Other Statistical SoftwareStatistics 506

Other Software

While R, Stata and SAS are the most popular statistical software amongst practicing data analysts, there are a number of other pieces of statistical software that non-statistically oriented people tend to use. The goal of these notes is to offer you a passing familiarity with these tools and their strengths and weaknesses so if you encounter someone using them, you can understand where they're coming from.

None of the material in these notes will appear on problem sets or the midterm.

One common theme that will come up is that *none* of these other pieces of software offers support for the same wide range of statistical models and analyses that R, Stata and SAS cover. In some cases (Python in particular) they may support the models, but not more advanced usages of them. They will support basic analyses like t-tests and linear regression, though they may require an additional tool (such as Excel's Analyse-it) to avoid carrying out these analyses manually.

Programming Languages

Python

Python is an open source general purpose programming language that has, in recent years, become popular as a tool for statistical analysis. It's use is covered heavily in Statistics 507, which is why we only include it here.

Python supports dynamic documents via Quarto, or others such as jupyter. RStudio has support for these, as well as running interactive Python sessions just like R.

Because Python is a general purpose programming language and not purely for statistical analysis, it requires a number of third-party libraries to perform any statistical analysis. It also can be more frustrating to install than R.

NumPy

NumPy adds support for large matrix-style objects and functions associated with them. An example from the quickstart:

```
>>> import numpy as np
>>> a = np.array([20, 30, 40, 50])
>>> b = np.arange(4)
>>> b
array([0, 1, 2, 3])
>>> c = a - b
>>> c
array([20, 29, 38, 47])
>>> b**2
array([0, 1, 4, 9])
>>> 10 * np.sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a < 35
array([ True,  True, False, False])
>>> A = np.array([[1, 1],
...               [0, 1]])
>>> B = np.array([[2, 0],
...               [3, 4]])
>>> A * B     # elementwise product
array([[2, 0],
       [0, 4]])
>>> A @ B     # matrix product
array([[5, 4],
       [3, 4]])
>>> A.dot(B)  # another matrix product
array([[5, 4],
       [3, 4]])
```

Pandas

Pandas introduces an analogue of R's `data.frame`: the `DataFrame`. It is built on top of NumPy. An example from the getting started tutorials:

```
>>> import pandas as pd
>>> df = pd.DataFrame(
...     {
...         "Name": [
```

```
...             "Braund, Mr. Owen Harris",
...             "Allen, Mr. William Henry",
...             "Bonnell, Miss. Elizabeth",
...         ],
...         "Age": [22, 35, 58],
...         "Sex": ["male", "male", "female"],
...     }
... )
...
>>> df
                       Name  Age     Sex
0   Braund, Mr. Owen Harris   22    male
1  Allen, Mr. William Henry   35    male
2  Bonnell, Miss. Elizabeth   58  female
>>> df["Age"]
0    22
1    35
2    58
Name: Age, dtype: int64
>>> df["Age"].max()
58
>>> titanic = pd.read_csv("data/titanic.csv")
>>> titanic.head()
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S

[5 rows x 12 columns]
>>> titanic["Age"].shape
(891,)
>>> above_35 = titanic[titanic["Age"] > 35]
>>> above_35.head()
    PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
1             2         1       1  ...  71.2833   C85         C
6             7         0       1  ...  51.8625   E46         S
11           12         1       1  ...  26.5500  C103         S
13           14         0       3  ...  31.2750   NaN         S
15           16         1       2  ...  16.0000   NaN         S
```

```
[5 rows x 12 columns]
>>> above_35.shape
(217, 12)
```

SciPy

SciPy implements a large suite of scientific computing functions. A lot of these may not be interesting to us as statisticians except in niche situations, such as fast fourier transformations or signal processing. However it does have a `stats` subpackage that is very handy and implements basic statistical analyses. For example here's a linear regression example from the SciPy API reference:

```
>>> import numpy as np
>>> from scipy import stats
>>> rng = np.random.default_rng()
>>> x = rng.random(10)
>>> y = 1.6*x + rng.random(10)
>>> res = stats.linregress(x, y)
>>> res.slope
2.0401139933368753
>>> res.intercept
0.22541055389034326
```

`statsmodels`

`statsmodels` is a library focused solely on statistical data exploration, hypothesis testing, and modeling estimation. It implements a wide range of statistical analyses, though does not handle as many models as R's various packages do. An example from getting started:

```
>>> import statsmodels.api as sm
>>> import pandas
>>> from patsy import dmatrices
>>> df = sm.datasets.get_rdataset("Guerry", "HistData").data
>>> y, X = dmatrices('Lottery ~ Literacy + Wealth + Region', data=df, return_type='datafra
>>> y[:3]
   Lottery
0     41.0
1     38.0
2     66.0

>>> X[:3]
```

```
   Intercept  Region[T.E]  Region[T.N]  ...  Region[T.W]  Literacy  Wealth
0        1.0          1.0          0.0  ...          0.0      37.0    73.0
1        1.0          0.0          1.0  ...          0.0      51.0    22.0
2        1.0          0.0          0.0  ...          0.0      13.0    61.0

[3 rows x 7 columns]
>>> mod = sm.OLS(y, X)    # Describe model
>>> res = mod.fit()       # Fit model
>>> print(res.summary())  # Summarize model
                            OLS Regression Results
==============================================================================
Dep. Variable:                Lottery   R-squared:                       0.338
Model:                            OLS   Adj. R-squared:                  0.287
Method:                 Least Squares   F-statistic:                     6.636
Date:                Fri, 05 May 2023   Prob (F-statistic):           1.07e-05
Time:                        13:59:50   Log-Likelihood:                -375.30
No. Observations:                  85   AIC:                             764.6
Df Residuals:                      78   BIC:                             781.7
Df Model:                           6
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept      38.6517      9.456      4.087      0.000      19.826      57.478
Region[T.E]   -15.4278      9.727     -1.586      0.117     -34.793       3.938
Region[T.N]   -10.0170      9.260     -1.082      0.283     -28.453       8.419
Region[T.S]    -4.5483      7.279     -0.625      0.534     -19.039       9.943
Region[T.W]   -10.0913      7.196     -1.402      0.165     -24.418       4.235
Literacy       -0.1858      0.210     -0.886      0.378      -0.603       0.232
Wealth          0.4515      0.103      4.390      0.000       0.247       0.656
==============================================================================
Omnibus:                        3.049   Durbin-Watson:                   1.785
Prob(Omnibus):                  0.218   Jarque-Bera (JB):                2.694
Skew:                          -0.340   Prob(JB):                        0.260
Kurtosis:                       2.454   Cond. No.                         371.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified
```

Julia

Julia is somewhat of a combination of Python and R - it's technically an open source general-purpose programming language like Python, but is oriented towards numerical and statistical analyses like R. It made a big splash when it appeared on the scene about 10 years ago, and is moderately popular, but never lived up to it's promise to dethrone R as the most common statistical software.

Julia was designed to be much more efficient than existing high-level interactive languages. Its syntax is very similar to R. Taken from the Julia documentation:

```
julia> 3 \ 6
2.0

julia> inv(3) * 6
2.0

julia> A = [4 3; 2 1]; x = [5, 6];

julia> A \ x
2-element Vector{Float64}:
  6.5
 -7.0

julia> inv(A) * x
2-element Vector{Float64}:
  6.5
 -7.0
```

Here's a regression example from GLM example:

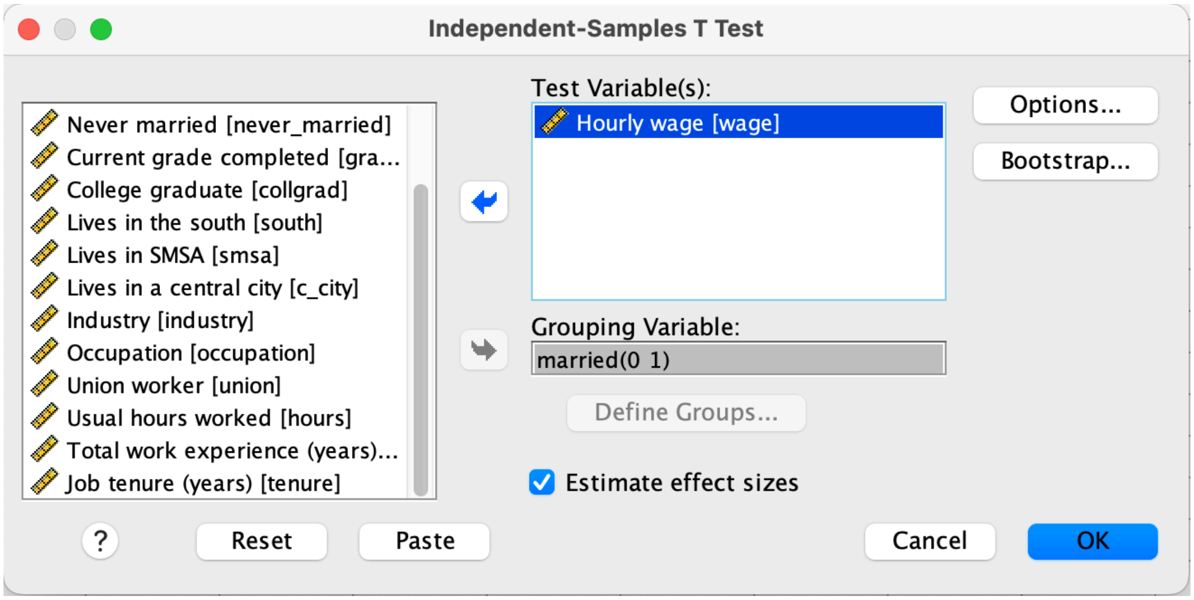
```
julia> using DataFrames, GLM, StatsBase

julia> data = DataFrame(X=[1,2,3], Y=[2,4,7])
3×2 DataFrame
 Row   X      Y
       Int64  Int64

   1      1      2
   2      2      4
   3      3      7
```

```
julia> ols = lm(@formula(Y ~ X), data)
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredCho

Y ~ 1 + X

Coefficients:

                 Coef.  Std. Error      t  Pr(>|t|)  Lower 95%  Upper 95%

(Intercept)  -0.666667    0.62361   -1.07    0.4788   -8.59038    7.25704
X             2.5         0.288675   8.66    0.0732   -1.16797    6.16797


julia> round(r2(ols); digits=5)
0.98684

julia> round(aic(ols); digits=5)
5.84252

julia> round.(vcov(ols); digits=5)
2×2 Matrix{Float64}:
  0.38889  -0.16667
 -0.16667   0.08333
```

Graphical Software

SPSS

SPSS used to be one of the dominant pieces of proprietary statistical software amongst people who wanted to do statistical analysis but weren't actual statisticians. SPSS has a user-friendly interface that can be entirely driven via GUI, allowing users to accomplish almost everything without ever writing any syntax (SPSS calls their code "syntax"). For example, here is the dialog for carrying out a two-sample t-test:



which produces as an output

T-Test

[DataSet1] /Users/jerrick/Desktop/nlsw88.sav

Group Statistics

	Married	N	Mean	Std. Deviation	Std. Error Mean
Hourly wage	Single	804	8.08	6.336	.223
	Married	1442	7.59	5.399	.142

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means							
						Significance				95% Confidence Interval of the Difference	
		F	Sig.	t	df	One-Sided p	Two-Sided p	Mean Difference	Std. Error Difference	Lower	Upper
Hourly wage	Equal variances assumed	11.222	<.001	1.931	2244	.027	.054	.489	.253	-.008	.985
	Equal variances not assumed			1.845	1452.197	.033	.065	.489	.265	-.031	1.008

Independent Samples Effect Sizes

		Standardizer[a]	Point Estimate	95% Confidence Interval	
				Lower	Upper
Hourly wage	Cohen's d	5.752	.085	-.001	.171
	Hedges' correction	5.754	.085	-.001	.171
	Glass's delta	5.399	.091	.004	.177

a. The denominator used in estimating the effect sizes.
Cohen's d uses the pooled standard deviation.
Hedges' correction uses the pooled standard deviation, plus a correction factor.
Glass's delta uses the sample standard deviation of the control (i.e., the second) group.

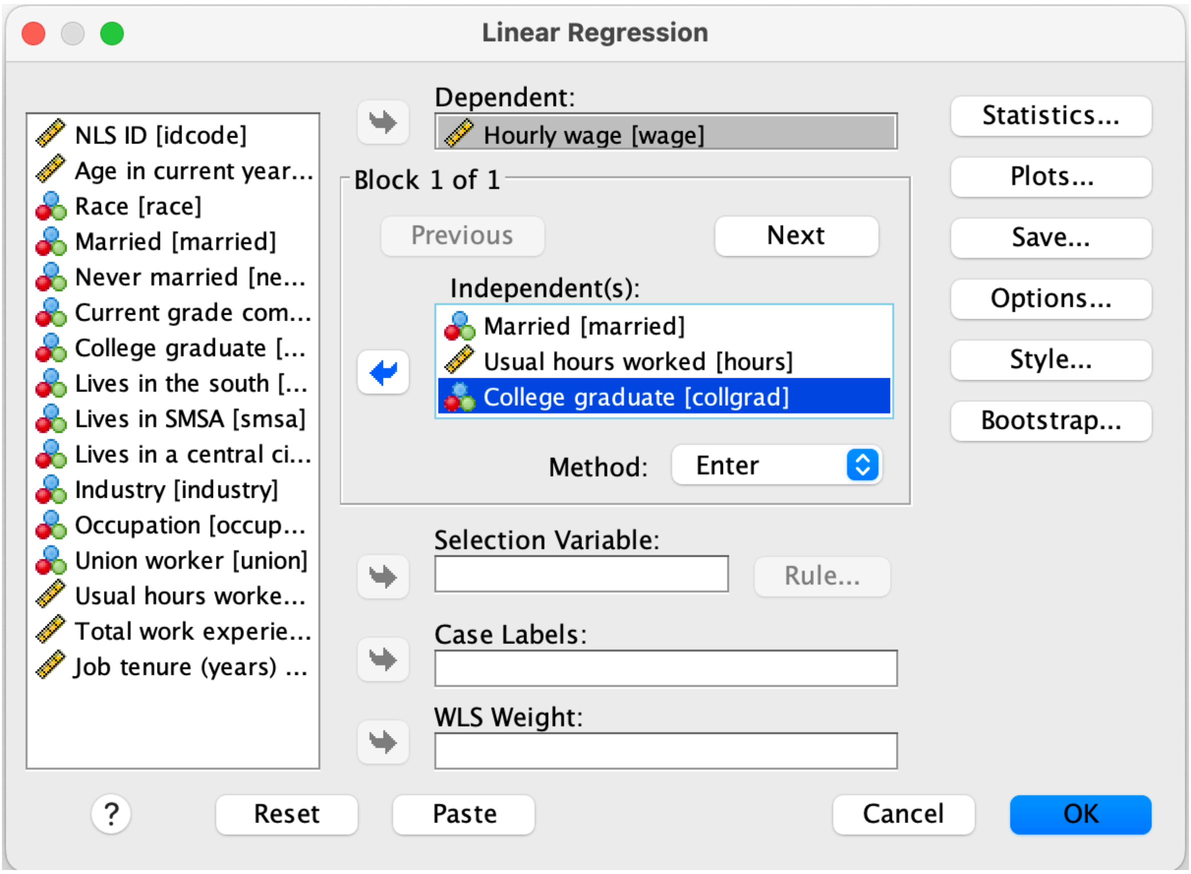
The corresponding syntax would be:

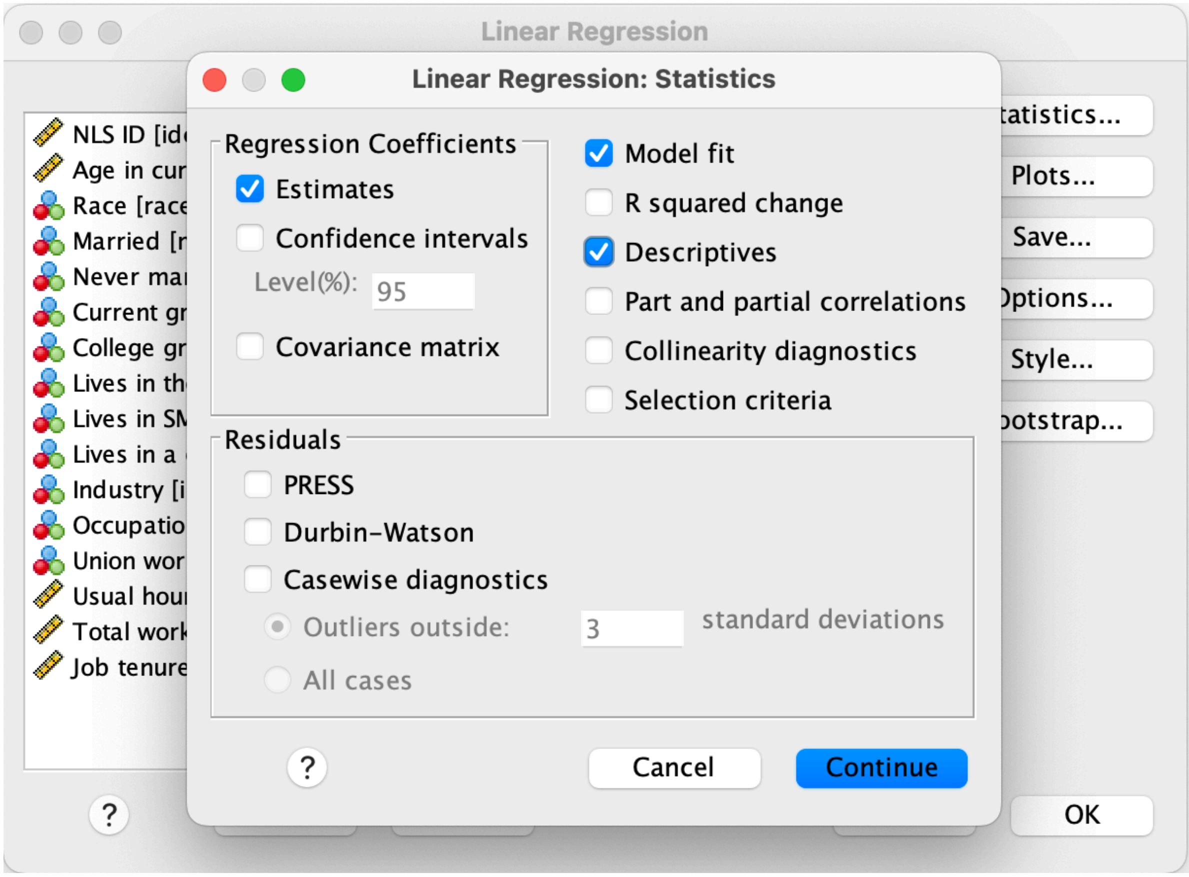
```
T-TESTS GROUPS=married(0 1)
  /VARIABLES=wage
  /ES DISPLAY(TRUE).
```

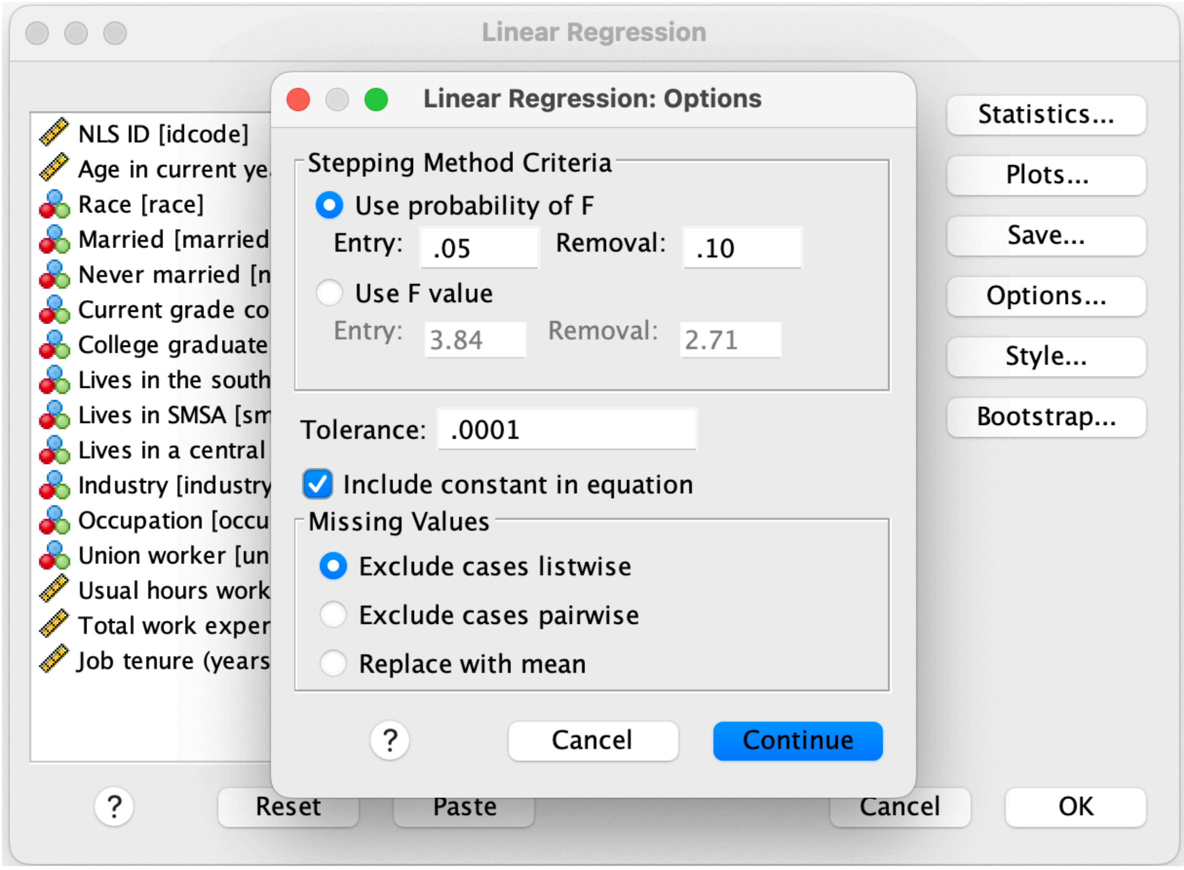
The capitalization is by convention, and is not enforced. Note that the vast majority of SPSS users do not know and never use the syntax, they use the dialog boxes.

SPSS's use over the last 5-10 years has waned substantially. This is primarily due to the other statistical software catching up in terms of user-friendliness. SPSS also supports far less advanced models than R, Stata, SAS, and Python.

Another example, this of linear regression. Often in these dialog boxes, more complex options are in sub-dialogs.




Linear Regression
Linear Regression: Statistics
Regression Coefficients
Estimates
Confidence intervals
Level(%): 95
Covariance matrix
Model fit
R squared change
Descriptives
Part and partial correlations
Collinearity diagnostics
Selection criteria
Residuals
PRESS
Durbin-Watson
Casewise diagnostics
Outliers outside: 3 standard deviations
All cases
Cancel
Continue
OK



producing as output

Regression

Descriptive Statistics

	Double-click to activate	d. Deviation	N
Hourly wage	7.77	5.758	2242
Married	.64	.479	2242
Usual hours worked	37.22	10.509	2242
College graduate	.24	.425	2242

Correlations

		Hourly wage	Married	Usual hours worked	College graduate
Pearson Correlation	Hourly wage	1.000	-.042	.159	.267
	Married	-.042	1.000	-.143	.006
	Usual hours worked	.159	-.143	1.000	.085
	College graduate	.267	.006	.085	1.000
Sig. (1-tailed)	Hourly wage	.	.023	<.001	<.001
	Married	.023	.	.000	.389
	Usual hours worked	.000	.000	.	.000
	College graduate	.000	.389	.000	.
N	Hourly wage	2242	2242	2242	2242
	Married	2242	2242	2242	2242
	Usual hours worked	2242	2242	2242	2242
	College graduate	2242	2242	2242	2242

Variables Entered/Removed[a]

Model	Variables Entered	Variables Removed	Method
1	College graduate, Married, Usual hours worked[b]	.	Enter

a. Dependent Variable: Hourly wage

b. All requested variables entered.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.301[a]	.091	.090	5.494

a. Predictors: (Constant), College graduate, Married, Usual hours worked

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	6747.189	3	2249.063	74.524	<.001[b]
	Residual	67540.633	2238	30.179		
	Total	74287.822	2241			

a. Dependent Variable: Hourly wage

b. Predictors: (Constant), College graduate, Married, Usual hours worked

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	4.414	.480		9.204	<.001
	Married	-.294	.245	-.025	-1.203	.229
	Usual hours worked	.073	.011	.134	6.544	<.001
	College graduate	3.468	.274	.256	12.660	<.001

a. Dependent Variable: Hourly wage

The corresponding syntax is:

```
REGRESSION
  /DESCRIPTIVES MEAN STDDEV CORR SIG N
  /MISSING LISTWISE
  /STATISTICS COEFF OUTS R ANOVA
  /DEPENDENT wage
  /METHOD=ENTER married hours collgrad.
```

Excel

It's almost impossible to not have some familiarity with Excel. Excel is a good tool for managing small to moderate sized data, and a lot of data projects start out with an Excel spreadsheet. Excel supports some basic statistical tests by default, such as a t-test:

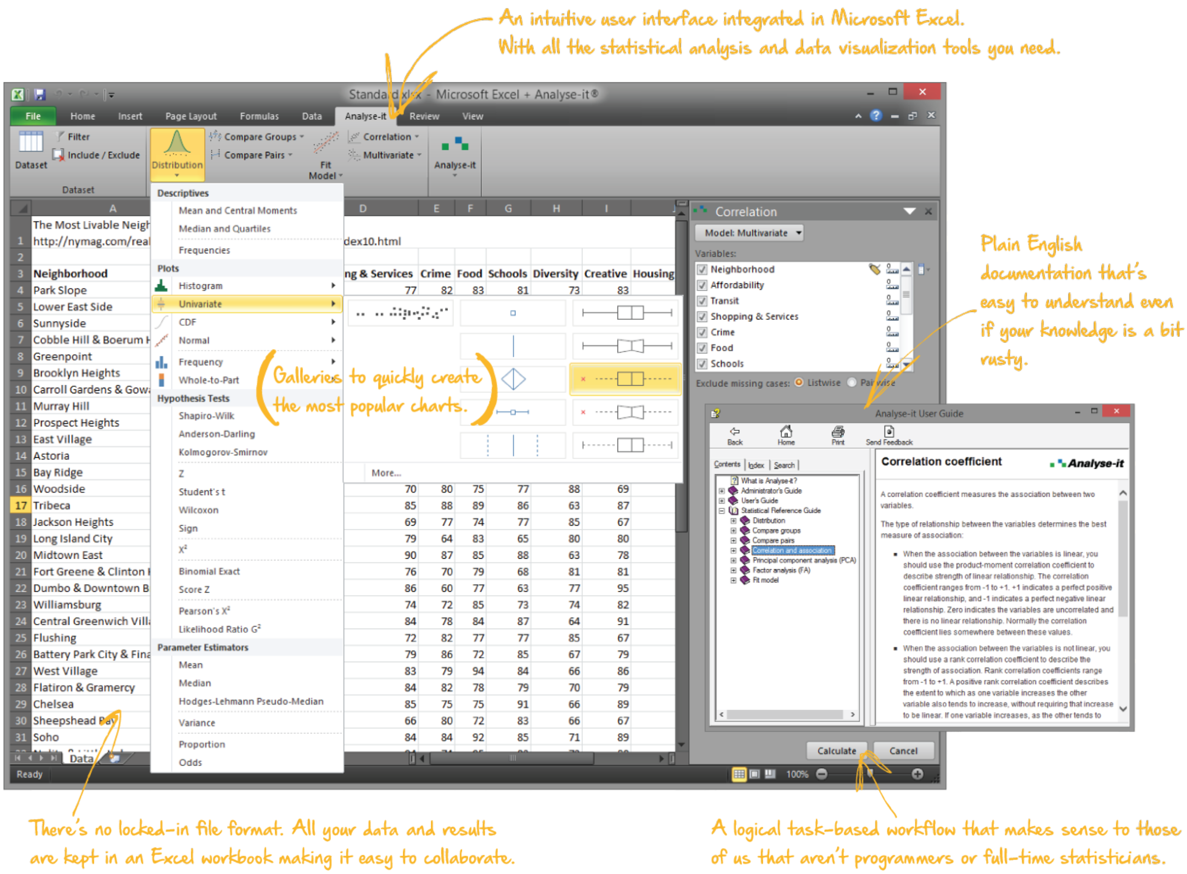
```
=TTEST(A2:A10,B2:B10,2,1)
```

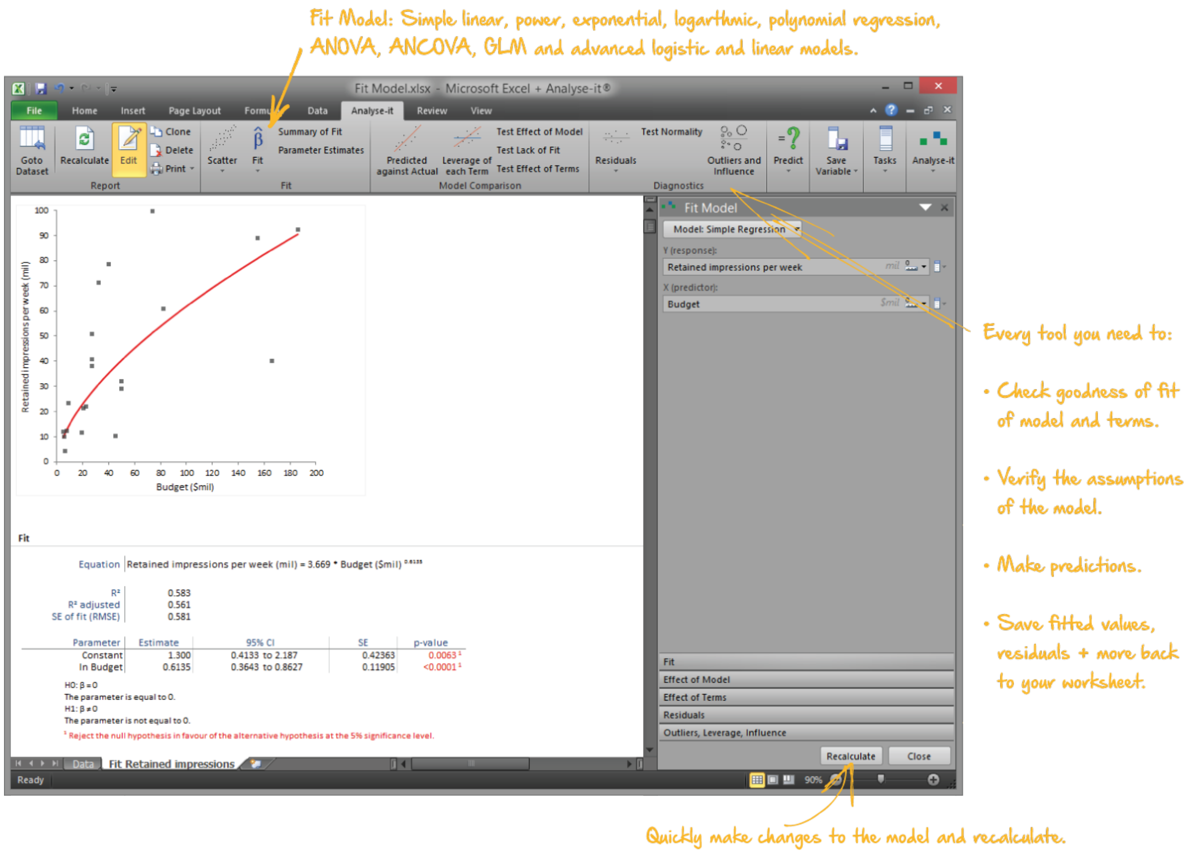
In addition, the Analysis ToolPak adds a few more analysis, such as linear regression:

```
=LINEST(A2:A5,B2:B5,,FALSE)
=LINEST(E2:E12,A2:D12,TRUE,TRUE)
```

Analyse-it

The add-in Analyse-it is an optional purchased component that adds more statistical tools to Excel. Here's some example screenshots from their product page:


An intuitive user interface integrated in Microsoft Excel.
With all the statistical analysis and data visualization tools you need.
Plain English documentation that's easy to understand even if your knowledge is a bit rusty.
Galleries to quickly create the most popular charts.
There's no locked-in file format. All your data and results are kept in an Excel workbook making it easy to collaborate.
A logical task-based workflow that makes sense to those of us that aren't programmers or full-time statisticians.



Using Excel solely for data-management

A more common use of Excel is to use it to manage and manipulate data. Features such as vlookup and pivot tables can be powerful and often much quicker than writing code.

However, keep in mind that such usage does not fall into the reproducible research paradigm, and you should keep good notes of what you've done.

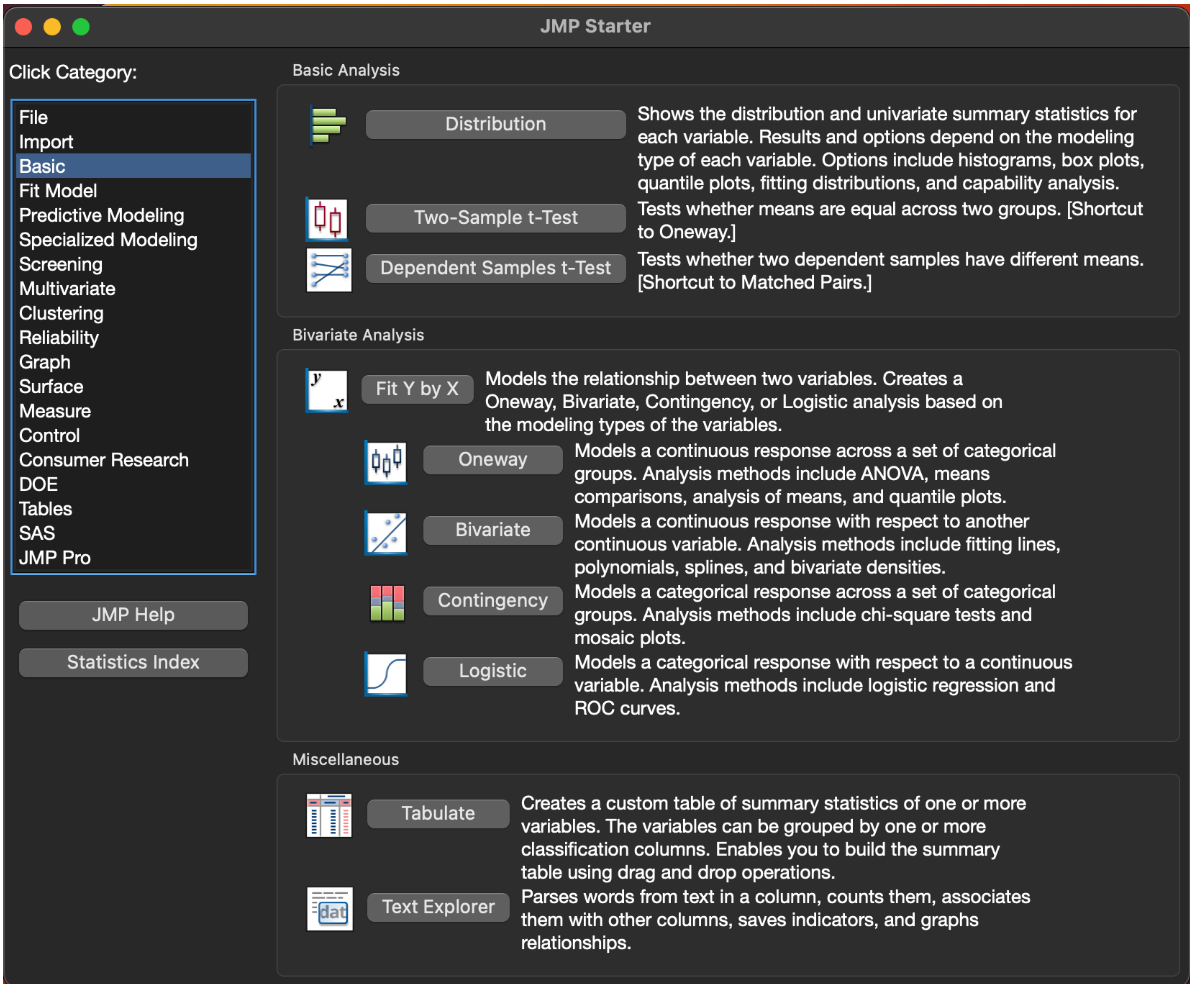
Excel can export to .csv format, which is usually easier to import into statistical software than .xlsx files.

JMP

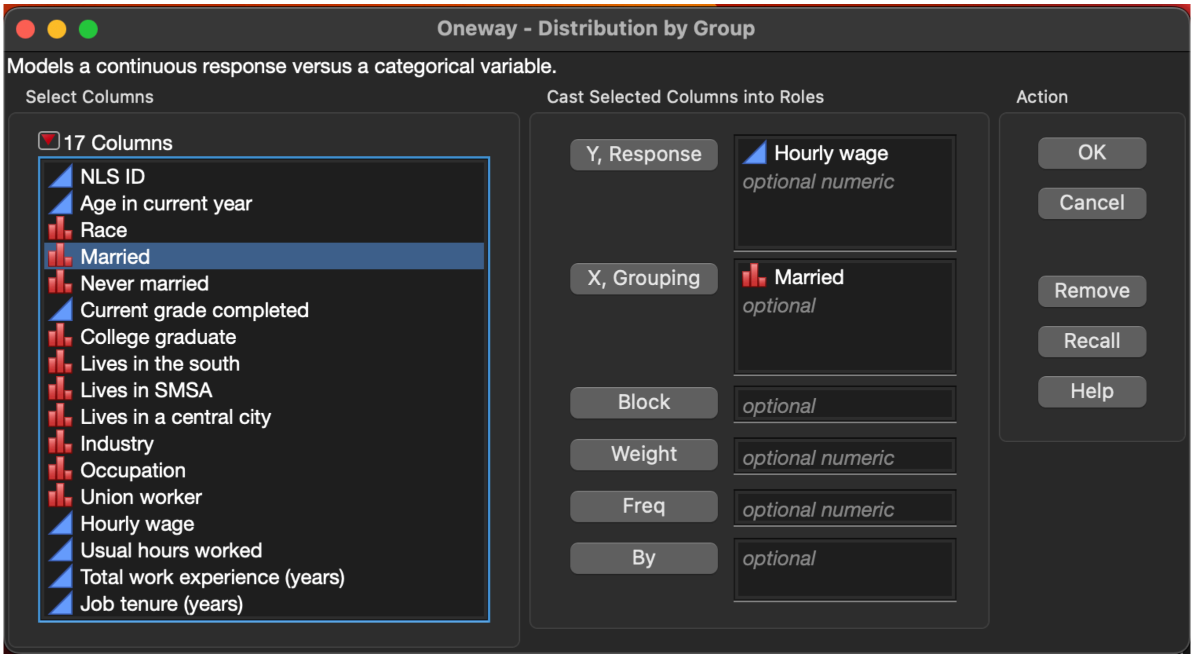
JMP (pronounced "jump") is a statistical analysis suite offered by SAS. JMP is designed to used more for data exploration and visualization than SAS, and as such offers a more GUI-based interaction mode rather than SAS's code-based interaction. Of the various GUI-based statistical software (SPSS, JMP, Prism) it is the most modern, though as usual, it doesn't offer the depth of models.

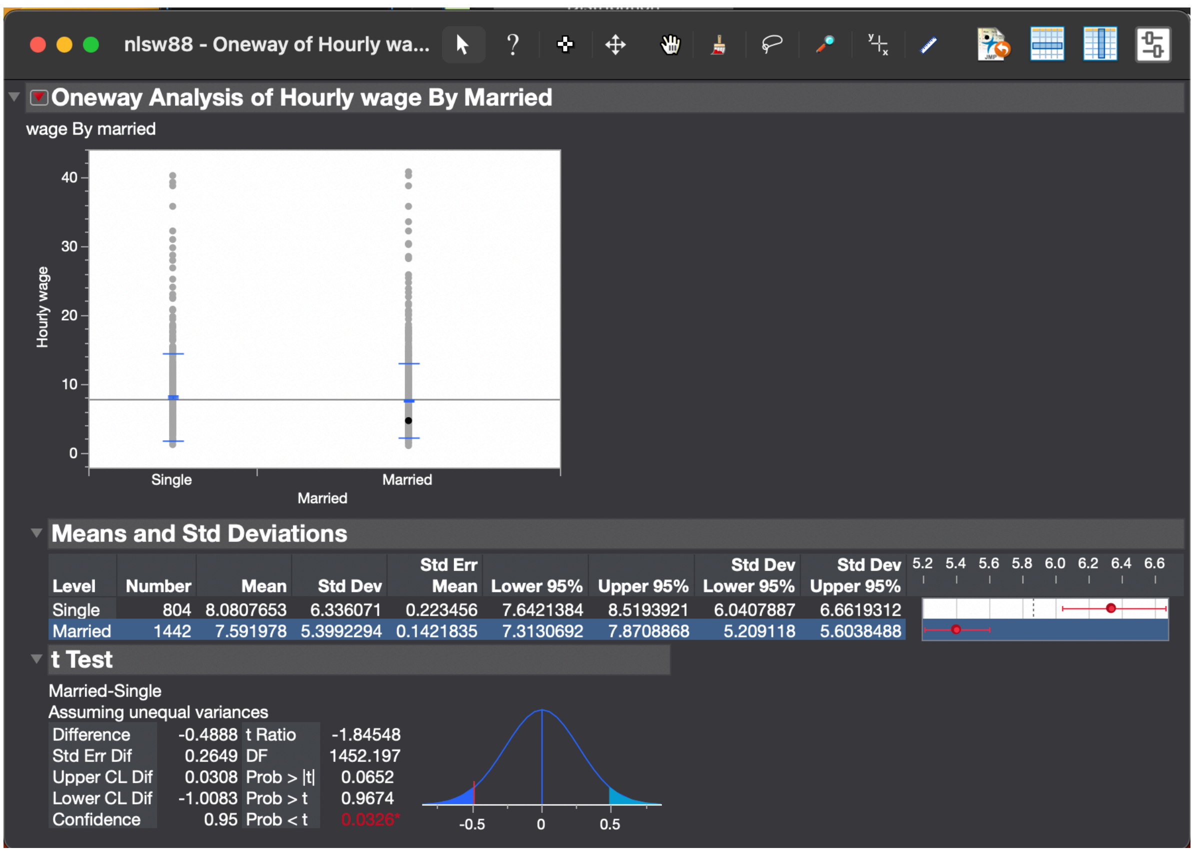
There is the JSL, JMP Scripting Language, that can be used to generate reproducible scripts.

The main interface to choose your analysis:



Repeating the t-test from SPSS:

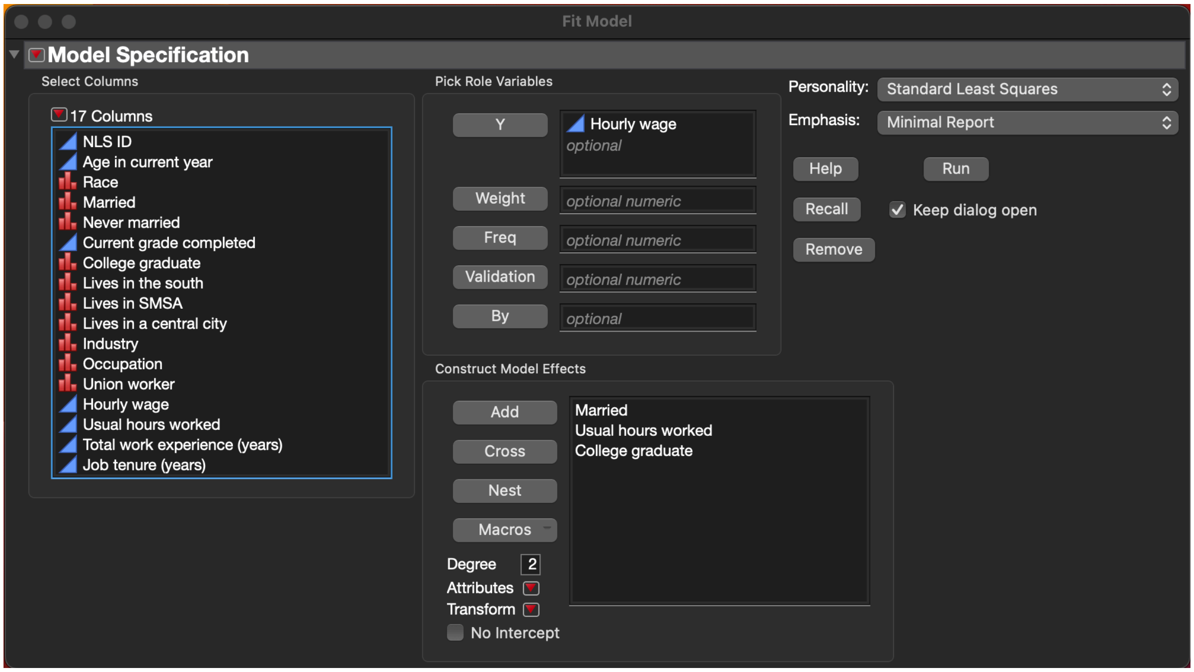




Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%	Std Dev Lower 95%	Std Dev Upper 95%
Single	804	8.0807653	6.336071	0.223456	7.6421384	8.5193921	6.0407887	6.6619312
Married	1442	7.591978	5.3992294	0.1421835	7.3130692	7.8708868	5.209118	5.6038488

Difference	-0.4888	t Ratio	-1.84548
Std Err Dif	0.2649	DF	1452.197
Upper CL Dif	0.0308	Prob > |t|	0.0652
Lower CL Dif	-1.0083	Prob > t	0.9674
Confidence	0.95	Prob < t	0.0326*

Again, repeating the regression from SPSS:



nlsw88 - Fit Least...

Response Hourly wage

Effect Summary

Source	Logworth	PValue
College graduate	34.804	0.00000
Usual hours worked	10.131	0.00000
Married	0.640	0.22914

Remove Add Edit FDR

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio
Lack Of Fit	169	5264.396	31.1503	1.0349
Pure Error	2069	62276.237	30.0997	Prob > F
Total Error	2238	67540.633		0.3682
				Max RSq
				0.1617

Summary of Fit

RSquare	0.090825
RSquare Adj	0.089606
Root Mean Square Error	5.493543
Mean of Response	7.774525
Observations (or Sum Wgts)	2242

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	3	6747.189	2249.06	74.5241
Error	2238	67540.633	30.18	Prob > F
C. Total	2241	74287.822		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	6.0013271	0.450744	13.31	<.0001*
Married[Single]	0.1471652	0.122342	1.20	0.2291
Usual hours worked	0.0732949	0.0112	6.54	<.0001*
College graduate[Not college grad]	-1.734112	0.136971	-12.66	<.0001*

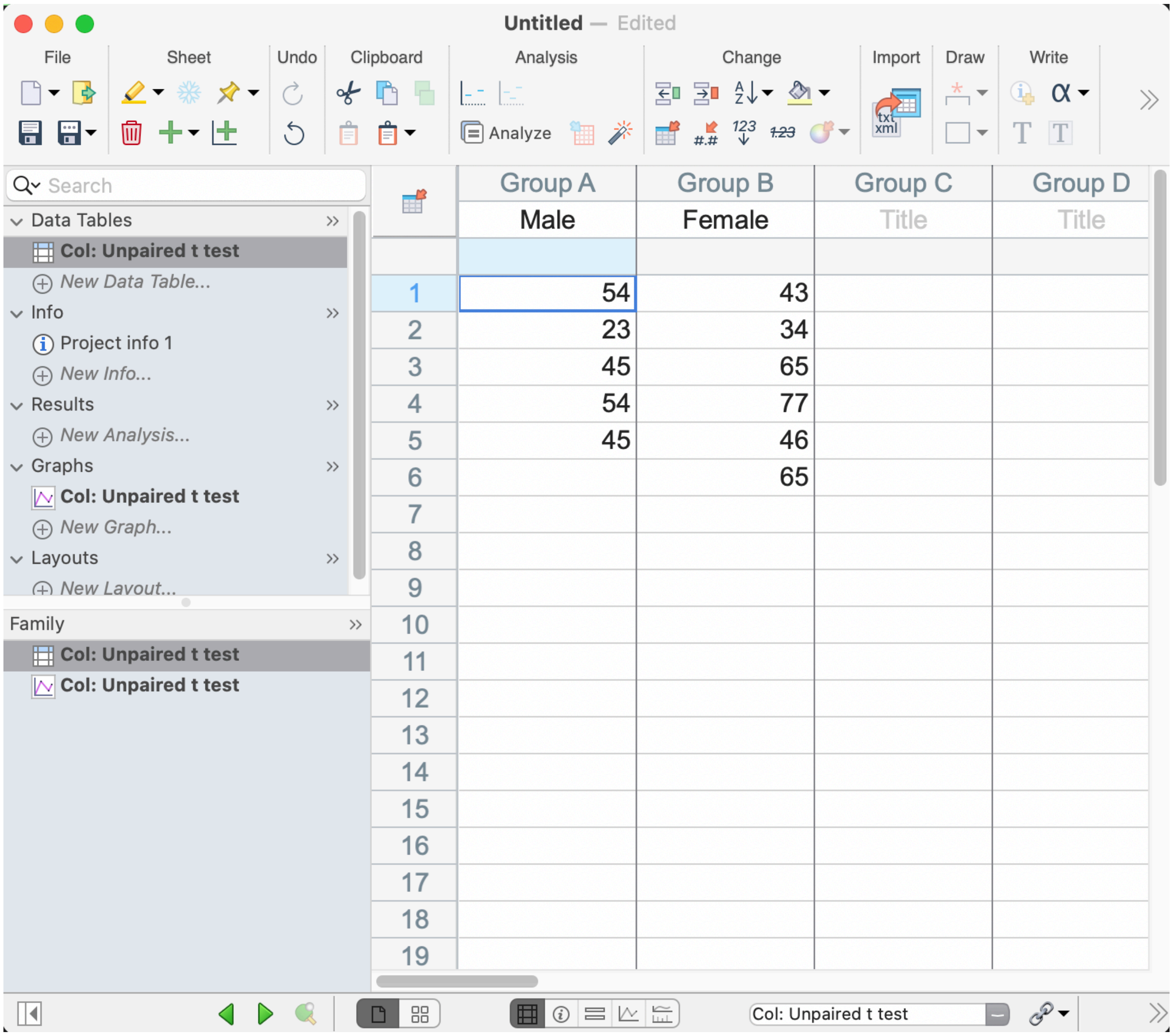
Effect Tests

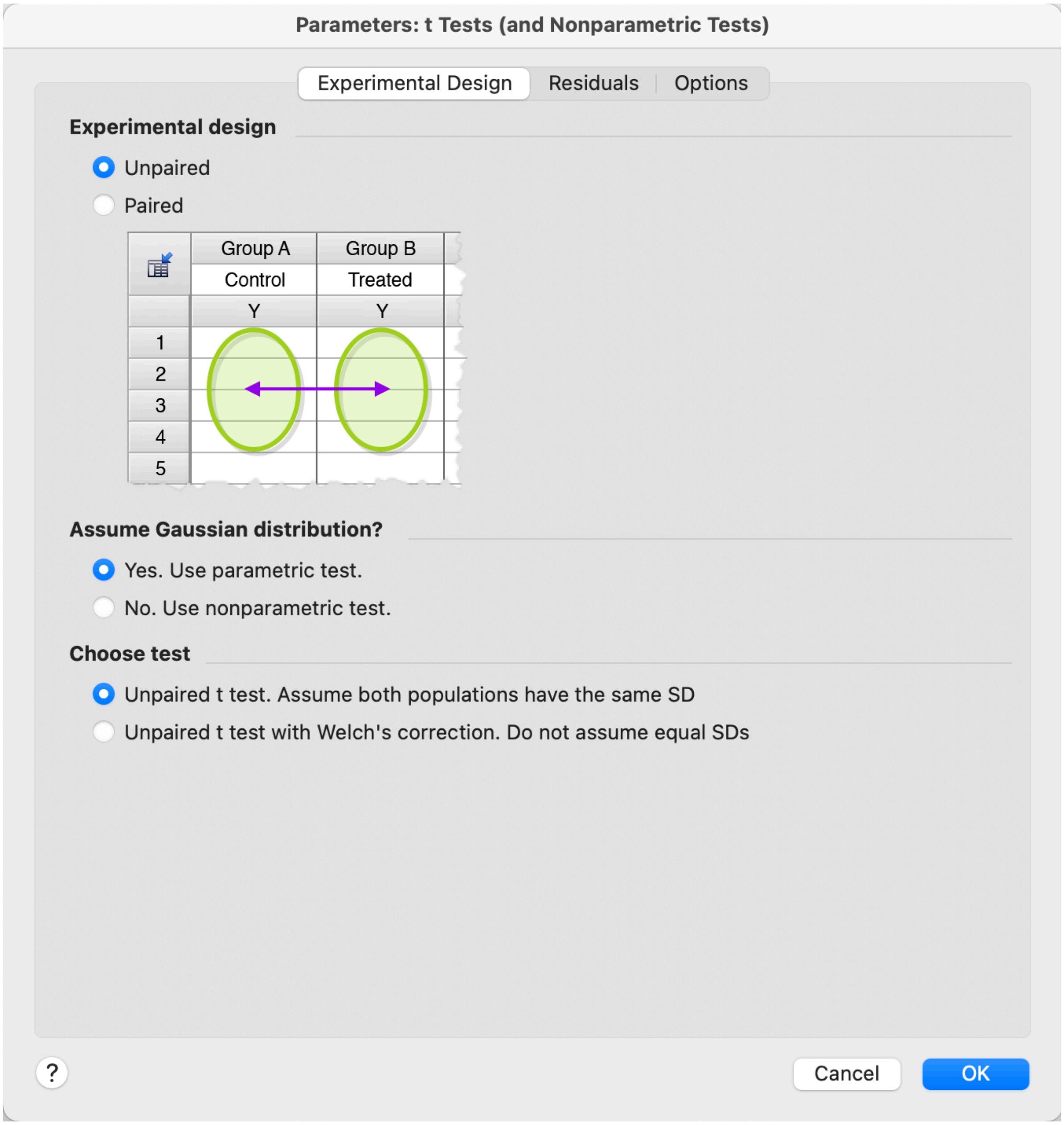
Effect Details

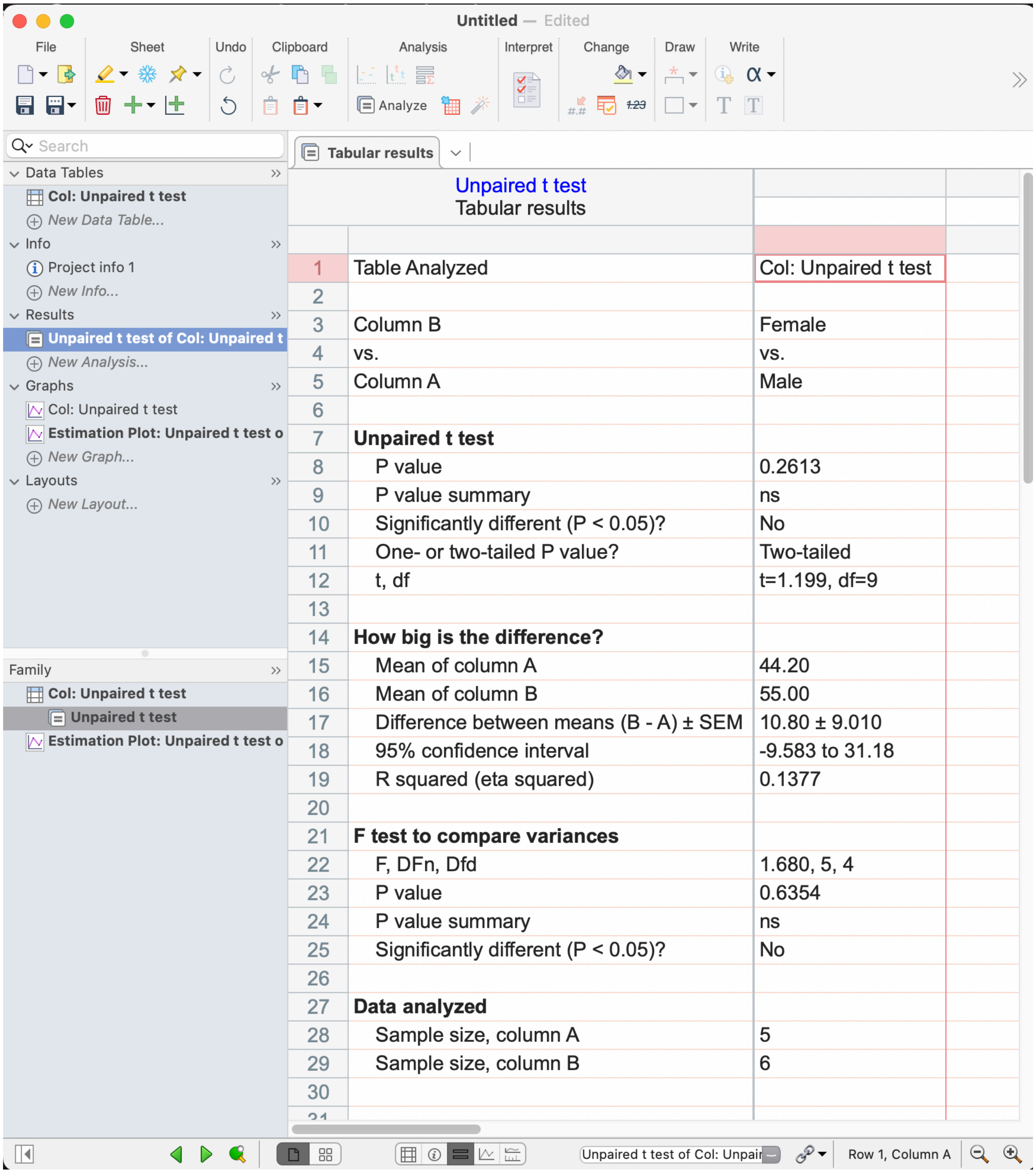
(The discrepancy in coefficients is in how SPSS and JMP handle binary categorical variables. The model fits are identical.)

Graphpad Prism

Graphpad Prism is very similar to JMP in that is a entirely GUI-based interaction that offers a limited subset of analyses. It is very popular amongst users with small data and little statistical experience. It operates similarly to JMP. One quirk is that it often (though not always) wants data stored in non-rectangular fashion, in a form that would be incompatible with lots of other software.




Parameters: t Tests (and Nonparametric Tests)
Experimental Design
Residuals
Options
Experimental design
Unpaired
Paired
Group A
Group B
Control
Treated
Y
Y
1
2
3
4
5
Assume Gaussian distribution?
Yes. Use parametric test.
No. Use nonparametric test.
Choose test
Unpaired t test. Assume both populations have the same SD
Unpaired t test with Welch's correction. Do not assume equal SDs
?
Cancel
OK



Unpaired t test
Tabular results

1	Table Analyzed	Col: Unpaired t test
2		
3	Column B	Female
4	vs.	vs.
5	Column A	Male
6		
7	**Unpaired t test**	
8	P value	0.2613
9	P value summary	ns
10	Significantly different (P < 0.05)?	No
11	One- or two-tailed P value?	Two-tailed
12	t, df	t=1.199, df=9
13		
14	**How big is the difference?**	
15	Mean of column A	44.20
16	Mean of column B	55.00
17	Difference between means (B - A) ± SEM	10.80 ± 9.010
18	95% confidence interval	-9.583 to 31.18
19	R squared (eta squared)	0.1377
20		
21	**F test to compare variances**	
22	F, DFn, Dfd	1.680, 5, 4
23	P value	0.6354
24	P value summary	ns
25	Significantly different (P < 0.05)?	No
26		
27	**Data analyzed**	
28	Sample size, column A	5
29	Sample size, column B	6
30		

Numerical Analysis Software

MATLAB

MATLAB is one of several programming languages with a focus on numerical analysis. There's also Octave which is mostly an open-source implementation of MATLAB.) MATLAB primarily comes into use for most statisticians due to it's efficient and powerful matrix support. This example comes from the MATLAB help center:

```
>> A = [1 2 0; 2 5 -1; 4 10 -1]
>> A
A = 3×3

     1     2     0
     2     5    -1
     4    10    -1
>> B = A'
>> C = A * B
>> C
C = 3×3

     5    12    24
    12    30    59
    24    59   117
>> % Let's solve Ax = b
>> b = [1;3;5]
>> x = A\b
>> x
x = 3×1

     1
     0
    -1
>> eig(A)
ans = 3×1

    3.7321
    0.2679
    1.0000
>> svd(A)
ans = 3×1
```

```
   12.3171
    0.5149
    0.1577
```

MATLAB also supports a limited set of statistical models. From the `fitlm` documentation:

```
>> load carsmall
>> tbl = table(Weight,Acceleration,Model_Year,MPG,'VariableNames',{'Weight','Acceleration'
>> lm = fitlm(tbl,'MPG~Weight+Acceleration')
lm =
Linear regression model:
    MPG ~ 1 + Weight + Acceleration

Estimated Coefficients:
                     Estimate         SE         tStat       pValue
                    __________    __________    _______    __________

    (Intercept)         45.155        3.4659     13.028    1.6266e-22
    Weight          -0.0082475    0.00059836    -13.783    5.3165e-24
    Acceleration       0.19694       0.14743     1.3359       0.18493


Number of observations: 94, Error degrees of freedom: 91
Root Mean Squared Error: 4.12
R-squared: 0.743,  Adjusted R-Squared: 0.738
F-statistic vs. constant model: 132, p-value = 1.38e-27
>> tbl.Model_Year = categorical(tbl.Model_Year)
>> lm = fitlm(tbl,'MPG~Weight+Model_Year')
lm =
Linear regression model:
    MPG ~ 1 + Weight + Model_Year

Estimated Coefficients:
                     Estimate         SE         tStat       pValue
                    __________    __________    _______    __________

    (Intercept)          40.11        1.5418     26.016    1.2024e-43
    Weight          -0.0066475    0.00042802    -15.531    3.3639e-27
    Model_Year_76       1.9291       0.74761     2.5804      0.011488
    Model_Year_82       7.9093       0.84975     9.3078    7.8681e-15
```

```
Number of observations: 94, Error degrees of freedom: 90
Root Mean Squared Error: 2.92
R-squared: 0.873,  Adjusted R-Squared: 0.868
F-statistic vs. constant model: 206, p-value = 3.83e-40
```

Others

The two big other numerical analysis software are

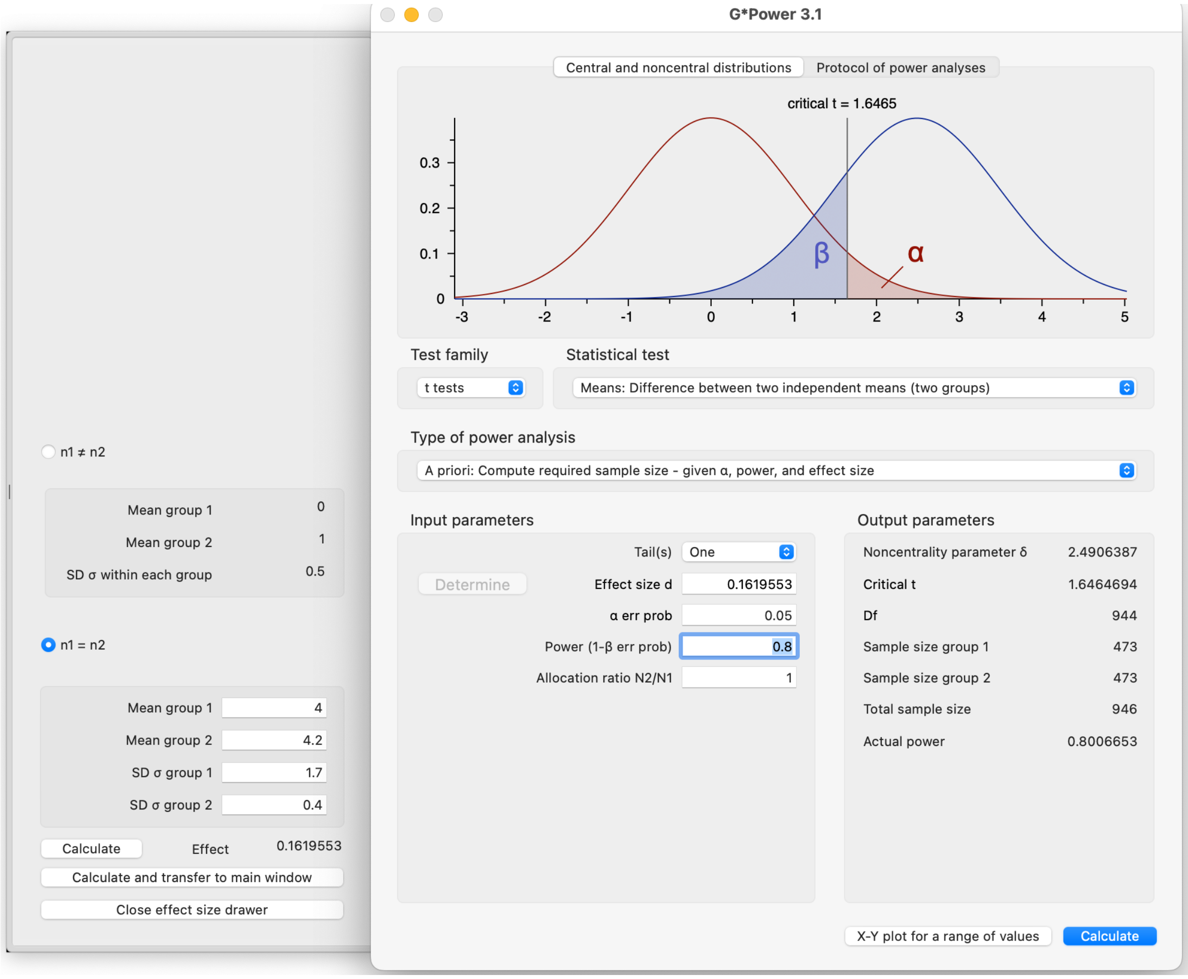
- Maple
- Wolfram Mathematica

Miscellaneous

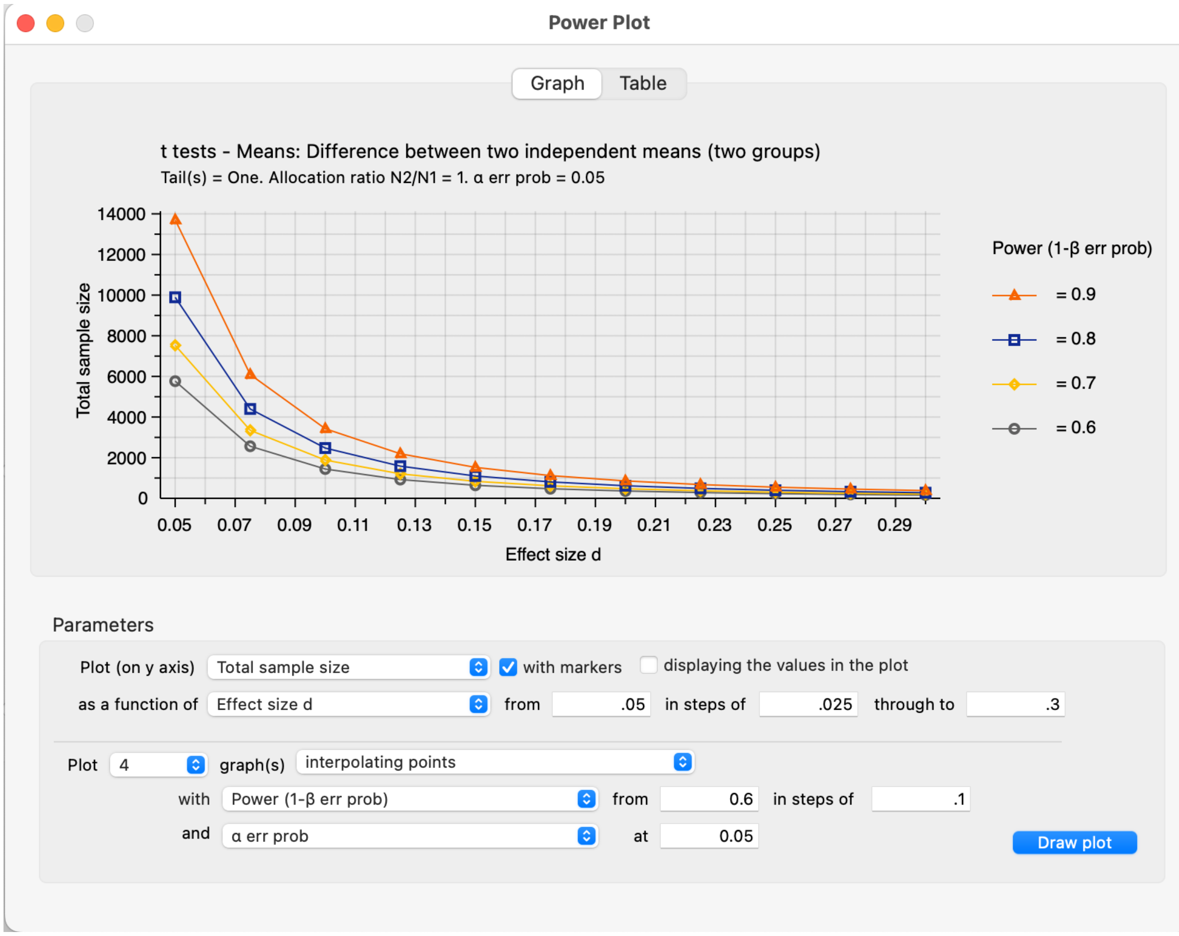
G*Power

G*Power is open-source software used in power analysis/sample size calculations. While most software has built-in power calculations, a lot of analysts prefer a custom-built solution like G*Power. As with any power analysis, obtaining a useful result from G*Power requires assumed values of all parameters of the model (primarily means and covariance matrices), as well as an understanding of the results are only as good as the guesses for the parameters.

Here's an example of a two-sample t-test. Note that standardized effect sizes (in this case d) can be manually input, or calculated in the side drawer.

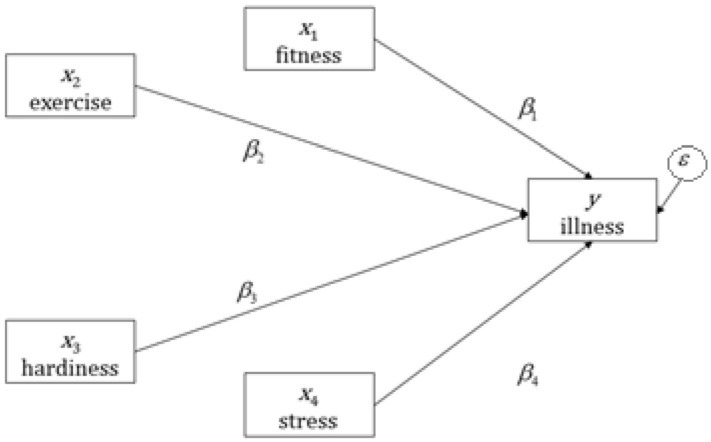


G*Power can also generate plots showing how power concerns change as other parameters change.



Mplus

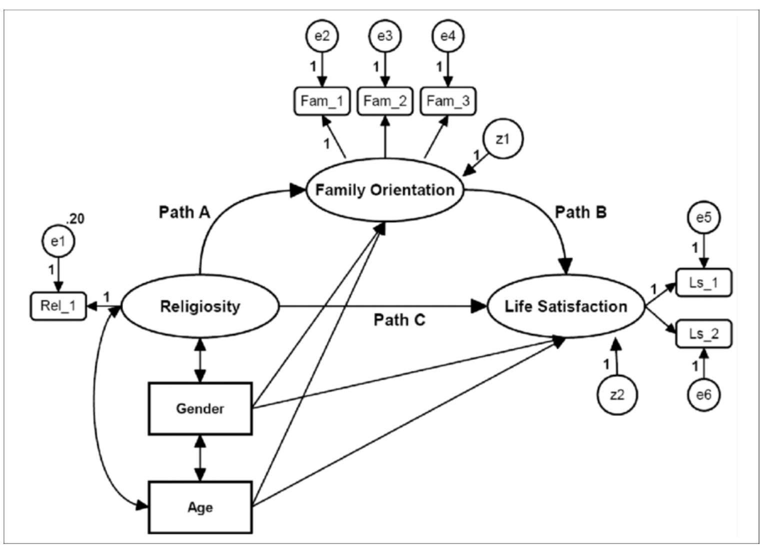
Mplus is *extremely* powerful software for fitting path analysis models, also known as structural equation models (SEM). These are models which can be represented via direct acyclic graphs (DAGs). For example, linear regression can be represented with this:



$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$$

Multiple regression

In this example there is a single outcome, *illness*. However, more complex models can be represented in a DAG:



While these models can be fit in other software (R's lavaan package, Stata's `sem` command, Amos for SPSS), Mplus is incredibly powerful and can fit complex models that the other software cannot handle.

Unfortunately, the code to fit such models is complex and finicky. For example, here is a relatively simple SEM:

```
Mplus VERSION 6
MUTHEN & MUTHEN
04/25/2010  10:58 PM

INPUT INSTRUCTIONS

  TITLE:  cont3

          Classic structural equation model with multiple indicators
          used in a study of the stability of alienation.

        Source:
          Wheaton, B., Muthen, B., Alwin, D., & Summers, G. (1977).
          Assessing the reliability and stability in panel models.
          In D.R. Heise (ed), Sociological Methodology 1977.
          San Francisco:  Jossey-Bass.

  DATA:  FILE IS wheacov.dat;
         TYPE IS COVARIANCE;
         NOBS ARE 932;

  VARIABLE:  NAMES ARE anomia67 power67 anomia71 power71 educ sei;

             USEVAR =  anomia67 power67 anomia71 power71 educ sei;

  MODEL:

  !        first the measurement model part using the keyword BY:

          ses BY educ sei;
          alien67 BY anomia67 power67;
          alien71 BY anomia71 power71;

  !        next the structural model part using the keyword ON:

          alien71 ON alien67 ses;
          alien67 ON ses;

  !        and then adding correlated residuals over time using
  !        the keyword WITH:
```

```
        anomia67 WITH anomia71;
        power67 WITH power71;

OUTPUT:

        sampstat tech1 standardized modindices(0);
```

The output from this model is very large, see this example for the full output. Some of this is skippable (e.g. the description and Source) but most of it is required precisely.