The Journal of Open Source Software

rempsyc: Convenience functions for psychology

Rémi Thériault [1]

1 Departement of Psychology, Université du Québec à Montréal, Québec, Canada

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

{rempsyc} is an R package of convenience functions that make the analysis-to-publication workflow faster, easier, and less error-prone. It affords easily customizable APA plots (via {ggplot2}) and nice APA tables exportable to Word (via {flextable}). It makes it easy to run statistical tests, check assumptions, and automatize various tasks. It is a package mostly geared at researchers in the psychological sciences but people from all fields can benefit from it.

Statement of need

There are many reasons to use R (R Core Team 2022) for analyzing and reporting data from research studies. R is more compatible with the ideals of open science (Quintana 2020). In contrast to commercial software: (a) it is free to use; (b) it makes it easy to share a fully comprehensive analysis script; (c) it is transparent as anyone can look at the formulas or algorithms used in a given package; (d) the community can quickly contribute new packages based on current needs; (e) it generates better-looking figures; and (f) it helps reduce copy-paste errors so common in psychology. The latter point is a substantial one because according to some estimates, up to 50% of articles in psychology have at least one statistical error (Nuijten et al. 2016).

However, R has a major downside for R novices: its steep learning curve due to its programmatic interface, in contrast to perhaps more user-friendly point-and-click software. Of course, this flexibility is also a strength, as the R community can, and increasingly does, mobilize to produce packages that make using R as easy as possible (e.g., the *easystats* ecosystem Lüdecke et al. [2019] 2023). The {rempsyc} package contributes to this momentum by providing convenience functions that remove as much friction as possible between your script and your manuscript (in particular, if you are using Microsoft Word).

There are mainly three things that go into a manuscript: text, tables, and figures. {rempsyc} does not generate publication-ready text summarizing analyses; for this, see the {report} package (Makowski et al. [2021] 2023). Instead, {rempsyc} focuses on the production of publication-ready tables and figures. Below, I go over a few quick examples of those.

Examples Features

Publication-Ready Tables

Formatting your table properly in R is already a time-consuming task, but fortunately several packages take care of the formatting within R [e.g., the {broom} or {report} packages, Robinson, Hayes, and Couch (2022); Makowski et al. ([2021] 2023); and there are several others]. Exporting these formatted tables to Microsoft Word remains a challenge however.

Some packages do export to Word (e.g., Stanley and Spence 2018), but their formatting is often rigid especially when using analyzes that are not supported by default.

{rempsyc} solves this problem by allowing maximum flexibility: you manually create the data frame exactly the way you want, and then only use the magical function, `nice_table()`, on the resulting data frame. `nice_table()` works on any data frame, even non-statistical ones. For example, it will work on the `mtcars` data set.

```
## Suggested APA citation: Thériault, R. (2022). rempsyc: Convenience functions for psyc
## (R package version 0.1.1) [Computer software]. https://rempsyc.remi-theriault.com

library(rempsyc)

nice_table(
  mtcars[1:3, ],
  title = c("Table 1", "Motor Trend Car Road Tests"),
  note = c("The data was extracted from the 1974 Motor Trend US magazine.",
           "* p < .05, ** p < .01, *** p < .001"))
```

Table 1

Motor Trend Car Road Tests

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.00	6.00	160.00	110.00	3.90	2.62	16.46	0.00	1.00	4.00	4.00
21.00	6.00	160.00	110.00	3.90	2.88	17.02	0.00	1.00	4.00	4.00
22.80	4.00	108.00	93.00	3.85	2.32	18.61	1.00	1.00	4.00	1.00

Note. The data was extracted from the 1974 Motor Trend US magazine.

* p < .05, ** p < .01, *** p < .001

One of its main benefit however is the automatic formatting of statistical symbols and its integration with other packages. We can for example create a {broom} table and then apply `nice_table()` on it.

```
library(broom)
model <- lm(mpg ~ cyl + wt * hp, mtcars)
(stats.table <- tidy(model, conf.int = TRUE))

## # A tibble: 5 × 7
##   term        estimate std.error statistic  p.value  conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>     <dbl>     <dbl>
## 1 (Intercept)  49.5      3.66        13.5   1.58e-13  42.0       57.0
## 2 cyl          -0.365    0.509       -0.718 4.79e- 1  -1.41       0.678
## 3 wt           -7.63     1.52        -5.01  2.93e- 5 -10.7       -4.51
## 4 hp           -0.108    0.0298      -3.64  1.14e- 3  -0.169     -0.0473
## 5 wt:hp         0.0258   0.00799      3.23  3.22e- 3   0.00944    0.0422
```

```
nice_table(stats.table, broom = "lm")
```

Term	*b*	*SE*	*t*	*p*	95% CI
(Intercept)	49.49	3.66	13.51	< .001	[41.97, 57.01]
cyl	-0.37	0.51	-0.72	.479	[-1.41, 0.68]
wt	-7.63	1.52	-5.01	< .001	[-10.75, -4.51]
hp	-0.11	0.03	-3.64	.001	[-0.17, -0.05]
wt × hp	0.03	0.01	3.23	.003	[0.01, 0.04]

We can do the same with a {report} table.

```
library(report)
model <- lm(mpg ~ cyl + wt * hp, mtcars)
(stats.table <- as.data.frame(report(model)))

## Parameter    | Coefficient |          95% CI | t(27) |      p | Std. Coef. | Std. Coef
## ------------------------------------------------------------------------------------------
## (Intercept) |       49.49 | [ 41.97, 57.01] | 13.51 | < .001 |      -
0.18 |    [-0.36, -0.01] |
## cyl         |       -0.37 | [ -1.41,  0.68] | -0.72 | 0.479  |      -
0.11 |    [-0.42,  0.20] |
## wt          |       -7.63 | [-10.75, -4.51] | -5.01 | < .001 |      -
0.62 |    [-0.85, -0.40] |
## hp          |       -0.11 | [ -0.17, -0.05] | -3.64 | 0.001  |      -
0.29 |    [-0.53, -0.04] |
## wt × hp     |        0.03 | [  0.01,  0.04] |  3.23 | 0.003  |       0.29 |    [ 0.11
##             |             |                 |       |        |            |
## AIC         |             |                 |       |        |            |
## AICc        |             |                 |       |        |            |
## BIC         |             |                 |       |        |            |
## R2          |             |                 |       |        |            |
## R2 (adj.)   |             |                 |       |        |            |
## Sigma       |             |                 |       |        |            |

nice_table(stats.table)
```

Parameter	Fit	*b*	95% CI (*b*)	*t*	*df*	*p*	β	95% CI (β)
(Intercept)		49.49	[41.97, 57.01]	13.51	27	< .001	-0.18	[-0.36, -0.01]
cyl		-0.37	[-1.41, 0.68]	-0.72	27	.479	-0.11	[-0.42, 0.20]
wt		-7.63	[-10.75, -4.51]	-5.01	27	< .001	-0.62	[-0.85, -0.40]
hp		-0.11	[-0.17, -0.05]	-3.64	27	.001	-0.29	[-0.53, -0.04]
wt × hp		0.03	[0.01, 0.04]	3.23	27	.003	0.29	[0.11, 0.47]
AIC	147.01							
AICc	150.37							
BIC	155.80							
R2	0.89							
R2 (adj.)	0.87							
Sigma	2.17							

The {report} package provides quite comprehensive tables, so one may request an abbreviated table with the short argument.

```
nice_table(stats.table, short = TRUE)
```

Parameter	*b*	*t*	*df*	*p*	β	95% CI (β)
(Intercept)	49.49	13.51	27	< .001	-0.18	[-0.36, -0.01]
cyl	-0.37	-0.72	27	.479	-0.11	[-0.42, 0.20]
wt	-7.63	-5.01	27	< .001	-0.62	[-0.85, -0.40]
hp	-0.11	-3.64	27	.001	-0.29	[-0.53, -0.04]
wt × hp	0.03	3.23	27	.003	0.29	[0.11, 0.47]

For convenience, it is also possible to highlight significant results for better visual discrimination, using the `highlight` argument[1].

```
my_table <- nice_table(stats.table, short = TRUE, highlight = 0.001)
my_table
```

Parameter	*b*	*t*	*df*	*p*	β	95% CI (β)
(Intercept)	**49.49**	**13.51**	**27**	**< .001**	**-0.18**	**[-0.36, -0.01]**
cyl	-0.37	-0.72	27	.479	-0.11	[-0.42, 0.20]
wt	**-7.63**	**-5.01**	**27**	**< .001**	**-0.62**	**[-0.85, -0.40]**
hp	-0.11	-3.64	27	.001	-0.29	[-0.53, -0.04]
wt × hp	0.03	3.23	27	.003	0.29	[0.11, 0.47]

One can easily save the resulting table to Word with `flextable::save_as_docx()`, specifying the object name and desired path.

```
flextable::save_as_docx(my_table, path = "nice_tablehere.docx")
```

Additionally, tables created with `nice_table()` are {flextable} objects (Gohel and Skintzos

2022), and can be modified as such[2].

Formattting Results of Analyses

{rempsyc} also provides its own set of functions to prepare statistical tables before they can be fed to `nice_table()` and saved to Word.

t tests

```
stats.table <- nice_t_test(
  data = mtcars,
  response = c("mpg", "disp", "drat"),
  group = "am",
  warning = FALSE)
stats.table

##   Dependent Variable         t       df            p         d   CI_lower
## 1                mpg -3.767123 18.33225 1.373638e-03 -1.477947 -2.2659731
## 2               disp  4.197727 29.25845 2.300413e-04  1.445221  0.6417834
## 3               drat -5.646088 27.19780 5.266742e-06 -2.003084 -2.8592770
##     CI_upper
## 1 -0.6705686
## 2  2.2295592
## 3 -1.1245498

nice_table(stats.table)
```

Dependent Variable	*t*	*df*	*p*	*d*	95% CI
mpg	-3.77	18.33	.001	-1.48	[-2.27, -0.67]
disp	4.20	29.26	< .001	1.45	[0.64, 2.23]
drat	-5.65	27.20	< .001	-2.00	[-2.86, -1.12]

Contrasts

```
nice_contrasts(data = mtcars,
               response = c("mpg", "disp"),
               group = "cyl",
               covariates = "hp") -> contrasts
contrasts

##   Dependent Variable Comparison df         t            p         d   CI_lower
## 1                mpg      4 - 8 28  3.663188 1.028617e-03  3.587739  2.6675232
## 2                mpg      6 - 8 28  1.290359 2.074806e-01  1.440495  0.8577536
## 3                mpg      4 - 6 28  3.640418 1.092089e-03  2.147244  1.3689365
## 4               disp      4 - 8 28 -6.040561 1.640986e-06 -4.803022 -
5.7699794
```

```
## 5            disp      6 - 8 28 -4.861413 4.051110e-05 -3.288726 -
4.2638686
## 6            disp      4 - 6 28 -2.703423 1.153440e-02 -1.514296 -
2.2759030
##      CI_upper
## 1  4.4207911
## 2  2.0045019
## 3  3.0621741
## 4 -3.8531752
## 5 -2.2649722
## 6 -0.8836528

nice_table(contrasts, highlight = .001)
```

Dependent Variable	Comparison	*df*	*t*	*p*	*d*	95% CI
mpg	4 - 8	28	3.66	.001	3.59	[2.67, 4.42]
	6 - 8	28	1.29	.207	1.44	[0.86, 2.00]
	4 - 6	28	3.64	.001	2.15	[1.37, 3.06]
disp	**4 - 8**	**28**	**-6.04**	**< .001**	**-4.80**	**[-5.77, -3.85]**
	6 - 8	**28**	**-4.86**	**< .001**	**-3.29**	**[-4.26, -2.26]**
	4 - 6	28	-2.70	.012	-1.51	[-2.28, -0.88]

Moderations

```
stats.table <- nice_mod(
  data = mtcars,
  response = "mpg",
  predictor = "gear",
  moderator = "wt")
stats.table

##   Dependent Variable Predictor df         b          t          p         sr2
## 1                mpg      gear 28  5.615951  1.9437108 0.06204275 0.028488305
## 2                mpg        wt 28  1.403861  0.4301493 0.67037970 0.001395217
## 3                mpg   gear:wt 28 -1.966931 -2.1551077 0.03989970 0.035022025
##       CI_lower   CI_upper
## 1 0.0000000000 0.08418650
## 2 0.0000000000 0.01331121
## 3 0.0003502202 0.09723370

nice_table(stats.table)
```

Dependent Variable	Predictor	*df*	*b*	*t*	*p*	sr^2	95% CI
mpg	gear	28	5.62	1.94	.062	.03	[0.00, 0.08]
	wt	28	1.40	0.43	.670	.00	[0.00, 0.01]
	gear × wt	28	-1.97	-2.16	.040	.04	[0.00, 0.10]

Regressions

```
model1 <- lm(mpg ~ cyl + wt * hp, mtcars)
model2 <- lm(qsec ~ disp + drat * carb, mtcars)
mods <- nice_lm(list(model1, model2))
mods

##   Model Number Dependent Variable Predictor df            b          t
## 1            1                mpg       cyl 27 -0.365239089 -0.7180977
## 2            1                mpg        wt 27 -7.627489287 -5.0146028
## 3            1                mpg        hp 27 -0.108394273 -3.6404181
## 4            1                mpg     wt:hp 27  0.025836594  3.2329593
## 5            2               qsec      disp 27 -0.006222635 -1.9746464
## 6            2               qsec      drat 27  0.227692395  0.1968842
## 7            2               qsec      carb 27  1.154106215  0.7179431
## 8            2               qsec drat:carb 27 -0.477539959 -1.0825727
##              p          sr2     CI_lower   CI_upper
## 1 4.788652e-01 0.0021596150 0.0000000000 0.01306786
## 2 2.928375e-05 0.1053130854 0.0089876445 0.20163853
## 3 1.136403e-03 0.0555024045 0.0005550240 0.11934768
## 4 3.221753e-03 0.0437733438 0.0004377334 0.09898662
## 5 5.861684e-02 0.0702566891 0.0000000000 0.19796621
## 6 8.453927e-01 0.0006984424 0.0000000000 0.01347203
## 7 4.789590e-01 0.0092872897 0.0000000000 0.05587351
## 8 2.885720e-01 0.0211165564 0.0000000000 0.09136014

nice_table(mods, highlight = TRUE)
```

Dependent Variable	Predictor	*df*	*b*	*t*	*p*	*sr*^{2}	95% CI
mpg	cyl	27	-0.37	-0.72	.479	.00	[0.00, 0.01]
	wt	**27**	**-7.63**	**-5.01**	**< .001**	**.11**	**[0.01, 0.20]**
	hp	**27**	**-0.11**	**-3.64**	**.001**	**.06**	**[0.00, 0.12]**
	wt × hp	**27**	**0.03**	**3.23**	**.003**	**.04**	**[0.00, 0.10]**
qsec	disp	27	-0.01	-1.97	.059	.07	[0.00, 0.20]
	drat	27	0.23	0.20	.845	.00	[0.00, 0.01]
	carb	27	1.15	0.72	.479	.01	[0.00, 0.06]
	drat × carb	27	-0.48	-1.08	.289	.02	[0.00, 0.09]

Simple Slopes

```
model1 <- lm(mpg ~ gear * wt, mtcars)
model2 <- lm(disp ~ gear * wt, mtcars)
my.models <- list(model1, model2)
simple.slopes <- nice_lm_slopes(my.models, predictor = "gear", moderator = "wt")
simple.slopes

##   Model Number Dependent Variable Predictor (+/-1 SD) df         b         t
## 1            1                mpg      gear (LOW-wt) 28  7.540509 2.0106560
## 2            1                mpg     gear (MEAN-wt) 28  5.615951 1.9437108
## 3            1                mpg     gear (HIGH-wt) 28  3.691393 1.7955678
## 4            2               disp      gear (LOW-wt) 28 50.510710 0.6654856
## 5            2               disp     gear (MEAN-wt) 28 35.797623 0.6121820
## 6            2               disp     gear (HIGH-wt) 28 21.084536 0.5067498
##            p         sr2 CI_lower   CI_upper
## 1 0.05408136 0.030484485        0 0.08823243
## 2 0.06204275 0.028488305        0 0.08418650
## 3 0.08336403 0.024311231        0 0.07551496
## 4 0.51118526 0.003234637        0 0.02113980
## 5 0.54535707 0.002737218        0 0.01919662
## 6 0.61629796 0.001875579        0 0.01548357

nice_table(simple.slopes)
```

Dependent Variable	Predictor (+/-1 *SD*)	*df*	*b*	*t*	*p*	sr^2	95% CI
mpg	gear (LOW-wt)	28	7.54	2.01	.054	.03	[0.00, 0.09]
	gear (MEAN-wt)	28	5.62	1.94	.062	.03	[0.00, 0.08]
	gear (HIGH-wt)	28	3.69	1.80	.083	.02	[0.00, 0.08]
disp	gear (LOW-wt)	28	50.51	0.67	.511	.00	[0.00, 0.02]
	gear (MEAN-wt)	28	35.80	0.61	.545	.00	[0.00, 0.02]
	gear (HIGH-wt)	28	21.08	0.51	.616	.00	[0.00, 0.02]

Correlation Matrix

It is also possible to export a coloured correlation matrix to Microsoft Excel. The `cormatrix_excel()` function has several benefits over conventional approaches. The base R `cor()` function for example does not use rounded values and the console is impractical for large matrices. One may manually round values and export it to a `.csv` file, which is an improvement but still unsatisfying.

The {apaTables} package (Stanley and Spence 2018) allows exporting the correlation matrix to Word in an APA format, and in many cases this is very satifying for APA requirements. Hovever, the Word format is not suitable for large matrices, as it will often spread beyond the document's margin limits.

Another approach is to export to an image, like {correlation} package does (Makowski et al. 2020). For very small matrices, this works extremely well, and the colour is an immense help to quickly identify which correlations are strong or weak, positive or negative. Again, however, this does not work so well for large matrices because labels might overlap or navigating the large figure becomes difficult.

When the goal is more exploratory, rather than reporting, and we have large matrices, it can be more useful to export it to Excel. In {rempsyc}, we combine the idea of using a coloured correlation matrix from the {correlation} package with the idea of exporting to Excel using {openxlsx2} (Barbone and Garbuszus 2023).

We also provide some quality of life-improvements, like freezing the first row and column so as to be able to easily see to which variables the correlations relate, regardless of how far or deep we are within the large correlation matrix.

The colour represents the strength of the correlation, whereas the stars represent how significant the *p* value is.[3] The exact *p* values are provided in a second tab for reference purposes, so all information is readily available in a convenient format.

```
cormatrix_excel(data = infert,
                filename = "cormatrix1",
                select = c("age", "parity", "induced", "case", "spontaneous",
                           "stratum", "pooled.stratum"))
```

	A	B	C	D	E	F	G	H	I
1	Paramete	age	parity	induced	case	spontane	stratum	pooled.stratum	
2	age	1.0	.08	-.10	.0	-.08	-.21 ***	-.17 *	
3	parity	.08	1.0	.45 ***	.01	.31 ***	-.31 ***	.12	
4	induced	-.10	.45 ***	1.0	.02	-.27 ***	-.10	.16 *	
5	case	.0	.01	.02	1.0	.36 ***	.0	.0	
6	spontane	-.08	.31 ***	-.27 ***	.36 ***	1.0	.06	.21 ***	
7	stratum	-.21 ***	-.31 ***	-.10	.0	.06	1.0	.75 ***	
8	pooled.st	-.17 *	.12	.16 *	.0	.21 ***	.75 ***	1.0	
9									

r_values p_values

	A	B	C	D	E	F	G	H	I
1	Paramete	age	parity	induced	case	spontane	stratum	pooled.stratum	
2	age	1.0	.08	-.10	.0	-.08	-.21 ***	-.17 *	
3	parity	.08	1.0	.45 ***	.01	.31 ***	-.31 ***	.12	
4	induced	-.10	.45 ***	1.0	.02	-.27 ***	-.10	.16 *	
5	case	.0	.01	.02	1.0	.36 ***	.0	.0	
6	spontane	-.08	.31 ***	-.27 ***	.36 ***	1.0	.06	.21 ***	
7	stratum	-.21 ***	-.31 ***	-.10	.0	.06	1.0	.75 ***	
8	pooled.st	-.17 *	.12	.16 *	.0	.21 ***	.75 ***	1.0	
9									

r_values p_values

Publication-Ready Figures

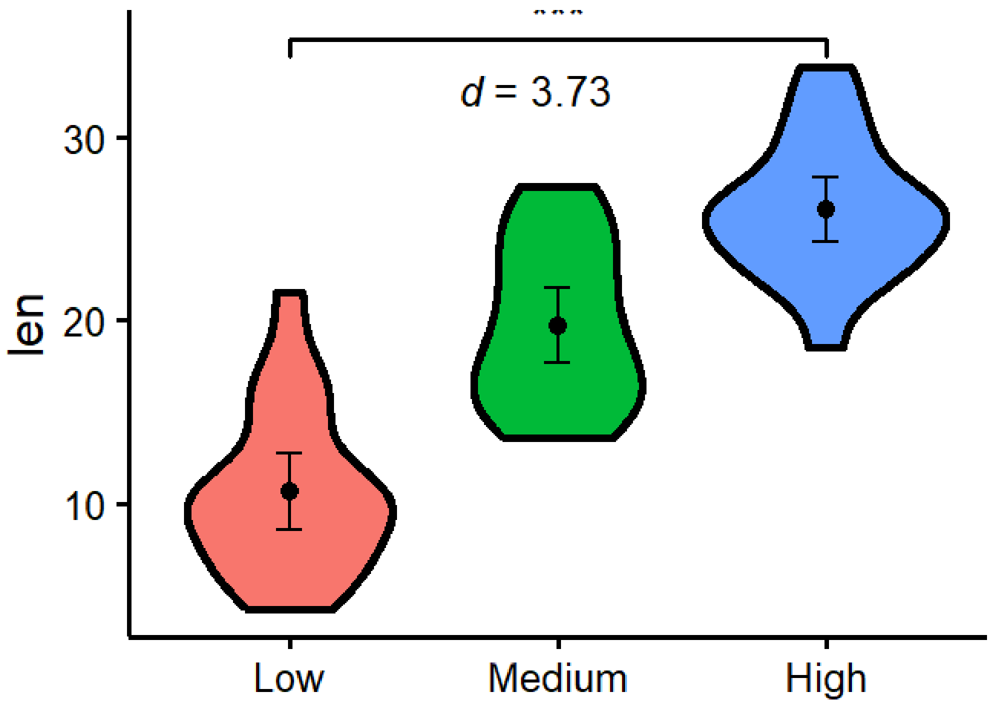
Preparing figures according to APA style, having them look good, and being able to save them in high-resolution with the proper ratios is often challenging. Working with {ggplot2} (Wickham 2016) provides tremendous flexibility, but an unintended consequence is that doing even trivial operations can at times be daunting.

This is why {rempsyc} prepares a few plot types for you, so they are ready to be saved to your preferred format (`.pdf`, `.tiff`, or `.png`).

Violin Plots

For an example of such use in publication, see Thériault et al. (2021).

```
nice_violin(data = ToothGrowth,
            group = "dose",
            response = "len",
            xlabels = c("Low", "Medium", "High"),
            comp1 = 1,
            comp2 = 3,
            has.d = TRUE,
            d.y = 30)
```



One can easily save the resulting figure with `ggplot2::ggsave()`, specifying the desired file name, extension, and resolution.

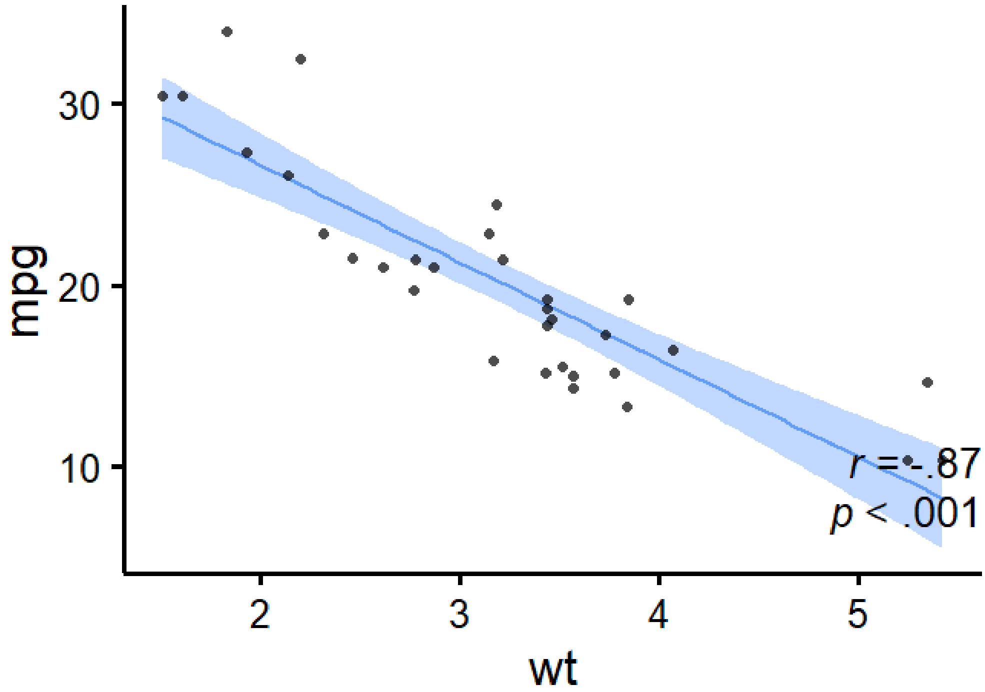
```
ggplot2::ggsave('nice_violinplothere.pdf', width = 7, height = 7,
                unit = 'in', dpi = 300)
```

Recommended dimensions for saving {rempsyc} figures is 7 inches wide and 7 inches high at 300 dpi, which makes sure that the resolution is high enough even if saving to non-vector graphics formats like `.png`. That said, scalable vector graphics formats like `.pdf` or `.eps` are still recommended for high-resolution submissions to scientific journals. Additionally, figures are {ggplot2} objects (Wickham 2016), and can be modified as such.

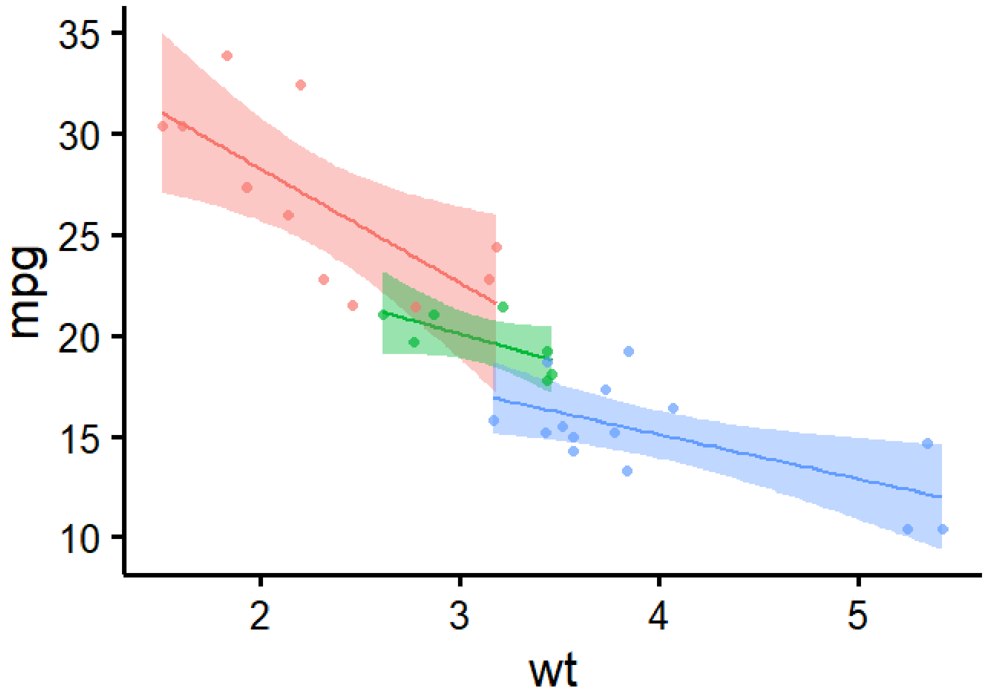
Scatter Plots

For an example of such use in publication, see Krol et al. (2020).

```
nice_scatter(data = mtcars,
             predictor = "wt",
             response = "mpg",
             has.confband = TRUE,
             has.r = TRUE,
             has.p = TRUE)
```



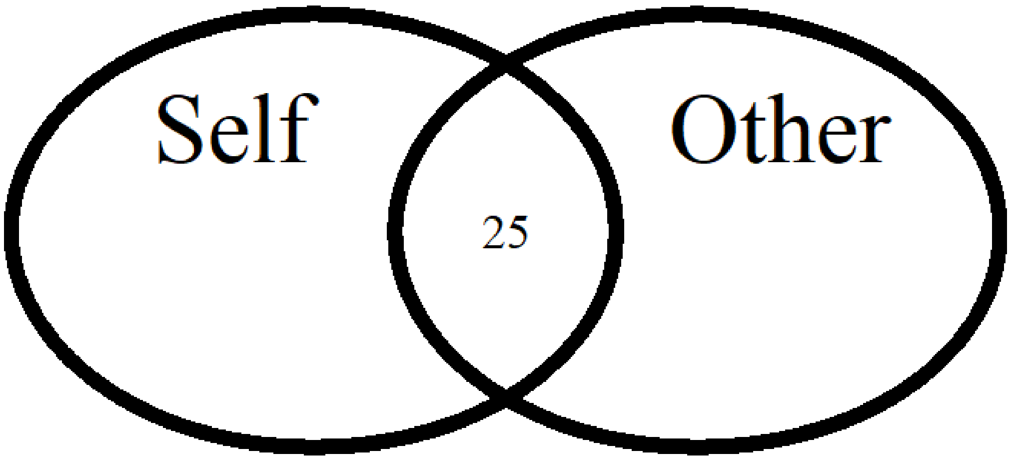

```
nice_scatter(data = mtcars,
             predictor = "wt",
             response = "mpg",
             group = "cyl",
             has.confband = TRUE)
```



Overlapping Circles

For psychologists using the Inclusion of Other in the the Self Scale (Aron, Aron, and Smollan 1992), it can be useful to interpolate the original discrete scores (1 to 7) into a group average representation of the conceptual self-other overlap. For an example of such use in publication, see Thériault et al. (2021).

```
overlap_circle(3.5)
```



Testing assumptions

When comes time to test assumptions of a linear model, the best option is the `check_model()` function from *easystats*' {performance} package, which allows direct visual evaluation of assumptions (Lüdecke et al. 2021). Indeed, visual assessment of diagnostic plots is recommended over statistical tests since they are overpowered in large samples and underpowered in small samples (Kozak and Piepho 2018).

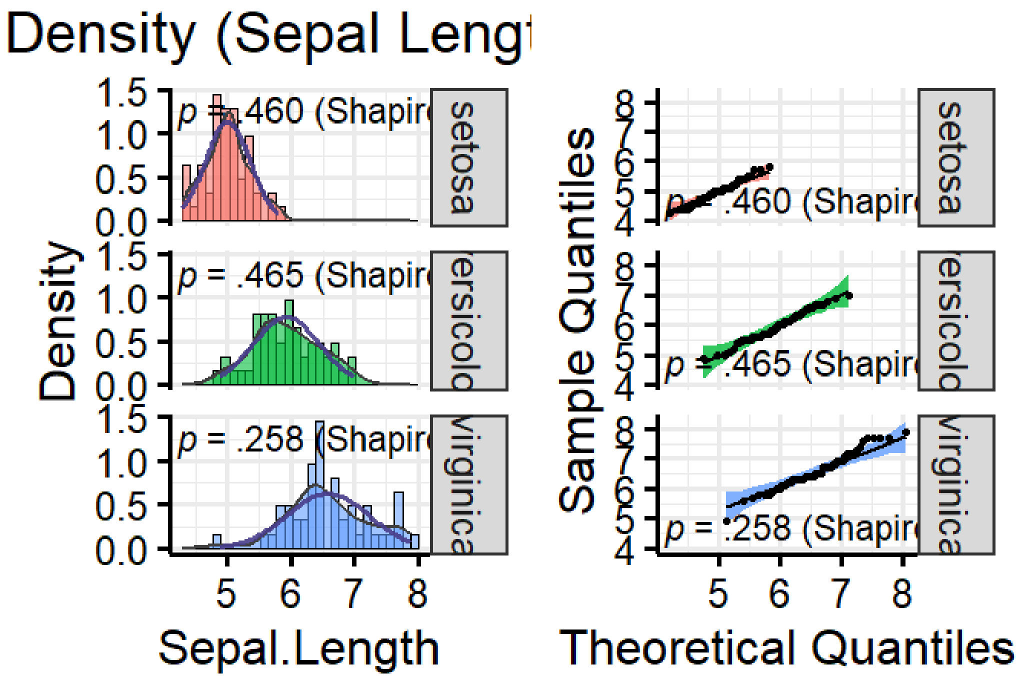
That said, if for whatever reason one wants to check objective asumption tests for a linear model, rempsyc makes this easy with the `nice_assumptions()` function, which provide p values for normality (Shapiro-Wilk), homoscedasticity (Breusch-Pagan) and autocorrelation of residuals (Durbin-Watson) in one call. .

Categorical Predictors

`nice_normality()` makes it easy to visually check normality in the case of categorical predictors (i.e., when using groups), through a combination of quantile-quantile plots, density plots, and histograms.

```
nice_normality(data = iris,
               variable = "Sepal.Length",
               group = "Species",
               shapiro = TRUE,
               histogram = TRUE,
```

```
                   title = "Density (Sepal Length)")
```

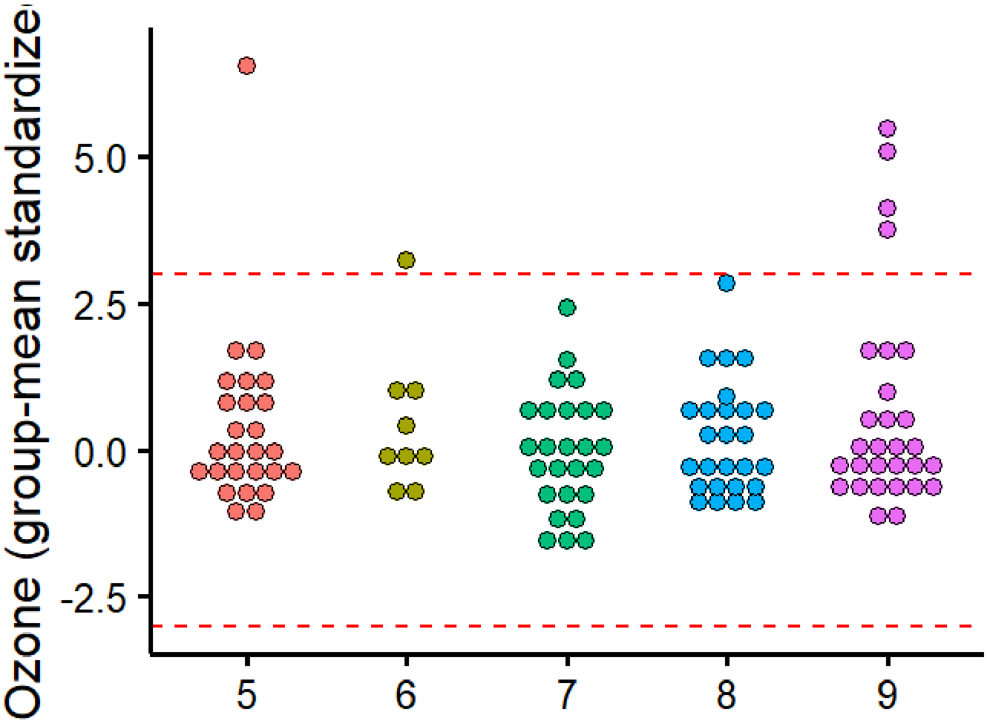


Similarly for univariate outliers using the median absolute deviation (MAD, Leys et al. 2013).

```
plot_outliers(
  airquality,
  group = "Month",
  response = "Ozone")

## Bin width defaults to 1/30 of the range of the data. Pick better value with
## `binwidth`.

## Warning: Removed 37 rows containing missing values (`stat_bindot()`).
```



Univariate outliers based on the MAD can also be simply requested with `find_mad()`.[4]

```
find_mad(airquality, names(airquality), criteria = 3)
```

```
## 8 outlier(s) based on 3 median absolute deviations for variable(s):
##  Ozone, Solar.R, Wind, Temp, Month, Day
##
## Outliers per variable:
##
## $Ozone
##   Row Ozone_mad
## 1  30  3.218284
## 2  62  3.989131
## 3  99  3.488081
## 4 101  3.025573
## 5 117  5.261028
## 6 121  3.333911
##
## $Wind
##   Row Wind_mad
## 1   9 3.049871
## 2  48 3.225825
```

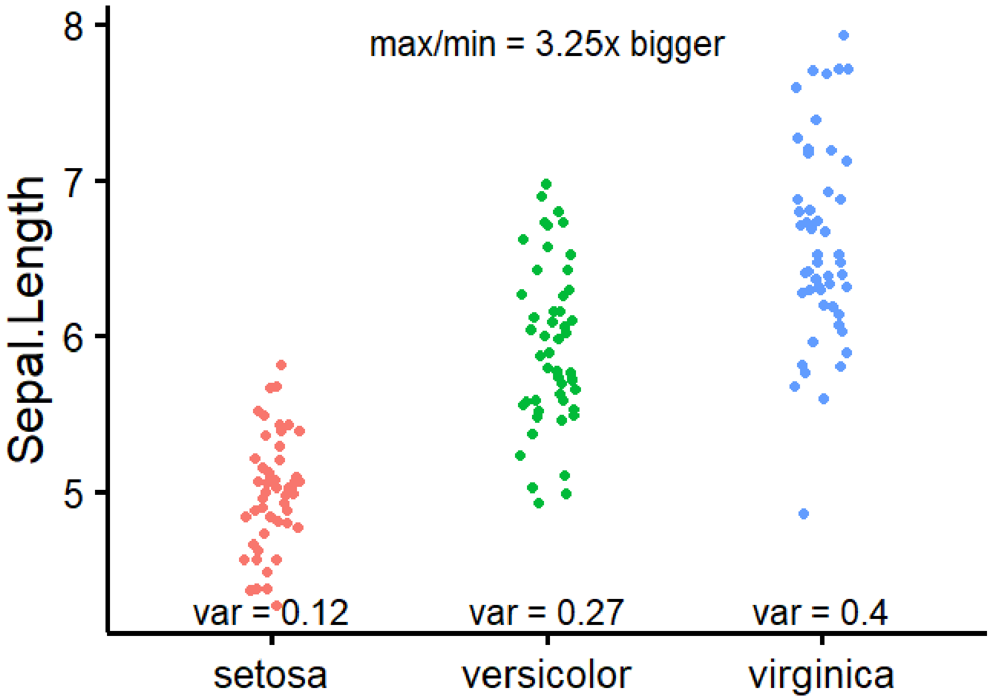
Homoscedasticity can also be checked numerically with `nice_var()` or visually with `nice_varplot()`.

```
nice_var(data = iris,
         variable = names(iris[1:4]),
         group = "Species")
```

```
##        Species Setosa Versicolor Virginica Variance.ratio Criteria
## 1 Sepal.Length  0.124      0.266     0.404            3.3        4
```

```
## 2  Sepal.Width  0.144      0.098     0.104          1.5        4
## 3 Petal.Length  0.030      0.221     0.305         10.2        4
## 4  Petal.Width  0.011      0.039     0.075          6.8        4
##   Heteroscedastic
## 1           FALSE
## 2           FALSE
## 3            TRUE
## 4            TRUE

nice_varplot(data = iris,
             variable = "Sepal.Length",
             group = "Species")
```



Utility functions

Finally, with the idea of making the analysis workflow easier in mind, {rempsyc} also has a few other utility functions. `nice_na()` allows reporting item-level missing values per scale, as well as participant's maximum number of missing items by scale, as per recommendations (Parent 2013).

`extract_duplicates()` creates a data frame of only observations with a duplicated ID or participant number, so they can be investigated more thoroughly. best_duplicate() allows to follow-up on this investigation and only keep the "best" duplicate, meaning those with the fewer number of missing values, and in case of ties, the first one.

`nice_reverse()` permits the automatic reverse-coding of scores so common for psychology questionnaires, provided the minimum and maximum score values are known.

There are other functions that the reader can explore at their leisure on the package official website. However, hopefully, this overview has given the reader a gentle introduction to this package.

Availability

The {rempsyc} package is available on CRAN, and can be installed using `install.packages("rempsyc")`. The full tutorial website can be accessed at: https://rempsyc.remi-theriault.com/.

Acknowledgements

I would like to thank Hugues Leduc, Jay Olson, Charles-Étienne Lavoie, and Björn Büdenbender for statistical or technical advice that helped inform some functions of this package and/or useful feedback on this manuscript. I would also like to acknowledge funding from the Social Sciences and Humanities Research Council of Canada.

References

Aron, Arthur, Elaine N Aron, and Danny Smollan. 1992. "Inclusion of Other in the Self Scale and the Structure of Interpersonal Closeness." *Journal of Personality and Social Psychology* 63 (4): 596. https://doi.org/10.1037/0022-3514.63.4.596.

Barbone, Jordan Mark, and Jan Marvin Garbuszus. 2023. *Openxlsx2: Read, Write and Edit 'Xlsx' Files*. https://github.com/JanMarvin/openxlsx2.

Gohel, David, and Panagiotis Skintzos. 2022. *Flextable: Functions for Tabular Reporting*. https://CRAN.R-project.org/package=flextable.

Kozak, Marcin, and H-P Piepho. 2018. "What's Normal Anyway? Residual Plots Are More Telling Than Significance Tests When Checking ANOVA Assumptions." *Journal of Agronomy and Crop Science* 204 (1): 86–98. https://doi.org/10.1111/jac.12220.

Krol, Sonia A, Rémi Thériault, Jay A Olson, Amir Raz, and Jennifer A Bartz. 2020. "Self-Concept Clarity and the Bodily Self: Malleability Across Modalities." *Personality and Social Psychology Bulletin* 46 (5): 808–20. https://doi.org/10.1177/0146167219879126.

Leys, Christophe, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. 2013. "Detecting Outliers: Do Not Use Standard Deviation Around the Mean, Use Absolute Deviation Around the Median." *Journal of Experimental Social Psychology* 49 (4): 764–66. https://doi.org/10.1016/j.jesp.2013.03.013.

Lüdecke, Daniel, Mattan S. Ben-Shachar, Indrajeet Patil, Philip Waggoner, and Dominique Makowski. 2021. "performance: An R Package for Assessment, Comparison and Testing of Statistical Models." *Journal of Open Source Software* 6 (60): 3139. https://doi.org/10.21105/joss.03139.

Lüdecke, Daniel, Dominique Makowski, Mattan S. Ben-Shachar, Indrajeet Patil, Brenton M. Wiernik, Etienne Bacher, and Rémi Thériault. (2019) 2023. *easystats: Streamline Model Interpretation, Visualization, and Reporting*. https://easystats.github.io/easystats/.

Makowski, Dominique, Mattan S. Ben-Shachar, Indrajeet Patil, and Daniel Lüdecke. 2020. "Methods and Algorithms for Correlation Analysis in r." *Journal of Open Source Software* 5 (51): 2306. https://doi.org/10.21105/joss.02306.

Makowski, Dominique, Daniel Lüdecke, Indrajeet Patil, Rémi Thériault, Mattan S. Ben-Shachar, and Brenton M. Wiernik. (2021) 2023. *report: Automated Reporting of Results and Statistical Models*. https://easystats.github.io/report/.

Nuijten, Michèle B, Chris HJ Hartgerink, Marcel ALM Van Assen, Sacha Epskamp, and Jelte M Wicherts. 2016. "The Prevalence of Statistical Reporting Errors in Psychology (1985–2013)." *Behavior Research Methods* 48: 1205–26. https://doi.org/doi.org/10.3758/s13428-015-0664-2.

Parent, Mike C. 2013. "Handling Item-Level Missing Data: Simpler Is Just as Good." *The Counseling Psychologist* 41 (4): 568–600. https://doi.org/10.1177%2F0011000012445176.

Quintana, D. S. 2020. *Five Things about Open and Reproducible Science That Every Early Career Researcher Should Know*. https://osf.io/2jt9u.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Robinson, David, Alex Hayes, and Simon Couch. 2022. *Broom: Convert Statistical Objects into Tidy Tibbles*. https://CRAN.R-project.org/package=broom.

Stanley, David J, and Jeffrey R Spence. 2018. "Reproducible Tables in Psychology Using the apaTables Package." *Advances in Methods and Practices in Psychological Science* 1 (3): 415–31. https://doi.org/10.1177/2515245918773743.

Thériault, Rémi, Jay A Olson, Sonia A Krol, and Amir Raz. 2021. "Body Swapping with a Black Person Boosts Empathy: Using Virtual Reality to Embody Another." *Quarterly Journal of Experimental Psychology* 74 (12): 2057–74. https://doi.org/10.1177/17470218211024826.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

[1] This argument can be used logically, as `TRUE` or `FALSE`, but can also be provided with a numeric value representing the cut-off threshold for the *p* value

[2] A great resource for this is the {flextable} e-book: https://ardata-fr.github.io/flextable-book/

[3] For convenience, colours are only used when the corresponding *p* value is at least smaller than .05

[4] Once one has identified outliers, it is also possible ot winsorize them with the `winsorize_mad()` function.