In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

In [3]:
```
%matplotlib inline
```

Ex1

by: Madeleine Ekblom, Matias Jääskeläinen, Jakub Kubečka

Time used: 4h

Problem 1

In [4]:
```
iters = 1000
```

a)

sampling 100 numbers from normal distribution $N(2, 1)$ with `numpy`

In [5]:
```
x_normal = np.random.normal(2,1,iters)
```

b)

sampling 100 numbers by MH algorithm. $\mu = 2$, $\epsilon = 0.5$ and $x_0 = 4.2$

In [6]:
```
x_arr = np.zeros(iters+1)

x_arr[0] = 42
```

In [7]:
```
def logprob(x,mu):
    return -(x-mu)**2/2/np.log(np.sqrt(2*np.pi))
```

The Metropolis-Hastings algorithm

In [8]:
```
for i in range(1,iters+1):
    x_prime = np.random.normal(2,0.5,1)
    P_x_prev = logprob(x_arr[i-1],2)
    P_x_prime = logprob(x_prime,2)
    r = np.minimum(0,P_x_prime-P_x_prev)
    r_prime = np.log(np.random.uniform())
    if r_prime <= r:
        x_arr[i] = x_prime
    else:
        x_arr[i] = x_arr[i-1]
```

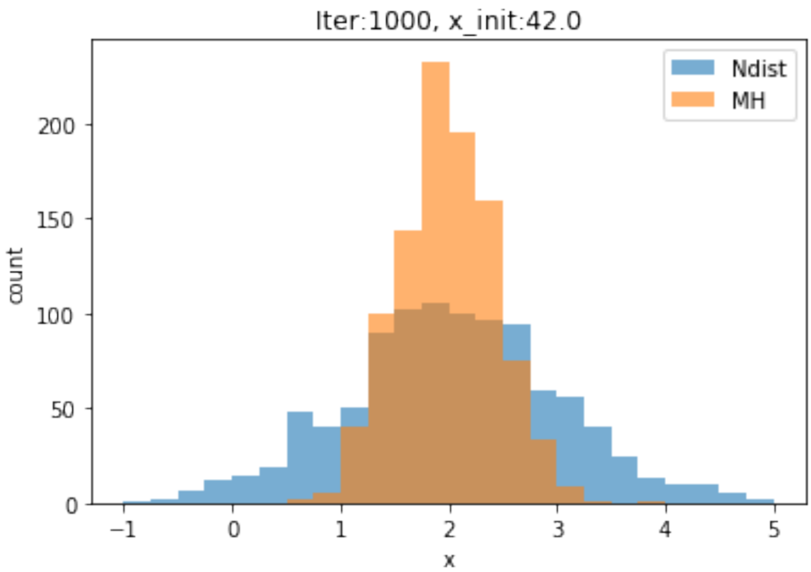
check the zero division error

c)

Plot histograms of the sampled distributions

In [9]:
```
bins = list(np.linspace(-1,5,25))
```

In [10]:
```
plt.hist(x_normal,bins=bins,label="Ndist",alpha=0.6)
plt.hist(x_arr,bins=bins,label="MH",alpha=0.6)
plt.legend()
plt.xlabel("x")
plt.ylabel("count")
plt.title("Iter:{}, x_init:{}".format(iters,x_arr[0]))
plt.savefig("iter{}x_init{}.png".format(iters,x_arr[0]))
```



1000

Update pictures

Problem 2

a)

In [11]:
```
!head -n 1 npf_train_full.csv
```

id,date,event,partlybad,HYY_META.CO2168.mean,HYY_META.CO2168.std,HYY_META.CO2336.mean,HYY_META.CO2336.std,HYY_META.CO242.mean,HYY_META.CO242.std,HYY_META.CO2504.mean,HYY_META.CO2504.std,HYY_META.Glob.mean,HYY_META.Glob.std,HYY_META.H2O168.mean,HYY_META.H2O168.std,HYY_META.H2O336.mean,HYY_META.H2O336.std,HYY_META.H2O42.mean,HYY_META.H2O42.std,HYY_META.H2O504.mean,HYY_META.H2O504.std,HYY_META.H2O672.mean,HYY_META.H2O672.std,HYY_META.H2O84.mean,HYY_META.H2O84.std,HYY_META.NET.mean,HYY_META.NET.std,HYY_META.NO168.mean,HYY_META.NO168.std,HYY_META.NO336.mean,HYY_META.NO336.std,HYY_META.NO42.mean,HYY_META.NO42.std,HYY_META.NO504.mean,HYY_META.NO504.std,HYY_META.NO672.mean,HYY_META.NO672.std,HYY_META.NO84.mean,HYY_META.NO84.std,HYY_META.NOx168.mean,HYY_META.NOx168.std,HYY_META.NOx336.mean,HYY_META.NOx336.std,HYY_META.NOx42.mean,HYY_META.NOx42.std,HYY_META.NOx504.mean,HYY_META.NOx504.std,HYY_META.NOx672.mean,HYY_META.NOx672.std,HYY_META.NOx84.mean,HYY_META.NOx84.std,HYY_META.O3168.mean,HYY_META.O3168.std,HYY_META.O342.mean,HYY_META.O342.std,HYY_META.O3504.mean,HYY_META.O3504.std,HYY_META.O3672.mean,HYY_META.O3672.std,HYY_META.O384.mean,HYY_META.O384.std,HYY_META.Pamb0.mean,HYY_META.Pamb0.std,HYY_META.PAR.mean,HYY_META.PAR.std,HYY_META.PTG.mean,HYY_META.PTG.std,HYY_META.RGlob.mean,HYY_META.RGlob.std,HYY_META.RHIRGA1250.mean,HYY_META.RHIRGA1250.std,HYY_META.RHIRGA168.mean,HYY_META.RHIRGA168.std,HYY_META.RHIRGA336.mean,HYY_META.RHIRGA336.std,HYY_META.RHIRGA42.mean,HYY_META.RHIRGA42.std,HYY_META.RHIRGA504.mean,HYY_META.RHIRGA504.std,HYY_META.RHIRGA672.mean,HYY_META.RHIRGA672.std,HYY_META.RHIRGA84.mean,HYY_META.RHIRGA84.std,HYY_META.RPAR.mean,HYY_META.RPAR.std,HYY_META.SO2168.mean,HYY_META.SO2168.std,HYY_META.SO242.mean,HYY_META.SO242.std,HYY_META.SO2504.mean,HYY_META.SO2504.std,HYY_META.SO2672.mean,HYY_META.SO2672.std,HYY_META.SO284.mean,HYY_META.SO284.std,HYY_META.SWS.mean,HYY_META.SWS.std,HYY_META.T168.mean,HYY_META.T168.std,HYY_META.T42.mean,HYY_META.T42.std,HYY_META.T504.mean,HYY_META.T504.std,HYY_META.T672.mean,HYY_META.T672.std,HYY_META.T84.mean,HYY_META.T84.std,HYY_META.UV_A.mean,HYY_META.UV_A.std,HYY_META.UV_B.mean,HYY_META.UV_B.std,HYY_META.WD168.cos.mean,HYY_META.WD168.sin.mean,HYY_META.WD504.cos.mean,HYY_META.WD504.sin.mean,HYY_META.WS168.mean,HYY_META.WS168.std,HYY_META.WS42.mean,HYY_META.WS42.std,HYY_META.WS504.mean,HYY_META.WS504.std,HYY_META.WS672.mean,HYY_META.WS672.std,HYY_META.WS84.mean,HYY_META.WS84.std,CS.mean,CS.std

In [12]:
```
!tail -n 1 npf_train_full.csv
```

```
724,2011-08-19,nonevent,FALSE,383.698145695364,8.41835109776843,38
4.052631578947,8.1303885635616,386.4368,9.90153634069176,384.10559
2105263,7.95293587620714,332.744477611941,243.722644610723,10.9465
562913907,0.96944454636499,10.7603289473684,0.97461041139059,11.32
88590604027,0.963614767005468,10.6419078947368,0.972328218065212,1
0.5509271523179,0.966246587570259,11.1830463576159,0.9725485375222
4,233.307016129032,222.197399642324,0.0576129032258064,0.081376820
0743471,0.0592356687898089,0.0922617611241827,0.040516129032258,0.
0581957461335035,0.0569871794871794,0.0872106820313916,0.056666666
6666666,0.0813462002866469,0.0396153846153846,0.0701416503699857,0
.452709677419355,0.528702540380166,0.451656050955414,0.51017197594
8581,0.409677419354838,0.443383419068627,0.423589743589744,0.52889
6536888678,0.366858974358974,0.382359068681275,0.426730769230769,0
.470985464417223,28.490641025641,6.29622249058523,25.9176923076923
,6.56013826035985,30.1044585987261,6.50199913980155,30.75826923076
92,6.44457512936383,26.9583974358974,6.4044984691251,991.398518123
67,0.463660025451013,677.851993603412,499.677061671318,-0.00299317
697228144,0.0142345400543102,47.6348614072495,26.4752376852984,NA,
NA,60.1882781456953,16.6408218050908,60.1140789473684,16.955763885
6929,63.270067114094,16.5533978098008,59.9791447368421,17.18988440
19891,60.4536423841059,17.434293656792,62.0378145695365,16.7186548
459713,30.3781130063966,20.9208567647097,0.118560767590618,0.09764
76636039375,NA,NA,NA,NA,NA,NA,NA,NA,890.5,9.3943652452232,16.46941
36460554,2.94935381333397,16.2458742004264,3.03113567713969,16.113
7886872999,2.89687999903405,15.8202454642476,2.8685144319875,16.35
23906083244,3.05512184577985,18.4752622601279,12.8384812571365,0.8
8440416221985,0.726461428742508,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,N
A,NA,NA,0.00247587872340426,0.000902461182541416
```

b)

In [13]:
```
!wc -l npf_train_full.csv
```

```
725 npf_train_full.csv
```

c)

In [2]:
```
!sed  $'s/,/\t/g' npf_train_full.csv > npf_train_full.tsv
```

for linux

```
sed 's/,/\t/g' npf_train_full.csv > npf_train_full.tsv
```

d)

In [3]:
```
!awk 'NR==1 {for(i=1;i<=NF;i++){f[$i]=i}}{if (NR>1) {print $(f["eve
nt"])}}' npf_train_full.tsv | sort -u
```

```
II
Ia
Ib
nonevent
```

Problem 3

In [16]:
```
npf_df = pd.read_csv("npf_train_full.csv")
```

In [17]:
```
npf_df.set_index("id",inplace=True)
```

Printing the first lines of the DF

In [18]:
```
npf_df.head()
```

Out[18]:

	date	event	partlybad	HYY_META.CO2168.mean	HYY_META.CO2168.std	HYY_MET/
id						
1	2000-01-23	nonevent	False	373.496585	0.189497	
2	2000-01-25	nonevent	False	381.752738	1.701439	
3	2000-02-13	nonevent	False	376.723579	0.468817	
4	2000-02-17	nonevent	False	378.600367	1.934180	
5	2000-02-23	nonevent	False	380.528120	0.802001	

5 rows × 127 columns

c)

In [19]:
```
npf_df.drop("partlybad",axis='columns',inplace=True)
```

In [20]:
```
npf_df.describe()
```

Out[20]:

	HYY_META.CO2168.mean	HYY_META.CO2168.std	HYY_META.CO2336.mean	HYY_ME
count	724.000000	724.000000	724.000000	
mean	381.658324	3.489664	381.673616	
std	11.258353	3.462871	11.221999	
min	356.526871	0.104654	356.796486	
25%	373.312402	0.960250	373.325678	
50%	380.963558	2.209440	380.973339	
75%	388.683335	4.852608	388.749373	
max	413.101159	20.960630	413.176522	

8 rows × 124 columns

Plotting scatterplot matrix

In [ ]:
```
import seaborn as sns
sns.pairplot(npf_df.iloc[:,2:12])
```

plt.figure() for i in range(2,12): plt.subplot(2,5,i-1) sns.boxplot(x="event", y=npf_df.iloc[:,i],data=npf_df)

for i in range(2,12): plt.figure() sns.boxplot(x="event", y=npf_df.iloc[:,i],data=npf_df) plt.savefig("{}.png".format(npf_df.columns[i]))

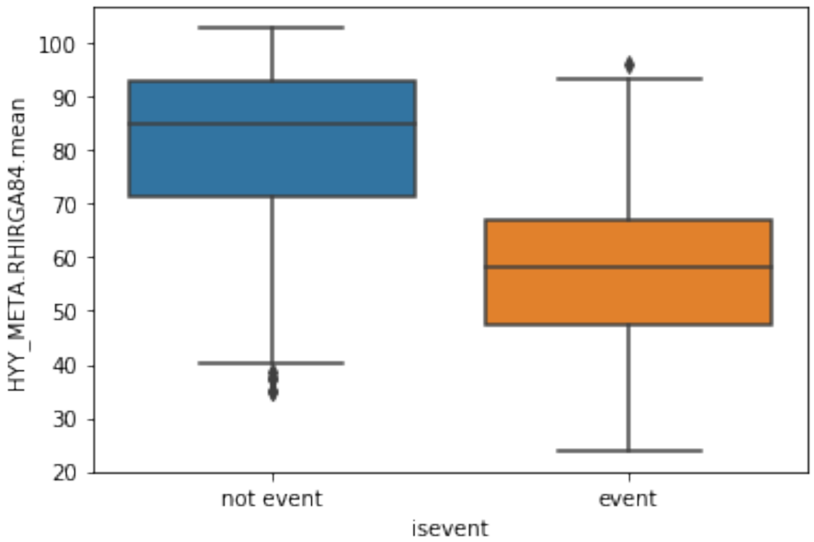
In [42]:
```
npf_df["isevent"] = npf_df.event.values != "nonevent"
```

In [43]:
```
np.sum(npf_df.isevent)
```

Out[43]: 330

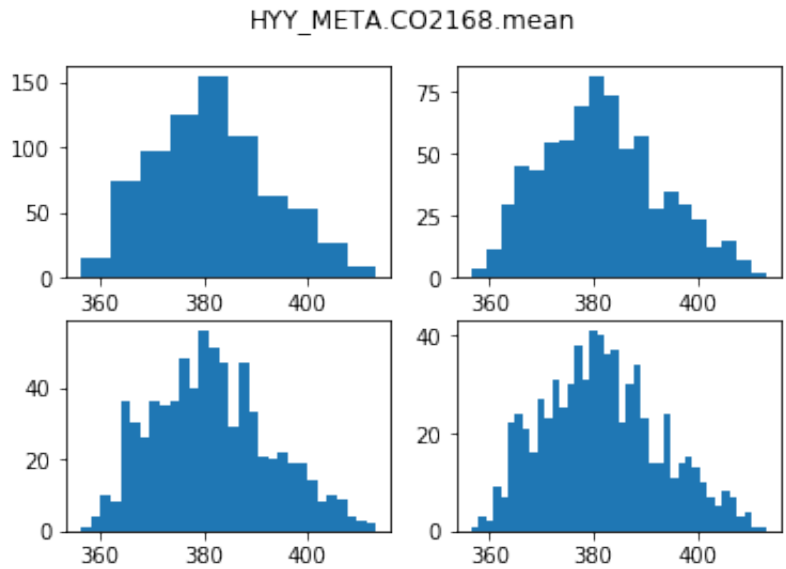
In [44]:
```
sns.boxplot(x="isevent", y=npf_df["HYY_META.RHIRGA84.mean"],data=np
f_df)
plt.xticks(ticks=[False,True],labels=["not event","event"])
```

Out[44]:
```
([<matplotlib.axis.XTick at 0x7fb7c05880d0>,
  <matplotlib.axis.XTick at 0x7fb7c05aba10>],
 <a list of 2 Text xticklabel objects>)
```



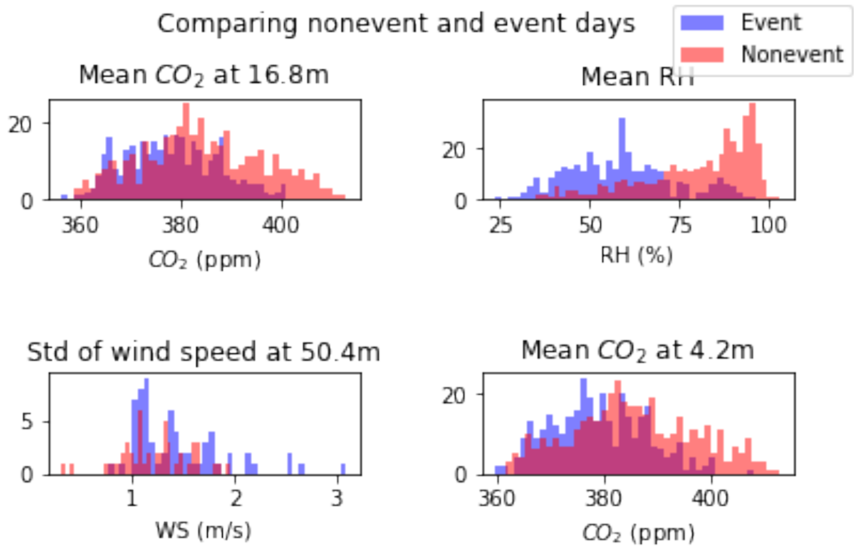
In [33]:
```
plt.figure()
#plt.subplots(2,2)
col = 2
#print(npf_df.columns[col])
plt.subplot(2,2,1)
plt.hist(x=npf_df.iloc[:,col],bins=10);#,data=npf_df)
plt.subplot(2,2,2)
plt.hist(x=npf_df.iloc[:,col],bins=20);#,data=npf_df)
plt.subplot(2,2,3)
plt.hist(x=npf_df.iloc[:,col],bins=30);#,data=npf_df)
plt.subplot(2,2,4)
plt.hist(x=npf_df.iloc[:,col],bins=40);#,data=npf_df)
plt.suptitle(npf_df.columns[col])
```

Out[33]:
```
Text(0.5, 0.98, 'HYY_META.CO2168.mean')
```



In [45]:
```
npf_df["datetime"] = pd.to_datetime(npf_df.date)
```

In [41]:
```
# Divide data into nonevent and event days
npf_df_true = npf_df[npf_df.isevent == True]
npf_df_false = npf_df[npf_df.isevent == False]


# plot subplots with different variables
fig, ax = plt.subplots()

plt.suptitle('Comparing nonevent and event days')

plt.subplot(2,2,1)
plt.hist(x=npf_df_true['HYY_META.CO2168.mean'], bins=40, color='b',
alpha=0.5, label='Event');
plt.hist(x=npf_df_false['HYY_META.CO2168.mean'], bins=40, color='r'
, alpha=0.5, label='Nonevent');
plt.xlabel('$CO_2$ (ppm)')
plt.title('Mean $CO_2$ at 16.8m')

plt.subplot(2,2,2)
plt.hist(x=npf_df_true['HYY_META.RHIRGA84.mean'], bins=40, color='b
', alpha=0.5);
plt.hist(x=npf_df_false['HYY_META.RHIRGA84.mean'], bins=40, color='
r', alpha=0.5);
plt.xlabel('RH (%)')
plt.title('Mean RH')


plt.subplot(2,2,3)
plt.hist(x=npf_df_true['HYY_META.WS504.std'], bins=40, color='b', a
lpha=0.5);
plt.hist(x=npf_df_false['HYY_META.WS504.std'], bins=40, color='r',
alpha=0.5);
plt.xlabel('WS (m/s)')
plt.title('Std of wind speed at 50.4m')

plt.subplot(2,2,4)
plt.hist(x=npf_df_true['HYY_META.CO242.mean'], bins=40, color='b',
alpha=0.5);
plt.hist(x=npf_df_false['HYY_META.CO242.mean'], bins=40, color='r',
alpha=0.5);
plt.xlabel('$CO_2$ (ppm)')
plt.title('Mean $CO_2$ at 4.2m')


fig.legend()
fig.tight_layout(pad=3.0)
```



Problem 4

First eq. to prove:

```
l_P = log(p_P)
    = log(prod_i(p_i))
    = sum_i(log(p_i))
    = sum_i(l_i)
```

Secondly eq. to prove:

```
l_S = max_j(l_j) + log{sum_i[exp(l_i-max_j(l_j))]}
    = max_j(l_j) + log{sum_i[exp(l_i)/exp(max_j(l_j))]}
    = max_j(l_j) + log{sum_i[exp(l_i)]/exp(max_j(l_j))}
    = max_j(l_j) + log{sum_i[exp(l_i)]} - log{exp(max_j(l_j))}
    = max_j(l_j) + log{sum_i[exp(l_i)]} - max_j(l_j)
    = log{sum_i[exp(l_i)]}
    = log{sum_i[exp(log(p_i))]}
    = log{sum_i[p_i]}
    = log{p_S}
```

In [9]:

```
import numpy as np
def p(x):
    return np.exp(-x**2/2)/np.sqrt(2*np.pi)
def l(x):
    return -x**2/2/np.log(np.sqrt(2*np.pi))
```

In [10]:
```
p(100)/(p(100)+p(100.01))
```

```
/Users/jakub/anaconda3/lib/python3.6/site-packages/ipykernel_launc
her.py:1: RuntimeWarning: invalid value encountered in double_scal
ars
  """Entry point for launching an IPython kernel.
```

Out[10]:
```
nan
```

Now, using the 'logaritm trick'

```
y = p(100)/(p(100)+p(100.01))
  = 1/( 1 + p(100.01)/p(100) )
  = 1/( 1 + exp(l(100.01)-l(100)) )
```

In [11]:
```
1/( 1 + np.exp(l(100.01)-l(100)) )
```

Out[11]:
```
0.7480551473673893
```