Selbst-organisierende, adaptive Systeme

Globalübung 2: Experiment-Design: Statistical Tests

Universität Augsburg University

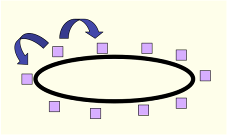
Erinnerung

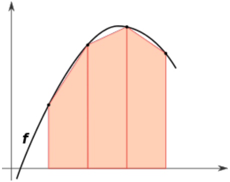
- Experimente beginnen mit *Behauptungen*
- Behauptungen sind X ist besser als Y
- Wir haben es zu tun mit:
  - unabhängigen Variablen (Faktoren, Parameter)
  - abhängigen Variablen (Messwerte, kausal beeinflusste Größen)
- Erhebung statistischer Kenngrößen (Mittelwert, Median, Standardabweichung, etc.)

Heute

- Aussagekräftigkeit der erhobenen Werte
- Ist der Mittelwert von X *signifikant* größer als der von Y?

Erinnerung [1]





- Rekursive Berechnungsaufgabe (z.B. Trapezregel zur numerischen Integration)
- Aufteilung als Binärbaum in Ringpuffer von Prozessoren
- Zwei Strategien (Heuristiken)
  - KOSO (**k**eep **o**ne **s**end **o**ne)
  - KOSO* (**k**eep **o**ne **s**end **o**ne to the least heavily loaded neighbor)

[1] [Gregory et al.(1996)Gregory, Gao, Rosenberg, and Cohen]

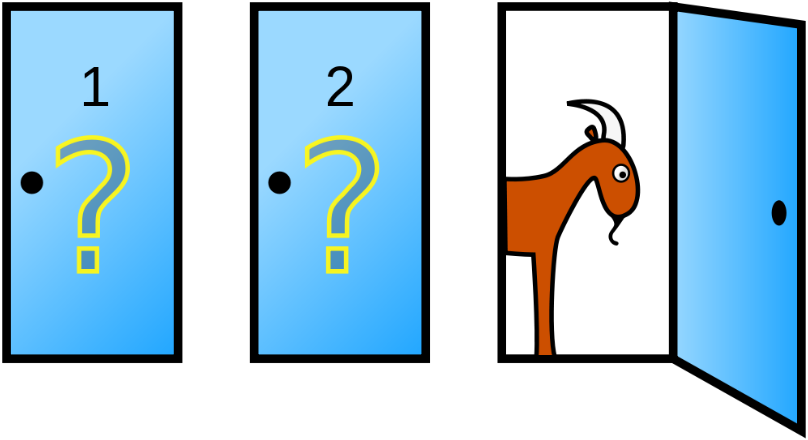
Die Idee von statistischen Tests

Institute for Software & Systems Engineering

Um zu zeigen, dass der Mittelwert der Population X *signifikant* von Y abweicht, ...

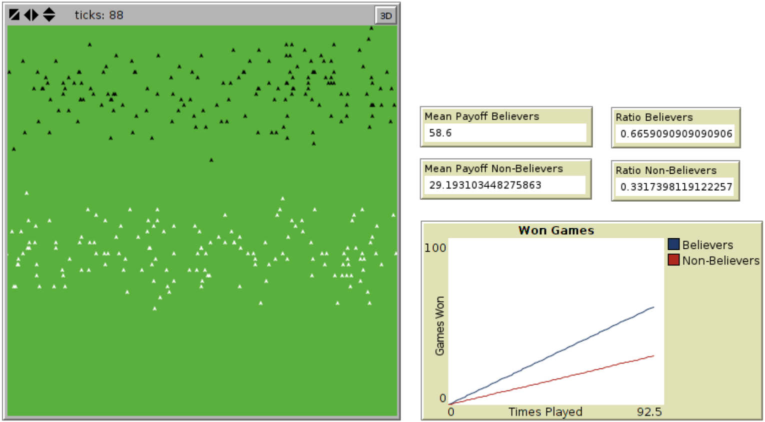
1. Behauptung (*Nullhypothese* H_0): $X = Y$
2. Führe ein Experiment durch und messe die Stichprobenstatistik (z.B. Mittelwert, Standardabweichung)
3. Berechne Wahrscheinlichkeit, dass dieser Unterschied *zufällig* auftrat (geeigneter *Test*)
   - Aufgetretener Unterschied sehr unwahrscheinlich $\rightarrow$ *lehne* H_0 *ab*
   - Ansonsten *behalte* H_0 bei (nicht genügend "Beweise")

Das “Monty-Hall-Problem”



Beispiel

Behauptung: “Wechsler” gewinnen häufiger als “Bleiber”



Darker turtles switch doors more often.
Lighter turtles switch doors less often.

Rohdaten

Institute for Software & Systems Engineering

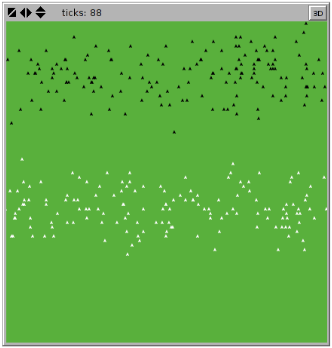
Abhängige Variablen (Messungen): Eine Zeile pro Turtle: Anteil gewonnener Spiele (von 88 Ticks)

blacks.csv

1	0.674
2	0.629
3	0.517
4	0.663
5	0.674
6	0.596
7	0.596
8	0.652
9	0.685
10	0.607
11	0.64
12	0.685
13	0.652
14	0.64
15	0.73
16	0.64
17	0.629
18	0.708

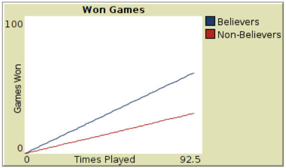
whites.csv

1	0.315
2	0.303
3	0.247
4	0.326
5	0.292
6	0.393
7	0.348
8	0.393
9	0.348
10	0.292
11	0.337
12	0.258
13	0.371
14	0.27
15	0.371
16	0.292
17	0.416
18	0.292



Darker turtles switch doors more often.
Lighter turtles switch doors less often.

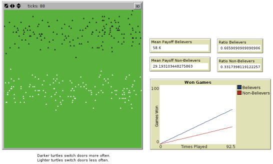
Mean Payoff Believers	58.6
Ratio Believers	0.6659090909090906
Mean Payoff Non-Believers	29.193103448275863
Ratio Non-Believers	0.3317398119122257

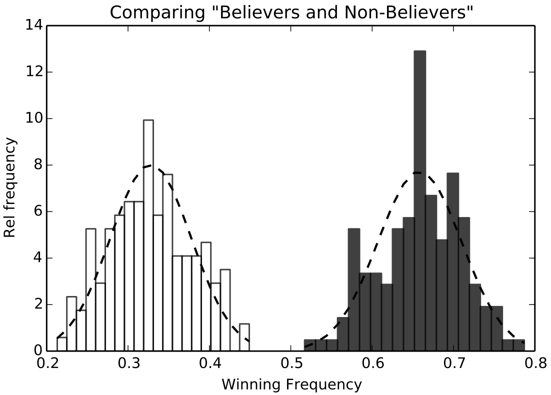


Explorative Datenanalyse

Institute for Software & Systems Engineering

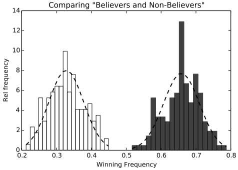
Statistische Kenngrößen + Visualisierung






```
Mean whites:  0.328
Std whites:  0.049
N whites:  145
----------------
Mean blacks:  0.658
Std blacks:  0.052
N blacks:  155
```

Explorative Datenanalyse – Code

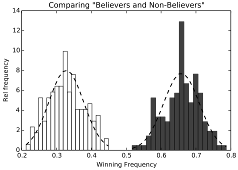



```
# read sample data (obtained from Netlogo runs)
whites = np.genfromtxt("whites.csv", dtype=float)
blacks = np.genfromtxt("blacks.csv", dtype=float)

n, bins, patches = P.hist(whites, 20, normed=1, histtype='bar')
P.setp(patches, 'facecolor', 'w', 'alpha', 0.75)
```

```
# try and fit a Gaussian distribution
mu = np.mean(whites)
sigma = np.std(whites)

y = P.normpdf( bins, mu, sigma)
l = P.plot(bins, y, 'k--', linewidth=1.5)
```

Explorative Datenanalyse – Code

Institute for Software & Systems Engineering




```
# read sample data (obtained from Netlogo runs)
whites = np.genfromtxt("whites.csv", dtype=float)
blacks = np.genfromtxt("blacks.csv", dtype=float)

n, bins, patches = P.hist(whites, 20, normed=1, histtype='bar')
P.setp(patches, 'facecolor', 'w', 'alpha', 0.75)
```

```
# try and fit a Gaussian distribution
mu = np.mean(whites)
sigma = np.std(whites)

y = P.normpdf( bins, mu, sigma)
l = P.plot(bins, y, 'k--', linewidth=1.5)
```

Student t-test

Annahmen für den *unabhängigen, 2-Stichproben T-Test*

- Stichproben haben gleiche Varianz (spezielle Varianztests)
- Bei uns: `0.049` vs. `0.052` ✓
- Stichprobengröße kann variieren ✓

Berechne Testgröße t:

$$t = \frac{\bar{X} - \bar{Y}}{s_{XY} \cdot \sqrt{\frac{1}{N_X} + \frac{1}{N_Y}}}$$

wobei

$$s_{XY} = \sqrt{\frac{(N_X - 1)s_X^2 + (N_Y - 1)s_Y^2}{N_X + N_Y - 2}}$$

Freiheitsgrade: $df = N_X + N_Y - 2$

Im Beispiel

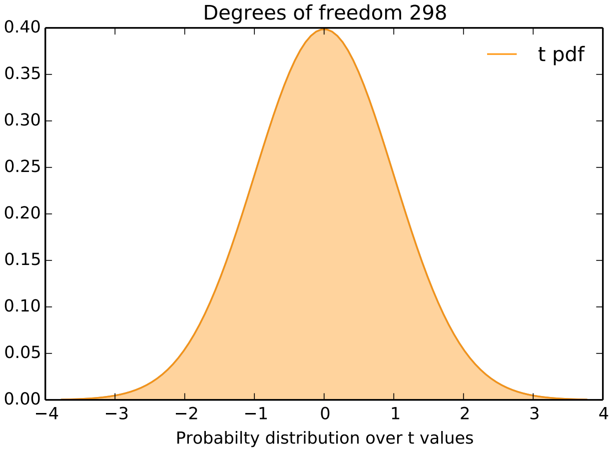
Institute for Software & Systems Engineering

```
# manual computation of student t-test
mu1 = np.mean(whites)
mu2 = np.mean(blacks)
var1 = np.var(whites, ddof=1)
var2 = np.var(blacks, ddof=1)
N1 = len(whites)
N2 = len(blacks)

sxy = np.sqrt ( ( (N1 - 1)*var1 + (N2 - 1)*var2 ) / float(N1 + N2 - 2) )
t = np.divide ((mu1-mu2), (sxy * np.sqrt(1./N1 + 1./N2) ))
df = N1 + N2 - 2
```

```
t = -56.2426699188
df = 298
```

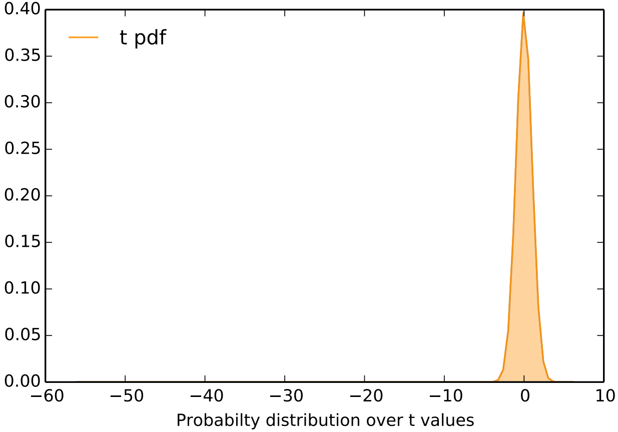
Student t-distribution

Institute for
Software & Systems
Engineering



t = -56.2426???

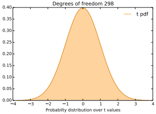
Student t-distribution



t = -56.2426!

$p = 9.9224 \cdot 10^{-161}$

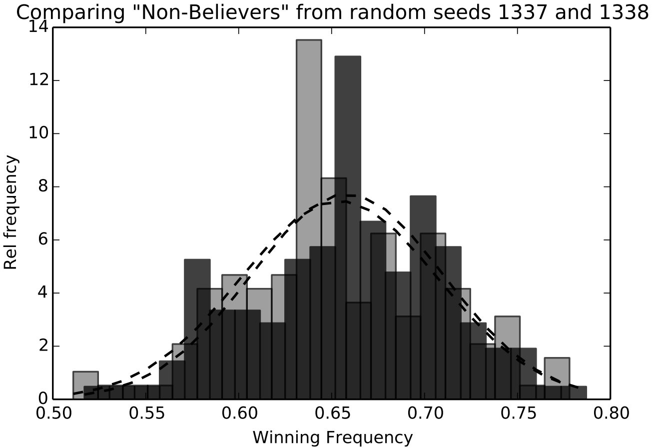
Student t-distribution: Code




```
# use np.abs to get upper tail
p = st.distributions.t.sf(np.abs(t), df) * 2
print "Probability of sample outcome by chance: ", p

alpha = 0.05
if p < alpha:
    print "Significant"
else:
    print "Not significant"
```

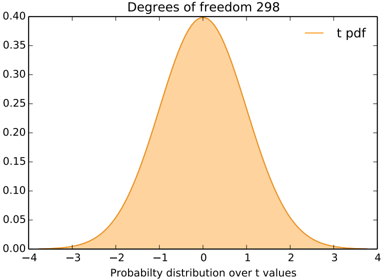
```
Probability of sample outcome by chance:  9.92242823716e-161
Significant
```

Comparing two samples from the same group



Test statistics

```
t = -0.7179
df = 298
```




```
p =  0.473345631582
Not significant
```

Basic usage

Institute for Software & Systems Engineering

```
# read sample data (obtained from Netlogo runs)
whites = np.genfromtxt("whites.csv", dtype=float)
blacks = np.genfromtxt("blacks.csv", dtype=float)

# now for statistical testing
[t, prob] = st.ttest_ind(whites, blacks)

if prob < 0.01:
    print ' SIGNIFICANT t=', t, ' prob = ', prob
else:
    print ' insignificant t=', t, ' prob = ', prob
```

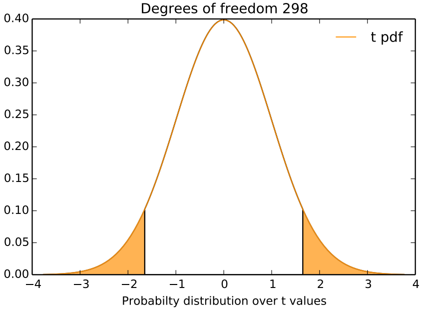
A few reminders ...

- Abhängige Variablen in Algorithmen → *instrumentierter Code*

```
while ( not converged() ) {
  // instrumentation
  if (doProtocol)
    protocolIteration();

  // do actual work
}
```

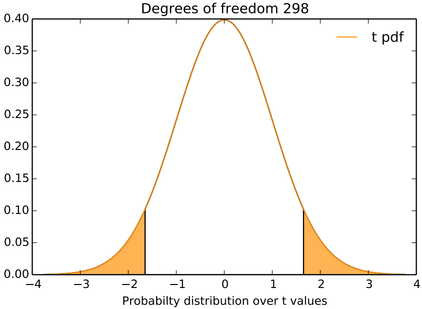
- Test muss passend zu den Annahmen (Varianz, Verteilung, Typ der Attribute) sein
  - Man-Whitney-U Test
  - Wilcoxon Test
  - ...
- Was tun wenn wir $H_0 : \mu_1 \leq \mu_2$ widerlegen wollen? (also zeigen, dass $\mu_2 > \mu_1$, nicht nur $\mu_1 \neq \mu_2$.

Student t-distribution two-tailed



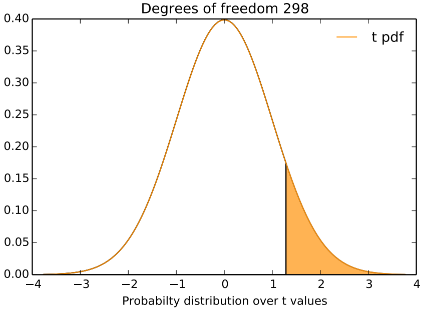

```
[t, prob] = st.ttest_ind(whites, blacks)

alpha = 0.1
if prob < alpha:
   print ' SIGNIFICANT t=', t, ' prob = ', prob
else:
   print ' insignificant t=', t, ' prob = ', prob
```

Student t 2-tailed: Behind the scenes




```
[t, prob] = st.ttest_ind(whites, blacks)

# prob denotes the probability of getting
# a value as extreme as t!
def _ttest_finish(df, t):
    """Common code between all 3 t-test functions."""
    prob = distributions.t.sf(np.abs(t), df) * 2 # use np.abs to get upper tail
```

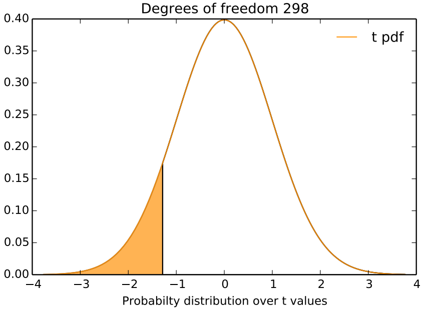
Student t-distribution one-tailed




```
[t, prob] = st.ttest_ind(whites, blacks)

alpha = 0.1
if prob/2 < alpha and t > 0:
    print ' SIGNIFICANT t=', t, ' prob = ', prob
else:
    print ' insignificant t=', t, ' prob = ', prob
```

Student t-distribution one-tailed

Institute for Software & Systems Engineering




```
[t, prob] = st.ttest_ind(whites, blacks)

alpha = 0.1
if prob/2 < alpha and t < 0:
    print ' SIGNIFICANT t=', t, ' prob = ', prob
else:
    print ' insignificant t=', t, ' prob = ', prob
```

Varianzreduktion

Beispiel: Generiere zufälliges Array der Länge n mit Permutationen von $\langle 0, \ldots, n-1 \rangle$ und führe m Versuche aus (z.B. $n = 5$, $m = 4$)

Algorithmus A

4	3	2	1	0
4	2	1	3	0
0	1	2	3	4
2	0	1	3	4

$\mu = 4.7$, $\sigma = 4.0$

Algorithmus B

0	1	2	3	4
0	2	1	3	4
0	4	2	1	3
1	0	3	2	4

$\mu = 3.4$, $\sigma = 2.0$

Idee: Eliminiere Varianz durch zufällige Probleminstanzen! Teste Differenz $|\mu_A - \mu_B|$ auf Signifikanz $\rightarrow$ *paired t-test*

Varianzreduktion

Beispiel: Generiere zufälliges Array der Länge n mit Permutationen von $\langle 0, \dots, n-1 \rangle$ und führe m Versuche aus (z.B. $n = 5$, $m = 4$) BESSER

Algorithmus A

4	3	2	1	0

0	2	1	3	4

0	1	2	3	4

2	0	1	3	4

$\mu_A = 3.6$, $\sigma_A = 2.0$

Algorithmus B

4	3	2	1	0

0	2	1	3	4

0	1	2	3	4

2	0	1	3	4

$\mu_B = 4.6$, $\sigma_B = 1.8$

Idee: Eliminiere Varianz durch zufällige Probleminstanzen! Teste Differenz $|\mu_A - \mu_B|$ auf Signifikanz $\rightarrow$ *paired t-test*

Institute for Software & Systems Engineering

"You blew it, and you blew it big! Since you seem to have difficulty grasping the basic principle at work here, I'll explain. After the host reveals a goat, you now have a one-in-two chance of being correct. Whether you change your selection or not, the odds are the same. There is enough mathematical illiteracy in this country, and we don't need the world's highest IQ propagating more. Shame!"

— Scott Smith, Ph.D. University of Florida (vos Savant 1990a)

Hollywood Interpretation:

`https://www.youtube.com/watch?v=Zr_xWfThjJ0`

Quellen I

DE Gregory, Lixin Gao, AL Rosenberg, and PR Cohen.
An empirical study of dynamic scheduling on rings of processors.
In *Parallel and Distributed Processing, 1996., Eighth IEEE Symposium on*, pages 470–473. IEEE, 1996.