Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

FS24 CAS PML - Python

Niklaus Johner

niklausbernhard.johner@bfh.ch

FS24 CAS PML - Python

17. scientific computing with scipy

Was ist scipy?

- *scipy* ist eine Sammlung von mathematischen Funktionen und Algorithmen um mit Daten zu arbeiten
- *scipy* ist auf *numpy* basiert
- Ist in sub-Paketen organisiert
- Man importiert normalerweise individuelle Pakete

```
from scipy import linalg
```

Packeten

- **constants**: Physical and mathematical constants
- **cluster**: Clustering algorithms
- **fftpack**: Fast Fourier Transform routines
- **integrate**: Integration and ordinary differential equation solvers
- **interpolate**: Interpolation and smoothing splinesioInput and Output
- **linalg**: Linear algebra
- **ndimage**: N-dimensional image processing
- **odr**: Orthogonal distance regression
- **optimize**: Optimization and root-finding routines
- **signal:** Signal processing
- **sparse**: Sparse matrices and associated routines
- **spatial**: Spatial data structures and algorithms
- **special**: Special functions
- **stats**: Statistical distributions and functions

optimization with *optimize*

- Importiert mit:

```
from scipy import optimize
```

- Funktion an Daten anpassen

```
# fit function to data
optimize.curve_fit(func, xdata, ydata)
```

```
In [71]: x = np.arange(0,10)

In [72]: y = 2*x + 3 + np.random.random(10)

In [73]: optimize.curve_fit(lambda x, a, b: a*x + b, x, y)
Out[73]:
(array([ 2.01604381,  3.47456626]), array([[ 0.00083143, -0.00374144],
        [-0.00374144,  0.02369581]]))
```

Zusätzliche Folien

optimization with *optimize*

- Skalarfunktion minimieren:

```
# Minimize scalar function of one or more variables
optimize.minimize(func, x0)
```

```
In [68]: optimize.minimize(lambda x: (x[0]+4)**2 + x[1]**2,[0, 0])
Out[68]:
      fun: 4.15731431839641e-16
 hess_inv: array([[  5.00000000e-01,   3.12421838e-09],
       [  3.12421838e-09,   1.00000000e+00]])
      jac: array([ -1.43082524e-09,  -2.24644908e-08])
  message: 'Optimization terminated successfully.'
     nfev: 16
      nit: 3
     njev: 4
   status: 0
  success: True
        x: array([ -4.00000001e+00,  -1.86828260e-08])
```

statistics with *stats*

- Importiert mit:

```
from scipy import stats
```

- Daten beschreiben:

```
# description of data
# (min, max, average, variance...)
stats.describe(m)
```

- Korrelation:

```
# pearson correlation and p-value
stats.pearsonr(b, b)
```

- ttest:

```
# t-test for 1 population
# (test that mean of a is mean)
stats.ttest_1samp(b, mean)
```

clustering with *cluster*

- Importiert mit:

```
from scipy import cluster
```

- Daten clustern mit dem *kmeans* Algorithmus:

```
# Classify a set of observations into k clusters
# using the k-means algorithm.
cluster.vq.kmeans2(data, k)
```

```
In [87]: data = np.array([[1,2],[0.1,1],[3,0],[4,-0.1]])

In [88]: cluster.vq.kmeans2(data, 2)
Out[88]:
(array([[ 0.55,  1.5 ],
        [ 3.5 , -0.05]]), array([0, 0, 1, 1], dtype=int32))
```