REFACIL

RJose

29 de julio de 2010

Índice general

Prefacio

Esta es una invitación a usar software libre, de calidad profesional y multiplataforma para análisis estadístico. Nuestra propuesta es *Gnumeric* para univariado y *R* para todo pero especialmente para multivariado. Estas aplicaciones representan el fruto del esfuerzo continuado de una comunidad internacional que se ha propuesto ofrecer una alternativa gratis a los paquetes estadísticos profesionales de alta calidad y que pueden ser supremamente costosos. Los productos rivales de *R*, que es para trabajo pesado, pueden ser mejores en cuanto a la comunicación con el usuario, pero *R* les gana a todos en su riqueza de servicios que se puede expandir con más de 2000 paquetes. Para trabajo liviano, lo mejor es utilizar *Gnumeric* que también es gratis y sus autores están empeñados en convertirlo en una herramienta de muy alta calidad.

Tanto *Gnumeric* como *R* vienen sin garantía. Eso quiere decir que pueden tener bugs, errores de programación que causará que algún resultado salga erróneo. Lo que uno tiene que hacer es mirar los datos a ver si la respuesta tiene sentido. O verificar la función deseada probándola con un ejemplo de un libro o que uno ya haya hecho a mano. O por medio de una simulación. Si hay discrepancia, lo más probable para este momento de la historia es que uno descubra que sus cuentas a mano están mal hechas y entonces uno estará muy agradecido, como en mi caso. Pero también es posible encontrar un bug, para lo cual uno lo reporta en el link que a propósito se tiene en la página web oficial de cada aplicación. Así fue como el autor de este material encontró y reportó para *Gnumeric* el bug 614746 sobre la documentación del ZTEST, el cual fue aceptado, y para *R* el bug 14316 sobre regresión polinomial, el cual no fue aceptado.

Este documento es introductorio y está dirigido a quienes ya tienen un buen conocimiento de la estadística y desean adherirse al movimiento libertador del software libre y de alta calidad. Esto es parte de una tendencia más alta que propugna por una ciencia libre y para todos.

José

Capítulo 1

Gnumeric

1 ***Objetivo.*** *Bajar, instalar y correr Gnumeric.*

Para análisis estadístico univariado, usar *R* puede resultar incómodo. Lo mejor es usar un paquete adecuadamente simple y directo. Nuestra propuesta es *Gnumeric*, que es gratis y muy profesional. Sin embargo, no todos comparten la misma idea y hay cursos de estadística básica con *R*. Un ejemplo: *Elementary statistics with R* by Chi Yau (2010)

`http://www.r-tutor.com/`

Debido a que tanto *R* como *Gnumeric* representan un esfuerzo patriótico multinacional para liberarse de las grandes compañías productoras de Software y que pueden cobrar muy caro (varios miles de dólares), *R* y *Gnumeric* combinan bien con *Linux* que es gratis:

1.0.1. Linux a la orden

Un programa que permite que uno use un computador se llama sistema operativo. Por la facilidad de manejo, Windows ha sido el sistema operativo preferido por la mayoría de personas, en tanto que *Linux* fue durante décadas patrimonio exclusivo de gurús muy diestros. Pero éso ha cambiado: si alguien ha deseado ponerle fin a la dulce esclavitud de Microsoft y sus imponentes productos, ahora tiene una opción para hacerlo. Personalmente recomiendo la versión de Linux conocida como *OpenSUSE*, la cual es gratis, y en mi opinión ya es accesible a la gente que desea un sistema operativo gratis, poderoso, fácil de usar para las tareas comunes y corrientes, y además en permanente evolucion. En realidad, hay una fuerte competencia por simplicidad y efectividad en todas las versiones de Linux

y no es raro oír aplausos a una cualquiera. Fedora, Debian, Ubunto, Kubunto son otras versiones de Linux muy famosas.
La forma natural de cambiarse a OpenSUSE es como sigue: uno no borra su sistema antiguo, al cual uno está acostumbrado. Más bien, uno consigue un disco duro y lo dedica para su nuevo sistema. OpenSUSE se baja de la red y se instala tan fácilmente como Windows. Es posible que OpenSUSE ponga problemas con los periféricos, por ejemplo, las impresoras, debido a que los drivers para Linux son más escasos que los que son para Windows. Con todo, la industria ahora es más consiente del poder expansivo de Linux y está haciendo un esfuerzo consistente por producir drivers multiplataforma.

1.0.2. Gnumeric

La versión libre y muy profesional de Excel es *Gnumeric*. Lo primero que hacemos es instalarlo.

2 ***Instalando Gnumeric***

Con la conección a Internet activa, sobre Windows, *Gnumeric* se instala con un click. Sobre OpenSuse Linux, se siguen los siguientes pasos:

En la esquina izquierda-abajo, está el ícono de OpenSuse. Haga click sobre él, vaya a *computer*. Elija *Install Software*. Teclee su clave de Administrador y espere a que la aplicación YAST2 se inicialice. En el buscador escriba

```
Gnumeric
```

Termine con *Enter*. Siga las instrucciones (active algunas casillas donde se explica qué ha de bajarse de la red) y espere a que la aplicación YAST2, el instalador, se cierre por haber concluido su trabajo.

3 ***Corriendo Gnumeric***

Sobre Windows se inicia con un click sobre el ícono correspondiente. Para correr *Gnumeric* en Linux: haga click sobre el *KickOff Application launcher*, esquina inferior izquierda de la pantalla, y elija *terminal*. Sobre la terminal escriba

```
Gnumeric
```

Espere algunos segundos y verá como se despliega la aplicación. *Gnumeric* tiene la imagen y semejanza de *Excel* y ha procurado ser compatible con *Excel* al máximo. Para verificarlo, haga algunas operaciones, una suma, una resta, pinte alguna gráfica. Veamos dos ejemplos más específicos: el cálculo de una media y el de un histograma de frecuencias absolutas.

4 ***Hallemos media la media y la varianza de los datos***
1, 2, 3, 4, 2, 3, 3, 2, 3.

Para usar *Gnumeric*: teclee los datos sobre una columna, vaya al menu *Tools*, luego a *Statistical analysis*, y luego a *Descriptive statistics*. Una ventana de diálogo se abrirá con el cursor titilando sobre una casilla: con el cursor señale la región de la columna que contiene los datos y acepte con *OK*. *Gnumeric* despliega una hoja aparte con muchos resultados, entre los cuales están la media, la varianza y la desviación. Ejemplo: sobre los datos propuestos, *Gnumeric* da una media de 2.55 y una varianza de 0.77. Es buena idea verificar el número de datos, que deben ser 9 y aparecen como *count*.

5 ***Como hacer en Gnumeric un histograma***

Gnumeric calcula los *histogramas* a partir de los datos como sigue: se teclean los datos uno por uno sobre una columna y sin dejar vacíos. Al lado se teclea otra columna que contiene la lista de los cortes de los intervalos que definen los rangos para clasificar los datos. Por ejemplo, si los datos son números que representan las notas de un examen, la columna de datos contiene las calificaciones, y la columna de rangos contiene los números 0,1,2,3,4,5, un número por celda. Luego se llama el menú *Tools* seguido de *Statistical analysis*, luego *Frequency tables* y de nuevo *frequency tables*. Aparece una ventana de diálogo, y sobre la casilla de diálogo *Input range*, uno señala con el cursor la región de la columna que contiene los datos. Después uno señala la tarjeta *Categories* y elije *Category ranges*, luego uno señala la segunda columna, aquella que contiene los bordes de los rangos. Pasa uno a la tarjeta *Graphs and options* y elije *Column chart*. Se termina con *OK*. *Gnumeric* despliega una hoja donde uno encuentra la tabla de frecuencias y el histograma. Es posible que el histograma tape la tabla de frecuencias, por éso hay que moverlo a un lado.

6 ***Gnumeric es suficiente***

Explorando las diversas opciones de *statisitcal analysis*, uno se puede dar cuenta de que *Gnumeric* tiene más que lo que se puede ver en un curso de estadística básico.

1.0.3. Las tablas de OpenOffice y Gnumeric

Las tablas de todos los libros son muy limitadas, por ejemplo, las tablas de la F vienen con significancia 0.01, 0.02, 0.05 y los demás valores se quedan en el misterio. La ventaja inmediata de *Gnumeric* es que nos da valores críticos para

cualquier significancia. Esa virtud se aprende a manejar muy fácilmente y hay varios métodos. El primero es:
Se busca en el menú principal la sección *insert*, luego una va a *function* y allí busca el subconjunto de funciones *statistics* y luego se señala la función deseada que lleva el mismo nombre que la función que proponemos teclear. Puede pasar que al teclear números, el programa no los acepte. El remedio es cambiar de modo al teclado de números, lo cual se hace oprimiendo una vez la tecla *Bloq Num*, en la parte superior derecha de dicho teclado. O también se puede ver qué sirve mejor, si una coma o un punto para señalar los decimales.
También se puede programar Gnumeric al igual que Excel, lo cual se hace casilla por casilla. Se puede usar copy+paste desde este documento. Al hacerlo, *Gnumeric* puede insertar una comilla o un espacio, lo cual se investiga no en la celda donde se teclea sino en el editor de línea, inmediatamente abajo de la barra del menú. Dichas inserciones hay que borrarlas. Los comandos que se explican a continuación son los más elementales y muy útiles:

1. La *z*: la instrucción

   =normsinv(1- 0.04)

   da el *z-crítico* para la significancia $\alpha = 0{,}4$ con la cola superior. En éste caso el *z* crítico es 1.75068607125217.

   La instrucción

   =1-normsdist(zExp)

   da la significancia del *z*-experimental con la cola superior, es decir, la probabilidad de encontrar un *z* mayor que el *z* del experimento. Ejemplo: la significancia de la cola superior del valor $z = 1{,}75068607125217$ es 0.04.

2. La *t* depende de los grados de libertad. Para hallar el *t-crítico* con **dos colas** para la significancia 0.05 y 500 gl se teclea

   =tinv(0.05,500)

   lo cual da 1.96471983746779. Vemos que con 500 gls, la *t* funciona igual que una *z*.

   La instrucción

   =tdist(1.96,50,2)

   da la significancia del *t*-experimental 1.96, con 50 grados de libertad y con 2 colas. *Gnumeric* da como significancia 0.05558087405522, lo cual dice

que con 50 grados de libertad, la desviación de la muestra se puede considerar una magnífica estimación de la desvación poblacional y por tanto, la t da casi igual a la z. La instrucción

=tdist(1.96,50,1)

da la significancia del t-experimental 1.96, con 50 grados de libertad y con la cola superior, que es 0.02779043702761.

3. La F:

   =finv(0.01,7,5)

   da el *F-crítico* con la cola superior para la significancia 0.01 con 7 gl en el numerador y 5 en el denominador. La respuesta es 10.4555108917609.

   La instrucción

   =fdist(10.4,7,5)

   da la significancia con la cola superior del valor $F_{exp} = 10{,}4$ con 7 gl en el numerador y 5 gl en el denominador. La respuesta es 0.01.

4. La Chi-cuadrado χ^2. Para hallar el χ^2-*crítico* derecho con significancia 0.01, 30 gl y la cola superior se teclea

   =chiinv(0.01,30)

   lo cual da 50.892.

   La instrucción

   =chidist(50.892,30)

   da la significancia de la cola superior del valor de χ-experimental igual a 50.892 y 30 grados de libertad, la cual es 0.01.

   Trampa para saber en *Gnumeric* el chi-cuadrado crítico de la cola izquierda: si la significancia asignada a la cola izquierda es 0.025 y los gl son 8, uno teclea

   =chiinv(1-0.025,8)

   y da 2.17973074725265.

1.0.4. Un ejemplo con *Gnumeric*

Decidamos con significancia 0.04 y la cola superior que 2 es la media de los datos siguientes sabiendo que la desviación poblacional es 0.9. Los datos son: 1, 2, 3, 4, 2, 3, 3, 2, 3.

Solución: teclee los datos sobre una columna y active una casilla por ahí cerca en donde *Gnumeric* pondría su respuesta. Es muy buena idea ponerle un título, nombre o label a las respuestas de *Gnumeric*. Para ello, en una casilla cualquiera, ponga el nombre zTestDC que significa test z con dos colas y haga click en la casilla del lado para que sea ahí donde *Gnumeric* ponga su respuesta. Sobre el menú *Insertar*, elija *Function*, luego elija la categoría *statistics* y luego busque *ztest*. Lea el mensaje que dice que se trata de una prueba de dos colas. Termine con OK. Se desplegará una ventana de diálogo que se rellena así: haga click sobre la casilla titulada *ref*, y a continuación señale la región que contiene los datos. Sobre la casilla *x* ponga la media de la hipótesis nula, en este caso, 2. Sobre la casilla marcada con *stddev* ponga el valor de la desviación poblacional, 0.9. Termine con OK.
Gnumeric pondrá su respuesta donde uno le indicó. Este valor no es el *z*-crítico sino que es la significancia con dos colas del *z*-experimental, es decir, la proporción de eventos que son más extremos con dos colas que el *z*-experimetal de nuestros datos.
Estas mismas tablas de la *Z* se pueden usar para comparar proporciones y pares de medias cuando uno conoce las desviaciones poblacionales. Lo que hemos visto para la *z*, se adapta a los demás estadígrafos.
Gnumeric ofrece en su menu *Tools* servicios muy útiles: compara varianzas y puede ejecutar un test de comparación de dos medias en todos los 3 casos, cuando se tienen varianzas iguales, varianzas diferentes y cuando los datos son apareados.

Capítulo 2

R para todos

7 ***Objetivo.*** *Veremos cómo instalar R y como se corre un primer programa.*

2.1. ¿Qué es *R*?

Este proyecto *R* es la versión de software libre de *S-plus*, y cuyo objetivo es producir software de alta calidad para quehaceres científicos centrados en la estadística. El software es gratis y se distribuye sin garantía, pero debido a que tiene usuarios por montones, podemos esperar que está muy depurado. Originalmente se diseñó para ambientes Unix y Linux, dirigidos por instrucciones por consola, pero en la actualidad se trabaja en la versión orientada a ventanas, en la cual todo se hace por medio de un click y, como veremos más luego, ya tenemos versiones muy buenas. Hay versiones para Microsoft Windows, Linux y Mac.

2.2. Instalación

Para instalar *R*, búsquelo en Google o acceda a la siguiente dirección:
http://www.r-project.org/
Allí, busque un link que diga download *R*, después elija un servidor y a continuación una versión apropiada a su sistema, Linux, Windows o Mac. Si tiene problemas instalando una versión para 64 bits, elija una de 32 y vuelva a tratar. Si tiene problemas con la última versión disponible, elija una anterior.

Si su sistema es OpenSUSE Linux:

1. Conéctese a

   http://software.opensuse.org/search

2. Déle al buscador la orden de buscar

   R-base.

3. Después, haga click en un botón que diga

   1-Click Install.

   Siga las instrucciones y de la clave del Administrador cuando el sistema lo requiera.

4. El sistema debe terminar reportando una instalción exitosa.

   El sistema puso en algún lugar un directorio llamado RProject y ahí guardó todo lo referente al paquete.

En el suguiente link hay instrucciones visuales para instalar *R* sobre Windows y también una potente *GUI* (graphic user interfase) para Windows llamada *Tinn-R* y que cada quien puede probar:

`http://bioinformatics.ualr.edu/resources/tutorials/Tinn-R_installation.html`

Nuestras instrucciones que están más abajo, son para otra *GUI* llamada *R Commander*, la cual sirve para todas las plataformas. Se considera que *R Commander* está más depurada que *Tinn-R*, aunque esta última también ha despertado aplausos.

2.3. Iniciando *R*

El paquete *R* necesita un directorio o folder para trabajar. Por favor, primero cree un directorio o folder llamado *RProjectU1* (el sufijo *U1* significa usuario 1), donde pondrá todo lo referente a *R*. A continuación, cree otro directorio dentro de *RProjectU1* y póngale nombre *RWorkU1*.

Para iniciar *R*:

- Sobre Windows, *R* se inicia con un click sobre el ícono de *R*. Debería indicarle a *R* el camino para llegar a *RWorkU1*.
- En Linux, abra una consola o terminal y simplemente teclee *R*.

R saluda ofreciendo sus credenciales y dando algunas indicaciones. Por ejemplo, para salir de *R* se debe teclear q(). Después de saludar, R espera órdenes. Para ello, escribe el símbolo
>
al cual añade un cursor, que puede ser un bloquecito negro. Si el cursor de *R* es un bloquecito sin relleno, uno debe hacer click sobre la ventana que lo contiene para poder activar el editor, y el cursor se rellenaría de negro. Después uno teclea lo que desee.

8 ***Ejercicio*** *Verificar que para salir de R basta teclear*

```
q()
```

Por favor, recuerde que siempre que se le de una orden a R, debe terminarse con

Enter

Por favor, salga de R y vuelva a entrar. Al tratar de salir, R preguntará si desea salvar algún archivo de trabajo. Responda con una n de not. En una futura ocasión, cuando haya trabajado y desea grabar su trabajo, responda y de yes.

9 ***Ejercicio*** *Corremos el demo de gráficas. Para ello, se escribe*

```
demo(graphics)
```

En respuesta, R pone un título y solicita permiso para seguir. Uno lo autoriza. R promete dibujar una gráfica, para lo cual abrirá una ventana, pero quizá no se vea nada: hay que despejar el mundo ara que se vean dos ventanas: la de edición, con el signo >, y la de gráficas. Cuando la ventana de gráficas se ha activado, en la barra inferior aparece su logo y se puede hacer click sobre él para hacerla visible.
R pedirá permiso para una segunda gráfica. Uno accede: para ello, uno hace click sobre la ventana de la terminal o consola, la ventana de R propiamente dicha y oprime Enter. La ventana graficadora se habrá minimizado. Dicha ventana se llama R Graphcis: Device 2. Uno la maximiza haciendo click sobre ella y uno deberá encontrar una gráfica. Así mismo se pueden ver las demás gráficas. Cuando R termine su demo reportará en la ventana de R:

```
par(oldpar)
```

Después de ésto, R estará listo para ejecutar otra orden.

Capítulo 3

Medias, varianzas, gráficas

10 ***Objetivo.*** *Correremos R sobre ejemplos sencillos relacionados con la normal univariada, tablas de contingencia y regresión. Normalmente, éso se hace más fácil con Gnumeric, pero lo hacemos en R para comparar las respuestas de Gnumeric con las que nos den en R y así poder sentirnos seguros.*

3.1. Comunicación con R

Aprendamos como se le hace llegar los datos a *R* y cómo se apodera uno de los resultados.

11 ***Definición local***

Uno puede definir directamente sobre *R* los datos. Hay que tener presente que toda definición se hace mediante la función *c()* que concatena o pone en cadena los datos y con la función $<-$ que asigna un nombre al conjunto de datos. Esta flechita se compone del signo menor que, $<$, seguido de un guión corto, $-$. Ejemplo:

```
#INTRODUCCION DE DATOS
x <- c(1,3.2,4,5,2.3,5,4,3,4,5,7,2)
# listar x
x
x[1] + x[4]
```

El programa anterior genera un vector de datos llamado x y que contiene los datos listados. Para listar un objeto simplemente se teclea su nombre. El comando x[1] + x[4] suma el primer elemento de x con el cuarto. La respuesta debe ser 6.

Debido a que uno se equivoca muy fácilmente al teclear, es más práctico copiar y pegar. Para tal fin, se ha dispuesto un archivo aparte, llamado *comandos*, en formato text, del cual puede copiar y pegar lo que desee. Copie el programa anterior del archivo de comandos al clipboard y péguelo en *R*. Siempre que pegue un programa debe terminar con *enter*.
Verifique que se puede copiar todo el programa completo, pegar a *R* y ejecutar (*R* ejecuta en secuencia). Esto permite hacer librerías personales de comandos en *R*, las cuales se guardan en un archivo de texto y para usarlas se usa copiar y pegar.

Cuando uno copia y pega programas, quizá se tenga un programa hecho para un objeto x pero uno necesita correrlo sobre z. La solución es hacer un clon de z y llamarlo x:

```
#REUSO DE PROGRAMAS
#Tenemos z
z <- c(1,3.2,4,5,2.3)
 #clonamos z sobre x
x<-z
#Programa que procesa x
x[1] + x[4]
```

Este programa, y casi todos, también está en el archivo de comandos. De allí se copia y se pega en la consola de *R*.

12 *Tecleando datos desde el editor*

Para hacerle llegar los datos a *R* también contamos con la siguiente instrucción que abre un *editor de datos*, una hoja de cálculo interna, y asigna un nombre a los datos tecleados. Uno teclea los datos en la hoja de cálculo emergente y para terminar sobre Linux hace click sobre *quit* o en Windows hace click derecho en el mouse y elige *cerrar*. Es posible que la manera de cerrar, la cual graba lo tecleado, cambie ligereramente de acuerdo a la versión.
La instrucción para abrir el editor sobre Linux:

```
#EDITOR DE DATOS SOBRE LINUX
#Abrir el editor (se graba al cerrarlo con quit)
sueldoA <-edit(data.frame())
#Publica sueldoA
sueldoA
```

La instrucción para abrir el editor sobre Windows:

```
#EDITOR DE DATOS SOBRE WINDOWS
#Abrir el editor
#Se graba cerrándolo con click derecho en el mouse + cerrar
sueldoA <-edit(as.data.frame(NULL))
#Publica sueldoA
sueldoA
```

Uno también puede ponerle títulos a cada columna, borrando el encabezado que dice *V*1 y poniendo el nombre deseado. Se puede hacer modificaciones de cualquier casilla pero antes hay que borrar todo lo que allí haya con la tecla *backspace*.
En un primer ejercicio, teclee simplemente una columna de datos. Si más luego quiere teclear datos con muchas columnas, amplie la ventana del editor, para lo cual arrastrar su borde derecho es la mejor opción. También se puede navegar con el tabulador.

13 ***Corregir datos***

Si hemos usado el editor de datos para introducir unos datos y queremos hacer una corrección, podemos usar la siguiente idea:

```
#CORRECION DE DATOS
#Ya se tiene sueldoA
sueldoB <- edit(sueldoA)
t.test(sueldoA,sueldoB)
```

3.2. La media

Cuando uno puede calcular una media con *R*, uno siente que al fín pisó tierra firme.

14 ***La media y varianza de un vector de datos***

Si queremos hallar la *media* de la variable *sueldoA*, que nos da una muestra aleatoria de sueldos de la persona A, usamos:

```
#LA MEDIA
#Programa en dos partes, corra primero una después la otra.
#Parte 1:
sueldoA <- c(1, 3.2, 4, 5, 2.3)
```

```
mean(sueldoA)
summary(sueldoA)
var(sueldoA)
#desviación
sd(sueldoA)
#dibuja sueldoA
plot(sueldoA)
#
#Parte 2
boxplot(sueldoA)
```

Detalle técnico: *boxplot* muestra la media y sus cuartiles vecinos. No muestra el error estándard. Para saber más, teclee

```
#AYUDA
#Teclee q para salir de la ayuda
help(boxplot)
boxplot.stats
```

Detalle técnico: *boxplot* muestra la media y sus cuartiles vecinos. No muestra el error estándard. Para saber más, teclee

```
help(boxplot)
```

Sobre la ayuda se avanza con la tecla para avanzar sobre página. Para salir de la ayuda, teclee *q*. Para saber más sobre *algo* siempre se utiliza el mismo formalismo, invocar

```
help(algo)
```

3.3. Histogramas

Los *histogramas* en *R* son muy naturales. En referencia al siguiente programa, el procedimiento *seq* tiene 3 parámetros: donde se empieza el histograma, donde se termina y el ancho de cada rango. El parámetro *prob* dice si la envolvente se hace en frecuencia absoluta o en relativa: puede tomar dos valores T y F. Uno cambia del uno al otro y se queda con el que más le guste. El comando *lines* presenta una curva que se ajusta al histograma. El grado de suavización se ajusta con *bw* (bandwith): si uno quiere una curva con muchas arrugas, debe ser pequeño, pero si quiere una envolvente muy suavizada, debe ser grande.

```
#HISTOGRAMA
x<-c(1,2,1,3,2,3,2,4,3,2,4,2,2,1)
hist(x, seq(0.5, 4.5, 1), plot = TRUE, prob=T)
#Envolvente del histograma
lines(density(x, bw=1))
#Densitómetro de barras
#rug= representación unidimensional gráfica
rug(x, side=1)
```

Para utilizar un histograma en un documento sobre Windows: haga click derecho sobre la ventana de la gráfica, elija la opción de copiar al clipboard y péguela en el documento. Hay dos opciones, mire a ver cuál le gusta más. Sobre Linux es más complicado, al menos por ahora, y se explicará más abajo.

15 ***Un test con la t***

Supongamos que queremos poner a prueba la idea de que el sueldo promedio de A es 1.2. Para ello se usa un *test-t* y se teclea

```
#TEST HO SOBRE LA MEDIA
sueldoA <- c(1,3.2,4,5,2.3)
t.test(sueldoA,mu=1.2)
```

Para uno aceptar o rechazar la hipótesis nula, de que la media es 1.2, uno mira el *p-value*. Este valor da la probabilidad de encontrar por azar datos más extremos que los nuestros. Si es más grande que 0.05, uno la acepta, pero si es menor que 0.05, *R* le está diciendo a uno que es muy improbable encontrar por azar datos más extremos que los nuestros, con respecto a la H_o estudiada, y que sería mejor buscarse otra explicación, por lo cual uno podría rechazar la H_o.

3.4. Comparación de medias

Quisieramos hacer una *comparación de medias* con un *test t* entre dos columnas de datos. Si de antemano se sabe que las varianzas de las poblaciones de las cuales provienen los datos son iguales, para comparar medias poblacionales usamos

```
t.test(sueldoA,sueldoC, var.equal = T)
```

pero si dichas varianzas son diferentes, se puede especificar

```
t.test(sueldoA,sueldoC, var.equal = F)
```

Si los datos son apareados:

```
t.test(sueldoA,sueldoC, paired = T)
```

Si uno desea cambiar el nivel de confianza o hacer varias especificaciones:

```
t.test(sueldoA,sueldoC,conf.level = 0.99)
t.test(sueldoA,sueldoB,paired = T, conf.level = 0.99)
```

Si uno tiene más preguntas, uno puede pedir ayuda a ver qué logra:

```
help(t.test)
```

Para salir de la ayuda, teclear q de quit, que significa salir.

El siguiente programa tiene todo y está en el archivo de comandos:

```
#PRUEBA t PARA COMPARAR MEDIAS
sueldoA <- c(1, 3.2, 4, 5, 2.3)
t.test(sueldoA,mu=1.2)
sueldoB <- c(2.1, 1.3, 3.1, 2, 1.5)
t.test(sueldoA,sueldoB)
sueldoC <- c(2, 2.1, 3.1, 1.2, 1.5, 2.3, 1.7, 1.8)
t.test(sueldoA,sueldoC, var.equal = T)
t.test(sueldoA,sueldoC, var.equal = F)
t.test(sueldoA,sueldoC,conf.level = 0.99)
#Para un test de datos apareados, mismo número de elementos
t.test(sueldoA,sueldoC, paired = T)
t.test(sueldoA,sueldoB,paired = T, conf.level = 0.99)
#Avance sobre la ayuda con el navegador de páginas
#en el teclado.
#Para salir de la ayuda, teclee q
help(t.test)
```

16 *Test de varianzas*

¿Cómo se sabe que las varianzas poblacionales son iguales o diferentes? Por medio de un *test de comparación varianzas*:

```
#COMPARACION DE VARIANZAS
sueldoA <- c(1, 3.2, 4, 5, 2.3)
sueldoB <- c(2.1, 1.3, 3.1, 2, 1.5)
var.test(sueldoA, sueldoB)
var.test(sueldoA,sueldoB, conf.level = 0.99)
```

17 *Aumentando el rigor*

Los ejercicios sobre comparación de medias y varianzas entre dos columnas de datos se hacen más fácil en *Gnumeric* que nos permite comparar varianzas y diferenciar entre datos independientes y apareados. Pero si queremos aumentarle rigor a nuestros análisis, *R* puede comenzar a parecer interesante. Por ejemplo, para poder aplicar un test *t* debemos verificar que los datos vienen de una distribución normal. Para un *test de normalidad* es mejor tener bastantes datos. De momento, lo mejor es generarlos usando un generador interno. Usamos el *test de Kolmogorov-Smirnov*. Todo queda como sigue:

```
#TEST KOLMOGOROV SMIRNOV SOBRE NORMALIDAD
require(grDevices);
require(graphics)
#Parte 1: Esto se corre primero
#Generamos 500 observaciones al azar de la normal,media = 40, sigma = 3.
w<-rnorm(500, mean = 40, sd = 3)
#Histograma de w
hist(w, seq(28, 52, 1), prob=T)
#Envolvente del histograma de datos
lines(density(w, bw=1))
#Envolvente según la Ho: función densidad de la normal, media = 40, sigma = 3.
#add = TRUE significa que la curva se añade a la gráfica anterior.
#add = FALSE significa que se crea una gráfica nueva.
curve((1/(2*pi* 9)^0.5) * (exp( -(x-40)^2/(2*3^2)) ) , 28,52, add = TRUE, col = "red")
#Densitómetro de barras
rug(w, side=1)
#
#Parte 2: Esto se corre (copia y pega)  después
#
#La gráfica K-S
qqnorm(w)
#La Ho (normalidad) daría una línea.
qqline(w)
#Test de Kolgomogorov - Smirnov para normalidad
ks.test(w, "pnorm", mean=40, sd=3)
```

Tengamos presente que sobre el dibujo, uno puede percibir anormalidad si los datos se desvían de la línea recta dibujada por el procedimiento *qqline()*. Esta metodología, de hacer muchos dibujos, es inmanente y necesaria a *R* y a todo investigador: en proyectos complejos está por descontado que uno puede equivocarse y por éso se requiren medios expeditos de retroalimentación. Nuestra manera predilecta será hacer gráficas y digerirlas al máximo: *R* se hizo precisamente para ésto, para que uno pueda estar seguro de lo que cree por estar apoyado en una base amplia de factores diversos. Pensando en ello, deberíamos hacer algo para mejorar nuestro programa anterior: hagamos el histograma y la envolvente de los datos según la H_o, la hipótesis de normalidad. Tengamos en cuenta que *la fórmula de la densidad de la normal* con media μ y desviación σ es:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

El contraste entre lo que se cree en la H_o, la normalidad, y lo que se ve en los datos se mide por un estadígrado llamado D y su significancia con dos colas es reportada como el p-value (la significancia es con dos colas porque D involucra un valor absoluto, que no distingue si hay exceso o hay defecto). El *p-value* es la significancia de nuestros datos, es decir, es la probabilidad de que por azar se halle un valor más extremo que el nuestro. Cuando los datos se ajustan a una normal, el p-value debe dar grande, mucho más grande que 0.05, lo cual dice que por puro azar y bajo la hipótesis de normalidad, es perfectamente corriente que se obtengan valores de p-value mucho menores, de datos mucho más extremos, es decir, tendiendo a ser anormales.

Si hacemos lo mismo con la distribución uniforme, obtenemos:

```
#Esto se corre primero
#Generamos 500 observaciones al azar de la uniforme
w<-runif(500)
 #Histograma de w
hist(w, seq(-3, 3, 1), prob=T)
#Envolvente según la Ho: densidad de la normal
curve((1/(  sd(w) * ((2*pi)^0.5)  ))  * (exp( -(x-mean(w))^2/(2*(sd(
#Densitómetro de barras
rug(w, side=1)
#
#Esto se corre después
#
#dibujo K-S
qqnorm(w)
#Si fuese perfecta normalidad daría una línea.
qqline(w)
#Test de Kolgomogorov - Smirnov para normalidad
ks.test(w, "pnorm", mean = mean(w), sd = sd(w))
```

3.5. Tablas

Las *tablas* son la generalización más inmediata de los vectores:

18 ***Tablas a partir de vectores***

Si uno va a hacer una prueba *t* para comparar las medias de dos vectores, uno debe tener a la mano dos vectores de datos. Pero si uno desea hacer una anova, lo que uno necesita es una *tabla* con varias columnas y no 3 o 4 vectores de datos. Para poner tablas a disposición de *R* hay varios caminos .

1. Teclear la tabla con el editor interno, el cual sirve igual de bien para teclear un vector o una tabla.

2. Formar una tabla a partir de vectores existentes.

3. Leer la tabla de un archivo, para lo cual uno tiene que grabarlos previamente en formato adecuado.

Para convertir uno o más vectores de la misma dimensión, mismo número de datos, en una tabla, uno adecúa y ejecuta el programa siguiente:

```
#CREAR TABLA A PARTIR DE VECTORES
GastosA<-c( 1,3,4,2,3,5,6,4,3)
GastosB<-c( 2,3,1,2,4,3,4,2,1)
#Unimos los datos en un sólo objeto
z<-data.frame(GastosA,GastosB)
z#publicamos la tabla en la consola.
```

Para manejar los elementos de una tabla hay varias maneras:

```
#MANEJO DE TABLA NUMERICA
a<- c(2001,2002,2003,2007,2010)
b <- c(2001+1,2002+2,2003+1,2007+3,2011)
t <- data.frame(a,b)
#Diversas publicaciones
t
t[1]
t[1,1]
t[5,2]
t$a
#Sume
t[1,1] + t[5,2]
t$a[1] + t$b[5]
```

19 ***Cambio de tipo numérico a factor***

Cuando uno ha codificado una variable con números, *R* lo interpretará como variable cuantitativa, *numérica*. Pero quizá a uno le interese que sea entendido como *factor*, es decir como *variable categórica*. Por ejemplo, el año 2001 no es un número sino el año en que nos fue bien. Y el 2005 fue cuando tuvimos recesión. Con estas connotaciones, los años no son números sino factores. Para *cambiar una variable de tipo numérico a factor*, se conserva la variable antigua pero se crea un duplicado de tipo factor. Para vectores:

```
#CAMBIO DE TIPO: DE NUMERICO A FACTOR  (CATEGORICO)
#Si uno tiene un vector
#Se copia un vector de números a otro de tipo factor
a<- c(2001,2002,2003,2007,2010)
b <- as.factor(a)
#Los números se pueden sumar
a[1]+a[2]
#Los factores no:
b[1]+b[2]
```

para cambiar el tipo de una columna dentro de una tabla:

```
#CAMBIO DE TIPO DE UNA COLUMNA DENTRO DE UNA TABLA
a <- c(2001,2002,2003,2007,2010)
b <- c(2004,2005,2006,2007,2009)
tabla <- data.frame(a,b)
tabla
#Cambiar  a de numérico a factor:
#se copia a en c, pero como factor.
#pero se deja a como numérico.
#La simbología padre$hijo
#denota el hijo de tal padre, en este caso
#c es hijo (columna) de tabla.
tabla$c <- as.factor(tabla$a)
tabla
tabla$a[1] + tabla$b[1]
tabla$a[1] + tabla$c[1]
```

20 *Tablas de contingencia*

Gnumeric estudia *tablas! de contingencia* tradicionales. *R* estudia tablas de contingencia como muestra el programa siguiente:

```
#TABLAS DE CONTINGENCIA DE 2 DIMENSIONES
edad<-c(1,2,3,4,5,6,7,8)
puntaje<-c(8,7,8,10,20,20,16,17)
#Se hacen categorías
edad.cat<-cut(edad,breaks=2)
puntaje.cat<-cut(puntaje,breaks=2)
#Se calcula la tabla de contingencia
d<-table(edad.cat,puntaje.cat)
d
plot(d)
#Se hace el test de independencia
chisq.test(edad.cat,puntaje.cat)
```

¿Qué hace este programa?
Supongamos que uno desea probar que los perros más viejos son más talentosos que los jóvenes. Entonces uno registra en un vector la *edad* y en otro el *puntaje* que bien puede ser cuantitativo o categórico. El puntaje de nuestro caso es cuantitativo. Se sobreentiende que *edad* y *puntaje* se aparean naturalmente. Si uno mira los datos, uno se da cuenta que en realidad los perros viejos tienen mejor puntaje. *R* nos ayuda de la siguiente forma: el particiona los datos en dos clases, conforme se lo pedimos en el procedimento *cat* (categorizar) y con el parámetro *breaks = 2* (número de particiones). En segundo lugar, las particiones se hacen de tal forma que la probabilidad de hallar dependencia entre los los dos factores *edad* y *puntaje* sea lo máximo posible. Los rangos de la particiones que *R* hace aparecen en la tabla. Cuando hay muchos datos queda bien tomar un número mayor de categorías.
Después ejecuta el test usual chi-cuadrado. *R* dice que no está muy seguro, lo cual es una forma educada de decir que hay un prerequisito que no se cumple. En este caso, que hay una o más casillas con menos de 4 elementos y lo mínimo recomenddo es 5.

3.6. Importar y exportar datos

Es usual que uno tenga los datos grabados en una hoja de cálculo, quizá *Excel* o *Gnumeric*. Nuestro problema es hacerle llegar los datos a *R*.

21 ***Importar datos desde Excel***

Si los datos están en *Excel*, se *graban o se regraban* como tipo *text (MS-DOS)*. Si están en *Gnumeric* se graban como tipo *txt* y se responde *OK* a todo lo demás.

Atención: *R* maneja variables cuantitativas (numéricas) y categóricas (factores). A veces se crea ambivalencia de manera inadvertida. Por ejemplo: hay una variable *Prom* (promoción) que indica el año en que salieron graduados unos profesionales. Es natural que uno codifque el año de Prom en números, digamos el 2001. Como consecuencia, *R* tomará la columna *Prom* como los datos de una variable cuantitativa. Cada vez que uno quiera hacer un análisis, es necesario decidir si la forma como uno ha codificado sus variables corresponde a los propósitos que uno tenga en el momento. La *GUI* que usaremos diferencia entre factores y variables cuantitativas muy claramente y a los factores les asigna el papel de *variables explicativas* y a las variables cuantitativas les asigna el papel de *variables respuesta o de salida*.

Para fijar ideas, consideremos un estudio que evalúa los sueldos promedios de unos profesionales. Los datos se presentan en la tabla siguiente. Una tabla como esta corresponde a una planilla de datos, tal como se llena en el campo, o en las encuestas o en los experimentos. Por éso, a este tipo de tablas las llamamos *tablas naturales*. Para la *GUI*, las variables *Prof*, *Univ* y *Gen* son factores (variables explicativas) en tanto que Prom, Sueldo y estrato son posibles variables respuesta. Luego aprenderemos a usar la *GUI* para cambiar el tipo de las variables. La tabla es la siguiente:

Prof	Prom	Univ	Gen	Sueldo	Estrato
Civil	2006	U1	M	2	4
Ind	2004	U2	H	3	5
Ind	2003	U2	M	2	2
Elec	2001	U3	H	1	3
Sist	2003	U1	M	2	4
Civil	2001	U2	H	3	6
Sist	2004	U3	M	2	6
Elec	2003	U2	M	1	3
Sist	2001	U2	H	2	2
Civil	2002	U3	M	3	3

Si uno desea, uno puede teclear los datos y salvarlos. Pero si uno quiere ahorrarse la tecleada, la forma de hacerlo es la siguiente:
Busque estos datos en el archivo adjunto, el que contiene los comandos, cópielos al clipboard, péguelos en *Gnumeric* y grábelos con nombre *tsueldos* y de tipo *txt*. Para salvar el archivo en *Gnumeric* usamos la opción *File + Save as*. Cuando se desplieque un cuadro de diálogo, elija el tipo de archivo en *File Type* como *Text*. Déle un nombre a su archivo, por ejemplo, *tsueldo*. Grábelo. A las preguntas restantes responda de la manera más simple o ignórelas (haga click sobre Save).

Inmediatamente después de grabar, revise el camino para llegar a su archivo *tsueldo.txt* y anótelo. En Linux puede ser algo así:

```
AJose/RProjectU1/RWorkU1/tsueldo.txt
```

Sobre Windows, un camino podría ser al estilo

```
C:/AJose/RProjectU1/WorkU1/tsueldo.txt
```

22 *Leer datos en R*

Podemos ahora *leer datos* en un archivo desde *R*:
Sobre Linux:

```
#IMPORTACION DE DATOS DESDE UN ARCHIVO TIPO txt sobre LINUX
tsueldos = read.table(file("AJose/RProjectU1/RWorkU1/tsueldos.txt"), encoding="latin1")
#Forma sinónima
tsueldos = read.table(file("AJose/RProjectU1/RWorkU1/tsueldos.txt"))
tsueldos #para listar un objeto, se teclea su nombre
```

Sobre Windows:

```
#IMPORTACION DE DATOS DESDE UN ARCHIVO TIPO txt sobre WINDOWS
#header = T significa que cada variable tiene un título
#como encabezado; header = F, significa que no hay título.
tsueldos <-read.table(file
  ("C:/AJose/RProjectU1/RWorkU1/tsueldos.txt"), header = T)
#para listar un objeto, se teclea su nombre
tsueldos
```

Modifique esta instrucción teniendo en cuenta sus sistema operacional y cambiando el camino al archivo por el camino verdadero de su archivo *tsueldo*. Es mejor hacer las modificaciones o ediciones en algún procesador de palabra. Copie su nueva instrucción al clipborad, active con un click a *R*, y péguela. Oprima Enter. Verá a continuación que *R* despliega los datos que leyó.

23 *Exportando resultados*

Uno puede ver los resultados que produce *R* pero ¿qué hacer si uno desea *exportar datos* a un documento personal?
Existen varias maneras. La primera es negrear con el cursor la zona que uno desee copiar, copiarla al clipboard y luego pegarla en el documento que uno previamente tiene abierto para incluir los resultados.

Otra manera es el menu de *R* y buscar en algún lugar una opción que le permita grabar el output. Puede ser el menu *scrollback* seguido de *save output*. Elegir estas opciones lo lleva a uno a crear un archivo, el cual se actualiza a voluntad.
Quizá uno quiera utilizar el mismo archivo como mesón de trabajo: en ése caso es necesario tener presente que cada vez que se inicializa *R* y se ordena grabar el output, uno borrará todo lo anterior. O bien, cada sesión debe originar su propio archivo con su propio nombre.

3.7. Gráficas

Cuando alguien se queja de que el paquete que usa no hace gráficas con suficiente calidad, de seguro no está usando *R* (en realidad, eso es tan difícil, que *R* también sufre).

24 ***Gráficas en pantalla***

Si uno desea una única *gráfica* que despliegue información de varios vectores de datos:

```
#VARIOS VECTORES DE DATOS EN LA MISMA GRAFICA
#Generamos los datos de unos gastos
ProyA<-c( 1,3,4,2,3,5,6,4,3)
ProyB<-c( 2,4,1,3,4,3,4,2,1)
#Unimos los datos en un sólo objeto
z<-data.frame(ProyA, ProyB)
#Gráfica:
#Dibujamos los datos en pantalla
#col = colores, 2 = rojo, 3= verde.
matplot(z,axes=F,frame=T,type='b',ylab="",col = 2:3)
# Se añaden los Títulos
title(ylab="Gastos", xlab = "Meses")
title("Gastos por mes \n proyecto A  y B")
#Caja de letreros: posición, contenido, lty = line types, colores
legend(1, 5, c(paste(" = ", c("A","B"))), lty = 1, col = 2:3)
#Dibuje la escala en el eje X
axis(1)
#Dibuje la escala en el eje Y
axis(2)
```

Si uno desea que varias gráficas aparezcan en la misma página:

```
#VARIAS GRAFICAS POR PAGINA
#cargar paquete
data(LifeCycleSavings, package="datasets")
#Listar datos
LifeCycleSavings
#
#Todas las gráficas en la misma página,
```

```
#formando una matriz de  3  filas y 2 columnas.
#par = modifique parámetros
par(mfrow=c(3,2), pch=16)
#Dibuje histogramas:
#LifeCycleSavings es al archivo de datos que contiene
#las variables sr, ddpi, pop15, pop75, dpi:
hist(LifeCycleSavings$sr,  plot = TRUE, breaks="Sturges",
  col="darkgray")
hist(LifeCycleSavings$ddpi,  plot = TRUE,  breaks="Sturges",
  col="darkgray")
hist(LifeCycleSavings$pop15,  plot = TRUE, breaks="Sturges",
  col="darkgray")
hist(LifeCycleSavings$pop75,  plot = TRUE,  breaks="Sturges",
  col="darkgray")
hist(LifeCycleSavings$dpi,  plot = TRUE,  breaks="Sturges",
  col="darkgray")
#Retorne a modo gráfico 1 x 1
par(mfrow=c(1,1), pch=16)
```

25 *Grabando gráficas*

Para *grabar gráficas* sobre Windows:
Se pueden copiar al clipboard y de allí pegar a donde uno desee.

Para grabar gráficas sobre Linux sobre Windows en formato *png*, el cual no consume tanto espacio:

Grabar una gráfica es dibujar en un archivo la gráfica y por éso se usa exactamente las mismas instrucciones en ambos casos. El código para la consola es una adaptación del siguiente, pero éso también puede hacerse desde la GUI,

```
#GRABAR GRAFICAS
#Parte 1
#Generamos los datos
GastosA<-c( 1,3,4,2,3,5,6,4,3)
GastosB<-c( 2,3,1,2,4,3,4,2,1)
#Unimos los datos en un sólo objeto
z<-data.frame(GastosA,GastosB)
#Dibujamos los datos en pantalla
matplot(z,axes=T,frame=T,type='b',ylab="")
#Títulos
title(ylab="Gastos", xlab = "Meses")#label for y-axis;
title("Gastos por mes \n proyecto A  y B")
#Se inicializa un archivo con formato png
#que ocupa poco espacio.
```

```
png(file="AJose/RProjectU1/RWorkU1/graf1.png")
#Dibujamos los datos en el archivo
#matplot: función que dibuja funciones
matplot(z,axes=F,frame=T,type='b',ylab="")
#Títulos
title(ylab="Gastos", xlab = "Meses")#label for y-axis;
title("Gastos por mes \n proyecto A  y B")
#Cerramos el archivo
dev.off()
#
#Parte 2
#Ahora grabamos un histograma
HGastosA<-hist(GastosA)
#Dibujo en pantalla
plot(HGastosA,main = paste("Histograma de GastosA"))
#Se inicializa un archivo con formato png
#que ocupa poco espacio.
png(file="AJose/RProjectU1/RWorkU1/graf2.png")
#Dibujamos los datos en el archivo
plot(HGastosA,main = paste("Histograma de HGastosA"))
#Cerramos el archivo: device off
dev.off()
#Terminar
#El archivo se puede editar con un procesador de imágenes.
```

Este programa creará una figura en el folder mencionado. El nombre de la figura es *graf1.png* y se abre y edita con cualquier aplicación que procese gráficas.

Capítulo 4

La interfase gráfica de R

Aprendamos a trabajar con *R* por medio de una *GUI* (Graphic User Interface). Una GUI es una interfase entre un programa y el usuario que ha sido diseñada para ser operada preferencialmente con un click. Podremos trabajar con *R* con tanta facilidad como se trabaja con *Excel* o con *Gnumeric*.

26 *El Rcmdr*

Ya habíamos dicho que existe una *GUI* para Windows que se llama *Tinn-R*. La *GUI* que probaremos es multiplataforma (sirve para Windows, Linux y Mac) y se llama *R Commander* y se instala desde *R*, dado que previamente uno se ha conectado a *Internet*. Para instalarla en Windows, uno busca un menu que diga *instalar paquete*, elige un servidor y el paquete *Rcmdr* y lo instala. Para instalarlo en Linux, uno debe entrar como administrador antes de llamar a *R*, y una vez con los derechos adecuados, uno llama a *R* e inserta la siguiente orden, la cual también sirve para Windows

```
# INSTALAR EL R COMMANDER, UNA GUI PARA R
install.packages("Rcmdr", dependencies=TRUE)
```

Después de oprimir *Enter*, *R* empieza un arduo trabajo que en Linux puede llevar quizá media hora. Uno puede almorzar entre tanto. Una vez terminado el trabajo, R abre la GUI. Pero si no lo hace, se abre con el comando

```
#ENCENDER EL R COMMANDER
library(Rcmdr)
```

R tratará de montar el *R Commander* y seguramente alegará que falta algún archivo. Lo mejor es decirle que no, que no lo instale, a no ser que se necesite

expresamente. O uno puede decirle que si a todo y ayudarle a escoger, por ejemplo, un servidor adecuado de donde bajar el archivo. Tiempo para un café. Con la versión que tengo actualmente de *R* sobre Linux, esta reclamadera lo hace siempre como si fuese la primera vez.
Cuando termine, uno debe buscar una nueva aplicación que estará abierta quizá detrás de la ventana de *R*. Y estaremos listos para experimentar. Uno puede explorar la barra de menus y darse cuenta de las opciones disponibles. Después de descubrir que tenemos la opción de trabajar con *R* con la misma facilidad con que se trabaja en *Excel*, uno se pregunta ¿para qué sirve acceder a *R* desde la consola de comandos si uno tiene una *GUI*? Pues hay tres casos importantes:

1. Cuando uno tiene tareas repetitivas y complejas: uno las resuelva una vez, graba el programa correspondiente (lo cual también puede hacerse con la *GUI*) y después invoca uno el programa cada vez que lo necesite.

2. Cuando uno requiera una modalidad de un paquete ya montado sobre la GUI pero que no esté implementado sobre ella.

3. Cuando uno necesite un paquete que no esté implementado en la *GUI*. Recordemos que *R* tiene como 2000 paquetes a nuestra disposición. Atención: si tiene pensado instalar *R* y sus paquetes en muchas máquinas, baje lo que necesite una sóla vez, guárdelo en una carpeta apropiada y de allí saque todas las copias que necesite. Si no hace éso, puede correr el riesgo de una penalización por demasiadas visitas innecesarias al sito de *R*, que se llama *cran* (cran significa *comprehensive R archive network*).

Mientras que uno trabaja con la *GUI*, uno minimaliza la ventana de *R*. Para salir de *R*, se teclea q(). Si uno mata a R, puede tener problema con la GUI que queda como espectro molesto e inútil.

27 *¡Y ahora qué hago?*

La primera tarea que uno puede hacer es introducir un vector de datos y sacarle la media. Para ello, uno activa el menu *Data* y señala *New data Set*. Se abre el editor de datos y uno le pone el nombre a sus datos, por ejemplo, sueldoA, y rellena una columna con los datos, que se ha desplegado. Se termina con *Quit*. Después de ello, uno va al menu *statistcs* y busca alli *summary* or *numerical summary* y lo ejecuta. En el output uno encontrará la media y seguramente la varianza. La GUI permite hacer correcciones a voluntad de archivos de datos, para ello es suficiente hacer click en el botón *Edit data set* y preceder en correspondencia.

Dado que uno tiene un columna de datos, uno mira aquí y allá hasta que encuentre el menu apropiado para hacer una prueba *t* para poner a prueba la creencia de que la media vale tanto y con la significancia requerida.

Uno tiene la opción de editar varios objetos de datos. ¿Con cuál de todos trabajará *R*? Para elegir el objeto de trabajo, hay una casilla en donde aparece uno de los nombres de los objetos de datos. Si se hace click sobre dicha casilla, se despliega un diálogo en el cual uno puede activar uno de los objetos y con ése es que *R* trabajará.

Bueno, ¿qué hacer si queremos estudiar la regresión de un primer vector sobre otro? ¿O una comparación de medias? ¿O una comparación de varianzas? Atención: *R* Commander es una *GUI* inteligente, lo cual significa que ofrece sugerencias de acuerdo a los datos que uno tenga. La forma como hace las sugerencias es simple y directa: las opciones de menu que estén activada y que aparecen en negro reteñido son las opciones que el *R* Commander sugiere para hacer. Las opciones que no corresponden quedan desactivadas en gris claro. Precaución: ser inteligente es algo difícil. Por ahora, la inteligencia de la *GUI* no es muy alta y es mejor confiar en uno mismo, o uno tiene que exponerse a hacer cosas sin sentido que la *GUI* y *R* le permitirán.

Capítulo 5

Regresión y Anovas

28 ***Objetivo.*** *Aprenderemos a hacer regresión univariada y diversos tipos de anovas para comparación de medias.*

5.1. Regresión lineal univariada

En *regresión lineal* se estudia la hipótesis de dependencia lineal entre dos vectores de datos. Para hacer una regresión desde la consola se usa un programa como el siguiente que se usa con copy+paste desde el archivo de comandos:

```
#REGRESION UNIVARIADA ENTRE DOS VECTORES
#Parte 1
x <- c(2,3,4,1,4,5,3)
y<- c(2,4,5,6,7,8,9)
#El símbolo ~ indica que se estudia una relación entre
#la variable dependiente a la izquierda y las dependientes
#a la derecha.
#La regresión se estudia por medio de
#los modelos lineales (linear models).
#z es el nombre del modelo a estudiar.
z<-lm(x ~ y)
#Diagrama de dispersión:
#Cargar paquete
library(car)
scatterplot(sueldoA~sueldoB, reg.line=lm, smooth=TRUE, labels=FALSE,
  boxplots='xy', span=0.5)
#Parte 2
```

```
plot(z)
```

Si los datos están en una tabla de dos columnas (sueldoA, sueldoB), se adecúa el siguiente programa:

```
#REGRESION ENTRE DOS COLUMNAS DE UNA TABLA
sueldoA <- c(2,3,4,1,4,5,3)
sueldoB<- c(2,4,5,6,7,8,9)
sueldos <- data.frame(sueldoA,sueldoB)
z<-lm(sueldoA ~ sueldoB, data = sueldos)
z
#Cargar paquete que dibuja
library(car)
scatterplot(sueldoA~sueldoB, reg.line=lm, smooth=TRUE,
  labels=FALSE,   boxplots='xy', span=0.5, data=sueldos)
plot(z)
```

Es inteligente mirar las gráficas y las tablas de salida a ver si es correcta la designación de cuál variable es independiente o explicativa y cuál es dependiente o de respuesta. Si la asignación hecha es incorrecta, es suficiente escribir *sueldoB* $\sim$ *sueldoA* en vez de *sueldoA* $\sim$ *sueldoB*.

Si uno desea hacer una regresión lineal usando la *GUI*, uno no puede presentarse con dos vectores. Uno debe presentarse con una tabla que contenga dos columnas, y con títulos para cada columna, la cual se puede editar con *Data* y *New data Set*. En general, *R* no trabaja sino con un objeto. Si uno quiere trabajar con varios objetos, uno debe armar con ellos un objeto mayor y trabajar con el. En particular, si uno tiene dos vectores, uno puede tratar de hacer una tabla con ellos, para lo cual en *Data + Merge data set*, uno puede elegir que objetos unir en un nuevo objeto, al cual se le dará su nombre particular. Como uno quiere pegar dos columnas, uno debe elegir una opción que especifique que la operación *merge* (pegar, unir) se hace por columnas. Si uno no especifica que la unión se hace por columnas, ésta se hará for filas y como consecuencia los datos de las dos columnas se unen y se produce una nueva gran columna.
Generalmente uno tiene en memoria varios objetos. Pero *R* no trabaja sino con uno solamente. Por consiguiente, debe existir la forma de elegir con cual objeto trabajar. Por éso, la *GUI* tiene una casilla donde aparece el nombre de alguno de los objetos. Si uno hace click sobre la casilla, se despliega una lista con todos los objetos en memoria. Uno señala su objeto elegido y será con él con el que se trabaje.

Supongamos entonces que uno tiene un objeto *sueldos* (que tal vez tiene dos columnas y que salió de pegar dos objetos columna, *sueldoA*, *sueldoB*, que representan los sueldos de dos compañeros que trabajan en ventas). Para hacer una regresión lineal entre las dos columnas, uno va a *statistics, fit models, linear regression*. Allí uno puede elegir cual es la variable de respuesta y cual la de entrada o explicativa. Para asegurarse de que las cosas son como uno las ha imaginado, lo mejor es hacer una gráfica, la cual se hace mediante el menu *Graphs + scatterplot* (diagrama de dispersión). Allí uno debe escoger quien va en las x's y quien en las y's. Si uno desea grabar la gráfica, sobre el mismo menu uno busca y ejecuta la opción correspondiente.

5.2. Anovas

Las cosas sencillas son más fáciles con *Gnumeric*, pero las más complicadas son más fáciles en *R*. Por ejemplo, una *anova* sencilla, de una o dos vías, para comparar medias, se puede hacer en *Gnumeric*. De hecho, una tabla de dos o tres vectores no activa la casilla de las anovas de la *GUI*. Pero para anovas más complicadas y a partir de planillas naturales (las que salen directamente de la encuesta o del experimento) se analizan mejor con *R*.

29 *Anova de una vía*

La tabla de datos para un experimento que estudia la incidencia de la temperatura sobre la producción promedio en kg/m^2 de un sembradío de mora es la siguiente:

9^o C	18^o C	25^oC
1	3	1
2	4	1
1	5	2

Es muy sencillo lograr que *Excel* o *Gnumeric* nos hagan la *anova* (el análisis de varianza) para los datos de la mora. Si es sobre *Gnumeric*, uno teclea los datos sobre una planilla, los negrea con el cursor, va a *tools*, luego a *statistical analysis*, luego a *anova*, y allí escoge *one way anova*. El resultado en *Excel* luce como sigue:

Anova for three temperatures						
Origin of variation	SS	DF	Mean Square	F-Exp	Probab	Critical F
Among groups	14,22222222	2	7,111111111	12,8	0,0068453	5,143249382
Within groups	3,333333333	6	0,555555556			
Total	17,55555556	8				

Gnumeric produce la misma tabla aunque la titula algo diferente, por ejemplo, en vez de *among groups* escribe *between groups*. Los valores de *p-value* y del valor crítico difieren por allá en el 4 decimal. *Gnumeric* tiene una fama de ser muy exacto y *Excel* no. Por eso, si se requiere mucha precisión, uno puede preferir las respuestas de *Gnumeric*, el cual acompaña la tabla con los valores de la media y de la varianza por columna.
Para hacer la *anova* de la mora en *R* hay que ponerle los datos como fueron registrados en el campo. La idea es que uno tiene viveros con termostatos que gradúan y conservan la temperatura y que para evitar sesgos, por ejemplo de posicionamiento, los tratamientos se asignan al azar a los viveros. Esto es parte del protocolo para garantizar que los datos vengan al azar. Supongamos pues que la lista de datos fue

Tratamiento	Prod
nueve	1
docho	3
nueve	2
vcinco	1
nueve	1
vcinco	1
docho	4
vcinco	2
docho	5

donde hemos usado las contracciones *docho = dieciocho*, *vcinco = veinticinco*.
El programa en *R* para la anova podría ser:

```
#ANOVA UNIFACTORIAL
#Los tratamientos a los 9 viveros
Trat<-c( "nueve", "docho", "nueve","vcinco",
        "nueve", "vcinco", "docho", "vcinco","docho")
#Producción de los viveros
```

```
Prod<-c( 1,3,2,1,1,1,4,2,5)
#Se hace una tabla a partir de dos vectores
Mora<-data.frame(Trat,Prod)
Mora
#Tabla bidimensional de frecuencias absolutas
z<-table(Mora$Prod,Mora$Trat)
z
#analysis of variance
w<-aov(Mora$Prod ~ Mora$Trat, projections = TRUE )
summary(w)
#Comparación de medias por pares según Tukey
#Se ordenan las diferencias entre medias antes del test
tt <- TukeyHSD(w, ordered = TRUE, conf.level = 0.99)
plot(tt)
```

Este programa arroja un output casi idéntico al de *Gnumeric* y al de *Excel*. Añade la convención de indicar la significancia de los datos por medio de estrellas: dos estrellas, **, dicen que la significancia es menor que 0.01, tres, que es menor que 0.001.

Es posible que alguien prefiera usar una codificación numérica para los tratamientos. En tal caso, es posible que se le ocurra usar un programa del siguiente estilo:

```
#Los tratamientos a los 9 viveros
Trat<-c( 9, 18, 9,25, 9, 25, 18, 25,18)
#Producción de los viveros
Prod<-c( 1,3,2,1,1,1,4,2,5)
#Se hace una tabla a partir de dos vectores
Mora<-data.frame(Trat,Prod)
Mora
#Tabla bidimensional de frecuencias absolutas
z<-table(Mora$Prod,Mora$Trat)
z
#analysis of variance
w<-aov(Mora$Prod ~ Mora$Trat, projections = TRUE )
summary(w)
```

Sucede que *R* toma en serio los números y produce lo que no debe. Para que proceda como queremos, hay que decirle que los números son sólo símbolos que indican una clasificación categórica, como blanco y rojo, y no numérica:

```
#Los tratamientos a los 9 viveros
trat<-c( 9, 18, 9,25, 9, 25, 18, 25,18)
#Tome los números como categorías o factores
Trat <- as.factor(trat)
#Producción de los viveros
Prod<-c( 1,3,2,1,1,1,4,2,5)
#Se hace una tabla a partir de dos vectores
Mora<-data.frame(Trat,Prod)
Mora
#Tabla bidimensional de frecuencias absolutas
z<-table(Mora$Prod,Mora$Trat)
z
#analysis of variance
w<-aov(Mora$Prod ~ Mora$Trat, projections = TRUE )
summary(w)
#Comparación de medias por pares según Tukey
#Se ordenan las diferencias entre medias antes del test
tt <- TukeyHSD(w, ordered = TRUE, conf.level = 0.99)
plot(tt)
```

En un experimento real o en una encuesta, lo más seguro es que uno haya tecleado los datos a *Excel* o a *Gnumeric*. Para hacerlos llegar a *R* se regraban en un archivo tipo text, el cual es entendible por *R*. Para fijar ideas, consideremos la tabla que estudia los sueldos entre unos profesionales, reportada en el capítulo anterior y que seguramente ya fue grabada. Para hacerla llegar a *R* por medio del *GUI*, uno va a *Data + Import data + from text file* y allí localiza el archivo correspondiente. Al abrirlo se abre un diálogo y hay que darle un nombre, el nombre que llevará la tabla en *R*, y uno deja el resto sin tocar y abre el archivo. Puede luego mostrarlo o hacerle diversas correcciones. Y uno notará que esta tabla activa la casilla de las anovas (que se busca en el área de estadística y de medias). Y puede uno entonces hacer diversas comparaciones de medias.
Si uno prefiere usar la consola de *R*, uno podría teclear alguna adaptación del programa siguiente, que se ejecuta sobre la tabla *tsueldos*, la misma del capítulo pasado. El siguiente programa para Linux debe adecuarse teniendo en cuenta el camino al archivo que contiene los datos de *tsueldos*:

```
#PROGRAMA PARA  LA ANOVA DE UNA VIA EN LINUX
tsueldos <-  read.table(file("AJose/RProjectU1/RWorkU1/tsueldos.txt"), header = T)
tsueldos
#Tabla bidimensional de frecuencias absolutas  sueldo  vs prof
z<-table(tsueldos$Sueldo,tsueldos$Prof)
```

```
z
#analysis of variance
w <- aov(tsueldos$Sueldo ~ tsueldos$Prof, projections = TRUE)
summary(w)
#Comparación de medias por pares según Tukey
#Se ordenan las diferencias entre medias antes del test
tt <- TukeyHSD(w, ordered = TRUE, conf.level = 0.99)
plot(tt)
```

Para Windows, uno puede usar adaptar la siguiente versión:

```
#PROGRAMA PARA  LA ANOVA DE UNA VIA SOBRE WINDOWS
tsueldos <-read.table(file("C:/AJose/RProjectU1/RWorkU1/tsueldos.txt"), header = T)
#Publicamos en consola el archivo
tsueldos
#
#Tabla bidimensional de frecuencias absolutas  sueldo  vs prof
z<-table(tsueldos$Sueldo,tsueldos$Prof)
z
#analysis of variance
w <- aov(tsueldos$Sueldo ~ tsueldos$Prof, projections = TRUE)
summary(w)
#Comparación de medias por pares según Tukey
#Se ordenan las diferencias entre medias antes del test
tt <- TukeyHSD(w, ordered = TRUE, conf.level = 0.99)
plot(tt)
```

Si uno desea comparar la salida de *R* con la de *Gnumeric*, es necesario darle a *Gnumeric* la siguiente tabla que reporta los sueldos individuales por profesión:

Civil	Eléct	Ind	Sist
2	1	3	2
3	1	2	2
3			2

5.3. Anova con bloqueo

Tenemos a los datos de un experimento con mitocondrias, en el cual medimos la producción de ATP dependiendo de la temperatura y del del pH. Nuestro objetivo es saber si hay una diferencia de medias entre los diversos niveles de pH, guardando un control sobre la temperatura a la que se hace el experimento. Usamos un test de *anova de bloques o con bloqueo*:

Mitocondria producido de ATP			
	pH 4	pH 6	pH 8
30^o	5	8	6
36^o	7	10	8
42^o	4	7	3

Uno puede procesar estos datos en *Gnumeric:* uno elige el menu *Tools*, luego *statistics*, luego *anova* y después escoge *Two factors + one row per sample*.

Como *Gnumeric* hace las cosas tan fáciles, *R* no es necesario para este tipo de casos. *R* se vuelve interesante cuando uno tiene una tabla natural, como la siguiente y de la cual pudo haber venido la tabla anterior:

pH + Temp → Prod		
pH	Tem	Prod
4pH	30G	5
4pH	36G	7
6pH	36	10
8pH	42G	3
6pH	30G	8
4pH	42G	4
8pH	30G	6
6pH	42G	7
8pH	36G	8

Podemos usar la consola para hacer que *R* nos haga la anova con bloqueo:

```
#ANOVA CON BLOQUEO
pH <- c(4,4,6,8,6,4,8,6,8)
pH <-as.factor(pH)
temp <- c(30,36,36,42,30,42,30,42,36)
temp <- as.factor(temp)
prod <- c(5,7,10,3,8,4,6,7,8)
Mitoc <-data.frame(temp,pH,prod)
Mitoc
#Analysis of variance:
#Prod es la variable respuesta
#temp es la variable experimental, la importante
# pH es la variable secundaria que se controla
```

```
w<-aov((Mitoc$prod ~ Mitoc$temp  + Mitoc$pH), projections = TRUE )
summary(w)
#
#Comparación de medias por pares según Tukey
#Se ordenan las diferencias entre medias antes del test
tt <- TukeyHSD(w, ordered = TRUE, conf.level = 0.95)
plot(tt)
```

30 ***Manejando los datos que vienen con R***

Hay *paquetes* de datos y paquetes de programas y hay paquetes que traen ambas cosas. El siguiente es un ejemplo de manejo de paquetes de datos:

```
#MANEJO DE PAQUETES DE DATOS
#Se corre por partes
#Parte uno
#Listar todos los paquetes de datos
data()
#Cargar paquete
data(CO2, package = "datasets")
#Publicar  información sobre el paquete
help("CO2")
#Parte 2
#Listar datos
CO2
#TEST DE BARTLETT para comparar varias varianzas.
bartlett.test(uptake ~ Treatment, data = CO2)
```

5.4. Anova bifactorial

Una *anova bifactorial o de dos factores* se maneja tanto en *Gnumeric* como en *R*.

31 ***Ejemplo en Gnumeric de anova bifactorial*** *Tools + statistical analysis + ANOVA + two factor*

Primero uno teclea la siguiente matriz de datos que reporta el puntaje dado por la mamá a su niña que está aprendiendo a comportarse como la ama de su casa. Vemos que al cambiar los niveles de cada factor, hay cambio en los promedio del puntaje. Por tanto, esperamos que cada factor sea importante, es decir, sus p-value deben ser muy pequeños. De manera semejante, los promedios por par de

tratmientos son diferentes tanto a lo largo de las filas como de las columnas. Por consiguiente, esperamos que la interacción sea estadísticamente significativa. La tabla es la siguiente:

Puntaje para tareas caseras		
	Trapear	No trapear
barrer	5	2.1
	4.3	2.2
	4.4	2.4
	4.6	2.1
no barrer	2.2	−1
	2.2	−2
	2.1	−1
	1.9	−2

Atención: para darle los datos a *Gnumeric*, se negrean cuando él los solcite. Por tanto, los títulos no cuentan. Para recuperar la información perdida, *Gnumeric* tiene una casilla que debe llenarse y que pregunta sobre el número de datos por muestra (*rows per sample*). En nuestro ejemplo, dicho valor es 4.

Es hermoso ver el despligue que hace *Gnumeric* al analizar esta tabla de conteo.

32 *Anova bifactorial en R*

Cuando uno tiene una tabla natural y quiere una anova bifactorial, uno usa R. Para lo cual uno adpata el siguiente programa a su archivo específico. El programa toma un paquete de datos que viene con *R* y le hace una anova bifactorial.

```
#ANOVA BIFACTORIAL
#Cargar paquete
data(CO2)
#Publicar  información sobre el paquete
#teclear q para salir de la ayuda
help("CO2")
#Listar datos
CO2
#Anova bifactorial
# uptake = variable respuesta
#Treatment, type = variables explicativas
AnovaModel.1 <- (lm(uptake ~ Treatment*Type, data=CO2))
```

```
summary(AnovaModel.1)
w <-CO2
#Tabla bidimensional de frecuencias absolutas
z<-table(w$Treatment,w$Type)
z
```

Podemos verificar que un modelo bifactorial es igual a un modelo con bloqueo pero cuando se incluyen las interacciones. El símbolo para la interacción de dos factores es dos puntos. El nuevo programa es:

```
#ANOVA BLOQUEO MAS INTERACCIONES
#Cargar paquete
data(CO2)
#Publicar  información sobre el paquete
#teclear q para salir de la ayuda
#help("CO2")
#Listar datos
CO2
#Anova bifactorial
# uptake = variable respuesta
#Treatment, type = variables explicativas
AnovaModel.1 <-
  (lm(uptake ~ Treatment+Type + Treatment:Type, data=CO2))
summary(AnovaModel.1)
w <-CO2
#Tabla bidimensional de frecuencias absolutas
z<-table(w$Treatment,w$Type)
z
```

5.5. Manovas

La palabra *manova* es una contracción de *multivariate analysis of variance* y se supone que es una extensión natural de la filosofía optimizante del diseño de experimentos: si los datos pesan más juntos que separados, entonces los diseños deben generalizarse al máximo. Hemos visto cómo se generalizan incluyendo más y más variables experimentales, que se controlan, lo mismo que sus interacciones. Pero falta incluír en esta generalización lo que las manovas si tienen: tratamiento conjunto no de una sino de varias variables respuesta y también con sus interacciones.

La hipótesis nula de una manova es que las variables experimentales, las que se controlan, no tienen ninguna incidencia ni sobre el promedio de cada una de las variables respuesta ni sobre sus interacciones. El estadígrafo de contraste es en definitiva una F. La H_o se rechaza con la cola superior. La hipótesis alterna dice que algún subconjunto de factores experimentales o de sus interacciones inciden sobre algún subconjunto de variables respuesta o sobre sus interacciones.

Las siguientes instrucciones publican la ayuda que a cerca de las manovas viene con *R*:

```
help(manova)
help(summary.manova)
```

Veamos un ejemplo que viene con *R* al cual el autor le ha añadido algunos comentarios. Explicamos las cosas nuevas antes de verlo:
Sabemos utlizar el procedimiento *c()* (de concatenar) para crear un vector de datos. Si los datos son números, se sobreentiende que el vector es numérico. Para unir varios vectores en una tabla, hemos usado el procedimiento *data.frame()*. Este procedimiento se generaliza en otro que pega no sólo vectores sino también tablas y se llama *cbind()*. Para cambiar de tipo un tabla numérica se usa *factor()* que la convierte en categórica.
Podemos *generar tablas* de la siguiente forma:

```
uso <- factor(gl(2,10), labels=c("Bajo", "Alto"))
```

Este programa crea un vector, *uso*, de tipo factor o categórico que tiene 2 niveles o categorías, *Bajo* y *Alto*, y que se listan en réplicas seguidas de a 10. Dicha instrucción produce el vector:
Bajo Bajo Bajo Bajo Bajo Bajo Bajo Bajo Bajo Bajo Alto Alto Alto Alto Alto Alto Alto Alto Alto Alto
El procedimiento

```
aditivo <- factor(gl(2, 5, length=20),
                  labels=c("Bajo", "Alto"))
```

crea un vector de tipo factor o categórico que tiene 2 niveles o categorías, *Bajo* y *Alto*, y que se listan en réplicas seguidas de a 5. Las réplicas se alternan hasta llegar a un total de 20 datos. Dicha instrucción produce el vector:
Bajo Bajo Bajo Bajo Bajo Alto Alto Alto Alto Alto Bajo Bajo Bajo Bajo Bajo Alto Alto Alto Alto Alto

La siguiente es una tabla natural que presenta los datos de un experimento que estudia la dependencia de 3 variables respuesta (*degaste, brillo, opacidad*) ante dos variables experimentales: *uso* y *aditivo = cantidad de aditivo protector* que se le da a una cinta plástica.

```
##DIVERSAS FORMAS DE DEFINIR TABLAS
#Una tabla natural  se define por sus 3 columnas
desgaste <- c(6.5, 6.2, 5.8, 6.5, 6.5, 6.9, 7.2, 6.9, 6.1, 6.3,
              6.7, 6.6, 7.2, 7.1, 6.8, 7.1, 7.0, 7.2, 7.5, 7.6)
brillo <- c(9.5, 9.9, 9.6, 9.6, 9.2, 9.1, 10.0, 9.9, 9.5, 9.4,
               9.1, 9.3, 8.3, 8.4, 8.5, 9.2, 8.8, 9.7, 10.1, 9.2)
opacidad <- c(4.4, 6.4, 3.0, 4.1, 0.8, 5.7, 2.0, 3.9, 1.9, 5.7,
                 2.8, 4.1, 3.8, 1.6, 3.4, 8.4, 5.2, 6.9, 2.7, 1.9)
#El procedimiento cbind (pegar) forma grandes tablas a partir
#de otras mas pequeñas
Y <- cbind(desgaste,brillo, opacidad)
Y
#El procedimiento factor transforma un vector numérico en otro
# de tipo factor (categórico)
#gl(n,k) significa que hay n niveles con k réplicas
#gl(2,10) significa que hay 2 niveles con 10 réplicas
#labels=c("Low", "High") significa que los dos niveles son Bajo y Alto
uso <- factor(gl(2,10), labels=c("Bajo", "Alto"))
uso
aditivo <- factor(gl(2, 5, length=20), labels=c("Bajo", "Alto"))
aditivo
Tabla <- cbind( uso, aditivo, Y)
Tabla
```

El análisis de todo experimento, con datos y manova, es el siguiente:

```
##MANOVA: multivariate analysis of variance (comparación de medias)
## Ejemplo sobre la producción de cinta plástica tomado
#de Krzanowski (1998, p. 381)
#Una tabla natural  se define por sus 3 columnas
desgaste <- c(6.5, 6.2, 5.8, 6.5, 6.5, 6.9, 7.2, 6.9, 6.1, 6.3,
              6.7, 6.6, 7.2, 7.1, 6.8, 7.1, 7.0, 7.2, 7.5, 7.6)
brillo <- c(9.5, 9.9, 9.6, 9.6, 9.2, 9.1, 10.0, 9.9, 9.5, 9.4,
               9.1, 9.3, 8.3, 8.4, 8.5, 9.2, 8.8, 9.7, 10.1, 9.2)
opacidad <- c(4.4, 6.4, 3.0, 4.1, 0.8, 5.7, 2.0, 3.9, 1.9, 5.7,
                 2.8, 4.1, 3.8, 1.6, 3.4, 8.4, 5.2, 6.9, 2.7, 1.9)
#El procedimiento cbind (pegar) forma grandes tablas
#a partir de otras mas pequeñas
Y <- cbind(desgaste,brillo, opacidad)
Y
#El procedimiento factor transforma un vector numérico en
#otro de tipo factor (categórico)
#gl(n,k) significa que hay n niveles con k réplicas
#gl(2,10) significa que hay 2 niveles con 10 réplicas
#labels=c("Low", "High") significa que los dos niveles son Bajo y Alto
uso <- factor(gl(2,10), labels=c("Bajo", "Alto"))
uso
aditivo <- factor(gl(2, 5, length=20), labels=c("Bajo", "Alto"))
aditivo
```

```
#MANOVA para estudiar la incidencia de uso y aditivo sobre
#las variables desgaste, brillo, opacidad y sus interacciones:
fit <- manova(Y ~ uso * aditivo)
#Tablas anova  multifactorial (una variable respuesta a la vez)
summary.aov(fit)
Tabla de estadígrafos de  Pillai
#
summary(fit)
#Tabla de estadígrafos de Wilks
summary(fit, test="Wilks")
```

Al ver la salida de este programa, uno se da cuenta que se interpreta de igual forma que todo los visto hasta ahora. En particular, las dos variables experimentales fueron seleccionadas sabiamente y dió lo que se esparaba, que un cambio entre sus diversos niveles cambia significativamente las variables respuesta. Más detalladamente, ambas variables influyen sobre el desgaste, el brillo se ve influenciado por el régimen de uso pero no por el de aditivo, y en cuanto a opacidad, el experimento no revela sensibilidad a los cambios sufridos en las variables experimentales. Significancia por defecto = 0.05.

5.6. Paquetes

El poder de R está en sus más de 2000 *paquetes* que extienden sus servicios básicos. Por esto es que ninguna GUI podrá ser lo suficientemente amplia para ser autosuficiente: sin programación por consola, es imposible hacer buen uso de ellos.

33 ***Datos que vienen con R***

Uno encuentra en *R* diversas tablas de datos, con las cuales uno puede experimentar. Para listar todas las tablas de datos dentro de *R*, uno puede ir a *Data + data in packages* y activar la opcion *list data sets in packages*. O también, en la consola uno puede teclear el comando

```
data(package = .packages(all.available = TRUE))
```

Allí hay datos para todos los gustos. Puede ser conveniente saber exactamente qué, así que listamos los paquetes y el tema principal de sus datos:
boot: de todo.
car: ciencias sociales.
cluster: análisis de clusters.
coda: regresión lineal simple.
datasets:natural sciences.

effects: social sciences.
ergm: networks
fEcofin: series de tiempo.
gdata: medicina.
gtools: Elisa (inmunología).
HSAUR: medicina.
ineq: algo sobre Filipinas.
ipred: cáncer (de pulmón).
lattice: varios.
maps: mapas diversos.
MASS: medicina.
Matrix: matrices.
mice: datos diversos + missing data.
mitools: multiple inputations (=?)
mlbench: biología.
mlogit: datos para regresión logística.
network: redes.
nlme: science.
np: sueldos
nws: tarjetas de crédito.
plm: ciencias sociales.
rmeta: meta-analysis (=?)
rpart: varios.
sandwich: economía.
sem: economía.
sna: sicología social.
sp: imágenes y otros.
statmod: soldadura(?)
strucchange: ciencias sociales.
survey: educación y política.
survival: sobrevivientes a enfermedades.
TeachingDemos: datos para enseñar detalles de estadística.
timeseries: series de tiempo.
vcd: medicina.
xtable: puntajes en un examen.
Zelig: ciencia política.

34 ***Instalación de paquetes***

Cuando una baja *R* por primera vez, lo que recibe es la parte básica y por eso el nombre oficial de lo que uno recibe es *R-base*.
Esta entrega viene con varios paquetes listos para ser usados si tan sólo se les invoca por medio de la función *library(?)*. Pero hay otros paquetes que es necesario bajar pues no son parte de *R-base*.
Para bajar paquetes sobre Windows: se usa el menu de R: si el paquete aparece listado, se procede por inercia. Si el paquete no aparece listado, se baja la versión para Windows desde la CRAN. La *CRAN* (la casa madre de R) tiene paquetes para todo lo que uno se imagine y para dos mil cosas más. Por ejemplo, tiene muchos paquetes para genética y biología molecular. Para ello, uno busca *cran* en cualquier motor de búsqueda, digamos Google, y de allí se conecta a la página web de *R*. Sobre dicha página, uno busca un link que lo lleve a los paquetes (packages). Siguiendo dicho link, uno llega a una lista alfabética de paquetes y allí busca el paquete deseado. Al buscar, hay que tener presente que la lista distingue mayúsculas de minúsculas tanto en la primera letra como en las demás y que los números van primero que las letras. Uno puede examinar la lista de paquetes con una descripción de media línea sobre su función.
Cuando uno haya bajado el paquete y lo haya puesto en alguna carpeta, vuelve a usar el menu de R para instalarlo. Y después ya lo puede correr.
Sobre Linux: uno va a CRAN, y de allí baja la versión para Linux. Cuando uno haya bajado y salvado su archivo en una carpeta apropiada, uno abre una terminal encima de dicha carpeta y con derechos *su* (superuser), uno teclea el siguiente comando:

```
R CMD INSTALL rgl_0.91.tar.gz
```

en donde uno reemplaza el archivo listado por el que corresponda.

Capítulo 6

Para saber más

Hay en *R* tanta diversidad de quehaceres y tan altos niveles de calidad y de complejidad que *R* bien puede elegirse como una profesión. El paso siguiente a lo que hemos hecho puede darse a lo largo de alguna de las siguientes recomendaciones:

- *R* permite hacer trabajo con mucho detalle, para expertos, incluyendo imponentes gráficas, que se pueden confeccionar a todo dar. Uno puede convencerse de éso echándole una ojeada al siguiente documento:

  http://math.illinoisstate.edu/dhkim/Rstuff/Rtutor.html

- Hay una exhuberante galería de gráficas con su código en el sitio siguiente:

  `http://addictedtor.free.fr/graphiques/`

- Un muy bien lugar para profundizar sobre las opciones de *R* en estadística es *Quick-R* en el siguiente sitio:

  `http://www.statmethods.net/index.html`

- Para saber más sobre el *Rcmdr*, la *GUI* que nosotros usamos, un puede conectarse al siguiente sitio:

  `http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/`

  donde uno puede encontrar un link a un *Introductory manual*. En la ayuda de la GUI también hay un link al mismo documento.

- Para aprender estadística: *Estadística y decisiones*, José Rodríguez. Lo fundamental en el Vol 1. Para multivariado con *R*, el vol II. Este material puede bajarse de

  ```
  http://www.evoljava.com
  ```

  Creo que la forma más comoda y segura de proseguir por este largo camino es estudiando este material.

- Es posible aunque improbable que *R* se atasque debido a un volumen excesivamente grande de datos, por ejemplo, al estudiar macroproyectos. Si éso llegase a pasar, es necesario tener en cuenta que manejar grandes cantidades de datos es de por sí una profesión, la cual se implementa con una base de datos. La predilecta por los usuarios de *R* parece ser una versión apropiada de *MySQL* (My es el nombre de una niña, hija de uno de los desarrolladores) que se llama, por supuesto, RMySQL y que se baja de la sección de paquetes de

  ```
  http://cran.r-project.org/
  ```

  Las instrucciones para conectar *RMySQL* a *R* pueden encontrarse en el link *R data Import/Export* de la ayuda en línea de *R* que se llama con

  ```
  help.start()
  ```

- El poder de *R* es enciclopédico con una excelente bibliografía en línea. Para saber qué tiene *R* para determinado tema, se va a la página de la *cran* y se busca el menu *Search*. En la ventana emergente uno busca *R site search* y allí teclea el tema. En respuesta se desplegará toda una serie de opciones a elegir. Si no da nada, de seguro es que hay mala ortografía.

- Una muy buena ayuda puede venir de trabajar en equipo: comparta este material con sus compañeros y arme un grupo de estudio.

Índice alfabético