Spreadsheet Munging Strategies

Duncan Garmonsway

Contents

W	'elcome	Ē
1	Setup 1.1 Packages 1.2 Data	7
2	2.3 Meaningfully formatted rows	21 15 17 21 25 25
3	3.1 Simple unpivoting	29 32 38
4	4.1 Small multiples with all headers present for each multiple	69 69 73 73 74
5	5.1 An example formatting lookup	77 78 88 90 91 93
6	Data validation	95
7	Formulas	97
8	- · · · · · · · · · · · · · · · · · · ·	
9		0 3

,	4	CONTENTS
4	4	CONTENTS

9.2	Vaccinations	09
9.3	US Crime	11

Welcome

This is a work-in-progress book about getting data out of spreadsheets, no matter how peculiar. The book is designed primarily for R users who have to extract data from spreadsheets and who are already familiar with the tidyverse. It has a cookbook structure, and can be used as a reference, but readers who begin in the middle might have to work backwards from time to time.

R packages that feature heavily are

- unpivotr: deals with non-tabular data, especially from spreadsheets.
- tidyxl: imports non-tabular data from Excel files

Tidyxl and unpivotr are much more complicated than readxl, and that's the point. Tidyxl and unpivotr give you more power and complexity when you need it.

Please help me to improve this book by opening a GitHub issue or tweeting.

The online version of this book is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Figure 1: Creative Commons License

6 CONTENTS

Chapter 1

Setup

This section describes how the code in the book is set up.

1.1 Packages

Here are the packages used by the code in this book. The last three are my own: tidyxl, unpivotr and smungs. You will need to install the latest versions from CRAN or GitHub.

```
library(tibble)
library(tidyr)
library(dplyr)
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##
       filter, lag
## The following objects are masked from 'package:base':
##
##
       intersect, setdiff, setequal, union
library(purrr)
library(stringr)
library(readr)
library(readxl)
library(tidyxl)
library(unpivotr)
library(smungs) # GitHub only https://github.com/nacnudus/smungs
```

1.2 Data

The examples draw from a spreadsheet of toy data, included in the unpivotr package. It is recommended to download the spreadsheet and have open it in a spreadsheet application while you read the book.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
```

8 CHAPTER 1. SETUP

Chapter 2

Tidy-ish tables

This chapter is a gentle introduction, by taking what you already know about importing tidy tabular data (with read.csv() or the readr package), and shows you how to how to do the same things with tidyxl and unpivotr. It works up to tables that are mostly tidy, but have subtle problems.

2.1 Clean & tidy tables

	Α	В
1	Name	Age
2	Matilda	1
3	Nicholas	3
4	Olivia	5

If the tables in the spreadsheet are clean and tidy, then you should use a package like readxl. But it's worth knowing how to emulate readxl with tidyxl and unpivotr, because some *almost* clean tables can be handled using these techniques.

Clean and tidy means

- One table per sheet
- A single row of column headers, or no headers
- A single data type in each column
- Only one kind of sentinel value (to be interpreted as NA)
- No meaningful formatting
- No data buried in formulas
- No need to refer to named ranges

Here's the full process.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
xlsx_cells(path, sheet = "clean") %>%
```

##

1

2

3

<dbl> <chr>

1 Matilda

3 Nicholas

5 Olivia

```
behead("N", header) %>%
select(row, data_type, header, character, numeric) %>%
spatter(header) %>%
select(-row)

## # A tibble: 3 x 2
## Age Name
```

tidyxl::xlsx_cells() imports the spreadsheet into a data frame, where each row of the data frame describes one cell of the spreadsheet. The columns row and col (and address) describe the position of the cell, and the value of the cell is in one of the columns error, logical, numeric, date, character, depending on the type of data in the cell. The column data_type says which column the value is in. Other columns describe formatting and formulas.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
xlsx_cells(path, sheet = "clean") %>%
  select(row, col, data_type, character, numeric)
```

```
## # A tibble: 8 x 5
##
       row
              col data_type character numeric
##
     <int> <int> <chr>
                             <chr>
                                           <dbl>
## 1
                1 character Name
                                              NA
## 2
         1
                2 character Age
                                              NA
         2
## 3
                1 character Matilda
                                              NA
         2
## 4
                             <NA>
                2 numeric
                                               1
## 5
         3
                1 character Nicholas
                                              NA
## 6
         3
                2 numeric
                             <NA>
                                               3
## 7
         4
                1 character Olivia
                                              NA
## 8
         4
                2 numeric
                             < N A >
                                               5
```

unpivotr::behead() takes one level of headers from a pivot table and makes it part of the data. Think of it like tidyr::gather(), except that it works when there is more than one row of headers (or more than one column of row-headers), and it only works on tables that have first come through unpivotr::as_cells() or tidyxl::xlsx_cells().

```
xlsx_cells(path, sheet = "clean") %>%
select(row, col, data_type, character, numeric) %>%
behead("N", header)
```

```
## # A tibble: 6 x 6
##
             col data_type character numeric header
##
     <int> <int> <chr>
                            <chr>
                                         <dbl> <chr>
## 1
         2
               1 character Matilda
                                            NA Name
## 2
         2
                            <NA>
               2 numeric
                                             1 Age
## 3
         3
               1 character Nicholas
                                            NA Name
         3
                            <NA>
                                             3 Age
## 4
                2 numeric
                1 character Olivia
## 5
         4
                                            NA Name
## 6
                2 numeric
                            <NA>
                                             5 Age
```

unpivotr::spatter() spreads key-value pairs across multiple columns, like tidyx1::spread(), except that it handles mixed data types. It knows which column contains the cell value (i.e. the character column or the numeric column), by checking the data_type column. Just like tidyr::spread(), it can be confused by extraneous data, so it's usually a good idea to drop the col column first, and to keep the row column.

```
xlsx_cells(path, sheet = "clean") %>%
  select(row, col, data_type, character, numeric) %>%
  behead("N", header) %>%
  select(-col) %>%
  spatter(header) %>%
  select(-row)
## # A tibble: 3 x 2
##
       Age Name
     <dbl> <chr>
##
## 1
         1 Matilda
## 2
         3 Nicholas
## 3
         5 Olivia
```

In case the table has no column headers, you can spatter the col column instead of a nonexistent header column.

```
xlsx_cells(path, sheet = "clean") %>%
  dplyr::filter(row >= 2) %>%
  select(row, col, data_type, character, numeric) %>%
  spatter(col) %>%
  select(-row)
## # A tibble: 3 x 2
##
     `1`
                .2.
##
     <chr>>
              <dbl>
## 1 Matilda
                  1
## 2 Nicholas
                  3
## 3 Olivia
                  5
```

Tidyxl and unpivotr are much more complicated than readxl, and that's the point: tidyxl and unpivotr give you more power and complexity when you need it.

```
read_excel(path, sheet = "clean")
## # A tibble: 3 x 2
##
     Name
                 Age
##
     <chr>>
               <dbl>
## 1 Matilda
## 2 Nicholas
                   3
## 3 Olivia
                   5
read_excel(path, sheet = "clean", col_names = FALSE, skip = 1)
## # A tibble: 3 x 2
     X__1
                X_{-2}
##
##
     <chr>>
               <dbl>
## 1 Matilda
                   1
## 2 Nicholas
                   3
## 3 Olivia
                   5
```

2.2 Almost-tidy tables

For tables that are already 'tidy' (a single row of column headers), use packages like readxl that specialise in importing tidy data.

For everything else, read on.

2.2.1 Transposed (headers in the first row, data extends to the right)

	Α	В	С	D
1	Name	Matilda	Nicholas	Olivia
2	Age	1	3	5

Most packages for importing data assume that the headers are in the first row, and each row of data is an observation. They usually don't support the alternative: headers in the first column, and each column of data is an observation.

You can hack a way around this by importing without recognising any headers, transposing with t() (which outputs a matrix), placing the headers as names, and converting back to a data frame, but this almost always results in all the data types being converted.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
read_excel(path, sheet = "transposed", col_names = FALSE) %>%
   t() %>%
   `colnames<-`(.[1, ]) %>%
   .[-1, ] %>%
   as_tibble()
```

```
## # A tibble: 3 x 2
## Name Age
## <chr> <chr>
## 1 Matilda 1
## 2 Nicholas 3
## 3 Olivia 5
```

Tidyxl and unpivotr are agnostic to the layout of tables. Importing the transpose is the same is importing the usual layout, merely using the "W" (west) direction instead of "N" (north) when beheading the headers.

```
xlsx_cells(path, sheet = "transposed") %>%
behead("W", header) %>%
select(col, data_type, header, character, numeric) %>%
spatter(header) %>%
select(Name, Age)
```

```
## # A tibble: 3 x 2
## Name Age
## <chr> <dbl>
## 1 Matilda 1
## 2 Nicholas 3
## 3 Olivia 5
```

2.2.2 Other stuff on the same sheet

	Α	В	С	D
1	Title text			
2				
3		Name	Age	
4		Matilda	1	
5		Nicholas	3	
6				
7				Footnote

It will be more complicated when the table doesn't begin in cell A1, or if there are non-blank cells above, below or either side of the table.

If you know at coding time which rows and columns the table occupies, then you can do the following.

- Blank or non-blank cells above the table: use the skip argument of readx1::read_excel().
- Blank or non-blank cells either side of the table: use the col_types argument of readxl::read_excel() to ignore those columns.
- Blank or non-blank cells below the table: use n_max argument of readxl::read_excel() to ignore those rows.

```
## # A tibble: 2 x 2
## Name Age
## <chr> <dbl>
## 1 Matilda 1
## 2 Nicholas 3
```

If you don't know at coding time which rows and columns the table occupies (e.g. when the latest version of the spreadsheet is published and the table has moved), then one strategy is to read the spreadsheet with tidyxl::xlsx_cells() first, and inspect the results to determine the boundaries of the table. Then use those boundaries as the skip, n_max and col_types arguments to readxl::read_excel()

- 1. Read the spreadsheet with tidyxl::xlsx_cells(). Filter the result for sentinel values, e.g. the cells containing the first and final column headers, and a cell in the final row of data.
- 2. Construct the arguments skip, n_max and col_types so that readxl::read_excel() gets the exact dimensions of the table.

```
# Step 1: read the spreadsheet and filter for sentinel values to detect the
# top-left and bottom-right cells
cells <- xlsx cells(path, sheet = "notes")</pre>
rectify(cells)
## # A tibble: 7 x 5
    `row/col` `1(A)`
                          `2(B)`
                                   `3(C)` `4(D)`
##
        <int> <chr>
                          <chr>
                                   <chr> <chr>
## 1
           1 Title text <NA>
                                   <NA>
                                           <NA>
## 2
            2 <NA>
                      <NA>
                                   <NA>
                                           <NA>
## 3
            3 <NA>
                          Name
                                   Age
                                           <NA>
## 4
            4 <NA>
                         Matilda 1
                                           <NA>
## 5
           5 <NA>
                         Nicholas 3
                                           <NA>
## 6
            6 <NA>
                          <NA>
                                   <NA>
                                           <NA>
## 7
             7 <NA>
                          <NA>
                                   <NA>
                                          Footnote
top_left <-
  dplyr::filter(cells, character == "Name") %>%
  select(row, col)
top_left
## # A tibble: 1 x 2
##
      row col
##
   <int> <int>
## 1
         3
# It can be tricky to find the bottom-right cell because you have to make some
# assumptions. Here we assume that only cells within the table are numeric.
bottom_right <-
  dplyr::filter(cells, data_type == "numeric") %>%
  summarise(row = max(row), col = max(col))
bottom_right
## # A tibble: 1 x 2
##
       row col
##
     <dbl> <dbl>
## 1
        5
# Step 2: construct the arguments `skip` and `n_max` for read_excel()
skip <- top_left$row - 1L</pre>
n_rows <- bottom_right$row - skip</pre>
read_excel(path, sheet = "notes", skip = skip, n_max = n_rows)
## # A tibble: 2 x 2
##
    Name
                Age
##
     <chr>>
              <dbl>
## 1 Matilda
                  1
## 2 Nicholas
Here's another way using only tidyxl and unpivotr.
# Step 2: filter for cells between the top-left and bottom-right, and spatter
# into a table
cells %>%
  dplyr::filter(between(row, top_left$row, bottom_right$row),
         between(col, top_left$col, bottom_right$col)) %>%
```

```
select(row, col, data_type, character, numeric) %>%
behead("N", header) %>%
select(-col) %>%
spatter(header) %>%
select(-row)

## # A tibble: 2 x 2
## Age Name
## <dbl> <chr>
## 1  1 Matilda
## 2  3 Nicholas
```

2.3 Meaningfully formatted rows

	Α		В	
1	Age		Height	
2		1		2
3		3		4
4		5		6

As with clean, tidy tables, but with a second step to interpret the formatting.

Sometimes whole rows in a table are highlighted by formatting them with, say, a bright yellow fill. The highlighting could mean "this observation should be ignored", or "this product is no longer available". Different colours could mean different levels of a hierarchy, e.g. green for "pass" and red for "fail".

There are three steps to interpreting this.

- 1. Import the table, taking only the cell values and ignoring the formatting.
- 2. Import one column of the table, taking only the formatting and not the cell values.
- 3. Use dplyr::bind_cols() to append the column of formatting to the table of cell values. You can then interpret the formatting however you like.

Step 1 is the same as clean, tidy tables.

Step 2 uses tidyxl::xlsx_cells() to load the data, tidyxl::xlsx_formats(), and several tidyverse functions to link the two and filter for only one column. Why only one column? Because if a whole row is highlighted, then you only need to know the highlighting of one column to know the highlighting of all the others.

This is a special case of the following section, meaningfully formatted cells. Here <code>dplyr::bind_cols()</code> can be used as a shortcut, because we are joining exactly n rows of formatting to n rows of data. The following sections is a more general case that can be used instead of this procedure.

```
# Step 1: import the table taking only cell values and ignoring the formatting
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
x <- read_excel(path, sheet = "highlights")</pre>
```

```
# Step 2: import one column of the table, taking only the formatting and not the
# cell values
# `formats` is a pallette of fill colours that can be indexed by the
# `local_format_id` of a given cell to get the fill colour of that cell
fill_colours <- xlsx_formats(path) $local fill patternFill fgColor rgb
# Import all the cells, filter out the header row, filter for the first column,
# and create a new column `fill_colour` of the fill colours, by looking up the
# local_format_id of each cell in the `fill_colours` pallette.
fills <-
  xlsx_cells(path, sheet = "highlights") %>%
  dplyr::filter(row >= 2, col == 1) %>% # Omit the header row
  mutate(fill_colour = fill_colours[local_format_id]) %>%
  select(fill_colour)
# Step 3: append the `fill` column to the rest of the data
bind_cols(x, fills) %>%
 select(Age, Height, fill_colour)
```

Note that the fill colour is expressed as an RGB value with transparency in the first two letters, e.g. FFFFFF00 is FF (opaque), with FFFF00 (yellow).

Here's another way using only tidyxl and unpivotr.

```
fill_colours <- xlsx_formats(path)$local$fill$patternFill$fgColor$rgb

xlsx_cells(path, sheet = "highlights") %>%
  mutate(fill_colour = fill_colours[local_format_id]) %>%
  select(row, col, data_type, character, numeric, fill_colour) %>%
  behead("N", header) %>%
  select(-col, -character) %>%
  spatter(header) %>%
  select(-row)
```

```
## # A tibble: 3 x 3
## cfill_colour Age Height
## cchr> <dbl> <dbl> <dbl>
## 1 <NA> 1 2
## 2 FFFFFF00 3 4
## 3 <NA> 5 6
```

##

<chr>

2 FFFFFF00 <NA>

1 <NA>

<chr>

<NA>

2.4 Meaningfully formatted cells

	Α	В	С
1	Name	Age	Height
2	Matilda	1	2
3	Nicholas	3	4
4	Olivia	5	6

If single cells are highlighted, rather than whole rows, then the highlights probably indicate something about the column rather than the row. For example, a highlighted cell in a column called "age" of a table of medical patients, might mean "the age of this patient is uncertain".

One way to deal with this is to create a new column in the final table for each column in the original that has any highlighted cells. For example, if highlighted cells mean "this value is uncertain", and some cells in the age and height columns are highlighted, then you could create two new columns: uncertain_age, and uncertain_height, by following the procedure of meaningfully formatted rows for each column age and height.

```
# Step 1: import the table taking only cell values and ignoring the formatting
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
x <- read_excel(path, sheet = "annotations")</pre>
# Step 2: import one column of the table, taking only the formatting and not the
# cell values
# `formats` is a pallette of fill colours that can be indexed by the
# `local format id` of a given cell to get the fill colour of that cell
fill_colours <- xlsx_formats(path) $local fill patternFill fgColor rgb
# Import all the cells, filter out the header row, filter for the first column,
\# and create new columns `something_fill` of the fill colours, by looking up the
# local_format_id of each cell in the `formats` pallette.
fills <-
  xlsx_cells(path, sheet = "annotations") %>%
  dplyr::filter(row >= 2, col >= 2) %>% # Omit the header row and name column
  mutate(fill_colour = fill_colours[local_format_id]) %>%
  select(row, col, fill_colour) %>%
  spread(col, fill_colour) %>%
  select(-row) %>%
  set_names(paste0(colnames(x)[-1], "_fill"))
fills
## # A tibble: 3 x 2
     Age fill Height fill
```

```
## 3 <NA>
            FF92D050
# Step 3: append the `fill` column to the rest of the data
bind_cols(x, fills)
## # A tibble: 3 x 5
             Age Height Age_fill Height_fill
    Name
##
    <chr>
            <dbl> <dbl> <chr>
                                <chr>
## 1 Matilda
              1
                      2 <NA>
                                 <NA>
## 2 Nicholas
                      4 FFFFFF00 <NA>
                3
## 3 Olivia
                5
                      6 <NA>
                                FF92D050
Here's the same thing, but using only tidyxl and unpivotr
fill_colours <- xlsx_formats(path) $local fill patternFill fgColor rgb
cells <-
 xlsx_cells(path, sheet = "annotations") %>%
 mutate(fill_colour = fill_colours[local_format_id]) %>%
 select(row, col, data_type, character, numeric, fill_colour)
cells
## # A tibble: 12 x 6
       row col data_type character numeric fill_colour
##
    <int> <int> <chr>
                         <chr> <dbl> <chr>
                                    NA <NA>
## 1
       1 1 character Name
## 2
             2 character Age
                                     NA <NA>
        1
        1 3 character Height
2 1 character Matilda
                                     NA <NA>
## 3
## 4
                                     NA <NA>
## 5
       2 2 numeric <NA>
                                      1 <NA>
                                       2 <NA>
## 6
       2 3 numeric <NA>
           1 character Nicholas
## 7
       3
                                     NA <NA>
## 8
                                      3 FFFFFF00
       3 2 numeric <NA>
## 9
       3 3 numeric <NA>
                                      4 <NA>
       4 1 character Olivia
                                   NA <NA>
## 10
       4 2 numeric <NA>
## 11
                                      5 <NA>
             3 numeric <NA>
## 12
                                      6 FF92D050
values <-
 cells %>%
 select(-fill_colour) %>%
 behead("N", header) %>%
 select(-col) %>%
 spatter(header)
values
## # A tibble: 3 x 4
    row Age Height Name
## <int> <dbl> <dbl> <chr>
## 1 2 1
                 2 Matilda
## 2
      3 3
                   4 Nicholas
            5
## 3
        4
                   6 Olivia
fills <-
 cells %>%
 behead("N", header) %>%
 mutate(header = paste0(header, "_fill")) %>%
select(row, header, fill_colour) %>%
```

```
spread(header, fill_colour)
fills
## # A tibble: 3 x 4
##
       row Age_fill Height_fill Name_fill
     <int> <chr>
                     <chr>
                                  <chr>>
         2 <NA>
## 1
                                  <NA>
                     < NA >
## 2
         3 FFFFFF00 <NA>
                                  <NA>
## 3
         4 <NA>
                     FF92D050
                                  <NA>
left_join(values, fills, by = "row") %>%
  select(-row)
## # A tibble: 3 x 6
##
       Age Height Name
                            Age_fill Height_fill Name_fill
     <dbl> <dbl> <chr>
##
                            <chr>>
                                      <chr>
                                                   <chr>>
                2 Matilda <NA>
                                      <NA>
                                                   <NA>
## 2
         3
                4 Nicholas FFFFFF00 <NA>
                                                   <NA>
## 3
         5
                            <NA>
                                      FF92D050
                6 Olivia
                                                   <NA>
```

Another way would be to make the table what I call "extra-tidy". If it is tidy, then each row is an observation, and each column is a variable. To make it "extra-tidy", you gather() the variables so that each row is one observation of one variable. This works best when every variable has the same data type, otherwise the values will be coerced, probably to a character.

```
(x <- read_excel(path, sheet = "annotations"))</pre>
## # A tibble: 3 x 3
                Age Height
     Name
##
     <chr>
              <dbl> <dbl>
## 1 Matilda
                  1
## 2 Nicholas
                  3
## 3 Olivia
                  5
# Extra-tidy
extra_tidy <-
  x %>%
  gather(variable, value, -Name) %>%
  arrange(Name, variable)
extra_tidy
## # A tibble: 6 x 3
##
     Name
              variable value
##
                        <dbl>
     <chr>>
              <chr>>
## 1 Matilda Age
## 2 Matilda Height
                            2
## 3 Nicholas Age
                            3
## 4 Nicholas Height
                            4
## 5 Olivia
              Age
                            5
## 6 Olivia
                            6
              Height
```

With an extra-tidy dataset, the formatting can now be appended to the values of individual variables, rather than to whole observations.

```
# Extra-tidy, with row and column numbers of the original variables
extra_tidy <-
read_excel(path, sheet = "annotations") %>%
```

```
mutate(row = row_number() + 1L) %>%
 gather(variable, value, -row, -Name) %>%
 group_by(row) %>%
 mutate(col = row_number() + 1L) %>%
 ungroup() %>%
 select(row, col, Name, variable, value) %>%
 arrange(row, col)
extra_tidy
## # A tibble: 6 x 5
    row col Name
##
                      variable value
   <int> <int> <chr>
                       <chr>
                               <dbl>
## 1 2 2 Matilda Age
      2 3 Matilda Height
## 2
                                   2
      3 2 Nicholas Age
## 3
                                   3
## 4
    3 3 Nicholas Height
                                   4
## 5
      4 2 Olivia Age
     4 3 Olivia Height
## 6
                                   6
# `formats` is a pallette of fill colours that can be indexed by the
# `local_format_id` of a given cell to get the fill colour of that cell
fill_colours <- xlsx_formats(path) $local fill patternFill fgColor rgb
# Import all the cells, filter out the header row, filter for the first column,
# and create a new column `uncertain` based on the fill colours, by looking up
# the local_format_id of each cell in the `formats` pallette.
fills <-
 xlsx_cells(path, sheet = "annotations") %>%
 dplyr::filter(row >= 2, col >= 2) %>% # Omit the header row and name column
 mutate(fill_colour = fill_colours[local_format_id]) %>%
 select(row, col, fill_colour)
fills
## # A tibble: 6 x 3
      row col fill_colour
   <int> <int> <chr>
##
## 1
      2 2 <NA>
## 2
      2
            3 <NA>
## 3
      3
            2 FFFFFF00
## 4
       3
            3 <NA>
## 5
       4 2 <NA>
            3 FF92D050
# Step 3: append the `fill` column to the rest of the data
left_join(extra_tidy, fills, by = c("row", "col"))
## # A tibble: 6 x 6
##
      row col Name
                       variable value fill_colour
## <int> <int> <chr>
                       <chr> <dbl> <chr>
## 1 2 2 Matilda Age
                                 1 <NA>
## 2
      2
            3 Matilda Height
                                 2 <NA>
      3 2 Nicholas Age
## 3
                                   3 FFFFFF00
                                 4 <NA>
## 4 3 3 Nicholas Height
## 5 4 2 Olivia Age
                                 5 <NA>
## 6
    4
            3 Olivia Height 6 FF92D050
```

Here's the same extra-tidy version, but using only tidyxl and unpivotr.

```
fill_colours <- xlsx_formats(path)$local$fill$patternFill$fgColor$rgb
xlsx_cells(path, sheet = "annotations") %>%
 mutate(fill_colour = fill_colours[local_format_id]) %>%
 select(row, col, data_type, character, numeric, fill_colour) %>%
 behead("W", Name) %>%
 behead("N", variable) %>%
 select(-data_type, -character, value = numeric)
## # A tibble: 6 x 6
##
      row col value fill_colour Name
                                         variable
##
    <int> <int> <dbl> <chr>
                                         <chr>>
## 1
        2 2
                   1 <NA>
                                Matilda Age
## 2
        2
            3
                   2 <NA>
                               Matilda Height
## 3
        3
            2
                 3 FFFFFF00 Nicholas Age
        3
## 4
             3
                  4 <NA>
                                Nicholas Height
        4
             2 5 <NA>
## 5
                                Olivia Age
                                Olivia Height
## 6
                   6 FF92D050
```

2.5 Layered meaningful formatting

	Α	В	С
1	Name	Weight	Price
2	Knife	7	8
3	Fork	5	6
4	Spoon	3	4
5	Teaspoon	1	2

Sometimes different kinds of formatting relate to clearly different aspects of an observation, e.g. yellow highlight for "uncertain data" and red text for "product no longer available". Both yellow highlighting and red text in the same row would indicate uncertain data and unavailability of the product at the same time.

Deal with it by reading each kind of formatting into a separate column, e.g. fill colour into one column, font colour into another, bold/not-bold into a another, etc.

```
# Step 1: import the table taking only cell values and ignoring the formatting
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
x <- read_excel(path, sheet = "combined-highlights")

# Step 2: import one kind of formatting of one column of the table
# `formats` is a pallette of fill colours that can be indexed by the</pre>
```

```
# `local_format_id` of a given cell to get the fill colour of that cell
fill_colours <- xlsx_formats(path) $local fill patternFill fgColor rgb
font_colours <- xlsx_formats(path)$local$font$color$rgb</pre>
# Import all the cells, filter out the header row, filter for the first column,
# and create a new column `fill` of the fill colours, by looking up the
# local_format_id of each cell in the `formats` pallette.
formats <-
 xlsx_cells(path, sheet = "combined-highlights") %>%
  dplyr::filter(row >= 2, col == 1) %>% # Omit the header row
 mutate(fill_colour = fill_colours[local_format_id],
        font_colour = font_colours[local_format_id]) %>%
  select(fill_colour, font_colour)
# Step 3: append the `fill` column to the rest of the data
bind_cols(x, formats)
## # A tibble: 4 x 5
           Weight Price fill_colour font_colour
##
    <chr>
             <dbl> <dbl> <chr>
                                    <chr>
## 1 Knife
                7
                       8 <NA>
                                     FF000000
## 2 Fork
                 5
                     6 FFFFFF00 FF000000
                 3
## 3 Spoon
                    4 <NA> FFFF0000
                 1
                       2 FFFFFF00
                                     FFFF0000
## 4 Teaspoon
Here's the same thing, but using only tidyxl and unpivotr.
fill_colours <- xlsx_formats(path) $local fill patternFill fgColor rgb
font_colours <- xlsx_formats(path)$local$font$color$rgb</pre>
cells <-
 xlsx_cells(path, sheet = "combined-highlights") %>%
  mutate(fill colour = fill colours[local format id],
        font_colour = font_colours[local_format_id]) %>%
  select(row, col, data_type, character, numeric, fill_colour, font_colour) %>%
  behead("N", header) %>%
  behead("W", Name) %>%
  select(-col, -character)
values <-
  cells %>%
  select(-fill_colour, -font_colour) %>%
  spread(header, numeric)
formats <- distinct(cells, row, fill_colour, font_colour)</pre>
left_join(values, formats, by = "row") %>%
select(-row)
## # A tibble: 4 x 6
    ##
                    <dbl> <dbl> <chr>
    <chr> <chr>
                                              <chr>
##
                       8
                                              FF000000
## 1 numeric Knife
                               7 <NA>
## 2 numeric Fork
                         6
                               5 FFFFFF00 <NA>
## 3 numeric Spoon
                        4
                               3 <NA>
                                           FFFF0000
```

4 numeric Teaspoon 2 1 FFFFFF00 FFFF0000

2.6 Hierarchies in formatting

Α	В	
Name	Score	
Matilda	7	
Nicholas	5	
Olivia	3	
Paul	1	
	Name Matilda <i>Nicholas</i> Olivia	Name Score Matilda 7 Nicholas 5 Olivia 3

Different kinds of formatting might also represent different levels of a hierarchy, e.g.

formatting	interpretation
none italic bold	good satisfactory poor
bold & italic	fail

When each kind of formatting relates to a different level of one hierarchy, import the different kinds of formatting into different columns, and then combine them into a third column, perhaps using paste(), or case_when().

```
# Step 1: import the table taking only cell values and ignoring the formatting
x <- read_excel(path, sheet = "highlight-hierarchy")</pre>
## # A tibble: 4 x 2
##
    Name
             Score
##
     <chr>
## 1 Matilda
## 2 Nicholas
                  5
                  3
## 3 Olivia
## 4 Paul
                  1
# Step 2: import one kind of formatting of one column of the table
# `formats` is a pallette of fill colours that can be indexed by the
# `local_format_id` of a given cell to get the fill colour of that cell
bold <- xlsx_formats(path)$local$font$bold</pre>
italic <- xlsx_formats(path)$local$font$italic</pre>
# Import all the cells, filter out the header row, filter for the first column,
```

Here it is again, using only tidyxl and unpivotr.

```
## # A tibble: 4 x 6
     row bold italic grade
                                       Score
                               Name
## <int> <lgl> <lgl> <chr>
                               <chr>
                                       <dbl>
## 1
      2 FALSE FALSE good
                              Matilda
## 2
      3 FALSE TRUE satisfactory Nicholas
                                          5
## 3
      4 TRUE FALSE poor Olivia
      5 TRUE TRUE fail
## 4
                              Paul
                                          1
```

2.7 Sentinel values in non-text columns

	Α	В	С	
1	Name	Subject	Score	
2	Matilda	Music		7
3	Nicholas	Classics	NA	
4	Olivia			3
5	Paul	NA	C	

R packages like readr recognise NA as a sentinel value that means "Not Applicable", or "Not Available", or anything you want. It doesn't affect the data type of a column when NA is one of the values. Some datasets use other symbols as a sentinel value, e.g. N/A or ., or a combination, in which case you can instruct readr to interpret those values as sentinels, and it will import them all as NA.

But what if the data uses more than one kind of sentinel value. For example, Statistics New Zealand uses ... to mean "Not applicable", and ..C to mean "Confidentialised". Most tools will either regard both values as NA, or coerce the whole column to characters.

```
read csv("a, b,
          1, 2,
                  3
          4, ..., ..C",
         na = c("...", "..C")) # Regard both values as NA
## # A tibble: 2 x 3
##
         a
               b
##
     <dbl> <dbl> <dbl>
## 1
               2
## 2
         4
              NA
                    NA
read_csv("a, b,
                  С
          1, 2,
                  3
          4, ..., ..C",
         na = "")
                                # Coerce the whole column to characters
## # A tibble: 2 x 3
         a b
     <dbl> <chr> <chr>
         1 2
## 1
## 2
                   ..C
```

A better procedure is to import the sentinel values into their own column, or even into separate TRUE/FALSE columns for each kind of sentinel.

Note that sentinel values relate the value in the cell, rather than to the whole row, so the first step is to make the dataset *extra-tidy* as in the section "Already a tidy table but with meaningful formatting of single cells".

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
x <- read_excel(path, sheet = "sentinels")</pre>
## # A tibble: 4 x 3
##
    Name Subject Score
    <chr>
##
             <chr>
                       <chr>>
## 1 Matilda Music
## 2 Nicholas Classics NA
## 3 Olivia ... 3
## 4 Paul
             NΑ
                       ..C
# Extra-tidy
extra_tidy <-
 gather(x, variable, value, -Name) %>%
 arrange(Name, variable)
extra_tidy
## # A tibble: 8 x 3
##
    Name
           variable value
##
    <chr>
             <chr> <chr>
## 1 Matilda Score
## 2 Matilda Subject Music
## 3 Nicholas Score
                      NA
## 4 Nicholas Subject Classics
## 5 Olivia Score
                       3
## 6 Olivia Subject ...
## 7 Paul
             Score
                       ..C
## 8 Paul
              Subject NA
With an extra-tidy dataset, the sentinels can now be appended to the values of individual variables, rather
than to whole observations.
# Extra-tidy, with row and column numbers of the original variables, and the
# sentinels omitted
extra_tidy <-
 read_excel(path, sheet = "sentinels", na = c("NA", "...", "...")) %>%
 mutate(row = row_number() + 1L) %>%
 gather(variable, value, -row, -Name) %>%
 group_by(row) %>%
```

```
extra_tidy
## # A tibble: 8 x 5
##
     row col Name
                      variable value
  <int> <int> <chr>
##
                      <chr>
                              <chr>>
## 1 2 2 Matilda Subject Music
## 2
      2
           3 Matilda Score
## 3
       3
           2 Nicholas Subject Classics
## 4
      3 3 Nicholas Score <NA>
## 5
      4 2 Olivia Subject <NA>
## 6
           3 Olivia Score
      4
```

mutate(col = row_number() + 1L) %>%

select(row, col, Name, variable, value) %>%

ungroup() %>%

arrange(row, col)

```
## 7
         5
               2 Paul
                          Subject
                                   <NA>
## 8
         5
               3 Paul
                          Score
                                    <NA>
# Import all the cells, and filter for sentinel values
sentinels <-
 xlsx cells(path, sheet = "sentinels") %>%
  dplyr::filter(character %in% c("NA", "...", "...C")) %>%
 mutate(sentinel = character) %>%
  select(row, col, sentinel)
sentinels
## # A tibble: 4 x 3
       row col sentinel
##
##
     <int> <int> <chr>
## 1
         3
               3 NA
## 2
         4
               2 ...
## 3
         5
               2 NA
## 4
         5
               3 ..C
# Join the `sentinel` column to the rest of the data
left_join(extra_tidy, sentinels, by = c("row", "col"))
## # A tibble: 8 x 6
##
      row col Name
                          variable value
                                             sentinel
     <int> <int> <chr>
##
                          <chr>
                                    <chr>>
                                             <chr>>
              2 Matilda Subject Music
                                             <NA>
## 2
         2
               3 Matilda Score
                                             <NA>
## 3
         3
               2 Nicholas Subject Classics <NA>
               3 Nicholas Score
## 4
         3
                                    <NA>
## 5
         4
               2 Olivia
                          Subject <NA>
## 6
         4
               3 Olivia
                          Score
                                    3
                                             <NA>
## 7
         5
               2 Paul
                          Subject
                                   <NA>
                                             NA
## 8
               3 Paul
                          Score
                                    <NA>
                                             ..C
Here's another version using only tidyxl and unpivotr, which provides isolate_sentinels() to make this
much more straightforward.
xlsx_cells(path, sheet = "sentinels") %>%
  select(row, col, data_type, character, numeric) %>%
  isolate_sentinels(character, c("NA", "...", "...C")) %>%
  behead("W", Name) %>%
  behead("N", variable) %>%
  select(Name, variable, character, numeric, sentinel)
## # A tibble: 8 x 5
##
    Name
              variable character numeric sentinel
     <chr>>
              <chr>
                       <chr>
                                   <dbl> <chr>
## 1 Matilda Subject Music
                                      NA <NA>
## 2 Matilda Score
                       <NA>
                                       7 <NA>
## 3 Nicholas Subject Classics
                                      NA <NA>
## 4 Nicholas Score
                       <NA>
                                      NA NA
## 5 Olivia
              Subject
                       <NA>
                                      NA ...
## 6 Olivia
              Score
                       < NA >
                                       3 <NA>
## 7 Paul
              Subject
                       <NA>
                                      NA NA
## 8 Paul
                       <NA>
                                      NA ..C
              Score
```

Chapter 3

Pivot tables

	А	В	С	D	Е	F	G
1							
2				Female		Male	
3				Matilda	Olivia	Nicholas	Paul
4		Humanities	Classics	1	2	3	0
5			History	3	4	5	1
6		Performano	Music	5	6	9	2
7			Drama	7	8	12	3

This part introduces pivot tables. Tidyxl and unpivotr come into their own here, and are (as far as I know) the only packages to acknowledge the intuitive grammar of pivot tables.

Pivot tables are ones with more than one row of column headers, or more than one column of row headers, or both (and there can be more complex arrangements). Tables in that form take up less space on a page or a screen than 'tidy' tables, and are easier for humans to read. But most software can't interpret or traverse data in that form; it must first be reshaped into a long, 'tidy' form, with a single row of column headers.

It takes a lot of code to reshape a pivot table into a 'tidy' one, and the code has to be bespoke for each table. There's no general solution, because it is ambiguous whether a given cell is part of a header or part of the data

There are some ambiguities in 'tidy' tables, too, which is why most functions for reading csv files allow you to specify whether the first row of the data is a header, and how many rows to skip before the data begins. Functions often guess, but they can never be certain.

Pivot tables, being more complex, are so much more ambiguous that it isn't reasonable to import them with a single function. A better way is to break the problem down into steps:

- 1. Identify which cells are headers, and which are data.
- 2. State how the data cells relate to the header cells.

The first step is a matter of traversing the cells, which is *much easier* if you load them with the tidyxl package, or pass the table through as_cells() in the unpivotr package. This gives you a table of cells and their properties; one row of the table describes one cell of the source table or spreadsheet. The first two properties are the row and column position of the cell, which makes it easy to filter for cells in a particular region of the spreadsheet. If the first row of cells is a header row, then you can filter for row == 1.

Here is an example of a pivot table where the first two rows, and the first two columns, are headers. The other cells contain the data. First, see how the cells are laid out in the source file by importing it with readxl.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
original <- read_excel(path, sheet = "pivot-annotations", col_names = FALSE)
print(original, n = Inf)</pre>
```

```
## # A tibble: 6 x 6
     X 1
##
                 X 2
                           X__3
                                    X__4
                                           X__5
                                                     X__6
##
     <chr>>
                  <chr>
                           <chr>>
                                    <chr>
                                           <chr>
                                                     <chr>
## 1 <NA>
                  <NA>
                           Female <NA>
                                           Male
                                                     <NA>
## 2 <NA>
                  <NA>
                           Matilda Olivia Nicholas Paul
                                    2
## 3 Humanities Classics 1
                                           3
                                                     0
## 4 <NA>
                           3
                                    4
                                           5
                                                     1
                  History
                                                     2
## 5 Performance Music
                           5
                                    6
                                           9
## 6 <NA>
                           7
                                                     3
                  Drama
                                           12
```

Compare that with the long set of cells, one per row, that tidyxl gives. (Only a few properties of each cell are shown, to make it easier to read).

```
cells <- xlsx_cells(path, sheets = "pivot-annotations")
select(cells, row, col, data_type, character, numeric) %>%
    print(cells, n = 20)
```

```
## # A tibble: 32 x 5
##
        row
              col data_type character numeric
##
      <int> <int> <chr>
                             <chr>
                                           <dbl>
##
   1
          2
                4 character Female
##
    2
          2
                5 blank
                             <NA>
                                              NΑ
##
    3
          2
                6 character Male
                                              NA
    4
          2
                7 blank
                                              NA
##
                             <NA>
##
   5
          3
                4 character Matilda
                                              NA
##
    6
          3
                5 character Olivia
                                              NA
##
    7
          3
                6 character Nicholas
                                              NA
##
   8
          3
                7 character Paul
                                              NA
   9
##
          4
                2 character Humanities
                                              NA
                3 character Classics
## 10
          4
                                              NA
## 11
          4
                4 numeric
                             <NA>
                                               1
                                               2
## 12
                5 numeric
                             <NA>
## 13
          4
                6 numeric
                             <NA>
                                               3
                                               0
## 14
          4
                7 numeric
                             <NA>
## 15
          5
                2 blank
                             <NA>
                                              NA
## 16
          5
                3 character History
                                              NA
## 17
                4 numeric
                                               3
          5
                             <NA>
## 18
          5
                5 numeric
                             <NA>
                                               4
## 19
          5
                             <NA>
                                               5
                6 numeric
## 20
          5
                7 numeric
                              <NA>
                                                1
## # ... with 12 more rows
```

A similar result is obtained via unpivotr::as_cells().

```
original <- read_excel(path, sheet = "pivot-annotations", col_names = FALSE)
as_cells(original) %>%
arrange(row, col) %>%
print(n = 20)
```

```
## # A tibble: 36 x 4
```

```
##
         row
               col data_type chr
##
       <int> <int> <chr>
                                <chr>
##
    1
           1
                  1 chr
                                <NA>
                                <NA>
##
    2
                  2 chr
           1
##
    3
           1
                  3 chr
                                Female
    4
                                <NA>
##
           1
                  4 chr
    5
##
           1
                  5 chr
                                Male
##
    6
           1
                  6 chr
                                <NA>
##
    7
           2
                  1 chr
                                <NA>
           2
##
    8
                  2 chr
                                <NA>
##
    9
           2
                  3 chr
                                Matilda
           2
##
   10
                  4 chr
                                Olivia
           2
##
   11
                  5 chr
                                Nicholas
           2
## 12
                  6 chr
                                Paul
## 13
           3
                  1 chr
                                Humanities
##
  14
           3
                  2 chr
                                Classics
           3
##
  15
                  3 chr
                                1
##
   16
           3
                  4 chr
                                2
##
           3
                                3
  17
                  5 chr
##
   18
           3
                  6 chr
                                0
## 19
           4
                  1 chr
                                <NA>
## 20
           4
                  2 chr
                                History
## # ... with 16 more rows
```

(One difference is that read_excel() has filled in some missing cells with blanks, which as_cells() retains. Another is that read_excel() has coerced all data types to character, whereas xlsx_cells() preserved the original data types.)

The tidyxl version is easier to traverse, because it describes the position of each cell as well as the value. To filter for the first row of headers:

```
dplyr::filter(cells, row == 2, !is_blank) %>%
 select(row, col, character, numeric)
## # A tibble: 2 x 4
##
       row
             col character numeric
     <int> <int> <chr>
                              <dbl>
## 1
         2
               4 Female
                                 NA
## 2
         2
               6 Male
                                 NA
```

Or to filter for cells containing data (in this case, we know that only data cells are numeric)

```
dplyr::filter(cells, data_type == "numeric") %>%
  select(row, col, numeric)
```

```
## # A tibble: 16 x 3
##
                col numeric
         row
##
       <int> <int>
                        <dbl>
##
    1
            4
                   4
                             1
##
    2
            4
                   5
                             2
    3
                   6
##
            4
                             3
##
    4
            4
                   7
                             0
    5
                   4
##
            5
                             3
##
    6
            5
                   5
                             4
##
                   6
    7
            5
                             5
##
    8
            5
                   7
                             1
            6
                   4
##
    9
                             5
```

```
## 10
            6
                   5
                             6
                   6
## 11
            6
                             9
## 12
            6
                   7
                             2
            7
                   4
                             7
## 13
## 14
            7
                   5
                             8
            7
                   6
## 15
                            12
## 16
                             3
```

By identifying the header cells separately from the data cells, and knowing exactly where they are on the sheet, we can associated the data cells with the relevant headers.

To a human it is intuitive that the cells below and to the right of the header Male represent males, and that ones to the right of and below the header Postgraduate qualification represent people with postgraduate qualifications, but it isn't so obvious to the computer. How would the computer know that the header Male doesn't also relate to the column of cells below and to the left, beginning with 2?

This section shows how you can express the relationships between headers and data cells, using the unpivotr package.

3.1 Simple unpivoting

The behead() function takes one level of headers from a pivot table and makes it part of the data. Think of it like tidyr::gather(), except that it works when there is more than one row of headers (or more than one column of row-headers), and it only works on tables that have first come through as_cells() or tidyxl::xlsx_cells().

3.1.1 Two clear rows of text column headers, left-aligned

	А	В	С	D	E	F	G
1							
2				Female		Male	
3				Matilda	Olivia	Nicholas	Paul
4		Humanities	Classics	1	2	3	0
5			History	3	4	5	1
6		Performano	Music	5	6	9	2
7			Drama	7	8	12	3

Here we have a pivot table with two rows of column headers. The first row of headers is left-aligned, so "Female" applies to the first two columns of data, and "Male" applies to the next two. The second row of headers has a header in every column.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = "pivot-annotations") %>%
    dplyr::filter(col >= 4, !is_blank) %>% # Ignore the row headers in this example
    select(row, col, data_type, character, numeric)
all_cells
```

```
## # A tibble: 22 x 5
## row col data_type character numeric
```

```
##
       <int> <int> <chr>
                               <chr>
                                             <dbl>
##
    1
           2
                  4 character Female
                                                NA
                  6 character Male
##
    2
           2
                                                NA
    3
           3
                  4 character Matilda
##
                                                NΑ
##
    4
           3
                 5 character Olivia
                                                NA
    5
##
           3
                 6 character Nicholas
                                                NA
##
    6
           3
                  7 character Paul
                                                NA
##
    7
           4
                  4 numeric
                               <NA>
                                                 1
##
    8
           4
                 5 numeric
                               <NA>
                                                 2
##
    9
                  6 numeric
                               <NA>
                                                 3
##
  10
                  7 numeric
                               <NA>
                                                 0
          with 12 more rows
```

The behead() function takes the 'melted' output of as_cells(), tidyxl::xlsx_cells(), or a previous behead(), and three more arguments to specify how the header cells relate to the data cells.

The outermost header is the top row, "Female" NA "Male" NA. The "Female" and "Male" headers are above and to-the-left-of the data cells. We express this as a compass direction, north-north-west, or "NNW". We also give the headers a name, sex, and say which column of all_cells contains the value of the header cells—it's usually the character column.

```
all_cells %>%
behead("NNW", sex)
```

```
## # A tibble: 20 x 6
##
        row
               col data_type character numeric sex
##
      <int> <int> <chr>
                               <chr>
                                            <dbl> <chr>
                 4 character Matilda
##
    1
           3
                                               NA Female
##
    2
           3
                 5 character Olivia
                                               NA Female
    3
##
           4
                 4 numeric
                               <NA>
                                                 1 Female
##
    4
                 5 numeric
                               <NA>
                                                 2 Female
    5
                                                 3 Female
##
           5
                 4 numeric
                               <NA>
##
    6
           5
                 5 numeric
                               <NA>
                                                 4 Female
##
    7
           6
                               <NA>
                                                 5 Female
                 4 numeric
##
    8
           6
                 5 numeric
                               <NA>
                                                 6 Female
           7
##
    9
                 4 numeric
                               <NA>
                                                 7 Female
           7
##
   10
                 5 numeric
                               <NA>
                                                 8 Female
##
  11
           3
                 6 character Nicholas
                                               NA Male
## 12
           3
                 7 character Paul
                                               NA Male
## 13
           4
                 6 numeric
                               <NA>
                                                 3 Male
##
   14
           4
                 7 numeric
                               <NA>
                                                 0 Male
##
   15
           5
                 6 numeric
                               <NA>
                                                 5 Male
##
   16
           5
                 7 numeric
                               <NA>
                                                 1 Male
##
   17
           6
                 6 numeric
                               <NA>
                                                 9 Male
##
   18
           6
                 7 numeric
                               <NA>
                                                 2 Male
## 19
           7
                 6 numeric
                               <NA>
                                                12 Male
## 20
           7
                 7 numeric
                                                 3 Male
                               <NA>
```

That did half the job. The value 2 in row 4 column 5 is indeed a score of a female. But the value "matilda" in row 3 column 4 isn't a population – it's another header. The next step is to strip that second level of column headers. This time, the compass direction is "N", because the headers are directly above the associated data cells, and we call it name, because it represents names of people.

```
all_cells %>%
  behead("NNW", sex) %>%
  behead("N", `name`)
```

```
## # A tibble: 16 x 7
##
             col data_type character numeric sex
                                                      name
      <int> <int> <chr>
                            <chr> <dbl> <chr> <chr>
##
##
   1
          4
                4 numeric
                            <NA>
                                            1 Female Matilda
##
   2
          4
                5 numeric
                            <NA>
                                             2 Female Olivia
##
   3
                4 numeric <NA>
                                            3 Female Matilda
          5
   4
                                            4 Female Olivia
##
          5
               5 numeric
                            <NA>
                                            5 Female Matilda
##
   5
          6
               4 numeric
                            <NA>
##
   6
          6
                5 numeric
                            <NA>
                                            6 Female Olivia
   7
          7
##
                4 numeric
                            <NA>
                                            7 Female Matilda
##
   8
          7
                5 numeric
                            <NA>
                                            8 Female Olivia
   9
                6 numeric
                                            3 Male
                                                      Nicholas
##
          4
                            < NA >
## 10
          4
                7 numeric
                            <NA>
                                            0 Male
                                                      Paul
                                                      Nicholas
## 11
          5
                6 numeric
                            < NA >
                                            5 Male
## 12
          5
                7 numeric
                                            1 Male
                                                      Paul
                            <NA>
## 13
          6
                6 numeric
                            <NA>
                                            9 Male
                                                      Nicholas
## 14
                                            2 Male
          6
                7 numeric
                            <NA>
                                                      Paul
## 15
          7
                6 numeric
                            <NA>
                                           12 Male
                                                      Nicholas
## 16
                7 numeric
                                            3 Male
          7
                            <NA>
                                                      Paul
```

A final step is a normal clean-up. We drop the row, col and character columns, and we rename the numeric column to score, which is what it represents.

```
all_cells %>%
behead("NNW", sex) %>%
behead("N", `name`) %>%
select(score = numeric, sex, `name`)
```

```
## # A tibble: 16 x 3
##
      score sex
                   name
##
      <dbl> <chr> <chr>
##
          1 Female Matilda
   1
          2 Female Olivia
##
   2
##
   3
          3 Female Matilda
##
   4
          4 Female Olivia
    5
          5 Female Matilda
##
##
    6
          6 Female Olivia
##
   7
          7 Female Matilda
##
          8 Female Olivia
   8
##
   9
          3 Male
                   Nicholas
## 10
          0 Male
                   Paul
## 11
          5 Male
                   Nicholas
## 12
                   Paul
          1 Male
## 13
          9 Male
                   Nicholas
## 14
          2 Male
                   Paul
## 15
         12 Male
                   Nicholas
## 16
          3 Male
                   Paul
```

3.1.2 Two clear rows and columns of text headers, top-aligned and left-aligned

	Α	В	С	D	Е	F	G
1							
2				Female		Male	
3				Matilda	Olivia	Nicholas	Paul
4		Humanities	Classics	1	2	3	0
5			History	3	4	5	1
6		Performano	Music	5	6	9	2
7			Drama	7	8	12	3

There are no new techniques are used, just more compass directions: "W" for headers directly to the left of the data cells, and "WNW" for headers left-and-above the data cells.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
all_cells <-
  xlsx_cells(path, sheets = "pivot-annotations") %>%
  dplyr::filter(!is_blank) %>%
  select(row, col, data_type, character, numeric) %>%
  print()
## # A tibble: 28 x 5
##
       row col data_type character numeric
##
      <int> <int> <chr>
                           <chr>
                                         <dbl>
##
         2
              4 character Female
                                           NA
   1
##
   2
         2
              6 character Male
                                           NA
##
   3
         3
              4 character Matilda
                                           NA
##
   4
         3
              5 character Olivia
                                           NA
##
  5
         3
              6 character Nicholas
                                           NA
##
         3
              7 character Paul
                                           NA
  6
##
  7
         4
               2 character Humanities
                                           NA
               3 character Classics
##
  8
         4
                                           NA
## 9
               4 numeric <NA>
                                            1
## 10
         4
               5 numeric
                           <NA>
                                            2
## # ... with 18 more rows
all_cells %>%
  behead("NNW", sex) %>%
                          # As before
  behead("N", `name`) %>% # As before
  behead("WNW", field) %>% # Left-and-above
  behead("W", subject) %>% # Directly left
  rename(score = numeric) %>%
  select(-row, -col, -character)
```

```
## # A tibble: 16 x 6
##
                                   field
                                              subject
     data_type score sex
                          name
                                              <chr>>
##
     <chr> <dbl> <chr> <chr>
                                   <chr>
   1 numeric
               1 Female Matilda Humanities Classics
##
## 2 numeric
                  2 Female Olivia Humanities Classics
## 3 numeric
                  3 Female Matilda Humanities History
                 4 Female Olivia Humanities History
## 4 numeric
```

```
5 numeric
                   3 Male
                            Nicholas Humanities Classics
##
                   0 Male
##
   6 numeric
                            Paul
                                     Humanities Classics
   7 numeric
##
                   5 Male
                            Nicholas Humanities History
##
   8 numeric
                   1 Male
                            Paul
                                     Humanities History
##
   9 numeric
                   5 Female Matilda Performance Music
                   6 Female Olivia Performance Music
## 10 numeric
                   7 Female Matilda Performance Drama
## 11 numeric
## 12 numeric
                   8 Female Olivia Performance Drama
## 13 numeric
                   9 Male
                            Nicholas Performance Music
## 14 numeric
                   2 Male
                            Paul
                                     Performance Music
## 15 numeric
                   12 Male
                            Nicholas Performance Drama
## 16 numeric
                                     Performance Drama
                   3 Male
                            Paul
```

3.1.3 Multiple rows or columns of headers, with meaningful formatting

	Α	В	С	D	Е	F	G
1							
2				Female		Male	
3				Matilda	Olivia	Nicholas	Paul
4		Humanities	Classics	1	2	3	0
5			History	3	4	5	1
6		Performano	Music	5	6	9	2
7			Drama	7	8	12	3

This is a combination of the previous section with meaningfully formatted rows. The section meaninfully formatted cells doesn't work here, because the unpivoting of multiple rows/columns of headers complicates the relationship between the data and the formatting.

- 1. Unpivot the multiple rows/columns of headers, as above, but keep the row and col of each data cell.
- 2. Collect the row, col and formatting of each data cell.
- 3. Join the data to the formatting by the row and col.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = "pivot-annotations") %>%
    dplyr::filter(!is_blank) %>%
    select(row, col, data_type, character, numeric) %>%
    print()
```

```
## # A tibble: 28 x 5
##
              col data_type character
##
      <int> <int> <chr>
                             <chr>>
                                           <dbl>
##
    1
                 4 character Female
                                               NA
##
    2
          2
                 6 character Male
                                               NA
    3
                 4 character Matilda
                                              NA
                5 character Olivia
##
    4
          3
                                              NΑ
##
    5
          3
                 6 character Nicholas
                                              NA
##
    6
          3
                7 character Paul
                                              NA
   7
                 2 character Humanities
                                              NA
##
    8
                 3 character Classics
                                              NA
```

```
4
               4 numeric
                           <NA>
## 10
         4
               5 numeric
                           <NA>
## # ... with 18 more rows
unpivoted <-
 all_cells %>%
 behead("NNW", sex) %>%
                          # As before
 behead("N", `name`) %>% # As before
 behead("WNW", field) %>% # Left-and-above
 behead("W", subject) %>% # Directly left
 rename(score = numeric) %>%
 select(-character)
                                   # Retain the row and col for now
unpivoted
## # A tibble: 16 x 8
             col data_type score sex
                                                 field
                                                             subject
                                        name
##
      <int> <int> <chr>
                          <dbl> <chr> <chr>
                                                 <chr>
                                                             <chr>
##
   1
               4 numeric
                               1 Female Matilda Humanities Classics
         4
##
  2
         4
               5 numeric
                               2 Female Olivia Humanities Classics
  3
                               3 Female Matilda Humanities History
         5
              4 numeric
## 4
         5
              5 numeric
                             4 Female Olivia Humanities History
                             3 Male Nicholas Humanities Classics
## 5
         4
               6 numeric
##
  6
         4
               7 numeric
                             0 Male Paul
                                                 Humanities Classics
##
  7
               6 numeric
                             5 Male Nicholas Humanities History
                              1 Male
                                      Paul
## 8
         5
               7 numeric
                                                 Humanities History
                               5 Female Matilda Performance Music
##
  9
         6
              4 numeric
## 10
         6
                               6 Female Olivia
              5 numeric
                                                 Performance Music
## 11
         7
              4 numeric
                              7 Female Matilda Performance Drama
                             8 Female Olivia
## 12
         7
              5 numeric
                                                 Performance Drama
                             9 Male Nicholas Performance Music
## 13
         6
              6 numeric
## 14
         6
               7 numeric
                              2 Male Paul
                                                 Performance Music
               6 numeric
## 15
         7
                              12 Male Nicholas Performance Drama
## 16
         7
               7 numeric
                               3 Male
                                       Paul
                                                 Performance Drama
# `formats` is a pallette of fill colours that can be indexed by the
# `local_format_id` of a given cell to get the fill colour of that cell
fill_colours <- xlsx_formats(path)$local$fill$patternFill$fgColor$rgb
fill_colours
##
  [1] NA
                  NA
                             NΑ
                                        NA
                                                   NA
                             "FFFFFF00" "FF92D050" "FFFFFF00" NA
  [7] NA
## [13] NA
                  "FFFFFF00" NA
                                        NA
                                                   NA
## [19] NA
                  NA
                             NA
                                        "FFFFFF00" "FFFFFF00" NA
## [25] NA
                  "FFFFFF00" NA
                                        NA
                                                   NA
## [31] NA
                  NA
                             NA
                                        NA
                                                   NA
                                                              NA
## [37] NA
                  NA
                             NA
                                        NA
                                                   NA
                                                              NA
## [43] NA
                  NA
                             NA
                                        NΑ
                                                   NA
                                                              NA
## [49] NA
                  NA
                                        NA
                                                   NA
## [55] NA
                  NA
                             "FFFFC7CE" NA
                                                   NA
# Import all the cells, filter out the header row, filter for the first column,
# and create a new column `approximate` based on the fill colours, by looking up
# the local_format_id of each cell in the `formats` pallette.
annotations <-
 xlsx cells(path, sheets = "pivot-annotations") %>%
 dplyr::filter(row >= 4, col >= 4) %>% # Omit the headers
```

```
mutate(fill_colour = fill_colours[local_format_id]) %>%
  select(row, col, fill_colour)
annotations
## # A tibble: 16 x 3
##
        row
              col fill_colour
##
      <int> <int> <chr>
##
    1
          4
                 4 <NA>
##
    2
          4
                 5 FFFFFF00
    3
                 6 <NA>
##
          4
##
    4
          4
                 7 <NA>
##
    5
          5
                 4 FFFFFF00
##
    6
          5
                5 <NA>
##
    7
          5
                 6 <NA>
                7 <NA>
##
    8
          5
##
    9
          6
                 4 <NA>
##
  10
          6
                 5 <NA>
##
                 6 <NA>
  11
          6
##
   12
          6
                7 <NA>
##
  13
          7
                 4 <NA>
## 14
          7
                 5 <NA>
          7
## 15
                 6 FFFFFF00
## 16
          7
                 7 <NA>
left_join(unpivoted, annotations, by = c("row", "col")) %>%
  select(-row, -col)
## # A tibble: 16 x 7
##
      data_type score sex
                                        field
                                                     subject
                                                              fill_colour
                              name
##
      <chr>
                 <dbl> <chr>
                              <chr>
                                        <chr>
                                                     <chr>
                                                               <chr>
##
    1 numeric
                     1 Female Matilda Humanities
                                                     Classics <NA>
##
    2 numeric
                     2 Female Olivia
                                        Humanities
                                                     Classics FFFFFF00
##
   3 numeric
                     3 Female Matilda Humanities
                                                    History
                                                              FFFFFF00
##
    4 numeric
                     4 Female Olivia
                                        Humanities
                                                    History
                                                              <NA>
##
                     3 Male
                              Nicholas Humanities
                                                     Classics <NA>
    5 numeric
##
    6 numeric
                     0 Male
                              Paul
                                        Humanities
                                                     Classics <NA>
                     5 Male
##
    7 numeric
                              Nicholas Humanities
                                                     History
                                                              <NA>
##
    8 numeric
                     1 Male
                              Paul
                                        Humanities
                                                    History
                                                              <NA>
##
    9 numeric
                     5 Female Matilda
                                        Performance Music
                                                               <NA>
                     6 Female Olivia
                                        Performance Music
                                                               <NA>
## 10 numeric
## 11 numeric
                     7 Female Matilda
                                        Performance Drama
                                                               <NA>
## 12 numeric
                     8 Female Olivia
                                        Performance Drama
                                                               <NA>
```

3.2 Complex unpivoting

13 numeric

14 numeric

15 numeric

16 numeric

When behead() isn't powerful enough (it makes certain assumptions, and it doesn't understand formatting), then you can get much more control by using enhead(), which joins together two separate data frames of data cells and header cells.

Nicholas Performance Music

Nicholas Performance Drama

Performance Music

Performance Drama

<NA>

<NA>

<NA>

FFFFFF00

This kind of unpivoting is always done in two stages.

9 Male

2 Male

12 Male

3 Male

Paul

Paul

- 1. Identify which cells are headers, and which are data
- 2. State how the data cells relate to the header cells.

3.2.1 Two clear rows of text column headers, left-aligned {#2RL}

	Α	В	С	D	Е	F	G
1							
2				Female		Male	
3				Matilda	Olivia	Nicholas	Paul
4		Humanities	Classics	1	2	3	0
5			History	3	4	5	1
6		Performano	Music	5	6	9	2
7			Drama	7	8	12	3

The first stage, identifying header vs data cells, is simply filtering.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = "pivot-annotations") %>%
    dplyr::filter(col >= 4, !is_blank) %>% # Ignore the row headers in this example
    select(row, col, data_type, character, numeric) %>%
    print()
```

```
## # A tibble: 22 x 5
##
       row col data_type character numeric
##
      <int> <int> <chr>
                           <chr>
                                       <dbl>
##
         2
              4 character Female
                                          NA
   1
##
   2
         2
               6 character Male
                                          NA
##
               4 character Matilda
   3
         3
                                          NA
##
   4
         3
              5 character Olivia
                                          NΑ
##
   5
         3
              6 character Nicholas
##
   6
         3
              7 character Paul
                                          NA
   7
##
         4
              4 numeric
                           <NA>
                                           1
##
   8
         4
               5 numeric
                          <NA>
                                           2
  9
                           <NA>
                                           3
##
               6 numeric
## 10
         4
               7 numeric
                           <NA>
                                           0
## # ... with 12 more rows
```

View the cells in their original positions on the spreadsheet
rectify(all_cells)

```
## # A tibble: 6 x 5
##
     `row/col` `4(D)`
                        `5(E)` `6(F)`
                                         `7(G)`
##
         <int> <chr>
                        <chr> <chr>
                                         <chr>>
## 1
             2 Female <NA>
                               Male
                                         <NA>
             3 Matilda Olivia Nicholas Paul
## 2
## 3
             4 1
                        2
                               3
## 4
             5 3
                        4
                               5
                                         1
## 5
             6 5
                        6
                               9
                                         2
             7 7
                        8
                                         3
## 6
                               12
```

8

9

5

6

7

1 Paul

4 5 Matilda

```
first_header_row <-
 dplyr::filter(all_cells, row == 2) %>%
 select(row, col, sex = character)
 # the title of this header is 'sex'
 # the cells are text cells (`"Female"` and `"Male"`) so take the value in the
  # '`character` column.
first_header_row
## # A tibble: 2 x 3
##
      row col sex
    <int> <int> <chr>
## 1
      2 4 Female
## 2
        2
              6 Male
second header row <-
 dplyr::filter(all_cells, row == 3) %>%
 select(row, col, name = character)
 # The title of this header is 'name'.
  # The cells are text cells, so take the value in the '`character` column.
second_header_row
## # A tibble: 4 x 3
      row col name
    <int> <int> <chr>
##
## 1
      3 4 Matilda
## 2
       3 5 Olivia
## 3
        3 6 Nicholas
## 4
        3
              7 Paul
data_cells <-
 dplyr::filter(all_cells, data_type == "numeric") %>%
 select(row, col, score = numeric)
 # The data is exam scores in certain subjects, so give the data that title.
 # The data is numeric, so select only that 'value'. If some of the data was
 # also text or true/false, then you would select the `character` and `logical`
# columns as well as `numeric`
```

The second stage is to declare how the data cells relate to each row of column headers. Unpivotr provides a set of functions for this, derived from the points of the compass.

Starting from the point of view of a data cell, the relevant column header from the second row of headers is the one directly north (up), or "N".

```
enhead(data_cells, second_header_row, "N")
## # A tibble: 16 x 4
##
      row col score name
##
     <int> <int> <dbl> <chr>
## 1
        4
            4
                 1 Matilda
## 2
        4
             5
                  2 Olivia
## 3
        4
            6
                 3 Nicholas
## 4
      4
            7
                 0 Paul
## 5
            4
                 3 Matilda
      5
## 6
       5
            5
                 4 Olivia
## 7
      5
            6 5 Nicholas
```

```
## 10
           6
                 5
                        6 Olivia
## 11
                 6
                        9 Nicholas
           6
## 12
                 7
                        2 Paul
          7
## 13
                 4
                        7 Matilda
## 14
          7
                 5
                        8 Olivia
## 15
          7
                 6
                       12 Nicholas
## 16
                        3 Paul
```

The first row of headers, from the point of view of a data cell, is either directly north (up), or north and west (up and left), or "NNW".

```
enhead(data_cells, first_header_row, "NNW")
```

```
## # A tibble: 16 x 4
##
        row
               col score sex
      <int> <int> <dbl> <chr>
##
##
    1
                 4
                        1 Female
##
    2
           4
                 5
                        2 Female
##
    3
           5
                 4
                        3 Female
##
    4
           5
                 5
                        4 Female
    5
##
           6
                 4
                        5 Female
    6
                 5
##
           6
                        6 Female
    7
##
           7
                 4
                        7 Female
##
    8
           7
                 5
                        8 Female
##
    9
                 6
                        3 Male
                 7
## 10
           4
                        0 Male
## 11
           5
                 6
                        5 Male
## 12
                 7
           5
                        1 Male
## 13
           6
                 6
                        9 Male
                 7
## 14
           6
                        2 Male
## 15
           7
                 6
                       12 Male
## 16
                 7
                        3 Male
```

Piping everything together, we get a complete, tidy dataset, and can finally drop the row and col columns.

```
data_cells %>%
  enhead(first_header_row, "NNW") %>%
  enhead(second_header_row, "N") %>%
  select(-row, -col)
```

```
## # A tibble: 16 x 3
##
      score sex
                   name
##
      <dbl> <chr> <chr>
##
   1
          1 Female Matilda
    2
          2 Female Olivia
##
##
    3
          3 Female Matilda
##
    4
          4 Female Olivia
##
    5
          5 Female Matilda
##
    6
          6 Female Olivia
   7
##
          7 Female Matilda
   8
          8 Female Olivia
   9
                   Nicholas
##
          3 Male
## 10
          0 Male
                   Paul
## 11
          5 Male
                   Nicholas
## 12
          1 Male
                   Paul
## 13
          9 Male
                   Nicholas
```

```
## 14 2 Male Paul
## 15 12 Male Nicholas
## 16 3 Male Paul
```

3.2.2 Two clear columns of text row headers, top-aligned

1 Female Male 3 Matilda Olivia Nicholas Paul	
3 Matilda Olivia Nicholas Paul	
4 Humanities Classics 1 2 3	0
5 History 3 4 5	1
6 Performanc Music 5 6 9	2
7 Drama 7 8 12	3

This is almost the same as Two clear rows of text column headers, left-aligned, but with different compass directions: "W" for directly west (left), and "WNW" for west and north (left and up).

("NNW" and "WNW" look like synonyms. They happen to be synonyms in enhead(), but they aren't in behead().

In this example, the table has no column headers, only row headers. This is artificial here, but sometimes table are deliberately laid out in transpose form: the first column contains the headers, and the data extends in columns from left to right instead of from top to bottom.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = "pivot-annotations") %>%
    dplyr::filter(row >= 3, !is_blank) %>% # Ignore the column headers in this example
    select(row, col, data_type, character, numeric) %>%
    print()
```

```
## # A tibble: 26 x 5
##
       row col data_type character numeric
##
      <int> <int> <chr>
                            <chr>
                                          <dbl>
##
   1
          3
               4 character Matilda
                                             NA
##
   2
          3
                5 character Olivia
                                             NA
##
   3
          3
                6 character Nicholas
##
   4
          3
               7 character Paul
                                             NA
   5
               2 character Humanities
##
                                             NA
##
   6
          4
                3 character Classics
                                             NA
##
   7
                4 numeric
                            <NA>
                                              1
          4
                                              2
##
   8
                5 numeric
                            <NA>
   9
                6 numeric
                            <NA>
                                              3
                                              0
                7 numeric
                            <NA>
## 10
## # ... with 16 more rows
```

View the cells in their original positions on the spreadsheet
rectify(all_cells)

```
<chr>
                                 <chr> <chr> <chr>
##
        <int> <chr>
         3 <NA>
## 1
                        <NA> Matilda Olivia Nicholas Paul
          4 Humanities Classics 1 2
## 2
## 3
          5 <NA> History 3
                                                       1
                                       6
## 4
           6 Performance Music 5
                                               9
                                                        2
                                    8 12
## 5
           7 <NA>
                        Drama 7
first_header_col <-
 dplyr::filter(all_cells, col == 2) %>%
 select(row, col, field = character)
 # the title of this header is 'field', meaning 'group of subjects'.
  # The cells are text cells ("Humanities", ""Performance") so take the value
  # in the '`character` column.
first_header_col
## # A tibble: 2 x 3
    row col field
## <int> <int> <chr>
## 1 4 2 Humanities
## 2
        6
             2 Performance
second_header_col <-
 dplyr::filter(all_cells, col == 3) %>%
 select(row, col, subject = character)
 # The title of this header is 'subject'
  # The cells are text cells (`"history"`, etc.) so take the value in the
  # '`character` column.
second_header_col
## # A tibble: 4 x 3
## row col subject
## <int> <int> <chr>
## 1
      4 3 Classics
       5 3 History
## 2
## 3
       6 3 Music
## 4
      7
             3 Drama
data_cells <-
 dplyr::filter(all_cells, data_type == "numeric") %>%
 select(row, col, score = numeric)
 # The data is examp scores in certain subjects, so give the data that title.
  # The data is numeric, so select only that 'value'. If some of the data was
  # also text or true/false, then you would select the `character` and `logical`
  # columns as well as `numeric`
data_cells %>%
  enhead(first_header_col, "WNW") %>%
  enhead(second_header_col, "W") %>%
 select(-row, -col)
## # A tibble: 16 x 3
##
   score field
                      subject
##
     <dbl> <chr>
                      <chr>
## 1 1 Humanities Classics
## 2
       2 Humanities Classics
## 3 3 Humanities Classics
```

```
##
          O Humanities Classics
         3 Humanities History
##
   5
##
         4 Humanities History
##
   7
          5 Humanities History
##
         1 Humanities History
  9
         5 Performance Music
##
          6 Performance Music
## 10
## 11
         9 Performance Music
## 12
         2 Performance Music
## 13
         7 Performance Drama
## 14
         8 Performance Drama
         12 Performance Drama
## 15
## 16
          3 Performance Drama
```

3.2.3 Two clear rows and columns of text headers, top-aligned and left-aligned

	А	В	С	D	E	F	G
1							
2				Female		Male	
3				Matilda	Olivia	Nicholas	Paul
4		Humanities	Classics	1	2	3	0
5			History	3	4	5	1
6		Performano	Music	5	6	9	2
7			Drama	7	8	12	3

This is a combination of the previous two sections. No new techniques are used.

- 1. Identify which cells are headers, and which are data
- 2. State how the data cells relate to the header cells.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = "pivot-annotations") %>%
    dplyr::filter(!is_blank) %>%
    select(row, col, data_type, character, numeric) %>%
    print()
```

```
## # A tibble: 28 x 5
##
       row col data_type character numeric
##
      <int> <int> <chr>
                            <chr>
                                         <dbl>
##
   1
               4 character Female
                                            NA
##
          2
                6 character Male
                                            NA
   2
##
          3
               4 character Matilda
                                            NA
##
   4
          3
               5 character Olivia
                                            NA
   5
          3
               6 character Nicholas
               7 character Paul
##
   6
          3
                                            NA
##
   7
               2 character Humanities
##
   8
               3 character Classics
                                            NA
   9
               4 numeric <NA>
                                            1
                                             2
## 10
               5 numeric
                            <NA>
```

```
## # ... with 18 more rows
\# View the cells in their original positions on the spreadsheet
rectify(all cells)
## # A tibble: 6 x 7
   `row/col` `2(B)`
                        `3(C)`
                                 `4(D)` `5(E)` `6(F)`
                                                       `7(G)`
##
      <int> <chr>
                       <chr>
                                 <chr> <chr> <chr>
                                                        <chr>>
## 1
          2 <NA>
                       <NA>
                                 Female <NA> Male
                                                        <NA>
## 2
          3 <NA>
                         <NA>
                                 Matilda Olivia Nicholas Paul
## 3
          4 Humanities Classics 1
                                    2
                                               3
                        History 3
## 4
          5 <NA>
                                              5
                                                      1
## 5
          6 Performance Music
                                 5
                                       6
                                              9
                                                      2
                                    8 12
                                                   3
           7 <NA>
                                 7
## 6
                         Drama
first_header_row <-
 dplyr::filter(all_cells, row == 2) %>%
  select(row, col, sex = character)
 # the title of this header is 'sex'
 # the cells are text cells (`"Female"` and `"Male"`) so take the value in the
  # '`character` column.
first_header_row
## # A tibble: 2 x 3
    row col sex
## <int> <int> <chr>
      2 4 Female
## 1
## 2
        2
             6 Male
second header row <-
 dplyr::filter(all_cells, row == 3) %>%
 select(row, col, name = character)
 # The title of this header is 'name'.
  # The cells are text cells, so take the value in the '`character` column.
second_header_row
## # A tibble: 4 x 3
    row col name
## <int> <int> <chr>
      3 4 Matilda
## 1
## 2
        3
             5 Olivia
## 3
        3 6 Nicholas
## 4
        3
             7 Paul
first_header_col <-
 dplyr::filter(all_cells, col == 2) %>%
  select(row, col, field = character)
  # the title of this header is 'field', meaning 'group of subjects'.
  # The cells are text cells ("Humanities"', ""Performance"') so take the value
  # in the '`character` column.
first_header_col
## # A tibble: 2 x 3
##
    row col field
## <int> <int> <chr>
## 1 4 2 Humanities
      6 2 Performance
## 2
```

```
second_header_col <-</pre>
 dplyr::filter(all_cells, col == 3) %>%
 select(row, col, subject = character)
 # The title of this header is 'subject'
 # The cells are text cells (`"history", etc.) so take the value in the
 # '`character` column.
second_header_col
## # A tibble: 4 x 3
## row col subject
   <int> <int> <chr>
## 1
       4
           3 Classics
## 2
       5
           3 History
## 3
        6
             3 Music
     7
## 4
              3 Drama
data_cells <-
 dplyr::filter(all_cells, data_type == "numeric") %>%
 select(row, col, score = numeric)
 # The data is examp scores in certain subjects, so give the data that title.
 # The data is numeric, so select only that 'value'. If some of the data was
 # also text or true/false, then you would select the `character` and `logical`
 # columns as well as `numeric`
data_cells %>%
 enhead(first_header_row, "NNW") %>%
 enhead(second_header_row, "N") %>%
 enhead(first_header_col, "WNW") %>%
 enhead(second_header_col, "W") %>%
```

```
## # A tibble: 16 x 5
##
     score sex name
                      field
                                  subject
     <dbl> <chr> <chr> <chr>
                                   <chr>
        1 Female Matilda Humanities Classics
## 1
        2 Female Olivia Humanities Classics
## 2
## 3
       3 Female Matilda Humanities History
       4 Female Olivia Humanities History
## 5
       3 Male Nicholas Humanities Classics
## 6
       O Male Paul Humanities Classics
## 7
       5 Male Nicholas Humanities History
## 8
       1 Male Paul Humanities History
## 9
        5 Female Matilda Performance Music
## 10
        6 Female Olivia Performance Music
## 11
       7 Female Matilda Performance Drama
## 12
       8 Female Olivia Performance Drama
       9 Male Nicholas Performance Music
## 13
## 14
       2 Male Paul Performance Music
## 15 12 Male Nicholas Performance Drama
       3 Male Paul Performance Drama
## 16
```

select(-row, -col)

11

3

1

12

	Α	В	С	D	Е	F	G	Н	I	J
1										
2					Female				Male	
3				Leah	Matilda	Olivia	Lenny	Max	Nicholas	Paul
4			Classics	3	1	2	4	3	3	0
5		Humanities	History	8	3	4	7	5	5	1
6			Literature	1	1	9	3	12	7	5
7			Philosophy	5	10	10	8	2	5	12
8			Languages	5	4	5	9	8	3	8
9			Music	4	10	10	2	4	5	6
10		Performanc	Dance	4	5	6	4	12	9	2

3.2.4 Centre-aligned headers

Headers aren't always aligned to one side of the data cells that they describe.

Drama

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <- xlsx_cells(path, sheets = "pivot-centre-aligned")
rectify(all_cells)</pre>
```

```
##
   # A tibble: 10 x 10
##
       `row/col` `2(B)`
                              `3(C)`
                                                `5(E)`
                                                        `6(F)` `7(G)`
                                                                        `8(H)`
                                                                                 `9(I)`
##
           <int> <chr>
                              <chr>>
                                                <chr>
                                                        <chr>>
                                                                 <chr>
                                                                         <chr>>
                                                                                 <chr>
                                                                                Male
##
                2 <NA>
                              <NA>
                                                Female <NA>
                                                                 <NA>
                                                                         <NA>
    1
                                        <NA>
##
    2
                3 <NA>
                              <NA>
                                        Leah
                                                Matil~ Olivia Lenny
                                                                        Max
                                                                                 Nichol~
                4 <NA>
                                                                                 3
##
    3
                              Classics 3
                                                1
                                                        2
                                                                4
                                                                         3
##
    4
                5 Humaniti~ History
                                        8
                                                3
                                                        4
                                                                7
                                                                         5
                                                                                 5
    5
                                                                3
                                                                         12
                                                                                 7
##
                6 <NA>
                              Literat~ 1
                                                1
                                                        9
                              Philoso~ 5
                                                                         2
##
    6
                7 <NA>
                                                10
                                                        10
                                                                8
                                                                                 5
    7
                8 <NA>
                                                                9
                                                                         8
                                                                                 3
##
                              Languag~ 5
                                                4
                                                        5
                9 <NA>
                                                10
                                                                2
                                                                                 5
##
                              Music
                                                        10
                                                                         4
##
    9
               10 Performa~
                             Dance
                                        4
                                                5
                                                        6
                                                                 4
                                                                         12
                                                                                 9
  10
               11 <NA>
                              Drama
                                        2
                                                7
                                                                         1
                                                                                 12
          with 1 more variable: `10(J)`
                                             <chr>
```

Looking at that table, it's not immediately obvious where the boundary between Female and Male falls, or between Humanities and Performance. A naive approach would be to match the inner headers to the outer ones by proximity, and there are four directions to do so: "ABOVE", "LEFT", "BELOW", and "RIGHT".

But in this case, those directions are too naive.

- Languages is closest to the Performance header, but is a humanity.
- Lenny is the same distance from Female as from Male.

You can fix this by justifying the header cells towards one side of the data cells that they describe, and then use a direction like "NNW" as usual. Do this with justify(), providing the header cells with a second set of cells at the positions you want the header cells to move to.

- header_cells is the cells whose value will be used as the header
- corner_cells is the cells whose position is in one corner of the domain of the header (e.g. the top-left-hand corner).

In the original spreadsheet, the borders mark the boundaries. So the corner cells of the headers can be found by filtering for cells with a particular border.

```
all_cells <-
  xlsx_cells(path, sheets = "pivot-centre-aligned") %>%
  select(row, col, is_blank, data_type, character, numeric, local_format_id)
formats <- xlsx_formats(path)</pre>
top_borders <- which(!is.na(formats$local$border$top$style))</pre>
left_borders <- which(!is.na(formats$local$border$left$style))</pre>
first_header_row_corners <-
  dplyr::filter(all_cells, row == 2, local_format_id %in% left_borders) %>%
  select(row, col)
first_header_row_corners
## # A tibble: 2 x 2
##
      row col
   <int> <int>
##
## 1
       2 4
## 2
        2
               7
first_header_col_corners <-
  dplyr::filter(all_cells, col == 2, local_format_id %in% top_borders) %>%
  select(row, col)
first_header_col_corners
## # A tibble: 2 x 2
##
     row col
## <int> <int>
## 1
       4
               2
## 2
        9
Next, get the first row and first column of header cells as usual.
first_header_row <-
  dplyr::filter(all_cells, !is_blank, row == 2) %>%
  select(row, col, sex = character)
  # the title of this header is 'sex'
  # the cells are text cells (""Female" and ""Male") so take the value in the
  # '`character` column.
first_header_row
## # A tibble: 2 x 3
     row col sex
   <int> <int> <chr>
##
## 1
     2 5 Female
## 2
        2
               9 Male
first_header_col <-
 dplyr::filter(all_cells, !is_blank, col == 2) %>%
  select(row, col, field = character)
  # the title of this header is 'field', meaning 'group of subjects'.
  # The cells are text cells ("Humanities"', ""Performance"') so take the value
  # in the '`character` column.
first_header_col
## # A tibble: 2 x 3
     row col field
##
## <int> <int> <chr>
```

```
## 1
        5
              2 Humanities
## 2
        10
              2 Performance
And now justify the header cells to the same positions as the corner cells.
first_header_row <- justify(first_header_row, first_header_row_corners)</pre>
first_header_col <- justify(first_header_col, first_header_col_corners)</pre>
first_header_row
## # A tibble: 2 x 3
     row col sex
## <int> <int> <chr>
## 1
      2 4 Female
       2
## 2
              7 Male
first_header_col
## # A tibble: 2 x 3
##
     row col field
## <int> <int> <chr>
## 1
      4 2 Humanities
## 2
        9
              2 Performance
The rest of this example is the same as "Two clear rows and columns of text headers, top-aligned and
left-aligned".
second_header_row <-
 dplyr::filter(all_cells, row == 3) %>%
  select(row, col, name = character)
  # The title of this header is 'name'.
  # The cells are text cells, so take the value in the '`character` column.
second_header_row
## # A tibble: 7 x 3
##
      row col name
   <int> <int> <chr>
##
## 1
       3 4 Leah
## 2
       3 5 Matilda
## 3
       3
             6 Olivia
## 4
        3
             7 Lenny
## 5
        3 8 Max
## 6
        3
             9 Nicholas
## 7
        3 10 Paul
second_header_col <-
 dplyr::filter(all_cells, col == 3) %>%
  select(row, col, subject = character)
  # The title of this header is 'subject'
  # The cells are text cells (`"history"`, etc.) so take the value in the
  # '`character` column.
second_header_col
## # A tibble: 8 x 3
##
     row col subject
   <int> <int> <chr>
## 1 4 3 Classics
## 2
       5
             3 History
```

10

5 Female Leah

... with 46 more rows

```
## 3
               3 Literature
## 4
         7
               3 Philosophy
               3 Languages
## 5
## 6
        9
               3 Music
## 7
        10
               3 Dance
## 8
        11
               3 Drama
data_cells <-
  dplyr::filter(all_cells, data_type == "numeric") %>%
  select(row, col, score = numeric)
  # The data is examp scores in certain subjects, so give the data that title.
  # The data is numeric, so select only that 'value'. If some of the data was
  # also text or true/false, then you would select the `character` and `logical`
  # columns as well as `numeric`
data_cells %>%
  enhead(first_header_row, "NNW") %>%
  enhead(second_header_row, "N") %>%
  enhead(first_header_col, "WNW") %>%
  enhead(second_header_col, "W") %>%
  select(-row, -col)
## # A tibble: 56 x 5
##
      score sex
                   name
                           field
                                       subject
##
      <dbl> <chr> <chr>
                           <chr>>
                                       <chr>
##
          3 Female Leah
                           Humanities Classics
   1
##
          1 Female Matilda Humanities Classics
   3
          2 Female Olivia Humanities Classics
##
          8 Female Leah
                           Humanities History
   5
         3 Female Matilda Humanities History
##
##
         4 Female Olivia Humanities History
##
   7
         1 Female Leah
                           Humanities Literature
##
         1 Female Matilda Humanities Literature
          9 Female Olivia Humanities Literature
##
  9
```

3.2.5 Multiple rows or columns of headers, with meaningful formatting

Humanities Philosophy

	А	В	С	D	Е	F	G
1							
2				Female		Male	
3				Matilda	Olivia	Nicholas	Paul
4		Humanities	Classics	1	2	3	0
5			History	3	4	5	1
6		Performano	Music	5	6	9	2
7			Drama	7	8	12	3

This is a combination of the previous section with Meaningfully formatted cells. The section Meaningfully formatted rows doesn't work here, because the unpivoting of multiple rows/columns of headers complicates

the relationship between the data and the formatting.

- 1. Unpivot the multiple rows/columns of headers, as above, but keep the row and col of each data cell.
- 2. Collect the row, col and formatting of each data cell.
- 3. Join the data to the formatting by the row and col.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
all cells <-
  xlsx_cells(path, sheets = "pivot-annotations") %>%
  dplyr::filter(!is_blank) %>%
  select(row, col, data_type, character, numeric) %>%
 print()
## # A tibble: 28 x 5
##
        row
             col data_type character
                                       numeric
##
      <int> <int> <chr>
                             <chr>>
                                          <dbl>
##
    1
          2
                4 character Female
                                             NA
##
   2
          2
                                             NA
                6 character Male
##
   3
          3
                4 character Matilda
                                             NA
##
   4
          3
                5 character Olivia
                                             NΑ
##
   5
          3
                6 character Nicholas
                                             NA
##
          3
                7 character Paul
  6
                                             NA
##
   7
          4
               2 character Humanities
                                             NΑ
##
   8
                3 character Classics
                                             NA
##
  9
          4
                4 numeric
                                              1
                             <NA>
## 10
          4
                5 numeric
                             < NA >
                                              2
## # ... with 18 more rows
# View the cells in their original positions on the spreadsheet
rectify(all_cells)
## # A tibble: 6 x 7
     `row/col` `2(B)`
                            `3(C)`
                                     `4(D)`
                                             `5(E)` `6(F)`
                                                              `7(G)`
##
         <int> <chr>
                                     <chr>
                                             <chr>
##
                            <chr>
                                                    <chr>
                                                              <chr>>
## 1
             2 <NA>
                            <NA>
                                     Female <NA>
                                                     Male
                                                              <NA>
             3 <NA>
                            <NA>
## 2
                                     Matilda Olivia Nicholas Paul
                                             2
## 3
             4 Humanities Classics 1
                                                     3
                                                              0
## 4
             5 <NA>
                            History
                                     3
                                             4
                                                     5
                                                              1
## 5
                                     5
                                             6
                                                     9
                                                              2
             6 Performance Music
## 6
             7 <NA>
                            Drama
                                     7
                                             8
                                                     12
                                                              3
first header row <-
  dplyr::filter(all_cells, row == 2) %>%
  select(row, col, sex = character)
  # the title of this header is 'sex'
  # the cells are text cells (`"Female"` and `"Male"`) so take the value in the
  # '`character` column.
first_header_row
## # A tibble: 2 x 3
##
       row
           col sex
##
     <int> <int> <chr>
## 1
         2
               4 Female
## 2
         2
               6 Male
second header row <-
 dplyr::filter(all_cells, row == 3) %>%
```

```
select(row, col, name = character)
 # The title of this header is 'name'.
  # The cells are text cells, so take the value in the '`character` column.
second_header_row
## # A tibble: 4 x 3
##
      row col name
## <int> <int> <chr>
       3 4 Matilda
## 1
## 2
       3 5 Olivia
## 3
       3 6 Nicholas
## 4
       3
             7 Paul
first_header_col <-
 dplyr::filter(all_cells, col == 2) %>%
 select(row, col, field = character)
 # the title of this header is 'field', meaning 'group of subjects'.
  # The cells are text cells (`"Humanities"`, `"Performance"`) so take the value
  # in the '`character` column.
first_header_col
## # A tibble: 2 x 3
     row col field
   <int> <int> <chr>
##
## 1
       4 2 Humanities
## 2
        6
             2 Performance
second header col <-
 dplyr::filter(all_cells, col == 3) %>%
 select(row, col, subject = character)
 # The title of this header is 'subject'
  # The cells are text cells (`"history"', etc.) so take the value in the
  # '`character` column.
second_header_col
## # A tibble: 4 x 3
    row col subject
## <int> <int> <chr>
## 1 4 3 Classics
      5 3 History
## 2
## 3
       6
            3 Music
     7
## 4
              3 Drama
data_cells <-
 dplyr::filter(all_cells, data_type == "numeric") %>%
 select(row, col, score = numeric)
  # The data is exam scores in certain subjects, so give the data that title.
  # The data is numeric, so select only that 'value'. If some of the data was
  # also text or true/false, then you would select the `character` and `logical`
  # columns as well as `numeric`
unpivoted <-
 data_cells %>%
  enhead(first_header_row, "NNW") %>%
 enhead(second_header_row, "N") %>%
 enhead(first_header_col, "WNW") %>%
```

```
enhead(second_header_col, "W")
  # Don't delet the `row` and `col` columns yet, because we need them to join on
  # the formatting
# `formats` is a pallette of fill colours that can be indexed by the
# `local_format_id` of a given cell to get the fill colour of that cell
fill_colours <- xlsx_formats(path)$local$fill$patternFill$fgColor$rgb
# Import all the cells, filter out the header row, filter for the first column,
# and create a new column `approximate` based on the fill colours, by looking up
# the local_format_id of each cell in the `formats` pallette.
annotations <-
  xlsx cells(path, sheets = "pivot-annotations") %>%
 dplyr::filter(row >= 4, col >= 4) % # Omit the headers
 mutate(fill_colour = fill_colours[local_format_id]) %>%
  select(row, col, fill_colour)
annotations
## # A tibble: 16 x 3
##
       row col fill_colour
##
     <int> <int> <chr>
              4 <NA>
##
  1
         4
         4
               5 FFFFFF00
##
   2
## 3
         4
              6 <NA>
## 4
         4
              7 <NA>
## 5
              4 FFFFFF00
         5
## 6
         5
              5 <NA>
## 7
         5
              6 <NA>
## 8
         5
              7 <NA>
## 9
         6
              4 <NA>
## 10
         6
              5 <NA>
## 11
         6
              6 <NA>
              7 <NA>
## 12
         6
## 13
         7
              4 <NA>
## 14
         7
              5 <NA>
## 15
         7
              6 FFFFFF00
## 16
               7 <NA>
         7
left_join(unpivoted, annotations, by = c("row", "col")) %>%
select(-row, -col)
## # A tibble: 16 x 6
##
     score sex name
                          field
                                      subject fill_colour
      <dbl> <chr> <chr>
                          <chr>
##
                                      <chr>
                                               <chr>
##
         1 Female Matilda Humanities Classics <NA>
   1
##
         2 Female Olivia Humanities Classics FFFFFF00
##
         3 Female Matilda Humanities History FFFFFF00
  .3
##
   4
         4 Female Olivia Humanities History <NA>
## 5
         3 Male Nicholas Humanities Classics <NA>
##
  6
        O Male Paul Humanities Classics <NA>
## 7
        5 Male Nicholas Humanities History <NA>
         1 Male Paul Humanities History <NA>
## 8
## 9
         5 Female Matilda Performance Music
                                               <NA>
## 10
         6 Female Olivia Performance Music
                                               <NA>
        7 Female Matilda Performance Drama
## 11
                                               <NA>
```

```
8 Female Olivia Performance Drama
                                                <NA>
## 12
## 13
         9 Male Nicholas Performance Music
                                                <NA>
## 14
         2 Male
                Paul
                         Performance Music
                                               <NA>
        12 Male
                                               FFFFFF00
## 15
                 Nicholas Performance Drama
## 16
         3 Male
                  Paul
                          Performance Drama
                                               <NA>
```

3.2.6 Mixed headers and notes in the same row/column, distinguished by formatting

	· · · · · · · · · · · · · · · · · · ·							
	Α	В	С	D	Е	F	G	
1								
2				Female		Male	0 = absent	
3				Matilda	Olivia	Nicholas	Paul	
4		Humanities	Classics	1	2	3	0	
5		Excl. project work	History	3	4	5	1	
6		Performance	Music	5	6	9	2	
7		Incl. written exam	Drama	7	8	12	3	

This doesn't use any new techniques. The trick is, when selecting a row or column of header cells, to filter out ones that have the 'wrong' formatting (formatting that shows they aren't really headers). In this example, cells with italic or red text aren't headers, even if they are in amongst header cells.

First, identify the IDs of formats that have italic or red text.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
formats <- xlsx_formats(path)</pre>
italic <- which(formats$local$font$italic)</pre>
# For 'red' we can either look for the RGB code for red "FFFF0000"
red <- which(formats$local$font$color$rgb == "FFFF0000")</pre>
red
## [1] 12 13 14 40 41
# Or we can find out what that code is by starting from a cell that we know is
# red.
red_cell_format_id <-</pre>
  xlsx_cells(path, sheets = "pivot-notes") %>%
  dplyr::filter(row == 5, col == 2) \%
  pull(local_format_id)
red_cell_format_id
## [1] 40
red_rgb <- formats$local$font$color$rgb[red_cell_format_id]</pre>
red <- which(formats$local$font$color$rgb == red_rgb)</pre>
red
```

```
## [1] 12 13 14 40 41
```

Now we select the headers, filtering out cells with the format IDs of red or italic cells.

```
all_cells <-
  xlsx_cells(path, sheets = "pivot-notes") %>%
  dplyr::filter(!is blank) %>%
  select(row, col, character, numeric, local_format_id) %>%
 print()
## # A tibble: 31 x 5
       row col character numeric local format id
##
     <int> <int> <chr> <dbl>
                                           <int>
## 1
       2
            4 Female
                              NA
                                               18
## 2
        2 6 Male
                               NA
                                               18
## 3
        2 7 0 = absent
                              NA
                                               39
## 4 3 4 Matilda
## 5 3 5 Olivia
                                               20
                               NA
                               NA
                                               21
## 6
       3 6 Nicholas
                              NA
                                               20
## 7
       3 7 Paul
                              NA
                                               21
       4 2 Humanities
4 3 Classics
## 8
                              NA
                                               18
## 9
                               NA
                                               19
## 10
        4
             4 <NA>
                                1
                                               33
## # ... with 21 more rows
first_header_row <-
 dplyr::filter(all_cells, row == 2, !(local_format_id %in% c(red, italic))) %>%
 select(row, col, sex = character)
 # the title of this header is 'sex'
 # the cells are text cells (`"Female"` and `"Male"`) so take the value in the
  # '`character` column.
first_header_row
## # A tibble: 2 x 3
##
    row col sex
## <int> <int> <chr>
## 1 2 4 Female
## 2
       2
              6 Male
first_header_col <-
 dplyr::filter(all_cells, col == 2, !(local_format_id %in% c(red, italic))) %>%
  select(row, col, qualification = character)
  # the title of this header is 'field', meaning 'group of subjects'.
  # The cells are text cells ("Humanities", ""Performance") so take the value
  # in the '`character` column.
first_header_col
## # A tibble: 2 x 3
      row col qualification
## <int> <int> <chr>
## 1
      4 2 Humanities
## 2
        6
              2 Performance
second_header_col <-
 dplyr::filter(all_cells, col == 3) %>%
 select(row, col, subject = character)
 # The title of this header is 'subject'
 # The cells are text cells (`"history", etc.) so take the value in the
  # '`character` column.
```

```
data_cells %>%
  enhead(first_header_row, "NNW") %>%
  enhead(first_header_col, "WNW") %>%
  select(-row, -col)
```

```
## # A tibble: 16 x 3
##
      score sex
                    qualification
##
      <dbl> <chr>
                    <chr>
##
    1
          1 Female Humanities
##
    2
          2 Female Humanities
##
    3
          3 Female Humanities
    4
##
          4 Female Humanities
##
    5
          3 Male
                    Humanities
##
    6
          0 Male
                    Humanities
    7
          5 Male
                    Humanities
##
    8
          1 Male
                    Humanities
##
    9
          5 Female Performance
## 10
          6 Female Performance
          7 Female Performance
## 11
## 12
          8 Female Performance
## 13
          9 Male
                    Performance
## 14
          2 Male
                    Performance
## 15
         12 Male
                    Performance
## 16
          3 Male
                    Performance
```

3.2.7 Mixed levels of headers in the same row/column, distinguished by formatting

	Α	В	С	D
1				
2			Matilda	Nicholas
3		Humanities		
4		Classics	1	3
5		History	3	5
6		Performance		
7		Music	5	9
8		Drama	7	12

Normally different levels of headers are in different rows, or different columns, like Two clear rows of text column headers, left-aligned. But sometimes they coexist in the same row or column, and are distinguishable by formatting, e.g. bold for the top level, italic for the mid level, and plain for the lowest level.

In this example, there is a single column of row headers, where the levels are shown by different amounts of indentation. The indentation is done by formatting, rather than by leading spaces or tabs.

The first step is to find the format IDs of all the different levels of indentation.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
formats <- xlsx_formats(path)</pre>
indent0 <- which(formats$local$alignment$indent == 0)</pre>
indent1 <- which(formats$local$alignment$indent == 1)</pre>
indent0
        1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
##
  [1]
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 45 47 48
## [47] 49 50 51 52 53 54 55 56 57 58 59
indent1
## [1] 44 46
Now we use these format IDs to indentify the different levels of headers in the first column.
  xlsx_cells(path, sheets = "pivot-hierarchy") %>%
  dplyr::filter(!is_blank) %>%
  select(row, col, data_type, character, numeric, local_format_id) %>%
 print()
## # A tibble: 16 x 6
##
       row col data_type character
                                        numeric local_format_id
##
      <int> <int> <chr>
                            <chr>
                                          <dbl>
                                                           <int>
##
          2
               3 character Matilda
                                             NA
                                                              18
   1
##
  2
          2
                4 character Nicholas
                                             NA
                                                              42
##
  3
          3
               2 character Humanities
                                             NΑ
                                                              18
##
   4
          4
               2 character Classics
                                             NA
                                                              44
         4
               3 numeric <NA>
##
  5
                                              1
                                                              20
##
  6
          4
               4 numeric <NA>
                                              3
                                                              45
  7
               2 character History
##
          5
                                             NA
                                                              44
##
   8
          5
               3 numeric <NA>
                                               3
                                                              20
##
  9
         5
               4 numeric
                            <NA>
                                               5
                                                              45
## 10
          6
                2 character Performance
                                                              20
                                             NA
## 11
         7
                2 character Music
                                             NA
                                                              44
          7
## 12
                3 numeric
                           <NA>
                                              5
                                                              20
         7
## 13
               4 numeric
                            <NA>
                                               9
                                                              45
## 14
          8
               2 character Drama
                                             NA
                                                              46
                                                              24
                                              7
## 15
          8
                3 numeric
                            <NA>
## 16
                4 numeric
                            <NA>
                                             12
                                                              47
field <-
  dplyr::filter(all_cells, col == 2, local_format_id %in% indent0) %>%
  select(row, col, field = character)
  # the title of this header is 'field', meaning 'group of subjects'.
  # The cells are text cells (`"Humanities"`, `"Performance"`) so take the value
  # in the '`character` column.
```

```
## # A tibble: 2 x 3
## row col field
```

field

```
## <int> <int> <chr>
## 1
     3 2 Humanities
              2 Performance
## 2
subject <-
 dplyr::filter(all_cells, col == 2, local_format_id %in% indent1) %>%
 select(row, col, subject = character)
 # The title of this header is 'subject'
 # The cells are text cells (`"history"`, etc.) so take the value in the
  # '`character` column.
subject
## # A tibble: 4 x 3
    row col subject
## <int> <int> <chr>
## 1 4 2 Classics
## 2
       5 2 History
## 3
      7 2 Music
       8
## 4
             2 Drama
name <-
 dplyr::filter(all_cells, row == 2) %>%
 select(row, col, name = character)
 # The title of this header is 'name'.
 # The cells are text cells, so take the value in the '`character` column.
## # A tibble: 2 x 3
    row col name
## <int> <int> <chr>
      2 3 Matilda
## 1
## 2
        2
              4 Nicholas
data cells <-
 dplyr::filter(all_cells, data_type == "numeric") %>%
 select(row, col, score = numeric)
 # The data is exam scores in certain subjects, so give the data that title.
  # The data is numeric, so select only that 'value'. If some of the data was
  # also text or true/false, then you would select the `character` and `logical`
  # columns as well as `numeric`
data_cells %>%
  enhead(field, "WNW") %>%
  enhead(subject, "W") %>%
 enhead(name, "N") %>%
select(-row, -col)
## # A tibble: 8 x 4
## score field
                    subject name
##
   <dbl> <chr>
                      <chr>
                              <chr>>
## 1
      1 Humanities Classics Matilda
## 2
       3 Humanities Classics Nicholas
## 3
       3 Humanities History Matilda
## 4
       5 Humanities History Nicholas
     5 Performance Music Matilda
## 5
## 6 9 Performance Music Nicholas
## 7
      7 Performance Drama Matilda
```

8 12 Performance Drama Nicholas

3.2.8 Repeated rows/columns of headers within the table

	Α	В	С	D	E	F
1						
2				Term 1	Term 2	Term 3
3		Classics	Matilda	8	7	5
4			Nicholas	6	9	7
5			Olivia	6	8	9
6			Paul	6	2	3
7				Term 1	Term 2	Term 3
8		History	Matilda	6	4	2
9			Nicholas	3	4	8
10			Olivia	1	7	8
11			Paul	7	9	7
12				Term 1	Term 2	Term 3
13		Music	Matilda	4	2	4
14			Nicholas	7	2	9
15			Olivia	8	5	1
16			Paul	8	2	3
17				Term 1	Term 2	Term 3
18		Drama	Matilda	9	5	1
19			Nicholas	9	0	9
20			Olivia	7	6	4
21			Paul	8	9	4

Repetitions can simply be ignored. Select one of the sets of headers, and use it for all the data. In this example, the data cells are easy to distinguish from the headers mixed in among them, because only the data cells have the numeric data type.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = "pivot-repeated-headers") %>%
    dplyr::filter(!is_blank) %>%
    select(row, col, data_type, character, numeric) %>%
    print()
```

```
## # A tibble: 80 x 5
##
       row col data_type character numeric
     <int> <int> <chr> <chr> <dbl>
##
              4 character Term 1
##
  1
         2
                                        NA
##
         2
              5 character Term 2
## 3
             6 character Term 3
         2
         3
             2 character Classics
## 5
             3 character Matilda
         3
                                         NA
## 6
         3
              4 numeric <NA>
                                         1
## 7
                          <NA>
         3
             5 numeric
                                         8
## 8
         3
               6 numeric
                          <NA>
                                         7
## 9
               3 character Nicholas
         4
                                         NA
## 10
         4
               4 numeric
                          <NA>
                                         3
## # ... with 70 more rows
# View the cells in their original positions on the spreadsheet
rectify(all_cells)
## # A tibble: 20 x 6
                                `4(D)` `5(E)` `6(F)`
      `row/col` `2(B)`
                       `3(C)`
##
         <int> <chr>
                                <chr> <chr> <chr>
##
                        <chr>
                                Term 1 Term 2 Term 3
             2 <NA>
## 1
                        <NA>
## 2
             3 Classics Matilda 1
                                       8
                                             7
## 3
             4 <NA>
                       Nicholas 3
                                       1
## 4
             5 <NA>
                       Olivia 4
                                       0
                                             1
## 5
             6 <NA>
                                2
                       Paul
                                       4
                                             8
## 6
             7 <NA>
                       <NA>
                                Term 1 Term 2 Term 3
## 7
           8 History Matilda 4
                                     7
## 8
            9 <NA>
                       Nicholas 3
                                     5
## 9
            10 <NA>
                       Olivia 9
                                       8
                                             5
## 10
          11 <NA>
                                6
                       Paul
                                       2
                                             Ω
## 11
           12 <NA>
                       <NA>
                                Term 1 Term 2 Term 3
           13 Music Matilda 2
## 12
                                      9
                     Nicholas 1
## 13
           14 <NA>
                                       7
                                             7
                    Olivia O
## 14
           15 <NA>
                                       3
                                              5
## 15
           16 <NA>
                       Paul
                                2
                                       2
            17 <NA>
                       <NA>
## 16
                                Term 1 Term 2 Term 3
## 17
            18 Drama
                       Matilda 9
                                       8
## 18
            19 <NA>
                       Nicholas 1
                                       3
                                              4
## 19
            20 <NA>
                       Olivia 6
                                              4
                                       1
                       Paul
                                              2
## 20
            21 <NA>
                                6
                                       0
# The 'term' headers appear four times, but only the first one is needed.
term <-
 dplyr::filter(all_cells, row == 2) %>%
  select(row, col, term = character)
  # the title of this header is 'field', meaning 'group of subjects'.
  # The cells are text cells ("Humanities"', ""Performance"') so take the value
  # in the '`character` column.
## # A tibble: 3 x 3
##
      row col term
    <int> <int> <chr>
## 1
      2 4 Term 1
## 2
       2
             5 Term 2
```

```
## 3
              6 Term 3
        2
subject <-
 dplyr::filter(all cells, col == 2) %>%
  select(row, col, subject = character)
 # The title of this header is 'subject'
 # The cells are text cells (`"history"`, etc.) so take the value in the
  # '`character` column.
subject
## # A tibble: 4 x 3
     row col subject
## <int> <int> <chr>
## 1
       3
             2 Classics
## 2
       8
             2 History
## 3
       13
              2 Music
## 4
             2 Drama
     18
name <-
 dplyr::filter(all cells, col == 3) %>%
 select(row, col, name = character)
 # The title of this header is 'name'.
  # The cells are text cells, so take the value in the '`character` column.
## # A tibble: 16 x 3
##
      row col name
     <int> <int> <chr>
##
## 1
       3
             3 Matilda
## 2
       4
             3 Nicholas
## 3
       5
             3 Olivia
## 4
       6
            3 Paul
## 5
       8
           3 Matilda
## 6
       9 3 Nicholas
      10 3 Olivia
11 3 Paul
## 7 10
## 8
## 9 13 3 Matilda
## 10 14 3 Nicholas
      15 3 Olivia
## 11
## 12 16
             3 Paul
## 13 18
             3 Matilda
## 14 19
             3 Nicholas
## 15
       20
              3 Olivia
## 16
        21
              3 Paul
# The data cells are distinguished from the 'term' headers by their data type --
# the data cells are numeric, whereas the term headers are character.
data_cells <-
 dplyr::filter(all_cells, data_type == "numeric") %>%
 select(row, col, score = numeric)
  # The data is exam scores in certain subjects, so give the data that title.
  # The data is numeric, so select only that 'value'. If some of the data was
  # also text or true/false, then you would select the `character` and `logical`
  # columns as well as `numeric`
data_cells
```

A tibble: 48 x 3

```
##
    row col score
##
    <int> <int> <dbl>
## 1 3 4 1
## 2
       3
          5
## 3
      3
          6
               7
## 4
     4 4 3
## 5
     4 5
     4 6 2
## 6
     5 4 4
## 7
## 8 5 5 0
## 9
     5 6 1
## 10
      6
          4
## # ... with 38 more rows
data_cells %>%
 enhead(term, "N") %>%
 enhead(subject, "NNW") %>%
 enhead(name, "W") %>%
 select(-row, -col)
```

```
## # A tibble: 48 x 4
## score term subject name
     <dbl> <chr> <chr> <chr>
##
## 1
        1 Term 1 Classics Matilda
## 2
       8 Term 2 Classics Matilda
       7 Term 3 Classics Matilda
## 3
       3 Term 1 Classics Nicholas
## 4
## 5
       1 Term 2 Classics Nicholas
       2 Term 3 Classics Nicholas
       4 Term 1 Classics Olivia
## 7
       O Term 2 Classics Olivia
## 8
## 9
       1 Term 3 Classics Olivia
## 10
       2 Term 1 Classics Paul
## # ... with 38 more rows
```

${\bf 3.2.9}\quad {\bf Headers\ amongst\ the\ data}$

	А	В	С	D	Е	
1						
2				Classics		
3			Term 1	Term 2	Term 3	
4		Matilda	3	5	2	
5		Nicholas	5	3	5	
6		Olivia	9	6	5	
7		Paul	6	3	7	
8				History		
9			Term 1	Term 2	Term 3	
10		Matilda	9	2	2	
11		Nicholas	3	4	1	
12		Olivia	7	9	3	
13		Paul	5	4	8	
14				Music		
15			Term 1	Term 2	Term 3	
16		Matilda	2	2	7	
17		Nicholas	7	4	3	
18		Olivia	9	9	1	
19		Paul	1	8	9	
20				Drama		
21			Term 1	Term 2	Term 3	
22		Matilda	0	5	4	
23		Nicholas	8	0	6	
24		Olivia	4	0	6	
25		Paul	9	7	8	

This happens when what is actually a row-header, instead of being presented to the left of the data, is presented above the data. (Alternatively, what is actually a column header, instead of being presented above the data, is presented to the side.)

The way to handle it is to pretend that it is a row header, and use the "WNW" direction as normal.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
all_cells <-
  xlsx_cells(path, sheets = "pivot-header-within-data") %>%
  dplyr::filter(!is_blank) %>%
  select(row, col, data_type, character, numeric, local_format_id) %>%
  print()
## # A tibble: 80 x 6
##
              col data_type character numeric local_format_id
        row
##
      <int> <int> <chr>
                             <chr>>
                                         <dbl>
                                                          <int>
##
   1
          2
                3 character Classics
                                            NA
                                                              2
##
    2
          3
                3 character Term 1
                                            NΑ
                                                             20
##
    3
          3
                4 character Term 2
                                            NA
                                                             37
##
   4
          3
                                                             21
                5 character Term 3
                                            NA
##
   5
          4
               2 character Matilda
                                            NA
                                                             18
##
   6
          4
                3 numeric
                             <NA>
                                             4
                                                             18
##
    7
                4 numeric
                             <NA>
                                             0
                                                             27
##
   8
          4
                5 numeric
                             <NA>
                                             7
                                                             19
##
   9
          5
                2 character Nicholas
                                                             20
                                            NA
## 10
                3 numeric
                                             4
                                                             20
          5
                             <NA>
## # ... with 70 more rows
# View the cells in their original positions on the spreadsheet
rectify(all_cells)
## # A tibble: 24 x 5
                         `3(C)`
                                   `4(D)` `5(E)`
##
      `row/col` `2(B)`
##
          <int> <chr>
                          <chr>
                                   <chr> <chr>
              2 <NA>
                                          <NA>
##
                         Classics <NA>
   1
              3 <NA>
                                   Term 2 Term 3
##
    2
                         Term 1
  3
##
              4 Matilda 4
                                   0
                                          7
##
   4
              5 Nicholas 4
                                   6
                                          2
##
   5
              6 Olivia
                         9
                                   9
                                          9
##
    6
              7 Paul
                         5
                                   0
   7
              8 <NA>
                                   <NA>
                                          <NA>
##
                         History
##
              9 <NA>
                         Term 1
                                   Term 2 Term 3
   8
##
   9
             10 Matilda
                         0
                                   4
                                          2
## 10
             11 Nicholas 2
                                   5
                                          2
## # ... with 14 more rows
bold <- which(xlsx_formats(path)$local$font$bold)</pre>
# The subject headers, though mixed with the data and the 'term' headers, are
# distinguishable by the data type "character" and by being bold.
subject <-
  dplyr::filter(all_cells,
         col == 3,
         data_type == "character",
         local format id %in% bold) %>%
  select(row, col, subject = character)
```

```
# The title of this header is 'subject'
 # The cells are text cells (`"history"`, etc.) so take the value in the
 # '`character` column.
subject
## # A tibble: 4 x 3
## row col subject
## <int> <int> <chr>
## 1
      2 3 Classics
## 2
      8 3 History
            3 Music
## 3
       14
## 4
     20
            3 Drama
# We only need one set of the 'term' headers
term <-
 dplyr::filter(all_cells, row == 3, data_type == "character") %>%
 select(row, col, term = character)
 # the title of this header is 'field', meaning 'group of subjects'.
 # The cells are text cells ("Humanities"', ""Performance"') so take the value
 # in the '`character` column.
## # A tibble: 3 x 3
    row col term
## <int> <int> <chr>
## 1 3 3 Term 1
## 2
       3
            4 Term 2
## 3
       3
            5 Term 3
name <-
 dplyr::filter(all_cells, col == 2) %>%
 select(row, col, name = character)
 # The title of this header is 'name'.
 # The cells are text cells, so take the value in the '`character` column.
name
## # A tibble: 16 x 3
##
    row col name
    <int> <int> <chr>
## 1
       4
             2 Matilda
       5
## 2
             2 Nicholas
## 3
       6 2 Olivia
## 4
       7 2 Paul
           2 Matilda
## 5 10
## 6 11 2 Nicholas
## 7 12 2 Olivia
## 8
      13
             2 Paul
            2 Matilda
## 9
       16
            2 Nicholas
## 10
      17
## 11
     18
            2 Olivia
## 12 19
             2 Paul
             2 Matilda
## 13
       22
## 14 23
            2 Nicholas
## 15 24
             2 Olivia
## 16 25
            2 Paul
```

```
# The data cells are distinguished from the 'subject' headers by their data
# type -- the data cells are numeric, whereas the term headers are character.
data_cells <-
  dplyr::filter(all_cells, data_type == "numeric") %>%
  select(row, col, score = numeric)
 # The data is exam scores in certain subjects, so give the data that title.
 # The data is numeric, so select only that 'value'. If some of the data was
 # also text or true/false, then you would select the `character` and `logical`
  # columns as well as `numeric`
data_cells
## # A tibble: 48 x 3
      row col score
##
     <int> <int> <dbl>
## 1
       4
              3
## 2
         4
               4
## 3
        4
## 4
       5
             3
## 5
       5
              4
                   6
## 6
       5
             5
                   2
## 7
       6
             3
## 8
        6
              4
                   9
## 9
         6
              5
                   9
## 10
         7
              3
## # ... with 38 more rows
data_cells %>%
  enhead(subject, "WNW") %>%
  enhead(term, "N") %>%
 enhead(name, "W") %>%
 select(-row, -col)
## # A tibble: 48 x 4
##
     score subject term name
##
     <dbl> <chr>
                 <chr> <chr>
## 1
        4 Classics Term 1 Matilda
         O Classics Term 2 Matilda
## 2
## 3
        7 Classics Term 3 Matilda
## 4
       4 Classics Term 1 Nicholas
       6 Classics Term 2 Nicholas
## 5
## 6
       2 Classics Term 3 Nicholas
## 7
       9 Classics Term 1 Olivia
## 8
       9 Classics Term 2 Olivia
## 9
       9 Classics Term 3 Olivia
## 10
       5 Classics Term 1 Paul
## # ... with 38 more rows
```

Chapter 4

Small multiples

You might have heard the term 'small multiples' in the context of graphs, but it also occurs in spreadsheets, when an array of small tables could be combined into a single table.

To import an array of small tables, start by writing the code to import one, and then apply that to each in turn.

- 1. Write the code to import one table.
- 2. Wrap that code in a function.
- 3. Partition the whole spreadsheet so that each table is in one partition.
- 4. Map the function over the partitions.

4.1 Small multiples with all headers present for each multiple

	Α	В	С	D	Е	F	G
		- D		J		1	U
1	Classics				History		
2	Name	Score	Grade		Name	Score	Grade
3	Matilda	1	F		Matilda	3	D
4	Olivia	2	D		Olivia	4	С
5							
6	Music				Drama		
7	Name	Score	Grade		Name	Score	Grade
8	Matilda	5	В		Matilda	7	Α
9	Olivia	6	В		Olivia	8	Α

The code to import one of these multiples will be simple.

```
cells %>%
  behead("NNW", subject) %>%
  behead("N", header) %>%
  select(-col, -local_format_id) %>%
  spatter(header) %>%
  select(-row)
```

2 Classics D

The first table is in rows 1 to 4, columns 1 to 3, so we start by writing the code to import only that table.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
all_cells <-
  xlsx_cells(path, sheets = "small-multiples") %>%
  dplyr::filter(!is_blank) %>%
  select(row, col, data_type, character, numeric, local_format_id)
table1 <- dplyr::filter(all_cells, row %in% 1:4, col %in% 1:3)
table1 %>%
  behead("NNW", subject) %>%
  behead("N", header) %>%
  select(-col, -local_format_id) %>%
  spatter(header) %>%
  select(-row)
## # A tibble: 2 x 4
    subject Grade Name
                            Score
     <chr>
             <chr> <chr>
                            <dbl>
## 1 Classics F
                   Matilda
                                1
```

We wrap that code in a function, to be applied to each separate table.

Olivia

```
unpivot <- function(cells) {
  cells %>%
    behead("NNW", subject) %>%
    behead("N", header) %>%
    select(-col, -local_format_id) %>%
    spatter(header) %>%
    select(-row)
}
```

Now we partition the spreadsheet into the separate tables. This is done by identifying a corner cell in each table.

```
formats <- xlsx_formats(path)
italic <- which(formats$local$font$italic)

corners <-
   all_cells %>%
   dplyr::filter(local_format_id %in% italic) %>%
   select(row, col)

partitions <- partition(all_cells, corners)
partitions</pre>
```

```
## # A tibble: 4 x 3
     corner_row corner_col cells
##
          <dbl>
                     <dbl> <list>
                         1 <tibble [10 x 6]>
## 1
             1
## 2
             1
                         5 <tibble [10 x 6]>
                         1 <tibble [10 x 6]>
## 3
              6
## 4
              6
                         5 <tibble [10 x 6]>
```

Finally, map the unpivoting function over the partitions, and combine the results.

```
partitions %>%
 mutate(cells = map(cells, unpivot)) %>%
 unnest() %>%
 select(-corner_row, -corner_col)
## # A tibble: 8 x 4
##
    subject Grade Name
                          Score
    <chr> <chr> <chr> <chr> <dbl>
##
## 1 Classics F
                 Matilda
## 2 Classics D
                  Olivia
## 3 History D
                Matilda
## 4 History C
                Olivia
## 5 Music B
                  Matilda
## 6 Music
             В
                   Olivia
           Α
                              7
## 7 Drama
                   Matilda
## 8 Drama
                   Olivia
```

4.2 Same table in several worksheets/files (using the sheet/file name)

		Α	В	С			Α	
	1		Matilda	Nicholas		1		Ν
	2	Classics	1		3	2	Music	
	3	History	3		5	3	Drama	
4 E								

Because tidyxl() imports cells from multiple sheets into the same data frame, tables on separate sheets can be imported by mapping over the different sheets. Just name each sheet in the xlsx_cell() call, or don't name any to import them all.

As far as tidyxl() is concerned, the particular sheet (aka 'tab') that a cell is on is another coordinate like row and col, so the full location of a cell is its row, its col, and its sheet.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = c("humanities", "performance")) %>%
    dplyr::filter(!is_blank) %>%
    select(sheet, row, col, data_type, character, numeric)
all_cells
## # A tibble: 16 x 6
```

```
##
     sheet row col data_type character numeric
     <chr> <int> <int> <int> <chr>
##
                                 <chr>
                                            <dbl>
   1 humanities 1
                    2 character Matilda
##
                                              NA
## 2 humanities
                 1
                      3 character Nicholas
## 3 humanities
                      1 character Classics
                                              NA
                 2 2 numeric <NA>
## 4 humanities
                                               1
```

```
5 humanities
                           3 numeric
                                       <NA>
                                                       3
   6 humanities
                     3
##
                           1 character History
                                                      NA
                           2 numeric
##
  7 humanities
                     3
                                       <NA>
                                                       3
                     3
## 8 humanities
                           3 numeric
                                       <NA>
                                                      5
## 9 performance
                     1
                           2 character Matilda
                                                      NΑ
## 10 performance
                     1 3 character Nicholas
                                                      NA
## 11 performance
                     2
                         1 character Music
                                                      NA
                     2
                         2 numeric
## 12 performance
                                       <NA>
                                                       5
## 13 performance
                     2
                           3 numeric
                                       <NA>
                                                       9
## 14 performance
                     3
                         1 character Drama
                                                      NA
## 15 performance
                     3
                           2 numeric
                                       <NA>
                                                       7
## 16 performance
                     3
                                                      12
                           3 numeric
                                       <NA>
```

To prepare the sheets to be mapped over, use tidyr::nest(). The data column contains the cells of each sheet.

```
all_cells %>%
  nest(-sheet)

## # A tibble: 2 x 2
## sheet data
## <chr> tist>
## 1 humanities <tibble [8 x 5]>
## 2 performance <tibble [8 x 5]>
```

The function to unpivot each table in this case will be a couple of behead() statements. Further clean-up can be saved until the end.

```
unpivot <- function(cells) {
  cells %>%
    behead("N", name) %>%
    behead("W", subject)
}
```

After mapping the unpivot function over each sheet of cells, use tidyr::unnest() to show every row of data again.

```
all_cells %>%
  nest(-sheet) %>%
  mutate(data = map(data, unpivot)) %>%
  unnest()
```

```
## # A tibble: 8 x 8
##
    sheet
                row
                        col data_type character numeric name
                                                               subject
    <chr>>
               <int> <int> <chr>
                                     <chr>
                                               <dbl> <chr>
                                                                <chr>
                                                     1 Matilda Classics
## 1 humanities
                 2
                          2 numeric
                                     <NA>
## 2 humanities
                    2
                          3 numeric
                                                     3 Nicholas Classics
                                     <NA>
## 3 humanities
                    3
                                                     3 Matilda History
                         2 numeric
                                     <NA>
## 4 humanities
                    3
                         3 numeric
                                     <NA>
                                                    5 Nicholas History
## 5 performance
                    2
                         2 numeric
                                     <NA>
                                                    5 Matilda Music
## 6 performance
                    2
                          3 numeric
                                     <NA>
                                                    9 Nicholas Music
## 7 performance
                    3
                          2 numeric
                                     <NA>
                                                    7 Matilda Drama
## 8 performance
                    3
                          3 numeric
                                     <NA>
                                                    12 Nicholas Drama
```

Finally, do the clean-up operations that were saved until now.

```
all_cells %>%
nest(-sheet) %>%
```

```
mutate(data = map(data, unpivot)) %>%
  unnest() %>%
  transmute(field = sheet,
            name,
            subject,
            score = numeric)
## # A tibble: 8 x 4
     field
                          subject
                                   score
     <chr>>
                 <chr>
                          <chr>
                                   <dbl>
## 1 humanities Matilda Classics
## 2 humanities Nicholas Classics
                                       3
## 3 humanities Matilda History
                                       3
## 4 humanities Nicholas History
                                       5
## 5 performance Matilda Music
                                       5
## 6 performance Nicholas Music
                                       9
## 7 performance Matilda Drama
                                       7
## 8 performance Nicholas Drama
                                      12
```

4.3 Same table in several worksheets/files but in different positions

П							Α	
		Α	В	С		1	Table of sco	or
	1	Table of sco	ores			2	By subject	
	2					3		
	3	Subject	Matilda	Olivia		4	Subject	N
	4	Classics	1		2	5	Classics	П
	5	History	3		4	6	History	
Ш							•	_

This is almost the same as the section "Same table in several worksheets/files (using the sheet/file name)". The only difference is that the function you write to unpivot the table must also *find* the table in the first place, and be robust to differences in the placement and context of the table on each sheet.

In this example, both tables begin in the same column, but there is an extra row of notes above one of the tables. There are a few ways to tackle this problem. Here, we filter for the Subject cell, which is either A3 or A4, and then extend the selection to include the whole table.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = c("female", "male")) %>%
    dplyr::filter(!is_blank) %>%
```

```
select(sheet, row, col, data_type, character, numeric)
all_cells
## # A tibble: 21 x 6
##
              row
                   col data_type character
                                                numeric
     <chr> <int> <int> <chr>
                                 <chr>>
                                                   <dbl>
##
   1 female
                    1 character Table of scores
                                                     NA
##
   2 female
                3
                     1 character Subject
                                                     NA
## 3 female
                3 2 character Matilda
                                                     NA
## 4 female 3 3 character Olivia
                                                     NA
## 5 female 4
                     1 character Classics
                                                     NA
## 6 female
              4
                  2 numeric <NA>
                                                      1
## 7 female
                    3 numeric
                                 <NA>
                                                      2
## 8 female
              5
                                                     NA
                      1 character History
## 9 female
                5
                      2 numeric <NA>
                                                      3
                                                      4
## 10 female
                5
                      3 numeric
                                 <NA>
## # ... with 11 more rows
unpivot <- function(cells) {</pre>
  cells %>%
   dplyr::filter(character == "Subject") %>%
   pull(row) %>%
    {dplyr::filter(cells, row >= .)} %>%
   behead("N", name) %>%
    behead("W", subject)
}
all_cells %>%
  nest(-sheet) %>%
  mutate(data = map(data, unpivot)) %>%
  unnest() %>%
  select(sex = sheet, name, subject, score = numeric)
## # A tibble: 8 x 4
    sex name
                    subject score
    <chr> <chr>
                    <chr>>
                            <dbl>
## 1 female Matilda Classics
## 2 female Olivia Classics
## 3 female Matilda History
## 4 female Olivia History
           Nicholas Classics
                                3
## 5 male
                                0
## 6 male
           Paul
                    Classics
## 7 male
           Nicholas History
                                5
## 8 male
           Paul
                    History
                                1
```

4.4 Implied multiples

Implied multiples look like a single table, but many of the headers appear more than once. There is a dominant set of headers that are on the same 'level' (e.g. in the same row) as the other headers.

See a real-life case study

	В	С	D	E	F	G	Н	1
1	Humanitie	es			Performa	nce		
2	Classics	Grade	History	Grade	Music	Grade	Drama	Grade
3	1	F	3	D	5	В	7	Α
4	2	D	4	С	6	В	8	Α

In the example, the header "Grade" is repeated, but it really belongs in each case to the header "Classics", "History", "Music" or "Drama". Those subject headers serve two purposes: as title of each small multiple, and as the unstated "Score" header of their columns. The difficulty is in associating a grade with its corresponding score.

- 1. Filter for the "Classics", "History", "Music" and "Drama" headers, and assign them to a variable to be enhead()ed later. You could think of this as faking a set of headers that doesn't exist, but is implied.
- 2. Meanwhile, behead() the original "Classics", "History" (etc.) cells and then overwrite them with "Score"

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
all_cells <-
    xlsx_cells(path, sheets = "implied-multiples") %>%
    dplyr::filter(!is_blank) %>%
    select(row, col, data_type, character, numeric)
```

Filter for the "Classics", "History", "Music" and "Drama" headers, and assign them to a variable to be enhead()ed later.

```
subjects <-
all_cells %>%
dplyr::filter(col >= 2, row == 2, character != "Grade") %>%
select(row, col, subject = character)
subjects
## # A tibble: 4 x 3
```

```
## row col subject
## < <int> <int> <int> <int> Chr>
## 1 2 2 Classics
## 2 2 4 History
## 3 2 6 Music
## 4 2 8 Drama
```

Meanwhile, behead() the original "Classics", "History" (etc.) cells and then overwrite them with "Score".

```
all_cells %>%
behead("NNW", "field") %>%
behead("N", "header") %>%
behead("W", "name") %>%
enhead(subjects, "NNW") %>% # Reattach the filtered subject headers
mutate(header = if_else(header == "Grade", header, "Score")) %>%
select(-col) %>%
spatter(header) %>%
select(-row)
```

```
## # A tibble: 8 x 5
## field name subject Grade Score
## <chr> <chr> <chr> <chr> ## 1 Humanities Matilda Classics F 1
```

##	2	Humanities	Matilda	History	D	3
##	3	${\tt Performance}$	${\tt Matilda}$	Drama	Α	7
##	4	${\tt Performance}$	${\tt Matilda}$	Music	В	5
##	5	Humanities	Olivia	${\tt Classics}$	D	2
##	6	Humanities	Olivia	History	C	4
##	7	${\tt Performance}$	Olivia	Drama	Α	8
##	8	Performance	Olivia	Music	R	6

Formatting

This part explains in detail how to extract and interpret cell and in-cell formatting. Earlier sections have used formatting, but haven't explained exactly how it works. The motivating example is a particularly pernicious gotcha: superscript symbols.

The formatting of a cell is available via a lookup table. Well, not a lookup table – a lookup list-of-lists-(of-lists-)of-vectors. It seems complicated, but in fact it is straightforward to find out a cell's formatting.

- 1. Obtain the vector of formats that you need.
- 2. Look up the cell's style_format or local_format_id in that vector.

5.1 An example formatting lookup

Α	В
bold	
italic	
<u>underline</u>	
strikethrou _{	gh
red text	
font size	14
font arial	
yellow fill	
black borde	r
thick borde	r
dashed bor	der
row height	30
	column width 16.76
Bad' style	
	bold italic underline strikethroug red text font size font arial yellow fill black borde thick borde dashed bor row height

This example shows how to look up whether a cell is bold.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
cells <-
    xlsx_cells(path, sheet = "formatting") %>%
    select(row, col, character, style_format, local_format_id)
cells
```

A tibble: 14 x 5

```
##
                                     style_format
                                                      local_format_id
        row
              col character
##
                                     <chr>
      <int> <int> <chr>
                                                                 <int>
                1 bold
##
   1
          1
                                     Normal
                                                                     6
                                                                    8
##
                1 italic
                                     Normal
   2
          2
##
          3
                1 underline
                                     Normal
                                                                    51
##
   4
          4
                1 strikethrough
                                     Normal
                                                                    52
   5
          5
                1 red text
                                     Normal
                                                                    12
##
   6
          6
                1 font size 14
                                     Normal
                                                                    53
##
   7
          7
               1 font arial
                                     Normal
                                                                    54
##
   8
          8
                1 yellow fill
                                     Normal
                                                                    11
##
   9
          9
                1 black border
                                     Normal
                                                                    43
                1 thick border
                                                                    55
## 10
         10
                                     Normal
## 11
         11
               1 dashed border
                                     Normal
                                                                    56
## 12
         12
                1 row height 30
                                     Normal
                                                                    1
## 13
         13
                2 column width 16.76 Normal
                                                                    1
## 14
         14
                1 Bad' style
                                     Explanatory Text
                                                                    57
formats <- xlsx_formats(path)</pre>
bold <- formats $local font bold # The list of lists of lists of vectors
bold
   [1] FALSE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE
## [12] FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [56] FALSE FALSE FALSE TRUE
mutate(cells, bold = bold[local_format_id])
## # A tibble: 14 x 6
##
       row
              col character
                                     style_format
                                                      local_format_id bold
##
      <int> <int> <chr>
                                     <chr>>
                                                                 <int> <lgl>
                                                                     6 TRUE
##
                1 bold
                                     Normal
   1
          1
##
   2
          2
                1 italic
                                     Normal
                                                                     8 FALSE
                                                                    51 FALSE
##
   3
          3
                1 underline
                                     Normal
##
   4
          4
               1 strikethrough
                                     Normal
                                                                    52 FALSE
##
   5
          5
                1 red text
                                     Normal
                                                                    12 FALSE
```

```
6
          6
                1 font size 14
                                      Normal
                                                                      53 FALSE
   7
          7
                1 font arial
                                                                      54 FALSE
##
                                      Normal
##
   8
          8
                1 yellow fill
                                      Normal
                                                                      11 FALSE
##
  9
          9
                1 black border
                                      Normal
                                                                      43 FALSE
## 10
                1 thick border
                                      Normal
                                                                      55 FALSE
         10
## 11
         11
                1 dashed border
                                      Normal
                                                                      56 FALSE
## 12
         12
                1 row height 30
                                      Normal
                                                                      1 FALSE
## 13
         13
                2 column width 16.76 Normal
                                                                      1 FALSE
## 14
         14
                1 Bad' style
                                      Explanatory Text
                                                                      57 FALSE
```

A quick way to see what formatting definitions exist is to use str(). (Scroll past this for now – you don't need to memorise it).

```
formats <- xlsx_formats(path)
str(formats)

## List of 2
## $ local:List of 6
## ..$ numFmt : chr [1:59] "General" "General" "General" "General" ...</pre>
```

```
##
                            :List of 10
                               : logi [1:59] FALSE TRUE TRUE FALSE FALSE TRUE ...
##
        .. ..$ bold
        .. .. $\talic : logi [1:59] FALSE FALSE FALSE FALSE FALSE ...
##
        .. ..$ underline: chr [1:59] NA NA NA NA ...
##
        .... strike : logi [1:59] FALSE FALSE FALSE FALSE FALSE ...
##
        ....$ vertAlign: chr [1:59] NA NA NA NA ...
        ....$ size : num [1:59] 11 11 11 11 11 11 11 11 11 ...
##
        .. ..$ color
                                :List of 4
        .....$ rgb : chr [1:59] "FF000000" "FF000000" "FF000000" "...
##
##
        .....$ theme : chr [1:59] NA NA NA NA ...
        \dots ... $\ indexed: int [1:59] NA ...
        .....$ tint : num [1:59] NA ...
##
        .. ..$ name
                                : chr [1:59] "Calibri" "Calibri" "Calibri" "Calibri" ...
##
        ....$ family : int [1:59] 2 2 2 2 2 2 2 2 2 2 ...
        ....$ scheme : chr [1:59] NA NA NA NA ...
##
        ..$ fill
                          :List of 2
##
        .. .. $ patternFill :List of 3
##
        .. .. ..$ fgColor
                                      :List of 4
##
        .. .. .. s rgb
                                     : chr [1:59] NA NA NA NA ...
       ..... s theme : chr [1:59] NA NA NA NA ...
##
##
       .. .. .. .. $ indexed: int [1:59] NA ...
##
        .....$ tint : num [1:59] NA ...
##
        .. .. .. $ bgColor :List of 4
##
        ..... s rgb : chr [1:59] NA NA NA NA ...
##
        .....$ theme : chr [1:59] NA NA NA NA ...
        .. .. .. .. $ indexed: int [1:59] NA ...
##
        .....$ tint : num [1:59] NA ...
        .. .. ..$ patternType: chr [1:59] NA NA NA NA ...
        ....$ gradientFill:List of 8
        .....$ type : chr [1:59] NA NA NA NA ...
##
        .. .. .. $ degree: int [1:59] NA ...
##
        .....$ left : num [1:59] NA ...
##
        .. .. $\frac{1}{2}$ right : num [1:59] NA ...
        .... $\top : num [1:59] NA ...
        .. .. ..$ bottom: num [1:59] NA ...
##
##
       .. ... $ stop1 :List of 2
        \dots ... $\$ position: num [1:59] NA ...
##
##
        .. .. .. $ color :List of 4
##
        ..... s rgb : chr [1:59] NA NA NA NA ...
##
        ..... theme : chr [1:59] NA NA NA NA ...
        .. .. ... $ indexed: int [1:59] NA ...
        ..... tint : num [1:59] NA ...
##
        .. ... $ stop2 :List of 2
##
        ..... $\text{position: num [1:59] NA ...
        .. .. ...$ color :List of 4
        ..... s rgb : chr [1:59] NA NA NA NA ...
##
        ..... s theme : chr [1:59] NA NA NA NA ...
##
##
        .. .. .. .. $ indexed: int [1:59] NA ...
        ..... stint : num [1:59] NA ...
##
        ..$ border
                             :List of 12
        ....$ diagonalDown: logi [1:59] FALSE FALSE FALSE FALSE FALSE FALSE ....
##
        ....$ diagonalUp : logi [1:59] FALSE FALSE FALSE FALSE FALSE FALSE ...
##
        ....$ outline : logi [1:59] FALSE FA
##
##
        .. ..$ left
                                      :List of 2
```

```
.....$ style: chr [1:59] NA "thin" NA NA ...
    .. .. ..$ color:List of 4
##
##
    .. .. .. s rgb
                       : chr [1:59] NA "FF000000" NA NA ...
     ..... theme : chr [1:59] NA NA NA NA ...
##
##
    .....$ indexed: int [1:59] NA ...
    .. .. .. .. $ tint : num [1:59] NA ...
##
     .. ..$ right
                       :List of 2
##
     .. .. ..$ style: chr [1:59] NA "thin" NA NA ...
##
    .. .. ..$ color:List of 4
                       : chr [1:59] NA "FF000000" NA NA ...
##
     .. .. .. s rgb
     ..... theme : chr [1:59] NA NA NA NA ...
##
     .....$ indexed: int [1:59] NA ...
    .....$ tint : num [1:59] NA ...
##
##
     .. ..$ start
                       :List of 2
##
     .. .. ..$ style: chr [1:59] NA NA NA NA ...
##
     .. .. ..$ color:List of 4
##
    .. .. .. s rgb
                        : chr [1:59] NA NA NA NA ...
##
    ..... theme : chr [1:59] NA NA NA NA ...
     .....$ indexed: int [1:59] NA ...
##
    .. .. .. ..$ tint
##
                       : num [1:59] NA ...
##
    .. ..$ end
                       :List of 2
     .. .. ..$ style: chr [1:59] NA NA NA NA ...
##
     .. .. ..$ color:List of 4
                        : chr [1:59] NA NA NA NA ...
##
    .. .. .. s rgb
##
    ..... theme : chr [1:59] NA NA NA NA ...
     .....$ indexed: int [1:59] NA ...
##
                       : num [1:59] NA ...
     .. .. .. ..$ tint
##
    .. ..$ top
                       :List of 2
    .. .. ..$ style: chr [1:59] NA "thin" NA NA ...
##
##
     .. .. ..$ color:List of 4
##
     .. .. .. s rgb
                        : chr [1:59] NA "FF000000" NA NA ...
##
    ..... theme : chr [1:59] NA NA NA NA ...
##
    .. .. .. .. s indexed: int [1:59] NA ...
                       : num [1:59] NA ...
##
     .. .. .. ..$ tint
                       :List of 2
##
     .. ..$ bottom
##
    .....$ style: chr [1:59] NA NA "thin" NA ...
##
    .. .. ..$ color:List of 4
##
     .. .. .. s rgb
                        : chr [1:59] NA NA "FF000000" NA ...
##
    ..... theme : chr [1:59] NA NA NA NA ...
##
    .....$ indexed: int [1:59] NA ...
                       : num [1:59] NA ...
     .. .. .. stint
##
     .. ..$ diagonal
                       :List of 2
##
    .....$ style: chr [1:59] NA NA NA NA ...
##
    .. .. ..$ color:List of 4
     .. .. .. s rgb
                        : chr [1:59] NA NA NA NA ...
     ..... theme : chr [1:59] NA NA NA NA ...
##
##
    .. .. .. .. s indexed: int [1:59] NA ...
                       : num [1:59] NA ...
##
     .. .. .. ..$ tint
##
     .. ..$ vertical
                       :List of 2
##
     .....$ style: chr [1:59] NA NA NA NA ...
##
    .. .. ..$ color:List of 4
##
    .. .. .. s rgb
                       : chr [1:59] NA NA NA NA ...
##
     ..... theme : chr [1:59] NA NA NA NA ...
    .. .. .. .. $ indexed: int [1:59] NA ...
```

```
##
    .....$ tint : num [1:59] NA ...
    .. .. $ horizontal :List of 2
##
##
    .. .. ..$ style: chr [1:59] NA NA NA NA ...
##
    .. .. ..$ color:List of 4
##
    ..... s rgb : chr [1:59] NA NA NA NA ...
##
    .....$ theme : chr [1:59] NA NA NA NA ...
    .. .. .. .. $ indexed: int [1:59] NA ...
##
    .....$ tint : num [1:59] NA ...
##
    ..$ alignment :List of 8
    ....$ horizontal : chr [1:59] "general" "center" "general" "general" ....
##
    .. ..$ vertical
                        : chr [1:59] "bottom" "bottom" "bottom" "bottom" ...
    ....$ wrapText : logi [1:59] FALSE FALSE FALSE FALSE FALSE FALSE ...
##
    ....$ readingOrder : chr [1:59] "context" "context" "context" "context" ...
##
##
    ....$ indent : int [1:59] 0 0 0 0 0 0 0 0 0 ...
##
    ....$ justifyLastLine: logi [1:59] FALSE FALSE FALSE FALSE FALSE ...
    ....$ shrinkToFit : logi [1:59] FALSE FALSE FALSE FALSE FALSE FALSE ....
##
##
    ....$ textRotation : int [1:59] 0 0 0 0 0 0 0 0 0 ...
##
    ..$ protection:List of 2
    ....$ locked: logi [1:59] TRUE TRUE TRUE TRUE TRUE TRUE ...
##
    .... $\frac{1}{2}$ hidden: logi [1:59] FALSE FALSE FALSE FALSE FALSE FALSE ....
##
##
   $ style:List of 6
                 : Named chr [1:2] "General" "General"
    ....- attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
                :List of 10
##
    ..$ font
##
    ....$ bold : Named logi [1:2] FALSE FALSE
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    ....$ italic : Named logi [1:2] FALSE FALSE
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ....$ underline: Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    ....$ strike : Named logi [1:2] FALSE FALSE
##
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    .... $\text{vertAlign: Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
                : Named num [1:2] 11 11
##
    .. ..$ size
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ....$ color :List of 4
##
    .....$ rgb : Named chr [1:2] "FF000000" "FF9C0006"
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .....$ theme : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    .. .. ..$ indexed: Named int [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .....$ tint : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
                 : Named chr [1:2] "Calibri" "Calibri"
##
    .. ..$ name
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    ....$ family : Named int [1:2] 2 2
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ....$ scheme : Named chr [1:2] NA NA
##
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..$ fill :List of 2
##
    .. .. $ patternFill :List of 3
##
    .. .. ..$ fgColor :List of 4
```

```
: Named chr [1:2] NA "FFFFC7CE"
##
    .. .. ... s rgb
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s theme : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. .. .. .. $ indexed: Named int [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... stint : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. ... s bgColor
                    :List of 4
##
                    : Named chr [1:2] NA "FFCCCCFF"
    .. .. .. s rgb
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    \dots ... ... $\frac{1}{2} \text{ Named chr } [1:2] \text{ NA NA}
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ... $ indexed: Named int [1:2] NA NA
##
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ... \mbox{tint} : Named num [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    .. .. ..$ patternType: Named chr [1:2] NA "solid"
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. .. $ gradientFill:List of 8
##
    .....$ type : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ..$ degree: Named int [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. .. $ left : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ..$ right : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .....$ top : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ..$ bottom: Named num [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ..$ stop1 :List of 2
    .. .. ... $\text{position: Named num [1:2] NA NA}
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. .. ...$ color :List of 4
##
    ..... s rgb : Named chr [1:2] NA NA
##
    ##
    ..... theme : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. .. .. .. $ indexed: Named int [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. .. ... $\tint : Named num [1:2] NA NA
    ##
    .. .. ..$ stop2 :List of 2
##
    .. .. ... $ position: Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. .. .. ..$ color :List of 4
    ..... s rgb : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s theme : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. .. ... ... $ indexed: Named int [1:2] NA NA
```

```
##
    ..... stint : Named num [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..$ border :List of 12
    ....$ diagonalDown: Named logi [1:2] FALSE FALSE
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    ....$ diagonalUp : Named logi [1:2] FALSE FALSE
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    ....$ outline : Named logi [1:2] FALSE FALSE
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
                :List of 2
    .. ..$ left
    .. .. ..$ style: Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. ... ..$ color:List of 4
##
    ..... s rgb : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s theme : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. .. .. $ indexed: Named int [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s tint : Named num [1:2] NA NA
##
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
                     :List of 2
    .. ..$ right
    .. .. ..$ style: Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. ... $\text{color:List of 4}$
    ..... rgb : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    ..... s theme : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. .. .. $ indexed: Named int [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ... \$ tint : Named num [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
                     :List of 2
    .. ..$ start
    .. .. ..$ style: Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. ... $\text{color:List of 4}$
##
    .. .. .. s rgb
                     : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s theme : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    .. .. ... s indexed: Named int [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... stint : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
                     :List of 2
##
    .. ..$ end
    .. .. ..$ style: Named chr [1:2] NA NA
##
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    .. ... $\text{color:List of 4}$
##
    ..... s rgb : Named chr [1:2] NA NA
    ..... ... - attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s theme : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. ... $ indexed: Named int [1:2] NA NA
```

```
..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    ..... stint : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
                     :List of 2
    .. ..$ top
##
    .. .. ..$ style: Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    .. ...$ color:List of 4
##
    .. .. .. s rgb
                     : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... theme : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    .. .. .. .. $ indexed: Named int [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    ..... stint : Named num [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. ..$ bottom
                     :List of 2
##
    .. .. ..$ style: Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. ...$ color:List of 4
##
##
    ..... s rgb : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    ..... s theme : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. .. .. $ indexed: Named int [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s tint : Named num [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
                     :List of 2
##
    .. ..$ diagonal
    .. .. ..$ style: Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. ... ..$ color:List of 4
##
    .. .. .. s rgb
                      : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    ..... s theme : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. .. .. .. $ indexed: Named int [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    ..... s tint : Named num [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
                     :List of 2
    .. ..$ vertical
    .. .. ..$ style: Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. ...$ color:List of 4
##
##
    ..... s rgb : Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    ..... s theme : Named chr [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
    .. .. ... $ indexed: Named int [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
    ..... stint : Named num [1:2] NA NA
##
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
    .. .. $ horizontal :List of 2
##
##
    .. .. ..$ style: Named chr [1:2] NA NA
    ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
```

```
##
     .. .. ..$ color:List of 4
                        : Named chr [1:2] NA NA
##
     .. .. .. s rgb
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
     ..... theme : Named chr [1:2] NA NA
##
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
     .. .. ... $ indexed: Named int [1:2] NA NA
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
                        : Named num [1:2] NA NA
##
     .. .. .. ..$ tint
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
     ..$ alignment :List of 8
##
     ....$ horizontal
                          : Named chr [1:2] "general" "general"
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
                          : Named chr [1:2] "bottom" "bottom"
##
     .. ..$ vertical
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
     .. ..$ wrapText
                          : Named logi [1:2] FALSE FALSE
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
     ....$ readingOrder
                         : Named chr [1:2] "context" "context"
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
                          : Named int [1:2] 0 0
     .. ..$ indent
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
     ....$ justifyLastLine: Named logi [1:2] FALSE FALSE
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
##
                          : Named logi [1:2] FALSE FALSE
     ....$ shrinkToFit
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
                         : Named int [1:2] 0 0
     ...$ textRotation
##
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
     ..$ protection:List of 2
    .. .. $\text{locked: Named logi [1:2] TRUE TRUE}
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
##
     .... $\frac{1}{2}$ hidden: Named logi [1:2] FALSE FALSE
##
     ..... attr(*, "names")= chr [1:2] "Normal" "Explanatory Text"
```

Why is this so complicated? For one thing, there are too many types of formatting available to include in the data frame given by xlsx_cells().

Consider borders: each cell can have a border on each of its four sides, as well as through the middle of the cell horizontally, vertically, diagonally up and diagonally down. Each border can have its own colour and linetype. Colour can be expressed as an RGB value, a theme number with or without a tint, or an index number.

To express that in a data frame would take (4 sides + 4 through the middle) * (4 ways to express colour + 1 linetype) = 40 columns. Just for borders.

Instead, Excel dynamically defines combinations of formatting, as they occur, and gives ID numbers to those combinations. Each cell has a formatting ID, which is used to look up its particular combination of formats. Note that this means two cells that are both bold can have different formatting IDs, e.g. if one is also italic.

There is also a hierarchy of formatting. The first formatting to be applied is the 'style'. Every cell has a style, which by default is the 'normal' style. You can reformat all cells of the 'normal' style at once by updating the 'normal' style. Style formats are available under xlsx_formats()\$style

When you modify the format of a particular cell, then that modification is local to that cell. The cell's local formatting is available under xlsx_formats()\$local. Both \$style and \$local have the same structure, so it's easy to switch from checking a cell's style-level formatting to its local formatting.

Here's an example of looking up both the local bold formatting and the style-level bold formatting of a cell.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")</pre>
cells <-
 xlsx_cells(path, sheet = "formatting") %>%
 select(row, col, character, style_format, local_format_id) %>%
 dplyr::filter(row == 1, col == 1)
cells
## # A tibble: 1 x 5
      row col character style_format local_format_id
##
    <int> <int> <chr>
                          <chr>
                                                 <int>
## 1
                          Normal
        1
              1 bold
                                                     6
formats <- xlsx_formats(path)</pre>
local_bold <- formats$local$font$bold</pre>
local_bold
## [1] FALSE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE
## [12] FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [56] FALSE FALSE FALSE TRUE
style_bold <- formats$style$font$bold</pre>
style_bold
##
            Normal Explanatory Text
                              FALSE
##
             FALSE
mutate(cells,
      style bold = style bold[style format],
      local_bold = local_bold[local_format_id])
## # A tibble: 1 x 7
      row col character style_format local_format_id style_bold local_bold
    <int> <int> <chr>
                          <chr>
                                               <int> <lgl>
                                                                  <1g1>
## 1
              1 bold
                          Normal
                                                     6 FALSE
                                                                  TRUE
       1
```

Most of the time you will use the local formatting. You only need to check the style formatting when styles have been used in the spreadsheet (rare) and you want to ignore any local modifications of that style for particular cells.

Conditional formatting is an obvious omission. It isn't supported by tidyxl because it doesn't encode any new information; it's responds to cell values, which you already have. If you think you need it, feel free to open an issue.

5.2 Common formats

	Α	В
1	bold	
2	italic	
3	<u>underline</u>	
4	strikethrou	gh
5	red text	
6	font size	14
7	font arial	
8	yellow fill	
9	black borde	er
10	thick borde	r
11	dashed bor	der
12	row height	30
13		column width 16.76
14	Bad' style	

This example shows how to look up the most common formats.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
cells <-
    xlsx_cells(path, sheet = "formatting") %>%
    select(row, col, character, style_format, local_format_id, height, width)
formats <- xlsx_formats(path)</pre>
```

```
bold <- formats$local$font$bold</pre>
italic <- formats$local$font$italic</pre>
underline <- formats$local$font$underline</pre>
strikethrough <- formats$local$font$strike</pre>
font_colour <- formats$local$font$color$rgb</pre>
fill_colour <- formats$local$fill$patternFill$fgColor$rgb</pre>
font_size <- formats$local$font$size</pre>
font name <- formats$local$font$name</pre>
border_colour <- formats$local$border$right$color$rgb</pre>
border_linetype <- formats$local$border$right$style
mutate(cells,
       bold = bold[local_format_id],
       italic = italic[local_format_id],
       underline = underline[local_format_id],
       strikethrough = strikethrough[local_format_id],
       font_colour = font_colour[local_format_id],
       font_size = font_size[local_format_id],
       font_name = font_name[local_format_id],
       fill_colour = fill_colour[local_format_id],
       border_colour = border_colour[local_format_id],
       border_linetype = border_linetype[local_format_id])
```

```
## # A tibble: 14 x 17
##
       row col character
                            style_format local_format_id height width bold
##
     <int> <int> <chr>
                            <chr>
                                                 <int> <dbl> <dbl> <lgl>
             1 bold
                            Normal
                                                              8.71 TRUE
## 1
         1
                                                     6
                                                         15
## 2
         2
              1 italic
                           Normal
                                                     8
                                                         15
                                                              8.71 FALSE
## 3
         3
             1 underline Normal
                                                    51
                                                         15
                                                              8.71 FALSE
## 4
             1 strikethro~ Normal
                                                              8.71 FALSE
         4
                                                    52
                                                         15
## 5
         5
              1 red text
                           Normal
                                                    12
                                                         15
                                                              8.71 FALSE
## 6
         6
             1 font size ~ Normal
                                                    53
                                                         18.8 8.71 FALSE
## 7
         7
             1 font arial Normal
                                                    54 15
                                                              8.71 FALSE
## 8
         8
             1 yellow fill Normal
                                                    11
                                                         15
                                                              8.71 FALSE
## 9
        9
              1 black bord~ Normal
                                                    43
                                                         15
                                                              8.71 FALSE
                                                         15
## 10
        10
             1 thick bord~ Normal
                                                    55
                                                              8.71 FALSE
## 11
        11
             1 dashed bor~ Normal
                                                    56 15
                                                              8.71 FALSE
## 12
        12
              1 row height~ Normal
                                                         30
                                                            8.71 FALSE
                                                     1
                                                         15 17.4 FALSE
## 13
        13
              2 column wid~ Normal
                                                     1
              1 Bad' style Explanatory~
                                                         15
                                                              8.71 FALSE
## 14
        14
                                                    57
## # ... with 9 more variables: italic <lgl>, underline <chr>,
## #
      strikethrough <lgl>, font_colour <chr>, font_size <dbl>,
## #
      font_name <chr>, fill_colour <chr>, border_colour <chr>,
## #
      border_linetype <chr>
```

5.3 In-cell formatting

	Α	В	С	D	Е
1	in-cell: bold, italic, <u>und</u>	erline, strike	through, super	^{rscript,} red,aria	ı,size 14
2					
3	ID	Count			
4	A1-TEST	1			
5	A2-PRODUCTION	2			
6	A3-PRODUCTION	3			

The previous section was about formatting applied at the level of cells. What about when multiple formats are applied within a single cell? A single word in a string might be a different colour, to stand out.

Unlike cell-level formatting, in-cell formatting is very limited, so it can be provided as a data frame with the following columns.

- bold
- italic
- underline
- strike
- vertAlign
- size
- color rgb
- \bullet color_theme
- color indexed
- color_tint
- font
- family
- scheme

There is one of these data frames for each cell, and they are kept in a list-column called character_formatted.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
xlsx_cells(path, sheet = "in-cell formatting") %>%
    select(address, character_formatted)
```

```
## # A tibble: 9 x 2
##
     address character_formatted
##
     <chr>
             st>
## 1 A1
             <tibble [9 x 14]>
## 2 A3
             <tibble [1 x 14]>
## 3 B3
             <tibble [1 x 14]>
             <tibble [1 x 14]>
## 4 A4
             <NULL>
## 5 B4
             <tibble [2 x 14]>
## 6 A5
## 7 B5
             <NULL>
## 8 A6
             <tibble [1 x 14]>
## 9 B6
```

The way to access these data frames is via tidyr::unnest(). In this example, a single cell has a long string of words, where each word is formatted differently.

```
xlsx_cells(path, sheet = "in-cell formatting") %>%
  dplyr::filter(address == "A1") %>%
  select(address, character_formatted) %>%
  unnest()
## # A tibble: 9 x 15
     address character bold italic underline strike vertAlign <chr> <chr> <chr> <chr> <chr> <dgl> <lgl> <chr> <chr> <
##
                                                                                      size
##
                                                                                     <dbl>
          in-cell: FALSE FALSE <NA>
bold, TRUE FALSE TRUE <NA>
italic, FALSE TRUE <NA>
underline, FALSE FALSE single
strikethrough, FALSE FALSE <NA>
superscript, FALSE FALSE <NA>
red
                                                              FALSE <NA>
## 1 A1
                                                                                          0
## 2 A1
                                                              FALSE <NA>
                                                                                          0
                                                              FALSE <NA>
## 3 A1
                                                                                          0
## 4 A1
                                                              FALSE <NA>
                                                                                          0
## 5 A1
                                                              TRUE
                                                                       <NA>
                                                                                          0
                                                              FALSE superscript
## 6 A1
                                                                                          0
             red,
## 7 A1
                                 FALSE FALSE <NA>
                                                             FALSE <NA>
                                                                                          0
## 8 A1
             arial,
                               FALSE FALSE <NA>
                                                              FALSE <NA>
                                                                                          0
                               FALSE FALSE <NA>
               size 14
## 9 A1
                                                              FALSE <NA>
                                                                                          0
## # ... with 7 more variables: color_rgb <chr>, color_theme <int>,
     color_indexed <int>, color_tint <dbl>, font <chr>, family <int>,
```

It's hard to think of a plausible example, so what follows is an implausible one that nevertheless occurred in real life.

5.4 Multiple pieces of information in a single cell, with meaningful formatting

	Α	В	С	D	Е
1	in-cell: bold, italic, <u>und</u>	erline, strike	through, super	^{rscript,} red,aria	ı,size 14
2					
3	ID	Count			
4	A1-TEST	1			
5	A2-PRODUCTION	2			
6	A3-PRODUCTION	3			

The following table of products and their production readiness combines three pieces of information in a single cell.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
xlsx_cells(path, sheet = "in-cell formatting") %>%
    dplyr::filter(address != "A1") %>%
    rectify()
```

```
## 2 4 A1-TEST 1
## 3 5 A2-PRODUCTION 2
## 4 6 A3-PRODUCTION 3
```

In the ID column, the first section "A1", "A2", "A3" is the product ID. The second section "TEST", "PRODUCTION" is the production readiness, and the formatting of "TEST" and "PRODUCTION" shows whether or not manufacturing failed. In the file, one of those strings is formatted red with a strikethrough, indicating failure.

One way to extract the formatting is by unnesting, as above, but in this case we can get away with mapping over the nested data frames and pulling out a single value.

```
strikethrough <-
    xlsx_cells(path, sheet = "in-cell formatting") %>%
    dplyr::filter(address != "A1", col == 1) %>%
    mutate(strikethrough = map_lgl(character_formatted, ~ any(.x$strike))) %>%
    select(row, col, character, strikethrough)
```

This can then be joined onto the rest of the table, in the same way as the section "Already a tidy table but with meaningful formatting of single cells".

```
cells <-
    xlsx_cells(path, sheet = "in-cell formatting") %>%
    dplyr::filter(address != "A1") %>%
    select(row, col, data_type, character, numeric)

strikethrough <-
    xlsx_cells(path, sheet = "in-cell formatting") %>%
    dplyr::filter(address != "A1", col == 1) %>%
    mutate(strikethrough = map_lgl(character_formatted, ~ any(.x$strike))) %>%
    select(row, strikethrough)

left_join(cells, strikethrough, by = "row") %>%
    behead("N", header) %>%
    select(-col) %>%
    spatter(header) %>%
    select(ID, strikethrough, Count)
```

5.5 Superscript symbols

	Α	В
1	Name	Score
2	Paul ^a	9 ¹
3	Matilda	10

This is pernicious. What was Paula's score, in the table below?

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
read_excel(path, sheet = "superscript symbols")</pre>
```

```
## # A tibble: 2 x 2
## Name Score
## <chr> <chr> ## 1 Paula 91
## 2 Matilda 10
```

The answer is, it's not Paula, it's Paul (superscript 'a'), who scored 9 (superscript '1').

This sort of thing is difficult to spot. There's a clue in the 'Score' column, which has been coerced to character so that the author could enter the superscript '1' (Excel doesn't allow superscripts in numeric cells), But it would be easy to interpret that as an accident of translation, and simply coerce back to numeric with as.integer().

With tidyxl, you can count the rows of each element of the character_formatted column to identify cells that have in-cell formatting.

```
xlsx_cells(path, sheet = "superscript symbols") %>%
  dplyr::filter(data_type == "character") %>%
  dplyr::filter(map_int(character_formatted, nrow) != 1) %>%
  select(row, col, character)
```

```
## # A tibble: 2 x 3
## row col character
## <int> <int> <chr>
## 1 2 1 Paula
## 2 2 91
```

The values and symbols can then be separated by assuming the value is the first string, and the symbol is the second.

```
character = if_else(is.na(numeric), character, NA_character_)) %>%
 select(row, col, numeric, character, symbol)
## Warning in if_else(row > 1 & col == 2 & data_type == "character",
\mbox{\tt \#\#} as.numeric(character), : NAs introduced by coercion
## # A tibble: 6 x 5
##
      row col numeric character symbol
    <int> <int> <dbl> <chr>
                                <chr>
## 1
      1 1
                    NA Name
                                 <NA>
## 2
        1
            2
                   NA Score
                                <NA>
       2 1
## 3
                  NA Paul
## 4
       2 2
                    9 <NA>
                                1
     3 1 NA Matilda <NA> 3 2 10 <NA> <NA>
## 5
## 6
```

Data validation

TODO: rework the vignette?

Formulas

TODO: rework the vignette?

Other gotchas

This part is a collection of gotchas that don't fit anywhere else.

8.1 Non-text headers e.g. dates

	Α	В	С
1	Name	01/01/2018	01/01/2017
2	Matilda	2	4
3	Nicholas	1	3

At the time of writing, readxl doesn't convert Excel dates to R dates when they are in the header row.

Using tidyxl and unpivotr, you can choose to make a cell of any data type into a tidy 'header', and you can reformat it as text before **spatter()** turns it into the header of a data frame. Another way to format headers as part of the **behead()** will be shown later.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
xlsx_cells(path, sheet = "non-text headers") %>%
  behead("W", name) %>%
  behead("N", `academic-year`) %>%
  mutate(`academic-year` = strftime(`academic-year`, "%Y")) %>%
  select(row, data_type, `academic-year`, name, numeric) %>%
  spatter(`academic-year`) %>%
  select(-row)
```

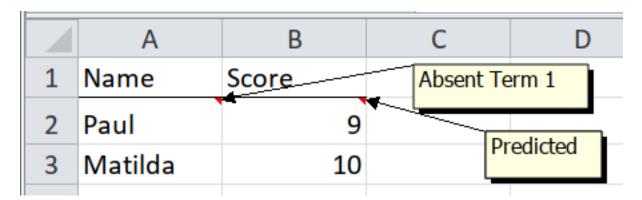
When a single set of headers is of mixed data types, e.g. some character and some date, behead() chooses the correct ones using the data_type column, before converting them all to text via format().

```
xlsx_cells(path, sheet = "non-text headers") %>%
 select(row, col, data_type, character, numeric, date) %>%
 behead("N", header)
## # A tibble: 6 x 7
      row col data_type character numeric date
                                                              header
    <int> <int> <chr>
                          <chr>
                                   <dbl> <dttm>
                                                               <chr>
## 1
        2
              1 character Matilda
                                      NA NA
                                                              Name
              2 numeric <NA>
                                        2 NA
                                                              2018-01-01
        2
                                        4 NA
                                                              2017-01-01
## 3
              3 numeric <NA>
        3
              1 character Nicholas
                                        NA NA
                                                              Name
## 5
        3
              2 numeric <NA>
                                        1 NA
                                                              2018-01-01
              3 numeric
                          <NA>
                                         3 NA
                                                              2017-01-01
```

To format a header when a single set of headers are of mixed data types, you can specify a function for each data type in the call to behead().

```
## # A tibble: 6 x 7
##
           col data_type character numeric date
                                                              header
    <int> <int> <chr>
                         <chr>
                                    <dbl> <dttm>
                                                              <chr>
## 1
        2
             1 character Matilda
                                       NA NA
                                                              NAME
## 2
              2 numeric <NA>
                                       2 NA
                                                              2018
## 3
        2
             3 numeric <NA>
                                        4 NA
                                                              2017
## 4
        3
             1 character Nicholas
                                       NA NA
                                                              NAME
## 5
        3
                                        1 NA
                                                              2018
              2 numeric <NA>
## 6
        3
              3 numeric <NA>
                                        3 NA
                                                              2017
```

8.2 Data embedded in comments



Comment strings are availabe in the comment column, just like character. Comments can have formatting, but tidyxl doesn't yet import the formatting. If you need this, please open an issue. It would probably be imported into a comment_formatted column, similarly to character_formatted.

```
path <- system.file("extdata", "worked-examples.xlsx", package = "unpivotr")
xlsx_cells(path, sheet = "comments") %>%
  select(row, col, data_type, character, numeric, comment) %>%
  behead("N", "header")
```

```
## # A tibble: 4 x 7
##
       row col data_type character numeric comment
                                                          header
     <int> <int> <chr> <chr> <dbl> <chr>
                                                            <chr>
                                        NA Absent Term 1 Name
        2
## 1
              1 character Paul
         2
               2 numeric <NA>
                                          9 Predicted Score
## 3
         3 1 character Matilda
                                         NA <NA>
                                                            Name
## 4
         3
              2 numeric <NA>
                                          10 <NA>
                                                            Score
Comments apply to single cells, so follow the same procedure as "Already a tidy table but with meaningful
formatting of single cells".
cells <-
  xlsx_cells(path, sheet = "comments") %>%
  select(row, col, data_type, character, numeric, comment)
cells
## # A tibble: 6 x 6
      row col data_type character numeric comment
     <int> <int> <chr> <chr> <dbl> <chr>
                                        NA <NA>
## 1
       1 1 character Name
## 2
              2 character Score
                                         NA <NA>
        2 character Score .... ....
2 1 character Paul NA Absent Term 1
2 2 numeric <NA> 9 Predicted
3 1 character Matilda NA <NA>
3 2 numeric <NA> 10 <NA>
       2
## 3
       2 2 numeric <NA>
## 4
## 5
## 6
values <-
  cells %>%
  select(-comment) %>%
  behead("N", header) %>%
  select(-col) %>%
  spatter(header)
values
## # A tibble: 2 x 3
      row Name
                   Score
## <int> <chr>
                   <dbl>
## 1
       2 Paul
## 2
         3 Matilda
                      10
comments <-
  cells %>%
  behead("N", header) %>%
  mutate(header = paste0(header, "_comment")) %>%
  select(row, header, comment) %>%
  spread(header, comment)
comments
## # A tibble: 2 x 3
##
       row Name_comment Score_comment
     <int> <chr>
## 1
         2 Absent Term 1 Predicted
         3 <NA>
                         <NA>
left_join(values, comments, by = "row") %>%
select(-row)
```

A tibble: 2 x 4

8.3 Named ranges

TODO

Case studies

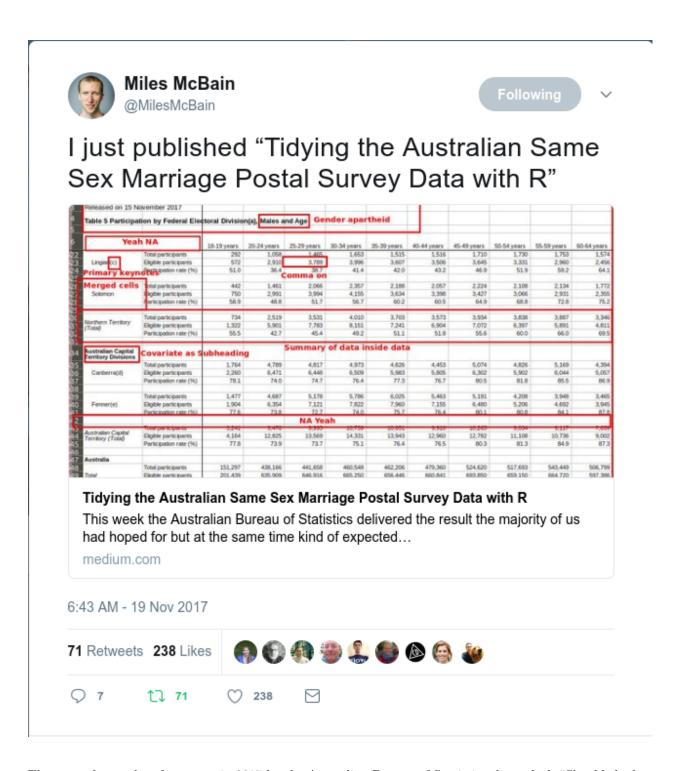
This is a collection of spreadsheets found in the wild. Some are as easy to mung as the examples; others are harder because their structure is less consistent.

Seeing and reading the code will help you guage how much work is still involved in munging a spreadsheet. Attempting them for yourself and checking the model answer will help you to hone your instincts.

The spreadsheet files are provided in the smungs package on GitHub. Install as follows.

```
# install.packages("devtools") # If you don't already have it
devtools::install_github("nacnudus/smungs")
```

9.1 Australian Marriage Survey



These are the results of a survey in 2017 by the Australian Bureau of Statistics that asked, "Should the law be changed to allow same-sex couples to marry?"

There are two tables with structures that are similar but different. Download the file. Original source.

	A	В	C							K	L	М	N	0	Р
	Australian Bureau of Statistics	ustra	alla	n Bu	rea	u of	Stat	istics	3						
	1800.0 Australian M	arriane I a	w Pos	tal Survey	2017										
	Released on 15 Novemb		411 1 0 3	tai cai vey	, 2017										
5	Table 1 Response by St	tate and Ter	ritory												
				Response	clear						Eligible Pa	rticipants			
5		Yes		No		Total		Respons		Response n		Non-resp		Tota	l
7	Name Carella Walaa	no.	%	no.	%	no.	%	no.	%	no.	%	no.	%	no.	
	New South Wales Victoria	2,374,362 2,145,629		1,736,838 1,161,098	42.2	4,111,200 3,306,727	100 100	4,111,200 3,306,727	79.2 81.4	11,036 11,028	0.2	1,065,445 743,634	20.5	5,187,681 4,061,389	1
	VICIONA Queensland	1,487,060	64.9 60.7	961.015	35.1 39.3	2.448.075	100	2.448.075	81.4 77.7	7.088	0.3	695,710	18.3 22.1	3,150,873	
	South Australia	592,528	62.5	356,247	37.5	948,775	100	948,775	79.5	2,778	0.2	242,027	20.3	1,193,580	
	Western Australia	801,575	63.7	455,924	36.3	1,257,499	100	1,257,499	78.3	3,188	0.2	346,333	21.6	1,607,020	
	Tasmania	191,948	63.6	109,655	36.4	301,603	100	301,603	79.5	805	0.2	77,020	20.3	379,428	
ŀ	Northern Territory(b)	48,686	60.6	31,690	39.4	80,376	100	80,376	58.2	229	0.2	57,496	41.6	138,101	
	Australian Capital Territory(c)	175,459	74.0	61,520	26.0	236,979	100	236,979	82.3	534	0.2	50,595	17.6	288,108	
	Australia	7,817,247	61.6	4,873,987	38.4	12,691,234	100	12,691,234	79.3	36,686	0.2	3,278,260	20.5	16,006,180	
	(-) la alcada a blanda assa :														
	(a) Includes blank responses	- 1 # 0 10	C1:> 1-1-		District	1 :::>									
	(b) Includes Christmas Island a (c) Includes Jervis Bay (within t						orra)								
	(c) includes Jervis Day (within t	He DIVISION OF F	enner) and	NOTION ISIATIO (within the L	DIVISION OF CALL	Jelia)								
	© Commonwealth of Australia 2	2017													
	Australian Bureau of Statistics	ustra	alia	n Bu	rea	u of	Stat	istics	3	K	L	M	N	0	Р
	Australian Bureau of Statistics			n Bu	rea				J	K	L	М	N	0	Р
2	Australian Bureau of	arriage La		n Bu	rea				J	K	L	М	N	0	P
2	Australian Bureau of Statistics 1800.0 Australian M	arriage La er 2017	aw Post	n Bu tal Survey	rea				J	K	L	M	N	0	P
3	Australian Bureau of Statistics 1800.0 Australian M Released on 15 November	arriage La er 2017	aw Post	n Bu tal Survey	rea				J	K	Eligible Pa		N	0	P
3 4 5	Australian Bureau of Statistics 1800.0 Australian M Released on 15 November	arriage La er 2017	aw Post	n Bu tal Survey	rea		Stat			K Response no				O	P
2 3 1 5 5 64	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fe	arriage La er 2017 ederal Electo Yes	aw Post	n Bu tal Survey sion(a) Response	rea , 2017	u of	Stat	istics	e clear		ot clear(b)	rticipants Non-resp	onding	Total	
4 5	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fe	arriage La er 2017 ederal Electo Yes 36,249	oral Divi	n Bu tal Survey sion(a) Response No 22,510	rea , 2017	u of	Stat	Respons	e clear 79.0	145	ot clear(b)	rticipants Non-resp 15,487	onding	Total 74,391	1
4 5 6	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novemb Table 2 Response by Fe	arriage La er 2017 ederal Elector Yes 36,249 30,054	oral Divi	n Bu tal Survey sion(a) Response No 22,510 25,573	rea , 2017 clear	U of	Stat	Respons 58,759 55,627	e clear 79.0 75.8	145 154	0.2 0.2	rticipants Non-resp 15,487 17,632	onding 20.8 24.0	Total 74,391 73,413	1
4 5 6 7	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fermania Divisions Bass Braddon Denison	arriage La er 2017 ederal Electo Yes 36,249 30,054 45,005	61.7 54.0 73.8	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992	rea , 2017 clear 38.3 46.0 26.2	Total 58,759 55,627 60,997	Stat	Respons 58,759 55,627 60,997	e clear 79.0 75.8 82.1	145 154 167	0.2 0.2 0.2	Non-resp 15,487 17,632 13,092	onding 20.8 24.0 17.6	Total 74,391 73,413 74,256	1 1 1
4 5 6 7 8	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fe	yes 36,249 30,054 45,005 44,746	61.7 54.0 73.8 68.8	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322	rea 38.3 46.0 26.2 31.2	Total 58,759 55,627 60,997 65,068	100 100 100 100	Respons 58,759 55,627 60,997 65,068	e clear 79.0 75.8 82.1 82.5	145 154 167 163	0.2 0.2 0.2 0.2	15,487 17,632 13,092 13,005	onding 20.8 24.0 17.6 17.3	Total 74,391 73,413 74,256 78,836	1 1 1
4 5 6 7 8 9	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fe Tasmania Divisions Bass Braddon Denison Franklin Lyons	arriage La er 2017 ederal Elector Yes 36,249 30,054 45,005 44,746 35,894	61.7 54.0 73.8 68.8 58.7	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258	38.3 46.0 26.2 31.2 41.3	Total 58,759 55,627 60,997 65,088 61,152	100 100 100 100	Respons 58,759 55,627 60,997 65,068 61,152	79.0 75.8 82.1 82.5 77.9	145 154 167 163 176	0.2 0.2 0.2 0.2 0.2 0.2	Non-resp 15,487 17,632 13,092 13,052 17,204	onding 20.8 24.0 17.6 17.3 21.9	Total 74,391 73,413 74,256 78,836 78,532	1: 1: 1: 1:
2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fe	yes 36,249 30,054 45,005 44,746	61.7 54.0 73.8 68.8	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322	rea 38.3 46.0 26.2 31.2	Total 58,759 55,627 60,997 65,068	100 100 100 100	Respons 58,759 55,627 60,997 65,068	e clear 79.0 75.8 82.1 82.5	145 154 167 163	0.2 0.2 0.2 0.2	15,487 17,632 13,092 13,005	onding 20.8 24.0 17.6 17.3	Total 74,391 73,413 74,256 78,836	1: 1: 1: 1:
2 3 3 4 5 5 5 5 6 6 7 7 7 7 7 7	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Featler States of the Statistics of the Statistics of Statisti	arriage La er 2017 ederal Elector Yes 36,249 30,054 45,005 44,746 35,894	61.7 54.0 73.8 68.8 58.7	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258	38.3 46.0 26.2 31.2 41.3	Total 58,759 55,627 60,997 65,088 61,152	100 100 100 100	Respons 58,759 55,627 60,997 65,068 61,152	79.0 75.8 82.1 82.5 77.9	145 154 167 163 176	0.2 0.2 0.2 0.2 0.2 0.2	Non-resp 15,487 17,632 13,092 13,052 17,204	onding 20.8 24.0 17.6 17.3 21.9	Total 74,391 73,413 74,256 78,836 78,532	1: 1: 1: 1:
2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fe Tasmania Divisions Bass Braddon Denison Franklin Lyons	arriage La er 2017 ederal Elector Yes 36,249 30,054 45,005 44,746 35,894	61.7 54.0 73.8 68.8 58.7	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258	38.3 46.0 26.2 31.2 41.3	Total 58,759 55,627 60,997 65,088 61,152	100 100 100 100	Respons 58,759 55,627 60,997 65,068 61,152	79.0 75.8 82.1 82.5 77.9	145 154 167 163 176	0.2 0.2 0.2 0.2 0.2 0.2	Non-resp 15,487 17,632 13,092 13,052 17,204	onding 20.8 24.0 17.6 17.3 21.9	Total 74,391 73,413 74,256 78,836 78,532	11 11 11 11 11
64 5 66 7 88 9 0 11 22 3	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrane 2 Response by Feather State 1 Response by Feather State 2 Respo	arriage La er 2017 ederal Electo Yes 36,249 30,054 45,005 44,746 35,894 191,948	61.7 54.0 73.8 68.8 58.7 63.6	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258 109,655	38.3 46.0 26.2 31.2 41.3 36.4	Total 58,759 55,627 60,997 65,068 61,152 301,603	100 100 100 100 100 100	Respons 58,759 55,627 60,997 65,068 61,152 301,603	e clear 79.0 75.8 82.1 82.5 77.9 79.5	145 154 167 163 176 805	0.2 0.2 0.2 0.2 0.2 0.2 0.2	Non-resp 15,487 17,632 13,092 13,005 17,204 77,020	onding 20.8 24.0 17.6 17.3 21.9 20.3	Total 74,391 73,413 74,256 78,836 78,532 379,428	11 11 11 11
2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Featland M Released on 15 Novembrable 2 Released on 15 Novembrable	arriage La er 2017 ederal Electo Yes 36,249 30,054 45,005 44,746 35,894 191,948	61.7 54.0 73.8 68.8 58.7 63.6	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258 109,655	38.3 46.0 26.2 41.3 36.4	Total 58,759 55,627 60,997 65,068 61,152 301,603	100 100 100 100 100 100	Respons 58,759 55,627 60,997 65,068 61,152 301,603	e clear 79.0 75.8 82.1 82.5 77.9 79.5	145 154 167 163 176 805	0.2 0.2 0.2 0.2 0.2 0.2 0.2	Non-resp 15,487 17,632 13,092 13,605 17,204 77,020	onding 20.8 24.0 17.6 17.3 21.9 20.3	Total 74,391 73,413 74,256 78,836 78,532 379,428	1 1 1 1 1 1
2 3 3 4 4 5 5 5 5 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Fe Tasmania Divisions Bass Braddon Denison Franklin Lyons Tasmania (Total) Northern Territory Divisions Lingiari(c) Solomon	arriage La er 2017 ederal Electo Yes 36,249 30,054 45,005 44,746 35,894 191,948	61.7 54.0 73.8 68.8 68.7 63.6	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258 109,655	7 Celear 38.3 46.0 26.2 31.2 41.3 36.4 45.5 34.7	Total 58,759 55,679 65,068 61,152 301,603	100 100 100 100 100 100 100	Respons- 58,759 55,627 60,997 65,088 61,152 301,603	e clear 79.0 75.8 82.1 82.5 77.9 79.5	145 154 167 163 176 805	0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2	15,487 15,487 17,632 13,092 13,605 17,204 77,020	onding 20.8 24.0 17.6 17.3 21.9 20.3	Total 74,391 73,413 74,256 78,836 78,532 379,428	1 1 1 1 1 1
2 3 3 4 5 5 6 6 7 6 8 9 7 0 7 1 7 2 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Feather Statistics Tasmania Divisions Bass Braddon Denison Franklin Lyons Tasmania (Total) Northern Territory Divisions Lingiari(c) Solomon Northern Territory (Total) Australian Capital Territory Divisions	arriage La er 2017 ederal Electo Yes 36,249 30,054 45,005 44,746 35,894 191,948 19,026 29,660 48,686	61.7 54.0 73.8 68.8 58.7 63.6 54.5 65.3 60.6	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258 109,655 15,898 15,792 31,690	38.3 46.0 26.2 41.3 36.4 45.5 34.7 39.4	Total 58,759 55,627 60,997 65,068 61,152 301,603 34,924 45,452 80,376	100 100 100 100 100 100 100 100	Respons 58,759 55,627 60,997 65,068 61,152 301,603 34,924 45,452 80,376	e clear 79.0 75.8 82.1 82.5 77.9 79.5 50.0 66.6 58.2	145 154 167 163 176 805	02 02 02 02 02 02 02 02 02 02	Non-resp 15,487 17,632 13,095 17,204 77,020 34,854 22,642 57,496	onding 20.8 24.0 17.6 17.3 21.9 20.3 49.9 33.2 41.6	Total 74,391 73,413 74,26 78,836 78,532 379,428 69,884 68,217 138,101	11 11 11 11 11 11
22 33 44 55 56 56 56 57 70 71 72 73 74 77 77 77	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by February 15 Novembrable 2	arriage La er 2017 ederal Electo Yes 36,249 30,054 45,005 44,746 35,894 191,948 19,026 29,660 48,686	61.7 54.0 73.8 68.8 58.7 63.6 54.5 65.3 60.6	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258 109,655 15,898 15,792 31,690 31,361	38.3 46.0 26.2 31.2 41.3 36.4 45.5 34.7 39.4	Total 58,759 55,627 60,997 65,068 61,152 301,603 34,924 45,452 80,376	100 100 100 100 100 100 100 100	Respons 58,759 55,627 60,997 65,088 61,152 301,603 34,924 45,452 80,376	e clear 79.0 75.8 82.1 82.5 77.9 79.5 50.0 66.6 58.2	145 154 167 163 176 805 106 123 229	02 02 02 02 02 02 02 02 02 02 02	15,487 17,632 13,092 13,605 17,204 77,020 34,854 22,642 57,496	onding 20.8 24.0 17.6 17.3 21.9 20.3 49.9 33.2 41.6	Total 74,391 73,413 74,256 78,836 78,532 379,428 69,884 68,217 138,101	10 10 10 10 10 10 10 10
2 3 3 4 4 5 5 5 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7	Australian Bureau of Statistics 1800.0 Australian M Released on 15 Novembrable 2 Response by Feather Statistics Tasmania Divisions Bass Braddon Denison Franklin Lyons Tasmania (Total) Northern Territory Divisions Lingiari(c) Solomon Northern Territory (Total) Australian Capital Territory Divisions	arriage La er 2017 ederal Electo Yes 36,249 30,054 45,005 44,746 35,894 191,948 19,026 29,660 48,686	61.7 54.0 73.8 68.8 58.7 63.6 54.5 65.3 60.6	n Bu tal Survey sion(a) Response No 22,510 25,573 15,992 20,322 25,258 109,655 15,898 15,792 31,690	38.3 46.0 26.2 41.3 36.4 45.5 34.7 39.4	Total 58,759 55,627 60,997 65,068 61,152 301,603 34,924 45,452 80,376	100 100 100 100 100 100 100 100	Respons 58,759 55,627 60,997 65,068 61,152 301,603 34,924 45,452 80,376	e clear 79.0 75.8 82.1 82.5 77.9 79.5 50.0 66.6 58.2	145 154 167 163 176 805	02 02 02 02 02 02 02 02 02 02	Non-resp 15,487 17,632 13,095 17,204 77,020 34,854 22,642 57,496	onding 20.8 24.0 17.6 17.3 21.9 20.3 49.9 33.2 41.6	Total 74,391 73,413 74,26 78,836 78,532 379,428 69,884 68,217 138,101	10 10 10 10 10 10 10

9.1.1 The full code listing

```
cells <- xlsx_cells(smungs::ozmarriage)
formats <- xlsx_formats(smungs::ozmarriage)

table_1 <-
    cells %>%
    dplyr::filter(sheet == "Table 1", row >= 5L, !is_blank) %>%
    mutate(character = str_trim(character)) %>%
    behead("NNW", "population") %>%
    behead("NNW", "response") %>%
    behead("N", "unit") %>%
    behead("W", "state") %>%
    arrange(row, col) %>%
    select(row, data_type, numeric, state, population, response, unit) %>%
```

```
spatter(unit) %>%
  select(-row)
state <-
  cells %>%
  dplyr::filter(sheet == "Table 2",
               row >= 5L,
                col == 1L,
               !is_blank,
                formats$local$font$bold[local_format_id]) %>%
  select(row, col, state = character)
table_2 <-
  cells %>%
  dplyr::filter(sheet == "Table 2",
               row >= 5L,
                !is_blank) %>%
  mutate(character = str_trim(character)) %>%
  behead("NNW", "population") %>%
  behead("NNW", "response") %>%
  behead("N", "unit") %>%
  behead("W", "territory") %>%
  enhead(state, "NNW") %>%
  arrange(row, col) %>%
  select(row, data_type, numeric, state, territory, population, response,
        unit) %>%
  spatter(unit) %>%
  select(-row)
all_tables <- bind_rows("Table 1" = table_1, "Table 2" = table_2, .id = "sheet")
all_tables
## # A tibble: 1,176 x 7
                                                        -%-
##
      sheet state
                              population
                                            response
                                                               no. territory
                                            <chr>
##
      <chr>
              <chr>>
                              <chr>>
                                                       <dbl> <dbl> <chr>
## 1 Table 1 New South Wales Eligible Par~ Non-respo~ 20.5 1.07e6 <NA>
## 2 Table 1 New South Wales Eligible Par~ Response ~ 79.2 4.11e6 <NA>
## 3 Table 1 New South Wales Eligible Par~ Response ~ 0.2 1.10e4 <NA>
## 4 Table 1 New South Wales Eligible Par~ Total
                                                           5.19e6 <NA>
                                                     100
## 5 Table 1 New South Wales Response cle~ No
                                                       42.2 1.74e6 <NA>
## 6 Table 1 New South Wales Response cle~ Total
                                                      100
                                                           4.11e6 <NA>
## 7 Table 1 New South Wales Response cle~ Yes
                                                       57.8 2.37e6 <NA>
## 8 Table 1 Victoria
                             Eligible Par~ Non-respo~ 18.3 7.44e5 <NA>
## 9 Table 1 Victoria
                             Eligible Par~ Response ~ 81.4 3.31e6 <NA>
## 10 Table 1 Victoria
                             Eligible Par~ Response ~ 0.3 1.10e4 <NA>
```

9.1.2 Step by step

... with 1,166 more rows

9.1.2.1 Table 1

The first rows, up to the column-headers, must be filtered out. The trailing rows below the table will be treated us row-headers, but because there is no data to join them to, they will be dropped automatically.

That is handy, because otherwise we would have to know where the bottom of the table is, which is likely to change with later editions of the same data.

Apart from filtering the first rows, the rest of this example is 'textbook'.

```
table_1 <-
  cells %>%
  dplyr::filter(sheet == "Table 1", row >= 5L, !is_blank) %>%
  mutate(character = str_trim(character)) %>%
  behead("NNW", "population") %>%
  behead("NNW", "response") %>%
  behead("NNW", "unit") %>%
  behead("W", "state") %>%
  arrange(row, col) %>%
  select(row, data_type, numeric, state, population, response, unit) %>%
  spatter(unit) %>%
  select(-row)
```

```
## # A tibble: 63 x 5
##
      state
                     population
                                            response
                                                                   -%-
                                                                          no.
##
      <chr>
                      <chr>
                                            <chr>>
                                                                 <dbl> <dbl>
##
   1 New South Wales Eligible Participants Non-responding
                                                                  20.5 1.07e6
## 2 New South Wales Eligible Participants Response clear
                                                                  79.2 4.11e6
## 3 New South Wales Eligible Participants Response not clear(~
                                                                  0.2 1.10e4
## 4 New South Wales Eligible Participants Total
                                                                 100
                                                                       5.19e6
## 5 New South Wales Response clear
                                                                  42.2 1.74e6
## 6 New South Wales Response clear
                                            Total
                                                                 100
                                                                       4.11e6
## 7 New South Wales Response clear
                                                                  57.8 2.37e6
                                            Yes
## 8 Victoria
                     Eligible Participants Non-responding
                                                                  18.3 7.44e5
## 9 Victoria
                     Eligible Participants Response clear
                                                                  81.4 3.31e6
## 10 Victoria
                     Eligible Participants Response not clear(~
                                                                  0.3 1.10e4
## # ... with 53 more rows
```

9.1.2.2 Table 2

This is like Table 1, broken down by division rather than by state. The snag is that the states are named in the same column as their divisions. Because the state names are formatted in bold, we can isolate them from the division names. With them out of the way, unpivot the rest of the table as normal, and then use enhead() at the end to join the state names back on.

Since tables 1 and 2 are so similar structurally, they might as well be joined into one.

```
cells <- xlsx_cells(smungs::ozmarriage)
formats <- xlsx_formats(smungs::ozmarriage)

state <-
   cells %>%
   dplyr::filter(sheet == "Table 2",
        row >= 5L,
        col == 1L,
        !is blank,
```

```
formats$local$font$bold[local_format_id]) %>%
  select(row, col, state = character)
table 2 <-
  cells %>%
  dplyr::filter(sheet == "Table 2",
               row >= 5L,
                !is blank) %>%
  mutate(character = str_trim(character)) %>%
  behead("NNW", "population") %>%
  behead("NNW", "response") %>%
  behead("N", "unit") %>%
  behead("W", "territory") %>%
  enhead(state, "NNW") %>%
  arrange(row, col) %>%
  select(row, data_type, numeric, state, territory, population, response,
         unit) %>%
  spatter(unit) %>%
  select(-row)
all_tables <-
  bind_rows("Table 1" = table_1, "Table 2" = table_2, .id = "sheet") %>%
  select(sheet, state, territory, population, response, `%`, no.)
all_tables
```

```
## # A tibble: 1,176 x 7
                                                              -%-
##
     sheet state
                            territory population
                                                   response
                                                                      no.
##
     <chr> <chr>
                            <chr> <chr>
                                                   <chr>
                                                             <dbl> <dbl>
                                     Eligible Par~ Non-respo~ 20.5 1.07e6
## 1 Table 1 New South Wales <NA>
## 2 Table 1 New South Wales <NA>
                                     Eligible Par~ Response ~ 79.2 4.11e6
## 3 Table 1 New South Wales <NA>
                                     Eligible Par~ Response ~ 0.2 1.10e4
## 4 Table 1 New South Wales <NA>
                                     Eligible Par~ Total
                                                            100 5.19e6
## 5 Table 1 New South Wales <NA>
                                     Response cle~ No
                                                             42.2 1.74e6
                                                            100 4.11e6
## 6 Table 1 New South Wales <NA>
                                     Response cle~ Total
## 7 Table 1 New South Wales <NA>
                                     Response cle~ Yes
                                                             57.8 2.37e6
## 8 Table 1 Victoria
                                     Eligible Par~ Non-respo~ 18.3 7.44e5
                           <NA>
## 9 Table 1 Victoria
                            <NA>
                                     Eligible Par~ Response ~ 81.4 3.31e6
## 10 Table 1 Victoria
                           <NA>
                                     Eligible Par~ Response ~ 0.3 1.10e4
## # ... with 1,166 more rows
```

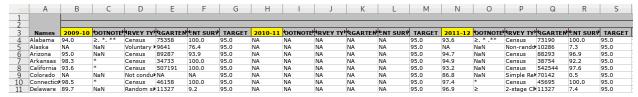
9.2. VACCINATIONS 109

9.2 Vaccinations



This is a real-life example of implied multiples. Implied multiples look like a single table, but many of the headers appear more than once. There is a dominant set of headers that are on the same 'level' (e.g. in the same row) as the other headers.

In this case, there is a small multiple for each year of data. The year headers are highlighted in yellow in the screenshot.



The way to unpivot this is to realise that the year cells represent two different things: the year (obviously) and a statistic (percentage vaccinated). It would have been easier to unpivot if the years had been put into a separate row of headers, so we will pretend that that was in fact the case.

- 1. Filter for the year cells and store in a variable to enhead() later.
- 2. behead() everything else as usual, and then overwite the year headers with percentage_vaccinated.
- 3. enhead() the year cells.

The original spreadsheet has been replaced by an annual refresh, so you will need to use the file from the smungs package.

```
## # A tibble: 42 x 3
##
       row col year
##
     <int> <int> <chr>
              2 2009-10
## 1
         3
              8 2010-11
## 2
         3
## 3
         3
             14 2011-12
## 4
         3 20 2012-13
## 5
        3 26 2013-14
## 6
         3 32 2014-15
## 7
         3 38 2015-16
## 8
         3 44 2009-10
## 9
         3 50 2010-11
## 10
         3
             56 2011-12
## # ... with 32 more rows
cells %>%
 select(row, col, data_type, character) %>%
 behead("NNW", "series") %>%
 behead("NNW", "population") %>%
 behead("W", "state") %>%
 behead("N", "header") %>%
 mutate(header = if_else(str_detect(header, "20[0-9]{2}-[0-9]{2}"),
                         "percent_vaccinated",
                        header),
        header = str_replace_all(str_to_lower(header), " ", "_")) %>%
 enhead(years, "NNW") %>%
 select(row, series, population, state, year, header, character) %>%
 spatter(header, character) %>%
 select(series, population, state, year, percent_vaccinated, percent_surveyed,
        everything())
```

```
## # A tibble: 2,226 x 11
##
      series population state year percent_vaccina~ percent_surveyed
##
      <chr>
              <chr>
                         <chr> <chr> <chr>
                                                       <chr>
                                                                        <int>
## 1 School ~ All kinde~ Alab~ 2009~ 94.0
                                                       100.0
## 2 School ~ All kinde~ Alab~ 2010~ NA
                                                       NA
                                                                            4
## 3 School ~ All kinde~ Alab~ 2011~ 93.6
                                                       100.0
                                                                            4
## 4 School ~ All kinde~ Alab~ 2012~ 92.8
                                                       100.0
                                                                            4
## 5 School ~ All kinde~ Alab~ 2013~ 92.0
                                                       100.0
## 6 School ~ All kinde~ Alab~ 2014~ 93.5
                                                                            4
                                                       100.0
## 7 School ~ All kinde~ Alab~ 2015~ 93.1
                                                                            4
                                                       100.0
## 8 School ~ All kinde~ Alab~ 2009~ NReq
                                                       100.0
                                                                            4
## 9 School ~ All kinde~ Alab~ 2010~ NA
                                                      NA
## 10 School ~ All kinde~ Alab~ 2011~ NReq
                                                       100.0
## # ... with 2,216 more rows, and 4 more variables: footnotes \langle chr \rangle,
## # survey_type <chr>, target <chr>, total_kindergarten_population <chr>
```

9.3. US CRIME 111

9.3 US Crime



I'm staring at this nightmare right now and wondering why @hadleywickham's Tidy Data principles aren't taught EVERYWHERE

vita.had.co.nz/papers/tidy-da...



These are two tables of numbers of crimes in the USA, by state and category of crime. Confusingly, they're numbered Table 2 and Table 3. Table 1 exists but isn't included in this case study because it is so straightforward.

9.3.1 Table 2

Table 2																						-
Crime in the United States								-		-	_			-						-		
by Region, Geographic Divi-	sion, and State, 2	015-2016			Murder	and		_						_								
					nonnegl		Rape		Rap													
			Violent cr	rime ³	manslau		(revised defi		(legacy def		Robbe	rv.	Aggravated	assault	Property (rime	Burglan	v l	Larceny	theft	Motor vehic	cle theft
		_		Rate per		Rate per		Rate per	110000000000000000000000000000000000000	Rate per		Rate per		Rate per		Rate per		Rate per		Rate per		Rate per
Area	Year	Population ²		100.000		100,000		100.000		100.000		100.000		100.000		100.000		100.000		100.000		100.000
1200000										7.000										100000		
	2015	320,896,618	1,234,183	384.6	15,883	4.9	126,134	39.3	91,261	28.4	328,109	102.2	764,057	238.1	8,024,115	2,500.5	1,587,564	494.7	5,723,488	1,783.6	713,063	222.2
United States Total ^{6, 7, 8, 9}	2016	323,127,513	1,283,058	397.1	17,250	5.3	130,603	40.4	95,730	29.6	332,198	102.8	803,007	248.5	7,919,035	2,450.7	1,515,096	468.9	5,638,455	1,745.0	765,484	236.
	Percent		10000	2000	200	13702	2000	2000	920	-200	1.000	25.00	1000		1000	1999	52.0	-	1000	1000	6000	100
	change		+4.0	+3.2	+8.6	+7.9	+3.5	+2.8	+4.9	+4.2	+1.2	+0.5	+5.1	+4.4	-1.3	-2.0	-4.6	-5.2	-1.5	-2.2	+7.4	+6.6
	2015	56,184,737	180,474	321.2	1,967	3.5	16,415	29.2	11,912	21.2	56,202	100.0	105,890	188.5	951,940	1.694.3	157,910	281.1	737,087	1,311.9	56,943	101.3
Northeast ⁶	2016	56,209,510	178,244	317.1	1,955	3.5	16,651	29.6	12,219	21.7	53,033	94,3	106,605	189.7	909,897	1,618.8	142,720	253.9	709,721	1,262.6	57,456	102.3
	Percent																					
	change		-1.2	-1.3	-0.6	-0.7	+1.4	+1.4	+2.6	+2.5	-5.6	-5.7	+0.7	+0.6	-4.4	-4.5	-9.6	-9.7	-3.7	-3.8	+0.9	+0.5
	****		40.1	****					4.45	***					242.00			***	100.0			
	2015	14,710,229	42,121	286.3	326	2.2	4,602	31.3	3,374	22.9	9,629	65.5	27,564	187.4	257,073	1,747.6	46,260	314.5	193,051	1,312.4	17,762	120.7
New England	2016 Percent	14,735,525	41,598	282.3	292	2.0	4,505	30.6	3,328	22.6	9,407	63.8	27,394	185.9	243,769	1,654.3	42,095	285.7	182,985	1,241.8	18,689	126.8
	change		-1.2	-1.4	-10.4	-10.6	-2.1	-2.3	-1.4	-1.5	-2.3	-2.5	-0.6	-0.8	-5.2	-5.3	-9.0	-9.2	-5.2	-5.4	+5.2	+5.0
	- Change		11.0	-101	-10.1	10.0	-	-	-114	110	- 410	- 410	-0.0	-0.0		710	-7.0	7.6	-516	2.1	1016	- 200
	2015	3,584,730	7,938	221.4	116	3.2	798	22.3	585	16.3	2,925	81.6	4,099	114.3	65,610	1,830.3	10,286	286.9	48,898	1,364.1	6,426	179.3
Connecticut	2016	3,576,452	8,123	227.1	78	2.2	763	21.3	561	15.7	2,703	75.6	4,579	128.0	64,664	1,808.0	10,045	280.9	47,512	1,328.5	7,107	198.7
	Percent																					
	change		+2.3	+2.6	-32.8	-32.6	-4.4	-4.2	-4.1	-3.9	-7.6	-7.4	+11.7	+12.0	-1.4	-1.2	-2.3	-2.1	-2.8	-2.6	+10.6	+10.9
	2015	1,329,453	1,726	129.8	23	1.7	461	34.7	337	25.3	311	23.4	931	70.0	24,361	1,832.4	4,694	353.1	18,853	1,418.1	814	61.2
Maine	2016	1,331,479	1,726	123.8	20	1.5	412	30.9	300	22.5	266	20.0	950	71.3	21,912	1,632.4	4,003	300.6	17,134	1,286.8	775	58.2
	Percent	1,002,415	2,040	140.0	20		712	30.3	300		200	20.0	350	71.5	21,712	1,040.1	4,005	200.0	11,104	1,200.0	- 112	20.2
	change		-4.5	4.7	-13.0	-13.2	-10.6	-10.8	-11.0	-11.1	-14.5	-14.6	+2.0	+1.9	-10.1	-10.2	-14.7	-14.9	-9.1	-9.3	-4.8	-4.9
	Jacobi et al																					
	2015	6,784,240	26,453	389.9	131	1.9	2,082	30.7	1,538	22.7	5,266	77.6	18,974	279.7	114,547	1,688.4	21,826	321.7	84,676	1,248.1	8,045	118.6
Massachusetts	2016	6,811,779	25,677	376.9	134	2.0	2,128	31.2	1,592	23.4	5,365	78.8	18,050	265.0	106,339	1,561.1	19,193	281.8	79,088	1,161.0	8,058	118.3
	Percent		-2.9	-3.3	+2.3	+1.9	+2.2	+1.8	+3.5	+3.1	+1.9	+1.5	4.9	-5.3	-7.2	-7.5	-12.1	12.4		-7.0	+0.2	- 0.1
	change		-2.9	-3.3	+2.3	+1.9	+2.2	+1.8	+3.5	+3.1	+1.9	+1.5	4.9	-5.3	-7.2	-7.5	-12.1	-12.4	-6,6	-7.0	+0.2	-0.2
	2015	1,330,111	2,692	202.4	15	1.1	644	48.4	468	35.2	475	35.7	1,558	117.1	23,453	1,763.2	3,482	261.8	19,076	1,434.2	895	67.3
New Hampshire	2016	1,334,795	2,637	197.6	17	1.3	582	43.6	423	31.7	427	32.0	1,611	120,7	20,194	1,512.9	2,963	222.0	16,360	1,225,7	871	65.3
	Percent																					
	change		-2.0	-2.4	+13.3	+12.9	-9.6	-9.9	-9.6	-9.9	-10.1	-10.4	+3.4	+3.0	-13.9	-14.2	-14.9	-15.2	-14.2	-14.5	-2.7	-3.0
Rhode Island	2015	1,055,607	2,565 2,524	243.0	31 29	2.9	467 442	44.2	342 328	32.4 31.0	547 540	51.8	1,520	144.0	20,038	1,898.2	3,938	373.1 358.6	14,714	1,393.9	1,386	131.3
Knode Island	Percent	1,036,426	2,524	238.9	29	2.7	442	41.8	328	31.0	540	31.1	1,513	143.2	20,058	1,898.7	3,788	338.0	14,074	1,389.0	1,596	151.1
	change		-1.6	-1.7	-6.5	-6.5	-5.4	-5.4	-4.1	-4.2	-1.3	-1.4	-0.5	-0.5	+0.1		-3.8	-3.9	-0.3	-0.3	+15.2	+15.1
			-1.0		0.0												.5.0		3.3			
	2015	626,088	747	119.3	10	1.6	150	24.0	104	16.6	105	16.8	482	77.0	9,064	1,447.7	2,034	324.9	6,834	1,091.5	196	31.3
Vermont	2016	624,594	989	158.3	14	2.2	178	28.5	124	19.9	106	17.0	691	110.6	10,602	1,697.4	2,103	336.7	8,217	1,315.6	282	45.1
	Percent		200	10000	77/8675	100000	1000	972570	55555	2000	100	7.00	100	2002	07220	5922	100	1000	1172014	2000	0000	1,11200
	change		+32.4	+32.7	+40.0	+40.3	18.7	+19.0	+19.2	+19.5	+1.0	+1.2	+43,4	+43.7	+17.0	+17.2	+3.4	+3.6	+20.2	+20.5	+43.9	+44.2
	2015	41,474,508	138,353	333.6	1.641	4.0	11,813	28.5	8.538	20.6	46,573	112.3	78,326	188.9	694,867	1.675.4	111,650	269.2	544.036	1,311.7	39,181	94.5
Middle Arlantic	2015	41,474,308	136,555	220.5	1,641	4.0	12 146	20.3	9.901	21.4	40,373	105.2	70,320	101.0	666 129	1,606.1	100,625	242.6	526 226	1,311.7	39,181	94.3

9.3.1.1 Simple version

This is straightforward to import as long as you don't care to organise the hierarchies of crimes and areas. For example, Conneticut is within the division New England, which itself is within the region Northeast, but if you don't need to express those relationships in the data then you can ignore the bold formatting.

The only slight snag is that the header cells in row 5 are blank. There is a header for the units "Rate per 100,000", but no header for the units "Count" – the cells in those positions are empty. It would be a problem if the cells didn't exist at all, because behead("N", "unit") wouldn't be able to associate data cells with missing header cells. Fortunately they do exist (because they have formatting), they are just empty or NA. To make sure they aren't ignored, use drop_na = FALSE in behead(), and then later fill the blanks in the units column with "Count".

```
cells <-
  xlsx_cells(smungs::us_crime_2) %>%
  mutate(character = map_chr(character_formatted,
                             ~ ifelse(is.null(.x), character, .x$character[1])),
         character = str_replace_all(character, "\n", " "))
cells %>%
  dplyr::filter(row >= 4L) %>%
  select(row, col, data_type, character, numeric) %>%
  behead("NNW", "crime") %>%
  behead("N", "unit", drop_na = FALSE) %>%
  behead("WNW", "area") %>%
  behead("W", "year") %>%
  behead("W", "population") %>%
  dplyr::filter(year != "Percent change") %>%
  mutate(unit = if else(unit == "", "Count", unit)) %>%
  select(row, data_type, numeric, unit, area, year, population, crime) %>%
```

9.3. US CRIME 113

```
spatter(unit) %>%
select(-row)
```

```
## # A tibble: 1,320 x 6
##
      area
                          year population crime
                                                       Count `Rate per 100,~
##
      <chr>>
                          <chr> <chr>
                                                       <dbl>
                                                                         <dbl>
##
   1 United States Total 2015
                                320896618
                                           Aggravate~ 7.64e5
                                                                         238.
   2 United States Total 2015
                                                                         495.
                                320896618
                                           Burglary
                                                      1.59e6
## 3 United States Total 2015
                                320896618
                                           Larceny-t~ 5.72e6
                                                                        1784.
## 4 United States Total 2015
                                320896618
                                           Motor veh~ 7.13e5
                                                                         222.
## 5 United States Total 2015
                                320896618 Murder an~ 1.59e4
                                                                           4.9
## 6 United States Total 2015
                                320896618
                                           Property ~ 8.02e6
                                                                        2500.
## 7 United States Total 2015
                                320896618
                                           Rape (leg~ 9.13e4
                                                                          28.4
## 8 United States Total 2015
                                320896618
                                           Rape (rev~ 1.26e5
                                                                         39.3
## 9 United States Total 2015
                                                                         102.
                                320896618
                                           Robbery
                                                      3.28e5
## 10 United States Total 2015
                                320896618
                                           Violent c~ 1.23e6
                                                                         385.
## # ... with 1,310 more rows
```

9.3.1.2 Complex version

cells %>%

If you do mind about grouping states within divisions within regions, and crimes within categories, then you have more work to do using enhead() rather than behead().

- 1. Select the header cells at each level of the hierarchy and store them in their own variables. For example, filter for the bold cells in row 4, which are the categories of crimes, and store them in the categories variable.
- 2. Select the data cells, and use enhead() to join them to the headers.

In fact the headers unit, year, population can be handled by behead(), because they aren't hierarchichal, so only the variables category, crime, region, division and state are handled by enhead().

```
cells <-
  xlsx cells(smungs::us crime 2) %>%
  mutate(character = map_chr(character_formatted,
                              - ifelse(is.null(.x), character, .x$character[1])),
         character = str_replace_all(character, "\n", " "))
formats <- xlsx_formats(smungs::us_crime_2)</pre>
categories <-
  cells %>%
  dplyr::filter(row == 4L,
                data_type == "character",
                formats$local$font$bold[local_format_id]) %>%
  select(row, col, category = character)
categories
## # A tibble: 2 x 3
##
       row
             col category
     <int> <int> <chr>
## 1
         4
               4 Violent crime
              16 Property crime
crimes <-
```

```
dplyr::filter(row == 4L, data_type == "character") %>%
  mutate(character = if_else(character %in% categories$category,
                           "Total",
                           character)) %>%
  select(row, col, crime = character)
crimes
## # A tibble: 13 x 3
##
       row col crime
     <int> <int> <chr>
             1 Area
## 1
        4
## 2
        4 2 Year
## 3 4 3 Population
       4 4 Total
## 4
## 5 4 6 Murder and nonnegligent manslaughter ## 6 4 8 Rape (revised definition)
## 7 4 10 Rape (legacy definition)
## 8 4 12 Robbery
      4 14 Aggravated assault
## 9
## 10 4 16 Total
## 11 4 18 Burglary
       4 20 Larceny-theft
## 12
         4 22 Motor vehicle theft
## 13
regions <-
 cells %>%
 dplyr::filter(row >= 6L,
               col == 1L,
               data_type == "character",
               formats$local$font$bold[local_format_id]) %>%
  select(row, col, region = character)
regions
## # A tibble: 5 x 3
## row col region
## <int> <int> <chr>
## 1
      6 1 United States Total
## 2
       9 1 Northeast
## 3 45 1 Midwest
## 4 90 1 South
## 5 153 1 West
divisions <-
 cells %>%
  dplyr::filter(row >= 6L,
               col == 1L,
               data_type == "character",
               !formats$local$font$bold[local_format_id],
               !str_detect(character, "^ {5}")) %>%
  select(row, col, division = character)
divisions
## # A tibble: 21 x 3
##
      row col division
     <int> <int> <chr>
## 1 12 1 New England
```

9.3. US CRIME 115

```
##
        33
             1 Middle Atlantic
             1 East North Central
## 3
       48
## 4 66
             1 West North Central
      93
             1 South Atlantic
## 5
## 6
       123
             1 East South Central
## 7 138
             1 West South Central
## 8 156
             1 Mountain
## 9
             1 Pacific
       183
## 10
       201
            1 Puerto Rico
## # ... with 11 more rows
states <-
 cells %>%
 dplyr::filter(row >= 6L,
              col == 1L,
              data_type == "character") %>%
 mutate(character = if_else(str_detect(character, "^ {5}"),
                          str_trim(character),
                           "Total")) %>%
 select(row, col, state = character)
states
## # A tibble: 77 x 3
##
      row col state
     <int> <int> <chr>
##
## 1
       6 1 Total
## 2
        9
             1 Total
## 3
      12
             1 Total
## 4
     15 1 Connecticut
## 5 18 1 Maine
## 6 21
            1 Massachusetts
             1 New Hampshire
## 7
      24
## 8 27
            1 Rhode Island
## 9 30 1 Vermont
## 10 33
             1 Total
## # ... with 67 more rows
cells %>%
 dplyr::filter(row >= 5L, col >= 2L) %>%
 select(row, col, data_type, character, numeric) %>%
 behead("N", "unit") %>%
 behead("W", "year") %>%
 behead("W", "population") %>%
 enhead(categories, "NNW") %>%
 enhead(crimes, "NNW") %>%
 enhead(regions, "WNW") %>%
 enhead(divisions, "WNW", drop = FALSE) %>%
 enhead(states, "WNW", drop = FALSE) %>%
 dplyr::filter(year != "Percent change") %>%
 select(value = numeric, category, crime, region, division, state, year, population)
## # A tibble: 2,640 x 8
##
       value category
                                    region division state year population
                         crime
##
       <dbl> <chr>
                         <chr>
                                    <chr> <chr>
                                                <chr> <chr> <chr>
## 1 42121 Violent crime Total
                                    North~ New Eng~ Total 2015 14710229
## 2 286. Violent crime Total
                                   North~ New Eng~ Total 2015 14710229
```

```
3 41598
              Violent crime Total
                                       North~ New Eng~ Total 2016
                                                                    14735525
##
        282. Violent crime Total
##
   4
                                       North~ New Eng~ Total 2016
                                                                    14735525
##
        326
              Violent crime Murder an~ North~ New Eng~ Total 2015
                                                                    14710229
          2.2 Violent crime Murder an~ North~ New Eng~ Total 2015
##
   6
                                                                    14710229
##
   7
        292
              Violent crime Murder an~ North~ New Eng~ Total 2016
                                                                    14735525
   8
              Violent crime Murder an~ North~ New Eng~ Total 2016
                                                                    14735525
##
          2
              Violent crime Rape (rev~ North~ New Eng~ Total 2015
   9
                                                                    14710229
         31.3 Violent crime Rape (rev~ North~ New Eng~ Total 2015
## 10
                                                                   14710229
## # ... with 2,630 more rows
```

9.3.1.3 Table 3

A	В	С	D	E	F	G	Н	1	J	К	L	M	N
Table 3													
Crime in the United States													
by State, 2016													
				Violent	Murder and nonnegligent	Rape (revised	Rape (legacy		Aggravated	Property		Larceny-	Motor
State	Area		Population	crime ¹	manslaughter	definition")	definition ¹)	Robbery	assault	crime	Burglary	theft	theft
ALABAMA	Metropolitan Statistical Area		3,716,889										
		Area actually reporting	99.2%	20,793	331	1,489	1,084	4,174	14,799	114,605	26,966	78,072	9
		Estimated total	100.0%	20,949	332	1,498	1,091	4,200	14,919	115,564	27,156	78,771	
	Cities outside metropolitan areas		520,422										
		Area actually reporting	96.6%	3,551	41	263	187	410	2,837	19,841	4,173	14,332	
		Estimated total	100.0%	3,646	42	269	193	421	2,914	20,291	4,275	14,641	
	Nonmetropolitan counties		625,989										
		Area actually reporting	94.5%	1,220	31	141	95	61	987	7,096	2,490	3,934	
		Estimated total	100.0%	1,291	33	149	101	65	1,044	7,507	2,634	4,162	
ALASKA	State Total		4,863,300	25,886	407	1,916	1,385	4,686	18,877	143,362	34,065	97,574	1
		Rate per 100,000 inhabitants		532.3	8.4	39.4	28.5	96.4	388.2	2,947.8	700.5	2,006.3	
ALASKA	Metropolitan Statistical Area		350,298										
		Area actually reporting	100.0%	3,751	35	583	439	741	2,392	17,143	2,243	12,620	
	Cities outside metropolitan areas		128,053										
		Area actually reporting	97.4%	964	2	165	119	71	726	4,069	655	3,101	
		Estimated total	100.0%	989	2	169	123	73	745	4,177	672	3,184	
	Nonmetropolitan counties		263,543										
		Area actually reporting	100.0%	1,226	1 Ent	er 301	195	36	874	3,556	1,138	1,962	
	State Total		741,894	5,966	54	1,053	757	850	4,011	24,876	4,053	17,766	
		Rate per 100,000 inhabitants		804.2	7.0	141.9	102.0	114.6	540.6	3,353.0	546.3	2,394.7	
ARIZONA	Metropolitan Statistical Area		6,583,506										
		Area actually reporting	98.3%	27,522	334	2,997	2,238	6,931	17,260	196,398	35,424	143,602	1
		Estimated total	100.0%	27,789	338	3,025	2,258	6,971	17,455	198,543	35,844	145,152	1
	Cities outside metropolitan areas		124,210										
		Area actually reporting	100.0%	4,342	32	250	184	65	3,995	5,937	1,199	4,069	
	Nonmetropolitan counties		223,355										
		Area actually reporting	100.0%	452	10	15	10	19	408	1,952	693	1,054	
	State Total		6,931,071	32,583	380	3,290	2,452	7,055	21,858	206,432	37,736	150,275	1
		Rate per 100,000 inhabitants		470.1	5.5	47.5	35.4	101.8	315.4	2,978.4	544.4	2,168.1	
ARKANSAS	Metropolitan Statistical Area		1,851,896										
		Area actually reporting	99.9%	11,573	144	1,315	967	1,711	8,403	68,739	15,650	47,839	
		Estimated total	100.0%	11,576	144	1,316	968	1,711	8,405	68,762	15,655	47,856	
	Cities outside metropolitan areas		515,688										
	(1) (4) (4) (4) (4) (4) (4) (4) (4) (4) (4	Area senially reporting	Q4 194	2 025	AA	471	3/13	3.47	2.063	18 438	4 926	12 537	

This table is confusing to humans, let alone computers. The Population column seems to belong to a different table altogether, so that's how we'll treat it.

- 1. Import the Population column and the state/area headers to the left.
- 2. Import the crime-related column headers, and the state/area headers to the left.
- 3. Join the two datasets.

The statistic header ends up having blank values due to the cells being blank, so these are manually filled in.

The hierarchy of crime (e.g. 'robbery' is within 'violent crime') is ignored. That would be handled in the same way as for Table 2.

9.3. US CRIME 117

```
behead("W", "statistic", drop_na = FALSE) %>%
 mutate(statistic = case_when(is.na(statistic) ~ "Population",
                              statistic == "" ~ "Population",
                              TRUE ~ str_trim(statistic))) %>%
 dplyr::filter(data_type == "numeric",
                !str_detect(area, regex("total", ignore_case = TRUE)),
               statistic != "Estimated total") %>%
 select(data_type, numeric, state, area, statistic) %>%
 spatter(statistic)
crime <-
 cells %>%
 dplyr::filter(row >= 4, col != 5L) %>%
 behead("WNW", "state") %>%
 behead("WNW", "area") %>%
 behead("W", "statistic", formatters = list(character = str_trim)) %>%
 behead("N", "crime") %>%
 dplyr::filter(data_type == "numeric",
               !str_detect(area, regex("total", ignore_case = TRUE)),
               !is.na(statistic),
               statistic != "") %>%
 mutate(statistic = case_when(statistic == "Area actually reporting" ~ "Actual",
                              statistic == "Estimated total" ~ "Estimated")) %>%
 select(data_type, numeric, state, area, statistic, crime) %>%
 spatter(statistic)
left_join(population, crime)
## Joining, by = c("state", "area")
## # A tibble: 1,480 x 7
##
     state
                       `Area actually r~ Population crime
                                                            Actual Estimated
             area
##
     <chr>
             <chr>
                                  <dbl>
                                             <dbl> <chr>
                                                            <dbl>
                                                                       <dbl>
## 1 ALABAMA Cities ~
                                  0.966
                                            520422 Aggrav~ 2.84e+3
                                                                        2914
## 2 ALABAMA Cities ~
                                  0.966
                                            520422 Burgla~ 4.17e+3
                                                                        4275
## 3 ALABAMA Cities ~
                                  0.966
                                            520422 Larcen~ 1.43e+4
                                                                       14641
## 4 ALABAMA Cities ~
                                  0.966
                                            520422 Motor ~ 1.34e+3
                                                                        1375
                                            520422 Murder~ 4.10e+1
## 5 ALABAMA Cities ~
                                                                          42
                                0.966
## 6 ALABAMA Cities ~
                                0.966 520422 Popula~ 9.66e-1
                                                                           1
## 7 ALABAMA Cities ~
                                            520422 Proper~ 1.98e+4
                                                                       20291
                                0.966
## 8 ALABAMA Cities ~
                                 0.966
                                            520422 Rape (~ 1.87e+2
                                                                         193
## 9 ALABAMA Cities ~
                                  0.966
                                            520422 Rape (~ 2.63e+2
                                                                         269
## 10 ALABAMA Cities ~
                                  0.966
                                            520422 Robbery 4.10e+2
                                                                         421
```

... with 1,470 more rows