Chapter 2: Summarizing data

Introductory Statistics for Engineering Experimentation

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

Slides by Douglas Bates

Outline

Outline

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

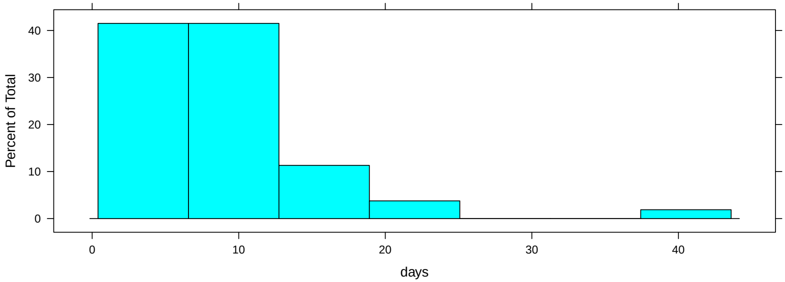
- Example 2.1.1 describes the `railcar` data, which are 53 observations of the number of days that a customer holds a rail car after delivery. The authors say the data are in a file named *railcar.txt*. We will use the dataset called `railcar` from the `EngrExpt` package for *R*.
- All of the data sets 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).

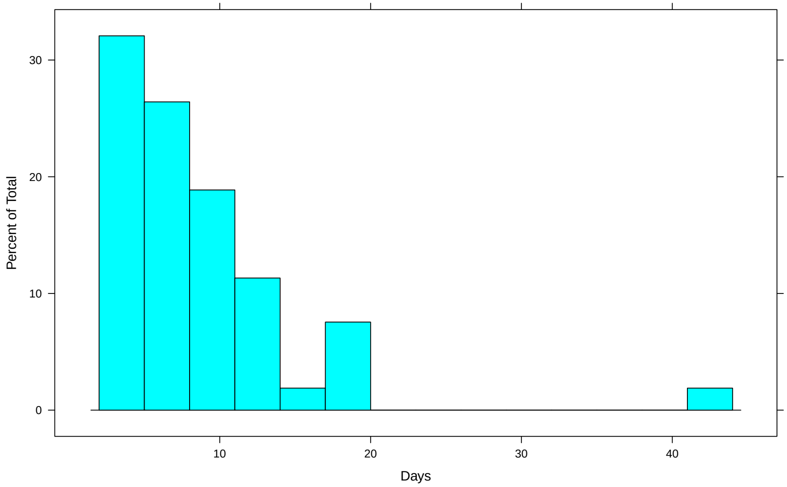


Creating a histogram

- 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

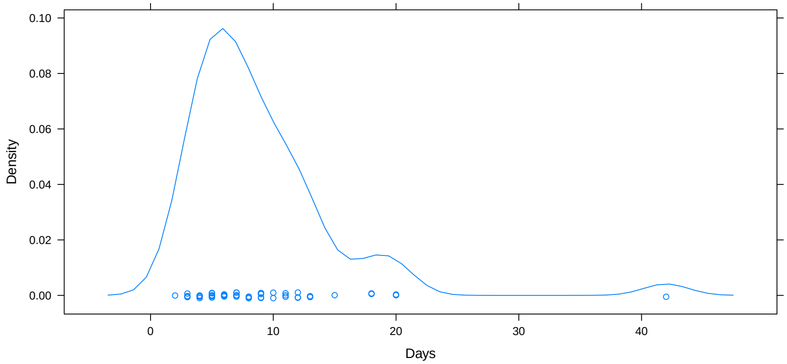


Density plots

- 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).

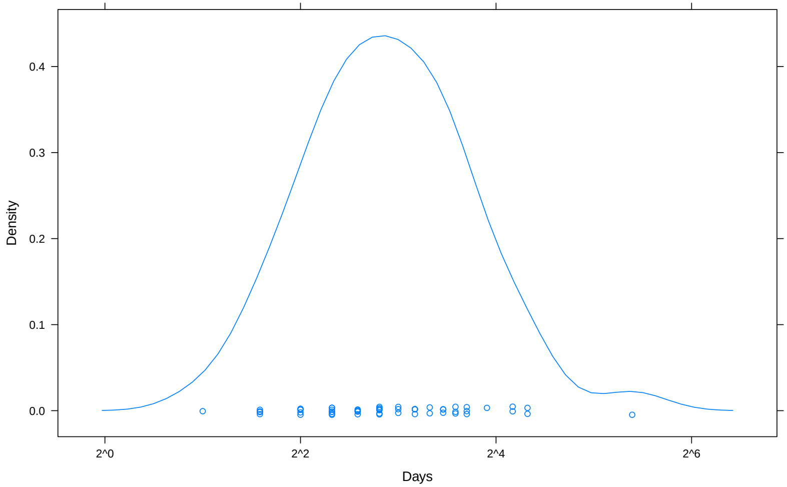
```
> densityplot(~ days, railcar, xlab = "Days")
```

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).



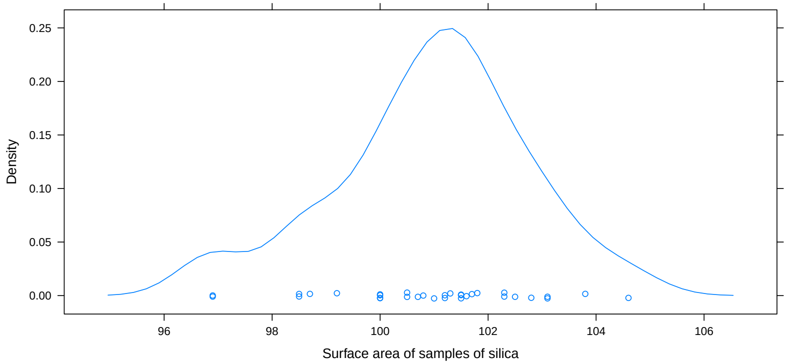
Silica surface area data, example 2.2

- 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 ...
```

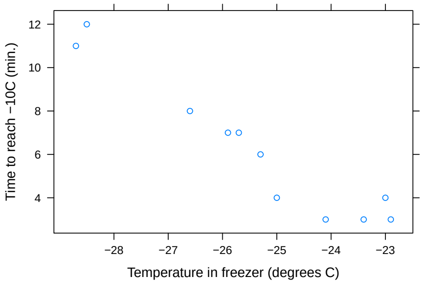
- Initially we want to create a plot of just the OEM panels (Figure 2.5). We can use the `subset` function to extract these panels only. Note that the equality comparison is `==`, not just `=`.

```
> 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`.

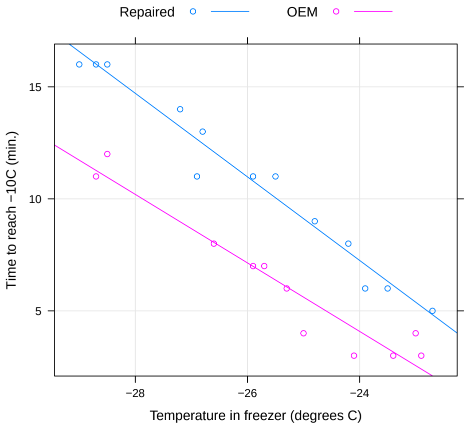
```
> xyplot(time ~ temp, timetemp, subset = type == "OEM")
```

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

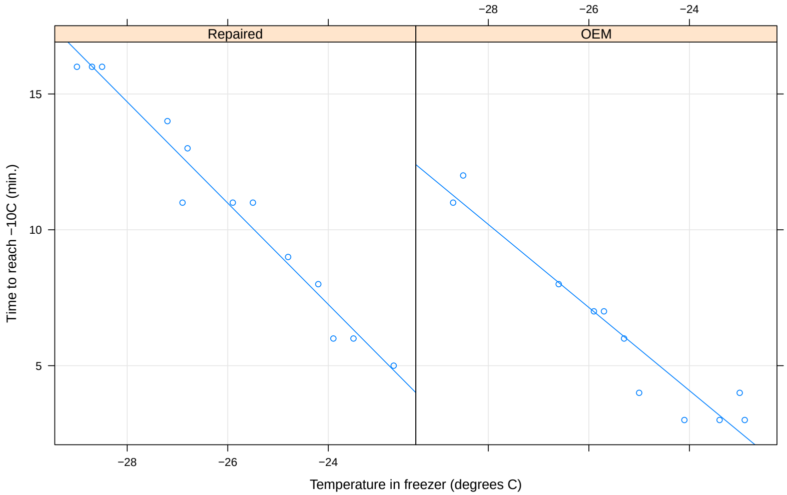
```
> xyplot(time ~ temp, timetemp, groups = type, type = c("g","p",
```



We added “reference” lines to facilitate comparison.

Separate plot panels with common axes

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

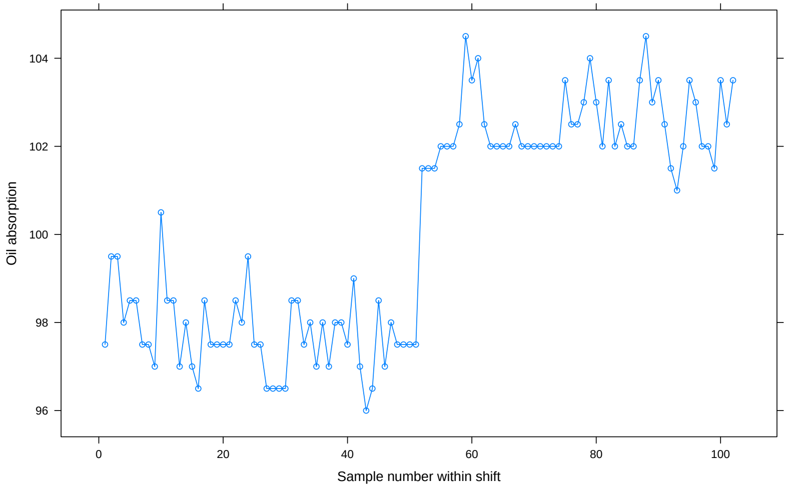


Runs charts or time-series plots

- 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")
```



Measures of locations

- In addition to graphical representations of data we can also provide numerical summaries.
- Typically we provide measures of location, such as the sample mean
$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$
or the sample median, which is the middle data value (formula 2.2.2). Notice that both of these are given in the output of the `summary` of a numeric variable.
- The sample mean is the point at which the data could be balanced. As such it uses all the information in the data but is sensitive to outliers.
- The sample median is not sensitive to outliers, which is why it is used to summarize highly skewed distributions such as distributions of salaries.

Sample quantiles

- The median is an example of a sample *quantile*, written $q(p), 0 < p < 1$ in the text (formula 2.2.4). The median is $q(0.5)$.
- The first and third quartiles are $q(0.25)$ and $q(0.75)$. That is, they are points that divide the data with $\frac{1}{4}$ below and $\frac{3}{4}$ above and vice versa. The quartiles are also given in the summary output as well as the minimum and maximum.

```
> with(subset(timetemp, type == "OEM"), summary(time))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   3.500   6.000   6.182   7.500  12.000

> with(railcar, summary(days))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   5.000   7.000   8.887  11.000  42.000

> with(surfarea, summary(area))

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

Measures of spread

- The simplest measure of spread of the data is the range of the data values. However, it is sensitive to the size of the sample and to outliers and not of much value in data analysis.
- The *inter-quartile range*

  $$\mathrm{IQR} = q(0.75) - q(0.25)$$

  is the width of the "middle half" of the data and provides a more reproducible measure of spread.
- The most commonly used measures of spread are the *sample variance*

  $$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

  and its square root, s, called the *sample standard deviation*.

Evaluating IQR, s, etc.

- The inter-quartile range can be evaluated from the summary output. There is also a function IQR to evaluate it directly.
- The function sd evaluates the sample standard deviation. (var evaluates the sample variance).
- Functions mean and median evaluate the corresponding measures of location.
- The function quantile evaluates an arbitrary quantile from the data.

```
> with(surfarea,
+      c(IQR = IQR(area), mean = mean(area),
+        s = sd(area), var = var(area),
+        Q1 = quantile(area, 0.25), Q3 = quantile(area, 0.75)))

       IQR       mean          s        var     Q1.25%
  1.925000 100.937500   1.796547   3.227581 100.000000
    Q3.75%
101.925000
```

Statistics and parameters

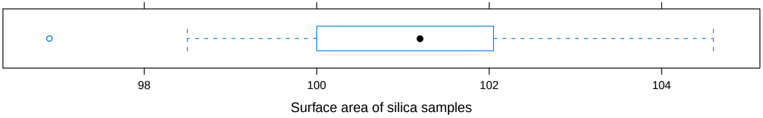
- Quantities like $\bar{y}$ and s are *sample statistics*. (Any value that can be calculated from the data alone is called a statistic.)
- When performing statistical inference we have a probability model for the distribution of the characteristic of interest within the population. This probability distribution has corresponding quantities which we call *population parameters*.
- We use Greek letters for the population parameters: the population mean is written μ and the population standard deviation is written σ.

Box-and-whisker plots

- A `boxplot` displays the a "five-number summary" (minimum, first quartile, median, third quartile, maximum) of a numeric variable.
- The box covers the region from the first to the third quartiles. Whiskers extend to the minimum and maximum of the data unless values are too far from the middle of the data and are considered potential outliers. The default definition of "too far" is more than 1.5 times the width of the box away from the box.
- Boxplots are frequently shown vertically, but this wastes space. I prefer the horizontal orientation, especially for comparative boxplots.

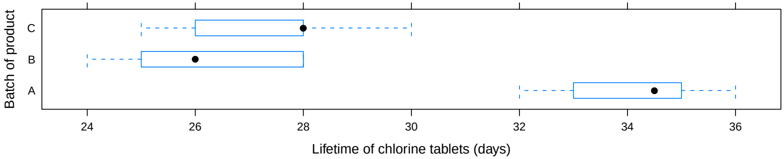
```
> bwplot(~ area, surfarea, xlab = "Surface area of Silica sample
```

Comparative boxplots



- To produce a comparative box-and-whisker plot, provide the categorical variable on the left hand side of the formula.

```
> bwplot(batch ~ time, tablets)
```



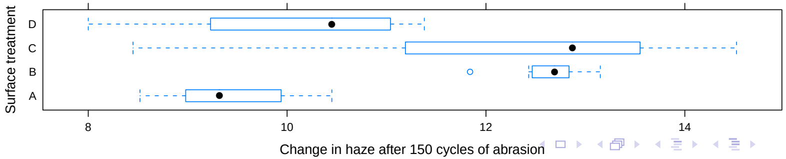
Designed experiments

- In a designed experiment certain variables or *factors* are systematically varied while observing the value of a response.
- If a covariate is categorical we store it as a data type called a factor. The values of a factor must come from a fixed set called the `levels` of the factor. Occasionally we will use *ordered factors* (data type is `ordered`) where the levels have a natural ordering.
- For a single factor we often use comparative boxplots to evaluate the change in the response with changing levels of the factor. Sometimes comparative dotplots or comparative density plots are more effective.

The optical lens coating data, example 2.3.1

- This is a balanced, designed experiment on the abrasion resistance of four different lens coatings (see description in text). The description *balanced* means that there were the same number of observations on each level of the factor.

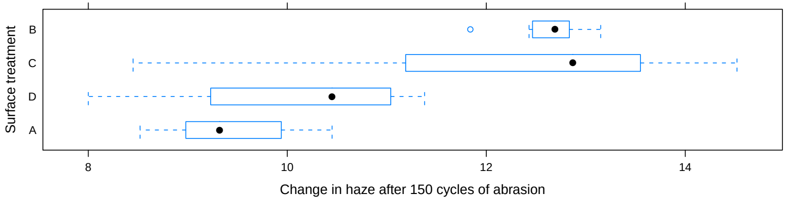
```
> str(dhaze)

'data.frame': 28 obs. of  2 variables:
 $ treatment: Factor w/ 4 levels "A","B","C","D": 1 1 1 1 1 1 1 2 2 2 .
 $ dhaze    : num  8.52 9.21 10.45 10.23 8.75 ...

> xtabs(~ treatment, dhaze)

treatment
A B C D
7 7 7 7
```



Re-ordering factor levels

- If the levels of a factor are in an arbitrary order we can impose an order to make the plot more informative. A common way of doing this is to order the levels by increasing mean response (function `reorder`)

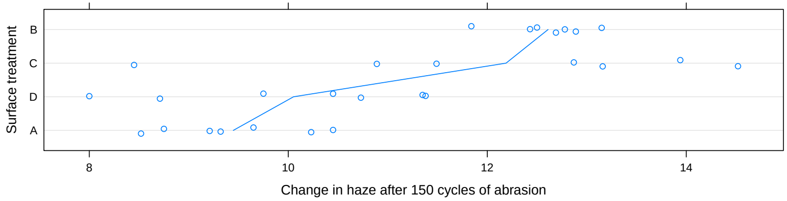
```
> bwplot(reorder(treatment, dhaze) ~ dhaze, dhaze)
```



Comparative dotplots

- Box-and-whisker plots reduce the data to 5 summary statistics. This can help to eliminate clutter in a plot but can also miss important characteristics. With only 7 observations per group like this we can plot the original data by group to see more detail.
- To avoid overplotting of points with the same response value, we "jitter" the y values and use an open circle (`pch=21`) as the plot character. We also use `type` to join the averages.

```
> dotplot(reorder(treatment, dhaze) ~ dhaze, dhaze, pch = 21,
+         jitter.y = TRUE, type = c("p", "a"))
```



Multi-factor experiments

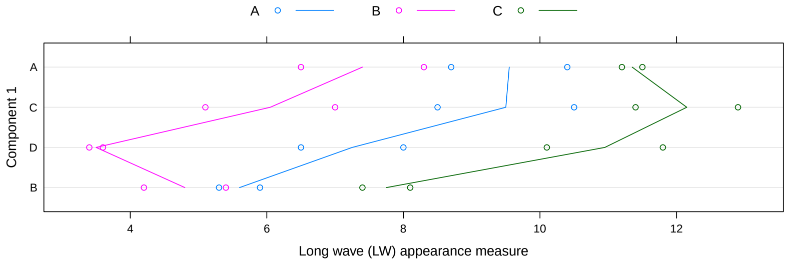
- In the `dhaze` data there is one experimental factor, `treatment`.
- In other experiments we may vary multiple factors, preferably in a *factorial design* where each combination of levels of factors has the same number of observations.
- We can use `xtabs` to determine the number of observations at each combination.

```
> str(lw)  # Example 2.3.2

'data.frame': 24 obs. of  3 variables:
 $ lw   : num  10.4 8.7 6.5 8.3 11.5 11.2 5.3 5.9 4.2 5.4 ...
 $ comp1: Factor w/ 4 levels "A","B","C","D": 1 1 1 1 1 1 2 2 2 2 ...
 $ comp2: Factor w/ 3 levels "A","B","C": 1 1 2 2 3 3 1 1 2 2 ...

> xtabs(~ comp2 + comp1, lw)

     comp1
comp2 A B C D
    A 2 2 2 2
    B 2 2 2 2
    C 2 2 2 2
```

Interaction plots

- An *interaction plot* shows the response as a function of one factor with symbols or line types determined by a second factor.
- Typically the response is on the vertical axis. I prefer the horizontal. I also reorder factor levels.

```
> dotplot(reorder(comp1, lw) ~ lw, lw,
+          groups = comp2, type = c("p", "a"))
```


A
B
C
Component 1
A
C
D
B
Long wave (LW) appearance measure