx_i a unidades naturales.

Ejemplo:

Para mostrar un ejemplo sencido, supongamos que hemos tomado datos sobre un proceso que está afectado por el tiempo de reacción en minutos (ξ_1) y la temperatura en ºC (ξ_2). El rango sobre el que se ha trabajado para cada variable es ξ_1 = [20, 30] y ξ_2 = [140-150]. Para convertir estas variables naturales en variables codificadas aplicaríamos

$$x_1 = \frac{\xi_1 - [\max(\xi_1) + \min(\xi_1)]/2}{[\max(\xi_1) - \min(\xi_1)]/2} = \frac{\xi_2 - 25}{5}$$

$$x_2 = \frac{\xi_2 - [\max(\xi_2) + \min(\xi_2)]/2}{[\max(\xi_2) - \min(\xi_2)]/2} = \frac{\xi_2 - 145}{5}$$

De esta forma trabajamos en un rango [-1, 1] para ambas variables, siendo el valor central de 0. Por ejemplo, x_1 = 0 correspondería a ξ_1 = 25.
Imaginemos que hemos ajustado un modelo de primer grado cuya ecuación es la siguiente:

$$\hat{y} = 39{,}22 + 0{,}662x_1 + 0{,}214x_2$$

En el primer paso, tomando como referencia el punto central de nuestro diseño $x_1 = x_2 = 0$, en variables codificadas, nos moveríamos 0,662 unidades en la dirección de x_1 por cada 0,214 unidades de x_2, con una pendiente de 0,214/0,662 = 0,323. Elegimos la variable cuyo coeficiente absoluto es mayor, b_1 en este caso, para escoger el tamaño de incremento. Decidimos incrementar dicha variable 5 unidades de tiempo, equivalente en unidades codificadas a $\Delta x_1 = 1$. Por tanto, cada paso a lo largo de la dirección de máxima pendiente en ascenso aportará un incremento de:

$$\Delta x_1 = 1$$
$$\Delta x_2 = 0{,}323$$

correspondiente a

$$\Delta\xi_1 = \Delta x_1 \cdot 5 = 5 \text{ min}$$
$$\Delta\xi_2 = \Delta x_2 \cdot 5 = 1{,}615 \text{ ºC}$$

ANÁLISIS DE SUPERFICIE DE RESPUESTA DE SEGUNDO ORDEN

En la situación en la que nos encontremos relativamente próximos al óptimo, será necesario incorporar curvatura al modelo para aproximarlo a la respuesta a predecir. En la mayoría de los casos bastará con ajustar un polinomio de grado dos:

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum\sum_{i<j} \beta_{ij} x_i x_j + \epsilon$$

La figura 2 representa un ejemplo de una superficie de respuesta de un modelo polinómico de segundo orden, donde se representa la *productividad* esperada (variable respuesta) en función de dos factores: *temperatura* y *presión*.

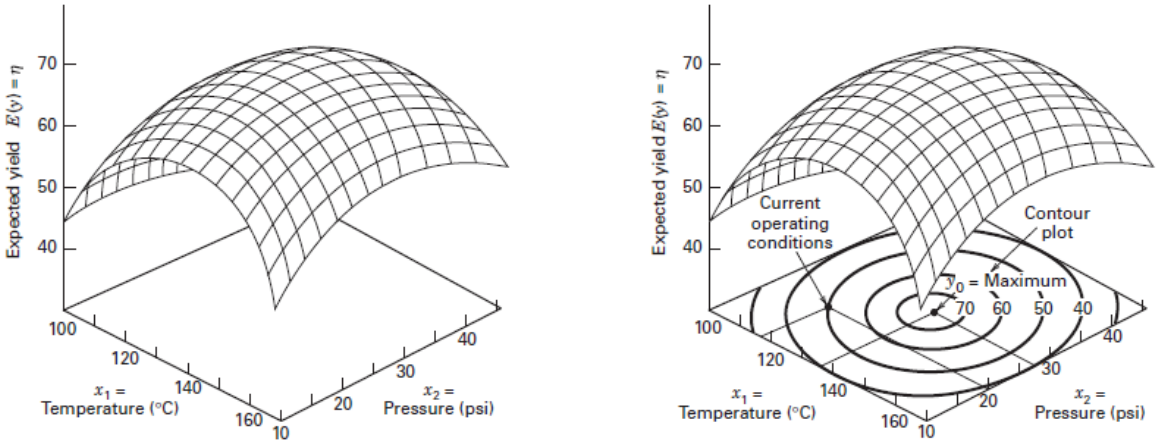


Figura 2. Superficie de respuesta tridimensional y gráfico de contornos correspondiente.

LOCALIZACIÓN DEL PUNTO ETACIONARIO

Los niveles o valores de los factores x_1, x_2, ..., x_k que optimizan la respuesta serán un conjunto para los que las derivadas parciales serán igual a 0:

$$\frac{\partial \hat{y}}{\partial \mathbf{x_1}} = \frac{\partial \hat{y}}{\partial \mathbf{x_2}} = \ldots = \frac{\partial \hat{y}}{\partial \mathbf{x_3}} = 0$$

Este se denomina **punto estacionario**, y puede representar un punto de **máxima respuesta**, de **mínima respuesta** o **punto silla** en la que la respuesta aumenta o disminuye a partir del punto estacionario en función de la dirección en la que nos movamos (ver figura 3).

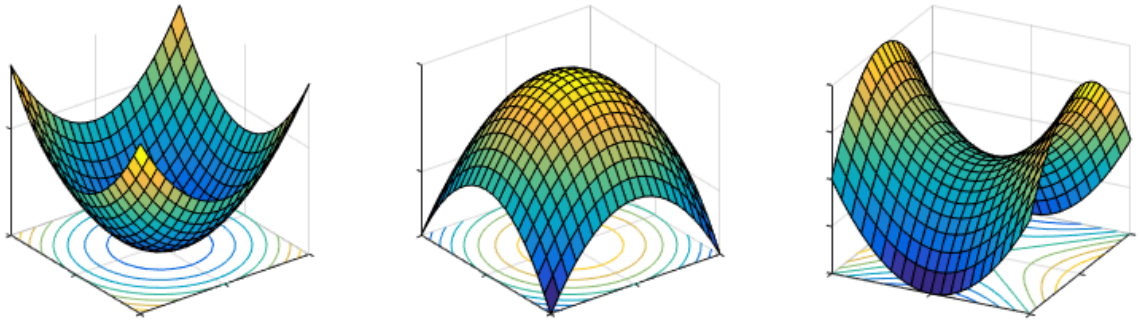


Figura 3. Punto estacionario mínimo (izquierda), máximo (centro) y punto silla (derecha).

Una ecuación matemática útil para la localización del punto estacionario sería la siguiente:

$$\hat{y} = \hat{\beta}_0 + x'b + x'Bx$$

siendo

$$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_k \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ . \\ . \\ . \\ \hat{\beta}_k \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \hat{\beta}_{11}, \hat{\beta}_{12}/2, \dots, \hat{\beta}_{1k}/2 \\ \hat{\beta}_{22}\ , \dots, \hat{\beta}_{2k}/2 \\ . \\ . \\ . \\ sym. \quad\quad\quad \hat{\beta}_{kk} \end{bmatrix}$$

donde **x** es el vector (k x 1) de los factores, **b** es el vector (k x 1) de los coeficientes de regresión de primer grado, y **B** la matriz simétrica (k x k) cuyos elementos en la diagonal son los coeficientes cuadráticos puros ($\hat{\beta}_{ii}$), y los de fuera de la diagonal corresponden a ½ de los coeficientes cuadráticos mixtos ($\hat{\beta}_{ij}, i \neq j$). La derivada de $\hat{y}$ con respecto a los elementos del vector $\boldsymbol{x}$ igualada a 0 es:

$$\frac{\partial \hat{y}}{\partial \mathbf{x}} = \mathbf{b} + 2\mathbf{B}\mathbf{x} = 0$$

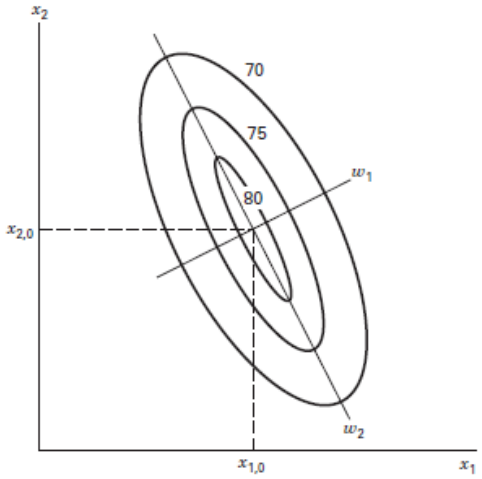
El punto estacionario será la solución a la ecuación anterior:

$$\mathrm{x}_s = -\frac{1}{2}\mathbf{B}^{-1}\mathbf{b}$$

Finalmente, podemos conocer el valor de la respuesta en dicho punto estacionario sustituyendo el valor del punto en la ecuación

$$\hat{y}_s = \hat{\beta}_0 + \frac{1}{2}\mathbf{x}'_s\mathbf{b}$$

Para caracterizar la superficie de respuesta o determinar el tipo de punto estacionario, se recurre al **análisis canónico**, en el cual se transforma el modelo a un nuevo sistema de coordenadas con el origen situado en el punto estacionario $\mathbf{x}_s$, rotando los ejes de este nuevo sistema para que sean paralelos a los ejes principales de la superficie de respuesta ajustada:



La **forma canónica** del modelo tomaría la forma:

$$\hat{y} = \hat{y}_s + \lambda_1 w_1^2 + \lambda_2 w_2^2 + \ldots + \lambda_k w_k^2$$

donde $\{w_i\}$ son las variables canónicas (variables independientes transformadas) y $\{\lambda_i\}$ las constantes o autovalores de la matriz $\mathbf{B}$.

La naturaleza de la superficie de respuesta se puede determinar a partir del punto estacionario y los signos y magnitudes de los valores $\{\lambda_i\}$. Teniendo en cuenta que el punto estacionario se encuentra dentro del entorno experimental:

- $\{\lambda_i\}$ positivas: $\mathbf{x}_s$ es un punto de mínima respuesta
- $\{\lambda_i\}$ negativas: $\mathbf{x}_s$ es un punto de máxima respuesta
- $\{\lambda_i\}$ positivas y negativas: $\mathbf{x}_s$ es un punto silla

Además, la superficie tendrá mayor pendiente en la dirección w_i para la que $|\lambda_i|$ es mayor.

Ejemplo:

Supongamos que contamos con el siguiente modelo, a partir del cual queremos obtener el valor y naturaleza del punto estacionario:

$$\hat{y} = 79{,}94 + 0{,}995x_1 + 0{,}515x_2 - 1{,}376{x_1}^2 - {x_2}^2 + 0{,}25x_1x_2$$

El vector **b** y la matriz **B** serán:

$$\mathbf{b} = \begin{bmatrix} 0{,}995 \\ 0{,}515 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} -1{,}376 & 0{,}125 \\ 0{,}125 & -1{,}001 \end{bmatrix}$$

Por tanto, el punto estacionario será:

$$\boldsymbol{x_s} = -\frac{1}{2}\begin{bmatrix} -0{,}7345 & -0{,}0917 \\ -0{,}0917 & -1{,}001 \end{bmatrix}\begin{bmatrix} 0{,}995 \\ 0{,}515 \end{bmatrix} = \begin{bmatrix} 0{,}389 \\ 0{,}306 \end{bmatrix}$$

es decir, $x_{1,s}$ = 0,389 y $x_{2,s}$ = 0,306, en términos de variables codificadas.

Sustituyendo en la ecuación:

$$\hat{y}_s = \hat{\beta}_0 + \frac{1}{2}\mathbf{x'}_s\mathbf{b}$$

obtenemos que la respuesta predicha en el punto estacionario es de 80,21.

Para expresar el modelo en su forma canónica debemos calcular primero los autovalores de la matriz **B,** teniendo en cuenta que λ_1 y λ_2 son las raíces de la ecuación determinante:

$$|\mathrm{B} - \lambda\,\mathrm{I}| = 0$$

$$\begin{bmatrix} -1{,}376 - \lambda & 0{,}125 \\ 0{,}125 & -1{,}001 - \lambda \end{bmatrix} = 0$$

lo cual se reduce a:

$$\lambda^2 + 2{,}378\lambda + 1{,}363 = 0$$

Las raíces de esta ecuación cuadrática son λ_1 = -0,963 y λ_2 = -1,414, por lo que la forma canónica del modelo será:

$$\hat{y} = 80{,}21 - 0{,}963w_1^2 - 1{,}414w_2^2$$

Siendo ambos valores de λ negativos, podemos concluir que el punto estacionario es un máximo.

Múltiples respuestas

Considerar el efecto que sobre más de una variable respuesta tienen k factores implica obtener un modelo de superficie de respuesta adecuado para cada una de ellas, para así tratar de encontrar el conjunto de condiciones que optimizan todas las respuestas o las mantiene en un determinado rango. Un método relativamente directo para encontrar las condiciones óptimas cuando contamos con un bajo número de factores es el de **superponer los gráficos de contorno** de cada respuesta. Sin embargo, cuando contamos con más de tres factores, este método se vuelve algo más complejo puesto que los gráficos de contorno son bidimensionales, por lo que tendríamos que mantener constante una o más variables para representar las (k – 2) variables restantes.

Un método matemático más formal supone utilizar la técnica de optimización simultánea propuesta por Derringer y Suich (1980), la cual hace uso de **funciones de deseabilidad**. Se basa en convertir cada respuesta y_i en una función de deseabilidad individual $\boldsymbol{d_i}$ que varíe entre 0 y 1 y que represente la cercanía de la respuesta a un valor ideal o deseado

$$0 \leq d_i \leq 1$$

Si y_i alcanza el valor deseado, entonces $\boldsymbol{d_i}$ = 1. Si por el contrario la respuesta alcanzada se encuentra fuera de un intervalo de tolerancia, entonces $\boldsymbol{d_i}$ = 0. A partir de las deseabilidades individuales, se calcula una deseabilidad general (**D**). Cada d_i se escoge para maximizar D:

$$\boldsymbol{D} = (d_1 \cdot d_2 \cdot \ldots \cdot d_m)^{1/m}$$

donde m es el número de variables respuesta.

Según la ecuación anterior podemos ver que, si alguna de las deseabilidades individuales es 0, lo será también la deseabilidad total.

Si el objetivo (***T***) es **maximizar** una respuesta *y*, entonces su deseabilidad individual será:

$$d = \begin{cases} 0 & y < L \\ \left(\dfrac{y - L}{T - L}\right)^r & L \leq y \leq T \\ 1 & y > T \end{cases}$$

donde $\boldsymbol{L}$ es el límite mínimo, y $\boldsymbol{r}$ un valor de peso asociado. Si r = 1, la función de deseabilidad es lineal (ver figura 4). Escoger r > 1 pone énfasis en acercarse lo máximo posible al valor objetivo.

Si el objetivo es **minimizar** una respuesta y, entonces su deseabilidad individual será:

$$d = \begin{cases} 1 & y < T \\ \left(\dfrac{U - y}{U - T}\right)^r & T \leq y \leq U \\ 0 & y > U \end{cases}$$

donde ***U*** es el límite máximo. Tanto el límite mínimo como máximo lo establece el experimentador en función de los valores que considere oportunos.

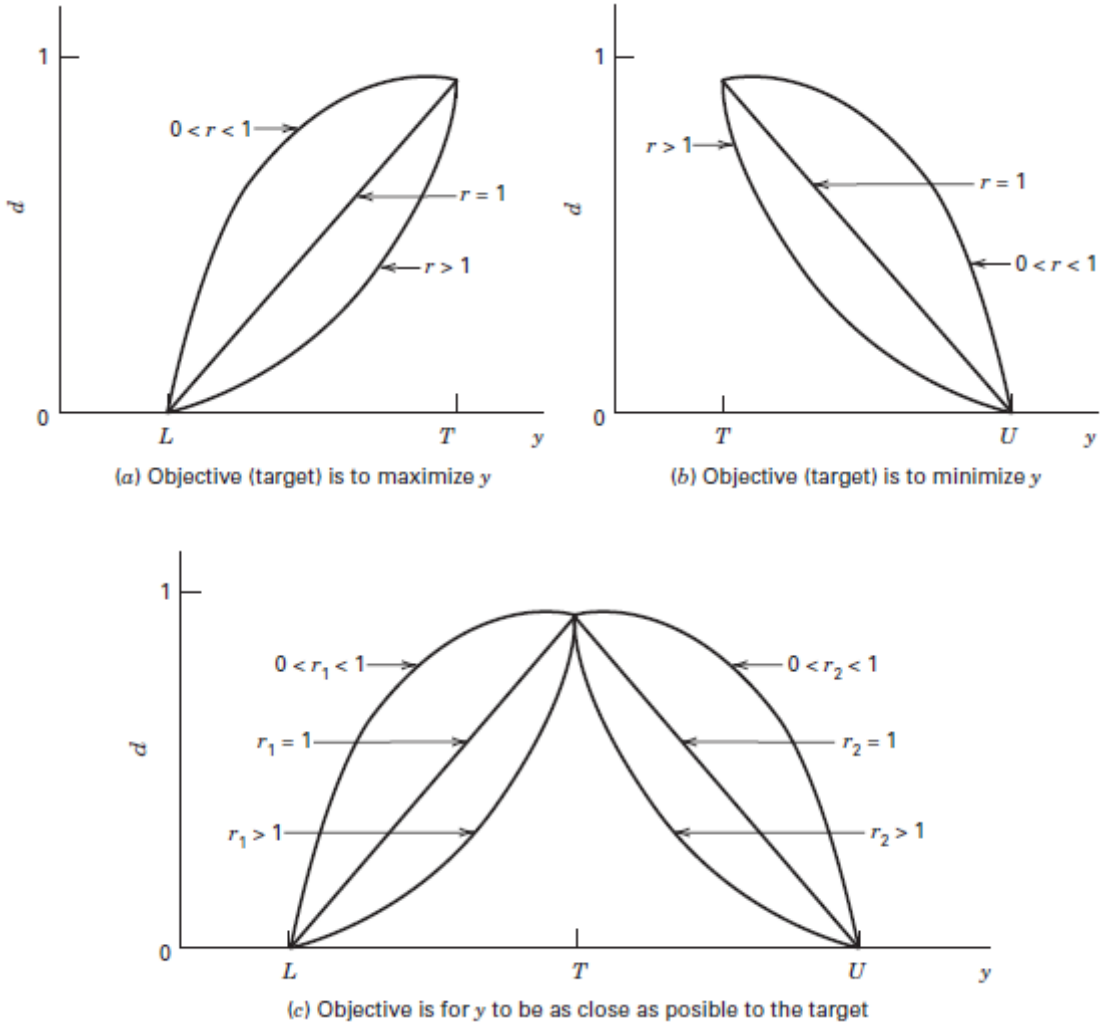


Figura 4.. Funciones de deseabilidad individual para optimización simultánea de varias respuestas.

DISEÑOS DE SUPERFICIE DE RESPUESTA

Para estimar los parámetros del modelo de manera efectiva, debemos aplicar un diseño experimental apropiado para recopilar los datos necesarios. Algunas de las características más importantes de un buen diseño son:

- Proporciona una distribución razonable de puntos de datos y, por tanto, de información
- Permite estudiar la adecuación del modelo (falta de ajuste)
- Proporciona estimadores precisos de los coeficientes del modelo
- Proporciona una estimación interna del error
- Permite realizar experimentos en bloques
- No requiere de un gran número de experimentos
- No requiere de demasiados niveles de las variables independientes

Además de las características mencionadas, es conveniente que el diseño escogido sea **ortogonal,** en el que los términos del modelo ajustado, y, por tanto, las estimaciones de los parámetros, no presenten correlación. En este caso, la varianza de la respuesta esperada es expresable como la suma ponderada de las varianzas de los parámetros estimados.

Que el diseño sea **invariante por rotación** resulta también interesante ya que asegura que la varianza de la respuesta esperada dependa solamente de la distancia de un punto al centro del diseño, y no de la dirección (ver figura 5). Esto es importante ya que la localización del óptimo es desconocida.

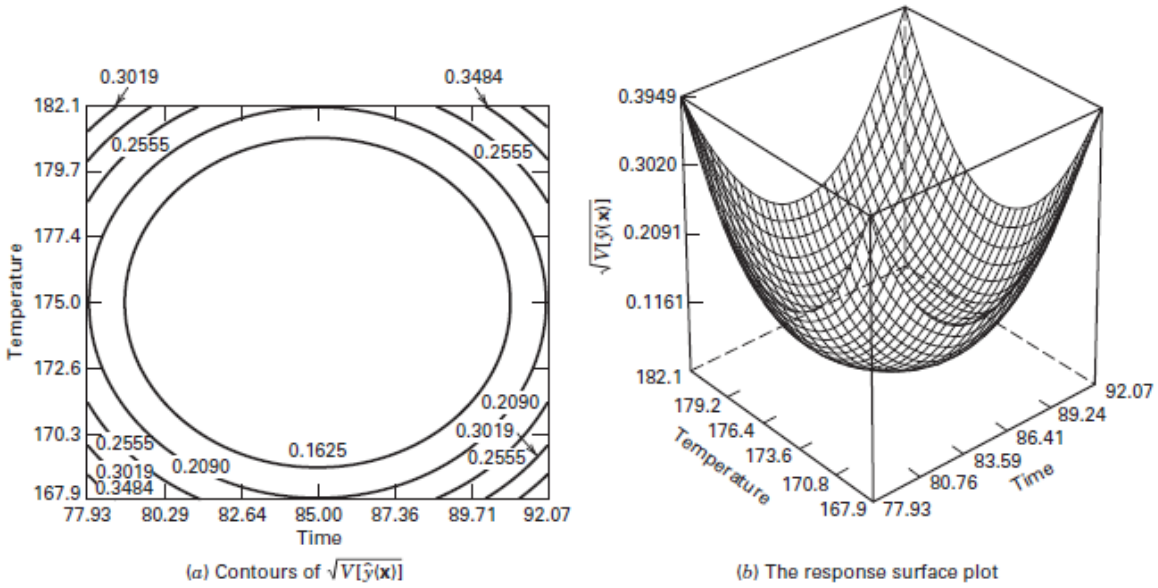


Figura 5. Contornos de desviación estándar de la respuesta estimada de un diseño CCD rotable.

Diseños para modelos de primer orden

Los diseños ortogonales de primer orden son la única clase de diseños que minimizan la varianza de los coeficientes de regresión $\{\hat{\beta}_i\}$ en este escenario. Estos incluyen:

- Diseños factoriales 2^k
- Fracciones de la serie 2^k
- Diseños simplex

Diseños factoriales 2^k

Los diseños factoriales 2^k son útiles en estadíos experimentales iniciales de un estudio de superficie de respuesta, proporcionando el número mínimo de combinaciones factoriales.

En este tipo de diseños, se consideran dos niveles para cada factor k_i: un valor máximo codificado como +1, y un valor mínimo codificado como -1, elaborando todas las posibles combinaciones de los niveles de los k factores. Por tanto, obtenemos 2^k combinaciones o tratamientos, y el diseño obtenido es ortogonal. Por ejemplo, si contamos con un diseño 2323 con tres factores (A, B y C), las combinaciones resultantes se recogen en la tabla 1:

Run	*A*	*B*	*C*
1	−	−	−
2	+	−	−
3	−	+	−
4	+	+	−
5	−	−	+
6	+	−	+
7	−	+	+
8	+	+	+

Tabla 1. Combinación de factores resultante de un diseño factorial 2^3.

Uno de los inconvenientes de este tipo de diseños es que no permiten la estimación del error experimental, pero este puede resolverse incluyendo observaciones repetidas en el centro del diseño, sin que esto influya sobre las estimaciones de los parámetros o la ortogonalidad del diseño. En este caso, β_0 representaría la media de todas las observaciones.

Fracciones de la serie 2^k

Para un diseño factorial 2^k, el número de combinaciones y parámetros a estimar aumenta rápidamente con el aumento del número de factores. En función del coste del experimento, puede plantearse la omisión de algunas de las combinaciones o puntos experimentales de este diseño, considerándose únicamente un conjunto de 2^{k-m} tratamientos, siendo $k \geq m$. Como restricción, hay que tener en cuenta que el diseño resultante debe mantener mínimo $k + 1$ puntos (por el número de parámetros a estimar), además de la propiedad de ortogonalidad, a ser posible.

Diseño simplex

La principal característica de un diseño simplex es que requiere $N = k + 1$ observaciones para ajustar un modelo de primer orden con k factores. Geométricamente, los puntos del diseño representan los vértices de una figura regular (por ejemplo, un triángulo equilátero si $k = 2$, o un tetraedro si $k = 3$).

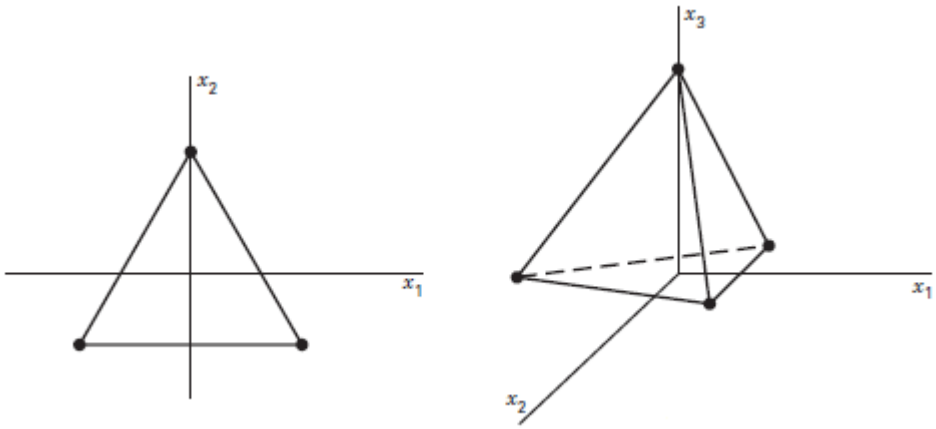


Figura 6. Diseño simplex k = 2 factores (izquierda) y k = 3 factores (derecha).

Diseños para modelos de segundo orden

Para estimar los parámetros de un modelo polinómico de segundo grado podrían utilizarse diseños factoriales 3^k, pero ello requeriría de un número de combinaciones muy alto. Existen otra clase de diseños que permiten ajustar este tipo de modelo con un número menor de combinaciones. Entre ellos están:

- Diseños Centrales Compuestos
- Diseños Box-Behnken

Diseño Central Compuesto (CCD)

Se trata de la clase más popular para ajustar modelos de regresión cuadrática. Dispone de tres variantes:

- Circunscrito (CCC)
- Centrado en las caras (CCF)
- Inscrito (CCI)

CCC

Los puntos experimentales a obtener con este tipo de diseño son:

$$N = n_f + 2k + n_c$$

donde

- n_f: 2^k puntos factoriales: máximo y mínimo del rango de cada variable
- $2k$ puntos axiales: nuevos extremos máximo y mínimo de cada factor
- n_c puntos centrales replicados. En el libro *Design and Analysis of Experiments* (Douglas C. Montgomery) recomiendan de 3 a 5 réplicas.

El diseño central compuesto requiere especificar, además, un parámetro adicional determinante para ubicar los puntos axiales:

- Distancia (α) de los puntos axiales al centro del diseño. Para cumplir la propiedad de rotabilidad, se utiliza:

$$\alpha = (2^k)^{1/4}$$

El diseño cuenta, pues, con **5 niveles por factor (-α, -1, 0, +1, +α)**: punto axial mínimo, punto factorial mínimo, punto central, punto factorial máximo y punto axial máximo (ver figura 7 para el caso de 2 y 3 factores).

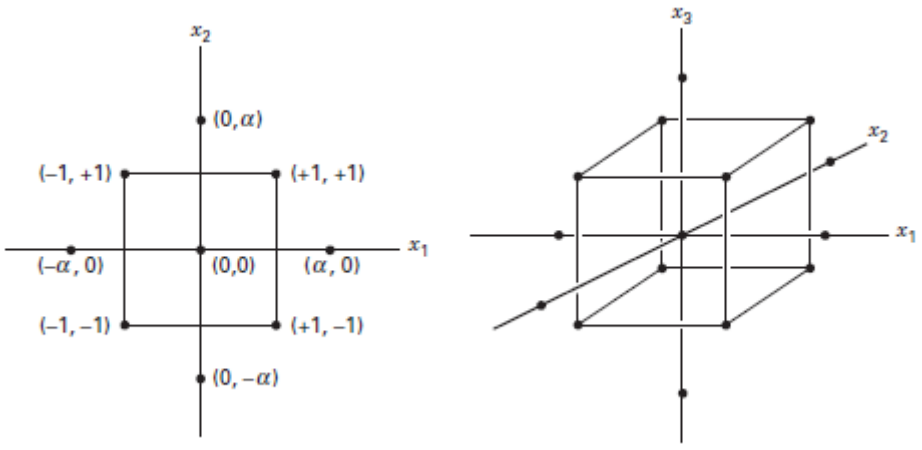


Figura 7. Diseño central compuesto inscrito (CCI) para k = 2 factores (izquierda) y k = 3 factores (derecha).

CCF

Para esta variante se emplea **α = 1**, con lo que todos los puntos axiales recaen sobre la superficie de las caras del diseño, como puede verse en la figura 8. Este diseño mantiene el número de **3 niveles por factor (-1, 0, +1)**, en lugar de 5, lo cual puede ser difícil o costoso en algunas situaciones. Se evita, además, combinaciones con valores extremos de los factores. El número de réplicas del punto central también puede ser menor (n_c = 2 o 3 según el libro *Design and Analysis of Experiments*) para lograr una varianza estable. Sin embargo, este diseño no mantiene la propiedad de rotabilidad, como puede verse en la figura 9, aunque la ausencia de esta propiedad puede compensar el ahorro de recursos experimentales.

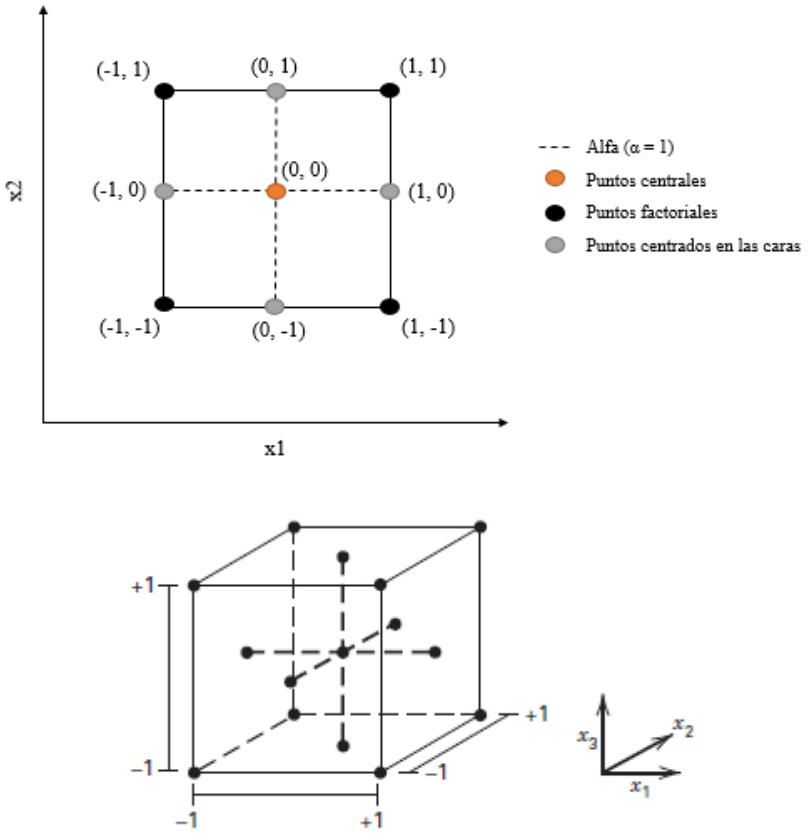


Figura 8. Diseño CCD centrado en las caras para k = 2 y k = 3.

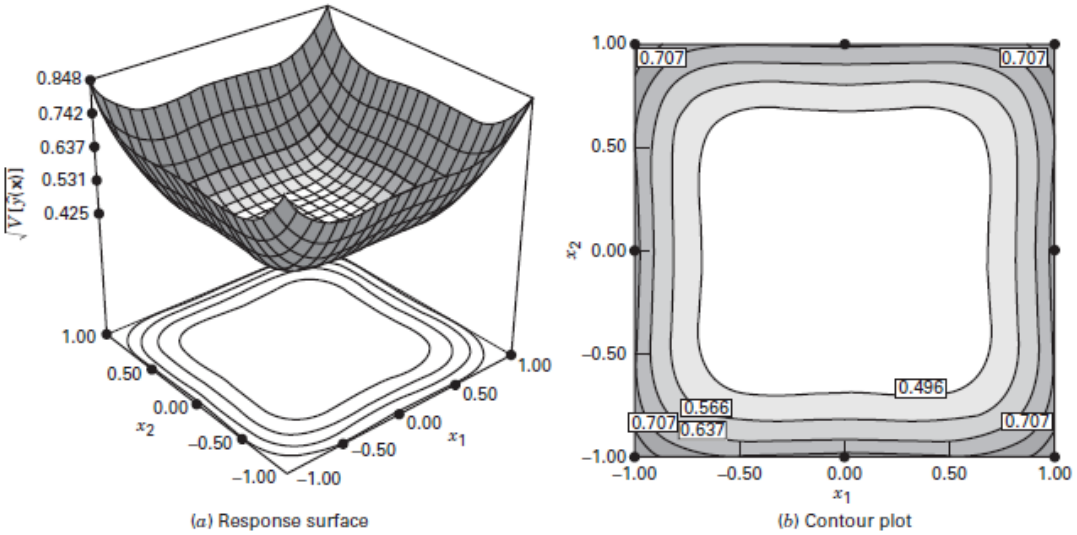


Figura 9. Evolución de la desviación estándar de la respuesta estimada de un diseño CCD centrado en las caras con k = 3, n_c = 3 y x_3 = 0.

CCI

La versión inscrita del diseño central compuesto se aplica cuando los límites inferior y superior de cada factor son realmente límites de operabilidad. Requiere también de cinco niveles por factor. Cada nivel factorial queda reducido proporcionalmente con respecto a un diseño CCC, de manera que se usan los niveles establecidos de cada factor como los puntos axiales, quedando el diseño factorial situado dentro de estos límites: los puntos axiales se sitúan a ±1, y los puntos del cubo del diseño a ± √1/2.

En la siguiente imagen puede verse la diferencia entre el diseño CCI y los dos explicados anteriormente, donde los valores -1 y +1 indican el rango factorial:

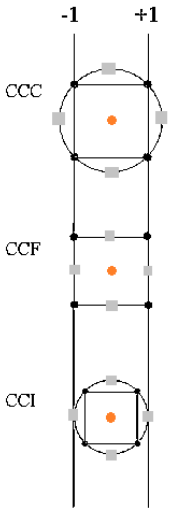


Figura 10. Comparación entre los tres tipos de variantes de un diseño CCD.

Diseño Box-Behnken

Este diseño, propuesto por Box and Behnken (1960), se utiliza para ajustar superficies de respuesta con al menos tres factores, con **tres niveles (-1, 0, +1) por factor**. De este diseño es destacable que no se incluye ningún punto experimental con valores máximos de todos los factores, es decir, no contempla puntos en los vértices del diseño cúbico o puntos factoriales 2^k (ver figura 11). Esto supone una ventaja si estas condiciones no son operables por limitaciones físicas o económicas del proceso experimental.

El número de experimentos requeridos para este diseño viene dado por:

$$N = 2k(k - 1) + n_c$$

Como los diseños de Box-Behnken suelen tener menos puntos experimentales que un diseño CCD, son menos costosos de llevar a cabo con un mismo número de factores.

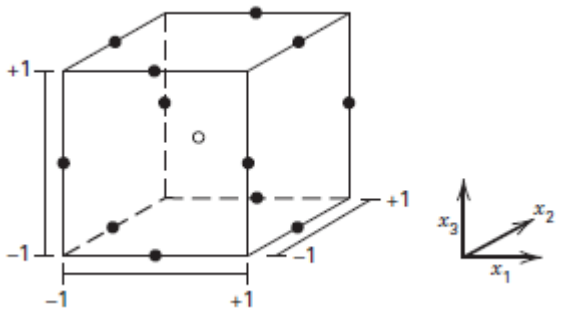


Figura 11. Diseño Box-Behnken con k = 3.

BLOQUEO DE DISEÑOS

Uno de los aspectos importantes a tener en cuenta cuando elegimos un diseño de superficie de respuesta es la posibilidad de llevar a cabo los experimentos en grupos o **bloques**. Cuando el número de corridas experimentales es muy grande como para llevarlas a cabo todas de una sola tanda, se corre el riesgo de introducir error en el experimento debido a posibles variables interferentes o cambios en las condiciones experimentales si pasa un tiempo considerable entre experimentos del mismo diseño. Es por ello que llevar a cabo el experimento en bloques nos permite separar y estimar de forma independiente los efectos del bloqueo de los efectos de los factores, que son los que nos interesa estimar. A esta situación se le llama **bloqueo ortogonal**. Este nombre viene del hecho de que el diseño se divide de tal forma que el efecto

de los bloques o grupos no afecte la estimación de los parámetros del modelo. Los bloques pueden hacer referencia a días, material, etc.

Para un diseño central compuesto, el número de bloques ortogonales depende del número de factores, número de corridas experimentales, y del diseño que se escoja. Por ejemplo, el diseño CCF no permite este tipo de bloqueo, mientras que el CCC sí lo permite. Por otro lado, el diseño Box-Behnken posibilita el bloqueo si contamos con cuatro o cinco factores. Generalmente, cuando procedemos a agrupar los experimentos en dos bloques, uno de ellos contendrá los puntos factoriales más algunas réplicas, mientras que el otro contendrá los puntos axiales junto con el resto de réplicas (ver figura 12), teniendo en cuenta que las réplicas han de repartirse equitativamente entre todos los bloques. En el caso de tres bloques, repartiríamos los puntos factoriales entre dos bloques, dejando los puntos axiales únicamente en uno de ellos, repartiendo de nuevo equitativamente las réplicas del punto central entre los tres bloques.

$$\mathbf{D} = \begin{array}{c} \\ \left[\begin{array}{cc} -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1.414 & 0 \\ -1.414 & 0 \\ 0 & 1.414 \\ 0 & -1.414 \\ 0 & 0 \\ 0 & 0 \end{array}\right] \end{array} \begin{array}{l} x_1 \quad x_2 \\ \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right\} \text{Block 1} \\ \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right\} \text{Block 2} \end{array}$$

Figura 12. Diseño CCD rotable con k = 2 y 12 experimentos en dos bloques.

EJEMPLOS EN R

En R está disponible el paquete **rsm** (https://cran.r-project.org/web/packages/rsm/index.html) que proporciona funciones para generar diseños de superficie de respuesta (CCD y Box-Behnken), ajustar los correspondientes modelos de primer y segundo orden, generar gráficos de superficie, obtener la trayectoria de ascenso/descenso en máxima pendiente y llevar a cabo el análisis canónico. De todas las funcionalidades disponibles, mostraremos en este documento las funciones más destacadas y útiles.

Crear diseños CCD

CCD

Para crear un diseño central compuesto tenemos disponible la función **ccd** del paquete **rsm**. Algunos de los principales argumentos de esta función son:

- `basis`: número de factores
- `n0`: número de puntos centrales. Se duplican si oneblock = TRUE
- `randomize`: valor lógico para indicar si se desea aleatorizar las corridas experimentales
- `alpha`: valor numérico o *string* ("*orthogonal*", "*rotatable*", "*spherical*", "*faces*") que indica la posición de los puntos axiales.
- `inscribed`: valor lógico para indicar si queremos aplicar la versión del diseño inscrito.
- `oneblock`: valor lógico para indicar si se quiere que el diseño pertenezca a un único bloque
- `blocks`: *string* con el nombre del factor de bloqueo
- `coding`: lista de fórmulas para codificar los factores del diseño

A continuación, se muestra un ejemplo de aplicar la función **ccd** para generar un diseño CCC (rotable) aleatorizado en dos bloques (días), con dos factores (temperatura: x1, pH: x2) y cuatro réplicas del punto central. El rango ([-1, +1]) de cada factor será:

- x_1: [30 – 50]
- x_2: [4,4 – 6,6]

```
library(rsm)

# Diseño CCC
set.seed(123)
diseno_ccc <- ccd(basis = 2,
                  n0 = 2,
                  randomize = FALSE,
                  alpha = "rotatable",
```

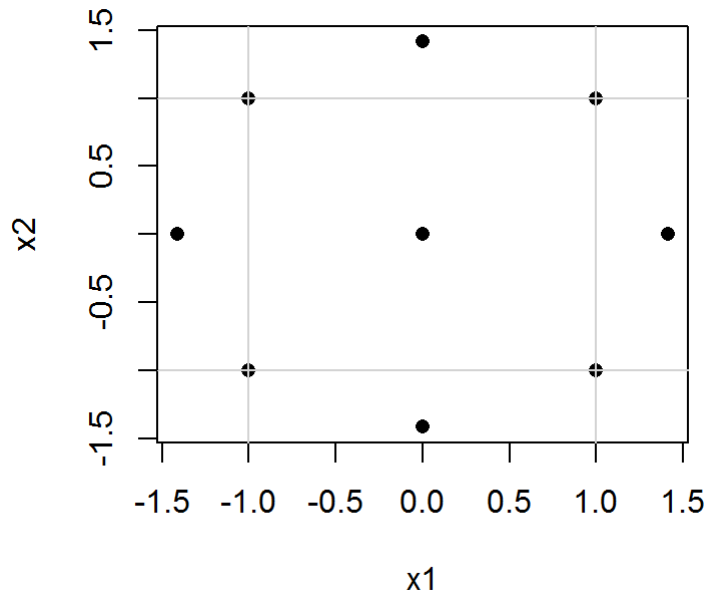
```
                  oneblock= TRUE,
                  inscribed = FALSE,
                  coding = list(x1 ~ (Temp - 40)/10, x2 ~ (pH - 5.5)/1.10))

diseno_ccc
```

```
##    run.order std.order     Temp       pH
## 1          1         1 30.00000 4.400000
## 2          2         2 50.00000 4.400000
## 3          3         3 30.00000 6.600000
## 4          4         4 50.00000 6.600000
## 5          5         5 40.00000 5.500000
## 6          6         6 40.00000 5.500000
## 7          1         1 25.85786 5.500000
## 8          2         2 54.14214 5.500000
## 9          3         3 40.00000 3.944365
## 10         4         4 40.00000 7.055635
## 11         5         5 40.00000 5.500000
## 12         6         6 40.00000 5.500000
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ (Temp - 40)/10
## x2 ~ (pH - 5.5)/1.1
```

```
# Diseño en unidades codificadas
as.data.frame(diseno_ccc)
```

```
##    run.order std.order        x1        x2
## 1          1         1 -1.000000 -1.000000
## 2          2         2  1.000000 -1.000000
## 3          3         3 -1.000000  1.000000
## 4          4         4  1.000000  1.000000
## 5          5         5  0.000000  0.000000
## 6          6         6  0.000000  0.000000
## 7          1         1 -1.414214  0.000000
## 8          2         2  1.414214  0.000000
## 9          3         3  0.000000 -1.414214
## 10         4         4  0.000000  1.414214
## 11         5         5  0.000000  0.000000
## 12         6         6  0.000000  0.000000
```

```
# Distribucion grafica de los puntos del diseño
plot(diseno_ccc[ , c(3:4)], pch = 16)
abline(h = c(1, -1), col = "lightgrey")
abline(v = c(1, -1) , col = "lightgrey")
```



Con dos factores, el valor de α que se ha aplicado para crear los puntos axiales es de $(2^2)^{1/4}$ = 1,4142. Por tanto, los niveles codificados para cada factor son de (-1.4142, -1, 0, 1, 1.4142). El objeto creado guarda estos valores codificados, aunque al imprimir en la consola muestra los valores naturales.

Para generar un diseño CCI, especificaríamos `alpha = "spherical"`. Para generar un diseño CCF especificaríamos `alpha = "faces"` junto con `inscribed = TRUE`. Incluir `alpha = "orthogonal"` es útil cuando un experimento con datos para un modelo de primer grado se expande para incluir los datos de los puntos axiales necesarios para ajustar un modelo de segundo grado. En este caso, estos datos son añadidos en un nuevo bloque que ha de ser ortogonal con respecto al resto de bloques que contienen los datos del cubo del diseño.

BOX-BEHNKEN

Para crear diseños Box-Behnken tenemos disponible la función **`bbd()`** del paquete **`rsm`**. Sus argumentos son los siguientes:

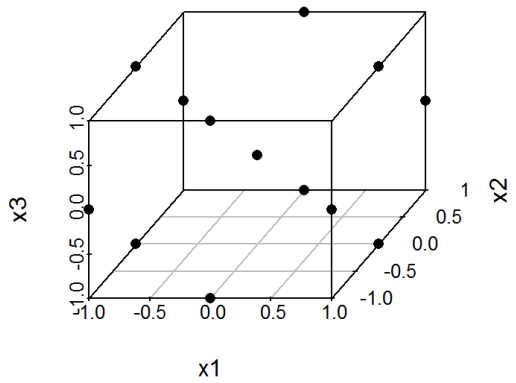
- `k`: número de factores (mínimo 3, máximo 7)
- `n0`: número de puntos centrales
- `block`: valor lógico especificando especificando si se desea bloquear el diseño, o *string* con el nombre del factor de bloqueo. Solo se pueden bloquear diseños con 4 o 5 factores.
- `randomize`: valor lógico para indicar si se desea aleatorizar las corridas experimentales
- `coding`: lista de fórmulas para codificar los factores del diseño (opcional)

A continuación, se muestra un ejemplo de cómo crear un diseño Box-Behnken aleatorizado con tres factores y tres puntos centrales:

```r
# Diseño Box-Behnken
set.seed(123)
diseno_bbd <- bbd(k = 3,
                  n0 = 3,
                  coding = list(x1 ~ (Temp - 20)/3,
                                x2 ~ (pH - 50)/10,
                                x3 ~ (Vel - 140)/95),
                  randomize = TRUE,
                  block = FALSE) # TRUE solo con k = 4 o 5

diseno_bbd
```

```
##    run.order std.order Temp pH Vel
## 1          1         6   23 50  45
## 2          2        12   20 60 235
## 3          3        10   20 60  45
## 4          4         9   20 40  45
## 5          5         1   17 40 140
## 6          6         3   17 60 140
## 7          7        14   20 50 140
## 8          8         8   23 50 235
## 9          9        11   20 40 235
## 10        10        15   20 50 140
## 11        11         4   23 60 140
## 12        12         2   23 40 140
## 13        13        13   20 50 140
## 14        14         5   17 50  45
## 15        15         7   17 50 235
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ (Temp - 20)/3
## x2 ~ (pH - 50)/10
## x3 ~ (Vel - 140)/95
```

```r
library(scatterplot3d)

# Distribucion grafica de los puntos del diseño
scatterplot3d(diseno_bbd[ , c(3:5)], pch = 16, angle = 55)
```



Análisis (modelo de primer orden)

Tomaremos del libro *Design and Analysis of Experiments* el ejemplo de un experimento químico en el que se deciden estudiar dos factores (x_1: tiempo; x_2: temperatura), con datos sobre una variable respuesta (y_1: rendimiento). *En el libro se muestra el resultado de llevar a cabo el análisis con el software Design Expert, aquí mostraremos como hacerlo en R.*

Imaginemos que deseamos maximizar la respuesta y_1, pero no sabemos con exactitud cuales pueden ser los valores de x_1 y x_2 más apropiados para maximizarla. Por tanto, decidimos empezar aplicando un diseño factorial 2_k aumentado con cinco réplicas del punto central (x_1 = 0, x_2 = 0) para poder estimar el error experimental y la adecuación de un modelo. Se decide probar un rango inicial de tiempo de [30 - 40] min, y temperatura de [150 - 160]ºF. En este caso contamos con el diseño y los datos recogidos en un archivo *.excel*, descargable en mi repositorio. Importaremos este archivo y codificaremos los factores al intervalo [-1, 1] con la función `coded.data()`, también perteneciente al paquete `rsm`. Para codificarlos, aplicaremos las ecuaciones de conversión:

$$x_1 = \frac{\xi_1 - [max(\xi_1) + min(\xi_1)]/2}{max(\xi_1) - min(\xi_1)]/2} = \frac{\xi_1 - (40+30)/2}{(40-30)/2} = \frac{\xi_1 - 35}{5}$$

$$x_2 = \frac{\xi_2 - [max(\xi_2) + min(\xi_2)]/2}{max(\xi_2) - min(\xi_2)]/2} = \frac{\xi_2 - (160+150)/2}{(160-150)/2} = \frac{\xi_2 - 155}{5}$$

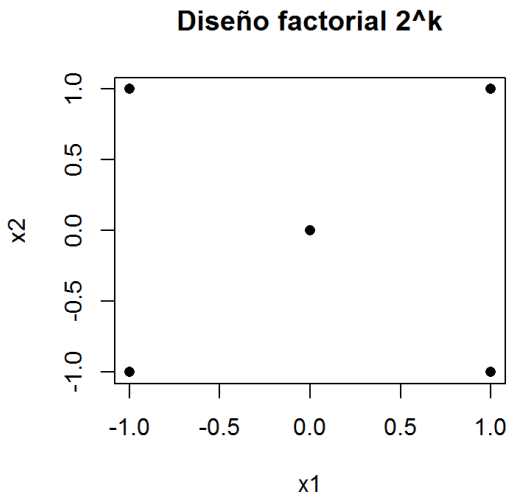
```
library(readxl)

# Importamos los datos
datos_ccc <- read_excel("datos_rsm_ccc.xlsx", sheet = 1)
datos_ccc
```

```
## # A tibble: 9 x 5
##   run.order tiempo temperatura bloque    y1
##       <dbl>  <dbl>       <dbl>  <dbl> <dbl>
## 1         1     30         150      1  39.3
## 2         2     30         160      1  40
## 3         3     40         150      1  40.9
## 4         4     40         160      1  41.5
## 5         5     35         155      1  40.3
## 6         6     35         155      1  40.5
## 7         7     35         155      1  40.7
## 8         8     35         155      1  40.2
## 9         9     35         155      1  40.6
```

```
# Codificamos los factores
datos_ccc <-  coded.data(data = datos_ccc,
                         x1 ~ (tiempo - 35)/5, x2 ~ (temperatura - 155)/5)

as.data.frame(datos_ccc)
```

```
##   run.order x1 x2 bloque   y1
## 1         1 -1 -1      1 39.3
## 2         2 -1  1      1 40.0
## 3         3  1 -1      1 40.9
## 4         4  1  1      1 41.5
## 5         5  0  0      1 40.3
## 6         6  0  0      1 40.5
## 7         7  0  0      1 40.7
## 8         8  0  0      1 40.2
## 9         9  0  0      1 40.6
```

```
# Grafico del diseño
plot(datos_ccc[ , c(2:3)], pch = 16, main = "Diseño factorial 2^k")
```



Con los datos disponibles, nos disponemos a ajustar un modelo de primer orden mediante mínimos cuadrados con la función `rsm()`, cuyos argumentos básicos son:

- **`formula`**: fórmula con los factores y variable respuesta para el modelo. Se debe incluir al menos uno de los términos `FO()` (modelo lineal) o `SO()` (polinomio de segundo orden con intercciones). Con `TWI()` podemos añadir interacciones entre factores, y con `PQ()` se añaden términos cuadráticos puros.
- **`data`**: datos con los que generar el modelo.

Generaremos también el gráfico de superficie en 3D con la función `persp()` y el correspondiente gráfico de contornos con la función `contour()`:

```
# Modelo de primer orden
modelo1 <- rsm(y1 ~ FO(x1, x2), data = datos_ccc)

# Resultado del ajuste
summary(modelo1)
```

```
##
## Call:
## rsm(formula = y1 ~ FO(x1, x2), data = datos_ccc)
##
##              Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 40.444444   0.057288 705.9869 5.451e-16 ***
## x1           0.775000   0.085932   9.0188  0.000104 ***
## x2           0.325000   0.085932   3.7821  0.009158 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.941,  Adjusted R-squared:  0.9213
## F-statistic: 47.82 on 2 and 6 DF,  p-value: 0.0002057
##
## Analysis of Variance Table
##
## Response: y1
##             Df  Sum Sq Mean Sq F value    Pr(>F)
## FO(x1, x2)   2 2.82500 1.41250 47.8213 0.0002057
## Residuals    6 0.17722 0.02954
## Lack of fit  2 0.00522 0.00261  0.0607 0.9419341
## Pure error   4 0.17200 0.04300
##
## Direction of steepest ascent (at radius 1):
##        x1        x2
## 0.9221944 0.3867267
##
## Corresponding increment in original units:
##      tiempo temperatura
##    4.610972    1.933633
```

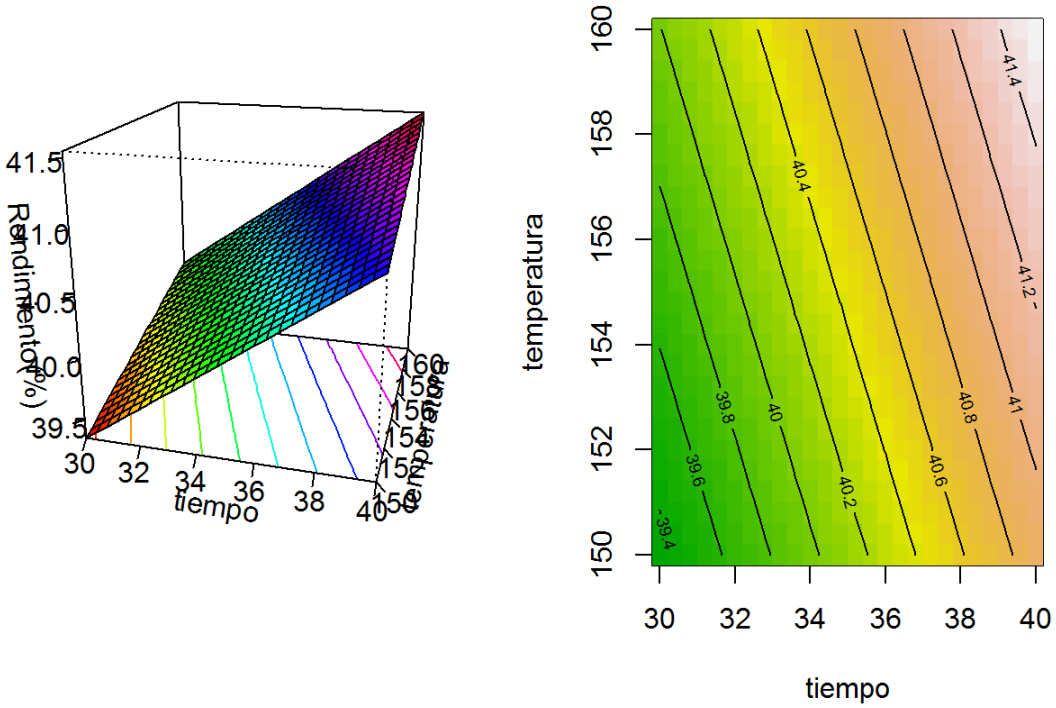
La significancia estadística de los coeficientes estimados, y por tanto del efecto de los factores, viene determinada por los valores del estadístico t del *t-test* y el p-valor asociado (p-valor < 0,05 supone un efecto significativo). Del resultado del ajuste podemos ver que los estimadores $\beta_0 = 40$, $\beta_1 = 0{,}775$ y $\beta_2 = 0{,}325$son estadísticamente significativos (p-valor < 0.05). En conjunto, la regresión resulta estadísticamente significativa (F = 47,82, p-valor < 0,05). El coeficiente de determinación ajustado también presenta un valor alto ($R^2_{adj} = 0{,}921$). De la tabla de análisis de varianza (ANOVA) del modelo lineal podemos ver que la falta de ajuste de este modelo no es estadísticamente significativa (p-valor = 0,941). En conclusión, el modelo se aproxima bastante a los datos y resulta útil para modelarlos. Su ecuación en unidades codificadas sería la siguiente:

$$\hat{y} = 40{,}444 + 0{,}775x_1 + 0{,}325x_2$$

Al tratarse de un buen ajuste con un modelo lineal, podemos concluir que no nos encontramos en el entorno del máximo de la variable respuesta de la función que estamos buscando, lo cual puede verse también en el gráfico de superficie del modelo:

```
# Superficie 3D
par(mfrow = c(1,2))
```

```
persp(modelo1, x2 ~ x1,
      zlab = "Rendimiento(%)",
      contours = list(z = "bottom", col = "colors"), # posicion y color
      at = c(summary(modelo1$canonical$xs)),
      theta = 15, # coordenadas graficas
      phi = 20)

# Grafico de contornos
contour(modelo1, ~ x1 + x2, image = TRUE)
```

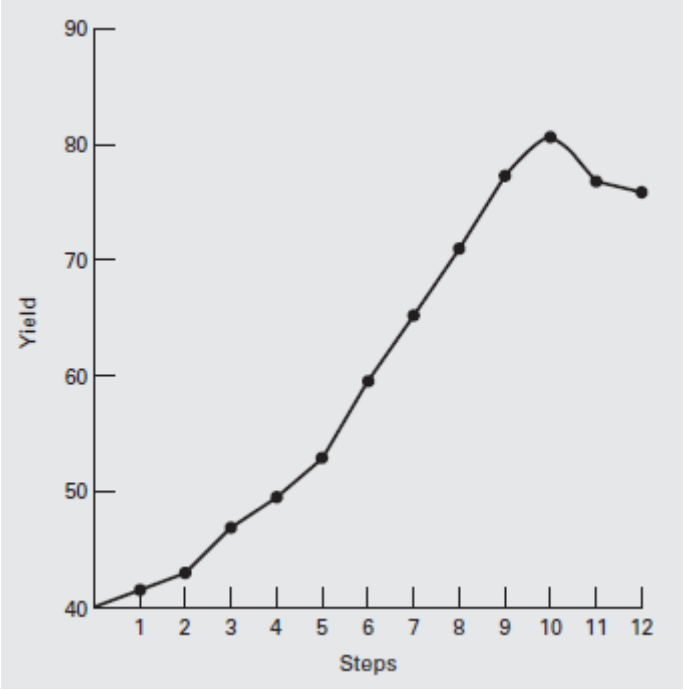


La última información que nos devuelve el ajuste es la dirección de ascenso en pendiente máxima, tanto en unidades codificadas como naturales. En concreto, por cada 4,61 min de tiempo que aumentemos, incrementaremos la temperatura en 1,93ºF. Podemos ver gráficamente que es el factor x_1 el que provoca un mayor aumento en la respuesta (es el factor con el mayor coeficiente absoluto de regresión).

El experimentador en este caso redondea estos valores a 5 min y 2ºF. Por tanto, el siguiente paso es volver a realizar los experimentos y tomar nuevas medidas de y_1 para incrementos secuenciales de $\Delta\xi_1$=5 min y $\Delta\xi_2$=2ºF. Los resultados se muestran en la siguiente tabla tomada del libro:

Steps	x_1	x_2	ξ_1	ξ_2	y
Origin	0	0	35	155	
Δ	1.00	0.42	5	2	
Origin + Δ	1.00	0.42	40	157	41.0
Origin + 2Δ	2.00	0.84	45	159	42.9
Origin + 3Δ	3.00	1.26	50	161	47.1
Origin + 4Δ	4.00	1.68	55	163	49.7
Origin + 5Δ	5.00	2.10	60	165	53.8
Origin + 6Δ	6.00	2.52	65	167	59.9
Origin + 7Δ	7.00	2.94	70	169	65.0
Origin + 8Δ	8.00	3.36	75	171	70.4
Origin + 9Δ	9.00	3.78	80	173	77.6
Origin + 10Δ	10.00	4.20	85	175	80.3
Origin + 11Δ	11.00	4.62	90	179	76.2
Origin + 12Δ	12.00	5.04	95	181	75.1

De la tabla podemos ver que, hasta el décimo incremento, los valores de rendimiento aumentan, y a partir de aquí, incrementar más x_1 y x_2 provoca una disminución. Esta misma evolución puede verse gráficamente en la siguiente figura:



Podemos ver en la evolución gráfica que es en torno al punto ξ_1 = 85min y ξ_2 = 175ºF donde puede encontrarse el óptimo o máximo del valor del rendimiento, por lo que el siguiente paso será generar un nuevo diseño tomando estos como valores centrales.

Análisis (modelo de segundo orden)

Continuando con el ejemplo anterior, se decide aumentar el diseño factorial con los puntos axiales ($x_1 = 0, x_2 \pm \alpha$) y ($x_1 \pm \alpha, x_2 = 0$) para generar un diseño CCC que nos permita evaluar el ajuste de un polinomio de grado 2. El rango elegido para el factor tiempo ξ_1 es [80, 90]min, y para la temperatura ξ_2 [170, 180]ºF. Además, se ha considerado una variable respuesta adicional de interés: peso molecular (y_2). Todos los experimentos del diseño son llevados a cabo en una sola tanda, por lo que se decide no incluir la variable de bloqueo.

Volvemos a importar los datos que tenemos almacenados en un archivo *.excel*:

```
# Importamos los datos
datos_ccc <- read_excel("datos_rsm_ccc.xlsx", sheet = 2)
datos_ccc
```

```
## # A tibble: 13 x 6
##    run.order tiempo temperatura bloque    y1    y2
##        <dbl>  <dbl>       <dbl>  <dbl> <dbl> <dbl>
##  1         1   80          170       1  76.5  2940
##  2         2   80          180       1  77    3470
##  3         3   90          170       1  78    3680
##  4         4   90          180       1  79.5  3890
##  5         5   85          175       1  79.9  3480
##  6         6   85          175       1  80.3  3200
##  7         7   85          175       1  80    3410
##  8         8   85          175       1  79.7  3290
##  9         9   85          175       1  79.8  3500
## 10        11   92.1        175       1  78.4  3360
## 11        12   77.9        175       1  75.6  3020
## 12        13   85          182.      1  78.5  3630
## 13        14   85          168.      1  77    3150
```

Esta vez, las ecuaciones para codificar los factores serán:

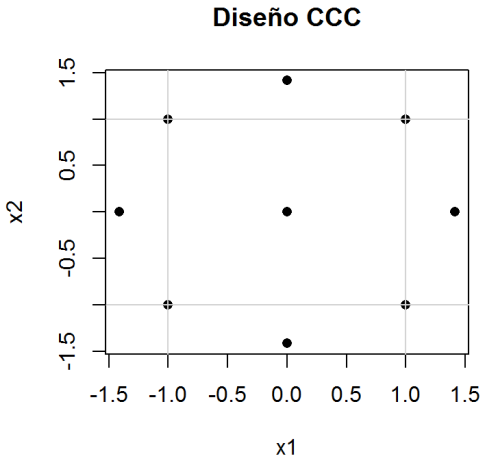
$$x_1 = \frac{\xi_1 - [max(\xi_1) + min(\xi_1)]/2}{max(\xi_1) - min(\xi_1)]/2} = \frac{\xi_1 - 90 + 80)/2}{(90 - 80)/2} = \frac{\xi_1 - 85}{5}$$

$$x_2 = \frac{\xi_2 - [max(\xi_2) + min(\xi_2)]/2}{max(\xi_2) - min(\xi_2)]/2} = \frac{\xi_2 - (180 + 170)/2}{(180 - 170)/2} = \frac{\xi_2 - 175}{5}$$

```
# Codificamos los factores
datos_ccc <- coded.data(data = datos_ccc,
                        x1 ~ (tiempo - 85)/5, x2 ~ (temperatura - 175)/5)

as.data.frame(datos_ccc)
```

```
##    run.order     x1     x2 bloque   y1   y2
## 1          1 -1.000 -1.000      1 76.5 2940
## 2          2 -1.000  1.000      1 77.0 3470
## 3          3  1.000 -1.000      1 78.0 3680
## 4          4  1.000  1.000      1 79.5 3890
## 5          5  0.000  0.000      1 79.9 3480
## 6          6  0.000  0.000      1 80.3 3200
## 7          7  0.000  0.000      1 80.0 3410
## 8          8  0.000  0.000      1 79.7 3290
## 9          9  0.000  0.000      1 79.8 3500
## 10        11  1.414  0.000      1 78.4 3360
## 11        12 -1.414  0.000      1 75.6 3020
## 12        13  0.000  1.414      1 78.5 3630
## 13        14  0.000 -1.414      1 77.0 3150
```

```
# Grafico del diseño
plot(datos_ccc[ , c(2:3)], pch = 16, main = "Diseño CCC")
abline(h = c(1, -1), col = "lightgrey")
abline(v = c(1, -1) , col = "lightgrey")
```



Nos disponemos a ajustar un modelo de segundo orden a los datos, centrándonos solo en la variable respuesta y_1:

```
# Modelo polinomico de segundo grado
modelo2.y1 <- rsm(y1 ~ SO(x1, x2), data = datos_ccc)

# (En caso de tener en cuenta el factor bloqueo, el codigo sería):
# modelo2 <- rsm(y1 ~ bloque + SO(x1, x2), data = datos_ccc)

# Resultado del ajuste
summary(modelo2.y1)
```

```
##
## Call:
## rsm(formula = y1 ~ SO(x1, x2), data = datos_ccc)
##
##              Estimate Std. Error  t value  Pr(>|t|)
```

```
## (Intercept) 79.939955   0.119089 671.2644 < 2.2e-16 ***
## x1           0.995050   0.094155  10.5682 1.484e-05 ***
## x2           0.515203   0.094155   5.4719  0.000934 ***
## x1:x2        0.250000   0.133145   1.8777  0.102519
## x1^2        -1.376449   0.100984 -13.6303 2.693e-06 ***
## x2^2        -1.001336   0.100984  -9.9158 2.262e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.9827, Adjusted R-squared:  0.9704
## F-statistic: 79.67 on 5 and 7 DF,  p-value: 5.147e-06
##
## Analysis of Variance Table
##
## Response: y1
##             Df  Sum Sq Mean Sq  F value    Pr(>F)
## FO(x1, x2)   2 10.0430  5.0215  70.8143 2.267e-05
## TWI(x1, x2)  1  0.2500  0.2500   3.5256    0.1025
## PQ(x1, x2)   2 17.9537  8.9769 126.5944 3.194e-06
## Residuals    7  0.4964  0.0709
## Lack of fit  3  0.2844  0.0948   1.7885    0.2886
## Pure error   4  0.2120  0.0530
##
## Stationary point of response surface:
##        x1        x2
## 0.3892304 0.3058466
##
## Stationary point in original units:
##      tiempo temperatura
##    86.94615   176.52923
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1] -0.9634986 -1.4142867
##
## $vectors
##          [,1]       [,2]
## x1 -0.2897174 -0.9571122
## x2 -0.9571122  0.2897174
```

En el *summary* del modelo podemos ver que tanto los estimadores de los parámetros de primer y segundo orden (β_1, β_2, β_{21} y β_{22}) son significativos (p-valor < 0,05) con un error estándar relativamente bajo, siendo el tiempo (x_1) el factor con un efecto más significativo sobre el rendimiento (mayor coeficiente absoluto de regresión). El efecto que no resulta significativo es el de la interacción entre factores β_{12} (p-valor = 0,102). Podríamos optar en este caso, a excluir del modelo este término de interacción no significativo, incluyendo solo los términos de primer orden (`FO()`) y cuadráticos puros (`PQ()`):

```
# Modelo sin interacciones
modelo2.y1.sinI <- rsm(y1 ~ FO(x1, x2) + PQ(x1, x2), data = datos_ccc)

# Resultado del ajuste
summary(modelo2.y1.sinI)
```

```
##
## Call:
## rsm(formula = y1 ~ FO(x1, x2) + PQ(x1, x2), data = datos_ccc)
##
##             Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 79.93995    0.13660 585.2155 < 2.2e-16 ***
## x1           0.99505    0.10800   9.2135 1.559e-05 ***
## x2           0.51520    0.10800   4.7704  0.001408 **
## x1^2        -1.37645    0.11583 -11.8831 2.310e-06 ***
## x2^2        -1.00134    0.11583  -8.6447 2.489e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.974,  Adjusted R-squared:  0.961
## F-statistic: 75.02 on 4 and 8 DF,  p-value: 2.226e-06
##
## Analysis of Variance Table
##
## Response: y1
##             Df  Sum Sq Mean Sq F value    Pr(>F)
## FO(x1, x2)   2 10.0430  5.0215 53.8227 2.290e-05
## PQ(x1, x2)   2 17.9537  8.9769 96.2186 2.538e-06
## Residuals    8  0.7464  0.0933
## Lack of fit  4  0.5344  0.1336  2.5206    0.1962
## Pure error   4  0.2120  0.0530
##
## Stationary point of response surface:
##        x1        x2
## 0.3614555 0.2572577
##
## Stationary point in original units:
##      tiempo temperatura
##    86.80728   176.28629
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1] -1.001336 -1.376449
##
## $vectors
##    [,1] [,2]
## x1    0   -1
## x2   -1    0
```

El ajuste del modelo en su conjunto sigue siendo significativo (F = 79,67, p-valor = 5,14e-6), y además presenta una alta bondad de ajuste (R^2_{adj}) = 0,97 y falta de ajuste no significativa (p-valor = 0,288), por lo que en conjunto la aproximación de la curvatura en la respuesta por parte este modelo es adecuada. Su ecuación, en unidades codificadas sería la siguiente:

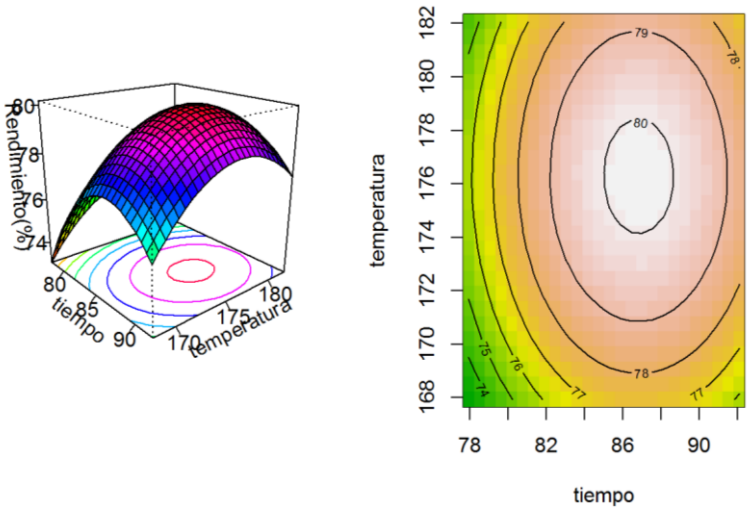
$$\hat{y}_1 = 79{,}939 + 0{,}995x_1 + 0{,}515x_2 - 1{,}376x_{21} - 1{,}001x_{22}$$

Esta vez, el *summary* nos ha proporcionado el análisis canónico, en lugar de las coordenadas de ascenso en máxima pendiente, al tratarse de un modelo polinómico de grado dos. Lo que nos indica son las coordenadas del punto óptimo o punto estacionario, tanto en unidades

codificadas como naturales. Si nos fijamos en los valores propios del análisis canónico (λ_1 = -1,001 y λ_2 = -1,376), ambos valores tienen signo negativo, por lo que el punto estacionario es un punto de respuesta máxima. Por tanto, la combinación de tiempo y temperatura que se estima que proporcione el rendimiento máximo es:

Tiempo (x_1) = 86,8 min; Temperatura (x_2) = 176,28ºF

```
# Superficie 3D
par(mfrow = c(1,2))
persp(modelo2.y1.sinI, x2 ~ x1,
      zlab = "Rendimiento(%)",
      contours = list(z = "bottom", col = "colors"), # posicion y color
      at = c(summary(modelo2.y1.sinI$canonical$xs)),
      theta = 50, # coordenadas graficas
      phi = 20)

# Grafico de contornos
contour(modelo2.y1.sinI, ~ x1 + x2, image = TRUE)
```



Con la función `predict()` podemos obtener la predicción del modelo sobre el rendimiento en este punto máximo:

```
# Valor predicho de rendimiento (%) en el punto estacionario
predict(modelo2.y1.sinI, coded.data(data.frame(tiempo = 86.8,
                                               temperatura = 176.28),
                                    x1 ~ (tiempo - 85)/5,
                                    x2 ~ (temperatura - 175)/5))
```

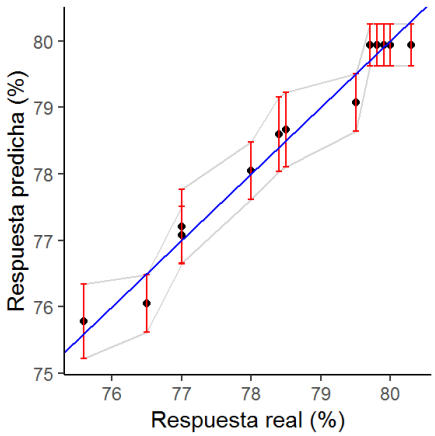
```
##        1
## 80.18605
```

Además, podemos comparar todos los valores reales de rendimiento obtenidos en el experimento con los predichos por el modelo, junto con el intervalo de confianza (*lwr*, *upr*):

```
# Calculamos las predicciones del modelo
pred.y1 <- predict(modelo2.y1.sinI, datos_ccc, interval = "confidence")

# Creamos un dataset con la variable respuesta real (y1) y la predicha (fit
)
# por el modelo, junto con el intervalo de confianza (lwr, upr)
library(dplyr)
predicciones <- data.frame(cbind(y1 = datos_ccc$y1, pred.y1))
predicciones <- mutate(predicciones, res = y1 - fit)
predicciones <- predicciones[ , c(1, 2, 5, 3, 4)]
predicciones
```

```
##       y1       fit          res       lwr       upr
## 1   76.5 76.05192  0.44808372 75.62054 76.48329
## 2   77.0 77.08232 -0.08232187 76.65095 77.51369
## 3   78.0 78.04202 -0.04201678 77.61064 78.47339
## 4   79.5 79.07242  0.42757763 78.64105 79.50379
## 5   79.9 79.93995 -0.03995461 79.62496 80.25495
## 6   80.3 79.93995  0.36004539 79.62496 80.25495
## 7   80.0 79.93995  0.06004539 79.62496 80.25495
## 8   79.7 79.93995 -0.23995461 79.62496 80.25495
## 9   79.8 79.93995 -0.13995461 79.62496 80.25495
## 10  78.4 78.59489 -0.19488847 78.03808 79.15170
## 11  75.6 75.78089 -0.18088636 75.22408 76.33770
## 12  78.5 78.66638 -0.16638417 78.10957 79.22319
## 13  77.0 77.20939 -0.20939066 76.65258 77.76620
```

```
# Grafico de predicciones vs valor real
library(ggplot2)
ggplot(predicciones, aes(x = y1, y = fit)) +
  geom_point() +
  geom_line(aes(y = lwr), col = "lightgrey")+
  geom_line(aes(y = upr), col = "lightgrey")+
  geom_errorbar(aes(ymin = lwr, ymax = upr), colour="red") +
  labs(x = "Respuesta real (%)", y = "Respuesta predicha (%)") +
  geom_abline(slope = 1, color = "blue") +
  theme_classic()
```



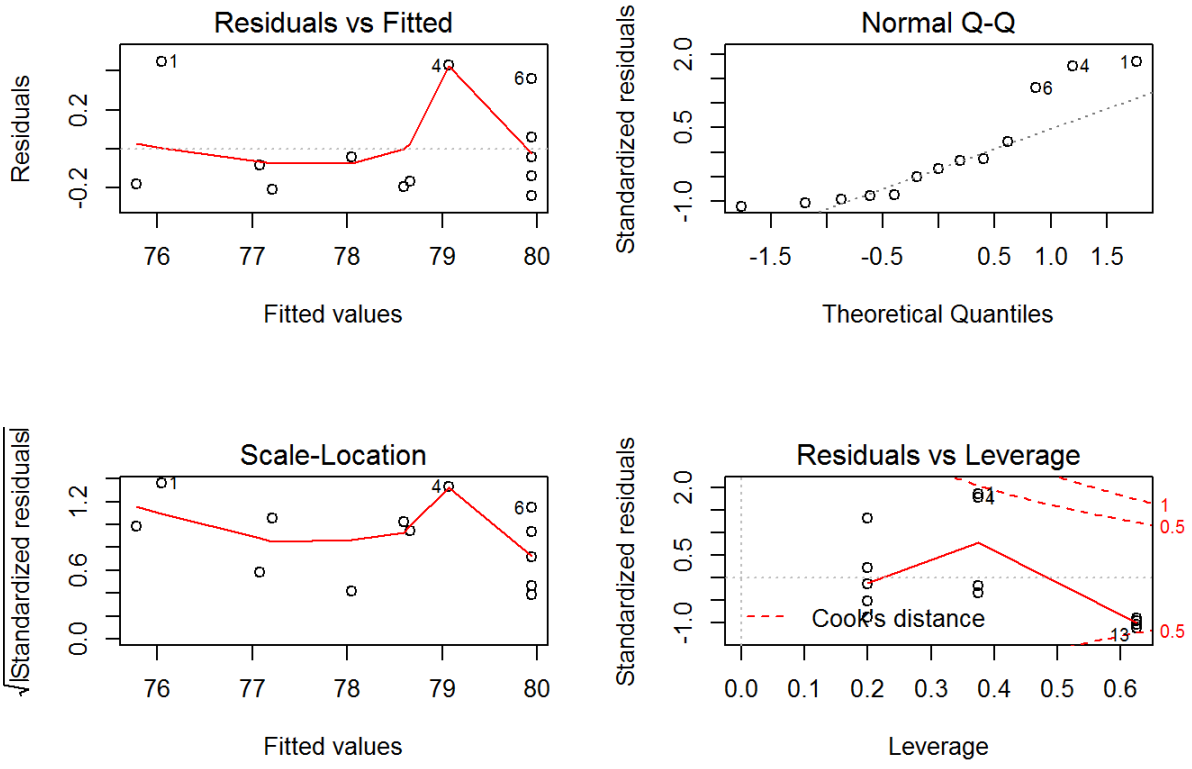
Como podemos ver, los residuos o diferencia entre los valores reales y predichos es pequeña.

Lo que habría que hacer como último paso es llevar a cabo un experimento confirmatorio con los valores de los factores obtenidos y corroborar que se obtiene una respuesta próxima a la estimada por el modelo (rendimiento (y_1) ≈ 80,18%).

Gráficos de diagnóstico

Un aspecto importante a la hora de validar la adecuación del modelo es también la de evaluar ciertas características como la normalidad de los residuos y su relación con respecto a los valores predichos, entre otros. Podemos obtener gráficos para analizar estos aspectos simplemente aplicando la función `plot()` al objeto que guarda el modelo:

```
par(mfrow = c(2, 2))

# Diagnostico grafico de los datos del modelo
plot(modelo2.y1.sinI)
```



El primero de los gráficos (*Residuals vs. Fitted*) muestra la relación entre los residuos (valor real - valor predicho) y los valores predichos. La distribución de los puntos debería ser más o menos aleatoria, lo cual se cumple. El segundo gráfico (*Normal Q-Q*) es un gráfico de normalidad de los residuos. Los puntos que no encajan en esta distribución aparecen alejados

de la diagonal. El tercer gráfico (*Scale-Location*) muestra si los residuos están distribuidos de manera equitativa a lo largo del rango de los predictores. También deben tener una distribución aproximadamente aleatoria, lo cual se cumple. El último gráfico (*Residuals vs Leverale*) muestra los puntos con mayor influencia (*leverage*) sobre la regresión. En este caso no exceden los límites marcados en rojo (*Cook's distance*), lo cual se trataría de casos influyentes que podrían alterar los resultados de la regresión en caso de excluirlos.

Análisis de múltiples respuestas

Considerar el análisis de múltiples respuestas supone ajustar un modelo adecuado para cada una de ellas con el fin de determinar la combinación de los valores de los factores que optimizan dichas respuestas, o que las mantienen en unos rangos deseados. En el ejemplo anterior solo analizamos el rendimiento (y_1), pero ahora tendremos también en cuenta la otra variable respuesta disponible (y_2: peso molecular).

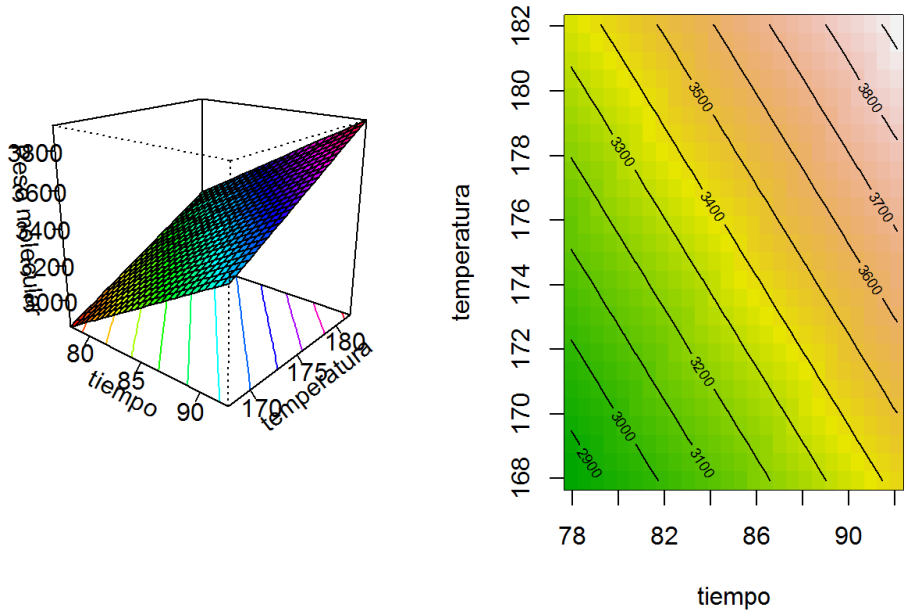
Comenzamos ajustando para y_2 un modelo lineal (un polinomio no se ajusta bien a esta variable), a partir del cual poder obtener su correspondiente gráfico de contornos:

```
# Modelo polinomico de primer grado para y2
modelo2.y2 <- rsm(y2 ~ FO(x1, x2), data = datos_ccc)

summary(modelo2.y2)
```

```
##
## Call:
## rsm(formula = y2 ~ FO(x1, x2), data = datos_ccc)
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 3386.154     45.936 73.7151 5.151e-15 ***
## x1           205.126     58.561  3.5028    0.0057 **
## x2           177.367     58.561  3.0287    0.0127 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.682,  Adjusted R-squared:  0.6184
## F-statistic: 10.72 on 2 and 10 DF,  p-value: 0.003254
##
## Analysis of Variance Table
##
## Response: y2
##             Df Sum Sq Mean Sq F value   Pr(>F)
## FO(x1, x2)   2 588196  294098  10.721 0.003254
## Residuals   10 274311   27431
## Lack of fit  6 208591   34765   2.116 0.244266
## Pure error   4  65720   16430
##
## Direction of steepest ascent (at radius 1):
##        x1        x2
```

```
## 0.7564352 0.6540687
##
## Corresponding increment in original units:
##      tiempo temperatura
##    3.782176    3.270343
```

```
# Superficie 3D
par(mfrow = c(1,2))
persp(modelo2.y2, x2 ~ x1,
      zlab = "Peso molecular",
      contours = list(z = "bottom", col = "colors"),
      at = c(summary(modelo2.y2$canonical$xs)),
      theta = 40,
      phi = 20)

# Grafico de contornos
contour(modelo2.y2, ~ x1 + x2, image = TRUE, main = "")
```



Supongamos que las condiciones que queremos que se cumplan para cada una de las respuestas es la siguiente:

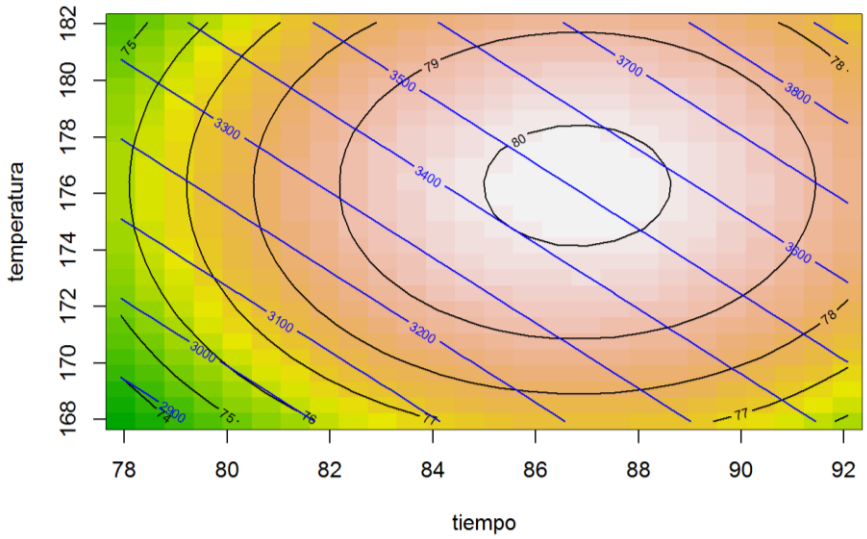
Maximizar: $y_1 \geq 77$ (valor deseable ≈ 80)

Minimizar: $y_2 \leq 3400$ (valor deseable ≈ 3200)

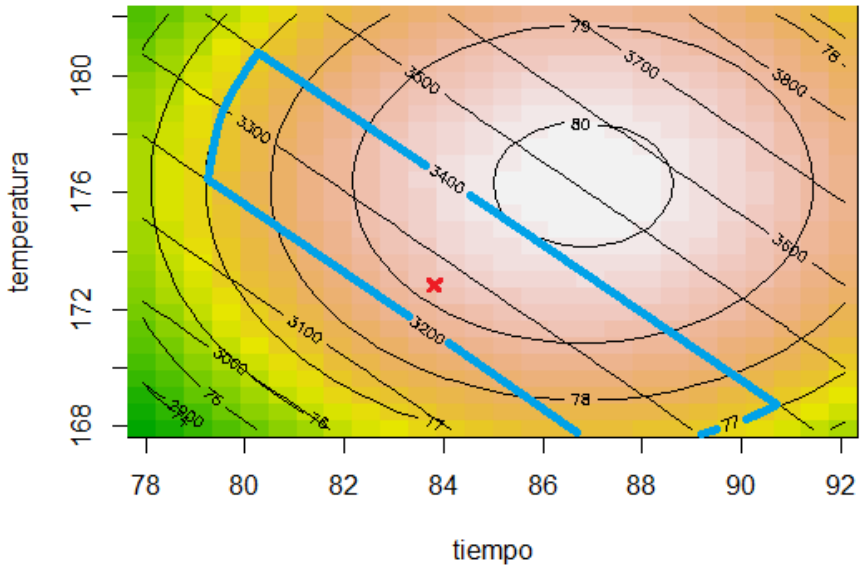
OPTIMIZACIÓN GRÁFICA

Una de las formas más simples de proceder es superponer los dos gráficos de contorno correspondientes a y_1 e y_2 para inspeccionar la zona del espacio factorial que cumpla con los requisitos, y de ella escoger posibles valores óptimos:

```
# Grafico de contornos de y1
contour(modelo2.y1.sinI, ~ x1 + x2, image = TRUE)

# Grafico de contornos de y2, superpuesto al anterior con "add = TRUE"
contour(modelo2.y2, ~ x1 + x2, image = FALSE, add = TRUE, col = "blue")
```



En base a la inspección visual del gráfico, y sabiendo que queremos maximizar y_1 mientras que y_2 sea menor a 3400, podríamos optar por escoger como punto dentro de la zona marcada en azul, en función del criterio del experimentador y las características del experimento:



Podemos ver que las posibilidades se encuentran entre mayor tiempo con menor temperatura, o menor tiempo con mayor temperatura. Una opción podría ser la marcada por la cruz roja: $\xi_1 \approx 84$ min, $\xi_2 \approx 173$ºF, el cual es un punto intermedio en el que no necesitaríamos mucho tiempo ni mucha temperatura.

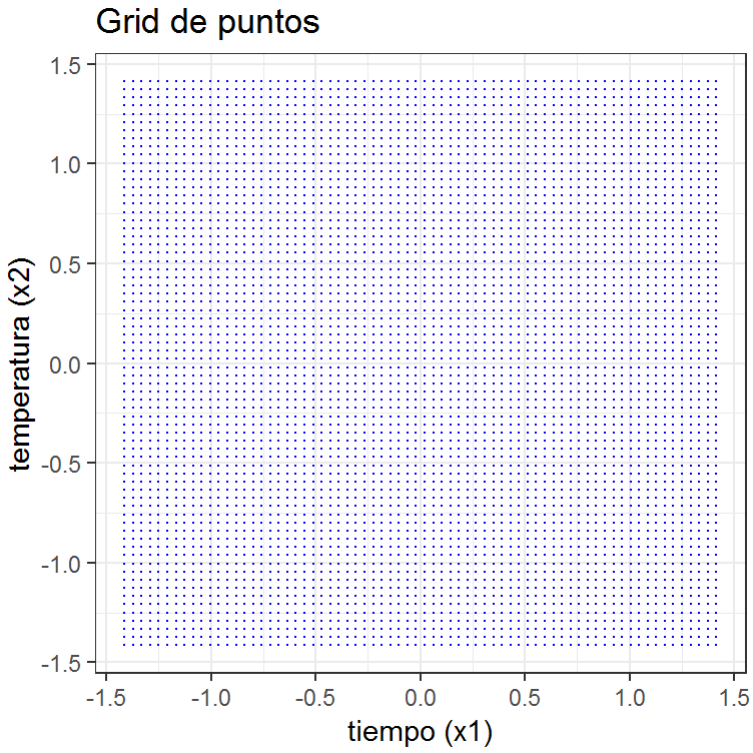
OPTIMIZACIÓN NUMÉRICA

Si queremos contar con una estimación más precisa de las condiciones óptimas de los factores, o si contamos con cuatro o más variables respuesta, es aconsejable recurrir al cálculo de la función de deseabilidad (*D*) sobre los valores predichos por los modelos. El paquete `rsm` no cuenta con esta funcionalidad, pero sí podemos utilizar herramientas útiles en el paquete `desirability`. A continuación, se explica paso a paso como calcular y obtener el máximo de la función DD para encontrar los valores óptimos de x_1 y x_2 manteniendo las mismas restricciones anteriores sobre y_1 e y_2:

```
library(desirability)
```

1. Creamos un nuevo conjunto de valores codificados (escogemos $n = 70$) comprendidos entre el mínimo y máximo axial de cada factor ([-1,4142, 1,4142]), y generamos todas las posibles combinaciones entre ambos (70 x 70 = 4900 valores). Estos nos servirán para obtener las predicciones de y_1 e y_2 a lo largo de todo el espacio muestral de los factores:

```
# Creamos el grid de valores
gridP <- expand.grid(x1 = seq(-1.4142, 1.4142, length = 70),
                     x2 = seq(-1.4142, 1.4142, length = 70))

ggplot(gridP, aes(x = x1, y = x2)) +
  geom_point(size = 0.05, color = "blue") +
  labs(x = "tiempo (x1)", y = "temperatura (x2)", title = "Grid de puntos")
 +
  theme_bw()
```

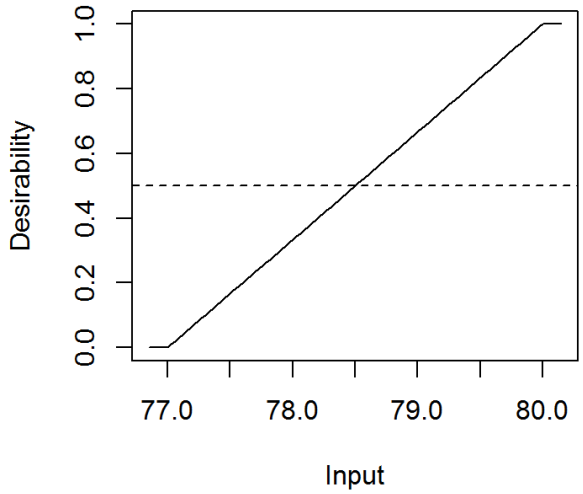



```
# Calculamos y añadimos las predicciones de los modelos para "y1" e "y2" so
bre
# los puntos creados
gridP$pred.y1 <- predict(modelo2.y1.sinI, gridP)
gridP$pred.y2 <- predict(modelo2.y2, gridP)

head(gridP)
```

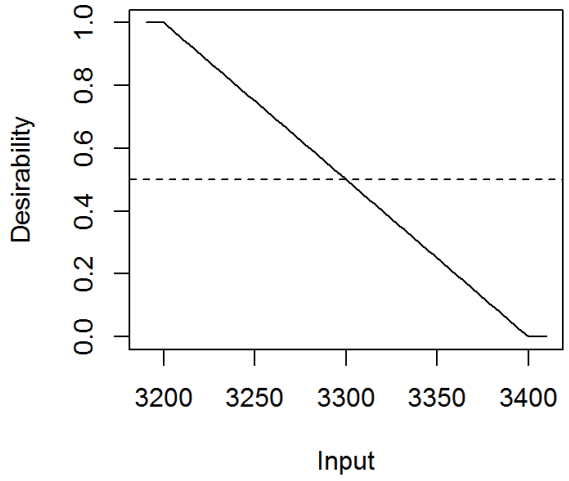
```
##          x1      x2  pred.y1  pred.y2
## 1 -1.414200 -1.4142 73.04868 2845.233
## 2 -1.373209 -1.4142 73.24674 2853.641
## 3 -1.332217 -1.4142 73.44017 2862.049
## 4 -1.291226 -1.4142 73.62898 2870.458
## 5 -1.250235 -1.4142 73.81316 2878.866
## 6 -1.209243 -1.4142 73.99272 2887.274
```

2. Creamos los objetos que contengan las funciones para el cálculo de deseabilidad individuales (d_1 y d_2) y la deseabilidad global D. Las deseabilidades individuales se definen con la función `dMax()` si se desea maximizar una respuesta, `dMin()` si se desea minimizar, o `dTarget()` si se desea mantener en un rango específico. Deberemos especificar el límite inferior L (`low`), superior U (`high`) y el valor de peso r (`scale`):

```r
# Definimos la funcion de deseabilidad individual para maximizar y1
d_y1 <- dMax(low = 77, high = 80, scale = 1)
plot(d_y1)
```




```r
# Definimos la funcion de deseabilidad individual para minimizar y2
d_y2 <- dMin(low = 3200, high = 3400, scale = 1)
plot(d_y2)
```



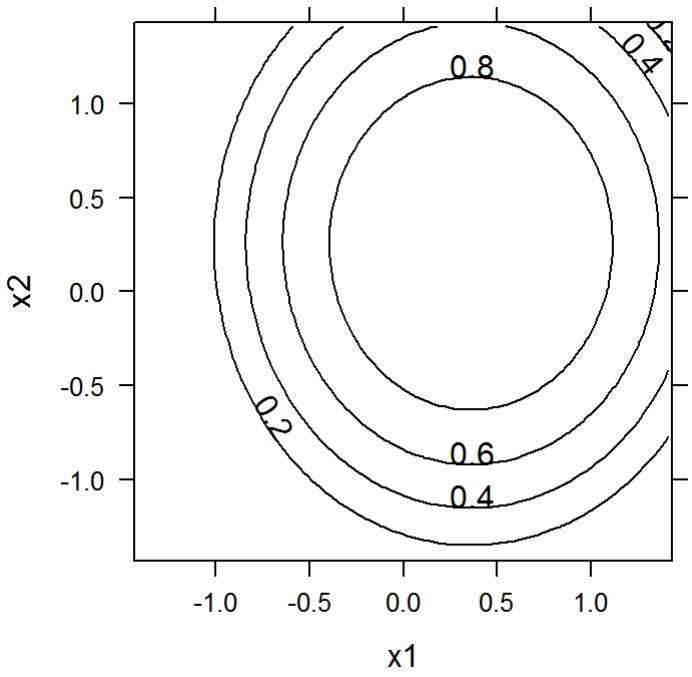

```r
# Definimos la funcion para calcular la deseabilidad global D
funcion_D <- dOverall(d_y1, d_y2)

# Calculamos D con la funcion predict()
D <- predict(funcion_D, gridP[, 3:4], all = TRUE)
```

```
# Unimos todos los datos obtenidos hasta ahora
gridP <- cbind(gridP, D)

head(gridP)
```

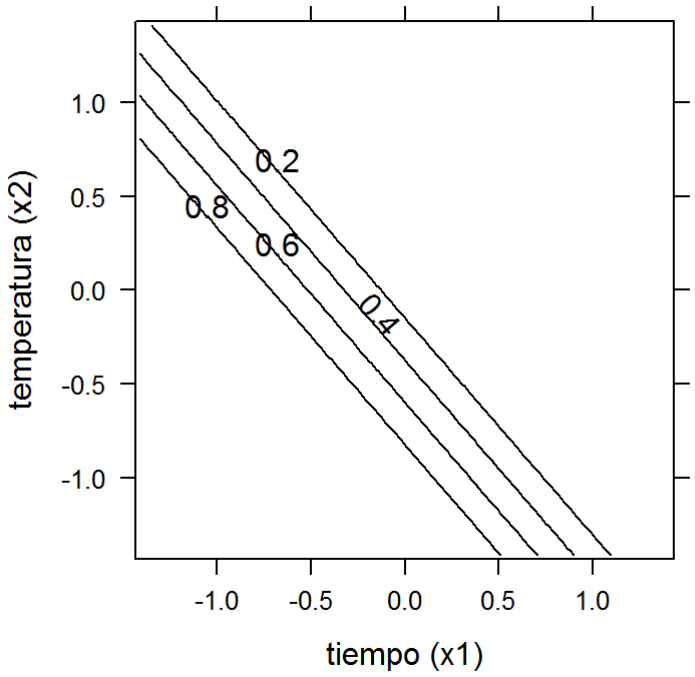
```
##          x1      x2  pred.y1  pred.y2 D1 D2 Overall
## 1 -1.414200 -1.4142 73.04868 2845.233  0  1       0
## 2 -1.373209 -1.4142 73.24674 2853.641  0  1       0
## 3 -1.332217 -1.4142 73.44017 2862.049  0  1       0
## 4 -1.291226 -1.4142 73.62898 2870.458  0  1       0
## 5 -1.250235 -1.4142 73.81316 2878.866  0  1       0
## 6 -1.209243 -1.4142 73.99272 2887.274  0  1       0
```

Acabamos obteniendo 4900 combinaciones de factores, la predicción correspondiente de ambas variables respuesta y los respectivos valores de deseabilidad individual y la global (*Overall*), que será la que utilizemos para encontrar la combinación óptima de factores. Podemos observar el aspecto del gráfico de contornos de las deseabilidades individuales d_1 y d_2:

```
library(lattice)

# Grafico deseabilidad individual para y1
contourplot(D1 ~ x1 + x2, gridP, aspect = 1, as.table = TRUE)
```

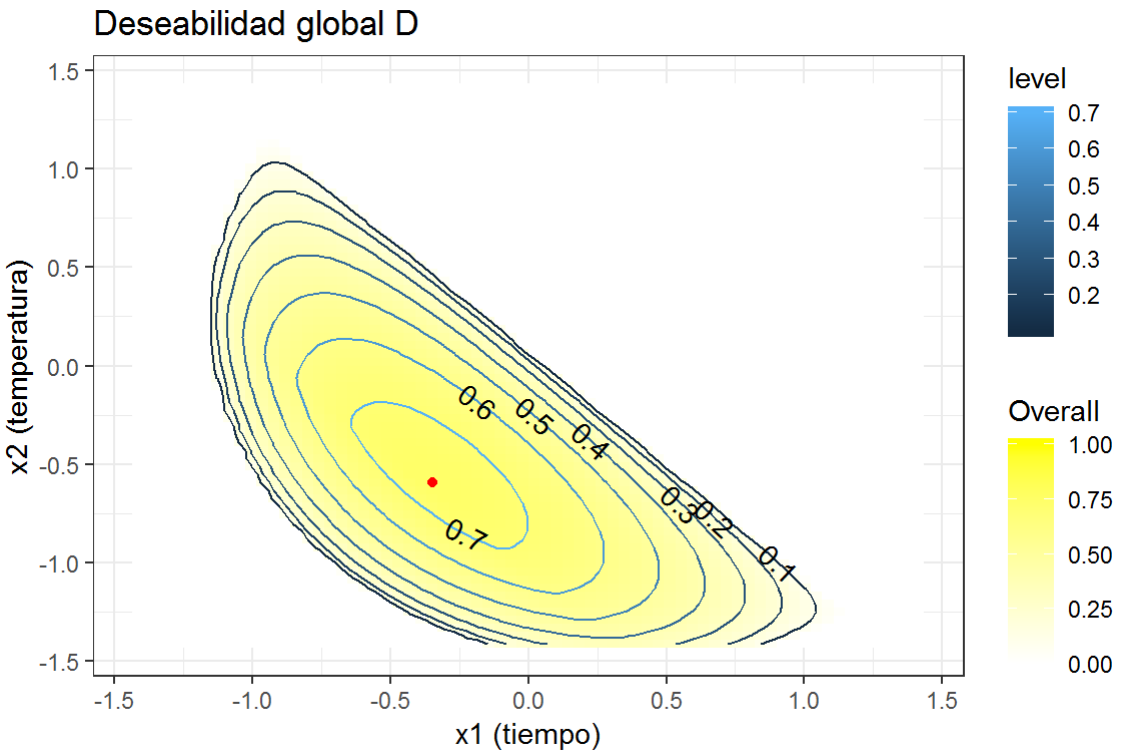


```
# Grafico deseabilidad individual para y2
contourplot(D2 ~ x1 + x2, gridP,
            aspect = 1,
            as.table = TRUE,
            xlab = "tiempo (x1)",
            ylab = "temperatura (x2)")
```



3. Por último, producimos el gráfico de contornos de la deseabilidad global *D*, resultado de la combinación de d_1 y d_2, y extraemos de los 4900 valores, el que mayor *D* presenta (marcado con un punto rojo en el gráfico):

```
# Grafico de contornos de la deseabilidad global D
library(ggplot2)
library(metR)

ggplot(gridP, aes(x = x1, y = x2, z = Overall)) +
  geom_tile(aes(fill = Overall))+
  stat_contour(aes(colour = stat(level)))+
  geom_text_contour() +
  geom_point(
    aes(x = gridP[which(gridP$Overall == max(gridP$Overall)), ]$x1,
        y = gridP[which(gridP$Overall == max(gridP$Overall)), ]$x2),
    color = "red")+
  labs(x = "x1 (tiempo)",
       y = "x2 (temperatura)",
       title = "Deseabilidad global D") +
  scale_fill_continuous(low="white", high="yellow", limits=c(0,1)) +
  theme_bw()
```




```
# Condiciones con valor máximo de D
gridP[which(gridP$Overall == max(gridP$Overall)), ]
```

```
##              x1         x2  pred.y1 pred.y2        D1        D2   Overall
## 1427 -0.3484261 -0.5943739 78.76618 3209.26 0.5887252 0.9536979 0.749310
```

Según este método, las condiciones que maximizan el rendimiento (y_1) y minimizan el peso molecular (y_2), dentro de las restricciones establecidas, sería el par de factores con mayor *D* (*Overall* = 0,7493):

x_1 = -0,3484
x_2 = -0,5943

o lo que es lo mismo:

$x_1 = (\xi_1 - 35)/5$
$x_2 = (\xi_2 - 155)/5$

Tiempo = 83,25 min
Temperatura = 172,02ºF

El rendimiento estimado en este punto es de aproximadamente 78,77% con un peso molecular aproximado de 3209,26 Mn, por lo que se cumplen nuestras restricciones. Recordar que se debe validar diseñando un experimento con estas condiciones para confirmar que da unos valores de rendimiento y peso molecular próximos a los estimados.

R Commander

R-Commander (https://www.rcommander.com/) es una Interfaz Gráfica de Usuario (GUI) gratuita que permite utilizar el lenguaje de programación R sin necesidad de programar. El uso es a base de clicks, pero también aparece el código correspondiente a cada acción en la pestaña *R Script*. Para el diseño de experimentos y su análisis básico puede utilizarse el plugin "RcmdrPlugin.DoE": (https://www.rdocumentation.org/packages/RcmdrPlugin.DoE/versions/0.12-3).

Tanto R-Commander como el plugin son instalables como cualquier otro paquete con el comando:

```
install.packages("Rcmdr")
install.packages("RcmdrPlugin.DoE")
```

Para iniciar R-Commander una vez iniciado R o RStudio, basta con el comando:

```
library(Rcmdr)

# Si se cierra y se quiere volver a abrir dentro de una misma sesion abiert
a de R, utilizar:
Commander()
```

Una vez abierta la ventana de R-Commander, hay que cargar el plugin, dirigiéndose a la opción de la barra de menú "*Tools -> Load Rcmdr plug-in(s) -> OK*".

- Para crear un diseño, dirigirse a la opción "*Design -> Create design*".
- Para obtener el modelo: "*Design -> Analyze Design -> Response surface model*".
- Para obtener el gráfico del modelo: "*Design -> Analyze Design -> Response surface plots*".
- Para realizar un análisis de varianza del modelo (ANOVA): "*Models -> Hypothesis tests -> ANOVA table*".

BIBLIOGRAFÍA

Douglas C. Montgomery. Design and Analysis of Experiments, 8th Edition (2012).

Raymond H. Myers, Douglas C. Montgomery, Christine M. Anderson-Cook. Response surface methodology : process and product optimization using designed experiments, 3rd Edition.

https://www.itl.nist.gov/div898/handbook/pri/section3/pri3364.htm

https://cran.r-project.org/web/packages/rsm/rsm.pdf

https://cran.r-project.org/web/packages/desirability/vignettes/desirability.pdf