00:28 Thursday 24th September, 2015
See updates and corrections at http://www.stat.cmu.edu/~cshalizi/mreg/

Lecture 7: Diagnostics and Modifications for Simple Regression

36-401, Fall 2015, Section B

22 September 2015

Contents

In the previous lectures, we have laid out what regression analysis is for, why we use statistical models to do it, the assumptions of the simple linear regression model, and estimation and prediction for the simple regression model using both the method of maximum likelihood and the method of least squares. We could, at this point, follow the traditional route in a class like this of plunging into detailed inferential calculations for the model (this is how you find a confidence interval for such-and-such, etc.). However, those calculations have little point if the assumptions for the model do not hold. Thus, we will look at model checking *first*, and in later lectures go on to the inferential theory.

It is somewhat more traditional to talk about "diagnostics" for the regression model, rather than "model checking"; which name is best depends on how formal you want to sound. Just as in medicine, a good diagnostic procedure is one which will *differentiate* between health (for us: the model assumptions hold) and illness (the assumptions are violated). Indeed, a really good diagnostic procedure will distinguish between different sorts of illness (tell us *which* assumptions are violated). More mathematically, we want to find consequences of the assumptions which will hold no matter what the parameter values might be, but would be hard to achieve if the assumptions fail. (Checking whether $\sum_i \hat{\beta}_0 + \hat{\beta}_1 x_i = \sum_i y_i$ is not a useful diagnostic, because while that will be true if the assumptions are all right, it would still be true if they were all wrong[1].) The simplest and most useful check all involve the residuals.

1 The Residuals

In previous lectures, we defined the **residual** at the i^{th} data point as the difference between the realized value of the response y_i and what the estimated model would predict:

$$e_i = y_i - \hat{m}(x_i)$$

which, for the simple linear regression model, is just

$$e_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$$

Residuals vs. the noise The residuals are very closely related to the noise variables ϵ_i, but with some important differences. If we take the basic equation for the simple linear model, $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, we can re-arrange it to read

$$\epsilon_i = Y_i - (\beta_0 + \beta_1 x_i)$$

This has the same form as the equation for the residuals, but it involves the true parameters, not their estimates. If we take that equation for the i^{th} residual and substitute in the equation for Y_i,

$$e_i = \beta_0 + \beta_1 x_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) + \epsilon_i \quad (1)$$

$$(\beta_0 - \hat{\beta}_0) + (\beta_1 - \hat{\beta}_1) x_i + \epsilon_i \quad (2)$$

The terms in parentheses on the right-hand side are hopefully small, but they're not (in general) zero.

Residuals as weighted sums of noise To understand what's going on with the residuals, it's helpful to write them as a weighted sum of the ϵ's. Going back to previous lectures,

$$\hat{\beta}_0 = \beta_0 + \frac{1}{n}\sum_{i=1}^{n}\left(1 - \overline{x}\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i \quad (3)$$

$$\hat{\beta}_1 = \beta_1 + \sum_{i=1}^{n}\frac{x_i - \overline{x}}{n s_X^2}\epsilon_i \quad (4)$$

[1] I could give you a demonstration of that equation, but that would spoil one of the problems in the current homework

Substitute these in to the equation for e_i:

$$\begin{aligned} e_i &= \epsilon_i + \frac{1}{n}\sum_{j=1}^{n}\left(1 - \overline{x}\frac{x_j - \overline{x}}{s^2_X}\right)\epsilon_j + x_i \sum_{j=1}^{n}\frac{x_j - \overline{x}}{ns^2_X}\epsilon_j && (5) \\ &= \sum_{j=1}^{n}\left(\delta_{ij} + \frac{1}{n} + (x_i - \overline{x})\frac{x_j - \overline{x}}{ns^2_X}\right)\epsilon_j && (6) \end{aligned}$$

using the "Kronecker delta" ($\delta_{ij} = 0$ if $i \neq j$, $= 1$ if $i = j$) and some algebra. The factor in parenthesis is a weight which gets applied to each ϵ_j when summing them up — a weight which depends on the x's alone. Let's abbreviate this weight by c_{ij}.

One of the assumptions of the simple linear model is that $\mathbb{E}\left[\epsilon_i|X\right] = 0$. Since we've shown that e_i is a weighted sum of ϵ_j, it follows that

$$\mathbb{E}\left[e_i|X\right] = \sum_j c_{ij}\mathbb{E}\left[\epsilon_j|X\right] = 0$$

Since the simple linear model assumes that the ϵ's are uncorrelated and all have variance σ^2, even conditional on X,

$$\begin{aligned} \mathrm{Var}\left[e_i|X\right] &= \sum_j \mathrm{Var}\left[c_{ij}\epsilon_j|X\right] && (7) \\ &= \sum_j c^2_{ij}\mathrm{Var}\left[\epsilon_j|X\right] && (8) \\ &= \sigma^2\sum_{j=1}^{n} c^2_{ij} && (9) \end{aligned}$$

(I will not bother writing out the sum explicitly.) From here, one can go on to show

$$\mathrm{Var}\left[e_i\right] = \frac{n-2}{n}\sigma^2$$

though again I omit the details, so as not to spoil a future assignment.

If we make the Gaussian noise assumption, the ϵ_j are independent Gaussians. It thus follows that e_i also has a Gaussian distribution.

Contrast between residuals and noise terms The sum of the noise terms which produced the data is rarely zero. The *expectation value* of the sum of the noise is zero,

$$\mathbb{E}\left[\sum_{i=1}^{n}\epsilon_i\right] = \sum_{i=1}^{n}\mathbb{E}\left[\epsilon_i\right] = 0$$

but the *variance* is not:

$$\mathrm{Var}\left[\sum_{i=1}^{n}\epsilon_i\right] = \sum_{i=1}^{n}\mathrm{Var}\left[\epsilon_i\right] = n\sigma^2$$

so the sum of the noise terms can't be exactly zero all the time:

$$\sum_{i=1}^{n}\epsilon_i \neq 0$$

(Indeed, if the ϵ_i follow a Gaussian distribution, then $\sum_{i=1}^{n}\epsilon_i \sim N(0, n\sigma^2)$, and the probability that $\sum_{i=1}^{n}\epsilon_i = 0$ is zero, not one.) Similarly, while the ϵ are uncorrelated with X,

$$\mathrm{Cov}\left[X, \epsilon\right] = \mathbb{E}\left[X\epsilon\right] - \mathbb{E}\left[\epsilon\right]\mathbb{E}\left[X\right] = 0$$

the ϵ_i don't have a zero *sample* correlation with X:

$$\sum_{i=1}^{n}\epsilon_i(x_i - \overline{x}) \neq 0$$

On the other hand, such equations do hold exactly and deterministically for the residuals. In particular,

$$\sum_{i=1}^{n}e_i = 0$$

when the simple linear model is estimated by least squares, no matter what. This is because this equation is a consequence of the estimating equations and the estimating equations alone — it applies all the time, on any data set, not just with probability 1 but without exception. Similarly,

$$\sum_{i=1}^{n}(x_i - \overline{x})e_i = 0$$

also follows from the estimating equations. I will not go over the derivations, so as not to spoil the homework you are currently doing.

(If we think of $(e_1, e_2, \ldots e_n)$ as an n-dimensional vector, these two equations tell us that not every vector is possible. Only vectors which live in an $(n-2)$-dimensional subspace are allowed. This is, so to speak, the same $n-2$ as in $\mathrm{Var}\left[e_i\right]$.)

These two equations imply that even when the ϵ's are independent, the residuals are not. (After all, if we know all but one residual, the last one is completely determined.) However, the dependence is typically very slight and subtle, and it gets weaker as n grows (because each e_i is making a comparatively-small contribution to the sum that must be zero). So the residuals should show only negligible correlation (Exercise 1).

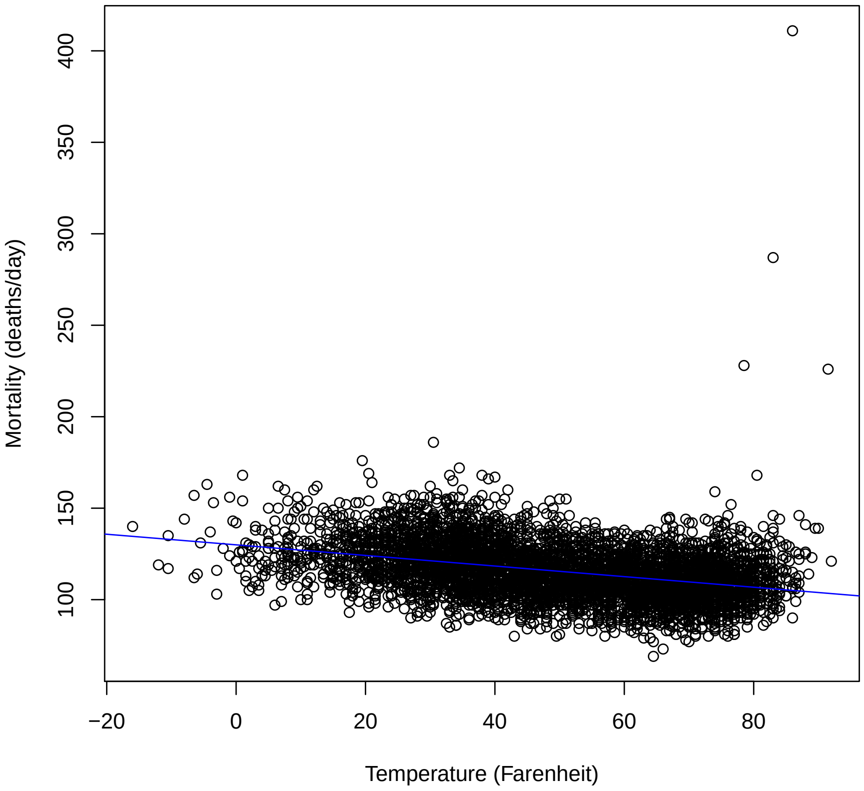
1.1 Summary on Properties of the Residuals

Let's sum up the most relevant observations from the last couple of paragraphs.

1. The residuals should have expectation zero, conditional on x, $\mathbb{E}\left[e_i|X = x\right] = 0$. (The residuals should also have an over-all sample mean of exactly zero.)

2. The residuals should show a constant variance, unchanging with x.
3. The residuals can't be completely uncorrelated with each other, but the correlation should be extremely weak, and grow negligible as $n \rightarrow \infty$.
4. If the noise is Gaussian, the residuals should also be Gaussian.

Each one of these points leads to a diagnostic, to a check on the model. These take the form of our plots, which you should always, always make for any regression you run.



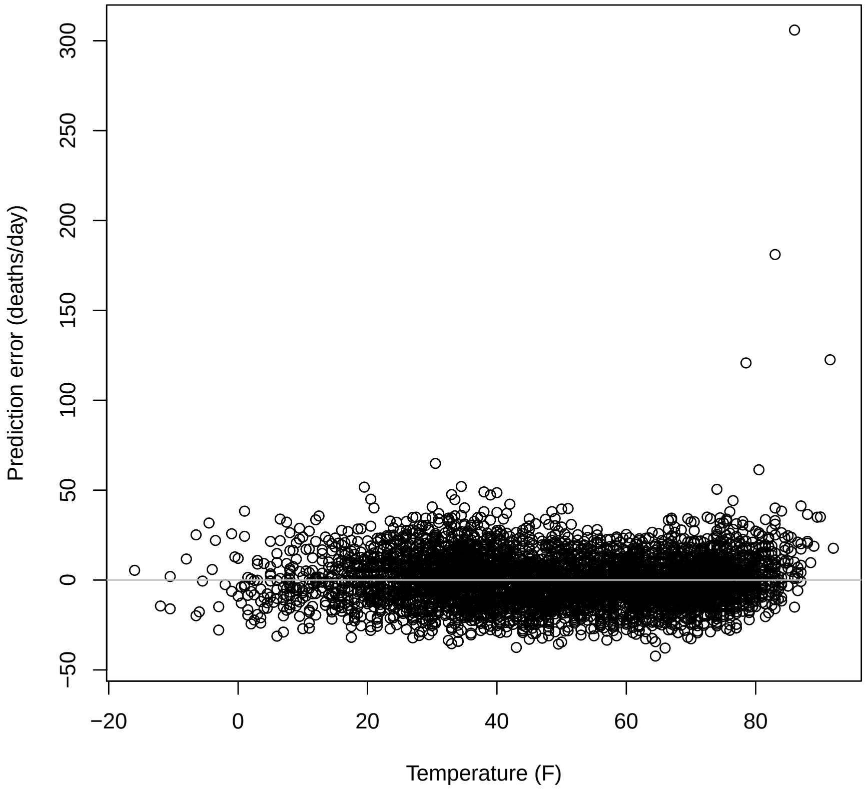
```
# Load the data set from the last homework
library(gamair)
data(chicago)

# Plot deaths each day vs. temperature
plot(death ~ tmpd, data=chicago,
     xlab="Temperature (Farenheit)",
     ylab="Mortality (deaths/day)")

# Estimate and store a linear model
death.temp.lm <- lm(death ~ tmpd, data=chicago)
abline(death.temp.lm, col="blue")
```

FIGURE 1: *Plot of the data from the last homework, used as a running example, along with the estimated linear model (in blue).*

1.2 Plot the Residuals Against the Predictor

Make a scatter-plot with the residuals on the vertical axis and the predictor variable on the horizontal axis. Because $\mathbb{E}\left[e|X=x\right]=0$, and $\mathrm{Var}\left[e|X=x\right]$ is constant, this should, ideally, look like a constant-width blur of points around a straight, flat line at height zero. Deviations from this — changing width, curvature, substantial regions of the x axis where the average residuals are either positive or negative — are all signs that the model is mis-specified. In particular, curved or stepped patterns indicate that $\mathbb{E}\left[e|X=x\right]\neq 0$, which in turn means that $\mathbb{E}\left[\epsilon|X=x\right]\neq 0$, which means that the simple-linear part of the simple linear regression model is wrong[2]. One needs either more predictor variables (getting us into multiple regression), or a different functional form for the regression (getting us into nonlinear regression), or both.

[2] See §5.



```
# Always plot residuals vs. predictor variable
plot(chicago$tmpd, residuals(death.temp.lm),
     xlab="Temperature (F)",
     ylab="Prediction error (deaths/day)")
abline(h=0, col="grey")
```

FIGURE 2: *Residuals (vertical axis) vs. the predictor variable of temperature (horizontal axis).*

Plotting Against Another Variable If you have other potential predictor variables, you should be able to plot the residual against them, and also see a flat line around zero. If not, that's an indication that the other variable does in fact help predict the response, and so you should probably incorporate that in your model. (Part of the residuals from your simple linear model are really the contributions of that other predictor, which you were treating as noise out of ignorance[3].) In particular, if you make such a plot and you see the points in it fall around a straight line, that's an excellent sign that you need a multiple linear regression model.

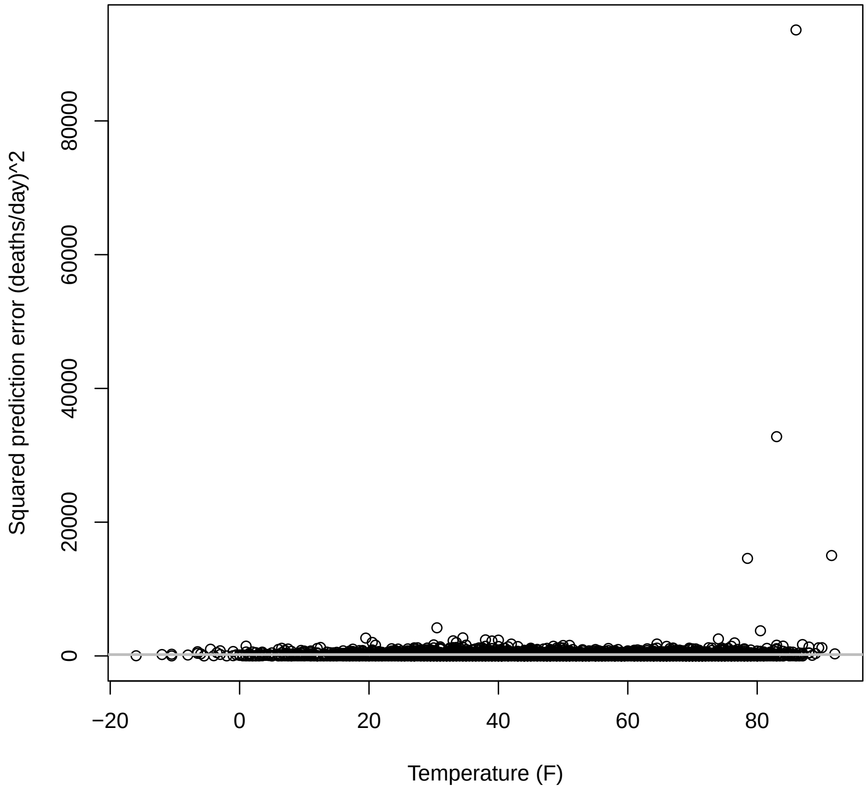
(Exercise: make a plot of the residuals from this model against one of the pollution variables from the data set. Does it look like noise?)

1.3 Plot the Magnitude of the Residuals Against the Predictor

Because $\mathbb{E}\left[e|X=x\right]=0$, $\mathrm{Var}\left[e|X=x\right]=\mathbb{E}\left[e^2|X=x\right]$. (Why?) This means we can check whether the variance of the residuals is constant by plotting the *squared* residuals against the predictor variable. This should give a scatter of points around a flat line, whose height should be around the in-sample MSE. Regions of the x axis where the residuals are persistently above or below this level are evidence of a problem with the simple linear regression model. This could be due to non-constant noise variance ("heteroskedasticity", in the jargon), or due to getting the functional form of the regression wrong. One can often get a clue as to what is driving the problem by looking to see whether the regions where the squared residuals are too big are also regions where the residuals are persistently above or below zero.

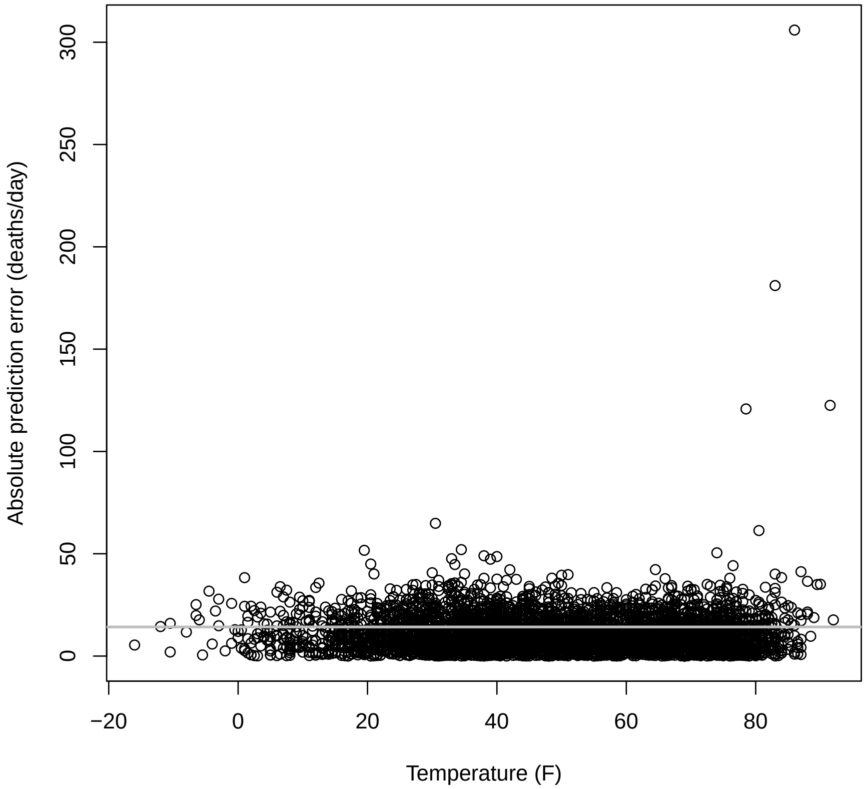
Sometimes, particularly when the model is not doing so well, squaring the residuals leads to a visually uninformative plot, because big residuals lead to really, really big squared residuals, and it's hard to make out any detail. A common fix is to then plot the absolute value of the residuals, with the reference horizontal line being at the square root of the mean squared error.

[3] Whether there really is such a thing as "noise", or only predictor variables we have neglected, is a deep and subtle question we don't have to answer here.



```
# Always plot residuals^2 vs.
# predictor variable
plot(chicago$tmpd, residuals(death.temp.lm)^2,
     xlab="Temperature (F)",
     ylab="Squared prediction error (deaths/day)^2")
abline(h=mean(residuals(death.temp.lm)^2),
       lwd=2, col="grey")
```

FIGURE 3: *Squared residuals vs. temperature.*



```
plot(chicago$tmpd, abs(residuals(death.temp.lm)),
     xlab="Temperature (F)",
     ylab="Absolute prediction error (deaths/day)")
abline(h=sqrt(mean(residuals(death.temp.lm)^2)),
       lwd=2, col="grey")
```

FIGURE 4: *Absolute residuals vs. temperature; plotting these rather than the squared residuals reduces the* visual *impact of the few huge residuals.*

1.4 Plot the Residuals Against Coordinates and Each Other

Lots of the time, our data were collected in a certain order — each data point has some coordinates, in space or in time or both. Under the simple linear regression model, these shouldn't matter[4], so you should always plot the residuals against the coordinates. (Even if you have no coordinates, you should always plot residuals against the row numbers of the data set.) Clusters of nearby observations with unusually high or low residuals are a bad sign.

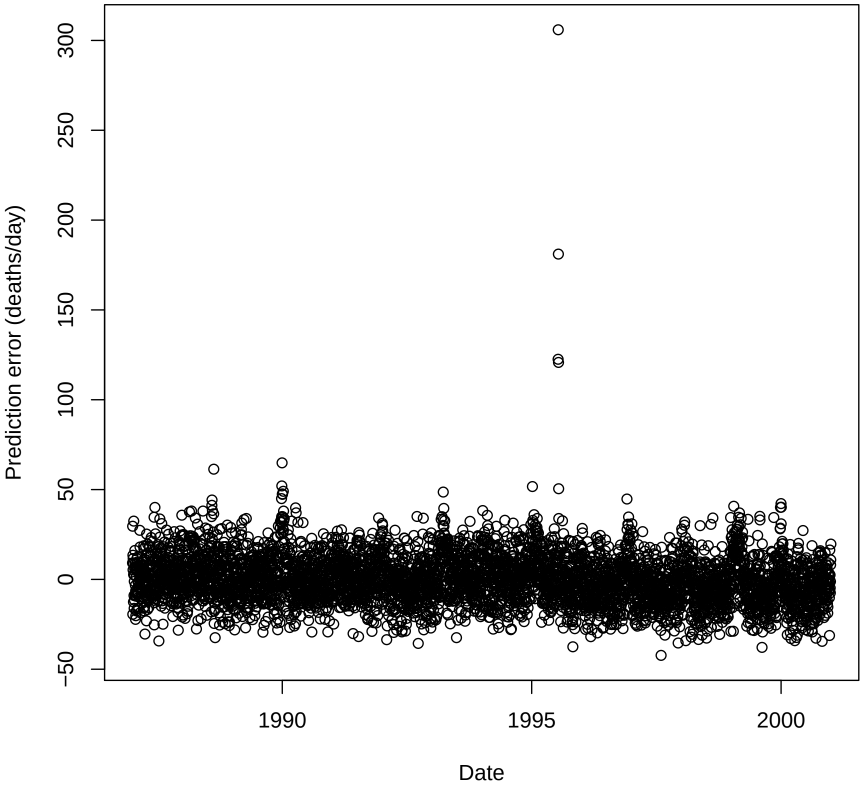
Of course, a certain amount of apparent clustering will happen naturally in any random process, so if this looks worrisome, one should really do some sort of formal test. Fortunately, there are lots of good test procedures for finding "runs" in what should be random noise. A quick hack, though, is simply to put the residuals in a totally random order and re-plot them:

```
sample(residuals(my.model))
```

will take the residuals vector of `my.model` and randomly permute it; plotting these permuted residuals against the coordinate will then give an example of how things should look. Do this a few times, and you'll get a good sense of how much apparent clumping of residuals should be produced by chance. This trick can also be used when plotting the residuals, or squared residuals, against the predictor variable, and can be formalized as what's called a "permutation test".

(Exercise: Make a few plots of the permuted, shuffled residuals against date.)

[4] Unless the predictor variable is one of the coordinates, of course.



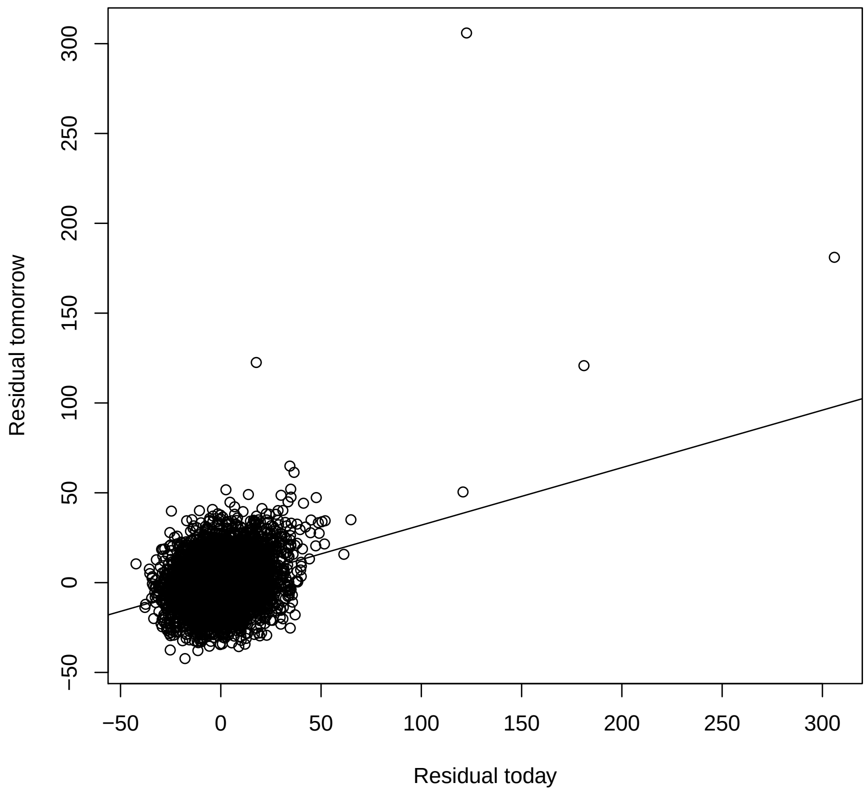
```
# Always plot residuals vs. coordinate
plot(as.Date(chicago$time,origin="1993-12-31"),
     residuals(death.temp.lm), xlab="Date",
     ylab="Prediction error (deaths/day)")
```

Figure 5: *Residuals vs. date.*

Residuals vs. Residuals A related diagnostic plot is particularly useful when the observations are taken at regular intervals along some axis (usually time but occasionally distance): make a scatter-plot with one point for each observation except the very last; the horizontal coordinate comes from that point's residual, and the vertical coordinate comes from the *next* point's residual. Ideally, this should give a blob of points with no particular structure. If the residuals are Gaussian, follow any other bell-ish distribution, it should show a circular blob. (If they were uniform, the blob should fill out a square — why?) Falling along a line or curve, or even a tilted ellipse, would be an indication of correlation between successive residuals, which in turn may be a sign that the noise is correlated[5].

Again, if you're not sure whether you're looking at a worryingly big departure from blob-hood, try permuting the residuals before re-plotting.

[5] Or, again, it could be a sign that we've got the functional form wrong, so it's our systematic error which is correlated — see §5.



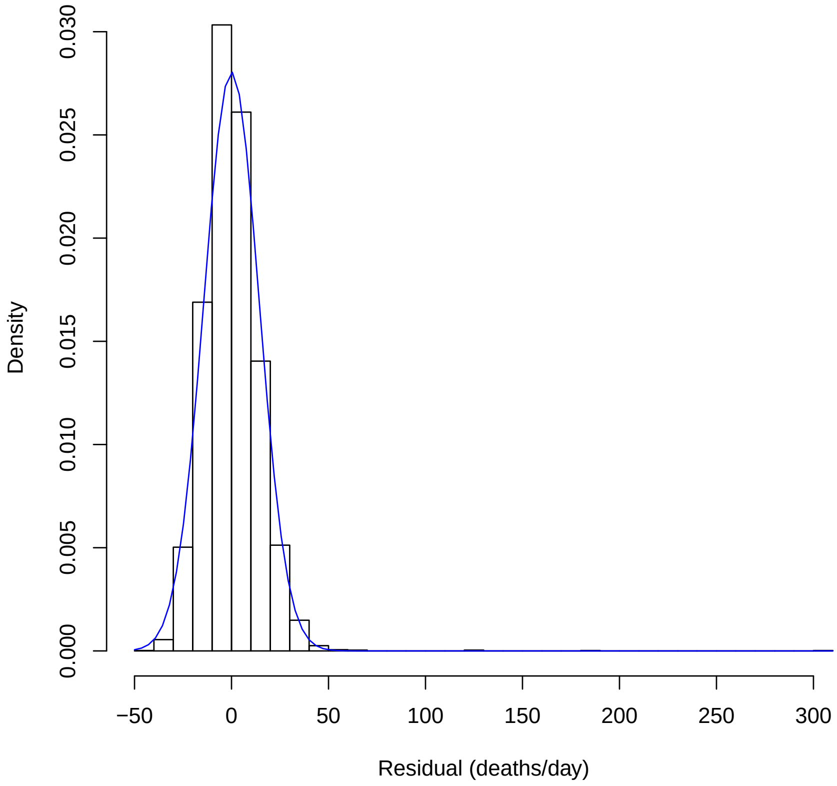
```
# Always plot successive residuals against each other
  # head() and tail() here are used to get "every day except the last"
  # and "every day except the first", respectively
  # see help(head)
plot(head(residuals(death.temp.lm),-1),
     tail(residuals(death.temp.lm),-1),
     xlab="Residual today",
     ylab="Residual tomorrow")
abline(lm(tail(residuals(death.temp.lm),-1) ~ head(residuals(death.temp.lm),-1)))
```

FIGURE 6: *Residuals for each day (except the first) plotted as a function of the residuals of the day before. The straight line shows a regression of tomorrow's residual on today's residual, which ideally should be a totally flat line.*

1.5 Plot the Distribution of the Residuals

Under the Gaussian noise assumption, the residuals should also follow a Gaussian distribution. We should therefore make plots of the distribution of the residuals, and compare that to a Gaussian.

The most basic plot of the distribution for the residuals is of course a histogram. This should be over-laid with a Gaussian probability density — but which Gaussian? The most reasonable one has mean 0 (because we know the residuals average to 0), and the same standard deviation as the residuals (because that's the MLE of the standard deviation in a Gaussian model). At that point, one can *see* whether the distribution of residuals looks like that of the best-fitting Gaussian.



```
# Always plot distribution of residuals
hist(residuals(death.temp.lm),breaks=40,
     freq=FALSE,
     xlab="Residual (deaths/day)", main="")
curve(dnorm(x,mean=0,sd=sd(residuals(death.temp.lm))),
      add=TRUE,col="blue")
```

FIGURE 7: *Histogram of the residuals, on a density scale, and the theoretical Gaussian distribution with the same mean (0) and standard deviation.*

Q-Q plots An alternative is what's called a "quantile-quantile" or "Q-Q" plot. (The textbook, old-fashionedly, calls this a "normal probability plot.) This takes a bit more thought to get used to than visually comparing density estimates, but with practice becomes most sensitive and less subjective. Here's the idea.

As you remember from intro. prob., knowing the cumulative distribution function (CDF) tells us all there is to know, mathematically, about a probability distribution; call this $F(x) = \mathbb{P}(X \leq x)$. If the distribution is continuous, the CDF has an inverse function, F^{-1}, where $F^{-1}(p)$ is the unique x such that $\mathbb{P}(X \leq x) = p$. This is called the **quantile function** — it tells us what level of x will contain a fraction p of the total probability. Since saying things like "the 0.95 quantile" is rather awkward, we usually pronounce it as "the 95^{rm} percentile", meaning the value greater than or equal to 95% of the population. If we know the quantile function, we can invert it to get the CDF, so the quantile function also completely determines the probability distribution[6].

As p varies from 0 to 1, $F^{-1}(p)$ will vary from the smallest possible value for the distribution up to the largest possible value. If our distribution is a Gaussian, with mean μ and variance σ^2, then $F^{-1}(p) = \sigma\Phi^{-1}(p) + \mu$, where Φ is the standard Gaussian CDF. (Why?) So if instead of plotting F^{-1} against p, we make a plot where $F^{-1}(p)$ goes on one axis and $\Phi^{-1}(p)$ goes on the other, as we sweep p from 0 to 1 we'll get a straight line. Conversely, if we weren't sure whether the distribution F we were interested in was Gaussian, but we did one of these quantile-quantile plots against the standard Gaussian and a got a straight line, then we'd know F was, in fact, Gaussian.

With a finite sample from a distribution (like, say, the vector of residuals), we don't really have F or F^{-1}. However, we can use the **sample quantiles** or **empirical quantiles**. Start with our observations, say $x_1, x_2, \ldots x_n$. Now put them in increasing order: to help distinguish the ordered from unordered observations, I'll use a common convention where the subscripts in parentheses show order, so

$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n-1)} \leq x_{(n)}$$

(These ordered values of the data are sometimes called the **order statistics**.) Now $x_{(i)}$ is $\geq$ a fraction i/n of the sample observations, so $\hat{F}^{-1}(i/n) = x_{(i)}$. When we make a $Q - Q$ plot against a Gaussian distribution, we therefore put $x_{(i)}$ on one axis, and $\Phi^{-1}(i/n)$ on the other, and hope to see a straight line if the distribution of x is indeed Gaussian[7]

To sum up: we put the data in order, and then plot $x_{(i)}$ against $\Phi^{-1}(i/n)$. If the data are from a Gaussian distribution, these points should fall along a straight line. Small wiggles around the line are to be anticipated from chance;

[6] In R, the CDF and quantile functions have names beginning with `p` and `q` — `pnorm` and `qnorm` for the Gaussian, `pexp` and `qexp` for the exponential, etc.

[7] You might worry about what happens when $i = n$, since $\Phi^{-1}(1) = \infty$. The answer is that I have slightly over-simplified what goes on that axis. When you call `qqnorm` in R, it actually goes through a calculation to find $\mathbb{E}\left[X_{(i)}\right]$ for $i \in 1:n$ under a Gaussian distribution, and plots *that* against the observed value $x_{(i)}$. As $n \to \infty$, $\mathbb{E}\left[X_{(i)}\right] \to \Phi^{-1}(i/n)$, but the difference can be important for small n, and/or i/n close to 0 or 1.

systematic deviations from a line indicate systematic departures from a Gaussian distribution.

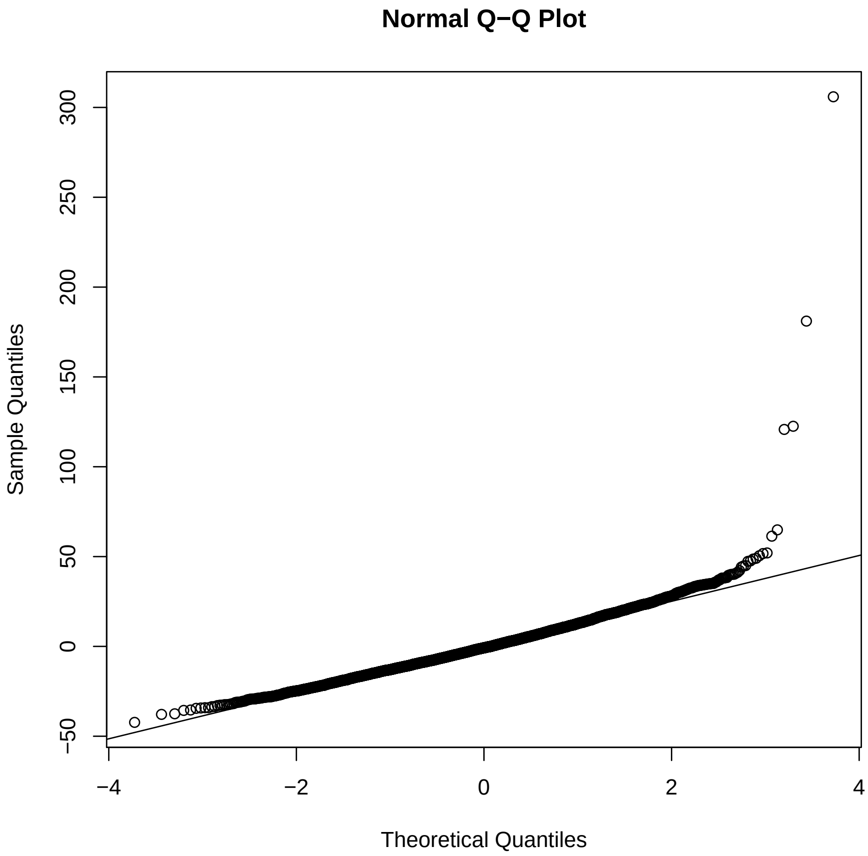
Q-Q plots for other distributions One can make a Q-Q plot for data against any other distribution one likes, provided one knows the reference distribution's quantile function; R just provides code for the Gaussian case, however.

Q-Q plots for two data distributions With two data sets, say $x_1, \ldots x_n$ and y_1, y_m, one can compare their sample quantiles. The easiest way is when $n = m$, since then one just plots $x_{(i)}$ against $y_{(i)}$. (If $n \neq m$, one might pick a grid of p values, work out $\hat{F}_x^{-1}(p)$ and $\hat{F}_y^{-1}(p)$ by interpolation, and plot those against each other.) This should show points around the $x = y$ line when x and y are both drawn from the same distribution.

P-P plots An alternative to $Q-Q$ plots are $P-P$ plots, where both axes run from 0 to 1. Call the horizontal coordinate p_1 and the vertical coordinate p_2; what we plot is $p_2 = \hat{F}(F^{-1}(p_1))$. $F^{-1}(p_1)$ is the quantile which ought, under the distribution F, to be $\geq$ a fraction p_1 of the population; $\hat{F}(F^{-1}(p_1))$ shows the actual fraction of the sample which it exceeds. R doesn't have a built-in function to make this — can you write one?

Formal tests Comparing the histogram to a theoretical density can be formalized with a χ^2 test[8]. Checking whether the $Q - Q$ plot follows a straight line can be formalized in the Kolmogorov-Smirnov test. In both cases, since we're really estimating the a parameter of the reference distribution (the standard deviation), we need to take some care to account that in a hypothesis test. (A tweak to the K-S test which does this, when testing for Gaussianity, is the "Lilliefors test", which you can find in the package `nortest`.)

[8] This is why the visual comparison is informally called "χ by eye".



```
# An alternative: plot vs. theoretical Gaussian distribution
qqnorm(residuals(death.temp.lm))
qqline(residuals(death.temp.lm))
```

Figure 8: *QQ plot of the residuals, using the standard Gaussian distribution as the reference distribution.*

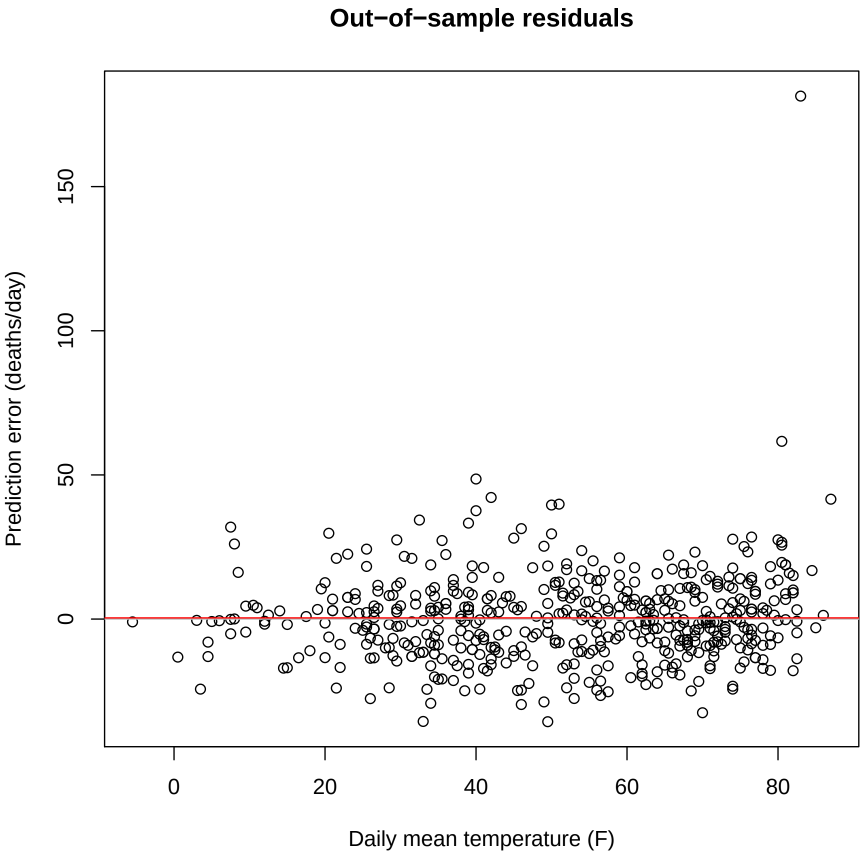
```r
# Always look at whether the model can extrapolate to
# new data
# Basic check: randomly divide into two parts, here say 90% of the data
# vs. 10%
# Use the "training set" to estimate the model
training.rows <- sample(1:nrow(chicago), size=round(nrow(chicago)*0.9),
                        replace=FALSE)
training.set <- chicago[training.rows,]
# We'll use the "testing set" to see how well it does
testing.set <- chicago[-training.rows,]
# Estimate the model on the training set only
training.lm <- lm(death ~ tmpd, data=training.set)
# Make predictions on the testing set
  # The model didn't get to see these points while it was being
  # estimated, so this really checks (or tests) whether it can
  # predict
testing.preds <- predict(training.lm, newdata=testing.set)
# Unfortunately residuals() doesn't know about the new data set
# so calculate the residuals by hand
testing.residuals <- testing.set$death-testing.preds
```

FIGURE 9: *Code setting up a* random *division of the data into training and testing sets, and looking at how well the model does on points in the testing set (which it didn't get to see during estimation).*

1.6 Generalization

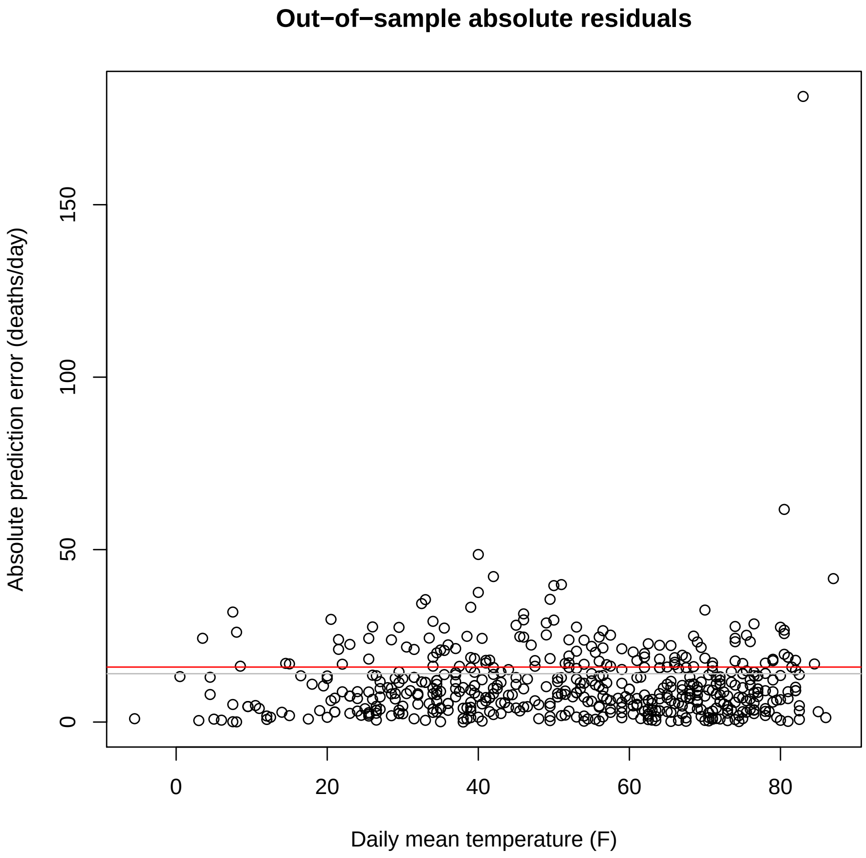
If the model assumptions are correct, it should be able to work about equally well on new data from the same source. Because the parameters were adjusted to fit the data we used to estimate the model, we should expect the prediction errors on new data to be slightly larger in magnitude, but they shouldn't be biased or otherwise show patterns. An important basic check on the model is therefore to divide the data into two parts, estimate the model on one part, the **training set**, and then examine the predictions and the residuals on the rest of the data, the **testing set**.

We can either make the division into training and testing sets by random sampling, or systematically. A random division ensures that the testing set has (almost) the same distribution as the training set. The averaged squared error on the testing set is therefore an unbiased estimate of the true mean squared error on new data. We will topic later in the course, under the heading of "cross-validation", since it is one of the most useful ways of selecting among competing models.



```
# Plot our residuals against the predictor variable
plot(testing.set$tmpd, testing.residuals,
     xlab="Daily mean temperature (F)",
     ylab="Prediction error (deaths/day)",
     main="Out-of-sample residuals")
abline(h=0,col="grey")
abline(h=mean(testing.residuals),col="red")
```

FIGURE 10: *Plot of residuals vs. temperature for the testing set. Remember that the data points here were* not *available to the model during estimation. The grey line marks the average we'd see on the training set (zero), while the red line shows the average on the testing set.*



```
# Plot absolute residuals vs. predictor variable
plot(testing.set$tmpd, abs(testing.residuals),
     xlab="Daily mean temperature (F)",
     ylab="Absolute prediction error (deaths/day)",
     main="Out-of-sample absolute residuals")
abline(h=sqrt(mean(residuals(training.lm)^2)),col="grey")
abline(h=sqrt(mean(testing.residuals^2)),col="red")
```

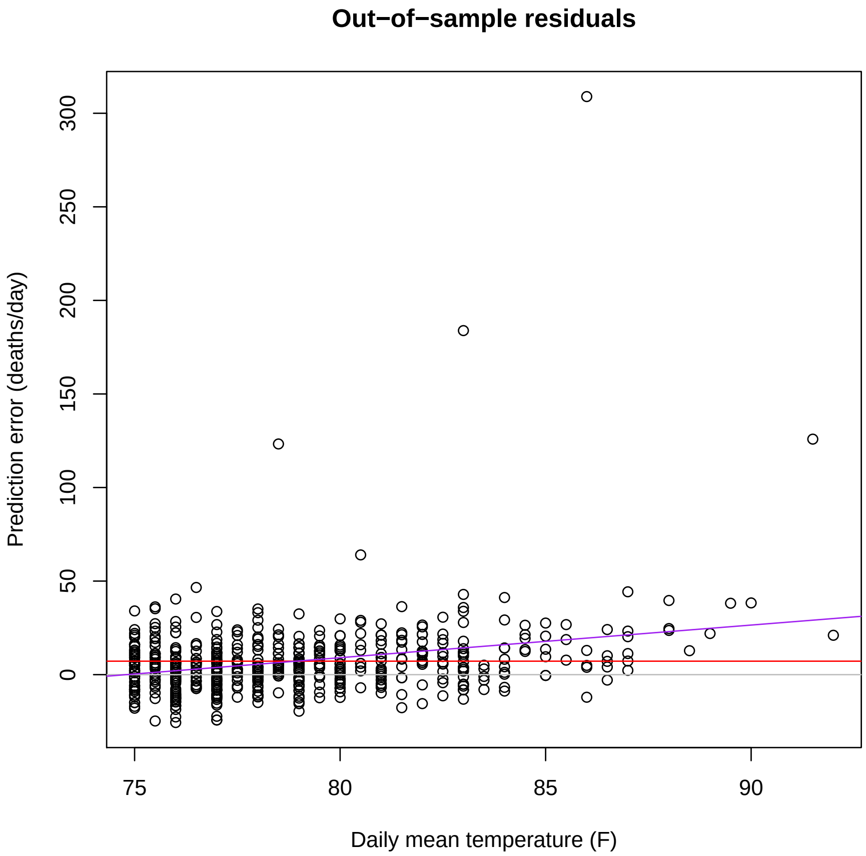
FIGURE 11: *As in Figure 10, but looking at the squared residuals.*

```
# Find the low-temperature days
lowtemp.rows <- which(chicago$tmpd < 75)  # About 90% of the data
# Divide into low- and high- temperature data sets
lowtemp.set <- chicago[lowtemp.rows,]
hightemp.set <- chicago[-lowtemp.rows,]
# Estimate the model on the colder days only
lowtemp.lm <- lm(death ~ tmpd, data=lowtemp.set)
   # For you: how much do the parameters change, as compared to
   # using all the data?
# Now predict on the high-temperature days
  # Again, these are new data points, but now systematically
  # different (because of their temperature) from the
  # data used to estimate
hightemp.preds <- predict(lowtemp.lm, newdata=hightemp.set)
# Calculate our own residuals
hightemp.residuals <- hightemp.set$death-hightemp.preds
```

FIGURE 12: *Setting up a division of the data into a low-temperature training set and a high-temperature testing set.*

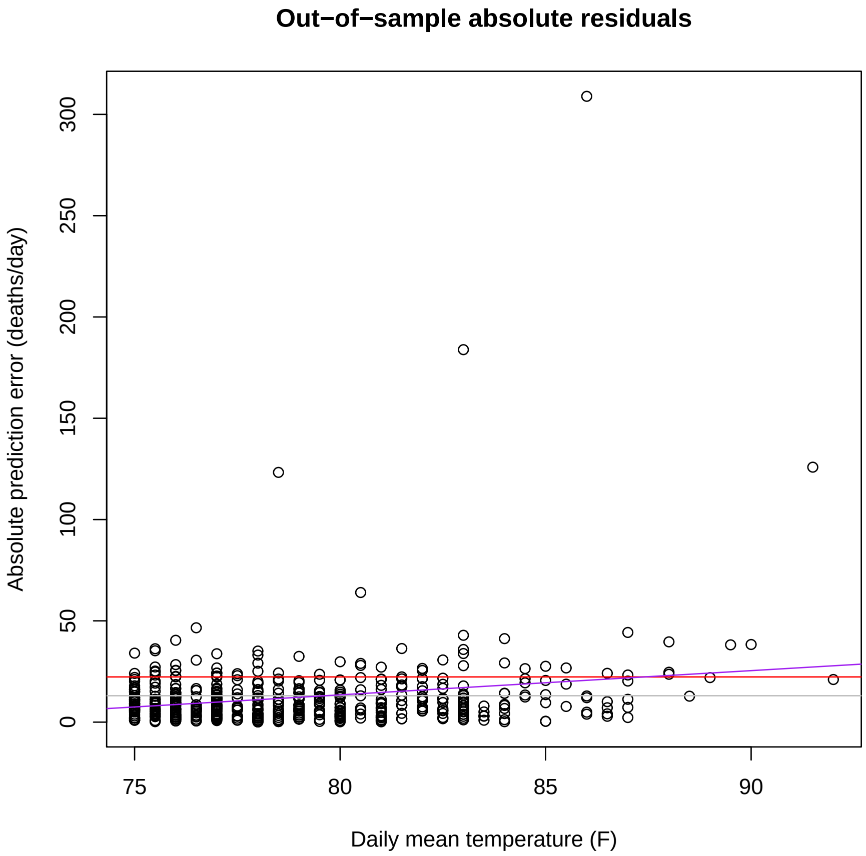
Extrapolative Generalization An alternative to random division, which can be even more useful for model checking, is to systematically make the testing set into a part of the data where the over-all fit of the model seems dubious based on other diagnostics. With our running examples, for instance, the model seems to do decently at lower temperatures, but starts to look iffy at high temperatures. We might, then, fit the model to low temperatures, and see whether it can extrapolate to high temperatures, or whether it seems to make systematic errors there. (We could also estimate the model on the high-temperature portion of the data, and see how well it extrapolates to the lower temperatures, or estimate on the middle range and see about both extremes, etc., etc.)

Generalize All the Things! *All* of the diagnostic plots discussed earlier can be combined with the trick of estimating the model on one part of the data, and then seeing whether it generalizes to the testing set. This is slightly more complicated than doing everything on the full data, but arguably has more power to detect problems with the model.



```
# Plot residuals vs. temperature
plot(hightemp.set$tmpd, hightemp.residuals,
     xlab="Daily mean temperature (F)",
     ylab="Prediction error (deaths/day)",
     main="Out-of-sample residuals")
# Flat line at 0 (ideal, if the model is right)
abline(h=0,col="grey")
# Flat line at the mean of the new residuals
abline(h=mean(hightemp.residuals),col="red")
# Regressing the new residuals on temperature does not look good...
abline(lm(hightemp.residuals ~ hightemp.set$tmpd),col="purple")
```

FIGURE 13: *Residuals vs. temperature, where the testing set here consists of days with temperature* ≥ 75 *degrees Farenheit, and the training set only those* < 75 *degrees. The grey line indicates the average residual we'd see on the training data (zero); the red line the average residual on the testing data; the purple a regression of residual on temperature, which ideally should have had slope zero.*



```
# Similar plots for the absolute residuals
plot(hightemp.set$tmpd, abs(hightemp.residuals),
     xlab="Daily mean temperature (F)",
     ylab="Absolute prediction error (deaths/day)",
     main="Out-of-sample absolute residuals")
abline(h=sqrt(mean(residuals(lowtemp.lm)^2)),col="grey")
abline(h=sqrt(mean(hightemp.residuals^2)),col="red")
abline(lm(abs(hightemp.residuals) ~ hightemp.set$tmpd),col="purple")
```

FIGURE 14: *As in Figure 13, but for absolute residuals.*

2 Nonlinear Functions of X

When we plot the residuals against the predictor variable, we may see a curved or stepped pattern. This strongly suggests that the relationship between Y and X is not linear. At this point, it is often useful to fall back to, in fact, plotting the y_i against the x_i, and try to guess at the functional form of the curve.

2.1 Transformations

The easy case is when $Y = \beta_0 + \beta_1 f(x) + \epsilon$ for some fixed, easily-computed function f like $\sqrt{x}$, x^2, $\log x$, $\sin x$, etc. We then calculate $f(x_i)$, and run a simple linear regression of y_i on $f(x_i)$. The interpretation of the coefficients hardly changes, except that we need to replace X everywhere with $f(X)$ — $\beta_0 = \mathbb{E}\left[Y|f(X) = 0\right]$, β_1 is the difference $\mathbb{E}\left[Y|f(X) = f_0\right]$ and $\mathbb{E}\left[Y|f(X) = f_0 - 1\right]$, etc. This is called "transforming the predictor", and it only works this simply when the transformation itself doesn't have to be estimated, but can just be guessed at. Ideally, in fact, we derive the transformation from some physical (chemical, biological, psychological, sociological, economic, ...) theory. For instance, there are good reasons in physiology and psychology to say that an organism's behavioral response to a stimulus should vary with the logarithm of the stimulus's physical intensity[9]. A good check on such a transformation is to plot the y_i against the $f(x_i)$, and see that the data now fall on a straight line.

2.2 Nonlinear Least Squares

If the transformation does have to be estimated, but the functional form is known[10], then the method of least squares (or maximum likelihood) still applies. Taking the derivative of the mean squared error (or the log likelihood) with respect to the parameters and setting them equal to zero gives a set of normal or estimating equations. Usually, however, these equations are nonlinear, and don't have a closed-form solution. Finding the solution is thus called "solving a nonlinear least squares (NLS) problem". When we have theoretical reasons to use some thoroughly nonlinear model, say $Y = \beta_0 x^{\beta_1} + \epsilon$, we can still estimate it using NLS in this way. Aside from needing to solve the estimating equations numerically, there are some special considerations to NLS, which we'll either cover later in this class (time permitting) or in 402[11].

[9] This is also (roughly) true, at least in people, of the perceived intensity of the stimulus. Full sunlight carries $\approx 1000\mathrm{W/m}^2$ of power, while the light of a full moon is $\approx 0.025\mathrm{W/m}^2$, yet moonlight doesn't seem forty thousand times dimmer. The logarithmic relationship between perceived and physical intensity is sometimes called "Fechner's law" in psychophysics.

[10] For instance, if $Y = \beta_0 + \beta_1 \log{(x + \beta_2)} + \epsilon$, for unknown k.

[11] The relevant R commands are `optim`, which is a general-purpose minimization function, and `nls`, which is, unsurprisingly, specialized to nonlinear least squares.

2.3 Smoothing

An alternative to using a parametrized nonlinear model is to try to let the data tell us what the appropriate curve is — to take a "non-parametric" approach. The most basic sort of curve-fitting would just take a little interval around any point x, say from $x - h$ to $x + h$, and average all the y_i where x_i was in the interval:

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} y_i I_{[x-h,x+h]}(x_i)}{\sum_{j=1}^{n} I_{[x-h,x+h]}(x_j)}$$

(Here $I_{[a,b]]}(x)$ is the **indicator function** for the interval $[a, b]$, i.e., 1 if $a \leq x \leq b$, and 0 otherwise.)

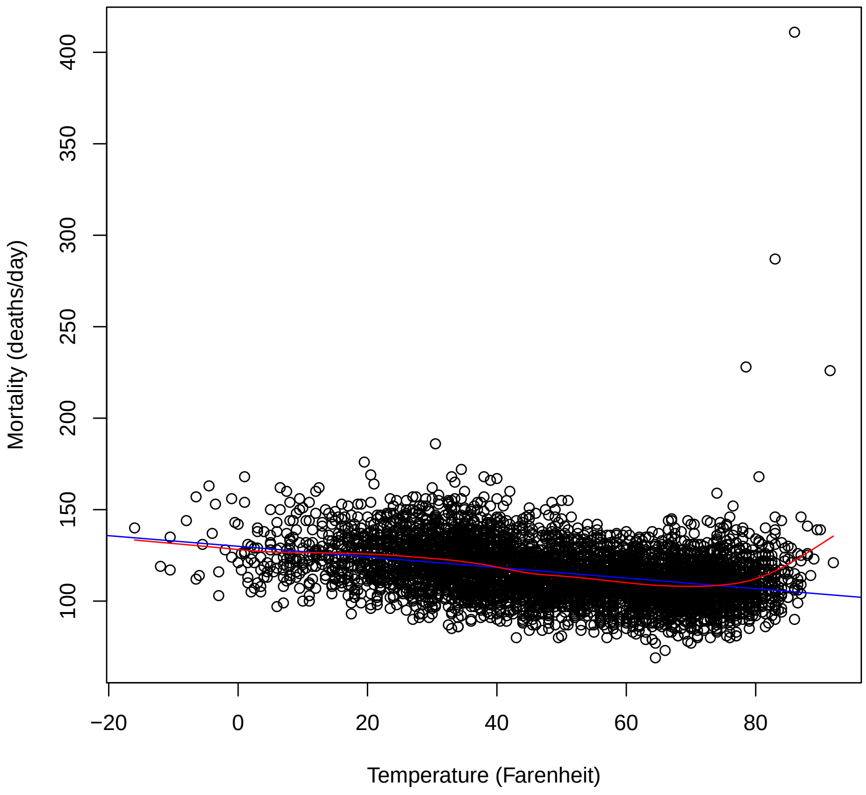
This sort of local averaging gives an $\hat{m}(x)$ which can make jerk steps as x changes. Another approach is **spline smoothing**: this looks for the function, called a **spline** which comes closest to the data points, subject to a constraint on the average curvature of the function. Allowing no curvature at all gives back the least squares line; allowing unlimited curvature gives a function which interpolates exactly between the data points[12]. Of course one has to decide how much curvature to allow; the best idea is generally to do what's called "cross-validation": hold back a little bit of the data, fit the spline with some level of curvature to the rest of the data, and see how well it predicts the held-back part; pick the curvature which generalizes best to unseen data points. While there is lot of R code for splines, because they're very useful, the most user-friendly is called `smooth.spline`.

```
smooth.spline(x,y,cv=TRUE)
```

returns an object which contains the estimated spline. (The default, `CV=FALSE`, is to use a fast approximation to cross-validation called "generalized cross-validation"; `CV=TRUE` is often a bit more accurate, if you can afford the time.) The commands `fitted` and `residuals` work on this object just like they do on `lm` objects; `predict` works very similarly, but the argument giving the new values must be a vector called `x`, not a data frame. If passed into the `lines` or `points` command, we get a plot of the fitted values.

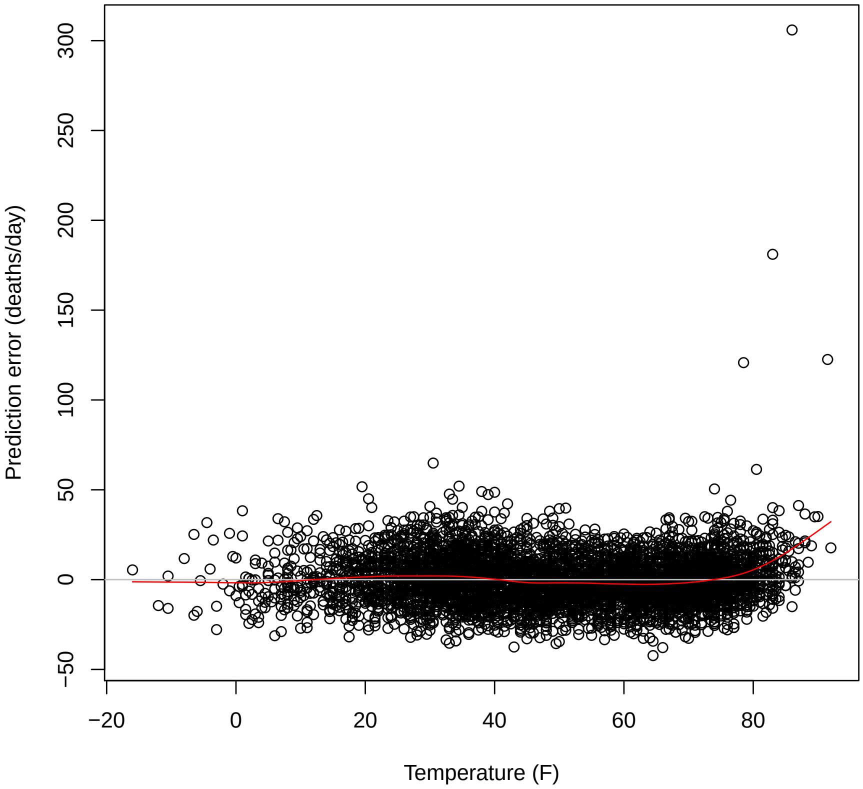
`smooth.spline` can also be used on the squared residuals (not the absolute residuals – why?) to try to get an estimate of the conditional variance, if you're pretty sure that the functional form of the regression is right.

[12] This turns out to be equivalent to a kind of weighted local averaging.



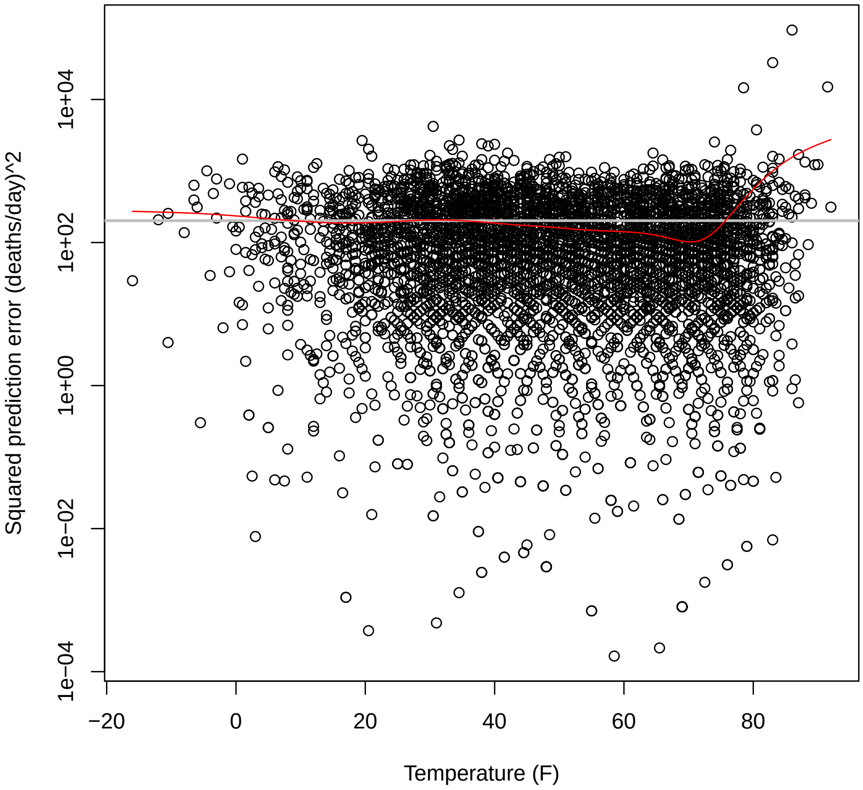
```
plot(death ~ tmpd, data=chicago,
     xlab="Temperature (Farenheit)",
     ylab="Mortality (deaths/day)")
death.temp.ss <- smooth.spline(x=chicago$tmpd, y=chicago$death, cv=TRUE)
abline(death.temp.lm, col="blue")
lines(death.temp.ss, col="red")
```

FIGURE 15: *Scatter-plot of mortality as a function of temperature, along with the estimated linear model (blue) and the estimated smoothing spline (red). Notice how the smoothing spline tracks the linear model over a wide range of temperatures, but then turns* up *for high temperatures.*



```
plot(chicago$tmpd, residuals(death.temp.lm),
     xlab="Temperature (F)",
     ylab="Prediction error (deaths/day)")
abline(h=0, col="grey")
lines(smooth.spline(x=chicago$tmpd, y=residuals(death.temp.lm), cv=TRUE),
      col="red")
```

FIGURE 16: *Estimating a smoothing spline on the residuals of the linear model. Ideally, the spline would be close to zero everywhere; here we see it's working pretty well,* except *at high temperature.*



```
plot(chicago$tmpd, residuals(death.temp.lm)^2, log="y",
     xlab="Temperature (F)",
     ylab="Squared prediction error (deaths/day)^2")
abline(h=mean(residuals(death.temp.lm)^2),
       lwd=2, col="grey")
lines(smooth.spline(x=chicago$tmpd, y=residuals(death.temp.lm)^2,
                    cv=TRUE),
      col="red")
```

FIGURE 17: *Smoothing spline of squared residuals versus temperature. (Notice the logarithmic scale for the vertical axis, to compensate for the fact that some of the residuals are really big.) If we thought that the functional form of the regression was right, this would be a reasonable estimate of the conditional variance. Should we think that?*

3 Transforming the Response

Another way to try to accommodate nonlinearity is to transform the response variable, rather than the predictor. That is, one imagines the model is

$$g(Y) = \beta_0 + \beta_1 x + \epsilon_i$$

for some invertible function g. In more old-fashioned sources, like our textbook, this is advocated as a way of handling non-constant variance, or non-Gaussian noise[13]. A better rationale is that it might in fact be true. Since the transformation g has an inverse, we can write

$$Y = g^{-1}\left(\beta_0 + \beta_1 x + \epsilon_i\right)$$

Even if $\epsilon_i \sim N(0, \sigma^2)$, this implies hat Y will have a non-Gaussian distribution, with a non-linear relationship between $\mathbb{E}\left[Y|X=x\right]$ and x, and a non-constant variance. If that's actually the case, we'd like to incorporate that into the model. For instance, Y might be an integer-valued count variable, or even a binary-valued categorical variable, and then we pretty much have to have some sort of non-Gaussian noise in Y. (Indeed, the noise around $\mathbb{E}\left[Y|X=x\right]$ isn't even strictly additive.)

Let me illustrate these points by working through a log transformation of Y. Suppose

$$\log Y = \beta_0 + \beta_1 x + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$, independent of x. The inverse function to log is exp, so this is logically equivalent to

$$Y = e^{\beta_0 + \beta_1 x} e^{\epsilon} = e^{\beta_0 + \beta_1 x} \xi$$

where $\xi = e^{\epsilon}$ follows a **log-normal** distribution, with $\mathbb{E}\left[\log \xi\right] = 0$, $\mathrm{Var}\left[\log \xi\right] = \sigma^2$. One can show[14] that $\mathbb{E}\left[\xi\right] = e^{\sigma^2/2}$, that $\mathrm{Var}\left[\xi\right] = (e^{\sigma^2} - 1)e^{\sigma^2}$, that the median of ξ is 1, and that ξ is, consequently, skewed to the right, with an asymmetric distribution. Conversely, if $Y = e^{\beta_0 + \beta_1 x} + \epsilon$, with $\epsilon \sim N(0, \sigma^2)$, then it is *not* the case that $\log Y = \beta_0 + \beta_1 x + \eta$, for some Gaussian η.

The point of this is that transforming the response thoroughly changes the interpretation of the parameters: β_0 is not $\mathbb{E}\left[Y|X=x\right]$, but $\mathbb{E}\left[g(Y)|X=x\right]$, β_1 is the slope of $\mathbb{E}\left[g(Y)|X=x\right]$, etc. It also implies very particular, even peculiar, models for noise around the regression line. This makes it impossible to compare mean squared errors or log-likelihoods before and after transformations[15]. One can, however, look at the residuals for the *transformed* response, and see whether they are flat in the predictor variable, etc.

[13] To be quite honest, this seems like as clear a case of being enslaved by one's own tools as one could hope to find in the sciences. More charitably: it made a certain amount of sense before people figures out how to deal with non-constant variance (around 1935) and non-Gaussian noise (around 1980).

[14] Check Wikipedia if you don't believe me.

[15] More exactly, log-likelihoods *can* be compared, but we need to compensate for the transformation.

3.1 Box-Cox Transformations

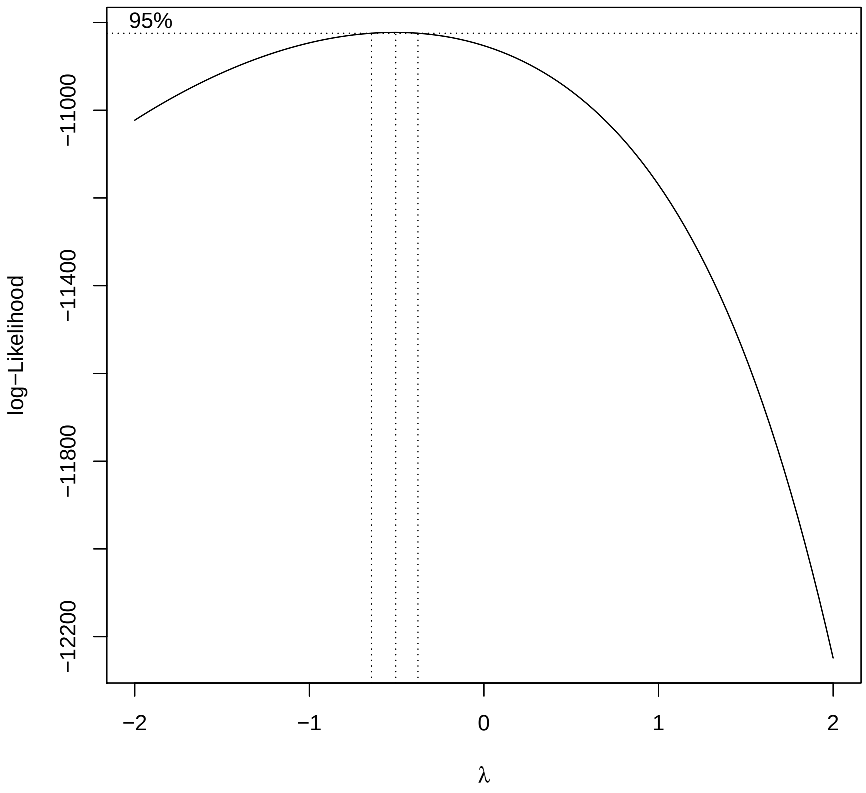
The great statisticians G. E. P. Box and D. R. Cox introduced a family of transformations which includes powers of Y and taking the logarithm of Y, parameterized by a number λ:

$$b_\lambda(y) = \frac{y^\lambda - 1}{\lambda}$$

In the limit $\lambda \rightarrow 0$, this becomes $\log y$ (Exercise 2). If one assumes that

$$\begin{aligned} b_\lambda(Y) &= \beta_0 + \beta_1 x + \epsilon && (10)\\ \epsilon &\sim N(0, \sigma^2) && (11)\\ \epsilon \ &\text{independent of} \ x && (12)\\ & && (13) \end{aligned}$$

then one can estimate λ by maximizing the likelihood. This is implemented in R through the function `boxcox` in the package `MASS`.

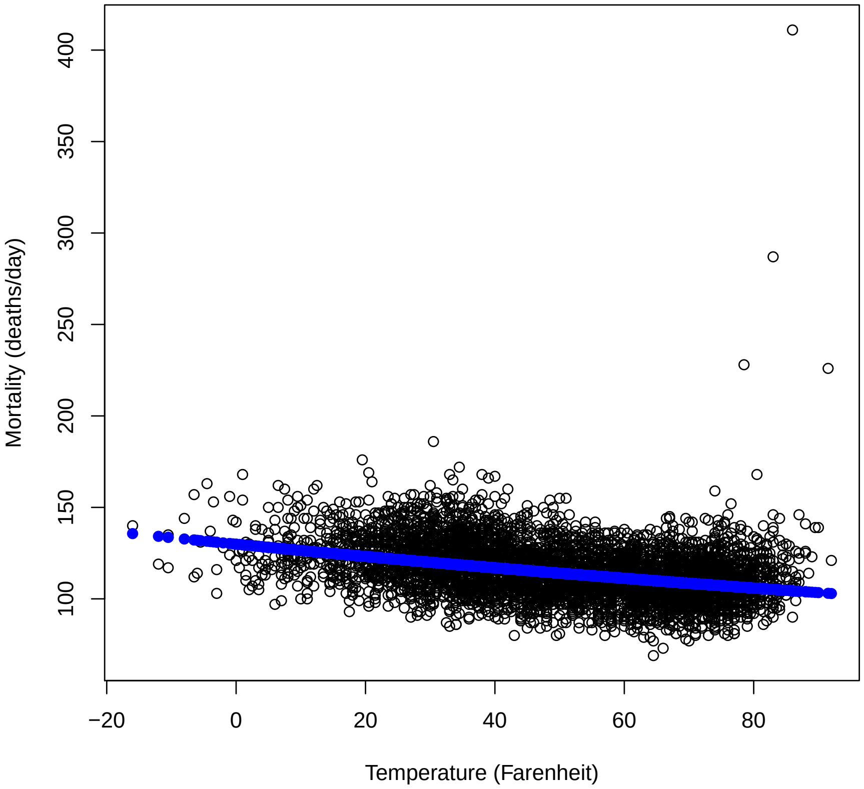


```
library(MASS)
# Works with a previously-fit lm model
boxcox(death.temp.lm)
```

FIGURE 18: *Plot of the log-likelihood of different values of λ in the Box-Cox transformation, applied to the regression of deaths on temperature. The dashed lines indicate an approximate 95% confidence interval for λ.*

```
# You can also give a model formula, and suppress plotting
bc.death.temp <- boxcox(death ~ tmpd, data=chicago, plotit=FALSE)
# Result is a list showing lambda values (£x component) vs. log-likelihood
# (the £y component)
# We can use this to get a (rough) maximum value for lambda
(lambda.hat <- bc.death.temp$x[which.max(bc.death.temp$y)])

## [1] -0.5
```

FIGURE 19: *Another way of using the `boxcox` function.*



```
chicago$bc.death <- (chicago$death^lambda.hat-1)/lambda.hat
bc.lm <- lm(bc.death ~ tmpd, data=chicago)
plot(death ~ tmpd, data=chicago,
     xlab="Temperature (Farenheit)",
     ylab="Mortality (deaths/day)")
points(chicago[,"tmpd"], (lambda.hat*fitted(bc.lm)+1)^(1/lambda.hat), pch=19,
       col="blue")
```

FIGURE 20: *Actual data (hollow circles) and fitted values from the Box-Cox transformation.*

It is important to remember that this family of transformations is just something that Box and Cox made up because it led to tractable math. There is absolutely no justification for it in probability theory, general considerations of mathematical modeling, or scientific theories. There is also no reason to think that the correct model will be one where some transformation of Y is a linear function of the predictor variable plus Gaussian noise. Estimating a Box-Cox transformation by maximum likelihood does not relieve us of the need to run all the diagnostic checks *after* the transformation. Even the best Box-Cox transformation may be utter rubbish.

I will add that while Box-Cox transformations have been part of courses like this since they were introduced in 1964, I have never encountered a real, non-textbook data-analysis problem where they helped, with *one* exception, which I will discuss immediately.

Log transformations with log-normal multiplicative noise The usual argument for anticipating Gaussian noise is the central limit theorem: if lots of little disturbing causes add up their contributions to the response variable, and they're roughly independent and of comparable size, we'll expect that their net effect is Gaussian. If instead of adding up they multiply together, however, the central limit theorem does not apply directly. Since logarithms turn multiplication into addition, when effects multiply we may anticipate log-normal fluctuations. If the simple linear regression model applies after taking the log of everything,

$$\begin{aligned} \log Y &= \beta_0 + \beta_1 logx + \epsilon && (14)\\ \epsilon &\sim N(0,\sigma^2) && (15)\\ \epsilon &\text{ independent of } X && (16) \end{aligned}$$

then, undoing the log,

$$\begin{aligned} Y &= e^{\beta_0} x^{\beta_1} \eta && (17)\\ \eta &\sim \log N(0,\sigma^2) \\ \eta &\text{ independent of } X \end{aligned}$$

This sort of model is not actually unheard of in practice, because there really are situations where causes multiply.

Eq. 17 is a *different model* from

$$Y = e^{\beta_0} x^{\beta_1} + \epsilon$$

We would estimate the formal model by doing a linear regression of $\log Y$ on $\log X$ with Gaussian noise. We would estimate the latter model by nonlinear least squares. xs

4 Looking Forward

Non-constant variance in a linear model is actually comparatively easy to handle, if we can work out what variance is: the trick is a modification of the method of least squares called "weighted least squares", which we will cover later in the course. Similarly, correlated noise can be handled through a modification called "generalized least squares", which we'll also cover. There are a range of simulation-based techniques for doing inference when the Gaussian-noise assumption fails; these have opaque, forcedly-whimsical names like "the jackknife" and "the bootstrap", and we'll get to these towards the end of the course. Dealing with nonlinearity will occupy us for much of 402, though in a few weeks we will look at polynomial regression.

The easiest direction to go in is to add more predictor variables, and hope that the response is linear in each of them. We will explore this path after the midterm.

5 Residuals under Misspecification

(This section is optional, but strongly recommended.)

Suppose that the conditional expectation function $\mu(x) \equiv \mathbb{E}[Y|X=x]$ is not linear in x, but we use a linear model anyway. We know, from the first lecture, that there is a best linear approximation to this μ, with slope $\beta_1 = \mathrm{Cov}[X,Y]/\mathrm{Var}[X]$ and intercept $\beta_0 = \mathbb{E}[Y] - \beta_1 \mathbb{E}[X]$. So we can write

$$Y = \mu(x) + \epsilon \quad (18)$$

$$Y = \beta_0 + \beta_1 x + (\mu(x) - \beta_0 + \beta_1 x) + \epsilon \quad (19)$$

It's annoying to keep writing out $\mu(x) - \beta_0 + \beta_1 x$, so let's define that to be a new function, $\nu(x) \equiv \mu(x) - \beta_0 + \beta_1 x$. You can show (Exercise 3) that

$$\mathbb{E}[\nu(X)] = 0$$

and that

$$\mathrm{Cov}[\nu(X), X] = 0$$

However, it is not in general the case that

$$\mathbb{E}[\nu(X)|X=x] = 0$$

Suppose now that — through ignorance or as a deliberate strategy — we fit a linear model by least squares. It will be the case that

$$Y_i = \beta_0 + \beta_1 x_i + \eta_i$$

where η_i has expectation zero and is uncorrelated with X. But what we are treating as noise is partially the real noise, and partially the systematic effect of ignoring the nonlinearities:

$$\eta_i = \epsilon_i + \nu(x_i)$$

Thus, what looks like noise to the linear model will *not* have conditional expectation zero:

$$\mathbb{E}\left[\eta|X=x\right] = \nu(x)$$

The conditional variance doesn't alter,

$$\mathrm{Var}\left[\eta|X=x\right] = \mathrm{Var}\left[\epsilon|X=x\right]$$

but it's no longer equal to the expected square:

$$\mathbb{E}\left[\eta^2|X=x\right] = \nu^2(x) + \mathrm{Var}\left[\epsilon|X=x\right]$$

As mentioned in earlier lectures, when we run least squares, our slope and intercept will still tend to converge on those of the optimal linear model. We can even still use the expressions we worked out for $\hat{\beta}_0$ (or $\hat{\beta}_1$) in terms of β_0 (or β_1) and a weighted sum of random variables — but now those are the η_i, not the ϵ_i. This implies that conditional expectation of the residuals will *not* be zero (in general), the conditional expectation of the squared residuals will not be constant, and so forth.

Exercises

To think through or for practice, not to hand in.

1. Using Eq. 6, find the covariance, conditional on all the x_i, between e_i and e_k, for arbitrary $i, k \in 1:n$.
2. Show that $b_\lambda(y) \rightarrow \log y$ as $\lambda \rightarrow 0$. *Hint:* L'Hôpital's rule.
3. Consider the $\nu(x)$ function from §5.
   (a) Show that $\mathbb{E}\left[\nu(X)\right] = 0$. *Hint:* Show that $\mathbb{E}\left[Y\right] = \mathbb{E}\left[\mu(X)\right]$.
   (b) Show that $\mathrm{Cov}\left[X, \nu(X)\right] = 0$. *Hint:* Show that $\mathrm{Cov}\left[X, Y\right] = \mathrm{Cov}\left[X, \mu(X)\right]$.