Assignment 6

Due Tuesday March 17 at 11:59pm on Quercus

The assignment is due on the date shown above. An assignment handed in after the deadline is late, and may or may not be accepted (see course outline). My solutions to the assignment questions will be available when everyone has handed in their assignment.

You are reminded that work handed in with your name on it must be *entirely your own work*.

Assignments are to be handed in on Quercus. See `https://www.utsc.utoronto.ca/~butler/c32/quercus1.nb.html` for instructions on handing in assignments in Quercus. Markers' comments and grades will be available there as well.

Begin with the usual:

```
library(tidyverse)
```

Hand in question 2 below. Question 3 is a bonus question, if you want an extra challenge. If you want the bonus points, hand in question 3 as well.

1. Work through chapter 13 of PASIAS.
2. The file `http://ritsokiguess.site/STAC33/xgrades.csv` contains a data frame with some marks for some students on some tests.
   (a) (2 marks) Read in the data frame and display at least some of it. (It has 12 rows).

   Solution:

```
my_url <- "http://ritsokiguess.site/STAC33/xgrades.csv"
grades <- read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   ID = col_double(),
##   Test = col_character(),
##   Year = col_double(),
##   Fall = col_double(),
##   Spring = col_double(),
##   Winter = col_double()
## )
```

```
grades
```

```
## # A tibble: 12 x 6
##       ID Test     Year  Fall Spring Winter
##    <dbl> <chr>   <dbl> <dbl>  <dbl>  <dbl>
##  1     1 Math     2008    15     16     19
##  2     1 Math     2009    12     13     27
##  3     1 Writing  2008    22     22     24
##  4     1 Writing  2009    10     14     20
##  5     2 Math     2008    12     13     25
##  6     2 Math     2009    16     14     21
##  7     2 Writing  2008    13     11     29
##  8     2 Writing  2009    23     20     26
##  9     3 Math     2008    11     12     22
## 10     3 Math     2009    13     11     27
## 11     3 Writing  2008    17     12     23
## 12     3 Writing  2009    14      9     31
```

(b) (6 marks) The instructor who awarded these marks wants to rearrange the data frame as shown below:

```
## # A tibble: 18 x 5
##       ID  Year Quarter  Math Writing
##    <dbl> <dbl> <chr>   <dbl>   <dbl>
##  1     1  2008 Fall       15      22
##  2     2  2008 Fall       12      13
##  3     3  2008 Fall       11      17
##  4     1  2008 Spring     16      22
##  5     2  2008 Spring     13      11
##  6     3  2008 Spring     12      12
##  7     1  2008 Winter     19      24
##  8     2  2008 Winter     25      29
##  9     3  2008 Winter     22      23
## 10     1  2009 Fall       12      10
## 11     2  2009 Fall       16      23
## 12     3  2009 Fall       13      14
## 13     1  2009 Spring     13      14
## 14     2  2009 Spring     14      20
## 15     3  2009 Spring     11       9
## 16     1  2009 Winter     27      20
```

```
## 17     2  2009 Winter     21       26
## 18     3  2009 Winter     27       31
```

By making the data frame longer and/or wider or using other tools as appropriate, convert the data frame you read in in the previous part to be laid out this way.

Solution: I deliberately left this for you to figure out. I'm expecting you to try some things and see whether they bring you closer to the answer.

Usually the place to start is a `pivot_longer`, and the place to begin seems to be to get all the marks in one column. Think first about how you would make the data tidier for yourself; those last three columns are all marks, just for different things:

```
grades %>%
  pivot_longer(Fall:Winter, names_to = "Quarter",
    values_to = "Score")
```

```
## # A tibble: 36 x 5
##       ID Test     Year Quarter Score
##    <dbl> <chr>   <dbl> <chr>   <dbl>
##  1     1 Math     2008 Fall       15
##  2     1 Math     2008 Spring     16
##  3     1 Math     2008 Winter     19
##  4     1 Math     2009 Fall       12
##  5     1 Math     2009 Spring     13
##  6     1 Math     2009 Winter     27
##  7     1 Writing  2008 Fall       22
##  8     1 Writing  2008 Spring     22
##  9     1 Writing  2008 Winter     24
## 10     1 Writing  2009 Fall       10
## # ... with 26 more rows
```

I happened to note that the column with the semesters in it needed to be called `Quarter`, but if you want to call it `semester`, that's good too. Two marks for getting that far.

If we take this and make it wider by splitting `Test` out into its own columns, one for `Math` and one for `Writing`, this is going to get us close. This one is `pivot_wider`:

```
grades %>%
  pivot_longer(Fall:Winter, names_to = "Quarter",
    values_to = "Score") %>%
  pivot_wider(names_from="Test", values_from="Score")
```

```
## # A tibble: 18 x 5
##       ID  Year Quarter  Math Writing
##    <dbl> <dbl> <chr>   <dbl>   <dbl>
##  1     1  2008 Fall       15      22
##  2     1  2008 Spring     16      22
##  3     1  2008 Winter     19      24
##  4     1  2009 Fall       12      10
##  5     1  2009 Spring     13      14
##  6     1  2009 Winter     27      20
##  7     2  2008 Fall       12      13
##  8     2  2008 Spring     13      11
##  9     2  2008 Winter     25      29
## 10     2  2009 Fall       16      23
## 11     2  2009 Spring     14      20
## 12     2  2009 Winter     21      26
## 13     3  2008 Fall       11      17
## 14     3  2008 Spring     12      12
## 15     3  2008 Winter     22      23
## 16     3  2009 Fall       13      14
## 17     3  2009 Spring     11       9
## 18     3  2009 Winter     27      31
```

Another two marks for getting that far.

Good. But now, *pay careful attention*: if you look at what we are aiming for, all the 2008 values are first, then the Quarters within 2008, then the students within that. So we need to do some sorting: by year, quarter and ID in that order:

```
grades %>%
  pivot_longer(Fall:Winter, names_to = "Quarter",
    values_to = "Score") %>%
  pivot_wider(names_from="Test", values_from="Score") %>%
  arrange(Year, Quarter, ID)
```

```
## # A tibble: 18 x 5
##       ID  Year Quarter  Math Writing
##    <dbl> <dbl> <chr>   <dbl>   <dbl>
##  1     1  2008 Fall       15      22
##  2     2  2008 Fall       12      13
##  3     3  2008 Fall       11      17
##  4     1  2008 Spring     16      22
##  5     2  2008 Spring     13      11
##  6     3  2008 Spring     12      12
##  7     1  2008 Winter     19      24
##  8     2  2008 Winter     25      29
##  9     3  2008 Winter     22      23
## 10     1  2009 Fall       12      10
## 11     2  2009 Fall       16      23
## 12     3  2009 Fall       13      14
## 13     1  2009 Spring     13      14
## 14     2  2009 Spring     14      20
## 15     3  2009 Spring     11       9
## 16     1  2009 Winter     27      20
## 17     2  2009 Winter     21      26
## 18     3  2009 Winter     27      31
```

And that is what we were aiming for.

I think it is unlikely that you will be able to make it work by going wider and then longer, but if you can, and you can get to the right place, four marks for an appropriate pivoting and two for an appropriate sorting.

3. This is a bonus question; there are 4 bonus points for a complete answer to this one, which, if earned, allow you to score more than the maximum for this assignment. If you want a shot at the bonus points, hand in your answer to this one as well.

   The Toronto Wolfpack play rugby league, in a league with a lot of English teams (and one French one). They play at Lamport Stadium on King. The file `http://www.utsc.utoronto.ca/~butler/assgt_data/rl.txt` contains some scores from the league that the Wolfpack play in, along with some other leagues. Unfortunately, the data are rather untidy, so we have a fair bit of tidying work to do. Our aim is to create a data frame with the following columns: the date on which each game was played (as text), the name of the home team, the score of the home team, the name of the away team, the score of the away team, and a code for the league in which each game was played (the things like `CH` and `L1` that you see at the end of the line in the data file). Note that the team names have a variable number of words (compare Bradford Bulls and York City Knights, for example). You'll also have to deal with some of the rows being dates and some of them being game results without dates (how do you tell the difference?)

   (a) The data file has one column that has *no* column name. Pretend that the file is a `.csv`, read it in, and display some of the data frame. What name has the one column acquired?

Solution:

Remember (or find out) that a file with no column names is read in with `col_names=F`:

```
my_url <- "http://www.utsc.utoronto.ca/~butler/assgt_data/rl.txt"
results <- read_csv(my_url, col_names=F)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character()
## )
```

```
results
```

```
## # A tibble: 29 x 1
##    X1
##    <chr>
##  1 Sun 21st Jul
##  2 Manly Sea Eagles 36 - 24 Parramatta Eels NRL
##  3 Gold Coast Titans 18 - 38 Melbourne Storm NRL
##  4 Widnes Vikings 19 - 24 Toronto Wolfpack CH
##  5 Swinton Lions 30 - 12 Barrow Raiders CH
##  6 Leigh Centurions 48 - 12 Batley Bulldogs CH
##  7 Featherstone Rovers 50 - 6 Rochdale Hornets CH
##  8 Dewsbury Rams 28 - 28 Halifax RLFC CH
##  9 York City Knights 25 - 24 Bradford Bulls CH
## 10 Workington Town 52 - 4 Keighley Cougars L1
## # ... with 19 more rows
```

The column is called `X1`.

You'll know if you did it right, because the first row should be a date, and if you forgot the `col_names=F`, that'll end up as the name of the column.

(b) Construct a (rather long) pipeline that converts the data frame you read in from the file into the desired format. There is some flexibility about how you do this, but you might want to use some of the following tools. If you have not seen them before, you'll need to find out how they work:

- `mutate` (you'll probably use this a lot)
- `str_detect`
- `ifelse`
- `fill`
- `filter`
- `select`
- `separate`
- `str_count` for counting words
- `word` for extracting words from text
- `extract` if you are clever with regular expressions

You should hand in your pipeline code and the output it produces.

Solution: This is what I did (other things are undoubtedly possible):

```
results %>%
  mutate(is_result=str_detect(X1, " - ")) %>%
  mutate(date=ifelse(is_result, NA, X1)) %>%
  fill(date) %>%
  filter(is_result) %>%
  select(-is_result) %>%
  separate(X1, into=c("home_stuff", "away_stuff"), sep=" - ") %>%
  mutate(n_words_home=1+str_count(home_stuff, " ")) %>%
  mutate(home_score=word(home_stuff, n_words_home)) %>%
  mutate(home_team_name=word(home_stuff, 1, n_words_home-1)) %>%
  mutate(n_words_away=1+str_count(away_stuff, " ")) %>%
  mutate(away_score=word(away_stuff, 1)) %>%
  mutate(away_team_name=word(away_stuff, 2, n_words_away-1)) %>%
  mutate(league=word(away_stuff, n_words_away)) %>%
  select(-home_stuff, -away_stuff, -n_words_home, -n_words_away) -> results_tidy
results_tidy
```

```
## # A tibble: 25 x 6
##    date        home_score home_team_name       away_score away_team_name  league
##    <chr>       <chr>      <chr>                <chr>      <chr>           <chr>
##  1 Sun 21st J~ 36         Manly Sea Eagles     24         Parramatta Eels NRL
##  2 Sun 21st J~ 18         Gold Coast Titans    38         Melbourne Storm NRL
##  3 Sun 21st J~ 19         Widnes Vikings       24         Toronto Wolfpa~ CH
##  4 Sun 21st J~ 30         Swinton Lions        12         Barrow Raiders  CH
##  5 Sun 21st J~ 48         Leigh Centurions     12         Batley Bulldogs CH
##  6 Sun 21st J~ 50         Featherstone Rovers  6          Rochdale Horne~ CH
##  7 Sun 21st J~ 28         Dewsbury Rams        28         Halifax RLFC    CH
##  8 Sun 21st J~ 25         York City Knights    24         Bradford Bulls  CH
##  9 Sun 21st J~ 52         Workington Town      4          Keighley Couga~ L1
## 10 Sun 21st J~ 0          North Wales Crusade~ 30         Doncaster RLFC  L1
## # ... with 15 more rows
```

Display your code and the output it produces. If the output is correct and it looks plausible that it would come from your code, you are good. (If your output could not have come from your code, expect a fast zero for this question.)

I have another way below.

Detailed explanation:

- the first two `mutate`s: Each row of the data file is either a date or a game result. The game results all have a space followed by a minus sign followed by another space in them, and the dates don't. I created a new column `is_result` that is TRUE if the line of the file is a game result and FALSE otherwise. (`str_detect` is a way of checking whether one piece of text (its first input) contains another smaller piece of text (its second input). It returns TRUE if the second thing is found in the first thing, and FALSE otherwise.)

  Next, I made a new column called `date` that is the thing in `X1` if it is not a result and NA (missing) otherwise. `ifelse` takes three inputs, (i) something that is either true or false, (ii) the value to return if it is true, (iii) the value to return if it is false. Hint: `NA` standing for a missing value does *not* have quotes around it.

  This one is a little tricky because you want `date` to have a missing value if `is_result`

is TRUE, and the value of `X1` if `is_result` is FALSE, which might be the opposite way around from what you're thinking.

- I used `fill` to fill in the missing values in `date` with the non-missing date above them (which will be the date on which each game was played). This is as simple as feeding `fill` a column name, and it replaces all the missings with the previous non-missing value in that column.

  Extra: these dates are actually 2019. They could be turned into genuine R dates by using `dmy` from the `lubridate` package (because they will be a day, a month and a year if we glue the year on the end):

```
library(lubridate)
```

```
##
## Attaching package:  'lubridate'
```

```
## The following object is masked from 'package:base':
##
##    date
```

```
results %>%
  mutate(is_result=str_detect(X1, " - ")) %>%
  mutate(date=ifelse(is_result, NA, X1)) %>%
  fill(date) %>%
  mutate(date=dmy(str_c(date, " 2019")))
```

```
## # A tibble: 29 x 3
##    X1                                            is_result date
##    <chr>                                         <lgl>     <date>
##  1 Sun 21st Jul                                  FALSE     2019-07-21
##  2 Manly Sea Eagles 36 - 24 Parramatta Eels NRL   TRUE      2019-07-21
##  3 Gold Coast Titans 18 - 38 Melbourne Storm NRL  TRUE      2019-07-21
##  4 Widnes Vikings 19 - 24 Toronto Wolfpack CH     TRUE      2019-07-21
##  5 Swinton Lions 30 - 12 Barrow Raiders CH        TRUE      2019-07-21
##  6 Leigh Centurions 48 - 12 Batley Bulldogs CH    TRUE      2019-07-21
##  7 Featherstone Rovers 50 - 6 Rochdale Hornets CH TRUE      2019-07-21
##  8 Dewsbury Rams 28 - 28 Halifax RLFC CH          TRUE      2019-07-21
##  9 York City Knights 25 - 24 Bradford Bulls CH    TRUE      2019-07-21
## 10 Workington Town 52 - 4 Keighley Cougars L1     TRUE      2019-07-21
## # ... with 19 more rows
```

  but we haven't done dates and times, so I didn't expect you to do this.

- I kept only the rows that are game results (getting rid of the ones that are dates), and got rid of the column `is_result` (we don't need it any more). The rows we want to keep are the ones where `is_result` is TRUE.

  The `filter` needs to be first, because if you get rid of `is_result` first, you won't have any idea about which rows to keep!

- Now I tackle the stuff in column `X1`. This is: the name of the home team, the score of the home team (points), space-minus-space, the score of the away team, the name of the away team, and an abbreviation for the league the game was played in. First, I use `separate` to split `X1` into the stuff before the space-minus-space, called `home_stuff`, and the stuff after, called `away_stuff`. You need to give `separate` the input `into` to say what

the separated pieces are going to be called, and the input `sep` to say what separates the first piece from the second piece. In `separate`, `sep` can be a number (eg. `sep=2` would mean "separate after the second character") or a piece of text (eg. `sep="?"` would mean separate the stuff before the `?` from the stuff after it.)

Using `sep="-"` would also work, but would give you some extra space characters (on the end of `home_stuff` and on the beginning of `away_stuff`). This will mess up your counting of words down below, and, if you are not aware (and careful) will produce things that make no sense. For example:

```
d <- tribble(
    ~a,
    "hello - you guys",
    "you guys - hello"
)
d %>% separate(a, into=c("before", "after"), sep=" - ") %>%
    mutate(c1=str_count(before, " "), c2=str_count(after, " "))
```

```
## # A tibble: 2 x 4
##   before   after       c1    c2
##   <chr>    <chr>    <int> <int>
## 1 hello    you guys     0     1
## 2 you guys hello        1     0
```

Compare that with this:

```
d %>% separate(a, into=c("before", "after"), sep="-") %>%
    mutate(c1=str_count(before, " "), c2=str_count(after, " "))
```

```
## # A tibble: 2 x 4
##   before      after          c1    c2
##   <chr>       <chr>       <int> <int>
## 1 "hello "    " you guys"     1     2
## 2 "you guys " " hello"        2     1
```

The second version gets the number of words right, but by accident, because there is actually an extra space at the beginning or end of the text. So expect to lose something here for `sep="-"` as opposed to `sep=" - "`.

- `home_stuff` contains the name of the home team (a variable number of words) and the home team's score (the last "word"). The function `str_count` counts the number of instances of any character (its second input) in a piece of text. I used this to count how many words each entry in `home_stuff` has, including the numbers at the end. The number of words is, keep in mind, *one more* than the number of spaces: how many words and spaces are there in `Hello World`?

- The function `word` looks like this, for example: `word(x, 1, 3)`. This would take the text in `x`, and return the first through third (inclusive) words in it. (`word(x, 2)` returns just the second word of `x`.) The home team's score is the last word of `home_stuff`, and you know how many words `home_stuff` has because you worked it out in the previous part. I created a column containing just the home team's score, for which I need the last word, which is the word whose number appears in what I called `n_words_home`.

  This extracts the home team's score *as text* (see the top of the column). You could (here at least) use `parse_number` to pull out the only numbers in `home_stuff`, but that

would get defeated by a team *name* that contained a number, like "Bassenthwaite RLFC (1895)".[1]

- The name of the home team is all the words of `home_stuff` except for the last one (which is the home team's score): that is, if there are n words in `home_stuff`, the home team's name is words 1 through $n-1$ inclusive. I used `word` to make a column containing the home team's name.

- I made a column containing the number of words in `away_stuff`, using `str_count` again.

- I used the same ideas to extract (i) the away team's score, (ii) the away team's name, (iii) the league abbreviation from `away_stuff`. I got rid of the columns `home_stuff` and `away_stuff`, and saved the resulting data frame.

  This was a lot to do, but all the same ideas as before. In `away_stuff`, the first word is the score, words 2 through `n_words_away-1` is the team name, and word number `n_words_away` is the league abbreviation.

  Extra: this is not quite good, because two of the new columns you obtained are not displayed here. (On yours, you should be able to click the right-arrow to see them.)

  If you want the columns in a sensible order, `select` the columns you want in the order you want them, eg.:

```
results_tidy %>%
  select(date, home_team_name, home_score,
         away_team_name, away_score, league) -> results_tidy
results_tidy
```

```
## # A tibble: 25 x 6
##    date        home_team_name       home_score away_team_name  away_score league
##    <chr>       <chr>                <chr>      <chr>           <chr>      <chr>
##  1 Sun 21st J~ Manly Sea Eagles     36         Parramatta Eels 24         NRL
##  2 Sun 21st J~ Gold Coast Titans    18         Melbourne Storm 38         NRL
##  3 Sun 21st J~ Widnes Vikings       19         Toronto Wolfpa~ 24         CH
##  4 Sun 21st J~ Swinton Lions        30         Barrow Raiders  12         CH
##  5 Sun 21st J~ Leigh Centurions     48         Batley Bulldogs 12         CH
##  6 Sun 21st J~ Featherstone Rovers  50         Rochdale Horne~ 6          CH
##  7 Sun 21st J~ Dewsbury Rams        28         Halifax RLFC    28         CH
##  8 Sun 21st J~ York City Knights    25         Bradford Bulls  24         CH
##  9 Sun 21st J~ Workington Town      52         Keighley Couga~ 4          L1
## 10 Sun 21st J~ North Wales Crusade~ 0          Doncaster RLFC  30         L1
## # ... with 15 more rows
```

Extra: I said there was another way. If you know about regular expressions as a means of matching and otherwise wrangling text, you might have suspected that there was a way of using them here. The secret is to use `extract`. This uses "capture groups" to match pieces of text and to pull them out into variables. Here's how it works here:

```
results %>%
  mutate(is_result=str_detect(X1, " - ")) %>%
  mutate(date=ifelse(is_result, NA, X1)) %>%
  fill(date) %>%
  filter(is_result) %>%
  select(-is_result) %>%
  extract(X1, into=c("home_team", "home_score", "away_score",
                     "away_team", "league"),
          regex="^(.*) ([0-9]+) - ([0-9]+) (.*) (.{2,3})$")
```

```
## # A tibble: 25 x 6
##    home_team            home_score away_score away_team       league date
##    <chr>                <chr>      <chr>      <chr>           <chr>  <chr>
##  1 Manly Sea Eagles     36         24         Parramatta Eels NRL    Sun 21st J~
##  2 Gold Coast Titans    18         38         Melbourne Storm NRL    Sun 21st J~
##  3 Widnes Vikings       19         24         Toronto Wolfpa~ CH     Sun 21st J~
##  4 Swinton Lions        30         12         Barrow Raiders  CH     Sun 21st J~
##  5 Leigh Centurions     48         12         Batley Bulldogs CH     Sun 21st J~
##  6 Featherstone Rovers  50         6          Rochdale Horne~ CH     Sun 21st J~
##  7 Dewsbury Rams        28         28         Halifax RLFC    CH     Sun 21st J~
##  8 York City Knights    25         24         Bradford Bulls  CH     Sun 21st J~
##  9 Workington Town      52         4          Keighley Couga~ L1     Sun 21st J~
## 10 North Wales Crusade~ 0          30         Doncaster RLFC  L1     Sun 21st J~
## # ... with 15 more rows
```

Gosh, for me that actually worked *first time*!

You can see that it's a lot more compact than the first way, but it depends on your being able to wrangle regular expressions. The layout of `extract` is you start with the column you want to pull things out of, then the names that the pulled-out pieces are going to have, and then a regular expression that captures them. The brackets in the regular expression are the usual capture groups; since I am creating five new columns, I need five capture groups in the regular expression. My regular expression says:

- start at the beginning of `X1`
- after that, match any number of any characters (and capture them)
- then a space
- one or more digits (captured)
- a space, a dash and a space
- one or more digits again (captured)
- a space
- any text (captured) followed by a space
- exactly two or three of any characters (captured)
- the end of the string.

You might be wondering how "any number of any characters" will capture a team name and not the stuff after it (eg. the numbers). The answer to that is that the regular expression parser will find a way to make the whole regular expression match if it can, even if it has to try several different possibilities to do it. (Ask a computer scientist how this works.) The last thing I did

was to notice that the league abbreviations were always two or three characters, and I didn't want those to get mixed up with the away team's name. I could probably have been less careful about this (probably "the last word" would have done it); again, I would probably be defeated by team names with numbers *in* them. But it works here.

Finally, `extract` gets rid of the column `X1` from which we were pulling stuff, on the (reasonable) assumption that we probably don't need it any more. (If you really want to keep it, there's an option in `extract` to do so.)

(c) Now we can finally do some analysis. How many games were played on each date?

Solution:

```
results_tidy %>% count(date)
```

```
## # A tibble: 4 x 2
##   date             n
##   <chr>        <int>
## 1 Fri 19th Jul     3
## 2 Sat 20th Jul     6
## 3 Sun 21st Jul    14
## 4 Thu 18th Jul     2
```

(d) What were the two highest scores obtained by home teams, and which teams obtained them? Hint: sorting.

Solution:

Sort the data frame in descending order by home team's scores:

```
results_tidy %>%
  arrange(desc(home_score))
```

```
## # A tibble: 25 x 6
##    date       home_team_name      home_score away_team_name    away_score league
##    <chr>      <chr>               <chr>      <chr>             <chr>      <chr>
##  1 Sat 20th ~ Toulouse Olympique  56         Sheffield Eagles  18         CH
##  2 Sun 21st ~ Workington Town     52         Keighley Cougars  4          L1
##  3 Sun 21st ~ Featherstone Rovers 50         Rochdale Hornets  6          CH
##  4 Sun 21st ~ Leigh Centurions    48         Batley Bulldogs   12         CH
##  5 Sat 20th ~ Sydney Roosters     48         Newcastle Knights 10         NRL
##  6 Thu 18th ~ Wigan Warriors      46         Wakefield Trinity 16         SL
##  7 Sat 20th ~ West Wales Raiders~ 44         Coventry Bears    16         L1
##  8 Sun 21st ~ Salford Red Devils  40         Catalan Dragons   14         SL
##  9 Fri 19th ~ Penrith Panthers    40         St George Illawa~ 18         NRL
## 10 Sun 21st ~ Manly Sea Eagles    36         Parramatta Eels   24         NRL
## # ... with 15 more rows
```

Toulouse, 56 points, and Workington Town, 52.

Extra: because we sorted the whole data frame, the high scoring home teams and their opposition are still associated, so we know that Sheffield and Keighley were on the wrong end of big defeats.

(e) Which were the two lowest away scores, and the teams that scored them? Try the obvious idea first, then find out what goes wrong with it and then fix it (hint: turn text into numbers).

Solution:

The same idea is the one to try first:

```
results_tidy %>%
  arrange(away_score)
```

```
## # A tibble: 25 x 6
##    date        home_team_name      home_score away_team_name   away_score league
##    <chr>       <chr>               <chr>      <chr>            <chr>      <chr>
##  1 Sat 20th J~ Sydney Roosters     48         Newcastle Knigh~ 10         NRL
##  2 Sun 21st J~ Swinton Lions       30         Barrow Raiders   12         CH
##  3 Sun 21st J~ Leigh Centurions    48         Batley Bulldogs  12         CH
##  4 Sun 21st J~ London Broncos      32         St Helens        12         SL
##  5 Sat 20th J~ Canberra Raiders    20         Wests Tigers     12         NRL
##  6 Sun 21st J~ Salford Red Devils  40         Catalan Dragons  14         SL
##  7 Sat 20th J~ Newcastle Thunder   34         London Skolars   16         L1
##  8 Sat 20th J~ West Wales Raiders~ 44         Coventry Bears   16         L1
##  9 Thu 18th J~ Wigan Warriors      46         Wakefield Trini~ 16         SL
## 10 Sun 21st J~ Castleford Tigers   27         Warrington Wolv~ 18         SL
## # ... with 15 more rows
```

If you scroll down, you'll see that there are some lower scores (single digits) that didn't get sorted properly. The reason for that is that the away scores are *text* even though they look like numbers (look at the top of the column) and so have been sorted into "alphabetical order" where 10 comes before 8 (the first character of "10" is a 1, which is alphabetically before the first character of "8".)

So, to fix it, we need to turn the away scores into numbers first, either like this:

```
results_tidy %>%
  mutate(away_score=as.numeric(away_score)) %>%
  arrange(away_score)
```

```
## # A tibble: 25 x 6
##    date        home_team_name     home_score away_team_name    away_score league
##    <chr>       <chr>              <chr>      <chr>                  <dbl> <chr>
##  1 Sun 21st J~ Workington Town    52         Keighley Cougars           4 L1
##  2 Sun 21st J~ Featherstone Rove~ 50         Rochdale Hornets           6 CH
##  3 Thu 18th J~ Brisbane Broncos   28         Canterbury Bulld~          6 NRL
##  4 Sat 20th J~ Sydney Roosters    48         Newcastle Knights         10 NRL
##  5 Sun 21st J~ Swinton Lions      30         Barrow Raiders            12 CH
##  6 Sun 21st J~ Leigh Centurions   48         Batley Bulldogs           12 CH
##  7 Sun 21st J~ London Broncos     32         St Helens                 12 SL
##  8 Sat 20th J~ Canberra Raiders   20         Wests Tigers              12 NRL
##  9 Sun 21st J~ Salford Red Devils 40         Catalan Dragons           14 SL
## 10 Sat 20th J~ Newcastle Thunder  34         London Skolars            16 L1
## # ... with 15 more rows
```

or directly do the conversion in the `arrange`:

```
results_tidy %>%
  arrange(as.numeric(away_score))
```

```
## # A tibble: 25 x 6
##    date        home_team_name     home_score away_team_name    away_score league
##    <chr>       <chr>              <chr>      <chr>             <chr>      <chr>
##  1 Sun 21st J~ Workington Town    52         Keighley Cougars  4          L1
##  2 Sun 21st J~ Featherstone Rove~ 50         Rochdale Hornets  6          CH
##  3 Thu 18th J~ Brisbane Broncos   28         Canterbury Bulld~ 6          NRL
##  4 Sat 20th J~ Sydney Roosters    48         Newcastle Knights 10         NRL
##  5 Sun 21st J~ Swinton Lions      30         Barrow Raiders    12         CH
##  6 Sun 21st J~ Leigh Centurions   48         Batley Bulldogs   12         CH
##  7 Sun 21st J~ London Broncos     32         St Helens         12         SL
##  8 Sat 20th J~ Canberra Raiders   20         Wests Tigers      12         NRL
##  9 Sun 21st J~ Salford Red Devils 40         Catalan Dragons   14         SL
## 10 Sat 20th J~ Newcastle Thunder  34         London Skolars    16         L1
## # ... with 15 more rows
```

This converts the text away scores into numbers purely for the purpose of sorting them. It doesn't touch the `away_score` values themselves; they are still text.

Lowest scorers are Keighley (at Workington), Rochdale (at Featherstone) and Canterbury (at Brisbane), the last of these in the Australian league. Unsurprisingly, all three lost.

Extra: how would you work out the highest and lowest scores altogether, home and away teams both? You need to get all the scores together into one column, which suggests `gather`:

```
results_tidy %>%
  gather(venue, score, ends_with("score")) -> d
d
```

```
## # A tibble: 50 x 6
##    date         home_team_name        away_team_name   league venue      score
##    <chr>        <chr>                 <chr>            <chr>  <chr>      <chr>
##  1 Sun 21st Jul Manly Sea Eagles      Parramatta Eels  NRL    home_score 36
##  2 Sun 21st Jul Gold Coast Titans     Melbourne Storm  NRL    home_score 18
##  3 Sun 21st Jul Widnes Vikings        Toronto Wolfpack CH     home_score 19
##  4 Sun 21st Jul Swinton Lions         Barrow Raiders   CH     home_score 30
##  5 Sun 21st Jul Leigh Centurions      Batley Bulldogs  CH     home_score 48
##  6 Sun 21st Jul Featherstone Rovers   Rochdale Hornets CH     home_score 50
##  7 Sun 21st Jul Dewsbury Rams         Halifax RLFC     CH     home_score 28
##  8 Sun 21st Jul York City Knights     Bradford Bulls   CH     home_score 25
##  9 Sun 21st Jul Workington Town       Keighley Cougars L1     home_score 52
## 10 Sun 21st Jul North Wales Crusaders Doncaster RLFC   L1     home_score 0
## # ... with 40 more rows
```

and then

```
d %>% arrange(desc(as.numeric(score)))
```

```
## # A tibble: 50 x 6
##    date         home_team_name        away_team_name      league venue     score
##    <chr>        <chr>                 <chr>               <chr>  <chr>     <chr>
##  1 Sat 20th Jul Toulouse Olympique    Sheffield Eagles    CH     home_sco~ 56
##  2 Sun 21st Jul Workington Town       Keighley Cougars    L1     home_sco~ 52
##  3 Sun 21st Jul Featherstone Rovers   Rochdale Hornets    CH     home_sco~ 50
##  4 Sun 21st Jul Leigh Centurions      Batley Bulldogs     CH     home_sco~ 48
##  5 Sat 20th Jul Sydney Roosters       Newcastle Knights   NRL    home_sco~ 48
##  6 Thu 18th Jul Wigan Warriors        Wakefield Trinity   SL     home_sco~ 46
##  7 Sat 20th Jul West Wales Raiders RL Coventry Bears      L1     home_sco~ 44
##  8 Sun 21st Jul Salford Red Devils    Catalan Dragons     SL     home_sco~ 40
##  9 Fri 19th Jul Penrith Panthers      St George Illawarra NRL    home_sco~ 40
## 10 Sun 21st Jul Gold Coast Titans     Melbourne Storm     NRL    away_sco~ 38
## # ... with 40 more rows
```

The highest away score was the Melbourne Storm at the Gold Coast Titans.

Or:

```
d %>% arrange(as.numeric(score))
```

```
## # A tibble: 50 x 6
##    date         home_team_name        away_team_name      league venue     score
##    <chr>        <chr>                 <chr>               <chr>  <chr>     <chr>
##  1 Sun 21st Jul North Wales Crusaders Doncaster RLFC      L1     home_sco~ 0
##  2 Sun 21st Jul Workington Town       Keighley Cougars    L1     away_sco~ 4
##  3 Sun 21st Jul Featherstone Rovers   Rochdale Hornets    CH     away_sco~ 6
##  4 Thu 18th Jul Brisbane Broncos      Canterbury Bulldogs NRL    away_sco~ 6
##  5 Sat 20th Jul Sydney Roosters       Newcastle Knights   NRL    away_sco~ 10
##  6 Fri 19th Jul Hull KR               Huddersfield Giants SL     home_sco~ 12
##  7 Sun 21st Jul Swinton Lions         Barrow Raiders      CH     away_sco~ 12
##  8 Sun 21st Jul Leigh Centurions      Batley Bulldogs     CH     away_sco~ 12
##  9 Sun 21st Jul London Broncos        St Helens           SL     away_sco~ 12
## 10 Sat 20th Jul Canberra Raiders      Wests Tigers        NRL    away_sco~ 12
## # ... with 40 more rows
```

The North Wales Crusaders failed to score at home to Doncaster. A frustrating experience for the home fans. (The other low scores are Keighley at Workington and Rochdale at Featherstone that we saw before.)

A technique thing: I gave the tidied data frame a name, because I could see that I wanted to do two things with it. Pipes are linear, and this one bifurcates (splits into two), so I gave it a name at the point where it split.

Notes

[1]I don't think there are any rugby league team names like that, but if you were doing this with German soccer teams, there are quite a lot of those with numbers in their names, usually the year the club was formed. The best known of these is Schalke 04, who were formed in 1904 and play in Gelsenkirchen.