Choosing things in dataframes

Packages

The usual:

```
library(tidyverse)
```

Doing things with data frames

Let's go back to our Australian athletes:

```
athletes
```

```
# A tibble: 202 x 13
   Sex    Sport      RCC   WCC    Hc    Hg  Ferr   BMI   SSF
   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 female Netball   4.56  13.3  42.2  13.6    20  19.2  49
 2 female Netball   4.15   6    38    12.7    59  21.2 110.
 3 female Netball   4.16   7.6  37.5  12.3    22  21.4  89
 4 female Netball   4.32   6.4  37.7  12.3    30  21.0  98.3
 5 female Netball   4.06   5.8  38.7  12.8    78  21.8 122.
 6 female Netball   4.12   6.1  36.6  11.8    21  21.4  90.4
 7 female Netball   4.17   5    37.4  12.7   109  21.5 107.
 8 female Netball   3.8    6.6  36.5  12.4   102  24.4 157.
 9 female Netball   3.96   5.5  36.3  12.4    71  22.6 101.
10 female Netball   4.44   9.7  41.4  14.1    64  22.8 126.
# i 192 more rows
# i 1 more variable: Wt <dbl>
```

Choosing a column

```
athletes %>% select(Sport)
```

```
# A tibble: 202 x 1
   Sport
   <chr>
 1 Netball
 2 Netball
 3 Netball
 4 Netball
 5 Netball
 6 Netball
 7 Netball
 8 Netball
 9 Netball
10 Netball
# i 192 more rows
```

Choosing several columns

```
athletes %>% select(Sport, Hg, BMI)
```

```
# A tibble: 202 x 3
   Sport       Hg   BMI
   <chr>    <dbl> <dbl>
 1 Netball   13.6  19.2
 2 Netball   12.7  21.2
 3 Netball   12.3  21.4
 4 Netball   12.3  21.0
 5 Netball   12.8  21.8
 6 Netball   11.8  21.4
 7 Netball   12.7  21.5
 8 Netball   12.4  24.4
 9 Netball   12.4  22.6
10 Netball   14.1  22.8
# i 192 more rows
```

Choosing consecutive columns

```
athletes %>% select(Sex:WCC, BMI)
```

```
# A tibble: 202 x 5
   Sex    Sport      RCC   WCC   BMI
   <chr>  <chr>    <dbl> <dbl> <dbl>
 1 female Netball   4.56  13.3  19.2
 2 female Netball   4.15   6    21.2
 3 female Netball   4.16   7.6  21.4
 4 female Netball   4.32   6.4  21.0
 5 female Netball   4.06   5.8  21.8
 6 female Netball   4.12   6.1  21.4
 7 female Netball   4.17   5    21.5
 8 female Netball   3.8    6.6  24.4
 9 female Netball   3.96   5.5  22.6
10 female Netball   4.44   9.7  22.8
# i 192 more rows
```

```
1:5
```

Choosing all-but some columns

```
athletes %>% select(-(RCC:LBM))
```

```
# A tibble: 202 x 4
   Sex    Sport       Ht    Wt
   <chr>  <chr>    <dbl> <dbl>
 1 female Netball   177.  59.9
 2 female Netball   173.  63
 3 female Netball   176   66.3
 4 female Netball   170.  60.7
 5 female Netball   183   72.9
 6 female Netball   178.  67.9
 7 female Netball   177.  67.5
 8 female Netball   174.  74.1
 9 female Netball   174.  68.2
10 female Netball   174.  68.8
# i 192 more rows
```

```
athletes %>% select(Sex:Sport, Ht:Wt)
```

Select-helpers

Other ways to select columns: those whose name:

- `starts_with` something
- `ends_with` something
- `contains` something
- `matches` a “regular expression”
- `everything()` select all the columns

Columns whose names begin with S

```
athletes %>% select(starts_with("S"))
```

```
# A tibble: 202 x 3
   Sex    Sport      SSF
   <chr>  <chr>    <dbl>
 1 female Netball   49
 2 female Netball  110.
 3 female Netball   89
 4 female Netball   98.3
 5 female Netball  122.
 6 female Netball   90.4
 7 female Netball  107.
 8 female Netball  157.
 9 female Netball  101.
10 female Netball  126.
# i 192 more rows
```

Columns whose names end with C

either uppercase or lowercase:

```
athletes %>% select(ends_with("c"))
```

```
# A tibble: 202 x 3
     RCC   WCC    Hc
   <dbl> <dbl> <dbl>
 1  4.56  13.3  42.2
 2  4.15   6    38
 3  4.16   7.6  37.5
 4  4.32   6.4  37.7
 5  4.06   5.8  38.7
 6  4.12   6.1  36.6
 7  4.17   5    37.4
 8  3.8    6.6  36.5
 9  3.96   5.5  36.3
10  4.44   9.7  41.4
# i 192 more rows
```

Case-sensitive

This works with any of the select-helpers:

```
athletes %>% select(ends_with("C", ignore.case=FALSE))
```

```
# A tibble: 202 x 2
     RCC   WCC
   <dbl> <dbl>
 1  4.56  13.3
 2  4.15   6
 3  4.16   7.6
 4  4.32   6.4
 5  4.06   5.8
 6  4.12   6.1
 7  4.17   5
 8  3.8    6.6
 9  3.96   5.5
10  4.44   9.7
# i 192 more rows
```

Column names containing letter R

```
athletes %>% select(contains("r"))
```

```
# A tibble: 202 x 3
   Sport      RCC  Ferr
   <chr>    <dbl> <dbl>
 1 Netball  4.56    20
 2 Netball  4.15    59
 3 Netball  4.16    22
 4 Netball  4.32    30
 5 Netball  4.06    78
 6 Netball  4.12    21
 7 Netball  4.17   109
 8 Netball  3.8    102
 9 Netball  3.96    71
10 Netball  4.44    64
# i 192 more rows
```

Exactly two characters, ending with T

In regular expression terms, this is `^.t$`:

- `^` means "start of text"
- `.` means "exactly one character, but could be anything"
- `$` means "end of text".

```
athletes %>% select(matches("^.t$"))
```

```
# A tibble: 202 x 2
      Ht    Wt
   <dbl> <dbl>
 1  177.  59.9
 2  173.  63
 3  176   66.3
 4  170.  60.7
 5  183   72.9
 6  178.  67.9
 7  177.  67.5
 8  174.  74.1
```

Choosing columns by property

- Use `where` as with summarizing several columns
- eg, to choose text columns:

```
athletes %>% select(where(is.character))
```

```
# A tibble: 202 x 2
   Sex    Sport
   <chr>  <chr>
 1 female Netball
 2 female Netball
 3 female Netball
 4 female Netball
 5 female Netball
 6 female Netball
 7 female Netball
 8 female Netball
 9 female Netball
10 female Netball
# i 192 more rows
```

Choosing rows by number

```
athletes %>% slice(16:25)
```

```
# A tibble: 10 x 13
   Sex    Sport      RCC   WCC    Hc    Hg  Ferr   BMI   SSF
   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 female Netball   4.25  10.7  39.5  13.2   127  24.5 157.
 2 female Netball   4.46  10.9  39.7  13.7   102  24.0 116.
 3 female Netball   4.4    9.3  40.4  13.6    86  26.2 182.
 4 female Netball   4.83   8.4  41.8  13.4    40  20.0  71.6
 5 female Netball   4.23   6.9  38.3  12.6    50  25.7 144.
 6 female Netball   4.24   8.4  37.6  12.5    58  25.6 201.
 7 female Netball   3.95   6.6  38.4  12.8    33  19.9  68.9
 8 female Netball   4.03   8.5  37.7  13      51  23.4 104.
 9 female BBall     3.96   7.5  37.5  12.3    60  20.6 109.
10 female BBall     4.41   8.3  38.2  12.7    68  20.7 103.
# i 1 more variable: Wt <dbl>
```

Non-consecutive rows

```
athletes %>%
  slice(10, 13, 17, 42)
```

```
# A tibble: 4 x 13
  Sex    Sport      RCC   WCC    Hc    Hg  Ferr   BMI   SSF
  <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 female Netball  4.44   9.7  41.4  14.1    64  22.8  126.
2 female Netball  4.02   9.1  37.7  12.7   107  23.0   77 
3 female Netball  4.46  10.9  39.7  13.7   102  24.0  116.
4 female Row      4.37   8.1  41.8  14.3    53  23.5   98 
# i 1 more variable: Wt <dbl>
```

A random sample of rows

```
athletes %>% slice_sample(n=8)
```

```
# A tibble: 8 x 13
  Sex    Sport      RCC   WCC    Hc    Hg  Ferr   BMI   SSF
  <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 female Swim      4.31  5.3   41.1  13.9    60  20.9  62.9
2 male   T400m     4.55  5.55  42.6  14.4   106  21.2  34.1
3 male   TSprnt    5.69 10.8   50.5  18.5    53  24.5  42.3
4 female Tennis    4.66  6.4   40.9  13.9   109  18.4  38.2
5 male   WPolo     5.34  6.2   49.8  17.2   143  27.8  75.7
6 female Netball   4.32  6.4   37.7  12.3    30  21.0  98.3
7 male   WPolo     4.95  7.5   44.5  15      50  24.3 106.
8 female Row       4.46  9.5   41.5  14.5    92  23.0  83
# i 1 more variable: Wt <dbl>
```

Rows for which something is true

```
athletes %>% filter(Sport == "Tennis")
```

```
# A tibble: 11 x 13
   Sex    Sport    RCC   WCC    Hc    Hg  Ferr   BMI   SSF
   <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 female Tennis  4      4.2  36.6  12      57  25.4 109
 2 female Tennis  4.4    4    40.8  13.9    73  22.1  98.1
 3 female Tennis  4.38   7.9  39.8  13.5    88  21.2  80.6
 4 female Tennis  4.08   6.6  37.8  12.1   182  20.5  68.3
 5 female Tennis  4.98   6.4  44.8  14.8    80  17.1  47.6
 6 female Tennis  5.16   7.2  44.3  14.5    88  18.3  61.9
 7 female Tennis  4.66   6.4  40.9  13.9   109  18.4  38.2
 8 male   Tennis  5.66   8.3  50.2  17.7    38  23.8  56.5
 9 male   Tennis  5.03   6.4  42.7  14.3   122  22.0  47.6
10 male   Tennis  4.97   8.8  43    14.9   233  22.3  60.4
11 male   Tennis  5.38   6.3  46    15.7    32  21.1  34.9
# i 1 more variable: Wt <dbl>
```

More complicated selections

```
athletes %>% filter(Sport == "Tennis", RCC < 5)
```

```
# A tibble: 7 x 13
  Sex    Sport    RCC   WCC    Hc    Hg  Ferr   BMI   SSF
  <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 female Tennis  4      4.2  36.6  12      57  25.4 109
2 female Tennis  4.4    4    40.8  13.9    73  22.1  98.1
3 female Tennis  4.38   7.9  39.8  13.5    88  21.2  80.6
4 female Tennis  4.08   6.6  37.8  12.1   182  20.5  68.3
5 female Tennis  4.98   6.4  44.8  14.8    80  17.1  47.6
6 female Tennis  4.66   6.4  40.9  13.9   109  18.4  38.2
7 male   Tennis  4.97   8.8  43    14.9   233  22.3  60.4
# i 1 more variable: Wt <dbl>
```

Another way to do "and"

```
athletes %>% filter(Sport == "Tennis") %>%
  filter(RCC < 5)
```

```
# A tibble: 7 x 13
  Sex    Sport    RCC   WCC    Hc    Hg  Ferr   BMI   SSF
  <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 female Tennis  4      4.2  36.6  12      57  25.4 109
2 female Tennis  4.4    4    40.8  13.9    73  22.1  98.1
3 female Tennis  4.38   7.9  39.8  13.5    88  21.2  80.6
4 female Tennis  4.08   6.6  37.8  12.1   182  20.5  68.3
5 female Tennis  4.98   6.4  44.8  14.8    80  17.1  47.6
6 female Tennis  4.66   6.4  40.9  13.9   109  18.4  38.2
7 male   Tennis  4.97   8.8  43    14.9   233  22.3  60.4
# i 1 more variable: Wt <dbl>
```

Either/Or

```
athletes %>% filter(Sport == "Tennis" | RCC > 5)
```

```
# A tibble: 66 x 13
   Sex    Sport    RCC   WCC    Hc    Hg  Ferr   BMI   SSF
   <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 female Row     5.02   6.4  44.8  15.2    48  19.8  91
 2 female T400m   5.31   9.5  47.1  15.9    29  21.4  57.9
 3 female Field   5.33   9.3  47    15      62  25.3 103.
 4 female TSprnt  5.16   8.2  45.3  14.7    34  20.3  46.1
 5 female Tennis  4      4.2  36.6  12      57  25.4 109
 6 female Tennis  4.4    4    40.8  13.9    73  22.1  98.1
 7 female Tennis  4.38   7.9  39.8  13.5    88  21.2  80.6
 8 female Tennis  4.08   6.6  37.8  12.1   182  20.5  68.3
 9 female Tennis  4.98   6.4  44.8  14.8    80  17.1  47.6
10 female Tennis  5.16   7.2  44.3  14.5    88  18.3  61.9
# i 56 more rows
# i 1 more variable: Wt <dbl>
```

Sorting into order

```
athletes %>% arrange(RCC)
```

```
# A tibble: 202 x 13
   Sex    Sport      RCC   WCC    Hc    Hg  Ferr   BMI   SSF
   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 female Netball   3.8    6.6  36.5  12.4   102  24.4 157.
 2 female Netball   3.9    6.3  35.9  12.1    78  20.1  70
 3 female T400m     3.9    6    38.9  13.5    16  19.4  48.4
 4 female Row       3.91   7.3  37.6  12.9    43  22.3 126.
 5 female Netball   3.95   6.6  38.4  12.8    33  19.9  68.9
 6 female Row       3.95   3.3  36.9  12.5    40  24.5  74.9
 7 female Netball   3.96   5.5  36.3  12.4    71  22.6 101.
 8 female BBall     3.96   7.5  37.5  12.3    60  20.6 109.
 9 female Tennis    4      4.2  36.6  12      57  25.4 109
10 female Netball   4.02   9.1  37.7  12.7   107  23.0  77
# i 192 more rows
# i 1 more variable: Wt <dbl>
```

Breaking ties by another variable

```
athletes %>% arrange(RCC, BMI)
```

```
# A tibble: 202 x 13
   Sex    Sport      RCC   WCC    Hc    Hg  Ferr   BMI   SSF
   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 female Netball   3.8    6.6  36.5  12.4   102  24.4 157.
 2 female T400m     3.9    6    38.9  13.5    16  19.4  48.4
 3 female Netball   3.9    6.3  35.9  12.1    78  20.1  70
 4 female Row       3.91   7.3  37.6  12.9    43  22.3 126.
 5 female Netball   3.95   6.6  38.4  12.8    33  19.9  68.9
 6 female Row       3.95   3.3  36.9  12.5    40  24.5  74.9
 7 female BBall     3.96   7.5  37.5  12.3    60  20.6 109.
 8 female Netball   3.96   5.5  36.3  12.4    71  22.6 101.
 9 female Tennis    4      4.2  36.6  12      57  25.4 109
10 female Netball   4.02   9.1  37.7  12.7   107  23.0  77
# i 192 more rows
# i 1 more variable: Wt <dbl>
```

Descending order

```
athletes %>% arrange(desc(BMI))
```

```
# A tibble: 202 x 13
   Sex    Sport   RCC   WCC    Hc    Hg  Ferr   BMI   SSF
   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 male   Field  5.48   6.2  48.2  16.3    94  34.4  82.7
 2 male   Field  4.96   8.3  45.3  15.7   141  33.7 114.
 3 male   Field  5.48   4.6  49.4  18     132  32.5  55.7
 4 female Field  4.75   7.5  43.8  15.2    90  31.9 132.
 5 male   Field  5.01   8.9  46    15.9   212  30.2 112.
 6 male   Field  5.01   8.9  46    15.9   212  30.2  96.9
 7 male   Field  5.09   8.9  46.3  15.4    44  30.0  71.1
 8 female Field  4.58   5.8  42.1  14.7   164  28.6 110.
 9 female Field  4.51   9    39.7  14.3    36  28.1 136.
10 male   WPolo  5.34   6.2  49.8  17.2   143  27.8  75.7
# i 192 more rows
# i 1 more variable: Wt <dbl>
```

"The top ones"

```
athletes %>%
  arrange(desc(Wt)) %>%
  slice(1:7) %>%
  select(Sport, Wt)
```

```
# A tibble: 7 x 2
  Sport    Wt
  <chr> <dbl>
1 Field  123.
2 BBall  114.
3 Field  111.
4 Field  108.
5 Field  103.
6 WPolo  101
7 BBall  100.
```

Another way

```
athletes %>%
  slice_max(order_by = Wt, n=7) %>%
  select(Sport, Wt)
```

```
# A tibble: 7 x 2
  Sport    Wt
  <chr> <dbl>
1 Field  123.
2 BBall  114.
3 Field  111.
4 Field  108.
5 Field  103.
6 WPolo  101
7 BBall  100.
```

Create new variables from old ones

```
athletes %>%
  mutate(wt_lb = Wt * 2.2) %>%
  select(Sport, Sex, Wt, wt_lb) %>%
  arrange(Wt)
```

```
# A tibble: 202 x 4
   Sport   Sex       Wt wt_lb
   <chr>   <chr>  <dbl> <dbl>
 1 Gym     female  37.8  83.2
 2 Gym     female  43.8  96.4
 3 Gym     female  45.1  99.2
 4 Tennis  female  45.8 101.
 5 Tennis  female  47.4 104.
 6 Gym     female  47.8 105.
 7 T400m   female  49.2 108.
 8 Row     female  49.8 110.
 9 T400m   female  50.9 112.
10 Netball female  51.9 114.
```

Turning the result into a number

Output is always data frame unless you explicitly turn it into something else, eg. the weight of the heaviest athlete, as a number:

```
athletes %>% arrange(desc(Wt)) %>%
  pluck("Wt", 1) -> heavy
heavy
```

```
[1] 123.2
```

```
heavy * 2.2
```

```
[1] 271.04
```

Or the 20 heaviest weights in descending order:

```
athletes %>%
  arrange(desc(Wt)) %>%
  slice(1:20) %>%
  pluck("Wt")
```

Another way to do the last one

```
athletes %>%
  arrange(desc(Wt)) %>%
  slice(1:20) %>%
  pull("Wt")
```

```
 [1] 123.20 113.70 111.30 108.20 102.70 101.00 100.20  98.0
[11]  97.00  96.90  96.30  94.80  94.80  94.70  94.70  94.6
```

`pull` grabs the column you name *as a vector* (of whatever it contains).

To find the mean height of the women athletes

Two ways:

```
athletes %>% group_by(Sex) %>% summarize(m = mean(Ht))
```

```
# A tibble: 2 x 2
  Sex          m
  <chr>  <dbl>
1 female  175.
2 male    186.
```

```
athletes %>%
  filter(Sex == "female") %>%
  summarize(m = mean(Ht))
```

```
# A tibble: 1 x 1
      m
  <dbl>
1  175.
```

Summary of data selection/arrangement "verbs"

Verb	Purpose
`select`	Choose columns
`slice`	Choose rows by number
`slice_sample`	Choose random rows
`slice_max`	Choose rows with largest values on a variable (also `slice_min`)
`filter`	Choose rows satisfying conditions
`arrange`	Sort in order by column(s)
`mutate`	Create new variables
`group_by`	Create groups to work with
`summarize`	Calculate summary statistics (by groups if defined)
`pluck`	Extract items from data frame
`pull`	Extract a single column from a data frame as a vector

Looking things up in another data frame

- Suppose you are working in the nails department of a hardware store and you find that you have sold these items:

```
my_url <- "http://ritsokiguess.site/datafiles/nail_sales.cs
sales <- read_csv(my_url)
sales
```

```
# A tibble: 6 x 2
  product_code sales
  <chr>        <dbl>
1 061-5344-6      10
2 161-0090-0       6
3 061-5388-2       2
4 161-0199-4       8
5 061-5375-2       5
6 061-4525-2       3
```

Product descriptions and prices

- but you don't remember what these product codes are, and you would like to know the total revenue from these sales.
- Fortunately you found a list of product descriptions and prices:

```
my_url <- "http://ritsokiguess.site/datafiles/nail_desc.csv
desc <- read_csv(my_url)
desc
```

```
# A tibble: 7 x 5
  product_code description    size         qty price
  <chr>        <chr>          <chr>      <dbl> <dbl>
1 061-4525-2   spike nail     "10\""         1  1.49
2 061-5329-4   masonry nail   "1.5\""      112  8.19
3 061-5344-6   finishing nail "1\""        1298  6.99
4 061-5375-2   roofing nail   "1.25\""     192  6.99
5 061-5388-2   framing nail   "4\""         25  8.19
6 161-0090-0   wood nail      "1\""         25  2.39
7 161-0199-4   panel nail     "1-5/8\""     20  4.69
```

The lookup

- How do you "look up" the product codes to find the product descriptions and prices?
- `left_join`.

```
sales %>% left_join(desc)
```

```
# A tibble: 6 x 6
  product_code sales description    size         qty price
  <chr>        <dbl> <chr>          <chr>      <dbl> <dbl>
1 061-5344-6      10 finishing nail "1\""       1298  6.99
2 161-0090-0       6 wood nail      "1\""         25  2.39
3 061-5388-2       2 framing nail   "4\""         25  8.19
4 161-0199-4       8 panel nail     "1-5/8\""     20  4.69
5 061-5375-2       5 roofing nail   "1.25\""     192  6.99
6 061-4525-2       3 spike nail     "10\""         1  1.49
```

What we have

- this looks up all the rows in the *first* dataframe that are also in the *second*.
- by default matches all columns with same name in two dataframes (`product_code` here)
- get *all* columns in *both* dataframes. The rows are the ones for that `product_code`.

So now can work out how much the total revenue was:

```
sales %>% left_join(desc) %>%
  mutate(product_revenue = sales*price) %>%
  summarize(total_revenue = sum(product_revenue))
```

```
# A tibble: 1 x 1
  total_revenue
          <dbl>
1          178.
```

More comments

- if any product codes are not matched, you get NA in the added columns
- anything in the *second* dataframe that was not in the first does not appear (here, any products that were not sold)
- other variations (examples follow):
  - if there are two columns with the same name in the two dataframes, and you only want to match on one, use by with one column name
  - if the columns you want to look up have different names in the two dataframes, use by with a "named list"

Matching on only some matching names

- Suppose the `sales` dataframe *also* had a column `qty` (which was the quantity sold):

```
sales %>% rename("qty"="sales") -> sales1
sales1
```

```
# A tibble: 6 x 2
  product_code   qty
  <chr>        <dbl>
1 061-5344-6      10
2 161-0090-0       6
3 061-5388-2       2
4 161-0199-4       8
5 061-5375-2       5
6 061-4525-2       3
```

- The `qty` in `sales1` is the quantity sold, but the `qty` in `desc` is the number of nails in a package. These should *not* be matched: they are different things.

Matching only on product code

```
sales1 %>%
  left_join(desc, join_by(product_code))
```

```
# A tibble: 6 x 6
  product_code qty.x description    size       qty.y price
  <chr>         <dbl> <chr>          <chr>      <dbl> <dbl>
1 061-5344-6       10 finishing nail "1\""       1298  6.99
2 161-0090-0        6 wood nail      "1\""         25  2.39
3 061-5388-2        2 framing nail   "4\""         25  8.19
4 161-0199-4        8 panel nail     "1-5/8\""     20  4.69
5 061-5375-2        5 roofing nail   "1.25\""     192  6.99
6 061-4525-2        3 spike nail     "10\""         1  1.49
```

- Get `qty.x` (from `sales1`) and `qty.y` (from `desc`).

Matching on different names 1/2

- Suppose the product code in `sales` was just `code`:

```
sales %>% rename("code" = "product_code") -> sales2
sales2
```

```
# A tibble: 6 x 2
  code        sales
  <chr>       <dbl>
1 061-5344-6     10
2 161-0090-0      6
3 061-5388-2      2
4 161-0199-4      8
5 061-5375-2      5
6 061-4525-2      3
```

- How to match the two product codes that have different names?

Matching on different names 2/2

- Use `join_by`, but like this:

```
sales2 %>%
  left_join(desc, join_by(code == product_code))
```

```
# A tibble: 6 x 6
  code        sales description    size         qty price
  <chr>       <dbl> <chr>          <chr>      <dbl> <dbl>
1 061-5344-6     10 finishing nail "1\""       1298  6.99
2 161-0090-0      6 wood nail      "1\""         25  2.39
3 061-5388-2      2 framing nail   "4\""         25  8.19
4 161-0199-4      8 panel nail     "1-5/8\""     20  4.69
5 061-5375-2      5 roofing nail   "1.25\""     192  6.99
6 061-4525-2      3 spike nail     "10\""         1  1.49
```

Other types of join

- `right_join`: interchanges roles, looking up keys from second dataframe in first.
- `anti_join`: give me all the rows in the first dataframe that are *not* in the second. (Use this eg. to see whether the product descriptions are incomplete.)
- `full_join`: give me all the rows in both dataframes, with missings as needed.

Full join here

```
sales %>% full_join(desc)
```

```
# A tibble: 7 x 6
  product_code sales description    size         qty price
  <chr>        <dbl> <chr>          <chr>      <dbl> <dbl>
1 061-5344-6      10 finishing nail "1\""       1298  6.99
2 161-0090-0       6 wood nail      "1\""         25  2.39
3 061-5388-2       2 framing nail   "4\""         25  8.19
4 161-0199-4       8 panel nail     "1-5/8\""     20  4.69
5 061-5375-2       5 roofing nail   "1.25\""     192  6.99
6 061-4525-2       3 spike nail     "10\""         1  1.49
7 061-5329-4      NA masonry nail   "1.5\""      112  8.19
```

- The missing `sales` for "masonry nail" says that it was in the lookup table `desc`, but we didn't sell any.

The same thing, but with anti_join

Anything in first df but not in second?

```
desc %>% anti_join(sales)
```

```
# A tibble: 1 x 5
  product_code description  size       qty price
  <chr>        <chr>        <chr>    <dbl> <dbl>
1 061-5329-4   masonry nail "1.5\""    112  8.19
```

```
# sales %>% anti_join(desc)
```

Masonry nails are the only thing in our product description file that we did not sell any of.