Doing things with data frames

Doing things with data frames

Let's go back to our Australian athletes:

```
##
## -- Column specification ----------------------------
## cols(
##   Sex = col_character(),
##   Sport = col_character(),
##   RCC = col_double(),
##   WCC = col_double(),
##   Hc = col_double(),
##   Hg = col_double(),
##   Ferr = col_double(),
##   BMI = col_double(),
##   SSF = col_double(),
##   `%Bfat` = col_double(),
##   LBM = col_double(),
##   Ht = col_double()
```

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
## # ... with 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
## # ... with 192 more rows
```

Choosing consecutive columns

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

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

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
## # ... with 192 more rows
```

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.
## # ... with 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
```

Case-sensitive

This works with any of the select-helpers:

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

```
## # 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
```

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
## # ... with 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
```

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
```

Choosing rows by number

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

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

Non-consecutive rows

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

```
## # A tibble: 4 x 13
##   Sex    Sport     RCC   WCC    Hc    Hg  Ferr   BMI
##   <chr>  <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 female Netball  4.44   9.7  41.4  14.1    64  22.8
## 2 female Netball  4.02   9.1  37.7  12.7   107  23.0
## 3 female Netball  4.46  10.9  39.7  13.7   102  24.0
## 4 female Row      4.37   8.1  41.8  14.3    53  23.5
## # ... with 5 more variables: SSF <dbl>,
## #   %Bfat <dbl>, LBM <dbl>, Ht <dbl>, Wt <dbl>
```

A random sample of rows

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

```
## # A tibble: 8 x 13
##   Sex    Sport     RCC   WCC    Hc    Hg  Ferr   BMI
##   <chr>  <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 male   TSprnt   5.59   7.9  49.7  17.2   220  23.6
## 2 male   Row      5.04   7.1  44    14.8    64  25.8
## 3 male   Row      5.09  10.1  44.9  14.8   118  26.8
## 4 female T400m    4.27   6.9  44.1  14.7    45  20.4
## 5 female Netball  4.24   8.4  37.6  12.5    58  25.6
## 6 male   Swim     5.33   5.2  47.8  16.1   176  21.4
## 7 female Netball  4.46  10.9  39.7  13.7   102  24.0
## 8 male   BBall    4.73   6.7  42.8  14.9     8  19.8
## # ... with 5 more variables: SSF <dbl>,
## #   %Bfat <dbl>, LBM <dbl>, Ht <dbl>, Wt <dbl>
```

Rows for which something is true

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

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

More complicated selections

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

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

Either/Or

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

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

Sorting into order

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

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

Breaking ties by another variable

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

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

Descending order

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

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

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)
```

```
## [1] 123.2
```

Or the 20 heaviest weights in descending order:

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

```
##  [1] 123.20 113.70 111.30 108.20 102.70 101.00
##  [7] 100.20  98.00  97.90  97.90  97.00  96.90
## [13]  96.30  94.80  94.80  94.70  94.70  94.60
## [19]  94.25  94.20
```

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
##  [7] 100.20  98.00  97.90  97.90  97.00  96.90
## [13]  96.30  94.80  94.80  94.70  94.70  94.60
## [19]  94.25  94.20
```

`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
`print`	Display non-default # of rows/columns
`slice`	Choose rows by number
`sample_n`	Choose random rows
`filter`	Choose rows satisfying conditions
`arrange`	Sort in order by column(s)
`mutate`	Create new variables
`group_by`	Create groups to summarize by
`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

Recall the tuberculosis data set, tidied:

```
tb3
```

```
## # A tibble: 35,750 x 5
##    iso2   year gender age    freq
##    <chr> <dbl> <chr>  <chr> <dbl>
##  1 AD     1996 m      014       0
##  2 AD     1996 m      1524      0
##  3 AD     1996 m      2534      0
##  4 AD     1996 m      3544      4
##  5 AD     1996 m      4554      1
##  6 AD     1996 m      5564      0
##  7 AD     1996 m      65        0
##  8 AD     1996 f      014       0
##  9 AD     1996 f      1524      1
## 10 AD     1996 f      2534      1
```

Actual country names

Found actual country names to go with those abbreviations, in spreadsheet:

```
my_url <-
  "http://www.utsc.utoronto.ca/~butler/c32/ISOCountryCodes081507.xlsx"
```

Note trick for reading in `.xlsx` from URL:

```
f <- tempfile()
download.file(my_url, f)
country_names <- read_excel(f)
```

- set up temporary file
- download spreadsheet to there
- read it from temporary file (which is “local”)

The country names

```
country_names
```

```
## # A tibble: 252 x 3
##    Code  Code_UC Country
##    <chr> <chr>   <chr>
##  1 ad    AD      Andorra
##  2 ae    AE      United Arab Emirates
##  3 af    AF      Afghanistan
##  4 ag    AG      Antigua and Barbuda
##  5 ai    AI      Anguilla
##  6 al    AL      Albania
##  7 am    AM      Armenia
##  8 an    AN      Netherlands Antilles
##  9 ao    AO      Angola
## 10 aq    AQ      Antarctica
## # ... with 242 more rows
```

Looking up country codes

Matching a variable in one data frame to one in another is called a **join** (database terminology):

```
tb3 %>% left_join(country_names, by = c("iso2" = "Code_UC"))
```

```
## # A tibble: 35,750 x 7
##    iso2   year gender age    freq Code  Country
##    <chr> <dbl> <chr>  <chr> <dbl> <chr> <chr>
##  1 AD     1996 m      014       0 ad    Andorra
##  2 AD     1996 m      1524      0 ad    Andorra
##  3 AD     1996 m      2534      0 ad    Andorra
##  4 AD     1996 m      3544      4 ad    Andorra
##  5 AD     1996 m      4554      1 ad    Andorra
##  6 AD     1996 m      5564      0 ad    Andorra
##  7 AD     1996 m      65        0 ad    Andorra
##  8 AD     1996 f      014       0 ad    Andorra
##  9 AD     1996 f      1524      1 ad    Andorra
```

Total cases by country

```
options(dplyr.summarise.inform=FALSE)
```

```
tb3 %>%
  group_by(iso2) %>%
  summarize(cases = sum(freq)) %>%
  left_join(country_names, by = c("iso2" = "Code_UC")) %>%
  select(Country, cases)
```

```
## # A tibble: 213 x 2
##    Country                cases
##    <chr>                  <dbl>
##  1 Andorra                   64
##  2 United Arab Emirates     487
##  3 Afghanistan            80005
##  4 Antigua and Barbuda       21
##  5 Anguilla                   1
##  6 Albania                 2467
```

or even sorted in order

```
tb3 %>%
  group_by(iso2) %>%
  summarize(cases = sum(freq)) %>%
  left_join(country_names, by = c("iso2" = "Code_UC")) %>%
  select(Country, cases) %>%
  arrange(desc(cases))
```

```
## # A tibble: 213 x 2
##    Country        cases
##    <chr>          <dbl>
##  1 China        4065174
##  2 India        3966169
##  3 Indonesia    1129015
##  4 South Africa  900349
##  5 Bangladesh    758008
##  6 Vietnam       709695
##  7 <NA>          603095
```

Comments

- This is probably not quite right because of:
  - the 1994-1995 thing
  - there is at least one country in `tb3` that was not in `country_names` (the NA above). Which?

```
tb3 %>%
  anti_join(country_names, by = c("iso2" = "Code_UC")) %>%
  distinct(iso2)
```

```
## # A tibble: 6 x 1
##   iso2
##   <chr>
## 1 CD
## 2 ME
## 3 <NA>
## 4 PS
## 5 RS
## 6 TL
```

Doing things one row at a time

A data frame `d`:

```
## # A tibble: 3 x 2
##      x1    x2
##   <dbl> <dbl>
## 1    10    13
## 2    11     8
## 3     3     4
```

Want largest value in each row.

Try number 1

```r
d %>% mutate(mx = max(x1, x2))
```

```
## # A tibble: 3 x 3
##       x1    x2    mx
##    <dbl> <dbl> <dbl>
## 1    10    13    13
## 2    11     8    13
## 3     3     4    13
```

- Fails because `max` finds the largest of *all* values in `x1` and `x2` (and repeats answer 3 times), rather than max of the two values in each row.

Try number 2

```
d %>% rowwise() %>%
  mutate(mx = max(x1, x2))
```

```
## # A tibble: 3 x 3
##       x1    x2    mx
##    <dbl> <dbl> <dbl>
## 1     10    13    13
## 2     11     8    11
## 3      3     4     4
```

- Works because rowwise works one row at a time: “find max of all numbers in 1st row”, then “all in 2nd” etc.

Calculations for groups

- Back to Australian athletes data: suppose we want the correlation between height and weight for athletes of each sport separately:

```
athletes %>% group_by(Sport) %>%
  summarize(correl = cor(Ht, Wt))
```

```
## # A tibble: 10 x 2
##    Sport   correl
##    <chr>    <dbl>
##  1 BBall    0.880
##  2 Field    0.471
##  3 Gym      0.653
##  4 Netball  0.363
##  5 Row      0.873
##  6 Swim     0.860
##  7 T400m    0.828
##  8 Tennis   0.826
##  9 TSprnt   0.806
## 10 WPolo    0.595
```

Another way

- Break the data set into groups first:

```
athletes %>% nest_by(Sport)
```

```
## # A tibble: 10 x 2
##    Sport               data
##    <chr>   <list<tibble>>
##  1 BBall        [25 x 12]
##  2 Field        [19 x 12]
##  3 Gym           [4 x 12]
##  4 Netball      [23 x 12]
##  5 Row          [37 x 12]
##  6 Swim         [22 x 12]
##  7 T400m        [29 x 12]
##  8 Tennis       [11 x 12]
##  9 TSprnt       [15 x 12]
## 10 WPolo        [17 x 12]
```

Comments

- It looks as if all the other variables have disappeared, but they are all hiding in the column `data`.
- each thing in `data` is a *dataframe* (inside a dataframe!)
- when a column consists of things that are *not* single numbers, pieces of text, etc., it's called a **list-column** (see `list` in column header).
- to use this, we work `rowwise` on each sport and the `data` that belongs to that sport.

The rowwise way

```
athletes %>% nest_by(Sport) %>%
  mutate(correl = with(data, cor(Ht, Wt)))
```

```
## # A tibble: 10 x 3
##    Sport             data correl
##    <chr>   <list<tibble>>  <dbl>
##  1 BBall         [25 x 12]  0.880
##  2 Field         [19 x 12]  0.471
##  3 Gym            [4 x 12]  0.653
##  4 Netball       [23 x 12]  0.363
##  5 Row           [37 x 12]  0.873
##  6 Swim          [22 x 12]  0.860
##  7 T400m         [29 x 12]  0.828
##  8 Tennis        [11 x 12]  0.826
##  9 TSprnt        [15 x 12]  0.806
## 10 WPolo         [17 x 12]  0.595
```

Another use of list-columns

- Let's find the five-number summary of heights for male and female athletes:

```
athletes %>%
  group_by(Sex) %>%
  summarize(q = quantile(Ht))
```

```
## # A tibble: 10 x 2
##    Sex        q
##    <chr>  <dbl>
##  1 female  149.
##  2 female  171.
##  3 female  175
##  4 female  180.
##  5 female  196.
##  6 male    165.
##  7 male    180.
##  8 male    186.
##  9 male    191
## 10 male    209.
```

Better

```
athletes %>% group_by(Sex) %>%
  summarize(q = list(quantile(Ht)))
```

```
## # A tibble: 2 x 2
##   Sex    q
##   <chr>  <list>
## 1 female <dbl [5]>
## 2 male   <dbl [5]>
```

- each five-number summary is in a list-column, labelled by which Sex it belongs to.

To get this out

- to look at it:

```
athletes %>% group_by(Sex) %>%
  summarize(q = list(quantile(Ht))) %>%
  unnest(q)
```

```
## # A tibble: 10 x 2
##    Sex        q
##    <chr>  <dbl>
##  1 female  149.
##  2 female  171.
##  3 female  175
##  4 female  180.
##  5 female  196.
##  6 male    165.
##  7 male    180.
##  8 male    186.
##  9 male    191
## 10 male    209.
```

Even better

- quantile produces a "named vector" with the quantiles labelled:

```
quantile(athletes$Ht)
```

```
##       0%     25%     50%     75%    100%
## 148.900 174.000 179.700 186.175 209.400
```

- which we turn into dataframe thus:

```
enframe(quantile(athletes$Ht))
```

```
## # A tibble: 5 x 2
##   name  value
##   <chr> <dbl>
## 1 0%     149.
## 2 25%    174
## 3 50%    180.
## 4 75%    186.
## 5 100%   209.
```

for males and female athletes

```
athletes %>% group_by(Sex) %>%
  summarize(q = list(enframe(quantile(Ht)))) %>%
  unnest(q)
```

```
## # A tibble: 10 x 3
##     Sex    name  value
##     <chr>  <chr> <dbl>
##   1 female 0%      149.
##   2 female 25%     171.
##   3 female 50%     175
##   4 female 75%     180.
##   5 female 100%    196.
##   6 male   0%      165.
##   7 male   25%     180.
##   8 male   50%     186.
##   9 male   75%     191
##  10 male   100%    209.
```

Showing off

- we see `pivot_wider` later:

```
athletes %>% group_by(Sex) %>%
  summarize(q = list(enframe(quantile(Ht)))) %>%
  unnest(q) %>%
  pivot_wider(names_from = name, values_from = value)
```

```
## # A tibble: 2 x 6
##   Sex     `0%` `25%` `50%` `75%` `100%`
##   <chr>  <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 female  149.  171.  175   180.   196.
## 2 male    165.  180.  186.  191    209.
```

Simulation

- if I take a sample of 16 observations from a normal distribution with mean 100 and SD 15, what will its mean look like?
- one sample:

```
rnorm(16, 100, 15)
```

```
##  [1]  71.42762 110.90188 121.94001  90.40792
##  [5]  85.75109  72.39900 119.84313 108.99881
##  [9]  99.08615 114.11937  81.80634  99.93786
## [13]  90.21122 117.36629 108.88385  95.17335
```

- do it lots of times (say 1000).
- set up dataframe with 1000 samples numbered, and do it rowwise.

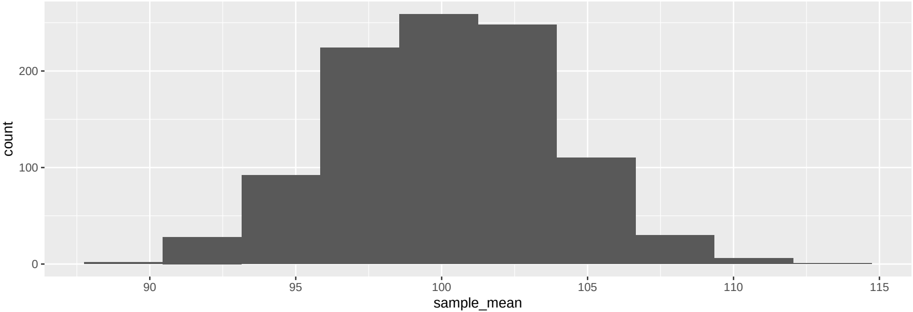
The simulation

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rnorm(16, 100, 15))) %>%
  mutate(sample_mean = mean(sample)) -> d
d
```

```
## # A tibble: 1,000 x 3
##      sim sample     sample_mean
##    <int> <list>           <dbl>
##  1     1 <dbl [16]>        95.3
##  2     2 <dbl [16]>       105.
##  3     3 <dbl [16]>       101.
##  4     4 <dbl [16]>       104.
##  5     5 <dbl [16]>       104.
##  6     6 <dbl [16]>        95.4
##  7     7 <dbl [16]>       102.
##  8     8 <dbl [16]>        97.7
```

A histogram of the sample means

```
ggplot(d, aes(x = sample_mean)) + geom_histogram(bins = 10)
```



- sample means are closer to 100 than individual observations are
- distribution of sample means is also normal