Stat 849: Plotting responses and covariates

Douglas Bates

Department of Statistics
University of Wisconsin, Madison

2010-09-03

Outline

Graphics systems in R

- R provides three different high-level graphics systems

  base graphics The system from the original S language developed at Bell Labs during the 1980's

  lattice Developed by Deepayan Sarkar during his Ph.D. here at U. of Wisconsin. It is based on an earlier system called "trellis" developed by Bill Cleveland.

  ggplot2 Developed by Hadley Wickham during his Ph.D. at Iowa State. As Hadley says "ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts."

- Deepayan and Hadley were both awarded (in different years) the Chambers Award for the outstanding contribution to statistical computing and graphics by a student.

Pros and cons of base graphics

- Base graphics consists of high-level graphics functions or methods (functions such as `plot`, `pairs`, `histogram`) and functions that add to an existing plot (`lines`, `points`, `segments`, `text`, `abline`)
- Most introductory books on R describe the base graphics system. That is, you can read about it in many places.
- The graphics model corresponded to graphics devices from the 1980's, which were pen plotters and special (very expensive) terminals. It predates any ideas of windowing systems.
- The graphics system uses a set of graphics parameters that are set by calls to the `par()` function.
- It is easy to produce plots in this system. It is difficult to produce good plots. I only use base graphics for residual plots and recommend you do likewise.

Pros and cons of lattice graphics

- Lattice is a very flexible system based on formula/data specification of the plot.
- There are many, many options in some of the lattice graphics functions. Look at the result of `help(xyplot, package = "lattice")` some time.
- You can't easily add to a lattice plot. Customization of plots frequently involves writing "panel functions" which is a bit advanced for most people. A single call to a lattice function can be quite complicated.
- The data visualization techniques in lattice are very good. Deepayan's book and associated web site provide lots of good examples.
- I will recommend lattice for one exploratory technique, which is producing a "scatterplot matrix" using the `splom()` function.

Pros and cons of ggplot2 graphics

- The system has been carefully designed to produce informative and attractive data plots.
- It is based on a consistent, although sometimes a bit obscure, set of principles about graphical displays.
- Like any new system it requires a bit of getting used to. Sometimes you get frustrated that something that should be easy doesn't seem to be possible.
- There is an active mailing list for `ggplot2`, Hadley's book, of which the important introductory chapter is freely available, and the web site http://had.co.nz/ggplot2
- For this course I will use `ggplot2` with the two exceptions (residual plots and `splom()`) mentioned earlier.

An example using the brain weight data

```
1 library(alr3)
2 library(ggplot2)
```

```
Loading required package: reshape
Loading required package: plyr
Loading required package: grid
Loading required package: proto
```

- In line 1 we attach the `alr3` package to provide access to the `brains` data. Line 2 attaches the `ggplot2` package which, in turn, loads several other packages.
- Now confirm the structure of the `brains` data

```
1 str(brains)
```

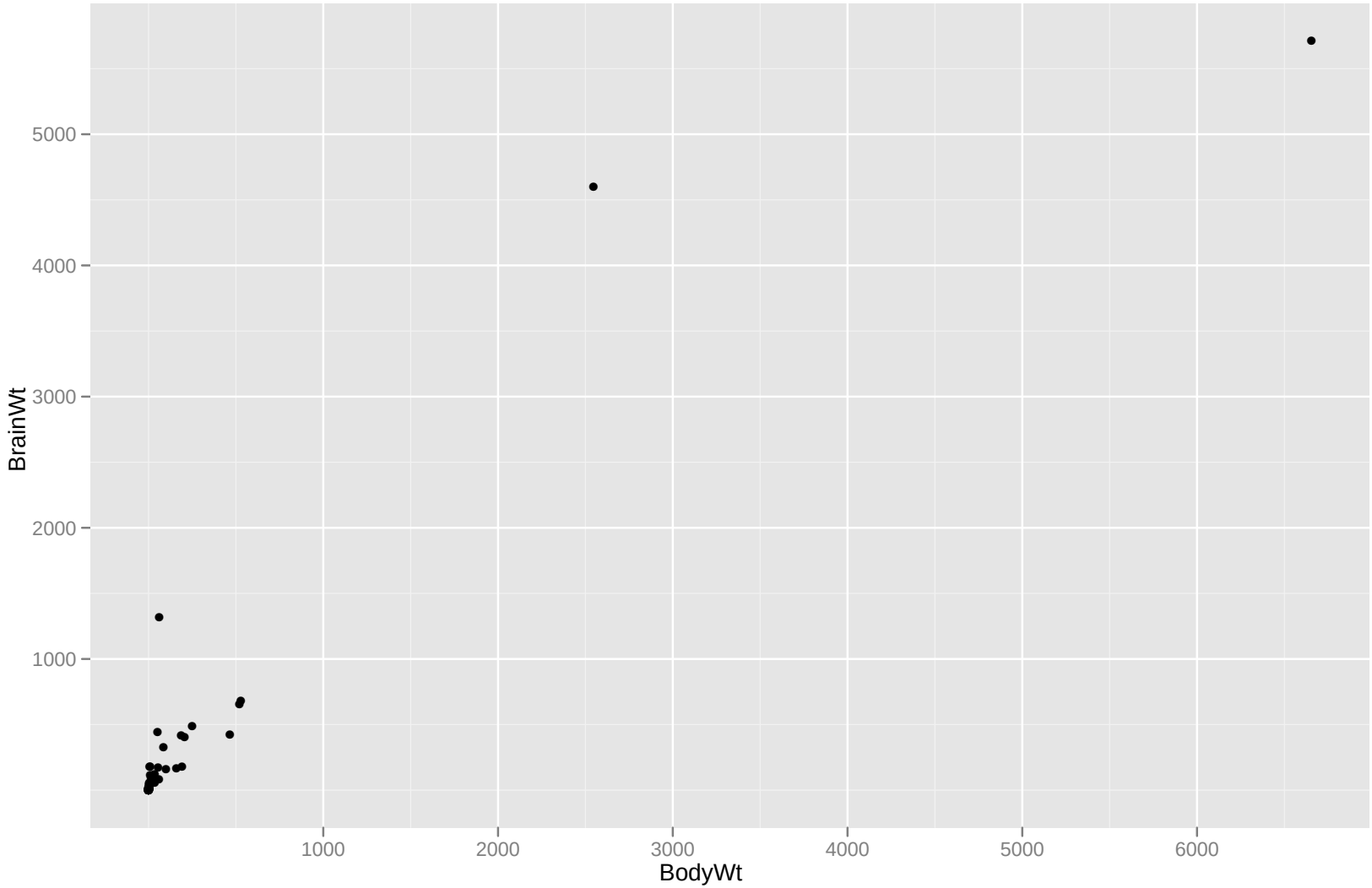
```
'data.frame': 62 obs. of  2 variables:
 $ BrainWt: num  44.5 15.5 8.1 423 119.5 ...
 $ BodyWt : num  3.38 0.48 1.35 464.98 36.33 ...
```

Using qplot() to produce a simple scatter plot

- The simplest way to use ggplot2 is through the qplot() (quick plot) function,
- The first two arguments are x and y. We also need to specify the data set and we must name that argument.
- Other, optional arguments specify how x/y data values determine the geometric aspects of the plot, how they are scaled, etc.
- The default geom property is point. Many other geom properties can be used.
- The default scales are linear. We can use the log argument to change the x or the y axis, or both, to a logarithmic scale. It turns out we need this for the brains data.
- We can use other arguments to add layers to the plot or to change the display of points or lines. A common enhancement for us will be to add a linear regression lines or smoother lines.
- Try to use informative axis labels (arguments xlab and ylab)

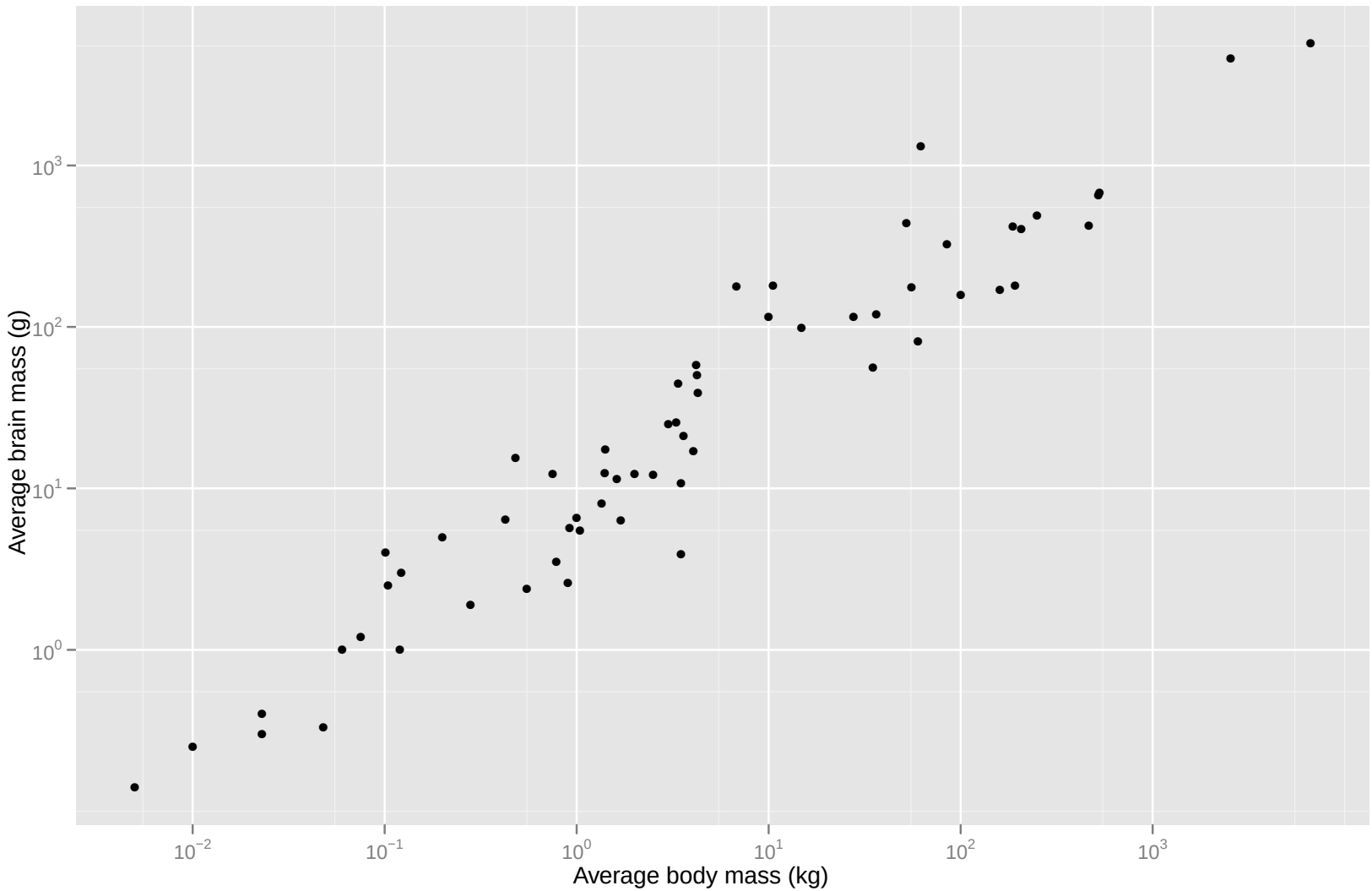
A simple scatter plot of the brains data

```
1 qplot(BodyWt, BrainWt, data=brains)
```



Log-log scatter plot

```
1 qplot(BodyWt, BrainWt, data=brains, xlab="Average body ma
2         ylab="Average brain mass (g)", log="xy")
```



Saving plots

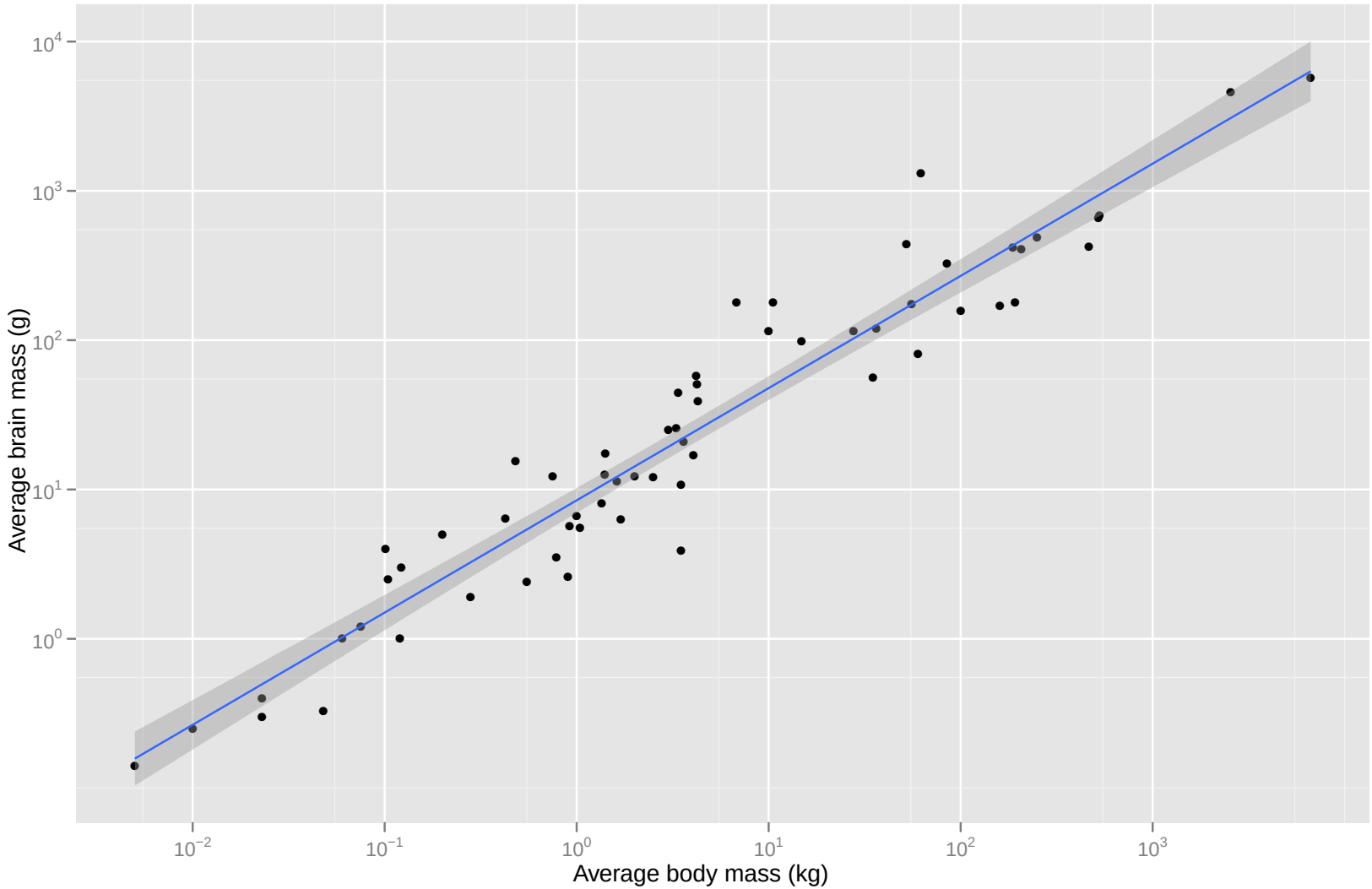
- Producing a publication quality plot often requires many attempts working with variations of the same basic plot.
- We can save the source code and modify that. The ggplot2 package also gives us the option of saving the plot, which includes all the information about the data, the geometry, the labels, etc. We will use this to show variations without repeating the same long source line.

```
1 p <- qplot(BodyWt, BrainWt, data=brains, xlab="Average b
2              ylab="Average brain mass (g)", log="xy")
3 summary(p)
```

```
 data: BrainWt, BodyWt [62x2]
mapping:  x = BodyWt, y = BrainWt
scales:   x, y
faceting: facet_grid(. ~ ., FALSE)
-----------------------------------
geom_point:
```

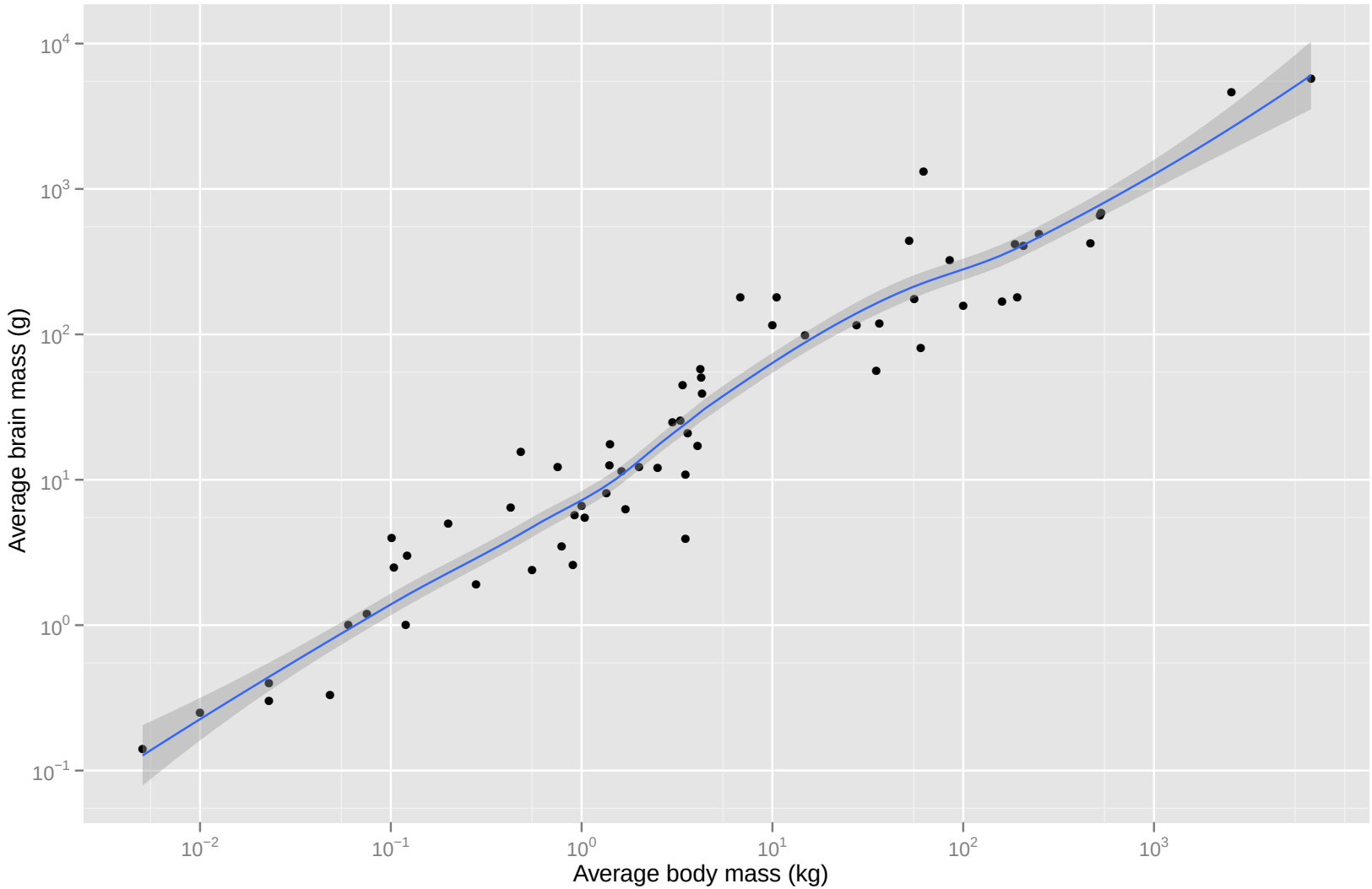
Adding a simple linear regression line

```
1  p + geom_smooth(method="lm")
```



Adding a scatterplot smoother line

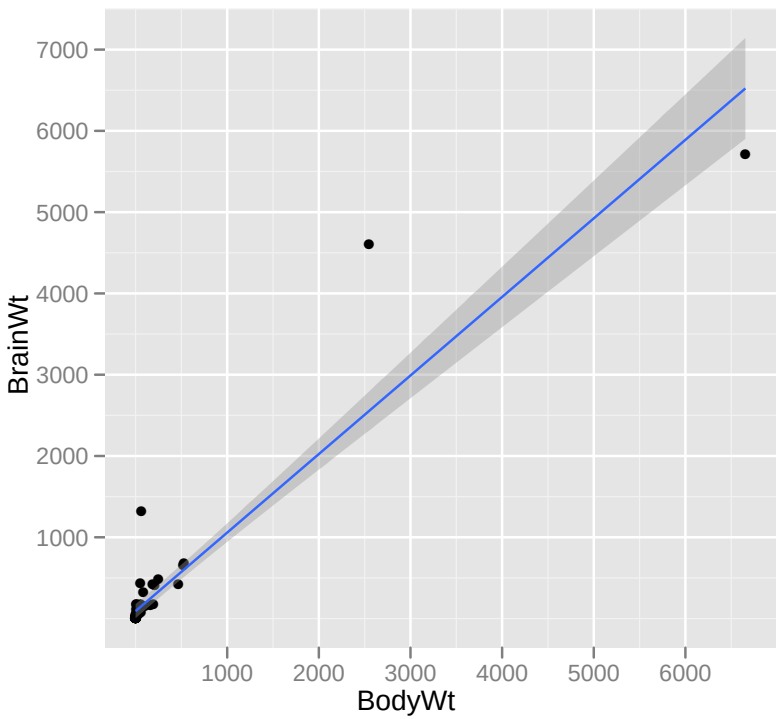
```
1  p + geom_smooth()
```

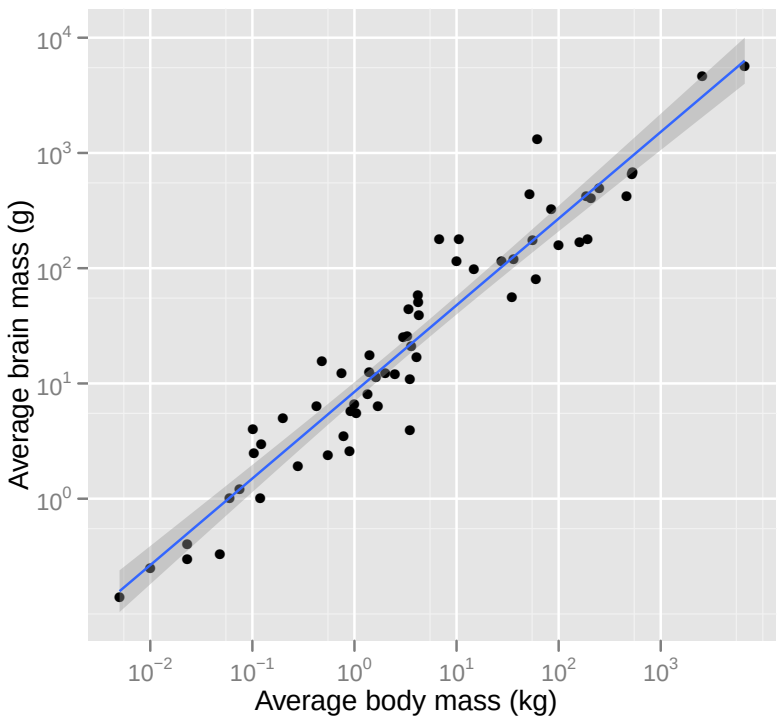


Multiple plots in one figure

- We can use an explicit `print` with a `viewport` specification to render a plot at a particular location in a frame. The second argument is `TRUE` for the first plot and `FALSE` thereafter.

```
1 print(qplot(BodyWt, BrainWt, data=brains)+geom_smooth(me
2       TRUE, vp=viewport(0.25, 0.5, 0.5, 1))
3 print(p + geom_smooth(meth="lm"), FALSE, viewport(0.75,0
```





Incorporating a categorical covariate - Cathedrals data

- The `cathedral` data set in the `alr3` package provides data on the height and width of Gothic and Romanesque cathedrals

```
1 str(cathedral)
```

```
'data.frame': 25 obs. of  3 variables:
 $ Type  : Factor w/ 2 levels "Gothic","Romanesque": 2 2 2
 $ Height: int  75 80 68 64 83 80 70 76 74 100 ...
 $ Length: int  502 522 425 344 407 451 551 530 547 519 ..
```

- Note that the `Type` is a `factor`. Always convert categorical variables to `factors`.
- The data description, accessed as `?cathedral`, indicates that the units of the measurements are in feet. Informative axis labels should include units, when known.

Setting up the basic plot specification

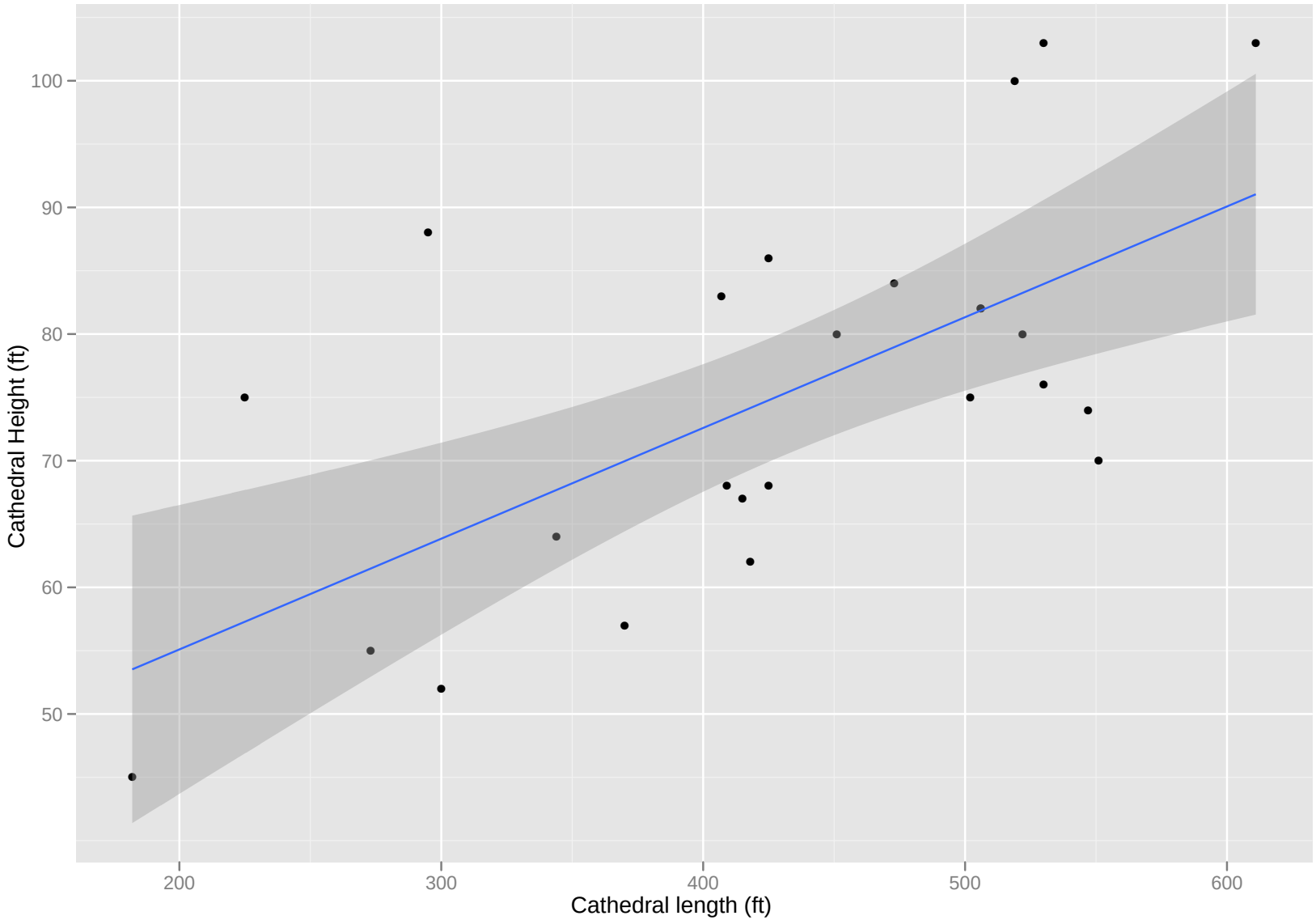
- Our basic plot specification for these data is

```
p1 <- qplot(Length, Height, data=cathedral,
            xlab="Cathedral length (ft)",
            ylab="Cathedral Height (ft)") +
    geom_smooth(meth="lm")
```

- We can render that plot as is or we can incorporate the `Type` factor into the plot, either by creating one panel per type (function `facet_geom()` or `facet.wrap()`) or by distinguishing according to point shape, linetype, color or any combination of these.

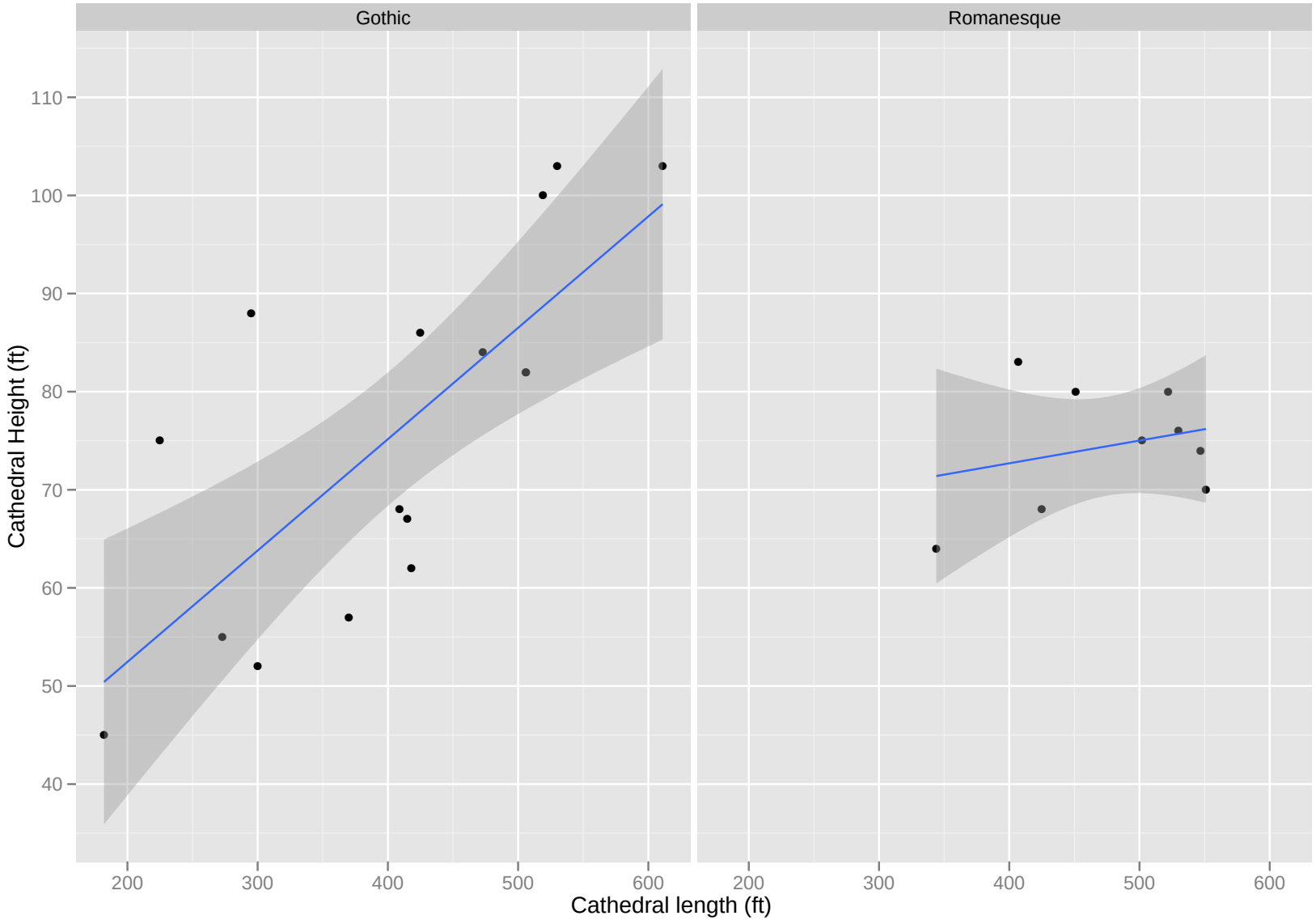
Basic scatterplot of the cathedral data

```
1 p1
```



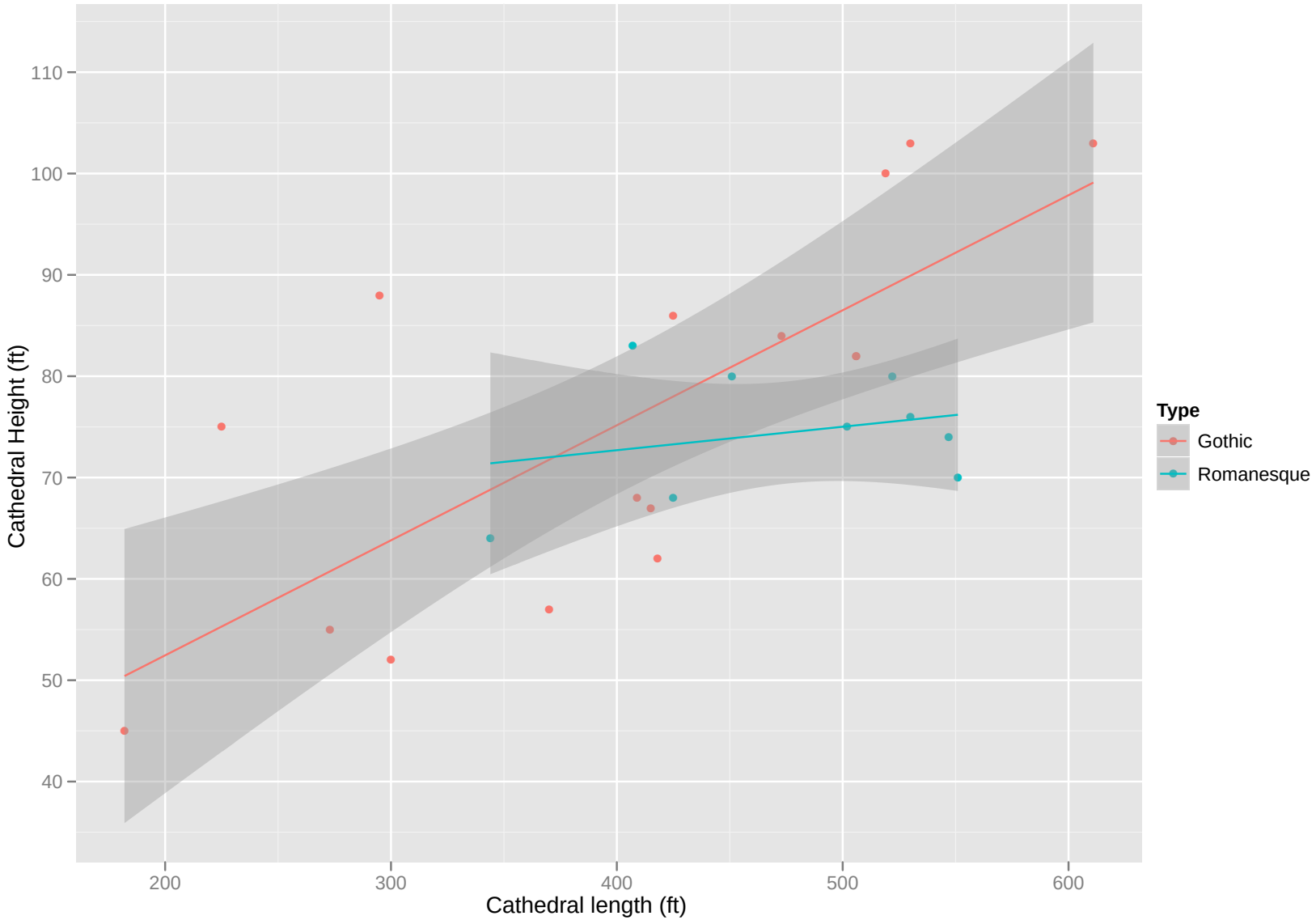
Separate panels by cathedral type

```
1  p1 + facet_grid(. ~ Type)
```



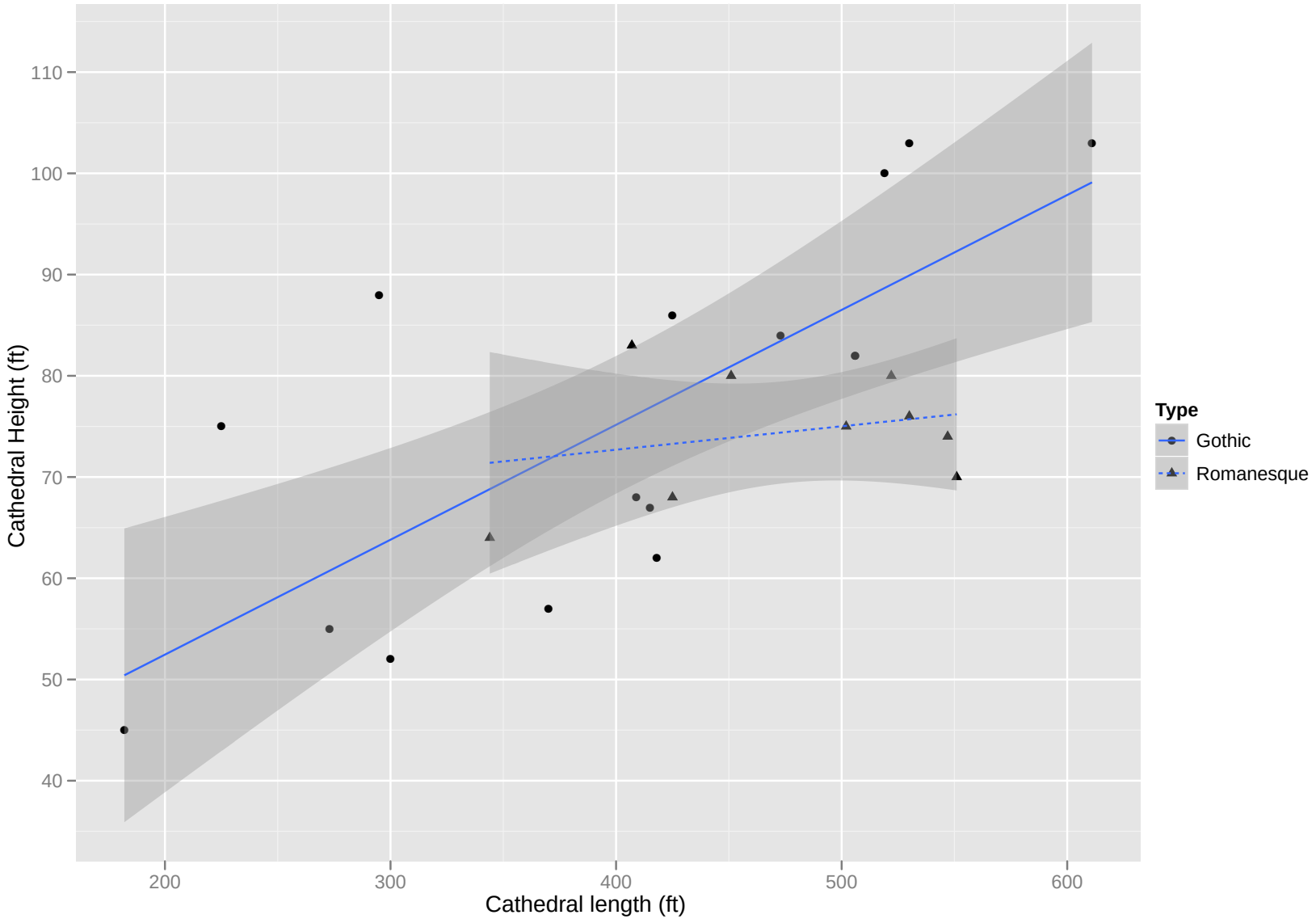
Overlaid plots using different colors for Type

```
1  p1 + aes(color=Type)
```



Overlaid plots using point shapes and linetypes for Type

```
1  p1 + aes(shape=Type,linetype=Type)
```



The `longshoots` data

- Apple trees produce "long shoots" (which may grow as much as 15 to 20 cm over a growing season) as well as "short shoots". Samples of both of these types of shoots were taken from trees in an orchard every few days during a growing season (106 days), and they were then analyzed in a laboratory. We will look at only long shoots. Among the measurements taken was a count of the number of "stem units" on each shoot.
- See `?longshoots` for a description of the data. Variables include:

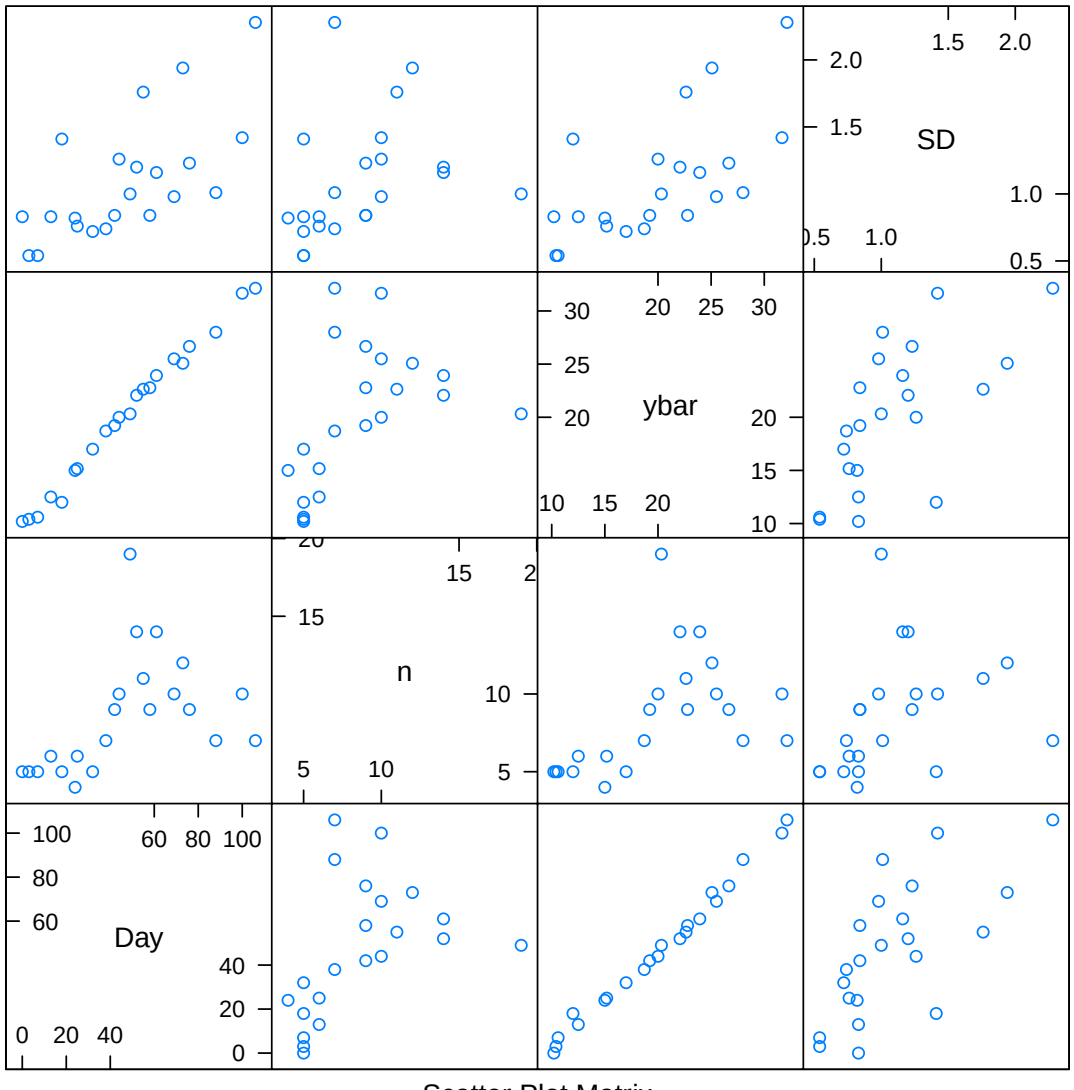
Days	days from dormancy
n	number of shoots sampled
ybar	average number of stem units
SD	within-day standard deviation

pairs or splom plots

- A "scatterplot matrix" plot is often used in the initial exploration of data sets like this, consisting of a response variable and several covariates
- The base graphics function `pairs()` and the lattice graphics function `splom()` produce such plots
- In my opinion these plots are overused. You can get an impression of some of the features of the data but generally the panels are too small to see anything useful.
- It is almost never appropriate to include such a plot in a final report.
- A categorical covariate or a categorical response, in most cases, just wastes space on the plot.
- `splom()` has the advantage over `pairs()` that a categorical covariate can be represented by the point shape or color.

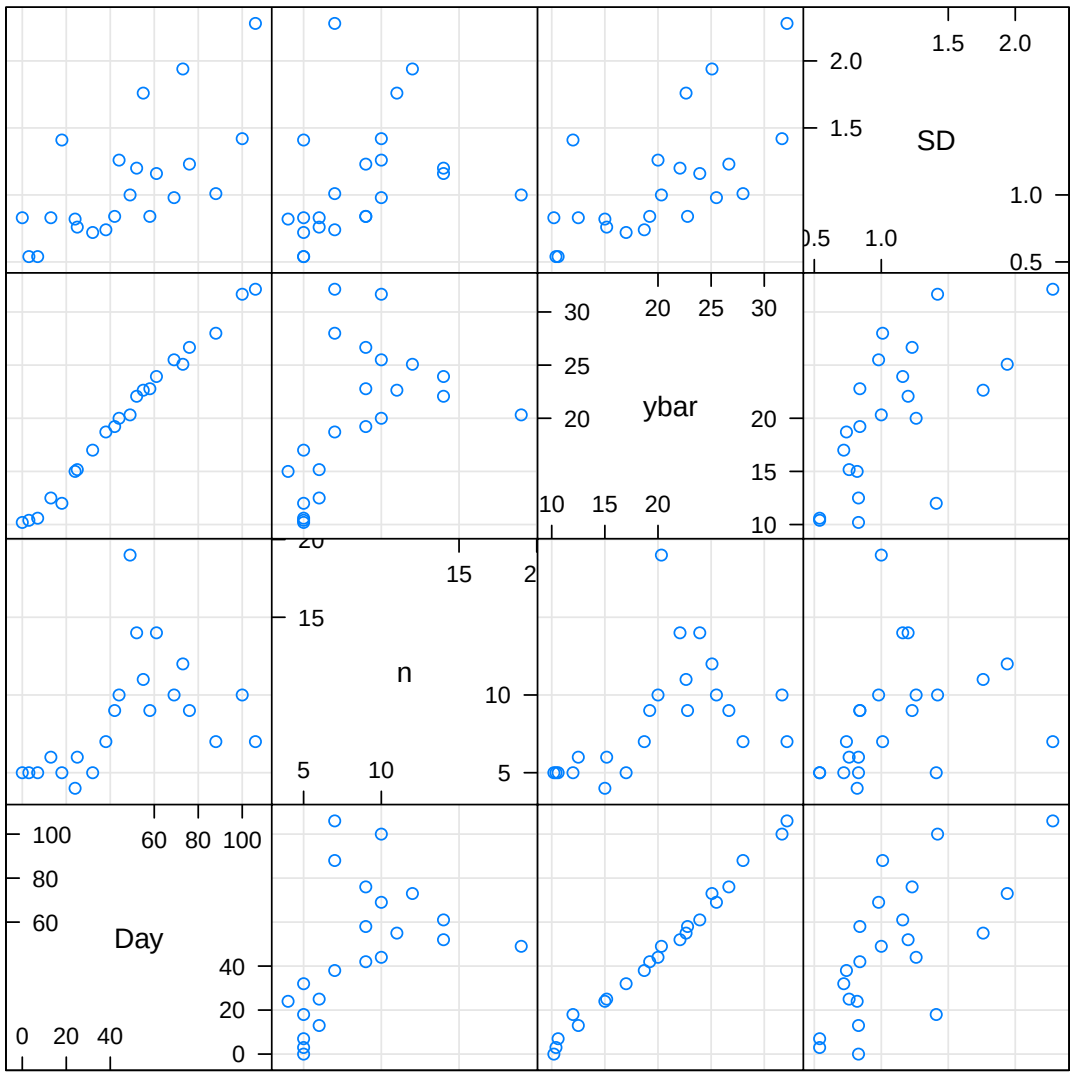
splom plot of the longshoots data

```
1 lattice::splom(~ longshoots[,1:4])
```



splom plot of the longshoots data including a grid

```
1  lattice::splom(~ longshoots[,1:4], type=c("p","g"))
```



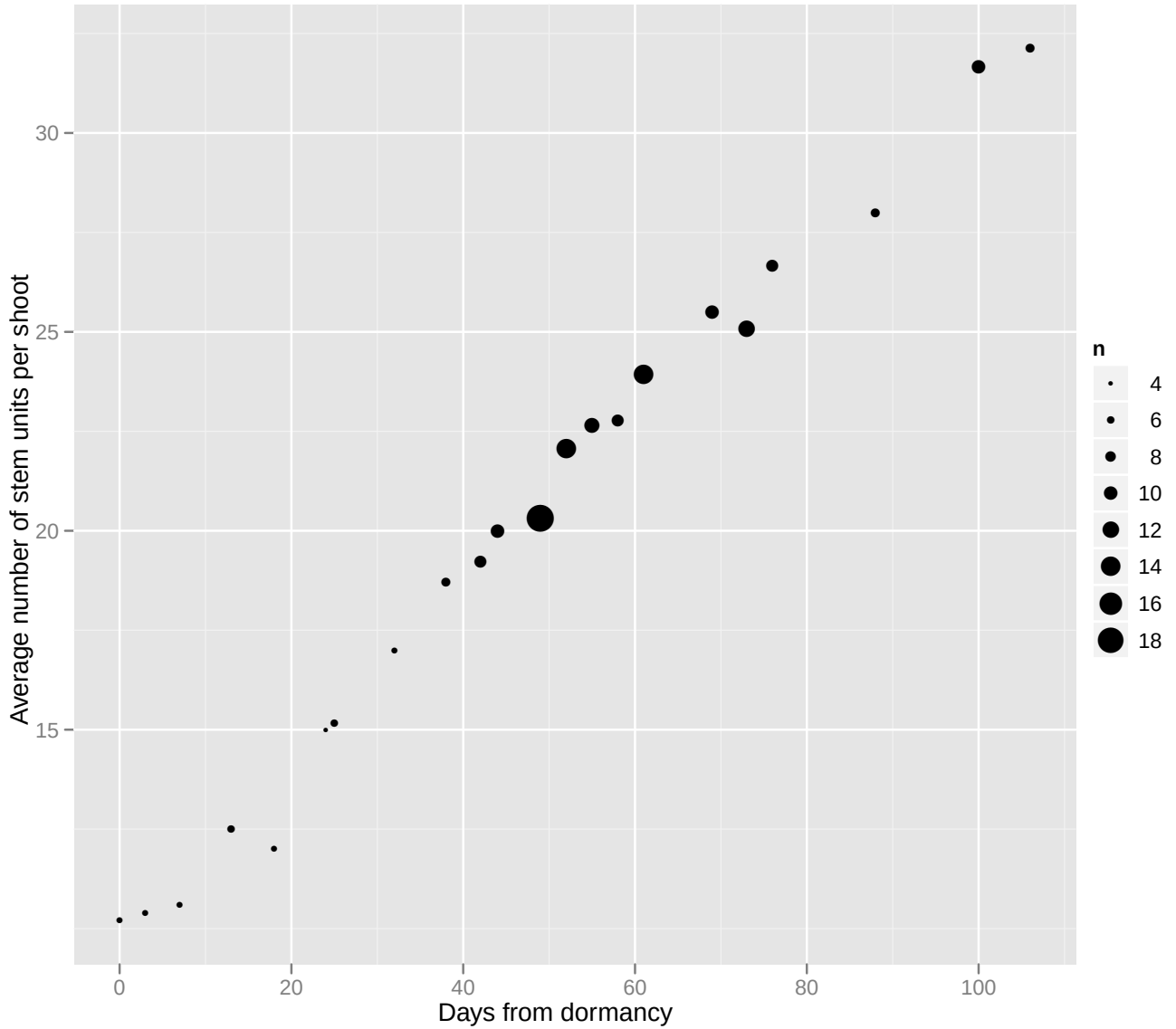
Conclusions from the splom plot

- We can see a strong, apparently linear relationship, between `ybar` and `Day`.
- The standard deviation, `SD` increases with `Day`, and hence with `yhat`.
- `n` seems to peak near the middle of the experiment.
- In fitting models we should take into account the fact that some measurements are based on more observations of shoots than others.
- We can easily represent `n` as point size in `ggplot2`. First create the base plot

```
1 p2 <- qplot(Day, ybar, data=longshoots,
2             xlab="Days from dormancy",
3             ylab="Average number of stem units per shoot")
```

yhat vs. Days with point size proportional to n

```
1 print(p2 + aes(size=n), TRUE, viewport(0.5,0.5,0.8,1))
```



The allshoots data

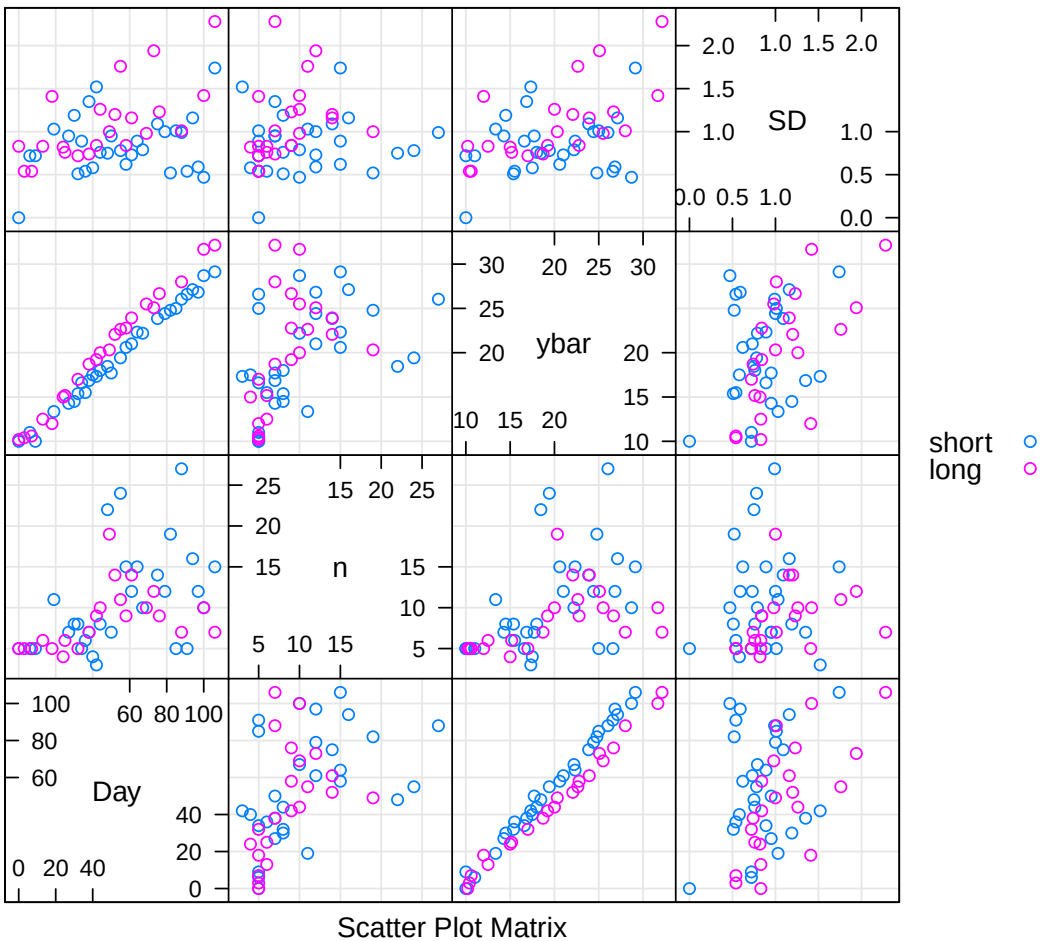
- The allshoots data includes measurements on both the long and the short shoots. The Type variable should be a factor and we convert it

```
str(allshoots <-
    within(allshoots,
           Type <- factor(Type,
                          labels=c("short","long"))))
```

```
'data.frame':        52 obs. of  5 variables:
$ Day : int  0 6 9 19 27 30 32 34 36 38 ...
$ n   : int  5 5 5 11 7 8 8 5 6 7 ...
$ ybar: num  10 11 10 13.4 14.3 ...
$ SD  : num  0 0.72 0.72 1.03 0.95 1.19 0.51 0.89 0.54 1.3
$ Type: Factor w/ 2 levels "short","long": 1 1 1 1 1 1 1 1
```

A splom plot with a categorical covariate

```
lattice::splom(~ allshoots[,1:4], allshoots,
                 type=c("p","g"), groups=Type,
                 auto.key=list(space="right"))
```



Scatter Plot Matrix

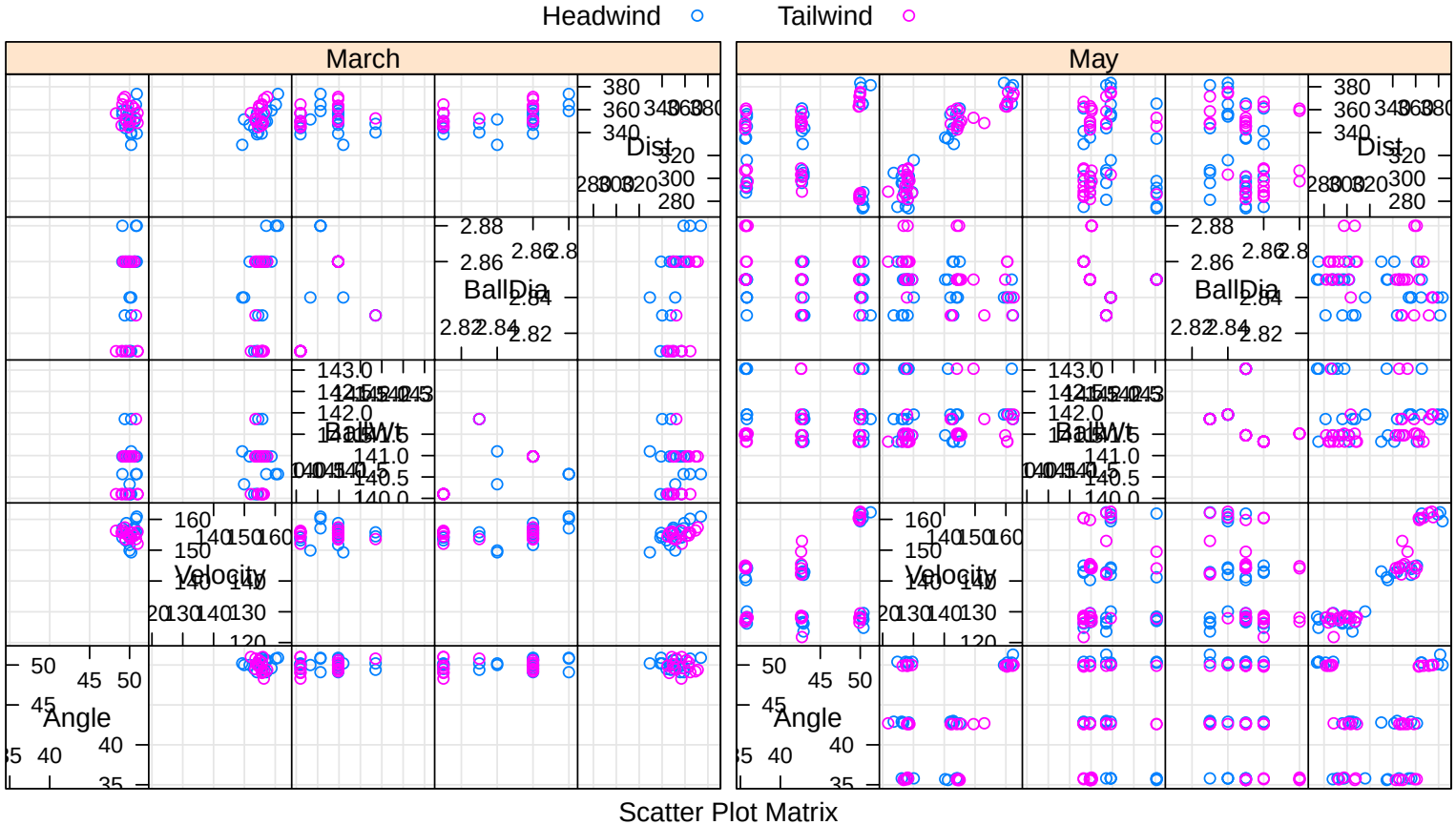
domedata1 from the alr3 package

The domedata1 data frame contains the results of two experiments to see if manipulating the air conditioning fans in the Minneapolis Metrodome can affect the distance travelled by a baseball

```
1 str(domedata1)
```

```
'data.frame': 96 obs. of  7 variables:
 $ Date    : Factor w/ 2 levels "March","May": 1 1 1 1 1 1
 $ Cond    : Factor w/ 2 levels "Headwind","Tailwind": 1 1
 $ Angle   : num  49.1 49.1 49.4 49.5 49.6 49.9 50 50 50 50
 $ Velocity: num  154 157 156 153 159 ...
 $ BallWt  : num  141 141 142 140 141 ...
 $ BallDia : num  2.86 2.88 2.83 2.81 2.86 2.86 2.81 2.84 2
 $ Dist    : num  356 359 348 344 359 ...
```

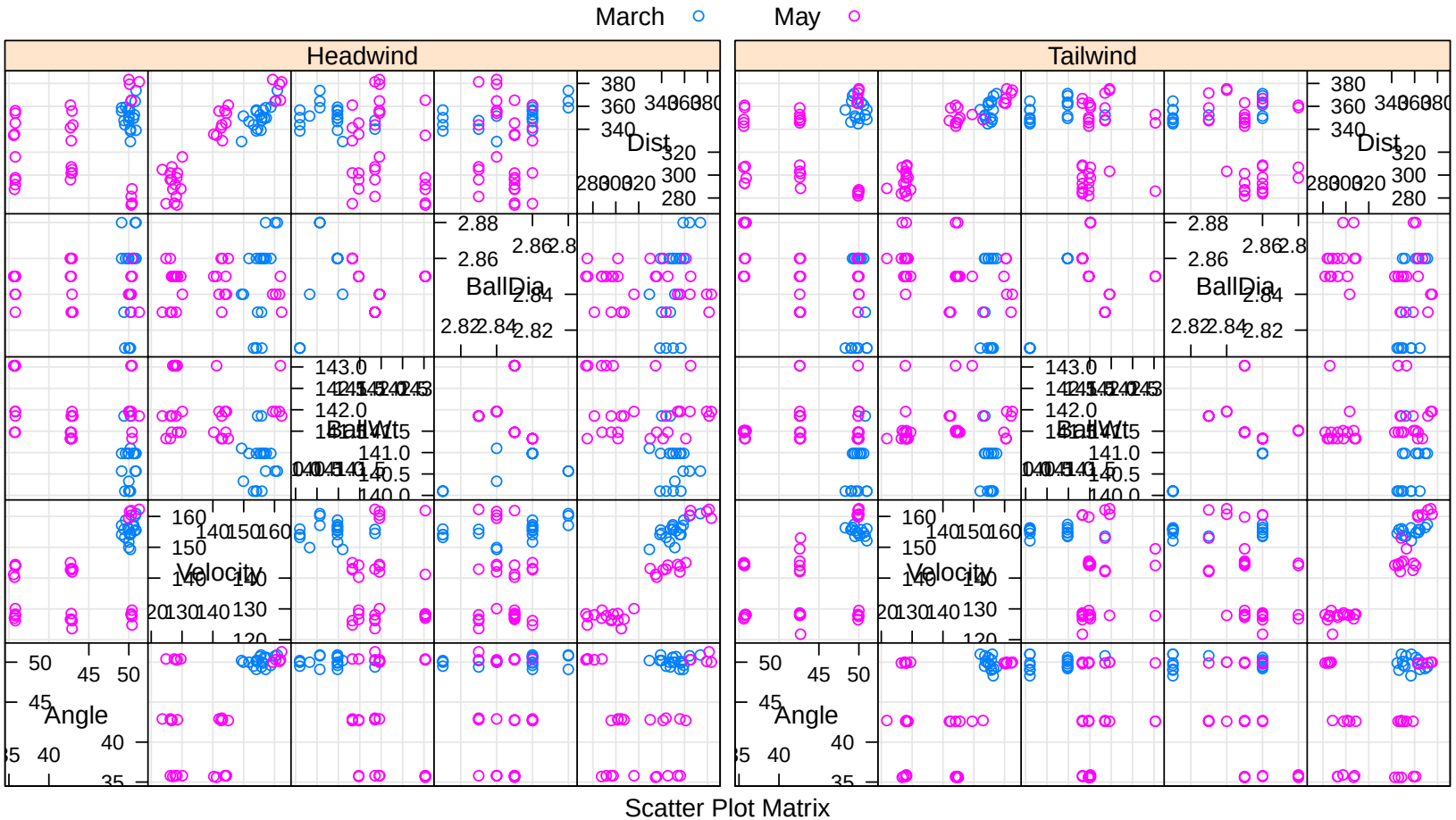
Domedata - splom



Scatter Plot Matrix

Three `Velocity` ranges were used in May but only 1 in March, strongly affecting the distance - same with `Angle`. The `BallWt` ranges were different in March and May.

Splom with colors for Date and panels by Cond



Scatter Plot Matrix

Setting up the basic plots

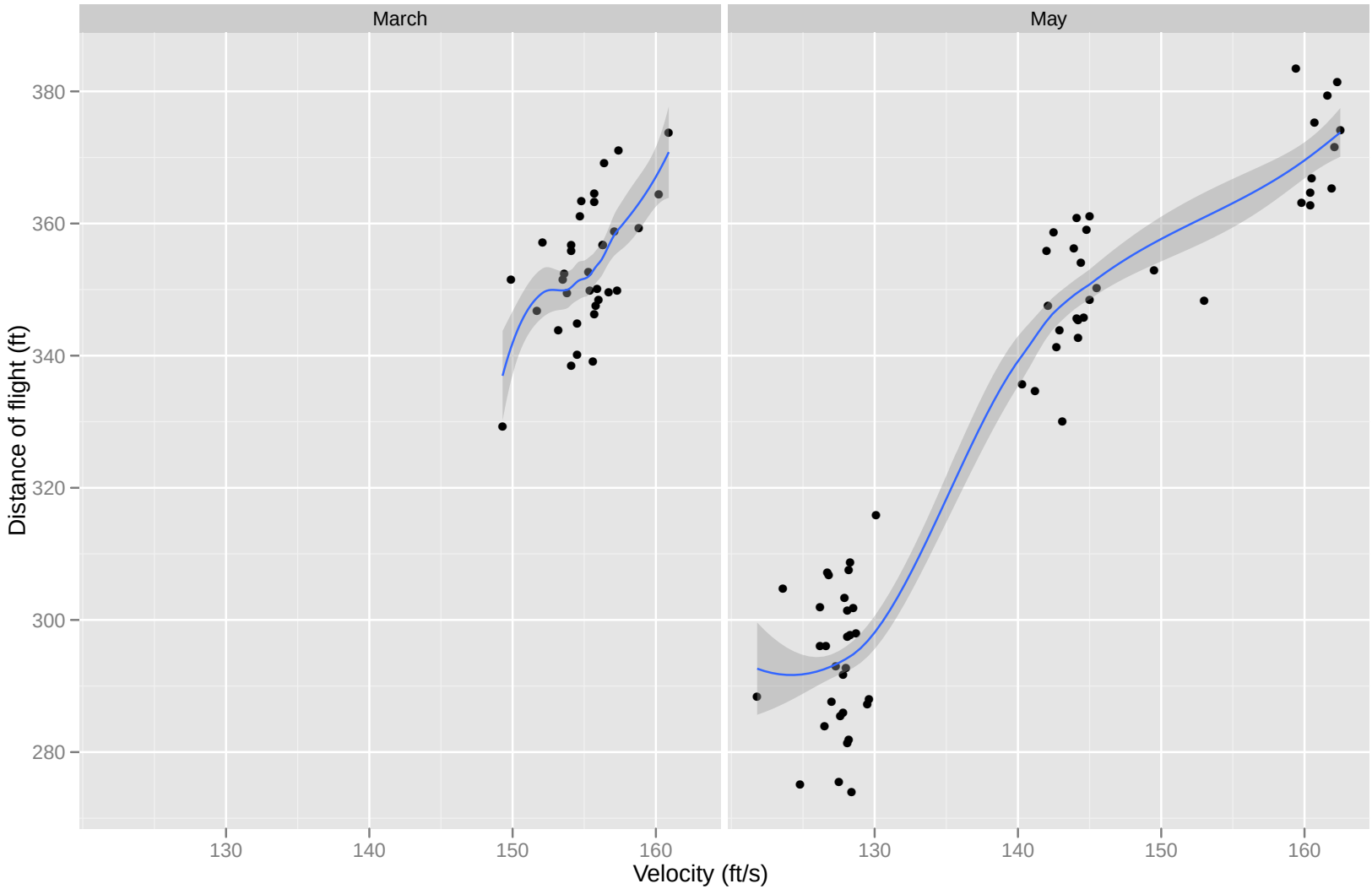
We set up plots for all the data and for each Date

```
dm3 <- subset(domedata1, Date=="March")
dm5 <- subset(domedata1, Date=="May")
xl <- "Velocity (ft/s)"
yl <- "Distance of flight (ft)"
sm <- geom_smooth()  # default scatterplot smoother
smlm <- geom_smooth(meth="lm") # regression line
p <- qplot(Velocity, Dist, data=domedata1,
           xlab=xl, ylab=yl)
p3 <- qplot(Velocity, Dist, data=dm3,
            xlab=xl, ylab=yl)
p5 <- qplot(Velocity, Dist, data=dm5,
            xlab=xl, ylab=yl)
```

Now examine this relationship by Date and by Date and Cond

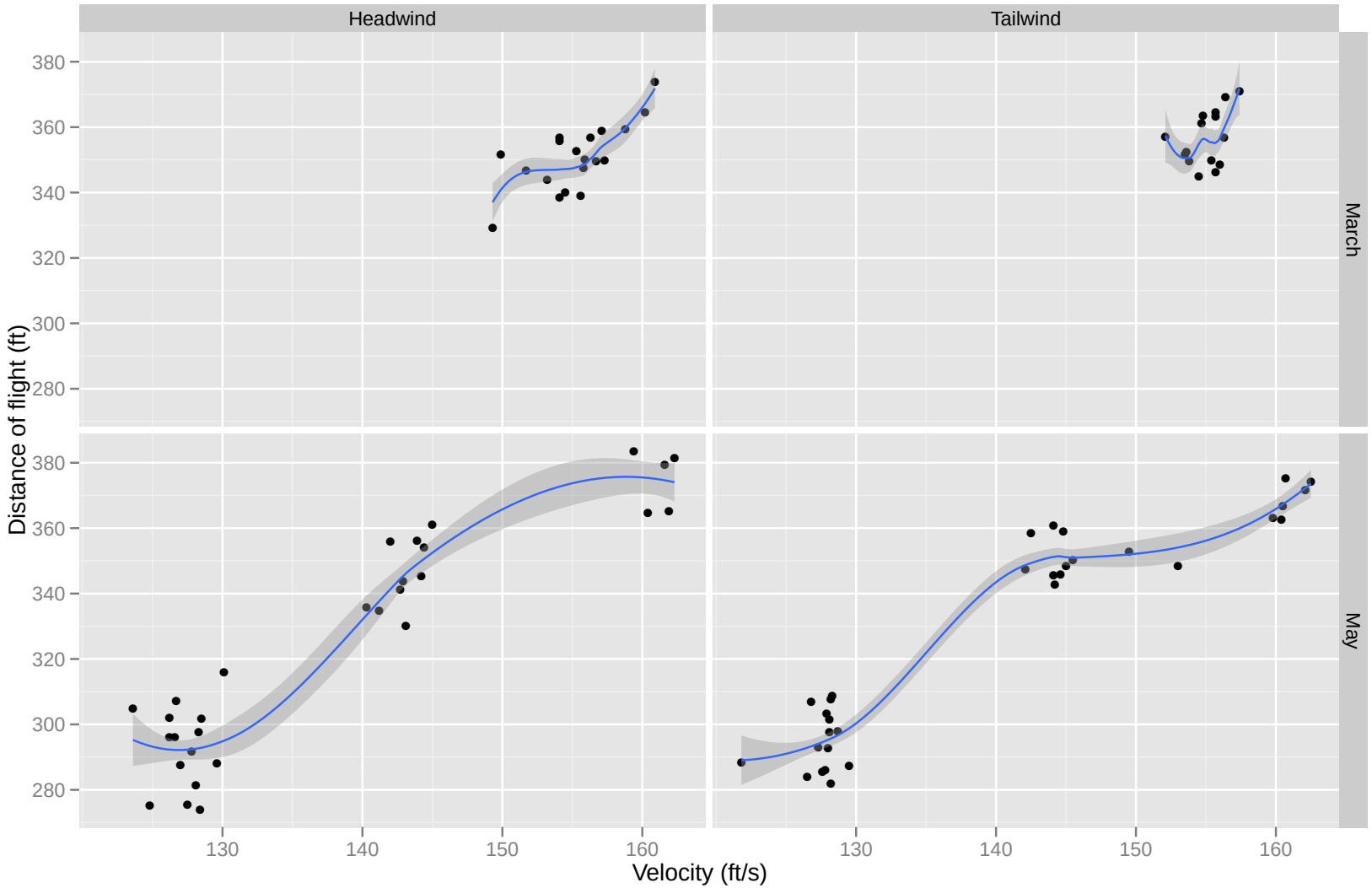
Distance *vs.* Velocity by Date

```
1  p+facet_grid(~Date)+sm
```



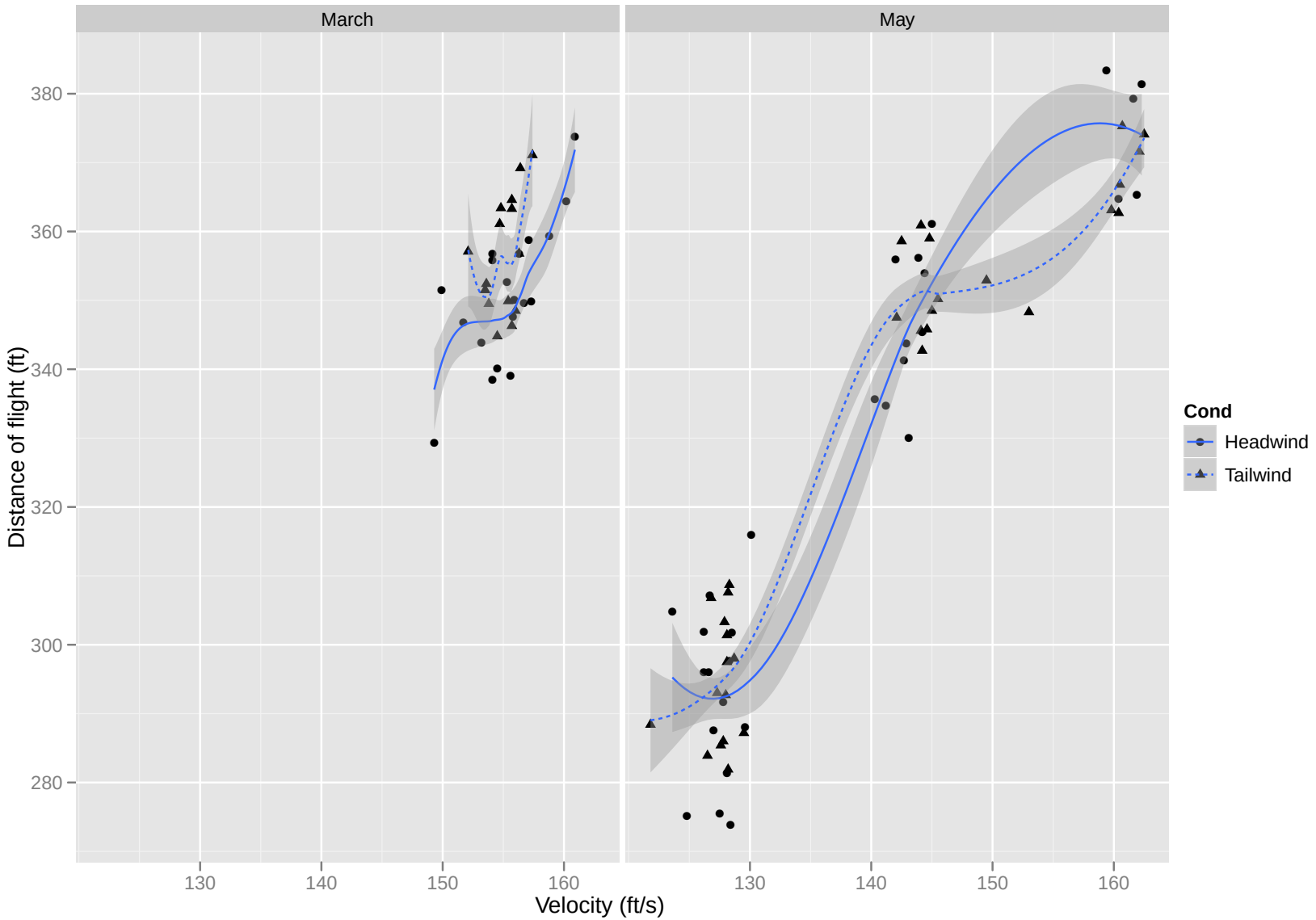
Distance *vs.* Velocity by Date and Condition

```
1  p+facet_grid(Date~Cond)+sm
```



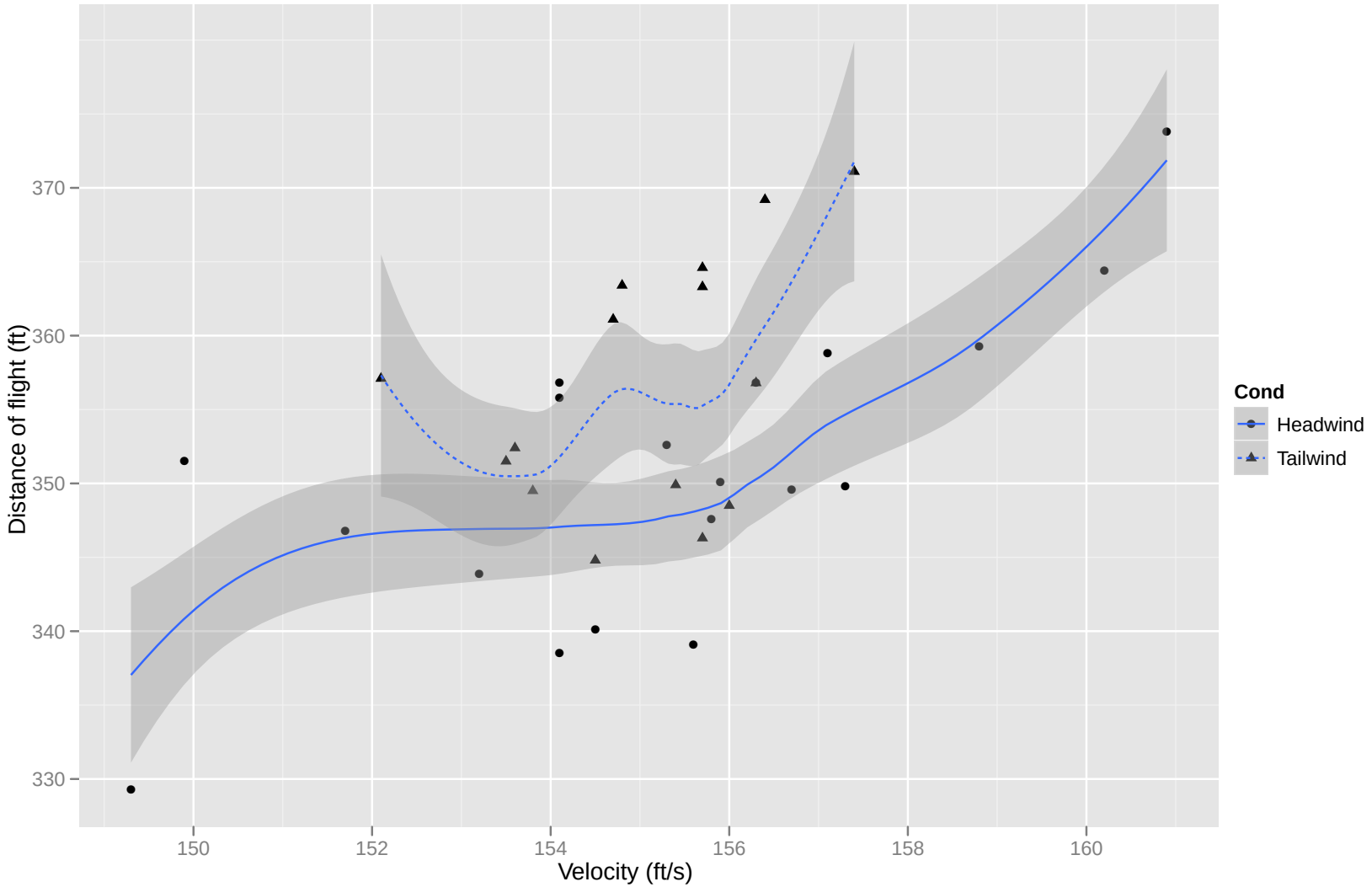
Distance *vs.* Velocity by Condition

```
1 p+aes(shape=Cond, linetype=Cond)+facet_grid(.~Date)+sm
```



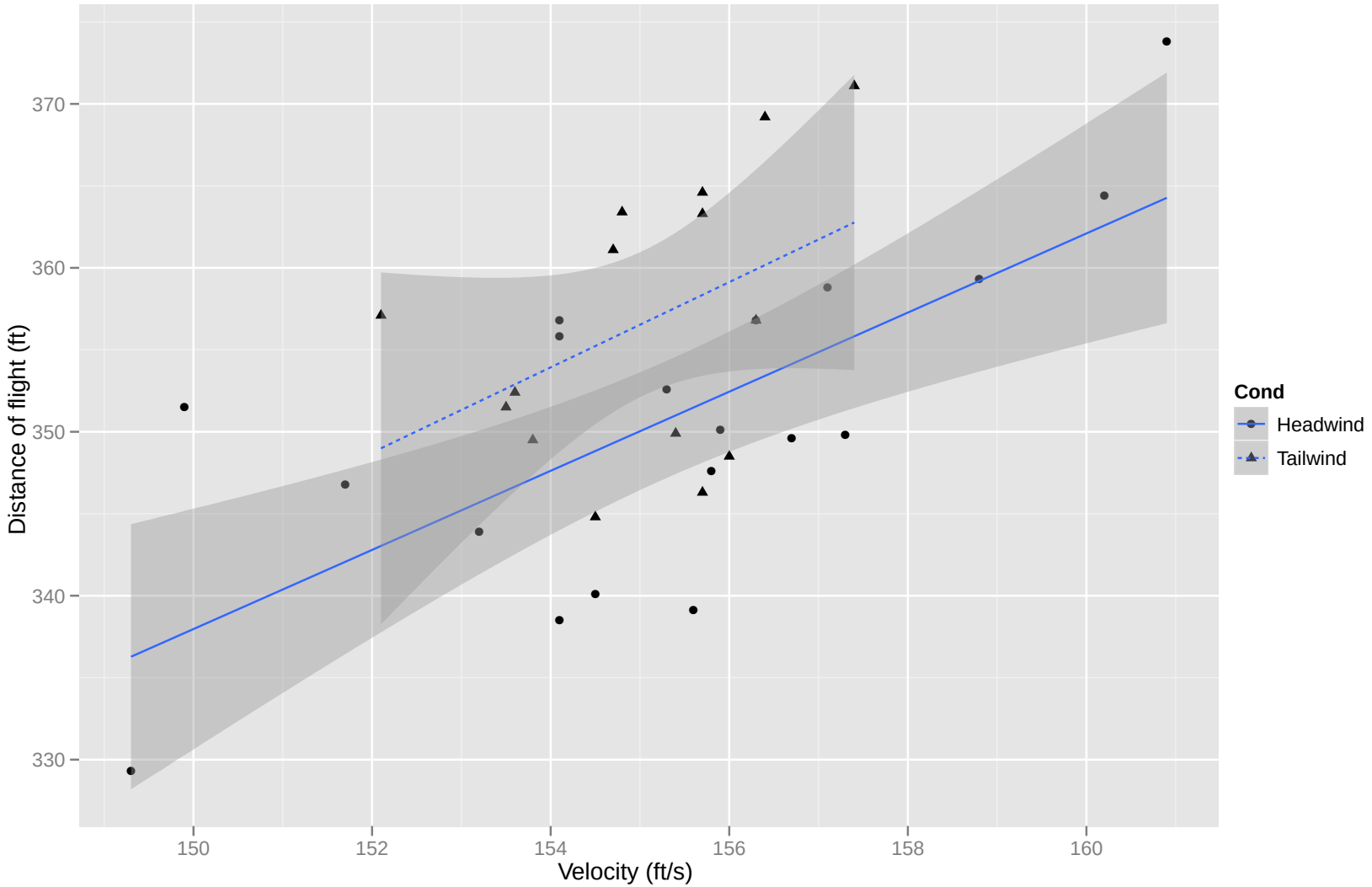
March Distance *vs.* Velocity by Condition

```
1 p3+aes(shape=Cond, linetype=Cond)+sm
```



March Distance *vs.* Velocity by Condition

```
1 p3+aes(shape=Cond, linetype=Cond)+smlm
```



Summary

- The importance of good data graphics for determining and for conveying the structure of data cannot be overemphasized.
- Reports of a statistical analysis of data, such as Master's Exam reports in our department, should always contain informative data graphics. Your most important conclusions should be presented in a "key graph" early in a report, with informative axis labels and a caption that allows it to be understood by itself. Busy managers may just skim the report but they always look at the figures.
- Informative, attractive data plots can be prepared reasonably quickly with `ggplot2`. The structure of the package allows you to build the plot in layers.
- We also use the `splom()` function from the `lattice` package for exploratory graphics. Most of the time it is not a good choice for presentation graphics.