Merging data, Dates

Richard Upton
6 August 2015

Merging data

Combining data from different sources and in different formats into an R dataframe

We might merge data when:

- Setting up data for a graphical analysis
- Setting up data for a NONMEM analysis
- Combining the results of NONMEM analyses(simulations)

R data import

R can read data in many formats:

- see the "R Data Import/Export" manual
- Spreadsheet like (*.csv)
- Native Excel (*.xls, *.xlsx)
- Minitab, S-PLUS, SAS, SPSS, Stata, Systat
- Relational databases (mySQL etc.)
- Image files
- Text files (*.txt)
- Web pages (*.html, *.xml)

Data preparation

In pharmacometrics, data might arrive as:

- Excel spreadsheets (*.csv, *.xlsx)
- Proprietary statistical software (SAS transport files)
- Word documents (tables)
- PDF documents (tables)
- Handwritten experimental records

Comma separated variable file (*.csv)

Get the data into a *.csv file for reading into R

This is not trivial and is error prone!

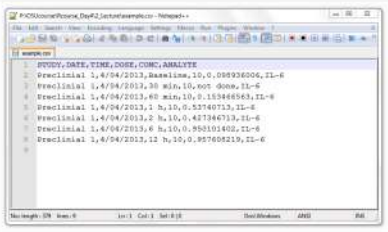
*.csv is a text document (can be read with Notepad etc.)

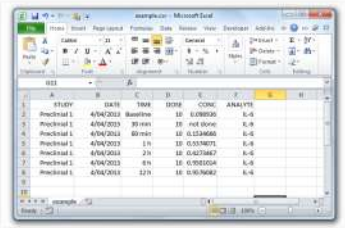
Simple, readable non-proprietary format

Data structured as columns

Columns are separated by comma's
(not in all countries!)

Data often has a header row with column names





Reading csv data

- the "read.csv" function

```
> exampledata <- read.csv("example.csv", na.strings=c("not done"),
stringsAsFactors=F)
> exampledata
```

```
      STUDY      DATE     TIME DOSE    CONC ANALYTE
1 Preclinial 1 4/04/2013 Baseline   10 0.09894    IL-6
2 Preclinial 1 4/04/2013   30 min   10      NA    IL-6
3 Preclinial 1 4/04/2013   60 min   10 0.15347    IL-6
4 Preclinial 1 4/04/2013      1 h   10 0.53741    IL-6
5 Preclinial 1 4/04/2013      2 h   10 0.42735    IL-6
6 Preclinial 1 4/04/2013      6 h   10 0.95010    IL-6
7 Preclinial 1 4/04/2013     12 h   10 0.95761    IL-6
```

- see also the "skip" and "header" arguments ?read.csv

The "rbind" function

- Joins two dataframes by row
- They need the same columns with the same names

```
> data1
```

```
  ID     TIME    CONC
1  1 Baseline 0.09894
2  1   30 min 0.62152
3  1   60 min 0.15347
```

```
> data2
```

```
  ID     TIME    CONC
1  2 Baseline 0.09894
2  2   30 min 0.07504
3  2   60 min 0.15347
```

The "rbind" function

```
> alldata <- rbind(data1,data2)
> alldata
```

```
  ID     TIME    CONC
1  1 Baseline 0.09894
2  1   30 min 0.62152
3  1   60 min 0.15347
4  2 Baseline 0.09894
5  2   30 min 0.07504
6  2   60 min 0.15347
```

The "cbind" function

- Joins two dataframes by column
- They need the same number of rows

```
> data1
```

```
  ID     TIME    CONC
1  1 Baseline 0.09894
2  1   30 min 0.62152
3  1   60 min 0.15347
4  2 Baseline 0.09894
5  2   30 min 0.07504
6  2   60 min 0.15347
```

```
> data2
```

```
     SEX
1   male
2   male
3   male
4 female
5 female
6 female
```

The "cbind" function

```
> alldata <- cbind(data1,data2)
> alldata
```

```
  ID     TIME    CONC    SEX
1  1 Baseline 0.09894   male
2  1   30 min 0.62152   male
3  1   60 min 0.15347   male
4  2 Baseline 0.09894 female
5  2   30 min 0.07504 female
6  2   60 min 0.15347 female
```

The powerful "merge" function

- Joins two dataframes by common columns

```
> data1
```

```
  ID TIME    CONC
1  1    0 0.09894
2  1   30 0.62152
3  1   60 0.15347
```

```
> data2
```

```
  ID TIME    MAP
1  1    0 0.6351
2  1   10 0.1444
3  1   90 0.1130
```

The "merge" function

```
> alldata <- merge(data1,data2, all=T)
> alldata
```

```
  ID TIME    CONC    MAP
1  1    0 0.09894 0.6351
2  1   10      NA 0.1444
3  1   30 0.62152     NA
4  1   60 0.15347     NA
5  1   90      NA 0.1130
```

Reshaping data

The classic problem is converting from wide to long format

I wish this was easy, but it isn't

reshape is a function in the base package

reshape is also a package by Hadley Wickham which provides the "melt" and "cast" functions

most jobs can be done using melt & cast

see also the "tidyr" package by Hadley Wickham

Melt & Cast

The presumption is that all data can be "melted" down into one dependent variable column and a number of identifier columns.

Sounds implausible, but it seems to be the case.

Data can be "cast" into new formats by splitting the identifier into rows or columns.

Melt & Cast

```
> datawide
```

```
  TIME CONC1 CONC2
1    0  0.55  0.98
2   30  0.11  0.49
3   60  0.22  0.02
```

Melt

```
> library(reshape)
> datalong <- melt(datawide, id=c("TIME"), na.rm=T)
> datalong
```

```
  TIME variable value
1    0    CONC1  0.55
2   30    CONC1  0.11
3   60    CONC1  0.22
4    0    CONC2  0.98
5   30    CONC2  0.49
6   60    CONC2  0.02
```

Cast

```
> library(reshape)
> datawide2 <- cast(datalong, TIME~variable)
> #formula is column ~ row
> datawide2
```

```
  TIME CONC1 CONC2
1    0  0.55  0.98
2   30  0.11  0.49
3   60  0.22  0.02
```

Reshaping data - Summary

For big data sets, reshaping can get complex

The toolkit of rbind, cbind, melt and cast can do most jobs

melting & casting takes practice

Break the job down into smaller parts

An R script provides an audit trail for the reshape

If source data is updated, the R script is easily re-run

Dates

Pharmacometrics is all about dates and times!

Dose and observations are recorded as a date and time

We need to be fluent in converting dates and times to:

- "Time After First Dose" (TAFD)
- "Time After Dose" (TAD)
- "Nominal Sample Time" (TNOM)

TNOM is needed for pooling data across subjects

Most data sets need all three times!

The problem with dates and times

These all record the same event:

- 14 February 1994 10:00
- 14 Feb 1994 10:00
- 14 Feb 1994 10 am
- 14 Feb 1994 10:00:00
- 14/2/1994 10:00
- 2/14/1994 10:00
- 2/14/94 10:00
- 2-14-94 10:00

Dates and Excel

Excel will corrupt dates and times!

It think's it's being helpful by automatically converting to local format

You can't turn this off when opening a spreadsheet

If working with non-US date and time formats, avoid Excel

POSIX Dates and Times

R represent dates and times in POSIX (UNIX) format.

POSIXlt: A standardized structured list: seconds, minutes, hours, day of the month, months after the first of the year, years since 1900, day of the week starting on Sunday, day of the year, Daylight Savings Time flag.

POSIXct: Number of seconds elapsed since midnight Coordinated Universal Time (UTC), 1 January 1970.

There are functions that convert to and from these formats

There are functions for time calculations for these formats

POSIX Dates and Times

Typically:

- textstring to POSIX
- time calculation in POSIX
- POSIX to textstring or number

Use inbuilt time functions if convenient

When complete numerical control is needed, use POSIXct

Both formats print as "2004-04-16 16:30:00 CST" despite different internal representation

The strptime function - text to POSIX

```
> dosedatetime <- "4/15/2004 14:00"
> dosedatetime <- strptime( dosedatetime, format = "%m/%d/%Y %H:%M")
> dosedatetime
```

```
[1] "2004-04-15 14:00:00 CST"
```

```
> dosedatetime <- "2004-4-15 2:00 PM"
> dosedatetime <- strptime( dosedatetime, format = "%Y-%m-%d %I:%M %p")
> dosedatetime
```

```
[1] "2004-04-15 14:00:00 CST"
```

see ?strptime for textstring codes

The difftime function

```
> dosedatetime <- "4/15/2004 14:00"
> dosedatetime <- strptime( dosedatetime, format = "%m/%d/%Y %H:%M")
>
> PKdatetime <- "4/16/2004 09:16"
> PKdatetime <- strptime( PKdatetime, format = "%m/%d/%Y %H:%M")
>
> TAFD <- difftime(PKdatetime,dosedatetime, units="mins")
> TAFD
```

```
Time difference of 1156 mins
```

```
> TAFD <- difftime(PKdatetime,dosedatetime, units="hours")
> TAFD
```

```
Time difference of 19.27 hours
```

The difftime function

```
> TAFD
```

```
Time difference of 19.27 hours
```

```
> class(TAFD)
```

```
[1] "difftime"
```

```
> TAFD <- as.numeric(TAFD)
> TAFD
```

```
[1] 19.27
```

Adding and subtracting times

Time can be added and subtracted in seconds
Results are in POSIXct format

```
> dosedatetime
```

```
[1] "2004-04-15 14:00:00 CST"
```

```
> TAFDh <- 26.5
> TAFDsec <- TAFDh*60*60
> PKdatetime <- dosedatetime + TAFDsec
> PKdatetime
```

```
[1] "2004-04-16 16:30:00 CST"
```

The strftime function - POSIXct to textstring

PKdatetime is class POSIXct

```
> PKdatetime
```

```
[1] "2004-04-16 16:30:00 CST"
```

```
> PKdatetimetext <- strftime(PKdatetime, format = "%m-%d-%Y %I:%M %p")
> PKdatetimetext
```

```
[1] "04-16-2004 04:30 PM"
```

The strftime function - POSIXct to textstring

PKdatetime is class POSIXct

```
> PKdatetext <- strftime(PKdatetime, format = "%m/%d/%Y")
> PKdatetext
```

```
[1] "04/16/2004"
```

Dates and Times - Summary

Date and time data can be messy

Check databases for mixed date and time formats

NONMEM can read dates and times in limited formats

R provides ways of manipulating date and time data to and from:

- TAFD (Time after First Dose)
- TAD (Time After Dose)
- TNOM (Nominal Time after Dose)