Capstone Project - DEEPAR TIMESERIES SNOW DEPTH PREDICTION

Ian Denness

14/08/2021

Required Packages - Core

```
#load the relevant libraries
library(dplyr) #basic data manipulation
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
library(ggplot2) #visualisation
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
library(tidyr) #tidying datasets
library(stringr) #string manipulation
library(lubridate) #modification of dates
library(infotheo) #equal depth and width binning
library(rvest) #webscraping
```

```
## Warning: package 'rvest' was built under R version 4.0.2
```

```
#clear the data
rm(list=ls())
```

Required Packages - Additional

Following on from some of the readings, and external links eg Datacamp, the below packages have also been used through assignment.

```r
#load the relevant libraries

library(skimr) #data interogation and inspection
```

```
## Warning: package 'skimr' was built under R version 4.0.2
```

```r
library(janitor) #data cleaning, text tidying
library(naniar) #used for visualing missing values
```

```
## Warning: package 'naniar' was built under R version 4.0.2
```

```r
library(VIM) #used for Hotdeck and kNN imputation
library(tidyverse) #includes dplyr, ggplot, stringr and others.
library(forcats) #factor modification.
library(ggpubr) #ggarrange to print multiple charts on one page, to shorten the document
library(simputation) #regression imputation, hotdeck and kNN imputation
library(missForest) #random forest imputation
```

Data

This assignment looks to download a snow depth data set to provide it for a predictive time series model (not included in this script)

Get dataset

Dataset 1

- File Source 1 sourced as a csv from data.gov.au as per the below website.

Two versions are available, this will use the 2020 dataset.

2017 Data https://data.gov.au/dataset/ds-vic-ce53c7c4-06ff-4802-9bbf-658ed6570aa6/details?q=snowfall (https://data.gov.au/dataset/ds-vic-ce53c7c4-06ff-4802-9bbf-658ed6570aa6/details?q=snowfall) 2020 Data https://arcc.vic.gov.au/wp-content/uploads/2021/06/Data-2020-Daily-Snow-Depth-Records_Falls-Creek.csv (https://arcc.vic.gov.au/wp-content/uploads/2021/06/Data-2020-Daily-Snow-Depth-Records_Falls-Creek.csv)

Import Datasets into R

To import the data the following steps where taken,

- Locate the datasource and determine how to import, i.e. what the file type is (html table, xlsx, csv etc), (for dataset 2 we are going to combine three observations as the weblink wont allow the ability to download multiple observations into one dataset.)
- For this example, webscrape directly, so copy the html link,
- Use the appropriate import function, in this case, given we have a CSV use read.csv(), (for dataset 2 the downloaded files were .zip file and required a temp download location to allow unzipping of the file.)
- Additionally use the janitor function to clean the names to remove leading zeros and inconsistencies in formating,
- Inspect the data, determine its layout,
- Preliminary data wrangling, remove redundant observations, get rid of blank data and duplicate data. For this file the top two rows contain blanks, and two columns duplicate the date, these are removed during importation,
- We now have the base data, and we can review the data types.

Dataset 1

```
#file_name <- 'https://arcc.vic.gov.au/wp-content/uploads/2021/06/Data-2020-Daily-Snow-Depth-Records_Falls-Creek.
csv'

file_name <- 'https://arcc.vic.gov.au/wp-content/uploads/2021/06/Data-2020-Daily-Snow-Depth-Records_Falls-Creek.c
sv'

#First import to review the data and make sure it comes into R.

falls_orig_df = read.csv(file_name, sep=",",
                         header = FALSE)

#Key observations are that there are empty rows, and duplicated columns.  Lets modify the read.csv line to remove
these.  Some of the columns are duplicated and have no header, lets remove these.

# We look to select just the years with natural snow depth to 2006

falls_df = read.csv(file_name, skip = 3, header = FALSE) %>%
  select(V1, V6, V9, V12, V15, V18, V21, V24, V27, V30, V33, V36, V39, V42, V45, V48)

falls_df <- falls_df[0:127,]

# Rename the column names
colnames(falls_df) <- c("date","2020","2019","2018","2017","2016","2015","2014","2013","2012","2011","2010","200
9","2008","2007","2006")
```

Understand / Tidy and Manipulate

Dataset 1 Tidy data, and type conversion.

Dataset one upon inspection has come in with the following observations

1 - Tidydata - The data is wide and needs to be changed to long. As per the above the original dataset was split into two, each one is made longer individually, with an appropriate feature engineering

2 - Combine the date fields to make one data point.

3 - Type conversion, date is incomplete and shown as a character and needs to be converted to a date. It will need the year added to it.

4 - Missing data (treated later).

The variables are hidden in the wide format but include date (as a character), excluding the year, snow type and the observation snow depth.

Steps 1 - 4 (Dataset1)

The data is still untidy and we have two wide format dataframes. We have the day (which is the date) and the year and the observation is the snow depth for each of the categorical variable. We will need to turn this into a long format, by gathering the year and snow depth. As we join the dataframes back, we can mutate a new variable in describing the datatype which gives the categorical variable.

- We will use gather to gather key columnns into groups using year and snow depth.
- After gathering, we have a longer dataframe, we will then unite the new year column to the date.
- We now have a long data frame, and the date is still a factor but in a closer format to what we would expect, with day, month and year. We will also feature engineer to show which resort we are looking at.
- We will convert the character types to factors to help with analysis at a later statge (resort and snow type.)

Make the dataset long

```
falls_long <- falls_df %>%
  gather(year, snow_depth, -date) %>%
    unite(date, date, year, sep = "-")
```

Type conversion.

```
## Date is a character, all the other class types look good (we made two of them).  Lets change the date type, an
d extract some calendar variables to be used as factors for analyis.

#Change from character to date, and glimpse again.
falls_long$date <- as.Date(falls_long$date,format = "%d-%b-%Y")

#Feature Engineering.  Now that we have converted to a date, Using date functions, lets trim these out so that we
can see if Month or Year are insightful.

falls_long <- falls_long %>%
  mutate(month = month(date),
         year = year(date))

#we now have additional categorical variables, is the time of year, or the year different.

#finally the snow season rarely runs into October, this month will be removed from the dataset.

falls_long <- falls_long %>%
  mutate(month = as.factor(month),
         year = as.factor(year))
```

Scan 1

Consider NAs for the datasets

Using the Naniar package is a simple visual way of considering whether or not data is missing. Additionally, the skimr package, is a good way to quickly look at the statistics of a dataset. (Neither are shown here to make the report succinct).

Additionally, just as a double check, lets make sure there are 365 days per year (and 366 for a leap year)

```
#we now have datasets that we need to check for completeness
falls_long %>%
  is.na() %>%
  colSums()
```

```
##       date snow_depth      month       year
##          0        176          0          0
```

There are several fields with missing data. Before joining, lets complete the datasets, we will have to use some form of imputation to replace the NA values. We will consider each dataset in turn.

Dataset 1 - falls_df upon inspection has many NAs. These could be for several possible outcomes, so lets explore these.

1 - Missing data, the instrumentation was unavailable

2 - The season hadn’t commenced, and there was no snow, so no one had started measuring (this is not a role of the Bureau of Meteorology so possible)

3 - Some other random error.

Option 1 - Hotdeck is a method where the imputed value is simply replaced with the last observed value. As we have an ordered factor (snow type) we use this so that values from the wrong snow type aren’t used, and we push the order value to ensure that the date is considered in order.

Option 2 - kNN (k Nearest Neighbours) This looks at the relationship of nearest neighbours and not just the immediate (previous known observed result). Choosing the size of the neighbourhood and the weighting of similar neighbours can influence the result.

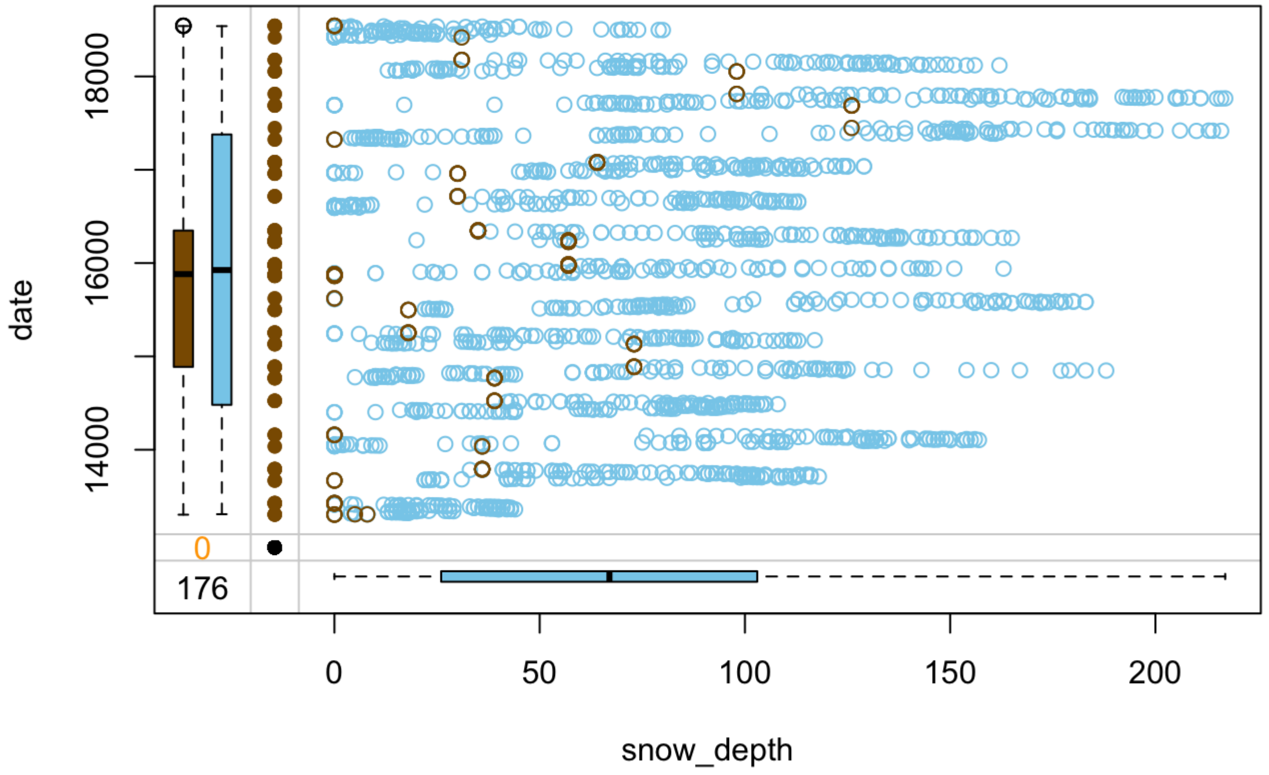
Both require the package (VIM).

Analysis of both types will be considered, and the final imputed values reviewed for outliers.

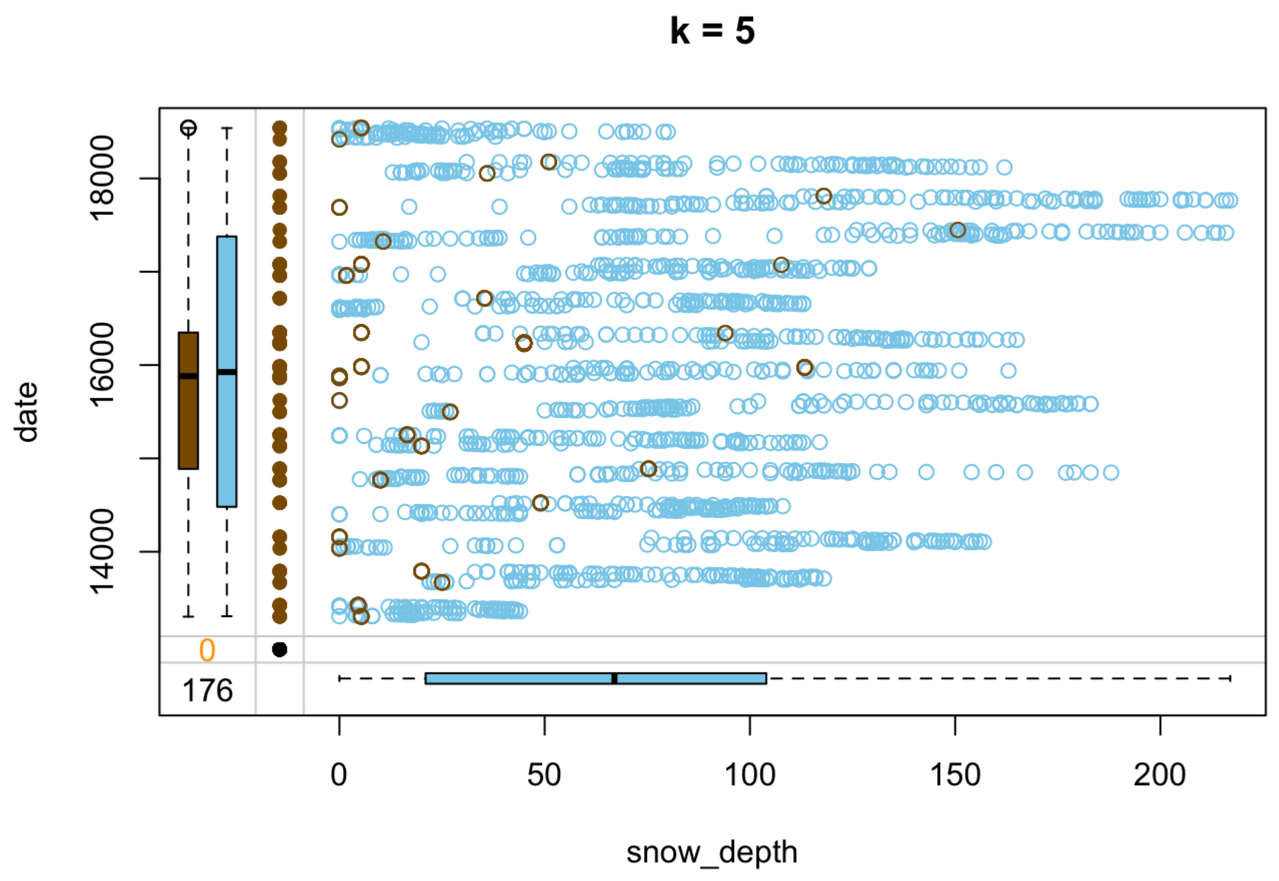
```
# 'snow_depth is the variable with the missing observations.
# "snow' a two level factor that is important, and we want to keep this, additionally we want to ensure that the
 date is kept.

#requires the package VIM.  Use the hotdeck function
falls_df_imp <- hotdeck(falls_long, variable = "snow_depth",
                        ord_var = "date")

falls_df_imp %>%
  is.na() %>%
  colSums()
```

```
##           date     snow_depth          month           year snow_depth_imp
##              0              0              0              0              0
```

```
#hotdeck dataset, review the outcome.
falls_df_imp1 <- falls_df_imp %>%
    select(snow_depth, snow_depth_imp,date) %>%
    marginplot(delimiter="imp")
```



```
#Another method is k Nearest Neighbours.  In this case the variable k is the number of neighbours to consider, an
d weighting the distance helps to reduce the likelihood of a random observation driving the outcome.
falls_df_imp2 <- falls_long%>%
  kNN(variable = "snow_depth",
      k = 3,
      numFun = weighted.mean,
      weightDist = TRUE)

# Review the output
falls_df_imp3 <- falls_df_imp2 %>%
    select(snow_depth, snow_depth_imp,date) %>%
    marginplot(delimiter="imp",main = "k = 5")
```

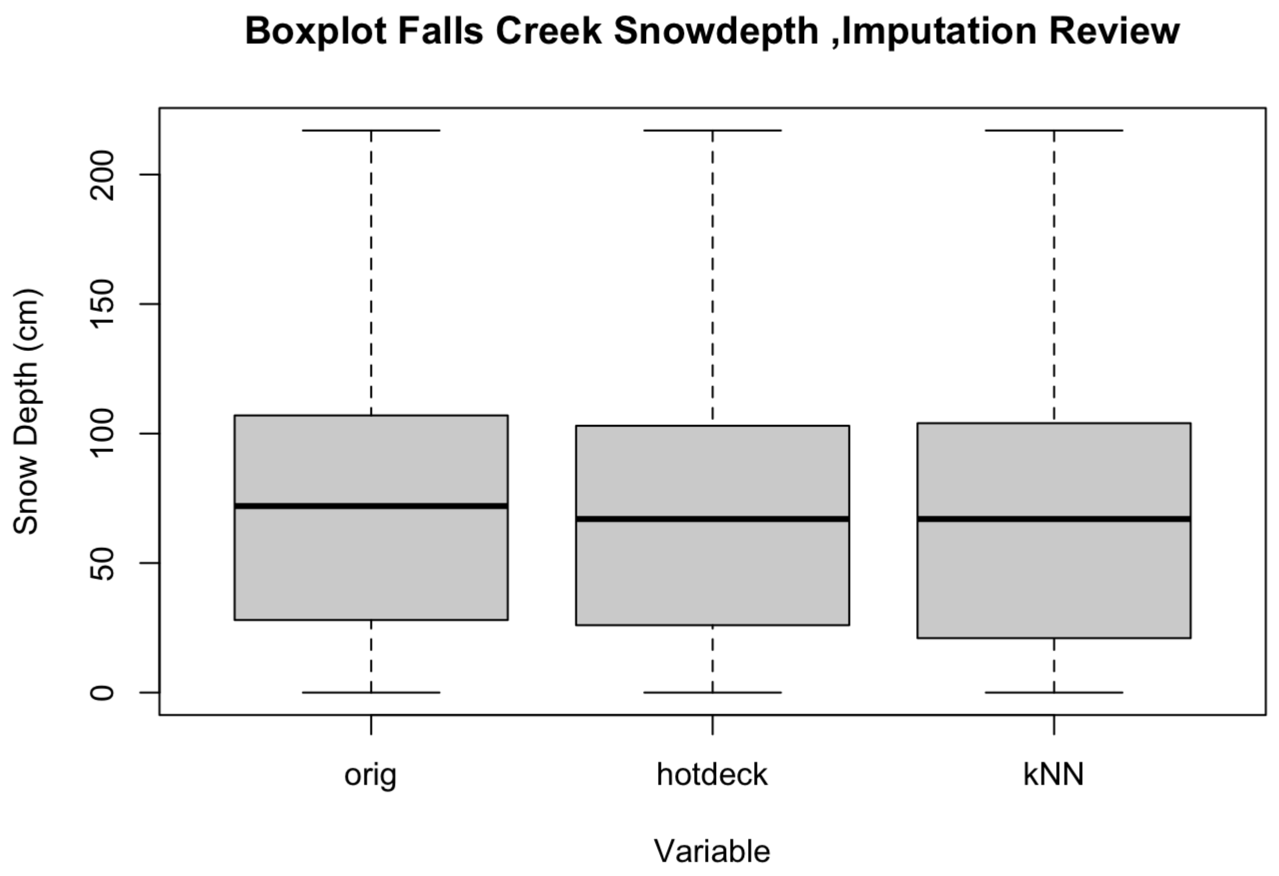


```
falls1 <- falls_long
falls2 <- falls_df_imp2

write.csv(falls1,"falls1.csv")
write.csv(falls2,"falls2.csv")
```

The imputed data for both conversion types look consistent with the data. The imputed values shown by the brown, visually appear to line up with where they belong. If we had simply imputed as zeros, we could have missed some key data points during the year.

We have two possible methods that we have reviewed for imputation (excluding mean imputation). Before making a recommendation, lets compare outliers.

```r
# original data set for outlier detection, select just the snow depth
falls_df_noimp_out <- falls_long %>%
  select(snow_depth)

# rename for cbind and plotting
falls_df_noimp_out <- falls_df_noimp_out %>%
  rename(orig = snow_depth)

# imputed data set 1 hotdeck select just the snow depth
falls_df_imp_out <- falls_df_imp %>%
  select(snow_depth)

# rename for cbind and plotting
falls_df_imp_out <- falls_df_imp_out %>%
  rename(hotdeck = snow_depth)

# imputed data set 1 hotdeck select just the snow depth
falls_df_imp_out2 <- falls_df_imp2 %>%
  select(snow_depth)

# rename for cbind and plotting
falls_df_imp_out2 <- falls_df_imp_out2 %>%
  rename(kNN = snow_depth)

falls_df_out <-
  cbind(falls_df_noimp_out, falls_df_imp_out, falls_df_imp_out2)

boxplot(falls_df_out,
        main = "Boxplot Falls Creek Snowdepth ,Imputation Review",
        xlab = "Variable",
        ylab = "Snow Depth (cm)")
```



Imputing the data shows that all missing observations have been completed, and in comparing outliers, no outliers appear present in the original data or the imputed kNN. For proceding the final data set will use the imputed kNN method to reduce any bias for outliers which is consistent wit the original data.