Course Content

• Using the tidyverse packages
  – Data Import
  – Filtering, selecting and sorting
  – Restructuring data
  – Grouping and Summarising
  – Extending and Merging

• Custom functions
Tidyverse
https://www.tidyverse.org/

- Collection of R packages
  - Aims to fix many of core R's structural problems
  - Common design and data philosophy
  - Designed to work together, but integrate seamlessly with other parts of R
Tidyverse Packages

• Tibble - data storage
• ReadR - reading data from files
• TidyR - Model data correctly
• DplyR - Manipulate and filter data
• Ggplot2 - Draw figures and graphs
Tidyverse Philosophies

• All data stored in a tibble (like a data frame)

• Make all functions perform a single, simple operation
  – Function names should be descriptive - normally a verb

  – All functions take a tibble as their first argument, allowing 'pipes' to be constructed

  – All (well most) functions return a tibble

• Use functional programming (not OO)
  – Don't modify existing data - create a modified copy
Using Tidyverse
Installation and calling

- `install.packages("tidyverse")`
- `library(tidyverse)`

```r
-- Attaching packages ---------- tidyverse 1.2.1 --
v ggplot2 3.1.0   v purrr 0.2.5
v tibble 2.0.1   v dplyr 0.7.8
v tidyr 0.8.2    v stringr 1.3.1
v readr 1.3.1    v forcats 0.3.0
```
Reading Files with readr

• Provides functions which mirror R's `read.table`
  – `read_csv("file.csv")`
  – `read_tsv("file.tsv")`
  – `read_delim("file.txt")`
  – `read_fwf("file.txt", col_positions=c(1,3,6))`
Reading files with readr

- Output is always a tibble
- No name translation
- Guesses appropriate formats
  - Says what formats were guessed (can get it wrong though!)
  - No string to factor conversion
Reading files with readr

```r
> read_tsv("trumpton.txt") -> trumpton
  Parsed with column specification:
  cols(
    LastName = col_character(),
    FirstName = col_character(),
    Age = col_double(),
    Weight = col_double(),
    Height = col_double()
  )
> trumpton
  # A tibble: 7 x 5
  #  LastName FirstName Age Weight Height
  #   <chr>    <chr>  <dbl>  <dbl>  <dbl>
  1 Hugh     Chris   26     90    175
  2 Pew      Adam    32    102    183
  3 Barney   Daniel  18     88    168
  4 McGrew   Chris   48     97    155
  5 Cuthbert Carl    28     91    188
  6 Dibble   Liam    35     94    145
  7 Grub     Doug    31     89    164
```

Fixing guessed columns

> read_tsv("import_problems.txt")

Parsed with column specification:
cols(
  Chr = col_double(),
  Gene = col_character(),
  Expression = col_double(),
  Significance = col_character()
)

Warning: 133 parsing failures.

<table>
<thead>
<tr>
<th>row</th>
<th>col</th>
<th>expected</th>
<th>actual</th>
<th>file</th>
</tr>
</thead>
<tbody>
<tr>
<td>1041</td>
<td>Chr</td>
<td>a double</td>
<td>X</td>
<td>'import_problems.txt'</td>
</tr>
<tr>
<td>1042</td>
<td>Chr</td>
<td>a double</td>
<td>X</td>
<td>'import_problems.txt'</td>
</tr>
<tr>
<td>1043</td>
<td>Chr</td>
<td>a double</td>
<td>X</td>
<td>'import_problems.txt'</td>
</tr>
<tr>
<td>1044</td>
<td>Chr</td>
<td>a double</td>
<td>X</td>
<td>'import_problems.txt'</td>
</tr>
<tr>
<td>1045</td>
<td>Chr</td>
<td>a double</td>
<td>X</td>
<td>'import_problems.txt'</td>
</tr>
</tbody>
</table>

See problems(...) for more details.

- Types are guessed on first 1000 lines
- Warnings for later mismatches
- Invalid values converted to NA
Fixing guessed columns

# A tibble: 1,174 x 4

  Chr Gene     Expression  Significance
  <dbl> <chr>         <dbl>       <chr>
1     1  Depdc2       9.19   NS
2     1   Sulf1       9.66   NS
3     1    Rpl7       8.75 0.050626416
4     1    Phf3       8.43   NS
5     1  Khdrbs2      8.94   NS
6     1    Prim2      9.64   NS
7     1  Hs6st1       9.60 0.03441748
8     1 BC050210     8.74   NS
9     1  Tmem131      8.99   NS
10    1     Aff3      10.8    NS
Fixing guessed columns

```r
read_tsv(
  "import_problems.txt",
  guess_max=1000000
)
```

Parsed with column specification:
```r
cols(
  Chr = col_character(),
  Gene = col_character(),
  Expression = col_double(),
  Significance = col_character()
)
```

# A tibble: 1,174 x 4

<table>
<thead>
<tr>
<th>Chr</th>
<th>Gene</th>
<th>Expression</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;chr&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;chr&gt;</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9.19</td>
<td>NS</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>9.66</td>
<td>NS</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8.75</td>
<td>0.050626416</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>8.43</td>
<td>NS</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>8.94</td>
<td>NS</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>9.64</td>
<td>NS</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>9.60</td>
<td>0.03441748</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>8.74</td>
<td>NS</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>8.99</td>
<td>NS</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>10.8</td>
<td>NS</td>
</tr>
</tbody>
</table>
# ... with 1,164 more rows
Fixing guessed columns

```r
read_tsv(
  "import_problems.txt",
  col_types=cols(Chr=col_character(), Significance=col_double())
)
```

Warning: 982 parsing failures.

```r
# A tibble: 1,174 x 4
  Chr  Gene   Expression Significance
  <chr> <chr>      <dbl>           <dbl>
1     1  Depdc2   9.19            NA
2     1  Sulf1    9.66            NA
3     1  Rpl7     8.75          0.0506
4     1  Phf3     8.43            NA
5     1  Khdrbs2  8.94            NA
6     1  Prim2    9.64            NA
7     1  Hs6st1   9.60          0.0344
8     1  BC050210 8.74           NA
9     1  Tmem131  8.99           NA
10    1  Aff3     10.8           NA
# ... with 1,164 more rows
```

See problems(...) for more details.
Exercise 1
Reading Data into Tibbles
Filtering, Selecting, Sorting etc.
Subsetting and Filtering

- **slice** pick rows by position
- **select** pick columns by name/position
- **arrange** sort rows
- **distinct** deduplicate rows
- **filter** pick rows based on the data
Trumpton

# A tibble: 7 x 5

<table>
<thead>
<tr>
<th>LastName</th>
<th>FirstName</th>
<th>Age</th>
<th>Weight</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hugh</td>
<td>Chris</td>
<td>26</td>
<td>90</td>
<td>175</td>
</tr>
<tr>
<td>Pew</td>
<td>Adam</td>
<td>32</td>
<td>102</td>
<td>183</td>
</tr>
<tr>
<td>Barney</td>
<td>Daniel</td>
<td>18</td>
<td>88</td>
<td>168</td>
</tr>
<tr>
<td>McGrew</td>
<td>Chris</td>
<td>48</td>
<td>97</td>
<td>155</td>
</tr>
<tr>
<td>Cuthbert</td>
<td>Carl</td>
<td>28</td>
<td>91</td>
<td>188</td>
</tr>
<tr>
<td>Dibble</td>
<td>Liam</td>
<td>35</td>
<td>94</td>
<td>145</td>
</tr>
<tr>
<td>Grub</td>
<td>Doug</td>
<td>31</td>
<td>89</td>
<td>164</td>
</tr>
</tbody>
</table>
Using \texttt{slice} or \texttt{select}

\texttt{slice(data, rows)}
\texttt{select(data, cols)}

\texttt{slice(trumpton, 1, 4, 7)}

\texttt{select(LastName, Age, Height)}

\begin{verbatim}
# A tibble: 3 x 5
    LastName FirstName  Age Weight Height
   <chr>     <chr>   <dbl>  <dbl>  <dbl>
1  Hugh     Chris     26     90    175
2 McGrew    Chris     48     97    155
3    Grub    Doug     31     89    164

# A tibble: 7 x 3
    LastName Age Height
   <chr>  <dbl>  <dbl>
1    Hugh   26    175
2    Pew    32    183
3  Barney   18    168
4 McGrew   48    155
5 Cuthbert 28    188
6    Dibble 35    145
7    Grub   31    164
\end{verbatim}
Using slice and select

slice(select(trumpton, LastName, Age, Height), 1, 4, 7)

# A tibble: 3 x 3
     LastName Age Height
  <chr>   <dbl>  <dbl>
1  Hugh     26    175
2 McGrew    48    155
3  Grub     31    164
The `%>%` operator (pipe)

- All* tidyverse functions take a tibble as the first argument and return a modified tibble

- Often need to 'chain' several functions together.

- The `%>%` operator takes the argument on its left and makes it the first argument to the function on its right
  - Correct logical flow to the code
  - Easier to read and interpret
  - Cleaner (fewer intermediate variables, less typing)

*Well nearly all, anyway.
The %>% operator (pipe)

data %>% head()

head(data)

tibble(a=1:10, b=21:30) %>% nrow()
Using slice and select

trumpton %>% select(LastName, Age) %>% slice(1,4,7)

# A tibble: 3 x 3
  LastName Age Height
<chr>    <dbl> <dbl>
1 Hugh    26    175
2 McGrew  48    155
3 Grub    31    164
Using `slice` and `select`

```r
trumpton %>%
  select(LastName, Age) %>%
  slice(1,4,7)
```

# A tibble: 3 x 3

<table>
<thead>
<tr>
<th>LastName</th>
<th>Age</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hugh</td>
<td>26</td>
<td>175</td>
</tr>
<tr>
<td>McGrew</td>
<td>48</td>
<td>155</td>
</tr>
<tr>
<td>Grub</td>
<td>31</td>
<td>164</td>
</tr>
</tbody>
</table>
Defining Selected Columns

- Common rules used throughout tidyverse.

- Single definitions (name, position or function)
  
  Positive  
  weight, height, length, 1, 2, 3, last_col()

  Negative  
  -chromosome, -start, -end, -1, -2, -3

- Range selections
  
  3:5  -(3:5)  height:length  -(height:length)

- Functional selections (positive or negative)

  starts_with()  -starts_with()
  ends_with()  -ends_with()
  contains()  -contains()
  matches()  -matches
Using `select` helpers

```r
colnames(child.variants)  # CHR POS dbSNP REF ALT QUAL GENE ENST MutantReads COVERAGE MutantReadPercent
child.variants %>%
  select(REF, COVERAGE)     # REF COVERAGE
select(-CHR, -ENST)        # POS dbSNP REF ALT QUAL GENE MutantReads COVERAGE MutantReadPercent
select(5:last_col())       # ALT QUAL GENE ENST MutantReads COVERAGE MutantReadPercent
select(POS:GENE)           # POS dbSNP REF ALT QUAL GENE
select(-(POS:GENE))        # CHR ENST MutantReads COVERAGE MutantReadPercent
select(starts_with("Mut")) # MutantReads MutantReadPercent
select(-ends_with("t", ignore.case = TRUE)) # CHR POS dbSNP REF QUAL GENE MutantReads COVERAGE
select(contains("Read"))   # MutantReads MutantReadPercent
```
arrange (sorting)  
distinct (deduplication)  

trumpton %>%  
distinct(FirstName, .keep_all = TRUE) %>%  
arrange(Height)  

# A tibble: 6 x 5  
  LastName FirstName Age Weight Height  
  <chr>    <chr>   <dbl>  <dbl>   <dbl>  
1 Dibble   Liam     35     94    145  
2 Grub     Doug     31     89    164  
3 Barney   Daniel   18     88    168  
4 Hugh     Chris    26     90    175  
5 Pew      Adam     32    102    183  
6 Cuthbert Carl    28     91    188  

You need .keep_all=TRUE if you want to see more than the district column. “keep_all” has a dot before it
arrange (sorting)  
distinct (deduplication)

trumpton %>%  
distinct(FirstName, .keep_all = TRUE) %>%  
arrange(desc(Height))

# A tibble: 6 x 5  
  LastName FirstName Age  Weight Height  
  <chr>     <chr>   <dbl>   <dbl>   <dbl>  
1 Cuthbert  Carl   28      91     188  
2 Pew       Adam   32     102     183  
3 Hugh      Chris  26      90     175  
4 Barney    Daniel 18      88     168  
5 Grub      Doug   31      89     164  
6 Dibble    Liam   35      94     145
# Functional row selection using `filter`

```r
dplyr::trumpton %>%
dplyr::filter(Height>170)
```

# A tibble: 3 x 5

<table>
<thead>
<tr>
<th>LastName</th>
<th>FirstName</th>
<th>Age</th>
<th>Weight</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hugh</td>
<td>Chris</td>
<td>26</td>
<td>90</td>
<td>175</td>
</tr>
<tr>
<td>Pew</td>
<td>Adam</td>
<td>32</td>
<td>102</td>
<td>183</td>
</tr>
<tr>
<td>Cuthbert</td>
<td>Carl</td>
<td>28</td>
<td>91</td>
<td>188</td>
</tr>
</tbody>
</table>
Exercise 2
Filtering and selecting
More clever filtering
Multi-condition filter

trumpton %>%
  filter(Height > 170) %>%
  filter(Age > 30)

# A tibble: 1 x 5
  LastName FirstName Age Weight Height
   <chr>    <chr> <dbl> <dbl> <dbl>
1  Pew      Adam   32    102  183
Multi-condition filter

```r
trumpton %>%
  filter(Height > 170, Age > 30)
```

# A tibble: 1 x 5

<table>
<thead>
<tr>
<th>LastName</th>
<th>FirstName</th>
<th>Age</th>
<th>Weight</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pew</td>
<td>Adam</td>
<td>32</td>
<td>102</td>
<td>183</td>
</tr>
</tbody>
</table>

, = logical AND
& = logical AND
| = logical OR
! = logical NOT
Multi-condition filter

trumpton %>%
  filter(Height > 170 | Age > 30)

# A tibble: 6 x 5
  LastName FirstName Age Weight Height
  <chr>    <chr>   <dbl>  <dbl>  <dbl>
1 Hugh     Chris   26     90    175
2 Pew      Adam    32    102    183
3 McGrew   Chris   48     97    155
4 Cuthbert Carl   28     91    188
5 Dibble   Liam    35     94    145
6 Grub     Doug    31     89    164

,  = logical AND
&  = logical AND
|  = logical OR
!  = logical NOT
Using `filter` with `%in%`

```r
> hits
[1] "FGFR1"  "RASAL1" "GLB1L2" "DNAH1"  "PTH1R"

child.variants %>%
distinct(GENE, .keep_all = TRUE) %>%
filter(GENE %in% hits)
```

# A tibble: 5 x 11

<table>
<thead>
<tr>
<th>CHR</th>
<th>POS</th>
<th>dbsNP</th>
<th>REF</th>
<th>ALT</th>
<th>Qual</th>
<th>GENE</th>
<th>ENST</th>
<th>MutantReads</th>
<th>COVERAGE</th>
<th>MutantReadPerce~</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;chr&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;dbl&gt;</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>134226278</td>
<td>rs3802928</td>
<td>C</td>
<td>T</td>
<td>200</td>
<td>GLB1L2</td>
<td>13</td>
<td>43</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>113539822</td>
<td>rs1674101</td>
<td>A</td>
<td>G</td>
<td>200</td>
<td>RASAL1</td>
<td>19</td>
<td>22</td>
<td>86</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>46944274</td>
<td>rs1138518</td>
<td>T</td>
<td>C</td>
<td>200</td>
<td>PTH1R</td>
<td>32</td>
<td>75</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>52430526</td>
<td>rs12163565</td>
<td>G</td>
<td>A</td>
<td>200</td>
<td>DNAH1</td>
<td>38</td>
<td>50</td>
<td>76</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>38271182</td>
<td>.</td>
<td>TG</td>
<td>T</td>
<td>200</td>
<td>FGFR1</td>
<td>9</td>
<td>31</td>
<td>29</td>
</tr>
</tbody>
</table>
Using `filter` with `str_detect`

```
child.variants %>%
distinct(GENE, .keep_all = TRUE) %>%
filter(str_detect(GENE,"ZFP"))
```

<table>
<thead>
<tr>
<th>CHR</th>
<th>POS</th>
<th>dbSNP</th>
<th>REF</th>
<th>ALT</th>
<th>QUAL</th>
<th>GENE</th>
<th>ENST</th>
<th>MutantReads</th>
<th>COVERAGE</th>
<th>MutantReadPerce~</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>68598007</td>
<td>rs1177648</td>
<td>A</td>
<td>G</td>
<td>200</td>
<td>ZFP90</td>
<td>ENST00000398253</td>
<td>43</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>88552370</td>
<td>rs3751673</td>
<td>A</td>
<td>G</td>
<td>53</td>
<td>ZFPM1</td>
<td>ENST00000319555</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>5292030</td>
<td>rs620652</td>
<td>A</td>
<td>G</td>
<td>200</td>
<td>ZFP161</td>
<td>ENST00000357006</td>
<td>28</td>
<td>71</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>57065189</td>
<td>rs145011</td>
<td>T</td>
<td>C</td>
<td>200</td>
<td>ZFP28</td>
<td>ENST00000301318</td>
<td>59</td>
<td>137</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>50768672</td>
<td>.</td>
<td>GT</td>
<td>G</td>
<td>200</td>
<td>ZFP64</td>
<td>ENST00000216923</td>
<td>36</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>180276402</td>
<td>rs168726</td>
<td>C</td>
<td>T</td>
<td>200</td>
<td>ZFP62</td>
<td>ENST00000502412</td>
<td>74</td>
<td>83</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>106814656</td>
<td>rs2920048</td>
<td>G</td>
<td>C</td>
<td>200</td>
<td>ZFPM2</td>
<td>ENST00000407775</td>
<td>33</td>
<td>79</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>144332012</td>
<td>rs6558339</td>
<td>T</td>
<td>C</td>
<td>200</td>
<td>ZFP41</td>
<td>ENST00000330701</td>
<td>32</td>
<td>37</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>115818949</td>
<td>rs2282076</td>
<td>A</td>
<td>T</td>
<td>200</td>
<td>ZFP37</td>
<td>ENST00000374227</td>
<td>18</td>
<td>43</td>
</tr>
</tbody>
</table>
Using `filter` with other string operations

```r
startsWith(col, "text")
endsWith(col, "text")
grepl(pattern, col)
```

These are different to the select helpers `starts_with` and `ends_with` which are used for picking columns
Using `filter` with `is_` functions

```r
# A tibble: 8 x 2
  sample value
  <chr>  <dbl>
1 A      9.98
2 A      8.58
3 A      10.4
4 A      11.4
5 B      9.75
6 B      11.2
7 B      NA
8 B      NA
```

```r
data.with.na %>%
  filter(!is.na(value))
```

```r
# A tibble: 6 x 2
  sample value
  <chr>  <dbl>
1 A      9.98
2 A      8.58
3 A      10.4
4 A      11.4
5 B      9.75
6 B      11.2
```

`is_numeric`  
`is_character`  
`is_finite`  
`is_infinite`  
`is_null`  
`is_empty`
### Transforming data in a filter

```r
trumpton %>%
  filter(log(Height) > 5)
```

```r
# A tibble: 6 x 5

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LastName</td>
<td>FirstName</td>
<td>Age</td>
<td>Weight</td>
<td>Height</td>
</tr>
<tr>
<td>---</td>
<td>----------</td>
<td>-----------</td>
<td>-----</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>1</td>
<td>Hugh</td>
<td>Chris</td>
<td>26</td>
<td>90</td>
<td>175</td>
</tr>
<tr>
<td>2</td>
<td>Pew</td>
<td>Adam</td>
<td>32</td>
<td>102</td>
<td>183</td>
</tr>
<tr>
<td>3</td>
<td>Barney</td>
<td>Daniel</td>
<td>18</td>
<td>88</td>
<td>168</td>
</tr>
<tr>
<td>4</td>
<td>McGrew</td>
<td>Chris</td>
<td>48</td>
<td>97</td>
<td>155</td>
</tr>
<tr>
<td>5</td>
<td>Cuthbert</td>
<td>Carl</td>
<td>28</td>
<td>91</td>
<td>188</td>
</tr>
<tr>
<td>6</td>
<td>Grub</td>
<td>Doug</td>
<td>31</td>
<td>89</td>
<td>164</td>
</tr>
</tbody>
</table>
```

log, abs, sqrt, nchar, substr, tolower, toupper, etc.
Exercise 3
More clever filtering
Restructuring Data
'Tidy' Data Format

• Tibbles give you a 2D data structure where each column must be of a fixed data type
• Often data can be put into this sort of structure in more than one way
• Is there a right / wrong way to structure your data?

• Tidyverse has an opinion!
Long vs Wide Data Modelling

• Consider a simple experiment:

• Two genes tested (ABC1 and DEF1)
• Two conditions (WT and KO)
• Three replicates for each condition
Wide Format

- Compact
- Easy to read
- Shows linkage for genes
- No explicit genotype or replicate
- Values spread out over multiple rows and columns
- Not extensible to more metadata

<table>
<thead>
<tr>
<th>Gene</th>
<th>WT_1</th>
<th>WT_2</th>
<th>WT_3</th>
<th>KO_1</th>
<th>KO_2</th>
<th>KO_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC1</td>
<td>8.86</td>
<td>4.18</td>
<td>8.90</td>
<td>4.00</td>
<td>14.52</td>
<td>13.39</td>
</tr>
<tr>
<td>DEF1</td>
<td>29.60</td>
<td>41.22</td>
<td>36.15</td>
<td>11.18</td>
<td>16.68</td>
<td>1.64</td>
</tr>
</tbody>
</table>
### Long Format

<table>
<thead>
<tr>
<th>Gene</th>
<th>Genotype</th>
<th>Replicate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC1</td>
<td>WT</td>
<td>1</td>
<td>8.86</td>
</tr>
<tr>
<td>ABC1</td>
<td>WT</td>
<td>2</td>
<td>4.18</td>
</tr>
<tr>
<td>ABC1</td>
<td>WT</td>
<td>3</td>
<td>8.90</td>
</tr>
<tr>
<td>ABC1</td>
<td>KO</td>
<td>1</td>
<td>4.00</td>
</tr>
<tr>
<td>ABC1</td>
<td>KO</td>
<td>2</td>
<td>14.52</td>
</tr>
<tr>
<td>ABC1</td>
<td>KO</td>
<td>3</td>
<td>13.39</td>
</tr>
<tr>
<td>DEF1</td>
<td>WT</td>
<td>1</td>
<td>29.60</td>
</tr>
<tr>
<td>DEF1</td>
<td>WT</td>
<td>2</td>
<td>41.22</td>
</tr>
<tr>
<td>DEF1</td>
<td>WT</td>
<td>3</td>
<td>36.15</td>
</tr>
<tr>
<td>DEF1</td>
<td>KO</td>
<td>1</td>
<td>11.18</td>
</tr>
<tr>
<td>DEF1</td>
<td>KO</td>
<td>2</td>
<td>16.68</td>
</tr>
<tr>
<td>DEF1</td>
<td>KO</td>
<td>3</td>
<td>1.64</td>
</tr>
</tbody>
</table>

- More verbose (repeated values)
- Explicit genotype and replicate
- All values in a single column
- Extensible to more metadata
Converting to "Tidy" format

# A tibble: 3 x 8
  Gene  Chr  Start     End Sample1_WT Sample2_WT Sample3_KO Sample4_KO
  <chr> <dbl>   <dbl>   <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 Gnai3    2  163898  167465       9.39       10.9       33.5       81.9
2 Pbsn     5  4888573 4891351      91.7        59.6       45.3       82.3
3 Cdc45    7  1250084 1262669      69.2        36.1       54.4       38.1

• Put all measures into a single column
• Add a 'sample' and 'genotype' column
• Duplicate the gene information as required
  – Or separate it into a different table
Converting to "Tidy" format

```r
# A tibble: 3 x 8
  Gene  Chr Start  End  WT_1  WT_2  KO_1  KO_2
  <chr> <dbl>   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>
1 Gnai3     2   163898 167465  9.39  10.9  33.5  81.9
2 Pbsn      5  4888573 4891351 91.7  59.6  45.3  82.3
3 Cdc45     7  1250084 1262669 69.2  36.1  54.4  38.1
```

non.normalised %>%
gather(sample, value, -Gene,-Chr,-Start,-End) %>%
separate(sample,into=c("genotype","replicate"),convert = TRUE,sep="_")
Converting to "Tidy" format

```r
# A tibble: 12 x 7
  Gene Chr Start    End  genotype replicate value
  <chr> <dbl>   <dbl>  <dbl>      <chr>   <int>     <dbl>
1     Gnai3   2   163898  167465  WT           1       9.39
2      Pbsn   5  4888573  4891351 WT           1      91.7
3     Cdc45   7  1250084 1262669  WT           1      69.2
4     Gnai3   2   163898  167465  WT           2      10.9
5      Pbsn   5  4888573  4891351 WT           2      59.6
6     Cdc45   7  1250084 1262669  WT           2      36.1
7     Gnai3   2   163898  167465  KO           1      33.5
8      Pbsn   5  4888573  4891351 KO           1      45.3
9     Cdc45   7  1250084 1262669  KO           1      54.4
10    Gnai3   2   163898  167465  KO           2     81.9
11     Pbsn   5  4888573  4891351 KO           2     82.3
12    Cdc45   7  1250084 1262669  KO           2     38.1
```
Tidying operations

• **gather**
  – Takes multiple columns of the same type and puts them into a pair of key-value columns

• **separate**
  – Splits a delimited column into multiple columns

• **spread**
  – Takes a key-value column pair and spreads them out to multiple columns of the same type

• **unite**
  – Combines multiple columns into one
Converting to "Tidy" format

```r
# A tibble: 3 x 8
     Gene   Chr  Start   End  WT_1  WT_2  KO_1  KO_2
     <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Gna13    2  163898 167465  9.39  10.9  33.5  81.9
 2 Pbsn     5 4888573 4891351 91.7  59.6  45.3  82.3
 3 Cdc45    7 1250084 1262669 69.2  36.1  54.4  38.1
```

non.normalised %>%
  gather(sample, value, -Gene,-Chr,-Start,-End) %>%
  separate(sample,into=c("genotype","replicate"),convert = TRUE, sep="_")

convert=TRUE makes separate re-detect the type of the column, so replicate becomes a numeric value
Converting to "Tidy" format

- Can use `select` to split duplicated information into separate tibbles

---

# A tibble: 12 x 4

<table>
<thead>
<tr>
<th>Gene</th>
<th>genotype</th>
<th>replicate</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;chr&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;int&gt;</td>
<td>&lt;dbl&gt;</td>
</tr>
<tr>
<td>1</td>
<td>Gnai3</td>
<td>WT</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Pbsn</td>
<td>WT</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Cdc45</td>
<td>WT</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Gnai3</td>
<td>WT</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Pbsn</td>
<td>WT</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Cdc45</td>
<td>WT</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Gnai3</td>
<td>KO</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Pbsn</td>
<td>KO</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>Cdc45</td>
<td>KO</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Gnai3</td>
<td>KO</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>Pbsn</td>
<td>KO</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>Cdc45</td>
<td>KO</td>
<td>2</td>
</tr>
</tbody>
</table>

---

# A tibble: 3 x 4

<table>
<thead>
<tr>
<th>Gene</th>
<th>Chr</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;chr&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;dbl&gt;</td>
</tr>
<tr>
<td>1</td>
<td>Gnai3</td>
<td>2</td>
<td>163898</td>
</tr>
<tr>
<td>2</td>
<td>Pbsn</td>
<td>5</td>
<td>4888573</td>
</tr>
<tr>
<td>3</td>
<td>Cdc45</td>
<td>7</td>
<td>1250084</td>
</tr>
</tbody>
</table>

- These can be recombined later on as needed.
Exercise 4
Restructuring data into ‘tidy’ format
Mutating, Grouping and Summarising
Mutating, Grouping and Summarising

• **mutate** create a new variable from existing variables

• **group_by** sets groups for summarisation

• **ungroup** removes grouping information

• **summarise** collapse grouped variables

• **count** count grouped variables
Creating columns with `mutate`

```r
trumpton %>% mutate(bmi = Weight/(Height/100)^2)
```

# A tibble: 7 x 6

<table>
<thead>
<tr>
<th></th>
<th>LastName</th>
<th>FirstName</th>
<th>Age</th>
<th>Weight</th>
<th>Height</th>
<th>bmi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hugh</td>
<td>Chris</td>
<td>26</td>
<td>90</td>
<td>175</td>
<td>29.4</td>
</tr>
<tr>
<td>2</td>
<td>Pew</td>
<td>Adam</td>
<td>32</td>
<td>102</td>
<td>183</td>
<td>30.5</td>
</tr>
<tr>
<td>3</td>
<td>Barney</td>
<td>Daniel</td>
<td>18</td>
<td>88</td>
<td>168</td>
<td>31.2</td>
</tr>
<tr>
<td>4</td>
<td>McGrew</td>
<td>Chris</td>
<td>48</td>
<td>97</td>
<td>155</td>
<td>40.4</td>
</tr>
<tr>
<td>5</td>
<td>Cuthbert</td>
<td>Carl</td>
<td>28</td>
<td>91</td>
<td>188</td>
<td>25.7</td>
</tr>
<tr>
<td>6</td>
<td>Dibble</td>
<td>Liam</td>
<td>35</td>
<td>94</td>
<td>145</td>
<td>44.7</td>
</tr>
<tr>
<td>7</td>
<td>Grub</td>
<td>Doug</td>
<td>31</td>
<td>89</td>
<td>164</td>
<td>33.1</td>
</tr>
</tbody>
</table>
Tricks with `mutate` – Creating categories

```r
trumpton %>%
  mutate(Category = if_else(Height > 180, "Tall", "Short"))
```

# A tibble: 7 x 6

<table>
<thead>
<tr>
<th>LastName</th>
<th>FirstName</th>
<th>Age</th>
<th>Weight</th>
<th>Height</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hugh</td>
<td>Chris</td>
<td>26</td>
<td>90</td>
<td>175</td>
<td>Short</td>
</tr>
<tr>
<td>Pew</td>
<td>Adam</td>
<td>32</td>
<td>102</td>
<td>183</td>
<td>Tall</td>
</tr>
<tr>
<td>Barney</td>
<td>Daniel</td>
<td>18</td>
<td>88</td>
<td>168</td>
<td>Short</td>
</tr>
<tr>
<td>McGrew</td>
<td>Chris</td>
<td>48</td>
<td>97</td>
<td>155</td>
<td>Short</td>
</tr>
<tr>
<td>Cuthbert</td>
<td>Carl</td>
<td>28</td>
<td>91</td>
<td>188</td>
<td>Tall</td>
</tr>
<tr>
<td>Dibble</td>
<td>Liam</td>
<td>35</td>
<td>94</td>
<td>145</td>
<td>Short</td>
</tr>
<tr>
<td>Grub</td>
<td>Doug</td>
<td>31</td>
<td>89</td>
<td>164</td>
<td>Short</td>
</tr>
</tbody>
</table>
Tricks with `mutate` – replacing values

data.with.na %>%
  mutate(value = replace(value, value > 10, 10))

> data.with.na
# A tibble: 8 x 2
  sample value
  <chr>  <dbl>
1 A      9.98
2 A      8.58
3 A      10.4
4 A      11.4
5 B      9.75
6 B      11.2
7 B      NA
8 B      NA

# A tibble: 8 x 2
  sample value
  <chr>  <dbl>
1 A      9.98
2 A      8.58
3 A      10
4 A      10
5 B      9.75
6 B      10
7 B      0
8 B      0

data.with.na %>%
  mutate(value = replace_na(value, 0))
Grouping and Summarising Workflow

1. Load a tibble with repeated values in one or more columns

2. Use `group_by` to select all of the categorical columns you want to combine to define your groups

3. Run `summarise` saying how you want to combine the quantitative values

4. Run `ungroup` to remove any remaining group information
Grouping and Summarising Workflow

1. Load a tibble with repeated values in one or more columns

2. Use `group_by` to select all of the categorical columns you want to combine to define your groups

3. Run `summarise` saying how you want to combine the quantitative values

4. Run `ungroup` to remove any remaining group information
Grouping and Summarising

> group.data

# A tibble: 8 x 5

<table>
<thead>
<tr>
<th>Sample</th>
<th>Genotype</th>
<th>Sex</th>
<th>Height</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>WT</td>
<td>M</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>WT</td>
<td>F</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>WT</td>
<td>F</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>WT</td>
<td>M</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>KO</td>
<td>M</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>KO</td>
<td>F</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>KO</td>
<td>F</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>KO</td>
<td>M</td>
<td>27</td>
</tr>
</tbody>
</table>

• Want to get the average Height and Length for each combination of sex and genotype
Grouping and Summarising
Workflow

1. Load a tibble with repeated values in one or more columns

2. Use `group_by` to select all of the categorical columns you want to combine to define your groups

3. Run `summarise` saying how you want to combine the quantitative values

4. Run `ungroup` to remove any remaining group information
## Grouping and Summarising

<table>
<thead>
<tr>
<th>Discard</th>
<th>Group</th>
<th>Group</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="X" /></td>
<td><img src="image" alt="✓" /></td>
<td><img src="image" alt="✓" /></td>
<td><img src="image" alt="✓" /></td>
<td><img src="image" alt="✓" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample</th>
<th>Genotype</th>
<th>Sex</th>
<th>Height</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;dbl&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;chr&gt;</td>
<td>&lt;dbl&gt;</td>
<td>&lt;dbl&gt;</td>
</tr>
</tbody>
</table>

- **Categorical**
- **Quantitative**

- Want to get the average Height and Length for each combination of sex and genotype
Grouping and Summarising

```r
# A tibble: 8 x 5
# Groups:   Genotype, Sex [4]
     Sample Genotype Sex   Height Length
       <dbl>    <chr> <chr>  <dbl>  <dbl>
1       1      WT  M         15    200
2       2      WT  F         13    185
3       3      WT  F         14    221
4       4      WT  M         18    265
5       5      KO  M         26    120
6       6      KO  F         22    165
7       7      KO  F         19    143
8       8      KO  M         27    110
```

Discard Group Group Mean Median
Sample Genotype Sex Height Length
<dbl> <chr>    <chr>  <dbl>  <dbl>
Discard
 Group Group
 Mean Median

```r
group.data %>% group_by(Genotype,Sex)
```
Grouping and Summarising Workflow

1. Load a tibble with repeated values in one or more columns

2. Use `group_by` to select all of the categorical columns you want to combine to define your groups

3. Run `summarise` saying how you want to combine the quantitative values

4. Run `ungroup` to remove any remaining group information
Grouping and Summarising

group.data %>%
group_by(Genotype, Sex) %>%
count()

# A tibble: 4 x 3
# Groups:   Genotype, Sex [4]

  Genotype Sex    n
  <chr>   <chr> <int>
1 KO      F      2
2 KO      M      2
3 WT      F      2
4 WT      M      2
Grouping and Summarising

```r
group.data %>%
  group_by(Genotype, Sex) %>%
  summarise(Height2 = mean(Height), Length = median(Length))
```

# A tibble: 4 x 4
# Groups:   Genotype [2]

<table>
<thead>
<tr>
<th>Genotype</th>
<th>Sex</th>
<th>Height2</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>KO</td>
<td>F</td>
<td>20.5</td>
<td>154</td>
</tr>
<tr>
<td>KO</td>
<td>M</td>
<td>26.5</td>
<td>115</td>
</tr>
<tr>
<td>WT</td>
<td>F</td>
<td>13.5</td>
<td>203</td>
</tr>
<tr>
<td>WT</td>
<td>M</td>
<td>16.5</td>
<td>232.</td>
</tr>
</tbody>
</table>

If you want the count of values as part of a summarised result use the n() function.
Grouping and Summarising Workflow

1. Load a tibble with repeated values in one or more columns

2. Use `group_by` to select all of the categorical columns you want to combine to define your groups

3. Run `summarise` saying how you want to combine the quantitative values

4. Run `ungroup` to remove any remaining group information
Ungrouping

• A summarise operation removes the last level of grouping ("Sex" in our worked example)

• Other levels of grouping ("Genotype") remain annotated on the data, so you could do an additional summarisation if needed

• If you’re not going to use them it’s a good idea to use `ungroup` to remove remaining groups so they don’t interfere with other operations
Grouping affects lots of operations
Find the tallest member of each Sex

group.data %>%
  arrange(desc(Height)) %>%
  group_by(Sex) %>%
  slice(1)

# A tibble: 2 x 5
# Groups:   Sex [2]
  Sample Genotype Sex   Height Length
  <dbl> <chr>    <chr>  <dbl>  <dbl>
1    6   KO      F       22    165
2    8   KO      M       27    110
Normalise the Length by the average for that sex

# Grouping affects lots of operations

group.data %>%
  mutate(Diff = Length - mean(Length))
Exercise 5
Mutating, Grouping and Summarising
Extending tibbles

- `add_row` adds an additional row
- `bind_rows` join two tibbles by row
- `add_column` adds an additional column
- `bind_cols` join two tibbles by column
- `rename` renames an existing column
Joining tibbles \(x\) and \(y\)

- **left_join**: join matching values from \(y\) into \(x\)
- **right_join**: join matching values of \(x\) into \(y\)
- **inner_join**: join \(x\) and \(y\) keeping only rows in both
- **full_join**: join \(x\) and \(y\) keeping all values in both
### Join types

#### join1

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simon</td>
<td>3</td>
</tr>
<tr>
<td>Steven</td>
<td>6</td>
</tr>
<tr>
<td>Felix</td>
<td>2</td>
</tr>
</tbody>
</table>

#### join2

<table>
<thead>
<tr>
<th>name</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Felix</td>
<td>10</td>
</tr>
<tr>
<td>Anne</td>
<td>25</td>
</tr>
<tr>
<td>Simon</td>
<td>36</td>
</tr>
</tbody>
</table>

#### right_join

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Felix</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Anne</td>
<td>NA</td>
<td>25</td>
</tr>
<tr>
<td>Simon</td>
<td>3</td>
<td>36</td>
</tr>
</tbody>
</table>

#### left_join

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simon</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>Steven</td>
<td>6</td>
<td>NA</td>
</tr>
<tr>
<td>Felix</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

#### inner_join

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simon</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>Felix</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

#### full_join

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simon</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>Steven</td>
<td>6</td>
<td>NA</td>
</tr>
<tr>
<td>Felix</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Anne</td>
<td>NA</td>
<td>25</td>
</tr>
</tbody>
</table>
Joining options

• **by** specify the columns to join on
  - Simple name if it’s the same between both
    by="gene"
  - Paired names if they differ between x and y
    by=c("gene" = "gene_name")

• **suffix** the text suffix for duplicated column names
Rejoining split tables
Find the highest value for each genotype

```r
> gathered.data
# A tibble: 12 x 4
   Gene genotype replicate value
   <chr> <chr>        <int> <dbl>
1  Gnai3  WT           1   9.39
2  Pbsn   WT           1  91.7
3  Cdc45  WT           1  69.2
4  Gnai3  WT           2  10.9
5  Pbsn   WT           2  59.6
6  Cdc45  WT           2  36.1
7  Gnai3  KO           1  33.5
8  Pbsn   KO           1  45.3
9  Cdc45  KO           1  54.4
10 Gnai3  KO           2  81.9
11 Pbsn   KO           2  82.3
12 Cdc45  KO           2  38.1
```

```r
> gathered.annotation
# A tibble: 3 x 4
     Gene Chr Start     End
   <chr> <dbl>   <dbl>   <dbl>
1  Gnai3    2  163898  167465
2  Pbsn     5  4888573 4891351
3  Cdc45    7 1250084 1262669
```
Rejoining split tables
Find the highest value for each genotype

gathered.data %>% arrange(desc(value)) %>% group_by(genotype) %>% slice(1)

# A tibble: 2 x 4
# Groups:   genotype [2]
Gene  genotype replicate value
<chr> <chr>        <int> <dbl>
1 Pbsn KO           2  82.3
2 Pbsn WT           1  91.7

gathered.data %>% arrange(desc(value)) %>% group_by(genotype) %>% slice(1) %>%
left_join(gathered.annotation)

# A tibble: 2 x 7
# Groups:   genotype [2]
Gene  genotype replicate value Chr Start   End     <chr> <chr>        <int> <dbl> <dbl> <dbl>
1 Pbsn KO           2  82.3  5 4888573 4891351
2 Pbsn WT           1  91.7  5 4888573 4891351
Exercise 6
Extending and Joining
Custom Functions

bmi <- function(w, h) {
  h <- h/100
  h = h^2
  return(w/h)
}

> bmi(90,175)
[1] 29.38776

> bmi(c(90,102), c(175,183))
[1] 29.38776 30.45776
summarise.gene <- function(tbl, genename) {
  tbl %>%
    filter(GENE == genename) %>%
    filter(str_length(REF) == 1, str_length(ALT) == 1) %>%
    group_by(REF, ALT) %>%
    count()
}

child %>%
  summarise.gene("PLEC")

# A tibble: 6 x 3
# Groups:   REF, ALT [6]
  REF ALT  n
1 A   C   1
2 A   G   9
3 C   T   6
4 G   A   8
5 T   C   6
6 T   G   1
Custom functions with mutate

\[ \text{trumpton} \%\% \text{mutate}(\text{bmi}=\text{Weight}/(\text{Height}/100)^2) \]

\[ \text{trumpton} \%\% \text{mutate}(\text{bmi} = \text{bmi}(\text{Weight}, \text{Height})) \]

\[
\begin{align*}
\text{bmi} &\leftarrow \text{function}(w, h) \{
    h &\leftarrow h/100 \\
    h &\leftarrow h^2 \\
    h &\leftarrow h^2 \\
    \text{return}(w/h)
\}\}
\end{align*}
\]
Custom functions and grouping

```r
trumpton %>%
  mutate(agegroup = ifelse(Age > 30, "Old", "Young")) %>%
  group_by(agegroup) %>%
  summarise(bmi = mean(bmi(Weight, Height)))

# A tibble: 2 x 2
agegroup   bmi
<chr>       <dbl>
1 Old        37.2
2 Young      28.8
```

plot.fireman <- function(data) {
  data %>%
  ggplot(aes(Weight,Height, size=Age)) +
  geom_point() +
  ggtitle(data$old)
}

trumpton %>%
  mutate(old = ifelse(Age>30, "Old", "Young")) %>%
  group_by(old) %>%
  do(plots = plot.fireman(.))

# A tibble: 2 x 2
old  plots
* <chr> <list>
1 Old    <gg>
2 Young  <gg>
plot.fireman <- function(data) {
  data %>%
  ggplot(aes(Weight, Height, size=Age)) +
  geom_point() +
  ggtitle(data$old)
}

trumpton %>%
  mutate(old=ifelse(Age>30,"Old","Young")) %>%
  group_by(old) %>%
  do(plots=plot.fireman(.)) %>%
  pull(plots)
Exercise 7
Custom Functions