Data Science with R
Summarising Data

Graham.Williams@togaware.com

9th June 2014

Visit http://onepager.togaware.com/ for more OnePageR's.

The required packages for this module include:

```
library(rattle)          # The weatherAUS dataset.
library(plyr)            # Group by operations.
```

As we work through this chapter, new R commands will be introduced. Be sure to review the command's documentation and understand what the command does. You can ask for help using the `?` command as in:

```
?read.csv
```

We can obtain documentation on a particular package using the *help=* option of `library()`:

```
library(help=rattle)
```

This chapter is intended to be hands on. To learn effectively, you are encouraged to have R running (e.g., RStudio) and to run all the commands as they appear here. Check that you get the same output, and you understand the output. Try some variations. Explore.

1 Load the Data

We use the full **weatherAUS** dataset from rattle (Williams, 2014) to illustrate data summarisation over a more complex dataset.

```
ds <- weatherAUS
names(ds)  <- normVarNames(names(ds)) # Lower case variable names.
names(ds)
```

```
##  [1] "date"            "location"        "min_temp"
##  [4] "max_temp"        "rainfall"        "evaporation"
##  [7] "sunshine"        "wind_gust_dir"   "wind_gust_speed"
## [10] "wind_dir_9am"    "wind_dir_3pm"    "wind_speed_9am"
....
```

```
head(ds)
```

```
##         date location min_temp max_temp rainfall evaporation sunshine
## 1 2008-12-01   Albury     13.4     22.9      0.6          NA       NA
## 2 2008-12-02   Albury      7.4     25.1      0.0          NA       NA
## 3 2008-12-03   Albury     12.9     25.7      0.0          NA       NA
....
```

```
tail(ds)
```

```
##             date location min_temp max_temp rainfall evaporation sunshine
## 88763 2014-04-20    Uluru     10.3     29.6        0          NA       NA
## 88764 2014-04-21    Uluru     11.3     30.5        0          NA       NA
## 88765 2014-04-22    Uluru     10.1     31.6        0          NA       NA
....
```

```
ds[sample(nrow(ds), 6),]
```

```
##             date     location min_temp max_temp rainfall evaporation
## 42691 2011-01-30    Melbourne     16.4     38.1      0.0         7.4
## 74988 2010-03-21        Perth     19.5     33.1      0.0         6.0
## 46982 2011-08-14     Portland      2.5     15.4      0.2         1.0
....
```

```
str(ds)
```

```
## 'data.frame': 88768 obs. of  24 variables:
##  $ date           : Date, format: "2008-12-01" "2008-12-02" ...
##  $ location       : Factor w/ 49 levels "Adelaide","Albany",..: 3 3 3 3 3 ...
##  $ min_temp       : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
....
```

```
summary(ds)
```

```
##       date                location        min_temp       max_temp
##  Min.   :2007-11-01   Canberra: 2279   Min.   :-8.5   Min.   :-3.8
##  1st Qu.:2010-03-08   Sydney  : 2187   1st Qu.: 7.6   1st Qu.:18.0
##  Median :2011-08-03   Adelaide: 2036   Median :12.0   Median :22.5
....
```

2 Dataset Indexing

Often we will be on the lookout for oddities or data typing that need fixing up. Once identified we will use the operations covered in a separate session on *Transforming* data.

We start by looking at some of the data. This introduces the concept of indexing our data frame.

```
ds[1,]                  # First observation.

##         date location min_temp max_temp rainfall evaporation sunshine
## 1 2008-12-01   Albury     13.4     22.9      0.6          NA       NA
##   wind_gust_dir wind_gust_speed wind_dir_9am wind_dir_3pm wind_speed_9am
## 1             W              44            W          WNW             20
....

ds[1,1]                 # First observation's first variable.

## [1] "2008-12-01"

ds[1:2,]                # First two observations.

##         date location min_temp max_temp rainfall evaporation sunshine
## 1 2008-12-01   Albury     13.4     22.9      0.6          NA       NA
## 2 2008-12-02   Albury      7.4     25.1      0.0          NA       NA
##   wind_gust_dir wind_gust_speed wind_dir_9am wind_dir_3pm wind_speed_9am
....

ds[1:2, 3:4]            # First two observations and variables 3 and 4.

##   min_temp max_temp
## 1     13.4     22.9
## 2      7.4     25.1

head(ds[3:4], 2)        # Single dimension treated as variable index.

##   min_temp max_temp
## 1     13.4     22.9
## 2      7.4     25.1

head(ds[,3:4], 2)       # Or we can leave the observation index empty.

##   min_temp max_temp
## 1     13.4     22.9
## 2      7.4     25.1
```

3 Textual Summaries

The `summary()` command provides a quick univariate overview of our dataset.

```
summary(ds, digits=6)

##       date                  location        min_temp       max_temp   
##  Min.   :2007-11-01   Canberra: 2279   Min.   :-8.5   Min.   :-3.8  
##  1st Qu.:2010-03-08   Sydney  : 2187   1st Qu.: 7.6   1st Qu.:18.0  
##  Median :2011-08-03   Adelaide: 2036   Median :12.0   Median :22.5  
##  Mean   :2011-07-29   Brisbane: 2036   Mean   :12.2   Mean   :23.1  
##  3rd Qu.:2012-11-27   Darwin  : 2036   3rd Qu.:16.8   3rd Qu.:28.0  
##  Max.   :2014-04-25   Hobart  : 2036   Max.   :33.9   Max.   :48.1  
##                       (Other) :76158   NA's   :669    NA's   :493   
##     rainfall       evaporation       sunshine     wind_gust_dir
##  Min.   :  0.0   Min.   : 0      Min.   : 0      W      : 5850  
##  1st Qu.:  0.0   1st Qu.: 3      1st Qu.: 5      SE     : 5796  
##  Median :  0.0   Median : 5      Median : 8      N      : 5739  
##  Mean   :  2.5   Mean   : 5      Mean   : 8      S      : 5669  
##  3rd Qu.:  0.8   3rd Qu.: 7      3rd Qu.:11      SSE    : 5583  
##  Max.   :371.0   Max.   :82      Max.   :14      (Other):53353  
##  NA's   :1656    NA's   :32687   NA's   :35530   NA's   : 6778  
##  wind_gust_speed  wind_dir_9am    wind_dir_3pm   wind_speed_9am
##  Min.   :  6     N      : 7200   SE     : 6928   Min.   : 0.0  
##  1st Qu.: 31     SE     : 5668   W      : 6115   1st Qu.: 7.0  
##  Median : 39     E      : 5568   S      : 6071   Median :13.0  
##  Mean   : 40     SSE    : 5508   WSW    : 5846   Mean   :14.2  
##  3rd Qu.: 48     S      : 5345   SSE    : 5804   3rd Qu.:20.0  
##  Max.   :135     (Other):52838   (Other):56048   Max.   :87.0  
##  NA's   :6738    NA's   : 6641   NA's   : 1956   NA's   :1159  
##  wind_speed_3pm  humidity_9am    humidity_3pm    pressure_9am 
##  Min.   : 0.0   Min.   :  0.0   Min.   :  0.0   Min.   : 980  
##  1st Qu.:13.0   1st Qu.: 57.0   1st Qu.: 37.0   1st Qu.:1013  
##  Median :19.0   Median : 70.0   Median : 52.0   Median :1017  
##  Mean   :18.8   Mean   : 68.7   Mean   : 51.6   Mean   :1017  
##  3rd Qu.:24.0   3rd Qu.: 83.0   3rd Qu.: 66.0   3rd Qu.:1022  
##  Max.   :87.0   Max.   :100.0   Max.   :100.0   Max.   :1041  
##  NA's   :1170   NA's   :1495    NA's   :1389    NA's   :8273  
##   pressure_3pm    cloud_9am       cloud_3pm        temp_9am   
##  Min.   : 979   Min.   :0       Min.   :0       Min.   :-5.9  
##  1st Qu.:1010   1st Qu.:1       1st Qu.:2       1st Qu.:12.3  
##  Median :1015   Median :5       Median :5       Median :16.7  
##  Mean   :1015   Mean   :4       Mean   :4       Mean   :17.0  
##  3rd Qu.:1020   3rd Qu.:7       3rd Qu.:7       3rd Qu.:21.5  
##  Max.   :1040   Max.   :9       Max.   :9       Max.   :40.2  
##  NA's   :8245   NA's   :32337   NA's   :33339   NA's   :1040  
....
```

4 Textual Summaries—Warning

Do be weary of the results provided by `summary()`. The `summary()` command rounds the results to 4 digits by default. This can surprise us sometimes when we find `min()` and the reported minimum value from `summary()` disagree! Let's look at some random data and notice the reported minimum value.

```
eg <- sample(1e6:(1e7-1), 100)
max(eg)

## [1] 9882363

min(eg)

## [1] 1146522

summary(eg)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1150000 3050000 5120000 5350000 8050000 9880000

summary(eg, digits=4)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1147000 3051000 5123000 5348000 8051000 9882000

summary(eg, digits=5)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1146500 3050700 5123000 5348100 8050900 9882400

summary(eg, digits=6)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1146520 3050710 5123030 5348100 8050880 9882360

summary(eg, digits=7)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1146522 3050710 5123028 5348103 8050881 9882363
```


5 PlyR: Summarise per Group to new Data Frame

The plyr (Wickham, 2014) package provides a clean and consistent approach to transforming data. We can easily, for example, transform a data frame into a new smaller data frame grouped by the location.

```
temps <- ddply(ds, "location", summarise,
               max=max(max_temp, na.rm=TRUE),
               min=min(min_temp, na.rm=TRUE))
temps

##            location  max  min
## 1          Adelaide 45.7  0.7
## 2            Albany 38.9  1.8
## 3            Albury 44.8 -2.5
....
```

The plyr package also provides the `.()` function as a convenient mechanism for listing variable names without the need to quote them. The function becomes more convenient when we have multiple variables to list.

```
temps <- ddply(ds, .(location), summarise,
               max=max(max_temp, na.rm=TRUE),
               min=min(min_temp, na.rm=TRUE))
temps

##            location  max  min
## 1          Adelaide 45.7  0.7
## 2            Albany 38.9  1.8
## 3            Albury 44.8 -2.5
....
```

We can review the resulting values, ordered by the maximum temperature.

```
temps[order(temps$max, decreasing=TRUE),]

##            location  max  min
## 49          Woomera 48.1  0.7
## 22            Moree 47.3 -3.3
## 20 MelbourneAirport 46.8 -0.4
....
```

Similarly, but ordered by the minimum temperature.

```
head(temps[order(temps$min),])

##        location  max  min
## 24  MountGinini 31.1 -8.5
## 41  Tuggeranong 40.1 -8.2
## 10     Canberra 42.0 -8.0
....
```

6 PlyR: Summarise per Group to Original Data Frame

Transform a data frame by adding the group summaries per original observation, simply by replacing `summarise` with `transform`

```
temps <- ddply(ds, .(location), transform,
               max=max(max_temp, na.rm=TRUE),
               min=min(min_temp, na.rm=TRUE))
```

Now notice that the top few values for `min` and `max` are constant, since they belong to the same group (`Adelaide`).

```
head(temps[c("date", "location", "min_temp", "min", "max_temp", "max")])

##         date location min_temp min max_temp  max
## 1 2008-07-01 Adelaide      8.8 0.7     15.7 45.7
## 2 2008-07-02 Adelaide     12.7 0.7     15.8 45.7
## 3 2008-07-03 Adelaide      6.2 0.7     15.1 45.7
....
```

If we same a few observations we see the various values of `min` and `max` across different `locations`.

```
temps[sample(nrow(temps), 10),
      c("date", "location", "min_temp", "min", "max_temp", "max")]

##             date      location min_temp  min max_temp  max
## 88579 2013-10-18       Woomera      7.7  0.7     27.8 48.1
## 87470 2010-07-08       Woomera      0.7  0.7     15.3 48.1
## 77879 2009-09-08       Walpole      5.5  3.4     17.8 39.3
....
```

7 PlyR: Select One Observation Per Group

We can also select a single observation per group, using some criteria to decide which observation to pick. We replace the `summarise` or `transform` with a function to select the observation of interest.

```
temps <- ddply(ds, .(location),
               function(x) x[x$max_temp == max(x$max_temp, na.rm=TRUE),])
head(temps[1:7])
```

```
##         date location min_temp max_temp rainfall evaporation sunshine
## 1       <NA>     <NA>       NA       NA       NA          NA       NA
## 2 2009-01-28 Adelaide     30.7     45.7        0        13.0     12.5
## 3 2010-01-18   Albany     17.8     38.9        0        11.8     12.8
....
```

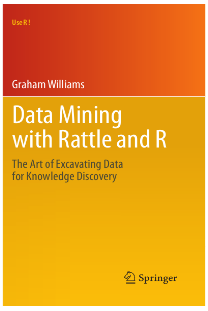
Notice the unexpected rows of missing values. The vector comparison, using `==()`, will return `NA` whenever comparing NA's and an index to `[()` of NA will return an NA row for each observation. We can get around this issue of missing values by testing whether we get `TRUE` from the comparison, rather than `FALSE` or `NA` by using `identical()`.

```
temps <- ddply(ds, .(location),
               function(x) x[sapply(x$max_temp == max(x$max_temp, na.rm=TRUE),
                                    identical, TRUE),])
head(temps[1:7])
```

```
##         date      location min_temp max_temp rainfall evaporation sunshine
## 1 2009-01-28      Adelaide     30.7     45.7        0        13.0     12.5
## 2 2010-01-18        Albany     17.8     38.9        0        11.8     12.8
## 3 2009-02-07        Albury     22.3     44.8        0          NA       NA
....
```

8 Further Reading

The Rattle Book, published by Springer, provides a comprehensive introduction to data mining and analytics using Rattle and R. It is available from Amazon. Other documentation on a broader selection of R topics of relevance to the data scientist is freely available from http://datamining.togaware.com, including the Datamining Desktop Survival Guide.



This module is one of many OnePageR modules available from http://onepager.togaware.com. In particular follow the links on the website with a * which indicates the generally more developed OnePageR modules.

9 References

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Wickham H (2014). *plyr: Tools for splitting, applying and combining data*. R package version 1.8.1, URL http://CRAN.R-project.org/package=plyr.

Williams GJ (2009). "Rattle: A Data Mining GUI for R." *The R Journal*, **1**(2), 45–55. URL http://journal.r-project.org/archive/2009-2/RJournal_2009-2_Williams.pdf.

Williams GJ (2011). *Data Mining with Rattle and R: The art of excavating data for knowledge discovery*. Use R! Springer, New York. URL http://www.amazon.com/gp/product/1441998896/ref=as_li_qf_sp_asin_tl?ie=UTF8&tag=togaware-20&linkCode=as2&camp=217145&creative=399373&creativeASIN=1441998896.

Williams GJ (2014). *rattle: Graphical user interface for data mining in R*. R package version 3.0.4, URL http://rattle.togaware.com/.

This document, sourced from SummaryO.Rnw revision 419, was processed by KnitR version 1.6 of 2014-05-24 and took 3.6 seconds to process. It was generated by gjw on nyx running Ubuntu 14.04 LTS with Intel(R) Xeon(R) CPU W3520 @ 2.67GHz having 4 cores and 12.3GB of RAM. It completed the processing 2014-06-09 10:27:58.