14 – Lease Squares Fit to a Line

Physics 281 – Class 14

Grant Wilson

Class 13 Exercises

```
from matplotlib import pyplot as plt
from scipy.stats import chi2 as chi2
from scipy.stats import norm as norm
import numpy as np

#E13.1
#experiment with 40 degrees of freedom
v = 40.
#measured chisq = 80
chisq=80.

#the probability of getting this value by chance is
#given by the cumulative distribution function
prob_less = chi2.cdf(chisq,v)
prob_greater = 1.-prob_less
print "Probability of pulling a smaller value at random: "+str(prob_less*100.)+'%'
print "Probability of pulling a larger value at random: "+str(prob_greater*100.)+'%'
```

Class 13 Exercises

```
#E13.2
#simulation shows that random drive times are drawn from a normal
#distribution with mean=23 min. and std_dev = 0.75 min.
#The "confidence" is just the probability that the actual drive
#time is NOT likely to have been pulled from this distribution at random.
actual = 21.
mu = 23.
sigma = 0.75

#again, use the cdf to get the probability
prob_less = norm.cdf(actual,loc=mu,scale=sigma)
prob_more = 1.-prob_less

print "The probability that Adnan could have made the drive in less than or equal to 21 minutes is " + str(prob_less)+'%'
print "This is a "+str((23.-21.)/sigma)+" sigma result."
```

14 – Lease Squares Fit to a Line

Physics 281 – Class 14

Grant Wilson

Linear Interpolation is **not** fitting to a line.

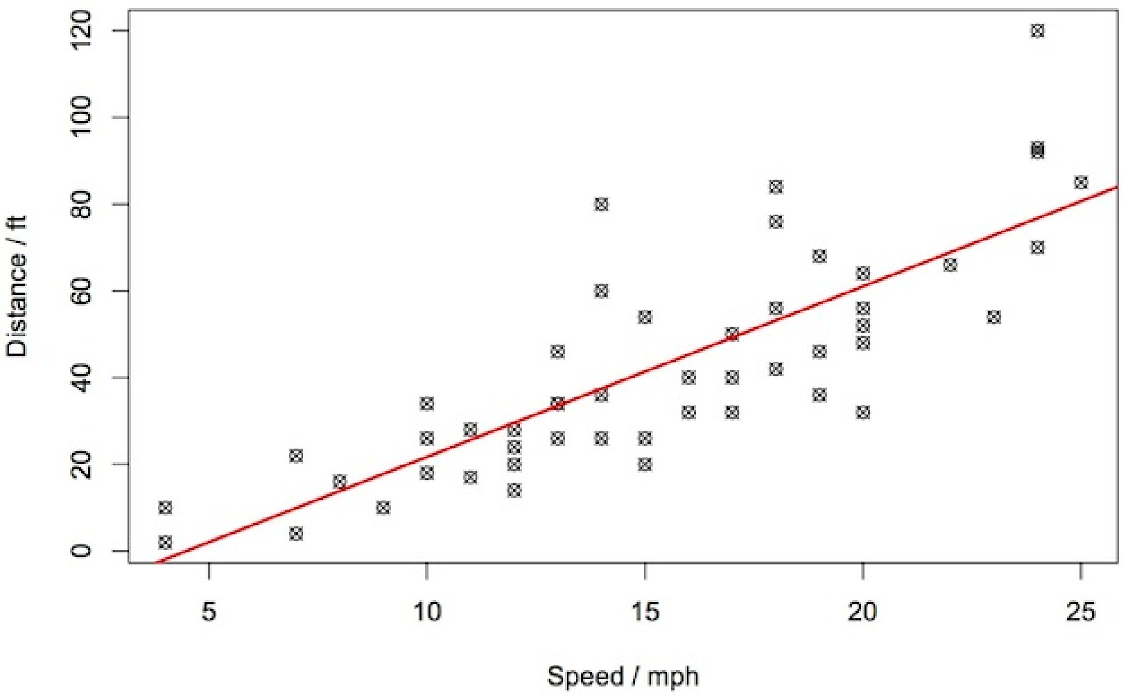
Fitting to a line is **not** linear interpolation.

Why fit data to a model?

Why fit data to a model?

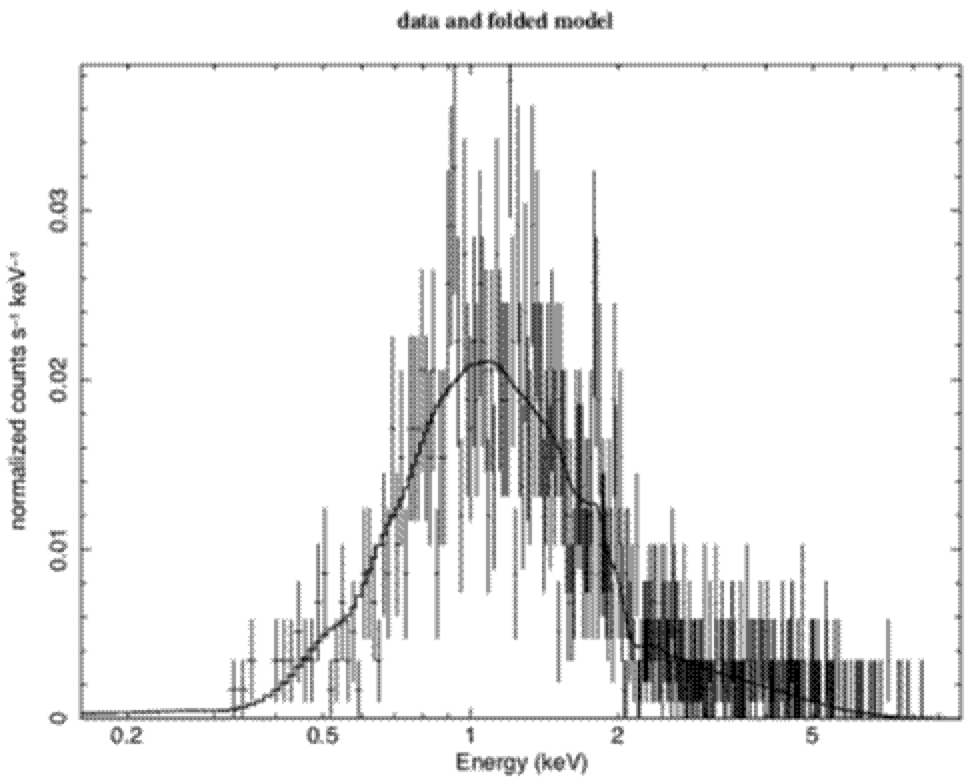
- gain insight about a physical system
- use knowledge of a physical system to illuminate trends in our data
- data compression
- extrapolation (!)
- Explore consistency between model and data
- provide a continuous expression of values around where we have measurements (even if the measurements are noisy)

What does it mean to find a good fit?



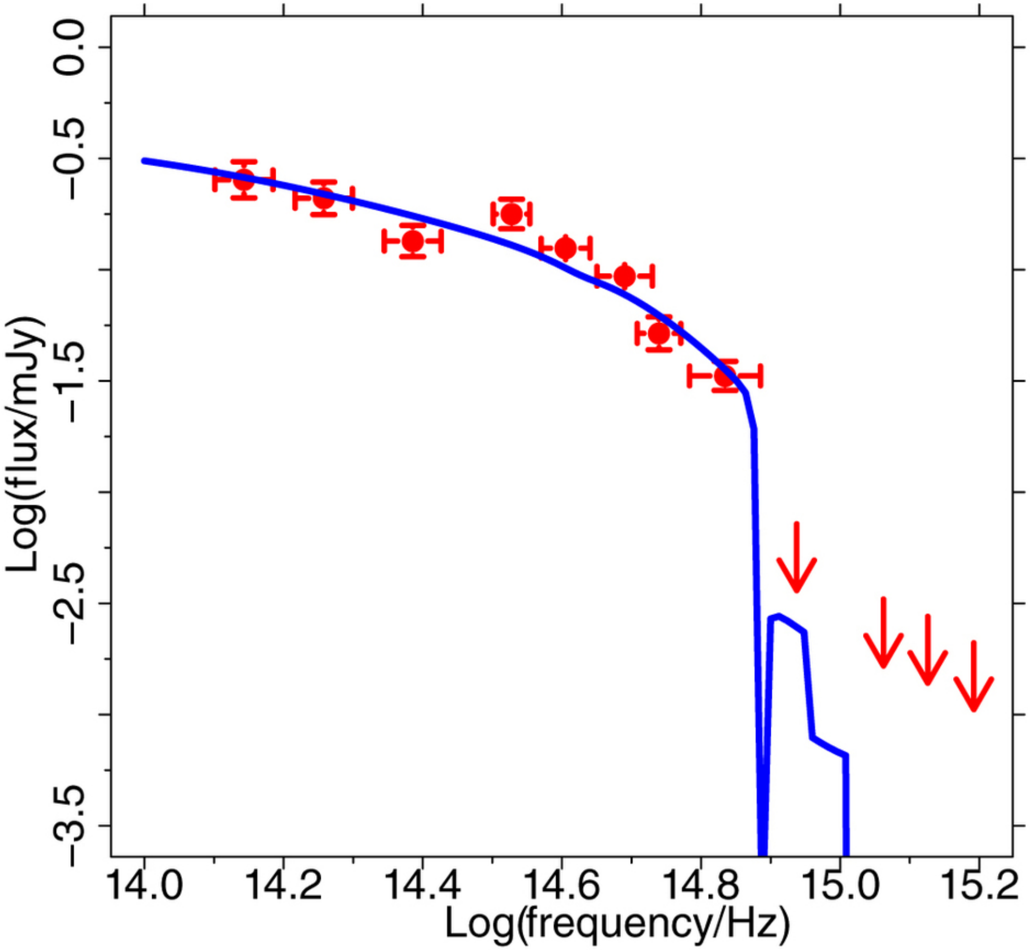
Is this a good fit to the data?

What does it mean to find a good fit?



Is this a good fit to the data?

What does it mean to find a good fit?

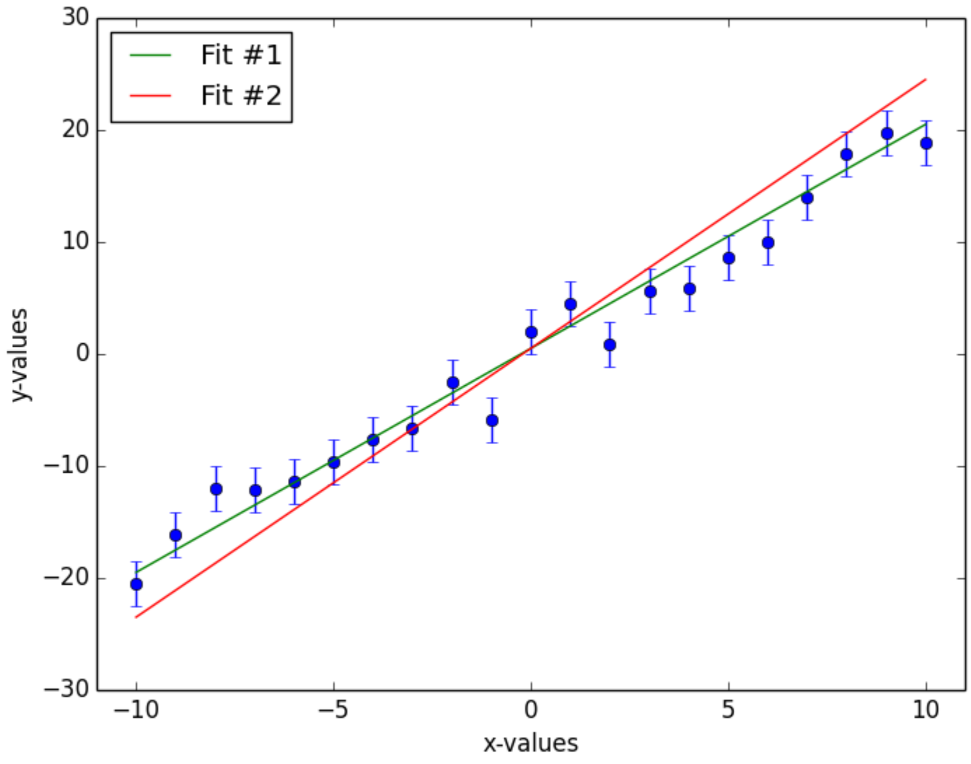


Is this a good fit to the data?

What does it mean to find the “best fit?”

What does it mean to find the “best fit?”

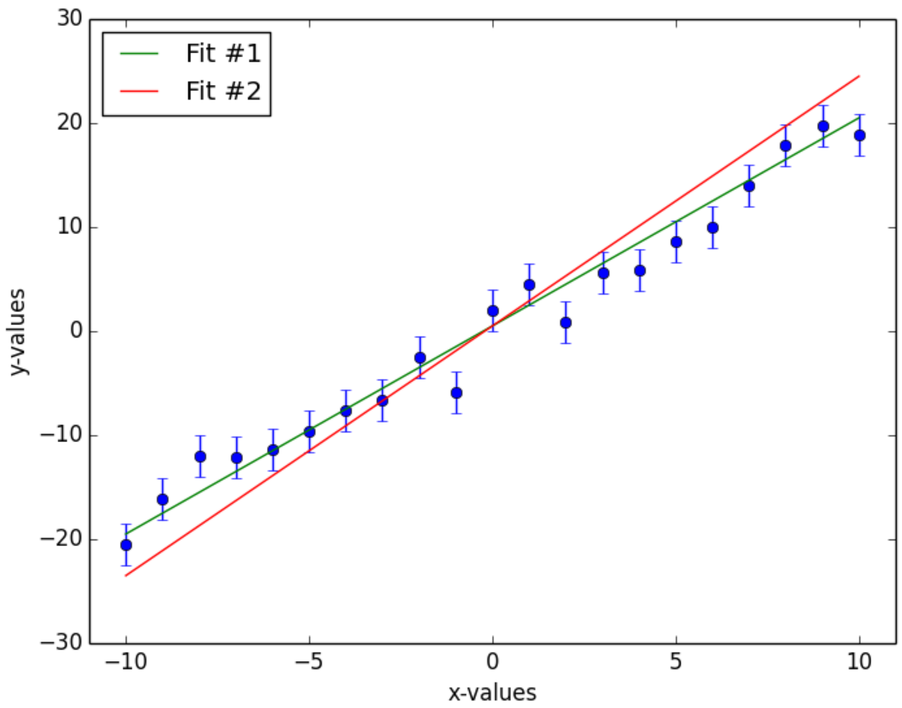
- Clearly we need some kind of statistic to tell us what to do.

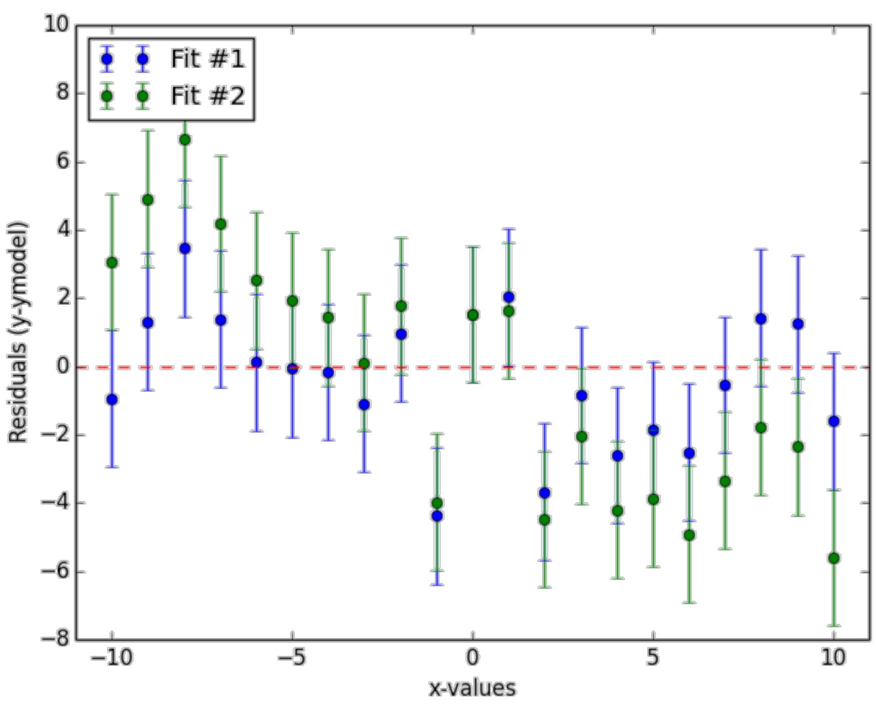


Which fit is better?
Why?

What does it mean to find the "best fit?"

- Clearly we need some kind of statistic to tell us what to do.





Enter the χ^2 Statistic

- Remember

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - f(x_i; \vec{a}))^2}{\sigma_i^2}$$

- This is an error-weighted sum of the squares of the distances (in y) between the model and the data.

- Least Squares Fitting is the art of finding the model that **minimizes** this quantity.

Fitting Data to a Line

- Start off with something relatively easy and manageable.

- Fit a set of N-data points, (x,y) with error σ, to a line of the form

$$y = a + bx$$

- Recipe for Approach #1
  1. write down expression for χ^2
  2. find equations for minimizing χ^2 for each of our "free parameters"
  3. solve the system of equations from step #2

- Step 1 - write down expression for χ^2

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - (a + bx_i))^2}{\sigma^2}$$

- Step 2 – find equations for minimizing χ^2 for parameters *a* and *b*.

$$\frac{d\chi^2}{da} = 0$$

$$\frac{d\chi^2}{db} = 0$$

- Step 1 - write down expression for χ^2

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - (a + bx_i))^2}{\sigma^2}$$

- Step 2 – find equations for minimizing χ^2 for parameters *a* and *b*.

$$\frac{d\chi^2}{da} = 0 \xrightarrow{\text{yields}} -2\sum_{i=1}^{N}(y_i - a - bx_i) = 0$$

$$\frac{d\chi^2}{db} = 0 \xrightarrow{\text{yields}} -2\sum_{i=1}^{N} x_i(y_i - a - bx_i) = 0$$

- Now we have two simultaneous equations in two unknowns:

$$Na + \sum x_i \, b = \sum y_i$$

$$\sum x_i \, a + \sum x_i^2 \, b = \sum x_i y_i$$

(talk about how to solve this for a and b)

- Now we have two simultaneous equations in two unknowns:

$$Na + \sum x_i \, b = \sum y_i$$

$$\sum x_i \, a + \sum x_i^2 \, b = \sum x_i y_i$$

(talk about how to solve this for a and b)

- The best-fit values of a and b are:

$$a = \frac{\sum x_i^2}{N \sum x_i^2 - \sum x_i \sum x_i} \sum y_i + \frac{-\sum x_i}{N \sum x_i^2 - \sum x_i \sum x_i} \sum x_i y_i$$

$$b = \frac{-\sum x_i}{N \sum x_i^2 - \sum x_i \sum x_i} \sum y_i + \frac{N}{N \sum x_i^2 - \sum x_i \sum x_i} \sum x_i y_i$$

Exercise

- Code up these two relations as functions and then use them to find the best-fit line for the following three data points:

- x = [-10.,0.,10.], y = [-20.5, 2.02, 18.9]

- Make a plot of the data (with error bars=2) and the corresponding best-fit line.

The Errors on the Fitted Parameters

- As scientists, determining a and b isn't enough. We need an estimate of the errors on each parameter.

- Use error propagation to determine σ_a^2 and σ_b^2:

$$\sigma_a^2 = \sum_{i=1}^{N} \sigma^2 \left(\frac{da}{dy_i}\right)^2$$

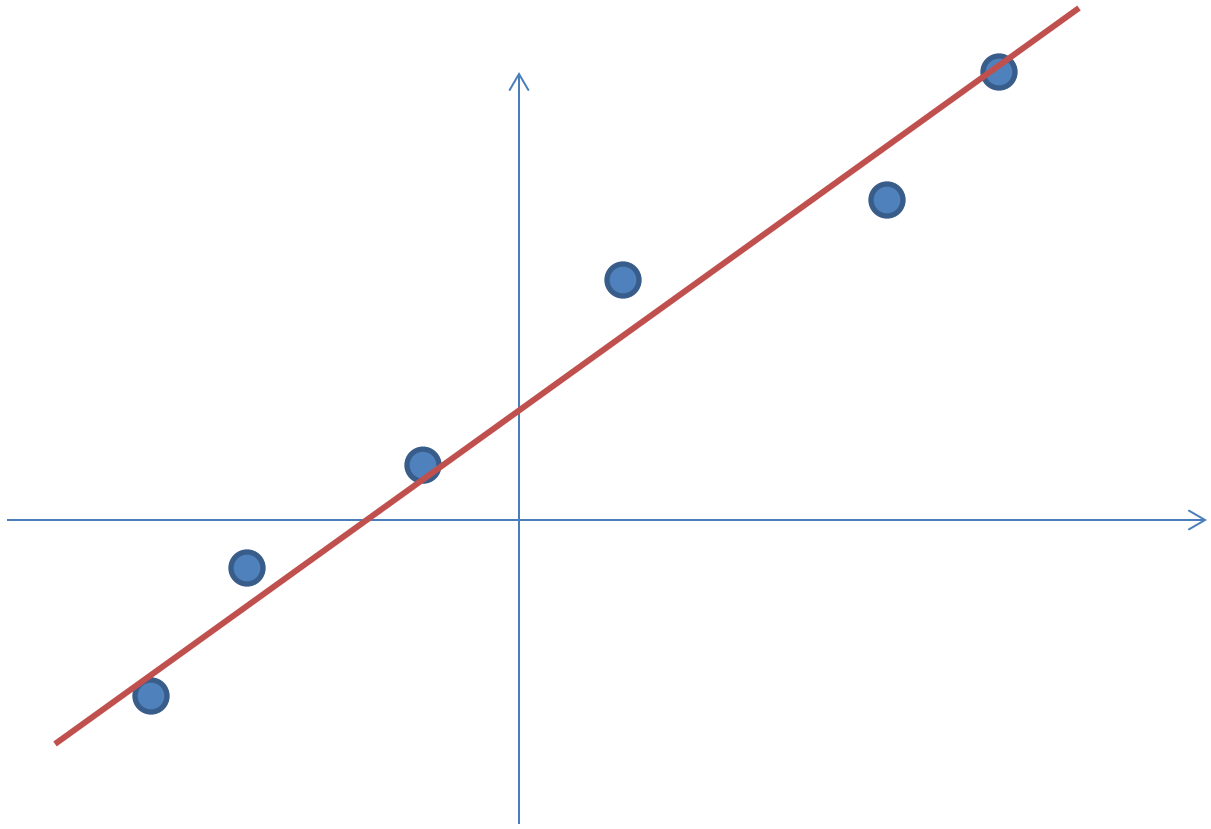
$$\sigma_b^2 = \sum_{i=1}^{N} \sigma^2 \left(\frac{db}{dy_i}\right)^2$$

The result:

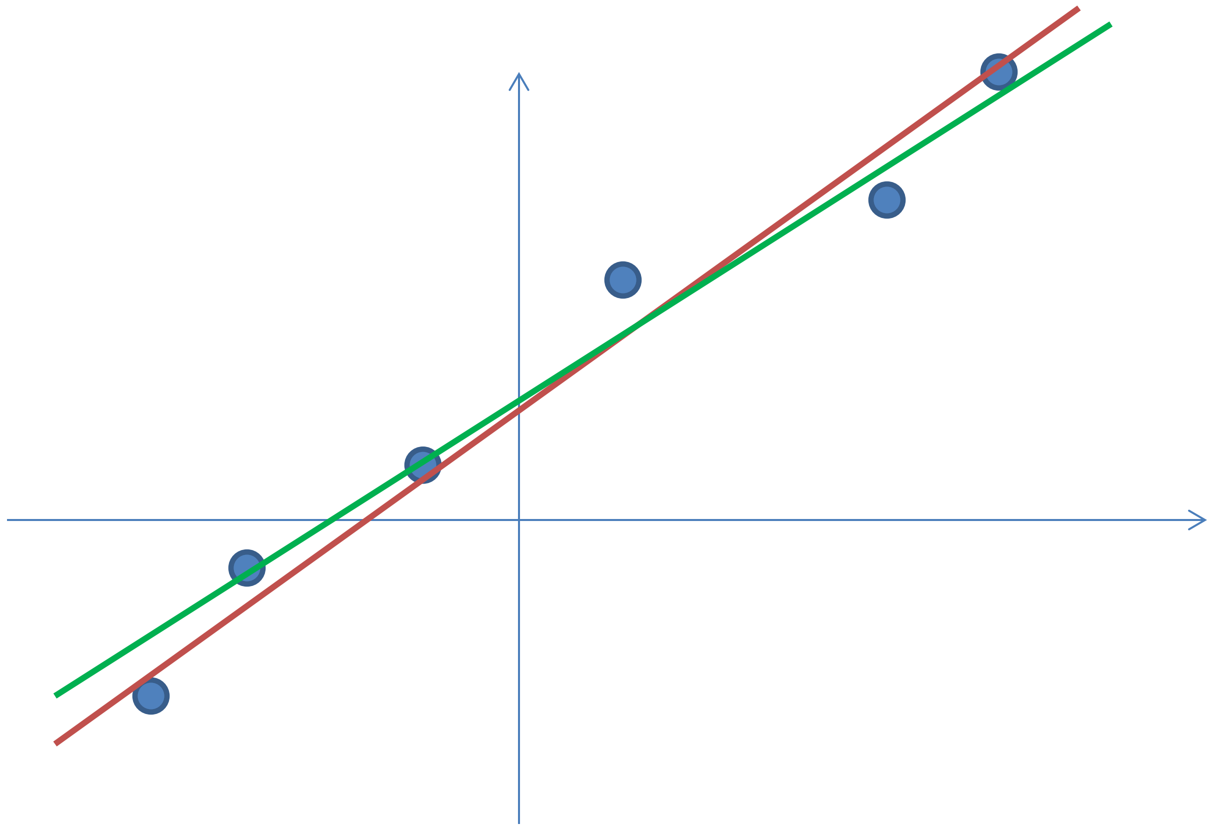
$$\sigma_a^2 = \sum_{i=1}^{N} \sigma^2 \left(\frac{da}{dy_i}\right)^2 = \frac{\sum x_i^2}{N \sum x_i^2 - \sum x_i \sum x_i} \sigma^2$$

$$\sigma_b^2 = \sum_{i=1}^{N} \sigma^2 \left(\frac{db}{dy_i}\right)^2 = \frac{N}{N \sum x_i^2 - \sum x_i \sum x_i} \sigma^2$$

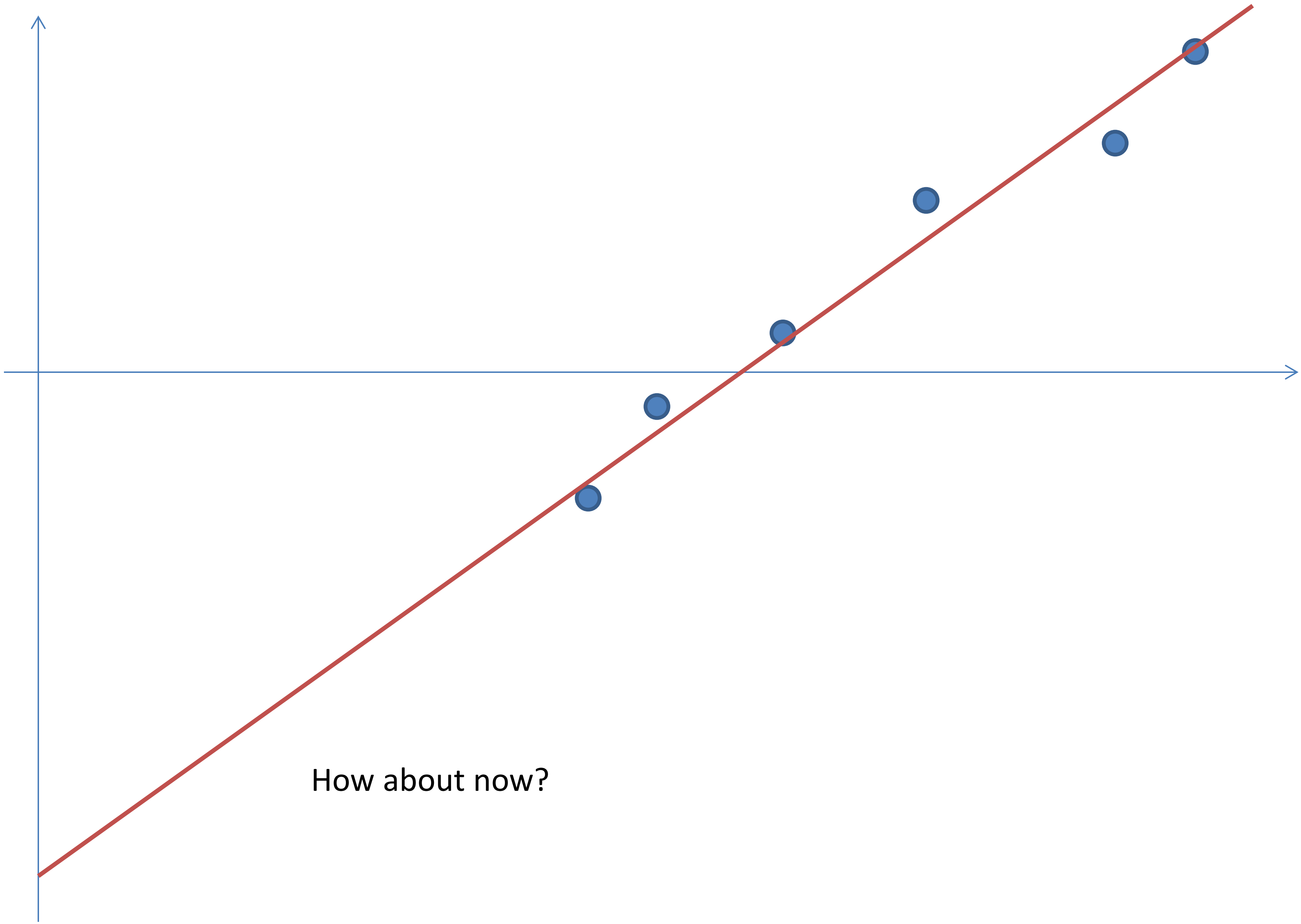
Take a moment to code these up as functions. As you're doing it, think: are these errors independent?



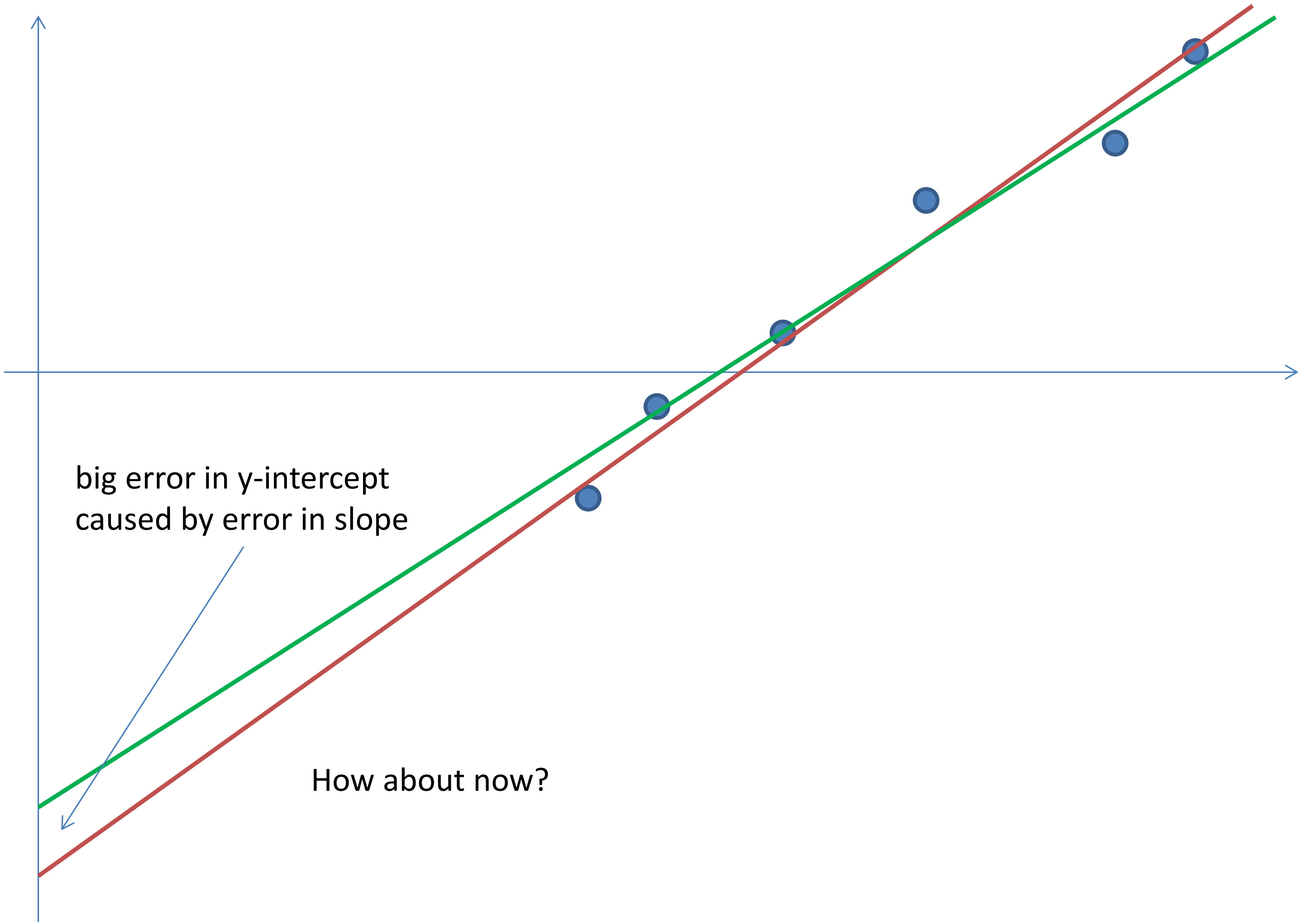
Imagine an error in the slope of this line. How would that affect the y-intercept of the line?



Imagine an error in the slope of this line. How would that affect the y-intercept of the line?



How about now?


big error in y-intercept
caused by error in slope
How about now?

Parameter-Parameter Covariance

- These "linked" errors are captured in the "covariance" of the parameters:

$$\sigma_{ab}^2 = \sum_{i=1}^{N} \sigma^2 \left(\frac{da}{dy_i}\right)\left(\frac{db}{dy_i}\right) = \frac{-\sum x_i}{N \sum x_i^2 - \sum x_i \sum x_i} \sigma^2$$

- Take a moment to code the covariance up as a function.

- The bigger the covariance, the more linked the parameters.

- Perfectly independent parameters will have $\sigma_{ab}^2 = 0$.

- And finally, to encapsulate it all we have the "covariance matrix"

$$Cov(a, b) = \begin{pmatrix} \sigma_a^2 & \sigma_{ab}^2 \\ \sigma_{ab}^2 & \sigma_b^2 \end{pmatrix}$$

Exercise

- This is a very important simulation that will give you some insight about covariance.

1. Generate fake data set #1 with 20 points:
    1. slope = 4.
    2. y-intercept = -10
    3. errors = 1.
    4. x1 = np.linspace(-20.,20.,npts)

2. Generate fake data set #2 with 20 points:
    1. slope = 3.
    2. y-intercept = -20
    3. errors = 1
    4. x2 = np.linspace(30.,60.,npts)

3. Fit both synthetic data sets to a line and find the best-fit slope and y-intercept for your data. Also calculate the errors in both the slope and the y-intercept. Calculate the covariance of the two parameters as well.

4. Now repeat steps 1-3 1000 times. Plot the resulting best-fit y-intercepts versus the best-fit slopes. What do you see in the two cases? How does this compare to your parameter errors and the covariance?

If Time … A Simpler Approach

- We have N-data points (20 in the last exercise) and we are looking for the best-fit line.

- Consider:

$$y_1 = a + bx_1$$
$$y_2 = a + bx_2$$

Could you solve these for a and b?

If Time … A Simpler Approach

- We have N-data points (20 in the last exercise) and we are looking for the best-fit line.

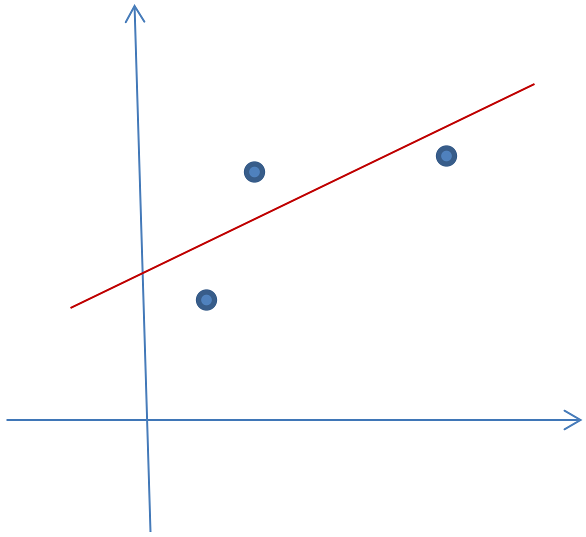
- Consider:

$$y_1 = a + bx_1$$
$$y_2 = a + bx_2$$

- Could you solve these for a and b? How about:

$$y_1 = a + bx_1$$
$$y_2 = a + bx_2$$
$$y_3 = a + bx_3$$

- We already know the problem here. There is no line guaranteed to pass through a general set of 3 points. Instead we search for the "least squares" solution.



Let’s generalize

(temporarily ignoring σ):

$$
\begin{aligned}
y_1 &= a + bx_1 \\
y_2 &= a + bx_2 \\
y_3 &= a + bx_3 \\
&\vdots \\
y_N &= a + bx_N
\end{aligned}
\quad \longrightarrow \quad
\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{pmatrix}
=
\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix}
\begin{pmatrix} a \\ b \end{pmatrix}
$$

or even more simply:

$$\vec{y} = M\vec{p}$$

Let's generalize:

$$
\begin{aligned}
y_1 &= a + bx_1 \\
y_2 &= a + bx_2 \\
y_3 &= a + bx_3 \\
&\vdots \\
y_N &= a + bx_N
\end{aligned}
\quad \longrightarrow \quad
\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}
$$

or even more simply:

$$\vec{y} = M\vec{p}$$

With Linear Algebra

- Our system of N-equations is:

$$\vec{y} = M\vec{p}$$

- Multiply both sides by transpose of M:

$$M^T\vec{y} = M^T M\,\vec{p}$$

- The matrix $M^T M$ is square so we can possibly take its inverse, $(M^T M)^{-1}$. Multiply both sides of eq (2) above by the inverse to get $\vec{p}$

$$\vec{p} = (M^T M)^{-1} M^T \vec{y}$$

- The Gauss-Markov Theorem states that $\vec{p}$ is the least squares estimate of a and b.

- The covariance matrix is simply $(M^T M)^{-1}$

Exercise

- Try the linear algebra approach on this example:
- x = np.array([0.,1.,2.,3.,4.])
- y = np.array([0.05,0.88,2.06,2.95,4.13])

- Construct M by hand.
- The transpose of M, M^T is just M.transpose()
- Build alpha = $(M^T M)$
- Find the inverse of alpha with:

What could possibly go wrong?

- The last slides showed that with a little linear algebra we can make the equations for the best-fit very simple. But what about:
  - different errors on each data point?
  - repeated data points that add no new info?
  - more complicated model functions?

- We'll tackle all of these next class.