PROJECT

⇧ | ⇩ | ⇨

Metadata

Key	Value
author	AUTHOR
date	DATE
dataset	`AmesHousing.txt` © 2015 American Statistical Association
url	AmesHousing.txt, save as CSV

- Metadata

Abstract

⇦ | ⇧ | ⇩ | ⇨

Explain the executive summary and findings. You HAVE TO. this is the first thing they see.

TOC

⇦ | ⇧ | ⇩ | ⇨

- PROJECT
    - Metadata
- Abstract: executive summary of this file.
- TOC: what it says on the tin.
- Solution: is the "justification" for providing the answer above.
    - Code - Setup: is ignorable boilerplate code that sets up stdlib, third party, constants, and seeds, acts as a safety valve for dependency checking.
- Appendix: back of the project
    - Instructions: grading criterion for the project if known in advance.
    - Changelog: proof of work.

Solution

⇦ | ⇧ | ⇩ | ⇨

Here's some inlinelatex to worry about

- Code - Setup

Code - Setup

⇦ | ⇧ | ⇩ | ⇨

The following is boilerplate I've accumulated over the years and can be safely disregarded by the reader until it becomes relevant as part of deeper inquiry.

In [1]:
```
# stdlib imports
import __main__
import os
import sys
import pprint
import importlib
import subprocess
# hyper-critical imports
try:
    from IPython import get_ipython
    IPYTHON_SHELL = get_ipython()  # None if running in vanilla python
except ImportError:
    IPYTHON_SHELL = None

SCRIPT_FILEPATH = ''
SCRIPT_FILENAME = ''
if IPYTHON_SHELL:
    SCRIPT_DIRPATH = str(IPYTHON_SHELL.run_line_magic('pwd', ' # %pwd is a "magic" command  https
    if hasattr(__main__, '__vsc_ipynb_file__'):  # vscode
        SCRIPT_FILEPATH = __main__.__vsc_ipynb_file__
    elif hasattr(__main__, '__session__'):  # localhost:8888/notebooks
        SCRIPT_FILEPATH = __main__.__session__
else:
    SCRIPT_FILEPATH = __file__
    SCRIPT_DIRPATH = os.path.dirname(SCRIPT_FILEPATH)

SCRIPT_FILENAME = os.path.splitext(os.path.basename(SCRIPT_FILEPATH))[0]

pprint.pprint(dict(dir=SCRIPT_DIRPATH, fp=SCRIPT_FILEPATH, fn=SCRIPT_FILENAME), indent=2)

AUTO_INSTALL = False
packages = {}  # {'Werkzeug': 'werkzeug'}  # package name: module name
modules = ['matplotlib', 'pandas', 'ipympl', 'ipywidgets']
packages.update({m: m for m in modules})
package, module = '', ''
install_string = ' '.join(packages)
try:
    for package, module in packages.items():
        importlib.import_module(module)
except ImportError:
    print(f'"pip install {package}"', file=sys.stderr)
    if AUTO_INSTALL:
        %pip install {install_string}
    else:
        raise

# third party imports
import matplotlib.pyplot as plt

EXES = {
    'pandoc': {'exists': False},
    'latex': {'exists': False, 'win32': 'miktex', 'linux': 'texlive'},
}

for exe, data in EXES.items():
    if IPYTHON_SHELL:
        if sys.platform == 'win32':
            _res = !where.exe pandoc
            _exists = not _res[0].startswith('INFO')
        else:
            _res = !which pandoc
            _exists = bool(_res)
    else:
        if sys.platform == 'win32':
            _res = subprocess.check_output('where.exe pandoc', universal_newlines=True)
            _exists = _res.startswith('INFO')
        else:
            _res = subprocess.check_output('which pandoc', universal_newlines=True)
            _exists = bool(_res)
    EXES[exe]['exists'] = _exists

print('pandoc:', EXES['pandoc']['exists'])
if not EXES['pandoc']['exists']:
    print('you must install pandoc!', file=sys.stderr)
    if sys.platform == 'win32':
        print('    choco install pandoc -y', file=sys.stderr)
    else:
        print('    sudo apt install pandoc -y', file=sys.stderr)
        print('    sudo yum install pandoc -y', file=sys.stderr)
        print('    dnf install pandoc -y', file=sys.stderr)
    raise RuntimeError()

print('latex:', EXES['latex']['exists'])
plt.rcParams.update({'text.usetex': EXES['latex']['exists']})  # , 'font.family': 'Helvetica'  #

# command to tell the notebook to plt.show() IN THE NOTEBOOK, otherwise you call plt.show()
%matplotlib inline
# command to tell the notebook to render interactable matplotlib, requires pip install ipympl
# %matplotlib widget

print('# pip install', install_string)
if not IPYTHON_SHELL:
    raise RuntimeError('Not sure how, but somehow this code is being run through a non-iPython en
```

Appendix

⇦ | ⇧ | ⇩ | ⇨

- Instructions
- Changelog

Instructions

⇦ | ⇧ | ⇩ | ⇨

Changelog

- 2026-01-29 10:00 - initial commit

⇧ | ⇩