SalStat Statistics Package - Users Manual

Alan James Salmoni

Contents

Chapter 1

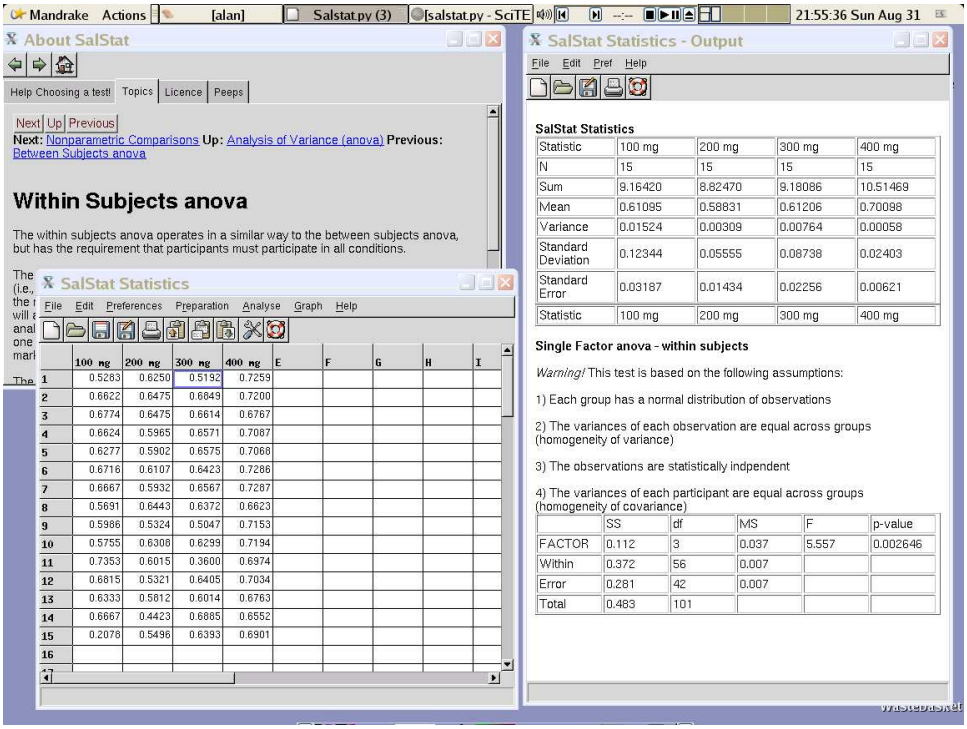
Introduction

SalStat statistics is a small application designed for the rapid analysis of scientific data. There is a special emphasis upon the sciences like psychology.

It is available under the terms of the GNU General Public License (the "GPL"), and a copy is present both with the program code, and in the program itself (use the menu, click on "help", then "about", and click on the tab marked "license". There is also a copy in the appendix of this document). In short this means that you can do what you want with the source code - change it, sell it, give it away for free, but you must also make the source code available for anyone upon request should you distribute your changes, or change the license. By the way, it costs you nothing unless you want to pay!

If you have ever used a statistics package before (like SPSS or StatView), you may be able to just start the program and be productive in a short space of time: It has been designed to be as simple as possible. No-one claims that statistics is a simple subject, but I wrote this program to be as transparent as possible to the user.

In case you are interested, it is written in the language of Python (http://www.python.org) which is very easy to learn and yet very powerful. The GUI widgets (the things you see on the screen) were created using the wxPython toolkit (http://www.wxpython.org), one of the coolest GUI toolkits around. Thanks to everyone who helped make Python and wxPython what they are (see the appendices for more information).



SalStat in action on GNU/Linux

Chapter 2

History of SalStat

The idea for SalStat began forming in my mind a number of years ago when I realised that SPSS would not run on my computer (it was a small one with a small memory by todays standards). Over time, I got a more powerful computer that could run SPSS, but I couldn't afford it, and later on, I actually taught psychology undergraduates how to use SPSS for basic statistical analysis (up to multi-factorial analysis of variance). Every time I taught, the same problems came up again and again: For example. the data entry was confusing and varied according to whether you were performing a within or between subjects test. This was confusing to people who weren't familiar with SPSS or even statistics, and I felt that it got in the way of how productive the students were. In addition, my colleagues and myself relied greatly upon the top commercial packages and found them all lacking in some way or other, which gave me an "itch" that I had to scratch.

Therefore, I saw that there was a need for a simple and easy to use statistics package available on as many different operating systems as possible which could be used by virtually anyone with a little bit of statistical knowledge. This created a few problems in itself. Obviously, in order to be easy to use, it had to have a graphic user interface (GUI) as many people find the command line interface intimidating. Besides, there are already good packages available with the command line interface (such as R or lisp-stat - see appendix C for more details). GUI programming is difficult to do across different platforms, as they all do things in different ways. This meant either that I had to write a new version of SalStat for each platform (which was difficult as I didn't have a MacIntosh to work on), or I had to use a language that offered cross-

platform capabilities with a powerful yet simple way to do a GUI interface. For example, Visual Basic would allow me to do the program on Windows easily, but it would not be available for Unix, Linux or the MacIntosh.

I considered many languages. My first choice was Ada, a language commissioned by the US Department of Defense. It was cross-platform, was compiled (which makes the program faster), and was a rock-solid language in terms of reliability (it is used in things like avionics and missile guidance systems). However, GUI programming was difficult as the only truly cross-platform interface was the Tk toolkit which didn't offer all that I wanted from a GUI. My next consideration was Object Pascal, a version of Pascal by Borland, but that ran only on Windows (Delphi) and Linux (Kylix) which meant no program for the other Unices and MacIntosh. I also briefly considered C or C++, but programmers on these languages seemed to spend most of their time debugging memory problems than anything, and I wanted a working package more than a programming challenge.

Finally I settled on the Python language. Python is a modern object oriented language that had a reputation for being easy to use. Its syntax was clear and interpreters were available for most platforms around. Not only did it offer the possibility of running programs on Unix, Linux, MacIntosh and Windows, but it also had good cross-platform GUI toolkits. The standard was called Tkinter, based upon Tk, but I decided not to use that due to the non-standard widgets (widgets are things like buttons and edit boxes). The best option left to me was a toolkit called wxPython. At the time, it wasn't available for the MacIntosh, but work was progressing on it, and I hoped that it would be ready by the time SalStat was finished. wxPython offered me all the GUI widgets that I needed (and then some!), along with native look and feel, and a simple object oriented way of programming them.

My decision to use the GNU General Public License (the "GPL") for this work was for pragmatic reasons. If had had released it under a commercial/proprietary license, I would have to offer support and do some marketing which is not something I am interested in. My objective was not to make money, but rather to make the best statistics package that I could, and the GPL offered me protection of my intellectual property in a way that no other license did. In addition, being open source, I was more likely to get people involved in contributing towards the project which would enable the code to be made more solid over time.

Since then, SalStat has come along in leaps and bounds. I have had some feedback of a number of kind people all around the world, and I find it heartening that folk from so many different countries find my work useful.

To date, I have heard from people in: UK, Ireland, US, Canada, Slovakia, France, Germany, Switzerland, the Phillipines, Australia, Brazil, and I think in Mexico too. I am interested in hearing how many people use it, so drop me an email if you can.

Chapter 3

What Does SalStat Do?

SalStat can analyse data. It is not intended to be a fully functional complete and heavyweight alternative to packages like SAS, R or S-Plus. It was designed to allow quick analysis of data for people not too familiar with computers, or those who just do not have access to the commercial packages. As such, it is not as powerful as many other packages (for more information, see the SalStat website at http://salstat.sunsite.dk for alternatives), though with future development, this may change.

3.1 Descriptive Statistics

(Parametric and non parametric are mixed here) N (count), sum, mean, variance, standard deviation, standard error, sum of squares, sum of squared deviations, coefficient of variation, minimum, maximum, range, number of missing cells, geometric mean, harmonic mean, skewness, kurtosis, median, median absolute deviation, mode, interquartile range, number of unique levels of data.

3.2 Inferential Statistics

t test (1 sample)

t test (paired)

t test (unpaired)

1 sample sign test

2 sample sign test

F test for variance ratio

Wilcoxon Ranked Sums test

Mann-Whitney U test

Kolmogorov-Smirnov test

Paired permutation test

Pearsons correlation

Spearmans rho correlation

Kendalls tau correlation

Point biserial r correlation

Linear regression

single factor analysis of variance (between subjects)

single factor analysis of variance (within subjects)

Kruskal-Wallis H

Friedman test

Chapter 4

Getting SalStat

SalStat is available for download from the world wide web. Being a Python program, it is available in source code (as Python is interpreted - don't worry if you don't know what this means), and the best place to get it is from its home site: http://salstat.sunsite.dk. This is where all the latest and greatest releases are located, as well as additional tests that can be downloaded and added. The download edition is fully functional (in other words, it is not save-disabled or limited to only a certain number of conditions), and will cost nothing. You can pay if you want (rather I should say that donations to ensure continuing development are most welcome!), but there is absolutely no obligation to do so.

There are essentially 2 versions available. The first is much bigger and contains every file you need to run SalStat. Just download this, run the installer and run SalStat! This is only available for Linux and Windows however.

Otherwise, you will have to download Python and wxPython before you can run SalStat.

If you are using Linux or a version of Unix, then you could well have Python already installed. Again, the website has the best instructions for you, so go there and head for the "download"page!

Chapter 5

Installing SalStat

Installation varies according to which platform you are using.

5.1 General

There are two ways to install SalStat onto a computer. The first, and simplest, is to get the all-inclusive file from the download page. This contains everything needed to run SalStat, but it is only available for Windows and Linux.

The other alternative is to download the much smaller SalStat code package, and also get Python and wxPython.

5.2 Linux

At the SalStat website, there is a single downloadable file which will contain all the necessary files to run SalStat. It includes a Python interpreter and all the GUI stuff needed, and will not interfere with any existing Python installation.

Most Linux users will already have Python installed. If not, they will need to get it from http://www.python.org. The next thing needed is wxPython which can be had from http://www.wxpython.org, and once those are installed, SalStat can be run.

5.3 Windows

Like Linux, there is a single downloadable which has all the files necessary to run SalStat. However, for potential developers or people who already have Python and wxPython installed, the source code alone can be dowloaded and used.

5.4 MacIntosh OSX

I shall be honest here. I don't have a Mac or access to OSX. This isn't because I don't want it (believe me, I would love a Mac!), but it does mean that I cannot develop a single file like I can for Windows and Linux. However, Ludger Humbert who does have a nice new Mac wrote to me with the following guide (apologies for any mistakes as they are my fault - like I said I don't have a Mac, and most of it is Greek to me):

First of all, install an X-server onto the Mac (this may be complicated, but it allows access to the plethora of applications developed for the X-Windowing system), and I think Orobor was mentioned (http://wrench.et.ic.ac.uk/adrian/software/oroborosx/) and Ludger recommeds installing the software with fink (http://fink.sourceforge.net). Orobor integrates well with Aqua, though I should point out that there are important differences (such as the menu appearing at the top of the window frame and not at the top of the screen).

Once all this is done, get Python and wxPython (see just above for where to get them), and once they are installed, get the SalStat source code, unpack it somewhere, and run salstat.py.

5.5 Unix

Like OSX, I have little or no experience of the Unices, but Python is available for virtually all of them, as I believe is wxPython. Get those 2 first, then get SalStat, and run salstat.py from a terminal. You may encounter problems as I think you may need to compile wxPython for your platform. If you do get it to succeed, let me know as I am interested in knowing what platforms it runs on.

Chapter 6

Using SalStat - entering data

When SalStat is running, you will see 2 different windows on your screen. One has a grid like pattern on it, and the other is mostly blank. Both have menus.

The window with the grid on is where you can enter your data. It works like a spreadsheet (although there are no functions available currently). Each cell contains a single datum, and you can actually input anything you want - either a number or a string of letters. The grid will be called the data grid from now on. The window with the blank sheet is called the output sheet - once you have done some data analysis, this is where your results will appear.

Most popular statistics packages demand that (for between subjects designs at least) the data are entered in a particular way. SalStat has been designed to be intuitive, and differs in that each condition is entered into its own column. In other words, if you have an experiment with one variable that has 3 levels (say, amount of alcohol - 2 units, 4 units or 6 units), then you will need to enter your data into three columns: one column for each condition.

The data grid can be customised to your liking. If you find the cells too large or small they can easily be changed: if you go to the menu, click on "Preferences" -> 'Change Cell Size...".

A small dialog box will come up that has 2 buttons on the bottom (marked "okay" and "cancel"), and a couple of what are called "spin controls". Next to the box with a number in it are 2 small arrows pointing up and down. Click on these to change the values. The top box number refers to how wide

the columns are, and the lower box is how high the rows are. Try changing the values of both, then click on "okay", and you will see that the data grid has changed all of the cells sizes. If you only want to change the size of one column or row, put the mouse point over the line next to the column or row you want to change. You will see that the mouse cursor changes from the normal arrow into a line with an arrow at each end. This means you can press the left button and drag the line (with the button held down) to where you want.

As each column in SalStat refers to a different condition, each column heading can also be changed: These may be referred to as variable names. Of course, you can keep the default headings of A, B, C and so on, but it may be easier when you read your results to have headings more meaningful to yourself. Using the above example of 3 levels of alcohol, you could use headings of "2 units", "4 units", and "6 units". To change the headings, just go to the menu, click on "Preferences" -> "Variables...".

A new window will appear, and in this there is another grid, but this time smaller. Here, you can put in whatever variable names you want into the top row. If you have missing data (data where you were not able to collect a value), you can also enter what you want your missing data to be. By default, the value is -99.999, but if you have some real data of this value, you would want to change the missing value to something more suitable.

If you have some data saved on disk, it can be loaded. Currently, SalStat only loads ASCII data - this means simply plain text, and with no variable names. This is done by going to the menu and clicking on "menu -> open...", and a standard dialog will appear allowing you to select the file you want to load. If your data are too big for the grid, don't worry because SalStat will automatically adjust its size to fit the data.

Saving data is done the same way. From the menu, click on "File -> Save As...", and you will get a dialog where you can enter the file name.

If you need to add an extra column or row to the grid, go to the menu, click on "edit" -> "Add Columns and Rows...", and a dialog will appear where you can enter the number of rows or columns you want to add.

From the same place, you can also delete rows or columns. The column or row deleted will be the one with the "focus" (this is the cell where you are currently entering data).

Chapter 7

Using SalStat - dealing with the output

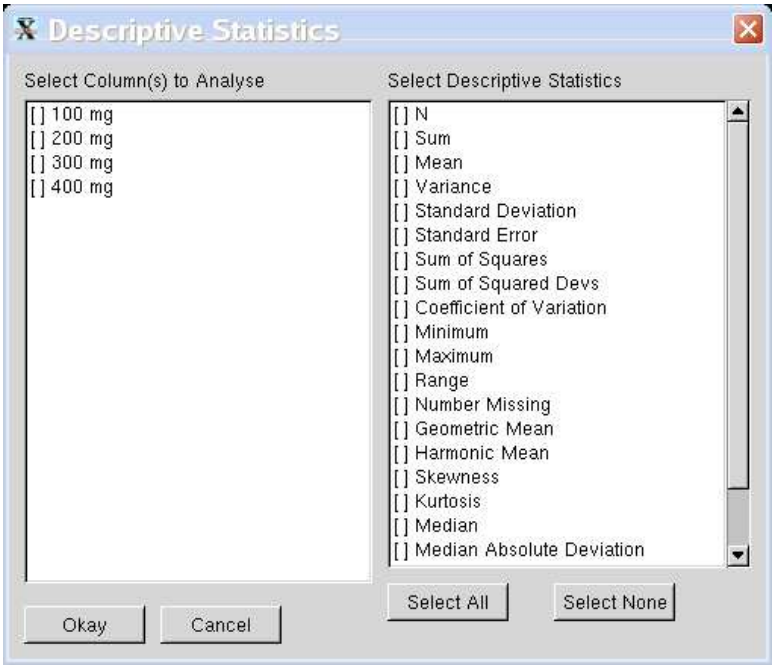
The output is in html format - this is the format used for web pages, and your results can be opened in any web browser and most word processors to be edited. It is displayed in the output sheet - the window with the large space saying "SalStat Statistics".

When you have done some analysis, the results will all appear there. If you wish to save the results, click on the menu "File" -> "Save As...", and a dialog will appear allowing you to enter a filename.

Chapter 8

Descriptive Statistics

The first stage of analysis once the data have been entered is usually to get some descriptive statistics. In SalStat, this is a very easy process. Once the data are entered into the grid (remember, each condition goes into its own column), go to the menu bar, click on "Prepare" and select "Descriptive Statistics...".



Descriptive Statistics

A dialog box will appear. On each side, there are two large boxes, and underneath these, there are four buttons (marked "Okay", "Cancel", "Select All", and "Select None" from left to right). The big box on the left holds a list of columns. However, it only shows columns that contain data: This is because there is no point in getting descriptive statistics for an empty data set (I can tell you that the N, sum, mean, sample variance and so on will be zero!). If you want descriptive statistics for a column, just click on the small box next to it to select it. Next, you will need to select which descriptive statistics you want. This is where the big box on the right hand side comes in, for it contains a list of statistics that you may want. Of course, it is a long list, and may grow as SalStat develops even further, so to save you time, there are two buttons underneath one of which you can click on to select the lot, the other of which you can click on to deselect them all.

Once all this is done, just click on the "Okay" button, and the descriptive statistics will appear in the output window, all in a nice table ready for you to read.

Chapter 9

Transforming Data

If you find that your data are skewed or kurtotic, you may want to transform the data. There may indeed be other reasons for transforming data too. This is quite a simple process in SalStat. Just go to the top menu, click on "Prepare" and then the "Transform..." option. A dialog box will appear just like below.

On the left hand side of the dialog box is a list of grid columns, just like you would see if you wanted some descriptive statistics. This list only contains grid columns that actually contain data (because there's no point in transforming nothing!). Just click on the small box to the left of each column name if you want to change it.

To the right hand side, there are four buttons, and beneath them a nice little box where you can type text in. The text will show the function that you wish to apply to your data, and you can type in your own functions.

However, there are a set of functions that are commonly used in transforming data. SalStat has four of these built in, and clicking on one of the four buttons will show the function in the text box.

The functions are: square root, square, logarithm, and reciprocal. When SalStat is transforming data, it takes the data from the columns that you indicated, applies the function to each cell in the column and puts the new data into the first empty cell it can find.

You will also find that the column heading (the variable name) also changes: if you used one of the prepared function buttons, the name of the function

will appear along with the column originally used. If you typed in your own function, then that function will be used for the variable name.

9.1 Python functions.

As mentioned earlier, you can type in your own functions. Because SalStat is a Python application, the syntax is basic Python. The functions use the term 'x' to denote the cell.

9.1.1 Examples:

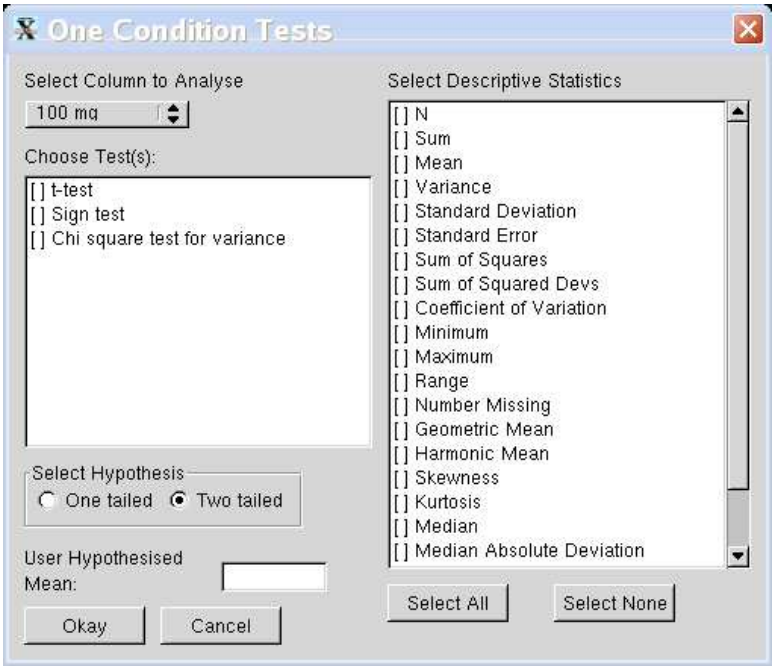
If you wanted to add 2 to each column, the function would be 'x + 2' or '2 + x'. If you wanted to get the cube root, you would type 'x * x * x' or 'x ** 3'. There are a range of functions available if you use the 'math' module. Just remember to type 'math.' before the function, and you can type in things like: 'math.atan(x)' to get the arc tangent, 'math.cos(x)' to get the cosine, 'math.exp(x)' to get the exponential and so on. Consult the Python documentation for further details (http://www.python.org).

Chapter 10

Analysing 1 Sample Tests

SalStat is very easy to use to analyse a single sample test (one sample tests are when you only have a single data set that you wish to compare against a mean). Go to the menu, click on "analyse", then "1 Sample Tests...", and a dialog box will appear.

At the top left hand side of this box, there is a drop down box. Clicking the arrow on the right hand side of the box will display a list of non-empty columns. Of course, if you have entered no data, then no columns will appear, so get rid of the box and enter some data! (but you knew that already, didn't you?)



One Sample Tests

Underneath this is a check list box like the 2 used when you got the descriptive statistics. To select a test, you will need to click on the small box to the left of the test, and it will be used to analyse your data.

If you only want the test result, there are 2 more things to do. The first is to select the tail of the hypothesis (more information in the chapter on analysing 2 sample tests). The 2 tailed hypothesis is kindly selected for you by default, and this seems to work for most analyses, although you can select a one tailed hypothesis if you are lucky enough to have evidence to support a claim for a directional hypothesis.

One sample tests differ from most others in that you are only looking at one data set. This leads to a problem of what are they being compared against, and the way to do this with SalStat is to enter what is termed a "User Hypothesised Mean" in the little text box underneath where you indicate the tail of your hypothesis. What this value is depends upon what you think the population mean is, and it is really up to you to decide what to use.

Once this is done, you can click the "okay" button, and the results will dutifully appear in the output window ready for your inspection! Simple, eh?

Of course, many people want to find out more information than just a test statistics (and p-value). On the right hand side of the window, there is a list of descriptive statistics (just like the one when you were getting descriptive statistics), but with both parametric and non-parametric tests present. Again, you can select which tests you want, or you can be completely greedy and get the lot by clicking on the button marked "Select All" at the bottom of the screen. When you are ready, click on "okay".

10.1 Parametric Comparisons

10.1.1 One Sample t-test

The one sample t-test is a parametric test on one sample of data which calculates the mean of the sample and compares it to a hypothesised mean. It is like a 2 sample t-test, but the user must provide the mean to which the data are compared. This comparison is to check whether the means differ significantly, and this is shown in the p value.

To perform a one sample t-test, enter the data into a single column on the data grid. Go to the menu bar and click on "Analyse" -> "One Sample Tests...". A dialog box will appear. Using the drop-down box at teh top of the dialog box, select the column which contains the data. Check "t-test", and enter the hypothesised mean into the box marked "User Hypothesised Mean", select a one or two tailed hypothesis (two tailed is selected by default), and click the "Okay" button.

Output consists of the user hypothesised mean, the degrees of freedom (in brackets), the t statistic, and the approximate p value.

10.2 Nonparametric Comparisons

10.2.1 One Sample Sign Test

The one sample sign test is a non parametric test which does much the same thing as the one sample t-test but with non parametric data, therefore comparing the median against the hypothesised value. It is not very robust, and

should be used with caution. Again, the user must provide a hypothesised mean against which the data will be compared like the one sample t-test.

Enter the data into a single column on the data grid. Go to the menu bar and click on "Analyse" -> "One Sample Tests...". A dialog box will appear. Using the drop-down box at teh top of the dialog box, select the column which contains the data. Check “sign test”, enter the hypothesised mean into the box marked “User Hypothesised Mean”, select a one or two tailed hypothesis (two tailed is selected by default), and click the “Okay” button.

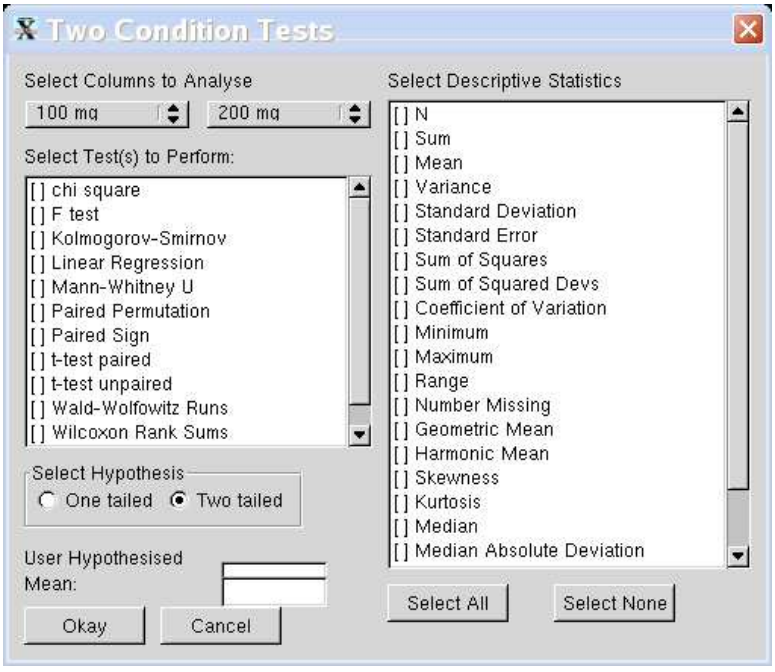
Output consists of the user hypothesised mean, the value for N, the value for z and the p value.

10.2.2 Chi Square Test for Variance

Chapter 11

Analysing 2 Sample Tests

Analysing tests that have 2 samples is almost as easy as doing the one sample tests - maybe even easier! Go to the menu, click on "Analyse" and then "2 sample tests...", and the following dialog box will appear.



Two Sample Tests

This dialog has 2 drop down boxes at the top. These are what you use to specify which columns of data are to be used in the analysis. Just like the one

conditions tests dialog, on the right hand side is a list of descriptive statistics that you can select if you wish. The major difference for this dialog though is that there are a lot more tests available, but they are all grouped together in alphabetical order. Of course, if you want to, SalStat will not stop you from performing both parametric and nonparametric tests on your data if you so wish, and you may perform them at the same time!

Just select the tests you want, and check the right tail of hypothesis for the experiment and click okay. The results will appear on the output window.

NOTE - the F-test for variance ratios need to have a user hypothesised variance entered, and this is done in the little text box just above the "Cancel" button.

11.1 Parametric Comparisons

11.1.1 Unpaired t-test

The unpaired t-test is a parametric test that analysis 2 samples of data, and examines whether the means of each sample differ by chance or due to experimental manipulation. The unpaired test differs from the paired test in that participants must only be subjected to one condition and not both. If participants are subjected to both conditions, use the paired t-test.

Enter the data for each condition into its own column (in other words, one column for one condition). Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check "t-test unpaired" from the list in the box marked "Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default), and click on the "Okay" button.

Output consists of the degree of freedom (in brackets), the t statistic and the p-value which gives the likelihood that the 2 samples differ by chance.

11.1.2 Paired t-test

The paired t-test is a parametric test which compares 2 samples of data and tests whether they differ significantly. It differs from the unpaired t-test in

that the paired t-test insists that participants take part in both conditions. If participants took part in only one condition, use the unpaired t-test.

Enter the data for each condition into its own column (in other words, one column for one condition). Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check "t-test paired" from the list in the box marked "Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default), and click on the "Okay" button.

Output consists of the degree of freedom (in brackets), the t statistic and the p-value which gives the likelihood that the 2 samples differ by chance.

11.1.3 F test for Variance Ratio

The F-test is a parametric test for 2 samples of data and it compares the variances of each sample to see if they differ. It can be used to check whether the assumptions of the t-test have been violated or not, and shows whether the variances of the 2 samples are the same or whether one group is larger or smaller than the other.

Enter the data for each condition into its own column (in other words, one column for one condition). Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check "F-test" from the list in the box marked "Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default), enter the user hypothesised ratio into the box marked "User Hypothesised Ratio" and click on the "Okay" button.

Output consists of the 2 degrees of freedom (in brackets), a value for F and the p-value which gives the likelihood that the 2 samples' variances differ by chance.

11.2 Nonparametric Comparisons

11.2.1 Paired Sign Test

The paired sign test is a non parametric test that compares 2 samples to see if they differ due to chance or not. The samples must be within subjects (participants did both conditions), and is useful because it makes few assumptions about the data, but is not a very powerful test. The Wilcoxon signed rank test may be used if greater power is needed.

Enter the data into 2 columns, each sample having its own column. Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check “Paired sign” from the list in the box marked “Non Parametric Tests”, select a one or two tailed hypothesis (two tailed is selected by default) and click on the “Okay” button.

Output consists of a value for N, a value for z, and the p value.

11.2.2 Wilcoxon Signed Ranks Test

The Wilcoxon signed ranks test is a non parametric test for within subjects designs (participants must take part in both conditions) that tests 2 samples of data. It analyses the same type of data as the paired sign test, but is more powerful This test first ranks the data, and takes the difference between the ranks of each pair.

Enter the data into 2 columns, each sample having its own column. Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check “Wilcoxon Signed Ranks” from the list in the box marked “Non Parametric Tests”, select a one or two tailed hypothesis (two tailed is selected by default) and click on the “Okay” button.

Output consists of a value for z, a value for t and a p value.

11.2.3 Wilcoxon Rank-Sums Test

This test is a non parametric test designed to analyse 2 samples of data that are unrelated (in other words, participants took part in only one condition). For this test, the observations of both conditions are combined and then ranked, and then each condition is summed. If they are equal, either sum may be used, else the smaller of the two is used.

Enter the data into 2 columns, each sample having its own column. Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check "Wilcoxon Ranks Sum" from the list in the box marked "Non Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default) and click on the “Okay” button.

Output consists of a value for t and a p value.

11.2.4 Mann-Whitney U Test

The Mann-Whitney U test is a non parametric test to analyse 2 samples of data that are unrelated (participants therefore only took part in one condition). Both samples must be continuous, and compares the ranks rather than the observations like the t-test does. Because ranks are used, it is resistant to outliers.

Enter the data into 2 columns, each sample having its own column. Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check "Mann-Whitney U" from the list in the box marked "Non Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default) and click on the “Okay” button.

Output consists of a value for z, a small value for U, a big value for U and a p value.

11.2.5 Kolmogorov-Smirnov Test

The Kolmogorov Smirnov test is a non parametric test that analyses 2 unrelated samples (participants only did one condition), and the data must be

continuous. Because of the way it compares the distributions, it may be influenced by outliers and should be used cautiously if they are suspected.

Enter the data into 2 columns, each sample having its own column. Go to the menu, click on "Analyse" then "2 Condition Tests..." and a new window will appear. Select the 2 columns which contain the data using the 2 drop down boxes at the top of the window. Check "Kolmogorov-Smirnov" from the list in the box marked "Non Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default) and click on the "Okay" button.

Output consists of a value for D, and a p value.

11.3 Correlations (parametric and nonparametric)

Correlations are used to see how well 2 sets of data are correlated with each other. Correlations vary from -1.0 to +1.0. The closer the correlation is to 0.0, the less the correlation is. Correlations closer to -1.0 or +1.0 show a high correlation - in other words, as one varies, the other varies with it. Note that correlations are only for observational data, and a high correlation does not imply a causal link, rather an association of some kind between the samples. Positive correlations reflect a situation whereby both increase and decrease with each other, whereas negative values show that as one sample increases, the other decreases and vice versa.

11.3.1 Pearsons Correlation Coefficient

Pearsons correlation is a parametric correlation which uses the distributions of the data, and therefore the data must be continuous. Obviously, it uses 2 samples of data, but because it uses parametric data, it should be used with caution when outliers are suspected.

Data are entered in one column for each sample. Go to the menu, click on “Analyse” and then “2 Condition Tests...” and a new window will appear. Using the 2 drop down boxes at the top of the window, select which columns contain the data, check “Pearsons Correlation” from the box marked “Parametric Tests”, select a one or two tailed hypothesis (two tailed is selected by default), and click on the “Okay” button.

The output consists of the degrees of freedom (in brackets), a value for r (which shows the correlation direction and strength), and a p value.

11.3.2 Spearmans Rank Correlation

Spearmans rho is a correlation for non parametric data which unlike Pearsons correlation uses the rank ordering of the data, and is therefore not susceptible to outliers in the data.

Data are entered in one column for each sample. Go to the menu, click on "Analyse" and then "2 Condition Tests..." and a new window will appear. Using the 2 drop down boxes at the top of the window, select which columns contain the data, check "Spearmans Correlation" from the box marked "Non Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default), and click on the "Okay" button.

The output consists of the degrees of freedom (in brackets), a value for rho (which shows the correlation direction and strength), and a p value.

11.3.3 Kendalls Coefficient of Rank

Kendalls tau is another non parametric correlation, is very similar to the Spearmans rho correlation and is resistant to outliers.

Data are entered in one column for each sample. Go to the menu, click on "Analyse" and then "2 Condition Tests..." and a new window will appear. Using the 2 drop down boxes at the top of the window, select which columns contain the data, check "Kendalls tau Correlation" from the box marked "Non Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default), and click on the "Okay" button.

The output consists of a value for tau (which shows the correlation direction and strength), and a p value.

11.3.4 Point Biserial r Correlation

The point biserial r correlation is for non parametric data, but the first sample must contain dichotomous data (which means that there should only be 2 categories like male and female).

Data are entered in one column for each sample, one of which must contain dichotmous data. Go to the menu, click on "Analyse" and then "2 Condition Tests..." and a new window will appear. Using the 2 drop down boxes at the top of the window, select which columns contain the data, with the dichotmous data being the first column selected. Check "Point Biserial r" from the box marked "Non Parametric Tests", select a one or two tailed hypothesis (two tailed is selected by default), and click on the "Okay" button.

The output consists of the degrees of freedom (in barckets), a value for r (which gives the correlation direction and strength), and the p value.

11.3.5 Linear Regression

Linear regressions are useful when you have 2 variables (one independent and one dependent) and wish to examine the relationship between them more closely. Linear regressions can be used to predict one variable from another, and it uses 2 samples of data.

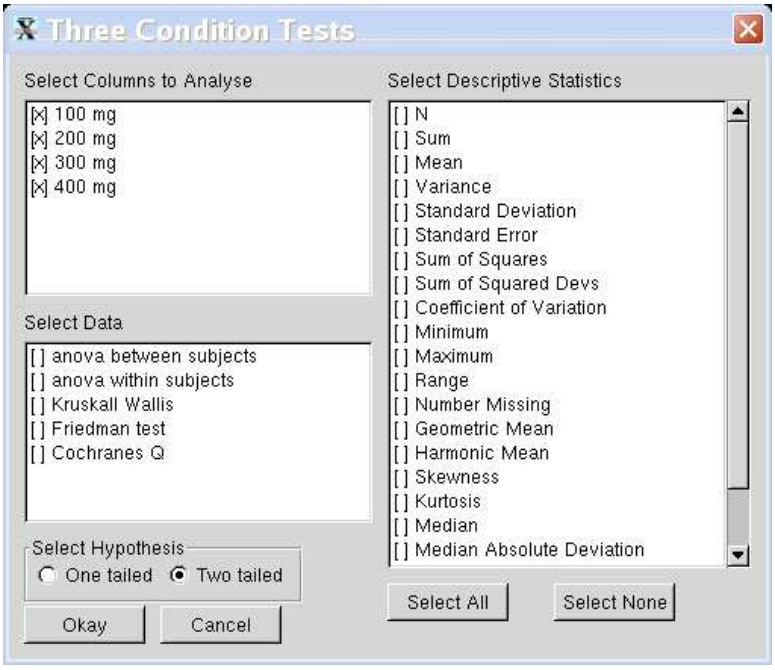
Data are entered with one column on the data grid for each sample. Go to the menu, click on “Analyse”, then “2 Condition Tests...” and a new window will appear. Select the columns of data to be analysed using the 2 drop down boxes at the top of the window. The independent variable must be selected using the first (left hand) box, and the dependent variable using the second (right hand) box. Select “Linear Regression” from the box marked “Non Parametric Tests”, choose a one or two tailed hypothesis (two tailed is selected by default), and click on the “Okay” button.

The output consists of the slope, the intercept, a value for r, the p value, and the estimated standard error.

Chapter 12

Analysing 3 Sample Tests

3 or more sample tests are used when you have 3 or more samples - quite simple really! For example, we could expand the above experiment with alcohol and reaction time by having more than 2 different levels of alcohol - let's say the first level would be no alcohol (zero units), the second would be 2 units and the third would be 4 units. This means that there would be three conditions in the experiment.



Three or more Sample Tests

To analyse this kind of experimental design, you might consider using a t test, firstly to compare 0 units against 2 units, then 2 units against 4 units, then again to compare 0 units to 4 units. That way you would be comparing all the conditions against each other.

However, in reality you would not be able to do this because you would be analysing the same set of data three times. Because statistics is based upon, well, statistics, the chance of your getting a significant result increases each time you analyse a set of data, which increases the likelihood of there being a Type I error - when a significant result is found (and the null hypothesis rejected) when there wasn't really a significant result in the first place.

When you have more than 3 conditions, you need to use other tests, and probably the most famous (and in the authors experience, the most used) test is the "analysis of variance" (or anova for short). This test will allow you to compare a large number of conditions with each other, and tell you whether any of them differ significantly.

The problem with anova is that they don't actually tell you where the significant difference is, they only tell you whether one exists, and there are ways to find out where the difference lies but that will be explained later in the section.

The anova is for parametric data. If you have non parametric data, there are equivalents that you can use with SalStat: The Kruskal-Wallis H test, and the Friedman Chi Square, both of which act like a non-parametric anova, but SalStat deals with them in the same way. The Kruskal-Wallis H test is for between subjects designs and the Friedman test is for within subjects designs.

Once you have entered your data, click on the "analyse" menu, then select "3+ Sample Tests..." for the appropriate dialog to appear. This dialog is a bit different from the other 2 test dialogs that you have encountered (assuming that you have followed this book in a linear order of course), but the main difference is that you will not select which columns you want to analyse by using drop down boxes - this is because there may be more than 2 columns you want to analyse. Instead, you have a check list to select your columns, rather like the dialog box that appeared when you wanted to get a set of descriptive statistics. Only columns with data will appear there, and to

select one, all you have to do is click on the box to the columns left and it will be selected (click again to de-select it).

Like the other test dialogs, there is a list of descriptive statistics to the right hand side and you can select which ever ones you want (if you want any of course which is usually a good idea). The next thing to do is select which test you want - again using a check list box (perhaps SalStat should be called "CheckListStat" instead?), and finally select which hypothesis you want.

SalStat differs from other statistics packages by allowing you to choose mutually exclusive tests: You can choose an anova for within or between subjects, and get the non parametric equivalents at the same time. This might not be orthodox operation for statistics packages, but let's face it - if you wanted to analyse the same set of data using different tests, you could do so anyway, but it would just take longer. SalStat allows you to cheat in a more efficient manner, and leaves the decisions up to you.

Okay, so you have done all you need to do: Selected the columns you want to analyse (necessary), chosen the test (not necessary, but if you don't, there will be no results to look at), and chosen the hypothesis (one or two tailed). Optionally, you have a list of descriptive statistics to choose from, but that's your choice. The next thing to do is click on the "okay" button.

Wait a short while (depending upon how large your set of data is), and the results should appear in the output window. The results for an anova differ slightly from the other tests in that they show a lot more information. You should be shown a table where the first column lists the factor you are analysing, the error, and the total. The next column shows the Sum of Squares (SS) for each factor, then the degrees of freedom (df) followed by the Mean Square (MS). These are of interest, but the main thing for you to look at now are the last 2 columns. The first of these is titled "F", and this is your test statistics. If it is less than 1.0, then it will not be significant no matter what your data look like. The final column lists the p-value, and you should be aiming for a p of less than 0.05 (or 5%) which means you have a significant result.

The Friedman and Kruskal-Wallis H tests do not have as much detail as the anova which shows a table with the degrees of freedom, sums of squares and so on.

12.1 Parametric Comparisons

12.1.1 Analysis of Variance (anova)

The analysis of variance (or anova) is a parametric test that compares samples of data much like a t-test but can simultaneously compare 2 or more conditions. For example, if you have 3 conditions, you could use a series of t-tests to analyse the data, but this repeated analysis would increase the likelihood of committing a Type I error (when the null hypothesis rejected but should have been accepted). The anova gets around this neatly, and can analyse between subjects or within subjects designs. Obviously, being a parametric tests, it is susceptible to outliers and the data should be checked beforehand to ensure that they have been dealt with adaquately. A significant results indicates only that a significant difference exists somewhere. It doesn't say exactly where and further testing is required to ascertain this.

12.1.1.1 Between Subjects anova

The between subjects anova is for experimental designs in which each participant only does one condition. If the participants do more than one condition, then use the within subjects anova (below).

Each condition must be entered into a separate column on the data grid. When that is done, go to the menu and click on "Analyse", and then "3+ Condition Tests...", and a new window will appear. From the list in the top left hand window, select which columns you wish to analyse, and then select "anova between subjects" from the list of tests below. Choose a one or two tailed hypothesis (two tailed is selected by default), and click on the button marked "Okay".

The anova differ from other tests in that a table of information is provided. The first column contains the sources of the information - in this case, the factor, the error and the total. The next column in the table contains the sum of squares for the sources, then the degrees of freedom followed the mean square for the factor and the error term are provided (there is no mean square for the total). The penultimate column contains the f-ratio (the test statistics) just for the factor, and the final column contains the p value which reports the probability that the f-ratio is significant.

Further analysis using post-hoc tests are not yet available in SalStat.

12.1.1.2 Within Subjects anova

The within subjects anova operates in a similar way to the between subjects anova, but has the requirement that participants must participate in all conditions.

The data are entered as one column for each condition, and they must be matched (i.e., each row must contain the data for a single participant). When that is done, go to the menu and click on "Analyse", and then "3+ Condition Tests...", and a new window will appear. From the list in the top left hand window, select which columns you wish to analyse, and then select "anova within subjects" from the list of tests below. Choose a one or two tailed hypothesis (two tailed is selected by default), and click on the button marked "Okay".

The output of a within subjects anova is a little bit more complicated that the between subjects anova. Again, the data are in a table, but there is an extra factor, the "within subjects variance" which shows how much variance each participant had between the conditions.

As for the table itself, the first column contains the source, the second the degrees of freedom, the third the mean square, the fourth the f-ratio, and the fifth column contains the p value.

Not yet available in SalStat.

12.2 Nonparametric Comparisons

The non parametric comparisons may be used when the data are parametric. As they rely upon ranking, they are not as susceptible to outliers as the anova are. Like anova, they can compare data from 2 or more conditions without increasing the chance of a Type I error.

12.2.1 Kruskal-Wallis H

The Kruskal-Wallis H test is a non parametric between subjects test - participants must have took part in one condition only.

Enter the data like the between subjects anova - each column must contain the data for each condition. Go to the menu and click on "Analyse", and then "3+ Condition Tests...", and a new window will appear. From the list in the top left hand window, select which columns you wish to analyse, and then select "Kruskal-Wallis" from the list of tests below. Choose a one or two tailed hypothesis (two tailed is selected by default), and click on the button marked "Okay".

This is a lot simpler than the anova. The output simply consists of the degrees of freedom (in brackets), a value for H (the test statistic), and the p value.

12.2.2 Friedmans Test

Freidmans test is like a non parametric anova for within subjects designs. It can compare more than 2 conditions and is robust against outliers. Participants must have taken part in all conditions, and must be matched on the data grid (each participants data is entered on a single row).

Enter the data like the between subjects anova - each column must contain the data for each condition. Go to the menu and click on "Analyse", and then "3+ Condition Tests...", and a new window will appear. From the list in the top left hand window, select which columns you wish to analyse, and then select "Friedman test" from the list of tests below. Choose a one or two tailed hypothesis (two tailed is selected by default), and click on the button marked "Okay".

The output of the Friedman test, like the Kruskal_Wallis test, is a lot simpler than the anova. Reported statistics are the degrees of freedom (in brackets), the value for chi (the test statistic) and the p value.

12.2.3 Cochranes Q Test

The Cochranes Q test is a nonparametric test designed to examine whether there are differences between three or more matched samples, much like a nonparametric repeated-measures analysis of variance. The Cochranes Q test differs from the Friedmans test because it deals only with dichotomous data (there can only be 2 categories, such as male / female, or old / young). If

you try to analyse data in more than 2 categories, SalStat will give results that cannot be relied upon.

To perform this test, enter the data with each column containing the data from one condition. Then go to the menu and select "3+ Condition Tests...", and the test dialog window will appear. From this, select "Cochranes Q" from the test selection box and choose an appropriate hypothesis. Click the "Okay" button.

The output returns a value for Q (the test statistic), the degrees of freedom (df) and the probability value.

Chapter 13

Scripting SalStat and Making Your Own Tests

One of the exciting developments of SalStat has been the ability for users to write their own scripts. These scripts can access the internals of SalStat quite easily, and allow the user to automate almost anything that they want, from loading a file, getting a set of descriptives, writing a new test using these descriptives as well as performing an anova, then putting the results into a webpage.

Sadly, there is support for only one language - Python, which is the language in which SalStat is written. The good news is that Python is easy to learn, has lots of online free documentation available, and has a remarkably clear syntax.

In case you think Python is not suitable, remember that it is used by NASA (and SalStat!).

In this chapter, there will be an introduction to scripting. Sadly, this will assume a basic knowledge of Python (knowing what variables are, what functions are and so on), but if you know another computer language, it will be easy to pick up. In the appendices, there is a full reference of commands especially for SalStat. If you are using the single executable, then the range of commands will be relatively limited. If, however, you have Python installed (and it's usually worth it!), you will have an enormous amount of tools available, with everything from modules that can

control web browsers, send emails, query databases, draw complex graphics, make sounds and much more. See the Python website for more details: http://www.python.org/documentation.html

13.1 Starting Scripting

To get started on scripting, you will need to open the scripting window. This is done by clicking on the "Analyse" menu, and selecting "Open Script Window...". A new window frame will appear, with a toolbar at the top, and most of the window will be taken up by an edit box. This edit box will accept typing (try it...), cut, copy, paste (the toolbar allows access to these functions, but you can use <ctrl-x> to cut, <ctrol-c> to copy and <ctrl-v> to paste text. There are also buttons for saving and loading scripts to file.

The one new button is the first on the toolbar. This is the "run script" button, and pressing it will execute the script using the built-in Python interpreter.

13.1.1 Manipulating Data

To start with, type in

```
display (“This is SalStat Scripting”)
```

- only type in what's between the diagonal brackets, not the brackets themselves (and don't forget the quotation marks!). Once that is done, click on the "Go" button (marked with an arrow), and look at the output window. You should see "This is SalStat Scripting" written somwhere.

What is happening is this. Writing `display` tells the computer that you want to display something on the html window. What you want on there is in the open brackets, but because you want to print a string, the words will need to be in quotation marks.

Delete everything you've typed and type in:

```
a = 'This is a Python script'

display(a)
```

Then press the Go button. Hopefully, the words "This is a Python script" will be printed on the html window. What you have done is made a variable

(called 'a'), and assigned to it the words "This is a Python script". The next line says that you want to display what is in the variable 'a'.

Of course, all of this doesn't mean very much. Other functions can be called too. If you have some data on the grid and you want to put it into a variable, the *GetData* function is used (PLEASE NOTE: Python is case-sensitive. `GetData` is different from `getdata` or `Getdata` - make sure you use the capitals in the right place). Just use *variable* = GetData(*column*), and the variable will be given a list of the data from the column specified. Example:

```
a = GetData(0)
```

This generates the variable called *a*, and asks that the contents of column 0 be copied into this variable from the grid (ANOTHER NOTE: in SalStat, the grid columns start at 0, not 1). Type some data into the grid and try it out. Of course, to see what *a* contains, write:

```
display(a)
```

afterwards, and you will see the contents of the first column there. Data can also be put back into the grid with the PutData command, but you have to specify the column where the data are going to inserted, and the data itself. Try:

```
PutData(1, a)
```

and the data from the first column that you just got will be copied to the second column (column number 1).

Of course, these commands can be mixed if you want, so typing:

```
PutData(4, GetData(0))
```

is valid - it makes a copy of column 0's data, and puts it into column 4.

13.1.2 Descriptive Statistics

Now how's about doing something statistical with all of this? Simple. To start with, let's get the descriptive statistics for column 0. This is slightly more complex than the previous commands, but is still quite straightforward. Type in:

```
x = Describe(a)
```

What this does is retrieve a complete set of descriptive statistics for you, and puts them into the variable *x*. However, if you typed in *display(x)* underneath, you probably didn't get what you expected! This is because you need to specify which descriptives you want before you can access them. This is just as easy: to access them, type the variable you had (the *x* in the current example, then type a full stop (period) after it, then the descriptive name. So to show the sum, the mean and the variance, you would type:

```
display(x.sum)
```

```
display(x.mean)
```

```
display(x.variance)
```

A full list of the descriptive statistics available is in the appendices.

Obviously, there are mathematical operators available, so you can type in:

```
y = x.mean + x.variance
```

13.1.3 Analysis

To get the results of a test using scripts is a little bit more complicated. Let's look at how to do a paired t-test:

```
t, df, p = DoPairedTTest(col1, col2, 2)
```

First of all, the *DoPairedTTest* bit just tells SalStat what test you want. The first 2 items in the brackets are the data. These can be lists of data, or, if they are single whole numbers, the data from the relevant column will be retrieved. Of course, you can pass one list and one column number if you want to! The third number is the tail. 2 refers to a 2 tailed test, whereas 1 refers to... well, a 1 tailed test. No other numbers are valid for that bit.

However, what does the *t, df, p* bit mean? Well, doing a t-test will result in 3 bits of data: the t statistic, the df or degrees of freedom, and the probability. When you call a `DoPairedTTest` function, three data items are returned in the order of: t, df, probability. To make it simple, the above example puts the results into very similarly known variables, but you can use what you want. For example, if you really wanted to, you could type:

```
df, prob, t = PairedTTest(x,y)
```

This could be confusing, because the test will always return t first, then the df, then the probability, so your t statistic will be in a variable named *df*, the df will be in a variable named *prob*, and the probability will be in a variable called *t*. All very confusing, so remember to use variable names that aren't coonfusing to you.

(Notice that no tail was indicated just above, only variable names: This is because if you don't specify a tail, the test defaults to a 2 tailed test).

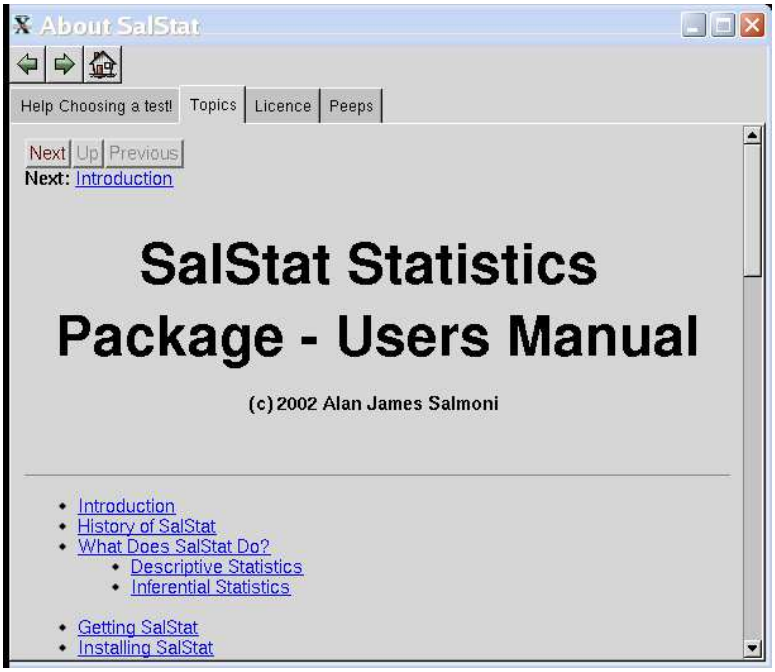
Scripting is a very powerful addition to SalStat. If you have the algorithms for a test that isn't in SalStat, you can write your own version of it quite easily. Of course, you could do this with Python on its own anyway, but with SalStat, you get the benefit of access to a good structure of descriptive statistics that are easily accessible, and a GUI to look at your data. Alternatively, you can get tests off other people and use them if you want, or extend existing tests. Scripting is also useful for batch processing: If you have a lot of data sets that have to be analysed in exactly the same way, you can use scripting to automate the entire process.

If you wish, you can have access to the "raw" functions and classes in the statistics module using Python. To do this, you will need to put a copy of the *salstat_stats.py* module in the site-packages directory, and call import salstat_stats in your program, and all the classes and functions contained therein will be available.

Chapter 14

Getting Help

SalStat has an integrated help system. If you go to the menu, click on "Help" and choose one of the options below, a new window will appear. There are 3 buttons, one with a left arrow (the "back" button), one with a right arrow (the "forward" button), and a third with a house on it (the "home" button). The help is a basic hypertext system, and you can browse it just like you would with a web browser, using the home button to go back to where you started, and the back and forward buttons to move through your history. Underneath this there are 4 tabs with titles on them, and you can click on these tabs to bring up a different aspect of the help system.



The Help Box

The first tab is titled "Help choosing a test". This interactive help system will ask you a series of questions until it finally suggests a test that you can perform with details about the test and how to get it working.

The second tab is titled "Topics" and details general help about SalStat - in other words, how to enter data, get descriptive statistics and how to analyse. There are also pages about each test that is offered and what is needed to do the test.

The third tab is titled "License" and contains the license under which SalStat is offered in full. This license is called the GNU General Public License ("GPL" for short), and is very popular for open source software. In short it entitles you to use the software for any purpose (within the law of course), and you may inspect the source code to check I am not downloading files from your computer. You can even change the source code to your liking if you wish with the only proviso being that if you distribute the changes, you must make the source available.

The fourth tab is titled "Peeps" (thank you Harry Enfield & co!), and has a brief list of all the people who have been involved in or help with SalStat over the course of its development.

Most of the help offered is also contained in this document, but sometimes it is useful to have help immediately on hand, and other times it is useful to have a paper document available for perusal.

Chapter 15

Advanced Guide for Programmers

The beauty of SalStat is that it is a free program. This doesn't just mean that you can have it for no money, but it means mostly that you can do lots of interesting things with it if you have the time, skills and inclination.

SalStat is *open source*. What this means is that the source code to the program is available, and you don't have to sign any non-disclosure agreements to see it, or even use it. The license that the program comes under (the GNU General Public License) specifies that you can access the code if you want, make any modifications you want, or give it away or sell it to other people if you want. Get more information from the Free Software Foundation (http://www.fsf.org).

Because the source code is freely available and modifiable, anyone with the skills can tailor SalStat to their own needs. If you want to write a new function, it can be done easily. If you don't like parts of the GUI, you can change them to suit yourself.

This section is designed to illustrate the internal architecture of SalStat in order to make reprogramming it less of a chore. First of all, I shall specify the overall build of the program, and then expand on the classes and functions contained within each module.

Appendix A - Technical Details

SalStat was written using the Python language (http://www.python.org) and wxPython to construct the GUI (http://www.wxpython.org). The terminal (character) based version (still to be released) uses the curses module of Python. Python was chosen for a number of reasons: Firstly, it is easy to read and write which made the initial coding and the subsequent maintenance of the program a simpler business. Secondly, any code written in Python can be released under any license desired. Thirdly, Python is a truly cross-platform language with ports to all major platforms. Fourthly, due to its open source background, Python is also a surprisingly mature language, and any rare bugs that exist tend to be fixed very quickly. Fifthly, the community around Python is very dynamic and supportive. Sixthly, being easy to learn, it is easier for people to modify the code for their purposes if they so wish. Seventhly, it is a "free" language and not under proprietary control.

Development was done on an intel workstation running no name components. The operating system was Mandrake Linux 8.0 to 8.2, but it was also tested occasionally on a Windows 98 and NT workstation to test whether it was truly cross-platform.

The design of SalStat is modular: The module with the statistics test relies heavily upon the "stats.py" and "pstat.py" modules written by Gary Strangman of Harvard University (http://ref!) and released under the GPL. Because of this, SalStat also has to be released under the full GPL (not the LGPL). The IDE used was initially Emacs (http://www.emacs.org) and then the wonderful SciTE editor (http://www.scintilla.org), a newer generation editor that is fast, flexible, and works superbly for editing Python code.

Appendix B - Other Statistics Packages

Free/Libre/Open Source packages

R - R is a free version of the S-Plus language that is particularly suited to heavy duty statistical manipulation.

Lisp-stat - this is a mature project by Luke Tierney and is again a language suitable for statistical manipulation. It also has a GUI called Vista.

Macanova - This project is well suited to analysis of variance, but despite the name, it is not solely for that purpose.

Statist and StatistX

Commercial Packages

SPSS

StatView

SAS

Minitab

Statistica

Appendix C - API details for scripting

These functions can be called instantly from the scripting window. They are called by clicking on the "Go" button on the toolbar (the first button on the left with a circular red arrow). Please note that Python is case-sensitive - the letter 'A' is different from 'a', so make sure that you use the correct case. The following functions have the syntax and a brief description of what they do. If you use a scripting window, you canmix and match any number of functions and even write your own tests.

15.1 Data Access Functions

In order to perform some analysis, you first of all need data to analyse. To enter data, you can either type them into the grid and retrieve them from there, or you can enter them into a variable at the scripting window (and even put them into the grid it you wish).

`function GetData(`*`datasource`*`)`

the datasource can be a list or an integer. If it is a list, then the list will be used as the data. Passing an integer will retrieve the specified column of data from the datagrid. *NOTE*: When calling this, remember that in SalStat, the first column of data is column 0, not 1. Example: GetData(1) will get data from column 1 on the grid.

`function PutData(`*`datasource`*`, `*`data`*`)`

this puts the data contained the in variable *data* into the grid at column *datasource*. Example: `PutData(0, ["3","5.67","7.89"])` will put the numbers

3, 5.67 and 7.89 into the first column of the grid.

`function display(`*`data`*`)`

this takes whatever is contained in *data* and shows it in the output window. Example: `display("Hi")` will write "Hi" to the output window. Each new entry is automatically started on a new line (i.e., it has a base level of html formatting), but other html can be included if needed.

15.2 Descriptive Statistics

`class Describe(`*`datasource`*`)` - This class will get all the descriptive statistics for *datasource*. Each individual statistic is accessed as an attribute of this class. *datasource* may be a list of data, or an integer to represent the column on the data grid. Example: `a = Describe(1)` will get descriptive statistics for column 1 on the grid and store them in the variable *a*.

Available statistics (all for non-missing data except for, well, the missing data statistic:

`Name` - the variable name (the column heading)

`N` - number of the data;

`sum` - the sum of the data;

`mean` - the mean of the data;

`samplevar` - the variance of the data;

`stddev` - the standard deviation of the data;

`stderr` - the standard error of the data;

`sumsquares` - the sum of squares of the data;

`ssdevs` - the sum of squared deviations of the data;

`coeffvar` - the coefficient of variation of the data;

`minimum` - the minimum value of the data;

`maximum` - the maximum value of the data;

`range` - the range of the data (maximum - minimum);

`numbermissing` - the number of missing data;

`geomean` - the geometric mean of the data;

`harmmean` - the harmonic mean of the data;

`skewness` - the skewness of the data;

`kurtosis` - the kurtosis of the data;

`median` - the median of the data;

`mad` - the median absolute deviation of the data;

`mode` - the mode of the data;

`interquartilerange` - the interquartile range of the data;

`uniques` - the number of unique levels (i.e., non-occuring values) of the data;

To those unfamiliar with object-oriented approaches, access to these might be a little but unusual at first, but it is quite easy. Simply add the descriptive after the variable name (with a dot in between!) and they can be used. Example:

`a = Describe(0)` - gets the descriptives for the data in the first grid column;

`display(a.N)` - shows on the output sheet how many non-missing data there are;

`a.samplevar * a.samplevar` - the sample variance is squared here.

`MyOwnRange = a.maximum - a.minimum` - this works out the range (which is already calculated and accessed as a.range).

15.3 Inferential Statistics

These perform the actual tests. Each test returns more than one value (for example, the t-test returns the value of t, the degrees of freedom, and the associated probability). The way to script these are as follows:

```
t, df, p = DoUnpairedTTest(0,1)
```

This will perform an unpaired t test. Obviously, the variable t will hold the value of t, the df will hold the degrees of freedom and so on. Like the descriptive statistics, you will need to specify the data source, and this is done in the brackets. You can form a list of data and send those, or you

can specify a column from the grid. In the above example, the data from columns 0 and 1 will be retrieved and used.

All these tests assume a 2-tailed hypothesis. If that is what you want, then you don't need to specify anything else, but if you want a one tailed test, just add a '1' after the data sources like so:

```
t, df, p = DoUnpairedTTest(0, 1, 1)
```

Remember that you can call these variables anything you want, but the order in which they are returned is fixed. The following is acceptable:

```
p, t, df = DoUnpairedTTest(0, 1)
```

This will perform an unpaired t-test using the data from columns 0 and 1 tested against a 2-tailed hypothesis, but the value of t will be stored in the variable p, the degrees of freedom will be in t, and the p-value will be stored in df!

`DoOneSampleTTest(`*`source, usermean, tail = 2`*`)`

This is the one sample t-test and returns *t, df, probability.* The user hypothesised mean needs to be specified.

`DoOneSampleSignTest(`*`source, usermean, tail = 2`*`)`

This performs the one sample sign test, a nonparametric equivilent of the one sample t-test. Again, the user hypothesised mean needs to be specified. Returns *nplus, nminus, z, probability.*

`DoChiSquareVariance(`*`source, usermean, tail = 2`*`)`

This performs the chi square for variance ratio's. Like the other one sample tests, the user hypothesised mean should be specified. This test returns *chi, df, probability.*

`DoPairedTTest(`*`source1, source2, tail = 2`*`)`

This is for a paired t-test and returns *t, df, probability.*

`DoUnpairedTTest(`*`source1, source2, tail = 2`*`)`

This is the unpaired t-test and returns *t, df, probability.*

`DoPearsonsCorrelation(`*`source1, source2, tail=2`*`)`

This performs the Pearsons correlation, a correlation for parametric data and returns *t, r, df, probability.*

`DoFTest(`*source1, source2, uservariance, tail=2*`)`

This does an F-test, and uniquely among the 2-sample tests, an extra value needs to be specfied, the user hypothesised variance. Returned values are *t, r, df, probability.*

`DoSignTest(`*source1, source2, tail=2*`)`

This is the sign test, a nonparametric equivilent of the paired t-test, and it returns *z, probability.*

`DoKendallsCorrelation(`*source1, source2, tail=2*`)`

This performs a Kendalls tau correlation and returns *tau, z, probability.*

`DoKSTest(`*source1, source2, tail=2*`)`

This performs a Kolmogorov-Smirnov test and returns *d, probability.*

`DoSpearmansCorrelation(`*source1, source2, tail=2*`)`

This does a Spearmans nonparametric correlation and it returns *rho, t, df, probability.*

`DoRankSums(`*source1, source2, tail=2*`)`

This performs a Wilcoxon rank sums test, a nonparametric version of the unpaired t-test and it returns *z, probability.*

`DoSignedRanks(`*source1, source2, tail=2*`)`

This is for the Wilcoxon sgned ranks test, and returns *z, wt, probability.*

`DoMannWhitneyTest(`*source1, source2, tail=2*`)`

This does a Mann-Whitney U test and returns *bigu, smallu, z, probability.*

`DoLinearRegression(`*source1, source2, tail=2*`)`

This should perform a linear regression and it returns *df, r, slope, intercept, estimated standard error, probability.*

`DoPairedPermutation(`*source1, source2, tail = 2*`)`

This gets you a paired permutation test and returns *nperm, probability.*

15.4 Probability Functions

These functions will return a p-value given the right data except for two of them which return the critical value for a given p-value. All of the information

requested in these functions is essential - without it, SalStat will not be able to generate an answer.

`GetChiProb(`*chi, df*`)`

Given chi and the degrees of freedom, this returns the associated p-value.

`GetInverseChiProb(`*prob, df*`)`

Returns the value of chi for the given probability and df.

`GetZProb(`*z*`)`

Returns the p-value of the z-score.

`GetKSProb(`*ks*`)`

Given the critical value from a Kolmogorov-Smirnov test, this return its probability.

`GetTProb(`*t, df*`)`

Returns the probability for a t-test (or Pearsons and Spearmans correlation)

`GetFProb(`*F, df1, df2*`)`

Returns the p-value for an anova given F and the two df's.

`GetInverseFProb(`*prob, df1, df2*`)`

The inverse of the above, this returns the critical value of F given the probability and degrees of freedom specified.

15.5 Other Functions

Because SalStat uses a Python interpreter, you may have access to all the Python functions. One of the great things about Python is that it has an enormous range of modules available for almost any purpose, from the basics of file manipulation, right down to controlling a web browser or sending emails. Instead of providing you with an introduction to Python, I will instead guide you to the best source of documentation at the Python web site (http://www.python.org). Just click on documentation.

Another source of information is usenet. There are a couple of groups dedicated to Python, the best of which is "comp.lang.python". If you are new

to the language and have a question, then the chances are that it has been asked already so try searching Google groups before posting a message to see what people have said before. If you do have an original question, you will find the group to be one of the most knowledgable, friendly and helpful groups on usenet, which really is saying something!

Appendix D - Thanks!

Many people have encouraged me to write this over time, and I would like to thank them. Apologies for anyone who has been missed out - contact me and I will amend this!

Contributors (unwitting or otherwise)

Gary Strangman of Harvard University (author of stats.py and pstat.py which made my task so much easier), Federico Seville of the University of Phillipines, Ludger Humbert (who has the honour of being the first person to get SalStat working on OSX I believe), and Dave Mitchell of Queens University in Belfast.

Others

Nick Perham, Robert Hughes, Hans Neth and Phil Barnes - my PhD colleagues who gave me encouragement and feedback (among *many* others!)

The South Wales Linux Users Group for their advice.

The Python, wxWindows and wxPython community (especially Robin Dunn - software craftsman) for their work.

The comp.lang.python newsgroup, one of the friendliest and knowledgable newsgroups around.

My parents and step-parents (Mum, Dad, Chris, Ann). Obviously.

My brothers (David, Rob, Howard).

My grandmother for many hours of patience!

Me (well, let's face it. Without my motivation and skills, this project simply wouldn't exist!)

Appendix E - Future Improvements

This is a list of future improvements that I would like to implement into SalStat. If you have any others, let me know. I cannot promise to do them, but if they are good, I will try to see what I can do.

Other language interpreters: Tcl, Perl mainly, maybe Lisp and Scheme.

Multi-factorial anova with simple effects easily accessible (probably one of the most requested features in a modern stats package).

MANOVA and MANCOVA

Multiple regression, factor analysis, power analysis and lots of other tests.

Other file format import (Excel, SPSS at least).

Improved copy and paste on the data grid - currently, it only supports cutting, copying and pasting one cell, but whole blocks are aimed for.

Including new tests into the GUI - if anyone writes a useful test, SalStat should enable access to it from the GUI. This could be complex as each test may require different information from the user, but a generic user interface may be workable.

Porting to more platforms (mostly Mac OS7/8/9, BeOS and QNX) - any other requests?

Better graphing/charting facilities, maybe with links to external graphing programs for professional output (like Gnuplot or Dislin).

Integration with Numeric to allow faster analysis of very large data sets (Numeric is a Python module that is compiled from C and allows very fast and powerful matrix manipulation).

CD-ROM's available for sale from the web site.

Creation of raw data from analysis results - this isn't for cheating, rather for creating datasets for demonstration or teaching purposes.