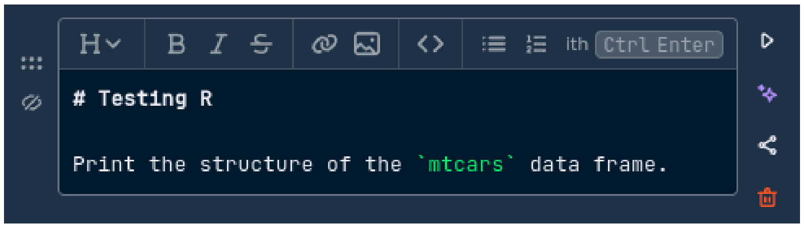
GETTING STARTED - COURSE INFRASTRUCTURE - DATALAB

Introduction to data science / DSC 105 / Fall 2024

Marcus Birkenkrahe

August 17, 2024

1. Go to your DataCamp dashboard.
2. You can get to `DataLab` from the dashboard selector but we're going to do something different: Open the "Introduction to R" course dashboard: select `Learn` and type `Introduction to R` in the search.
3. You can see that there are almost as many offerings in Python as in R - if you narrow it down, you can see that R has a lead on probability and stats, while Python leads for machine learning. Narrow the search to `R` and find the `Introduction to R` course (assigned to you later in this class).
4. In the course dashboard, find the `Create Course Notes` under `Resources` in the right sidebar and click on `+`.
5. You are now in the `DataLab` application. This is an interactive notebook like Emacs + Org-mode earlier. It's got some advanatages and some disadvantages. Open a `New Workbook`.
6. Though you came from an R course, these notebooks are set to Python by default. Change that in the `Environment` tab by selecting `R` as `Workbook language`, and run `Start session` to restart the environment.
7. In the workbook editor, you can now see that `R` instead of `Python` is offered, alongside `Text`, `SQL` and `Chart`. Click on `Text` and then repeat the commands that you used in Emacs before (almost) to the letter: this text is `Markdown` - headlines are started with `#`, and code font is obtained with `` ` `` (backtick) rather than `=`.



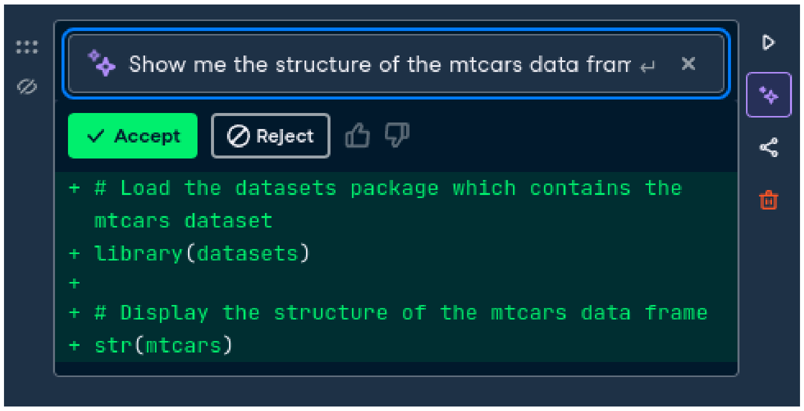
```
# Testing R

Print the structure of the `mtcars` data frame.
```

8. Now, instead of writing code in the text, you need to add an `R` block with `+`: You have the choice to add code or `tell our AI what to do`. Why don't we try that! At the prompt, enter

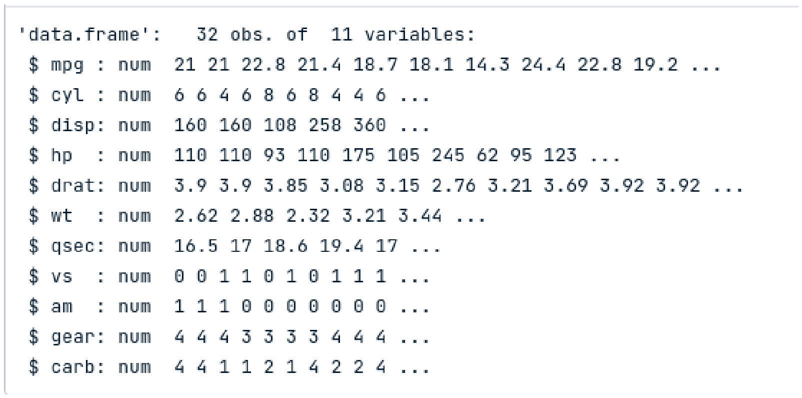
   `Show me the structure of the mtcars data frame.`

9. This is what you get - if it's correct (as shown) then click `Accept`.



```
+ # Load the datasets package which contains the
  mtcars dataset
+ library(datasets)
+
+ # Display the structure of the mtcars data frame
+ str(mtcars)
```

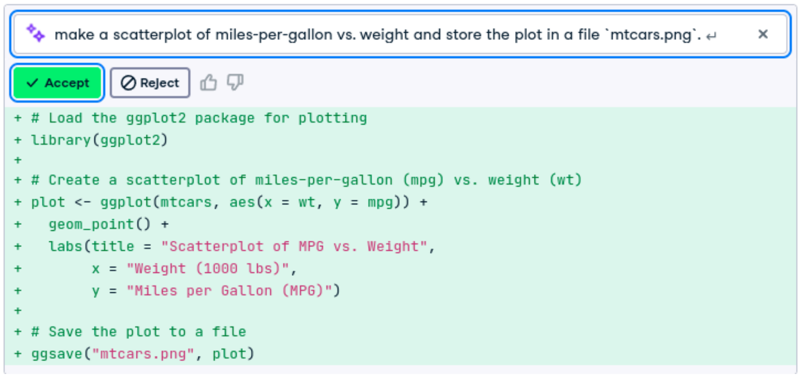
10. You now have a code block and you can cancel the AI prompt entry field and run the code by clicking on the play button (or typing `CTRL + RET` on the keyboard). You should get the same result as in Emacs.



```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

11. Let's move on with the plot: put the following into a text box, and then repeat it at the AI prompt:

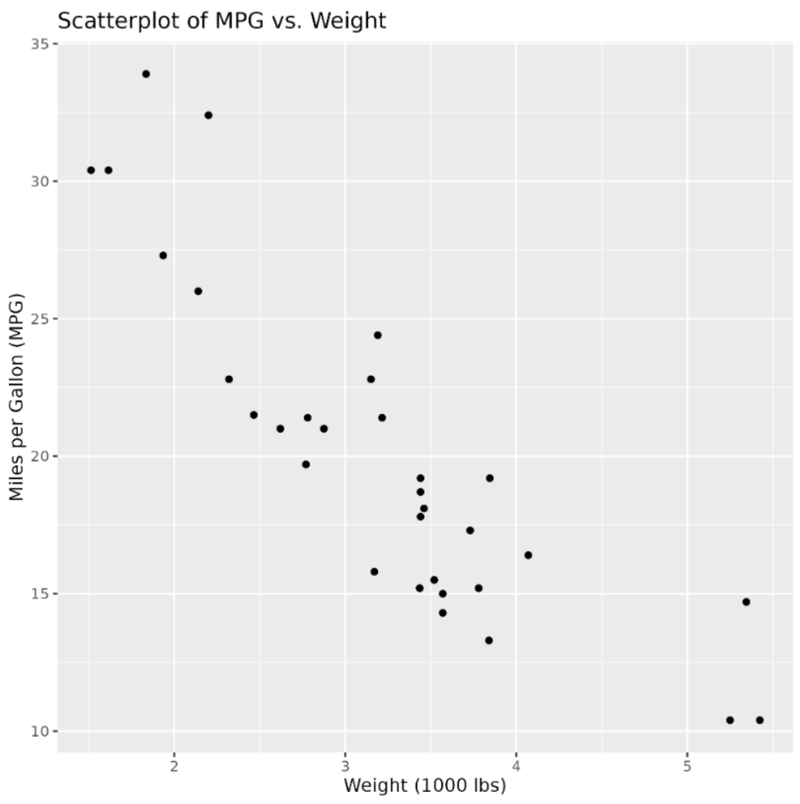
```
Make a scatterplot of miles-per-gallon vs. weight
and store the plot in a file ‘mtcars.png‘.
```



make a scatterplot of miles-per-gallon vs. weight and store the plot in a file `mtcars.png`. ↵
Accept
Reject

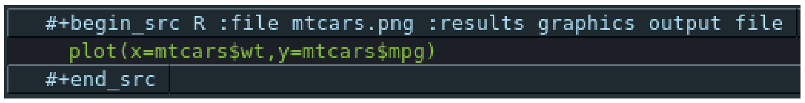
```
+ # Load the ggplot2 package for plotting
+ library(ggplot2)
+
+ # Create a scatterplot of miles-per-gallon (mpg) vs. weight (wt)
+ plot <- ggplot(mtcars, aes(x = wt, y = mpg)) +
+   geom_point() +
+   labs(title = "Scatterplot of MPG vs. Weight",
+        x = "Weight (1000 lbs)",
+        y = "Miles per Gallon (MPG)")
+
+ # Save the plot to a file
+ ggsave("mtcars.png", plot)
```

Because you saved the result to a file, it won't show in the notebook. You need to open it using the tab `File > Show Workbook Files`:



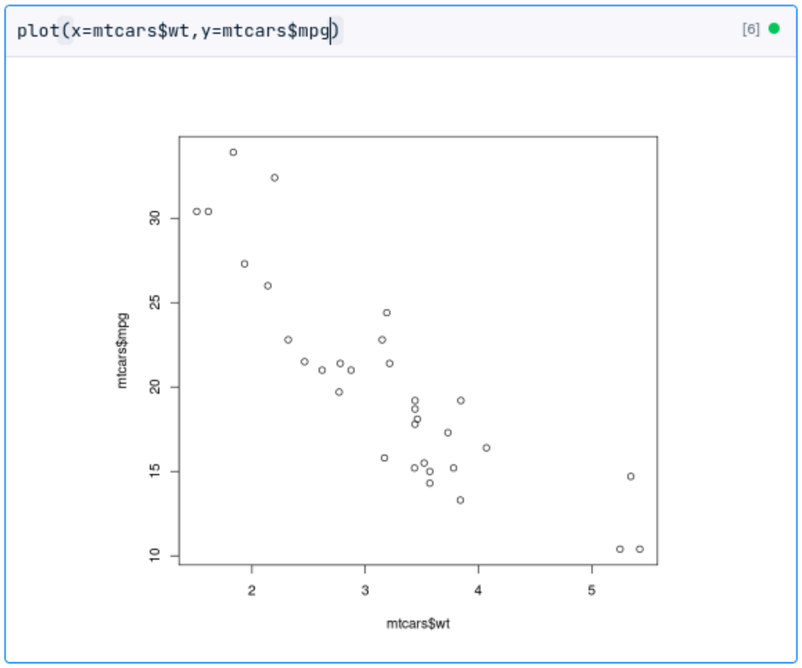
You recognize the scatterplot from Emacs though the code is vastly more complicated than necessary, leading to a customized plot.

12. The AI result is markedly different from what we did in Emacs (below). Let's see what DataLab does when we use this command:



```
#+begin_src R :file mtcars.png :results graphics output file
  plot(x=mtcars$wt,y=mtcars$mpg)
#+end_src
```

Add a new code block, enter the one-line command and run the block - you will notice that the R console suggests completions to you, e.g. when you type `plot(x==mtcars$`, all the possible columns of `mtcars` are suggested to you.



```
plot(x=mtcars$wt,y=mtcars$mpg)
```

This time, the plot is shown in the notebook itself just like in Emacs, which both stores the plot and shows it.

13. Before leaving, give your workbook a title: `Testing R` will be suggested when you click on the title field. This workbook will be permanent as long as you have a DataCamp account. Though if you aren't subscribed to my DataCamp classroom, you'll have a limited number of workbooks and a limited number of AI prompts available.

Issues with interactive notebooks

There are multiple problems with this approach:

- The interface changes often
- Having AI assistance at your fingertips (which you could also add to Emacs) is dangerous for beginners: you learn nothing or very little, and the code tends to be more complicated than necessary.
- You can only use one language per notebook at a time (R or Python), and only interpreted languages (not C,C++, Java, etc.).

- You rely on an Internet connection
- You can save and share your notebook only in the `.ipynb` (I-Python) format (that's plain-text but needs notebook software to be processed). You can also render Emacs-Org-mode notebooks in this format for upload to commercial notebook tools

Which tool to use when

When competing, you might have to use commercial tools (like Datalab, Jupyter, VSCode, RStudio, Colab) but whenever possible you should revert to FOSS tools (like Emacs + Org-mode). They're usually superior by design and offer much more flexibility and productivity gains.