Jupyter Notebooks und LaTeX

Ulrich Reus

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument soll anhand einiger Beispiele die Möglichkeiten von Jupyter-Notebooks aufzeigen, Literate-Programming zu realisieren. Beim Literate-Programming wird der Quellcode und die dazu gehörende Dokumentation integriert in einem Dokument entwickelt und gepflegt. Der prinzipielle Ablauf ist folgender:

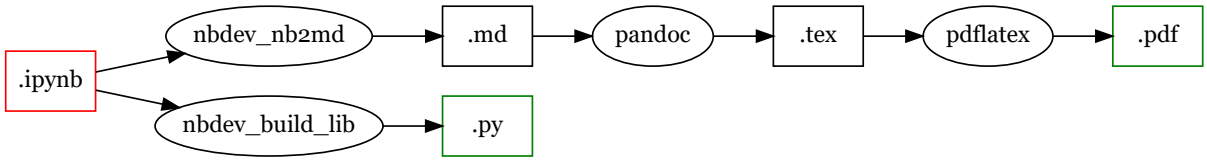

Abbildung 1: Ablauf Jupyter-Notebook nach PDF bzw. .PY

D.h. aus dem Jupyter-Notebook (.ipynb) wird im ersten Prozess über die Zwischenschritte Markdown und LaTeX am Ende ein PDF-Dokument. Im zweiten Prozess wird aus dem selben Jupyter-Notebook der lauffähige Python-Code erzeugt. Ermöglicht wird dies durch den Einsatz des Frameworks nbdev. nbdev ist in Python geschrieben, genauso wie die Datenanalyse-Software, die Sie im Rahmen dieser Veranstaltung erstellen sollen. Falls Sie eine Einführung in Python benötigen, so wäre z.B. das Buch von Kofler[1] geeignet. Einen fundierten und unterhaltsamen Überblick über sog. NoSQL-Datenbanken bietet ein Vortrag von Fowler[2], der per Youtube verfügbar ist.

2. Jupyter Notebook

Jupyter Notebooks sind im Grunde JSON-Dateien, die mit Hilfe der Web-Anwendung Jupyter-notebook oder Jupyter-lab editiert werden. Ein Notebook besteht aus einer Abfolge von Zellen, die entweder Markdown-Text, (Python-)Quellcode oder Rohdaten enthalten. Diese Zellen können beliebig oft und in beliebiger Reihenfolge ausgeführt werden. Auf diese Weise können Code und Dokumentation in einem Dokument gepflegt und gespeichert werden. Das Framework `nbdev`[3] unterstützt dieses Vorgehen durch die Bereitstellung eines Vorgehensmodells und den dazu passenden Werkzeugen.

[1] [@kofler_python_2018]
[2] [@fowler_introduction_2013]
[3] [@fastai_welcome_2021]

3. Markdown

In diesem Notebook sind verschiedene Beispiele für die Möglichkeiten zur Dokumentation von Programmcode zusammengestellt, die bei der Konvertierung des Notebooks nach LaTeX (und dann nach PDF) verwendet werden können. Im Prinzip steht die komplette Markdown-Syntax zur Verfügung, die von Jupyter-Notebooks unterstützt wird. Über die hier gezeigten Beispiele könnte dieses Cheat-Sheet helfen.

- Listen (dies ist selbst eine solche)
- internal references
- Mathematische Formeln
- weitere Blöcke
  - Bilder
  - Tabellen
  - Quellcode
  - Plots
- Verweise und Fußnoten
- Verschiedene Hinweisblöcke

3.1. Listen

Nicht nummerierte Liste siehe oben.

3.1.1. Nummerierte Liste

1. Eins
2. Zwei
   1. Zwei Unterpunkt eins
   2. Zwei Unterpunkt zwei
3. Drei

3.2. Bilder

Ein einfaches Beipielbild von der Festplatte. Übliche Dateitypen sind .svg, .jpg und .png:

Abbildung 2: Ein Zahnwal

Graphviz ist ein Werkzeug, um Graphen zu erzeugen. Auch PlantUML basiert auf Graphviz und ist in Jupyter-Notebooks einsetzbar.

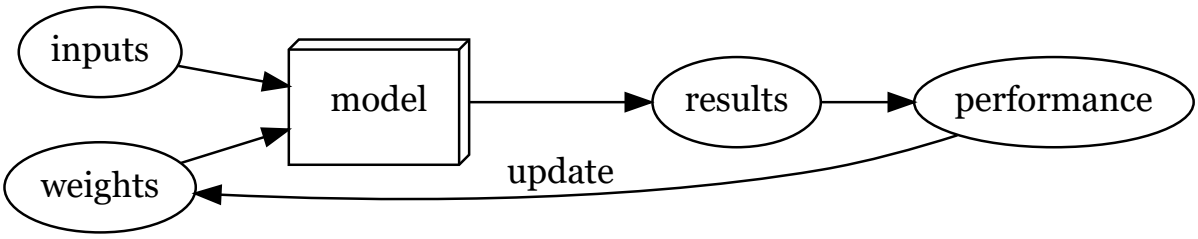

Abbildung 3: svg

Wie hier zu sehen, werden von `nbdev2md` keine korrekten Bildunterschriften erzeugt. Daher sollte ein per graphviz erzeugtes Bild manuell eingebunden werden. Eventuell wird zu einem späteren Zeitpunkt eine entsprechende Funktionalität zur Verfügugng gestellt.

3.3. Tabellen

Für die Erzeugung von Tabellen gibt es verschiedene Möglichkeiten in Markdown selbst, aber auch mit LaTeX. Hier einige Beispiele:

3.3.1. Markdown

Tabelle 1: Preise und Vorteile diverser Früchte

Fruit	Price	Advantages
Bananas	$1.34	• built-in wrapper • bright color
Oranges	$2.10	• cures scurvy • tasty

Die obige Tabelle wird im Jupyter-Notebook nicht korrekt angezeigt, im PDF-Dokument aber sehr wohl, wenn vor und nach der Tabelle Leerzeilen eingefügt werden.

Bei der unteren Tabellenform funktioiert beides:

Tabelle 2: Dies ist die Überschrift zur zweiten Tabelle. Wenn die Überschrift sehr lang sein sollte, kann sie sich auch über mehr als eine Zeile erstrecken.

Stretch/Untouched	ProbDistribution	Accuracy
linksbündig	zentriert	.843

3.3.2. LaTeX

Es ist auch möglich Tabellen und andere Elemente in LaTeX-Code einzufügen.

Age	Frequency
18–25	15
26–35	33
36–45	22

Hier das Beispiel einer Formel:

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$$

Die Formel selbst sollte in einer eigenen “Raw”-Zelle stehen und in dieser oberhalb und unterhalb von einer Leerzeile umgeben sein. Dies ist erforderlich, um mögliche Probleme bei der Konvertierung zu vermeiden.

4. nbdev

Für die nbdev-Werkzeuge werden die einzelnen Code-Zellen mit speziellen Kommenaren versehen, um nvdev zu signalisieren, wie der jeweilige Code-Block zu verarbeiten ist. Um zu sehen, was die unterschiedlichen Kommentare bewirken, sehen Sie sich bitte die aus diesem Notebook generierte PDF-Datei sowie die generierte Datei examples.py im Verzeichnis `nb2ltx` an.

- `#export`: Zelle wird nicht angezeigt; Code wird von `nbdev` in die Moduldatei exportiert
- `#exports`: Zelle und Ausgabe der Zelle werden angezeigt; Code wird von nbdev in die Moduldatei exportiert

```
# exports
def wirdAngezeigt():
    return 'Sie werden exportiert und angezeigt!'

print('Hi')
```

```
Hi
```

- `#hide`: Von der folgenden Zelle wird weder der Quellcode noch die Ausgabe angezeigt
- `#hide_input`: Von dieser Zelle wird nicht der Quellcode, sondern nur die Ausgabe angezeigt:

133.33333333333331

```
print("Aufruf der Funktion 'verstecken()':")
verstecken()
```

```
Aufruf der Funktion 'verstecken()':



'Wir verstecken Sie...'
```

- #sonstiger Kommentar oder kein Kommentar: Zelle und auch das Ergebnis werden angezeigt

```
# dies ist ein Test...
wirdAngezeigt()
```

```
'Sie werden exportiert und angezeigt!'
```

```
import sys
print('Zelle ohne Kommentar')
sys.getdefaultencoding()
```

```
Zelle ohne Kommentar



'utf-8'
```

4.1. Markdown Tests

4.1.1. Umlaute

... sind möglich! Umlauttest: öääüÄÜÖß

5. Beispielhafte Modulverwendung

```
1 import nb2ltx.klauswert as klw
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

```
1 /Users/uli/.pyenv/versions/3.7.9/envs/nb2ltx/lib/python3.7/
     site-packages/pandas/compat/__init__.py:97: UserWarning:
     Could not import the lzma module. Your installed Python is
      incomplete. Attempting to use lzma compression will
     result in a RuntimeError.
2   warnings.warn(msg)
```

5.1. Daten laden

```
1 pbx2020 = klw.Klauswert('rawData/KlausurPunkte.xlsx')
```

5.2. Notenliste erzeugen

Erst die Daten erzeugen und dann ein Histogramm zeichnen.

```
1 #hide_output
2 nListe = pbx2020.getNoten()
3 print(nListe.head().to_markdown())
```

	Name	Note
0	Alexander	3
1	Andre	3.3
2	Celine	3
3	Christian	3.7
4	Christopher	5

```
1 notenBuckets = pbx2020.getNotenDefinition()['Note']
2 notenBuckets
```

```
0      1.0
1      1.3
2      1.7
3      2.0
4      2.3
5      2.7
6      3.0
7      3.3
8      3.7
9      4.0
10     5.0
Name: Note, dtype: float64
```

```
noList = []
for note in notenBuckets:
    noList.append({'Note': note, 'Anzahl': nListe[nListe['
       Note'] == note].count()[0]})

notenVerteilung = pd.DataFrame(noList)
display(Markdown(notenVerteilung.to_markdown()))
```

	Note	Anzahl
0	1	0
1	1.3	0
2	1.7	2
3	2	0
4	2.3	3
5	2.7	0
6	3	8
7	3.3	2
8	3.7	8
9	4	8
10	5	6

5.3. Graphische Darstellung der Notenverteilung

```
1 fig = plt.figure()
2 ax = fig.add_axes([0,0,1,1])
3 ax.bar(notenVerteilung['Note'], notenVerteilung['Anzahl'],
     0.2)
4 plt.xticks(notenVerteilung['Note'])
5 plt.show()
6 fig.get_size_inches()
```

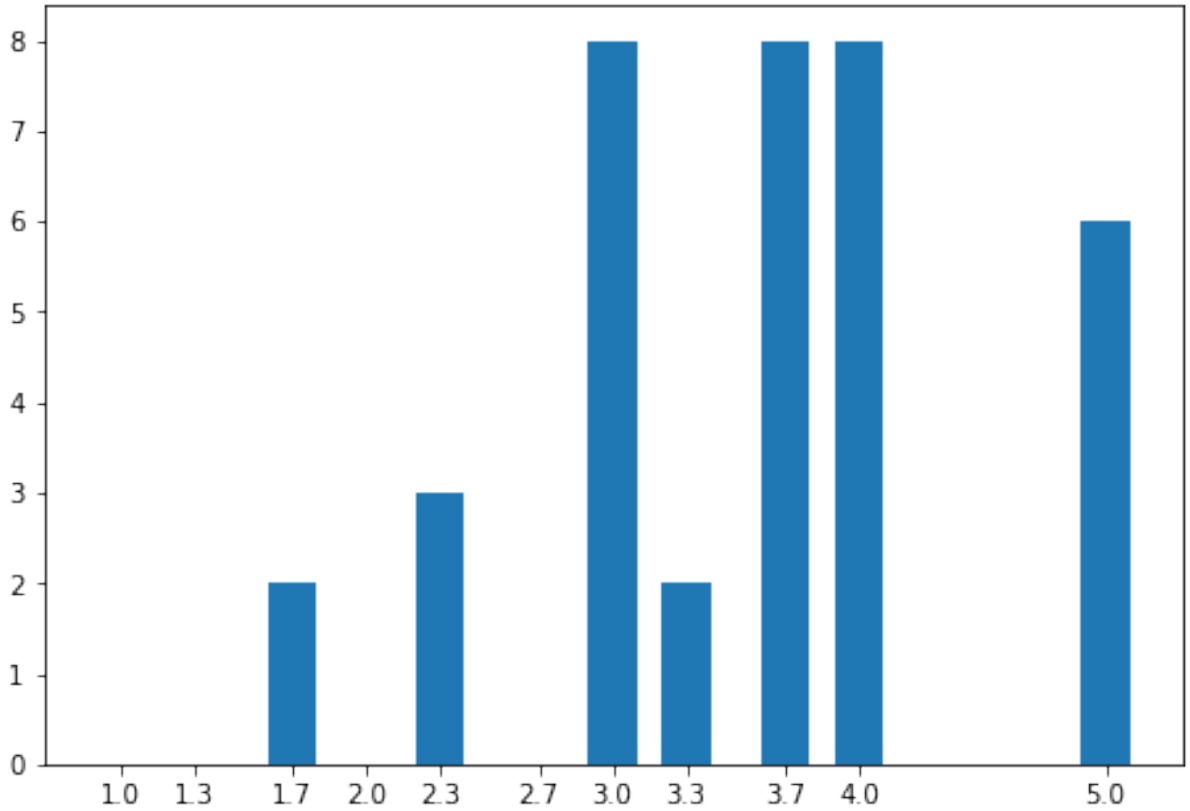

Abbildung 4: png

```
1 array([6., 4.])
```

Es geht auch etwas kleiner:

```
1 fig = plt.figure(figsize=(4, 2.67), dpi=120)
2 ax = fig.add_axes([0,0,1,1])
```

```
3 ax.bar(notenVerteilung['Note'], notenVerteilung['Anzahl'],
     0.2)
4 plt.xticks(notenVerteilung['Note'])
5 plt.show()
6 fig.get_size_inches()*fig.dpi
```

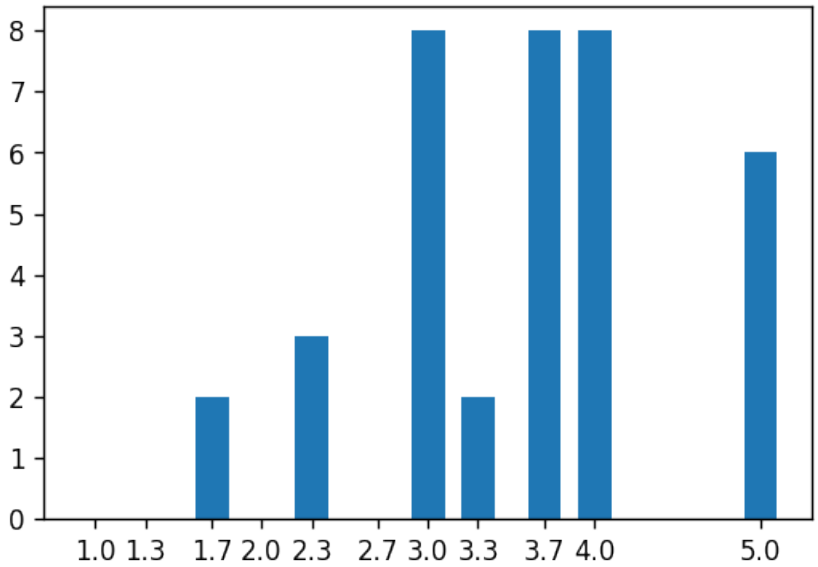

Abbildung 5: png

```
1 array([480. , 320.4])
```

ï»¿

Anhang

A. klauswert

> Ein Modul zur Klausurauswertung

A.1. Module einbinden

```
# exports
import pandas as pd
import yaml
```

A.2. Klasse “Klauswert”

A.2.1. Standard Konfiguration

```
# exports
stdConfig = '''headerPC: 4
pointCols: "A:L"
headerMC: 1
markCols: "N:O"
headerMP: 1
mxPointCols:  "A:L"
'''
```

```
pbx2020 = Klauswert('rawData/KlausurPunkte.xlsx')
```

```
my_show_doc(Klauswert)
```

class Klauswert

> Klauswert(**fname**, **configTxt**=*'headerPC: 4\npointCols: "A:L"\nheaderMC: 1\nmarkCols: "N:O"\nheaderMP: 1\nmxPointCols: "A:L"\n'*)

Aufgabe der Klasse: Klausurauswertung
Konfiguration: YAML-Text (stdConfig)
Ausgangslage: .xlsx-Datei mit einer Zeile je Student. In den Spalten

config['pointCols'] stehen die Punkte für die jeweiligen Aufgaben. In den Spalten config['markCols'] ist die Notenverteilung ersichtlich

```
pbx2020 = Klauswert('rawData/KlausurPunkte.xlsx')
print(pbx2020.getConfig()['headerMC'])
```

```
1
```

A.2.2. Die Funktionen

```
my_show_doc(Klauswert.getAllData)
```

Klauswert.getAllData

> Klauswert.getAllData()

Aufgabe: gesamten Dataframe zurück geben.

```
display(Markdown(pbx2020.getAllData().head().to_markdown()))
```

	Name	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	sumP
0	Alexander	4	6	4	7	11	4	4	5	5	4.5	10.5	65
1	Andre	4	3.5	4	8	20.5	4	3	3.5	3.5	3	4	61
2	Celine	4	3.5	2	7	13.5	5	5	5	4.5	5	11	65.5
3	Christian	3.5	2.5	1	6.5	17	2.5	5	5	4	6	4	57
4	Christopher	3.5	0.5	0	4	7.5	0	0	0	0	0	0	15.5

Die folgende Tabelle ist eine Kopie der Codeausgabe der vorherigen Zelle:

	Name	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	sumP
0	Alexander	4	6	4	7	11	4	4	5	5	4.5	10.5	65
1	Andre	4	3.5	4	8	20.5	4	3	3.5	3.5	3	4	61
2	Celine	4	3.5	2	7	13.5	5	5	5	4.5	5	11	65.5

	Name	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	sumP
3	Christian	3.5	2.5	1	6.5	17	2.5	5	5	4	6	4	57
4	Christopher	3.5	0.5	0	4	7.5	0	0	0	0	0	0	15.5

```
dfKerg=pd.read_excel('rawData/KlausurPunkte.xlsx', header=4,
    usecols='A:L')
```

```
my_dispMarkdown(dfKerg.head())
```

	Name	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
0	Alexander	4	6	4	7	11	4	4	5	5	4.5	10.5
1	Andre	4	3.5	4	8	20.5	4	3	3.5	3.5	3	4
2	Celine	4	3.5	2	7	13.5	5	5	5	4.5	5	11
3	Christian	3.5	2.5	1	6.5	17	2.5	5	5	4	6	4
4	Christopher	3.5	0.5	0	4	7.5	0	0	0	0	0	0

```
my_show_doc(Klauswert.getDurchgefallen)
```

Klauswert.getDurchgefallen

> Klauswert.getDurchgefallen()

Liefert eine Liste mit den Namen der Teilnehmer, die die Klausur nicht bestanden haben, sowie deren Zielerreichungswert.

```
my_dispMarkdown(pbx2020.getDurchgefallen().head())
```

	Name	pktProz	Note
4	Christopher	17.2222	5
15	Kamil	25	5
20	Lukas	45	5

	Name	pktProz	Note
34	Tim	47.2222	5
35	Tobias	44.4444	5

	Name	pktProz	Note
4	Christopher	17.2222	5
15	Kamil	25	5
20	Lukas	45	5
34	Tim	47.2222	5
35	Tobias	44.4444	5