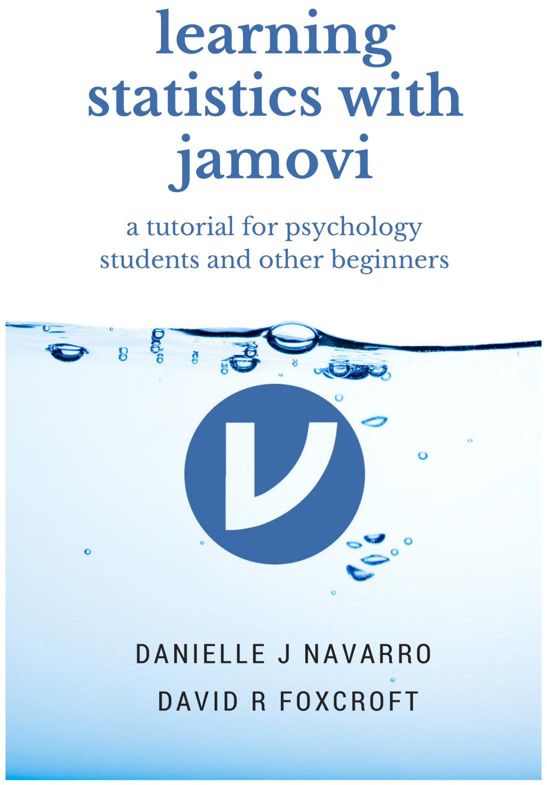

learning statistics with jamovi

a tutorial for psychology students and other beginners

DANIELLE J NAVARRO

DAVID R FOXCROFT

learning statistics with jamovi

Danielle J. Navarro & David R. Foxcroft

23 June 2022 (version 0.75)

Contents

learning statistics with jamovi covers the contents of an introductory statistics class, as typically taught to undergraduate psychology, health or social science students. The book covers how to get started in jamovi as well as giving an introduction to data manipulation. From a statistical perspective, the book discusses descriptive statistics and graphing first, followed by chapters on probability theory, sampling and estimation, and null hypothesis testing. After introducing the theory, the book covers the analysis of contingency tables, correlation, *t*-tests, regression, ANOVA and factor analysis. Bayesian statistics are touched on at the end of the book.

This book is an adaptation of DJ Navarro (2018). Learning statistics with R: A tutorial for psychology students and other beginners. (Version 0.6). https://learningstatisticswithr.com/.

Citation: Navarro DJ and Foxcroft DR (2022). learning statistics with jamovi: a tutorial for psychology students and other beginners. (Version 0.75). https://dx.doi.org/10.24384/hgc3-7p15

Chapter 1

Pragmatic matters

The garden of life never seems to confine itself to the plots philosophers have laid out for its convenience. Maybe a few more tractors would do the trick.
– Roger Zelazny[1]

This is a somewhat strange chapter, even by my standards. My goal in this chapter is to talk a bit more honestly about the realities of working with data than you'll see anywhere else in the book. The problem with real world data sets is that they are *messy*. Very often the data file that you start out with doesn't have the variables stored in the right format for the analysis you want to do. Sometimes there might be a lot of missing values in your data set. Sometimes you only want to analyse a subset of the data. Et cetera. In other words, there's a lot of **data manipulation** that you need to do just to get the variables in your data set into the format that you need it. The purpose of this chapter is to provide a basic introduction to these pragmatic topics. Although the chapter is motivated by the kinds of practical issues that arise when manipulating real data, I'll stick with the practice that I've adopted through most of the book and rely on very small, toy data sets that illustrate the underlying issue. Because this chapter is essentially a collection of techniques and doesn't tell a single coherent story, it may be useful to start with a list of topics:

- Tabulating and cross-tabulating data
- Logical expressions in jamovi
- Transforming and recoding a variable
- A few more mathematical functions and operations
- Extracting a subset of the data

As you can see, the list of topics that the chapter covers is pretty broad, and there's a lot of content there. Even though this is one of the longest and hardest

[1]The quote comes from Home is the Hangman, published in 1975.

chapters in the book, I'm really only scratching the surface of several fairly different and important topics. My advice, as usual, is to read through the chapter once and try to follow as much of it as you can. Don't worry too much if you can't grasp it all at once, especially the later sections. The rest of the book is only lightly reliant on this chapter so you can get away with just understanding the basics. However, what you'll probably find is that later on you'll need to flick back to this chapter in order to understand some of the concepts that I refer to here.

1.1 Tabulating and cross-tabulating data

A very common task when analysing data is the construction of frequency tables, or crosstabulation of one variable against another. These tasks can be achieved in jamovi and I'll show you how in this section.

1.1.1 Creating tables for single variables

Let's start with a simple example. As the father of a small child I naturally spend a lot of time watching TV shows like In the *Night Garden*. In the nightgarden.csv file, I've transcribed a short section of the dialogue. The file contains two variables of interest, speaker and utterance. Open up this data set in jamovi and take a look at the data in the 'spreadsheet' view. You will see that the data looks something like this:

'speaker' variable: upsy-daisy upsy-daisy upsy-daisy upsy-daisy tombliboo tombliboo makka-pakka makka-pakka makka-pakka makka-pakka 'utterance' variable: pip pip onk onk ee oo pip pip onk onk

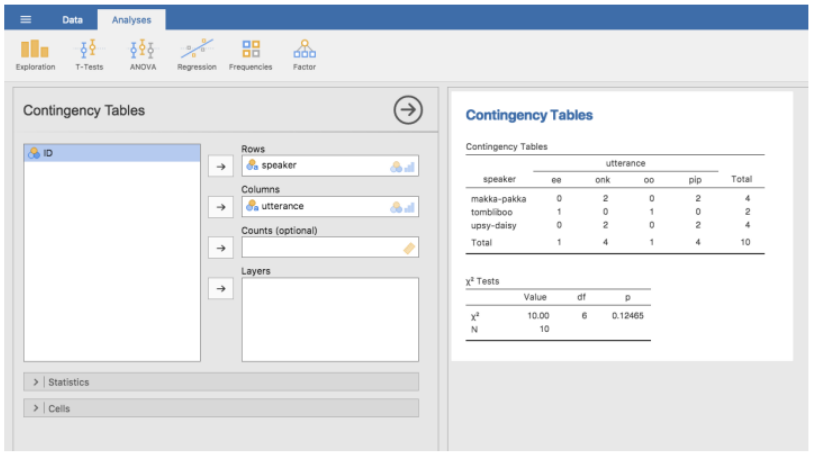
Looking at this it becomes very clear what happened to my sanity! With these as my data, one task I might find myself needing to do is construct a frequency count of the number of words each character speaks during the show. The jamovi 'Descriptives' screen has a check box called 'Frequency tables' which does just this, see Table **??**.

levels	**Counts**	**percent of Total**	**Cumulative percent**
makka-pakka	4	40percent	40percent
tombliboo	2	20percent	60percent
upsy-daisy	4	40percent	100percent

Table 1.1: Frequency table for the speaker variable

The output here tells us on the first line that what we're looking at is a tabulation of the speaker variable. In the 'Levels' column it lists all the different speakers that exist in the data, and in the 'Counts' column it tells you how many times that speaker appears in the data. In other words, it's a frequency table.

In jamovi, the 'Frequency tables' check box will only produce a table for single variables. For a table of two variables, for example combining speaker and utterance so that we can see how many times each speaker said a particular utterance, we need a cross-tabulation or contingency table. In jamovi you can do this by selecting the 'Frequencies' - 'Contingency Tables' - 'Independent Samples' analysis, and moving the speaker variable into the 'Rows' box, and the utterances variable into the 'Columns' box. You then should have a contingency table like the one shown in Figure **??**.



Contingency Tables

Contingency Tables

	utterance				
speaker	ee	onk	oo	pip	Total
makka-pakka	0	2	0	2	4
tombliboo	1	0	1	0	2
upsy-daisy	0	2	0	2	4
Total	1	4	1	4	10

χ^2 Tests

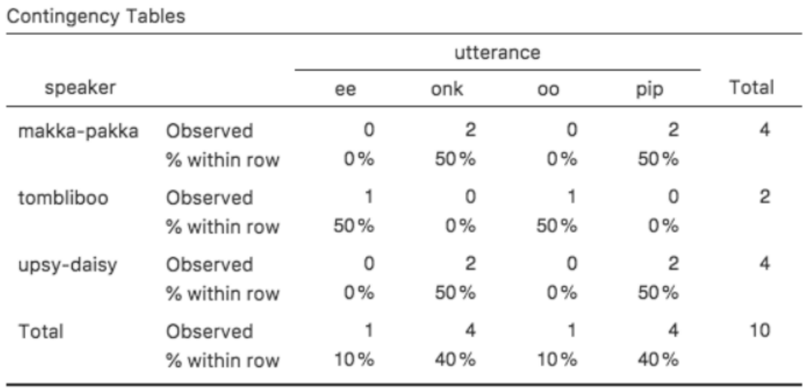
	Value	df	p
χ^2	10.00	6	0.12465
N	10		

Figure 1.1: Contingency table for the speaker and utterances variables

Don't worry about the "χ^2 Tests" table that is produced. We are going to cover this later on in the chapter on [Categorical data analysis]. When interpreting the contingency table remember that these are counts, so the fact that the first row and second column of numbers corresponds to a value of 2 indicates that Makka-Pakka (row 1) says "onk" (column 2) twice in this data set.

1.1.2 Adding percentages to a contingency table

The contingency table shown in Figure **??** shows a table of raw frequencies. That is, a count of the total number of cases for different combinations of levels of the specified variables. However, often you want your data to be organised in terms of percentages as well as counts. You can find the check boxes for different percentages under the 'Cells' option in the 'Contingency Tables' window. First, click on the 'Row' check box and the Contingency Table in the output window will change to the one in Figure **??**.

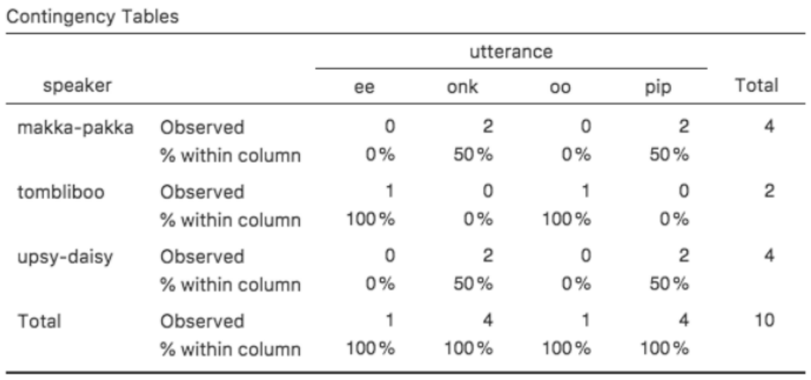


Contingency Tables

speaker		utterance				Total
		ee	onk	oo	pip	
makka-pakka	Observed	0	2	0	2	4
	% within row	0%	50%	0%	50%	
tombliboo	Observed	1	0	1	0	2
	% within row	50%	0%	50%	0%	
upsy-daisy	Observed	0	2	0	2	4
	% within row	0%	50%	0%	50%	
Total	Observed	1	4	1	4	10
	% within row	10%	40%	10%	40%	

Figure 1.2: Contingency table for the speaker and utterances variables, with row percentages

What we're looking at here is the percentage of utterances made by each character. In other words, 50% of Makka-Pakka's utterances are "pip", and the other 50% are "onk". Let's contrast this with the table we get when we calculate column percentages (uncheck 'Row' and check 'Column' in the Cells options window), see Figure **??**. In this version, what we're seeing is the percentage of characters associated with each utterance. For instance, whenever the utterance "ee" is made (in this data set), 100% of the time it's a Tombliboo saying it.



Contingency Tables

speaker		utterance				Total
		ee	onk	oo	pip	
makka-pakka	Observed	0	2	0	2	4
	% within column	0%	50%	0%	50%	
tombliboo	Observed	1	0	1	0	2
	% within column	100%	0%	100%	0%	
upsy-daisy	Observed	0	2	0	2	4
	% within column	0%	50%	0%	50%	
Total	Observed	1	4	1	4	10
	% within column	100%	100%	100%	100%	

Figure 1.3: Contingency table for the speaker and utterances variables, with column percentages

1.2 Logical expressions in jamovi

A key concept that a lot of data transformations in jamovi rely on is the idea of a **logical value**. A logical value is an assertion about whether something is true or false. This is implemented in jamovi in a pretty straightforward way. There are two logical values, namely TRUE and FALSE. Despite the simplicity, logical values are very useful things. Let's see how they work.

1.2.1 Assessing mathematical truths

In George Orwell's classic book 1984 one of the slogans used by the totalitarian Party was "two plus two equals five". The idea being that the political domination of human freedom becomes complete when it is possible to subvert even the most basic of truths. It's a terrifying thought, especially when the protagonist Winston Smith finally breaks down under torture and agrees to the proposition. "Man is infinitely malleable", the book says. I'm pretty sure that this isn't true of humans[2] and it's definitely not true of jamovi. jamovi is not infinitely malleable, it has rather firm opinions on the topic of what is and isn't true, at least as regards basic mathematics. If I ask it to calculate $2 + 2$[3], it always gives the same answer, and it's not bloody 5!

Of course, so far jamovi is just doing the calculations. I haven't asked it to explicitly assert that $2 + 2 = 4$ is a true statement. If I want jamovi to make an explicit judgement, I can use a command like this: $2 + 2 == 4$

What I've done here is use the **equality operator**, ==, to force jamovi to make a "true or false" judgement.[4] Okay, let's see what jamovi thinks of the Party slogan, so type this into the compute new variable 'formula' box:

$$2 + 2 == 5$$

And what do you get? It should be a whole set of 'false' values in the spreadsheet column for your newly computed variable. Booyah! Freedom and ponies for all! Or something like that. Anyway, it was worth having a look at what happens if I try to force jamovi to believe that two plus two is five by making a statement like $2 + 2 = 5$. I know that if I do this in another program, say R, then it throws up an error message. But wait, if you do this in jamovi you get a whole set of 'false' values. So what is going on? Well, it seems that jamovi is being

[2] I offer up my teenage attempts to be "cool" as evidence that some things just can't be done.

[3] You can do this in the Compute new variable screen, though just calculating 2 + 2 for every cell of a new variable is not very useful!

[4] Note that this is a very different operator to the equals operator =. A common typo that people make when trying to write logical commands in jamovi (or other languages, since the "= versus ==" distinction is important in many computer and statistical programs) is to accidentally type = when you really mean ==. Be especially cautious with this, I've been programming in various languages since I was a teenager and I still screw this up a lot. Hmm. I think I see why I wasn't cool as a teenager. And why I'm still not cool.

pretty smart and realises that you are testing whether it is TRUE or FALSE that $2 + 2 = 5$, regardless of whether you use the correct **equality operator**, $==$, or the equals sign "$=$".

Anyway, it was worth having a look at what happens if I try to force jamovi to believe that two plus two is five by making a statement like $2 + 2 = 5$. I know that if I do this in another program, say R, then it throws up an error message. But wait, if you do this in jamovi you get a whole set of 'false' values. So what is going on? Well, it seems that jamovi is being pretty smart and realises that you are testing whether it is TRUE or FALSE that $2 + 2 = 5$, regardless of whether you use the correct **equality operator**, $==$, or the equals sign "$=$".

1.2.2 Logical operations

So now we've seen logical operations at work. But so far we've only seen the simplest possible example. You probably won't be surprised to discover that we can combine logical operations with other operations and functions in a more complicated way, like this: $3 \times 3 + 4 \times 4 == 5 \times 5$ or this $SQRT(25) == 5$

Not only that, but as Table **??** illustrates, there are several other logical operators that you can use corresponding to some basic mathematical concepts. Hopefully these are all pretty self-explanatory. For example, the **less than** operator $<$ checks to see if the number on the left is less than the number on the right. If it's less, then jamovi returns an answer of TRUE, but if the two numbers are equal, or if the one on the right is larger, then jamovi returns an answer of FALSE.

In contrast, the **less than or equal to** operator $<=$ will do exactly what it says. It returns a value of TRUE if the number of the left hand side is less than or equal to the number on the right hand side. At this point I hope it's pretty obvious what the **greater than** operator $<$ and the **greater than or equal to** operator $<=$ do!

Next on the list of logical operators is the **not equal to** operator != which, as with all the others, does what it says it does. It returns a value of TRUE when things on either side are not identical to each other. Therefore, since $2 + 2$ isn't equal to 5, we would get 'true' as the value for our newly computed variable. Try it and see:

$$2 + 2 \ != 5$$

We're not quite done yet. There are three more logical operations that are worth knowing about, listed in Table **??**. These are the **not** operator !, the **and** operator and, and the **or** operator or. Like the other logical operators, their behaviour is more or less exactly what you'd expect given their names. For instance, if I ask you to assess the claim that "either $2 + 2 = 4$ or $2 + 2 = 5$"

you'd say that it's true. Since it's an "either-or" statement, all we need is for one of the two parts to be true. That's what the or operator does:[5]

operation	**operator**	**example input**	**answer**
less than	$<$	$2 < 3$	TRUE
less than or equal to	$<$	$2 < = 2$	TRUE
greater than	$>$	$2 > 3$	FALSE
greater than or equal to	$> =$	$2 > = 2$	TRUE
equal to	$= =$	$2 = = 3$	FALSE
not equal to	!=	2 != 3	TRUE

Table 1.2: Some logical operators

operation	**operator**	**example input**	**answer**
not	NOT	NOT(1==1)	FALSE
or	or	(1==1) or (2==3)	TRUE
and	and	(1==1) and (2==3)	FALSE

Table 1.3: Some more logical operators

$$(2 + 2 == 4) \text{ or } (2 + 2 == 5)$$

On the other hand, if I ask you to assess the claim that "both $2 + 2 = 4$ and $2 + 2 = 5$" you'd say that it's false. Since this is an and statement we need both parts to be true. And that's what the and operator does:

[5]Now, here's a quirk in jamovi. When you have simple logical expressions like the ones we have already met, e.g. 2 + 2 == 5 then jamovi neatly states 'false' (or 'true') in the corresponding spreadsheet column. Underneath the hood, jamovi stores 'false' as 0 and 'true' as 1. When we have more complex logical expressions, such as (2+2 == 4) or (2+2 == 5), then jamovi just displays either 0 or 1, depending whether the logical expression is evaluated as false, or true.

$$(2+2==4) \text{ and } (2+2==5)$$

Finally, there's the not operator, which is simple but annoying to describe in English. If I ask you to assess my claim that "it is not true that $2+2=5$" then you would say that my claim is true, because actually my claim is that "$2+2=5$ is false". And I'm right. If we write this in jamovi we use this:

$$NOT(2+2==5)$$

In other words, since $2+2==5$ is a FALSE statement, it must be the case that $NOT(2+2==5)$ is a TRUE one. Essentially, what we've really done is claim that "not false" is the same thing as "true". Obviously, this isn't really quite right in real life. But jamovi lives in a much more black or white world. For jamovi everything is either true or false. No shades of grey are allowed.

Of course, in our $2+2=5$ example, we didn't really need to use the "not" operator NOT and the "equals to" operator $==$ as two separate operators. We could have just used the "not equals to" operator $!=$ like this:

$$2+2\ !=5$$

1.2.3 Applying logical operation to text

I also want to briefly point out that you can apply these logical operators to text as well as to logical data. It's just that we need to be a bit more careful in understanding how jamovi interprets the different operations. In this section I'll talk about how the equal to operator == applies to text, since this is the most important one. Obviously, the not equal to operator != gives the exact opposite answers to == so I'm implicitly talking about that one too, but I won't give specific commands showing the use of $!=$.

Okay, let's see how it works. In one sense, it's very simple. For instance, I can ask jamovi if the word "cat" is the same as the word "dog", like this:

"cat" == "dog" That's pretty obvious, and it's good to know that even jamovi can figure that out. Similarly, jamovi does recognise that a "cat" is a "cat": "cat" == "cat" Again, that's exactly what we'd expect. However, what you need to keep in mind is that jamovi is not at all tolerant when it comes to grammar and spacing. If two strings differ in any way whatsoever, jamovi will say that they're not equal to each other, as with the following: " cat" == "cat" "cat" == "CAT" "cat" == "c a t"

You can also use other logical operators too. For instance jamovi also allows you to use the $>$ and $>$ operators to determine which of two text 'strings' comes first, alphabetically speaking. Sort of. Actually, it's a bit more complicated than that, but let's start with a simple example: