CHAPTER 5

Factors

Often your data needs to be grouped by category: blood pressure by age range, accidents by auto manufacturer, and so forth. R has a special collection type called a *factor* to track these categorized values.

Creating Factors 5.1

It's time to take inventory of the ship's hold. We'll make a vector for you with the type of booty in each chest.

To categorize the values, simply pass the vector to the `factor` function:

```
> chests <- c('gold', 'silver', 'gems', 'gold', 'gems')
> types <- factor(chests)
```

There are a couple differences between the original vector and the new factor that are worth noting. Print the `chests` vector:

```
> print(chests)
[1] "gold"   "silver" "gems"   "gold"   "gems"
```

You see the raw list of strings, repeated values and all. Now print the `types` factor:

```
> print(types)
[1] gold   silver gems   gold   gems
Levels: gems gold silver
```

Printed at the bottom, you'll see the factor's "levels" - groups of unique values. Notice also that there are no quotes around the values. That's because they're not strings; they're actually integer references to one of the factor's levels.

Let's take a look at the underlying integers. Pass the factor to the `as.integer` function:

```
> as.integer(types)
[1] 2 3 1 2 1
```

You can get only the factor levels with the `levels` function:

```
> levels(types)
[1] "gems"   "gold"   "silver"
```

Plots With Factors 5.2

You can use a factor to separate plots into categories. Let's graph our five chests by weight and value, and show their type as well. We'll create two vectors for you; `weights` will contain the weight of each chest, and `prices` will track how much the chests are worth.

Now, try calling `plot` to graph the chests by weight and value.

```
> weights <- c(300, 200, 100, 250, 150)
> prices <- c(9000, 5000, 12000, 7500, 18000)
> plot(weights, prices)
```

We can't tell which chest is which, though. Fortunately, we can use different plot characters for each type by converting the factor to integers, and passing it to the `pch` argument of `plot`.

```
> plot(weights, prices, pch=as.integer(types))
```

"Circle", "Triangle", and "Plus Sign" still aren't great descriptions for treasure, though. Let's add a legend to show what the symbols mean.

The `legend` function takes a location to draw in, a vector with label names, and a vector with numeric plot character IDs.

```
> legend("topright", c("gems", "gold", "silver"), pch=1:3)
```

Next time the boat's taking on water, it would be wise to dump the silver and keep the gems!

If you hard-code the labels and plot characters, you'll have to update them every time you change the plot factor. Instead, it's better to derive them by using the `levels` function on your factor:

```
> legend("topright", levels(types), pch=1:length(levels(types)))
```

Chapter 5 Completed

A long inland march has brought us to the end of Chapter 5. We've stumbled across another badge!

Factors help you divide your data into groups. In this chapter, we've shown you how to create them, and how to use them to make plots more readable.

Continue