HW 01 - Pet names

Susan Beckenham

Figure 1: Photo by Jovana Askrabic on Unsplash

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.

What You'll Learn

By the end of this assignment, you will be able to:

- Clone a GitHub repository to JupyterHub
- Navigate the RStudio interface
- Edit and knit R Markdown documents
- Use basic R functions to explore data
- Make commits with meaningful messages
- Push changes to GitHub
- Generate a PDF document for submission

Don't worry if these terms are unfamiliar! We'll walk through each step carefully.

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:

- have a GitHub account
- are a member of the course GitHub organization
- have successfully logged in and authenticated in the JupyterHub

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.

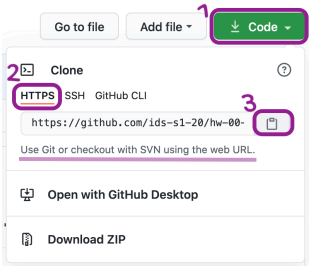
Workflow

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.

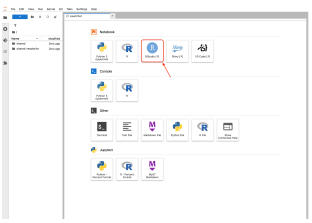
IMPORTANT: If there is no GitHub repo created for you for this assignment, it means I didn't have your GitHub username as of when I assigned the homework. Please let me know your GitHub username asap, and I can create your repo.

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 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.



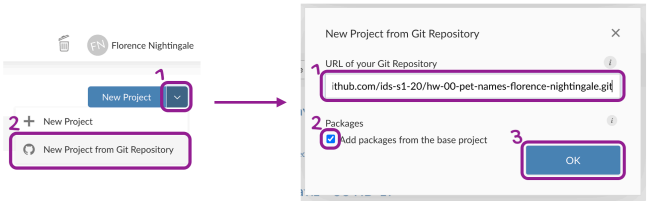
Step 2. Go to JupyterHub and open RStudio

Go to JuptyerHub and then **open an RStudio Notebook**.

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**.

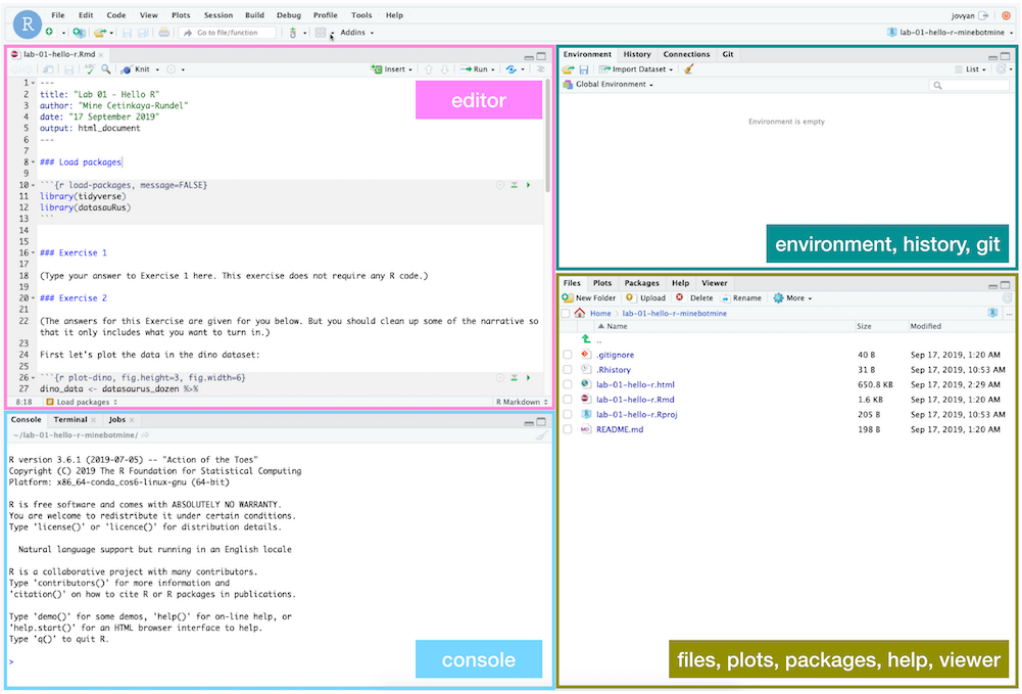


Checkpoint: You should now see your project files in the Files pane (bottom right). If you don't see a file called `Homework Instructions`, something went wrong - ask for help before continuing.

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.

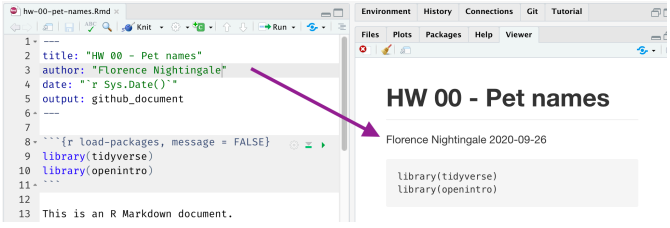
Warm up

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

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.

The top portion of your R Markdown file (between the three dashed lines) is called **YAML**. It stands for "YAML Ain't Markup Language". It is a human friendly data serialization standard for all programming languages. All you need to know is that this area is called the YAML (we will refer to it as such) and that it contains meta information about your document.

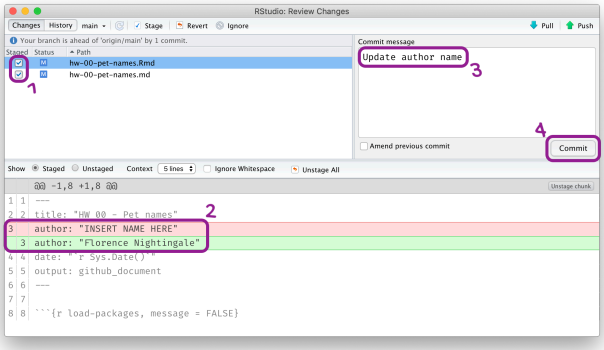


Step 2: Commit

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.

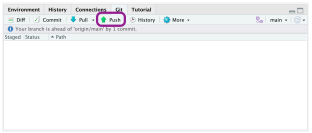
Next, click on **Diff**. This will pop open a new window that shows you the **diff**erence between the last committed state of the document and its current state that includes your changes. If you're happy with these changes, click on the checkboxes of all files in the list, and type *"Update author 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 may need to authenticate with GitHub. Follow the prompts to complete the authentication process.

Note: The first time you push, you may need to set up authentication. If you encounter issues, refer to the authentication guide posted on Canvas or ask for help during office hours.

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

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

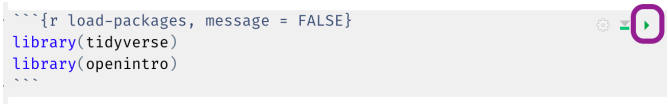
Checkpoint: After pushing, go to your GitHub repo in your web browser and refresh the page. You should see your updated file with your name in it. If you don't see the changes, try pushing again or ask for help.

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.

Checkpoint: If the packages loaded successfully, you should see no error messages in red. If you see an error like "there is no package called…", let your instructor know.

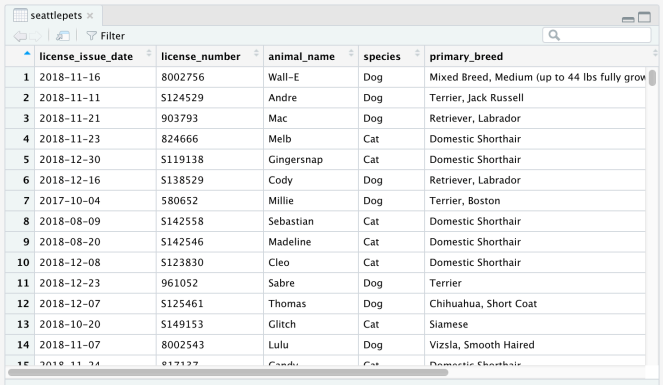
Data

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)
```



seattlepets

Filter

	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 grow
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	S80652	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

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`.

Common Issues and Solutions

As you work through this assignment, you might encounter some issues. Here are the most common ones:

Knitting errors:

- **Error: "there is no package called…"** → Run `library(tidyverse)` and `library(openintro)` in your Console first
- **Error: "object not found"** → Make sure you've run all code chunks before the one causing the error
- **Can't find the Knit button** → Look at the top of the editor pane (where your .Rmd file is open)

Git/GitHub issues:

- **Git pane is empty** → You haven't made any changes yet, or you need to save your file first
- **Can't push** → Make sure you committed first (commit before push!)
- **Authentication error** → Check with your instructor about the authentication setup for JupyterHub

R Markdown issues:

- **Output doesn't match what I see in Console** → Click the **Run All Chunks Above** button (down arrow icon) before knitting
- **My changes aren't showing** → Make sure you saved the file (Ctrl+S or Cmd+S)

General tips:

- Save your work frequently (Ctrl+S or Cmd+S)
- Knit often to catch errors early
- Read error messages carefully - they often tell you what's wrong
- Don't hesitate to ask for help!

Exercises

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

   There are 52,519 pets in the dataset.
   After completing this exercise:

a. Write your answer in the R Markdown document under Exercise 1
b. Click **Knit** to generate the output
c. Go to the **Git pane**, click **Diff**, then check the boxes next to all changed files
d. Type commit message: `"Completed Exercise 1"`
e. Click **Commit**, then click **Push**
f. Verify your Git pane is cleared (no files listed)

2. Again, according to the data dictionary, how many variables do we have for each pet? Write your answer in the R Markdown document under Exercise 2.

   There are 7 variables for each pet.
   After completing this exercise:
   Knit → Commit with message `"Completed Exercise 2"` → Push

3. What are the three most common pet names in Seattle? 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_names, and display the results sorted in descending order. The 'and then' in the previous sentence maps to %>%, the pipe operator, which takes what comes before it and plugs it in as the first argument of the function that comes after it."

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

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. The three most frequent names are Lucy, Charlie, and Luna. It is interesting that there are the largest amount for a N?A answer of 483.

After completing this exercise:

Knit → Commit with message `"Completed Exercise 3"` → Push

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
```

4. 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) %>%
  arrange(species,n) %>%
  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
```

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.

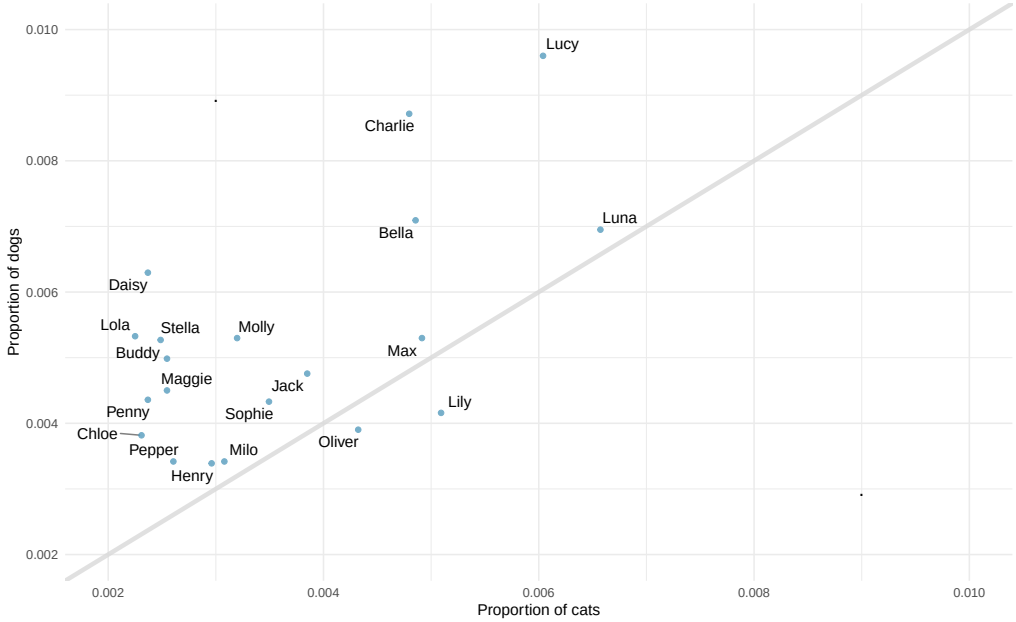
Which option groups all the cats together and all the dogs together, with names ranked by frequency within each species?

Arrange(species,n) would do the arrangement as required. *After completing this exercise:*

Knit → Commit with message `"Completed Exercise 4` → Push

5. 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.

Tip: You don't need to understand all the code that creates this visualization - that will come later in the course. For now, just look at the plot and answer the questions about what you observe.



a. What names are more common for cats than dogs? The ones above the line or the ones below the line?

b. 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?

After completing this exercise:

Unable to complete as there is an error with a missing file - libMagick++ Error in dyn.load(file, DLLpath = DLLpath, …) : unable to load shared object '/srv/rlibs/magick/libs/magick.so': libMagick++-6.Q16.so.8: cannot open shared object file: No such file or directory

Knit → Commit with message `"Completed Exercise 5"` → Push

To submit to Canvas:

1. In RStudio, click the **Knit** dropdown menu (next to the Knit button)
2. Select **Knit to tufte_handout** to generate a PDF
3. Download the PDF file from the Files pane
4. Upload the PDF to Canvas

Final Checkpoint: Visit your GitHub repo one more time to confirm all your work is there. We will grade what we see in your repo on GitHub!