

# Tables in R Markdown

Melbourne Statistical Consulting Platform  
University of Melbourne  
April 2024

# Including tables in an R Markdown document

```
library(gt)
```

```
election_data <- tribble(  
  ~party, ~seats_won,  
  "Australian Greens",      3,  
  "Australian Labor Party", 55,  
  "Liberal",                21,  
  "The Nationals",          6,  
  "Other Candidates",       3  
)  
gt(election_data)
```

The `gt()` function in the `gt` package makes a table from a data frame. (You've seen this a few times before!)

In the knitted document or the RStudio Viewer panel (bottom-right), you'll see something like this:

party	seats_won
Australian Greens	3
Australian Labor Party	55
Liberal	21
The Nationals	6
Other Candidates	3

# Captions and headings

```
election_data %>%  
  gt(caption = "Victorian state election 2018 lower house results") %>%  
  cols_label(  
    party = "Party",  
    seats_won = "Number of seats won",  
  )
```

You can change column names either in your original data frame using `rename()`, or using the `cols_label()` function after `gt()`.

You can also provide a caption for your table.

In the knitted document:

Victorian state election 2018 lower house results

Party	Number of seats won
Australian Greens	3
Australian Labor Party	55
Liberal	21
The Nationals	6
Other Candidates	3

# Making tables of summary statistics

```
pig_behaviour_data <-  
  read_csv("pig_behaviour_by_time.csv")  
pig_behaviour_summary <- pig_behaviour_data %>%  
  group_by(Housing, Treatment) %>%  
  summarise(  
    across(Upright:Nosing_pen,  
      ~mean(. / Total_pigs, na.rm = TRUE))) %>%  
  ungroup()
```

pig\_behaviour\_summary

```
# A tibble: 4 × 9  
  Housing Treatment Upright Aggression Chewing_pig Playing  
  <chr>    <chr>      <dbl>      <dbl>      <dbl>    <dbl>  
1 FC      C          0.527      0.025      0.208  0.0217  
2 FC      HC          0.601      0.0153     0.178  0.0188  
3 PS      C          0.724      0.0143     0.23   0.0257  
4 PS      HC          0.723      0.0340     0.238  0.0113  
# i 3 more variables: Vocalising <dbl>,  
#   Exploring_pen <dbl>, Nosing_pen <dbl>
```

# Making tables of summary statistics

```
pig_behaviour_summary %>%  
  gt()
```

Housing	Treatment	Upright	Aggression	Chewing_pig	Playing	Vocalising	Exploring_pen	Nosing_pen
FC	C	0.5266667	0.02500000	0.2083333	0.02166667	0.1150000	0.3300000	0.03500000
FC	HC	0.6011765	0.01529412	0.1776471	0.01882353	0.1070588	0.3364706	0.04705882
PS	C	0.7242857	0.01428571	0.2300000	0.02571429	0.1414286	0.4357143	0.08285714
PS	HC	0.7226415	0.03396226	0.2377358	0.01132075	0.1660377	0.3792453	0.05660377

# Controlling decimal places

```
pig_behaviour_summary %>%  
  gt() %>%  
  fmt_number(Upright:Nosing_pen, decimals = 3)
```

You can use `fmt_number()` more than once to specify different decimal places for different columns. The columns can be specified similar to in `select()`: a single column (e.g. `Playing`), a list of columns (e.g. `c(Aggression, Vocalising)`) or as a range (`Upright:Nosing_pen`).

Think carefully about the appropriate number of decimal places to display. How precise were the original measurements? What would represent a meaningful difference?

Housing	Treatment	Upright	Aggression	Chewing_pig	Playing	Vocalising	Exploring_pen	Nosing_pen
FC	C	0.527	0.025	0.208	0.022	0.115	0.330	0.035
FC	HC	0.601	0.015	0.178	0.019	0.107	0.336	0.047
PS	C	0.724	0.014	0.230	0.026	0.141	0.436	0.083
PS	HC	0.723	0.034	0.238	0.011	0.166	0.379	0.057

# Converting proportions to percentages

```
pig_behaviour_summary %>%  
  gt() %>%  
  fmt_number(Upright:Nosing_pen, scale_by = 100,  
             decimals = 1, pattern = "{x}%")
```

Here we can use the `scale_by` option to multiply all of the proportions by 100. The `pattern` option allows us to add text before or after the number to be displayed, which is represented by the placeholder `{x}`.

Housing	Treatment	Upright	Aggression	Chewing_pig	Playing	Vocalising	Exploring_pen	Nosing_pen
FC	C	52.7%	2.5%	20.8%	2.2%	11.5%	33.0%	3.5%
FC	HC	60.1%	1.5%	17.8%	1.9%	10.7%	33.6%	4.7%
PS	C	72.4%	1.4%	23.0%	2.6%	14.1%	43.6%	8.3%
PS	HC	72.3%	3.4%	23.8%	1.1%	16.6%	37.9%	5.7%

# Column alignment

```
pig_behaviour_summary %>%  
  gt() %>%  
  fmt_number(Upright:Nosing_pen, scale_by = 100,  
             decimals = 1, pattern = "{x}%") %>%  
  cols_align("center", c(Housing, Treatment))
```

Column alignment is controlled using `cols_align()` and specifying `"left"`, `"right"` or `"center"` (note American spelling).

Housing	Treatment	Upright	Aggression	Chewing_pig	Playing	Vocalising	Exploring_pen	Nosing_pen
FC	C	52.7%	2.5%	20.8%	2.2%	11.5%	33.0%	3.5%
FC	HC	60.1%	1.5%	17.8%	1.9%	10.7%	33.6%	4.7%
PS	C	72.4%	1.4%	23.0%	2.6%	14.1%	43.6%	8.3%
PS	HC	72.3%	3.4%	23.8%	1.1%	16.6%	37.9%	5.7%



# Grouping rows and columns

```
pig_behaviour_summary %>%  
  group_by(Housing) %>%  
  gt() %>%  
  fmt_number(Upright:Nosing_pen, scale_by = 100,  
             decimals = 1, pattern = "{x}%") %>%  
  cols_align("center", Treatment) %>%  
  tab_spanner("Percentage of time engaging in behaviour",  
             Upright:Nosing_pen)
```

`group_by()` called **before** `gt()` will group rows of a table together based on one or more factor variables. The grouping factor will no longer be shown as a column in the table.

`tab_spanner()` will add a heading above the table which spans across multiple columns.

Treatment	Percentage of time engaging in behaviour						
	Upright	Aggression	Chewing_pig	Playing	Vocalising	Exploring_pen	Nosing_pen
FC							
C	52.7%	2.5%	20.8%	2.2%	11.5%	33.0%	3.5%
HC	60.1%	1.5%	17.8%	1.9%	10.7%	33.6%	4.7%
PS							
C	72.4%	1.4%	23.0%	2.6%	14.1%	43.6%	8.3%
HC	72.3%	3.4%	23.8%	1.1%	16.6%	37.9%	5.7%

# Fixing up labels

```
pig_behaviour_summary %>%
  mutate(Housing = fct_recode(Housing,
    "Farrowing crate" = "FC",
    "PigSAFE" = "PS"),
    Treatment = fct_recode(Treatment,
    "Control" = "C",
    "Human contact" = "HC")) %>%
  group_by(Housing) %>%
  gt() %>%
  fmt_number(Upright:Nosing_pen, scale_by = 100,
    decimals = 1, pattern = "{x}%") %>%
  cols_align("left", Treatment) %>%
  cols_label(Chewing_pig = "Chewing pig",
    Exploring_pen = "Exploring pen",
    Nosing_pen = "Nosing pen") %>%
  tab_spanner("Percentage of time engaging in behaviour",
    Upright:Nosing_pen)
```

Treatment	Percentage of time engaging in behaviour						
	Upright	Aggression	Chewing pig	Playing	Vocalising	Exploring pen	Nosing pen
Farrowing crate							
Control	52.7%	2.5%	20.8%	2.2%	11.5%	33.0%	3.5%
Human contact	60.1%	1.5%	17.8%	1.9%	10.7%	33.6%	4.7%
PigSAFE							
Control	72.4%	1.4%	23.0%	2.6%	14.1%	43.6%	8.3%
Human contact	72.3%	3.4%	23.8%	1.1%	16.6%	37.9%	5.7%

You've already seen the tools to do this - it's just a matter of putting it all together!

# Adding bold to headings

```
pig_behaviour_summary %>%
  mutate(Housing = fct_recode(Housing,
    "Farrowing crate" = "FC",
    "PigSAFE" = "PS"),
    Treatment = fct_recode(Treatment,
    "Control" = "C",
    "Human contact" = "HC")) %>%
  group_by(Housing) %>%
  gt() %>%
  fmt_number(Upright:Nosing_pen, scale_by = 100,
    decimals = 1, pattern = "{x}%") %>%
  cols_align("left", Treatment) %>%
  cols_label(Chewing_pig = "Chewing pig",
    Exploring_pen = "Exploring pen",
    Nosing_pen = "Nosing pen") %>%
  tab_spanner("Percentage of time engaging in behaviour",
    Upright:Nosing_pen) %>%
  tab_style(
    locations = list(cells_column_spanners(),
      cells_column_labels()),
    style = cell_text(weight = "bold")
  ) %>%
  tab_style(
    locations = cells_row_groups(),
    style = cell_text(style = "italic")
  )
```

Treatment	Percentage of time engaging in behaviour						
	Upright	Aggression	Chewing pig	Playing	Vocalising	Exploring pen	Nosing pen
<i>Farrowing crate</i>							
Control	52.7%	2.5%	20.8%	2.2%	11.5%	33.0%	3.5%
Human contact	60.1%	1.5%	17.8%	1.9%	10.7%	33.6%	4.7%
<i>PigSAFE</i>							
Control	72.4%	1.4%	23.0%	2.6%	14.1%	43.6%	8.3%
Human contact	72.3%	3.4%	23.8%	1.1%	16.6%	37.9%	5.7%

The `tab_style()` function can be used to change font options for parts of the table, e.g. to make headings bold. (Sadly, the code to do this is quite unwieldy!)

# Changing borders and row spacing

```
pig_behaviour_summary %>%
  mutate(Housing = fct_recode(Housing,
    "Farrowing crate" = "FC",
    "PigSAFE" = "PS"),
    Treatment = fct_recode(Treatment,
    "Control" = "C",
    "Human contact" = "HC")) %>%
  group_by(Housing) %>%
  gt() %>%
  fmt_number(Upright:Nosing_pen, scale_by = 100,
    decimals = 1, pattern = "{x}%") %>%
  cols_align("left", Treatment) %>%
  cols_label(Chewing_pig = "Chewing pig",
    Exploring_pen = "Exploring pen",
    Nosing_pen = "Nosing pen") %>%
  tab_spanner("Percentage of time engaging in behaviour",
    Upright:Nosing_pen) %>%
  tab_options(data_row.padding = px(3),
    row_group.padding = px(3),
    row_group.border.top.color = "black",
    row_group.border.top.width = px(1),
    row_group.border.bottom.style = "none",
    column_labels.border.top.color = "black",
    column_labels.border.bottom.color = "black",
    column_labels.border.bottom.width = px(1),
    table_body.border.top.style = "none",
    table_body.border.bottom.color = "black",
    table_body.hlines.style = "none")
```

Treatment	Percentage of time engaging in behaviour						
	Upright	Aggression	Chewing pig	Playing	Vocalising	Exploring pen	Nosing pen
<i>Farrowing crate</i>							
Control	52.7%	2.5%	20.8%	2.2%	11.5%	33.0%	3.5%
Human contact	60.1%	1.5%	17.8%	1.9%	10.7%	33.6%	4.7%
<i>PigSAFE</i>							
Control	72.4%	1.4%	23.0%	2.6%	14.1%	43.6%	8.3%
Human contact	72.3%	3.4%	23.8%	1.1%	16.6%	37.9%	5.7%

The `tab_options()` function allows us to change the visual appearance of the table. One common example of this is to reduce the spacing and remove horizontal lines between table rows.

# Exporting to Word or Excel

Option 1: the `gtsave()` function will save a `gt()` table in various formats, include `.docx`. Most, but not all, of the options to customise appearance will apply to the Word version of the table.

```
pig_behaviour_summary %>%  
  gt() %>%  
  gtsave("pig_behaviour_summary.docx")
```

Option 2: save the results to a CSV or Excel file and open in Excel. You can save a data frame to a CSV file using the `write_csv()` function:

```
write_csv(pig_behaviour_summary,  
          "pig_behaviour_summary.csv")
```

Alternatively, the `writexl` package has a `write_xlsx()` function to save a data frame to an Excel spreadsheet:

```
library(writexl)  
write_xlsx(pig_behaviour_summary,  
           "pig_behaviour_summary.xlsx")
```

# Other possibilities

Documentation and examples for the **gt** package: <https://gt.rstudio.com/>

The **gtsummary** package produces "Table 1" style summaries (e.g. demographics, possibly by treatment group) in a style similar to what you'd find in an academic journal. More information: <http://www.danielsjoberg.com/gtsummary/>

## Exercise 4.1.