Imported into R

```
galtonHeights
```

```
  Family_Id Father Mother     RankedSons
RankedDaughters
1         1   18.5    7.0       13.2, ,    9.2,
9.0, 9.0
2         2   15.5    6.5   13.5, 12.5,       5.5,
5.5,
3         3   15.0    4.0       11.0, ,
8.0, ,
4         4   15.0    4.0    10.5, 8.5,    7.0,
```

```
4.5, 3.0
5         5   15.0   -1.5 12.0, 9.0, 8.0   6.5,
2.5, 2.5
```

Transforming from one format to the other

- reshape2 package
- tidyr package:
  - gather() - multiple columns to key-value pairs (“short->long”)
  - spread() - key-value pair columns to multiple columns (“long->short”)
  - separate() - split single column
  - unite() - join columns together

Separate

Use separate() when you have columns that have more than one variable in each cell.

```
separatedHeights =
  galtonHeights %>%
  separate(RankedSons, into = paste("Son", seq(1,3),
sep="_"), ",") %>%
  separate(RankedDaughters, into = paste("Daughter",
seq(1,3), sep="_"), ",")
separatedHeights
```

```
  Family_Id Father Mother Son_1 Son_2 Son_3 Daughter_1
Daughter_2
1         1   18.5    7.0  13.2                   9.2
9.0
2         2   15.5    6.5  13.5  12.5             5.5
```

```
5.5
3         3   15.0    4.0  11.0                  8.0
4         4   15.0    4.0  10.5   8.5            7.0
4.5
5         5   15.0   -1.5  12.0   9.0   8.0      6.5
2.5
  Daughter_3
1        9.0
2
3
4        3.0
5        2.5
```

Gather

Use gather() when you have columns that aren't variables.

```
gatheredHeights = separatedHeights %>%
  gather(Relation, ChildHeight, Son_1:Daughter_3) %>%
  separate(Relation, into = c("Relation",
"GenderRank"), "_") %>%
  select(-GenderRank)
gatheredHeights
```

```
   Family_Id Father Mother Relation ChildHeight
1          1   18.5    7.0      Son        13.2
2          2   15.5    6.5      Son        13.5
3          3   15.0    4.0      Son        11.0
4          4   15.0    4.0      Son        10.5
5          5   15.0   -1.5      Son        12.0
6          1   18.5    7.0      Son
7          2   15.5    6.5      Son        12.5
8          3   15.0    4.0      Son
9          4   15.0    4.0      Son         8.5
10         5   15.0   -1.5      Son         9.0
11         1   18.5    7.0      Son
12         2   15.5    6.5      Son
13         3   15.0    4.0      Son
14         4   15.0    4.0      Son
15         5   15.0   -1.5      Son         8.0
16         1   18.5    7.0 Daughter         9.2
17         2   15.5    6.5 Daughter         5.5
18         3   15.0    4.0 Daughter         8.0
19         4   15.0    4.0 Daughter         7.0
20         5   15.0   -1.5 Daughter         6.5
21         1   18.5    7.0 Daughter         9.0
22         2   15.5    6.5 Daughter         5.5
23         3   15.0    4.0 Daughter
24         4   15.0    4.0 Daughter         4.5
25         5   15.0   -1.5 Daughter         2.5
26         1   18.5    7.0 Daughter         9.0
27         2   15.5    6.5 Daughter
28         3   15.0    4.0 Daughter
29         4   15.0    4.0 Daughter         3.0
30         5   15.0   -1.5 Daughter         2.5
```

Cleanup time!

Cleaning data is often easier with long data.

```
getHeight = function(x) { as.numeric(x) + 60 }
cleanedHeights = gatheredHeights %>%
  mutate(ChildGender = ifelse(Relation == "Son", "M",
"F"),
         Father = getHeight(Father),
         Mother = getHeight(Mother),
         ChildHeight = getHeight(ChildHeight)) %>%
  filter(!is.na(ChildHeight)) %>%
  group_by(Family_Id) %>%
  mutate(Rank = as.integer(rank(desc(ChildHeight),
ties.method = "first"))) %>%
  ungroup() %>%
  select(-Relation)
cleanedHeights
```

```
Source: local data frame [21 x 6]

   Family_Id Father Mother ChildHeight ChildGender
Rank
1          1   78.5   67.0        73.2           M
1
2          2   75.5   66.5        73.5           M
1
3          3   75.0   64.0        71.0           M
1
4          4   75.0   64.0        70.5           M
1
5          5   75.0   58.5        72.0           M
1
6          2   75.5   66.5        72.5           M
2
7          4   75.0   64.0        68.5           M
2
8          5   75.0   58.5        69.0           M
```

```
8          5   75.0   58.5        69.0           M
2
9          5   75.0   58.5        68.0           M
3
10         1   78.5   67.0        69.2           F
2
11         2   75.5   66.5        65.5           F
3
12         3   75.0   64.0        68.0           F
2
13         4   75.0   64.0        67.0           F
3
14         5   75.0   58.5        66.5           F
4
15         1   78.5   67.0        69.0           F
3
16         2   75.5   66.5        65.5           F
4
17         4   75.0   64.0        64.5           F
4
18         5   75.0   58.5        62.5           F
5
19         1   78.5   67.0        69.0           F
4
20         4   75.0   64.0        63.0           F
5
21         5   75.0   58.5        62.5           F
6
```

Spread

Use Spread() when you have a column with different variables in different rows.

```
spreadHeights = cleanedHeights %>%
  spread(Rank, ChildHeight)
spreadHeights
```

```
Source: local data frame [10 x 10]

   Family_Id Father Mother ChildGender    1    2    3
4    5    6
1          1   78.5   67.0           F   NA 69.2 69.0
69.0   NA   NA
2          1   78.5   67.0           M 73.2   NA   NA
NA   NA   NA
3          2   75.5   66.5           F   NA   NA 65.5
65.5   NA   NA
4          2   75.5   66.5           M 73.5 72.5   NA
NA   NA   NA
5          3   75.0   64.0           F   NA 68.0   NA
NA   NA   NA
6          3   75.0   64.0           M 71.0   NA   NA
```

```
NA   NA   NA
7           4   75.0   64.0           F   NA   NA 67.0
64.5 63.0   NA
8           4   75.0   64.0           M 70.5 68.5   NA
NA   NA   NA
9           5   75.0   58.5           F   NA   NA   NA
66.5 62.5 62.5
10          5   75.0   58.5           M 72.0 69.0 68.0
NA   NA   NA
```

Unite

Use Unite() when you have a column that isn't a complete variable.

```
unitedHeights = spreadHeights %>%
  unite(ChildHeights, 5:10, sep=",")
unitedHeights
```

```
Source: local data frame [10 x 5]

   Family_Id Father Mother ChildGender
ChildHeights
1           1   78.5   67.0           F
NA,69.2,69,69,NA,NA
2           1   78.5   67.0           M
73.2,NA,NA,NA,NA,NA
3           2   75.5   66.5           F
NA,NA,65.5,65.5,NA,NA
4           2   75.5   66.5           M
73.5,72.5,NA,NA,NA,NA
5           3   75.0   64.0           F
NA,68,NA,NA,NA,NA
6           3   75.0   64.0           M
71,NA,NA,NA,NA,NA
7           4   75.0   64.0           F
NA,NA,67,64.5,63,NA
8           4   75.0   64.0           M
70.5,68.5,NA,NA,NA,NA
9           5   75.0   58.5           F
NA,NA,NA,66.5,62.5,62.5
10          5   75.0   58.5           M
72,69,68,NA,NA,NA
```

A few words about (de)Normalization

- Benefits for both wide and narrow formats.
- Easy to convert between them.
- Consider your “base” format
- Make life easy on yourself

A word about (de)Normalization

- What if we added to our data? such as
  - Name
  - Age
  - Weight
- This would be stored redundantly for parents of multiple children
- We could move data on individuals to a separate table

Normalized Tables

- People(Id, Weight, Sex)
- Heredity(MaleId, FemaleId, ChildId)

One last tip: Renaming Columns

A common task with an easy solution.

```
# usually something like this:
names(heights)[[2]] = "Dad"; # not
particularly robust
```

```
# or this:
heights$Dad = heights$Father # not
particularly efficient
heights$Father = NULL
```

```
Error in rename_(.data, .dots =
lazyeval::lazy_dots(...)) :
  object 'heights' not found
```