Chapter 2: Summarizing data

Introductory Statistics for Engineering Experimentation

Peter R. Nelson, Marie Coffin and Karen A.F. Copeland

Slides by Douglas Bates

Outline

Section 2.1: Univariate data

- Graphs can
  - summarize the data
  - show typical and atypical values
  - highlight relationships between variables
  - show how the data are spread out, which we call the *distribution* of the data
- Often we observe multiple characteristics on each experimental run or observation. We call such data *multivariate*. If we only observe one characteristic we say the data are *univariate*.
- Typical plots of univariate, numeric data are *histograms* (`histogram`) or *density plots* (`densityplot`), *box-and-whisker plots* (`bwplot`) and *dotplots* (`dotplot`). (Names in parentheses are the names of the corresponding *R* function from the `lattice` package.)

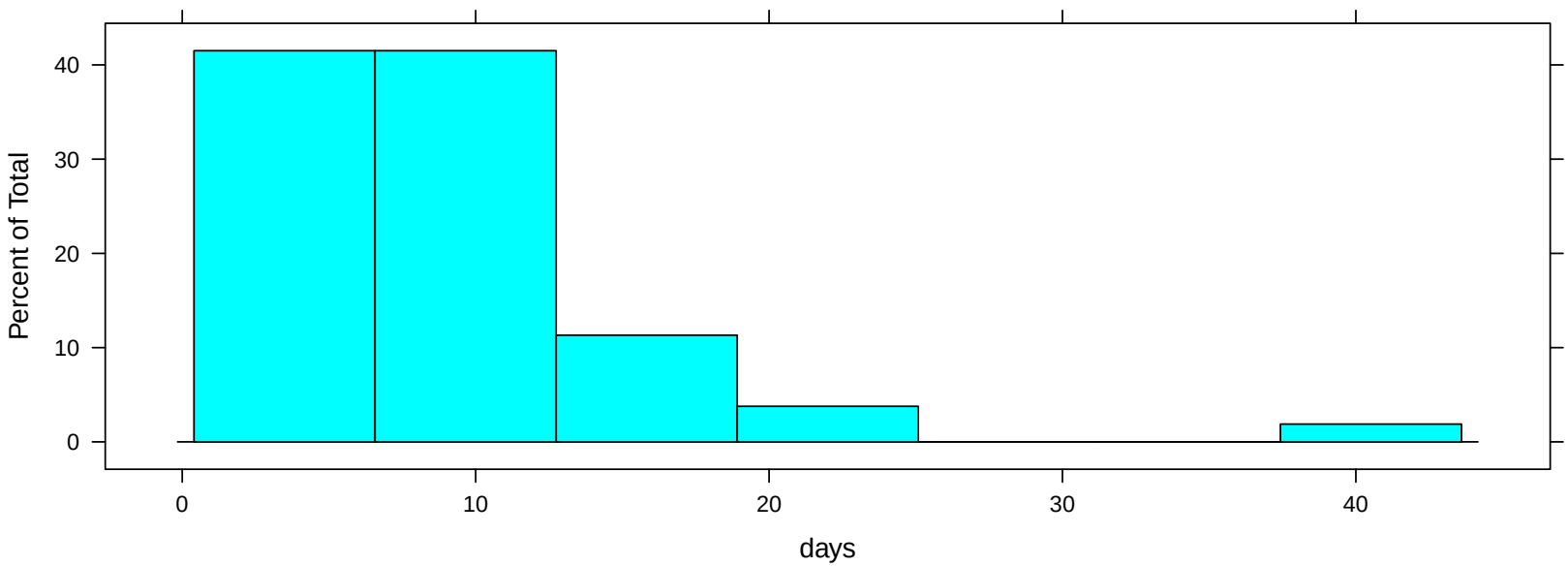
The railcar data

- Example 2.1.1 describes the `railcar` data, which are 53 observations of the number of days that a customer holds a rail car after delivery. The authors say the data are in a file named *railcar.txt*. We will use the dataset called `railcar` from the `EngrExpt` package for *R*.
- All of the data sets are described in Appendix A, starting on page 424. When you start *R* you should attach the EngrExpt package, using `library(EngrExpt)`. To check on the form of a data set use, e.g., `str(railcar)`, and compare the result to the description in Appendix A.

```
> library(EngrExpt)
> str(railcar)

'data.frame': 53 obs. of  1 variable:
 $ days: int  4 42 4 4 3 5 5 5 3 7 ...
```

Histograms

- A histogram is a simple bar chart of the number of observations in each of a set of adjacent, constant-width intervals.
- It was popular in the days when all graphics, and most calculations, needed to be done by hand. It shows the distribution of the data (see the comments in example 2.1.1).



Creating a histogram

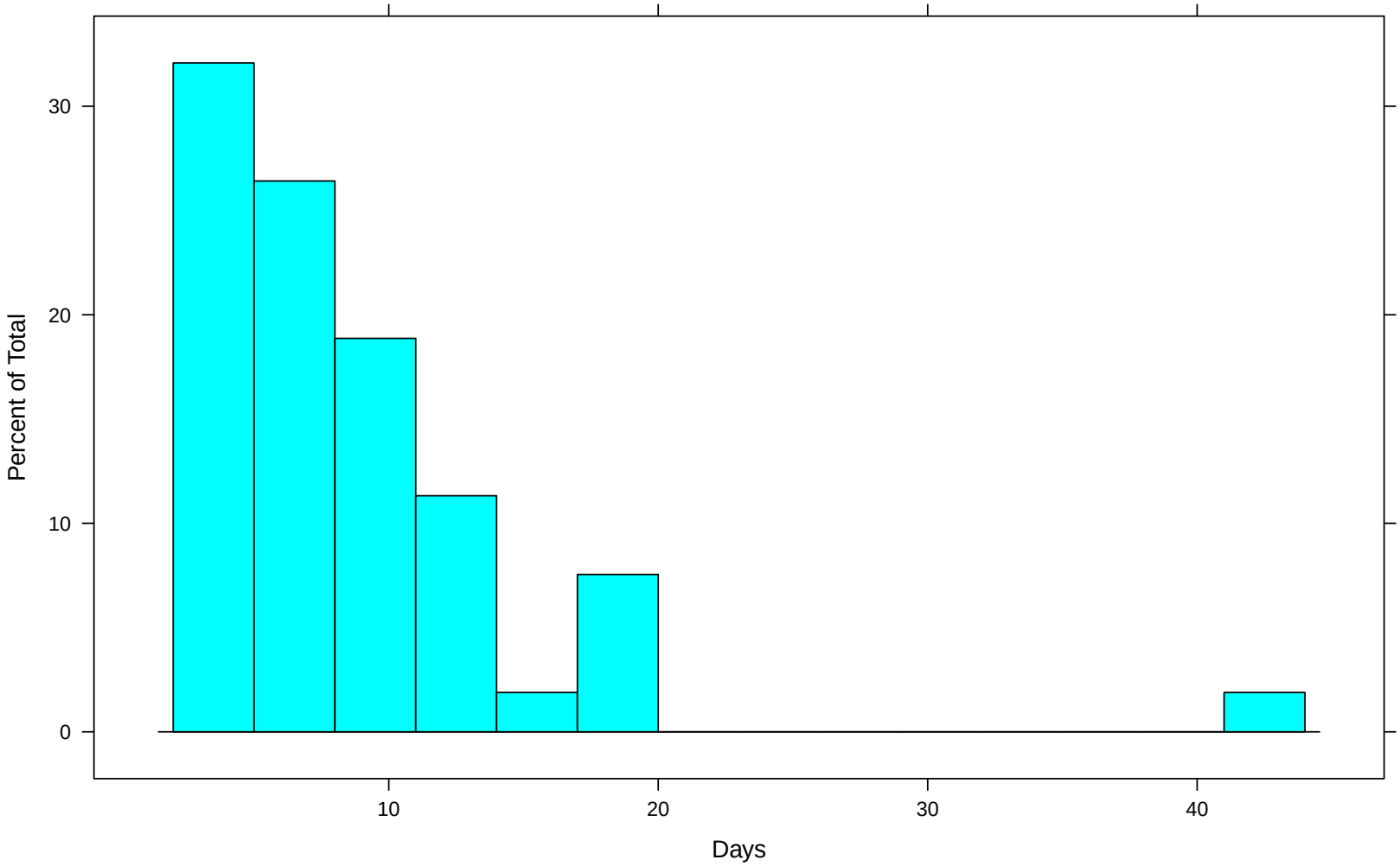
- The text gives a description of creating a histogram by hand.
- In *R*, you can use the `histogram` function. The plot on the previous slide was created with

  ```
  > histogram(~ days, railcar)
  ```

- There are several optional arguments for the `histogram` function. To create a plot like Figure 2.1 we specify the break points for the intervals (argument `breaks`) and also change the label on the x-axis (argument `xlab`).

```
> histogram(~ days, railcar, breaks = seq(2, 44, 3), xlab = "Day
```

Histogram like Figure 2.1

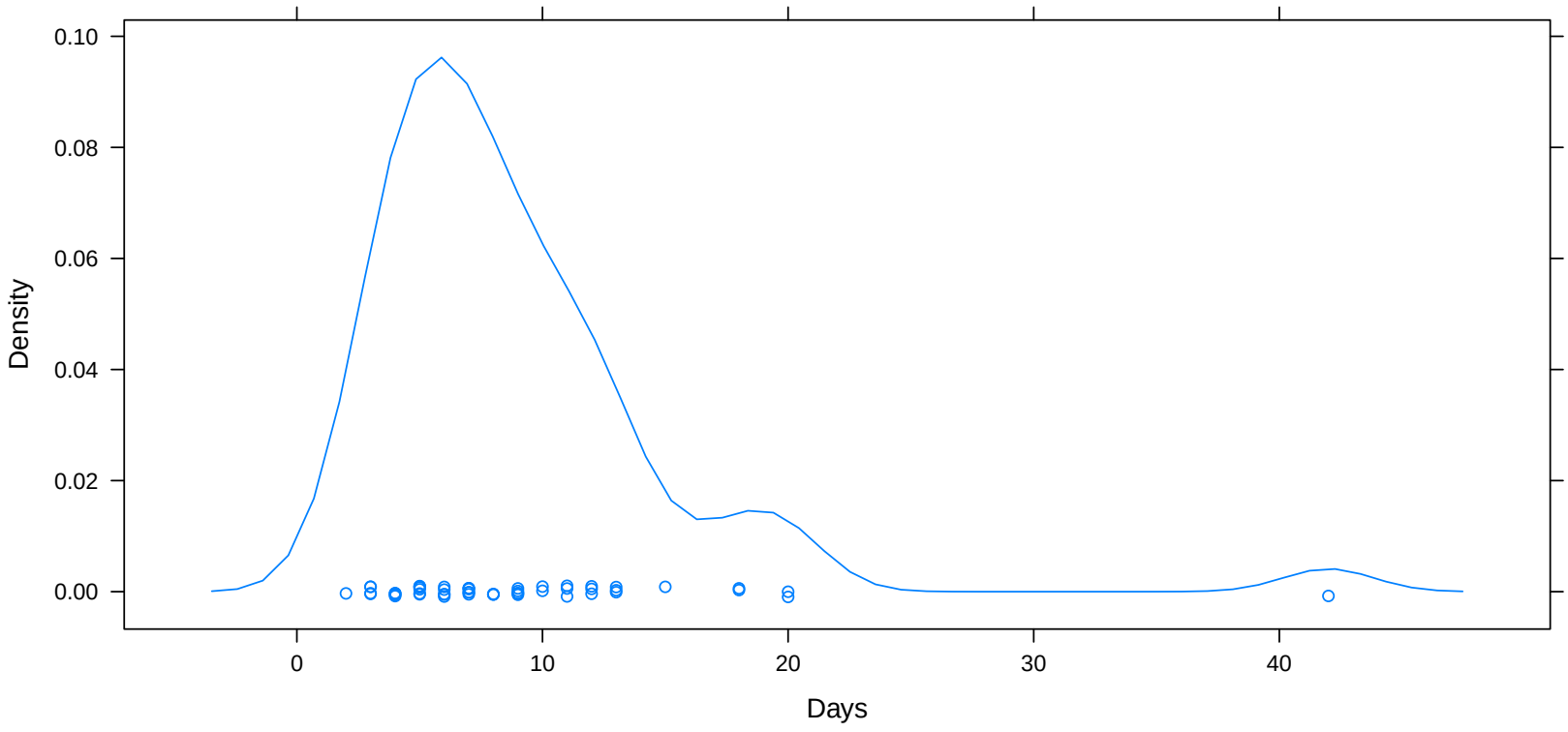


Density plots

- This topic is not covered in the text. It requires more sophisticated software than they were using.
- Notice that the histogram shape depends on the somewhat arbitrary choice of intervals.
- If we are interested in the shape of the distribution of the observations we can use an alternative called a density plot.
- Without going into details, an empirical density plot centers a narrow, "bell-curve" density at each observed data point and sums the result. The version in *R* also adds a "rug" with the original data values plotted as points and jittered vertically (so you can see multiple data points with the same value).

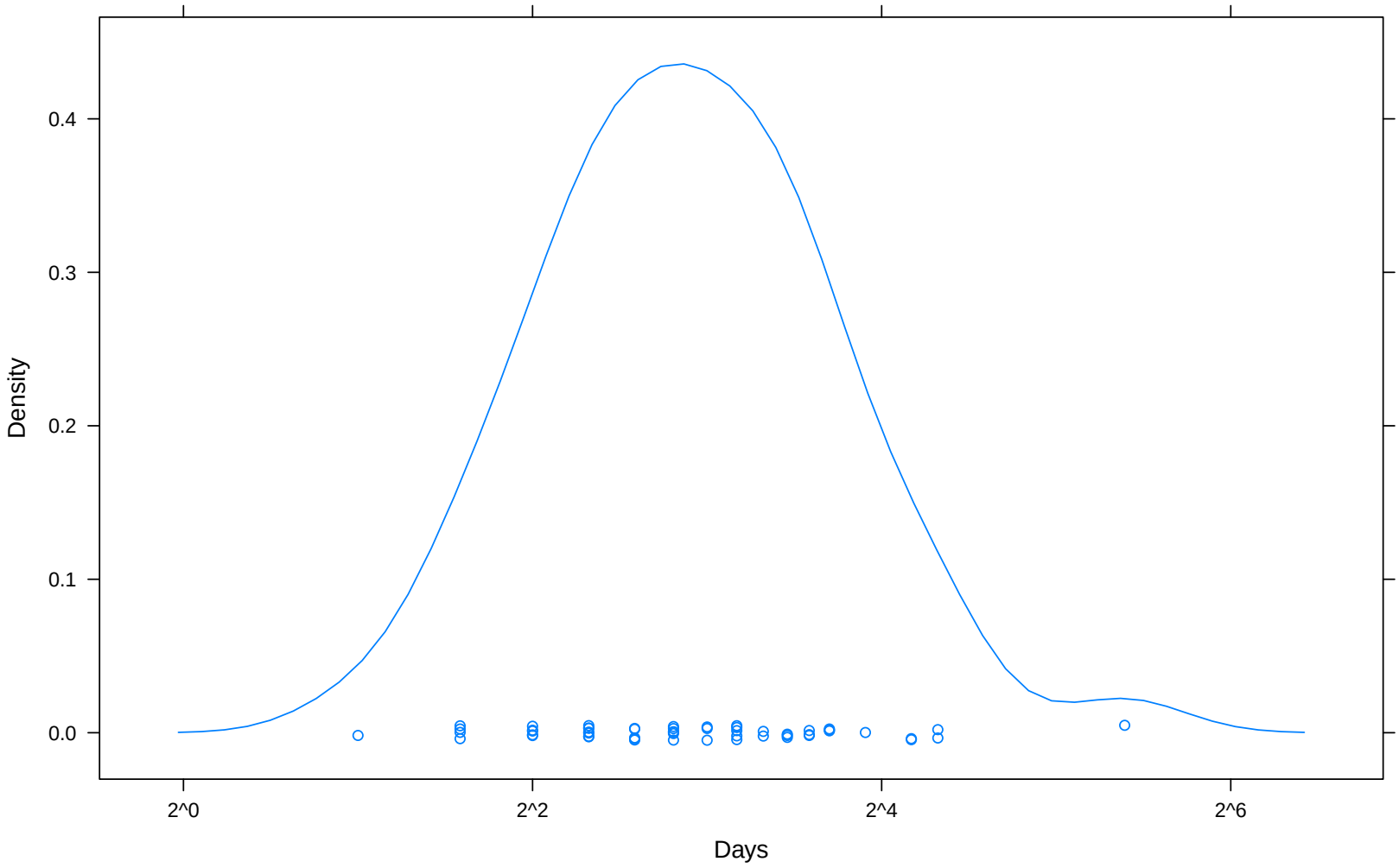
```
> densityplot(~ days, railcar, xlab = "Days")
```

Density plot of the rail data.



The skewness of this plot indicates that we may want to consider the logarithm of the days.

Density plot of the rail data (logarithmic scale).



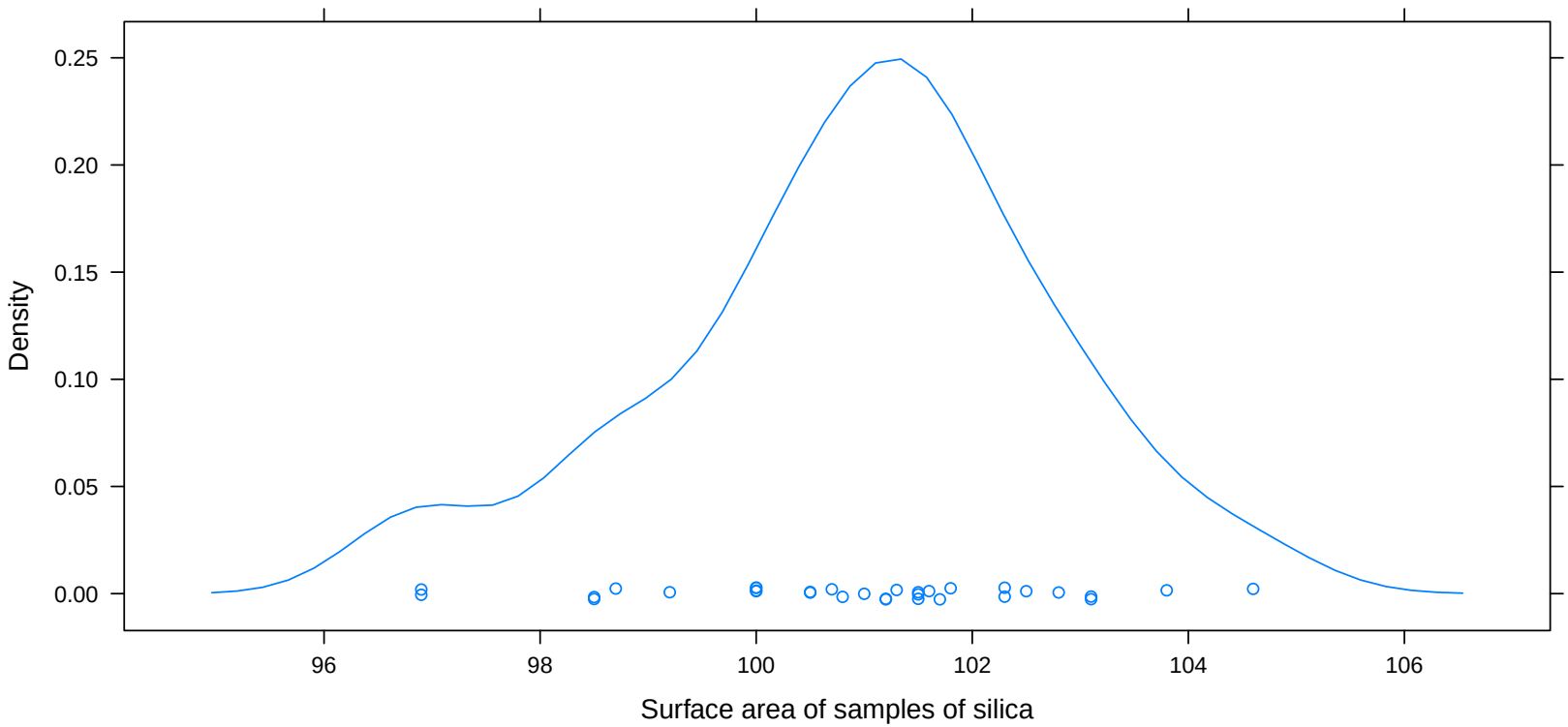
Silica surface area data, example 2.2

- The `surfarea` data are measurements of the surface area of samples of silica.

```
> str(surfarea)

'data.frame': 32 obs. of  1 variable:
 $ area: num  102 100 101 103 104 ...

> summary(surfarea)

      area
 Min.   : 96.9
 1st Qu.:100.0
 Median :101.2
 Mean   :100.9
 3rd Qu.:101.9
 Max.   :104.6

> ## alternative form for univariate data
> with(surfarea, summary(area))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   96.9   100.0   101.2   100.9   101.9   104.6
```

Comparing a histogram and density plot for surfarea

- The histogram shown in Figure 2.2 indicates a very flat distribution.
- A density plot shows more of a "bell-curve" distribution.



Plots of bivariate data

- Although we do analyze univariate data, in practice we are often interested in "input-response" data, the simplest form of which is *bivariate* (two variables).
- The `xyplot` function produces a scatterplot of data values based on a formula in the first argument. Many optional arguments can be used; for us the `type` and `groups` arguments are important.
- The `timetemp` data describe the time for a panel to get to the testing temperature as a function of the temperature of the freezer in which is was stored. There are two types of panels, OEM and repaired. See

```
> ?timetemp
```

Subsetting the data

```
> str(timetemp)
```

```
'data.frame': 24 obs. of  3 variables:
 $ time: int  5 11 16 9 14 6 6 16 13 11 ...
 $ temp: num  -22.7 -25.5 -29 -24.8 -27.2 -23.5 -23.9 -28.7 -26.8 -25.9
 $ type: Factor w/ 2 levels "Repaired","OEM": 1 1 1 1 1 1 1 1 1 1 ...
```

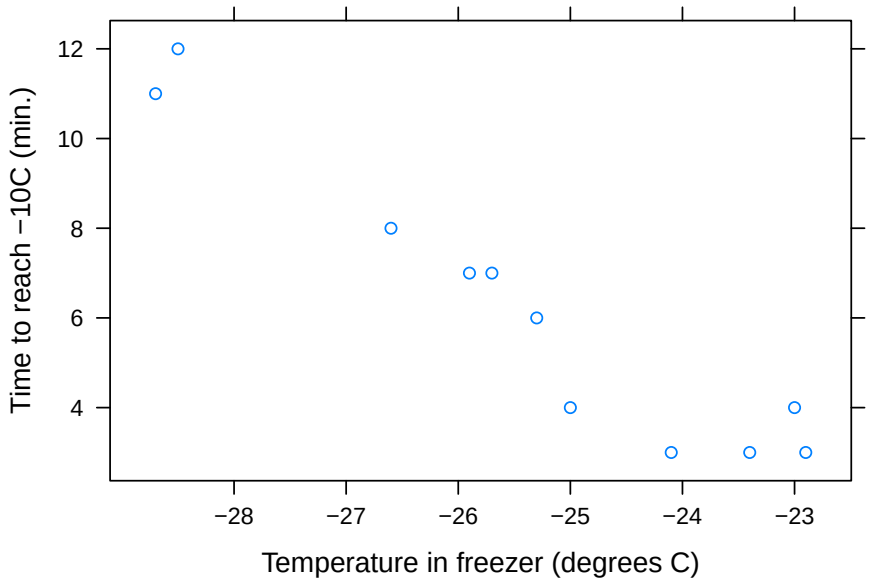
- Initially we want to create a plot of just the OEM panels (Figure 2.5). We can use the `subset` function to extract these panels only. Note that the equality comparison is `==`, not just `=`.

```
> OEM <- subset(timetemp, type == "OEM")
```

- A simple plot is obtained with

```
> xyplot(time ~ temp, OEM)
```

Scatter-plot of OEM panel data only



- An alternative way to generate this plot is to use the `subset` argument in the call to `xyplot`.

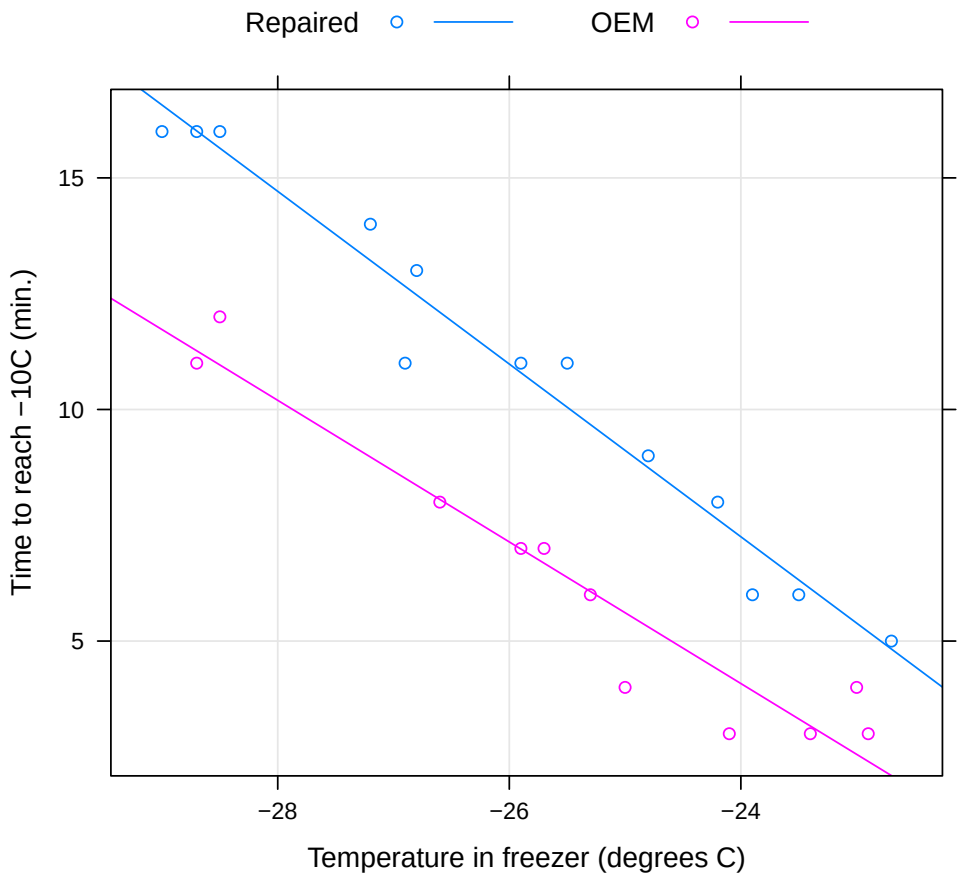
```
> xyplot(time ~ temp, timetemp, subset = type == "OEM")
```

Overlaying multiple groups

- Frequently we want to compare the behavior of two or more groups, such as the OEM panels versus the repaired panels, which have an extra coat of paint.
- The `xyplot` function provides two ways of doing this. Using the `groups` argument provides different symbols and/or line types in the same panel. Usually we use the `auto.key` argument as well to add a key to the plot so we know which symbol corresponds to which group.
- An alternative it to use multiple panels (but the same axis ranges on each panel to facilitate comparison). This is done by modifying the formula.

Different groups on the same plot panel

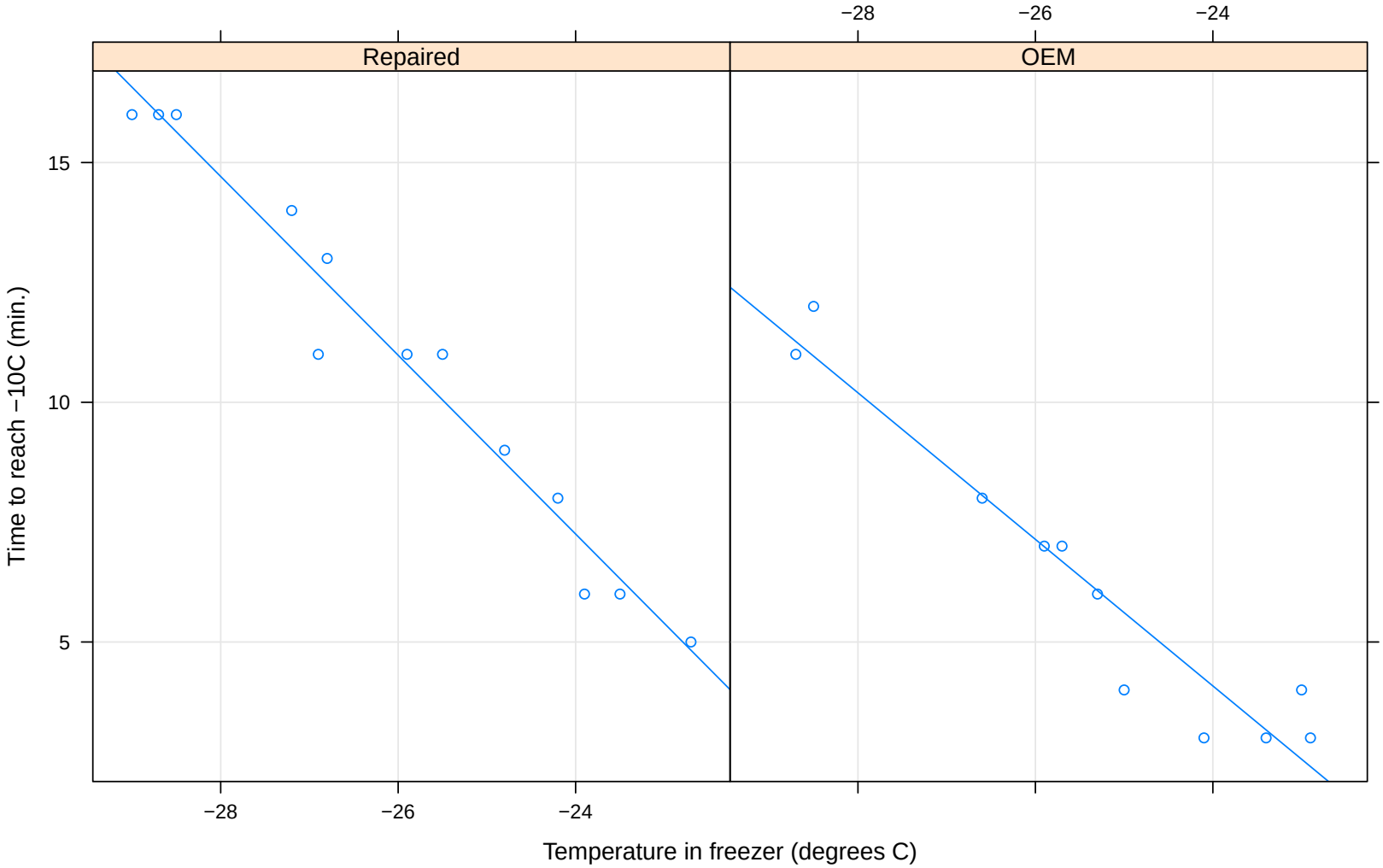
```
> xyplot(time ~ temp, timetemp, groups = type, type = c("g","p",
```



We added "reference" lines to facilitate comparison.

Separate plot panels with common axes

```
> xyplot(time ~ temp | type, timetemp, type = c("g","p","r"))
```



Runs charts or time-series plots

- In an `xyplot` we typically plot a response versus a *covariate*. Often the most important covariate in a production process is the time order in which units are produced.
- This type of data are called *time-series* data. When we plot such data we usually join the points because the order is important.
- Sometimes there is a variable in the data frame indicating the order. Often we omit it because it is redundant. We can generate a sequence of indices starting at 1 with the `seq_along` function.
- The `absorb` data set is such a time series of measurements on samples taken in one shift.

Runs chart of the absorb data

```
> xyplot(absorb ~ seq_along(absorb), absorb, type = "b")
```