

Chapter 6

Diagnostics and Modifications for Simple Regression

In the previous chapters, 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 chapters 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 wrong¹.) The simplest and most useful check all involve the residuals.

6.1 The Residuals

In previous chapters, 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

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

predict:

$$e_i = y_i - \hat{m}(x_i) \quad (6.1)$$

which, for the simple linear regression model, is just

$$e_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) \quad (6.2)$$

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

$$\epsilon_i = Y_i - (\beta_0 + \beta_1 x_i) \quad (6.3)$$

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 ,

$$e_i = \beta_0 + \beta_1 x_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) + \epsilon_i \quad (6.4)$$

$$= (\beta_0 - \hat{\beta}_0) + (\beta_1 - \hat{\beta}_1) x_i + \epsilon_i \quad (6.5)$$

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 chapters,

$$\hat{\beta}_0 = \beta_0 + \frac{1}{n} \sum_{i=1}^n \left(1 - \bar{x} \frac{x_i - \bar{x}}{s_X^2}\right) \epsilon_i \quad (6.6)$$

$$\hat{\beta}_1 = \beta_1 + \sum_{i=1}^n \frac{x_i - \bar{x}}{n s_X^2} \epsilon_i \quad (6.7)$$

Substitute these in to the equation for e_i :

$$e_i = \epsilon_i + \frac{1}{n} \sum_{j=1}^n \left(1 - \bar{x} \frac{x_j - \bar{x}}{s_X^2}\right) \epsilon_j + x_i \sum_{j=1}^n \frac{x_j - \bar{x}}{n s_X^2} \epsilon_j \quad (6.8)$$

$$= \sum_{j=1}^n \left(\delta_{ij} + \frac{1}{n} + (x_i - \bar{x}) \frac{x_j - \bar{x}}{n s_X^2} \right) \epsilon_j \quad (6.9)$$

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 $\mathbb{E}[\epsilon_i | X] = 0$. Since we've shown that e_i is a weighted sum of ϵ_j , it follows that

$$\mathbb{E}[e_i | X] = \sum_j c_{ij} \mathbb{E}[\epsilon_j | X] = 0 \quad (6.10)$$

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

$$\text{Var}[e_i|X] = \sum_j \text{Var}[c_{ij}\epsilon_j|X] \quad (6.11)$$

$$= \sum_j c_{ij}^2 \text{Var}[\epsilon_j|X] \quad (6.12)$$

$$= \sigma^2 \sum_{j=1}^n c_{ij}^2 \quad (6.13)$$

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

$$\text{Var}[e_i] = \frac{n-2}{n} \sigma^2 \quad (6.14)$$

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,

$$\mathbb{E}\left[\sum_{i=1}^n \epsilon_i\right] = \sum_{i=1}^n \mathbb{E}[\epsilon_i] = 0 \quad (6.15)$$

but the *variance* is not:

$$\text{Var}\left[\sum_{i=1}^n \epsilon_i\right] = \sum_{i=1}^n \text{Var}[\epsilon_i] = n\sigma^2 \quad (6.16)$$

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

$$\sum_{i=1}^n \epsilon_i \neq 0 \quad (6.17)$$

(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 ,

$$\text{Cov}[X, \epsilon] = \mathbb{E}[X\epsilon] - \mathbb{E}[\epsilon]\mathbb{E}[X] = 0 \quad (6.18)$$

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

$$\sum_{i=1}^n \epsilon_i(x_i - \bar{x}) \neq 0 \quad (6.19)$$

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

$$\sum_{i=1}^n e_i = 0 \quad (6.20)$$

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,

$$\sum_{i=1}^n (x_i - \bar{x})e_i = 0 \quad (6.21)$$

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, \dots, 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 $\text{Var}[e_i]$.)

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](#)).

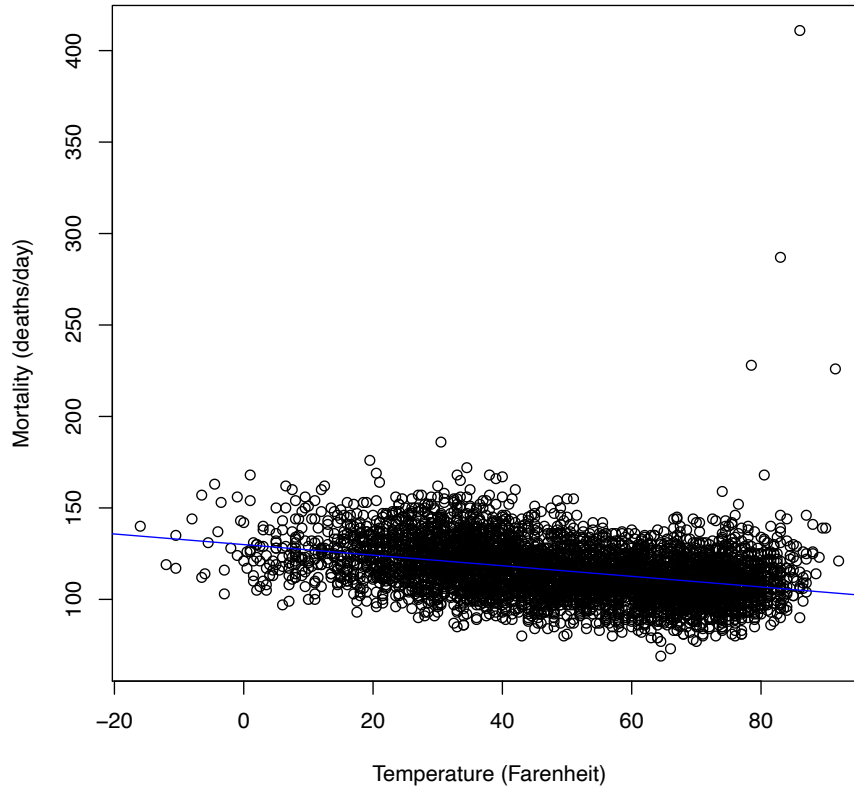
6.1.1 Summary on Properties of the Residuals

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

1. The residuals should have expectation zero, conditional on x , $\mathbb{E}[e_i | X = x] = 0$. (The residuals should also have an over-all sample mean of exactly zero.)
2. The residuals should show a nearly-constant variance [2](#).
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.

²We've actually seen a formula which lets us work out the variance of the residuals — which one?



```
# 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 (Fahrenheit)", ylab = "Mortality (deaths/day)")

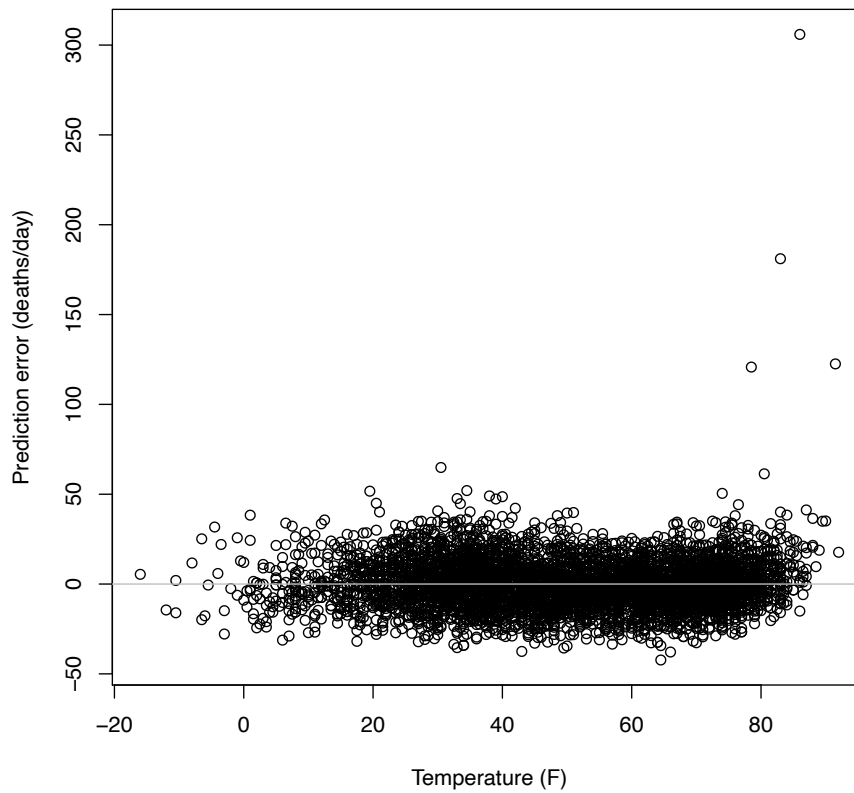
# Estimate and store a linear model
death.temp.lm <- lm(death ~ tmpd, data = chicago)
abline(death.temp.lm, col = "blue")
```

FIGURE 6.1: Plot of the data from the last homework, used as a running example, along with the estimated linear model (in blue).

6.1.2 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 $\mathbb{E}[e|X = x] = 0$, and $\text{Var}[e|X = x]$ 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 $\mathbb{E}[e|X = x] \neq 0$, which in turn means that $\mathbb{E}[\epsilon|X = x] \neq 0$, which means that the simple-linear part of the simple linear regression model is wrong³. 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.

³See §6.5



```
# 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 6.2: *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 ignorance⁴.) 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.

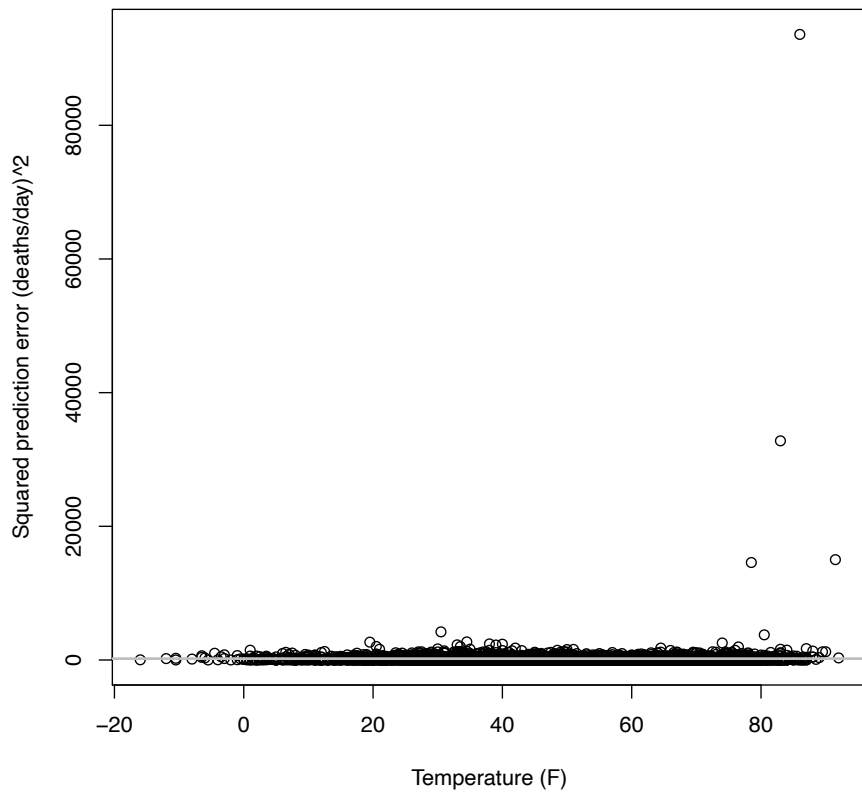
(Exercise: make a plot of the residuals from this model against one of the pollution variables from the data set. Does it look like noise?)

6.1.3 Plot the Magnitude of the Residuals Against the Predictor

Because $\mathbb{E}[e|X = x] = 0$, $\text{Var}[e|X = x] = \mathbb{E}[e^2|X = x]$. (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.

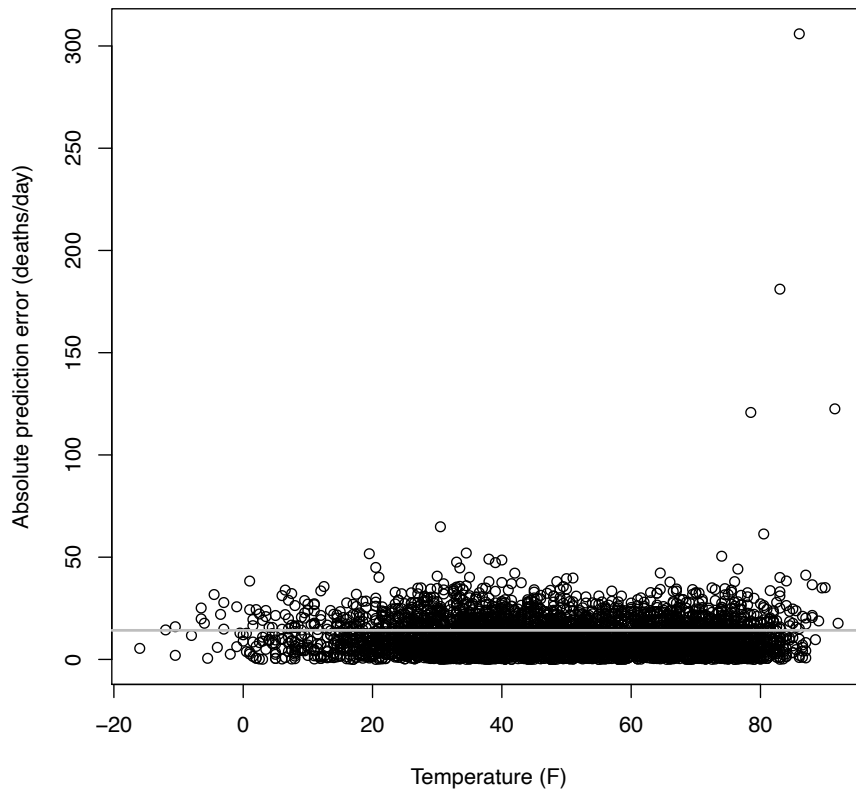
Sometimes, particularly when the model is not doing so well, squaring the residuals leads to a visually uninformative plot, because big residuals lead to really, really big squared residuals, and it's hard to make out any detail. A common fix is to then plot the absolute value of the residuals, with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

⁴Whether 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 6.3: *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 6.4: *Absolute residuals vs. temperature; plotting these rather than the squared residuals reduces the visual impact of the few huge residuals.*

6.1.4 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 matter⁵, 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.

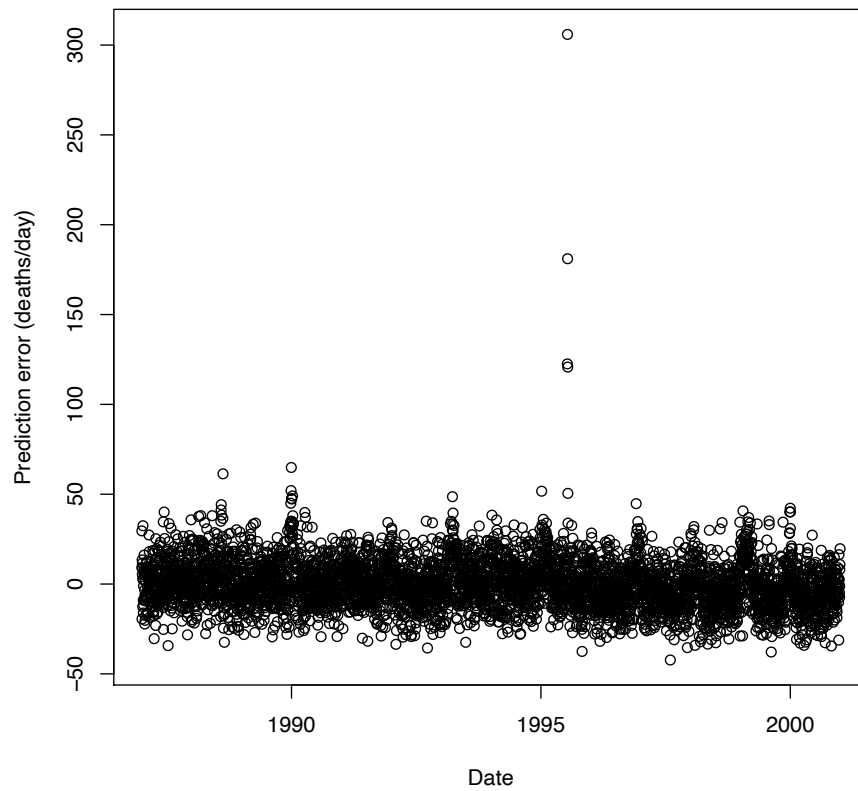
Of course, a certain amount of apparent clustering will happen naturally in any random process, so if this looks worrisome, one should really do some sort of formal test. Fortunately, there are lots of good test procedures for finding “runs” in what should be random noise. A quick hack, though, is simply 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.)

⁵Unless the predictor variable is one of the coordinates, of course.



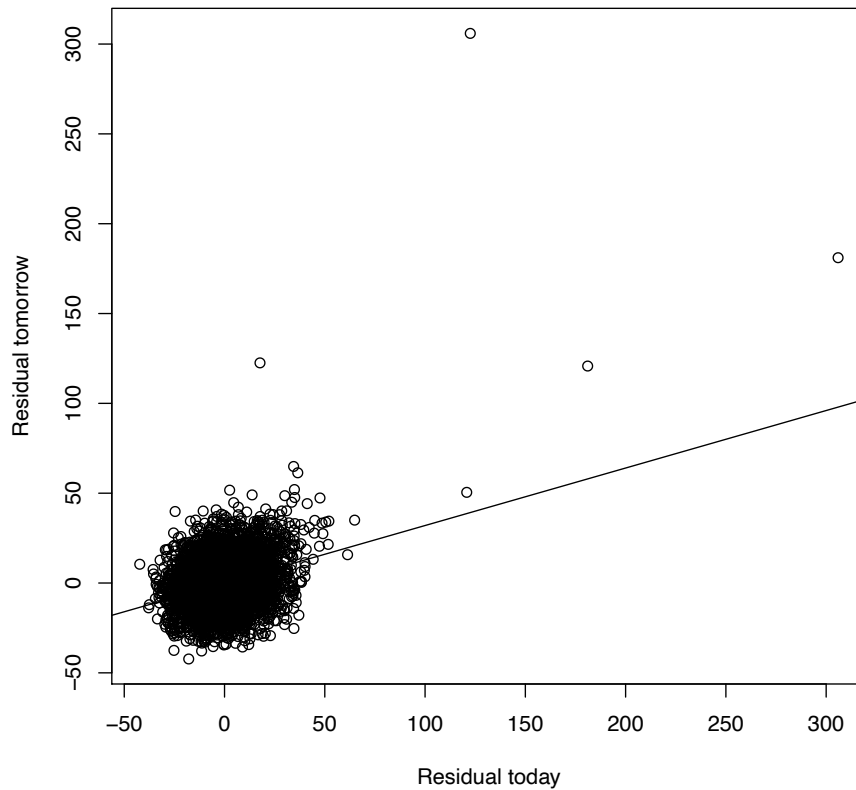
```
# Always plot residuals vs. coordinate
plot(as.Date(chicago$time, origin = "1993-12-31"), residuals(death.temp.lm),
     xlab = "Date", ylab = "Prediction error (deaths/day)")
```

FIGURE 6.5: *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 correlated⁶

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.

⁶Or, again, it could be a sign that we've got the functional form wrong, so it's our systematic error which is correlated — see §6.5



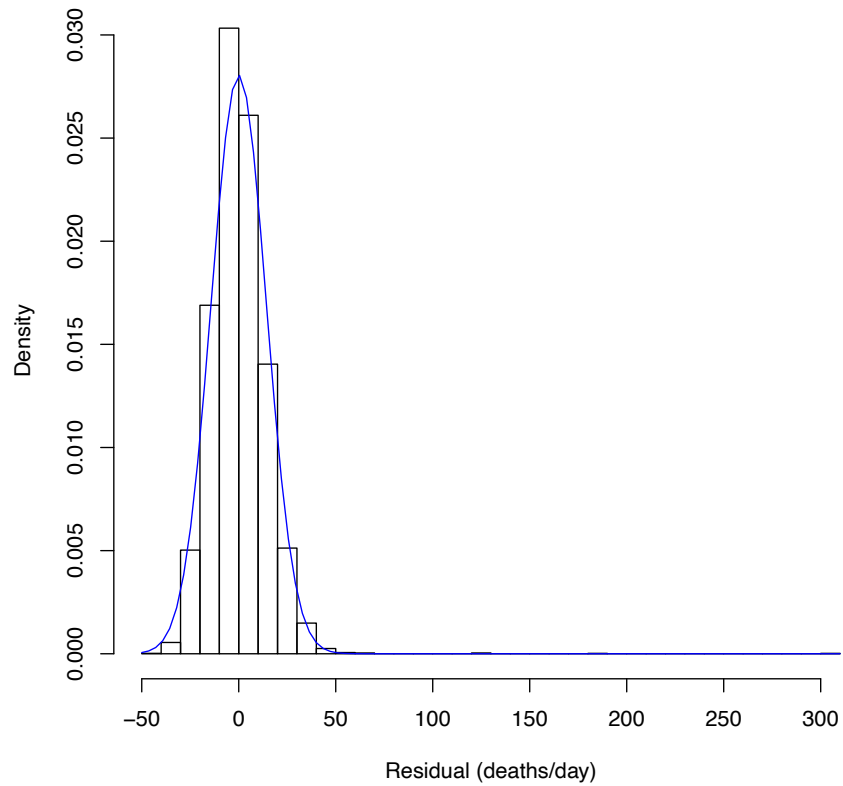
```
# Always plot successive residuals against each other head() and tail() here
# are used to get 'every day except the last' and 'every day except the
# first', respectively see help(head)
plot(head(residuals(death.temp.lm), -1), tail(residuals(death.temp.lm), -1),
      xlab = "Residual today", ylab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

FIGURE 6.6: *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.*

6.1.5 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 6.7: 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 95th 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 distribution⁷.

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, \dots, 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

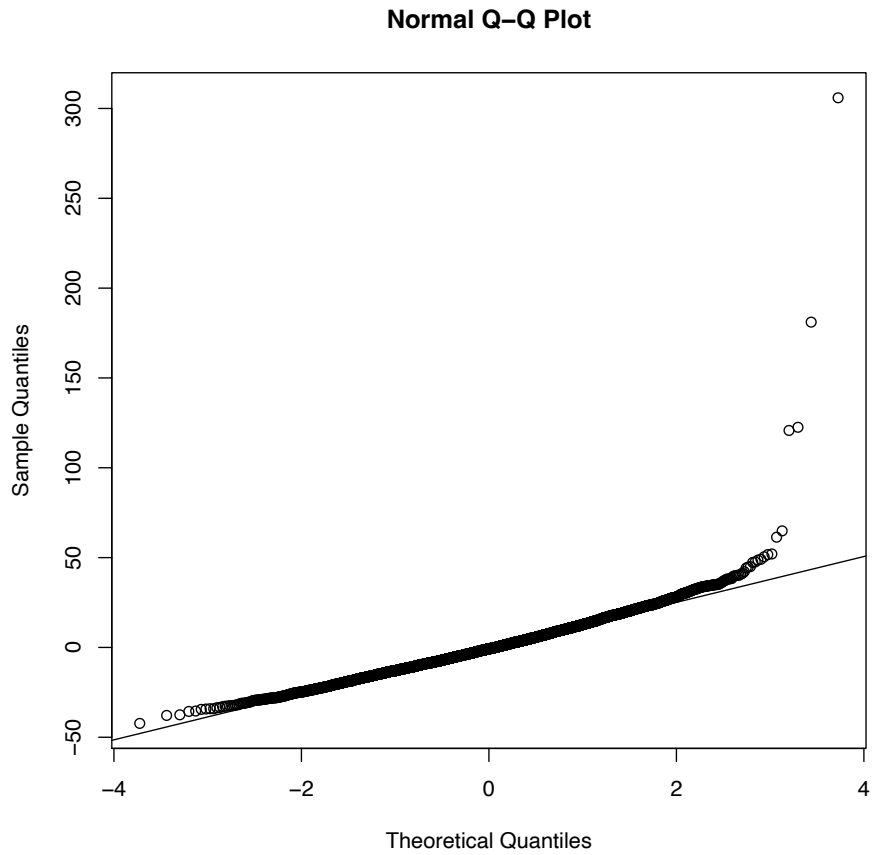
$$x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n-1)} \leq x_{(n)} \quad (6.22)$$

(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 Gaussian⁸.

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

⁷In 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.

⁸You 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 $\mathbb{E}[X_{(i)}]$ for $i \in 1 : n$ under a Gaussian distribution, and plots *that* against the observed value $x_{(i)}$. As $n \rightarrow \infty$, $\mathbb{E}[X_{(i)}] \rightarrow \Phi^{-1}(i/n)$, but the difference can be important for small n , and/or i/n close to 0 or 1.



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

FIGURE 6.8: QQ plot of the residuals, using the standard Gaussian distribution as the reference 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, \dots, x_n and y_1, \dots, 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 test⁹. 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`.)

⁹This is why the visual comparison is informally called "χ by eye".

```

# 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

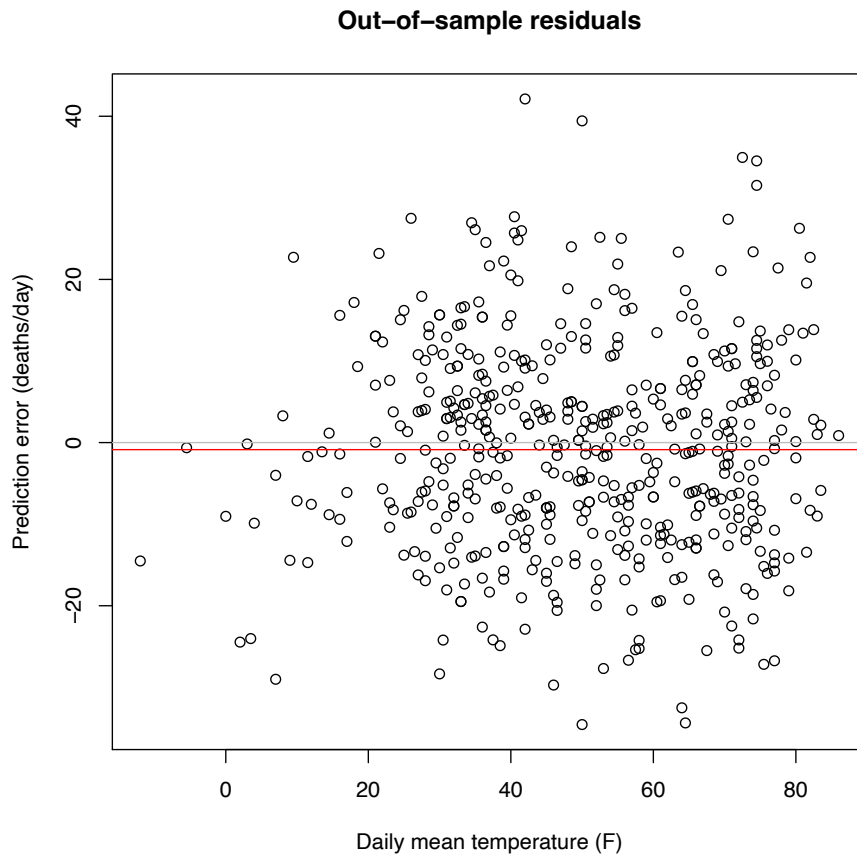
```

FIGURE 6.9: 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).

6.1.6 Generalization

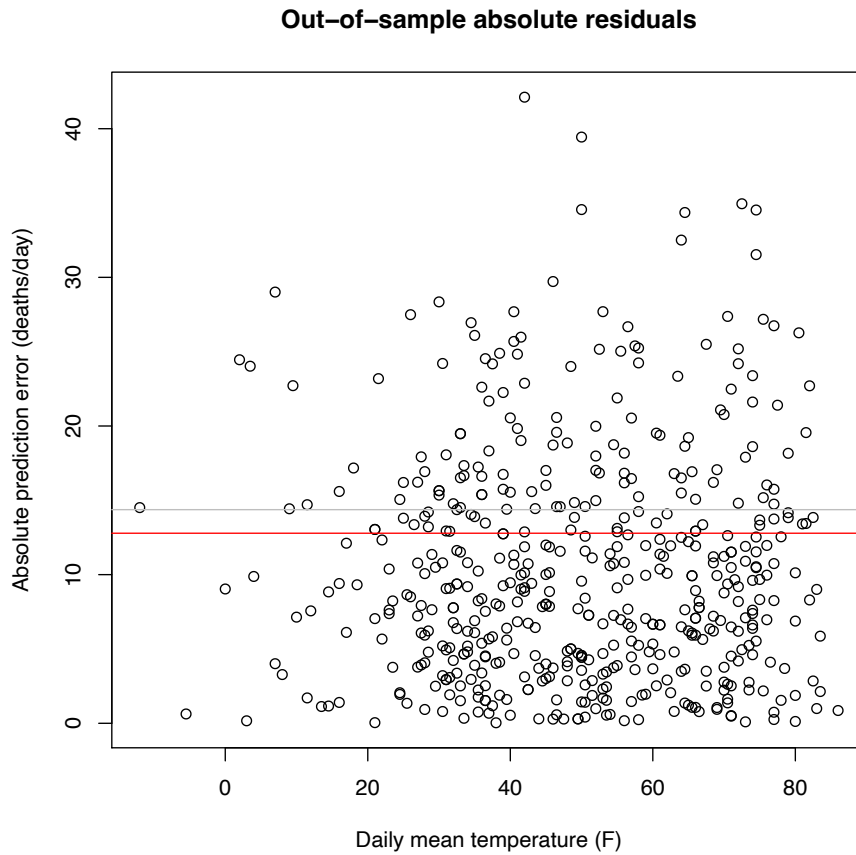
If the model assumptions are correct, it should be able to work about equally well on new data from the same source. Because the parameters were adjusted to fit the data we used to estimate the model, we should expect the prediction errors on new data to be slightly larger in magnitude, but they shouldn't 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 6.10: *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 6.11: As in Figure 6.10, but looking at the squared residuals.

```

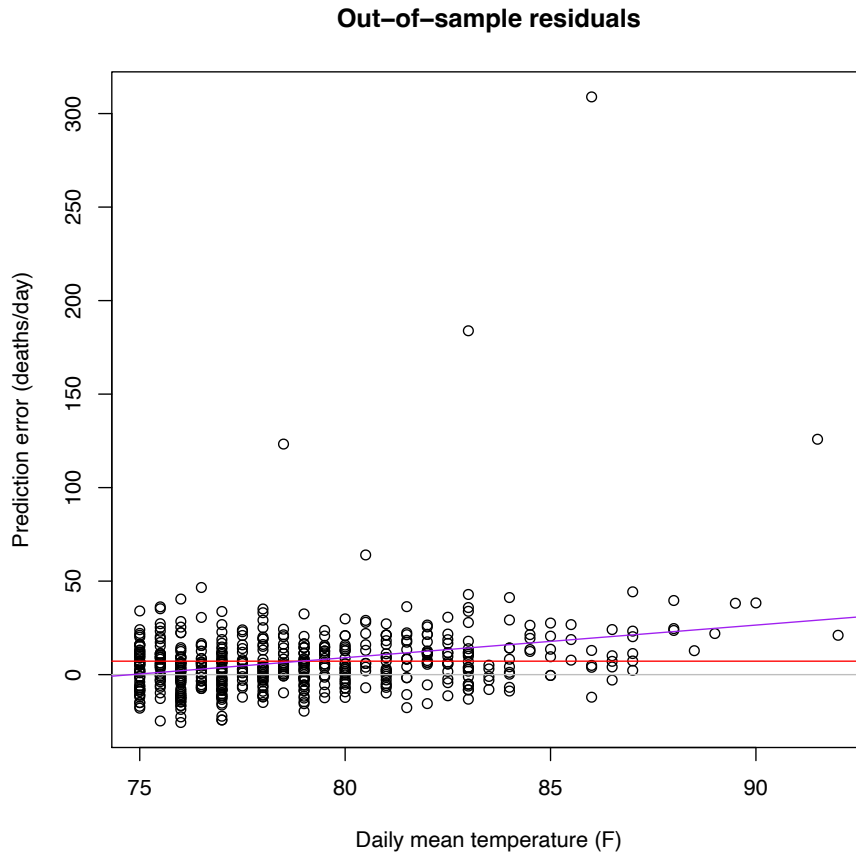
# Find the low-temperature days
lowtemp.rows <- which(chicago$tmpd < 75) # About 90% of the data
# Divide into low- and high- temperature data sets
lowtemp.set <- chicago[lowtemp.rows, ]
hightemp.set <- chicago[-lowtemp.rows, ]
# Estimate the model on the colder days only
lowtemp.lm <- lm(death ~ tmpd,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

```

FIGURE 6.12: *Setting up a division of the data into a low-temperature training set and a high-temperature testing set.*

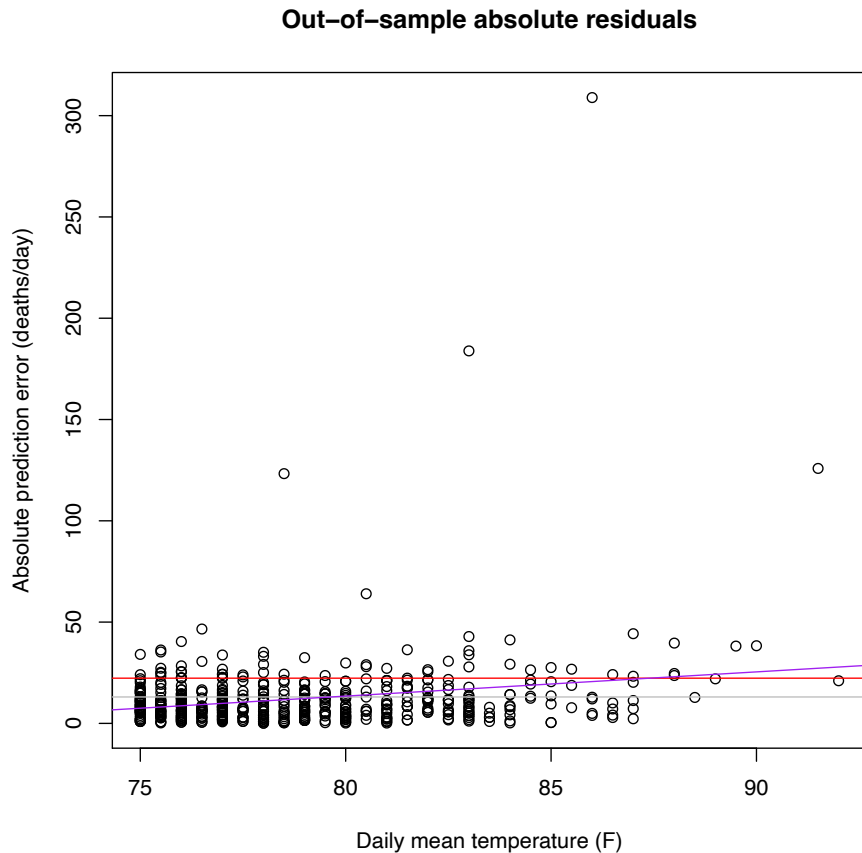
Extrapolative Generalization An alternative to random division, which can be even more useful for model checking, is to systematically make the testing set into a part of the data where the over-all fit of the model seems dubious based on other diagnostics. With our running examples, for instance, the 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 6.13: *Residuals vs. temperature, where the testing set here consists of days with temperature ≥ 75 degrees Fahrenheit, 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 6.14: As in Figure 6.13, but for absolute residuals.