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

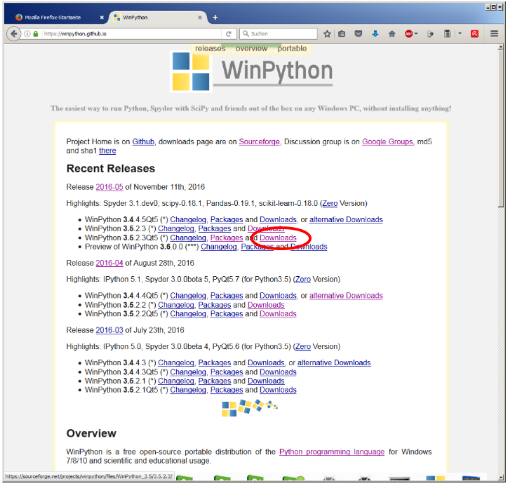
Alternativen:

- Anaconda (plattformunabhängig)
- Offizielle Webseite: `https://www.python.org/`

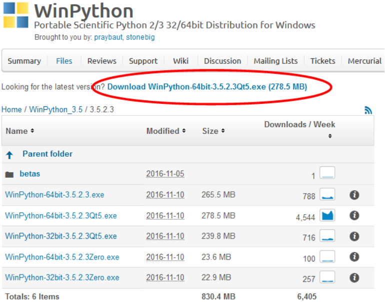
WinPython

Download

`https://winpython.github.io/`



Installationspaket herunterladen …

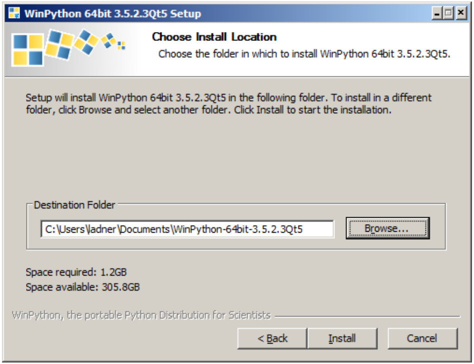


… und ausführen.

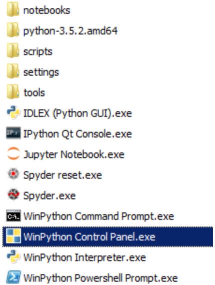
WinPython

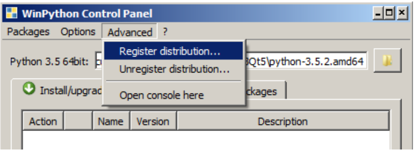
Installation

Wähle geeignetes Verzeichnis



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



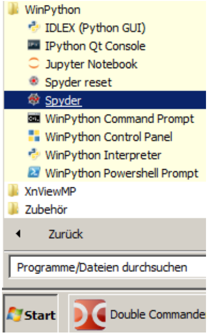


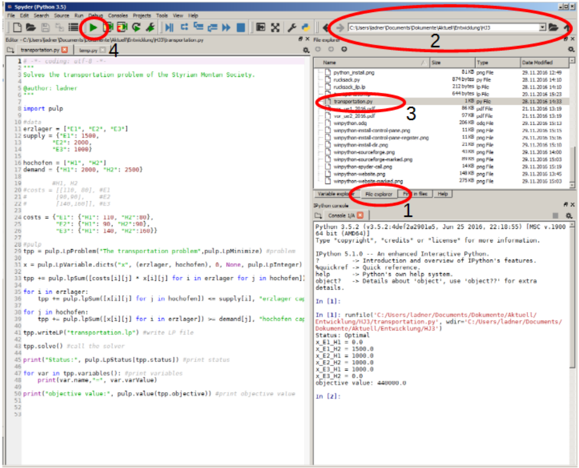
WinPython

IDE optional

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

Starte Spyder





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

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

Einfaches LP

Zielfunktion

$$\text{maximiere } 3x_1 + x_2 + x_3$$
$$x_1 \geq 0,\ x_2 \geq 0$$

```
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

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

```
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

$$\max \sum_j p_j \, x_j$$
$$\text{st.} \sum_j w_j \, x_j \leq c$$
$$x_j \in \{0, 1\}$$

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

$$\min \sum_i \sum_j c_{ij}\, x_{ij}$$

$$\text{st.} \sum_j x_{ij} \leq E_i \qquad \forall i$$

$$\sum_i x_{ij} \geq H_j \qquad \forall j$$

$$x_{ij} \in \mathbb{N}_0$$

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

$$\min \sum_i \sum_j c_{ij}\, x_{ij}$$
$$x_{ij} \in \mathbb{N}_0$$

```
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

$$\text{st.} \sum_j x_{ij} \leq E_i \qquad \forall i$$

$$\sum_i x_{ij} \geq H_j \qquad \forall j$$

```
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[erzlager[idx]] = inner_dict
```

Weiterführende Literatur

Links

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