In this module we introduce approaches to manipulate and transform our data.

The required packages for this module include:

```r
library(rattle)  # The weatherAUS datasets and normVarNames()
library(ggplot2)  # Visualise the transforms.
library(plyr)    # Transform using ddplyr()
library(dplyr)   # Transform using dplyr()
library(reshape2) # melt() and dcast()
```

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 Data

```r
library(rattle)
ls <- weatherAUS
names(ls) <- normVarNames(names(ls)) # Lower case variable names.
str(ls)
```

### '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 ...

....
2 Factors
3 Factors: Drop Unused Levels

The complete list of levels of a factor are maintained even when we take a subset of a dataset which contains only a subset of the original levels.

For example, suppose we subset the weather dataset on location:

```r
cities <- c("Adelaide", "Brisbane", "Canberra", "Darwin")
levels(ds$location)
## [1] "Adelaide"  "Albany"   "Albury"  
## [4] "AliceSprings" "BadgersCreek" "Ballarat" 
## [7] "Bendigo"    "Brisbane"   "Cairns"  
## [10] "Canberra"   "Cobar"     "CoffsHarbour" ....
summary(ds$location)
## Adelaide Albany Albury AliceSprings  
## 2036 1883 1883 1883  
## BadgersCreek Ballarat Bendigo Brisbane  
## 1852 1883 1883 2036 ....
dss <- subset(ds, location %in% cities)
levels(dss$location)
## [1] "Adelaide"  "Brisbane" "Canberra" "Darwin" 
summary(dss$location)
## Adelaide Brisbane Canberra Darwin  
## 2036 2036 2279 2036  
```
4 Factors: Reorder Levels

By default the levels of a factor are ordered alphabetically. We can change the order simply by providing `levels=` to `factor()`.

```r
levels(dss$location)
## [1] "Adelaide" "Brisbane" "Canberra" "Darwin"

summary(dss$location)
## Adelaide Brisbane Canberra Darwin
## 2036 2036 2279 2036

dss$location <- factor(dss$location, levels=rev(levels(dss$location)))

levels(dss$location)
## [1] "Darwin" "Canberra" "Brisbane" "Adelaide"

summary(dss$location)
## Darwin Canberra Brisbane Adelaide
## 2036 2279 2036 2036
```
5 Data Frame: Add a Column

Here we simply name the column as part of the data frame and it gets added to it.

```r
d$s.temp_range <- ds$max_temp - ds$min_temp
str(ds)
```

```
## 'data.frame': 88768 obs. of 25 variables:
##  $ date      : Date, format: "2008-12-01" "2008-12-02" ...
##  $ location  : Factor w/ 49 levels "Adelaide","Albany",...
##  $ min_temp  : num 13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
##  ....
p <- ggplot(ds, aes(x=temp_range))
p <- p + geom_bar(binwidth=1)
p
```

![Graph showing the distribution of temperature ranges.](image-url)
6 Transform: Add a Column

An alternative is to use `transform()` which can be neater when adding several columns, avoiding the use of the `$` nomenclature.

```r
ds <- transform(ds,
               temp_range=max_temp-min_temp,
               excess=rainfall-evaporation)
sum(ds$excess, na.rm=TRUE)
## [1] -176068
str(ds)
## 'data.frame': 88768 obs. of 26 variables:
##  $ date : Date, format: "2008-12-01" "2008-12-02" ...
##  $ location : Factor w/ 49 levels "Adelaide","Albany",...
##  $ min_temp : num 13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 ...
##  ...
## ggplot(ds, aes(x=excess)) + geom_bar(binwidth=1)
```

```
        0
count
        2000
        4000
        6000
 0  100  200
excess
```

Copyright © 2013-2014 Graham@togaware.com Module: TransformO Page: 6 of 16
7 Subset Data

Exercise: Research the subset() function and illustrate its usage.
8 Transform Using DplyR

The `plyr` (Wickham, 2014a) package provides a collection of the most useful functions for manipulating data. It’s concepts, once understood, are very powerful and allow us to express numerous tasks simply and efficiently.

Like `apply()`, the `plyr` functions operate on data frames, matrices, lists, vectors or arrays. An operation is applied to some collection of items (e.g., each group of observations or group of list elements) in the input data structure, and the results are packaged into a new data structure.

Generally, the pattern is like `ddply(data, variables, function, ...)` where in this case (as indicated by the first d) the input data is a data frame and the result (the second d) is also a data frame. The rows of the data frame will be grouped by the variables identified, and for each group the function is applied to obtain the resulting data. The remaining arguments are treated as arguments to the function.

**Exercise:** Explore and provide examples.
9 Summarise Data Using dplyr()

*dplyr* (Wickham and Francois, 2014) introduces a grammar of data manipulation and processes data much more efficiently than *plyr* (Wickham, 2014a) (anywhere from 20 times to 1000 times faster) and other R packages through parallel processing using Rcpp (Eddelbuettel and Francois, 2014).

```r
weatherAUS %>%
  group_by(Location) %>%
  summarise(total = sum(Rainfall)) %>%
  arrange(desc(total)) %>%
  head(5)
```

## Source: local data frame [5 x 2]

<table>
<thead>
<tr>
<th>Location</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Darwin</td>
<td>11092</td>
</tr>
<tr>
<td>SydneyAirport</td>
<td>5295</td>
</tr>
<tr>
<td>MountGambier</td>
<td>4050</td>
</tr>
<tr>
<td>Perth</td>
<td>3723</td>
</tr>
<tr>
<td>Bendigo</td>
<td>3286</td>
</tr>
</tbody>
</table>
## 10 Removing Columns

```r
library(tidyverse)

tail(ds$excess)
## [1] NA NA NA NA NA

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"
## [13] "wind_speed_3pm" "humidity_9am" "humidity_3pm"
## [16] "pressure_9am" "pressure_3pm" "cloud_9am"
## [19] "cloud_3pm" "temp_9am" "temp_3pm"
## [22] "rain_today" "risk_mm" "rain_tomorrow"
## [25] "temp_range" "excess"

d$excess <- NULL
tail(d$excess)
## NULL

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"
## [13] "wind_speed_3pm" "humidity_9am" "humidity_3pm"
## [16] "pressure_9am" "pressure_3pm" "cloud_9am"
## [19] "cloud_3pm" "temp_9am" "temp_3pm"
## [22] "rain_today" "risk_mm" "rain_tomorrow"
## [25] "temp_range" "excess"
```
11 Subset Data

Exercise: Discuss the subset function.
12 Wide to Long Data

Let’s take a sample dataset to illustrate the concepts of wide and long data.

dss <- subset(ds, date==max(date))
dim(dss)
## [1] 49 25

head(dss)
## date location min_temp max_temp rainfall evaporation
## 1883 2014-04-25 Albury 5.3 22.5 0.0 NA
## 3735 2014-04-25 BadgerysCreek 16.5 21.2 1.8 NA
## 5587 2014-04-25 Cobar 12.9 30.5 0.0 7.4
....

This data is in wide format. We can convert it to long format, which is sometimes useful when using, for example, ggplot2 (Wickham and Chang, 2014). We use reshape2 (Wickham, 2014b) to do this. In long format we essentially maintain a single measurement per observation. The measurement for our data are all those columns recording some measure of the weather—that is, all variables except for date and location.

library(reshape2)
dssm <- melt(dss, c("date", "location"))
## Warning: attributes are not identical across measure variables; they will be dropped
dim(dssm)
## [1] 1127 4

head(dssm)
## date location variable value
## 1 2014-04-25 Albury min_temp 5.3
## 2 2014-04-25 BadgerysCreek min_temp 16.5
## 3 2014-04-25 Cobar min_temp 12.9
....

tail(dssm)
## date location variable value
## 1122 2014-04-25 Hobart temp_range 9.1
## 1123 2014-04-25 Launceston temp_range 17.8
## 1124 2014-04-25 AliceSprings temp_range 19.3
....
dssm[sample(nrow(dssm), 6),]
## date location variable value
## 415 2014-04-25 Melbourne wind_dir_3pm N
## 672 2014-04-25 Nuriootpa pressure_9am 1017.7
## 166 2014-04-25 Ballarat evaporation <NA>
This is now clearly long data.
13 Long to Wide Data

dssmc <- dcast(dssm, date + location ~ variable)

## [1] 49 25

dim(dssmc)
## [1] 49 25

head(dss)
## date location min_temp max_temp rainfall evaporation sunshine wind_gust_dir wind_gust_speed wind_dir_9am wind_dir_3pm
## 1 2014-04-25 Adelaide 9 23.3 0.2 <NA> <NA> NNW 39 NE WNW 13
## 2 2014-04-25 Albany <NA> 21 <NA> 2.2 4.4 <NA> <NA> <NA> SSW <NA>
## 3 2014-04-25 CoffsHarbour 14.4 25.0 0.0 3.0 <NA> <NA> <NA> <NA> <NA>
## 4 2014-04-25 Moree 17.7 30.0 0.4 NA <NA> <NA> <NA> <NA> <NA>
## 5 2014-04-25 Newcastle 11.0 21.2 11.8 NA <NA> <NA> <NA> <NA> <NA>
## 6 2014-04-25 Albury 5.3 22.5 0.0 NA <NA> <NA> NNW 39 NE WNW 13

head(dssmc)
## date location min_temp max_temp rainfall evaporation sunshine wind_gust_dir wind_gust_speed wind_dir_9am wind_dir_3pm
## 1 2014-04-25 Adelaide 9 23.3 0.2 <NA> <NA> NNW 39 NE WNW 13
## 2 2014-04-25 Albany <NA> 21 <NA> 2.2 4.4 <NA> <NA> <NA> SSW <NA>
## 3 2014-04-25 CoffsHarbour 14.4 25.0 0.0 3.0 <NA> <NA> <NA> <NA> <NA>
## 4 2014-04-25 Moree 17.7 30.0 0.4 NA <NA> <NA> <NA> <NA> <NA>
## 5 2014-04-25 Newcastle 11.0 21.2 11.8 NA <NA> <NA> <NA> <NA> <NA>
## 6 2014-04-25 Albury 5.3 22.5 0.0 NA <NA> <NA> NNW 39 NE WNW 13

....

14 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.
15 References


