

Advanced R

(with Tidyverse)

Simon Andrews

V2021-09

Course Content

- Expanding knowledge
 - More functions and operators
- Improving efficiency
 - More options for elegant code
- Awkward cases
 - Dealing with real data
- Tidyverse operations
 - Data Import
 - Filtering, selecting and sorting
 - Restructuring data
 - Grouping and Summarising
 - Extending and Merging
- Custom functions

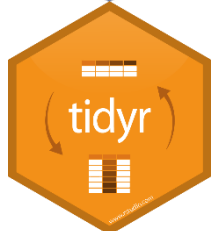
Tidyverse Packages



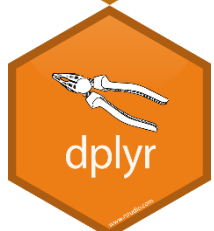
- Tibble - data storage



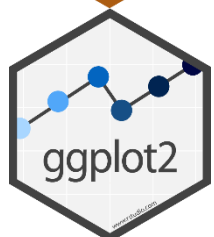
- ReadR - reading data from files



- TidyR - Model data correctly



- DplyR - Manipulate and filter data



- Ggplot2 - Draw figures and graphs





Reading Files with readr

- Tidyverse functions for reading text files into tibbles
 - `read_delim("file.txt")`
`read_csv("file.csv")`
`read_tsv("file.tsv")`
 - `read_fwf("file.txt", col_positions=c(1, 3, 6))`
 - `read_excel("file.xlsx", sheet="Sheet1")`



The `read_excel` function isn't in core tidyverse. You need to install the `readxl` package to get it

Reading files with readr



```
> read_delim("trumpton.txt") -> trumpton
Rows: 7 Columns: 5
```

```
-- Column specification -----
```

```
Delimiter: "\t"
```

```
chr (2): LastName, FirstName
```

```
dbl (3): Age, weight, Height
```

```
i Use `spec()` to retrieve the full column specification for this data.
```

```
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
> 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

Reading files with readr



```
chr (2): LastName, FirstName  
dbl (3): Age, weight, Height
```

```
> spec(trumpton)  
cols(  
  LastName = col_character(),  
  FirstName = col_character(),  
  Age = col_double(),  
  Weight = col_double(),  
  Height = col_double()  
)
```



Import Problems

```
> read_delim("import_problems.txt") -> problem_data
```

```
Rows: 1042 Columns: 4
```

```
-- Column specification -----
```

```
Delimiter: "\t"
```

```
chr (2): Gene, Significance
```

```
dbl (2): Chr, Expression
```

```
> head(problem_data)
```

	Chr	Gene	Expression	Significance
	<dbl>	<chr>	<dbl>	<chr>
1	1	Depdc2	9.19	NS
3	1	Rpl7	8.75	0.050626416

```
> tail(problem_data)
```

	Chr	Gene	Expression	Significance
	<dbl>	<chr>	<dbl>	<chr>
4	6	2610528E23Rik	9.55	NS
6	NA	Huwe1	8.54	NS

Warning message:

One or more parsing issues, see `problems()` for details

Import Problems



```
> problems(problem_data)

# A tibble: 1 x 5
  row    col expected actual file
<int> <int> <chr>    <chr> <chr>
1  1042     1 a double Y   import_problems.txt
```

- Data not matching the guessed type
- Wrong number of fields

Lazy Evaluation



- When `read_delim` is run all lines are found but only 1000 lines are read
- More data is read interactively as the file is actually used

+ Faster to load initially

- Problems possibly not spotted until later in the script
- File is held open so can't be overwritten

- Calling `problems()` forces the whole file to be read
- Lazy evaluation can be disabled with `lazy=FALSE`

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 Su1f1         9.66 NS
3     1 Rp17          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
```

- Chr should be character
- Significance should be numeric

Fixing guessed columns



```
read_delim(  
  "import_problems.txt",  
  lazy=FALSE,  
  col_types=cols(Chr=col_character(), significance=col_double())  
)
```

One or more parsing issues, see `problems()` for details

```
> problems()
```

```
# A tibble: 874 x 5
```

	row	col	expected	actual	file
	<int>	<int>	<chr>	<chr>	<chr>
1	2	4	a double	NS	import_problems.txt
2	3	4	a double	NS	import_problems.txt
3	5	4	a double	NS	import_problems.txt
4	6	4	a double	NS	import_problems.txt
5	7	4	a double	NS	import_problems.txt
6	9	4	a double	NS	import_problems.txt
7	10	4	a double	NS	import_problems.txt
8	11	4	a double	NS	import_problems.txt
9	13	4	a double	NS	import_problems.txt
10	14	4	a double	NS	import_problems.txt

```
# ... with 864 more rows
```

```
# 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
```

Unwanted header lines



```
read_delim("unwanted_headers.txt")
```

```
Error: Could not guess the delimiter.
```

```
read_csv("unwanted_headers.txt")
```

```
Delimiter: ","
```

```
chr (1): # Format version 1.0
```

```
> problems()
```

```
# A tibble: 4 x 5
```

	row	col	expected	actual	file
	<int>	<int>	<chr>	<chr>	<chr>
1	3	5	1 columns	5 columns	unwanted_headers.txt
2	4	5	1 columns	5 columns	unwanted_headers.txt
3	5	5	1 columns	5 columns	unwanted_headers.txt
4	6	5	1 columns	5 columns	unwanted_headers.txt

```
# Format version 1.0
```

```
# Created 20/05/2020
```

```
Gene,Strand,Group_A,Group_B,Group_C
```

```
ABC1,+,5.30,4.69,4.84
```

```
DEF1,-,14.97,15.66,15.92
```

```
HIJ1,-,2.17,3.14,1.94
```

```
# A tibble: 5 x 1
```

```
`# Format version 1.0`
```

```
<chr>
```

```
1 # Created 20/05/2020
```

```
2 Gene
```

```
3 ABC1
```

```
4 DEF1
```

```
5 HIJ1
```

Unwanted header lines



```
read_delim(  
  "unwanted_headers.txt",  
  lazy=FALSE,  
  skip=2  
)
```

```
# Format version 1.0  
# Created 20/05/2020  
Gene,Strand,Group_A,Group_B,Group_C  
ABC1,+,5.30,4.69,4.84  
DEF1,-,14.97,15.66,15.92  
HIJ1,-,2.17,3.14,1.94
```

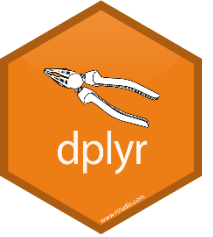
```
read_delim(  
  "unwanted_headers.txt",  
  lazy=FALSE,  
  comment="#"  
)
```

```
# A tibble: 3 x 5  
  Gene   Strand Group_A Group_B Group_C  
  <chr> <chr>   <dbl>  <dbl>  <dbl>  
1 ABC1  +       5.3     4.69   4.84  
2 DEF1  -      15.0    15.7   15.9  
3 HIJ1  -       2.17    3.14   1.94
```

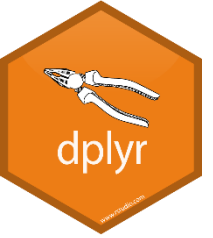
chr (2): Gene, Strand
dbl (3): Group_A, Group_B, Group_C

Exercise 1

Reading Data into Tibbles

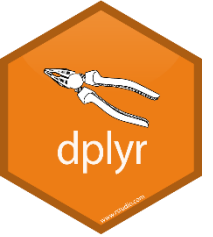


Filtering, Selecting, Sorting etc.



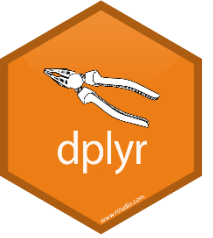
Subsetting and Filtering

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



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
```



Using `select` or `slice`

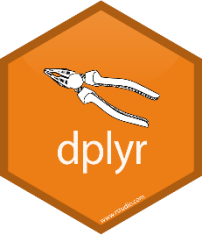
```
select(data, cols)  
slice(data, rows)
```

```
trumpton %>%  
  select(LastName, Age, Height)
```

```
# 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
```

```
trumpton %>%  
  slice(1,4,7)
```

```
# 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
```

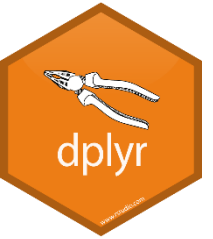


Using `select` and `slice`

```
trumpton %>%  
  select(LastName, Age, Height) %>%  
  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
```

Functional row selection using `filter`

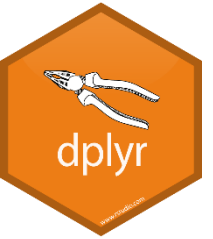


```
trumpton %>%  
  filter(Height>170)
```

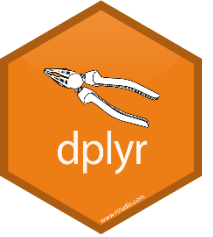
```
# A tibble: 3 x 5
```

	LastName	FirstName	Age	Weight	Height
	<chr>	<chr>	<dbl>	<dbl>	<dbl>
1	Hugh	Chris	26	90	175
2	Pew	Adam	32	102	183
3	Cuthbert	Carl	28	91	188

Using `select` helpers



<code>colnames(child.variants)</code>	CHR POS dbSNP REF ALT QUAL GENE ENST MutantReads COVERAGE MutantReadPercent
<code>child.variants %>%</code>	
<code> select(REF, COVERAGE)</code>	REF COVERAGE
<code> select(REF, everything())</code>	REF CHR POS dbSNP ALT QUAL GENE ENST MutantReads COVERAGE MutantReadPercent
<code> select(-CHR, -ENST)</code>	POS dbSNP REF ALT QUAL GENE MutantReads COVERAGE MutantReadPercent
<code> select(-REF, everything())</code>	CHR POS dbSNP ALT QUAL GENE ENST MutantReads COVERAGE MutantReadPercent REF
<code> select(5:last_col())</code>	ALT QUAL GENE ENST MutantReads COVERAGE MutantReadPercent
<code> select(POS:GENE)</code>	POS dbSNP REF ALT QUAL GENE
<code> select(-(POS:GENE))</code>	CHR ENST MutantReads COVERAGE MutantReadPercent
<code> select(starts_with("Mut"))</code>	MutantReads MutantReadPercent
<code> select(-ends_with("t", ignore.case = FALSE))</code>	
<code> select(contains("Read"))</code>	CHR POS dbSNP REF ALT QUAL GENE ENST MutantReads COVERAGE MutantReads MutantReadPercent



arrange (sorting)

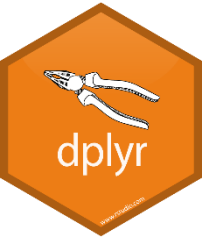
distinct (deduplication)

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

```
# A tibble: 6 x 5  
  LastName FirstName    Age weight Height  
  <chr>    <chr>    <dbl> <dbl> <dbl>  
1 Dibble   Liam      35     94    145  
2 McGrew Chris    48     97    155  
3 Grub     Doug      31     89    164  
4 Barney  Daniel    18     88    168  
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 distinct column. “**keep_all**” has a dot before it



arrange (sorting)

distinct (deduplication)

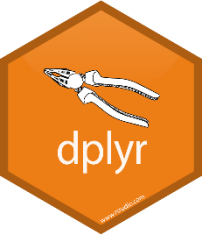
```
trumpton %>%  
  arrange(desc(Height)) %>%  
  distinct(FirstName, .keep_all = TRUE)
```

A tibble: 6 x 5

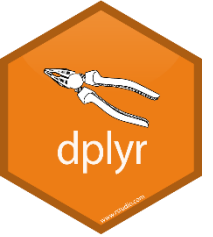
	LastName	FirstName	Age	Weight	Height
	<chr>	<chr>	<dbl>	<dbl>	<dbl>
1	Cuthbert	Carl	28	91	188
2	Pew	Adam	32	102	183
3	Hugh	Chris	26	90	175
4	Barney	Daniel	18	88	168
5	Grub	Doug	31	89	164
6	Dibble	Liam	35	94	145

Exercise 2

Filtering and selecting



More clever filtering

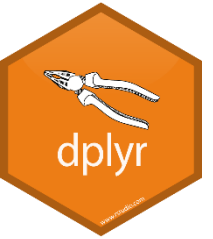


Multi-condition filter

```
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
```

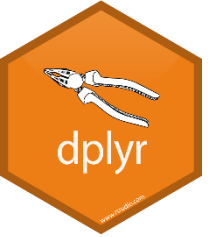
Multi-condition filter



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

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

& = logical AND
| = logical OR
! = logical NOT

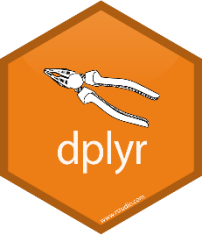


Multi-condition filter

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

& = logical AND
| = logical OR
! = logical NOT

```
# 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
```



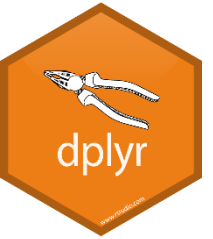
Multi-condition filter

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

```
# A tibble: 1 x 5  
  LastName FirstName    Age Weight Height  
  <chr>    <chr>    <dbl> <dbl> <dbl>  
1 Barney   Daniel     18     88    168
```

& = logical AND
| = logical OR
! = logical NOT

Using filter with %in%



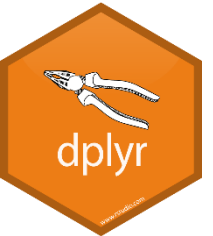
```
> hits  
[1] "FGFR1" "RASAL1" "GLB1L2" "DNAH1" "PTH1R"
```

```
child.variants %>%  
  filter(GENE %in% hits)
```

```
# A tibble: 5 x 11
```

CHR	POS	dbSNP	REF	ALT	QUAL	GENE	ENST	MutantReads	COVERAGE	MutantReadPerce~	
<chr>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	
1	11	134226278	rs3802928	C	T	200	GLB1L2	ENST03898~	13	43	30
2	12	113539822	rs1674101	A	G	200	RASAL1	ENST05465~	19	22	86
3	3	46944274	rs1138518	T	C	200	PTH1R	ENST04495~	32	75	42
4	3	52430526	rs12163565	G	A	200	DNAH1	ENST04203~	38	50	76
5	8	38271182	.	TG	T	200	FGFR1	ENST04259~	9	31	29

Using filter with str_detect

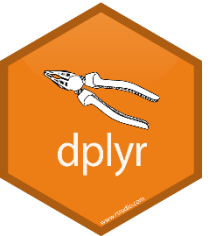


```
child.variants %>%  
  filter(str_detect(GENE, "ZFP"))
```

A tibble: 9 x 11

	CHR	POS	dbSNP	REF	ALT	QUAL	GENE	ENST	MutantReads	COVERAGE	MutantReadPerce~
	<chr>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>
1	16	68598007	rs1177648	A	G	200	ZFP90	ENST0398253	43	100	43
2	16	88552370	rs3751673	A	G	53	ZFPM1	ENST0319555	4	23	17
3	18	5292030	rs620652	A	G	200	ZFP161	ENST0357006	28	71	39
4	19	57065189	rs145011	T	C	200	ZFP28	ENST0301318	59	137	43
5	20	50768672	.	GT	G	200	ZFP64	ENST0216923	36	41	87
6	5	180276402	rs168726	C	T	200	ZFP62	ENST0502412	74	83	89
7	8	106814656	rs2920048	G	C	200	ZFPM2	ENST0407775	33	79	41
8	8	144332012	rs6558339	T	C	200	ZFP41	ENST0330701	32	37	86
9	9	115818949	rs2282076	A	T	200	ZFP37	ENST0374227	18	43	41

Using filter with str_detect

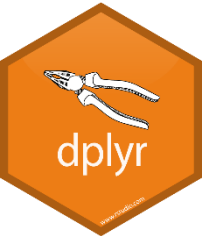


```
child.variants %>%  
  filter(str_detect(GENE, "Z.P"))
```

A tibble: 15 x 11

	CHR	POS	dbSNP	REF	ALT	QUAL	GENE	ENST	MutantReads	COVERAGE	MutantReadPercent
	<chr>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>
1	7	99569394	rs17295356	G	A	200	AZGP1	ENST0292401	9	34	26
2	12	51636259	rs1049467	C	T	200	DAZAP2	ENST0549555	62	68	91
3	3	137786442	rs442800	T	C	200	DZIP1L	ENST0327532	9	32	28
4	3	108403086	rs9856097	T	C	200	DZIP3	ENST0361582	26	30	86
5	20	56179586	rs6123710	G	A	200	ZBP1	ENST0371173	15	44	34
6	18	5292030	rs620652	A	G	200	ZFP161	ENST0357006	28	71	39
7	19	57065189	rs145011	T	C	200	ZFP28	ENST0301318	59	137	43
8	9	115818949	rs2282076	A	T	200	ZFP37	ENST0374227	18	43	41

Using `filter` with other string operations



```
child %>%  
  select(REF,ALT) %>%  
  filter(startswith(REF, "GAT"))
```

```
# A tibble: 3 x 2  
  REF     ALT  
  <chr> <chr>  
1 GATA   G  
2 GATAT GAT  
3 GAT    G
```

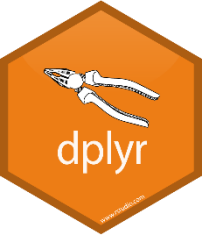
```
child %>%  
  select(GENE,ENST) %>%  
  filter(endswith(ENST, "878"))
```

```
# A tibble: 4 x 2  
  GENE     ENST  
  <chr>   <chr>  
1 CIB3    ENST0269878  
2 KCTD18  ENST0359878  
3 KIAA1407 ENST0295878  
4 RBM33   ENST0401878
```



These are different to the select helpers `starts_with` and `ends_with` which are used for picking columns

Using `filter` with `is` functions



```
> 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
```

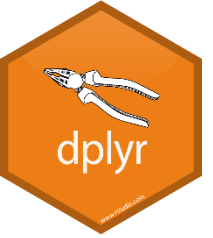
```
data.with.na %>%
  filter(!is.na(value))
```

```
# 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_finite`
`is_infinite`
`is.nan`



Note that some functions have dots whilst others have an underscore.



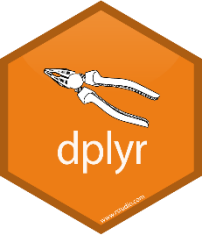
Transforming data in a filter

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

```
# 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 Barney    Daniel     18     88    168  
4 McGrew    Chris      48     97    155  
5 Cuthbert  Carl       28     91    188  
6 Grub      Doug       31     89    164
```

log
abs
sqrt
nchar
substr
tolower
toupper
etc.

Transforming filter examples



```
trumpton %>%  
  filter(str_detect(tolower(LastName), "h"))
```

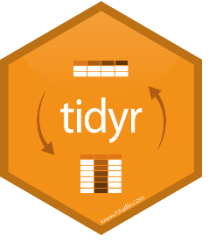
```
trumpton %>%  
  filter(weight*0.16 > 15)
```

```
trumpton %>%  
  filter(nchar(LastName) == nchar(FirstName))
```

log
abs
nchar
str_sub
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?



Journal of Statistical Software
August 2014, Volume 59, Issue 10. <http://www.jstatsoft.org/>

- Tidyverse has an opinion!

Tidy Data

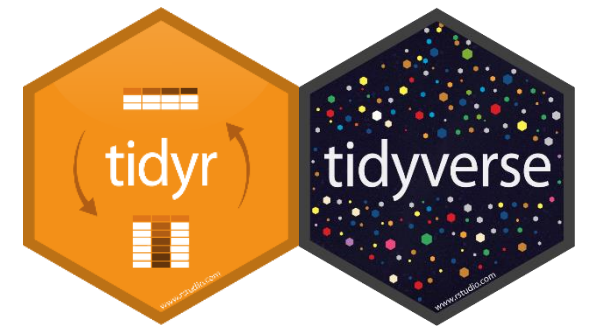
Hadley Wickham
RStudio

Wide Format

Gene	WT_1	WT_2	WT_3	KO_1	KO_2	KO_3
ABC1	8.86	4.18	8.90	4.00	14.52	13.39
DEF1	29.60	41.22	36.15	11.18	16.68	1.64

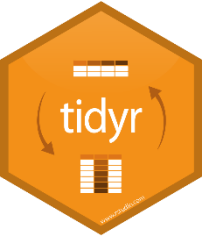
- Compact
- Easy to read
- Shows linkage for genes
- No explicit genotype or replicate
- Values spread out over multiple rows and columns
- Not extensible to more metadata

Long Format



Gene	Genotype	Replicate	Value
ABC1	WT	1	8.86
ABC1	WT	2	4.18
ABC1	WT	3	8.90
ABC1	KO	1	4.00
ABC1	KO	2	14.52
ABC1	KO	3	13.39
DEF1	WT	1	29.60
DEF1	WT	2	41.22
DEF1	WT	3	36.15
DEF1	KO	1	11.18
DEF1	KO	2	16.68
DEF1	KO	3	1.64

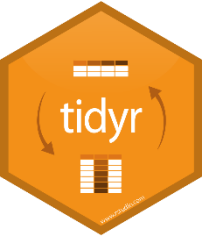
- More verbose (repeated values)
- Explicit genotype and replicate
- All values in a single column
- Extensible to more metadata



Converting to "Tidy" format

```
# A tibble: 3 x 8
  Gene      Chr  Start      End  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
```

- Put all measures into a single column
- Add a 'genotype' and 'replicate' column
- Duplicate the gene information as required
 - Or separate it into a different table



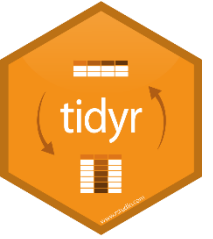
Converting to "Tidy" format

```
# 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 %>%
```

```
  pivot_longer(cols=WT_1:KO_2, names_to="sample", values_to="value") %>%
  separate(sample,into=c("genotype","replicate"),convert = TRUE,sep="_")
```

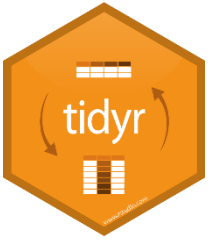
Converting to "Tidy" format



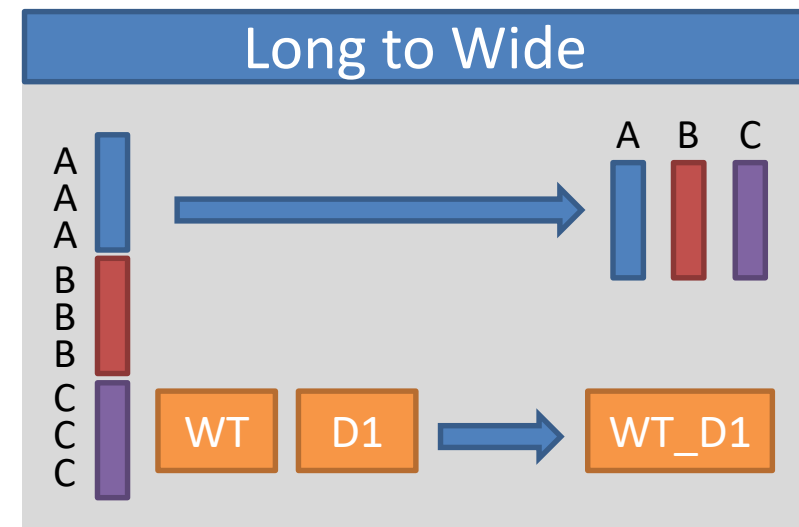
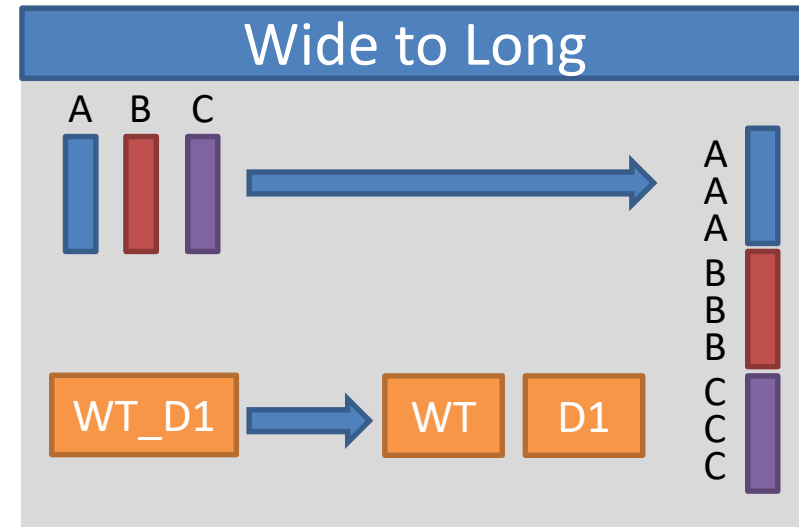
```
# A tibble: 12 x 7
```

```
  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
```

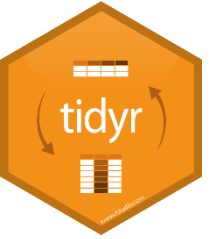
Tidying operations



- **pivot_longer**
 - 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
- **pivot_wider**
 - 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



```
non.normalised %>%
```

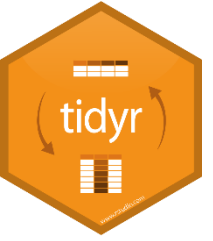
```
  pivot_longer(  
    cols=WT_1:KO_2,  
    names_to="sample",  
    values_to="value"  
  ) %>%
```

```
  separate(  
    col=sample,  
    into=c("genotype", "replicate"),  
    sep="_",  
    convert = TRUE  
  )
```

```
# 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
```



convert=TRUE makes separate re-detect the type of the column, so replicate becomes a numeric value



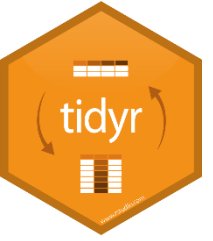
Pivoting Examples

```
> pivot.data
# A tibble: 4 x 3
  gene      WT      KO
  <chr> <dbl> <dbl>
1 ABC1  18608  7831
2 DEF1  31988  55502
3 GHI1   7647  93299
4 JKL1  96002  47945
```

- Log transform all of the values
- Pivot longer
 - Which columns are we pivoting?
 - What do we want to call the new column of names?
 - What do we want to call the new column of values?

```
pivot.data %>%
  pivot_longer(
    cols=WT:KO,
    names_to = "Condition",
    values_to = "Count"
  ) -> pivot.long
```

```
# A tibble: 8 x 3
  gene      Condition Count
  <chr> <chr>      <dbl>
1 ABC1  WT          18608
2 ABC1  KO           7831
3 DEF1  WT          31988
4 DEF1  KO          55502
5 GHI1  WT           7647
6 GHI1  KO          93299
7 JKL1  WT          96002
8 JKL1  KO          47945
```

Pivoting Examples

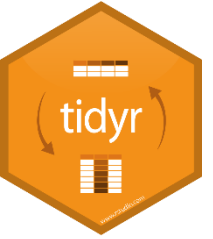
```
> pivot.long
# A tibble: 8 x 3
  gene Condition Count
<chr> <chr>     <dbl>
1 ABC1 WT         14.2
2 ABC1 KO         12.9
3 DEF1 WT         15.0
4 DEF1 KO         15.8
5 GHI1 WT         12.9
6 GHI1 KO         16.5
7 JKL1 WT         16.6
8 JKL1 KO         15.5
```

```
pivot.long %>%
  pivot_wider(
    names_from = Condition,
    values_from = Count
  )
```

```
# A tibble: 4 x 3
  gene      WT      KO
<chr> <dbl> <dbl>
1 ABC1  14.2  12.9
2 DEF1  15.0  15.8
3 GHI1  12.9  16.5
4 JKL1  16.6  15.5
```

- Plot WT vs KO
- Pivot wider
 - Which column of names?
 - Which column of values?

Converting to "Tidy" format



- You can use `select` to split duplicated information into separate tibbles

```
# 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
```

```
# 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
```

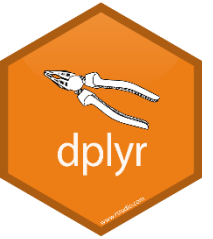
- These can be recombined later on as needed.

Exercise 4

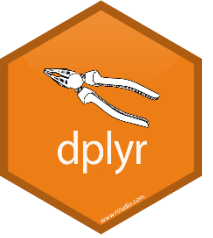
Restructuring data into 'tidy' format

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



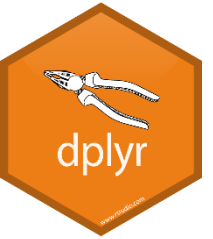
Creating columns with mutate

```
trumpton %>%  
  mutate(  
    weight_stones=Weight*0.16,  
    height_feet=Height*0.033  
  )
```

```
# A tibble: 7 x 7
```

	LastName	FirstName	Age	Weight	Height	weight_stones	height_feet
	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Hugh	Chris	26	90	175	14.4	5.78
2	Pew	Adam	32	102	183	16.3	6.04
3	Barney	Daniel	18	88	168	14.1	5.54
4	McGrew	Chris	48	97	155	15.5	5.12
5	Cuthbert	Carl	28	91	188	14.6	6.20
6	Dibble	Liam	35	94	145	15.0	4.78
7	Grub	Doug	31	89	164	14.2	5.41

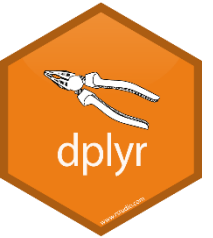
Tricks with mutate – Creating categories



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

```
# A tibble: 7 x 6  
  LastName FirstName   Age Weight Height Category  
  <chr>      <chr>      <dbl> <dbl> <dbl> <chr>  
1 Hugh      Chris       26     90    175 Short  
2 Pew      Adam        32    102    183 Tall  
3 Barney   Daniel       18     88    168 Short  
4 McGrew   Chris       48     97    155 Short  
5 Cuthbert Carl        28     91    188 Tall  
6 Dibble   Liam        35     94    145 Short  
7 Grub     Doug        31     89    164 Short
```

Tricks with mutate – replacing values



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

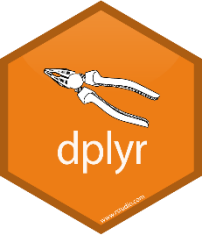
```
> data.with.na  
# A tibble: 8 x 2  
  sample value  
  <chr>   <dbl>  
1 A       9.98  
2 A       8.58  
3 A      10.4  
4 A      11.4  
5 B       9.75  
6 B      11.2  
7 B      NA  
8 B      NA
```

```
# 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      NA  
8 B      NA
```

```
data.with.na %>%  
  mutate(value = replace_na(value,0))
```

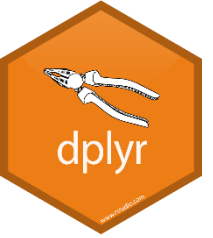
```
# A tibble: 8 x 2  
  sample value  
  <chr>   <dbl>  
1 A       9.98  
2 A       8.58  
3 A      10.4  
4 A      11.4  
5 B       9.75  
6 B      11.2  
7 B       0  
8 B       0
```


Grouping and Summarising Workflow

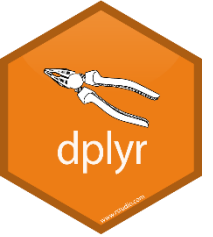


1. Load a tibble with repeated values in one or more columns
2. Use `group_by` to select all of the categorical columns you want to combine to define your groups
3. Run `summarise` saying how you want to combine the quantitative values or `count` to see how many values are in each group
4. Run `ungroup` 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 or `COUNT` to see how many values are in each group
4. Run `ungroup` to remove any remaining group information



Grouping and Summarising

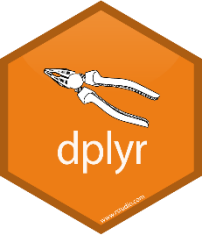
```
> group.data
```

```
# A tibble: 8 x 5
```

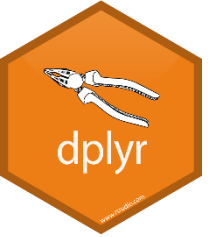
	Sample	Genotype	Sex	Height	Length
	<dbl>	<chr>	<chr>	<dbl>	<dbl>
1	1	WT	M	15	200
2	2	WT	F	13	185
3	3	WT	F	14	221
4	4	WT	M	18	265
5	5	KO	M	26	120
6	6	KO	F	22	165
7	7	KO	F	19	143
8	8	KO	M	27	110

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

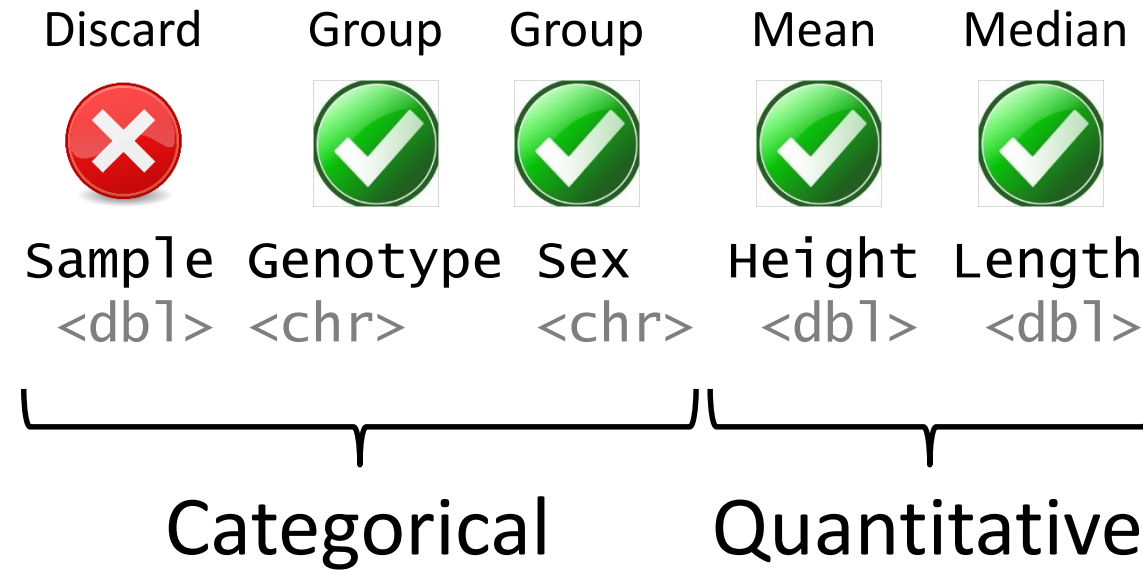
Grouping and Summarising Workflow



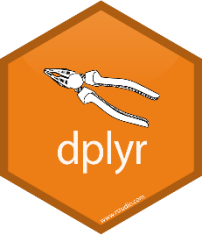
1. Load a tibble with repeated values in one or more columns
2. Use `group_by` to select all of the categorical columns you want to combine to define your groups
3. Run `summarise` saying how you want to combine the quantitative values or `COUNT` to see how many values are in each group
4. Run `ungroup` to remove any remaining group information



Grouping and Summarising



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



Grouping and Summarising

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

```
# A tibble: 8 x 5
```

```
# Groups:   Genotype, Sex [4]
```

	Sample <dbl>	Genotype <chr>	Sex <chr>	Height <dbl>	Length <dbl>
1	1	WT	M	15	200
4	4	WT	M	18	265
2	2	WT	F	13	185
3	3	WT	F	14	221
5	5	KO	M	26	120
8	8	KO	M	27	110
6	6	KO	F	22	165
7	7	KO	F	19	143

Discard



Sample
<dbl>

Group



Genotype
<chr>

Group



Sex
<chr>

Mean



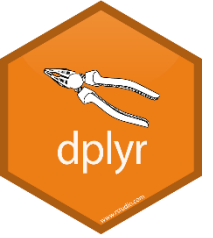
Height
<dbl>

Median

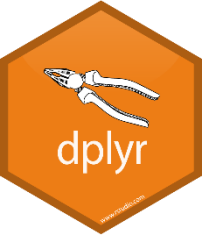


Length
<dbl>

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 or `count` to see how many values are in each group
4. Run `ungroup` to remove any remaining group information

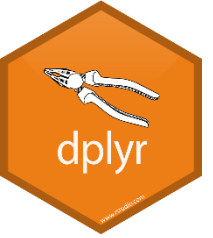


Grouping and Summarising

```
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
```

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



Grouping and Summarising

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

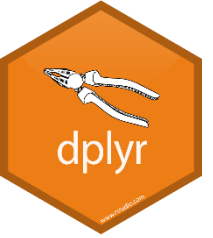
```
# A tibble: 4 x 4  
# Groups:   Genotype [2]  
  Genotype Sex Height2 Length  
  <chr>    <chr>   <dbl> <dbl>  
1 KO      F      20.5  154  
2 KO      M      26.5  115  
3 WT      F      13.5  203  
4 WT      M      16.5  232.
```

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

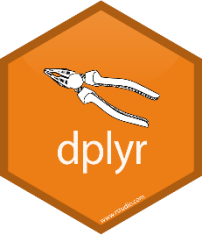


If you want the count of values as part of a summarised result use the n() function

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

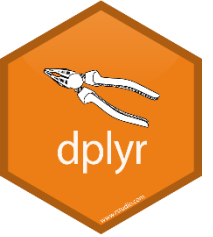


Ungrouping

- A summarise operation removes the 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

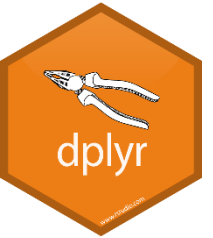


```
group.data %>%  
  arrange(desc(Height)) %>%  
  group_by(Sex) %>%  
  slice(1)
```

```
# 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
```

Grouping affects lots of operations

Normalise by mean centring the Length values



```
group.data %>%  
  mutate(Diff=Length - mean(Length))
```

```
# A tibble: 8 x 6  
  Sample Genotype Sex Height Length Diff  
  <dbl> <chr> <chr> <dbl> <dbl> <dbl>  
1     1 WT M 15 200 23.9  
2     2 WT F 13 185 8.88  
3     3 WT F 14 221 44.9  
4     4 WT M 18 265 88.9  
5     5 KO M 26 120 -56.1  
6     6 KO F 22 165 -11.1  
7     7 KO F 19 143 -33.1  
8     8 KO M 27 110 -66.1
```

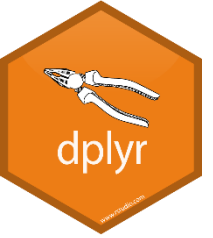
```
group.data %>%  
  group_by(Genotype) %>%  
  mutate(Diff=Length - mean(Length))
```

```
# A tibble: 8 x 6  
# Groups:   Genotype [2]  
  Sample Genotype Sex Height Length Diff  
  <dbl> <chr> <chr> <dbl> <dbl> <dbl>  
1     1 WT M 15 200 -17.8  
2     2 WT F 13 185 -32.8  
3     3 WT F 14 221 3.25  
4     4 WT M 18 265 47.2  
5     5 KO M 26 120 -14.5  
6     6 KO F 22 165 30.5  
7     7 KO F 19 143 8.5  
8     8 KO M 27 110 -24.5
```

Exercise 5

Mutating, Grouping and Summarising

Extending and Joining



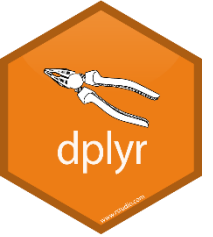
Extending tibbles

- `add_row` adds a single row
- `bind_rows` join tibbles by row
- `add_column` adds a column
- `bind_cols` join tibbles by column
- `rename` rename a column

```
trumpton %>%  
  add_row(  
    FirstName="Simon",  
    LastName="Andrews",  
    Age=39,  
    weight=80,  
    Height=185  
  )
```

```
trumpton %>%  
  add_column(  
    vegetarian = c(T,F,F,T,F,F,T)  
  )
```

```
trumpton %>%  
  rename(Surname=LastName)
```

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
  name  count
1 Simon     3
2 Laura     6
3 Felix     2
```

```
> join2
  name  percentage
1 Felix     10
2 Anne     25
3 Simon     36
```

left_join(join1,join2)

	name	count	percentage
1	Simon	3	36
2	Laura	6	NA
3	Felix	2	10

right_join(join1,join2)

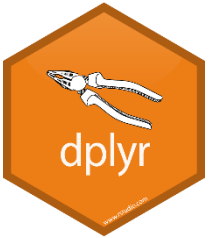
	name	count	percentage
1	Felix	2	10
2	Anne	NA	25
3	Simon	3	36

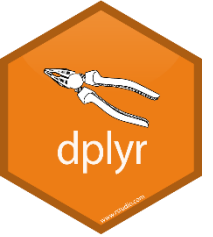
inner_join(join1,join2)

	name	count	percentage
1	Simon	3	36
2	Felix	2	10

full_join(join1,join2)

	name	count	percentage
1	Simon	3	36
2	Laura	6	NA
3	Felix	2	10
4	Anne	NA	25



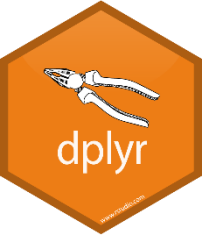


Rejoining split tables

Find the highest value for each genotype

```
> 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
```

```
> 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) %>%  
  ungroup()
```

```
# A tibble: 2 x 4  
  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) %>%  
  ungroup() %>%  
  left_join(gathered.annotation)
```

```
# A tibble: 2 x 7  
  Gene  genotype replicate value  Chr  Start  End  
  <chr> <chr>         <int> <dbl> <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

Custom Functions

Function name

Function Arguments

```
bmi <- function(weight, height) {  
  height/100 -> height  
  height^2 -> height  
  return(weight/height)  
}
```

Code Block

Return value

```
> bmi(90, 175)  
[1] 29.38776
```

```
> bmi(c(90, 102), c(175, 183))  
[1] 29.38776 30.45776
```

Custom function with mutate

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

```
trumpton %>%  
  mutate(bmi=calc_bmi(weight, Height))
```

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


Custom Functions with Tidyverse

```
summarise.gene <- function(tbl, genename="NANOG") {  
  tbl %>%  
    filter(GENE==genename) %>%  
    filter(str_length(REF) == 1, str_length(ALT) == 1) %>%  
    group_by(REF, ALT) %>%  
    count() %>%  
    ungroup() %>%  
    return()  
}  
  
child %>%  
  summarise.gene("PLEC")
```

```
# A tibble: 6 x 3  
  REF     ALT     n  
  <chr> <chr> <int>  
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 Tidyverse

```
count_mutations <- function(tbl, col=REF) {  
  tbl %>%  
    group_by({{ col }}) %>%  
    count() %>%  
    ungroup() %>%  
    arrange(desc(n)) %>%  
    return()  
}
```

```
child %>%  
  count_mutations(col=ALT)
```

```
# A tibble: 257 x 2  
  ALT      n  
  <chr> <int>  
1 C      6480  
2 G      6404  
3 A      6275  
4 T      6103  
5 GA       43  
6 TA       37  
7 GC       33  
8 AG       24  
9 CT       22  
10 CA      20  
# ... with 247 more rows
```

Exercise 7

Custom Functions