Mango Pagedown Test

Mango The Cat

2019-07-23

1 Chapter Name 1
1.1 Headings 1
1.2 Fonts 1
1.2.1 Body 1
1.2.2 Titles 1
1.2.3 Numbering 1
1.2.4 Code 1
1.2.5 Equations 2
1.3 Plots 2
1.4 Exercises and info boxes 4
1.4.1 Exercises 4
1.4.2 Warnings 5
1.4.3 Info 5
1.5 Python 5

1 Chapter Name

1.1 Headings

Our training sections usually start at level 2.

1.2 Fonts

I've managed to change the fonts to Open Sans in `css/mango-fonts.css` that's about it.

1.2.1 Body

Open Sans Light

1.2.2 Titles

Might be OK Open Sans. We used to have level 3 as orange which is easy enough. The new font should be Din Regular I think. I've set this in the CSS but it might need to be installed. And even then it might default back to Open Sans. Need to watch this.

1.2.3 Numbering

Currently use `{chapter}.{section}.{sub-section}`.

Nice to haves would be a chapter front page saying Chapter `{Chapter Number}` `{Chapter Name}`.

1.2.4 Code

My guess is that the default code is OK but we might want to fancy it up a bit.

```
# A non-executed R-block

f <- function(x) {
  if (is.null(x)) 0 else x
}
```

where as python code might look like

```
# python code looks like this
def f(x):
    """Some big long docstring"""
    return(0 if x is null else x)
```

A code block that returns stuff looks like

```
f <- function(x) {
  if (is.null(x)) 0 else x
}

f(NULL)
```

```
## [1] 0
```

I wonder if that code output could be nicer. Not high priority.

1.2.5 Equations

Is it mathjax? I think so. So... It's double dollars for a big equation. I'm changing the font for math I think to Times then serif.

$$E = \gamma mc^2$$

And single dollars for inline $E = mc^2$ equations.

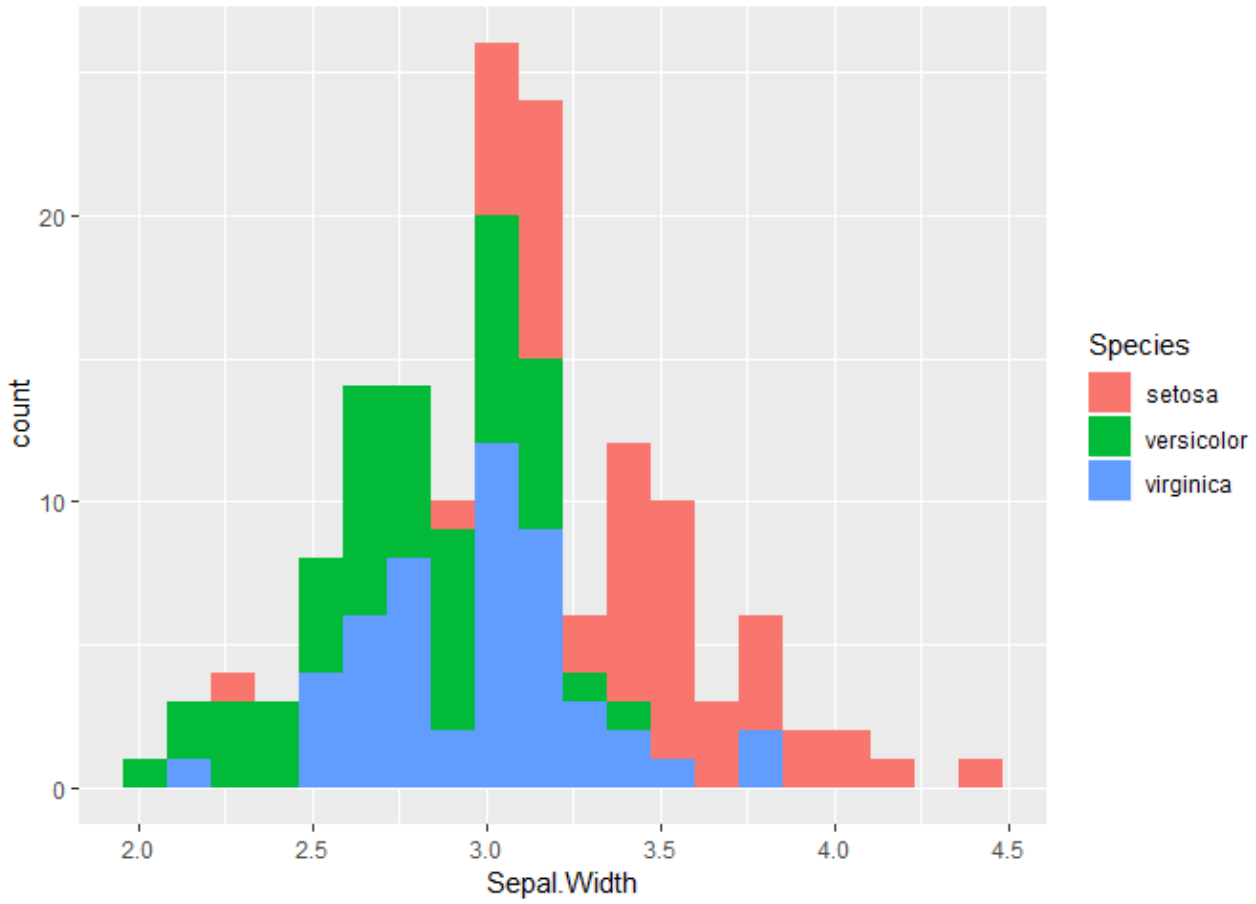
1.3 Plots

We do a fair amount of plots with code

```
library(ggplot2)
ggplot(iris) +
  geom_point(aes(x = Sepal.Width, y = Sepal.Length, colour = Species))
```

We can manually control size but should have nie defaults

We might do plots with suppressed code.



1.4 Exercises and info boxes

1.4.1 Exercises

We're going to need some way to make exercise boxes. This could be by applying a class to a section heading: `{.exercise}`

Exercise

It looks like you can use a section header and a class. The code here looks like:

```
#### Exercise {.exercisebox - }
```

And then this puts everything until the next section heading in a div with the exercise class. The dash is the part that suppresses the section label. Everything else can be styled with css.

1. Question 1
2. Question 2

And what about overrun?

1.4.2 Warnings

Watch out!

Some important warning message. Again, the dash removes the numbering. There is examples of this already in the existing code.

A warning with no title. You just get straight into it. It looks OK actually.

1.4.3 Info

Interesting info

When we're done can put icons and style it nicely.

1.5 Python

Does Python just work? Well it does. Mostly. I'm having some issues with plots too that needs working out.

```
import pandas as pd
df = pd.DataFrame({'a': [1,2,3], 'b':['a', 'b', 'c']})
print(df.head())
```

```
##    a  b
## 0  1  a
## 1  2  b
## 2  3  c
```

Looks OK. I don't love the syntax highlighting but that might be a Pandoc thing. And does it carry through?

```
df.shape
```

```
## (3, 2)
```

OK Good. What about plots?

For plots to work you must set an environment variable. Ideally set this up for your system rather than inside the notebook, or else it won't work elsewhere.

For me this looks like:

```
QT_QPA_PLATFORM_PLUGIN_PATH=C:/Users/dashton/AppData/Local/Continuum/anaconda3/Library/pl
```

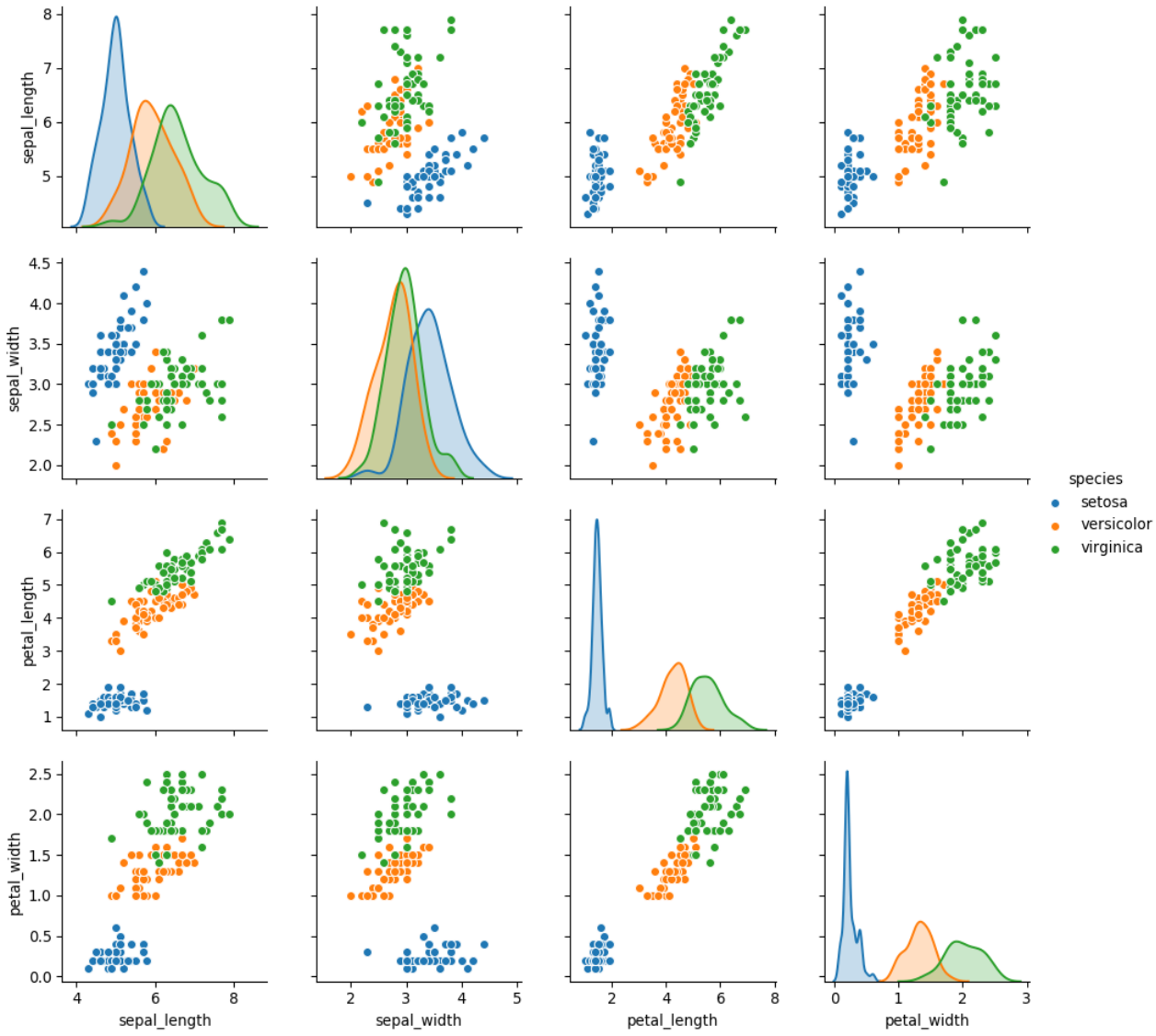
On a Windows machine I click on the start menu and type "environment" and it gives an option to "Edit the system environment variables". Add a new user variable with the above for your system.

On a Mac you probably put `export QT_QPA_PLATFORM_PLUGIN_PATH='/path/to/anaconda/plugins/platforms'` inside .bash_profile (untested).

```
import seaborn as sns
import matplotlib.pyplot as plt
iris = sns.load_dataset("iris")
sns.pairplot(iris, hue="species")
```

```
## <seaborn.axisgrid.PairGrid object at 0x0000023FADF8E208>
```

```
plt.show()
```

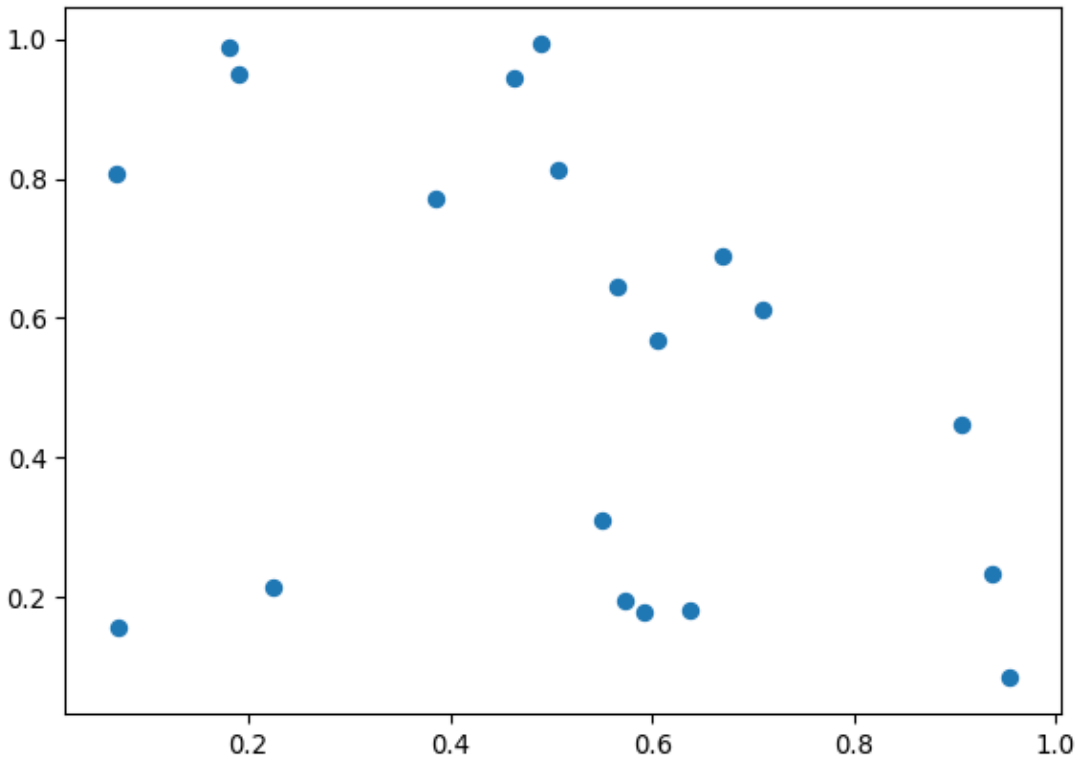


What about pure matplotlib

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.random(20)
y = np.random.random(20)
plt.scatter(x, y)
```

```
## <matplotlib.collections.PathCollection object at 0x0000023FAFC826D8>
```

```
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0