Lagerhaltung

Dickbauer Y., Moser P., Perner M.

PS Computergestützte Modellierung, WS 2016/17

January 6, 2017

Outline

Aufgabenstellung

Ein Großhändler steht vor folgendem Lagerhaltungsproblem: Die Lagerung eines Produktes kostet 2 Euro pro Tag, eine Bestellung beim Lieferanten verursacht Kosten von 20 Euro (unabhängig von der Bestellmenge), der Verdienstentgang für ein nachgefragtes, aber nicht vorhandenes Produkt beträgt 15 Euro.

Die tägliche Nachfragemenge sei

1. binomialverteilt mit n $= 7$ und p $= 0.5$
2. poissonverteilt mit $\lambda = 3$.

Aufgabenstellung

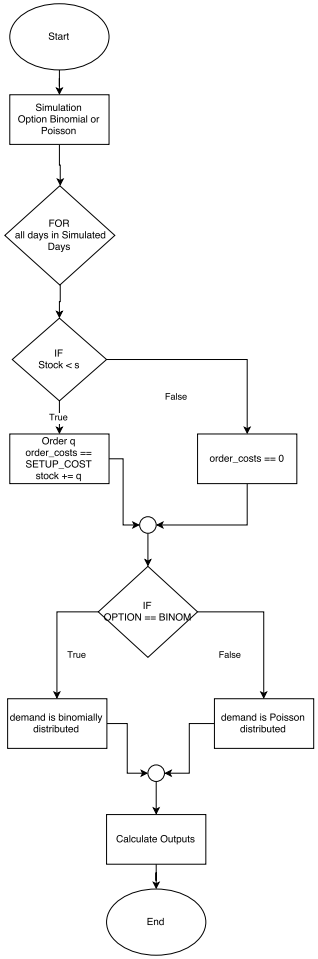
Die Bestellregel für den Disponenten soll lauten: Bestelle q Stück, wenn weniger als s Stück auf Lager sind. Bestimmen Sie dazu durch Simulation kostenoptimale Werte q, s (Variation von q und s zwischen 2 und 8 Stück mit $q \geq s$).

- Eingabe: -
- Output: Lagerbestand zu Periodenstart, Einkauf, Nachfrage, Verdienstentgang und Gesamtkosten je Periode

Das Problem der Lagerhaltung

- Es entstehen Kosten für
  - Lagerhaltung = Holding Cost
  - Ordering Cost
  - Vertriebsentgang = Penalty Cost
- Es wird eine Bestellmenge q festgelegt
- Es wird ein safety stock s festgelegt
- falls stock ¡ s wird q bestellt

Flow Chart


Start
Simulation
Option Binomial or
Poisson
FOR
all days in Simulated
Days
IF
Stock < s
False
True
Order q
order_costs ==
SETUP_COST
stock += q
order_costs == 0
IF
OPTION == BINOM
True
False
demand is binomially
distributed
demand is Poisson
distributed
Calculate Outputs
End

Main Funktion - Programmeinstieg

```
1   def main():
2       print('Simulate␣Option␣a)␣-␣Binom:')
3       sum_costs_binom = simulation(OPTION_BINOM)
4
5       for i in range(5): print()
6
7       print('Simulate␣Option␣b)␣-␣Poisson:')
8       sum_costs_poisson = simulation(OPTION_POISSON)
9
10      print('Sum␣costs␣of␣all␣periods␣of␣Option␣a:␣{}'.format(sum_costs_binom))
11      print('Sum␣costs␣of␣all␣periods␣of␣Option␣b:␣{}'.format(sum_costs_poisson))
```

- Funktion simulation(): siehe .py file

Funktion random_binom(n, p)

- Erzeugt binomialverteilte Zufallszahl
- $z := |\{i | u_i < p\}|$

```
1  def random_binom(n, p):
2      z = 0
3      # create n random numbers
4      # between [0,1]:
5      for i in range(n):
6          rand = random.random()
7          z += 1 if rand < p else 0
8      return z
```

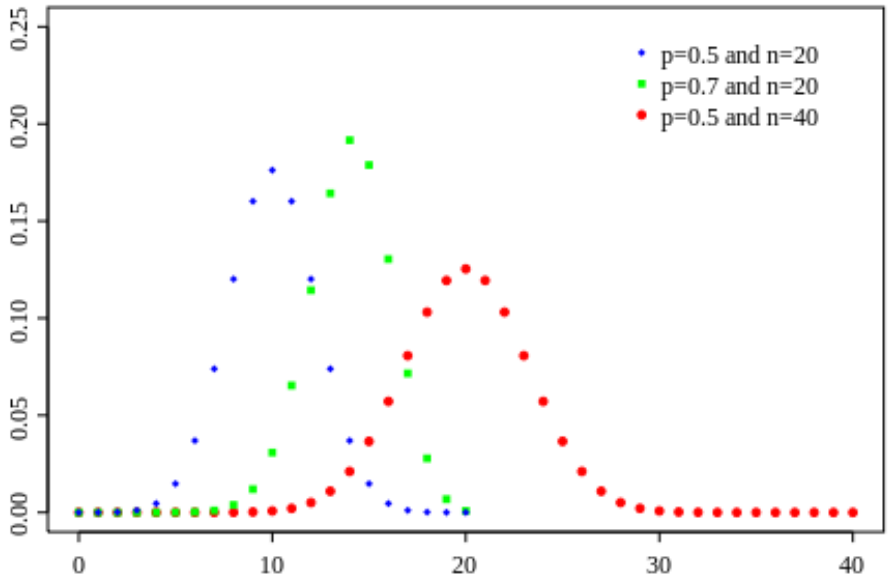


Figure 1: Wahrscheinlichkeitsverteilung der Binomialverteilung (Wikipedia)

Funktion random_poisson(λ)

- Erzeugt poisson verteilte Zufallszahl
- $\prod_{k}^{i=1} u_i \leq e^{-\lambda} < \prod_{k-1}^{i=1} u_i$
- Sobald obige Bedingung zutrifft ist P(λ) verteilte Zufallszahl k - 1

```
1   def random_poisson(lambd):
2     k = 0
3     u_list = []
4     middle_value = exp(-lambd)
5     while True:
6       k += 1
7       u_list.append(random.random())
8       left_side = product(u_list)
9       right_side = product(u_list[0:-1])
10      if left_side <= middle_value < right_side:
11        break
12    return k - 1
```

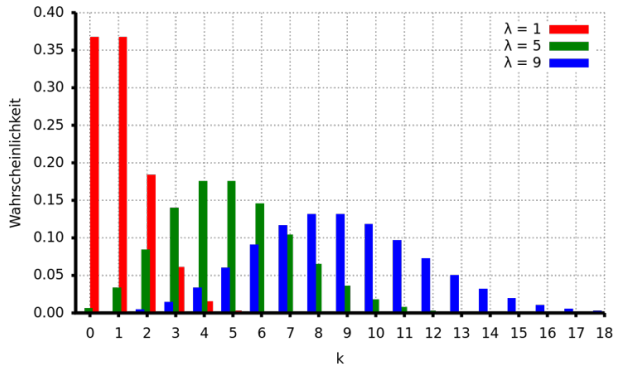


Figure 2:
Wahrscheinlichkeitsverteilung der Poisson-Verteilung (Wikipedia)