ORIE 3120: Practical Tools for OR, DS, and ML

Linear Regression

Professor Udell

Operations Research and Information Engineering
Cornell

April 9, 2020

Announcements

- submit recitation by 4:30pm ET Friday
- homework due 2:30pm ET Wednesday
- project milestone 1 due Sunday 4/18/2020 at noon

Outline

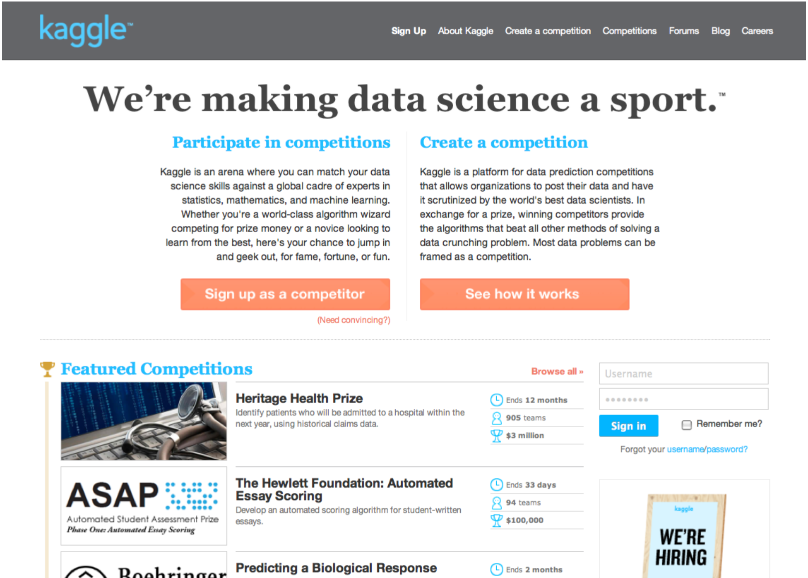
Part 1: Linear Regression - Introduction

Regression can

- investigate how variables are related,
- predict future values of variables, and
- show how output variables will change if we change input variables.

. . . the last step can be useful for control!

Regression examples (from Kaggle)



Regression for insurance claim predictions

Claim Prediction Challenge (Allstate)

Prize pool	Teams	Completed
$10,000	107	5 months ago

Many factors contribute to the frequency and severity of car accidents including how, where and under what conditions people drive, as well as what they are driving.

Bodily Injury Liability Insurance covers other people's bodily injury or death for which the insured is responsible. **The goal of this competition is to predict Bodily Injury Liability Insurance claim payments based on the characteristics of the insured's vehicle.**

Regression for finance

Benchmark Bond Trade Price Challenge

Prize pool	Teams	Ends
$17,500	168	33 days

The Benchmark Bond Trade Price Challenge is a competition to predict the next price that a US corporate bond might trade at. Contestants are given information on the bond including current coupon, time to maturity and a reference price computed by **Benchmark Solutions**. Details of the previous 10 trades are also provided.

Benchmark Solutions

Regression for social network analysis

Wikipedia's Participation Challenge

Prize pool	Teams	Completed
$10,000	96	6 months ago

WIKIPEDIA
The Free Encyclopedia

Wikipedia is the encyclopedia that can be edited by everybody but a majority of our editors make few edits (less than 50). We want to grow our editor community in terms of diversity and size so we need to understand why editors stop editing.

Between 2005 and 2007, newbies started having real trouble successfully joining the Wikimedia community. Before 2005 in the English Wikipedia, nearly 40% of new editors would still be active a year after their first edit. After 2007, only about 12-15% of new editors were still active a year after their first edit. Post-2007, lots of people were still trying to become Wikipedia editors. What had changed, though, is that they were increasingly failing to integrate into the Wikipedia community, and failing increasingly quickly. The Wikimedia community had become too hard to penetrate. For more information about these trends, please have a look at http://strategy.wikimedia.org/wiki/March_2011_Update

We want a quantitative answer to what factors predict future editing behavior, in order to ensure that the content of Wikipedia continues to grow both quantitively and qualitatively.

Regression setup

we want to predict output given inputs

- p input variables $X_1, \ldots, X_p \in \mathbf{R}$
  - also called "predictors", "independent variables", "covariates"
- output variable $Y \in \mathbf{R}$
  - also called "outcome", "response", "dependent variable", "label", "target" ...

In the Allstate example:

- Y is the cost of an insurance claim.
- $X_1, \ldots, X_p$ are the properties of the insured and his/her vehicle, e.g., credit score, age of the vehicle, ...

Linear models

linear model assumption: for some **parameters** $\beta_0, \ldots, \beta_p \in \mathbf{R}$,

$$E(Y|X_1, \ldots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

hence

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

where ϵ is variation of Y about $E(Y|X_1, \ldots, X_p)$

- β_0 is the **intercept**
- $\beta_1, \ldots, \beta_p$ are called **regression coefficients**
  - also called "slopes" or "partial derivatives":

$$\beta_j = \frac{\partial}{\partial X_j} E(Y|X_1, \ldots, X_p)$$

- ϵ is the unpredictable variation in Y
  - also called the "noise", "error", or "residual variation"

Outline

Data

observe $(Y_i, X_{i,1}, \dots, X_{i,p})$, for $i = 1, \dots, n$

- n observations total
- i = index of "observation" = "subject" = "row in data spreadsheet"
- so the linear regression model can be rewritten as

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \cdots + \beta_p X_{i,p} + \epsilon_i$$

- notice that $\beta_0, \beta_1, \dots, \beta_p$ do not depend on i
- the columns of the data spreadsheet are $Y_i, X_{i,1}, \dots, X_{i,p}$

i	Y_i	$X_{i,1}$	$X_{i,2}$	$\dots$	$X_{i,p}$
1	2.3	1.1	6.2	$\dots$	5.9
2	12.7	2.4	5.4	$\dots$	9.6
3	6.3	0.9	6.9	$\dots$	1.5

Least squares estimation

What if we don't know the output Y_i for some subject i?
Predict it!

- Given estimates $\hat{\beta}_0, \ldots, \hat{\beta}_p$ for the regression coefficients, predict Y_i as

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{i,1} + \hat{\beta}_2 X_{i,2} + \cdots + \hat{\beta}_p X_{i,p}$$

- $\hat{Y}_i$ is an estimate of $E(Y_i | X_{i,1}, \ldots, X_{i,p}) = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \cdots + \beta_p X_{i,p}$
- $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ find the "best" predictor by minimizing

$$\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Least squares estimation

What if we don't know the output Y_i for some subject i? Predict it!

- Given estimates $\hat{\beta}_0, \ldots, \hat{\beta}_p$ for the regression coefficients, predict Y_i as

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{i,1} + \hat{\beta}_2 X_{i,2} + \cdots + \hat{\beta}_p X_{i,p}$$

- $\hat{Y}_i$ is an estimate of $E(Y_i | X_{i,1}, \ldots, X_{i,p}) = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \cdots + \beta_p X_{i,p}$
- $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ find the "best" predictor by minimizing

$$\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Q: Why minimize the square?

Residuals and Fitted Values

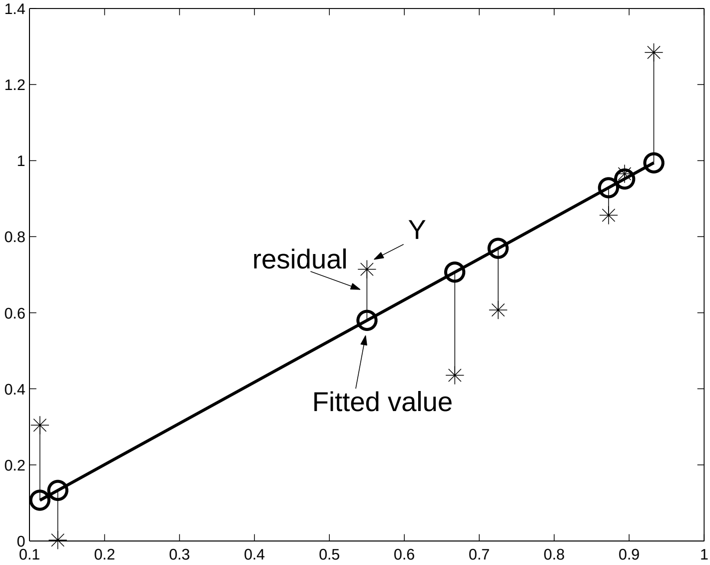
$$
\begin{aligned}
\text{Fitted value:} \quad \hat{Y}_i &= \hat{\beta}_0 + \hat{\beta}_1 X_{i,1} + \hat{\beta}_2 X_{i,2} + \cdots + \hat{\beta}_p X_{i,p} \\
\text{Residual:} \quad \widehat{\epsilon}_i &= Y_i - \hat{Y}_i \qquad (\text{estimates } \epsilon_i)
\end{aligned}
$$

Least-squares: makes the sum of square residuals

$$\sum_{i=1}^{n} \widehat{\epsilon}_i^2$$

as small as possible.

Least Squares, in pictures



When is a model linear?

Linear regression assumes

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon.$$

Definition: A model is **linear** if Y is linear in the **parameters** (linear in $\beta_0, \ldots, \beta_p$).

Key point: Covariates $X_1, \ldots, X_p$ can be anything observable. They can be nonlinear functions of measured quantities.

Nonlinear models

- Nonlinear regression is covered in more advanced courses
- For this course, you only need to know a nonlinear model when you see it

Example:

$$Y = \beta_0 + \beta_1 \exp(\beta_2 X) + \epsilon$$

Nonlinear models

- Nonlinear regression is covered in more advanced courses
- For this course, you only need to know a nonlinear model when you see it

Example:

$$Y = \beta_0 + \beta_1 \exp(\beta_2 X) + \epsilon$$

This model is nonlinear because Y is a nonlinear function of β_2.

Nonlinear models

- Nonlinear regression is covered in more advanced courses
- For this course, you only need to know a nonlinear model when you see it

Example:

$$Y = \beta_0 + \beta_1 \exp(\beta_2 X) + \epsilon$$

This model is nonlinear because Y is a nonlinear function of β_2.

Example:

$$Y = \beta_0 + \beta_1^3 \beta_2^4 \exp(X) + \epsilon$$

Nonlinear models

- Nonlinear regression is covered in more advanced courses
- For this course, you only need to know a nonlinear model when you see it

Example:

$$Y = \beta_0 + \beta_1 \exp(\beta_2 X) + \epsilon$$

This model is nonlinear because Y is a nonlinear function of β_2.

Example:

$$Y = \beta_0 + \beta_1^3 \beta_2^4 \exp(X) + \epsilon$$

This model is nonlinear in the parameters, but can be rewritten as a linear model in $\beta_1' = \beta_1^3 \beta_2^4$

$$Y = \beta_0 + \beta_1' \exp(X) + \epsilon$$

Which of these models are linear in the parameters?

$$1. Y = \beta_0 + \beta_1 \times (\mathrm{CavityWidth}) + \beta_2 \times (\mathrm{CavityWidth})^2,$$

$$2. Y = \beta_0 + \beta_1 \times (\mathrm{CavityWidth}) + (\beta_2 \times \mathrm{CavityWidth})^2,$$

$$3. Y = \beta_0 + \beta_1 \times (\mathrm{CavityWidth}) + \exp(\beta_2 \times \mathrm{CavityWidth}),$$

- (yes) none of them
- (no) 1 only
- (up) 2 only
- (down) 3 only
- (coffee) more than one

Which of these can be rewritten as a linear model?

$$1. Y = \beta_0 + \beta_1 \times (\mathrm{CavityWidth}) + \beta_2 \times (\mathrm{CavityWidth})^2,$$

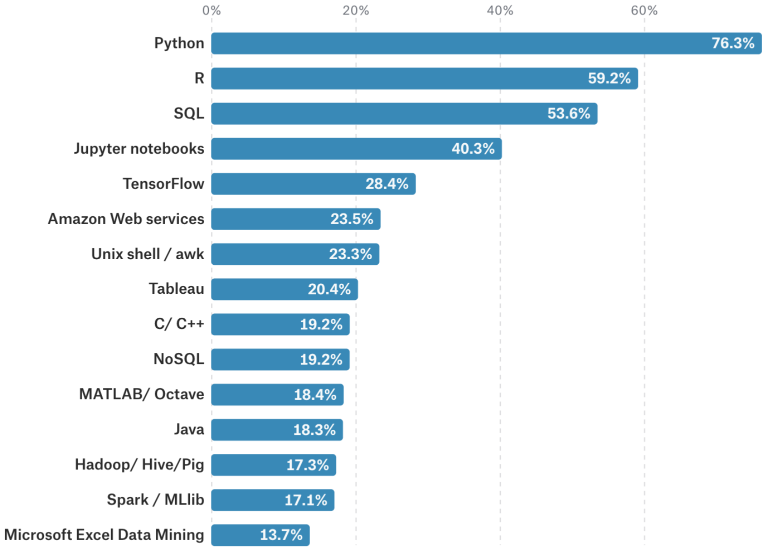
$$2. Y = \beta_0 + \beta_1 \times (\mathrm{CavityWidth}) + (\beta_2 \times \mathrm{CavityWidth})^2,$$

$$3. Y = \beta_0 + \beta_1 \times (\mathrm{CavityWidth}) + \exp(\beta_2 \times \mathrm{CavityWidth}),$$

- (yes) none of them
- (no) 1 only
- (up) 2 only
- (down) 3 only
- (coffee) more than one

Outline

Why python?


0%
20%
40%
60%
Python
76.3%
R
59.2%
SQL
53.6%
Jupyter notebooks
40.3%
TensorFlow
28.4%
Amazon Web services
23.5%
Unix shell / awk
23.3%
Tableau
20.4%
C/ C++
19.2%
NoSQL
19.2%
MATLAB/ Octave
18.4%
Java
18.3%
Hadoop/ Hive/Pig
17.3%
Spark / MLlib
17.1%
Microsoft Excel Data Mining
13.7%

How to access python

two options to use python:

- install Anaconda python distribution on your computer
- use Google colab: no installation needed

see `https://people.orie.cornell.edu/mru8/orie3120/read/jupyter.pdf` for details on using jupyter

A guide to python packages we use

- numpy (np): mathematical operations (esp. linear algebra)
- scipy: more mathematical operations (including statistics)
- pandas (pd): manipulate data tables (dataframes)
- matplotlib (plt): for plotting
- seaborn: for statistical plots
- sklearn: for machine learning (regression and beyond)
- statsmodels (sm): statistical models

A guide to python packages we use

- numpy (np): mathematical operations (esp. linear algebra)
- scipy: more mathematical operations (including statistics)
- pandas (pd): manipulate data tables (dataframes)
- matplotlib (plt): for plotting
- seaborn: for statistical plots
- sklearn: for machine learning (regression and beyond)
- statsmodels (sm): statistical models

Q: why so many packages?

A guide to python packages we use

- numpy (np): mathematical operations (esp. linear algebra)
- scipy: more mathematical operations (including statistics)
- pandas (pd): manipulate data tables (dataframes)
- matplotlib (plt): for plotting
- seaborn: for statistical plots
- sklearn: for machine learning (regression and beyond)
- statsmodels (sm): statistical models

Q: why so many packages?
A: python is a lightweight, flexible language with a large developer community

Outline

Example: Electricity Usage

- We are managing a large complex of apartments in the Northeast.
- We pay for the electricity used by our residents.
- We would like to predict electricity usage so that we can estimate how much money should be set aside.

Demo:
`https://github.com/madeleineudell/orie3120-sp2020/blob/master/demos/linear-regression.ipynb`

Predictors (Independent Variables)

The following predictors were measured during data collection:

- year (1989 – 1994)
- month (1 – 12)
- usage (of electricity for month)
- temperature (average temperature for month)

We added two additional predictors:

- yearcts $=$ year $+$ (month - 1)$/$12.
- tempsqr $=$ temperature^2.

Here is python code for loading and plotting the data

```
usage = pd.read_csv(’elec_usage.txt’)
usage[’tempsqr’] = usage[’temperature’]^2
usage[’yearcts’] = usage[’year’] + (usage[’month’]-1)/12
usage.plot.scatter(x=’temperature’, y=’usage’)
seaborn.pairplot(usage)
```

We fit this linear regression model

$$\texttt{usage} = \beta_0 + \beta_1 \texttt{temp} + \beta_2 \texttt{temp}^2 + \epsilon$$

Here

- $Y = \texttt{usage}$
- $X_1 = \texttt{temp}$
- $X_2 = \texttt{temp}^2$

Here's how we fit this linear regression model

```
Y, X = dmatrices('usage ~ 1 + temperature + temperature^2
                 data=usage, return_type='dataframe')
model = sm.OLS(Y, X).fit()
```

Here's how to get Python to tell us the estimated regression coefficients

```
model.summary()
```

Output of model summary

```
  OLS Regression Results
==============================================================================
Dep. Variable:                  usage   R-squared:                       0.785
Model:                            OLS   Adj. R-squared:                  0.768
Method:                 Least Squares   F-statistic:                     45.76
Date:                Sun, 05 Apr 2020   Prob (F-statistic):           4.00e-16
Time:                        15:16:34   Log-Likelihood:                -210.02
No. Observations:                  55   AIC:                             430.0
Df Residuals:                      50   BIC:                             440.1
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                      coef    std err          t      P>|t|      [0.025      0
------------------------------------------------------------------------------
Intercept        -289.2222   2358.062     -0.123      0.903   -5025.530    444
temperature        -0.5207      0.792     -0.657      0.514      -2.112
temperature ^ 2    -0.8402      0.785     -1.071      0.290      -2.416
month              -1.0414      8.117     -0.128      0.898     -17.344      1
year              -11.9054     97.321     -0.122      0.903    -207.381     18
yearcts            12.1097     98.509      0.123      0.903    -185.751     20
==============================================================================
Omnibus:                        0.462   Durbin-Watson:                   1.166
Prob(Omnibus):                  0.794   Jarque-Bera (JB):                0.216
Skew:                           0.153   Prob(JB):                        0.898
```

This command also provided other statistical analyses

- Standard errors, p-values and t-values for each β_j
- Residual standard error
- Multiple R-squared and Adjusted R-squared
- F statistic and associated p-value

When are these statistical analyses appropriate?

Statistical quantities in summary (standard errors, p-values, t-values, R squared, F statistic, ...) are computed assuming that $\epsilon_1, \ldots, \epsilon_n$

1. are mutually independent
2. are independent of $X_{i,j}$
3. are normally distributed
4. have a constant variance

- when these assumptions are true, we can use these quantities to lead us towards better models.
- if these assumptions are false, these statistics can be misleading.
- we can alter the data to make assumptions more true; sometimes improves model fit, too!

To check whether these assumptions are true, we must look at the residuals. We will show how to do this later.

Regression coefficients and standard errors

- Estimated regression coefficients: $\hat{\beta}_0, \ldots, \hat{\beta}_p$
- Std. Errors
  - the jth std. error is the standard deviation of $\hat{\beta}_j$
  - an approximate 95% confidence interval for β_j is Estimate $\pm$ (2)(Std. Error)
  - Used to quantify error in the estimates

Statistical tests

we may assess how well the model fits using a variety of statistical tests:

- t value
  - t value = Estimate $\hat{\beta}_j$ / Std. Error of $\hat{\beta}_j$
  - how big is the coefficient's mean relative to its error?
- p value
  - the probability of a t value as large or larger than the one actually observed, if $\beta_j = 0$.
  - If this probability is small, then β_j is probably not 0.
  - Formally, we reject the hypothesis that $\beta_j = 0$ at the $\alpha = 0.05$ significance level if p value < 0.05.

More statistical tests

- Residual std. error
  - estimates the standard deviation of the ϵ_i
- F statistic
  - tests the hypothesis that $\beta_1 = 0, \beta_2 = 0, \ldots,$ and $\beta_p = 0$ i.e., that Y is NOT related to ANY of the predictors
  - often it is obvious that this hypothesis is false
  - used occasionally for more sophisticated procedures built on top of linear regression.

Variance explained

- Multiple R Squared
  - A number between 0 and 1 that tells how well the data is explained by the linear model.
  - Can be used to choose between different predictors.
- Adjusted R Squared
  - Like multiple R squared, but adjusted for the number of predictor variables.
- These are explained on the next few slides.

Multiple R Squared

- Multiple R Squared is the squared correlation between the observed and fitted values,

$$R^2 = \rho(Y, \hat{Y})^2$$

- it is "multiple" because $\hat{Y}$ uses all of the predictors
- $\rho(Y, \hat{Y})$ is the sample correlation

$$\rho(Y, \hat{Y}) = \frac{\sum_i (Y_i - \mathrm{avg}(Y))(\hat{Y}_i - \mathrm{avg}(\hat{Y}))}{\sqrt{\sum_i (Y_i - \mathrm{avg}(Y))^2}\sqrt{\sum_i (\hat{Y}_i - \mathrm{avg}(\hat{Y}))^2}}$$

- The closer R^2 is to 1, the better $\hat{Y}_i$ predicts Y_i.

Why use R^2?

- R^2 can be used to determine which set of predictors is best.
  - bigger R^2 is better
- problem: R^2 is biased in favor of more predictors
  - adding predictors increases R^2
  - even if the additional predictors are not related to Y_i
- The bias of R^2 can be fixed by using

$$\text{Adjusted } R^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

  (recall n=number of observations, p=number of covariates)
- Adjusted R^2 is preferred to R^2

Outline

How to check assumptions that undergird statistics?

Statistics computed are valid if $\epsilon_1, \ldots, \epsilon_n$

1. **independence I:** are mutually independent
2. **independence II:** are independent of covariates
3. **normality:** are normally distributed
4. **homoskedasticity:** have a constant variance

To check whether these assumptions are true, we must look at the residuals

Residuals analysis: mutual independence

Let's look at each assumption and see how it can be checked.

Assumption 1. $\epsilon_1, \ldots, \epsilon_n$ are mutually independent

- this assumption might be violated if the observations are in time or spatial order
- check by: plotting $\widehat{\epsilon}_i$ versus $\widehat{\epsilon}_{i-1}$
  - should see no pattern

Residuals analysis – checking mutual independence with a scatterplot

This code will show scatterplots from data with mutual independence.

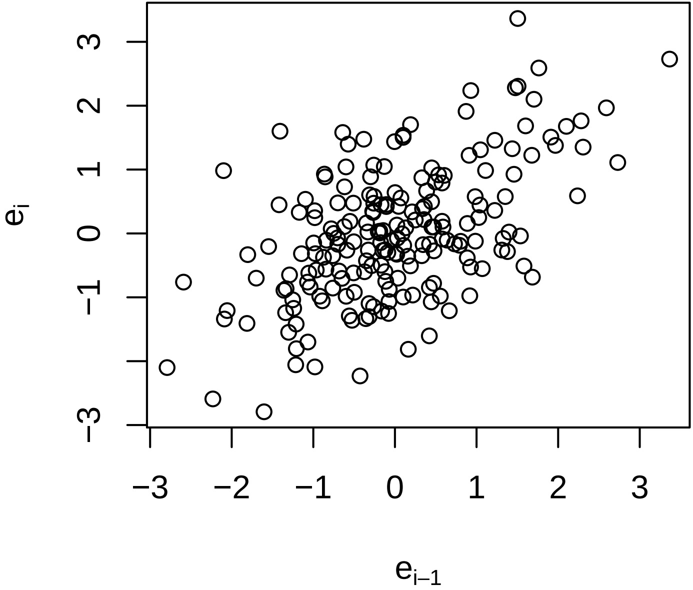
```
# generate data
n   = 500 # number of observations
eps = randn(n) # independent normal(0,1)
x   = 10*rand(n) # uniform(0,10)
y   = x + eps

# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid

plt.scatter(resid[:-1], resid[1:])
```

Residuals analysis – checking mutual independence with a scatterplot

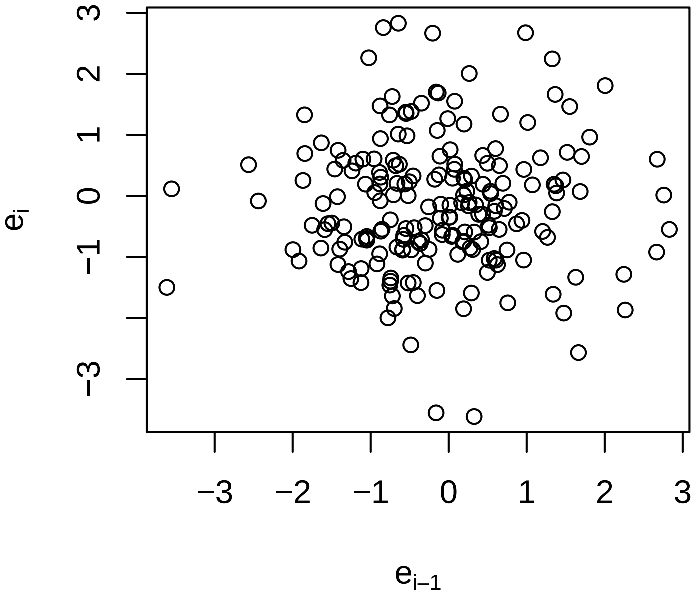
This code will show scatterplots from data **without** mutual independence.

```
# generate data
n   = 500   # number of observations
a   = 1     # use this to control the correlation
w   = randn(n+1) # independent normal(0,1)
eps = w[:-1] + a*w[1:] # normal, not independent
x   = 10*rand(n) # uniform(0,10)
y   = x + eps

# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid

plt.scatter(resid[:-1], resid[1:])
```

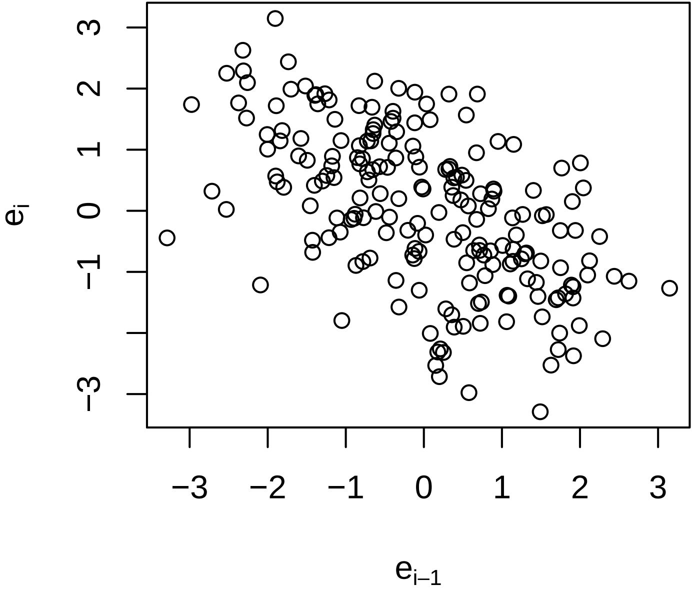
Mutually independent residuals? yes/no



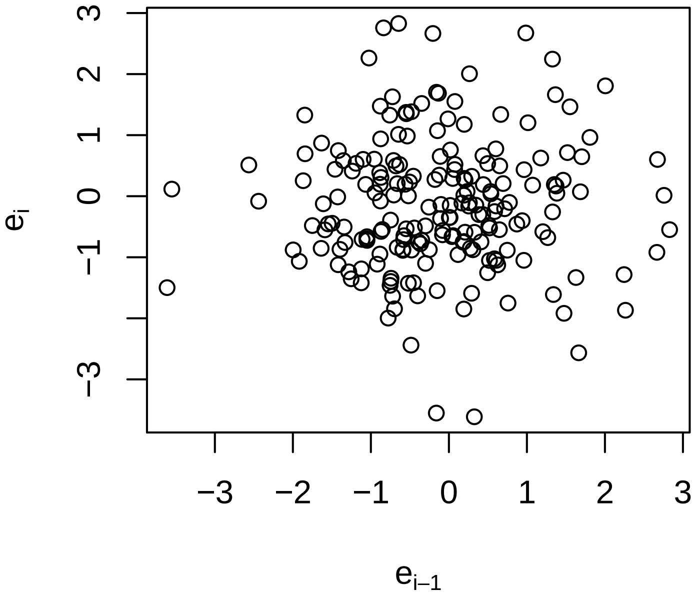
Mutually independent residuals? yes/no



Mutually independent residuals? yes/no



Mutually independent residuals? yes/no



Check mutual independence with autocorrelations

- plot the autocorrelation function:

$$r(t) = \mathrm{corr}(\widehat{\epsilon}_i, \widehat{\epsilon}_{i-t})$$

  - $r(t)$ should be 0 for all $t > 0$ (except for random variation)
  - no (or only a few) autocorrelations should be outside the test bounds
  - t is called the lag
- the scatterplots only looked at lag $= 1$
  - of course, we could have looked at other lags
  - but autocorrelations let us look at all lags simultaneously

Check mutual independence with autocorrelations

This code plots the autocorrelation for data with mutual independence.

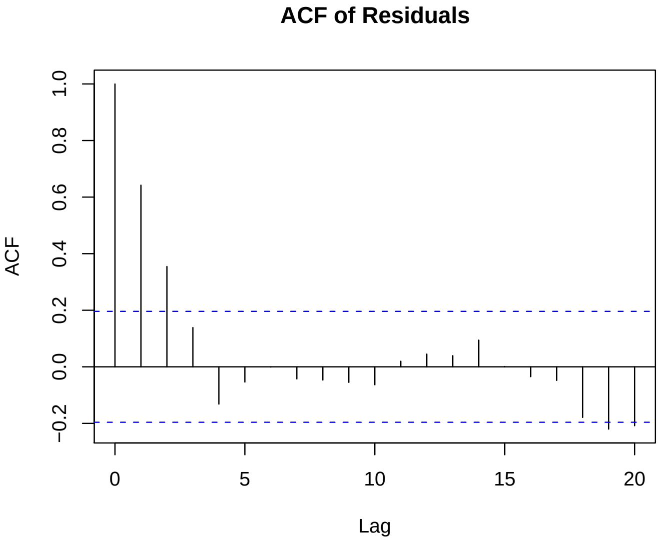
```
n   = 500   # number of observations
eps = randn(n) # independent normal(0,1)
x   = 10*rand(n) # uniform(0,10)
y   = x + eps

# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid

plt.acorr(resid)
```

Check mutual independence with autocorrelations

This code plots the autocorrelation for data **without** mutual independence.

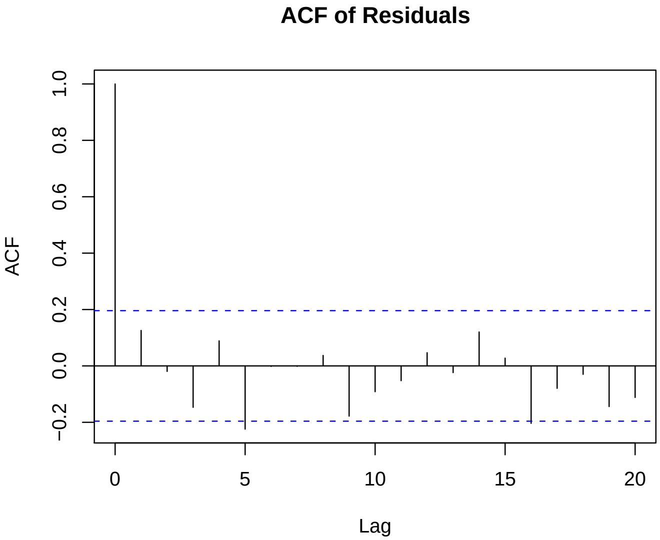
```
n   = 500   # number of observations
a   = 1     # use this to control the correlation
w   = randn(n+1) # independent normal(0,1)
eps = w[:-1] + a*w[1:] # normal, not independent
x   = 10*rand(n) # uniform(0,10)
y   = x + eps

# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid

plt.acorr(resid)
```

Check mutual independence with autocorrelations



quick poll: (yes) independent (no) not independent

Check mutual independence with autocorrelations



quick poll: (yes) independent (no) not independent

Residuals analysis – linear in the predictors

Assumption 2. model is linear in the predictors (the $X_{i,j}$)

- equivalently, $\epsilon_1, \ldots, \epsilon_n$ are independent of all $X_{i,j}$
- Check by: plotting $\widehat{\epsilon_i}$ versus $X_{i,j}$ for $j = 1, \ldots, p$
- we should see that the average value of the $\widehat{\epsilon_i}$ does not depend on $X_{i,j}$.
- if it does, then there is a problem

Residuals analysis – linear in the predictors

Plot residuals vs covariates to test linearity

```
plt.subplot(2,1,1)
p = plt.scatter(x,y,marker='o',label="observed")
plt.scatter(x,yhat,marker="+",color="red",label="
plt.legend()

plt.subplot(2,1,2)
plt.scatter(x,resid)
plt.xlabel("x")
plt.ylabel("residual")
```

Residuals analysis – linear in the predictors

This code forms a model for which outcome is **not** linear in the predictor.

```
n   = 500 # number of observations
eps = randn(n) # independent normal(0,1)
x   = 10*rand(n) # uniform(0,10)
y   = x + x**2 + eps

# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid
yhat = model.predict()
```

Residuals analysis – linear in the predictors

Here's how we fix the fit on the previous slide:
use the square as a feature

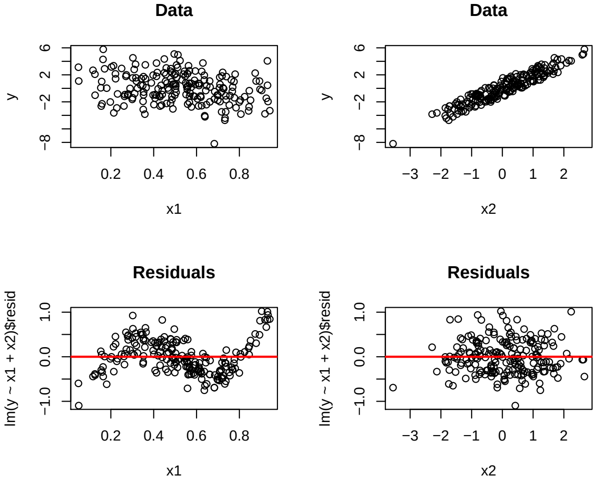
```
df = pd.DataFrame()
df['x'] = x
df['xsq'] = x**2
model = sm.OLS(y, df).fit()
```

Residuals detect nonlinearity better than raw data

In the next slide, data are simulated from:

```
n = 200
x1 = beta(2,2,n)
x2 = randn(n)
y = sin(2*math.pi*x1) + 2*x2 + .23*randn(n)
```

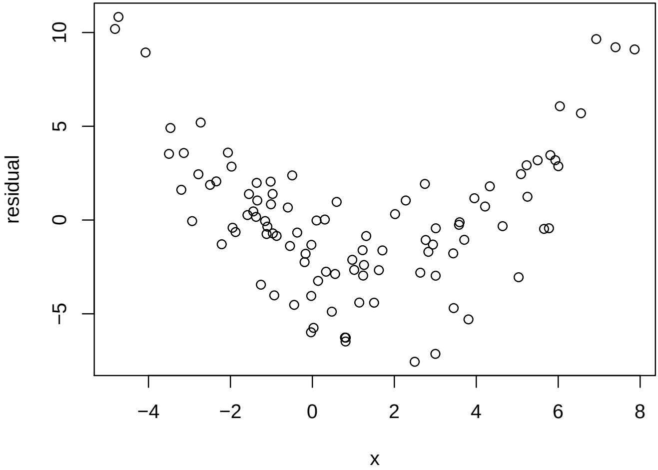
Residuals detect nonlinearity better than raw data



Raw data: Scatter due to X_2 obscures nonlinearity in X_1

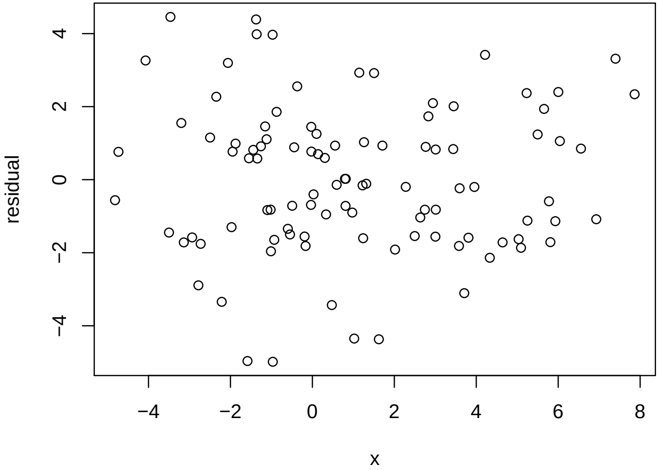
Residuals: Scatter due to X_2 is removed and nonlinearity in X_1 is revealed

Checking for nonlinearity



quick poll: (yes) linear in x (no) not linear in x

Checking for nonlinearity



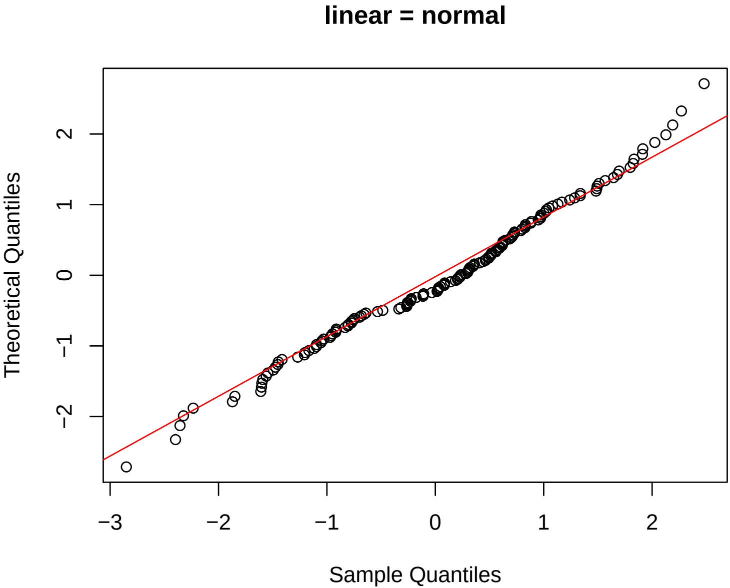
quick poll: (yes) linear in x (no) not linear in x

Residuals analysis – normal distribution

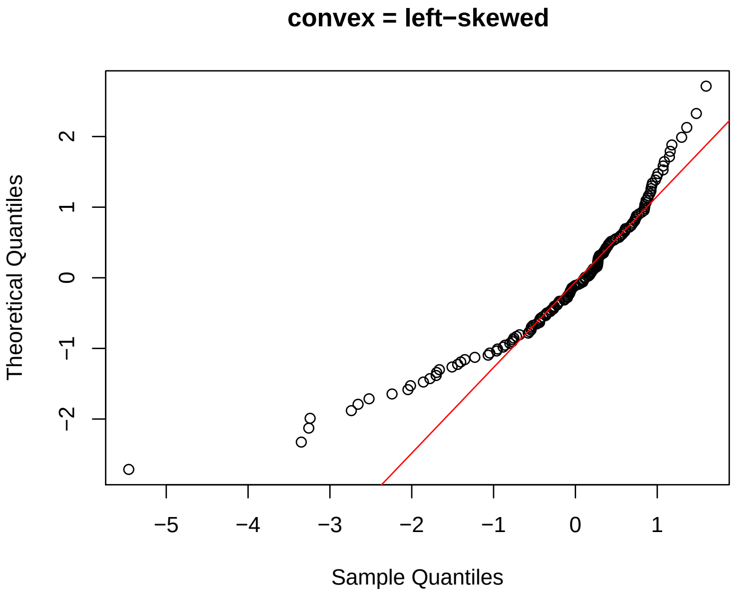
Assumption 3. $\epsilon_1, \ldots, \epsilon_n$ are normally distributed

- normal probability plot
- should see a straight line
- a pattern means skewness or heavy-tails

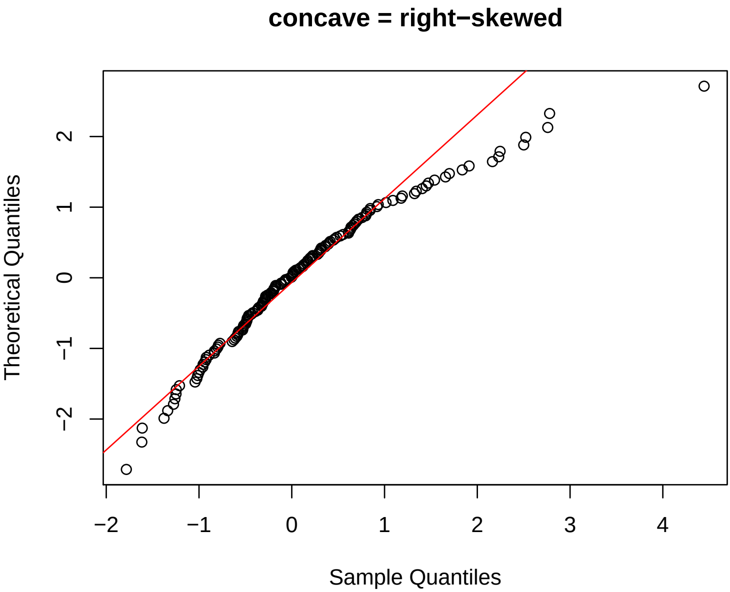
Interpreting normal plots



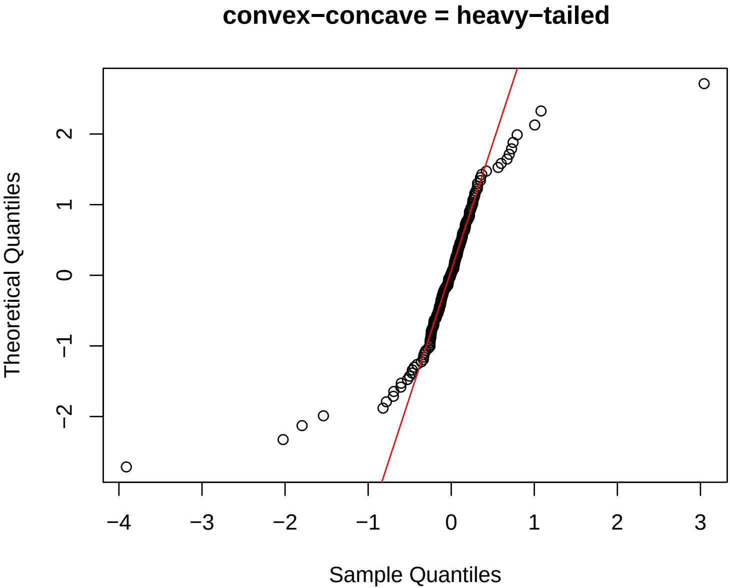
Interpreting normal plots



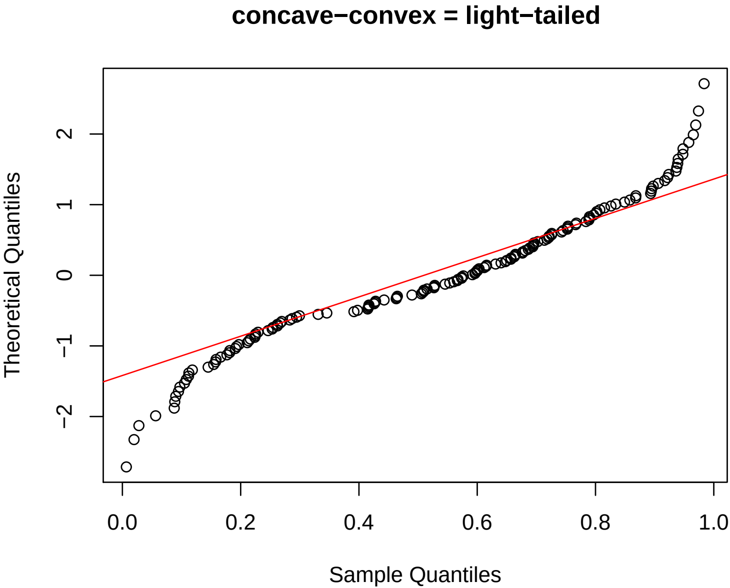
Interpreting normal plots



Interpreting normal plots



Interpreting normal plots



Checking for normal errors

This code generates q-q plots for normal residuals:

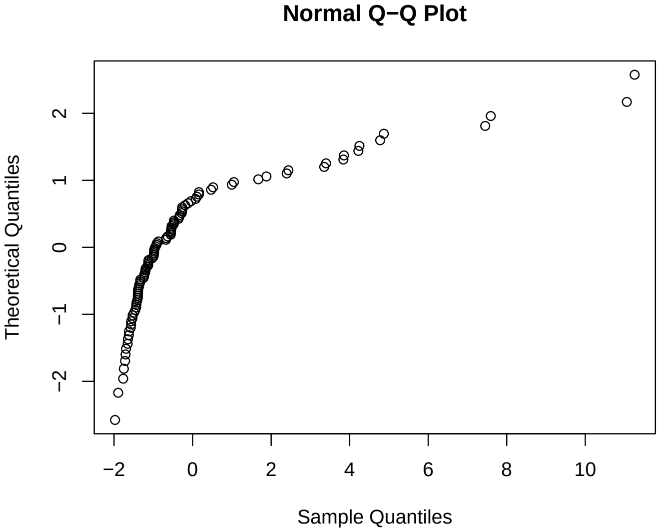
```
n=500
eps = randn(n) # normal residuals
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

sm.qqplot(model.resid, line='45');
```

Checking for normal errors

This code generates q-q plots for residuals that are **not** normal:

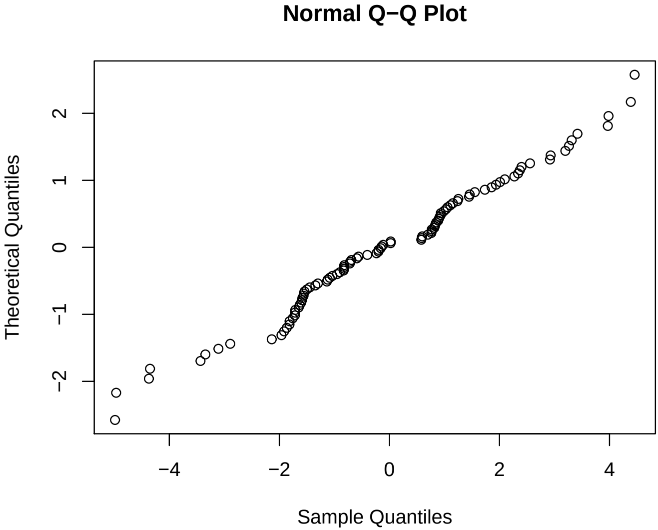
```
n=500
eps = exp(randn(n)) # not normal
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

sm.qqplot(model.resid, line='45');
```

Checking for normal errors



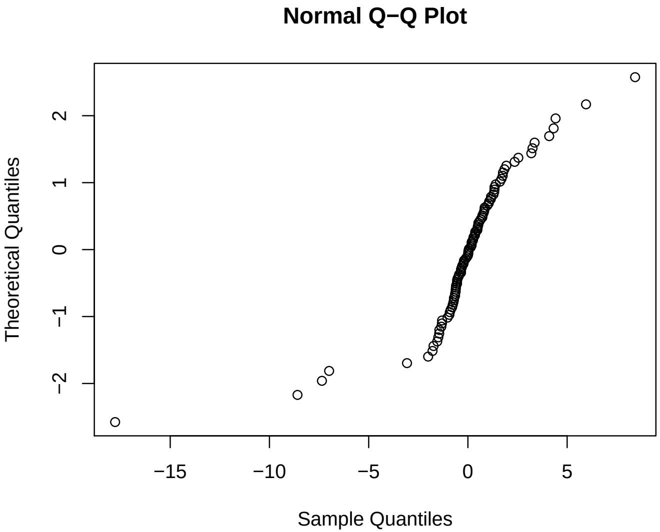
poll: (yes) residuals are normal (no) residuals are not normal

Checking for normal errors



poll: (yes) residuals are normal (no) residuals are not normal

Checking for normal errors



poll: (yes) residuals are normal (no) residuals are not normal

Residuals analysis – constant variance

Assumption 4. $\epsilon_1, \ldots, \epsilon_n$ have a constant variance

- plot absolute residuals against fitted values
- plot absolute residuals against $X_{i,j}$ for each j
  - should see that the distribution does not depend on $X_{i,j}$
  - if it does, then the variance is not constant
- we call non-constant variance "heteroscedasticity"

Checking for normal errors

This code generates absolute residual plots for constant variance:

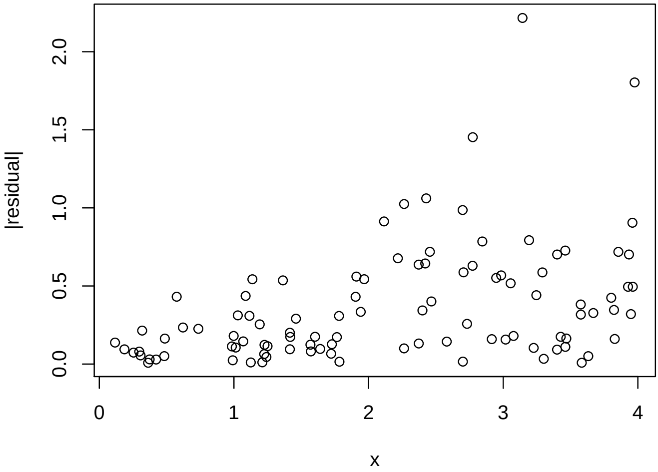
```
n=500
eps = randn(n)
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()
plt.scatter(x,np.abs(model.resid))
```

Checking for normal errors

This code generates absolute residual plots for **non-constant** variance:

```
n=500
x = 10*rand(n)
eps = x*randn(n) # variance of noise depends on x
y = x + eps
model = sm.OLS(y,x).fit()
plt.scatter(x,np.abs(model.resid))
```

Checking for non-constant variance



poll: (yes) variance is constant (no) variance is not constant

Strategy for regression data analysis:

1. Decide: what problem(s) are you trying to solve?
   - keep the problem in mind while doing the remaining steps
2. Find (or collect) useful data
3. Find a useful model
   - all models are wrong (George Box)
   - some models are useful
4. Check model
   - how well does the model fit the data?
5. Modify model, if necessary
6. Use model to solve problem(s)