

Week 3 Practical


Data manipulation, Feature selection, and Correlation analysis

Overview

In this Practical we will introduce a range of useful R packages for data manipulation, feature selection, and correlation analysis.

How to use this practical?

The practical provides a set of tasks you should follow to learn a set of skills you will need for your assignments. This is by no means intended to be a complete guide to all the functions and methods available out there for a specific task. It rather focuses on commonly used methods and a minimum you need to understand to be successful in your assignments.

Icon  indicates a challenge you should try to solve. Those challenges are highly recommended.

Objectives

- Impute missing values.
- Perform feature selection and feature filtering tasks.
- Perform correlation analysis and plot correlograms.

Sharing results

While tasks should be self-explanatory, some of the challenges might be more complex. Feel free to share your results or questions in the course discussion forum. Results to all the challenges will be available UPON REQUEST (discussion forum or email).

Datasets

In **Task 1**, we will be using data from CDRC 2011 Census Geodata pack - Camden (E09000007) - <https://data.cdrc.ac.uk/dataset/introduction-spatial-data-analysis-and-visualisation-r/resource/practical-data>. This is the same dataset we used in Practical 1 and Table 1 provides an overview of the available columns.

Table 1. Camden census data description

Column Name	Description	Data type
OA	Output Area - Output areas (OA) were created for Census data, specifically for the output of census estimates. The OA is the lowest geographical level at which census estimates are provided.	Nominal
White_British	The percentage of White British population in the output area	Numeric, continuous
Low_Occupancy	The percentage of low-occupied households in the output area	Numeric, continuous
Unemployed	The unemployment rate for the output area	Numeric, continuous
Qualification	The percentage of residents who have Level 1 qualifications or above.	Numeric, continuous

For **Task 2** and **Task 3**, we will rely on Iris dataset – the most commonly utilized dataset in machine learning. More details about the dataset can be found here <https://www.kaggle.com/uciml/iris>.

Before getting started

Before going through the tasks, make sure you have a new R Notebook created in RStudio, as shown in Figure 1.

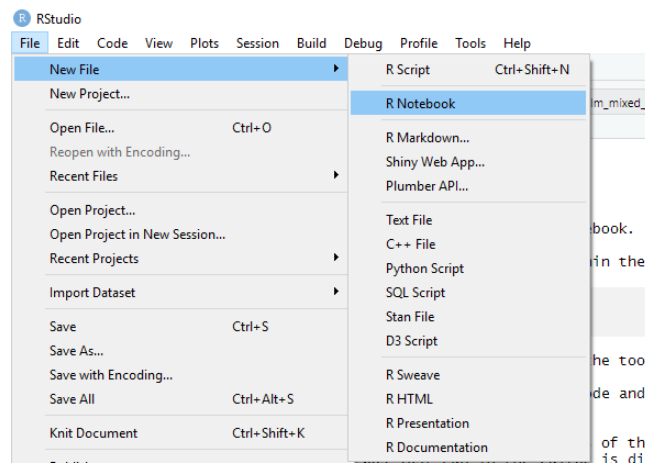


Figure 1. Opening R Notebook in RStudio

Each section of the code (marked by the grey box) should be executed within a single chunk in your Notebook.

Task 1. Handling missing values

As we discussed in Week 2 and Week 3, missing values are one of the obstacles in building predictive models. Chapter 2 (in the textbook) talks about different approaches and reasons for choosing a specific way to impute missing values.

R has several packages that deal with missing values. We will be using MICE (Multivariate Imputation via Chained Equations), as one of the commonly used packages by R users. MICE creates multiple imputations as compared to a single imputation (such as mean) and takes care of uncertainty in missing values. Please note that Chapter 2 discusses single imputation (e.g., mean or median). The principle we are using here is exactly the same, except that we will be creating multiple datasets (i.e., for each missing value we will be imputing several values).

We will be working with the same dataset we used in Practical 2:

```
# Load data
data <- read.csv(url("http://bit.ly/infs5100_camden_data"))
head(data)
```

As we did last week, you can load data either from the URL provided or from your local drive.

If you run `summary(data)`, you will see that the dataset currently has no missing values.

```
# Get summary
summary(data)
```

For the purpose of this practical, we will introduce some missing values. We can do that using the `prodNA` function from the `missForest` package. **Please notice that you might have to install `missForest` package before running the code.**

```
# Load library
library(missForest)
```

```
# Generate 10% missing values at Random
data.mis <- prodNA(data, noNA = 0.1)
```

The function above generates 10% missing values at random. To make sure we did it right, try to run `summary` again. Note that this time you should be able to see certain number of missing values.

```
# Check missing values introduced in the data
summary(data.mis)
```

white_British	Low_Occupancy	Unemployed	Qualification
Min. : 7.882	Min. : 0.000	Min. : 0.000	Min. :11.64
1st Qu.:35.909	1st Qu.: 6.015	1st Qu.: 2.479	1st Qu.:35.87
Median :44.629	Median : 9.864	Median : 4.188	Median :54.22
Mean :44.840	Mean :11.565	Mean : 4.544	Mean :51.12
3rd Qu.:54.555	3rd Qu.:16.159	3rd Qu.: 6.187	3rd Qu.:66.19
Max. :78.035	Max. :64.286	Max. :18.623	Max. :88.07
NA's :81	NA's :75	NA's :85	NA's :69

Before going further, we should make sure to remove all the categorical variables (the first column in this case).

```
# Remove categorical variables - OA, the first column in this case
data.mis <- subset(data.mis, select = -c(OA))
head(data.mis)
```

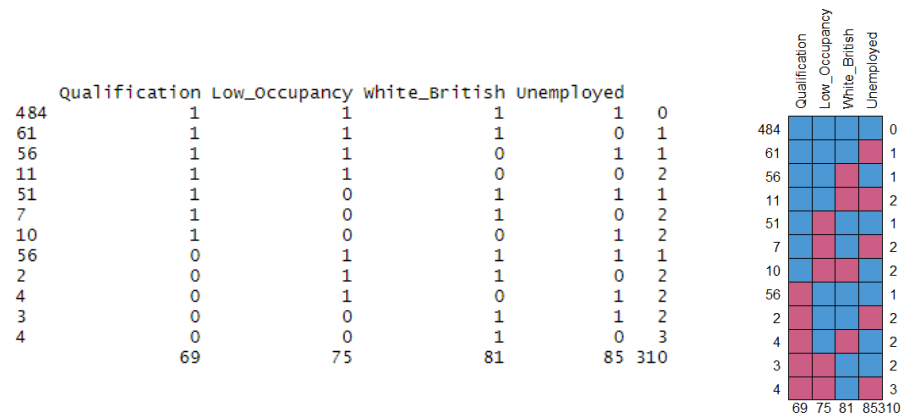
Also, make sure you have installed the `mice` package.

```
# Load library
library(mice)
```

A very useful function provided in the `mice` package is `md.pattern()`. This function provides a tabular form and a visual representation of missing values present in each variable in a dataset.

```
# Explore missing values
md.pattern(data.mis, rotate.names = TRUE)
```

In this example, the output would look like this (please note that your results might be slightly different as we generated missing values on random):



What we can see here is that there are 484 records without any missing values. There are 61 observations with missing values in `Unemployed`, 56 in `White_British`, and so on.

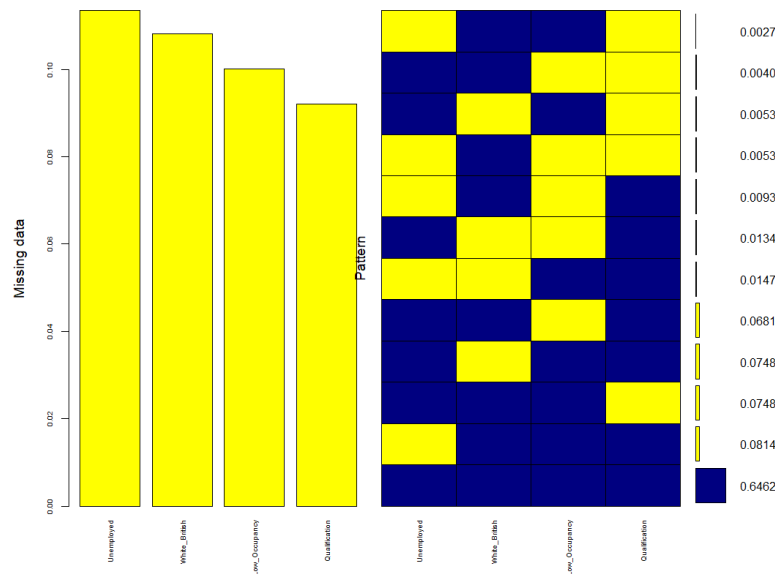
Feel free to share your result in the course discussion forum so we can comment those further.

Another commonly used approach to visualize missing value is using the `aggr` function from the `VIM` package. As always, make sure `VIM` package is installed already.

```
# Load library
library(VIM)
```

```
missing.plot <- aggr(data.mis, col=c("navyblue", "yellow"),
  numbers = TRUE, sortVars = TRUE,
  lables=names(data.mis), cex.axis=.6,
  gap=3, ylab=c("Missing data", "Pattern"))
```

The output should be similar to:



Now that we explored missing values, let's see how we can impute those.

```
# Impute missing values
data.imputed <- mice(data.mis, m=3, maxit = 50, method = 'pmm', seed = 500)
summary(data.imputed)
```

```
Number of multiple imputations: 3
Imputation methods:
white_British Low_Occupancy Unemployed Qualification
"pmm" "pmm" "pmm" "pmm"
PredictorMatrix:
white_British Low_Occupancy Unemployed Qualification
white_British 0 1 1 1
Low_Occupancy 1 0 1 1
Unemployed 1 1 0 1
Qualification 1 1 1 0
```

Here is the outline of the arguments we used to run the method:

- `m = 3`, refers to the number of imputed datasets. Recall that `mice` is different from what you read in the textbook. Instead of providing a single value, the method imputes multiple datasets. In this case, we set 3. We will get back to this shortly.
- `maxit = 50`, refers to the number of iterations taken to impute missing values. You can leave this one at default.
- `method`, refers to the method used in imputation. In this case, we used predictive mean matching. Challenge 1 invites you to try other methods as well.

If you look at the documentation for `mice`

(<https://www.rdocumentation.org/packages/mice/versions/3.6.0/topics/mice>), you will notice that `defaultMethod` argument has a list of four values:

A vector of length 4 containing the default imputation methods for 1) numeric data, 2) factor data with 2 levels, 3) factor data with > 2 unordered levels, and 4) factor data with > 2 ordered levels. By default, the method uses `pmm`, predictive mean matching (numeric data) `logreg`, logistic regression imputation (binary data, factor with 2 levels) `polyreg`, polytomous regression imputation for unordered categorical data (factor > 2 levels) `polr`, proportional odds model for (ordered, > 2 levels).

This means that even if we kept the column with the categorical data, the method would be able to handle it. However, in our case, the OA column is simply an ID of an area. That is why it is not very useful in those analysis.

Let's explore the imputed values:

```
# Explore imputed values
data.imputed$imp$Unemployed
```

	1 <dbl>	2 <dbl>	3 <dbl>
1	1.2121212	5.0000000	2.3809524
6	4.7101449	2.5423729	4.7872340
13	1.4084507	4.6242775	3.2608696
19	2.6315789	4.1958042	4.1958042
36	2.5125628	2.5423729	2.9288703
37	2.3121387	2.2026432	1.4218009
39	9.1803279	4.7120419	4.0229885
44	6.0498221	4.3750000	4.4776119
66	0.5208333	2.9166667	2.4822695
78	1.2903226	1.0152284	1.7793594

1-10 of 85 rows

Previous 1 2 3

Since we set $m=3$, you should see 3 columns – one for each of the imputed datasets.

We can, of course, select any of those, using `complete()` function.

```
# Select second complete dataset (out of 3)
data.complete <- complete(data.imputed, 2)
head(data.complete)
```

From here, you can use this dataset for any further analysis you intend to do. In our case, that would be building predictive models.

Can you plot histograms for each of the four columns (`White_British`, `Low_Occupancy`, `Unemployed`, `Qualification`) for the original dataset (the one you loaded at the beginning of the practical) and the new one with imputed missing values? Are there any observable differences?

? Challenge 1. In the example above, we used `pmm` - Predictive Mean Matching – as a method to impute data. As you can see in the documentation (<https://www.rdocumentation.org/packages/mice/versions/3.6.0/topics/mice>), there are many other built-in univariate imputation methods available.

Choose two (2) additional methods (for instance, `cart` or `quadratic`) and repeat the imputation procedure from above.

Can you explain how the methods you selected are different from `pmm`? Can you observe any differences in imputed data?

Task 2. Feature selection

As we have discussed in the first three weeks of the course, data sets often include a large number of features (or dimensions). The technique of extracting a subset of relevant features is called **feature selection** (Chapter 2 in the textbook). Feature selection is performed to remove redundant and irrelevant features and can enhance the interpretability of the model, speed up the learning process and improve the performance of the predictive model.

The most commonly used techniques include:

- **Brute-force** approach, where we try all possible feature subsets as input to data mining algorithm.
- **Embedded** approaches, where feature selection occurs naturally as part of the data mining algorithm.
- **Filter** approaches when features are selected before data mining algorithm is run.
- **Wrapper** approaches, where we use the data mining algorithm as a black box to find best subset of attributes.

In this task, we will focus on Filter approaches. Specifically, we will be using `mlr3` (Machine Learning in R – Next Generation <https://cran.r-project.org/web/packages/mlr3/index.html>) library. Make sure that, as we go ahead through this task, you have all the libraries installed.

Filter methods assign an importance value to each feature. Based on these values the features can be ranked and a feature subset can be selected. You can find a comprehensive list of filters here <https://mlr3filters.mlr-org.com/>. We will be focusing on `information_gain` as this is one of the commonly used filters that supports Classification and Regression tasks. Please note that this filter also requires `FSelectorRcpp` package to be installed.

Before we start, make sure you loaded required packages. Also, here we will set seed (to a random value) so that we make sure we generate reproducible results.

```
set.seed(999)
library(mlr3)
library(mlr3filters)
library(mlr3learners)
```

```
filter.importance = flt("information_gain") # Creating a new filter
iris.task <- tsk("iris") # Creating a Task() object from the Iris dataset
iris.feature.importance <- filter.importance$calculate(iris.task)
as.data.table(iris.feature.importance)
```

You should see the output similar (or the same) as below:

feature <chr>	score <dbl>
Petal.Width	0.9554360
Petal.Length	0.9402853
Sepal.Length	0.4521286
Sepal.Width	0.2672750

4 rows

The output should be rather straightforward. Here we can see that `Petal.Width` has the highest predictive power, whereas `Sepal.Width` can likely be ignored for the classification task. Please note that here we are looking at four features only. In case of the higher number of features, feature selection becomes more important.

For the purpose of your assignments, and beyond, it is likely that you will want to plot feature importance.

```
library(mlr3viz)
```

```
autoplot(iris.feature.importance)
```

The final step in feature selection process is to extract the top N of the most important features, regardless of the criteria you choose. To do that, we will be using `mlr3pipelines` that allow us to set:

- `filter.nfeat` - number of features to select. Mutually exclusive with `frac` and `cutoff`,
- `filter.frac` - fraction of features to keep. Mutually exclusive with `nfeat` and `cutoff`, or
- `filter.cutoff` - minimum value of filter heuristic for which to keep features. Mutually exclusive with `nfeat` and `frac`.

```
library(mlr3pipelines)
```

```
po = po("filter", filter.importance, filter.nfeat=3)
filtered.task = po$train(list(iris.task))[[1]]
filtered.task$feature_names
```

Here we set number of features to return to 3 – which means that our pipeline should return the list of the three most important features for this specific task.

To obtain the final dataset, you can run code along those lines, depending on your use case:

```
iris.filtered = subset(iris, select = filtered.task$feature_names)
head(iris.filtered)
```


Challenge 2. In the example above, we used `information_gain` as a method to select relevant features. As you can see in the documentation (mlr3filters.mlr-org.com), there are many other implemented filters.

Choose three (3) additional and relevant methods and repeat the feature selection process from above.

Can you explain how the methods you selected are different from `information_gain`?
Can you observe any differences in the feature importance?

Task 3. Correlation analysis

Very often, in exploring your datasets, you will rely on correlation analysis and correlograms. For the purpose of building predictive models, you will want to avoid using highly correlated features (please refer to Chapter 2). Thus, it is a common practice to explore the association between the variables before tackling the prediction task.

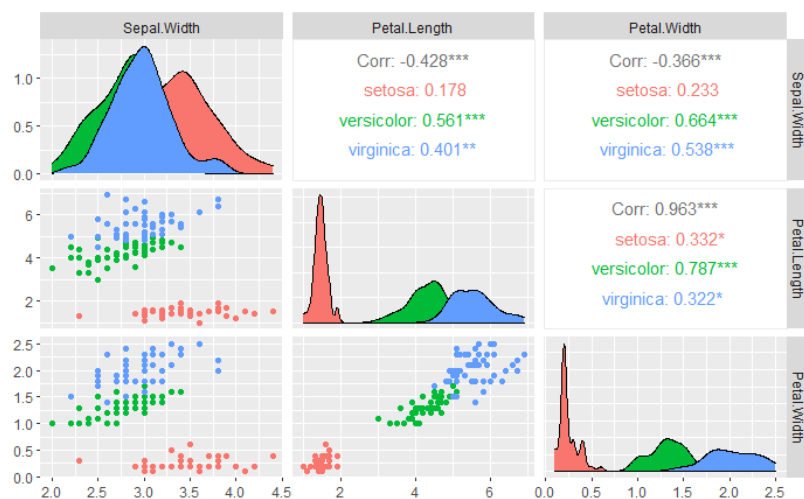
Here you can find a rich set of resources on this topic <https://www.r-graph-gallery.com/correlogram.html>.

In this task, we will briefly introduce GGally library as one way of exploring correlation between variables.

```
# Load library
library(GGally)
```

```
# Assess the distribution and correlation of variables in Iris dataset
ggpairs(iris, columns = 2:4, ggplot2::aes(colour=Species))
```

The plot should be similar to the one below:



How would you interpret the plot above?

Citation and Copyright

Contains National Statistics data Crown copyright and database right 2020.

Contains Ordnance Survey data Crown copyright and database right 2020.