…In this post I'll address how I used `{KableExtra}` to nicely print a frequency table of the categorical & ordinal questions I had in my survey. You can also do what I describe below in other packages, however I enjoy using `{KableExtra}` for its rich vignette and clearly defomed functions.

The Problem

In my pilot survey I had ~20 questions that were categorical, ordinal and were simple constructs not requiring a thorough analysis but only a quick review at the distribution of responses. I wanted to print all of these variables in one formatted table and address any anomalies if needed.

The issue was, how can I format printing of all categorical variables in their chronological order, along with the original question and the distribution of responses?

Basically, how can I achieve the following output:

Explore our data

First, let's loda the packages we'll need and look at our data:

```
library(tidyverse)
library(here)
library(readxl)
library(knitr)
library(kableExtra)
library(janitor)
library(scales)

df_survey <- read_xlsx(here("content", "post", "printing-survey-table",
"data","survey.xlsx"))
head(df_survey)
```

```
## # A tibble: 6 x 12
##   Q2     Q4     Q6     Q7    Q11_1   Q13_1   Q15   Q26   Q27   Q30   Q32
Q37
##
## 1 Female Salar~ Simil~ Cent~ Once a~ At lea~ Never No        No    Yes
## 2 Female Salar~ Below~ South Severa~ Every ~ Never No        No    Yes
## 3 Female Stude~ Below~ South Betwee~ I've n~ Never No        No    Yes
## 4 Female Unemp~ Simil~ Cent~ Once a~ At lea~ Last~ No        No    Yes
## 5 Male   Salar~ Simil~ Cent~ Once a~ At lea~ Last~ Yes   2-3 ~ Yes   No    No
## 6 Female Salar~ Simil~ Cent~ Severa~ Last t~ Never No        No    Yes
```

So we have a lot of information describing our sample data records. While this data is fabricated, it mirrors a common survey dataset: Each row represents a respondent with answers to various questions. With respect to continuous variables I did a different analysis, so for the purpose of the following post we'll need only character columns. Let's start by removing anything other than the relevant columns:

```
df_char <- df_survey %>%
  janitor::clean_names () %>%
  select_if(is.character)
```

We'll need to change the data to a long form so that we can print it for efficient reading. An easy approach will be to use the `pivot_longer` argument, rendering all our columns in one long table:

```
df_long <- df_char %>%
  pivot_longer(q2:q37, names_to = "question") %>%
  count(question, value) %>%
  group_by(question) %>%
  mutate(pct = percent(n/sum(n)))

# Print table
kbl(df_long) %>%
  kable_styling() %>%
  scroll_box(height = "550px")
```

question	value	n	pct
q11_1	Between once a week to once in a month	5	14.7%
q11_1	Less than once a month	4	11.8%
q11_1	Once a day	6	17.6%
q11_1	Once a week	4	11.8%
q11_1	Several times a day	6	17.6%
q11_1	Several times a week	8	23.5%
q11_1	NA	1	2.9%
q13_1	About every week	1	2.9%
q13_1	At least once a month	13	38.2%
q13_1	Every three months	5	14.7%
q13_1	Every three to six months	4	11.8%
q13_1	I've never bought	5	14.7%
q13_1	Last time I bought was over half a year ago	6	17.6%
q15	Last half a year	3	8.8%
q15	Last month	2	5.9%
q15	Last three months	3	8.8%
q15	Last week	1	2.9%
q15	Last year	1	2.9%
q15	Never	18	52.9%
q15	Over a year ago	6	17.6%
q2	Female	15	44.1%
q2	Male	17	50.0%

Great, this prints nicely, but we're left with several issues to address[2]:

1. We're missing the actual questions – Notice how we only have "q11_1" but not a description of what the actual question or what the variable is. You can add a question label within Qualtrics, but I still wanted to have the question itself presented along with the question number.

2. Questions were reorganized alphabetically – Once we ran the `pivot_longer` R sorted our dataframe alphabetically according to the question column, but we might want it ordered according to the survey layout. Of course this is contingent on your data; I wanted to present it aligned to the order of the survey questions.

3. Some of the responses aren't ordered – Notice how some of the responses are randomly ordered, when ideally we'd want them to be ordered by hierarchy. For example question q11_1 describes frequency responses that aren't hierarchically ordered.

4. Remove duplicated information – Our question column, and if we add another one with the question's text, will have duplicate information. While the value changes within questions printing the question column for each row is redundant. In addition, once we'll add the question title it'll be even more cluttered and any additional irrelevant text should be removed.

So then, let's address these issues individually.

Adding information to our questions

When you download the survey data from Qualtrics you also receive it with the original questions. When I personally analyzed the data I removed it, but here it's perfect for our display of additional information. There's also a great function for doing exactly that from the `{qualtRics}` package, but I was having trouble connecting to the platform's API through my Qualtrics user.

Adding the questions was straightforward: Just combine the current *data_long* with a dataset containing my questions. We'll use a copy of the original survey data (of course fabricated for purpose of the survey) that only contains the questions:

```
df_questions <- read_xlsx(here("content", "post", "printing-survey-table",
"data", "questions.xlsx"))
df_questions[,1:3]
```

```
## # A tibble: 1 x 3
##   Q2     Q4                   Q6
##
## 1 Gender What's your occupat~ The average income for an individual is X,
you're~
```

Great! we see our question's text as values with the variables being the questions themselves. Now let's render it in a long format so that each row is a question id with the corresponding text as a value, and then we'll join it with our current dataset of answers:

```
df_q_clean <- df_questions %>%
  clean_names() %>%
  pivot_longer(cols = q2:q37, names_to = "question", values_to = "text")

df_long_joined <-  left_join(df_long, df_q_clean)

head(df_long_joined)
```

```
## # A tibble: 6 x 5
## # Groups:   question [1]
##   question value                           n pct   text
##
## 1 q11_1    Between once a week to onc~     5 14.7% Every how often do you
consu~
## 2 q11_1    Less than once a month          4 11.8% Every how often do you
consu~
## 3 q11_1    Once a day                      6 17.6% Every how often do you
consu~
## 4 q11_1    Once a week                     4 11.8% Every how often do you
consu~
## 5 q11_1    Several times a day             6 17.6% Every how often do you
consu~
## 6 q11_1    Several times a week            8 23.5% Every how often do you
consu~
```

Perfect. However, as you can see, our new text column provides the same information across the same questions, which seems kind of redundant. We'll keep it for now and address it soon when we turn to print our table.

Reordering within and across questions

The next issue on the list is that we want some of our questions to be organized not by the count frequency or some randomness, but by hierarchy of the answer options. For example 'a few times a day', 'Once a day', 'several times a week' and so on as a hierarachal structure in my ordinal variables.

Alas, I don't have a magical automated method and would be grateful to hear about other options you encountered or thought of. I thought of using factors to reorder the levels, but once I pivot my data into a long format the answers are again sorted alphabetically. Instead I decided to manually combine my current dataframe with an identical one I saved where I ranked each relevant ordinal question manually. Though a tedious task, this manual workload is more efficient than automating everything.

```
# Save the sorted response file and use that to rank
# write_csv(df_long_q_sorted, here("content", "post", "printing-survey-table",
"data", "answers_hir.csv"))

answer_hir <- read_csv(here("content", "post", "printing-survey-table", "data",
"answers_hir.csv"))
head(answer_hir)
```

```
## # A tibble: 6 x 3
##   question value                       rank
##
## 1 q2       Female                        NA
## 2 q2       Male                          NA
## 3 q4       Salaried employee             NA
## 4 q4       Self employed                 NA
## 5 q4       Student                       NA
## 6 q4       Student,Salaried employee     NA
```

We now have our new guide in which we ranked our questions. Notice that the first answers are NA, but that's because the nominal variables have no intrinsic hierarchy. Now let's use this dataframe to create a value with which to sort our answers:

```
df_long_ranked <- left_join(x = df_long_joined, y = answer_hir)
head(df_long_ranked)
```

```
## # A tibble: 6 x 6
## # Groups:   question [1]
##   question value                          n pct   text
rank
##
## 1 q11_1    Between once a week to o~      5 14.7% Every how often do you c~
5
## 2 q11_1    Less than once a month         4 11.8% Every how often do you c~
6
## 3 q11_1    Once a day                     6 17.6% Every how often do you c~
2
## 4 q11_1    Once a week                    4 11.8% Every how often do you c~
4
## 5 q11_1    Several times a day            6 17.6% Every how often do you c~
1
## 6 q11_1    Several times a week           8 23.5% Every how often do you c~
3
```

We'll reorder by rank along with solving the next issue which is the way our questions are ordered. Basically, we want it to be ordered by the question value and not using an alphabetic sort. For example, we'd like q_5 to appear before q11_1, similar to how it appeared in the survey. I'll apply some regex (regular expression) manipulation to capture only the numbers and use that to sort by.
Displayed as follows:

```
df_long_q_sorted <- df_long_ranked %>%
  mutate(q_num = str_remove_all(question, "[a-z]"),
    q_num = str_replace_all(q_num, "_", "."),
    q_num = str_remove(q_num, "\\.$"),
    q_num = as.numeric(q_num)) %>%
  group_by(question) %>%
  arrange(q_num, rank) %>%
  ungroup() %>%
  select(-c(q_num, rank)) %>%
  relocate(text, .after = question)

head(df_long_q_sorted)
```

```
## # A tibble: 6 x 5
##   question text                     value                 n pct
##
## 1 q2       Gender                   Female               15 44.1%
## 2 q2       Gender                   Male                 17 50.0%
## 3 q2       Gender                   Other                 2 5.9%
## 4 q4       What's your occupation? Salaried employee    18 52.9%
## 5 q4       What's your occupation? Self employed         3 8.8%
## 6 q4       What's your occupation? Student               8 23.5%
```

Fantastic!

I found that first using the regex and then sorting by rank doesn't properly work, so instead I implemented it along with sorting the questions. Again, if the order of questions and answers doesn't matter in your data you can just skip past some of the stages.

Great, now that we have all our data formatted properly, we can turn to the printing!

KableExtra

In my initial round of exploring the pilot survey I used `{KableExtra}` and its powerful features. You might find other packages better to work with when knitting to Word. With that said, it's possible (and very effective) to knit to Html and copy that into a Word document. Despite the copy + paste requirement, I found it to be the better approach for keeping all the aesthetics and formatting integrated in the original document. Oh, and I also had my questions originally in Hebrew which was easier to knit to Html altogether.

removing redundant information

As we saw earlier, the argument is pretty straight forward. We can address the redundant information we have – question and text column appearing with each answer (our final issue) – within the KableExtra object using `collapse_rows`:

```
df_long_q_sorted %>%
  kbl(col.names = c("Question", "Text", "Answer", "n", "%")) %>%
  kable_styling(full_width = F) %>%
  column_spec(1, bold = T) %>%
  collapse_rows(columns = c(1,2), valign = "top") %>%
  scroll_box(height = "750px")
```

Question	Text	Answer	n	%
q2	Gender	Female	15	44.1%
		Male	17	50.0%
		Other	2	5.9%
q4	What's your occupation?	Salaried employee	18	52.9%
		Self employed	3	8.8%
		Student	8	23.5%
		Student,Salaried employee	3	8.8%
		Student,Unemployed	1	2.9%
		Unemployed	1	2.9%
q6	The average income for an individual is X, you're income	Below average	16	47.1%

Question	Text	Answer	n	%
	is:	Similar to average	13	38.2%
		Above average	5	14.7%
q7	Where do you live in Israel	Center	16	47.1%
		South	18	52.9%
q11_1	Every how often do you consume chocolate?	Several times a day	6	17.6%
		Once a day	6	17.6%
		Several times a week	8	23.5%
		Once a week	4	11.8%
		Between once a week to once in a month	5	14.7%
		Less than once a month	4	11.8%
		NA	1	2.9%
q13_1	Every how often do you buy chocolate?	About every week	1	2.9%
		At least once a month	13	38.2%
		Every three months	5	14.7%
		Every three to six months	4	11.8%
		Last time I bought was over half a year ago	6	17.6%
		I've never bought	5	14.7%
q15	When did you last attend a party?	Last week	1	2.9%

That easy? Yes!

The trick here that I love is collapsing the column, an argument also common in other packages such as {formattable} I look forward to explore. Collapsing a column makes printing in rmarkdown really easy and efficient, something I found lacking in other platforms I learned such as SPSS. I also added a `column_spec` to bold the first column. Of course you can also remove the scroll box by not using the `scroll_box` option at the end, which will print your whole table.

If you're looking for a word format, you can just copy & paste your html output (Perfect for working with Hebrew text for example, a little more on that below). You can find an additional example by Hao Zhu, the creator of the package, or here's my attempt below:

Just select all and copy it into a word document

Figure 1: Just select all and copy it into a word document

If you want the table to be a little more formally formatted I recommend exploring aesthetic arguments such as `kable_classic`, `kable_minimal` and others from the {KableExtra} family. Here's a short example using the `kable_minimal` with an Html output:

```
df_long_q_sorted %>%
  head(10) %>%
  kbl() %>%
  kable_styling(full_width = F) %>%
  column_spec(1, bold = T) %>%
  collapse_rows(columns = c(1,2), valign = "top")
```

Example using kable_minimal as a table theme

Figure 2: Example using kable_minimal as a table theme

Wait, but what if I want a simple table in word?

Let’s say you want a simple kable table when knitting in word, you can just add ‘df_print: kable’ to the YAML of your document or alternatively, you can explore other options in the Rmarkdown book that elegantly print dataframes. How do we remove the redundant information when printing? Just replace the duplicated values with an empty string:

```
df_long_q_sorted %>%
  mutate(across(c(question, text), ~ ifelse(duplicated(.x), " ", .x))) %>%
  select(`Question` = question, `Text` = text, `Value` = value, n, `%` = pct) %>%
  head(20)
```

Outputting a table using df_print: kable in the YAML section

Figure 3: Outputting a table using df_print: kable in the YAML section

Voila!

I first removed redundant text by using the `across` along with a conditional argument to remove duplicated text. Basically the formula (`~ ifelse`) reads as take anyone of the specified columns and pass it to a conditional statement that if true (if the word is duplicated), add a space character instead. Below is a screenshot when rendered to word, and you can continue to format it with or without other packages.

Right-to-Left languages?

I found it difficult knitting Hebrew characters to a Word output but easily done when rendering Html documents. Here’s a short example, without the whole pre-processing, using some Hebrew questions:

```
hebrew_example <- read_xlsx(here("content", "post", "printing-survey-table", "data",
"hebrew_example.xlsx"))

hebrew_example %>%
  select(pct, n, value, text, question) %>%
  mutate(pct = percent(pct)) %>%
  # Reverse the order of questions
  kbl(col.names = c("%", "פריט" ,"שאלה" ,"תשובה" ,"שכיחות"), align = 'r') %>%
  kable_styling(full_width = F) %>%
  column_spec(5, bold = T) %>%
  collapse_rows(columns = c(4,5), valign = "top") %>%
  scroll_box(height = "500px")
```

%	שכיחות	תשובה	שאלה	פריט
47.1%	16	נקבה	מגדר	**q2**
52.9%	18	זכר		
52.9%	18	שכיר	סטטוס תעסוקתי	**q4**
8.8%	3	עצמאי		
23.5%	8	סטודנט		
8.8%	3	סטודנט, שכיר		
2.9%	1	סטודנט, מובטל		
2.9%	1	מובטל		
17.6%	6	לא	האם תסכים להשתתף במחקר המשך?	**q37**
73.5%	25	כן		
8.8%	3	חסר		

Notice how I reversed the columns direction to have it compatible for a right-to-left reading. I also changed the locale setting (not shown) using the `Sys.setlocale` argument. While this still needs some additional work (for example the ‘?’ isn’t aligned), it’s definitely a good start.