The Street Where You Live

Data Computing

Computing project

People's addresses involve streets, lanes, courts, avenues, and so on. How many such road-related words are in common use?

In answering this question, you would presumably want to look at lots of addresses and extract the road-related term. You could do this by eye, reading down a list of a few hundred or thousand addresses. But if you want to do it on a really large scale, a city or state or country, you would want some automated help, for instance, a computer program that discards the sorts of entries you have already identified to give a greater concentration of unidentified terms. In this activity, you're going to build such a program.

	address
1	1404 WOODSIDE TERRACE
2	26151 LAKESHORE BLVD 824
3	8900 DEERLAND GROVE DRIVE
4	2833 ROSENBURG ROAD
5	6201 DAYBROOK CIR
6	3413 OAK TRAIL
7	4558 B CAPITAL BLVD
8	2117 MARINER CIRCLE
9	1522 SAVANNAH PLACE DR
10	14002 GOVERNORS CT

Addresses from Wake County, North Carolina

Some resources:

1. The file `http://tiny.cc/dcf/street-addresses.csv` contains about 15000 street addresses of registered voters in Wake County, North Carolina.

```
Wake_county <- read.csv("http://tiny.cc/dcf/street-addresses.csv", stringsAsFactors = FALSE)
```

2. The file "`http://tiny.cc/dcf/CMS_ProvidersSimple.rds`" has street address of about 900,000 medicare service providers. Download the file to save it on your own system, then read it in under a convenient name.

```
download.file(url="http://tiny.cc/dcf/CMS_ProvidersSimple.rds",
              destfile = "YourNameForTheFile.rds")
Medicare <- readRDS("YourNameForTheFile.rds")
```

Street words (e.g. "ST", "LANE") are easy for a human reader to recognize. Look through a few dozen of the addresses to identify some common ones. You can put these into a regex, separated by the *or* symbol so that the match will occur when any of the patterns are detected, e.g. `"ST|RD|ROAD"`.

This will be very easy for the most common street words. The hard part is to find the rarely used street words. To help identify such addresses, filter out the ones that match your existing pattern. Then look through the ones that make it through the filter to see other words that you would like to add to your pattern.

This is an iterative process. Every time you extend your street-word regex pattern, you may encounter cases you didn't notice previously in the addresses that don't match the pattern.

To get you started, read the following R statements. Next to each line, give a short explanation of what the line contributes to the task.

For each of the regexes, explain in simple everyday language what pattern is being matched.

```
pattern <- "ST|RD|ROAD"
LeftOvers <-
  Wake_county %>%
  filter( ! grepl(pattern, address),
          ! grepl(" BOX ", address)
  )
```

	address
1	2117 MARINER CIRCLE
2	101 EPPING WAY
3	04-I ROBIN CIRCLE
4	NCSU BOX 15637
5	4719 BROWN TRAIL
6	130 THE WINERY

LeftOvers

When you have this working on the small sample, use a larger sample and, eventually, the whole data set. It's practically impossible to find a method that will work perfectly on all new data, but do the best you can.

For carrying out our survey of address street types, it's not sufficient to know that an address matches any one of a number of possibilities. You need to know *which* particular possibility it matches. To set this up, first modify your pattern mark the part you want to extract. Surround that part by parentheses. In the very simple pattern, this would look like

```
pattern <- "(ST|RD|ROAD)" # Note the parentheses!
```

This pattern will match exactly the same cases as the original: the parentheses are not part of the pattern. Instead, the parentheses are used by data verbs such as `tidyr::extract()` which will pull the matching component into a new variable. For example:

```
Wake_county %>%
  tidyr::extract(address, into = "street_word",
                 regex = pattern,
                 remove = FALSE) %>%
  head(5)
```

```
##               address street_word
## 1 2117 MARINER CIRCLE        <NA>
## 2      101 EPPING WAY        <NA>
## 3        PO BOX 58592        <NA>
## 4  5102 ECHO RIDGE RD          RD
## 5        PO BOX 37218        <NA>
```

You can see at a glance that the simple pattern is not capturing the circles and ways.

Your turn: In your report, implement your method and explain how it works, line by line. Present your result: how many addresses there are of each kind of road word?