In [140]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
```

In [ ]:
```

```

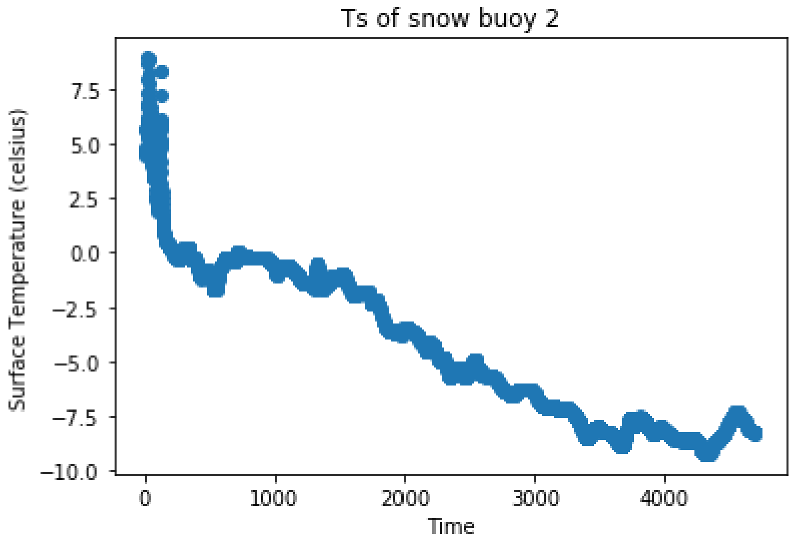
In [ ]:
```

```

In [118]:
```
df = pd.read_csv('snow_buoy2.dat', sep='\s+', skiprows=0)
bp = df[df['BP'] > 0]['BP']
ts = df['Ts']
ta = df['Ta']

print("----BP----")
print("mean", np.mean(bp))
print("std", np.std(bp))
print("var", np.var(bp))

print()
print("----Ts----")
print("mean", np.mean(ts))
print("std", np.std(ts))
print("var", np.var(ts))

print()
print("----Ta----")
print("mean", np.mean(ta))
print("std", np.std(ta))
print("var", np.var(ta))
```

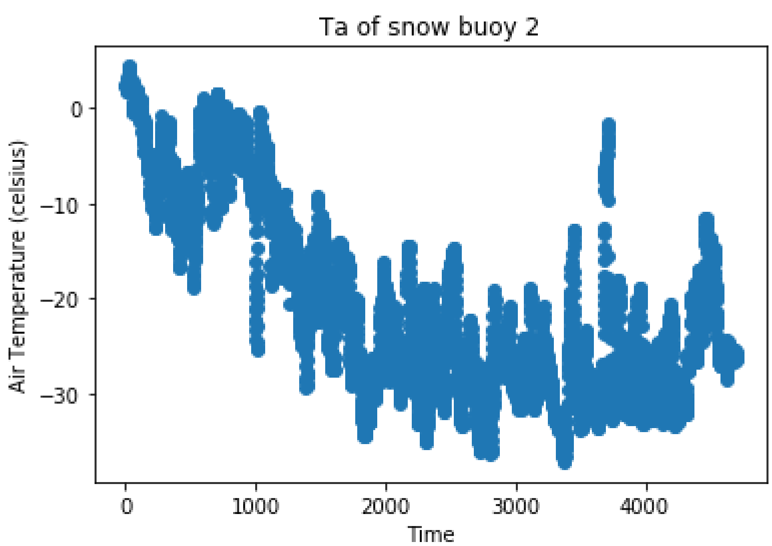
```
----BP----
mean 1012.5731432219603
std 10.622122640818933
var 112.82948939659818

----Ts----
mean -4.326175782081335
std 3.5339884481609642
var 12.489074351735141

----Ta----
mean -20.4050649074271
std 10.195306217935038
var 103.94426887746484
```

In [119]:
```
#plt.scatter(np.arange(len(bp)), bp)
#plt.title("BP of AXIB2")
#plt.ylabel("Barometric Pressure")
#plt.xlabel("Time")
#plt.show()
```

In [120]:
```
plt.scatter(np.arange(len(ts)), ts)
plt.title("Ts of snow buoy 2")
plt.ylabel("Surface Temperature (celsius)")
plt.xlabel("Time")
plt.savefig('./Ts_snow_buoy2.png')
plt.show()
```



In [121]:
```
plt.scatter(np.arange(len(ta)), ta)
plt.title("Ta of snow buoy 2")
plt.ylabel("Air Temperature (celsius)")
plt.xlabel("Time")
plt.savefig('./Ta_snow_buoy2.png')
plt.show()
```



In [ ]:

In [138]:
```
axib1 = pd.read_csv('axib1.dat', sep='\s+', skiprows=0)
axib2 = pd.read_csv('axib2.dat', sep='\s+', skiprows=0)
axib_df = pd.concat([axib1, axib2])
axib_len = len(axib_df)
axib_ts = np.array(axib_df['Ts'])
axib_ta = np.array(axib_df['Ta'])
axib_ts_var = np.var(axib_ts)
axib_ta_var = np.var(axib_ta)

iceb1 = pd.read_csv('iceb1.dat', sep='\s+', skiprows=0)
iceb2 = pd.read_csv('iceb2.dat', sep='\s+', skiprows=0)
iceb_df = pd.concat([iceb1, iceb2])
iceb_len = len(iceb_df)
iceb_ts = np.array(iceb_df['Ts'])
iceb_ta = np.array(iceb_df['Ta'])
iceb_ts_var = np.var(iceb_ts)
iceb_ta_var = np.var(iceb_ta)

svp1 = pd.read_csv('svp1.dat', sep='\s+', skiprows=0)
svp2 = pd.read_csv('svp2.dat', sep='\s+', skiprows=0)
svp_df = pd.concat([svp1, svp2])
svp_len = len(svp_df)
svp_ts = np.array(svp_df['Ts'])
svp_ta = np.array(svp_df['Ta'])
svp_ts_var = np.var(svp_ts)
svp_ta_var = np.var(svp_ta)

sb1 = pd.read_csv('snow_buoy1.dat', sep='\s+', skiprows=0)
sb2 = pd.read_csv('snow_buoy2.dat', sep='\s+', skiprows=0)
sb_df = pd.concat([sb1, sb2])
sb_len = len(sb_df)
sb_ts = np.array(sb_df['Ts'])
sb_ta = np.array(sb_df['Ta'])
sb_ts_var = np.var(sb_ts)
sb_ta_var = np.var(sb_ta)
```

In [149]:
```
# axib out
rest_df = pd.concat([iceb_df, svp_df, sb_df])
rest_len = len(rest_df)
rest_ts = np.array(rest_df['Ts'])
rest_ta = np.array(rest_df['Ta'])
rest_ts_var = np.var(rest_ts)
rest_ta_var = np.var(rest_ta)

# Ts
F = axib_ts_var / rest_ts_var
p_value = stats.f.cdf(F, axib_len-1, rest_len-1)
print("Ts p value", p_value)

# Ta
F = axib_ta_var / rest_ta_var
p_value = stats.f.cdf(F, axib_len-1, rest_len-1)
print("Ta p value", p_value)
```

```
Ts p value 0.0
Ta p value 0.6418616759304354
```

In [150]:
```
# iceb out
rest_df = pd.concat([axib_df, svp_df, sb_df])
rest_len = len(rest_df)
rest_ts = np.array(rest_df['Ts'])
rest_ta = np.array(rest_df['Ta'])
rest_ts_var = np.var(rest_ts)
rest_ta_var = np.var(rest_ta)

# Ts
F = iceb_ts_var / rest_ts_var
p_value = stats.f.cdf(F, iceb_len-1, rest_len-1)
print("Ts p value", p_value)

# Ta
F = iceb_ta_var / rest_ta_var
p_value = stats.f.cdf(F, iceb_len-1, rest_len-1)
print("Ta p value", p_value)
```

```
Ts p value 0.9999999999999999
Ta p value 0.9999999999999999
```

In [151]:
```
# svp out
rest_df = pd.concat([axib_df, iceb_df, sb_df])
rest_len = len(rest_df)
rest_ts = np.array(rest_df['Ts'])
rest_ta = np.array(rest_df['Ta'])
rest_ts_var = np.var(rest_ts)
rest_ta_var = np.var(rest_ta)

# Ts
F = svp_ts_var / rest_ts_var
p_value = stats.f.cdf(F, svp_len-1, rest_len-1)
print("Ts p value", p_value)

# Ta
F = svp_ta_var / rest_ta_var
p_value = stats.f.cdf(F, svp_len-1, rest_len-1)
print("Ta p value", p_value)
```

```
Ts p value 2.385506506402608e-290
Ta p value 0.0
```

In [152]:
```
# sb out
rest_df = pd.concat([axib_df, iceb_df, svp_df])
rest_len = len(rest_df)
rest_ts = np.array(rest_df['Ts'])
rest_ta = np.array(rest_df['Ta'])
rest_ts_var = np.var(rest_ts)
rest_ta_var = np.var(rest_ta)

# Ts
F = sb_ts_var / rest_ts_var
p_value = stats.f.cdf(F, sb_len-1, rest_len-1)
print("Ts p value", p_value)

# Ta
F = sb_ta_var / rest_ta_var
p_value = stats.f.cdf(F, sb_len-1, rest_len-1)
print("Ta p value", p_value)
```

```
Ts p value 0.0
Ta p value 0.0
```

In [157]:
```
sb_len-1
sb_ta_var
```

Out[157]: 109.06747144253855

In [158]:
```
rest_len-1
rest_ta_var
```

Out[158]: 633.0345216243048

In [ ]:
```
```

Code

Case Study 4

Import Data and install/load relevant packages

Hide

```
data = read.table("gauge.txt", head=T)
data = data[order(data['density']),]
library(L1pack) #Library for LAD = Least Absolute Deviations Regression Line
library(quantreg) #Library for Quantile Regressio n line
```

Scenario 1: Fitting

Use the data to fit the gain, or a transformation of gain, to density. Try sketching the least squares line on a scatter plot.

Do the residuals indicate any problems with the fit?

Hide

```
title = "Scatter plot of the Density and Gain"
y.axis = "Gain"
x.axis = expression('Density (g/cm'^3*')')
plot(data, main= title, pch=1, cex=1, col="orangered1", ylab=y.axis, xlab=x.axis) #Scatt
er plot of the data
fit = lm(gain~density, data) #Calculate the Least Squares Regression line for the given
 data
abline(fit, col="blue1") #Try to fit the Least Squares Regression line to the data
legend('topright', legend=c('Least Square Regression Line', 'Data'), col=c("blue1","oran
gered1"), lty=1)
```

Since the data is looking like that of an exponential distribution, we will take the log of the gain variables to achieve a linear graph (to satisfy Least Squares Line condition). Moreover the data does not follow the line drawn across the graph.

Hide

```
title1 = "Scatter plot of Density and log(Gain)"
y.axis1 = "log(Gain)"
log.data = data.frame(data$density, log(data$gain)) #Create a new data set where the we
 take the log of the gain variables
plot(log.data, main=title1, ylab=y.axis1, xlab=x.axis, pch=1, cex=1, col="darkorange")
fit.log = lm(log.data.gain.~data.density, log.data)
abline(fit.log, col="blue1")
legend('topright', legend=c('Least Square Regression Line', 'Data'), col=c("blue1","dark
orange"), lty=1)
```

Since there are replicated gain measurements, we will average thereplicate measurements to create a single variable that corresponds to the variables (explanatory and response variables need to be linear to meet the conditions required for Least Squares Line). Notice that the data fits to the regression line.

Hide

```
title2 = "Scatter plot of the average of Density and log(Gain)"
mean.log.data = aggregate.data.frame(list(gain=log.data$log.data.gain.),list(density=lo
g.data$data.density), FUN=mean)
plot(mean.log.data, main=title2, xlab=x.axis, ylab=y.axis1, pch=1, cex=1, col="darkgolde
nrod1")
fit.mean.log = lm(gain~density, mean.log.data)
abline(fit.mean.log, col="blue1")
legend('topright', legend=c('Least Square Regression Line', 'Data'), col=c("blue1","dark
orange"), lty=1)
```

Perform least squares regression to fit the line to the data

Hide

```
LS = lm(gain~density, data=mean.log.data) #Least Square regression
LAD = lad(gain~density, data=mean.log.data) #Least Absolute Deviations Regression
QRL = rq(gain~density, data=mean.log.data) #Quantile Regression Line
plot(log.data, main=title, xlab=x.axis, ylab=y.axis1)
legend("topright", legend=c('Least Squares', 'Least Absolute Deviation','50% Quantile'),
col=c('green','blue','red'), lty=1)
abline(LS, col="green", lwd=2) #Draw a line for Least Square Regression
abline(LAD, col="blue", lwd=2) #Draw a line for LAD Regression
abline(QRL, col="red", lwd=1) #Draw a line for 50% Quantile Regression

plot(log.data, xlim=c(0.3,0.34), ylim=c(4.2,4.8), main = title, xlab=x.axis, ylab=y.axis
1) #Zoomed in version to visualize regression lines
legend("topright", legend=c('Least Squares', 'Least Absolute Deviation','50% Quantile'),
col=c('green','blue','red'), lty=1)
abline(LS, col="green", lwd=2)
abline(LAD, col="blue", lwd=5)
abline(QRL, col="red", lwd=2)

cor = cor(mean.log.data) #Find the correlation coefficients
cor.rsq = append(cor, summary(LS)$r.squared) #A vector of the correlation coefficients a
nd r squared value

LS
LAD
QRL
cor.rsq
```

We will proceed further and check the conditions required for linear regression. Such conditions include Linearity, Normality of the residuals and Constant Variability.

Hide

```
title3 = "Residuals of log(Gain)"
LAD.Residual.gain = log.data['log.data.gain.'] - rep(predict(LAD), each=10) #Calculate f
or the LAD gain
QRL.Residual.gain = log.data['log.data.gain.'] - rep(predict(QRL), each=10) #Calculate f
or the QRL gain

LAD.Residual = data.frame(log.data['data.density'], LAD.Residual.gain)
QRL.Residual = data.frame(log.data['data.density'], QRL.Residual.gain)

plot(fit.log$residuals*-1, main='Residual plot (Least Squares)', ylab='density', col='gr
een4') #Residual plot of Least Sqaures
abline(0,0,col='red') #A fitted line to the residual plot
plot(LAD.Residual$log.data.gain.*-1, main='Residual plot (LAD)', ylab='density', col='bl
ue2') #Residual plot of LAD
abline(0,0,col='red') #A fitted line to the residual plot
plot(QRL.Residual$log.data.gain.*-1, main='Residual plot (Quantile)', ylab='density', co
l='darkmagenta') #Residual plot of Quantile Regression
abline(0,0,col='red') #A fitted line to the residual plot



hist((fit.log$residuals*-1),xlim=c(-0.2,0.3), col='coral3', xlab='density', main='Histog
ram: Least Squares Regression Line', breaks=15) #Histogram of the least sqaures plot is
 normal
qqnorm(fit.log$residuals*-1, col='burlywood4') #qqplot of the Least Squares residual plo
t
qqline(fit.log$residuals, col='red1')

hist(LAD.Residual$log.data.gain.*-1, xlim=c(-0.2,0.3), xlab='density', main='Histogram:
 LAD Regression Line', breaks=15, col='darkgoldenrod3') #Histogram of the LAD is normal
qqnorm(LAD.Residual$log.data.gain.*-1, col='darkolivegreen') #qqplot of the LAD residual
plot
qqline(LAD.Residual$log.data.gain., col='red1')

hist(QRL.Residual$log.data.gain.*-1, xlim=c(-0.2,0.3), xlab='density', main='Histogram:
 LAD Regression Line', breaks=15, col='darkblue') #Histogram of the LAD is normal
qqnorm(QRL.Residual$log.data.gain.*-1, col='blueviolet') #qqplot of the QRL residual plo
t
qqline(QRL.Residual$log.data.gain., col='red1')



plot(fit.log$residuals*-1, ylim=c(-.6,.6), main= 'Adjusted Residual plot (Least Square
s)', ylab='density', col='green4') #Check for constant Variability with an adjusted plot
for Least Squares
abline(0,0,col='red')
plot(LAD.Residual$log.data.gain.*-1, ylim=c(-.6,.6), main='Adjusted Residual plot of LAD
Line', ylab='density', col='blue2') #Check for constant Variability with an adjusted plo
t for LAD
abline(0,0,col='red') #A fitted line to the residual plot
plot(QRL.Residual$log.data.gain.*-1, ylim=c(-.6,.6), main='Adjusted Residual plt of Quan
tile Regression Line', ylab='density', col='darkmagenta') #Check for constant Variabilit
```

```
y with an adjusted plot for Quantile Regression
abline(0,0,col='red') #A fitted line to the residual plot
```

Scenario 3: Cross Validation

To check how well your procecure works, omit the set of measurements corresponding to the block of density 0.508, apply your “estimation”/calibration procedure to the remaining data, and provide an interval estimate for the density of a block with an average reading of 38.6.

Where does the actual density fall in the interval? Try the same test, for the set of measurements at the 0.001 density.

Hide

```
omit.log.data = log.data[-c(1:10),] #omit data
omit.mean.log.data = mean.log.data[-1,] #omit data

New.LS = lm(gain~density, omit.mean.log.data) #Find the Least Squares Regression Line fo
r the new data
avg.density = mean(omit.mean.log.data$density) #Find the mean of the density for the new
data
sigma = sum((omit.mean.log.data$density - avg.density)^2)

plot(omit.log.data, main='Scatter plot of the Density and Gain with omitted data', xlab=
x.axis, ylab=y.axis1)
abline(LS, col='red1')
legend('topright', legend=c('Least Squares Regression Line'), col=c('red'), lwd=1)

s2 = aggregate(list(var=LS.Residual$LS.Residual.gain), by=list(density=LS.Residual$data.
density), FUN=var)
s =sqrt(mean(s2$var))

bounds = 10*s*sqrt((1/9-1) + [(density-avg.density)^2/sigma)]

PredictLogGain <- function(density)
predict(LS, data.frame(density=density))
```

Scenario3

Code

dog

3/16/2019

ok

ok

Hide

```
library(quantreg)
library(L1pack)
library(ggplot2)
```

Hide

```
data <- read.table('gauge.txt', header=TRUE)
sorted_data <- data[order(data$density), ]  # Sort from least to greatest density
```

Hide

```
title <- 'Gain vs Snow Density'
ylab <- expression('Snow Density (g/cm'^3*')')
xlab <- 'Gain'

plot(x=sorted_data$gain, y=sorted_data$density, main=title, xlab=xlab, ylab=ylab, pch=16
)
# Take log transformation of response variable (gain)

xlab = 'log(Gain)'
title <- 'log(Gain) vs. Density'

log_data = data.frame(log(data['gain']), data['density'])

plot(log_data, main=title, xlab=xlab, ylab=ylab, pch=16) #ylim=c(2.3, max(log_data)), pc
h=16)
```

Hide

```
# Average
title = "Average log(Gain) vs Snow Density"

# Partition by the densities...
log_data.avg = aggregate(list(gain=log_data$gain),
                         by=list(density=log_data$density),
                         FUN=mean)

avg_gain <- log_data.avg$gain
avg_density <- log_data.avg$density

plot(x=avg_gain, y=avg_density, main=title, xlab=xlab, ylab=ylab, pch=16)
```

Hide

```
# Predictions
ls <- lm(density~gain, data=log_data.avg)
lad <- lad(density~gain, data=log_data.avg)
quant <- rq(density~gain, tau=.5, data=log_data.avg)

# Get Intercepts
ls_i <- ls$coefficients[1]
ls_s <- ls$coefficients[2]
lad_i <- lad$coefficients[1]
lad_s <- lad$coefficients[2]
q_i <- quant$coefficients[1]
q_s <- quant$coefficients[2]
```

Hide

```
getLSLogDensity <- function(gain) {
  return (predict(ls, data.frame(gain=gain)))
}

getLSpred <- function(density) {
  return((density - ls_i) / ls_s)
}

getLADpred <- function(density) {
  return((density - lad_i) / lad_s)
}

getQuantPred <- function(density) {
  return((density - q_i) / q_s)
}
```

Hide

```
# 95% prediction and confidence intervals of log(gain) using density
t <- qt(.975, df=3-2)
gain.mean <- mean(log_data.avg$gain)
sum_ <- sum((log_data.avg$gain - gain.mean) ^ 2)

s2 <- aggregate( list(variance=least.squares.residuals$gain),
                 by=list(density=least.squares.residuals$density),
                 FUN=var)
var <- s2$variance
s <- sqrt(mean(var))

getCIwidth <- function(density) {
  return (t * s * sqrt(1/m + (density-gain.mean)^2 / summation))
}

getPIwidth <- function(density) {
  return (t * s * sqrt(1 + 1/m + (density-gain.mean)^2 / summation))
}

getCiBounds <- function(density) {
  eval(center.expr)
  eval(ci.width.expr)
  return (c(center - width, center + width))
}

CiLoBound <- function(density) {
  return(getLSLogDensity(density) - getCIwidth(density))
}

CiUpBound <- function(density) {
  return(getLSLogDensity(density) + getCIwidth(density))
}

PiLoBound <- function(density) {
  return(getLSLogDensity(density) - getPIwidth(density))
}
PiUpBound <- function(density) {
  return(getLSLogDensity(density) + getPIwidth(density))
}
```

Hide

```
# Add bands around least squares line
plot(df.log, main=title, xlab=x.axis, ylab=y.log.axis, xlim=x.range, ylim=y.range)
abline(least.squares, col='red')

ci.col <- 'red'
pi.col <- 'blue'
symbol <- '-'
size <- 1.5
line.type <- 3
line.width <- 0.7
donky_ci <- data.frame(density=log_data.avg$density,
                                   lower=CiLoBound(log_data.avg$density),
                                   upper=CiUpBound(log_data.avg$density))
```