Data Analysis

Christopher Simpkins
chris.simpkins@gatech.edu

Data Analysis

- Modeling data
- Single-variable data analysis
  – Comparing data to theoretical distributions with probability plots
- Two-variable data analysis
  – Linear regression

Lecture material taken or adapted from Phillip K. Janert, Data Analysis with Open Source Tools, O'Reilly, 2010

Modeling Data

- A model is a simplified representation of a data set
  - A Normal distribution specified by a mean and a variance
  - A linear relationship between two variables specified with a slope and a y-intercept
- All models are wrong, some are useful
- We seek useful models with goodness of fit tests
- What can we do with a model?
  - Understand
  - Describe
  - Predict

Normal Quantile Plots

- Say we have a data set we think is drawn from a normal distribution
- If we plot the data in ascending order versus the quantiles of the standard normal (N(0, 1)) distribution, we should get a nearly straight line with positive slope
- A quantile of a sample, q(f), is a value for which a specified fraction f of values is less than or equal to the value q(f)
  - For example, the median is q(0.5), because exactly half the values should be less than or equal to the median
- Intuition: if we plot the 1st through nth values of our data against the corresponding quantiles, we're plotting cumulative distribution against normal quantiles, which should give us a straight line

Calculating the Quantiles

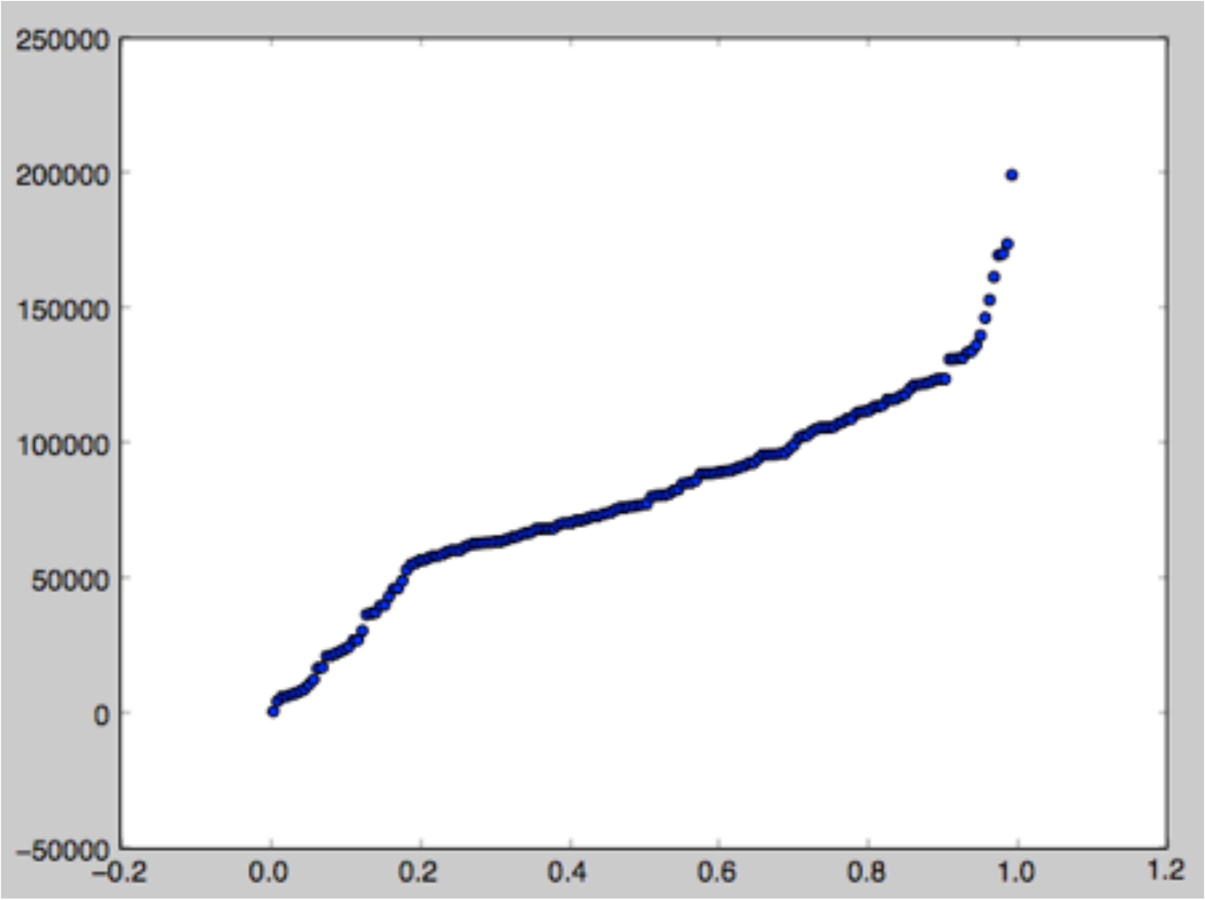
- The quantiles of a standard normal distribution is complicated
- We can approximate them with a simple procedure:
  - For each rank i in our data set, starting with 1 (not 0)
    - » The quantile i is i/(n+1), where n is the number of data points in our data set
- So, we have our data, which will be plotted on the y-axis, and we can generate the quantiles, which will be plotted on the x-axis
- Let's do this in PyLab

Generating a Normal Quantile Plot

- Generating a normal quantile plot is simple in PyLab

```
In [1]: data = loadtxt('lab1-all.dat')

In [2]: data.sort()

In [3]: n = len(data)

In [4]: quantiles = [float(i)/float(n+1) for i in range(1, n+1)]

In [5]: scatter(quantiles, data)
Out[5]: <matplotlib.collections.PathCollection at 0x104798450>
```

The Normal Quantile Plot for Salary Data



- Looks roughly straight, especially if you discard outliers
- So we can conclude that our salary data is normally distributed

Normal Data

- Once we know we have normally distributed data, we can use the plethora of mathematical tools that apply to normally-distributed data
  - Summary statistics, hypothesis tests
- As an example, let's look at the salary data for a different organization and see if the average salaries are different

Comparing Two Gaussian Data Sets

```
In [1]: lab1 = loadtxt('lab1-all.dat')

In [2]: lab2 = loadtxt('lab2-all.dat')

In [3]: lab1.mean()
Out[3]: 80747.553293413162

In [4]: lab2.mean()
Out[4]: 100692.06108108112
```

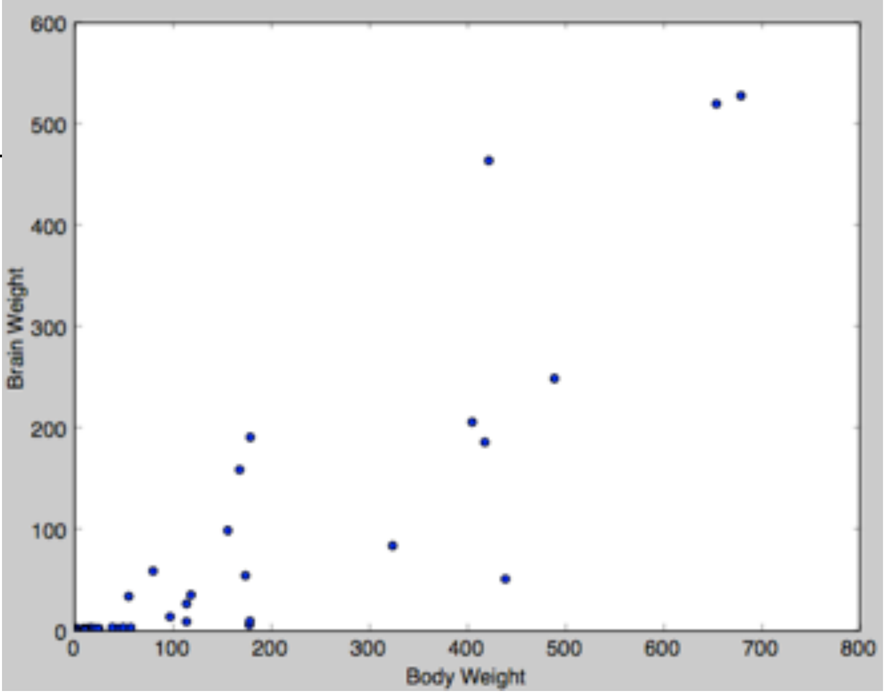
- Looks like there's a big difference in salaries between these two organizations
- But is the difference statistically significant?
- This is the sort of question we can answer now that we know we have normally-distributed data sets

Linear Regression

- A linear regression model relates a dependent variable to an independent variable linearly, that is, $y = mx + b$
  - y is also called the response variable and x the regressor
- If a data set is a good fit for a linear regression model, then we can predict the values of the response variable for any value of the regressor (not just the pairs in the data set)
- Recall the brain weight as a function of body weight example from the previous lecture (next slide)

Scatter Plot of Brain Weight vs. Body Weight

```
In [70]: brain = loadtxt('brain-body.dat', usecols=(1,))
In [71]: body = loadtxt('brain-body.dat', usecols=(2,))
In [72]: scatter(body, brain)
Out[72]: <matplotlib.collections.PathCollection at 0x1189a1ad0>

In [73]: ylabel('Brain Weight')
Out[73]: <matplotlib.text.Text at 0x116edecd0>

In [74]: xlabel('Body Weight')
Out[74]: <matplotlib.text.Text at 0x116ee1810>

In [75]: axis([0, 800, 0, 600])
Out[75]: [0, 800, 0, 600]
```



R^2

- R^2, or the coefficient of determination, tells us how well a linear model will predict the response variable given the regressor
  – More precisely, it tells us how much the model explains the data, or reduces the mean square error of guesses
  – An R2 of .70 means that the model reduces MSE of guesses by 70%; higher R2 values are better

$$R^2 = 1 - \frac{Var(\epsilon)}{Var(Y)}$$

- For linear least squares models, R^2 is equal to the square of Pearson's correlation coefficient, ρ
- This makes it easy to calculate R^2 in PyLab

Calculating R^2 in PyLab

- `numpy.corrcoef()` returns a matrix of correlation coefficients. We want [0,1] (a single number)
- So here's how we calculate R^2 in PyLab

```
In [25]: p = corrcoef(body, brain)[0,1]

In [26]: p**2
Out[26]: 0.78926956832186612
```

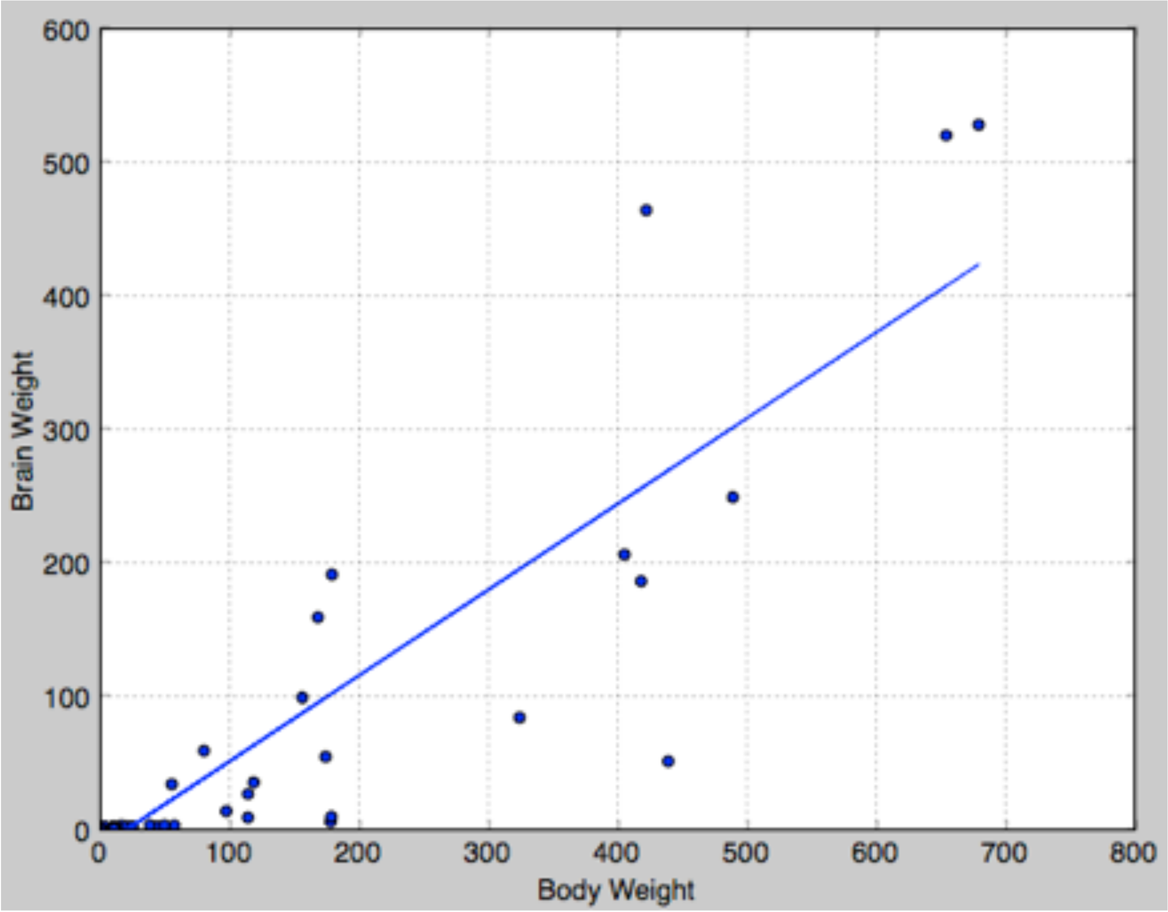
- Looks like a linear least squares model will fit the data well
- Now let's actually calculate the line

Fitting a Linear Least Squares Regression Line in PyLab

- Calculate the slope and intercept, m, b, with
  - `polyfit(x, y, 1)`
  - The 1 means fit a first-degree polynomial (a line)
- Generate the y values of the line with
  - `polyval([m, b], x)`
  - Note that for our data, x is body weight, y is brain weight
  - Here's how it looks

```
In [27]: (m, b) = polyfit(body, brain, 1)

In [28]: y = polyval([m, b], body)

In [29]: plot(body, y)
Out[29]: [<matplotlib.lines.Line2D at 0x103f38710>]

In [30]: grid(True)
```

Our Finished Regression Line



- And we can use the `polyval` function to predict the response variable for any regressor value

Conclusion

- We've now seen
  – a visual test for goodness of fit of a Normal distribution to a univariate data set
  – a numerical test for goodness of fit of a linear least squares regression line to a bivariate data set
- We've seen examples of what can be done once we have models
  – Prediction of unseen values
  – Comparing means of the normally-distributed data
- We've only scratched the surface of quantitative data analysis
  – But you now have a feel for the process and practical tools to do it yourself