# Lecture 9: Predictive Inference

There are (at least) three levels at which we can make predictions with a regression model: we can give a single best guess about what $Y$ will be when $X = x$, a *point prediction*; we can try to guess the whole probability distribution of possible $Y$ values, a *distributional* prediction; or we can, less ambitiously, try to make an *interval* prediction, saying "with such-and-probability, $Y$ will be in the interval between here and there".

## 1 Confidence intervals for conditional means

The conditional mean at any particular $x$ is just a number; we can do inference on it as though it were a parameter; it is, after all, a function of the parameters. More specifically, the true conditional mean is

$$m(x) \equiv \mathbb{E}\left[Y|X=x\right] = \beta_0 + \beta_1 x \tag{1}$$

while our estimate of the conditional mean is

$$\widehat{m}(x) = \widehat{\beta}_0 + \widehat{\beta}_1 x \tag{2}$$

(See note on notation below.)

We've seen, in lecture 5, that

$$\widehat{m}(x) = \beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i \tag{3}$$

so that

$$\mathbb{E}\left[\widehat{m}(x)\right] = \beta_0 + \beta_1 x = m(x) \tag{4}$$

and

$$\mathrm{Var}\left[\widehat{m}(x)\right] = \frac{\sigma^2}{n}\left(1 + \frac{(x-\overline{x})^2}{s_X^2}\right) \tag{5}$$

Under the Gaussian noise assumption, $\widehat{m}(x)$ is Gaussianand

$$\widehat{m}(x) \sim N\left(m(x), \frac{\sigma^2}{n}\left(1 + \frac{(x-\overline{x})^2}{s_X^2}\right)\right) \tag{6}$$

Notice how the variance grows as we move further and further away from the center of the data along the $x$ axis. Also notice how all the unknown parameters show up on the right-hand side of the equation.

**Exact confidence intervals** At this point, getting confidence intervals for $m(x)$ works just like getting confidence intervals for $\beta_0$ or $\beta_1$: we use as our standard error

$$\widehat{\mathrm{se}}\left[\widehat{m}(x)\right] = \frac{\widehat{\sigma}}{\sqrt{n-2}}\sqrt{1 + \frac{(x-\overline{x})^2}{s_X^2}} \tag{7}$$

and then find

$$\frac{\widehat{m}(x) - m(x)}{\widehat{\mathrm{se}}\left[\widehat{m}(x)\right]} \sim t_{n-2} \tag{8}$$

by entirely parallel arguments. $1-\alpha$ confidence intervals follow as before as well.

**Notation** The textbook, following an old tradition, talks about $\widehat{y}$ as the conditional mean. This is not a good tradition, since it leads to great awkwardness in distinguishing the true conditional mean from our estimate of it. Hence my use of $m(x)$ and $\widehat{m}(x)$.

### 1.1 Large-$n$ approximation

As $n$ grows, the $t$ distribution with $n-2$ degrees of freedom becomes, approximately, the standard Gaussian. It follows that for large $n$,

$$\frac{\widehat{m}(x) - m(x)}{\widehat{\mathrm{se}}\left[\widehat{m}(x)\right]} \approx N(0,1) \tag{9}$$

so

$$\widehat{m}(x) \approx N(m(x), \widehat{\mathrm{se}}\left[\widehat{m}(x)\right]^2) \tag{10}$$

and an approximate $1-\alpha$ confidence interval for $m(x)$ is

$$\widehat{m}(x) \pm z_{\alpha/2}\frac{\widehat{\sigma}}{\sqrt{n}}\sqrt{1 + \frac{(x-\overline{x})^2}{s_X^2}} \tag{11}$$

(It doesn't matter whether we use $n-2$ or $n$ in the denominator for $\widehat{\mathrm{se}}$.) Notice that the width of this interval $\rightarrow 0$ as $n \rightarrow \infty$.

### 1.2 Confidence intervals and transformations

Transforming the predictor variable raises no particular issues. Transforming the response, however, is quite delicate.

When we transform the response, the model becomes

$$g(Y) = \beta_0 + \beta_1 x + \epsilon \tag{12}$$

for $\epsilon$ IID Gaussian, $N(0,\sigma^2)$. Now

$$\mathbb{E}\left[g(Y) \mid X = x\right] = \beta_0 + \beta_1 x \tag{13}$$

and so if we go through the calculates above, we get confidence intervals for $\mathbb{E}\left[g(Y) \mid X = x\right]$, the conditional expectation of the *transformed* response.

In general, however,

$$\mathbb{E}\left[Y \mid X = x\right] \neq g^{-1}(\beta_0 + \beta_1 x) \tag{14}$$

so just applying $g^{-1}$ to the confidence limits for $\mathbb{E}\left[g(Y) \mid X = x\right]$ won't give us a confidence interval for $\mathbb{E}\left[Y|X = x\right]$.

## 2 Prediction Interval

A $1-\alpha$ **prediction interval** for $Y|X = x$ is a an interval $[l, u]$ where

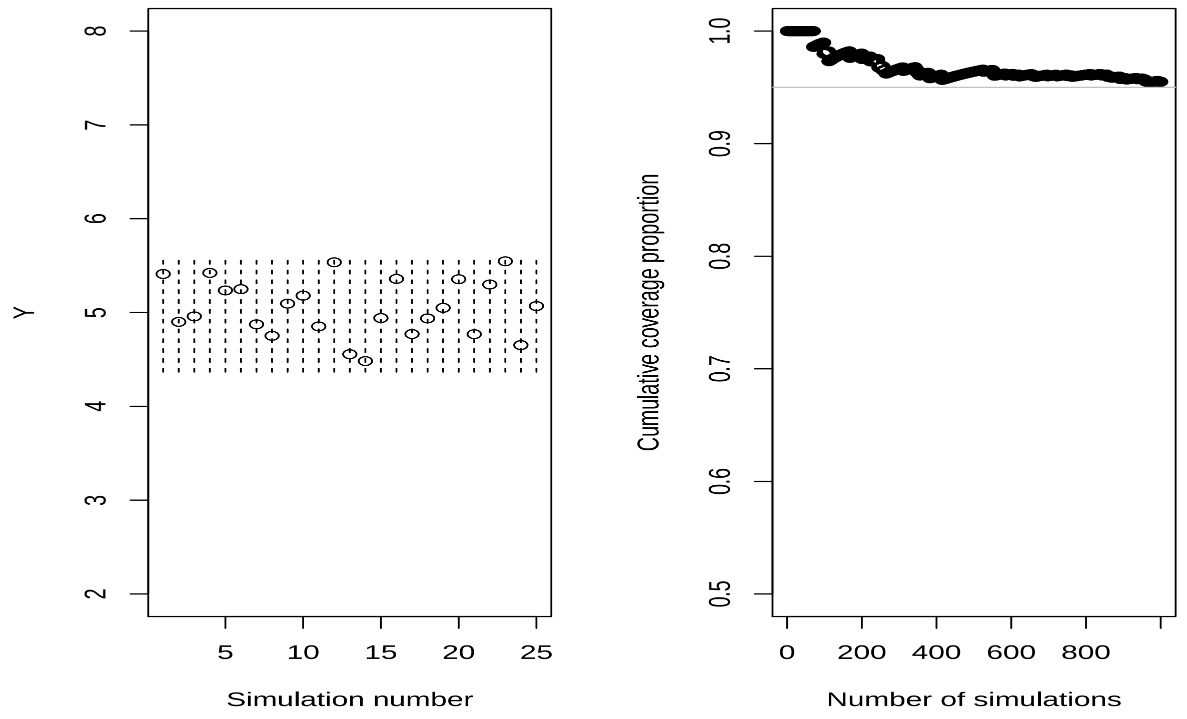
$$\mathbb{P}\left(l \leq Y \leq u | X = x\right) = 1 - \alpha \tag{15}$$

```
# Simulate a Gaussian-noise simple linear regression model
# Inputs: x sequence; intercept; slope; noise variance; switch for whether to
  # return the simulated values, or run a regression and return the estimated
  # model
# Output: data frame or coefficient vector
sim.gnslrm <- function(x, intercept, slope, sigma.sq, mdl=TRUE) {
  n <- length(x)
  y <- intercept + slope*x + rnorm(n,mean=0,sd=sqrt(sigma.sq))
  if (mdl) {
    return(lm(y~x))
  } else {
    return(data.frame(x=x, y=y))
  }
}

# Read in a model and get it to give a prediction interval at a given x
  # This will be convenient when we want to have lots of models make predictions
  # at the same point
# Inputs: the model, the value of x
# Output: vector giving prediction interval
extract.pred.int <- function(mdl,x,level=0.95) {
    predict(mdl,newdata=data.frame(x=x),interval="prediction",level=level)
}


# No magic numbers!
x.new <- 1/137
m <- 1000
alpha <- 0.05
beta.0 <- 5
beta.1 <- -2
sigma.sq <- 0.1
```

FIGURE 1: *Code setting up a simulation of a Gaussian-noise simple linear regression model, along a fixed vector of $x_i$ values, followed by some default values we'll use in the later simulations.*



```
# Simulate Y from the model
y.new <- sim.gnslrm(x=rep(x.new,m),beta.0,beta.1,sigma.sq,mdl=FALSE)$y
# All the prediction intervals are the same (because x isn't changing)
pred.int <- beta.0 + beta.1*x.new + sqrt(sigma.sq)*qnorm(c(alpha/2,1-alpha/2))
names(pred.int) <- c("lwr","upr") # Names make for clearer code
par(mfrow=c(1,2)) # Set up for 2 side-by-side plots
# Plot the first 25 runs of Y (so we can see what's happening)
plot(1:25, y.new[1:25], xlab="Simulation number", ylab="Y", ylim=c(2,8))
# Add vertical segments for the prediction intervals
segments(x0=1:25, x1=1:25, y0=pred.int["lwr"], y1=pred.int["upr"], lty="dashed")
# For each Y, check if it's covered by the interval
covered <- (y.new >= pred.int["lwr"]) & (y.new <= pred.int["upr"])
# Plot the running mean of the fraction of Y's covered by the interval
plot(1:m, cumsum(covered)/(1:m), xlab="Number of simulations",
     ylab="Cumulative coverage proportion", ylim=c(0.5,1))
abline(h=1-alpha, col="grey") # Theoretical coverage level
par(mfrow=c(1,1)) # Restore ordinary plot layout for later
```

FIGURE 2: *Demonstration of the coverage of the prediction intervals. Here, we are seeing what would happen if we got to use the true coefficients, which are $\beta_0 = 5$, $\beta_1 = -2$, $\sigma^2 = 0.1$; we are always trying to predict $Y$ when $X = 1/137$.*

Since $Y|X = x \sim N(m(x), \sigma^2)$, it would be a simple matter to find these limits if we knew the parameters: the lower limit would be $m(x) + z_{\alpha/2}\sigma$, and the upper limit $m(x) + z_{1-\alpha/2}\sigma$. Unfortunately, we don't know the parameters.

However, we do know how the parameters are related to our estimates, so let's try to use that:

$$\begin{aligned} Y|X = x &\sim N(m(x), \sigma^2) && (16)\\ &= m(x) + N(0, \sigma^2) && (17)\\ &= \widehat{m}(x) + N\left(0, \frac{\sigma^2}{n}\left(1 + \frac{(x-\overline{x})^2}{s_X^2}\right)\right) + N(0, \sigma^2) && (18)\\ &= \widehat{m}(x) + N\left(0, \sigma^2\left(1 + \frac{1}{n} + \frac{(x-\overline{x})^2}{ns_X^2}\right)\right) && (19)\end{aligned}$$

where in the last line I've used the fact that, under the assumptions of the model, the new $Y$ we're trying to predict is independent of the old $Y$ used to estimate the parameters. The variance, as we've seen, has two parts: the true noise variance about the regression line, plus the variance coming from our uncertainty in where that regression line is. Both parts of the variance are proportional to $\sigma^2$. Let's call the whole thing $\sigma^2_{pred}(x)$.

So, we have a random variable with a Gaussian distribution centered at $\widehat{m}(x)$ and with a variance $\sigma^2_{pred}(x)$ proportional to $\sigma^2$. We can estimate that variance as

$$s^2_{pred}(x) = \widehat{\sigma}^2 \frac{n}{n-2}\left(1 + \frac{1}{n} + \frac{(x-\overline{x})^2}{ns_X^2}\right) \tag{20}$$

Going through the now-familiar argument once again,

$$\frac{Y - \widehat{m}(x)}{s_{pred}(x)} \mid X = x \sim t_{n-2} \approx N(0,1) \tag{21}$$

and we can use this to give prediction intervals. The prediction interval is

$$C(x) = \widehat{m}(x) \pm z_{\alpha/2} s_{\mathrm{pred}}(x)$$

and we have that

$$P(Y \in C(x)|X = x) \approx 1 - \alpha.$$

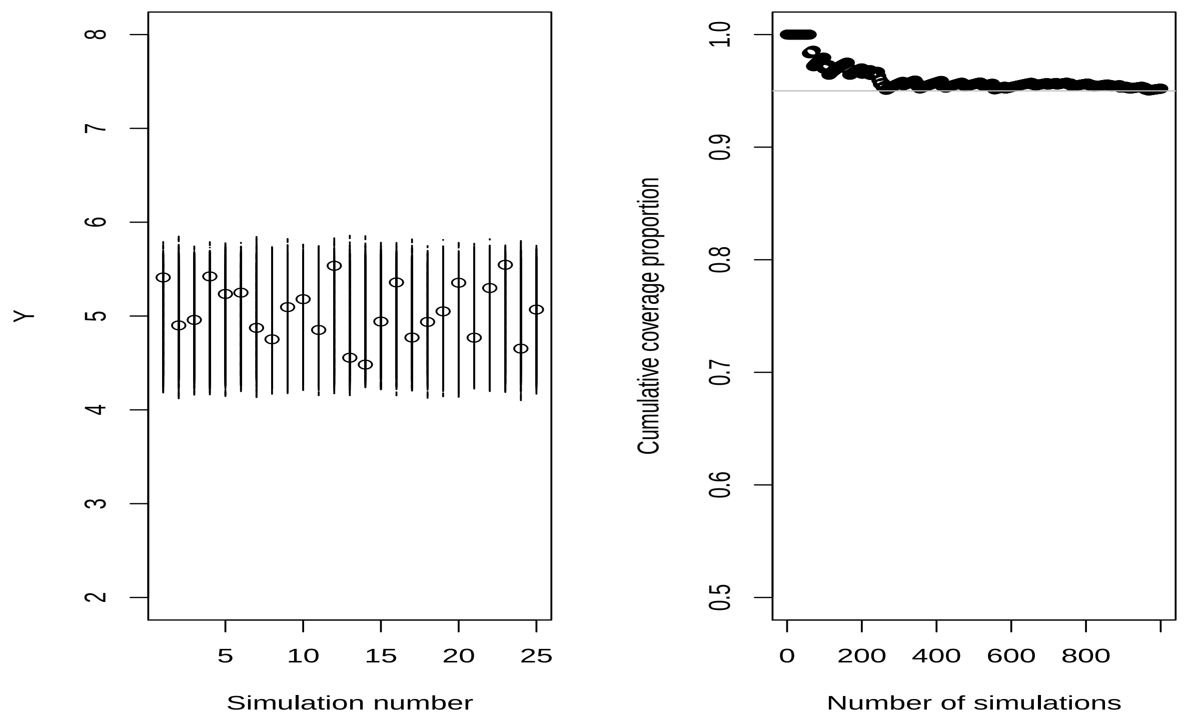
Again, as usual, as $n \to \infty$, the $t$ distribution turns into a standard Gaussian, while $s^2_{pred}(x) \to \sigma^2_{pred}(x) \to \sigma^2$. With *enough* data, then, our prediction intervals approach the ones we'd use if we knew the parameters and they were exactly our point estimates. Notice that the width of these prediction intervals does *not* go to zero as $n \to \infty$ — there is always some noise around the regression line!

## 2.1 Prediction intervals and transformations

Transforming the predictor variable raises no issues for prediction intervals. If we've transformed the response, though, we need to take account of it.

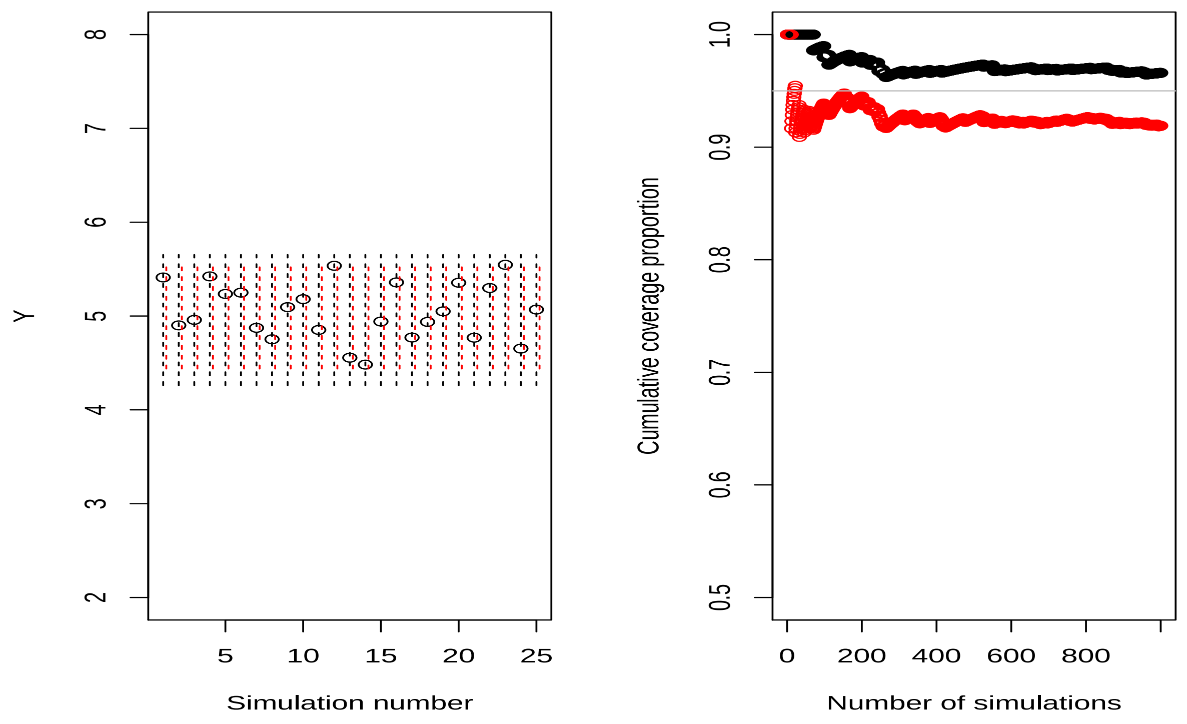
A model with a transformed response looks like this:

$$g(Y) = \beta_0 + \beta_1 X + \epsilon \tag{22}$$



```
# Run simulations where we get a new estimate of the model on each run,
# but with fixed X vector (to keep it simple)
x.seq <- seq(from=-5, to=5, length.out=42)
# Run the simulation many times, and give a _list_ of estimated models
  # simplify=FALSE forces the return value to be a list
mdls <- replicate(m,  sim.gnslrm(x=x.seq,beta.0,beta.1,sigma.sq,mdl=TRUE),
                  simplify=FALSE)
# Extract the prediction intervals for every one of the models
pred.ints <- sapply(mdls, extract.pred.int, x=x.new)
rownames(pred.ints)[2:3] <- names(pred.int) # Fix the names
# Now make plots like the previous figure
par(mfrow=c(1,2))
plot(1:25, y.new[1:25], xlab="Simulation number", ylab="Y", ylim=c(2,8))
segments(x0=1:25, x1=1:25, y0=pred.ints["lwr",], y1=pred.ints["upr",], lty="dashed")
covered <- (y.new >= pred.ints["lwr",]) & (y.new <= pred.ints["upr",])
plot(1:m, cumsum(covered)/(1:m), xlab="Number of simulations",
     ylab="Cumulative coverage proportion", ylim=c(0.5,1))
abline(h=1-alpha, col="grey")
par(mfrow=c(1,1))
```

FIGURE 3: *As in Figure 2, but we are now using coefficients estimated by drawing 42 observations from the model, with the X's being evenly spaced from −5 to 5. Here, as you can see from the code, each prediction is made on the basis of a* different *random realization of the data before estimating the model. (See §3 below for details on how to use* `predict` *to return intervals.)*



```
# What's the coverage if we use just one estimate of the model?
  # Pick the first two, arbitrarily, to show how this varies
# Get the prediction interval for our x.new
pred.ints <- sapply(mdls[1:2], extract.pred.int, x=x.new)
rownames(pred.ints)[2:3] <- c("lwr","upr")
# Make the plots
par(mfrow=c(1,2))
plot(1:25, y.new[1:25], xlab="Simulation number", ylab="Y", ylim=c(2,8))
segments(x0=1:25, x1=1:25, y0=pred.ints["lwr",1], y1=pred.ints["upr",1], lty="dashed")
# Slightly off-set one of the intervals for visibility
segments(x0=0.2+1:25, x1=0.2+1:25, y0=pred.ints["lwr",2], y1=pred.ints["upr",2],
         lty="dashed", col="red")
# Calculate two cumulative coverage proportions
covered.1 <- (y.new >= pred.ints["lwr",1]) & (y.new <= pred.ints["upr",1])
covered.2 <- (y.new >= pred.ints["lwr",2]) & (y.new <= pred.ints["upr",2])
plot(1:m, cumsum(covered.1)/(1:m), xlab="Number of simulations",
     ylab="Cumulative coverage proportion", ylim=c(0.5,1))
points(1:m, cumsum(covered.2)/(1:m), col="red")
abline(h=1-alpha, col="grey")
par(mfrow=c(1,1))
```

FIGURE 4: *As in Figure 3, but all the new realizations of $Y$ are being predicted based on the coefficients of one single estimate of the coefficients (the first estimate for the black intervals, the second estimate for the red). — The code for all three figures is very similar; could you write one function which, with appropriate arguments, would make all three of them?*

for $\epsilon$ IID Gaussian, and some invertible, non-linear function $g$. Since $g$ is invertible, it must be either increasing or decreasing; to be definite, I'll say it's increasing, but it should be clear as we go what needs to change for decreasing transformations.

When we estimated the model after transforming $Y$, what we have above gives us a prediction interval for $g(Y)$. Remember what this means:

$$\mathbb{P}\left(L \leq g(Y) \leq U | X = x\right) = 1 - \alpha \tag{23}$$

Since $g$ is an increasing function, so is $g^{-1}$, and therefore

$$\{L \leq g(Y) \leq U\} \Leftrightarrow \left\{g^{-1}(L) \leq Y \leq g^{-1}(U)\right\} \tag{24}$$

Since the two events are logically equivalent, they must have the same probability, no matter what we condition on:

$$\mathbb{P}\left(g^{-1}(L) \leq Y \leq g^{-1}(U) \mid X = x\right) = 1 - \alpha \tag{25}$$

Thus, we get a prediction interval for $Y$ by taking the prediction interval for $g(Y)$ and undoing the transformation.

## 3 Prediction intervals in R

For linear models, all of the calculations needed to find confidence intervals for $\widehat{m}$ or prediction intervals for $Y$ are automated into the `predict` function, introduced in Lecture 5.

```
predict(object, newdata, interval=c("none", "confidence", "prediction"), level=0.95)
```

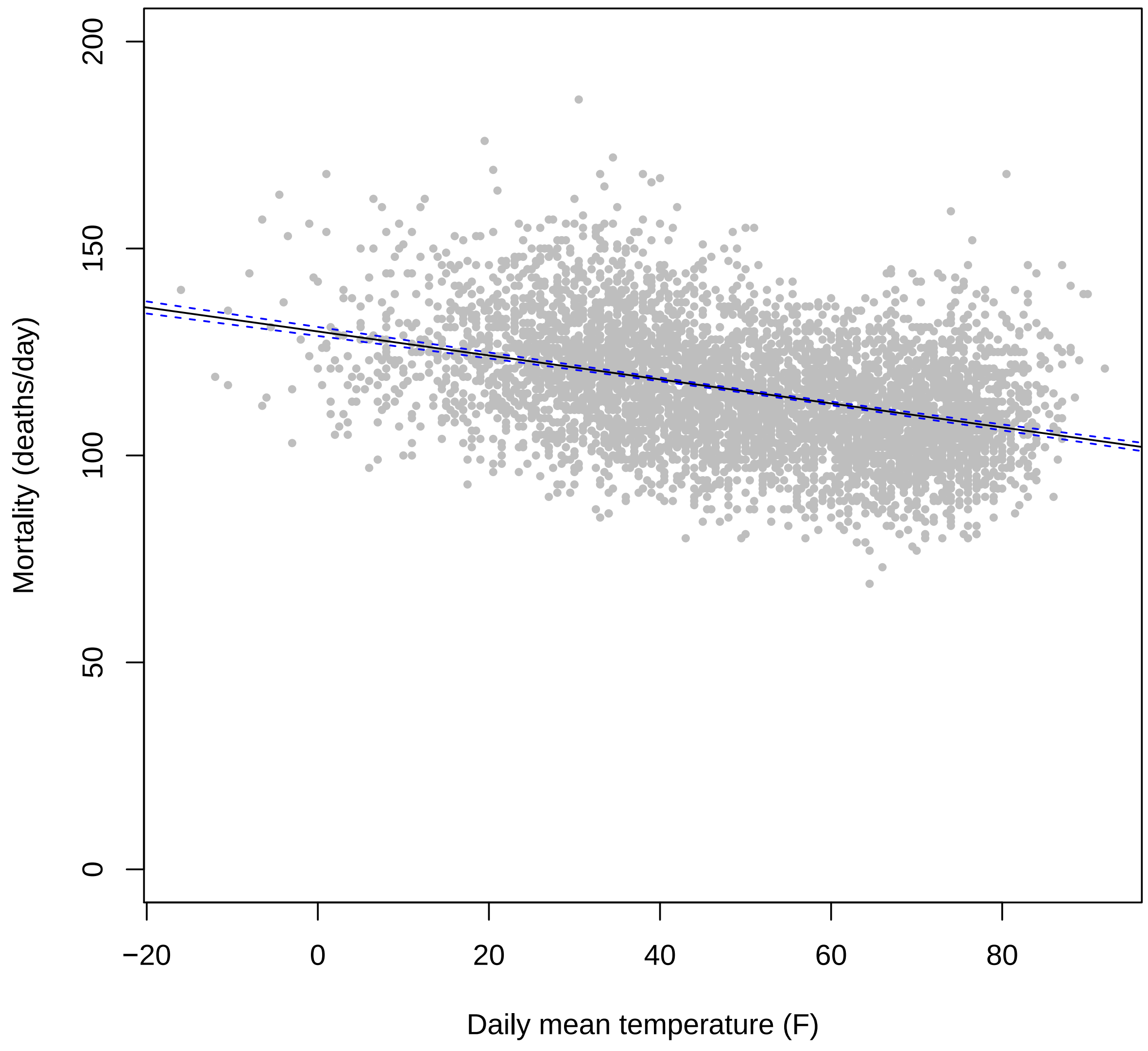
The `object` argument is the estimated model returned by `lm`; `newdata` is a data frame containing a column whose name matches that of the predictor variable. We saw these arguments before; what's new are the other two. `interval` controls whether to give point predictions (`"none"`, the default) or intervals, and if so which kind. `level` is of course the confidence level (default 0.95 for tradition's sake.)

To illustrate, let's revisit our old friend `chicago`:

```
library(gamair); data(chicago)
death.temp.lm <- lm(death ~ tmpd, data=chicago)
```

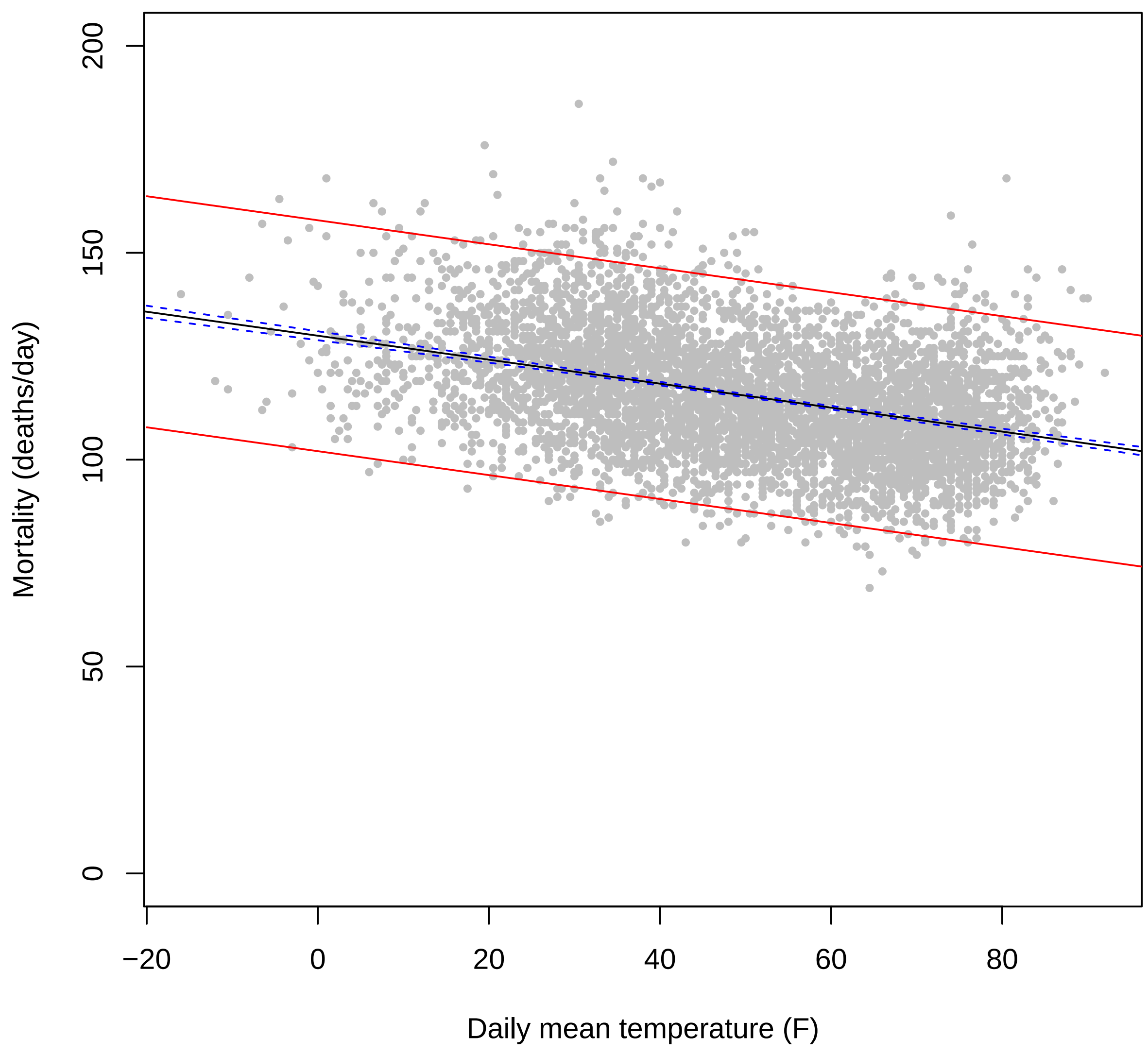
Figure 5 shows a scatter-plot of the data and the estimated line, together with confidence limits for the conditional mean at each point Because we have thousands of data points and reasonably large $s^2_X$, the confidence limits are quite narrow, though you can see, from the plot, how they widen as we move away from the mean temperature.

Figure 6 shows the *prediction* limits for the same model. These are much wider, because their width is mostly coming from (the estimate of) $\sigma$, the noise around the regression line, the model being very confident that it knows what the line is. Despite their width, the bands don't include all the data points. This is not, in itself, alarming — they should only contain about 95% of the data points! I will leave it as an exercise to check what the actual coverage level is here.



```
plot(death ~ tmpd, data=chicago, pch=19, cex=0.5, col="grey", ylim=c(0,200),
     xlab="Daily mean temperature (F)", ylab="Mortality (deaths/day)")
abline(death.temp.lm)
temp.seq <- seq(from=-20, to=100, length.out=100)
death.temp.CIs <- predict(death.temp.lm, newdata=data.frame(tmpd=temp.seq),
                          interval="confidence")
lines(temp.seq, death.temp.CIs[,"lwr"], lty="dashed", col="blue")
lines(temp.seq, death.temp.CIs[,"upr"], lty="dashed", col="blue")
```

FIGURE 5: *Data from the Chicago death example (grey dots), together with the regression line (solid black) and the 95% confidence limits on the conditional mean (dashed blue curves). I have restricted the vertical range to help show the confidence limits, though this means some high-mortality days are off-screen.*



```
plot(death ~ tmpd, data=chicago, pch=19, cex=0.5, col="grey", ylim=c(0,200),
     xlab="Daily mean temperature (F)", ylab="Mortality (deaths/day)")
abline(death.temp.lm)
temp.seq <- seq(from=-20, to=100, length.out=100)
death.temp.CIs <- predict(death.temp.lm, newdata=data.frame(tmpd=temp.seq),
                            interval="confidence")
lines(temp.seq, death.temp.CIs[,"lwr"], lty="dashed", col="blue")
lines(temp.seq, death.temp.CIs[,"upr"], lty="dashed", col="blue")
death.temp.PIs <- predict(death.temp.lm, newdata=data.frame(tmpd=temp.seq),
                          interval="prediction")
lines(temp.seq, death.temp.PIs[,"lwr"], col="red")
lines(temp.seq, death.temp.PIs[,"upr"], col="red")
```

FIGURE 6: *Adding 95% prediction intervals (red) to the previous plot.*