STAT 305 Homework 2

Please submit an R source package file, `robust_0.1.tar.gz`. We'll grade it by running

```
install.packages(pkgs="robust_0.1.tar.gz", repos=NULL, type="source")
require("robust")
example(lad)
?lad
?area
?print.lad
?coef.lad
?predict.lad
```

and by running other tests on your code, and by reading your code.

1 Least absolute deviations regression: a lad() function

Your package will have a function `lad(x, y)` which takes as arguments vectors `x` and `y` and returns an object having the (new) class `"lad"` and consisting of a list with these components:

- `coefficients`, a named numeric vector of length 2 containing the coefficients $(\hat{\beta}_0, \hat{\beta}_1)$ of a linear model and having the names `"(Intercept)"` and `"x"`
- `fitted.values`, a vector of the $\{\hat{y}_i\}$
- `residuals`, a vector of the $\{e_i = y_i - \hat{y}_i\}$

(The name "`lad`" stands for "least absolute deviations." `lad()` will be a function somewhat like `lm()`, which uses the usual "least squared deviations" regression.)

To find $(\hat{\beta}_0, \hat{\beta}_1)$, use `optim()` with its (default) Nelder-Mead algorithm to minimize the sum of absolute deviations (SAD) of data y_i from predictions $\hat{y}_i$. That is, minimize

$$SAD(\beta_0, \beta_1) = SAD(\beta_0, \beta_1; \vec{x}, \vec{y}) = \sum_{i=1}^{n} |y_i - \beta_0 - \beta_1 x_i|$$

over (β_0, β_1). Use the least squares estimates from `lm()` as initial values.

Test your function by running

```
area = read.csv("http://www.stat.wisc.edu/~jgillett/305/2/farmLandArea.csv")
lad(x=area$land, y=area$farm)
```

Make your function code general–we may test it on several data sets, so don't write any dependence on `"farmLandArea.csv"` in `lad()`.

2 Write an R package

Create a package `robust` containing:

- the function `lad()`, described above
- the data frame `area` (which should have no persisting dependence on `"farmLandArea.csv"`)
- the method `print.lad(x, ...)`, which writes the named coefficients vector to the console
- the method `coef.lad(object, ...)`, which takes a `lad` object `object` and returns its vector of coefficients. (This will allow `abline(reg)` to work, where `reg` is a regression model object returned by `lad()`, making it easy to add a `lad` regression line to a scatterplot. See `?abline` if you want more understanding of its `reg` parameter.)
- the method `predict.lad(object, new.x, ...)`, which takes a `lad` object `object` and a vector `new.x` and returns a vector containing `lad`'s predictions at the x values in `new.x`.

Each of these items should be documented so that R's help via `?` works for them. Document the `...` parameter in each of the methods via

```
@param ...  further arguments passed to or from other methods
```

(This "`...`" parameter is necessary to pass `check()` without warnings, and to make our new methods behave correctly in object oriented programming situations we didn't discuss.)

After installing your package, when we run `example(lad)`, it must

- call your `lad()` function for the farmland dataset
- use `print()` to print $\hat{\beta}_0$ and $\hat{\beta}_1$
- make a scatterplot with the `lm()` line, the `lad()` line in a different color, and a legend
- use `predict()` to find $\hat{y}$ for `x` values corresponding to the 0, 1/4, 1/2, 3/4, and 1 quantiles of the data; add these five predictions as solid green points to the graph

Cautions: `lad()` should not produce a graph. Do not call `print.lad()`, `coef.lad()`, or `predict.lad()` directly: use `print()`, `abline()`, and `predict()` on your `lad` object.

3 Optional (for extra credit in the form of candy)

- Include test code in your package. I recommend `http://r-pkgs.had.co.nz/tests.html`.
- Write a function `base.function.most.methods()` that returns a list with components `base.function`, the base R generic function having the most methods, and `n`, the number of its methods. Hint: See `?base` for how to list base functions, `?methods` for how to list the methods of a generic function, and `?try` for how to prevent errors produced by calling `methods()` on non-generic functions from terminating your function.