

Angewandte Statistik I

INF3223 im Wintersemester 2018/19

Statistik

“Data science is the study of the generalizable extraction of knowledge from data” (Wikipedia)

Stochastik umschließt Wahrscheinlichkeitstheorie und Statistik

- beschreibende Statistik
- Wahrscheinlichkeitstheorie
- Schlußfolgernde Statistik

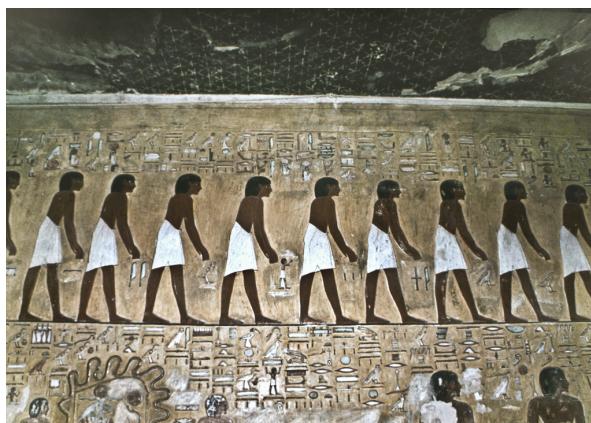
Geschichte

Frühzeit



Quelle: Clemens Schmillen, Wikipedia, CC BY-SA 3.0

Abgaben und Volkszählung in Ägypten



Tal der Könige. Quelle: Manfred Lentz, CC BY-NC-SA 2.0

Griechen, Babylonier

Musiklehre: arithmetisches, harmonisches, geometrisches Mittel



Sappho mit achtsaitiger Barbitos auf Vase. Quelle: Wikipedia, Public Domain.

Mittelalter, Aufklärung, Moderne

Stochastik [griech. *stochastikē* (*téchnē*) = zum Erraten gehörend(e Kunst)
(Begriff von Jakob Bernoulli)



- Jakob Bernoulli (Schweiz, 16. Jh)
- Thomas Bayes (England, 18. Jh)
- Carl Friedrich Gauß (D, 19. Jh)
- Gregor Mendel (D, 19. Jh)

Heute

I keep saying the sexy job in the next ten years will be statisticians. People think I'm joking, but who would've guessed that computer engineers would've been the sexy job of the 1990s? The ability to take data—to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it [...]

(Hal Varian, Chief Economist at Google. 2008)

Schlagworte

**big data, Kostenexplosion, Volkszählung, Datenkrake
Wirtschaftswissenschaftliche Spieltheorie, Aktienkurse, ...
Wissenschaft, machine learning, ...**

Beispiele

Chancen im Lotto, T€uro? Globale Erderwärmung, Kausalität, Lerneffekt in Wahrnehmung, ...

Ziele

Statistik-Werkzeuge mit denen der wissenschaftliche Alltag bearbeitet wird.

- Kennenlernen
- Verstehen
 - dazu Theorie
- Anwendungsbereiche
- Anwenden
- Programmieren
- Ergebnis
 - interpretieren und darstellen

Über mich

Physiker, Biologe, Elektroingenieur, Administrator.

Kein Statistiker

Alternativ besuchen Sie bitte eine spezielle Vorlesung z.B. Statistik I+II in den Kognitionswissenschaften oder Mathematik.

Ich werde versuchen, Ihnen die Verbindung von all dem, die *praktische* Statistik nahezubringen.

Über Sie

Was sind Ihre Vorkenntnisse in

1. Mathematik
 - Lineare Algebra
 - Analysis
2. Statistik
 - Kombinatorik
 - Wahrscheinlichkeitsrechnung
3. Programmieren
4. Python
5. Notebooks

Was sind Ihre Erwartungen?

Spezielle Problemstellung?

Formalien

Links

Homepage *Neuronale Informationsverarbeitung*

<http://www.nip.uni-tuebingen.de/teaching/lectures-seminars/winter-term-201819/zeitplan-angewandte-statistik-i.html>
<http://www.nip.uni-tuebingen.de/teaching/lectures-seminars/winter-term-201819/zeitplan-angewandte-statistik-i.html>

Informatik

<http://www.wsi.uni-tuebingen.de/studium> (<http://www.wsi.uni-tuebingen.de/studium>)

ilias

- Veranstaltungen
 - Wintersemester 2018-2019
 - 7 Mathematisch-Naturwissenschaftliche Fakultät
 - Informatik
 - Neuronale Informationsverarbeitung (Prof. Wichmann)
 - Angewandte Statistik I (WiSe 2018/19)

Anmeldung mit Passwort

- XXXXX
- Bei Problemen bitte Email an mich (Siehe Homepage)

Folien der Vorlesung, Skript

Email an Mitglieder

- hier findet alle Kommunikation statt
- Terminänderungen falls nötig

Forum

- Für Fragen
- ... und Antworten

Übungsaufgaben

- **Angewandte** Statistik
- Zu Ihrer eigenen Verständnisvertiefung
- jede / jede zweite Woche
- Besprechung in der Übungsstunde
- Voraussetzung für Teilnahme an Prüfung: 50% richtig bearbeitet
 - Bonusregelung?
- Nicht Teil der Benotung

Klausur

- Schriftliche Klausur
- Termin: 20. Februar 2019 14:15 (=Vorlesung)
- Raum: F119 Sand 6
- Dauer: 90 Minuten
- Voraussetzung: 50% der Punkte aus den Übungsaufgaben

Anmeldung zur Klausur

Siehe Prüfungsordnung bzw. neue Richtlinien.

Prüfungsordnungen 2010, 2015 und 2016

Anmeldeverfahren

Studierende der X-INF- Studiengänge müssen alle Prüfungen (Haupt- oder Nachprüfung) anmelden, für die sie eine Note verbucht haben wollen. Bei Nichtbestehen einer Klausur gibt es keine automatische Anmeldung zu Nachklausuren. Auch Master-Studierende nach der Prüfungsordnung 2016, d.h. alle Masterstudierenden, die seit WS 2016/17 ihr Studium begonnen haben, können sich jetzt über Campus anmelden und ihre Noten über Campus einsehen.

Kognitionswissenschaft: Wenn die Prüfungsanmeldung in Campus nicht möglich ist, melden sich Studierende der Kognitionswissenschaft beim jeweiligen Dozenten an und nicht schriftlich im Prüfungssekretariat.

Anmeldefristen

Die Anmeldung zu Prüfungen ist ab 1.12.2018 möglich. Die Anmeldeperiode endet in Campus einen Werktag vor der jeweiligen Prüfung.

Rücktritt von Prüfungen

Falls Studierende aus gesundheitlichen Gründen an einer angemeldeten Prüfung nicht teilnehmen können, müssen sie innerhalb von 3 Werktagen nach dem Prüfungstermin dem Prüfungssekretariat ein ärztliches Attest vorlegen. Andernfalls wird die Prüfung als nicht erschienen verzeichnet und mit 5,0 bewertet.

auch abmelden, falls nicht aufgrund der Kriterien innerhalb des Kurses zugelassen!

Veranstaltungsspezifische Regeln für die Teilnahme an Nachklausuren

Die Zulassung zu einer eventuellen Nachklausur in Veranstaltungen des FB Informatik ist veranstaltungsspezifisch geregelt, wird zu Beginn des Semesters in der Veranstaltung bekanntgegeben

Eventuelle Nachprüfung mit Hauptprüfungspflicht

Es gibt eine Hauptprüfung (schriftlich) und es liegt im Ermessen des Dozenten, ob es eine Nachprüfung (hier mündlich) gibt, was auch erst vom Ausgang der Hauptprüfung abhängig gemacht werden kann. An der Nachprüfung kann nur teilnehmen, wer an der Hauptprüfung teilgenommen hat.

Terminübersicht

(vorläufig)

Datum	Titel
17.10.2018	Einführung: Statistik, Python
24.10.2018	Beschreibende Statistik
31.10.2018	(--Allerheiligen--)
07.11.2018	Korrelation und Regression
14.11.2018	Wahrscheinlichkeitstheorie
21.11.2018	Diskrete Verteilungen
28.11.2018	Kontinuierliche Verteilungen
05.12.2018	Sätze der Statistik
12.12.2018	Schätzer, Chi-Quadrat
19.12.2018	Tests
26.12.2018	(--Weihnachtsferien--)
02.01.2019	(--Weihnachtsferien--)
09.01.2019	t-Test, Vertrauensintervalle
16.01.2019	Bootstrapping, Varianzanalyse
23.01.2019	Lineares Modell, OLS, Pandas, Patsy
30.01.2019	OLS, Versuchsplanung
06.02.2019	Repetitorium
20.02.2019	Klausur

Übungsstunden

- Ergänzung zur Vorlesung
- Besprechen der Übungsaufgaben
- Fragen

Fragen

- Verständnisproblem? Sofort fragen.
- Am Kapitelende
- Nach den Vorlesungen
- In den Übungsstunden
- Per ilias-Mail

Rückmeldungen

- gerne offen für Korrekturen und Verbesserungsvorschläge

Fragen?

Zeitaufwand

- 6 ECTS
- Vorlesung
 - Nachbereitung
- Übungsaufgaben
 - *Angewandte Statistik = Praxis*
 - Damit arbeiten
 - Verständnis überprüfen (Sie und ich)
- Übungsstunden
 - Besprechen der Aufgaben

Python

- *Universelle* Programmiersprache
- zahlreiche Bibliotheken
- Open Source
- Unterstützte Plattformen:
 - Linux, Windows, Mac, ...

Anaconda / Miniconda

Einfach zu installierende Distribution

Don't panic

Sie werden mit kleinem Aufwand Python *im Vorbeigehen* lernen und hoffentlich großen Nutzen daraus ziehen.

Inhalt Vorlesung

0. Python

- Anaconda Distribution
- numpy
- scipy
- matplotlib

1. Beschreibende Statistik

- absolute und relative Häufigkeit
- kummulierte Häufigkeitsverteilung
- Kennwerte
- Darstellungsformen

1.1 Zweidimensionale Verteilungen

- Kontingenz
- Korrelation
- Regression

2. Wahrscheinlichkeitstheorie

- Wahrscheinlichkeit
- Zufallsvariable
- Axiome von Kolmogoroff
 - Laplace Wahrscheinlichkeit
 - Verbundwahrscheinlichkeit
- bedingte Wahrscheinlichkeit
 - Unabhängigkeit
- Theorem von Bayes
- Kennzahlen
 - Erwartungswert
 - Varianz
 - Schiefe und Kurtosis
- Zufallszahlen
- Gesetz der großen Zahl

2.1 diskrete Verteilungen

- Binomial
- Poisson
- Multinomial
- Summen

2.2 kontinuierliche Verteilungen

- Wahrscheinlichkeitsdichte pdf
- kummulierte Verteilungsfunktion cdf
- Perzentil-Funktion
- Uniform
- Exponential
- Normal
- Gemeinsame und bedingte Verteilung
- Korrelation
- Funktion mehrere Zufallsvariablen

3. Schließende Statistik

3.1 Sätze der Statistik

3.2 Schätzungen

- Maximum Likelihood Methode
- Robuste Schätzungen

3.3 Tests

- Teststatistik
- einzelne und gepaarte Stichproben
- p-Wert
- Sinn und Unsinn

3.4 Vertrauensintervalle

- verschiedene Verteilungen
- Bootstrap Methode
- Vergleich von Schätzung, Test und Vertrauensintervall

4. Datenanalyse

- Nominale Daten
 - Chi-Quadrat-Test
- Varianzanalyse
 - Mehrere Stichproben
 - Multiple Vergleiche und Tests
 - verbundene Stichproben

4.1 Multivariate Statistik

- Mehrdimensionale Zufallsvariable
- Schätzung von Erwartungswert und Kovarianz
- Mehrdimensionale Normalverteilung
- Regression
- multiple Regression

5. Lineare Modelle

- $y = f(x)$
- abhängige und unabhängige Variable
- Parameterschätzung
- Zusammenhang mit
 - t-Test
 - Anova
 - Regression

6. Versuchsplanung

- Stichprobenplanung
- Auswertung von Umfragen

Weiterführende Vorlesungen

- **Angewandte Statistik II** (Fortsetzung: Komplexere Statistik, Bayes-Statistik)
 - Hauptkomponentenanalyse PCA
 - Unabhängigkeitsanalyse ICA
 - general linear models GLM
 - Bayes Statistik
- **Kapitel Statistik** (Informatik im Rahmen der Mathematik IV)
- **Statistik I+II für Kognitionswissenschaftler** (mathematische Behandlung)
- **trees** (weiterführend, spezielle Statistik)
- **Zeitreihen** (weiterführend, spezielle Statistik)
- **Machine Learning / Intelligent Systems** (weiterführend)

Literatur

in der UB mehrfach vorhandene Bücher

- Fahrmeir, Künstler, Pigeot, Tutz: Statistik, Springer-Verlag Berlin Heidelberg, 6. Auflage 2007
- Stahel: Statistische Datenanalyse, Vieweg&Sohn, 5. Auflage 2008

wissenschaftliche Artikel, Quellen im Internet

- zum jeweiligen Kapitel

Weiter

[Python installieren \(002_PythonInstallation.ipynb\)](#)

[Motivation Python \(003_MotivationPython.ipynb\)](#)

Fragen?

Einrichten von Python

Kurzversion für Fortgeschrittene

Python 3 (3.7) zusammen mit einigen Paketen (-Versionen):

python	3.7.0
ipython	6.5.0
jupyter	1.0.0
jupyter_client	5.2.3
jupyter_core	4.4.0
notebook	5.6.0
matplotlib	2.2.3
numpy	1.15.1
pandas	0.23.4
patsy	0.5.0
scipy	1.1.0
statsmodels	0.9.0

[August 2018] Start des jupyter notebook

```
ImportError: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.20' not found
        (required by python3.7/site-packages/zmq/backend/cython/../../../.././.
./libzmq.so.5)
```

lösbar durch downgrade der libgcc:

```
conda search libgcc
conda install libgcc=5.2.0
```

Ausführlicher für Einsteiger

In der Vorlesung und in den Übungen kommt Python 3 zusammen mit einigen Paketen (-Versionen) zum Einsatz:

Damit die Übungen (und später natürlich auch richtige Berechnungen!) auf dem eigenen Computer laufen, sollte eine komplette Umgebung installiert werden.

- systemweit Python
 - apt-get install python3 / yum install python3 / ...
 - Download <https://www.python.org/downloads/windows/> (<https://www.python.org/downloads/windows/>) und installieren
 - Pakete
 - apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-panda (systemweit)
 - pip install xxx --user (innerhalb Python)
- Anaconda Distribution
 - <https://conda.io/docs/user-guide/install/download.html> (<https://conda.io/docs/user-guide/install/download.html>)
 - Paket-Manager "conda"

Anaconda Distribution

Als Basis benutzt man am einfachsten die Anaconda-Python-Distribution (aktuell Python 3.7)

- kostenlos und frei
- Continuum Analytics, Austin TX,

Die Anaconda Distribution ist eine Software-Sammlung rund um Python, einschließlich der eigentlichen Python Programmiersprache, einem Manager *conda* und zahlreichen paketierten Bibliotheken (sog. *packages*), die aufeinander abgestimmt sind.

Vorteile

- als User möglich, benötigt keinen root/Administrator-Zugang
- Versionskontrolle
- keine inkompatiblen Pakete
- virtuelle Umgebung:
 - spezielle Pakete und Versionen
 - kann geteilt werden
 - Pfade abgeschirmt vom System

Nachteil

- jede Installation benötigt Platz (derzeit ca. 3GB)
- virtuelle Umgebung benötigt zusätzlichen Platz (nur diff zur Basis)

miniconda

Man kann sich mit der kleineren Variante *miniconda* begnügen (Download hier <http://conda.pydata.org/miniconda.html> (<http://conda.pydata.org/miniconda.html>)), muß dann aber alle erforderlichen Pakete mit

```
conda install paketname
```

nachladen. Abhängigkeiten werden automatisch aufgelöst.

- Continuum verdient Geld mit Support und verwendet selbst Python für Analysen.

Hinweise

USB-Stick

- mit Dateisystem, auf dem Links angelegt werden können (zB. ext2)
- aktives Verzeichnis auf den Stick wechseln und einen relativen Pfad angeben
- Pfad hinzufügen, wenn keine Kollision mit systemeigenem python- bzw. Stick-Verzeichnis
 - sonst *export PATH=/path/of/Stick/anaconda3/bin:\$PATH*

[ana, mini] conda

Überprüfen, ob conda funktioniert

Linux / Mac OSX: Öffne eine Konsole mit Prompt

Windows: Gib anaconda in das Suchfeld des Start-Menüs ein.

In der Kommandozeile `conda list` eingeben, das sollte die installierten Bibliotheken (packages) in der Arbeitsumgebung auflisten, beispielsweise:

Paketname	Version	
openssl	1.0.2p	h14c3975_0
pip	10.0.1	py37_0
python	3.7.0	hc3d631a_0
qt	5.9.6	h8703b6f_2
...		

Mehr zu conda hier: <https://conda.io/docs/user-guide/getting-started.html> (<https://conda.io/docs/user-guide/getting-started.html>). Wenn conda nicht gefunden wird, wurde die anaconda-Distribution nicht korrekt installiert.

Pakete nachinstallieren oder aktualisieren

Manchmal benötigt man eine zusätzliche oder neuere Version einer Bibliothek. Mittels **conda** lässt sich diese leicht suchen und installieren:

```
conda list
conda search pandas
Fetching package metadata: ....
pandas      0.8.1      np16py26_0      pkgs/free
pandas      0.8.1      np16py27_0      pkgs/free
....
pandas      0.23.4     py36h04863e7_0  pkgs/main
pandas      0.23.4     py36hf8a1672_0  conda-forge
pandas      0.23.4     py37h04863e7_0  pkgs/main
pandas      0.23.4     py37hf8a1672_0  conda-forge
```

Standard ist immer die neueste passende. Wenn man die Version 0.19.1 von pandas haben will :

```
conda install pandas=0.19.1
```

Abhängigkeiten zu anderen Bibliotheken/Paketen werden automatisch aufgelöst. Obacht: auch downgrade.

Alternativ kann auch das Python-eigene `pip install` benutzt werden.

```
pip install ggplot          # wenn nicht mittels conda verfügbar
```

Python

Python als (interaktive) Sprache

```
ipython
>>> print('hello world')
```

Python als Interpreter/Compiler

```
python hello.py
```

Python im Jupyter Notebook

- innerhalb des Programmtextes bewegen
- Programmzeilen und Daten ändern
 - zusammen mit den Ausgaben
 - Daten
 - Graphiken
- speichern
- weitergeben

Link: <http://jupyter-notebook.readthedocs.io/en/latest/> (<http://jupyter-notebook.readthedocs.io/en/latest/>)

Aufruf

Das Notebook lebt im Browser (geliefert vom lokalen Server)

- Export als html, pdf, ...

Linux / Mac OSX: an der Kommandozeile

```
jupyter notebook
```

Windows: In der Startmenü-Suche

```
jupyter notebook
```

eingeben und anklicken.

Formatierung

- Quelltext
- Kommentare als "reStructured" Text
- *LaTeX*-Formeln $a^2+b^2=c^2$

Info zur *markdown* Syntax: <http://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Working%20With%20Markdown%20Cells.html> (<http://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Working%20With%20Markdown%20Cells.html>)

Ein Notebook wird im .ipynb-Format (*JSON*) gespeichert.

- (sprich: "Jason") JavaScript Object Notation

Virtuelle Umgebung

Möchte man unterschiedliche Bibliotheken oder Versionen davon benutzen, bietet sich eine virtuelle Umgebung für jede Konfiguration an. Hier zB. Python 3.4, wenn miniconda im eigenen miniconda3 Verzeichnis installiert wurde:

```
~/miniconda3/bin/conda create -n myconda python=3.4 numpy=1.6    # Python in V  
3.4, numpy in V1.6  
source ~/miniconda3/bin/activate myconda  
(myconda) [w@W ~/miniconda3]$
```

(Windows: activate myconda)

Das setzt Umgebungsvariablen wie zB. den Pfad korrekt. Nun reicht zum Starten des lokalen Notebook-Servers ein

```
jupyter notebook
```

und es öffnet sich der default-Browser mit der conda-Startseite: Inhalt des Startverzeichnisses mit beispielsweise Notebooks.

Die Umgebung verläßt man wieder mit

```
source deactivate
```

Laborbuch

Man kann das [Notebook] als Laborbuch ansehen und verwenden.

- Ideen zu Papier^H^H^H^H^H^H Bits bringen
- Schnelle Berechnungen ausführen
- Daten analysieren mitsamt graphischer Ausgabe, Statistik, ...

Link: <http://jupyter-notebook.readthedocs.io/en/latest/index.html> (<http://jupyter-notebook.readthedocs.io/en/latest/index.html>)

Der Umgang ist einfach und fällt nach einer gewissen Einarbeitungsphase leicht.

- <https://github.com/jupyter/notebook/blob/master/docs/source/examples/Notebook/Running%20Code.ipynb> (<https://github.com/jupyter/notebook/blob/master/docs/source/examples/Notebook/Running%20Code.ipynb>),
- http://bebi103.caltech.edu/2015/tutorials/t0b_intro_to_jupyter_notebooks.html (http://bebi103.caltech.edu/2015/tutorials/t0b_intro_to_jupyter_notebooks.html) oder

Und hier <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks> (<https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>) ein Ausblick, was man machen kann.

Auch diese Vorlesung ist in Notebooks konzipiert.

Gerne darf man Text und Code daraus in seinem eigenen Notebook verwenden.

Veröffentlichen

- <http://nbviewer.jupyter.org/> (<http://nbviewer.jupyter.org/>)
- <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks#statistics-machine-learning-and-data-science> (<https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks#statistics-machine-learning-and-data-science>)

Bibliotheken

Wir verwenden einige Bibliotheken. Alle sind *open source* und werden von Freiwilligen entwickelt und betreut.

Dokumentationen finden sich ausführlich im Web, die meisten Pakete haben eigene Webseiten.

Außerdem in den Paketen, sogar im Notebook zugänglich:

- Tab-Taste zur Auto-Vervollständigung / Auswahl
- Fragezeichen zum Hilfeaufruf

Hauptsächlich verwenden wir für die statistischen Analysen

Bibliothek	Link
scipy.stats	http://docs.scipy.org/doc/scipy/reference/stats.html (http://docs.scipy.org/doc/scipy/reference/stats.html)
statsmodels	http://statsmodels.sourceforge.net/stable/index.html (http://statsmodels.sourceforge.net/stable/index.html)
Pandas	http://pandas.pydata.org/pandas-docs/stable/index.html (http://pandas.pydata.org/pandas-docs/stable/index.html)
	aufbauend auf
numpy	und
scipy	http://docs.scipy.org/doc/ (http://docs.scipy.org/doc/)
matplotlib	http://matplotlib.org/ (http://matplotlib.org/)

und nun viel Spaß...

```
In [1]: print('Hallo Statistik-Studentinnen und -Studenten!')  
Hallo Statistik-Studentinnen und -Studenten!
```

011_Folien

2018 年 11 月 24 日

1 Beschreibende Statistik

auch *deskriptive* oder *empirische* Statistik

1.1 Übersicht

- Warum Statistik

LagemaSS

Streuung

Graphiken

Abhängigkeit

2 Beschreibende Statistik

2.0.1 Daten oft komplex

```
In [1]: from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline

In [5]: # we have some data x (from below...)
print('data x is a {} and has {} values of {}'
      .format(type(x), x.shape, x.dtype))
c = int(len(x)/2)
print('x = {} ...'.format(x[:16]))
print('... {} ...'.format(x[c-8:c+8]))
print('... {} ...'.format(x[-16:]))

data x is a <class 'numpy.ndarray'> and has (3136,) values of int64
x = [128 128 128 128 128 127 126 126 126 126 128 129 130 131 131 129 127] ...
```

```
... [134 146 150 147 139 131 124 120 120 122 127 135 143 148 147 139] ...
... [125 128 131 132 132 130 128 127 126 125 126 126 127 128 128 128]
```

2.0.2 Graphik

```
In [6]: '''occurence of data values'''

fig = plt.figure(figsize=(12,1))                      # create long graph
plt.scatter(x, np.zeros_like(x))                      # draw dots along x-axis (y=0)
fig.gca().get_yaxis().set_visible(False)               # switch off y-axis numbers
plt.xlabel('data');                                  # label: suppress object
```

2.0.3 Datenreduktion

... auf einige KenngröSSen:

- LagemaSS
- Streuung
- Verteilung

Finde wichtige Information

Ohne relevante Information zu verlieren

```
In [4]: '''load data of an example image "Gabor-patch"
show histogram
This is the source of the data from above...'''

from matplotlib import image as mpi      # image related routines

img = mpi.imread('data/gabor.png')        # read image data from file
fig = plt.figure(figsize=(12, 4))         # rect canvas 12x4 inches
fig.add_subplot(1, 2, 1)                  # 1 row, 2 columns: 1st graph = image
plt.imshow(img, cmap=plt.cm.gray, interpolation='nearest') # the pixel image

fig.add_subplot(1, 2, 2)                  # 1 row, 2 columns: 2nd sub-graph = hist
imgvector3D = np.array(img)              # convert colors to values in matrix
lenx, leny = imgvector3D.shape[:2] # width x height [x colors]
print('image shape = {}x{}'.format(lenx, leny))
# take first color (R), round to 8Bit integer, flatten x and y
```

```

x = np.rint(255*np.reshape(imgvector3D[:, :, 0], lenx*leny)).astype(int)
plt.hist(x, bins=np.linspace(0, 256, 256+1)) # histogram with 1Bit-bins
# a (middle) line scan
print('{} linescan'.format(x[int(lenx/2)*leny:int(lenx/2+1)*leny]))
image shape = 56x56
[120 122 127 135 143 148 147 139 124 105 89 81 88 110 142 177 202 207
187 145 91 43 15 20 58 119 185 236 255 236 185 119 58 20 15 43
91 145 187 207 202 177 142 110 88 81 89 105 124 139 147 148 143 135
127 122] linescan

```

In [7]: np.histogram?

3 LagemaSS

3.1 Modus

Der Wert, der am häufigsten vorkommt - Es kann mehrere Modi geben - Modus kann atypisch sein

```

In [8]: nmax = 0
for i in range(256):
    n = x[x==i].shape[0]
    if n > nmax:
        nmax = n
        imax = i
print('the modus of data x is {} and has {} values'.format(imax, nmax))

```

the modus of data x is 128 and has 116 values

Nachteil:

- Modus kann Randwerte annehmen
- Es kann zwei oder mehr Modi geben ("multimodale Verteilung")

3.2 Median

- Mindestens eine Hälfte der Werte ist kleiner gleich
- mindestens eine Hälfte der Werte größer gleich

In der sortierten Liste aller Daten $x_{(i)}$ mit $i \in \{1 \dots n\}$ und $x_{(1)} \leq \dots \leq x_{(N)}$ ist der Median

$$\tilde{x} = \begin{cases} x_{(\frac{N+1}{2})} & N \text{ ungerade} \\ \frac{1}{2}(x_{(\frac{N}{2})} + x_{(\frac{N}{2}+1)}) & N \text{ gerade} \end{cases}$$

3.2.1 Minimale Betrags-Abweichung

Sei $f(t) = \sum_{i=1}^N |x_i - t|$

dann gilt

$$f(t) = \sum_{i=1}^N |x_i - t| \geq \sum_{i=1}^N |x_i - \tilde{x}| = f(\tilde{x}) \quad \forall t \in \mathbb{R}$$

\times schlange: Median

Beweis (für ungerade $N = 2k + 1$):

- $x_{(i)}$ d.h. x ist sortiert.
- x_i d.h. x ist nicht sortiert.

$$\begin{aligned} \sum_{i=1}^N |x_i - \tilde{x}| &= \sum_{i=1}^N |x_{(i)} - \tilde{x}| \\ &= \tilde{x} - x_{(1)} + \dots + \tilde{x} - x_{(k-1)} + x_{(k+1)} - \tilde{x} + \dots + x_{(N)} - \tilde{x} \\ &= -x_{(1)} - \dots - x_{(k-1)} + x_{(k+1)} + \dots + x_{(N)} \\ &= t - x_{(1)} + \dots + t - x_{(k-1)} + x_{(k+1)} - t + \dots + x_{(N)} - t \\ &\leq |t - x_{(1)}| + \dots + |t - x_{(k-1)}| + |t - x_{(k)}| + |x_{(k+1)} - t| + \dots + |x_{(N)} - t| \\ &= |x_{(1)} - t| + \dots + |x_{(k-1)} - t| + |x_{(k)} - t| + |x_{(k+1)} - t| + \dots + |x_{(N)} - t| \\ &= \sum_{i=1}^N |x_{(i)} - t| \\ &= \sum_{i=1}^N |x_i - t| \end{aligned}$$

3.2.2 Median unter linearer Transformation

$$y = ax + b \quad \Rightarrow \quad \tilde{y} = a\tilde{x} + b$$

3.2.3 Wie den Median berechnen?

`numpy` stellt die Funktion `median()` bereit.

```
In [9]: xmedian = np.median(x) # <== numpy calculates the median
print('Median = {:.3f}'.format(xmedian)) # (three meaningful digits)
```

```
Median = 127.000
```

```
In [11]: '''sorted data - values vs index'''

xsorted=np.sort(x[:])                                # sort by numpy; ':' makes copy
medx = np.median(xsorted)                            # the median of sorted data
n = len(xsorted)                                    # how many values?
n1 = int(np.round(n/2.-1))                          # highest index under
n2 = int(np.round(n/2.))                            # lowest index above median
print('median={:.4f} between x({})={:.4f} and x({})={:.4f}'.
      format(medx, n1, xsorted[n1], n2, xsorted[n2]))
indices = np.arange(n)                               # array of index numbers

fig = plt.figure(figsize=(10,5))                      # wide canvas
plt.plot( indices, xsorted, 'b,' , label='x')# at each value-point step upward 1
plt.xlabel('sorted index')
plt.ylabel('values')
plt.plot((0, n), 2*[medx], 'y--' , label='median') # include the median in x-
plt.plot(2* [.5*n] , (0., xsorted[-1]), 'y--')# and y-direction
plt.axis((0, n, 0, x.max()))                         # restrict area to full data range
plt.legend(loc='lower right');
```

median=127.0000 between x(1567)=127.0000 and x(1568)=127.0000

```
In [12]: '''sorted data - index vs value'''
```

```
# same data as above
print('median={:.4f} between x({})={:.4f} and x({})={:.4f}'.
      format(medx, n1, xsorted[n1], n2, xsorted[n2]))

fig = plt.figure(figsize=(10,5))                      # wide canvas
# x-axis now really x-values; y-axis has indices
plt.plot( xsorted, indices, 'g,' , label='x')
plt.xlabel('sorted index')
plt.ylabel('values')
plt.plot(2* [medx] , (0, n), 'y--' , label='median')# include the median in y-
plt.plot((0., xsorted[-1]), 2* [.5*n] , 'y--')# and x-direction
plt.axis((0, x.max(), 0, n))                        # restrict area to full data range
plt.legend(loc='lower right');
```

median=127.0000 between x(1567)=127.0000 and x(1568)=127.0000

4 Mittelwerte

4.1 Arithmetisches Mittel

Gegeben seien $i = \{1 \dots N\}$ Werte x_i , dann ist deren arithmetisches Mittel

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

Beispiel Gewicht der Personen im Fahrstuhl - Wären alle gleich schwer, wie groß ist dieser Durchschnitt?

4.1.1 Minimale quadratische Abweichung

Sei $f(t) = \sum_{i=1}^N (x_i - t)^2$
dann gilt $f(t) \geq f(\bar{x}) \quad \forall t \in \mathbb{R}$
最小方差

$$\begin{aligned} f(t) &= \sum_{i=1}^N [(x_i - \bar{x}) + (\bar{x} - t)]^2 \\ &= \sum_{i=1}^N (x_i - \bar{x})^2 + 2 \sum_{i=1}^N (\bar{x} - t)(x_i - \bar{x}) + \sum_{i=1}^N (\bar{x} - t)^2 \end{aligned}$$

Beweis:

$$\begin{aligned} &= \sum_{i=1}^N (x_i - \bar{x})^2 + 2(\bar{x} - t) \sum_{i=1}^N (x_i - \bar{x}) + (\bar{x} - t)^2 \sum_{i=1}^N 1 \\ &= f(\bar{x}) + 0 + n(\bar{x} - t)^2 \\ &\geq f(\bar{x}) \end{aligned}$$

4.1.2 Wie den Mittelwert berechnen?

- *numpy-ndarray* hat die Methode `.mean()`
- *numpy* hat die Funktion `np.mean()`

```
In [13]: fig = plt.figure(figsize=(12,1))
plt.scatter(x, np.zeros_like(x), label='x')
xmean = x.mean()
print('mean of x = {:.5f}'.format(xmean))
print('median of x = {:.5f}'.format(xmedian))
plt.plot(2*xmean, [-.5, .5], 'r--', label='mean')
fig.gca().get_yaxis().set_visible(False);      # switch off y-axis numbers
```

```
mean    of x = 127.36575
median of x = 127.00000
```

4.1.3 Lineare Transformation

Das arithmetische Mittel transformiert sich ebenso wie die Daten unter einer linearen Transformation:

$$y_i = ax_i + b \quad \Rightarrow \quad \bar{y} = a\bar{x} + b$$

Beweis: [ÜA]

4.1.4 Summe von Mittelwerten

Gegeben seien N Werte x_i mit Mittelwert \bar{x} und M Werte y_j mit Mittelwert \bar{y} . Dann ist das gemeinsame arithmetische Mittel

$$\bar{z} = \frac{1}{N+M}(N \cdot \bar{x} + M \cdot \bar{y})$$

im allgemeinen $\neq \frac{1}{2}(\bar{x} + \bar{y})$ ##### Beweis [ÜA]

4.1.5 Merkwürdiges Beispiel

Wenn beide Teilmittelwerte kleiner werden, wird das gemeinsame Mittel auch kleiner?

Die Einkommen aller N Einwohner in Stadt A $a_i \ i \in \{1..N\}$ seien niedriger als diejenigen der M Einwohner in Stadt B mit $b_j \ j \in \{1..M\}$: $a_i < b_j \ \forall i, j$

Der Reichste aus A zieht nach B

- Das Durchschnittseinkommen in A wird kleiner
- Das Durchschnittseinkommen in B wird kleiner

Aber - Das Durchschnittseinkommen alle Einwohner in beiden Städten zusammen bleibt gleich!

4.1.6 Anwendung

- *herkömmlicher* Mittelwert
 - $N * \bar{x} = \sum x_i$
- Sanfte Berücksichtigung der "Fehler" um den Mittelwert
- Starke Berücksichtigung der weit außen liegenden "Fehler"

4.2 Andere Mittelwerte

Beispiel Preissteigerung Die Inflation beschreibt die jährliche Preissteigerung bei etwa gleichbleibendem Warenkorb. Als Referenz wird ein Jahr festgelegt und darauf auf 100% normiert. Quelle: destatis.de

- Index[Jahr]
- Rate[Jahr] = Index[Jahr] / Index[Jahr-1]
- Steigerung = (Rate - 1) × 100 %

```
In [14]: import pandas as pd                      # Pandas handles data conveniently
         inflation = pd.read_table('data/inflation_de.dat', sep=' +',
                                      header=None, engine='python')      # separator allows reg-expressions: 1+
         years = np.asarray(inflation[0])           # first column: years
         indices = np.asarray(inflation[1])          # second column: total inflation rates
         plt.plot(years, indices, label='indices 2010=100%') # relativ to 2010 = 100%
         print('#years= {} , difference= {} years, total ratio = +{:3f}%'.format(
               len(years), years[0]-years[-1], 100*(indices[0]/indices[-1]-1.)))
# calculate single rates every year
         rates = np.asarray([indices[j]/indices[j+1] for j, ind in enumerate(indices[:-1])])
         plt.plot(years[:-1], 100*rates, label='yearly rates [%]')
         ratemean = rates.mean()-1.                  # mean of yearly growing rates
         print('mean yearly growth rate = {:.3f}%'.format(100*ratemean))
         plt.plot([years[0], years[-1]], 2*[100*(1+ratemean)], 'g--',
                  label='average yearly rate [%]')
# base line: no change=100% of previous year
         plt.plot([years[0], years[-1]], 2*[100], 'k:')
         plt.legend(loc='lower right');

#years= 27 , difference= 26 years, total ratio = +55.698%
mean yearly growth rate = 1.723%
```

Test $\text{Index}[2017] = \text{Index}[1990] \cdot \text{mittlereRate}^{26}$

```
In [15]: print('from index[1990] = {} with {} times a rate of {}'
            .format(indices[-1], years[0]-years[-1], rates.mean()))
         print('leads to index[{}] of {}'
            .format(years[0], indices[-1]*rates.mean()**((years[0]-years[-1]))))
         print('but in [{}] should be {}'.format(years[0], indices[0]))
```

```
from index[1990] = 70.2 with 26 times a rate of 1.0172322892824832
leads to index[2017] of 109.46131973108793
but in [2017] should be 109.3
```

4.2.1 mittlere Steigerung?

Sei der Ausgangswert p_0 und die **Rate** im i-ten Intervall zum vorherigen $x_i = \frac{p_i}{p_{i-1}}$, dann gilt bis zum Intervall n $p_0 \cdot x_1 \cdot x_2 \cdot \dots \cdot x_n = p_0 \cdot \frac{p_1}{p_0} \cdot \frac{p_2}{p_1} \cdot \dots \cdot \frac{p_n}{p_{n-1}} = p_n$

Gesucht ist die mittlere Wachstumsrate w , so dass gilt $p_0 \cdot w^n = p_n$ beziehungsweise $x_1 \cdot \dots \cdot x_n = w^n$

Dann ist w , die mittlere Wachstumsrate, das *geometrische Mittel* der einzelnen Wachstumsraten.

4.3 Geometrisches Mittel

$$\bar{x}_{geom} = \sqrt[N]{x_1 \cdot x_2 \cdot \dots \cdot x_N} = \left(\prod_{i=1}^N x_i \right)^{\frac{1}{N}}$$

```
In [16]: m0 = rates.mean()
print('arithmetic mean = {:.5f}%'.format(100*(m0-1)))

import scipy.stats      # contains geometric mean and lots of statistics...
m2 = scipy.stats.gmean(rates)
print('geometric mean  = {:.5f}% by scipy.stats.gmean()'.format(100*(m2-1)))

arithmetic mean = 1.72323%
geometric mean  = 1.71746% by scipy.stats.gmean()
```

```
In [17]: print('The real data in {} is {}')
          .format(years[0], indices[0]))
print('Starting with {} in {} the arithmetic mean {:.6f} gives {:.2f} in {}'
      .format(indices[-1], years[-1], m0,
              indices[-1]*(m0)**(years[0]-years[-1]), years[0]))
print('Starting with {} in {} the geometric  mean {:.6f} gives {:.2f} in {}'
      .format(indices[-1], years[-1], m2,
              indices[-1]*(m2)**(years[0]-years[-1]), years[0]))
```

The real data in 2017 is	109.3
Starting with 70.2 in 1991 the arithmetic mean 1.017232 gives 109.46 in 2017	
Starting with 70.2 in 1991 the geometric mean 1.017175 gives 109.30 in 2017	

4.3.1 Ein weiteres Gegen-Beispiel

Jeden Tag tanken für 20 bei sich ändernden Preisen. Welches ist der durchschnittliche Preis?

Man erhält - bei einem Preis von $e /1$ - für $20 - l = \frac{20}{e}$ Liter.

```
In [18]: d = 10 # number of days
fixedeuros = 20. # amount in Eur
ppl = np.array([1.64,1.62,1.30,1.49,1.50,1.58,1.45,1.38,1.30,1.34])# prices
liters = fixedeuros/ppl
totaleuros = d*fixedeuros
totalliters = liters.sum()
print('total liters bought: {:.4f} and total money {:.2f}EUR spent'
      .format(totalliters, totaleuros))
print('makes an average price of {:.4f} /l'.format(totaleuros/totalliters))
print('arithmetic mean: {:.4f} /l'.format(ppl.mean()))
print('geometric mean: {:.4f} /l'.format(scipy.stats.gmean(ppl)))
```

total liters bought: 137.9356 and total money 200.00EUR spent
makes an average price of 1.4500 /l
arithmetic mean: 1.4600 /l
geometric mean: 1.4550 /l

4.4 harmonisches Mittel

$$\bar{x}_{harm} = \frac{1}{\frac{1}{N} \sum_{i=1}^N \frac{1}{x_i}}$$

- Sinnvoll bei Verhältniszahlen
- kann durchaus extremere Unterschiede zeigen wie im Beispiel $x = \{1, 4, 4\}$ mit *arithmetischem* Mittel 3 und *harmonischem* Mittel 2

```
In [19]: print('harmonic mean: {:.4f} /l'.format(scipy.stats.hmean(ppl)))
```

harmonic mean: 1.4500 /l

4.4.1 Anwendung Elektronik: Parallelschaltung von Widerständen

Ohmsches Gesetz: $R = \frac{U}{I}$, bei gleicher Spannung U addieren sich die Stromstärken I

$$R_{parallel} = \frac{U}{I_1 + I_2 + \dots + I_n} = \frac{U}{\frac{U}{R_1} + \dots + \frac{U}{R_n}} = \frac{1}{\frac{1}{R_1} + \dots + \frac{1}{R_n}}$$

4.5 Mittelwerte im Vergleich

$$x_{min} \leq \bar{x}_{harm} \leq \bar{x}_{geom} \leq \bar{x}_{arithm} \leq x_{max}$$

4.6 Zusammenfassung

Der Mittelwert liefert **eine** wichtige Kennzahl für die Daten - Modus

$$x_i : h_i \geq h_j \quad \forall j$$

- Median

$$\tilde{x} = x_{(\frac{N+1}{2})} \quad (N \text{ ungerade})$$

- arithmetisches Mittel

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

- geometrisches Mittel

$$\bar{x}_{geom} = \sqrt[N]{x_1 \cdot x_2 \cdot \dots \cdot x_N}$$

- harmonisches Mittel

$$\bar{x}_{harm} = \frac{1}{\frac{1}{N} \sum_{i=1}^N \frac{1}{x_i}}$$

5 Fragen?

012_Folien

2018 年 11 月 23 日

```
In [1]: from matplotlib import pyplot as plt  
import numpy as np  
%matplotlib inline
```

1 Beschreibende Statistik

1.1 Übersicht

- Warum Statistik

LagemaSS

Streuung

Graphiken

Abhängigkeit Bei gleichem LagemaSS können sich die zugrundeliegenden Daten unterscheiden:

```
In [3]: '''survey for age of visitors in theme park'''  
a = np.array([15, 16, 64, 15, 66, 66, 15, 16])  
b = np.array([34, 33, 34, 34, 36])  
c = np.array([8, 68, 50, 48, 4, 23, 13, 27, 79, 22])  
print('average age in group A: {:.1f} B: {:.1f} C: {:.1f} years'  
      .format(a.mean(), b.mean(), c.mean()))
```

average age in group A: 34.1 B: 34.2 C: 34.2 years

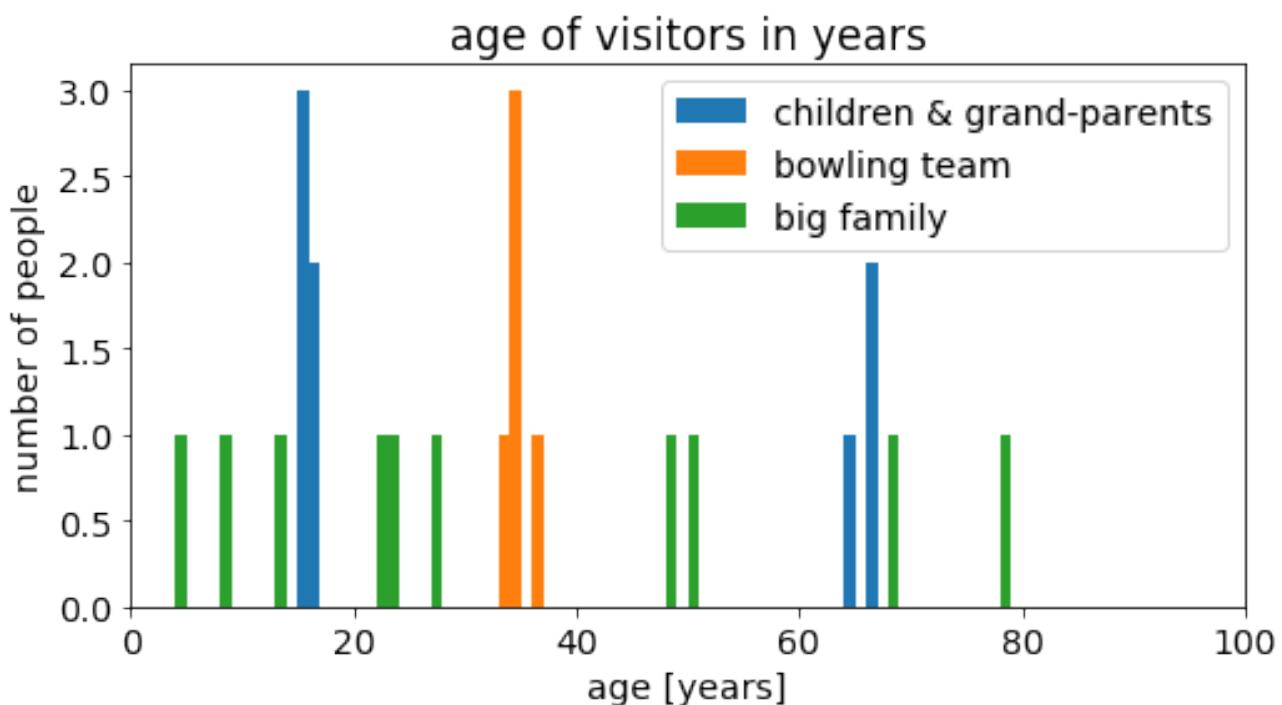
```
In [4]: '''same mean, different width'''  
print('average age in group A: {:.1f} B: {:.1f} C: {:.1f} years'  
      .format(a.mean(), b.mean(), c.mean()))  
bins = range(80)  
plt.figure(figsize=(8, 4))  
plt.hist(a, bins=bins, label='children & grand-parents')
```

```

plt.hist(b, bins=bins, label='bowling team')
plt.hist(c, bins=bins, label='big family')
plt.title('age of visitors in years')
plt.xlabel('age [years]')
plt.ylabel('number of people')
plt.xlim(0, 100)
plt.legend();

```

average age in group A: 34.1 B: 34.2 C: 34.2 years



2 StreuungsmaSSe

Spannweite Der Bereich vom kleinsten zum größten Wert

$$R = x_{\max} - x_{\min}$$

```
In [5]: print('range of age in \ngroup A: {:.2d} \ngroup B: {:.2d} \ngroup C: {:.2d}' +
           .format(a.max()-a.min(), b.max()-b.min(), c.max()-c.min()))
```

range of age in
group A: 51
group B: 3
group C: 75

2.1 Empirische Varianz

$$s^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$$

2.1.1 Verschiebungssatz

$$s^2 = \bar{x^2} - \bar{x}^2$$

Beweis: [ÜA]

```
In [6]: '''calculate mean and variance "on the fly"'''
np.random.seed(654321)
xx = 2*np.random.random(size=10) - 1. # randomly sampled -1 .. +1
print(xx)
xxsum = 0                      # initialize the sum
xxqsum = 0                      # the square-sum
n = 0                           # the number of measures
for xi in xx:                  # after every step: tell me mean and standard deviation
    xxsum += xi
    xxqsum += xi**2
    n += 1                       # to distinguish from i, even if nx = 1 + i
    print('{:2d} th step has mean {:.5.2f} and std {:.5.2f}'.
          format(n, xxsum/n, np.sqrt(xxqsum/n-(xxsum/n)**2)))
[-0.53601194  0.46506068  0.92618017  0.38634326  0.71896458  0.67740375
 -0.84280437 -0.85675517 -0.93527115  0.93672658]
1 th step has mean -0.54 and std  0.00
2 th step has mean -0.04 and std  0.50
3 th step has mean  0.29 and std  0.61
4 th step has mean  0.31 and std  0.53
5 th step has mean  0.39 and std  0.50
6 th step has mean  0.44 and std  0.47
7 th step has mean  0.26 and std  0.63
8 th step has mean  0.12 and std  0.69
9 th step has mean  0.00 and std  0.73
10 th step has mean 0.09 and std  0.75
```

2.1.2 Stichprobenvarianz

Die Stichprobenvarianz ist leicht unterschiedlich definiert durch

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Hintergrund:

- Die Anzahl der Freiheitsgrade ist $N - 1$, da Nebenbedingung $\sum_{i=1}^N (x_i - \bar{x}) = 0$
- Für $N=1$ ist die Varianz nicht definiert
 - anstatt 0 bei der empirischen Varianz
- Für große N ist sie asymptotisch gleich

2.1.3 Eigenschaften

Unter der linearen Abbildung

$$y_i = ax_i + b$$

ergibt sich

$$s_y^2 = a^2 s_x^2$$

2.2 Standardabweichung

$$s = +\sqrt{s^2}$$

Unter der linearen Abbildung

$$y_i = ax_i + b$$

ergibt sich

$$s_y = |a| s_x$$

```
In [7]: '''example with three groups visiting Volksfest'''

print('std. dev. of age in A: {:.1f}  B: {:.1f}  C: {:.1f}'
      .format(a.std(), b.std(), c.std()))

std. dev. of age in A: 24.2  B: 1.0  C: 24.4
```

3 Histogramme

Viele Daten: von der Strichliste zum Histogramm

3.0.1 Diskrete Ereignisse

Beispiel: Münze, Würfel, Karten, Körpergrößen in 5cm-Schritten, ...

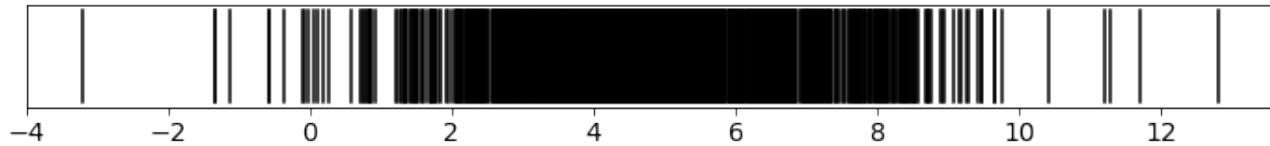
3.0.2 kontinuierliche Ereignisse

KörpergröSSe, Temperatur

Klasseneinteilung führt wieder zurück auf den diskreten Fall

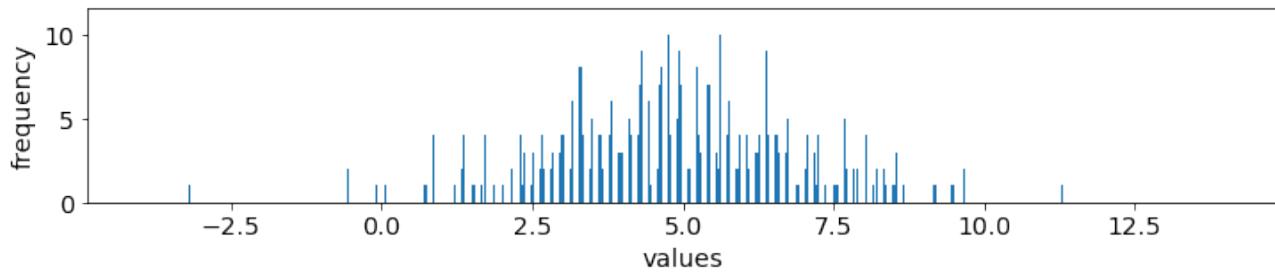
In [8]: *''': thousand random numbers'''*

```
np.random.seed(9876543)
xmany = np.random.normal(loc=5., scale=2., size=1000)
xmany[0] = -3.2                         # extra values for later
xmany[1] = 12.8
plt.figure(figsize=(12,1))
plt.plot(2*xmany, [0, 1], 'k-')
plt.yticks([]);
```



In [9]: `plt.figure(figsize=(12, 2))`

```
bins = np.linspace(-4, 14, (14+4)*40+1)
hist_full = plt.hist(xmany, bins, rwidth=0.4)
plt.xlabel('values')
plt.ylabel('frequency');
```



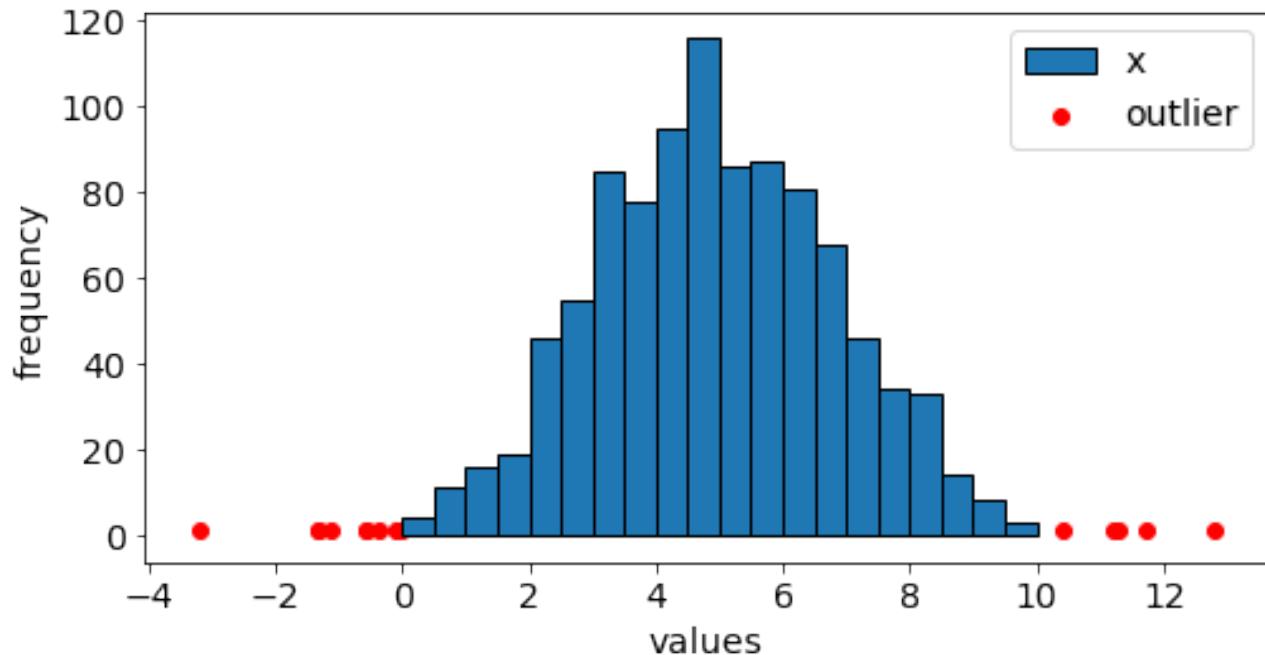
In [10]: *''': information from histogram'''*

```
plt.figure(figsize=(8, 4))
bins = np.linspace(0,10,21)
hist_20 = plt.hist(xmany, bins, label='x', edgecolor='black')
outlier = xmany[np.logical_or(xmany<0, xmany>10)]
```

```

plt.scatter(outlier, np.ones_like(outlier), color='red', label='outlier')
plt.xlabel('values')
plt.ylabel('frequency')
plt.legend();

```



3.0.3 Klasseneinteilung

Sinnvoll für Informationsgehalt. Faustregel: - bis zu 20 Klassen - Anzahl Klassen $n_{\text{bins}} \equiv \sqrt{n}_{\text{data}}$ (für n_{data} bis 100 bzw. 400) - Klassenbreite $w_{\text{bins}} = \frac{3.49 \cdot \sigma}{\sqrt[3]{n_{\text{data}}}}$ (Scotts Regel) David W. Scott: On optimal and data-based histogram. Biometrika 3 66, 1979, S. 605–610

Dabei **Breite** der Klassenbalken - möglichst konstant - sonst kodiere Anzahl in *Fläche* (nicht Höhe)

```
In [11]: '''optimal histogram according to Scott-s rule'''

plt.figure(figsize=(8, 4))
n = int(np.round(np.sqrt(min(400, xmany.shape[0]))))
# print('optimal number of bins = {:.0f}'.format(n))
w = 3.49*xmany.std()/xmany.shape[0]**0.333333333333
# print('optimal bin size = {:.3f}'.format(w))
xstart = xmany.mean() - n/2.*w
xend = xmany.mean() + n/2.*w
print('Optimal histogram has {} bins, each {} wide in the range{} .. {}'
      .format(n, np.round(w, decimals=3), np.round(xstart, decimals=2),
              np.round(xend, decimals=2)))
bins_opt = np.linspace(xstart, xend, n+1) # number of bin-edges within range
```

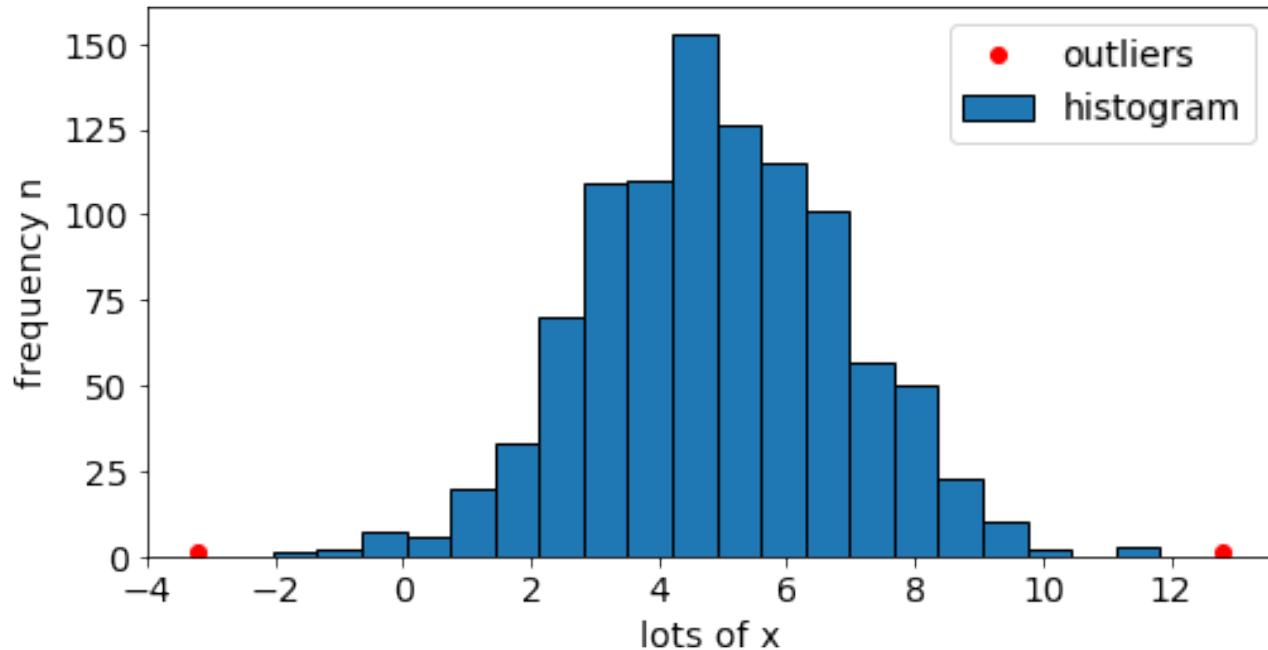
```

hist_opt = plt.hist(xmany, bins_opt, label='histogram', edgecolor='black');

xoutliers = xmany[np.logical_or(xmany>xend, xmany<xstart)]
youtliers = np.ones_like(xoutliers)
plt.plot(xoutliers, youtliers, 'ro', label='outliers')
plt.legend(loc='upper right')
plt.xlabel('lots of x')
plt.ylabel('frequency n');

```

Optimal histogram has 20 bins, each 0.693 wide in the range -2.03 .. 11.83



3.0.4 Information

- Häufigste Werte: "Modus"
- Ausreißer
- Mittelwert (=Schwerpunkt)
- Wertebereich
- Form der Verteilung

3.1 Modus

In [12]: *'once again the histogram with 20 bins'*

```

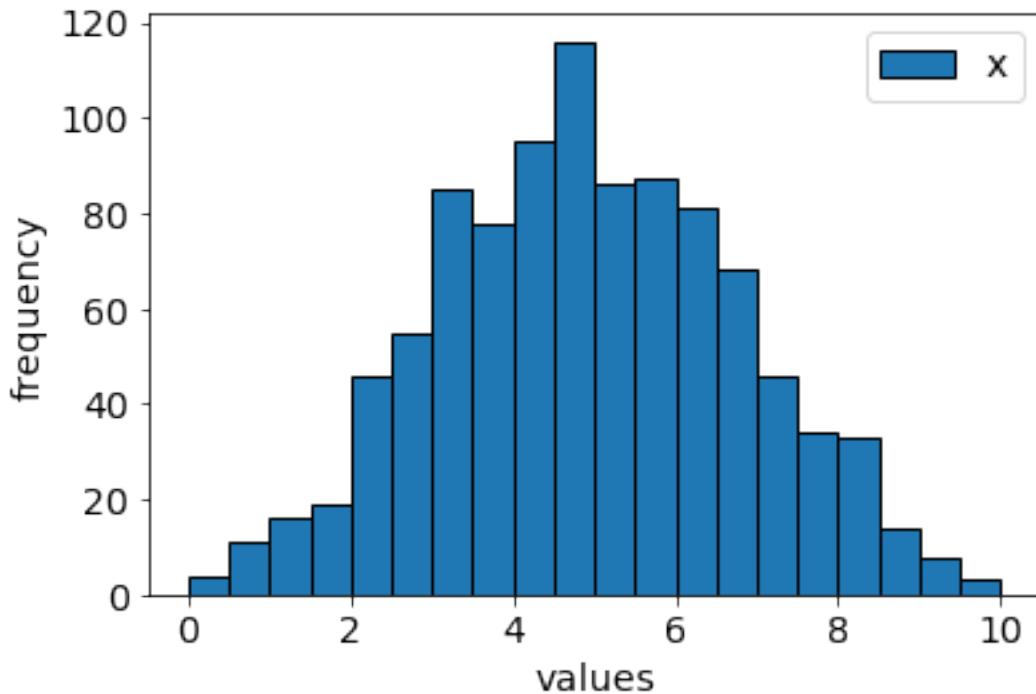
bins = np.linspace(0, 10, 21)
hist_20 = plt.hist(xmany, bins, edgecolor='black', label='x') # show histogram (again)

```

```

frequencies = hist_20[0]          # first sub-array contains frequencies
borders = hist_20[1]              # 2nd sub-array contains bin borders
fmax = max(frequencies)          # maximum frequency
imax = np.argmax(frequencies)    # ... and its location
xlower = borders[imax]           # start of max-bin
xhigher = borders[imax+1]        # ... and end
plt.xlabel('values')
plt.ylabel('frequency')
plt.legend(loc='upper right');

```



In [13]: `print(hist_20)`

```
(array([ 4.,  11.,  16.,  19.,  46.,  55.,  85.,  78.,  95., 116.,  86.,
       87.,  81.,  68.,  46.,  34.,  33.,  14.,   8.,   3.]), array([ 0. ,  0.5,  1. ,  1.5,  2. ,
      5.5,  6. ,  6.5,  7. ,  7.5,  8. ,  8.5,  9. ,  9.5, 10. ]), <a list of 20 Patch objects>)
```

In [14]: `'''show Modus'''`

```

hist_20 = plt.hist(xmany, bins, edgecolor='black') # show histogram (again)
frequencies = hist_20[0]                      # first sub-array contains frequencies
borders = hist_20[1]                          # 2nd sub-array contains bin borders
fmax = max(frequencies)                      # maximum frequency
imax = np.argmax(frequencies)                # ... and its location
xlower = borders[imax]                        # start of max-bin

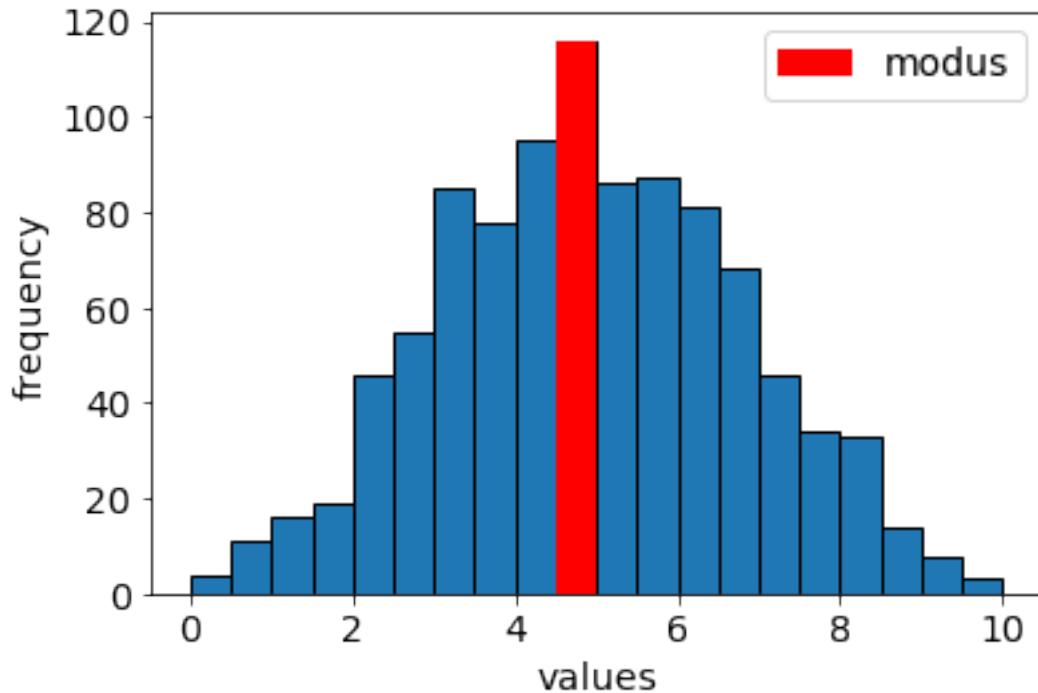
```

```

xhigher = borders[imax+1]                      # ... and end
print("The modus of 'hist20' is {} values in the bin number {}".format(fmax, imax))
print(" ranging from {:.2f} to {:.2f}".format(xlower, xhigher)) # highlight the modus-bin
plt.bar(x=xlower, width=(xhigher-xlower), bottom=0, height=fmax, color='red',
        label='modus', align='edge')
plt.xlabel('values')
plt.ylabel('frequency')
plt.legend(loc='upper right');

```

The modus of 'hist20' is 116.0 values in the bin number 9
ranging from 4.50 to 5.00



```

In [15]: from matplotlib import image as mpi          # image related routines

img = mpi.imread('data/gabor.png')                  # read image data from file
imgvector3D = np.array(img)                         # convert colors to values in matrix
# select color red only, round from 0..1 -> 0..255 integers
x = np.rint(255*imgvector3D[:, :, 0].flatten()).astype(int)

```

```

In [16]: '''Modus from Gabor-patch image'''
mybins = np.linspace(-0.5, 256.5, 257+1)
b, v = np.histogram(x, bins=mybins)
print(v[124:134])

```

```

print(b[124:133])
xm = np.argmax(b)
print('the modus of x is {} (with {} values between{} and {})'
      .format(xm, b[xm], v[xm], v[xm+1]))

```

[123.5 124.5 125.5 126.5 127.5 128.5 129.5 130.5 131.5 132.5]
[68 84 107 107 116 82 89 90 78]
the modus of x is 128 (with 116 values between127.5 and 128.5)

4 Quantile

Grenzwert, vor dem der Anteil (das Quantum) an Datenwerten liegt.

4.0.1 Quartile

- Unteres Quartil $x_{0.25}$: 25% und
- Oberes Quartil $x_{0.75}$: 75% der Werte
- Median entspricht dem 50% Quartil

Interquartils-Abstand $d_Q = x_{0.75} - x_{0.25}$

Ein StreuungsmaSS, robust gegen AusreiSSer

4.0.2 Perzentile

- 5%-Perzentil: niedrigste 5% der Werte
- 95%-Perzentil: ohne oberste 5% der Werte

Quantile sind einfach abzulesen aus der kumulativen Verteilungsfunktion:

5 Empirische kumulative Verteilungsfunktion F(x)

- Relative Häufigkeiten $h_i = \frac{n_i}{N}$
 - beispielsweise aus einem gebinnten Histogramm
- Reihenfolge gemäSS Wert beibehalten
- aufstapeln

```

In [17]: '''make a cumulative histogram cdf'''
hist_cumul = plt.hist(xmany, bins=np.linspace(0, 10, 21), cumulative=True,
                      edgecolor='black')
plt.title('cumulative distribution function')
print('histogram {} ... {}'.format(hist_20[0][:5], hist_20[0][-4:]))
print('cumulated {} ... {}'.format(hist_cumul[0][:5], hist_cumul[0][-4:]))

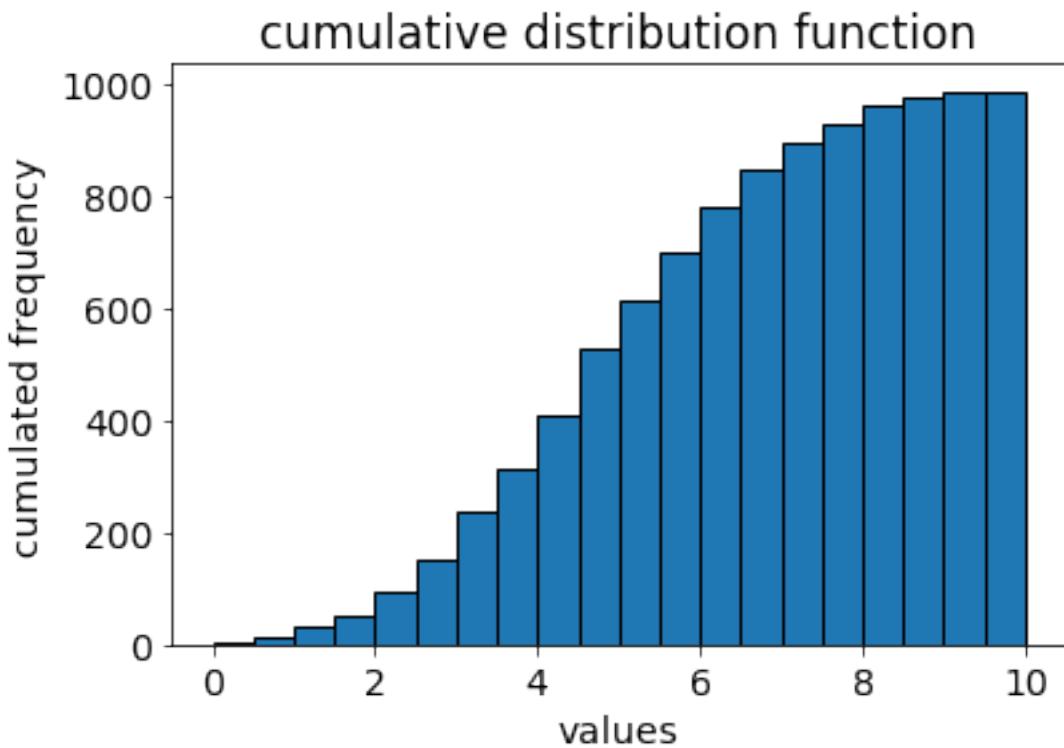
```

```

plt.xlabel('values')
plt.ylabel('cumulated frequency');

histogram [ 4. 11. 16. 19. 46.] ... [33. 14. 8. 3.]
cumulated [ 4. 15. 31. 50. 96.] ... [960. 974. 982. 985.]

```



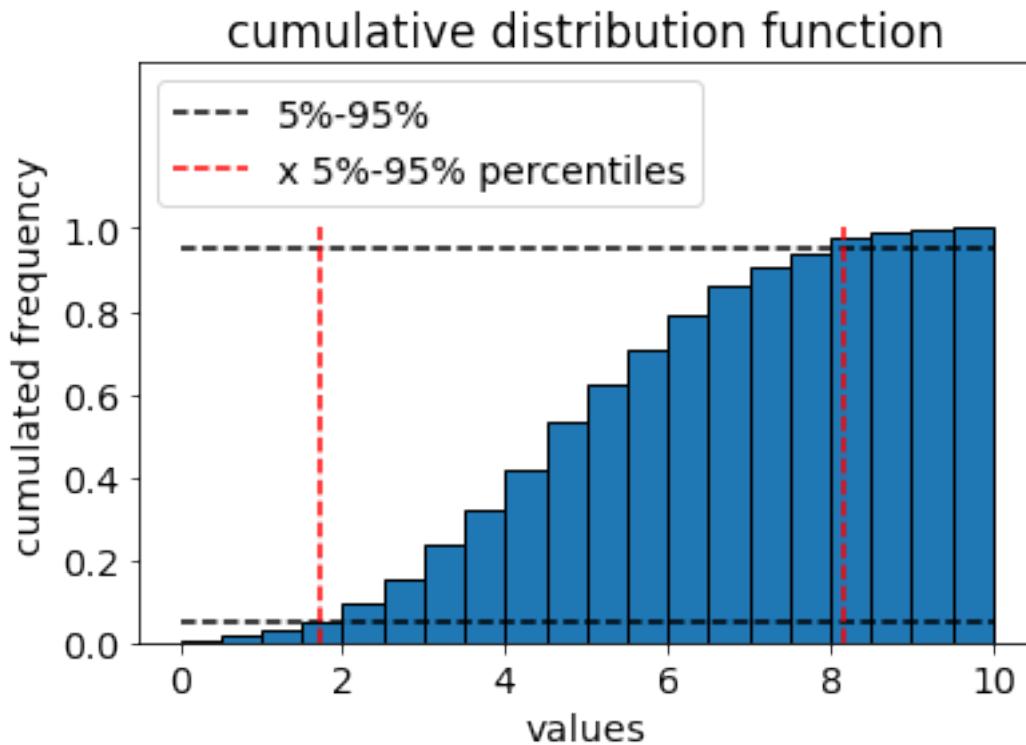
```

In [18]: '''calculate quantiles'''

hist_cumul = plt.hist(xmany, bins=np.linspace(0, 10, 21), cumulative=True,
                      density=True, edgecolor='black')
plt.title('cumulative distribution function')
plt.ylim(0.0, 1.4)
plt.yticks(list(np.arange(0.0, 1.01, 0.2)))
x05, x95 = np.percentile(xmany, (5, 95))
print(' 5% percentile = {:.2f} ... 95% percentile = {:.2f}'.format(x05, x95))
plt.plot([0, 10], 2*[.05], 'k--', label='5%-95%')
plt.plot([0, 10], 2*[.95], 'k--')
plt.plot(2*[x05], [0, 1], 'r--', label='x 5%-95% percentiles')
plt.plot(2*[x95], [0, 1], 'r--')
plt.xlabel('values')
plt.ylabel('cumulated frequency')
plt.legend(loc='upper left');

```

5% percentile = 1.72 ... 95% percentile = 8.15



5.0.1 Schnelles Ablesen der Information:

- Lage: Median, (Modus)
- Bereich: Quartile, Perzentile
- Form der Verteilung: S-Kurve
 - wenn Verteilung *ein* Maximum hat (“unimodal”)
 - Modus: 增长最大的
 - Media: 中位数

6 Kumulierter Mittelwert

6.1 Arithmetisches Mittel

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

$$\approx \bar{x}' = \frac{1}{\sum_{k=1}^{N_k} h_k} \sum_{k=1}^{N_k} h_k(x_k) \cdot x_k$$

7 Varianz

$$\begin{aligned}\sigma^2 &= \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2 \\ \approx \sigma'^2 &= \frac{1}{\sum_{k=1}^{N_k} h_k - 1} \sum_{k=1}^{N_k} h_k(x_k) \cdot (x_k - \bar{x})^2\end{aligned}$$

7.1 Standardabweichung

$$\begin{aligned}s &= \sqrt{\sigma^2} \\ s' &= \sqrt{\sigma'^2}\end{aligned}$$

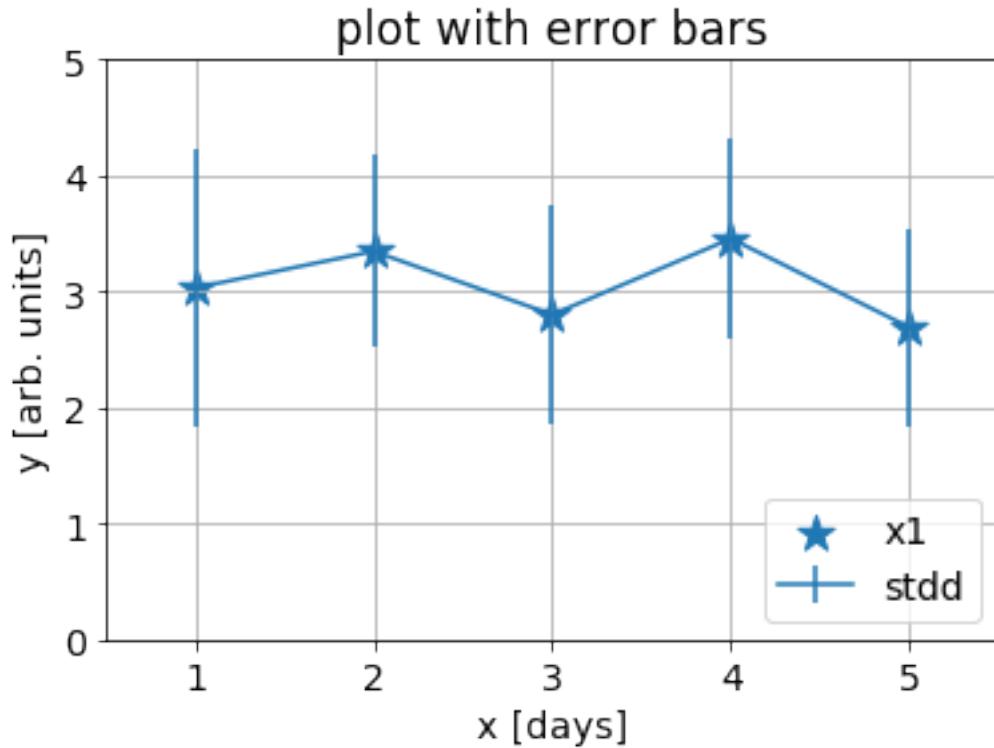
8 Graphische Darstellung

8.1 Fehlerbalken

- arithmetischer Mittelwert als Symbol
 - Siehe http://matplotlib.org/api/markers_api.html
- Fehlerbalken +/-Standardabweichung σ
- Ausreißer als Sterne

```
In [19]: '''data graph typical in science
           error bars are standard deviations
           '''

N = 5
x = 1 + np.arange(N)                                # fixed x values
y = 3 + np.random.randn(N, 20)                      # 20 random y values for each x
plt.axis([0.5, 5.5, -0, 5.])                       # range of plot
plt.errorbar(x, np.mean(y, axis=1), yerr=np.std(y, axis=1), label='std')
plt.scatter(x, np.mean(y, axis=1), marker='*', s=200, label='x1');
plt.title('plot with error bars')
plt.xlabel('x [days]')
plt.ylabel('y [arb. units]')
plt.grid(b=True, which=u'major', axis=u'both')
plt.legend(loc='lower right');
```



8.1.1 Gute Praxis

- Achsen beschriften: `xlabel, ylabel`
- *ehrliche* Achsen
 - 0 mit einbeziehen, wenn absolute Zahlen
 - sonst deutlich kennzeichnen (Lücke)
 - entsprechende Zahlen bzw. tickmarks: `xaxis.set_ticklabels`
 - sinnvollen Bereich: `axis`
- Überschrift: `title`
 - in Veröffentlichungen Bildunterschrift mit Text; keine Überschrift
- Graphen beschriften: `label` und `legend`
- deutlich unterscheiden (auch für schwarz-weiSS-Druck)
 - Symbole: `o . , p < ...`
 - Linienstil: `- -- .. -.`
 - Farbe wenn nötig
- nicht überfrachten
- manchmal hilfreich
 - Gitterlinien: `grid`
 - Spiegelstriche: `ax.tick_params(axis='y', direction='out')` und `ax.yaxis.tick_left()`

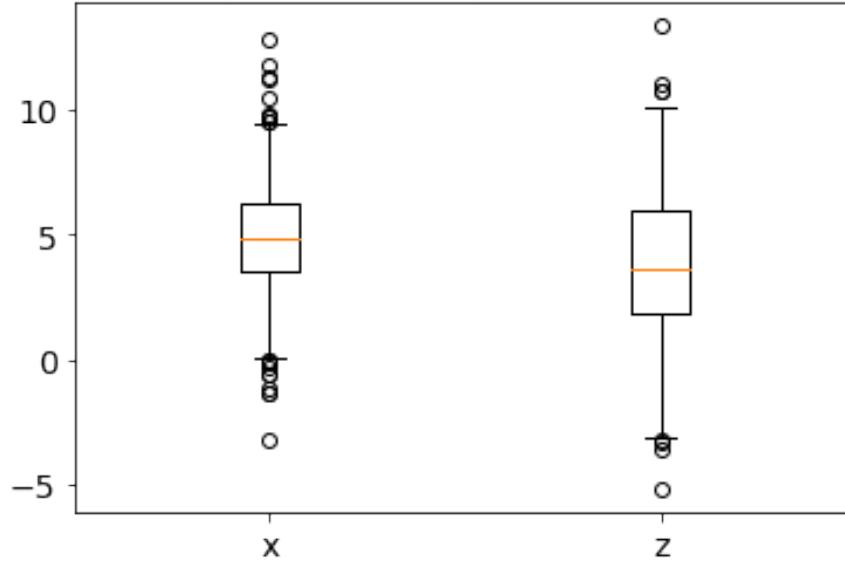
8.2 Box-Plot

- Median als Strich
- Größe der Box:
 - Quartile $Q_{1/4} \dots Q_{3/4}$
- Länge der Striche:
 - $Q_{1/4} - \frac{3}{2} \cdot (Q_{3/4} - Q_{1/4}) \dots Q_{3/4} + \frac{3}{2} \cdot (Q_{3/4} - Q_{1/4})$
 - Merkregel: Werte innerhalb “ $\pm 100\%$ ”-Quartile
- Ausreißer als Sterne +

In [20]: *'''data graph with boxes and whiskers show inter-quartile-distances'''*

```
np.random.seed(9876543)
zmany = np.random.normal(loc=4., scale=3., size=400)# another data set
data = [xmany, zmany] # combine 2 data sets
plt.boxplot(data, labels=['x', 'z'], whis=[1, 99]) # name labels; 1%-99% percentiles
plt.title('box plot with inter-quartile box, percentile whiskers and outliers');
```

box plot with inter-quartile box, percentile whiskers and outliers

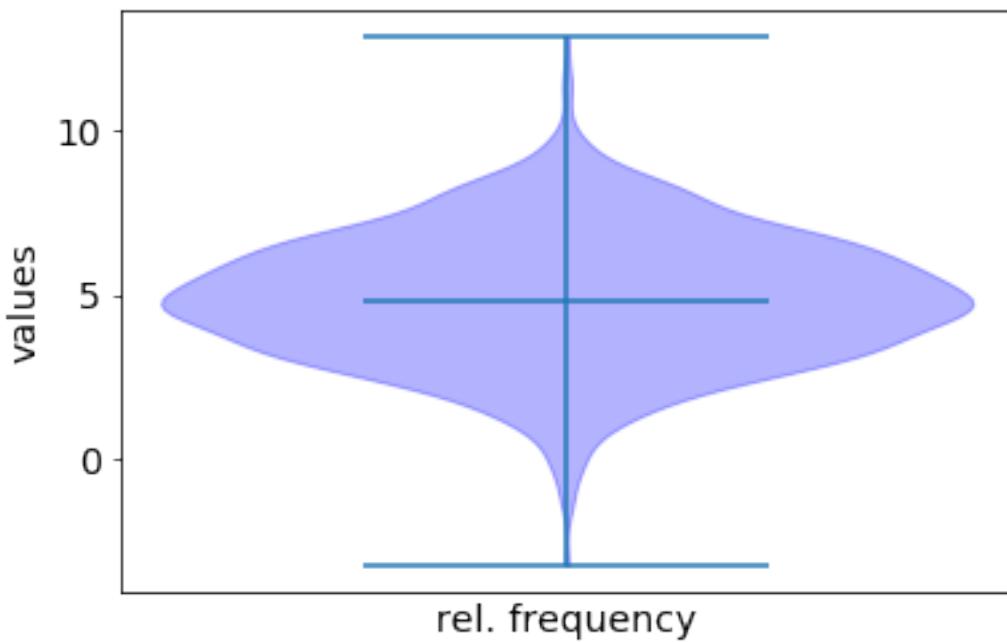


plt.boxplot?

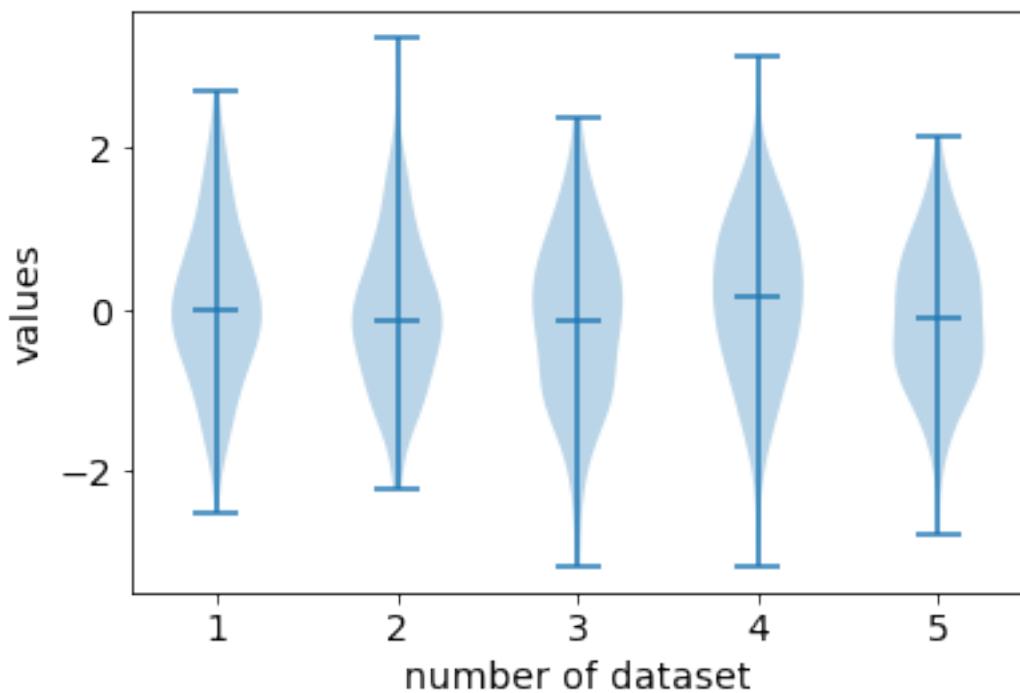
8.3 Violinen-Plot

In [21]: *vp = plt.violinplot(xmany, showmedians=True)*
vp['bodies'][0].set_color('blue')
plt.ylabel('values')

```
plt.xlabel('rel. frequency')
plt.xticks([]); # off, numbers of no meaning here
```



```
In [22]: xx = np.random.normal(size=(5, 100)) # draw 5x100 values into 2D vector
plt.violinplot([x for x in xx], showmedians=True)
plt.ylabel('values')
plt.xlabel('number of dataset');
```



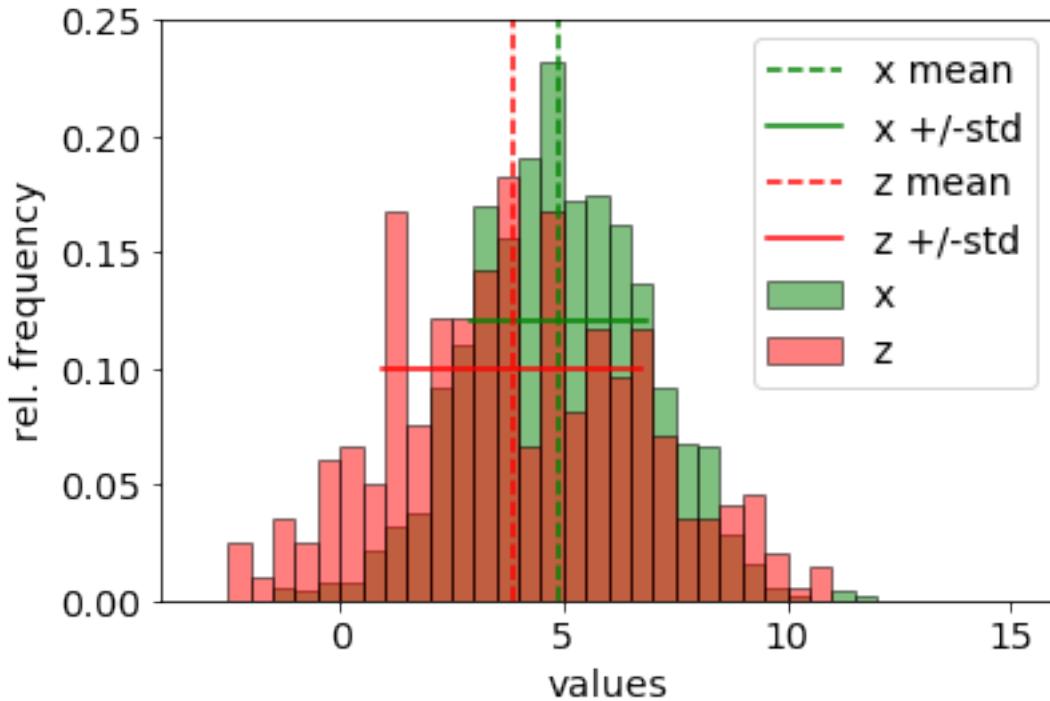
9 Fragen?

10 Vergleich zweier Stichproben

- Kennwerte
- Histogramm-Stapel
- kumulierte Verteilung

```
In [23]: '''compare characteristic numbers / histograms of two distributions'''
bins = np.linspace(-3, 12, 31)                                     # half steps
xmean = xmany.mean()
zmean = zmany.mean()
xstd = np.std(xmany)
zstd = np.std(zmany)
plt.axis([-4, 16, 0.0, 0.25])
print('x mean = {:.2f}  std = {:.2f}'.format(xmean, xstd))
plt.hist(xmany, bins=bins, color='green', density=True, edgecolor='black',
         label='x', alpha=.5)    # 1st histogram x, shine through
print('z mean = {:.2f}  std = {:.2f}'.format(zmean, zstd))
plt.hist(zmany, bins=bins, color='red', density=True, edgecolor='black',
         label='z', alpha=.5)    # 2nd histogram z, shine through
plt.plot(2*[xmean], [0, .25], 'g--', label='x mean')
plt.plot([xmean-xstd, xmean+xstd], [.12, .12], 'g-', label='x +/-std')
plt.plot(2*[zmean], [0, .25], 'r--', label='z mean')
plt.plot([zmean-zstd, zmean+zstd], [.1, .1], 'r-', label='z +/-std')
plt.xlabel('values')
plt.ylabel('rel. frequency')
plt.legend();

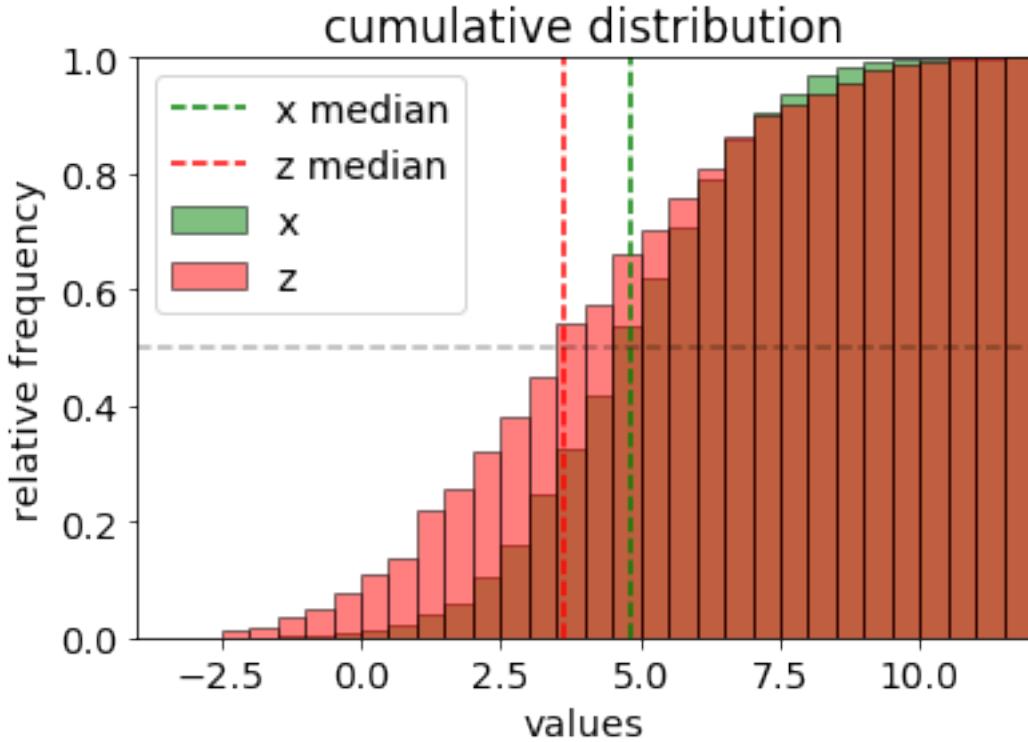
x mean = 4.90  std = 1.99
z mean = 3.84  std = 2.91
```



```
In [24]: '''two distributions
culmulative & normalized for better comparison
'''

xmedn = np.median(xmany)
zmedn = np.median(zmany)
print('x median = {:.2f} z median = {:.2f}'.format(xmedn, zmedn))
plt.hist(xmany, bins=bins, color='green', density=True, edgecolor='black',
         cumulative=True,
         label='x', alpha=.5)    # 1st histogram x, cumulative density
plt.hist(zmany, bins=bins, color='red', density=True, edgecolor='black',
         cumulative=True,
         label='z', alpha=.5)    # 2nd histogram z, cumulative density
plt.title('cumulative distribution')
plt.xlabel('values')
plt.ylabel('relative frequency')
plt.plot(2*[xmedn], [0, 1.0], 'g--', label='x median')
plt.plot(2*[zmedn], [0, 1.0], 'r--', label='z median')
plt.plot([-4, 12], 2*[0.5], 'k--', alpha=.3)
plt.axis([-4, 12, 0., 1.])
plt.legend(loc='upper left');

x median = 4.83  z median = 3.65
```



10.1 QQ-Plot

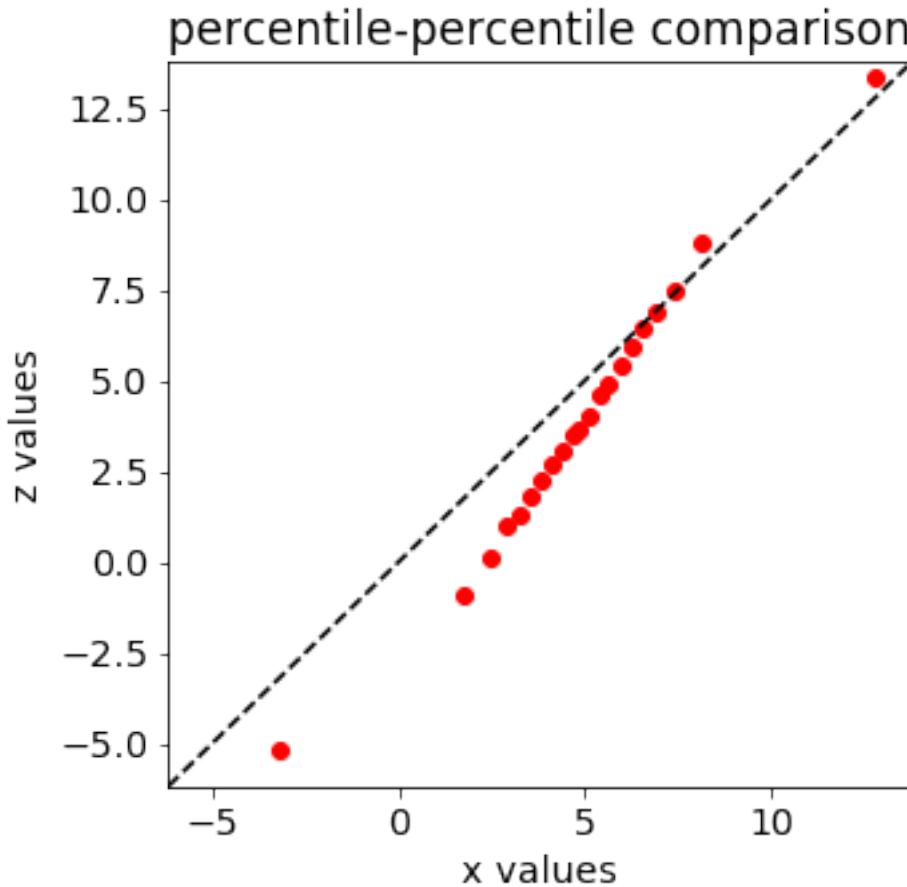
Gegenüberstellung der Quantile

In [26]: *'''compare two distributions by percentiles'''*

```

percentiles = np.linspace(0, 100, 20+1)      # 5% steps
percX = np.percentile(xmany, percentiles)
percZ = np.percentile(zmany, percentiles)
fig = plt.figure(figsize=(5, 5))
plt.plot(percX, percZ, 'ro')
diag = (np.min((xmany.min(), zmany.min()))-1, np.max((xmany.max(), zmany.max()))+1)
plt.plot(diag, diag, 'k--')
plt.xlabel('x values')
plt.ylabel('z values')
plt.xlim(diag)
plt.ylim(diag)
plt.title('percentile-percentile comparison');

```



11 Wertebereiche

- Nominal
 - Bsp: Münze: Kopf/Zahl, gezogene Farbe
- ordinale Daten
 - Bsp: Schadensklasse, Zufriedenheit "sehr, ja, nein, garnicht"
 - Kategorien
- Diskrete Daten
 - Bsp: gewürfelte Zahl, Alter
- Kontinuierliche Daten
 - Bsp: Temperatur, Alter,
 - prozentuale Werte

11.0.1 Weitere Unterscheidung für nominale und ordinale Daten:

- dichotom/binär/binomial

- polytom/multinomial

Mittels Klasseneinteilung diskretisieren und in Histogramm darstellen.

12 Zusammenfassung Kennzahlen

12.0.1 Lageparameter

- Median
- Modus
- Arithmetisches Mittel
- Harmonisches Mittel, ...
- ⇒ (wichtigste) erste Kennzahl

12.0.2 Streuung, Breite der Verteilung:

- Wertebereich Min-Max
- Standardabweichung
- Quartile
- Perzentile
- ⇒ weitere Kennzahl(en) beschreiben die Streuung

12.0.3 Ausblick: Form der Verteilung

```
In [27]: '''Python reports some main parameters'''

    # import scipy.stats
    print('mean = {:.2f} median = {:.2f}'
          .format(xmany.mean(), np.median(xmany)))
    print('standard deviation = {:.2f}'
          .format(xmany.std()))
    print('quartile range = {:.2f} to {:.2f}'
          .format(np.percentile(xmany, 25), np.percentile(xmany, 75)))
    print('1% - 99% range = {:.2f} to {:.2f}'
          .format(np.percentile(xmany, 1), np.percentile(xmany, 99)))

mean = 4.90 median = 4.83
standard deviation = 1.99
quartile range = 3.53 to 6.27
1% - 99% range = 0.05 to 9.41
```

13 Zusammenfassung Graphik

- Mittelwert und Streuung - “Fehlerbalken-Diagramm”

- Boxplot
- Violinenplot
- Histogramm
- kumulierte Verteilungsfunktion

14 Ausblick

- mehrdimensionale Daten
- SchlieSSende Statistik arbeitet mit Wahrscheinlichkeiten
- Stochastische Modelle

15 Fragen?

021_Folien

November 23, 2018

```
In [5]: from matplotlib import pyplot as plt
        import numpy as np
        %matplotlib inline
```

1 Beschreibende Statistik

1.1 Übersicht

LagemaSS

Streuung

Graphiken

Mehrdimensionale Daten

- Diskrete Verteilung
- Kontinuierliche Verteilung

Abhängigkeit

1.2 Bivariate Stichproben

Beispiele:

- zwei Würfel
- Strom und Spannung
- Luftdruck und Höhe über Meer
- Regentropfen auf Blatt Papier (Koordinaten x und y)

2 Diskrete Verteilung

2.0.1 Kontingenztafel der *absoluten Häufigkeiten*

```
In [6]: print('== two coins ==')
results = [[0, 1], [1, 1], [1, 1], [0, 0], [1, 0], [1, 1], [1, 1], [0, 1],
           [1, 1], [1, 0], [0, 1], [0, 0], [0, 1], [1, 0], [1, 0], [0, 0],
           [0, 1], [0, 0], [1, 1], [0, 0]]
print('#1 \ #2 | head | tail |')
print('-----')                                     # calculate numbers
print('head    |{:5d} |{:5d} |'.format(results.count([0, 0]), results.count([0, 1])))
print('tail    |{:5d} |{:5d} |'.format(results.count([1, 0]), results.count([1, 1])))
```

```

==== two coins ====
#1 \ #2 | head | tail |
-----
head   |    5 |    5 |
tail   |    4 |    6 |

```

2.0.2 ... mit Randverteilung

```
In [7]: '''calculate absolute frequencies (Python list)'''
n11 = results.count([0, 0])
n12 = results.count([0, 1])
n21 = results.count([1, 0])
n22 = results.count([1, 1])
print('#1 \ #2| head | tail ||  sum ')
print('-----')
print('head   |{:5d} |{:5d} ||{:5d} '.format( n11,      n12,      n11+n12))
print('tail   |{:5d} |{:5d} ||{:5d} '.format( n21,      n22,      n21+n22))
print('=====')
print('sum     |{:5d} |{:5d} ||{:5d} '.format(n11+n21, n12+n22, n11+n12+n21+n22))

#1 \ #2| head | tail ||  sum
-----
head   |    5 |    5 ||    10
tail   |    4 |    6 ||    10
=====
sum     |    9 |   11 ||    20
```

```
In [8]: '''calculate absolute frequencies (numpy ndarray)'''
npres = np.array(results)           # convert data to ndarray
n11 = (npres==[0, 0]).all(axis=1).sum() # comparing (2val)-array => boolean array
n12 = (npres==[0, 1]).all(axis=1).sum() # ... only if both are True ("all")
n21 = (npres==[1, 0]).all(axis=1).sum() # ... then count as "1" (else False="0")
n22 = (npres==[1, 1]).all(axis=1).sum() # ... summing up these gives total number
print('#1 \ #2| head | tail ||  sum ')
print('-----')
print('head   |{:5d} |{:5d} ||{:5d} '.format( n11,      n12,      n11+n12))
print('tail   |{:5d} |{:5d} ||{:5d} '.format( n21,      n22,      n21+n22))
print('=====')
print('sum     |{:5d} |{:5d} ||{:5d} '.format(n11+n21, n12+n22, n11+n12+n21+n22))

#1 \ #2| head | tail ||  sum
-----
head   |    5 |    5 ||    10
tail   |    4 |    6 ||    10
=====
sum     |    9 |   11 ||    20
```

2.0.3 relative Häufigkeiten

$$f_{ij} = \frac{h_{ij}}{n} \quad f \in [0, 1]$$

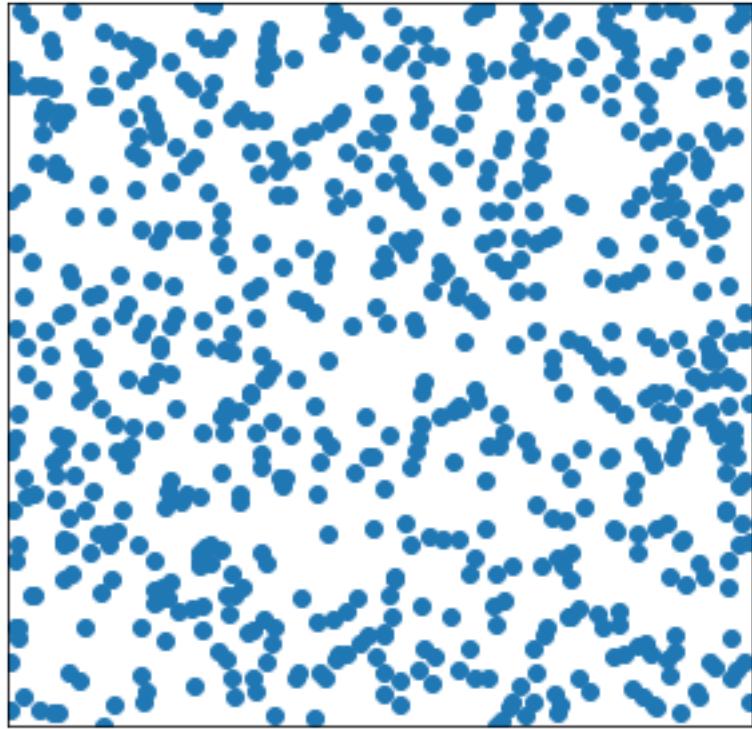
```
In [9]: '''calculate relative frequencies (numpy ndarray to list)'''
nprel = np.array(results)      # place holder for calculating relative frequency
n = nprel.shape[0]              # number of value(pairs): 1st dimension of shape
f11 = nprel.tolist().count([0, 0])/n      # since a list again,
f12 = nprel.tolist().count([0, 1])/n      # ... we can count
f21 = nprel.tolist().count([1, 0])/n      # ... different elements
f22 = nprel.tolist().count([1, 1])/n      # ... as before
print('#1 \ #2| head | tail || sum ')
print('-----')
print('head    |{:5.2f} |{:5.2f} ||{:5.2f} '.format( f11,      f12,      f11+f12))
print('tail    |{:5.2f} |{:5.2f} ||{:5.2f} '.format( f21,      f22,      f21+f22))
print('===== ')
print('sum     |{:5.2f} |{:5.2f} ||{:5.2f} '.format(f11+f21, f12+f22, f11+f12+f21+f22))

#1 \ #2| head | tail || sum
-----
head    | 0.25 | 0.25 || 0.50
tail    | 0.20 | 0.30 || 0.50
=====
sum     | 0.45 | 0.55 || 1.00
```

3 Kontinuierliche Verteilung

```
In [8]: '''Two dimensional random variable: raindrops on a paper'''
f = plt.figure(figsize=(5, 5))          # square
np.random.seed(987654)                  # initialize random generator to same
rain = np.random.random((2, 700))        # draw 2x700 [0...1] random numbers
plt.scatter(rain[0], rain[1])           # let it rain
plt.axis([0, 1, 0, 1])                 # (restrict frame)
plt.title('rain drops keep falling on my paper') # (headline)
ax = plt.gca()                         # (get instance of axes drawn)
ax.axes.get_xaxis().set_visible(False)  # (no tickmarks on x)
ax.axes.get_yaxis().set_visible(False); # ( and y axis)
```

rain drops keep falling on my paper

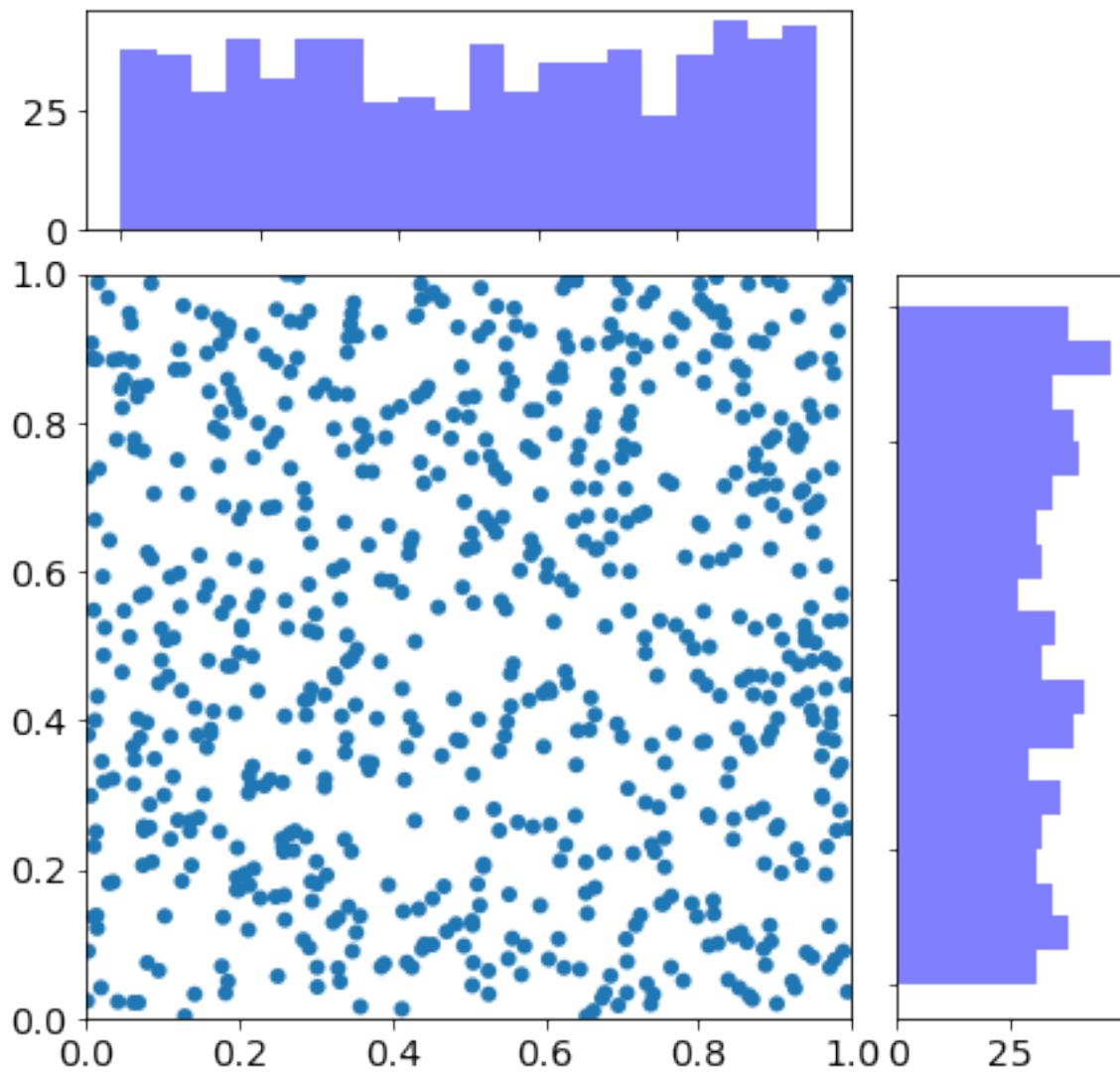


4 Streudiagramm mit Randverteilung

der einzelnen Komponenten (Dimensionen)

```
In [10]: '''three sub-graph-axis-sets: axmx, axmy and axxy in one figure'''

fig = plt.figure(figsize=(7, 7))
bins = np.linspace(0.0, 1.0, 21) # 20 bins from 0 to 1
# define a 4x4 grid and distribute the virtual squares:
# upper marginal 3x1, until 3rd column
axmx = plt.subplot2grid((4, 4), (0, 0), colspan=3)
# right marginal 1x3, start 2nd row, last col
axmy = plt.subplot2grid((4, 4), (1, 3), rowspan=3)
# main window, size 3x3, start 2nd row
axxy = plt.subplot2grid((4, 4), (1, 0), colspan=3, rowspan=3)
# x-marginal histogram
axmx.hist(rain[0], color='b', bins=bins, label='x', alpha=.5)
axmy.hist(rain[1], color='b', bins=bins, label='y', alpha=.5,
           orientation='horizontal') # y-marginal, rotated
axxy.scatter(rain[0], rain[1], edgecolor='') # let it rain in big xy-panel
axmx.xaxis.set_ticklabels([]) # no tickmarks
axmy.yaxis.set_ticklabels([])
axxy.axis([0, 1, 0, 1]); # restrict area to full data range
```



5 Kenngrößen mehrdimensionaler Daten

5.1 Mittelwert

Randverteilung erlaubt Bestimmung des 2D-Mittelwerts

5.2 Varianz

Randverteilung erlaubt Bestimmung der 2D-Streuung(en)

es sollt auch in 2D

```
In [11]: print('the mean of rain-X is {:.2f}'.format(rain[0].mean()))
      print('the mean of rain-Y is {:.2f}'.format(rain[1].mean()))
```

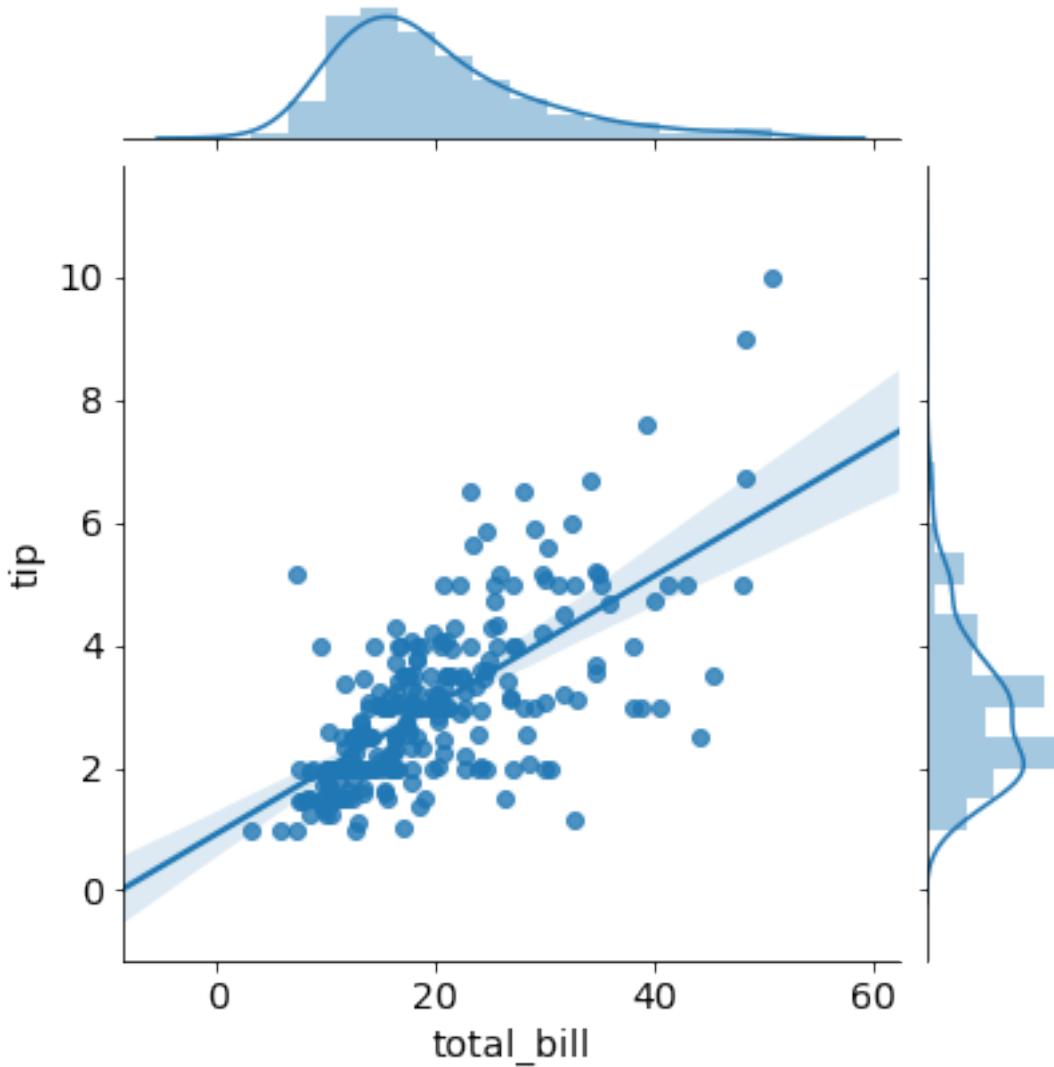
```
the mean of rain-X is 0.51
the mean of rain-Y is 0.51
```

```
In [12]: print('the variance of rain-X is {:.2f}'.format(rain[0].var()))
      print('the variance of rain-Y is {:.2f}'.format(rain[1].var()))
```

the variance of rain-X is 0.09
the variance of rain-Y is 0.09

5.3 seaborn - weitere Bibliothek zur Datenexploration

```
In [5]: '''seaborn allows easy data exploration'''
import seaborn as sns
tips = sns.load_dataset("tips") # Load one of the data sets that come with seaborn
sns.jointplot("total_bill", "tip", tips, kind='reg');
```

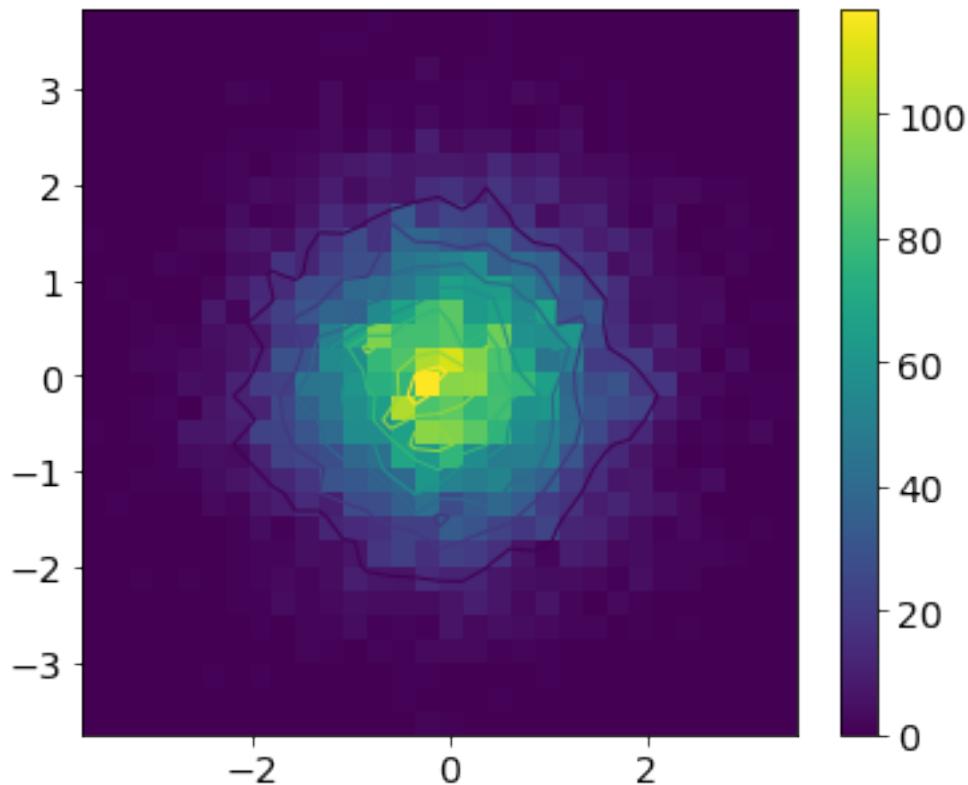


5.4 Dichte-Verteilung: 2D-Histogramm

Bei zu vielen Datenpunkten gibt es analog zum eindimensionalen Fall *Histogramm* die Möglichkeit in die dritte Dimension z die Häufigkeitsverteilung aufzutragen.

Matplotlib hat dazu `hist2d` farbkodiert.

```
In [6]: '''two dimensional density plot'''
x1 = np.random.normal(size=10000) # random x values
x2 = np.random.normal(size=10000) # random y values
f = plt.figure(figsize=(6, 5))
counts, xbins, ybins, image = plt.hist2d(x1, x2, bins=[30,30]) # counts-matrix
plt.colorbar() # visualize translation color-number
plt.contour(xbins[:-1], ybins[:-1], counts.T, linewidths=1); # to use for contours
```



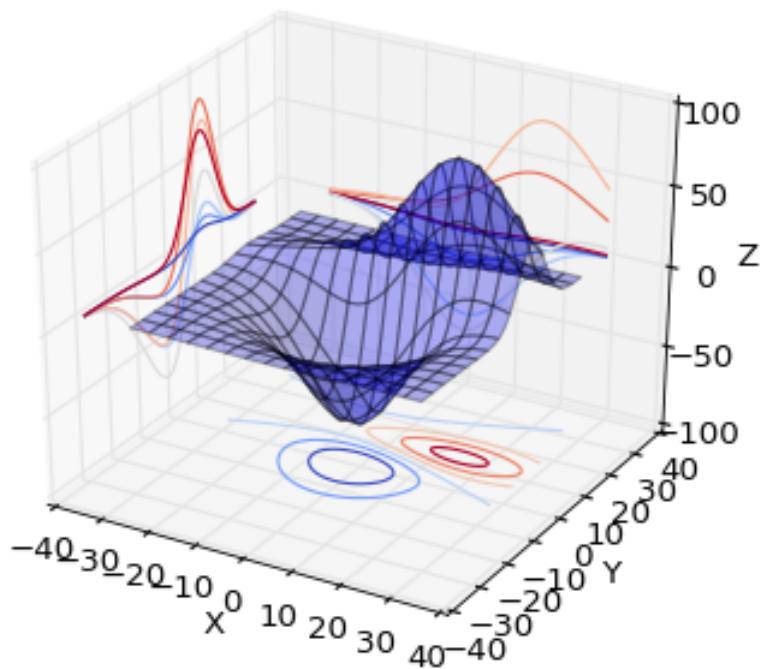
```
plt.hist2d?
```

5.5 3D-Darstellung

zweidimensionale Basis: $y = f(x_1, x_2)$ Graphische Darstellung - diskret: mit *Nadeln* - kontinuierlich: 3D-Graphik

In [13]:

Out[13]:

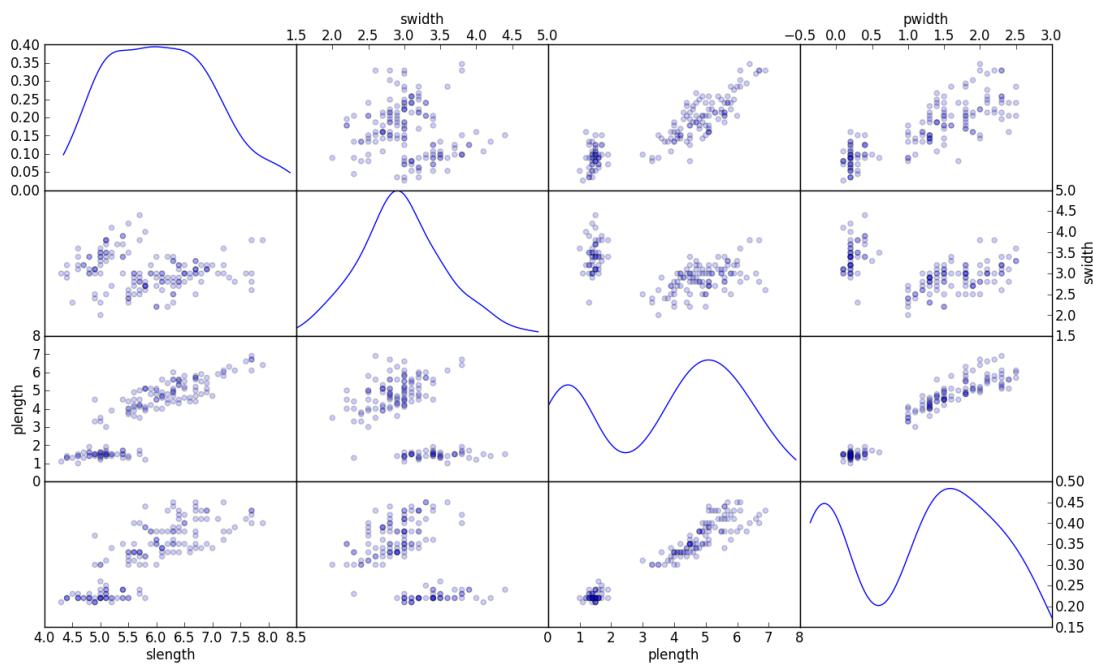


5.6 höherdimensional

- Streumatrix

In [14] :

Out [14] :



6 Zusammenfassung beschreibende mehrdimensionale Statistik

- Datenreduktion Kenngrößen
 - Mittelwerte
 - Standardabweichungen
- Struktur in den Daten erkennen
 - Form der Verteilung
 - Form der Randverteilung
- Anschauliche Darstellung
 - Kontingenztabelle, Vierfeldertafel
 - 2D-Histogramm

Ausblick:

- Abhängigkeit mehrerer Variabler
 - Korrelation
 - Regression

7 Links

- Matplotlib Graphikgalerie <http://matplotlib.org/gallery.html>
- Pandas Graphiken: <http://pandas.pydata.org/pandas-docs/stable/visualization.html>

8 Fragen?

022_Folien

2018 年 11 月 23 日

```
In [2]: from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline
```

1 Beschreibende Statistik

1.1 Übersicht

LagemaSS

Streuung

Graphiken

Mehrdimensionale Daten

Abhängigkeit

- Korrelation 相关
- Regression 回归

1.2 Mehrdimensionale Daten - Zusammenhang?

Beispiele:

- zwei Würfel: sollten unabhängig sein
- Strom und Spannung: festes Verhältnis gemäß Ohmschen Gesetzes
- Luftdruck und Höhe über Meer: Barometrische Höhenformel

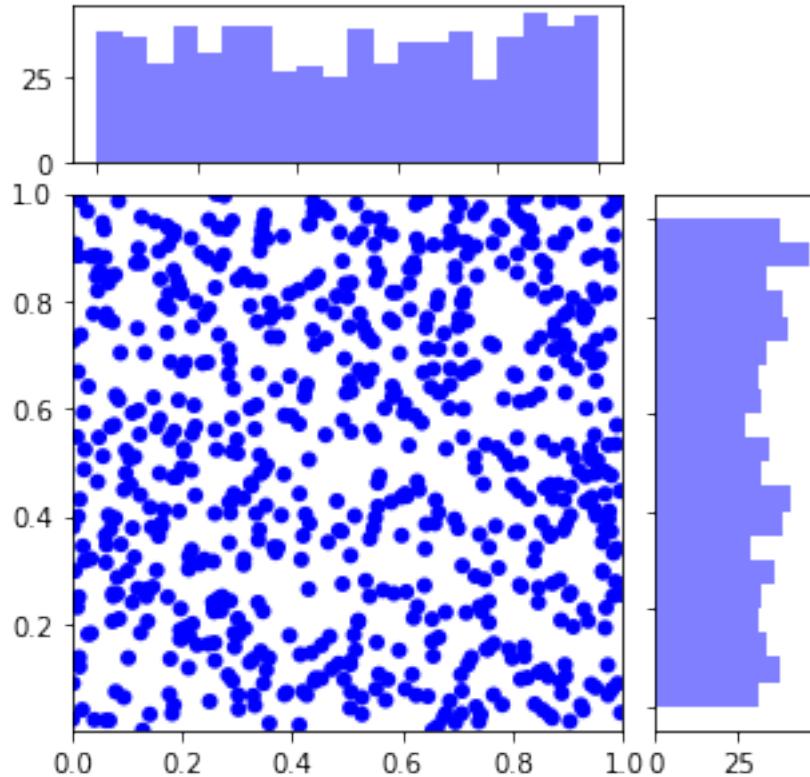
Frage: Hängt x und y zusammen?

```
In [3]: '''reshape some of the data by sorting'''
np.random.seed(987654)                      # initialize random generator to same
rain = np.random.random((2, 700))              # draw 2x700 equally distrib random numbers
xa = np.sort(rain[0])                          # sort x with numpy
xb = xa[:]                                     # make a copy of the object
ya = np.sort(rain[1])                          # sort y with numpy
n = int(rain.shape[1]/2)                        # half index
yb = np.append(ya[n:], ya[:n])                 # reorder 1st and 2nd half by split&join
```

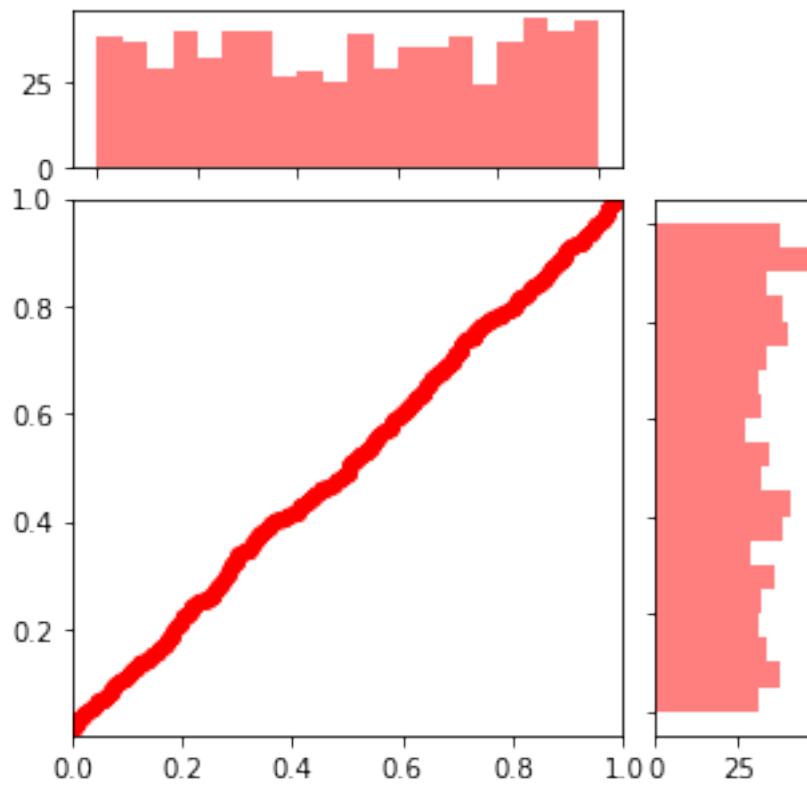
```
In [4]: '''scatterplot for two dimensional data
           together with their marginal distributions'''

def xywithmarginals(x, y, col):
    fig = plt.figure(figsize=(5, 5))                      # square canvas
    bins = np.linspace(0.0, 1.0, 21)                      # 20 bins from 0 to 1
    # define a 4x4 grid and distribute the virtual squares:
    # upper marginal 3x1, until 3rd col.
    axmx = plt.subplot2grid((4, 4), (0, 0), colspan=3)
    # right marginal 1x3, start 2nd/last
    axmy = plt.subplot2grid((4, 4), (1, 3), rowspan=3)
    # main window, size 3x3, skip 1st row
    axxy = plt.subplot2grid((4, 4), (1, 0), colspan=3, rowspan=3)
    axmx.hist(x, color=col, bins=bins, label='x', alpha=.5) # x-marginal histogram
    axmx.xaxis.set_ticklabels([])                           # no tickmarks
    axmy.hist(y, color=col, bins=bins, label='y', alpha=.5,
              orientation='horizontal')                     # y-marginal, rotated
    axmy.yaxis.set_ticklabels([])
    axxy.scatter(x, y, color=col, edgecolor='')          # let it rain in big xy-panel
    axxy.axis([x.min(), x.max(), y.min(), y.max()]);# restrict area to full data range
```

```
In [5]: xywithmarginals(rain[0], rain[1], 'b')
```



```
In [6]: xywithmarginals(xa, ya, 'r')
```



2 Zusammenhang - Korrelation

2.1 Frage:

Je grÖSSer x desto grÖSSer y?

2.2 Gesucht

MaSS, welches Gemeinsamkeit berücksichtigt

2.3 Antwort

$$\sum_{i=1}^N x_i \cdot y_i$$

Daten zentrieren

2.4 Kovarianz

Definition Kovarianz:

$$\text{Cov}_{XY} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$$

ist die Summe der quart. ein Point hat kein Kovarianz. Es ist undefiniert.

2.4.1 Eigenschaften

Symmetrie $\text{Cov}_{XY} = \text{Cov}_{YX}$

Varianz: $\text{Var}(X) = \text{Cov}_{XX}$

In [7]: *'covariance matrix'*

```
covariance = np.cov(rain[0], rain[1]) # part of numpy
print('covariance matrix:\n{}\n'.format(covariance)) # four possible combinations
print('covariance_XY = {:.8f}'.format(covariance[0][1]))# pick x-y off diagonal

covariance matrix:
[[0.0893181  0.00214176]
 [0.00214176  0.08634365]]

covariance_XY = 0.00214176
```

In [8]: *'variance?'*

```
print('variance of x is {:.8f} and of y is {:.8f}'
      .format(rain[0].var(), rain[1].var()))

variance of x is 0.08919050 and of y is 0.08622030
```

In [10]: *'variance with: delta degree of freedom = 1'*

```
print('variance of x is {:.8f} and of y is {:.8f}'
      .format(rain[0].var(ddof=1), rain[1].var(ddof=1)))

variance of x is 0.08931810 and of y is 0.08634365
```

In [11]: `np.cov?`

`np.var?`

2.5 Produkt-Moment-Korrelation nach Pearson:

2.5.1 Vorgehen: Standardisieren

1. Zentrieren

$$x''_i = x_i - \bar{x} \quad y''_i = y_i - \bar{y}$$

2. Stauchen auf Standardabweichung

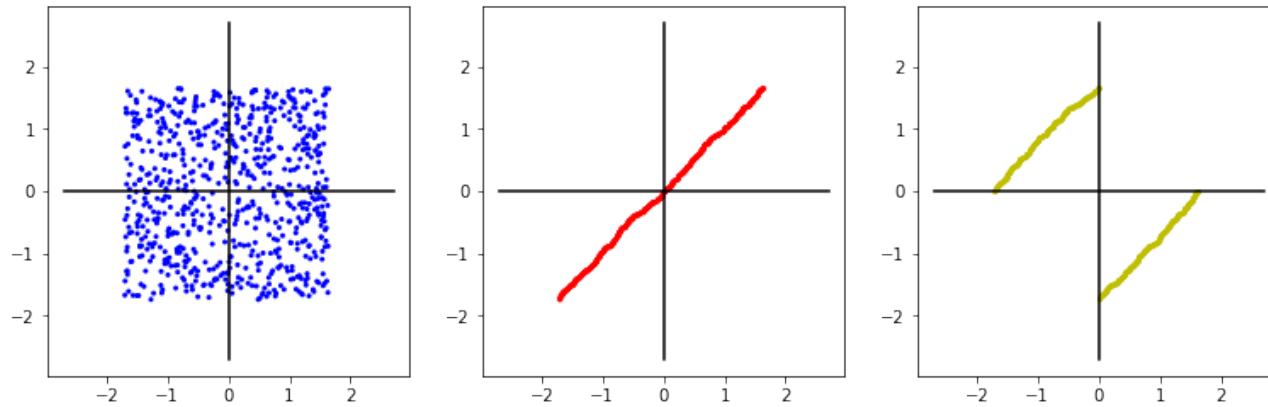
$$x'_i = \frac{x''_i}{\sigma_x} \quad y'_i = \frac{y''_i}{\sigma_y}$$

2.5.2 \Rightarrow Neue Verteilungen x' und y'

- Mittelwerte $\bar{x}' = 0$ bzw. $\bar{y}' = 0$
- Standardabweichung $\sigma'_x = 1$ bzw. $\sigma'_y = 1$.

```
In [12]: '''same data, different co-relation x_i to y_j'''

f = plt.figure(figsize=(13,4))
f.add_subplot(1,3,1)
x2 = rain[0]-rain[0].mean()
y2 = rain[1]-rain[1].mean()
x1 = x2/x2.std(ddof=1) # standardized x/y
y1 = y2/y2.std(ddof=1)
plt.scatter(x1, y1, c='b', marker='.', edgecolor='')
plt.plot((0., 0.), (-2.7, 2.7), 'k-')
plt.plot((-2.7, 2.7), (0., 0.), 'k-');
f.add_subplot(1,3,2)
x2 = xa-xa.mean()
y2 = ya-ya.mean()
x1 = x2/x2.std(ddof=1)
y1 = y2/y2.std(ddof=1)
plt.scatter(x1, y1, c='r', marker='.', edgecolor='')
plt.plot((0., 0.), (-2.7, 2.7), 'k-')
plt.plot((-2.7, 2.7), (0., 0.), 'k-');
f.add_subplot(1,3,3)
x2 = xb-xb.mean()
y2 = yb-yb.mean()
x1 = x2/x2.std(ddof=1)
y1 = y2/y2.std(ddof=1)
plt.scatter(x1, y1, c='y', marker='.', edgecolor='')
plt.plot((0., 0.), (-2.7, 2.7), 'k-')
plt.plot((-2.7, 2.7), (0., 0.), 'k-');
```



```
In [18]: x1 = rain[0]-rain[0].mean()/rain[0].std(ddof=1) # standardized x
y1 = rain[1]-rain[1].mean()/rain[1].std(ddof=1) # standardized y
print('covariance matrix is \n{}'.format(np.cov(x1, y1)))# internal already ddof=1

covariance matrix is
[[0.0893181  0.00214176]
 [0.00214176  0.08634365]]
```

2.5.3 Zusammenhang

- Quadranten I und III
- Quadranten II und IV.

... haben unterschiedliche Vorzeichen. Sinnvoll daher:

2.5.4 Definition Korrelationskoeffizient nach Pearson

$$r_{XY} = \frac{1}{N-1} \sum_{i=1}^N x'_i \cdot y'_i$$

```
In [19]: '''calculate Pearson correlation coefficient of data [x[N], y[N]]'''

def pearsonrxy(xy):
    return ((xy[0]-xy[0].mean())*(xy[1]-xy[1].mean())).sum() / (
        xy.shape[1]-1) / (xy[0].std(ddof=1) * xy[1].std(ddof=1))

print('Pearson correlation coefficient of rain is {:.6f}'.
      .format(pearsonrxy(rain)))
```

Pearson correlation coefficient of rain is 0.024389

2.5.5 Eigenschaften

- $-1 \leq r_{XY} \leq +1$
- $y = x \Rightarrow r_{XY} = +1$
- $y = -x \Rightarrow r_{XY} = -1$

2.5.6 Korrelation

In ursprünglichen Koordinaten

Korrelationskoeffizient

$$r_{XY} = \frac{\text{Cov}_{XY}}{\sigma_X \sigma_Y}$$

mit der *Kovarianz*

$$\text{Cov}_{XY} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$$

```
In [20]: '''Correlation coefficient Pearson r

    input: two data arrays to be correlated: a and b of type ndarray
    output: r_ab with ddof=1
    '''

def ccpearson( a, b ):
    cc = np.corrcoef(a, b)[0][1]      # side element of correlation matrix
    return cc

    print('Covariance rain:')
    print('original           = {:.6f}'.format(ccpearson(rain[0], rain[1])))
    print('sorted lin          = {:.6f}'.format(ccpearson(xa, ya)))
    print('sorted with offset= {:.6f}'.format(ccpearson(xb, yb)))
```

```
Covariance rain:
(original)      =  0.024389
(sorted lin)    =  0.999489
(sorted with offset)= -0.521914
```

2.6 Ergebnis Korrelation nach Pearson

$$r_{XY} = \frac{\text{Cov}_{XY}}{\sigma_X \sigma_Y}$$

mit der *Kovarianz*

$$\text{Cov}_{XY} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$$

- Bei exakter Übereinstimmung: $y = x$ ist $r = 1$.
- Bei umgekehrtem Vorzeichen aber übereinstimmendem Betrag $y = -x$ wird $r = -1$.
- Bei grober Entsprechung liegt r zwischen 0 und 1.
- Sind x und y unkorreliert, geht r gegen 0.

2.6.1 Bedeutung der Korrelation nach Pearson

- $r = +1$ für Gleichheit $y = a \cdot x$
- $r > 0$ für mittleren linearen Zusammenhang $y \sim x$
- $r < 0$ für mittleren linearen Zusammenhang $y \sim -x$
- $r = -1$ für Antikorrelation $y = -a \cdot x$
- $r = 0$ wenn *kein* (linearer) Zusammenhang

Stärke (willkürlich) nach Betrag

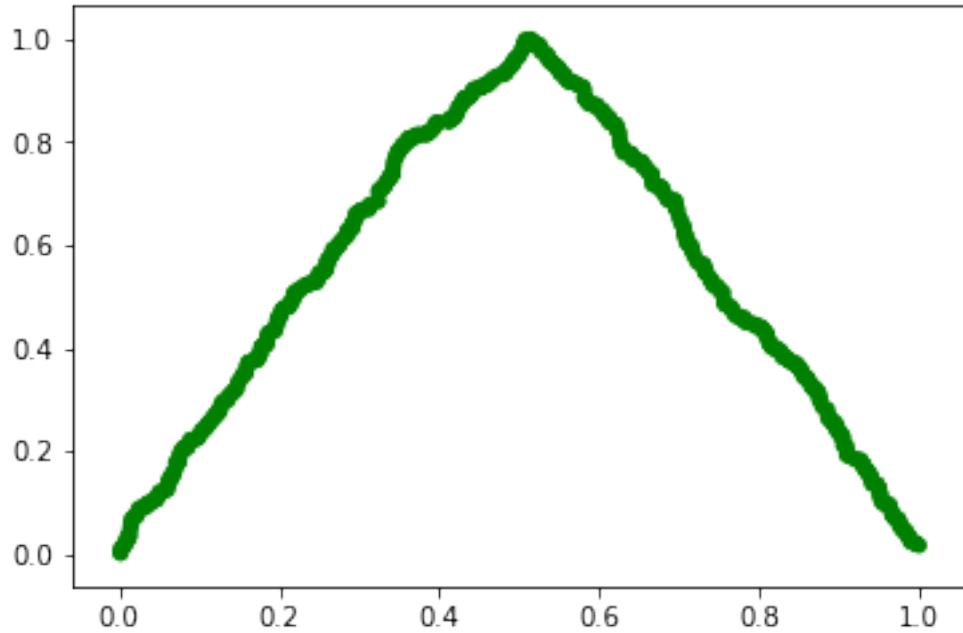
- bis 0,2 => sehr geringe Korrelation
- bis 0,5 => geringe Korrelation
- bis 0,7 => mittlere Korrelation
- bis 0,9 => hohe Korrelation
- über 0,9 => sehr hohe Korrelation

Obacht Korrelationskoeffizient trifft nur Aussage über **linearen** Zusammenhang. 相关性只针对线性相关来说

2.7 Zusammenhang nicht-linear?

```
In [21]: '''example for dependency - but no correlation'''
yc = np.append( np.sort(rain[1][:n]), np.sort(rain[1][n:])[::-1] )    # re-sort
plt.scatter(xa, yc, c='g', marker='o', edgecolor='')
print('Covariance (/)= {:.6f}'.format(ccpearson(xa, yc))) # fct defined above

Covariance (/)= -0.040828
```



2.7.1 Stückweise Korrelation

- Abschnitte
 - jede Abhängigkeit kann stückweise als linear angesehen werden
- Linearisieren
 - Aus Theorie eine Modellfunktion bilden
 - Abbilden der Daten

2.7.2 Ausblick Regression

2.7.3 Verhalten unter linearer Transformation

Unter der linearen Transformation

$$x' = a \cdot x + b$$

$$y' = c \cdot y + d$$

bleibt betragsmäßig

$$r_{x'y'} = r_{xy} = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y} = \frac{\text{Cov}(x', y')}{\sigma_{x'} \sigma_{y'}}$$

das Vorzeichen ändert sich bei unterschiedlichen Vorzeichen von a und c .

2.8 Korrelation und Kausalität

Keine Aussage über die Ursache

- $y(x)$?
- $x(y)$?
- $x(z)$ und $y(z)$?
- Zufall?

Beispiele

- Schein-Korrelation durch Versteckte Variable
 - Schulkinder: Geschicklichkeit korreliert mit KörpergröSSe
 - * Grund: versteckte Variable "Alter"
- Inhomogenitäts-Korrelation
 - SchuhgröSSe vs. Einkommen: Männer, Frauen
- Schein-Korrelation durch Ausreisser / Standardisierung
 - zufällige Variation innerhalb Streuung
- oft bei Zeitreihen

3 Fragen?

023_Folien

2018 年 11 月 23 日

```
In [1]: from matplotlib import pyplot as plt  
import numpy as np  
%matplotlib inline
```

1 Beschreibende Statistik

1.1 Übersicht

LagemaSS

Streuung

Graphiken

Mehrdimensionale Daten

Abhängigkeit

- Korrelation
- Regression
 - linear
 - polynomial

1.2 Mehrdimensionale Daten - Regression

Korrelation

- Voraussetzung: Zusammenhang von zusammengehörigen Daten ist nachgewiesen
- Korrelationskoeffizient $\neq 0$

Beispiele:

- Strom und Spannung: festes Verhältnis gemäß Ohmschen Gesetzes
- Luftdruck und Höhe über Meer: Barometrische Höhenformel

1.3 Frage: Wie stark hängt y von x ab?

Beispiel: Fahrstrecke x und Fahrzeit t

- Zusammenhang: je mehr Zeit t , desto weiter die Strecke s
 - linear $x \sim t$
 - Proportionalitätsfaktor "Geschwindigkeit v ": $x = v \cdot t$
- Für $x = 10\text{km}$ benötigen Sie $t = 12\text{Minuten}$

In [4]: *'''Python as a pocket calculator: calculate car speed'''*

```
xmin = 12                      # time in minutes
xh = xmin/60                     # time in hours
y = 10                           # distance in km
v = y / xh                       # speed in km/h
print('Going {} km in {} mins, your speed is {} km/h'.format(y, xmin, v))
```

Going 10 km in 12 mins, your speed is 50.0 km/h

Anwendung Bei linearer Abhängigkeit $x = v \cdot t$ erlaubt der Proportionalitätsfaktor Vorhersagen

Beispielsweise: wie lang ist die zurückgelegte Strecke in 1 Sekunde?

$$x = v \cdot t = \frac{10\text{km}}{10\text{min}} \cdot 1\text{s} = \frac{10000\text{m}}{10 \cdot 60\text{s}} \cdot 1\text{s} = 16.7\text{m}$$

Ergebnis: In der Reaktionszeit von 1s legt ein Auto im Stadtverkehr 17m zurück

1.3.1 Mehrere Messungen

Mehrere (n) Messungen y_i zu verschiedenen Werten der x -Variable, hier Schrittperiodendauer (Zeiten, Längen, Alter, Höhe, ...)

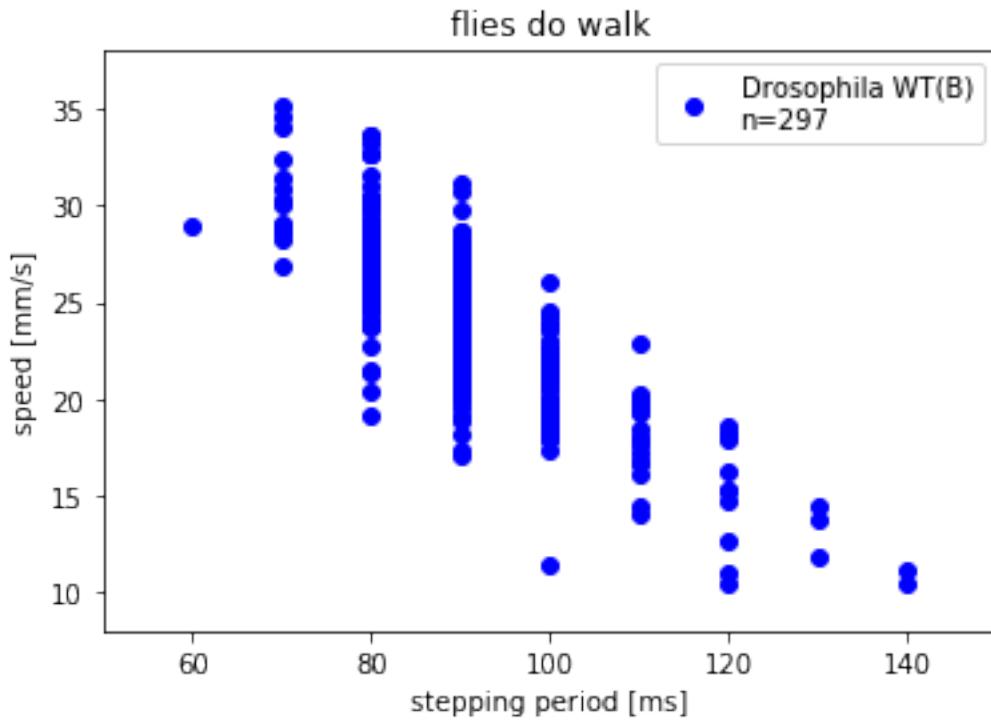
In [13]: *'''read data: speed vs stepping period of walking fly Drosophila
two columns space-separated: period [ms], speed [mm/s]'''*

```
data = np.genfromtxt('data/Drosophila_geschw-periode.dat').T # transpose
print('data is of size {}, a {} of {} of {}'.format(data.shape, type(data), type(data[0]), type(data[0, 1])))
print('period {}'.format(data[0, :5]))    # 1d-array of periods
print('speed {}'.format(data[1, :5]))     # 1d-array of speed

data is of size (2, 297), a <class 'numpy.ndarray'> of <class 'numpy.ndarray'> of <class 'numpy.float64'>
period [60. 70. 70. 70. 70.] ...
speed [29. 34. 34.6 30.3 26.9] ...
```

In [14]: *'''show the data'''*

```
plt.plot(data[0], data[1], 'bo', label='Drosophila WT(B)\nn={}'.format(len(data[0])))
plt.axis((50, 150, 8, 38))
plt.title('flies do walk')
plt.xlabel('stepping period [ms]')
plt.ylabel('speed [mm/s]')
plt.legend(loc='upper right');
```



1.3.2 Korrelation?

```
In [15]: '''correlation between speed and period?'''
r_xy = np.corrcoef(data)[0, 1]
print('correlation coefficient r_xy = {:.3f}'.format(r_xy))
```

correlation coefficient r_xy = -0.819

1.3.3 Regression?

lineare Abhängigkeit - mehrere x-Werte (Periodendauern) - mehrere Messungen (Geschwindigkeit)

Ausgleichsgerade

1.3.4 Methode der Kleinsten Quadratischen Abweichung

(C. F. Gauss 1809, A. M. Legendre 1805)

$$\operatorname{argmin}_{a,b} \sum_i (y_i - f_{a,b}(x_i))^2$$

des linearen Zusammenhangs

$$y = f_{a,b}(x) = a \cdot x + b$$

führt zu

$$a = \frac{\sum_i (y_i - \bar{y})x_i}{\sum_i (x_i - \bar{x})x_i}$$

$$b = \bar{y} - a \cdot \bar{x}$$

Die Steigung oder lineare Abhängigkeit a wird auch **Regressionskoeffizient** genannt und entspricht

$$a = r_{XY} \cdot \frac{\sigma_Y}{\sigma_X}$$

$$\begin{aligned} a &= \frac{\sum_i (y_i - \bar{y})x_i}{\sum_i (x_i - \bar{x})x_i} \\ &= r_{XY} \frac{\sigma_Y}{\sigma_X} \\ \text{Beweise: [ÜA]} \quad &= \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} \cdot \frac{\sigma_Y}{\sigma_X} \\ &= \end{aligned}$$

Im Allgemeinen

$$r_{XY} = r_{YX}$$

$$a_{XY} \neq a_{YX}$$

$$a_{XY} \neq \frac{1}{a_{YX}}$$

Später mehr...

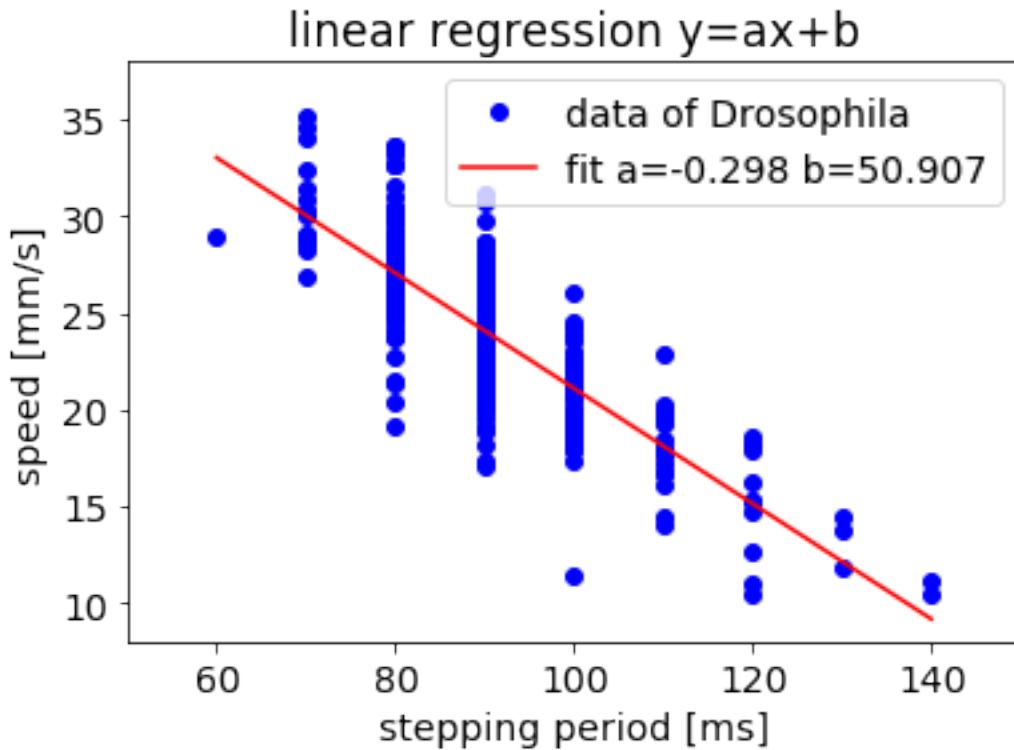
```
In [7]: '''linear function y=f(x)=a*x+b
        input: x: scalar or np.array (x values)
                a: scalar (regression parameter)
                b: scalar (offset parameter)
        output: same type and size as x           '''
def fct_y( x, a, b ):
    return a*x + b

''' make handy names for data: '''
x = data[0]      # the period durations
y = data[1]      # the measured walking speed

'''calculate linear regression according to formula above'''
a = ((y - y.mean())*x).sum() / ((x - x.mean())*x).sum()
b = y.mean() - a*x.mean()
print('manually fitted: y = {:.3f}*x + {:.3f}'.format(a, b))

manually fitted: y = -0.298*x + 50.907
```

```
In [8]: '''show the fit result'''
plt.plot(x, y, 'bo', label='data of Drosophila')          # raw data as dots
plt.plot(x, fct_y(x, a, b), 'r-', label='fit a={:.3f} b={:.3f}'.format(a, b))
plt.title('linear regression y=ax+b')                      # fitted line
plt.axis((50, 150, 8, 38))
plt.xlabel('stepping period [ms]')
plt.ylabel('speed [mm/s]')
plt.legend(loc='upper right');
```



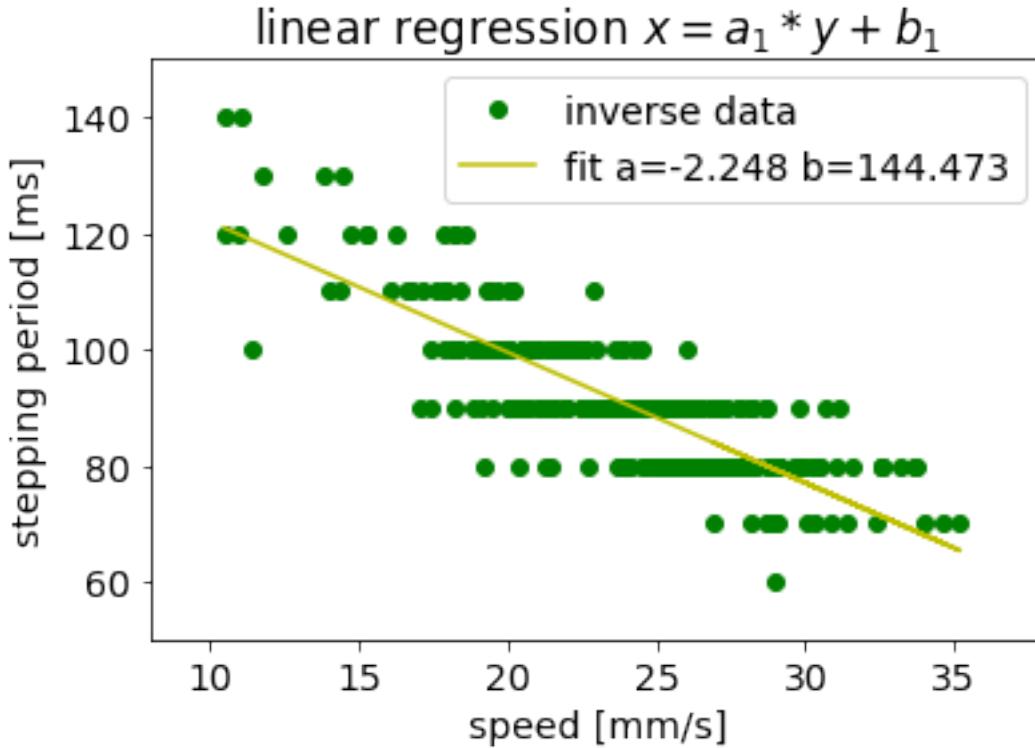
1.3.5 fit with scipy.stats

```
In [9]: '''fit it with scipy directly'''
from scipy import stats          # contains also fitting routines,
fit = stats.linregress(x, y)      #   e.g. linear regression fit
print('fitted by scipy: y = {:.3f}*x + {:.3f}'.format(fit.slope, fit.intercept))

fitted by scipy: y = -0.298*x + 50.907
```

```
In [11]: '''fit the reverse values x(y)'''
fit_r = stats.linregress(y, x)          # linear regression of reverse data
print('linear fit: x = {:.3f}*y + {:.3f}'.format(fit_r.slope, fit_r.intercept))
plt.plot(y, x, 'go', label='inverse data')
plt.plot(y, fct_y(y, fit_r.slope, fit_r.intercept), 'y-',
         label='fit a={:.3f} b={:.3f}'.format(fit_r.slope, fit_r.intercept))
plt.title('linear regression $x=a_1*y+b_1$') # yes, matplotlib speaks LaTeX
plt.axis((8, 38, 50, 150))
plt.ylabel('stepping period [ms]')
plt.xlabel('speed [mm/s]')
plt.legend(loc='upper right');

linear fit: x = -2.248*y + 144.473
```



```
In [12]: ''' Correlation coefficient Pearson r, see 022 lecture'''
def ccpearson( a, b ):
    return np.corrcoef(a, b)[0][1]      # side element of correlation matrix

'''compare fits of data versus reverse-data'''
print('slope {:.3f} * inverse slope {:.3f} = {:.3f} does not equal 1.000'
      .format(fit.slope, fit_r.slope, fit.slope*fit_r.slope))
print('but correlation coefficients match: {:.5f} = {:.5f}'
      .format(ccpearson(x, y), ccpearson(y, x)))

slope -0.298 * inverse slope -2.248 = 0.670 does not equal 1.000
but correlation coefficients match: -0.81869 = -0.81869
```

1.4 Nicht-lineares Beispiel: Wurfparabel

```
In [13]: np.random.seed(98765)
deltaT = 0.1                      # time steps in seconds
period = 6                          # duration of flight in sec
g = 9.81                            # gravity constant
ae = 0.15                           # speed dependent noise amplitude
be = 0.05                           # constant noise amplitude
v0 = 15                             # speed at start in m/s
numOfPoints = int(period/deltaT)     # measured points
t_a = np.zeros(numOfPoints)          # time value[]
```

```

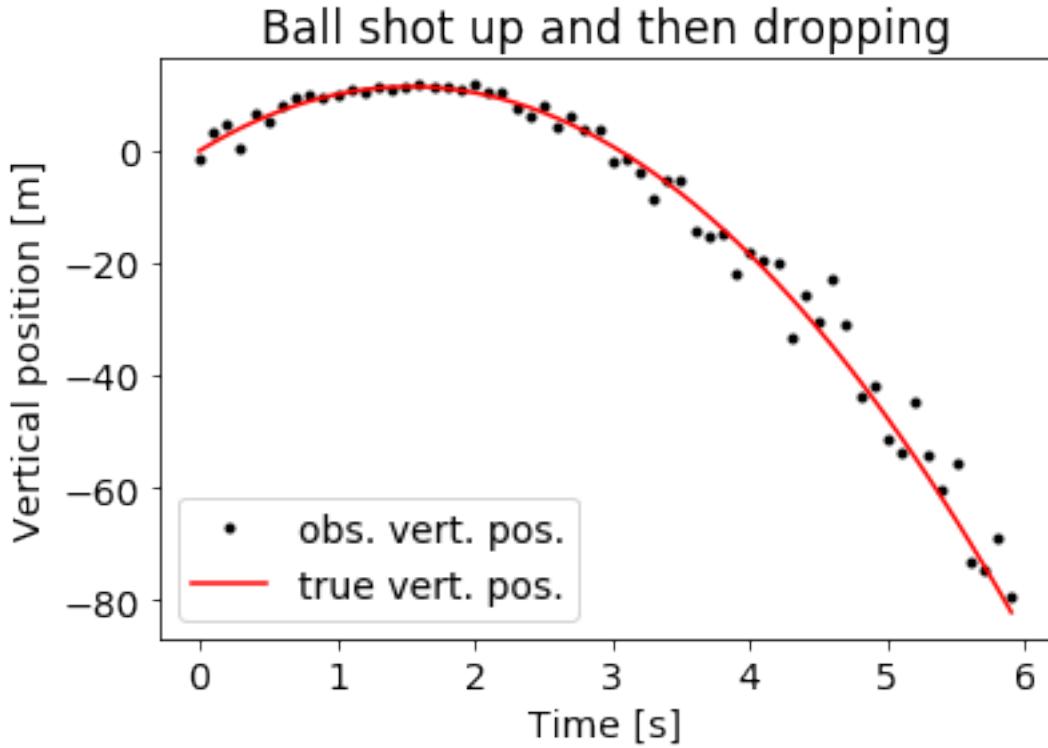
v_a = np.zeros(numOfPoints)          # vertical speed[]
yt_a = np.zeros(numOfPoints)         # y theoretically[]
y_a = np.zeros(numOfPoints)          # y result: with noise[]
for it in range(numOfPoints):
    t = it*deltaT                  # time now
    t_a[it] = t                     # store
    v = v0-g*t                      # speed reduced by gravity
    v_a[it] = v                     # store
    yt = -g*t**2/2+v0*t            # squared by gravitational acceleration
    yt_a[it] = yt                  # store
    error = np.random.normal(loc=0, scale=ae*abs(v)+be)
    y_a[it] = yt+error             # add noise

```

```

In [14]: plt.title('Ball shot up and then dropping')
plt.ylabel('Vertical position [m]')
plt.xlabel('Time [s]')
plt.plot(t_a, y_a, 'k.', label='obs. vert. pos.') # observed points with noise
plt.plot(t_a, yt_a, 'r-', label='true vert. pos.') # theoretical position
plt.legend(loc="lower left");

```



```
In [15]: '''How to fit a polynomial dependency (example: squared)?'''
```

```
np.polyfit?
```

```
In [16]: '''use polyfit (square----v) to extract parabolic parameters'''
```

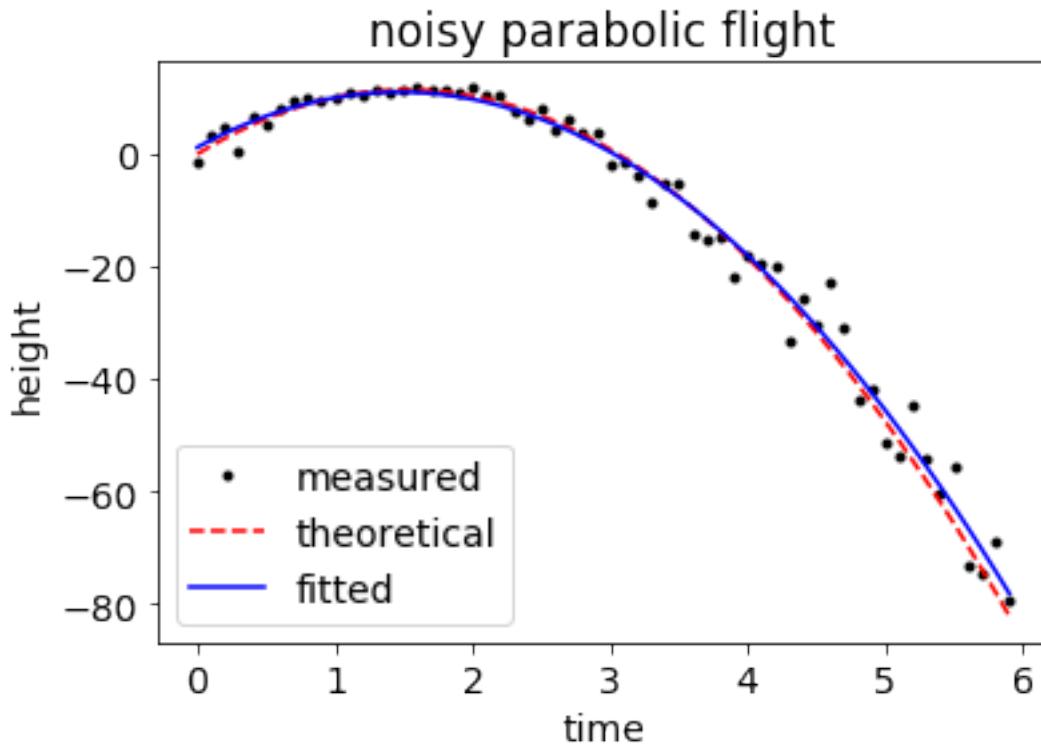
```
pp = np.polyfit(t_a, y_a, 2)          # polynomparameters from fit of degree 2
print('fitted polynom has parameters pp={}'.format(pp))
```

```
fitted polynom has parameters pp=[ -4.54965459  13.37918126   1.2131438 ]
```

```
In [17]: '''compare measured and fitted values (with randomly drawn distribution)'''

plt.plot(t_a, y_a, 'k.', label='measured')
plt.plot(t_a, yt_a, 'r--', label='theoretical')
py = np.poly1d(pp)           # define polynomial function with fitted parameters
plt.plot(t_a, py(t_a), 'b-', label='fitted')# show fitted locations versus time
plt.legend(loc='lower left')
plt.title("noisy parabolic flight")
plt.xlabel('time')
plt.ylabel('height')
print('gravitational constant = {:6.3f} m/s^2'.format(-2*pp[0])) # from  $y \sim -g*t^2/2$ 
print('starting speed was v0 = {:6.3f} m/s'.format(pp[1]))      # derivative at t=0

gravitational constant = 9.099 m/s^2
starting speed was v0 = 13.379 m/s
```



Siehe auch

- `numpy.polyfit`
- `scipy.optimize.curve_fit`

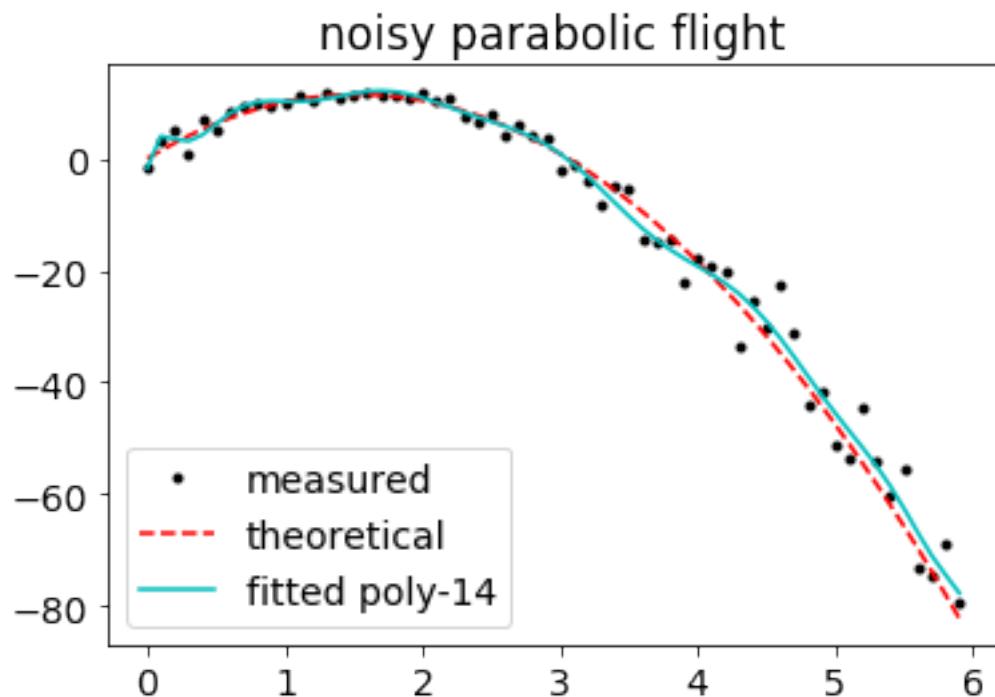
```
In [18]: p14 = np.polyfit(t_a, y_a, 14)      # polynomparameters from fit of degree 14 (?)
print('fitted polynom of degree {} has parameters p14=\n{}'.format(len(p14)-1, p14))
plt.plot(t_a, y_a, 'k.', label='measured')
```

```

plt.plot(t_a, yt_a, 'r--', label='theoretical')
py = np.poly1d(p14)                      # define polynomial function with parameters
plt.plot(t_a, py(t_a), 'c-', label='fitted poly-14')
plt.legend(loc='lower left')
plt.title("noisy parabolic flight");

fitted polynom of degree 14 has parameters p14=
[ -1.71045973e-03   7.25881832e-02  -1.37704393e+00   1.54074336e+01
 -1.12969223e+02   5.70392805e+02  -2.02844834e+03   5.10788156e+03
 -9.02800340e+03   1.09398432e+04  -8.70967673e+03   4.23721214e+03
 -1.11180856e+03   1.33379787e+02  -1.67218166e+00]

```



```

In [19]: '''compare polynomial coefficients and residual error'''
pp14, residual14, rk, sv, rc = np.polyfit(t_a, y_a, 14, full=True)
pp2, residual2, rk, sv, rc = np.polyfit(t_a, y_a, 2, full=True)
print('polynomial models with {} / {} parameters decreases residual error from {:.1f} to {:.1f}'.
      format(len(pp2), len(pp14), residual2[0], residual14[0]))
for ppx in [pp2, pp14]:
    l = len(ppx)
    print('last 3 parameters of {:2d}-parameter model: {:.8.2f}, {:.8.2f}, {:.8.2f}'.
          format(l, ppx[l-3], ppx[l-2], ppx[l-1]))

polynomial models with 3 / 15 parameters decreases residual error from 668.9 to 581.9
last 3 parameters of 3-parameter model:    -4.55,     13.38,     1.21
last 3 parameters of 15-parameter model: -1111.81,   133.38,   -1.67

```

Ergebnis:

- obwohl der Restfehler weniger wird
 - für das 14-parametrische Modell
 - gegenüber dem (korrekten) parabolischen
- ist das Ergebnis zweifelhaft
 - Ein Ball steigt nicht wieder
 - der quadratische Koeffizient ist unsinnig hoch

2 Fragen?**3 Zusammenfassung beschreibende Statistik**

- Datenreduktion
- Kenngrößen herausarbeiten
 - Mittelwert
 - Standardabweichung
 - Korrelationskoeffizient
- Struktur in den Daten erkennen
 - Form (Normalverteilung, Gleichverteilung, ...)
- Anschauliche Darstellung
 - Box-Plot
 - Fehlerbalken
 - Histogramm
 - 2D-Histogramm
- Vergleichbarkeit unter ähnlichen Bedingungen
- Abhängigkeit mehrerer Variablen
 - Korrelation
 - Regression

4 Fragen an die Statistik

- Was ist die *wahre Größe* bei mehreren Messungen/Daten?
- Wie genau ist das erlangte Wissen?
- Was sind Ausreißer
- Effekt zwischen unterschiedlichen Versuchsbedingungen?
 - andere Wert?
 - andere Verteilung?
 - Zufall?
- Abhängigkeit wenn nichtlinear?

5 Ausblick

- Wahrscheinlichkeitstheoretische Modelle
 - Entspricht das Histogramm einer bekannten Verteilung?
 - Wie kommen die Daten zustande?
- Schließende Statistik arbeitet mit Wahrscheinlichkeiten
 - Gibt es grundlegende Parameter der Daten?

6 Links

- Matplotlib Graphikgalerie <http://matplotlib.org/gallery.html>
- Pandas Graphiken: <http://pandas.pydata.org/pandas-docs/stable/visualization.html>

7 Fragen?

031_Folien

2018 年 11 月 23 日

```
In [1]: import numpy as np                      # mathematical methods
         from matplotlib import pyplot as plt      # plotting methods
         %matplotlib inline
```

1 Wahrscheinlichkeitstheorie

Stochastik fasst als Oberbegriff die Gebiete Wahrscheinlichkeitstheorie und Statistik zusammen.

Wikipedia: Stochastik (von altgriechisch στοχαστική τέχνη stochastik techn; lateinisch ars conjectandi):

1.1 *Kunst des Vermutens*

1.2 Diskrete Wahrscheinlichkeitstheorie - Überblick

1. Zufall

- Zufallsvariable
- Zufallsexperiment
- WahrscheinlichkeitsmaSS
- Wahrscheinlichkeitsverteilung
- Axiome von Kolmogorov
- Kombinatorik

2. Wahrscheinlichkeiten

- bedingte Wahrscheinlichkeit
- gemeinsame Wahrscheinlichkeit
- totale Wahrscheinlichkeit
- Satz von Bayes

3. KenngröSSen

- Erwartungswert
- Varianz

4. Modell-Verteilungen

- Laplace-Experiment
- Bernoulli-Experiment
- Binomialverteilung
- Geometrische Verteilung

- Poissonverteilung
- Wahrscheinlichkeitsverteilung
- Verteilungsfunktion
- Summe von Verteilungen

5. Zusammenfassung

- Ausblick kontinuierliche Wahrscheinlichkeitstheorie
- Fragen

1.3 Fragestellung

- theoretischer Unterbau zum Zufall
- Modell für relative Häufigkeiten aus beschreibender Statistik
- Interpretation des Modells als Brücke zur Wirklichkeit

1.4 Wahrscheinlichkeit

Physik: Quantenmechanik, Thermodynamik, Geophysik (Erdbeben), Wetter

Biologie: Genetik (Mendel), Ökologie (Populationen) <-> Individuen

Sozialwissenschaften: Populationen

Informatik: Verschlüsselung, Zufallszahlen

2 Grundbegriffe

3 Zufallsvariable

- Merkmal X in der Welt
- möglicher Ausgang eines Zufallsexperiments
 - steht nicht fest
 - ist nicht berechenbar
 - sondern **zufällig**
- Wahrscheinlichkeit
 - inhärent
 - gleichbleibend

Beispiele:

- Werfen einer Münze
- Würfeln
- Roulette
- Elektronik: Spannung und Rauschspannung
- Biologie/Genetik: Erbsen

3.1 erweiterte Zufallsvariable

Erweiterbar auf prinzipiell messbare Werte, wenn jedoch die Rahmenbedingungen zu komplex sind.

Beispiele:

- Roulette
- Anteil der Mädchen an heutigen Geburten
- Sonntagsfrage

3.2 Realisierung

Das konkrete Ergebnis eines Zufallsexperiments nennt man Realisierung einer Zufallsvariable.

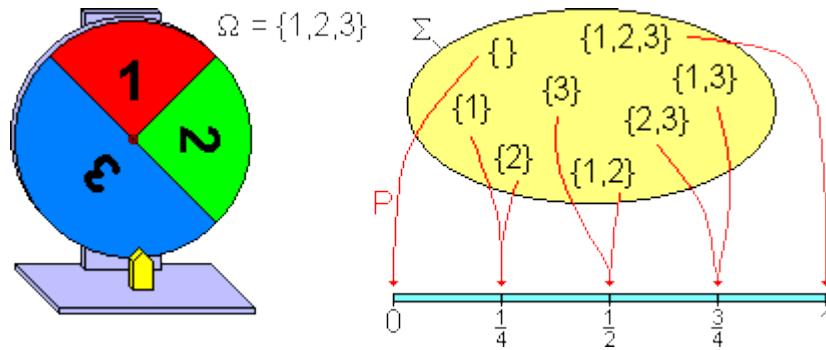
- Messung
- Stichprobenergebnis
- Ziehung
- Erhebung

4 Modellvorstellung "Zufallsexperiment"

(Quelle: Wikipedia. Authorin: Frau Holle Joxemai4, CC BY-SA 3.0)

In [1] :

Out [1] :



4.1 Ockhams Rasiermesser

occam's razor, Sparsamkeitsprinzip, lex parsimoniae

Von mehreren möglichen Erklärungen für ein und denselben Sachverhalt ist die einfachste Theorie allen anderen vorzuziehen.

Eine Theorie ist einfach, wenn sie möglichst wenige Variablen und Hypothesen enthält, und wenn diese in klaren logischen Beziehungen zueinander stehen, aus denen der zu erklärende Sachverhalt logisch folgt.

- benannt nach Wilhelm von Ockham (1288–1347)
- stammt von dem Philosophen Johannes Clauberg (1622–1665)

(Wikipedia)

4.2 Wahrscheinlichkeitsraum (Ω, Σ, P)

- **Ergebnisraum** $\Omega = \{\omega_i\}$:
 - Menge aller elementaren Ergebnisse
- **Ereignisraum** Σ
 - bestehend aus allen möglichen Kombinationen von Ergebnissen aus Ω
- **WahrscheinlichkeitsmaSS** P
 - ordnet jedem Ereignis aus Σ eine Wahrscheinlichkeit P zu

5 Zufallsexperiment

Eines der **Ergebnisse** (Elementarereignisse) kommt heraus, wird gemessen, wird realisiert.

- Ergebnis ω
- Ergebnismenge $\Omega = \{\omega_1, \omega_2, \dots\}$: Menge aller Elementarereignisse

Ereignis A hat stattgefunden, Z.B.: "keine eins"

- Menge aller Ereignisse: alle möglichen Ergebnis-Kombinationen
- σ -Algebra Σ auf Ω
- $A \in \Sigma$
- z.B. Ereignis $A = \{\omega_2 \cup \omega_3\}$

Beispiel: Glücksrad

- $\Omega = \{1, 2, 3\}$
- $\Sigma = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$
- Ereignis A: "zwei" $A = \{2\}$
- Ereignis B: "nicht eins" $B = \{2, 3\}$

6 WahrscheinlichkeitsmaSS P

Das WahrscheinlichkeitsmaSS P ordnet jedem Ereignis aus Σ eine Wahrscheinlichkeit zu - $P: \Sigma \rightarrow [0, 1]$

Beispiel Dem Ereignis "nicht eins" $A = (\omega_2 \cup \omega_3)$ wird die Wahrscheinlichkeit $P(A) = P(\omega_2) + P(\omega_3) = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$ zugeordnet.

Wahrscheinlichkeitsraum nominal, kategorial, diskret Ω hat endlich viele Elemente A_i

Ist $A = \{\omega_1 \cup \omega_2\}$, dann ist $P(A) = P(\omega_1) + P(\omega_2)$

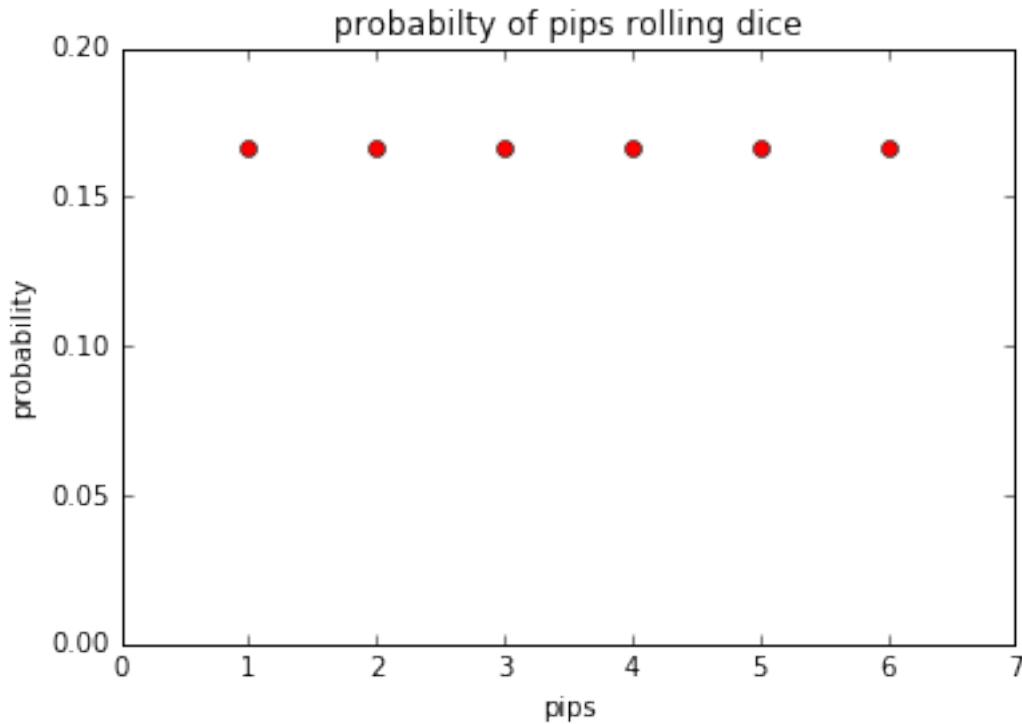
Wahrscheinlichkeitsraum kontinuierlich, stetig Es gibt unendlich viele Elemente, meist $x \in \mathbb{R}$

Die Wahrscheinlichkeit für A (einen Bereich) ist $P(A) = \int_{x \in A} f(x) dx$ mit der Wahrscheinlichkeitsdichte f (später mehr)

6.1 Wahrscheinlichkeitsverteilung

In [2]: *'----dice---*

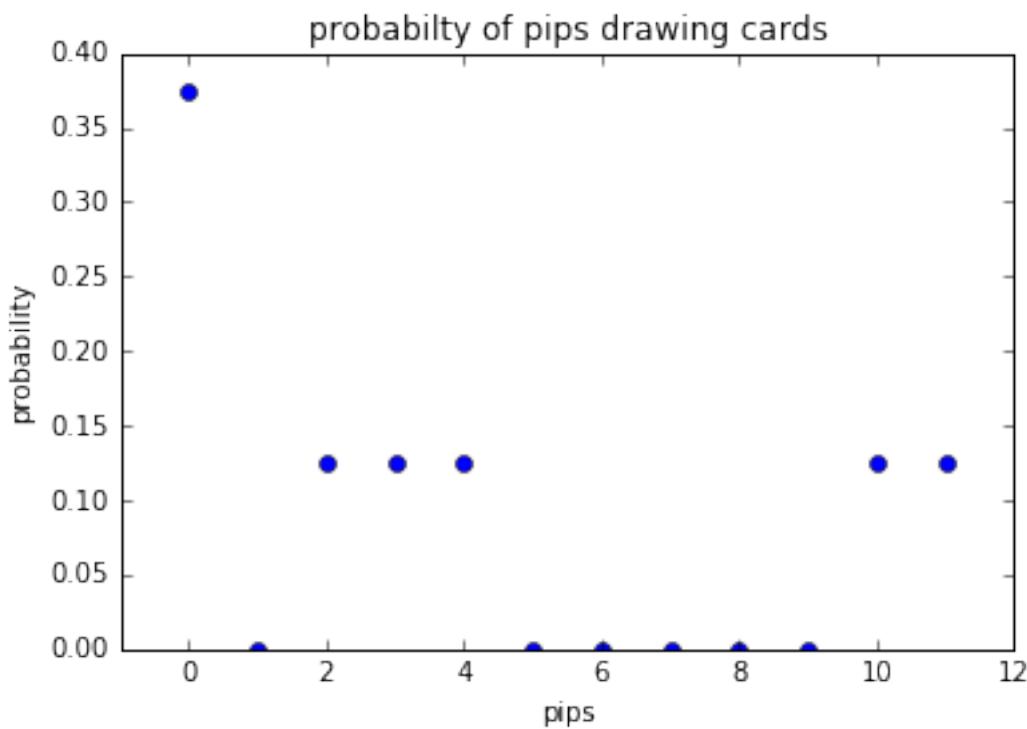
```
x = np.array([1, 2, 3, 4, 5, 6])
p = np.ones_like(x)/6.
plt.plot(x, p, 'ro')
plt.title('probabilty of pips rolling dice')
plt.xlim(0, 7)
plt.ylim(0, 0.2)
plt.xlabel('pips')
plt.ylabel('probability');
```



In [3]: *'----cards---*

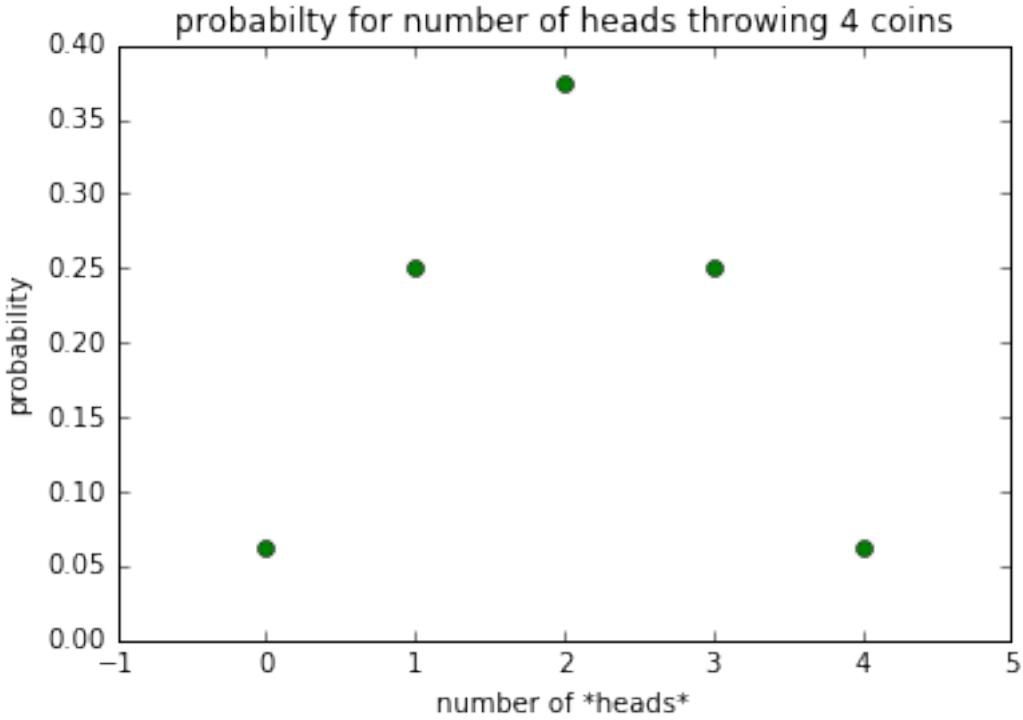
```
x = np.arange(12)           # {0..11} range of possible pips
p = np.zeros_like(x)
p[0] = 3                   # '7', '8', '9'
p[2] = 1                   # J
p[3] = 1                   # Q
p[4] = 1                   # K
p[10] = 1                  # '10'
p[11] = 1                  # Ace
p = p*4.0/32.0             # 4 colours / 32 cards
plt.plot(x, p, 'bo')
plt.xlim(-1, 12)
plt.ylim(0, 0.4)
plt.title('probabilty of pips drawing cards')
```

```
plt.xlabel('pips')
plt.ylabel('probability');
```



In [4]: '''---4 coins---'''

```
x = np.arange(5) # {0..4}
# {0000}{0001,0010,0100,1000}{0011,0101,0110,1001,1010,1100}...
p = np.array([1, 4, 6, 4, 1])
p = p/16.0
plt.plot(x, p, 'go')
plt.title('probabiltiy for number of heads throwing 4 coins')
plt.xlim(-1, 5)
plt.ylim(0, 0.4)
plt.xlabel('number of *heads*')
plt.ylabel('probability');
```



7 Kolmogorov-Axiome

Andrei Kolmogorov (1930er Jahre); In Übereinstimmung mit alltäglicher Erfahrung und Umgangssprache:
Gesamtheit

$$P(\Omega) = 1$$

Unmögliches Ereignis

$$P(\emptyset) = 0$$

Wahrscheinlichkeiten

$$0 \leq P(A) \leq 1$$

Eintreten oder Nichteintreten eines Ereignisses

$$P(A) + P(\bar{A}) = 1$$

Addition der Wahrscheinlichkeiten bei disjunkten Ereignissen

$$P(A \cup B) = P(A) + P(B) \quad \text{falls } A \cap B = \emptyset$$

-Additivität

$$\begin{aligned} \sum_{i=1}^n P(\omega_i) &= 1 \\ \omega_i \cap \omega_j &= \emptyset \quad \forall i \neq j \\ P(\omega_1 \cup \omega_2 \cup \dots) &= \sum P(\omega_i) \end{aligned}$$

8 Kombinatorik

Woher kommen die Wahrscheinlichkeiten?

Was ist der Ereignisraum Σ

8.0.1 Venn-Diagramm 韦恩图

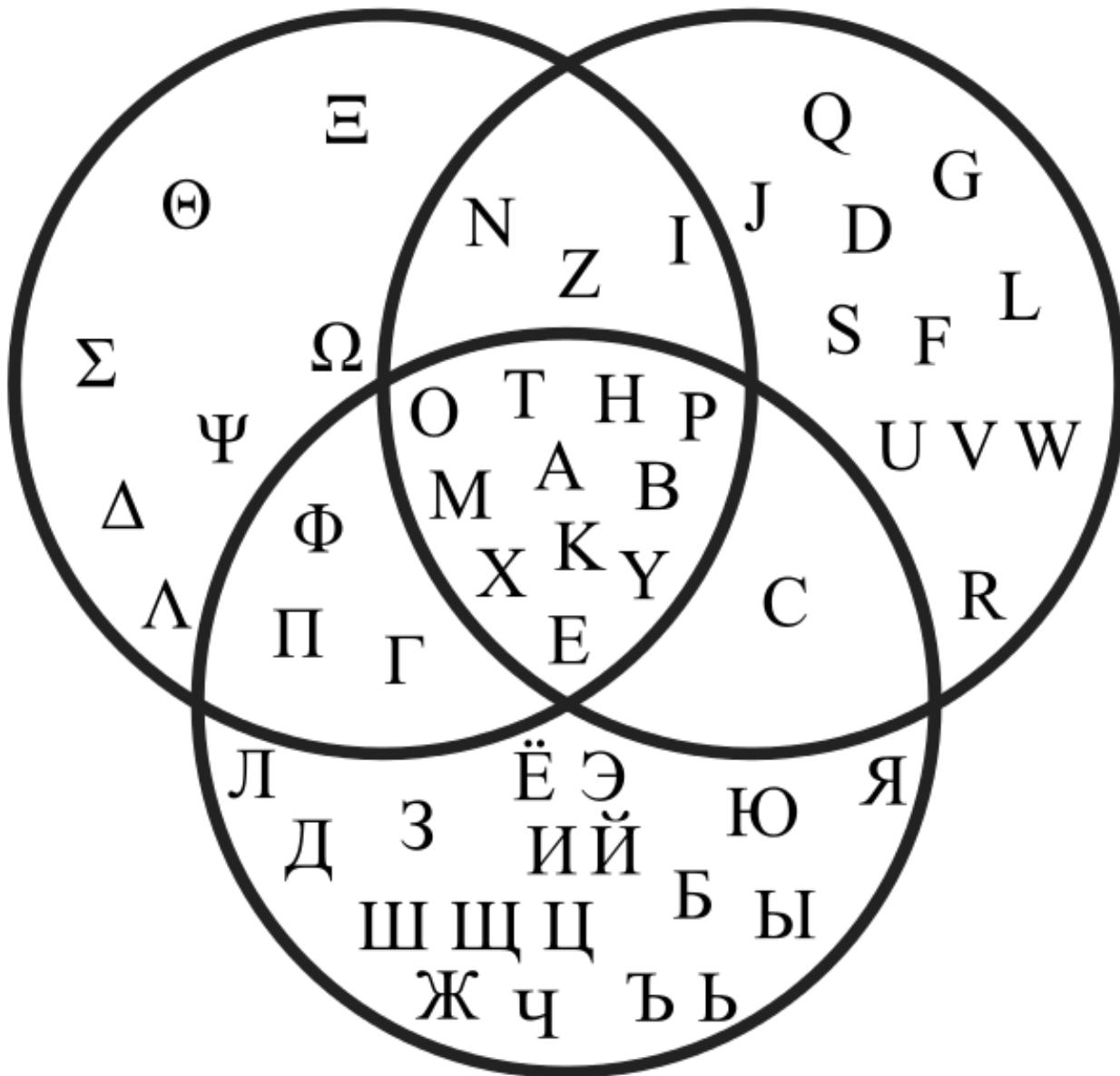
In der Mengenalgebra lassen sich alle Ereignisse im Venn-Diagramm darstellen.

Beispiel: Verteilung von Buchstaben in lateinischer, griechischer oder kyrillischer Schrift.

(Quelle: Wikipedia. Author: Tilman Piesk, Public domain)

In [2] :

Out [2] :



8.0.2 Mögliche Wertebereiche für X

- nominal:

- keine Reihenfolge
- zB. Farbe der Kugel "rot", "schwarz", "weiß"
- dichotom
 - "ja"/"nein" oder 0/1 für Bernoulli Experiment
 - zB. Münzwurf
- ordinal
 - Reihenfolge
 - kein Abstand
 - zB. bei einer "Stimme zu"-Umfrage: "nicht", "etwas", "ziemlich", "sehr"
- metrisch - diskret
 - $x_i, i \in \{1 \dots N\}, N \in \mathbb{N}$
 - Abstand und Nullpunkt
 - zB. Augenzahl beim Würfeln, mm-Einteilung Meterstab
- metrisch - stetig
 - $x \in \mathbb{R}$
 - kontinuierlich
 - zB. Helligkeit im Raum, Abstand

9 Spieletheorie

Ab 18./19. Jhd einzeln Gesichtspunkte, z.B. Bernoulli

1928 John v. Neumann: Grundbegriffe der Spieltheorie

Seit 1994 8x Alfred-Nobel-Gedächtnispreis für Wirtschaftswissenschaften

10 Laplace Experiment

Pierre Simon Marquis de Laplace (1749-1827)

- alle Elementarereignisse sind *gleich* wahrscheinlich

$$P(\omega_i) = p = \frac{1}{M} = \frac{1}{|\Omega|} \quad i \in \{1 \dots M\}$$

- Wahrscheinlichkeit = $\frac{\text{Anzahl für A günstige Ergebnisse}}{\text{Anzahl für A mögliche Ergebnisse}}$

$$P(A) = \frac{|A|}{|\Omega|} = \frac{N}{M}$$

```
In [5]: '''example dice: even number?'''
O = [1, 2, 3, 4, 5, 6]
A = [2, 4, 6]
print('P(A={}) under Omega={} is {}'.format(A, O, len(A) / len(O)))

P(A=[2, 4, 6]) under Omega=[1, 2, 3, 4, 5, 6] is 0.5
```

11 Zusammenfassung

- Zufallsvariable
- Wahrscheinlichkeitsraum (Ω, Σ, P)
- Zufallsexperiment
- Kolmogorov-Axiome

11.1 Ausblick

- Verbundene Ereignisse
- Wert des Zufallsexperiments
- Wetteinsatz und Wettgewinn

12 Fragen?

032_Folien

2018 年 11 月 23 日

```
In [1]: import numpy as np                      # mathematical methods
         from matplotlib import pyplot as plt      # plotting methods
         %matplotlib inline
```

0.1 Diskrete Wahrscheinlichkeitstheorie - Überblick

1. Zufall

- Zufallsvariable, Zufallsexperiment, WahrscheinlichkeitsmaSS, Wahrscheinlichkeitsverteilung, Axiome von Kolmogorov
- Laplace-Experiment

2. Zusammengesetzte Wahrscheinlichkeiten

- Verbundwahrscheinlichkeit
- bedingte Wahrscheinlichkeit
- absolute Wahrscheinlichkeit
- Satz von Bayes

3. KenngröSSen

- Erwartungswert & Varianz

4. Modellverteilungen

1 Zufallsexperiment

1.1 Zufallsvariable

1.2 Wahrscheinlichkeitsraum (Ω, Σ, P)

1.3 Kolmogorov-Axiome

2 Kombinatorik

Woher kommen die Wahrscheinlichkeiten?

Was ist der Ereignisraum Σ

3 Laplace Experiment

Pierre Simon Marquis de Laplace (1749-1827)

- alle Elementarereignisse sind *gleich* wahrscheinlich

$$P(\omega_i) = p = \frac{1}{M} = \frac{1}{|\Omega|} \quad i \in \{1 \dots M\}$$

- Wahrscheinlichkeit = $\frac{\text{Anzahl für A günstige Ergebnisse}}{\text{Anzahl für A mögliche Ergebnisse}}$

$$P(A) = \frac{|A|}{|\Omega|} = \frac{N}{M}$$

4 Mehrdimensionale Zufallsexperimente

5 Verbundwahrscheinlichkeit

6 gemeinsame Wahrscheinlichkeit

joint probability Zwei Zufallsvariablen bilden einen Verbund.

6.0.1 Beispiel zwei unterscheidbare Würfel

Zwei faire Würfel (ein roter, ein grüner) mit Wahrscheinlichkeiten $p_{g,i} = \frac{1}{6}$ und $p_{r,i} = \frac{1}{6}$

Ergebnissraum Ω besteht aus $6 \times 6 = 36$ Tupeln

Entsprechende Wahrscheinlichkeiten in Kontingenztabelle $6 \times 6 = 36$ möglichen Elementarereignisse $p_i = P(\omega_i) = \frac{1}{36}$

grün \ rot	1	2	3	4	5	6
1	c	a				
2	c	a				
3	a	a	a	a	a	a
4			a			
5			a			
6	a			s		

- Ereignis S: Sechserpasch
 - 1 Ereignis
 - Ergebnis (6,6)
- Ereignis A: mindestens eine Drei
 - 6+5 Möglichkeiten (3,*) und (*,3) (ohne nochmals (3,3))
 - $p = 11/36$
- Ereignis C: Mäxchen
 - 2 Möglichkeiten
 - $p=1/18$

7 Bedingte Wahrscheinlichkeit

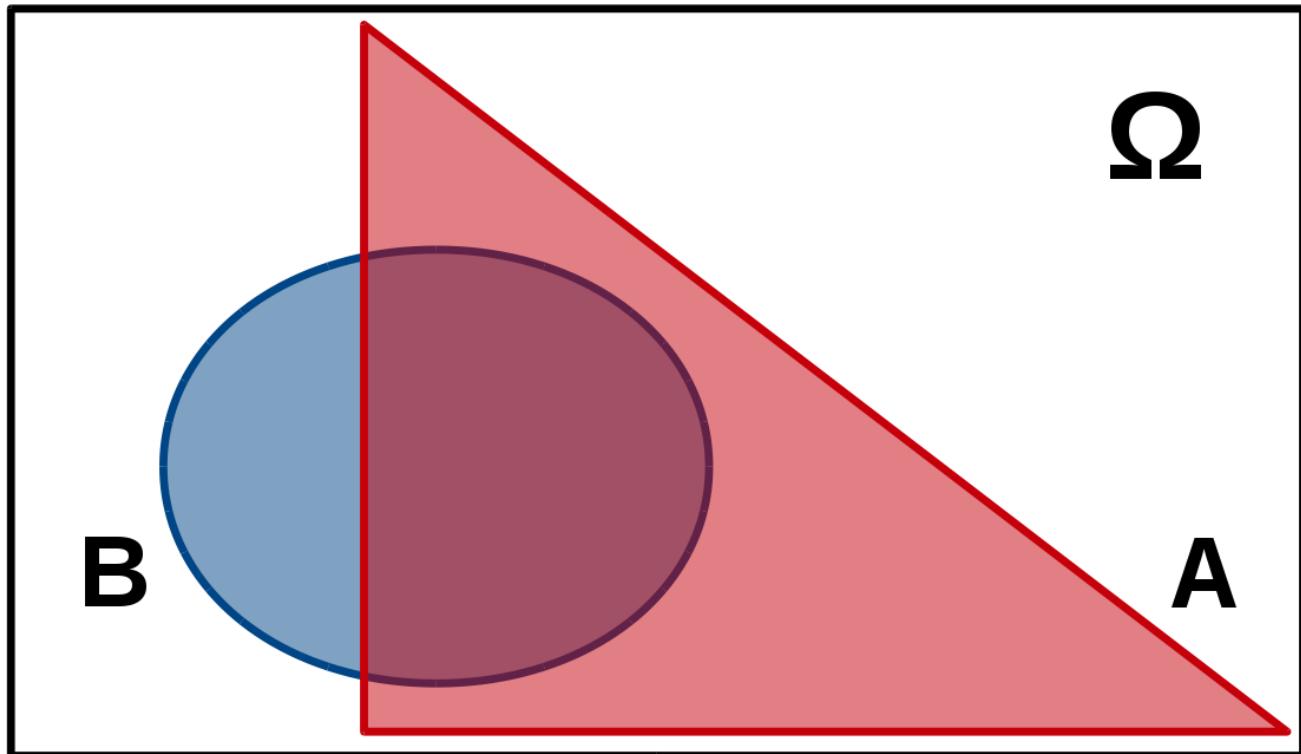
Unter der Voraussetzung, daß Ereignis B bereits eingetreten ist, wie wahrscheinlich ist dann A?

Beispiele

- Wahrscheinlichkeit für "sechs", wenn festgestellt wurde, daß Augenzahl "gerade"
- Wahrscheinlichkeit von Herzinfarkt, wenn über 60

In [6]:

Out [6]:



7.1 Definition bedingte Wahrscheinlichkeit

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \quad \text{wenn } p(B) > 0$$

7.1.1 Laplace-Experiment

$$p(A|B) = \frac{|A \cap B|}{|B|}$$

z.B. "sechs" wenn "gerade"

- Variante "günstige" / "mögliche": $p(\{6\}|\{2,4,6\}) = \frac{|\{6\} \cap \{2,4,6\}|}{|\{2,4,6\}|} = \frac{|\{6\}|}{|\{2,4,6\}|} = \frac{1}{3}$
- Variante bedingte Wahrscheinlichkeit: $p(\{6\}|\{2,4,6\}) = \frac{p(\{6\} \cap \{2,4,6\})}{p(\{2,4,6\})} = \frac{p(\{6\})}{p(\{2,4,6\})} = \frac{1/6}{1/2} = \frac{1}{3}$

z.B. "gerade" wenn "sechs" - Variante bedingte Wahrscheinlichkeit: $p(\{2,4,6\}|\{6\}) = \frac{p(\{2,4,6\} \cap \{6\})}{p(\{6\})} = \frac{p(\{6\})}{p(\{6\})} = \frac{1/6}{1/6} = 1$

7.2 Produktregel der absoluten Wahrscheinlichkeit

$$p(A \cap B) = p(A|B) \cdot p(B)$$

- Formel der Definition für bedingte Wahrscheinlichkeit umgestellt
- Anschaulich klar für A UND B
 - zuerst muß B eintreten
 - dann auch noch A (unter der Voraussetzung B)

Daraus auch

$$p(A \cap B) = p(B|A) \cdot p(A)$$

7.3 Unabhängigkeit zweier Ereignisse

Wenn die Wahrscheinlichkeit für ein Ereignis nicht von einem anderen Ereignis B abhängt, so nennt man die beiden Ereignisse stochastisch unabhängig.

$$P(A) = P(A|B_1) = P(A|B_2) = \dots$$

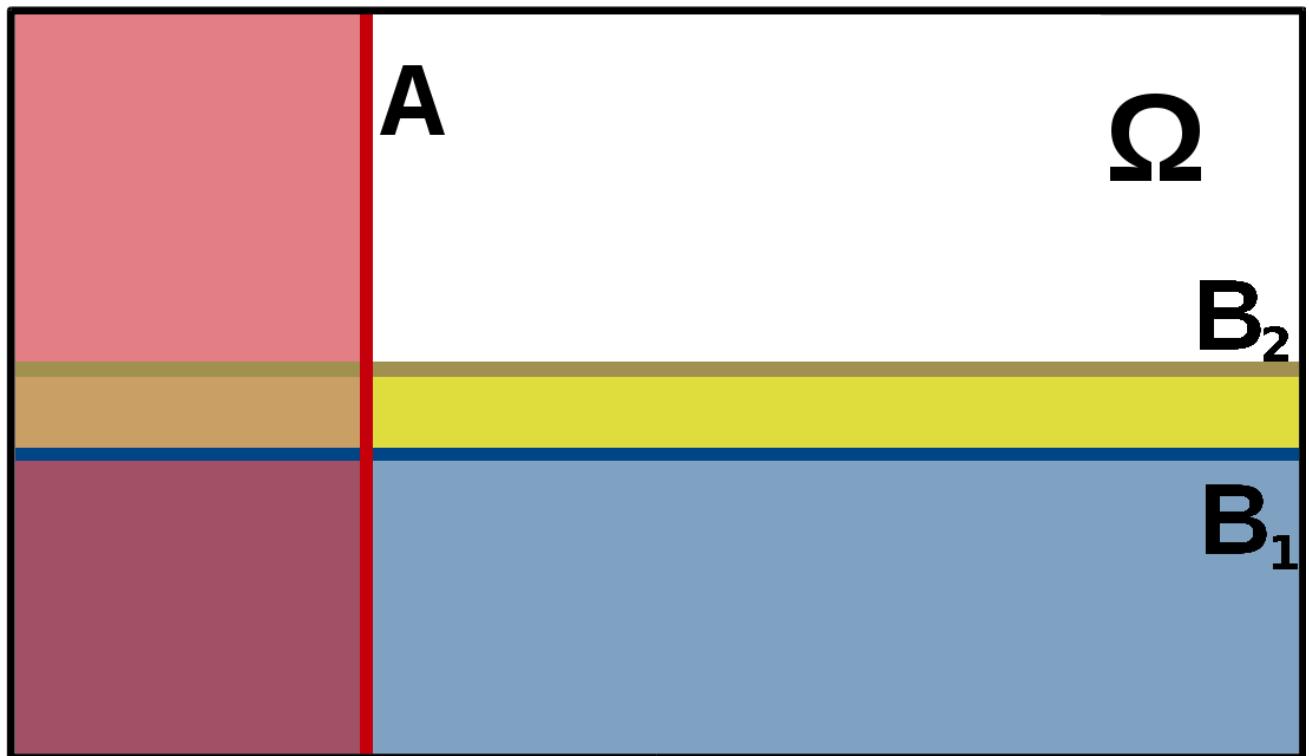
$$P(A \cap B) = P(A) \cdot P(B)$$

$$P(A|B) = P(A) \quad \text{wenn } P(B) > 0$$

$$P(B|A) = P(B) \quad \text{wenn } P(A) > 0$$

In [7] :

Out [7] :



Beispiele stochastischer Unabhängigkeit

- zwei Würfel - Augenzahl-1 und Augenzahl-2
- Farbe und Wert einer Spielkarte
- Farbe und Form von Erbsen

Gegenbeispiele

- Einkommen und Geschlecht
- [ÜA] Würfeln I - Bei Augenzahl > 4 bekommt man 30ct - bei Augenzahl 6 bekommt man 70ct
 - Welche Wahrscheinlichkeit hat das Ereignis "beide Preise"?
- [ÜA] Würfeln II - Bei Augenzahl 5 bekommt man 30ct - bei Augenzahl 6 bekommt man 1
 - Welche Wahrscheinlichkeit hat das Ereignis "beide Preise"?

7.3.1 Produktregel für unabhängige Ereignisse

Wiederholung der allgemeinen Produktregel

$$p(A \cap B) = p(A|B) \cdot p(B)$$

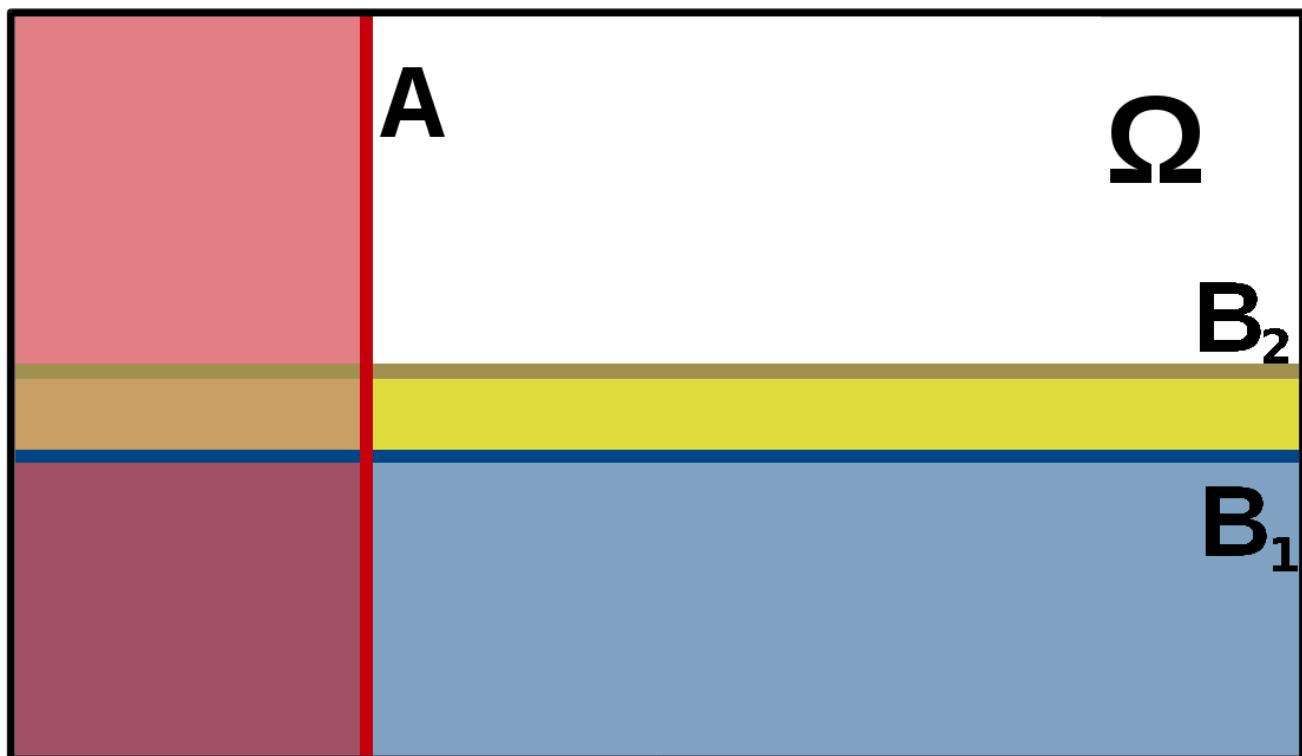
Bei unabhängigen Ereignissen ergibt sich daraus

$$p(A \cap B) = p(A) \cdot p(B)$$

Diese Eigenschaft *definiert* die Unabhängigkeit.

In [7] :

Out [7] :



7.4 Satz von der totalen Wahrscheinlichkeit

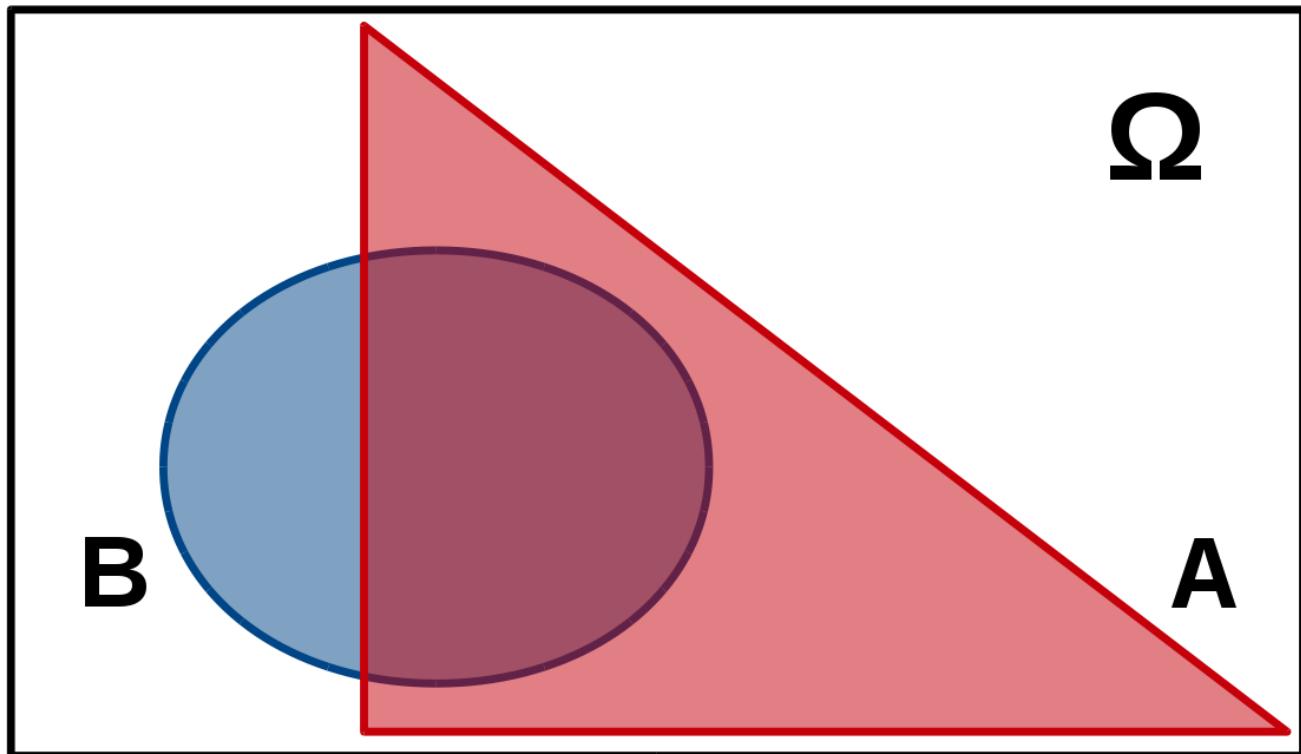
Sei $\{A_k\}$, $k \in \{1, \dots, K\}$ eine disjunkte Zerlegung von Ω . Dann gilt für $B \subset \Omega$

$$P(B) = \sum_{k=1}^K P(B|A_k) \cdot P(A_k)$$

Venn-Diagramm

In [8]:

Out[8]:



8 Beispiel: Medizinischer Test

- Ein medizinischer Test zeigt bei 98% der Kranken ein “positives” Ergebnis.
- Die Fehlerrate bei Gesunden beträgt 3%, fälschlicherweise ein “positives” Ergebnis zu melden.
- Ereignis A: “Patient ist krank”
- Ereignis B: “Testergebnis positiv”
- Die Krankheit ist selten und nur mit 0,1% in der Bevölkerung vertreten.

Wie groß ist die Wahrscheinlichkeit bei positivem Test krank zu sein?

- Gesucht ist $P(A|B) = P(\text{krank}|\text{Test+})$

$$P(B|A) = 0,98$$

Berechnung $P(B|\bar{A}) = 0,03$

$$P(A) = 0,001$$

[Disjunkte Zerlegung] A und \bar{A} mit $P(\bar{A}) = 0,999$

[Definition bedingte W.] $P(A|B) = \frac{P(A \cap B)}{P(B)}$

[Produktregel] $P(A \cap B) = P(B|A) \cdot P(A)$

[Satz der totalen W.] $P(B) = \sum_{i=1}^2 P(B|A_i) \cdot P(A_i) = P(B|A) \cdot P(A) + P(B|\bar{A}) \cdot P(\bar{A})$

$$\begin{aligned} P(A|B) &= \frac{P(B|A) \cdot P(A)}{P(B|A) \cdot P(A) + P(B|\bar{A}) \cdot P(\bar{A})} \\ &= \frac{0,98 \cdot 0,001}{0,98 \cdot 0,001 + 0,03 \cdot 0,999} \\ &= \frac{0,00098}{0,00098 + 0,02997} \\ &= 0,032 \end{aligned}$$

9 der Satz von Bayes

Sei A_1, \dots, A_K eine disjunkte Zerlegung von Ω . Dann gilt:

$$P(A_j|B) = \frac{P(B|A_j) \cdot P(A_j)}{\sum_{i=1}^K P(B|A_i) \cdot P(A_i)} = \frac{P(B|A_j) \cdot P(A_j)}{P(B)}$$

Mindestens für ein $i \in \{1, \dots, K\}$ muß $P(A_i) > 0$ und $P(B|A_i) > 0$ gelten.

9.0.1 Historisch

Benannt nach englischem Mathematiker Thomas Bayes

1763 posthum veröffentlicht in: *An Essay Towards Solving a Problem in the Doctrine of Chances*

In [9]: *'''Quelle: Wikipedia, Gemeinfrei. Erstellt: 19th century(?), reprinted 1936, 1981, 1988'''*

Out[9]:



9.1 SchluSSfolgern

Hat es nachts geregnet wenn der Rasen morgens naSS ist?

- Regenwahrscheinlichkeit lag nach Wettervorhersage bei 40%

$$p(\text{Regen}) = 0.40$$

- Totale Wahrscheinlichkeit

$$p(\text{na}) = p(\text{na}|\text{Regen}) \times p(\text{Regen}) + p(\text{na}|\overline{\text{Regen}}) \times p(\overline{\text{Regen}})$$

Satz von Bayes

$$\begin{aligned}
 p(\text{Regen}|\text{na}) &= \frac{p(\text{na}|\text{Regen}) \times p(\text{Regen})}{p(\text{na})} \\
 &= \frac{p(\text{na}|\text{Regen}) \times p(\text{Regen})}{\sum_i p(\text{na}|\text{Regen}_i) \times p(\text{Regen}_i)} \\
 &= \frac{100\% \times 40\%}{100\% \times 40\% + 0\% \times 60\%} = 100\%
 \end{aligned}$$

Hat es nachts geregnet wenn der Rasen morgens naSS ist - II?

- Nachbars Rasensprenger hat Zufalls-Zeitschaltuhr, die zu 20% anspringt
- Nachbarsfrau gießt jeden zweiten Tag und jedes fünfte Mal fällt die Gieskanne um

$$\begin{aligned}
 p(\text{Regen}|\text{na}) &= \frac{p(\text{na}|\text{Regen}) \times p(\text{Regen})}{p(\text{na})} \\
 &= \frac{p(\text{na}|\text{Regen}) \times p(\text{Regen})}{p(\text{na}|\text{Regen}) \times p(\text{Regen}) + p(\text{na}|\text{RS}) \times p(\text{RS}) + p(\text{na}|\text{G}) \times p(\text{G})} \\
 &= \frac{100\% \times 40\%}{70\%} = 57\%
 \end{aligned}$$

In [3]: *'''make use of Bayes formula'''*

```

p_wet = (1.00*0.40+1.00*0.20+0.20*0.50)
print('p(wet)      ={:.4f}'.format(p_wet))
p_rain = 1.00*0.40/p_wet
print('p(rain|wet)={:.4f}'.format(p_rain))

p(wet)      =0.7000
p(rain|wet)=0.5714

```

9.2 Ergebnis:

mehrere mögliche Ursachen spielen für die bedingte Wahrscheinlichkeit eine Rolle und können "entlastend" wirken.

explaining away

9.3 Anwendung:

- Gericht
 - Zeugenaussagen
 - Wahrheitsgehalt gewichten
- Medizin
 - Symptome \Rightarrow Prognose
- Bayes-Statistik
 - ausführlich in **Angewandte Statistik II**

10 Zusammenfassung

- Zufallsvariable
- Wahrscheinlichkeitsraum (Ω, Σ, P)
- Zufallsexperiment
 - Wahrscheinlichkeit $P(X=x_i)$
 - Ergebnisse $\omega_i \in \Omega$
 - Ereignisse $x_i \in \Sigma$
- Verbundene Ereignisse
 - bedingte Wahrscheinlichkeit $p(A|B) = \frac{p(A \cap B)}{p(B)}$ wenn $p(B) > 0$
 - totale Wahrscheinlichkeit $P(B) = \sum_{k=1}^K P(B|A_k) \cdot P(A_k)$
 - Verbundwahrscheinlichkeit $p(A \cap B) = p(A|B) \cdot p(B)$
 - Unabhängigkeit
 - * $p(A|B) = p(A)$
 - * $p(A \cap B) = p(A) \cdot p(B)$
 - Satz von Bayes \$ bestimmt $p(A|B)$ mittels $p(B|A)$

10.1 Ausblick

- Wert des Zufallsexperiments
- Wetteinsatz und Wettgewinn

11 Fragen?

033_Folien

November 23, 2018

```
In [1]: import numpy as np # mathematical methods
```

1 Wahrscheinlichkeitstheorie

1.1 Diskrete Wahrscheinlichkeitstheorie - Überblick

1. Zufall

- Zufallsvariable
- Zufallsexperiment
- WahrscheinlichkeitsmaSS
- Wahrscheinlichkeitsverteilung
- Axiome von Kolmogorov

2. Zusammengesetzte Wahrscheinlichkeiten

- bedingte Wahrscheinlichkeit
- gemeinsame Wahrscheinlichkeit
- totale Wahrscheinlichkeit
- Satz von Bayes

3. KenngröSSen

- Erwartungswert
- Varianz

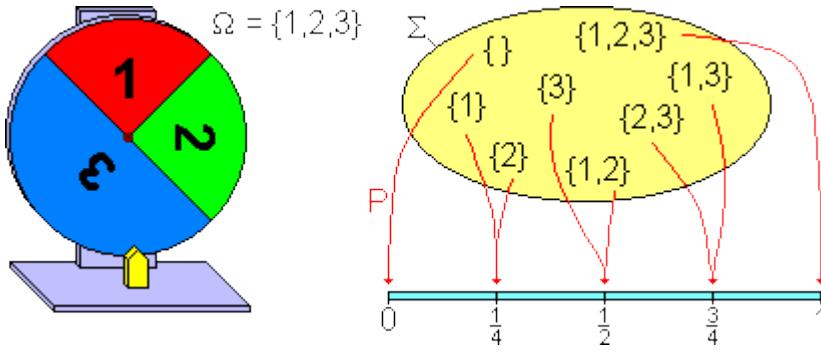
4. Modellverteilungen

1.2 Wahrscheinlichkeitsraum (Ω, Σ, P)

- Ergebnisraum $\Omega = \{\omega_i\}$: Menge aller elementaren Ergebnisse
- Ereignisraum Σ , bestehend aus allen möglichen Teilmengen von Ω
- WahrscheinlichkeitsmaSS P
 - ordnet jedem Ereignis aus Σ eine Wahrscheinlichkeit P zu
- Zufallsexperiment
- Kolmogorov-Axiome
 - Additivität

In [1]:

Out[1]:



2 Erfolg?

2.1 Beispiel Glücksrad

```
In [3]: '''wheel of fortune
    win 1      if number 3
    lose 1     if number 1
    no effect if number 2 '''
N = 100                                # number of games to play
# wheel randomization: 3's probability is twice the one of 1 and 2
x = np.random.choice([1, 2, 3], size=N, p=[1/4, 1/4, 1/2])
print(x[:20], '...')                      # show 1st 20 numbers drawn from wheel
wallet = 0                                 # start with empty pockets, assume credit for bet
for i in x:                               # all N bets sum up to
    wallet += i-2                          # win = number - bet
print('my win: {}'.format(wallet))

[3 3 3 1 3 3 3 3 1 2 3 3 3 2 3 2 2 2 3 1] ...
my win: 23
```

```
In [4]: '''---cards reloaded---'''
# cards    7 8 9   10   J   Q   K   A
x = np.array([0, 10, 2, 3, 4, 11])      # value of card
p = np.ones_like(x)/8.                     # probability of card 4/32
p[0]*=3.                                  # except for x=0  3 cards: 7,8,9
y = p*x                                    # probability times value
original = np.get_printoptions()           # allow formatted printing
np.set_printoptions(formatter={'float': '{: 8.4f}'.format})
print('x={}'.format(1.0*x))
print('p={}'.format(p))
print('y={}'.format(y))
ym = y.sum()                               # average value
print('drawing one card gives in average a value of {}'.format(ym))
print(' total pips in deck = {}'.format(32.*ym))
np.set_printoptions(**original)             # restore print formatting
print(' btw. mean of x is {}'.format(x.mean()))

x=[ 0.0000  10.0000   2.0000   3.0000   4.0000   11.0000]
p=[ 0.3750   0.1250   0.1250   0.1250   0.1250   0.1250]
y=[ 0.0000   1.2500   0.2500   0.3750   0.5000   1.3750]
drawing one card gives in average a value of 3.75
 total pips in deck = 120.0
btw. mean of x is 5.0
```

3 Erwartungswert

Beschreibende Statistik Mittelwert aus mehreren Beobachtungen

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

Beim Histogramm (Klasseneinteilung)

$$\approx \bar{x}' = \frac{1}{\sum_{k=1}^{N_k} h_k} \sum_{k=1}^{N_k} h_k(x_k) \cdot x_k$$

Häufigkeitsinterpretation Der Wert der bei sehr vielen $n \rightarrow \infty$ Versuchen im Mittel erreicht wird.
↔ wahrscheinlichster Wert

Wert, den wir fairerweise erwarten

- Wettquote, langfristige Gewinnchance
- Abgeleitet aus theoretischen Überlegungen

3.0.1 die Werte und deren Verteilung

Der Erwartungswert hängt von den Werten *und* von der Wahrscheinlichkeitsverteilung für die Werte ab - seltene tragen weniger dazu bei als häufigere - ist ein *fester* Wert

3.1 Definition Erwartungswert

Sei X eine diskrete Zufallsvariable

- mit **Ergebnissen** (Elementarereignissen) $x_i \ i \in \{1, 2, \dots, N\}$
- und deren **Wahrscheinlichkeitsverteilung** $P(X = x)$, Abkürzung $p_i = p(X = x_i)$

dann ist der **Erwartungswert** von X :

$$\mu_X = \mathcal{E}(X) = \sum_{i=1}^N p_i \cdot x_i$$
$$\mathcal{E}(X) = \mu = \sum_{i=1}^N p_i \cdot x_i$$

3.1.1 Beispiel Glücksrad von vorhin

Einsatz: 2

Gewinn: Zahl

```
x=-1 p=1/4
x= 0 p=1/4
x=+1 p=1/2
```

$$\mathcal{E}(X) = \frac{1}{4}$$

```
In [5]: '''example: wheel of fortune'''
xbsp = np.array([-1, 0, 1])      # outcomes -1 if number=1; +1 if number=3; 0 else
pbps = np.array([.25, .25, .50]) # probability according to angle on wheel
print('normalized probability distribution? sum={:.3f}'.format(pbps.sum()))
expectation = np.asarray([x*p for x, p in zip(xbsp, pbps)]).sum()
print('expectation value of x = {:.3f}'.format(expectation))

normalized probability distribution? sum=1.000
expectation value of x = 0.250
```

3.2 Transformationsregel für Erwartungswerte

Sei $g(x)$ eine reelle Funktion, dann gilt für $Y = g(X)$

$$\mathcal{E}(Y) = \mathcal{E}(g(X)) = \sum_{i=1}^N g(x_i)p_i$$

3.2.1 Lineare Transformation für Erwartungswerte

Sei $Y = aX + b$, dann ist

$$\mathcal{E}(Y) = \mathcal{E}(aX + b) = a\mathcal{E}(X) + b$$

Beweis $\mathcal{E}(aX + b) = \sum_{i=1}^N (ax_i + b)p_i = a \sum_{i=1}^N x_i p_i + b \sum_{i=1}^N p_i = a\mathcal{E}(X) + b \cdot 1$

3.3 Zusammenfassung Erwartungswert

Definition:

$$\mathcal{E}(X) = \mu_X = \sum_{i=1}^N p_i \cdot x_i$$

- entspricht dem arithmetischen Mittelwert, wenn Laplace'sche Gleichverteilung angenommen wird
 - $p = 1/N$
- entspricht dem arithmetischen Mittelwert, wenn relative Häufigkeiten (Klasseneinteilung) angenommen werden
 - $p_i = \frac{n_i}{N}$
- Erstes Moment von x über seiner Wahrscheinlichkeitsverteilung $p_i = p(x_i)$
- durch Wahrscheinlichkeitsverteilung festgelegt.
 - charakteristischer Wert
 - jedes einzelne Zufallsergebnis kann und wird durchaus vom Erwartungswert abweichen
 - das durchschnittliche Ergebnis von vielen gleichen Zufallsexperimenten

4 Streuung

4.1 Definition Varianz

Sei X eine Zufallsvariable mit Ausprägungen $x_i \quad i \in \{1, 2, \dots, N\}$, Erwartungswert μ und Wahrscheinlichkeitsverteilung $P(X = x)$, Abkürzung $p_i = p(X = x_i)$, dann ist die Varianz von X

$$\sigma^2 = \text{Var}(X) = \sum_{i=1}^N p_i(x_i - \mu)^2$$

```
In [6]: '''variance for example wheel of fortune - a'''
# erw = np.asarray([x*p for x, p in zip(xbsp, pbsp)]).sum() # from above
# variance defined
var_a = np.asarray([p*(x-expectation)**2 for x, p in zip(xbsp, pbsp)]).sum()
print('variance as defined = {:.4f}'.format(var_a))

variance as defined = 0.6875
```

$$\text{Var}(X) = \sigma^2 = \sum_{i=1}^N p_i(x_i - \mu)^2$$

4.1.1 Varianz als quadratische Abweichung

Führt man eine neue Zufallsvariable $(X - \mu)^2$ ein, so ist

$$\text{Var}(X) = \mathcal{E}((X - \mu)^2)$$

Interpretation Erwartete quadratische Abweichung von X vom Mittelwert μ .

Empirische Statistik: *durchschnittliche* quadratische Abweichung

$$\tilde{s}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

In [7]: *'variance for example wheel of fortune - b'*

```
var_b = np.asarray([x*p for x, p in zip(xbsp-expectation)**2, pbsp]]).sum()
print('variance as expected squared difference = {:.4f}'.format(var_b))
```

variance as expected squared difference = 0.6875

4.1.2 Verschiebungssatz für Varianzen

$$\text{Var}(X) = \mathcal{E}(X^2 - \mathcal{E}(X)^2) = \mathcal{E}(X^2) - \mu^2$$

$$\text{Var}(X) = \mathcal{E}((X - \mu)^2)$$

$$= \mathcal{E}(X^2 - 2\mu X + \mu^2)$$

Beweis

$$\begin{aligned} &= \mathcal{E}(X^2) + \mathcal{E}(-2\mu X) + \mathcal{E}(\mu^2) \\ &= \mathcal{E}(X^2) - 2\mu \mathcal{E}(X) + \mu^2 \\ &= \mathcal{E}(X^2) - 2\mu^2 + \mu^2 \end{aligned}$$

In [8]: *'variance for example wheel of fortune - c'*

```
var_c = np.asarray([x*p for x, p in zip(xbsp**2, pbsp]]).sum()
var_c -= expectation**2
print('variance centralized = {:.4f}'.format(var_b))
```

variance centralized = 0.6875

4.1.3 Varianz unter linearer Transformation

$$\text{Var}(aX + b) = a^2 \text{Var}(X)$$

Beweis: $\text{Var}(aX + b) = \mathcal{E}([aX + b - \mathcal{E}(aX + b)]^2)$

$$\begin{aligned} &= \mathcal{E}([aX + b - a\mathcal{E}(X) - b]^2) \\ &= a^2 \mathcal{E}([X - \mathcal{E}(X)]^2) = a^2 \text{Var}(X) \end{aligned}$$

In [9]: *'displacement rule for example wheel of fortune'*

```
expT = np.asarray([x*p for x, p in zip(2*xbsp-3, pbsp]]).sum()
varT = np.asarray([x*p for x, p in zip(((2*xbsp-3)-expT)**2, pbsp]]).sum()
print('variance under linear transformation 2x-3 = {:.4f}'.format(varT))
```

variance under linear transformation 2x-3 = 2.7500

5 Zusammenfassung:

5.0.1 Wichtige Kennzahlen einer Zufallsvariablen X sind

Erwartungswert

$$\mathcal{E}(X) = \mu_X = \sum_{i=1}^N p_i \cdot x_i$$

Varianz

$$\begin{aligned} \text{Var}(X) = \sigma_X^2 &= \sum_{i=1}^N p_i (x_i - \mu)^2 \\ &= \mathcal{E}((X - \mu_X)^2) \end{aligned}$$

Beides sind *feste Größen* der Zufallsvariablen X

- Ergebnisse, Werte(bereich) x_i
- Wahrscheinlichkeitsverteilung p_i

5.0.2 Vergleiche mit

Arithmetischem Mittelwert

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

Empirischer Varianz

$$s^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$$

In [10]: *random variable X in Python*

```
omega = np.array([1, 2, 3]) # possible results of random experiment, e.g. wheel of fortune
probabilities = np.array([.25, .25, .50]) # ... and their corresponding probabilities
```

In [11]: *draw:*

```
    out of random variable X with elementary results X1..X3
    with probabilities p1..p3 (same number as X, sum up to 1 (normalization))
    N=100 random experiments'
xi = np.random.choice(omega, size=100, p=probabilities)
print('the result of {} draws gave {:.4f} points in average'.format(len(xi), xi.mean()))
```

the result of 100 draws gave 2.3300 points in average

In [12]: *calculate Expectation Value*

```
expectation = (probabilities*omega).sum()          # numpy '*' multiplies element wise
'calculate Variance'
Var = (probabilities*(omega-expectation)**2).sum()
print('Wheel of fortune has expectation value of {} with standard deviation {:.5f}'.format(expectation, np.sqrt(Var)))
```

Wheel of fortune has expectation value of 2.25 with standard deviation 0.82916

6 Fragen?

041_Folien

2018 年 11 月 23 日

```
In [1]: import numpy as np                      # mathematical methods
         from matplotlib import pyplot as plt      # plotting methods
         %matplotlib inline
```

0.1 Diskrete Wahrscheinlichkeitstheorie - Überblick

1. Zufall

- Zufallsvariable, Zufallsexperiment, WahrscheinlichkeitsmaSS, Wahrscheinlichkeitsverteilung, Axiome von Kolmogorov

2. Zusammengesetzte Wahrscheinlichkeiten

- bedingte Wahrscheinlichkeit, gemeinsame Wahrscheinlichkeit, totale Wahrscheinlichkeit, Satz von Bayes

3. KenngröSSen

- Erwartungswert & Varianz

4. Modellverteilungen

- Laplace-Experiment
- Bernoulli-Experiment
- Binomiale Verteilung
- Geometrische Verteilung
- Poissonverteilung

1 Laplace Experiment

- endlich viele (M) Elementarereignisse 最终 M 个元素事件发生
- alle Elementarereignisse sind gleich wahrscheinlich 所有的元素事件是等概率的

$$P(\omega_i) = p = \frac{1}{M} = \frac{1}{|\Omega|} \quad i \in \{1 \dots M\}$$

- Wahrscheinlichkeit = $\frac{\text{Anzahl für A günstige Ergebnisse}}{\text{Anzahl für A mögliche Ergebnisse}}$

$$p(A) = \frac{|A|}{|\Omega|} = \frac{N}{M}$$

Beispiele:

- Münzwurf, Würfel, Urne mit farbigen Kugeln, Roulette, Kartenspiel (Spieltheorie)
- Ziffern von π (Mathematik)

1.1 Kennzahlen**Erwartungswert**

$$\mathcal{E}(X) = \mu_X = \bar{x} = \frac{1}{M} \sum_{i=1}^M x_i$$

Varianz

$$\text{Var}(X) = \frac{1}{M} \sum_{i=1}^M (x_i - \bar{x})^2$$

[ÜA] Bitte nachrechnen!

2 Bernoulli-Experiment

Ereignis A tritt ein oder tritt nicht ein

$$\Omega = \{A, \bar{A}\}$$

Binäre Zufallsvariable X (*Indikatorvariable*) kann nur Werte $\omega \in \{0, 1\}$ annehmen.

$$X = \begin{cases} 1 & \text{wenn } A \text{ zutrifft} \\ 0 & \text{wenn } A \text{ nicht zutrifft} \end{cases}$$

Beispiele

- Münzwurf: Kopf / Zahl
- Produktion: innerhalb Toleranz / Ausschuss
- Geburten: Mädchen / Jungen
- Psychophysik: gesehen / nicht gesehen

Unterscheidung

- MuSS nicht Laplace sein, es darf $p(A) \neq p(\bar{A})$

2.1 Bernoulli-Verteilung

Die Wahrscheinlichkeit für A sei $\pi = \text{const.}$

$$\begin{aligned} P(A) &= P(X=1) = \pi \\ P(\bar{A}) &= P(X=0) = 1 - \pi \end{aligned}$$

2.2 Kennzahlen

Der Erwartungswert einer Bernoulli-Zufallsvariablen ist

$$\mathcal{E}(X) = \pi$$

Die Varianz ist

$$\text{Var}(X) = \pi(1 - \pi)$$

[ÜA] Bitte nachrechnen!

3 Mehrfache identische, unabhängige Wiederholung eines Zufallsexperiments - i.i.d.

Beispiel

- Mehrfaches unabhängiges Werfen einer Münze
- Gleichzeitiges unabhängiges Werfen mehrerer identischer Münzen

Identischer Wahrscheinlichkeitsraum: Zufallsvariable, Wahrscheinlichkeitsverteilung

- *i.i.d.*: identically independent distributed

4 Binomial-Verteilung

n -malige unabhängige Wiederholung des gleichen Bernoulli-Zufallsexperiments ("i.i.d."): Zufallsvariable X ist jetzt die Anzahl der (positiven) Einzel-Ereignisse $A_i \in \{0, 1\}$

$$P(A_i = 1) = \pi \quad \forall i$$

$$X = \sum_{i=1}^n A_i$$

4.0.1 Beispiel:

drei Münzwürfe mit ein-und-derselben fairen Münze, wie oft kommt Kopf?

X : Mögliche Ergebnisse	Wahrscheinlichkeit	p=1/2
0 : {(0,0,0)}	(1-p)(1-p)(1-p)	p = 1/8
1 : {(1,0,0), (0,1,0), (0,0,1)}	p(1-p)(1-p)+(1-p)p(1-p)+(1-p)(1-p)p	p = 3/8
2 : {(1,1,0), (1,0,1), (0,1,1)}	p*p*(1-p)+p*(1-p)*p+(1-p)*p*p	p = 3/8
3 : {(1,1,1)}	p^3	p = 1/8

4.1 Wahrscheinlichkeitsverteilung

$$P(X=x) = \binom{n}{x} \pi^x (1-\pi)^{(n-x)} \quad x \in \{0 \dots n\}$$

$\binom{n}{k}$ (sprich: " n über k ") Anzahl der Variationen von k aus n Elementen.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \cdot (n-1) \cdot \dots \cdot (n-k+1)}{k \cdot (k-1) \cdot \dots \cdot 1}$$

Python:

```
In [3]: from scipy.special import comb
n = 49
k = 6
nck = comb(n, k, exact=False)
print('({}={} choose k={}) yields {}'.format(n, k, nck))
```

```
(n=49 choose k=6) yields 13983816.0
```

```
In [4]: print('binomial triangle')
    for n in range(10):
        nf = np.math.factorial(n)
        tr = [int(nf/(np.math.factorial(k)*np.math.factorial(n-k))) for k in range(n+1)]
        print('{:2d}: {}'.format(n, tr))
    print('...')

binomial triangle
0: [1]
1: [1, 1]
2: [1, 2, 1]
3: [1, 3, 3, 1]
4: [1, 4, 6, 4, 1]
5: [1, 5, 10, 10, 5, 1]
6: [1, 6, 15, 20, 15, 6, 1]
7: [1, 7, 21, 35, 35, 21, 7, 1]
8: [1, 8, 28, 56, 70, 56, 28, 8, 1]
9: [1, 9, 36, 84, 126, 126, 84, 36, 9, 1]
...
...
```

4.1.1 Normierung

$$\sum_{i=1}^{N=n+1} p_i = \sum_{x=0}^n \binom{n}{x} \pi^x (1-\pi)^{(n-x)} = 1$$

4.1.2 Parameter

Die Binomialverteilung \mathcal{B} lässt sich durch zwei Parameter

- n : Anzahl Versuche
- π : Wahrscheinlichkeit für das Eintreten des positiven Falles

eindeutig beschreiben: $\mathcal{B}_{(n,\pi)}$

Sie ist dann eine Funktion der Zufallsvariablen X , der Anzahl der positiven Ergebnisse:

$$\mathcal{B}_{(n,\pi)}(x) = P(X=x) = \binom{n}{x} \pi^x (1-\pi)^{(n-x)} \quad x \in \{0 \dots n\}$$

4.2 Kennzahlen

Der **Erwartungswert** einer Binomial-verteilten Zufallsvariablen X ist

$$\mathcal{E}(X) = n \cdot \pi$$

Die **Varianz** ist

$$\text{Var}(X) = n \cdot \pi \cdot (1 - \pi)$$

4.3 Python

Statistik-Paket `scipy.stats`

- Referenz <http://docs.scipy.org/doc/scipy/reference/stats.html>
- open source unter BSD Lizenz <https://www.scipy.org/scipylib/license.html>
- gehostet auf <https://github.com/scipy/scipy>
- Version SciPy 1.1.0 released 2018-05-05
- verwenden:

```
In [5]: '''statistical methods'''
```

```
from scipy import stats
```

```
stats.binom?
```

```
In [6]: '''binomial distribution - an example'''
```

```
n = 10      # number of repetitions
p = .5       # probability pi of single event
x = 5        # one example result: number of successes
print('probability of x={} as number of "positive" events is Binomial({}, {}, {}) = {:.5f}'.
      format(x, x, n, p, stats.binom(n, p).pmf(x)))
print('expectation value of Binomial({}, {}) = {:.5f}'.
      format(n, p, stats.binom(n, p).expect()))
```

```
probability of x=5 as number of "positive" events is Binomial(5, 10, 0.5) = 0.24609
```

```
expectation value of Binomial(10, 0.5) = 5.00000
```

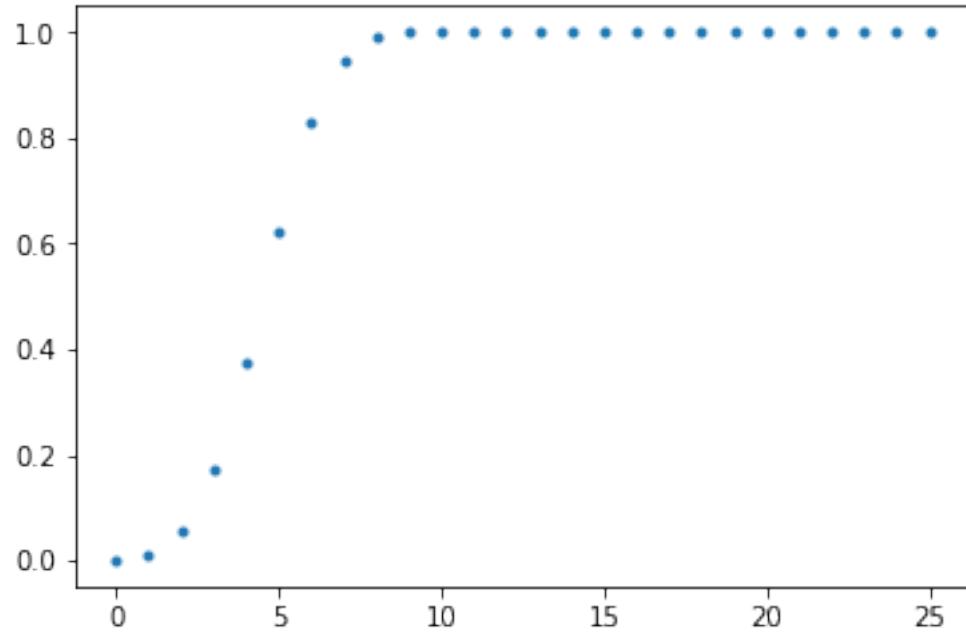
```
In [3]: from scipy import stats
```

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

```
x = np.arange(26)
```

```
plt.plot(stats.binom(10, .5).cdf(x), '.')
```

```
Out[3]: [<matplotlib.lines.Line2D at 0x7f0f2c6392b0>]
```



In [7]: *'''Binomial probability distributions'''*

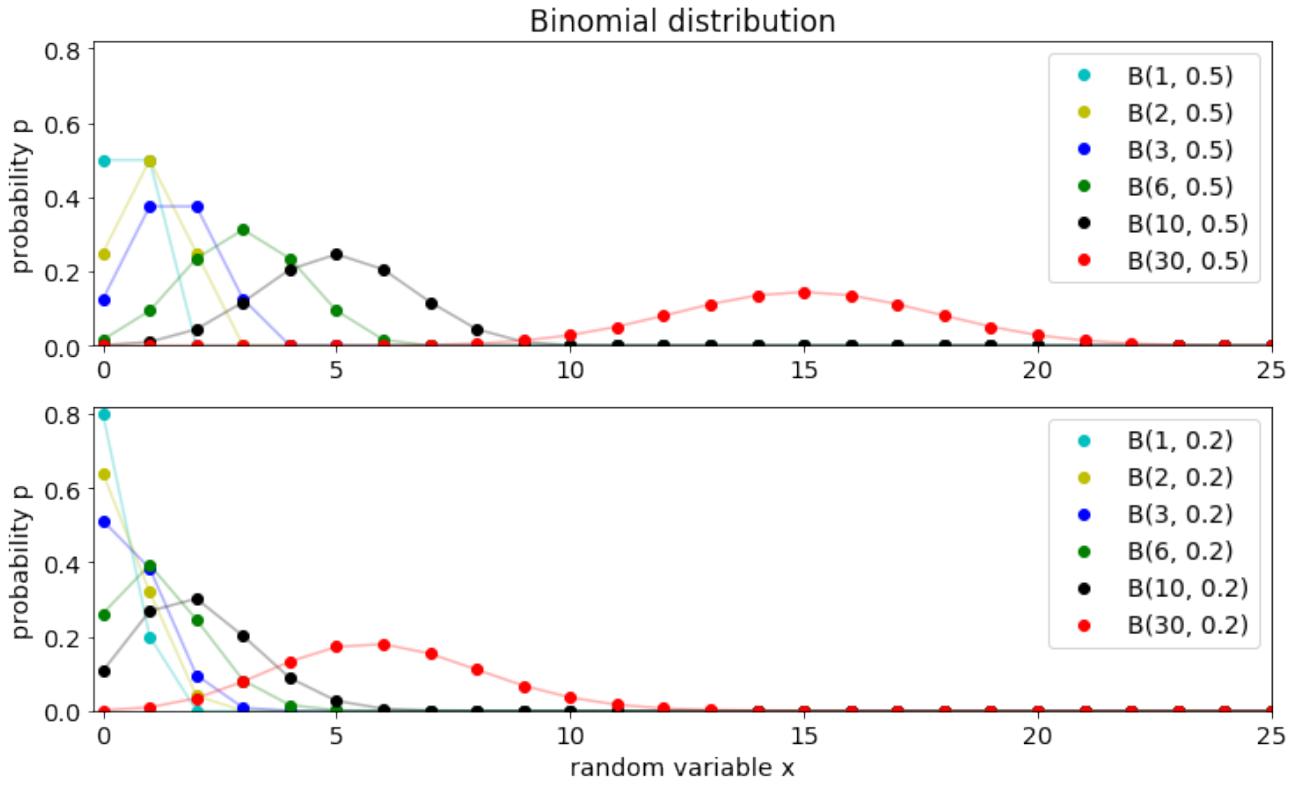
```

xs = np.arange(26)                                # number of successes x
p = 0.5                                         # probability pi of 1st examples
colors = ['c', 'y', 'b', 'g', 'k', 'r']          # cycle through colors
ns = [1, 2, 3, 6, 10, 30]                         # variety of n to show

f = plt.figure(figsize=(12, 7))                  # big canvas
f.add_subplot(2, 1, 1)                            # top figure for 1st examples
plt.title('Binomial distribution')
plt.ylabel('probability p')
plt.axis((-2, 25., 0., 0.82))
for i, n in enumerate([1, 2, 3, 6, 10, 30]):
    ps = stats.binom(n, p).pmf(xs)
    plt.plot(xs, ps, colors[i]+'.o', label='B({}, {})'.format(n,p))
    plt.plot(xs, ps, colors[i]+'-', alpha=.3)
plt.legend()

f.add_subplot(2, 1, 2)                            # bottom figure for 2nd examples
plt.ylabel('probability p')
plt.xlabel('random variable x')
p = 0.2                                         # probability pi of 2nd examples
plt.axis((-2, 25., 0., 0.82))
for i, n in enumerate([1, 2, 3, 6, 10, 30]):
    ps = stats.binom(n, p).pmf(xs)
    plt.plot(xs, ps, colors[i]+'.o', label='B({}, {})'.format(n,p))
    plt.plot(xs, ps, colors[i]+'-', alpha=.3)
plt.legend();

```



4.4 Eigenschaften

Addition Sind $X \sim \mathcal{B}(n, \pi)$ und $Y \sim \mathcal{B}(m, \pi)$ Binomial-verteilt und voneinander unabhängig, so ist

$$X + Y \sim \mathcal{B}(n + m, \pi)$$

Symmetrie Sei $X \sim \mathcal{B}(n, \pi)$, dann gilt für $Z = n - X$

$$Z \sim \mathcal{B}(n, 1 - \pi)$$

5 Geometrische Verteilung

n-malige unabhängige Wiederholung (*i.i.d.*) des gleichen Bernoulli-Zufallsexperiments: Zufallsvariable X ist Anzahl der Versuche mit Wahrscheinlichkeit π , die nötig sind **bis das erste Mal** das positive Ereignis $A_i \in \{0, 1\}$ eintritt.

$$P(A_i=1) = \pi \quad \forall i$$

$$X = n \quad \text{für} \quad A_{i < n} = 0, \quad A_n = 1$$

Beispiel Rechnerabstürze Sei p die Wahrscheinlichkeit, dass ein PC abstürzt. Wie lange kann man im Mittel arbeiten?

Das Ereignis könnte so aussehen

Stunden	1	2	3	4	5	6	7	8	9
Absturz	0	0	0	0	0	0	1		

- $x = 6$

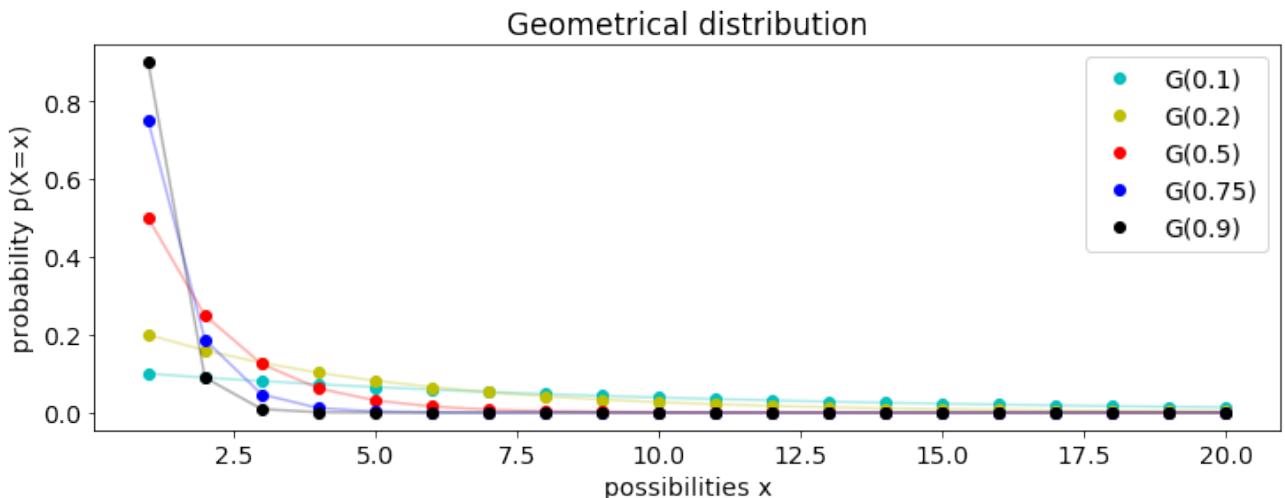
- 5h kein Absturz mit jeweils Wahrscheinlichkeit $(1 - \pi)$ also $p = (1 - \pi)^5$
- 1h der Absturz mit Wahrscheinlichkeit π
- Gesamtwahrscheinlichkeit $p_6 = (1 - \pi)^5 \cdot \pi$

5.1 Wahrscheinlichkeitsverteilung

$$P(X=x) = (1 - \pi)^{x-1} \cdot \pi \quad x \in \{1 \dots n\}$$

In [8]: *'''Geometric probability distributions'''*

```
f = plt.figure(figsize=(12, 4))
plt.title('Geometrical distribution')
plt.xlabel('possibilities x')
plt.ylabel('probability p(X=x)')
colors = ['c', 'y', 'r', 'b', 'k', 'r']
xs = np.arange(1., 21)
for i, pi in enumerate([.1, .2, .5, .75, .9]):
    ps = stats.geom(pi).pmf(xs)
    plt.plot(xs, ps, colors[i] + 'o', label='G({})'.format(pi))
    plt.plot(xs, ps, colors[i] + '-.', alpha=.3)
plt.legend();
```



5.1.1 Normierung:

Die Geometrische Wahrscheinlichkeitsverteilung ist normiert:

$$\sum_{i=1}^{\infty} p_i = \sum_{x=0}^{\infty} (1 - \pi)^{(x-1)} \cdot \pi = 1$$

In [9]: *'''is geometrical distribution normalized? Not as a proof but ...'''*

```
pi = 0.4 # probability as an example
nmax = 24 #
p_sum = 0. # sum all probabilities
for x in np.arange(1, nmax):
```

```

p = pi * (1-pi)**(x-1)
p_sum += p
print('x={:2d} has p={:.5f} which sums up to {:.5f}'
      .format(x, p, p_sum))

x= 1 has p=0.40000 which sums up to 0.40000
x= 2 has p=0.24000 which sums up to 0.64000
x= 3 has p=0.14400 which sums up to 0.78400
x= 4 has p=0.08640 which sums up to 0.87040
x= 5 has p=0.05184 which sums up to 0.92224
x= 6 has p=0.03110 which sums up to 0.95334
x= 7 has p=0.01866 which sums up to 0.97201
x= 8 has p=0.01120 which sums up to 0.98320
x= 9 has p=0.00672 which sums up to 0.98992
x=10 has p=0.00403 which sums up to 0.99395
x=11 has p=0.00242 which sums up to 0.99637
x=12 has p=0.00145 which sums up to 0.99782
x=13 has p=0.00087 which sums up to 0.99869
x=14 has p=0.00052 which sums up to 0.99922
x=15 has p=0.00031 which sums up to 0.99953
x=16 has p=0.00019 which sums up to 0.99972
x=17 has p=0.00011 which sums up to 0.99983
x=18 has p=0.00007 which sums up to 0.99990
x=19 has p=0.00004 which sums up to 0.99994
x=20 has p=0.00002 which sums up to 0.99996
x=21 has p=0.00001 which sums up to 0.99998
x=22 has p=0.00001 which sums up to 0.99999
x=23 has p=0.00001 which sums up to 0.99999

```

5.2 Kennzahlen

Der **Erwartungswert** einer geometrisch-verteilten Zufallsvariablen X ist

$$\mathcal{E}(X) = \frac{1}{\pi}$$

Die **Varianz** ist

$$\text{Var}(X) = \frac{1 - \pi}{\pi^2}$$

5.2.1 Für das Beispiel der Rechnerabstürze ergibt sich

- ein Erwartungswert von 5 Stunden
- bei einer Streuung von $\sqrt{\frac{0.8}{0.2^2}} = 4,5$ Stunden

6 Poisson-Verteilung

Die Anzahl von seltenen Ereignissen innerhalb eines festen Intervalls werden durch die Poisson-Statistik beschrieben.
 $X_i \in \{0, 1, 2, \dots\}$

Voraussetzungen:

- kein gleichzeitiges Eintreten zweier Ereignisse
- Wahrscheinlichkeit für Ereignis in kleinem Intervall Δt ist $\lambda \cdot \Delta t$
 - λ ist eine Intensitätsrate
 - Ereignisse sind selten
- Wahrscheinlichkeit für $X_i = x$ ist in jedem beliebigen Intervall gleicher Länge gleich

6.1 Wahrscheinlichkeitsverteilung

$$P(X=x) = \frac{\lambda^x}{x!} e^{-\lambda}$$

- Einziger Parameter der Poissonverteilung ist die Rate λ : $\mathcal{P}_l(\lambda)$
- Funktion von x , der Anzahl der Ereignisse im betrachteten Intervall

6.2 Kennzahlen

Der **Erwartungswert** einer Poisson-verteilten Zufallsvariablen $X \sim \mathcal{P}_l(\lambda)$ ist

$$\mathcal{E}(X) = \lambda$$

Die **Varianz** ist

$$\text{Var}(X) = \lambda$$

Beispiel Sternschnuppen Sei $\lambda = 0.2$ die Wahrscheinlichkeit eine Sternschnuppe pro Minute zu sehen. Wie lange muß man im Mittel warten um eine zu sehen?

Das Ereignis könnte so aussehen

Minute	1	2	3	4	5	6	7	8	9	10
Schnuppe	0	0	0	1	0	0	0	0	1	0

6.2.1 Für das Beispiel der Sternschnuppen ergibt sich

- ein Erwartungswert von 0.2 je Minute
- bei einer Streuung von 0.2 je Minute

6.3 Eigenschaften

Intervallstreckung Sei $X \sim \mathcal{P}_l(\lambda)$ auf dem Einheitsintervall, dann ist Z im Intervall der Länge l Poissonverteilt mit

$$Z \sim \mathcal{P}_l(l \cdot \lambda)$$

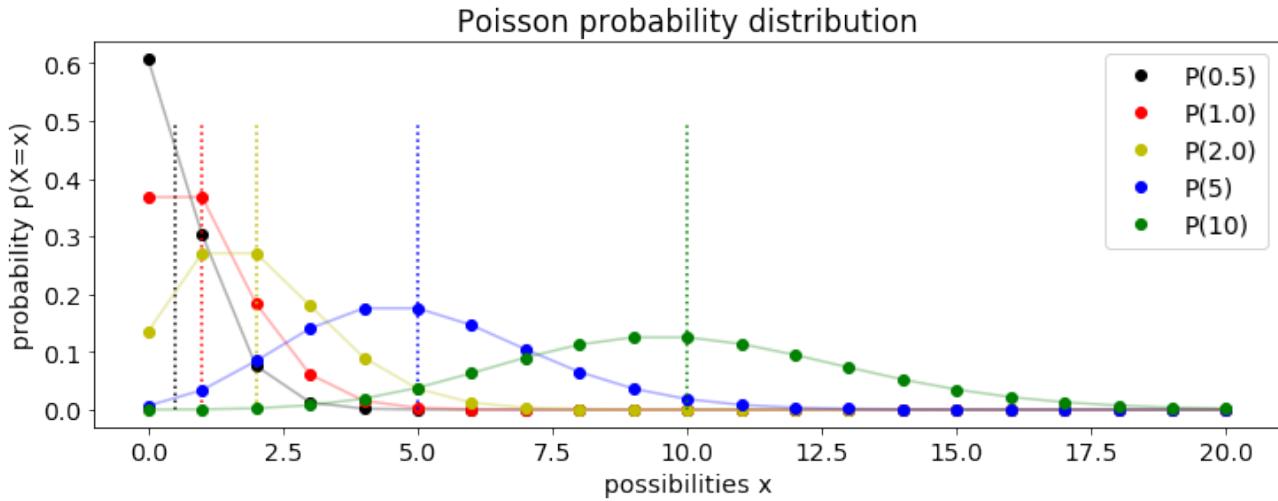
6.4 Wahrscheinlichkeitsverteilung

```
In [10]: '''Poisson probability distributions'''
f = plt.figure(figsize=(12, 4))
plt.title('Poisson probability distribution')
plt.xlabel('possibilities x')
plt.ylabel('probability p(X=x)')
```

```

colors = ['k', 'r', 'y', 'b', 'g', 'r']
xs = np.arange(0., 21)
for i, lbda in enumerate([.5, 1., 2., 5, 10]):
    ps = stats.poisson(lbda).pmf(xs)
    plt.plot(xs, ps, colors[i]+'.', label='P({})'.format(lbda)) # dots at p
    plt.plot(xs, ps, colors[i]+'-', alpha=.3) # connecting line
    plt.plot(2*[stats.poisson(lbda).expect()], [0, 0.5], colors[i]+':') # expectation
plt.legend();

```



6.5 Kumulierte Verteilungsfunktion

$$F(x) = \sum_{i=1}^j p(x_i) \quad j = \max(i \mid x_i \leq x)$$

$$F(x) = \sum_{x_i \leq x} p(x_i)$$

In [11]: *'Poisson probability distributions'*

```

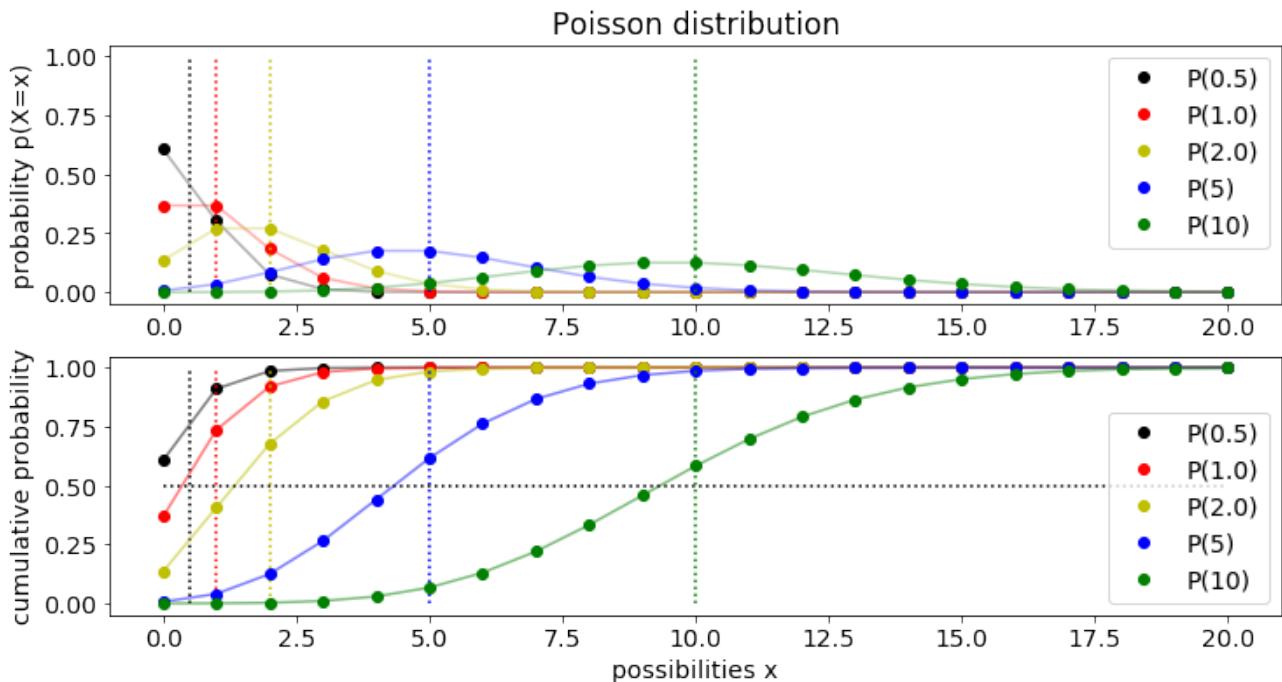
f = plt.figure(figsize=(12, 6))
f.add_subplot(2, 1, 1)
plt.title('Poisson distribution')
plt.ylabel('probability p(X=x)')
colors = ['k', 'r', 'y', 'b', 'g', 'r']
xs = np.arange(0., 21)
for i, lbda in enumerate([.5, 1., 2., 5, 10]):
    ps = stats.poisson(lbda).pmf(xs)
    plt.plot(xs, ps, colors[i]+'.', label='P({})'.format(lbda)) # dots at p
    plt.plot(xs, ps, colors[i]+'-', alpha=.3) # connecting line
    plt.plot(2*[stats.poisson(lbda).expect()], [0, 1.0], colors[i]+':') # expectation
plt.legend()
f.add_subplot(2, 1, 2)

```

```

plt.xlabel('possibilities x')
plt.ylabel('cumulative probability')
colors = ['k', 'r', 'y', 'b', 'g', 'r']
x = np.arange(0., 21)
for i, lbda in enumerate([.5, 1., 2., 5, 10]):
    ps = stats.poisson.cdf(x, lbda)
    plt.plot(x, ps, colors[i] + 'o', label='P({})'.format(lbda))
    plt.plot(x, ps, colors[i] + '-.', alpha=.5)
    plt.plot(2*[stats.poisson(lbda).expect()], [0, 1.0], colors[i] + ':')
plt.plot([0, 20], 2*[.5], 'k:')
plt.legend(loc='lower right');

```



7 Diskrete Verteilungen - Zusammenfassung

7.0.1 Voraussetzungen

i.i.d. Unabhängige identische Wiederholung

- der selbe Wahrscheinlichkeitsraum

Bernoulli-Experiment

- zwei (dichotome) Fälle A, \bar{A}
- Wahrscheinlichkeiten

$$P(X=A) = \pi \quad P(X=\bar{A}) = 1 - \pi$$

7.0.2 Binomialverteilung

- n Bernoulli-Experimente
- Zufallsvariable ist Anzahl der *Erfolge* $x \in [0 \dots n]$
- Parameter der Binomialverteilung:
 - n = Anzahl der durchgeführten Wiederholungen
 - π = Wahrscheinlichkeit des Bernoulli-Experiments

7.0.3 Geometrische Verteilung

- Bernoulli-Experimente bis zum ersten Eintreten von *Erfolg*
- Zufallsvariable ist Anzahl der Versuche $x \in [0 \dots]$
- Parameter der Geometrischen Verteilung
 - π = Wahrscheinlichkeit des Bernoulli-Experiments

7.0.4 Poissonverteilung

- Seltene Ereignisse in festen Intervallen
- Zufallsvariable ist Anzahl der Beobachtungen $x \in [0 \dots]$
- Parameter der Poissonverteilung
 - Rate λ

8 Zusammenfassung Python

Statsitik-Bibliothek `scipy.stats` enthält diskrete Verteilungen

```
bernoulli(p)
binom(n, p)
geom(p)
poisson(mu)      # "lambda" ist reserviertes Schlüsselwort

np.random.choice # wie .rvs() für Laplace-Verteilungen - und beliebig andere
```

Funktionen und Methoden

```
.expect()          # Erwartungswert einer Verteilung
.pmf(x)           # Wahrscheinlichkeitsverteilung "probability mass function"
.cdf(x)           # Verteilungsfunktion "cumulative distribution function"
.rvs()            # Zufallsergebnis "random variables"
                  # (optional Anzahl der Werte)
```

In [12]: *'''passing parameters to scipy.stats functions, example binomial'''*

```
num = 8
prob = 0.50
x = [0, 1, 2, 3]
ps1 = stats.binom(num, prob).pmf(x)          # two parameters, one (list of) value(s)
ps2 = stats.binom(n=num, p=prob).pmf(x)        # my preferred syntax
ps3 = stats.binom(p=prob, n=num).pmf(x)        # no problem if exchanged
ps4 = stats.binom.pmf(x, num, prob)            # often found, need to know order
```

```
ps5 = stats.binom.pmf(x, n=num, p=prob)      # python-like with key word list
for psi in [ps2, ps3, ps4, ps5]:
    if (psi == ps1).all():
        print('identical')
    else:
        print('{} not identical with {}'.format(psi, ps1))

identical
identical
identical
identical

In [13]: '''freeze a function'''
bdistrib = stats.binom(n=num, p=prob)      # from now on includes parameters num and prob
print(bdistrib)                            # object?
print(bdistrib.rvs(5))                     # get random variables
print(bdistrib.pmf(x))                     #     or probability from binomial(parameters)

<scipy.stats._distn_infrastructure.rv_frozen object at 0x7fc3199a4cf8>
[4 6 7 1 2]
[0.00390625 0.03125   0.109375   0.21875   ]
```

9 Ausblick

- mehrdimensionale Verteilungen
- Kontinuierliche Verteilungen

10 Fragen?

042_Folien

November 23, 2018

```
In [1]: import numpy as np          # mathematical methods
         from scipy import stats      # statistical methods
         from matplotlib import pyplot as plt    # plotting methods
         %matplotlib inline
```

0.1 Diskrete Wahrscheinlichkeitstheorie - Überblick

1. Zufall

- Zufallsvariable, Zufallsexperiment, Wahrscheinlichkeitsmaß, Wahrscheinlichkeitsverteilung, Axiome von Kolmogorov

2. Zusammengesetzte Wahrscheinlichkeiten

- bedingte Wahrscheinlichkeit, gemeinsame Wahrscheinlichkeit, totale Wahrscheinlichkeit, Satz von Bayes

3. Kenngrößen

- Erwartungswert & Varianz

4. Modellverteilungen

- eindimensional

multivariate Verteilungen

multivariate Verteilungen

- bivariate Verteilung
- zusammengesetzte Verteilung
- Erwartungswert
- Varianz
- Kovarianz
- stochastische Unabhängigkeit

1 Zweidimensionale Verteilungen: *bivariat*

- vektorielle Zufallsvariable $W = (X, Y)$

Beispiele

- roter und grüner Würfel gleichzeitig geworfen
- Roulette *rot* und *ungerade*
- Koordinaten (x_i, y_i)
- Alter und Blutdruck bei Patienten
- Pixel Helligkeit und Farbe

1.1 Ein Ergebnis (Elementarereignis)

vektoriell

$$w = (x, y)$$

1.2 Verbundene Wahrscheinlichkeit

$$P(w) = P(X=x \cap Y=y) = p(x,y) = p_{xy}$$

1.3 Stochastische Unabhängigkeit

$$P(X=x \cap Y=y) = P(X=x) \cdot P(Y=y)$$

1.4 Erwartungswert

$$\mu = \begin{pmatrix} \mathcal{E}(X) \\ \mathcal{E}(Y) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N_x} x_i \cdot P(X=x_i) \\ \sum_{j=1}^{N_y} y_j \cdot P(Y=y_j) \end{pmatrix}$$

1.4.1 mit den Randverteilungen

$$P(X=x_i) = \sum_{j=1}^{N_y} P(X=x_i \cap Y=y_j)$$

$$P(Y=y_i) = \sum_{i=1}^{N_x} P(X=x_i \cap Y=y_i)$$

1.4.2 Kontingenztafel

Am Beispiel zweier Münzen

Wahrscheinlichkeit p_{xy} und Randverteilungen p_x und p_y

M1\ M2	0	1	
0	1/4	1/4	1/2
1	1/4	1/4	1/2
	1/2	1/2	1

Ein mögliches Beispiel für Werte x_{ij}

M1\ M2	0	1	
0	0	1	
1	1	4	

1.5 Erwartungswert einer Summe von Zufallsvariablen

Für zwei Zufallsvariablen X und Y ist $Z = X + Y$ wieder eine Zufallsvariable. Es gilt

$$\mathcal{E}(Z) = \mathcal{E}(X + Y) = \mathcal{E}(X) + \mathcal{E}(Y)$$

Beweis

$$\mathcal{E}(X + Y) = \sum_{i=1}^N (x_i + y_i) p_i = \sum_{i=1}^N x_i p_i + \sum_{i=1}^N y_i p_i = \mathcal{E}(X) + \mathcal{E}(Y)$$

Was sind i und N ?

Beispiel 2 Münzen, Anzahl "Kopf"; erste: $i \quad X \quad p \quad 1 \quad 0 \quad 1/2 \quad \mathcal{E}(X) = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = \frac{1}{2}$

zweite: Dasselbe für Y : $\mathcal{E}(Y) = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = \frac{1}{2}$

Nun die Summe (Anzahl "Zahl") der zwei Münzen $i \ Z \ p \ 1 \ 0 \ 1/4 \ 2 \ 1 \ 1/2$ $\mathcal{E}(Z) = 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} = 1$
 $3 \ 2 \ 1/4$

Gemeinsames Werfen von Münze X und Münze Y ergibt Summe "Anzahl Kopf" Z.

Elementarzerlegung des gemeinsamen Vektors W : $\forall i \in \{1 \dots N\}$ gilt $z_i = x_i + y_i$.

i	X	Y	Z=X+Y	p
1	0	0	0	1/4
2	0	1	1	1/4
3	1	0	1	1/4
4	1	1	2	1/4

4 Kombinationsmöglichkeiten, N=4, alle durchlaufen mit i

$$\begin{aligned}\mathcal{E}(X+Y) &= 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} = 1 \\ \mathcal{E}(X) &= 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} = \frac{1}{2} \\ \mathcal{E}(Y) &= 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} = \frac{1}{2} \\ \mathcal{E}(X) + \mathcal{E}(Y) &= 2 \cdot \frac{1}{2} = 1\end{aligned}$$

X \ Y	0	1	
0	1/4	1/4	1/2
1	1/4	1/4	1/2
	1/2	1/2	1

$$\sum_{k=1}^4 (x_k + y_k) \cdot p_k = \sum_{i=1}^2 \sum_{j=1}^2 (x_i + y_j) p_{ij}$$

$$\begin{aligned}\text{Damit: } &= \sum_{i=1}^2 x_i \sum_{j=1}^2 p_{ij} + \sum_{j=1}^2 y_j \sum_{i=1}^2 p_{ij} \\ &= \sum_{i=1}^2 x_i p_i + \sum_{j=1}^2 y_j p_j\end{aligned}$$

mit $k = 2 \cdot (i-1) + j$

1.6 Anwendung bi- / multivariate Verteilung:

Beispiele

- Verkehrszählung nach Verkehrsmittel und Gesamtaufkommen
- Schadstoffausstoß und Gesamtbelastung
- Vogelzug mehrerer Arten pro Tag und Gesamtanzahl

1.6.1 Addition Poisson-Verteilungen

Sind $X \sim \mathcal{P}(\lambda_1)$ und $Y \sim \mathcal{P}(\lambda_2)$ Poisson-verteilt und voneinander unabhängig, so ist

$$X + Y \sim \mathcal{P}(\lambda_1 + \lambda_2)$$

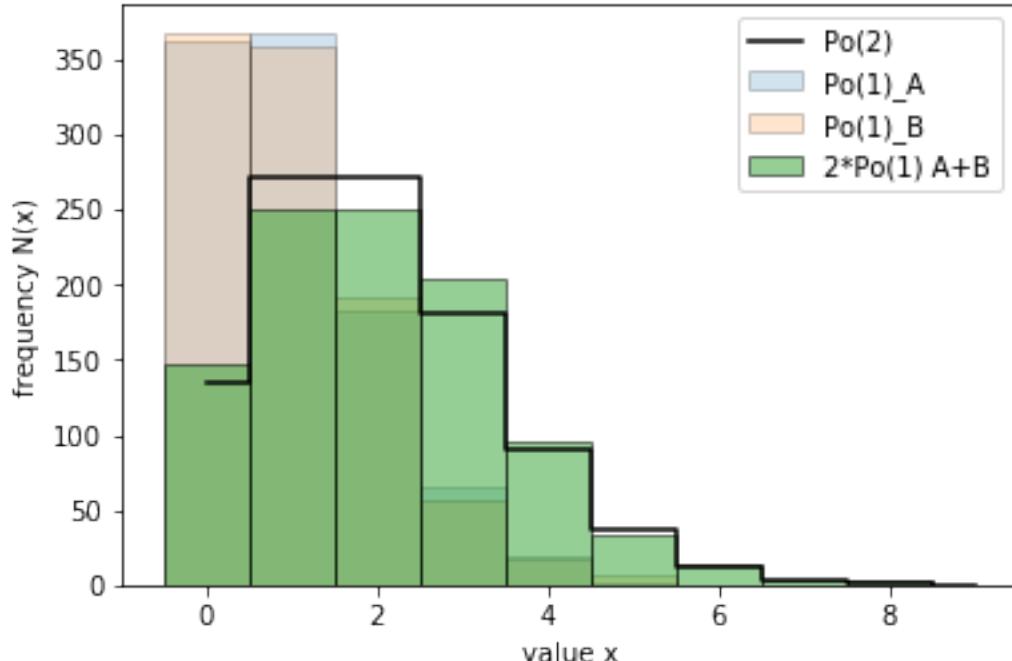
```
In [3]: '''test summation of Poisson distributions
           according to rate lambda
           ...
11 = 1.                                     # lambda of distribution A and B
x = np.arange(0., 21)                         # outcomes
N = 1000                                       # number of samples to draw
np.random.seed(987654)
xa = stats.poisson(11).rvs(size=N)             # outcomes "A" of Poisson(lambda=1)
xb = stats.poisson(11).rvs(size=N)             # outcomes "B" of Poisson(lambda=1)
```

```

xab = xa + xb                               # sum of the two
print('Po(1)_A = ', xa[:30], '...')
print('Po(1)_B = ', xb[:30], '...')
print('sum A+B = ', xab[:30], '...')
# compare the two single distributions, theire sum and the theoretical
ha = plt.hist(xa, bins=np.arange(10), alpha=0.2, label='Po(1)_A',
               align='left', edgecolor='black')
hb = plt.hist(xb, bins=np.arange(10), alpha=0.2, label='Po(1)_B',
               align='left', edgecolor='black')
hab = plt.hist(xab, bins=np.arange(10), alpha=0.5, label='2*Po(1) A+B',
               align='left', edgecolor='black')
po2 = stats.poisson.pmf(x, 2*11) # theoretical distribution with double lambda
plt.step(x[:10], N*po2[:10], 'k--', where='mid', label='Po(2)')
plt.xlabel('value x')
plt.ylabel('frequency N(x)')
plt.legend();

Po(1)_A = [0 1 0 1 0 1 2 0 0 1 2 0 0 3 2 0 1 0 0 2 2 0 0 0 0 0 2 2 0 1 1] ...
Po(1)_B = [1 0 0 1 1 1 0 0 2 1 1 0 0 2 2 0 0 0 0 3 1 0 0 1 0 1 2 0 2 3] ...
sum A+B = [1 1 0 2 1 2 2 0 2 2 3 0 0 5 4 0 1 0 0 5 3 0 0 1 0 3 4 0 3 4] ...

```



```

In [6]: '''compare theoretical Poisson(2) with sum of two Poisson(1) distributions'''
po11 = np.asarray([
    np.asarray([
        y*x for j,y in enumerate(po1) for k,x in enumerate(po1) if j+k==i]).sum()
    for i in range(10)])
for i in range(10):
    print('P1+P1](x={})={:.7f} P2(x={})={:.7f}'.format(i, po11[i], i, po2[i]))

[P1+P1](x=0)=0.1353353 P2(x=0)=0.1353353
[P1+P1](x=1)=0.2706706 P2(x=1)=0.2706706
[P1+P1](x=2)=0.2706706 P2(x=2)=0.2706706
[P1+P1](x=3)=0.1804470 P2(x=3)=0.1804470
[P1+P1](x=4)=0.0902235 P2(x=4)=0.0902235
[P1+P1](x=5)=0.0360894 P2(x=5)=0.0360894

```

[P1+P1] (x=6)=0.0120298	P2(x=6)=0.0120298
[P1+P1] (x=7)=0.0034371	P2(x=7)=0.0034371
[P1+P1] (x=8)=0.0008593	P2(x=8)=0.0008593
[P1+P1] (x=9)=0.0001909	P2(x=9)=0.0001909

1.7 Erwartungswert eines Produkts von Zufallsvariablen

Für zwei **unabhängige** Zufallsvariablen X und Y gilt:

$$\mathcal{E}(X \cdot Y) = \mathcal{E}(X) \cdot \mathcal{E}(Y)$$

Beweis ÜA

Beispiel zwei Münzen

i	j	k	X	Y	X*Y	p=px*py
1	1	1	0	0	0	1/4
1	2	2	0	1	0	1/4
2	1	3	1	0	0	1/4
2	2	4	1	1	1	1/4

$$\begin{aligned}\mathcal{E}(X) &= 0 \cdot \frac{1}{4} \cdot 2 + 1 \cdot \frac{1}{4} \cdot 2 = \frac{1}{2} \\ \mathcal{E}(Y) &= 0 \cdot \frac{1}{4} \cdot 2 + 1 \cdot \frac{1}{4} \cdot 2 = \frac{1}{2} \\ \mathcal{E}(X \cdot Y) &= 0 \cdot \frac{1}{4} \cdot 3 + 1 \cdot \frac{1}{4} = \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2}\end{aligned}$$

2 Varianz(en) mehrdimensionaler Verteilung

$$\text{Var}(W) = \begin{pmatrix} \text{Var}(X) \\ \text{Var}(Y) \end{pmatrix} = \begin{pmatrix} \sum_x (x - \mathcal{E}(X))^2 p_x \\ \sum_y (y - \mathcal{E}(Y))^2 p_y \end{pmatrix}$$

3 Kovarianz

Definition:

$$\text{Cov}(X, Y) := \mathcal{E}((X - \mathcal{E}(X))(Y - \mathcal{E}(Y))) = \sum_x \sum_y (x - \mu_x)(y - \mu_y) \cdot p_{xy}$$

Verschiebungssatz

$$\text{Cov}(X, Y) = \mathcal{E}(X \cdot Y) - \mathcal{E}(X) \cdot \mathcal{E}(Y)$$

Varianz

$$\text{Var}(X) = \text{Cov}(X, X)$$

Beweis: [ÜA]

3.0.1 Bedeutung der Kovarianz für die Varianz einer Summe von Zufallsvariablen:

$$\begin{aligned}\text{Var}(X + Y) &= \mathcal{E}((X + Y - \mathcal{E}(X) - \mathcal{E}(Y))^2) \\ &= \mathcal{E}((X - \mathcal{E}(X))^2 + (Y - \mathcal{E}(Y))^2 + 2(X - \mathcal{E}(X)) \cdot (Y - \mathcal{E}(Y))) \\ &= \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y)\end{aligned}$$

3.0.2 Extremfälle der Kovarianz für die Varianz einer Summe von Zufallsvariablen

vollständige Korrelation $X, Y=X$

$$\begin{aligned}\text{Var}(X + Y) &= \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y) \\ &= \text{Var}(X) + \text{Var}(Y) + 2(\mathcal{E}(XY) - 2\mathcal{E}(X)\mathcal{E}(Y)) \\ &= \text{Var}(X) + \text{Var}(X) + 2(\mathcal{E}(X^2) - \mathcal{E}^2(X)) \\ &= 4\text{Var}(X) = \text{Var}(2X)\end{aligned}$$

vollständige Antikorrelation $X, Y=-X$

$$\begin{aligned}\text{Var}(X + Y) &= \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y) \\ &= \text{Var}(X) + \text{Var}(Y) + 2(\mathcal{E}(XY) - \mathcal{E}(X)\mathcal{E}(Y)) \\ &= \text{Var}(X) + \text{Var}(X) - 2(\mathcal{E}(X^2) - \mathcal{E}^2(X)) \\ &= (2 - 2)\text{Var}(X) = \text{Var}(0X)\end{aligned}$$

4 Korrelationskoeffizient

Definition:

$$\text{Korr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}} = \frac{\text{Cov}(X, Y)}{\sigma_x \sigma_y}$$

Eigenschaften

- $\text{Korr} \in [-1, 1]$
- $\text{Korr} = +1$ wenn $X = Y$
- $\text{Korr} = -1$ wenn $X = -Y$
- $\text{Korr} = 0$ wenn X, Y stochastisch unabhängig
 - (nicht unbedingt umgekehrt)

5 Stochastische Unabhängigkeit

X und Y sind genau dann stochastisch unabhängig, wenn

$$\forall (x, y) \in \Omega : P(X=x \cap Y=y) = P(X=x) \cdot P(Y=y)$$

\Rightarrow

$$\text{Cov}(X, Y) = 0$$

bzw.

$$\mathcal{E}(X \cdot Y) = \mathcal{E}(X) \cdot \mathcal{E}(Y)$$

6 Zusammenfassung multivariate Wahrscheinlichkeitsverteilung

- Eine Wahrscheinlichkeit auf mehrere Zufallsvariable, z.B. $p(x, y)$
- Erwartungswerte
- Varianzen
- Kovarianz
- Korrelation
- Unabhängigkeit

6.1 Mehr als zwei Dimensionen?

- Erwartungswert $\mathcal{E}(X)$
- Varianz $\text{Var}(X)$
- Matrix gegenseitiger Abhangigkeiten
 - Wie bei empirischer Statistik “2D-Scatterplots”
- Kovarianzmatrix
 - Diagonale: Varianzen
 - Seiten: Kovarianzen

7 Ausblick

Kontinuierliche Verteilungen

8 Fragen?

051_Folien

November 23, 2018

```
In [1]: import numpy as np                      # mathematical methods
         from scipy import stats                  # statistical methods
         from matplotlib import pyplot as plt     # plotting methods
         %matplotlib inline
```

1 Wahrscheinlichkeitstheorie

- 1.1 Zufallsvariable und Wahrscheinlichkeitsraum
- 1.2 Erwartungswert und Varianz
- 1.3 Diskrete Zufallsvariablen und Wahrscheinlichkeitsverteilungen
- 1.4 Mehrdimensionale Verteilungen
- 1.5 Kontinuierliche Zufallsvariable und Wahrscheinlichkeitsverteilungen
- 1.6 Wahrscheinlichkeitsverteilung kontinuierlicher Zufallsvariable

Unendlich viele dichte Ereignisse (meist $x \in \mathbb{R}$) möglich.

Beispiele

- Rauschspannung
- Aktienkurse
- Chemikalienkonzentrationen

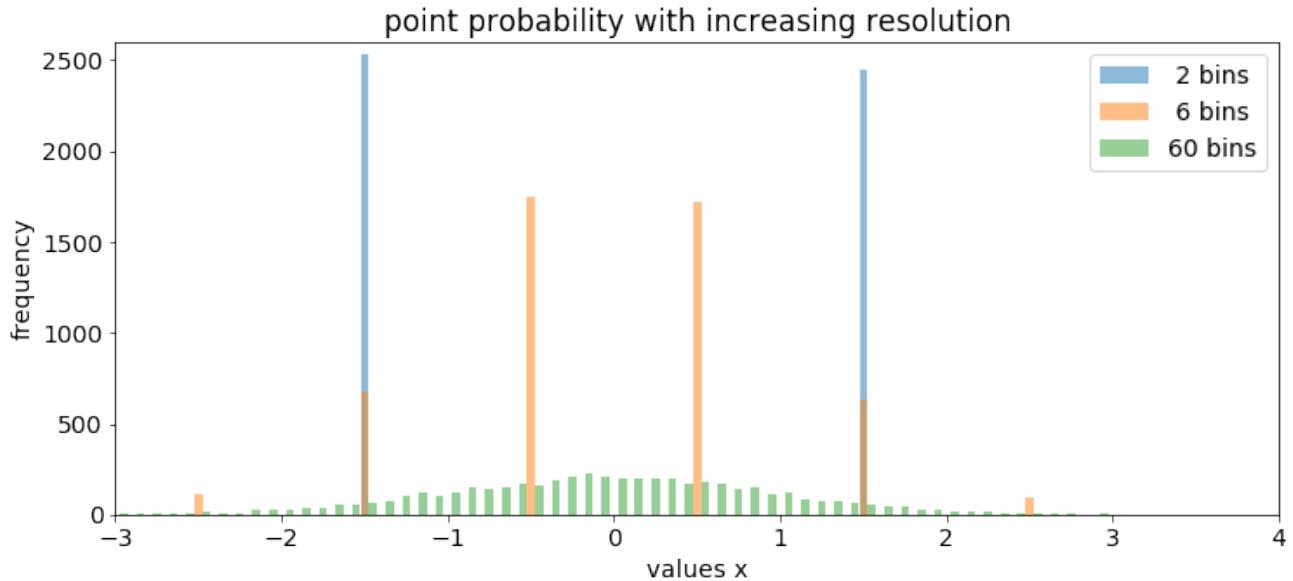
1.6.1 Problem: Punktwahrscheinlichkeit

$$P(x = a) \rightarrow 0$$

```
In [3]: '''discrete --> continuous distribution: single probability'''
f = plt.figure(figsize=(12, 5))
np.random.seed(9876543)
x = stats.norm(0., 1.).rvs(size=5000)      # draw random numbers

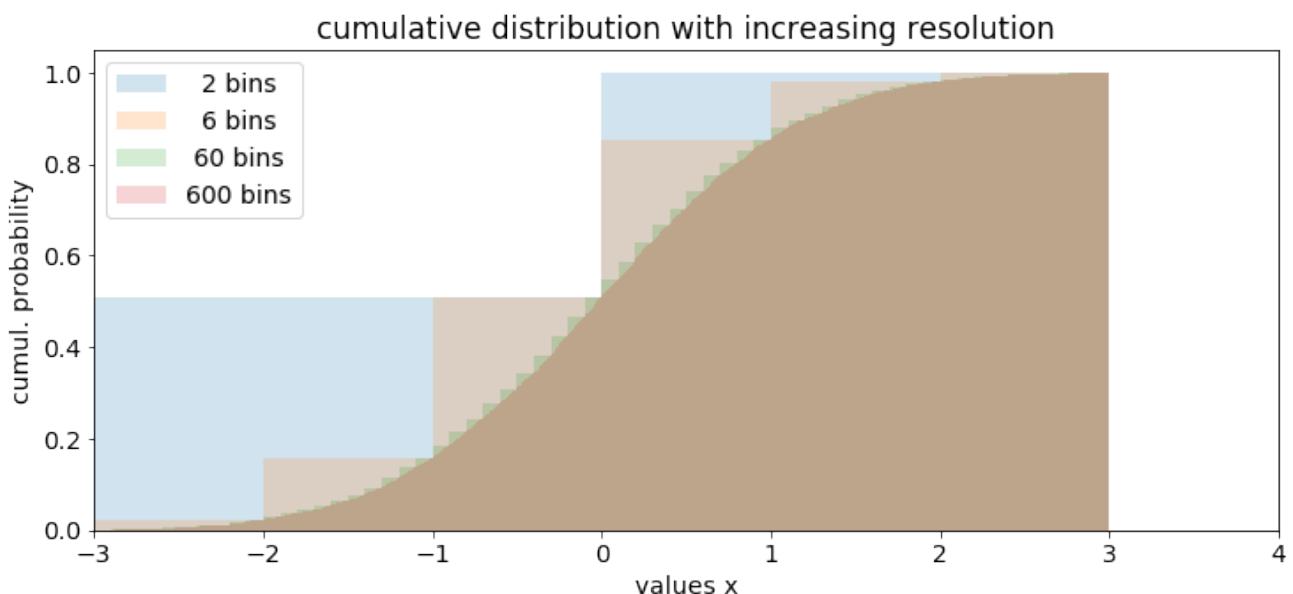
for n in (2, 6, 60):                      # discretize size to n bins
    h = plt.hist(x, bins=np.linspace(-3., 3., n+1), align='mid',
                  rwidth=0.008*n, alpha=0.5, label='{:2d} bins'.format(n))
    print('discrete {:3d} has center probabilities {}'.format(n,
                                                               .format(n, h[0][max(0, int(n/2-3)):min(n, int(n/2+3))])))
plt.axis((-3., 4., 0, 2600 ))
plt.title('point probability with increasing resolution')
plt.xlabel('values x')
plt.ylabel('frequency')
plt.legend();

discrete  2 has center probabilities [2537. 2443.]
discrete  6 has center probabilities [ 117.  675. 1745. 1716.  632.   95.]
discrete 60 has center probabilities [205. 226. 207. 198. 195. 201.]
```



```
In [4]: '''discrete --> continuous distribution: cumulative'''
f = plt.figure(figsize=(12, 5))
for n in (2, 6, 60, 600):                      # discretisize to n bins
    h = plt.hist(x, bins=np.linspace(-3., 3., n+1), density=True,
                  cumulative=True, alpha=0.2, label='{:3d} bins'.format(n))
    print('discrete {:4d} bins have center probabilities {}'.format(
        n, h[0][max(0, int(n/2-2)):min(n, int(n/2+2))]))
plt.axis((-3., 4., 0, 1.05 ))
plt.title('cumulative distribution with increasing resolution')
plt.xlabel('values x')
plt.ylabel('cumul. probability')
plt.legend();

discrete    2 bins have center probabilities [0.50943775 1.      ]
discrete    6 bins have center probabilities [0.15903614 0.50943775 0.85401606 0.98092369]
discrete   60 bins have center probabilities [0.46787149 0.50943775 0.54919679 0.58835341]
discrete  600 bins have center probabilities [0.50421687 0.50943775 0.5126506  0.51767068]
```



Gemeinsamkeit Steigung:

$$\text{diskret : } \frac{\Delta F}{\Delta x} = m$$

$$\text{kontinuierlich : } \frac{dF}{dx} = f$$

1.7 Wahrscheinlichkeitsdichte $f(x)$

$$f(x) \geq 0$$

$$P(a \leq x \leq b) = \int_a^b f(x) dx \leq 1$$

Möglich:

$$f(x) \not\leq 1$$

Normierung

$$\int_{x=-\infty}^{\infty} f(x) dx = 1$$

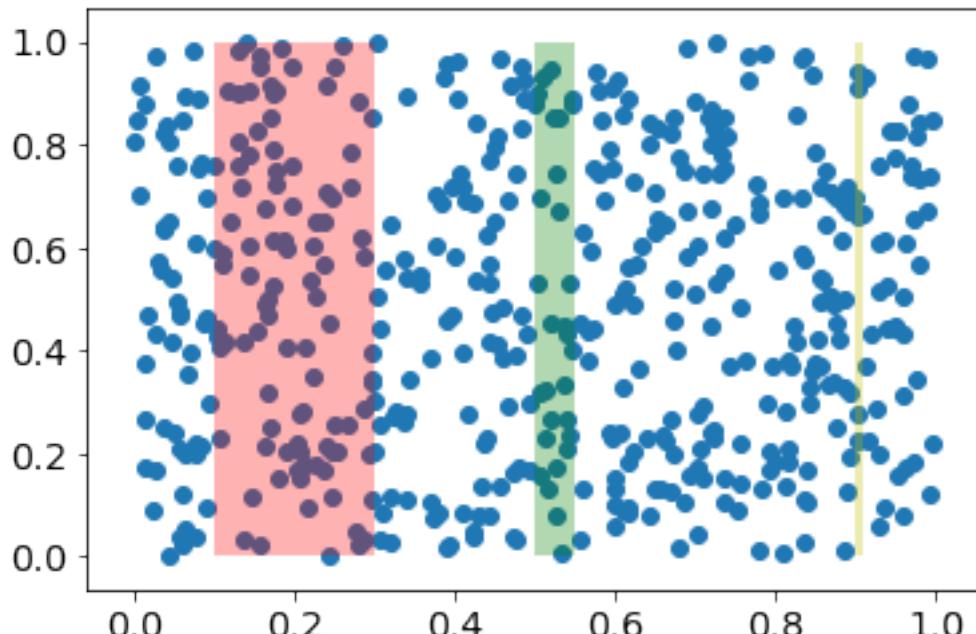
In [6]: *'raindrops are falling - show density'*

```
np.random.seed(98765432)
x = np.random.rand(500) # 500 raindrops x coordinate 0..1
y = np.random.rand(500) #           y coordinate 0..1
plt.scatter(x,y) # show the raindrops
# define three stripes (location, width, color)
s = ((.1, .2, 'r'), (.5, .05, 'g'), (.9, .01, 'y'))
for si in s: # choose one of above
    a, w, col = si # separate location a and width w
    # barh(y-bottom, x-width, y-height, x-left)
    plt.barh(.5, w, 1., a, color=col, alpha=.3)
    # select x between stripe borders
    n = x[np.logical_and(a < x, x <= a+w)].shape[0]
    print('in width={:.2f} are {:.3d} drops or {:.1f}%, density={:.1f}'.
          format(w, n, n/500*100, n/w/500))
```

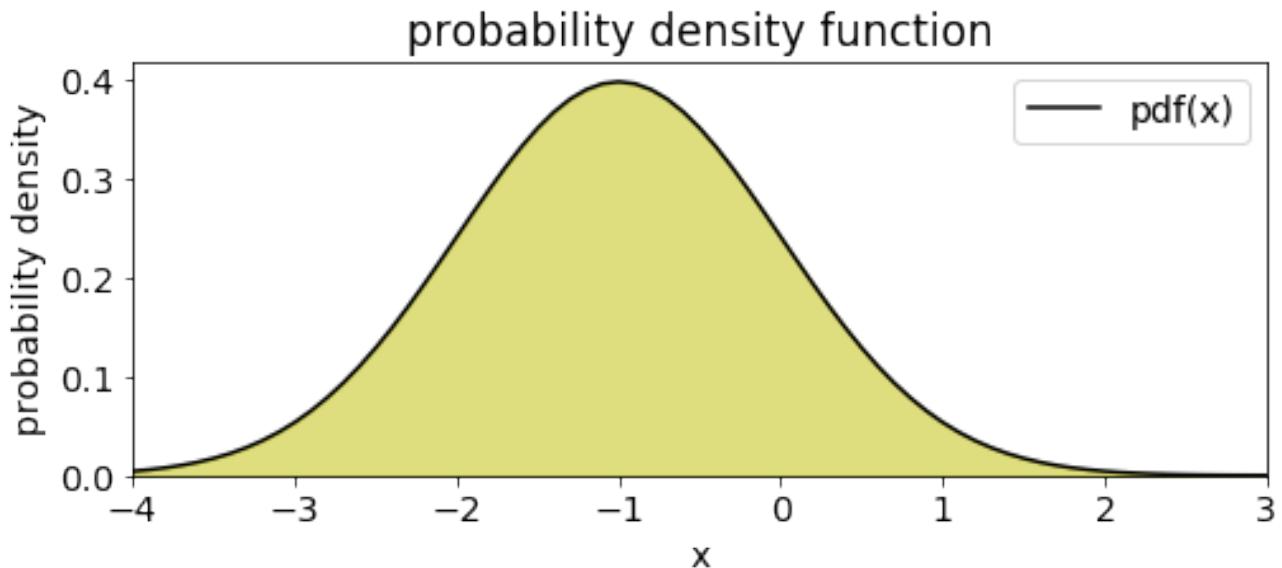
in width=0.20 are 100 drops or 20.0%, density=1.0

in width=0.05 are 29 drops or 5.8%, density=1.2

in width=0.01 are 6 drops or 1.2%, density=1.2



```
In [6]: '''probability density function of x'''
xgrid = np.linspace(-4, 3, 70+1) # just some x values
distrib = stats.norm(-1., 1)      # define a probability distribution & freeze this
f = plt.figure(figsize=(8, 3))
plt.title('probability density function')
plt.plot(x, distrib.pdf(x), 'k-', label='pdf(x)')
# make area under curve yellow
plt.fill_between(x, 0, distrib.pdf(x), color='y', alpha=0.5)
plt.axis((-4., 3., 0, 0.42))
plt.xlabel('x')
plt.ylabel('probability density')
plt.legend();
```

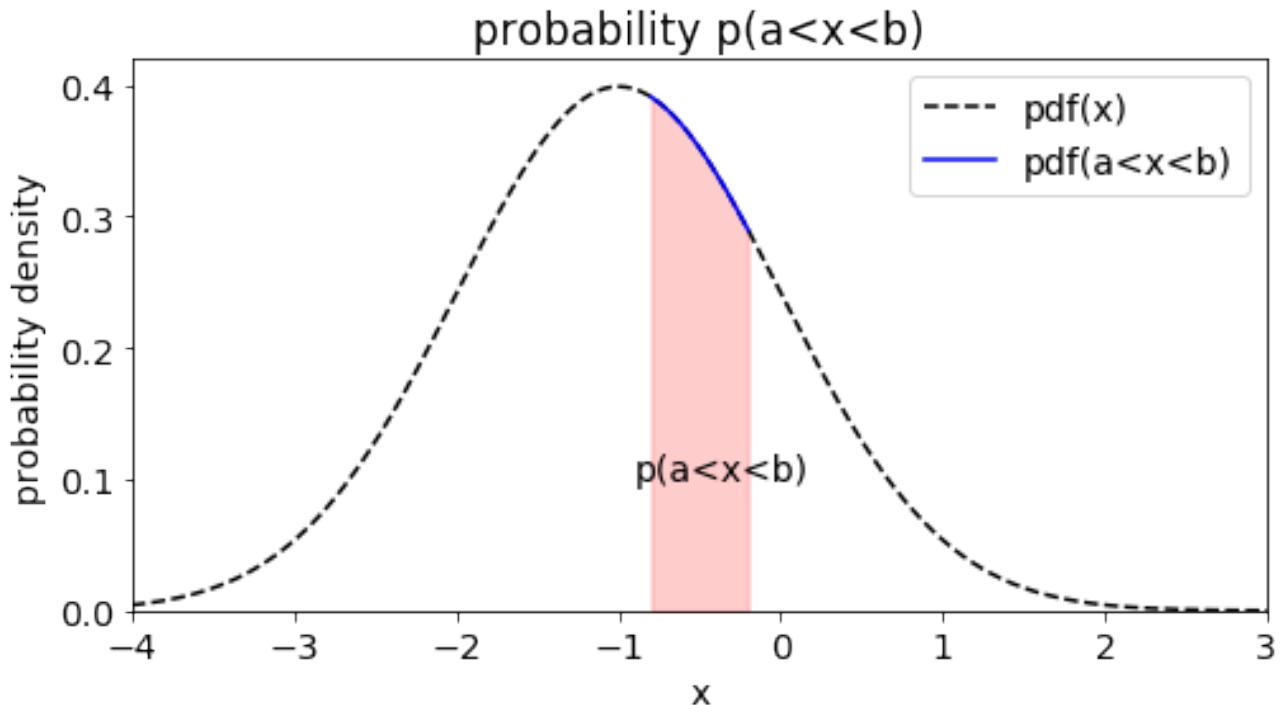


```
In [7]: '''probability of x between a and b'''
x = np.linspace(-6, 4, 1001)          # dense enough x values for smooth graphics
a = -0.8                                # chose lower
b = -0.2                                # and upper boundary of ROI

f = plt.figure(figsize=(8, 4))
plt.title('probability p(a<x<b)')
# complete distribution, frozen from above
plt.plot(x, distrib.pdf(x), 'k--', label='pdf(x)')
xab = np.linspace(a, b, 11) # select ROI, make smooth enough
plt.plot(xab, distrib.pdf(xab), 'b-', label='pdf(a<x<b)') # highlight ROI in blue
plt.fill_between(xab, 0, distrib.pdf(xab), color='r', alpha=0.2) # and make area visible
plt.text(-0.9, 0.1, 'p(a<x<b)')
plt.axis((-4., 3., 0, 0.42))
plt.xlabel('x')
plt.ylabel('probability density')
plt.legend();

ptotal = distrib.pdf(x).mean()*(4.-(-6.))           # normalized?
xab = np.linspace(a, b, 1001)                         # dense enough x values
p = distrib.pdf(xab).mean()*(b-a)                    # pseudo integral /
print('probability of x between {:.2f} and {:.2f} is {:.3f} (of norm={:.4f})'
      .format(a, b, p, ptotal))
```

probability of x between -0.80 and -0.20 is 0.209 (of norm=0.9990)



1.8 (Wahrscheinlichkeits-) Verteilungsfunktion

$$F(x) = \int_{x'=-\infty}^x f(x') \, dx'$$

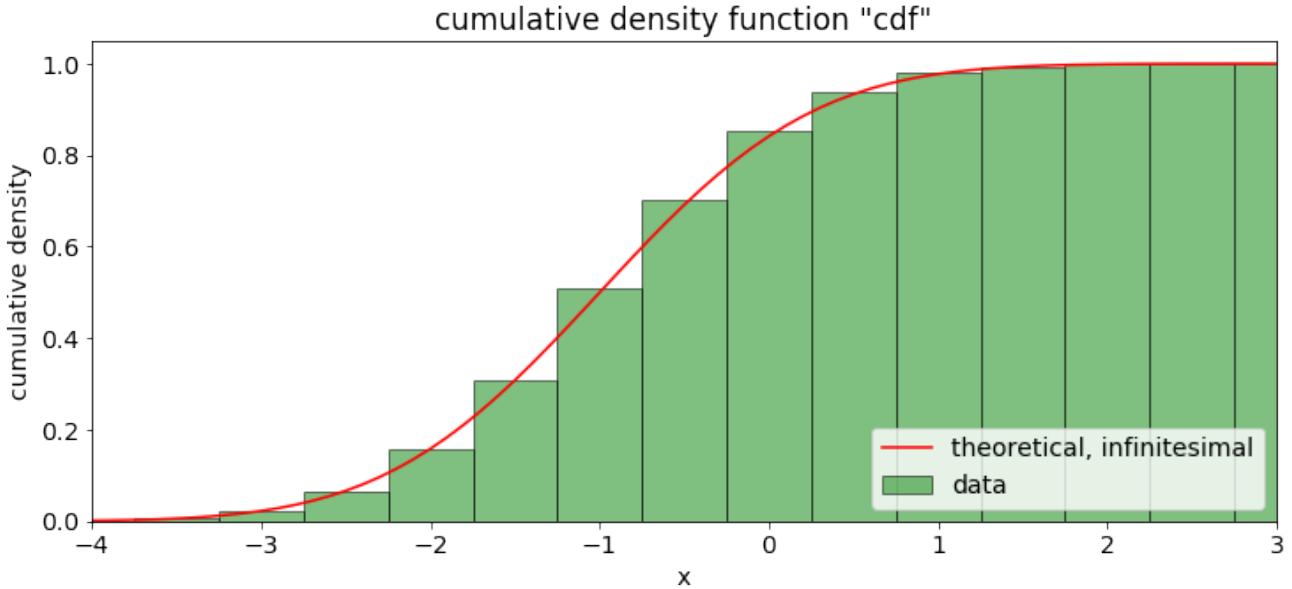
Vergleiche mit diskreter Verteilungsfunktion:

$$F(x) = \sum_{x_i <= x} p(x_i)$$

```
In [8]: ''' N=10 bins as discrete example above
      x from above: origin was standard normal distribution'''

np.random.seed(9876543)
x = distrib.rvs(size=5000)           # draw random numbers from chosen norm
f = plt.figure(figsize=(12, 5))
plt.hist(x, bins=np.linspace(-4., 3., 14+1), label='data',
         color='green', cumulative=True, alpha=0.5, density=True, align='right',
         edgecolor='black', linewidth=1.)

# cdf
xi = np.linspace(-4., 3., 70+1)      # make an x-axis with finer resolution
y = distrib.cdf(xi)                  # use the cdf-fct (with data x) for xi
plt.plot(xi, y, 'r-', label='theoretical, infinitesimal')
plt.axis((-4., 3., 0, 1.05))
plt.xlabel('x')
plt.ylabel('cumulative density')
plt.title('cumulative density function "cdf"')
plt.legend(loc='lower right');
```



1.9 Eigenschaften der Verteilungsfunktion

für integrierbare Dichten $f(x)$

$$F(x) = \int_{-\infty}^x f(x') dx'$$

- $F(x)$ ist monoton in x
- $F(-\infty) = 0$
- $F(\infty) = 1$
- $P(a \leq x \leq b) = \int_a^b f(x) dx = F(b) - F(a)$
- $P(X \geq c) = 1 - F(c)$
- Punktwahrscheinlichkeit: wenn $a = x = b$ dann $P(X=b) = F(b) - F(b) = 0$
wichtig!

1.9.1 Anschaulich

```
In [10]: '''cumulated probability of x up to a'''
r0 = -4.
r1 = 4.
x = np.linspace(r0, r1, 801)      # dense enough a values
x1 = -0.8                         # choose an x: x1
p1 = distrib.cdf(x1)              # calculate x1' probability
print('cumulated probability up to x1={:.2f} is {:.2f}'.format(x1, p1))

f = plt.figure(figsize=(8, 8))
f.add_subplot(211)                  # 2 subplots, this 1st column, 1st row
plt.title('probability density function')
plt.plot(x, distrib.pdf(x), 'k--', label='pdf(x)')
xa = np.linspace(r0, x1, 81)       # x values up to x1
# plot blue pdf-line and fill space below (from 0 upwards to pdf)
plt.plot(xa, distrib.pdf(xa), 'b-', label='pdf(x<x1)')
plt.fill_between(xa, 0, distrib.pdf(xa), color='r', alpha=0.2)
plt.ylim(0, 0.42)                  # include max p ~0.4
plt.legend()

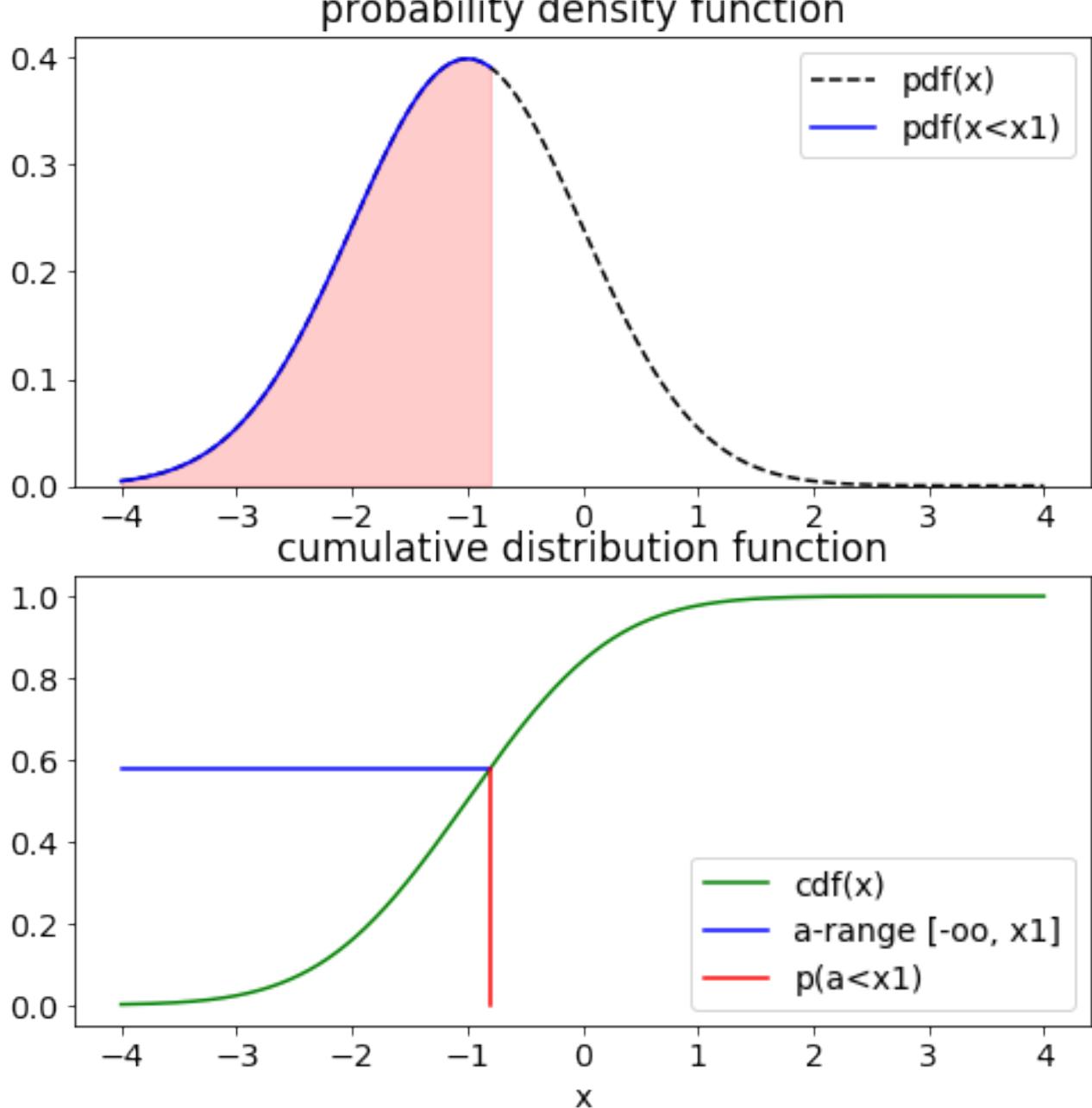
f.add_subplot(212)                  # from the 2 subplots: this 1st columns, 2nd row
plt.title('cumulative distribution function')
```

```

plt.xlabel('x')
x = np.linspace(r0, r1, 801)      # dense enough x values
# cdf of complete distribution over all x
plt.plot(x, distrib.cdf(x), 'g-', label='cdf(x)')
cc = distrib.cdf(x1)              # cdf(x=x1)
# blue line for range of x from left to x1 in height cc
plt.plot([r0, x1], 2*[cc], 'b-', label='a-range [-oo, x1]')
# red line for p=0 to cc at x=x1
plt.plot(2*[x1], [0, cc], 'r-', label='p(a<x1)')
plt.legend(loc='lower right');

cumulated probability up to x1=-0.80 is 0.58

```



In [11]: '''probability of x between a and b'''

```
x = np.linspace(-4, 4, 801)      # dense enough x values
```

```

a = -0.8                      # choose lower
b = -0.2                      # and upper boundary
p = distrib.cdf(b)-distrib.cdf(a) # calculate probability
print('probability of x between {:.2f} and {:.2f} is {:.3f}'
      .format(a, b, p))

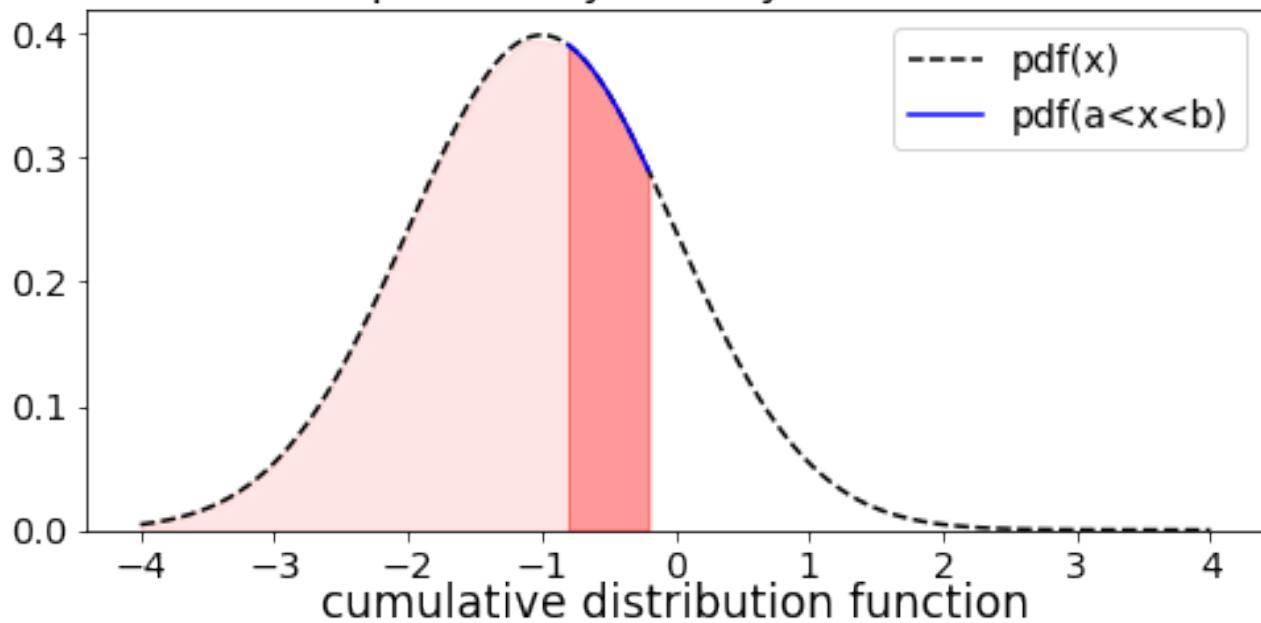
f = plt.figure(figsize=(8, 8))
f.add_subplot(211)              # 2 subplots, this 1st column, 1st row
plt.title('probability density function')
plt.plot(x, distrib.pdf(x), 'k--', label='pdf(x)')
xab = np.linspace(-4, a, 11)    # x values from "-infty" to a
plt.fill_between(xab, 0, distrib.pdf(xab), color='r', alpha=0.1)
xab = np.linspace(a, b, 11)     # x values from a to b
# plot blue pdf-line from a to b and fill space below (from 0 upwards to pdf)
plt.plot(xab, distrib.pdf(xab), 'b-', label='pdf(a<x<b)')
plt.fill_between(xab, 0, distrib.pdf(xab), color='r', alpha=0.4)
plt.ylim(0, 0.42)
plt.legend()

f.add_subplot(212)
plt.title('cumulative distribution function')
plt.xlabel('x')
# green cdf of complete distribution over all x
plt.plot(x, distrib.cdf(x), 'g-', label='cdf(x)')
cc = distrib.cdf([a, b])        # cdf of left a and right b
# blue line for range of x from a to b in height cdf(a)
plt.plot([a, b], 2*[cc[0]], 'b-', label='x-range [a, b]')
# red line for p=cdf(a) to cdf(b) at x=b
plt.plot(2*[b], cc, 'r-', label='p(a<x<b)')
plt.plot(2*[b], [0, cc[0]], 'r:', label='p( x<a)')
plt.legend(loc='lower right');

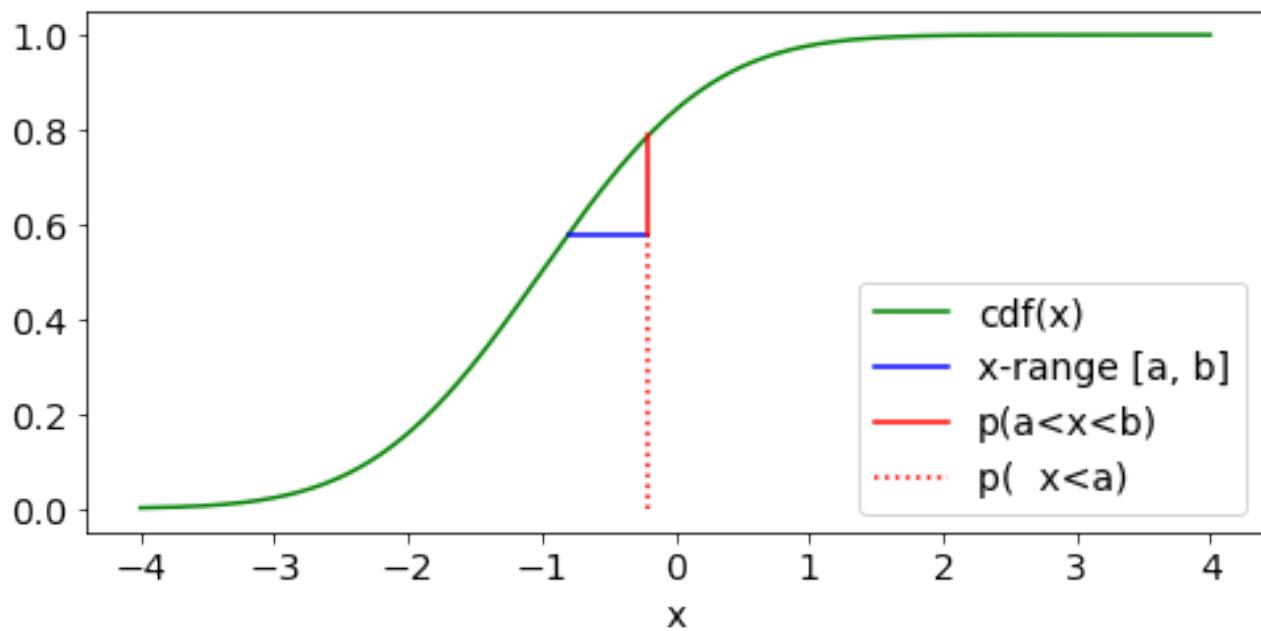
```

probability of x between -0.80 and -0.20 is 0.209

probability density function



cumulative distribution function



1.9.2 Vergleich mit diskreten Verteilungen

Variable	diskret	kontinuierlich
Wert x	$x_i \quad i \in \mathbb{N}$	$x \in \mathbb{R}$
Wahrscheinlichkeit p	$p(X=x_i) = p_i$	$p(a \leq x \leq b) = \int_a^b f(x)dx$
.	pmf()	pdf()
Verteilungsfunktion F	Schritte $\Delta F = p_x$	kontinuierlich $dF = f dx$
.	cdf()	cdf()

1.10 Definition Erwartungswert

Sei X eine Zufallsvariable auf \mathbb{R} mit Wahrscheinlichkeitsdichte $f(x)$, dann ist der Erwartungswert von X

$$\mathcal{E}(X) = \mu = \int_{-\infty}^{\infty} f(x) \cdot x \, dx$$

Vergleich diskreter Erwartungswert

$$\mathcal{E}(X) = \mu = \sum_{i=1}^N p(x_i) \cdot x_i$$

```
In [12]: '''meaning of expectation value of x as weighted integral'''
r0, r1 = (-2.0, 2.0)                      # borders
ndistrib = stats.norm(-0.2, 0.3)            # define and freeze a (normal) probability distribution
x = np.linspace(r0, r1, 40*round(r1-r0)+1)  # dense enough values
px = ndistrib.pdf(x)                        # get x's probability

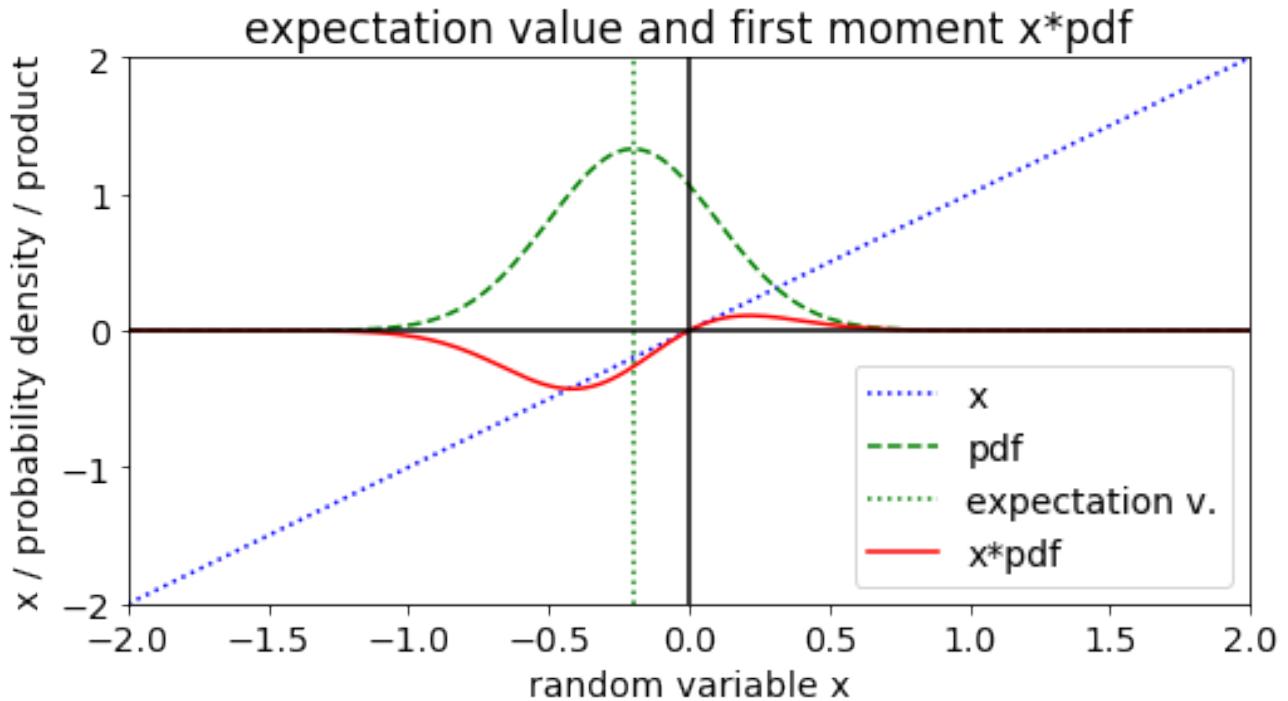
f = plt.figure(figsize=(8, 4))                # space for graphics
plt.title('expectation value and first moment x*pdf')
plt.xlabel('random variable x')
plt.ylabel('x / probability density / product')
plt.plot(x, x,    'b:', label='x')           # x (just repeated as y value)
plt.plot(x, px,   'g--', label='pdf')         # probability density of x - as a weight

# calculate and show expectation value
print('expectation value of x under pdf is {:.3f}'.format(ndistrib.expect()))
plt.plot(2*[ndistrib.expect()], [-2, 2], 'g:', label='expectation v.')

# expectation value is integral of local x*px function
# meaning: x weighted with its probability density
# integral is then area under red, here negative
# approximation: sum over the 160 small rectangles of size dx
dx = (r1-r0)/(len(x)-1)                      # length of approximation rectangle
expectation_approx = dx*(x*px).sum()          # product is point-wise
print('approximated expectation value is {:.3f}'.format(expectation_approx))

plt.plot(x, x*px, 'r-', label='x*pdf')        # local product x with pdf
plt.plot([r0, r1], 2*[0], 'k-')               # coordinate system
plt.plot(2*[0], [r0, r1], 'k-')               # set borders
plt.axis((r0, r1, r0, r1))                   # who is who

expectation value of x under pdf is -0.200
approximated expectation value is -0.200
```



1.11 Definition Varianz

Sei X eine Zufallsvariable auf \mathbb{R} mit Wahrscheinlichkeitsdichte $f(X)$ und Erwartungswert μ , dann

$$\begin{aligned}\text{Var}(X) &= \sigma^2 = \int_{-\infty}^{\infty} (x - \mu)^2 \cdot f(x) dx \\ &= E((x - \mu)^2)\end{aligned}$$

1.11.1 Verschiebungssatz:

$$\text{Var}(X) = E(x^2) - (E(x))^2$$

Beweis: [ÜA]

1.12 Rechenregeln

Sei $g(x)$ eine reelle Funktion. Dann gilt für $Y = g(X)$

$$E(Y) = E(g(X)) = \int_{-\infty}^{\infty} f(x) \cdot g(x) dx$$

1.12.1 lineare Transformation Erwartungswert

Für $Y = aX + b$ ergibt sich

$$E(Y) = E(aX + b) = aE(x) + b$$

Beweis:

$$\begin{aligned}\int_{-\infty}^{\infty} (ax + b)f(x) dx &= \int_{-\infty}^{\infty} axf(x) dx + \int_{-\infty}^{\infty} bf(x) dx \\ &= aE(X) + b \cdot 1\end{aligned}$$

1.12.2 lineare Transformation Varianz

Für die Varianz unter der linearen Transformation $Y = aX + b$ ergibt sich

$$\mathcal{V}\text{-}\nabla(Y) = \mathcal{V}\text{-}\nabla(aX + b) = a^2 \cdot \mathcal{V}\text{-}\nabla(X)$$

Beweis: Wie bei der (\rightarrow) diskreten Definition

$$\begin{aligned} \mathcal{V}\text{-}\nabla(aX + b) &= \mathcal{E}([aX + b - \mathcal{E}(aX + b)]^2) \\ &= \mathcal{E}([aX + b - a\mathcal{E}(X) - b]^2) \\ &= a^2\mathcal{E}([X - \mathcal{E}(X)]^2) \\ &= a^2\mathcal{V}\text{-}\nabla(X) \end{aligned}$$

1.13 Anwendung der linearen Transformation: Standardisieren

Mittels der speziellen linearen Transformation $Z = \frac{1}{\sigma}(X - \mu)$ ergibt sich

$$\mathcal{E}(Z) = 0$$

$$\text{Var}(Z) = 1$$

Bitte merken für später

1.14 Definition Schiefe

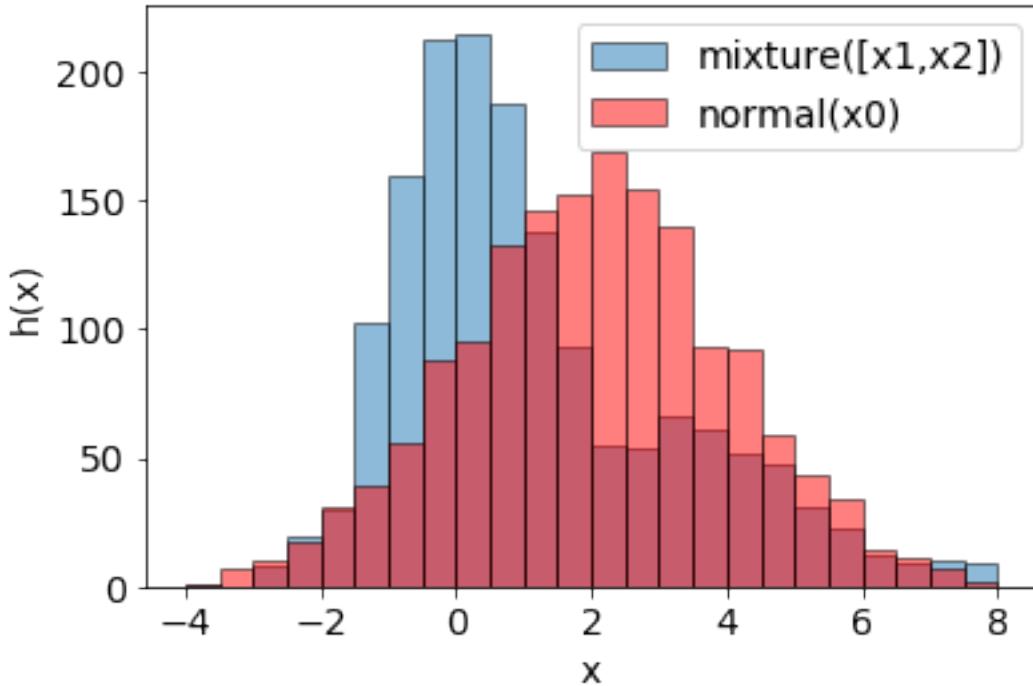
$$\begin{aligned} \text{Schiefe}(X) &= \frac{1}{\sigma^3} \int_{-\infty}^{\infty} (x - \mu)^3 \cdot f(x) dx \\ &= \mathcal{E}(z^3) \end{aligned}$$

- = 0 symmetrische Verteilung
- > 0 linkssteile Verteilung
- < 0 rechtssteile Verteilung

In [13]: *'''Show skewness of distributions'''*

```
np.random.seed(234567)
x0 = stats.norm(2, 2).rvs(size=1600)          # to compare: normal distribution
x1 = stats.norm(0, 1).rvs(size=1000)          # first standard normal distribution
x2 = stats.norm(3, 2).rvs(size=600)           # second normal: shifted and broadened to "right"
bins=np.linspace(-4, 8, 25)
x3 = np.concatenate((x1, x2))                 # mixture of x1 and x2
plt.hist(x3, alpha=.5, bins=bins, label='mixture([x1,x2])', edgecolor='black')
plt.hist(x0, alpha=.5, bins=bins, label='normal(x0)', color='r', edgecolor='black')
print('skewness(Normal distribution) = {:.3f}'.format(stats.norm(2, 2).stats(moments = 's')))
print('skewness(Normal data x0)      = {:.3f}'.format(stats.skew(x0)))
print('skewness(Mix data[x1,x2])    = {:.3f}'.format(stats.skew(x3)))
plt.xlabel('x')
plt.ylabel('h(x)')
plt.legend();

skewness(Normal x0)    = 0.009
skewness(Mix [x1,x2]) = 0.926
```



1.15 Definition Wölbung, Exzeß, Kurtosis

Mit dem *vierten Moment* $m_4 = \int_{-\infty}^{\infty} (x - \mu)^4 \cdot f(x) dx$ ist

$$\text{Kurtosis}(X) = \frac{m_4}{(\sigma^2)^2} - 3$$

Mit der Verschiebung um 3 ist gewährleistet, daß die Normalverteilung die Kurtosis 0 hat.

- $\$ = 0$ wie Normalverteilung \$
- $\$ > 0$ spitzer, langschwänziger\$
- $\$ < 0$ stumpfer, tailliert\$

In [17]: *'''Show kurtosis of distributions'''*

```

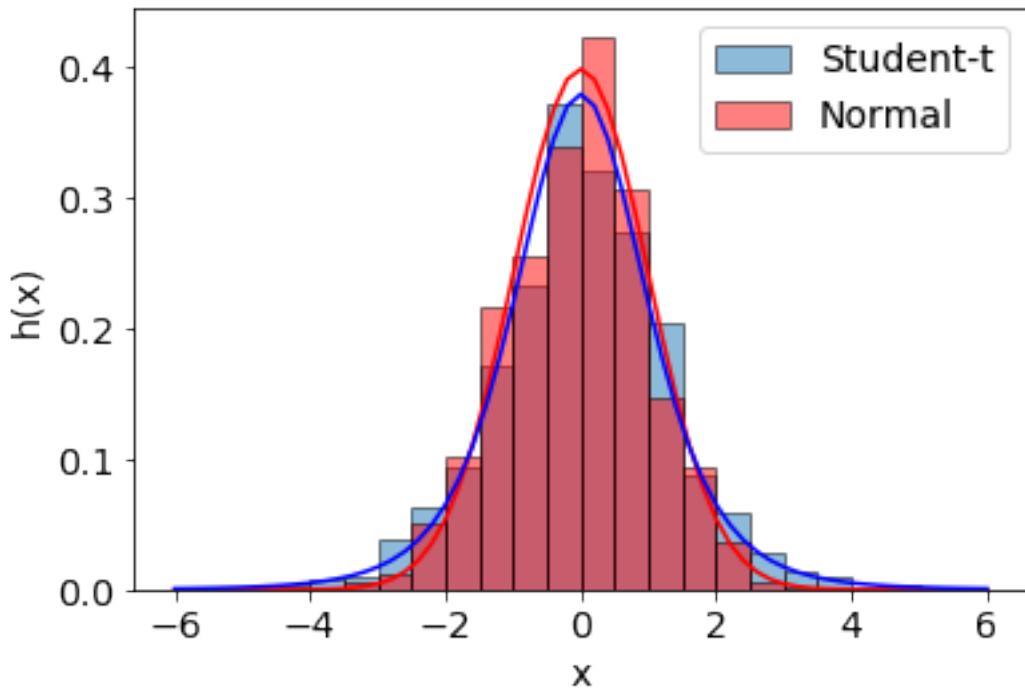
np.random.seed(123456)
xgrid = np.linspace(-6, 6, 60+1)
distrib1 = stats.norm(0, 1)
distrib2 = stats.t(5)
x1 = distrib1.rvs(size=1000)
x2 = distrib2.rvs(size=1000)
bins=np.linspace(-6, 6, 25)
plt.hist(x2, alpha=.5, bins=bins, density=True, label='Student-t', edgecolor='black')
plt.hist(x1, alpha=.5, bins=bins, density=True, label='Normal', color='r', edgecolor='black')
plt.plot(xgrid, distrib1.pdf(xgrid), 'r-')
plt.plot(xgrid, distrib2.pdf(xgrid), 'b-')
print('skewness(Normal)      = {:.6f}'.format(distrib1.stats(moments='k')))
print('kurtosis(Normal data) = {:.6f}'.format(stats.kurtosis(x1)))
print('kurtosis(Student-t)    = {:.6f}'.format(distrib2.stats(moments='k')))
print('kurtosis(Student-t data) = {:.6f}'.format(stats.kurtosis(x2)))

plt.xlabel('x')
plt.ylabel('h(x)')
plt.legend();

skewness(Normal)      = 0.000
kurtosis(Normal data) = -0.040

```

```
kurtosis(Student-t)      =  6.000
kurtosis(Student-t data) =  6.449
```



1.16 Ausblick

- verschiedene wichtige kontinuierliche Verteilungen
- Woher kommt die Wahrscheinlichkeitsdichte $f(x)$ / die Verteilungsfunktion $F(x)$?

2 Zusammenfassung kontinuierliche Zufallsvariablen

- Zufallsexperiment
- Zufallsvariable
 - $X : x \in \mathbb{R}$
- Wahrscheinlichkeitsdichte $f(x)$
 - Grenzwert zur objektiven Häufigkeitsverteilung $\frac{dp}{dx} = \frac{\Delta h}{\Delta x}$
 - subjektive (Theorie, Interpretation)
 - Normierung $[0, 1]$
- Wahrscheinlichkeitsdichte-Verteilungsfunktion
 - $F(x) = \int_{-\infty}^x f(x') dx'$
- Wahrscheinlichkeit
 - $p(a \leq x \leq b) = \int_a^b f(x) dx = F(b) - F(a)$
- Kennzahlen
 - Erwartungswert $E(X) = \mu = \int_{-\infty}^{\infty} f(x) \cdot x dx$
 - Varianz $\text{Var}(X) = \sigma^2 = \int_{-\infty}^{\infty} (x - \mu)^2 \cdot f(x) dx$
 - Schiefe und Kurtosis

3 Zusammenfassung Python

Statsitik-Bibliothek `scipy.stats` enthält kontinuierliche Verteilungen

Funktionen und Methoden

```
.expect()      # Erwartungswert  
.pdf(x)       # WahrscheinlichkeitsdichteVerteilung "probability density function"  
.cdf(x)       # Verteilungsfunktion "cumulative density function"  
.rvs()        # Zufallsergebnis "random variables"  
               # (optional Anzahl der Werte)- Python: `scipy.stats`  
.mean()  
.var()  
.std()  
.kurtosis()  
...
```

```
In [15]: from scipy import stats
```

```
distrib = stats.norm(2, 3)    # "freeze" a normal distribution around mu=2 with sigma=3  
print('mean =          {}'.format(distrib.mean()))  
print('variance =       {}'.format(distrib.var()))  
print('standard deviation = {}'.format(distrib.std()))  
print('norm of p =       {}'.format(distrib.moment(0)))  
m, v, s, k = distrib.stats(moments = 'mvsk')  
print('expectation value =  {}'.format(m))  
print('variance =         {}'.format(v))  
print('skew =             {}'.format(s))  
print('kurtosis =         {}'.format(k))  
  
mean =           2.0  
variance =        9.0  
standard deviation = 3.0  
norm of p =        1.0  
expectation value = 2.0  
variance =        9.0  
skew =            0.0  
kurtosis =         0.0
```

4 Fragen?

052_Folien

November 23, 2018

```
In [1]: import numpy as np                      # mathematical methods
         from scipy import stats                  # statistical methods
         from matplotlib import pyplot as plt     # plotting methods
         %matplotlib inline
```

1 Wahrscheinlichkeitstheorie

1.0.1 Zufallsvariable und Wahrscheinlichkeitsraum

1.0.2 Diskrete Zufallsvariablen und Wahrscheinlichkeitsverteilungen

1.0.3 Kontinuierliche Zufallsvariable und Wahrscheinlichkeitsverteilungen

Wahrscheinlichkeitsdichte

- Unendlich viele dichte Ereignisse ($x \in \mathbb{R}$)

kontinuierliche Verteilungen

- Gleich- / Rechteckverteilung
- Normalverteilung
- Exponentialverteilung
- besondere Verteilungen

1.0.4 Wiederholung: diskrete binomiale Wahrscheinlichkeitsverteilungen

- Bernoulli-Experiment
- mehrere unabhängige identische Wiederholungen *i.i.d*

Binomialverteilung

- Anzahl des Eintretens

Geometrische Verteilung

- Anzahl der Versuche bis zum Eintreten

Poissonverteilung

- Anzahl seltener Ereignisse im Intervall

1.0.5 Gemeinsamkeit: Anzahl $\in \mathbb{N}$

2 Multinomiale Verteilung

Erweiterung des Bernoulli-Zufall-Experiments auf mehrere mögliche Ergebnisse (Elementarereignisse):

$$\Omega = \{a_i\} \quad i > 2$$

Experiment: Ein Objekt wird daraus gezogen/beobachtet/gemessen/...

Beispiele

- Roulette
- Objekte bei Verkehrszählung
- Eigenschaften bei Populationen
- Parteien zur Wahl

```
np.random.multinomial?  
multinomial(n, pvals, size=None)  
The multinomial distribution is a multivariate generalisation of the binomial distribution.
```

In [3]: *'''multinomial example: a single die'''*

```
N = 6  
omega = 1+np.arange(6)  
print('x: ', omega)  
ps = N*[1/N]  
x = np.random.multinomial(n=1, pvals=ps)  
print('f: ', x)  
print('we obtained a {}'.format(omega[x==1]))  
  
x: [1 2 3 4 5 6]  
f: [0 0 1 0 0 0]  
we obtained a [3]
```

In [4]: *'''multinomial with two dice'''*

```
x2 = np.random.multinomial(n=2, pvals=ps) # a combined experiment: throw two dice at once  
print(omega)  
print(x2)  
pasch = len(x2[x2==2]) > 0 # 2 of same kind? vector has (one) element: a 2  
if pasch:  
    print('we got a {}-pasch'.format(omega[x2==2])) # show the only place of the 2  
else:  
    print('we got a {} and a {}'.format(omega[x2==1][0], omega[x2==1][1])) # show 1st and 2nd place of two 1s  
  
[1 2 3 4 5 6]  
[0 1 0 0 1 0]  
we got a 2 and a 5
```

In [5]: *'''repeat throwing two dice five times'''*

```
x25 = np.random.multinomial(n=2, pvals=ps, size=5) # five combined experiments: two dice at once  
print(' ', omega, ' omega')  
print(x25, ' results')  
print(x25.shape, 'shape of results-matrix')  
pasches = len(x25[x25==2])  
print('we got {} of same kind'.format(pasches))  
  
[1 2 3 4 5 6] omega  
[[0 0 0 1 1 0]  
 [0 0 1 0 0 1]  
 [0 1 0 1 0 0]  
 [0 0 0 1 0 1]  
 [0 0 0 1 1 0]] results  
(5, 6) shape of results-matrix  
we got 0 of same kind
```

In [6]: `print(omega, 'numbers')
print(x25.sum(axis=0), 'frequency total')`

```
[1 2 3 4 5 6] numbers  
[0 1 1 4 2 2] frequency total
```

```
In [7]: '''another variant: without replacement'''
N = 49
p49 = N*[1/N]
omega49 = np.arange(1, 49+1)
n49 = 6
lotto = np.random.choice(a=omega49, size=n49, replace=False) # draw 6 out of 49 without replacement
print('Lotto today gives {}.'.format(np.sort(lotto)))

Lotto today gives [ 3 18 31 34 43 49]. Maybe.
```

2.1 Zusammenfassung Multinomiale Verteilung

- Mehr als zwei mögliche Elementarereignisse
- i.i.d. Wiederholungen / Kombinationen möglich

2.1.1 Python

- Keine Verteilung in `scipy.stats`
- Möglichkeiten Zufallszahlen zu gewinnen
 - `np.random.choice()`
 - `np.random.multinomial()`

2.2 Wiederholung: Mittelwert \Leftrightarrow Erwartungswert

Arithmetischer **Mittelwert** von n Ereignissen x_j

$$\bar{x} = \frac{1}{n} \sum_{j=1}^n x_j$$

Diskrete Ereignisse $x_i, i \in \{1 \dots N\}$ mit absoluter Häufigkeit H_i , $\sum H_i = n$, dann

$$\bar{x} = \frac{1}{n} \sum_{i=1}^N H_i \cdot x_i$$

Mit relativen Häufigkeiten $h_i = \frac{H_i}{n}$ und damit $\sum h_i = 1$

$$\bar{x} = \sum_{i=1}^N h_i \cdot x_i$$

Übergang zur (theoretischen) Wahrscheinlichkeit p_i mit $\sum p_i = 1$: diskreter **Erwartungswert** der Zufallsvariable X

$$\mathcal{E}(X) = \mu = \sum_{i=1}^N p_i \cdot x_i$$

mit $p_i = p(X=x_i)$

3 Kontinuierliche Verteilungen: $x \in \mathbb{R}$

3.0.1 Wiederholung: Erwartungswert

Diskret

$$\mathcal{E}(X) = \mu = \sum_{i=1}^N p_i \cdot x_i$$

mit $p_i = p(X=x_i)$

Kontinuierlich

$$\mathcal{E}(X) = \mu = \int_{-\infty}^{\infty} f(x) \cdot x \, dx$$

mit Wahrscheinlichkeitsdichte $f(x)$

3.0.2 Wiederholung: kontinuierliche Zufallsvariable

Kontinuierliche Zufallsvariable X mit $x \in \mathbb{R}$

Problem: weil Punktwahrscheinlichkeit $P(x=a) \rightarrow 0$

Lösung (endliche) Wahrscheinlichkeitsdichte $f(x)$

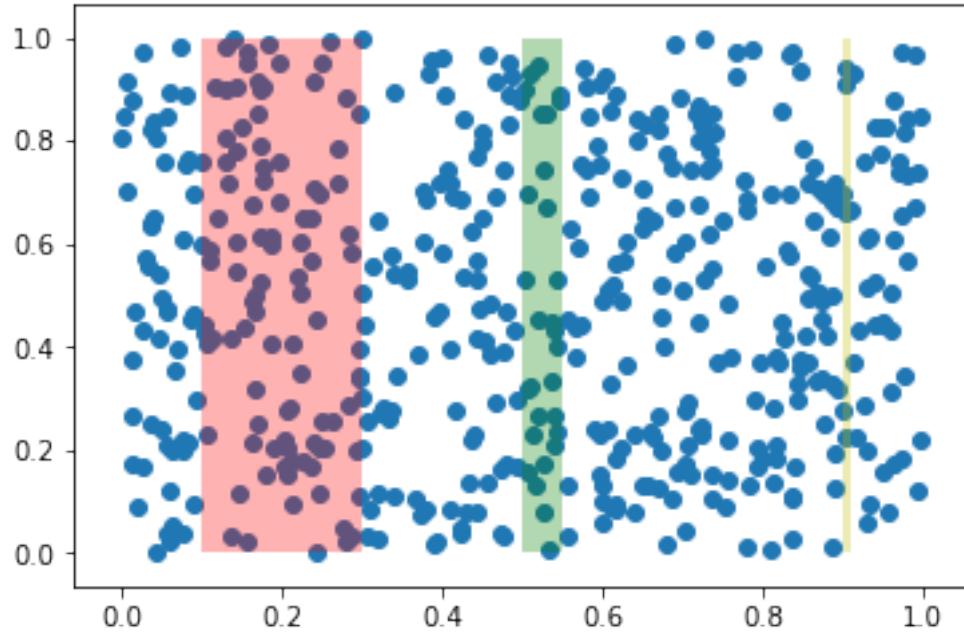
$$f(x) \geq 0$$

$$P(a \leq x \leq b) = \int_a^b f(x) dx \leq 1$$

Normierung $\int_{x=-\infty}^{\infty} f(x) dx = 1$

In [8]: *'''500 raindrops are falling - show density'''*

```
in width=0.20 are 100 drops, density=500.0
in width=0.05 are 29 drops, density=580.0
in width=0.01 are 6 drops, density=600.0
```



4 Kontinuierliche Verteilungen

5 Gleichverteilung / Rechteckverteilung

$$x \in [a, b]$$

$$f(x) = const$$

$$= \frac{1}{b-a}$$

5.1 Kennwerte Rechteck-/Gleichverteilung

$$\mathcal{E}(X) = \frac{a+b}{2}$$

$$Median(X) = \frac{a+b}{2}$$

$$\text{Var}(X) = \sigma^2 = \frac{(b-a)^2}{12}$$

$$\sigma = \frac{|b-a|}{2\sqrt{3}}$$

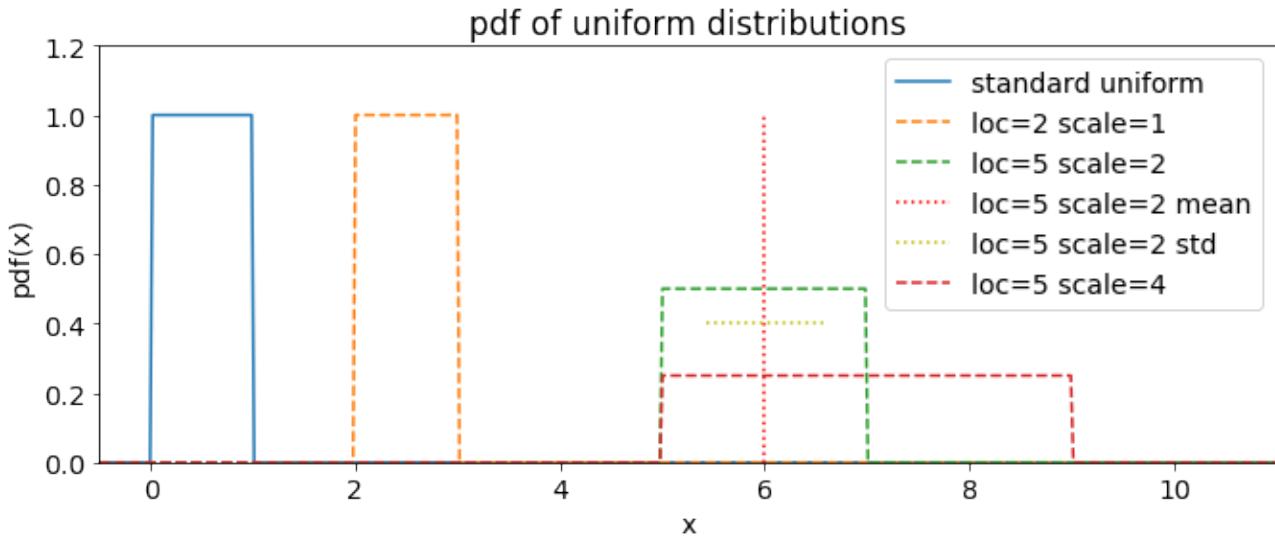
Beweis: [ÜA]

```
In [9]: '''uniform distribution % rectangle shaped
           probability density > 0 in range [a...b] = [loc...loc+range]
                           = 0 elsewhere
           ...
           # uniform without parameters is in range [0, 1]
           print('uniform()      has expectation value {:.5f} and variance {:.5f}'.
                 format(stats.uniform.expect(), stats.uniform.var()))
           ...
           # uniform in range [5..7] = [5..5+2]
           unif_5_2 = stats.uniform(loc=5.0, scale=2.0)          # freeze these parameters
           m52 = unif_5_2.expect()                            # frozen distribution's expectation
           s52 = unif_5_2.std()                               # ... and standard deviation
           print('uniform(5, 2) has expectation value {:.5f} and std-dev. {:.5f}'.format(m52, s52))

uniform()      has expectation value 0.50000 and variance 0.08333
uniform(5, 2) has expectation value 6.00000 and std-dev. 0.57735
```

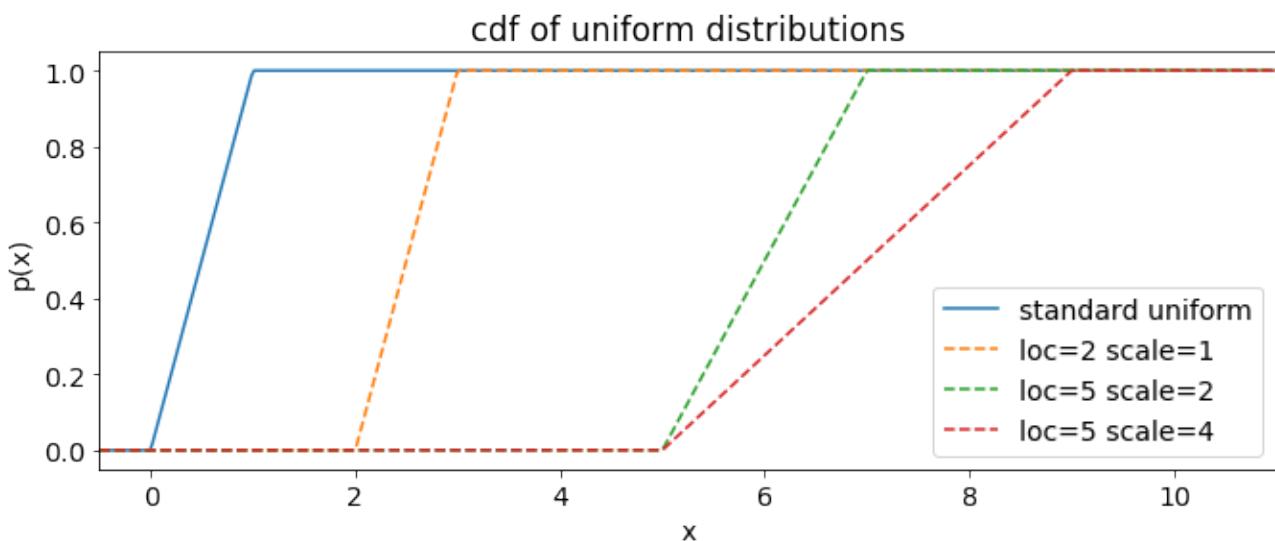
5.1.1 Wahrscheinlichkeitsdichte $f(x)$, pdf

```
In [10]: '''uniform distribution examples: probability density '''
plt.figure(figsize=(11,4))
x = np.linspace(-.5, 11., 465)
plt.plot(x, stats.uniform.pdf(x), label='standard uniform')
plt.plot(x, stats.uniform(loc=2., scale=1.).pdf(x), '--', label='loc=2 scale=1')
plt.plot(x, unif_5_2.pdf(x), '--', label='loc=5 scale=2')
plt.plot(2*[m52], [0, 1], 'r:', label='loc=5 scale=2 mean')
plt.plot([m52-s52, m52+s52], 2*[.4], 'y:', label='loc=5 scale=2 std')
plt.plot(x, stats.uniform(loc=5., scale=4).pdf(x), '--', label='loc=5 scale=4')
plt.xlabel('x')
plt.ylabel('pdf(x)')
plt.xlim(-0.5, 11.)
plt.ylim(0., 1.2)
plt.legend(loc='upper right')
plt.title('pdf of uniform distributions');
```



5.1.2 Verteilungsfunktion $F(x)$, kumulative Verteilung, cdf

```
In [11]: '''uniform cumulative distribution function examples'''
plt.figure(figsize=(11,4))
x = np.linspace(-.5, 11., 465)
plt.plot(x, stats.uniform.cdf(x), label='standard uniform')
plt.plot(x, stats.uniform(loc=2., scale=1.).cdf(x), '--', label='loc=2 scale=1')
plt.plot(x, unif_5_2.cdf(x), '--', label='loc=5 scale=2')
plt.plot(x, stats.uniform(loc=5., scale=4).cdf(x), '--', label='loc=5 scale=4')
plt.xlabel('x')
plt.ylabel('p(x)')
plt.xlim(-0.5, 11.)
plt.ylim(-0.05, 1.05)
plt.legend(loc='lower right')
plt.title('cdf of uniform distributions');
```



6 Normalverteilung

$x \in \mathbb{R}$ Wahrscheinlichkeitsdichte $f(x)$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \mathcal{N}(\mu, \sigma^2)$$

6.0.1 Standardnormalverteilung $\phi(x)$

Spezialfall $\mu = 0$ und $\sigma^2 = 1$:

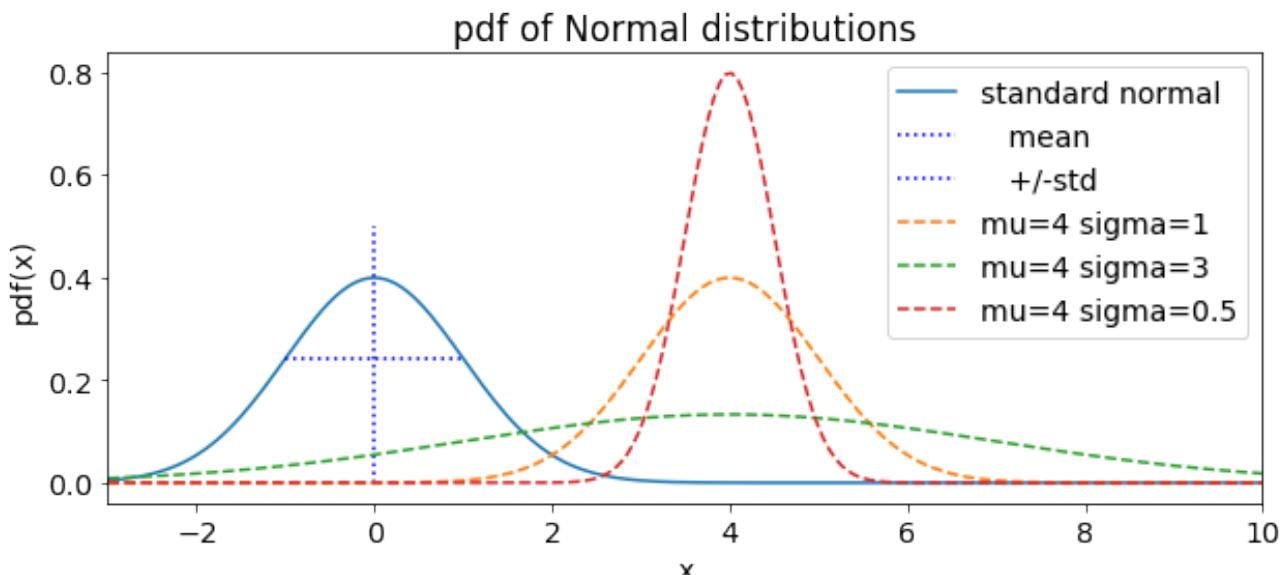
$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} = \mathcal{N}(0, 1)$$

6.1 Kennwerte der Gauß'schen Normalverteilung

$$\begin{aligned}\mathcal{E}(X) &= \mu \\ \text{Var}(X) &= \sigma^2 \\ \text{Schiefe} &= 0 \\ \text{Kurtosis} &= 0\end{aligned}$$

In [12]: *'a bunch of normal "Gaussian" distributions'*

```
plt.figure(figsize=(10,4))
x = np.linspace(-3., 10., 201)
plt.plot(x, stats.norm.pdf(x), label='standard normal')
m = stats.norm.mean()           # mean (expectation value) of distribution
s = stats.norm.std()            # standard deviation of distribution
h = stats.norm.pdf(1)           # pdf (x=sigma=1) for height
plt.plot(2*[m], [0, 0.5], 'b:', label='mean')      # vertical line at mean
plt.plot([m-s, m+s], 2*[h], 'b:', label=' +/- std') # horizontal line +/- std
# plot three other distributions at mu=4, different sigma (=std, =sqrt(var))
plt.plot(x, stats.norm(loc=4., scale=1.).pdf(x), '--', label='mu=4 sigma=1')
plt.plot(x, stats.norm(loc=4., scale=3.).pdf(x), '--', label='mu=4 sigma=3')
plt.plot(x, stats.norm(loc=4., scale=.5).pdf(x), '--', label='mu=4 sigma=0.5')
plt.xlim((-3, 10))
plt.xlabel('x')
plt.ylabel('pdf(x)')
plt.title('pdf of Normal distributions')
plt.legend();
```



7 (kumulierte) Wahrscheinlichkeitsfunktion

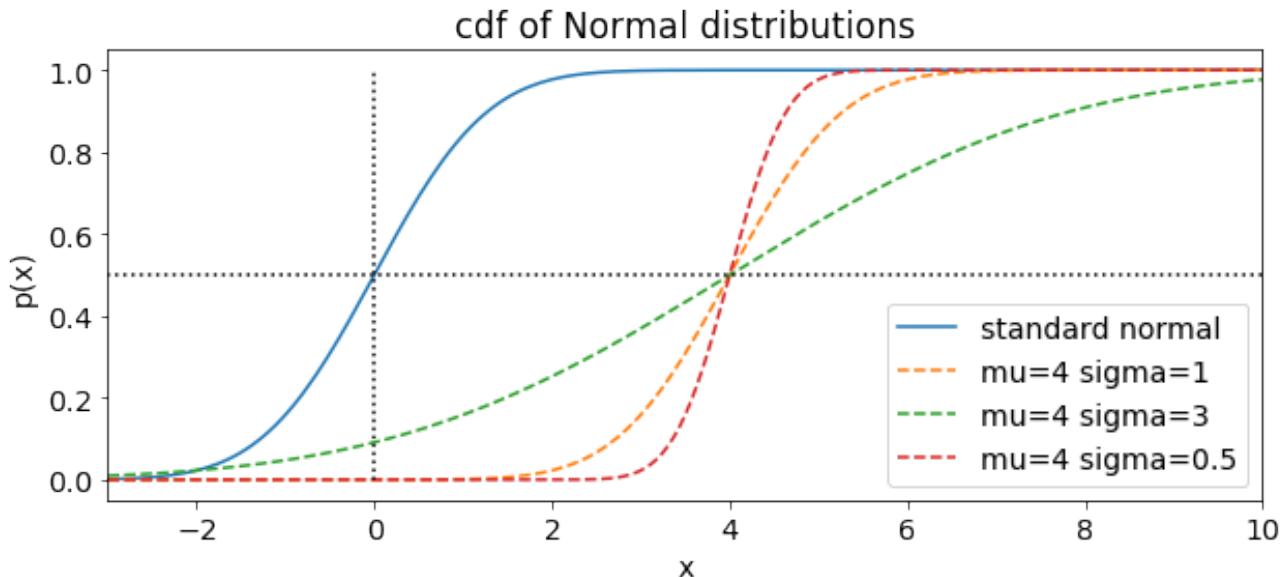
$$F(x) = \int_{-\infty}^x f(a) da$$

7.1 Standardnormalverteilungsfunktion

Für die Standardnormalverteilung $\mu = 0$ und $\sigma = 1$

$$\Phi(x) = \int_{-\infty}^x \phi(a) da$$

```
In [13]: '''a bunch of normal Gaussian cdfs'''
plt.figure(figsize=(10,4))
x = np.linspace(-3., 10., 201)
# the same four distributions as before:
plt.plot(x, stats.norm.cdf(x), label='standard normal')
plt.plot(x, stats.norm(loc=4., scale=1.).cdf(x), '--', label='mu=4 sigma=1')
plt.plot(x, stats.norm(loc=4., scale=3.).cdf(x), '--', label='mu=4 sigma=3')
plt.plot(x, stats.norm(loc=4., scale=.5).cdf(x), '--', label='mu=4 sigma=0.5')
plt.plot([-3., 10.], 2*[.5], 'k:')
plt.plot(2*[0.], [0., 1.], 'k:')
plt.xlabel('x')
plt.ylabel('p(x)')
plt.xlim((-3, 10))
plt.title('cdf of Normal distributions')
plt.legend(loc='lower right');
```



7.1.1 Quantile

```
In [14]: '''interesting quantiles'''
p = np.array([.50, .75, .90, .95, .975, .99])      # interesting percentiles
z = stats.norm.ppf(p)                                    # and their probability
print('cumulative probability p = {}'.format(np.round(p, decimals=3)))
print('located at           z = {}'.format(np.round(z, decimals=3)))

cumulative probability p = [ 0.5      0.75      0.9      0.95      0.975    0.99 ]
located at           z = [ 0.        0.674     1.282     1.645     1.96      2.326]
```

7.1.2 Tabelle der Standardnormalverteilung

Historisch Quelle: https://en.wikipedia.org/wiki/Standard_normal_table oder jedes Statistiklehrbuch.

7.1.3 68-95-99,7-Prozent-Abschätzungsregel

- 68% der Werte liegen in $\mu \pm \sigma$
- 95% der Werte liegen in $\mu \pm 2 \cdot \sigma$
- 99.7% der Werte liegen in $\mu \pm 3 \cdot \sigma$

```
In [15]: '''1-2-3-sigma = 68, 95, 99.7'''
z = np.array([1., 2., 3.])           # range 1, 2, 3 (sigma)
p = stats.norm.cdf(z)               # has cumulated probability
pr = 2*p-1.                        # probability range excluding both tails
original = np.get_printoptions()    # save print options
np.set_printoptions(formatter={'float': '{: 6.2f}'.format}) # print 2 digits precision
print('range of +/- sigma: z = {}'.format(z))
print('probability inside: p = {} %'.format(100*pr))
np.set_printoptions(**original)      # restore print options

range of +/- sigma: z = [ 1.00   2.00   3.00]
probability inside: p = [ 68.27  95.45  99.73] %
```

7.2 Eigenschaften

(Wichtig für später)

7.2.1 lineare Transformation

Ist $X \sim \mathcal{N}(\mu, \sigma^2)$, dann ist $Y = aX + b$ ebenfalls normalverteilt mit:

$$Y \sim \mathcal{N}(a \cdot \mu + b, a^2 \cdot \sigma^2)$$

7.2.2 Addition

Sind $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ und $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ beide unabhängig, dann ist $Y = X_1 + X_2$

$$Y \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

7.2.3 Standardisierung

Die spezielle lineare Transformation $Z = \frac{1}{\sigma}X - \frac{\mu}{\sigma}$ ergibt

$$Z \sim \phi = \mathcal{N}(0, 1)$$

weil darunter (siehe oben)

$$\mathbb{E}(Z) = 0$$

$$\text{Var}(Z) = 1$$

Umgekehrt kann man aus der Standardnormalverteilung ϕ jede Normalverteilung $\mathcal{N}(\mu, \sigma^2)$ erhalten durch

$$X = \sigma Z + \mu$$

7.3 Anpassen, fit()

Eine Verteilung an vorhandene Daten anfitten

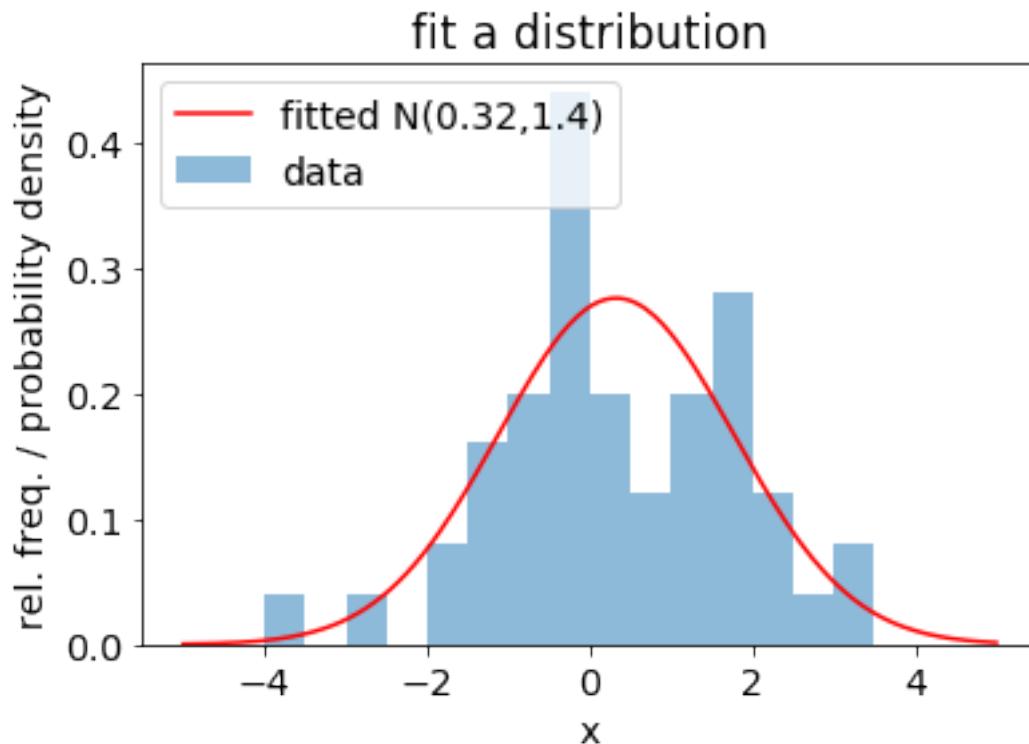
```
In [16]: '''fit distribution to data'''
np.random.seed(98765432)
data = stats.norm(0.7, 1.5).rvs(size=50)          # fake data
(mu, sigma) = stats.norm.fit(data)                # find best fitting Gauss for data
plt.hist(data, bins=np.linspace(-5, 5, 21), label='data',
```

```

        normed=True, alpha=.5)                      # use relative frequencies
print('fitted normal distribution has parameters mu={:.3}, sigma={:.3}'
      .format(mu, sigma))
xgrid = np.linspace(-5, 5, 201)
plt.plot(xgrid, stats.norm(mu, sigma).pdf(xgrid), 'r-', label='fitted N({:.2},{:.2})'.format(mu, sig
plt.title('fit a distribution')
plt.xlabel('x')
plt.ylabel('rel. freq. / probability density')
plt.legend(loc='upper left');

fitted normal distribution has parameters mu=0.325, sigma=1.45

```



8 Exponentialverteilung

Stetige Erweiterung der *Geometrischen Verteilung* für $x \in \mathbb{R}$: Dauer bis zum Eintreten eines Ereignisses.

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

Kenngrößen

$$\begin{aligned}\mathcal{E}(X) &= \frac{1}{\lambda} \\ x_{med} &= \frac{\ln 2}{\lambda} \\ \text{Var}(X) &= \frac{1}{\lambda^2}\end{aligned}$$

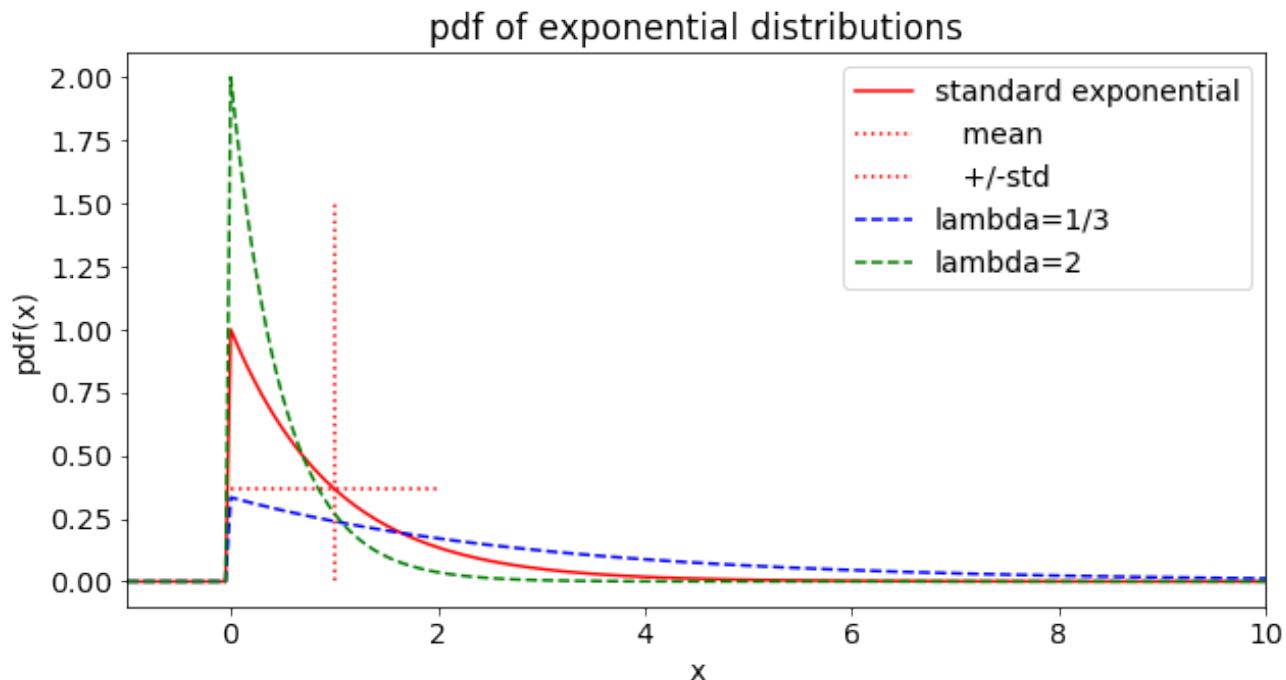
8.0.1 Realisierung in scipy.stats

```
stats.expon?
An exponential continuous random variable.
The probability density function for `expon` is::
    expon.pdf(x) = exp(-x)      for ``x >= 0``.
The probability density above is defined in the "standardized" form.
To shift and/or scale the distribution use the ``loc`` and ``scale`` parameters.
Specifically, ``expon.pdf(x, loc, scale)`` is identically equivalent to ``expon.pdf(y) / scale``
with ``y = (x - loc) / scale``. A common parameterization for `expon` is in terms of the
rate parameter ``lambda``, such that ``pdf = lambda * exp(-lambda * x)``. This
parameterization corresponds to using ``scale = 1 / lambda``.
```

8.0.2 Wahrscheinlichkeitsdichte $f(x)$, pdf

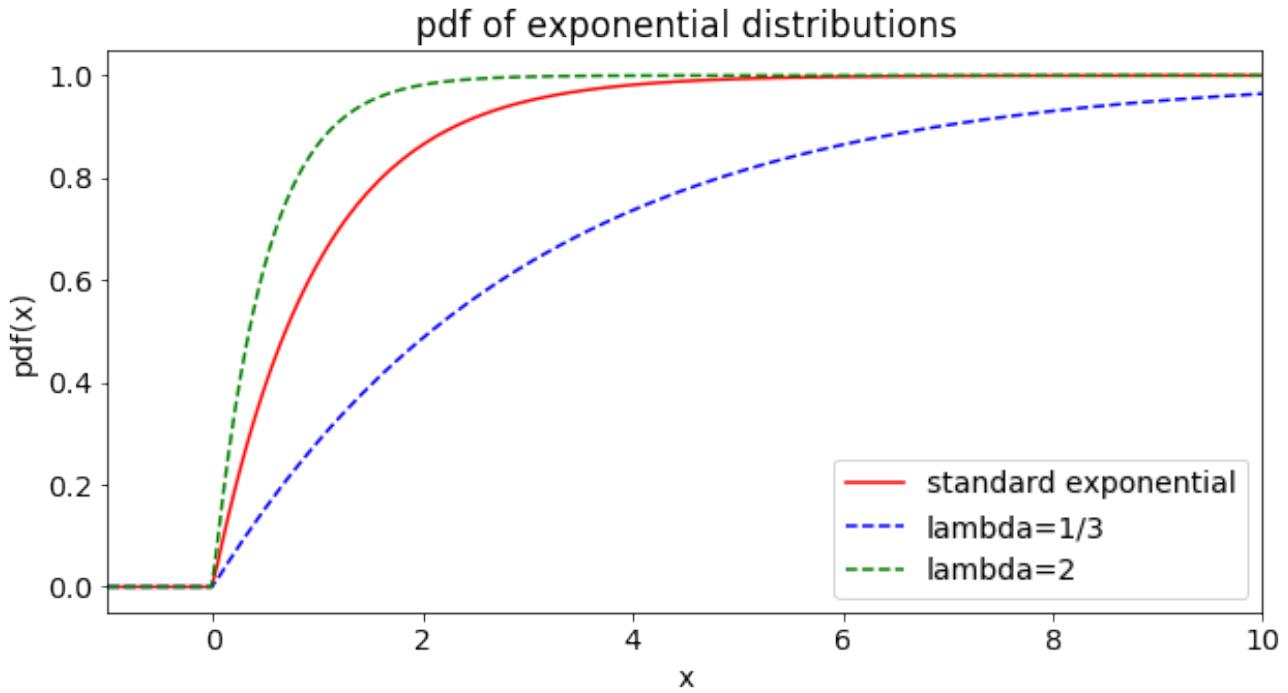
In [17]: *'exponential distribution'*

```
plt.figure(figsize=(10,5))
x = np.linspace(-1., 10., 221)      # x values (negatives do not matter)
plt.plot(x, stats.expon.pdf(x), 'r-', label='standard exponential')
m = stats.expon.mean()              # mean (expectation value) of distribution
s = stats.expon.std()               # and standard deviation
h = stats.expon.pdf(m)             # just for height at pdf(m):
plt.plot(2*[m], [0, 1.5], 'r:', label='mean')      # vertical line for m
plt.plot([m-s, m+s], 2*[h], 'r:', label='+/-std') # horiz. line for m +/- std
# two other exponentials with lambda=1/3 and 2
plt.plot(x, stats.expon(scale=3.).pdf(x), 'b--', label='lambda=1/3')
plt.plot(x, stats.expon(scale=.5).pdf(x), 'g--', label='lambda=2')
plt.xlim((-1, 10))
plt.xlabel('x')
plt.ylabel('pdf(x)')
plt.title('pdf of exponential distributions')
plt.legend();
```



8.0.3 Wahrscheinlichkeitsverteilung $F(x)$, cdf

```
In [18]: '''cdf of exponential distributions from before'''
plt.figure(figsize=(10,5))
x = np.linspace(-1., 10., 201)
plt.plot(x, stats.expon.cdf(x), 'r-', label='standard exponential')
plt.plot(x, stats.expon(scale=3.).cdf(x), 'b--', label='lambda=1/3')
plt.plot(x, stats.expon(scale=.5).cdf(x), 'g--', label='lambda=2')
plt.xlim((-1, 10))
plt.xlabel('x')
plt.ylabel('pdf(x)')
plt.title('pdf of exponential distributions')
plt.legend(loc='lower right');
```



9 Ausblick

9.0.1 AuSSergewöhnliche Verteilungen

10 Zusammenfassung

Kontinuierliche Verteilungen $x \in \mathbb{R}$

Rechteck-/Gleichverteilung

$$f(x) = \begin{cases} \text{const.} & x \in [a, b] \\ 0 & \text{sonst} \end{cases}$$

Normalverteilung, Gaußverteilung

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \mathcal{N}(\mu, \sigma^2)$$

Standardnormalverteilung $\phi(x)$

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} = \mathcal{N}(0, 1)$$

Exponentialverteilung

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

11 Zusammenfassung kontinuierliche V. in Python `scipy.stats`

Funktionen und Methoden

```
.uniform(loc, scale)      # borders    loc ... loc+scale
.expon(scale)              # exponent   1/scale
.norm(loc, scale)          # mu, sigma

.expect()                  # Erwartungswert
.pdf(x)                    # WahrscheinlichkeitsdichteVerteilung "probability density function"
.cdf(x)                    # Verteilungsfunktion "cumulative density function"
.ppf(p)                    # Umkehrung davon: percentile
.rvs()                     # Zufallsergebnis "random variables"
                           # (optional Anzahl der Werte)
.fit()                     # Anfitten einer Verteilung an Ereignisse/Daten

.expect()                  # Erwartungswert
.var()                     # Varianz
.std()                     # Standardabweichung
.kurtosis()                # Kurtosis
...
...
```

12 Fragen?

053_Folien

November 23, 2018

```
In [1]: import numpy as np          # mathematical methods
         from scipy import stats      # statistical methods
         from matplotlib import pyplot as plt    # plotting methods
         %matplotlib inline
```

1 Wahrscheinlichkeitstheorie

1.0.1 Zufallsvariable und Wahrscheinlichkeitsraum

1.0.2 Diskrete Zufallsvariablen und Wahrscheinlichkeitsverteilungen

1.0.3 Kontinuierliche Zufallsvariable und Wahrscheinlichkeitsverteilungen

Kontinuierliche Verteilungen

- Rechteckverteilung, Gauß'sche Normal-Verteilung, Exponentialverteilung

Mehrdimensionale Verteilungen

Außergewöhnliche Beispiele kontinuierlicher Verteilungen

- Pareto-Verteilung
- Cauchy-Verteilung

Zusammengesetzte Verteilungen

1.0.4 Wiederholung: Gleichverteilung / Rechteckverteilung

$$x \in [a, b]$$

$$f(x) = \text{const} = \frac{1}{b - a}$$

$$\mathcal{E}(X) = \mu = \frac{a + b}{2}$$

$$\text{Var}(X) = \sigma^2 = \frac{(b - a)^2}{12}$$

1.0.5 Wiederholung: Normalverteilung \mathcal{N}

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\mathcal{E}(X) = \mu$$

$$\text{Var}(X) = \sigma^2$$

$$\text{Schiefe} = 0$$

$$\text{Kurtosis} = 0$$

Standardnormalverteilung $\phi(z)$

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

$$\mathcal{E}(Z) = 0$$

$$\text{Var}(Z) = 1$$

(kumulierte) Wahrscheinlichkeitsfunktion

$$F(x) = \int_{-\infty}^x f(x') \, dx'$$

68-95-99,7-Prozent-Regel

1.0.6 Wiederholung: Exponentialverteilung

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

$$\mathcal{E}(X) = \frac{1}{\lambda}$$

$$x_{med} = \frac{\ln 2}{\lambda}$$

$$\text{Var}(X) = \frac{1}{\lambda^2}$$

2 Außergewöhnliche Verteilungen

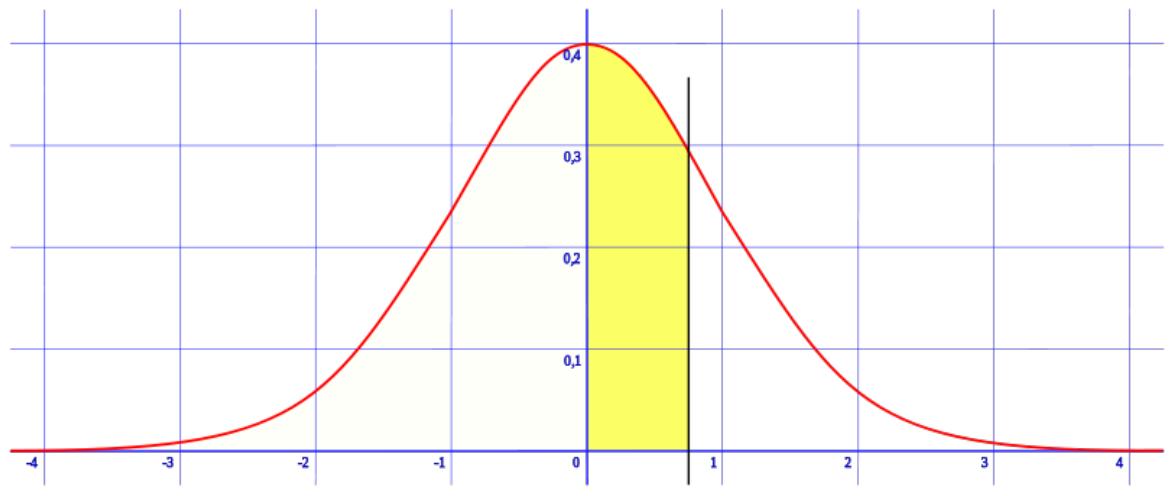
2.0.1 "Pareto-Effekt"

oder 80-zu-20-Regel:

- 80% der Ergebnisse werden mit 20% des Gesamtaufwandes erreicht
- Verbleibende 20% der Ergebnisse benötigen mit 80% die meiste Arbeit

2.1 Pareto Verteilung

$$f(x) = \begin{cases} \frac{k \cdot x_{\min}^k}{x^{(k+1)}} & \text{für } x \geq x_{\min} \\ 0 & \text{sonst} \end{cases}$$



Beispiel aus der Praxis

Quelle: Wikipedia (CC0 1.0 Verzicht auf das Copyright)

Wahrscheinlichkeitsdichte Pareto-Verteilung

$$f(x) = \begin{cases} \frac{k \cdot x_{\min}^k}{x^{(k+1)}} & \text{für } x \geq x_{\min} \\ 0 & \text{sonst} \end{cases}$$

Speziell mit Paramtern $k = 2$ und $x_{\min} = 1$:

Wahrscheinlichkeitsdichte

$$f(x) = \begin{cases} \frac{2}{x^3} & \text{für } x \geq 1 \\ 0 & \text{sonst} \end{cases}$$

Verteilungsfunktion

$$F(x) = \begin{cases} 1 - \frac{1}{x^2} & \text{für } x \geq 1 \\ 0 & \text{sonst} \end{cases}$$

2.1.1 Erwartungswert

$$\mathcal{E}(X) = \int_1^{\infty} \frac{2}{x^2} dx = 2$$

Wahrscheinlichkeitsdichte

$$f(x) = \begin{cases} \frac{2}{x^3} & \text{für } x \geq 1 \\ 0 & \text{sonst} \end{cases}$$

2.1.2 Besonderheit: Varianz

$$\mathcal{E}(X^2) = \int_1^{\infty} \frac{2}{x} dx = \infty$$

Und damit auch

$$\text{Var}(X) = \mathcal{E}(X^2) - (\mathcal{E}(X))^2 = \infty$$

2.1.3 Modus:

$$x_{\min}$$

2.1.4 Median:

$$x_{\min} \sqrt[k]{2}$$

2.1.5 letztes Quintil:

$$Q_{-}\{0.8\} = x_{-}\{\min\} \sqrt[k]{5}$$

$$\mathcal{E}(X|X > Q_{0.8}) = x_{\min} \frac{k}{k-1} / 5^{\frac{k-1}{k}}$$

80% ergibt sich für $k \approx 1,16$.

```
In [2]: '''German Cities' population > 10.000'''
# source: https://www.destatis.de/DE/ZahlenFakten/LaenderRegionen/Regionales/
#           Gemeindeverzeichnis/Administrativ/Archiv/GVAuszugQ/AuszugGV3QAktuell.html
import pandas as pd # allows easy import from data
limit = 14815 # 10000 # cities, not all villages, arbitrary
data = pd.read_table('data/Staedte2015_10kplus.csv', sep=';', ) # read destatis data
print('population of German cities:')
print(data[:8])
pop = np.asarray([r[1] for i, r in enumerate(data.values)]) # get data in array
pop = pop[pop>=limit] # restrict above limit
```

```
population of German cities:
      Name      0
0        Berlin  3421829
1      Hamburg  1746342
2     München  1407836
3       Köln  1034175
4 Frankfurt am Main  701350
5    Stuttgart  604297
6   Düsseldorf  598686
7     Dortmund  575944
```

```
In [4]: '''fit empirical data to pareto distribution'''
    pfit = stats.pareto.fit(pop, floc=0)      #fit shape and scale, keep location fixed@0
    print('German cities populations can be fitted as Pareto({:.3f}, {:.0f}) distributed'
          .format(pfit[0], pfit[2]) )
```

German cities populations can be fitted as Pareto(1.310, 14825) distributed

```
In [5]: '''Quintiles of German cities' population'''
    # empirical quintiles from sorted cities' index
quintiles = np.round(np.linspace(0, pop.shape[0]-1, 5+1)).astype(int)
    # theor. quintiles of fitted pareto distribution; be careful @ borders
fitq = stats.pareto.ppf([.0005, .2, .4, .6, .8, .9995], b=pfit[0], scale=pfit[2])
print('From {} real cities,'.format(pop.shape[0]))
empqpop = np.zeros_like(fitq)    # (space for) empirical population in fitted quintiles
for q in range(5):
    pfrom, pto = (fitq[q], fitq[q+1])
    part = pop[(pfrom<pop) & (pop<pto)]
    empqpop[q+1] = empqpop[q] + part.shape[0]
    print('      {:.0f} are in {:.0f}. fit-quintile from {:.6.0f} to {:.7.0f}'.
          format(empqpop[q+1]-empqpop[q], q+1, pfrom, pto))
```

From 974 real cities,

170	are in 1. fit-quintile from	14831	to	17579
213	are in 2. fit-quintile from	17579	to	21898
198	are in 3. fit-quintile from	21898	to	29844
211	are in 4. fit-quintile from	29844	to	50668
181	are in 5. fit-quintile from	50668	to	4916705

```
In [6]: '''Test of pareto principle 80%20-20%80'''
n1 = np.asarray([n for n in pop[:182]]).sum()
n2 = np.asarray([n for n in pop[182:]]).sum()
print('{} habitants in 20% biggest cities'.format(n1))
print('{} habitants in 80% smaller cities'.format(n2))
```

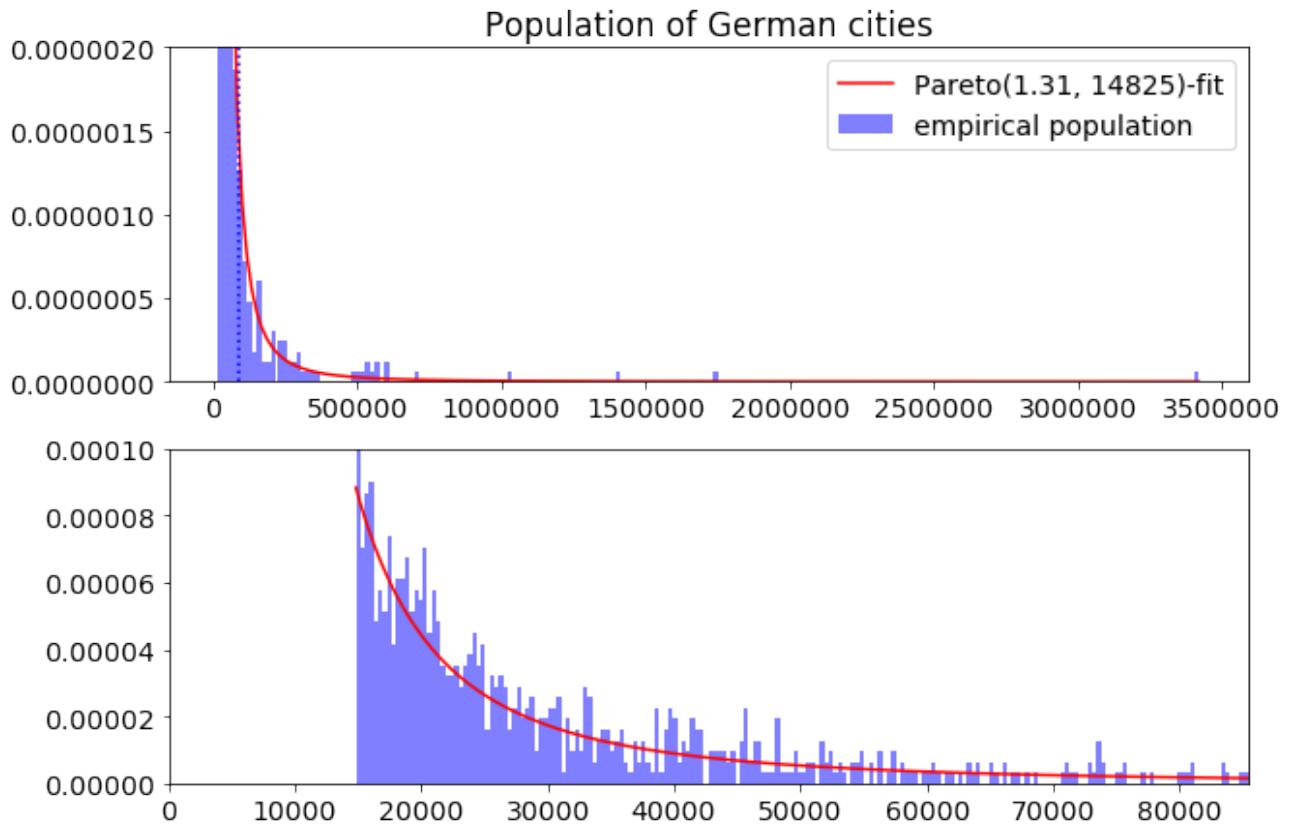
32302386 habitants in 20% biggest cities
20010417 habitants in 80% smaller cities

```
In [7]: '''German cities - Pareto distribution - histogram'''
    f = plt.figure(figsize=(10,7))
    f.add_subplot(211)
    plt.title('Population of German cities')
    # plot empirical data
    pthreas = 85400    # Tuebingen
    bins = np.linspace(pop.min(), pop.max(), 201)
    plt.hist(pop, color='b', normed=True, bins=bins, alpha=.5,
             label='empirical population')
    # plot the fit
```

```

popi = np.linspace(pop.min(), pop.max(), 201)
plt.plot(popi, stats.pareto.pdf(popi, b=pfit[0], scale=pfit[2]), 'r-',
         label='Pareto({:.2f}, {:.0f})-fit'.format(pfit[0], pfit[2]))
plt.ylim(0, 0.000002);
plt.legend()
plt.plot(2*[pthreas], [0, 0.000002], 'b:')
f.add_subplot(212)
# plot empirical data up to threshold
bins = np.linspace(pop.min(), pthreas, 201)
plt.hist(pop, color='b', normed=True, bins=bins, alpha=.5)
# plot fit
popi = np.linspace(pop.min(), pthreas, 201)
plt.plot(popi, stats.pareto.pdf(popi, b=pfit[0], scale=pfit[2]), 'r-')
plt.axis((0, pthreas, 0, 0.0001));

```



```

In [8]: '''German cities - Pareto distribution - x-axis: index of city'''
f = plt.figure(figsize=(10,4))
f.add_subplot(121) # --- plot first quantile ---
plt.title('Population of German cities')
# plot empirical data
plt.plot(pop, 'b-', label='empirical population')
plt.xlim(0, 1.2*empqpop[1]) # show only 1st 20% of cities
plt.ylim(0, 1.05*pop.max()) # max range including biggest city
plt.plot(2*[empqpop[1]], (0, pop.max()), 'k--',
         label='fitted quintiles') # mark the border between 1st and 2nd
# plot fitted data compressed to city after city
n_q1 = np.int(empqpop[1]) # number of cities in highest quintile
a_q1 = pop[0] # between 3,5Mio and
b_q1 = pop[n_q1-1] # and ~50K
print('{} biggest cities have populations range {} to {}'.format(n_q1, b_q1, a_q1))
x = np.linspace(0.999, 0.8, n_q1) # probabilities for biggest quintile

```

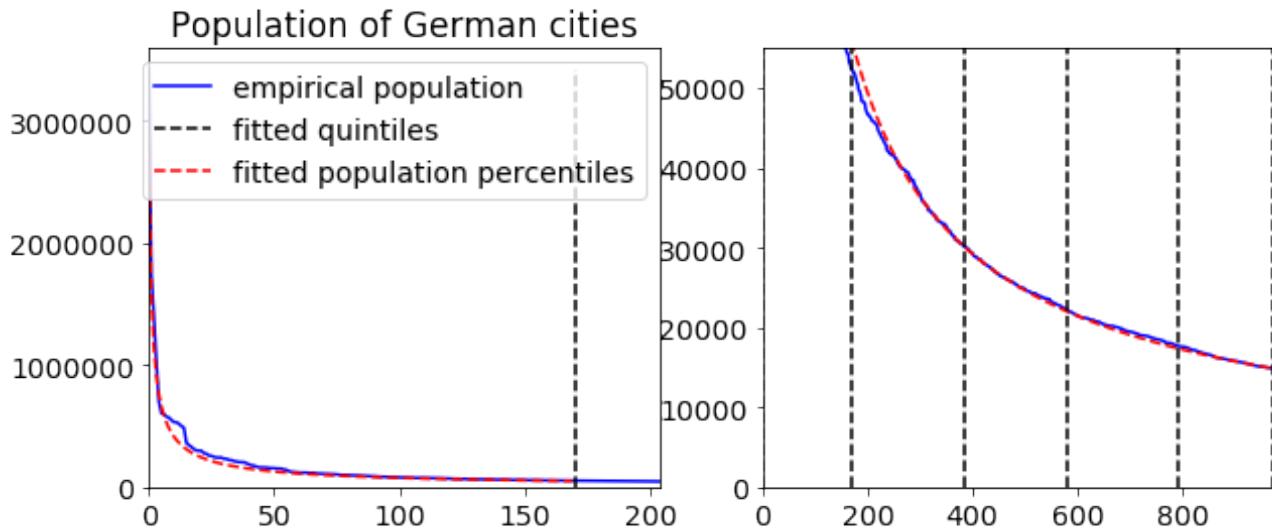
```

popf = stats.pareto.ppf(x, b=pfit[0], scale=pfit[2])    # city population at percentile
plt.plot(popf, 'r--', label='fitted population percentiles')
plt.legend(loc='upper right')

f.add_subplot(122)                                     # --- same plot for whole range ---
plt.plot(pop, 'b-')
plt.xlim(0, empqpop[-1])                             # show complete range to last quintile border
ymax = 1.05*pop[n_q1]                                # max range starting 2nd quintile
# plot fitted data compressed to city after city
n_q1 = pop.shape[0]                                   # number of cities total
x = np.linspace(0.999, 0.001, n_q1)                 # probabilite percentiles for number of cities
popf = stats.pareto.ppf(x, b=pfit[0], scale=pfit[2])  # city population at percentile
plt.plot(popf, 'r--')
for q in empqpop:
    plt.plot((q, q), (0, ymax), 'k--'); # show quintiles of fitted distribution

```

170 biggest cities have populations range 52400 to 3421829.



3 Cauchy/Lorentz-Verteilung

Familie von Verteilungen mit

$$f(x) = \frac{1}{\pi} \frac{s}{s^2 + (x - t)^2}$$

Speziell mit dem Zentrum $t = 0$ und der Breite $s = 1$ Standard-Cauchy-Verteilung:

$$f(x) = \frac{1}{\pi} \frac{1}{1 + x^2}$$

Kumulative Verteilungsfunktion

$$F(x) = \frac{1}{2} + \frac{1}{\pi} \cdot \arctan x$$

3.1 Anwendung

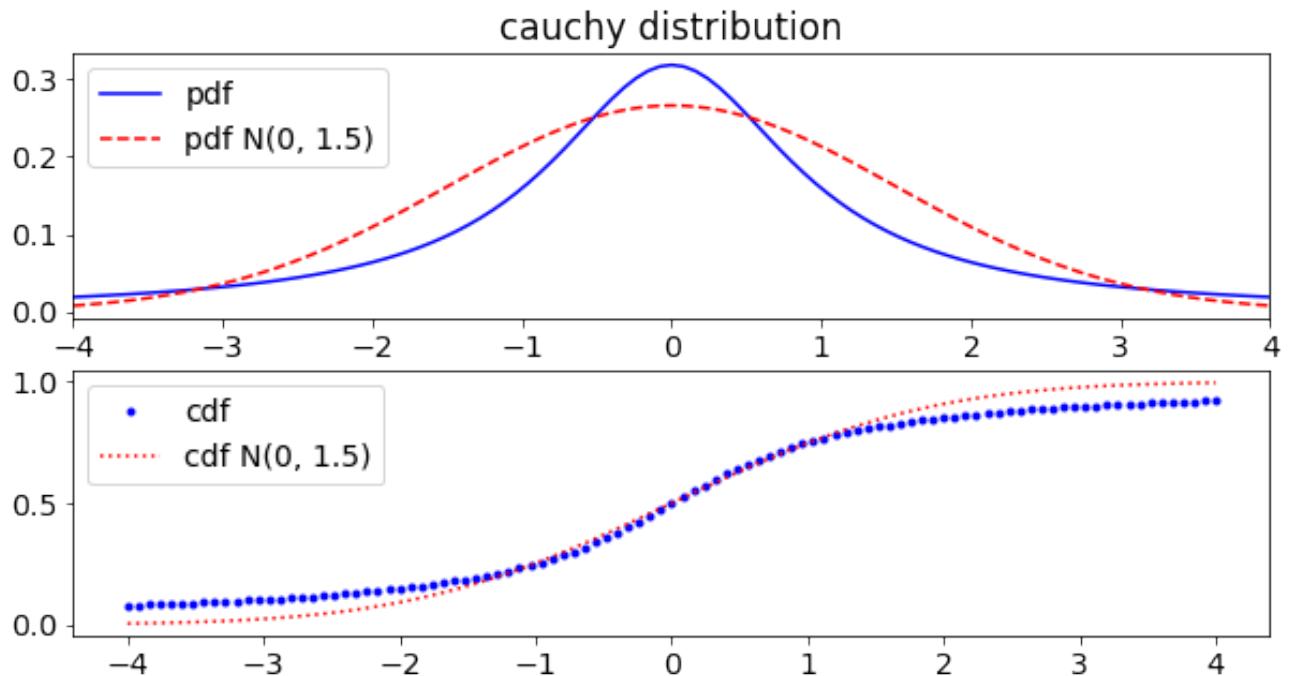
Physik:

- Resonante Schwingung
- Form von Spektrallinien

Beziehung zur Normalverteilung Der Quotient aus zwei unabhängigen standardnormalverteilten Zufallsvariablen ist Standard-Cauchy-verteilt.

Beziehung zu Student'schen t-Verteilung Die Standard-Cauchy-Verteilung ist der Spezialfall der studentschen t-Verteilung mit einem Freiheitsgrad.

```
In [10]: '''cauchy distribution - compare to Normal distribution'''
X = 4. # x = {-X..X}
x = np.linspace(-X, X, 101)
f = plt.figure(figsize=(10,5))
f.add_subplot(211)
plt.xlim((-X, X))
plt.title('cauchy distribution')
plt.plot(x, stats.cauchy.pdf(x), 'b-', label='pdf')
plt.plot(x, stats.norm(scale=1.5).pdf(x), 'r--', label='pdf N(0, 1.5)')
plt.legend(loc='upper left');
f.add_subplot(212)
plt.plot(x, stats.cauchy.cdf(x), 'b.', label='cdf')
plt.plot(x, stats.norm(scale=1.5).cdf(x), 'r:', label='cdf N(0, 1.5)')
plt.legend(loc='upper left');
```



3.1.1 Besonderheit:

Keines der Momente ist definiert,

die Integrale für Erwartungswert, Varianz, ... konvergieren alle **nicht!**

Moment-Integrale berechnen über Residuensatz. Oder Fouriertransformation....

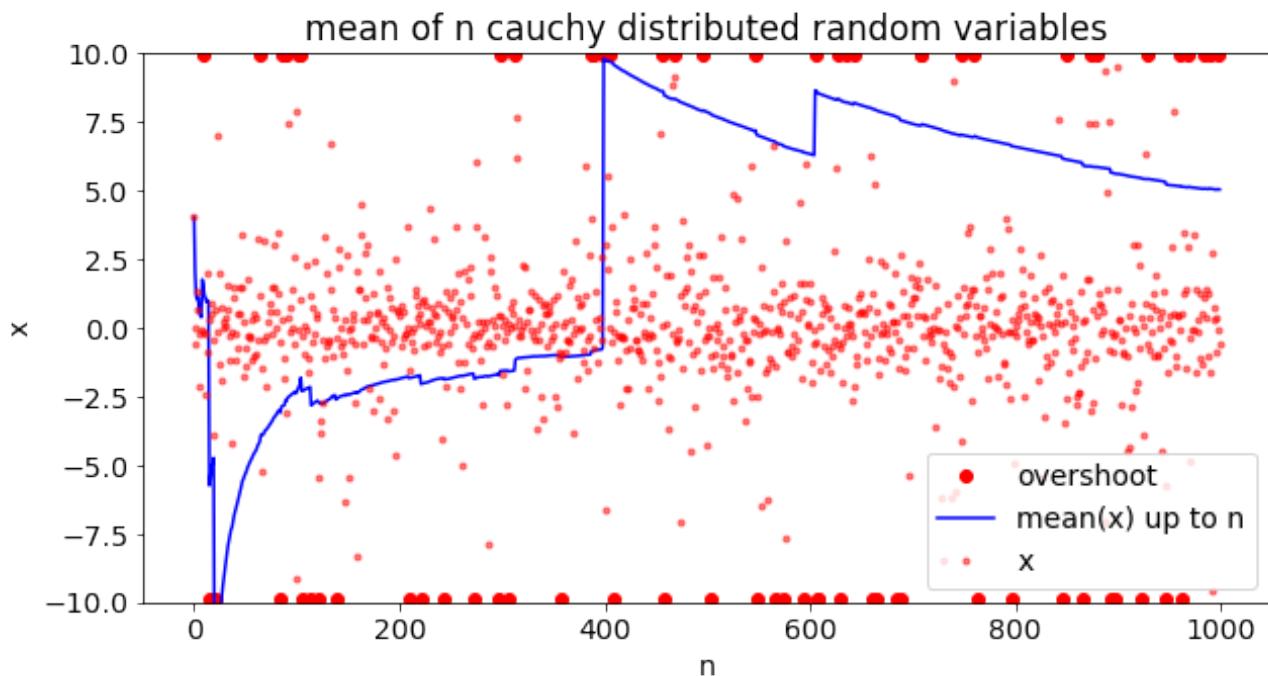
Varianz ist berechenbar:

$$\begin{aligned}
 E[X^2] &\propto \int_{-\infty}^{\infty} \frac{x^2}{1+x^2} dx \\
 &= \int_{-\infty}^{\infty} \left(1 - \frac{1}{1+x^2}\right) dx \\
 &= \int_{-\infty}^{\infty} dx - \int_{-\infty}^{\infty} \frac{1}{1+x^2} dx \\
 &= \int_{-\infty}^{\infty} dx - \pi \\
 &= \infty
 \end{aligned}$$

Erwartungswert:

- undefiniert
- Warum nicht Null aufgrund von Symmetrie?

```
In [11]: '''simulate expectation value of cauchy distribution'''
N = 1000
# samples to draw
A = 10.
# plot x = {-A..A}
np.random.seed(987624)
x = np.random.standard_cauchy(N)
# fix random (have a nice example)
n = np.arange(N)
# draw according cauchy distribution
# i for x-axis; means of x until i
m = np.asarray([x[:i+1].mean() for i in n]) # mean for 0..i each
f = plt.figure(figsize=(10,5))
plt.ylim(-A, A)
q = np.ones_like(x)*2.*A
# q out of plot region
q[x>A] = A-.1
# if x out of plot region, set q to border
plt.plot(n, q, 'ro', label='overshoot')
q = np.ones_like(x)*2.*A
# and
q[x<-A] = -A+.1
# ...
# same for "undershoot"
plt.plot(n, q, 'ro')
plt.plot(n, m, 'b-', label='mean(x) up to n') # the means up to i
plt.plot(n, x, 'r.', alpha=.5, label='x') # values of drawn samples
plt.legend(loc='lower right')
plt.xlabel('n')
plt.ylabel('x')
plt.title('mean of n cauchy distributed random variables');
```



4 Zusammenfassung

Außergewöhnliche Verteilungen

- Pareto-Verteilung
- Cauchy/Lorentz-Verteilung

Kennwerte

- ?!

5 Fragen?