Well, learning R had at least two major benefits for me: 1.) I could improve my statistical knowledge a lot, simply by using formulas, asking why certain R commands do not automatically give the same results like SPSS, reading R resources and papers etc. and 2.) the possibilities of data visualization are way better in R than in SPSS (though SPSS can do well as well…). Of course, there are even many more reasons to use R.

Still, one thing I often miss in R is a beautiful output of simple statistics or maybe even advanced statistics. Not always as plot or graph, but neither as "cryptic" console output. I'd like to have a simple table view, just like the SPSS output window (though the SPSS output is not "beautiful"). That's why I started writing functions that put the results of certain statistics in HTML tables. These tables can be saved to disk or, even better for quick inspection, shown in a web browser or viewer pane (like in RStudio viewer pane).

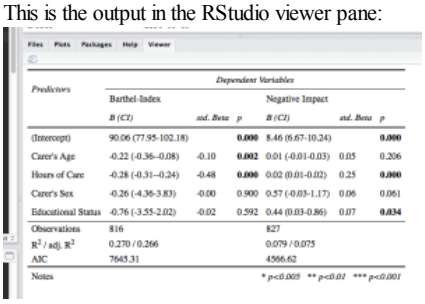
All of the following functions are available in my sjPlot-package on CRAN.

(Generalized) Linear Models

The first two functions, which I already published last year, can be used to display (generalized) linear models and have been described here. Yet I want to give another short example for quickly viewing at linear models:

```
require(sjPlot) # load package
# Fit "dummy" models. Note that both models
# and only differ in their dependent variabl
data(efc)
# fit first model
fit1 <- lm(barthtot ~ c160age + c12hour + c1
# fit second model
fit2 <- lm(neg_c_7 ~ c160age + c12hour + c16
# Print HTML-table to viewer pane
sjt.lm(fit1, fit2,
       labelDependentVariables=c("Barthel-In
       labelPredictors=c("Carer's Age", "Hou
       showStdBeta=TRUE, pvaluesAsNumbers=TR
```

This is the output in the RStudio viewer pane:



Files Plots Packages Help Viewer

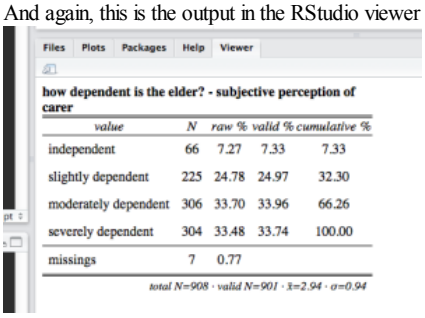
Predictors	Dependent Variables					
	Barthel-Index			Negative Impact		
	B (CI)	std. Beta	p	B (CI)	std. Beta	p
(Intercept)	90.06 (77.95-102.18)		**0.000**	8.46 (6.67-10.24)		**0.000**
Carer's Age	-0.22 (-0.36--0.08)	-0.10	**0.002**	0.01 (-0.01-0.03)	0.05	0.206
Hours of Care	-0.28 (-0.31--0.24)	-0.48	**0.000**	0.02 (0.01-0.02)	0.25	**0.000**
Carer's Sex	-0.26 (-4.36-3.83)	-0.00	0.900	0.57 (-0.03-1.17)	0.06	0.061
Educational Status	-0.76 (-3.55-2.02)	-0.02	0.592	0.44 (0.03-0.86)	0.07	**0.034**
Observations	816			827		
R^2 / adj. R^2	0.270 / 0.266			0.079 / 0.075		
AIC	7645.31			4566.62		
Notes				* p<0.005 ** p<0.01 *** p<0.001		

Frequency Tables

Another (new) function is `sjt.frq` which prints frequency tables (the next example uses value and variable labels, but the simplest function call is just `sjt.frq(variable)`).

```
require(sjPlot) # load package
# load sample data
data(efc)
# retrieve value and variable labels
variables <- sji.getVariableLabels(efc)
values <- sji.getValueLabels(efc)
# simple frequency table
sjt.frq(efc$e42dep,
        variableLabels=variables['e42dep'],
        valueLabels=values[['e42dep']])
```

And again, this is the output in the RStudio viewer pane:



Files Plots Packages Help Viewer

how dependent is the elder? - subjective perception of carer

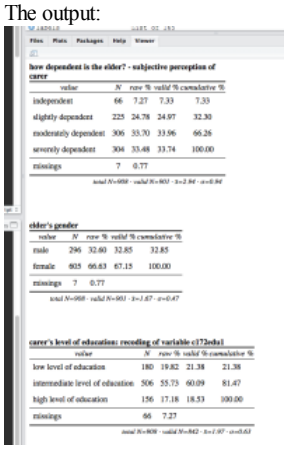
value	N	raw %	valid %	cumulative %
independent	66	7.27	7.33	7.33
slightly dependent	225	24.78	24.97	32.30
moderately dependent	306	33.70	33.96	66.26
severely dependent	304	33.48	33.74	100.00
missings	7	0.77		

total N=908 · valid N=901 · x̄=2.94 · σ=0.94

You can print frequency tables of several variables at once:

```
sjt.frq(as.data.frame(cbind(efc$e42dep, efc$e
        variableLabels=list(variables['e42dep
        valueLabels=list(values[['e42dep']],
```

The output:



how dependent is the elder? - subjective perception of carer

value	N	raw %	valid %	cumulative %
independent	66	7.27	7.33	7.33
slightly dependent	225	24.78	24.97	32.30
moderately dependent	306	33.70	33.96	66.26
severely dependent	304	33.48	33.74	100.00
missings	7	0.77		

total N=908 · valid N=901 · x̄=2.94 · σ=0.94

elder's gender

value	N	raw %	valid %	cumulative %
male	296	32.60	32.85	32.85
female	605	66.63	67.15	100.00
missings	7	0.77		

total N=908 · valid N=901 · x̄=1.67 · σ=0.47

carer's level of education: recoding of variable c172edu1

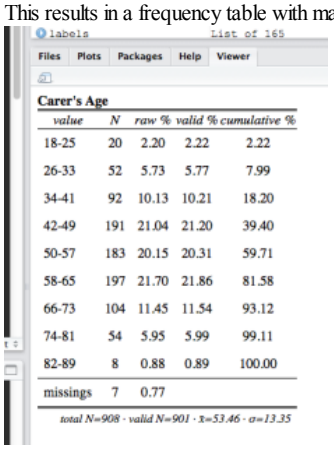
value	N	raw %	valid %	cumulative %
low level of education	180	19.82	21.38	21.38
intermediate level of education	506	55.73	60.09	81.47
high level of education	156	17.18	18.53	100.00
missings	66	7.27		

total N=908 · valid N=842 · x̄=1.97 · σ=0.63

When applying SPSS frequency tables, especially for variable with many unique values (e.g. age or income), this often results in very long, unreadable tables. The `sjt.frq` function, however, can automatically group variables with many unique values:

```
sjt.frq(efc$c160age,
        variableLabels=list("Carer's Age"),
        autoGroupAt=10)
```

This results in a frequency table with max. 10 groups:



Carer's Age

value	N	raw %	valid %	cumulative %
18-25	20	2.20	2.22	2.22
26-33	52	5.73	5.77	7.99
34-41	92	10.13	10.21	18.20
42-49	191	21.04	21.20	39.40
50-57	183	20.15	20.31	59.71
58-65	197	21.70	21.86	81.58
66-73	104	11.45	11.54	93.12
74-81	54	5.95	5.99	99.11
82-89	8	0.88	0.89	100.00
missings	7	0.77		

total N=908 · valid N=901 · x̄=53.46 · σ=13.35

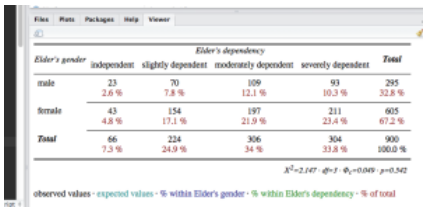
You can also specify whether the row with median value and both upper and lower quartile are highlighted. Furthermore, the complete HTML-code is returned for further use, separated into style sheet and table content. In case you have multiple frequency tables, the function returns a list with HTML-tables.

Contingency Tables

The second new function in the sjPlot-package (while I'm writing this posting, source code and windows binaries of version 1.1 are available, Mac binaries will follow soon…) is `sjt.xtab` for printing contingency tables.

The simple function call prints observed values and cell percentages:

```
# prepare sample data set
data(efc)
efc.labels <- sji.getValueLabels(efc)
sjt.xtab(efc$e16sex, efc$e42dep,
         variableLabels=c("Elder's gender", "
         valueLabels=list(efc.labels[['e16sex
```



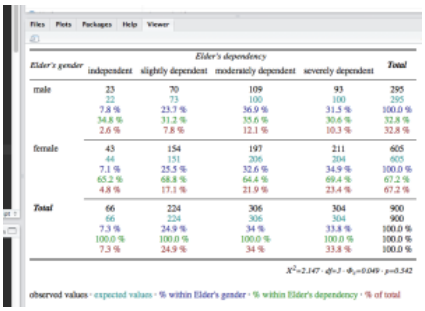
Elder's gender	Elder's dependency: independent	slightly dependent	moderately dependent	severely dependent	Total
male	23 2.6 %	70 7.8 %	109 12.1 %	93 10.3 %	295 32.8 %
female	43 4.8 %	154 17.1 %	197 21.9 %	211 23.4 %	605 67.2 %
Total	66 7.3 %	224 24.9 %	306 34 %	304 33.8 %	900 100.0 %

X^2=2.147 · df=3 · Φ_C=0.049 · p=0.542

observed values · expected values · % within Elder's gender · % within Elder's dependency · % of total

Observed values are obligatory, while cell, row and column percentages as well as expected values can be added via parameters. An example with all possible information:

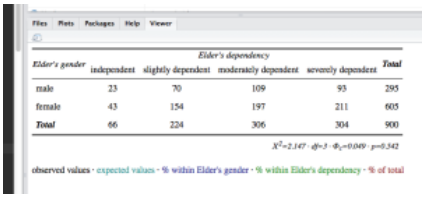
```
sjt.xtab(efc$e16sex, efc$e42dep,
         variableLabels=c("Elder's gender", "
         valueLabels=list(efc.labels[['e16sex
         showRowPerc=TRUE, showColPerc=TRUE,
```



Elder's gender	Elder's dependency: independent	slightly dependent	moderately dependent	severely dependent	Total
male	23	70	109	93	295
	22	73	100	100	295
	7.8 %	23.7 %	36.9 %	31.5 %	100.0 %
	34.8 %	31.2 %	35.6 %	30.6 %	32.8 %
	2.6 %	7.8 %	12.1 %	10.3 %	32.8 %
female	43	154	197	211	605
	44	151	206	204	605
	7.1 %	25.5 %	32.6 %	34.9 %	100.0 %
	65.2 %	68.8 %	64.4 %	69.4 %	67.2 %
	4.8 %	17.1 %	21.9 %	23.4 %	67.2 %
Total	66	224	306	304	900
	66	224	306	304	900
	7.3 %	24.9 %	34 %	33.8 %	100.0 %
	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
	7.3 %	24.9 %	34 %	33.8 %	100.0 %

And a simple one, w/o horizontal lines:

```
sjt.xtab(efc$e16sex, efc$e42dep,
         variableLabels=c("Elder's gender", "
         valueLabels=list(efc.labels[['e16sex
         showCellPerc=FALSE, showHorizontalLi
```



Elder's gender	Elder's dependency: independent	slightly dependent	moderately dependent	severely dependent	Total
male	23	70	109	93	295
female	43	154	197	211	605
Total	66	224	306	304	900

All colors can be specified via parameters, as well as the constant string values. See `?sjt.frq` resp. `?sjt.xtab` for detailed information.

If you have more ideas on which "quick" statistics are suitable for printing the results in the viewer pane, let me know. I will try to include them into my package…

Tagged: data visualization, R, rstats, SPSS, Statistik

Comments are closed.

Top 3 Posts from the past 2 days

- ggplot2: Cheatsheet for Visualizing Distributions
- Books and lessons about ggplot2
- R and (Software) Relatives

Search & Hit Enter

Top 9 articles of the week

1. R skills attract the highest salaries
2. ggplot2: Cheatsheet for Visualizing Distributions
3. Installing R packages
4. Using apply, sapply, lapply in R
5. Books and lessons about ggplot2
6. Reproducible research, training wheels, and knitr
7. Basics of Histograms
8. Box-plot with R – Tutorial
9. A million ways to connect R and Excel

Sponsors