Tutorial 10

ST2137-2420

```python
import pandas as pd
import numpy as np
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols

import matplotlib.pyplot as plt
```

Material

This tutorial covers concepts on linear regression, corresponding to chapter 9 from the textbook.

Crabs dataset

The file `crab.txt` contains data on female horseshoe crabs. The columns in the data are :

- Colour, recorded as an integer (values from $1-4$),
- Spine condition (1 = both good, 2 = one worn or broken, 3 = both worn or broken).
- Width of the carapace in cm,
- Number of satellites a female crab has,
- Weight of the crab in in grams, and
- Whether a female crab has a satellite (1=yes, 0=no).

1. Fit the following model to the dataset:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 I(X_2 = 2) + \beta_3 I(X_2 = 3) + \beta_4 X_1 I(X_2 = 2) + \beta_5 X_1 I(X_2 = 3) + e$$

where

- Y: weight
- X_1: width
- X_2: spine

2. Write code that will extract the following quantities:

- The p-value for the t-test for $H_0 : \beta_0 = 0$.
- The estimate of β_4.
- The residual sum of squares.
- The estimate of σ^2.
- The adjusted R^2 value.

3. Compute the prediction for width = 27, spine = 1.

R code

```r
### Q1
crab <- read.table("data/crab.txt", header = TRUE)
```

```
lm_crab <- lm(weight ~ width*as.factor(spine), data=crab)
summary(lm_crab)
```

```

Call:
lm(formula = weight ~ width * as.factor(spine), data = crab)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.21372 -0.11963  0.00676  0.14984  0.79156 

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)    
(Intercept)             -4.65914    0.49664  -9.381   <2e-16 ***
width                    0.27067    0.01825  14.833   <2e-16 ***
as.factor(spine)2        1.91050    1.05774   1.806   0.0727 .  
as.factor(spine)3        0.86047    0.59602   1.444   0.1507    
width:as.factor(spine)2 -0.07243    0.04186  -1.730   0.0854 .  
width:as.factor(spine)3 -0.03455    0.02213  -1.561   0.1204    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2641 on 167 degrees of freedom
Multiple R-squared:  0.7966,	Adjusted R-squared:  0.7906 
F-statistic: 130.8 on 5 and 167 DF,  p-value: < 2.2e-16
```

```
### Q2

# The $p$-value for the $t$-test for $H_0: \beta_0 = 0$.
summary_out <- summary(lm_crab)
summary_out$coefficients[1,4]
## [1] 4.627938e-17
summary_out$coefficients["(Intercept)", "Pr(>|t|)"]
## [1] 4.627938e-17

# The estimate of $\beta_4$.
coef(lm_crab)[5]
## width:as.factor(spine)2
##             -0.07242784
summary_out$coefficients["width:as.factor(spine)2", "Estimate"]
## [1] -0.07242784

# The residual sum of squares.
sum(summary_out$residuals^2)
## [1] 11.64621
sum(summary_out$residuals^2)/167
## [1] 0.06973777

# The estimate of $\sigma^2$.
summary_out$sigma^2
## [1] 0.06973777

# The adjusted $R^2$ value.
summary_out$adj.r.squared
## [1] 0.7905509
```

```r
### Q3

new_data <- data.frame(width = 27, spine = factor(1, levels=c(1,2,3)))
predict(lm_crab, newdata = new_data)
```

```
      1
2.64906
```

Python code

```python
# Q1
crab = pd.read_csv("data/crab.txt", sep="\\s+")
# crab.head()
lm1 = ols('weight ~ width*C(spine, Treatment)', data=crab).fit()
lm1.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 weight   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.791
Method:                 Least Squares   F-statistic:                     130.8
Date:                Mon, 14 Apr 2025   Prob (F-statistic):           7.29e-56
Time:                        14:46:05   Log-Likelihood:                -12.072
No. Observations:                 173   AIC:                             36.14
Df Residuals:                     167   BIC:                             55.06
Df Model:                           5
Covariance Type:            nonrobust
===================================================================================================
                                      coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------
Intercept                          -4.6591      0.497     -9.381      0.000      -5.640      -3.679
C(spine, Treatment)[T.2]            1.9105      1.058      1.806      0.073      -0.178       3.999
C(spine, Treatment)[T.3]            0.8605      0.596      1.444      0.151      -0.316       2.037
width                               0.2707      0.018     14.833      0.000       0.235       0.307
width:C(spine, Treatment)[T.2]     -0.0724      0.042     -1.730      0.085      -0.155       0.010
width:C(spine, Treatment)[T.3]     -0.0346      0.022     -1.561      0.120      -0.078       0.009
==============================================================================
Omnibus:                       51.063   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              202.419
Skew:                          -1.058   Prob(JB):                     1.11e-44
Kurtosis:                       7.858   Cond. No.                     1.86e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.86e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

```python
lm1.pvalues['Intercept']
## np.float64(4.627938309867887e-17)

lm1.params.iloc[4]
## np.float64(-0.0724278375934586)
# lm1.params['width:C(spine, Treatment)[T.2]']
```

```
lm1.ssr#/(173-6)
## np.float64(11.646207241101145)

lm1.mse_resid
## np.float64(0.0697377679107853)
lm1.rsquared_adj
## np.float64(0.7905509176280321)
```

```
new_df = sm.add_constant(pd.DataFrame({'width': [27.0],  'spine': [1]}),
                         has_constant='add')
lm1.predict(new_df, )
```

```
0    2.64906
dtype: float64
```

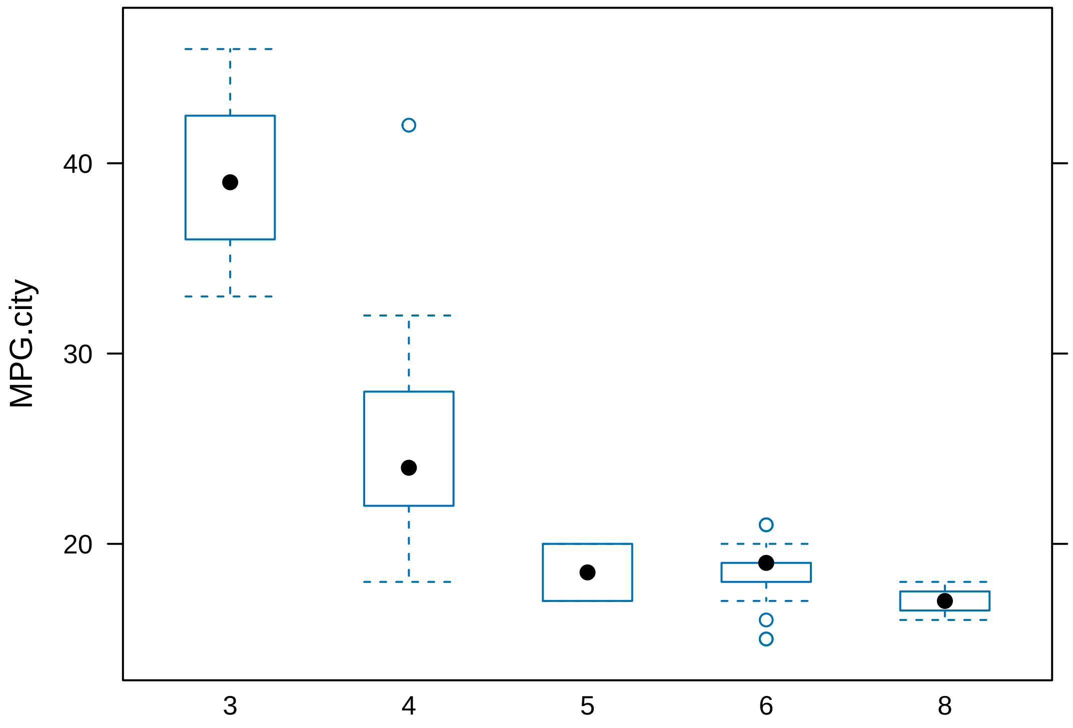
Cars dataset

The `Cars93` dataset from the `MASS` package in R contains information on 93 cars. The description of the 27 columns can be found in `?Cars93`. Consider the two columns `MPG.city` and `Cylinders`. Both of these variables are numeric. Please skim through the help page to understand the dataset before proceeding.

4. Create a boxplot of `MPG.city` (y-axis) vs `Cylinders` (x-axis), comment on the distribution of values.
5. First, fit an ANOVA model to this data to assess if there is any significant difference in mileage between the three groups. Then, fit a simple linear regression model of `MPG.city` versus `Cylinders` (as integers). Inspect the two outputs. What is the difference between these two models? Ignoring the assumptions that need to be met, which is the “correct” one to use?

R code

```
library(lattice)
library(MASS)
cars2 <- Cars93[Cars93$Cylinders != "rotary", ]
bwplot(MPG.city ~ Cylinders, data=cars2)
```



```
cars_anova <- aov(MPG.city ~ Cylinders, data=cars2)
cars2$Cylinders <- as.integer(as.character(cars2$Cylinders))
cars_slr <- lm(MPG.city ~ Cylinders, data=cars2)

summary(cars_anova)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
Cylinders    4 1881.8   470.4   41.15 <2e-16 ***
Residuals   87  994.7    11.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(cars_slr)
```

```

Call:
lm(formula = MPG.city ~ Cylinders, data = cars2)

Residuals:
    Min      1Q  Median      3Q     Max
-7.2888 -2.2984 -0.3659  1.8648 17.7498

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   37.134      1.694  21.919  < 2e-16 ***
Cylinders     -2.961      0.330  -8.975 3.89e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.107 on 90 degrees of freedom
Multiple R-squared:  0.4723,    Adjusted R-squared:  0.4664
F-statistic: 80.54 on 1 and 90 DF,  p-value: 3.888e-14
```

Python code

```python
Cars93 = pd.read_csv("data/Cars93.csv")
Cars93.rename(columns={'MPG.city': 'MPG_city'}, inplace=True)
Cars93 = Cars93[Cars93.Cylinders != 'rotary']

anova_mod1 = ols('MPG_city ~ C(Cylinders, Treatment)', data=Cars93).fit()
anova_mod1.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:               MPG_city   R-squared:                       0.654
Model:                            OLS   Adj. R-squared:                  0.638
Method:                 Least Squares   F-statistic:                     41.15
Date:                Mon, 14 Apr 2025   Prob (F-statistic):           2.56e-19
Time:                        14:46:06   Log-Likelihood:                -240.05
No. Observations:                  92   AIC:                             490.1
Df Residuals:                      87   BIC:                             502.7
Df Model:                           4
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                       39.3333      1.952     20.148      0.000      35.453      43.214
C(Cylinders, Treatment)[T.4]   -14.4762      2.011     -7.198      0.000     -18.473     -10.479
C(Cylinders, Treatment)[T.5]   -20.8333      3.087     -6.749      0.000     -26.969     -14.698
C(Cylinders, Treatment)[T.6]   -20.9140      2.045    -10.229      0.000     -24.978     -16.850
C(Cylinders, Treatment)[T.8]   -22.3333      2.333     -9.571      0.000     -26.971     -17.696
==============================================================================
Omnibus:                       48.735   Durbin-Watson:                   2.042
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              211.148
Skew:                           1.642   Prob(JB):                     1.41e-46
Kurtosis:                       9.655   Cond. No.                         15.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
"""
```

```python
reg_mod1 = ols('MPG_city ~ Cylinders', data=Cars93).fit()
reg_mod1.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:               MPG_city   R-squared:                       0.654
Model:                            OLS   Adj. R-squared:                  0.638
Method:                 Least Squares   F-statistic:                     41.15
Date:                Mon, 14 Apr 2025   Prob (F-statistic):           2.56e-19
Time:                        14:46:06   Log-Likelihood:                -240.05
No. Observations:                  92   AIC:                             490.1
Df Residuals:                      87   BIC:                             502.7
Df Model:                           4
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------------
Intercept          39.3333      1.952     20.148      0.000      35.453      43.214
Cylinders[T.4]    -14.4762      2.011     -7.198      0.000     -18.473     -10.479
Cylinders[T.5]    -20.8333      3.087     -6.749      0.000     -26.969     -14.698
Cylinders[T.6]    -20.9140      2.045    -10.229      0.000     -24.978     -16.850
Cylinders[T.8]    -22.3333      2.333     -9.571      0.000     -26.971     -17.696
==============================================================================
Omnibus:                       48.735   Durbin-Watson:                   2.042
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              211.148
Skew:                           1.642   Prob(JB):                     1.41e-46
Kurtosis:                       9.655   Cond. No.                         15.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
"""
```

There is no "correct" model here. ANOVA is meant for comparison of means between groups. Regression is for modeling one variable on another. It just so happens that the explanatory variable that defines groups is also numeric, which means that technically, we have no problems fitting an SLR.

However, take a look at the output:

- ANOVA models have to estimate the means for 4 groups. SLR only needs to estimate an intercept and a slope. If we have only a few observations, or many groups, SLR uses the information in the data more effectively.
- Now think about the purpose of SLR. We typically want to understand how much mileage changes (on average) for each change in cylinders. We could use the model to estimate the mileage for cars with 7 cylinders (which probably will never be made). But if we are not going to do so, perhaps SLR is not appropriate in this case.
- Is the trend really linear? This is a strong argument *against* SLR in this case.

Population dataset

The dataset in `sg_population.csv` was obtained from the Singapore Department of Statistics. It contains information on the resident population in Singapore since 1950. Population growth is sometimes modeled with the following equation:

$$P = \frac{K}{1 + ae^{-bX}}$$

where:

- P is the population in year X
- a and b are constants to be estimated.
- K is a constant that defines the maximum possible population.

6. Read the data into R and Python and plot it.
7. Prove that the equation can be re-written as

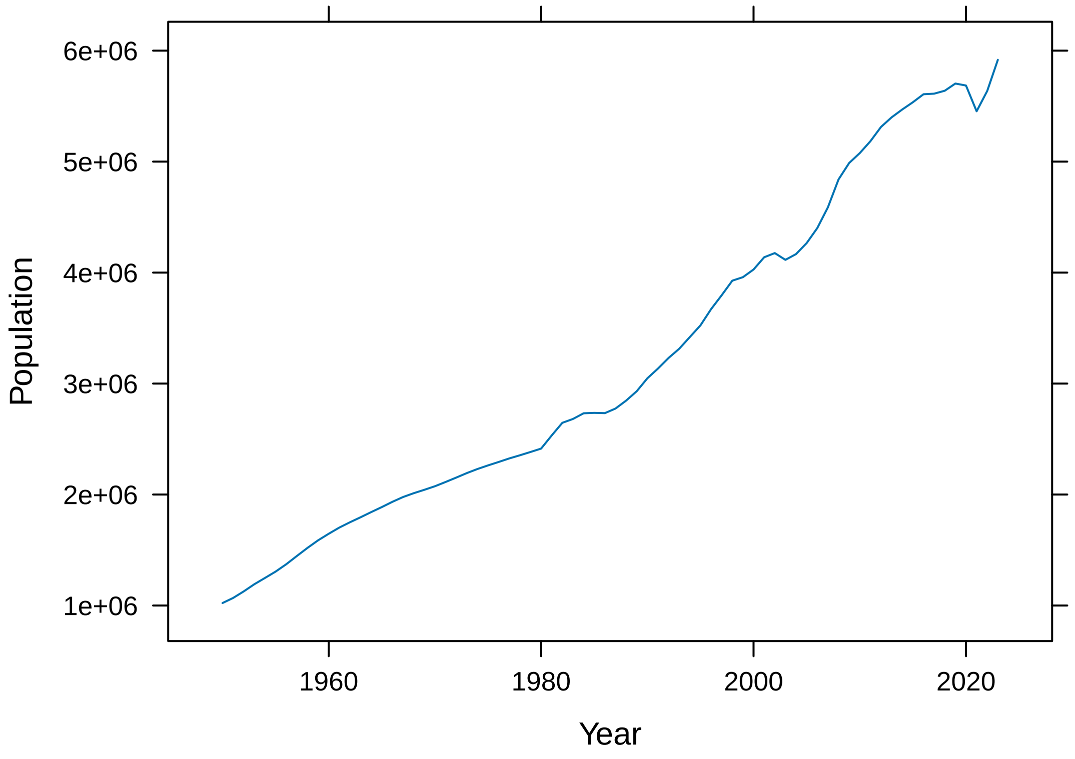
$$\underbrace{\ln\left(\frac{K}{P} - 1\right)}_{Y} = \ln a - bX$$

8. Using $K = 7 \times 10^6$, fit a simple linear regression model and report the estimates of a and b for this data.
9. Use the model to estimate the mean population until year 2050 and plot it, along with the original data.

Solution

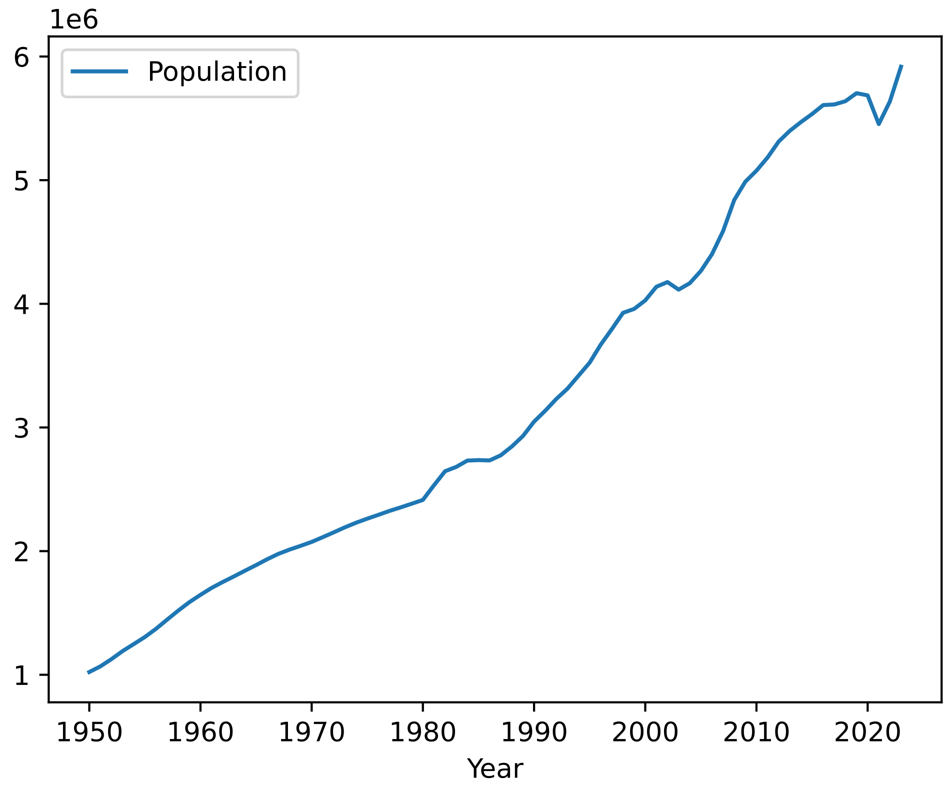
R code

```r
library(lattice)
sg_pop <- read.csv("../data/sg_population.csv")
colnames(sg_pop) <- c("Year", "Population")
xyplot(Population ~ Year, data=sg_pop, type="l")
```



Python code

```python
sg_pop = pd.read_csv("../data/sg_population.csv")
sg_pop.columns = ['Year', 'Population']

sg_pop.plot(x='Year', y='Population');
```



To fit the model, we create the new column corresponding to the Y variable.

R code

```
K <- 7e6
sg_pop$y  <- log(K/sg_pop$Population - 1)
lm_sg_pop <- lm(y ~ Year, data=sg_pop)

est_coef <- unname(coef(lm_sg_pop))
b <- -est_coef[2]
a <- exp(est_coef[1])
cat("The values of a and b are:", format(a,  digits=3), "and",
    format(b, digits=3), ".")
```

```
The values of a and b are: 9.34e+37 and 0.0439 .
```

Python code

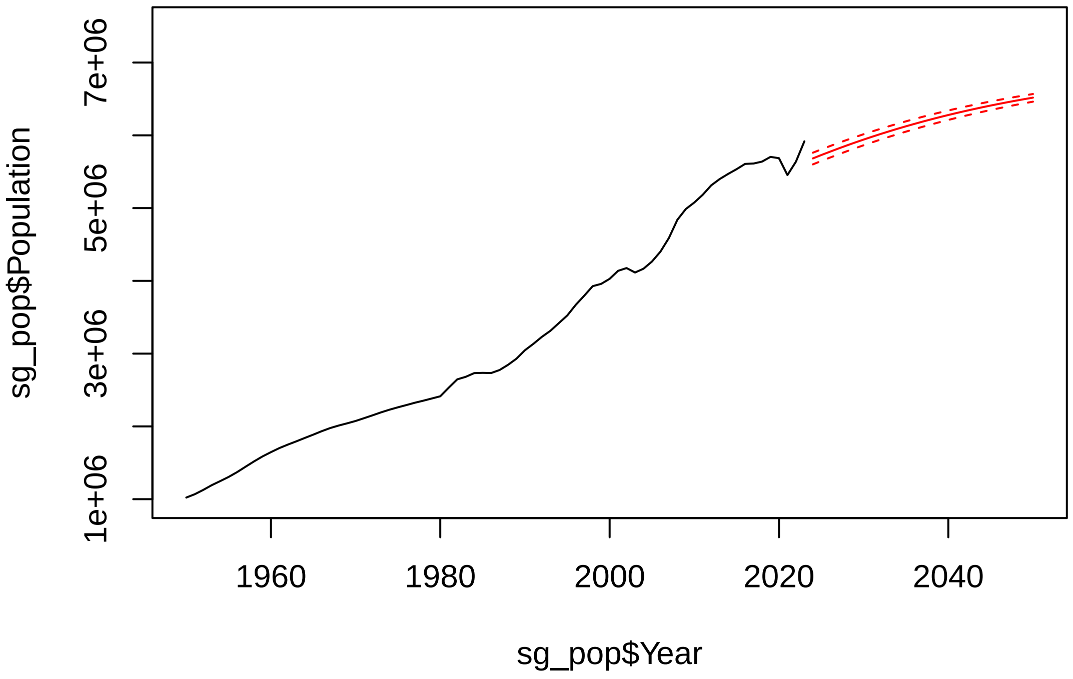
```
K = 7000000
sg_pop['y'] = np.log(K/sg_pop['Population'] - 1)
lm_sg_pop = ols('y ~ Year', data=sg_pop).fit()

#lm_sg_pop.summary()

a = np.exp(lm_sg_pop.params['Intercept'])
b = -lm_sg_pop.params['Year']

print(f"The values of a and b are : {a: .3g} and {b: .4f}")
```

```
The values of a and b are :  9.34e+37 and  0.0439
```

Our last task is to create the plot.

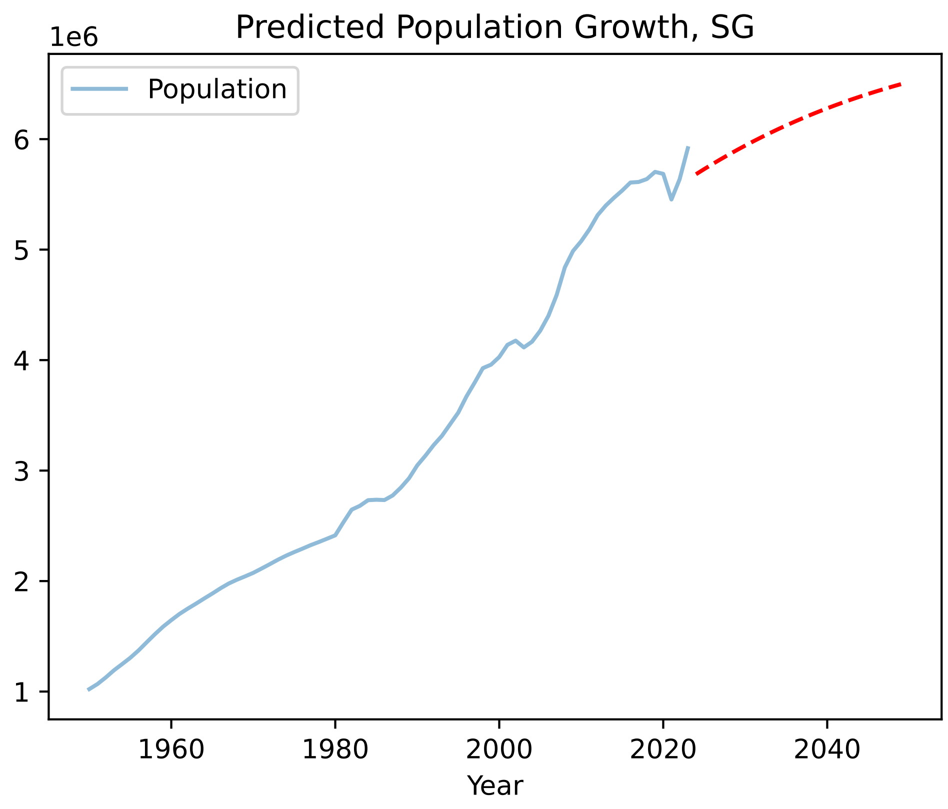
R code

```
new_df <- data.frame(Year= 2024:2050)
yy_pred <- predict(lm_sg_pop, new_df, interval = "confidence")
pop_pred <- K/(1 + exp(yy_pred))

plot(sg_pop$Year, sg_pop$Population, type="l",
     xlim=c(1950, 2050), ylim=c(1e06, 7.5e06))
lines(new_df$Year, pop_pred[,1], col="red")
lines(new_df$Year, pop_pred[,2], col="red", lty=2)
lines(new_df$Year, pop_pred[,3], col="red", lty=2)
```



Python code

```
new_df = sm.add_constant(pd.DataFrame({'Year' : np.arange(2024, 2050)}))
predictions_out = lm_sg_pop.get_prediction(new_df)
predicted_means = K / (1 + np.exp(predictions_out.predicted))

ax = sg_pop.plot(x='Year', y='Population', alpha=0.5 )
ax.set_title('Predicted Population Growth, SG');
ax.plot(new_df.Year, predicted_means, color='red', linestyle="dashed");
```


Predicted Population Growth, SG
1e6
Population
6
5
4
3
2
1
1960
1980
2000
2020
2040
Year