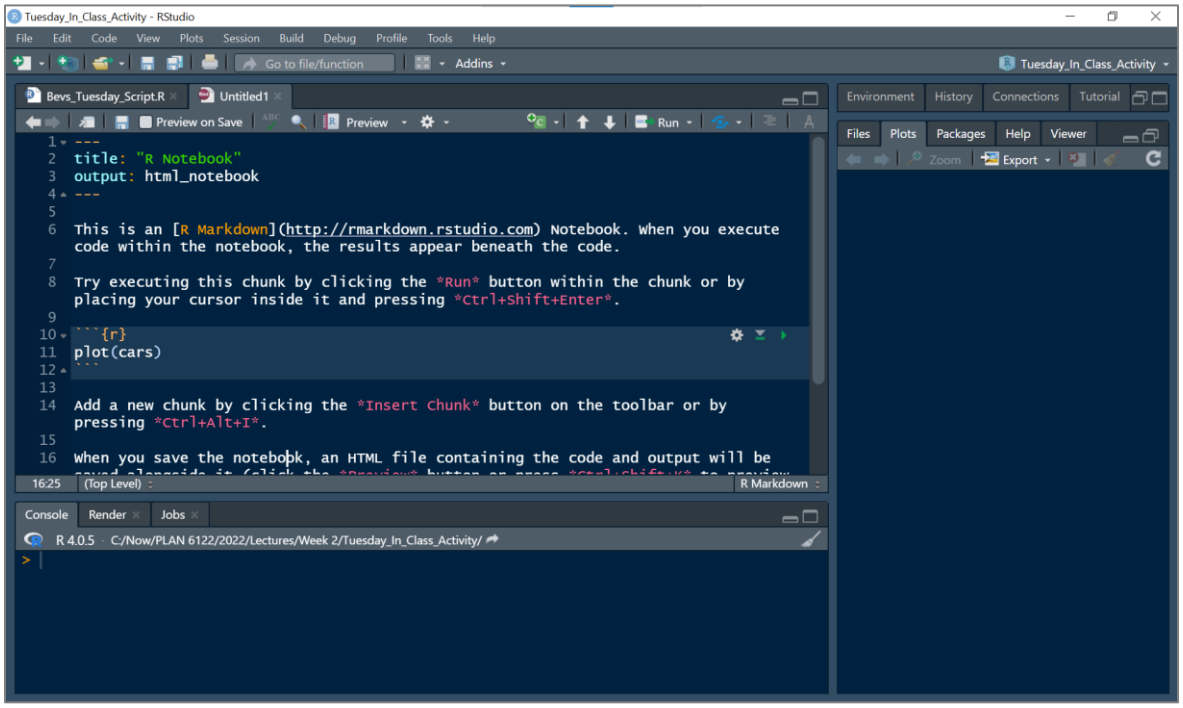
OVERVIEW

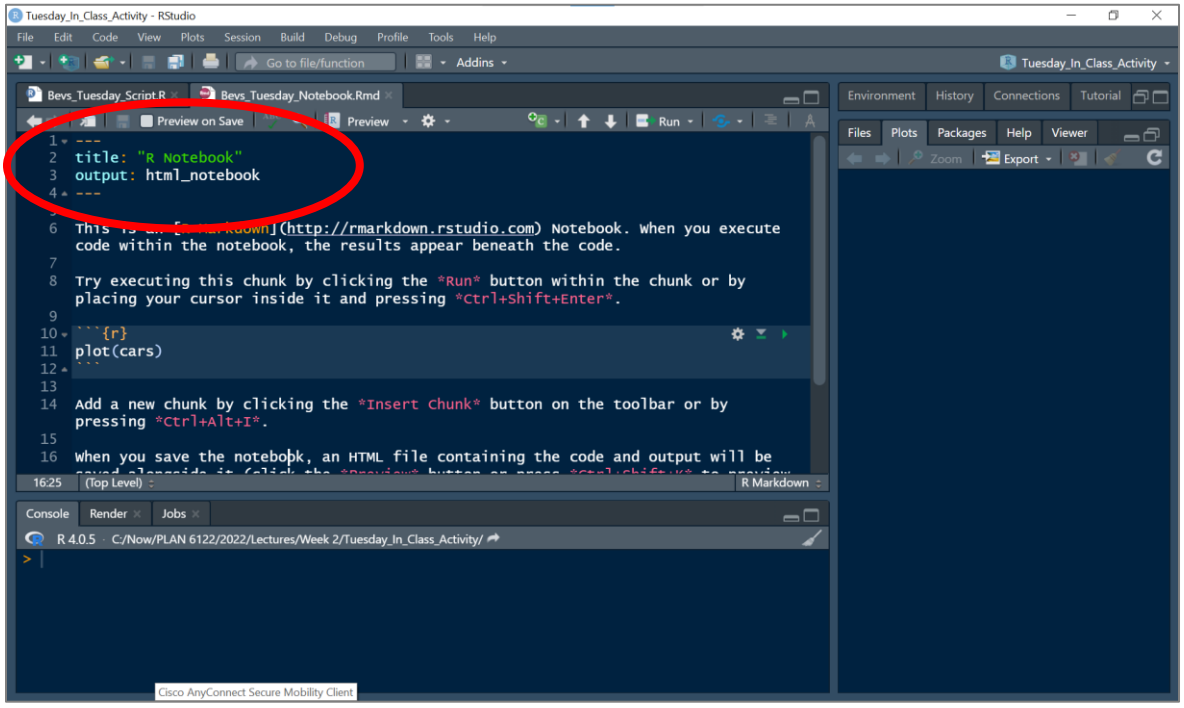
This Required Practice exercise provides some more structured exploration and practice with R Notebooks (R Markdown format). We will mix Markdown sections with code chunks, build familiarity with basic data types, and experiment with importing a tabular dataset. Because this is an in-class exercise, **there is nothing you need to submit**—the goal is to apply what we have read and seen in the lectures.

PRACTICE FORMATTING TEXT WITH R MARKDOWN

Launch RStudio and create a new **R Notebook** by selecting this option from the **New File** button list  and the template shown below will appear:



Give it a name and decide whether you want to add it to an existing **R Project** or store it as standalone **R Notebook** (either is fine for our purposes).



At the very top of the notebook we have the YAML or "Yet Another Markdown Language" header which begins and ends with three dashes --- and at the very least, specifies the title and output format when the document is rendered by the knitr package. There can be other options specified in the YAML, particularly if you are rendering to a format other than HTML (see the assigned reading from this week). Add the following to the YAML header:

- title: The title of your document. Note, this is not the same as the file name.
- author: Your name.
- date: This is the date that the file is created.

Below the YAML is where the Markdown sections typically begin. These are sections of text that usually explain or contextualize the code and graphics and are formatted using Markdown syntax.

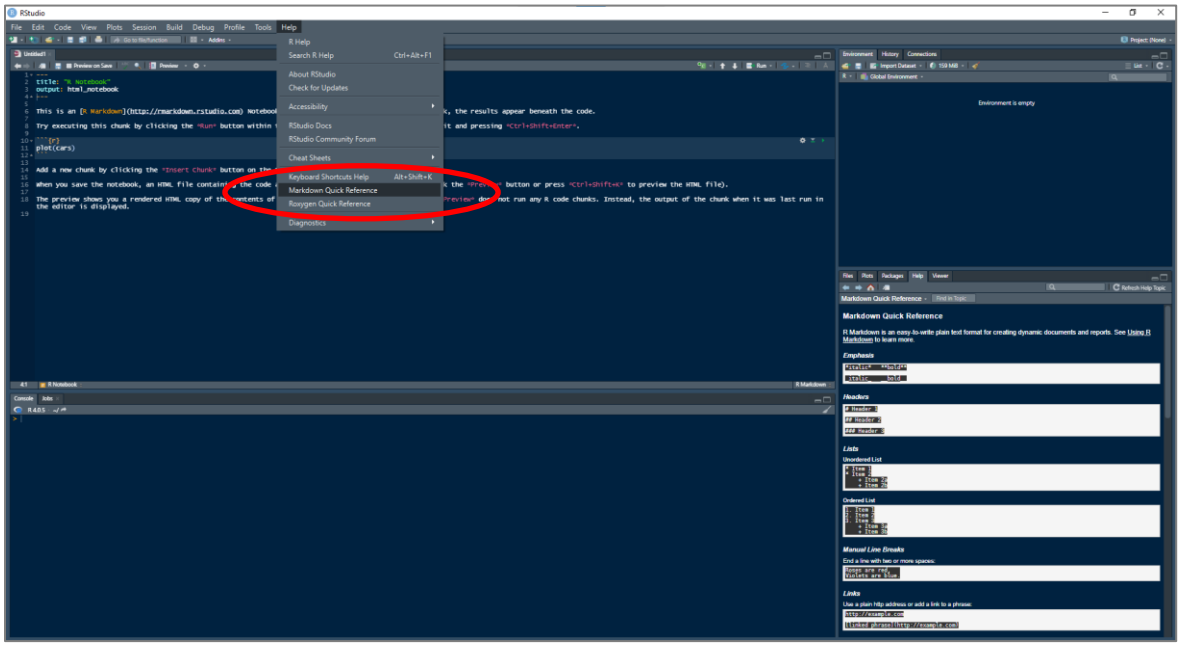
- #: a header element.
- **: bold text.
- *: italic text.
- ` : code blocks.

*** **Your turn**: Spend a few moments reviewing this website to become familiar with what participatory budgeting is and (in general), how it works. Then, use what you have learned about text formatting to add 3-5 sentences as a Markdown section below the YAML that explain—in your own words—the following:

- What participatory budgeting is;
- How long has it been around and where did it originate?;
- Why do advocates argue that participatory budgeting is desirable?;

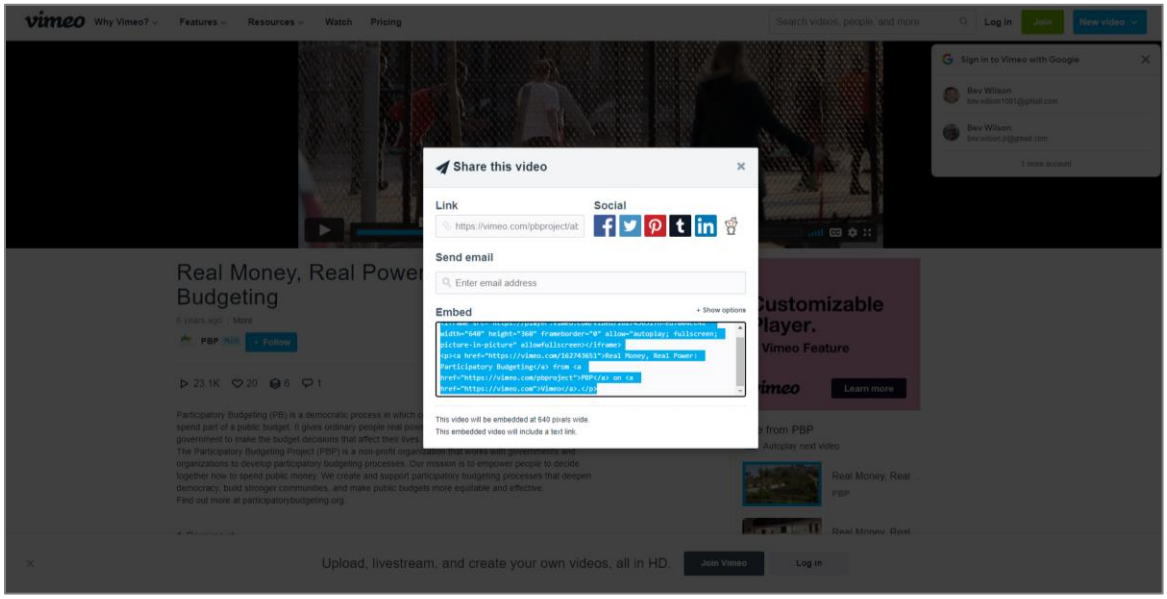
- Make sure to make gratuitous use of **bold**, *italics*, etc.

You have access to the Markdown Quick Reference directly from RStudio (**Help** → **Markdown Quick Reference**) as shown below:



Now you should **Save**, then **Knit** your notebook and inspect your work (either in the **Viewer** tab or in a web browser).

Let's assume that the video hosted on that webpage might be useful for others who view our rendered R Notebook (tl;dr). We can insert the video using R Markdown and it will render in the output when we knit the document. All you need to do is open the embed code for the video itself (i.e., click the Share button to see the embed code), then cut-and-paste it into your Markdown section:




```
<iframe src="https://player.vimeo.com/video/162743651?h=ed7004cc41" width="640" height="360" frameborder="0" allow="autoplay; fullscreen; picture-in-picture" allowfullscreen data-external="1"></iframe>

<p><a href="https://vimeo.com/162743651">Real Money, Real Power: Participatory Budgeting</a> from <a href="https://vimeo.com/pbproject">PBP</a> on <a href="https://vimeo.com">Vimeo</a>.</p>
```

What we have above is HTML code. Without going into lots of detail, the <iframe> tag is what inserts the video (or another document) inline with the Markdown text. The **src** attribute provides the URL for the video, while the remaining attributes outline the specifics of *how* the video will appear. For our purposes, the most important of these attributes is data-external="1" which is currently required for the **knitr** package to properly render video content. This attribute is typically **NOT SPECIFIED** in the embed code you find on the web by default and **YOU WILL NEED TO ADD IT** in order for video content to render in your output.

The remaining HTML code shown above (starting with the paragraph tag <p>) inserts some additional descriptive text with hyperlinks immediately below the video.

*** **Your turn**: Embed this video in your Markdown section after the introductory sentences you have written. Keep or remove the part that begins with the paragraph tag <p> if you like. Knit your notebook again and inspect your work (either in the **Viewer** tab or in a web browser).

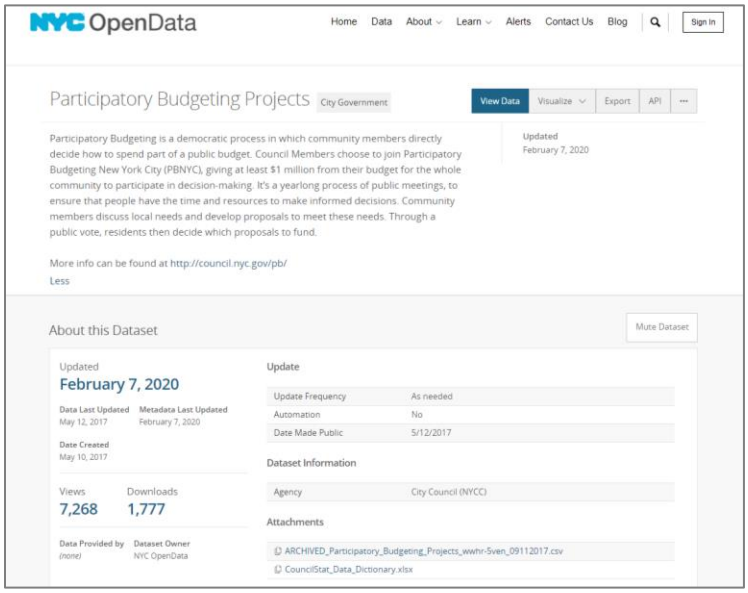
Celebrate! 😇

TIME TO GET CHUNKY!

As you know, the other component of an **R Notebook** is code chunks. Recall that code chunks are where the R code lives in an **R Notebook**. These are easy to spot because they always have three backticks followed by {r} as shown below:

```
```{r chunk-name-with-no-spaces}
code goes here
```
```

Accompanying this document on the Collab site, you will find a .csv file called "Participatory_Budgeting_Projects.csv". This file was taken from the New York City open data portal and can be found here. You can see that this dataset was created in 2017 and last updated in early 2020.



Take a few moments to explore the dataset by scrolling down the page and viewing the "Columns in this Dataset" and "Table Preview" sections in particular. Note that there are 20 columns or attributes in the dataset.

First, let's write some code that loads R packages we need. Add a horizontal rule or page break after your Markdown section, then insert a code chunk like this:

```
```{r label="GetTidyverse", message = FALSE, warning = FALSE}

library(tidyverse)

```
```

The backticks and curly braces let us know that this is a code chunk. The **label** option is not required, but can be helpful when you render the notebook (otherwise you will see "unnamed-chunk" in **the Render** tab next to the **Console** tab).

The other two options (described in greater detail here) suppress all warnings and messages that might otherwise appear in the rendered HTML output, making it unnecessarily cluttered and hard to read.

*** **Your turn**: Review the documentation for **readr:: read_delim** paying special attention to the "Arguments" and "Examples" sections of the help page. Note that the double colon above tells us that the function **read_delim** is located within the **readr** package, which happens to be part of the tidyverse suite of R packages.

- Insert a new code chunk and write a line of code that imports data stored in the .csv file and assigns it to an object called nyc_ppb
    - The delimiter (i.e., character used to indicate breaks between columns) for this data is the semicolon **;**
    - If you are not working within an R Project, you will need to specify the location of the .csv file **OR** move the .csv file into the same directory as your notebook
- Use the **str** function to get a quick sense of the contents of the dataset. Note that we can see in the rendered output the dimensions of the dataset (always written as rows x columns), the names of the columns, and the data type of each column.
- Knit your notebook again and inspect your work (either in the **Viewer** tab or in a web browser).

We can also explore some of the functionality described in the second assigned reading for today. Take another look at this, then proceed insert a new code chunk like the one below:

```
```{r label="Create2017Dataset", message = FALSE, warning = FALSE}

nyc_ppb_2017 <- nyc_ppb %>%
  filter(`Vote Year` == 2017) %>%
  filter(!is.na(Cost))

```
```

The above code uses the pipe operator, which we will talk about next week, to "chain together" lines of code. It then uses the **filter** function from the **dplyr** package to create a new dataset that only contains participatory budgeting records from 2017 that are not missing the cost attribute.

*** **Your turn**: Insert a new code chunk, then referring to the Wickham piece from today:

- Create a new column in the nyc_ppb_2017 object that is a logical vector indicating whether the Address attribute/column is missing
    - You will want to combine the **mutate** function introduced in the lead reading with the **is.na** function
    - Use the **View** function or the **str** function to verify that it worked
    - Celebrate!
- Experiment with subsetting using the nyc_ppb_2017 object (hint, see below for a jumpstart)

```
```{r label="Subset2017Dataset", message = FALSE, warning = FALSE}

nyc_ppb_2017[1:3, ]     # Subsets rows by position
nyc_ppb_2017[, 1:3]     # Subsets columns by position
nyc_ppb_2017[1:3, 1:3] # Subsets rows and columns by position

```
```

- As we saw on the previous page, the **filter** function is used to extract rows of interest. Experiment with the **select** function, which is used to extract columns of interest using the nyc_ppb_2017 object (hint, see below for a jumpstart)
    - Modify the code shown below to instead extract only Council District, Category, Title, Votes, Winner, Cost, Borough Code, Latitude, and Longitude
    - Can you pipe in the **arrange** function to sort the resulting object by

```
```{r label="SelectColumnsfromDataset", message = FALSE, warning = FALSE}

names(nyc_ppb_2017)

basics_nyc_ppb_2017 <- nyc_ppb_2017 %>%
  select(Category, Title, Address, `Borough Code`)

```
```

We are not covering other **dplyr** functions here like **summarize** or **count** because we saw them in the Marvel Comics example from last time. This is a soft introduction to **dplyr** and data wrangling in advance of the main event in Week 4.