EMAT30007 Applied Statistics

Lab 8: Model Building

Nikolai Bode

This lab investigates model buiding for Linear Models (LMs) in statistics. In the first part, we will look a how to construct models with different types of predictors and in the second part, we will explore common pitfalls in statistical analysis with LMs.

There is no data associated with this lab. Instead, you will simulate your own data. The idea behind this is to develop your understanding of how certain features in data arise.

(1) Different types of predictors

In this part of the lab, we will work through different types of predictors that can be used in constructing LMs. For each of the predictors, your task is to:

- simulate data including the response and predictor(s).

- fit a LM with only quantitative predictors and perform model checking using residual plots to see what goes wrong when the predictors in your data are not quantitative.

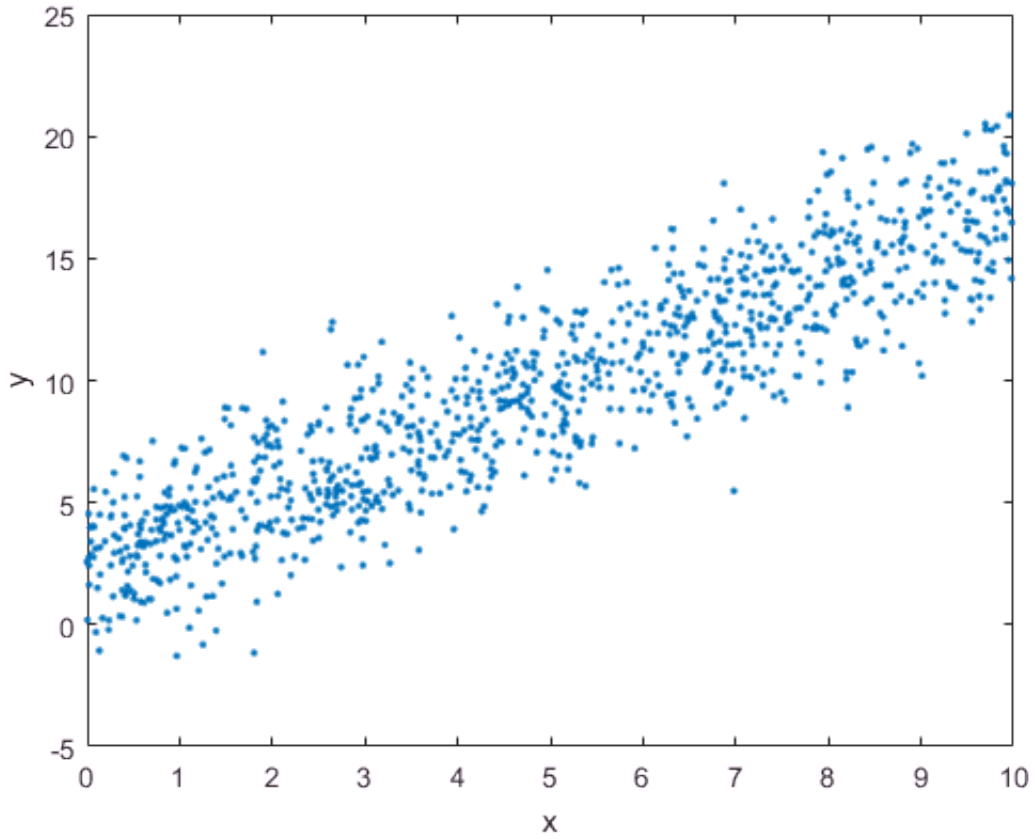
- fit an appropriate LM to this data to verify the data you constructed behaves as it should do.

- perform model checking using residual plots we have encountered before.

1.1. Quantitative predictor

This is the predictor type we have been looking at in lectures 6 and 7. Simulate data for one quantitative predictor. To do so, think about the structure of LMs, $Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$, where $\varepsilon_i \sim N(0, \sigma)$.

```
% One solution:

x = random('Uniform',0,10,[1000,1]); % predictor
beta0 = 2.3; % intercept
beta1 = 1.5; % effect of predictor
epsilon = random('Normal',0,2.1,[1000,1]); % error vector N(0,2.1)
y = beta0 + beta1.*x + epsilon;

% check the scatterplot of this data:
clf
plot(x,y,'.')
xlabel('x')
ylabel('y')
```




```
% fit a LM:
data = table(x,y,'VariableNames',{'predictor','response'});
% show first few rows of table
data(1:5,:)
```

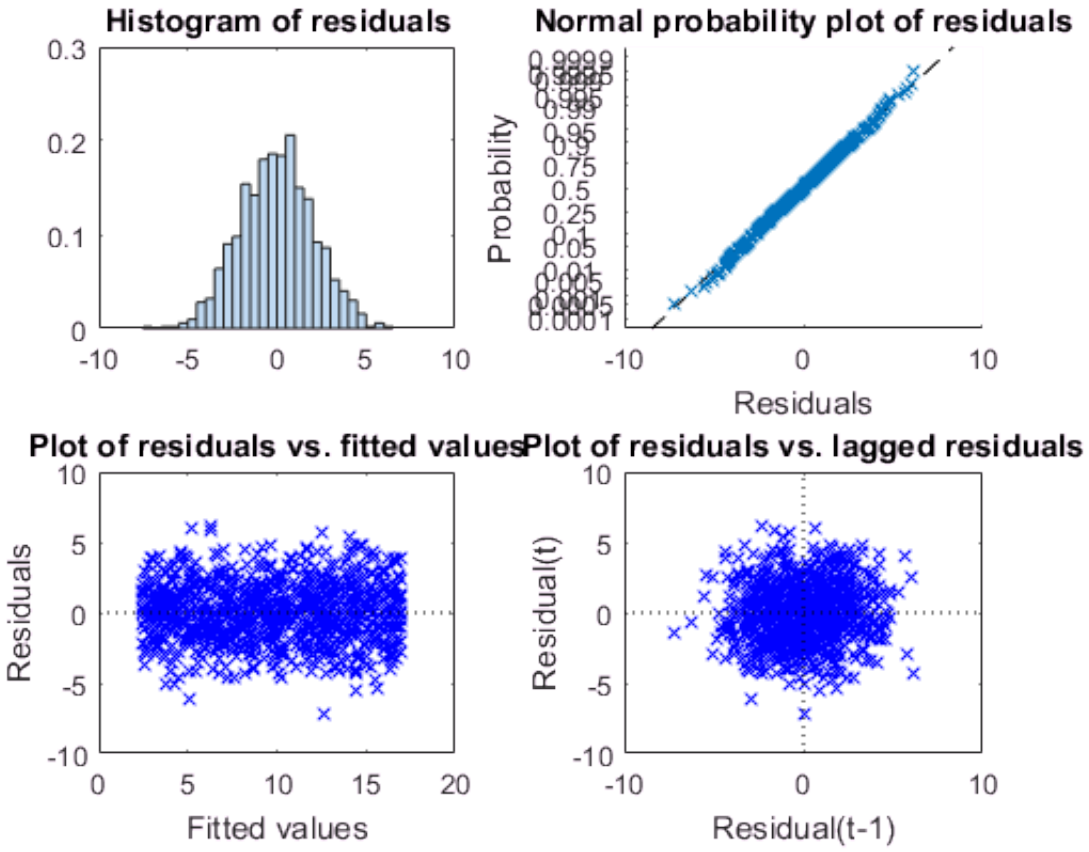
```
ans =
    predictor    response
    _________    ________

    6.1466       10.265
     7.038       11.503
    9.6233       15.213
    8.1117       14.762
    6.8212       12.436
```

```
m1 = fitlm(data, 'response~predictor')
```

```
m1 =

Linear regression model:
    response ~ 1 + predictor

Estimated Coefficients:
                   Estimate       SE       tStat       pValue
                   ________    ________    ______    __________

    (Intercept)    2.4074       0.12697    18.961    1.0362e-68
    predictor      1.4624      0.022328    65.495             0
```

```
Number of observations: 1000, Error degrees of freedom: 998
Root Mean Squared Error: 2.05
R-squared: 0.811,  Adjusted R-Squared 0.811
F-statistic vs. constant model: 4.29e+03, p-value = 0
```

```
% we approximately recover the parameter values we decided on above.

% residual plots:
% plot distribution of residuals (for outliers etc.)
clf
subplot(2,2,1)
plotResiduals(m1)
% Q-Q plot to check normality
subplot(2,2,2)
plotResiduals(m1,'probability')
% residuals versus fitted values
subplot(2,2,3)
plotResiduals(m1,'fitted')
% auto-correlation (via lagged residuals)
subplot(2,2,4)
plotResiduals(m1,'lagged')
```




```
%... these plots obviously look great, because we constructed our data to
%satisfy all LM assumptions.
```

1.2. Qualitative predictor

Construct data for a response with one quantitative and one qualitative predictor. You can simply extend the model from section 1.1 and you can choose how many levels you want the qualitative predictor to have. Some tips:

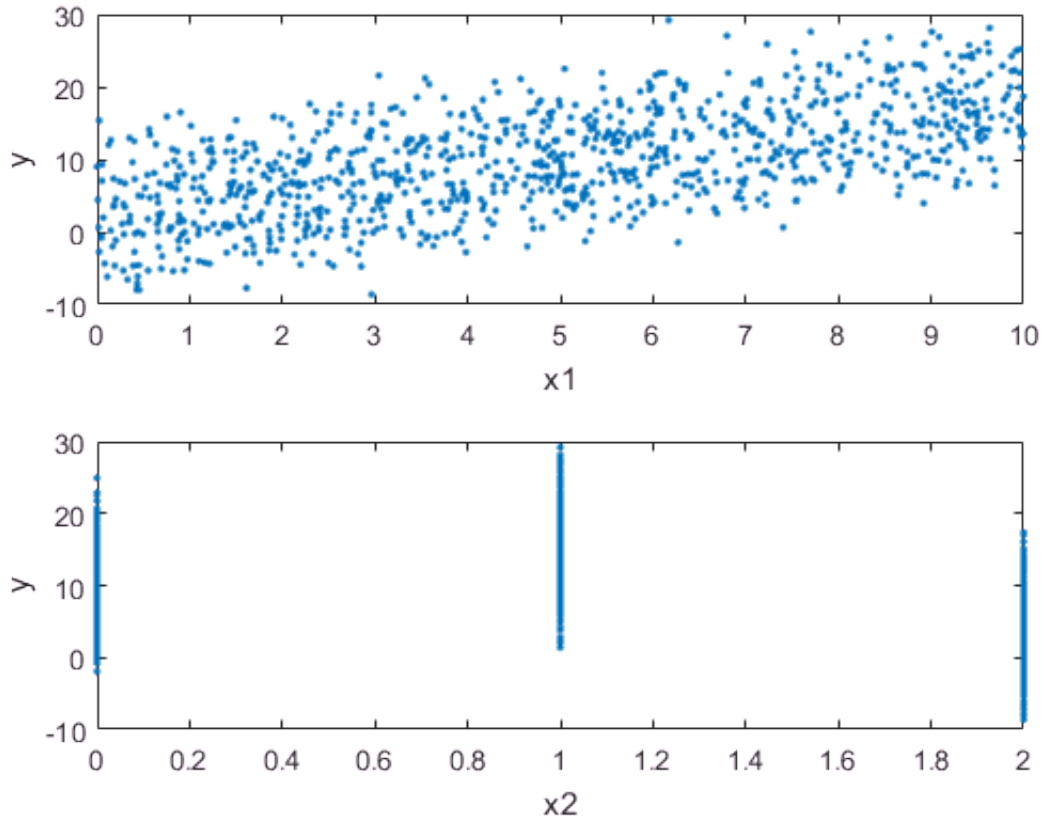
- the first level of the qualitative predictor is often absorbed into the intercept.

- parameters for qualitative predictor levels measure effects relative to the intercept.

- Use the structure of LMs with qualitative predictors when simulating your data.

- before fitting a LM with qualitative predictors in Matlab, you need to let it know that some variables are qualitative. If 'data' is a table with data and 'x' is one entry, use 'data.x = nominal(data.x)' in Matlab.

```
% One solution, we will use a qualitative predictor with 3 levels:

x1 = random('Uniform',0,10,[1000,1]); % quantitative predictor
x2 = zeros(1000,1); % this will be our qualitative predictor
x2(1:400) = 0; % level 0
x2(401:700) = 1; % level 1
x2(701:1000) = 2; % level 2
beta0 = 2.3; % intercept
beta1 = 1.5; % effect of predictor x1
beta2 = 6; % effect of level 1 of x2
beta3 = -5.5; % effect of level 2 of x2
epsilon = random('Normal',0,3.5,[1000,1]); % error vector N(0,3.5)
% create some dummy variables for simulating the data:
dum1 = zeros(1000,1);
dum2 = zeros(1000,1);
dum1(find(x2==1)) = 1;
dum2(find(x2==2)) = 1;
y = beta0 + beta1.*x1 + beta2.*dum1 + beta3.*dum2 + epsilon;

% look at the scatterplots of this data for the two predictors:
clf
subplot(2,1,1)
plot(x1,y,'.')
xlabel('x1')
ylabel('y')
subplot(2,1,2)
plot(x2,y,'.')
xlabel('x2')
ylabel('y')
```

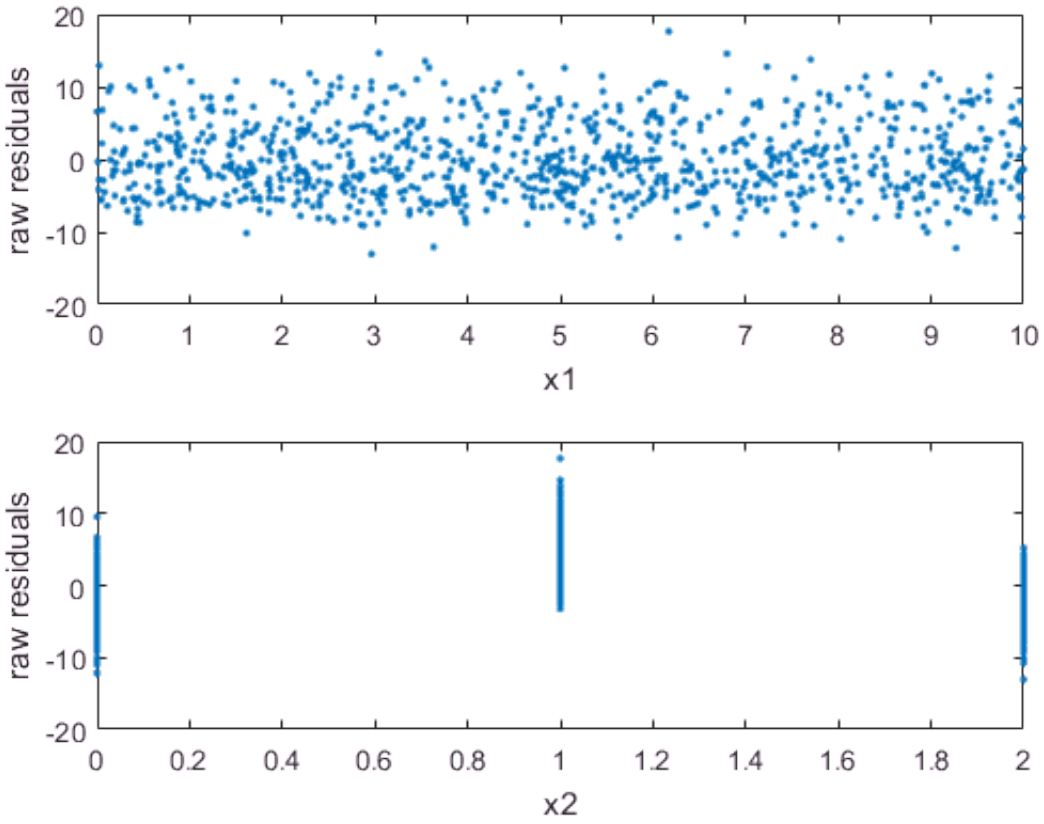



```
% fit a LM with two quantitative predictors to see how this is not
% appropriate:
data = table(x1,x2,y,'VariableNames',{'predictor1','predictor2','response'});
m1 = fitlm(data, 'response~predictor1+predictor2')
```

```
m1 =

Linear regression model:
    response ~ 1 + predictor1 + predictor2

Estimated Coefficients:
                   Estimate      SE        tStat       pValue
                   ________    _______    _______    ___________

    (Intercept)      4.762     0.37358     12.747     1.3824e-34
    predictor1      1.4807     0.05743     25.782    1.0098e-112
    predictor2     -2.3568     0.19778    -11.916     1.0625e-30


Number of observations: 1000, Error degrees of freedom: 997
Root Mean Squared Error: 5.19
R-squared: 0.452,  Adjusted R-Squared 0.451
F-statistic vs. constant model: 411, p-value = 7.58e-131
```

```
% compare the two residual plots of predictors x1 and x2 versus residuals:
clf
subplot(2,1,1)
plot(x1,m1.Residuals{:,1},'.')
xlabel('x1')
ylabel('raw residuals')
```

```
subplot(2,1,2)
plot(x2,m1.Residuals{:,1},'.')
xlabel('x2')
ylabel('raw residuals')
```

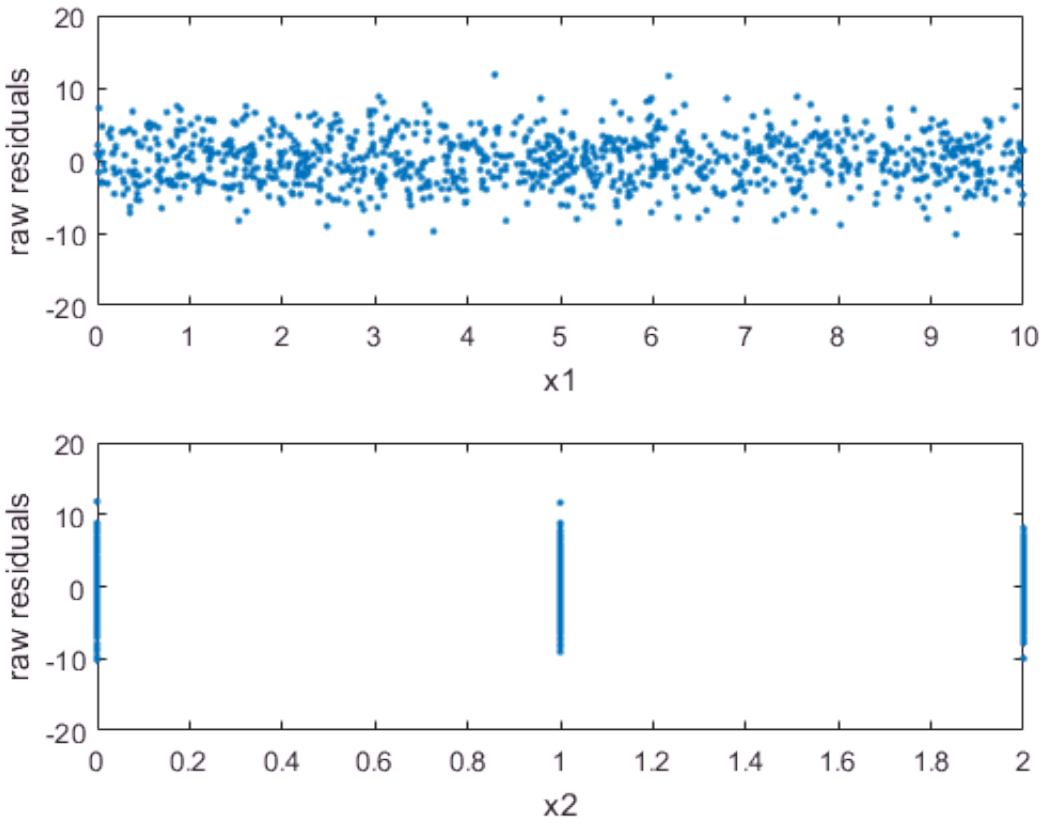



```
% ... there is no trend for x1, as the model explains the effect of x1, but
% the plot for x2 still has a trend for the different levels, so the effect
% of x2 is not captured by the model.

% tell Matlab that x2 is a qualitative predictor:
data.predictor2 = nominal(data.predictor2);
% now fit a LM with a qualitative predictor (notice how the formula is the same):
m2 = fitlm(data, 'response~predictor1+predictor2')
```

```
m2 =

Linear regression model:
    response ~ 1 + predictor1 + predictor2

Estimated Coefficients:
                    Estimate       SE        tStat       pValue
                    ________    ________    _______    ___________

    (Intercept)       2.2879      0.25589     8.9406     1.8434e-18
    predictor1        1.5304     0.037916     40.362    8.0368e-212
    predictor2_1      5.8158      0.26198       22.2     4.9943e-89
    predictor2_2     -5.4481      0.26179    -20.811     3.7201e-80


Number of observations: 1000, Error degrees of freedom: 996
Root Mean Squared Error: 3.43
R-squared: 0.762,  Adjusted R-Squared 0.761
```

```
F-statistic vs. constant model: 1.06e+03, p-value = 1.81e-309
```

```
% now we recover the constructed effects of the levels of x2 and the trend
% in the residuals plotted against x2 disappears:
clf
subplot(2,1,1)
plot(x1,m2.Residuals{:,1},'.')
xlabel('x1')
ylabel('raw residuals')
subplot(2,1,2)
plot(x2,m2.Residuals{:,1},'.')
xlabel('x2')
ylabel('raw residuals')
```

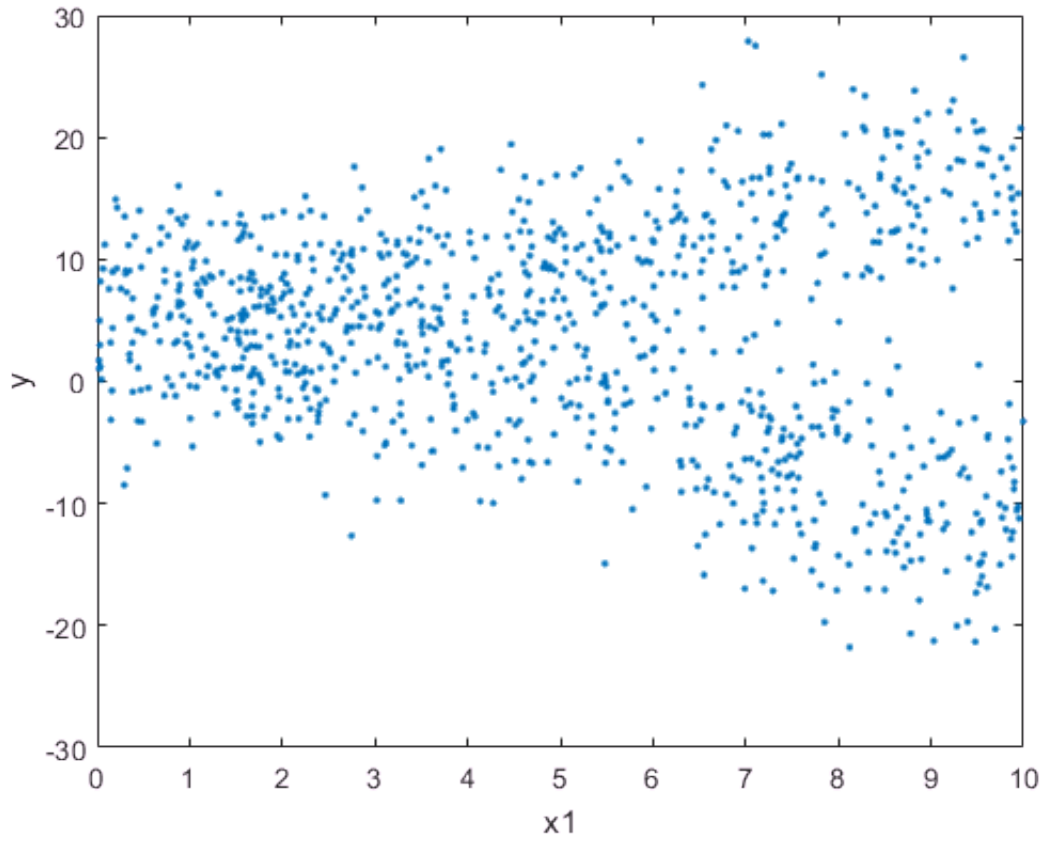


1.3. Interaction terms

Construct data with one quantitative and one qualitative predictor (2 levels). Include an interaction term between the two predictor so that the slope of the quantitative predictor changes for the levels of the qualitative predictor (cf slide in the lecture notes). Tip: in Matlab, in 'fitlm()', use '*' to include an interaction between two predictors.

```
% One solution:

x1 = random('Uniform',0,10,[1000,1]); % quantitative predictor
x2 = zeros(1000,1); % this will be our qualitative predictor
x2(1:500) = 0; % level 0
```

```
x2(501:1000) = 1; % level 1
beta0 = 2.3; % intercept
beta1 = 1.5; % baseline effect of predictor x1
beta2 = 6; % baseline effect of level 1 of x2
beta3 = -3.5; % interaction between x1 and level 1 of x2
epsilon = random('Normal',0,5,[1000,1]); % error vector N(0,5)
% construct response:
y = beta0 + beta1.*x1 + beta2.*x2 + beta3.*x1.*x2 + epsilon;

% look the scatterplot of this data for predictor x1:
clf
plot(x1,y,'.')
xlabel('x1')
ylabel('y')
```

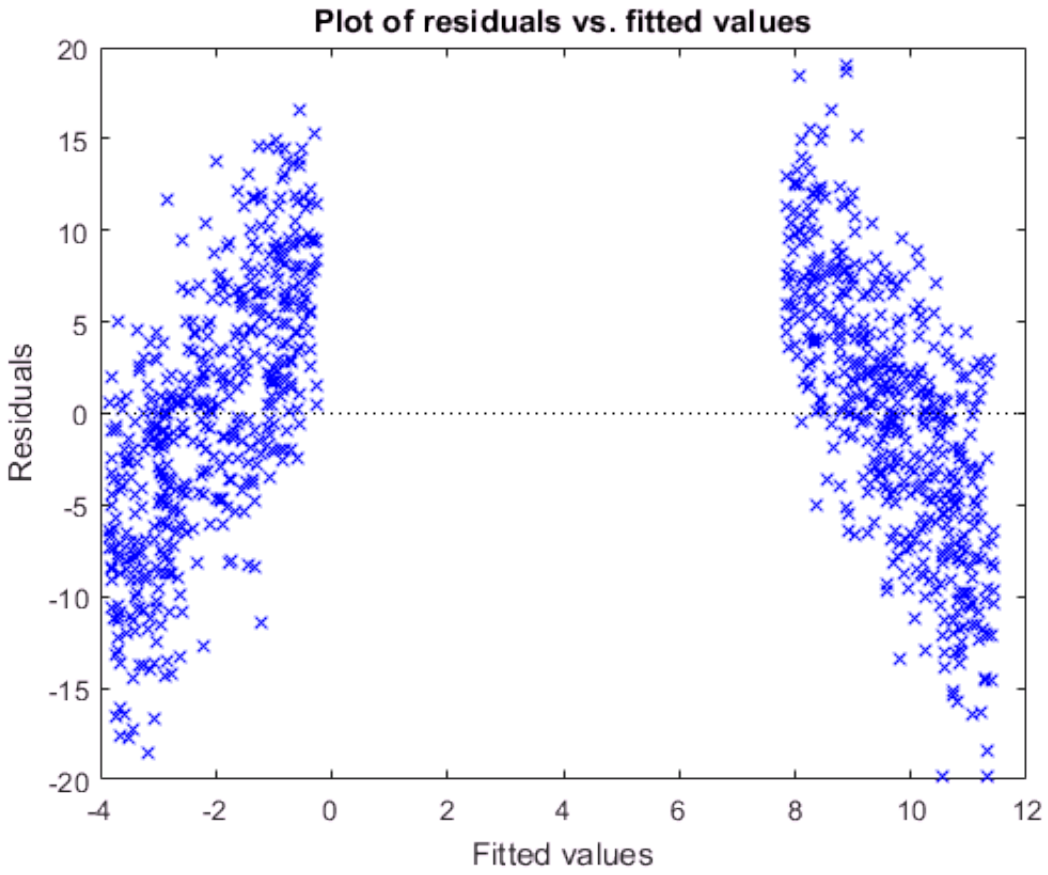



```
% fit a LM with two quantitative predictors to see how this is not
% appropriate:
data = table(x1,x2,y,'VariableNames',{'predictor1','predictor2','response'});
m1 = fitlm(data, 'response~predictor1+predictor2')
```

```
m1 =

Linear regression model:
    response ~ 1 + predictor1 + predictor2

Estimated Coefficients:
                   Estimate       SE        tStat       pValue
                   ________    ________    _______    __________

    (Intercept)      11.442     0.49689     23.027     2.0227e-94
```

```
    predictor1     -0.36154    0.078133    -4.6272     4.1951e-06
    predictor2      -11.678     0.45074    -25.908    1.4354e-113

Number of observations: 1000, Error degrees of freedom: 997
Root Mean Squared Error: 7.12
R-squared: 0.414,  Adjusted R-Squared 0.413
F-statistic vs. constant model: 352, p-value = 2.36e-116
```

```
% we find significant effects, but...
% plotting residuals versus fitted values...
clf
plotResiduals(m1,'fitted')
```

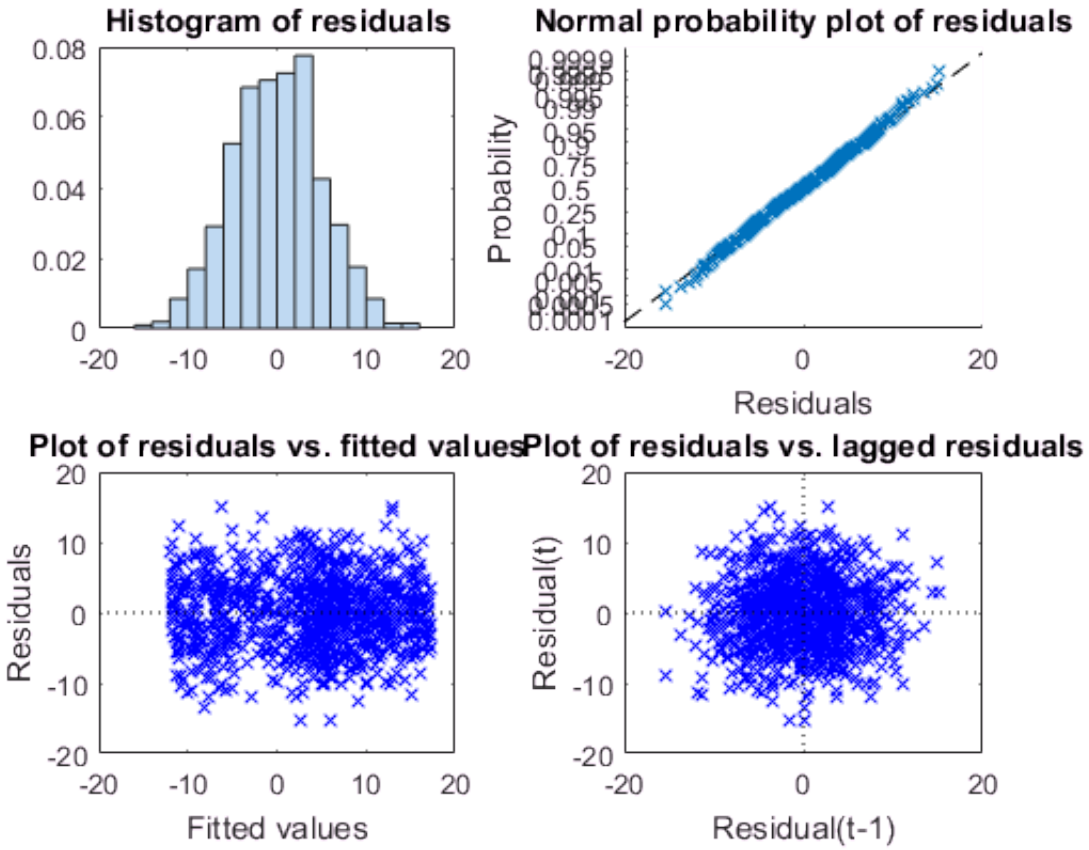



```
% ... shows a clear separation of residuals into two clusters with each
% cluster showing a trend in the mean residual values.

% fit the correct LM with qualitative predictor x2 and an interaction term:
data.predictor2 = nominal(data.predictor2);
m2 = fitlm(data, 'response~predictor1*predictor2')
```

```
m2 =

Linear regression model:
    response ~ 1 + predictor1*predictor2

Estimated Coefficients:
                            Estimate       SE        tStat       pValue
                            ________    ________    _______    __________

    (Intercept)               2.4275      0.45107     5.3816     9.2059e-08
    predictor1                1.4849     0.080125     18.533     4.1542e-66
```

```
predictor2_1                5.8268      0.63646      9.155     3.0067e-19
predictor1:predictor2_1    -3.5044      0.11038    -31.748     2.2338e-153

Number of observations: 1000, Error degrees of freedom: 996
Root Mean Squared Error: 5.02
R-squared: 0.709,  Adjusted R-Squared 0.708
F-statistic vs. constant model: 807, p-value = 4.11e-266
```

```
% we approximately recover the parameters we used to construct the data and
% the residual plots now look fine:
% plot distribution of residuals (for outliers etc.)
clf
subplot(2,2,1)
plotResiduals(m2)
% Q-Q plot to check normality
subplot(2,2,2)
plotResiduals(m2,'probability')
% residuals versus fitted values
subplot(2,2,3)
plotResiduals(m2,'fitted')
% auto-correlation (via lagged residuals)
subplot(2,2,4)
plotResiduals(m2,'lagged')
```

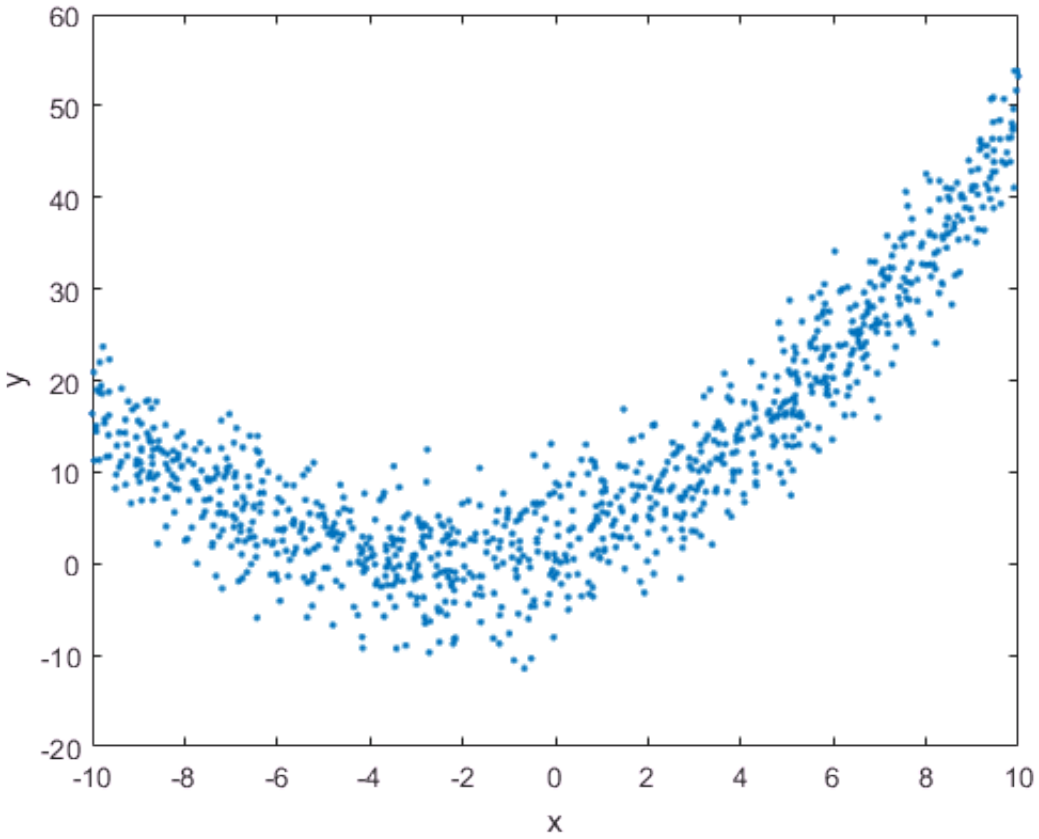



```
```

1.4. Polynomials of predictors

Now let's construct data with a response that is a polynomial of one quantitative predictor, e.g. a second-order polynomial: $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i$.

```
% One solution:

x = random('Uniform',-10,10,[1000,1]); % predictor
beta0 = 2.3; % intercept
beta1 = 1.5; % effect of predictor
beta2 = 0.3; % effect of square of predictor
epsilon = random('Normal',0,4.3,[1000,1]); % error vector N(0,4.3)
y = beta0 + beta1.*x + beta2.*x.*x + epsilon;

% check the scatterplot of this data:
clf
plot(x,y,'.')
xlabel('x')
ylabel('y')
```

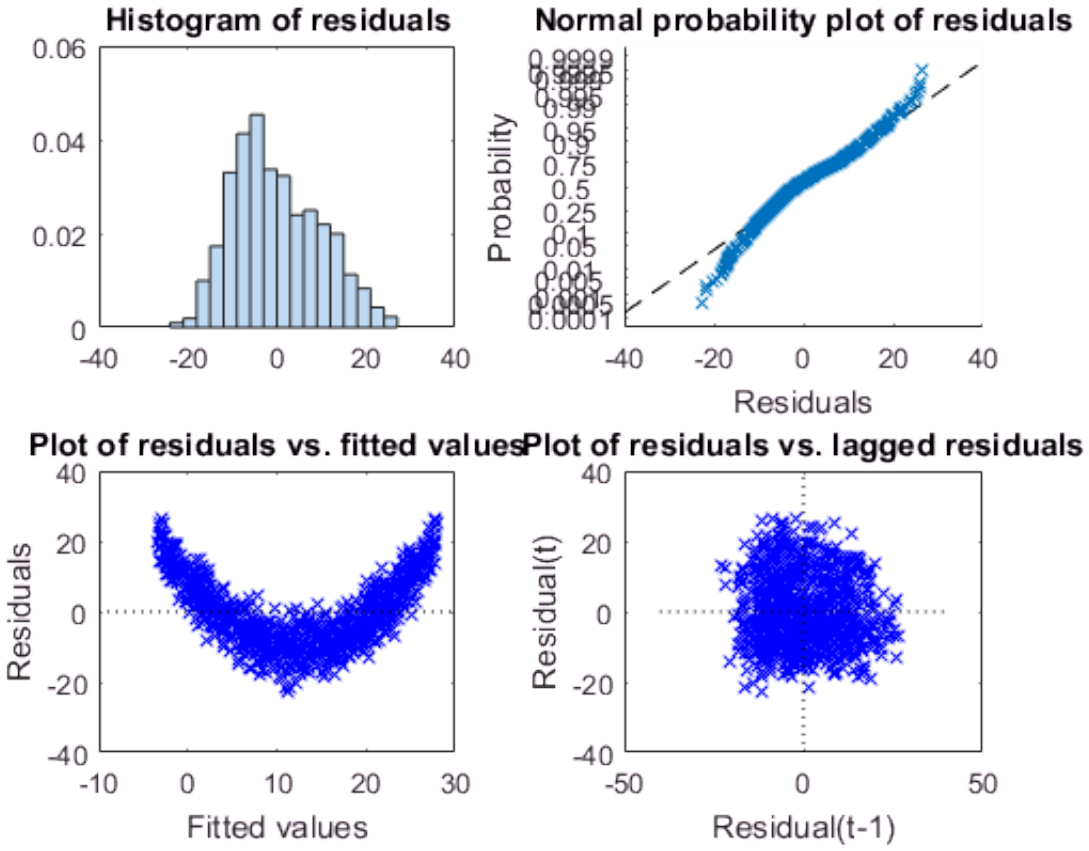



```
% fit a LM with only one quantitative predictor:
data = table(x,y,'VariableNames',{'predictor','response'});
m1 = fitlm(data, 'response~predictor')
```

```
m1 =

Linear regression model:
    response ~ 1 + predictor

Estimated Coefficients:
                   Estimate       SE       tStat       pValue
```

```
                   ________    ________    ______    ___________

    (Intercept)    12.255       0.31029    39.496    3.4502e-206
    predictor      1.5607      0.053807    29.005    1.1965e-134


Number of observations: 1000, Error degrees of freedom: 998
Root Mean Squared Error: 9.81
R-squared: 0.457,  Adjusted R-Squared 0.457
F-statistic vs. constant model: 841, p-value = 1.2e-134
```

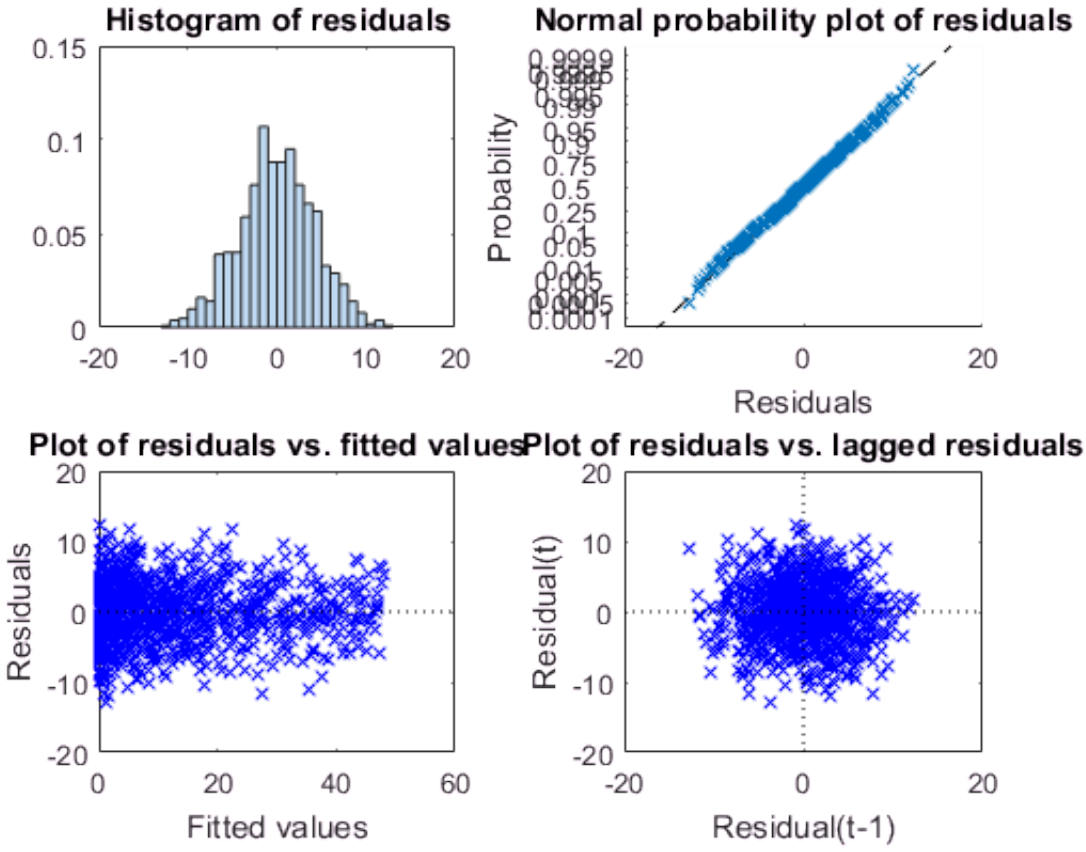
```
% the residual plots for this model fit are all over the place:
% plot distribution of residuals (for outliers etc.)
clf
subplot(2,2,1)
plotResiduals(m1)
% Q-Q plot to check normality
subplot(2,2,2)
plotResiduals(m1,'probability')
% residuals versus fitted values
subplot(2,2,3)
plotResiduals(m1,'fitted')
% auto-correlation (via lagged residuals)
subplot(2,2,4)
plotResiduals(m1,'lagged')
```




```
%... so we need to fit a better model to the data.

% fit a LM with the correct polynomial of the predictor:
data2 = table(x,x.^2,y,'VariableNames',{'x','xsquared','response'});
m2 = fitlm(data2, 'response~x+xsquared')
```

```
m2 =

Linear regression model:
    response ~ 1 + x + xsquared

Estimated Coefficients:
                   Estimate       SE        tStat       pValue
                   ________    _________    ______    __________

    (Intercept)     2.1918       0.20374    10.758    1.2988e-25
    x               1.5382      0.023312    65.983             0
    xsquared       0.30272     0.0046054    65.732             0


Number of observations: 1000, Error degrees of freedom: 997
Root Mean Squared Error: 4.25
R-squared: 0.898,  Adjusted R-Squared 0.898
F-statistic vs. constant model: 4.4e+03, p-value = 0
```

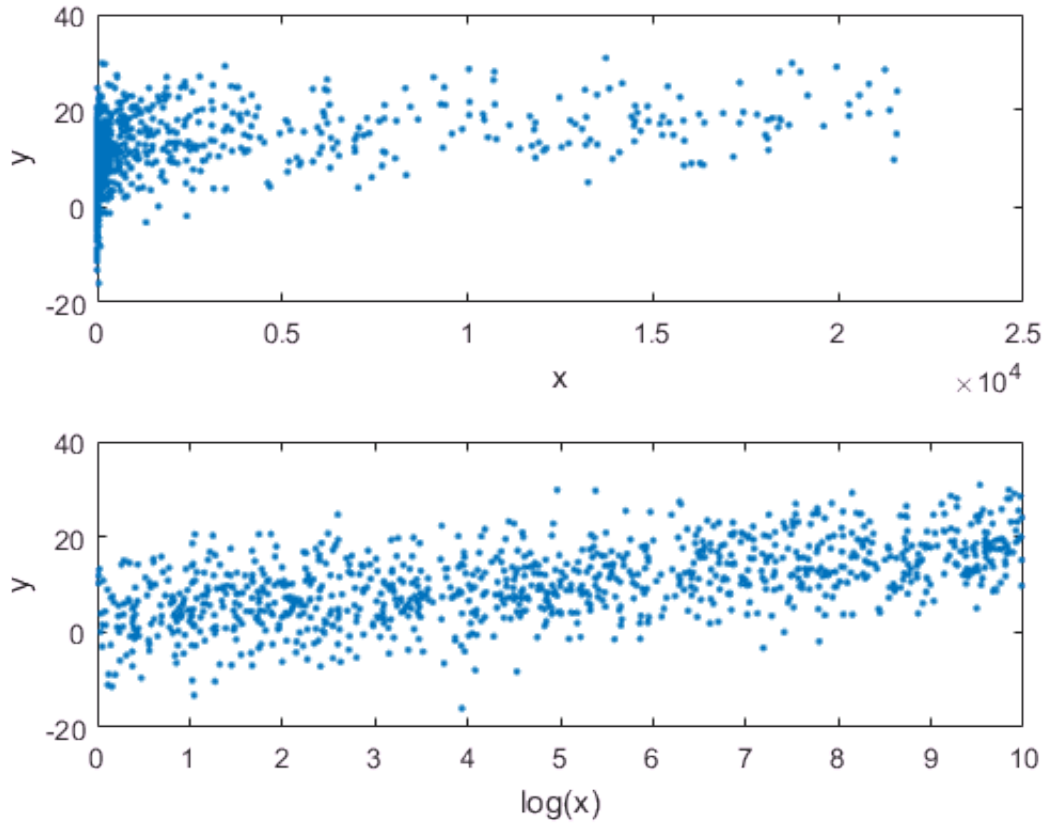
```
% we approximately recover the correct parameter values and the residual
% plots look as they should do:
clf
subplot(2,2,1)
plotResiduals(m2)
% Q-Q plot to check normality
subplot(2,2,2)
plotResiduals(m2,'probability')
% residuals versus fitted values
subplot(2,2,3)
plotResiduals(m2,'fitted')
% auto-correlation (via lagged residuals)
subplot(2,2,4)
plotResiduals(m2,'lagged')
```



1.5. Data transformations

Consider a response that is a function of one quantitative predictor. However, instead of being a linear function of the predictor, the response is a linear function of the logarithm of the predictor.

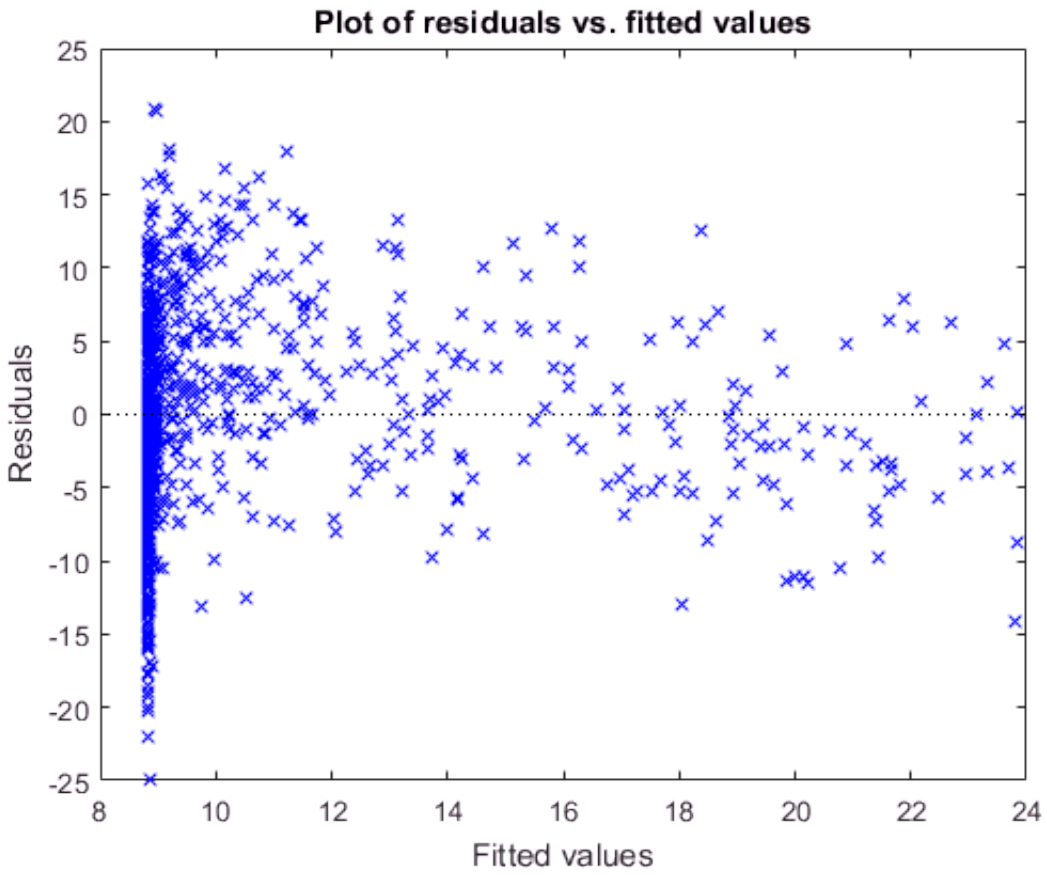
```
% Example:

x0 = random('Uniform',0.001,10,[1000,1]);
x = exp(x0); % predictor
beta0 = 3.1; % intercept
beta1 = 1.5; % effect of log(predictor)
epsilon = random('Normal',0,6.3,[1000,1]); % error vector N(0,6.3)
y = beta0 + beta1.*log(x) + epsilon;

% check the scatterplot of this data:
clf
subplot(2,1,1)
plot(x,y,'.')
xlabel('x')
ylabel('y')
subplot(2,1,2)
plot(log(x),y,'.')
xlabel('log(x)')
ylabel('y')
```




```
% fit a LM with only one quantitative predictor:
data = table(x,y,'VariableNames',{'predictor','response'});
m1 = fitlm(data, 'response~predictor')
```

```
m1 =

Linear regression model:
    response ~ 1 + predictor

Estimated Coefficients:
                    Estimate         SE        tStat       pValue
                   __________    __________    ______    ___________

    (Intercept)         8.833       0.24497    36.057    5.8498e-183
    predictor      0.00069539    4.7402e-05     14.67     2.8705e-44


Number of observations: 1000, Error degrees of freedom: 998
Root Mean Squared Error: 6.96
R-squared: 0.177,  Adjusted R-Squared 0.177
F-statistic vs. constant model: 215, p-value = 2.87e-44
```

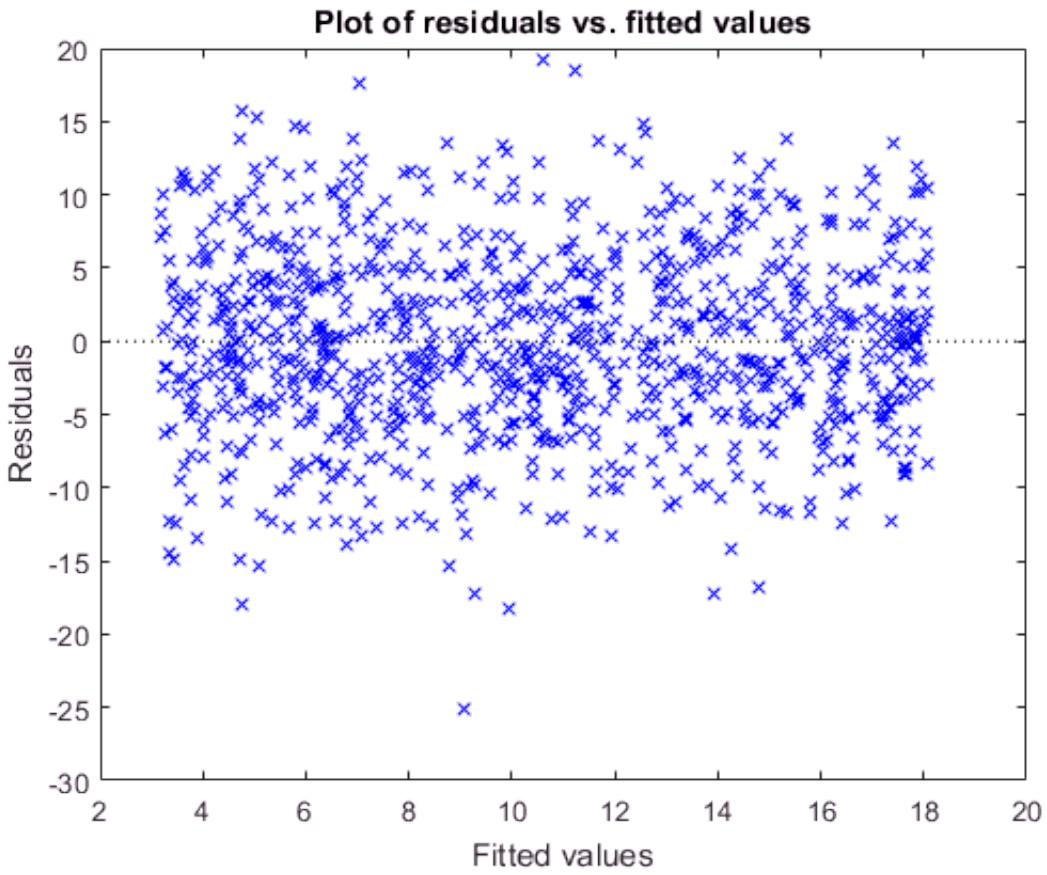
```
% according to this model the predictor has an effect, but plotting the
% fitted values versus the residuals is not entirely satisfying:
clf
plotResiduals(m1,'fitted')
```




```
%... so we need to fit a better model to the data.

% fit a LM with log(x) as the predictor:
data2 = table(log(x),y,'VariableNames',{'logx','response'});
m2 = fitlm(data2, 'response~logx')
```

```
m2 =

Linear regression model:
    response ~ 1 + logx

Estimated Coefficients:
                   Estimate       SE       tStat       pValue
                   ________    ________    ______    __________

    (Intercept)    3.1724       0.38419    8.2575    4.6824e-16
    logx           1.4938      0.067822    22.025    6.2989e-88


Number of observations: 1000, Error degrees of freedom: 998
Root Mean Squared Error: 6.3
R-squared: 0.327,  Adjusted R-Squared 0.326
F-statistic vs. constant model: 485, p-value = 6.3e-88
```

```
% we approximately recover the correcr parameter values and the residual
% plots look as they should do:
clf
plotResiduals(m2,'fitted')
```



(2) Common pitfalls

In the second part of the lab, we will look at a few common pitfalls in statistical analysis using LMs. In the following, construct data sets that lead to the specified pitfalls. Similary to the first part, your task is to:

- simulate data including the response and predictor(s).

- fit a LM to see what goes wrong for the pitfall you have implemented.

- perform model checking using residual plots we have encountered before.

2.1. Multicollinearity

We have already seen an example for multicollinearity in lab 7. So we will not repeat this here.

2.2. Violated model assumptions

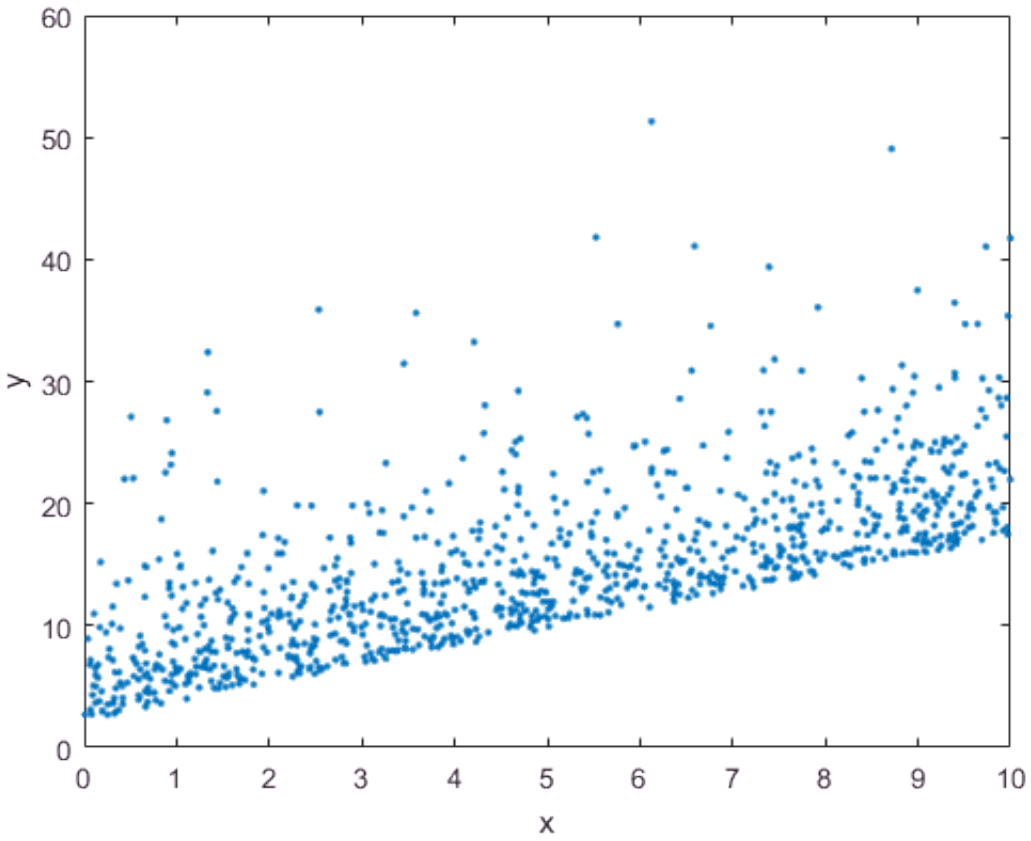
Perhaps the most common issue with LM analyses is that the assumptions of the model are not satisfied. Typically, this relates to the error distribution, which we assume to be normal, with constant variance and comprised of independent samples in LM. The following examples show what the residual plots of models fitted to data for which certain assumtions do not hold look like. In general, when

model assumptions are violated, we cannot use the standard framework of hypothesis tests and model selection.

2.2.1. Non-normal error distribution

Construct data for which the errors do not follow a norma distribution (e.g. exponential or gamma distributed).

```
x = random('Uniform',0,10,[1000,1]); % predictor
beta0 = 2.3; % intercept
beta1 = 1.5; % effect of predictor
epsilon = random('Exponential',5,[1000,1]); % error vector
y = beta0 + beta1.*x + epsilon;

% check the scatterplot of this data:
clf
plot(x,y,'.')
xlabel('x')
ylabel('y')
```

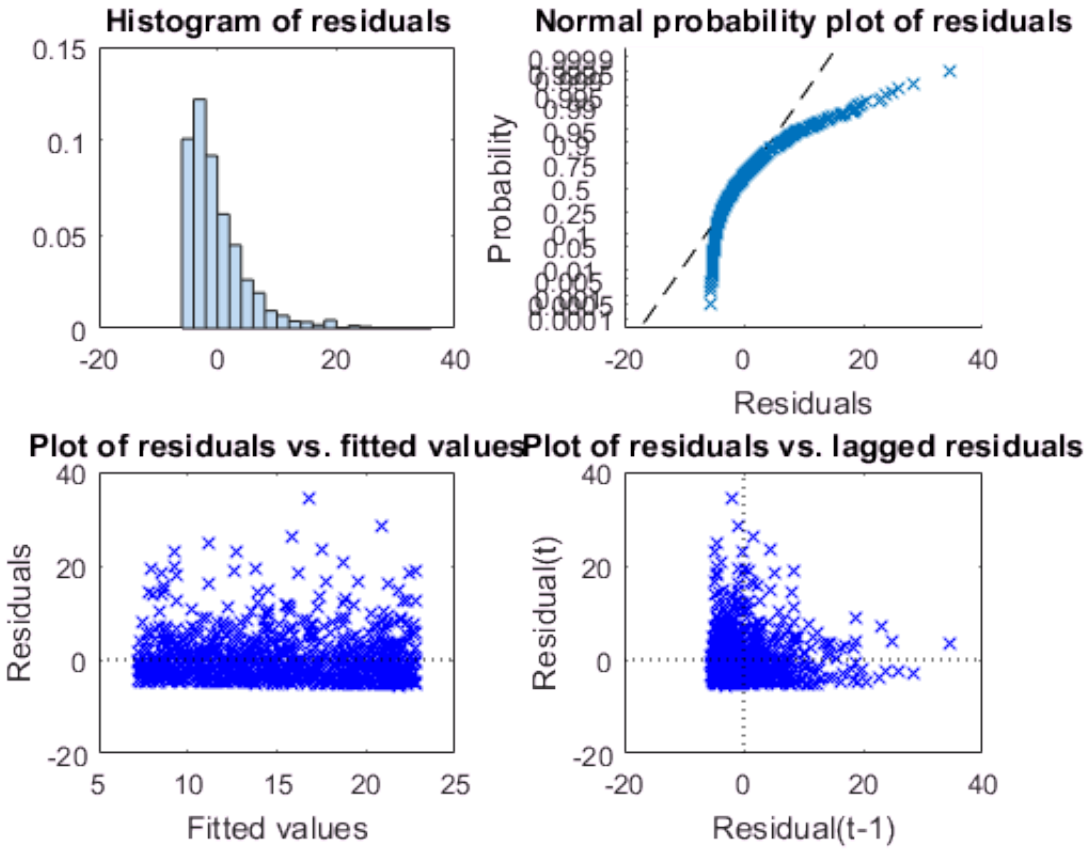



```
% fit a LM:
data = table(x,y,'VariableNames',{'predictor','response'});
m1 = fitlm(data, 'response~predictor')
```

```
m1 =

Linear regression model:
```

```
    response ~ 1 + predictor

Estimated Coefficients:
                   Estimate       SE       tStat       pValue
                   ________    ________    ______    ___________

    (Intercept)    7.1606       0.32021    22.362     4.1178e-90
    predictor      1.5674      0.055505    28.238    2.0852e-129


Number of observations: 1000, Error degrees of freedom: 998
Root Mean Squared Error: 5.15
R-squared: 0.444,  Adjusted R-Squared 0.444
F-statistic vs. constant model: 797, p-value = 2.09e-129
```

```
% residual plots do not suggest the error distribution is normal:

% plot distribution of residuals (for outliers etc.)
clf
subplot(2,2,1)
plotResiduals(m1)
% Q-Q plot to check normality
subplot(2,2,2)
plotResiduals(m1,'probability')
% residuals versus fitted values
subplot(2,2,3)
plotResiduals(m1,'fitted')
% auto-correlation (via lagged residuals)
subplot(2,2,4)
plotResiduals(m1,'lagged')
```

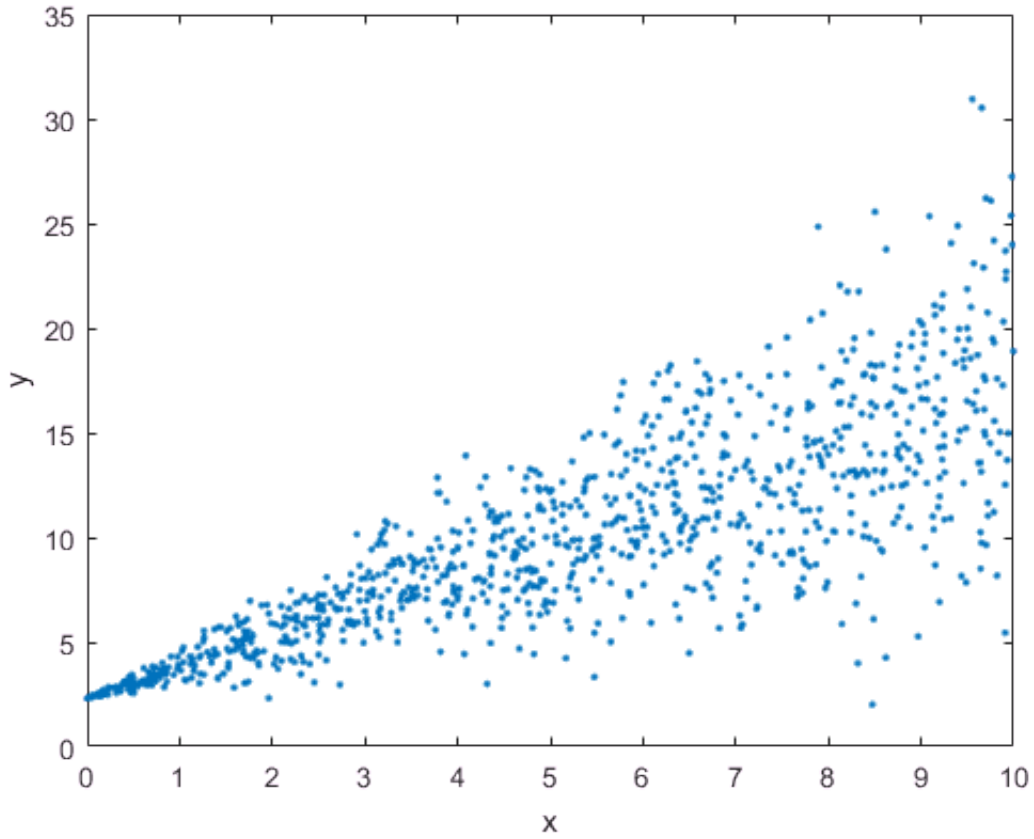


2.2.2. Non-constant variance of predictors

Construct data for which the errors follow a normal distribution, but the

variance of this distribution is not constant (heteroscedasticity).

```
x = random('Uniform',0,10,[1000,1]); % predictor
beta0 = 2.3; % intercept
beta1 = 1.5; % effect of predictor
epsilon = random('Normal',0,0.5*x,[1000,1]); % error vector
y = beta0 + beta1.*x + epsilon;

% check the scatterplot of this data:
clf
plot(x,y,'.')
xlabel('x')
ylabel('y')
```

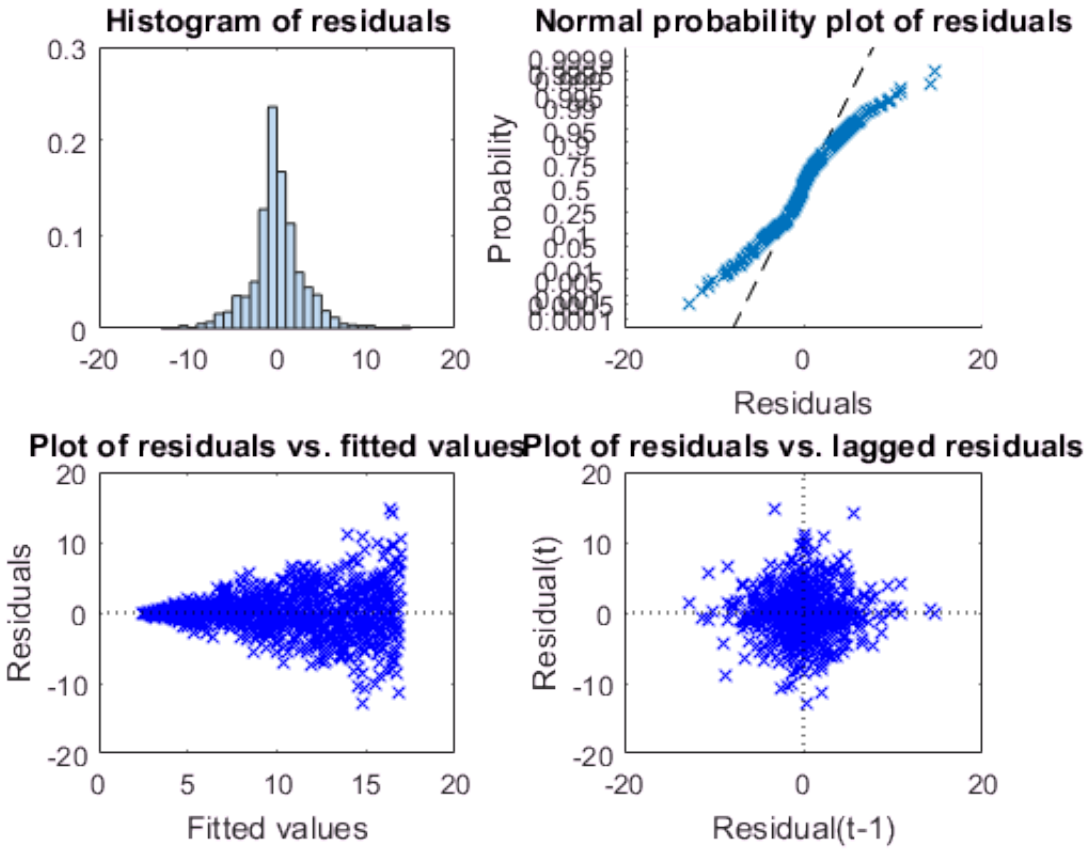



```
% fit a LM:
data = table(x,y,'VariableNames',{'predictor','response'});
m1 = fitlm(data, 'response~predictor')
```

```
m1 =

Linear regression model:
```

```
    response ~ 1 + predictor

Estimated Coefficients:
                   Estimate       SE       tStat       pValue
                   ________    ________    ______    ___________

    (Intercept)    2.4507       0.19202    12.763     1.1558e-34
    predictor      1.4512      0.032701    44.378    2.1787e-238


Number of observations: 1000, Error degrees of freedom: 998
Root Mean Squared Error: 2.98
R-squared: 0.664,  Adjusted R-Squared 0.663
F-statistic vs. constant model: 1.97e+03, p-value = 2.18e-238
```

```
% In the plot of fitted values versus residuals, it is clear to see how the
% variance of the residuals 'fans' out. This is a clear indication for
% heteroscedasticity.

% plot distribution of residuals (for outliers etc.)
clf
subplot(2,2,1)
plotResiduals(m1)
% Q-Q plot to check normality
subplot(2,2,2)
plotResiduals(m1,'probability')
% residuals versus fitted values
subplot(2,2,3)
plotResiduals(m1,'fitted')
% auto-correlation (via lagged residuals)
subplot(2,2,4)
plotResiduals(m1,'lagged')
```



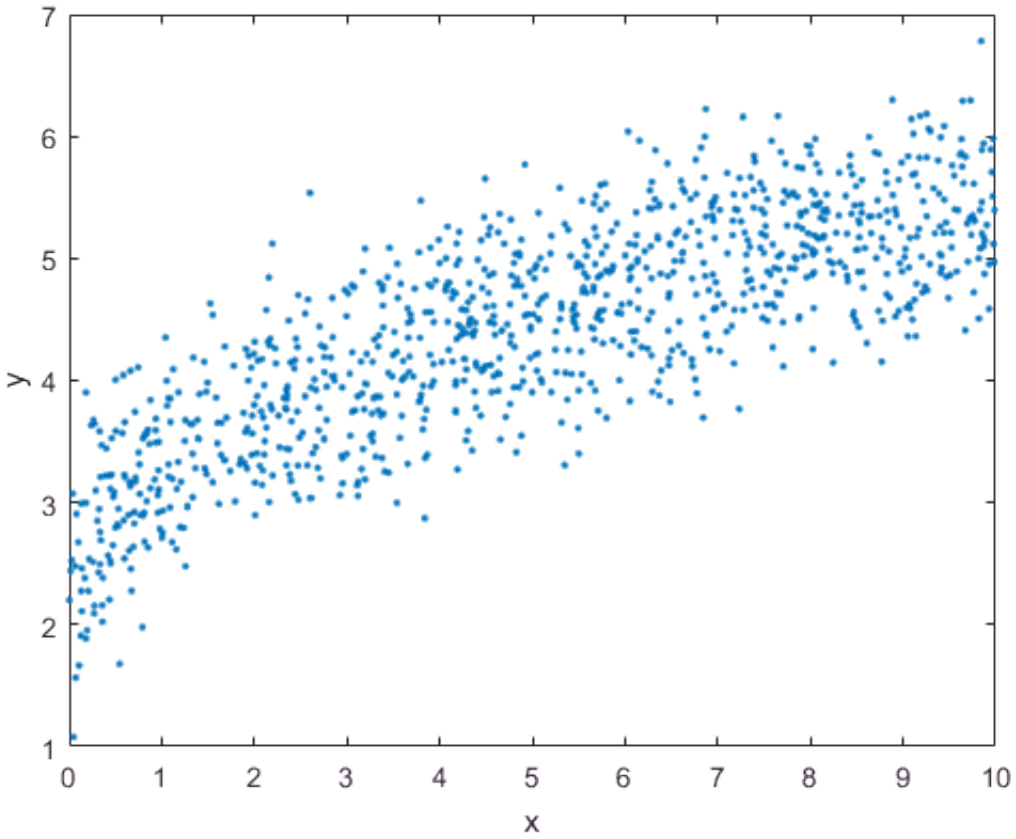
2.2.3. Data points not independent

We won't go through this example here. Data points that are not independent arise commonly in time-series data (data points close in time are strongly correlated).

2.3. Extrapolating beyond model scope

As discussed in lectures, we need to be careful when interpreting model fits beyond the range of the data the model was fitted to. A nice way to illustrate this issue, is to construct a response that is a non-linear function of a predictor which can be approximated linearly for part of the range of the predictor (e.g. consider $y = \sqrt{x}$).

```
x = random('Uniform',0,10,[1000,1]); % predictor
beta0 = 2.3; % intercept
beta1 = 1; % effect of predictor
epsilon = random('Normal',0,0.5,[1000,1]); % error vector
y = beta0 + beta1.*sqrt(x) + epsilon;

% check the scatterplot of this data:
clf
plot(x,y,'.')
xlabel('x')
ylabel('y')
```

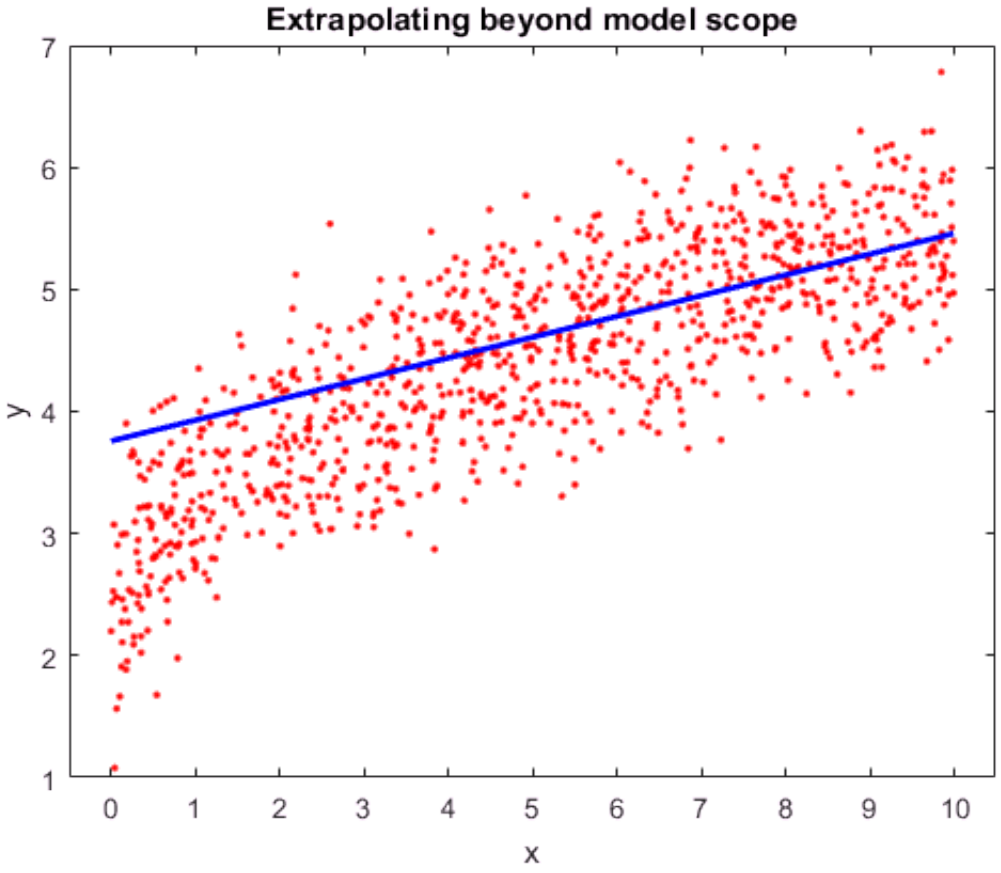



```
% fit LM to only part of the data:
xtmp = x(find(x>=5));
ytmp = y(find(x>=5));
data = table(xtmp,ytmp,'VariableNames',{'predictor','response'});
m1 = fitlm(data, 'response~predictor')
```

```
m1 =

Linear regression model:
    response ~ 1 + predictor

Estimated Coefficients:
                    Estimate       SE       tStat       pValue
                    ________    ________    ______    __________

    (Intercept)      3.7574      0.11974     31.38    3.6822e-120
    predictor       0.17023     0.015758    10.803      1.376e-24


Number of observations: 501, Error degrees of freedom: 499
Root Mean Squared Error: 0.506
R-squared: 0.19,  Adjusted R-Squared 0.188
F-statistic vs. constant model: 117, p-value = 1.38e-24
```

```
% plot the fit of this LM to the entire data set:
w = linspace(min(x),max(x));
figure()
gscatter(x,y)
line(w,feval(m1,w),'Color','b','LineWidth',2)
title('Extrapolating beyond model scope');
```

```
xlabel('x');
ylabel('y');
```




```
%... it's clear to see that the model fit performs poorly for values of x
%close to zero.
```

2.4. Overfitting

We won't go through an example here. The extreme case of overfitting occurs when we fit a model with as many parameters as there are data points.