Chapter 1

Data Reshaping

Most of the time, our data is in the form of a data frame and we are interested in exploring the relationships. However most procedures in R expect the data to show up in a 'long' format where each row is an observation and each column is a covariate. In practice, the data is often not stored like that and the data comes to us with repeated observations included on a single row. This is often done as a memory saving technique or because there is some structure in the data that makes the 'wide' format attractive. As a result, we need a way to convert data from 'wide' to 'long' and vice-versa.

1.1 Package `tidyr`

There is a common issue with obtaining data with many columns that you wish were organized as rows. For example, I might have data in a grade book that has several homework scores and I'd like to produce a nice graph that has assignment number on the x-axis and score on the y-axis. Unfortunately this is incredibly hard to do when the data is arranged in the following way:

```
grade.book <- rbind(
  data.frame(name='Alison',  HW.1=8, HW.2=5, HW.3=8),
  data.frame(name='Brandon', HW.1=5, HW.2=3, HW.3=6),
  data.frame(name='Charles', HW.1=9, HW.2=7, HW.3=9))
grade.book
```

```
##      name HW.1 HW.2 HW.3
## 1  Alison    8    5    8
## 2 Brandon    5    3    6
## 3 Charles    9    7    9
```

What we want to do is turn this data frame from a *wide* data frame into a *long* data frame. In MS Excel this is called *pivoting*. Essentially I'd like to create a data frame with three columns: `name`, `assignment`, and `score`. That is to say that each homework datum really has three pieces of information: who it came from, which homework it was, and what the score was. It doesn't conceptually matter if I store it as 5 columns or 5 rows so long as there is a way to identify how a student scored on a particular homework. So we want to reshape the HW1 to HW5 columns into two columns (assignment and score).

This package was built by the sample people that created `dplyr` and `ggplot2` and there is a nice introduction at: http://blog.rstudio.org/2014/07/22/introducing-tidyr/

`gather()` and `spread()`

As with the `dplyr` package, there are two main verbs to remember:

1. Gather - Gather multiple columns that are related into two columns that contain the original column name and the value. For example for columns `HW1`, ..., `HW5` we would gather them into two column

`HomeworkNumber` and `Score`. In this case, we refer to `HomeworkNumber` as the key column and `Score` as the value column. So for any key:value pair you know everything you need.

2. Spread - This is the opposite of gather. This takes a key column (or columns) and a results column and forms a new column for each level of the key column(s).

```
library(tidyr)
# first we gather the score columns into columns we'll name Assesment and Score
tidy.scores <- grade.book %>% gather( key=Assessement, value=Score, HW.1:HW.3 )
tidy.scores

##      name Assessement Score
## 1  Alison        HW.1     8
## 2 Brandon        HW.1     5
## 3 Charles        HW.1     9
## 4  Alison        HW.2     5
## 5 Brandon        HW.2     3
## 6 Charles        HW.2     7
## 7  Alison        HW.3     8
## 8 Brandon        HW.3     6
## 9 Charles        HW.3     9
```

To spread the key:value pairs out into a matrix, we use the `spread()` command.

```
# first we gather the score columns into columns we'll name Assesment and Score
tidy.scores %>% spread( key=Assessement, value=Score )

##      name HW.1 HW.2 HW.3
## 1  Alison    8    5    8
## 2 Brandon    5    3    6
## 3 Charles    9    7    9
```

One way to keep straight which column should be the `key` column and which is the `value` is that the `key` is the category, while `value` is the numerical value or response.

1.2 Exercises

1. Suppose we are given information about the maximum daily temperature from a weather station in Flagstaff, AZ. The file is available at the GitHub site that this book is hosted on.

```
FlagTemp <-  read.csv(
  'https://github.com/dereksonderegger/STA_570L_Book/raw/master/data/FlagMaxTemp.csv',
   header=TRUE, sep=',')
```

This file is in a wide format, where each row represents a month and the columns X1, X2, ..., X31 represent the day of the month the observation was made.

(a) Convert data set to the long format where the data has only four columns: `Year, Month, Day, Tmax`.

(b) Calculate the average monthly maximum temperature for each Month in the dataset (So there will be 360 mean maximum temperatures).

(c) Convert the average month maximums back to a wide data format where each line represents a year and there are 12 columns of temperature data (one for each month) along with a column for the year.