

PuLP – ein Python LP-Modellierer

How to use PuLP

Dr. Klaus Ladner

Institut für Statistik und Operations Research

1. Dezember 2016

Was ist PuLP?

Ein LP-Modellierer

- ▶ CBC (COIN-OR Branch und Cut) – Solver
- ▶ PuLP – Python-LP-Modellierer
(CBC, GLPK, Gurobi, CPLEX)
- ▶ Python – universelle höhere Programmiersprache

Warum PuLP?

Python/PuLP/CBC

- ▶ Teil eines Programmier-Universums/keine Insel
- ▶ 3-teiliger modularer Aufbau/kein Monolith
- ▶ pythonisch (einfach)
- ▶ skaliert (löst große Probleme)
- ▶ open source (frei verwendbar)

Installation

Allgemeines

WinPython – Python Distribution für Windows

- ▶ einfache Installation (monolithisch)
- ▶ Scientific-Python (viele Pakete)
- ▶ ua. Python, Spyder (IDE), PuLP, CBC
- ▶ ca. 1 GB Speicherplatz
- ▶ auch ohne Admin-Rechte möglich
- ▶ kann von USB-Stick starten

Alternativen:

- ▶ Anaconda (plattformunabhängig)
- ▶ Offizielle Webseite: <https://www.python.org/>

WinPython

Download

<https://winpython.github.io/>

The screenshot shows the WinPython website. At the top, there are navigation links for 'releases', 'overview', and 'portable'. The main heading is 'WinPython' with a logo consisting of four colored squares (blue, yellow, blue, yellow). Below the heading is the tagline: 'The easiest way to run Python, Spyder with SciPy and friends out of the box on any Windows PC, without installing anything!'. The page is divided into sections: 'Project Home is on [Github](#), downloads page are on [Sourceforge](#). Discussion group is on [Google Groups](#), md5 and sha1 [here](#)'. The 'Recent Releases' section lists three releases: '2016_05' (November 11th, 2016), '2016_04' (August 28th, 2016), and '2016_03' (July 23th, 2016). Each release includes a list of highlights and links to 'Changelog', 'Packages and Downloads', and 'alternative Downloads'. The 'Overview' section at the bottom states: 'WinPython is a free open-source portable distribution of the [Python programming language](#) for Windows 7/8/10 and scientific and educational usage.'

Installationspaket herunterladen ...

WinPython
Portable Scientific Python 2/3 32/64bit Distribution for Windows
Brought to you by: [praybait](#), [stonebg](#)

Summary Files Reviews Support Wiki Discussion Mailing Lists Tickets Mercurial

Looking for the latest version? [Download WinPython 64bit-3.5.2.3Qt5.exe \(278.5 MB\)](#)

Home / WinPython_3_5 / 3.5.2.3

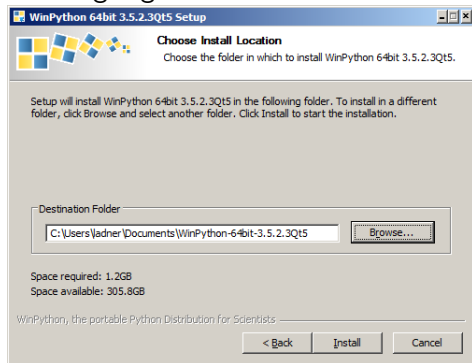
Name	Modified	Size	Downloads / Week
Parent folder			
betas	2016-11-05		1
WinPython-64bit-3.5.2.3.exe	2016-11-10	265.5 MB	788
WinPython-64bit-3.5.2.3Qt5.exe	2016-11-10	278.5 MB	4,544
WinPython-32bit-3.5.2.3Qt5.exe	2016-11-10	239.8 MB	716
WinPython-64bit-3.5.2.3Zero.exe	2016-11-10	23.6 MB	100
WinPython-32bit-3.5.2.3Zero.exe	2016-11-10	22.9 MB	257
Totals: 6 Items		830.4 MB	6,405

... und ausführen.

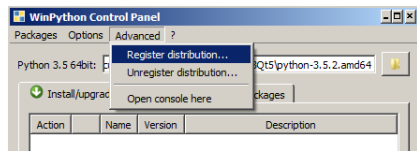
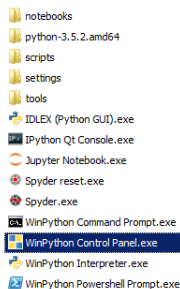
WinPython

Installation

Wähle geeignetes Verzeichnis



Optional: Starte Control Panel aus zuvor gewähltem Verzeichnis und registriere.

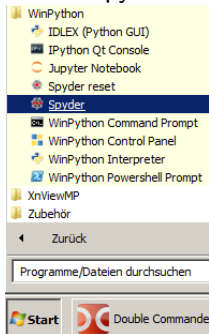


WinPython

IDE optional

Öffne Datei im File explorer (rechte Maustaste:
Edit), Run file (F5)

Starte Spyder



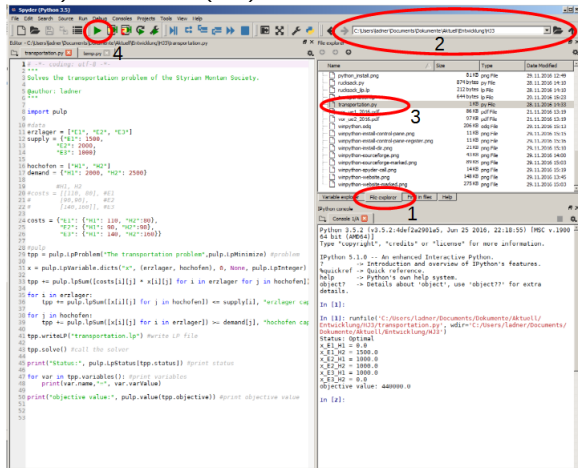
WinPython

- IDLEX (Python GUI)
- IPython Qt Console
- Jupyter Notebook
- Spyder reset
- Spyder**
- WinPython Command Prompt
- WinPython Control Panel
- WinPython Interpreter
- WinPython Powershell Prompt
- XnViewMP
- Zubehör

Zurück

Programme/Dateien durchsuchen

Start Double Commande



The screenshot shows the Spyder Python 3.5.2 IDE interface. A File Explorer window is open, displaying the file system path `C:\Users\ladner\Documents\Dokumente\Aktuell\Entwicklung\K12`. The file `transportation.py` is selected and circled in red, with a red circle labeled '3'. The File Explorer's 'View' menu is open, showing options like 'File explorer', 'File', and 'WebP', with 'File explorer' selected and circled in red, labeled '1'. The Spyder code editor shows the following Python code:

```
1 # -*- coding: utf-8 -*-
2 ...
3 Solve the transportation problem of the Styrian Mountain Society.
4
5 @author: Ladner
6 ***
7
8 import pulp
9
10 #Data
11 erzlager = ["E1", "E2", "E3"]
12 supply = {"E1": 1500,
13           "E2": 2000,
14           "E3": 1000}
15
16 hochofen = ["H1", "H2"]
17 demand = {"H1": 2000, "H2": 2500}
18
19 #Cost
20 #costs = [{"i": "E1", "j": "H1", "cost": 80},
21           {"i": "E1", "j": "H2", "cost": 90},
22           {"i": "E2", "j": "H1", "cost": 100},
23           {"i": "E2", "j": "H2", "cost": 110},
24           {"i": "E3", "j": "H1", "cost": 120},
25           {"i": "E3", "j": "H2", "cost": 130}]
26
27 #Problem
28 tpp = pulp.LpProblem("The transportation problem", pulp.LpMinimize) #problem
29 x = pulp.LpVariable.dicts("x", (erzlager, hochofen), 0, None, pulp.LpInteger)
30
31 tpp += pulp.lpSum([costs[i][j] * x[i][j] for i in erzlager for j in hochofen])
32
33 tpp += pulp.lpSum([supply[i] * x[i][j] for i in erzlager for j in hochofen])
34
35 for i in erzlager:
36     tpp += pulp.lpSum([x[i][j] for j in hochofen]) == supply[i], "erzlager %s" % i
37
38 for j in hochofen:
39     tpp += pulp.lpSum([x[i][j] for i in erzlager]) >= demand[j], "hochofen %s" % j
40
41 tpp.writeLP("transportation.lp") #write LP file
42
43 tpp.solve() #call the solver
44
45 print("Status:", pulp.LpStatus[tpp.status]) #print status
46
47 for var in tpp.variables(): #print variables
48     print(var.name, "=", var.value)
49
50 print("objective value:", pulp.value(tpp.objective)) #print objective value
51
52
53
```

Übersicht

Drei Probleme

- ▶ Einfaches LP (LP-File, Ergebnisausgabe, stetige Variable)
- ▶ Rucksack-Problem ($\sum_i x_i$, binäre Variable)
- ▶ Transport-Problem ($\sum_i \sum_j x_{ij}$, ganzzahlige Variable)

Einfaches LP

Formulierung

$$\begin{array}{rllll} \text{maximiere} & 3x_1 & + & x_2 & + & x_3 & & & & \\ \text{bezüglich} & 3x_1 & + & 2x_2 & + & 2x_3 & \leq & 10 & & \\ & -x_1 & + & 3x_2 & - & x_3 & \leq & 13 & & \\ & & & -x_2 & - & x_3 & \leq & 7 & & \\ & 2x_1 & & & & + & x_3 & = & 2 & \\ & x_1 & \geq & 0, & x_2 & \geq & 0 & & & \end{array}$$

Einfaches LP

Zielfunktion

maximiere $3x_1 + x_2 + x_3$
 $x_1 \geq 0, x_2 \geq 0$

```
import pulp

lp_simple = pulp.LpProblem("A simple
                           lp problem", pulp.LpMaximize)

x1 = pulp.LpVariable("x_1", 0)
x2 = pulp.LpVariable("x_2", 0)
x3 = pulp.LpVariable("x_3")

lp_simple += 3*x1 + x2 + x3,
            "objective function"
```

Einfaches LP

Restriktionen

$$\begin{array}{rcll} \text{bezüglich} & 3x_1 + 2x_2 + 2x_3 & \leq & 10 \\ & -x_1 + 3x_2 - x_3 & \leq & 13 \\ & & -x_2 - x_3 & \leq 7 \\ & 2x_1 & + x_3 & = 2 \end{array}$$

```
lp_simple += 3*x1 + 2*x2 + 2*x3 <= 10,
            "first constraint"
lp_simple += -x1 + 3*x2 -x3 <= 13,
            "second constraint"
lp_simple += -x2 -x3 <= 7,
            "third constraint"
lp_simple += 2*x1 + x3 == 2, "equation"
```

Einfaches LP

LP-File

```
lp_simple.writeLP("lp_simple.lp")
```

```
\* A simple lp problem *\
Maximize
objective_function: 3 x_1 + x_2 + x_3
Subject To
equation: 2 x_1 + x_3 = 2
first_constraint: 3 x_1 + 2 x_2 + 2 x_3 <= 10
second_constraint: - x_1 + 3 x_2 - x_3 <= 13
third_constraint: - x_2 - x_3 <= 7
Bounds
x_3 free
End
```

Einfaches LP

Ergebnis

```
lp_simple.solve()

print("Status:",
      pulp.LpStatus[lp_simple.status])
for var in lp_simple.variables():
    print(var.name, "=", var.varValue)
print("objective value:",
      pulp.value(lp_simple.objective))
```

```
Status: Optimal
x_1 = 6.0
x_2 = 3.0
x_3 = -10.0
objective value: 11.0
```

Rucksack-Problem

Formulierung

LP

$$\begin{aligned} \max \quad & \sum_j p_j x_j \\ \text{st.} \quad & \sum_j w_j x_j \leq c \\ & x_j \in \{0, 1\} \end{aligned}$$

Daten

j	1	2	3	4	5
p_j	10	6	3	8	1
w_j	10	6	4	9	3

$$c = 19$$

Rucksack-Problem

Daten

j	1	2	3	4	5
p_j	10	6	3	8	1
w_j	10	6	4	9	3

$$c = 19$$

```
import pulp

#data
profits = [10,6,3,8,1]
weights = [10,6,4,9,3]
c = 19
```

Rucksack-Problem

Zielfunktion

$$\max \sum_j p_j x_j$$
$$x_j \in \{0, 1\}$$

```
rucksack_ilp = pulp.LpProblem("The
                        rucksack ILP",
                        pulp.LpMaximize) #problem

x = pulp.LpVariable.dicts("x",
    [x+1 for x in range(5)],
    0, 1, pulp.LpInteger) #variables

rucksack_ilp += pulp.lpSum(
    [profits[j-1] * x[j] for j in x]),
"objective function"
```


Rucksack-Problem

Restriktion

$$\text{st. } \sum_j w_j x_j \leq c$$

```
rucksack_ilp += pulp.lpSum(  
[weights[j-1] * x[j] for j in x]) <= c,  
"capacity constraints"
```

```
Status: Optimal  
x_1 = 1.0  
x_2 = 0.0  
x_3 = 0.0  
x_4 = 1.0  
x_5 = 0.0  
objective value: 18.0
```

Transport-Problem

Formulierung

LP

$$\begin{aligned} \min \quad & \sum_i \sum_j c_{ij} x_{ij} \\ \text{st.} \quad & \sum_j x_{ij} \leq E_i \quad \forall i \\ & \sum_i x_{ij} \geq H_j \quad \forall j \\ & x_{ij} \in \mathbb{N}_0 \end{aligned}$$

Daten

Euro	H_1	H_2
E_1	110	80
E_2	90	90
E_3	140	160

$$E = (1.500, 2.000, 1.000)$$

$$H = (2.000, 2.500)$$

Transport-Problem

Daten

Euro	H_1	H_2
E_1	110	80
E_2	90	90
E_3	140	160

$$E = (1.500, 2.000, 1.000)$$

$$H = (2.000, 2.500)$$

```
erzlager = ["E1", "E2", "E3"]
supply = {"E1": 1500,
          "E2": 2000,
          "E3": 1000}
hochofen = ["H1", "H2"]
demand = {"H1": 2000, "H2": 2500}
costs = {"E1": {"H1": 110, "H2": 80},
         "E2": {"H1": 90, "H2": 90},
         "E3": {"H1": 140, "H2": 160}}
```

Transport-Problem

Zielfunktion

$$\min \sum_i \sum_{j \in \mathbb{N}_0} c_{ij} x_{ij}$$

```
tpp = pulp.LpProblem("The transportation
problem",pulp.LpMinimize) #problem

x = pulp.LpVariable.dicts("x",
(erzlager, hochofen),
0, None, pulp.LpInteger) #variables

tpp += pulp.lpSum(
[costs[i][j] * x[i][j] for i in erzlager
for j in hochofen]), "objective function"
```

Transport-Problem

Restriktionen

$$\text{st. } \sum_j x_{ij} \leq E_i \quad \forall i$$
$$\sum_i x_{ij} \geq H_j \quad \forall j$$

```
for i in erzlager:
    tpp += pulp.lpSum([x[i][j]
    for j in hochofen]) <= supply[i],
    "erzlager capacity
    constraints {}".format(i)

for j in hochofen:
    tpp += pulp.lpSum([x[i][j]
    for i in erzlager]) >= demand[j],
    "hochofen capacity
    constraints {}".format(j)
```

Transport-Problem

Ergebnis

```
Status: Optimal
x_E1_H1 = 0.0
x_E1_H2 = 1500.0
x_E2_H1 = 1000.0
x_E2_H2 = 1000.0
x_E3_H1 = 1000.0
x_E3_H2 = 0.0
objective value: 440000.0
```

Tipps & Tricks

Kosten-Matrix als CSV-File

CSV-File durch zB Excel gespeichert

```
Euro;H1;H2  
E1;110;80  
E2;90;90  
E3;140;160
```

```
import pandas as pd  
df = pd.read_csv("transportation_costs.csv",  
                 sep=";", index_col=0)  
costs = df.to_dict(orient="index")  
erzlager = df.index.tolist()  
hochofen = df.columns.tolist()
```

Tipps & Tricks

Vektoren als CSV-File

CSV-File durch zB Excel gespeichert

```
supply;demand  
1500;2000  
2000;2500  
1000;
```

```
df = pd.read_csv("transportation_vectors.csv"  
                , sep=";")  
supply = {erzlager[i] : df.supply[i]  
          for i in range(len(erzlager))}  
demand = {hochofen[j] : df.demand[j]  
          for j in range(len(hochofen))}
```


Tipps & Tricks

Kosten-Matrix als Liste von Listen

```
                #H1, H2
cost_lists = [[110, 80], #E1
              [90, 90], #E2
              [140,160]] #E3

costs = {}
for idx, row in enumerate(cost_lists):
    inner_dict = {}
    for idx_inner, column in enumerate(row):
        inner_dict[hochofen
                  [idx_inner]] = column
    costs[erzlagar[idx]] = inner_dict
```

Weiterführende Literatur

Links

- ▶ PuLP:
 - ▶ <https://pythonhosted.org/PuLP/>
- ▶ Python:
 - ▶ <https://docs.python.org/3/>
 - ▶ <http://www.python-course.eu/>