Data Preparation Template
The **weatherAUS** Dataset

Graham Williams

2 July 2018

This template provides a guide to the typical sequence of steps a data scientist begins with in analyzing a new dataset. See The Essentials of Data Science (2017) for details. The sequence can be extracted into separate files, corresponding to the logical steps along the way. Each can be run sequentially. Alternatively you can maintain a single template file with steps.

The actual R templates are available from https://essentials.togaware.com. In particular the various chapters in https://onepager.togaware.com delve into much more detail and offer many alternative processing options.

We collect here the packages used throughout this document.

```
# Load required packages from local library into R.

library(tidyverse)    # ggplot2, tibble, tidyr, readr, purr, dplyr
library(rattle)       # comcat(), weatherAUS, normVarNames().
library(magrittr)     # Pipe operator %>% %<>% %T>% equals().
library(lubridate)    # Dates and time.
library(stringi)      # String concat operator %s+%.
library(stringr)      # String manipulation: str_replace().
library(randomForest) # Impute missing values with na.roughfix()
library(FSelector)    # Feature selection: information.gain().
library(scales)       # Include commas in numbers.
library(xtable)       # Generate LaTeX tables.
```

1 Data Sources

Our first step is to identify the data sources and to discuss access of the data, including restrictions, with the data owners. This should be documented in our tool of choice, perhaps using knitr or notebooks. Documentation should include the data sources, integrity, providence, and dates of collection and access.

The dataset we use is the *weatherAUS* dataset containing Australian weather observations. It covers observations from weather stations around Australia. The data is scraped from the Australian Bureau of Meteorology[1] and has been collected daily since 2007.

Below we identify that the source dataset is located in `data/.` from our local folder. The dataset is also directly available as a sample dataset with the `rattle.data` package in R, and also from the Internet.

```
# Name of the dataset.

dsname <- "weatherAUS"

# Identify the source location of the dataset.

dsloc <- "data"

# Alternatives might include:
#
# dsloc <- "C:/Users/graham/Projects"
# dsloc <- "~/projects"
# dsloc <- "http://rattle.togaware.com"

# Construct the path to the dataset and display some if it.

dsname %s+% ".csv" %>%
  file.path(dsloc, .) %T>%
  cat("Dataset:", ., "\n\n") %T>%
  {
     paste("head", .) %>%
     system(intern=TRUE) %>%
     sub("\r", "\n", .) %>%
     print()
  } ->
dspath
```

```
## Dataset: data/weatherAUS.csv
##
##  [1] "\"Date\",\"Location\",\"MinTemp\",\"MaxTemp\",\"Rainfall\",\"Evapora...
##  [2] "2008-12-01,\"Albury\",13.4,22.9,0.6,NA,NA,\"W\",44,\"W\",\"WNW\",20,...
##  [3] "2008-12-02,\"Albury\",7.4,25.1,0,NA,NA,\"WNW\",44,\"NNW\",\"WSW\",4,...
##  [4] "2008-12-03,\"Albury\",12.9,25.7,0,NA,NA,\"WSW\",46,\"W\",\"WSW\",19,...
....
```

[1] http://www.bom.gov.au/climate/dwo/

2 Data Ingestion

Next we ingest the data into R and identify any particular issues with doing so. A function appropriate to the source dataset format (`read_csv()` in this case) is used to ingest the data. We then store the dataset into a variable naming the dataset as a variable named `weatherAUS`.

```
# Ingest the dataset.

dspath %>%
  read_csv() ->
weatherAUS

weatherAUS
```

```
## # A tibble: 145,460 x 24
##    Date       Location MinTemp MaxTemp Rainfall Evaporation Sunshine
##    <date>     <chr>      <dbl>   <dbl>    <dbl> <chr>       <chr>
##  1 2008-12-01 Albury      13.4    22.9      0.6 <NA>        <NA>
##  2 2008-12-02 Albury       7.4    25.1      0   <NA>        <NA>
##  3 2008-12-03 Albury      12.9    25.7      0   <NA>        <NA>
##  4 2008-12-04 Albury       9.2    28        0   <NA>        <NA>
##  5 2008-12-05 Albury      17.5    32.3      1   <NA>        <NA>
##  6 2008-12-06 Albury      14.6    29.7      0.2 <NA>        <NA>
##  7 2008-12-07 Albury      14.3    25        0   <NA>        <NA>
##  8 2008-12-08 Albury       7.7    26.7      0   <NA>        <NA>
##  9 2008-12-09 Albury       9.7    31.9      0   <NA>        <NA>
## 10 2008-12-10 Albury      13.1    30.1      1.4 <NA>        <NA>
## # ... with 145,450 more rows, and 17 more variables: WindGustDir <chr>,
## #   WindGustSpeed <int>, WindDir9am <chr>, WindDir3pm <chr>,
## #   WindSpeed9am <int>, WindSpeed3pm <int>, Humidity9am <int>,
## #   Humidity3pm <int>, Pressure9am <dbl>, Pressure3pm <dbl>, Cloud9am <int>,
## #   Cloud3pm <int>, Temp9am <dbl>, Temp3pm <dbl>, RainToday <chr>,
## #   RISK_MM <dbl>, RainTomorrow <chr>
```

3 Generic Template Variables

A generic variable (`ds`) will be used to reference the storage in memory of this dataset. All of our following activity will then refer to this generic variable rather than the name of the original dataset.

```
# Prepare the dataset for usage with our template.

ds <- get(dsname)
```

We keep the original dataset available so that we can restart our wrangling process as data processing mistakes are made, as they likely will be. That way we do not need to download the source dataset again. This is good practise even though in our case here it takes only a few seconds to download (depending on our Internet connection) and little time to parse to data.

Below we store the original dataset as an R Data file on disk and then remove it to free up memory for processing. Saving to disk for this dataset takes just a second or so.

```
# Save the dataset to disk as a binary R Data for backup.

fpath <- "data"

dsname %s+% ".RData" %>%
  file.path(fpath, .) %T>%
  cat("Saving:", ., "\n\n") ->
fname
```

```
## Saving: data/weatherAUS.RData
```

```
if (! dir.exists(fpath))  dir.create(fpath)
if (! file.exists(fname)) save(weatherAUS, file=fname)

# Remove the original dataset to save on memory.

rm(weatherAUS)
```

Later, we can more quickly load the dataset. This small sample dataset takes a fraction of a second generally to load, compared to the several seconds to download and parse from the Internet.

```
# Test the loading of the saved dataset.

load(fname) %>% print()
```

```
## [1] "weatherAUS"
```

The result returned by `load()` is printed as the name of the object(s) loaded from the file. That worked, so we clean up.

```
# Cleanup to save on memory.

rm(weatherAUS)
```

4 Data Observation and Preparation

We can observe with some simple commands that the weatherAUS dataset has 145,460 observations of 24 variables.

The 24 variables observed in the dataset can be readily observed using `tibble::glimpse()`.

```
# A glimpse into the dataset.

glimpse(ds)
```

```
## Observations: 145,460
## Variables: 24
## $ Date          <date> 2008-12-01, 2008-12-02, 2008-12-03, 2008-12-04, 200...
## $ Location      <chr> "Albury", "Albury", "Albury", "Albury", "Albury", "A...
## $ MinTemp       <dbl> 13.4, 7.4, 12.9, 9.2, 17.5, 14.6, 14.3, 7.7, 9.7, 13...
## $ MaxTemp       <dbl> 22.9, 25.1, 25.7, 28.0, 32.3, 29.7, 25.0, 26.7, 31.9...
## $ Rainfall      <dbl> 0.6, 0.0, 0.0, 0.0, 1.0, 0.2, 0.0, 0.0, 0.0, 1.4, 0....
## $ Evaporation   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ Sunshine      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ WindGustDir   <chr> "W", "WNW", "WSW", "NE", "W", "WNW", "W", "W", "NNW"...
## $ WindGustSpeed <int> 44, 44, 46, 24, 41, 56, 50, 35, 80, 28, 30, 31, 61, ...
## $ WindDir9am    <chr> "W", "NNW", "W", "SE", "ENE", "W", "SW", "SSE", "SE"...
## $ WindDir3pm    <chr> "WNW", "WSW", "WSW", "E", "NW", "W", "W", "W", "NW",...
## $ WindSpeed9am  <int> 20, 4, 19, 11, 7, 19, 20, 6, 7, 15, 17, 15, 28, 24, ...
## $ WindSpeed3pm  <int> 24, 22, 26, 9, 20, 24, 24, 17, 28, 11, 6, 13, 28, 20...
## $ Humidity9am   <int> 71, 44, 38, 45, 82, 55, 49, 48, 42, 58, 48, 89, 76, ...
## $ Humidity3pm   <int> 22, 25, 30, 16, 33, 23, 19, 19, 9, 27, 22, 91, 93, 4...
## $ Pressure9am   <dbl> 1007.7, 1010.6, 1007.6, 1017.6, 1010.8, 1009.2, 1009...
## $ Pressure3pm   <dbl> 1007.1, 1007.8, 1008.7, 1012.8, 1006.0, 1005.4, 1008...
## $ Cloud9am      <int> 8, NA, NA, NA, 7, NA, 1, NA, NA, NA, NA, 8, 8, NA, N...
## $ Cloud3pm      <int> NA, NA, 2, NA, 8, NA, NA, NA, NA, NA, NA, 8, 8, 7, N...
## $ Temp9am       <dbl> 16.9, 17.2, 21.0, 18.1, 17.8, 20.6, 18.1, 16.3, 18.3...
## $ Temp3pm       <dbl> 21.8, 24.3, 23.2, 26.5, 29.7, 28.9, 24.6, 25.5, 30.2...
## $ RainToday     <chr> "No", "No", "No", "No", "No", "No", "No", "No", "No"...
## $ RISK_MM       <dbl> 0.0, 0.0, 0.0, 1.0, 0.2, 0.0, 0.0, 0.0, 1.4, 0.0, 2....
## $ RainTomorrow  <chr> "No", "No", "No", "No", "No", "No", "No", "No", "Yes...
```

5 Normalise Variable Names

We first note that the variable names do not conform to our standards and so we change them to suit. The advantage is that we then have a well defined and consistent naming scheme. The disadvantage is that the names now differ from the original, though the mapping is simple enough and we capture the mapping within `vnames`.

```
# Review the variables to optionally normalise their names.

names(ds)
```

```
##  [1] "Date"          "Location"      "MinTemp"       "MaxTemp"
##  [5] "Rainfall"      "Evaporation"   "Sunshine"      "WindGustDir"
##  [9] "WindGustSpeed" "WindDir9am"    "WindDir3pm"    "WindSpeed9am"
## [13] "WindSpeed3pm"  "Humidity9am"   "Humidity3pm"   "Pressure9am"
## [17] "Pressure3pm"   "Cloud9am"      "Cloud3pm"      "Temp9am"
## [21] "Temp3pm"       "RainToday"     "RISK_MM"       "RainTomorrow"
```

```
# Capture the original variable names for use later on.

vnames <- names(ds)

# Normalise the variable names.

names(ds) %<>% normVarNames() %T>% print()
```

```
##  [1] "date"            "location"        "min_temp"        "max_temp"
##  [5] "rainfall"        "evaporation"     "sunshine"        "wind_gust_dir"
##  [9] "wind_gust_speed" "wind_dir_9am"    "wind_dir_3pm"    "wind_speed_9am"
## [13] "wind_speed_3pm"  "humidity_9am"    "humidity_3pm"    "pressure_9am"
## [17] "pressure_3pm"    "cloud_9am"       "cloud_3pm"       "temp_9am"
## [21] "temp_3pm"        "rain_today"      "risk_mm"         "rain_tomorrow"
```

```
# Index the original variable names by the new names.

names(vnames) <- names(ds)

# We can then obtain the original variable name.

cat(vnames["min_temp"])
```

```
## MinTemp
```

6 Directly Normalise on Reading

If we are reading the data from a CSV file then we can include the process within a pipeline, converting the column names, and saving the result into the dataset named variable.

```
dspath %>%
  read_csv() %>%
  set_names(names(.) %>% normVarNames()) ->
weatherAUS
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Date = col_date(format = ""),
##   MinTemp = col_double(),
##   MaxTemp = col_double(),
##   Rainfall = col_double(),
##   WindGustSpeed = col_integer(),
##   WindSpeed9am = col_integer(),
##   WindSpeed3pm = col_integer(),
##   Humidity9am = col_integer(),
##   Humidity3pm = col_integer(),
##   Pressure9am = col_double(),
##   Pressure3pm = col_double(),
##   Cloud9am = col_integer(),
##   Cloud3pm = col_integer(),
##   Temp9am = col_double(),
##   Temp3pm = col_double(),
##   RISK_MM = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
weatherAUS
```

```
## # A tibble: 145,460 x 24
##    date       location min_temp max_temp rainfall evaporation sunshine
##    <date>     <chr>       <dbl>    <dbl>    <dbl> <chr>       <chr>
##  1 2008-12-01 Albury       13.4     22.9      0.6 <NA>        <NA>
##  2 2008-12-02 Albury        7.4     25.1      0   <NA>        <NA>
##  3 2008-12-03 Albury       12.9     25.7      0   <NA>        <NA>
....
```

7 Observations of the Data

We next observe the top, the bottom and a random sample of data. Often this will be instructive as it is here. We note that `evaporation` and `sunshine` are often but not always missing.

```
# Glimpse the dataset.

glimpse(ds)
```

```
## Observations: 145,460
## Variables: 24
## $ date            <date> 2008-12-01, 2008-12-02, 2008-12-03, 2008-12-04, 2...
## $ location        <chr> "Albury", "Albury", "Albury", "Albury", "Albury", ...
## $ min_temp        <dbl> 13.4, 7.4, 12.9, 9.2, 17.5, 14.6, 14.3, 7.7, 9.7, ...
## $ max_temp        <dbl> 22.9, 25.1, 25.7, 28.0, 32.3, 29.7, 25.0, 26.7, 31...
....
```

```
# Review the first few observations.

head(ds) %>% print.data.frame()
```

```
##         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>
## 4 2008-12-04   Albury      9.2     28.0      0.0        <NA>     <NA>
## 5 2008-12-05   Albury     17.5     32.3      1.0        <NA>     <NA>
....
```

```
# Review the last few observations.

tail(ds) %>% print.data.frame()
```

```
##         date location min_temp max_temp rainfall evaporation sunshine
## 1 2017-06-20    Uluru      3.5     21.8        0        <NA>     <NA>
## 2 2017-06-21    Uluru      2.8     23.4        0        <NA>     <NA>
## 3 2017-06-22    Uluru      3.6     25.3        0        <NA>     <NA>
## 4 2017-06-23    Uluru      5.4     26.9        0        <NA>     <NA>
## 5 2017-06-24    Uluru      7.8     27.0        0        <NA>     <NA>
....
```

```
# Review a random sample of observations.

sample_n(ds, size=6) %>% print.data.frame()
```

```
##         date     location min_temp max_temp rainfall evaporation sunshine
## 1 2009-01-06   Launceston     16.0     23.8      0.0         7.2     <NA>
## 2 2009-07-24 AliceSprings      1.6     23.2      0.0         5.4     10.5
## 3 2014-12-13  Williamtown     16.2     23.2       NA        <NA>     <NA>
## 4 2008-12-04        Perth     17.0     24.2      0.8         5.8      7.8
## 5 2009-02-04   Townsville     23.0     29.6     93.4        <NA>      0.4
....
```

8 Data Summary

```
# Traditional dataset summary to get started.

summary(ds)
```

```
##       date               location            min_temp        max_temp
##  Min.   :2007-11-01   Length:145460      Min.   :-8.50   Min.   :-4.80
##  1st Qu.:2011-01-11   Class :character   1st Qu.: 7.60   1st Qu.:17.90
##  Median :2013-06-02   Mode  :character   Median :12.00   Median :22.60
##  Mean   :2013-04-04                      Mean   :12.19   Mean   :23.22
##  3rd Qu.:2015-06-14                      3rd Qu.:16.90   3rd Qu.:28.20
##  Max.   :2017-06-25                      Max.   :33.90   Max.   :48.10
##                                          NA's   :1485    NA's   :1261
##     rainfall       evaporation          sunshine         wind_gust_dir
##  Min.   :  0.000   Length:145460      Length:145460      Length:145460
##  1st Qu.:  0.000   Class :character   Class :character   Class :character
##  Median :  0.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :  2.361
##  3rd Qu.:  0.800
##  Max.   :371.000
##  NA's   :3261
##  wind_gust_speed  wind_dir_9am       wind_dir_3pm       wind_speed_9am
##  Min.   :  6.00   Length:145460      Length:145460      Min.   :  0.00
##  1st Qu.: 31.00   Class :character   Class :character   1st Qu.:  7.00
##  Median : 39.00   Mode  :character   Mode  :character   Median : 13.00
##  Mean   : 40.03                                         Mean   : 14.04
##  3rd Qu.: 48.00                                         3rd Qu.: 19.00
##  Max.   :135.00                                         Max.   :130.00
##  NA's   :10263                                          NA's   :1767
##  wind_speed_3pm   humidity_9am     humidity_3pm     pressure_9am
##  Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 980.5
##  1st Qu.:13.00   1st Qu.: 57.00   1st Qu.: 37.00   1st Qu.:1012.9
##  Median :19.00   Median : 70.00   Median : 52.00   Median :1017.6
##  Mean   :18.66   Mean   : 68.88   Mean   : 51.54   Mean   :1017.6
##  3rd Qu.:24.00   3rd Qu.: 83.00   3rd Qu.: 66.00   3rd Qu.:1022.4
##  Max.   :87.00   Max.   :100.00   Max.   :100.00   Max.   :1041.0
##  NA's   :3062    NA's   :2654     NA's   :4507     NA's   :15065
##   pressure_3pm      cloud_9am       cloud_3pm        temp_9am
##  Min.   : 977.1   Min.   :0.00    Min.   :0.00    Min.   :-7.20
##  1st Qu.:1010.4   1st Qu.:1.00    1st Qu.:2.00    1st Qu.:12.30
##  Median :1015.2   Median :5.00    Median :5.00    Median :16.70
##  Mean   :1015.3   Mean   :4.45    Mean   :4.51    Mean   :16.99
##  3rd Qu.:1020.0   3rd Qu.:7.00    3rd Qu.:7.00    3rd Qu.:21.60
##  Max.   :1039.6   Max.   :9.00    Max.   :9.00    Max.   :40.20
##  NA's   :15028    NA's   :55888   NA's   :59358   NA's   :1767
....
```

9 Dealing With Dates

The tidyverse functions recognise date formats on ingesting the data. However if we find that we have dates represented using character strings then it is relatively simple to convert them to the Date date type. Here we convert the ingested date variable to a character string and then back to a date format

```
# Date data type conversion (if required). Set the appropriate date format.

head(ds$date) %>% as.character() %>% class()

## [1] "character"

ds$date %<>% as.character() %>% ymd() %>% as.Date()
class(ds$date)

## [1] "Date"

head(ds$date)

## [1] "2008-12-01" "2008-12-02" "2008-12-03" "2008-12-04" "2008-12-05"
## [6] "2008-12-06"
```

10 Dealing with Location

```
# How many locations are represented in the dataset.

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

```
## [1] 49
```

```
# Review the distribution of observations across levels.

ds %>%
  select(starts_with("rain_")) %>%
  sapply(table)
```

```
##     rain_today rain_tomorrow
## No      110319        110316
## Yes      31880         31877
```

```
# Note the  names of the rain variables.

ds %>%
  select(starts_with("rain_")) %>%
  names() %T>%
  print() ->
vrain
```

```
## [1] "rain_today"    "rain_tomorrow"
```

```
# Confirm these are currently character variables.

ds[vrain] %>% sapply(class)
```

```
##    rain_today rain_tomorrow
##   "character"   "character"
```

```
# Choose to convert these variables from character to factor.

ds[vrain] %<>%
  lapply(factor) %>%
  data.frame() %>%
  tbl_df()

# Confirm they are now factors.

ds[vrain] %>% sapply(class)
```

```
##    rain_today rain_tomorrow
##      "factor"      "factor"
```

```
# Verify the distribution has not changed.

ds %>%
```

```
  select(starts_with("rain_")) %>%
  sapply(table)
```

```
##     rain_today rain_tomorrow
## No      110319        110316
## Yes      31880         31877
```

```
ds %>%
  select(contains("_dir")) %>%
  names() %>%
  paste(collapse="|, \\verb|") %>%
  paste0("\\verb|", . , "|") %>%
  str_replace(", (\\\\\\verb[^,]+)$", ", and \\1") ->
wgvars

# Review the distribution of observations across levels.

ds %>%
  select(contains("_dir")) %>%
  sapply(table)
```

```
##     wind_gust_dir wind_dir_9am wind_dir_3pm
## E            9181         9176         8472
## ENE          8104         7836         7857
## ESE          7372         7630         8505
## N            9313        11758         8890
## NE           7133         7671         8263
....
```

```
# Levels of wind direction are ordered compas directions.

compass <- c("N", "NNE", "NE", "ENE",
             "E", "ESE", "SE", "SSE",
             "S", "SSW", "SW", "WSW",
             "W", "WNW", "NW", "NNW")

# Note the names of the wind direction variables.

ds %>%
  select(contains("_dir")) %>%
  names() %T>%
  print() ->
vdir
```

```
## [1] "wind_gust_dir" "wind_dir_9am"  "wind_dir_3pm"
```

```
# Confirm these are currently character variables.

ds[vdir] %>% sapply(class)
```

```
## wind_gust_dir  wind_dir_9am  wind_dir_3pm
##   "character"   "character"   "character"
```

```
# Convert these variables from character to factor.

ds[vdir] %<>%
  lapply(factor, levels=compass, ordered=TRUE) %>%
  data.frame() %>%
  tbl_df()

# Confirm they are now factors.

ds[vdir] %>% sapply(class)
```

```
##      wind_gust_dir wind_dir_9am wind_dir_3pm
## [1,] "ordered"     "ordered"    "ordered"
## [2,] "factor"      "factor"     "factor"
```

```
# Verify the distribution has not changed.

ds %>%
  select(contains("_dir")) %>%
  sapply(table)
```

```
##     wind_gust_dir wind_dir_9am wind_dir_3pm
## N            9313        11758         8890
## NNE          6548         8129         6590
## NE           7133         7671         8263
## ENE          8104         7836         7857
## E            9181         9176         8472
....
```

```
# Note the character remaining variables to be dealt with.

cvars <- c("evaporation", "sunshine")

# Review the values.

head(ds[cvars])
```

```
## # A tibble: 6 x 2
##   evaporation sunshine
##   <chr>       <chr>
## 1 <NA>        <NA>
## 2 <NA>        <NA>
## 3 <NA>        <NA>
....
```

```
tail(ds[cvars])
```

```
## # A tibble: 6 x 2
##   evaporation sunshine
##   <chr>       <chr>
## 1 <NA>        <NA>
## 2 <NA>        <NA>
```

```
## 3 <NA>        <NA>
....
```

```
sample_n(ds[cvars], 6)
```

```
## # A tibble: 6 x 2
##   evaporation sunshine
##   <chr>       <chr>
## 1 3.8         8.5
## 2 <NA>        <NA>
## 3 4           0.2
....
```

```
# Check the current class of the variables.

ds[cvars] %>% sapply(class)
```

```
## evaporation    sunshine
## "character" "character"
```

```
# Convert to numeric.

ds[cvars] %<>% sapply(as.numeric)

# Confirm the conversion.

ds[cvars] %>% sapply(class)
```

```
## evaporation    sunshine
##   "numeric"   "numeric"
```

```
## Identifiers and Targets ----------------

# Note the available variables.

ds %>%
  names() %T>%
  print() ->
vars
```

```
##  [1] "date"           "location"       "min_temp"       "max_temp"
##  [5] "rainfall"       "evaporation"    "sunshine"       "wind_gust_dir"
##  [9] "wind_gust_speed" "wind_dir_9am"   "wind_dir_3pm"   "wind_speed_9am"
## [13] "wind_speed_3pm" "humidity_9am"   "humidity_3pm"   "pressure_9am"
## [17] "pressure_3pm"   "cloud_9am"      "cloud_3pm"      "temp_9am"
## [21] "temp_3pm"       "rain_today"     "risk_mm"        "rain_tomorrow"
....
```

```
# Note the target variable.

target <- "rain_tomorrow"

# Place the target variable at the beginning of the vars.
```

```
c(target, vars) %>%
  unique() %T>%
  print() ->
vars
```

```
##  [1] "rain_tomorrow"   "date"            "location"        "min_temp"
##  [5] "max_temp"        "rainfall"        "evaporation"     "sunshine"
##  [9] "wind_gust_dir"   "wind_gust_speed" "wind_dir_9am"    "wind_dir_3pm"
## [13] "wind_speed_9am"  "wind_speed_3pm"  "humidity_9am"    "humidity_3pm"
## [17] "pressure_9am"    "pressure_3pm"    "cloud_9am"       "cloud_3pm"
## [21] "temp_9am"        "temp_3pm"        "rain_today"      "risk_mm"
....
```

```
# Note the risk variable - measures the severity of the outcome.

risk <- "risk_mm"

# Note any identifiers.

id <- c("date", "location")

## Generic Data Wrangling ----------------

# Initialise ignored variables: identifiers and risk.

ignore <- union(id, if (exists("risk")) risk) %T>% print()
```

```
## [1] "date"     "location" "risk_mm"
```

```
# Heuristic for indentifiers to possibly ignore.

ds[vars] %>%
  sapply(function(x) x %>% unique() %>% length()) %>%
  equals(nrow(ds)) %>%
  which() %>%
  names() %T>%
  print() ->
ids
```

```
## character(0)
```

```
# Add them if any to the variables to be ignored for modelling.

ignore <- union(ignore, ids) %T>% print()
```

```
## [1] "date"     "location" "risk_mm"
```

```
# Identify variables with only missing values.

ds[vars] %>%
  sapply(function(x) x %>% is.na %>% sum) %>%
  equals(nrow(ds)) %>%
```

```
  which() %>%
  names() %T>%
  print() ->
missing
```

```
## character(0)
```

```
# Add them if any to the variables to be ignored for modelling.

ignore <- union(ignore, missing) %T>% print()
```

```
## [1] "date"     "location" "risk_mm"
```

```
# Identify a threshold above which proportion missing is fatal.

missing.threshold <- 0.7

# Identify variables that are mostly missing.

ds[vars] %>%
  sapply(function(x) x %>% is.na() %>% sum()) %>%
  '>'(missing.threshold*nrow(ds)) %>%
  which() %>%
  names() %T>%
  print() ->
mostly
```

```
## character(0)
```

```
# Add them if any to the variables to be ignored for modelling.

ignore <- union(ignore, mostly) %T>% print()
```

```
## [1] "date"     "location" "risk_mm"
```

```
# Identify a threshold above which we have too many levels.

levels.threshold <- 20

# Identify variables that have too many levels.

ds[vars] %>%
  sapply(is.factor) %>%
  which() %>%
  names() %>%
  sapply(function(x) ds %>% extract2(x) %>% levels() %>% length()) %>%
  '>='(levels.threshold) %>%
  which() %>%
  names() %T>%
  print() ->
too.many
```

```
## character(0)
```

```
# Add them if any to the variables to be ignored for modelling.

ignore <- union(ignore, too.many) %T>% print()

## [1] "date"     "location" "risk_mm"

# Identify variables that have a single value.

ds[vars] %>%
  sapply(function(x) all(x == x[1L])) %>%
  which() %>%
  names() %T>%
  print() ->
constants

## character(0)

# Add them if any to the variables to be ignored for modelling.

ignore <- union(ignore, constants) %T>% print()

## [1] "date"     "location" "risk_mm"

# Note which variables are numeric.

vars %>%
  setdiff(ignore) %>%
  '['(ds, .) %>%
  sapply(is.numeric) %>%
  which() %>%
  names() %T>%
  print() ->
numc

##  [1] "min_temp"       "max_temp"       "rainfall"       "evaporation"
##  [5] "sunshine"       "wind_gust_speed" "wind_speed_9am"  "wind_speed_3pm"
##  [9] "humidity_9am"   "humidity_3pm"   "pressure_9am"   "pressure_3pm"
## [13] "cloud_9am"      "cloud_3pm"      "temp_9am"       "temp_3pm"

# For the numeric variables generate a table of correlations

ds[numc] %>%
  cor(use="complete.obs") %>%
  ifelse(upper.tri(., diag=TRUE), NA, .) %>%
  abs %>%
  data.frame %>%
  tbl_df %>%
  set_colnames(numc) %>%
  mutate(var1=numc) %>%
  gather(var2, cor, -var1) %>%
  na.omit %>%
  arrange(-abs(cor)) %T>%
```

```
  print() ->
mc
```

```
## # A tibble: 120 x 3
##    var1           var2              cor
##    <chr>          <chr>           <dbl>
##  1 temp_3pm       max_temp        0.985
##  2 pressure_3pm   pressure_9am    0.962
##  3 temp_9am       min_temp        0.907
....
```

```
# Any variables could be removed because highly correlated?

correlated <- c("temp_3pm", "pressure_3pm", "temp_9am")

# Add them if any to the variables to be ignored for modelling.

ignore <- union(ignore, correlated) %T>% print()
```

```
## [1] "date"         "location"     "risk_mm"      "temp_3pm"     "pressure_...
## [6] "temp_9am"
```

```
# Check the number of variables currently.

length(vars)
```

```
## [1] 24
```

```
# Remove the variables to ignore.

vars <- setdiff(vars, ignore) %T>% print()
```

```
##  [1] "rain_tomorrow"   "min_temp"        "max_temp"        "rainfall"
##  [5] "evaporation"     "sunshine"        "wind_gust_dir"   "wind_gust_speed"
##  [9] "wind_dir_9am"    "wind_dir_3pm"    "wind_speed_9am"  "wind_speed_3pm"
## [13] "humidity_9am"    "humidity_3pm"    "pressure_9am"    "cloud_9am"
## [17] "cloud_3pm"       "rain_today"
```

```
# Confirm they are now ignored.

length(vars)
```

```
## [1] 18
```

```
## Variable Selection ----------------

# Formula for modelling.

form <- formula(target %s+% " ~ .") %T>% print()
```

```
## rain_tomorrow ~ .
```

```
# Use correlation search to identify key variables.
# Could be useful to decide which variables to retain.
```

```
cfs(form, ds[vars])
```

```
## [1] "rainfall"     "sunshine"     "humidity_3pm" "cloud_3pm"    "rain_today"
```

```
# Any variables to remove because not useful?

vars %<>% setdiff(NULL) %T>% print()
```

```
##  [1] "rain_tomorrow"   "min_temp"        "max_temp"        "rainfall"
##  [5] "evaporation"     "sunshine"        "wind_gust_dir"   "wind_gust_speed"
##  [9] "wind_dir_9am"    "wind_dir_3pm"    "wind_speed_9am"  "wind_speed_3pm"
## [13] "humidity_9am"    "humidity_3pm"    "pressure_9am"    "cloud_9am"
## [17] "cloud_3pm"       "rain_today"
```

```
# Use information gain to identify variable importance.

information.gain(form, ds[vars]) %>%
  rownames_to_column() %>%
  arrange(attr_importance)
```

```
##            rowname attr_importance
## 1   wind_speed_9am     0.003982579
## 2     wind_dir_3pm     0.004698282
## 3   wind_speed_3pm     0.005159260
## 4         min_temp     0.005397631
## 5      evaporation     0.005694709
....
```

```
# Any variables to remove because not useful?

vars %<>% setdiff(NULL) %T>% print()
```

```
##  [1] "rain_tomorrow"   "min_temp"        "max_temp"        "rainfall"
##  [5] "evaporation"     "sunshine"        "wind_gust_dir"   "wind_gust_speed"
##  [9] "wind_dir_9am"    "wind_dir_3pm"    "wind_speed_9am"  "wind_speed_3pm"
## [13] "humidity_9am"    "humidity_3pm"    "pressure_9am"    "cloud_9am"
## [17] "cloud_3pm"       "rain_today"
```

```
## Further Wrangling ----------------

# Check the dimensions to start with.

dim(ds) %>% comcat()
```

```
## 145,460 24
```

```
# Identify observations with a missing target.

ds %>%
  extract2(target) %>%
  is.na() %T>%
  {sum(.) %>% comcat()} ->
missing.target
```

```
## 3,267

# Remove observations with a missing target.

ds %<>% filter(!missing.target)

# Confirm the filter delivered the expected dataset.

dim(ds) %>% comcat()

## 142,193 24

## Optional: Missing Value Imputation ----------------

# Count the number of missing values.

ds[vars] %>%  is.na() %>% sum() %>% comcat()

## 298,948

# Impute missing values.

ds[vars] %<>% na.roughfix()

# Confirm that no missing values remain.

ds[vars] %>%  is.na() %>% sum() %>% comcat()

## 0

## Optional: Remove Observations With Missing Values ----------------

# Initialise the list of observations to be removed.

omit <- NULL

# Review the current dataset.

ds[vars] %>% nrow() %>% comcat()

## 142,193

ds[vars] %>% is.na() %>% sum() %>% comcat()

## 0

# Identify any observations with missing values.

ds[vars] %>%
  na.omit() %>%
  attr("na.action") %T>%
  print() ->
mo
```

```
## NULL
```

```
# Record the observations to omit.
```

```
omit <- union(omit, mo) %T>% {length(.) %>% print()}
```

```
## [1] 0
```

```
# If there are observations to omit then remove them.
```

```
if (length(omit)) ds <- ds[-omit,]
```

```
# Confirm the observations have been removed.
```

```
ds[vars] %>% nrow() %>% comcat()
```

```
## 142,193
```

```
ds[vars] %>% is.na() %>% sum() %>% comcat()
```

```
## 0
```

```
## Normalise Factors ----------------
```

```
# Note which variables are categoric.
```

```
ds[vars] %>%
  sapply(is.factor) %>%
  which() %>%
  names() %T>%
  print() ->
catc
```

```
## [1] "rain_tomorrow" "wind_gust_dir" "wind_dir_9am"  "wind_dir_3pm"
## [5] "rain_today"
```

```
# Check the levels.
```

```
ds[catc] %>% sapply(levels)
```

```
## $rain_tomorrow
## [1] "No"  "Yes"
##
## $wind_gust_dir
##  [1] "N"   "NNE" "NE"  "ENE" "E"   "ESE" "SE"  "SSE" "S"   "SSW" "SW"  "WSW"
## [13] "W"   "WNW" "NW"  "NNW"
....
```

```
# Normalise the levels of all categoric variables.
```

```
for (v in catc)
  levels(ds[[v]]) %<>% normVarNames()
```

```
# Review the levels.
```

```
ds[catc] %>% sapply(levels)
```

```
## $rain_tomorrow
## [1] "no"  "yes"
##
## $wind_gust_dir
##  [1] "n"   "nne" "ne"  "ene" "e"   "ese" "se"  "sse" "s"   "ssw" "sw"  "wsw"
## [13] "w"   "wnw" "nw"  "nnw"
....
```

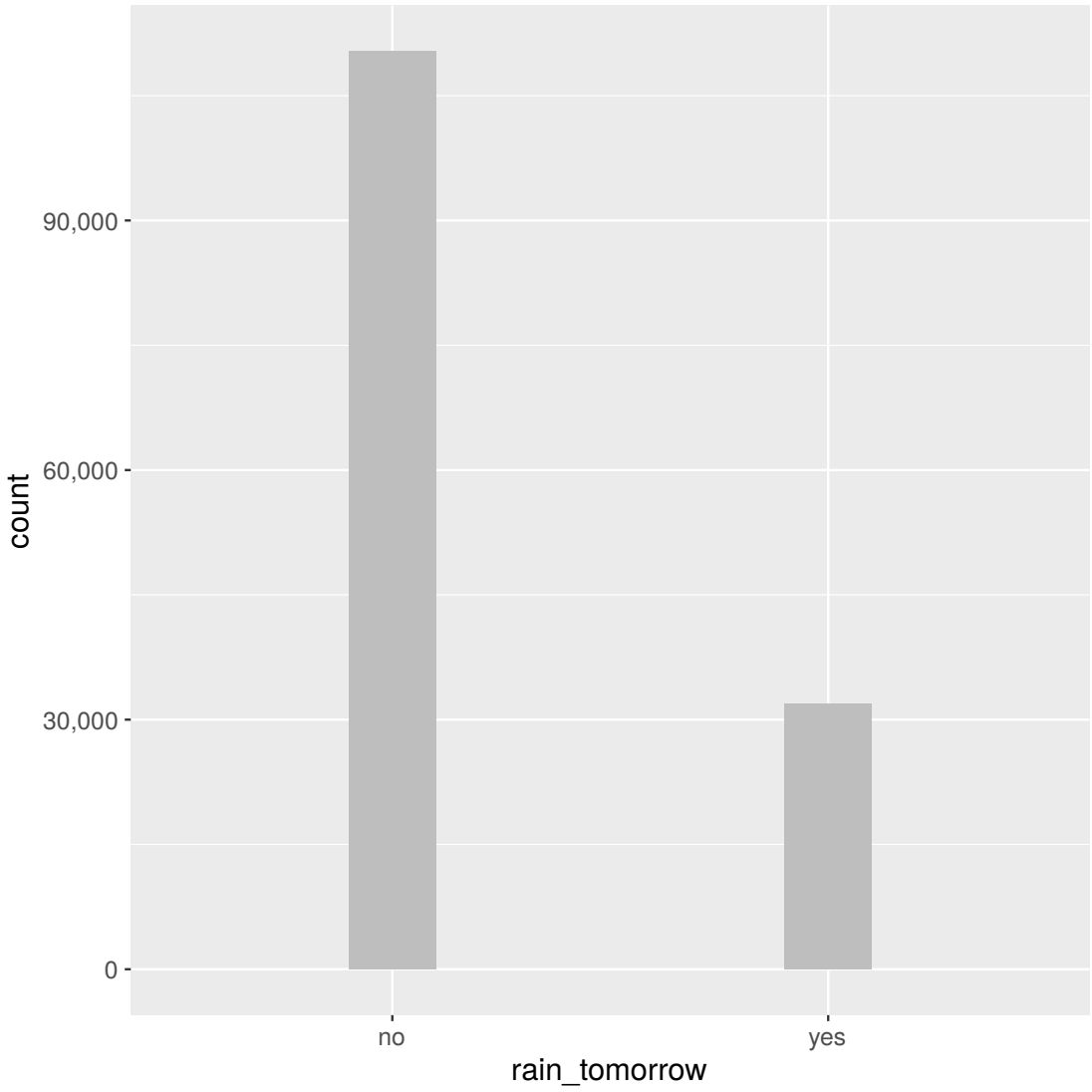
```
## Categoric Target ----------------

# Ensure the target is categoric.

class(ds[[target]])
```

```
## [1] "factor"
```

```
ds[[target]] %<>% as.factor()

# Confirm the distribution.

ds[target] %>% table()
```

```
## .
##     no    yes
## 110316  31877
```

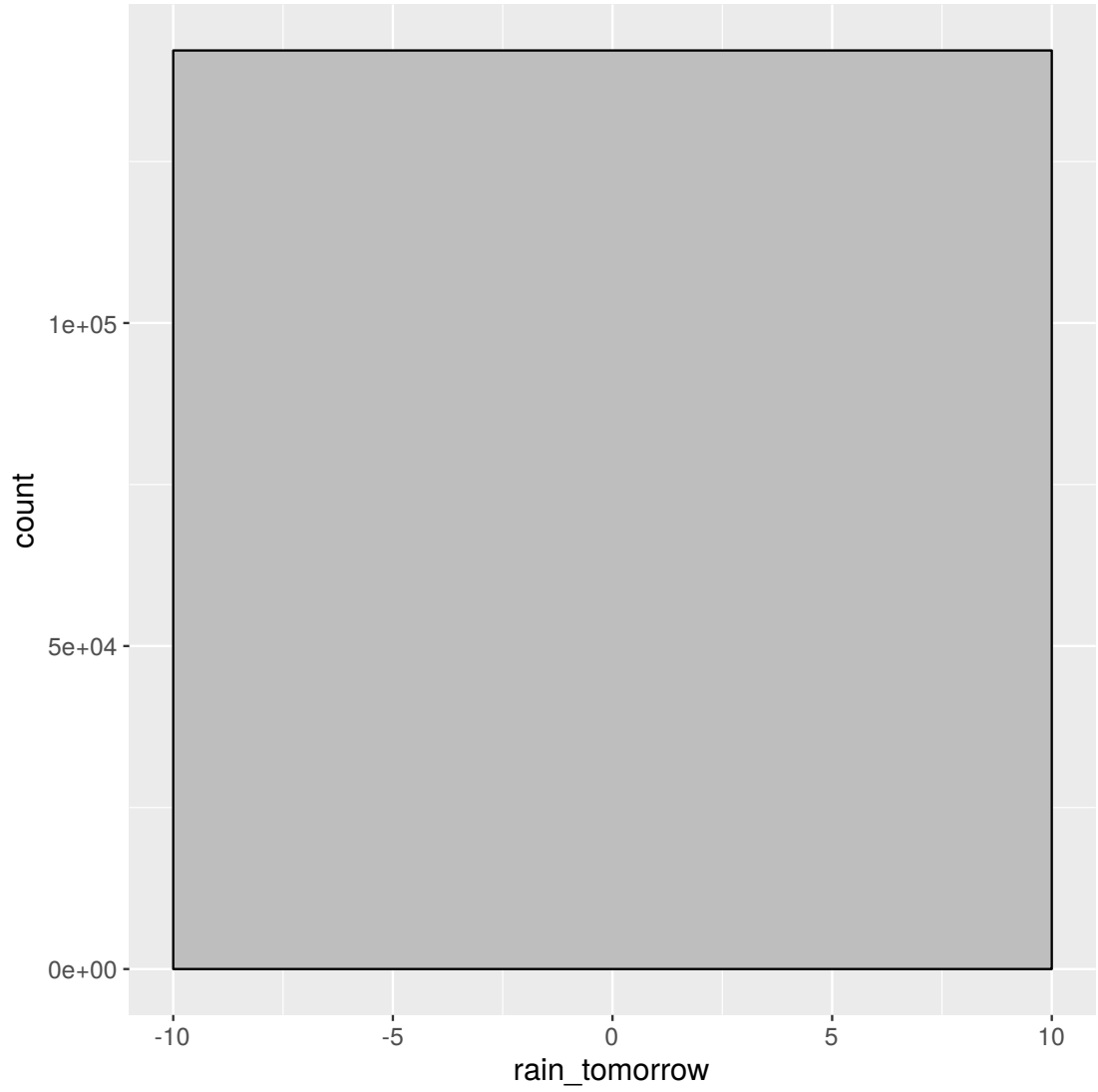
```
ds %>%
  ggplot(aes_string(x=target)) +
  geom_bar(width=0.2, fill="grey") +
  scale_y_continuous(labels=comma) +
  theme(text=element_text(size=14))
```




```
## Numeric Target - Alternative ----------------

# Ensure the target is numeric.

class(ds[[target]])
## [1] "factor"
ds[[target]] %<>% as.numeric()

# Confirm the distribution.

ds[target] %>% summary()
##  rain_tomorrow
```

```
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.224
##  3rd Qu.:1.000
....
```

```
ds %>%
  ggplot(aes_string(x=target)) +
  geom_histogram(fill="grey", col="black", binwidth=20) +
  theme(text=element_text(size=14))
```




```
#### META DATA -------------------------------
```

```
# Identify the input variables by name.

inputs <- setdiff(vars, target) %T>% print()
```

```
##  [1] "min_temp"       "max_temp"       "rainfall"       "evaporation"
##  [5] "sunshine"       "wind_gust_dir"  "wind_gust_speed" "wind_dir_9am"
##  [9] "wind_dir_3pm"   "wind_speed_9am" "wind_speed_3pm" "humidity_9am"
## [13] "humidity_3pm"   "pressure_9am"   "cloud_9am"      "cloud_3pm"
## [17] "rain_today"
```

```
# Identify the input variables by index.

inputi <- sapply(inputs,
                 function(x) which(x == names(ds)),
                 USE.NAMES=FALSE) %T>% print()
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 18 19 22
```

```
# Record the number of observations.

nobs <- nrow(ds) %T>% comcat()
```

```
## 142,193
```

```
# Confirm various subset sizes.

dim(ds)         %>% comcat()
```

```
## 142,193 24
```

```
dim(ds[vars])   %>% comcat()
```

```
## 142,193 18
```

```
dim(ds[inputs]) %>% comcat()
```

```
## 142,193 17
```

```
dim(ds[inputi]) %>% comcat()
```

```
## 142,193 17
```

```
# Identify the numeric variables by index.

ds %>%
  sapply(is.numeric) %>%
  which() %>%
  intersect(inputi) %T>%
  print() ->
numi
```

```
##  [1]  3  4  5  6  7  9 12 13 14 15 16 18 19
```

```
# Identify the numeric variables by name.

ds %>%
```

```
  names() %>%
  '['(numi) %T>%
  print() ->
numc
```

```
##  [1] "min_temp"       "max_temp"       "rainfall"       "evaporation"
##  [5] "sunshine"       "wind_gust_speed" "wind_speed_9am"  "wind_speed_3pm"
##  [9] "humidity_9am"   "humidity_3pm"   "pressure_9am"   "cloud_9am"
## [13] "cloud_3pm"
```

```
# Identify the categoric variables by index.

ds %>%
  sapply(is.factor) %>%
  which() %>%
  intersect(inputi) %T>%
  print() ->
cati
```

```
## [1]  8 10 11 22
```

```
# Identify the categoric variables by name.

ds %>%
  names() %>%
  '['(cati) %T>%
  print() ->
catc
```

```
## [1] "wind_gust_dir" "wind_dir_9am"  "wind_dir_3pm"  "rain_today"
```

```
#### SAVE THE DATASET --------------------------------

# Timestamp for the dataset - this is the general approach.

dsdate  <- "_" %s+% format(Sys.Date(), "%Y%m%d") %T>% print()
```

```
## [1] "_20180811"
```

```
# We will use a fixed timestamp to identify our file for convenience.

dsdate <- "_20180702"

# Filename for the saved dataset.

dsrdata <-
  file.path(fpath, dsname %s+% dsdate %s+% ".RData") %T>%
  print()
```

```
## [1] "data/weatherAUS_20180702.RData"
```

```
# Save relevant R objects to the binary RData file.
```

```
save(ds, dsname, dspath, dsdate, nobs,
     vars, target, risk, id, ignore, omit,
     inputi, inputs, numi, numc, cati, catc,
     file=dsrdata)
```

11 Data Exploration

We should always understand our data by exploring it in various ways. Include data summaries and various plots that give insights.