Розеттский камень

Пуассон, фея и три мексиканских негодяя

2019-09-16

Оглавление

Глава 1

Напутственное слово

Глава 2

Коан об установке софта

В этом коане мы рассмотрим установку и настройку программ для работы на языках программирования R и Python, а также установку и настройку программы Stata.

###Язык программирования R > R - это открытая среда программирования, помогающая в работе со статистическими данными. Для программирования на R подойдет программа RStudio.

Рассмотрим установку RStudio на Mac OS и Windows.

#####Инструкция по установке RStudio для Windows / Mac OS:

1. Загрузите и установите язык программирования R с официального сайта.

   - Версия для Windows: Выберите “Download R for Windows” ▸ “base” ▸ “Download R 3.x.x for Windows”.

   - Версия для Mac OS: Выберите “Download R for (Mac) OS X” ▸ “Latest Release” ▸ “R 3.x.x”.

2. Загрузите программу RStudio с официального сайта разработчика (выберите подходящую версию из предложенных опций). Возможностей бесплатной версии будет вполне достаточно для работы.

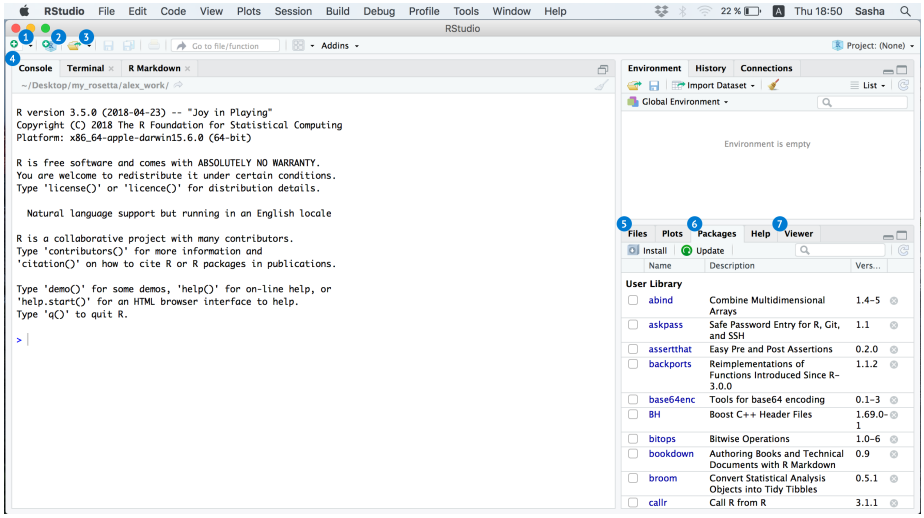
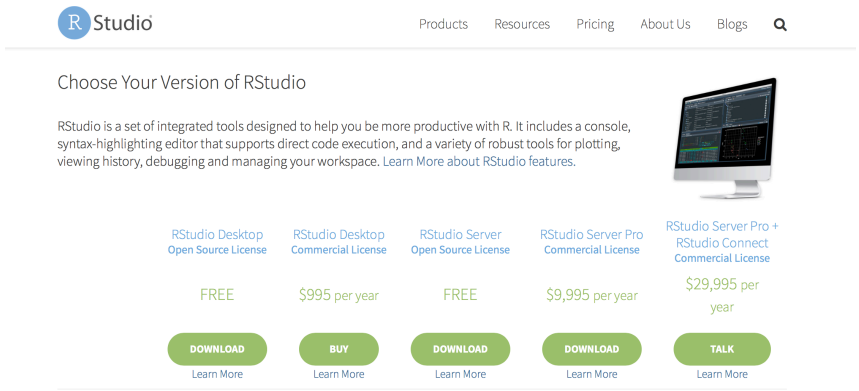


Рис. 2.1: *Интерфейс программы*



Готово, Вы можете использовать RStudio на вашем компьютере.

#####Начало работы

1. **New file** - Создание нового файла.
2. **New project** - Создание нового проекта.
3. **Open file** - Открытие существующего файла.
4. **Console** - Консоль, в которой набирается код.
5. **Files** - Список файлов, доступных для работы.
6. **Packages** - Список установленных пакетов, т.е. расширений. Также можно ознакомиться с ним, введя в консоль команду *installed.packages()*.

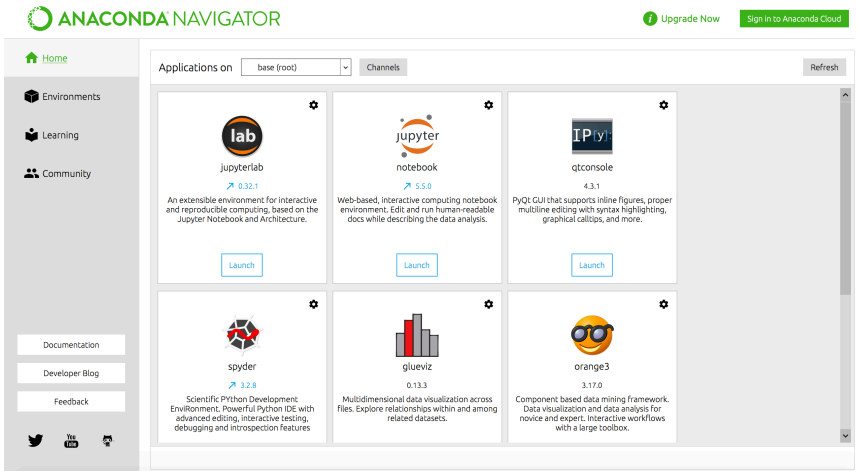
7. **Viewer** - Отображение введенного кода.

###Язык программирования Python > Python - это ещё одна открытая среда программирования, помогающая в работе со статистическими данными. Для программирования на Python подойдет программа Jupyter Notebook.

#####Установка

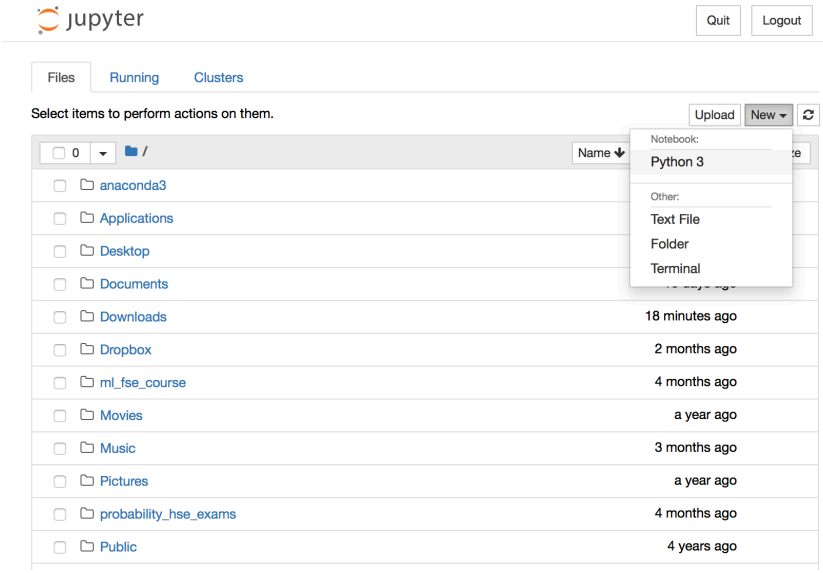
1. Загрузите и установите Anaconda с официального сайта.

2. После загрузки и установки откройте Anaconda Navigator, через который Вы сможете открыть программу Jupyter Notebook. Navigator.bb



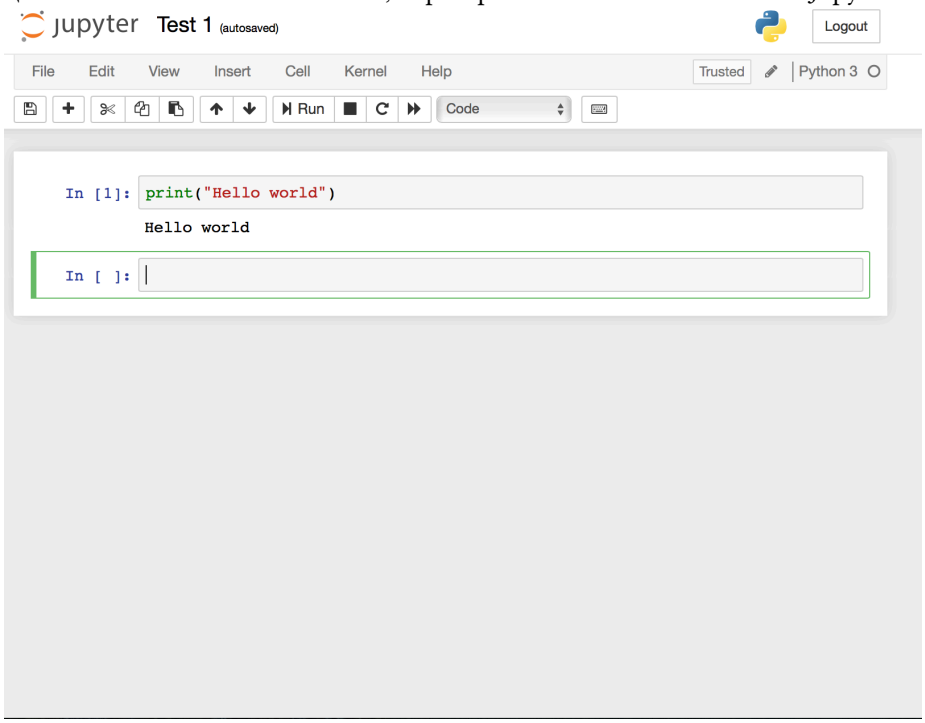
#####Начало работы

Открыв Jupyter Notebook, вы попадете на страницу, содержащую ваши сохраненные файлы. Чтобы создать новый файл, нажмите “New” ▸ “Notebook: Python



3". File in Jupyter.bb

Затем, в открывшемся окне, появится новый файл. Теперь все готово к работе. Вы можете вводить свой код и затем, используя комбинацию клавиш "Shift" + "Enter", проверять его исполнение. in Jupyter.bb



Interface.bb

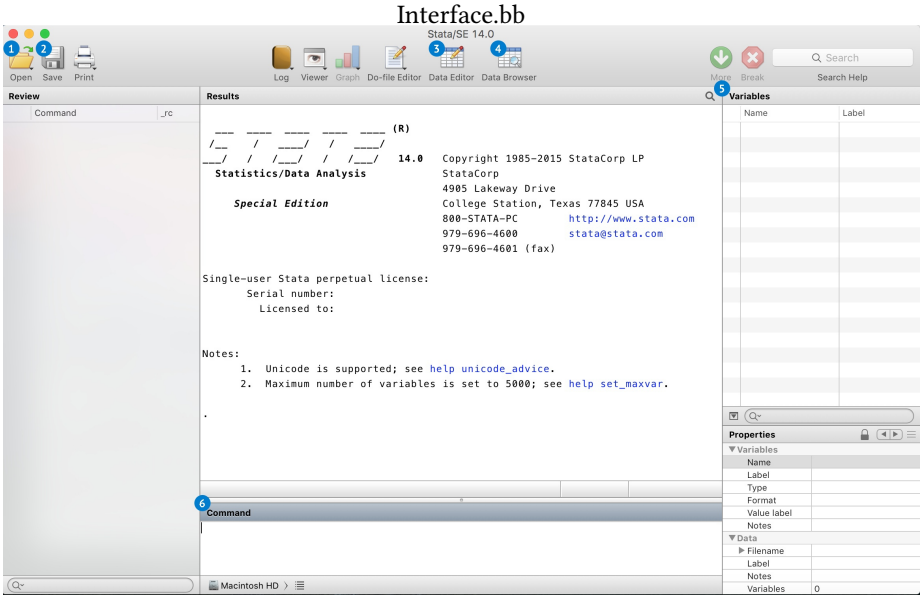


Рис. 2.2: *Интерфейс Stata*

###Программа STATA > Stata, в отличие от R и Python, является программой, а не языком программирования. Она также помогает в работе со статистическими данными.

#####Установка:

Для установки Stata необходимо загрузить актуальную версию с сайта компании-разработчика. Подойдут как Stata SE, так и Stata MP.

#####Начало работы:

1. **Open File** - открыть файл.
2. **Save** - сохранить файл.
3. **Data Editor** - редактирование данных.
4. **Data Browser** - просмотр данных.
5. **Variables** - список переменных.
6. **Command** - командная строка, в которой вводится код.

Глава 3

Коан о простой линейной регрессии

Построим простую линейную регрессию в R и проведем несложные тесты.

Загрузим необходимые пакеты.

```
library(tidyverse) # для манипуляций с данными и построения графиков
library(skimr) # для красивого summary
library(rio) # для чтения .dta файлов
library(car) # для линейных гипотез
library(tseries) # для теста на нормальность
library(sjPlot) # еще графики
```

Импортируем данные.

```
df = import("us-return.dta")
```

Исследуем наш датасет.

```
# skim_with(numeric = list(hist = NULL, p25 = NULL, p75 = NULL)) # опустим некоторые описательные характеристики
skim(df) # посмотрим на данные
```

Skim summary statistics
 n obs: 2664
 n variables: 22

-- Variable type:character ---
----
 variable missing complete   n min max empty n_unique
        B   0   2664 2664   0   6 2544   31

```
-- Variable type:numeric ------------------------------------------------------------------------
------------------------------------------------
 variable missing complete    n    mean      sd      p0     p25     p50
        A    2544      120 2664 60.5    34.79    1      30.75   60.5
    BOISE    2544      120 2664  0.017   0.097  -0.27   -0.045   0.015
   CITCRP    2544      120 2664  0.012   0.081  -0.28   -0.037   0.011
    CONED    2544      120 2664  0.019   0.05   -0.14   -0.012   0.019
   CONTIL    2544      120 2664 -0.0011  0.15   -0.6    -0.051   0
   DATGEN    2544      120 2664  0.0075  0.13   -0.34   -0.072   0.017
      DEC    2544      120 2664  0.02    0.099  -0.36   -0.051   0.024
    DELTA    2544      120 2664  0.012   0.096  -0.26   -0.053   0.013
   GENMIL    2544      120 2664  0.017   0.065  -0.15   -0.026   0.011
   GERBER    2544      120 2664  0.016   0.088  -0.29   -0.036   0.015
      IBM    2544      120 2664  0.0096  0.059  -0.19   -0.029   0.002
   MARKET    2544      120 2664  0.014   0.068  -0.26   -0.013   0.012
    MOBIL    2544      120 2664  0.016   0.08   -0.18   -0.032   0.013
    MOTOR    2544      120 2664  0.018   0.097  -0.33   -0.053   0.017
    PANAM    2544      120 2664  0.0035  0.13   -0.31   -0.065   0
     PSNH    2544      120 2664 -0.0042  0.11   -0.48   -0.049   0
   rkfree    2544      120 2664  0.0068  0.0022  0.0021  0.0052  0.0066
   RKFREE    2544      120 2664  0.0068  0.0022  0.0021  0.0052  0.0066
    TANDY    2544      120 2664  0.025   0.13   -0.25   -0.058   0.022
   TEXACO    2544      120 2664  0.012   0.08   -0.19   -0.037   0.01
    WEYER    2544      120 2664  0.0096  0.085  -0.27   -0.049  -0.002
     p75    p100     hist
  90.25   120     ▇▁▁▁▁▁▁▁
   0.07    0.38   ▇▁▁▁▁▁▁▁
  0.064    0.32   ▇▁▁▁▁▁▁▁
  0.045    0.15   ▇▁▁▁▁▁▁▁
  0.058    0.97   ▇▁▁▁▁▁▁▁
  0.078    0.53   ▇▁▁▁▁▁▁▁
  0.075    0.39   ▇▁▁▁▁▁▁▁
  0.063    0.29   ▇▁▁▁▁▁▁▁
   0.06    0.19   ▇▁▁▁▁▁▁▁
  0.065    0.23   ▇▁▁▁▁▁▁▁
   0.05    0.15   ▇▁▁▁▁▁▁▁
  0.062    0.15   ▇▁▁▁▁▁▁▁
  0.057    0.37   ▇▁▁▁▁▁▁▁
  0.084    0.27   ▇▁▁▁▁▁▁▁
  0.074    0.41   ▇▁▁▁▁▁▁▁
  0.043    0.32   ▇▁▁▁▁▁▁▁
 0.0078   0.013   ▇▁▁▁▁▁▁▁
 0.0078   0.013   ▇▁▁▁▁▁▁▁
  0.094    0.45   ▇▁▁▁▁▁▁▁
  0.048    0.4    ▇▁▁▁▁▁▁▁
   0.06    0.27   ▇▁▁▁▁▁▁▁
```

```
df = rename(df, n = A, date = B) # дадим столбцам более осмысленные названия
```

```
df = na.omit(df) # уберем строки с пропущенными наблюдениями
```

Будем верить в CAPM :) Оценим параметры модели для компании MOTOR. Соответственно, зависимая переменная - разница доходностей акций MOTOR и безрискового актива, а регрессор - рыночная премия.

```
#создаем новые переменные и добавляем их к набору данных
df = mutate(df, y = MOTOR - RKFREE, x = MARKET - RKFREE)
```

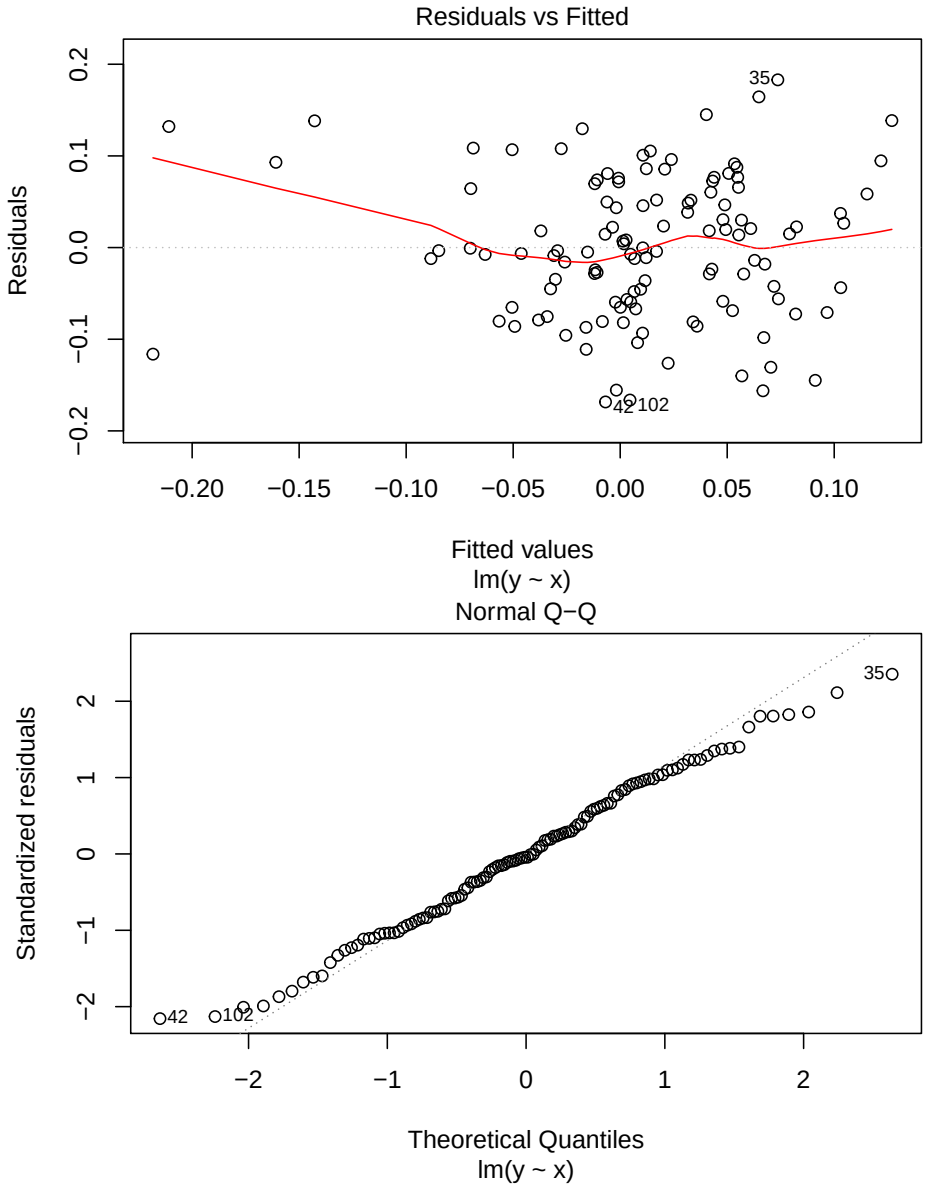
Строим нашу модель и проверяем гипотезу об адекватности регрессии.

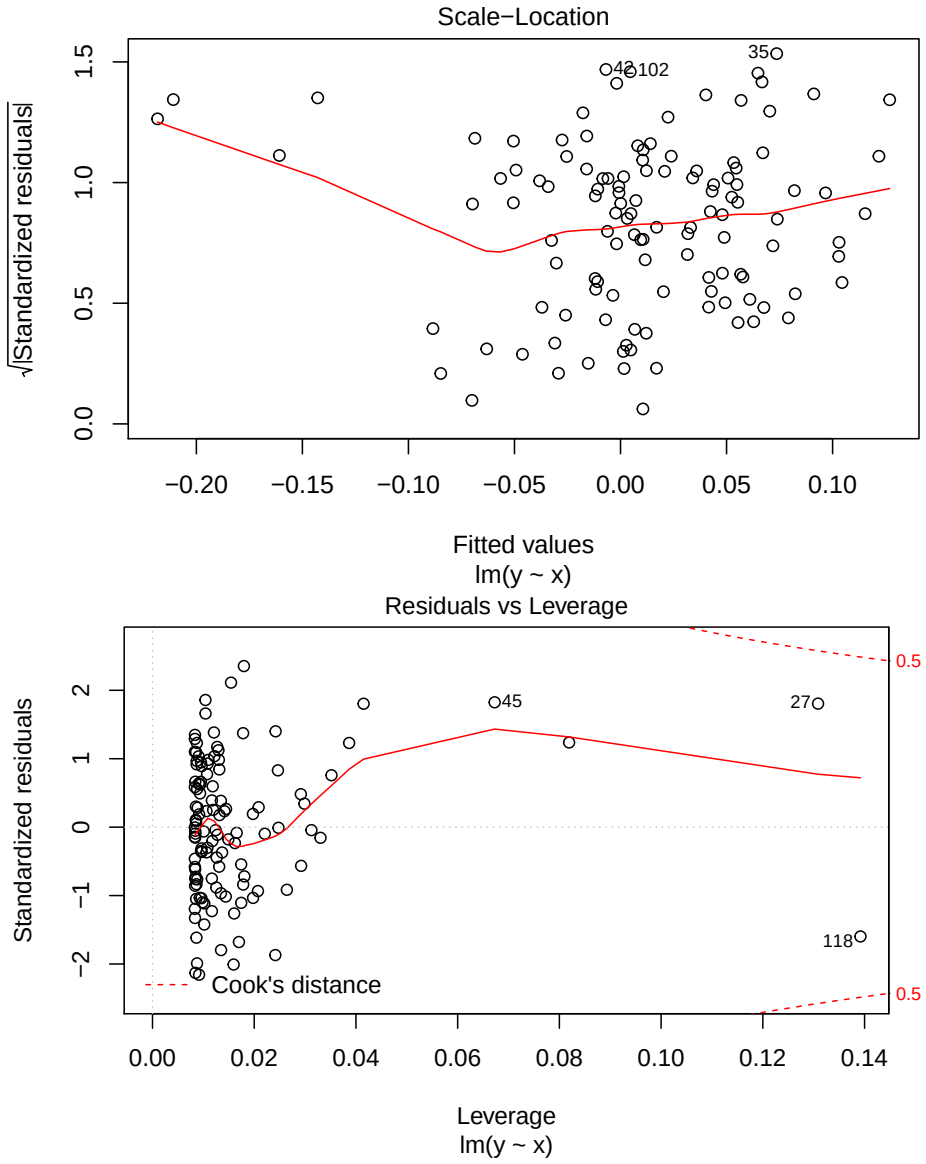
```
ols = lm(y ~ x, data = df)
summary(ols)
```

```
Call:
lm(formula = y ~ x, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.168421 -0.059381 -0.003399  0.061373  0.182991

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.005253   0.007200   0.730    0.467
x           0.848150   0.104814   8.092 5.91e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07844 on 118 degrees of freedom
Multiple R-squared:  0.3569,    Adjusted R-squared:  0.3514
F-statistic: 65.48 on 1 and 118 DF,  p-value: 5.913e-13
```

Вызовом одной функции получаем кучу полезных графиков. Можем визуально оценить наличие гетероскедастичности, нормальность распределения остатков, наличие выбросов.

```
plot(ols)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
35
42
102
Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x)



Строим доверительный интервал для параметров модели.

```
est = cbind(Estimate = coef(ols), confint(ols))
```

Проверим гипотезу о равенстве коэффициента при регрессоре единице.

```
linearHypothesis(ols, c("x = 1"))
```

Linear hypothesis test

Hypothesis:
x = 1

Model 1: restricted model
Model 2: y ~ x

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	119	0.73900			
2	118	0.72608	1	0.012915	2.0989	0.1501

Посмотрим на остатки :) Протестируем остатки регрессии на нормальность с помощью теста Харке-Бера.

$$H_0 : S = 0, K = 3,$$

где S — коэффициент асимметрии (Skewness), K — коэффициент эксцесса (Kurtosis)

```
jarque.bera.test(resid(ols))
```

Jarque Bera Test

data: resid(ols)
X-squared = 1.7803, df = 2, p-value = 0.4106

И тест Шапиро-Уилка.

$H_0 : \epsilon_i \sim N(\mu, \sigma^2)$

```
shapiro.test(resid(ols))
```

Shapiro-Wilk normality test

data: resid(ols)
W = 0.99021, p-value = 0.5531

Оба теста указывают на нормальность распределения остатков регрессии.

Сделаем прогноз модели по данным вне обучаемой выборки.

```
set.seed(7)
newData = df
newData = mutate(newData, x = x + rnorm(n = n())) # пошумим
yhat = predict(ols, newdata = newData, se = TRUE)
```

3.0.0.1. То же самое в стате

Загружаем данные.

use us-return.dta

end of do-file

Любуемся и даем новые названия столбцам.

```
summarize
ren A n
ren B date
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           A |       120        60.5    34.78505         1        120
           B |         0
       MOBIL |       120    .0161917    .0803075      -.178       .366
      TEXACO |       120    .0119417    .0797036      -.194       .399
         IBM |       120    .0096167     .059024      -.187        .15
-------------+--------------------------------------------------------
         DEC |       120     .01975    .0991438      -.364       .385
      DATGEN |       120    .0074833    .1275399      -.342       .528
       CONED |       120    .0185083    .0502719      -.139       .151
        PSNH |       120   -.0042167    .1094712      -.485       .318
       WEYER |       120    .0096333    .0850664      -.271        .27
-------------+--------------------------------------------------------
       BOISE |       120     .016675    .0974882      -.274       .379
       MOTOR |       120    .0181583    .0972656      -.331        .27
       TANDY |       120    .0250083     .127566      -.246       .454
       PANAM |       120    .0035167    .1318054      -.313       .406
       DELTA |       120    .0116917    .0959317       -.26       .289
-------------+--------------------------------------------------------
      CONTIL |       120     -.0011    .1506992        -.6       .974
      CITCRP |       120    .0118583    .0809719      -.282       .318
      GERBER |       120       .0164    .0877379      -.288       .234
      GENMIL |       120    .0165833    .0650403      -.148        .19
      MARKET |       120    .0139917    .0683532       -.26       .148
-------------+--------------------------------------------------------
      RKFREE |       120    .0068386    .0021869     .00207     .01255
      rkfree |       120    .0068386    .0021869     .00207     .01255
```

Убираем пропущенные значения и создаем новые переменные.

```
drop if n == .
gen y = MOTOR - RKFREE
gen x = MARKET - RKFREE
```

```
(2,544 observations deleted)
```

Строим модель и проверяем гипотезу об адекватности регрессии. Тут же получаем доверительные интервалы для коэффициентов.

```
reg y x
```

```
      Source |       SS           df       MS      Number of obs   =       120
-------------+----------------------------------   F(1, 118)       =     65.48
       Model |  .402913404         1  .402913404   Prob > F        =    0.0000
    Residual |  .726081541       118  .006153233   R-squared       =    0.3569
-------------+----------------------------------   Adj R-squared   =    0.3514
       Total |  1.12899494       119  .009487352   Root MSE        =    .07844

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .8481496   .1048138     8.09   0.000     .6405898    1.055709
       _cons |   .0052529   .0071999     0.73   0.467     -.009005    .0195107
------------------------------------------------------------------------------
```

Проверим гипотезу о равенстве коэффициента при регрессоре единице.

```
test x = 1
```

```
 ( 1)  x = 1

       F(  1,   118) =    2.10
            Prob > F =    0.1501
```

Сделаем предсказание по выборке и сохраним остатки.

```
predict u_hat, resid
predict y_hat
```

```
(option xb assumed; fitted values)
```

Протестируем остатки регрессии на нормальность с помощью теста Харке-Бера. На самом деле, это не совсем тест Харке-Бера. Оригинальный вариант ассимптотический и в нем нет поправки на размер выборки. В Stata есть. Подробнее здесь https://www.stata.com/manuals13/rsktest.pdf

```
sktest u_hat
```

```
                    Skewness/Kurtosis tests for Normality
                                                  ------ joint ------
    Variable |       Obs  Pr(Skewness)  Pr(Kurtosis) adj chi2(2)   Prob>chi2
-------------+-------------------------------------------------------------
       u_hat |       120     0.8841        0.1027        2.74         0.2539
```

И тест Шапиро-Уилка. Тут все аналогично R.

```
swilk u_hat
```

Shapiro-Wilk W test for normal data

Variable	Obs	W	V	z	Prob>z
u_hat	120	0.99021	0.942	-0.133	0.55310

Гипотеза о нормальности остатков не отвергается.

QQ - график

qnorm u_hat

График предсказанных значений против остатков.

```stata
rvfplot, yline(0)
```

График диагональных элементов матрицы-шляпницы против квадрата остатков (по сравнению с R оси поменялись местами).

lvr2plot

“““

График предсказанных значений против стандартизиованных остатков. Размер точек на графике зависит от расстояния Кука для данного наблюдения.

predict D, cooksd
predict standard, rstandard

graph twoway scatter standard y_hat [aweight=D], msymbol(oh) yline(0)

set seed 7

set obs 120
gen x_new = x+ 0.5 *rnormal()
gen y_hat_new = .8481496 * x_new+ .0052529

number of observations (_N) was 120, now 120

3.0.0.2. То же самое в python

Много хорошихх функций для статистических расчетов можно найти в пакете Statsmodels.

```
import pandas as pd # для работы с таблицами
import numpy as np # математика, работа с матрицами
```

```
import matplotlib.pyplot as plt # графики
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.graphics.gofplots as gf
from statsmodels.stats.outliers_influence import summary_table
import seaborn as sns # еще более классные графики
from scipy.stats import shapiro # еще математика
import statsmodels.discrete.discrete_model
```

При желании, можем кастомизировать графики :)

```
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=15)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=15)
```

Загрузим данные.

```
df = pd.read_stata('us-return.dta')
```

Избавимся от наблюдений с пропущенными значенями.

```
df.dropna(inplace=True) ##ИСПРАВИТЬ (выкинуть только пропуски целевой и объяснющей)
df.reset_index(drop=True, inplace=True)
```

Переименуем столбцы.

```
df = df.rename(columns={'A':'n', 'B': 'date'})
```

```
df['y'] = df['MOTOR'] - df['RKFREE']
df['x'] = df['MARKET'] - df['RKFREE']
```

Строим модель и читаем саммари :)

```
regr = smf.ols('y~x', data = df).fit()
regr.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.357
Model:                            OLS   Adj. R-squared:                  0.351
Method:                 Least Squares   F-statistic:                     65.48
Date:                Пн, 16 сен 2019   Prob (F-statistic):           5.91e-13
Time:                        11:15:11   Log-Likelihood:                136.18
No. Observations:                 120   AIC:                            -268.4
Df Residuals:                     118   BIC:                            -262.8
```

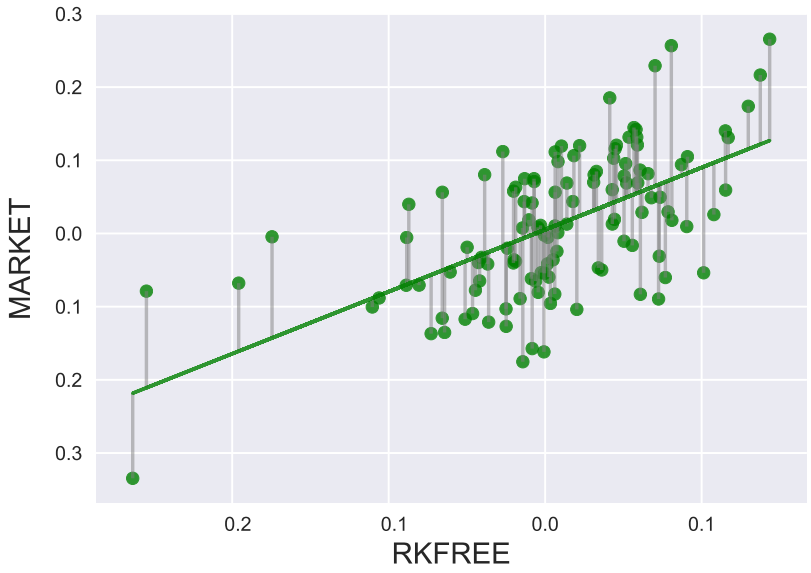
```
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0053      0.007      0.730      0.467      -0.009       0.020
x              0.8481      0.105      8.092      0.000       0.641       1.056
==============================================================================
Omnibus:                        2.684   Durbin-Watson:                   2.030
Prob(Omnibus):                  0.261   Jarque-Bera (JB):                1.780
Skew:                          -0.031   Prob(JB):                        0.411
Kurtosis:                       2.406   Cond. No.                         14.6
==============================================================================

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

Получить прогноз.

```
df['yhat'] = regr.fittedvalues
```

Красивые графики для остатков, выборосов и прочих радостей, как в R, придется строить ручками. Зато приятно поиграть с оформлением :)

```
fig, ax = plt.subplots()
ax.plot(df['x'],regr.fittedvalues, color='g', alpha =0.8)
ax.scatter(df['x'],regr.fittedvalues+regr.resid, color = 'g', alpha = 0.8, s = 40)
ax.vlines(df['x'],regr.fittedvalues,regr.fittedvalues+regr.resid, color = 'gray', alpha = 0.5)
plt.title('Линия регрессии и остатки')
plt.xlabel('RKFREE')
plt.ylabel('MARKET')
plt.show()
```



Строим доверительный интервал.

```
regr.conf_int()
```

```
                0         1
Intercept -0.009005  0.019511
x          0.640590  1.055709
```

И проведем F-test.

```
hypotheses = '(x = 1)'
regr.f_test(r_matrix = hypotheses)
```

```
<class 'statsmodels.stats.contrast.ContrastResults'>
<F test: F=array([[2.09891771]]), p=0.1500556415866233, df_denom=118, df_num=1>
```

Тест Шапиро. Такой же, как и в R. Для удобства можно поместить в табличку.

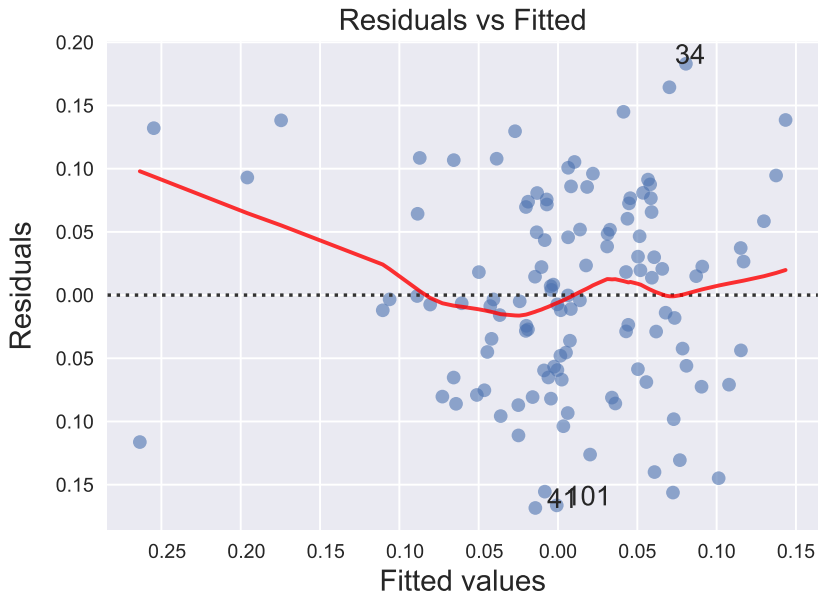
```
W, p_value = shapiro(regr.resid)
#pd.DataFrame(data = {'W': [round(W,3)], 'p_value': [round(p_value,3)]})
```

Генерируем новые данные и строим предсказание.

```
import random
random.seed(7)

newData = df['x'] + 0.5*np.random.normal(len(df))
prediction = regr.predict(newData)
```

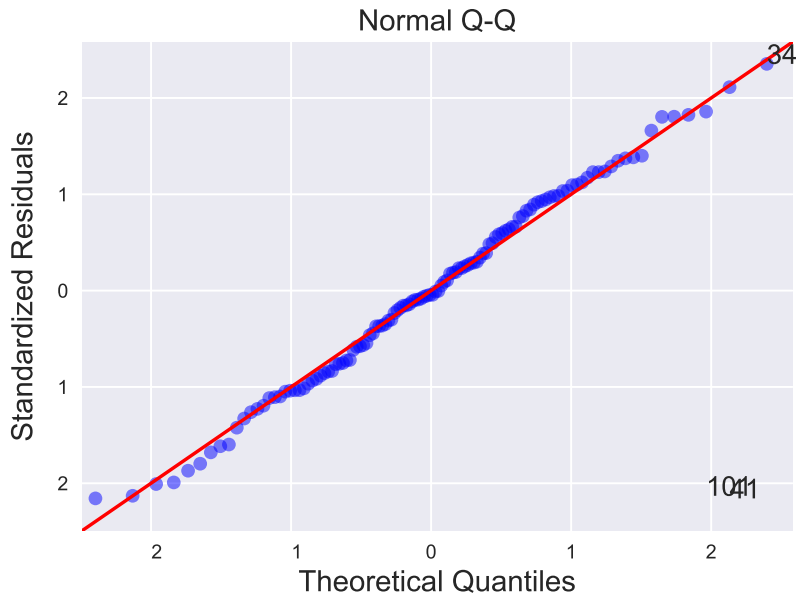
А теперь жесть! Построим графички, похожие на autoplot R.

```
fig_1 = plt.figure(1)

fig_1.axes[0] = sns.residplot(df['x'], df['y'],
                              lowess=True,
                              scatter_kws={'alpha': 0.6},
                              line_kws={'color': 'red', 'lw': 2, 'alpha': 0.8})

fig_1.axes[0].set_title('Residuals vs Fitted')
fig_1.axes[0].set_xlabel('Fitted values')
fig_1.axes[0].set_ylabel('Residuals')


#можем добавить метки потенциальных аутлаеров
abs_resid = abs(regr.resid).sort_values(ascending=False)
abs_resid_top3 = abs_resid[:3]

for i in abs_resid_top3.index:
    fig_1.axes[0].annotate(i,
                           xy=(regr.fittedvalues[i],
                               regr.resid[i]))
```



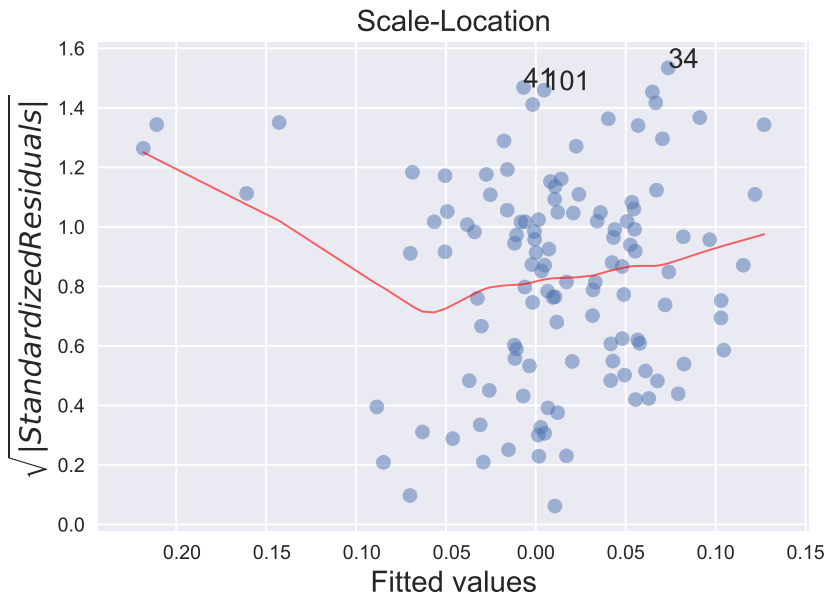

```
norm_residuals = regr.get_influence().resid_studentized_internal #сохраним стьюдентизированные остатки
```

```
QQ = gf.ProbPlot(norm_residuals)
fig_2 = QQ.qqplot(line='45', alpha=0.5, color='b', lw=1)


fig_2.axes[0].set_title('Normal Q-Q')
fig_2.axes[0].set_xlabel('Theoretical Quantiles')
fig_2.axes[0].set_ylabel('Standardized Residuals');

#и снова метки
abs_norm_resid = np.flip(np.argsort(abs(norm_residuals)), 0)
abs_norm_resid_top3 = abs_norm_resid[:3]

for r, i in enumerate(abs_norm_resid_top3):
    fig_2.axes[0].annotate(i,
                           xy=(np.flip(QQ.theoretical_quantiles, 0)[r],
                               norm_residuals[i]))
```




```
fig_3 = plt.figure(3)

plt.scatter(regr.fittedvalues, np.sqrt(abs(norm_residuals)), alpha=0.5)
sns.regplot(regr.fittedvalues, np.sqrt(abs(norm_residuals)),
            scatter=False,
            ci=False,
            lowess=True,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.6})
```

```
fig_3.axes[0].set_title('Scale-Location')
fig_3.axes[0].set_xlabel('Fitted values')
fig_3.axes[0].set_ylabel('$\sqrt{|Standardized Residuals|}$')

# и еще раз!)
abs_sq_norm_resid = np.flip(np.argsort(np.sqrt(abs(norm_residuals)), 0))
abs_sq_norm_resid_top3 = abs_sq_norm_resid[:3]

for i in abs_sq_norm_resid_top3:
    fig_3.axes[0].annotate(i, xy=(regr.fittedvalues[i],
                                  np.sqrt(abs(norm_residuals)[i])))
```




```
leverage = regr.get_influence().hat_matrix_diag #сохраняем элементы матрицы-шляпницы
cook_dist = regr.get_influence().cooks_distance[0] #И расстояние Кука

fig_4 = plt.figure(4)

plt.scatter(leverage, norm_residuals, alpha=0.5)
sns.regplot(leverage, norm_residuals,
            scatter=False,
            ci=False,
            lowess=True,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
```

```
fig_4.axes[0].set_xlim(0, 0.20)
```

(0, 0.2)

```
fig_4.axes[0].set_ylim(-3, 5)
```

(-3, 5)

```
fig_4.axes[0].set_title('Residuals vs Leverage')
fig_4.axes[0].set_xlabel('Leverage')
fig_4.axes[0].set_ylabel('Standardized Residuals')


leverage_top3 = np.flip(np.argsort(cook_dist), 0)[:3]

for i in leverage_top3:
    fig_4.axes[0].annotate(i,
                           xy=(leverage[i],
                               norm_residuals[i]))
plt.show()
```