Imperial College London

UNIVERSITY OF OXFORD

28 May 2024

Configuration Files, Seeding, Logging and Jupyter Notebook Best Practices

Brendan Martin

Configuration Files

Examples of what not to do

```
import matplotlib.pyplot as plt
import numpy as np

def my_func(x,y):
    return 3*x - y

num_epochs = 200
result = 0
for i in range(num_epochs):
    result += my_func(i,3)
```

Defining parameter variables at top of script

```
import matplotlib.pyplot as plt
import numpy as np

###### HYPERPARAMETERS ######
num_epochs = 200
y_constant = 3

def my_func(x,y):
    return 3*x - y

result = 0
for i in range(num_epochs):
    result += my_func(i,y_constant)
```

Using a configuration file

```yaml
# Example YAML configuration file
server:
  address: 127.0.0.1
  port: 8080

database:
  type: mysql
  host: localhost
  port: 3306
  username: root
  password: example

logging:
  level: INFO
  file: /var/log/app.log
```

Using a configuration file

```
###### PARAMETERS #######
SEED : 4436
Q : 4
n_backtest_days : 4275
first_prediction_day : 1004
target_feature : 1
num_lags : 1
```

Reading a yaml file into a python script

```python
import yaml
import sys

# Load the YAML configuration file
with open(sys.argv[1], 'r') as file:
    config = yaml.safe_load(file)

# Access the configuration data
server_address = config['server']['address']
server_port = config['server']['port']
database_type = config['database']['type']
database_host = config['database']['host']
database_port = config['database']['port']
database_username = config['database']['username']
database_password = config['database']['password']
logging_level = config['logging']['level']
logging_file = config['logging']['file']
```

```
$ python your_script.py --config /path/to/config.yaml
```

Why should you use a YAML config file?

- Easy to read
- Language agnostic - the same config file can be passed to a python and R script, for example
- YAML supports many different data types: strings, floats, booleans, lists, arrays, dictionaries, hashes
- You can have different config files for different environments (development, testing, production) without changing the code
- Collaboration: Easy for people who are not familiar with your specific language/code to interpret your hyperparameters
- Industry standard for “Infrastructure as Code” - e.g. Ansible, Kubernetes, Docker

Reading a YAML file into an R script

```r
library(yaml)

# Load the YAML configuration file
config <- yaml.load_file('config.yaml')

# Access the configuration data
server_address <- config$server$address
server_port <- config$server$port
database_type <- config$database$type
database_host <- config$database$host
database_port <- config$database$port
database_username <- config$database$username
database_password <- config$database$password
logging_level <- config$logging$level
logging_file <- config$logging$file

```

Passing parameters as optional arguments

```python
import argparse

# Create the parser
parser = argparse.ArgumentParser(description=
                    'Example script with optional arguments.')

# Add arguments
parser.add_argument('--config',
                    type=str,
                    help='Path to the configuration YAML file.')
parser.add_argument('--verbose',
                    action='store_true',
                    help='Enable verbose output.')

# Parse the arguments
args = parser.parse_args()

# Access the arguments
config_path = args.config
verbose = args.verbose
```

Passing parameters as optional arguments

```
$ python your_script.py --config /path/to/config.yaml
```

Logging

Overview

What is it?
The practice of recording and storing information about the execution of a software application.

What kinds of information?

- Errors and Exceptions
- System Events: startup, shutdown, user logins
- Debugging Information
- Performance Metrics

Best Practices: Log Levels

1. DEBUG
2. INFO
3. WARNING
4. ERROR
5. CRITICAL

Example Log Output File

```
2024-05-28 10:30:00,001 INFO [example_script] Script started
2024-05-28 10:30:00,002 DEBUG [example_script] Attempting to open a file
2024-05-28 10:30:00,003 ERROR [example_script] File not found: [Errno 2] 'file.txt'
2024-05-28 10:30:00,004 DEBUG [example_script] Attempting to divide by zero
2024-05-28 10:30:00,005 CRITICAL [example_script] Critical error: division by zero
2024-05-28 10:30:00,006 INFO [example_script] Script finished
```

Using the logging module in python

```python
import logging

# Configure logging to output all logs to a file
logging.basicConfig(
    filename='log_output.txt',  # Log output file
    level=logging.DEBUG,        # Log level = DEBUG
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
)

# Create a logger
logger = logging.getLogger('example_logger')

# Log messages of various severity levels
logger.debug('This is a debug message')
logger.info('This is an info message')
logger.warning('This is a warning message')
logger.error('This is an error message')
logger.critical('This is a critical message')

```

Seeding

Why use seeding?

- Reproducibility
- Comparing different algorithms on the same random data
- Testing often requires the same output every time the test is run

Numpy Random Generator

```python
import numpy as np

##### PARAMETERS #####
SEED = 42

# Create a Generator instance with a specific seed
rng = np.random.default_rng(SEED)

# Generate random numbers
print(rng.random())              # 0.7739560485559633
print(rng.integers(1, 100))      # 91
print(rng.uniform(1.5, 10.5))    # 7.045096283947272

```

THANK YOU!

References

Software Carpentry. `https://software-carpentry.org/lessons/index.html`. Accessed: 2024-05-24.