

HW 01 - Pet names

Meet the toolkit

Tina Huynh

The goal of this assignment is to introduce you to R, RStudio, Git, and GitHub, which you'll be using throughout the course both to learn the data science concepts discussed in the course and to analyze real data and come to informed conclusions.

Getting started

Prerequisites

This assignment assumes that you have reviewed the lectures titled “Meet the toolkit: Programming” and “Meet the toolkit: version control and collaboration”. If you haven't yet done so, please pause and complete the following before continuing.

Terminology

We've already thrown around a few new terms, so let's define them before we proceed.

- **R:** Name of the programming language we will be using throughout the course.
- **RStudio:** An integrated development environment for R. In other words, a convenient interface for writing and running R code.
- **Git:** A version control system.
- **GitHub:** A web platform for hosting version controlled files and facilitating collaboration among users.
- **Repository:** A Git repository contains all of your project's files and stores each file's revision history. It's common to refer to a repository as a repo.
 - In this course, each assignment you work on will be contained in a Git repo.
 - For individual assignments, only you will have access to the repo. For team assignments, all team members will have access to a single repo where they work collaboratively.
 - All repos associated with this course are housed in the course GitHub organization. The organization is set up such that students can only see repos they have access to, but the course staff can see all of them.

Starting slow

As the course progresses, you are encouraged to explore beyond what the assignments dictate; a willingness to experiment will make you a much better programmer! Before we get to that stage, however, you need to build some basic fluency in R. First, we will explore the fundamental building blocks of all of these tools.

Before you can get started with the analysis, you need to make sure you:



Figure 1: Photo by Jovana Askrabic on Unsplash

- have a GitHub account
- are a member of the course GitHub organization
- are a member of the course RStudio Cloud space

If you failed to confirm any of these, it means you have not yet completed the prerequisites for this assignment. Please go back to Prerequisites and complete them before continuing the assignment.

Workflow

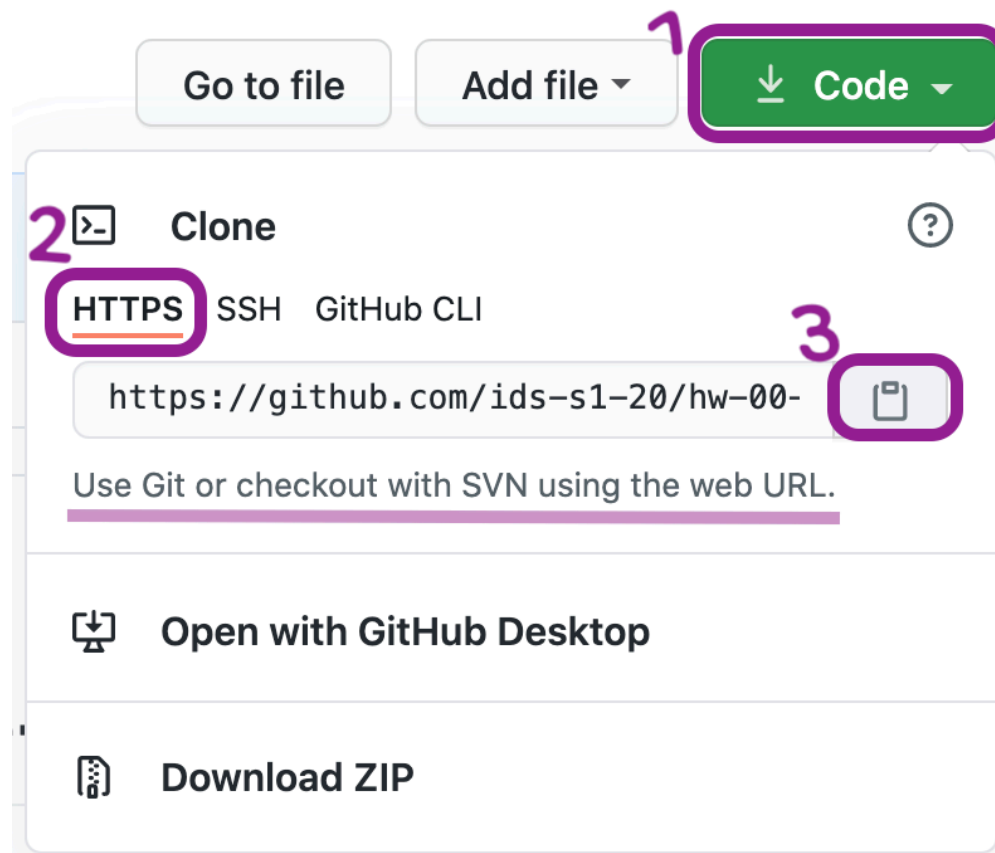
****IMPORTANT:**** If there is no GitHub repo created for you for this assignment, it means I didn't have y

For each assignment in this course you will start with a GitHub repo that I created for you and that contains the starter documents you will build upon when working on your assignment. The first step is always to bring these files into RStudio so that you can edit them, run them, view your results, and interpret them. This action is called **cloning**.

Then you will work in RStudio on the data analysis, making **commits** along the way (snapshots of your changes) and finally **push** all your work back to GitHub.

The next few steps will walk you through the process of getting information of the repo to be cloned, cloning your repo in a new RStudio Cloud project, and getting started with the analysis.

Step 1. Get URL of repo to be cloned



On GitHub, click on the green **Code** button, select **HTTPS** (this might already be selected by default, and if it is, you'll see the text *Use Git or checkout with SVN using the web URL* as in the image on the right). Click on the clipboard icon to copy the repo URL.

Step 2. Go to RStudio Cloud

Go to posit.cloud and then **navigate to the course workspace** via the left sidebar. It's very important that you do this for two reasons:

- It's only when you're in the course workspace that you'll be able to benefit from R packages I've pre-installed for you so that your project can be configured correctly.
- It's only when you're in the course workspace that your usage of RStudio Cloud won't count towards the free usage limits.

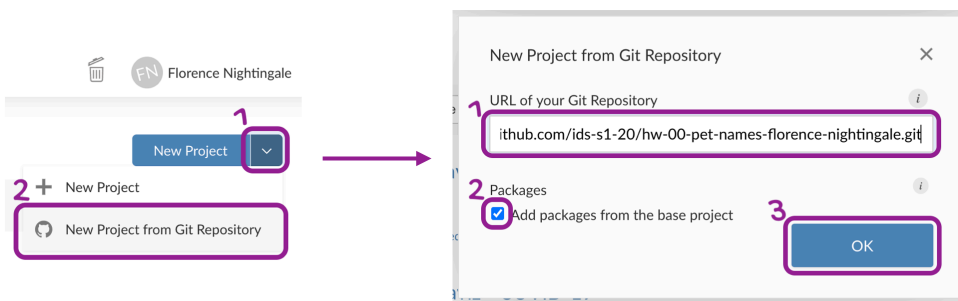


Before you proceed, confirm that you are in the course workspace by checking out what's on your top bar in RStudio Cloud.

Step 3. Clone the repo

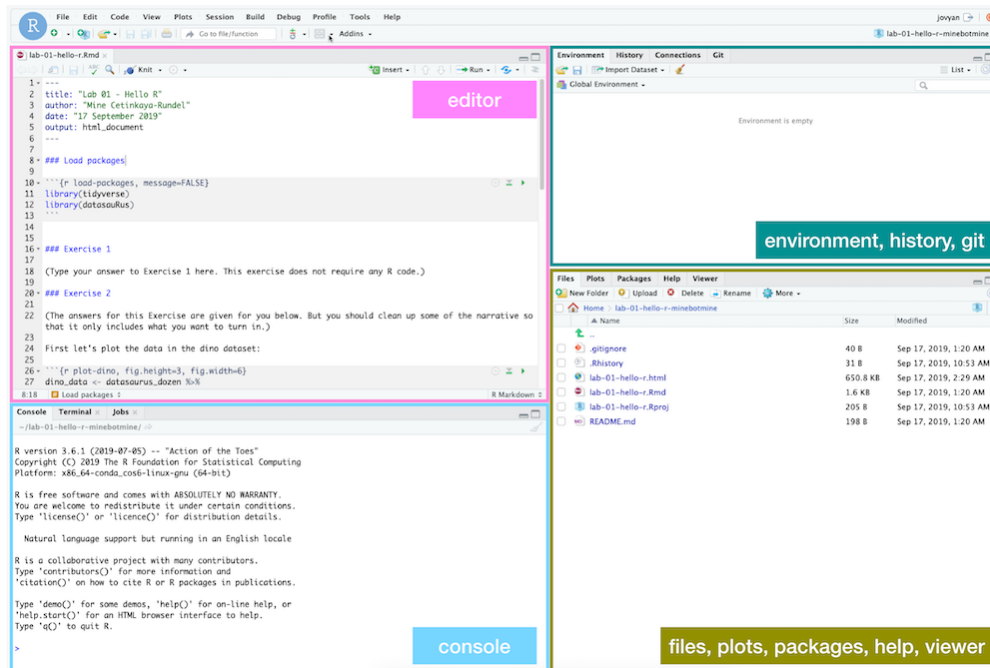
In RStudio, click on the **down arrow** next to New Project and then choose **New Project from Git Repository**.

In the pop-up window, **paste the URL** you copied from GitHub, make sure the box for **Add packages from the base project** is checked (it should be, by default) and then click **OK**.



Hello RStudio!

RStudio is comprised of four panes.



- On the bottom left is the Console, this is where you can write code that will be evaluated. Try typing `2 + 2` here and hit enter, what do you get?
- On the bottom right is the Files pane, as well as other panes that will come handy as we start our analysis.
- If you click on a file, it will open in the editor, on the top left pane.
- Finally, the top right pane shows your Environment. If you define a variable it would show up there. Try typing `x <- 2` in the Console and hit enter, what do you get in the **Environment** pane? Importantly, this pane is also where the **Git** interface lives. We will be using that regularly throughout this assignment.

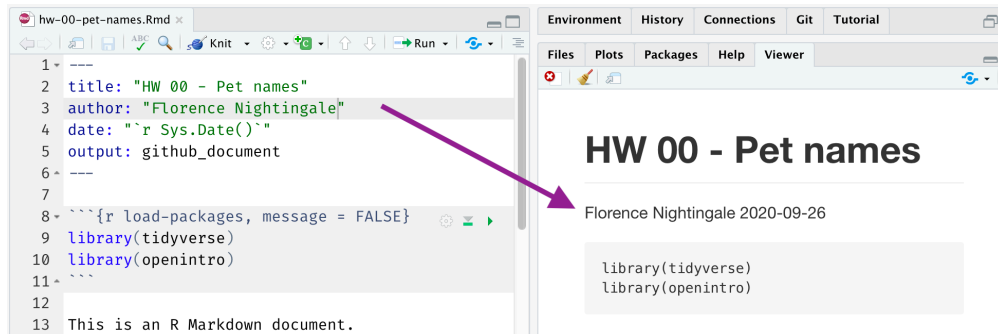
Warm up

Before we introduce the data, let's warm up with some simple exercises.

The top portion of your R Markdown file (between the three dashed lines) is called ****YAML****. It stands for

Step 1. Update the YAML

Open the R Markdown (Rmd) file in your project, change the author name to your name, and knit the document.

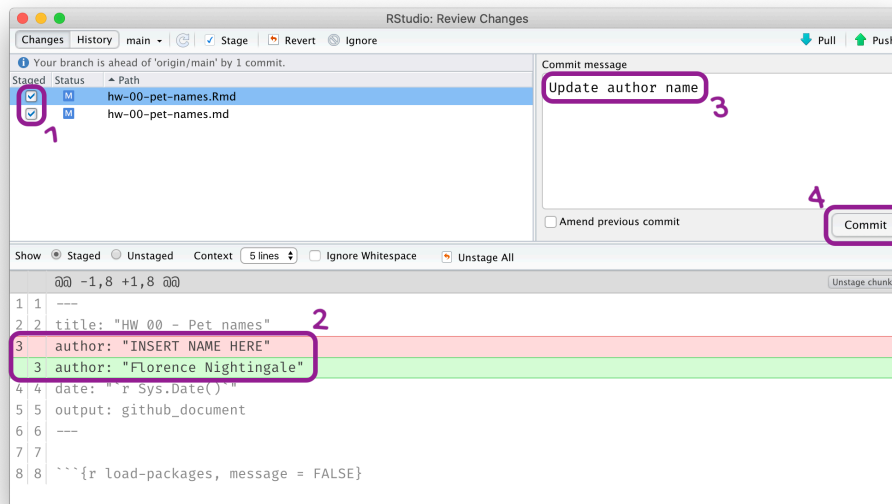


Step 2: Commit

Then Go to the **Git** pane in your RStudio.

You should see that your Rmd (R Markdown) file and its output, your md file (Markdown), are listed there as recently changed files.

Next, click on **Diff**. This will pop open a new window that shows you the **difference** between the last committed state of the document and its current state that includes your changes. If you're happy with these changes, click on the checkboxes of all files in the list, and type *"Update author name"* in the **Commit message** box and hit **Commit**.

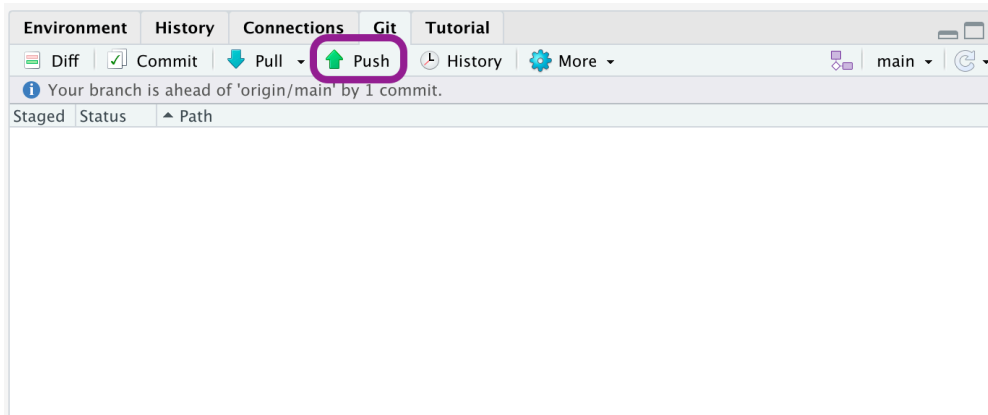


You don't have to commit after every change, this would get quite cumbersome. You should consider committing states that are *meaningful to you* for inspection, comparison, or restoration. In the first few assignments we will tell you exactly when to commit and in some cases, what commit message to use. As the semester progresses we will let you make these decisions.

Step 3. Push

Now that you have made an update and committed this change, it's time to push these changes to the web! Or more specifically, to your repo on GitHub. Why? So that others can see your changes. And by others,

we mean the course teaching team (your repos in this course are private to you and us, only). In order to push your changes to GitHub, click on **Push**.



This will prompt a dialogue box where you first need to enter your user name, and then your password. This might feel cumbersome. Bear with me... I *will* teach you how to save your password so you don't have to enter it every time. But for this one assignment you'll have to manually enter each time you push in order to gain some experience with it.

Thought exercise: Which of the above steps (updating the YAML, committing, and pushing) needs to talk to GitHub?¹

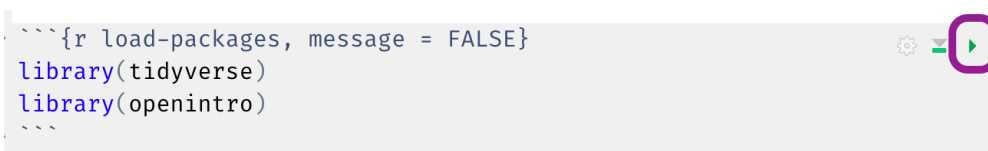
Updating the YAML is also a local operation, it doesn't require any communication with GitHub. Committing is a local operation, it doesn't require any communication with GitHub. Pushing requires communication with GitHub, as it is the step that sends your changes to GitHub.

Packages

R is an open-source language, and developers contribute functionality to R via packages. In this assignment we will use the following packages:

- **tidyverse**: a collection of packages for doing data analysis in a “tidy” way
- **openintro**: a package that contains the datasets from OpenIntro resources

We use the `library()` function to load packages. In your R Markdown document you should see an R chunk labelled `load-packages` which has the necessary code for loading both packages. You should also load these packages in your Console, which you can do by sending the code to your Console by clicking on the **Run Current Chunk** icon (green arrow pointing right icon).



Note that these packages also get loaded in your R Markdown environment when you **Knit** your R Markdown document.

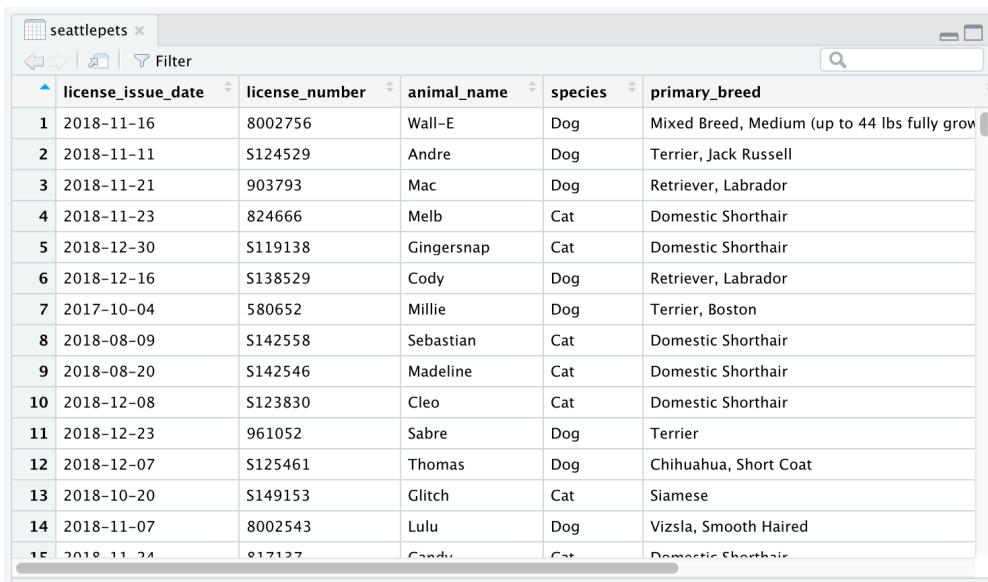
¹Only pushing requires talking to GitHub, this is why you're asked for your password at that point.

Data

The city of Seattle, WA has an open data portal that includes pets registered in the city. For each registered pet, we have information on the pet's name and species. The data used in this exercise can be found in the **openintro** package, and it's called **seattlepets**. Since the dataset is distributed with the package, we don't need to load it separately; it becomes available to us when we load the package.

You can view the dataset as a spreadsheet using the **View()** function. Note that you should not put this function in your R Markdown document, but instead type it directly in the Console, as it pops open a new window (and the concept of popping open a window in a static document doesn't really make sense...). When you run this in the console, you'll see the following **data viewer** window pop up.

```
View(seattlepets)
```



	license_issue_date	license_number	animal_name	species	primary_breed
1	2018-11-16	8002756	Wall-E	Dog	Mixed Breed, Medium (up to 44 lbs fully grown)
2	2018-11-11	S124529	Andre	Dog	Terrier, Jack Russell
3	2018-11-21	903793	Mac	Dog	Retriever, Labrador
4	2018-11-23	824666	Melb	Cat	Domestic Shorthair
5	2018-12-30	S119138	Gingersnap	Cat	Domestic Shorthair
6	2018-12-16	S138529	Cody	Dog	Retriever, Labrador
7	2017-10-04	580652	Millie	Dog	Terrier, Boston
8	2018-08-09	S142558	Sebastian	Cat	Domestic Shorthair
9	2018-08-20	S142546	Madeline	Cat	Domestic Shorthair
10	2018-12-08	S123830	Cleo	Cat	Domestic Shorthair
11	2018-12-23	961052	Sabre	Dog	Terrier
12	2018-12-07	S125461	Thomas	Dog	Chihuahua, Short Coat
13	2018-10-20	S149153	Glitch	Cat	Siamese
14	2018-11-07	8002543	Lulu	Dog	Vizsla, Smooth Haired
15	2018-11-24	817127	Gandy	Cat	Domestic Shorthair

You can find out more about the dataset by inspecting its documentation (which contains a **data dictionary**, name of each variable and its description), which you can access by running **?seattlepets** in the Console or using the Help menu in RStudio to search for **seattlepets**.

By running **?seattlepets** in the Console or using the Help menu in RStudio to search for **seattlepets**, you'll see documentation about the dataset pop up in the Help pane which contains information such as the source of the data, a brief description, and a data dictionary that describes each variable in the dataset.

Exercises

Exercise #1

According to the data dictionary, how many pets are included in this dataset?

There are a total of 52,519 pets in the dataset with 7 variables for each pet.

```
dimensions <- dim(seattlepets)

num_rows <- dimensions[1]
```



```
num_cols <- dimensions[2]

rm(dimensions)
```

Write your answer in your R Markdown document under Exercise 1, knit the document, commit your changes with a commit message that says “Completed Exercise 1”, and push. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

Exercise #2

Again, according to the data dictionary, how many variables do we have for each pet?

There are 7 variables for each pet in the dataset: - **license_issue_date**: Date the animal was registered with Seattle - **license_number**: Unique identifier for each pet license - **animal_name**: Name of the pet - **species**: Species of the pet (e.g., Dog, Cat, etc.) - **primary_breed**: Primary breed of the pet - **secondary_breed**: Secondary breed of the pet (if applicable) - **zip_code**: Zip code animal is registered in

```
column_variables <- colnames(seattlepets)
```

Write your answer in your R Markdown document under Exercise 2, knit the document, commit your changes with a commit message that says “Completed Exercise 2”, and push. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

Exercise #3

What are the three most common pet names in Seattle?

The most common pet names are *max*, *bella*, and *charlie*.

To do this you will need to count the frequencies of each pet name and display the results in descending order of frequency so that you can easily see the top three most popular names. The following code does exactly that.

The two lines of code can be read as "Start with the seattlepets data frame, and then count the animal_

The code below takes the **seattlepets** dataset from the **openintro** package and produces a frequency table of pet names. It first standardizes the **animal_name** variable by converting all names to lowercase and trimming any leading or trailing spaces. It then removes missing (NA) or blank entries to ensure only valid names remain. Finally, it counts the occurrences of each unique name, sorts them from most to least common, and labels the frequency column as **count** for clarity.

```
popular_names <- seattlepets %>%
  mutate(animal_name = animal_name |>
    tolower() |>
    trimws()) %>% # remove leading/trailing spaces
  filter(!is.na(animal_name), animal_name != "") %>%
  count(animal_name, sort = TRUE, name = "count")
```

Write your answer in your R Markdown document under Exercise 3. In this exercise you will not only provide a written answer but also include some code and output. You should insert the code in the code chunk provided for you, knit the document to see the output, and then write your narrative for the answer based on the output of this function, and knit again to see your narrative, code, and output in the resulting

document. Then, commit your changes with a commit message that says “Completed Exercise 3”, and push. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

Let’s also look to see what the most common pet names are for various species. For this we need to first `group_by()` the species, and then do the same counting we did before.

Looks like many of those NAs were cats. Poor unnamed kitties...

```
seattlepets %>%  
  group_by(species) %>%  
  count(animal_name, sort = TRUE)
```

```
## # A tibble: 16,823 x 3  
## # Groups:   species [4]  
##   species animal_name      n  
##   <chr>    <chr>      <int>  
## 1 Cat      <NA>         406  
## 2 Dog      Lucy          337  
## 3 Dog      Charlie        306  
## 4 Dog      Bella          249  
## 5 Dog      Luna           244  
## 6 Dog      Daisy          221  
## 7 Dog      Cooper         189  
## 8 Dog      Lola           187  
## 9 Dog      Max            186  
## 10 Dog     Molly          186  
## # i 16,813 more rows
```

But this output isn’t exactly what we wanted. We wanted to know the most common cat and dog names, but there are barely any cats present in this output! This is because there are more dogs than cats in the dataset overall. We can confirm this by counting the various species in the data.

6 pigs in the city? Ok... But we'll continue with cats and dogs.

```
seattlepets %>%  
  count(species, sort = TRUE)
```

```
## # A tibble: 4 x 2  
##   species      n  
##   <chr>    <int>  
## 1 Dog     35181  
## 2 Cat     17294  
## 3 Goat      38  
## 4 Pig        6
```

Let’s search for the top 5 cat and dog names. To do this, we can use the `slice_max()` function. The first argument in the function is the variable we want to select the highest values of, which is `n`. The second argument is the number of rows to select, which is `n = 5` for the top 5. It may be a bit confusing that both of these are `n`, but this is because we already have a variable called `n` in the data frame.

```
seattlepets %>%
  group_by(species) %>%
  count(animal_name, sort = TRUE) %>%
  slice_max(n, n = 5)
```

```
## # A tibble: 53 x 3
## # Groups:   species [4]
##   species animal_name     n
##   <chr>    <chr>      <int>
## 1 Cat      <NA>         406
## 2 Cat      Luna          111
## 3 Cat      Lucy          102
## 4 Cat      Lily           86
## 5 Cat      Max            83
## 6 Dog      Lucy          337
## 7 Dog      Charlie       306
## 8 Dog      Bella         249
## 9 Dog      Luna          244
## 10 Dog     Daisy         221
## # i 43 more rows
```

Exercise #4

Based on the previous output we can easily identify the most common cat and dog names in Seattle, but the output is sorted by `n` (the frequencies) as opposed to being organized by the `species`. Build on the pipeline to arrange the results so that they're arranged by `species` first, and then `n`. This means you will need to add one more step to the pipeline, and you have two options: `arrange(species, n)` or `arrange(n, species)`. You should try both and decide which one organizes the output by species and then ranks the names in order of frequency for each species.

```
popular_names <- seattlepets %>%
  mutate(animal_name = animal_name |>
    tolower() |>
    trimws()) %>%
  filter(!is.na(animal_name), animal_name != "") %>%
  count(species, animal_name, sort = TRUE, name = "count")
```

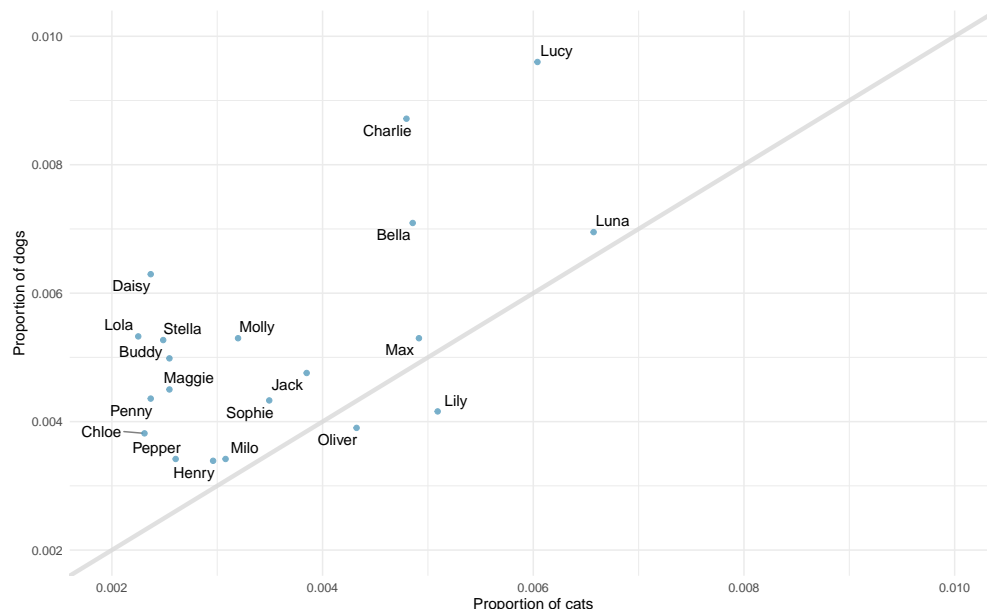
```
popular_names <- popular_names %>%
  group_by(species) %>%
  arrange(species, desc(count), animal_name)
```

```
top_species_names <- popular_names %>%
  group_by(species) %>%
  slice_max(order_by = count, n = 10, with_ties = FALSE)
```

```
# Start from popular_names (species, animal_name, count)
contingency_tbl <- top_species_names %>%
  pivot_wider(
    names_from = species,    # columns become species
    values_from = count,    # values are counts
    values_fill = NA        # fill missing combos with 0
  )
```

Write your answer in your R Markdown document under Exercise 4. In this exercise you're asked to complete the code provided for you. You should insert the code in the code chunk provided for you, knit the document to see the output, and then write your narrative for the answer based on the output of this function, and knit again to see your narrative, code, and output in the resulting document. Then, commit your changes with a commit message that says "Completed Exercise 4", and push. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

The following visualization plots the proportion of dogs with a given name versus the proportion of cats with the same name. The 20 most common cat and dog names are displayed. The diagonal line on the plot is the $x = y$ line; if a name appeared on this line, the name's popularity would be exactly the same for dogs and cats.



1. What names are more common for cats than dogs? The ones above the line or the ones below the line?
 - **Above the line** → proportion of dogs > proportion of cats (more common among dogs). **Names below the line** (e.g., *pepper*, *chloe*, *penny*, *lola*) are **more common for cats**.
 - **Below the line** → proportion of cats > proportion of dogs (more common among cats). **Names above the line** (e.g., *lucy*, *charlie*, *bella*) are **more common for dogs**.
2. Is the relationship between the two variables (proportion of cats with a given name and proportion of dogs with a given name) positive or negative? What does this mean in context of the data?
 - The relationship is **positive**: names that are common among cats tend also to be common among dogs.
 - In context, this means **popular names tend to be popular across both species**. For example, *luna* and *lucy* rank highly for both cats and dogs, while less popular names remain less common in both groups.

Now is a good time to commit and push your changes to GitHub with an appropriate commit message. Commit and push all changed files so that your Git pane is cleared up afterwards. Make sure that your last push to the repo comes before the deadline. You should confirm that what you committed and pushed are indeed in your repo that we will see by visiting your repo on GitHub.