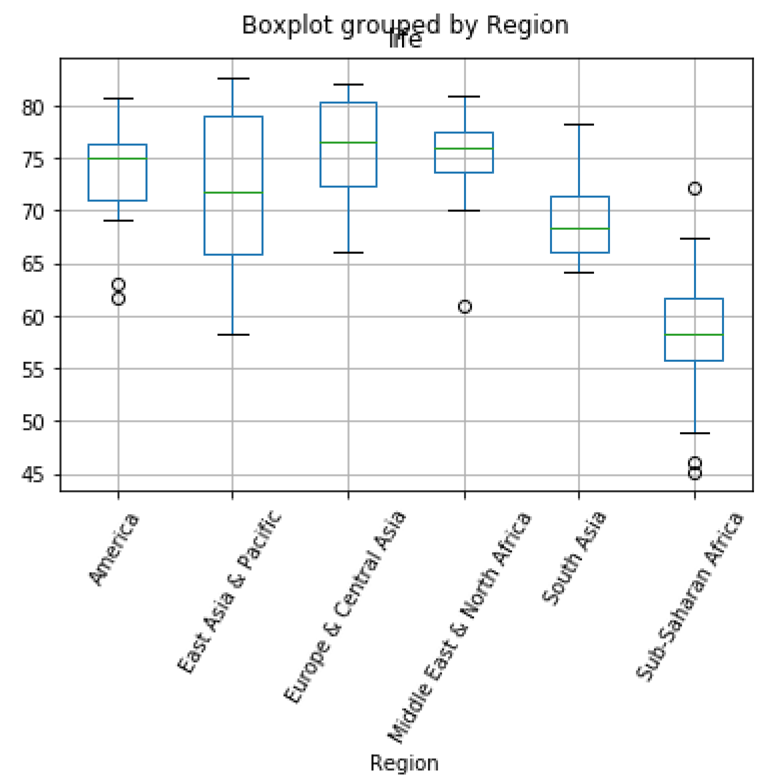
Exercise 1

Exploring categorical features

- Examine categorical variable 'Region' in the Gapminder dataset.
- Boxplots are particularly useful for visualizing categorical features such as this.

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt

filePath = "C:/Users/ljyan/Desktop/courseNotes/dataScience/machineLearning/data/"
filename = "gm_2008_region.csv"
file = filePath+filename
df = pd.read_csv(file, sep =  ',')
df.boxplot('life', 'Region', rot=60)
plt.show()
```



Creating dummy variables

- Scikit-learn does not accept non-numerical features. Sometimes it seems accept. See the KNN example of party votes.
- Binarize categorical variables by creating dummy variables.
- **Need drop one column from pd.get_dummies results as not all the resulting variables are independent**.

In [5]:
```
print(df.head())
df_region = pd.get_dummies(df)
print(df_region.head())

# Drop 'Region_America' from df_region
#Use the get_dummies() function again, this time specifying drop_first=True to
#drop the unneeded dummy variable (in this case, 'Region_America')
df_region = pd.get_dummies(df, drop_first=True)

#Note there is another way of dropping unnecessary dummy variable as below:
#df_region = df_region.drop('Region_America',axis = 1)
#This is necessary when we use get_dummies without setting drop_first = True
```

```
   population  fertility  HIV        CO2  BMI_male      GDP  BMI_female  life  \
0  34811059.0       2.73  0.1   3.328945  24.59620  12314.0    129.9049  75.3
1  19842251.0       6.43  2.0   1.474353  22.25083   7103.0    130.1247  58.3
2  40381860.0       2.24  0.5   4.785170  27.50170  14646.0    118.8915  75.5
3   2975029.0       1.40  0.1   1.804106  25.35542   7383.0    132.8108  72.5
4  21370348.0       1.96  0.1  18.016313  27.56373  41312.0    117.3755  81.5

   child_mortality                      Region
0             29.5  Middle East & North Africa
1            192.0          Sub-Saharan Africa
2             15.4                     America
3             20.0       Europe & Central Asia
4              5.2         East Asia & Pacific
   population  fertility  HIV        CO2  BMI_male      GDP  BMI_female  life  \
0  34811059.0       2.73  0.1   3.328945  24.59620  12314.0    129.9049  75.3
1  19842251.0       6.43  2.0   1.474353  22.25083   7103.0    130.1247  58.3
2  40381860.0       2.24  0.5   4.785170  27.50170  14646.0    118.8915  75.5
3   2975029.0       1.40  0.1   1.804106  25.35542   7383.0    132.8108  72.5
4  21370348.0       1.96  0.1  18.016313  27.56373  41312.0    117.3755  81.5

   child_mortality  Region_America  Region_East Asia & Pacific  \
0             29.5               0                           0
1            192.0               0                           0
2             15.4               1                           0
3             20.0               0                           0
4              5.2               0                           1
```

```
   Region_Europe & Central Asia  Region_Middle East & North Africa  \
0                             0                                  1   
1                             0                                  0   
2                             0                                  0   
3                             1                                  0   
4                             0                                  0   

   Region_South Asia  Region_Sub-Saharan Africa  
0                  0                          0  
1                  0                          1  
2                  0                          0  
3                  0                          0  
4                  0                          0  
```

Regression with categorical features

In [11]:

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score

y = df_region['life'].values
X = df_region.drop('life',axis = 1)

ridge = Ridge(alpha = 0.5, normalize = True)

ridge_cv = cross_val_score(ridge,X,y,cv= 5)

print(ridge_cv)
```

```
[0.86808336 0.80623545 0.84004203 0.7754344  0.87503712]
```

Dropping missing data

In [20]:
```
filePath = "C:/Users/ljyan/Desktop/courseNotes/dataScience/machineLearning/data/"
filename = "house-votes-84.csv"
file = filePath+filename
df = pd.read_csv(file, sep =  ',', header = None)
print(df.head())
```

```
           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
0  republican  n  y  n  y  y  y  n  n  n  y  ?  y  y  y  n  y
1  republican  n  y  n  y  y  y  n  n  n  n  n  y  y  y  n  ?
2    democrat  ?  y  y  ?  y  y  n  n  n  n  y  n  y  y  n  n
3    democrat  n  y  y  n  ?  y  n  n  n  n  y  n  y  n  n  y
4    democrat  y  y  y  n  y  y  n  n  n  n  y  ?  y  y  y  y
```

In [21]:
```
df.columns = ['party', 'infants', 'water', 'budget', 'physician', 'salvador','religious', 'satellite', 'aid', 'm
df[df == 'n'] = 0
df[df == 'y'] = 1
```

In [23]:

```
import numpy as np
df[df == '?'] = np.nan
print(df.isnull().sum())
print("Shape of Original DataFrame: {}".format(df.shape))
df = df.dropna()
print("Shape of DataFrame After Dropping All Rows with Missing Values: {}".format(df.shape))
```

```
party                0
infants              0
water                0
budget               0
physician            0
salvador             0
religious            0
satellite            0
aid                  0
missile              0
immigration          0
synfuels             0
education            0
superfund            0
crime                0
duty_free_exports    0
eaa_rsa              0
dtype: int64
Shape of Original DataFrame: (232, 17)
Shape of DataFrame After Dropping All Rows with Missing Values: (232, 17)
```

Imputing missing data in a ML Pipeline I

In [24]:
```python
from sklearn.preprocessing import Imputer
from sklearn.svm import SVC

imp = Imputer(missing_values= 'NaN', strategy= 'most_frequent', axis= 0)

clf = SVC()

steps = [('imputation', imp),
        ('SVM', clf)]
```

```
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Imputer i
s deprecated; Imputer was deprecated in version 0.20 and will be removed in 0.22. Import impute.SimpleImputer f
rom sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
```

Imputing missing data in a ML Pipeline II

In [26]:
```
from sklearn.preprocessing import Imputer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.metrics import classification_report

from sklearn.model_selection import train_test_split

y = df['party'].values
X = df.drop('party', axis=1).values

steps = [('imputation', Imputer(missing_values='NaN', strategy='most_frequent', axis=0)),
        ('SVM', SVC())]

pipeline =  Pipeline(steps)

#If I need only impute a matrix, then I should use
#imp.fit_transform(X).

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.3,random_state = 42)
pipeline.fit(X_train,y_train)
y_pred = pipeline.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    democrat       0.97      0.97      0.97        36
  republican       0.97      0.97      0.97        34

   micro avg       0.97      0.97      0.97        70
   macro avg       0.97      0.97      0.97        70
weighted avg       0.97      0.97      0.97        70


C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Impute
r is deprecated; Imputer was deprecated in version 0.20 and will be removed in 0.22. Import impute.SimpleImp
uter from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamm
a will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma expl
```

```
icitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Centering and scaling your data

The data X is not the right one.

In [27]:

```
from sklearn.preprocessing import scale
X_scaled = scale(X)

print("Mean of Unscaled Features: {}".format(np.mean(X)))
print("Standard Deviation of Unscaled Features: {}".format(np.std(X)))

print("Mean of Scaled Features: {}".format(np.mean(X_scaled)))
print("Standard Deviation of Scaled Features: {}".format(np.std(X_scaled)))
```

```
Mean of Unscaled Features: 0.5223599137931034
Standard Deviation of Unscaled Features: 0.49949978403915 97
Mean of Scaled Features: -8.733435430779787e-18
Standard Deviation of Scaled Features: 1.0

C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with in
put dtype object was converted to float64 by the scale function.
  warnings.warn(msg, DataConversionWarning)
```

Centering and scaling in a pipeline

The data X is not the right one.

In [29]:
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

steps = [('scaler', StandardScaler()),
        ('knn', KNeighborsClassifier())]

pipeline = Pipeline(steps)

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.3, random_state = 42)

knn_scaled = pipeline.fit(X_train,y_train)

knn_unscaled = KNeighborsClassifier().fit(X_train, y_train)

print('Accuracy with Scaling: {}'.format(knn_scaled.score(X_test,y_test)))
print('Accuracy without Scaling: {}'.format(knn_unscaled.score(X_test,y_test)))
```

```
Accuracy with Scaling: 0.9714285714285714
Accuracy without Scaling: 0.9714285714285714

C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with in
put dtype object was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with in
put dtype object was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with in
put dtype object was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

Bringing it all together I: Pipeline for classification

- Tune the parameter C and gamma. C controls the regularization strength. gamma controls the kernel coefficient.

In [31]:
```
from sklearn.model_selection import GridSearchCV
steps = [('scaler', StandardScaler()),
         ('SVM', SVC())]

pipeline = Pipeline(steps)

parameters = {'SVM__C':[1, 10, 100],
              'SVM__gamma':[0.1, 0.01]}
#'step_name__parameter_name'. Here, the step_name is SVM, and the parameter_names are C and gamma.

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 21)

cv = GridSearchCV(pipeline, param_grid = parameters, cv = 3)

cv.fit(X_train, y_train)

y_pred = cv.predict(X_test)

print("Accuracy: {}".format(cv.score(X_test, y_test)))
print(classification_report(y_test, y_pred))
print("Tuned Model Parameters: {}".format(cv.best_params_))
```

```
Accuracy: 0.9574468085106383
              precision    recall  f1-score   support

    democrat       0.96      0.96      0.96        23
  republican       0.96      0.96      0.96        24

   micro avg       0.96      0.96      0.96        47
   macro avg       0.96      0.96      0.96        47
weighted avg       0.96      0.96      0.96        47

Tuned Model Parameters: {'SVM__C': 10, 'SVM__gamma': 0.1}
```

Bringing it all together II: Pipeline for regression

In [35]:
```
#Prepare the data: Gapminder.
filePath = "C:/Users/ljyan/Desktop/courseNotes/dataScience/machineLearning/data/"
filename = "gm_2008_region.csv"
file = filePath+filename
df = pd.read_csv(file, sep =  ',')

df_region = pd.get_dummies(df, drop_first=True)
y = df_region['life'].values
X = df_region.drop('life',axis = 1)
```

In [36]:
```python
from sklearn.linear_model import ElasticNet
steps = [( 'imputation', Imputer(missing_values= 'NaN', strategy= 'mean', axis=0)),
         ('scaler', StandardScaler()),
         ('elasticnet', ElasticNet())]
pipeline = Pipeline(steps)

# Specify the hyperparameter space
parameters = {'elasticnet__l1_ratio':np.linspace(0,1,30)}
#Specify the hyperparameter space for the l1 ratio using the following notation: #'step_name__parameter_name'.
#Here, the step_name is elasticnet, and the parameter_name is l1_ratio.

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.4, random_state = 42)

gm_cv = GridSearchCV(pipeline,param_grid = parameters, cv = 3) #cv = 3 is actually default.

gm_cv.fit(X_train,y_train)

r2 = gm_cv.score(X_test, y_test)
print("Tuned ElasticNet Alpha: {}".format(gm_cv.best_params_))
print("Tuned ElasticNet R squared: {}".format(r2))
```

```
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Impute
r is deprecated; Imputer was deprecated in version 0.20 and will be removed in 0.22. Import impute.SimpleImp
uter from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Impute
r is deprecated; Imputer was deprecated in version 0.20 and will be removed in 0.22. Import impute.SimpleImp
uter from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Impute
r is deprecated; Imputer was deprecated in version 0.20 and will be removed in 0.22. Import impute.SimpleImp
uter from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_descent.py:492: ConvergenceWarnin
g: Objective did not converge. You might want to increase the number of iterations. Fitting data with very s
mall alpha may cause precision problems.
  ConvergenceWarning)
C:\Users\ljyan\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Impute
r is deprecated; Imputer was deprecated in version 0.20 and will be removed in 0.22. Import impute.SimpleImp
uter from sklearn instead.
```

In [ ]: