Assignment 3 for Statistical Computing and Empirical Mathods - Answers

Dr. Henry WJ Reeve

Teaching block 1 2021

Introduction

This document describes your third assignment for Statistical Computing and Empirical Methods (Unit EMATM0061) on the MSc in Data Science. Before starting the assignment it is recommend that you first watch video lectures 6, 7 and 8.

You are encouraged to discuss these questions with your colleagues.

Begin by creating an Rmarkdown document with html output. You are not expected to hand in this piece of work, but it is a good idea to get used to using Rmarkdown.

Several optional extra questions have also been included. These are marked by stars (*). It is recommended that you first complete all unstarred questions before proceeding through the starred questions.

1 Random experiments, events and sample spaces

Our first question focuses on key concepts introduced in Lecture 7. Firstly, write down the definition of **random experiment**, **event** and **sample space**.

In small groups come up with your own probabilistic example:

(Q) What is the random experiment in your example?

(A) I race a tortoise and hare to see who completes the race course first.

(Q) What are the possible outcomes in your example?

(A) Either the tortoise wins 1 or the hare wins 0 the race.

(Q) What are the events in your example?

(A) The set of events is $\mathcal{E} = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.

The empty-set $\emptyset$ is the event where neither wins. For simplicity we view this as impossible.

The event $\{0\}$ occurs when the hare wins.

The event $\{1\}$ occurs when the tortoise wins.

The event $\{0, 1\}$ occurs when either the tortoise or the hare wins, so this event always occurs.

(Q) What is the sample space in your example?

The sample space is $\Omega = \{0, 1\}$ and contains both the outcome in which the tortoise wins and the outcome in which the hare wins.

2 Tidy data and iteration

We now return to the data wrangling concepts discussed in section 7.

2.1 Missing data and iteration

In this task we investigate the effect of missing data and imputation on plots.

(Q) What library do we need to load to run the function below?

(A) Load the Tidyverse library

```
library(tidyverse)
```

The following function performs imputation by mean.

```
impute_by_mean<-function(x){

  mu<-mean(x,na.rm=1) # first compute the mean of x

  impute_f<-function(z){ # coordinate-wise imputation
    if(is.na(z)){
      return(mu) # if z is na replace with mean
    }else{
      return(z) # otherwise leave in place
    }
  }
  return(map_dbl(x,impute_f)) # apply the map function to impute across vector
}
```

(Q) Create a function called `impute_by_median` which imputes missing values based on the median of the sample, rather than the mean.

(A)

You can test your function on the following sample vector:

```
v<-c(1,2,NA,4)
impute_by_median(v)
```

```
## [1] 1 2 2 4
```

(Q) Next generate a data frame with two variables `x` and `y`. For our first variable `x` we have a sequence $(x_1, x_2, \ldots, x_n)$ where $x_1 = 0$, $x_n = 10$ and for each $i = 1, \ldots, n-1$, $x_{i+1} = x_i + 0.1$. For our second variable `y` we set $y_i = 5 \times x_i + 1$ for $i = 1, \ldots, n$. Generate data of this form and place within a data frame called `df_xy`.

(A)

```
df_xy%>%head(5)
```

```
##     x   y
## 1 0.0 1.0
## 2 0.1 1.5
## 3 0.2 2.0
## 4 0.3 2.5
## 5 0.4 3.0
```

The **map2()** function is similar to the **map()** function but iterates over two variables in parallel rather than one. You can learn more here https://purrr.tidyverse.org/reference/map2.html. The following simple example shows you how **map2_dbl()** can be combined with the **mutate()** function.

```
df_xy%>%
  mutate(z=map2_dbl(x,y,~.x+.y))%>%
  head(5)
```

```
##     x   y   z
## 1 0.0 1.0 1.0
## 2 0.1 1.5 1.6
## 3 0.2 2.0 2.2
## 4 0.3 2.5 2.8
## 5 0.4 3.0 3.4
```

We will now use **map2_dbl()** to generate a new data frame with missing data.

(Q) First create a function `sometimes_missing` with two variables `index` and `value`. The function should return `NA` if `index` is divisible by 5 and returns `value` otherwise.

(A)

```
sometimes_missing<-function(index,value){

  if(index%%5==0){
    return(NA)
  }else{
    return(value)
  }

}
```

Your function should produce the following outputs:

```
sometimes_missing(14,25)
```

```
## [1] 25
```

```
sometimes_missing(15,25)
```

```
## [1] NA
```

(Q) Next generate a new data frame called `df_xy_missing` with two variables `x` and `y`, but some missing data. For the first variable `x` we have a sequence $(x_1, \cdots, x_n)$, which is precisely the same as with `df_xy`. For the second variable `y` we have a sequence $(\tilde{y}_1, \cdots, \tilde{y}_n)$ where $\tilde{y}_i = \text{NA}$ if i is divisible by 5 and $\tilde{y}_i = y_i$ for i not divisible by 5. To generate the dataframe `d_xy_missing` you may want to make use of the functions **row_number()**, **map2_dbl()**, **mutate()** as well as **sometimes_missing()**.

(A)

```
df_xy_missing<-df_xy%>%
  mutate(y=map2_dbl(row_number(),y,sometimes_missing))

df_xy_missing<-df_xy%>%
  mutate(y=map2_dbl(.x=row_number(),.y=y,sometimes_missing))
```

Check that the first ten rows of your data frame are as follows:

```
df_xy_missing%>%
  head(10)
```

```
##      x   y
## 1  0.0 1.0
## 2  0.1 1.5
## 3  0.2 2.0
## 4  0.3 2.5
## 5  0.4  NA
## 6  0.5 3.5
## 7  0.6 4.0
## 8  0.7 4.5
## 9  0.8 5.0
## 10 0.9  NA
```

(Q) Create a new data frame `df_xy_imputed` with two variables `x` and `y`. For the first variable `x` we have a sequence $(x_1, \cdots, x_n)$, which is precisely the same as with `df_xy`. For the second variable `y` we have a sequence $(y'_1, \cdots, y'_n)$ which is formed from $(\tilde{y}_1, \cdots, \tilde{y}_n)$ by imputing any missing values with the median. To generate `df_xy_imputed` from "'df_xy_missing by applying a combination of the functions **mutate** and **impute_by_median()**.

(A)

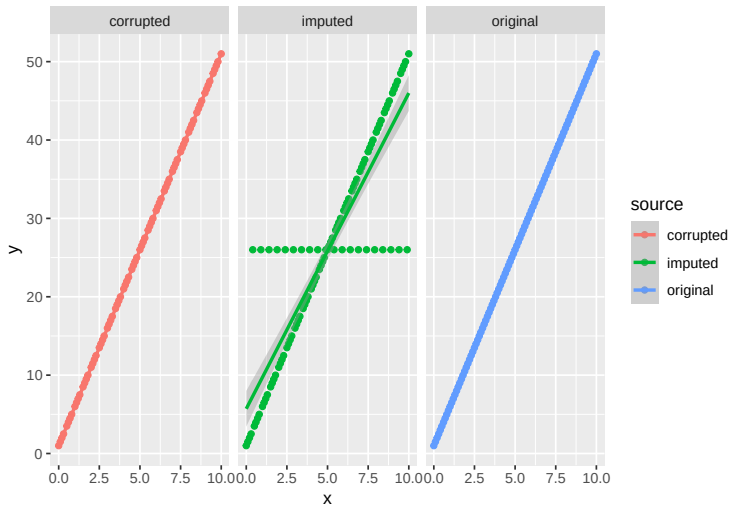
```
df_xy_impute<-df_xy_missing%>%
  mutate(y=impute_by_median(y))
```

Combine the dataframes df_xy, df_xy_missing and df_xy_impute within a single dataframe called df_combined, along with an additional column indicating the source of the data.

```
df_xy<-df_xy%>%
  mutate(source="original")

df_xy_missing<-df_xy_missing%>%
  mutate(source="corrupted")

df_xy_impute<-df_xy_impute%>%
  mutate(source="imputed")

df_combined<-rbind(df_xy,df_xy_missing,df_xy_impute)
```

Plot the original data, the corrupted data and the imputed data together together with a trend line for each sample.

```
ggplot(df_combined,aes(x=x,y=y,color=source))+geom_point()+
  facet_wrap(~source)+geom_smooth(method="lm")
```



(Q) Do the imputed values y_i' give reasonable estimates of the true values y_i?

(A) The estimates y_i' seem reasonable from a univariate perspective, where we focus on the property of central location. However, these estimates have the weakness of disregarding bivariate information which takes into account the values of other covariates. More sophisticated methods of handling missing data aim to take such information into account.

2.2 Tidying data with pivot functions

In this task you will read in data from a spreadsheet and apply some data wrangling tasks to tidy that data.

First download the excel spreadsheet entitled "**HockeyLeague.xlsx**". The excel file contains two spreadsheets - one with the wins for each team and one with the losses for each team. To read this spreadsheet into R we shall make use of the **readxl** library. You may need to install the library:

```
install.packages("readxl")
```

The following code shows how to read in a sheet within an excel file as a data frame. You will need to edit the `folder_path` variable to be the directory which contains your copy of the spreadsheet.

```
library(readxl) # load the readxl library

folder_path<-"your_folder_path..." # set this to the name of the
                                   # directory containing "HockeyLeague.xlsx"

file_name<-"HockeyLeague.xlsx" # set the file name

file_path<-paste(folder_path,file_name,sep="") # create the file_path

wins_data_frame<-read_excel(file_path,sheet="Wins") # read of a sheet from an xl file
```

Inspect the first 3 rows of the first five columns:

```
wins_data_frame %>%
  select(1:5)%>%
  head(3)
```

```
## # A tibble: 3 x 5
##   ...1   `1990`   `1991`   `1992`   `1993`
##   <chr>  <chr>    <chr>    <chr>    <chr>
## 1 Ducks  30 of 50 11 of 50 30 of 50 12 of 50
## 2 Eagles 24 of 50 12 of 50 37 of 50 14 of 50
## 3 Hawks  20 of 50 22 of 50 33 of 50 11 of 50
```

A cell value of the form "a of b" means that a games were won out of a total of b for that season. For example, the element for the "Ducks" row of the "1990" column is "30 of 50" meaning that 30 out of 50 games were won that season.

(Q) Is this tidy data?

(A) This would not usually be considered as tidy data. Firstly, the number of wins and the total number of games for a given season are contained in the same column, rather than being contained within a single column. Indeed the number of wins for different years are spread across multiple columns. A tidy data format would have a single column containing the number of wins for all of the seasons together with a separate column for the year.

(Q) Now apply your data wrangling skills to transform the "wins_data_frame" data frame object into a data frame called "wins_tidy" which contains the same information but has just four columns entitled "Team", "Year", "Wins", "Total". The "Team" column should contain the team name, the "Year" column should contain the year, the "Wins" column should contain the number of wins for that season and the "Total" column the total number of games for that season. The first column should be of character type and the remaining columns should be of integer type. You can do this by combining the following functions: **rename()**, **pivot_longer()**, **mutate()** and **separate()**.

(A)

You can check the shape of your data frame and the first five rows as follows:

```
wins_tidy%>% dim() # check the dimensions
```

```
## [1] 248   4
```

```
wins_tidy%>%head(5) # inspect the top 5 rows
```

```
## # A tibble: 5 x 4
##   Team   Year  Wins Total
##   <chr> <int> <int> <int>
## 1 Ducks  1990    30    50
## 2 Ducks  1991    11    50
## 3 Ducks  1992    30    50
## 4 Ducks  1993    12    50
## 5 Ducks  1994    24    50
```

The "HockeyLeague.xlsx" also contains a sheet with the losses for each team by season.

(Q) Apply a similar procedure to read the data from this sheet and transform that data into a dataframe called "losses_tidy" with four columns: "Team", "Year", "Losses", "Total" which are similar to thos in the "wins_tidy" data frame except for the "Losses" column gives the number of losses for a given season and team, rather than the number of losses.

(A)

```
w_l_narrow<-function(w_or_l){

  return(
    read_excel(file_path,sheet=w_or_l)%>%
      rename(Team=...1)%>%
      pivot_longer(!Team,names_to="Year",values_to="val")%>%
      mutate(Year=as.integer(Year))%>%
      separate(col=val,into=c(w_or_l,"Total"),sep=" of ",convert=TRUE)
    )
}

wins_tidy<-w_l_narrow(w_or_l="Wins")
losses_tidy<-w_l_narrow(w_or_l="Losses")
```

You may notice that the number of wins plus the number of losses for a given team, in a given year does not add up to the total. This is because some of the games are neither wins nor losses but draws. That is, for a given year the number of draws is equal to the total number of games minus the sum of the wins and losses.

(Q) Now combine your two data frames, "wins_tidy" and "losses_tidy", into a single data frame entitled "hockey_df" which has 248 rows and 9 columns: A "Team" column which gives the name of the team as a character, the "Year" column which gives the season year, the "Wins" column which gives the number of wins for that team in the given year, the "Losses" column which gives the number of losses for that team in the given year and the "Draws" column which gives the number of draws for that team in the given year,

the "Wins_rt" which gives the wins as a proportion of the total number of games (ie. Wins/Total) and similarly the "Losses_rt" and the "Draws_rt" which gives the losses and draws as a proportion of the total, respectively. To do this you can make use of the **mutate()** function. You may also want to utilise the **across()** function for a slightly neater solution.

(A)

```
hockey_df<-inner_join(wins_tidy,losses_tidy)%>%
  mutate(Draws=Total-Wins-Losses)%>%
  mutate(across(starts_with(c("Wins","Losses","Draws")),~.x/Total,
                .names="{.col}_rt"))
```

```
## Joining, by = c("Team", "Year", "Total")
```

The top five rows of your data frame should look as follows:

```
hockey_df%>% head(5)
```

```
## # A tibble: 5 x 9
##   Team   Year  Wins Total Losses Draws Wins_rt Losses_rt Draws_rt
##   <chr> <int> <int> <int>  <int> <int>   <dbl>     <dbl>    <dbl>
## 1 Ducks  1990    30    50     20     0    0.6       0.4      0
## 2 Ducks  1991    11    50     37     2    0.22      0.74     0.04
## 3 Ducks  1992    30    50      1    19    0.6       0.02     0.38
## 4 Ducks  1993    12    50     30     8    0.24      0.6      0.16
## 5 Ducks  1994    24    50      7    19    0.48      0.14     0.38
```

(Q) To conclude this task generate a summary data frame which displays, for each team, the median win rate, the mean win rate, the median loss rate, the mean loss rate, the median draw rate and the mean draw rate. The number of rows in your summary should equal the number of teams. These should be sorted in descending order or median win rate. You may want to make use of the following functions: **select()**, **group_by()**, **across()**, **arrange()**.

(A)

```
hockey_df%>%
  select(-Wins,-Draws,-Losses)%>%
  group_by(Team)%>%
  summarise(across(starts_with(c("Wins","Losses","Draws")),list(md=median,mn=mean),
                   .names="{substring(.col,1,1)}_{.fn}"))%>%
  arrange(desc(W_md))
```

```
## # A tibble: 8 x 7
##   Team         W_md  W_mn  L_md  L_mn  D_md  D_mn
##   <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Eagles      0.45  0.437 0.25  0.279 0.317 0.284
## 2 Penguins    0.45  0.457 0.3   0.310 0.133 0.232
## 3 Hawks       0.417 0.388 0.233 0.246 0.32  0.366
## 4 Ducks       0.383 0.362 0.34  0.333 0.25  0.305
## 5 Owls        0.32  0.333 0.3   0.33  0.383 0.337
## 6 Ostriches   0.3   0.309 0.4   0.395 0.267 0.296
## 7 Storks      0.3   0.284 0.22  0.283 0.48  0.433
## 8 Kingfishers 0.233 0.245 0.34  0.360 0.4   0.395
```

2.3 Most correlated variables (*)

This data wrangling task is slightly more challenging. You may want to return to this task once you have completed the unstarred questions in sections 3 and 4 below.

The objective is to investigate, for each numerical variable within a data set, which other numerical variables have the largest correlation (in absolute value).

In lecture 6 we introduced the following function called `max_cor_var`. The function entitled `max_cor_var` takes as input a data frame `df` and a column name `col_name`. It then extracts the variable with name col_name and determines which other numerical variables within the data set have the highest correlation (in absolute value) with that variable. It then returns a data frame containing the name of the variable "var_name" and the corresponding correlation "cor". Begin by making sure you understand the structure of the function.

```
max_cor_var<-function(df,col_name){
  # function to determine the variable with maximal correlation

  v_col<-df%>%select(all_of(col_name))
  # extract variable based on col_name

  df_num<-df%>%
    select_if(is.numeric)%>%
    select(-all_of(col_name))
  # select all numeric variables excluding col_name

  correlations<-unlist(map(df_num,
                           function(x){cor(x,v_col,use="complete.obs")}))
  # compute correlations with all other numeric variables

  max_abs_cor_var<-names(which(abs(correlations)==max(abs(correlations))))
  # extract the  variable name
  cor<-as.double(correlations[max_abs_cor_var])
  # compute the correlation

  return(data.frame(var_name=max_abs_cor_var,cor=cor))
  # return dataframe
}
```

(Q) Next generate a new function called "top_correlates_by_var" which takes input a data frame "df" and outputs a data frame with a single row. The column names of this output data frame should coincide with the names of the numerical columns within the input dataframe "df". For each column name, the value should be equal to variable name corresponding to the numerical variable which has the highest level of correlation (in absolute value) to the variable with that column name, but is not equal to it.

(A)

You can test your function as follows. By using the Palmer penguins data set you should obtain the following output.

```
library(palmerpenguins)

penguins%>%
  top_correlates_by_var()
```

```
## # A tibble: 1 x 5
##   bill_length_mm    bill_depth_mm     flipper_length_mm body_mass_g   year
##   <chr>             <chr>             <chr>             <chr>         <chr>
## 1 flipper_length_mm flipper_length_mm body_mass_g       flipper_leng~ flipper_l~
```

(Q) Next use a combination of the functions **group_by()**, **nest()**, **mutate()**, **select()**, **unnest()** together with your new function **top_correlates_by_var()** to determine those variables with highest correlation, broken down by species of penguin. Your results should be as follows.

```
## # A tibble: 3 x 5
## # Groups:   species [3]
##   species   bill_length_mm bill_depth_mm  flipper_length_mm body_mass_g
##   <fct>     <chr>          <chr>          <chr>             <chr>
## 1 Adelie    body_mass_g    body_mass_g    body_mass_g       bill_depth_mm
## 2 Gentoo    body_mass_g    body_mass_g    bill_depth_mm     bill_depth_mm
## 3 Chinstrap bill_depth_mm  bill_length_mm body_mass_g       flipper_length_mm
```

(A)

```
penguins%>%
  group_by(species)%>%
  nest()%>%
  mutate(top_cor=map(data,top_correlates_by_var))%>%
  select(-data)%>%
  unnest(cols=top_cor)%>%
  select(-year)
```

3 Elementary set theory

In this section we consider some of the concepts introduced in Lecture 7.

Remember that a set is just a collection of objects. All that matters for the identity of a set is the objects it contains. In particular, the elements within the set are unordered, so for example the set $\{1, 2, 3\}$ is **exactly** the same as the set $\{3, 2, 1\}$. In addition, since sets are just collections of objects, each object can only be either included or excluded and multiplicities do not change the nature of the set. In particular, the set $\{1, 2, 2, 2, 3, 3\}$ is **exactly** the same as the set $A = \{1, 2, 3\}$. In general there is no concept of "position" within a set, unlike a vector or matrix.

3.1 Rolling a dice

Let's suppose we role a dice. The set of possible outcomes is $\Omega := \{1, 2, 3, 4, 5, 6\}$ corresponding to the different sides of the dice which could land face up.

Let's introduce events A, B defined by $A := \{1, 2, 3\}$, $B := \{2, 4, 6\}$, $C := \{4, 5, 6\}$.

Unions

(Q) What is $A \cup B$? **(A)** $A \cup B = \{1, 2, 3, 4, 6\}$.

(Q) What is $A \cup C$? **(A)** $A \cup C = \{1, 2, 3, 4, 5, 6\}$.

Intersections

(Q) What is $A \cap B$? **(A)** $A \cap B = \{2\}$.

(Q) What is $A \cap C$? **(A)** $A \cap C = \emptyset$.

Complements

(Q) What is $A \setminus B$? **(A)** $A \setminus B = \{1, 3\}$.

(Q) What is $A \setminus C$? **(A)** $A \cap C = A = \{1, 2, 3\}$.

Disjointness

(Q) Are A and B disjoint? **(A)** No, since $A \cap B = \{2\} \neq \emptyset$. **(Q)** Are A and C disjoint? **(A)** Yes, since $A \cap C = \emptyset$.

Partitions

(Q) Write down a partition of $\Omega := \{1, 2, 3, 4, 5, 6\}$ consisting of two sets. **(A)** The sets A and C form a partition.

(Q) Write down another partition of Ω containing three sets. **(A)** The sets $\{1, 2\}$, $\{3, 4\}$ & $\{5, 6\}$ form a partition of Ω.

3.2 Complements, subsets and De Morgan's laws

Let Ω be a sample space. Recall that for an event $A \subseteq \Omega$ the complement $A^c \equiv \Omega \setminus A := \{\omega \in \Omega \ : \ \omega \notin A\}$. Take a pair of events $A \subseteq \Omega$ and $B \subseteq \Omega$.

Recall that to show two sets A_0 and A_1 are equal you must show that: (1) $x \in A_0$ implies $x \in A_1$ and (2) $x \in A_1$ implies $x \in A_0$.

Complements

(Q) Can you give an expression for $(A^c)^c$ without using the notion of a complement? **(A)** $A = (A^c)^c$

(Q) What is Ω^c? **(A)** $\Omega^c = \emptyset$.

Subsets

(Q) Show that if $A \subseteq B$ then $B^c \subseteq A^c$.

(A) Suppose that $A \subseteq B$. Now suppose $x \in B^c$. Then $x \notin B$. Hence, $x \notin A$ since if it were the case that $x \in A$ we would also have $x \in B$. Hence, $x \in A^c$. Since this holds for all elements of B^c we must have $B^c \subseteq A^c$.

Intersections

(Q) Show that $(A \cap B)^c = A^c \cup B^c$.

(A) First, we show that $x \in (A \cap B)^c$ implies $x \in A^c \cup B^c$. Suppose that $x \in (A \cap B)^c$. Then $x \notin A \cap B$, so either $x \notin A$, in which case $x \in A^c$ or $x \notin B$, in which case $x \in B^c$ (or both holds). Hence, if $x \in (A \cap B)^c$ we must have $x \in A^c \cup B^c$.

Second, we show that $x \in A^c \cup B^c$ implies $x \in (A \cap B)^c$.Suppose that $x \in A^c \cup B^c$. Then either $x \in A^c$, so $x \notin A$, so $x \notin A \cap B$, or $x \in B^c$, so $x \notin B$ so $x \notin A \cap B$. Either way $x \notin A \cap B$ and hence $x \in (A \cap B)^c$.

(Q) Let's suppose we have a sequence of events $A_1, A_2, \ldots, A_K \subseteq \Omega$. Can you write out an expression for $\left(\bigcap_{k=1}^K A_k\right)^c$ in terms of unions and complements?

(A) $\left(\bigcap_{k=1}^K A_k\right)^c = \bigcup_{k=1}^K A_k^c$.

Union

(Q) Show that $(A \cup B)^c = A^c \cap B^c$.

(A) Let's apply $(A \cap B)^c = A^c \cup B^c$ with A^c in place of A and B^c in place of B.

This gives the result $(A^c \cap B^c)^c = (A^c)^c \cup (B^c)^c$. By taking complements, and applying the result $(S^c)^c = S$ with A, B and $(A^c \cap B^c)$ in place of S yields

$$A^c \cap B^c = \{(A^c \cap B^c)^c\}^c = \{(A^c)^c \cup (B^c)^c\}^c = (A \cup B)^c,$$

as required.

(Q) Let's suppose we have a sequence of events $A_1, A_2, \ldots, A_K \subseteq \Omega$. Can you write out an expression for $\left(\bigcup_{k=1}^K A_k\right)^c$ in terms of intersections and complements?

(A) $\left(\bigcup_{k=1}^K A_k\right)^c = \bigcap_{k=1}^K A_k^c$.

Note The results $(A \cap B)^c = A^c \cup B^c$ and $(A \cup B)^c = A^c \cap B^c$ are often referred to as "De Morgan's laws".

3.3 Cardinality and the set of all subsets

Suppose that $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ contains K elements for some natural number $K \in \mathbb{N}$. Hence, Ω has cardinality K.

Let $\mathcal{E} := \{A \subseteq \Omega\}$ be the set of all subsets.

(Q) Give a formula for the cardinality of $\mathcal{E}$ in terms of K.

(A) The cardinality of $\{A \subseteq \Omega\}$ is 2^K.

3.4 Disjointness and partitions

Suppose we have a sample space Ω.

(Q) Can you think of a set which is disjoint from every other set? That is, find a set $A \subseteq \Omega$ such that $A \cap B = \emptyset$ for all sets $B \subseteq \Omega$.

(A) The emptyset $\emptyset$ satisfies $\emptyset \cap B = \emptyset$ for all sets $B \subseteq \Omega$.

Suppose we have events A_1, A_2, A_3, A_4 all subsets of Ω.

Define events S_1, S_2, S_3, S_4 all subsets of Ω by $S_1 := A_1$, $S_2 := A_2 \setminus A_1$, $S_3 := A_3 \setminus (A_1 \cup A_2)$, $S_4 := A_4 \setminus (A_1 \cup A_2 \cup A_3)$.

(Q) Show that S_1, S_2, S_3, S_4 form a **partition** of $A_1 \cup A_2 \cup A_3 \cup A_4$.

(A) We must show (1) that $S_1 \cup S_2 \cup S_3 \cup S_4 = A_1 \cup A_2 \cup A_3 \cup A_4$ and (2) that S_1, S_2, S_3, S_4 are disjoint.

(1) Take $x \in S_1 \cup S_2 \cup S_3 \cup S_4$. Then $x \in S_i = A_i \setminus_{j<i} A_j$ for some $i \in \{1,2,3,4\}$ and so $x \in A_i$ and hence $x \in A_1 \cup A_2 \cup A_3 \cup A_4$. On the other hand, if $x \in A_1 \cup A_2 \cup A_3 \cup A_4$ we may choose $i \in \{1,2,3,4\}$ as small as possible with $x \in A_i$. It follows that $x \in S_i = A_i \setminus_{j<i} A_j$ and so $x \in S_1 \cup S_2 \cup S_3 \cup S_4$. Hence, the sets $S_1 \cup S_2 \cup S_3 \cup S_4$ and $A_1 \cup A_2 \cup A_3 \cup A_4$ have exactly the same elements and so are equal.

(2) To show S_1, S_2, S_3, S_4 are disjoint let's show that there cannot be any $x \in S_{i_0} \cap S_{i_1}$ with $i_0 < i_1$. Indeed, if $x \in S_{i_1} = A_{i_1} \setminus \left(\bigcup_{j<i_1} A_j\right)$ so $x \notin A_{i_0}$ and so $x \notin S_{i_0}$.

3.5 Uncountable infinities (**)

This is a challenging optional extra. You may want to return to this question once you have completed section 4.

(Q) Show that the set of numbers $\Omega := [0,1]$ is uncountably infinite.

(A) We use a proof due to Cantor known as the *diagonalisation argument*. We shall suppose that $[0,1]$ is countable and then deduce a contradiction. So suppose that $[0,1]$ is countable and that there is an enumeration $(a_n)_{n\in\mathbb{N}}$ of $[0,1]$. For each $n \in \mathbb{N}$ let $(a_{n,j})_{j\in\mathbb{N}}$ be the corresponding decimal expansion, so each $a_{nj} \in \{0,1,\ldots,9\}$ and $a_n = 0.a_{n,1}a_{n,2}a_{n,3}a_{n,4}\ldots$. Now choose $x \in [0,1]$ with decimal expansion $(x_j)_{j\in\mathbb{N}}$ by setting $x_j \in \{1,2,\ldots,9\} \setminus \{a_{j,j}\}$ for all $j \in \mathbb{N}$. It follows that $x \neq a_n$ for all $n \in \mathbb{N}$, and hence $(a_n)_{n\in\mathbb{N}}$ is not an enumeration of $[0,1]$.

4 Introduction to probability

In this section we consider some of the concepts introduced in Lecture 8.

4.1 Probability of a complement

(Q) Prove that if Ω is a sample space, $S \subseteq \Omega$ is an event and $S^c \equiv \Omega \setminus S$ is its complement we have

$$\mathbb{P}(S^c) = 1 - \mathbb{P}(S).$$

(A) Define a sequence of sets $A_1, A_2, \ldots$ by $A_1 = S$, $A_2 = S^c$ and $A_i = \emptyset$ for all $i \geq 3$. It follows that the sequence $A_1, A_2, \ldots$ is pairwise disjoint. Moreover, $\bigcup_{i\in\mathbb{N}} A_i = S \cup S^c \cup \emptyset = \Omega$ and so $\mathbb{P}\left(\bigcup_{i\in\mathbb{N}} A_i\right) = 1$ by the first rule of probability. Moreover, we have seen that $\mathbb{P}(\emptyset) = 0$. Hence, by third rule of probability we have

$$1 = \mathbb{P}(\Omega) = \mathbb{P}\left(\bigcup_{i\in\mathbb{N}} A_i\right) = \sum_{i\in\mathbb{N}} \mathbb{P}(A_i) = \mathbb{P}(S) + \mathbb{P}(S^c) + \sum_{k\geq 3} \mathbb{P}(\emptyset) = \mathbb{P}(S) + \mathbb{P}(S^c).$$

By rearranging we have $\mathbb{P}(S^c) = 1 - \mathbb{P}(S)$.

4.2 The union bound

The third rule of probability states that if we a sequence of **disjoint** sets $A_1, A_2, A_3, \ldots$ then

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) = \sum_{k=1}^{\infty} \mathbb{P}(A_k).$$

We also no from the union bound that for **any** sequence of sets $A_1, A_2, A_3, \ldots$ whatsoever (not necessarily disjoint) we have

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) \leq \sum_{k=1}^{\infty} \mathbb{P}(A_k).$$

(Q) Give an example sequence of sets $A_1, A_2, A_3, \ldots$ in which $\mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) \neq \sum_{k=1}^{\infty} \mathbb{P}(A_k)$.

(A) Consider the sample space $\Omega = \{0\}$, so $\mathcal{E} = \{\emptyset, \{0\}\}$ and let $\mathbb{P}$ be the probability on $\mathcal{E}$ with $\mathbb{P}(\emptyset) = 0$ and $\mathbb{P}(\{0\}) = 1$. Now take $A_n = \{0\}$ for all $n \in \mathbb{N}$. It follows that

$$1 = \mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) < \infty = \sum_{k=1}^{\infty} \mathbb{P}(A_k).$$

4.3 Verifying the rules of probability

Consider a setting in which $\Omega = \{0, 1\}$, and $\mathcal{E} := \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.

Fix $q \in [0, 1]$ and define a probability $\mathbb{P} : \mathcal{E} \to [0, 1]$ by

$$\mathbb{P}(A) := \begin{cases} 0 & \text{if } A = \emptyset \\ 1 - q & \text{if } A = \{0\} \\ q & \text{if } A = \{1\} \\ 1 & \text{if } A = \{0, 1\}. \end{cases}$$

(Q) Verify that the probability space $(\Omega, \mathcal{E}, \mathbb{P})$ satisfies the three rules of probability.

(A) Recall that the rules are **1** We have $\mathbb{P}(A) \geq 0$ for all $A \in \mathcal{E}$; **2** The sample space has probability one $\mathbb{P}(\Omega) = 1$; **3** Given a sequence of disjoint sets $A_1, A_2, \ldots$, we have $\mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) = \sum_{k=1}^{\infty} \mathbb{P}(A_k)$.

The first two rules are immediate from the construction of $\mathbb{P}$. To prove the third rule we consider three cases.

1. $A_n = \emptyset$ for all $n \in \mathbb{N}$. Then $0 = \mathbb{P}(\emptyset) = \mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) = \sum_{k=1}^{\infty} \mathbb{P}(A_k)$.
2. There is exactly one $n \in \mathbb{N}$ with $A_n \neq \emptyset$. Then $\bigcup_{k=1}^{\infty} A_k = A_n$ and $\mathbb{P}(A_k) = 0$ for all $k \neq n$. Hence, we have $\mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) = \mathbb{P}(A_n)$ and $\sum_{k=1}^{\infty} \mathbb{P}(A_k) = \mathbb{P}(A_n) + \sum_{k \neq n} \mathbb{P}(A_k) = \mathbb{P}(A_n)$, as required.
3. There is one $n_0 \in \mathbb{N}$ with $A_{n_0} = \{0\}$ and one $n_1 \in \mathbb{N}$ with $A_{n_1} = \{1\}$ and $A_n = \emptyset$ for all $n \in \mathbb{N} \setminus \{n_0, n_1\}$. It follows that $\bigcup_{k=1}^{\infty} A_k = \Omega$ so $\mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right) = 1$. Moreover,

$$\sum_{k=1}^{\infty} \mathbb{P}(A_k) = \mathbb{P}(A_{n_0}) + \mathbb{P}(A_{n_1}) + \sum_{k \in \mathbb{N} \setminus \{n_0, n_1\}} \mathbb{P}(A_k) = (1 - q) + q = 1 = \mathbb{P}\left(\bigcup_{k=1}^{\infty} A_k\right),$$

   as required.

Moreover, there are no more possibilities with $A_1, A_2, \ldots$, disjoint.

4.4 Rotation invariant measures on $[0,1]$ (***)

This is a challenging optional extra.

I recommend completing the other questions on the assignment first.

Consider the sample space $\Omega := [0,1)$. Recall that for a real number $x \in \mathbb{R}$, the floor $\lfloor x \rfloor$ denotes the largest integer $n \in \mathbb{Z}$ with $n \leq x$. Given a number $a \in \mathbb{R}$ we define a *rotation* map $T_a : \Omega \to \Omega$ by

$$T_a(x) := (x + a) - \lfloor (x + a) \rfloor.$$

Observe that $T_a : [0,1] \to [0,1]$ is a one-to-one mapping.

A probability $\mathbb{P}$ on Ω is said to be *rotation invariant* if for any event $A \in \mathcal{E}$ and any $a \in \mathbb{R}$ we have $\mathbb{P}[T_a(A)] = \mathbb{P}(A)$.

(Q) Show that we cannot define a rotation invariant probability measure on $\mathcal{E} := \{A \subseteq \Omega\}$, the set of all sets.

(A) We define a family of subsets $\{S(x) : x \in [0,1)\}$ by $S(x) := \{y \in \Omega : y = T_a(x) \text{ for some } a \in \mathbb{Q}\}$. Observe that for all a,$b \in \mathbb{R}$ and $x \in T_a(T_b(x)) = T_{a+b}(x)$. Now suppose $S(x_0) \cap S(x_1) \neq \emptyset$. Then there exists $z = T_{a_0}(x_0) = T_{a_1}(x_1)$ for some $a_0, a_1 \in \mathbb{Q}$. Hence, if $w \in S(x_0)$ then $w = T_{a_w}(x_0)$ for some $a_w \in \mathbb{Q}$ and hence

$$w = T_{a_w}(T_{a_0}^{-1}(z_0)) = T_{a_w}(T_{a_0}^{-1}(T_{a_1}(x_1))) = T_{a_w - a_0 + a_1}(x_1),$$

and so $w \in S(x_1)$. Similarly, if $w \in S(x_1)$ then $w \in S(x_0)$. Hence, the family of sets $\{S(x) : x \in [0,1)\}$ form a partition of Ω. Now choose $Z \subset \Omega$ to be a set such that for all $x \in \Omega$, the intersection $S(x) \cap Z$ contains a single element, which we do by choosing a single element from each of the sets $S(x)$. It follows that $\{T_q(Z)\}_{q \in \mathbb{Q}}$ forms a countable partition of Ω and hence

$$1 = \mathbb{P}(\Omega) = \mathbb{P}\left(\bigcup_{q \in \mathbb{Q}} T_q(Z)\right) = \sum_{q \in \mathbb{Q}} \mathbb{P}\left(T_q(Z)\right) = \sum_{q \in \mathbb{Q}} \mathbb{P}\left(Z\right),$$

where we have used the rotation invariance property of $\mathbb{P}$. However, for all values $\mathbb{P}(Z)$ this equality is impossible. Hence, we conclude that we cannot define a rotation invariant probability measure on $\mathcal{E} := \{A \subseteq \Omega\}$, the set of all sets.

This means that we must restrict such probability measures to a "well-behaved" σ-algebra!

If you have completed all the questions on this assignment email `henry.reeve@bristol.ac.uk`.