Tesis de Máster

UNIVERSIDAD DEL PAÍS VASCO
EUSKAL HERRIKO UNIBERTSITATEA

UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO UNSAAC

INGENIERÍA COMPUTACIONAL Y SISTEMAS INTELIGENTES

"CONSTRUCCIÓN DE UNA APLICACIÓN INTERACTIVA PARA EL ESTUDIO DE LA OPINIÓN DE EXPERTOS RECOGIDA EN UNA ENCUESTA"

ALUMNO:

Quispe Onofre, Hugo Joel

ASESORA:

Itziar Irigoyen Garbizu

CUSCO-PERÚ

2011

Agradecimientos

En primer lugar agradezco a Dios por darme la oportunidad y la fuerza necesaria para poder alcanzar una nueva meta en mi vida académica y profesional. Agradezco a la "Universidad del País Vasco" (UPV/EHU) y a la "Universidad Nacional San Antonio Abad del Cusco" (UNSAAC), por el soporte institucional dado para la realización del presente trabajo. Agradezco a mi asesora Dra. Itziar Irigoyen Garbizu por su apoyo, colaboración y palabras de aliento para la realización de esta investigación, de igual manera agradezco al Dr. Yosu Yurramendi Mendizabal por sus recomendaciones y observaciones.

No puedo dejar de agradecer a mi familia, a mis padres Hugo D. Quispe Surco, María Reneé Onofre Ibarra, a mis hermanos Carlos Ramón y Edith Roxana, por permitirme soñar y crecer, brindándome su apoyo incondicional.

Abstract

To solve complex problems is frequent to seek the opinion of a group of experts in the field and from there try to get a consensus among their answers. This task is not easy, and hence the Delphi methodology has been designed specifically for this purpose. Generally it reflects the views of experts through surveys in successive rounds until it reaches a consensus response to the problem. This work has developed an interactive tool specifically designed for the study of the expert's opinions. This tool, on the one hand, allows to show the opinions of experts in a particular round of the Delphi methodology, so as to identify where are the agreements and disagreements between the experts, interacting with the tool, and on the other hand, it lets you view the evolution of each expert's opinions over successive rounds measuring the degree of consensus in a multivariated way.

Keywords

Consensus response, Delphi methodology, survey round, interactive tool.

Resumen

Para la resolución de problemas complejos es frecuente recurrir a la opinión de un grupo de expertos en la materia y a partir de ahí obtener una respuesta consensuada entre ellos. Esta tarea no es fácil, pero se cuenta con la metodología Delphi que está diseñada expresamente para este propósito. Generalmente se recogen las opiniones de los expertos a través de encuestas, en sucesivas rondas, hasta llegar a una respuesta consensuada del problema. En el presente trabajo se ha desarrollado una herramienta interactiva específica para el estudio de las opiniones de los expertos. Esta herramienta permite, por una parte, visualizar las opiniones de los expertos en una ronda particular de la metodología Delphi, mediante la cual se puede identificar dónde están los acuerdos y desacuerdos entre los expertos e interactuar con ella; y por otra parte, permite visualizar la evolución de las opiniones de cada experto a lo largo de las sucesivas rondas midiendo el grado de consenso de manera multivariante.

Palabras Claves

Respuesta consensuada, metodología Delphi, ronda de encuestas, herramienta interactiva.

Índice

Índice de Imágenes

Índice de tablas

1. Introducción

A mediados del siglo XX, se planteó el método Delphi con la intención de crear un instrumento capaz de realizar predicciones sobre un caso de catástrofe nuclear. Este método consiste en seleccionar a un grupo de expertos a quienes se les pregunta su opinión sobre cuestiones referidas a acontecimientos del futuro. De esta manera se estudian las respuestas de los expertos en sucesivas rondas, anónimas, al objeto de tratar de conseguir un consenso entre ellos, pero con la máxima autonomía por parte de los participantes. Para el estudio de las respuestas de los expertos habrá un gestor, o usuario final, que tendrá que valorar el grado de acuerdo entre los expertos y mostrar cómo van evolucionando las opiniones de los expertos a lo largo de las sucesivas rondas.

Dada una situación particular (como puede ser la mencionada anteriormente), en la cual se tiene a un grupo de conocedores o expertos de un tema y se recopilan las opiniones de dichos conocedores, se tiene como objetivo medir el grado de acuerdo que muestran los conocedores para lo cual se ve la necesidad de desarrollar una herramienta útil para realizar dicha tarea.

Esta herramienta debe brindar las facilidades de mostrar al gestor salidas gráficas, sencillas y fáciles de interpretar acerca del nivel de consenso de la recopilación de la información de opiniones de los expertos; dando como ventaja, por una parte, estudiar las respuestas de manera multivariante y poder ver de manera sintética los resultados. Y por otra parte, verificar bajo qué circunstancias se observan tanto acuerdos como desacuerdos, tratando de buscar un nivel de consenso entre todos los participantes.

La herramienta desarrollada tiene la característica de interactuar con el gestor, para que pueda simular situaciones hipotéticas sobre las respuestas de los expertos; es decir obtener respuestas simuladas de los participantes buscando un mayor grado de consenso.

La presente memoria está organizada de la siguiente manera: En el apartado 1 se hace una introducción del trabajo realizado, en los apartados 2 y 3 se describen la Metodología Delphi y el análisis de componentes principales, respectivamente. A

continuación, en los apartados 4 y 5, se describe el desarrollo de la aplicación y se ofrece el manual de usuario junto con un caso práctico, respectivamente. Por último, están las conclusiones derivadas de este trabajo y líneas futuras de trabajo.

1.1 Objetivo de la investigación

El objetivo de la investigación es el diseño e implementación de una herramienta interactiva útil para comparar y contrastar opiniones de un grupo de conocedores de un tema en particular, de manera sintética, tratando de llegar a un consenso entre ellos.

Para poder alcanzar el objetivo planteado es necesario investigar técnicas de reducción de datos, investigar teoría de la metodología de Delphi, desarrollar los algoritmos para el caso de estudio y finalmente realizar una demostración del uso de la herramienta desarrollada.

1.2 Metodología de la investigación

Para el presente trabajo de investigación, se ha considerado utilizar las siguientes metodologías de investigación:

- Investigativo: de primer aporte, el proyecto investigará la parte teórica del tema propio del caso de estudio.
- Descriptivo: el proyecto tendrá la intención de realizar una descripción de caso de estudio.
- Aplicativo: El proyecto presentará una herramienta final, la cual podrá ser utilizada para el objetivo del caso de estudio.

2. Método Delphi

El método Delphi consiste en la selección de un grupo de expertos a los que se les pregunta su opinión sobre cuestiones referidas a acontecimientos del futuro. Las estimaciones de los expertos se realizan en sucesivas rondas, anónimas, al objeto de tratar de conseguir consenso, pero con la máxima autonomía por parte de los participantes.

La capacidad de predicción de Delphi se basa en la utilización sistemática de un juicio intuitivo emitido por un grupo de expertos. Es decir, el método Delphi consiste en captar la opinión de expertos con la ayuda de cuestionarios sucesivos, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos. La encuesta se lleva a cabo de una manera anónima para evitar los efectos de "líderes".

Las preguntas se refieren, por ejemplo, a las probabilidades de realización de hipótesis o de acontecimientos con relación al tema de estudio. La calidad de los resultados depende, sobre todo, del cuidado que se ponga en la elaboración del cuestionario y en la elección de los expertos consultados.

En la familia de los métodos de pronóstico, habitualmente se clasifica el método Delphi dentro de los métodos cualitativos o subjetivos, y su formulación teórica comprende varias etapas sucesivas de envíos de cuestionarios, de vaciado y de explotación.

A continuación se describen las fases fundamentales de la metodología Delphi.

2.1 Fases del Método Delphi

El método Delphi sigue una serie de pasos o fases, entre ellas podemos mencionar las siguientes:

Fase 1: formulación del problema

Se trata de una etapa fundamental en la realización del método Delphi. Tratándose de un método de expertos, es importante definir con precisión el campo de investigación a fin de asegurar que todos los expertos reclutados y consultados poseen la misma noción del campo.

La elaboración del cuestionario debe ser llevada a cabo según ciertas reglas: las preguntas deben ser precisas, cuantificables (versan por ejemplo sobre probabilidades de realización de hipótesis y/o acontecimientos, la mayoría de las veces sobre datos de realización de acontecimientos) e independientes.

Fase 2: elección de expertos

La etapa es importante en cuanto que el término de "experto" es ambiguo. Con independencia de sus títulos, su función o su nivel jerárquico, el experto será elegido por su capacidad de encarar el futuro y posea conocimientos sobre el tema consultado.

La falta de independencia de los expertos puede constituir un inconveniente; por esta razón los expertos son aislados y sus opiniones son recogidas por vía postal o electrónica y de forma anónima; así pues se obtiene la opinión real de cada experto y no la opinión más o menos falseada por un proceso de grupo.

Fase 3: Elaboración y lanzamiento de los cuestionarios (en paralelo con la fase 2)

Los cuestionarios se elaborarán de manera que faciliten, en la medida en que una investigación de estas características lo permite, la respuesta por parte de los consultados. Preferentemente las respuestas habrán de poder ser cuantificadas y

ponderadas (año de realización de un evento, probabilidad de realización de una hipótesis, valor que alcanzará en el futuro una variable o evento,...).

Las respuestas del cuestionario generalmente son tipo *Likert*, donde el experto tiene que dar una puntuación siendo PMIN "totalmente en desacuerdo" y PMAX "totalmente de acuerdo" (PMIN < PMAX). Los valores de PMIN y PMAX se establecen en cada caso. Sin embargo, en ocasiones se recurre a respuestas categorizadas (Si/No; Mucho/Medio/Poco; Muy de acuerdo/ De acuerdo/ Indiferente/ En desacuerdo/Muy en desacuerdo).

Fase 4: Desarrollo práctico y explotación de resultados

En cuanto al desarrollo práctico, el cuestionario es enviado a cierto número de expertos (hay que tener en cuenta las no-respuestas y abandonos. Se recomienda que el grupo final no sea inferior a 25 participantes). El cuestionario va acompañado por una nota de presentación que precisa las finalidades, el espíritu del Delphi, así como las condiciones prácticas del desarrollo de la encuesta (plazo de respuesta, garantía de anonimato). Además, en cada cuestión, puede plantearse que el experto deba evaluar su propio nivel de competencia.

En cuanto a la explotación y estudio de los resultados, las técnicas estadísticas que se utilizan dependerán del tipo de respuesta del cuestionario. Es decir, para respuestas cuantitativas tipo *Likert* (ver **Fase 3**), se utilizan, por una parte, los estadísticos de tendencia central para medir la opinión media de los expertos; y por otra parte, para medir el grado de acuerdo se utilizan los estadísticos de dispersión, tales como la desviación estándar y el rango intercuartil. Para respuestas cualitativas, se estudian las respuestas en términos porcentuales tratando de ubicar a la mayoría de los consultados en una categoría. Cabe destacar que en ambas casos el tratamiento de las respuestas mencionado es univariante.

En el presente trabajo se ha diseñado y desarrollado una herramienta para el estudio y explotación de las respuestas obtenidas a través de los cuestionarios. Es decir, esta herramienta será útil en la mencionada **Fase 4** de la metodología. La herramienta permite, por una parte analizar una vuelta de las encuestas,

donde se podrá ver el grado de consenso inicial entre los participantes; en esta parte el gestor podrá interactuar con la herramienta para poder obtener nuevas respuestas basadas en situaciones hipotéticas. Por otra parte, la herramienta permitirá realizar el análisis de un grupo de encuestas de diferentes rondas, pudiendo ver el grado de consenso en cada ronda junto a la evolución de las opiniones. Además, la herramienta está diseñada únicamente para el estudio de variables cuantitativas pero se quiere mencionar que el tratamiento de las respuestas es multivariante, es decir, se consideran simultáneamente todas las respuestas de cada experto.

2.2 Ejemplo del método Delphi

Un ejemplo del uso de la metodología Delphi es la tesis intitulada: "Métodos de prospección. El método Delphi en la ingeniería Civil (Un camino para la prevención tecnológica)" de la Universidad Nacional de Ingeniería (Perú), donde se muestra la aplicación de la misma en la ingeniería Civil con ejemplos sobre la elección de cargas de diseño y daños en edificios. [Walter G. Meléndez Bernardo, 2009]

Algunos estudios realizados, en distintas disciplinas, son los siguientes:

Agroalimentario:

Tecnologías de Conservación de Alimentos.

La biotecnología aplicada al sector alimentario.

Tecnologías en el envasado agroalimentario.

Energía:

Energías Renovables.

Tecnologías avanzadas de conversión de combustibles fósiles.

Tendencias tecnológicas en transporte, distribución, almacenamiento y uso final de la energía.

Medio Ambiente Industrial

Gestión y Tratamiento de residuos industriales.

Bienes de equipo medio ambientales.

Tratamientos de aguas industriales.

Químico

Química Fina.

Química Básica Orgánica. Primeras Materias Plásticas.

Agroquímica.

Pasta, Papel y Cartón.

Tecnologías de la Información y las Comunicaciones

Industria de contenidos digitales.

Las TIC y la emergente economía digital.

Convergencia de infraestructuras y servicios en el sector de las telecomunicaciones.

Transportes

Aeronáutico.

Ferrocarril.

Naval.

Automoción.

Sectores Básicos y Transformadores

Tecnologías de Fabricación de Productos Metálicos.

Tecnologías de transformación de piezas de plásticos y materiales compuestos.

Bienes de equipo para la fabricación de piezas unitarias.

Sectores Tradicionales

Tecnologías de Diseño.

Tecnologías de automatización.

Tecnologías limpias y de reciclaje.

3. Métodos para el análisis de las respuestas

Como se ha mencionado anteriormente, la herramienta que se ha desarrollado está diseñada para el estudio de respuestas cuantitativas de la encuesta. Puesto que el objetivo es estudiar estas respuestas de manera multivariante para medir el grado de acuerdo entre los expertos, se ha utilizado el Análisis de Componentes Principales para tal fin. A continuación se explica en qué consiste esta técnica.

3.1 Análisis de componentes principales

El Análisis de Componentes Principales es una técnica estadística de síntesis de la información, o reducción de la dimensión (número de variables). Es decir, ante un banco de datos con muchas variables, el objetivo será reducirlas a un menor número perdiendo la menor cantidad de información posible. Para ello se construye una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos en el cual la varianza de mayor tamaño del conjunto de datos es capturada en el primer eje (llamado el Primer Componente Principal), la segunda varianza más grande es el segundo eje, y así sucesivamente. Es decir, los nuevos componentes principales o factores serán una combinación lineal de las variables originales, y además estarán incorrelacionadas entre ellas. A continuación pasamos a ver cómo se construye este proceso del cálculo de componentes principales.

3.2 Definición

El ACP construye una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos en el cual la varianza de mayor tamaño del conjunto de datos es capturada en el primer eje (llamado el Primer Componente Principal), la segunda varianza más grande es el segundo eje, y así sucesivamente. Para construir esta transformación lineal debe construirse primero la matriz de covarianza o matriz de coeficientes de correlación. Debido a la simetría de esta matriz existe una base completa de vectores propios de la misma. La transformación que lleva de las antiguas coordenadas a las coordenadas de la nueva base es precisamente la transformación lineal necesaria para reducir la dimensionalidad de datos. Además las coordenadas en la nueva base dan la composición en factores subyacentes de los datos iníciales.

Una de las ventajas de ACP para reducir la dimensionalidad de un grupo de datos, es que retiene aquellas características del conjunto de datos que contribuyen más a su varianza, manteniendo un orden de bajo nivel de los componentes principales e ignorando los de alto nivel. El objetivo es que esos componentes de bajo orden a veces contienen el "más importante" aspecto de esa información.

3.3 Pasos del Análisis de Componentes principales

Supongamos que para la encuesta con p preguntas la matriz X recoge las respuestas de los n expertos, una vez centradas. La construcción de los componentes se hace uno a uno, es decir, se empieza con la construcción del primer componente principal. El cálculo del **primer componente principal** C_1 será:

- $C_{i1} = u_{11}X_{i1} + \ldots + u_{p1}X_{ip}, i = 1, \ldots, n$

Y el vector de coeficientes u_1 será aquel que de la dirección que mejor aproxima a los datos según la suma de los cuadrados de las distancias entre los datos originales y las proyecciones sobre esta dirección. Dicho de otra manera, las proyecciones de los datos sobre esta dirección deben de contener la máxima variabilidad posible. Por tanto, la dirección u_1 será la dirección que maximice:

$S^2(C_1) = \frac{1}{n-1} C'_1 C_1 = \frac{1}{n-1} u'_1 X'X u_1 = u'_1 S u_1$, donde S es la matriz de varianzas-covarianzas de los datos originales X. Como el objetivo es encontrar la mejor dirección en el sentido mencionado anteriormente, se busca el vector u_1 de norma 1. La solución es el vector propio de S asociado al mayor de los valores propios.

Análogamente se construye el segundo componente principal, pero ahora, se añade que la nueva dirección u_2 que buscamos no esté correlacionada con la ya construida u_1. Otra vez, la solución a este problema es tomar u_2 como el vector propio de S asociado al segundo mayor valor propio. Este proceso permite construir p componentes principales de manera que la matriz de componentes principales viene dada como:

$C = XT$, donde $T = (u_1, u_2, \ldots, u_p)$ es la matriz ortogonal formada por los p vectores propios de la matriz de varianzas – covarianzas S. Llamaremos $\lambda_1 \ldots \lambda_p$ a los respectivos valores propios, donde $(\lambda_1 \geq \cdots \geq \lambda_p)$.

Por tanto calcular los componentes principales equivale a aplicar una rotación/reflexión a los datos originales.

Además, el espacio de dimensión q < p que mejor representa los datos es el definido por los vectores propios asociados a los q mayores valores propios de *S*, es decir, los primeros q componentes principales. Esto nos lleva, por una parte, a poder representar la información contenida en los datos originales en un especio de dimensión menor; pero por otra parte, también conlleva una pérdida de información. Esto hace que sea necesario medir qué porcentaje de variabilidad se retiene con los q componentes retenidos. Para ello, primero se cuantifica la variabilidad total de los datos originales como:

$$S^2(x_1) + \ldots + S^2(x_p) = traza(S) = (\lambda_1 + \lambda_2 + \ldots + \lambda_p)$$

Luego, se mide la variabilidad que retienen los q componentes principales como:

$$S^2(C_1) + \ldots + S^2(C_q) = (\lambda_1 + \lambda_2 + \ldots + \lambda_q)$$

Podemos calcular el porcentaje de la variabilidad que retienen los primeros q componentes como:

$$(\lambda_1 + \lambda_2 + \ldots + \lambda_q)/(\lambda_1 + \lambda_2 + \ldots + \lambda_p)$$

Por último, cabe destacar que cuando las variables originales no son conmensurables es adecuado estandarizar las variables originales antes de aplicar el análisis de componentes principales. En este caso, la construcción de los correspondientes componentes principales equivale a rehacer el proceso anterior utilizando la matriz de correlaciones en lugar de la matriz de varianzas-covarianzas.

3.4 Resultados relevantes para la implementación de la herramienta

1- Obtención de opiniones para situaciones hipotéticas del plano factorial.

Supongamos que las opiniones de los expertos están en la matriz que llamaremos *O*. El usuario final de la aplicación, dependiendo de la situación que le ocupa considerará alguna de las siguientes transformaciones antes de aplicar el análisis de componentes principales:

a) Cuando las p variables son conmensurables

$$X = f_1(O)$$
$$= O - \mathbf{1}\bar{o}',$$

donde $\bar{o}$ es el vector de medias. X es por tanto la matriz de datos centrada.

b) Cuando las p variables no son conmensurables

$$X = f_2(O)$$
$$= (O - \mathbf{1}\bar{o}')\, {D_s}^{-1},$$

donde D_s es la matriz diagonal con las desviaciones estándares de las variables originales. *X* es, por tanto, la matriz de datos estandarizada.

Sea *T* la matriz de rotación que construye los componentes principales y $C = XT$ los componentes principales. Pongamos que el usuario final de la herramienta considera el plano factorial formado por los componentes C_1 y C_2 y que quiere conocer qué opinión se corresponde para un punto hipotético (c_{01}, c_{02}) del plano factorial. Podemos estimar que la opinión correspondiente como:

a) Considerar $\boldsymbol{c_0} = (c_{01}, c_{02}, 0, \dots, 0)'$

b) $x_{0'} = \boldsymbol{c_0}' T'$

c) $\boldsymbol{o}_0 = f_i^{-1}(x_0)$, con *i* = 1 ó 2, dependiendo de la transformación inicial de los datos que se haya hecho.

2- Comparación de las opiniones en sucesivas vueltas

Supongamos que las opiniones de los expertos se han recogido en k sucesivas vueltas o rondas, y sean $O_1, \dots, O_k$ las matrices de datos correspondientes. Pongamos que las opiniones de la primera ronda (O_1) se han establecido como

"opiniones de base". Sea $X_1 = f_i(O_1)$ con $i = 1$ ó 2 la transformación adecuada en este caso. Entonces, $X_l = f_i(O_l)$ para $l = 2, \ldots, k$ de la misma manera.

Se calcula la matriz ortogonal T para los componentes principales de X_1, con $C^{(1)} = X_1 T$, y por tanto, $C^{(l)} = X_l T$ para ($l = 2, \ldots,$ k) contiene las opiniones de los expertos en el resto de las rondas tomando como referencia las opiniones en la primera vuelta.

3- Cuantificación de la variabilidad entre las opiniones de los expertos

Puesto que las opiniones de los expertos se van a representar según los q componentes principales, la variabilidad de las opiniones en una ronda particular se medirá como la suma de los q primeros valores propios.

En caso de comparación de las opiniones en sucesivas vueltas, se medirá la variabilidad de las opiniones de la vuelta de referencia como $\lambda_1 + \ldots + \lambda_q$; y la variabilidad de *l*-esima vuelta como $S^2{}_{C^{(l)}{}_1} + \ldots + S^2{}_{C^{(l)}{}_q}$.

4. Aplicación

4.1 Funcionalidad

El presente proyecto tiene como finalidad desarrollar una herramienta con las características de poder recopilar información externa basada en una ronda de encuestas de un grupo de expertos; el programa soporta desde una encuesta hasta un grupo de éstas llamándolas "vueltas". De igual manera la herramienta tiene la propiedad de mostrar al usuario el grado de consenso que existe entre los participantes basado en la lógica de reducción de variable, gracias al análisis de componentes principales; mostrándole al usuario una salida gráfica y una matriz de interpretación donde el usuario podrá reconocer el grado de acuerdo. Por otra parte el programa también tiene la propiedad de interactuar con el usuario para poder ver cómo sería factible un mayor acuerdo entre los participantes.

Una característica importante de la herramienta es que ésta es interactiva con el usuario cuando se realiza el análisis de una única encuesta, es decir, que se puede llegar a manipular la información y ser modificada desde una interfaz gráfica la cual permite producir nuevas situaciones hipotéticas de la información a priori, buscando simular un mayor consenso entre los participantes de las encuestas.

Para el desarrollo de la herramienta interactiva se eligió un conjunto de lenguajes de programación y herramientas estadísticas en los cuales se pudo plasmar el objetivo del presente proyecto. La herramienta final fue desarrollada sobre una plataforma compatible con el sistema operativo Windows.

4.2 Lenguajes de programación

Entre los lenguajes de programación que se han considerado para el desarrollo de la herramienta interactiva, para luego elegir entre ellos, se tienen los siguientes:

Java.

Java es un lenguaje de programación orientado a objetos, desarrollado por SUN Microsystems a principios de los años 90. El lenguaje de programación Java tiene las siguientes características:

- Orientado a Objetos.
- Permite ejecutar programas en diferentes Sistemas Operativos.
- Incluye soporte para trabajo en Red.
- Se puede ejecutar remotamente de forma segura.
- Fácil de usar como otros lenguajes orientados a Objetos, como C++.

Delphi.

Cabe resaltar que el "lenguaje de programación Delphi" es diferente al "método Delphi" que se considera como método de estudio para el presente trabajo.

Delphi es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual. En Delphi se utiliza como lenguaje de programación una versión moderna de Pascal llamada Object Pascal. Es producido comercialmente por la empresa estadounidense CodeGear (antes lo desarrollaba Borland), adquirida en mayo de 2008 por Embarcadero Technologies, una empresa del grupo Thoma Cressey Bravo. En sus diferentes variantes, permite producir archivos ejecutables para Windows, GNU/Linux y la plataforma .NET.

El lenguaje de programación Delphi tiene las siguientes características:

- Soporta la programación Orientada a Objetos.
- Encapsulamientos, como Public, Private y otros.
- Simplificación de Sintaxis de referencias de clases y punteros.

- Programación activada por eventos, posible gracias a la delegación de eventos.

Visual estudio C#.

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

El lenguaje de programación Visual Estudio C# tiene las siguientes características:

- Orientado a Objetos.
- Facilidad de uso del Lenguaje.
- Encapsulamiento, herencia y polimorfismo.
- Orientado a Componentes, permite crear propiedades, atributos y eventos.
- Gestión automática de Memoria, recolector de Basura.

Entre las herramientas de naturaleza estadística se han considerado:

R

R es un lenguaje y entorno de computación para gráficos estadísticos. Es un proyecto de GNU, que es similar al lenguaje S y fue desarrollado en los Laboratorios Bell (antes AT & T, ahora Lucent Technologies) por John Chambers y sus colegas.

R es un conjunto integrado de servicios de software para la manipulación de datos, cálculo y representación gráfica. Incluye:

- Un eficaz manejo de datos y almacenamiento.
- Un conjunto de operadores para el cálculo de matrices, matrices en particular.

- Una colección grande, coherente e integrada de herramientas intermedias para el análisis de datos, facilidades gráficas para el análisis y visualización de datos ya sea en pantalla o en papel.
- Una programación sencilla y eficaz del lenguaje que incluye condicionales, bucles, definida por el usuario y las funciones recursivas de entrada y salida de las instalaciones.

4.3 Diagramas de casos de uso

Para el desarrollo de la herramienta, se ha considerado el siguiente diagrama de casos de usos:

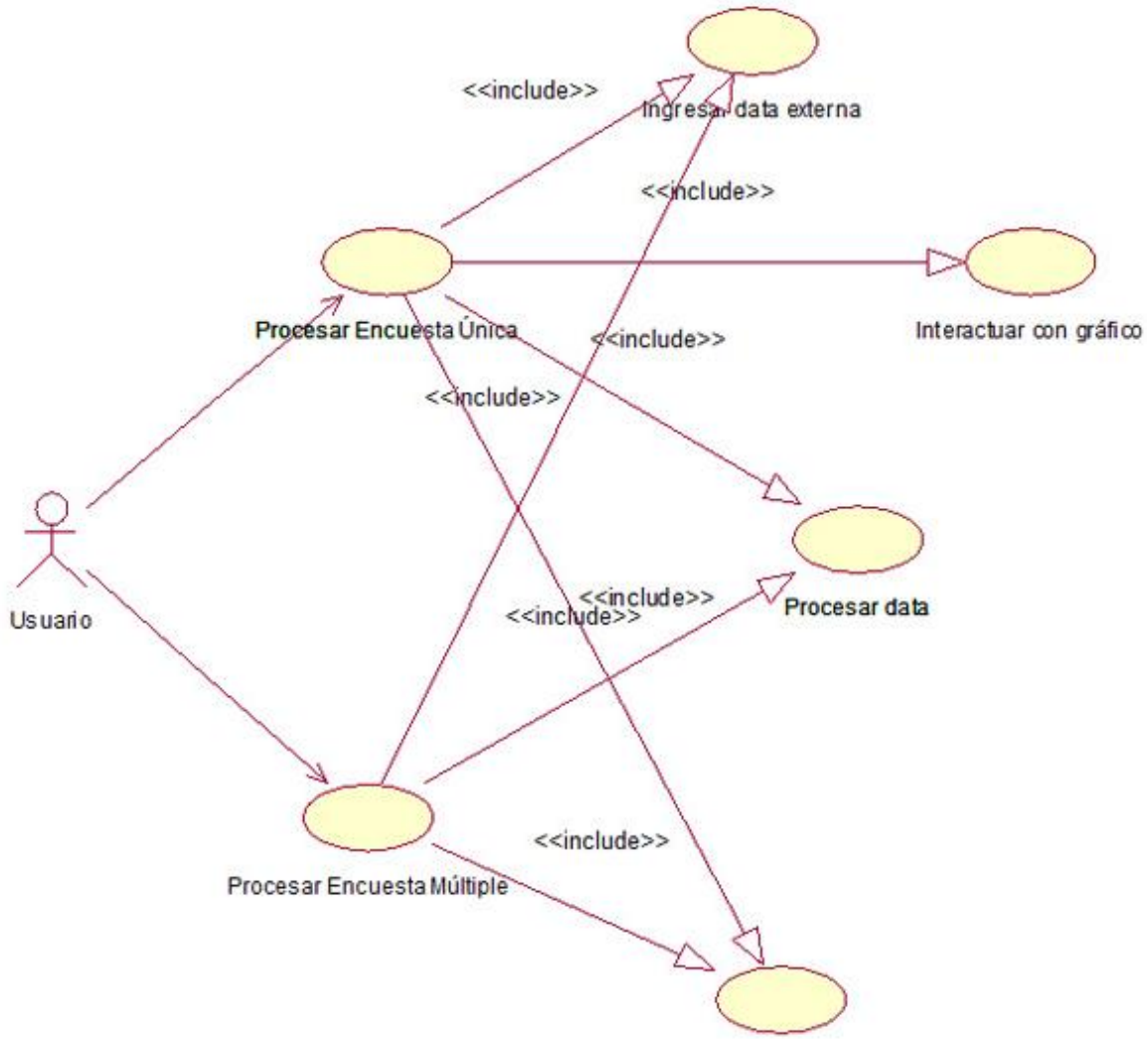

Figura 4.1 - Diagrama de casos de uso

4.4 Diagramas de caso de uso expandido

A continuación se especifica los casos de uso de manera expandida, es decir, detallando cada uno de ellos, de manera que se pueda tener un mejor entendimiento sobre ellos.

Caso de Uso	Procesar encuesta única
Actores	Usuario
Propósito	Procesar una encuesta cargada desde un archivo exterior.
Resumen	Una vez ingresada data desde un archivo exterior, el usuario podrá procesar la información para poder visualizar un gráfico final del grado de consenso entre los participantes, de igual manera podrá interactuar con dicha información para obtener nuevas situaciones basada en la información a priori.
Pre condiciones	Ingresar un archivo exterior con resultados obtenidos de una encuesta basada en el método Delphi. Verificar que la información ingresada sea coherente.
Post condiciones	Guardar la información cambiada para poder visualizarla en un futuro.
Flujo principal	1. (Usuario) Selecciona un archivo externo donde haya información de la encuesta basada en el método Delphi. 2. (Sistema) Reconoce si la información es coherente y procesa la información. 3. (Usuario) Selecciona si desea trabajar con la información estandarizada o como fue ingresada, si el usuario desea puede cambiar la información y poner nuevos valores. 4. (Sistema) Si es necesario, reprocesa la información ingresada que haya sido modificada por el usuario y muestra los componentes principales. 5. (Usuario) Selecciona que componentes principales. 6. (Usuario) Presiona el botón de interacción con el gráfico para poder simular nuevas situaciones sobre la encuesta ingresada. 7. (Sistema) Actualiza los datos que el usuario cambia al interactuar con el sistema.

	8. (Sistema) Muestra la variabilidad de los datos originales y los datos cambiados por el usuario al interactuar con el sistema. 9. (Usuario) Guarda la data en un archivo externo.

Tabla 4.1 - Caso de uso Procesar encuesta única

Caso de Uso	Procesar encuesta múltiple
Actores	Usuario
Propósito	Procesar una ronda de encuestas desde archivos exteriores independientemente por cada encuesta.
Resumen	Una vez ingresada las datas exteriores al sistema, el usuario podrá modificar y cambiar la posición de las datas, teniendo como resultado el grado de consenso en cada vuelta reflejada en las datas cargadas, de igual manera el usuario podrá visualizar la variabilidad por cada vuelta realizada en las encuestas.
Pre condiciones	Ingresar archivos exteriores con resultados obtenidos de una ronda de encuestas basada en el método Delphi. Verificar que la información ingresada sea coherente.
Post condiciones	El usuario puede seguir ingresando más vueltas de las encuestas como también eliminarlos del sistema.
Flujo principal	1. (Usuario) El usuario ingresa el conjunto de datos que desea procesar para visualizar el grado de consenso entre las encuestas ingresadas. 2. (Sistema) Verifica que la información sea coherente y procesa la información. 3. (Usuario) Escoge si desea realizar el proceso sobre la información estandarizada o tal como ha sido ingresada. El usuario puede modificar la información de cualquiera de las datas cargadas en el sistema; de igual manera el usuario puede modificar la posición de las vueltas eligiendo cual desea que sea la primera vuelta de las encuestas. 4. (Sistema) Si es necesario, el sistema vuelve a reprocesar la información cambiada por el usuario y muestra los componentes principales. 5. (Usuario) Elige sobre que componentes principales desea trabajar. 6. (Sistema) Muestra la salida gráfica y la variabilidad de cada vuelta que se haya realizado en la ronda de encuestas.

Tabla 4.2 - Caso de uso Procesar encuesta múltiple

Caso de Uso	Ingresar data externa
Actores	Usuario
Propósito	Ingresar al sistema data sobre las encuestas realizadas.
Resumen	El usuario selecciona desde un archivo exterior la data que contiene información sobre las encuestas basadas en el método Delphi.
Pre condiciones	La data debe estar basada en el método Delphi.
Post condiciones	La data es verificada y procesada.
Flujo principal	1. (Usuario) Elige un archivo exterior, desde un explorador de carpetas, la data con información basada en el método Delphi. 2. (Sistema) Verifica que la información sea coherente y lo muestra en la interfaz del sistema. 3. (Usuario) En caso se trate de encuestas múltiples, el usuario puede cargar y retirar datas (información de las encuestas) del sistema. 4. (Sistema) Actualiza la data.

Tabla 4.3 - Caso de uso Ingresar data externa

Caso de Uso	Procesar data
Actores	Usuario
Propósito	Procesar la información ingresada desde archivo(s) exterior(es) para poder visualizar el grado de consenso de los participantes de las encuestas.
Resumen	El sistema verifica que el usuario haya modificado la información ingresada y reprocesa la data, muestra la matriz de interpretación y los componentes principales que el usuario erigirá para ver la gráfica final y la variabilidad.
Pre condiciones	Debe existir data coherente (información de las encuestas) ingresada desde archivos exteriores.
Post condiciones	El sistema muestra la matriz de interpretación y los componentes principales. El usuario puede volver a reprocesar la información.
Flujo principal	1. (Sistema) Procesa la información obteniendo la matriz de interpretación y los componentes principales para mostrar el gráfico final. 2. (Usuario) Selecciona si desea trabajar sobre la información estandarizada o tal como fue ingresada, de igual manera puede modificar las datas con valores nuevos. 3. (Sistema) El sistema reconoce que el usuario ha modificado la información y la reprocesa obteniendo una nueva matriz de interpretación y nuevos componentes principales. 4. (Usuario) El usuario elige sobre que componentes principales desea mostrar el gráfico final de consenso.

Tabla 4.4 - Caso de uso Procesar data

Caso de Uso	Visualizar gráfico final
Actores	Usuario
Propósito	Poner visualizar el gráfico de consenso sobre la data ingresada de archivos exteriores.
Resumen	El usuario elige la opción de poder visualizar el gráfico final sobre la información que ha sido ingresada y reprocesar la misma.
Pre condiciones	La información ingresada debió ser procesada y mostrar los componentes principales sobre los cuales se mostrará el gráfico. En caso la información haya sido modificada, el sistema re procesará los datos.
Post condiciones	El usuario puede guardar la gráfica.
Flujo principal	1. (Usuario) Presiona el botón para visualizar la gráfica final. 2. (Sistema) Muestra la gráfica final basada en los componentes principales seleccionados por el usuario. 3. (Usuario) Si desea, puede guardar la gráfica obtenida para posteriores comparaciones.

Tabla 4.5 - Caso de uso Visualizar gráfico final

Caso de Uso	Interactuar con el gráfico
Actores	Usuario
Propósito	Poder interactuar con el gráfico final y modificar la información para obtener situaciones hipotéticas.
Resumen	Una vez procesada la información, el usuario puede interactuar con los resultados directamente con el gráfico obteniendo nueva información sobre la data ingresada.
Pre condiciones	La información debió estar procesada, en caso el usuario haya modificado la data se reprocesa la información.
Post condiciones	El usuario puede guardar la información modificada.
Flujo principal	1. (Usuario) Presiona el botón de interactuar con la salida gráfica y puede jugar con los resultados. 2. (Sistema) Reconoce los cambios realizados por el usuario y actualiza constantemente la data, de igual manera reconoce cual es el nuevo valor de la variabilidad de los datos. 3. (Usuario) Si desea, puede guardar la nueva data en un archivo externo para poder trabajar sobre este en un futuro.

Tabla 4.6 - Caso de uso Interactuar con el gráfico

4.5 Diagramas de flujo

A continuación se detalla el diagrama de flujos donde, como su nombre indica, se refleja el flujo, o pasos que la herramienta realiza al interactuar con el usuario.

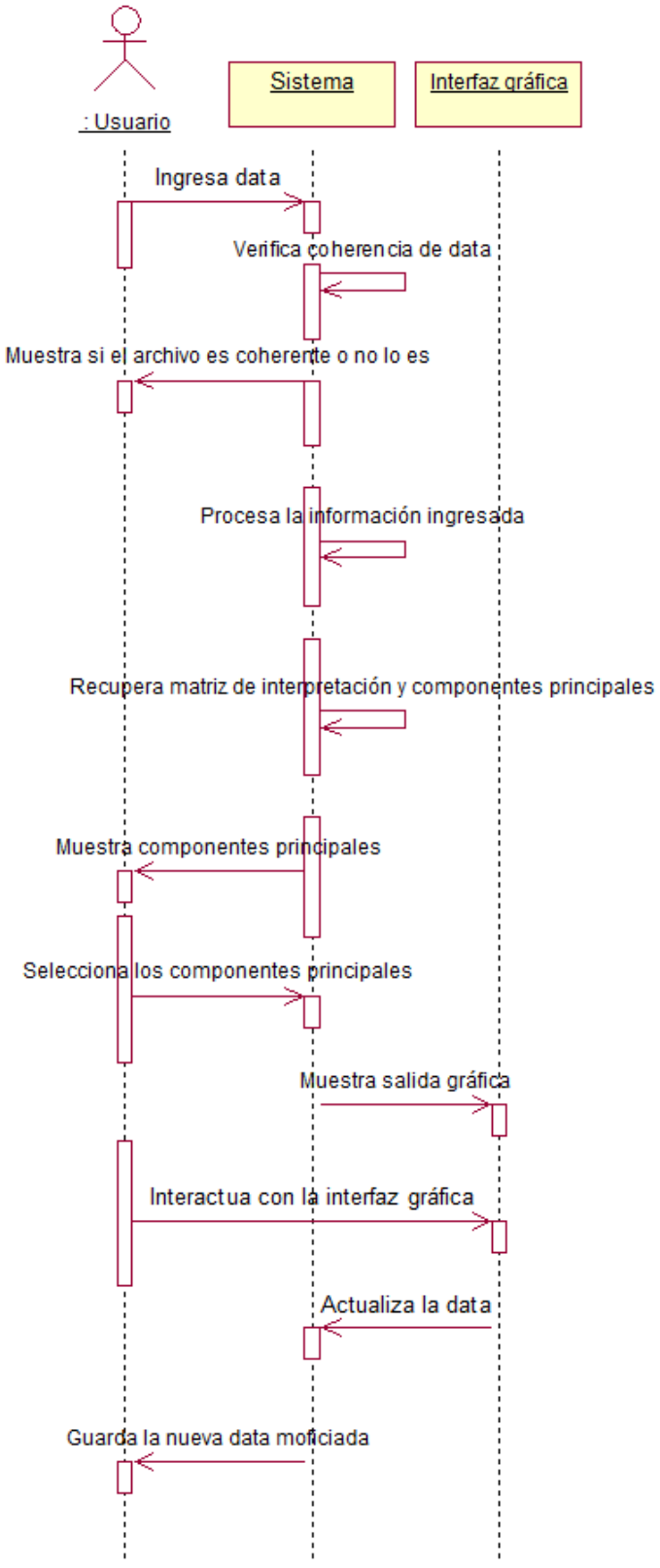

Figura 4.2 - Diagrama de Flujos

4.6 Dependencias entre paquetes

En la dependencia de paquetes, se especifican los paquetes, en este caso clases e interfaces, que forman parte de la herramienta. Las clases están formadas por código fuente donde se desarrolla la funcionalidad de la herramienta, mientras que las interfaces son las encargadas de interactuar con el usuario final, recopilando información y mostrando resultados.

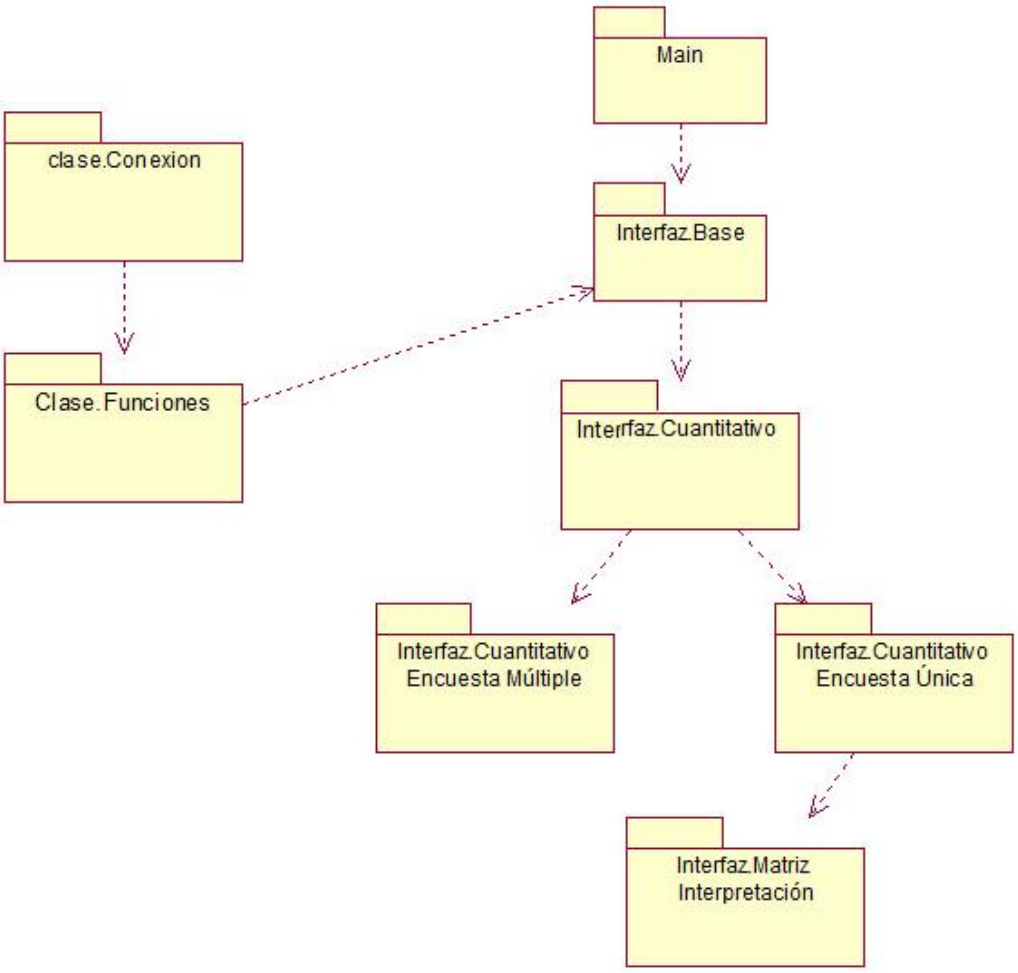

Figura 4.3 - Dependencias de paquetes

Clase conexión: clase básica que se utiliza con el fin de poder realizar una conexión entre la herramienta estadística R y el lenguaje de programación C# gracias a la librería de RCOM. De igual manera en la clase conexión podemos realizar las funciones básicas de interacción entre las dos herramientas como ejecutar, recuperar, asignar y graficar comando propios de R (Anexo I).

Clase funciones: clase que interactúa directamente con las interfaces finales. Esta clase permite realizar las siguientes funciones (Anexo II):

- Re direccionar la ubicación de la carpeta principal de R.
- Recuperar una matriz desde un archivo externo.
- Crear y filtrar una matriz para luego ser procesada.
- Procesar el análisis de componentes principales y recuperar la matriz de interpretación y sus componentes principales.
- Identificar que componentes principales ha seleccionado el usuario.
- Graficar los resultados usando los comandos plot, barplot, boxplot y ggplot en caso se desee interactuar con los gráficos.

Main: Interfaz principal donde el usuario puede seleccionar cuales de las opciones desea procesar, si desea realizar un análisis de encuesta única o un análisis de encuesta múltiple sobre datos cuantitativos.

Interfaz Base: Interfaz padre con características básicas (botón de cancelar y borrar). Este formulario será heredado por otras interfaces a fin de ahorrar y optimizar código fuente y poder realizar en un futuro, un adecuado mantenimiento del sistema.

Interfaz Cuantitativo: Interfaz heredado de la "interfaz base" donde existen características comunes de las interfaces de encuesta única y encuesta múltiple como por ejemplo la selección de un archivo exterior donde se visualiza la data en un componente llamado "DATAGRID" propio del lenguaje de programación elegido. Esta interfaz también cumple el papel de interfaz padre para la encuesta única y encuesta múltiple, siendo ventajoso para realizar mantenimiento del sistema.

Interfaz Cuantitativo encuesta única: Interfaz final para el usuario, heredada de la "interfaz Cuantitativa", donde este puede llegar a realizar un análisis de encuesta única seleccionando un archivo exterior y procesando la información. El resultado son salidas gráficas estáticas e interactivas donde el usuario puede simular nuevas situaciones de las encuestas.

Interfaz Cuantitativo encuesta múltiple: Interfaz final para el usuario heredada de la "interfaz Cuantitativa", donde este puede llegar a realizar un análisis de encuestas de múltiples archivos externos y procesar la información. El resultado son salidas gráficas de la evolución de la información y la variabilidad de la información a través de cada vuelta reconocida de los archivos exteriores.

Interfaz matriz de interpretación: Interfaz donde muestra al usuario una matriz de interpretación en la cual, como su nombre lo indica, el usuario puede llegar a interpretar del porque del consenso o el desacuerdo entre los participantes de las encuestas.

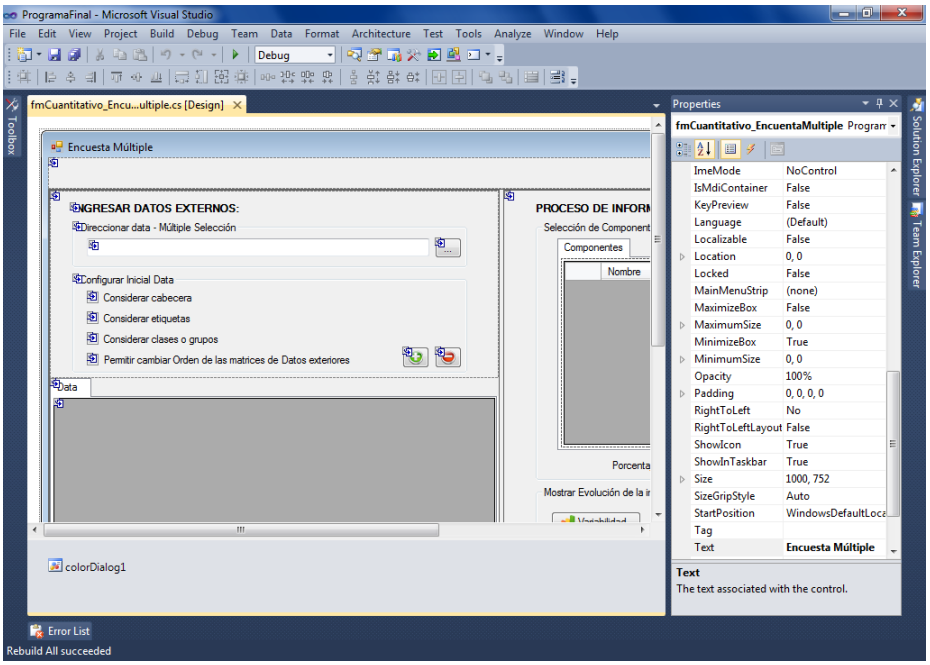

Figura 4.4 - Interfaz de desarrollo de la herramienta interactiva

5. Diseño fundamental y Manual de usuario

5.1 Interfaz de usuario

Se tiene las siguientes interfaces propias de la herramienta iterativa desarrollada:

Interfaz principal: Es la pantalla principal donde se podrá visualizar los diferentes tipos de análisis cuantitativo de las encuestas, ya sea de encuesta única o de encuesta múltiple.

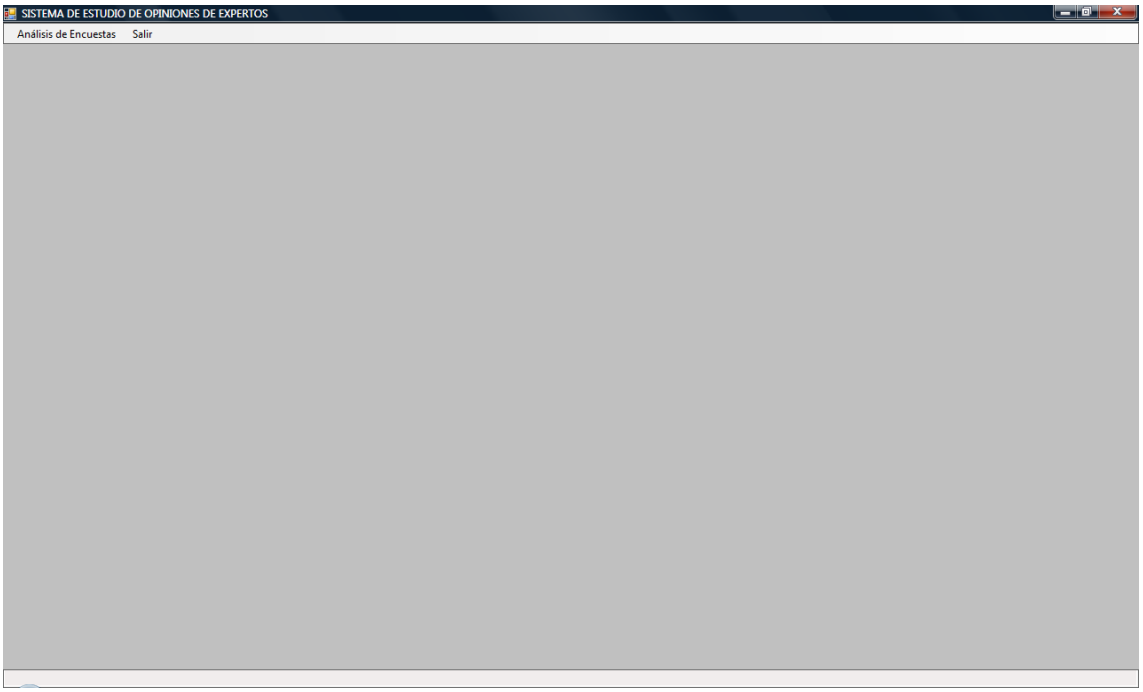

Figura 5.1 - Pantalla principal

Interfaz encuesta única: Es la pantalla donde se puede analizar y procesar los datos de una encuesta única en particular, obteniendo la matriz de interpretación, componentes principales, la gráfica final de consenso, interacción con la gráfica y la variabilidad después de haber modificado la data. Al final el usuario puede guardar la información modificada.

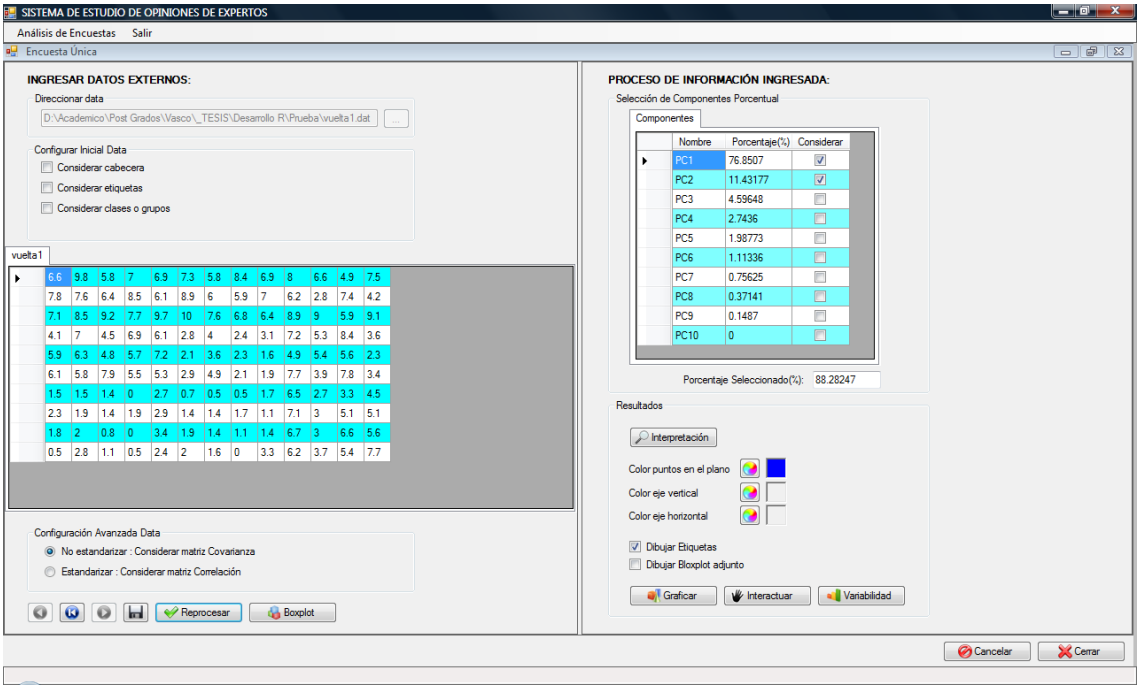

Figura 5.2 - Pantalla encuesta única

Interfaz encuesta múltiple: Es la pantalla donde se puede analizar y procesar los datos de múltiples encuestas, donde el usuario puede seleccionar cual encuesta es la primera vuelta y agregar y/o quitar nuevas encuestas. Se obtiene los componentes principales de la primera vuelta, la gráfica final de consenso de las diferentes vueltas y la variabilidad de las diferentes vueltas.

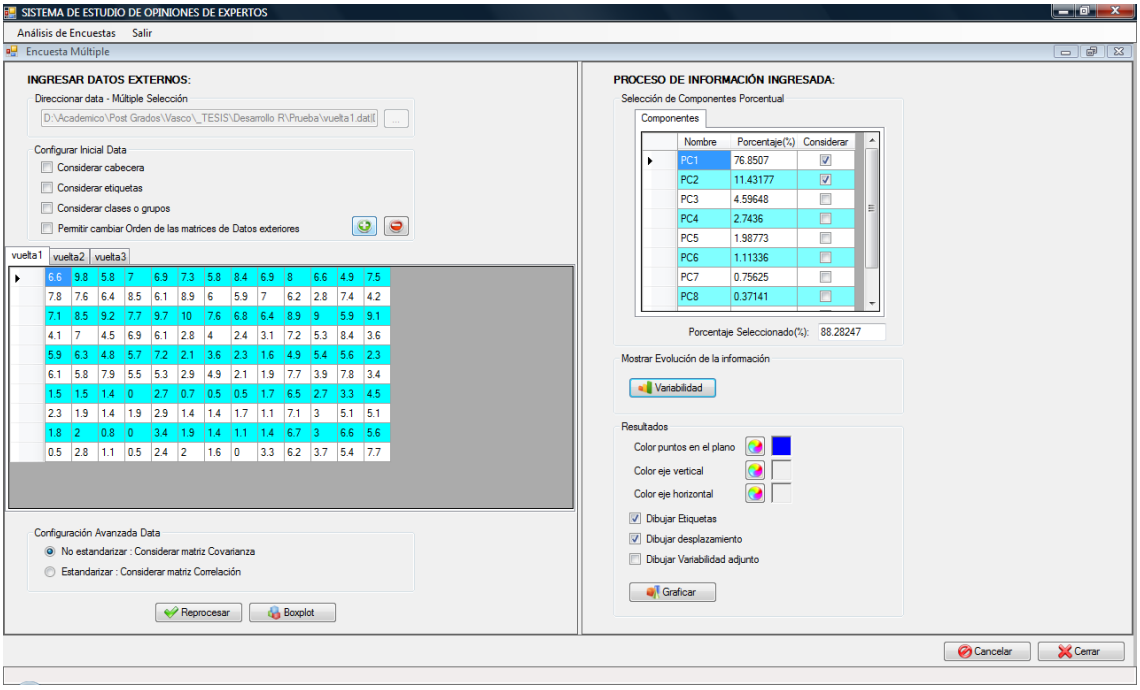

Figura 5.3 - Pantalla encuesta múltiple

5.2 Manual de usuario y ejemplo ilustrativo de la herramienta

Para poder entender de mejor manera el funcionamiento de la herramienta interactiva, se ha desarrollado un manual de usuario, el cual ilustra un ejemplo basado en el método Delphi del tema de "Innovación y desarrollo biotecnológico para Latino América - Uruguay proyectándose al año 2015".

Para el ejemplo ilustrativo, se tomará la primera encuesta de tipo Likert (Con PMIN = 0 y PMAX = 10) con un total de 13 preguntas, de un conjunto de encuestas del tema mencionado [Anexo III]. En esta primera encuesta, intenta provocar la reflexión del panel de expertos respecto a las tendencias más relevantes del futuro de la biotecnología en Uruguay para el año 2015.

Cabe mencionar, que se han simulado las respuestas dadas por los expertos y se ha considerado tres interacciones como número de vueltas realizadas sobre la encuesta. También se han considerado como número de participantes, la cantidad de 10 expertos.

A continuación se desarrolla el manual de usuario sobre el ejemplo ilustrativo:

1. Pre-requisitos: para el normal funcionamiento del sistema, son necesarios los siguientes pre-requisitos:

- Sistema operativo Windows XP sp2 o superior.
- Framework 2.0 o superior.
- Herramienta estadística R versión 2.13.1 con las librerías:
    - rscproxy_1.3-1
    - RGtk2_2.20.17
    - rggobi_2.1.17.
- Herramienta estadística GGplot.

2. Ingresar al sistema: una vez instalada la herramienta en su sistema operativo, se ingresa al sistema donde se muestra la pantalla principal con un fondo de color plomo. En la parte superior izquierda aparece el menú principal para poder realizar los diferentes análisis que permite la herramienta.

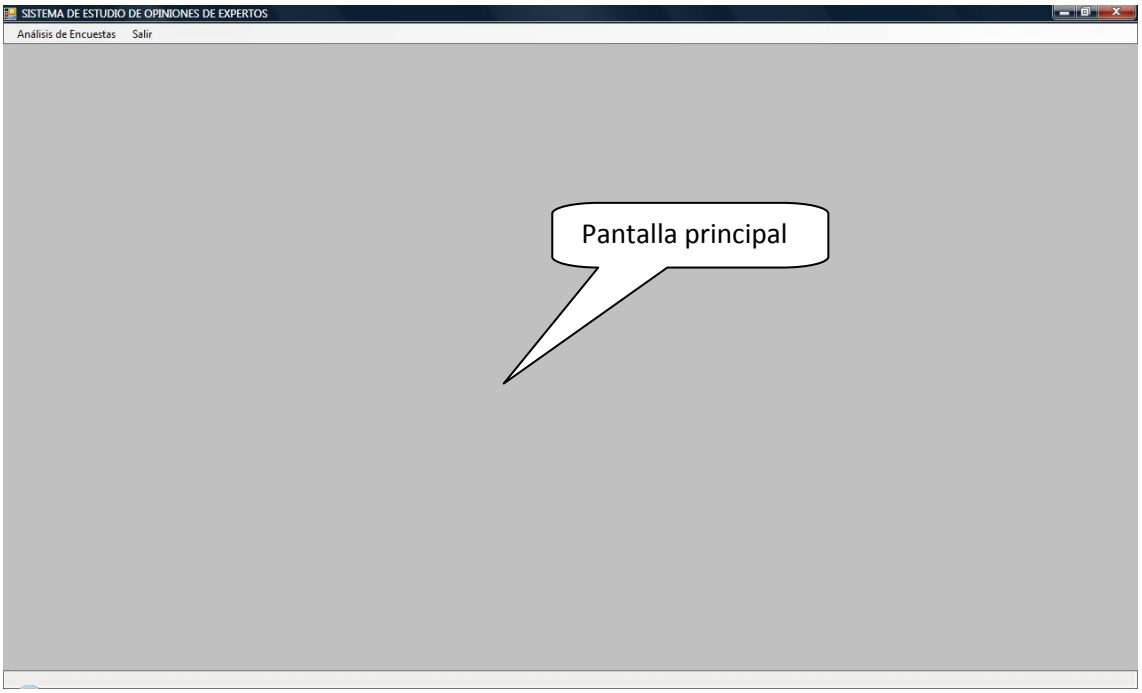

Figura 5.4 - Pantalla principal de la herramienta

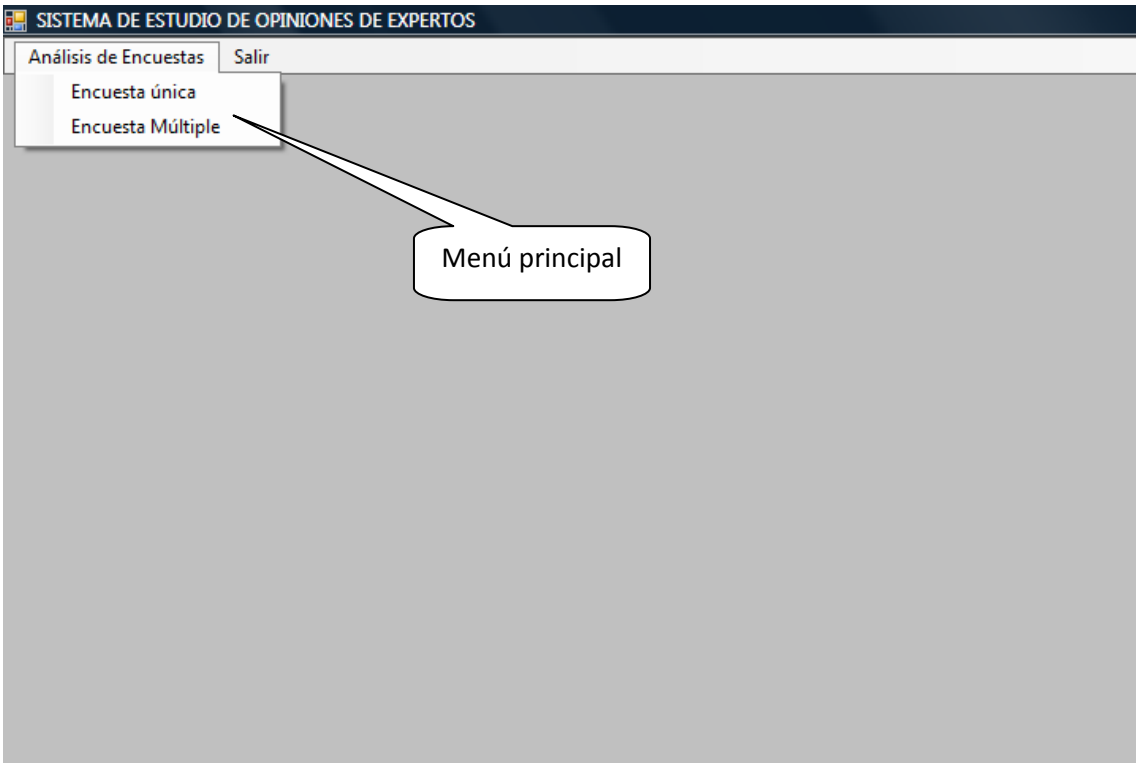

Figura 5.5 - Menú principal del sistema

3. Análisis de encuesta única: Gracias al menú principal podemos ingresar a la interfaz de "análisis de encuesta única" donde, como su nombre indica, podemos analizar una única encuesta de la ronda realizada. Nos presenta la siguiente interfaz:

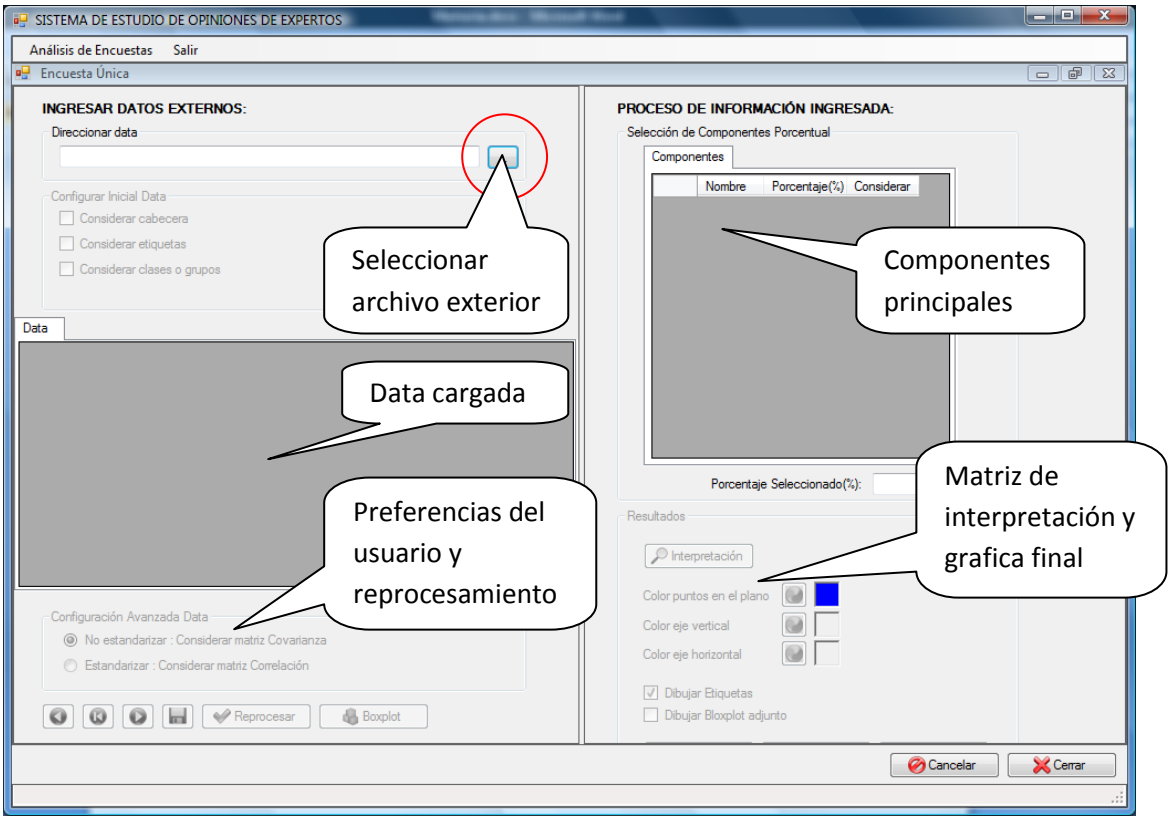

Figura 5.6 - Interfaz de análisis de encuesta única

Al seleccionar un archivo exterior (encuesta), nos muestra un navegador de carpetas para poder ubicar a nuestro archivo que deseamos cargar como data. Estos archivos pueden soportar las extensiones de *.dat, y *.txt.

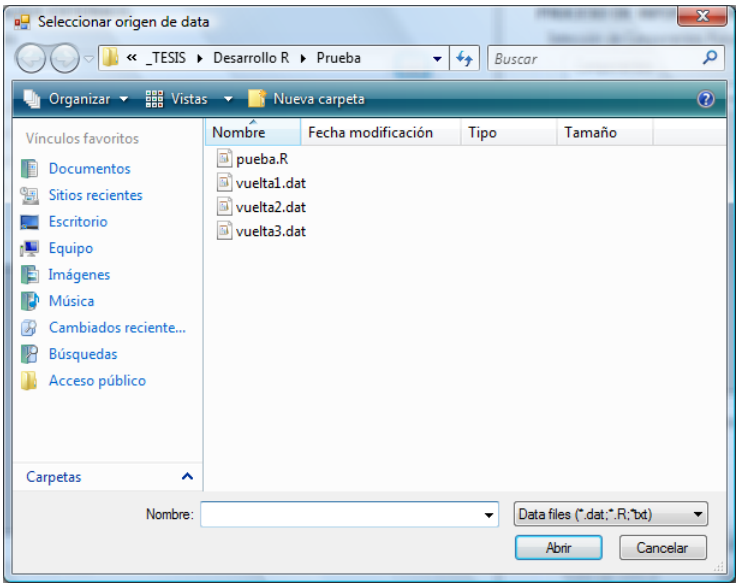

Figura 5.7 - Selección de un archivo exterior

Para nuestro ejemplo ilustrativo, seleccionaremos el primer archivo llamada "Vuelta 1" que tiene los siguientes datos:

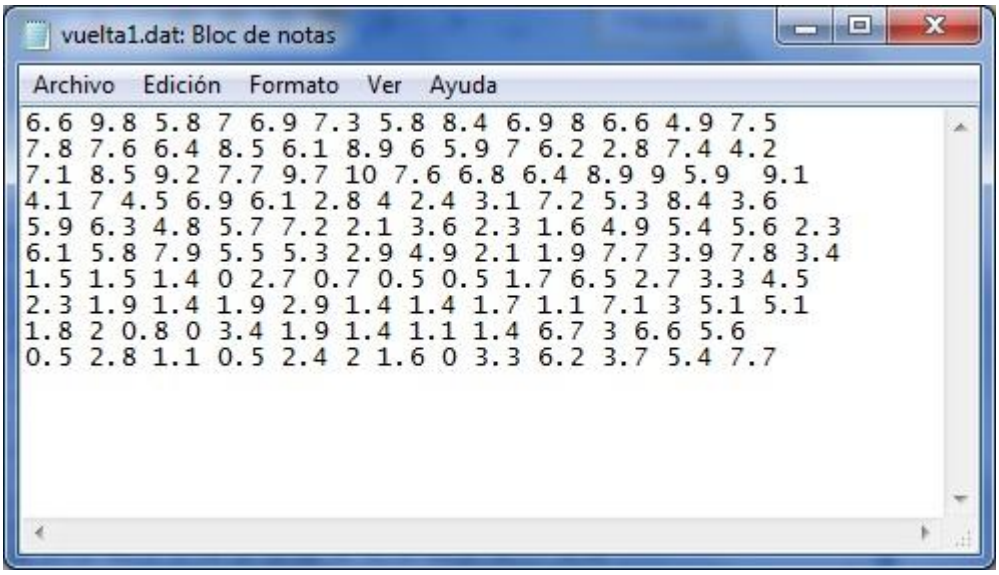

Figura 5.8 – Encuesta "Vuelta 1"

Una vez elegido un archivo exterior, el sistema se encarga de verificar si la información que ingresamos es correcta, es decir, si se trata de una matriz de 10x13 (10 expertos por 13 respuestas), y la mostrará en la interfaz en líneas de color intercalado blanco y celeste, para una mejor visualización; el usuario puede considerar cabecera, etiquetas y clases (el color de las celdas cambia a color plomo). A la información cargada le llamaremos "data" y esta puede ser modificada por el usuario si así lo desea.

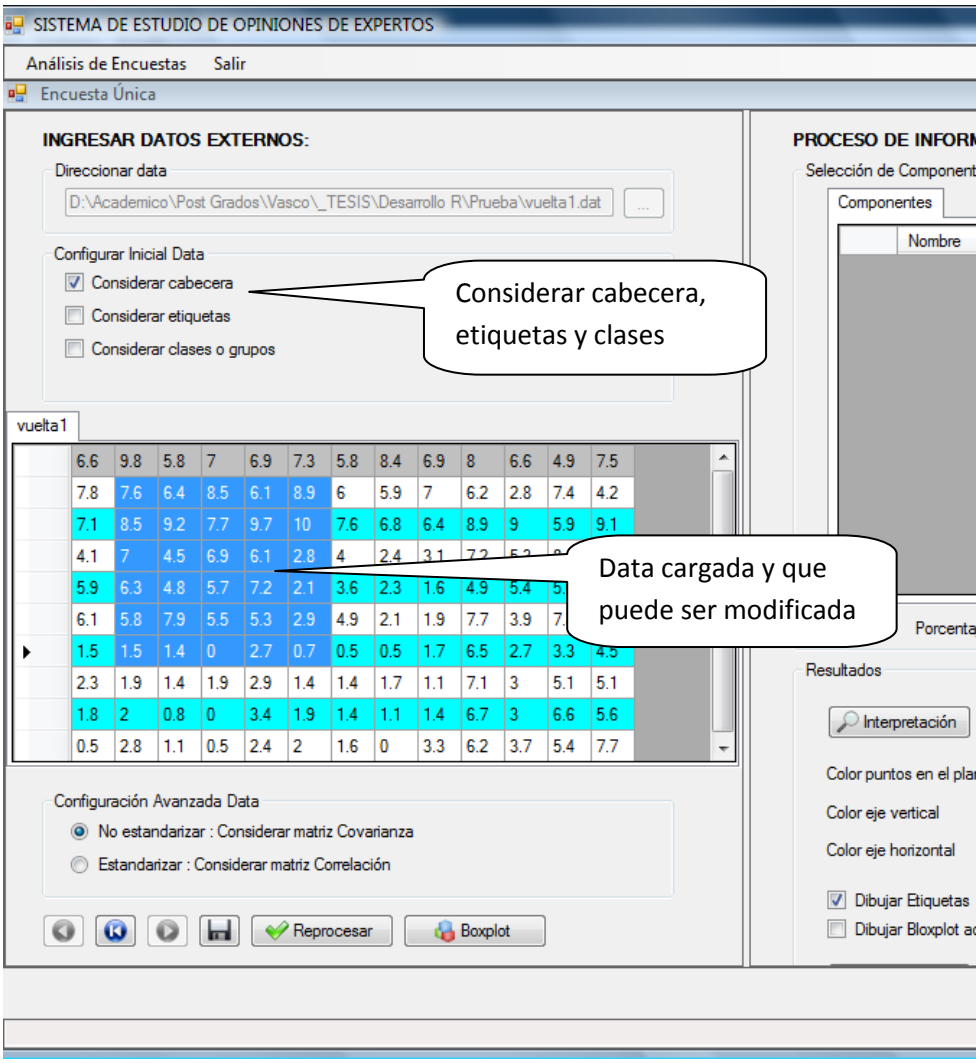

Figura 5.9 - Data cargada

En la parte inferior de la data podemos visualizar las preferencias del usuario para procesar la información (Configuración Avanzada Data), aquí el usuario elige si desea procesar la información estandarizada o tal como fue ingresada al sistema; para tomar esta decisión el usuario puede visualizar el boxplot. De igual manera puede visualizar botones de interacción de la data para poder volver a estados anteriores y posteriores después de haber modificado la misma; también se observa el botón para guardar la data después de haber hecho modificaciones y el botón de reprocesamiento (volver a obtener valores para las gráficas finales).

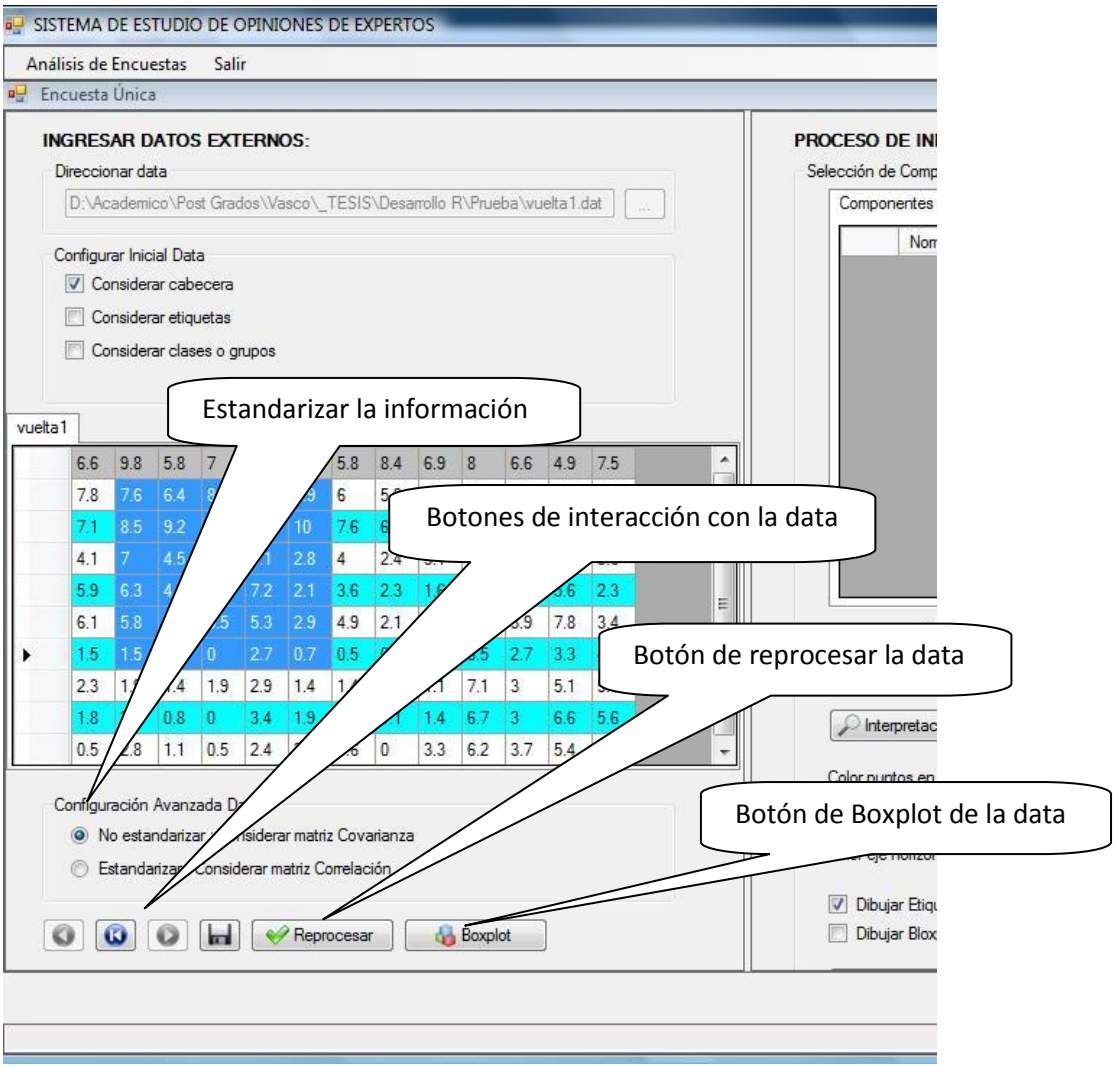

Figura 5.10 - Estandarización de la información, botones de interacción, reprocesar y boxplot

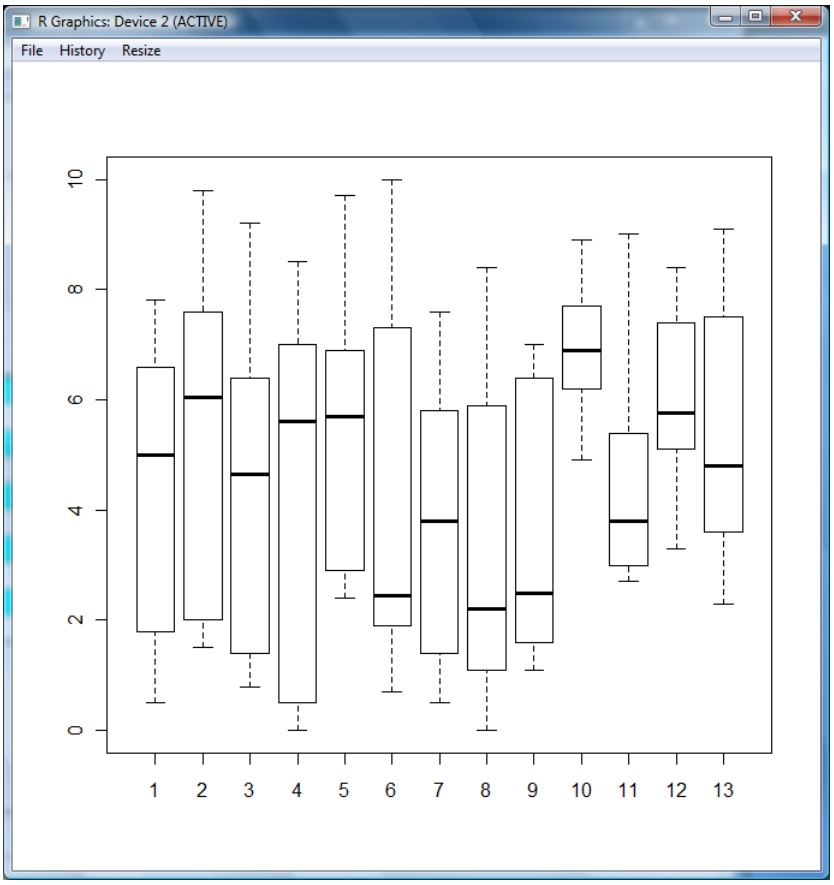

Figura 5.11 - Bloxplot

Después de haber procesado la información, el sistema se encarga de calcular y mostrar los componentes principales de la data, brindándonos la opción de poder seleccionar que componentes utilizaremos. Por defecto vienen seleccionados los dos mayores componentes, que sumados llegan a representar gran parte de la información de la data.

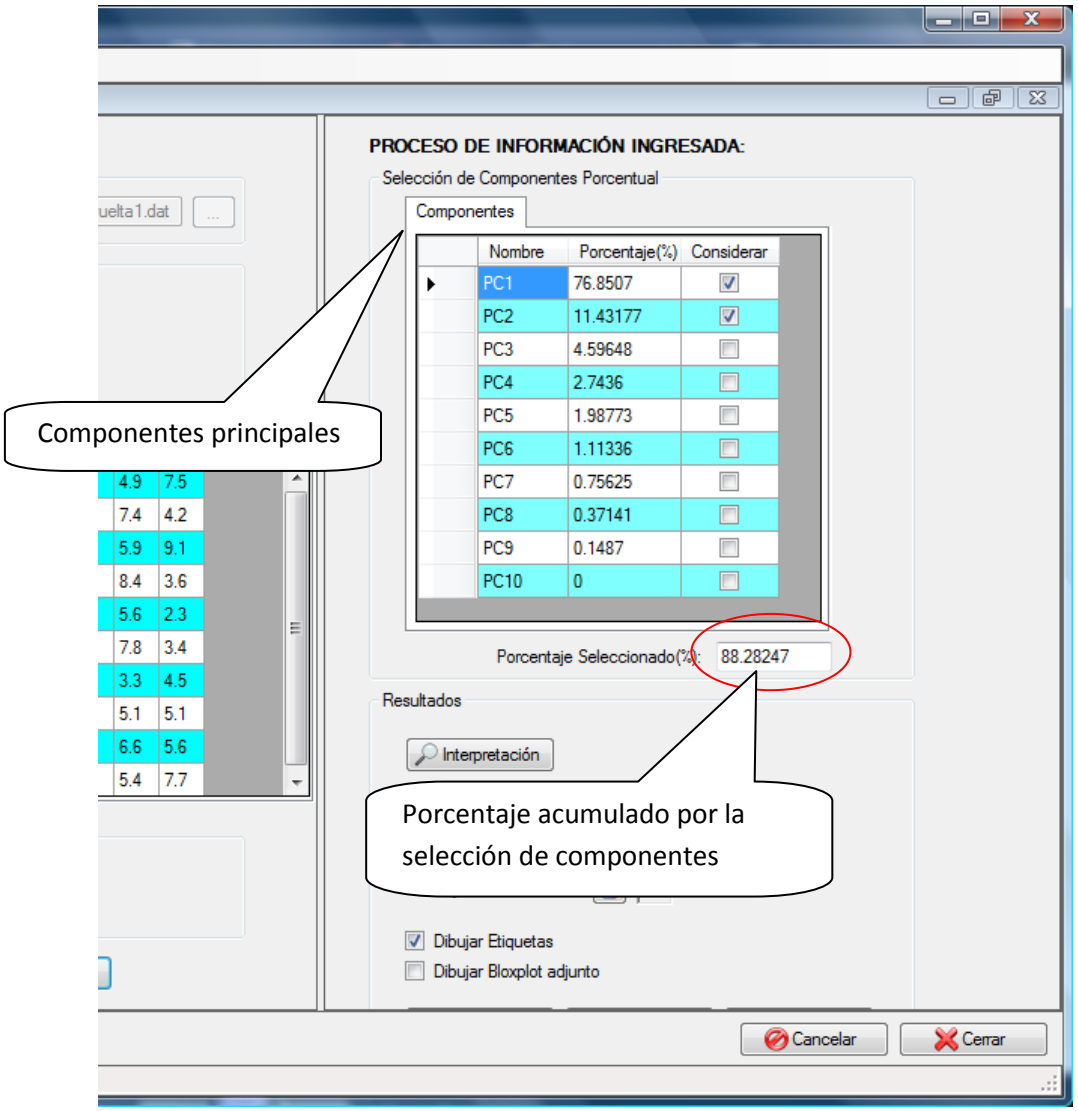

Figura 5.12 - Componentes principales

En la parte inferior de los componentes principales tenemos el apartado de "resultados" donde podemos recuperar la matriz de interpretación, así como también el gráfico final de consenso; la herramienta permite seleccionar el color de los puntos y de los ejes que queremos mostrar en el gráfico.

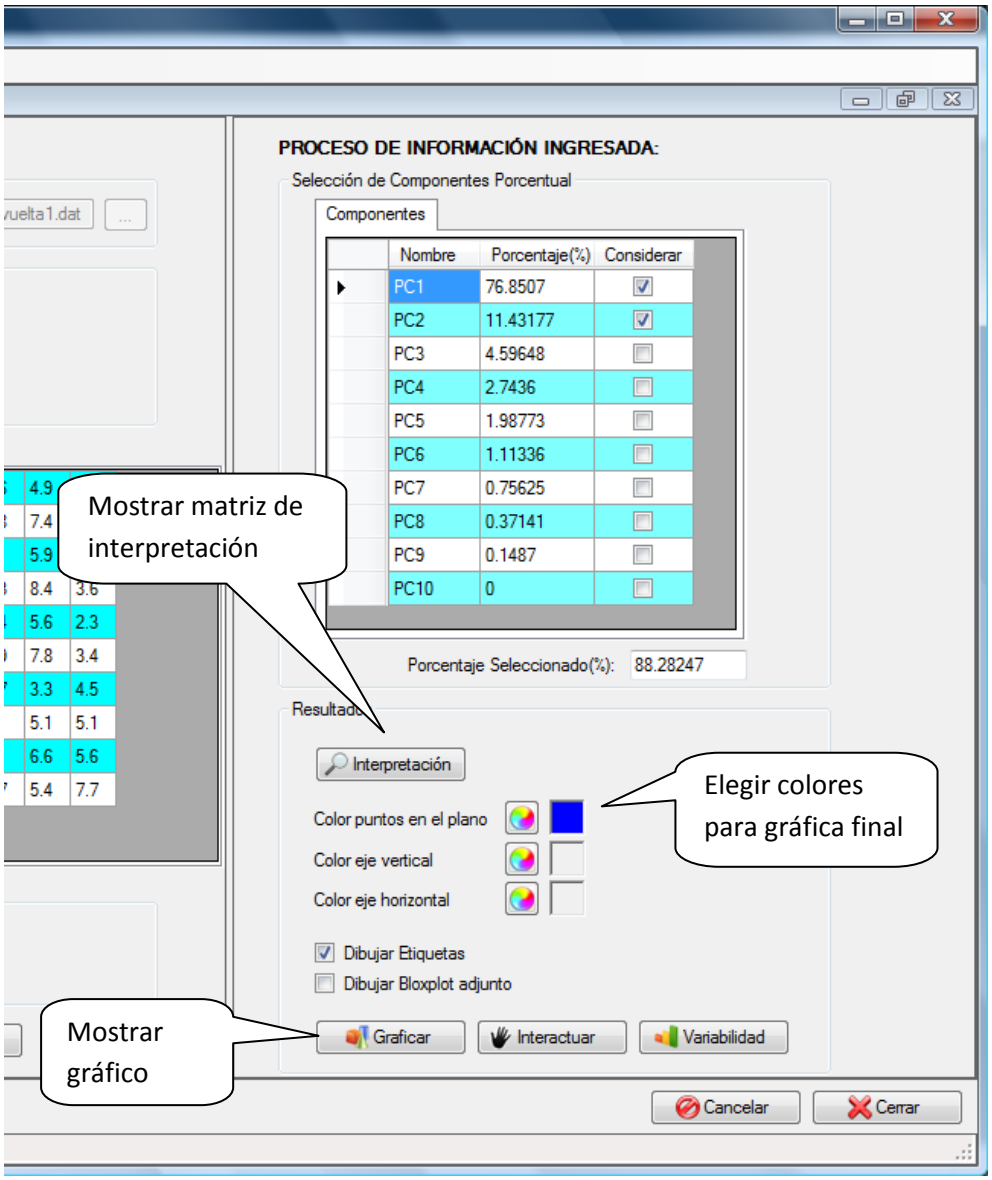

Figura 5.13 - Apartado de resultados

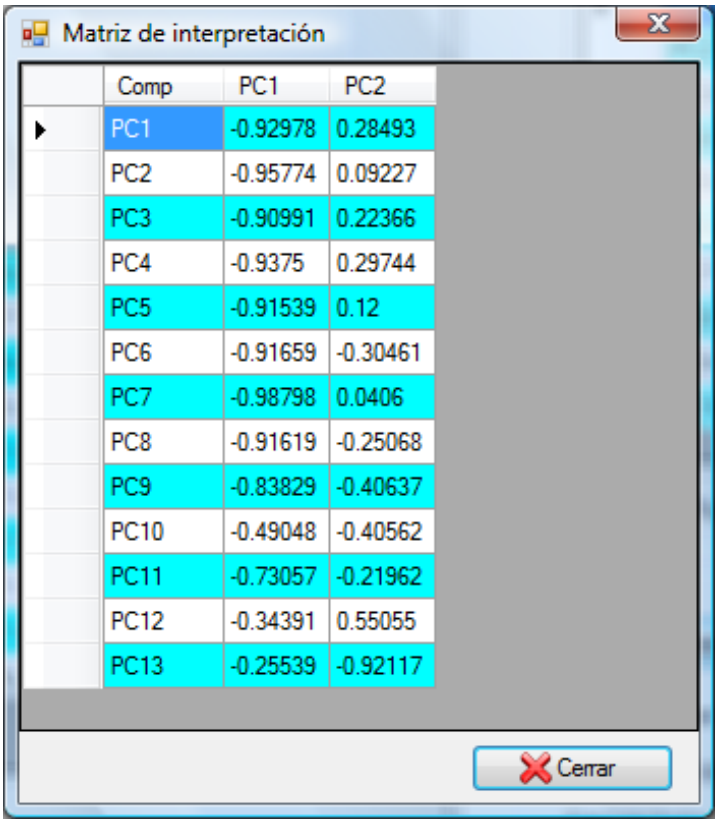

Figura 5.14 - Matriz de interpretación

Podemos ver gracias a la salida de consenso, que existen tres grupos de opiniones de los expertos, es decir, que existen tres tipos de opiniones sobre el tema entre los 10 participantes.

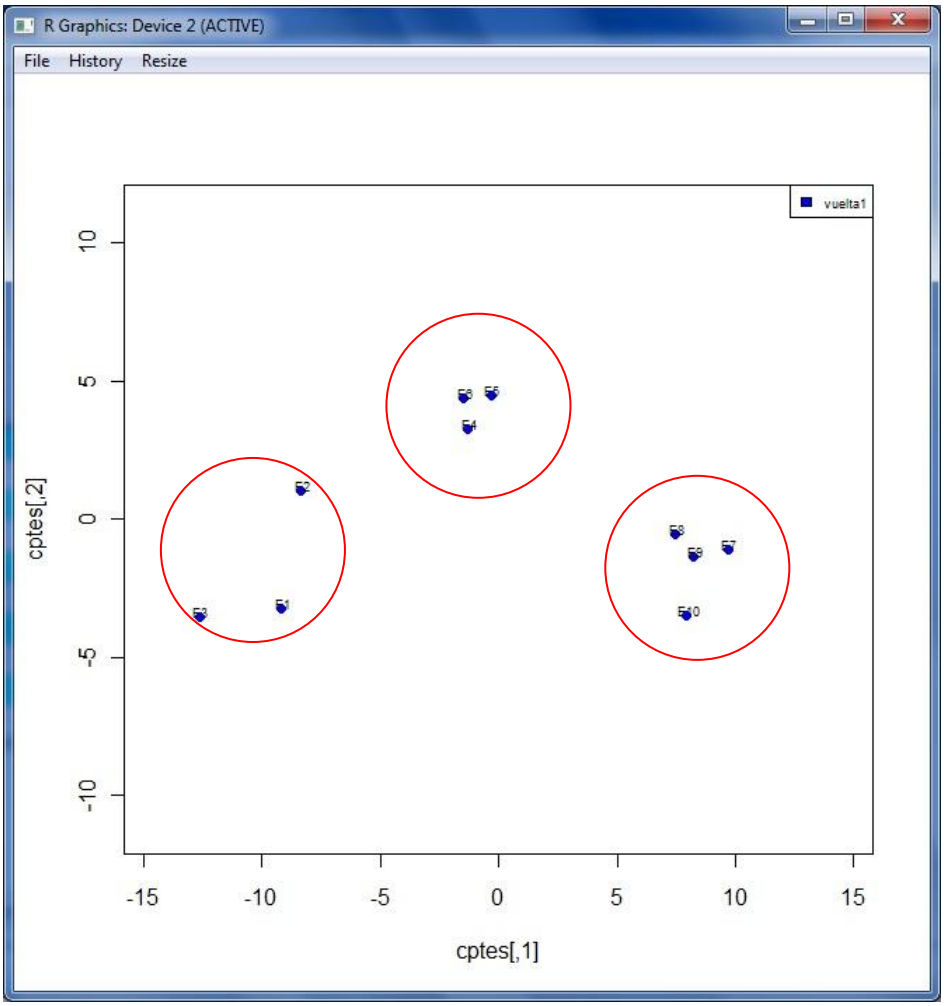

Figura 5.15 - Salida gráfica de consenso de la vuelta 1

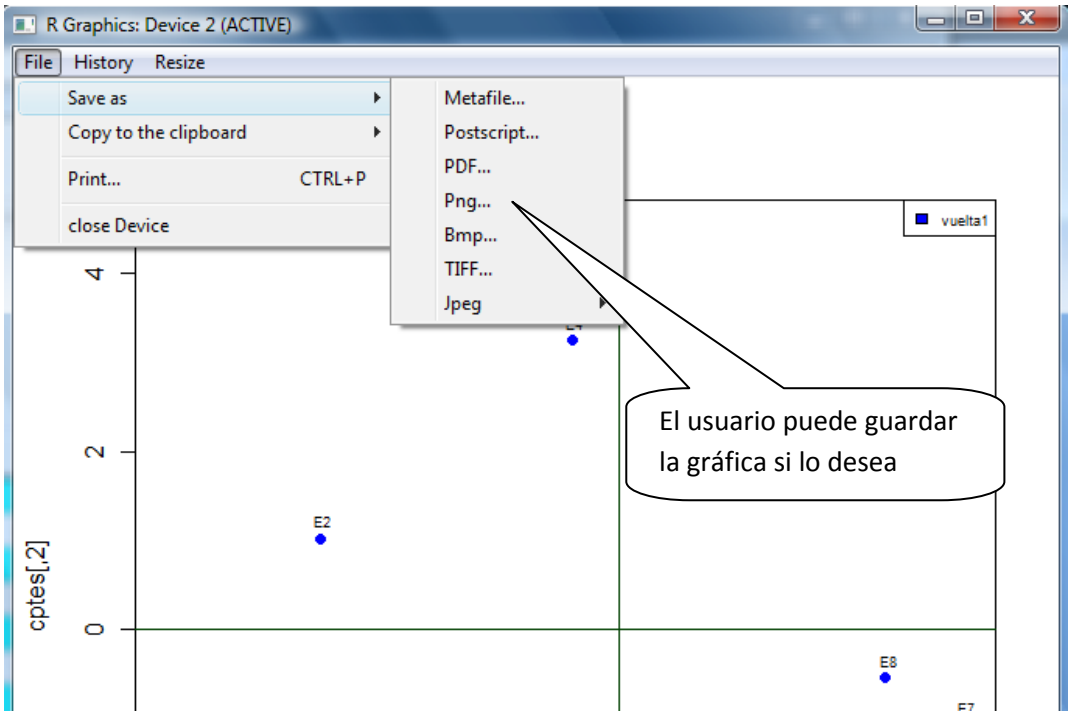

Figura 5.16 – Opción de guardar la salida gráfica de consenso

En caso el usuario desea interactuar con el resultado mostrado, presiona el botón de “interactuar” donde le mostrará una interfaz de color plomo y negro, y podrá interactuar con la salida gráfica, simulando, si así lo desea, un mayor grado de consenso entre los usuarios y paralelamente el sistema recuperará las respuestas simuladas, de esta manera el usuario simula una situación hipotética sobre la encuesta realizada. Otra opción que ofrece la herramienta es poder visualizar como ha variado el consenso gracias a la “variabilidad”.

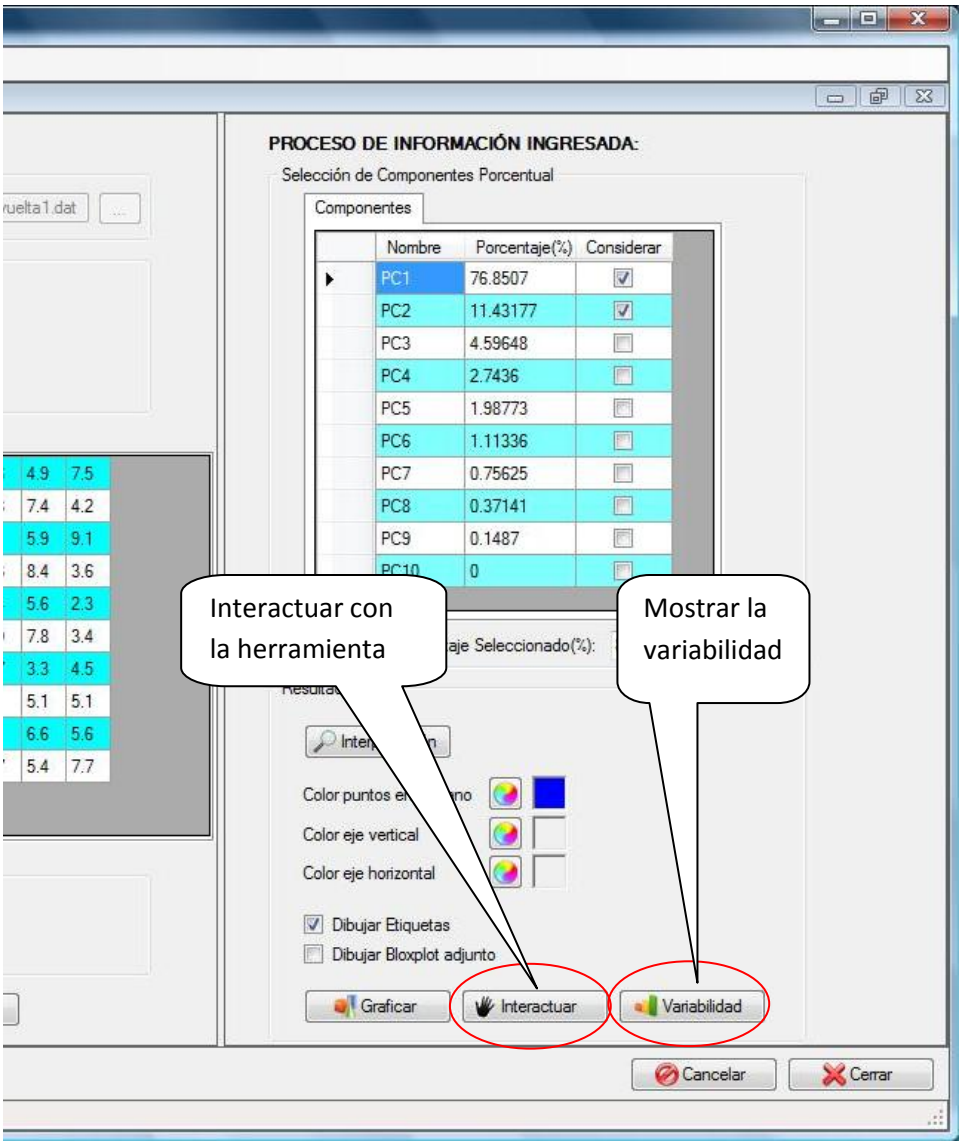

Figura 5.17 - Iteración con la herramienta y variabilidad

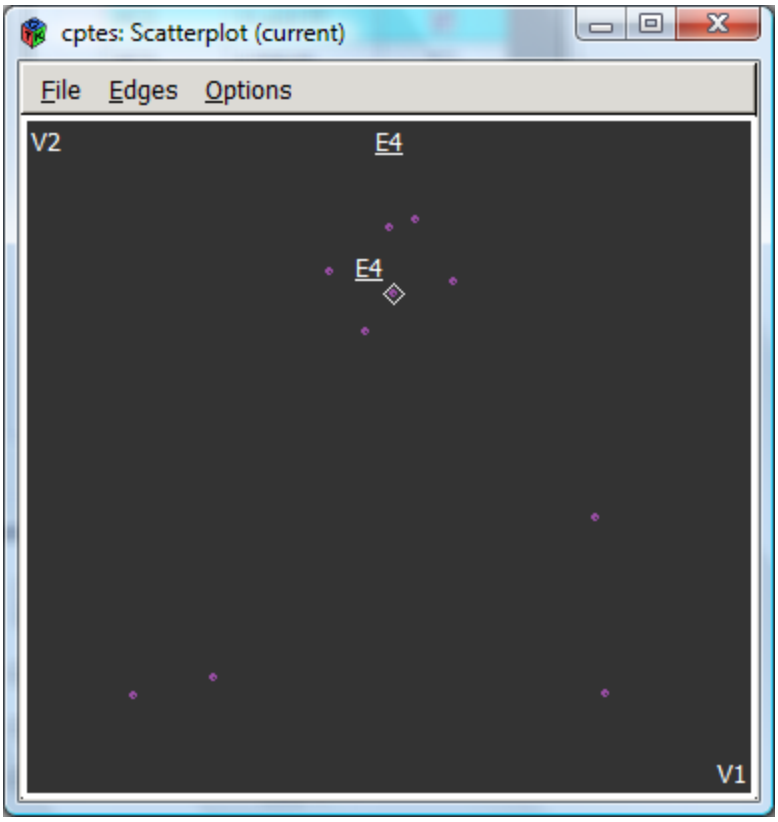

Figura 5.18 - Interacción con la herramienta

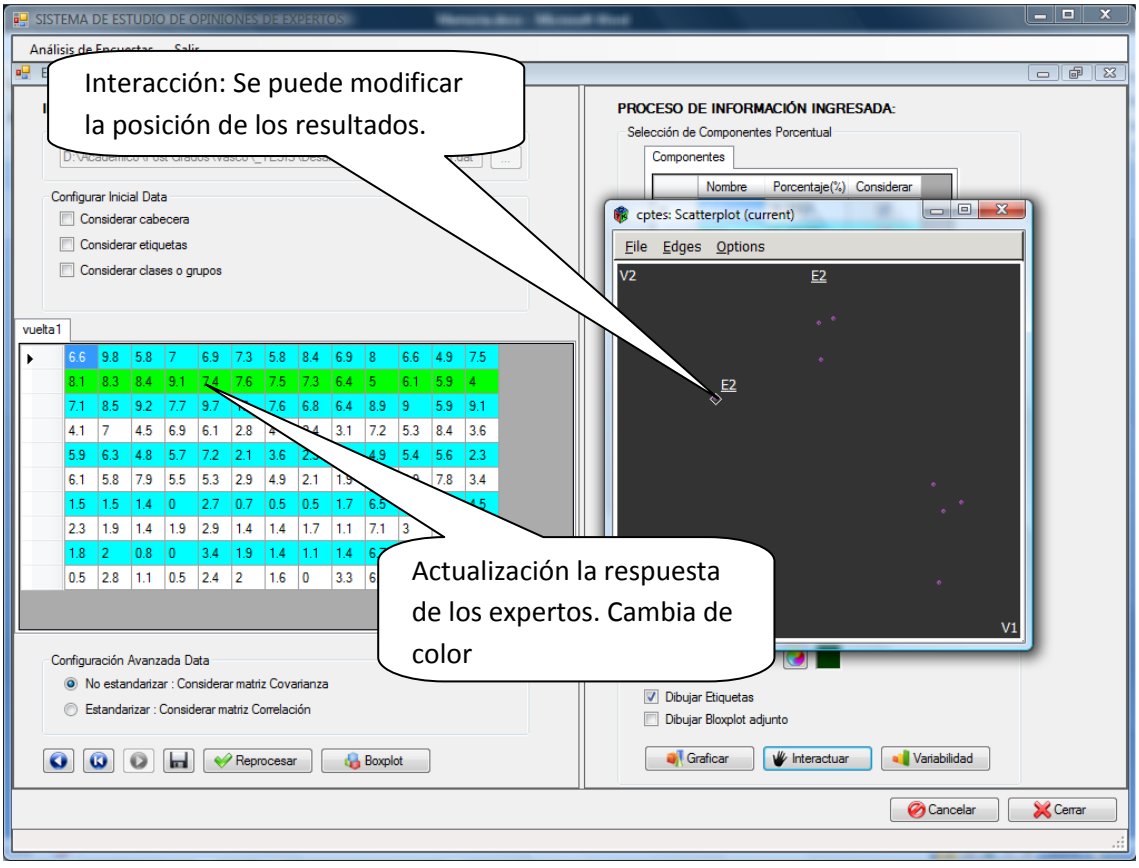

Figura 5.19 - Interacción y actualización de la data

Gracias a la variabilidad, podemos observar que el grado de consenso que ha simulado el usuario sobre los participantes ha tendido a cero, significa que existe un mayor acuerdo entre los expertos.

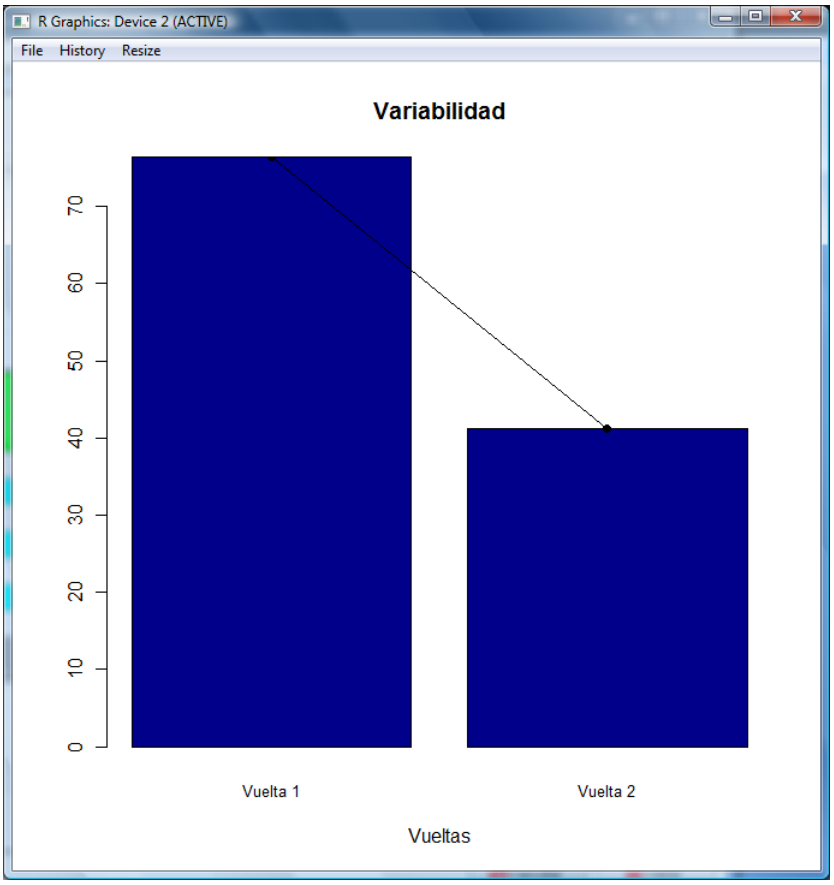

Figura 5.20 - Variabilidad después de interactuar con la herramienta

4. Análisis de encuesta múltiple: Regresando al menú principal, esta vez elegimos la opción de análisis de encuesta múltiple donde podremos realizar casi las mismas operaciones que en la encuesta única, con la diferencia de que puedo procesar más de una encuesta a la vez y visualizar una salida gráfica diferente donde me muestra la evolución de las opiniones de las diferentes vueltas que se han ingresado.

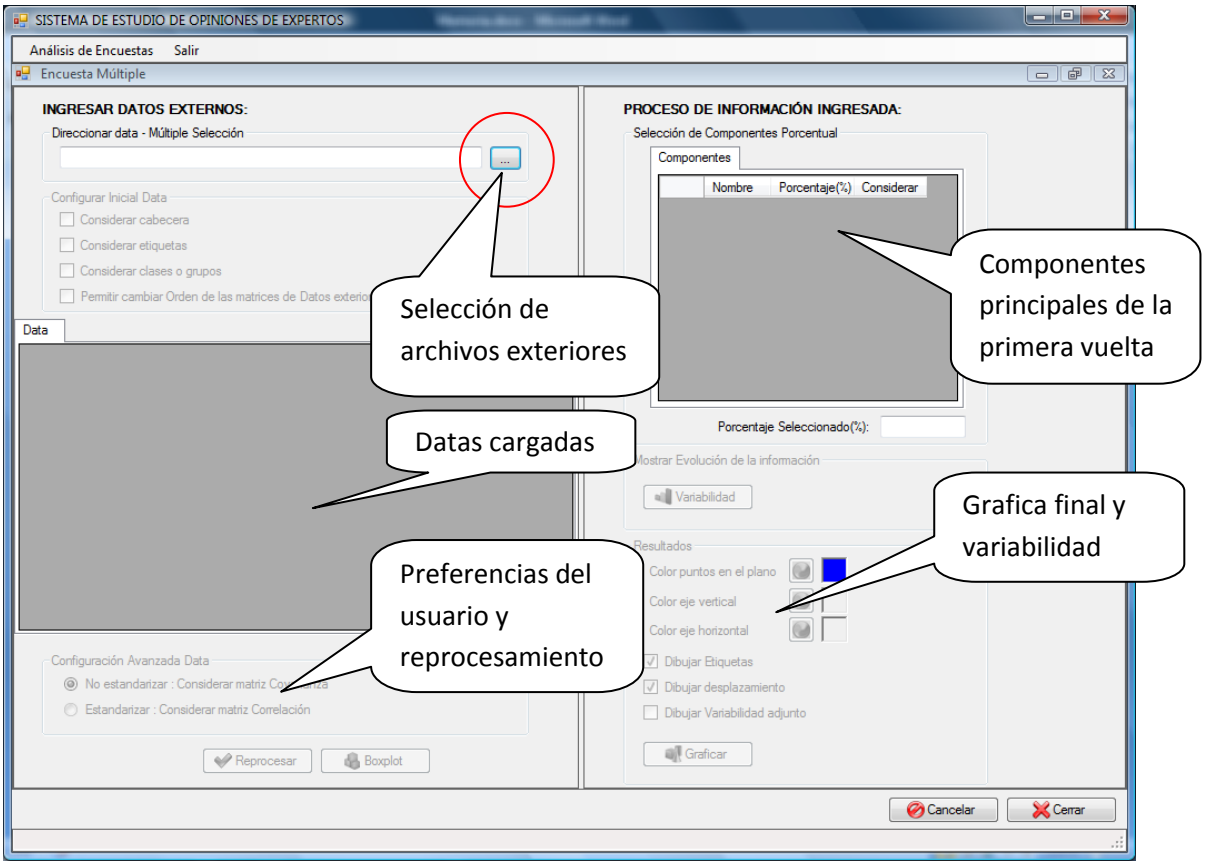

Figura 5.21 - Interfaz de análisis de encuestas múltiples

En esta interfaz, la herramienta me permite seleccionar más de un archivo a la vez y poder visualizarlo en distintas pestañas que se crean de manera dinámica en la parte de "data"; en la parte superior de la data se puede considerar cabecera, etiquetas y clases con la diferencia que también me permite poder cambiar de posición las datas ingresada, agregar y quitar las encuestas.

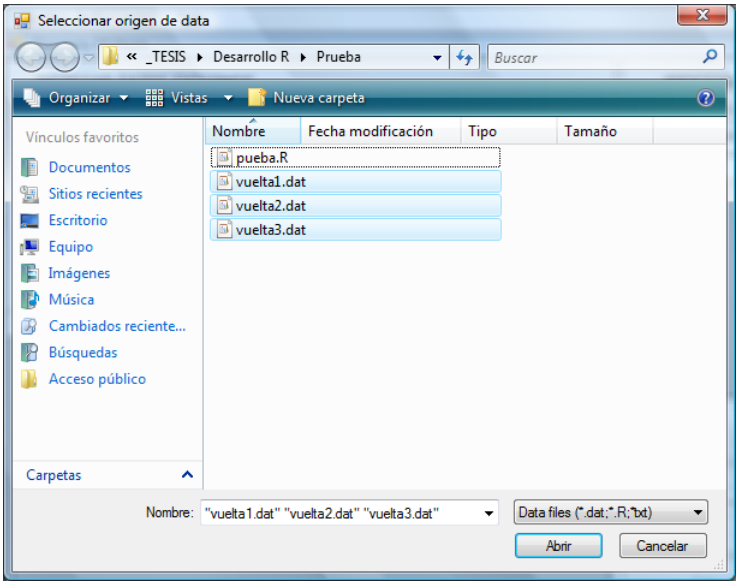

Figura 5.22 - Selección múltiple de uno o más archivos externos

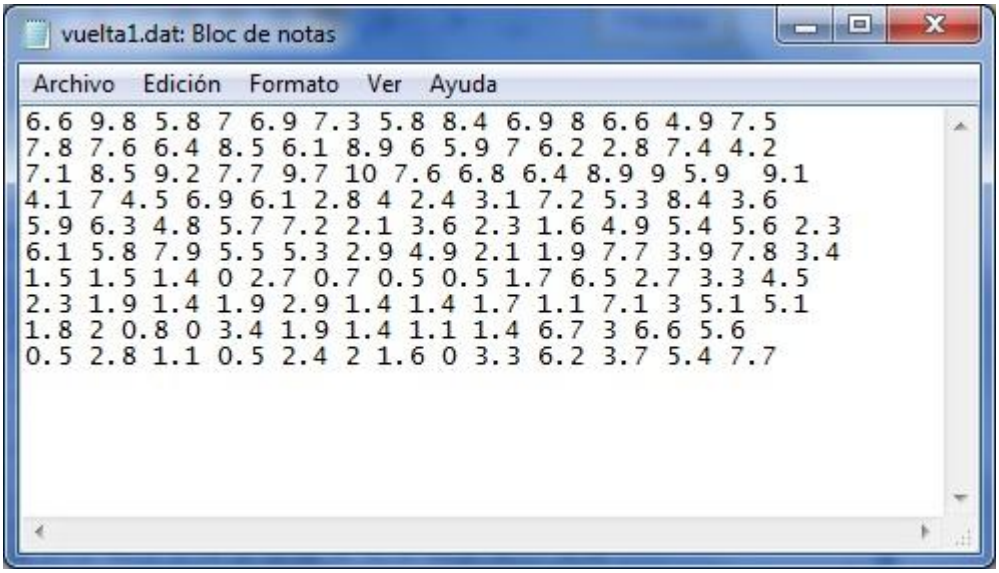

Figura 5.23 – Encuesta "Vuelta 1"

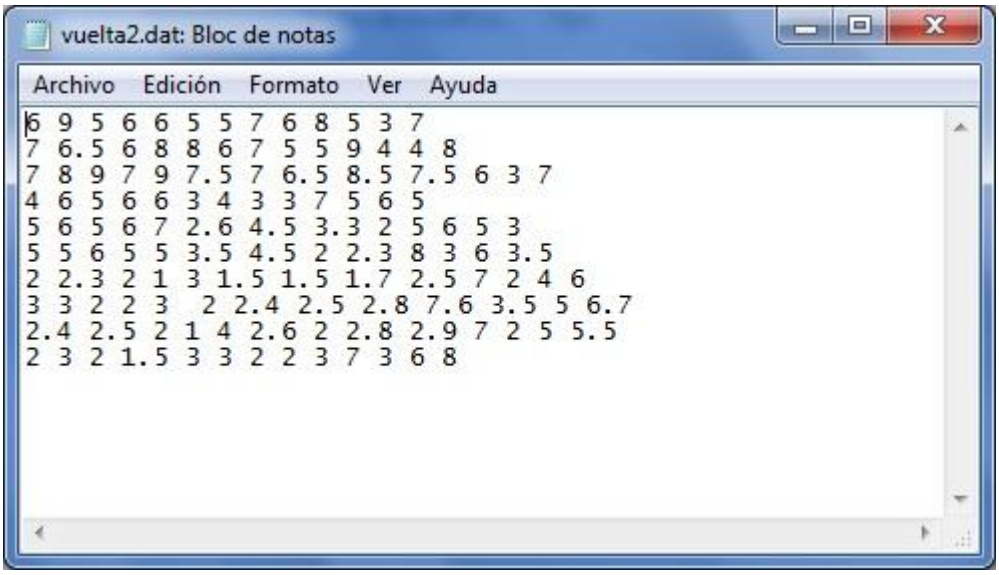

Figura 5.24 – Encuesta "Vuelta 2"

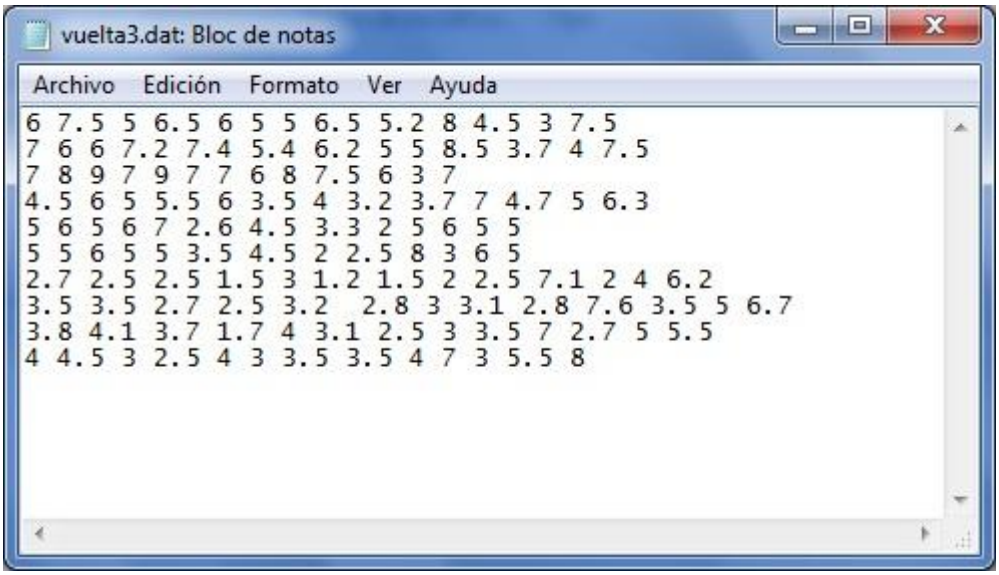

Figura 5.25 – Encuesta "Vuelta 3"

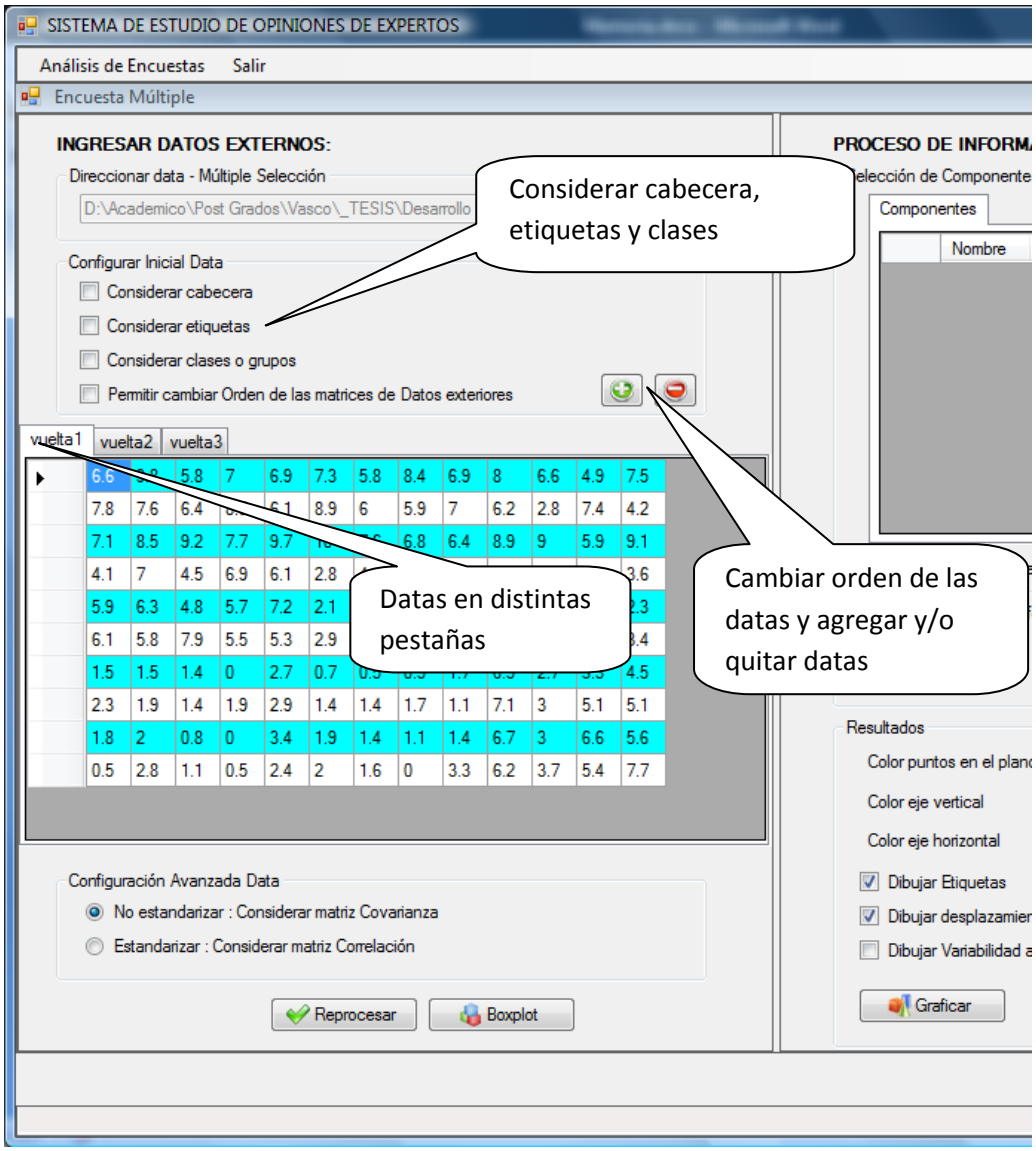

Figura 5.26 - Interfaz con tres datas cargadas y botones para agregar y quitar datas

Al igual que en el anterior caso, se puede mostrar un boxplot para que el usuario elija si desea estandarizar la información o no y procesar dicha información. Al lado derecho se mostrará los componentes principales que vienen por defecto seleccionados los dos mayores.

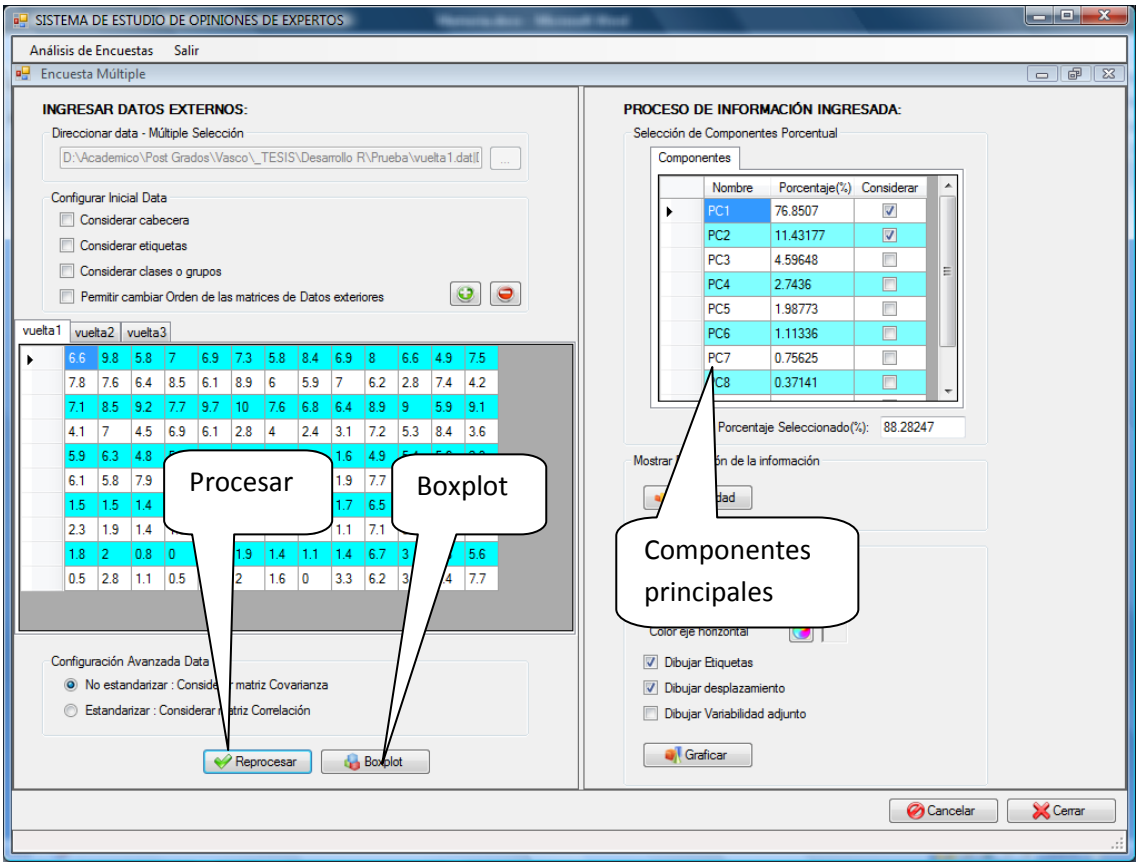

Figura 5.27 - Procesar, boxplot y componentes principales

En el apartado de resultados, podemos visualizar el botón de "variabilidad", donde podemos visualizar cómo ha ido evolucionando las opiniones y el gráfico final, donde podemos seleccionar, si queremos ver más detallado y elegir los colores. Se puede guardar las salidas gráficas para que luego pueda ser comparada.

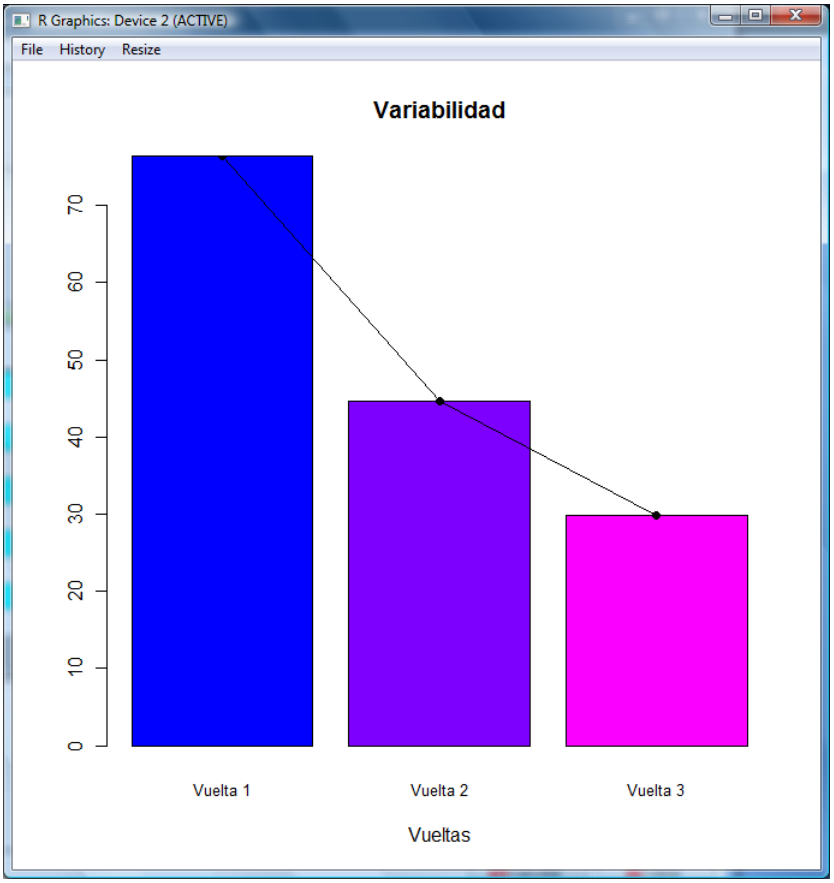

Figura 5.28 - Variabilidad de las tres vueltas en el análisis de encuestas múltiple

Gracias a la salida grafica de consensos, podemos observar que el grado de consenso se va centrado, lo que indica que los expertos participantes, cada vez tienden a tener la misma opinión en la encuesta.

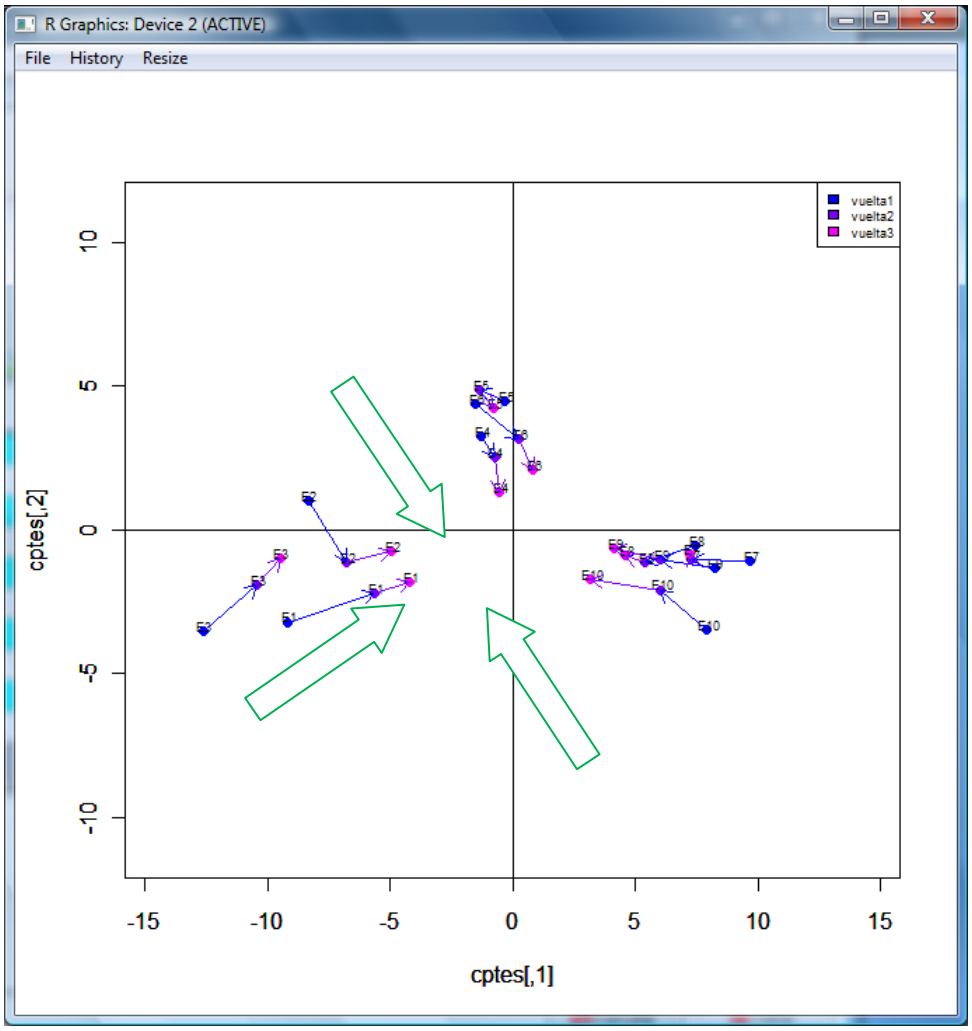

Figura 5.29 - Salida gráfica con flechas que indican cómo se han desplazado las opiniones de los expertos

Conclusiones y futuras líneas de investigación

Se ha llegado a las siguientes conclusiones en el presente proyecto:

1. La minería de datos, desde muchos puntos de vista, es una gran herramienta para el análisis y obtención de información siendo de utilidad para muchos campos. En particular la reducción de variables fue de gran utilidad para el desarrollo del proyecto.
2. La herramienta desarrollada en el presente proyecto de tesis, es útil para resumir y visualizar las posiciones de los expertos. Gracias a la herramienta, por una parte, se puede visualizar las opiniones de los expertos en una vuelta particular, mediante la cual se puede identificar dónde están los acuerdos y desacuerdos entre los expertos e interactuar con ella; y por otra parte, permite visualizar la evolución de las opiniones de cada experto a lo largo de las sucesivas vueltas midiendo el grado de acuerdo o consenso de manera multivariante.

Basado en el presente proyecto de investigación, se propone las siguientes futuras líneas de investigación:

1. Implementación de una herramienta interactiva para el estudio de la opinión de expertos recogida en una encuesta con datos cuantitativos, cualitativos y mixtos.
2. Implementación de una herramienta interactiva para el estudio de la opinión de expertos recogida en una encuesta con diferentes herramientas de programación y herramientas estadísticas buscando mejorar la calidad de interacción entre el sistema y usuario final, mejores presentaciones gráficas y creación de multimedia, viendo cómo se puede mejorar la herramienta. Para detectar las mejoras con mayor eficiencia, se considera muy interesante utilizar la herramienta en un caso real.
3. Implementación de una herramienta que considere todas las fases de método Delphi, utilizando plataformas de tecnologías Web.

Bibliografía

- **COX, T.F and COX, M.A.A.** (1994). Multidimensional Scaling. Chapman & Hall.
- **EVERITT, B.** (2005). An R and S-PLUS companion to multivariate analysis. Springer.
- **EVERITT, B. and DUNN, G.** (2001). Applied Multivariate Data Analysis. Hodder Arnold.
- **HAIR** (1999). Análisis Multi variante. Prentice-Hall.
- **HERNANDEZ, J. RAMÍRES, M.J. y FERRI, C.** (2005). Introducción a la minería de datos. Pearson Prentice-Hall.
- **MURREL, P.** (2005). R Graphics. Chapman and Hall.
- **PEÑA, D.** (2002.) Análisis de Datos Multi variantes. McGraw-Hill.
- **FACULTAD DE CC.EE. Y EMPRESARIALES. ESTE**, EL MÉTODO DELPHI, Universidad de Deusto.
- **JOSÉ LUIS VICENTE VILLARDÓN,** Análisis de componentes principales, Departamento de Estadística.
- **WALTER G. MELÉNDEZ BERNARDO,** Métodos de Prospección: El método Delhi y su aplicación en la Ingeniería Civil, Universidad Nacional de Ingeniería – Perú 2009.

Referencias Web

- http://www.calibrum.com/tf_delphi.htm
- http://www.calibrum.com/Products/Delphi/DelphiOnline.asp
- http://www.gtic.ssr.upm.es/encuestas/delphi.htm
- http://waltermelendez.blogspot.com/2009/05/metodos-de-prospeccion-el-metodo-delphi.html
- http://www.onudi.org.uy/downloads/ONUDI-MIEMBROS/Paises/Uruguay/Prospectiva_Tecnologica/BIOTECNOLOGIA/BIOTECN_Anexo_3_Cuestionario_Delphi.pdf
- http://www.codeproject.com/KB/cs/RtoCSharp.aspx
- http://r.789695.n4.nabble.com/C-and-R-td860173.html
- http://www.wilmott.com/messageview.cfm?catid=10&threadid=78402
- http://ebookee.org/Multidimensional-Scaling-Second-Edition_336906.html
- http://www.calibrum.com/Products/Delphi/DelphiOnline.asp
- http://www.kdnuggets.com

Anexos

Anexo I: Clase de conexión

```
public class CConexionR
    {
        #region Atributos
        public StatConnector rconn;

        #endregion Atributos

        #region constructores
        public CConexionR()
        {
            rconn = new STATCONNECTORSRVLib.StatConnector();
        }

        #endregion constructores

        #region metodos
        //Método para inicializar la conexión
        public bool start()
        {
            bool salida;
            try
            {
                rconn.Init("R");
                salida = true;
            }
            catch(Exception e)
            {
                salida = false;
            }

            return salida;
        }

        //Método para finalizar la conexión
        public bool finish()
        {
            bool salida;
            try
            {
```

```
                rconn.Close();
                salida = true;
            }
            catch (Exception e)
            {
                salida = false;
            }

            return salida;
        }

        //Método para asignar variable
        protected void asignar(string nombreVariable, object
valorVariable)
        {
            rconn.SetSymbol(nombreVariable, valorVariable);

        }

        //Método para ejecutar línea de comando
        protected void ejecutar(string lineaComando)
        {
            rconn.Evaluate(lineaComando);
        }

        //Método para recuperar variable
        protected object recuperar(string nombreVariable)
        {
            object salida;
            salida = rconn.GetSymbol(nombreVariable);

            return salida;
        }

        //Método para recuperar variable double
        protected double recuperar_Double(string nombreVariable)
        {
            double salida;
            salida = (double)rconn.GetSymbol(nombreVariable);

            return salida;
        }
```

```
        //Método para recuperar variable matriz de n filas
        protected object[] recuperarMatrizN(string nombreVariableMatriz)
        {
            object[] salida;
            salida = (object[])rconn.GetSymbol(nombreVariableMatriz);

            return salida;
        }

        //Método para recuperar variable matriz de n filas
        protected double[] recuperarMatrizN_Double(string
nombreVariableMatriz)
        {
            double[] salida;
            salida = (double[])rconn.GetSymbol(nombreVariableMatriz);

            return salida;
        }

        //Método para recuperar variable matriz de n filas x n columnas
        protected object[,] recuperarMatrizNxN(string
nombreVariableMatriz)
        {
            object[,] salida;
            salida = (object[,])rconn.GetSymbol(nombreVariableMatriz);

            return salida;
        }

        //Método para recuperar variable matriz de n filas x n columnas de
tipo double
        protected double[,] recuperarMatrizNxN_Double(string
nombreVariableMatriz)
        {
            double[,] salida;
            salida = (double[,])rconn.GetSymbol(nombreVariableMatriz);

            return salida;
        }

        //Método para graficar
```

```
public void graficar(string lineaComando)
{
    rconn.EvaluateNoReturn(lineaComando);
}

#endregion metodos
```

Anexo II: Clase de funciones

```
//Se redirecciona a la carpeta donde se desea trabajar
        public void direccionar(string direccion)
        {
            direccion = direccion.Replace("\\", "\\\\");
            string comando = "setwd(\"" + direccion + "\")";
            ejecutar(comando);
        }

        //Se recupera la matriz de datos desde archivo exterior
        public void asignarMatrizDatos(string nombreArchivo, bool
cabecera, int nro_columnas_matriz)
        {
            //Se reescribe el nombre del archivo
            nombreArchivo = nombreArchivo.Replace("\\", "\\\\");
            //Se asigna la matriz de datos
            string comando = "";
            if (cabecera)
                comando = "datos <- read.table(\"" + nombreArchivo +
"\",header=TRUE)";
            else
                comando = "datos <- read.table(\"" + nombreArchivo +
"\",header=FALSE)";
            ejecutar(comando);

            //Se recupera la matriz de datos
            comando = "mode(datos) <- 'character'";
            ejecutar(comando);
            comando = "datos <- datos";
            ejecutar(comando);
            matrizDatos = recuperarMatrizNxN("datos");
        }

        //Se ingresa los datos de manera directa creando un listado de
valores y una matriz
        public void crearMatrizDatos(string valores, string cabecera, int
nro_columnas_matriz)
        {
            string comando = "";
            comando = "valores <- c(" + valores + ")";
            ejecutar(comando);
            //Se recupera los valores
```

```
            if (cabecera != "")
            {
                comando = "cabecera <- c(" + cabecera + ")";
                ejecutar(comando);
                comando = "datos <- matrix(valores, ncol = " +
nro_columnas_matriz.ToString() + ", byrow = F, dimnames =
list(NULL,cabecera))";
            }
            else
            {
                comando = "datos <- matrix(valores, ncol = " +
nro_columnas_matriz.ToString() + ", byrow = F)";
            }
            ejecutar(comando);

            //Se recupera la matriz de datos
            comando = "mode(datos) <- 'character'";
            ejecutar(comando);
            comando = "datos <- datos";
            ejecutar(comando);
            matrizDatos = recuperarMatrizNxN("datos");
        }

        //Se filtra los datos de la tabla que ya ha sido recuperado
        public void filtrarMatrizDatos(int clase_ini, int clase_fin, int
columna_ini, int columna_fin)
        {
            //Se recupera la matriz de datos
            asignar("datos", matrizDatos);
            //Se asigna la matriz de datos filtrado
            string comando = "datos_filtro <- datos[" +
clase_ini.ToString() + ":" + clase_fin.ToString() + "," +
columna_ini.ToString() + ":" + columna_fin.ToString() + "]";
            ejecutar(comando);
            //Se convierte toda la matriz en valores numericos
            comando = "mode(datos_filtro) <- 'numeric'";
            ejecutar(comando);
            //Se recupera la matriz filtrada
            matrizFiltro = recuperarMatrizNxN_Double("datos_filtro");
        }
```

```
        //Se recupera la matriz filtrada pero proyectada sobre la matriz
principal
        public void filtrarMatrizDatos_Proyeccion(bool estandarizar)
        {
            //Se verifica que se puede hacer la proyección sobre la matriz
principal
            if (matrizFiltro_Principal != null)
            {
                //Se recupera la matriz filtrada
                asignar("datos_filtro", matrizFiltro);
                //Se recupera la matriz filtrada principal
                asignar("datos_filtro_principal", matrizFiltro_Principal);
                string comando = "";
                //Se recupera la matriz proyectada
                if (estandarizar)
                    comando = "datos_filtro <-
predict(prcomp(datos_filtro_principal,scale=TRUE),datos_filtro)";
                else
                    comando = "datos_filtro <-
predict(prcomp(datos_filtro_principal,scale=FALSE),datos_filtro)";
                ejecutar(comando);
                //Se convierte toda la matriz en valores numericos
                comando = "mode(datos_filtro) <- 'numeric'";
                ejecutar(comando);
                //Se recupera la matriz filtrada
                matrizFiltro = recuperarMatrizNxN_Double("datos_filtro");
            }
        }

        //Se recupera los labes para la salida gráfica
        public void recuperarEtiquetas(int clase_ini, int clase_fin, int
columna_etiquetas)
        {
            string comando = "";

            if (columna_etiquetas != -1)
            {
                //Se recupera la matriz de datos
                asignar("datos", matrizDatos);
                //Se asigna la matriz de etiquetas
                comando = "etiquetas <- datos[" + clase_ini.ToString() +
":" + clase_fin.ToString() + "," + columna_etiquetas.ToString() + "]";
```

```
                ejecutar(comando);
                comando = "etiquetas <- matrix(etiquetas, nrow = 1)";
                ejecutar(comando);
            }
            else
            {
                int nro_etiqueta = 1;
                for (int i = clase_ini; i <= clase_fin; i++)
                {
                    if (nro_etiqueta == 1)
                        comando = "etiquetas <- c('E1'";
                    else
                        comando = comando + ",
'E"+nro_etiqueta.ToString()+"'";
                    if (i == clase_fin)
                        comando = comando + ")";
                    nro_etiqueta++;
                }
                ejecutar(comando);
                comando = "etiquetas <- matrix(etiquetas, nrow = 1)";
                ejecutar(comando);
            }
            //Se recupera la matriz de labels
            comando = "mode(etiquetas) <- 'character'";
            ejecutar(comando);
            comando = "etiquetas <- etiquetas";
            ejecutar(comando);
            etiquetas = recuperarMatrizNxN("etiquetas");
        }

        //Se realiza el análisis de los componentes principales
        public void AnalisisComponentesPrincipales(bool estandarizar)
        {
            //Se recupera la matriz filtrada
            asignar("datos_filtro", matrizFiltro);
            string comando;

            asignar("acp_sdev", ACP_SDEV);
            if (estandarizar)
                comando = "acp_sdev <- prcomp(datos_filtro,
scale=TRUE)$sdev";
            else
```

```
                comando = "acp_sdev <- prcomp(datos_filtro,
scale=FALSE)$sdev";
            ejecutar(comando);
            //Se recupera el PCA de la desviación estandar
            ACP_SDEV = recuperarMatrizN_Double("acp_sdev");

            if (estandarizar)
                comando = "acp_rot <- prcomp(datos_filtro,
scale=TRUE)$rotation";
            else
                comando = "acp_rot <- prcomp(datos_filtro,
scale=FALSE)$rotation";
            ejecutar(comando);
            //Se recupera el PCA de la rotación
            ACP_ROT = recuperarMatrizNxN_Double("acp_rot");

            if (estandarizar)
                comando = "acp_x <- prcomp(datos_filtro, scale=TRUE)$x";
            else
                comando = "acp_x <- prcomp(datos_filtro, scale=FALSE)$x";
            ejecutar(comando);
            //Se recupera el PCA de la rotación
            ACP_X = recuperarMatrizNxN_Double("acp_x");
        }

        //Se recupera los valores propios
        public double[] recuperarValoresPropios()
        {
            //Se recupera ACP
            asignar("acp_sdev", ACP_SDEV);
            string comando;
            comando = "vp <- acp_sdev^2";
            ejecutar(comando);
            //Se recupera los valores propios
            valoresPropios = recuperarMatrizN_Double("vp");
            return recuperarMatrizN_Double("vp");
        }

        //Se recupera los valores propios en forma porcentual para
mostrarlo al usuario
        public double[] recuperarValoresPropios_Porcentual()
        {
```

```
            //Se recupera ACP
            asignar("acp_sdev", ACP_SDEV);
            string comando;
            comando = "vp <- acp_sdev^2";
            ejecutar(comando);
            comando = "vp_porcentual <- vp/sum(vp) * 100";
            ejecutar(comando);
            //Se recupera los valores propios
            valoresPropios = recuperarMatrizN_Double("vp_porcentual");
            return recuperarMatrizN_Double("vp_porcentual");
        }

        //Se recupera los componentes seleccionados para mostrar el
resultado final
        public void seleccionarComponentes(int[] seleccionados)
        {
            //Se recupera ACP
            asignar("acp_x", ACP_X);
            string comando;
            //Se pregunta por el número de valores propios seleccionados
            int nro = seleccionados.Length;
            //Se actualiza "cptes" que luego será graficado
            for (int i = 0; i < nro; i++)
            {
                if (i == 0)
                    comando = "cptes <- acp_x[," +
seleccionados[i].ToString() + "]";
                else
                    comando = "cptes <- cbind(cptes,acp_x[," +
seleccionados[i].ToString() + "])";
                ejecutar(comando);
            }

            //Se convierte toda la matriz en valores numericos
            comando = "mode(cptes) <- 'numeric'";
            ejecutar(comando);

            //Se recupera la matriz de componentes
            cptes = recuperarMatrizNxN_Double("cptes");

            //Una vez recuperado los componentes, se recupera la
variabilidad
```

```
            for (int i = 0; i < nro; i++)
            {
                if (i == 0)
                    comando = "v <- var(cptes[," +
seleccionados[i].ToString() + "])";
                else
                    comando = comando + " + var(cptes[," +
seleccionados[i].ToString() + "])";
            }
            ejecutar(comando);
            variabilidad = recuperar_Double("v");

            //Se actualiza la matriz de Variabilidad
            if ((MatrizVariabilidad == null) && (MatrizNombresVar ==
null))
            {
                MatrizVariabilidad = new double[1];
                MatrizVariabilidad[0] = variabilidad;
                MatrizNombresVar = new string[1];
                MatrizNombresVar[0] = "Vuelta 1";
            }
            else
            {
                int nroVariabilidad = MatrizVariabilidad.Length + 1;
                Array.Resize(ref MatrizVariabilidad, nroVariabilidad);
                MatrizVariabilidad[nroVariabilidad - 1] = variabilidad;
                int nroNrobresVar = MatrizNombresVar.Length + 1;
                Array.Resize(ref MatrizNombresVar, nroNrobresVar);
                MatrizNombresVar[nroNrobresVar - 1] = "Vuelta
"+nroNrobresVar.ToString();
            }
        }

        //Se recupera la matriza de interpretación con resultado double
        public object[,] recuperarMatrizInterpretacion(int
nro_columnas_matriz)
        {
            //Se recupera la matriz de datos
            asignar("datos_filtro", matrizFiltro);
            //Se recupera la matriz de componentes
            asignar("cptes", cptes);
```

```
            //Se recupera la matriz de interpretación
            string comando = "";
            comando = "matriz_interpretacion <- cor(datos_filtro,cptes)";
            ejecutar(comando);

            //Se recupera la matriz de datos
            comando = "mode(matriz_interpretacion) <- 'character'";
            ejecutar(comando);
            comando = "matriz_interpretacion <- matriz_interpretacion";
            ejecutar(comando);
            matrizInterpretacion =
recuperarMatrizNxN("matriz_interpretacion");

            return matrizInterpretacion;
        }

        //Se determina los límites de los ejes basado en la matriz
principal
        public void DeterminarLimitesGrafico(int aumento)
        {
            //Se recupera los componentes
            asignar("cptes", cptes);
            //Se determina los límites
            string comando = "";
            comando = "XLIM <- c(-max(abs(cptes[,1])) - " +
aumento.ToString() + ",max(abs(cptes[,1])) + " + aumento.ToString() + ")";
            ejecutar(comando);
            XLim = recuperarMatrizN_Double("XLIM");
            comando = "YLIM <- c(-max(abs(cptes[1,])) - " +
aumento.ToString() + ",max(abs(cptes[1,])) + " + aumento.ToString() + ")";
            ejecutar(comando);
            YLim = recuperarMatrizN_Double("YLIM");
        }

        //Mostrar salida gráfica del boxplot de la matriz filtrada
        public void graficarBoxPlot_MatrizFiltrada()
        {
            //Se recupera la matriz filtrada
            asignar("datos_filtro", matrizFiltro);
            string comando = "";
            comando = "boxplot(datos_filtro)";
            graficar(comando);
```

```
        }

        //Mostrar salida gráfica de matriz de interpretación
        public void graficarMatriz_Interpretacion()
        {
            //Se recupera la matriz de interpretación
            asignar("matriz_interpretacion", matrizInterpretacion);
            string comando = "";
            comando = "plot(matriz_interpretacion, pch = 16, col =
'RED')";
            graficar(comando);
        }

        //Mostrar leyenda en el grafico
        public void graficarLeyenda(string leyenda, string leyendaCol)
        {
            string comando = "";

            //Se colora la leyenda
            if (leyenda != "")
            {
                comando = "legend(\"topright\", c(" + leyenda + "), fill=
c(" + leyendaCol + "), cex = 0.6)";
                graficar(comando);
            }
        }

        //Mostrar salida gráfica de los resultados
        public void graficaResultado_Simple(int[] seleccionados, string
colorPtos, string colorH, string colorV, bool determinarLimites, bool
mostrarEtiquetas)
        {
            //Se recupera ACP
            asignar("cptes", cptes);

            string comando;

            //Se grafica "cptes"
            comando = "plot(cptes, pch=16";
            if (colorPtos != "")
                comando = comando + " , col = rgb(" + colorPtos + ",
maxColorValue=255)";
```

```
                else
                    comando = comando + " , col = 'BLACK'";
                if (determinarLimites)
                {
                    //Se recupera los límites de los ejes
                    asignar("XLIM", XLim);
                    asignar("XLIM", XLim);
                    comando = comando + " , xlim = XLIM, ylim = YLIM)";
                }
                else
                {
                    comando = comando + ")";
                }
                graficar(comando);

                //Se pregunta por el número de valores propios seleccionados
                int nro = seleccionados.Length;

                //Se grafica los labels
                if ((nro > 0) && (mostrarEtiquetas) && (etiquetas != null))
                {
                    //Se recupera los labels
                    asignar("etiquetas", etiquetas);

                    comando = "";
                    comando = "text(";
                    //for (int i = 0; i < nro; i++)
                    for (int i = 0; i < 2; i++)
                    {
                        if (i == 0)
                            comando = comando + "cptes[," +
seleccionados[i].ToString() + "]+0.1";
                        else
                            comando = comando + ",cptes[," +
seleccionados[i].ToString() + "]+0.2";
                        //if (i + 1 == nro)
                        if (i + 1 == 2)
                            comando = comando + ",labels=etiquetas,cex=0.6)";
                    }
                    graficar(comando);
                }
```

```
            //Se coloca los colores a las lineas
            try
            {
                if (colorH != "")
                {
                    comando = "abline(h=0, col = rgb(" + colorH + ",
maxColorValue=255))";
                    graficar(comando);
                }
                if (colorV != "")
                {
                    comando = "abline(v=0, col = rgb(" + colorV + ",
maxColorValue=255))";
                    graficar(comando);
                }
            }
            catch (Exception exception)
            { }
        }

        //Mostrar salida gráfica del resultado final y del boxplot
        public void graficaResultado_Simple_conBoxplot(int[]
seleccionados, string colorPtos, string colorH, string colorV, bool
mostrarEtiquetas)
        {
            string comando = "";
            comando = "par(mfrow=c(1,2))";
            graficar(comando);
            graficarBoxPlot_MatrizFiltrada();
            graficaResultado_Simple(seleccionados, colorPtos, colorH,
colorV, false, mostrarEtiquetas);
        }

        //Graficar sobre un resultado anterior en consultas múltiples
        public void graficarResultado_Multiple(int[] seleccionados, string
colorPtos, string colorH, string colorV, int aumento, bool
mostrarEtiquetas, bool mostrarFlechas, bool esMatrizPrincipal, bool
adjuntar)
        {
            string comando = "";

            //Se pregunta si se desea graficar la matriz principal
```

```
            if (esMatrizPrincipal)
            {
                //Se pregunta si se va a adjuntar un gráfico en el futuro
                if (adjuntar)
                {
                    comando = "par(mfrow=c(1,2))";
                    graficar(comando);
                }

                //Se determina los límites del gráfico
                DeterminarLimitesGrafico(aumento);
                //Se realiza el gráfico
                graficaResultado_Simple(seleccionados, colorPtos, colorH,
colorV, true, mostrarEtiquetas);
                //Se recuerda los componentes
                cptes_Anterior = cptes;
                //Se recuerda los colores
                color_Anterior = colorPtos;
            }
            else
            {
                comando = "par(new=TRUE)";
                graficar(comando);
                //Se realiza el gráfico
                graficaResultado_Simple(seleccionados, colorPtos, "", "",
true, mostrarEtiquetas);
                //Se grafica flechas para ver el desplazamiento
                if (mostrarFlechas)
                {
                    //Se recupera los componentes anteriores y el actual
                    asignar("cptes_ant", cptes_Anterior);
                    asignar("cptes", cptes);

                    comando = "arrows(cptes_ant[,1] , cptes_ant[,2] ,
cptes[,1] , cptes[,2] , col=rgb(" + color_Anterior + ",
maxColorValue=255), length = 0.10)";
                    graficar(comando);
                }
                //Se recuerda los componentes
                cptes_Anterior = cptes;
                //Se recuerda los colores
                color_Anterior = colorPtos;
```

```
            }
        }

        //Graficar la evolución de la información o variabilidad
        public void graficarVariabilidad(string[] colores)
        {
            string comando = "";

            //Se recupera la matriz de variabilidad
            asignar("MatrizVar", MatrizVariabilidad);
            //Se recupera el nombre de las vueltas para mostrar la
variabilidad
            asignar("Nombres", MatrizNombresVar);
            //Se grafica la variabilidad
            if (colores != null)
            {
                comando = "barplot.x.location <- barplot(MatrizVar,
main='Variabilidad'";
                int nro = colores.Length;
                for (int i = 0; i < nro; i++)
                {
                    if (i == 0)
                        comando = comando + ", col = c(rgb(" + colores[i]
+ ", maxColorValue=255)";
                    else
                        comando = comando + ", rgb(" + colores[i] + ",
maxColorValue=255)";
                    if (i == nro - 1)
                        comando = comando + ")";
                }
                comando = comando + ", xlab='Vueltas', names.arg=Nombres,
cex.names=0.8)";
            }
            else
                comando = "barplot.x.location <- barplot(MatrizVar,
main='Variabilidad', col = 'darkblue', xlab='Vueltas', names.arg=Nombres,
cex.names=0.8)";
            graficar(comando);

            //Se grafica punto en el barplot
            comando = "points(x = barplot.x.location, y = MatrizVar, col =
1, pch = 16)";
```

```
            graficar(comando);

            //Se grafica lineas entre los puntos
            for (int i = 1; i < MatrizVariabilidad.Length; i++)
            {
                //comando = "arrows(barplot.x.location[" + i.ToString() +
",],MatrizVar[" + i.ToString() + "],barplot.x.location[" + (i +
1).ToString() + ",],MatrizVar[" + (i + 1).ToString() + "],col=\"BLACK\",
length = 0.10)";
                comando = "segments(barplot.x.location[" + i.ToString() +
",],MatrizVar[" + i.ToString() + "],barplot.x.location[" + (i +
1).ToString() + ",],MatrizVar[" + (i + 1).ToString() + "],col=\"BLACK\")";
                graficar(comando);
            }
        }

        //Se desea interactuar con el usuario
        public void interactuarUsuario()
        {
            string comando = "";
            //Se recupera el nombre de las etiquetas
            asignar("etiquetas", etiquetas);
            //Se recupera los componentes
            asignar("cptes", cptes);
            //Se le asigna las etiquetas a los nombres de las filas de
cptes
            comando = "row.names(cptes) <- etiquetas";
            ejecutar(comando);
            //Se abra la libreria rggobi
            comando = "library(rggobi)";
            graficar(comando);
            comando = "g <- ggobi(cptes)";
            graficar(comando);
            comando = "d <- displays(g)[[1]]";
            graficar(comando);
            comando = "imode(d) <- \"Move\"";
            graficar(comando);
        }

        //Se recupera la matriz de componentes y se identifica que punto
se ha cambiado
```

```
        public double[,] recuperar_Identificar_PtoCambiado(int[]
seleccionados, bool estandarizar, ref int filaCambiada)
        {
            double[,] salida = null;

            string comando = "";
            //Se recupera los componentes
            asignar("cptes", cptes);
            //Se recupera los nuevos componentes cambiados por el usuario
            comando = "cptesUser <- dataset(d)";
            graficar(comando);

            int columnas = seleccionados.Length;
            int filas = (cptes.Length / columnas);

            for (int i = 0; i < filas; i++)
            {
                if (i == 0)
                    comando = "matrizCptesUser <-
matrix(c(cptesUser[1]),nrow=1)";
                else
                    comando = "matrizCptesUser <-
rbind(matrizCptesUser,c(cptesUser[" + (i + 1).ToString() + "]))";
                ejecutar(comando);
            }
            comando = "mode(matrizCptesUser) <- 'numeric'";
            ejecutar(comando);

            cptes_CambiadoUser =
recuperarMatrizNxN_Double("matrizCptesUser");
            //Se realiza una comparación
            //Se pregunta por el número de valores propios seleccionados
            filaCambiada = 0;
            bool cambiado = false;

            //Se recorre cada matriz comparando cual es el elemento que ha
cambiado
            for (int i = 0; i < filas; i++)
            {
                for (int j = 0; j < columnas; j++)
                {
                    //Se reconoce que el usuario haya cambiado el punto
```

```
                    if (Math.Round(System.Convert.ToDouble(cptes[i, j]),
5) != Math.Round(System.Convert.ToDouble(cptes_CambiadoUser[i, j]), 5))
                    {
                        //El usuario ha cambiado en la fila i
                        cambiado = true;
                        break;
                    }
                }
                if (cambiado)
                {
                    filaCambiada = i;
                    break;
                }
            }

            //Se pregunta si el usuario ha cambiado los puntos
            if (cambiado)
            {
                //Se recupera la matriz de rotacion de acp
                asignar("acp_rot", ACP_ROT);
                //Se recupera la matriz filtrada
                asignar("matriz_filtro", matrizFiltro);
                //Se recupera el punto cambiado
                for (int j = 0; j < columnas; j++)
                {
                    if (j == 0)
                        comando = "punto <- c(c(" +
cptes_CambiadoUser[filaCambiada, j].ToString();
                    else
                        comando = comando + ", " +
cptes_CambiadoUser[filaCambiada, j].ToString();
                    if (j == columnas - 1)
                        comando = comando + ")";
                }
                if (filas - columnas > 0)
                    comando = comando + ", rep(0, "+(filas -
columnas).ToString()+"))";
                else
                    comando = comando + ")";
                ejecutar(comando);
                //Se crea una matriz con el punto
                comando = "matrizPuntos <- matrix(punto, nrow=1)";
```

```
                ejecutar(comando);
                comando = "Z <- matrizPuntos %*% t(acp_rot)";
                ejecutar(comando);
                //Se halla la media
                comando = "media <- mean(matriz_filtro)";
                ejecutar(comando);

                //Se recupera las respuesta originales dependiendo de la
estandarización
                if (!estandarizar)
                {
                    comando = "Rptas_Var <- Z + media";
                    ejecutar(comando);
                    salida = recuperarMatrizNxN_Double("Rptas_Var");
                }
                else
                {
                    //Se halla la desviación estándar
                    comando = "DesvStandar <- sd(matriz_filtro)";
                    ejecutar(comando);
                    comando = "Rptas_Cor <- media + DesvStandar*Z";
                    ejecutar(comando);
                    salida = recuperarMatrizNxN_Double("Rptas_Cor");
                }

                //Se actualiza los componentes por los seleccionados por
el usuario
                cptes = cptes_CambiadoUser;
            }

            return salida;
        }

        //Para cerra la interfaz de ggobi
        public void cerrarGgobi()
        {
            string comando = "";
            comando = "close(g)";
            try
            {
                graficar(comando);
            }
```

```
            catch(Exception exception)
            { }
    }
```

Anexo III: Modelo de encuesta

RONDA 1

MEJORAMIENTO DE LA PRODUCCIÓN, CONTROL DE CALIDAD Y PROCESOS AGROINDUSTRIALES. NUEVOS BIO-PRODUCTOS Y BIOTECNOLOGÍAS PARA EL MEDIO AMBIENTE

1. Propagación de clonación in vitro de genotipos vegetales de calidad genética y sanitaria

Pensando de aquí al año 2015, ¿cuál será la **pertinencia** de este fenómeno para ... (Marque de 1 a 10, donde 1=nada pertinente y 10=absolutamente pertinente)

	Marque aquí
P1. Mejorar la competitividad de la industria nacional y posibilitar un aumento de las exportaciones	
P2. Desarrollar la investigación básica y aplicada	
P3. Satisfacer la demanda de los consumidores	
P4. Satisfacer la demanda de los productores	
P5. Preservar el medio ambiente	

Pensando de aquí al año 2015, ¿cuál será la **factibilidad** de este fenómeno en términos de ...(Marque de 1 a 10, donde 1=nada factible y 10=absolutamente factible)

	Marque aquí
P6. La disponibilidad de recursos económicos	
P7. La capacidad de los recursos humanos	
P8. La disponibilidad de las tecnologías adecuadas	
P9. La existencia de un entorno institucional y legal adecuado	

Si Uruguay se propusiera priorizar el tema arriba establecido de aquí al año 2015, ¿Qué grado de importancia le asignaría Ud. a las siguientes **recomendaciones**? (Marque de 1 a 10, donde 1=recomendación irrelevante y 10=recomendación imprescindible)

	Marque aquí
P10. Formación de alianzas estratégicas con centros y/o empresas del exterior	
P11. Promover la vinculación tecnológica entre el sector público y el sector privado	
P12. Brindar estímulos económico-fiscales del Estado (subvenciones, exenciones y desgravaciones fiscales, etc.)	
P13. Brindar estímulos estatales no económicos (regulación normativa, políticas estatales, apoyos, etc.)	

¿Qué otras recomendaciones específicas considera Ud. sería necesario tener en cuenta? (Escriba 2 en orden de importancia descendente)

(P14) 1. _____

(P15.) 2. _____