CAPÍTULO 2.
MARCO TEÓRICO

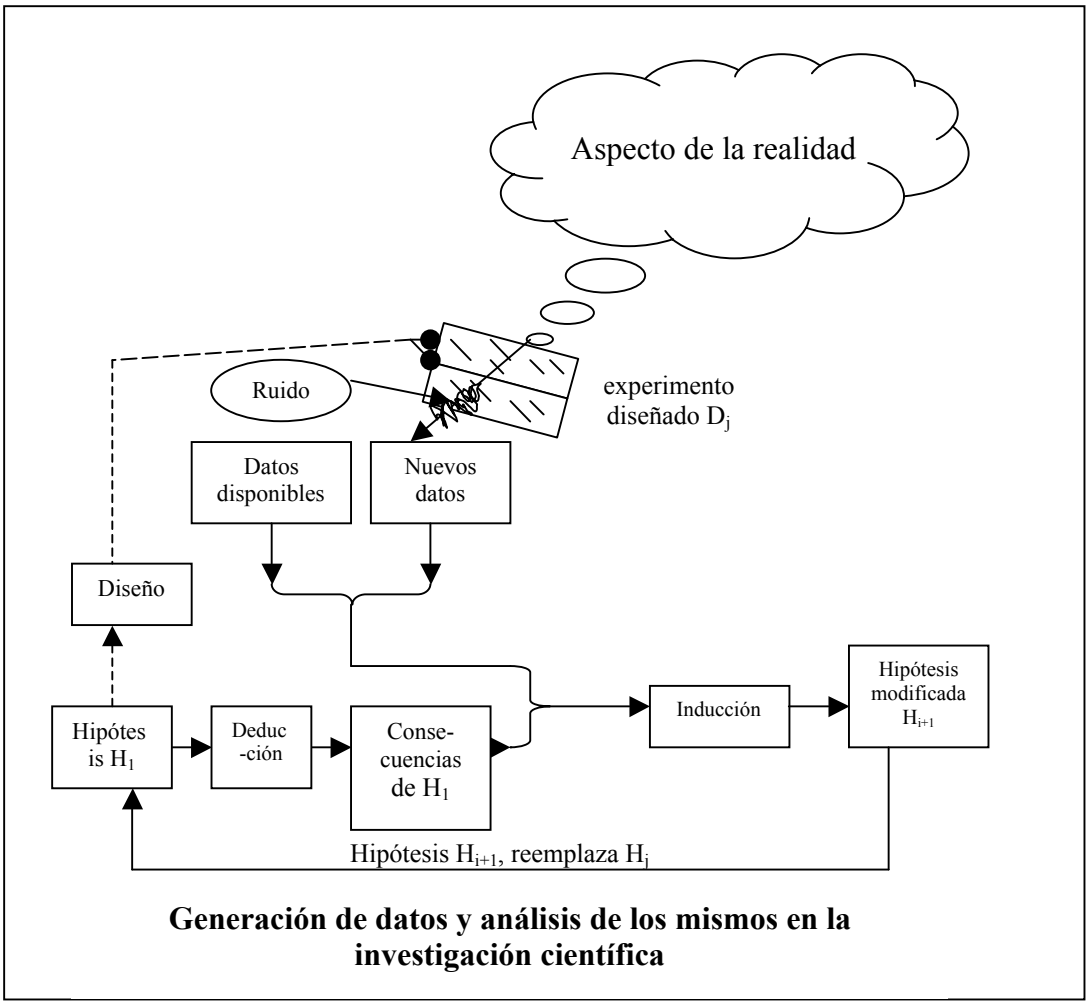
2. MARCO TEÓRICO

En este capítulo se presentan conceptos básicos necesarios para entender el presente trabajo, aquí se pueden encontrar conceptos sobre estadística paramétrica y no paramétrica, diseños experimentales, simulación y un pequeño tutorial del lenguaje de programación FORTRAN. Se explica brevemente los aspectos básicos de los diseños experimentales, contrastes, Bootstrap y sobre el Valor P (p-value), los cuales son utilizados a través de esta tesis.

2.1 El papel del Diseño Experimental

Normalmente es más eficiente estimar el efecto de varias variables simultáneamente. Cada diseño experimental contiene entonces un grupo de experimentos. Para cada ciclo iterativo no es necesario realizar un nuevo diseño. Algunas veces se utilizan los mismos datos para confrontarlos con sucesivas hipótesis. Sin embargo, cuando no se ve claramente qué modificación ha de realizarse a una hipótesis no satisfactoria, o cuando hace falta más confirmación de una hipótesis aparentemente satisfactoria, se precisarán de datos adicionales. Estos se generan con más experimentos dispuestos en un nuevo diseño experimental.

La elección del diseño depende de la hipótesis en vigor. El diseño elegido debe investigar las zonas obscuras que tenemos con el conocimiento actual del problema y cuya iluminación consideramos un avance importante. El diseño se representa por una ventana movible a través de la cual pueden observarse ciertos aspectos de la realidad más o menos distorsionados por el ruido ambiente como se muestra en la Figura 2.1.1.



Generación de datos y análisis de los mismos en la investigación científica

Figura 2.1.1

Aunque la hipótesis sobre el mundo real puede ser falsa o al menos inexacta, los datos en si están generados por esa realidad. Esta es la razón por la que el proceso de actualizar continuamente la hipótesis y comparar las deducciones acerca del mundo real con los datos obtenidos de éste puede llevarnos hacia la verdad.

Dos investigadores de la misma competencia enfrentados a un mismo problema partirán, en general, de distintos puntos, avanzando por caminos diferentes y aún así podrán llegar a la misma solución. Lo que se busca no es la uniformidad, sino la convergencia.

La convergencia hacia el resultado se producirá más rápidamente y con más seguridad si dispone de:

1. Métodos eficientes de diseño de experimentos, que le permitan obtener respuestas a sus preguntas que sean lo menos ambiguas y lo menos afectadas posible por los errores experimentales.

2. Análisis de sensibilidad de los datos, que indique lo que puede deducirse razonablemente de la hipótesis en vigor y de pie a nuevas ideas a considerar.

De estos dos recursos, el diseño del experimento es el más importante. Si el diseño experimental está mal elegido, de modo que los datos resultantes no contengan mucha información, poco se podrá extraer por muy detallado o sofisticado que sea el análisis. Por otra parte, si el diseño experimental se escoge apropiadamente, una gran cantidad de información está a su disposición y podría no ser necesario un análisis muy complicado. De hecho en muchos casos afortunados todas las conclusiones importantes se hacen patentes con sólo examinar los datos.

En cualquier investigación es de la mayor importancia: (1) definir claramente los objetivos de estudio que se va a llevar a cabo ; (2) asegurarse de que todos los interesados en el estudio están de acuerdo con esos objetivos; (3) estar de acuerdo en los criterios que determinarán que los objetivos se han alcanzado, y (4) tener previsto que, si los objetivos cambian, todas las partes interesadas lo conozcan y se pongan de acuerdo en los nuevos objetivos y criterios. (Véase Box 1988)

2.2 Diseño Factorial General

Los resultados del ANOVA para dos factores pueden ser extendidos a un caso general en donde a son los niveles del factor A, b son los niveles del factor B, c son los niveles del nivel C, y así sucesivamente, los cuales pueden ser arreglados en un experimento factorial, en el cual el número de réplicas es n. Los diseños factoriales son ampliamente

utilizados en experimentos en los que intervienen varios factores para estudiar el efecto conjunto de estos sobre una respuesta. Existen varios casos especiales del diseño factorial general que resultan importantes porque se usan ampliamente en el trabajo de investigación, y porque constituyen la base para otros diseños de gran valor práctico.

El diseño factorial fraccionario 2 k-p se usa en experimentos de escrutinio para identificar con rapidez y de manera eficiente el subconjunto de factores que son activos, y para obtener alguna información sobre la interacción. La propiedad de proyección de estos diseños hace posible en muchos casos examinar los factores activos con más detalle. (Véase ITC03)

2.3 Diseño Factorial General 2^k.

El más importante de los casos especiales ocurre cuando se tienen k factores, cada uno con dos niveles. Estos niveles pueden ser cuantitativos como sería el caso de dos valores de temperatura presión o tiempo. También pueden ser cualitativos como sería el caso de dos máquinas, dos operadores, los niveles "superior" e "inferior" de un factor, o quizás, la ausencia o presencia de un factor.

Una réplica completa de tal diseño requiere que se recopilen 2 x 2 x .... x 2 = 2^k observaciones y se conoce como diseño general 2^k.

Se supone que:

- Los factores son fijos.
- Los diseños son completamente aleatorios, en el orden de ejecución.
- Se satisface la suposición usual de normalidad.

El diseño 2^k es particularmente útil en las primeras fases del trabajo experimental, cuando es probable que haya muchos factores por investigar. Conlleva el menor número de corridas con las cuales pueden estudiarse k factores en un diseño factorial completo.

Debido a que sólo hay dos niveles para cada factor, debe suponerse que la respuesta es aproximadamente lineal en el intervalo de los niveles elegidos de los factores.

2.3.1 Diseño 2^2

El primer diseño de la serie 2^k es aquel que tiene sólo dos factores, A y B, cada uno con dos niveles. Arbitrariamente, los niveles del factor pueden llamarse "inferior" y "superior".

2.3.2 Diseño 2^3

Suponga que se encuentran en estudio tres factores A, B y C, cada uno con dos niveles. Este diseño se conoce como diseño factorial, 2^3 y las ocho combinaciones de tratamientos pueden representarse gráficamente mediante un cubo.

Existen en realidad tres notaciones distintas que se usan ampliamente para las corridas o ejecuciones en el diseño 2^k:

- La primera es la notación "+, -", llamada "geométrica".
- La segunda consiste en el uso de letras minúsculas para identificar las combinaciones de tratamientos.
- En la tercera se utilizan los dígitos 1 y 0 para denotar los niveles alto y bajo del factor, respectivamente.(Véase ITC03)

2.3.3 Contrastes

En muchos métodos de comparación múltiple se utiliza la idea de contraste. Cuando se lleva acabo un análisis de varianza por bloques y se rechaza la hipótesis nula; se sabe que algunos tratamientos producen diferencias, pero no sabemos cuales tratamientos promedios son las que producen estas diferencias.

Para poder llegar a identificar los tratamientos promedio que provocan estas diferencias, se utilizan los contrastes. En general, la comparación de medias de los tratamientos conlleva una combinación lineal de totales de tratamientos de la forma:

$$C = \sum_{i=1}^{a} C_i Y_i$$

Con la restricción de $\sum_{i=0}^{a} C_i = 0$. Tales combinaciones lineales se conocen como contrastes. (Ver Huesca 2002)

2.3.3.1 Contrastes Ortogonales

Un caso especial del procedimiento anterior es el de contrastes ortogonales. Dos contrastes con coeficientes $\{c_i\}$ y $\{d_i\}$ son ortogonales si

$$\sum_{i=1}^{a} C_i d_i = 0$$

(Consúltese Montgomery 1991)

2.4 Pruebas de hipótesis

Una hipótesis estadística es una asunción relativa a una o varias poblaciones, que puede ser cierta o no. Las hipótesis estadísticas se pueden contrastar con la información extraída de las muestras y tanto si se aceptan como si se rechazan se puede cometer un error. La hipótesis formulada con intención de rechazarla se llama hipótesis nula y se representa por H_0. Rechazar H_0 implica aceptar una hipótesis alternativa (H_1).

La situación se puede esquematizar:

	H_0 cierta	**H_0 falsa** **H_1 cierta**
H_0 rechazada	Error tipo I (α)	Decisión correcta (*)
H_0 no rechazada	Decisión correcta	Error tipo II (β)

Figura 2.4.1

(*) Decisión correcta que se busca

α = p(rechazar H_0|H_0 cierta)

β = p(aceptar H_0|H_0 falsa)

Potencia =1-β = p(rechazar H_0|H_0 falsa)

Detalles a tener en cuenta:

- α y β están inversamente relacionadas.
- Sólo pueden disminuirse las dos, aumentando n.

Los pasos necesarios para realizar un contraste relativo a un parámetro θ son:

1. Establecer la hipótesis nula en términos de igualdad

$$H_0 = \theta = \theta_0$$

2. Establecer la hipótesis alternativa, que puede hacerse de tres maneras, dependiendo del interés del investigador

$$H_1 = \theta \neq \theta_0$$

$$\theta > \theta_0$$
$$\theta < \theta_0$$

3. En el primer caso se habla de contraste bilateral o de dos colas, y en los otros dos de lateral (derecho en el 2º caso, o izquierdo en el 3º) o una cola.

4. Elegir un nivel de significación: nivel crítico para α

5. Elegir un estadístico de contraste: estadístico cuya distribución muestral se conozca en H_0 y que esté relacionado con θ y establecer, en base a dicha distribución, la región crítica: región en la que el estadístico tiene una probabilidad menor que α si H_0 fuera cierta y, en consecuencia, si el estadístico cayera en la misma, se rechazaría H_0.

   Obsérvese que, de esta manera, se está más seguro cuando se rechaza una hipótesis que cuando no. Por eso se fija como H_0 lo que se quiere rechazar. Cuando no se rechaza, no se ha demostrado nada, simplemente no se ha podido rechazar. Por otro lado, la decisión se toma en base a la distribución muestral en H_0, por eso es necesario que tenga la igualdad.

6. Calcular el estadístico para una muestra aleatoria y compararlo con la región crítica, o equivalentemente, calcular el "valor p" del estadístico (probabilidad de obtener ese valor, u otro más alejado de la H_0, si H_0 fuera cierta) y compararlo con α.(Ver HRC03)

2.5 Intervalos de confianza

En muchos experimentos en ingeniería e industria, el investigador sabe de antemano que existe diferencia entre las medias μ_1 y μ_2; por lo tanto, interesa poco probar la hipótesis $\mu_1=\mu_2$. Usualmente, el experimentador se mostrará más interesado en un intervalo de confianza para la diferencia entre las medias $\mu_1-\mu_2$.

Con objeto de definir un intervalo de confianza supongamos que θ es un parámetro desconocido. Para obtener una estimación del intervalo de θ se necesita la determinación de dos estadísticas L y U de manera que el enunciado de probabilidad

$$P(L \leq \theta \leq U) = 1-\alpha$$

Sea verdadero. El intervalo:

$$L \leq \theta \leq U$$

Se denomina intervalo de confianza de 100(1-α) por ciento para el parámetro θ. Para interpretar este intervalo debe tomarse en cuenta que si en muestras aleatorias repetidas, se consideran un gran número de estos intervalos, el 100(1-α) por ciento de ellos contendrá el valor verdadero de θ. Las estadísticas L y U se conocen como los límites inferior y superior de confianza, respectivamente, y 1-α se denomina coeficiente de confianza. Hay que notar que los intervalos de confianza poseen una interpretación de frecuencia; esto indica que no se sabe si la proposición es verdadera para una muestra específica. Lo cierto es que el método que se usa para obtener el intervalo de confianza ofrecerá proposiciones correctas en el 100(1-α) por ciento de las ocasiones. (Véase Montgomery 1991)

2.6 Bootstrap

Como es conocido un objetivo común en la investigación es realizar inferencias probabilísticas respecto a algún aspecto de interés de la población objeto de estudio. La estadística inferencial se ha ocupado de este problema estudiando cómo a partir de la información limitada que facilita un estimador $\hat{\theta}$, obtenido en una muestra aleatoria, podemos hacer afirmaciones, de tipo probabilístico, respecto al parámetro desconocido θ.

En la construcción de la inferencia estadística ocupa un papel central el reconocimiento de los estadísticos (estimadores) como variables aleatorias, y por lo tanto sometidas a fluctuaciones muestrales, lo cual conduce a uno de los elementos claves de la teoría estadística inferencial: la distribución muestral de un estadístico. Dada esa variabilidad que presentan los estimadores una de las preocupaciones del analista será la exactitud con la que las estimaciones captan al parámetro. La determinación de la exactitud de las estimaciones descansa sobre el cálculo del error típico del estadístico.

La estadística paramétrica ha abordado esta problemática apoyándose en suposiciones restrictivas respecto a las distribuciones de las variables aleatorias (p.e. normalidad, independencia) lo cual allanaba el camino pues permitía la derivación analítica de las distribuciones muestrales de los estadísticos y el cálculo de los parámetros que las caracterizan. Esta forma de trabajo ha demostrado ser una excelente estrategia cuando las suposiciones impuestas son adecuadas al tipo de datos que se generan en la disciplina en la que se va a emplear el modelo estadístico. Sin embargo, cuando los supuestos en que estos métodos se basan no concuerdan con los datos los resultados obtenidos con ellos son, lógicamente, menos de fiar. No son pocas las voces que ponen en duda que en las ciencias sociales el cumplimiento de estas suposiciones sobre distribuciones sea tan frecuente como parecería deducirse del generalizado uso de las técnicas paramétricas. Hinkle y Winstead (1990) justifican la persistencia en la utilización de la distribución normal en las ventajas computacionales que proporcionaba antes de la irrupción de los ordenadores. Solanas y Sierra (1992), muy acertadamente,

apuntan que la práctica habitual ha sido obviar el análisis de los supuestos y utilizar las técnicas estadísticas sin considerar la adecuación o no de las mismas. Por otro lado, aun cuando las suposiciones distribucionales sean admisibles la obtención del error típico, en contra de lo que podría parecer, no es posible para todos los estimadores.

Frente al enfoque tradicional han sido desarrollados, o más bien puestos de actualidad, una serie de métodos basados en cálculos intensivos por medio de ordenadores destinados a obtener medidas de la precisión de las estimaciones (errores típicos, sesgo, intervalos confidenciales). De entre tales métodos se va a hablar aquí del denominado Bootstrap, introducido por Efron en 1979.

El bootstrap se diferencia del enfoque tradicional paramétrico en que emplea un gran número de cálculos repetitivos para estimar la forma de la distribución muestral del estadístico, en lugar de fuertes suposiciones distribucionales y fórmulas analíticas. Esto permite al investigador hacer inferencias en casos donde tales soluciones analíticas son inviables, y donde tales suposiciones son insostenibles. El bootstrap no es, por tanto, un estadístico per se. Más bien, es una aproximación al uso de la estadística para hacer inferencias sobre los parámetros poblacionales.

El bootstrapping descansa en la analogía entre la muestra y la población de la cual la muestra es extraída. De acuerdo a Efron y Tibshirani (1986) dada una muestra con n observaciones el estimador no paramétrico de máxima verosimilitud de la distribución poblacional es la función de densidad de probabilidad que asigna una masa de probabilidad de $\frac{1}{n}$ a cada una de las observaciones. La idea central es que muchas veces puede ser mejor extraer conclusiones sobre las características de la población estrictamente a partir de la muestra que se maneja, que haciendo suposiciones quizás poco realistas sobre la población. El bootstrapping implica remuestreo (resampling) de los datos obtenidos en una muestra, con reemplazamiento, muchas veces para generar una estimación empírica de la distribución muestral completa de un estadístico.

El bootstrap puede considerarse como un tipo especial de simulación denominada simulación basada en los datos. Esto es, simulamos a partir de una estimación de la población basada en los datos (Efron y Tibshirani, 1993).

Lunneborg (1987) fija la utilidad del método bootstrap a tres niveles:

1. Valorar el sesgo y el error muestral de un estadístico calculado a partir de una muestra.

2. Establecer un intervalo de confianza para un parámetro estimado.

3. Realizar una prueba de hipótesis respecto a uno o más parámetros poblacionales.

2.6.1 Inferencia Estadística mediante Bootstrap.

Las técnicas de simulación agrupadas bajo el nombre genérico de bootstrap se ocupan de los mismos asuntos que la estadística paramétrica pero bajo otro enfoque. Como indican Efron y Tibshirani (1993) las ideas básicas de la estadística no han cambiado lo que ha cambiado es su implementación. La irrupción de los ordenadores ha aportado rapidez y flexibilidad en la aplicación de ciertas ideas hasta ahora aparcadas, permitiendo analizar aspectos no abordables con facilidad analíticamente (Diaconis y Efron, 1983).

La idea básica del bootstrap es tratar la muestra como si fuera la población, y aplicar el muestreo Monte Carlo para generar una estimación empírica de la distribución muestral del estadístico. La verdadera estimación Monte Carlo requiere un conocimiento total de la población, pero por supuesto éste no está generalmente disponible en la investigación aplicada. Típicamente, tenemos sólo una muestra extraída de esa población, debido a lo cual es por lo que necesitamos, antes de nada, inferir θ a partir de $\hat{\theta}$.

En el bootstrapping, tratamos la muestra como si fuera la población y realizamos un procedimiento del estilo Monte Carlo sobre la muestra. Esto se hace extrayendo un

gran número de "remuestras" de tamaño n de la muestra original aleatoriamente y con reposición. Así, aunque cada remuestra tendrá el mismo número de elementos que la muestra original, mediante el remuestreo con reposición cada remuestra podría tener algunos de los datos originales representados en ella más de una vez, y algunos que no aparecieran. Por lo tanto, cada una de estas remuestras probablemente será levemente y aleatoriamente diferente de la muestra original. Y como los elementos en estas remuestras varían levemente, un estadístico $\hat{\theta}^*$, calculado a partir de una de esas remuestras probablemente tomará un valor ligeramente diferente de los otros $\hat{\theta}^*$ y del $\hat{\theta}$ original. La afirmación fundamental del bootstrapping es que una distribución de frecuencias relativas de esos $\hat{\theta}^*$ calculada a partir de las remuestras es una estimación de la distribución muestral de $\hat{\theta}$ (Mooney y Duval, 1993). Más formalmente, los pasos básicos en la estimación bootstrap son los siguientes Efron, 1979; Hinckley, 1988; Efron y Tibshirani, 1993):

1. Construir una distribución de probabilidad empírica, $\hat{F}(x)$, a partir de la muestra asignando una probabilidad de $\frac{1}{n}$ a cada punto, x_1, x_2, ..., x_n. Esta es la función de distribución empírica (FDE) de x, la cual es el estimador no-paramétrico de máxima verosimilitud de la función de distribución de la población, F(X).

2. A partir de la FDE, $\hat{F}(x)$, se extrae una muestra aleatoria simple de tamaño n con reposición. Esta es una "remuestra", x*b.

3. Se calcula el estadístico de interés, $\hat{\theta}$, a partir de esa remuestra, dando $\hat{\theta}^*_b$.

4. Se repiten los pasos 2 y 3 B veces, donde B es un número grande. La magnitud de B en la práctica depende de las pruebas que se van a aplicar a los datos. En general, B debería ser de entre 50 a 200 para estimar el error típico de $\hat{\theta}$, y al menos de 1000 para estimar intervalos de confianza alrededor de $\hat{\theta}$ (Efron y Tibshirani, 1986, 1993).

5. Construir una distribución de probabilidad a partir de los B $\hat{\theta}^*b$ asignando una probabilidad de 1/B a cada punto, $\hat{\theta}^*_1, \hat{\theta}^*_2, ..., \hat{\theta}^*_B$. Esta distribución es la estimación bootstrap de la distribución muestral de $\hat{\theta}$, $\hat{F}^*(\hat{\theta})^*$. Esta distribución puede usarse para hacer inferencias sobre θ.

El estimador bootstrap del parámetro θ se define como:

$$\hat{\theta}^*_{(.)} = \frac{\sum_{b=1}^{B} \hat{\theta}^*_b}{B}$$

Figura 2.6.1

Es decir, como la media de los valores del estadístico calculados en las B remuestras bootstrap.

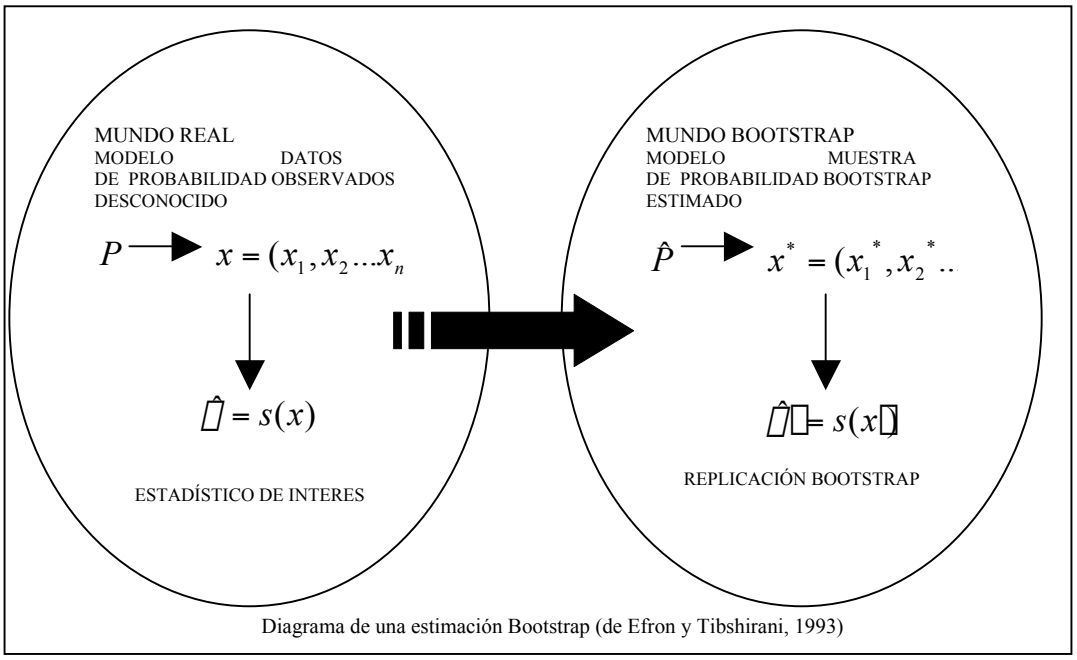


Diagrama de una estimación Bootstrap (de Efron y Tibshirani, 1993)

Figura 2.6.2

En algunas ocasiones se tiene algún conocimiento más que el estrictamente aportado por la muestra, por ejemplo se parámetros conoce la función de distribución de la variable objeto de estudio pero se desconocen los, que deben ser estimados. En estos casos se estimarían los parámetros a partir de la muestra y el remuestreo se realizaría a partir de la función teórica conocida, con los parámetros estimados, en lugar de a partir de la FDE construida a partir de la muestra. En este caso hablamos de Bootstrap Paramétrico. (Ver DAL00)

2.7 Estadística no paramétrica

Antes de discutir las técnicas no paramétricas, se debe considerar por qué las técnicas que se han venido usando se llaman técnicas paramétricas. Los parámetros son índices que etiquetan a las funciones dentro de una familia. Por ejemplo, hay un número infinito de distribuciones normales pero cada uno una de ellas está determinada de manera única por su media (μ) y su desviación estándar (σ).

Las técnicas usadas más comúnmente son llamadas paramétricas porque estas implican la estimación o prueba de los valores de un parámetro, comúnmente medias o proporciones. No es de sorprenderse, entonces, que los métodos no paramétricos son procedimientos que hacen su trabajo sin la referencia de parámetros específicos. Para muestras grandes, muchas técnicas no paramétricas pueden verse como los procedimientos comúnmente aplicados para rangos basados en teoría normal.

De manera general, una prueba no paramétrica es aquella que no hace una hipótesis acerca del valor de un parámetro en una función de densidad estadística, mientras que una prueba libre de distribución es aquella que no hace suposiciones respecto de la forma precisa de la población muestreada.

2.7.1 Rangos

Muchos de los procedimientos no paramétricos están basados en datos agrupados en rangos. Los datos se ordenan en rangos del más bajo al más alto y asignándoles en orden, los valores enteros del 1 al tamaño de la muestra. Los números empatados se resuelven asignándoles el promedio de los rangos que hubiesen recibido si no estuviesen empatados, p. e. 117, 119, 119, 125, 128 produce 1, 2.5, 2.5, 4, 5. (Si los dos 119s no estuvieran empatados se les hubiese asignado 2 y 3 respectivamente. El promedio de 2 y 3 es 2.5)

2.7.2 Uso de Software para estadística no paramétrica

El creciente acceso a las computadoras ha hecho obsoleto mucho del trabajo teórico acerca de las aproximaciones y correcciones para empates en los datos. Los empates eran un problema porque con empates, un conjunto de n observaciones no se reduce a un conjunto de rangos 1, ... , n. El conjunto de rangos particular depende del número y forma de los empates. En el pasado, se desarrollaron ajustes para el estadístico z para ajustar rangos empatados. Hoy, los valores críticos para pruebas no paramétricas exactas que involucran datos con empates se pueden calcular bajo demanda mediante programas computacionales especializados. (Ver Mehta 1992).

La mayoría de los analistas sólo pueden hacer lo que su software les permite hacer. En la medida en que las técnicas especializadas se encuentren disponibles en paquetes computacionales estándares, se usarán. Hasta entonces, los procedimientos se quedan en la repisa de las curiosidades. El creciente acceso a computadoras personales y programas estadísticos han causado una evolución en la manera da analizar datos. Estos cambios continúan con la aparición de cada nuevo paquete y actualización.

2.7.3 Ventajas de los procedimientos no paramétricos

1. Las pruebas no paramétricas requieren de datos con demandas muy estrictas. Para que los procedimientos paramétricos estándares sean válidos, se deben de cumplir ciertas condiciones subyacentes y suposiciones, en especial para tamaños de muestra pequeños.

2. Los procedimientos no paramétricos pueden ser usados para obtener una respuesta rápida sin demasiados cálculos.

3. Los métodos no paramétricos proporcionan un aire de objetividad cuando no existe una escala subyacente confiable (reconocida universalmente) para los datos originales y existe la preocupación de que los resultados las técnicas paramétricas puedan ser criticados por su dependencia en una métrica artificial. Por ejemplo, se le puede preguntar a los pacientes si se sienten muy incómodos, incómodos, neutrales, cómodos o muy cómodos. Que valores numéricos deben asignarse a las categorías de comodidad y ¿cómo se determina si los resultados cambiarían drásticamente con un ligero cambio en estos valores numéricos? Algunas de estas preocupaciones se eliminan cuando los datos se convierten en rangos.

4. Una ventaja histórica de las pruebas de rangos consiste en la facilidad para generar tablas de valores críticos exactos, si no existen datos empatados. El mismo valor crítico puede usarse para todos los conjuntos de datos con el mismo número de observaciones dado que todo conjunto de datos se puede reducir a rangos de 1, ... , n. Sin embargo esta ventaja se ha disminuido con el creciente acceso a computadoras personales.

5. Algunas veces los datos no constituyen una muestra de una población mayor. Estos datos son todos los que existen. Así que las técnicas paramétricas estándares basadas en muestreo de poblaciones mayores ya no son apropiadas. Dado que no existen poblaciones mayores, no existen parámetros de población a estimar. Sin embargo,

ciertos tipos de procedimientos no paramétricos pueden ser aplicados a dichos datos utilizando modelos de aleatorización. (Consúltese DAL00)

2.7.4 Desventajas de la Estadística no paramétrica

1. Si todos los supuestos del método estadístico paramétrico son satisfactorios realmente por los datos y si la medida tiene la fuerza requerida, las pruebas estadísticas no paramétricas pueden disipar la evidencia contra H_0.

2. Hasta ahora no hay métodos no paramétricos para probar las interacciones dentro del modelo de análisis de varianza, a menos que se hagan suposiciones especiales acerca de la aditividad. (Véase Siegel 1975)

2.8 Valor P (P-value)

La significancia estadística de un resultado es la probabilidad de que la relación observada (p.e. entre variables) o una diferencia (p.e. entre medias) en una muestra ocurran por pura casualidad (por pura suerte), y que en la población de la cual la muestra fue tomada, exista tal relación o diferencia. Usando términos menos técnicos uno puede decir que la significancia estadística de un resultado nos dice algo acerca del grado en que el resultado es "verdadero" (en el sentido de ser representativo de la población). Más técnicamente el Valor P representa un índice decreciente de confiabilidad de un resultado.

Mientras más alto sea el Valor P, menos podemos creer en que la relación observada entre variables en la muestra es un indicador confiable de la relación entre las variables respectivas en la población. Específicamente el Valor P representa la probabilidad de error que se encuentra envuelta en aceptar nuestro resultado observado como válido, en otras palabras, como "representativo de la población". Por ejemplo un Valor P de 0.05 (p.e. 1/20) indica que hay una probabilidad del 5% de que la relación entre las variables encontrada en nuestra muestra sea una "falla". En otras palabras, asumiendo que en la población no exista relación alguna entre esas variables, se puede

esperar que aproximadamente que una de cada 20 replicaciones del experimento será una en que las variables en cuestión sea igual o más fuerte que la esperada (esto no es lo mismo que decir que dado que existe una relación entre las variables, se pueden repetir los resultados 5% o 95% de las veces; cuando hay una relación entre las variables de la población la probabilidad de replicar el estudio y encontrar que la relación se conecta al poder estadístico del diseño). En muchas áreas de investigación, el Valor P de 0.05 es tratado como el "límite aceptable" de error. (Ver STA03)

2.9 Simulación

La simulación nos permite reunir información pertinente sobre el comportamiento del sistema porque ejecuta un modelo computarizado. Los datos recopilados se usan después para diseñar el sistema. La simulación no es una técnica de optimización. Más bien, es una técnica para estimar las medidas de desempeño del sistema modelado. El uso de la simulación se ha extendido a muchas áreas de la ciencia y la tecnología, como lo muestra la siguiente lista parcial:

1. Ciencia básica

    - Estimación del área bajo la curva o, de manera más general, evaluación de integrales múltiples.

    - Estimación de la constante p (= 3.14159)

    - Inversión de matrices

    - Estudio de difusión de partículas

2. Situaciones prácticas

   - Problemas industriales, incluido el diseño de sistemas de colas, redes de comunicación, control de inventarios y procesos químicos.

   - Problemas de negocios y de economía, incluidos el comportamiento del consumidor, determinación de precios, predicción económica y la operación total de una empresa.

   - Problemas de comportamiento y sociales, como la dinámica de la población, efectos del medio ambiente en la salud, estudios epidemiológicos y comportamiento de grupos.

   - Sistemas biomédicos, como el balance de fluidos, distribución de electrolitos en el cuerpo humano, proliferación de células sanguíneas y actividades cerebrales.

La estimación de la salida de la simulación se basa en un muestreo aleatorio, casi en la misma forma que cuando observamos una situación real. Esto significa que la salida de la simulación está sujeta a variaciones aleatorias, y así, como en cualquier experimento estadístico, se debe examinar con el uso de pruebas de inferencia estadística. Ver (Consúltese Taha 1997)

2.10 Tutorial de Fortran

2.10.1 ¿Qué es Fortran?

Fortran es lenguaje de propósito general, principalmente orientado a la computación matemática, por ejemplo en ingeniería. Fortran es un acrónimo de FORmula TRANslator, y originalmente fue escrito con mayúsculas como FORTRAN. Sin embargo la tendencia es poner sólo la primera letra con mayúscula, por lo que se escribe actualmente como Fortran. Fortran fue el primer lenguaje de programación de alto nivel. El desarrollo de Fortran inicio en la década de 1950 en IBM y han habido muchas versiones desde entonces.

2. 10.2 Razones para aprender Fortran

Fortran es un lenguaje de programación dominante usado en aplicaciones de ingeniería y matemáticas, por lo que es importante que se tenga bases para poder leer y modificar un código de Fortran. Algunas opiniones de expertos han dicho que Fortran será un lenguaje que pronto decaerá en popularidad y se extinguirá, lo que no ha sucedido todavía. Una de las razones para esta sobrevivencia es la inercia del software, ya que una vez que una compañía ha gastado muchos millones de dólares y de años en el desarrollo de software, no le es conveniente traducir el software a un lenguaje diferente, por el costo que implica y por ser una tarea difícil y laboriosa.

2. 10.3 Portabilidad

Una ventaja principal de Fortran es que ha sido estandarizado por ANSI y ISO, por lo que un programa escrito en ANSI Fortran se podrá ejecutar en cualquier computadora que tenga un compilador de Fortran.

ANSI = American National Standards Institute
ISO = International Standards Organization

2.10.4 Introducción a Fortran

Un programa de Fortran es una secuencia de líneas de texto. El texto debe de seguir una determinada sintaxis para ser un programa válido de Fortran. Las líneas que comienzan - con el caracter "c" son comentarios y no tienen otro propósito más que hacer los programas más legibles.

2. 10.5 Organización del Programa

Un programa de Fortran por lo general consiste de un programa principal o main (o manejador) y posiblemente varios subprogramas (o procedimientos o subrutinas). Por el momento se considerara que todas las sentencias están en el programa principal; los subprogramas se revisarán más adelante. La estructura del programa principal es:

```
program name
declarations
statements
stop
end
```

Figura 2.10.1

La sentencia `stop` es opcional y podría ser vista como redundante ya que el programa terminará cuando alcance el fin, pero se recomienda que el programa termine con la sentencia `stop` para resaltar que la ejecución del programa ahí termina.

2.10.6 Reglas para la posición en columnas

Fortran no es un lenguaje de formato libre, ya que tiene un conjunto de reglas estrictas de como el código fuente debe ser formateado. Las reglas más importantes son las reglas para la posición en columnas:

Col. 1	Blanco (espacio), o un carácter “c” o “*” para comentarios
Col 2-5	Etiqueta de sentencia (opcional)
Col. 6	Continuación de una línea previa (opcional)
Col 7-72	Sentencias
Col 73-80	Número de secuencia (opcional, raramente usado actualmente)

Figura 2.10.2

Muchas líneas de un programa de Fortran inician con 6 espacios y terminan antes de la columna 72, solamente el campo de sentencia es usado.

2. 10.7 Comentarios

Una línea que inicia con una letra "c" o un asterisco en la primera columna es un comentario. El comentario puede aparecer en cualquier lugar del programa. El colocarlos en el lugar preciso incrementan la legibilidad.

2.10.8 Continuación

Ocasionalmente una sentencia no cabe en una sola línea. Se puede dividir la sentencia en dos o más líneas, y usar la marca de continuación en la posición 6. Cualquier carácter puede ser usado en vez del signo "+" como carácter de continuación. Se considera un

buen estilo de programación usar el signo más, ampersand o números (2 para la segunda línea, 3 para la tercera y así sucesivamente).

2.10.9 Declaración y tipos de Variables

Los nombres de variables en Fortran consisten de 1 a 6 caracteres escogidos de la `a` a la `z` y de los dígitos del `0` al `9`. El primer carácter debe ser una letra. Fortran no diferencia entre mayúsculas y minúsculas, de hecho, asume que toda la entrada es con minúsculas.

Sin embargo hace poco los compiladores de Fortran empezaron a aceptar letras minúsculas. Cada variable debería ser definida con una declaración. Esto indica el tipo de la variable. Las declaraciones más comunes son:

integer
real
double precision
complex
logical
character

Figura 2.10.3

2.10.10 Asignación y Expresiones

La forma más simple de una expresión es una constante. Hay seis tipos de constantes, que corresponden con los tipos de datos que maneja Fortran.

1. Constantes enteras: Estas son números sin decimales

2. Constantes de tipo real. Estas incluyen a números de cualquier tipo

3. Notación científica: notación E se usa para números que se encuentran escritos en notación científica. Para constantes que rebasen la capacidad de almacenamiento

de un real se puede usar doble precisión. En este caso, la notación usada es el símbolo "D" en vez de "E"

4. Constantes complejas. Los cuales son indicados por un par de constantes (enteras o reales), separadas por una coma y encerrados entre paréntesis. El primer número denota la parte real y el segundo la parte imaginaria.

5. Constantes lógicas. Estas sólo pueden tener uno de dos posibles valores, que son: `TRUE. or .FALSE.`

6. Caracteres. Estas son por lo general usadas como un arreglo de caracteres, llamado cadena. Las expresiones más sencillas son de la forma:

operando	operador	operando

Figura 2.10.4

La precedencia de los operadores aritméticos en Fortran es (de la más alta jerarquía a la más baja):

**	exponenciación
*	multiplicación
/	división
+	suma
-	resta

Figura 2.10.5

Una asignación tiene la forma

```
nombre de variable = expresión
```

Figura 2.10.6

La interpretación es como sigue: se evalúa el lado derecho y se asigna el valor del resultado a la variable de la izquierda. La expresión de la derecha puede contener otras variables, pero estas nunca cambiaran de valor.

2.10.10.1 Expresiones Lógicas

Una expresión lógica puede tener solamente el valor de `.TRUE.` o de `.FALSE.`. Una valor lógico puede ser obtenido al comparar expresiones aritméticas usando los siguientes operadores relacionales:

.LT.	<
.LE.	<=
.GT.	>
.GE.	>=
.EQ.	=
.NE.	/=

Figura 2.10.7

Por lo que no se pueden usar símbolos como < or = para comparación en Fortran, por lo que se tienen que usar abreviaturas de dos letras encerradas con puntos. Las expresiones lógicas pueden ser combinadas con los operadores lógicos `.AND.` `.OR.` `.NOT.` que corresponden a los operadores lógicos conocidos Y, O y negación respectivamente.

2.10.11 La sentencia `if`

Una parte importante de cualquier lenguaje de programación son las sentencias condicionales. La sentencia más común en Fortran es `if`, la cual tiene varias formas de uso.

La forma más simple de la sentencia `if` es:

```
if (expresión lógica) sentencia
```

Figura 2.10.8

Lo anterior tiene que ser escrito en una sola línea. Si más de una sentencia necesita ser ejecutada dentro de la sentencia `if`, entonces la siguiente sintaxis deberá ser usada:

```
if   (expresión
lógica) then
   sentencias
endif
```

Figura 2.10.9

El flujo de ejecución es de arriba hacia abajo. Las expresiones condicionales son evaluadas en secuencia hasta que se encuentra una que es verdadera. Entonces el código asociado es ejecutado y el control salta a la siguiente sentencia después de la sentencia `endif`.

2.10.12 Ciclos

Para la repetir la ejecución de sentencias se usan los ciclos. Si se esta familiarizado con otros lenguajes de programación se habrá escuchado de los ciclos-for y de los ciclos-until, Fortran tiene solamente una construcción de ciclo, conocida como el ciclo `do`. Otros ciclos pueden ser simulados usando las sentencias `if` y `goto`.

2.10.12.1 Ciclos DO

El ciclo-`do` es usado para repetir un conjunto de sentencias una determinada cantidad de veces. Se muestra el siguiente ejemplo donde se calcula la suma de los enteros desde el 1 hasta n (suponiendo que a n se le ha asignado un valor previamente):

```
integer i, n, suma
            :
            :
            :
            suma = 0
            do 10 i = 1, n
            suma = suma + i
            write(*,*) 'i =', i
             write(*,*) 'suma =', suma
      10 continue
```

Figura 2.10.10

La variable en la sentencia `do` es incrementada en 1 por default. Sin embargo, se puede usar cualquier otro entero para el paso o incremento. La forma general del ciclo `do` es la siguiente:

```
do etiqueta  var =  expr1, expr2, expr3
 sentencias
etiq continue
```

Figura 2.10.11

donde:

- var es la variable del ciclo (conocida con frecuencia como el índice del ciclo) el cual deberá ser del tipo integer.
- expr1 indica el valor inicial de var,
- expr2 es el valor hasta el que llegará el índice, y
- expr3 es el incremento (step).

2.10.12.2 Ciclos `WHILE`

La forma más intuitiva para escribir un ciclo `while` es

```
while (expr lógica) do
                    sentencias
enddo
```

Figura 2.10.12

o de forma alterna

```
do while (expr lógica)
          sentencias
 enddo
```

Figura 2.10.13

2.10.13 Arreglos

Muchos cálculos científicos usan vectores y matrices. El tipo de dato usado en Fortran para representar tales objetos es el array. Un arreglo unidimensional corresponde a un vector, mientras que un arreglo bidimensional corresponde a una matriz. Para entender como son usados en Fortran, no solamente se requiere conocer la sintaxis para su uso, sino también como son guardados estos objetos en la memoria.

2.10.14 Subprogramas

Cuando un programa tiene más de cien líneas, es difícil de seguir. Los códigos de Fortran que resuelven problemas reales de ingeniería por lo general tienen decenas de miles de líneas. La única forma para manejar códigos tan grandes, es usar una aproximación modular y dividir el programa en muchas unidades independientes pequeñas llamadas subprogramas.

Un subprograma es una pequeña pieza de código que resuelve un subproblema bien definido. En un programa grande, se tiene con frecuencia que resolver el mismo subproblema con diferentes tipos de datos. En vez de replicar el código, estas tareas pueden resolverse con subprogramas. El mismo subprograma puede ser llamado varias veces con distintas entradas de datos. En Fortran se tienen dos tipos diferentes de subprogramas, conocidas como funciones y subrutinas.

2.10.14.1 Funciones

Las funciones en Fortran son bastante similares a las funciones matemáticas: ambas toman un conjunto de variables de entrada (parámetros) y regresan un valor de algún tipo. Al inicio de la sección se comento de los subprogramas definidas por el usuario, pero Fortran tiene también funciones incorporadas. Hay varias funciones incorporadas en Fortran. Algunas de las más comunes son:

Abs	valor absoluto
min	valor mínimo
max	valor máximo
sqrt	raíz cuadrada
sin	seno
cos	coseno
tan	tangente
atan	arco tangente
exp	exponencial (natural)
log	logaritmo (natural)

Figura 2.10.14

La sintaxis general de una función en Fortran es:

```
tipo function nombre (lista_de parámetros)
declaraciones
sentencias
return
end
```

Figura 2.10.15

La función es llamada simplemente usando el nombre de la función y haciendo una lista de argumentos entre paréntesis.

2.10.14.2 Subrutinas

Una función de Fortran puede devolver únicamente un valor. En ocasiones se desean regresar dos o más valores y en ocasiones ninguno. Para este propósito se usa la construcción subrutina. La sintaxis es la siguiente:

```
subroutine nombre (lista_de_parámetros)
declaraciones
sentencias
return
end
```

Figura 2.10.16

Observar que las subrutinas no tienen tipo y por consecuencia no pueden hacerse asignación al momento de llamar al procedimiento. Se debe observar que hay dos bloques de declaración de variables en el código. Primero, se declaran los parámetros de entrada/salida, es decir, las variables que son comunes al que llama y al que recibe la llamada. Después, se declaran las variables locales, esto es, las variables que serán sólo conocidas dentro del subprograma. Se pueden usar los mismos nombres de variables en diferentes subprogramas.

2.10.15 Sentencia Format

En las secciones anteriores se ha mostrado el formato libre de entrada/salida. Éste caso sólo usa una reglas predefinidas acerca de como los diferentes tipos (integers, reals, characters, etc.) serán mostrados. Por lo general un programador desea indicar algún formato de entrada o salida, por ejemplo, el número de decimales que tendrá un número real. Para este propósito Fortran tiene la sentencia format. La misma sentencia format puede ser usada para la entrada o salida.

Sintaxis

```
write(*, etiqueta) lista_de_variables
etiq format códigos_de_formato
```

Figura 2.10.17

La sentencia format puede estar en cualquier lugar dentro del programa. Hay dos estilos de programación: agrupar por parejas las sentencias (como en el ejemplo), o poner el grupo de sentencias format al final del (sub)programa.

2.10.16 Códigos comunes de formato

Las letras para códigos de formato más comunes son:

A	Cadena de Texto
D	Números de doble precisión, notación científica
E	Números reales, notación científica
F	Números reales, formato de punto fijo
I	Entero
X	Salto horizontal (espacio)
/	Salto vertical (nueva línea)

Figura 2.10.18

2.10.17 Sugerencias de depuración

Se ha estimado que cerca del 90% del tiempo que toma desarrollar un software comercial se usa en depurar y probar. Lo anterior no dice lo importante que es escribir buen código desde el primer momento. Todavía, se deben descubrir los errores (bugs).

2.10.18 Algunos errores comunes

Se muestran algunos errores comunes que se deben vigilar:

1. Asegurarse que las líneas termina en la columna 72. El resto será ignorado.
2. ¿Se corresponde la lista de parámetros con la lista de argumentos que se están pasando?
3. ¿Se corresponden los bloques comunes?
4. ¿Se esta haciendo división entera cuando se quiere división real?

2.10.19 Herramientas de depuración.

Si se tiene un error, se debe intentar localizarlo. Los errores de sintaxis son fáciles de hallar. El problema es cuando se tienen errores en tiempo de ejecución. La forma vieja para encontrar errores es agregar sentencias write en el código y tratar de llevarles la pista a las variables. Esto es un poco tedioso ya que se debe recompilar el código fuente cada vez que se haga algún cambio. Actualmente se pueden usar los depuradores (debuggers) que son una herramienta muy conveniente. Se puede avanzar en pasos a través del programa, ya sea línea por línea o se pueden colocar puntos de interrupción (breakpoints). También se pueden mostrar los valores de las variables que se quieran observar, entre varias tareas. (Ver FOR02)