Test

Dr. Chelsea Parlett-Pelleriti

Formatting

Header 1

Using a single # gives you a really large header!

Header 2

Using two # gives you a slightly smaller header!

Header 3

...and on

Header 4

...and on

Header 5

...and on

You may also want to use a bulleted list!

- just
- like
- this
- one

Or maybe an ordered list!

1. like
2. this
3. one

To emphasize your point you might want to use *italics* or **bold**.

To have something appear as code (using a monospace font), surround everything that is code with ticks ‘like this‘, so that it shows up `like this`.

If you’re the math-y type, you will be happy to know that you can write LaTeX in .qmd files as well! For a full line formula, use two $$ dollar signs at the begining and end of your LaTeX:

```
$$
\frac{\partial f}{\partial y} = x + 2y
$$
```

will appear as

$$\frac{\partial f}{\partial y} = x + 2y$$

For in-line LaTeX just use one $ in your text. For example, $x + \beta$

You may also want to put a link to a website for reference. Or, insert an image!

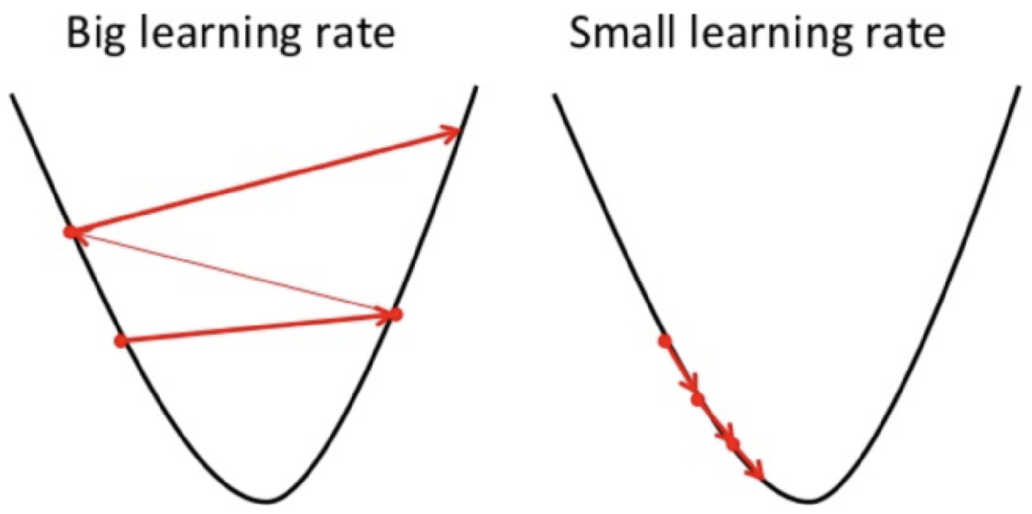

Figure 1: Caption

Code Blocks

One of the most useful things about markdown documents like this is that you can intersperse your writing with code!

For example, this Python code:

```python
import pandas as pd

df = pd.DataFrame({"a": [1,2], "b": [2,3]})
```

Notes

One important thing to remember is that when you render a .qmd file, all the code runs, so if your code does something computationally expensive you may not want to run it in your .qmd file.

One option is to use the `eval: false` argument when setting up your codeblock:

```python
print("Don't Run Me")
```

```python
print("Do Run Me")
```

```
Do Run Me
```

This is useful for showing someone what code you used without needing to show them the output or compute the code. For example you could show the way you set up your model without building or training it, and then load in a pre-trained model later to show them any output.

```python
# This code will not run it's just
# for show
import pandas as pd
import numpy as np
from plotnine import *

from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score #model evalu

# to save model
```

```
import pickle

df = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392ParlettPelle
                 sep = "\t")

df = df.dropna()

# set up X and y
predictors = ["List Price", "NumPages", "Weight (oz)", "Thick", "Height", "Width"]

X = df[predictors]
y = df["Amazon Price"]

# build model
m = LinearRegression()
m.fit(X,y)

# save trained model
with open("model.pkl", "wb") as f:
    pickle.dump(m,f)
```

The above code doesn't run, it's just to show people what you did.