Regression Lab

- Can we use total length to predict a possum's head length?
- Which possum body dimensions are most correlated with age and sex?
- Can we classify a possum's sex by its body dimensions and location?
- Can we predict a possum's trapping location from its body dimensions?

Linear

In [1]:
```
#Setting up the data
import pandas as pd
from chefboost import Chefboost as chef
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
possum = pd.read_csv('/Users/lukehenry/Documents/Jupyter/Regression Lab/poss
possum = possum.drop('case', axis = 1)
possum['age'] = possum['age'].fillna(possum['age'].median())
possum['footlgth'] = possum['footlgth'].fillna(possum['footlgth'].median())
possum = possum.drop(["Pop"], axis = 1)
possum['totlngth'] = possum['totlngth'].multiply(10) # or df['col1'] * 10
```

In [2]:
```
#Used to predict Possum's head length

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from numpy import mean
from numpy import absolute
from numpy import sqrt

total_scores = []
possum['sex'] = possum['sex'].replace({'m': 0, 'f': 1})
tests= ['sex','hdlngth','site','skullw','taill','earconch','eye','chest','be
for test in tests:
    X = possum.drop(test, axis=1)
    y= possum[test]

    #define cross-validation method to use
    cv = KFold(n_splits=5, random_state=1, shuffle=True)
    #builds multiple linear regression model
    model = LinearRegression()
    #uses k-fold CV to evaluate model
    scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error',
                             cv=cv, n_jobs=-1)
    #view RMSE
    print("Ability to predict",test,":",sqrt(mean(absolute(scores))))
    total_scores.append(sqrt(mean(absolute(scores))))
accuracy = mean(total_scores)
print("Overally Accuracy of predictions:", accuracy)
    #The lower the number (RMSE), the close it is to being accurate

    #under a 0.8 is considered to be a good model prediction
    #I can't say for certain whether my model is accurate, but i can compare
```

```
Ability to predict sex : 0.6626277303251604
Ability to predict hdlngth : 1.2828907203084372
Ability to predict site : 0.9577434884481242
Ability to predict skullw : 1.2025820651707253
Ability to predict taill : 1.0046624137597613
Ability to predict earconch : 1.342253852811912
Ability to predict eye : 0.8782844082304739
Ability to predict chest : 1.0649673340049974
Ability to predict belly : 1.3446086218046462
Ability to predict footlgth : 1.34240505564545
Overally Accuracy of predictions: 1.108302569050969
```

In [3]:
```
#Used to predict which body dimensions are correlated with sex and age

import pandas as pd
from sklearn.linear_model import LinearRegression

# Load data into a Pandas dataframe
# Select the attribute you want to predict (the dependent variable) and the
tests = ['sex','age']
for test in tests:
    X = possum.drop(test, axis=1)
    y = y= possum[test]

    # Fit the model
    reg = LinearRegression().fit(X, y)

    # Get the coefficients for each independent variable
    coefficients = pd.DataFrame(reg.coef_, X.columns, columns=['Coefficient'

    # Sort the coefficients by absolute value to see which attributes are mo
    coefficients_sorted = coefficients.sort_values(by=['Coefficient'], ascer

    # Print the sorted coefficients
    print(test)
    print(coefficients_sorted)
    print("")
```

```
sex
          Coefficient
eye         -0.113417
site        -0.094986
hdlngth     -0.047379
earconch    -0.030250
chest        0.030142
age          0.019439
belly        0.011772
skullw      -0.011758
footlgth    -0.011221
taill        0.010575
totlngth     0.003569

age
          Coefficient
sex          0.301501
eye          0.272165
chest        0.134809
belly        0.121892
footlgth    -0.094266
site        -0.082311
hdlngth      0.064726
earconch     0.057297
taill        0.053719
skullw       0.034103
totlngth    -0.002482
```

In [4]:
```
#Used to classify and predict possum sex and trapping location

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
tests = ['sex','site']
for test in tests:
    X = possum.drop(test, axis=1)
    y= possum[test]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,
    knn = KNeighborsClassifier(2)

    knn.fit(X,y)
    print(test,knn.score(X_test,y_test))
    print(knn.predict(X_test))
```

```
sex 0.7692307692307693
[0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
site 0.6538461538461539
[5 1 6 4 2 1 4 6 1 4 5 3 7 1 1 5 6 5 1 5 5 6 5 2 2 2 5 4 3 2 1 1 1 5 1 1 5
 6 6 6 4 3 6 1 1 6 5 6 1 4 2 3]
```

Logistic

In [16]:
```
#Setting up the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
possum = pd.read_csv('/Users/lukehenry/Documents/Jupyter/Regression Lab/poss
possum = possum.drop('case', axis = 1)
possum['age'] = possum['age'].fillna(possum['age'].mean())
possum['footlgth'] = possum['footlgth'].fillna(possum['footlgth'].mean())
possum = possum.drop(["Pop"], axis = 1)
possum['totlngth'] = possum['totlngth'].multiply(10) # or df['col1'] * 10
possum.sex = possum.sex.map({'m': 0, 'f': 1})
```

In [6]:
```
#No way of predicting head length with Logistic Regression since hdlngth is
```

In [17]:
```
#Used to predict which body dimensions are correlated with sex

import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split

# Load data into a pandas dataframe

# Define dependent and independent variables
y = possum['sex']
X = possum.drop('sex', axis = 1)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Fit logistic regression model
model = sm.Logit(y_train, sm.add_constant(X_train)).fit()

# View regression results
print(model.summary())

# Sort coefficients in descending order
coef_df = pd.DataFrame({'coef': model.params, 'p-value': model.pvalues})
coef_df = coef_df.sort_values(by='coef', ascending=False)

# Print top 5 predictors
print(coef_df.head())
```

```
Optimization terminated successfully.
         Current function value: 0.552825
         Iterations 6
                           Logit Regression Results
==============================================================================
===
Dep. Variable:                    sex   No. Observations:
83
Model:                          Logit   Df Residuals:
71
Method:                           MLE   Df Model:
11
Date:                Thu, 23 Mar 2023   Pseudo R-squ.:                   0.1
880
Time:                        01:12:16   Log-Likelihood:                -45.
884
converged:                       True   LL-Null:                       -56.
509
Covariance Type:            nonrobust   LLR p-value:                    0.03
088
==============================================================================
===
```

```
                 coef    std err          z      P>|z|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
const         23.1150     10.993      2.103      0.035       1.569      44.
662
site          -0.5061      0.267     -1.892      0.059      -1.030       0.
018
age            0.1437      0.154      0.933      0.351      -0.158       0.
445
hdlngth       -0.2733      0.142     -1.931      0.053      -0.551       0.
004
skullw        -0.0149      0.127     -0.117      0.906      -0.264       0.
234
totlngth       0.0183      0.012      1.502      0.133      -0.006       0.
042
taill          0.0793      0.236      0.336      0.737      -0.384       0.
543
footlgth      -0.0675      0.129     -0.523      0.601      -0.320       0.
185
earconch      -0.1266      0.127     -0.999      0.318      -0.375       0.
122
eye           -0.5510      0.300     -1.835      0.066      -1.139       0.
037
chest          0.0870      0.208      0.418      0.676      -0.321       0.
495
belly          0.0500      0.130      0.385      0.700      -0.204       0.
304
==============================================================================
===
            coef   p-value
const  23.115040  0.035497
age     0.143685  0.350622
chest   0.087003  0.675705
taill   0.079348  0.737230
belly   0.050005  0.700076
```

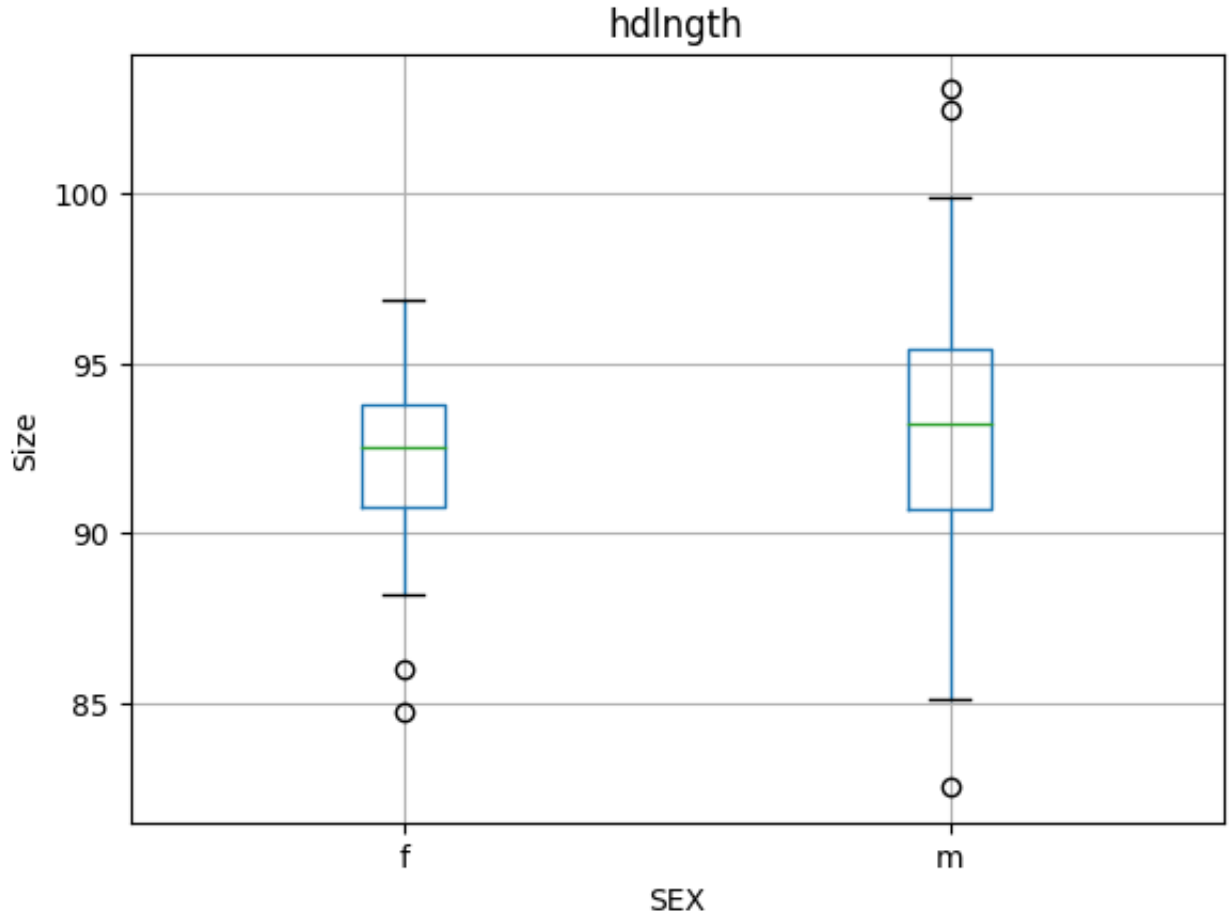
In [8]:
```
#Used to classify and predict possum sex and trapping location
tests = ['sex','site']
for test in tests:
    X = possum.drop(test, axis=1)
    y= possum[test]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,
    model = LogisticRegression()
    model.fit(X_train,y_train)
    y_pred = pd.Series(model.predict(X_test))
    y_test = y_test.reset_index(drop=True)
    print(test,"Accuracy:", metrics.accuracy_score(y_test, y_pred))
    print(test,"Precision:", metrics.precision_score(y_test, y_pred, average
    print(test,"Recall:", metrics.recall_score(y_test, y_pred, average='weig
```

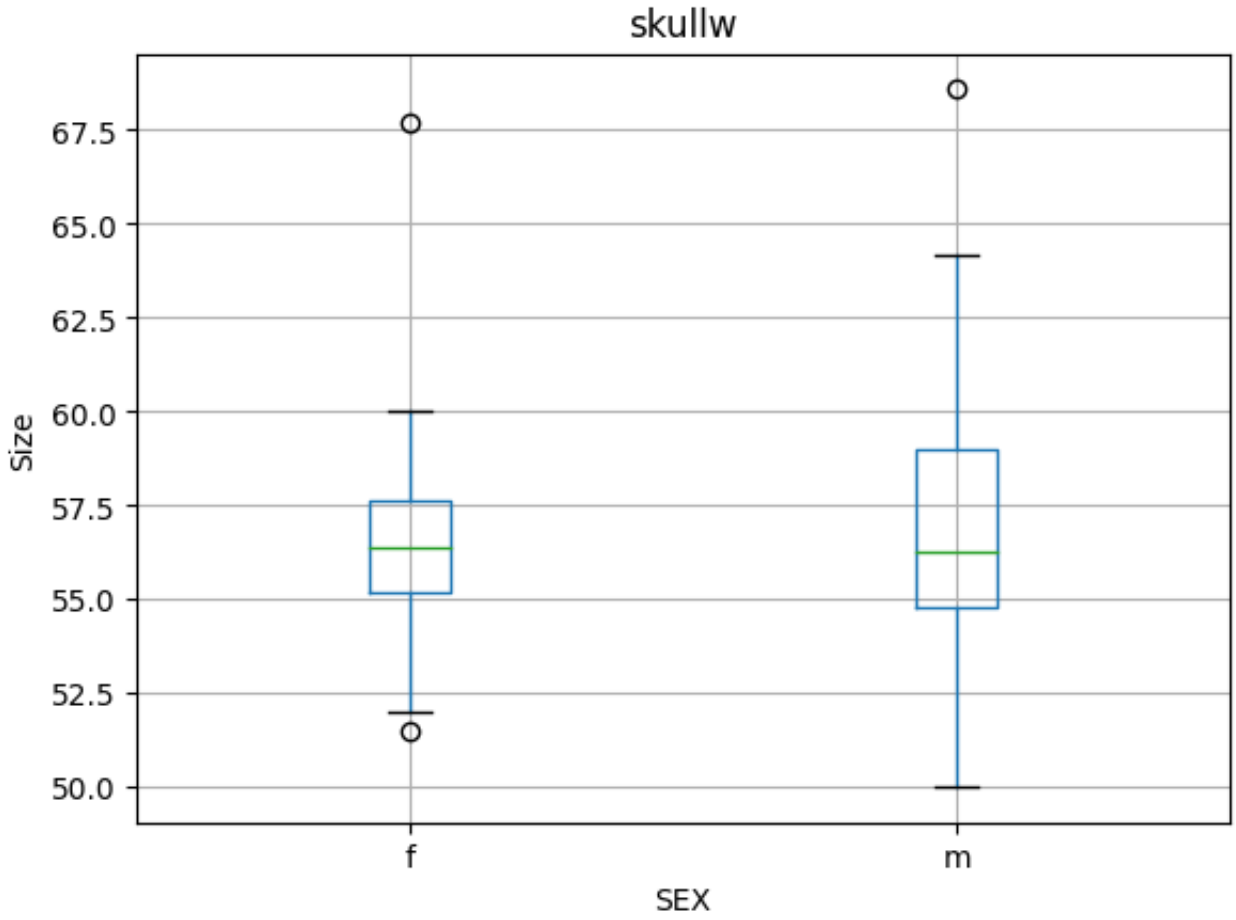
```
sex Accuracy: 0.5961538461538461
sex Precision: 0.5945701357466063
sex Recall: 0.5961538461538461
site Accuracy: 0.5384615384615384
site Precision: 0.5569331983805668
site Recall: 0.5384615384615384
```

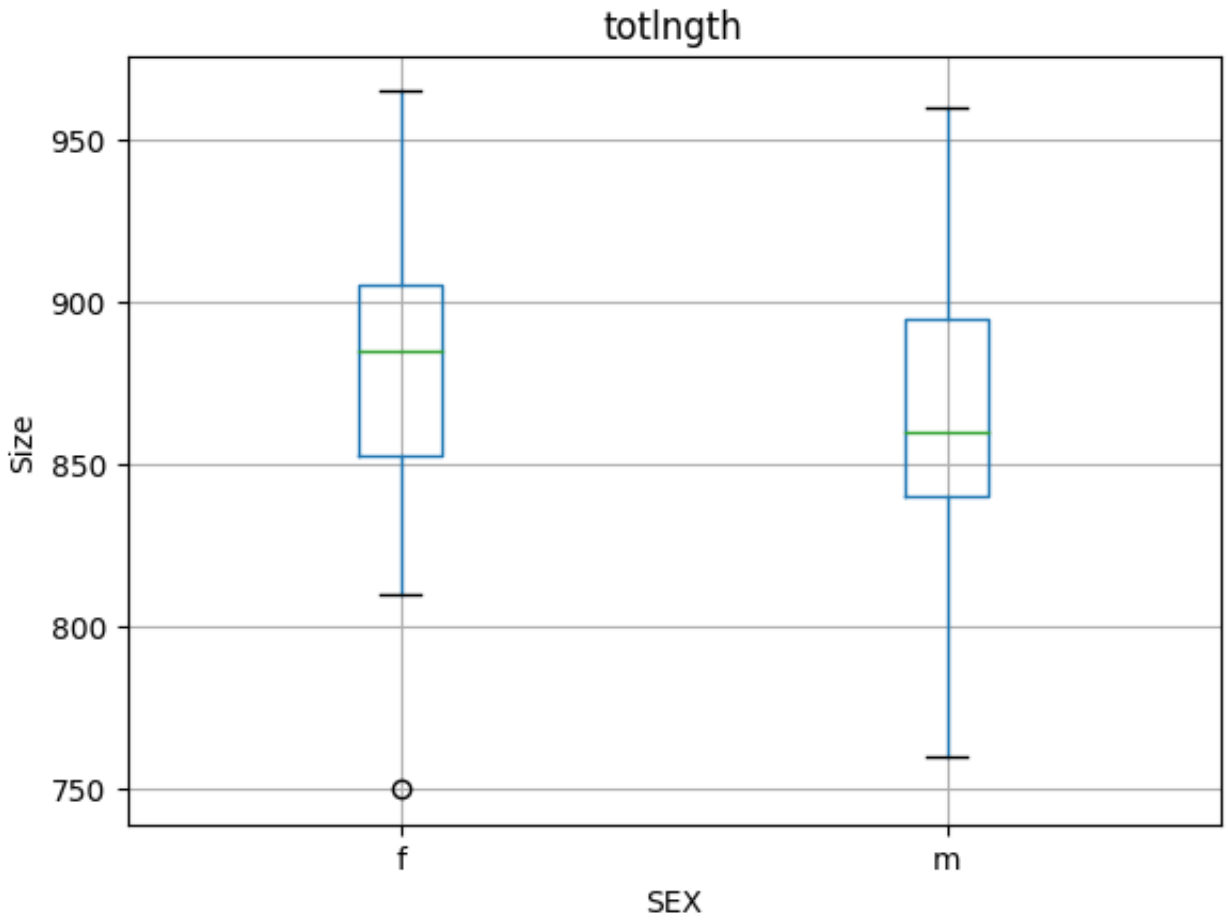
Graphs

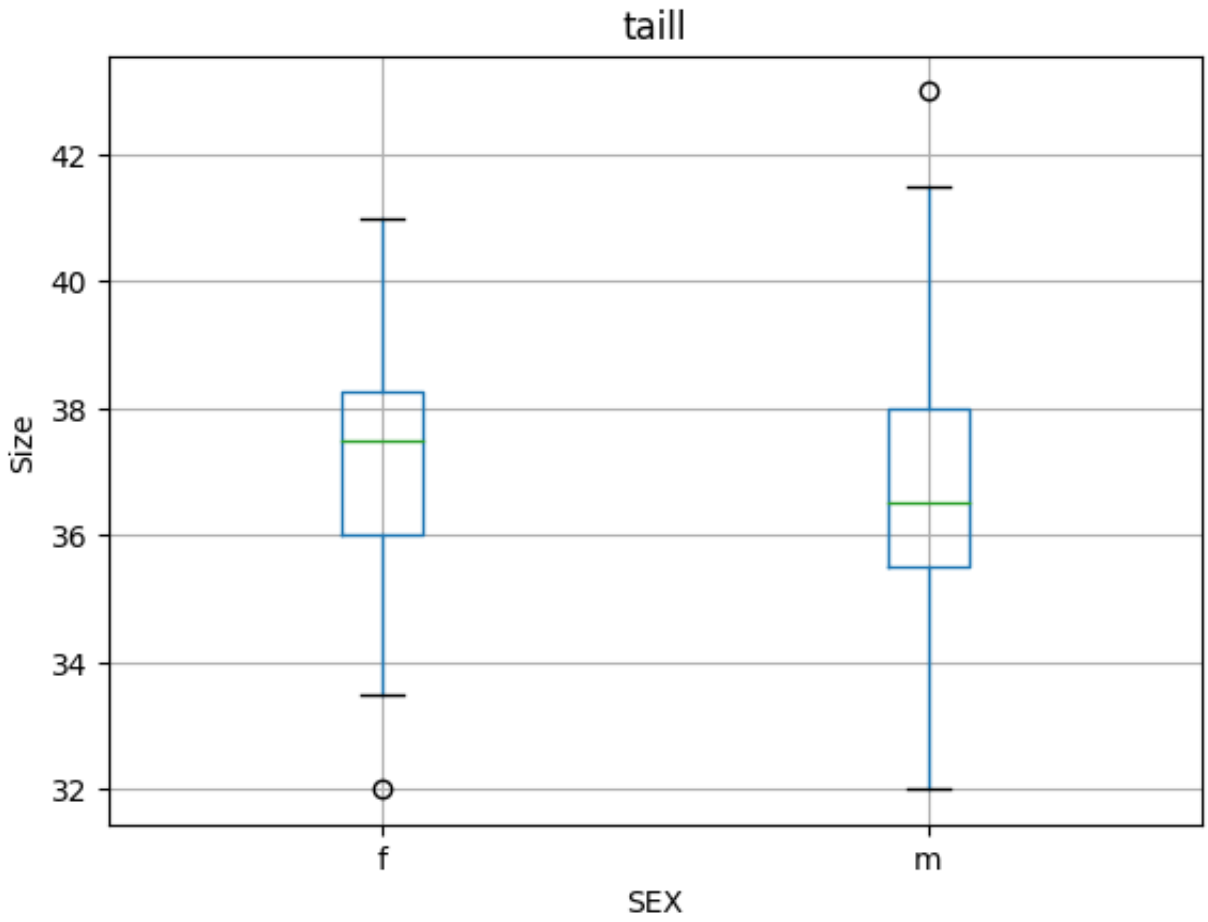
In [9]:
```
#Setting up the data
import pandas as pd
from chefboost import Chefboost as chef
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
possum = pd.read_csv('/Users/lukehenry/Documents/Jupyter/Regression Lab/poss
possum = possum.drop('case', axis = 1)
possum['age'] = possum['age'].fillna(possum['age'].median())
possum['footlgth'] = possum['footlgth'].fillna(possum['footlgth'].median())
possum = possum.drop(["Pop"], axis = 1)
possum['totlngth'] = possum['totlngth'].multiply(10) # or df['col1'] * 10
```

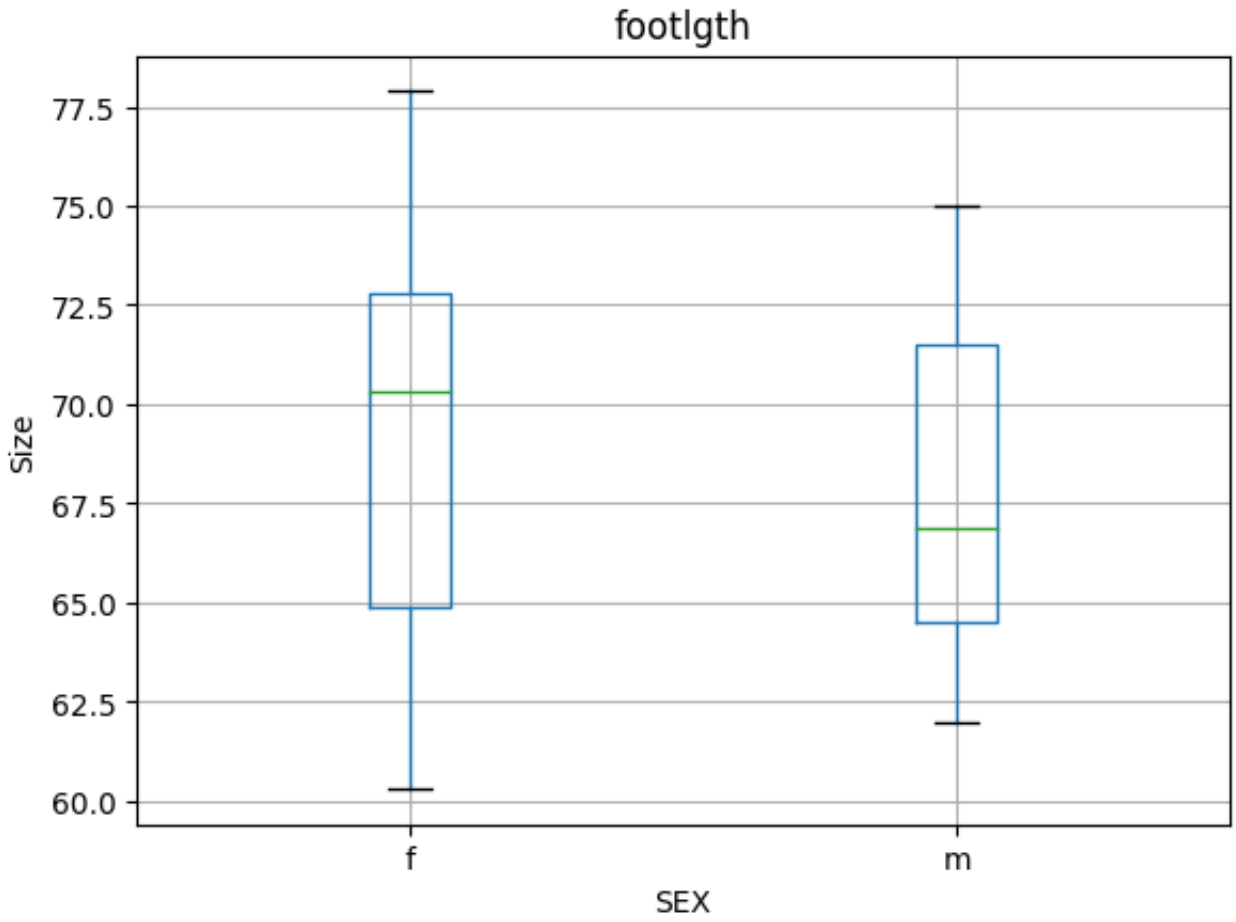
In [10]:
```
#Used to predict which body dimensions are correlated with sex and age

possumDimensions = possum[['hdlngth','skullw','totlngth','taill','footlgth',
possumSex = possum[['sex']]
possumAge = possum[['age']]
df = pd.merge(possumDimensions, possumSex, left_index=True, right_index=True
df.head()
for col in possumDimensions.columns:
    df.boxplot(column=col, by='sex')
    plt.title(col)
    plt.suptitle('')
    plt.xlabel('SEX')
    plt.ylabel('Size')
    plt.show()
df = pd.merge(possumDimensions, possumAge, left_index=True, right_index=True
for col in possumDimensions.columns:
    df.boxplot(column=col, by='age')
    plt.title(col)
    plt.suptitle('')
    plt.xlabel('AGE')
    plt.ylabel('Size')
    plt.show()
```

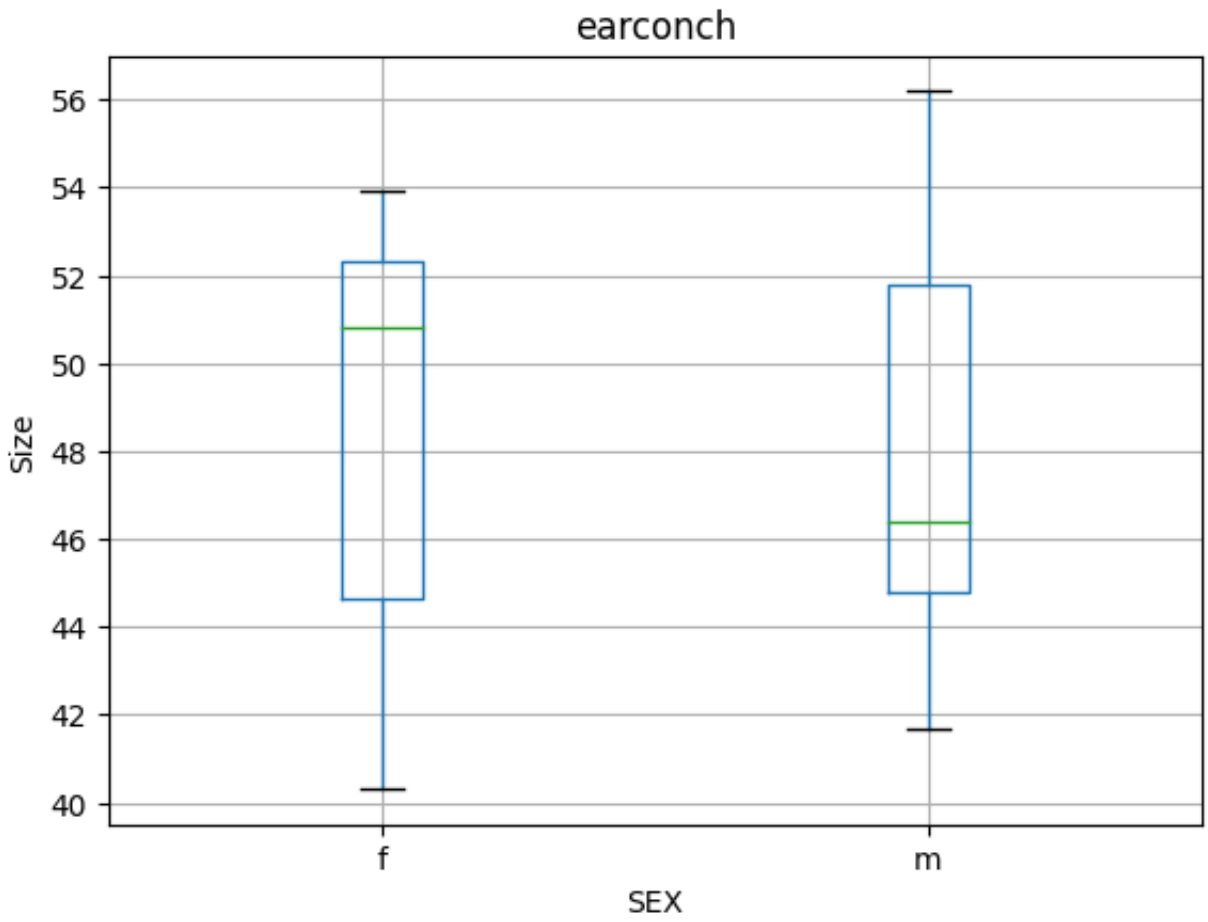


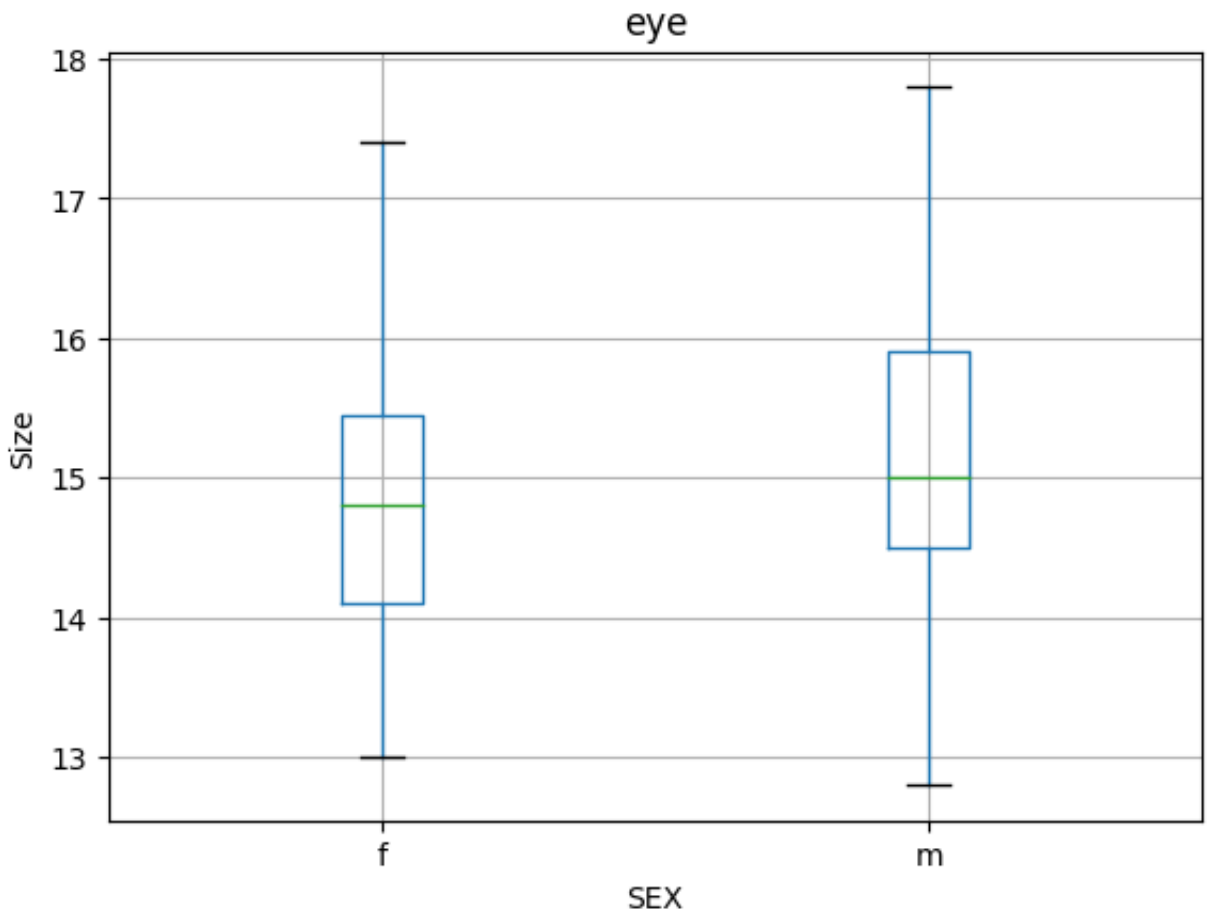

skullw
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
Size
f
m
SEX

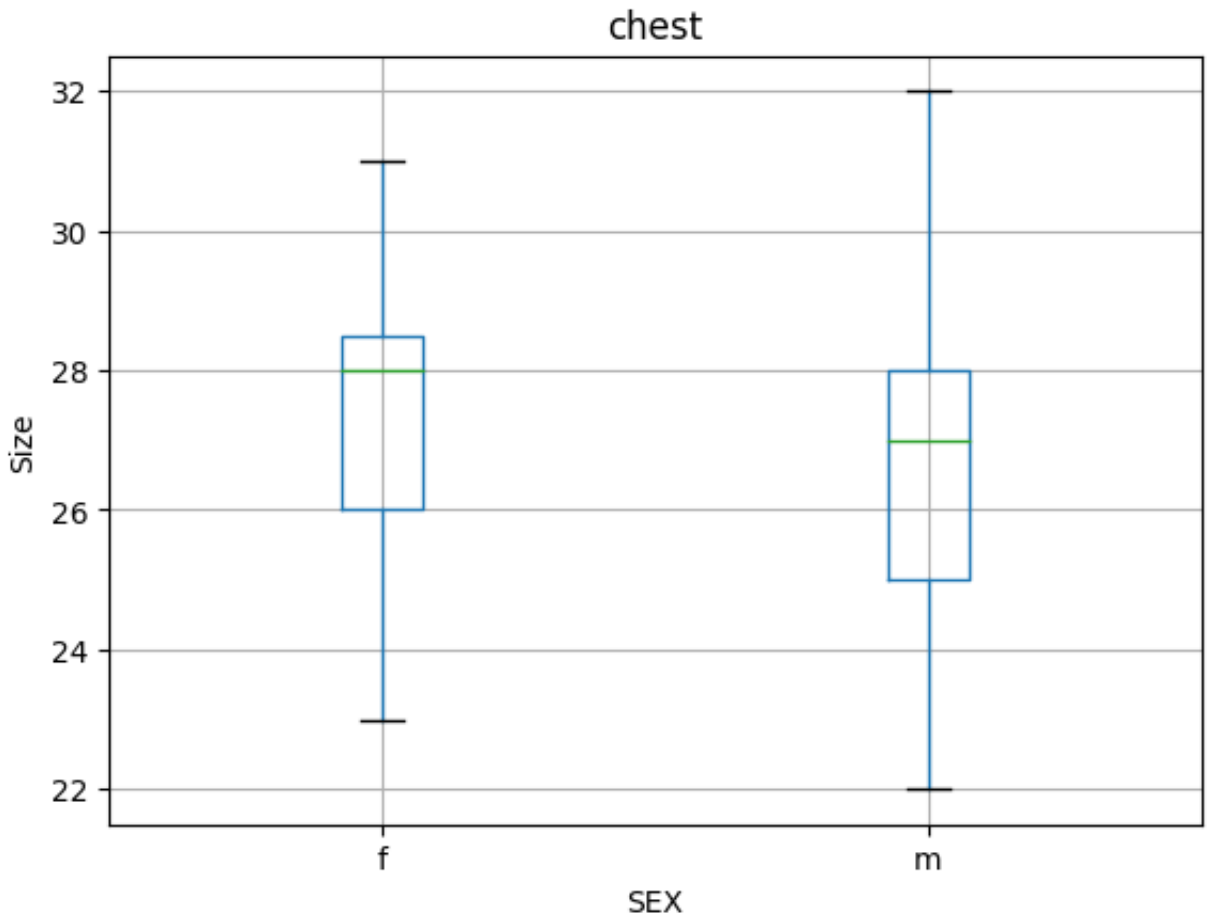

totlngth
950
900
850
800
750
Size
f
m
SEX

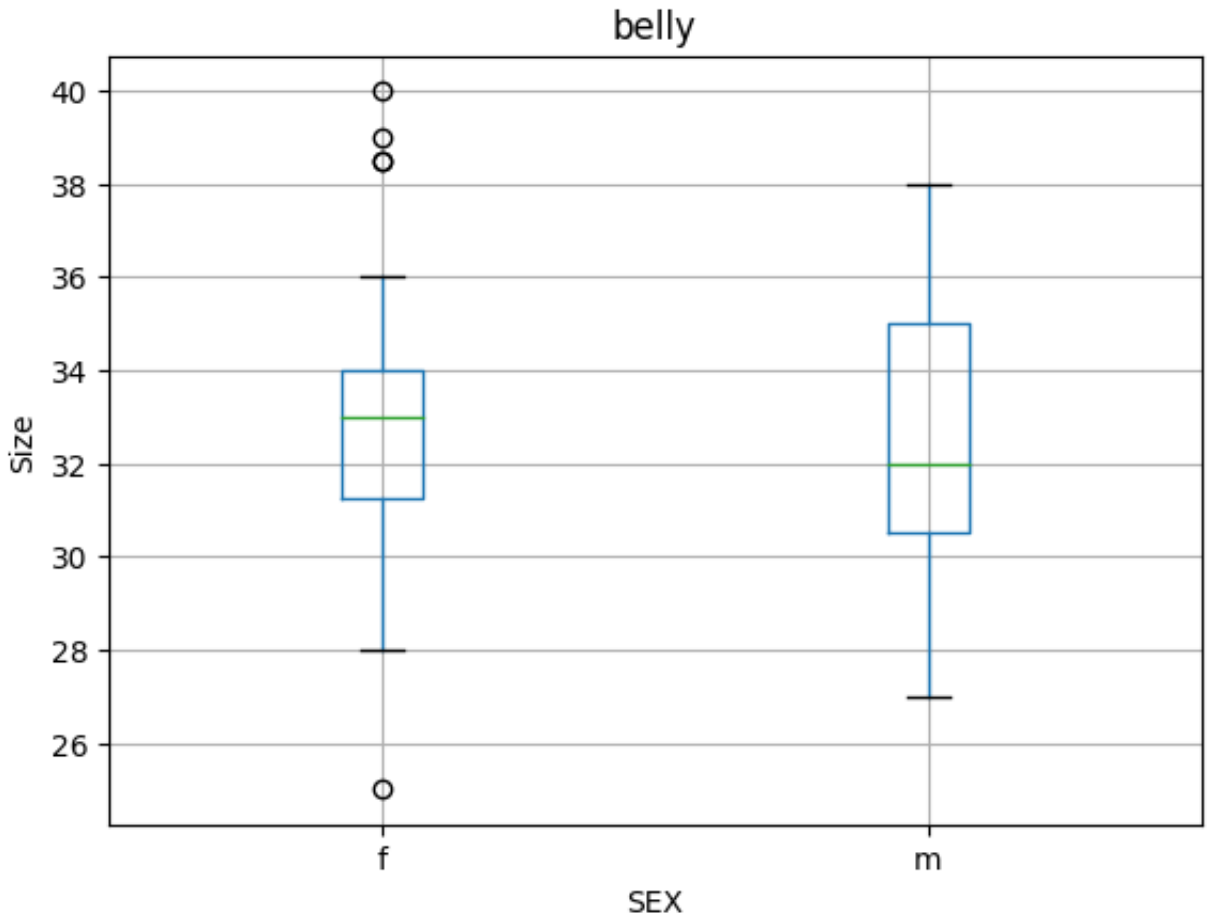

taill
42
40
38
36
34
32
Size
f
m
SEX

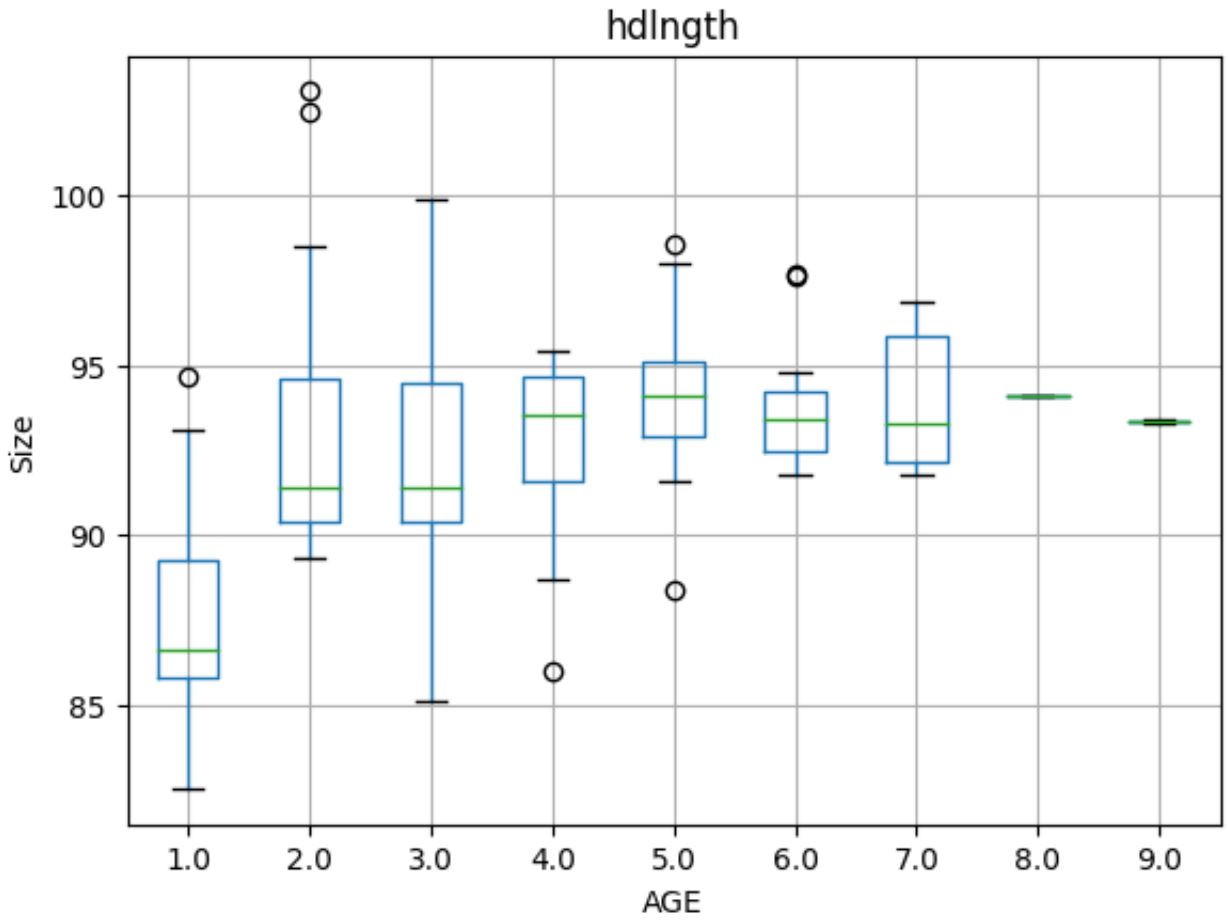

footlgth
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60.0
Size
f
m
SEX

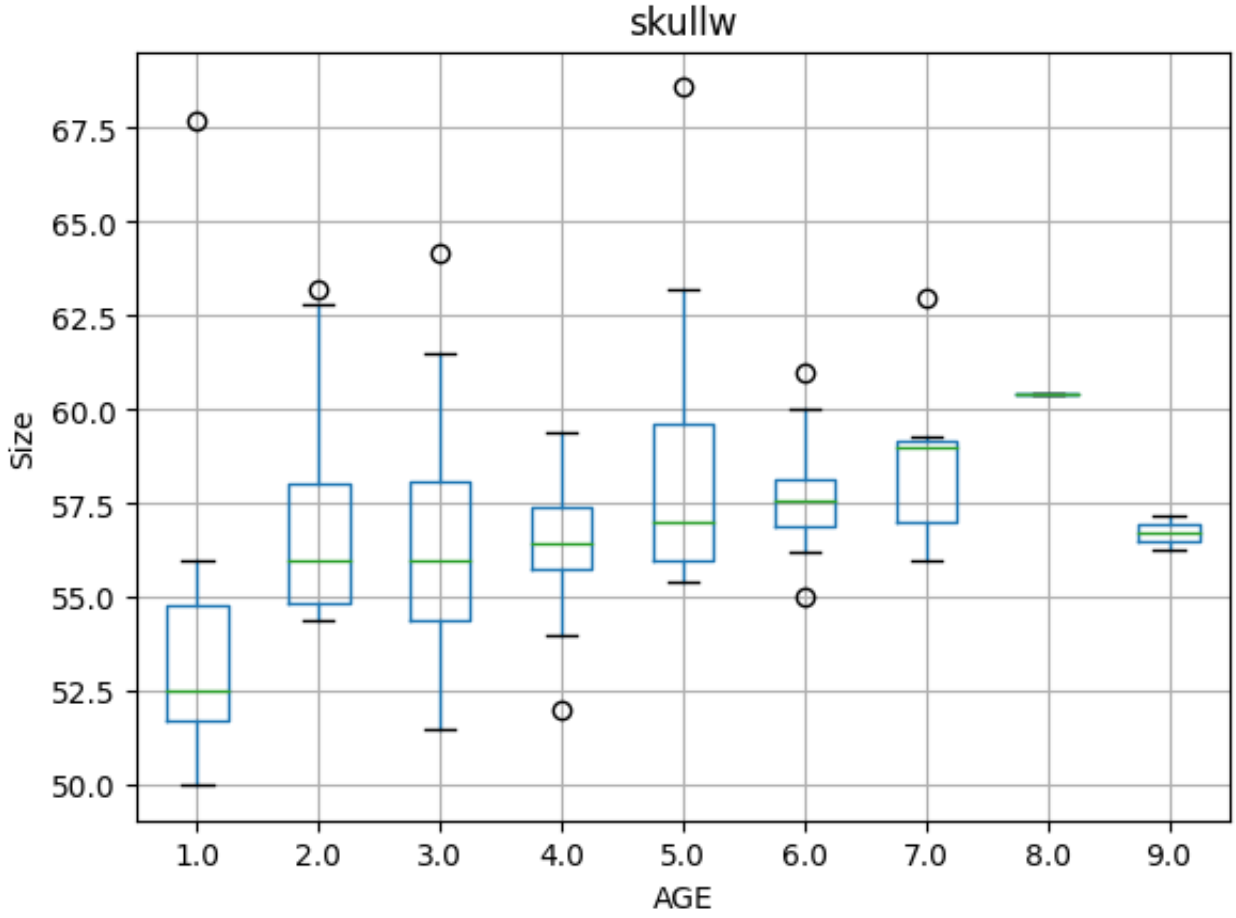

earconch
Size
56
54
52
50
48
46
44
42
40
f
m
SEX

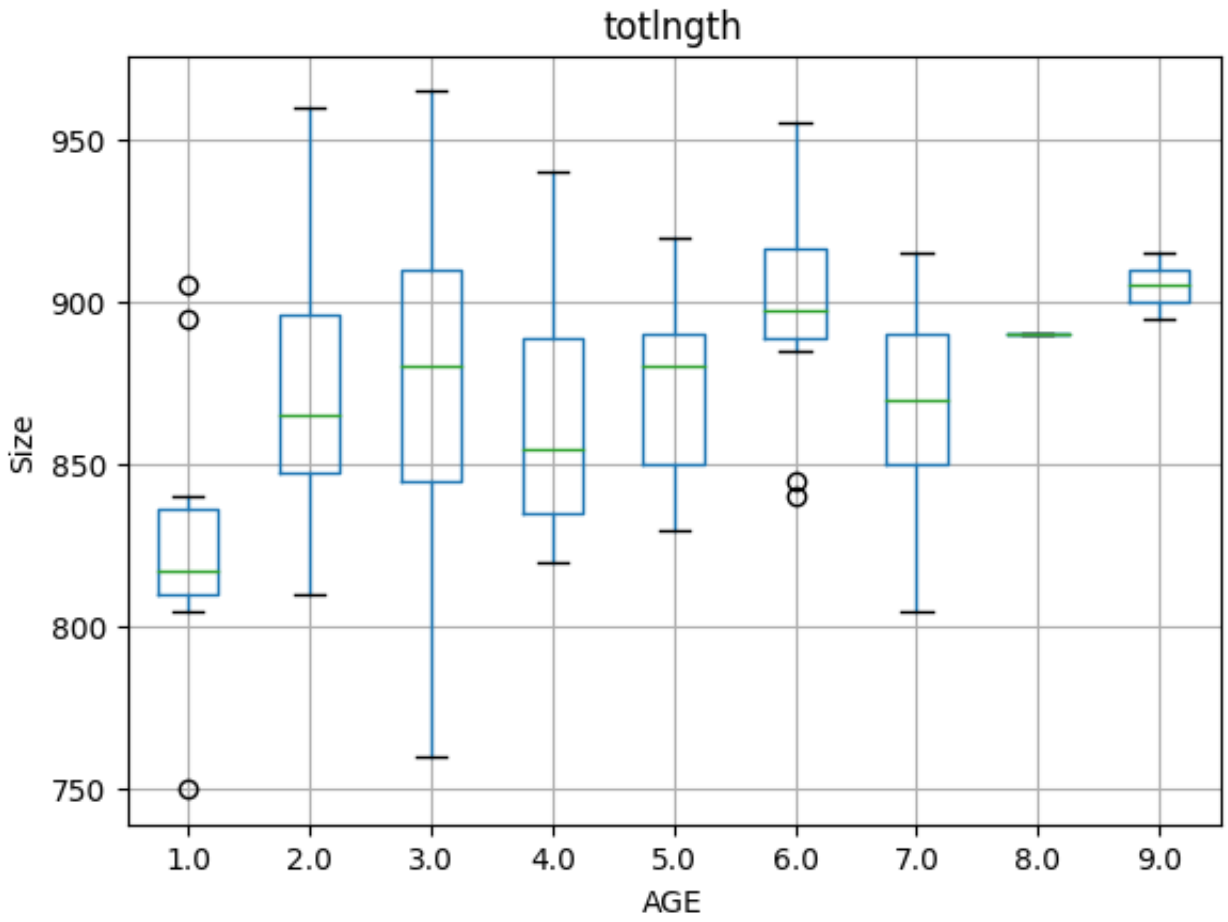

eye
18
17
16
15
14
13
Size
f
m
SEX

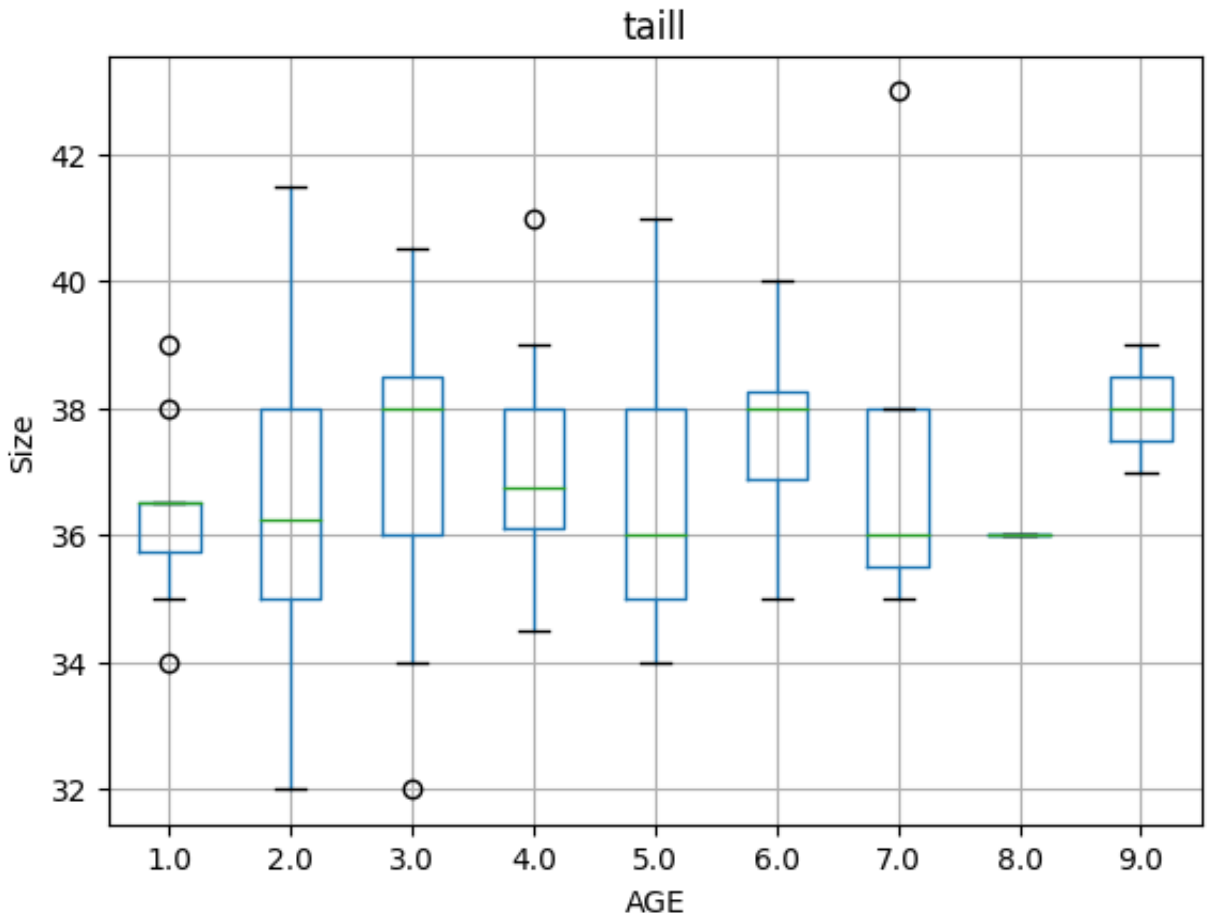

chest
32
30
28
26
24
22
Size
f
m
SEX

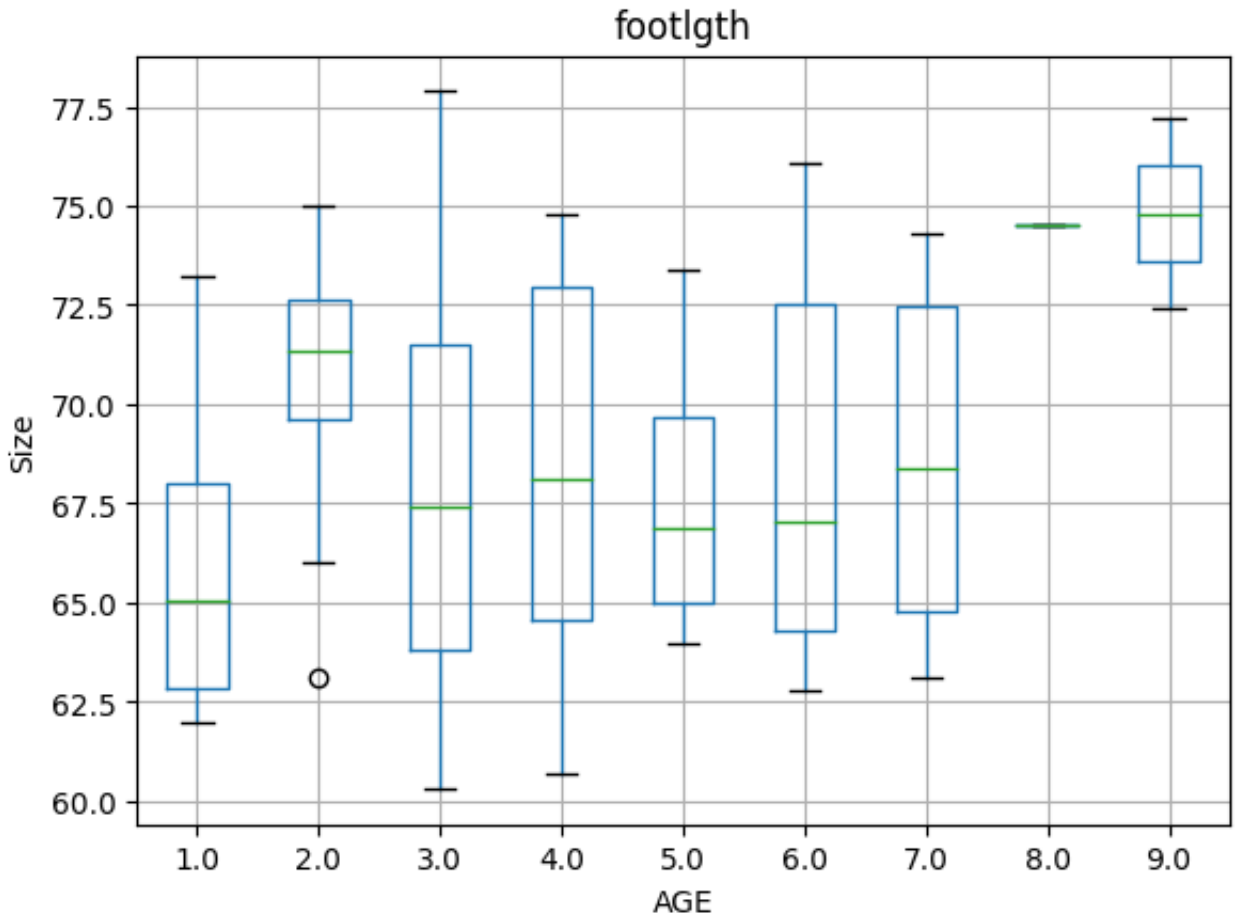

belly
Size
40
38
36
34
32
30
28
26
f
m
SEX

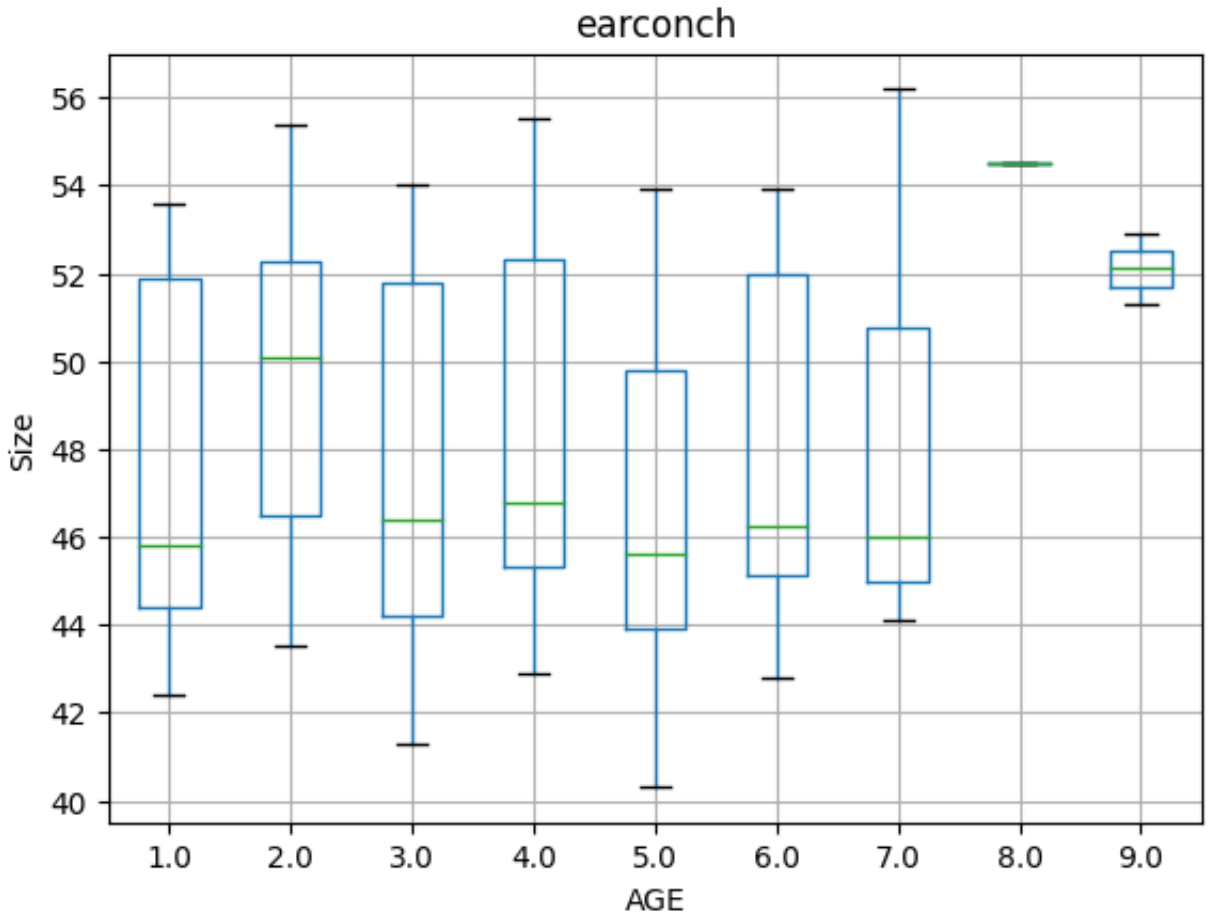

hdlngth
Size
100
95
90
85
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
AGE

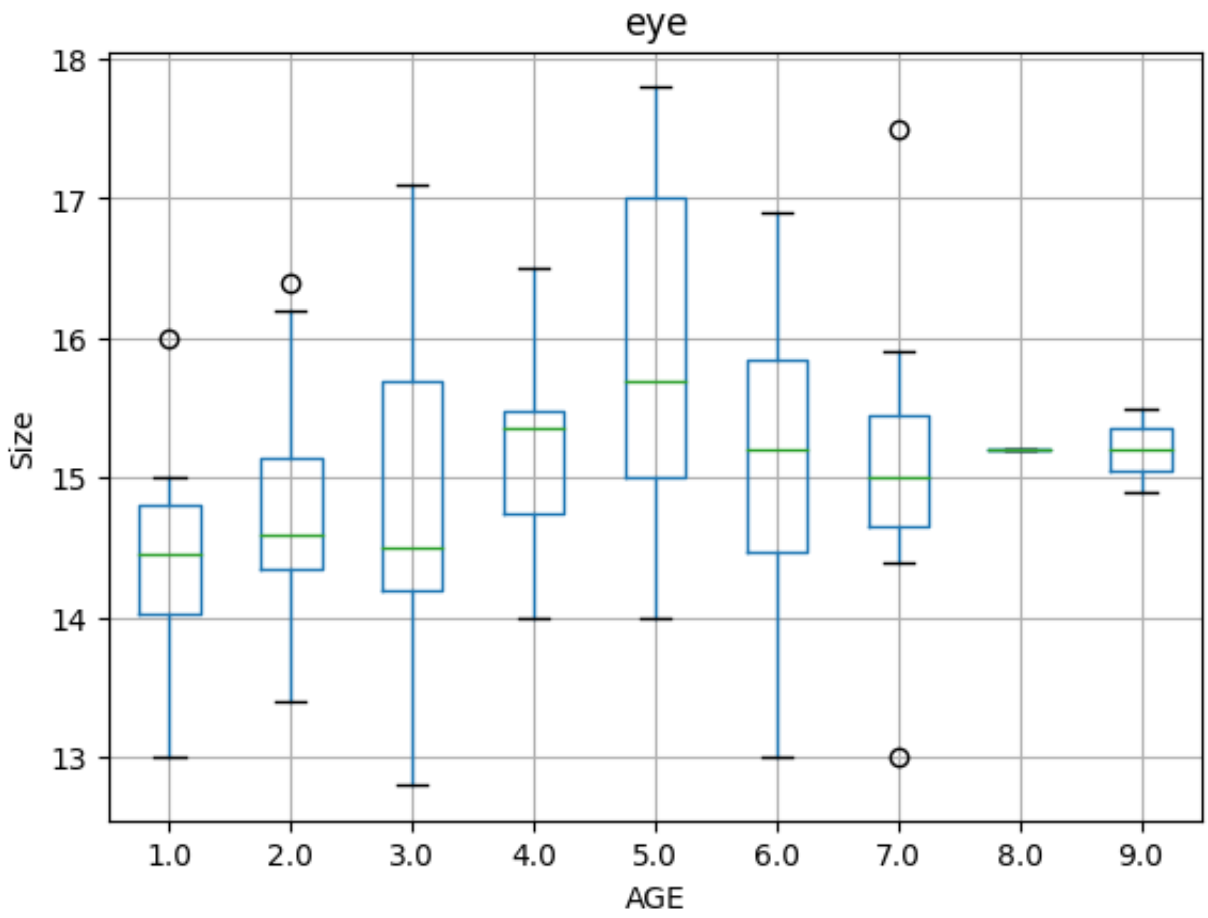

skullw
Size
AGE
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0

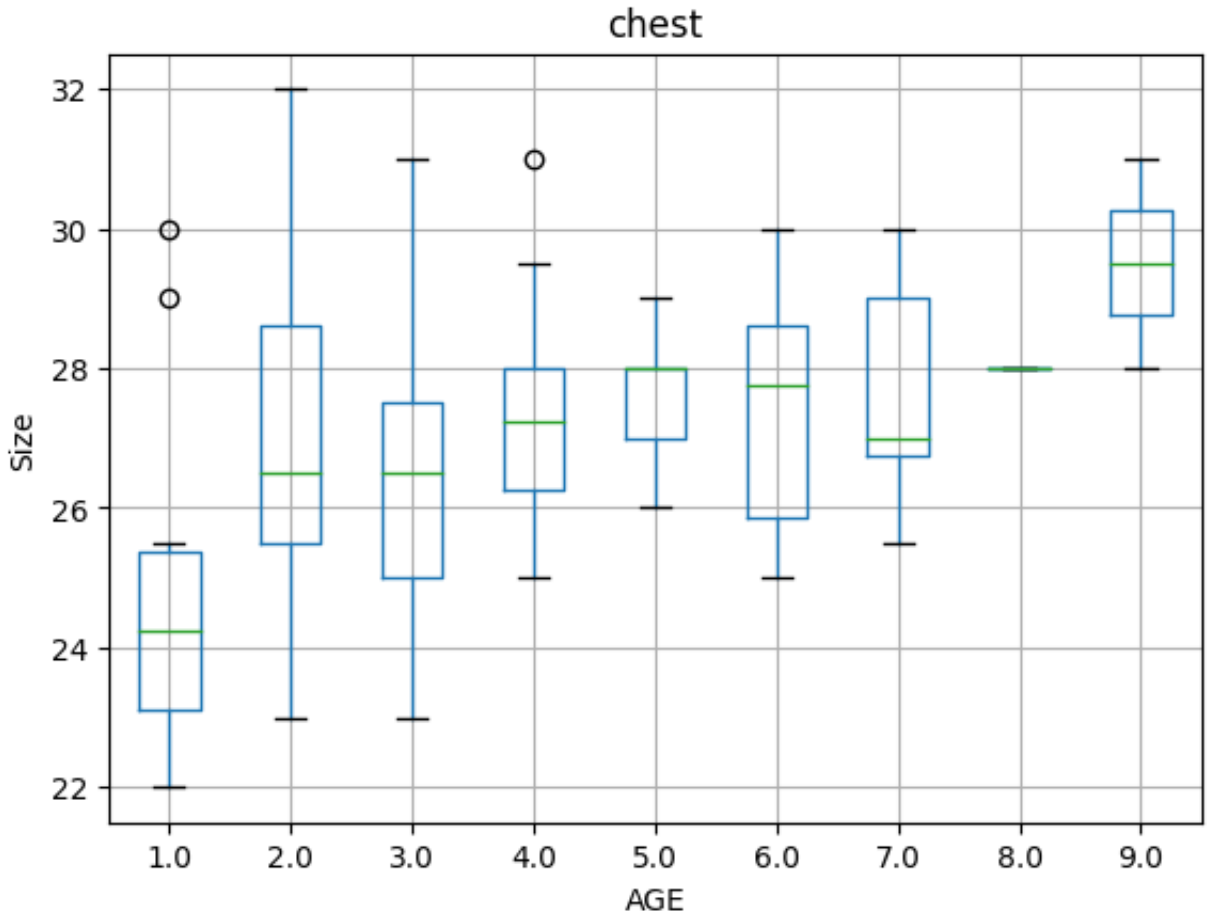

totlngth
Size
950
900
850
800
750
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
AGE

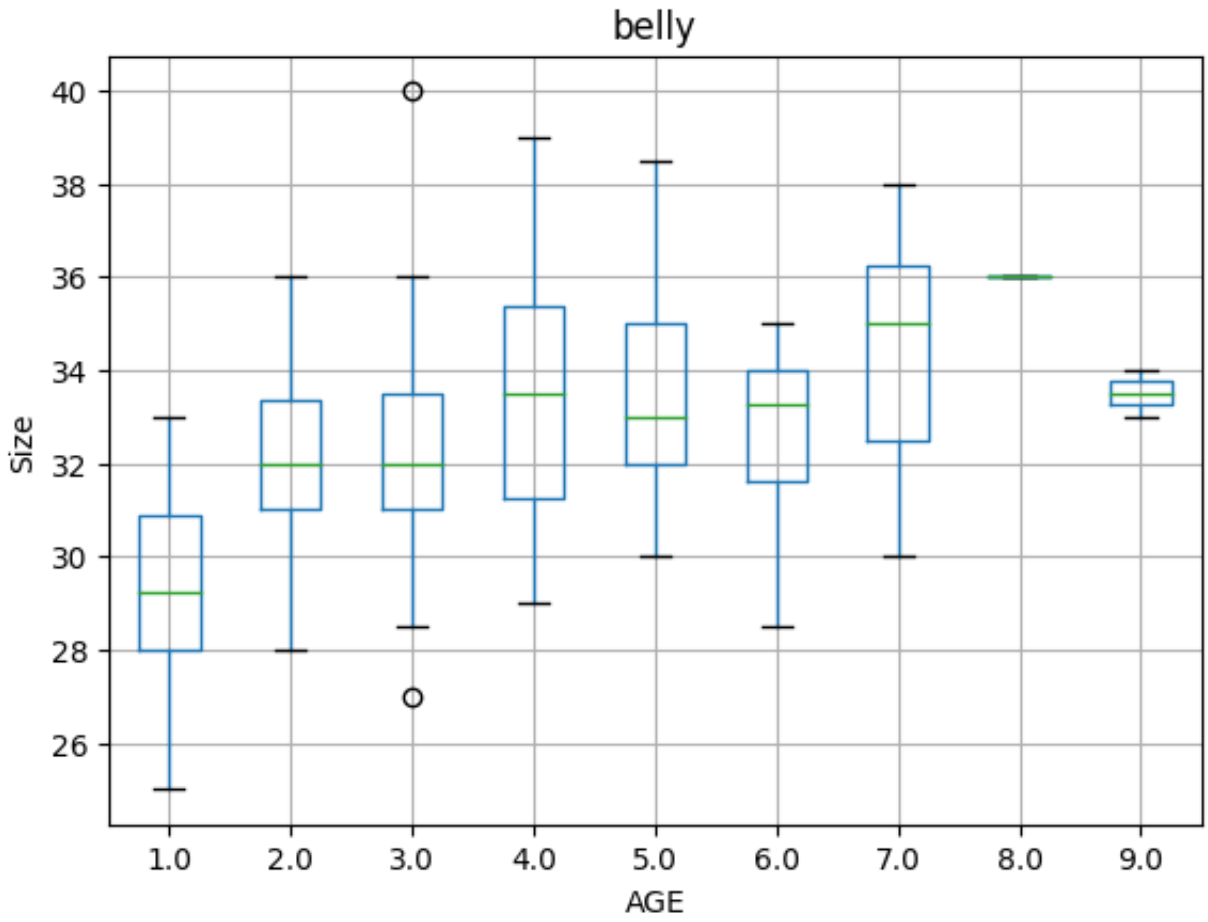

taill
Size
AGE
42
40
38
36
34
32
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0


footlgth
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60.0
Size
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
AGE


earconch
Size
AGE
56
54
52
50
48
46
44
42
40
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0

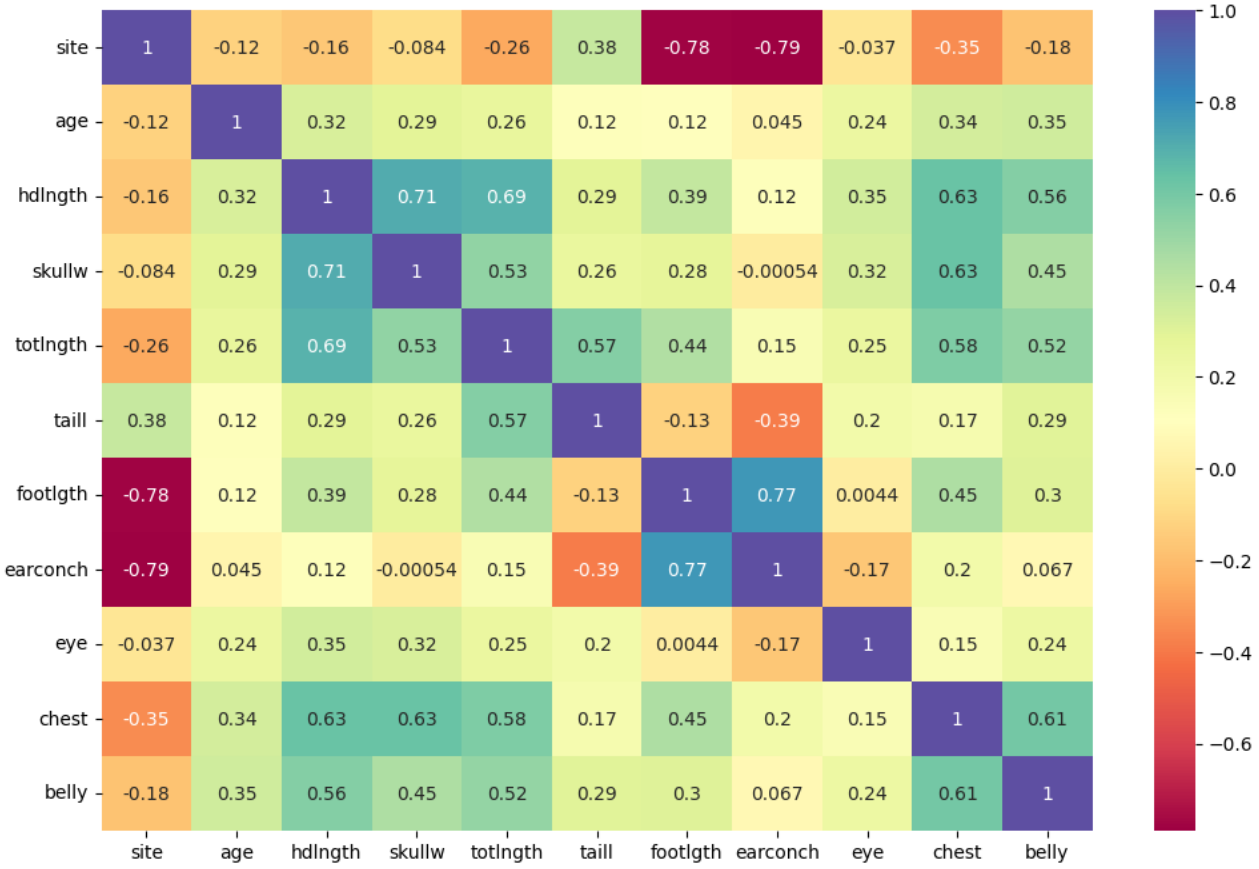

eye
Size
AGE
18
17
16
15
14
13
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0


chest
Size
AGE
32
30
28
26
24
22
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0

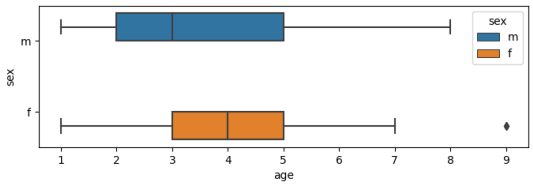


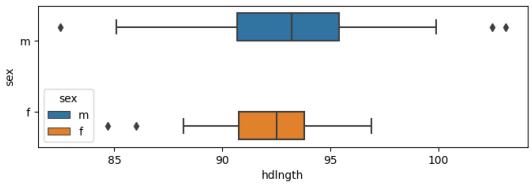
In [11]:
```
#Used to predict which body dimensions are correlated with sex and age

plt.figure(figsize = (12,8))
sns.heatmap(possum.corr(), annot = True, cmap = 'Spectral')
```

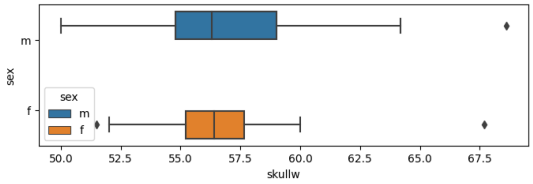
Out[11]: <Axes: >

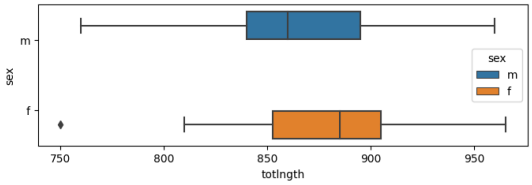


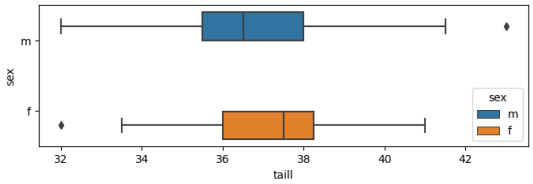
In [12]:
```
#More visually pleasing way to predict which body dimensions are correlated

flt=possum.select_dtypes(include='float')
fig,axes=plt.subplots(nrows=5,ncols=2,figsize=(20,20))
for i,j in enumerate(flt.columns):
                  sns.boxplot(x=possum[j], y=possum['sex'], hue=possum['sex'],
plt.subplots_adjust(wspace=0.5,hspace=0.9)
```

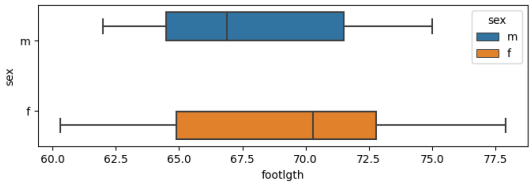

sex
m
f
1 2 3 4 5 6 7 8 9
age

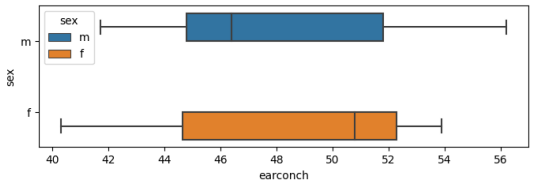

sex
m
f
85 90 95 100
hdlngth

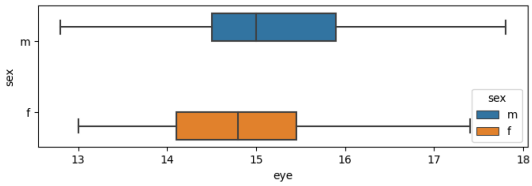

sex
m
f
50.0 52.5 55.0 57.5 60.0 62.5 65.0 67.5
skullw

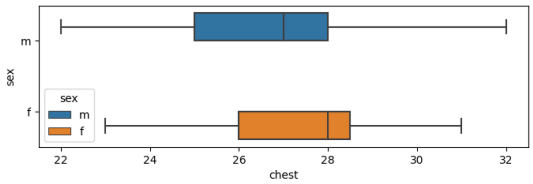

sex
m
f
750 800 850 900 950
totlngth

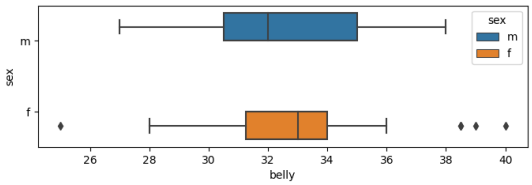

sex
m
f
32 34 36 38 40 42
taill


sex
m
f
60.0 62.5 65.0 67.5 70.0 72.5 75.0 77.5
footlgth


sex
m
f
40 42 44 46 48 50 52 54 56
earconch


sex
m
f
13 14 15 16 17 18
eye


sex
m
f
22 24 26 28 30 32
chest


sex
m
f
26 28 30 32 34 36 38 40
belly

Analysis

- Can we use total length to predict a possum's head length?
  - Linear: I was able to use a Linear Regression model to predict a possum's head length. However, it was not as accuarate as I would have liked it to be. I ran a prediction score on every dimension of the possum and my results stated that the prediciton score for the hdlngth was worse than the average. Making it below average.
  - Logistic: I was unable to use a Logistic Regression model to predict the possum's head length becasue it says that the model couldn't work with continuous values, and can only work with categorical
  - I found Lineaer Regression to be more effective in this case becasue it can hadnle conntinuous variables, unlike Logistic.
- Which possum body dimensions are most correlated with age and sex?
  - Linear: My Linear data showed that chest was the most positively correlated with sex, and eye was most positively correlated to age
  - Logistic: My Logistic data also showed that chest was the most positively correlated with sex, and I was not able to see what was most correlated with age using Logical Regression since age wasn't categorical
  - My graph data showed that foot length and ear conch were most correlated with sex, and skull width and belly size were most correlated to age
  - I found Lineaer Regression to be more effective in this case again becasue it can hadnle conntinuous variables. Both Linear and Logistic could analyze which dimensions were most correlated with sex, but since age is continuous, only Linear could work.
- Can we classify a possum's sex by its body dimensions and location?
  - Linear: Building a Linear Regression model, I was able to classify possums' sex by their body dimensions and location. The classification model was around 76.9% accurate
  - Logistic: Since sex was a categorical variable, I was able to classify it using Logistic Regression. My accuracy for this model was 59.6%
  - I found Lineaer Regression to be more effective in this case becasue the Linear model boasted a higher accuracy than Logistic.
- Can we predict a possum's trapping location from its body dimensions?
  - Linear: Using my Linear Regression model, I was able to predict trapping location of possum based on its body dimensions: being 65.3%
  - Logistic: Using my Logistic Regression model, I was able to predict the trapping location of possum based on its body dimension: being 53.8% accurate
  - I found Lineaer Regression to be more effective in this case becasue the Linear model boasted a higher accuracy than Logistic.

Sources

HTML: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul
Logistic: https://www.justintodata.com/logistic-regression-example-in-python/
Graph help: https://www.kaggle.com/code/andrewbaum/possum-length-regression
Coefficient help: https://scikit-learn.org/stable/modules/model_evaluation.html
Accuracy printing example (I just borrowed In[20]):
https://www.kaggle.com/code/umairnabeel/possum-sex-classification-logistic-regression