Data Science with R
Case Studies in Data Science

Graham.Williams@togaware.com

9th August 2014

Visit http://HandsOnDataScience.com/ for more Chapters.

In this chapter we work through numerous case studies using publicly available data, primarily from the Australian Government's open data web site, http://data.gov.au. We explain each of the steps and the R commands that achieve the required analyses.

The required packages for this chapter include:

```
library(xlsx)           # Read Excel spreadsheets.
library(rattle)         # normVarNames().
library(stringr)        # String manpiulation.
library(tidyr)          # Tidy the dataset.
library(dplyr)          # Data manipulation.
library(ggplot2)        # Visualise data.
library(scales)         # Include commas in numbers.
library(directlabels)   # Dodging labels for ggplot2.
```

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 ATO Web Analytics

The ATO provided some of its web log data for GovHack 2014. It is available on the Australian Government's data sharing web site `data.gov.au` and specifically within the ATO Web Analytics section. The CSV data files are provided within zip containers.

We record the locations here so that we can refer to them in the R code that follows.

```
datagovau <- "http://data.gov.au/dataset/"
atogovau  <- file.path(datagovau, "9b57d00a-da75-4db9-8899-6537dd60eeba/resource")
```

All the datasets are for the period from July 2013 to April 2014 broken down by month and by traffic source (internal or external). For convenience, since the period is used within the filenames, we also record the period as a string here, together with the actual months recorded.

```
period <- " - July 2013 to April 2014.csv"
months <- c("Jul-13", "Aug-13", "Sep-13", "Oct-13", "Nov-13",
            "Dec-13", "Jan-14", "Feb-14", "Mar-14", "Apr-14")
```

Each dataset is a comma-separated value (CSV) file. The actual filenames are a string of words separated by space. Each CSV file is contained within a separate ZIP archive, containing just that single CSV file. The ZIP files are named the same as the CSV file but all lowercase, no spaces, and replacing the .csv extension with .zip. Thus, for a ZIP file named something like:

`asamplefileofdataascsvjuly2013toapril2014.zip`

it will contain a CSV file named:

`A Sample File Of Data As CSV - July 2013 to April 2014.csv`

A support function will be useful to convert the CSV filenames into ZIP filenames (easier than the other way round). This involves string operations using stringr (Wickham, 2012), and we deploy the pipe operator (%>%) from magrittr (Bache and Wickham, 2014) for a simple and clearly expressed function.

```
csv2zip <- function(fname)
{
  fname %>% tolower() %>% str_replace_all(" |-|csv$", "") %>% str_c("zip")
}
```

To illustrate with the sample filenames as above:

```
tefname <- str_c("A Sample File Of Data As CSV", period)
tefname
```

```
## [1] "A Sample File Of Data As CSV - July 2013 to April 2014.csv"
```

That is converted into the following filename:

```
tezname <- csv2zip(tefname)
tezname
```

```
## [1] "asamplefileofdataascsvjuly2013toapril2014.zip"
```

This is how the providers of this particular dataset have chosen to package their data.

1.1 The ATO Web Analytics Datasets

There are seven datasets provided through data.gov.au. Each of the CSV filenames is noted and recorded here. We can then automatically download the ZIP file into a temporary location, extract the CSV, and read it into R.

The text descriptions are those provided on the web site.

Browser Access Browsers used to access the ATO website.

```
brfname <- str_c("Browser by month and traffic source", period)
brzname <- csv2zip(brfname)
```

Entry Pages Starting (entry) pages for the ATO website.

```
enfname <- str_c("Entry pages by month and traffic source", period)
enzname <- csv2zip(enfname)
```

Site Referrers Websites linking to ato.gov.au (entry referrers) that were used to access the ATO website content.

```
refname <- str_c("Entry referrers by month and traffic source", period)
rezname <- csv2zip(refname)
```

Exit Pages Pages on which users left the ATO website (exit pages).

```
exfname <- str_c("Exit pages by month and traffic source", period)
exzname <- csv2zip(exfname)
```

Local Keyword Searches Keywords and phrases used in the ATO website search engine.

```
kwfname <- str_c("Local keywords (top 100) by month and traffic source", period)
kwzname <- csv2zip(kwfname)
```

Operating System Access Operating system used to access the ATO website.

```
osfname <- str_c("Operating System (platform) by month and traffic", period)
oszname <- csv2zip(osfname)
```

Pages Viewed Pages viewed on the ATO website.

```
vifname <- str_c("Pages by month and traffic source", period)
vizname <- csv2zip(vifname)
```

In the following single page sections we analyse the various datasets in a variety of ways.

2 ATO Entry Pages

The path to the file containing the dataset to download is constructed as a URL. We begin with the string we created earlier, `atogovau`, and append an internal identifier from the web site that locates the dataset web page. We then point to the download area and the ZIP filename to download. The `file.path()` command adds the path separator "/" between its arguments:

```
fname <- file.path(atogovau,
                   "121dfe58-044b-4f21-b4ee-6f9335a44413",
                   "download",
                   enzname)
fname
```

```
## [1] "http://data.gov.au/dataset//9b57d00a-da75-4db9-8899-6537dd60eeba/reso...
```

We next create a new temporary local file using `tempfile()` and then cause the dataset to be downloaded from the Internet into this temporary file using `download.file()`:

```
temp  <- tempfile(fileext=".zip")
download.file(fname, temp)
```

The progress of the download will be displayed interactively.

The content of the ZIP file is then extracted using `unz()`, which reads the ZIP file and extracts the specified file. It actually opens a so-called connection and the output of that connection is piped here into `read.csv()`. We also remove the temporary file using `unlink()`.

```
entry <- temp %>% unz(enfname) %>% read.csv()
unlink(temp)
```

We will load this dataset into our generic variable `ds` after piping it into `tbl_df()` from dplyr (Wickham and Francois, 2014) as a convenience for printing the data frame:

```
dsname <- "entry"
ds     <- dsname %>% get() %>% tbl_df()
```

As we often do, the variable names are "simplified," or at least normalised, using `normVarNames()` from rattle (Williams, 2014). We record the list of available variables in *vars*.

```
names(ds) <- normVarNames(names(ds))
names(ds)
```

```
## [1] "entry_page" "month"      "source"     "views"      "visits"
```

```
vars <- names(ds)
```

Often we will want to experiment with our datasets and then perhaps revert to a clean original dataset. It is a good idea to take a copy of the dataset, as we do below, calling it *dso*. Note that this is the source dataset, *entry*, with the normalised variable names. We can remove (using `rm()`) the actual source dataset so as to reduce our memory footprint. This is particularly important for large datasets.

```
dso <- ds
rm(entry)
```

2.1 Explore

We can now explore some of the characteristics of the dataset. Our aim, as always, is to get familiar with it—part of *living and breathing the data.*

```
ds
```

```
## Source: local data frame [207,119 x 5]
##
##                                                                    entry_...
## 1                                                    http://www.ato.gov...
## 2                                                    http://www.ato.gov...
## 3            http://www.ato.gov.au/individuals/lodging-your-tax-return/e-...
## 4            http://www.ato.gov.au/individuals/lodging-your-tax-return/e-...
....
```

Out of the 207,119, there are 33,262 different entry points.

```
length(levels(ds$entry_page))
```

```
## [1] 33262
```

We summarise the remaining columns to get a feel for what theses columns might be recording.

```
summary(ds[-1])
```

```
##      month            source           views             visits
##  Apr-14 :23324   4       :     1   Min.   :      1   Min.   :      1
##  Jul-13 :23067   External:144400   1st Qu.:      4   1st Qu.:      1
##  Aug-13 :22849   Internal: 62718   Median :     18   Median :      4
##  Oct-13 :22736                     Mean   :   1019   Mean   :    169
##  Nov-13 :22478                     3rd Qu.:     88   3rd Qu.:     20
##  Sep-13 :21084                     Max.   :9106745   Max.   :1349083
##  (Other):71581                                       NA's   :1
```

The variable *month* appears to report the month and year. The *source* looks to record, presumably, whether the person browsing is external to the ATO or internal to the ATO. For an entry point the *views* looks to report the number of views for that month (and broken down between internal and external views). Similarly for *visits*.

There appears to be an odd value for source (the `4`) and we might wonder whether there is a data quality issue here. Let's have a look at the full frequency table for *month*:

```
table(ds$month)
```

```
##
##   Apr-14   Aug-13   Dec-13 External   Feb-14   Jan-14   Jul-13   Mar-14
##    23324    22849    19090        1    17222    17266    23067    18002
##   Nov-13   Oct-13   Sep-13
....
```

Something odd happening there. `External` should not be a value for *month*—it belongs to the next column (*source*). There is apparently some misalignment for this one observarion.

2.2 Data Quality Issue

Having identified a data quality issue we need to have a closer look to determine how widespread the issue is. Of course, initial indications are that there is a single bad observation in the dataset.

The first task is to identify where the errant observation is:

```
which(ds$source=="4")
```

```
## [1] 198499
```

We see that this is observation number 198,499.

Going back to the original source CSV file we need to find that row in the file itself. There are many ways to do this, including loading the file into a spreadsheet application or even just a text editor. A simple Linux command line approach will pipe the results of `tail` into `head` as in the following command line (replacing the `<filename>` with the actual filename). Even for extremely large files, this will take almost no time at all as both commands are very efficient, and is likely quicker than loading the data into Libre Office (or Micorsoft/Excel for that matter):

```
$ tail -n+198495 <filename>.csv | head -n10
```

We discovery that row 198,500 (given that the first row of the CSV file is the header row) has the literal value:

```
"http://www.ato.gov.au/content/00268103.htm "",March 2014""",External,4,3,
```

Compare that to another row that is valid:

```
http://www.ato.gov.au/content/00171495.htm,Mar-14,External,7,2
```

Something would appear to have gone wrong in the extraction of the dataset at source—the CSV file is the original from the web site so perhaps the extraction at the ATO was less than perfect.

We will simply remove that errant row:

```
dim(ds)
```

```
## [1] 207119      5
```

```
issue <- which(ds$source=="4")
issue
```

```
## [1] 198499
```

```
dso <- ds <- ds[-issue,]
dim(ds)
```

```
## [1] 207118      5
```

Notice the use of `which()` to identify the observation number that contains the error, and then `-issue` as the index of the data frame, which removes that row.

There is no further evidence that there is any other similar issues int he dataset for now, so we can proceed with our analysis.

2.3 Clean And Enhance

We might notice that the levels for the `month` are in alphabetic order whilst we would normally want these to be ordered chronologically. We can fix that:

```
levels(ds$month)
```

```
##  [1] "Apr-14"   "Aug-13"   "Dec-13"   "External" "Feb-14"   "Jan-14"
##  [7] "Jul-13"   "Mar-14"   "Nov-13"   "Oct-13"   "Sep-13"
```

```
ds$month <- factor(ds$month, levels=months)
levels(ds$month)
```

```
##  [1] "Jul-13" "Aug-13" "Sep-13" "Oct-13" "Nov-13" "Dec-13" "Jan-14"
##  [8] "Feb-14" "Mar-14" "Apr-14"
```

2.4 Explore Some More

The total number of views/visits to the ATO website may be interest.

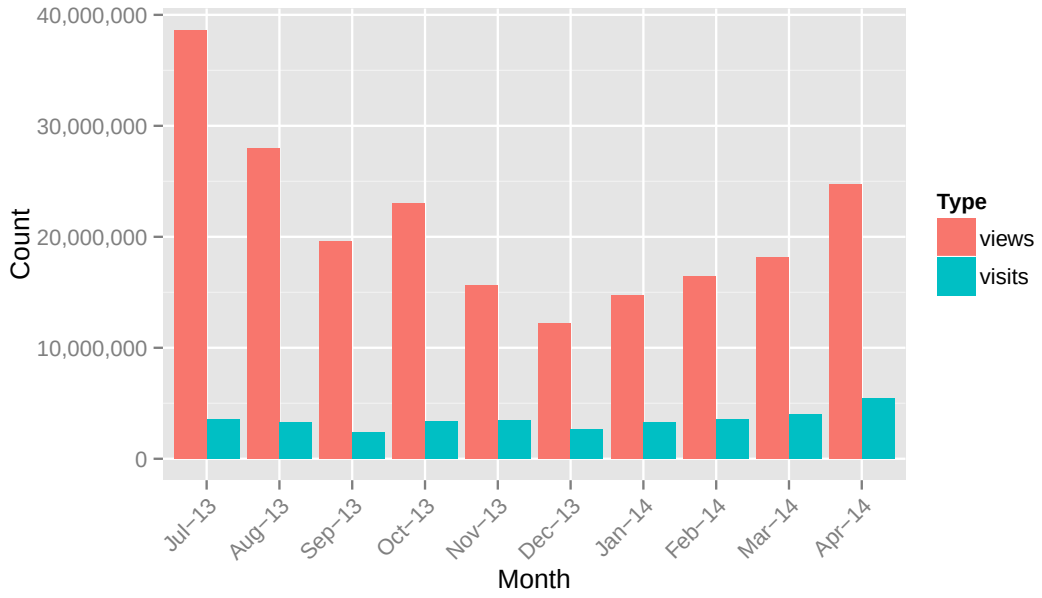
```
format(sum(ds$views), big.mark=",")
```

```
## [1] "211,110,271"
```

```
format(sum(ds$visits), big.mark=",")
```

```
## [1] "35,050,249"
```

We can then explore the views/visits per month.



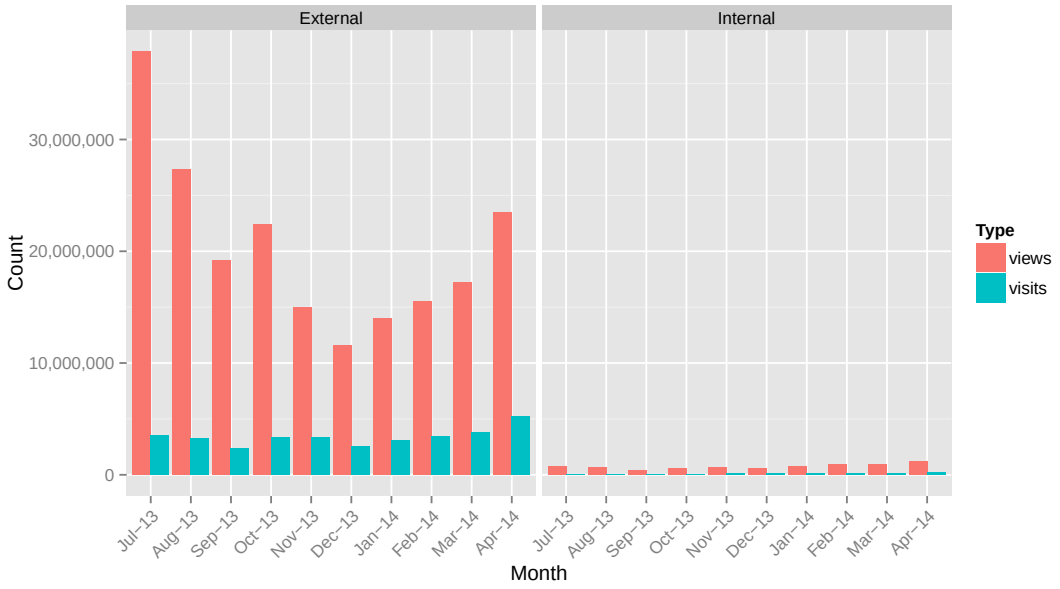

```
ds                                                  %>%
  group_by(month)                                   %>%
  summarise(views=sum(views), visits=sum(visits))   %>%
  gather(type, count, -month)                       %>%
  ggplot(aes(x=month, y=count, fill=type))           +
  geom_bar(stat="identity", position="dodge")        +
  scale_y_continuous(labels=comma)                   +
  labs(fill="Type", x="Month", y="Count")            +
  theme(axis.text.x=element_text(angle=45, hjust=1))
```

We can see an interesting pattern of views versus visits in that there's a reasonably flat number of visits over the period, however the number of views (and also we would suggest views per visit) is dramatically increased for July. We would really need to analyse the relative change in views/visit over time to confirm that observation, but we'll stay with the visual for now.

The July spike may well correspond to the Australian financial year ending in June and starting in July. We might also observe the holiday season around December when there must be less interest in taxation topics.

2.5 External versus Internal Views/Visits per Month

The breakdown between `External` and `Internal` may be of interest.




```
ds %>%
  group_by(month, source)                              %>%
  summarise(views=sum(views), visits=sum(visits))      %>%
  gather(type, count, -c(month, source))               %>%
  ggplot(aes(x=month, y=count, fill=type))              +
  geom_bar(stat="identity", position="dodge")           +
  scale_y_continuous(labels=comma)                      +
  labs(fill="Type", x="Month", y="Count")               +
  theme(axis.text.x=element_text(angle=45, hjust=1)) +
  facet_wrap(~source)
```

We immediately see that relatively speaking there are very few internal views/visits. This should not be surprising, as the ATO has only about 20,000 staff, compared to the population of Australia at over 23 million. Of course, access to the ATO web site is not limited to Australia. This distribution between the external and internal sources helps to garner our confidence in the quality of the data and in our understanding of the actual variables (i.e., we have perhaps a valid interpretation of what *source* records).

Given the clear differentiation between the external and internal populations, we might think to partition our analysis into the two cohorts. At a guess, we might expect the behaviours exhibited internally to be quite different to those exhibited through external accesses.

2.6 External Visits

We might want to examine just the external (i.e., the public, so we assume) browsing of the ATO web site.

```
ds <- ds %>% subset(source=="External")
dim(ds)
```

```
## [1] 144400      5
```

```
ds
```

```
## Source: local data frame [144,400 x 5]
##
##                                                                    entry_...
## 1                                                  http://www.ato.gov...
....
```

We now want to review some of the entry pages. Let's pick a random sample:

```
ds$entry_page[sample(nrow(ds), 10)]
```

```
##  [1] http://www.ato.gov.au/content/16689.htm                          ...
##  [2] http://www.ato.gov.au/tax-professionals/consultation--tax-practitione...
##  [3] http://www.ato.gov.au/content/00210181.htm                       ...
##  [4] http://www.ato.gov.au/general/new-legislation/in-detail/a-z-index/   ...
....
```

```
ds$entry_page %>% unique() %>% length()
```

```
## [1] 31363
```

We can list the most frequent ones:

```
tbl <- ds$entry_page %>% table()
summary(tbl)
```

```
## Number of cases in table: 144400
## Number of factors: 1
```

We should carefully take a look at this table as on first impressions it might not be what we expected. Why are there many zeros, and why does it only go up to 11?

This brings us back to ensuring we understand what the dataset records. We may have lost sight by now of the fact that this is not a unit level dataset—i.e., it is not a record of individual visits. Rather, the dataset consists of *views* and *visits* aggregated monthly. So to get a real indication of entry page's popularity over the whole period we should aggregate the overall data.

```
dsa <- ds %>% group_by(entry_page) %>% summarise(total=sum(visits))
head(dsa$total)
```

```
## [1] 197   3   1   2   2   1
```

Exercise: Split the URLs into components and aggregate and analyse.

3 ATO Browser Data

The ATO browser dataset is extracted in the same way. The specific project code is again obtained from the URL from data.gov.au. We download the ZIP file into a temporary location, extract the CSV, and read it into R.

```
fname <- file.path(atogovau,
                   "08e33518-dd6f-42d8-a035-f4c0fc7f18f3",
                   "download",
                   brzname)
temp  <- tempfile(fileext=".zip")
download.file(fname, temp)
browser <- read.csv(unz(temp, brfname))
unlink(temp)
```

```
dsname       <- "browser"
ds           <- dsname %>% get() %>% tbl_df()
names(ds)    <- normVarNames(names(ds))
names(ds)[3] <- "source"
names(ds)
```

```
## [1] "browser" "month"   "source"  "views"   "visits"
```

```
vars         <- names(ds)
dso          <- ds
ds
```

```
## Source: local data frame [1,357 x 5]
##
##                             browser  month   source   views visits
## 1                            Chrome Jul-13 External 7765921 691120
## 2                            Chrome Jul-13 Internal     454    110
## 3  Microsoft Internet Explorer 10.x Jul-13 External 6773492 557509
## 4                     Mobile Safari Jul-13 External 6067298 519093
## 5   Microsoft Internet Explorer 8.x Jul-13 External 5078805 455815
## 6   Microsoft Internet Explorer 8.x Jul-13 Internal  666710  69007
## 7                           Firefox Jul-13 External 4440016 436873
## 8                           Firefox Jul-13 Internal   19699   1387
## 9   Microsoft Internet Explorer 9.x Jul-13 External 3923983 386112
## 10                           Safari Jul-13 External 2168245 245044
## ..                              ...    ...      ...     ...    ...
```

3.1 Explore

Now we continue *living and breathing the data:*

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

```
## Source: local data frame [6 x 5]
##
##                       browser  month   source views visits
## 1233  Netscape Navigator 4.x Apr-14 External  4204   3698
....
```

```
summary(ds)
```

```
##                              browser         month          source
##  Chrome                          :  20   Mar-14 :163   External:1304
##  Firefox                         :  20   Oct-13 :143   Internal:  53
##  Microsoft Internet Explorer 6.x:  20   Apr-14 :141
##  Microsoft Internet Explorer 7.x:  20   Nov-13 :137
##  Microsoft Internet Explorer 8.x:  20   Feb-14 :136
##  External                        :  11   Jan-14 :132
##  (Other)                         :1246   (Other):505
....
```

We can see that the data records aggregated monthly observations of the browsers connecting to the ATO web site. The connection may be internal to the orgranisation or external. It would appear most connections are external. Given a browser, month, and source, we have provided for us the aggregated number of views and visits.

3.2 Clean And Enhance

We might notice that the levels for the `month` are in alphabetic order whilst we would normally want these to be ordered chronologically. We can fix that:

```
levels(ds$month)
```

```
##  [1] "Apr-14" "Aug-13" "Dec-13" "Feb-14" "Jan-14" "Jul-13" "Mar-14"
##  [8] "Nov-13" "Oct-13" "Sep-13"
```

```
ds$month <- factor(ds$month, levels=months)
levels(ds$month)
```

```
##  [1] "Jul-13" "Aug-13" "Sep-13" "Oct-13" "Nov-13" "Dec-13" "Jan-14"
##  [8] "Feb-14" "Mar-14" "Apr-14"
```

We might be interested in the average number of views per visit, We can simply add this as an extra variable of the dataset.

```
ds$ratio <- ds$views/ds$visits
summary(ds$ratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    2.00    3.33    4.00  151.00
```

3.3 Explore Internal Usage

Let's get some understanding of the internal versus external profiles of browser usage. Firstly, how many internal versus external visits? The dplyr package is the way to go here. We pass the dataset on to the `group_by()` function, grouping the dataset by the values of the `source` variable, and then on to `summarise()` to construct a new data frame tabling the results.

```
freq <- ds                       %>%
  group_by(source)               %>%
  summarise(total=sum(visits))
freq
```

```
## Source: local data frame [2 x 2]
##
##     source    total
## 1 External 33946553
## 2 Internal  1103712
```

We see that internal visits account for just 3% of all visits.

For the Internal users we now check which browsers are being used. Once again dplyr comes in handy.

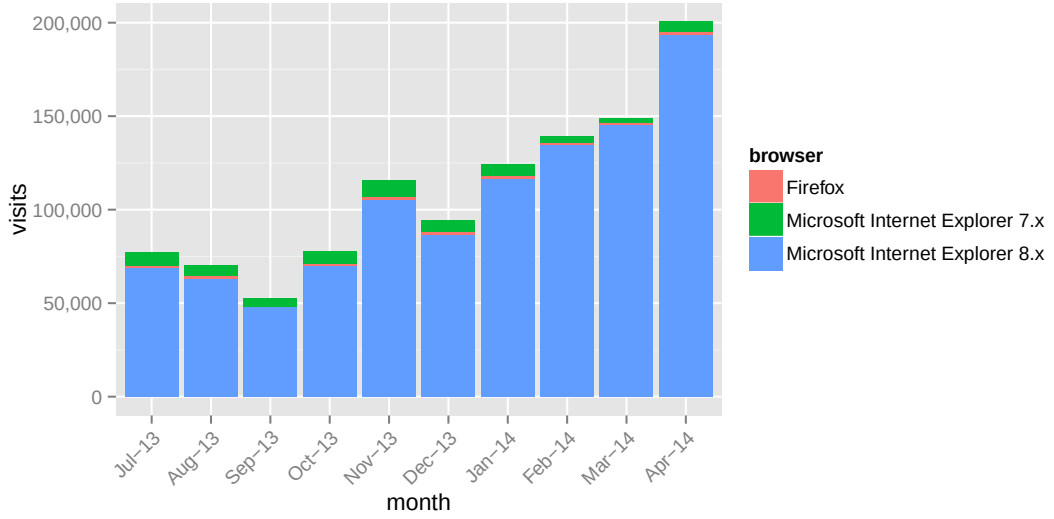
```
ib <- ds                         %>%
  filter(source == "Internal")   %>%
  group_by(browser)              %>%
  summarise(total=sum(visits))   %>%
  arrange(desc(total))
ib
```

```
## Source: local data frame [8 x 2]
##
##                            browser   total
## 1 Microsoft Internet Explorer 8.x 1032002
## 2 Microsoft Internet Explorer 7.x   56681
## 3                         Firefox   12793
## 4 Microsoft Internet Explorer 6.x    1440
## 5                          Chrome     792
## 6                        External       2
## 7 Microsoft Internet Explorer 9.x       1
## 8                          Safari       1
```

The ATO apparently deploys Microsoft Internet Explorer 8 as part of its standard operating environment. There's a few other browsers, but they are relatively rarely used.

3.4 Internal Usage Over Time

It could be interesting to explore the browser usage profile over time. For this we will plot using ggplot2 (Wickham and Chang, 2014). A simple bar chart might be informative. Here we include all observations where the source is Internal and we have a decent number of visits (smaller number of visits will not show up the in the plot even if they are included in the data because there are relatively far to few).




```
ds                                                  %>%
  filter(source=="Internal", visits > 1000)         %>%
  ggplot(aes(month, visits, fill=browser))           +
  geom_bar(stat="identity")                          +
  scale_y_continuous(labels=comma)                   +
  theme(axis.text.x=element_text(angle=45, hjust=1))
```

It is interesting, if also puzzling, to see quite an increase in visits over the 10 months. We might ask if the data is actually complete as a 4-fold increase in internal visits between September 2013 and April 2014 sounds rather sudden.

Exercise: Ensure the legend is the same order as the plot.

3.5 External Visits

We might compare the internal browser usage to the external browser usage

```
eb <- ds                          %>%
  filter(source == "External") %>%
  group_by(browser)            %>%
  summarise(total=sum(visits)) %>%
  arrange(desc(total))
head(eb, 10)
```

```
## Source: local data frame [10 x 2]
##
##                             browser   total
## 1                            Chrome 7508320
## 2  Microsoft Internet Explorer 10.x 5258583
## 3                     Mobile Safari 5016803
## 4                           Firefox 4296359
## 5   Microsoft Internet Explorer 8.x 3736043
## 6   Microsoft Internet Explorer 9.x 3458002
## 7                            Safari 2282737
## 8   Microsoft Internet Explorer 7.x 1498382
## 9                           Mozilla  634484
## 10  Microsoft Internet Explorer 6.x  151233
```

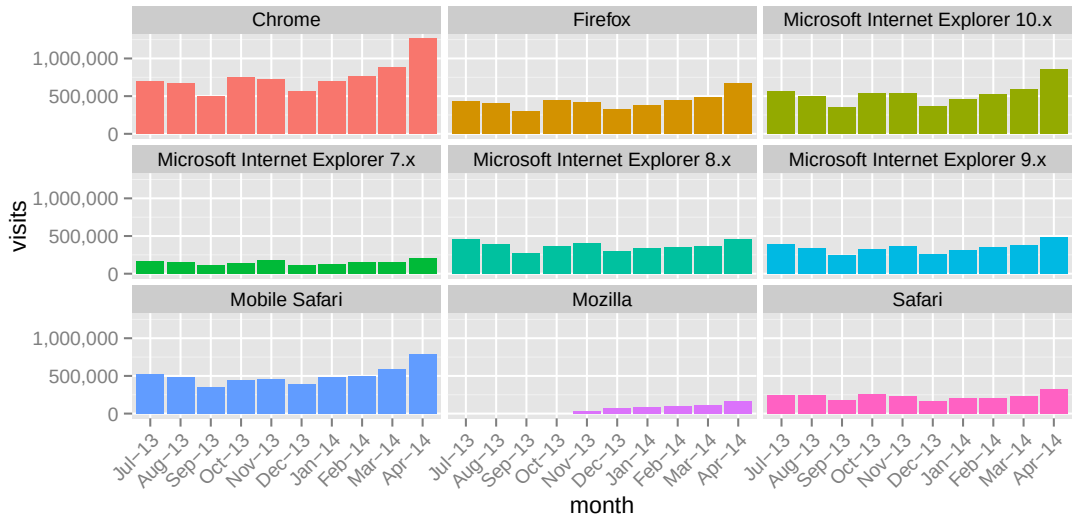
The ATO probably has careful control of the use of browsers within the organisation. Outside of the organistion, the choice of free and open source browsers is clearly more dominant.

```
tail(eb, 15)
```

```
## Source: local data frame [15 x 2]
##
##                            browser total
## 392                        steller     1
## 393             Swingtel SX3 Linux     1
## 394                   travelviajes     1
## 395                        TTECHNO     1
## 396                     TUUIUNMQAA     1
## 397           TyroTerminalAdapter      1
## 398      undefined GoogleToolbarBB     1
## 399                            w3m     1
## 400    windows internet explorer 9     1
## 401                     Windows NT     1
## 402                  X9 NOTE Linux     1
## 403              Xiaomi 2013061 TD     1
## 404 xmlset roodkcableoj28840ybtide     1
## 405                     ZTEU795 TD     1
## 406                    ZTEU880F1 TD     1
```

Looking at the other end of the scale, we see quite a spread of single hit browsers. In fact, out of the 406 there are 142 browsers with a single visit, 297 with less than 10 visits, 372 with less than 100 visits, 391 with less than 1000 visits, and 394with less than 20,000 visits.

3.6 External Visits




```
ds                                                    %>%
  filter(source == "External", visits > 20000)        %>%
  ggplot(aes(month, visits, fill=browser))             +
  geom_bar(stat="identity")                            +
  facet_wrap(~browser)                                 +
  scale_y_continuous(labels=comma)                     +
  theme(axis.text.x=element_text(angle=45, hjust=1))   +
  theme(legend.position="none")
```

4 ATO Top 100 Keywords

We now load the ATO top 100 keywords for analysis. Once again, the specific project ID is obtained from the project URL at data.gov.au. We download the ZIP file into a temporary location, extract the CSV, and read it into R as the *keywords* dataset.

```
fname    <- file.path(atogovau,
                      "c6b745ec-439c-4dc4-872a-f48f5368d58e",
                      "download",
                      kwzname)
temp     <- tempfile(fileext=".zip")
download.file(fname, temp)
keywords <- read.csv(unz(temp, kwfname))
unlink(temp)
```

We prepare the dataset for analysis, and as usual we copy it to the generic *ds*, normalise and simplify the column names, order the months chronologically, and save a copy as *dso*:

```
dsname       <- "keywords"
ds           <- dsname %>% get() %>% tbl_df()
names(ds)    <- normVarNames(names(ds))
names(ds)[1] <- "keyword"
names(ds)[3] <- "source"
ds$month     <- factor(ds$month, levels=months)
vars         <- names(ds)
dso          <- ds
```

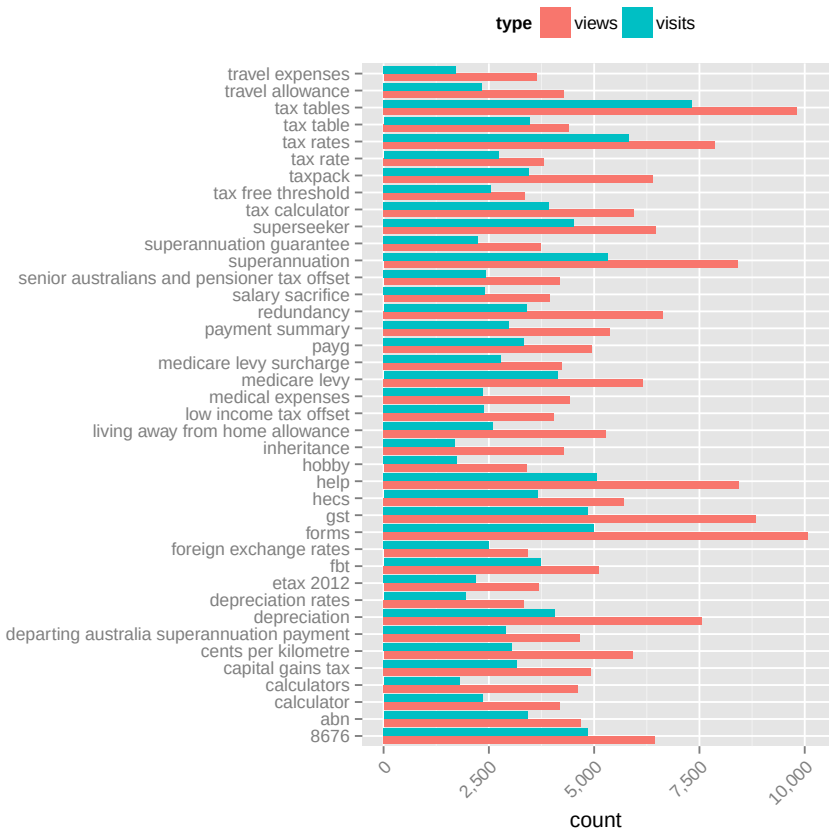
```
ds
```

```
## Source: local data frame [1,951 x 5]
##
##                            keyword  month   source views visits
## 1                          taxpack Jul-13 External  2262    942
## 2                          taxpack Jul-13 Internal    43     24
## 3                  payment summary Jul-13 External  2467   1050
## 4                  payment summary Jul-13 Internal    33     21
....
```

```
ds                                                  %>%
  group_by(keyword)                                 %>%
  summarise(views=sum(views), visits=sum(visits))   %>%
  arrange(desc(views))                              %>%
  head(40)
```

```
## Source: local data frame [40 x 3]
##
##                                     keyword views visits
## 1                                     forms 10078   5004
## 2                                 tax tables  9820   7325
## 3                                       gst  8837   4857
## 4                                      help  8439   5075
....
```

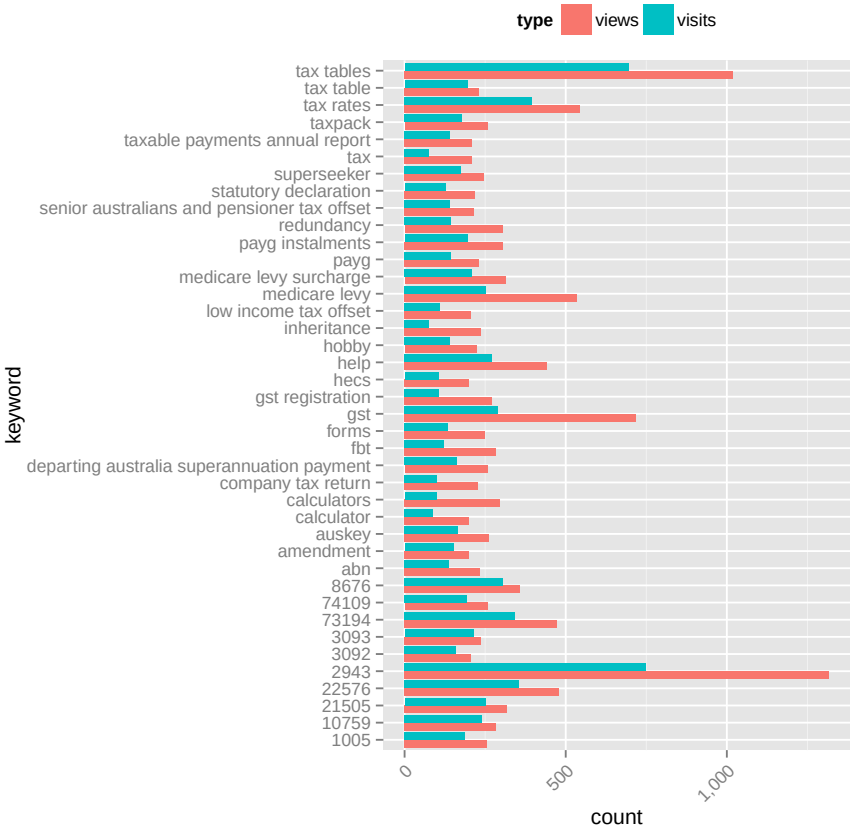
4.1 Plot Top 40




```
ds                                                    %>%
  group_by(keyword)                                   %>%
  summarise(views=sum(views), visits=sum(visits))     %>%
  arrange(desc(views))                                %>%
  head(40)                                            %>%
  gather(type, count, -keyword)                       %>%
  ggplot(aes(x=keyword, y=count, fill=type))           +
  geom_bar(stat="identity", position="dodge")          +
  scale_y_continuous(labels=comma)                     +
  theme(axis.text.x=element_text(angle=45, hjust=1))   +
  theme(legend.position="top")                         +
  labs(x="")                                           +
  coord_flip()
```

Exercise: Order by the count of views with highest at the top.

4.2 Internal Only




```
ds                                                    %>%
  subset(source=="Internal") %>%
  group_by(keyword)                                   %>%
  summarise(views=sum(views), visits=sum(visits))     %>%
  arrange(desc(views))                                %>%
  head(40)                                            %>%
  gather(type, count, -keyword)                       %>%
  ggplot(aes(x=keyword, y=count, fill=type))           +
  geom_bar(stat="identity", position="dodge")          +
  scale_y_continuous(labels=comma)                     +
  theme(axis.text.x=element_text(angle=45, hjust=1)) +
  theme(legend.position="top")                         +
  coord_flip()
```

5 ATO OS Data

The ATO operating system dataset is loaded and analysed. Once again, the specific project ID is obtained from the URL from data.gov.au. We download the ZIP file into a temporary location, extract the CSV, and read it into R.

6 Risk Scoring Dashboard

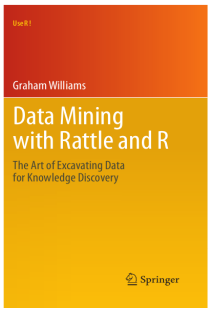
UNDER DEVELOPMENT

```
library(gdata)

ds <- read.xls("results.xlsx")
dim(ds)
```

7 Further Reading and Acknowledgements

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 chapter is one of many chapters available from `http://HandsOnDataScience.com`. In particular follow the links on the website with a * which indicates the generally more developed chapters.

8 References

Bache SM, Wickham H (2014). *magrittr: magrittr - a forward-pipe operator for R.* R package version 1.0.1, URL http://CRAN.R-project.org/package=magrittr.

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 (2012). *stringr: Make it easier to work with strings.* R package version 0.6.2, URL http://CRAN.R-project.org/package=stringr.

Wickham H, Chang W (2014). *ggplot2: An implementation of the Grammar of Graphics.* R package version 1.0.0, URL http://CRAN.R-project.org/package=ggplot2.

Wickham H, Francois R (2014). *dplyr: dplyr: a grammar of data manipulation.* R package version 0.2, URL http://CRAN.R-project.org/package=dplyr.

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 3.1.4, URL http://rattle.togaware.com/.

This document, sourced from CaseStudiesO.Rnw revision 479, was processed by KnitR version 1.6 of 2014-05-24 and took 8 seconds to process. It was generated by gjw on nyx running Ubuntu 14.04.1 LTS with Intel(R) Xeon(R) CPU W3520 @ 2.67GHz having 4 cores and 12.3GB of RAM. It completed the processing 2014-08-09 20:39:02.