Chapter 5

Hypervariate data

5.1 Four variables

READ: Cleveland pp. 272–292.

5.1.1 Environmental

Load the usual:

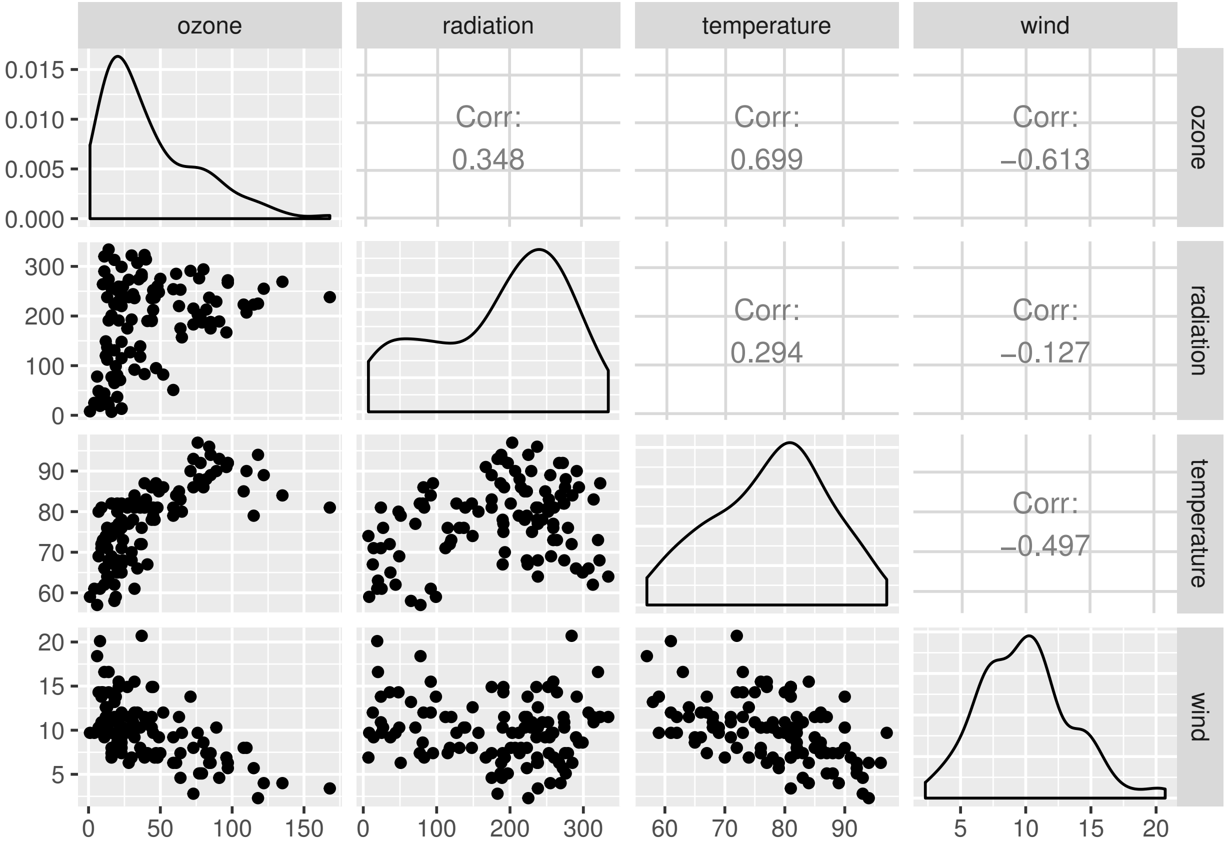
```r
load("lattice.RData")
library(ggplot2)
library(GGally)
library(broom)
library(tidyr)
```

The `environmental` data set contains four variables measured in New York on 111 days from May to September 1973:

- `ozone` in parts per billion;
- `radiation` in Langleys;
- `temperature` in degree Fahrenheit;
- `wind` in miles per hour.

We previously looked at the bivariate distribution between temperature and wind speed. Now we'll look at radiation, temperature, and wind as predictors of ozone.

```r
ggpairs(environmental)
```

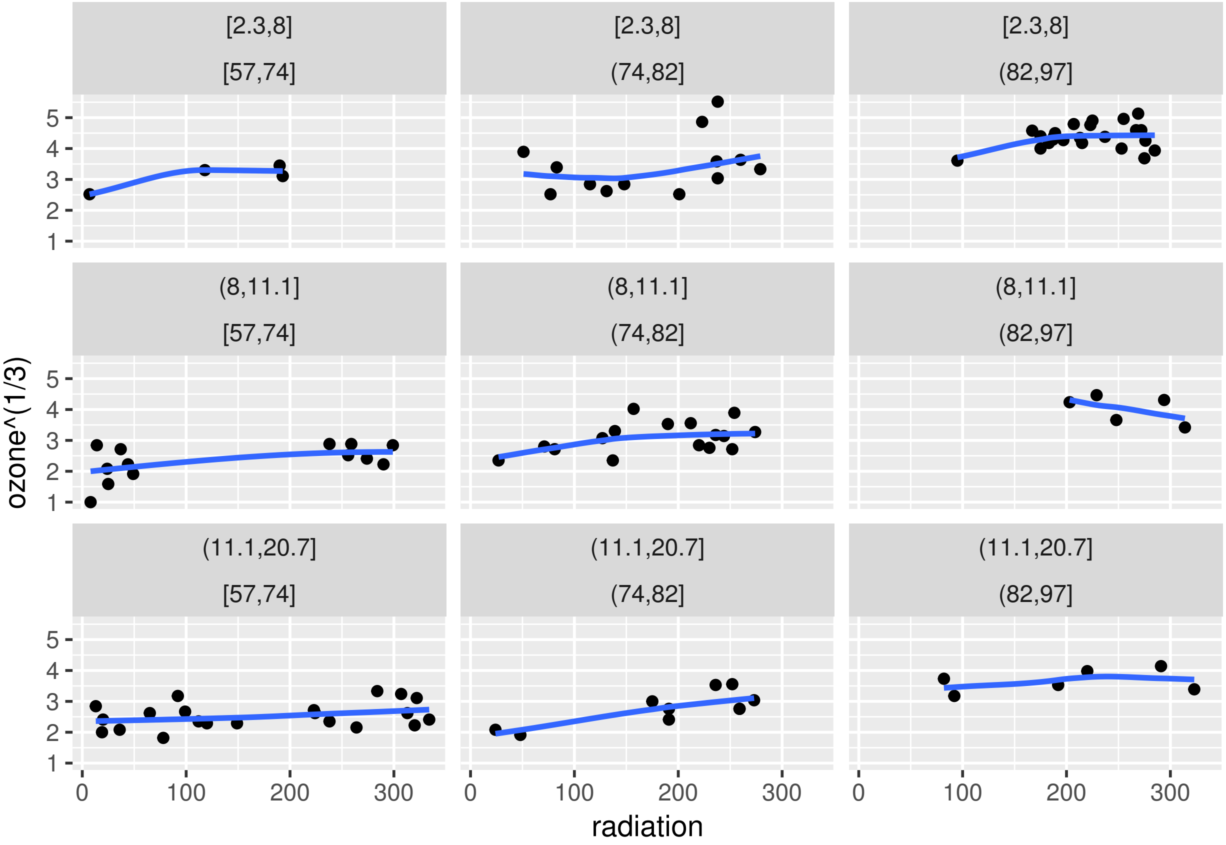


The paired plots showed that ozone is related to all three of the other variables. Since it doesn't look like any of the other variables are strongly collinear, we keep all three predictors. The density plot of ozone shows strong right skew, so some transformation will probably be beneficial. After some experimentation (not shown here), a cube root transformation seems reasonable (if somewhat hard to interpret.)

5.1.2 Faceting the data

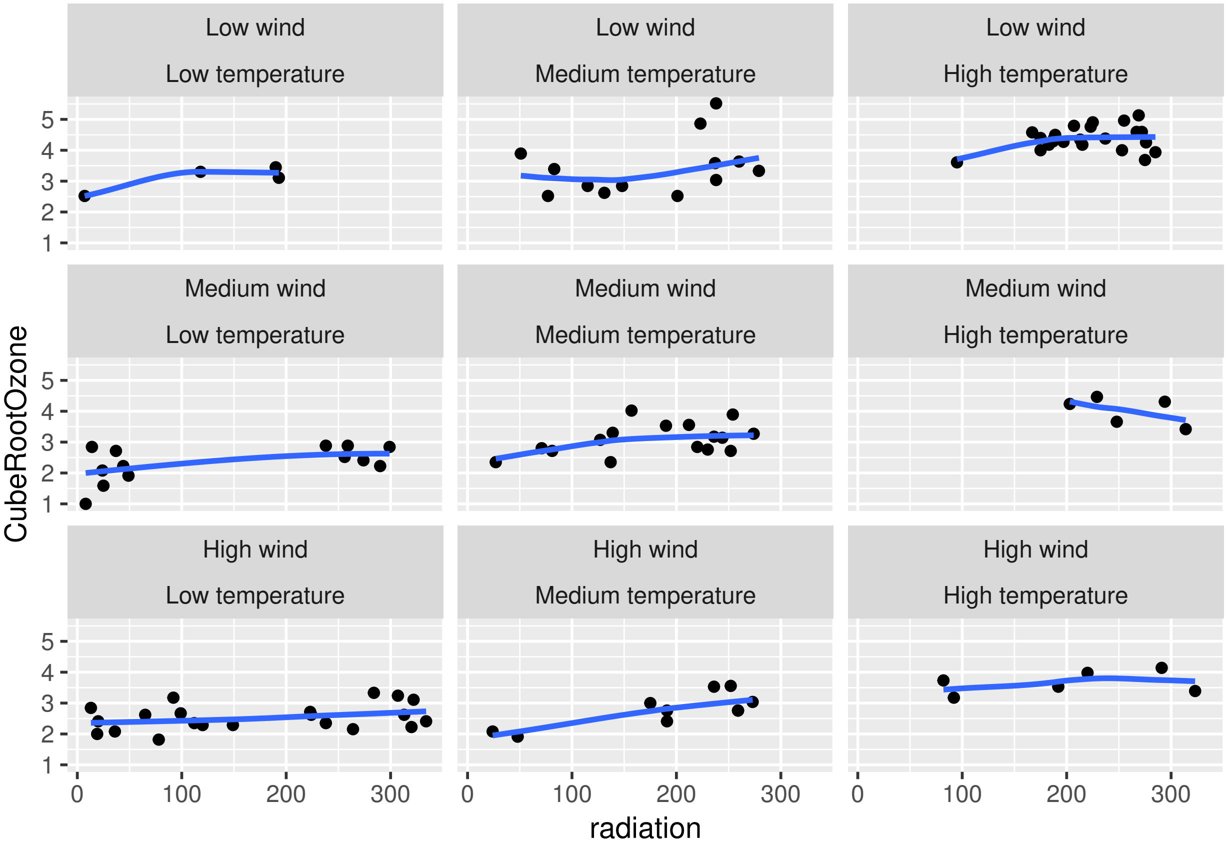
To consider all four variables at once, we now have to facet two ways. That is, pick one predictor variable (here, radiation), then cut the other two predictors into categories and create a grid of faceted plots. In the grid below, left to right gives low to high temperature, while top to bottom gives low to high wind.

```
ggplot(environmental, aes(x = radiation, y = ozone^(1/3))) + geom_point() +
    geom_smooth(span = 1, method.args = list(degree = 1, family = "symmetric"),
        se = FALSE) + facet_wrap(~cut_number(wind, n = 3) + cut_number(temperature,
    n = 3))
```



This is getting complicated, so it would be nice to give names to the categories to keep track of what's going on. We do that in the code below. Note that it could be more efficient to write a function to do the naming.

```
CubeRootOzone = environmental$ozone^(1/3)
radiation = environmental$radiation
temperature = environmental$temperature
wind = environmental$wind
radiation.cat = rep(NA, nrow(environmental))
radiation.cat[radiation <= quantile(radiation, 1/3)] = "Low radiation"
radiation.cat[radiation > quantile(radiation, 1/3) & radiation <= quantile(radiation,
    2/3)] = "Medium radiation"
radiation.cat[radiation > quantile(radiation, 2/3)] = "High radiation"
radiation.cat = factor(radiation.cat, levels = c("Low radiation", "Medium radiation",
    "High radiation"))
temperature.cat = rep(NA, nrow(environmental))
temperature.cat[temperature <= quantile(temperature, 1/3)] = "Low temperature"
temperature.cat[temperature > quantile(temperature, 1/3) & temperature <= quantile(temperature,
    2/3)] = "Medium temperature"
temperature.cat[temperature > quantile(temperature, 2/3)] = "High temperature"
temperature.cat = factor(temperature.cat, levels = c("Low temperature", "Medium temperature",
    "High temperature"))
wind.cat = rep(NA, nrow(environmental))
wind.cat[wind <= quantile(wind, 1/3)] = "Low wind"
wind.cat[wind > quantile(wind, 1/3) & wind <= quantile(wind, 2/3)] = "Medium wind"
wind.cat[wind > quantile(wind, 2/3)] = "High wind"
wind.cat = factor(wind.cat, levels = c("Low wind", "Medium wind", "High wind"))
environmental.cat = data.frame(environmental, CubeRootOzone, radiation.cat,
    temperature.cat, wind.cat)
ggplot(environmental.cat, aes(x = radiation, y = CubeRootOzone)) + geom_point() +
    geom_smooth(span = 1, method.args = list(degree = 1, family = "symmetric"),
        se = FALSE) + facet_wrap(~wind.cat + temperature.cat)
```

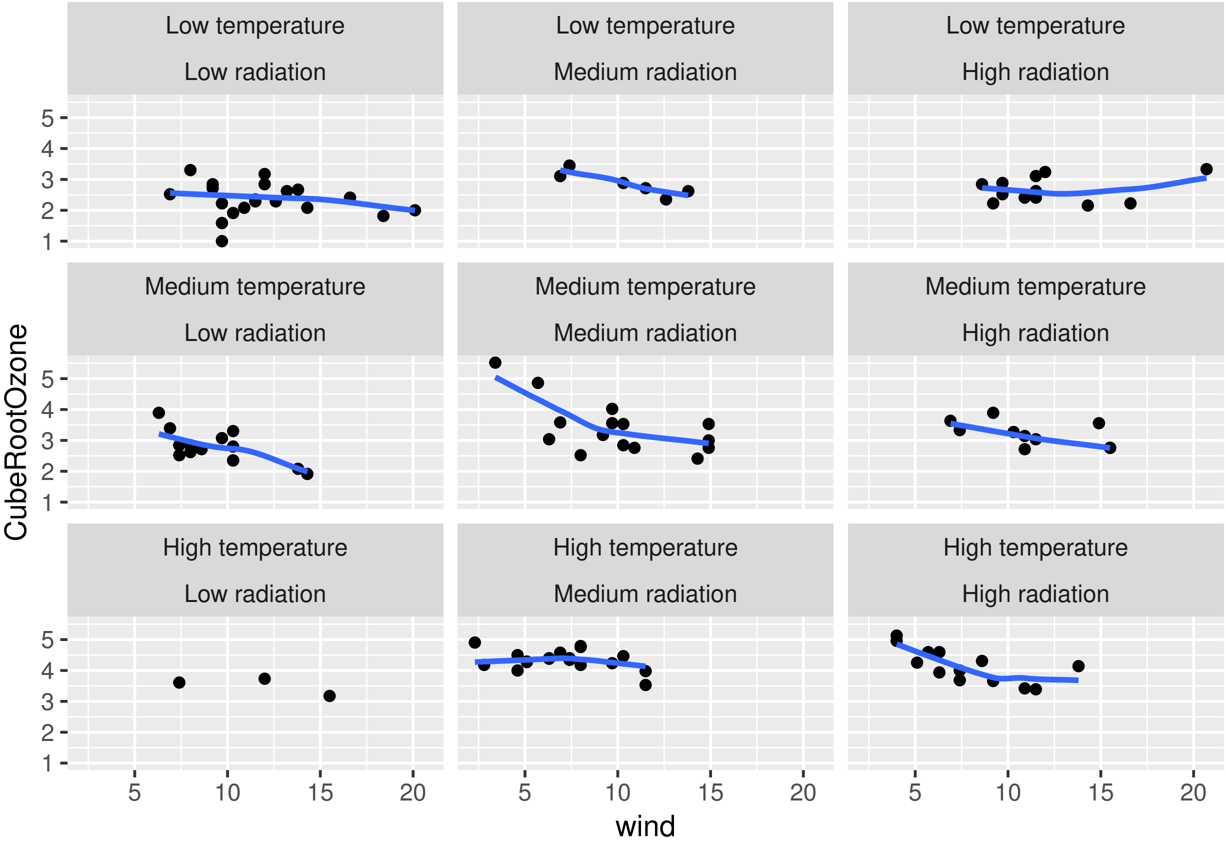


Ozone tends to increase with radiation. The levels of the fits, however, also change from top to bottom and from left to right, so the other variables will help predict as well.

Now make wind the main x-variable and facet on temperature and radiation.

```
ggplot(environmental.cat, aes(x = wind, y = CubeRootOzone)) + geom_point() +
    geom_smooth(span = 1, method.args = list(degree = 1, family = "symmetric"),
        se = FALSE) + facet_wrap(~temperature.cat + radiation.cat)
```

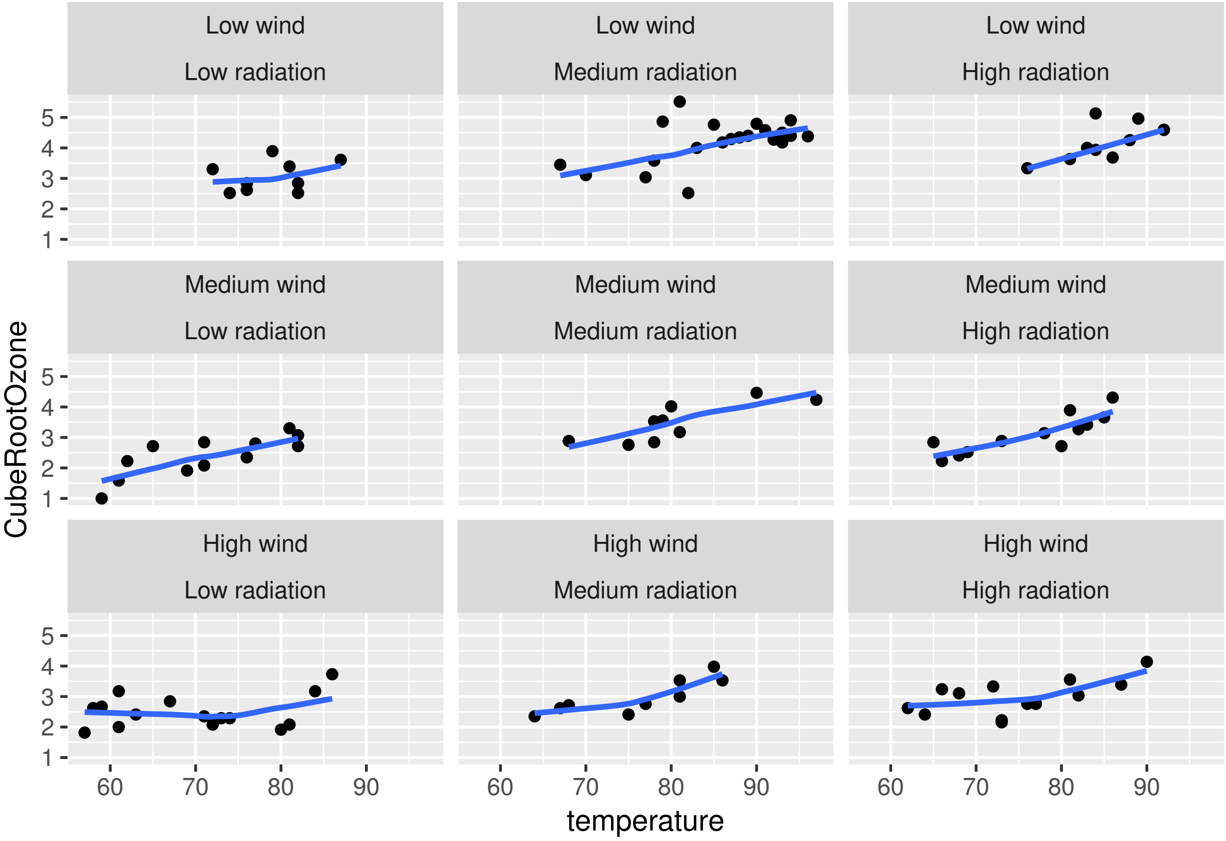
```
## Warning: Computation failed in `stat_smooth()`:
## NA/NaN/Inf in foreign function call (arg 1)
```



For once the error message is important. The high temperature, low radiation plot only has three observations. We make a mental note to *not* use the model for high temperature and low radiation situations later on because of the sparsity of data. Other than that, we see the relationship between ozone and wind is generally negative, though the slopes vary a lot.

Finally, make temperature the main x-variable and facet on wind and radiation.

```
ggplot(environmental.cat, aes(x = temperature, y = CubeRootOzone)) + geom_point() +
    geom_smooth(span = 1, method.args = list(degree = 1, family = "symmetric"),
        se = FALSE) + facet_wrap(~wind.cat + radiation.cat)
```



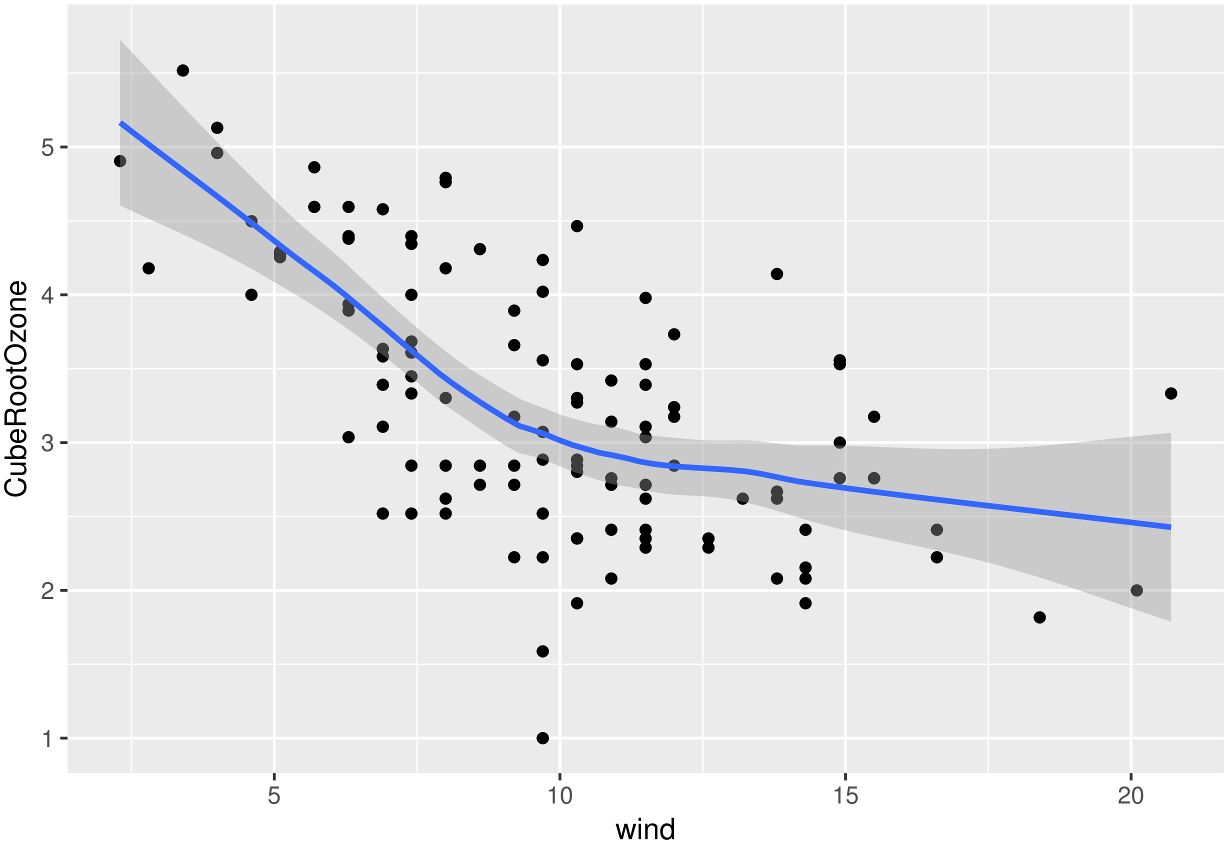
The slopes are pretty positive and consistent this time. Higher temperatures are associated with higher ozone.

5.1.3 Where is there no data?

Whenever the number of explanatory variables starts to get high, the **curse of dimensionality** becomes a concern: there may be large regions of predictor space where there's little or no data.

We're used to this idea with one predictor:

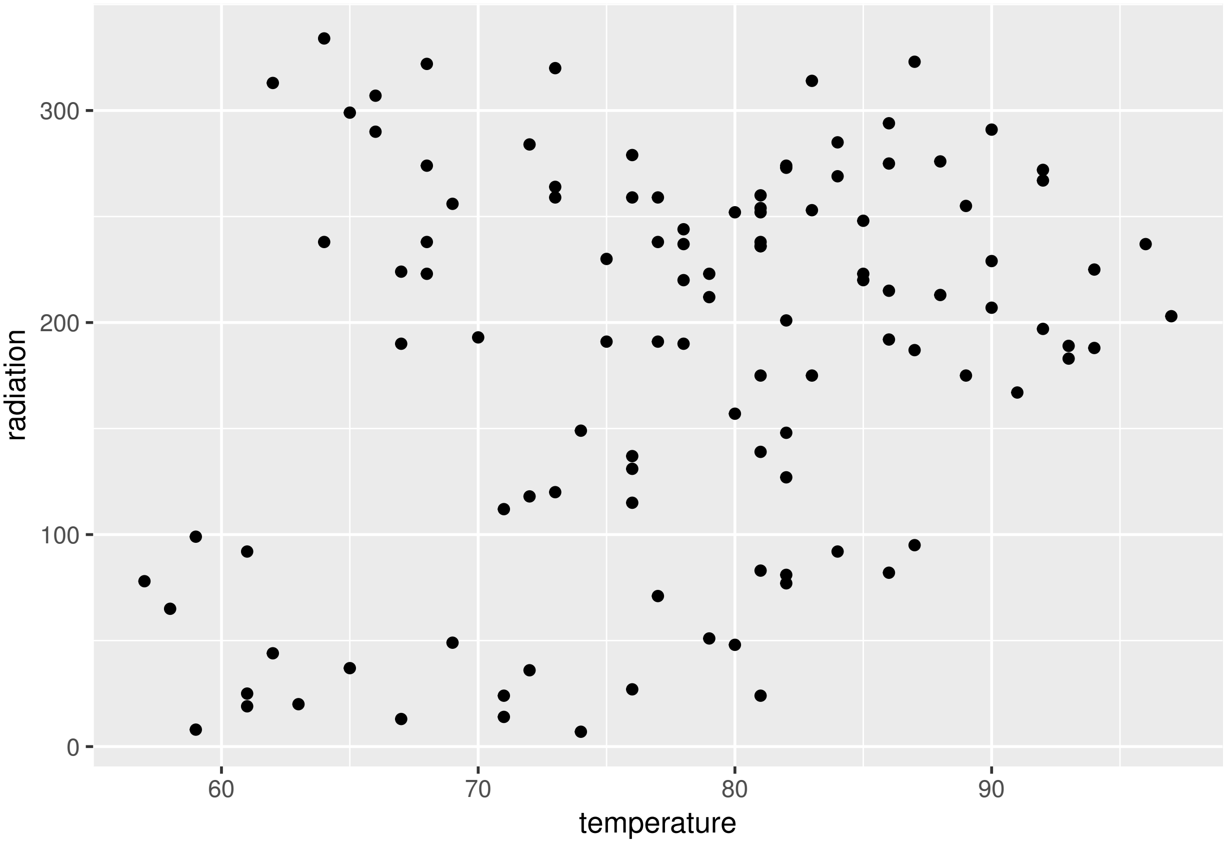
```
ggplot(environmental, aes(x = wind, y = CubeRootOzone)) + geom_point() + geom_smooth(span = 2/3,
    method.args = list(degree = 1))
```



As we go toward the extreme right, the confidence band gets wider, and then ggplot refuses to make any predictions to the right of the highest wind value – that would be extrapolation. We wouldn't want to use our model to predict the ozone concentration if the wind were, say, 35 miles per hour.

The same idea holds when there's more than one predictor, except the problem can be more subtle. Let's do a scatterplot of radiation against temperature.

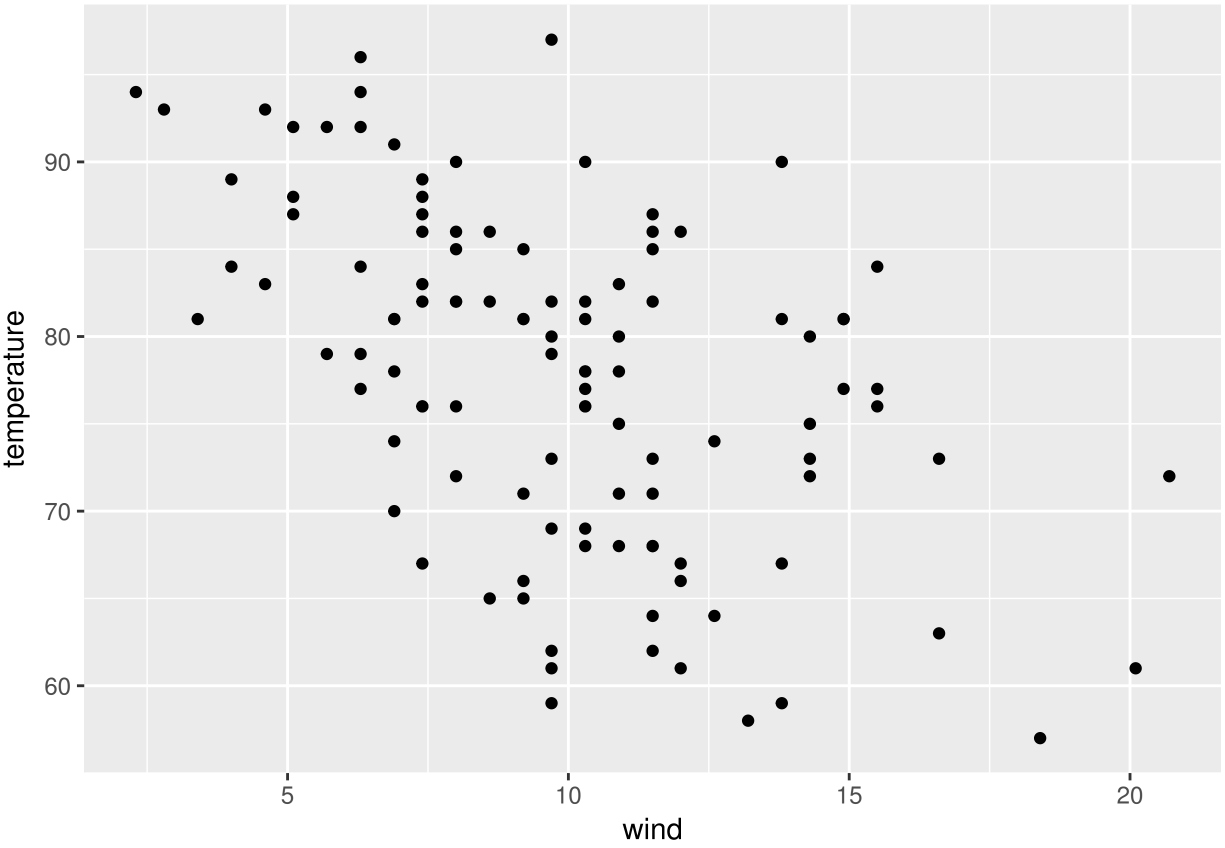
```
ggplot(environmental.cat, aes(x = temperature, y = radiation)) + geom_point()
```



Notice there are no observations in the bottom right corner. (We hinted at this when we faceted on both temperature and radiation above.) On the other hand, this is a fairly small corner of predictor space, so we might be able to ignore this.

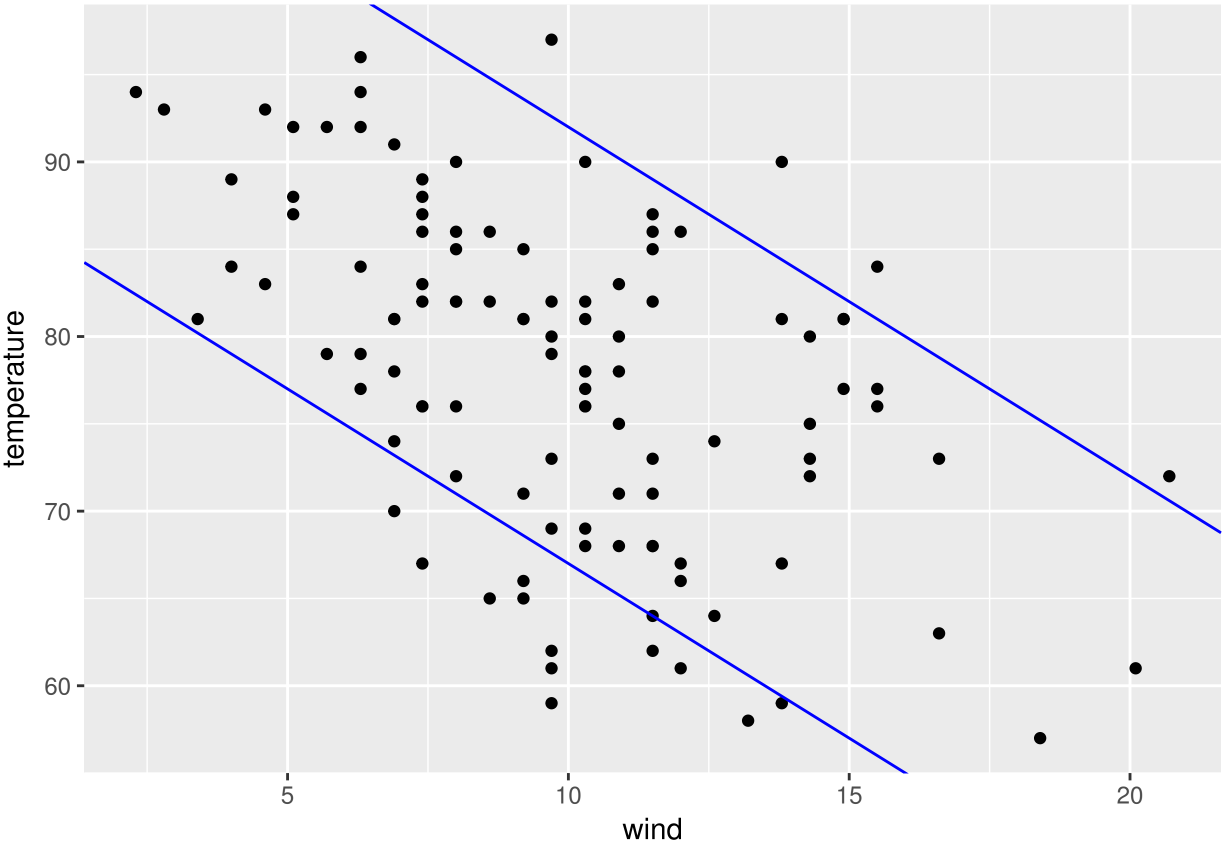
Now plot temperature against wind.

```
ggplot(environmental.cat, aes(x = wind, y = temperature)) + geom_point()
```



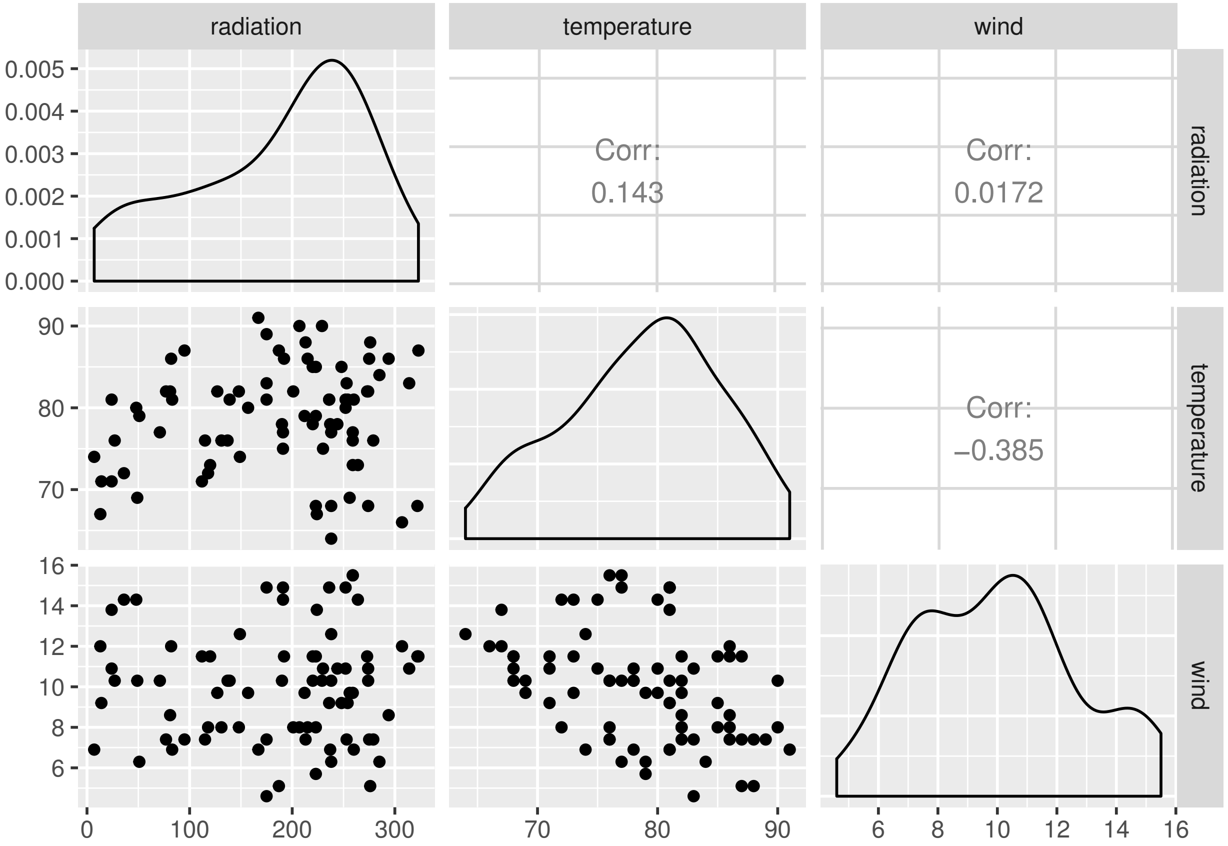
This is potentially a greater concern, because the vast majority of the data falls within a fairly narrow diagonal band of predictor space. Eyeballing the graph, it looks like for each value of wind, there's a range of about 25 degrees of temperature for which observations are dense. We add lines to the plot:

```
ggplot(environmental.cat, aes(x = wind, y = temperature)) + geom_point() + geom_abline(intercept = 112,
    slope = -2, color = "blue") + geom_abline(intercept = 87, slope = -2, color = "blue")
```



We shouldn't put too much creedence in our model outside the parallel lines. In addition, data is generally sparse outside temperatures of 60 to about 92 degrees and wind speeds outside 4 to 16 mph. We can *crop* predictor space and see if there are still "holes" in our data.

```
crop = (wind > 4) & (wind < 16) & (temperature < (-2 * wind + 112)) & (temperature >
    (-2 * wind + 87)) & (temperature > 60) & (temperature < 92)
environmental.crop = environmental.cat[crop, ]
ggpairs(environmental.crop, columns = 2:4)
```



Again, the main concern is the low radiation-high temperature corner, but it doesn't look that bad in context. We'll ignore it.

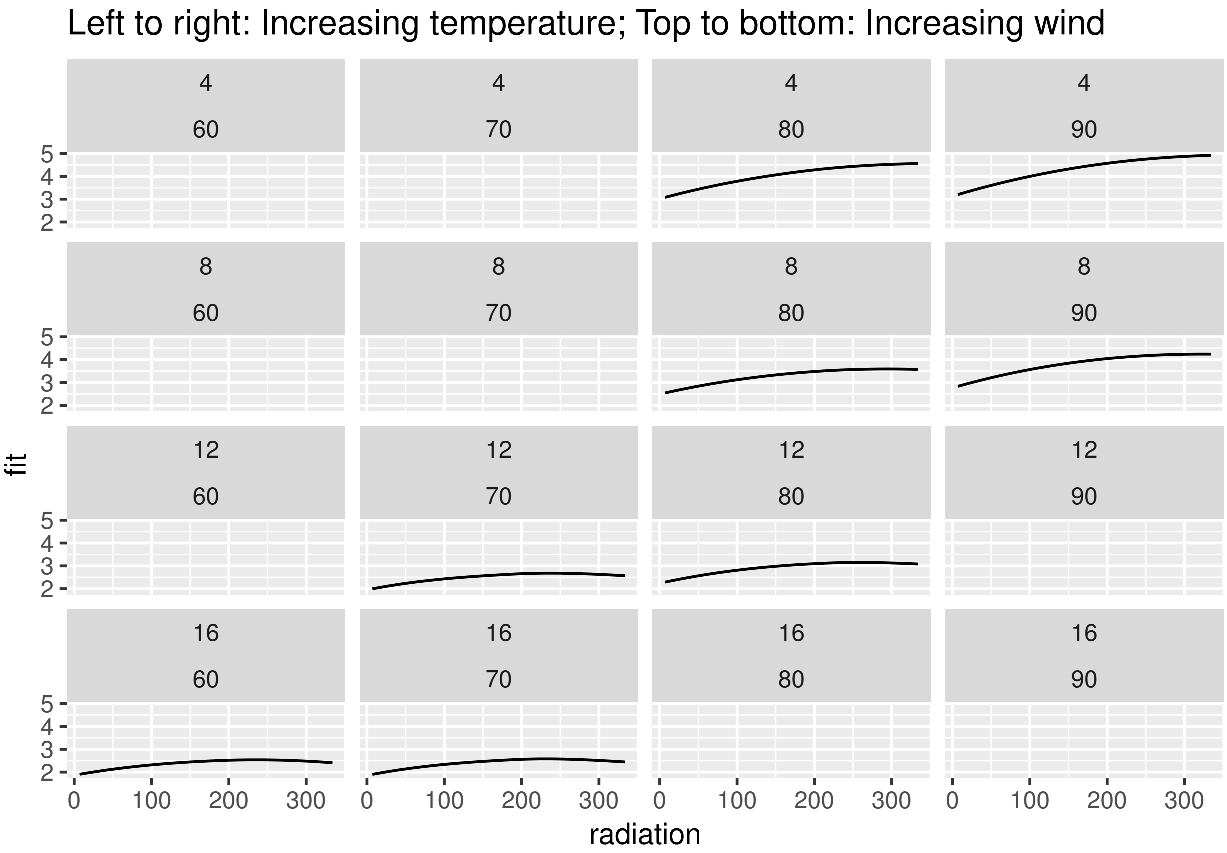
5.1.4 Visualizing the fit

The advantage of reducing the predictor space is that it's safer to fit a model with interactions. We first try a nonparametric loess model with a high `span` to reduce the risk of overfitting. ~~We include all two-way interactions, but not the three-way, since that's very hard to fit with limited data, let alone understand.~~ Nope, it's too hard to omit the three-way, so just throw it in there. There didn't seem to be any horrible outliers, so we fit using least squares.

```
environmental.lo = loess(CubeRootOzone ~ radiation * temperature * wind, span = 1)
```

Construct a grid and graph the relationship of fit to temperature, faceting on the other variables.

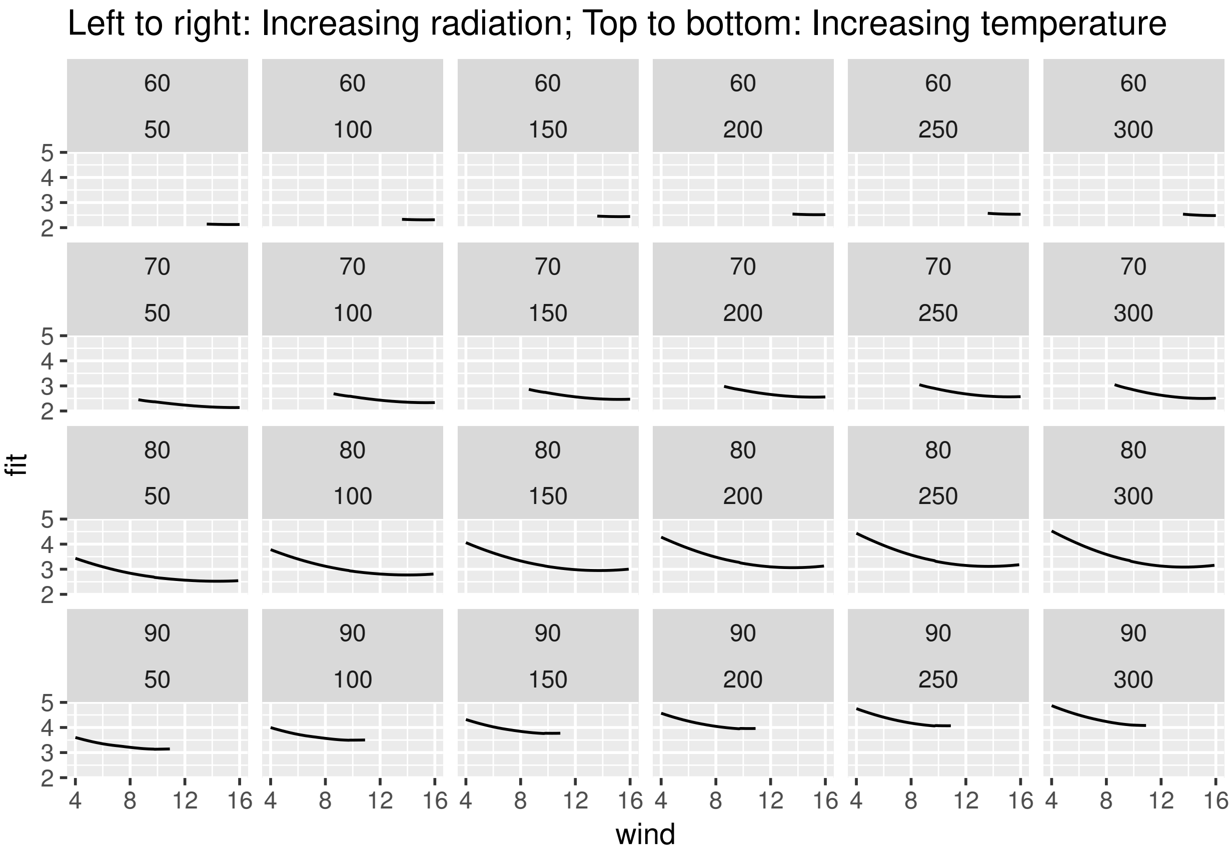
```
loess.grid = expand.grid(radiation = min(radiation):max(radiation), temperature = c(60,
    70, 80, 90), wind = c(4, 8, 12, 16))
environmental.predict = predict(environmental.lo, newdata = loess.grid)
environmental.df = data.frame(loess.grid, fit = as.vector(environmental.predict))
crop.grid = (loess.grid$temperature < (-2 * loess.grid$wind + 112)) & (loess.grid$temperature >
    (-2 * loess.grid$wind + 87))
environmental.df = environmental.df[crop.grid, ]
ggplot(environmental.df, aes(x = radiation, y = fit)) + geom_line() + facet_wrap(~wind +
    temperature, drop = FALSE) + labs(title = "Left to right: Increasing temperature; Top to bottom: Inc
```



The curves do seem to change slope going downward, justifying the radiation:wind interaction.

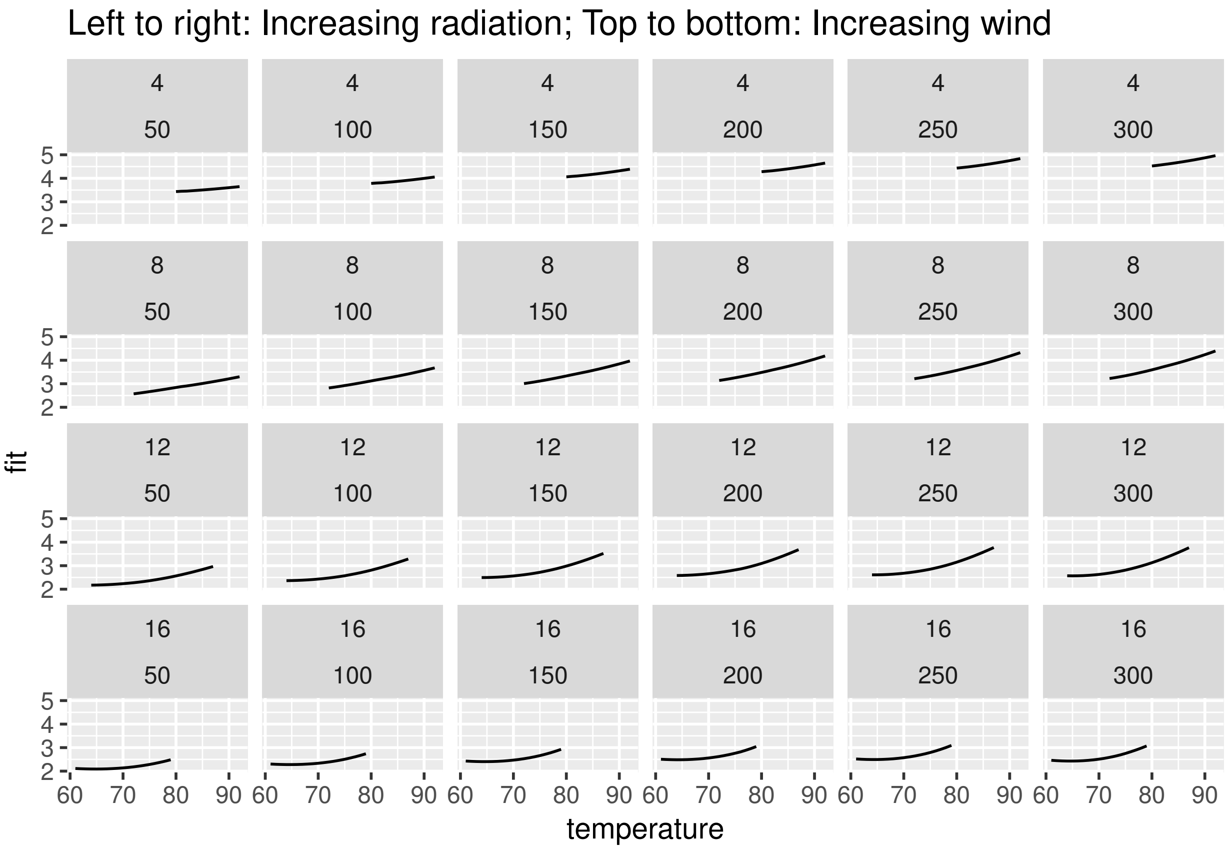
Now graph the relationship of fit with wind:

```
wind.grid = expand.grid(radiation = seq(50, 300, 50), temperature = c(60, 70,
    80, 90), wind = seq(4, 16, 0.1))
environmental.predict = predict(environmental.lo, newdata = wind.grid)
environmental.df = data.frame(wind.grid, fit = as.vector(environmental.predict))
wind.crop = (wind.grid$temperature < (-2 * wind.grid$wind + 112)) & (wind.grid$temperature >
    (-2 * wind.grid$wind + 87))
environmental.df = environmental.df[wind.crop, ]
ggplot(environmental.df, aes(x = wind, y = fit)) + geom_line() + facet_wrap(~temperature +
    radiation, drop = FALSE, ncol = 6) + labs(title = "Left to right: Increasing radiation; Top to bott
```



And the relationship of fit with temperature:

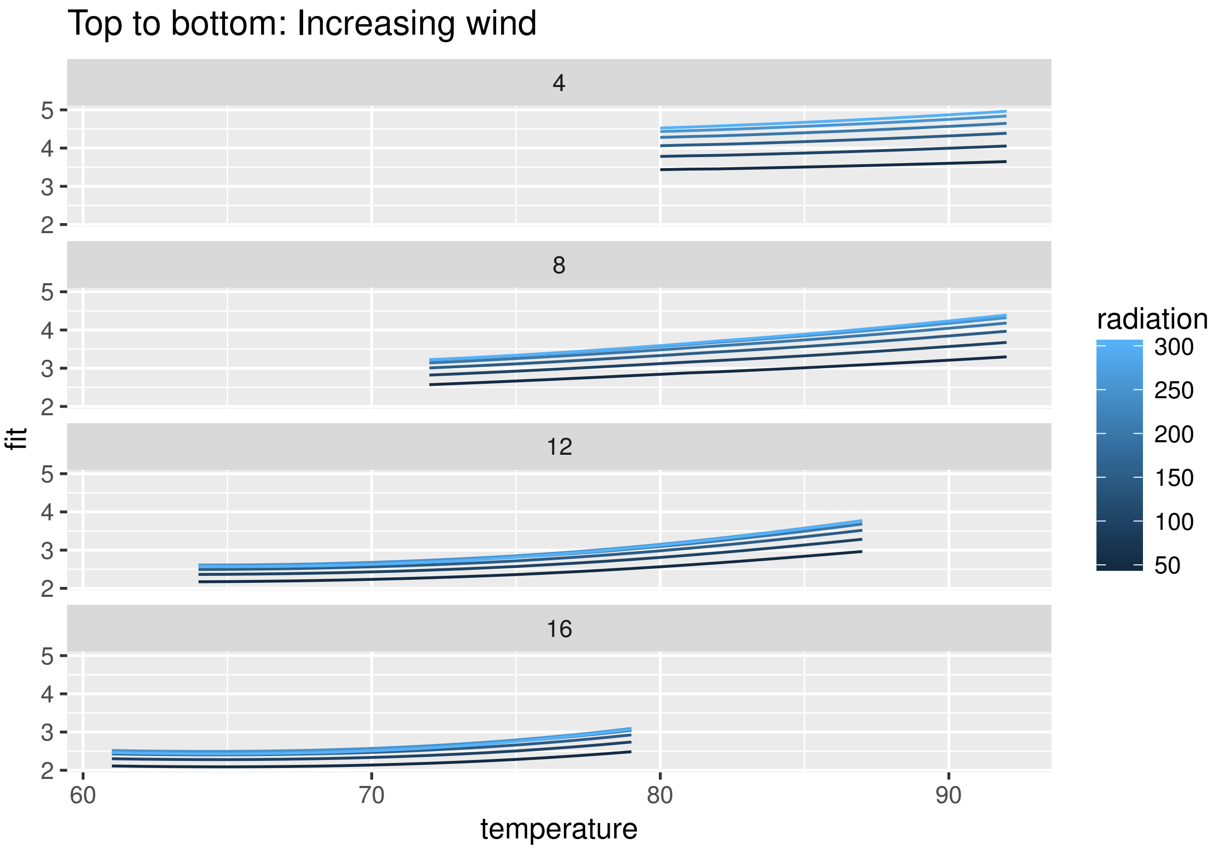
```
temperature.grid = expand.grid(radiation = seq(50, 300, 50), temperature = 61:92,
    wind = c(4, 8, 12, 16))
environmental.predict = predict(environmental.lo, newdata = temperature.grid)
environmental.df = data.frame(temperature.grid, fit = as.vector(environmental.predict))
temperature.crop = (temperature.grid$temperature < (-2 * temperature.grid$wind +
    112)) & (temperature.grid$temperature > (-2 * temperature.grid$wind + 87))
environmental.df = environmental.df[temperature.crop, ]
ggplot(environmental.df, aes(x = temperature, y = fit)) + geom_line() + facet_wrap(~wind +
    radiation, drop = FALSE, ncol = 6) + labs(title = "Left to right: Increasing radiation; Top to bott
```



Note that just about everything in these plots was curved, justifying the loess fit instead of `lm()`. In the last set, the slope seemed to change going downward, justifying the temperature:wind radiation.

To look more closely at the radiation:temperature interaction, we'll take the last set of plots and collapse all the columns of top of each other.

```
ggplot(environmental.df, aes(x = temperature, y = fit, group = radiation, color = radiation)) +
    geom_line() + facet_wrap(~wind, drop = FALSE, ncol = 1) + labs(title = "Top to bottom: Increasing wi
```

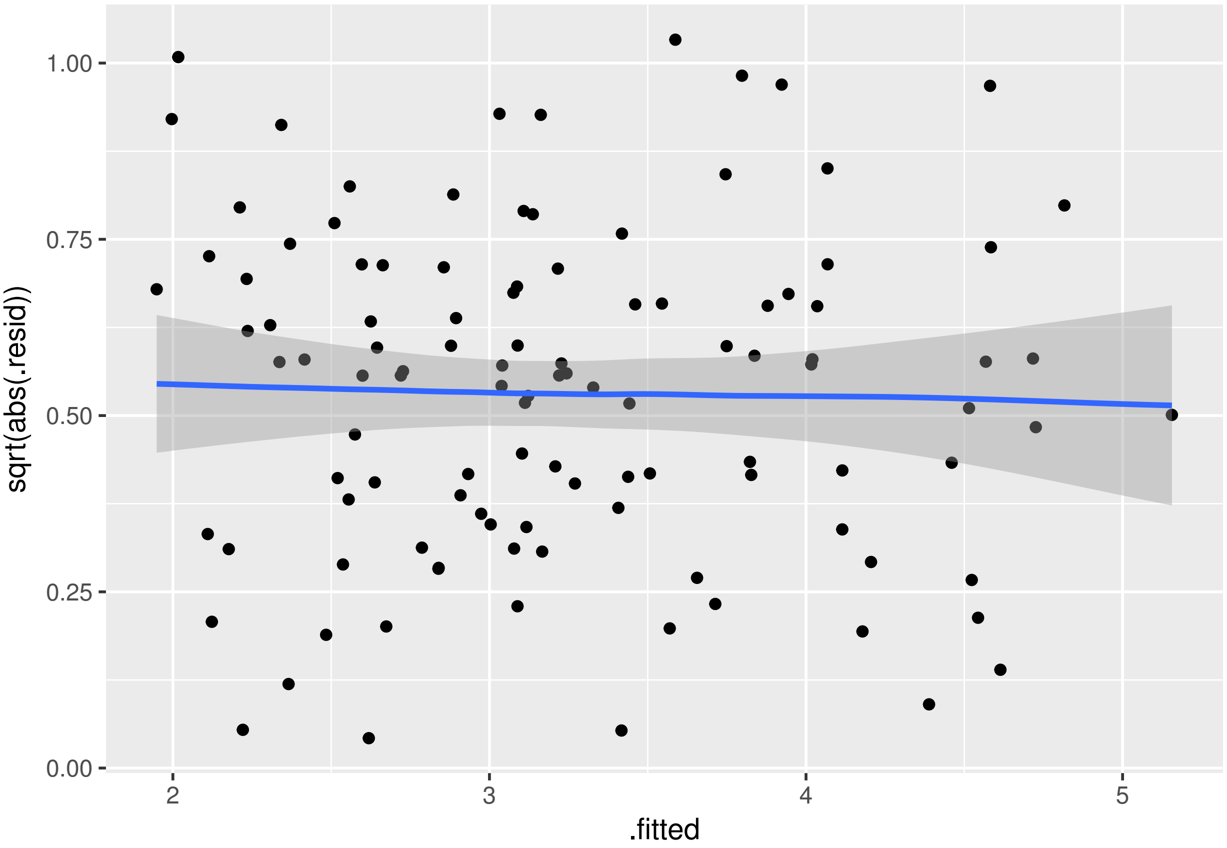


This is a nice one-plot summary of the model. (An alternative would be to draw contour plots faceted by the remaining explanatory variable; let's, uh, leave this as an exercise.) There's probably a little interaction between temperature and radiation, but not much. We could include or leave it out of the model depending on whether you prefer maximalism or minimalism.

5.1.5 Was the transformation a good idea?

Homoscedasticity is a good goal: even if we don't strictly need it, it's usually a signal that we're on the right track. We can do our usual spread-location plot of root absolute residuals:

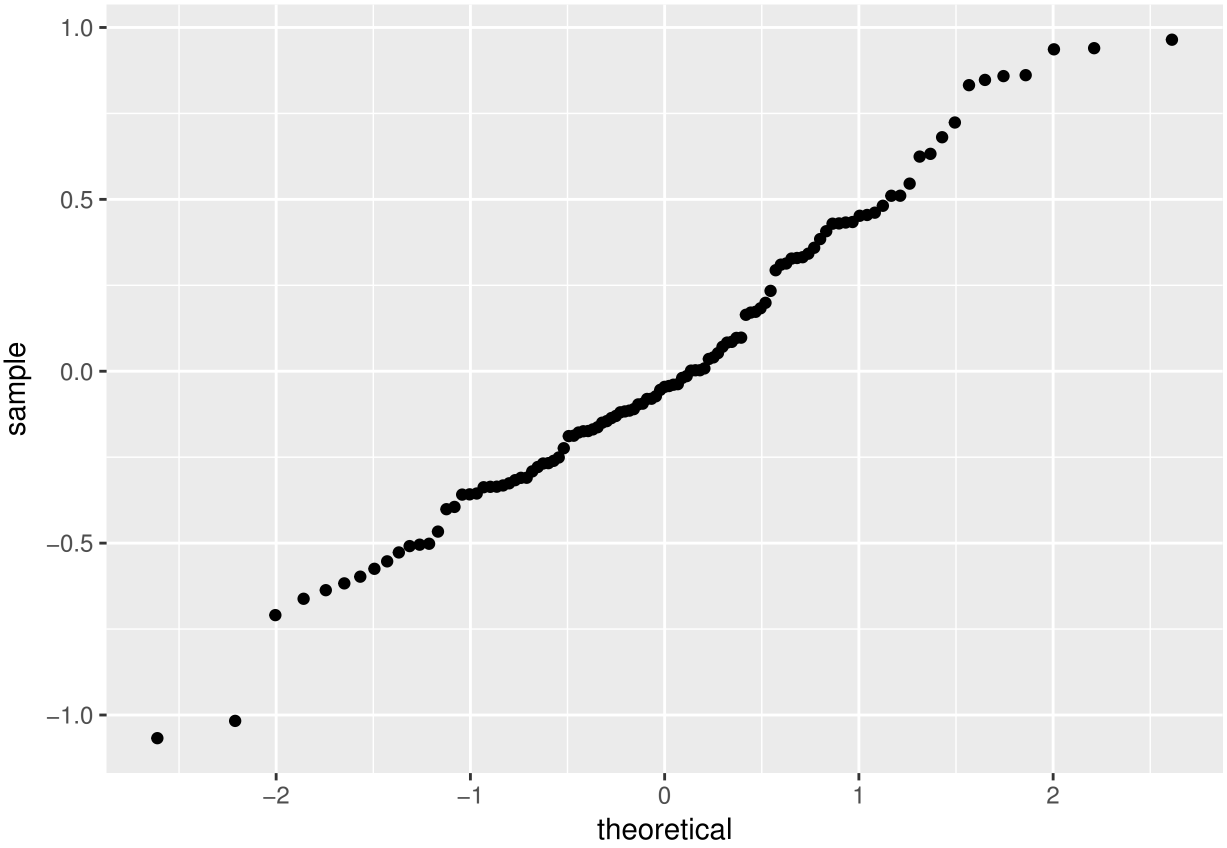
```
environmental.lo.broom = augment(environmental.lo)
ggplot(environmental.lo.broom, aes(x = .fitted, y = sqrt(abs(.resid)))) + geom_point() +
    geom_smooth(span = 1, method.args = list(degree = 1))
```



Looks like it was. (Trying out other transformations generally makes this plot worse.)

Check for normality:

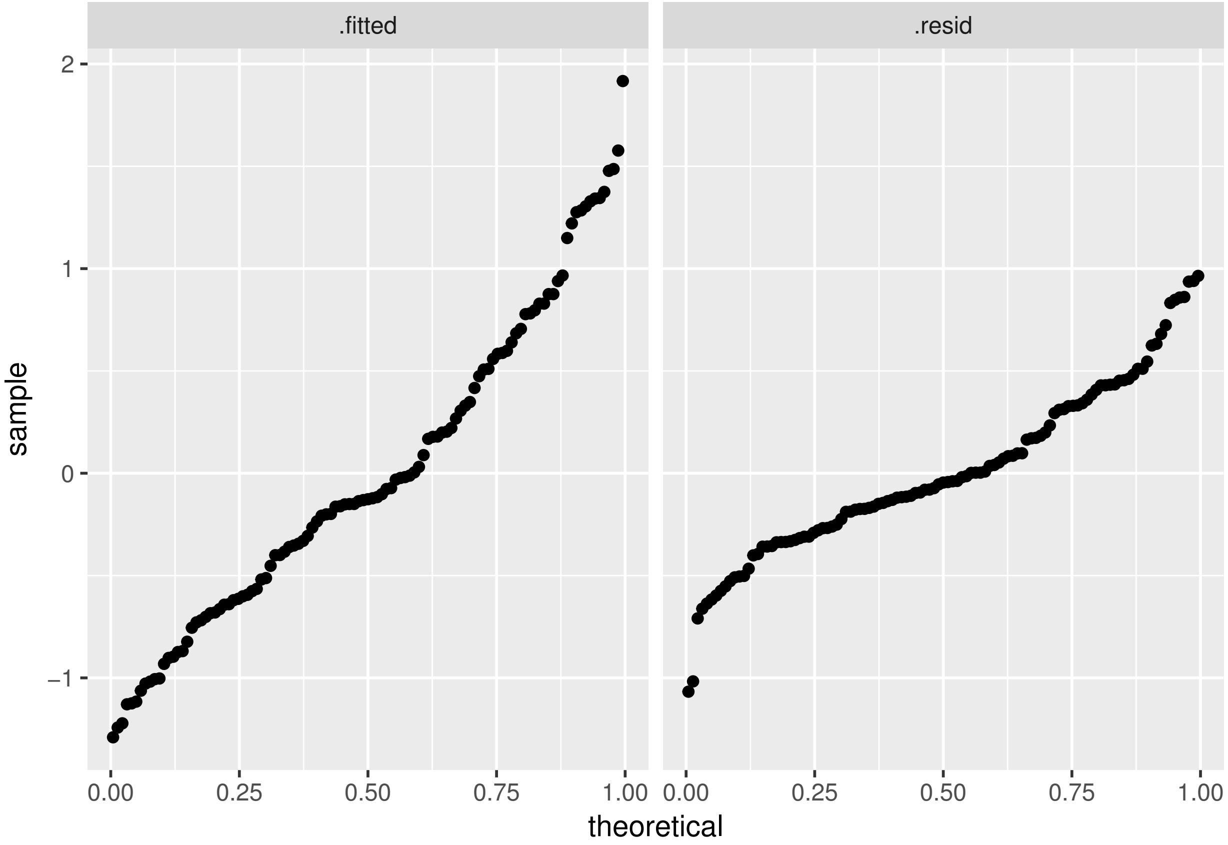
```
environmental.lo.broom = augment(environmental.lo)
ggplot(environmental.lo.broom, aes(sample = .resid)) + stat_qq()
```



It's surprisingly good, with no heavy tail or outlier issues. Least squares was an appropriate choice for our fitting method.

Finally, compare the fitted values and the residuals.

```
environmental.lo.broom$.fitted = environmental.lo.broom$.fitted - mean(environmental.lo.broom$.fitted)
environmental.lo.long = environmental.lo.broom %>% gather(component, value,
    c(.fitted, .resid))
ggplot(environmental.lo.long, aes(sample = value)) + stat_qq(distribution = "qunif") +
    facet_grid(~component)
```



We've explained a decent amount of the variation, though there's still a substantial amount left unexplained. That's okay: sometimes nature is inexplicable.

5.2 Four or more variables

READ: Cleveland pp. 293–301.

5.2.1 Hamster organs

Load stuff:

```r
load("lattice.RData")
library(ggplot2)
library(GGally)
library(tidyr)
```

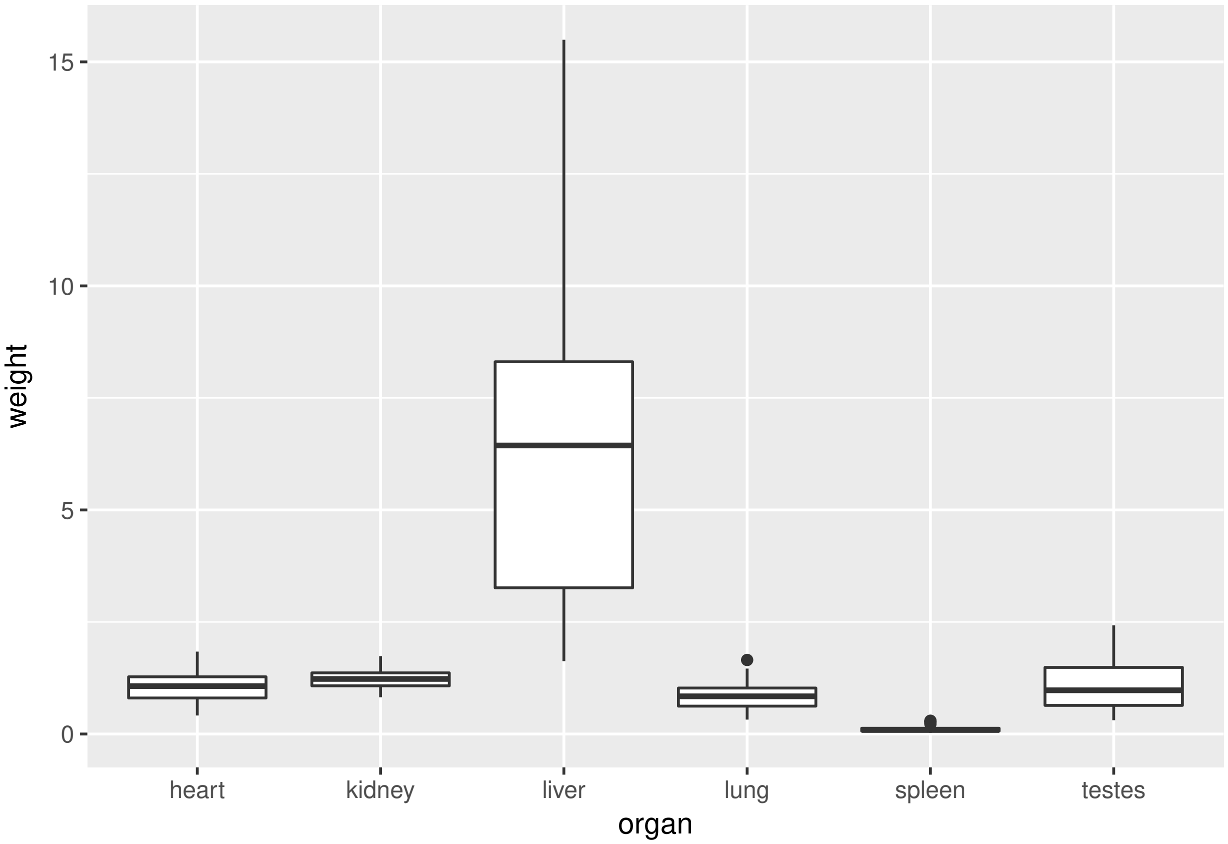
For some reason Cleveland has data on the weights (in grams) of six organs from each of 73 hamsters that died of a congenital heart problem. Here, there's no response variable: we're just interested in how the variables relate to each other. Before we jump to the multivariate stuff, we see what the univariate data looks like.

```r
summary(hamster)
```

```
##       lung            heart            liver            spleen
##  Min.   :0.3216   Min.   :0.4134   Min.   : 1.626   Min.   :0.0293
##  1st Qu.:0.6220   1st Qu.:0.8039   1st Qu.: 3.262   1st Qu.:0.0657
##  Median :0.8416   Median :1.0690   Median : 6.439   Median :0.0896
##  Mean   :0.8457   Mean   :1.0581   Mean   : 6.299   Mean   :0.1035
##  3rd Qu.:1.0260   3rd Qu.:1.2763   3rd Qu.: 8.307   3rd Qu.:0.1283
##  Max.   :1.6521   Max.   :1.8390   Max.   :15.492   Max.   :0.2959
##      kidney           testes
##  Min.   :0.8199   Min.   :0.3082
##  1st Qu.:1.0751   1st Qu.:0.6387
##  Median :1.2291   Median :0.9761
##  Mean   :1.2256   Mean   :1.0900
##  3rd Qu.:1.3642   3rd Qu.:1.4862
##  Max.   :1.7381   Max.   :2.4244
```

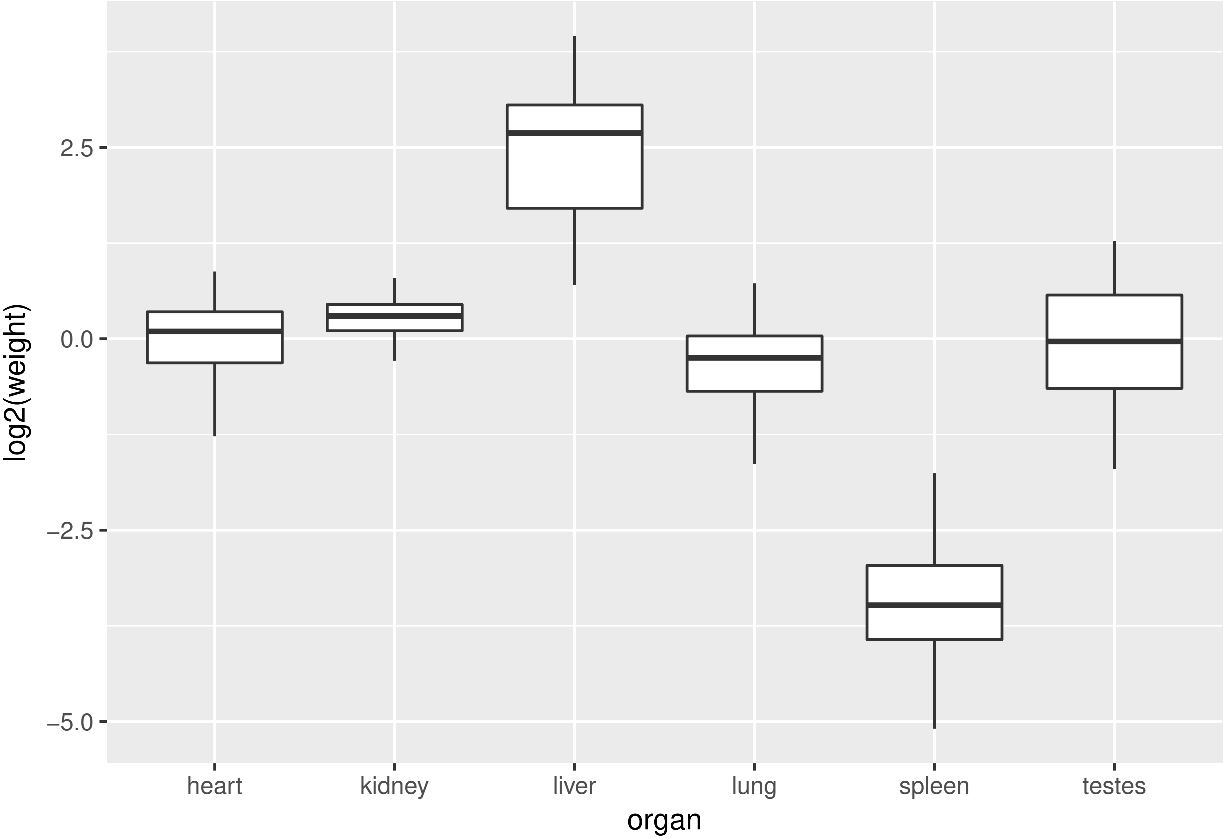
Let's put the data into long form and plot it.

```
hamster.long = hamster %>% gather(organ, weight)
ggplot(hamster.long, aes(x = organ, y = weight)) + stat_boxplot()
```



The different orders of magnitude make it hard to see what's going on. We can use a log scale instead.

```
ggplot(hamster.long, aes(x = organ, y = log2(weight))) + stat_boxplot()
```

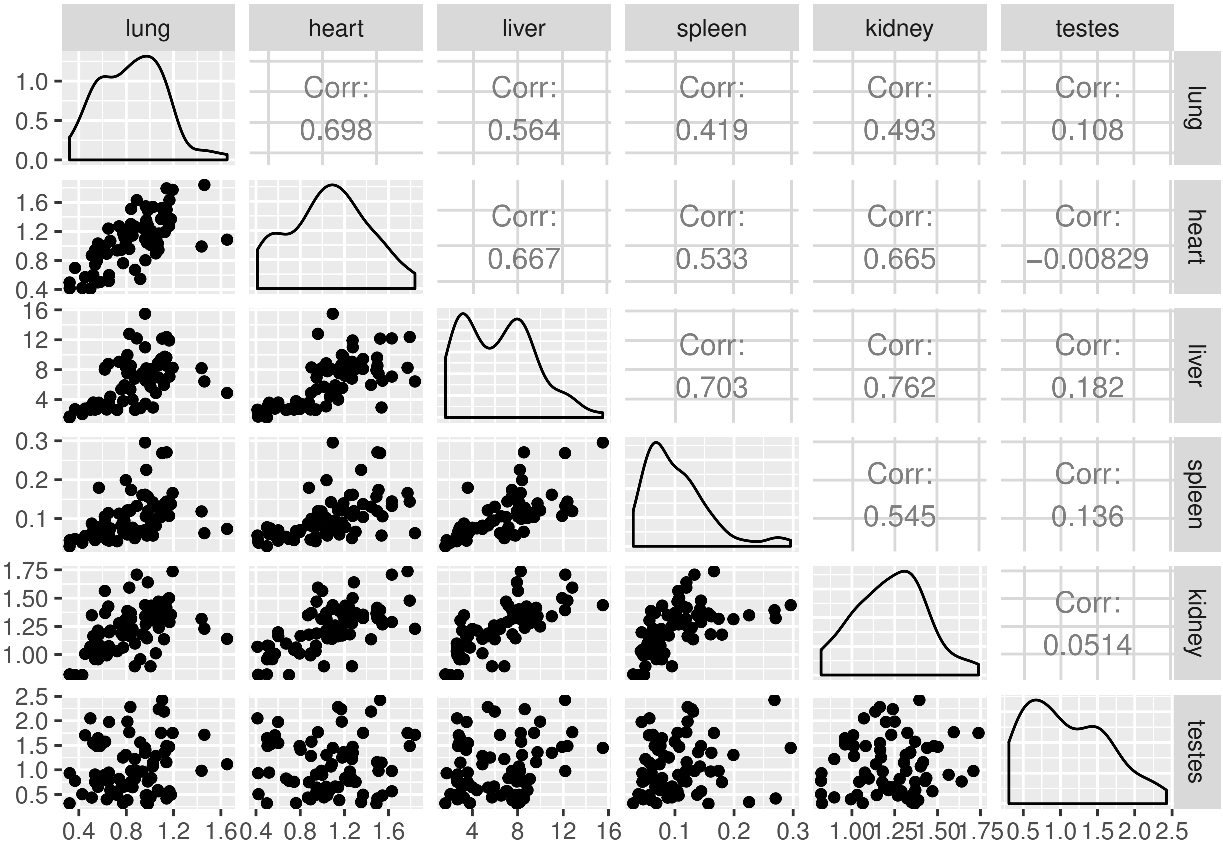


Now it’s clear that livers are the heaviest organ and spleens are the smallest. The others are of similar order of magnitude.

Note that the reason we transformed here was magnitude, not skewness. In scatterplots, we can use different scales on different axes, so we don’t necessarily have to pursue this transformation.

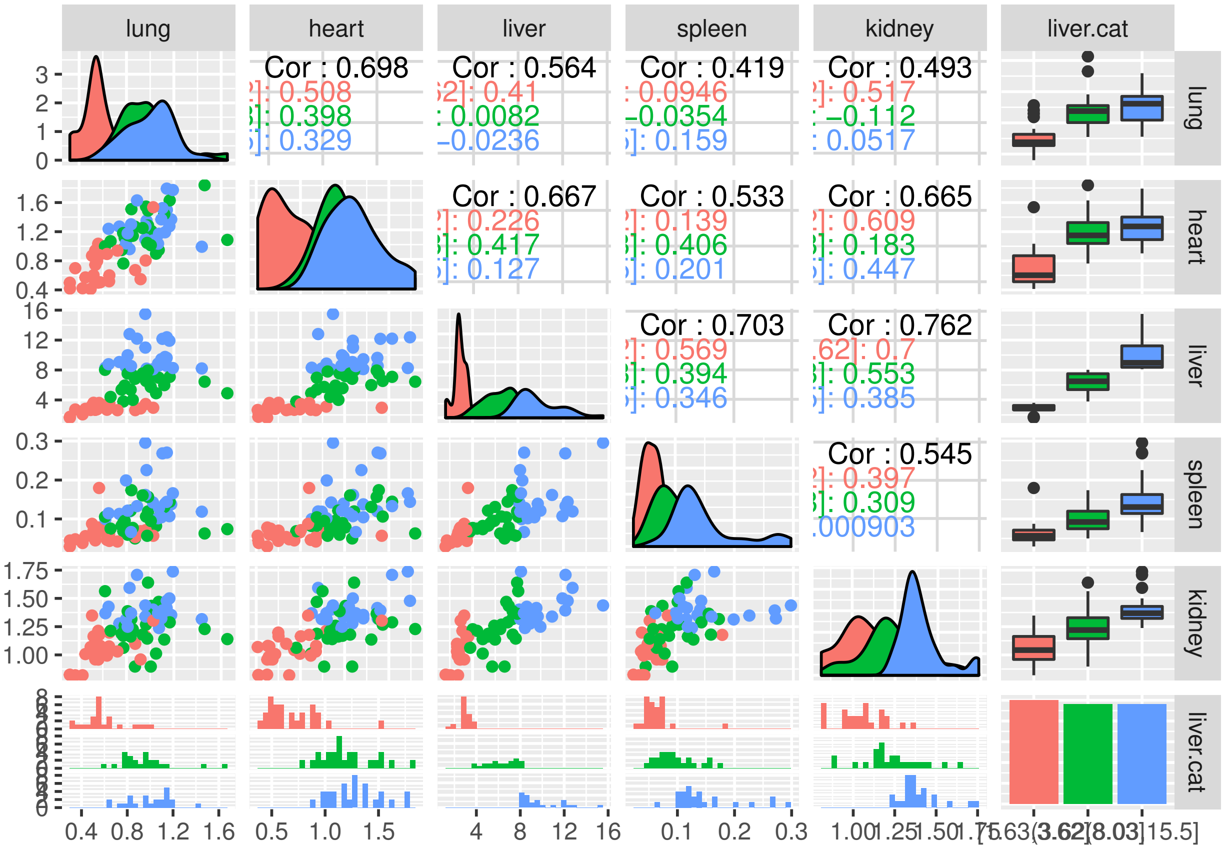
Let’s just draw paired scatterplots on the original scale:

```
ggpairs(hamster)
```



Some variables are highly associated, like heart-lung-liver and liver-kidney-spleen. Only testes weight doesn't seem to be related to anything else, so we'll forget it. In some ways liver seems to be the key variable, since it's highly correlated with everything besides testes. We can make chopped liver and add color:

```
liver.cat = cut_number(hamster$liver, n = 3)
ggpairs(data.frame(hamster[, 1:5], liver.cat), aes(color = liver.cat))
```



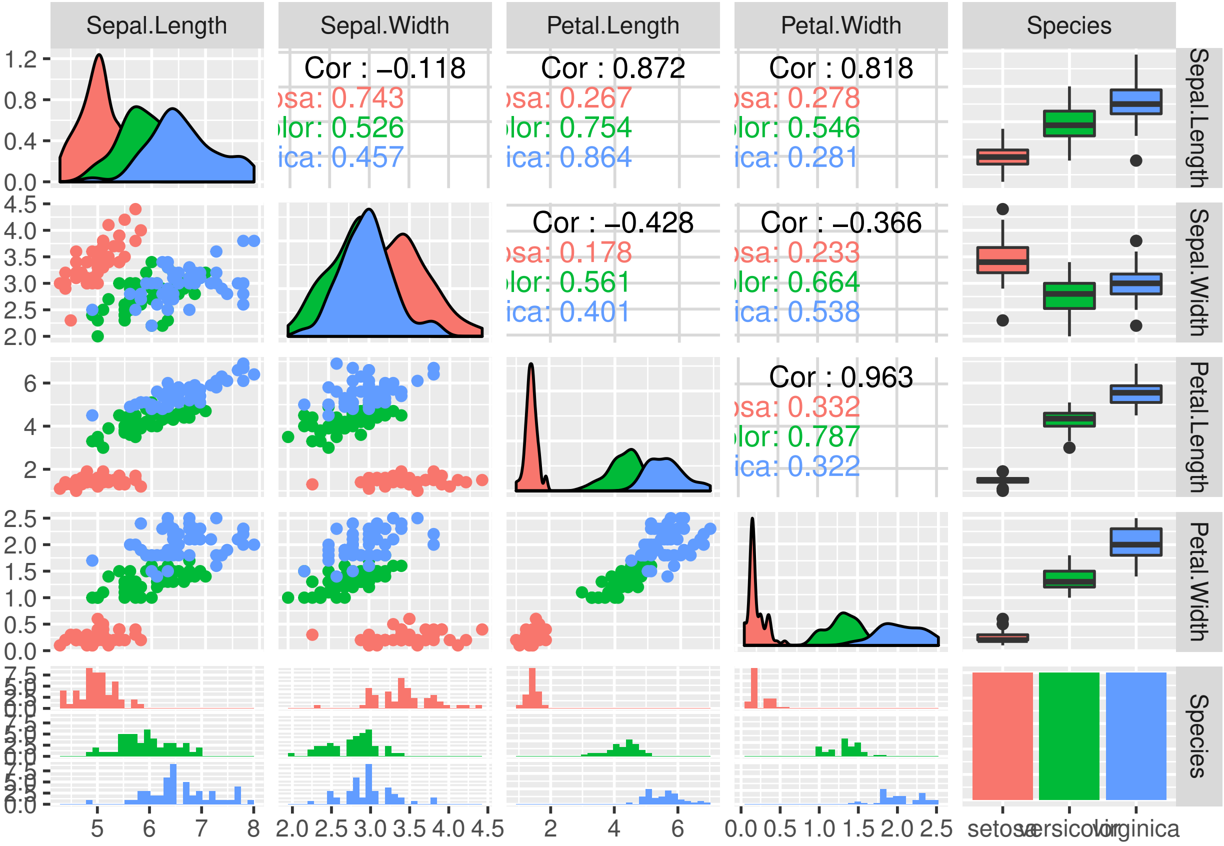
We see that hamsters with small livers have small other organs. Hamsters with big livers generally have the biggers kidneys and spleens, but their lung and heart sizes substantially overlap with those of medium-livered hamsters.

If you want to try out the *brushing* method described in Cleveland, you can try out the `rggobi` package. We're not going to bother though.

5.2.2 Is it the iris data? Yes, it's the iris data

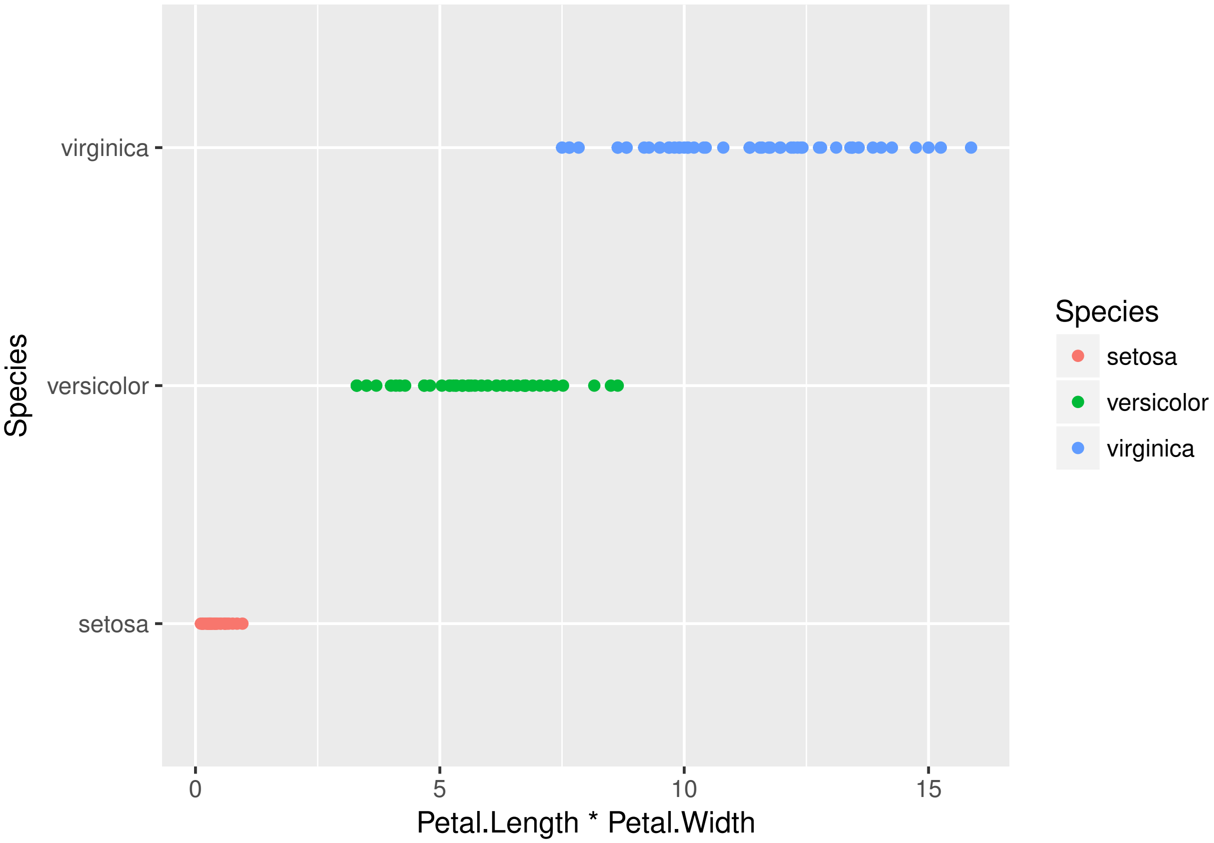
The iris data set, which is studied in every statistics class, consists of measures of sepal and petal length and width (in cm) of 50 setosa irises, 50 versicolor irises, and 50 virginica irises. We want to be able to tell the three species apart based on these measurements. In our case, we want a practical rule rather than a "machine learning" rule – one that we can understand without spending half our lives studying irises.

```
ggpairs(iris, aes(color = Species))
```



The 1D plots are clear enough – blue is generally bigger than green, and green is generally bigger than red (except in terms of sepal width.) Since after years of looking at this data set I still don't know what a sepal is, let's stick to petals. How do we combine length and width into one number? Well, if the petal was rectangular, length times width would give area. The petals obviously aren't rectangular, but we can still do the multiplication and get something roughly proportional to area.

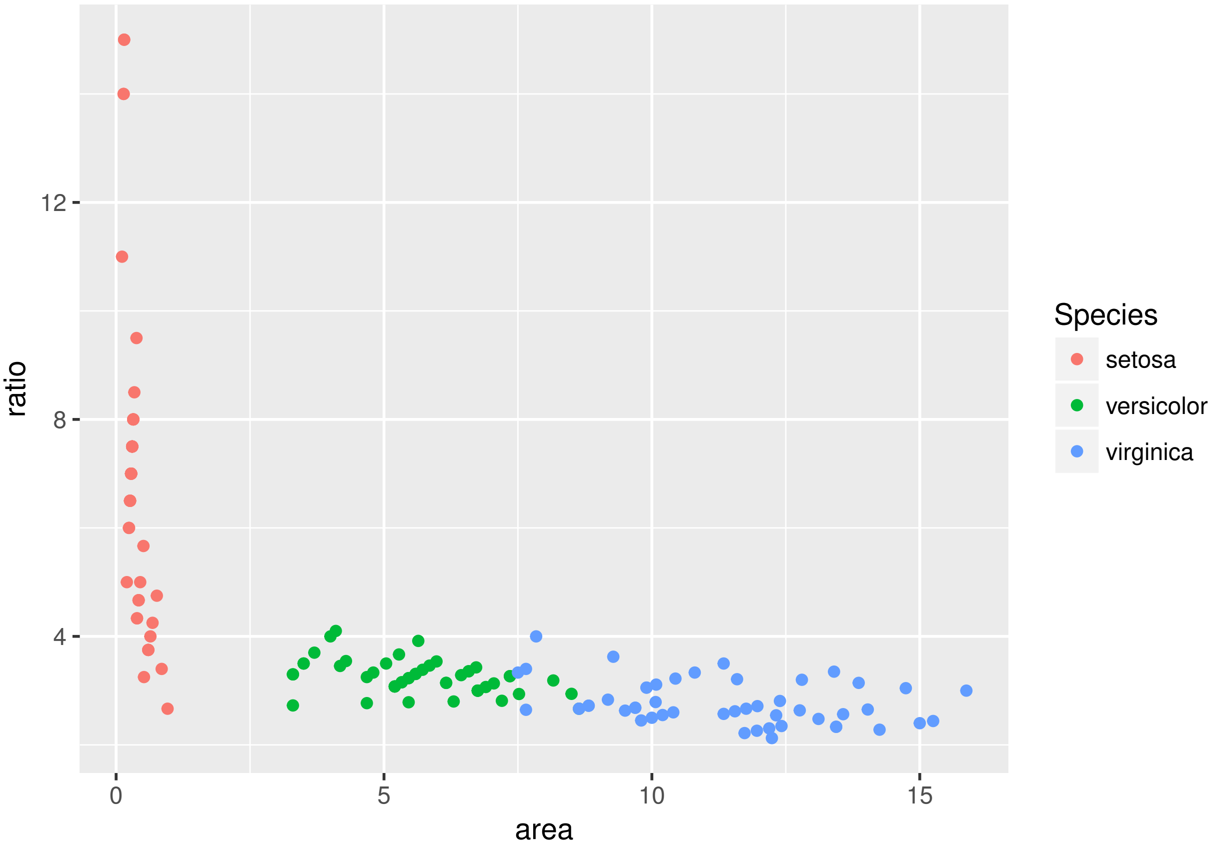
```
ggplot(iris, aes(y = Species, x = Petal.Length * Petal.Width, color = Species)) +
    geom_point()
```



It looks like apart from the very biggest versicolor and the very smallest virginica, petal area classifies correctly.

What can we plot area against? Well, the information you lose when you multiply is the ratio of the two numbers. If we have both the area and the ratio of length to width, then we lose no information (we can recover both length and width.) Let's scatterplot these:

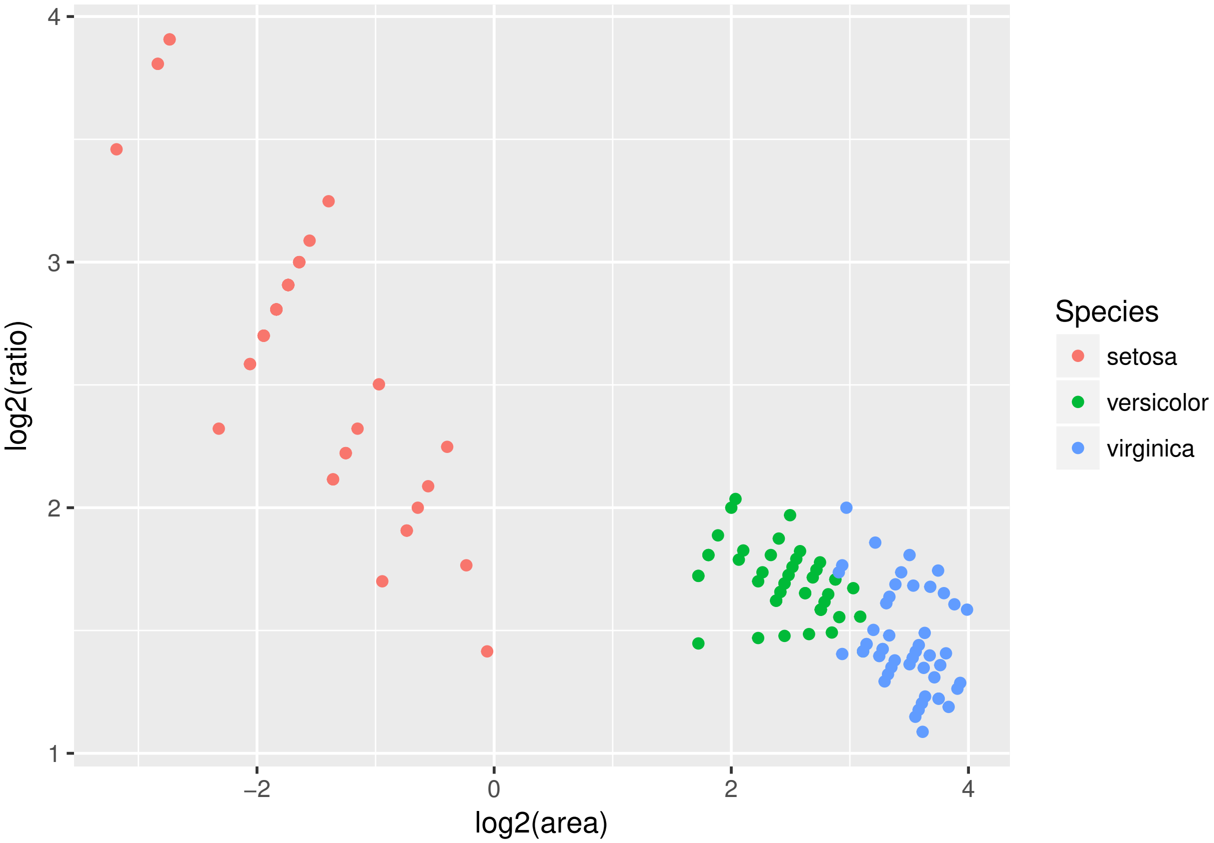
```
area = iris$Petal.Length * iris$Petal.Width
ratio = iris$Petal.Length/iris$Petal.Width
iris.size = data.frame(iris, area, ratio)
ggplot(iris.size, aes(x = area, y = ratio, color = Species)) + geom_point()
```



So we get a simple rule: If petal length times width is less than 1 cm^2, it's setosa. If it's 3 to 7, it's versicolor, and if it's more than 9, it's virginica. If it's between 7 and 9 then you need to look more carefully.

The boundary between versicolor and virginica is clearer on a log scale.

```
ggplot(iris.size, aes(x = log2(area), y = log2(ratio), color = Species)) + geom_point()
```



If we simply draw vertical lines at 1 and 3 (which would be 2 and 8 cm^2 on the original scale), we'll only be wrong 7 out of 150 times. (By drawing the line more carefully we can reduce the training set errors to 3, but this is unlikely to significantly reduce error on a new test set of data.)