Center for Statistics
Copenhagen Business School

July 2007
Tue Tjur

Introduction to ISUW

— the Windows version of Interactive StatUnit —

ISUW is a statistics package, a Delphi 5 application based on a collection of Turbo Pascal (later Borland Pascal) units (i.e. procedure libraries, roughly) for statistical analysis, developed from around 1990 under the name StatUnit.

ISUW is distributed free of charge and without any support obligations from my side. Accordingly, I take no responsibility for errors in the program. But I would certainly like to hear about them, and correct them if I can.

The latest version of ISUW can be downloaded from my download page

`ezlearn.cbs.dk/stat/hamat-2/tt/`

where other useful stuff can also be found.

In this paper I give a short description of the way ISUW functions and the procedures that are available. Notes on installation and directory structure are given in the section beginning on page 13. The full description of all commands can be found on the ISUW "on–line" help file ISUWHELP.TXT.

Tue Tjur
Copenhagen Business School
The Statistics Group
Solbjerg Plads 3
DK–2000 Frederiksberg
e-mail tuetjur@cbs.dk

How to run ISUW interactively — a brief summary.

ISUW is a statistics package or language. Just to place it somewhere in the landscape, it is far from the S+ family and far from SAS and SPSS. ISUW has some similarity with earlier versions of GENSTAT, in beeing exclusively command driven and not even pretending to help users who don't know what they want to do. ISUW's documentation is, I believe, rather exhaustive on the technical level, but I would not recommend anyone to use it as a textbook in statistics.

Compared to GENSTAT, I consider ISUW a lot more friendly in its user interface and command syntax. But also less general, in many respects. The limitations of ISUW are perhaps best outlined by some examples of what it can *not* do. ISUW can not read an Excel spreadsheet or a SAS dataset. The only communication available to other programs goes via text files in free format. Multivariate methods, like factor analysis, estimation in general variance/covariance component models or general matrix calculus are not implemented in ISUW. Also, it has no specialised tools for time series analysis, other than the obvious possibility of using lagged versions of the response variable as explanatory variables in a regression model. Phrases like data mining, neural networks, correspondence analysis or automatic model selection can not be found in the manual.

When you enter ISUW you will probably find it very different from most other Windows programs. There are no buttons on the screen, no small pictures of wastepaper baskets, pencils, post boxes or rotating bananas. There is only a Danish flag which is placed on the "front cover" without any obvious purpose; it will not switch language or bring you to Denmark or anywhere else if you click on it.

A general principle behind the design of ISUW — perhaps a reaction against some tendencies in the opposite direction — is that there should not be more on the screen than necessary at any time (with the above mentioned Danish flag as an obvious exception). As a consequence of this, ISUW is essentially mouse free. You can use the mouse to resize or move the ISUW window, and it works as usual when you are editing a program or selecting from a menu. But the buttons that control ISUW are not placed on the screen, they can be found on the keyboard where they are a lot easier to hit.

No rule without exceptions, and a useful exception is this. If you drag the mouse cursor into a window on the screen, a small message with summary hints (in particular concerning the keys you can use) will pop up. When, at some point, this becomes more irritating than useful, press Shift–F1 to switch this device off.

In the entry dialog for selection of working directory, move around in the directory tree by the cursor arrows. The Up/Down arrows work in the obvious way, the Left/Right arrows shift between "show siblings" and “show children”. The space bar can be used for both. Press Return when the desired working directory is highlighted. If you want to create a new working directory, press Escape instead of Return, edit the selected directory name and press Return. Notice that the working directory must be a subdirectory of the ISUW root directory (probably C:\ISUW, if you have followed my advices on installation).

To select the same working directory as last time — which is what you do most of the time — this dialog can be skipped by Return.

The first time you enter ISUW, we recommend that you select DEMOS as your working directory. This directory has a special status as the place where the ISUW demonstration programs can be executed. These programs are more or less self–explanatory, and represent a very easy way of getting started. But like all such demos they are extremely boring. Some may prefer to do it the hard way, by trial and error and reading the on–line manual.

In general, the Escape key is used when you leave a window or dialog box. Sometimes, in particular in dialogs, you can also leave by Return. In this case Return means “perform action”, if there is anything to perform, Escape means “leave without performing”.

In interactive mode, commands are written line by line in the command field in the bottom of ISUW's front window and executed when Return is pressed. In addition to standard editing keys, the following keys are active.

F1 displays ISUW's “on–line” help file. This is a single plain text file, not a web–like tree of help sheets. The first time you press F1, read the `HELP ON HELP` section in the top of file.

Ctrl–F1 displays the full command description for the command that is written in the command field. If a command has not been written, a menu for selection of command is displayed.

Escape clears the command line.

Escape from an empty command line allows you to `QUIT`.

Ctrl–Return truncates the command line from the cursors position.

Cursor Up/Down allows you to reuse (and reedit) earlier commands stacked in the upper window.

Cursor Right/Left with Ctrl completes/replaces command names lexicographically when the cursor is in the first connected word starting in position 1.

Cursor Right/Left with Ctrl completes/replaces vector names lexicographically when the cursor is in a connected word starting after position 1.

In addition you can define your own shortkeys, see the description of the `KEYS` command.

When a command is written from position 1 of the command field, the command name is completed automatically as soon as it is unique. Also, some standard beginnings of commands are completed, like `I` → `INCLUDE`, `LI` → `LIST`, `OP` → `OPEN`, `SA` → `SAVE` and `SK` → `SKIP`. This means that the keys you must use to write `INCLUDEALL` are actually `IA` (here you can even use `A` alone, as a predefined shortkey), whereas to write `SKIPLINE` you must type `SKL`. You will soon learn this, it is impossible to write anything else than a command from position 1.

You can also import a command to the command field from the command list associated with the help pages. Press F1 twice, select command with Cursor Up/Down or the mouse, and press BackSpace or LeftArrow.

With a blank in position 1, the command line is interpreted as a `COMPUTE` command.

The general syntax is

```
COMMAND [parameter1 [parameter2 [...]]]
```

Thus, parameters are in general separated by blanks. In special cases (`PLOT, TABULATE, ...`) a "pseudo blank" |, which can be written by the 1/2 key in the upper left corner of the keyboard, is used for subdivision of parameters.

When a command produces output, this is shown in a (light green) preview window. Leave this window by Return if you want output appended to the sessions output file, Escape if you don't. Use the command `SHOW` (without parameters) to look at the sessions output file.

It is possible, and often more convenient, to write an ISUW program (i.e. a sequence of ISUW commands, each occupying a line) and execute it by a `RUN` command. See the command descriptions for `EDIT` and `RUN`. In this case all output is written to the output file without any previewing. It is also possible to execute a program directly from the program editor by F9. After this, press F10 to display the output file.

A simple example.

Suppose we have an ASCII (plain text) file `EX1.TXT` of the form

```
 Dose   Response
0.968          0
0.909          1
...
1.689          1
0.524          0
```

consisting of a heading and 415 lines, each containing a value of a covariate x and a binary response y. This could be data from a classical dose–response study where 415 animals have been given a dose x of some drug, y being the binary response, e.g. 1 for reaction, 0 for no reaction. The following commands read this data set and fits a standard logit linear model with the base 10 logarithm of x as the independent variable, then fits the model with slope zero and performs the likelihood ratio test for the corresponding hypothesis of "no drug effect".

```
VAR X Y 415              { declares variates to hold data    }
OPEN EX1.TXT             { opens data file for input         }
SKIPLINE                 { skips heading line                }
READ X Y                 { reads the variates in parallel    }
COMPUTE LOG10X=LN(X)/LN(10)  { computes base 10 log(X)       }
FITLOGIT Y=1+LOG10X      { fits logistic regression model    }
LISTPARAMETERS           { lists parameter estimates         }
FITLOGIT Y=1             { fits reduced model                }
TEST                     { tests last against previous model }
```

In this example, output will be produced by the commands `FITLOGIT`, `LISTPARAMETERS` and `TEST`. Output from the two `FITLOGIT` commands contains information about the number of observations, the number of parameters estimated and the value of the $-2\log(\text{likelihood})$. Most of the information that you usually want after fit of a model (parameter estimates, fitted values, residuals, estimates of contrasts etc.) are extracted by other commands that refer to the last model fit command. `LISTPARAMETERS` is an example of such a command. The output from `LISTPARAMETERS` is a list of parameter estimates, their estimated standard deviations, the usual U–statistics (estimates divided by standard deviations, the Wald statistics) and the corresponding two–sided P–values for test of hypotheses of the form "parameter = 0". The output from `TEST` contains the likelihood–ratio test ($-2\log Q$) statistic for test of the last fitted model against the one before and the corresponding P–value under the usual χ^2–approximation.

Quite generally, output from ISUW commands is restricted to the necessary minimum. In this respect ISUW is very similar to (and perhaps even more extreme than) GENSTAT, and very different from SAS and SPSS.

Variates and factors.

The basic structures in ISUW are called *vectors*. A vector can be a *factor*, an array of bytes for storage of qualitative variables, or a *variate*, an array of single precision real numbers (7–8 significant digits) for storage of quantitative variables. Variates and factors are created by the commands `VARIATE` and `FACTOR`. In addition to this, variates (and in a few cases also factors) are often declared implicitly, for example by `COMPUTE` commands, like `LOG10X` in the example above (top of page 5).

EXAMPLE. To declare two variates `X` and `Y` of length 100, write

```
VAR X Y 100
```

Notice that we have written `VAR`, not `VARIATE`. Actually `V` would have been enough here. In ISUW programs, commands can be truncated as long as they are unambiguous, and in the command field the same rule applies since commands are completed automatically when they are recognised.

A value of a variate can be *missing*, which internally means that it has the value `-1.0E-37`. Missing values are recognised by many commands and treated appropriately as such.

A factor has, apart from its length, a property called its *number of levels*, an integer from 1 to 255 which specifies the maximal level allowed. This is given as an additional parameter in the declaration.

EXAMPLE. To declare a factor `SEX` of length 100 on 2 levels, write

```
FACTOR SEX 100 2
```

Names of vectors can be of length up to 8. The first character must be a letter `A..Z` or an underbar `_` , the remaining characters can also be digits `0..9`. Actually, the special characters `#, &, %, $` and `@` can also be used (even as first characters), but in general we recommend that you don't, because these characters are sometimes used for special purposes by ISUW. For example, vectors starting with a dollar sign have the special property that they are always deleted at exit from a program.

Vector names are case insensitive. In output from ISUW they are usually written in capitals. The length of a vector can in principle be up to around 2–3 billions (the upper limit for "long integers"), but since its values are stored in the computers RAM the realistic upper limit is 1–10 millions, depending on your computer and the number of vectors you need. At most 255 vectors can be in use simultaneously.

Vectors can be removed from memory (to save capacity or to release their names) by the command `DELETE`, and their names can be changed by the command `RENAME`.

Input from text files.

The `OPENINFILE`, `READ`, `SKIPITEM` and `SKIPLINE` commands are designed for input from ASCII (plain text) files in free format.

EXAMPLE. An ISUW program dealing with a data set of 178 units to be read from a file F:\HEIGHTS.DAT might begin something like this.

```
FACTOR SEX 178 2
VARIATE AGE HEIGHT 178
FACTOR GROUP 178 6
OPEN F:\HEIGHTS.DAT
READ * SEX AGE HEIGHT GROUP
```

This READ command assumes that the file has the data in standard format, like

```
001  1  23.1  178.2  1
002  2  43.6  173.1  4
 ...
```

where the first unit (or here, person) is a male (SEX=1), of age 23.1, etc. etc. The only separators allowed are blanks, newline symbols (and, in fact, any control characters in the range 0–31), commas and semicolons. The asterix in the READ command implies that the first item for each unit (here the unit or line number) is skipped.

The READ command above assumes that factor levels are represented by their numerical levels. If this is not the case, an equality sign followed by a comma separated list of level names can be appended to the factor name. For example,

```
READ * SEX=*,Male,Female AGE HEIGHT GROUP
```

would work if the file looked like this.

```
001  Male    23.1  178.2  1
002  Female  43.6  173.1  4
 ...
114  *       32.9  167.0  2
 ...
```

with level 1 of SEX coded as Male, level 2 as Female and level 0 (“the missing level”) as *.

Values of variates must be in standard format (like 1.2, -0.22, +2.0E7). The symbol * is recognised as a missing value (for variates only).

Various output commands.

To list the values of a vector or a set of vectors, use the commands LIST and LIST1. LIST is for parallel listing of vectors (usually of equal lengths). LIST1 is for listing of single vectors across the page. For both commands, formats can be used to determine width and number of digits after the decimal point. With LIST it is also possible to write factor levels as names.

The command RAMSTATUS displays a list of vectors present and the space they occupy.

The command SUMMARY can be used to display summary statistics like mean, variance, max and min.

To write a comment to the output file, use the command REMARK.

Data storage

in an internal binary file format is handled by the commands `SAVEDATA` and `GETDATA`. In their simplest form, these commands are used to save and restore all vectors present in an ISUW session. The command `SHOW` can (among many other things) display the contents of a data set without importing it, with indication of potential name conflicts. ISUW data sets are files with extension `.SUD` (for **S**tat**U**nit **D**ata) — do not try to edit them or handle them with other tools than ISUW.

Graphics.

The commands `PLOT` and `HISTOGRAM` are used for graphics. `PLOT` produces scatter plots (one variate against another). Colors and plot symbols can be chosen according to the levels of factors. Points can be connected by lines as desired, and overlayed plots can be produced. `HISTOGRAM` produces histograms for variates (or factors), optionally parallel histograms grouped by the levels of a factor. Headings and axis titles are controlled by the commands `FRAMETEXT`, `XTEXT` and `YTEXT`. Without these specifications reasonable default texts (variate names etc.) are used.

Graphics can be saved in JPEG format (*.JPG) or as bitmap (*.BMP) files, for later import to text handling programs or image processing programs. Graphics of a higher resolution (but without colors) can be produced via PostScript files, see the descriptions of the commands `OPENPSFILE`, `PSFRAME` and `CLOSEPSFILE`. Interactively rotatable 3–d graphics can be produced by `PLOT` and `HISTOGRAM`, by specification of an extra variate or factor.

Restrictions.

In most applications, data are given as a rectangular data set, i.e. a number of variates and factors of the same length, which is the number of "records" or "experimental units" or "patients" or "persons" or "runs" or "plots" or whatever, depending on the applied context. We shall use the word units. To restrict attention to a subset of the units set, use the commands `EXCLUDE`, `INCLUDE`, `INCLUDEALL`, `FOCUSONLEVEL`, `EXCLUDELEVEL` and `EXCLUDEMISSING`. These commands control a hidden array of booleans (all TRUE from the beginning), telling which units are "present". All ISUW commands for which this is relevant obey restrictions, in the sense that only units present are taken into account. For example, model fit commands and `COMPUTE` commands act only on the subset of data specified as "present".

WARNING. Restrictions act in parallel on all vectors, independently of their lengths. Parallel restrictions on vectors of different lengths are usually meaningless. Be careful — use `INCLUDEALL` as soon as restrictions are no longer required. Special care should be taken in connection with SORT, SAVEDATA and TABULATE — see the command descriptions.

For convenience, the important command INCLUDEALL can be executed by pressing A from an empty command field.

Parallel sorting of vectors

is performed by the command SORT. For example, if X and Y are vectors of the same length, the command SORT X Y will sort the values of the two vectors in such a way that the values of X are increasing, and the values of Y are increasing within the tie groups determined by X. Warning: Other vectors of the same length will not be sorted (unless a special form of the command is used), and the restriction array is also unchanged. This opens some obvious possibilites for fatal errors.

Computations.

Unit–by–unit computations are performed by COMPUTE. For example, if P is a variate with values between 0 and 1, and you want to create another variate LOGIT_P of the same length holding its logit transformed values, write

```
COMPUTE LOGIT_P=LN(P)-LN(1-P)
```

If the vector LOGIT_P does not exist, it is automatically declared as a variate of the same length as P. If it does exist, it *must* be a variate of that length. Values of P that are missing or not in the range]0,1[will result in missing values of LOGIT_P. Factors can also be handled in this way, and a lot of much more complicated transformations that are not just "unit by unit" are also possible. See the description of the COMPUTE command. Once you have learned the rules, ISUW is actually a quite strong tool for data transformations. See e.g. tutorial no. 6 on the DEMOS directory.

A COMPUTE command without a left hand side simply displays the result. For example, to display the mean and standard deviation of (the values present in) the variate X, just write

```
COMPUTE mean(X)
COMPUTE sqrt(variance(X))
```

For convenience, the COMPUTE command can be generated from an empty command field by pressing the key + (without an equality sign) or = (including the equality sign). Moreover, if the line in the command field starts with a blank it is interpreted as a COMPUTE command.

Other ways of assigning values/levels to variates/factors.

GENERATE assigns systematically varying levels to a factor. For example, if G is a factor of length 20 on 4 levels, the command

```
GENERATE G 3
```

will assign levels

```
1 1 1 2 2 2 3 3 3 4 4 4 1 1 1 2 2 2 3 3
```

to the factor. The level changes cyclically, the last parameter (here 3) determining the lag between change points.

`GROUP` is used for construction of a factor by interval grouping of a variate.

`TRANSFER` can be used to copy subvector into subvector. For example, if `X` is a vector of length 100, to split it into two vectors of length 50, write

```
VARIATE X1 X2 50
TRANSFER X  1  50   X1 1 50
TRANSFER X 51 100   X2 1 50
```

`TRANSFER` can also be used to copy the values or levels present in a given vector to a new (shorter) vector.

Tables and tabular summation.

`ONEWAYTABLE` produces one–way tables of counts for factors (number of units on each level) or variates (counts of values in specified intervals). Two– or Threewaytables of counts of units or sums of a given variate over level combinations for two or three factors are produced by the commands `TWOWAYTABLE` and `THREEWAYTABLE`.

The command `TABULATE` performs counting (of units) or summation (of variate values) over the cells of a cross classification determined by an arbitrary number of factors.

Statistical models.

`FITLINEARNORMAL` is for analysis of variance and regression. Output consists mainly of an ANOVA table, holding the sums of squares corresponding to removal of terms from the model formula, beginning with the last (i.e. what SAS users call the type I sums of squares), with the corresponding F–statistics (unlike SAS with pooled variance estimates as denominators) and the corresponding P–values.

`FITLOGLINEAR` is for multiplicative or log–linear models for Poisson or multinomial data.

`FITLOGITLINEAR` is for logistic regression models for binary or binomial data.

`FITNONLINEAR` is for a class of nonlinear regression models, including the generalised linear models with overdispersion, user–specified mean (or inverse link) and variance functions.

`FITCOXMODEL` is for proportional hazards models by Cox's likelihood, optionally with right censoring, left truncation and stratification.

FITMCLOGIT, FITMCPROBIT and FITMCCLOGLOG are for ordered categorical response models as described by P. McCullagh (JRSS B 42, 109–142), where the responses are assumed to be the result of a grouping with unknown cutpoints of continuous data from a linear position parameter model with error distribution logistic, normal or Compertz.

FITCLOGIT is for conditional logistic regression (like in matched case–control studies). FITCRASCH is for a special case of this, the conditional Rasch model (two way logit–additive model for binary data by conditioning on the row sums, arbitrary linear structure for column parameters).

FITNEGBIN is for log–linear models for negative binomial data, usually coming up as "Poisson data with overdispersion".

FITANOVA is for analysis of variance, including random effects or variance component models, in balanced orthogonal designs.

After any model fit command except FITANOVA, the command LIST-PARAMETERS produces a listing of the estimated parameters in the last model fitted, and the command SAVEFITTED can be used for extraction of fitted values, residuals and normed residuals from the last model fitted (whenever this makes sense, see the command descriptions). See also the command SAVEPARAMETERS, which can be used if you want to do further calculations involving the parameter estimates and their estimated standard devitations, and the command SAVENORMEDRESIDUALS, which can be used after FITLINEARNORMAL to compute exact T–distributed ("studentised") normed residuals. TESTMODELCHANGE can be used for computation of the likelihood ratio test or F–test for model reduction when two nested models have been fitted by any model fit command except FITANOVA. ESTIMATE (which can also be used in a special version after FITANOVA) outputs the estimates of specified linear combinations of the parameters and their estimated standard deviations.

All model fit commands involve the concept of a *model formula*, i.e. a code for a linear expression involving linear effects of covariates, additive effects of factors, interactions between factors etc. This concept is carefully explained in the description of FITLINEARNORMAL. The syntax is similar to the basic syntax common to most statistics packages, but there are some important differences. SAS users should be aware that the status of an explanatory variable as a "class" variable is determined by its type (factors are always "class" variables, variates never are). GENSTAT users should be aware that the + and * operators in ISUW roughly play the same role as + and . in GENSTAT, other operators are not allowed. But here the rules are different in many respects. The rules in ISUW are such that the model matrix is generated from the model formula by the simplest possible algorithm. Users of almost any other package than ISUW should be aware that a constant term (intercept), written as 1, must be present in the model formula if it should be in the model.

WARNING. The model matrix (or design matrix) determined by a model formula is not physically stored. What is kept is a code telling how to compute its elements from values or levels of existing variates and factors. Commands referring to the last model fit use the actual values/levels of vectors occurring in the last model. If these have been changed, incorrect results will come out. If some of them have been deleted, the information that can be extracted is reduced accordingly. For example, if some of the independent variables (or an offset variable, if such was present) has been deleted, `SAVEFITTED` will not work. If the response variate has been deleted, `SAVEFITTED` will be able to produce fitted values, but not residuals and normed residuals. Similarly, if a weight variate has been deleted, only fitted values and residuals, but not normed residuals, can be produced.

Tests, non–parametrics and descriptive statistics.

The command `WILCOXON` performs a two–sample Wilcoxon or Mann–Whitney test.

The command `SPEARMAN` computes Spearmans rank correlation and performs the test for "no ordinal correlation".

The command `BARTLETT` performs Bartlett's test for variance homogeneity in a one–way setting.

The command `CORMAT` writes the matrix of correlations for a set of variates, with optional indication of significances.

Calling other programs.

Other Windows programs (or documents to be opened by applications determined by their file extensions) can be called directly from ISUW by a `SHELL` command. For example, to edit a file PRG.ISU with NotePad (if you prefer this for ISUW's `EDIT` command), use the command

```
SHELL NOTEPAD PRG.ISU
```

Programming the keyboard.

The function keys F2..F12, alone or in combination with Alt, Ctrl or Shift, and the keys A..Z and 0..9 in combination with Alt or Ctrl can (with certain exceptions that are used for other things) be programmed. For example, the command

```
KEY AO SHOW!
```

will imply that the sessions output file is displayed whenever Alt–O is pressed (the exclamation sign means "Return").

Your programmed keys are automatically saved on exit and recovered at next startup under the same ISUW root directory.

ISUW programs.

ISUW commands can be written line by line on text files by commands of the form

```
EDIT program
```

and executed by F9 from the editor or by a `RUN` command from the command field.

Programs are saved as plain text files with extension .ISU . Make sure that you include this extension in the program name if you use another editor than ISUW's own.

To avoid long lines in programs, you can split them up by the "append–next–line symbol" \. Empty lines can be inserted and lines can be indented as desired to make the program more readable. Comments can be included in two ways:

(1) Lines starting with a percentage sign % are ignored.

(2) Text in curled parentheses { } within a line is ignored.

As opposed to `REMARKS`s, such comments are not echoed to the output file.

A primitive device for parameter substitution is also available, see the description of the command `SUBSTITUTE`. The command `GOTO` can be used for conditional branching and loops.

In programs, command names (but not vector names) can be truncated as long as they are unique. The command name `COMPUTE` can be omitted for `COMPUTE` commands with a left hand side. `COMPUTE` commands without a left hand side can be written with an equality sign = or a plus + as the first character of the expression to display.

A more primitive way of executing the commands on a text file can be activated by the command `OPENCOMMANDFILE`. You can use this command to import the lines of an ISUW program line by line to the command field. The advantage of this "slow mode" execution is that you can skip commands, modify the commands imported and use other commands in between. The demonstration programs on the directory DEMOS under the ISUW root directory are executed in this way.

Installation and directory structure.

To install ISUW, download the file ISUWINST.EXE to an empty directory on your harddisk. We recommend C:\ISUW to keep file names short. Unpack this "selfunpacking" file by executing it, for example by double clicking on it from Windows Explore, or calling it by its name from the Run window or a command (MS–DOS or equivalent) window from the same directory. This operation results in the creation of four files,

```
ISUW.EXE     (the executable program)
ISUWHELP.TXT (the "on-line" help file)
BORLNDMM.DLL (Delphi system file for memory management)
DEMOS.EXE    (self-unpacking pack of DEMOS files)
```

After this ISUWINST.EXE can be deleted. The entire ISUW package occupies less than 2 MB of the harddisk.

To install a new version of ISUW, overwriting the old version, simply repeat this. Here you can avoid four confirm–overwrite–dialogs by use of the option –o, i.e. by writing ISUWINST –o rather than just ISUWINST.

In the following we refer to two directories,

1. The ISUW *root directory.* This is the place where the files SHORTKEY.TXT and SHORTKEY.BIN, holding your shortkey definitions, are kept, and the file LASTDIR keeping the name of the last sessions working directory, your latest selection of size and position on the screen of the ISUW window and your latest choice of "hint mode" on/off.

If nothing else is specified, the ISUW root directory becomes the directory where you placed the four system files. But you can (and should, if the system files are placed on a shared network drive) define the ISUW directory as any other directory by giving a valid directory name (including drive letter and colon) as the first parameter in the call of ISUW.EXE. You can have several ISUW root directories for different applications, if you wish. If you ever get the (probably rather useless) idea of having two or more ISUW sessions running at the same time, make sure you do it from different ISUW root directories, otherwise there will be file sharing conflicts. However, ISUW makes some initial file checks that will usually prevent this error.

2. The ISUW *working directory.* This directory, which (with an exception to be explained below) must be a subdirectory (or sub–sub etc.) of the ISUW root directory, is selected (and sometimes created) from a dialog box when the session begins. Typically, you will use the same directory again and again, in this case you can skip the dialog by Return. The working directory is the place where all sorts of output is placed by default, and also the place where you will typically place data files before or during the session. The working directory becomes the current Windows directory throughout the session, which means that file names without a path specification refer to files on that directory.

The working directory can be selected directly in the call of ISUW.EXE by specification of the full name (including drive letter and colon) of an existing directory as the second parameter in the call of ISUW.EXE. In this case the working directory does not have to be a subdirectory of the ISUW root directory. The entry dialog is skipped, and the file LASTDIR is left unchanged.

Here, you can also extend the name of the desired working directory to the full name of an existing file with extension either ISU or SUD on that directory. In the first case, the ISU program is executed, in the second case the ISU data set is imported. This is useful if you want to set up your Windows computer in such a way that double clicking from Windows Explore on an ISU program or an ISU data set results in a startup of ISUW with the appropriate action. Write a command file - say ISUWBAT.BAT - containing a single line of the form

C:\ISUW\ISUW.EXE C:\ISUW %1

and make this your Windows default application for opening .ISU and .SUD files (Click Tools → Folder Options → File Types from Windows Explore).

If both directories are specified as the first two parameters to ISUW.EXE, additional parameters can be added. These parameters should constitute a valid ISUW command, which will be copied to the command field. If, in addition, the last of these parameters ends with an exclamation sign, the command will be executed immediately at entry to the program. In this way you can build some standard initialization into the call of ISUW, like import of a data set, execution of an ISUW program defining some local shortkeys or whatever.

EXAMPLE. On the computer I use at work, I have placed the ISUW system files on a directory named C:\DELPHI\ISUW1 because I am developing ISUW under Borland Delphi. However, my (only) ISUW root directory is C:\ISUW. Thus, the shortcut starting ISUW from my desktop has as its "target property" the command

C:\DELPHI\ISUW1\ISUW.EXE   C:\ISUW

However, I have a main application of ISUW related to a course called MPAS. For that reason, I have another shortcut on my desktop with the command

C:\DELPHI\ISUW1\ISUW.EXE   C:\ISUW   C:\ISUW\MPAS\07

in the target field, which means that I can go directly to this application without any entry dialog. Next year I am probably going to change 07 to 08, after having created C:\ISUW\MPAS\08

In between, I add a third and fourth parameter of the form GET DATA! to load a certain data set automatically at startup.

Another way of specifying automatic initialization goes as follows. If an ISUW program named AUTOEXEC.ISU exists on a working directory, this program is executed automatically at startup from that working directory. This works quite generally — i.e. also when the working directory is selected from the dialog.

Output.

ISUW writes output to a temporary file on the ISUW root directory named ISUWOUT.TMP. This is the file you look at in a somewhat modified form by the command `SHOW`. When an ISUW session ends you will be asked if you want to save this file on the working directory under another name. If you answer "No" here, the output file is lost. In spite of some special effects created by the `SHOW` command (ISUW prompts beeing replaced with special colors of command lines, error messages and notes beeing printed in special colors, `REMARK`s in italics etc.), ISUW output files are ordinary text files which can be edited and printed e.g. by NotePad or imported to standard text handling programs.

Whenever a command sent from the command field produces written output, this is shown in a light green preview window, which you leave by Return if you want output appended to ISUWOUT.TMP, Escape if you don't. For commands in a program executed by a `RUN` command the rule is different. Here, all output is written directly to ISUWOUT.TMP, unless you redirect it explicitely to another file (or the "wastpaper basket" NUL) by an `OUTFILE` command.

Commands and error messages are echoed to the output file by default. This means that the ISUW output file will contain a complete log of what has happended during the session. In some cases (for example to avoid echoes of the same `GOTO` loop again and again) you may prefer to switch this default off by an `ECHO` command. It is recommended that you keep the output file short, because the conversion of a long file to the format that ISUW `SHOW`s it in takes some time. Do not keep `LIST`ings of large datasets, they are useless anyway. Unless you do it to create an input file for another program — but in this case it is better to create a seperate file by an `OUTFILE` command.

Uninstalling ISUW.

ISUW does not make any hidden changes to your computers setup, and does not create files under other directories than the ISUW root directory (unless you do it intentionally, for example by selection of working directories elsewhere). Thus, the uninstallation is in principle just a matter of removing the ISUW root directory, after having copied what you want to keep to other directories. The shortcuts that you may have created are easy to remove.