TRABAJO DE FIN DE GRADO

Grado en Estadística

Título: Aplicativos para seguir un curso asíncrono: resultados.

Autora: Nerea Bielsa González

Director: José Antonio González Alastrué

Departamento: Estadística e Investigación operativa

Convocatoria: 09/07/15

Universitat de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

RESUMEN

Introducción: Mi objetivo es desarrollar aplicativos Shiny dinámicos para analizar datos acumulativos del curso de Bioestadística para no estadísticos. **Métodos:** el 12 de enero de 2015, un total de 93 alumnos habían realizado 4459 ejecuciones en e-status. He (1) depurado los datos; (2) analizado de la información disponible; (3) adaptado a Shiny para estudiar la dificultad de los ejercicios, su concordancia y el rendimiento de los estudiantes. R**esultados:** el análisis de concordancia de Bland-Altman permite visualizar el nivel de dificultad relativa; y el análisis de componentes principales muestra una primera dimensión de tamaño que retiene el 60.53% de varianza, apoyando la idea de que todos los ejercicios deben ser considerados en la evaluación final. **Discusión:** He mostrado las posibilidades de Shiny para facilitar el seguimiento de un curso no presencial, si bien su transportabilidad a otros entornos es incierta.

PALABRAS CLAVE

ACP, Shiny, Gráfico de Bland Altman, programación reactiva, R-studio.

ABSTRACT

Asynchronous applications to follow a course: results. Introduction: my goal is to develop dynamic applications with Shiny to analyze cumulative data from the online "Biostatistics course for non-statisticians". **Methods:** by January 12, 2015, a total of 93 students had conducted 4459 executions in e-status. I (1) debugged the data; (2) reviewed the available information; (3) developed Shiny tools to study the exercises difficulty, their consistency and the student performance. **Results:** The Bland-Altman agreement analysis shows their relative difficulty; and the principal component analysis highlights a first dimension retaining 60.53% variance, that supports the idea that all exercises should be considered in the final evaluation. **Discussion:** I have shown the Shiny possibilities to facilitate the monitoring of an online course, although its portability to other environments is uncertain.

KEY WORDS

ACP, Shiny, Bland Altman plot, reactive programming, R-studio.

CLASIFICACIÓN DEL TFG EN AMERICAN MATHEMATICAL SOCIETY

Siendo el punto 62-XX: Statistics, el proyecto se puede clasificar dentro de:

62-04: Explicit machine computation and programs (not the theory of computation or programming)

62-07: Data analysis

62-09: Graphical methods

62H25: Factor analysis and principal components; correspondence analysis

68M20: Performance evaluation; queueing; scheduling

ÍNDICE

I. INTRODUCCIÓN

El curso online "Bioestadística para no estadísticos" es principalmente *asíncrono*, en el sentido de que el alumno escoge sus tiempos y sus ritmos, lo que dificulta el seguimiento de su avance en los diferentes módulos del curso. Esto implica que seguir los alumnos requiere un proceso dinámico actualizable en cualquier momento. El objetivo principal del proyecto es conseguir un proceso sencillo, eficiente y actualizable de forma rápida que permita obtener este progreso de los alumnos y obtener los indicadores necesarios para realizar los análisis pertinentes.

Para ello he recurrido a la extensión de R-studio llamada Shiny, que permite crear aplicativos sencillos, con representaciones de gráficos, que facilitan los análisis de los alumnos y ejercicios. Además, también permite realizar aplicativos dinámicos, por lo que es muy adecuado para este conjunto de datos.

He creado aplicativos para analizar las diferentes piezas del curso, que son: alumnos y ejercicios, agrupados dentro de módulos. Para el análisis de resultados he utilizado 2 técnicas estadísticas, el gráfico de Bland Altman (BA) y el análisis de componentes principales (ACP). El primero pretende estudiar la dificultad de los ejercicios y la forma de su relación, si existe, con la nota global. El segundo pretende, por un lado, validar la premisa implícita detrás de una nota promedio: que todos los ejercicios contribuyen de forma positiva a una dimensión de tamaño que ordene a los alumnos según su rendimiento. En segundo lugar, pretende visualizar si éstos tienen un comportamiento diferencial en los distintos ejercicios a lo largo del seguimiento (dimensión de forma).

El trabajo está dividido en 3 partes principales: en primer lugar incluye la metodología, donde se explican los recursos informáticos y estadísticos utilizados, en este caso Shiny y los métodos ya citados de BA y ACP.

En segundo lugar incluye los resultados: la descriptiva de los datos y en los dos puntos restantes comenta el análisis estadístico de los datos obtenidos.

Por último, discute las implicaciones de estos resultados y aventura algunas conclusiones.

En cuanto a los agradecimientos en primer lugar, agradecer a José Antonio González la labor que ha realizado como director de trabajo de fin de grado, sobre todo por la sensación de trabajo en equipo y por el grado de implicación. Agradecer también a Erik Cobo, Jordi Cortés la confianza depositada en mí durante este tiempo y, junto a Roser Rius, agradecerles su imprescindible colaboración en el proyecto.

En segundo lugar, agradecer a mi madre, a Vicky, a Rut y en especial a mis abuelos, el apoyo emocional y la motivación que me han dado para poder llegar hasta aquí; a mi padre le agradezco

este apoyo y esta motivación, y además tengo que agradecerle, la manera en que me ha hecho a mí misma y los valores que me ha hecho entender a los largo de estos cuatro años.

Gracias también, a mis compañeros de clase Dani, Anna, Patricia, Raquel y, en especial, a Xavi y a Lluís, por la paciencia que han tenido conmigo en los momentos de estrés durante las horas compartidas, que no han sido pocas, y los ánimos que me han dado.

Agradecer también, a Sergi, la capacidad que ha tenido para mantenerme cuerda en los momentos críticos a lo largo del trabajo de fin de grado (y de los cuatro años de grado), las risas compartidas, y sobre todo su apoyo en los momentos complicados; y agradecer a Pili su apoyo y confianza, aunque haya sido en la distancia.

Y por último, gracias a Gala, por animarme a seguir y hacerme ver que podía con todo con sólo darme un lametazo.

II. METODOLOGÍA

II.1. Estructura, volcado y tratamiento de datos

El curso está formado por 5 módulos teóricos consecutivos y 1 práctico. Cada módulo teórico tiene 3 capítulos, y dentro de cada capítulo, hay un número de ejercicios, clasificados en problemas numéricos prácticos y test teóricos o de Guías de publicación (*Reporting Guidelines*: RG). Este número de ejercicios no es el mismo para todos los capítulos, es decir, varía según el capítulo. Por último, dentro de cada ejercicio hay un número determinado de preguntas, que también varía dependiendo del ejercicio.

Más con fines de aprendizaje que evaluativos, los alumnos pueden repetir los ejercidos tanto como quieran. Por ello, cada ejercicio tiene un número de ejecuciones determinado y cada vez que se realiza la carga de datos es posible que este número haya aumentado. Cada ejercicio permite un tiempo máximo pre-determinado para finalizar la ejecución.

El número de alumnos también es variable y, aunque actualmente no tenga aplicación práctica, los alumnos están divididos en grupos según el mes en el que iniciaron el curso.

El proceso de datos empieza con su carga desde el directorio http://ka.upc.es/test-r/. Se realizan 4 cargas de datos, es decir, se trabaja con 4 *data.frames* (conjuntos de datos) diferentes.

El primer *data.frame* (http://ka.upc.es/test-r/asignaturas.Rdata) tiene el nombre de *asig* y lista los módulos realizados en los diferentes años, con un identificador para cada uno de ellos. Está formado por 15 filas (5 módulos por tres años) y 2 columnas que son el nombre del módulo (*Nombre*) y el identificador del módulo (*Id*).

El segundo *data.frame* (http://ka.upc.es/test-r/ejecuciones.Rdata), denominado *ejec*, contiene 5709 filas y 5 columnas actualmente: cada fila es una ejecución de un alumno y las columnas indican el ejercicio del cual se ha realizado la ejecución (*Prob*), el alumno que ha realizado la ejecución (*Login*), la nota obtenida en el ejercicio (*Nota*), el identificador del módulo y año en que se encuentra el ejercicio (*Asig*); y la fecha y hora de la ejecución (*Fecha*).

El tercer *data.frame* (http://ka.upc.es/test-r/numalumnos.Rdata), denominado *nual*, contiene 38 filas y 3 columnas. Cada fila es el número de alumnos (*count(*)*) que hay en los diferentes grupos realizados a partir del mes en que el alumno se inscribió en el curso (*nombre*) y en los diferentes módulos (*asignatura_id*). Este *data.frame* no se utiliza ya que actualmente no se realiza ninguna clasificación de alumnos por mes de iniciación del curso.

Por último, el cuarto *data.frame* (http://ka.upc.es/test-r/problemas.Rdata) se denomina *prob*. Contiene todos los ejercicios de los diferentes módulos de los tres años; y está formado por 256 filas y 3 columnas. Cada fila representa un ejercicio, y muestra para cada uno: el identificador del módulo y el año al que pertenece (*Asig*), su identificador (*Idprob*) y su nombre (*Nombre*). La

mayoría de los ejercicios, aunque pertenezcan a módulos de años diferentes, tienen el mismo identificador, por lo que cada ejercicio se repite 3 veces dentro del *data.frame*, una por cada año. El tratamiento de los ejercicios que tienen diferente contenido según el año o que tienen un identificador diferente se explica más adelante.

A continuación, se muestran las relaciones entre los diferentes *data.frames:*

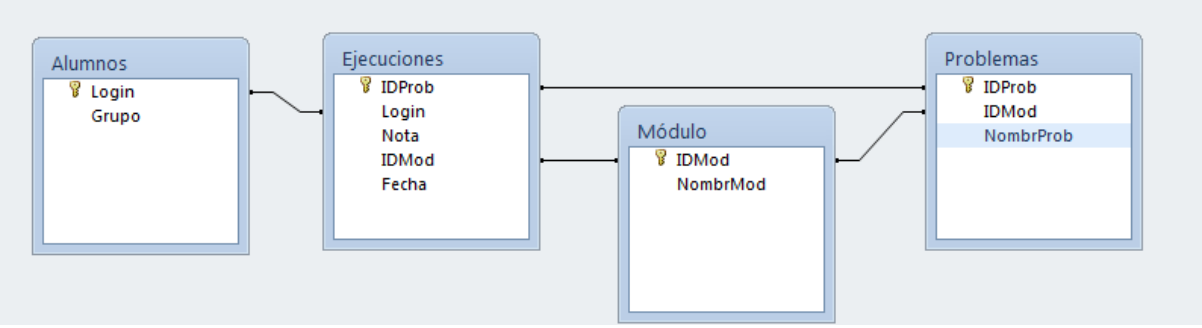


Figura II.1: Esquema de relaciones de los diferentes *data.frames*

Por su parte, la depuración consiste en lo siguiente.

En primer lugar, al *data.frame prob* se le añade una columna nueva con el módulo al que pertenece cada ejercicio sin distinción por años, es decir, a la nueva columna *mod* sólo se le asignan los valores 1, 2, 3, 4, y 5, sin ningún otro indicador que informe sobre en qué año se encuentra el módulo en el que está incluido el ejercicio.

En segundo lugar, se identifican los ejercicios no puntuables, y aquellos que tienen ejercicios equivalentes. Se entienden como no puntuables a los que permiten recoger la satisfacción, dedicación y valoración de su propio trabajo en el capítulo; así como los ejercicios denominados "Mis notas", que muestran la calificación obtenida en el módulo completo.

Para estos ejercicios no puntuables, el *data.frame prob* crea una columna nueva, llamada *NO*, que toma valor 1 si el ejercicio es no puntuable y valor "NA" si el ejercicio es puntuable. El siguiente paso es crear la lista *ppmod* que, diferenciando por módulos, muestra los ejercicios definidos como puntuables.

Además de seleccionar las ejecuciones de ejercicios puntuables, también elimina los alumnos "ficticios", creados para realizar pruebas. Esta selección se guarda en el *data.frame ejs.x*. Además, se le añade la columna *mod* para indicar el módulo del ejercicio.

El siguiente paso es el tratamiento de los ejercicios equivalentes. Se entienden por ejercicios equivalentes, aquellos que cambian por año pero mantienen el mismo objetivo y las mismas preguntas.

Para tratarlos, se crea la lista *equiv* que los contiene así como al ejercicio al que equivalen. El *data.frame ejs.x* substituye el identificador del ejercicio antiguo (equivalente) por el del nuevo (al

que equivale), de modo que los alumnos que tienen ejecuciones en los ejercicios equivalentes antiguos pasan a tenerlas en los ejercicios nuevos.

También actualiza la lista *ppmod* para eliminar los ejercicios equivalentes y dejar solo los identificadores de aquellos que interesan.

Todos estos pasos están implementados en funciones en un fichero de R aparte. En cada aplicativo realizado se cargan dos ficheros, el primero de nombre *functions12.R*, que es el que contiene las funciones; el segundo *parameters.R*, que contiene todos los parámetros necesarios en el código.

II.2. Introducción a Shiny

II.2.1. Características de Shiny

Shiny es una extensión de R con código abierto que proporciona un marco de trabajo elegante, sencillo y potente. Fue creado en 2012 por R-studio (extensión de R), para crear aplicaciones interactivas orientadas al análisis y visualización de datos a través de R. Además, permite compartir los resultados desarrollados de forma rápida y automática, ya que cada programa ejecutado se guarda en un servidor creado al empezar a trabajar con Shiny. Una de sus mayores ventajas es que para construir las aplicaciones, no se necesitan nociones básicas de HTML, CSS, JavaScript ni de ningún lenguaje de programación ajeno a R[(7)].

Incluye, también, un CSS3 UI (CSS Módulo de Interfaz de usuario básico de Nivel 3), que permite especificar las propiedades y los valores de la interfaz de usuario a nivel 3 de CSS en formato HTML y XML.

Las aplicaciones realizadas con Shiny se pueden ejecutar en el ordenador del usuario (local), o pueden ser almacenadas en el servidor y ejecutadas directamente en el ordenador del usuario. Para utilizar Shiny de forma local hay que instalar en R su extensión, que se realiza del mismo modo que la de cualquier otro paquete: en primer lugar, se instala el paquete (`install.packages("shiny")`) y, en segundo lugar, se carga la librería Shiny (`library(shiny)`). Esto permite visualizar los *outputs* de Shiny dentro de R; cuando se indica a R que el código se tiene que actualizar, se abrirá una pantalla nueva que mostrará el aplicativo. En caso de querer mostrar los aplicativos en un servidor web, accesible para todos aquellos a los que se les proporcione la *url*, la instalación es más compleja y es necesario solicitar una licencia [(16)].

En cuanto a soportes de ayuda, Shiny dispone de los foros de comunidad y de la documentación proporcionada por los desarrolladores, como una galería con ejemplos para mostrar su funcionamiento. Los ejemplos disponibles en la página han sido creados y cedidos por usuarios de Shiny[(9)].

II.2.2. Como funciona Shiny

Las aplicaciones Shiny tienen dos componentes: una interfaz de usuario, definida en un archivo de origen denominado *ui.R*, y un script de servidor, definido en un archivo de origen denominado *server.R*. Esto facilita su comprensión, ya que permite distinguir entre lo que el usuario va ver (la interfaz de usuario o *ui*) y como se tratan los datos (el controlador o *server*).

Se entiende por interfaz de usuario la aplicación que Shiny muestra y que, mediante los menús pertinentes, permite actualizar el programa.

El fichero *ui.R* contiene las instrucciones que definen la interfaz de usuario de la aplicación y el fichero *server.R* contiene la lógica de la aplicación, es decir, las instrucciones que realizan los cálculos o los gráficos necesarios.

Además, la interfaz de usuario (*ui.R*) explora las diferentes opciones de tratamiento de datos codificadas en el controlador (*server.R*). Estas opciones aparecen como un menú que el usuario adapta a los datos con los que está trabajando.

El intercambio de datos de *input* y *output* entre las dos partes de la aplicación se hace de manera automática, es decir, basta con modificar algún valor del menú (actualización de *input*) para que tanto *ui.R* como *server.R* se ejecuten y se muestre el nuevo *output*; muchas veces se refiere a Shiny como "programación reactiva"[(5)] ya que permite construir una interfaz que responde inmediatamente a cualquier cambio que el usuario hace. Del mismo modo se refiere a sus funciones como funciones que permiten crear "variables reactivas"[(7)], haciendo referencia a aquellas variables que serán modificadas cada vez que se realice un cambio en el *input*.

Shiny también ofrece la posibilidad de que el código no se re-ejecute automáticamente cuando se modifica una de sus opciones, ya que, si el coste computacional es elevado, sería muy ineficiente. Shiny incluye algunas opciones (como un botón *submit*) que permite manipular todas las opciones sin re-ejecutar el código; la ejecución definitiva se efectúa cuando se acciona dicho botón.

Para construir la interfaz de usuario, Shiny ofrece una colección básica de "*Widgets*" o elementos de control (como botones, listas, etc…), que permiten dar la apariencia deseada y requerida para el manejo y visualización de los datos.

II.2.3. Cómo construir una aplicación Shiny

Cada aplicación requiere un código de R individual que refiera a los dos códigos necesarios, es decir, *ui.R* y *server.R*.

Así, un aplicativo Shiny se compone, al menos, de estos dos ficheros, que pueden complementarse con cualquier dato adicional como guiones, funciones u otros recursos para apoyar la aplicación. Estos ficheros deben estar contenidos en un mismo directorio con el nombre de la aplicación.

A continuación, se profundiza un poco más en cada uno de ellos.

II.2.3.1. UI

Para crear la interfaz de usuario se necesitan tres funciones básicas: el Título, el *Input* y el *Output.*

Título

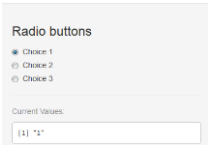
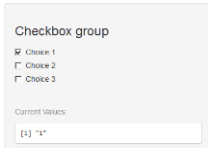
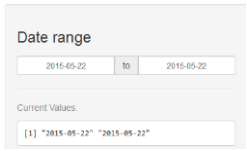
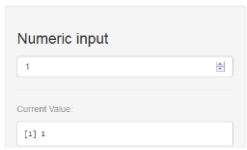
El título que aparecerá en el aplicativo se define con la función *headerPanel*.

Input

Dentro de la interfaz de usuario (*ui.R*) hay que definir los *inputs* que se pasan al archivo del servidor con los que éste es ejecutado (y re-ejecutado). Estos *inputs* aparecen en la aplicación en forma de barra lateral o menú; es decir, cada vez que cambia o selecciona un valor de este menú, Shiny ejecuta el código y muestra en pantalla los valores de *output* obtenidos con este/os nuevo/s *input/s*. La ubicación de esta barra lateral o menú también se define en la interfaz de usuario.

Existen muchas opciones (o *Widgets*) para ver e introducir *inputs* en el menú, Shiny proporciona una página donde los muestra, con un ejemplo y el código específico para cada uno de ellos.

La Tabla II.1 resume los *inputs* más utilizados y cómo permiten tratar los datos.

Función	**Imagen**	**Capacidad**
`radioButtons`		Permite seleccionar solo una opción
`checkboxGroupInput`		Permite seleccionar varias opciones
`dateRangeInput`		Permite seleccionar un rango de fechas
`sliderInput`		Permite seleccionar un rango de valores
`numericInput`		Permite seleccionar un valor numérico

`selectInput`		Permite escoger una opción mediante un desplegable

Tabla II.1: Resumen de las funciones de declaración de *inputs*, apariencia y datos que permiten tratar.

Output

El *output* funciona de forma similar al *input*. La interfaz de usuario especifica las funciones para ajustar la ventana de *outputs* a la apariencia deseada (tablas, gráficos…) y especifica la ubicación de cada uno de estos elementos.

El código de servidor define los *outputs* a mostrar y, mediante los *inputs* anteriores, ejecuta el código y muestra en la ventana de la aplicación el *output* pertinente a estos *inputs*.

La Tabla II.2 muestra las funciones de *output* más comunes.

Función	**Output**
`htmlOutput`	Crea un elemento de salida HTML
`imageOutput`	Crea una imagen como elemento de salida
`plotOutput`	Crea un gráfico como elemento de salida
`outputOptions`	Muestra el conjunto de opciones para un elemento de salida
`tableOutput`	Crea una tabla como elemento de salida
`textOutput`	Crea un texto como elemento de salida

Tabla II.2: Resumen de las funciones de *outputs* y de los elementos que permiten declarar.

II.2.3.2. SERVER

El código *server.R* se procesa cada vez que un *input* se modifica. En realidad, es la base de Shiny, ya que es el código "habitual" que se ejecutaría para mostrar los resultados en R pero adaptado a los *inputs* y a los *outputs* actualizables, que se muestran en la ventana de la aplicación mediante el archivo de interfaz de usuario.

Este fichero *server*, además de proporcionar los resultados, define la relación entre los *inputs* y los *outputs* a medida que se actualizan.

Reactive, render

Como los valores de entrada pueden cambiar en cualquier momento, los aplicativos de Shiny deben ser interactivos para poder actualizar inmediatamente los valores de salida:

Valores de entrada → Código R → Valores de salida

Las funciones *render* y *reactive* son dos funciones claves durante la ejecución del aplicativo Shiny, ya que permiten actualizar y mostrar cada modificación de *input* que se realice.

Reactive: El paquete de R Shiny incluye una librería de programación reactiva para estructurar la lógica de la aplicación. Este tipo de programación permite que, cuando cambien los valores de

entrada, sólo re-ejecute y actualice el código al que esta entrada hace referencia —sin necesidad de re-ejecutar todo el código restante.

Cada "click" del aplicativo modifica estos valores de entrada (*inputs*): la función *reactive* crea automáticamente un registro de los valores reactivos que se introducen y las expresiones a las qué incumben y que dependen de ellos. Si estos valores se modifican, la función *reactive* entiende que el *output* anterior queda obsoleto y lo actualiza con los resultados obtenidos con el nuevo *input*. Así, mediante esta dependencia de seguimiento, el cambio de un valor incluido en una función reactiva dará instrucciones automáticamente a todas las expresiones que directa o indirectamente dependan de este valor.

Una de las ventajas de *reactive* es que no hay necesidad de programar el código de forma reactiva; basta con incluir el valor que se quiere actualizar automáticamente en una función reactiva para que Shiny se encargue de ejecutar éste y el resto de código que depende de él.

Render: es una función reactiva que muestra los *outputs* en código HTML en la página de la aplicación. Al ser reactiva, R (o, en este caso, R-studio) sólo la ejecuta cuando cambian sus parámetros.

Existen varias maneras de definir una función *render*, según el tipo de salida que la aplicación deba mostrar (tabla, gráfico, etc...).

La Tabla II.3 muestra las diferentes posibilidades.

Función	***Output***
`renderPlot`	Define un gráfico reactivo.
`renderText`	Convierte un vector de caracteres en un solo elemento reactivo.
`renderPrint`	Captura cualquier *output* a mostrar en una cadena de valores reactiva.
`renderDataTable`	Permite mostrar en la interfaz de usuario un *data.frame* o una matriz (ambos reactivos) a través de la biblioteca *DataTables*.
`renderImage`	Permite definir una lista que contenga los atributos (o argumentos) necesarios para mostrar una imagen en la interfaz de usuario.
`renderTable`	Define una tabla reactiva.
`renderUI`	Es una característica experimental. Crea una versión reactiva de una función que genera un código HTML mediante la librería Shiny UI.

Tabla II.3: Resumen de las funciones *render* y los elementos que generan

Todas estas funciones definen, también, un espacio en la interfaz de usuario para mostrar los *output* obtenidos. Las funciones a utilizar se deciden según el tipo de *output* que se quiere y tienen que estar adaptadas a las definiciones de *inputs* y *outputs* vistas en las tablas anteriores.

Normalmente se utilizan directamente los *renders*, sin necesidad de crear una variable concreta reactiva, ya que, al ser funciones reactivas, permiten actualizar los *outputs* de forma directa.

No obstante, en algunos casos se declara una variable reactiva que luego se utiliza dentro de la función *render*.

Se muestra, a continuación, un pequeño ejemplo.

Normalmente los *inputs* se introducen del siguiente modo:

```
output$captionOUT <- renderPrint({
    mean(input$captionIN)
                        })
```

Es decir, se indica el nombre del *output* y del *input* (para que se pueda leer desde el archivo *ui.R*), el tipo de *output* que se quiere y el código R para obtenerlo.

Este caso indica un *output* llamado *captionOUT*, que se mostrará mediante *renderPrint*, y que depende del *input captionIN*, que supondremos que es una secuencia de números. Este ejemplo mostraría la media de los números contenidos en *captionIN*.

El proceso es similar en la función *reactive*, que define la variable que se utilizara como reactiva en las funciones *render*.

Se muestra un ejemplo:

```
datasetInput<-reactive({
    input$captionIN
                })
output$captionOUT <- renderPrint({
    mean(datasetInput())
                        })
```

Se declara una variable reactiva que contendrá el *input captionIN* al que da nombre el *datasetInput*. Esta variable reactiva *datasetInput* se utiliza después en la función *render*, es decir, esta variable (1) se procesa dentro de la función (en este caso solo se calcula la media), (2) se asigna al *output captionOUT* y (3) aparece en la aplicación mediante *renderPrint*.

Al indicar esta variable como reactiva, todas las funciones que la contengan o dependan de ella se re-ejecutaran y mostrarán los nuevos *outputs*. En este caso, cada vez que se modifique el valor de *captionIN*, se re-ejecutará la función *renderPrint* y se recalculará el nuevo valor de *captionOUT*.

Desde que las funciones *render* y *reactive* se unificaron, cada vez se usan menos los términos declarados como *reactive*, ya que la función *render* permite realizar las mismas actualizaciones de *outputs* sin necesidad de declarar ninguna variable como reactiva.

II.3. Aplicativos creados

A continuación expongo los objetivos y las salidas más relevantes de los 5 aplicativos desarrollados.

Los aplicativos están diseñados para analizar puntos concretos del curso sin necesidad de mirar el resto de factores. Están clasificados del siguiente modo: para realizar un análisis global (incluye

alumnos, módulos y ejercicios), el aplicativo 1, de nombre Análisis gráfico por módulos; para seguir los alumnos y el progreso del curso, el 2, llamado Flujo alumnos; para analizar los ejercicios de cada módulo los aplicativos 3 y 4, cuyos nombres son Análisis ejercicios y Análisis Bland Altman, respectivamente; y por último, para el análisis global de cada módulo, el 5 de nombre Análisis de Componentes Principales.

Para el primero, explico el código para generarlo. El punto 0 del Anexo muestra todos estos códigos de R adaptados a Shiny.

II.3.1. Rendimiento de los participantes

El primer aplicativo creado informa sobre el rendimiento de los alumnos, primero, en general (Figura II.2, izquierda)), y luego por ejercicio dentro de cada módulo (Figura II.2, derecha).

Figura II.2. Primer aplicativo. A la izquierda, la primera pestaña muestra la proporción de ejercicios realizados por cada alumno (filas) en cada módulo (columnas). A la derecha, las notas máximas de cada alumno (filas) en cada ejercicio (columnas) distinguiendo por módulo en cada pestaña.

El fichero *server.R* incluye el proceso seguido y su adaptación de Shiny. En primer lugar, inicializa el código *server.R* mediante la función *Shinyserver*() y depura la base de datos.

```
shinyServer(function(input, output) {
```

Crea un *data.frame* de notas máximas de todos los ejercicios sin clasificar por módulo (*notamaxtot*) y una lista con las notas medias de cada alumno en cada módulo (*notamedmod*).

Hasta aquí, no necesita programación reactiva, ya que los cálculos no dependen del *input* que le llega. Pero, a continuación, el *output* sí que necesita actualizarse: en primer lugar, declara que, para mostrar los nuevos *outputs* el proceso debe reiniciarse cada vez que se modifican los *inputs (render)*. Luego indica que el *output* será un gráfico (*renderPlot*) y le asigna el nombre *grafico* al *output* de la función.

```
output$grafico <- renderPlot({
```

A continuación, ordena los alumnos. Para ello, *server.R* recibe un *input* del archivo *ui.R* indicando por qué módulo los quiere ordenar o si se quiere ordenar alfabéticamente, para lo que usa:

```
i<-as.numeric(input$Orden)
```

El *data.frame notamedmodorden* contiene el orden de los alumnos.

Luego, se declaran el resto de *inputs*. El siguiente es *input$checkGroup*, que determina qué módulos se mostrarán. Las líneas inferiores incluyen *input$N*, donde N es el número de alumnos por página; e *input$mas* e *input$menos* que permiten trabajar con los alumnos por páginas y que permiten adelantar o retroceder estas páginas.

```
mx = subset (ejs.x, ejs.x$mod %in% as.numeric(input$checkGroup))
Ninput = input$N
pagina = min(K-1, max(0, input$mas-input$menos))
```

Un nuevo *output* llamado *numpg* indica el número de página que se muestra.

```
output$numpg = renderText({
      1+min(K-1, max(0, input$mas-input$menos))
    })
```

Para dibujar el gráfico del primer módulo seleccionado; (1) define sus parámetros; (2) escoge el módulo entre los pasados por *ui.R* `(input$checkGroup)`; (3) selecciona sus ejecuciones; y (4) dibuja el gráfico (cuyo proceso se explica más adelante). Finalmente, repite para el proceso para el resto de módulos seleccionados.

```
for (j in as.numeric(input$checkGroup)) {
```

Selecciona las notas máximas por alumno de cada ejercicio del módulo y procede a clasificar los ejercicios en aprobados, suspensos o no realizados —categorización que guarda en el nuevo *data.frame MX*.

Selecciona los alumnos de la lista ordenada (*notamedorden*) disponibles en la página seleccionada y los guarda en la variable *listanombres*, que depende del *input input$N*.

```
listanombres<-
rownames(notamedmodorden)[(Ninput*pagina+1):(Ninput*(pagina+1))]
```

Para dibujar el gráfico el gráfico recurre a un bucle (*for* mostrado anteriormente) que recorre los diferentes alumnos incluidos en la página: (1) selecciona el primer alumno de la lista ordenada; (2) busca su *Login* en el *data.frame MX*; (3) calcula su proporción de ejercicios aprobados, suspensos y no realizados; y (4) los pinta de forma proporcional en el recuadro del gráfico pertinente al alumno y al módulo. Los colores son: verde para los aprobados; naranja, suspendidos; y blanco, no realizados.

Si un alumno no aparece en el *data.frame MX* lo salta y Shiny procede a dibujar al siguiente.

Para algunos cálculos en el gráfico es necesario utilizar algunos de los *inputs*, como por ejemplo para calcular la altura de cada recuadro:

```
h = (250-((Ninput-1)*Sy))/Ninput
```

O para definir el límite del número de alumnos en el momento de escribir los nombres en el gráfico, ya que *listanombres* depende del *input Ninput*:

```
text(0, 250-((1:Ninput)-0.5)*(h+Sy), listanombres, cex=1, pos=2 , adj=1)
```

Una vez finalizada la primera iteración, pasa a la siguiente y así sucesivamente hasta dibujar todos los módulos seleccionados.

Hasta aquí llega la función *renderPlot*, que en el código finaliza añadiendo "*})*".

Antes de finalizar el archivo *server.R*, añade una tabla por módulo con las notas máximas de cada alumno por ejercicio. Para mostrarlas: (1) crea con ellas la lista *notamaxprob1*; (2) crea 5 funciones *renderDataTable*, una por módulo, que devuelven los nombres de los alumnos y sus notas máximas de los ejercicios de la lista *notamaxprob1* del módulo; y (3) genera las tablas *tablaX* —siendo X el módulo .

```
output$tabla1 <- renderDataTable(
return(cbind(rownames( notamaxprob1[[1]]), notamaxprob1[[1]]))
                    )
                …
output$tabla5 <- renderDataTable(
return(cbind(rownames( notamaxprob1[[5]]), notamaxprob1[[5]]))
                    )
})
```

La última línea cierra la inicialización del código *server*.

El archivo *ui.R* define la interfaz de usuario de la aplicación. Está dividido en dos partes, para declarar los *inputs* y los *outputs*, con el siguiente proceso: (1) igual que *server.R* inicializa *ui.R* mediante la función *ShinyUI()*; y (2) las funciones *fluidPage* y *fluidRow* dividen al tamaño deseado las filas y columnas del espacio del aplicativo y de los *outputs*. Shiny define filas para asegurar que los elementos de salida aparecerán en la misma línea; y columnas para determinar cuántas sobre el total de 12 ocupará cada elemento. En esta aplicación, el menú ocupa ¼ del espacio disponible y los *outputs* ¾; definidos con el parámetro *column*.

```
shinyUI(fluidPage(
  fluidRow(
    column(3,
             … INPUTS
           )
```

```
        column(9,
                … OUTPUTS
               )
      )
    ))
```

Dentro de los inputs aparecen diferentes funciones cuyos parámetros siguen estructuras similares: (1) nombre del input con el que trabaja el archivo *server.R*; (2) título de la opción que crea la función en el menú de la aplicación; (3) el resto de parámetros varía según la utilidad, las características y visualización de la función.

Las primeras funciones utilizadas son *actionButton;* estas dos funciones son las que trabajan con el número de páginas; definen los *inputs input$mas* e *input$menos* utilizados en el código *server.R* para definir la página a mostrar. La apariencia que toman en la aplicación son dos flechas que permiten avanzar y retroceder en las páginas. Además, entre las dos fechas aparece como *output* el número de págna que se está mostrando, implementado mediante la función *span*.

```
actionButton("menos", "", icon=icon("arrow-left")),
span(textOutput("numpg", inline=TRUE), style="font-size: 20px;"),
actionButton("mas", "", icon=icon("arrow-right")),
```

Figura II.3: Apariencia de la opción de avanzar y retroceder página en Shiny

La siguiente función es *sliderInput*, que define el *input input$N* en el código *server.R*, para definir el número de alumnos por páginas, con un mínimo de 1, y un máximo de 30 (por página), con un valor por defecto inicial de 15, y se muestra como una barra deslizante llamada Número de alumnos.

```
sliderInput("N", "Número de alumnos", min = 1, max = 30, value = 15),
```

Figura II.4: Apariencia de la opción Número de alumnos en Shiny

La función *selectInput* define el *input$Orden* del código *server.R* para ordenar los alumnos con las opciones *M1*, *M2*, *M3*, *M4*, *M5* y *Alfabético*. El nombre de la opción es Orden. Aparece en el menú como un desplegable a seleccionar una sola opción, teniendo *M1* por defecto:

```
selectInput("Orden", label = "Orden", choices = list("M1" = 1, "M2" =
2,"M3" = 3, "M4" = 4,"M5" = 5, "Alfabético"= 6), selected = 1),
```

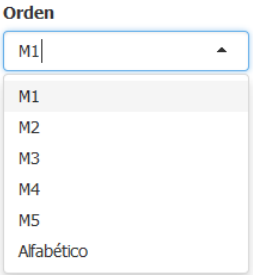


Figura II.5: Apariencia de la opción de selección de Orden mediante desplegable en Shiny

La última función de *inputs* (*checkboxGroupInput*) transmite al *server.R* qué módulos debe pintar. Aparece en el menú como un recuadro con las opciones *M1*, *M2*, *M3*, *M4* y *M5*, y permite seleccionar varias, teniendo *M1* por defecto. El nombre de la opción es Módulo.

```
checkboxGroupInput("checkGroup", label = h3("Módulos"), choices =
list("M1" = 1, "M2" = 2,"M3" = 3, "M4" = 4,"M5" = 5), selected = 1)
```

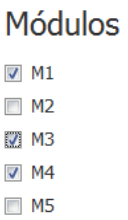


Figura II.6: Apariencia de la opción de selección múltiple Módulo en Shiny

La primera función de *outputs* es *mainPanel*, que indica con qué trabaja el panel principal, es decir, todo lo que mostrará como *output*.

La función *tabsetPanel* crea un *tabset* que divide los *outputs* en secciones independientes, en este caso, en pestañas diferentes. Dentro de cada *tabPanel* aparecen: (1) el título; (2) la función requerida por el *output* a mostrar; y (3) el nombre del *output*; que proviene de las funciones *render* del archivo *server.R*.

En el primer *tabPanel* aparece la función *plotOutput*; y los cincos siguientes, *dataTableOutput*, para mostrar las tablas como *output (tabla1, tabla2, tabla3, tabla4 y tabla5).*

II.3.2. Flujo de los alumnos

El segundo aplicativo muestra el flujo y progreso de los alumnos en cuanto a módulos aprobados y ejercicios realizados.

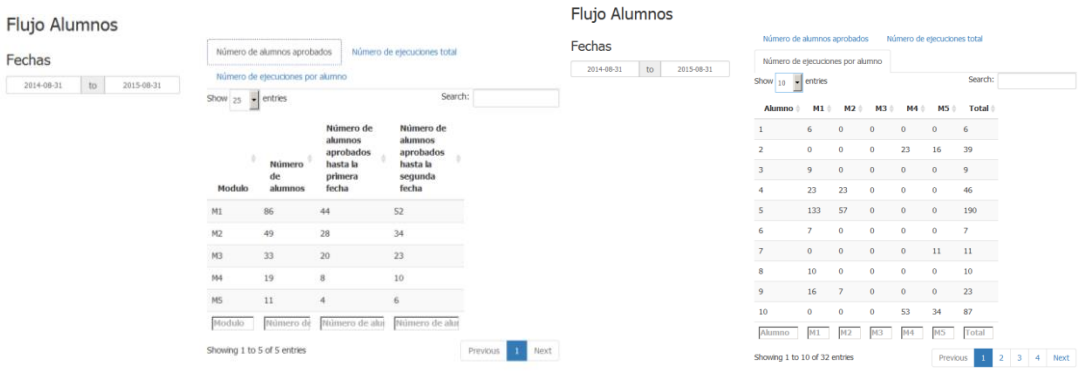


Figura II.7: A la izquierda, alumnos con el módulo aprobado según la fecha; a la derecha ejecuciones de ejercicios realizados por los alumnos según el módulo.

Primero selecciona un rango de fechas y muestra 3 tablas. En la primera (Figura II.7, izquierda), para cada módulo (filas) muestra por columnas: el número que han iniciado el módulo desde que el curso tuvo inicio, el número de aprobados hasta la primera fecha y el número de aprobados hasta la segunda fecha. La segunda tabla(no mostrada): el número total de ejecuciones que debería haber si todos los alumnos hubieran realizados todos los ejercicios; las realizadas hasta la primera fecha en número y porcentaje; y lo mismo hasta la segunda fecha. Por último, la tercera tabla (Figura II.7, derecha) muestra para los alumnos (filas), el número de ejecuciones realizadas en cada módulo (columnas).

II.3.3. Dificultad de los ejercicios.

El tercer aplicativo describe el rendimiento de los participantes en los diferentes ejercicios de cada módulo. La única opción a seleccionar es el módulo deseado. Para cada uno muestra un *forest plot* con el intervalo de confianza de la nota media de cada ejercicio (Figura II.8 izquierda). También presenta una tabla (no mostrada) que incluye para cada ejercicio (filas), su número de ejecuciones, nota media, desviación típica y número de valores NA, ya sea por no haber finalizado la ejecución o por no haber realizado ninguna ejecución del ejercicio hasta el momento. Por último presenta una tercera tabla (Figura II.8: derecha) con los identificadores de los ejercicios en la primera columna y su correspondiente nombre en la segunda columna.

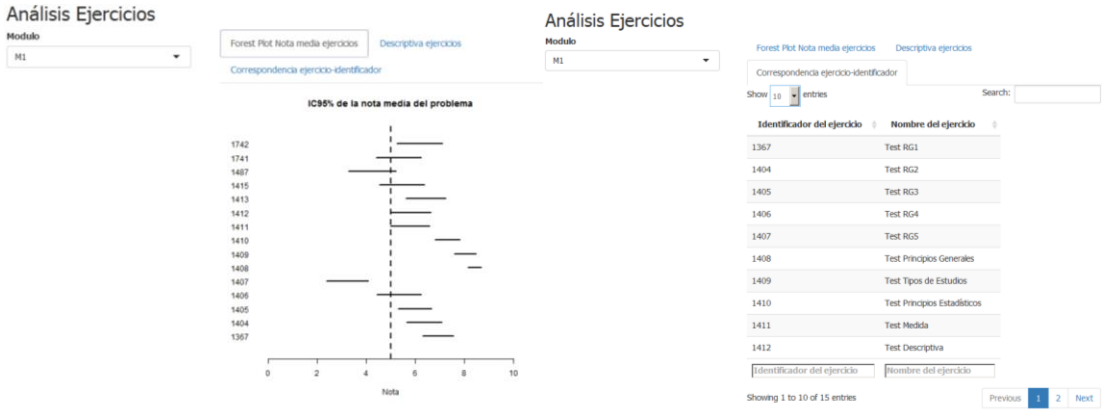


Figura II.8: A la izquierda se muestra la primera pestaña del aplicativo, que permite ver el *forest plot* de las notas medias de los ejercicios del módulo seleccionado, en este caso módulo 1; a la derecha se muestra la tabla que permite relacionar los identificadores con los nombres de los ejercicios del módulo seleccionado, correspondiente a la tercera pestaña del aplicativo.

II.3.4. Bland Altman

El cuarto aplicativo (Figura II.9) (1) permite seleccionar el módulo y el ejercicio a analizar; (2) genera el gráfico con la nota media del módulo en el eje X y la del ejercicio seleccionado en el Y; (3) muestra el gráfico de Bland Altman correspondiente; (4) genera un *forest plot* del $IC_{95\%}$ de las medias de las diferencias de cada ejercicio con el módulo ; y (5) presenta de nuevo la tabla con el identificador de cada ejercicio y su nombre.

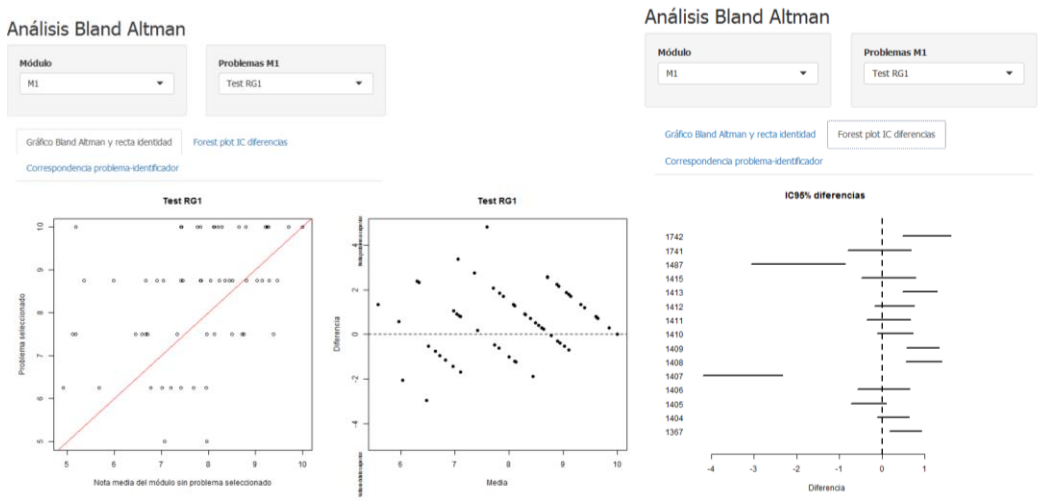


Figura II.9: A la izquierda, permite ver la dificultad de los ejercicios mediante el grafico de Bland Altman; a la derecha, el *forest plot* de las diferencias entre nota del ejercicio y la del módulo.

II.3.5. ACP

El aplicativo ACP realiza el análisis de componentes principales de los ejercicios de un mismo módulo. Las opciones que ofrece son (1) la selección del módulo, (2) la variable a mostrar, dónde hay que elegir entre el gráfico de variables (ejercicios), el gráfico de individuos (alumnos) o ambos, y (3) las dimensiones que se quieren ver (visualiza hasta la 5ª). El aplicativo muestra el ACP de las opciones seleccionadas.

Se muestra su apariencia en la Figura II.10:

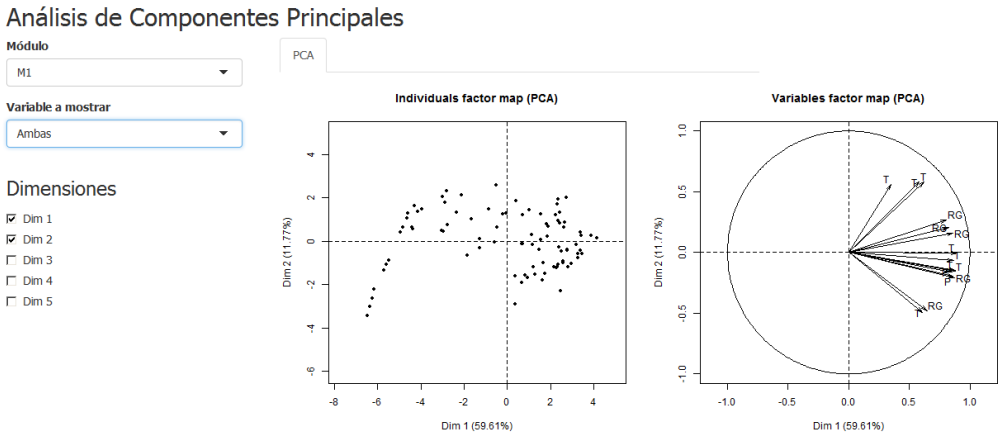


Figura II.10: Mapas factoriales de variables e individuos para el módulo1

II.4. Gráfico de Bland Altman[20]

El gráfico de diferencias de Bland-Altman (propuesto en 1986) permite analizar la concordancia entre dos métodos de medición. Consiste en una parcela de Tukey de "diferencia-media".

Es bien conocido que "correlación no implica concordancia"; por ejemplo, una gran diferencia de medias resultaría en una concordancia baja, pero podría tener aun así una alta correlación. Bland y Altman propusieron su método para resaltar todos los componentes necesarios para una buena concordancia.

Para la construcción de un gráfico de Bland y Altman hay que considerar un conjunto de *n* casos y evaluar los dos métodos de medida en ellos. El gráfico los representa mediante su media en la abscisa (eje x), y su diferencia en la ordenada (eje y).

Las coordenadas cartesianas S(x,y) de muestras con valores de X_1 y X_2 del primer y segundo método, respectivamente, se calculan del siguiente modo:

$$S(x,y) = (\frac{X_1 + X_2}{2} \cdot (X_1 - X_2))$$

El gráfico de Bland y Altman permite investigar la existencia de cualquier diferencia sistemática entre las medidas e identificar posibles valores atípicos. Permite observar si la diferencia de medias tiene sesgo (diferencias de 0); y si la desviación típica (sd) de las diferencias (que mide las fluctuaciones aleatorias en torno a esta media) es constante a lo largo de sus valores medios (que estiman la 'magnitud' subyacente). Si el valor medio de la diferencia difiere significativamente de 0 en un t-test, indicará la presencia de sesgo fijo — que puede ser ajustado restando la diferencia media del nuevo método. Para evitar los problemas originados por la falta de un cálculo de potencia previo y la posible multiplicidad, es común calcular los intervalos de confianza del 95% para cada método de comparación. Si las diferencias dentro del cálculo de $media \pm 1{,}96 \cdot sd$ no son clínicamente importantes, los dos métodos se pueden usar indistintamente. Ésta es una de las ventajas de este gráfico, ya que permite establecer unos límites de concordancia.

Aplicación en el proyecto: Como disponemos de p ejercicios o medidas en lugar de 2, en este caso, se usa el grafico del Bland Altman no para comparar dos métodos de medición, sino comparar la dificultad de un determinado ejercicio perteneciente a un módulo con la dificultad del resto de ejercicios del módulo. Aunque el objetivo del gráfico de no sea el mismo, su construcción sí que lo es.

II.5. Análisis de Componentes Principales

El Análisis de Componentes Principales (ACP) 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 es reducirlas a un menor número perdiendo la menor cantidad de información posible. Se intenta pasar de m a p variables.

Esta técnica fue inicialmente desarrollada por Pearson a finales del siglo XIX y posteriormente por Hotelling en los años 30 del siglo XX, aunque empezó a utilizarse en los años sesenta con la aparición de los ordenadores.

Estas nuevas variables, llamadas "componentes principales (CP)" o "factores" son una combinación lineal de las originales, que son, también, independientes entre sí. Un aspecto clave en ACP es la interpretación de los factores, ya que ésta no viene dada a priori, sino que se deduce exploratoriamente tras observar la relación de los factores con las variables iniciales.

Dado que pretenden retener el máximo de la información contenida, entendida como diferencias originales entre los casos, parten del principio de que lo interesante es retener aquella información compartida por las variables originales. Es decir, el ACP tiene sentido si existen altas correlaciones entre las variables, indicativo de que existe información repetida y, por tanto, pocos factores explican gran parte de la variabilidad total.

ACP define los factores de forma que el primero recoja la mayor proporción posible de la variabilidad original; el segundo la máxima entre la restante, y así sucesivamente. Del total de factores, se eligen aquellos que recojan el porcentaje de variabilidad "suficiente", a los que se denomina CP.

Una vez seleccionados los CP, se representan en una matriz dónde cada elemento representa los coeficientes factoriales de las variables o las correlaciones entre las variables y los componentes principales. Por tanto, la matriz tiene tantas columnas como CP y tantas filas como variables.

Un ACP será más fácilmente interpretable si tienen las siguientes características:

• Los coeficientes factoriales son próximos a 1.

• Una variable tiene coeficientes elevados con pocos factores.

• No hay factores con coeficientes similares.

El punto VI.3.1 del Anexo explica el cálculo y la extracción de CP.

Aplicación en el proyecto: decidimos realizar el ACP ya que permite validar la premisa de que todos los ejercicios contribuyen de forma positiva a una dimensión de tamaño que ordene a los alumnos según su rendimiento; y, además, visualiza si éstos tienen un comportamiento diferencial en los distintos ejercicios a lo largo del seguimiento (dimensión de forma). Es una técnica adecuada cuando se tienen datos multidimensionales, como en este caso.

III. RESULTADOS

Este apartado describe los datos; las salidas de los aplicativos; los BA; y los ACP.

III.1. Descriptiva de los datos

Por brevedad, centraré la descriptiva en los ejercicios del módulo 1.

El primer módulo contiene 15 ejercicios que abordan aspectos diferentes del aprendizaje: 5 contienen preguntas tipo test sobre guías de publicación o *reporting guidelines* (RG); 5 sobre los aspectos teóricos introducidos en los 3 capítulos del módulo; 3 sobre el software R; y 2 problemas numéricos clásicos. Para interpretar estos resultados conviene recordar que los alumnos pueden ejecutar el ejercicio tantas veces como quieran, sabiendo que finalmente les puntuará la nota máxima que saquen. La Tabla III.1 muestra este máximo en los 85 alumnos que han iniciado este módulo.

	Identificador			**Sin imputación de Na's**								**Con imputación de Na's**	
Nombre	**Ejercicio**	**ACP**	**Nº medio de ejs.**	**Min.**	**Q1**	**Mediana**	**Media**	**Q3**	**Máx.**	**Sd**	**Na's**	**Media**	**Sd**
Test RG1	1367	RG1a	1.3	2.5	6.3	8.8	7.9	8.8	10	1.6	11	6.9	3.0
Test RG2	1404	RG1b	1.2	5.0	6.9	7.5	7.9	8.8	10	1.6	17	6.4	3.4
Test RG3	1405	RG1c	1.5	3.8	6.3	7.5	7.4	8.8	10	1.5	17	6.0	3.2
Test RG4	1406	RG3a	0.8	6.3	7.5	8.8	8.7	10	10	1.2	34	5.3	4.4
Test RG5	1407	RG3b	0.7	5.0	6.3	7.5	7.9	10	10	1.9	52	3.2	4.1
Test Principios Generales	1408	T1a	2.3	5.0	7.5	8.8	8.4	10	10	1.4	0	8.4	1.4
Test Tipos de Estudios	1409	T1b	1.7	5.0	7.5	8.8	8.4	9.1	10	1.3	4	8.0	2.2
Test Principios Estadísticos	1410	T1c	2.0	2.5	7.5	7.5	7.8	8.8	10	1.6	6	7.3	2.5
Test Medida	1411	T3a	1.4	3.8	6.3	8.1	8.0	10	10	1.8	24	5.8	3.9
Test Descriptiva	1412	T3b	1.0	5.0	8.0	9.0	8.4	9.0	10	1.4	26	5.8	4.0
Test R1	1413	T2a	1.1	5.0	7.5	8.8	8.6	10	10	1.4	22	6.4	3.9
Test R2	1415	T2b	0.8	5.0	7.5	8.8	8.7	10	10	1.5	33	5.5	4.4
Test R3	1487	T3c	0.5	5.0	8.8	10	9.1	10	10	1.2	47	4.2	4.6
Peak Expiratory Flow	1741	P3a	0.9	5.0	8.0	9.0	8.8	10	10	1.3	35	5.3	4.5
ALT/GPT	1742	P3b	1.3	5.0	8.3	10	9.2	10	10	1.1	29	6.2	4.4

Tabla III.1: Tabla descriptiva de los ejercicios del primer módulo

Como se puede ver en las notas mínimas, sólo 4 casos el ejercicio queda en suspenso. Los valores de los estadísticos mostrados en las siguientes columnas son relativamente altos; por ejemplo,

tanto la media como la mediana tienen valores superiores a 7.5; y la columna del tercer cuartil es siempre superior a 8.5.

Los valores NA's representan el número de ejercicios que, o bien el alumno no ha empezado, o bien ha "dejados a medias", es decir, que ha empezado la ejecución pero no la ha dado por finalizada. A estos ejercicios con valor NA se les asigna un 0 para poder realizar el análisis.

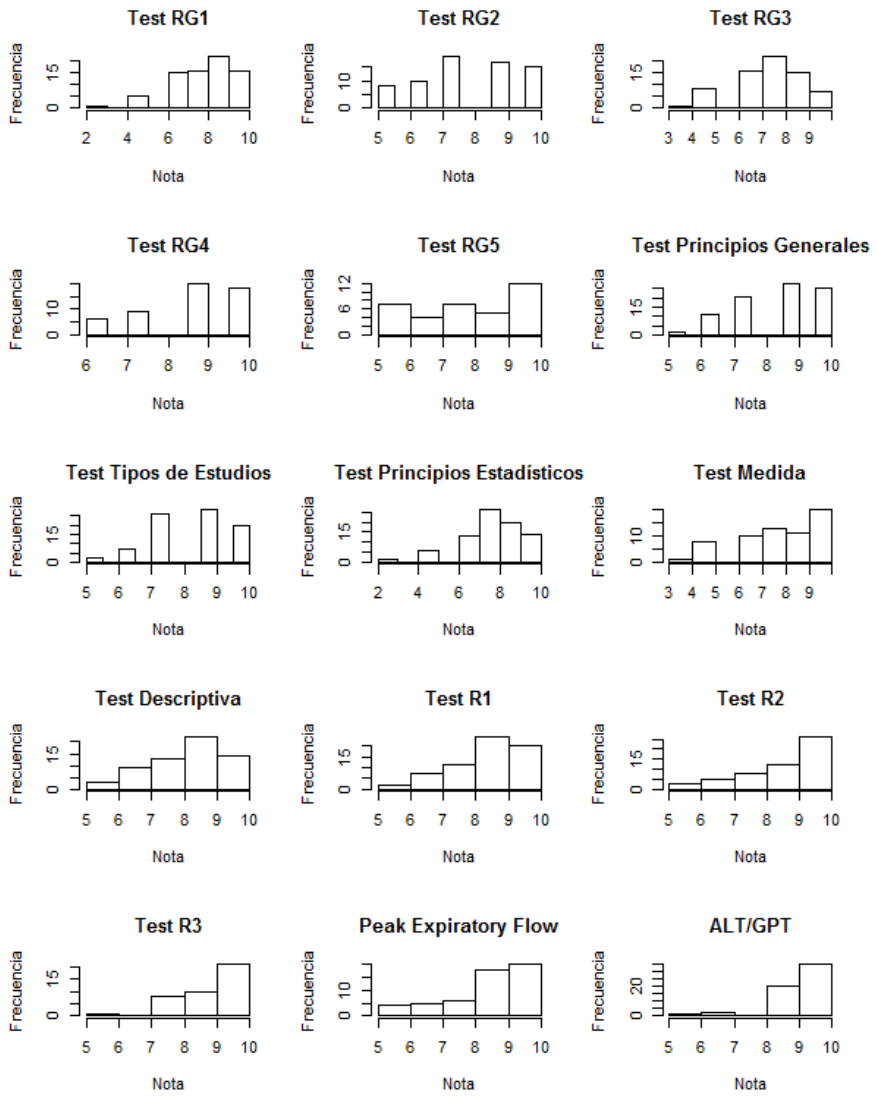


Figura III.1: Histogramas de los 15 ejercicios del módulo 1

La Figura III.1: **Histogramas de los 15 ejercicios del módulo 1**muestra los histogramas de cada ejercicio. Destacan Peak Expitaroy Flow y ALT/GTP como ejercicios en que la mayoría de ejecuciones tienen valores por encima de 8. Se observa también que el Test Descriptiva y el test R1 son los que más se asemejan al comportamiento de una normal. Los únicos ejercicios con

alguna ejecución con nota suspensa son Test RG1, Test RG3, Test principios estadísticos y Test Medida.

La Figura III.2: ***Boxplots* de los 15 ejercicios del módulo 1** muestra los *boxplots* de los 15 ejercicios:

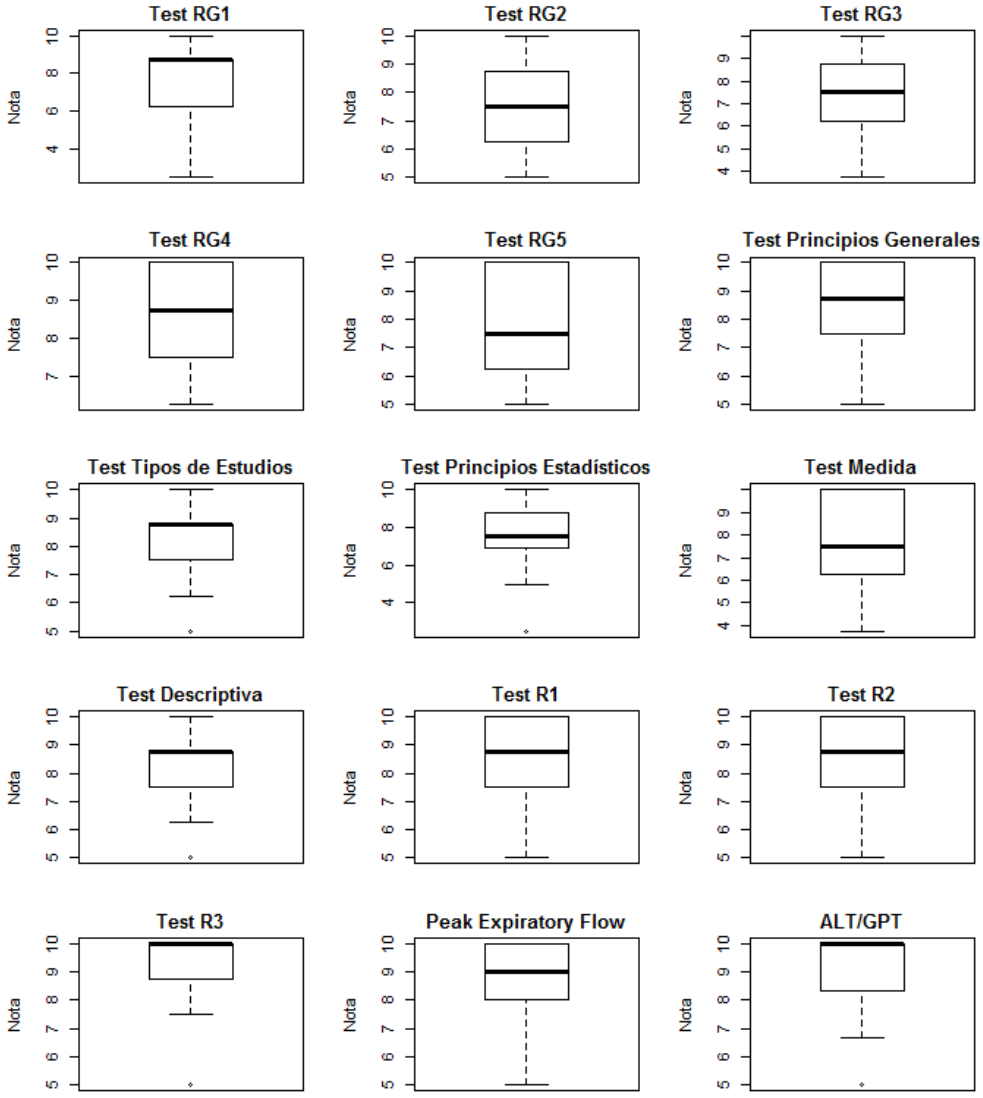


Figura III.2: *Boxplots* de los 15 ejercicios del módulo 1

Destacan los *outliers*, siempre por debajo, de los ejercicios Test Tipo de Estudio, Test Principios Estadísticos, Test Descriptiva, Test R3 y ALT/GTP, posiblemente observados en alumnos que solo hayan hecho una ejecución y tengan una nota baja (rango de 2 a 5).

El punto VI.1 del Anexo contiene las descriptivas, histogramas y *boxplots* de los módulos 2 a 5, cuyos resultados no se interpretan.

III.2. Bland Altman

Explico a continuación cómo interpretar los gráficos de Bland Altman y el *forest plot* del primer aplicativo, mostrando algunos ejemplos particulares.

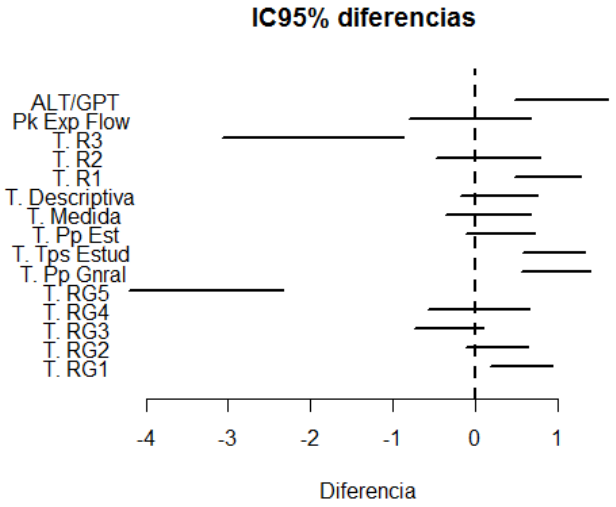


Figura III.3: *Forest plot* del eje y (diferencias) del gráfico de Bland Altman. Etiquetado manual.

El *forest plot* de la Figura III.3: ***Forest plot* del eje y (diferencias) del gráfico de Bland Altman**, permite ver que hay 2 ejercicios más "complicados" que el resto del módulo, bien porque tienen un número elevado de ejecuciones sin terminar o bien porque el número de alumnos que aún no han los ha realizado es elevado (y el aplicativo asigna un 0). Estos son los Test RG5 y Test R3, ejercicios finales del tercer y último capítulo del módulo, que podrían ser "abandonados" por alumnos que se conformaran con el aprobado. La Tabla III.2 muestra que el número de NA's de estos 2 ejercicios es mayor que en el resto de ejercicios.

Nombre	ALT/GTP	Pk Exp Flow	T. R3	T. R2	T. R1	T. Descriptiva	T. Medida	T. pp Est	T. Tps Estud	T. Pp Gnral	T. RG5	T. RG4	T. RG3	T. RG2	T. RG1
Nº NA's	29	35	47	33	22	26	24	6	4	0	52	34	17	17	11

Tabla III.2: Tabla del número de ejecuciones con valor NA de cada uno de los ejercicios del primer módulo

Los ejercicios terminados y realizados por más alumnos tienen un intervalo de confianza más estrecho y superior a 0, como por ejemplo los ejercicios Test principios generales (T. pp Gnral) y Test tipos de estudios (T. Tps Estud), que corresponden a los dos primeros test del capítulo 1.

Usamos el gráfico de Bland Altman para comprobar si esta dificultad de un ejercicio respecto a la del resto del módulo se puede resumir con el análisis de sus medias. De los 67 ejercicios selecciono aquellos con patrones más comunes de comportamiento. La Figura III.4 muestra la distribución de los alumnos de los ejercicios Test RG1 y Test RG2. Aquellos alumnos que están por encima de la línea horizontal que delimita el 0 han obtenido en el ejercicio una nota superior

a la media del módulo; y los que están por debajo, inferior. Los alumnos orientados en la zona inferior izquierda tienen una nota media baja en el módulo y han sacado una nota aún más baja en el ejercicio; y de forma similar, los de la zona superior derecha tienen una nota media alta en el módulo y una nota aún más alta en el ejercicio.

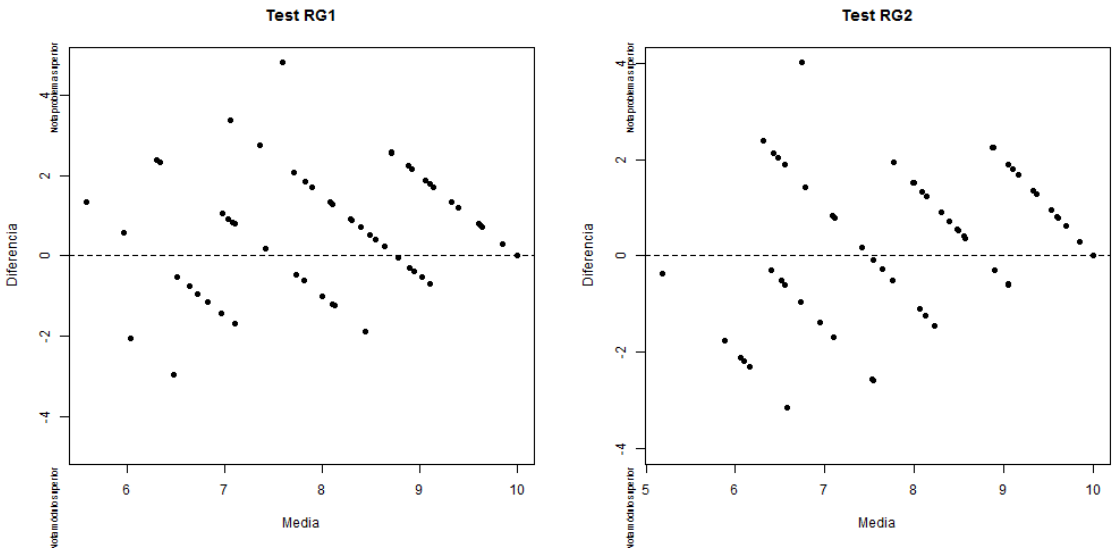


Figura III.4: Gráficos de Bland Altman de los Test RG1 y Test RG2

Los Test RG1 y RG2 son "neutros", en el sentido de que su nota se corresponde con la del módulo. Es decir, los alumnos con una nota del módulo "baja" (entre 5 y 7) obtienen también una nota baja en el ejercicio; y lo mismo para aquellos alumnos con buena nota (de 7 a 10). Además, estos gráficos muestran una dispersión homogénea según la nota media. En resumen, son ejercicios en los que puede darse por buena la clasificación en "fácil" o "difícil" proporcionada por el *forest plot* anterior.

Los ejercicios Capacidad diagnóstica (módulo 2) y Test tipo de estudios (módulo 1) son ejemplos de ejercicios "fáciles". Su número de alumnos difiere debido a que el número de personas que han iniciado los dos módulos es distinto. Estos ejercicios tienen la mayoría de alumnos en la zona superior 0. Es decir, para la mayoría de los alumnos, la nota del ejercicio es mayor que la nota media del módulo.

En el caso del problema Capacidad diagnóstica, es más sencillo saber que el ejercicio es fácil, ya que la mayoría de alumnos está en la zona superior derecha, lo cual indica que tienen nota alta de módulo y, también, nota alta del ejercicio.

En el Test Tipos de Estudios los puntos están distribuidos sobre todo por la zona central superior al 0, lo cual indica que no han obtenido una nota tan alta en el ejercicio, pero sí una nota superior a la media del módulo.

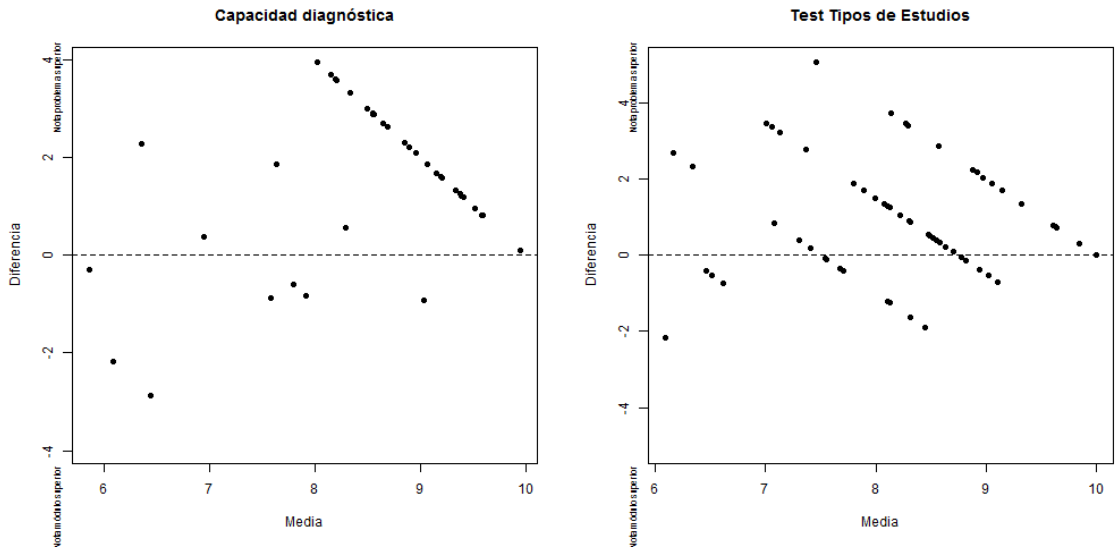


Figura III.5: Gráficos de Bland Altman de los ejercicios Capacidad diagnóstica y Test Tipos de Estudios

Cuando un ejercicio sea difícil, la mayoría de puntos estarán por debajo del 0. Los ejercicios Test RG7 y Test RG8 del módulo 2 presentan la mayoría de sus puntos por debajo de la línea del 0: el alumno ha obtenido una nota en el ejercicio inferior a su nota media en el módulo. El Test RG7 parece que tiene pocos alumnos que no hayan finalizado la ejecución del ejercicio o que no hayan realizado aún el ejercicio y tiene, también, más alumnos por encima de la recta que delimita el 0. En cambio el ejercicio Test RG8 tiene más alumnos que no han finalizado la ejecución o que aún no han realizado el ejercicio y un número menor de alumnos que estén por encima del 0, por lo que dentro del nivel de dificultad, se podría decir que el Test RG8 es más complicado que el Test RG7.

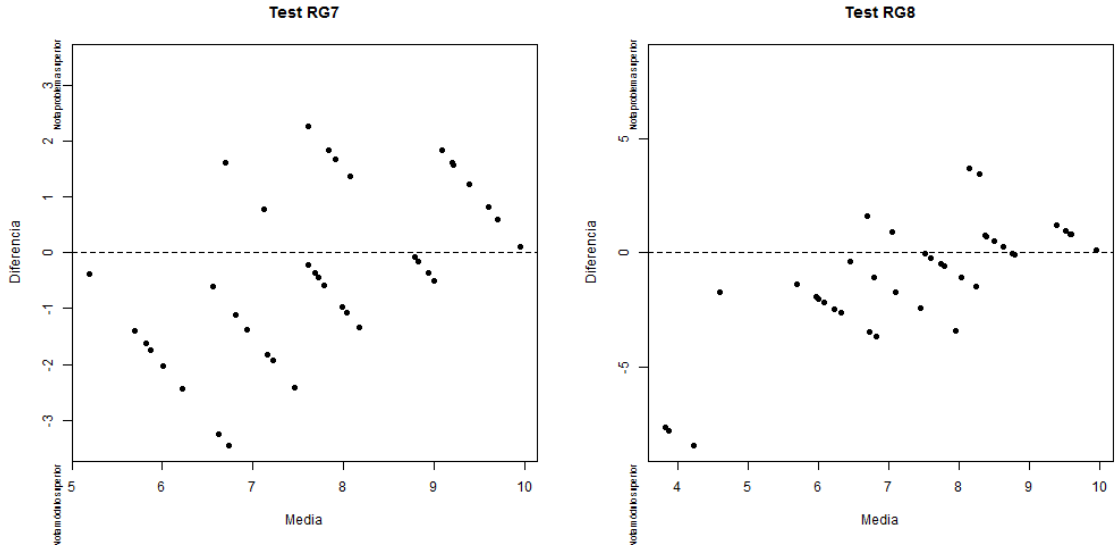


Figura III.6: Gráficos de Bland Altman de los Test RG7 y Test RG8

Los siguientes gráficos de Bland Altman a comentar son el Test RG5 y el Test R3, con un patrón de comportamiento bastante común en esta base de datos. Las líneas de puntos de la zona inferior izquierda indican que muchos de los alumnos que han realizado estos ejercicios no han finalizado la ejecución o bien, que ni si quiera han ejecutado aún el ejercicio y se les ha asignado como nota un 0. El Test RG5 parece que es un ejercicio difícil, ya que la mayoría de alumnos (sin tener en cuenta los que componen la línea de puntos de la zona inferior izquierda), están situados por debajo de la recta que delimita el 0, es decir, en el ejercicio han obtenido una nota inferior a la media del módulo. En cambio, el Test R3 parece un ejercicio fácil, ya que el comportamiento de los puntos es el contrario, es decir, la mayoría se sitúan por encima de la recta que delimita el 0. Tiene sentido que sean ejercicios con menos ejecuciones ya que son los últimos ejercicios de las categorías Test de software R y Test de RG.

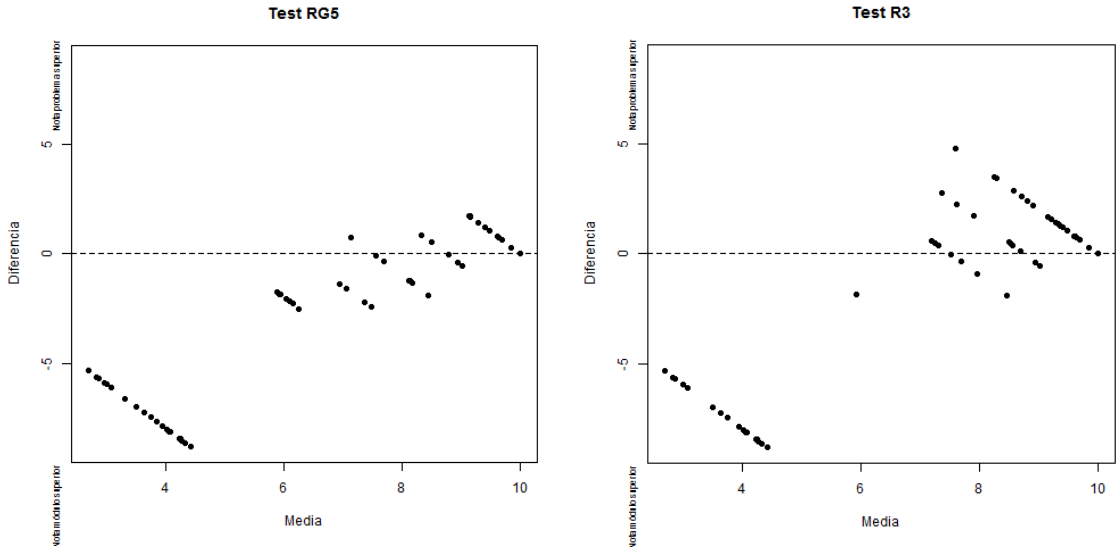


Figura III.7: Gráficos de Bland Altman de los Test RG5 y Test R3

El resto de ejercicios de los diferentes módulos aparecen en el punto 0 del Anexo. Conforme se avanza en los módulos el número de puntos que aparecen es menor debido a que 85 alumnos se han inscrito en el curso pero, hasta ahora, pocos lo han acabado.

III.3. Análisis de componentes principales

El quinto aplicativo realiza el ACP. En el ejemplo que muestro, dispongo como variables de los 15 ejercicios del módulo 1, que intento reducir a un número menor de dimensiones (variables) que expliquen la máxima variabilidad posible.

El criterio de nomenclatura es: T a los test teóricos; RG, a los test de guías de publicación; y P a los ejercicios prácticos. Como dentro de cada capítulo puede haber varios ejercicios de cada una de estas categorías, la etiqueta contiene: (1) tipo de ejercicio; (2) capítulo al que pertenece; y (3)

una letra para distinguir diferentes ejercicios de una misma categoría dentro de cada capítulo. Como temporalmente la base de datos no diferencia a qué capitulo corresponde cada ejercicio, he realizado de modo manual este etiquetado.

La función ACP del paquete *FactoMineR* de R proporciona el resultado de la Figura III.8:

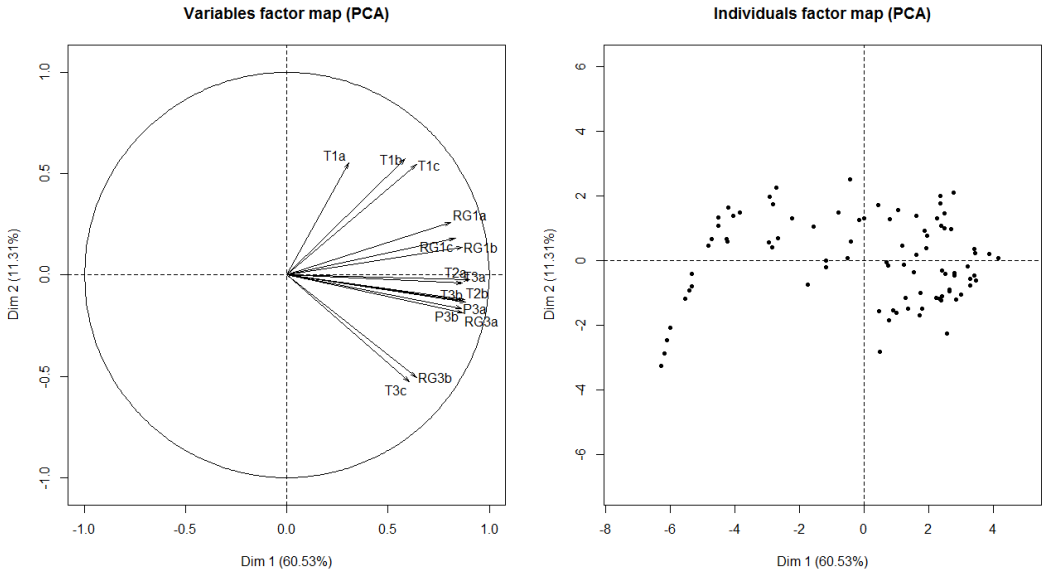


Figura III.8: Mapa de componentes principales para variables e individuos de las dimensiones 1 y 2

La primera dimensión explica el 60.53% de la variabilidad total de los datos. El mapa factorial de variables agrupa todos los ejercicios del módulo en el lado derecho de esta primera dimensión. A este fenómeno se le denomina factor "tamaño", ya que indica que aquellas "unidades más grandes" (mejores alumnos) tienen valores mayores en todas las variables. Luego, esta dimensión permitiría ordenar a los alumnos por su rendimiento. Por tanto, al tener todos los ejercicios una relación positiva con ella, apoya emplear a todas las variables en obtener la nota global del alumno en el módulo. Así pues, afirmamos que los ejercicios del módulo siguen un mismo comportamiento. Queda por valorar si en el futuro conviene asignar diferentes pesos a cada ejercicio.

Bajo este comportamiento, en el mapa factorial de individuos, en el lado derecho están ubicados los alumnos que tienen una nota media del módulo 1 alta, y en el lado izquierdo los alumnos que tienen una nota media del módulo 1 inferior.

La segunda dimensión, que explica un 11.31% de la variabilidad total, separa los ejercicio del capítulo 1 y del capítulo 3, este fenómeno es denominado factor "forma". En la zona superior aparecen los ejercicios del capítulo 1 y, en la zona inferior, los ejercicios del capítulo 3. Luego, bajo el mapa factorial de individuos, en la zona superior están situados aquellos alumnos que obtienen notas más altas en el capítulo 1 que en el capítulo 3, y en la zona inferior, aquellos alumnos que obtienen notas más altas en el capítulo 3 que en el capítulo 1.

El punto VI.1.1 del Anexo, muestra la descriptiva bivariante de los ejercicios del módulo 1, que en este caso, es la matriz de correlaciones entre éstos. Para la mayoría de los ejercicios, los coeficientes de correlación se corresponden con las agrupaciones que se muestran en el mapa factorial de variables.

Para analizar con más detalles los mapas factoriales, seleccionamos los 2 individuos extremos de cada uno de los cuatro cuadrantes, de los cuales se obtienen los siguientes datos:

	Alumno	**Formación**	**Especialidad**	**Afiliación**	**Media Capítulo 1**	**Media Capítulo 2**	**Media Capítulo 3**
Cuadrante izquierdo superior	1	Medicina	Nefrología	Parc Taulí	7.9	0	0
	2	Medicina	- - -	Hosp.Clínico	7.5	0	0
Cuadrante izquierdo inferior	3	Medicina	Endocrinología	Hosp.Clínico	1.5	0	0
	4	Medicina	Angiología y cirugía vascular	Patronat Baleares	1.25	0	0
Cuadrante derecho superior	5	Farmacia	Secretaria CEIC	HUG	10	9.4	6.4
	6	Biología- Química	Coordinadora de ensayos	HUF	9.4	10	7.5
Cuadrante derecho inferior	7	Enfermería	Salud mental	Red de la Salud	7.3	8.8	9.8
	8	Fisioterapia	- - -	Unidad de Murcia	5.6	6.9	7.7

Tabla III.3: Tabla de las características de los alumnos correspondientes a punto extremos en el mapa factorial de individuos

Después de analizar la tabla concluimos que, el cuadrante izquierdo superior ubica alumnos con buena nota media en el capítulo 1, pero que no han avanzado más al finalizar éste, es decir, no tienen ninguna ejecución del capítulo 2 ni del capítulo 3; en el cuadrante izquierdo inferior, individuos con nota media baja en el capítulo 1, y que no han realizado ninguna ejecución del resto de los capítulos; en el cuadrante derecho superior, alumnos con nota media elevada en el capítulo 1 y una nota media inferior en el 3; y, por último, en el cuadrante derecho superior, alumnos con nota media alta en el capítulo 3 y una nota media inferior en el 1.

A continuación analizamos la dimensión 1 y la 3. En este caso, la dimensión 3 explica un 6.28% de la variabilidad total. Ahora, la dimensión 1 también, concluye un factor tamaño; en cambio, la dimensión 3 no permite concluir un factor forma.

Tanto el gráfico de variables como el de individuos muestran un comportamiento similar al de las dimensiones 1 y 2 exceptuando los ejercicios T3c y RG3b, que antes apuntaban en dirección derecha inferior y ahora apuntan en dirección derecha superior. Estos ejercicios corresponden a al Test RG5 y al Test R3, con identificador 1407 y 1487, respectivamente.

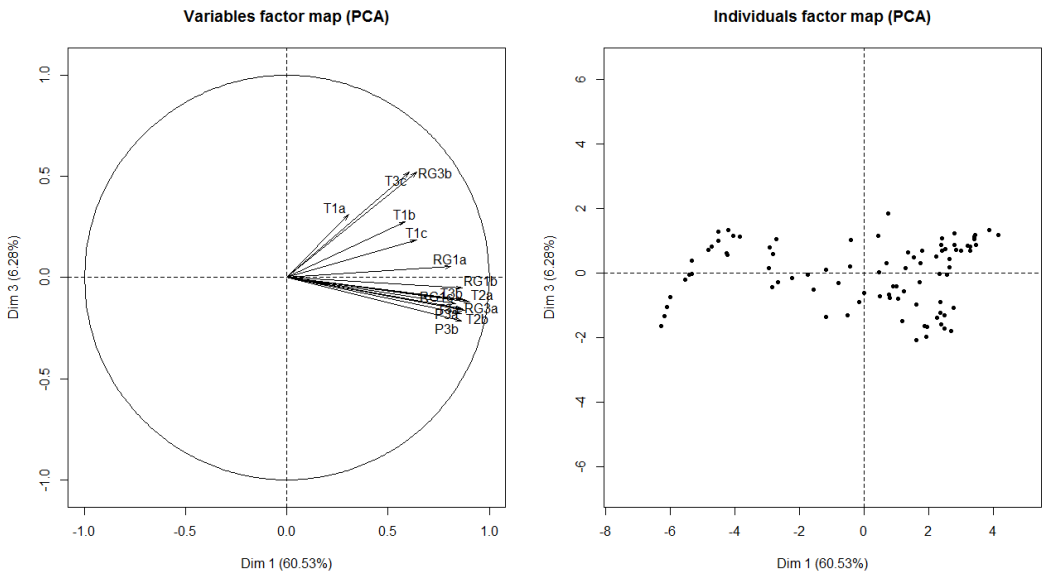


Figura III.9: Mapa de componentes principales para variables e individuos de las dimensiones 1 y 3

Estos dos ejercicios, de comportamiento similar, tienen más ejecuciones sin finalizar o más alumnos que aún no han realizado el ejercicio —ya que son ejercicios finales del capítulo 3.

Por otro lado, en la zona derecha superior aparecen T1a, T1b, T1c y RG1a, que son los ejercicios que menos alumnos han dejado a medias o no han realizado—también con sentido ya que son los primeros ejercicios del capítulo 1.

Luego, concluimos que la dimensión 3 separa las variables (ejercicios) según el número de ejecuciones.

En el mapa factorial, los alumnos siguen la misma distribución que para las dimensiones 1 y 2.

Los mapas factoriales, tanto de variables como de individuos del resto de módulos, aparecen en el punto III.2 del Anexo.

Igual que en los gráficos BA, conforme avanzan los módulos disminuye el número de puntos en el mapa factorial de individuos. Para todos los módulos, la dimensión 1 permite visualizar el factor “forma”.

IV. CONCLUSIONES

Presento 5 aplicativos dinámicos Shiny para realizar el seguimiento curso asíncrono online de "Bioestadística para no estadísticos" mediante indicadores de rendimiento por módulo, ejercicio y alumno. Son sencillos y facilitan la interpretación con gráficos.

A modo de ejemplo, he llevado a cabo el análisis del módulo 1 mediante el análisis de componentes principales, que permite realizar un análisis global del módulo; y los gráficos de Bland Altman, que permiten estudiar los ejercicios.

Los aplicativos apoyan las conjeturas establecidas al inicio del proyecto. Los aplicativos 1 y 2, que los alumnos van progresando; los aplicativos 3 y 4, que los ejercicios tienen una contribución positiva cara a la evaluación de los alumnos; y por último, el aplicativo 5, que los 5 módulos siguen comportamientos similares.

Los aplicativos Shiny permiten resaltar algunas limitaciones, sean del diseño global, o sean del desarrollo Shiny. En primer lugar, los módulos están subdivididos en capítulos, pero en el volcado de datos no existe ningún identificador que permita clasificar cada ejercicio en su correspondiente capitulo; lo que dificulta y hace laborioso el análisis —por ejemplo el ACP requiere asignar los capítulos manualmente.

Otro punto débil actual es no poder acceder al nivel interno de los ejercicios, es decir, a las diferentes preguntas de cada ejercicio. Si esto se pudiera llevar a cabo, el nivel de análisis general sería más profundo.

Un inconveniente del proceso de tratamiento de datos es la no separación de los valores NA's, ya que estos se pueden deber a la no finalización de la ejecución del ejercicio o a que el ejercicio aún no ha sido realizado.

El último inconveniente a destacar es la disminución de alumnos conforme se avanza en los módulos, ya que esto limita los análisis de los últimos capítulos a indicadores muy acotados y poco fiables.

Dadas estas limitaciones, las posibles mejoras futuras son las siguientes: añadir un identificador a cada ejercicio que permita clasificarlo en el capítulo al que pertenece; crear un subnivel dentro de cada ejercicio que permita analizar las preguntas que éste contiene de forma individual; y separar los valores NA's en ejecuciones de ejercicios no finalizadas y en ejercicios que aún no han sido realizados, ya que esto ofrecería la posibilidad de obtener indicadores útiles que permitirían mejorar el análisis de los diferentes ejercicios.

La ventaja de los aplicativos creados es que cada uno de ellos está pensado para analizar diferentes puntos específicos del curso, por lo que si se quiere analizar una cuestión concreta, basta con acudir al aplicativo que permite visualizarla.

En cuanto a las implicaciones prácticas, se recomienda a los directores del curso que realicen regularmente un análisis global del progreso del curso. Se recomienda también, analizar a fondo aquellos ejercicios que muestren un número elevado de valores NA’s debido a ejecuciones sin finalizar, ya que esto puede ser indicador de que el ejercicio puede no estar bien planteado, o que se está dejando poco tiempo para realizar la ejecución.

La transportabilidad de estos aplicativos a otros entornos es desconocida, ya que puede ser complicado encontrar bases de datos planteadas del mismo modo y con las mismas características. Por otro lado, a lo largo de la programación de los códigos de Shiny se ha buscado generar códigos sencillos, de modo que es relativamente fácil adaptar estos códigos a las características de las bases de datos que se quieran analizar.

V. BIBLIOGRAFÍA[1]

(1) http://analisisydecision.es/aprendiendo-shiny-con-vosotros/

(2) http://analisisydecision.es/aprendiendo-shiny-server-r-ui-r/

(3) http://cran.r-project.org/web/packages/shiny/shiny.pdf

(4) http://eprints.ucm.es/10149/1/T30728.pdf

(5) http://fransvandunne.com/2012/12/una-brillaa-shiny-new-approach-to-data-mining/?lang=es

(6) http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/AMult/tema3am.pdf

(7) http://rstudio.github.io/shiny/tutorial/

(8) http://shiny.rstudio.com/articles/basics.html

(9) http://shiny.rstudio.com/help/

(10) http://shiny.rstudio.com/reference/shiny/latest/

(11) http://signalr.net/

(12) http://www.datanalytics.com/2011/02/02/1387/

(13) http://www.fuenterrebollo.com/Economicas/ECONOMETRIA/MULTIVARIANTE/ACP/ACP.pdf

(14) http://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&ved=0CF8QFjAH&url=http%3A%2F%2Fwebmelisa.es%2Fdocs%2Fxornadar%2FShiny.pdf&ei=DitsVdqVEua1sQSizYC4DA&usg=AFQjCNGp1KWQus-guuh-qU1-zwm-78E2ZA&bvm=bv.94455598,d.cWc

(15) http://www.rstudio.com/products/shiny/shiny-user-showcase/

(16) http://rstudio.github.io/shiny-server/latest/#quick-start

(17) http://www.um.es/ae/tShiny/

(18) http://www.um.es/ae/tShiny/taller-shiny-r.pdf

(19) http://www.um.es/ae/tShiny/taller-shiny-r-AMPLIADO.pdf

(20) https://en.wikipedia.org/wiki/Bland%E2%80%93Altman_plot

(21) https://www.fisterra.com/mbe/investiga/conc_numerica/conc_numerica.asp

(22) https://www.mhe.es/universidad/ciencias_matematicas/pena/home/CAPITULO.PDF

(23) https://www.uam.es/personal_pdi/ciencias/abaillo/MatEstI/Tema4.pdf

[1] Se facilitan los links de los recursos utilizados, accesibles todos ellos a día 26/06/15

(24) CONSORT 2010 Explanation and Elaboration: updated guidelines for reporting parallel group randomised trials (David Moher, Sally Hopewell, Kenneth F Schulz, Victor Montori, Peter C Gøtzsche, P J Devereaux, Diana Elbourne, Matthias Egger, Douglas G Altman, February 2010)

(25) Strengthening the Reporting of Observational Studies in Epidemiology (STROBE): Explanation and Elaboration (Jan P Vandenbroucke,1 Erik von Elm, Douglas G Altman, Peter C Gøtzsche, Cynthia D Mulrow, Stuart J Pocock, Charles Poole, James J Schlesselman, Matthias Egger, October 2007)

(26) The PRISMA Statement for Reporting Systematic Reviews and Meta-Analyses of Studies That Evaluate Health Care Interventions: Explanation and Elaboration (Alessandro Liberati, Douglas G. Altman, Jennifer Tetzlaff, Cynthia Mulrow, Peter C. Gøtzsche, John P. A. Ioannidis, Mike Clarke, P. J. Devereaux, Jos Kleijnen, David Moher , June 2009)

(27) Transparent Reporting of a multivariable prediction model for Individual Prognosis Or Diagnosis (TRIPOD): Explanation and Elaboration FREE (Karel G.M. Moons, PhD; Douglas G. Altman, DSc; Johannes B. Reitsma, MD, PhD; John P.A. Ioannidis, MD, DSc; Petra Macaskill, PhD; Ewout W. Steyerberg, PhD; Andrew J. Vickers, PhD; David F. Ransohoff, MD; and Gary S. Collins, PhD)

VI. ANEXO

VI.1. Descriptivas

VI.1.1. Descriptiva bivariante ejercicios módulo1

	RG1a	RG1b	RG1c	RG3a	RG3b	T1a	T1b	T1c	T3a	T3b	T2a	T2b	T3c	P3a	P3b
RG1a	1.000	0.762	0.735	0.594	0.402	0.243	0.623	0.666	0.674	0.689	0.702	0.620	0.409	0.584	0.605
RG1b	0.762	1.000	0.895	0.658	0.460	0.267	0.507	0.582	0.726	0.696	0.812	0.675	0.466	0.670	0.696
RG1c	0.735	0.895	1.000	0.592	0.371	0.213	0.500	0.587	0.709	0.656	0.818	0.653	0.405	0.647	0.648
RG3a	0.594	0.658	0.592	1.000	0.631	0.241	0.396	0.429	0.747	0.825	0.732	0.822	0.485	0.855	0.843
RG3b	0.402	0.460	0.371	0.631	1.000	0.092	0.226	0.245	0.472	0.518	0.490	0.569	0.862	0.542	0.558
T1a	0.243	0.267	0.213	0.241	0.092	1.000	0.424	0.393	0.159	0.171	0.249	0.230	-0.012	0.263	0.182
T1b	0.623	0.507	0.500	0.396	0.226	0.424	1.000	0.676	0.443	0.400	0.463	0.418	0.183	0.399	0.372
T1c	0.666	0.582	0.587	0.429	0.245	0.393	0.676	1.000	0.532	0.495	0.484	0.466	0.200	0.426	0.414
T3a	0.674	0.726	0.709	0.747	0.472	0.159	0.443	0.532	1.000	0.879	0.780	0.723	0.490	0.654	0.761
T3b	0.689	0.696	0.656	0.825	0.518	0.171	0.400	0.495	0.879	1.000	0.791	0.739	0.574	0.741	0.765
T2a	0.702	0.812	0.818	0.732	0.490	0.249	0.463	0.484	0.780	0.791	1.000	0.788	0.544	0.779	0.765
T2b	0.620	0.675	0.653	0.822	0.569	0.230	0.418	0.466	0.723	0.739	0.788	1.000	0.465	0.918	0.820
T3c	0.409	0.466	0.405	0.485	0.862	-0.012	0.183	0.200	0.490	0.574	0.544	0.465	1.000	0.448	0.449
P3a	0.584	0.670	0.647	0.855	0.542	0.263	0.399	0.426	0.654	0.741	0.779	0.918	0.448	1.000	0.800
P3b	0.605	0.696	0.648	0.843	0.558	0.182	0.372	0.414	0.761	0.765	0.765	0.820	0.449	0.800	1.000

VI.1.2. Descriptivas módulo 2

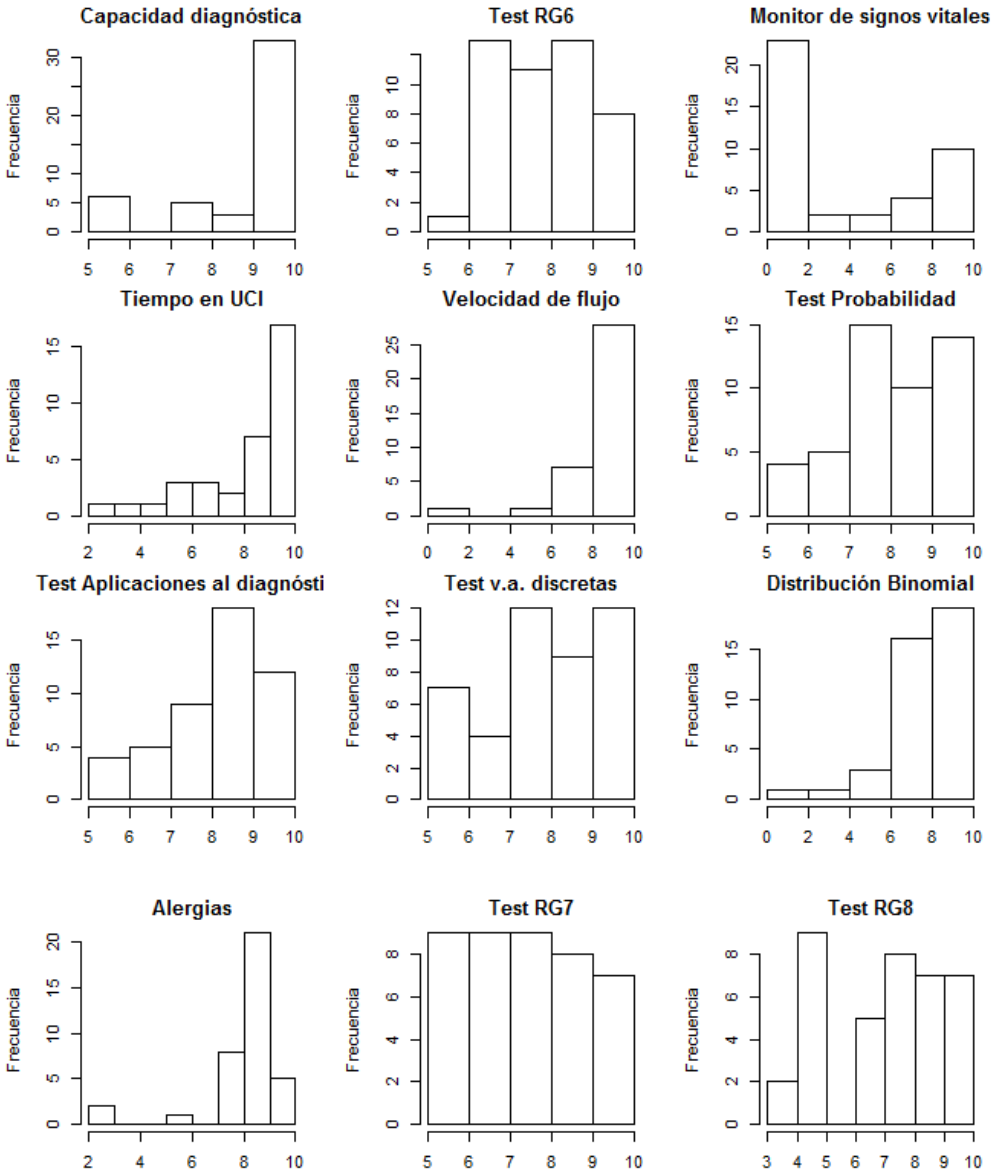
Tabla descriptiva de los ejercicios del módulo 2

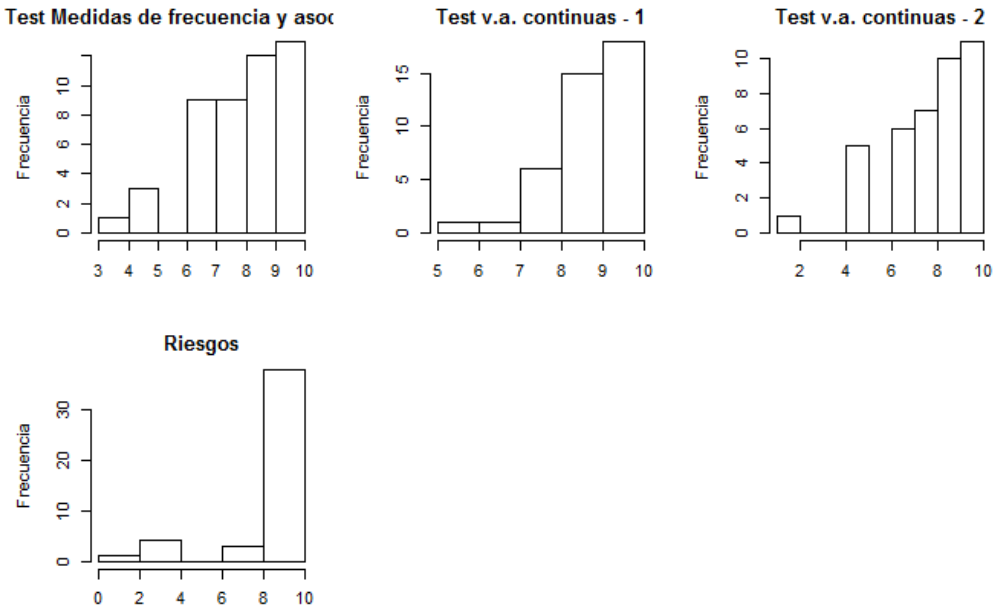
			Sin imputación de Na's								**Con imputación de Na's**	
Nombre	**Ident. Ejercicio**	**Nº medio de ejs.**	**Mín.**	**Q1**	**Mediana**	**Media**	**Q3**	**Máx.**	**Sd**	**Na's**	**Media**	**Sd**
Capacidad diagnóstica	1422	1.2	5.0	8.6	10	9.0	10	10	1.7	1	8.8	2.1
Test RG6	1424	1.8	5.0	6.3	7.5	7.9	8.8	10	1.4	2	7.4	2.4
Monitor de signos vitales	1425	1.5	0.0	0.0	0.0	3.5	7.5	10	4.4	7	3.0	4.2
Tiempo en UCI	1428	1.5	2.2	7.2	8.9	8.4	10	10	2.2	13	6.0	4.3
Velocidad de flujo	1430	0.9	0.0	8.8	10	9.0	10	10	2.0	11	6.8	4.3
Test Probabilidad	1433	2.0	5.0	7.5	8.1	8.2	10	10	1.6	0	8.2	1.6
Test Aplicaciones al diagnóstico	1436	1.9	5.0	7.5	8.8	8.2	9.1	10	1.5	0	8.3	1.5
Test v.a. discretas	1437	1.7	5.0	7.2	7.5	7.9	10	10	1.7	4	7.2	2.8
Distribución Binomial	1439	1.1	0.0	6.9	7.9	7.9	8.9	10	2.0	8	6.5	3.6

Alergias	1440	1.2	2.9	7.1	8.6	8.1	8.6	10	1.6	11	6.1	3.8
Test RG7	1441	1.6	5.0	6.3	7.5	7.4	8.8	10	1.8	6	6.3	3.1
Test RG8	1443	2.1	3.8	5.0	7.5	7.2	8.8	10	2.0	10	5.6	3.5
Test Medidas de frecuencia y asociación	1444	1.7	3.8	6.3	8.8	8.0	10	10	1.7	1	7.9	2.0
Test v.a. continuas - 1	1450	1.1	5.0	8.8	8.8	9.0	10	10	1.2	7	7.5	3.5
Test v.a. continuas - 2	1457	1.9	1.3	6.3	8.8	7.8	10	10	2.0	8	6.4	3.6
Riesgos	1675	2.3	0.0	8.8	10	8.8	10	10	2.4	2	8.3	3.2

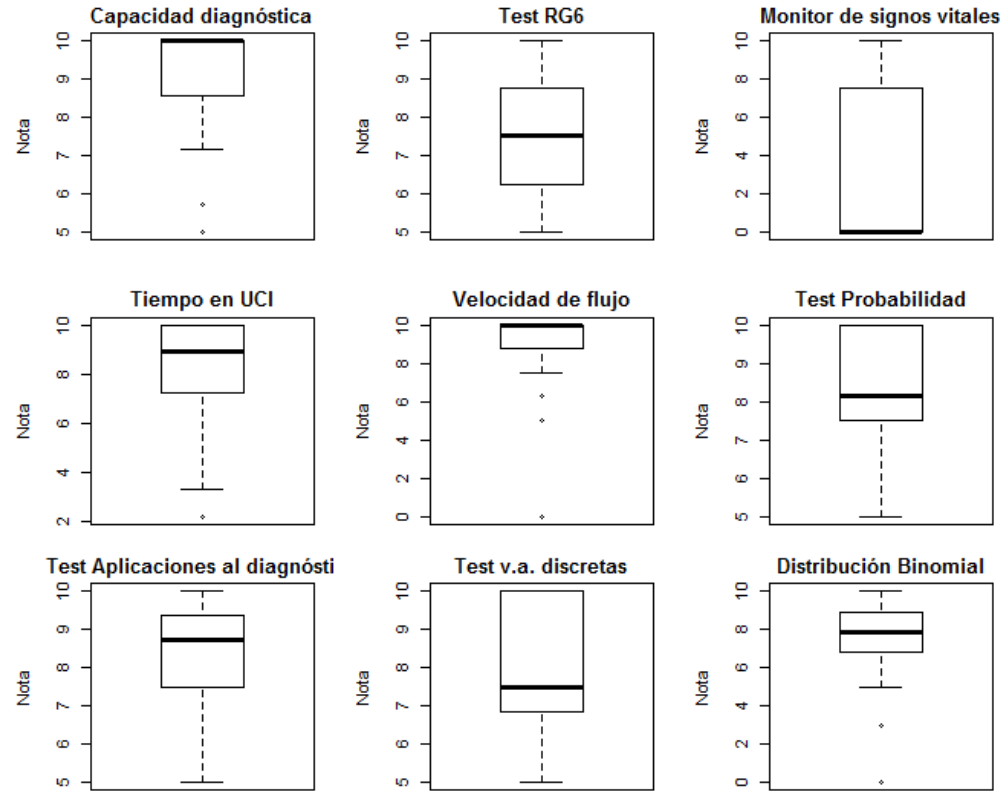
Número de alumnos que han iniciado el módulo = 48

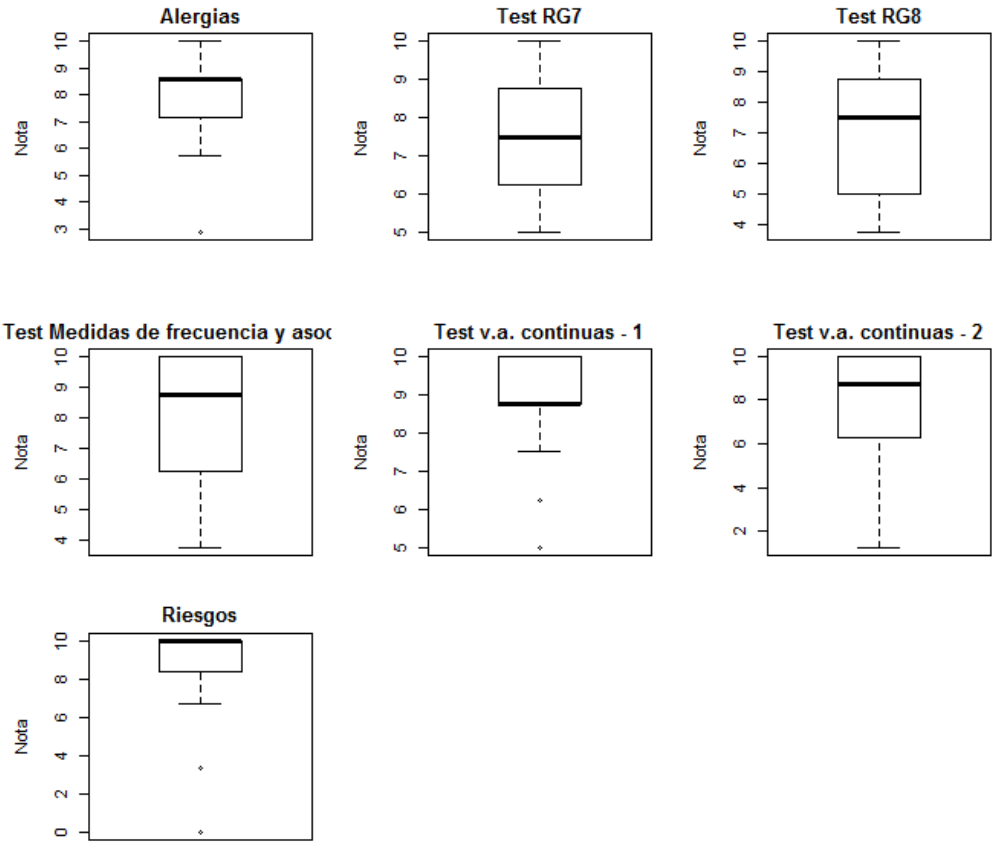
Histograma de los ejercicios del módulo 2





Boxplots de los ejercicios del módulo 2





VI.1.3. Descriptivas módulo 3

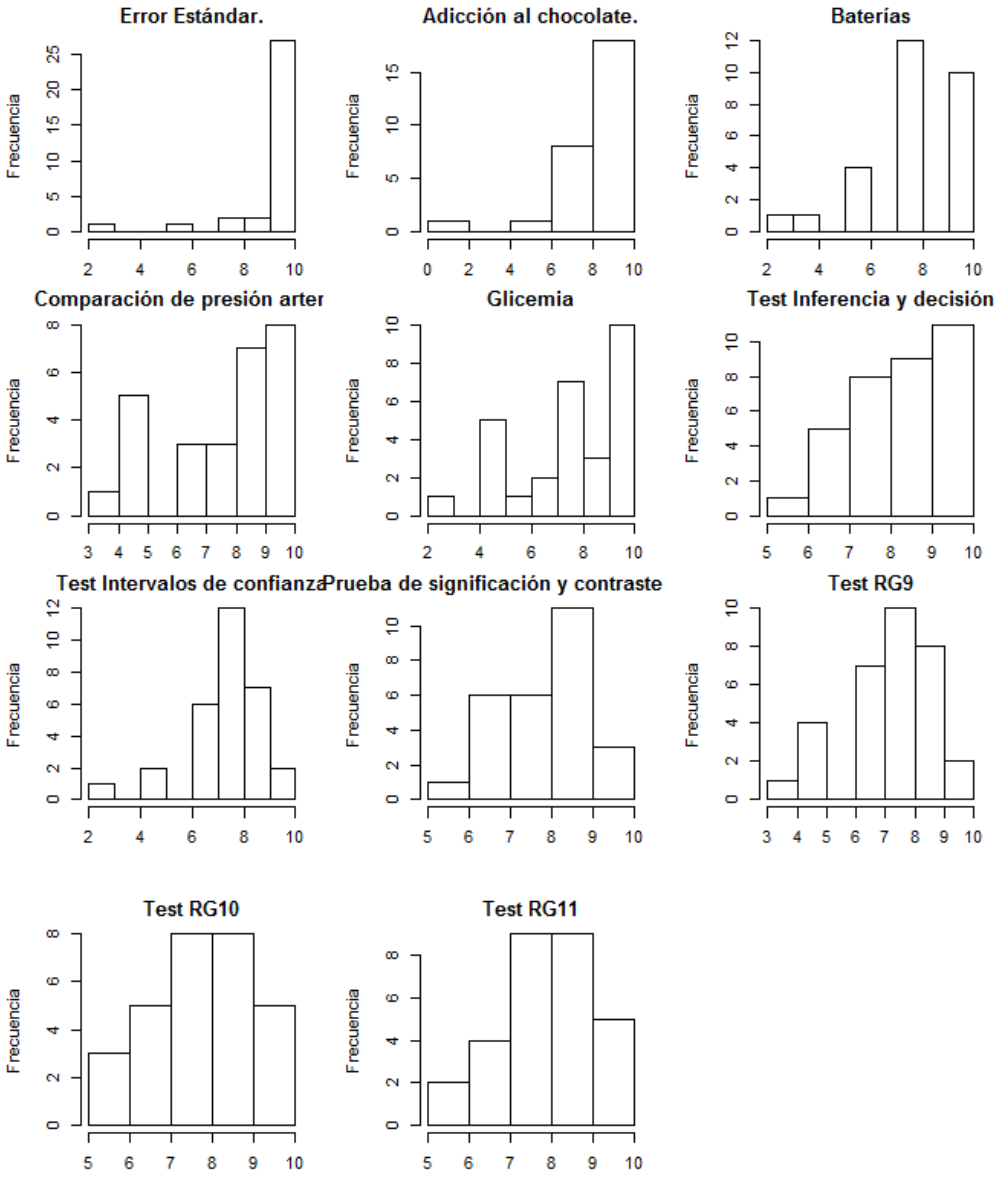
Tabla descriptiva de los ejercicios del módulo 3

			Sin imputación de Na's								Con imputación de Na's	
Nombre	**Ident. Ejercicio**	**Nº medio de eje.**	**Min.**	**Q1**	**Mediana**	**Media**	**Q3**	**Máx.**	**Sd**	**Na's**	**Media**	**Sd**
Error Estándar	1465	1.4	3.0	10	10	9.4	10	10	1.6	1	9.1	2.2
Adicción al chocolate	1466	1.1	0.0	8.0	9.5	8.6	10	10	2.2	6	7.0	3.9
Baterías	1467	1.7	2.0	7.0	8.0	8.0	10	10	2.0	6	6.6	3.6
Comparación de presión arterial	1468	1.2	4.0	6.0	9.0	7.9	10	10	2.0	7	6.5	3.6
Glicemia	1472	2.2	2.0	6.0	8.0	7.9	10	10	2.2	5	6.6	3.4
Test Inferencia y decisión	1476	1.8	5.0	8.0	9.0	8.5	10	10	1.5	0	8.4	1.5
Test Intervalos de confianza	1477	1.8	2.0	6.0	8.0	7.6	9.0	10	1.6	4	6.5	2.8
Test Prueba de significación y contraste	1478	1.5	5.0	7.0	9.0	8.8	9.0	10	1.4	7	6.6	3.3

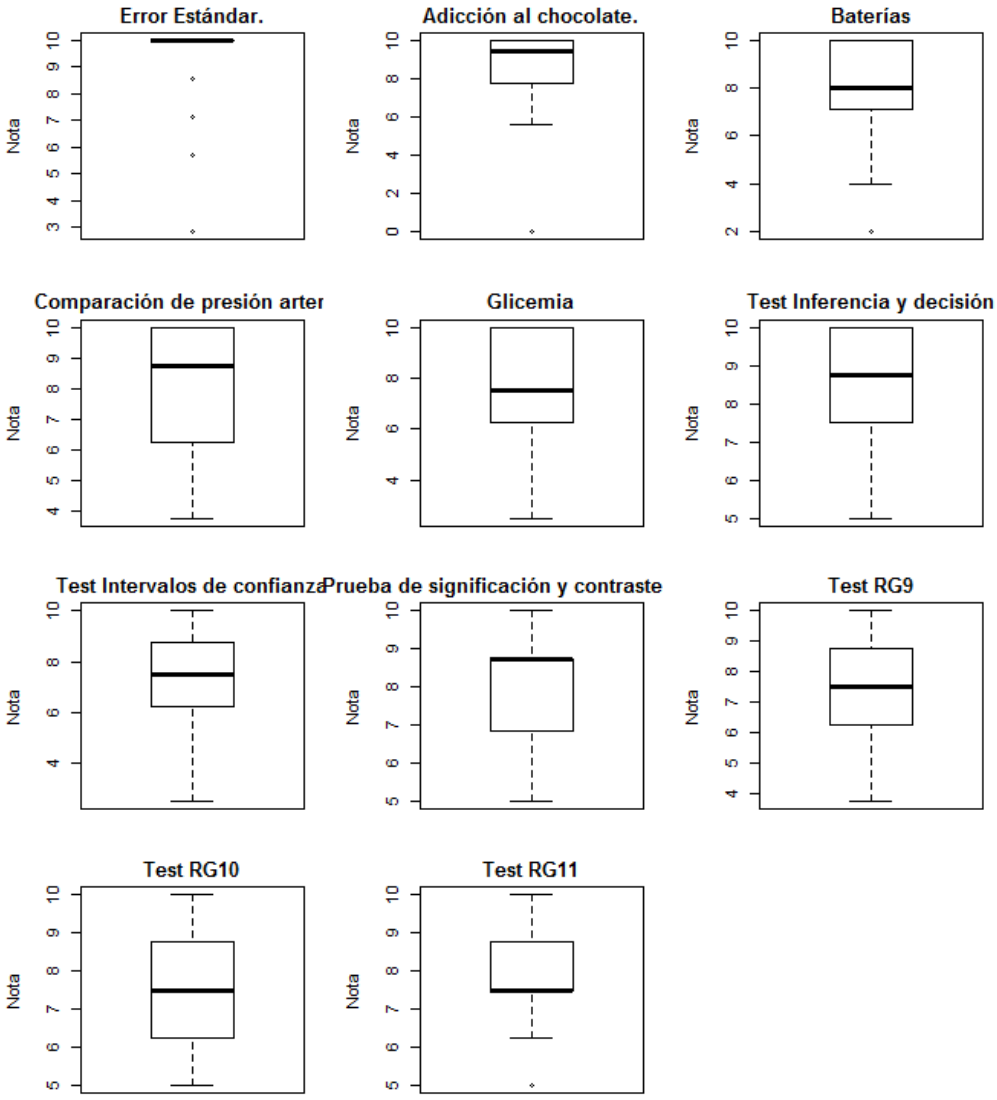
de hipótesis												
Test RG9	1479	1.8	4.0	6.0	8.0	7.4	9.0	10	1.5	2	6.8	2.3
Test RG10	1483	1.5	5.0	6.0	8.0	8.0	9.0	10	1.6	5	6.7	3.2
Test RG11	1484	1.1	5.0	800	8.0	8.2	9.0	10	1.4	5	6.8	3.2

Número de alumnos que han iniciado el módulo = 34

Histograma de los ejercicios del módulo 3



Boxplots de los ejercicios del módulo 3



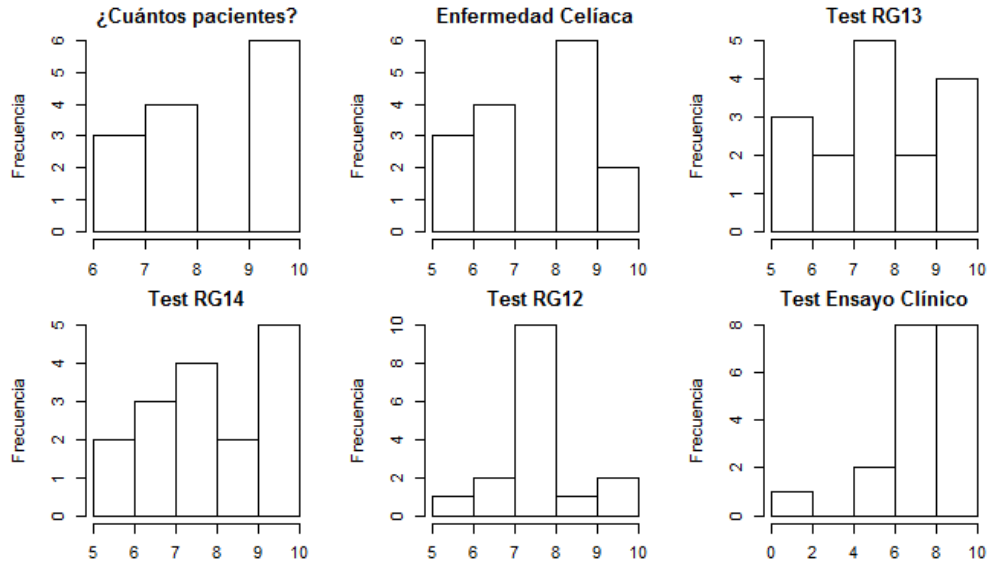
VI.1.4. Descriptivas módulo 4

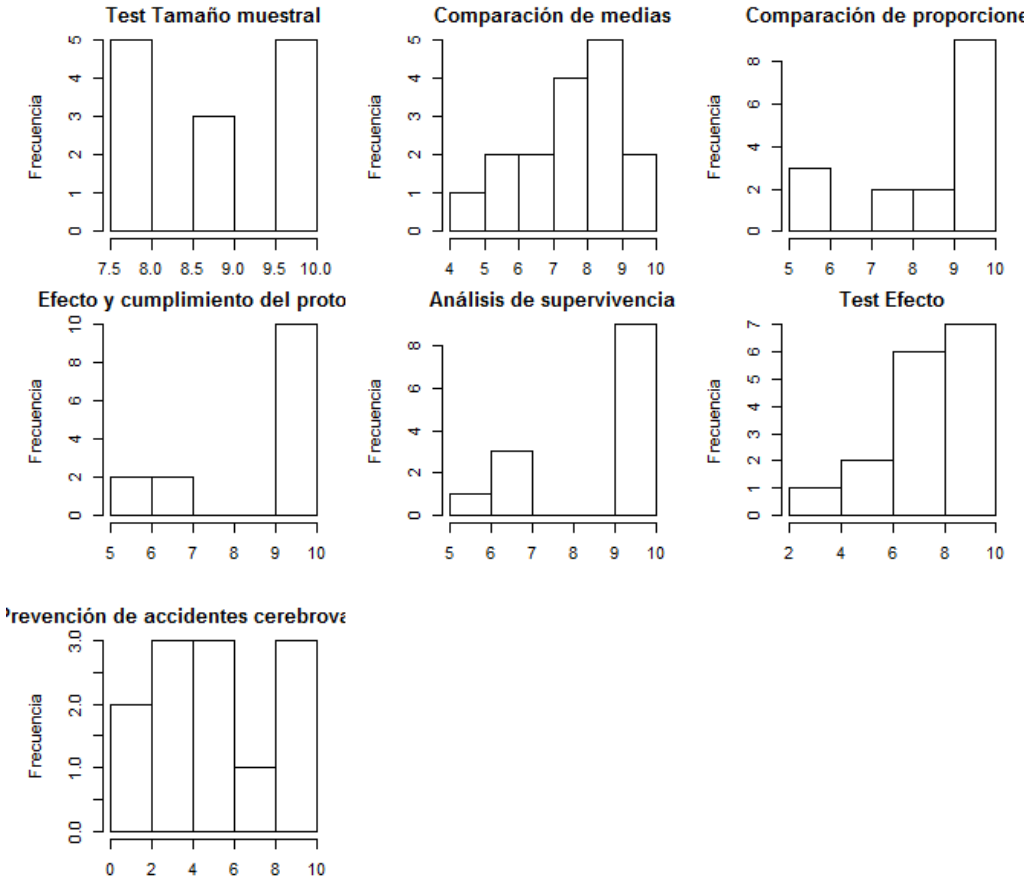
Tabla descriptiva de los ejercicios del módulo 4

			Sin imputación de Na's								Con imputación de Na's	
Nombre	**Ident. Ejercicio**	**Nº medio de ejs.**	**Min.**	**Q1**	**Mediana**	**Media**	**Q3**	**Máx.**	**Sd**	**Na's**	**Media**	**Sd**
¿Cuántos pacientes?	1490	1.9	6.0	8.0	8.0	8.5	10	10	1.6	6	5.9	4.2
Enfermedad Celíaca	1493	1.6	5.0	6.7	8.3	7.4	8.3	10	1.7	4	6.1	3.5
Test RG13	1655	2.5	5.0	6.3	7.5	7.7	9.1	10	1.8	3	6.4	3.2
Test RG14	1656	2.5	5.0	6.3	7.5	7.9	10	10	1.9	3	6.6	3.3
Test RG12	1657	3.2	5.0	7.5	7.5	7.6	7.5	10	1.3	3	6.8	2.6
Test Ensayo Clínico	1658	4.1	1.3	6.3	7.5	7.4	8.8	10	1.5		7.7	1.5
Test Tamaño muestral	1659	1.0	7.5	7.5	8.8	8.8	10	10	1.3	6	6	4.2
Comparación de medias	1660	1.6	4.4	6.7	7.8	7.8	8.9	10	1.6	3	6.9	2.8
Comparación de proporciones	1661	1.4	5.7	7.1	10	8.7	10	10	1.8	3	7.9	3.2
Efecto y cumplimiento del protocolo	1667	1.6	5.0	7.5	10	8.8	10	10	2.0	5	6.7	4.3
Análisis de supervivencia	1668	1.2	5.0	6.7	10	8.9	10	10	1.8	6	6.2	4.5
Test Efecto	1669	2.0	2.5	6.3	7.5	7.7	10	10	2.2	3	6.5	3.4
Prevención de accidentes cerebrovasculares	1738	2.2	0.0	2.5	7.8	5.0	8.2	10	3.7	7	3.5	3.9

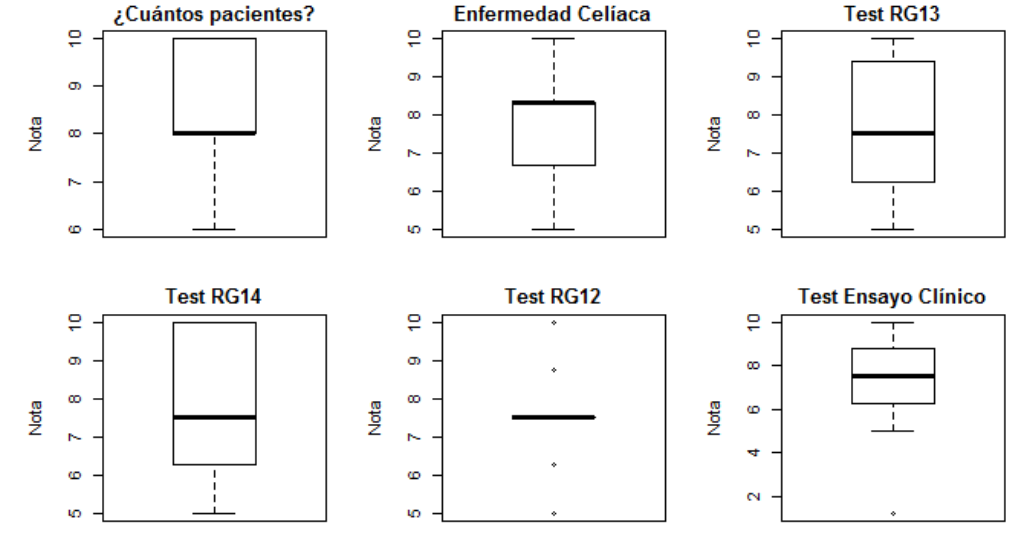
Número de alumnos que han iniciado el módulo = 19

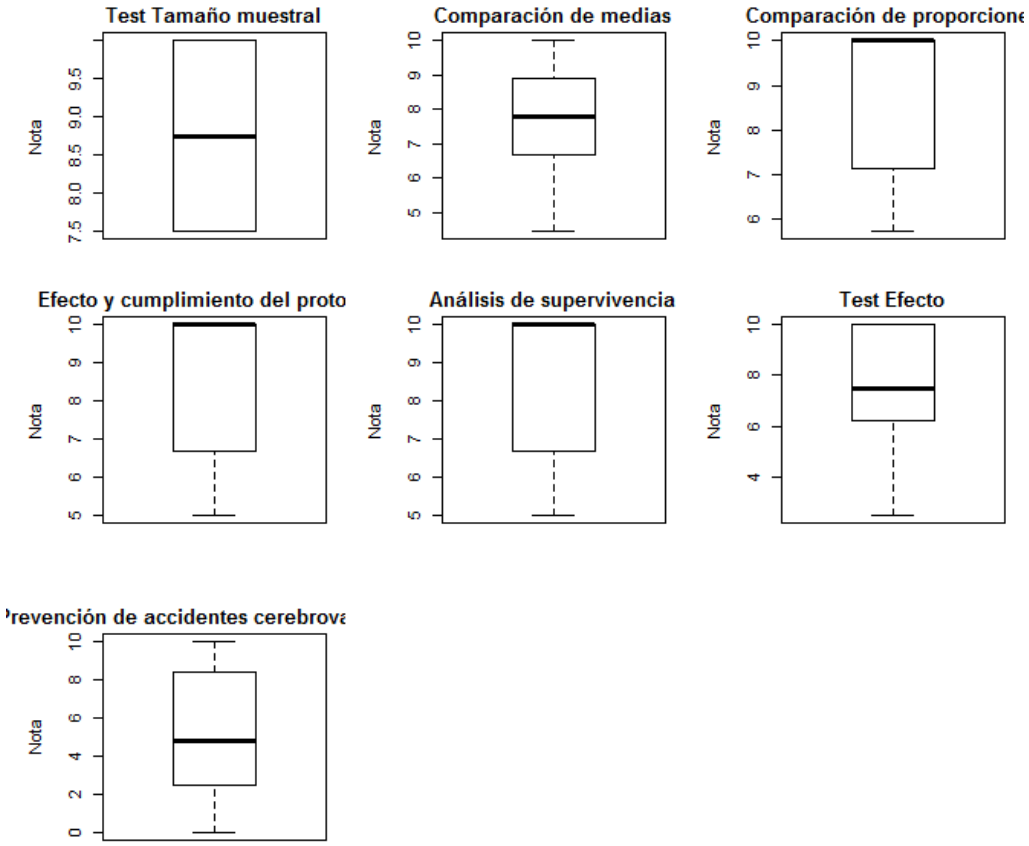
Histograma de los ejercicios del módulo 4





Boxplots de los ejercicios del módulo 4





VI.1.5. Descriptivas módulo 5

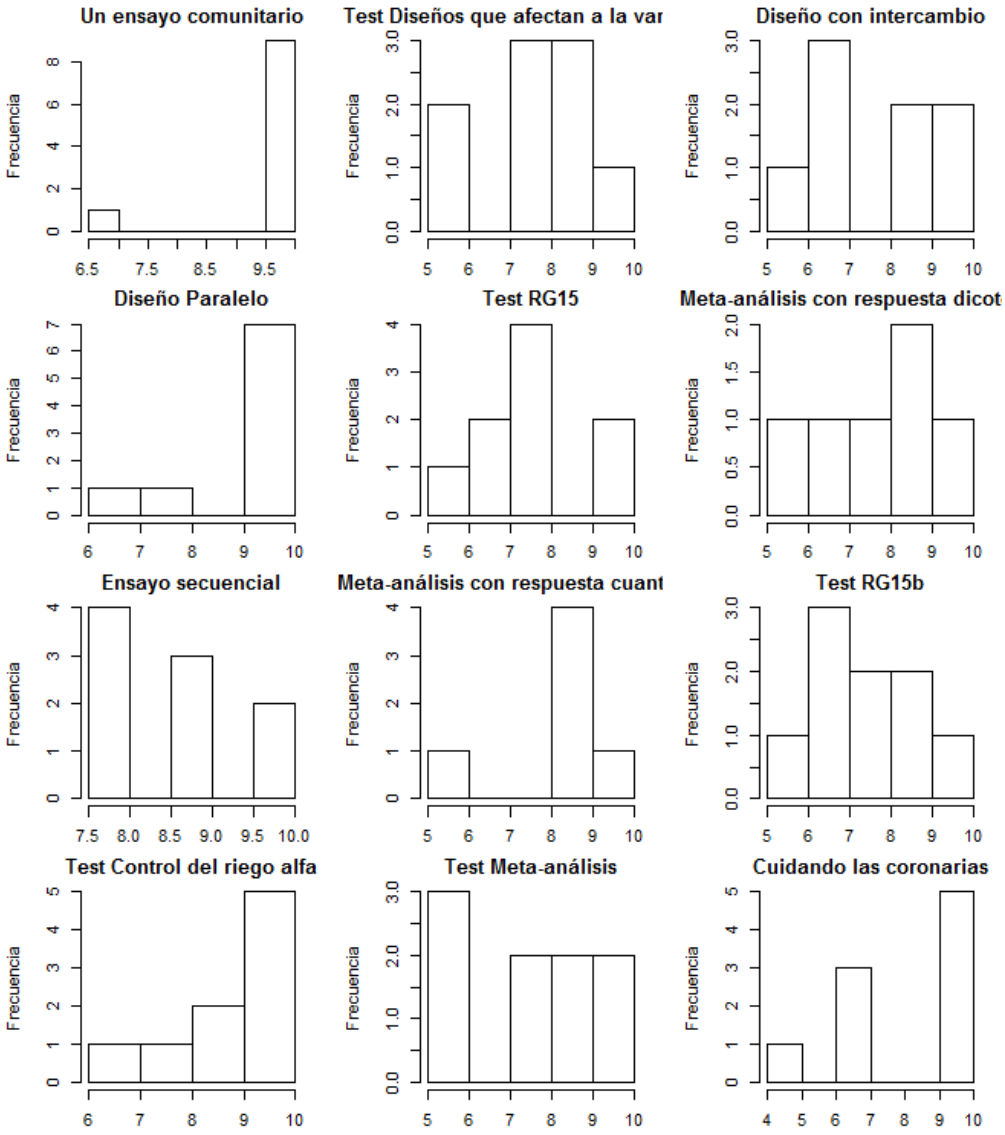
Tabla descriptiva de los ejercicios del módulo 5

			Sin imputación de Na's								Con imputación de Na's	
Nombre	**Ident. Ejercicio**	**Nº medio de eis.**	**Min.**	**Q1**	**Mediana**	**Media**	**Q3**	**Máx.**	**Sd**	**Na's**	**Media**	**Sd**
Un ensayo comunitario	1672	2.5	6.7	10	10	9.7	10	10	1.4	0	9.4	1.4
Test Diseños que afectan a la varianza	1677	1.2	5.0	7.5	7.5	7.6	8.8	10	1.6	1	6.9	2.8
Diseño con intercambio	1679	1.4	5.0	6.3	7.5	7.7	9.0	10	1.9	2	6.4	3.6
Diseño Paralelo	1680	1.8	6.0	10	10	9.3	10	10	1.4	1	8.4	3.1
Test RG15	1683	1.1	5.0	6.3	7.5	7.5	7.5	10	1.6	1	6.8	2.7
Meta-análisis con respuesta dicotómica	1686	1.3	5.0	6.9	8.5	7.8	9.0	9.5	1.6	4	5.0	4.2
Ensayo secuencial	1694	2.5	7.5	7.5	7.8	8.5	8.8	10	1.0	1	7.6	2.7
Meta-análisis con respuesta cuantitativa	1696	0.9	5.0	9.0	9.0	8.5	9.0	10	1.6	4	8.4	4.4

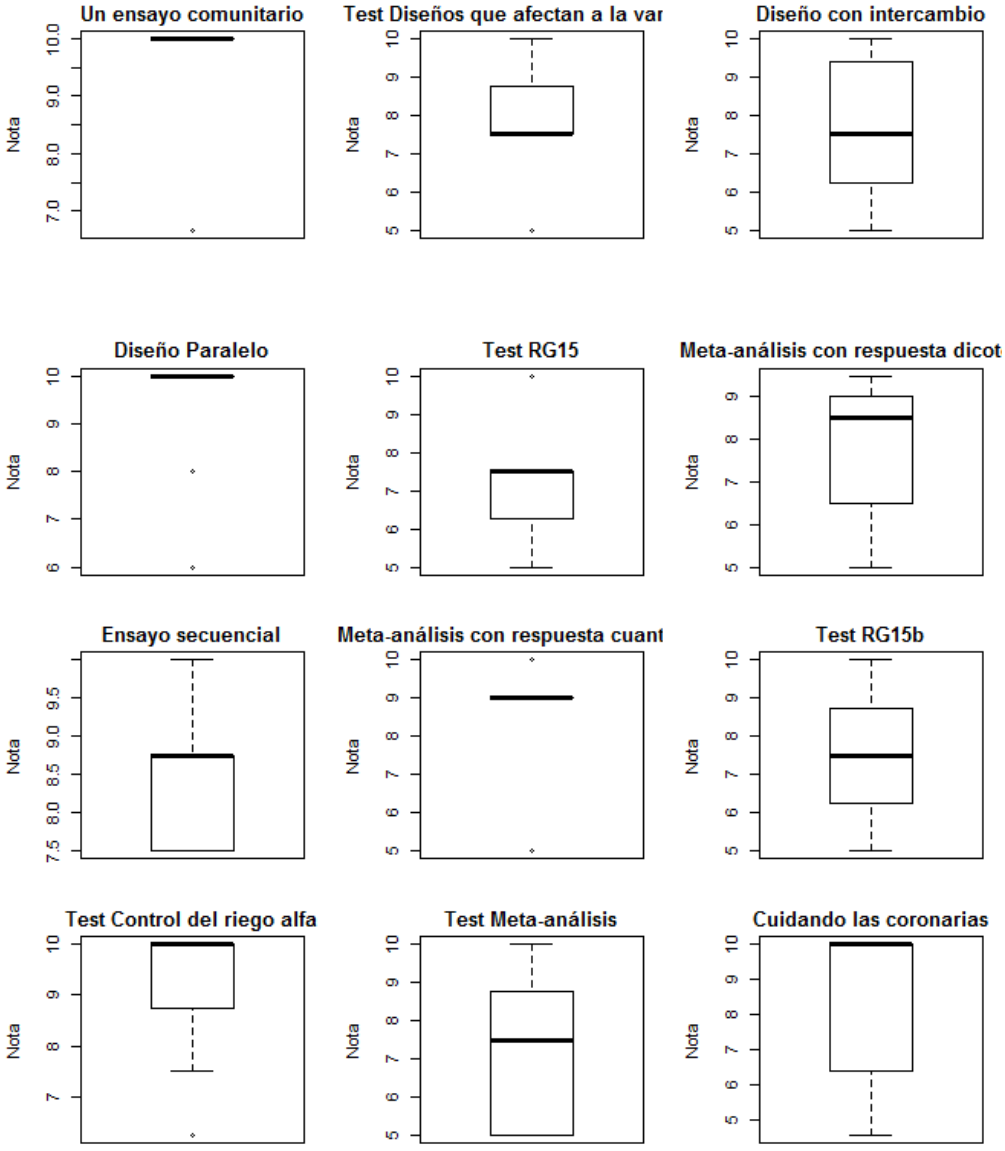
Test RG15b	1698	1.7	5.0	6.3	7.5	7.4	8.8	10	1.5	1	6.6	2.6
Test Control del riego alfa	1699	2.0	6.3	8.8	10	9.0	10	10	1.3	1	7.3	3.0
Test Meta-análisis	1700	1.3	5.0	5.0	7.5	7.5	8.8	10	2.0	1	6.8	2.9
Cuidando las coronarias	1709	1.9	4.6	6.4	10	8.2	10	10	2.2	1	7.3	3.2

Número de alumnos que han iniciado el módulo = 10

Histograma de los ejercicios del módulo 5



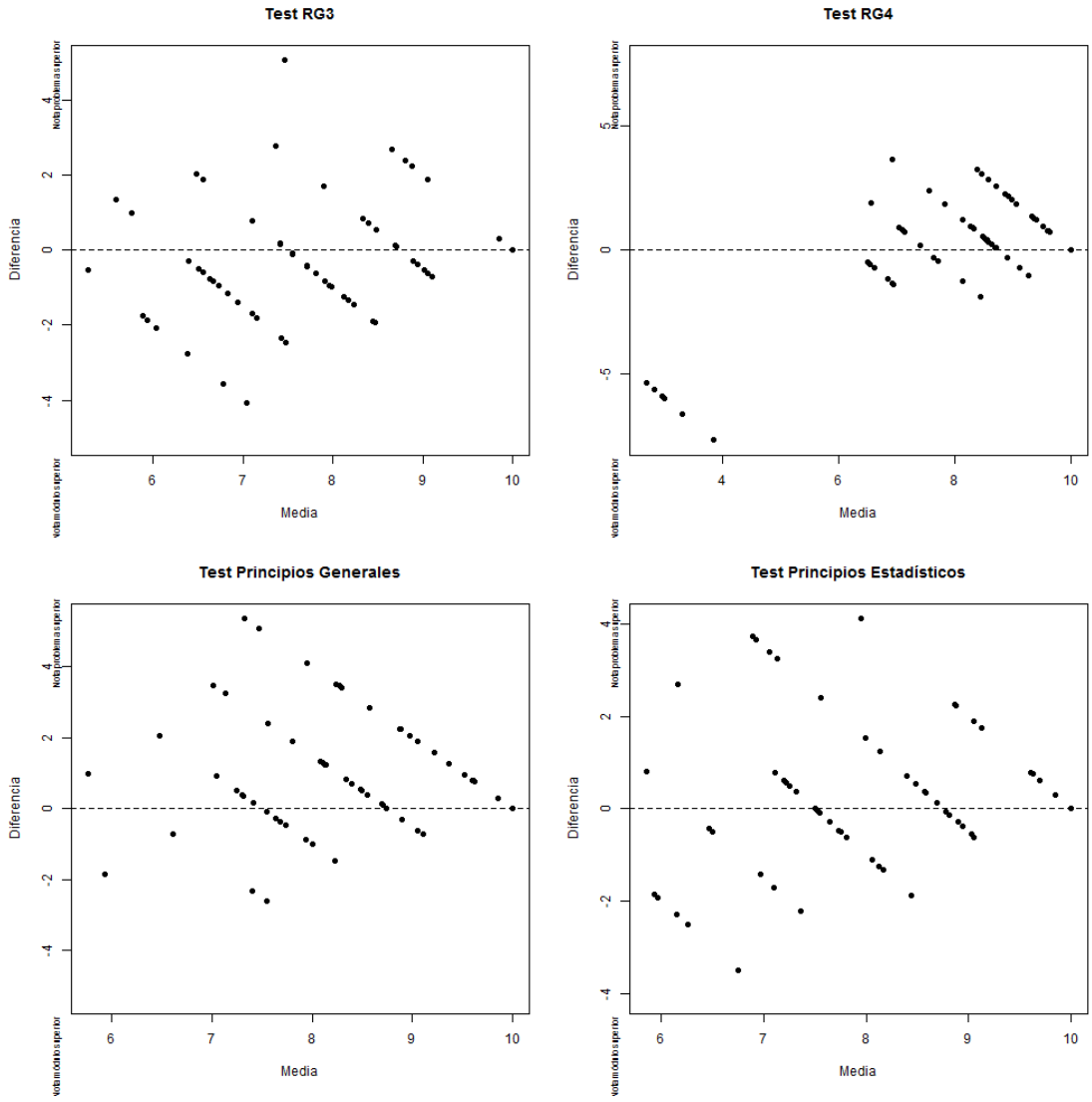
Boxplots de los ejercicios del módulo 5

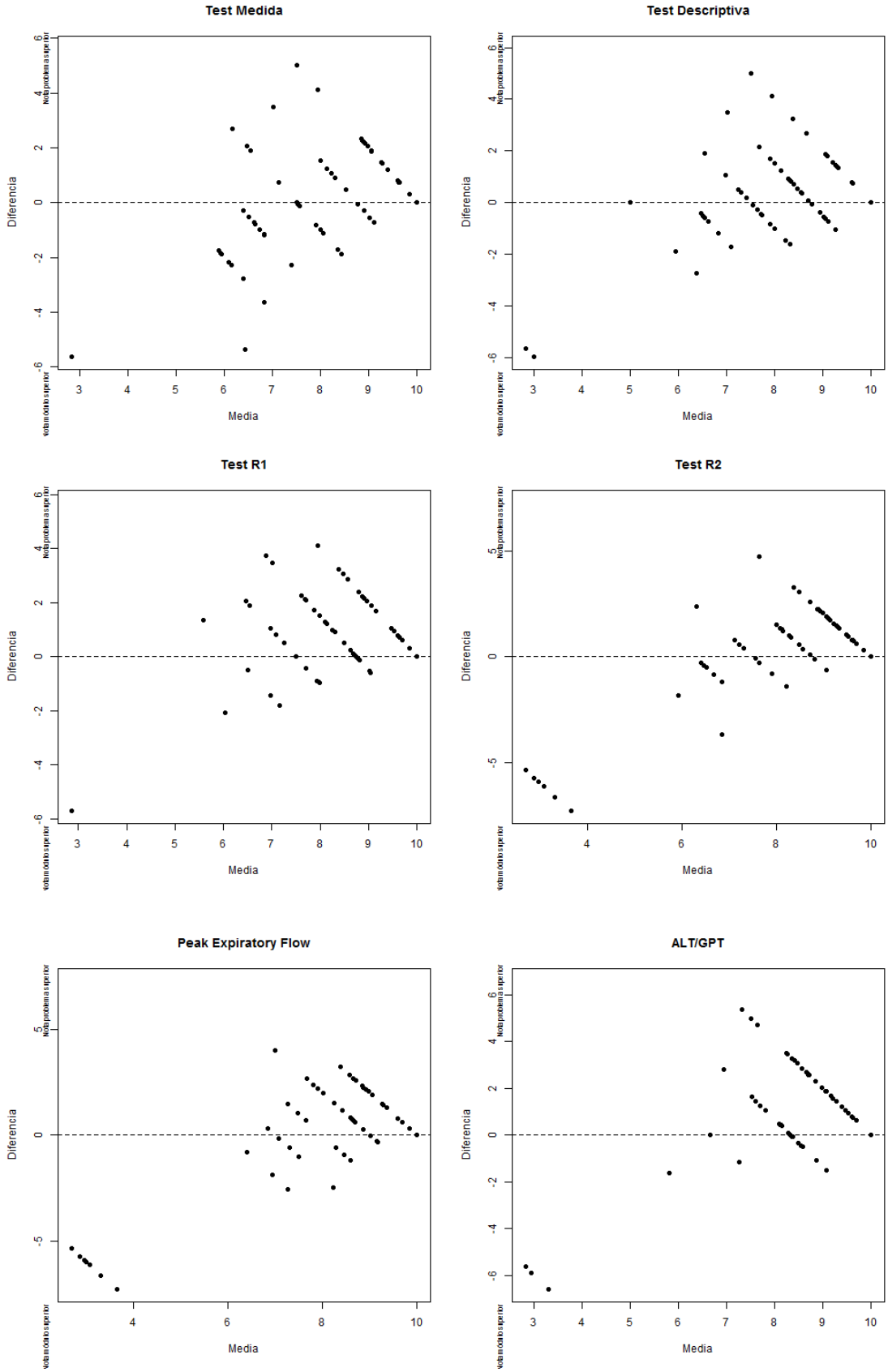


VI.2. Gráficos Baland Altman

VI.2.1. Gráficos Bland Altman módulo 1

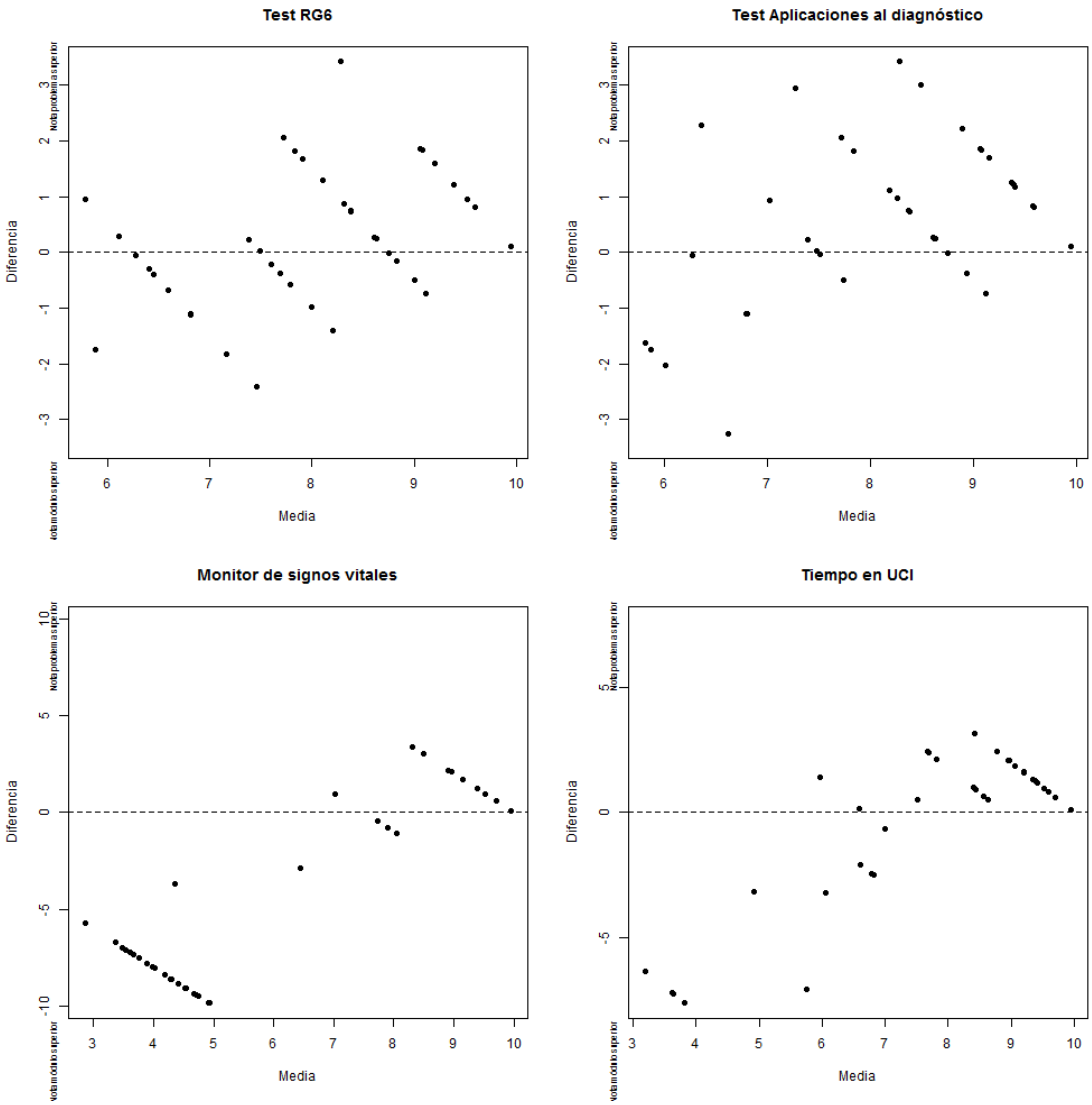
Gráficos de Bland Altman de los ejercicios del módulo 1 que no han sido comentados en el punto 0

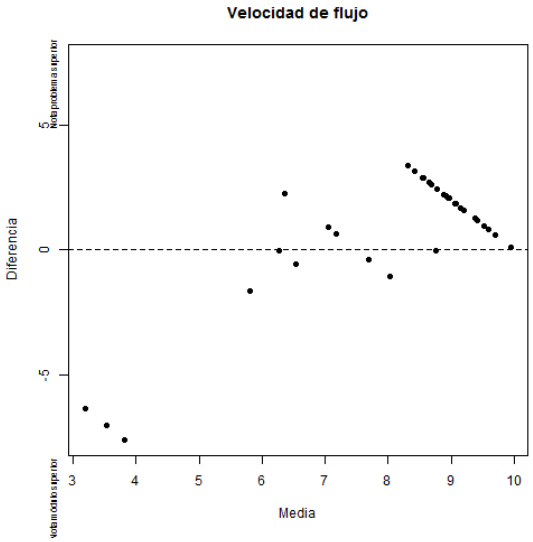


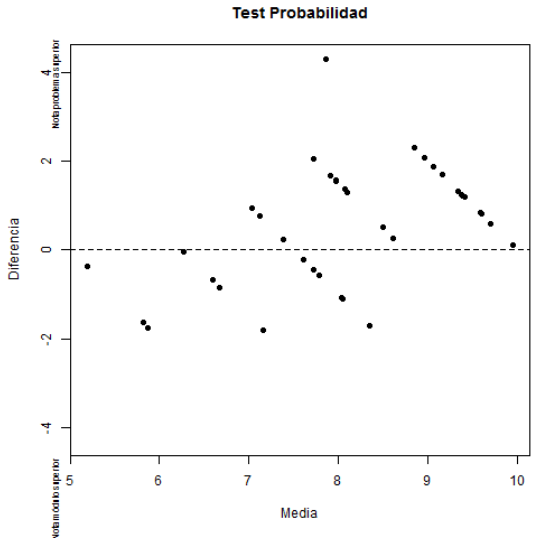

Test Medida
Test Descriptiva
Test R1
Test R2
Peak Expiratory Flow
ALT/GPT
Diferencia
Media
Nota problema superior
Volumen ocho superior

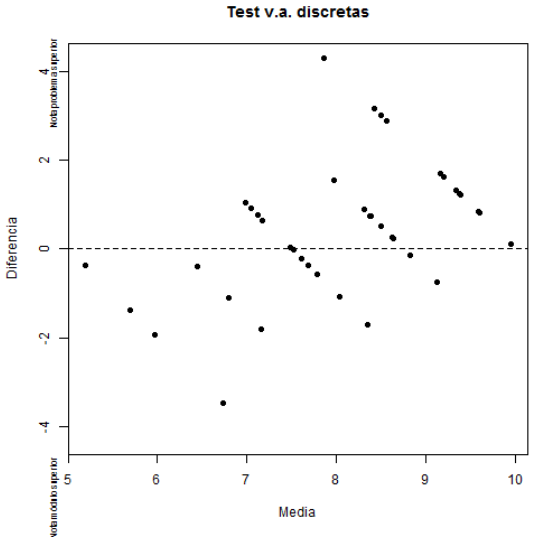
VI.2.2. Gráficos Bland Altman módulo 2

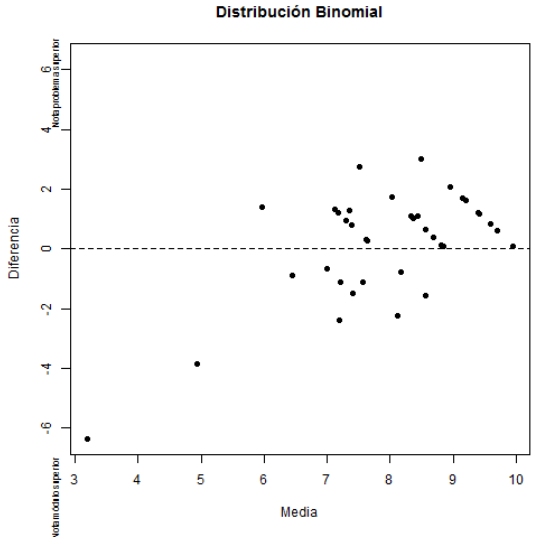
Gráficos de Bland Altman de los ejercicios del módulo 2 que no han sido comentados en el punto 0

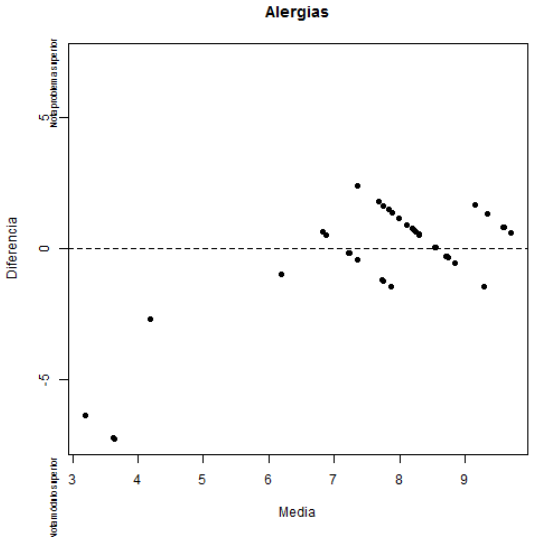


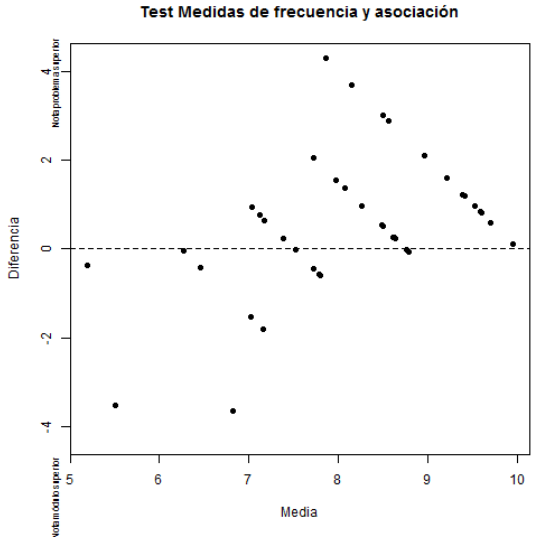

Velocidad de flujo
Diferencia
Media

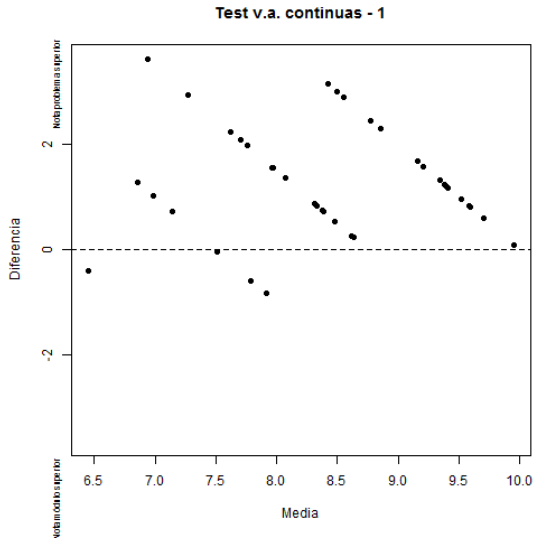

Test Probabilidad
Diferencia
Media

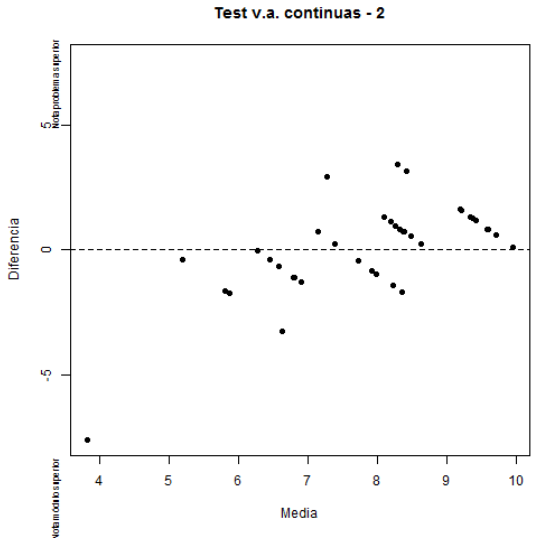

Test v.a. discretas
Diferencia
Media

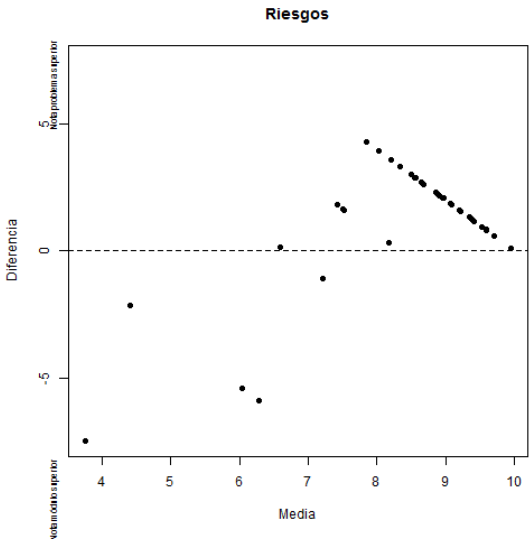

Distribución Binomial
Diferencia
Media


Alergias
Diferencia
Media


Test Medidas de frecuencia y asociación
Diferencia
Media

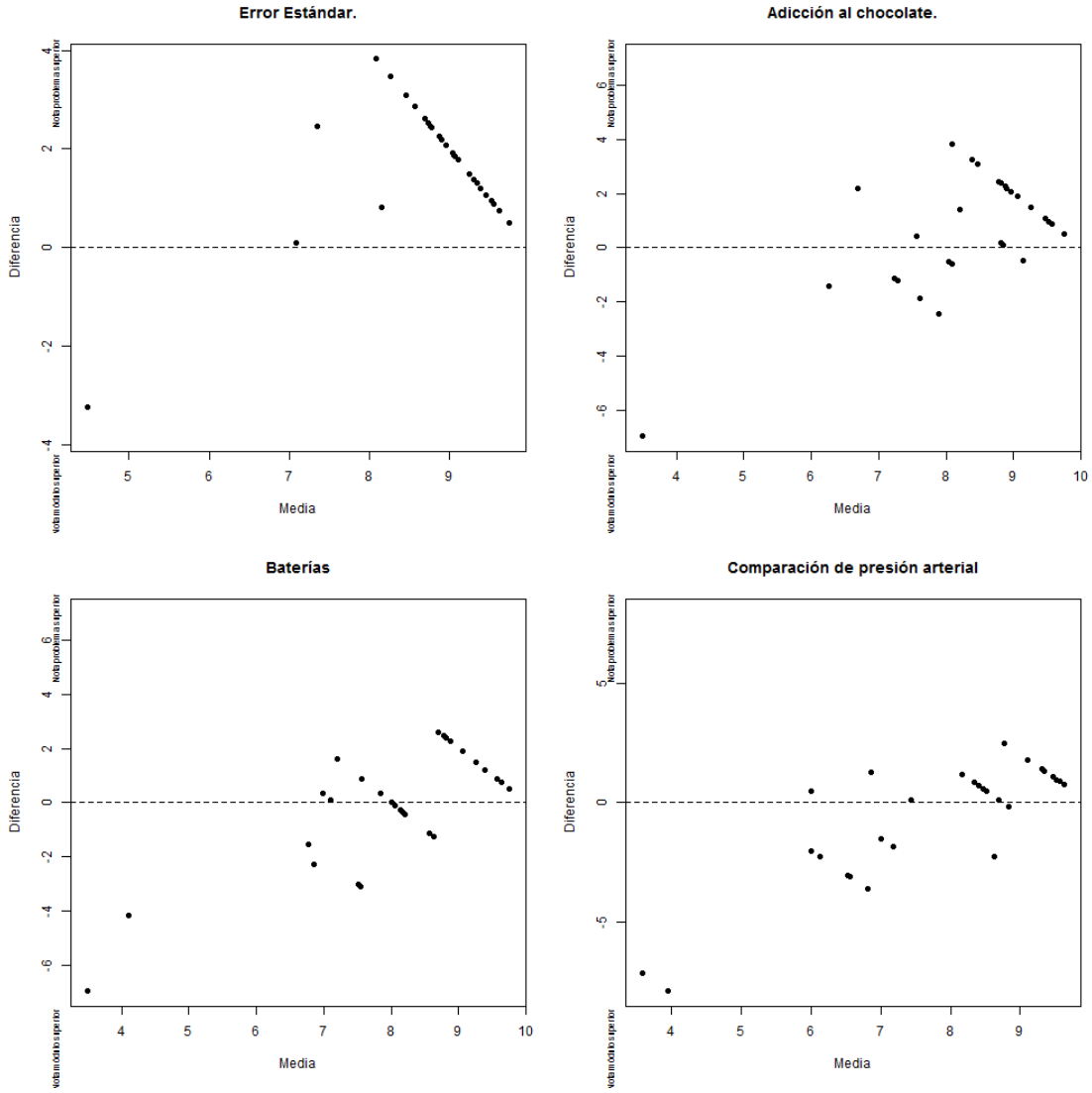

Test v.a. continuas - 1
Diferencia
Media
Nota problema superior
Volumen medio superior
6.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0
-2
0
2

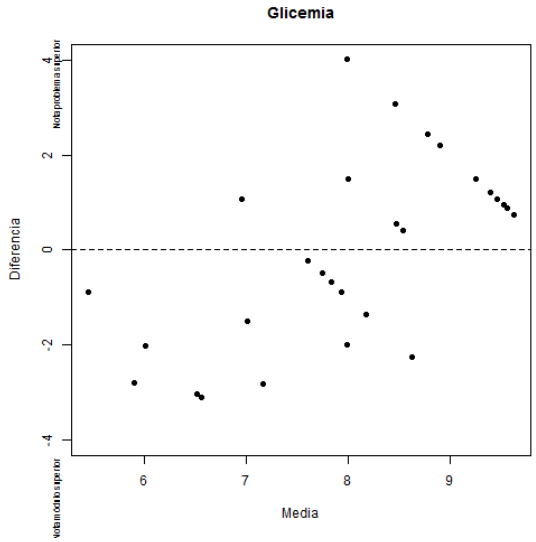

Test v.a. continuas - 2
Diferencia
Media
Nota problema superior
Volumen medio superior
4
5
6
7
8
9
10
-5
0
5

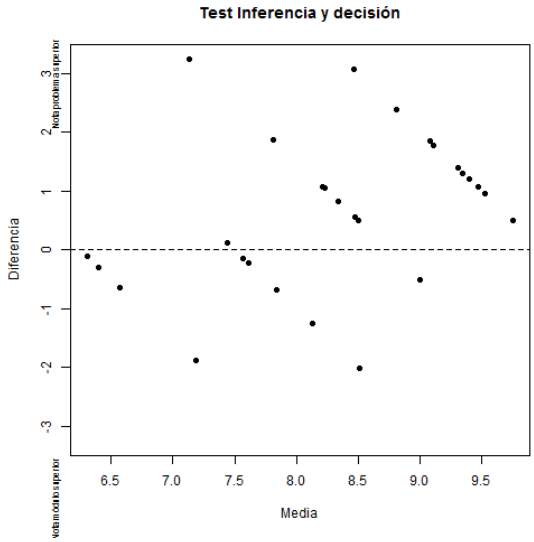

Riesgos
Diferencia
Media
Nota problema superior
Volumen medio superior
4
5
6
7
8
9
10
-5
0
5

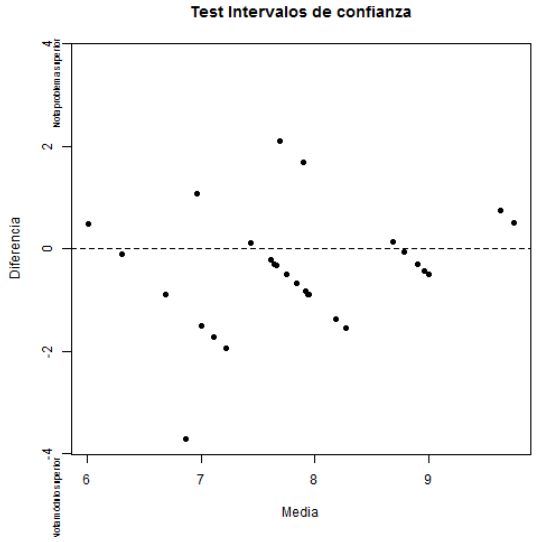
VI.2.3. Gráficos Bland Altman módulo 3

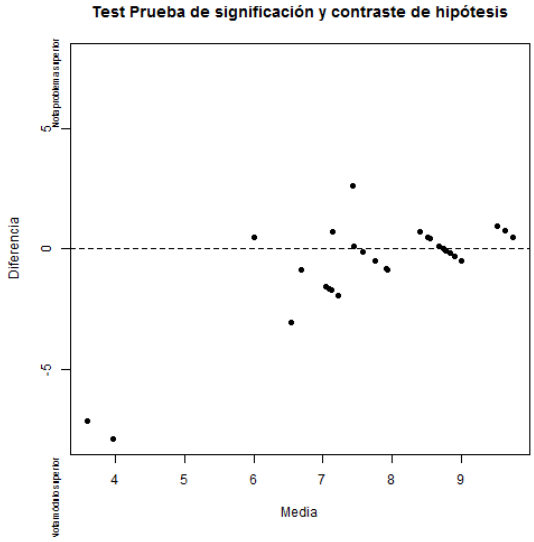
Gráficos de Bland Altman de los ejercicios del módulo 3

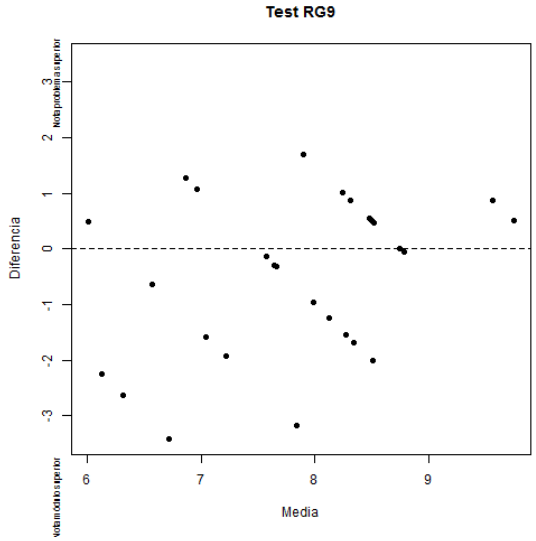


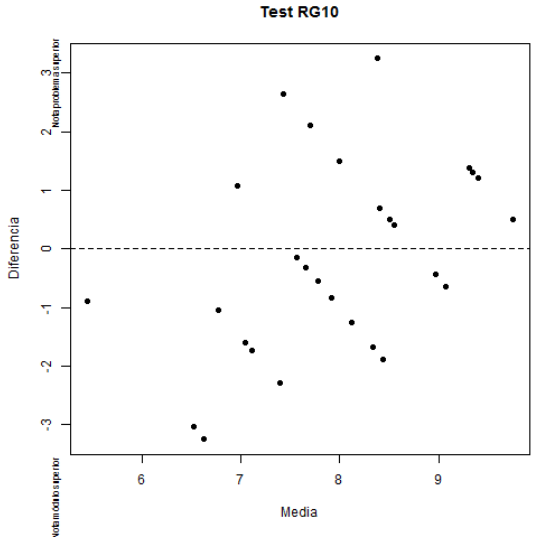

Glicemia
Diferencia
Media
Nota problema superior
Nota problema inferior

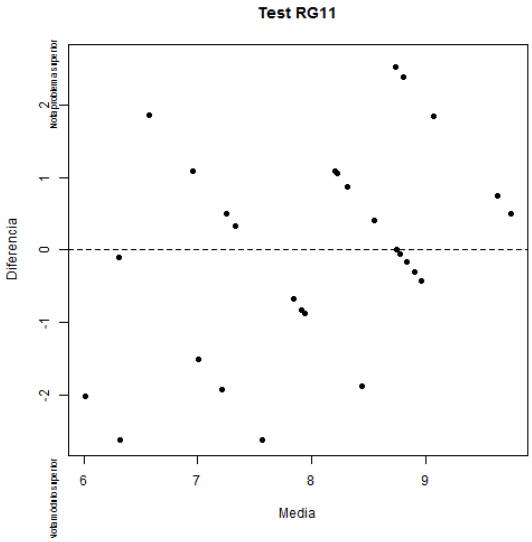

Test Inferencia y decisión
Diferencia
Media
Nota problema superior
Nota problema inferior


Test Intervalos de confianza
Diferencia
Media
Nota problema superior
Nota problema inferior


Test Prueba de significación y contraste de hipótesis
Diferencia
Media
Nota problema superior
Nota problema inferior

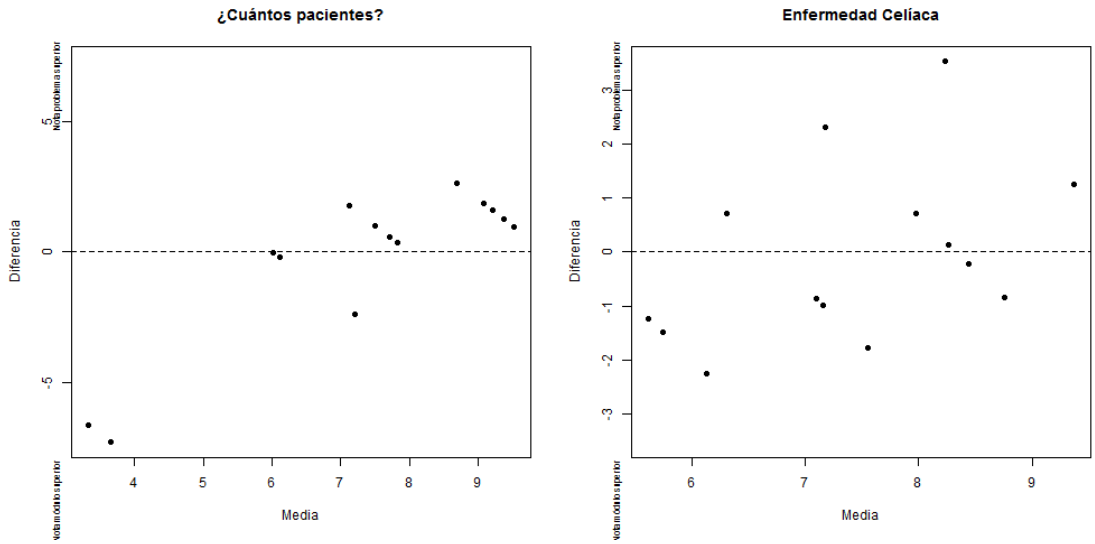

Test RG9
Diferencia
Media
Nota problema superior
Nota problema inferior

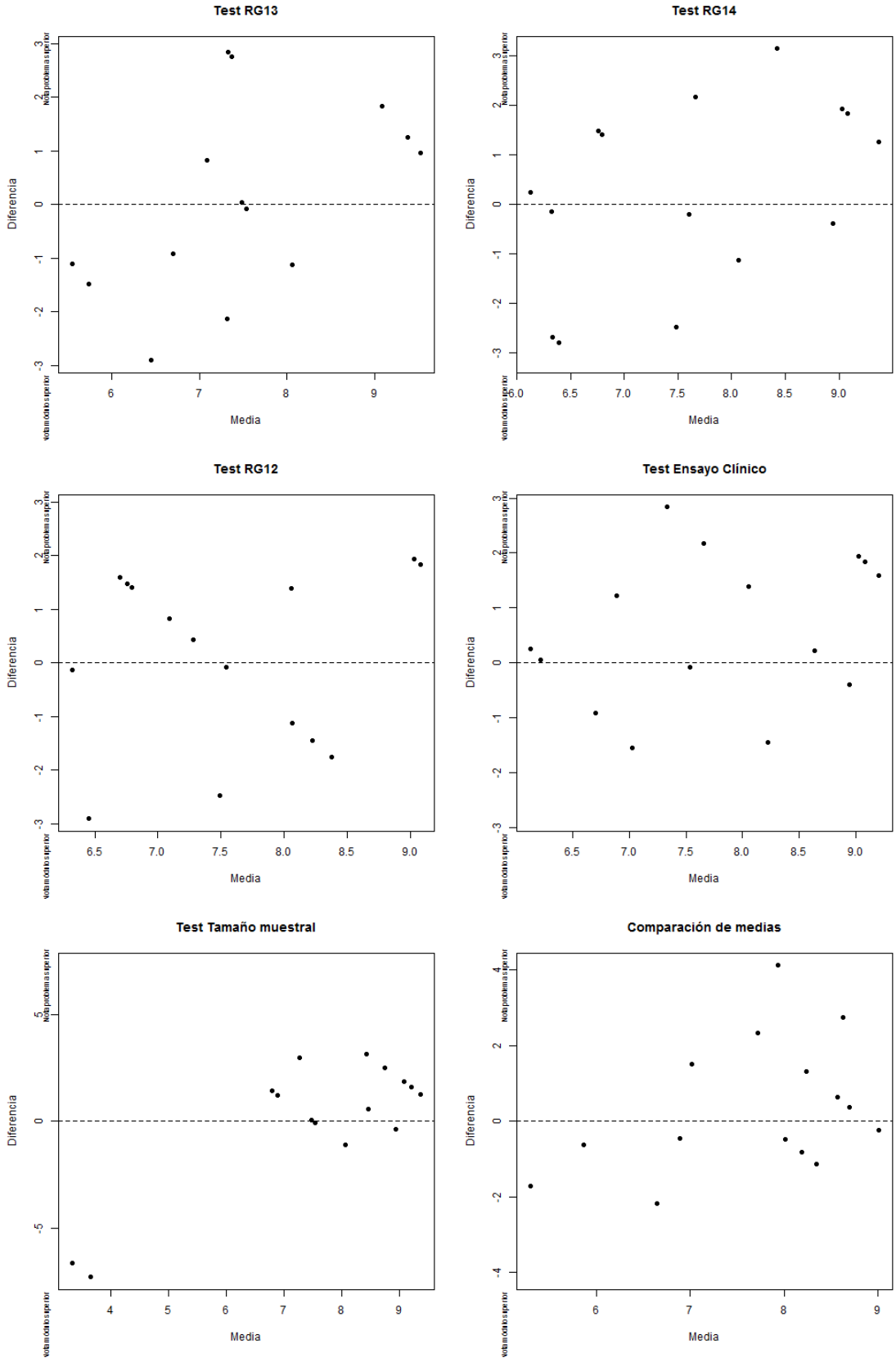

Test RG10
Diferencia
Media
Nota problema superior
Nota problema inferior



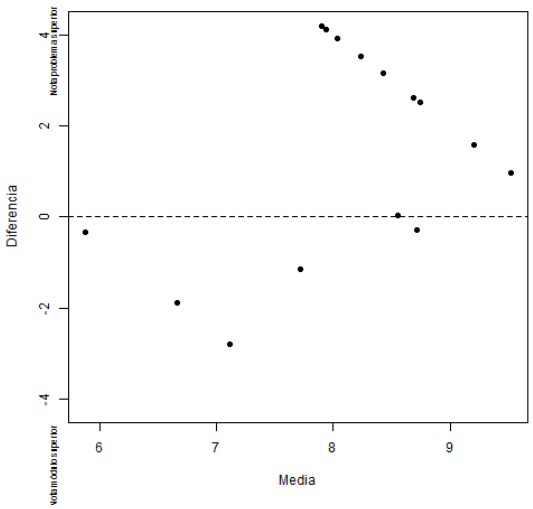
VI.2.4. Gráficos Bland Altman módulo 4

Gráficos de Bland Altman de los ejercicios del módulo 4

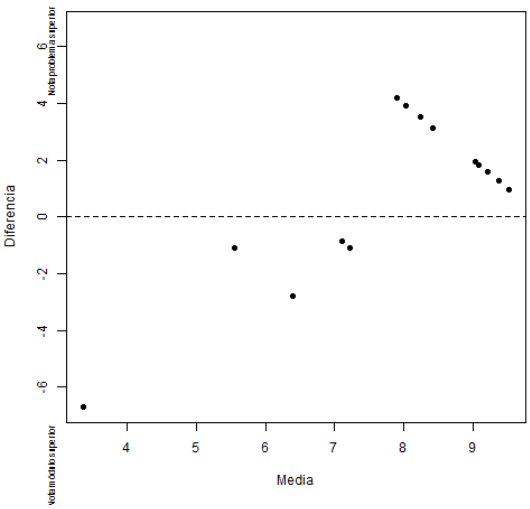



Test RG13
Test RG14
Test RG12
Test Ensayo Clínico
Test Tamaño muestral
Comparación de medias
Diferencia
Media
Nota: problema a superar
Victoria como superior

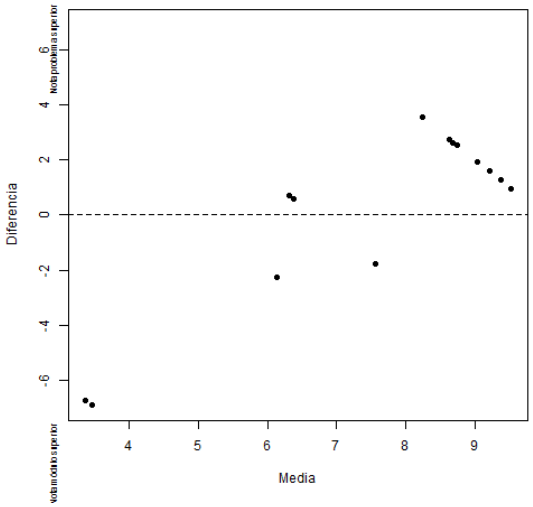
Comparación de proporciones



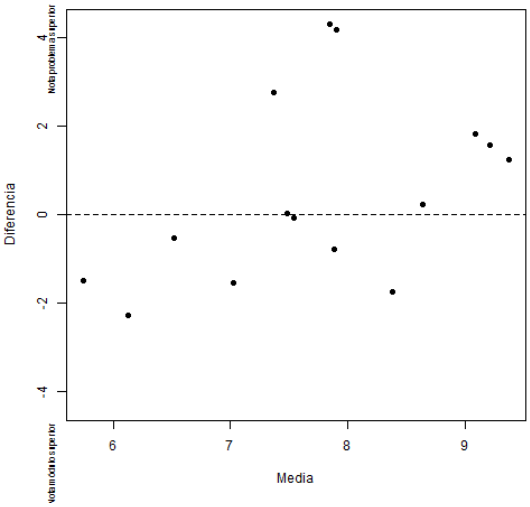
Efecto y cumplimiento del protocolo



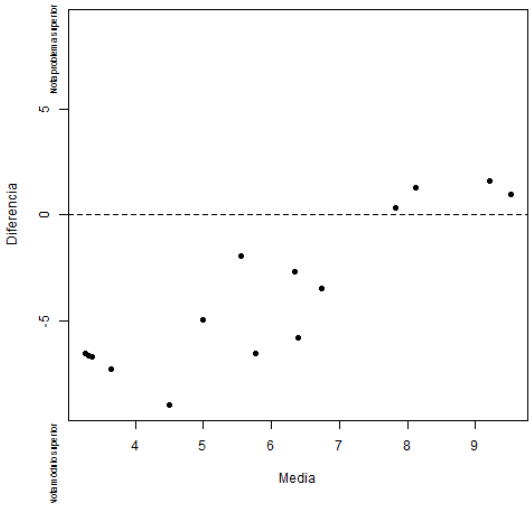
Análisis de supervivencia



Test Efecto

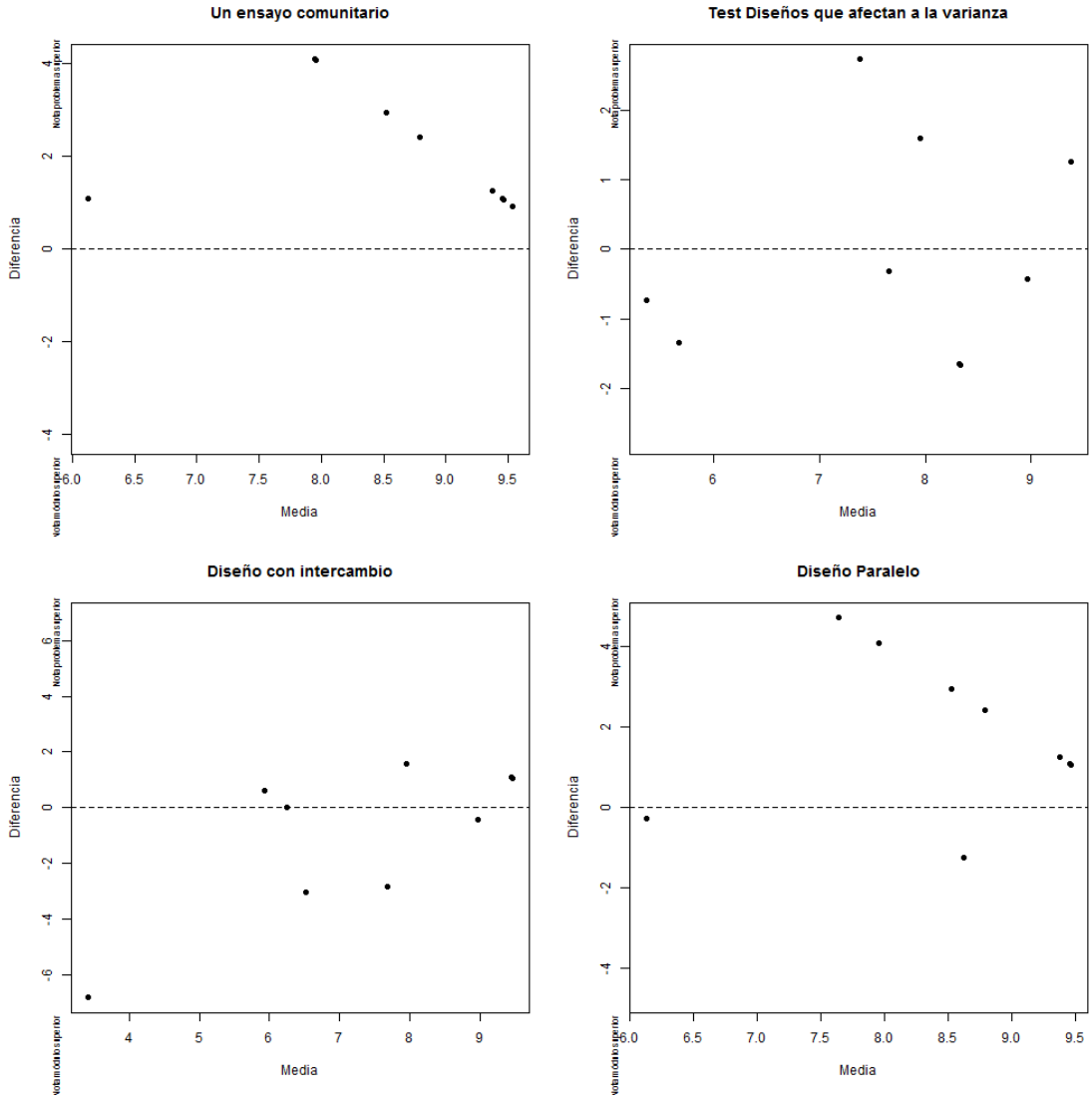


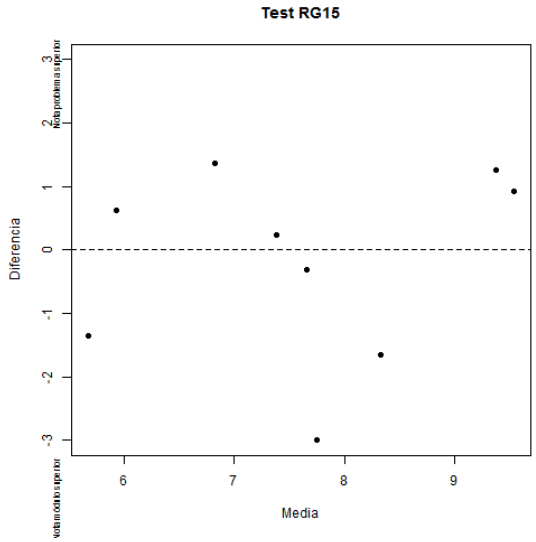
Prevención de accidentes cerebrovasculares

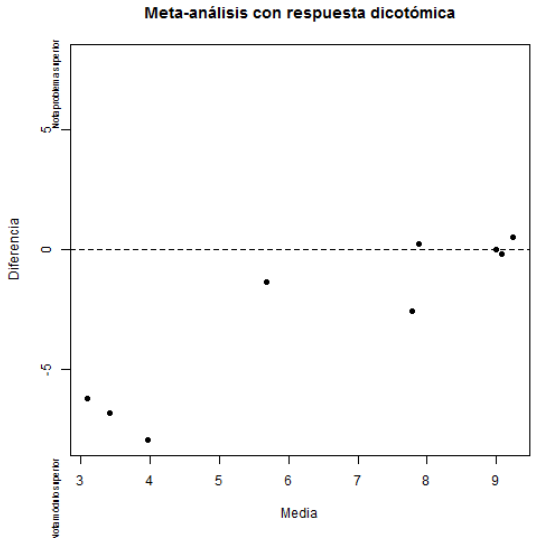


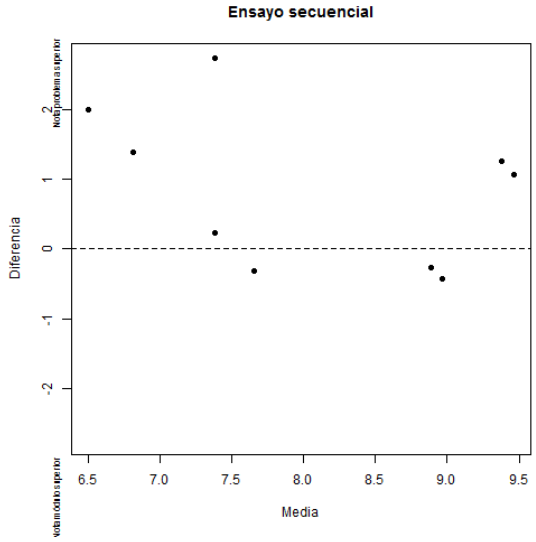
VI.2.5. Gráficos Bland Altman módulo 5

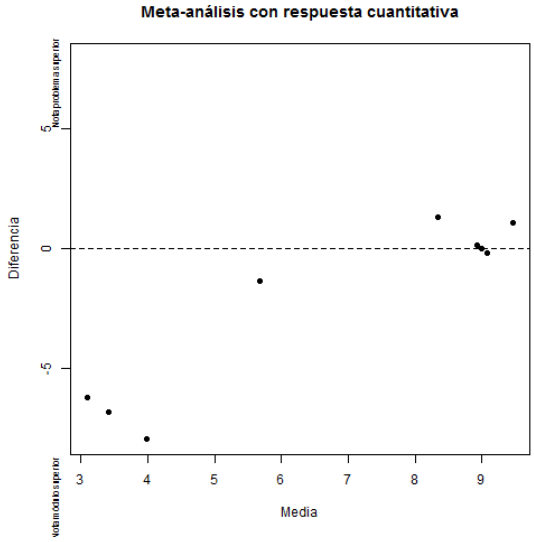
Gráficos de Bland Altman de los ejercicios del módulo 5

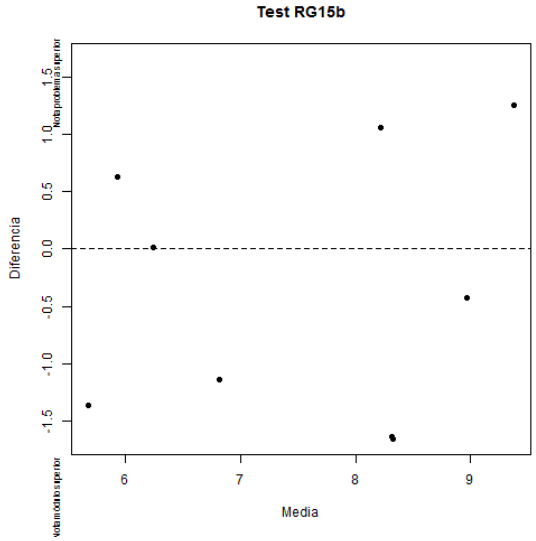


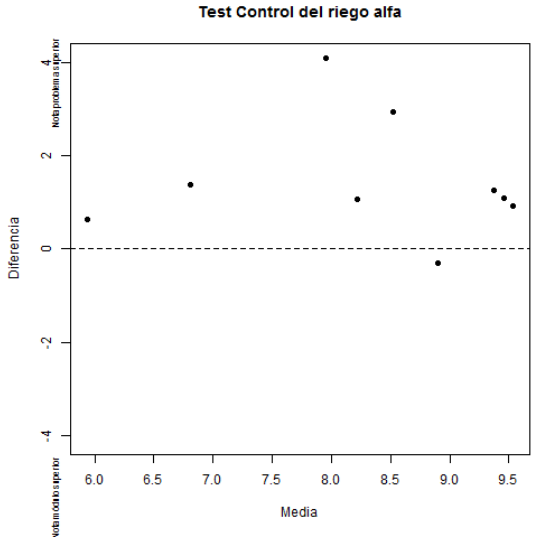

Test RG15
Diferencia
Media

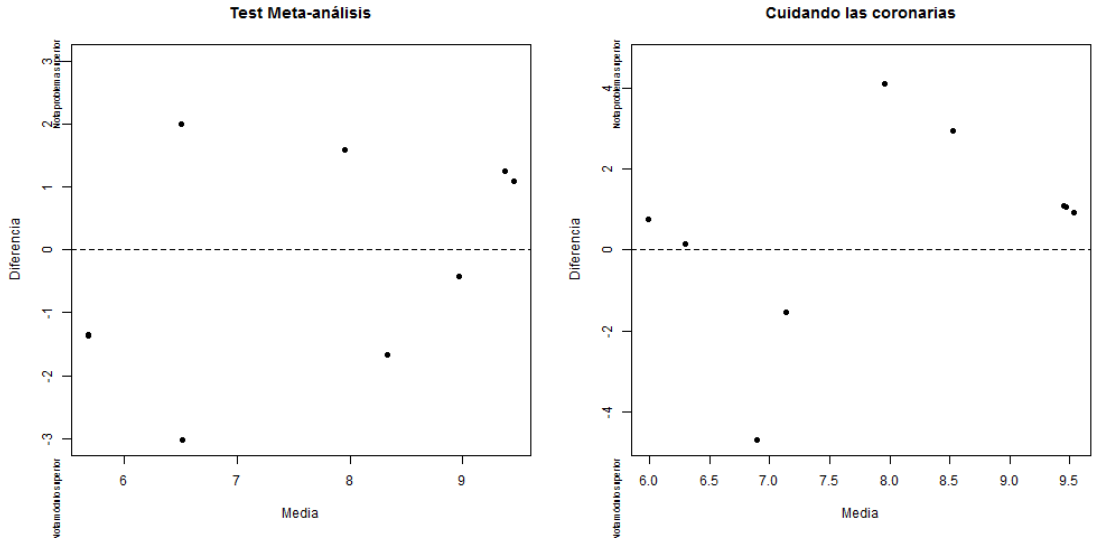

Meta-análisis con respuesta dicotómica
Diferencia
Media


Ensayo secuencial
Diferencia
Media


Meta-análisis con respuesta cuantitativa
Diferencia
Media


Test RG15b
Diferencia
Media


Test Control del riego alfa
Diferencia
Media



VI.3. ACP

En este anexo se explica el cálculo y la extracción de las componentes principales y se muestran los mapas factoriales de variables e individuos del análisis de componentes principales para los diferentes módulos combinando la primera y la segunda dimensión y la primera y la tercera dimensión.

VI.3.1. Cálculo y extracción de CP[6]

A partir de una serie de *m* variables (x_1, x_2, $x_3 \ldots x_m$), se calcula un nuevo conjunto de variables (y_1, y_2, $y_3 \ldots y_p$) incorrelacionadas cuyas varianzas deben decrecer progresivamente.

Cada y_j (donde $j=1 \ldots p$) es una combinación lineal de las *m* variables (x_m) originales:

$$y_j = a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jp}x_p = a'_j x$$

Siendo $a'_j = (a_{1j}, a_{2j}, \ldots, a_{pj})$ un vector de constantes y $x = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$

Para mantener la ortogonalidad de la trasformación de variables, se impone que el módulo del vector $a'_j = (a_{1j}, a_{2j}, \ldots, a_{pj})$ sea igual a 1 $(a'_j \cdot a_j = 1)$ y calcula el primer componente buscando la a_1 que maximice la varianza de y_1, siempre teniendo en cuenta la restricción anterior.

La segunda componente principal se calcula de forma similar imponiendo que la variable y_2 obtenida este incorrelacionada con y_1; del mismo modo, se van obteniendo $a_j's$, que permitan obtener hasta y_p variables incorrelacionadas entre si y cuyas varianzas serán cada vez menores.

Para la extracción de estos CP se utiliza el método de multiplicadores de *Lagrange*.

El objetivo es maximizar la función $a'_1 \sum a_1$ sujeta a la restricción $a'_j \cdot a_j = 1$, donde la incógnita es a_1, que es el vector desconocido que da la combinación lineal óptima. Así, se construye la función: $L(a_1) = a'_1 \sum a_1 - \lambda(a'_1 \cdot a_1 - 1)$, y se busca maximizar $(\sum - \lambda I)\, a_1 = 0$. En realidad, la función a maximizar es un sistema lineal de ecuaciones, por lo que para que el sistema tenga una solución distinta a 0 la matriz $(\sum - \lambda I)$ tiene que ser singular, y esto implica que el determinante tiene que ser igual a 0, es decir: $|\sum - \lambda I| = 0$.

La matriz de covarianzas $\sum$ es de orden p y si además se define como positiva o semi-positiva tendrá p auto valores distintos (λ_1, λ_2, …, λ_p). Si se desarrolla la expresión anterior, se obtiene $\sum a_1 = \lambda I a_1$; Luego, la varianza de y_1 es igual a $\text{Var}(y_1) = \text{Var}(a'_1 x) = a'_1 \sum a_1 = a'_1 \lambda I a_1 = \lambda \cdot 1 = \lambda$. Por lo tanto, para maximizar las varianza de $y_1$se escoge el mayor autovalor λ_1 y el correspondiente autovector a_1.

El segundo componente principal ($y_2 = a'_2\, x$), se obtiene de forma parecida y es necesario que no esté correlacionado con el anterior componente principal, es decir, $\text{Cov}(y_1, y_2) = 0$, por lo tanto se requiere que $a'_2 \sum a_1 = 0$. Los vectores tienen que ser ortogonales $(a'_2 \cdot a_1 = 0)$. El siguiente paso es maximizar la varianza de y_2 $(a_2 \sum a_2)$, sujeto $a'_2 a_2 = 1$ y $a'_2 a_1 = 0$.

Se parte de la función $L(a_2) = a'_2 \sum a_2 - \lambda(a'_2 \cdot a_2 - 1) - \delta a'_2 \cdot a_1$; y se busca maximizar $2 \sum a_2 - 2\lambda a_2 - \delta a_1 = 0$; que multiplicado por a_1 permite obtener $2 a_1 \sum a_2 - \delta = 0$; siendo $a'_1 \cdot a_2 = a'_2 \cdot a_1 = 0$ y $a'_1 \cdot a_1 = 1$. De este modo la expresión final es $\frac{\vartheta L(a_2)}{\vartheta a_2} = 2 \sum a_2 - 2\lambda a_2 = (\sum - \lambda I)\, a_2 = 0$. Igual que para la primera componente, se elige λ como el segundo mayor valor propio de la matriz $\sum$ con su correspondiente autovector asociado a_2.

Este razonamiento se extiende al resto de componentes, de modo que al j-ésimo componente le corresponde el j-ésimo valor propio.

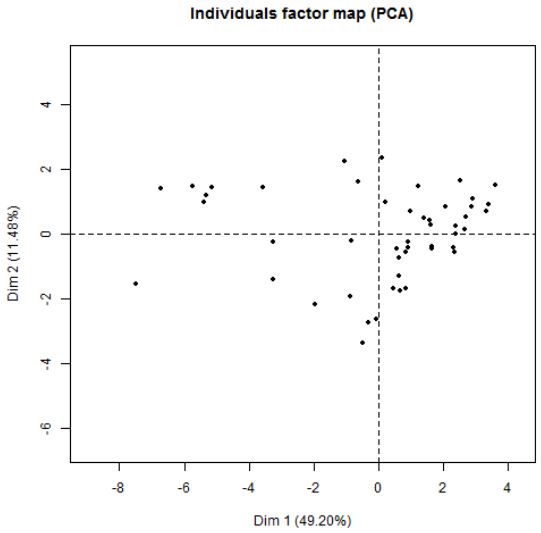
Por lo tanto, todos los componentes y_p se pueden expresar como el producto de una matriz formada por los vectores propios multiplicada por el vector *x* que contiene las variables originales (x_1, x_2, x_3…x_m) reducidas a (x_1, … x_p) variables:

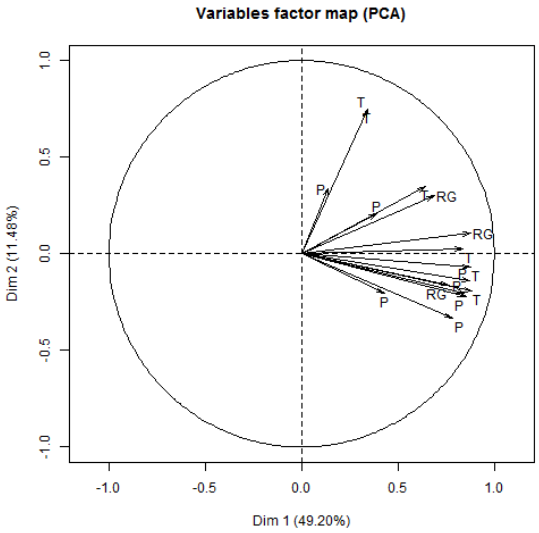
$$y = Ax \quad \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \ldots & a_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ a_{p1} & a_{p2} & \ldots & a_{pp} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} \text{ siendo } \begin{array}{l} Var(y_1) = \lambda_1 \\ Var(y_2) = \lambda_2 \\ \vdots \\ Var(y_p) = \lambda_p \end{array}$$

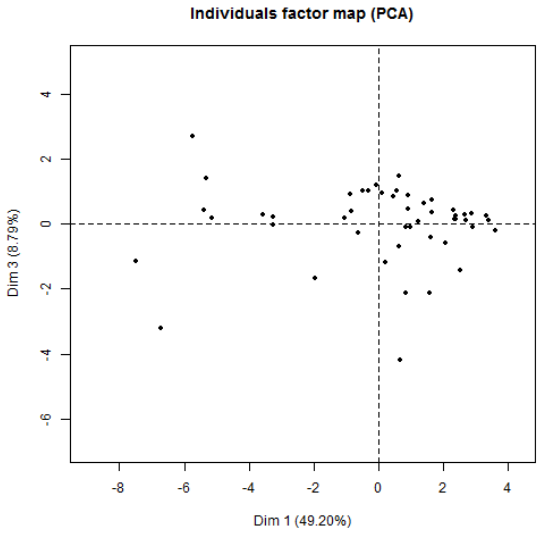
El hecho de que la suma de las varianzas de las variables originales y la suma de las varianzas de las componentes sean iguales permite hablar del porcentaje de varianza total que recoge un componente principal: $\frac{\lambda_i}{\sum_{i=1}^{p} \lambda_i} = \frac{\lambda_i}{\sum_{i=1}^{p} Var(x_i)}$.

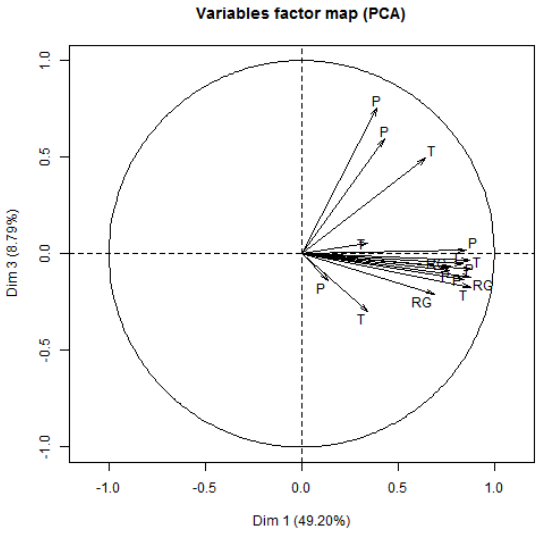
En la práctica, se busca seleccionar un número bajo de componentes que permita recoger el máximo porcentaje de variabilidad total $\sum_{i=1}^{p} Var(\lambda_i)$.

VI.3.2. ACP módulo 2

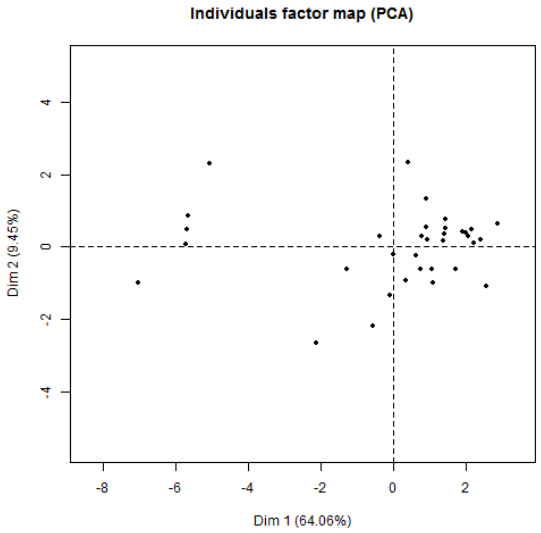


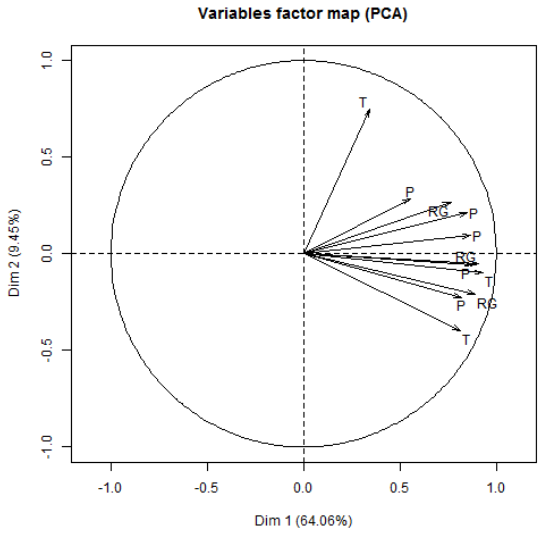


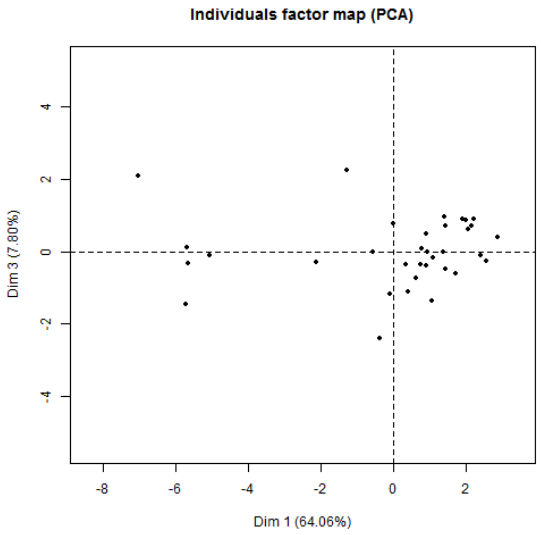


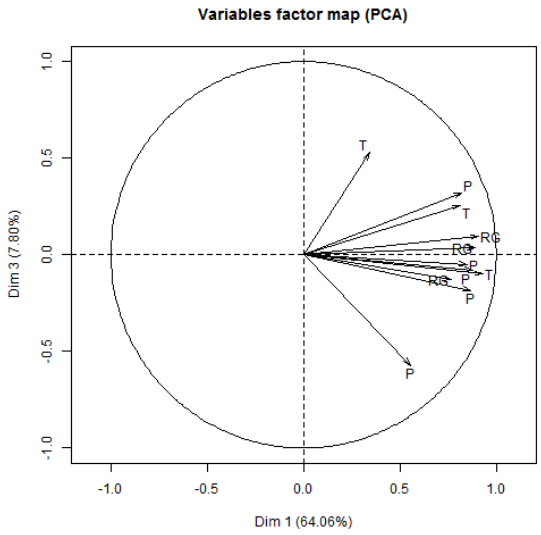


VI.3.3. ACP módulo 3

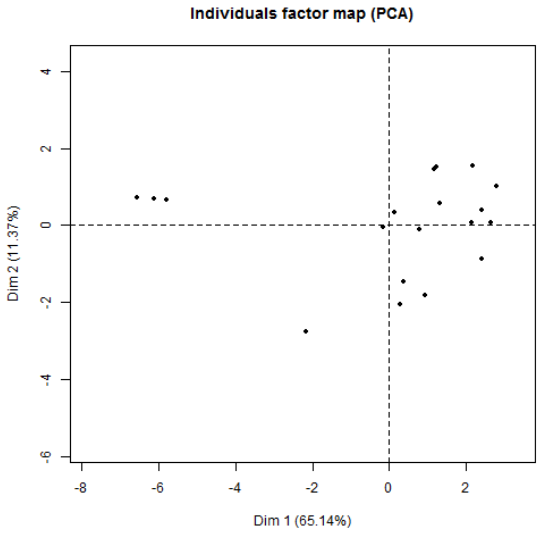


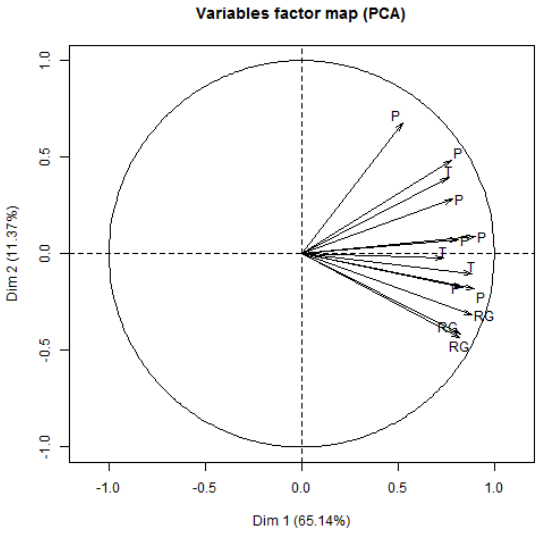


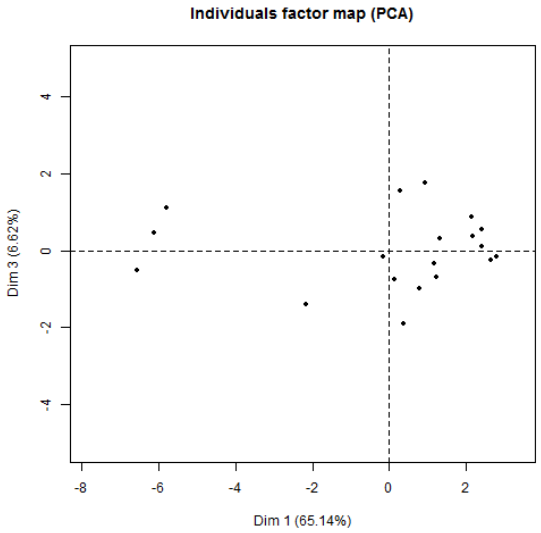


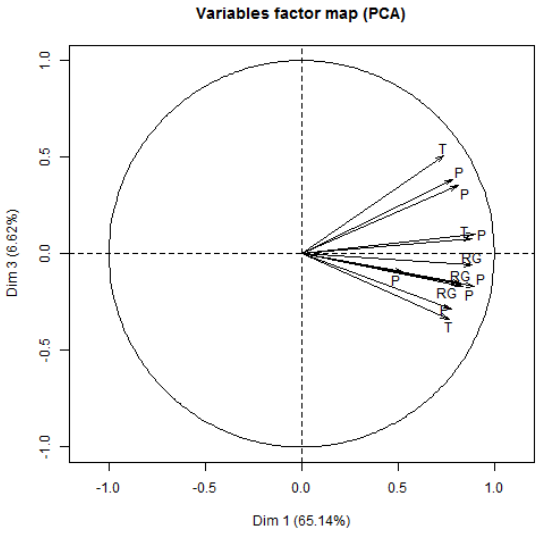


VI.3.4. ACP módulo 4

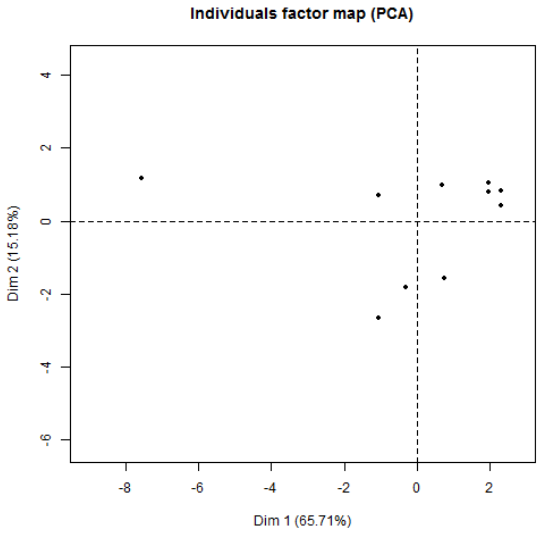


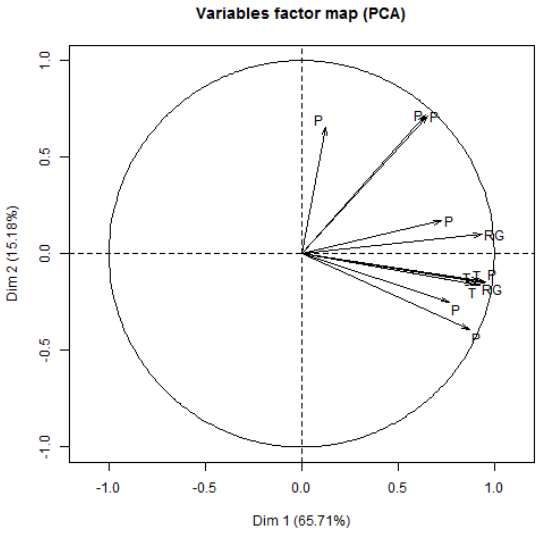


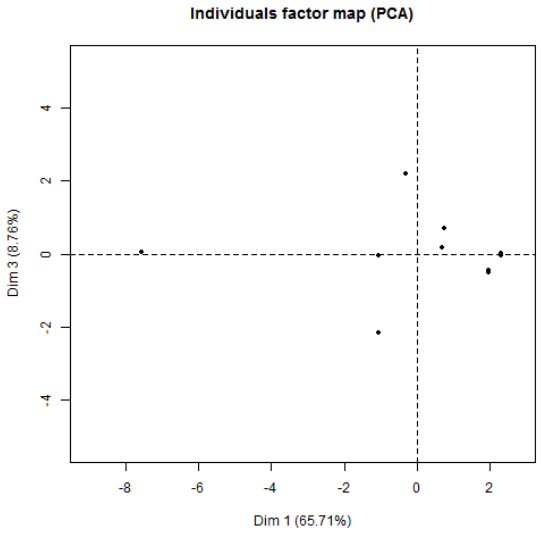


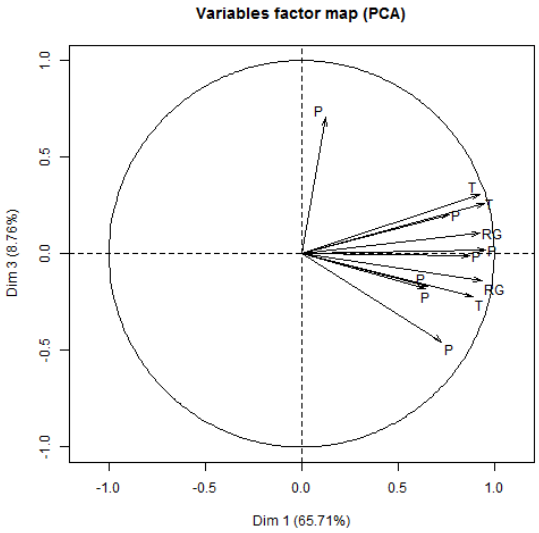


VI.3.5. ACP módulo 5









VI.4. Códigos R

VI.4.1. Parámetros

```
#Parámetros
#-- Distancia horizontal entre cuadros (módulos) del gráfico (app1)
Sx = 2
#-- Distancia vertical entre cuadros (módulos) del gráfico (app1)
Sy=1
#-- Problemas actual vs equivalente
equiv = list(c(1398, 1742), c(1693, 1709), c(1399, 1741), c(1492, 1738))


#-- Identificadores problemas
mod1ID=c(210, 222, 232)#Indentificadores problemas M1
mod2ID=c(212, 223, 233)#Indentificadores problemas M2
mod3ID=c(216, 224, 234)#Indentificadores problemas M3
mod4ID=c(218, 225, 235)#Indentificadores problemas M4
mod5ID=c(220, 228, 236)#Indentificadores problemas M5
#-- Número de módulos
NMODS<-5
#-- Parámetros altura anchura
alto<-250
ancho<-250
```

VI.4.2. Funciones

```
#-- Asignar módulo a prob y a ejs.x
asimod<-function(prob){
  prob$mod[prob$Asig %in% c(mod1ID)] = 1
  prob$mod[prob$Asig %in% c(mod2ID)] = 2
  prob$mod[prob$Asig %in% c(mod3ID)] = 3
  prob$mod[prob$Asig %in% c(mod4ID)] = 4
  prob$mod[prob$Asig %in% c(mod5ID)] = 5
  return(prob)}
asimod1<-function(ejs.x){
  ejs.x$mod[ejs.x$Asig %in% c(mod1ID)] = 1
  ejs.x$mod[ejs.x$Asig %in% c(mod2ID)] = 2
  ejs.x$mod[ejs.x$Asig %in% c(mod3ID)] = 3
  ejs.x$mod[ejs.x$Asig %in% c(mod4ID)] = 4
  ejs.x$mod[ejs.x$Asig %in% c(mod5ID)] = 5
  return(ejs.x)}
#-- Incluir identificador de cada problema de cada módulo
pmod<-function(prob){
  ppmod = list(
    m1 = as.character(sort(unique(prob$Idprob[prob$mod==1 &
is.na(prob$NO)]))),
    m2 = as.character(sort(unique(prob$Idprob[prob$mod==2 &
is.na(prob$NO)]))),
    m3 = as.character(sort(unique(prob$Idprob[prob$mod==3 &
is.na(prob$NO)]))),
    m4 = as.character(sort(unique(prob$Idprob[prob$mod==4 &
is.na(prob$NO)]))),
    m5 = as.character(sort(unique(prob$Idprob[prob$mod==5 & is.na(prob$NO)])))
  )
  return(ppmod)}
```

```
#-- Eliminar los problemas equivalentes
elimequiv<-function(equiv,ppmod,ejs.x){
  for (i in 1:length(equiv)) {
    s = which(ejs.x$Prob %in% equiv[[i]])
    ejs.x$Prob[s] = max(equiv[[i]])
    for (m in 1:5) {#quitarlos por modulo
      s = which(ppmod[[m]] %in% equiv[[i]] & ppmod[[m]] < max(equiv[[i]]))
      if (length(s)>0) ppmod[[m]] = ppmod[[m]][-s]
    }
  }
  return(list(ppmod,ejs.x))
}
#-- Cálculo de notas medias de alumnos por módulo
lista<-function(x,ppmod){
  notamaxtot = tapply(x$Nota, list(x$Login, x$Prob), max)
  #--Lista con notas medias por módulo y alumno
  notamaxtot[is.na(notamaxtot)]=0
  notamedmod<-list()
  notamedmod[[1]]<-apply(notamaxtot[,ppmod$m1],1,mean)
  notamedmod[[2]]<-apply(notamaxtot[,ppmod$m2],1,mean)
  notamedmod[[3]]<-apply(notamaxtot[,ppmod$m3],1,mean)
  notamedmod[[4]]<-apply(notamaxtot[,ppmod$m4],1,mean)
  notamedmod[[5]]<-apply(notamaxtot[,ppmod$m5],1,mean)
  return(notamedmod)
}
#-- Cálculo del número de ejecuciones totales de los alumnos por módulo
lista1<-function(x,ppmod){
  notamaxtot = tapply(x$Nota, list(x$Login, x$Prob), max)
  notamaxtot1<-notamaxtot
  Nprob<-
length(ppmod[[1]])+length(ppmod[[2]])+length(ppmod[[3]])+length(ppmod[[4]])+le
ngth(ppmod[[5]])
  #Cálculo de número de ejecuciones por módulo
  for (k in 1:length(rownames(notamaxtot1))){
    for ( i in 1:Nprob){
      if(is.na(notamaxtot1[k,i])){
        notamaxtot1[k,i]=0}
      else {notamaxtot1[k,i]=1}
    }
  }
  numejsmod<-list()
  numejsmod[[1]]<-apply(notamaxtot1[,ppmod$m1],2,sum)
  numejsmod[[2]]<-apply(notamaxtot1[,ppmod$m2],2,sum)
  numejsmod[[3]]<-apply(notamaxtot1[,ppmod$m3],2,sum)
  numejsmod[[4]]<-apply(notamaxtot1[,ppmod$m4],2,sum)
  numejsmod[[5]]<-apply(notamaxtot1[,ppmod$m5],2,sum)
  return(numejsmod)}
#-- Clasificación de problemas según sean test, RG o problemas
clasprob<-function(ppmod,notamaxprob,j){
  ppmod1<-vector()
  Np<-dim(notamaxprob)[2]
  for(i in 1:Np){
    ppmod1[i]<-unique(prob$Nombre[prob$Idprob==ppmod[[j]][i]])
   }
  posvect<- grep("RG", ppmod1)
  ppmod1[posvect]<-"RG"
  posvect2<-grep("Test", ppmod1)
```

```
  ppmod1[posvect2]<-"Test"
  for(i in 1:length(colnames(notamaxprob))){
    if((ppmod1[i]!="RG" && ppmod1[i]!="Test")){
      ppmod1[i]<-"Prob"
    }
  }
  colnames(notamaxprob)<-ppmod1
  return(notamaxprob)
}
#-- Cálculo IC
IC = function(datos, varianza = var(datos),
              nivel.conf = 0.95)
{
  z = qnorm((1 - nivel.conf)/2, lower.tail = FALSE)
  m = mean(datos)
  dt = sqrt(varianza/length(datos))
  c(m - z * dt, m + z * dt)
}
#--Gráfico Bland Altman
BlandAltman <- function(y1,y2,y3,tit){
  Bmean<-y2
  Bdif <- y2-y3
  ymax <- max(abs(Bdif))
  plot(Bmean ,Bdif ,ylim=c(-ymax,ymax),
       xlab="Medias",
       ylab="Diferencias",
       main=tit,pch=19)
  abline(h=0,lty=2)
  mtext("Fáciles",2,line= 0.5,
        at=1.1*ymax,adj=1,cex=0.7)
  mtext("Difíciles",2,line= 0.5,
        at=-1.1*ymax,adj=1,cex=0.7)
}
## ppmod1<-
c("RG1a","RG1b","RG1c","RG3a","RG3b","T1a","T1b","T1c","T3a","T3b","T2a","T2b"
,"T3c","P3a","P3b")
```

VI.4.3. Aplicativo Shiny 1 (Análisis gráfico por módulos)

Server

```
shinyServer(function(input, output) {

  #-- Carga de valores y funciones
  source("parameters.R")
  source("functions12.R")
  #-- Carga de datos
  load(url("http://ka.upc.es/test-r/asignaturas.Rdata"))
  load(url("http://ka.upc.es/test-r/ejecuciones.Rdata"))
  load(url("http://ka.upc.es/test-r/numalumnos.Rdata"))
  load(url("http://ka.upc.es/test-r/problemas.Rdata"))
  #-- Añadir el módulo de cada problema en el data.frame prob
  prob<-asimod(prob)
  #-- Identificar los problemas de cada módulo, sin contar no puntuables
  F = iconv(prob$Nombre, from="", to="UTF-8", "byte")
  #-- Eliminar los problemas de dedicación y de Mis notas
```

```
  noprobs = c(grep("^Dedica", F), grep("Mis notas", F))
  #-- Se asigna un 1 a los problemas no puntuables
  prob$NO[noprobs] = 1
  ppmod<-pmod(prob)
  #-- Eliminar los problemas a los que se haya asignado el 1
  ejs = ejec[!(ejec$Prob %in% prob$Idprob[noprobs]),]
  ejs.x = subset(ejs, ejs$Login!='dummy' & ejs$Login!='loli')
  #--- Eliminar problemas equivalentes
  ee <- elimequiv(equiv,ppmod,ejs.x)
  ppmod<- ee[[1]]
  ejs.x<- ee[[2]]
  #-- Añadir el módulo de cada problema en el data.frame ejs.x
  ejs.x<-asimod1(ejs.x)
  #-- Tabla con notas máximas de todos los problemas sin clasificar por
módulos
  notamaxtot = tapply(ejs.x$Nota, list(ejs.x$Login, ejs.x$Prob), max)
  dim(notamaxtot)
  notamaxtot<-as.data.frame(notamaxtot)
    #-- Data.frame de todos los alumnos con las notas medias por modulo
  #-- Asignación de 0 a todos los datos NA's para poder calcular la media
  notamaxtot[is.na(notamaxtot)]=0
  #-- Lista con las notas medias de cada alumno en cada modulo
  notamedmod<-list()
  notamedmod[[1]]<-apply(notamaxtot[,ppmod$m1],1,mean)
  notamedmod[[2]]<-apply(notamaxtot[,ppmod$m2],1,mean)
  notamedmod[[3]]<-apply(notamaxtot[,ppmod$m3],1,mean)
  notamedmod[[4]]<-apply(notamaxtot[,ppmod$m4],1,mean)
  notamedmod[[5]]<-apply(notamaxtot[,ppmod$m5],1,mean)
    #-- creación de primera salida Shiny
  output$grafico <- renderPlot({
    #-- Ordenación de los individuos según input
    i<-as.numeric(input$Orden)
    if (i == 6 ){
      notamedmodorden<-notamedmod[[1]]
      notamedmodorden<-as.data.frame(notamedmodorden)
     }
    else {
    notamedmod1=order(notamedmod[[i]],decreasing=TRUE)
    notamedmodorden=notamedmod[[i]][notamedmod1]
    notamedmodorden<-as.data.frame(notamedmodorden)
    }
    #-- Selección de los alumnos de los módulos marcados y cálculo de número
de páginas
    mx = subset (ejs.x, ejs.x$mod %in% as.numeric(input$checkGroup))
    Nal = length(unique(mx$Login))
    Ninput = input$N
    K = ceiling(length(rownames(notamedmodorden))/Ninput)
    output$numpg = renderText({
      1+min(K-1, max(0, input$mas-input$menos))
    })
    pagina = min(K-1, max(0, input$mas-input$menos))
    #-- Salida shiny
    output$pagina <- renderPrint({ return(pagina) })
    #-- Dibujar los módulos seleccionados
    notamaxprob <- list()
    par(mar=c(1,8,0,0))
    plot(c(0,260), c(260, 0), t='n', axes=FALSE, xlab='', ylab='')
```

```
    for (j in as.numeric(input$checkGroup)) {
    #-- Selección de los alumnos que están en la página actual
    select = subset (ejs.x, ejs.x$mod==j)
    #-- Notas máximas de los alumnos de los problemas del modulo j
    notamaxprob[[j]]= tapply(select$Nota, list(select$Login, select$Prob),
max)
    #-- Creación de MX que contiene las proporciones de problemas aprobados,
suspendidos y no realizados
    M=input$N
    MX = notamaxprob[[j]]
    MX[MX<5]=1
    MX[MX>=5]=0
    MX[is.na(MX)]=2
    #-- Selección de alumnos de la lista ordenada que entran en la página
seleccionada
    listanombres<-
rownames(notamedmodorden)[(Ninput*pagina+1):(Ninput*(pagina+1))]
    #-- Print gráfico
    w = (alto-5*Sx)/5
    #N=length(rownames(MX))
    h = (ancho-((Ninput-1)*Sy))/Ninput
    fr = array(NA, dim=c(3,5))
    colores = c('green', 'orange', 'beige')
    for (k in 1:M) {
    a = listanombres[k]
    print(a )
    if(!(a  %in% rownames(MX))){next}
    y = alto-(k-1)*(h+Sy)
    x = (j-1)*(w+Sx)
    rect(x, y-h, x+w, y, col="White")
    t = table(factor(MX[a,], levels=0:2))
    fr[,j] = t/length(ppmod[[j]])
    print(fr)
    for (q in 1:2){
    rect(x,y-h,x+w*fr[q,j],y,col=colores[q],border=NA)
    x=x+w*fr[q,j]
    }
    x = x+Sx
    par(xpd=NA)
    text(0, alto-((1:Ninput)-0.5)*(h+Sy), listanombres, cex=1, pos=2 , adj=1)
    par(xpd=FALSE)
    }
  }
},height = 500)
##-- Salida shiny. Pintar las tablas de notas máximas
  notamaxprob1<-list()
  notamaxtot[is.na(notamaxtot)]=0
  notamaxprob1[[1]]<-notamaxtot[,ppmod$m1]
  notamaxprob1[[2]]<-notamaxtot[,ppmod$m2]
  notamaxprob1[[3]]<-notamaxtot[,ppmod$m3]
  notamaxprob1[[4]]<-notamaxtot[,ppmod$m4]
  notamaxprob1[[5]]<-notamaxtot[,ppmod$m5]
  output$tabla1 <- renderDataTable(
    return(cbind(Alumno=rownames( notamaxprob1[[1]]), notamaxprob1[[1]])
    )
  )
  output$tabla2 <- renderDataTable(
```

```
    return(cbind(Alumno=rownames( notamaxprob1[[2]]), notamaxprob1[[2]])
    )
  )
  output$tabla3 <- renderDataTable(
    return(cbind(Alumno=rownames( notamaxprob1[[3]]), notamaxprob1[[3]])
    )
  )
  output$tabla4 <- renderDataTable(
    return(cbind(Alumno=rownames( notamaxprob1[[4]]), notamaxprob1[[4]])
      )
  )
  output$tabla5 <- renderDataTable(
    return(cbind(Alumno=rownames( notamaxprob1[[5]]), notamaxprob1[[5]])
      )
   )
})
```

UI

```
library(shiny)
shinyUI(fluidPage(titlePanel("Análisis gráfico por módulos"),
  fluidRow(
    column(3,
           actionButton("menos", "", icon=icon("arrow-left")),
           span(textOutput("numpg", inline=TRUE), style="font-size: 20px;"),
           actionButton("mas", "", icon=icon("arrow-right")),
           sliderInput("N", "Número de alumnos",
                       min = 1, max = 30, value = 15),
           selectInput("Orden", label = "Orden",
                       choices = list("M1" = 1, "M2" = 2,"M3" = 3, "M4" =
4,"M5" = 5, "Alfabetico"= 6), selected = 1),
           checkboxGroupInput("checkGroup", label = h3("Módulos"),
                              choices = list("M1" = 1, "M2" = 2,"M3" = 3, "M4"
= 4,"M5" = 5),selected = 1)
           ),
    column(9,
           mainPanel(
             tabsetPanel(
               tabPanel("Grafico", plotOutput("grafico",height = "2500px")),
               tabPanel("Tabla Notas máx. M1", dataTableOutput('tabla1')),
               tabPanel("Tabla Notas máx. M2", dataTableOutput('tabla2')),
               tabPanel("Tabla Notas máx. M3", dataTableOutput('tabla3')),
               tabPanel("Tabla Notas máx. M4", dataTableOutput('tabla4')),
               tabPanel("Tabla Notas máx. M5", dataTableOutput('tabla5')),
               tabPanel("Página actual", verbatimTextOutput("pagina"))
             )
           )
    )
  )
))
```

VI.4.4. Aplicativo Shiny 2 (Flujo alumnos)

Server

```
shinyServer(function(input, output) {
  #-- Carga de valores y funciones
  source("parameters.R")
  source("functions12.R")
  #-- Carga de datos
  load(url("http://ka.upc.es/test-r/asignaturas.Rdata"))
  load(url("http://ka.upc.es/test-r/ejecuciones.Rdata"))
  load(url("http://ka.upc.es/test-r/numalumnos.Rdata"))
  load(url("http://ka.upc.es/test-r/problemas.Rdata"))
  #-- Anadir el módulo de cada problema en el data.frame prob
  prob<-asimod(prob)
  #-- Identificar los problemas de cada módulo, sin contar no puntuables
  F = iconv(prob$Nombre, from="", to="UTF-8", "byte")
  #-- Eliminar los problemas de dedicación y de Mis notas
  noprobs = c(grep("^Dedica", F), grep("Mis notas", F))
  #-- Se asigna un 1 a los problemas no puntuables
  prob$NO[noprobs] = 1
  ppmod<-pmod(prob)
  #-- Eliminar los problemas a los que se haya asignado el 1
  ejs = ejec[!(ejec$Prob %in% prob$Idprob[noprobs]),]
  ejs.x = subset(ejs, ejs$Login!='dummy' & ejs$Login!='loli')
  #--- Eliminar problemas equivalentes
  ee <- elimequiv(equiv,ppmod,ejs.x)
  ppmod<- ee[[1]]
  ejs.x<- ee[[2]]
  #-- Añadir el módulo de cada problema en el data.frame ejs.x
  ejs.x<-asimod1(ejs.x)
  #-- Vector nombre filas
  Modulo<-c("M1","M2","M3","M4","M5")

output$tabla<-renderDataTable({
#-- Selección alumnos que han iniciado el modulo desde el inicio del curso
ejs.f0 <- ejs.x[ejs.x$Fecha>="2012-09-01 00:00:00" , ]
notamaxtot = tapply(ejs.f0$Nota, list(ejs.f0$Login, ejs.f0$Prob), max)
notamedmod<-lista(ejs.f0,ppmod)
#-- Número de alumnos
N<-length(rownames(notamaxtot))
tabla=array(NA, dim=c(5,3))
#-- Se considera que el módulo se ha iniciado cuando el alumno ha realizado
una ejecución
#-- Calculo de la tabla a mostrar (col 1)
for (j in 1:NMODS) {
  sumf0=0
  notamedmod[[j]]<-as.data.frame(notamedmod[[j]])
    for(k in 1:N){
      if(notamedmod[[j]][k,] != 0){
      sumf0=sumf0+1
      }
    }
 tabla[j,1]<-sumf0
}

#-- Selección y cálculo de alumnos primera fecha
```

```
ejs.f1<- ejs.x[ejs.x$Fecha<=input$dateRange[1] , ]
#-- Selección notas máximas
notamaxtot = tapply(ejs.f1$Nota, list(ejs.f1$Login, ejs.f1$Prob), max)
notamedmod<-lista(ejs.f1,ppmod)
#-- Calculo de la tabla a mostrar (col 2)
N<-length(rownames(notamaxtot))
for (j in 1:NMODS){
    sumf1=0
  notamedmod[[j]]<-as.data.frame(notamedmod[[j]])
  for(k in 1:N){
    if(notamedmod[[j]][k,]>=5){
      sumf1=sumf1+1
    }
  }
  tabla[j,2]<-sumf1
}
#-- Cálculo de alumnos aprobados por módulo
#-- Selección y cálculo de alumnos segunda fecha
ejs.f2<- ejs.x[ejs.x$Fecha<=input$dateRange[2], ]
notamaxtot = tapply(ejs.f2$Nota, list(ejs.f2$Login, ejs.f2$Prob), max)
notamedmod<-lista(ejs.f2,ppmod)
N<-length(rownames(notamaxtot))
#-- Cálculo de la tabla a mostrar (col 3)
for (j in 1:NMODS){
  sumf2=0
  notamedmod[[j]]<-as.data.frame(notamedmod[[j]])
  for(k in 1:N){
    if(notamedmod[[j]][k,]>=5){
      sumf2=sumf2+1
    }
  }
  tabla[j,3]<-sumf2
}
tabla<-as.table(tabla)
colnames(tabla)<-c("Número de alumnos", "Número de alumnos aprobados hasta la
primera fecha","Número de alumnos aprobados hasta la segunda fecha")
return(cbind(Modulo, tabla))
})
#-- Segunda tabla
output$tabla1<-renderDataTable({
#-- Ejecuciones desde el inicio del curso
tabla1=array(NA, dim=c(5,5))
ejs.f01 <- ejs.x[ejs.x$Fecha>="2012-09-01 00:00:00" , ]
#-- Cálculo de notas máximas
notamaxtot = tapply(ejs.f01$Nota, list(ejs.f01$Login, ejs.f01$Prob), max)
notamedmod<-lista(ejs.f01,ppmod)
#-- Número de alumnos
N<-length(rownames(notamaxtot))
tabla11=array(NA, dim=c(5,1))
#-- Tabla primera columna
for (j in 1:NMODS) {
  sumf01=0
  notamedmod[[j]]<-as.data.frame(notamedmod[[j]])
  for(k in 1:N){
    if(notamedmod[[j]][k,] != 0){
      sumf01=sumf01+1
    }
```

```
  }
  tabla11[j,1]<-sumf01
}
#-- Cálculo del número total de ejecuciones que deberían haber
for (j in 1:NMODS){
  tabla1[j,1]<-length(ppmod[[j]])*tabla11[j,1]
}
#-- Cálculo del número de problemas realizados (teniendo en cuenta todos los
alumnos) que hay hasta cierta fecha
ejs.f1<- ejs.x[ejs.x$Fecha<=input$dateRange[1], ]
notamaxtot = tapply(ejs.f1$Nota, list(ejs.f1$Login, ejs.f1$Prob), max)
numejsmod<-lista1(ejs.f1,ppmod)
#--Tabla segunda y tercera columna
for (j in 1:NMODS){
  tabla1[j,2]<-sum(numejsmod[[j]])
  tabla1[j,3]<-round((sum(numejsmod[[j]])/tabla1[j,1])*100,2)
}
#-- Cálculo del número de ejecuciones que hay hasta cierta fecha+1
ejs.f2<- ejs.x[ejs.x$Fecha<=input$dateRange[2], ]
notamaxtot = tapply(ejs.f2$Nota, list(ejs.f2$Login, ejs.f2$Prob), max)
numejsmod1<-lista1(ejs.f2,ppmod)
#-- Tabla cuarta y quinta columna
for (j in 1:NMODS){
  tabla1[j,4]<-sum(numejsmod1[[j]])
  tabla1[j,5]<-round((sum(numejsmod1[[j]])/tabla1[j,1])*100,2)
}
tabla1<-as.table(tabla1)
colnames(tabla1)<-c("Total de ejecuciones","Ejecuciones realizadas hasta la
primera fecha (n)", "Ejecuciones realizadas hasta la primera fecha
(%)","Ejecuciones realizadas hasta la segunda fecha (n)", "Ejecuciones
realizadas hasta la segunda fecha (%)")
return(cbind(Modulo, tabla1))
})
#-- Tercera pestaña ui
output$tabla2<-renderDataTable({
  ejsf1f2<- ejs.x[ejs.x$Fecha>=input$dateRange[1] &
ejs.x$Fecha<=input$dateRange[2],] #Input fechas date2
  nejsal1<-table(ejsf1f2$Login,ejsf1f2$mod)
  Total<-table(ejsf1f2$Login)
  nejsal<-cbind(nejsal1,Total)
  colnames(nejsal)<-c(Modulo,"Total")
  return(data.frame(Alumno=rownames(nejsal), nejsal))
})
})
```

UI

```
library(shiny)
shinyUI(fluidPage(titlePanel("Flujo Alumnos"),
  fluidRow(
    column(3,
  dateRangeInput("dateRange", label = h3("Fechas"), start = "2014-09-01",
                 end   = "2015-09-01")
    ),
    column(9,
        mainPanel(
          tabsetPanel(
```

```
              tabPanel("Número de alumnos aprobados",
dataTableOutput("tabla")),
              tabPanel("Número de ejecuciones total",
dataTableOutput("tabla1")),
              tabPanel("Número de ejecuciones por alumno",
dataTableOutput("tabla2"))
               )
             )
           )
         )
))
```

VI.4.5. Aplicativo Shiny 3 (Análisis ejercicios)

Server

```
shinyServer(function(input, output) {
  #-- Carga de valores y funciones
  source("parameters.R")
  source("functions12.R")
  #-- Carga de datos
  load(url("http://ka.upc.es/test-r/asignaturas.Rdata"))
  load(url("http://ka.upc.es/test-r/ejecuciones.Rdata"))
  load(url("http://ka.upc.es/test-r/numalumnos.Rdata"))
  load(url("http://ka.upc.es/test-r/problemas.Rdata"))
  #-- Anadir el módulo de cada problema en el data.frame prob
  prob<-asimod(prob)
  #-- Identificar los problemas de cada módulo, sin contar no puntuables
  F = iconv(prob$Nombre, from="", to="UTF-8", "byte")
  #-- Eliminar los problemas de dedicación y de Mis notas
  noprobs = c(grep("^Dedica", F), grep("Mis notas", F))
  #-- Se asigna un 1 a los problemas no puntuables
  prob$NO[noprobs] = 1
  ppmod<-pmod(prob)
  #-- Eliminar los problemas a los que se haya asignado el 1
  ejs = ejec[!(ejec$Prob %in% prob$Idprob[noprobs]),]
  ejs.x = subset(ejs, ejs$Login!='dummy' & ejs$Login!='loli')
  #--- Eliminar problemas equivalentes
  ee <- elimequiv(equiv,ppmod,ejs.x)
  ppmod<- ee[[1]]
  ejs.x<- ee[[2]]
  #-- Añadir el módulo de cada problema en el data.frame ejs.x
  ejs.x<-asimod1(ejs.x)
output$grafico <- renderPlot({
j<-as.numeric(input$modulo)
#-- Selección ejecuciones del módulo
modu<-subset(ejs.x, ejs.x$mod==j & !(is.na(ejs.x$Nota)))
#Notas máximas
notamaxprob = tapply(modu$Nota, list(modu$Login, modu$Prob), max)
notamaxprob[is.na(notamaxprob)]=0
#-- Cálculo IC's
int.conf = apply(notamaxprob,2,IC)
Np<-dim(notamaxprob)[2]
#-- Print Forest plot
plot(range(int.conf), c(0, 1+Np),
     type = "n", xlab = "Nota",main="IC95% de la nota media del problema",
```

```
     ylab = "",bty="n",yaxt="n",xlim=c(0,10),ylim=c(0,16))
for (i in 1:Np) {
  lines(int.conf[, i], rep(i,2),
        lwd=2)
}
mtext(ppmod[[j]],2,at=1:Np,line=1,adj=1,las=1)
abline(v = 5, lwd = 2, lty = 2)
},width = 500, height = 500)
output$tabla<-renderDataTable({
  j=as.numeric(input$modulo)
  #-- Selección ejecuciones del módulo
  modu<-subset(ejs.x, ejs.x$mod==j & !(is.na(ejs.x$Nota)))
  numejsprob<-table(modu$Prob)
  #-- Notas máximas
  notamaxprob = tapply(modu$Nota, list(modu$Login, modu$Prob), max)
  #-- Cálculo columnas tabla
  nas<-apply(is.na(notamaxprob),2,sum)
  nas<-as.vector(nas)
  notamaxprob[is.na(notamaxprob)]=0
  notamaxprob1<-round(apply(notamaxprob,2,mean),2)
  sdnotaprob<-round(apply(notamaxprob,2,sd),2)
  #-- Asignación columnas tabla
  tabla<-matrix(NA, nrow=dim(notamaxprob)[2],ncol=4)
  tabla[,1]<-numejsprob
  tabla[,2]<-notamaxprob1
  tabla[,3]<- sdnotaprob
  tabla[,4]<-nas
  tabla <- cbind(ppmod[[j]],tabla)
  tabla1<-as.data.frame(tabla)
  colnames(tabla1)<-c("Identificador del ejercicio", "Número de ejecuciones",
"Nota media", "Desviación", "Núm. de ejecuciones sin finalizar/ Núm. de
alumnos que no han realizado el ejercicio")
  return(tabla1)
  })
output$tabla1<-renderDataTable({
  j<-as.numeric(input$modulo)
  #-- Selección ejecuciones módulo
  modu<-subset(ejs.x, ejs.x$mod==j & !(is.na(ejs.x$Nota)))
  ppmod1<-vector()
  Np<- length(ppmod[[j]])
    #-- Nombres originales problemas
  for(i in 1:Np){
    ppmod1[i]<-unique(prob$Nombre[prob$Idprob==ppmod[[j]][i]])
  }
  tablaident<-cbind(ppmod[[j]],ppmod1)
  colnames(tablaident)<-c("Identificador del ejercicio", "Nombre del
ejercicio")
  return(tablaident)
})
})
```

UI

```
shinyUI(fluidPage(titlePanel("Análisis Ejercicios"),
  fluidRow(
    column(3,
           selectInput("modulo", label = "Módulo",
```

```
                        choices = list("M1" = 1, "M2" = 2,"M3" = 3, "M4" = 4,
                                       "M5" = 5), selected = 1)
    ),
    column(9,
           mainPanel(
             tabsetPanel(
               tabPanel("Forest Plot Nota media ejercicios",
plotOutput("grafico",height = "2500px")),
               tabPanel("Descriptiva ejercicios", dataTableOutput('tabla')),
               tabPanel("Correspondencia ejercicio-identificador",
dataTableOutput('tabla1'))
             )
           )
    )
  )
))
```

VI.4.6. Aplicativo Shiny 4 (Análisis Bland Altman)

Server

```
library(shiny)
shinyServer(function(input, output) {
  output$ui <- renderUI({
     switch(input$modulo,
           "1" =  selectInput("problema", label = "Problemas M1",
                              choices = list("Test RG1" = 1, "Test RG2" =
2,"Test RG3" = 3,"Test RG4" = 4,"Test RG5" = 5,"Test Principios Generales"= 6,
"Test Tipos de Estudios"=7, "Test Principios Estadísticos"=8,
"Test Medida"=9,"Test Descriptiva"=10,"Test R1"=11,"Test R2"=12,
"Test R3"=13, "Peak Expiratory Flow"=14,"ALT/GPT"=15), selected = 1),
           "2" =  selectInput("problema", label = "Problemas M2",
                              choices = list("Capacidad diagnóstica" = 1,
"Test RG6" = 2, "Monitor de signos vitales" = 3, "Tiempo en UCI" =
4,"Velocidad de flujo" = 5, "Test Probabilidad"= 6, "Test Aplicaciones al
diagnóstico"=7,"Test v.a. discretas"=8," Distribución Binomial"=9,
"Alergias"=10," Test RG7"=11,"Test RG8"=12,"Test Medidas de frecuencia y
asociación"=13,"Test v.a. continuas 1"=14,"Test v.a. continuas 2"=15,
"Riesgos" = 16), selected = 1),
           "3" =  selectInput("problema", label = "Problemas M3",
                              choices = list("Error Estándar" = 1, "Adicción
al chocolate" = 2, "Baterías" = 3, "Comparación de presión arterial" = 4,
"Glicemia" = 5,"Test Inferencia y decisión"= 6, "Test Intervalos de
confianza"= 7, "Test Prueba de significación y contraste de hipótesis"=8,"Test
RG9"=9,"Test RG10 "=10,"Test RG11"=11), selected = 1),
           "4" = selectInput("problema", label = "Problemas M4",
                             choices = list("Cuántos pacientes" = 1,
"Enfermedad Celíaca" = 2," Test RG13 " = 3,"Test RG14" = 4,"Test RG12 " =
5,"Test Ensayo Clínico"= 6,"Test Tamaño muestral"=7, "Comparación de
medias"=8,"Comparacion de proporciones "=9,"Efecto y cumplimiento del
protocolo"=10,"Análisis de supervivencia"=11,"Test Efecto"=12,"Prevención de
accidentes cerebrovasculares"=13),selected = 1),
           "5" =  selectInput("problema", label = "Problemas M5",
                              choices = list(" Un ensayo comunitario" = 1, "
Test Diseños que afectan a la varianza" = 2,"Disenyo con intercambio " = 3,
"Diseño Paralelo " = 4, "Test RG15" = 5,"Meta-analisis con respuesta
```

```
dicotómica"= 6, "Ensayo secuencial"=7, "Meta-análisis con respuesta
cuantitativa"=8," Test RG15b "=9, "Test Control del riesgo alfa"=10,"Test
Meta-análisis"=11,"Cuidando las coronárias"=12), selected = 1)
    )
  })
    #-- Carga de valores y funciones
    source("parameters.R")
    source("functions12.R")
    #-- Carga de datos
    load(url("http://ka.upc.es/test-r/asignaturas.Rdata"))
    load(url("http://ka.upc.es/test-r/ejecuciones.Rdata"))
    load(url("http://ka.upc.es/test-r/numalumnos.Rdata"))
    load(url("http://ka.upc.es/test-r/problemas.Rdata"))
    #-- Anadir el módulo de cada problema en el data.frame prob
    prob<-asimod(prob)
    #-- Identificar los problemas de cada módulo, sin contar no puntuables
    F = iconv(prob$Nombre, from="", to="UTF-8", "byte")
    #-- Eliminar los problemas de dedicación y de Mis notas
    noprobs = c(grep("^Dedica", F), grep("Mis notas", F))
    #-- Se asigna un 1 a los problemas no puntuables
    prob$NO[noprobs] = 1
    ppmod<-pmod(prob)
    #-- Eliminar los problemas a los que se haya asignado el 1
    ejs = ejec[!(ejec$Prob %in% prob$Idprob[noprobs]),]
    ejs.x = subset(ejs, ejs$Login!='dummy' & ejs$Login!='loli')
    #--- Eliminar problemas equivalentes
    ee <- elimequiv(equiv,ppmod,ejs.x)
    ppmod<- ee[[1]]
    ejs.x<- ee[[2]]
    #-- Añadir el módulo de cada problema en el data.frame ejs.x
    ejs.x<-asimod1(ejs.x)
output$grafico <- renderPlot({
  i=as.numeric(input$problema)
  j<-as.numeric(input$modulo)
  #-- Selección ejecuciones módulo
  modu<-subset(ejs.x, ejs.x$mod==j & !(is.na(ejs.x$Nota)))
  #-- Notas máximas
  notamaxprob = tapply(modu$Nota, list(modu$Login, modu$Prob), max)
  notamaxprob[is.na(notamaxprob)]=0
  #--Cálculo nota media módulo
  med<-apply(notamaxprob,1,mean)
  med<-as.data.frame(med)
  modu1<-cbind(notamaxprob,med$med)
  #--Selección alumnos con módulo aprobado
  modu1<-subset(modu1,modu1[,length(ppmod[[j]])+1]>=5)
  N=length(rownames(modu1))
  P=length(ppmod[[j]])
  x<-vector()
  y<-vector()
  par(mfrow=c(1,1))
  plot(c(0,11), c(11, 0), t='n', axes=FALSE, xlab='', ylab='')
  posmed<-length(ppmod[[j]])+1
  for (k in 1:N){
  y[k]<-modu1[k,i]
  al<-modu1[k,]
  #-- Cálculo nota media sin problema con el que se trabaja
  x[k]<-round(mean(al[-c(i,posmed)]),2)
```

```
  }
  a<-modu1[,i]
  b<-x
  par(mfrow=c(1,2))
  tit<-unique(prob$Nombre[prob$Idprob==ppmod[[j]][i]])
  #-- Gráfico regresión
  plot(y~x,main=tit,xlab='Nota media del módulo sin problema seleccionado',
ylab='Problema seleccionado')
  abline(0,1,col="red")
  #-- Gráfico Bland Atman
  BlandAltman (a,b,unique(prob$Nombre[prob$Idprob==ppmod[[j]][i]]))
},width = 1000, height = 500)
output$grafico3 <- renderPlot({
  i=as.numeric(input$problema)
  j<-as.numeric(input$modulo)
  modu<-subset(ejs.x, ejs.x$mod==j & !(is.na(ejs.x$Nota)))
  notamaxprob = tapply(modu$Nota, list(modu$Login, modu$Prob), max)
  notamaxprob[is.na(notamaxprob)]=0
  med<-apply(notamaxprob,1,mean)
  med<-as.data.frame(med)
  modu1<-cbind(notamaxprob,med$med)
  modu1<-subset(modu1,modu1[,length(ppmod[[j]])+1]>=5)
  N=length(rownames(modu1))
  posmed<-length(ppmod[[j]])+1
  x<-vector()
  Np<- dim(modu1)[2]-1
  difs<-array(NA, dim=c(length(rownames(modu1)),dim(modu1)[2]-1))
  for (i in 1:Np){
  for (k in 1:N){
    al<-modu1[k,] #probelmas modulo
    x[k]<-round(mean(al[-c(i,posmed)]),2)
  }
  a<-modu1[,i]
  a<-as.vector(a)
  #-- Para dibujar el forest plot de las diferencias
  difs[,i]<-a-x
  }
  #-- Cálculo IC diferencias
  int.conf = apply(difs,2,IC)
  #-- Print forest plot
  plot(range(int.conf), c(0, 1+Np),
       type = "n", xlab = "Diferencia",main="IC95% diferencias ",
       ylab = "",bty="n",yaxt="n")
  for (i in 1:Np) {
    lines(int.conf[, i], rep(i,2),
          lwd=2)
  }
  mtext(ppmod[[j]],2,at=1:Np,line=1,adj=1,las=1)
  abline(v = 0, lwd = 2, lty = 2)
},width = 700, height = 700)
output$tabla1<-renderDataTable({
  j<-as.numeric(input$modulo)
  modu<-subset(ejs.x, ejs.x$mod==j & !(is.na(ejs.x$Nota)))
  ppmod1<-vector()
  Np<- length(ppmod[[j]])
  for(i in 1:Np){
    ppmod1[i]<-unique(prob$Nombre[prob$Idprob==ppmod[[j]][i]])
```

```
  }
  tablaident<-cbind(ppmod[[j]],ppmod1)
  colnames(tablaident)<-c("Identificador del problema", "Nombre del problema")
  return(tablaident)
})
})
```

UI

```
shinyUI(fluidPage(
  titlePanel("Análisis Bland Altman"),
  fluidRow(
    column(3, wellPanel(
      selectInput("modulo", label = "Módulo",
      choices = list("M1" = 1, "M2" = 2,"M3" = 3, "M4" = 4,
             "M5" = 5), selected = 1)
    )),
    column(3, wellPanel(
     uiOutput("ui")
    )),
    column(9,
           mainPanel(
             tabsetPanel(
              tabPanel("Gráfico Bland Altman y recta identidad",
plotOutput("grafico",height = "2500px")),
               tabPanel("Forest plot IC diferencias",
plotOutput("grafico3",height = "2500px")),
               tabPanel("Correspondencia problema-identificador",
dataTableOutput('tabla1'))
             )
           )
    )
  )
))
```

VI.4.7. Aplicativo Shiny 5 (Análisis de Componentes Principales)

Server

```
library(FactoMineR)
library(shiny)
shinyServer(function(input, output) {
  #-- Carga de valores
  source("parameters.R")
  source("functions12.R")
  #-- Carga de datos
  load(url("http://ka.upc.es/test-r/asignaturas.Rdata"))
  load(url("http://ka.upc.es/test-r/ejecuciones.Rdata"))
  load(url("http://ka.upc.es/test-r/numalumnos.Rdata"))
  load(url("http://ka.upc.es/test-r/problemas.Rdata"))
  #-- Anadir el módulo de cada problema en el data.frame prob
  prob<-asimod(prob)
  #-- Identificar los problemas de cada módulo, sin contar no puntuables
  F = iconv(prob$Nombre, from="", to="UTF-8", "byte")
  #-- Eliminar los problemas de dedicación y de Mis notas
```

```
  noprobs = c(grep("^Dedica", F), grep("Mis notas", F))
  #-- Se asigna un 1 a los problemas no puntuables
  prob$NO[noprobs] = 1
  ppmod<-pmod(prob)
  #-- Eliminar los problemas a los que se haya asignado el 1
  ejs = ejec[!(ejec$Prob %in% prob$Idprob[noprobs]),]
  ejs.x = subset(ejs, ejs$Login!='dummy' & ejs$Login!='loli')
  #--- Eliminar problemas equivalentes
  ee <- elimequiv(equiv,ppmod,ejs.x)
  ppmod<- ee[[1]]
  ejs.x<- ee[[2]]
  #-- Añadir el módulo de cada problema en el data.frame ejs.x
  ejs.x<-asimod1(ejs.x)
 output$grafico4 <- renderPlot({
    var<-input$variable
    j=as.numeric(input$modulo)
    #-- Selección ejecuciones módulo
    modu<-subset(ejs.x, ejs.x$mod==j & !(is.na(ejs.x$Nota)))
    #-- Notas máximas de cada ejercicio
    notamaxprob = tapply(modu$Nota, list(modu$Login, modu$Prob), max)
    notamaxprob[is.na(notamaxprob)]=0
    #-- Clasificación/nomenclatura manual de los problemas
    notamaxprob<-clasprob(ppmod,notamaxprob,j,prob)
    #Print PCA
    PCAres<-PCA(notamaxprob, graph=FALSE)
    if( var == 1){plot(PCAres,as.numeric(input$dim[1:2]),choix =
"ind",label="var")}
    if( var == 2){plot(PCAres,as.numeric(input$dim[1:2]),choix =
"var",label="var")}
    if( var == 3){
      par(mfrow=c(1,2))
      plot(PCAres,as.numeric(input$dim[1:2]),choix = c("ind"),label="var")
      plot(PCAres,as.numeric(input$dim[1:2]),choix = c("var"))
    }
 },width = 900, height = 450)
})
```

UI

```
library(shiny)
shinyUI(fluidPage(
  titlePanel("Análisis de Componentes Principales"),
  fluidRow(
    column(3,
           selectInput("modulo", label = "Módulo",
                       choices = list("M1" = 1, "M2" = 2,"M3" = 3, "M4" = 4,
                                      "M5" = 5), selected = 1),
           selectInput("variable", label = "Variable a mostrar",
                       choices = list("Alumnos" = 1, "Ejercicios" = 2,"Ambas"
= 3), selected = 1),
           checkboxGroupInput("dim", label = h3("Dimensiones"),
                              choices = list("Dim 1" = 1, "Dim 2" = 2,"Dim 3"
= 3, "Dim 4" = 4,"Dim 5" = 5),
                              selected = c(1,2))
    ),
    column(9,
```

```
            mainPanel(
              tabsetPanel(
                tabPanel("PCA", plotOutput("grafico4",height = "2500px"))
              )
            )
    )
  )
))
```

VI.4.8. Accesibilidad de los aplicativos

Los aplicativos están disponibles en las siguiente webs; entre los días día 3 y 10 de Julio los alumnos aparecen de forma anónima, de modo que en lugar de aparecer el identificador de cada alumno, éstos aparecen numerados:

Aplicativo 1 - Análisis gráfico por módulos: http://shiny-eio.upc.edu/nerea/global/

Aplicativo 2 - Flujo alumnos: http://shiny-eio.upc.edu/nerea/Flujo/

Aplicativo 3 - Análisis ejercicios: http://shiny-eio.upc.edu/nerea/Ejercicios/

Aplicativo 4 - Análisis Bland Altman: http://shiny-eio.upc.edu/nerea/BlandAltman/

Aplicativo 5 - Análisis de Componentes Principales: http://shiny-eio.upc.edu/nerea/PCA/