There are a few snags to this, however: (a) The `blood.glucose` values are in random order; we do not want line segments connecting points haphazardly along the confidence curves; (b) the prediction limits, particularly the lower one, extend outside the plot region; and (c) the `matlines` command needs to be prevented from cycling through line styles and colours. Notice that the `na.exclude` setting (p. 115) prevents us from also having an observation omitted from the predicted values.

The solution is to *predict in a new data frame* containing suitable x values (here `blood.glucose`) at which to predict. It is done as follows:

```
> pred.frame <- data.frame(blood.glucose=4:20)
> pp <- predict(lm.velo, int="p", newdata=pred.frame)
> pc <- predict(lm.velo, int="c", newdata=pred.frame)
> plot(blood.glucose,short.velocity,
+      ylim=range(short.velocity, pp, na.rm=T))
> pred.gluc <- pred.frame$blood.glucose
> matlines(pred.gluc, pc, lty=c(1,2,2), col="black")
> matlines(pred.gluc, pp, lty=c(1,3,3), col="black")
```

What happens is that we create a new data frame in which the variable `blood.glucose` contains the values at which we want predictions to be made. `pp` and `pc` are then made to contain the result of `predict` for the new data in `pred.frame` with prediction limits and confidence limits, respectively.

For the plotting, we first create a standard scatterplot, except that we ensure that it has enough room for the prediction limits. This is obtained by setting `ylim=range(short.velocity, pp, na.rm=T)`. The function `range` returns a vector of length 2 containing the minimum and maximum values of its arguments. We need the `na.rm=T` argument to cause missing values to be skipped for the range computation; notice that `short.velocity` is included to ensure that points outside the prediction limits are not missed (although in this case there are none). Finally, the curves are added, using as x-values the `blood.glucose` used for the prediction and setting the line types and colours to more sensible values. The final result is seen in Figure 6.5.

6.4 Correlation

A correlation coefficient is a symmetric, scale-invariant measure of association between two random variables. It ranges from -1 to $+1$, where the extremes indicate perfect correlation and 0 means no correlation. The sign is negative when large values of one variable are associated with small values of the other and positive if both variables tend to be large or small

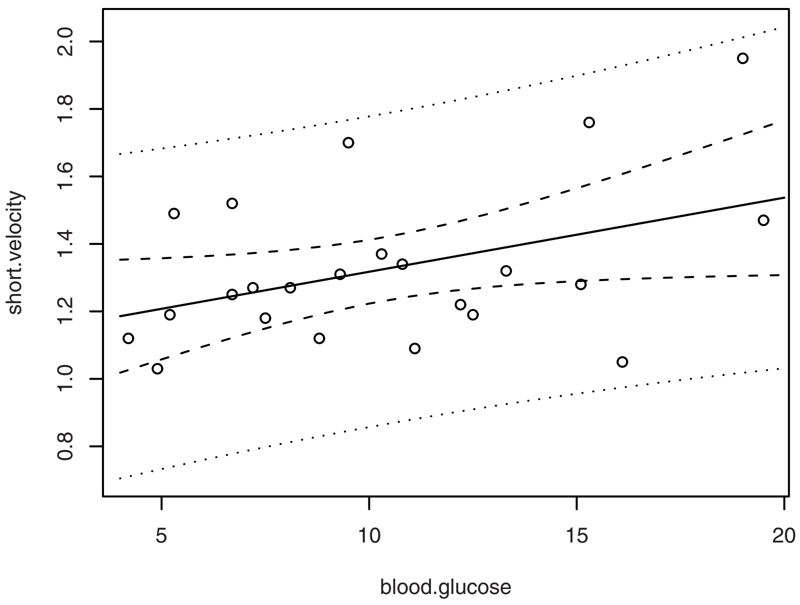

Figure 6.5. Plot with confidence and prediction bands.

simultaneously. The reader should be warned that there are many incorrect uses of correlation coefficients, particularly when they are used in regression-type settings.

This section describes the computation of parametric and nonparametric correlation measures in R.

6.4.1 *Pearson correlation*

The Pearson correlation is rooted in the two-dimensional normal distribution where the theoretical correlation describes the contour ellipses for the density. If both variables are scaled to have a variance of 1, then a correlation of zero corresponds to circular contours, whereas the ellipses become narrower and finally collapse into a line segment as the correlation approaches ± 1.

The empirical correlation coefficient is

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

It can be shown that $|r|$ will be less than 1 unless there is a perfect linear relation between x_i and y_i, and for that reason the Pearson correlation is sometimes called the "linear correlation".

It is possible to test the significance of the correlation by transforming it to a t-distributed variable (the formula is not particularly elucidating so we skip it here), which will be identical with the test obtained from testing the significance of the slope of either the regression of y on x or vice versa.

The function `cor` can be used to compute the correlation between two or more vectors. However, if it is naively applied to the two vectors in `thuesen`, the following happens:

```
> cor(blood.glucose,short.velocity)
Error in cor(blood.glucose, short.velocity) :
        missing observations in cov/cor
```

All the elementary statistical functions in R require either that all values be nonmissing or that you explicitly state what should be done with the cases with missing values. For `mean`, `var`, `sd`, and similar one-vector functions, you can give the argument `na.rm=T` to indicate that missing values should be removed before the computation. For `cor`, you can write

```
> cor(blood.glucose,short.velocity,use="complete.obs")
[1] 0.4167546
```

The reason that `cor` does not use `na.rm=T` like the other functions is that there are more possibilities than just removing incomplete cases or failing. If more than two variables are in play, it is also possible to use information from all nonmissing *pairs* of measurements (this might result in a correlation matrix that is not positive definite, though).

You can obtain the entire matrix of correlations between all variables in a data frame by saying, for instance,

```
> cor(thuesen,use="complete.obs")
               blood.glucose short.velocity
blood.glucose      1.0000000      0.4167546
short.velocity     0.4167546      1.0000000
```

Of course, this is more interesting when the data frame contains more than two vectors!

However, the calculations above give no indication of whether the correlation is significantly different from zero. To that end, you need `cor.test`. It works simply by specifying the two variables:

```
> cor.test(blood.glucose,short.velocity)

        Pearson's product-moment correlation

data:  blood.glucose and short.velocity
t = 2.101, df = 21, p-value = 0.0479
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.005496682 0.707429479
sample estimates:
      cor
0.4167546
```

We also get a confidence interval for the true correlation. Notice that it is exactly the same *p*-value as in the regression analysis in Section 6.1 and also that based on the ANOVA table for the regression model, which is described in Section 7.5.

6.4.2 *Spearman's ρ*

As with the one- and two-sample problems, you may be interested in nonparametric variants. These have the advantage of not depending on the normal distribution and, indeed, being invariant to monotone transformations of the coordinates. The main disadvantage is that its interpretation is not quite clear. A popular and simple choice is Spearman's rank correlation coefficient ρ. This is obtained quite simply by replacing the observations by their rank and computing the correlation. Under the null hypothesis of independence between the two variables, the exact distribution of ρ can be calculated.

Unlike group comparisons where there is essentially one function per named test, correlation tests are all grouped into `cor.test`. There is no special `spearman.test` function. Instead, the test is considered one of several possibilities for testing correlations and is therefore specified via an option to `cor.test`:

```
> cor.test(blood.glucose,short.velocity,method="spearman")

        Spearman's rank correlation rho

data:  blood.glucose and short.velocity
S = 1380.364, p-value = 0.1392
alternative hypothesis: true rho is not equal to 0
sample estimates:
     rho
0.318002

Warning message:
```

```
In cor.test.default(blood.glucose, short.velocity, method="spearman"):
  Cannot compute exact p-values with ties
```

6.4.3 Kendall's τ

The third correlation method that you can choose is Kendall's τ, which is based on counting the number of *concordant* and *discordant* pairs. A pair of points is concordant if the difference in the x-coordinate is of the same sign as the difference in the y-coordinate. For a perfect monotone relation, either all pairs will be concordant or all pairs will be discordant. Under independence, there should be as many concordant pairs as there are discordant ones.

Since there are many pairs of points to check, this is quite a computationally intensive procedure compared with the two others. In small data sets such as the present one, it does not matter at all, though, and the procedure is generally usable up to at least 5000 observations.

The τ coefficient has the advantage of a more direct interpretation over Spearman's ρ, but apart from that there is little reason to prefer one over the other.

```
> cor.test(blood.glucose,short.velocity,method="kendall")

         Kendall's rank correlation tau

data:  blood.glucose and short.velocity
z = 1.5604, p-value = 0.1187
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.2350616

Warning message:
In cor.test.default(blood.glucose, short.velocity, method="kendall"):
  Cannot compute exact p-value with ties
```

Notice that neither of the two nonparametric correlations is significant at the 5% level, which the Pearson correlation is, albeit only borderline significant.

6.5 Exercises

6.1 With the `rmr` data set, plot metabolic rate versus body weight. Fit a linear regression model to the relation. According to the fitted model, what is the predicted metabolic rate for a body weight of 70 kg? Give a 95% confidence interval for the slope of the line.

6.2 In the `juul` data set, fit a linear regression model for the square root of the IGF-I concentration versus age to the group of subjects over 25 years old.

6.3 In the `malaria` data set, analyze the log-transformed antibody level versus age. Make a plot of the relation. Do you notice anything peculiar?

6.4 One can generate simulated data from the two-dimensional normal distribution with a correlation of ρ by the following technique: (a) Generate X as a normal variate with mean 0 and standard deviation 1; (b) generate Y with mean ρX and standard deviation $\sqrt{1-\rho^2}$. Use this to create scatterplots of simulated data with a given correlation. Compute the Spearman and Kendall statistics for some of these data sets.