# First, we create function that produces same diagnostic plots as R

- most of this code is from https://robert-alvarez.github.io/2018-06-04-diagnostic_plots/ (https://robert-alvarez.github.io/2018-06-04-diagnostic_plots/)

In [178]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.stats import diagnostic
from statsmodels.stats.outliers_influence import OLSInfluence
import scipy.stats as stats
import pandas as pd
from statsmodels.graphics.gofplots import ProbPlot
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import scipy
import math
import itertools

def graph(formula, x_range, label=None):
    """
    Helper function for plotting cook's distance lines
    """
    x = x_range
    y = formula(x)
    plt.plot(x, y, label=label, lw=1, ls='--', color='red')

def diagnostic_plots(df, response, model_fit=None):

    X = df.drop([response],  axis=1)
    y = pd.DataFrame(df[response])

    """
    Function to reproduce the 4 base plots of an OLS model in R.

    ---
    Inputs:

    X: A numpy array or pandas dataframe of the features to use in building t

    y: A numpy array or pandas series/dataframe of the target variable of the

    model_fit [optional]: a statsmodel.api.OLS model after regressing y on X.
                          generated from X, y
    """

#     if not model_fit:
#         model_fit = sm.OLS(y, sm.add_constant(X)).fit()

    # create dataframe from X, y for easier plot handling
    dataframe = pd.concat([X, y], axis=1)
    # model values
    model_fitted_y = model_fit.fittedvalues

    main_figure, axs = plt.subplots(2,2, figsize=(15,10))
    ax1 = axs[0,0]
    ax2 = axs[0,1]
    ax3 = axs[1,0]
    ax4 = axs[1,1]


    # ############### 1. Residuals X Fitted ################################
    # model residuals
    model_residuals = model_fit.resid
    # absolute residuals
    model_abs_resid = np.abs(model_residuals)
    # Plot
    sns.residplot(model_fitted_y, dataframe.columns[-1], data=dataframe, lowe
    # Labels
```

```
    ax1.set_title('Residuals vs Fitted')
    ax1.set_xlabel('Fitted values')
    ax1.set_ylabel('Residuals')
    # Annotations
    abs_resid = model_abs_resid.sort_values(ascending=False)
    abs_resid_top_3 = abs_resid[:3]
    for i in abs_resid_top_3.index:
        ax1.annotate(i, xy=(model_fitted_y[i], model_residuals[i]));



    # ############### 2. QQ ##################################################
    # normalized residuals
    model_norm_residuals = model_fit.get_influence().resid_studentized_intern
    # QQ plot
    QQ = ProbPlot(model_norm_residuals)
    plot_lm_2 = QQ.qqplot(line='45', alpha=0.5, color='#4C72B0', lw=1, ax=ax2
    # Labels
    ax2.set_title('Normal Q-Q')
    ax2.set_xlabel('Theoretical Quantiles')
    ax2.set_ylabel('Standardized Residuals');
    # Annotations
    abs_norm_resid = np.flip(np.argsort(np.abs(model_norm_residuals)), 0)
    abs_norm_resid_top_3 = abs_norm_resid[:3]
    for r, i in enumerate(abs_norm_resid_top_3):
        ax2.annotate(i, xy=(np.flip(QQ.theoretical_quantiles, 0)[r], model_no

    # ############### 3. Scale X Location ####################################

    # absolute squared normalized residuals
    model_norm_residuals_abs_sqrt = np.sqrt(np.abs(model_norm_residuals))
    # Plot
    ax3.scatter(model_fitted_y, model_norm_residuals_abs_sqrt, alpha=0.5);
    sns.regplot(model_fitted_y, model_norm_residuals_abs_sqrt, scatter=False,
    # Labels
    ax3.set_title('Scale-Location')
    ax3.set_xlabel('Fitted values')
    ax3.set_ylabel('$\sqrt{|Standardized Residuals|}$');
    # Annotations
    abs_sq_norm_resid = np.flip(np.argsort(model_norm_residuals_abs_sqrt), 0)
    abs_sq_norm_resid_top_3 = abs_sq_norm_resid[:3]
    for i in abs_norm_resid_top_3:
        ax3.annotate(i, xy=(model_fitted_y[i], model_norm_residuals_abs_sqrt[



    # ############### 4. Residuals X Leverage ################################
    # leverage, from statsmodels internals
    model_leverage = model_fit.get_influence().hat_matrix_diag
    # Plot
    ax4.scatter(model_leverage, model_norm_residuals, alpha=0.5);
    sns.regplot(model_leverage, model_norm_residuals, scatter=False, ci=False
    # Labels
    ax4.set_xlim(0, max(model_leverage)+0.01)
    ax4.set_ylim(-3, 5)
    ax4.set_title('Residuals vs Leverage')
    ax4.set_xlabel('Leverage')
    ax4.set_ylabel('Standardized Residuals');
    # cook's distance, from statsmodels internals
    model_cooks = model_fit.get_influence().cooks_distance[0]
    # Annotations
    leverage_top_3 = np.flip(np.argsort(model_cooks), 0)[:3]
    for i in leverage_top_3:
        ax4.annotate(i, xy=(model_leverage[i], model_norm_residuals[i]));
    p = len(model_fit.params) # number of model parameters
    graph(lambda x: np.sqrt((0.5 * p * (1 - x)) / x), np.linspace(0.001, max(
    graph(lambda x: np.sqrt((1 * p * (1 - x)) / x), np.linspace(0.001, max(mo
```

```
132     ax4.legend(loc='upper right');
```

# 1. Baseline regression

- We start by fitting all features agains dependent variable. No tranformations.
- We also plot diagnostic plots.

In [39]:

```
1 # Load the data
2 df = pd.read_csv("police.txt",delimiter="    ")
3 feature_names = ["height", "weight", "fat", "pulse", "diast"]
4 df.describe()
5 # The bottom number (diastolic) is the pressure as the heart relaxes between b
6 # reading of 120/80 mm Hg is considered normal
```

Out[39]:

| | react | height | weight | fat | pulse | diast |
|---|---|---|---|---|---|---|
| **count** | 50.000000 | 50.000000 | 50.00000 | 50.000000 | 50.000000 | 50.000000 |
| **mean** | 0.316200 | 178.106000 | 78.35240 | 13.782400 | 73.080000 | 74.600000 |
| **std** | 0.048215 | 7.024081 | 11.46837 | 7.347631 | 12.910745 | 8.113984 |
| **min** | 0.221000 | 163.100000 | 54.28000 | 2.010000 | 50.000000 | 48.000000 |
| **25%** | 0.282500 | 174.150000 | 69.74500 | 9.682500 | 64.000000 | 70.500000 |
| **50%** | 0.311500 | 178.850000 | 77.00000 | 12.220000 | 72.000000 | 74.000000 |
| **75%** | 0.343500 | 180.900000 | 86.70500 | 18.277500 | 80.000000 | 78.000000 |
| **max** | 0.427000 | 193.800000 | 102.26000 | 32.630000 | 104.000000 | 90.000000 |

In [3]:
```
# Fit & print summary
results = smf.ols('react ~ height + weight + fat + pulse + diast', data=dat).f
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  react   R-squared:                       0.249
Model:                            OLS   Adj. R-squared:                  0.164
Method:                 Least Squares   F-statistic:                     2.916
Date:                Tue, 23 Jun 2020   Prob (F-statistic):             0.0233
Time:                        20:32:52   Log-Likelihood:                 88.319
No. Observations:                  50   AIC:                            -164.6
Df Residuals:                      44   BIC:                            -153.2
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.3095      0.202     -1.529      0.133      -0.717       0.098
height         0.0041      0.001      3.198      0.003       0.002       0.007
weight        -0.0031      0.001     -2.431      0.019      -0.006      -0.001
fat            0.0035      0.002      2.217      0.032       0.000       0.007
pulse          0.0003      0.001      0.522      0.605      -0.001       0.001
diast          0.0010      0.001      1.214      0.231      -0.001       0.003
==============================================================================
Omnibus:                        1.703   Durbin-Watson:                   1.645
Prob(Omnibus):                  0.427   Jarque-Bera (JB):                1.667
Skew:                           0.380   Prob(JB):                        0.435
Kurtosis:                       2.528   Cond. No.                     7.19e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 7.19e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

**Output is waring us that there might be string multicollinerarity, so let's calculate coviariance matrix**

In [45]:
```
corrMatrix = df.drop(["react"], axis=1).corr()
print (corrMatrix)
```

```
          height    weight       fat     pulse     diast
height  1.000000  0.648838  0.374040 -0.157647 -0.165856
weight  0.648838  1.000000  0.809507 -0.263843 -0.051700
fat     0.374040  0.809507  1.000000 -0.094830  0.044647
pulse  -0.157647 -0.263843 -0.094830  1.000000  0.234088
diast  -0.165856 -0.051700  0.044647  0.234088  1.000000
```

**Yes, we can see that there is linear dependenece between:**

- weight - height
- fat - weight - height
- pulse - diast

**Let us also plot each x_i vs residials**

In [224]:

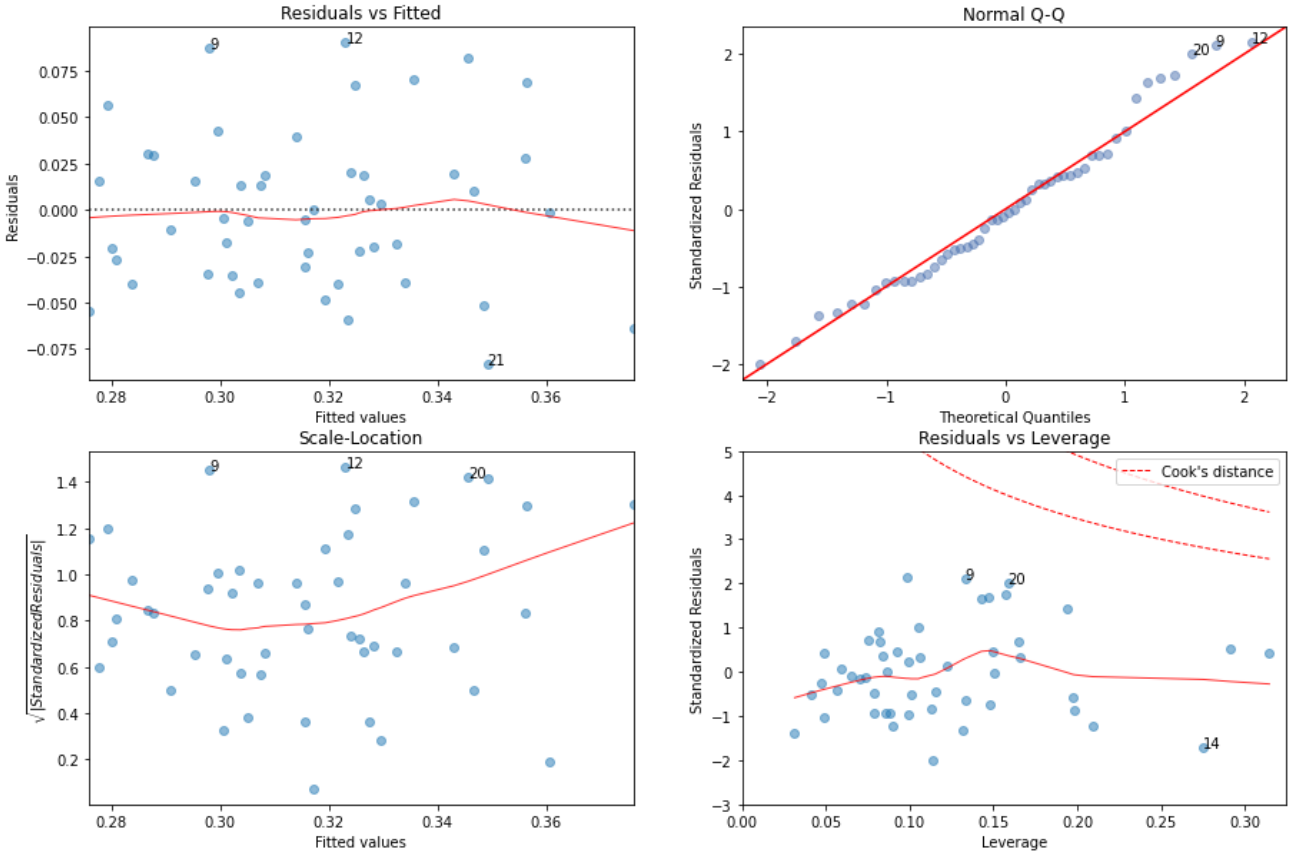
```
# model_residuals = results.resid
# main_figure, axs = plt.subplots(2, 3, figsize=(20,10))

# row = -1
# for ii, feature in enumerate(feature_names):
#     col = ii % 3
#     if col == 0:
#         row +=1
#     x_i = X[feature]
#     ax = axs[row,col]
#     # absolute residuals
#     model_abs_resid = np.abs(model_residuals)
#     # Plot
#     sns.residplot(x_i, df.columns[-1], data=df, lowess=True, scatter_kws={'a
#     # Labels
#     ax.set_title('Residuals vs {}'.format(feature))
#     ax.set_xlabel(feature)
#     ax.set_ylabel('Residuals')
#     # Annotations
#     abs_resid = model_abs_resid.sort_values(ascending=False)
#     abs_resid_top_3 = abs_resid[:3]
#     for i in abs_resid_top_3.index:
#         ax.annotate(i, xy=(x_i[i], model_residuals[i]));
```

## 1.1 Diagnostic Plots

In [48]:

```
diagnostic_plots(df, "react", results)
```




## 1.1.1 Description of plots

**Residual-Fitted**

- There is clear linear dependence between $e_i$ and $haty_i$

- Seems that $y$ and $X$ are lineary dependent
- Bottomline: Looks good

#### Normal Q-Q

- Almost perfectly diagonal
- Suggests that residuals are normaly distributed
- Bottomline: Looks good

#### Scale-Location

- There is slight curve
- Suggest slight heteroscedasticity
- Bottomline: Doesn't looks good

#### Residuals-Leverage

- All points with small Cook's distance
- No highly influential points - Suggest no outliers that have influence on the regression
- Bottomline: Looks good

## 1.2 Tests of Normality and Homoscedascticity

- Plots suggest that assumptions about the errors ( normality and homoscedasticity ) which are required for furhher analysis of the regressors holds
- I am only afraid about non-constant variance
- So let's perform Levene's test for homoscedascticity
- And also Shapiro-Wilk's test for normality

In [7]:
```
def perform_test(name, p,a,h0,h1):
    print(name)
    print("H0: {}".format(h0))
    print("H1: {}".format(h1))
    print("----")
    print("alpha: {}".format(a))
    print("p_val: {:.3f}".format(p))
    if p > a:
        print("Not rejecting H0")
    else:
        print("Rejecting H0 in favor of H1")
```

### 1.2.1 Levene test

In [8]:
```
residuals = results.resid.tolist()
residuals.sort()
half = math.floor(len(residuals) / 2)
a = residuals[:half]
b = residuals[half:]
p_val = scipy.stats.levene(a,b)[1]
perform_test("Levene test for equal variance", p_val, 0.05, "v1==v2", "v1!=v2"
```

```
Levene test for equal variance
H0: v1==v2
H1: v1!=v2
----
alpha: 0.05
p_val: 0.190
Not rejecting H0
```

### 1.2.2 Shapiro-Wilk test

In [9]:
```
p_val = scipy.stats.shapiro(residuals)[1]
perform_test("Shapiro-Wilk test for normality", p_val, 0.05, "F(x) ~ N", "F(x)
```

```
Shapiro-Wilk test for normality
H0: F(x) ~ N
H1: F(x) not from N
----
alpha: 0.05
p_val: 0.285
Not rejecting H0
```

**It seems that both assumptions hold and we can continue with our analysis**

## 1.3 Analysis of the explanation variables

In [10]:
```
1 <f<print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  react   R-squared:                       0.249
Model:                            OLS   Adj. R-squared:                  0.164
Method:                 Least Squares   F-statistic:                     2.916
Date:                Tue, 23 Jun 2020   Prob (F-statistic):             0.0233
Time:                        20:35:24   Log-Likelihood:                 88.319
No. Observations:                  50   AIC:                            -164.6
Df Residuals:                      44   BIC:                            -153.2
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.3095      0.202     -1.529      0.133      -0.717       0.098
height         0.0041      0.001      3.198      0.003       0.002       0.007
weight        -0.0031      0.001     -2.431      0.019      -0.006      -0.001
fat            0.0035      0.002      2.217      0.032       0.000       0.007
pulse          0.0003      0.001      0.522      0.605      -0.001       0.001
diast          0.0010      0.001      1.214      0.231      -0.001       0.003
==============================================================================
Omnibus:                        1.703   Durbin-Watson:                   1.645
Prob(Omnibus):                  0.427   Jarque-Bera (JB):                1.667
Skew:                           0.380   Prob(JB):                        0.435
Kurtosis:                       2.528   Cond. No.                     7.19e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.19e+03. This might indicate that there are strong multicollinearity or other numerical problems.
```

### 1.3.1 Interpretation of regression coefficients

**Intercept**

- Doesn't make sense to explain. It's not probbable that we would have some cop with zero height and weight.
- Moreover if some cop would have zero pulse I assume reaction time will go to infinity.

**height**

- Is statisticall significant (small p_val = 0.003) < alpha=0.05
- If it would be possible to fix other variables (which is not possible due to multi-collinearity between height-weight-fat) ..
- ...then if someone is 1cm taller, his reaction time increases by 0.0041 seconds (10cm ~ 0.04sec)

**weight**

- Again, statistically significant (small p_val = 0.019)
- Again we have problem with multi-collinearity
- But is seems that is someone is 1kg heavier, than his reaction time decreases by -0.0031 sec
- that seems strange to me, I would assume otherwise

**fat**

- Statistically significnat (small p_val = 0.032)
- Again we have problem with multi-collinearity
- If someone have 1% more bodyfat, then his reaction time increases by 0.0035 sec

- That also seems strange to me

**pulse** and **diast**

- both seems statistically insignificant (high p_vals)

## 1.4 Model quality

- Adj. R-squared: 0.164 is very small, model does not seem to be ideal
- Let us try to improve it

# 2. Transformations

## 2.1 First expand our data with transformations

- $\forall z_i, i \in 1 \dots p$
- $log(z_i)$
- $(z_i)^2$
- $(z_i)^{1/2}$

In [208]:
```
data_with_transformations = df.copy()
feature_names = ["height", "weight", "fat", "pulse", "diast"]
transformations = ["log", "pow2", "sqrt"]
for feature in feature_names:
    for transformation in transformations:
        if transformation == "pow2":
            data_with_transformations[feature + "_" + transformation] = np.pow
        if transformation == "sqrt":
            data_with_transformations[feature + "_" + transformation] = np.sqr
        if transformation == "log":
            data_with_transformations[feature + "_" + transformation] = np.log
```

## 2.2 We can also add some interactions

- BMI = weight / (height)^2
- fat_kg = fat * weight
- fat_BMI = fat * BMI
- I have tried these intercations, but they introduced problems to the assumption about the errors, so I did not used them for the final model

In [220]:
```
# data_with_transformations["BMI"] =  data_with_transformations["weight"] / np
# data_with_transformations["fat_kg"] =  data_with_transformations["fat"] * da
# data_with_transformations["fat_BMI"] =  data_with_transformations["fat"] * d
```

## 2.2 Now fit OLS on all combinations (with some contraints)

In [210]:

```
def brute_force_select(data, response, adj_r_threshold):
    # create combinations
    lst = data.columns.tolist()
    lst.remove(response)
    combs = []
    for i in range(1, len(lst)+1):
        els = [list(x) for x in itertools.combinations(lst, i)]
        combs.extend(els)

    # filter combs
    filtered_combs = []
    for comb in combs:
        bad = False
        for x in lst:
            if (x + "_pow2" in comb and not x in comb) or  (x in comb and  x +
                bad = True
        if not bad:
            filtered_combs.append(comb)

    # fit all combinations
    results = []
    print("fitting {} variants".format(len(filtered_combs)))
    for i, comb in enumerate(filtered_combs):
        formula = "{} ~ {}".format(response, ' + '.join(comb))
#         if i > 0 and i%1000 == 0:
#             print("{}/{}: {}".format(i+1, len(filtered_combs), formula))
        model = smf.ols(formula, data).fit()
        score = model.rsquared_adj
        if score > adj_r_threshold:
            results.append((score, model))

    # sort
    sorted_res = sorted(results,  key=lambda res: res[0], reverse=True)
    return sorted_res
```

In [214]:

```
threshold = 0.36
results = brute_force_select(data_with_transformations, "react", adj_r_thresho
print("{} models with adj. R^2 > {}".format(len(results), threshold))
```

```
fitting 3124 variants
34 models with adj. R^2 > 0.36
```

## 2.3 Now we select the best model

- I have analysed all diagnostic plots of models wtih Adj. R-squared: > 0.36
- Most of the diagnostic plots were quite similar so I selected the model with highest Adj. R-squared.
- Best models is: react ~ height + height_pow2 + weight_log + fat + diast

In [233]:

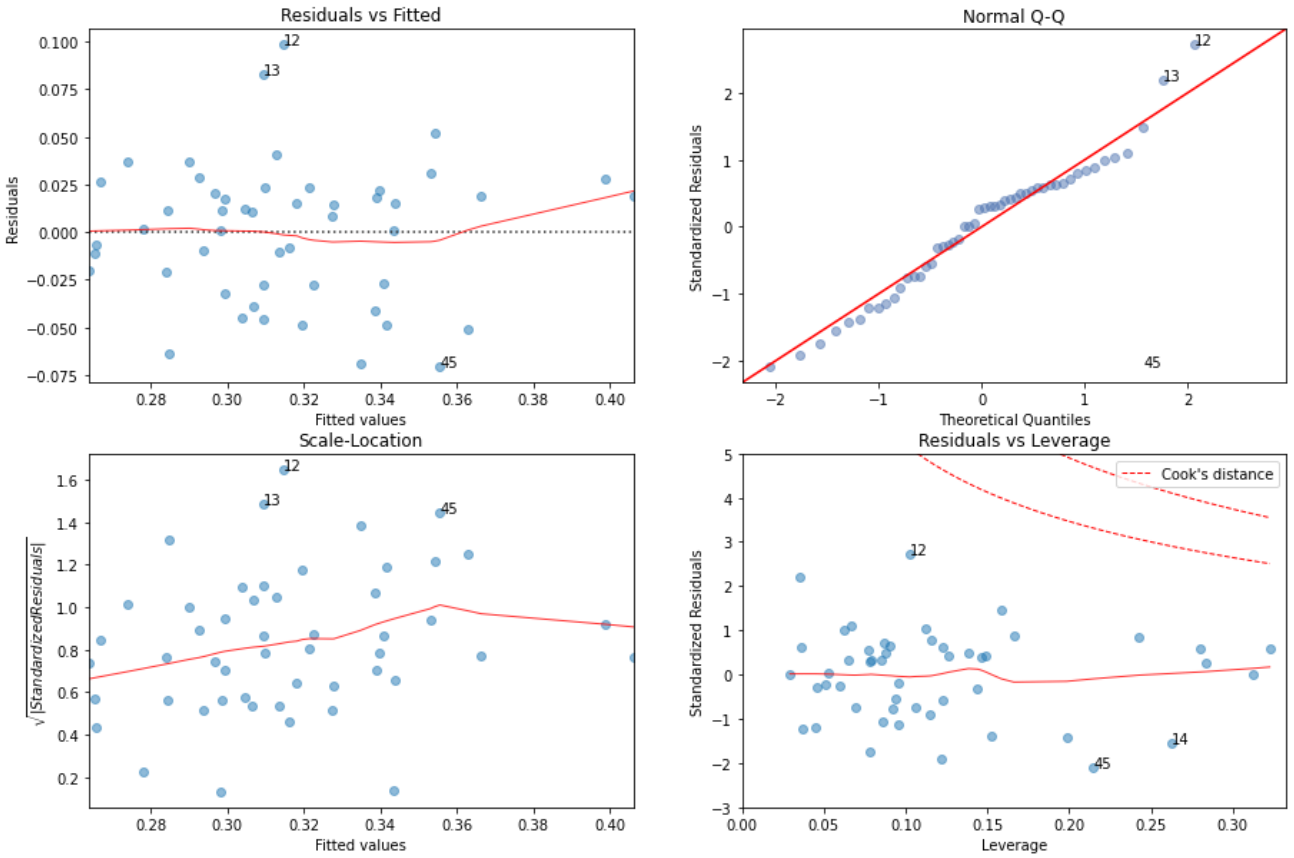
```
score, model = results[0]
print("model: {}".format(model.model.formula))
print("adj.R^2: {:.3f}".format(score))
```

```
model: react ~ height + fat + diast + height_pow2 + weight_log
adj.R^2: 0.372
```

## 2.4 Diagnostic plots + assumptions

In [156]:

```
used_columns = model.model.formula[8:].replace(" ", "").split("+")
used_columns.append("react")
data =  data_with_transformations[used_columns]
diagnostic_plots(data, "react", model)
```




- we can see that all plots looks quite good
- I allready described them for the baseline model and there are almost the same

### 2.4.1 Now we perform Leneve and Shapiro-Wilk test

In [242]:

```
residuals = model.resid.tolist()
residuals.sort()
half = math.floor(len(residuals) / 2)
a = residuals[:half]
b = residuals[half:]
p_val = scipy.stats.levene(a,b)[1]
perform_test("Levene test for equal variance", p_val, 0.05, "v1==v2", "v1!=v2"
print("\n")

p_val = scipy.stats.shapiro(residuals)[1]
perform_test("Shapiro-Wilk test for normality", p_val, 0.05, "eps ~ N", "eps r
```

```
Levene test for equal variance
H0: v1==v2
H1: v1!=v2
----
alpha: 0.05
p_val: 0.158
Not rejecting H0


Shapiro-Wilk test for normality
H0: eps ~ N
H1: eps not from N
----
alpha: 0.05
p_val: 0.170
Not rejecting H0
```

- We can see that we can't reject that errors are normaly distributed with contant variance

## 2.5 Interpretation of coefficients

- Které vysvětlující proměnné mají statisticky signifikantní vliv na reakční dobu?
- Jaká je interpretace jednotlivých regresních koeficientů?
- Co znamenají čísla uvedená v počítačovém výstupu?

In [225]:
```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  react   R-squared:                       0.436
Model:                            OLS   Adj. R-squared:                  0.372
Method:                 Least Squares   F-statistic:                     6.795
Date:                Wed, 24 Jun 2020   Prob (F-statistic):           8.99e-05
Time:                        16:15:43   Log-Likelihood:                 95.467
No. Observations:                  50   AIC:                            -178.9
Df Residuals:                      44   BIC:                            -167.5
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      9.2383      2.493      3.706      0.001       4.215      14.262
height        -0.0934      0.028     -3.311      0.002      -0.150      -0.037
fat            0.0034      0.001      2.560      0.014       0.001       0.006
diast          0.0015      0.001      2.160      0.036       0.000       0.003
height_pow2    0.0003   7.88e-05      3.467      0.001       0.000       0.000
weight_log    -0.2585      0.080     -3.226      0.002      -0.420      -0.097
==============================================================================
Omnibus:                        0.450   Durbin-Watson:                   1.672
Prob(Omnibus):                  0.798   Jarque-Bera (JB):                0.122
Skew:                           0.112   Prob(JB):                        0.941
Kurtosis:                       3.090   Cond. No.                     1.47e+07
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.47e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

**Intercept**

- Doesn't make sense to explain

**height** and **height ^2**

- Both statistically significant
- They are tied together
- y = fixed + b_1 * height + b_2 * height^2
- if height increases by 1cm we get
- y_new = fixed + b_1 * height + b_1 + b_2 * height^2 + 2 * b_2 * height + b_2
- y_new - y = 2 * b_1 * height + b_1 + b_2
- y_new - y = 2 * 0.0003 * height + 0.0003 - 0.0934
- y_new - y = 0.0006 * height - 0.0931
- Will this be increase or decrease ??
- 0.0006 * height - 0.0931 = 0 .... height ~ 155.2
- So if we assume only cops higher than 155cm (I would assume most of the cops are...)
- Then if a cop is 1cm taller, then his reaction time INCRASES by
- So if cop is 1 cm taller than other cop, then his reaction time is (0.0006 * height - 0.0931) seconds slower than the other cop
- Example: If cop is 180 cm high, then if there exist a cop with the same specifications but only 181cm high, then his reaction time is 0.0006 * 181 - 0.0931 = 0.0155 seconds slower

**fat**

- coeff 0.0034

- std_err 0.001
- p_val 0.014 ... significant alpha=0.05
- conf. interval [0.001, 0.006]
- Statistically significnat
- If someone have 1% more bodyfat, then his reaction time increases by 0.0032 sec
- That also seems strange to me I would asume otherwise
- It is possibly due to multicollinearity

**diast**

- coeff 0.0015
- std_err 0.001
- p_val 0.036.. significant alpha=0.05
- conf. interval [0.000, 0.003]
- Least significant from all the coefficients, but still significant
- Looks like increasing diastolic blood presure leads to slower reaction time

**weight_log**

- coeff -0.2585
- std_err 0.080
- p_val 0.002 ... significant alpha=0.05
- conf. interval [-0.420, -0.097]
- If log(weight) increases by one then the reaction time decreases by 0.2585 sec
- That means if weight increases about e=2.72 times
- There is some multicollinearity with height and with fat so in real word, it is hard to fix other params...

## 2.6 Outliers

- najděte pozdezřelá (vlivná nebo odlehlá) pozorování
- okomentujte splnění (nebo nesplnění) jednotlivých předpokladů lineárního regresního modelu.

---

- hight leverage points
  - remote X , does not affect estime (good leverage)
  - bud does not affect goodness
- high influence
  - impact estimate to shitty direction
  - exclucion -> regression looks a lot of different

- outlying in X --> measure leverage
- How much regression coeff changes is i-th deleted: DFBETA
- How much Y value changes is i-th deleted: DFFITS
- overall measure -- cook distance
- COVRATIO --- impact of data i on overall precision (> 1 --> data point co zlepsuje) (< 1 zhorsuje ...)

In [ ]:
```
# DFBEATS
# DFFITS
# HAT_DIAG
# COOK'S DISTANCE
# COV RATIO
```

In [304]:

```
ers_ols = model.get_influence()
influences = ers_ols.summary_frame()
influences = np.round(influences, decimals=3)
influences["cov_ratio"] = ers_ols.cov_ratio
influences = influences.drop(['standard_resid', 'student_resid', "dffits_inter
influences
```

Out[304]:

| | dfb_Intercept | dfb_height | dfb_fat | dfb_diast | dfb_height_pow2 | dfb_weight_log | cooks_d | hat_diag | df |
|---|---|---|---|---|---|---|---|---|---|
| **0** | -0.040 | 0.044 | 0.018 | -0.065 | -0.043 | -0.027 | 0.001 | 0.079 | 0.0 |
| **1** | -0.036 | 0.045 | -0.011 | 0.040 | -0.042 | -0.092 | 0.007 | 0.091 | 0.2 |
| **2** | -0.410 | 0.406 | -0.085 | -0.355 | -0.394 | -0.036 | 0.083 | 0.199 | -0.7 |
| **3** | 0.031 | -0.030 | -0.010 | 0.092 | 0.033 | -0.039 | 0.003 | 0.144 | -0.1 |
| **4** | -0.035 | 0.064 | 0.128 | -0.088 | -0.052 | -0.287 | 0.026 | 0.166 | 0.3 |
| **5** | 0.119 | -0.110 | 0.336 | -0.161 | 0.117 | -0.095 | 0.069 | 0.159 | 0.6 |
| **6** | 0.004 | -0.003 | -0.000 | -0.009 | 0.004 | -0.001 | 0.000 | 0.313 | 0.0 |
| **7** | -0.089 | 0.092 | -0.093 | -0.144 | -0.089 | -0.020 | 0.014 | 0.116 | 0.2 |
| **8** | 0.244 | -0.252 | -0.030 | 0.122 | 0.259 | 0.015 | 0.022 | 0.280 | 0.3 |
| **9** | 0.306 | -0.287 | 0.025 | 0.062 | 0.286 | -0.110 | 0.028 | 0.323 | 0.4 |
| **10** | 0.049 | -0.054 | -0.048 | 0.026 | 0.050 | 0.078 | 0.004 | 0.077 | 0.1 |
| **11** | -0.005 | 0.027 | 0.178 | -0.099 | -0.030 | -0.130 | 0.014 | 0.067 | 0.2 |
| **12** | -0.180 | 0.119 | -0.411 | 0.764 | -0.117 | 0.327 | 0.140 | 0.103 | 0.9 |
| **13** | -0.114 | 0.125 | -0.001 | 0.102 | -0.126 | -0.094 | 0.029 | 0.035 | 0.4 |
| **14** | 0.145 | -0.183 | -0.724 | -0.338 | 0.175 | 0.428 | 0.143 | 0.263 | -0.9 |
| **15** | 0.089 | -0.091 | -0.046 | 0.090 | 0.089 | 0.025 | 0.005 | 0.146 | 0.1 |
| **16** | -0.032 | 0.021 | -0.102 | -0.100 | -0.027 | 0.138 | 0.008 | 0.087 | 0.2 |
| **17** | -0.031 | 0.029 | 0.028 | 0.007 | -0.030 | 0.011 | 0.001 | 0.078 | 0.0 |
| **18** | -0.028 | 0.024 | 0.026 | 0.104 | -0.025 | 0.024 | 0.004 | 0.127 | 0.1 |
| **19** | -0.030 | 0.015 | -0.140 | 0.119 | -0.015 | 0.100 | 0.007 | 0.138 | 0.1 |
| **20** | 0.313 | -0.319 | 0.024 | 0.088 | 0.328 | 0.000 | 0.038 | 0.243 | 0.4 |
| **21** | 0.220 | -0.287 | -0.411 | 0.140 | 0.272 | 0.615 | 0.085 | 0.122 | -0.7 |
| **22** | -0.119 | 0.090 | -0.119 | -0.032 | -0.097 | 0.271 | 0.022 | 0.112 | 0.3 |
| **23** | 0.006 | -0.005 | -0.008 | 0.003 | 0.007 | -0.021 | 0.001 | 0.060 | -0.0 |
| **24** | -0.008 | -0.005 | -0.047 | 0.088 | 0.009 | 0.053 | 0.011 | 0.045 | -0.2 |
| **25** | 0.036 | -0.032 | -0.018 | -0.030 | 0.031 | -0.005 | 0.002 | 0.085 | 0.1 |
| **26** | 0.034 | -0.013 | 0.179 | -0.015 | 0.014 | -0.167 | 0.008 | 0.123 | -0.2 |
| **27** | 0.108 | -0.101 | 0.027 | -0.060 | 0.099 | -0.015 | 0.005 | 0.283 | 0.1 |
| **28** | 0.023 | 0.006 | 0.263 | -0.069 | -0.007 | -0.210 | 0.017 | 0.115 | -0.3 |
| **29** | 0.061 | -0.030 | 0.330 | -0.411 | 0.025 | -0.140 | 0.057 | 0.152 | -0.5 |
| **30** | 0.108 | -0.105 | 0.057 | -0.034 | 0.102 | 0.004 | 0.010 | 0.037 | -0.2 |
| **31** | -0.021 | 0.029 | 0.017 | -0.010 | -0.024 | -0.083 | 0.004 | 0.088 | 0.1 |
| **32** | 0.022 | -0.018 | 0.032 | 0.024 | 0.019 | -0.043 | 0.001 | 0.096 | -0.0 |
| **33** | -0.102 | 0.118 | 0.155 | 0.120 | -0.102 | -0.269 | 0.043 | 0.078 | -0.5 |
| **34** | -0.062 | 0.065 | 0.017 | -0.033 | -0.064 | -0.030 | 0.002 | 0.036 | 0.1 |
| **35** | -0.049 | 0.061 | 0.155 | 0.020 | -0.059 | -0.118 | 0.005 | 0.149 | 0.1 |

| | dfb_Intercept | dfb_height | dfb_fat | dfb_diast | dfb_height_pow2 | dfb_weight_log | cooks_d | hat_diag | df |
|---|---|---|---|---|---|---|---|---|---|
| **36** | -0.004 | 0.007 | -0.110 | -0.022 | -0.008 | 0.000 | 0.011 | 0.106 | -0.2 |
| **37** | 0.077 | -0.080 | -0.144 | -0.020 | 0.080 | 0.040 | 0.010 | 0.092 | -0.2 |
| **38** | -0.036 | 0.038 | 0.012 | -0.059 | -0.039 | 0.004 | 0.001 | 0.065 | 0.0 |
| **39** | 0.012 | -0.016 | -0.027 | 0.009 | 0.016 | 0.031 | 0.000 | 0.051 | -0.0 |
| **40** | -0.119 | 0.114 | 0.048 | -0.082 | -0.118 | 0.064 | 0.011 | 0.062 | 0.2 |
| **41** | 0.026 | -0.030 | -0.018 | 0.008 | 0.029 | 0.033 | 0.001 | 0.046 | -0.0 |
| **42** | 0.109 | -0.138 | -0.177 | -0.018 | 0.129 | 0.293 | 0.023 | 0.096 | -0.3 |
| **43** | 0.061 | -0.051 | 0.075 | 0.077 | 0.054 | -0.117 | 0.005 | 0.094 | -0.1 |
| **44** | 0.062 | -0.073 | -0.000 | 0.040 | 0.070 | 0.089 | 0.007 | 0.069 | -0.2 |
| **45** | -0.628 | 0.688 | 0.420 | 0.024 | -0.700 | -0.439 | 0.198 | 0.215 | -1.1 |
| **46** | -0.001 | 0.001 | -0.000 | 0.000 | -0.001 | 0.000 | 0.000 | 0.029 | 0.0 |
| **47** | -0.004 | 0.004 | -0.007 | -0.003 | -0.004 | 0.007 | 0.000 | 0.053 | 0.0 |
| **48** | 0.034 | -0.031 | 0.160 | -0.116 | 0.030 | -0.009 | 0.018 | 0.086 | -0.3 |
| **49** | -0.036 | 0.037 | 0.002 | 0.179 | -0.033 | -0.061 | 0.009 | 0.122 | 0.2 |

## Let's now use the criterions

In [305]:

```
p = 6
n = 50

# dfb > 1
for i in influences.columns.tolist():
    if i.startswith("dfb_"):
        influences[i] = np.abs(influences[i]) > 1

# cooks_d
# F = influences["cooks_d"]
# scipy.stats.f.cdf(F, p, n-p) > 0.5
influences["cooks_d"] = scipy.stats.f.cdf(influences["cooks_d"], p, n-p) > 0.5

# hat_diag
influences["hat_diag"] = influences["hat_diag"] > 3*p/n


# dffits 3 * sqrt( p/(n-p))
influences["dffits"] = np.abs(influences["dffits"]) > 3 * math.sqrt(p/(n-1))


# cov_ratio
influences["cov_ratio"] = np.abs(1 - influences["cov_ratio"]) > 3*p/(n-p)

influences
```

Out[305]:

| | dfb_Intercept | dfb_height | dfb_fat | dfb_diast | dfb_height_pow2 | dfb_weight_log | cooks_d | hat_diag | dffi |
|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | Fal |
| **1** | False | False | False | False | False | False | False | False | Fal |
| **2** | False | False | False | False | False | False | False | False | Fal |
| **3** | False | False | False | False | False | False | False | False | Fal |
| **4** | False | False | False | False | False | False | False | False | Fal |
| **5** | False | False | False | False | False | False | False | False | Fal |
| **6** | False | False | False | False | False | False | False | False | Fal |
| **7** | False | False | False | False | False | False | False | False | Fal |
| **8** | False | False | False | False | False | False | False | False | Fal |
| **9** | False | False | False | False | False | False | False | False | Fal |
| **10** | False | False | False | False | False | False | False | False | Fal |
| **11** | False | False | False | False | False | False | False | False | Fal |
| **12** | False | False | False | False | False | False | False | False | Fal |
| **13** | False | False | False | False | False | False | False | False | Fal |
| **14** | False | False | False | False | False | False | False | False | Fal |
| **15** | False | False | False | False | False | False | False | False | Fal |
| **16** | False | False | False | False | False | False | False | False | Fal |
| **17** | False | False | False | False | False | False | False | False | Fal |
| **18** | False | False | False | False | False | False | False | False | Fal |
| **19** | False | False | False | False | False | False | False | False | Fal |
| **20** | False | False | False | False | False | False | False | False | Fal |
| **21** | False | False | False | False | False | False | False | False | Fal |
| **22** | False | False | False | False | False | False | False | False | Fal |
| **23** | False | False | False | False | False | False | False | False | Fal |

| | dfb_Intercept | dfb_height | dfb_fat | dfb_diast | dfb_height_pow2 | dfb_weight_log | cooks_d | hat_diag | dffi |
|---|---|---|---|---|---|---|---|---|---|
| **24** | False | False | False | False | False | False | False | False | Fal |
| **25** | False | False | False | False | False | False | False | False | Fal |
| **26** | False | False | False | False | False | False | False | False | Fal |
| **27** | False | False | False | False | False | False | False | False | Fal |
| **28** | False | False | False | False | False | False | False | False | Fal |
| **29** | False | False | False | False | False | False | False | False | Fal |
| **30** | False | False | False | False | False | False | False | False | Fal |
| **31** | False | False | False | False | False | False | False | False | Fal |
| **32** | False | False | False | False | False | False | False | False | Fal |
| **33** | False | False | False | False | False | False | False | False | Fal |
| **34** | False | False | False | False | False | False | False | False | Fal |
| **35** | False | False | False | False | False | False | False | False | Fal |
| **36** | False | False | False | False | False | False | False | False | Fal |
| **37** | False | False | False | False | False | False | False | False | Fal |
| **38** | False | False | False | False | False | False | False | False | Fal |
| **39** | False | False | False | False | False | False | False | False | Fal |
| **40** | False | False | False | False | False | False | False | False | Fal |
| **41** | False | False | False | False | False | False | False | False | Fal |
| **42** | False | False | False | False | False | False | False | False | Fal |
| **43** | False | False | False | False | False | False | False | False | Fal |
| **44** | False | False | False | False | False | False | False | False | Fal |
| **45** | False | False | False | False | False | False | False | False | Tru |
| **46** | False | False | False | False | False | False | False | False | Fal |
| **47** | False | False | False | False | False | False | False | False | Fal |
| **48** | False | False | False | False | False | False | False | False | Fal |
| **49** | False | False | False | False | False | False | False | False | Fal |

## DFBEATS

- All data points low value
- Thus bo coeff will be dramatically changed if some data point is omitted

## COOK'S DISTANCE

- No outliers detected by the cooks distance
- No data points outlying significantly in both y and x

## HATDIAG

- No datapoints outlying in the x direction significatntly

## We can see that few data points have high COVRATIO

- Covariance matrix will be influenced by following data points
- high covratio => improves the precision of estimation
- High leverage point can make COVRATIO large - it will improve the precision (unless point is an outlier in y-space)

In [306]:
```
1 influences.loc[influences["cov_ratio"]]
```

Out[306]:

| | dfb_Intercept | dfb_height | dfb_fat | dfb_diast | dfb_height_pow2 | dfb_weight_log | cooks_d | hat_diag | dffi |
|---|---|---|---|---|---|---|---|---|---|
| **6** | False | False | False | False | False | False | False | False | Fal |
| **8** | False | False | False | False | False | False | False | False | Fal |
| **9** | False | False | False | False | False | False | False | False | Fal |
| **12** | False | False | False | False | False | False | False | False | Fal |
| **27** | False | False | False | False | False | False | False | False | Fal |

## And one data point have high DIFFITS

- The deletion influence of observation on the predicted or fitted value

In [308]:
```
1 influences.loc[influences["dffits"]]
```

Out[308]:

| | dfb_Intercept | dfb_height | dfb_fat | dfb_diast | dfb_height_pow2 | dfb_weight_log | cooks_d | hat_diag | dffi |
|---|---|---|---|---|---|---|---|---|---|
| **45** | False | False | False | False | False | False | False | False | Tru |

In [ ]:
```
1
```