Stærðfræði og reiknifræði – Skilaverkefni 6

Skilið fyrir kl. 22 á mánudag.

In [2]:
```
#BYRJA -- Keyrið til að frumstilla.
import numpy as np
import numpy.linalg as la
import numpy.random as npr
import scipy.stats as stat
import statsmodels.api as sm
import matplotlib.pyplot as plt
from urllib.request import urlopen
plt.rc('axes', axisbelow=True)
%matplotlib inline
# disp(x,y...) skrifar x,y... með 3 aukastöfum
def disp(*args): print(*(f'{a:.5f}' if isinstance(a,float) else a for a in args))
np.set_printoptions(precision=3, floatmode='fixed', suppress=True)
```

S6-A Vörusala

Þetta dæmi byggir á sýnidæminu í kafla [4.1.5 (https://cs.hi.is/strei/kafli04.html#innfeldi)](https://cs.hi.is/strei/kafli04.html#innfeldi). Söluverð vöru pr. einingu, kostnaðarverð pr. einingu og selt magn er gefið í þremur dálkum í skránni `http://cs.hi.is/strei/sala.txt` . Sú skrá er með línu efst með fyrirsögn, og henni þarf að sleppa í innlestrinum í lið 1.

Leiðbeiningar: Notið `urlopen` sbr. dæmi T3.5. Skoðið hjálp um `np.loadtxt` til að finna út hverng hægt er að sleppa fyrstu línunni (eða leitið aðstoðar hjá Google). Notið rökvísun (sbr. grein [2.2.9 (https://cs.hi.is/strei/kafli02.html#rokvisun)](https://cs.hi.is/strei/kafli02.html#rokvisun) til að leysa lið 3.

1. Lesið skrána inn þrjá vigra í Python
2. Reiknið í famhaldi heildarsöluverð
3. Gefið Python-skipanir til að ákvarða hve margar vörutegundir eru seldar með tapi

In [37]:
```
#A1
f = urlopen('https://cs.hi.is/strei/sala.txt')
(x,y,z) = np.loadtxt(f, skiprows=1).T
x,y,z
```

Out[37]:
```
(array([420.000, 580.000,  22.000, 210.000, 440.000, 120.000,  10.000]),
 array([430.000, 540.000,  20.000, 215.000, 435.000, 100.000,   8.000]),
 array([  7.000,   5.000, 117.000,  12.000,   7.000,  11.000,  81.000]))
```

In [59]:
```
#A2
utkoma = x*z
print('Heildarsöluverð:', sum(utkoma))
```

```
Heildarsöluverð: 16144.0
```

In [57]:
```
#A3
fjold = 0
nrvor = 0
for i in range(0, len(x)):
    if x[i]<y[i]:
        fjold+=1
        nrvor+=z[i]

print(fjold, 'Vörutegundir seldar með tapi. Alls:', nrvor, 'Vörur')
```

```
2 Vörutegundir seldar með tapi. Alls: 19.0 Vörur
```

S6-B Taylor-nálgun

Látum \$f\$ vera fallið
\$\$\notag f(x,y) = x^3 + xy + y^2 - 3x \$\$

1. [2 *stig*] Ákvarðið stigul \$f\$ og í framhaldi línulega Taylor-nálgun þess í punktinum \$a = (1,2)\$. Skrifið Python-fall `fhat` sem reiknar Taylor-nálgunina fyrir almennan vigur \$(x,y)\$. Prófið föllinn með því að reikna \$f(a_1,a_2)\$ og \$\hat{f}(a_1,a_2)\$; bæði ætti að gefa 4.
2. [2 *stig*] Búið til Python-fall sem finnur (u.þ.b.) hámarksskekkju í Taylor-nálguninni fyrir \$0.8 \leq x \leq 1.2\$ og \$1.8 \leq y \leq 2.2\$ (**Leiðbeining**: *Reiknið skekkjuna fyrir \$x\$ = 0.80, 0.81, 0.82,... og \$y\$ = 1.80, 1.81,... og finnið hámarkið*).

In [103]:
```
#B1


def f(x):
    fx = x[0]**3 + x[0]*x[1] + x[1]**2 - 3*x[0]
    return fx

def g(x):
    gx = np.array([3*x[0]**2 + x[0], 2*x[1] - x[0]])
    return gx

def fhat(x):
    a = np.array([1,2])
    fhatx = f(a) + np.dot(f(a), (x-a))
    return fhatx

a = np.array([1,2])
print(f(a))
print(fhat(a))
```

```
4
[4 4]
```

In [122]:
```
#B2

x = np.array([[0.8, 1.8], [0.9, 1.9], [1.0, 2.0], [1.1, 2.1], [1.2, 2.2]])
y = np.array([0.9, 1.9])

for i in range(0, len(x)):
    print(fhat(x[i]))

print('Hámark:', np.max(fhat(x)))
```

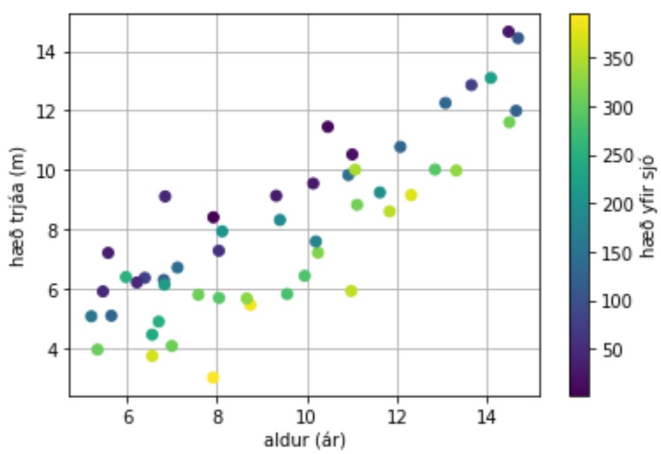
```
[3.200 3.200]
[3.600 3.600]
[4.000 4.000]
[4.400 4.400]
[4.800 4.800]
Hámark: 4.800000000000001
```

S6-C Jafna besta plans

Í reitnum hér á eftir eru búin til gervigögn sem gætu t.d. lýst hæð 5–15 ára trjáa sem fall af aldri þeirra og hæð yfir sjó (hys): trén hækka um einn metra á ári, en lækka um 1 m fyrir hverja 100 m sem þau standa hærra uppi í brekkunni:
$$ h_i = a_1 + a_2 x_i + a_3 y_i + {\varepsilon}_i\ \ (a_1 = 0.5, a_2 = 1, a_3 = -0.01) $$

þar sem h_i er hæð trés i, x_i er aldur þess og y_i er hys.

In [82]:
```
npr.seed(42)
aldur = 5 + npr.rand(50)*10
hys = npr.rand(50)*400
eps = npr.randn(50)
h = 0.5 + 1*aldur - 0.01*hys + eps;
plt.scatter(aldur, h, c=hys);
plt.xlabel('aldur (ár)')
plt.ylabel('hæð trjáa (m)')
plt.colorbar(label='hæð yfir sjó')
plt.grid()
```



Nú skal meta stika líkansins með því að beita "venjulegri aðferð minnstu kvaðrata", þ.e.a.s. með því að lágmarka kvaðratsummu frávika milli hæðar trjánna og spár líkansins. Með öðrum orðum skal lágmarka fallið:
\begin{align} S(a) &= \|Xa - h\|^2\\ &= \sum_{i=1}^{50} ((a_1 + a_2 x_i + a_3 y_i) - h_i)^2 \end{align}

þar sem X er 50×3 fylki með $x_{i1} = 1$ fyrir öll i, x í öðrum dálki og y í þeim þriðja. Til þess getum við notað Python-fallið `sm.OLS` (þar sem `sm` er `statsmodels.api` sem flutt er inn að ofan). OLS stendur fyrir *ordinary least squares*.

Maður byrjar á að setja aldur og hys inn í fylki X, t.d. með `X = np.c_[aldur, hys]` , svo þarf að bæta dálki af ásum fremst í X og það má gera með `X = sm.add_constant(X)` , svo er búið til líkan: `model = sm.OLS(hæð, X)` og að lokum eru stikarnir fundnir með `result = model.fit()` . Til að skrifa út niðurstöðuna er sagt `summ = result.summary(); print(summ)` og til að búa til vigur $a = (a_1, a_2, a_3)$ með stikunum má segja `a = result.params` .

1. [*2 stig*] Gerið allt þetta og prentið út `a` , stika fyrir besta plan
2. Hverju spáir líkanið um hæð 8 ára trés sem stendur í 150 m hæð yfir sjó?

In [96]:
```
## 1
X = np.c_[aldur, hys]
X = sm.add_constant(X)
model = sm.OLS(h, X)
result = model.fit()
summ  = result.summary();
print(summ)

a = result.params

## 2
print('________')
print('2: Hæð trés: 6.5m')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.911
Model:                            OLS   Adj. R-squared:                  0.907
Method:                 Least Squares   F-statistic:                     239.7
Date:                Mon, 24 Feb 2020   Prob (F-statistic):           2.20e-25
Time:                        12:37:22   Log-Likelihood:                -62.788
No. Observations:                  50   AIC:                             131.6
Df Residuals:                      47   BIC:                             137.3
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.8598      0.463      4.016      0.000       0.928       2.791
x1             0.8671      0.043     19.974      0.000       0.780       0.954
x2            -0.0104      0.001    -10.199      0.000      -0.012      -0.008
==============================================================================
Omnibus:                        1.624   Durbin-Watson:                   2.027
Prob(Omnibus):                  0.444   Jarque-Bera (JB):                0.982
Skew:                           0.327   Prob(JB):                        0.612
Kurtosis:                       3.211   Cond. No.                         871.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
________
2: Hæð trés: 6.5m
```

In [ ]: