Facultad Latinoamericana de Ciencias Sociales, FLACSO Ecuador
Departamento de Desarrollo, Ambiente y Territorio
Convocatoria 2017-2019

Tesis para obtener el título de maestría de Investigación en Economía del Desarrollo

Propuesta metodológica para la identificación de supermodularidad

Fausto Damián Jácome Pérez

Asesor: Wilson Pérez
Lectores: Jhon Cajas Guijarro y Juan Fernández Sastre

Quito, junio de 2022

Índice de contenidos

Lista de ilustraciones

Figuras

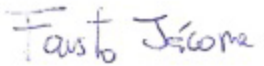
Tablas

Declaración de cesión de derecho de publicación de la tesis

Yo, Fausto Damián Jácome Pérez, autor de la tesis titulada "Propuesta metodológica para la identificación de supermodularidad" declaro que la obra es de mi exclusiva autoría, que la he elaborado para obtener el título de maestría de Investigación en Economía del Desarrollo concedido por la Facultad Latinoamericana de Ciencias Sociales, FLACSO Ecuador.

Cedo a la FLACSO Ecuador los derechos exclusivos de reproducción, comunicación pública, distribución y divulgación, bajo la licencia Creative Commons 3.0 Ecuador (CC BY-NC-ND 3.0 EC), para que esta universidad la publique en su repositorio institucional, siempre y cuando el objetivo no sea obtener un beneficio económico.

Quito, junio de 2022



Fausto Damián Jácome Pérez

Resumen

Se propone una metodología para la identificación de complementariedad con el uso del concepto de funciones supermodulares en *lattices* aplicada a datos estadísticos. Se proponen dos algoritmos cuyos resultados se enfrentan al test estándar usado para la medición de complementariedad con variables continuas que exige que las derivadas cruzadas de todas las variables sean no negativas. Los resultados muestran que los algoritmos propuestos son iguales o superiores al test estándar en cuanto a exhaustividad y score F1 pero con un gran costo en tiempo de computación, lo que abre la puerta a posteriores desarrollos sobre esta base.

Agradecimientos

A Wilson Pérez por las ideas y el apoyo en estos dos años.

Introducción

Este documento propone una metodología para la identificación de complementariedad con el uso del concepto de funciones supermodulares en *lattices* aplicada a datos estadísticos.

El concepto de supermodularidad es introducido por Donald Topkis en 1978 para referirse a funciones con diferencias crecientes o complementarias en el contexto de la optimización para estática comparativa. En este trabajo se utiliza el concepto de funciones supermodulares para verificar, a partir de un set de datos, si las relaciones entre las variables constituyen una función supermodular.

La aplicación de complementariedad se encuentra en diversos campos de estudio como el comportamiento del consumidor, teoría de monopolios, competición Cournot y Bertrand, modelos de investigación y desarrollo, búsqueda, matching, y teoría del crecimiento (Amir 2005).

En este trabajo se proponen dos algoritmos cuyos resultados se enfrentan al test estándar usado para la medición de complementariedad con variables continuas que exige que las derivadas cruzadas de todas las variables sean no negativas cal- culadas a partir del coeficiente de interacción entre las variables de la función (Carree, Lokshin, y Belderbos 2011).

Los resultados muestran que los algoritmos propuestos son iguales o superiores al test estándar en cuanto a precisión y exhaustividad lo que abre la puerta a posteriores desarrollos sobre esta base.

El documento se divide de la siguiente manera: en el capítulo 1 se definen las funciones supermodulares y sus aplicaciones, en el capítulo 2 se explica el problema econométrico y el desarrollo de los test paso a paso, el algoritmo para las simulaciones y el método de evaluación del rendimiento entre test y los resulta- dos. Finalmente, en la última sección, se detallan las conclusiones y próximos pasos.

Capítulo 1: Marco teórico

1.1. Funciones supermodulares

En el presente capítulo se caracterizan las funciones modulares. Todas las definiciones presentadas en esta sección, a menos que se especifique lo contrario, se toman del libro *Supermodularity y Complementarity* de Topkis 1998. Previo a la definición de supermodularidad, se introducen a continuación algunos conceptos necesarios:

Un conjunto parcialmente ordenado es un conjunto X en el cual, hay una relación binaria $\preccurlyeq$ que es reflexiva, antisimétrica y transitiva. El conjunto $R^n = \{x = (x_1, \dots, x_n): x_i \in R^1$ para $i = 1, \dots, n\}$ con la relación de orden donde $x' \leq x''$ en R^n si $x_i' \leq x_i''$ en R^1 para $i = 1, \dots, n$ es un conjunto parcialmente ordenado. Si X es un conjunto parcialmente ordenado y X' es un subconjunto de X. Si $x' \in X$ y $x \preccurlyeq x' (x \succcurlyeq x')$ para cada x en X', entonces x' es una cota superior (inferior) para X'.

Si x' en X' es una cota superior (inferior) para X', entonces x' es el elemento mayor (menor) de X'. Si x' está en X' y no existe otro x'' en X'' que cumpla que $x' \prec x'' (x' \succ x'')$, entonces x' es el elemento máximo (mínimo) de X'. Si el conjunto de cotas superiores (inferiores) de X' tiene un elemento menor (mayor), entonces en la menor cota superior (mayor cota inferior) de X' es el supremo (ínfimo) de X' denotado por $sup_x(X')(inf_x(X'))$. Si dos elementos, x' y x'', de un conjunto parcialmente ordenado X tienen una menor cota superior (mayor cota inferior) en X, es su unión (intersección) y se denota $x' \vee x'' (x' \wedge x'')$. Un conjunto parcialmente ordenado que contiene unión e intersección de cada par de elementos es una *lattice*. Para cualquier entero positivo n, R^n es una *lattice* con $x' \vee x'' = \sup(x', x'') = (x_1' \vee x_1'', \dots, x_n' \vee x_n'')$ y $x' \wedge x'' = \inf(x', x'') = (x_1' \wedge x_1'', \dots, x_n' \wedge x_n'')$ para x' y x'' en R^n.

Si X' es un subconjunto de una *lattice* X y X' contiene la intersección y unión, con respecto a X, de cada par de elementos de X', entonces X' es una *sublattice* de X. Por tanto, si un subconjunto X de R^n, $x' = (X_1', \dots, x_n') \in X$ y $x'' = (X_1'', \dots, x_n'') \in X$ implica que $(max\{x_1', x_1''\}, \dots, max\{x_n', x_n''\})$ y $(min\{x_1', x_1''\}, \dots, min\{x_n', x_n''\})$ están en X, entonces X es una *sublattice* de R^n.

Si X y T son conjuntos parcialmente ordenados y que $f(x, t)$ es una función real dentro del subconjunto S de $X \times T$. Para t en T, S_t denota la sección de S en t. Si $f(x, t'') - f(x, t')$ es

creciente (decreciente, estrictamente creciente o estrictamente decreciente) en x para $S_{t'} \cap S_{t''}$ para todo $t' \prec t''$ en T, entonces $f(x,t)$ tendrá diferencias crecientes (decrecientes, estrictamente crecientes, estrictamente decrecientes) en (x,t) de S. Si $f(x)$ es doblemente diferenciable en R^n, entonces $f(x)$ tiene diferencias crecientes en R^n si y solo si $\partial^2 f(x) \backslash \partial x_{i'}\, \partial x_{i''} \geq 0$ es creciente en $x_{i''}$ para toda i', i'' y todas las x.

Por ejemplo, si a $f(x)$, siendo una función de utilidad (o menos la función de costos), y x corresponde a n productos tal que $x = x_1, \ldots, x_n$ y se realiza un incremento en $\epsilon > 0$ a un producto i, $f\left(x + \epsilon u^i\right) - f(x)$ será la utilidad neta adicional del incremento el x_i. La $f(x)$ tendrá diferencias crecientes si y solo si, el neto de $f(x)$ para cualquier valor dado de ϵ en cualquier i' es siempre creciente en el nivel de cualquier otro producto i'', es decir que siempre será deseado más de i' a mayor monto disponible de cualquier otro producto i'', lo que se aplica a productos, actividades o en general, variables de decisión o parámetros. Así, este conjunto de productos son complementos y cada par de productos es complementarios entre sí, si evaluados en una función de utilidad tienen diferencias crecientes.

Si $f(x)$ es una función real en la *lattice* X y para todo x' y x'' en X se cumple que

$$f(x') + f(x'') \leq f(x' \vee x'') + f(x' \wedge x'') \qquad (1)$$

$f(x)$ es supermodular en X.

La transformación de funciones supermodulares cumple con las siguientes propiedades. Suponiendo que X es una *lattice*.

a. Si $f(x)$ es supermodular en X y $\alpha > 0$ entonces $\alpha f(x)$ es supermodular en X.
b. Si $f(x)$ y $g(x)$ son supermodulares en X, entonces $f(x) + g(x)$ es supermodular en X.
c. Si $f_k(x)$ es supermodular en X para $k = 1,2, \ldots$ y $\lim_{k\to\infty} f_k(x) = f(x)$ para todo x en X, entonces $f(x)$ es supermodular en X.

1.2. Uso de las funciones supermodulares y la complementariedad

La motivación de Topkis nace de la necesidad de dotar al estudio de la complementariedad de herramientas que incluyen supermodularidad en lattices y enfocarse en análisis y problemas relacionados a la estática comparativa monótona. En problemas de optimización, cambios en los parámetros afectan la función objetivo, en juegos no cooperativos, puede afectar la función de pagos de los jugadores y las estrategias posibles y en juegos cooperativos puede afectar la función característica y al conjunto de participantes (Topkis 1998).

A continuación, se presenta el caso para escalares que muestra Amir 2005, p. 638 del Teorema de Monotonicidad de Topkis, donde los parámetros y conjuntos de decisión son subconjunto de los reales.

Topkis consideró la siguiente familia parametrizada de problemas de optimización con restricciones, donde $A_s \subset A$, con el fin de obtener las condiciones suficientes sobre el objetivo y el conjunto de restricción que producen una solución óptima monótona:

$$a^*(s) = \arg\max\{F(s,a): a \in A_s\} \qquad (2)$$

Los conjuntos de parámetros y acciones, S y A son subconjuntos de R y A_s una correspondencia de S a A, con A_s como el conjunto de acciones posibles cuando el parámetro es s.

Dado el problema de optimización, el teorema de Topkis sostiene que, con $S, A \subset R$ y bajo el supuesto de que:

a. F tiene diferencias crecientes en (s,a).
b. $A_s = [g(s), h(s)]$, donde $h, g : S \to R$ son funciones crecientes que cumplen $g \leq h$.

Entonces, la selección óptima $a^*(s)$ será una función creciente.

1.3. Importancia de la complementariedad en la economía

Para definir la importancia de la complementariedad en la economía, Topkis 1998 apunta el contraste en la opinión de Samuelson cuando en 1947 dijo que en su opinión el problema de la complementariedad había recibido más atención que se merece por su importancia intrínseca; sin embargo, en 1974 el mismo Samuelson afirmó que era el momento de una mirada fresca y moderna al concepto de complementariedad aludiendo a que las cosas más simples suelen ser las más complicadas de entender completamente. Con esas palabras Samuelson reconocía la necesidad del estudio de la complementariedad.

Milgrom y Roberts 1995 hablan de la noción de complementariedad de Edgeworth que define a las actividades complementarias como aquellas que, si se realiza una (o más) de ellas, incrementa el retorno de hacer una o más de las otras, que en sus palabras y en el contexto que Edgeworth lo emplea, se traduce a unas derivadas parciales cruzadas positivas de alguna función de pagos, lo que asume divisibilidad de las variables de elección y diferenciabilidad de la función. Es por eso que, con el Teorema de Monotonicidad de Topkis, de la sección 1.2, se pueden simplificar los supuestos, en palabras de Amir 2005, el Teorema de Topkis puede prescindir de los requisitos de concavidad porque, si estos fallan localmente, el argumento máximo de la función estará en el límite y, por lo tanto, aumentará en el parámetro por la condición b del teorema descrito en la sección previa.

A continuación, se exponen algunas aplicaciones económicas desde la teoría de Topkis que se encuentran en Amir 2005. En un monopolio, si deseo ver si el precio óptimo una función creciente del costo, se define el beneficio como,

$$\Pi(p, c) = (p - c)D(p), \quad p \in [c, \infty)$$

Donde $p \in [c, \infty)$ es el precio y c el costo unitario, $D(p)$ es la función de demanda directa. Dado que $p \in [c, \infty)$ se cumple el supuesto b del Teorema de Monotonicidad de Topkis y dado que $\frac{\partial^2 \Pi(p,c)}{\partial p \partial c} = -D'(p) \geq 0$ cumple con supermodularidad, por lo que se puede concluir que el precio óptimo es creciente en el costo unitario.

Otra aplicación se da en el emparejamiento selectivo, el modelo de Becker de matrimonio, donde n hombres y n mujeres buscan formar n parejas. Cada sexo se ordena por productividad en un orden $\{1,2 \dots n\}$, donde 1 es la persona menos productiva y si una mujer i y un hombre j se emparejan, producen un excedente $f(i, j)$ como pareja. Si se quiere maximizar el excedente social $\sum f(i, j)$, en el caso de que $f(i, j)$ tenga diferencias

estrictamente crecientes se hará un emparejamiento selectivo con la configuración de emparejamiento que maximiza el excedente sería $(1,1), (2,2) \ldots (n,n)$. Por el contrario, se formarían dos parejas (i,j) y (i',j') tal que $i' > i$ y $j' < j$, dada la supermodularidad se debería sostener que $f(i',j) + f(i,j') > f(i',j') + f(i,j)$, lo que implica que las parejas propuestas (i,j) y (i',j') generarán un excedente inferior al óptimo.

1.4. Métodos de medición de complementariedad

Existe un consenso sobre la medición de complementariedad cuando se trata con variables continuas, por ejemplo, Leiponen 2005, Miravete y Pernias 2006, Carree, Lokshin y Belderbos 2011, Belderbos, Carree y Lokshin 2006, mencionan que, cuando las variables independientes son medidas como continuas, la complementariedad implica que todas las derivadas parciales cruzadas de la función f con respecto a las variables independientes deben ser positivas.

El tema de la complementariedad gana interés en el campo de las firmas, que en los casos discretos implican la toma de decisiones, como investigación con proveedores, universidades, demandantes o competencia, o el reemplazo de la investigación interna por compra externa, o la relación entre habilidades, innovación y colaboración investigación y desarrollo. Por lo que es importante investigar las relaciones entre variables para permitir una toma de decisiones óptima. Desde esta perspectiva, se utilizan únicamente variables dicotómicas que informan sobre si se utiliza o no cierta actividad dentro de la firma para evitar hacer supuestos sobre la forma funcional. Por tanto, la aproximación más habitual, presentada por Cassiman y Veugelers 2006, para el caso de 2 actividades de la firma, puede resumirse de la siguiente forma:

$$\Pi^{\mathrm{i}}\left(A_1^i, A_2^i, X^i; \theta; \beta\right) = \left(1 - A_1^i\right)\left(1 - A_2^i\right)\theta_{00} + A_1^i\left(1 - A_2^i\right)\theta_{10} + \left(1 - A_1^i\right)A_2^i\theta_{01} + A_1^i A_2^i\theta_{11} + X^i\beta + \varepsilon^{\mathrm{i}}$$

Donde i corresponde a la firma i y $A_j^i \in \{0,1\}$ para $j = 1,2$ que indica si la actividad de innovación es realizada por la firma. θ son los coeficientes producto de las acciones de la firma y X^i corresponde a las variables de control. Así, el test de complementariedad entre dos actividades dentro de la firma se calcula con la siguiente condición:

$$\theta_{11} + \theta_{00} \geq \theta_{10} + \theta_{01}$$

Lo que implica que existiría complementariedad en las dos actividades de la firma en el caso que el ejercer ambas actividades de forma simultánea da más beneficios que la suma de ejercer ambas actividades por separado[1].

Cassiman y Veugelers 2006 además propone un método adicional para testear la existencia de complementariedad con variables dicotómicas en el que supone que las covariables no son independientes de las actividades lo que generaría un sesgo en la estimación de los coeficientes y por tanto en el resultado del test de hipótesis de complementariedad. Para este caso Cassiman y Veugelers 2006 utiliza una regresión probit bivariada para tomar en cuenta la correlación entre variables de control que se asumen exógenas (Z^i) y las actividades, y describe el siguiente modelo:

$$A_1^{i^*} = Z^i\gamma_1 + v_1^i, \qquad A_1^i = 1\ si\ A_1^{i^*} > 0, \qquad 0\ caso\ contrario$$

$$A_2^{i^*} = Z^i\gamma_2 + v_2^i, \qquad A_2^i = 1\ si\ A_2^{i^*} > 0, \qquad 0\ caso\ contrario$$

$$E[v_1] = E[v_2] = 0, \qquad Var[v_1] = Var[v_2] = 1$$

$$Cov[v_1, v_2] = \rho$$

Y utilizan la correlación resultante como una prueba débil de complementariedad.

Como propuesta final realizan una predicción con el mismo modelo probit bivariado y lo utilizan como variables instrumentales en la regresión original.

[1] Nótese que θ_{00} no implica la ausencia de actividades, sino la ausencia de actividades $j = 1,2$, por lo se puede entender como un valor medio base de las firmas sin las actividades. En la parte derecha de la desigualdad en realidad se suma el valor base 2 veces más las actividades 1 y 2 por separado, mientras que en el izquierdo se suma el valor base más las actividades 1 y 2 más el valor base.

Capítulo 2: Problema econométrico

La estimación de supermodularidad de la función $f(x)$ bajo incertidumbre conlleva los dos retos principalmente:

a. Selección de la forma funcional de la estimación que permita la parametrización de funciones $f(x)$ no lineales con el objetivo de priorizar el ajuste.
b. Estimación de un incremento (Δx) mínimo en x tal que la probabilidad de que $f(x)$ sea supermodular supere un umbral p.[2]

Para abordar el problema planteado, se quiere determinar si $f(x)$ es supermodular. Sin embargo, lo que se tiene es un conjunto de observaciones (y_i^1, x_i^j) donde $i = 1, \dots, n$ y $j = 1, \dots, k$, lo que permite la estimación de $\hat{f}(x)$, tal que,

$$y_i = \hat{f}\left(x_i^1, \dots, x_i^k\right) + \mu_i \tag{3}$$

Sea $(x', x'') \in X$, siendo X en R^k una *lattice*, y

$$H(x', x'') = f(x' \vee x'') + f(x' \wedge x'') - f(x') - f(x'')$$

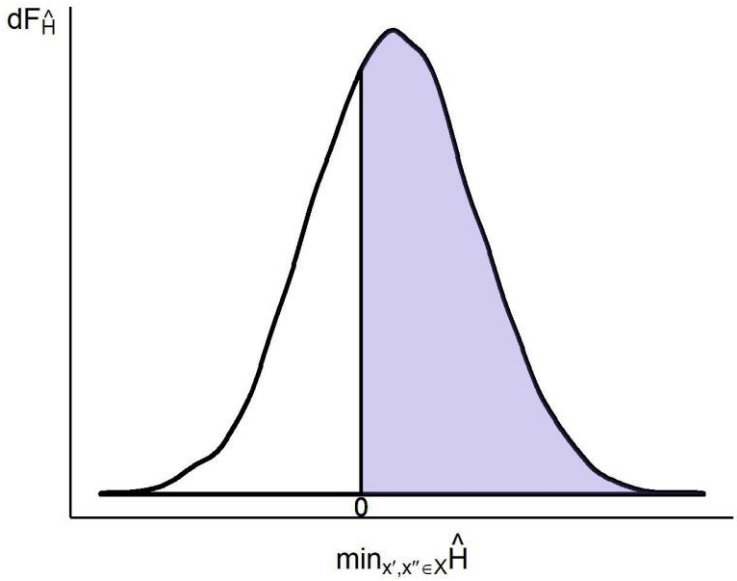
Por definición, si $H(x', x'') \geq 0, \forall (x', x'') \in X \leftrightarrow f(x)$ es supermodular en X. Sin embargo, se conoce $\hat{f}(x)$ y no $f(x)$, por lo tanto se estima

$$1 - F_{\widehat{H}}(0) = P\left(\widehat{H}(x', x'') \geq 0, \forall (x', x'') \in X\right)$$

$$= P\left(\min_{x', x'' \in X} \widehat{H}\,(x', x'') \geq 0\right) \tag{4}$$

[2] En la ecuación (1) si $x' - x'' \to \vec{0}$ implica que $f(x') + f(x'') = f(x' \vee x'') + f(x' \wedge x'')$ lo que en un contexto de incertidumbre supone la imposibilidad de obtener un resultado mediante el algoritmo planteado.

Figura 2.1. Función de densidad de $\min_{x',x''\in X} \widehat{H}$



Fuente: Trabajo investigativo

En la **Figura 2.1**, se muestra la densidad de la distribución de $\min_{x',x''\in X} \widehat{H}$, el área sombreada mide la probabilidad de que la función $f(x)$ sea supermodular.

Capítulo 3: Marco Metodológico

En base a lo discutido en la subsección 2.1 se han desarrollado dos test, ambos que utilizan los mismos argumentos de la función: base de datos, variable independiente, variables independientes, número de muestras y número de cortes que conforman la *lattice*.

3.1. Base de datos de test

Para la creación de las bases de datos de prueba se construyó una función en R que genera escenarios para el test (f_fge). En dicha función se definen el número de observaciones de la base de datos de test, las variables independientes y la función que construye la variable independiente $R^n \rightarrow R^1$ y las distribuciones aleatorias y sus parámetros.

```
#### Función generadora de escenarios ####

f_fge <- function(obs=100,
                  nomb=c('x1','x2','u'),
                  meth=c('rnorm'),
                  par=c('m=0,sd=1'),
                  func='y=x1*x2+u'
){
  # Se extrae de la función la variable independiente
  nomb_y <-
    (func %>%
       parse_expr(.) %>%
       as.character())[2]
  # Se extrae de la función
  fun_y <-
    (func %>%
       parse_expr(.) %>%
       as.character())[3]


  paste0('tibble(seq_id=1:',
         obs,
         ') %>% group_by(seq_id) %>% mutate(',
         paste0(
           nomb,'=',
           meth,
           '(',
           par,
           ',n=1)'
           ,collapse = ', '),
         ') %>% ungroup %>% mutate(',
         nomb_y,
         '=',
         fun_y,
         ')'
  ) %>%
    parse_expr %>%
    eval

}
```

3.2. Número de cortes de la *lattice*

El número de rectángulos en una matriz de $n \times n$ cuadrados viene dado por la serie A000537 en OEIS Foundation Inc. 2020 que sigue la fórmula del cuadrado de los números triangulares:

$$\sum_{m=1}^{N} m^3 = \left(\sum_{m=1}^{N} m\right)^2$$

Si se extiende la fórmula de la serie a d dimensiones con $k = m + 1$ vértices que forman las aristas de los N_d hiperrectángulos que se pueden formar, se tiene que:

$$N_d = \left(\frac{k(k-1)}{2}\right)^d \tag{5}$$

Como se observa en la ecuación (5), N_d crece exponencialmente con respecto al número de dimensiones (d), el cálculo de las N_d diferencias es un problema exigente computacionalmente, lo que limita la selección de k y d.

3.3. Estimación de la función

Un problema para la evaluación de la supermodularidad es el desconocimiento de la función a evaluar, ya que de saberlo no sería necesario el test. Como se ve en la ecuación 3 lo que se estima es$\hat{f}(x)$, que puede ser no lineal, para lo cual se utilizan regresiones polinómicas con las variables previa ortogonalización de los polinomios para evitar multicolinealidad y se incluye la interacción entre variables para contemplar efectos no aditivos. El algoritmo para la construcción de las interacciones se utilizan los elementos del conjunto potencia de variables independientes de cardinalidad mayor o igual a 2, multiplicadas por un coeficiente. Por ejemplo, si tengo 3 variables $X = \{x_1, x_2, x_3\}$, el conjunto potencia sería $P(x) = \{\emptyset, \{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_1, x_3\}\, \{x_2, x_3\}\, \{x_1, x_2, x_3\}\}$, si llamo como $P(x)'$ únicamente a los elementos de $P(x)$ con cardinalidad mayor o igual a 2 tendría para el ejemplo que $P(x)' = \{\{x_1, x_2\}, \{x_1, x_3\}\, \{x_2, x_3\}\, \{x_1, x_2, x_3\}\}$, se aplica el producto de Hadamard a cada subconjunto y se multiplica por el coeficiente y se construye la sumatoria de todos los elementos se tiene que $G(X) = \gamma_1 x_1 x_2 x_3 + \gamma_2 x_1 x_2 + \gamma_3 x_1 x_3 + \gamma_4 x_2 x_3$. Por tanto, la ecuación de estimación será:

$$\hat{f}(x) = \beta_0 + \sum_{j-1}^{n} \sum_{i=1}^{k} \beta_{ji} x_j^k + G(x) \qquad (6)$$

Donde j corresponde a la j-ésima variable, i a la i-ésimo grado del polinomio de la variable j y $G(x)$ a las interacciones entre las n variables como se definió previamente. Para la elección del grado del polinomio se evalúan todas las regresiones con $i = (1, 2, 3, 4, 5)$ y se evalúa la significancia de los términos de interacción con el algoritmo de selección del modelo por *stepwise* AIC y para la selección del polinomio se toma la de mayor R^2 ajustado.
Esta estimación como propuesta funciona para un escenario en exogeneidad, pero el método permite que se utilice cualquier estimación de la función dado que la estimación de la función y la verificación de su supermodularidad son procesos separados, por tanto, en caso de romperse cualquier supuesto, se pueden utilizar métodos como variables instrumentales u OLS en 2 etapas.

3.3.1. Test 1: test a partir de mínimos

Una vez se ha generado la base de datos del escenario $f(X)$ donde $X = (x_1, \ldots, x_i, \ldots, x_n)$, se crea la *lattice* X donde el mínimo de es $\min(X)$ y el máximo por $\max(X)$ y se divide con un ancho de celda de cada dimensión se obtiene como $b_i = (\max(x_i) - \min(x_i))/(k - 1)$ siendo k el número de vértices como se define en la ecuación (5). Una vez se tiene la *lattice*, se generan todas sus *sublattice* con un lado mínimo o múltiplo de b_i para cada dimensión i.

Se estima la función a partir de la ecuación (6) y se evalúa para todos los vértices de X, es decir, se tendrá $\hat{f}(x)$ para todo $x \in X$. Con lo que finalmente se calcula, para todo x' y x'', es decir, para todos los vértices de las *sublattice* previamente generadas. Se obtiene:

$$\widehat{H}(x', x'') = \hat{f}(\max(x', x'') + \hat{f}(\min(x', x'')) - \hat{f}(x') - \hat{f}(x'') \qquad (7)$$

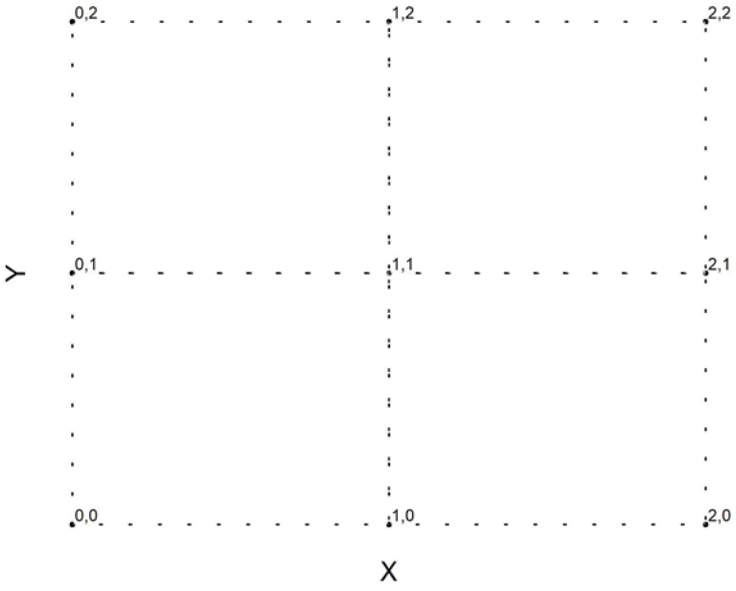
Se obtiene un vector de extensión N_d según la ecuación (5) con los valores de $\widehat{H}(x', x'')$ y se obtiene el mínimo valor mínimo del vector.

Se realiza el proceso para 100 remuestreos (en el Anexo 5 se muestra la distribución para los dos métodos propuestos en este tesis, se probó con 5, 10, 50, 100, 1000 y 5000 remuestreos y se observó que a partir de los 100 remuestreos, la función de distribución de probabilidad converge a la de 5000 remuestreos y en muchos casos deja de ser multimodal) de la base inicial con el fin de obtener una distribución de $\min_{x',x''\in X} \widehat{H}(x',x'')$ y estimar la probabilidad de que la función sea supermodular de acuerdo a la ecuación (4).

Para ejemplificar el algoritmo, suponemos que tenemos una base de datos con tres variables: x, y, z donde $z = f(x,y) + u$ donde $u \sim N(\mu, \sigma^2)$ y con un mínimo $\min(x,y) = (0,0)$ y un máximo $\max(x,y) = (2,2)$, y se quiere tres vértices $k = 3$ implica un $N_2 = 9$ con $b_x = 1$ y $b_y = 1$ (ver **Figura 3.1**). Se estiman 100 $\hat{f}(x,y)$ un por cada remuestreo de la base y se evalúa para todos los vértices de la lattice (ver **Figura 2**)

$X = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$

Figura 3.1. Lattice



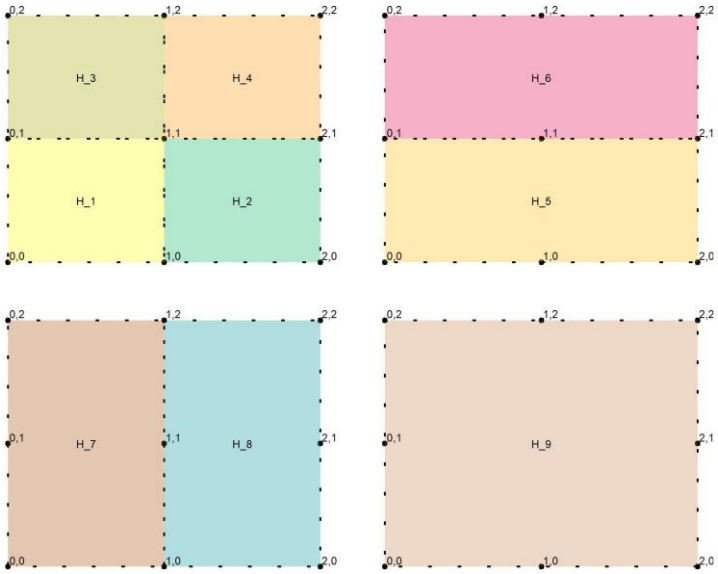
Fuente: Trabajo investigativo

Se evalúa la ecuación (7) para cada una de las *sublattices*, para las 100 $\hat{f}(x,y)$. Luego de lo cual se tiene una matriz 9×100.

$$\begin{matrix} H_{1,1} & \cdots & H_{1,100} \\ \vdots & \ddots & \vdots \\ H_{9,1} & \cdots & H_{9,100} \end{matrix}$$

Finalmente se obtiene el valor mínimo de las *sublattices* y se evalúa si es mayor a 0 obteniendo la $P(\min_{x',x'' \in X} \widehat{H}(x', x'') \geq 0)$ como resultado del test. El código paralelizado de este test realizado en R se encuentra en el Anexo 1.

Figura 2.2. Sublattices



Fuente: Trabajo investigativo

3.3.2. Test 2: test a partir de medias

Este test es una variación del anterior, en el que se toma el valor mínimo de H_i de todas las sublattices que es equivalente a

$$M_p(H_1, \ldots, H_N) = \left(\sum_{i=1}^{n} w_i H_i^p \right)^{\frac{1}{p}}$$

cuando $p \to -\infty$ y $w_i = 1 \quad \forall i$. En este segundo test se modifican esos parámetros para tomar en cuenta la distribución de datos dentro de la *lattice* y se utiliza el parámetro $p = 1$ de la media aritmética con el fin de hace menos estricta la interpretación del resultado, pero aun

fácilmente interpretable, es decir el resultado será la probabilidad de que en promedio la función presente diferencias crecientes. La ponderación de cada H_i viene dada por el número de observaciones que caen dentro de cada *sublattice* y si la condición de que la *sublattice* aporte nuevas observaciones para evitar ponderar *sublattices* que contienen a otras que aportan la totalidad de observaciones. Si defino al número de observaciones de la base de datos B que caen en una *lattice* L_i si $\min(L_i) \leq b_i \leq \max(L_i)\ \forall b_i \subseteq B$, entonces $|b_i|$ será el número de elementos de B que caen dentro de L_i. Puedo definir la ponderación w_i como

$$w_i = \frac{|b_i|}{\sum |b_i|}$$

Para ejemplificar utilizando el ejemplo previo del test 1, suponemos que tenemos una base de datos con tres variables: x, y, z donde $z = f(x, y) + u$ donde $u \sim N(\mu, \sigma^2)$ y con un mínimo $\min(x, y) = (0, 0)$ y un máximo $\max(x, y) = (2, 2)$, y se quiere tres vértices $k = 3$ implica un $N_2 = 9$ con $b_x = 1$ y $b_y = 1$ (ver **Figura**). Se estiman 100 $\hat{f}(x, y)$ un por cada remuestreo de la base y se evalúa para todos los vértices de la *lattice* (ver **Figura**)
$X = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)\}$

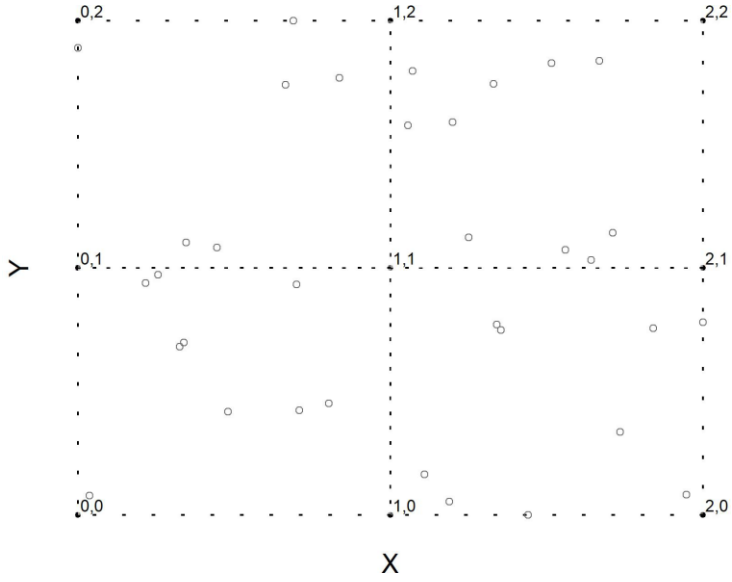
Se evalúa la ecuación (7) para cada una de las *sublattices*, para las 100 $\hat{f}(x, y)$. Luego de lo cual se tiene una matriz 9×100, y un vector 9×1 de pesos w_i.

$$H = \begin{bmatrix} H_{1,1} & \cdots & H_{1,100} \\ \vdots & \ddots & \vdots \\ H_{9,1} & \cdots & H_{9,100} \end{bmatrix}$$

$$W = \begin{bmatrix} w_i \\ \vdots \\ w_9 \end{bmatrix}$$

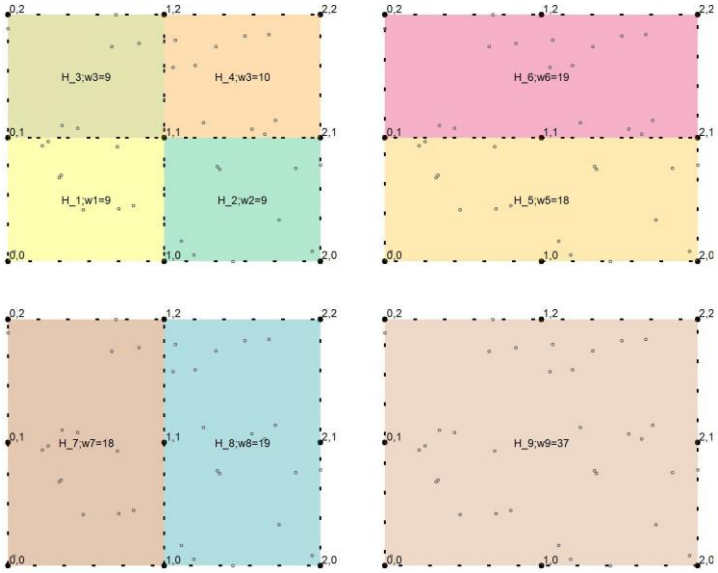
Finalmente se obtiene el promedio con la ecuación (8) y se evalúa si es mayor a 0 cada elemento del vector.

Figura 3.3. Lattice con datos



Fuente: Trabajo investigativo

Figura 3.4. Sublattices con ponderación



Fuente: Trabajo investigativo

$$H'W\left(W'\vec{1}\right)^{-1} \tag{8}$$

El algoritmo paralelizado en R se encuentra en el Anexo 2.

3.3. Evaluación del test

Para evaluar los test para mediciones de complementariedad con variables independientes continuas se realiza un benchmark comparando la significancia y el poder entre el test 1 y el test 2 contra el estándar que exige que las derivadas cruzadas de todas las variables sean no negativas calculadas a partir del coeficiente de interacción entre las variables de la función (ver Leiponen 2005, Miravete y Pernias 2006, Carree, Lokshin y Belderbos 2011, Belderbos, Carree y Lokshin 2006). Para la evaluación de los test se utilizarán funciones en de $R^2 \rightarrow R^1$ por simplicidad, así, el test benchmark queda definido como:

$$f(x_1, x_2) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_{12} x_1 x_2$$

Y $f(x_1, x_2)$ se acepta como supermodular si α_{12} es positivo y estad´ısticamente significativo. Se prueban las siguientes tres funciones

Tabla 3.1. Funciones y parámetros para el test

Función	Parámetros
$k \exp(x_1 + x_2) + u$	$x_1, x_2 \sim N(2,1)$ $u \sim N(0, \sigma)$ $\sigma = \{\exp 2, \exp 4, \dots, \exp 16\}$ $k = (-1)^q \; donde \; q \sim \; B(1, 0.5)$
$k \; x_1^{0.5} x_2^{0.5} + u$	$x_1, x_2 \sim N(10,1)$ acotado a $x_1, x_2 > 0$ $u \sim N(0, \sigma)$ $\sigma = \{1, \dots, 8\}$ $k = (-1)^q \; donde \; q \sim \; B(1, 0.5)$
$x_1 + x_2 + k \; x_1 x_2 + \; u$	$x_1, x_2 \sim N(2,1)$ $u \sim N(0, \sigma)$ $\sigma = \{2, 4, 6, \dots, 16\}$ $k = (-1)^q \; donde \; q \sim \; B(1, 0.5)$
$x_1^{k \, x_2} + x_2^{k \, x_1} + u$	$x_1, x_2 \sim N(2,1)$ acotado a $x_1, x_2 > 1$ $u \sim N(0, \sigma)$ $\sigma = \{2, 4, 6, \dots, 16\}$ $k = (-1)^q \; donde \; q \sim \; B(1, 0.5)$

Fuente: Trabajo investigativo

Para cada función se realizan 100 repeticiones por cada σ. El objetivo de las 100 repeticiones es calcular la precisión, exhaustividad y F1 score para cada caso, ya que dado que k puede tomar valores $(-1, 1)$ con $p = 0{,}5$. Cuando $k = 1$ la función será supermodular, mientras que para $k = -1$ será submodular y se espera que los test puedan identificar la diferencia. Mientras que el fin de cada σ es medir las variaciones en el poder y significancia ante casos de mayor varianza. Es necesario definir las medidas de comparación entre los resultados de cada test, para lo cual se introducen a continuación los conceptos de evaluación de medidas. Las definiciones se obtienen de Powers 2020. En la **Tabla** 3.2 se muestra la matriz de confusión de los resultados del test que servirá para la medición de las medidas de clasificación.

Tabla 3.2. Matriz de confusión

		Condición verdadera	
		Positivo	Negativo
Condición Predicha	Positivo	Verdadero positivo (VP)	Falso positivo (FP)
	Negativo	Falso negativo (FN)	Verdadero negativo (VN)

Fuente: Trabajo investigativo

Donde:

- VP: la función es supermodular $(k = 1)$ y el test la identifica como tal.
- FP: la función no es supermodular $(k = -1)$ y el test la identifica como supermodular.
- FN: la función es supermodular $(k = 1)$ y el test la identifica como no supermodular.
- VN: la función no es supermodular $(k = -1)$ y el test la identifica como no supermodular.

Con lo que se calcula la precisión, exhaustividad y F1-score de la siguiente manera:

$$precision = \frac{VP}{VP + FP}$$

$$exhaustividad = \frac{VP}{VP + FN}$$

$$F_1 = 2 \cdot \frac{precision \cdot exhaustividad}{precision + exhaustividad}$$

Se calculan los resultados para 100 valores positivos y negativos que construyen la matriz de confusión para cada test y para cada valor de varianza del error.

Capítulo 4. Comparación entre test de supermodularidad

Se evaluaron las funciones en $R^2 \rightarrow R$, exponencial base e, exponencial $x_1^{x_2} + x_2^{x_1}$, Cobb-Douglas y lineal como se especifica en la **Tabla** (la sintaxis del cálculo de los test están en el Anexo 3), para 100 simulaciones y se calcularon los casos verdadero positivo (VP), verdadero negativo (VN), falso positivo (FP) y falso negativo (FN) y a partir de ellos, la precisión, exhaustividad y como medida de resumen para la comparación se utiliza el F1-score (ver sintaxis de evaluación de los test en el Anexo 4).

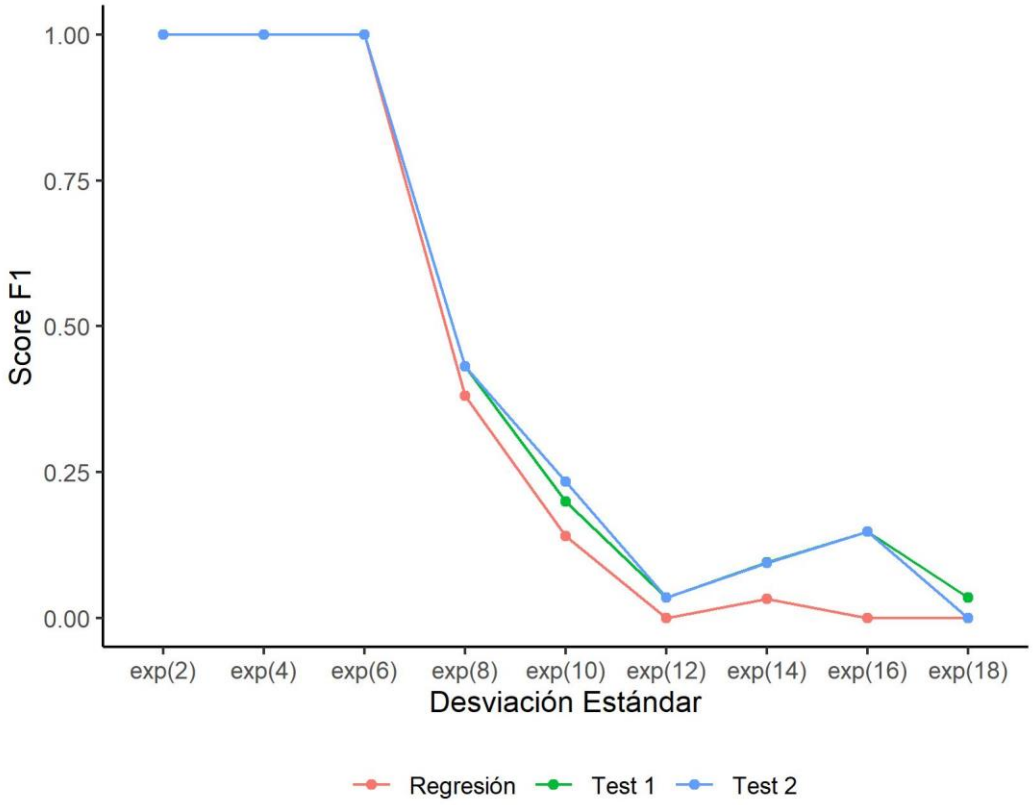
Para el caso de la función exponencial (**Tabla** 4.1), para los valores de mayor desviación estándar del error se observa que los tres test inflan los falsos negativos, pero en general el rendimiento de los dos test propuestos supera el del test estándar, sobre todo porque presentan una mayor exhaustividad. Esto se puede ver más claramente en la **Figura** 4.1 donde se observa que las líneas verde y azul, correspondientes a los test propuestos a partir de mínimos y medias respectivamente, se encuentran iguales o superiores a la línea rosa correspondiente al test estándar.

Tabla 4.1. Resultados de los test para la función $k\exp(x_1 + x_2) + u$

DS del error	Tipo de test	Verdadero positivo	Verdadero negativo	Falso positivo	Falso negativo	Preci.	Exhaust.	Score F1
	Estándar	45	55	0	0	1.000	1.000	1.000
exp(2)	Test 1	45	55	0	0	1.000	1.000	1.000
	Test 2	45	55	0	0	1.000	1.000	1.000
	Estándar	56	44	0	0	1.000	1.000	1.000
exp(4)	Test 1	56	44	0	0	1.000	1.000	1.000
	Test 2	56	44	0	0	1.000	1.000	1.000
	Estándar	53	47	0	0	1.000	1.000	1.000
exp(6)	Test 1	53	47	0	0	1.000	1.000	1.000
	Test 2	53	47	0	0	1.000	1.000	1.000
	Estándar	12	49	0	39	1.000	0.235	0.381
exp(8)	Test 1	14	49	0	37	1.000	0.275	0.431
	Test 2	14	49	0	37	1.000	0.275	0.431
	Estándar	4	47	0	49	1.000	0.075	0.140
exp(10)	Test 1	6	46	1	47	0.857	0.113	0.200
	Test 2	7	47	0	46	1.000	0.132	0.233
	Estándar	0	43	2	55	0.000	0.000	0.000
exp(12)	Test 1	1	43	2	54	0.333	0.018	0.034
	Test 2	1	44	1	54	0.500	0.018	0.035
	Estándar	1	40	4	55	0.200	0.018	0.033
exp(14)	Test 1	3	40	4	53	0.429	0.054	0.095
	Test 2	3	39	5	53	0.375	0.054	0.094
	Estándar	0	50	0	50	0.000	0.000	0.000
exp(16)	Test 1	4	50	0	46	1.000	0.080	0.148
	Test 2	4	50	0	46	1.000	0.080	0.148
	Estándar	0	44	5	51	0.000	0.000	0.000
exp(18)	Test 1	1	43	6	50	0.143	0.020	0.034
	Test 2	0	43	6	51	0.000	0.000	0.000

Fuente: Trabajo investigativo

Figura 4.1. Score F1 de los test para la función $k \exp(x_1 + x_2) + u$



Fuente: Trabajo investigativo

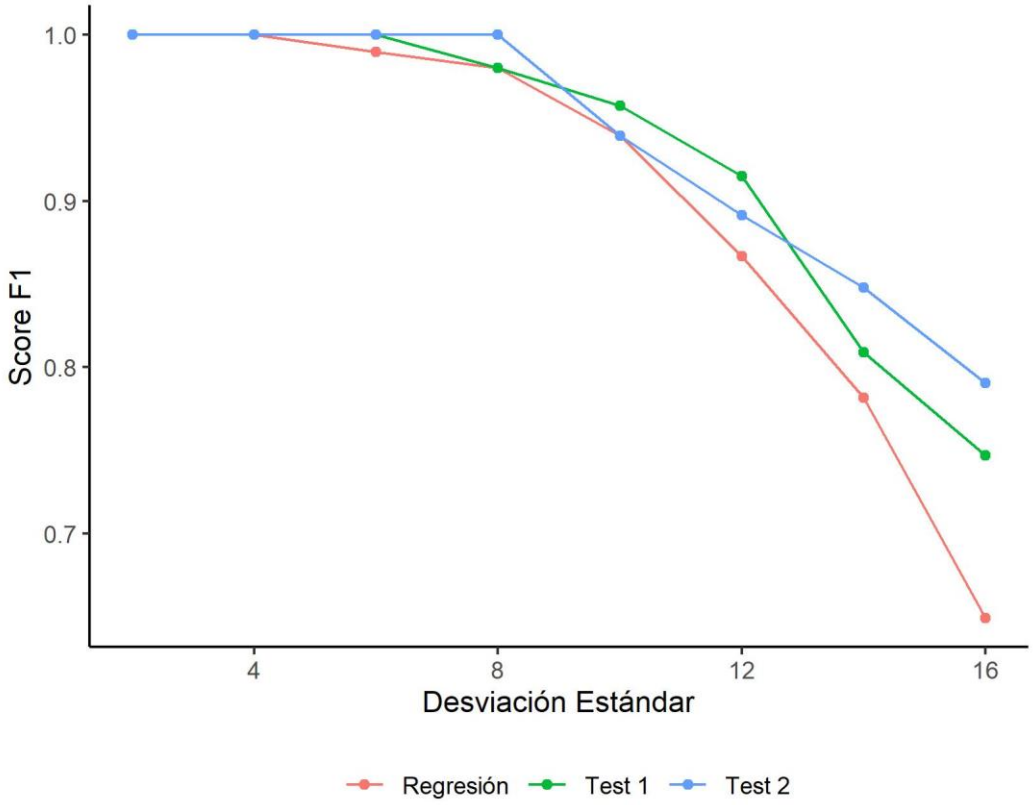
Para el caso de la función lineal (**Tabla**), al igual que en caso anterior, los test identifican correctamente cuando la función es supermodular en la menor desviación estándar del error. Pero a medida que el error aumenta, el rendimiento de los test propuestos es superior, por poco, al del estándar debido a la mayor exhaustividad. El resultado resumido por el score F1 se observa en la **Figura** el score F1 para los distintos niveles de error.

Tabla 4.2. Resultados de los test para la función $x_1 + x_2 + k\,x_1x_2 + u$

DS del error	Tipo de test	Verdadero positivo	Verdadero negativo	Falso positivo	Falso negativo	Preci.	Exhaust.	Score F1
2	Estándar	55	45	0	0	1.000	1.000	1.000
	Test 1	55	45	0	0	1.000	1.000	1.000
	Test 2	55	45	0	0	1.000	1.000	1.000
4	Estándar	51	49	0	0	1.000	1.000	1.000
	Test 1	51	49	0	0	1.000	1.000	1.000
	Test 2	51	49	0	0	1.000	1.000	1.000
6	Estándar	48	51	0	1	1.000	0.980	0.990
	Test 1	49	51	0	0	1.000	1.000	1.000
	Test 2	49	51	0	0	1.000	1.000	1.000
8	Estándar	49	49	0	2	1.000	0.961	0.980
	Test 1	49	49	0	2	1.000	0.961	0.980
	Test 2	51	49	0	0	1.000	1.000	1.000
10	Estándar	54	39	0	7	1.000	0.885	0.939
	Test 1	56	39	0	5	1.000	0.918	0.957
	Test 2	54	39	0	7	1.000	0.885	0.939
12	Estándar	39	49	0	12	1.000	0.765	0.867
	Test 1	43	49	0	8	1.000	0.843	0.915
	Test 2	41	49	0	10	1.000	0.804	0.891
14	Estándar	34	47	0	19	1.000	0.642	0.782
	Test 1	36	47	0	17	1.000	0.679	0.809
	Test 2	39	47	0	14	1.000	0.736	0.848
16	Estándar	25	48	0	27	1.000	0.481	0.649
	Test 1	31	48	0	21	1.000	0.596	0.747
	Test 2	34	48	0	18	1.000	0.654	0.791

Fuente: Trabajo investigativo

Figura 4.2. Score F1 de los test para la función $x_1 + x_2 + k\,x_1x_2 + u$



Fuente: Trabajo investigativo

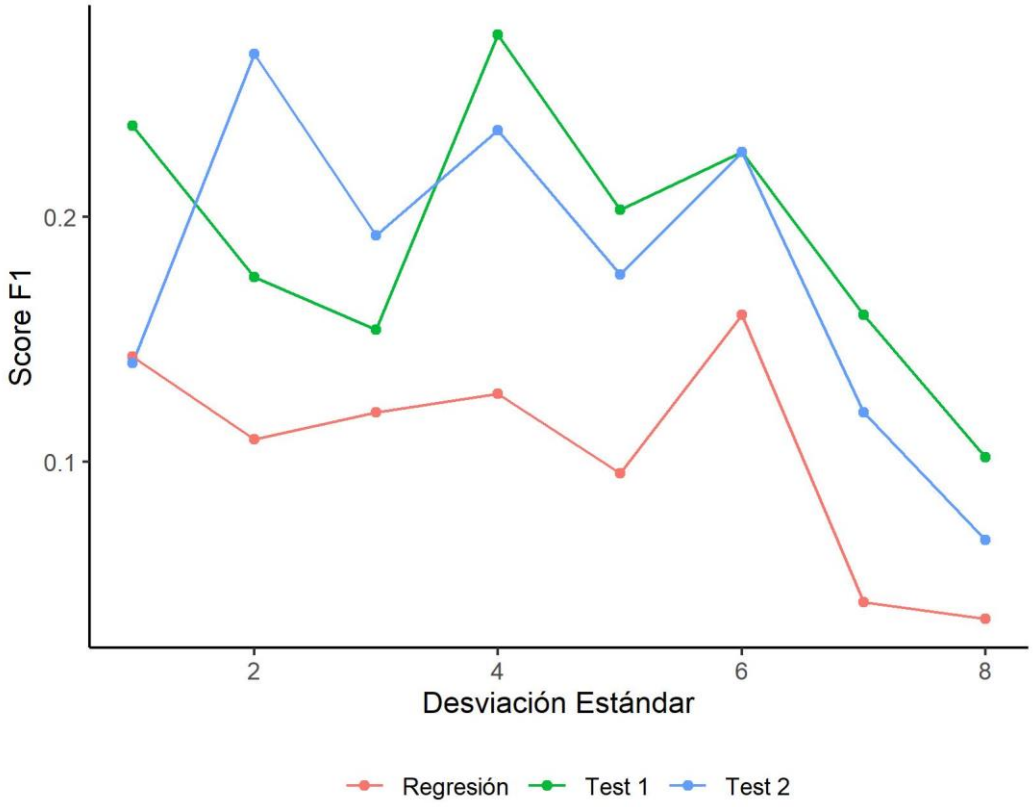
Para el caso de la función Cobb-Douglas (Tabla) el rendimiento de los tres test es inferior al visto en los casos anteriores. Para los tres test, el F1 disminuye a medida que se incrementa la desviación estándar del componente aleatorio de la función, pero en general, los test 1 y 2 tiene un desempeño mejor o igual que el del test estándar. El principal problema para todos los casos es el error tipo II o falsos negativos, es decir, se está sobre rechazando funciones que sí son supermodulares. De forma resumida, se ve en la Figura 4.2. que, si bien el score F1 es bajo dado el alto porcentaje de falsos negativos, en general, los test propuestos tienen mejor rendimiento que el estándar.

Tabla 4.3. Resultados de los test para la función $k\, x_1^{0.5} x_2^{0.5} + u$

DS del error	Tipo de test	Verdadero positivo	Verdadero negativo	Falso positivo	Falso negativo	Preci.	Exhaust.	Score F1
1	Estándar	4	48	0	48	1.000	0.077	0.143
	Test 1	7	48	0	45	1.000	0.135	0.237
	Test 2	4	47	1	48	0.800	0.077	0.140
2	Estándar	3	48	0	49	1.000	0.058	0.109
	Test 1	5	48	0	47	1.000	0.096	0.175
	Test 2	8	48	0	44	1.000	0.154	0.267
3	Estándar	3	53	1	43	0.750	0.065	0.120
	Test 1	4	52	2	42	0.667	0.087	0.154
	Test 2	5	53	1	41	0.833	0.109	0.192
4	Estándar	3	56	1	40	0.750	0.070	0.128
	Test 1	7	56	1	36	0.875	0.163	0.275
	Test 2	6	55	2	37	0.750	0.140	0.235
5	Estándar	3	40	2	55	0.600	0.052	0.095
	Test 1	7	38	4	51	0.636	0.121	0.203
	Test 2	6	38	4	52	0.600	0.103	0.176
6	Estándar	4	54	0	42	1.000	0.087	0.160
	Test 1	6	53	1	40	0.857	0.130	0.226
	Test 2	6	53	1	40	0.857	0.130	0.226
7	Estándar	1	54	0	45	1.000	0.022	0.043
	Test 1	4	54	0	42	1.000	0.087	0.160
	Test 2	3	53	1	43	0.750	0.065	0.120
8	Estándar	1	45	2	52	0.333	0.019	0.036
	Test 1	3	44	3	50	0.500	0.057	0.102
	Test 2	2	43	4	51	0.333	0.038	0.068

Fuente: Trabajo investigativo

Figura 4.3. Score F1 de los test para la función $k\, x_1^{0.5} x_2^{0.5} + u$



Fuente: Trabajo investigativo

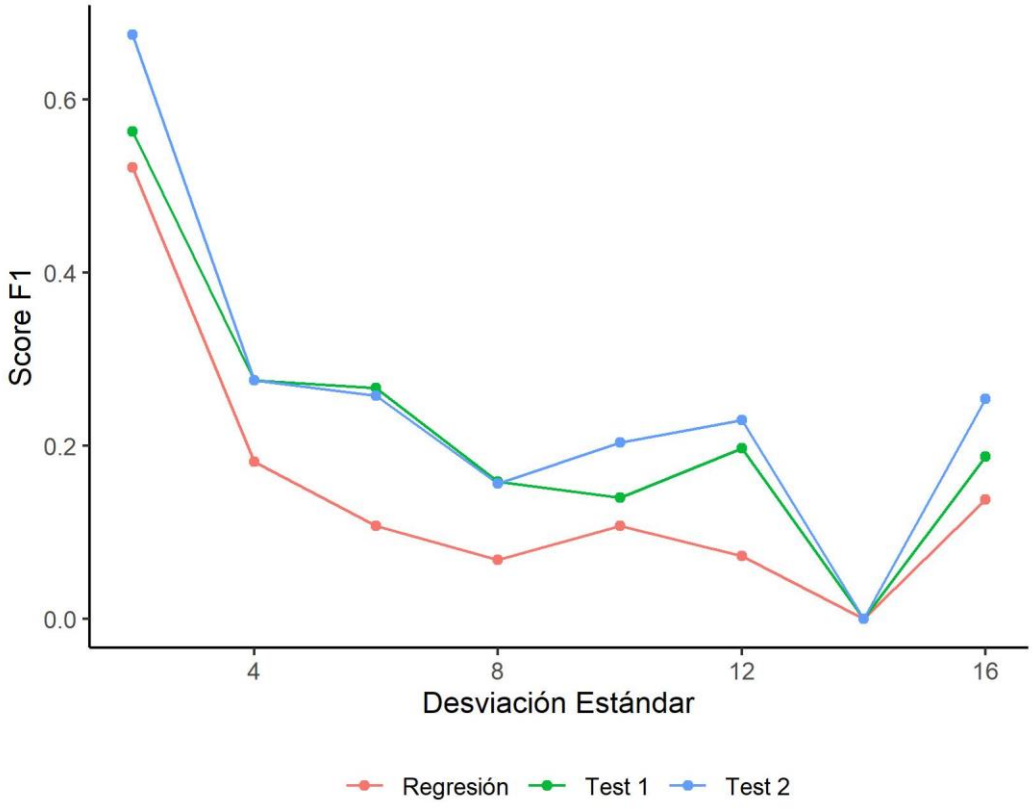
Para el caso de la función exponencial $x_1^{x_2} + x_2^{x_1}$ (**Tabla** 4.3), al igual que en los casos previos, los test tienen una buena precisión, pero fallan en la exhaustividad por un alto porcentaje de error tipo II. Para las mayores desviaciones estándar del error, el rendimiento de todos los test cae, aun así, los test propuestos tienen un mejor rendimiento tanto en precisión como exhaustividad. La figura 4.3 muestra el resumen por medio del score F1.

Tabla 4.4. Resultados de los test para la función $x_1^{k\,x_2} + x_2^{k\,x_1} + u$

DS del error	Tipo de test	Verdadero positivo	Verdadero negativo	Falso positivo	Falso negativo	Preci.	Exhaust.	Score F1
	Estándar	18	49	0	33	1.000	0.353	0.522
2	Test 1	20	49	0	31	1.000	0.392	0.563
	Test 2	26	49	0	25	1.000	0.510	0.675
	Estándar	5	50	0	45	1.000	0.100	0.182
4	Test 1	8	50	0	42	1.000	0.160	0.276
	Test 2	8	50	0	42	1.000	0.160	0.276
	Estándar	3	47	2	48	0.600	0.059	0.107
6	Test 1	8	48	1	43	0.889	0.157	0.267
	Test 2	8	46	3	43	0.727	0.157	0.258
	Estándar	2	43	2	53	0.500	0.036	0.068
8	Test 1	5	42	3	50	0.625	0.091	0.159
	Test 2	5	41	4	50	0.556	0.091	0.156
	Estándar	3	47	2	48	0.600	0.059	0.107
10	Test 1	4	47	2	47	0.667	0.078	0.140
	Test 2	6	47	2	45	0.750	0.118	0.203
	Estándar	2	47	1	50	0.667	0.038	0.073
12	Test 1	6	45	3	46	0.667	0.115	0.197
	Test 2	7	46	2	45	0.778	0.135	0.230
	Estándar	0	50	0	50	0.000	0.000	0.000
14	Test 1	0	50	0	50	0.000	0.000	0.000
	Test 2	0	49	1	50	0.000	0.000	0.000
	Estándar	4	46	0	50	1.000	0.074	0.138
16	Test 1	6	42	4	48	0.600	0.111	0.188
	Test 2	8	45	1	46	0.889	0.148	0.254

Fuente: Trabajo investigativo

Figura 4.4. Score F1 de los test para la función $x_1^{k\,x_2} + x_2^{k\,x_1} + u$



Fuente: Trabajo investigativo

Conclusiones

Se diseñaron dos test de supermodularidad y se pusieron a prueba contra el test estándar. Los resultados fueron positivos para los cuatro tipos de funciones que se probaron: exponencial base e, exponencial $x_1^{x_2} + x_2^{x_1}$, Cobb-Douglas y lineal. Los dos test propuestos, test 1 o test realizado a partir del mínimo de las diferencias y el test 2 o test realizado a partir de las diferencias medias fueron semejantes en su rendimiento, sin que ninguno destacara, pero superiores si se comparan con el test estándar. Esta diferencia en el rendimiento por el score F1 se debe principalmente a que el test estándar tiene una menor exhaustividad comparada con los test propuestos, es decir, el test estándar, comparado con los propuestos identifica incorrectamente con mayor frecuencia como no supermodulares a funciones que sí lo son. Por lo tanto, se considera que los test propuestos pueden funcionar bastante bien como tests complementarios al test estándar ya que tienen diferentes fortalezas y un resultado confirmado por los 3 test puede dar mucha más certeza del resultado.

Con respecto al rendimiento computacional, se encuentra que el tiempo de pro- cesamiento de los test 2 y 3 son muy superiores al del test estándar. Si bien de- pende del equipo de cómputo en el cual se corren los test, a modo de referencia, el tiempo del test estándar promedio fue de 0.001 segundos, mientras que el test 2 fue de 12.172 segundos y del test 3 de 17.563 segundos, lo que constituye 990100% y 1428539% mayor tiempo relativo al test estándar respectivamente[3]. Esto se da ya que en la creación de *lattices* y el cálculo sobre todas las posibles *lattices* requiere de un número elevado de cálculos en paralelo además de que el número de *lattices* construidas crece exponencialmente con el número de dimensiones de la función lo que eventualmente puede constituir en una limitante para el uso del test propuesto.

Finalmente, si bien el costo computacional es grande, esta metodología abre una oportunidad para la estimación de supermodularidad de cualquier función ya que, al constituirse de 2 partes independientes, la estimación de la función y la verificación de la condición de supermodularidad, da la oportunidad de modificar según la necesidad la estimación de la función, ya sea con métodos paramétricos o no paramétricos, sin que la condición de supermodularidad requiera un mayor ajuste ya que se construye a partir de la misma definición de supermodularidad.

[3] Se utilizaron 5 núcleos de un procesador Intel(R) Core(TM) i7-8750H 2.20GHz y 16Gb de RAM.

Lista de Referencias

Amir, Rabah. 2005. *"*Supermodularity and complementarity in economics: An elementary survey*"*. *Southern Economic Journal*: 636-660.

Belderbos, Ren´e, Martin Carree y Boris Lokshin. 2006. *"*Complementarity in R&D cooperation strategies*"*. *Review of Industrial Organization* 28 (4): 401-426.

Carree, Martin, Boris Lokshin y Ren´e Belderbos. 2011. *"*A note on testing for complementarity and substitutability in the case of multiple practices*"*. *Journal of productivity Analysis* 35 (3): 263-269.

Cassiman, Bruno, y Reinhilde Veugelers. 2006. *"*In search of complementarity in innovation strategy: Internal R&D and external knowledge acquisition*"*.

Management science 52 (1): 68-82.

Leiponen, Aija. 2005. *"*Skills and innovation*"*. *International Journal of Industrial Organization* 23 (5-6): 303-323.

Milgrom, Paul, y John Roberts. 1995. *"*Complementarities and fit strategy, structure, and organizational change in manufacturing*"*. *Journal of accounting and economics* 19 (2-3): 179-208.

Miravete, Eugenio J, y Jose C Pernias. 2006. *"*Innovation complementarity and scale of production*"*. *The Journal of Industrial Economics* 54 (1): 1-29.

OEIS Foundation Inc. 2020. *The On-Line Encyclopedia of Integer Sequences*. http://oeis.org.. Acceso: 26-01-2020.

Powers, David MW. 2020. *"*Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*"*. *arXiv preprint arXiv:2010.16061*.

Topkis, Donald D. 1998. *Supermodularity and Complementarity*. Princeton University Press. isbn: 9780691032443. http://www.jstor.org/stable/j. ctt7s83q.

Topkis, Donald M. 1978. *"*Minimizing a submodular function on a lattice*"*. *Operations research* 26 (2): 305-321.

Anexos
Anexo 1

```
# Sintaxis de cálculo de supermodularidad en R2

rm(list = ls())

library(foreach)
library(iterators)
library(parallel)
library(doParallel)
library(MASS)
library(tidyverse)
library(rlang)
library(ggplot2)
library(factoextra)
library(cluster)
library(here)

# Semilla para reproducibilidad de resultados
set.seed(123)

######## Funciones #########

#### Función de lista de regresiones ####
# Genera un objeto de clase formula con la ecuación para el cálculo
# de la orthogonal polynomial regression.

f_fformula <-
  function(ind = 'y',
           vec,
           nivel = c(5, 5)) {
    inter <-
      (rje::powerSet(vec, m = max(nivel)))[-1]

    form <-
      paste0(ind,
             '~',
             c(
               paste0('poly(', vec, ', degree = ', nivel, ', raw = T)'),
               inter[lapply(inter, length) > 1] %>%
                 map(~ .x %>% paste0(collapse = ':'))
             ) %>%
               unlist %>%
               paste(collapse = '+')) %>%
      as.formula

    poly <- paste0(ind,
                   '~',
                   paste0('poly(', vec, ', degree = ', nivel, ', raw = T)')
%>%
                     unlist %>%
                     paste(collapse = '+')) %>%
      as.formula

    return(list(form = form,
                poly = poly))
  }
```

```r
sm_dfmin <-
  function(bdd,
           ind = 'y',
           vec = c('x1', 'x2'),
           samples = 200,
           n_ver = 15) {
    # Modelo

    par_potencia <-
      cross2(.x = c(1:5),
             .y = c(1:5)) %>%
      map(lift(c))

    model <-
      par_potencia %>%
      map(.f = ~{
          full <- f_fformula(ind = ind,
                             vec = vec,
                             nivel = .x)
          invisible(capture.output(
            model <-
            full$form %>%
            lm(formula = .,
               data = bdd) %>%
            stepAIC(.,
                    direction = "both",
                    scope = list(lower=full$poly,upper=full$form))
          ))

          form <- (as.character(model$call))[2]
          r2 <- summary(model)$adj.r.squared

          list(form=form,r2=r2)
          })

    formul_fin <-
      unlist(transpose(model)$form)[which.max(transpose(model)$r2)] %>%
      as.formula

    lista_fr <-
      1:samples %>%
      map(.x = .,
          .f = ~ {
            bdd %>% sample_frac(., 1, replace = T) %>%
              lm(formul_fin,
                 data = .)
          })

    salto <-
      bdd %>%
      select(!!!syms(vec)) %>%
      map( ~ c(min(.x), max(.x))) %>%
      map( ~ (.x[2] - .x[1]) / (n_ver-1))

    rejilla <-
      bdd %>%
      dplyr::select(x1, x2) %>%
      map( ~ c(min(.x), max(.x))) %>%
      map( ~ seq(.x[1], .x[2], (.x[2] - .x[1]) / (n_ver-1))) %>%
      cross_df  %>%
      mutate(contador = 1:nrow(.))
```

```
rejilla_eval <-
  map_dfc(.x = lista_fr,
          .f = ~ predict(object = .x, newdata = rejilla)) %>%
  set_names(nm = paste0('pred_', 1:ncol(.))) %>%
  as.matrix

l_rejilla <-
  map(
    .x = 1:nrow(rejilla),
    .f =  ~
      rejilla %>%
      filter(x1 > rejilla$x1[.x] &
               x2 > rejilla$x2[.x])
  )

rejilla_f <-
  rejilla %>%
  filter(map_int(l_rejilla, nrow) > 0)

l_rejilla <-
  l_rejilla[map_int(l_rejilla, nrow) > 0]

nc <- (detectCores(logical = F) - 2)
cl = makeCluster(nc)
registerDoParallel(cl)

res <-
  foreach(
    x = 1:nrow(rejilla_f),
    .combine = cbind,
    .packages = c('tidyverse')
  ) %dopar% {
    p_min <- rejilla_f %>% slice(x) %>% as.list
    l_p_max <- l_rejilla %>% pluck(x)

    r <-
      map(.x = 1:nrow(l_p_max),
          .f =  ~ {
            p_max <- l_p_max %>% slice(.x) %>% as.list
            posicion <- rejilla %>%
              filter(x1 %in% c(p_min$x1, p_max$x1) &
                       x2 %in% c(p_min$x2, p_max$x2)) %>%
              pull(contador)

            (rejilla_eval[posicion[4],] +
                rejilla_eval[posicion[1],] -
                rejilla_eval[posicion[2],] -
                rejilla_eval[posicion[3],])
          }) %>%
      reduce(cbind)

    if (class(r) == 'matrix') {
      apply(r, 1, min)
    } else{
      r
    }

  } %>%
  apply(., 1, min)
```

```
  stopCluster(cl)

  return(res)

}
```

Anexo 2

```
# Sintaxis de cálculo de supermodularidad en R2

rm(list = ls())

library(foreach)
library(iterators)
library(parallel)
library(doParallel)
library(MASS)
library(tidyverse)
library(rlang)
library(ggplot2)
library(factoextra)
library(cluster)
library(here)

# Semilla para reproducibilidad de resultados
set.seed(123)

######## Funciones #########


#### Función de lista de regresiones ####
# Genera un objeto de clase formula con la ecuación para el cálculo
# de la orthogonal polynomial regression.

f_fformula <-
  function(ind = 'y',
           vec,
           nivel = c(5, 5)) {
    inter <-
      (rje::powerSet(vec, m = max(nivel)))[-1]

    form <-
      paste0(ind,
             '~',
             c(
               paste0('poly(', vec, ', degree = ', nivel, ', raw = T)'),
               inter[lapply(inter, length) > 1] %>%
                 map(~ .x %>% paste0(collapse = ':'))
             ) %>%
               unlist %>%
               paste(collapse = '+')) %>%
      as.formula

    poly <- paste0(ind,
                   '~',
                   paste0('poly(', vec, ', degree = ', nivel, ', raw = T)')
%>%
                     unlist %>%
                     paste(collapse = '+')) %>%
      as.formula

    return(list(form = form,
                poly = poly))
  }

##############################################
```

```r
sm_media <-
  function(bdd,
           ind = 'y',
           vec = c('x1', 'x2'),
           samples = 200,
           n_ver = 15) {

    # Modelo

    par_potencia <-
      cross2(.x = c(1:5),
             .y = c(1:5)) %>%
      map(lift(c))

    model <-
      par_potencia %>%
      map(.f = ~{
        full <- f_fformula(ind = ind,
                           vec = vec,
                           nivel = .x)
        invisible(capture.output(
          model <-
            full$form %>%
            lm(formula = .,
               data = bdd) %>%
            stepAIC(.,
                    direction = "both",
                    scope = list(lower=full$poly,upper=full$form))
        ))

        form <- (as.character(model$call))[2]
        r2 <- summary(model)$adj.r.squared

        list(form=form,r2=r2)
      })

    formul_fin <-
      unlist(transpose(model)$form)[which.max(transpose(model)$r2)] %>%
      as.formula

    lista_fr <-
      1:samples %>%
      map(.x = .,
          .f = ~ {
            bdd %>% sample_frac(., 1, replace = T) %>%
              lm(formul_fin,
                 data = .)
          })

    salto <-
      bdd %>%
      select(!!!syms(vec)) %>%
      map(~ c(min(.x), max(.x))) %>%
      map(~ (.x[2] - .x[1]) / (n_ver-1))

    rejilla <-
      bdd %>%
      dplyr::select(x1, x2) %>%
      map(~ c(min(.x), max(.x))) %>%
      map(~ seq(.x[1], .x[2], (.x[2] - .x[1]) / (n_ver-1))) %>%
```

```
  cross_df  %>%
  mutate(contador = 1:nrow(.))

rejilla_eval <-
  map_dfc(.x = lista_fr,
          .f = ~ predict(object = .x, newdata = rejilla)) %>%
  set_names(nm = paste0('pred_', 1:ncol(.))) %>%
  as.matrix

l_rejilla <-
  map(
    .x = 1:nrow(rejilla),
    .f =  ~
      rejilla %>%
      filter(x1 > rejilla$x1[.x] &
               x2 > rejilla$x2[.x])
  )

rejilla_f <-
  rejilla %>%
  filter(map_int(l_rejilla, nrow) > 0)

l_rejilla <-
  l_rejilla[map_int(l_rejilla, nrow) > 0]

nc <- (detectCores(logical = F) - 2)
cl = makeCluster(nc)
registerDoParallel(cl)

l_res <-
  foreach(
    x = 1:nrow(rejilla_f),
    .combine = 'append',
    .packages = c('tidyverse')
  ) %dopar% {
    p_min <-
      rejilla_f %>%
      slice(x) %>%
      as.list

    l_p_max <-
      l_rejilla %>%
      pluck(x)

    r <-
      map(.x = 1:nrow(l_p_max),
          .f =  ~ {
            p_max <-
              l_p_max %>%
              slice(.x) %>%
              as.list

            if (nrow(filter(
              bdd,
              x1 >= (p_max$x1 - salto$x1) &
              x1 < p_max$x1 &
              x2 >= (p_max$x2 - salto$x2) &
              x2 < p_max$x2
            )) > 0) {
              posicion <- rejilla %>%
                filter(x1 %in% c(p_min$x1, p_max$x1) &
```

```
                        x2 %in% c(p_min$x2, p_max$x2)) %>%
                pull(contador)

              peso <-
                nrow(filter(
                  bdd,
                  x1 >= p_min$x1 &
                    x1 < p_max$x1 &
                    x2 >= p_min$x2 &
                    x2 < p_max$x2
                ))

              res <-
                (rejilla_eval[posicion[4], ] +
                   rejilla_eval[posicion[1], ] -
                   rejilla_eval[posicion[2], ] -
                   rejilla_eval[posicion[3], ]) * peso

              list(mat_res = res,
                   peso = peso)
            }

        }) %>%
      compact %>%
      transpose

    if (!is_empty(r)) {
      return(list(list(
        matriz = t(reduce(r$mat_res, cbind)),
        peso = sum(unlist(r$peso))
      )))
    }
  }

  stopCluster(cl)

  res <-
    (l_res %>%
       transpose())$matriz %>%
    reduce(rbind) %>%
    colSums (na.rm = T) / ((l_res %>%
                              transpose())$peso %>%
                             reduce(sum))

  return(res)

}
```

Anexo 3

```
#### Testeo ####

sec_var <- exp(seq(2,16,2))

func1 <-
  foreach(x = sec_var) %do% {

    disp <- x

    res1 <-
      foreach(x = 1:100) %do% {

        print(paste0(disp," ",x))
        sm <- sample(c(1,-1), 1)

        bdd <-
          f_fge(
            nomb = c('x1', 'x2', 'u'),
            obs = 1000,
            par = c('m=2,sd=1',
                    'm=2,sd=1',
                    paste0('m=0,sd=', disp)),
            func = paste0('y=(', sm, ')*exp((x1+x2))+u')
          )

        r1 <-
          summary(bdd %>% lm(formula = 'y~x1+x2+x1:x2'))$coefficients

        r1 <- r1[row.names(r1)=='x1:x2']

        r1 <- ifelse(r1[1]>0 & r1[4]<0.05,T,F)

        r2 <-
          mean(sm_dfmin(bdd = bdd,
                        samples = 100) > 0)

        r3 <-
          mean(sm_media(bdd = bdd,
                        samples = 100) > 0)

        rm(bdd)

        tibble(sm=ifelse(sm==1,T,F),
               r1,
               r2,
               r3
        )
      } %>% reduce(bind_rows)
    res1
  }


saveRDS(func1,paste(here(),'res_exp.rds',sep = '/'))


#####################################################################
```

```r
sec_var <- 1:8

func2 <-
  foreach(x = sec_var) %do% {

    disp <- x

    res1 <-
      foreach(x = 1:100) %do% {

        print(paste0(disp," ",x))
        sm <- sample(c(1,-1), 1)

        bdd <-
          f_fge(
            nomb = c('x1', 'x2', 'u'),
            obs = 1100,
            par = c('m=10,sd=1',
                    'm=10,sd=1',
                    paste0('m=0,sd=', disp)),
            func = paste0('y=', sm, '*x1^(0.5)*x2^(0.5)+u')
          ) %>%
          filter(x1>0 & x2>0)

        r1 <-
          summary(bdd %>% lm(formula = 'y~x1+x2+x1:x2'))$coefficients

        r1 <- r1[row.names(r1)=='x1:x2']

        r1 <- ifelse(r1[1]>0 & r1[4]<0.05,T,F)

        r2 <-
          mean(sm_dfmin(bdd = bdd,
                        samples = 100) > 0)

        r3 <-
          mean(sm_media(bdd = bdd,
                        samples = 100) > 0)

        rm(bdd)

        tibble(sm=ifelse(sm==1,T,F),
               r1,
               r2,
               r3
        )
      } %>% reduce(bind_rows)
    res1
  }

saveRDS(func2,paste(here(),'res_cobb.rds',sep = '/'))

######################################################################

sec_var <- seq(2,16,2)

func3 <-
  foreach(x = sec_var) %do% {

    disp <- x
```

```
    res1 <-
      foreach(x = 1:100) %do% {

        print(paste0(disp," ",x))
        sm <- sample(c(1,-1), 1)

        bdd <-
          f_fge(
            nomb = c('x1', 'x2', 'u'),
            obs = 1000,
            par = c('m=2,sd=1',
                    'm=2,sd=1',
                    paste0('m=0,sd=', disp)),
            func = paste0('y=x1+x2+',sm,'*x1*x2+u')
          )

        r1 <-
          summary(bdd %>% lm(formula = 'y~x1+x2+x1:x2'))$coefficients

        r1 <- r1[row.names(r1)=='x1:x2']

        r1 <- ifelse(r1[1]>0 & r1[4]<0.05,T,F)

        r2 <-
          mean(sm_dfmin(bdd = bdd,
                        samples = 100) > 0)

        r3 <-
          mean(sm_media(bdd = bdd,
                        samples = 100) > 0)

        rm(bdd)

        tibble(sm=ifelse(sm==1,T,F),
               r1,
               r2,
               r3
        )
      } %>% reduce(bind_rows)
    res1
  }

saveRDS(func3,paste(here(),'res_lineal.rds',sep = '/'))

#######################################################################

sec_var <- seq(2,16,2)

func4 <-
  foreach(x = sec_var) %do% {

    disp <- x

    res1 <-
      foreach(x = 1:100) %do% {

        print(paste0(disp," ",x))
        sm <- sample(c(1,-1), 1)

        bdd <- f_fge(
          nomb = c('x1', 'x2', 'u'),
```

```
          obs = 2000,
          par = c('m=2,sd=1',
                  'm=2,sd=1',
                  paste0('m=0,sd=', disp)),
          func = paste0('y=x1^(', sm, '*0.1*x2)+', 'x2^(', sm,
'*0.1*x1)+u')
        ) %>%
          filter(x1>1 & x2>1)

        r1 <-
          summary(bdd %>% lm(formula = 'y~x1+x2+x1:x2'))$coefficients

        r1 <- r1[row.names(r1)=='x1:x2']

        r1 <- ifelse(r1[1]>0 & r1[4]<0.05,T,F)

        r2 <-
          mean(sm_dfmin(bdd = bdd,
                        samples = 100) > 0)

        r3 <-
          mean(sm_media(bdd = bdd,
                        samples = 100) > 0)

        rm(bdd)

        tibble(sm=ifelse(sm==1,T,F),
               r1,
               r2,
               r3
        )
      } %>% reduce(bind_rows)
    res1
  }

saveRDS(func4,paste(here(),'res_expexp.rds',sep = '/'))
```

Anexo 4

```
rm(list = ls())
library(tidyverse)
library(ggplot2)
library(here)


res_cobb <- read_rds('res_cobb.rds')
res_lineal <- read_rds('res_lineal.rds')
res_exp <- read_rds('res_exp.rds')
res_expexp <- read_rds('res_expexp.rds')

# X^X

res_expexp <-
  map2(.x =
         map(res_expexp,
             ~ {
               .x %>%
                 filter(!is.na(r2),!is.na(r3)) %>%
                 mutate(r2 = r2 > .95,
                        r3 = r3 > .95)
             }),
       .y = seq(2,18,2),
       .f =
         ~ {
           vec_nam <- c('r1', 'r2', 'r3')
           tabla <- .x
           SD <- .y
           map(.x = vec_nam,
               .f =
                 ~ {
                   tc <-
                     tabla %>%
                     group_by(sm, !!sym(.x)) %>%
                     tally

                   tp <-
                     filter(.data = tc, sm == T & !!sym(.x) == T)
                   tn <-
                     filter(.data = tc, sm == F & !!sym(.x) == F)
                   fp <-
                     filter(.data = tc, sm == F & !!sym(.x) == T)
                   fn <-
                     filter(.data = tc, sm == T & !!sym(.x) == F)

                   tibble(
                     SD = SD,
                     tipo = .x,
                     tp = ifelse(nrow(tp) == 0, 0, tp$n),
                     tn = ifelse(nrow(tn) == 0, 0, tn$n),
                     fp = ifelse(nrow(fp) == 0, 0, fp$n),
                     fn = ifelse(nrow(fn) == 0, 0, fn$n)

                   )
                 }) %>%
               reduce(bind_rows)
         }) %>%
```

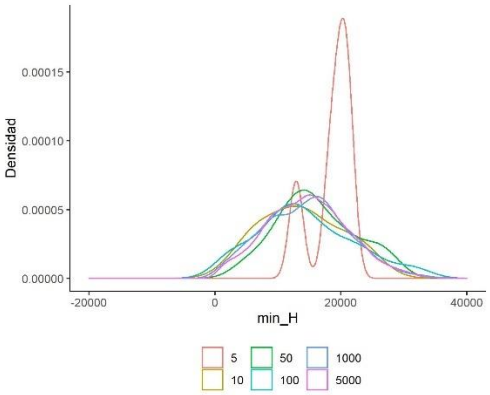
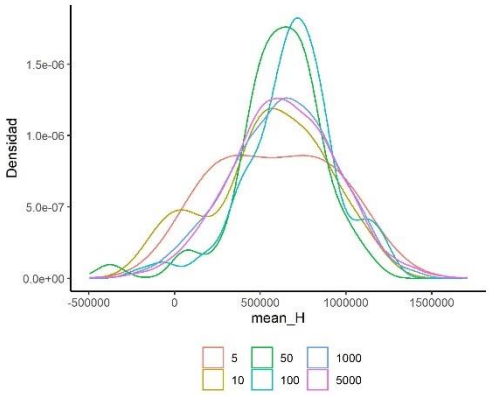
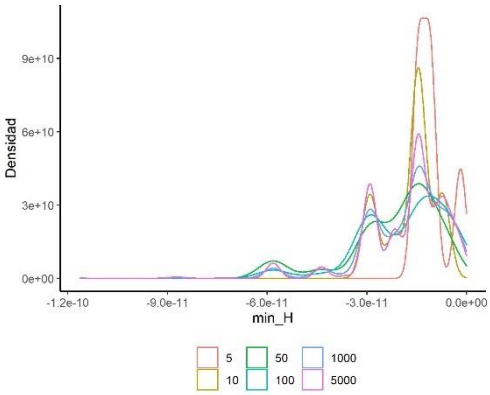
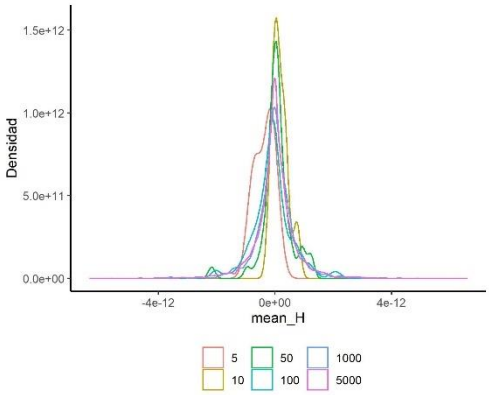
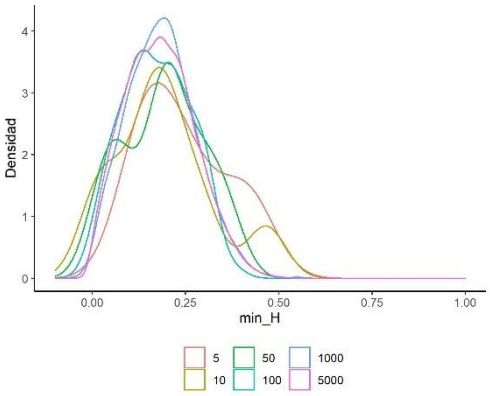
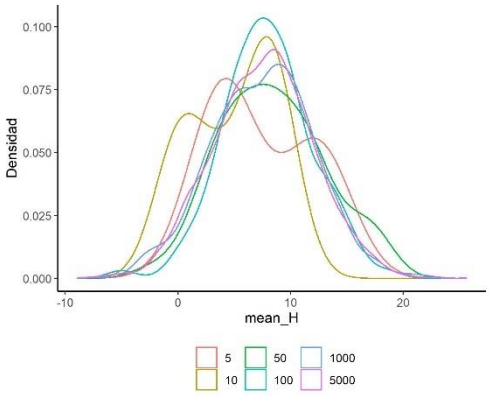
```
  reduce(bind_rows) %>%
  mutate(preci=tp/(tp+fp),
         exhau=tp/(tp+fn),
         f1=2*(preci*exhau)/(preci+exhau))

# Exponencial
res_exp <-
  map2(.x =
         map(res_exp,
             ~ {
               .x %>%
                 filter(!is.na(r2),!is.na(r3)) %>%
                 mutate(r2 = r2 > .95,
                        r3 = r3 > .95)
             }),
       .y = seq(2,18,2),
       .f =
         ~ {
           vec_nam <- c('r1', 'r2', 'r3')
           tabla <- .x
           SD <- .y
           map(.x = vec_nam,
               .f =
                 ~ {
                   tc <-
                     tabla %>%
                     group_by(sm, !!sym(.x)) %>%
                     tally

                   tp <-
                     filter(.data = tc, sm == T & !!sym(.x) == T)
                   tn <-
                     filter(.data = tc, sm == F & !!sym(.x) == F)
                   fp <-
                     filter(.data = tc, sm == F & !!sym(.x) == T)
                   fn <-
                     filter(.data = tc, sm == T & !!sym(.x) == F)

                   tibble(
                     SD = paste0("exp(", SD, ")"),
                     tipo = .x,
                     tp = ifelse(nrow(tp) == 0, 0, tp$n),
                     tn = ifelse(nrow(tn) == 0, 0, tn$n),
                     fp = ifelse(nrow(fp) == 0, 0, fp$n),
                     fn = ifelse(nrow(fn) == 0, 0, fn$n)

                   )
                 }) %>%
             reduce(bind_rows)
         }) %>%
  reduce(bind_rows) %>%
  mutate(preci=tp/(tp+fp),
         exhau=tp/(tp+fn),
         f1=2*(preci*exhau)/(preci+exhau))

# Cobb-Douglas

res_cobb <-
  map2(.x =
         map(res_cobb,
             ~ {
```

```
              .x %>%
                mutate(r2 = r2 > .95,
                       r3 = r3 > .95)
            }),
        .y = 1:8,
        .f =
          ~ {
            vec_nam <- c('r1', 'r2', 'r3')
            tabla <- .x
            SD <- .y
            map(.x = vec_nam,
                .f =
                  ~ {
                    tc <-
                      tabla %>%
                      group_by(sm, !!sym(.x)) %>%
                      tally

                    tp <-
                      filter(.data = tc, sm == T & !!sym(.x) == T)
                    tn <-
                      filter(.data = tc, sm == F & !!sym(.x) == F)
                    fp <-
                      filter(.data = tc, sm == F & !!sym(.x) == T)
                    fn <-
                      filter(.data = tc, sm == T & !!sym(.x) == F)

                    tibble(
                      SD = SD,
                      tipo = .x,
                      tp = ifelse(nrow(tp) == 0, 0, tp$n),
                      tn = ifelse(nrow(tn) == 0, 0, tn$n),
                      fp = ifelse(nrow(fp) == 0, 0, fp$n),
                      fn = ifelse(nrow(fn) == 0, 0, fn$n)

                    )
                  }) %>%
              reduce(bind_rows)
          }) %>%
  reduce(bind_rows) %>%
  mutate(preci=tp/(tp+fp),
         exhau=tp/(tp+fn),
         f1=2*(preci*exhau)/(preci+exhau))


# Lineal

res_lineal <-
  map2(.x =
         map(res_lineal,
             ~ {
               .x %>%
                 mutate(r2 = r2 > .95,
                        r3 = r3 > .95)
             }),
       .y = 1:8,
       .f =
         ~ {
           vec_nam <- c('r1', 'r2', 'r3')
           tabla <- .x
           SD <- .y
```

```
            map(.x = vec_nam,
                .f =
                  ~ {
                    tc <-
                      tabla %>%
                      group_by(sm, !!sym(.x)) %>%
                      tally

                    tp <-
                      filter(.data = tc, sm == T & !!sym(.x) == T)
                    tn <-
                      filter(.data = tc, sm == F & !!sym(.x) == F)
                    fp <-
                      filter(.data = tc, sm == F & !!sym(.x) == T)
                    fn <-
                      filter(.data = tc, sm == T & !!sym(.x) == F)

                    tibble(
                      SD = SD,
                      tipo = .x,
                      tp = ifelse(nrow(tp) == 0, 0, tp$n),
                      tn = ifelse(nrow(tn) == 0, 0, tn$n),
                      fp = ifelse(nrow(fp) == 0, 0, fp$n),
                      fn = ifelse(nrow(fn) == 0, 0, fn$n)

                    )
                  }) %>%
              reduce(bind_rows)
          }) %>%
  reduce(bind_rows) %>%
  mutate(preci=tp/(tp+fp),
         exhau=tp/(tp+fn),
         f1=2*(preci*exhau)/(preci+exhau))


####### Exportar resultados #######

xlsx::write.xlsx(res_exp,file = 'resultados.xlsx',sheetName = 'exp')
xlsx::write.xlsx(res_expexp,file = 'resultados.xlsx',sheetName =
'expexp',append = T)
xlsx::write.xlsx(res_cobb,file = 'resultados.xlsx',sheetName =
'cobb',append = T)
xlsx::write.xlsx(res_lineal,file = 'resultados.xlsx',sheetName =
'lineal',append = T)


####### Graficar #######

# res_exp <-
#   res_exp %>%
#   rowwise() %>%
#   mutate(SD=eval(parse(text=SD))) %>%
#   ungroup()

res_exp %>%
  mutate(across(.cols = everything(),~ifelse(is.nan(.),0,.)),
         SD=factor(SD,levels = paste0('exp(',seq(2,18,2),')'))
         ) %>%
  mutate(tipo=case_when(
    tipo=="r1" ~ "Regresión",
```

```
    tipo=="r2" ~ "Test 1",
    tipo=="r3" ~ "Test 2"
  )) %>%
  ggplot(data = ., aes(x=SD, y=f1, group=tipo)) +
  geom_line(aes(color = tipo)) +
  geom_point(aes(color = tipo)) +
  theme_classic() +
  theme(legend.position="bottom",
        legend.title=element_blank()) +
  labs(x ="Desviación Estándar", y = "Score F1")

ggsave("res_exp.jpg")


res_expexp %>%
  mutate(across(.cols = everything(),~ifelse(is.nan(.),0,.))) %>%
  mutate(tipo=case_when(
    tipo=="r1" ~ "Regresión",
    tipo=="r2" ~ "Test 1",
    tipo=="r3" ~ "Test 2"
  )) %>%
  ggplot(data = ., aes(x=SD, y=f1, group=tipo)) +
  geom_line(aes(color = tipo)) +
  geom_point(aes(color = tipo)) +
  theme_classic() +
  theme(legend.position="bottom",
        legend.title=element_blank()) +
  labs(x ="Desviación Estándar", y = "Score F1")

ggsave("res_expexp.jpg")

res_cobb %>%
  mutate(tipo=case_when(
    tipo=="r1" ~ "Regresión",
    tipo=="r2" ~ "Test 1",
    tipo=="r3" ~ "Test 2"
  )) %>%
  ggplot(data = ., aes(x=SD, y=f1, group=tipo)) +
  geom_line(aes(color = tipo)) +
  geom_point(aes(color = tipo)) +
  theme_classic() +
  theme(legend.position="bottom",
        legend.title=element_blank()) +
  labs(x ="Desviación Estándar", y = "Score F1")

ggsave("res_cobb.jpg")
```

Anexo 5

	$\min_{x',x'' \in X} \widehat{H}(x',x'')$	$\underset{x',x'' \in X}{\text{mean}} \widehat{H}(x',x'')$
$k \exp(x_1 + x_2) + u$		
$k\, x_1^{0.5} x_2^{0.5} + u$		
$x_1 + x_2 + k\, x_1 x_2 + u$		

$$x_1^{k\,x_2} + x_2^{k\,x_1} + u$$