# Stat7350-Session4_Tidy-Data

*Aleeza Gerstein*

*2019-03-13*

## Learning Objectives

### Part I: Transform Data

- Extract variables with `select()`
- Extract cases with `filter()`
- Arrange cases, with `arrange()`
- Make tables of summaries with `summarise()`
- Make new variables, with `mutate()`
- Do groupwise operations with `group_by()`
- Use `_join()` to join and filter datasets

### Part II: Tidy Data

- Reshape data so that it is tidy

## Pre-analysis workflow: packages & data prep

### Packages

```
suppressMessages(library(tidyverse))
suppressMessages(library(gridExtra))
suppressMessages(library(Hmisc)) #%nin%
suppressMessages(library(skimr))
```

#### Load Data for Part I

We're going to use a dataset that is built-in to the tidyverse, the iris dataset. We'll also load our old survey dataset to use for practice.

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
iris_df <- as_tibble(iris)
sc <- read_csv("data_output/surveys_complete.csv")
```

```
## Parsed with column specification:
## cols(
##   record_id = col_double(),
##   month = col_double(),
##   day = col_double(),
##   year = col_double(),
##   plot_id = col_double(),
##   species_id = col_character(),
##   sex = col_character(),
##   hindfoot_length = col_double(),
##   weight = col_double(),
##   genus = col_character(),
##   species = col_character(),
##   taxa = col_character(),
##   plot_type = col_character()
## )
```

It has 150 rows. Let's add a new vector with row number.

```
row <- 1:nrow(iris_df)
bind_cols(iris_df, row = row)
```

```
## # A tibble: 150 x 6
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species   row
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>   <int>
##  1          5.1         3.5          1.4         0.2 setosa      1
##  2          4.9         3            1.4         0.2 setosa      2
##  3          4.7         3.2          1.3         0.2 setosa      3
##  4          4.6         3.1          1.5         0.2 setosa      4
##  5          5           3.6          1.4         0.2 setosa      5
##  6          5.4         3.9          1.7         0.4 setosa      6
##  7          4.6         3.4          1.4         0.3 setosa      7
##  8          5           3.4          1.5         0.2 setosa      8
##  9          4.4         2.9          1.4         0.2 setosa      9
## 10          4.9         3.1          1.5         0.1 setosa     10
## # ... with 140 more rows
```

## Part I: Transform Data

### Keep a subset of columns

Using base R:

```
iris_df[iris_df[, "Sepal.Length"] > 7.5 & iris_df[, "Species"]=="virginica", ]
```

```
## # A tibble: 6 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          7.6         3            6.6         2.1 virginica
## 2          7.7         3.8          6.7         2.2 virginica
## 3          7.7         2.6          6.9         2.3 virginica
## 4          7.7         2.8          6.7         2   virginica
## 5          7.9         3.8          6.4         2   virginica
## 6          7.7         3            6.1         2.3 virginica
```

With the tidyverse:

```
filter(iris_df, Sepal.Length > 7.5, Species=="virginica")
```

```
## # A tibble: 6 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          7.6         3            6.6         2.1 virginica
## 2          7.7         3.8          6.7         2.2 virginica
## 3          7.7         2.6          6.9         2.3 virginica
## 4          7.7         2.8          6.7         2   virginica
## 5          7.9         3.8          6.4         2   virginica
## 6          7.7         3            6.1         2.3 virginica
```

```
filter(iris_df, Sepal.Length > 7.5 | Sepal.Length < 5.5, Species=="virginica")
```

```
## # A tibble: 7 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          7.6         3            6.6         2.1 virginica
## 2          4.9         2.5          4.5         1.7 virginica
## 3          7.7         3.8          6.7         2.2 virginica
## 4          7.7         2.6          6.9         2.3 virginica
## 5          7.7         2.8          6.7         2   virginica
## 6          7.9         3.8          6.4         2   virginica
## 7          7.7         3            6.1         2.3 virginica
```

### Subset by rows

```
iris_df[1:2,]
```

```
## # A tibble: 2 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
```

```
slice(iris_df, 1:2)
```

```
## # A tibble: 2 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
```

### Select specific columns

```
select(iris_df, Species, Petal.Length, Sepal.Length)
```

```
## # A tibble: 150 x 3
##    Species Petal.Length Sepal.Length
##    <fct>          <dbl>        <dbl>
##  1 setosa           1.4          5.1
##  2 setosa           1.4          4.9
##  3 setosa           1.3          4.7
##  4 setosa           1.5          4.6
##  5 setosa           1.4          5
```

```
##  6 setosa           1.7          5.4
##  7 setosa           1.4          4.6
##  8 setosa           1.5          5
##  9 setosa           1.4          4.4
## 10 setosa           1.5          4.9
## # ... with 140 more rows
```

```
select(iris_df, Sepal.Length : Petal.Width)
```

```
## # A tibble: 150 x 4
##    Sepal.Length Sepal.Width Petal.Length Petal.Width
##           <dbl>       <dbl>        <dbl>       <dbl>
##  1          5.1         3.5          1.4         0.2
##  2          4.9         3            1.4         0.2
##  3          4.7         3.2          1.3         0.2
##  4          4.6         3.1          1.5         0.2
##  5          5           3.6          1.4         0.2
##  6          5.4         3.9          1.7         0.4
##  7          4.6         3.4          1.4         0.3
##  8          5           3.4          1.5         0.2
##  9          4.4         2.9          1.4         0.2
## 10          4.9         3.1          1.5         0.1
## # ... with 140 more rows
```

```
select(iris_df, starts_with("S"))
```

```
## # A tibble: 150 x 3
##    Sepal.Length Sepal.Width Species
##           <dbl>       <dbl> <fct>
##  1          5.1         3.5 setosa
##  2          4.9         3   setosa
##  3          4.7         3.2 setosa
##  4          4.6         3.1 setosa
##  5          5           3.6 setosa
##  6          5.4         3.9 setosa
##  7          4.6         3.4 setosa
##  8          5           3.4 setosa
##  9          4.4         2.9 setosa
## 10          4.9         3.1 setosa
## # ... with 140 more rows
```

### Drop specific columns

```
select(iris_df, -(Sepal.Length : Petal.Width))
```

```
## # A tibble: 150 x 1
##    Species
##    <fct>
##  1 setosa
##  2 setosa
##  3 setosa
##  4 setosa
##  5 setosa
##  6 setosa
##  7 setosa
##  8 setosa
##  9 setosa
```

```
## 10 setosa
## # ... with 140 more rows
```

**Ordering with arrange (row-wise ordering based on specific columns)**

```
arrange(iris_df, Species, Sepal.Length, Sepal.Width)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.3         3            1.1         0.1 setosa
##  2          4.4         2.9          1.4         0.2 setosa
##  3          4.4         3            1.3         0.2 setosa
##  4          4.4         3.2          1.3         0.2 setosa
##  5          4.5         2.3          1.3         0.3 setosa
##  6          4.6         3.1          1.5         0.2 setosa
##  7          4.6         3.2          1.4         0.2 setosa
##  8          4.6         3.4          1.4         0.3 setosa
##  9          4.6         3.6          1           0.2 setosa
## 10          4.7         3.2          1.3         0.2 setosa
## # ... with 140 more rows
```

```
arrange(iris_df, desc(Species), Sepal.Length, Sepal.Width)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.9         2.5          4.5         1.7 virginica
##  2          5.6         2.8          4.9         2   virginica
##  3          5.7         2.5          5           2   virginica
##  4          5.8         2.7          5.1         1.9 virginica
##  5          5.8         2.7          5.1         1.9 virginica
##  6          5.8         2.8          5.1         2.4 virginica
##  7          5.9         3            5.1         1.8 virginica
##  8          6           2.2          5           1.5 virginica
##  9          6           3            4.8         1.8 virginica
## 10          6.1         2.6          5.6         1.4 virginica
## # ... with 140 more rows
```

Base R code equivalent

```
iris_df[order(iris_df$Species, iris_df$Sepal.Length, iris_df$Sepal.Width), ]
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.3         3            1.1         0.1 setosa
##  2          4.4         2.9          1.4         0.2 setosa
##  3          4.4         3            1.3         0.2 setosa
##  4          4.4         3.2          1.3         0.2 setosa
##  5          4.5         2.3          1.3         0.3 setosa
##  6          4.6         3.1          1.5         0.2 setosa
##  7          4.6         3.2          1.4         0.2 setosa
##  8          4.6         3.4          1.4         0.3 setosa
##  9          4.6         3.6          1           0.2 setosa
## 10          4.7         3.2          1.3         0.2 setosa
## # ... with 140 more rows
```

---

**CHALLENGE:**

With the `sc` dataset, filter all of the individuals from the `genus` is Dipodomys and `hindfoot_length` equal to or less than 35.

---

All of this can also be accomplished using the pipe. e.g.,

```
iris_df %>%
  select(3:5) %>%
  filter(Petal.Length < 2) %>%
  arrange(Petal.Width)
```

```
## # A tibble: 50 x 3
##    Petal.Length Petal.Width Species
##           <dbl>       <dbl> <fct>
##  1          1.5         0.1 setosa
##  2          1.4         0.1 setosa
##  3          1.1         0.1 setosa
##  4          1.5         0.1 setosa
##  5          1.4         0.1 setosa
##  6          1.4         0.2 setosa
##  7          1.4         0.2 setosa
##  8          1.3         0.2 setosa
##  9          1.5         0.2 setosa
## 10          1.4         0.2 setosa
## # ... with 40 more rows
```

---

**CHALLENGE: Use %>% to write a sequence of functions that:**

1. Filter `sc` to keep just the male records from 1980
2. Select the `month`, `day`, 'species' and 'hindfoot_length', columns
3. Arrange the results so that they are organized by calendar date

---

**Rename columns**

```
rename(iris_df, new_col_name = Species)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width new_col_name
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
```

```
## # ... with 140 more rows
```

### Obtain unique rows with distinct

```
distinct(iris_df, Species, .keep_all=TRUE)  keep all remaining variables (other than species)
```

```
## # A tibble: 3 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          7           3.2          4.7         1.4 versicolor
## 3          6.3         3.3          6           2.5 virginica
```

### Add columns with mutate

```
mutate(iris_df, Ratio = Sepal.Length / Sepal.Width, Sum = Sepal.Length + Sepal.Width)
```

```
## # A tibble: 150 x 7
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species Ratio   Sum
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>   <dbl> <dbl>
##  1          5.1         3.5          1.4         0.2 setosa   1.46   8.6
##  2          4.9         3            1.4         0.2 setosa   1.63   7.9
##  3          4.7         3.2          1.3         0.2 setosa   1.47   7.9
##  4          4.6         3.1          1.5         0.2 setosa   1.48   7.7
##  5          5           3.6          1.4         0.2 setosa   1.39   8.6
##  6          5.4         3.9          1.7         0.4 setosa   1.38   9.3
##  7          4.6         3.4          1.4         0.3 setosa   1.35   8
##  8          5           3.4          1.5         0.2 setosa   1.47   8.4
##  9          4.4         2.9          1.4         0.2 setosa   1.52   7.3
## 10          4.9         3.1          1.5         0.1 setosa   1.58   8
## # ... with 140 more rows
```

### Transmute

This does the same thing as `mutate` but drops existing columns.

```
transmute(iris_df, Ratio = Sepal.Length / Sepal.Width, Sum = Sepal.Length + Sepal.Width)
```

```
## # A tibble: 150 x 2
##    Ratio   Sum
##    <dbl> <dbl>
##  1  1.46   8.6
##  2  1.63   7.9
##  3  1.47   7.9
##  4  1.48   7.7
##  5  1.39   8.6
##  6  1.38   9.3
##  7  1.35   8
##  8  1.47   8.4
##  9  1.52   7.3
## 10  1.58   8
## # ... with 140 more rows
```

### Summarize and group by

```
summarise(iris_df, mean(Petal.Length))
```

```
## # A tibble: 1 x 1
##   `mean(Petal.Length)`
##                  <dbl>
## 1                 3.76
```

Summarize on many columns

```
summarise_all(iris_df[,1:4], mean)
```

```
## # A tibble: 1 x 4
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
##          <dbl>       <dbl>        <dbl>       <dbl>
## 1         5.84        3.06         3.76        1.20
```

Summarize by grouping column

```
iris_df %>%
  group_by(Species) %>%
  summarise(mean = mean(Sepal.Length))
```

```
## # A tibble: 3 x 2
##   Species     mean
##   <fct>      <dbl>
## 1 setosa      5.01
## 2 versicolor  5.94
## 3 virginica   6.59
```

Note you can also use dplyr without the pipe and with a mix of base formatting

```
summarise(group_by(iris_df, Species), mean(Sepal.Length))
```

```
## # A tibble: 3 x 2
##   Species    `mean(Sepal.Length)`
##   <fct>                     <dbl>
## 1 setosa                     5.01
## 2 versicolor                 5.94
## 3 virginica                  6.59
```

Summarise all columns

```
iris_df %>%
  group_by(Species) %>%
  summarise_all(mean)
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

```
sc %>%
  filter(weight < 150) %>%
  summarise(mean = mean(weight), n = n())
```

```
## # A tibble: 1 x 2
##    mean     n
##   <dbl> <int>
```

```
## 1  38.2 29684
```

```
sc %>%
  group_by(weight < 150) %>%
  summarise(mean_weight = mean(weight), n = n())
```

```
## # A tibble: 2 x 3
##   `weight < 150` mean_weight     n
##   <lgl>                <dbl> <int>
## 1 FALSE                181.    779
## 2 TRUE                  38.2 29684
```

---

**CHALLENGE:**

For `sc` divide the dataset into records with `weight` less than 150 and higher than 150 and then use `summarise` to find out the mean of each group and the number of records in each group

---

### Merging tibbles

There are several join functions for merging tibbles by a common key column similar to the merge function in base R. These `_join()?` functions include:

- `inner_join()`: returns join only for rows matching among both tibbles
- `full_join()`: returns join for all (matching and non-matching) rows of two tibbles
- `left_join()`: returns join for all rows in first tibble
- `right_join()`: returns join for all rows in second tibble
- `anti_join()`: returns for first tibble only those rows that have no match in the second one

Toy data:

```
band <- tribble(
   ~name,     ~band,
  "Mick",  "Stones",
  "John", "Beatles",
  "Paul", "Beatles"
)

instrument <- tribble(
    ~name,   ~plays,
   "John", "guitar",
   "Paul",   "bass",
  "Keith", "guitar"
)

instrument2 <- tribble(
    ~artist,   ~plays,
   "John", "guitar",
   "Paul",   "bass",
  "Keith", "guitar"
)
```

### Mutating joins

```
band %>% left_join(instrument, by = "name")
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
```

‘‘‘

```
band %>% right_join(instrument, by = "name")
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
## 3 Keith <NA>    guitar
```

```
band %>% full_join(instrument, by = "name")
```

```
## # A tibble: 4 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
## 4 Keith <NA>    guitar
```

```
band %>% inner_join(instrument, by = "name")
```

```
## # A tibble: 2 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
```

## Load Data for Part II

### Long versus wide

Long data formats have one observation and one measurement per row. So, multiple rows constitude a single observation.

Wide data has a every measurement in a single observation in a single row.

### Long to wide

Data is often recorded in a long format for efficiency. We're going to use a toy mammal dataset to illustrate. For every site a researcher visited, they wrote down the species of mammals they saw, and the density of those mammals. As this was just a running tally, the data ended up in a long format:

```
mammals <- tibble(site = c(1,1,2,3,3,3),
                  taxon = c('Suncus etruscus', 'Sorex cinereus',
                            'Myotis nigricans', 'Notiosorex crawfordi',
                            'Scuncus etruscus', 'Myotis nigricans'),
```

```
                    density = c(6.2, 5.2, 11.0, 1.2, 9.4, 9.6)
)
mammals
```

```
## # A tibble: 6 x 3
##    site taxon                density
##   <dbl> <chr>                  <dbl>
## 1     1 Suncus etruscus          6.2
## 2     1 Sorex cinereus           5.2
## 3     2 Myotis nigricans        11
## 4     3 Notiosorex crawfordi     1.2
## 5     3 Scuncus etruscus         9.4
## 6     3 Myotis nigricans         9.6
```

But, what if we wanted to easily compare abundances across sites? We can use `spread` to change the data into a wide format.

`Spread` requires you tell it the name of the column which contains values that will be the column names in your new data set - a so called "key" column. You also tell it which column contains the relevant numbers - the "values" column.

```
m_wide <- mammals %>%
  spread(taxon, density)

m_wide
```

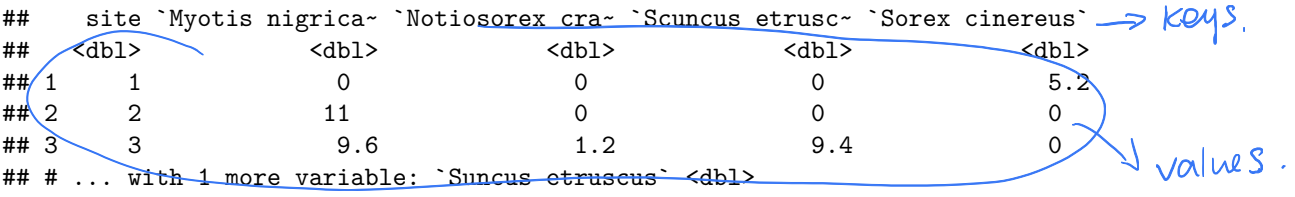
```
## # A tibble: 3 x 6
##    site `Myotis nigrica~ `Notiosorex cra~ `Scuncus etrusc~ `Sorex cinereus`
##   <dbl>            <dbl>            <dbl>            <dbl>            <dbl>
## 1     1               NA               NA               NA              5.2
## 2     2               11               NA               NA               NA
## 3     3              9.6              1.2              9.4               NA
## # ... with 1 more variable: `Suncus etruscus` <dbl>
```

You'll notice there are a lot of NA values. This is incredibly common in ecological and biological data. Sometimes, they are NA - they weren't recorded Other times, such as in this data set, they actually mean 0 observations. To add this information we use the `fill` argument in `spread`.

```
m_wide_0 <- mammals %>%
  spread(taxon, density, fill=0)

m_wide_0
```

```
## # A tibble: 3 x 6
##    site `Myotis nigrica~ `Notiosorex cra~ `Scuncus etrusc~ `Sorex cinereus`
##   <dbl>            <dbl>            <dbl>            <dbl>            <dbl>
## 1     1               0                0                0              5.2
## 2     2              11                0                0              0
## 3     3              9.6               1.2              9.4            0
## # ... with 1 more variable: `Suncus etruscus` <dbl>
```




Note that we could also fill those in right into the long data format using the `complete` function. In that function, we specify which columns we want all combinations of, and then supply a list of how new values should be filled in for other columns. If we don't give a column name in that list, it defaults to NA.

```
m_long_0 <- mammals %>%
  complete(site, taxon, fill=list(density=0))

m_long_0
```

```
## # A tibble: 15 x 3
```

```
##     site taxon                density
##    <dbl> <chr>                  <dbl>
##  1     1 Myotis nigricans         0
##  2     1 Notiosorex crawfordi     0
##  3     1 Scuncus etruscus         0
##  4     1 Sorex cinereus           5.2
##  5     1 Suncus etruscus          6.2
##  6     2 Myotis nigricans        11
##  7     2 Notiosorex crawfordi     0
##  8     2 Scuncus etruscus         0
##  9     2 Sorex cinereus           0
## 10     2 Suncus etruscus          0
## 11     3 Myotis nigricans         9.6
## 12     3 Notiosorex crawfordi     1.2
## 13     3 Scuncus etruscus         9.4
## 14     3 Sorex cinereus           0
## 15     3 Suncus etruscus          0
```

### Wide to long

In other cases the reverse is true - as some people recorde their data in a wide format. To go from wide to long we use the `gather` function, which "gathers up your wide data". It's a little bit , as you specify what you want the name of the new key column to be, what you want the name of the new values column to be, and then you can either specify which columns are to be gathered up (which can be tedious if there are a lot or they are spread) or you can specify which columns you want to exclude. I actually do things by exclusion quite often (as in `%nin%`).

```
m_long <- m_wide_0 %>%
  gather(Species_name, Density, -site)

m_long
```

```
## # A tibble: 15 x 3
##     site Species_name         Density
##    <dbl> <chr>                  <dbl>
##  1     1 Myotis nigricans         0
##  2     2 Myotis nigricans        11
##  3     3 Myotis nigricans         9.6
##  4     1 Notiosorex crawfordi     0
##  5     2 Notiosorex crawfordi     0
##  6     3 Notiosorex crawfordi     1.2
##  7     1 Scuncus etruscus         0
##  8     2 Scuncus etruscus         0
##  9     3 Scuncus etruscus         9.4
## 10     1 Sorex cinereus           5.2
## 11     2 Sorex cinereus           0
## 12     3 Sorex cinereus           0
## 13     1 Suncus etruscus          6.2
## 14     2 Suncus etruscus          0
## 15     3 Suncus etruscus          0
```

```
m_long <- m_wide_0 %>%
  gather(Species_name, Density, `Myotis nigricans`:`Suncus etruscus`)
```

### What if we have more than one value column

This gets tricky. We have to `unite` those columns into something that's later easy to separate. Here's an example. Let's create a new variable for height first.

```
mamh <- mammals %>%
  mutate(height=rnorm(6,30,3))
```

Now we want to do everything above, but with density and height. First, we unite them, using a _ as our separator. Note that we We could have used anything, but _ is often used because it's easy to see andd used for so few other things.

```
mamh2 <- mamh %>%
  unite(measurement, density, height, sep="_")

mamh2
```

① first unit the together

```
## # A tibble: 6 x 3
##    site taxon                measurement
##   <dbl> <chr>                <chr>
## 1     1 Suncus etruscus      6.2_28.6281726196867
## 2     1 Sorex cinereus       5.2_27.5337083045981
## 3     2 Myotis nigricans     11_31.3776017258493
## 4     3 Notiosorex crawfordi 1.2_29.9302517897248
## 5     3 Scuncus etruscus     9.4_28.6092269175507
## 6     3 Myotis nigricans     9.6_31.4090770082351
```

So now we keep going. Let's first fill iln all the empty species-site combos with zeroes. Here's one workflow to do that.

```
mamh_long <- mamh2 %>%
  spread(taxon, measurement, fill="0_0") %>%
  gather(taxon, measurement, -site)

mamh_long
```

```
## # A tibble: 15 x 3
##     site taxon                measurement
##    <dbl> <chr>                <chr>
##  1     1 Myotis nigricans     0_0
##  2     2 Myotis nigricans     11_31.3776017258493
##  3     3 Myotis nigricans     9.6_31.4090770082351
##  4     1 Notiosorex crawfordi 0_0
##  5     2 Notiosorex crawfordi 0_0
##  6     3 Notiosorex crawfordi 1.2_29.9302517897248
##  7     1 Scuncus etruscus     0_0
##  8     2 Scuncus etruscus     0_0
##  9     3 Scuncus etruscus     9.4_28.6092269175507
## 10     1 Sorex cinereus       5.2_27.5337083045981
## 11     2 Sorex cinereus       0_0
## 12     3 Sorex cinereus       0_0
## 13     1 Suncus etruscus      6.2_28.6281726196867
## 14     2 Suncus etruscus      0_0
## 15     3 Suncus etruscus      0_0
```

Now we want to restore our old measurements. For that, we have **separate**, which takes the relevant column, the new column names in a vector, and the pattern you match to split them.

```
mamh_long <- mamh_long %>%
  separate(measurement, into = c("density", "height"), sep="_") %>%
  mutate(density = as.numeric(density),
         height = as.numeric(height))

mamh_long
```

```
## # A tibble: 15 x 4
##     site taxon                density height
##    <dbl> <chr>                  <dbl>  <dbl>
##  1     1 Myotis nigricans         0      0
##  2     2 Myotis nigricans        11     31.4
##  3     3 Myotis nigricans         9.6   31.4
##  4     1 Notiosorex crawfordi     0      0
##  5     2 Notiosorex crawfordi     0      0
##  6     3 Notiosorex crawfordi     1.2   29.9
##  7     1 Scuncus etruscus         0      0
##  8     2 Scuncus etruscus         0      0
##  9     3 Scuncus etruscus         9.4   28.6
## 10     1 Sorex cinereus           5.2   27.5
## 11     2 Sorex cinereus           0      0
## 12     3 Sorex cinereus           0      0
## 13     1 Suncus etruscus          6.2   28.6
## 14     2 Suncus etruscus          0      0
## 15     3 Suncus etruscus          0      0
```

Note that because he new column was treated as a charcter(we just did a string split) we have to coerce the columns back into numeric.

---

**CHALLENGE**

Separate the taxon into two columns - Genus and species

---