Week 6 Assignment

Ellen Bledsoe

2024-02-29

Week 6 Assignment

Assignment Exercises

Set-up

Load the packages we will need. You can either load all of them individually (`readr`, `dplyr`, `tidyr`, `ggplot2`) or load the `tidyverse` package.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

1. Forest Area per Country (15 pts)

These data are downloaded from the WHO and contain the amount of forest (sq. km) per country. The first 3 rows of the file are metadata or empty, which we do not want. I've added the arguments `skip = 4` and `col_names = TRUE` to the `read_csv` function to deal with this.

```
forest <- read_csv("forest_per_country.csv", skip = 4, col_names = TRUE)
```

```
## Rows: 266 Columns: 35
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): Country Name, Country Code
## dbl (32): 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, ...
## lgl  (1): 2022
##
## 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.
```

a. Currently, this data is in a wide format. We want to convert this to a longer format and make it tidy. Use the `pivot_longer` function to do so. Overwrite the `forest` dataframe so that it contains the long version of the data.

Because the column names start with numbers, which R does not like, we need to put the column names either in backticks or quotation marks (e.g., "1990":"2022").

```
forest <- forest %>%
  pivot_longer(`1990`:`2022`,
               names_to = "Year",
               values_to = "Area")

forest
```

```
## # A tibble: 8,778 x 4
##    `Country Name` `Country Code` Year   Area
##    <chr>          <chr>          <chr> <dbl>
##  1 Aruba          ABW            1990    4.2
##  2 Aruba          ABW            1991    4.2
##  3 Aruba          ABW            1992    4.2
##  4 Aruba          ABW            1993    4.2
##  5 Aruba          ABW            1994    4.2
##  6 Aruba          ABW            1995    4.2
##  7 Aruba          ABW            1996    4.2
##  8 Aruba          ABW            1997    4.2
##  9 Aruba          ABW            1998    4.2
## 10 Aruba          ABW            1999    4.2
## # i 8,768 more rows
```

b. Remove any rows that have `NA` in the forest area column using the `drop_na()` function.

```
forest <- forest %>%
  drop_na(Area)

forest
```

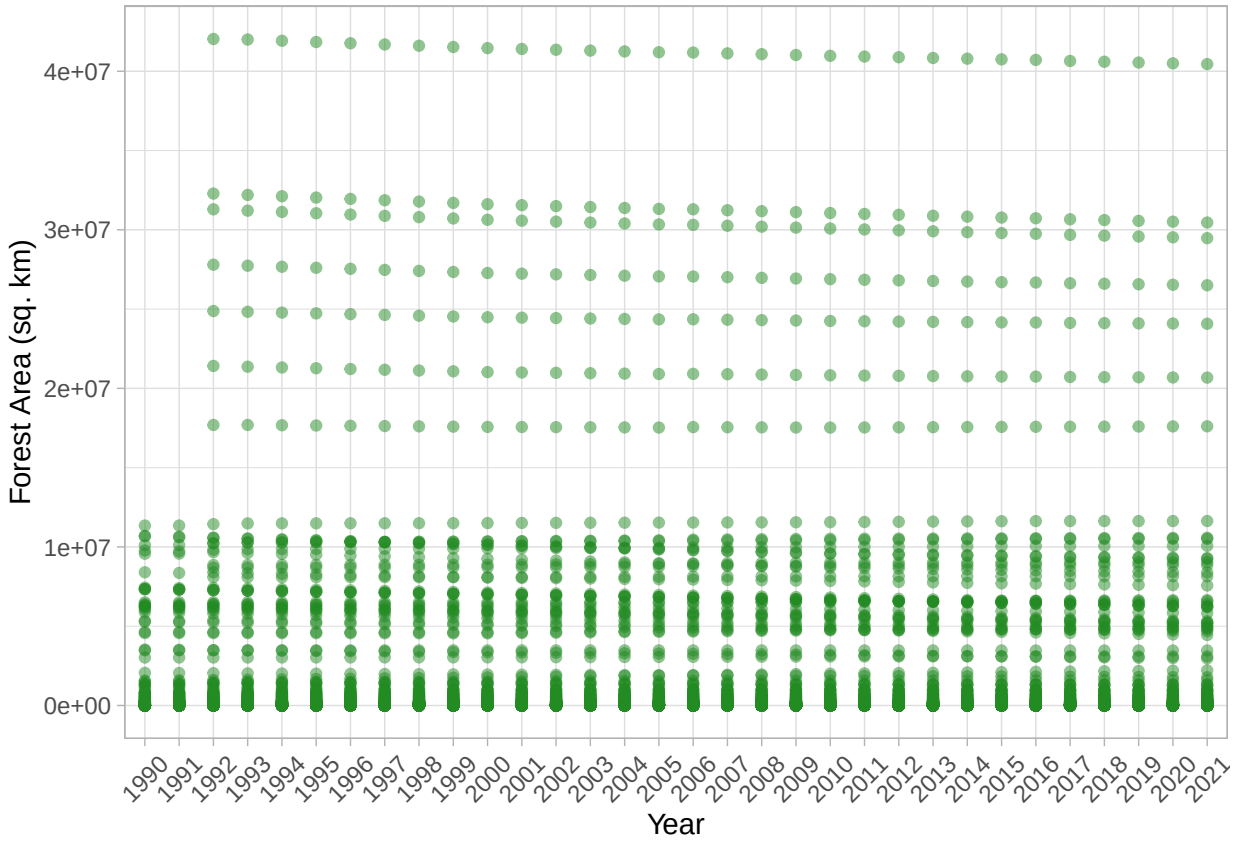
```
## # A tibble: 8,176 x 4
##    `Country Name` `Country Code` Year   Area
##    <chr>          <chr>          <chr> <dbl>
##  1 Aruba          ABW            1990    4.2
##  2 Aruba          ABW            1991    4.2
##  3 Aruba          ABW            1992    4.2
##  4 Aruba          ABW            1993    4.2
##  5 Aruba          ABW            1994    4.2
##  6 Aruba          ABW            1995    4.2
##  7 Aruba          ABW            1996    4.2
##  8 Aruba          ABW            1997    4.2
##  9 Aruba          ABW            1998    4.2
## 10 Aruba          ABW            1999    4.2
## # i 8,166 more rows
```

c. Let's remind ourselves how to plot the data. Make a scatterplot of the data with year on the x-axis and forest area on the y-axis. Make the points partially tranparent and the color "forestgreen." Add more descriptive axes labels and a theme.

Add the following line of code to the end of your ggplot code so we can see the years along the x-axis: `theme(axis.text.x = element_text(angle = 45, vjust = 0.5))`.

```
forest %>%
  ggplot(aes(x=Year, y=Area))+
  geom_point(aes(x=Year, y=Area), alpha = 0.5, color="forestgreen")+
  labs(x="Year", y="Forest Area (sq. km)")+
```

```
  theme_light()+
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```



2. OECD Data (10 pts)

We have some data from the Organisation for Economic Co-operation and Development (OECD) about various global fishing economies and sustainability. This dataset has the area of protected marine reserves.

Run this line of code to read in the file. Like the forest data, this data has a few rows of metadata at the top of the document that we need to skip.

```
oecd <- read_csv("oecd_annual_data.csv", skip = 4, col_names = TRUE)
```

```
## Rows: 127 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): OECD_member, Country
## dbl (23): 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, ...
##
## 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.
```

a. Use the `fill()` function to fill in the values in the first column.

```
oecd <- oecd %>%
  fill(OECD_member)

oecd
```

```
## # A tibble: 127 x 25
##    OECD_member Country   `2000` `2001` `2002` `2003` `2004` `2005` `2006` `2007`
##    <chr>       <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 OECD        Australia 3.77e5 3.77e5 4.00e5 4.00e5 4.02e5 4.06e5 4.12e5 4.17e5
##  2 OECD        Belgium   5.52e1 5.52e1 5.52e1 5.82e1 5.82e1 3.50e2 3.50e2 3.50e2
##  3 OECD        Canada    2.47e4 2.47e4 2.49e4 2.81e4 3.00e4 3.22e4 3.25e4 3.27e4
##  4 OECD        Chile     8.85e3 8.85e3 8.85e3 8.87e3 1.01e4 1.02e4 1.02e4 1.02e4
##  5 OECD        Colombia  2.94e4 2.94e4 2.94e4 2.94e4 2.94e4 6.09e4 6.09e4 6.09e4
##  6 OECD        Costa Ri~ 5.84e4 5.84e4 5.84e4 5.84e4 5.84e4 5.84e4 5.86e4 5.86e4
##  7 OECD        Denmark   7.68e3 7.68e3 7.68e3 9.45e3 1.19e4 1.23e4 1.23e4 1.30e4
##  8 OECD        Estonia   5.81e2 5.81e2 5.81e2 5.81e2 6.47e3 6.53e3 6.53e3 6.54e3
##  9 OECD        Finland   7.17e3 7.22e3 7.22e3 7.23e3 7.25e3 7.45e3 7.46e3 7.46e3
## 10 OECD        France    7.88e4 7.88e4 7.88e4 7.89e4 7.89e4 8.09e4 8.12e4 8.47e4
## # i 117 more rows
## # i 15 more variables: `2008` <dbl>, `2009` <dbl>, `2010` <dbl>, `2011` <dbl>,
## #   `2012` <dbl>, `2013` <dbl>, `2014` <dbl>, `2015` <dbl>, `2016` <dbl>,
## #   `2017` <dbl>, `2018` <dbl>, `2019` <dbl>, `2020` <dbl>, `2021` <dbl>,
## #   `2022` <dbl>
```

b. Use `pivot_longer()` to put the data in a tidy format. You'll need to use the same trick with the year column names as you did in 1a.

```
oecd <- oecd %>%
  pivot_longer(`2000`:`2022`,
               names_to = "Year",
               values_to = "Area")

oecd
```

```
## # A tibble: 2,921 x 4
##    OECD_member Country   Year     Area
##    <chr>       <chr>     <chr>   <dbl>
##  1 OECD        Australia 2000  376896.
##  2 OECD        Australia 2001  377198.
##  3 OECD        Australia 2002  399906.
##  4 OECD        Australia 2003  399923
##  5 OECD        Australia 2004  402052.
##  6 OECD        Australia 2005  406364.
##  7 OECD        Australia 2006  412438.
##  8 OECD        Australia 2007  417116.
##  9 OECD        Australia 2008  417560.
## 10 OECD        Australia 2009  442165.
## # i 2,911 more rows
```

3. Santa Cruz Rodents Data Cleaning (20 pts)

Start by reading in the rodent data from the Santa Cruz River, `capture_data.csv`.

Take a look at the data. You'll likely notice immediately that there are some issues to be fixed.

For this question, there isn't really a good way for me to show you the output in the answer key; you'll want to take a look at the data frame in R to make sure the issue got fixed.

a. Rename any column that needs to be renamed and save the output. This is the data frame we will use for the remainder of the questions.

```
rodents <- read_csv("capture_data.csv")
```

```
## Rows: 51 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (10): Site, Trap ID, Species, Status (R/N), Sex, Tail length, Hair samp...
## dbl   (4): Total Weight, Bag weight, Animal Weight, Hind foot length
## date  (1): Date
##
## 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.
```

```
rodents <- rodents %>%
  rename("TrapID" = `Trap ID`, "Status" = `Status (R/N)`, "TotalWeight" = `Total Weight`, "BagWeight" =

rodents
```

```
## # A tibble: 51 x 15
##    Date       Site     TrapID Species Status Sex   TotalWeight BagWeight
##    <date>     <chr>    <chr>  <chr>   <chr>  <chr>       <dbl>     <dbl>
##  1 2022-11-14 Heritage 4C     SIOC    N      F             134        18
##  2 2022-11-14 <NA>     4D     SIOC    N      M             136        18
##  3 2022-11-14 <NA>     4I     SIOC    N      <NA>           90        18
##  4 2022-11-14 <NA>     2H     REME    N      M              38        26
##  5 2022-11-14 <NA>     4J     SIOC?   N      <NA>           NA        NA
##  6 2022-11-14 <NA>     2F     REME    N      F              22        10
##  7 2022-11-15 <NA>     4C     SIOC    R      <NA>           NA        NA
##  8 2022-11-15 <NA>     4H     SIOC    N      F              95        11
##  9 2022-11-15 <NA>     1H     REME    N      <NA>           26         9
## 10 2022-11-15 <NA>     1B     REME    N      F              35         9
## # i 41 more rows
## # i 7 more variables: AnimalWeight <dbl>, HindfootLength <dbl>,
## #   TailLength <chr>, HairSample <chr>, Position <chr>, Handler <chr>,
## #   Notes <chr>
```

b. Next we need to fill in the missing values in the `Site` column.

```
rodents <- rodents %>%
  fill(Site)

rodents
```

```
## # A tibble: 51 x 15
##    Date       Site     TrapID Species Status Sex   TotalWeight BagWeight
##    <date>     <chr>    <chr>  <chr>   <chr>  <chr>       <dbl>     <dbl>
##  1 2022-11-14 Heritage 4C     SIOC    N      F             134        18
##  2 2022-11-14 Heritage 4D     SIOC    N      M             136        18
##  3 2022-11-14 Heritage 4I     SIOC    N      <NA>           90        18
##  4 2022-11-14 Heritage 2H     REME    N      M              38        26
##  5 2022-11-14 Heritage 4J     SIOC?   N      <NA>           NA        NA
##  6 2022-11-14 Heritage 2F     REME    N      F              22        10
##  7 2022-11-15 Heritage 4C     SIOC    R      <NA>           NA        NA
##  8 2022-11-15 Heritage 4H     SIOC    N      F              95        11
##  9 2022-11-15 Heritage 1H     REME    N      <NA>           26         9
## 10 2022-11-15 Heritage 1B     REME    N      F              35         9
## # i 41 more rows
## # i 7 more variables: AnimalWeight <dbl>, HindfootLength <dbl>,
## #   TailLength <chr>, HairSample <chr>, Position <chr>, Handler <chr>,
```

```
## #   Notes <chr>
```

c. In the `Species` column, there are 2 different species that have question marks next to their names. Using the `replace` function inside of a `mutate` function, remove the question marks (e.g., `SIOC?` should become `SIOC` and `DIME?` should become `DIME`.

(This is just for practice. In reality, we might want to create a code for unknown species or a column for unclear ID).

```
rodents <- rodents %>%
  mutate(Species = replace(Species, Species == "SIOC?", "SIOC")) %>%
  mutate(Species = replace(Species, Species == "DIME?", "DIME"))
```

d. If we look at the data classes for the columns, we can see that the column for tail length is character when it should be numeric. This usually indicates that there is a special character or letter somewhere in the column. As it turns out, in the last row, the value is `~15.5` instead of `15.5`. Use the `replace` function inside of `mutate` to convert that value to `15.5`.

```
rodents <- rodents %>%
  mutate(TailLength = replace(TailLength, TailLength == "~15.5", "15.5"))
```

e. In both the Hair Sample and Position columns, there is a `?`. Use the `na_if` function inside a `mutate` function to convert those `?` to `NA` values.

```
rodents <- rodents %>%
  mutate(HairSample = na_if(HairSample, "?"))



rodents <- rodents %>%
  mutate(Position = na_if(Position, "?"))
```

4. Remembering Joins (15 pts)

Let's remind ourselves about joins from Week 4.

Read in the vegetation data that goes along with the Santa Cruz rodent data. The .csv file is called `microsite_grouped_veg.csv`

a. Rename the columns that should be renamed. Use a consistent structure (and make the Site and Trap Location column names match those from the rodents data frame).

```
rodent_veg <- read_csv("microsite_grouped_veg.csv")
```

```
## New names:
## Rows: 80 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (4): Site, Trap Location, Type of Vegetation, Grouped_Veg dbl (4): ...1,
## Distance to Vegetation (m), Percent Veg Cover, Distance to Wa...
## 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.
## * `` -> `...1`
```

```
colnames(rodent_veg)
```

```
## [1] "...1"                      "Site"
## [3] "Trap Location"             "Distance to Vegetation (m)"
## [5] "Type of Vegetation"        "Percent Veg Cover"
```

```
## [7] "Distance to Water (m)"      "Grouped_Veg"
```

```
rodent_veg <- rodent_veg %>%
  rename("RecordID" = `...1`, "TrapID" = `Trap Location`, "DistanceToVegetation_m" = `Distance to Vegeta

rodent_veg
```

```
## # A tibble: 80 x 8
##    RecordID Site  TrapID DistanceToVegetation_m TypeOfVegetation PercentVegCover
##       <dbl> <chr> <chr>                   <dbl> <chr>                       <dbl>
##  1        1 Heri~ 2A                          0   Bermuda grass                50
##  2        2 Heri~ 2B                          0   Cheese bush                  30
##  3        3 Heri~ 2C                          5   Bermuda grass                 0
##  4        4 Heri~ 2D                          1   Salt cedar                   20
##  5        5 Heri~ 2E                          0   Bermuda grass                30
##  6        6 Heri~ 2F                          0   Cockleburr                   30
##  7        7 Heri~ 2G                          0.5 Unknown grass                20
##  8        8 Heri~ 2H                          0   Unknown grass                60
##  9        9 Heri~ 2I                          0   Cheesebush                   20
## 10       10 Heri~ 2J                          0   Bermuda grass                50
## # i 70 more rows
## # i 2 more variables: DistanceToWater_m <dbl>, GroupedVeg <chr>
```

b. Select the Site, Trap Location and Grouped Veg columns and save those as a new dataframe

```
rodent_veg_new <- rodent_veg %>%
  select(Site, TrapID, GroupedVeg)

rodent_veg_new
```

```
## # A tibble: 80 x 3
##    Site     TrapID GroupedVeg
##    <chr>    <chr>  <chr>
##  1 Heritage 2A     grass
##  2 Heritage 2B     shrubs
##  3 Heritage 2C     grass
##  4 Heritage 2D     shrubs
##  5 Heritage 2E     grass
##  6 Heritage 2F     forb
##  7 Heritage 2G     grass
##  8 Heritage 2H     grass
##  9 Heritage 2I     shrubs
## 10 Heritage 2J     grass
## # i 70 more rows
```

c. Use an `inner_join()` to join those same 2 data frames

```
rodent_veg_combined <- inner_join(rodents, rodent_veg_new, join_by(Site, TrapID))

rodent_veg_combined
```

```
## # A tibble: 51 x 16
##    Date       Site     TrapID Species Status Sex   TotalWeight BagWeight
##    <date>     <chr>    <chr>  <chr>   <chr>  <chr>       <dbl>     <dbl>
##  1 2022-11-14 Heritage 4C     SIOC    N      F             134        18
##  2 2022-11-14 Heritage 4D     SIOC    N      M             136        18
##  3 2022-11-14 Heritage 4I     SIOC    N      <NA>           90        18
```

```
##  4 2022-11-14 Heritage 2H     REME    N      M              38        26
##  5 2022-11-14 Heritage 4J     SIOC    N      <NA>           NA        NA
##  6 2022-11-14 Heritage 2F     REME    N      F              22        10
##  7 2022-11-15 Heritage 4C     SIOC    R      <NA>           NA        NA
##  8 2022-11-15 Heritage 4H     SIOC    N      F              95        11
##  9 2022-11-15 Heritage 1H     REME    N      <NA>           26         9
## 10 2022-11-15 Heritage 1B     REME    N      F              35         9
## # i 41 more rows
## # i 8 more variables: AnimalWeight <dbl>, HindfootLength <dbl>,
## #   TailLength <chr>, HairSample <chr>, Position <chr>, Handler <chr>,
## #   Notes <chr>, GroupedVeg <chr>
```

d. In your own words (~2-3 sentences), explain how the inner join in (c) worked.

The inner join in (c) worked by joining the two data frames based on matching the columns shared between the two, and then keeping all the columns in both data frames. Because I specified that I wanted to join by all 3 of the columns in rodent_veg_new, inner_join matched the rows in those columns to those in rodent_veg and kept the rest of the columns.

5. Santa Cruz Rodents Wrangling (20 pts)

Let's practice splitting and combining columns as well as pivoting the Santa Cruz rodent data.

a. Use the `separate()` function to split the date column into 3 separate columns.

```
rodents <- rodents %>%
  separate(Date, c("Year", "Month", "Day"), sep ="-")

rodents
```

```
## # A tibble: 51 x 17
##    Year  Month Day   Site     TrapID Species Status Sex   TotalWeight BagWeight
##    <chr> <chr> <chr> <chr>    <chr>  <chr>   <chr>  <chr>       <dbl>     <dbl>
##  1 2022  11    14    Heritage 4C     SIOC    N      F             134        18
##  2 2022  11    14    Heritage 4D     SIOC    N      M             136        18
##  3 2022  11    14    Heritage 4I     SIOC    N      <NA>           90        18
##  4 2022  11    14    Heritage 2H     REME    N      M              38        26
##  5 2022  11    14    Heritage 4J     SIOC    N      <NA>           NA        NA
##  6 2022  11    14    Heritage 2F     REME    N      F              22        10
##  7 2022  11    15    Heritage 4C     SIOC    R      <NA>           NA        NA
##  8 2022  11    15    Heritage 4H     SIOC    N      F              95        11
##  9 2022  11    15    Heritage 1H     REME    N      <NA>           26         9
## 10 2022  11    15    Heritage 1B     REME    N      F              35         9
## # i 41 more rows
## # i 7 more variables: AnimalWeight <dbl>, HindfootLength <dbl>,
## #   TailLength <chr>, HairSample <chr>, Position <chr>, Handler <chr>,
## #   Notes <chr>
```

b. Use `unite()` to bring them back together into 1 Date column

```
rodents <- rodents %>%
  unite("Date", Year, Month, Day, sep = "-")

rodents
```

```
## # A tibble: 51 x 15
##    Date     Site  TrapID Species Status Sex   TotalWeight BagWeight AnimalWeight
##    <chr>    <chr> <chr>  <chr>   <chr>  <chr>       <dbl>     <dbl>        <dbl>
```

```
##  1 2022-11~ Heri~ 4C     SIOC    N      F              134        18          116
##  2 2022-11~ Heri~ 4D     SIOC    N      M              136        18          118
##  3 2022-11~ Heri~ 4I     SIOC    N      <NA>            90        18           72
##  4 2022-11~ Heri~ 2H     REME    N      M               38        26           12
##  5 2022-11~ Heri~ 4J     SIOC    N      <NA>            NA        NA           NA
##  6 2022-11~ Heri~ 2F     REME    N      F               22        10           12
##  7 2022-11~ Heri~ 4C     SIOC    R      <NA>            NA        NA           NA
##  8 2022-11~ Heri~ 4H     SIOC    N      F               95        11           84
##  9 2022-11~ Heri~ 1H     REME    N      <NA>            26         9           17
## 10 2022-11~ Heri~ 1B     REME    N      F               35         9           26
## # i 41 more rows
## # i 6 more variables: HindfootLength <dbl>, TailLength <chr>, HairSample <chr>,
## #   Position <chr>, Handler <chr>, Notes <chr>
```

c. Summarize the data so that we have a count of each species per site. Save this output as a new data frame (do not overwrite the rodents data frame)

```
rodents_count <- rodents %>%
  group_by(Site, Species) %>%
  summarize(Count = n())
```

```
## `summarise()` has grouped output by 'Site'. You can override using the
## `.groups` argument.
```

```
rodents_count
```

```
## # A tibble: 7 x 3
## # Groups:   Site [2]
##   Site     Species Count
##   <chr>    <chr>   <int>
## 1 Drexel   CHPE        3
## 2 Drexel   DIME        5
## 3 Drexel   NEAB        1
## 4 Drexel   PEER        5
## 5 Drexel   SIOC        1
## 6 Heritage REME       10
## 7 Heritage SIOC       26
```

d. Convert the data from (c) from long format to wide format. Use an argument in the `pivot_wider` function to have all blank cells filled with 0 instead of `NA`.

```
rodents_count <- rodents_count %>%
  pivot_wider(names_from = Species, values_from = Count, values_fill = 0)

rodents_count
```

```
## # A tibble: 2 x 7
## # Groups:   Site [2]
##   Site      CHPE  DIME  NEAB  PEER  SIOC  REME
##   <chr>    <int> <int> <int> <int> <int> <int>
## 1 Drexel       3     5     1     5     1     0
## 2 Heritage     0     0     0     0    26    10
```

6. Mammals (20 pts)

The code chunk below has some made-up mammal data. Run the code chunk below to complete question 5.

```
mammals <- data.frame(site = c(1,1,2,3,3,3),
                      taxon = c('Suncus etruscus', 'Sorex cinereus',
                                'Myotis nigricans', 'Notiosorex crawfordi',
                                'Suncus etruscus', 'Myotis nigricans'),
                      density = c(6.2, 5.2, 11.0, 1.2, 9.4, 9.6),
                      mass = c(4.2, 5, 9.1, 8.6, 4.1, 8.7))
```

a. Use the `separate()` function to create columns for the genus and species (from the taxon column)

```
mammals <- mammals %>%
  separate(taxon, c("genus", "species"), sep= " ")

mammals
```

```
##   site      genus   species density mass
## 1    1     Suncus  etruscus     6.2  4.2
## 2    1      Sorex  cinereus     5.2  5.0
## 3    2     Myotis nigricans    11.0  9.1
## 4    3 Notiosorex crawfordi     1.2  8.6
## 5    3     Suncus  etruscus     9.4  4.1
## 6    3     Myotis nigricans     9.6  8.7
```

b. Use `pivot_longer` so that `density` and `mass` end up in one column and the values end up in another column

```
mammals <- mammals %>%
  pivot_longer(`density`:`mass`,
               names_to = "metric",
               values_to = "measurements")
mammals
```

```
## # A tibble: 12 x 5
##     site genus      species   metric  measurements
##    <dbl> <chr>      <chr>     <chr>          <dbl>
##  1     1 Suncus     etruscus  density          6.2
##  2     1 Suncus     etruscus  mass             4.2
##  3     1 Sorex      cinereus  density          5.2
##  4     1 Sorex      cinereus  mass             5
##  5     2 Myotis     nigricans density         11
##  6     2 Myotis     nigricans mass             9.1
##  7     3 Notiosorex crawfordi density          1.2
##  8     3 Notiosorex crawfordi mass             8.6
##  9     3 Suncus     etruscus  density          9.4
## 10     3 Suncus     etruscus  mass             4.1
## 11     3 Myotis     nigricans density          9.6
## 12     3 Myotis     nigricans mass             8.7
```

c. Even though the data from (b) is longer, it isn't tidier. Explain why not.

Each variable doesn't form a column. We should separate the "metric" column into two columns called "density" and "mass".

d. Use the `unite()` function to bring the genus and species column back together as one column with whatever separator you choose.

```
mammals <- mammals %>%
  unite("species_id", genus, species, sep="_" )
```

```
mammals
```

```
## # A tibble: 12 x 4
##     site species_id           metric  measurements
##    <dbl> <chr>                <chr>          <dbl>
##  1     1 Suncus_etruscus      density          6.2
##  2     1 Suncus_etruscus      mass             4.2
##  3     1 Sorex_cinereus       density          5.2
##  4     1 Sorex_cinereus       mass             5
##  5     2 Myotis_nigricans     density         11
##  6     2 Myotis_nigricans     mass             9.1
##  7     3 Notiosorex_crawfordi density          1.2
##  8     3 Notiosorex_crawfordi mass             8.6
##  9     3 Suncus_etruscus      density          9.4
## 10     3 Suncus_etruscus      mass             4.1
## 11     3 Myotis_nigricans     density          9.6
## 12     3 Myotis_nigricans     mass             8.7
```

e. Use `pivot_wider()` to bring the data frame back to it's original state.

```
mammals <- mammals %>%
  pivot_wider(names_from = metric, values_from = measurements)

mammals
```

```
## # A tibble: 6 x 4
##    site species_id           density  mass
##   <dbl> <chr>                  <dbl> <dbl>
## 1     1 Suncus_etruscus          6.2   4.2
## 2     1 Sorex_cinereus           5.2   5
## 3     2 Myotis_nigricans        11     9.1
## 4     3 Notiosorex_crawfordi     1.2   8.6
## 5     3 Suncus_etruscus          9.4   4.1
## 6     3 Myotis_nigricans         9.6   8.7
```