5.4: Getting an Overall Summary of a Variable

Up to this point in the chapter I've explained several different summary statistics that are commonly used when analysing data, along with specific functions that you can use in R to calculate each one. However, it's kind of annoying to have to separately calculate means, medians, standard deviations, skews etc. Wouldn't it be nice if R had some helpful functions that would do all these tedious calculations at once? Something like `summary()` or `describe()`, perhaps? Why yes, yes it would. So much so that both of these functions exist. The `summary()` function is in the `base` package, so it comes with every installation of R. The `describe()` function is part of the `psych` package, which we loaded earlier in the chapter.

5.4.1 "Summarising" a variable

The `summary()` function is an easy thing to use, but a tricky thing to understand in full, since it's a generic function (see Section 4.11. The basic idea behind the `summary()` function is that it prints out some useful information about whatever object (i.e., variable, as far as we're concerned) you specify as the `object` argument. As a consequence, the behaviour of the `summary()` function differs quite dramatically depending on the class of the object that you give it. Let's start by giving it a *numeric* object:

```
summary( object = afl.margins )
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   12.75   30.50   35.30   50.50  116.00
```

For numeric variables, we get a whole bunch of useful descriptive statistics. It gives us the minimum and maximum values (i.e., the range), the first and third quartiles (25th and 75th percentiles; i.e., the IQR), the mean and the median. In other words, it gives us a pretty good collection of descriptive statistics related to the central tendency and the spread of the data.

Okay, what about if we feed it a logical vector instead? Let's say I want to know something about how many "blowouts" there were in the 2010 AFL season. I operationalise the concept of a blowout (see Chapter 2) as a game in which the winning margin exceeds 50 points. Let's create a logical variable `blowouts` in which the i-th element is `TRUE` if that game was a blowout according to my definition,

```
blowouts <-  afl.margins > 50
blowouts
```

```
##   [1]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [12]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
##  [34]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
##  [56]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [67]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [78] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## [111] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [122]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
## [133] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
## [144]  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [155]  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
## [166] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

So that's what the `blowouts` variable looks like. Now let's ask R for a `summary()`

```
summary( object = blowouts )
```

```
##    Mode   FALSE    TRUE
## logical     132      44
```

In this context, the `summary()` function gives us a count of the number of `TRUE` values, the number of `FALSE` values, and the number of missing values (i.e., the `NA` s). Pretty reasonable behaviour.

Next, let's try to give it a factor. If you recall, I've defined the `afl.finalists` vector as a factor, so let's use that:

```
summary( object = afl.finalists )
```

```
##         Adelaide         Brisbane          Carlton      Collingwood
##               26               25               26               28
##         Essendon          Fitzroy        Fremantle          Geelong
##               32                0                6               39
##         Hawthorn        Melbourne  North Melbourne    Port Adelaide
##               27               28               28               17
##         Richmond         St Kilda           Sydney       West Coast
##                6               24               26               38
## Western Bulldogs
##               24
```

For factors, we get a frequency table, just like we got when we used the `table()` function. Interestingly, however, if we convert this to a character vector using the `as.character()` function (see Section 7.10, we don't get the same results:

```
f2 <- as.character( afl.finalists )
summary( object = f2 )
```

```
##    Length     Class      Mode
##       400 character character
```

This is one of those situations I was referring to in Section 4.7, in which it is helpful to declare your nominal scale variable as a factor rather than a character vector. Because I've defined `afl.finalists` as a factor, R *knows* that it should treat it as a nominal scale variable, and so it gives you a much more detailed (and helpful) summary than it would have if I'd left it as a character vector.

5.4.2 "Summarising" a data frame

Okay what about data frames? When you pass a data frame to the `summary()` function, it produces a slightly condensed summary of each variable inside the data frame. To give you a sense of how this can be useful, let's try this for a new data set, one that you've never seen before. The data is stored in the `clinicaltrial.Rdata` file, and we'll use it a lot in Chapter 14 (you can find a complete description of the data at the start of that chapter). Let's load it, and see what we've got:

```
load( "./data/clinicaltrial.Rdata" )
who(TRUE)
```

```
##    -- Name --     -- Class --    -- Size --
##    clin.trial     data.frame     18 x 3
##     $drug         factor         18
##     $therapy      factor         18
##     $mood.gain    numeric        18
```

There's a single data frame called `clin.trial` which contains three variables, `drug`, `therapy` and `mood.gain`. Presumably then, this data is from a clinical trial of some kind, in which people were administered different drugs; and the researchers looked to see what the drugs did to their mood. Let's see if the `summary()` function sheds a little more light on this situation:

```
summary( clin.trial )
```

```
##        drug         therapy    mood.gain
##  placebo :6   no.therapy:9   Min.   :0.1000
##  anxifree:6   CBT       :9   1st Qu.:0.4250
##  joyzepam:6                  Median :0.8500
##                              Mean   :0.8833
##                              3rd Qu.:1.3000
##                              Max.   :1.8000
```

Evidently there were three drugs: a placebo, something called "anxifree" and something called "joyzepam"; and there were 6 people administered each drug. There were 9 people treated using cognitive behavioural therapy (CBT) and 9 people who received no psychological treatment. And we can see from looking at the summary of the `mood.gain` variable that most people did show a mood gain (mean =.88), though without knowing what the scale is here it's hard to say much more than that. Still, that's not too bad. Overall, I feel that I learned something from that.

5.4.3 "Describing" a data frame

The `describe()` function (in the `psych` package) is a little different, and it's really only intended to be useful when your data are interval or ratio scale. Unlike the `summary()` function, it calculates the same descriptive statistics for any type of variable you give it. By default, these are:

- `var` . This is just an index: 1 for the first variable, 2 for the second variable, and so on.
- `n` . This is the sample size: more precisely, it's the number of non-missing values.
- `mean` . This is the sample mean (Section 5.1.1).
- `sd` . This is the (bias corrected) standard deviation (Section 5.2.5).
- `median` . The median (Section 5.1.3).
- `trimmed` . This is trimmed mean. By default it's the 10% trimmed mean (Section 5.1.6).
- `mad` . The median absolute deviation (Section 5.2.6).
- `min` . The minimum value.
- `max` . The maximum value.
- `range` . The range spanned by the data (Section 5.2.1).
- `skew` . The skewness (Section 5.3).
- `kurtosis` . The kurtosis (Section 5.3).
- `se` . The standard error of the mean (Chapter 10).

Notice that these descriptive statistics generally only make sense for data that are interval or ratio scale (usually encoded as numeric vectors). For nominal or ordinal variables (usually encoded as factors), most of these descriptive statistics are not all that useful. What the `describe()` function does is convert factors and logical variables to numeric vectors in order to do the calculations. These variables are marked with `*` and most of the time, the descriptive statistics for those variables won't make much sense. If you try to feed it a data frame that includes a character vector as a variable, it produces an error.

With those caveats in mind, let's use the `describe()` function to have a look at the `clin.trial` data frame. Here's what we get:

```
describe( x = clin.trial )
```

```
##           vars  n mean   sd median trimmed  mad min max range skew
## drug*        1 18 2.00 0.84   2.00    2.00 1.48 1.0 3.0   2.0 0.00
## therapy*     2 18 1.50 0.51   1.50    1.50 0.74 1.0 2.0   1.0 0.00
## mood.gain    3 18 0.88 0.53   0.85    0.88 0.67 0.1 1.8   1.7 0.13
##           kurtosis   se
## drug*        -1.66 0.20
## therapy*     -2.11 0.12
## mood.gain    -1.44 0.13
```

As you can see, the output for the asterisked variables is pretty meaningless, and should be ignored. However, for the `mood.gain` variable, there's a lot of useful information.
