Tables

Introduction

First, load {tidyverse} and {papaja} and import the data.

```
library(tidyverse)
library(papaja)
hai <- read_csv("https://decisionslab.unl.edu/data/thayer_stevens_2020_data1.csv") |>
  select(experiment, participant, condition, gender,
         panas_pre = panas_pre_pos, panas_post = panas_post_pos)
hai_long <- hai |>
  pivot_longer(contains("panas"), names_to = "prepost", values_to = "panas") |>
  mutate(condition = fct_recode(condition, "Control" = "control", "HAI" = "hai"),
         prepost = fct_recode(prepost, "Pre" = "panas_pre", "Post" = "panas_post"),
         prepost = fct_relevel(prepost, c("Pre", "Post")))
```

Now let's build a data frame that will be our table.

```
(condition_prepost_means <- hai_long |>
  group_by(condition, prepost) |>
  summarise(across(starts_with("panas"), ~ mean(.x, na.rm = TRUE))))
```

```
## # A tibble: 4 x 3
## # Groups:   condition [2]
##   condition prepost panas
##   <fct>     <fct>   <dbl>
## 1 Control   Pre      2.99
## 2 Control   Post     2.76
## 3 HAI       Pre      2.97
## 4 HAI       Post     3.23
```

Tables by {knitr}

The {knitr} package uses the `kable()` function to format tables.

```
library(knitr)
kable(condition_prepost_means)
```

condition	prepost	panas
Control	Pre	2.989873
Control	Post	2.760759
HAI	Pre	2.971795
HAI	Post	3.232051

Column and row names

You can control column names and row names with `col.names` and `row.names`.

```
kable(condition_prepost_means,
      col.names = c("Condition", "Prepost", "Mean PANAS"))
```

Condition	Prepost	Mean PANAS
Control	Pre	2.989873
Control	Post	2.760759
HAI	Pre	2.971795
HAI	Post	3.232051

Column alignment

By default, character columns are left aligned and numeric columns are right aligned. You can set alignment manually with the `align` argument with `l` = left, `c` = center, and `r` = right. You can just pass a character string with a series of those letters.

```
kable(condition_prepost_means,
      col.names = c("Condition", "Prepost", "Mean PANAS"),
      align = "rcl")
```

Condition	Prepost	Mean PANAS
Control	Pre	2.989873
Control	Post	2.760759
HAI	Pre	2.971795
HAI	Post	3.232051

Digit rounding

Round the digits for all numeric data columns with `digits` argument.

```
kable(condition_prepost_means,
      col.names = c("Condition", "Prepost", "Mean PANAS"),
      digits = 2)
```

Condition	Prepost	Mean PANAS
Control	Pre	2.99
Control	Post	2.76
HAI	Pre	2.97
HAI	Post	3.23

If you want different digits for different columns, you can pass a vector to the `digits` argument.

```
condition_prepost_means |>
  pivot_wider(id_cols = condition, names_from = prepost, values_from = panas) |>
  kable(digits = c(0, 2, 3))
```

condition	Pre	Post
Control	2.99	2.761
HAI	2.97	3.232

Table 1
PANAS scores by condition and prepost

Condition	Prepost	Mean PANAS
Control	Pre	2.989873
Control	Post	2.760759
HAI	Pre	2.971795
HAI	Post	3.232051

Table titles

Add a title to the table with the `caption` argument. The good news is that we can cross-reference easily (Table 1). The bad news is that with captions, tables in PDFs are automatically placed at the top of the page. We'll see how to fix this later.

```
kable(condition_prepost_means,
      col.names = c("Condition", "Prepost", "Mean PANAS"),
      caption = "PANAS scores by condition and prepost")
```

Supplementing kable with {kableExtra}

The `kable()` function is intentionally simple to use and therefore does not have a lot of additional functionality. The {kableExtra} package supplements the `kable()` functionality with additional formatting options by adding additional functions after the `kable()` function call using the `|>` pipe (a bit like how `ggplot()` works). Check out Create Awesome LaTeX Table with knitr::kable and kableExtra.

```
# install.packages("kableExtra")
library(kableExtra)
```

General styling

The `kable_styling()` function formats a number of things such as font size, table width, and table alignment. I'll also add `latex_options = "hold_position"` to keep the table in the text. Otherwise, it puts it at the top of the page.

```
kable(condition_prepost_means,
      caption = "PANAS scores by condition and prepost",
      col.names = c("Condition", "Prepost", "Mean PANAS"),
      booktabs = TRUE) |>
  kable_styling(font_size = 18, latex_options = "hold_position")
```

Table 2
PANAS scores by condition and prepost

Condition	Prepost	Mean PANAS
Control	Pre	2.989873
Control	Post	2.760759
HAI	Pre	2.971795
HAI	Post	3.232051

Labels spanning rows

If you want to label groups of rows, use `pack_rows()`. Let's get rid of the `condition` column and label the conditions explicitly.

```
condition_prepost_means2 <- condition_prepost_means |>
  ungroup(condition) |>
  select(-condition)
```

```
kable(condition_prepost_means2,
      booktabs = TRUE)
```

prepost	panas
Pre	2.989873
Post	2.760759
Pre	2.971795
Post	3.232051

```
kable(condition_prepost_means2,
      booktabs = TRUE,
      col.names = c("", "Mean PANAS")) |>
  pack_rows("Control", start_row = 1, end_row = 2) |>
  pack_rows("HAI", start_row = 3, end_row = 4)
```

	Mean PANAS
Control	
Pre	2.989873
Post	2.760759
HAI	
Pre	2.971795
Post	3.232051

Notice that we removed the first column name with `""`.

Labels spanning columns

You can labels groups of columns with the `add_header_above()` function. Let's rearrange the data into wide format to illustrate this.

```
(wide_means <- condition_prepost_means |>
   unite(cond_prepost, condition:prepost) |>
   pivot_wider(names_from = cond_prepost, values_from = panas))
```

```
## # A tibble: 1 x 4
##   Control_Pre Control_Post HAI_Pre HAI_Post
##         <dbl>        <dbl>   <dbl>    <dbl>
## 1        2.99         2.76    2.97     3.23
```

```
kable(wide_means, booktabs = TRUE)
```

Control_Pre	Control_Post	HAI_Pre	HAI_Post
2.989873	2.760759	2.971795	3.232051

Now that the data are in wide format, we can add the column names by repeating *Pre* and *Post* then add the headers.

```
kable(wide_means,
      booktabs = TRUE,
      col.names = rep(c("Pre", "Post"), 2),
      digits = 2) |>
  add_header_above(c("Control" = 2, "HAI" = 2))
```

Control		HAI	
Pre	Post	Pre	Post
2.99	2.76	2.97	3.23

Maybe we need a column stating this is Mean PANAS.

```
wide_means2 <- wide_means |>
  mutate(label = "Mean PANAS", .before = 1)
kable(wide_means2,
      booktabs = TRUE,
      col.names = c("", rep(c("Pre", "Post"), 2)),
      digits = 2) |>
  add_header_above(c("", "Control" = 2, "HAI" = 2))
```

	Control		HAI	
	Pre	Post	Pre	Post
Mean PANAS	2.99	2.76	2.97	3.23

Table footnotes

Add table notes with the `footnote()` function.

```
kable(condition_prepost_means,
      booktabs = TRUE,
      caption = "PANAS scores by condition and prepost",
      col.names = c("Condition*", "Prepost", "Mean PANAS")) |>
  kable_styling(latex_options = "hold_position") |>
  footnote(general = "Source: Thayer & Stevens (2021)",
           symbol = "73 control participants, 72 HAI participants",
           footnote_as_chunk = TRUE)
```

Table 3
PANAS scores by condition and prepost

Condition*	Prepost	Mean PANAS
Control	Pre	2.989873
Control	Post	2.760759
HAI	Pre	2.971795
HAI	Post	3.232051

Note: Source: Thayer & Stevens (2021)
* 73 control participants, 72 HAI participants

Landscape

Rotate wide tables with `landscape()` function.

```
kable(condition_prepost_means,
      booktabs = TRUE,
      caption = "PANAS scores by condition and prepost",
      col.names = c("Condition", "Prepost", "Mean PANAS")) |>
  kable_styling(latex_options = "hold_position") |>
  footnote(general = "Source: Thayer & Stevens (2021)",
           footnote_as_chunk = TRUE) |>
  landscape()
```

Table 4
PANAS scores by condition and prepost

Condition	Prepost	Mean PANAS
Control	Pre	2.989873
Control	Post	2.760759
HAI	Pre	2.971795
HAI	Post	3.232051

Note: Source: Thayer & Stevens (2021)

Tables by {papaja}

The {papaja} package uses the `kable()` function to format tables in APA format with the `apa_table()` function. You can use many of the same arguments that are available in the `kable()` function. You can control where the table is placed (`here`, `top`, `bottom`) with the `placement` argument. You can add a general footnote with the `note` argument.

```
apa_table(condition_prepost_means,
          caption = "PANAS scores by condition and prepost",
          col.names = c("Condition", "Prepost", "Mean PANAS"),
          placement = "h",
          note = "Source: Thayer & Stevens (2021)")
```

Table 5
PANAS scores by condition and prepost

Condition	Prepost	Mean PANAS
Control	Pre	2.99
Control	Post	2.76
HAI	Pre	2.97
HAI	Post	3.23

Note. Source: Thayer & Stevens (2021)

Notice the alignment is different, with everything left aligned. Let's right align the means.

```
apa_table(condition_prepost_means,
          caption = "PANAS scores by condition and prepost",
          col.names = c("Condition", "Prepost", "Mean PANAS"),
          align = c("l", "l", "r"),
          placement = "h",
          note = "Source: Thayer & Stevens (2021)")
```

Table 6
PANAS scores by condition and prepost

Condition	Prepost	Mean PANAS
Control	Pre	2.99
Control	Post	2.76
HAI	Pre	2.97
HAI	Post	3.23

Note. Source: Thayer & Stevens (2021)

You can rotate to landscape orientation with the `landscape = TRUE` argument.

```
apa_table(condition_prepost_means,
          caption = "PANAS scores by condition and prepost",
          col.names = c("Condition", "Prepost", "Mean PANAS"),
          align = c("l", "l", "r"),
          placement = "h",
          note = "Source: Thayer & Stevens (2021)",
          landscape = TRUE)
```

Table 7
PANAS scores by condition and prepost

Condition	Prepost	Mean PANAS
Control	Pre	2.99
Control	Post	2.76
HAI	Pre	2.97
HAI	Post	3.23

Note. Source: Thayer & Stevens (2021)

APA-formatted statistics by {papaja}

{papaja} also includes `apa_print()`, which extracts statistical values in APA format.

Linear regression

```
hai_lm <- lm(panas_post ~ condition * gender, data = hai)
summary(hai_lm)
```

```
##
## Call:
## lm(formula = panas_post ~ condition * gender, data = hai)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.80909 -0.58525  0.01475  0.48333  1.88333
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               2.68525    0.09578  28.035  < 2e-16 ***
## conditionhai              0.52385    0.13287   3.943 0.000122 ***
## genderMale                0.33142    0.20066   1.652 0.100662
## conditionhai:genderMale -0.18218    0.30884  -0.590 0.556142
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7481 on 153 degrees of freedom
## Multiple R-squared:  0.109,  Adjusted R-squared:  0.09149
## F-statistic: 6.236 on 3 and 153 DF,  p-value: 0.0005053
```

```
apa_print(hai_lm)
```

```
## $estimate
## $estimate$Intercept
## [1] "$b = 2.69$, 95\\% CI $[2.50, 2.87]$"
##
## $estimate$conditionhai
## [1] "$b = 0.52$, 95\\% CI $[0.26, 0.79]$"
##
## $estimate$genderMale
## [1] "$b = 0.33$, 95\\% CI $[-0.07, 0.73]$"
##
## $estimate$conditionhai_genderMale
## [1] "$b = -0.18$, 95\\% CI $[-0.79, 0.43]$"
##
## $estimate$modelfit
## $estimate$modelfit$r2
```

```
## [1] "$R^2 = .11$"
##
## $estimate$modelfit$r2_adj
## [1] "$R^2_{adj} = .09$"
##
## $estimate$modelfit$aic
## [1] "$\\mathrm{AIC} = 360.36$"
##
## $estimate$modelfit$bic
## [1] "$\\mathrm{BIC} = 375.64$"
##
##
##
## $statistic
## $statistic$Intercept
## [1] "$t(153) = 28.03$, $p < .001$"
##
## $statistic$conditionhai
## [1] "$t(153) = 3.94$, $p < .001$"
##
## $statistic$genderMale
## [1] "$t(153) = 1.65$, $p = .101$"
##
## $statistic$conditionhai_genderMale
## [1] "$t(153) = -0.59$, $p = .556$"
##
## $statistic$modelfit
## $statistic$modelfit$r2
## [1] "$F(3, 153) = 6.24$, $p = .001$"
##
##
##
## $full_result
## $full_result$Intercept
## [1] "$b = 2.69$, 95\\% CI $[2.50, 2.87]$, $t(153) = 28.03$, $p < .001$"
##
## $full_result$conditionhai
## [1] "$b = 0.52$, 95\\% CI $[0.26, 0.79]$, $t(153) = 3.94$, $p < .001$"
##
## $full_result$genderMale
## [1] "$b = 0.33$, 95\\% CI $[-0.07, 0.73]$, $t(153) = 1.65$, $p = .101$"
##
## $full_result$conditionhai_genderMale
## [1] "$b = -0.18$, 95\\% CI $[-0.79, 0.43]$, $t(153) = -0.59$, $p = .556$"
##
```

```
## $full_result$modelfit
## $full_result$modelfit$r2
## [1] "$R^2 = .11$, $F(3, 153) = 6.24$, $p = .001$"
##
##
##
## $table
## A data.frame with 6 labelled columns:
##
##                                 term estimate      conf.int statistic  df p.value
## 1                          Intercept     2.69  [2.50, 2.87]     28.03 153  < .001
## 2                      Conditionhai      0.52  [0.26, 0.79]      3.94 153  < .001
## 3                        GenderMale      0.33 [-0.07, 0.73]      1.65 153    .101
## 4 Conditionhai $\\times$ GenderMale     -0.18 [-0.79, 0.43]     -0.59 153    .556
##
## term     : Predictor
## estimate : $b$
## conf.int : 95\\% CI
## statistic: $t$
## df       : $\\mathit{df}$
## p.value  : $p$
## attr(,"class")
## [1] "apa_results" "list"
```

```
apa_table(apa_print(hai_lm)$table,
          caption = "Linear regression results",
          placement = "h")
```

Table 8
Linear regression results

Predictor	*b*	95% CI	*t*	*df*	*p*
Intercept	2.69	[2.50, 2.87]	28.03	153	$< .001$
Conditionhai	0.52	[0.26, 0.79]	3.94	153	$< .001$
GenderMale	0.33	[-0.07, 0.73]	1.65	153	.101
Conditionhai × GenderMale	-0.18	[-0.79, 0.43]	-0.59	153	.556

Let's clean up those predictor names.

```
hai_lm_table <- apa_print(hai_lm)$table
hai_lm_table <- hai_lm_table |>
  mutate(term = str_replace(term, "Conditionhai", "Condition"),
         term = str_replace(term, "GenderMale", "Gender"))
apa_table(hai_lm_table,
          caption = "Linear regression results",
          placement = "h")
```

Table 9
Linear regression results

term	*b*	95% CI	*t*	*df*	*p*
Intercept	2.69	[2.50, 2.87]	28.03	153	$< .001$
Condition	0.52	[0.26, 0.79]	3.94	153	$< .001$
Gender	0.33	[-0.07, 0.73]	1.65	153	.101
Condition × Gender	-0.18	[-0.79, 0.43]	-0.59	153	.556

How could we name the first column *Predictor* instead of *term*?

Other table packages

- `{gt}` RStudio's grammar of tables (logically like ggplot2)
- `{flextable}` Good Word output but a bit tricky to work with
- `{huxtable}` Very flexible but tricky to work with