

Interaktive Präsentationen mit Jupyter Notebook in der Statistiklehre

Tobias Bernstein, Thomas Hotz und Friedrich M. Philipp

Fachgebiet für Wahrscheinlichkeitsrechnung und mathematische Statistik
Institut für Mathematik
TU Ilmenau

Herbsttagung des AK Stochastik, 26 Oktober 2025

gefördert durch das BMFTR im Rahmen von **THInKI** – Thüringer Hochschulinitiative für KI im Studium

```
In [1]: import numpy as np # numerics
import scipy.stats as st # probability
import matplotlib.pyplot as plt # plots
import matplotlib as mpl # more plots
import pandas as pd # data handling
import statsmodels.api as sm # statistics
import statsmodels.formula.api as smf # statistical models
import ipywidgets as widgets # interactive
from IPython.display import display
from IPython.display import clear_output
import os.path # files

mpl.rcParams['figure.dpi'] = 150

def hide():
    from IPython import display
    import binascii
    import os
    uid = binascii.hexlify(os.urandom(8)).decode()
    html = """<div id="%s"></div>
    <script type="text/javascript">
        $(function(){
            var p = $("#%s");
            if (p.length==0) return;
            while (!p.hasClass("cell")) {
                p=p.parent();
                if (p.prop("tagName") == "body") return;
            }
            var cell = p;
            cell.find(".input").addClass("notvisible")
        });
    </script>""" % (uid, uid)
    display.display_html(html, raw=True)

from html.parser import HTMLParser
class NBParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        self.attr = {attr[0] : attr[1] for attr in attrs}

p = NBParser()
p.feed('<rise slide="fragment"/>')
p.attr['slide']
```

Out[1]: 'fragment'

Welches Problem lösen Jupyter Notebooks?

Verwendung unterschiedlicher Werkzeuge für ...

- theoretische Abschnitte in Skript/Präsentation
- praktische Beispiele in den Lehrveranstaltungen
- Übungen

**Notebooks können Text- und Codebestandteile enthalten;
erlauben Lehre aus einem Guss!**

attraktive Darstellung im Präsentationsmodus RISE

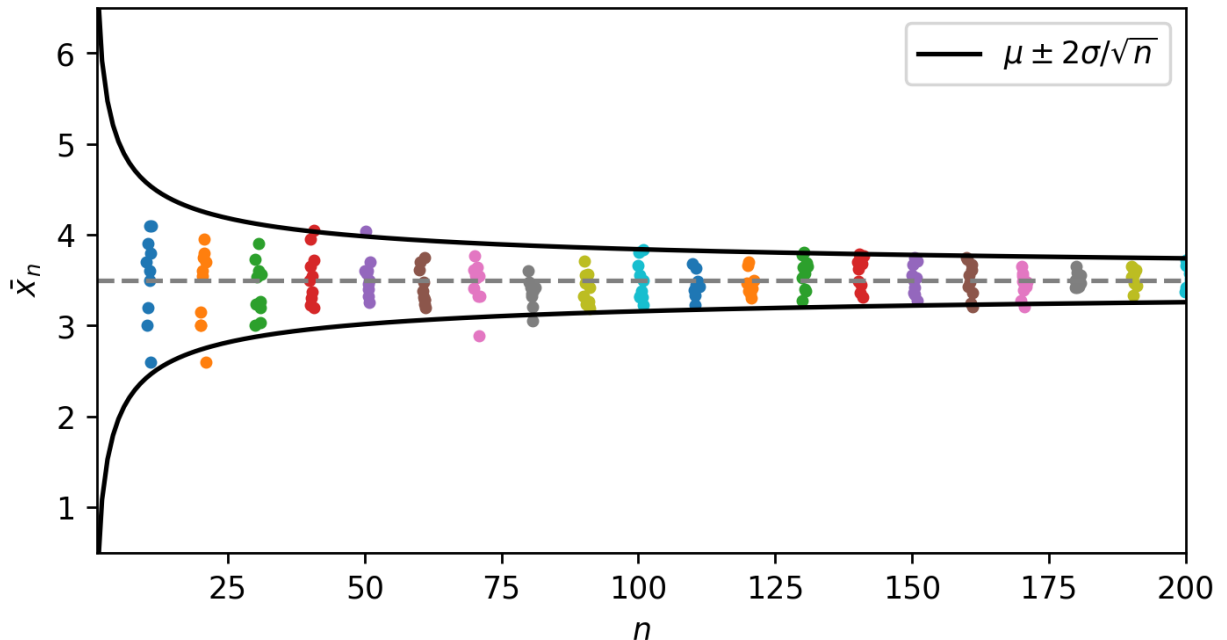
Satz (Gesetz der großen Zahlen im quadratischen Mittel): Seien $X_1, \dots, X_n \in \mathcal{L}^2(\Omega, \mathbf{P})$ unabhängig und identisch verteilte, quadratsummierbare Zufallsvariablen mit Erwartungswert $\mu = \mathbf{E}X_1 \in \mathbb{R}$ und Varianz $\sigma^2 = \mathbf{Var}X_1 \in [0, \infty)$. Dann gelten

$$\mathbf{E}\bar{X}_n = \mu \quad \text{und} \quad \mathbf{Var}\bar{X}_n = \frac{\sigma^2}{n},$$

und damit

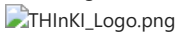
$$\|\bar{X}_n - \mu\|_2 = \sqrt{\mathbf{Var}\bar{X}_n} = \frac{\sigma}{\sqrt{n}}.$$

\bar{X}_n konvergiert also im quadratischen Mittel gegen μ .



Unsere Motivation

- integrierte Lehrinfrastruktur für Lehrende und Studierende
- interaktive Präsentationen in Vorlesungen
- organisiertes Bereitstellen von Übungen
- Förderung durch Bundesministerium für Forschung, Technologie und Raumfahrt (bzw. BMBF) im Rahmen des THInKI-Projekts



Was sind Jupyter Notebooks?

- interaktive Computing-Plattform
- JSON-Quelldatei: komfortable Bearbeitung im Browser/Programmierungsumgebung
- enthält u. a. die **Zellen**
 - Textzellen mit Markdown
 - Codezellen mit ausführbarem Code
- Bearbeitung live möglich → interaktiv
- gehostet auf einem **Notebook-Server** (lokal oder online)
- Zugriff über Browser
- Erweiterung für Präsentationen = **RISE (Reveal.js - Jupyter/IPython Slideshow Extension)**
- Installation: <https://jupyter.org/install> und <https://rise.readthedocs.io/en/latest/installation.html>

Text-Zellen

basieren auf **Markdown** = reiner Text + einfache „Markierungen“

- `*kursiv*` ergibt *kursiv*
- `**fett**` ergibt **fett**
- `***fett und kursiv***` ergibt ***fett und kursiv***
- `~durchgestrichen~` ergibt ~durchgestrichen~

- ``Code`` für `Code`
- Listen: `-` am Zeilenanfang, für Hierarchie einrücken
- Aufzählungen: `1.` am Zeilenanfang, z. B.
 1. erstens
 2. zweitens
 3. drittens (Nummerierung irrelevant, am einfachsten immer `1.`)
- `[Link auf ein Cheatsheet](https://www.markdownguide.org/cheat-sheet/)` ergibt [Link auf ein Cheatsheet](https://www.markdownguide.org/cheat-sheet/)
- **Überschriften** verschiedener Stufe: `#` bis `#####`; `##` geeignet für **Folientitel**
- um Zeichen mit besonderer Bedeutung zu erhalten, ggf. `\` voranstellen, also `*` für `*`

Fortgeschrittenes Markdown

- **mathematische Formeln** wie in `LaTeX` :

1. in der Zeile mit `$a^2 + b^2 = c^2$`, z. B. $a^2 + b^2 = c^2$

2. abgesetzt mit `$$ e^{\mathrm{x}} = \cos x + \mathrm{i} \sin x $$`

$$e^{ix} = \cos x + i \sin x$$

beachte: Hervorhebungen mit `**` nur für gesamte Formel

3. Definition von `LaTeX-Makros` in `$$...$$` möglich (siehe Beginn des Dokuments)


- `>` am Zeilenanfang für herausgehobene Blöcke (Zitate, Definitionen etc.)

Zitat:

Statistics state the state of the state.

Anonym.

- Bilder

- als Dateipfad: `![THInKI_Logo.png](attachment:THInKI_Logo.png)` 
- als Anhang der Zelle (einfügen per Drag and Drop); wird in der Notebook-Datei gespeichert `![jupy_logo.png](attachment:jupy_logo.png)`



- als Link: `![TUI](https://www.tu-ilmnau.de/fileadmin/_processed_/b/7/csm_Galerie_01__c_Hajo_Dietz_f0b1838c01.jpg)`



- **Tabellen:**

```
| Spalte 1 | Spalte 2 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Zelle 1,1 | Zelle 1,2 |
| Zelle 2,1 | Zelle 2,2 |
| Spalte 1 | Spalte 2 | | :- | :- | | Zelle 1,1 | Zelle 1,2 | | Zelle 2,1 | Zelle 2,2 |
```

- horizontale **Trennlinie** ---

- HTML möglich, insbesondere
 - `
` für **Zeilenumbruch** beispielsweise zwischen Überschrift und direkt folgender Liste
 - `🚀` für **Unicode** 🚀, insbesondere ` ` für **Leerzeichen**
- beachte: nicht alle Features verschiedener **Markdown-Varianten** funktionieren in jupyter notebooks

Eigene Designs und Fußzeile

- eigene (Corporate-)Designs mittels CSS
- z. B. Farben, Code-Highlighting oder Schriftarten
- Fußzeile (mit Logo) editierbar
- andere Varianten möglich

Tastatur-Kürzel für Jupyter Notebooks

Modus	Taste(nkombination)	Wirkung
Navigationsmodus	M	Zelle wird Markdown-Zelle
Navigationsmodus	Y	Zelle wird Code-Zelle
Navigationsmodus	Enter	Zelle bearbeiten
Bearbeitungsmodus	Esc	Bearbeitungsmodus verlassen
Bearbeitungsmodus	Strg+Enter	Zelle ausführen
Bearbeitungsmodus	Shift+Enter	Zelle ausführen und nächste Zelle auswählen
Bearbeitungsmodus	Alt+Enter	Zelle ausführen und darunter neue Zelle bearbeiten
Navigationsmodus	A	darüber neue Zelle
Navigationsmodus	B	darunter neue Zelle
Navigationsmodus	X	Zelle ausschneiden (und löschen)
Navigationsmodus	C	Zelle kopieren
Navigationsmodus	V	Zelle darunter einfügen

siehe auch im Menü *Help* → *Keyboard Shortcuts* oder Taste **H** im Navigationsmodus

Präsentationen mit RISE

- **Erweiterung** für jupyter notebook server (bis **Version 6**)
 - ermöglicht Anzeige als **Präsentation**
 - „überlange“ Folien erhalten eine **Scrollbar**
 - Folienübersicht verfügbar
 - Markierungen
 - Möglichkeiten zum Aufdecken von Slides
-
- verschiedene Slide Types, z. B. Slide, Sub-Slide, Fragment, Skip
 - Editierung außerhalb des Präsentationsmodus

- Tastatur-Kürzel:

Modus	Taste(nkombination)	Wirkung
Navigationsmodus	Alt+R	Präsentation mit aktueller Zelle starten
Präsentationsmodus	Alt+R	Präsentation beenden
Navigationsmodus	Shift+I	Zelle als neue Folie markieren
Navigationsmodus	Shift+G	Zelle als neuen Folienteil markieren
Navigationsmodus	Shift+K	Zelle zum Überspringen markieren
Präsentationsmodus	Space / PgDwn	nächste Folie
Präsentationsmodus	Shift+Space / PgUp	vorige Folie
Präsentationsmodus	W	Folienübersicht ein-/ausschalten
Präsentationsmodus	Shift+A	Malmodus
Präsentationsmodus	S	Farbwahl im Malmodus
Präsentationsmodus	-	Gemaltes löschen
Präsentationsmodus	T	Sprechermodus aktivieren

Diese Folie ist nicht im Präsentationsmodus zu sehen.

Code-Zellen

- bestehen aus 3 Teilen
 - **Prompt** nummeriert die Ausführung durch (**Reihenfolge** bei Abhängigkeiten!)
 - **Input** (Code)
 - **Output**
- ggf. *Kernel* → *Restart & Run All* zum Neustart und erneuter Ausführen in vorgesehener Reihenfolge

```
In [3]: x = 4
```

```
In [4]: x += 5
```

```
In [5]: # Ergebnis hängt von Ausführung der vorhergehenden Zellen ab.
print(x)
```

9

- Import von Paketen und Bibliotheken möglich
- einfachster Fall: gewöhnlicher Code mit Ausgabe

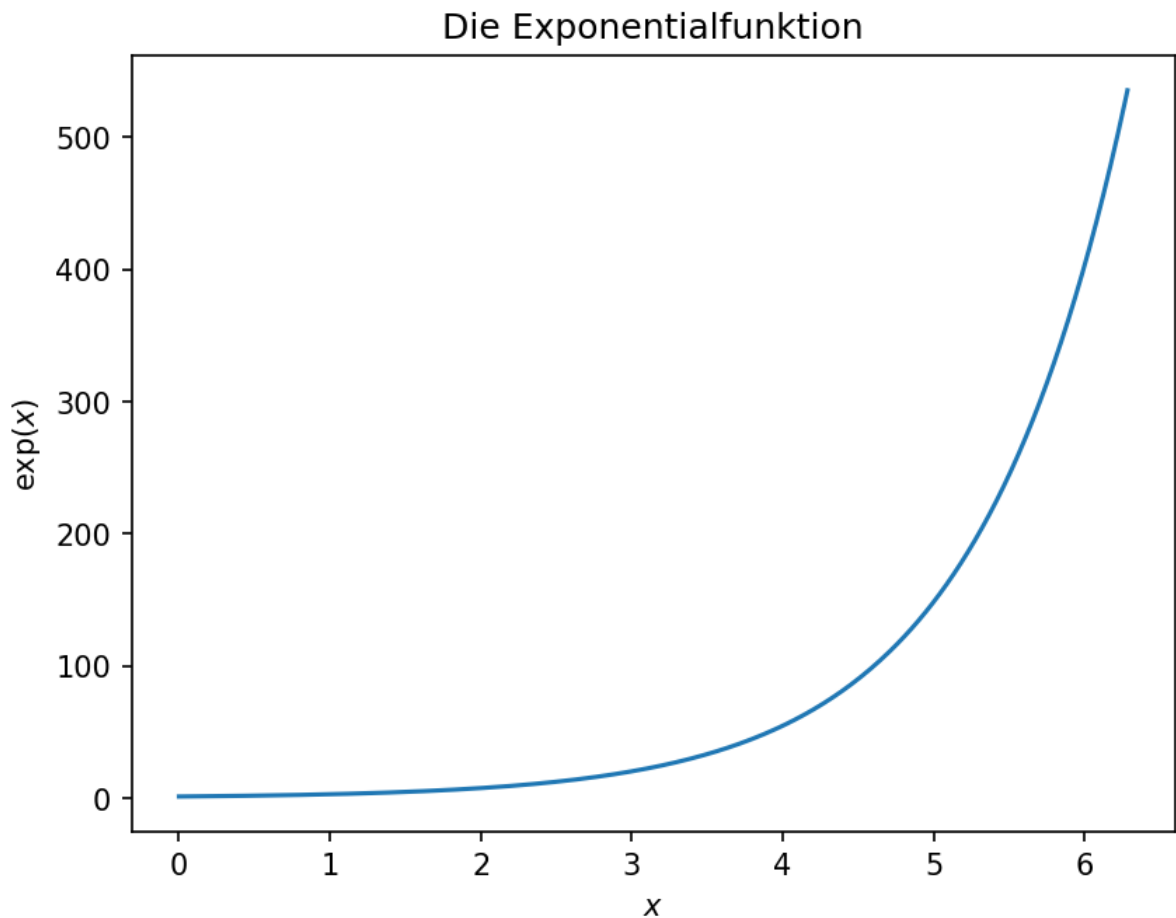
```
In [6]: import numpy as np
np.random.randint(6) + 1 # einmal würfeln
```

```
Out[6]: 1
```

- auch Plots als Ausgabe möglich:

```
In [7]: import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2 * np.pi, 400)
fig, ax = plt.subplots(dpi=150)
ax.plot(x, np.exp(x))
ax.set_xlabel("$x$")
ax.set_ylabel("$\exp(x)$")
ax.set_title("Die Exponentialfunktion");
```



- Pandas gibt `DataFrame` als Tabelle aus:

```
In [8]: import pandas as pd

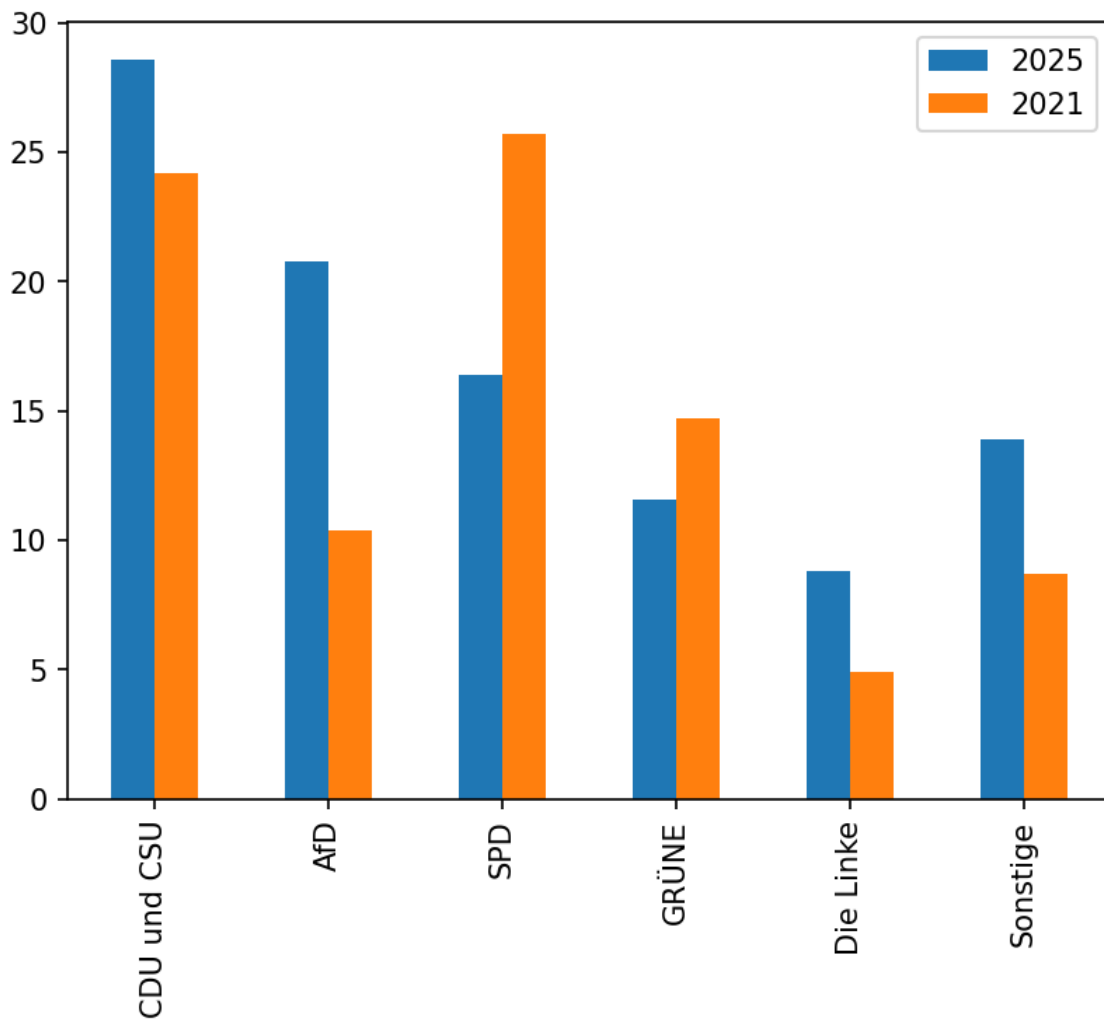
btw = pd.DataFrame({
    'CDU und CSU': pd.Series([28.6, 24.2], index=['2025', '2021']),
    'AfD': pd.Series([20.8, 10.4], index=['2025', '2021']),
    'SPD': pd.Series([16.4, 25.7], index=['2025', '2021']),
    'GRÜNE': pd.Series([11.6, 14.7], index=['2025', '2021']),
    'Die Linke': pd.Series([8.8, 4.9], index=['2025', '2021']),
    'Sonstige': pd.Series([13.9, 8.7], index=['2025', '2021'])
})
btw
```

```
Out[8]:
```

	CDU und CSU	AfD	SPD	GRÜNE	Die Linke	Sonstige
2025	28.6	20.8	16.4	11.6	8.8	13.9
2021	24.2	10.4	25.7	14.7	4.9	8.7

- Prompt / Input / Output kann mit Erweiterung `Hide_code versteckt` (W / E / R und mit `Shift+` wieder angezeigt) werden
- ggf. `None` als letzten Befehl, um Anzeige des Rückgabewerts zu verhindern

```
In [9]: btw.fillna(0).T.plot(kind='bar')
None
```



Alternative Kernel

- auch für **andere Programmiersprachen** stehen Kernel zur Verfügung
- beispielsweise [IRkernel](#) für **R**

Interaktive Widgets

- python -Paket [ipywidgets](#)
- ermöglicht Erstellung von „**Applets**“ mit Ein-/Ausgabefeldern, Slidern, Buttons etc.
- laufen **in einer Zelle**, genauer deren Output (kein Wechsel zwischen Zellen möglich)
- **vorher** ausführen (z.B. mittels Extension als `initialization cell` markieren) für Größe der Folie **und live neustarten**
- Prompt und Code ausblenden (s.o.)

Beispiel: Wahrscheinlichkeitsfunktion der Binomialverteilung

```
In [10]: import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import clear_output

n_slider = widgets.IntSlider(min=1, max=20, step=1, value=7, description='$n$', layout=widgets.Layout(width='50%'))
p_slider = widgets.FloatSlider(min=0, max=1, step=0.05, value=0.65, description='$p$', layout=widgets.Layout(width='50%'))
out = widgets.Output()

def plot_pmf(n, p):
    ks = np.arange(n + 1)
    with out:
        clear_output(wait=True)
        fig, ax = plt.subplots()
        ax.stem(ks, st.binom(n, p).pmf(ks), basefmt='k', markerfmt='.')
        ax.set_xlabel('$k$')
        ax.set_ylabel('Wahrscheinlichkeitsfunktion $f(k)$')
```

```
plt.show()

def handle(change):
    plot_pmf(n_slider.value, p_slider.value)

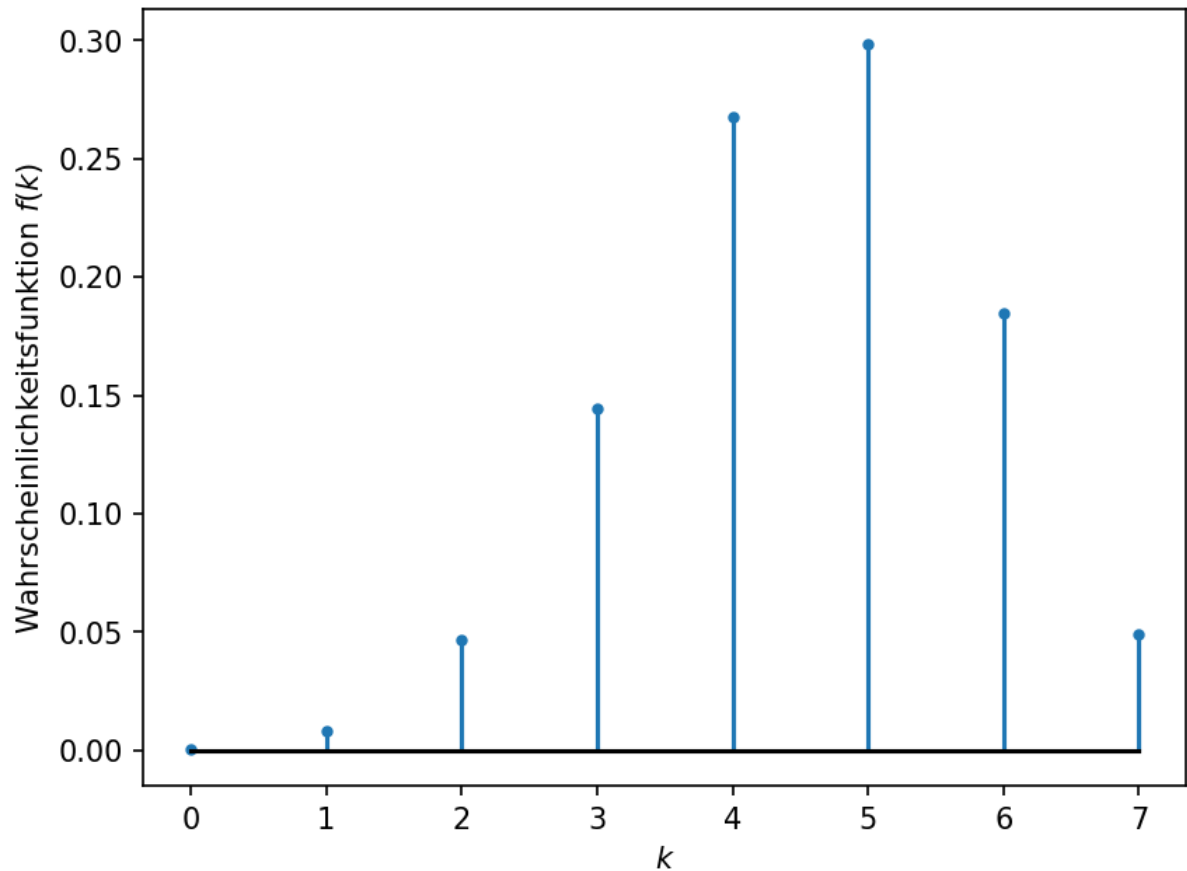
n_slider.observe(handle, names='value')
p_slider.observe(handle, names='value')

handle(None)
widgets.VBox(children=[n_slider, p_slider, out])
```

Out[10]:

n 7

p 0.65



Beispiel für ein komplexeres Widget: Wählermodell

```
In [11]: import time
import threading
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import clear_output
import matplotlib.colors as col
import matplotlib.colors as col
from IPython.display import display, clear_output

button_layout = widgets.Layout(width='200px', height='40px') # Set a specific width and height

L = 4

# Startzustand
t = 0
X = ( np.arange(L).reshape(L, 1) + np.arange(L).reshape(1, L) ) % 2
I = None
N = None
t_max = 1000
M = np.full(t_max + 1, np.nan)
M[0] = np.sum(X == 1)

# This Output widget will hold the grid or any visuals
out = widgets.Output()

with out:
```



```

fig, ax = plt.subplots()
ax.imshow(X, origin='lower',
          cmap=col.ListedColormap(["black", "white", "red"]),
          vmin=0, vmax=2)
ax.set_title(f'$t = {t}$')
ax.set_xticks(range(L))
ax.set_yticks(range(L))
plt.show()

# Dummy step function (replace with your actual grid update)
def step_func():
    global t, X, I, N, fig, ax

    if I is None:
        I = st.randint.rvs(0, L, size=2)
        X[I[0], I[1]] = 2
    elif N is None:
        N = I.copy()
        dim = st.randint.rvs(0, 2)
        N[dim] = ( I[dim] + 2 * st.randint.rvs(0, 2) - 1 ) % L
    else:
        X[I[0], I[1]] = X[N[0], N[1]]
        I = None
        N = None
        t += 1
        M[t] = np.sum(X == 1)

    with out:
        clear_output(wait=True)
        fig, ax = plt.subplots()
        ax.imshow(X, origin='lower',
                  cmap=col.ListedColormap(["black", "white", "red"]),
                  vmin=0, vmax=2)
        ax.set_title(f'$t = {t}$')
        ax.set_xticks(range(L))
        ax.set_yticks(range(L))
        if not N is None:
            ax.add_patch(plt.Rectangle(np.flip(N) - 0.5, 1, 1,
                                       color='red', fill=False, linewidth=3))
        plt.show()

# Buttons
play_button = widgets.Button(description="▶ Play", button_style='success', layout=button_layout)
pause_button = widgets.Button(description="⏏ Pause", button_style='warning', layout=button_layout)

# Animation control flag
is_running = False
animation_thread = None

def run_animation():
    global is_running
    while is_running:
        step_func()
        time.sleep(0.5) # 0.5 second interval

def on_play_clicked(b):
    global is_running, animation_thread
    if not is_running:
        is_running = True
        animation_thread = threading.Thread(target=run_animation, daemon=True)
        animation_thread.start()

def on_pause_clicked(b):
    global is_running
    is_running = False

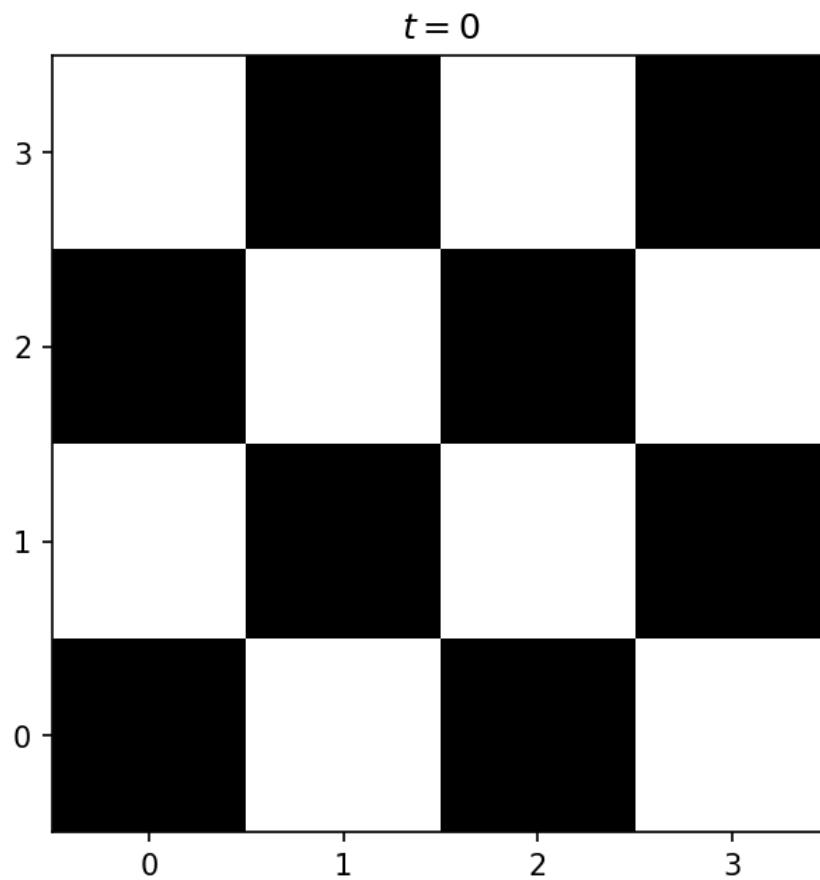
play_button.on_click(on_play_clicked)
pause_button.on_click(on_pause_clicked)

# Display everything
display(widgets.HBox([play_button, pause_button]))
display(out)

```

▶ Play

⏏ Pause

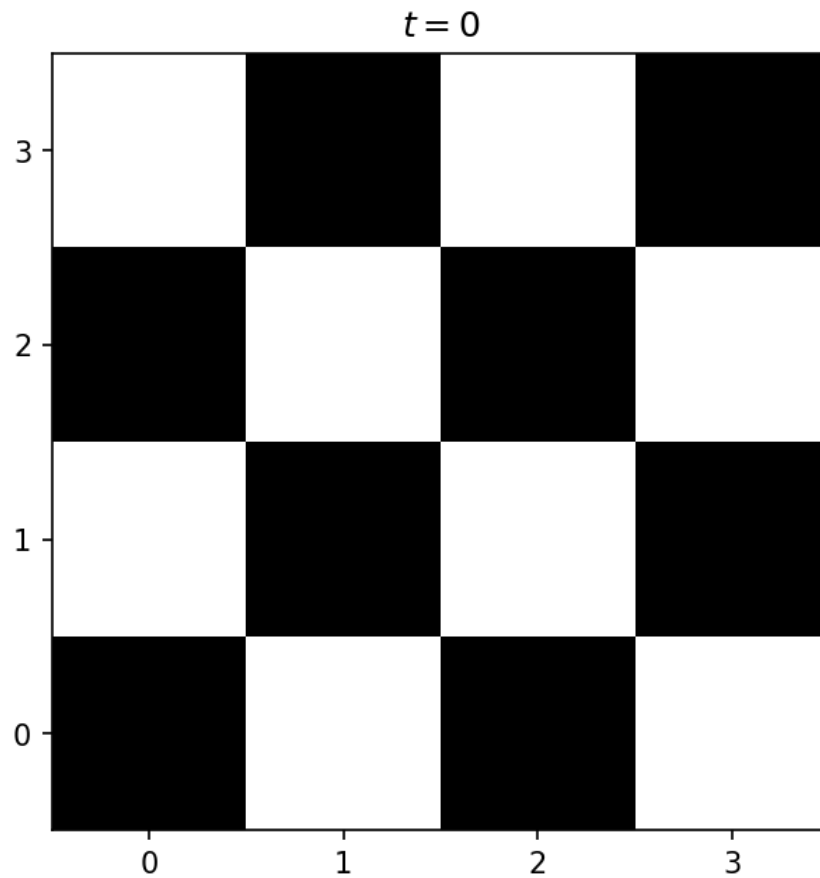


jupy_rise_toolkit

- Entwicklung im Rahmen des THInKI-Projekts
- entscheidende Beiträge und Ideen von Thomas Hotz, Stefan Heyder und Matthias Glock
- verfügbar unter https://github.com/Stochastik-TU-Ilmenau/jupy_rise_toolkit
- wesentliche Funktionen vorhanden
- Tests und vollständige Readme-Datei bis Ende November

hide-Funktion





PDF-Erzeugung

- Umwandlung der Notebooks in statisches Skript von Studierenden gewünscht
- automatisiertes Skript zur Umwandlung Notebook → HTML → PDF
- u. a. mit Inhaltsverzeichnis
- Seitenumbrüche können manuell hinzugefügt werden

Templates

```
In [13]: import os
import sys
import nbformat

def create_template(
    author: str = 'Author',
    title: str = 'Titel',
    date: str = None,
    path: str = None,
    filename: str = 'my_notebook' ):

    # default behaviour: use path of script that is calling the function
    if path == None:

        path = os.path.abspath('')

    # create new notebook
    nb = nbformat.v4.new_notebook()

    # add tilte
    cell_title = nbformat.v4.new_markdown_cell( '# ' + title )
    nb['cells'].append( cell_title )

    # add packages
    cell_packages = nbformat.v4.new_code_cell( 'import numpy as np' )
    nb['cells'].append( cell_packages )

    # add metadata TODO

    nb['metadata'].update({
        "rise": {
            "enable_chalkboard": True,
            "footer": "<div style=\"padding-left:6em;padding-bottom:0.1em;font-size:3em;\">>" +
```

```

        author + ": &nbsp; <i>" + title + "</i><img src=\"TU_Logo_SVG_crop.svg\" alt=\"TU Ilmenau\" height="
        "history": False,
        "progress": True,
        "reveal_shortcuts": {
            "chalkboard": {
                "download": "d",
                "toggleChalkboard": "shift-b",
                "toggleNotesCanvas": "shift-a"
            }
        },
        "scroll": True
    }
})

# write to file
cd = os.getcwd()

with open( path + '\\\\' + filename + '.ipynb', 'w' ) as f:
    nbformat.write(nb, f)

```

```

In [14]: if True:
        create_template( author = 'Tobias Bernstein', title = 'Mein erstes Jupyter Notebook', filename = 'from_nb' )

```

Live-Umfragen

- Vorbereiten von Umfrage und Ergebnis-Datei auf geeigneter Cloud (z. B. TU-Cloud oder Google Drive)
- Erstellen eines QR-Codes zum Zugriff auf Umfrage
- automatisches Auslesen der Ergebnisdatei
- Präsentation muss nicht verlassen werden!

```

In [15]: import qrcode
import requests
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import display
from io import StringIO

def show_QR_code( data: str, size: int = 10 ) -> None:

    # Generate QR code
    qr = qrcode.QRCode(
        version=1,
        error_correction=qrcode.constants.ERROR_CORRECT_L,
        box_size=size,
        border=4,
    )
    qr.add_data(data)
    qr.make(fit=True)

    # Create and show image
    img = qr.make_image(fill="black", back_color="white")
    display(img)
    # imgplot = plt.imshow(img)
    # plt.show()
    return None

def get_csv_from_cloud( url: str ) -> pd.DataFrame:

    # add /download
    url_answers = url + "/download"

    # press "Ergebnisse -> ... -> Tabellendokument neu exportieren"
    response = requests.get( url_answers )

    if response.status_code == 200:
        # Read CSV into pandas DataFrame
        csv_data = StringIO( response.content.decode( 'utf-8' ) )
        df = pd.read_csv( csv_data )
        return df
    else:
        print( "Failed to download CSV: ", response.status_code )
        return None

```

```

In [16]: show_QR_code("https://github.com/Stochastik-TU-Ilmenau/jupy_rise_toolkit", 25)

```



```
In [17]: url = "https://cloud.tu-ilmenau.de/s/YxQDbxXJxiWwHBS"
df = get_csv_from_cloud( url )
df.head()
```

```
Out[17]:
```

	Meldedatum	NeuerFall	AnzahlFall	AnzahlTodesfall	AnzahlGenesen
0	2020-10-01	0	2929	26	2903
1	2020-10-02	0	2929	25	2904
2	2020-10-03	0	2196	12	2184
3	2020-10-04	0	1156	9	1147
4	2020-10-05	0	2415	35	2380

Nutzen für die eigene Lehre

- Lehrende ...
 - nutzen **lokale Installation**
 - halten **Präsentation**
 - mit speziellem **mouse pointer**
 - mit (drahtloser) **Minitastatur**
 - **entwickeln (ohne RISE)** ggf. in Programmierumgebung (z.B. VSCode)
- Studierende ...
 - nutzen (ohne Installation) **JupyterHub** des Rechenzentrums (im Intranet/VPN) für notebook **ohne RISE**

- benötigen nur einen **Browser** (plattform-/hardware-unabhängig)
- können verborgene **Zusatzinhalte** zu Hause anschauen
- verwenden **Code-Beispiele** aus notebook für **Übungen**
- erhalten **PDF** für Notizen während der Vorlesung

Vorteile

- für Lehrende **leicht** zu erlernen und zu bedienen
- nur **eine** Quelldatei
- geringer Aufwand für Erstellen und Pflege der Inhalte
- Code kann **live** ausgeführt und verändert werden
- **interaktive** Elemente
- **live erhobene Daten** auswerten
- Präsentation und Code **in einer Umgebung / Datei**
- **open source** verfügbar für `python` und `R`
- **Zusatzinhalte** (heterogene Zielgruppen!) können eingefügt, aber nicht angezeigt werden

Nachteile

- gewisse **Instabilität**, ggf. Präsentation von Folie davor neu starten
- aktuell **Umbruch**: zukünftig nur für **JupyterLab** verfügbar
- **Ästhetik** des Skripts leidet
- eingeschränkte **Flexibilität** der Präsentation

Vorteile überwiegen!

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?

Mit freundlicher Unterstützung von

