Module 9
Data Cleaning

Andrew Jaffe
Instructor
Data

- We will be using multiple data sets in this lecture:
  - Salary, Monument, Circulator, and Restaurant from OpenBaltimore: [https://data.baltimorecity.gov/browse?limitTo=datasets](https://data.baltimorecity.gov/browse?limitTo=datasets)
  - Gap Minder - very interesting way of viewing longitudinal data
    - Data is here - [http://www.gapminder.org/data/](http://www.gapminder.org/data/)
  - [http://spreadsheets.google.com/pub?key=rMsQHawTObBb6_U2ESjKXYw&output=xls](http://spreadsheets.google.com/pub?key=rMsQHawTObBb6_U2ESjKXYw&output=xls)
Data Cleaning

In general, data cleaning is a process of investigating your data for inaccuracies, or recoding it in a way that makes it more manageable.

MOST IMPORTANT RULE - LOOK AT YOUR DATA!

Again - table, summarize, is.na, any, all are useful.
Data Cleaning

```r
> table(c(0, 1, 2, 3, NA, 3, 3, 2, 2, 3), useNA = "ifany")

   0  1  2  3 <NA>
 1  1  3  4  1

> table(c(0, 1, 2, 3, 2, 3, 2, 2, 3), useNA = "always")

   0  1  2  3 <NA>
 1  1  4  4  0

> tab <- table(c(0, 1, 2, 3, 2, 3, 2, 2, 3), c(0, 1, 2, 3, 2, 3, 4, 4, 3),
           +     useNA = "always")
> margin.table(tab, 2)

   0  1  2  3  4 <NA>
 1  1  2  4  2  0
```
\begin{verbatim}
> prop.table(tab)

     0    1    2    3    4  <NA>
0  0.1  0.0  0.0  0.0  0.0  0.0
1  0.0  0.1  0.0  0.0  0.0  0.0
2  0.0  0.0  0.2  0.0  0.2  0.0
3  0.0  0.0  0.4  0.0  0.0  <NA>
      0.0  0.0  0.0  0.0  0.0  0.0

> prop.table(tab, 1)

     0    1    2    3    4  <NA>
0  1.0  0.0  0.0  0.0  0.0  0.0
1  0.0  1.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.5  0.0  0.5  0.0
3  0.0  0.0  0.0  1.0  0.0  0.0
      <NA>
\end{verbatim}
Download Salary FY2012 Data

https://data.baltimorecity.gov/Financial/Baltimore-City-Employee-Salaries-FY2012/7ymi-bvp3

Download as a CSV and then read it into R as the variable Sal
Download Salary FY2012 Data

https://data.baltimorecity.gov/Financial/Baltimore-City-Employee-Salaries-FY2012/7ymi-bvp3

Download as a CSV and then read it into R as the variable `Sal`

```r
> Sal = read.csv("data/Baltimore_City_Employee_Salaries_FY2012.csv", as.is = TRUE)
> colnames(Sal)[1] = "Name"  # make uppercase
```
Data Cleaning

- `any()` - checks if there are any TRUES
- `all()` - checks if ALL are true

```
> Sal[1:4, ]

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>JobTitle</td>
<td>AgencyID</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Aaron,Patricia G</td>
<td>A03031</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Aaron,Patricia L</td>
<td>A29005</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Abaineh,Yohannes T</td>
<td>A65026</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Abdal-Rahim,Naim A</td>
<td>A64215</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency</td>
<td>HireDate</td>
<td>AnnualSalary</td>
<td>GrossPay</td>
</tr>
<tr>
<td>1</td>
<td>10/24/1979</td>
<td>$51862.00</td>
<td>$52247.39</td>
</tr>
<tr>
<td>2</td>
<td>09/25/2006</td>
<td>$64000.00</td>
<td>$59026.81</td>
</tr>
<tr>
<td>3</td>
<td>07/23/2009</td>
<td>$57900.00</td>
<td>$57129.79</td>
</tr>
<tr>
<td>4</td>
<td>03/30/2011</td>
<td>$34146.00</td>
<td>$35537.88</td>
</tr>
</tbody>
</table>

> any(is.na(Sal$Name))

[1] FALSE
Example of Cleaning:

For example, let's say gender was coded as Male, M, m, Female, F, f. Using Excel to find all of these would be a matter of filtering and changing all by hand or using if statements.

In R, you can simply do something like:

```r
data$gender[data$gender %in% c("Male", "M", "m")] <- "Male"
```

Sometimes though, it's not so simple. That's where functions that find patterns come in very useful.

```r
> table(gender)

    F FeMale FEMALE    Fm  M  Ma mAle  Male  MaLe MALE
   75  82  74  89  89  79  87  89  88  95
 Man   Woman
   73   80
```
Find/Replace and Regular Expressions

- R can do much more than find exact matches for a whole string
- Like Perl and other languages, it can use regular expressions.
- What are regular expressions?
- Ways to search for specific strings
- Can be very complicated or simple
- Highly Useful
'Find' functions

grep: grep, grepl, regexpr and gregexpr search for matches to argument pattern within each element of a character vector: they differ in the format of and amount of detail in the results.

grep(pattern, x, fixed=FALSE), where:

• pattern = character string containing a regular expression to be matched in the given character vector.

• x = a character vector where matches are sought, or an object which can be coerced by as.character to a character vector.

• If fixed=TRUE, it will do exact matching for the phrase anywhere in the vector (regular find)

> grep("Rawlings", Sal$Name)  # These are the indices/elements where the pattern match occurs

[1] 11755 11756 11757 11758

grep() returns something similar to which() on a logical statement
```r
> grep("Rawlings", Sal$Name)

[1] 11755 11756 11757 11758
```

```r
> grep("Rawlings", Sal$Name, value = TRUE)

[1] "Rawlings Jr,Anthony H" "Rawlings,Kellye A" "Rawlings,Paula M"
[4] "Rawlings,Stephanie C"
```

```r
> Sal[grep("Rawlings", Sal$Name),]
```

<table>
<thead>
<tr>
<th>Name</th>
<th>JobTitle</th>
<th>AgencyID</th>
</tr>
</thead>
<tbody>
<tr>
<td>11755 Rawlings Jr,Anthony H</td>
<td>AIDE BLUE CHIP</td>
<td>W02712</td>
</tr>
<tr>
<td>11756 Rawlings,Kellye A</td>
<td>EMERGENCY DISPATCHER</td>
<td>A99372</td>
</tr>
<tr>
<td>11757 Rawlings,Paula M</td>
<td>COMMUNITY AIDE</td>
<td>A04015</td>
</tr>
<tr>
<td>11758 Rawlings,Stephanie C</td>
<td>MAYOR</td>
<td>A01001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Agency</th>
<th>HireDate</th>
<th>AnnualSalary</th>
<th>GrossPay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Youth Summer</td>
<td>06/07/2011</td>
<td>$1</td>
<td>$1</td>
</tr>
<tr>
<td>Police Department</td>
<td>01/06/2003</td>
<td>$4</td>
<td>$5</td>
</tr>
<tr>
<td>R&amp;P-Recreation</td>
<td>12/10/2007</td>
<td>$1</td>
<td>$9</td>
</tr>
<tr>
<td>Mayors Office</td>
<td>12/07/1995</td>
<td>$1</td>
<td>$1</td>
</tr>
</tbody>
</table>
Grep Options

```r
> head(grep("Tajhgh", Sal$Name, value = TRUE))

[1] "Reynold,Tajhgh J"

> grep("Jaffe", Sal$Name)

integer(0)

> length(grep("Jaffe", Sal$Name))

[1] 0
```
A bit on Regular Expressions

- They can use to match a large number of strings in one statement
- . matches any single character
- * means repeat as many (even if 0) more times the last character
- ? makes the last thing optional
Using Regular Expressions

- Look for any name that starts with:
  - Payne at the beginning,
  - Leonard and then an S
  - Spence then a capital C

```r
> grep("Payne.*", x = Sal$Name, value = TRUE)
```

1. "Payne,Brittany N"  "Payne-Cooke,Shelley F"
2. "Payne,Denise I"  "Payne El,Jackie"
3. "Payne,Gary W"  "Payne,James D"
4. "Payne,James R"  "Payne,Jasman T"
5. "Payne Johnson,Nickole A"  "Payne,Jordan A"
7. "Payne,Mary A"  "Payne,Micah W"
8. "Payne,Michael N"  "Payne,Walter"
9. "Ray Payne,Marion J"  "Payne,Walter"
> grep("Leonard.?S", x = Sal$Name, value = TRUE)


> grep("Spence.*C.*", x = Sal$Name, value = TRUE)

[4] "Spencer, Michael C"
Let's say we wanted to sort the data set by Annual Salary:

```r
> class(Sal$AnnualSalary)
[1] "character"
```

```r
> sort(c("1", "2", "10"))  # not sort correctly (order simply ranks the data)
[1] "1" "10" "2"
```

```r
> order(c("1", "2", "10"))
[1] 1 3 2
```
Replace

So we must change the annual pay into a numeric:

```r
> head(as.numeric(Sal$AnnualSalary), 4)
```

```
[1] NA NA NA NA
```

R didn't like the $ so it thought turned them all to NA.

`sub()` and `gsub()` can do the replacing part.
Replacing and subbing

Now we can replace the $ with nothing (used fixed=TRUE because $ means something in regular expressions):

```r
> Sal$AnnualSalary <- as.numeric(gsub(pattern = "$", replacement = "", Sal$AnnualSalary, +       fixed = TRUE))
> Sal <- Sal[order(Sal$AnnualSalary, decreasing = TRUE), ]  # use negative to sort descending
> Sal[1:5, c("Name", "AnnualSalary", "JobTitle")]
```

<table>
<thead>
<tr>
<th>Name</th>
<th>AnnualSalary</th>
<th>JobTitle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernstein,Gregg L</td>
<td>238772</td>
<td>STATE'S ATTORNEY</td>
</tr>
<tr>
<td>Bealefeld III,Fredrick H</td>
<td>197700</td>
<td>EXECUTIVE LEVEL III</td>
</tr>
<tr>
<td>Black,Harry E</td>
<td>180000</td>
<td>EXECUTIVE LEVEL III</td>
</tr>
<tr>
<td>Sanchez,Alexander M</td>
<td>175000</td>
<td>EXECUTIVE LEVEL III</td>
</tr>
<tr>
<td>Barbot,Oxiris</td>
<td>170000</td>
<td>EXECUTIVE LEVEL III</td>
</tr>
</tbody>
</table>
Useful String Functions

Useful String functions

- `toupper()`, `tolower()` - uppercase or lowercase your data:
- `str_trim()` (in the stringr package) - will trim whitespace
- `nchar` - get the number of characters in a string
- `substr(x, start, stop)` - substrings from position start to position stop
- `strsplit(x, split)` - splits strings up - returns list!
- `paste()` - paste strings together - look at ?paste
Paste can be very useful for joining vectors together:

```r
> paste("Visit", 1:5, sep = "_")

[1] "Visit_1" "Visit_2" "Visit_3" "Visit_4" "Visit_5"

> paste("Visit", 1:5, sep = "_", collapse = " ")

[1] "Visit_1 Visit_2 Visit_3 Visit_4 Visit_5"

> paste("To", "is going be the ", "we go to the store!", sep = "day ")

[1] "Today is going be the day we go to the store!"

> # and paste0 can be even simpler see ?paste0
> paste0("Visit", 1:5)

[1] "Visit1" "Visit2" "Visit3" "Visit4" "Visit5"
```
> paste(1:5, letters[1:5], sep = "_")

[1] "1_a" "2_b" "3_c" "4_d" "5_e"

> paste(6:10, 11:15, 2000:2005, sep = "/")


> paste(paste("x", 1:5, sep = ""), collapse = "+")

[1] "x1+x2+x3+x4+x5"
Strsplit

```r
> x <- c("I really", "like writing", "R code")
> y <- strsplit(x, split = " ")
> y[[2]]

[1] "like"  "writing"

> sapply(y, function(x) x[1])  # on the fly

[1] "I"  "like" "R"

> sapply(y, function(x) x[2])  # on the fly

[1] "really"  "writing" "code"
```
Data Merging/Append

- Merging - joining data sets together - usually on key variables, usually "id"
- `merge()` is the most common way to do this with data sets
- `rbind/cbind` - row/column bind, respectively
  - `rbind` is the equivalent of "appending" in Stata or "setting" in SAS
  - `cbind` allows you to add columns in addition to the previous ways
- `reshape2` package also has a lot of information about different ways to reshape data (wide to long, etc) - but has a different (and sometimes more intuitive syntax)
- `t()` is a function that will transpose the data
Merging

```r
> base <- data.frame(id = 1:10, Age = seq(55, 60, length = 10))
> base[1:2, ]

<table>
<thead>
<tr>
<th>id</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55.00</td>
</tr>
<tr>
<td>2</td>
<td>55.56</td>
</tr>
</tbody>
</table>
```

```r
> visits <- data.frame(id = rep(1:8, 3), visit = rep(1:3, 8), Outcome = seq(10, + 50, length = 24))
> visits[1:2, ]

<table>
<thead>
<tr>
<th>id</th>
<th>visit</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10.00</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>11.74</td>
</tr>
</tbody>
</table>
```
> merged.data <- merge(base, visits, by = "id")
> merged.data[1:5,]

<table>
<thead>
<tr>
<th></th>
<th>id</th>
<th>Age</th>
<th>visit</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>55.00</td>
<td>1</td>
<td>10.00</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>55.00</td>
<td>3</td>
<td>23.91</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>55.00</td>
<td>2</td>
<td>37.83</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>55.56</td>
<td>2</td>
<td>11.74</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>55.56</td>
<td>1</td>
<td>25.65</td>
</tr>
</tbody>
</table>

> dim(merged.data)

[1] 24 4
> all.data <- merge(base, visits, by = "id", all = TRUE)
> tail(all.data)

<table>
<thead>
<tr>
<th>id</th>
<th>Age</th>
<th>visit</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>7</td>
<td>58.33</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>8</td>
<td>58.89</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>8</td>
<td>58.89</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>58.89</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>9</td>
<td>59.44</td>
<td>NA</td>
</tr>
<tr>
<td>26</td>
<td>10</td>
<td>60.00</td>
<td>NA</td>
</tr>
</tbody>
</table>

> dim(all.data)

[1]  26  4
Aside: Dates

You can convert date-like strings in the Date class (http://www.statmethods.net/input/dates.html for more info)

```r
> circ = read.csv("data/Charm_City_Circulator_Ridership.csv", as.is = TRUE)
> head(sort(circ$date))

[1] "01/01/2011" "01/01/2012" "01/02/2011" "01/02/2012" "01/03/2011"
[6] "01/03/2012"

> circ$date <- as.Date(circ$date, "%m/%d/%Y")  # creating a date for sorting
> head(circ$date)

[1] "2010-01-11" "2010-01-12" "2010-01-13" "2010-01-14" "2010-01-15"
[6] "2010-01-16"

> head(sort(circ$date))

[1] "2010-01-11" "2010-01-12" "2010-01-13" "2010-01-14" "2010-01-15"
[6] "2010-01-16"
```
Data Reshaping

Disclaimer: the `reshape` command in R is not remarkably intuitive.

- Wide - multiple measurements are variables / columns so that the data gets wider with more measurements
- Long - multiple measurements are rows so data gets longer with more measurements
- One example would be many ids with multiple visits
Example of Long/Wide

```r
> head(wide)

<table>
<thead>
<tr>
<th>id</th>
<th>visit1</th>
<th>visit2</th>
<th>visit3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Good</td>
<td>Good</td>
<td>Bad</td>
</tr>
</tbody>
</table>

> head(long)

<table>
<thead>
<tr>
<th>id</th>
<th>visit</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Good</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Good</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Bad</td>
</tr>
</tbody>
</table>
```
Data Reshaping

- Good resource: [http://www.ats.ucla.edu/stat/r/faq/reshape.htm](http://www.ats.ucla.edu/stat/r/faq/reshape.htm)

```r
> head(Indometh)  # this is long
```

<table>
<thead>
<tr>
<th>Subject</th>
<th>time</th>
<th>conc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>1.50</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>0.50</td>
<td>0.94</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>0.78</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>1.00</td>
<td>0.48</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1.25</td>
</tr>
<tr>
<td>5</td>
<td>1.25</td>
<td>0.37</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2.00</td>
</tr>
<tr>
<td>6</td>
<td>2.00</td>
<td>0.19</td>
</tr>
</tbody>
</table>
Data Reshaping

```r
> wide <- reshape(Indometh, v.names = "conc", idvar = "Subject", timevar = "time",
                  +    direction = "wide")
> head(wide)

    Subject conc.0.25 conc.0.5 conc.0.75 conc.1 conc.1.25 conc.2 conc.3
 1      1      1.50    0.94     0.78   0.48    0.37   0.19   0.12
12     2      2.03    1.63     0.71   0.70    0.64   0.36   0.32
23     3      2.72    1.49     1.16   0.80    0.80   0.39   0.22
34     4      1.85    1.39     1.02   0.89    0.59   0.40   0.16
45     5      2.05    1.04     0.81   0.39    0.30   0.23   0.13
56     6      2.31    1.44     1.03   0.84    0.64   0.42   0.24
```

```r
    conc.4 conc.5 conc.6 conc.8
 1   0.11   0.08   0.07   0.05
12  0.20   0.25   0.12   0.08
23  0.12   0.11   0.08   0.08
34  0.11   0.10   0.07   0.07
45  0.11   0.08   0.10   0.06
56  0.17   0.13   0.10   0.09
```
## Data Reshaping

```r
> dim(Indometh)

[1] 66 3
```

```r
> wide
```

<table>
<thead>
<tr>
<th>Subject</th>
<th>conc.0.25</th>
<th>conc.0.5</th>
<th>conc.0.75</th>
<th>conc.1</th>
<th>conc.1.25</th>
<th>conc.2</th>
<th>conc.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.50</td>
<td>0.94</td>
<td>0.78</td>
<td>0.48</td>
<td>0.37</td>
<td>0.19</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>2.03</td>
<td>1.63</td>
<td>0.71</td>
<td>0.70</td>
<td>0.64</td>
<td>0.36</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>2.72</td>
<td>1.49</td>
<td>1.16</td>
<td>0.80</td>
<td>0.80</td>
<td>0.39</td>
</tr>
<tr>
<td>34</td>
<td>4</td>
<td>1.85</td>
<td>1.39</td>
<td>1.02</td>
<td>0.89</td>
<td>0.59</td>
<td>0.40</td>
</tr>
<tr>
<td>45</td>
<td>5</td>
<td>2.05</td>
<td>1.04</td>
<td>0.81</td>
<td>0.39</td>
<td>0.30</td>
<td>0.23</td>
</tr>
<tr>
<td>56</td>
<td>6</td>
<td>2.31</td>
<td>1.44</td>
<td>1.03</td>
<td>0.84</td>
<td>0.64</td>
<td>0.42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>conc.4</th>
<th>conc.5</th>
<th>conc.6</th>
<th>conc.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.11</td>
<td>0.08</td>
<td>0.07</td>
</tr>
<tr>
<td>12</td>
<td>0.20</td>
<td>0.25</td>
<td>0.12</td>
</tr>
<tr>
<td>23</td>
<td>0.12</td>
<td>0.11</td>
<td>0.08</td>
</tr>
<tr>
<td>34</td>
<td>0.11</td>
<td>0.10</td>
<td>0.07</td>
</tr>
<tr>
<td>45</td>
<td>0.11</td>
<td>0.08</td>
<td>0.10</td>
</tr>
<tr>
<td>56</td>
<td>0.17</td>
<td>0.13</td>
<td>0.10</td>
</tr>
</tbody>
</table>
Data Reshaping

- If you've reshaped a data set - to get it back, just reshape it again

```r
> reshape(wide, direction = "long")[1:10, ]
```

<table>
<thead>
<tr>
<th>Subject</th>
<th>time</th>
<th>conc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.25</td>
<td>1 0.25</td>
<td>1.50</td>
</tr>
<tr>
<td>2.0.25</td>
<td>2 0.25</td>
<td>2.03</td>
</tr>
<tr>
<td>3.0.25</td>
<td>3 0.25</td>
<td>2.72</td>
</tr>
<tr>
<td>4.0.25</td>
<td>4 0.25</td>
<td>1.85</td>
</tr>
<tr>
<td>5.0.25</td>
<td>5 0.25</td>
<td>2.05</td>
</tr>
<tr>
<td>6.0.25</td>
<td>6 0.25</td>
<td>2.31</td>
</tr>
<tr>
<td>1.0.5</td>
<td>1 0.50</td>
<td>0.94</td>
</tr>
<tr>
<td>2.0.5</td>
<td>2 0.50</td>
<td>1.63</td>
</tr>
<tr>
<td>3.0.5</td>
<td>3 0.50</td>
<td>1.49</td>
</tr>
<tr>
<td>4.0.5</td>
<td>4 0.50</td>
<td>1.39</td>
</tr>
</tbody>
</table>

Note the row name change
### Data Reshaping - A Better Example

```r
> TB <- read.xlsx(file = '~/Dropbox/WinterRClass/Datasets/indicator_estimatedincidencealltbper100000.xlsx',
+ sheetName = "Data")
> head(TB, 1)
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X2004</td>
<td>X2005</td>
<td>X2006</td>
<td>X2007</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```r
> TB$NA. <- NULL
> head(TB, 1)
```

|-------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
Data Reshaping - A Better Example

```r
> colnames(TB) <- c("Country", paste("Year", 1990:2007, sep = "."))
> head(TB, 1)

<table>
<thead>
<tr>
<th>Country</th>
<th>Year.1990</th>
<th>Year.1991</th>
<th>Year.1992</th>
<th>Year.1993</th>
<th>Year.1994</th>
<th>Year.1995</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afganistan</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>Year.1996</td>
<td>Year.1997</td>
<td>Year.1998</td>
<td>Year.1999</td>
<td>Year.2000</td>
<td>Year.2001</td>
</tr>
<tr>
<td></td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>Year.2002</td>
<td>Year.2003</td>
<td>Year.2004</td>
<td>Year.2005</td>
<td>Year.2006</td>
<td>Year.2007</td>
</tr>
<tr>
<td></td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
</tr>
</tbody>
</table>
```
Data Reshaping - More is better!

```r
> TB.long <- reshape(TB, idvar = "Country", v.names = "Cases", times = 1990:2007,
+     direction = "long", timevar = "Year", varying = paste("Year", 1990:2007,
+     sep = "."))
> head(TB.long, 4)
```

<table>
<thead>
<tr>
<th>Country</th>
<th>Year</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghanistan.1990</td>
<td>1990</td>
<td>168</td>
</tr>
<tr>
<td>Albania.1990</td>
<td>1990</td>
<td>25</td>
</tr>
<tr>
<td>Algeria.1990</td>
<td>1990</td>
<td>38</td>
</tr>
<tr>
<td>American Samoa.1990</td>
<td>1990</td>
<td>21</td>
</tr>
</tbody>
</table>

```r
> rownames(TB.long) <- NULL
> head(TB.long, 4)
```

<table>
<thead>
<tr>
<th>Country</th>
<th>Year</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghanistan</td>
<td>1990</td>
<td>168</td>
</tr>
<tr>
<td>Albania</td>
<td>1990</td>
<td>25</td>
</tr>
<tr>
<td>Algeria</td>
<td>1990</td>
<td>38</td>
</tr>
<tr>
<td>American Samoa</td>
<td>1990</td>
<td>21</td>
</tr>
</tbody>
</table>