Introduction to Data Analysis
PHYS 132 Lab

University of California, Riverside

17 April 2014

Put everything in a single notebook. Clearly indicate which cells correspond to which exercises. The notebook should run correctly without error and produce the expected output when opened and run with `Evaluation > Evaluate Notebook`.

1 Plot Functions

1. Make the following modifications to `makePlot` from last week:
   - Make the data points black and the function a black line. Color is nice on the screen, but for printing, grayscale is better and black reads clearly.
   - Give the plot axes labels. The text for the labels should be a new argument to `makePlot`. You may make this an optional argument, but if you do, be sure your function works when it is not given.
2. Make a new function `makeFramePlot` that does what `makePlot` does above, but uses `Frame` and `FrameLabel` (hide the axes). When displaying fits, a frame with ticks on all sides is generally preferred. Use any color scheme you like for this, and take some time to play with the options available for `Plot` and `ListPlot`.
3. Make a new function `plotFit` that we will use below to plot our fits. (This should be a new standalone function that doesn't depend on the above functions.) Use your best judgment for this one: you may want to reuse code you wrote earlier. The minimum requirements:
   - Takes at least two required arguments: `data` and `function`.
   - Takes at least two optional arguments: `plotFunction` and `listPlotFunction`.
   - Plots `data` and `function` on the same plot using `plotFunction` and `listPlotFunction`.
   - Uses a frame, gives the plot frame labels, and plots the function over the full domain of the data.

2 Linear Model

In this exercise, you will import data from a text file and fit that data to a simple linear model,

$$y(x) = mx + b. \tag{1}$$

1. Use `SetDirectory[NotebookDirectory[]]` to set the working directory for your notebook. Do this at the very top of your notebook. Also add `Needs["ErrorBarPlots`"]`.
2. The data for this exercise is in `linear.tsv`.
3. Import the data with `Import`. Store the imported data in the variable `rawData`. Be careful to deal with the headings on the first line (you can use `HeaderLines`). The data is only in the first two columns: you can select it with `rawData[[All, {1, 2}]]`. Test that you can plot your data with `ListPlot`.
4. Fit the data with `Fit`. This function returns the fit as an expression.

5. Fit the data with `FindFit`. This function returns the values of the fitting parameters as a list of replacement rules. Write a single line expression using `FindFit` that can be passed as the function to `plotFit` and test that it works.
6. Write a function `plotLinearFit` that fits the data using `LinearModelFit` and plots the result with `plotFit`. You can do this inside a two or three line module: Just get the `"Function"` property from the model object.
7. Modify `plotLinearFit` to check if the data includes an uncertainty in y (assumed to be in the third column). If the uncertainty is given, it should fit using `Weights` equal to $1/\Delta y^2$. Test this works using `ErrorListPlot` for `listPlotFunction`. (Careful that you give a the correct list to `Weights`. The list should be a flat list of numbers. You can use something like `errors^(-2)`, where `errors = rawData[[All, 3]]`.)

3 Gaussian Model

In this exercise, you will import data from a text file and fit that data to a Gaussian (or normal) distribution,

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]. \tag{2}$$

1. First, define a function for the Gaussian distribution using normal mathematical input.
2. Do the same, but use the built in function `NormalDistribution`.
3. Test these agree by plotting each on the same plot (don't use `Show`, just use `Plot`).
4. The data for this exercise is in `normal.tsv`.
5. The actual data has been normalized to it's maximum value. Define a function for the normalized Gaussian. (At what value of x is $f(x)$ maximized?)
6. Fit the data using `FindFit` and plot the fit using `plotFit`.
7. Fit the data using `NonlinearModelFit`. Take into account the error and plot it with error bars using `plotFit`. You may wrap this into a function `plotGaussianFit` if you like. What is the chi-squared for the fit (see `"EstimatedVariance"`)?