Dokumentation für Programmierer zum Statistikprogramm `statist`

Dirk Melcher, Institut für Umweltsystemforschung , Uni Osnabrück
Artilleriestr. 34, 49069 Osnabrück
email: dmelcher@ramses.usf.uni-osnabrueck.de

Version vom 25.5.95
Achtung! Diese Dokumentation enthält veraltete Informationen!

1 Einleitung

Das Programm `statist` besteht aus den Modulen `statist.c, menue.c, data.c, funcs.c, procs.c` und `plot.c`. Zu jeder C-Datei gehört außerdem eine entsprechende Headerdatei.

Wie in der Einleitung bereits verkündet, hat das Programm den Anspruch, einigermaßen leicht erweiterbar zu sein. Hierzu sei zunächst die Philosophie des Programmes geschildert:

Im Programm wird streng zwischen den eigentlichen Rechenprozeduren und den Routinen, welche Datenmanagement, Menüführung und Dialog durchführen, unterschieden. Deswegen sollten *alle* Variablen in den Rechenprozeduren lokal sein! Das Programm muß die Rechenroutienen also mit den fertig zusammengestellten Daten versorgen. Die Rechenprozeduren übernehmen die eigentliche Rechenarbeit und schreiben das Ergebnis auf den Bildschirm.

2 Kompilieren

`statist` kann sowohl für UNIX als auch DOS kompiliert werden.

Für UNIX wird der gcc-Compiler vorrausgesetzt, der K&R cc-Compiler tut's nicht, da das Programm in Ansi-C geschrieben ist. Das Makefile für `gcc` heißt `makefile.unx`.

Unter DOS kann das Programm z.B. mit dem Kommandozeilen-Compiler `bcc` von BORLAND (Makefile `makefile.bcc`) oder mit der Portierung des gcc-Compilers `djgpp` (Makefile `makefile.djg`) kompiliert werden. (`djgpp` erstellt 32-Bit-Code für 386'er. Mit einer 386'er Version von `statist` lassen sich wesentlich größere Datenmengen bearbeiten.) Das Programm wird dann jeweils mit

`make -f<Makefile>` bzw.

Das utility `make`, welche das Makefile abarbeitet, ist sowohl im BORLAND-Paket als auch als gnu-utility erhältlich.

3 Ausgabeprozeduren

Um die Ausgabe von beliebeigen Meldungen `statist` flexibel handhaben zu können (Mitschreiben von Protokolldateien usw.), gibt es drei verschiedene Ausgabefunktionen, die im Stil von `printf` aufgerufen werden:

1. `out_d`: Ausgabe von Dialogtexten, daß heißt also, die Menütexte, Eingabeaufforderungen und alles andere, was nicht mit den eigentlichen Berechnungen zu tun hat.

2. `out_r`: Ausgabe aller Ergebnistexte, also alles, was mit den eigentlichen Berechnungen zu tun hat und in dem Modul `procs.c` steht.

3. `out_err`: Ausgabe aller Fehlermeldungen. Im Gegensatz zu den beiden anderen Routinen wird außer den 'normalen' `printf`-Argumenten zusätzlich als erstes Argument eine Fehlerkonstante übergeben, die die Art des Fehlers charakterisiert: `WAR` wird als Warnung ausgegeben, `ERR` ist ein 'normaler' Fehler, `MAT` ist ein Fehler und `MWA` eine Warnung innerhalb einer Rechenprozedur (werden mit ins Protokoll geschrieben) und `FAT` ist ein 'fataler' Fehler, der einen sofortigen Abbruch des Programmes bewirkt. Danach folgen wie üblich der Formatstring und die restlichen Argumente.

4 Menü

Will man eine Statistikprozedur hinzufügen, so schaut man zuerst, in welches Untermenü (Datei `menue.c`) sie am besten hineinpaßt. Momentan gibt es lediglich die Menüprozeduren `data_menu(), regress_menu(), test_menu()` und `misc_menu()`. Will man z.B. eine Testprozedur hinzufügen, so sucht man die Prozedur `test_menu()` und fügt mit einer fortlaufenden Nummer den Namen der neuen Prozedur zum Menütext hinzu, also z.B.

```
out_d(" 5 = U-Test von Mann und Whitney\n");
```

Sodann geht man in die `switch` Anweisung der Menüprozedur und fügt die entsprechende `case`-Marke für die hinzugefügte Nummer an (in unserem Beispiel also `case 5`). Hier werden dann die Dialoge geführt und die eigentliche Rechenprozedur aufgerufen.

5 Dialoge

Die Dialoge befinden sich ebenfalls in der Datei `menue.c`. Wie man wahrscheinlich gesehen hat, sind die Dialoge sehr primitiv. Es gibt allerdings ein Feature des Programmes, welches die Dialoge leicht verkompliziert: die Möglichkeit, daß der Benutzer jederzeit durch drücken der Returntaste einen Dialog abbricht und wieder ins Menü zurückkehrt.

Oft beschränkt sich der Dialog darauf, den Benutzer zu fragen, welche Spalten er welchen Variablen zuordnen möchte. Dabei kann man zwei Fälle unterscheiden:

1. Die Anzahl der Variablen (und damit der Spalten) ist festgelegt, z.B. 2 Spalten bei dem einfachen t-Test.

2. Die Anzahl der Spalten ist variabel, wie dies z.B. bei der Erstellung der Korrelationsmatrix der Fall ist.

Meistens tritt Fall 1 ein. Will man also für den U-Test zwei Spalten einlesen, dann wird einfach die Prozedur `getcols(2)` aufgerufen.

Bei jedem Einlesevorgang wird die globale bool'sche Variable `empty` neu gesetzt. Wird bei einer Eingabe nur die Entertaste gedrückt, dem Programm also eine Leerzeile übergeben, dann wird `empty` auf *true* gesetzt, sonst auf *false*. Deswegen sollte immer, wenn direkt oder über eine Prozdeur (wie bei `getcols`) eine Eingabeaufforderung stattgefunden hat, überprüft werden, ob die Eingabe 'leer' war oder nicht, um dem Benutzer die Möglichkeit zu geben, den Dialog jederzeit abzubrechen. Daher würde in unserem Beispiel der Aufruf einer Prozedur `u_test` folgendermaßen aussehen:

```
case 5: {
    getcols(2);
    if (!empty) {
       u_test(xx[acol[0]],nn[acol[0]], xx[acol[1]],nn[acol[1]]);
    }
}
break;
```

Die Namen der Variablen, die `u_test` übergeben werden müssen, werden im folgenden Abschnitt besprochen.

Bei vielen Statistikprozeduren ist es notwendig, daß die Spalten alle gleich viele Werte haben müssen. Dies kann mit der Funktion `equal_rows` getested werden. In diesem Fall kann man also die Zeile

```
 if (!empty) {
```

aus dem letzten Beispiel durch die Bedingung

```
 if ((!empty) && (equal_rows(2))) {
```

ersetzen (wenn z.B. 2 Spalten auf gleiche Anzahl überprüft werden sollen).

Falls die Anzahl der Spalten, die von einer Rechenprozedur benötigt werden, variabel ist, gestaltet sich der Dialog komplizierter. In diesem Falle kann man z.B. den Dialog für den Aufruf der Funktion `correl_matrix()` im Regreß-Menü übernehmen.

Muß man irgendwelche Parameter direkt vom Benutzer erfragen, so sollte man dies in einer bestimmten Form tun: Es gibt eine globale String-Variable `line`, in der prinzipiell alle Benutzereingaben abgespeichert werden. Dies sollte immer mit Hilfe einiger Makros erfolgen, die am Anfang des Programmes definiert sind. In der Regel wird eine Abfrage innerhalb eines `switch`-Statements erfolgen. Dann verwende man das Makro `GETBLINE`. Dieses Makro ließt die Benutzereingabe in `line` ein und bricht den Dialog ab, falls die Eingabe leer war. Will man eine Zahl vom Benutzer eingeben lassen, so verwende man nach Aufruf des Makros `GETBLINE` die Funktion `getint()` bzw. `getreal()` (letztere für `double` Zahlen). Diese Funktionen überprüfen, ob eine gültige Zahl eingegeben wurde. Sollen einzelne Zeichen eingelesen werden (z.B. 'j' oder 'n' bei ja/nein Verzeigungen), so tue man dies nach Aufruf des Makros `GETBLINE` mit `sscanf(line, "%c", &antwort)` oder sowas.

Die eigentliche Rechenprozedur sollte in die Datei `procs.c` eingefügt werden. Der Funktionsprototyp sollte als `extern` deklariert in die Headerdatei `procs.h` geschrieben werden. `funcs.h` gibt einen Überblick über bereits vorhandenen Funktionen, die mit benutzt werden können.

6 Variablen und Parameter

Alle globalen Variablen befinden sich als `extern` deklariet in `statist.h`. Außerdem befinden sich hier auch die Typ-Definitionen.

Die Spalten werden in den Array-Variablen `xx` gespeichert. Die Größe jeder Spalte wird in dem Array `nn` gespeichert. Die Spalte `xx[0]` enthält also `nn[0]` Werte. Durch den Aufruf von `getcols` werden lediglich den Spalten Variablen zugeordnet. Es soll ja z.B. dem Benutzer möglich sein, die Spalte 2 der Variable 1 einer Prozedur zuzuordnen. In der Array-Variablen `acol` wird diese Zuordnung gespeichert. Nehmen wir als Beispiel die Prozedur `u_test`: Diese Prozedur würde vier Parameter benötigen, nämlich zwei Feldvariablen `x` und `y`, welche die eigentlichen Beobachtungswerte enthalten, und `nx` und `ny`, die angeben, wie groß diese beiden Felder sind. Zunächst wird also `getcols(2)` aufgerufen, um den Dialog auszuführen, welcher die Zuordnung der Spalten zu den Variablen übernimmt. Daher lautet der Aufruf also

```
u_test(xx[acol[0]],nn[acol[0]], xx[acol[1]],nn[acol[1]]);
```

Es werden also *nicht* die Spalten 0 und 1 übergeben, sondern die, welche der Benutzer vorher der ersten und zweiten Variable zugeordnet hat!

Um die Datentypen etwas flexibler zu halten, wurde für Gleitkommazahlen (also für „normale“ Werte) der Typ `REAL` eingeführt, der momentan auf den Standardtyp `double` gesetzt ist. Aus Konsistenzgründen sollte man also immer den Typ `REAL` verwenden. Die Deklaration für die Funktion `u_test` sollte also etwa folgendermaßen aussehen:

```
void u_test(REAL x[], int nx, REAL y[], int ny);
```

Ist die Anzahl der Variablen, die einer Prozedur übergeben werden sollen, nicht festgelegt, so wird die Sache komplizierter. In diesem Falle gibt es einen Typ `PREAL`, der einen Pointer auf ein `REAL`-Array, also im Prinzip eine Spalte darstellt. `PREAL` ist folgendermaßen definiert:

```
typedef double REAL;
typedef REAL* PREAL;
```

Deklariert man nun ein Feld vom Typ `PREAL`, so hat man eine zweiminensionale Matrix:

```
PREAL  xx[MCOL];
```

Also muß eine zweidimensiomale Matrix immer als `PREAL x[]` einer Prozedur übergeben werden. Zusätzlich sollte dann noch die Anzahl der Spalten und Spalten angegeben werden. Der ganze Hokuspokus mit `PREAL` wird übrigens deswegen gemacht, um einer Prozedur eine Matrix als Parameter übergeben zu können, deren Spaltengröße vollkommen variabel ist. Wie in so einem Fall die Spalten den Variablen zugeordnet werden, schaue man sich anhand des Aufrufes der Prozedur `correl_matrix` an.

7 Fazit

Hier noch mal eine kurze Zusammenfasung, was zu tun ist, um `statist` um eine Prozedur zu erweitern:

1. In `procs.h` die Prozedur `extern` deklarieren (hierbei die Typen `REAL` bzw. `PREAL` verwenden).
2. In `procs.c` neue Prozedur definieren.
3. In `menue.c` das Menü suchen, in welches die neue Prozedur am besten hineinpaßt und den Menütext für die neue Funktion hineinschreiben
4. Im selben Menü die entsprechende `switch`-Anweisung suchen, die neue `case`-Marke eintragen und dort den Dialog und den Aufruf für die neue Prozedur hineinschreiben.

Noch eine Bitte am Schluß: Wenn jemand das Programm erweitert hat, wäre es *sehr* nett, wenn er mir die neue Version auch zukommen lassen würde!