IE6200: Engineernig Probability and Statistics

LAB 02: Random Variables and Distribution Functions

Prof. Mohammad Dehghani

Contents

1 Random Variable (RV)

A *random variable (RV)* is defined as the value of the given variable which represents the possible outcome of a probabilistic experiment. A random variable is denoted by uppercase letters such as X, Y, Z, etc. Specifically, the notation $X = x$ signifies the random variable X gets the particular value x (denoted by lowercase). There are two types of random variables explained in next section:

1. Discrete Random Variable
2. Continuous Random Variable

1.1 Discrete Random Variable

These variables can take only certain values within the range of integers.

1. Finite Discrete Variable: These variables can take only finite number of values within the range of integers. Example:
   - The number of heads in three tosses of a coin, where x = [0,1,2,3]
   - The number of games in the next World Series (best of up to seven games), where x = [4,5,6,7]
   - The number of boys in a randomly selected two-child family, where x = [0,1,2]
2. Infinite Discrete Variable: These variables can take infinite number of values within the range of integers. Example:
   - The number of arrivals at an emergency room between midnight and 6:00 a.m, where x = [0,1,2,. . .]
   - The number of applicants who have applied for a vacant position at a company, where x = [0,1,2,. . .]

1.2 Continuous Random Variable

These variables can take any value with the range of numbers. They have uncounctable number of values, meaning it is impossible to list all the possible outcomes even in an infinite amount of time. Examples of Continuous RV are:

- X ≡ The speed of a car in mph
- Y ≡ Lengths or areas of manufactured components in cmˆ2

Example: Consider a case when two dice are thrown together. Let X be a random variable that represents the sum of two dice when thrown together.

S = {‘(1,1)’, ‘(1,2)’, ‘(1,3)’, ‘(1,4)’, ‘(1,5)’,‘(1,6)’ ‘(2,1)’, ‘(2,2)’, ‘(2,3)’, ‘(2,4)’, ‘(2,5)’,‘(2,6)’ ‘(3,1)’, ‘(3,2)’, ‘(3,3)’, ‘(3,4)’, ‘(3,5)’,‘(3,6)’ ‘(4,1)’, ‘(4,2)’, ‘(4,3)’, ‘(4,4)’, ‘(4,5)’,‘(4,6)’ ‘(5,1)’, ‘(5,2)’, ‘(5,3)’, ‘(5,4)’, ‘(5,5)’,‘(5,6)’ ‘(6,1)’, ‘(6,2)’, ‘(6,3)’, ‘(6,4)’, ‘(6,5)’,‘(6,6)’}

where S is the sample space, a set of all possible outcomes or results of an experiment. The possible values of X are, therefore, either 2, 3, 4,. . . . or 12. we could find P(X = x), the probability that X takes on a value x. The possible values of X are, therefore, either 2, 3, 4,. . . . or 12. We could find P(X = x), the probability that X takes on a value x.

Table which contains some of the possible outcomes of the experiment

```
   X1 X2 Value.of.X      probs
1  1  1          2 0.02777778
2  2  1          3 0.02777778
3  3  1          4 0.02777778
4  4  1          5 0.02777778
```

```
5  5  1          6 0.02777778
6  6  1          7 0.02777778
```

2 Probability Distributions

Probability Distributions are functions that describe the likelihood of obtaining the possible values that a random variable can assume. It maps the member of a sample space to probability to their occurrence. The probabilities in the probability distribution of a random variable X must satisfy the following two conditions:

1. $0 \leq P(x) \leq 1$
2. $\sum P(x) = 1$

2.1 Probability Mass Function

The *probability mass function (PMF)* is a function that describes the probability distribution of a discrete random variable X. It creates a list of each possible value of X together with the probability that X takes that value in one trial of the experiment. Properties of PMF are:

1. $f(x) \geq 0$
2. $\sum_x f(x) = 1$
3. $\mathrm{P}(X = x) = f(x)$

PMF values for X = 2, 3, 4, 5,. . . ,11, 12, where X is a random variable that represents sum of two dice thrown together.

```
    Value.of.X f(x) = P(X=x)
1            2    0.02777778
2            3    0.05555556
3            4    0.08333333
4            5    0.11111111
5            6    0.13888889
6            7    0.16666667
7            8    0.13888889
8            9    0.11111111
9           10    0.08333333
10          11    0.05555556
11          12    0.02777778
```

2.2 Probability Density Function

The *probability density function (PDF)* is a function which maps the values of the variable in a certain interval to the probability of their occurrence. The probability density function characterizes the distribution of a continuous random variable. Properties of PDF are:

1. $f(x) \geq 0,\ for\ all\ x \in R$
2. $\int_{-\infty}^{\infty} f(x)\,dx = 1$
3. $\mathrm{P}[a \leq X \leq b] = \int_a^b f(x)\,dx$

Figure 1: PMF of X that represents sum of two dice when thrown together

2.3 Cumulative Distribution Function

A *cumulative distribution function (CDF)* is a function that the variables take a value less than or equal to the argument of the function. One of the advantages of the CDF is that it can be defined for any kind of random variable. CDF is denoted by a uppercase letter, like $F(x)$.

For a Discrete RV, cumulative distribution function of a random variable X can be defined as:

$$\boxed{F(x) = P(X \leq x) = \sum_{i=1}^{x} f(i)} \tag{1}$$

For a Continuous RV, cumulative distribution function of a random variable X can be defined as:

$$\boxed{F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)\, dt} \tag{2}$$

PMF and CDF for all the values of X together where X is a random variable that represents sum of two dice thrown together.

```
   x f(x) = PMF F(x) = CDF
1  2 0.02777778 0.02777778
2  3 0.05555556 0.08333333
3  4 0.08333333 0.16666667
4  5 0.11111111 0.27777778
```

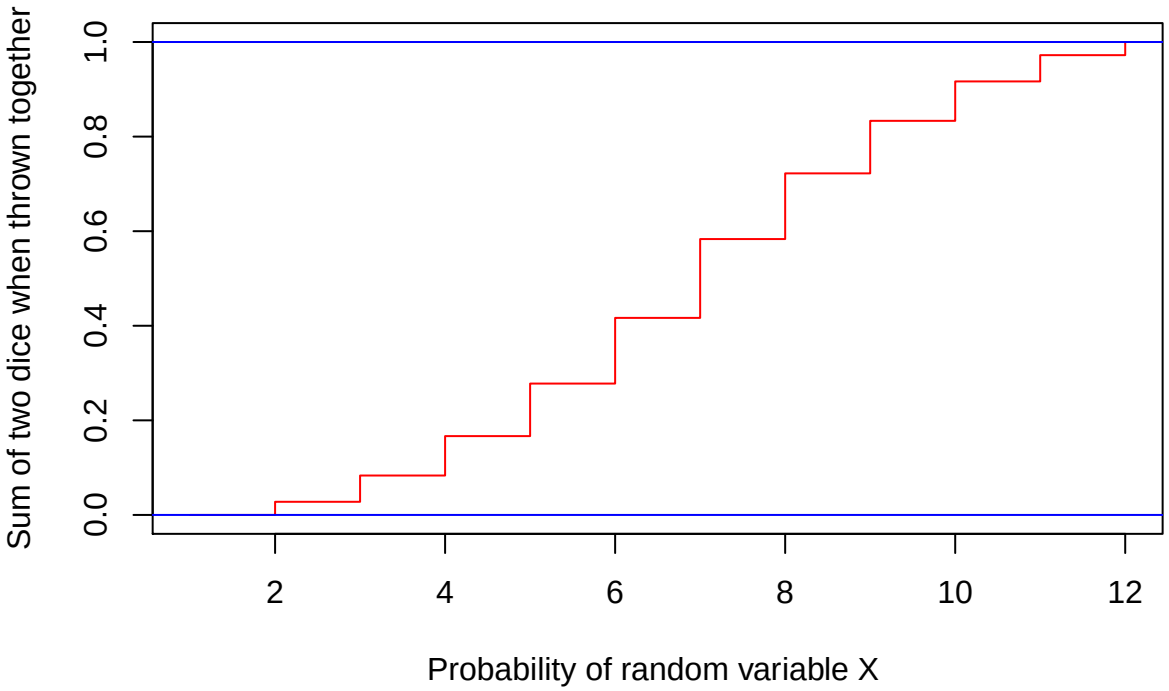

Figure 2: CDF of X that represents sum of two dice when thrown togetherr

```
5   6 0.13888889 0.41666667
6   7 0.16666667 0.58333333
7   8 0.13888889 0.72222222
8   9 0.11111111 0.83333333
9  10 0.08333333 0.91666667
10 11 0.05555556 0.97222222
11 12 0.02777778 1.00000000
```

2.4 Joint Probability Distribution

The *joint probability distribution* is a statistical measure which calculates the probability of two events occurring together and at the same point in time. This distribution is called bivariate distribution if we have only two random variables and multivariate distribution if we have more than two random variables.

Joint probability distribution is given by:

$$\boxed{f(x,y) = P(X = x, Y = y) \; for \; (x,y)} \tag{3}$$

Example: A dice is rolled twice. Let the random variables U and V be defined as following:

U $\equiv$ the maximum of the two rolls of the dice, and V $\equiv$ the sum of the two rolls of the dice

Find the Joint Probability Distributions.

A table for all the possible outcomes

```
    X1 X2 U  V      probs
1   1  1 1  2 0.02777778
2   2  1 2  3 0.02777778
3   3  1 3  4 0.02777778
4   4  1 4  5 0.02777778
5   5  1 5  6 0.02777778
6   6  1 6  7 0.02777778
7   1  2 2  3 0.02777778
8   2  2 2  4 0.02777778
9   3  2 3  5 0.02777778
10  4  2 4  6 0.02777778
11  5  2 5  7 0.02777778
12  6  2 6  8 0.02777778
13  1  3 3  4 0.02777778
14  2  3 3  5 0.02777778
15  3  3 3  6 0.02777778
16  4  3 4  7 0.02777778
17  5  3 5  8 0.02777778
18  6  3 6  9 0.02777778
19  1  4 4  5 0.02777778
20  2  4 4  6 0.02777778
21  3  4 4  7 0.02777778
22  4  4 4  8 0.02777778
23  5  4 5  9 0.02777778
24  6  4 6 10 0.02777778
25  1  5 5  6 0.02777778
26  2  5 5  7 0.02777778
27  3  5 5  8 0.02777778
28  4  5 5  9 0.02777778
29  5  5 5 10 0.02777778
30  6  5 6 11 0.02777778
31  1  6 6  7 0.02777778
32  2  6 6  8 0.02777778
33  3  6 6  9 0.02777778
34  4  6 6 10 0.02777778
35  5  6 6 11 0.02777778
36  6  6 6 12 0.02777778
```

Joint probability distribution table for U and V:

```
    V
U       2     3     4     5     6     7     8     9    10    11    12
  1 0.028 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  2 0.000 0.056 0.028 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  3 0.000 0.000 0.056 0.056 0.028 0.000 0.000 0.000 0.000 0.000 0.000
  4 0.000 0.000 0.000 0.056 0.056 0.056 0.028 0.000 0.000 0.000 0.000
  5 0.000 0.000 0.000 0.000 0.056 0.056 0.056 0.056 0.028 0.000 0.000
  6 0.000 0.000 0.000 0.000 0.000 0.056 0.056 0.056 0.056 0.056 0.028
```

3 Mean and Variance of a Random Variable

3.1 Mean or Expected Value

The *expected value* is the average value of repetitions of the same experiment it represents. The expected value of a random variable indicates its weighted average.

Expected value of a Discrete RV X is given by:

$$\boxed{\mu_x = E(x) = \sum_x xf(x)} \tag{4}$$

Expected value of a Continuous RV X is given by:

$$\boxed{\mu_x = E(x) = \int_{-\infty}^{\infty} xf(x)\,dx} \tag{5}$$

3.2 Variance of a Random Variable

The *variance of a random variable* is a measure of spread for a distribution of a random variable that determines the degree to which the values of a random variable differ from the expected value.

Let X be a random variable with probability distribution f(x). The variance of random variable X if is discrete:

$$\boxed{\sigma_x^2 = [E(X-\mu)^2] = \sum_x (x-\mu)^2 f(x)dx} \tag{6}$$

If random variable X is continuous:

$$\boxed{\sigma_x^2 = [E(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)\,dx} \tag{7}$$

We can also calculate variance using the formula:

$$\boxed{\sigma_x^2 = E(X^2) - \mu^2} \tag{8}$$

3.3 Covariance of two RVs

The *covariance of two RVs* is a statistic that tells you how correlated two random variables are. It is a measure of association between two variables. If two random variables are independent, then their covariance is zero.If their covariance is nonzero, then the value gives you an indication of how dependent they are.

The covariance of two RVs with means μ_x and μ_y respectively is given by

$$\boxed{\sigma_{xy} = \mu_{xy} - \mu_x \mu_y} \tag{9}$$

3.4 Correlation Coefficient

The *correlation coefficient* is the statistical measure that allows us to quantify the degree of correlation between two random variables X and Y. Correlation coefficient is a scale-free version of covariance.

The correlation coefficient between two random RVs with standard deviations σ_x and σ_y respectively is given by

$$\boxed{\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}} \tag{10}$$

Correlation Strength can range from -1 to 1 and can be interpreted as:

Correlation Strength (+/-)	Interpretation
.00 - 0.20	Weak or none
.20 - 0.40	Weak
.40 - 0.60	Moderate
.60 - 0.80	Strong
.80 - 1.00	Very Strong

4 Case Study: Bluebikes Boston

Bluebikes is a public bike share system available in Boston and it is an essential part of the transportation network of the city. Bluebikes provides monthly usage data of the network which is publicly available to download and analyze. In this case study, the raw data is for the usage of June 2018, August 2018, October 2018, December 2018 and January 2019. (Source: https://www.bluebikes.com/system-data)

The data has been filtered for only two stations, Christian Science Plaza (CSP) and Back Bay Station.

4.1 Attributes

- **Start Time:** This attribute provides when a user undocked a bike from the station and started their trip.
- **Stop Time:** Stop time tells us when a particular user docked back their bike and ended the trip.
- **Start Station Name:** It is the name of the station from where the trip began or in other words, where the bike was picked up from.
- **End Station Name:** It is the station where the bike was dropped off and the trip ended.
- **Month:** It is the month when the rider took the trip.
- **Date:** It is the date of the month when the trip was taken.
- **Hour:** Hour denotes at during which hour of the day trip started.
- **Day of Week:** It denotes what day of the week was the trip taken in numerical form. The week starts from Sunday i.e. 1 and ends on Saturday i.e. 7.

4.2 Required Packages

For this session, we will make use of tidyverse, lubridate and reshape2 packages.

- lubridate: This package is used to work with dates and times. It provides functionality to parse, extract, update, and perform algebraic operations on date time objects.
- reshape2: This package allows to restructure and aggregate data between wide and long formats

Let us go ahead and load these packages in the R session.

```
library(tidyverse)
library(lubridate)
library(reshape2)
```

You might need to install these packages if they are not installed already.

4.3 Importing the dataset

```
bikes_df <- read.csv('bluebikes.csv', header = TRUE, sep = ',')

head(bikes_df, 2) # structure of the dataset
```

```
   X           starttime            stoptime start.station.name
1 1 2018-06-01 00:14:10 2018-06-01 01:24:20           Back Bay
2 2 2018-06-01 00:17:59 2018-06-01 01:24:10           Back Bay
  end.station.name month date hour dayofweek
```

```
1          Back Bay     6    1    0          6
2          Back Bay     6    1    0          6
```

The goal is to now use this dataset to calcultate some basic metrics and answer the corresponding probability questions. The data will be transformed using the dplyr package of tidyverse as well as the reshape2 package.

4.4 PMF and CDF

In this part, the goal is to find out the number of bikes that are picked up from a particular station during a particular hour. Our test station is *Christian Science Plaza - Massachusetts Ave at Westland Ave*. We will use **R** to find out the number of bikes that were picked up from this station between 8:00 am and 9:00 am and then calculate the PMF and CDF of these numbers.

The first step is to select the columns that are required for this analysis. As we want to calculate how many bikes are being picked up from one station over the entire month for one hour, we need to select the start station, date and hour. We can select these columns using the select() function. This function keeps only the variables that are mentioned as its arguements.

```
# syntax for select() is to first mention the data you want to select the columns
# from, and then specify the columns

# selecting start station, month, date and hour
req_col <- select(bikes_df, start.station.name, month, date, hour)

head(req_col, 5)
```

```
   start.station.name month date hour
1           Back Bay     6    1    0
2           Back Bay     6    1    0
3                CSP     6    1    0
4                CSP     6    1    1
5                CSP     6    1    2
```

The second step is to now filter out the rows where the start station is *CSP* and hour is 8:00 o'clock. The function that does that is the filter() function and it works very similarly to how the select() function works. It returns the rows where the specified conditions are true.

```
# filtering only the required rows
req_row <- dplyr::filter(req_col, start.station.name == 'CSP' & hour == 8)

head(req_row, 5)
```

```
   start.station.name month date hour
1                CSP     6    1    8
2                CSP     6    1    8
3                CSP     6    1    8
4                CSP     6    1    8
5                CSP     6    1    8
```

If there is an error while using the filter() function, it might be because there is another filter() function from another package. In this case, we need to explicity mention the filter() function from dplyr package is to be used.

The third step is to count how many bikes were picked up each day of the month. For this, we will need to use group_by() and summarise() functions. The group_by() function groups the similar values together and summarise() function summarises the grouped values in the manner that is specified into a single row. In order to get a count, the function that needs to be specified is n().

```
# grouping values by date and hour. although we don't need hour since we only
# filtered to 8:00 o'clock but this is how one would group by a more granular level
daily <- group_by(req_row, month, date, hour)

# creating a new column name count for the number of bikes
daily <- summarise(daily, count = n())

head(daily)
```

```
 # A tibble: 6 x 4
# Groups:   month, date [6]
  month  date  hour count
  <int> <int> <int> <int>
1     1     2     8     2
2     1     3     8     2
3     1     4     8     2
4     1     6     8     1
5     1     7     8     3
6     1     8     8     3
```

Now we know how many bikes were picked up each day during the 8:00 o'clock, the next step is to find out for how many days were the same number of bikes were picked up. For this, we will again use the group_by() and summarise() functions, but this time we will group it by count.

```
days <- group_by(daily, count)

days <- summarise(days, num_days = n())

head(days)
```

```
 # A tibble: 6 x 2
  count num_days
  <int>    <int>
1     1       20
2     2       19
3     3       17
4     4        8
5     5        8
6     6        7
```

In order to get the PMF of the bike pickup by days, we can simply divide the number of days a specific

number of bikes were picked up by the total number of days.

```
pickup_pmf <- round(days$num_days/sum(days$num_days),3)

pickup_pmf
```

```
 [1] 0.154 0.146 0.131 0.062 0.062 0.054 0.069 0.015 0.077 0.062 0.031
[12] 0.046 0.015 0.046 0.015 0.008 0.008
```

And lastly, in order to get the cdf, we can use the cumsum() function over PMF which will give us the cumulative sum.

```
pickup_cdf <- round(cumsum(pickup_pmf),3)

pickup_cdf
```

```
 [1] 0.154 0.300 0.431 0.493 0.555 0.609 0.678 0.693 0.770 0.832 0.863
[12] 0.909 0.924 0.970 0.985 0.993 1.001
```

CDF is greater than 1 due to rounding errors.

And to bring it all together, we will use the cbind() function.

```
csp_freq <- cbind(days, pickup_pmf = pickup_pmf, pickup_cdf = pickup_cdf)

csp_freq
```

```
    count num_days pickup_pmf pickup_cdf
1       1       20      0.154      0.154
2       2       19      0.146      0.300
3       3       17      0.131      0.431
4       4        8      0.062      0.493
5       5        8      0.062      0.555
6       6        7      0.054      0.609
7       7        9      0.069      0.678
8       8        2      0.015      0.693
9       9       10      0.077      0.770
10     10        8      0.062      0.832
11     11        4      0.031      0.863
12     12        6      0.046      0.909
13     13        2      0.015      0.924
14     14        6      0.046      0.970
15     15        2      0.015      0.985
16     16        1      0.008      0.993
17     18        1      0.008      1.001
```

Use these functions to try and find out the number of bikes that were picked up only on weekends for each one hour interval

All of this can be done in a single step as will which is more convenient and succint by using the *pipe* %>% operator, which basically means *then*. We will also use the mutate() function which is used to creating new variables. This function always adds the new columns to the end of the dataset.

```
csp_freq <- bikes_df %>%
  select(start.station.name, month, date, hour) %>%
  dplyr::filter(start.station.name == 'CSP' & hour == 8) %>%
  group_by(month, date, hour) %>%
  summarise(count = n()) %>%
  group_by(count) %>%
  summarise(num_days = n()) %>%
  mutate(pickup_pmf = num_days/sum(num_days)) %>%
  mutate(pickup_cdf = cumsum(pickup_pmf))

csp_freq
```

```
# A tibble: 17 x 4
   count num_days pickup_pmf pickup_cdf
   <int>    <int>      <dbl>      <dbl>
 1     1       20    0.154        0.154
 2     2       19    0.146        0.3
 3     3       17    0.131        0.431
 4     4        8    0.0615       0.492
 5     5        8    0.0615       0.554
 6     6        7    0.0538       0.608
 7     7        9    0.0692       0.677
 8     8        2    0.0154       0.692
 9     9       10    0.0769       0.769
10    10        8    0.0615       0.831
11    11        4    0.0308       0.862
12    12        6    0.0462       0.908
13    13        2    0.0154       0.923
14    14        6    0.0462       0.969
15    15        2    0.0154       0.985
16    16        1    0.00769      0.992
17    18        1    0.00769      1
```

Using ggplot(), we can create a visualization for the PMF table

```
ggplot(csp_freq, aes(count, pickup_pmf)) +
  geom_bar(stat="identity", fill="steelblue")+
  theme_bw() +
  labs(x = 'Bikes Picked Up', y = 'Pickup Probability') +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_continuous("csp_pickup", labels = as.character(csp_freq$count),
                     breaks = csp_freq$count)
```

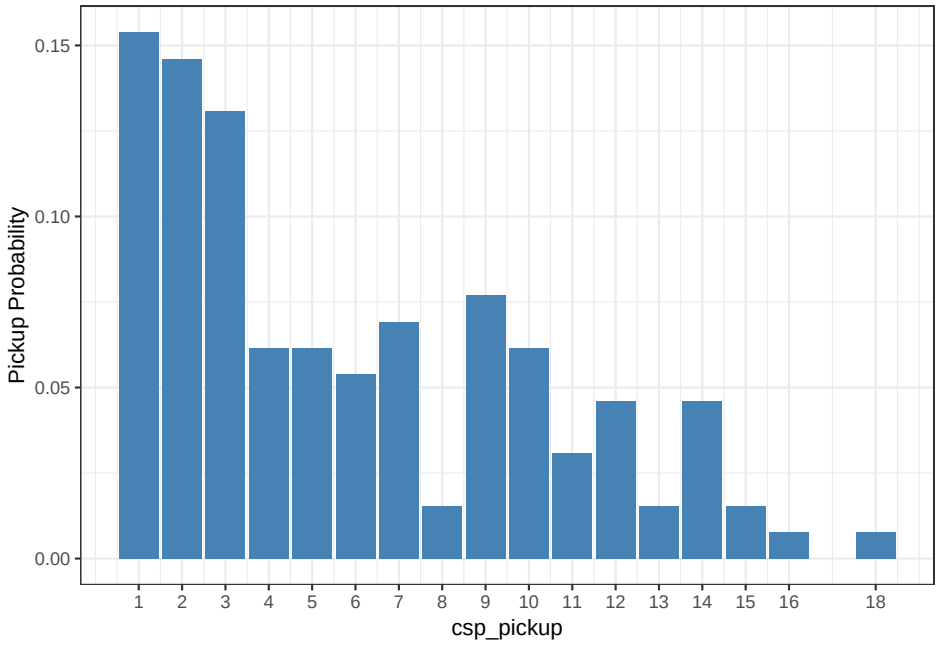

Figure 3: Bike Pickup PMF

4.5 Expected Value

In order to calculate the expected value of the number of bikes being picked up, we can use the weighted.mean() function. The first arguement for this function is the object containing values whose weighted mean is to be calculated and second are weights for those values.

```
weighted.mean(csp_freq$count, csp_freq$pickup_pmf)
```

```
[1] 5.861538
```

4.6 Joint Probability

To get an idea of how frequently are two stations used during a particular hour, we can make use of joint probability distribution. In this section, we will look at the bikes that were picked up from the CSP station **or** dropped off at the Back Bay station between 8:00 am and 9:00 am and then create a frequency table that determines number of days that a certain number of bikes were used from these two stations.

The first step is to create a frequency table for the Back Bay station similar to the one earlier, the only difference being Back Bay will be used as the end station.

```
bb_freq <- bikes_df %>%
  select(end.station.name, month, date, hour) %>%
  dplyr::filter(end.station.name == 'Back Bay' & hour == 8) %>%
  group_by(month, date, hour) %>%
  summarise(count = n()) %>%
  group_by(count) %>%
  summarise(num_days = n()) %>%
  mutate(pickup_pmf = num_days/sum(num_days)) %>%
```

```
  mutate(pickup_cdf = cumsum(pickup_pmf))

bb_freq
```

```
  # A tibble: 26 x 4
    count num_days pickup_pmf pickup_cdf
    <int>    <int>      <dbl>      <dbl>
  1     1       15     0.108       0.108
  2     2       12     0.0863      0.194
  3     3       10     0.0719      0.266
  4     4        7     0.0504      0.317
  5     5        3     0.0216      0.338
  6     6        8     0.0576      0.396
  7     7        2     0.0144      0.410
  8     8        3     0.0216      0.432
  9     9        3     0.0216      0.453
 10    10        5     0.0360      0.489
  # ... with 16 more rows
```

Now using the outer() function, we can create a frequency matrix. This function takes three arguements, first two being the arrays that need to go into the frequency matrix, and third what function to be used to combine the individual elements, which in our case in summation. We can also rename the columns and rows using colnames() and rownames() functions respectively. The count for the number of bikes picked up or dropped off is limited between 6 and 15.

```
joint_freq <- outer(csp_freq$num_days[6:15], bb_freq$num_days[6:15], FUN = "+")
rownames(joint_freq) <- csp_freq$count[6:15]
colnames(joint_freq) <- bb_freq$count[6:15]

joint_freq
```

```
     6  7  8  9 10 11 12 13 14 15
 6  15  9 10 10 12 12 12 16 11 11
 7  17 11 12 12 14 14 14 18 13 13
 8  10  4  5  5  7  7  7 11  6  6
 9  18 12 13 13 15 15 15 19 14 14
 10 16 10 11 11 13 13 13 17 12 12
 11 12  6  7  7  9  9  9 13  8  8
 12 14  8  9  9 11 11 11 15 10 10
 13 10  4  5  5  7  7  7 11  6  6
 14 14  8  9  9 11 11 11 15 10 10
 15 10  4  5  5  7  7  7 11  6  6
```

In ordet to get the probabilities from the frequency, we can divide the frequency matrix by the sum of of its elements. We will also use the round() function to round the digits to 3 decimal places.

```
joint_prob <- round(joint_freq/sum(joint_freq),3)

joint_prob
```

```
        6     7     8     9    10    11    12    13    14    15
6   0.014 0.009 0.010 0.010 0.012 0.012 0.012 0.015 0.011 0.011
7   0.016 0.011 0.012 0.012 0.013 0.013 0.013 0.017 0.012 0.012
8   0.010 0.004 0.005 0.005 0.007 0.007 0.007 0.011 0.006 0.006
9   0.017 0.012 0.012 0.012 0.014 0.014 0.014 0.018 0.013 0.013
10 0.015 0.010 0.011 0.011 0.012 0.012 0.012 0.016 0.012 0.012
11 0.012 0.006 0.007 0.007 0.009 0.009 0.009 0.012 0.008 0.008
12 0.013 0.008 0.009 0.009 0.011 0.011 0.011 0.014 0.010 0.010
13 0.010 0.004 0.005 0.005 0.007 0.007 0.007 0.011 0.006 0.006
14 0.013 0.008 0.009 0.009 0.011 0.011 0.011 0.014 0.010 0.010
15 0.010 0.004 0.005 0.005 0.007 0.007 0.007 0.011 0.006 0.006
```

The next step is to create a scatterplot to get a visual representation of the frequency table. To do so, we will first need to reshape the frequency matrix. This can be done so using the melt() function which converts the data from a wide to a long format. The function is very simple and only needs the object that needs to be reshaped as its arguement.

```
joint_df <- melt(joint_freq)
colnames(joint_df) <- c('csp_pickup', 'bb_dropoff', 'frequency')

head(joint_df, 10)
```

```
    csp_pickup bb_dropoff frequency
1            6          6        15
2            7          6        17
3            8          6        10
4            9          6        18
5           10          6        16
6           11          6        12
7           12          6        14
8           13          6        10
9           14          6        14
10          15          6        10
```

Now, to create a scatterplot, we will again make use of ggplot() function along with geom_point() as the geometric object.

```
ggplot(data = joint_df,aes(x=csp_pickup, y=bb_dropoff)) +
  geom_point(aes(size = frequency, color = frequency)) +
  labs(x = 'CSP Pickup', y = 'BB Dropoff') +
  scale_x_continuous("csp_pickup", labels = as.character(joint_df$csp_pickup),
                     breaks = joint_df$csp_pickup) +
  scale_y_continuous("bb_dropoff", labels = as.character(joint_df$bb_dropoff),
                     breaks = joint_df$bb_dropoff)
```

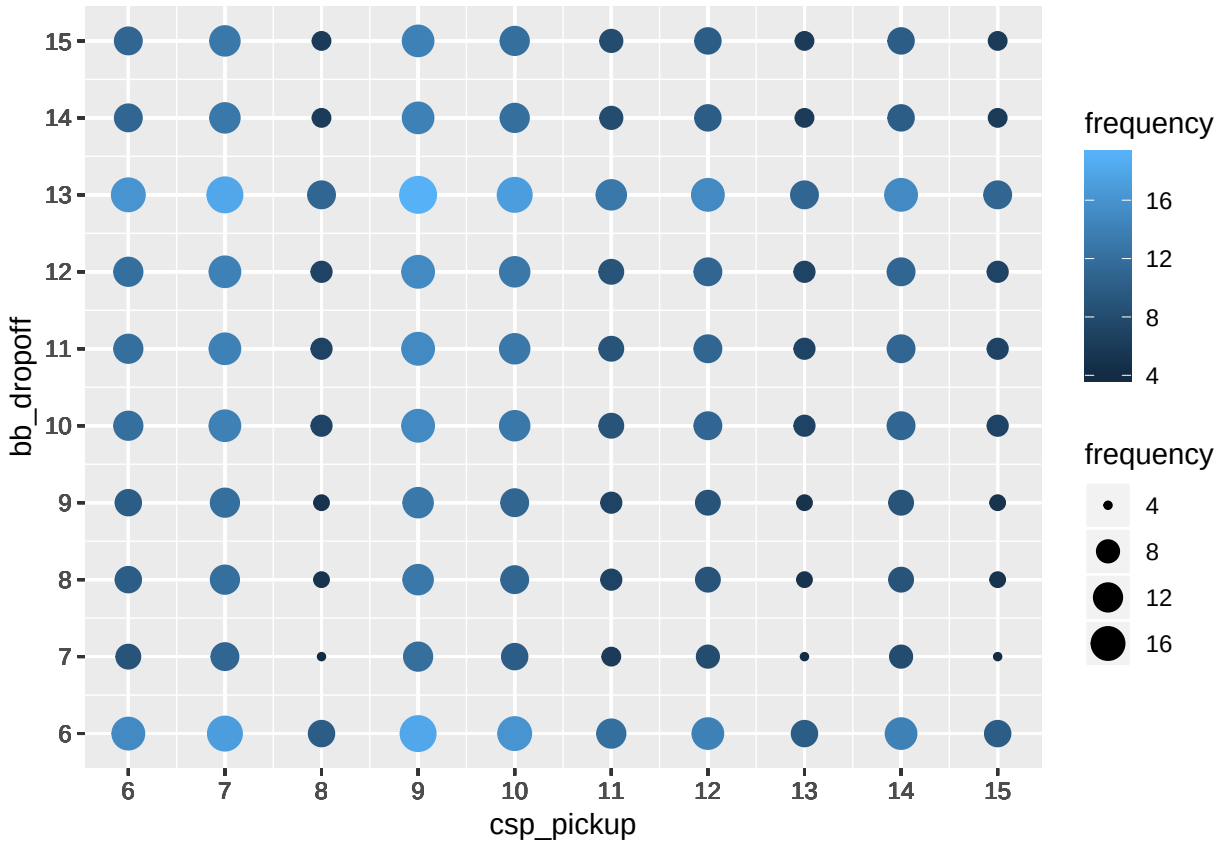

Figure 4: Frequency

4.7 Correlation Coefficient

In this part, we will try to calculate what is the correlation between a bike being picked up at one station and being dropped of at another station. The first step for this process is to create a data frame that tells us how many bikes were picked up or dropped off each day over of the month. We will again make use of the dplyr package functions to create such a data frame.

```
# data frame for the Christian Science Plaza stations
csp_8 <- bikes_df %>%
  select(start.station.name, month, date, hour) %>%
  dplyr::filter(start.station.name == 'CSP') %>%
  group_by(month = month, date) %>%
  summarise(count = n())

csp_8
```

```
 # A tibble: 153 x 3
# Groups:   month [?]
   month  date count
   <int> <int> <int>
 1     1     1    14
 2     1     2    25
 3     1     3    42
```

```
 4     1     4    40
 5     1     5    12
 6     1     6    32
 7     1     7    20
 8     1     8    33
 9     1     9    33
10     1    10    26
# ... with 143 more rows
```

```
# data frame for the Back Bay station
bb_8 <- bikes_df %>%
  select(end.station.name, month, date, hour) %>%
  dplyr::filter(end.station.name == 'Back Bay') %>%
  group_by(month = month, date) %>%
  summarise(count = n())

bb_8
```

```
 # A tibble: 153 x 3
# Groups:   month [?]
   month  date count
   <int> <int> <int>
 1     1     1    21
 2     1     2    46
 3     1     3    51
 4     1     4    52
 5     1     5     8
 6     1     6    18
 7     1     7    44
 8     1     8    38
 9     1     9    53
10     1    10    55
# ... with 143 more rows
```

Now, we will use the cor() function to calculate what the correlation coefficient is. The arguement for this functions are two objects for which we want to calculate the coefficient.

```
cor(csp_8$count, bb_8$count)
```

```
[1] 0.8217588
```

As we can see, pickup and dropoff of bikes are strongly correlated with a value of 0.82. This does not imply that the bikes that are being picked up are being dropped off at the station being analysed, but it can be inferred that on there is a good chance that a bike picked up from Christian Science Plaza maybe dropped off at Back Bay.