Humboldt Universitat zu Berlin

Livability in Berlin Districts: Comparative Analysis

Malgorzata Paulina Olesiewicz 598939
Qi Wu 601623
Aleksandra Kudaeva 599943

Project GitHub: https://github.com/Olesiewitch/SPL_BerlinDst

May 23, 2019

Contents

List of Figures

List of Tables

1 Motivation

Berlin is one of the fastest growing cities in Europe. After the fall of Berlin Wall, there was not much happening here until the beginning of 2010s when the city began quickly developing. Once "poor but sexy", now Berlin became *the place to be* for artists, entrepreneurs and young people at large. Unfortunately, Berlin was not quite ready for its success. Consequently, the city is facing a significant shortage of housing (Schultheis 2019) and for any newcomer looking for accommodation can be quite a daunting experience. So where is the best place to live in Berlin? Every district of Berlin has been its own town and today each of them have its own Senate, overseen by the Mayors office. It is well know that there is quite a high level of competition between districts. To compare the livability of Berlin districts, the Livability Index was created. The report will answer following questions:

- Which of the Berlin districts is the most livable?
- How well each of the districts perform in different aspect of livability?
- Does livability of the districts corresponds to average rent?
- Is there any difference between West and East Berlin in terms of livability?

2 Methodology

The concept of "livability" lacks clearly defined theoretical framework in the literature (Giap et al. 2014). This complex measure is often used in empirical studies, to rank and compare different geographical areas in terms of the "quality of life" or "well-being" offered to their inhabitants. The last three decades of research on happiness has shown that the well-being should be measured with social, economical and also subjective to the given group set of indicators (Diener & Suh 1997). In cities, livability is often defined in terms of greenery, security, economic stability, infrastructure and night life (Unit 2011). A number of city rankings have been published by institutions such as C40 or World Economic Forum (Okulicz-Kozaryn 2013). However, for the purpose of this study, each of the existing methodologies required its adaptation to the city district level.

2.1 Livability Standards in Cities

The "Methodology for Collection and Computation of Livability Standards in Cities" developed by Government of India breaks down the Index calculation to its primary components (Ministry of Urban Development 2017). The methodology divides the livability standards indicators in the cities into four pillars: 1. Institutional, 2. Social, 3. Economic and 4. Physical. For each pillar "Sub-Indexes" were calculated and weighted respectively with 25%, 25%, 5% and 45% towards final Index. To calculate the sub-Indexes, respective normalized indicators were added.
A total of 79 indicators are used for the calculation of the Livability Index. To eliminate the issue of difference in units and scales, each of the indicators was normalized using the equation 1 (variable has positive impact on the Index) or equation 2 (negative impact).

$$I_i^+ = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

$$I_i^- = \frac{x_{max} - x_i}{x_{max} - x_{min}} \tag{2}$$

2.2 Adaptation of the methodology to districts of Berlin

The Berlin Districts Livability Index adapts the described in the section 2.1 methodology for the purpose of district comparison. Taking into account current socio-economic situation and potential measurable differences between districts, the Index is based on four pillars following the argumentation:

1. Physical: Berlin is facing rapid population increase and shortage of housing (Schultheis 2019). It becomes increasingly difficult to find affordable housing in districts with good infrastructure. Nonetheless, availability of housing is more pressing issue than the infrastructure.

2.Social: Berlin is well known for night life, which brought more than 1.5 billion to the city in 2018 (thelocal.de 2019). Besides "cool" cultural side, other indicators such as education, health, safety and social support need to be considered. This is the biggest pillar of the Index with 16 indicators.

3. Economic: Rapidly growing start-up scene in Berlin and its reputation of being "Silicon Valley of Europe" brings more and more business to the city (Cooke & Zaby 2015). Consequently, there is a big competition between districts to attract big players planning to move here (e.g. Google) (Kuhn 2019).

4. Environmental: Due to the growing air pollution threat in Berlin (tagesspiegel 2019) and concerns regarding the environmental health in the cities, the Institutional Category of the Index from section 2.1 has been replaced by Environmental. The environmental health and greenery of the districts is considered.

The methodology for the Berlin District Livability Index is illustrated by Figure 1. Complete list of the indicators and their description can be found in Annex A.

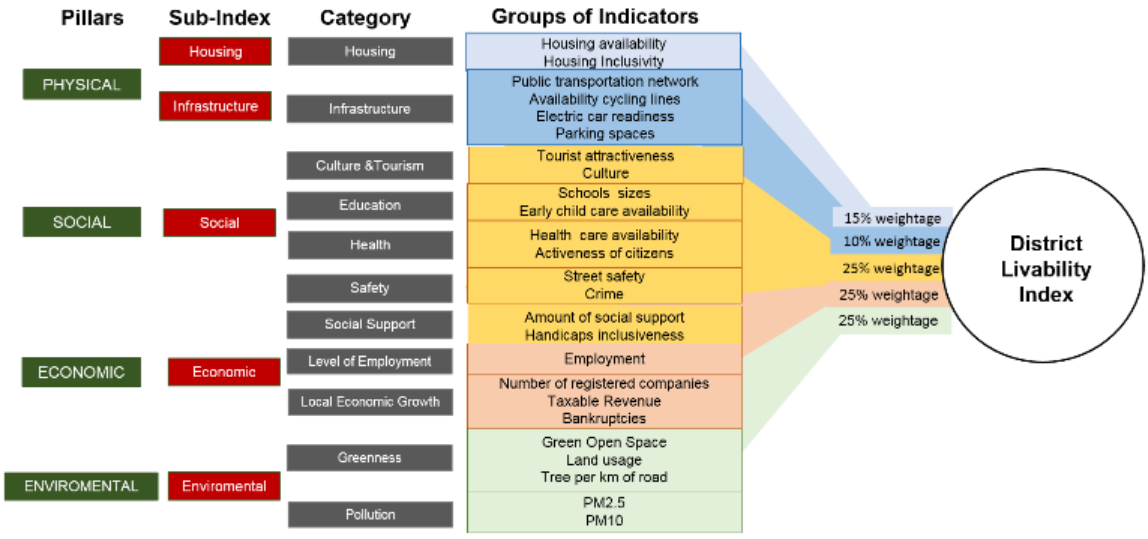

Figure 1: *Berlin Districts Livability Index Calculation Methodology*

3 Data Preparation

Accessing and processing the data for Livability Index Calculation was a major challenge of this project. The total of 46 variables have been collected in the final data set SPL_BerlinDst_Data_Desc.csv. This section describe the process of acquisition and preparation of the data from following sources:

- https://www.statistik-berlin-brandenburg.de
- https://daten.berlin.de
- https://de.statista.com
- https://www.berlin.de/
- https://berlin.kauperts.de/
- https://berlin.kauperts.de/
- https://data.technologiestiftung-berlin.de/dataset/bezirksgrenzen
- https://opendata-esri-de.opendata.arcgis.com/datasets/
- https://fbinter.stadt-berlin.de/fb/index.jsp
- http://www.stadtentwicklung.berlin.de//geoinformation/fis-broker/
- https://www.berlin.de/ba-lichtenberg/auf-einen-blick/buergerservice/gesundheit/artikel.596005.php

3.1 Accessing and Preparing the data

3.1.1 Statistik Berlin Brandenburg Data

The first data source, which was used, is the website of the Office of Statistics Berlin-Brandenburg, where data is stored according to the topics under different sub-pages.
The required data has been directly read into R from 14 sub-pages of the Regional section of the website. The changing component of the sub-pages URLs is not following any pattern, therefore all the URL addresses have been stored into a list (lstURL, Quantlet Data_Prep_1 lines 119:180). To obtain data stored under the URL addresses, the GetDataUnderURL function has been written. The function uses the components of *rvest* package, which was designed for extracting information from the websites. The function is structured as follow:

```
GetDataUnderURL = function(URL){
    # Function returns the data table from under the URL
    # address. Firstly, the URL address is read (webpage).
    # Secondly, the data of the type "table" is extracted
    # from the HTML document with the CSS selector
    # (?html_nodes). The data table is read, stored and
    # returned as a table.
    # Args:
    #   URL: URL address under which the required data are
    #        stored. The table which contains the data must
    #        be the first table [[1]] on the page.
    # Returns:
    # Matrix with the data stored in the first table under from
```

```
    # the URL address
    webpage = read_html(URL)
    table = html_table(html_nodes(webpage, "table")[[1]],
                       fill = TRUE,trim = TRUE)
    return(table)
}
```

The GetDatafromURL function was applied across all the URLs with the lapply function and the list of obtained tables (tblLst) was split into individual data frames.

```
tblLst = lapply(lstUrl, GetDataUnderURL) %>%
    list2env(envir = .GlobalEnv)
```

The obtained data tables have not been structured in traditional way and consequently the column names do not correspond to respective columns. Therefore hard-cording of column numbers was required when assigning data to specific variable vectors. To avoid potential mistakes, the code comments specify exactly, which data should be acquired (see Data_Prep_1 Quantlet lines 190:227). In total, data of 31 variables have been acquired from the website and stored into the data frame (in code WbsDt).The acquired data has been mutated to appropriate format.

```
# Merge the data vectors together into one data frame and split
# the number and the name of the district in X1 into two separate
# columns

wbsDt = data.frame(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,
                   X14,X15,X16,X17,X18,X19,X20,X21,X22,X23,X24,
                   X25,X26,X27,X28,X29,X30, X31,
                   stringsAsFactors = FALSE) %>%
    separate(col = X1,into = c("X0", "X1"),sep = "␣") %>%
    mutate_at(vars(X1), DistricToFullName) %>%
mutate_at(vars("X2":"X31"),DataToNumeric)
```

X1 is the Berlin districts names, which are long (often abbreviation are used), contain special signs and very often are not read-in properly from the data source. To make sure that each prepared data set is naming Berlin Districts in the same way (to allow easy merging), the DistrictToFullName function has been written and applied:

```
DistricToFullName = function (column){
    # Function re-names the districts of Berlin with their full
    # names without special signs. The function identifies the
    # district by three #or four letters of its name.
    # Args:
    #      column: vector or a column containing names of Berlin
    #      districts.Name can have any form, special signs can be
    #      use and additional information can be added to it.
    #      Only first 3-4letters need to be correct.
    #Returns:
    #Vector of replaced district names by its official names
```

```
    column[grepl("mit",column, ignore.case = TRUE)] = "Mitte"
    column[grepl("fri",column,
                 ignore.case = TRUE)] =
                 "Friedrichshain-Kreuzberg"
    column[grepl("pank",column,ignore.case = TRUE)] = "Pankow"
    column[grepl("mar",column,
                 ignore.case = TRUE)] = "Marzahn-Hellersdorf"
    column[grepl("char",column,
                ignore.case = TRUE)] =
                "Charlottenburg-Wilmersdorf"
    column[grepl("spa",column,ignore.case = TRUE)] = "Spandau"
    column[grepl("ste",column, ignore.case = TRUE)] = "Steglitz-Zehlendorf"
    column[grepl("tem",column, ignore.case = TRUE)] = "Tempelhof-Schoneberg"
    column[grepl("trep",column, ignore.case = TRUE)] = "Treptow-Kopenick"
    column[grepl("neu",column,ignore.case = TRUE)] = "Neukolln"
    column[grepl("lich",column,ignore.case = TRUE)] = "Lichtenberg"
    column[grepl("rein",column,ignore.case = TRUE)] = "Reinickendorf"

    return(column)
}
```

To make sure that the format of the acquired data is numeric the DataToNumeric function has been written and applied to all the variable columns (X2:X31):

```
DataToNumeric = function(column){
    # Function cleans up data in chosen column by replacing ","
    # with "." as a decimal symbol, removes empty space between
    # numbers and converts the data to numeric type.
    # Args:
    #     column: name of the column in the dataframe which
    #     values suppose to be cleaned up and converted to
    #     numeric type
    # Returns:
    # Vector with numeric values of converted data
    commatodot = (gsub(",", ".", column))
    num = as.numeric(gsub("[[:space:]]", "", commatodot))
    return(num)
}
```

Finally, Sport Data have been downloaded from the website as an excel file *SB_B05-01-00_2018j01_BE.xlsx* on the 15th of December. The process of selection and cleaning the data from the file executed in Quantlet Data_Prep_1 (lines 275:336) is pretty straight forward and therefore requires no further explanation.

3.1.2 Daten Berlin Data

The Berlin Open Data Platform has been the sources of two data sets for the project:

1. public transportation stops.csv
2. FallzahlenHZ 2012-2017.xlsx

To obtain the first data set, the whole zip file of Fahrplandaten needs to be downloaded. Only *stops.txt* file was used for the project. The text file was converted into a csv file with a full name as listed.The public transportation stops data contains a list of all stops in Berlin and Brandenburg together with their geographical coordinates. To obtain the total number of public transportation stops in each district, two steps approach was needed. Firstly, the data regarding geographical borders of each district has been read into R from the *bezirksgrenzen.kml* file (in the code dstrBdrd). In the file, each district has been assigned polygon(s) with its coordinates. The districts names have been assigned to the polygons accordingly:

```
dstLst = list( rein = dstrBrd [[1]],  # Reinickendorf
               char = dstrBrd[[2]],  # Charlottenburg-Wilmersd.
               trep = dstrBrd[[3]],  # Treptow-Kopenic
               pank = rbind(dstrBrd[[4]], dstrBrd[[5]], dstrBrd[[6]],
                            dstrBrd[[7]], dstrBrd[[8]]), # Pankow
               neu  = dstrBrd[[9]],  # Neukolln
               lich = dstrBrd[[10]],  # Lichtenberg
               marz = dstrBrd[[11]],  # Marzahn-Hellersdorf
               spa  = dstrBrd[[12]],  # Spandau
               steg = dstrBrd[[13]],  # Steglitz-Zehlendorf
               mit  = dstrBrd[[14]],  # Mitte
               fri  = dstrBrd[[15]],  # Friedrichshain-Kreu.
               tem  = dstrBrd[[16]])  # Tempelhof-Schoneberg
```

Secondly, the number of public transportation stops have been counted in each listed district (list in the code dstLst) using NrofStops function:

```
NrofStops= function(district){
    # Function checkes based on the geografical coordinates, how
    # many public transportation stops (in file bsStp) are
    # in the chosen Berlin district and counts their number.
    # Args:
    # district: names of the polygon file with the coordinates of
    # the given Berlin district.The longitude coordinates  must
    # be the first column of the file, the latitude the second
    # column.
    # Returns:
    # Number of public transportation stops in the given district
    # or on its border ( in whichstops function 1 = in the
    # polygon, 2 = on the border of the polygon ,0 = not in the
    # polygon).
    whichstops = point.in.polygon(bsStp$stop_lon, bsStp$stop_lat,
                                  district[, 1],district[, 2])
    nrstops = length(whichstops[whichstops != 0])

    return(nrstops)
}
# Apply NrofStops function to all the districts:
bsLst = lapply(dstLst, NrofBusStops)
```

The second used data set from the Berlin Open Data website is the criminal atlas of Berlin, which has accessed on the 27th of December 2018. The process of selection and cleaning the data from the file is executed in Data_Prep_1 Quantlet(lines 387:417).

3.1.3 Mobility Report Data

The number of parking spaces data has been acquired from the 2013 Berlin Mobility Report(page 32) and had to be hard-coded into R (lines 418:455). For the districts, which has not been mentioned in the report, the average value of observed data has been used.

3.1.4 de.statista Data

The last two data sets for this part of data cleaning has been acquired from the de.statista website, which a Statistic Database. Two excel files have been downloaded on November 20th 2018.

- *statistic_id652680_strassenbaeume-in-berlin-nach-bezirken-2017.xlsx*
- *statistic_id652716_oeffentliche-gruenflaechen-in-berlin-nach-bezirken-2017.xlsx*

The first data set is used to obtain number of trees per km of road data, whereas second, the number of Ha of Green Space. The process of selection and cleaning the data from the file is executed in Data_Prep_1 Quantlet(lines 456:509). In the final stage of this part of data acquisition, all the acquired data has been merge according to the districts names. The final output has been written into a *SPL_BerlinDst_Data_Prep_1.csv* file.

```
FnlDt = merge(wbsDt,spMbDt, by.y = "District") %>%
    merge(.,clbsDt, by.y = "District") %>%
    merge(.,crmDt,by.y = "District") %>%
    merge(.,prkDt,by.y = "District")%>%
    merge(.,bsStpDt,by.y = "District") %>%
    merge(.,trDt,by.y = "District") %>%
    merge(.,grSpDt,by.y = "District")  # merge all the data sets

write.csv2(FnlDt, "SPL_BerlinDst_Data_Prep_1/SPL_BerlinDst_Data_
Prep_1.csv")

# Read in best with:
# read.csv("SPL_BerlinDst_Data_Prep1/SPL_BerlinDst_Data_Prep_1.csv",
# sep = ";", dec = ",", row.names = 1, stringsAsFactors = FALSE)
```

3.2 Berlin Districts Data

See quantlet for Data preparation for this part: Data_Prep_2 Quantlet

3.2.1 Web Scraping and Sub-Districts in Berlin Districts

There are 12 districts in Berlin, and under each districts there are sub-districts, in total 96. Because some of our data only have the sub-districs information, so we create a dataframe to store the sub-districts in each district. With help of a web scraping package *rvest*. The R code is shown below:

```
# Ortsteil is the German word for subdistricts
Oteil = read_html(paste0("https://de.wikipedia.org/wiki/",
                         "Liste_der_Bezirke_und_Ortsteile_Berlins"))
```

```
Oteil = Oteil %>%
    html_nodes("table")%>%
    .[[3]] %>%  #the third table in this website give us a dataframe
    # lists all Orststeile in Berlin and their corresponding districts
    html_table()

# replace all the german letters with a predefined function called Replace
#detail would be shown in the next subsection3.3.2.
Oteil$Bezirk = Replace(Oteil$Bezirk)
Oteil$Ortsteil = Replace(Oteil$Ortsteil)

# create a dataframe OteilD which contains Ortsteil in 12 districts
OteilD = data.frame(matrix(ncol = length(Dstc$District),
                           nrow = 20))
# nrow can be any number bigger than the maxi number of Ortsteil in district

j = 1  # start with the first observation in dataframe Oteil

for (i in 1:length(Dstc$District)){  # a loop with number of districts

    j = j  # every loop start with current j value where last while loop ends
    k = 1  # k start from 1 for every column in the new dataframe OteilD

    while(Oteil[j,'Bezirk'] == Dstc[i,"District"]){
        OteilD[k,i] = Oteil[j,'Ortsteil']
        k = k+1
        j = j+1
    }
}
colnames(OteilD) = Dstc$District
```

3.2.2 Post codes in Berlin Districts

As post code were given in some data-set we found , for assigning some number of certain variable to each districts, we need to subtract all postcode in each district. R code is shown as below:

```
Pscd = read_excel(paste0("SPL_BerlinDst_Data_Prep_2/",
                         "ZuordnungderBezirkezuPostleitzahlen.xls"))
# create 12 vector to store the Post code for each district
# 3:12 are the columns with post code infomation
Dstc01 = as.numeric(unlist(as.list(Pscd[6:8,3:12])))
Dstc02 = as.numeric(unlist(as.list(Pscd[10:11,3:12])))
# ignore warning of NA since it doesn't effect our use of the data
Dstc03 = as.numeric(unlist(as.list(Pscd[13:15,3:12])))
Dstc04 = as.numeric(unlist(as.list(Pscd[17:20,3:12])))
Dstc05 = as.numeric(unlist(as.list(Pscd[22:23,3:12])))
Dstc06 = as.numeric(unlist(as.list(Pscd[25:27,3:12])))
Dstc07 = as.numeric(unlist(as.list(Pscd[29:32,3:12])))
Dstc08 = as.numeric(unlist(as.list(Pscd[34:36,3:12])))
Dstc09 = as.numeric(unlist(as.list(Pscd[38:39,3:12])))
Dstc10 = as.numeric(unlist(as.list(Pscd[41:42,3:12])))
Dstc11 = as.numeric(unlist(as.list(Pscd[44:45,3:12])))
```

```
Dstc12 = as.numeric(unlist(as.list(Pscd[47:48,3:12])))

Pscd= as.data.frame(cbind(Dstc01,Dstc02,Dstc03,Dstc04,
                          Dstc05,Dstc06,Dstc07,Dstc08,
                          Dstc09,Dstc10,Dstc11,Dstc12))

# compare post code of each Charging station, assgin a district to each
DsLd = rep(0,length(CgstN$Pcd))

for(i in 1:length(CgstN$Pcd)){
    for(j in 1:12){
        if(CgstN$Pcd[i] %in% Pscd[,j]){
            DsLd[i]=j
        }
    }
}
# check if all charging stations is assigned to a district
(DsLd !=0) == length(CgstN$Pcd)
# the result should all be False
```

3.2.3 Merging data and Writing csv file

Restaurant data was obtained by web-scraping and comparison to sub-districts. Cycling length, Street crossing and NR.of doctors are obtained from internet web-page:

1. http://www.stadtentwicklung.berlin.de//geoinformation/fis-broker/
2. https://www.berlin.de/ba-lichtenberg/auf-einen-blick/buergerservice/gesundheit

and data are merged to a csv file as below:

```
Nr = 1:12
QiDt = data.frame(Nr,Dstc$District, Nrct, Rest, Cyll, Nrdr, Nrsc)
colnames(QiDt) = c("Nr",
                   "District",
                   "Charging",
                   "Restaurants",
                   "Cycle",
                   "Doctors",
                   "Crossings")

QiDt = as.data.frame(QiDt); QiDt

write.csv(QiDt,"SPL_BerlinDst_Data_Prep_2.csv")
```

3.3 Air-Pollution

Preparation of air-pollution data required obtaining additional data and usage of several approximations. The whole process of data preparation can be divided in three parts:

1. Downloading the data from different sources

2. Data preparation: cleaning and reformatting

3. Aggregating and merging datasets

3.3.1 Loading required data

Air-pollution data was downloaded from https://fbinter.stadt-berlin.de/fb/index.jsp in xls format. Initial column names are poorly adapted for R processing (e.g. "PM10-Belastung (berechnetes Jahresmittel [mg/m^3]) 2015") as they are too long and contain special symbols. For the purpose of further analysis all the columns were renamed according to the correspondence stated in a matching table (*matching.csv*).

```
# Read air-pollution data from excel
ap15 = read_excel("./SPL_BerlinDst_Data_Prep_3/Air_Pollution_2015.xls",
                  sheet = 1)

# Original names are too long and contain special symbols and spaces
mtch = read.csv2("./SPL_BerlinDst_Data_Prep_3/matching.csv",
                 sep = ";",
                 stringsAsFactors = FALSE)  # Table with short names

names(ap15) = mtch$new[match(names(ap15), mtch$old)]  # Rename variables
```

The downloaded table consisted of 12,374 rows containing values of different air-pollution indicators and aggregated index values for section of main Berlin Streets (1,238). In our research we use PM10 and PM25 indicators because as they are regulated by the European Union and have clearly determined thresholds Agency (2016). Average value of PM10 pollution is 20.6 mg/m^3, and PM25 is 14,3 mg/m^3 (in 2015), which is below dangerous threshold. Statistics were calculated by means of the script provided below:

```
# descriptory statistics of air pollution data table
ap15 %>% summarize(Sections = n(),                  # Number of street sections
                   Streets  = n_distinct(Street),  # Number of unique streets
                   avg_PM10 = mean(PM10_yearly),   # Average value of PM10
                   avg_PM25 = mean(PM25_yearly))   # Average value of PM25
```

The raw data does not indicate in what city district a street is located. To merge the air pollution and previously described data, we had to downloaded additional information. We scraped an online table (correspondence table) from https://fbinter.stadt-berlin.de/fb/index.jsp to obtain street names and their corresponding city districts. As there was no option to download information for all districts in a single table, we had to use *rvest* package for each subpage. The R-script for this procedure is provided below:

```
for (i in 1:dim(dstr)[1]) {
    # Generate a link to data for all the districts and sub-districts
    link=paste0("https://berlin.kauperts.de/Bezirke/",
                dstr$District[i],
                "/Ortsteile/",
                dstr$Sub.district[i],
```

```
                "/Strassen")

    # Download the data from the web-page with generated link
    webpage = read_html(link)
    tbls    = html_nodes(webpage, "table")
    tab     = html_table(tbls)[[1]]

    # Add columns for district and sub-district
    tab$District    = dstr$District[i]
    tab$SubDistrict = dstr$Sub.district[i]

    # Add created table to the table for the whole Berlin
    StrMtch = rbind(StrMtch, tab)
}
```

Downloaded table contains information on 13,399 streets from 12 districts of Berlin.

3.3.2 Data preparation

Another problem was different ways of writing street names, e.g. Adamstr./ Adam Str. and so on. The first step to unification of street names was replacements of German special symbols and bringing everything to the lower case. For that purpose the function *ReplaceUmlauts* was written:

```
#replace all the Umlauts by latin equivalents
ReplaceUmlauts = function(clmn){
    #Description: Replaces german special symbols and turns to lower case
    #Author: Aleksandra Kudaeva
    #Input:  column where you want to replace umlauts
    #Output: column without umlauts (lower case)

    clmn = tolower(clmn)  #all strings to lower case

    #check if at least one element of a vector has any umlauts in it
    #replaces umlauts until there are no one left
    while(any(grepl("  |  |  |  ",clmn)) == TRUE) {
        clmn %<>%
            sub("  ", "ae", .) %<>%
            sub("  ", "oe", .) %<>%
            sub("  ", "ue", .) %<>%
            sub("  ", "ss", .)
    }
    return(clmn)
}
```

The next step was to subtract the unique part of a street name (e.g. Adamstr. - Adam). The following code illustrates the work of the replacement function and substitution procedure for the air pollution dataset (an analogical procedure was performed for correspondence table):

```
# Street name formatting (in order to merge with street-index matching table)
ap15$str = ap15$Street %>%
    ReplaceUmlauts() %>%  # Replace umlauts and switch to lower case
```

```
    sub("str.$|str$|-strasse|strasse|-str.$", "", .) %>%  # Delete street ind.
    sub("ak␣|as␣|ad␣", "", .)  # Delete AK, AS, AD in the beginning
```

3.3.3 Aggregating and merging datasets

The first step is to calculate weighted average PM10 and PM25 pollution for all streets presented in the air pollution data.

```
# Find PM10 and PM15 weighted averages for each street (by length of str.section)
Pltn = ap15 %>%
    group_by(str) %>%
    summarize(L = sum(Length),
              MinPM10 = min(PM10_yearly),
              MaxPM10 = max(PM10_yearly),
              AvgPM10 = weighted.mean(PM10_yearly, Length),
              MinPM25 = min(PM25_yearly),
              MaxPM25 = max(PM25_yearly),
              AvgPM25 = weighted.mean(PM25_yearly, Length)) %>%
    arrange(str)
```

After we unified street names, tables are ready to be merged by street name:

```
# Merge averaged air pollution tables with
PltnM = merge(Pltn, StrMtch, by="str", all.x = TRUE, all.y = FALSE)
```


```
# Merge averaged air pollution tables with
PltnM = merge(Pltn, StrMtch, by="str", all.x = TRUE, all.y = FALSE)
```


As a result of merged tables, we could define districts and postal codes for 1172 streets. 51 streets were not found in the matching table for street names and postal codes and were excluded from the final data set.

Finally, we can calculate an average air pollution for each of 12 Berlin districts. To do so, we have to approximate air pollution of the streets which are included to more than one district as it was not possible to find any information on how streets were distributed among districts we assume equal distribution (e.g. if the street was included into 3 districts we divide it's total length by 3 and include each part to each district). The calculations described above were completed with R-script provided below:

```
# Some streets are allocated to more than one district
# We assume here that such streets are equally devided among all districts
# to which they belong

ap = PltnM %>%
    filter(!is.na(PLZ)) %>%  #filter NAs
    group_by(str) %>%
    mutate(DistrictLength = L/n()) %>% #appr. length of street in the district
```

```
    group_by(District) %>%
    summarize(PM10 = weighted.mean(AvgPM10, DistrictLength),  # av. PM10
              PM25 = weighted.mean(AvgPM25, DistrictLength)) %>%  # av. PM25

    arrange(desc(PM10))
```

As a final result, we obtained a table consisting of 12 rows (corresponding to 12 Berlin districts) and 3 columns (District name, PM10 and PM25).

3.4 Data Merge

The data described in previous three sections needed to be merged to one table which is further used for index calculation. In order to merge datasets by name of the district we had to unify writing of district names (function *DistricToFullName*) and then merge. That was accomplished by means of the following R-Script:

```
dt2$District = DistricToFullName(dt2$District)
dt3$District = DistricToFullName(dt3$District)

lvbInDt = merge(dt1, dt2[,-1], by = "District") %>%
    merge(dt3, by = "District")
```

4 Liveability Index

In the following section the Livability Index for each district of Berlin is calculated. Firstly, code used for indicators calculation and their normalization is explained. Secondly, Sub- and Total Indexes will be calculated and thirdly, the results will be summarized.

4.1 Indicators

4.1.1 Calculating Indicators

To calculate Berlin Districts Livability Index the total of 33 indicators have been considered (See Annex A). To be able to compare the value of the indicators between districts accurately, the relative values (rather then the absolute) of the variables had to be considered. For this purpose, some of the variables have been converted to per capita or per Hectare terms. In the code, two functions have been written:

```
PerCapita = function(x){
    # Function divides the data per number of
    # inhabitants of given district to  obtain per capita value
    #Args:
    #    x: the column of which the data should decided
    #      per number of citizens
    # Returns:
    # The vector of data x divided per number of inhabitants
    # of given district
  x/lvbInDt$Population
}
PerHa = function(x){
    # Function divides the data by size in ha of given
    # district to obtain per ha value
    #Args:
    #    x: the column of which the data should
    #    divided by size of the district
    # Returns:
    # The vector of data x divided by size of the district in Ha
  x/lvbInDt$Size
}
```

The list of the indicators and their description may be found in Annex A. The two functions has been applied to following indicators:

Relative Unit	Indicator
Per Capita	lvSpc, hsAv, prkSp, trs, doc, trf, socHl
Per Ha	dns, trnDn, bkLN, sprCl, crChr,res, strCr, grSp
Original Units	hsAl, htlOc, std, grdSz, chU3, chU6, actSn, actJn,crm, dsb, emp, comp trRv, bnk, agrRe, tr, pm10, pm25

Table 1: *Units of the Indicators*

See Quantles: Data_Prep_1 Data_Prep_2 Data_Prep_3 Data_Merge Liv_Index_Calc

If neither of the functions has not be been applied, the variable is already expressed in the relative value or absolute value is reasonable to use as the indicator, e.g. total taxable revenue of the

companies in the given district. For more details see Annex A and 107:150 lines of code in Quantlet - "LivIndex_Calc".

4.2 Normalization of the Indicators

To avoid issues related to different scales and units of the indicators, all the variables were normalized. Depending on whether the indicator has positive or negative (in the code ngtInd) contribution to livability, equation 1 (in the code NormalizePositive function) or equation 2 (NormalizeNegative) from methodology section have been applied for normalization.

To run the calculation across each indicator the following loop has been used, where results are stored as "Score" (in the code "Scr") to a new column:

```
for (i in 1:(ncol(indDt))){  # Run fun. for every column
    if ((i) %in% ngtInd){
        indDt[,paste(colnames(indDt[i]),"Scr")] =
             NormalizeNegative(indDt[i])
    } else {
        indDt[,paste(colnames(indDt[i]),"Scr")] =
               NormalizePositive(indDt[i])
    }
}
```

The new set of normalized indicators was saved as *Index_Score_Data.csv* (see Annex A) and is the bases of all further calculations in this section.

4.3 Sub-Indexes and Total Liveability Index

4.3.1 Sub-Indexes Calculation

Based on the normalized indicators (Scores) five Sub-Indexes have been calculated: ***Housing*** (in the code Phy1In; 4 Indicators), ***Infrastructure*** (Phy2In; 4 Indicators), ***Social*** (SocIn; 16 Indicators),***Economic*** (EcoIn; 4 Indicators), ***Environmental*** (EnvIn; 5 Indicators). The 33 indicators have been split into those 4 categories (in code lines 211:256) and the sub-indexes have been calculated as simple sum of their scores:

$$Sub - Index_i = \sum_{a}^{b} IndicatorScore \tag{3}$$

where a in equation eq:3 stands for the first Indicator of the category and b for the last one. In the code, this operation has been done with *apply* function:

```
Phy1In = apply(phy1Ind, 1, sum)
Phy2In = apply(phy2Ind, 1, sum)
SocIn  = apply(socInd, 1, sum)
EcoIn  = apply(ecoInd, 1, sum)
EnvIn  = apply(envInd, 1, sum)
```

4.3.2 Total Livability Index Calculation

As described in the Methodology section, each Sub-Index has been weighted towards the Total Index with 15%, 10%, 25%, 25% and 25% respectively.The four Pillars of the Total Livability Index (Physical, Social, Economic and Environmental) have been calculated as follows:

$$Pillar_j = \sum_{1}^{a} Sub - Index_i \times weight\_of\_Sub - Index_i \tag{4}$$

where a is the number of Sub-Indexes belonging to given pillar. Other than Physical Pillar, where we sum Sub-Index 1 and 2 (Housing and Infrastructure), $a = 1$. For details, see methodology and Annex A.

The Total Livability Index is just sum of all the Pillars:

$$LivabilityIndex = \sum_{1}^{4} Pillar_i \tag{5}$$

In the code the calculation has been conducted as follows:

```
# Calculate the contribution of each pillar to the Liveability Index
Pllrs= data.frame(PhyPl   = (Phy1In*phys1W + Phy2In*phys2W),
                  SocPl = SocIn*socW,
                  EcoPl = EcoIn*ecoW,
                  EnvPl = EnvIn*envW)
# Calculate total Liveability Index
TotalIn = apply(Pllrs,1, FUN = sum)
```

4.4 Summary of the Results

The results of the Sub-Indexes, Pillars and Total Index calculations can be founds in Table 2. The districts, which obtained the highest score for each Sub-Index are *Charlottenburg-Wilmersdorf*, *Mitte*, *Friedrichshain-Kreuzberg*, *Mitte*, *Reinickendorf*, respectively. The district with the highest Livability Index is *Mitte* whereas the lowest score has been obtained by *Neukolln*.

In order to compare the distribution of the scores for each Sub-Index, the box plot of each of them has be drawn in Figure 8. For each Sub-Index the distribution of its quantiles is centered approximately around the half of maximum potential score in the category. Taking into account that our data has been normalized, this is to be expected.

District	Hous. Index	Infr. Index	Soc. Index	Eco. Index	Env. Index	Phys. Pillar	Soc. Pillar	Eco. Pillar	Env. Pillar	Total Index
Charlottenburg-Wilmersdorf	3.13	2.19	7.95	2.61	2.47	0.64	1.99	0.65	0.62	3.90
Friedrichshain-Kreuzberg	1.68	2.81	8.46	2.25	1.37	0.59	2.11	0.56	0.34	3.61
Lichtenberg	2.20	1.38	7.84	0.58	2.79	0.43	1.96	0.14	0.70	3.23
Marzahn-Hellersdorf	2.04	1.19	5.70	0.21	2.41	0.38	1.43	0.05	0.60	2.46
Mitte	1.69	3.81	7.92	3.47	1.47	0.74	1.98	0.87	0.37	3.95
Neukolln	1.24	1.39	5.13	1.01	2.03	0.33	1.28	0.25	0.51	2.37
Pankow	2.45	1.16	6.53	2.75	2.63	0.42	1.63	0.69	0.66	3.40
Reinickendorf	2.13	1.01	6.40	0.42	3.24	0.36	1.60	0.11	0.81	2.88
Spandau	1.93	0.48	6.79	0.06	2.85	0.26	1.70	0.02	0.71	2.69
Steglitz-Zehlendorf	2.68	0.68	6.72	1.49	2.61	0.37	1.68	0.37	0.65	3.08
Tempelhof-Schoneberg	1.87	1.46	6.61	2.03	1.83	0.41	1.65	0.51	0.46	3.02
Treptow-Kopenick	2.76	0.69	5.48	1.07	1.70	0.38	1.37	0.27	0.43	2.44
Max Score	4	4	16	4	5	1.00	4.00	1.00	1.25	7.25

Table 2: *Sub-Indexes, Pillars contribution and Total Livability Index for 12 districts of Berlin*

See Quantles: Data_Prep_1 Data_Prep_2 Data_Prep_3 Data_Merge Liv_Index_Calc

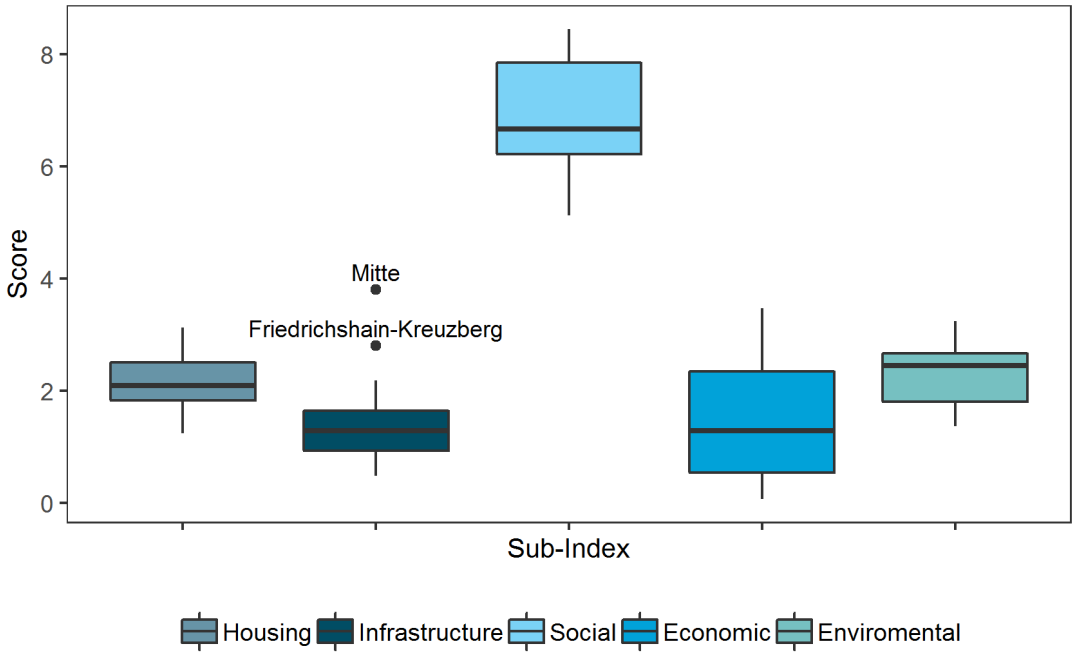

Figure 2: *Box Plot of all the Sub-Indexes*

See Quantles: Data_Prep_1 Data_Prep_2 Data_Prep_3 Data_Merge Liv_Index_Calc

Within each category, it has been checked for potential outliers to determine if any of the districts performed particularly well or bad. Following function to identify outliers has been used:

```
IsOutlier =  function(y) {
    # Function checks if the given observation in vector y is an
    # outlier or not.
    # The outlier is defined as an observation smaller than the
    # first quantile minus 1,5 times the interquartile range
    # of the # vector OR bigger than the third quantile plus
    # the 1.5 times the interquartile range of the vector. The
    # interquartile range # is calculated using IQR function.
    # For more info. see ?IQR.
    # Agrs:
    #      y: vector of data for which outliers are to be found.
    #      The type of data must be numeric.
    # Returns: Log. statement "TRUE" for the outlier, else "FALSE"
    return(y < quantile(y,
                        0.25) - 1.5 * IQR(y) |
               y > quantile(y, 0.75) + 1.5 * IQR(y))
}
```

As it is shown in Figure 8, the outliers has been observed only in the Infrastructure Sub-Index, where *Mitte* and *Friedrichshain-Kreuzberg*(very central neighborhoods) scored far better than the other districts. In all other Sub-Indexes, there is no districts, which scores would stood out from the rest. Therefore, the districts which obtained best and worst scores must vary across the indicators of those Sub-Indexes.

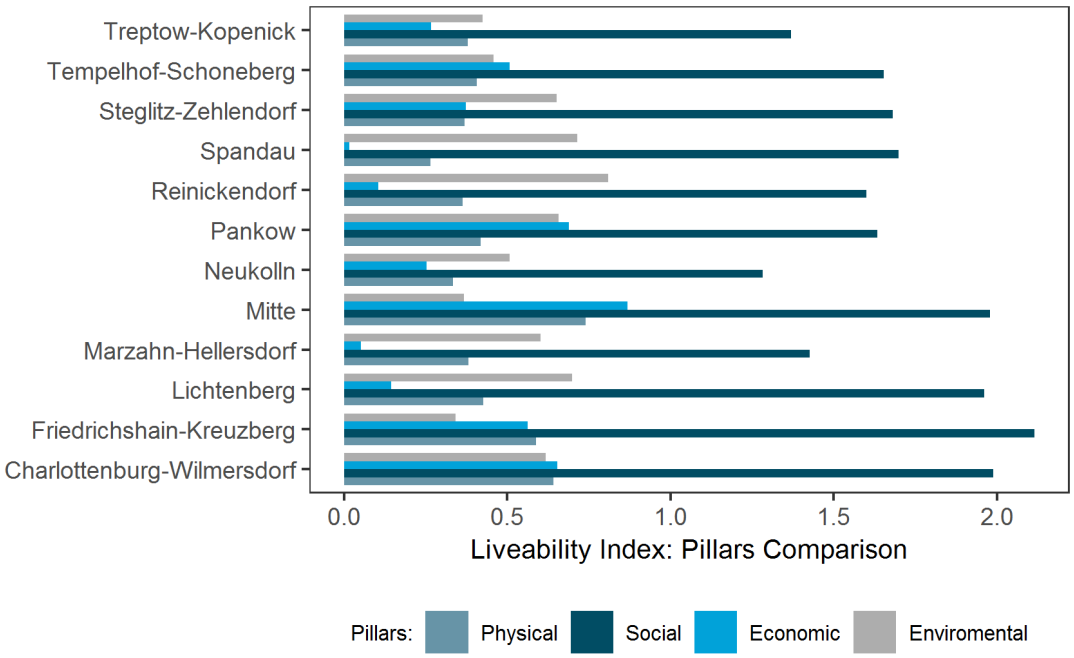

Figure 3: *Pillar Comparison for each district*

See Quantles: Data_Prep_1 Data_Prep_2 Data_Prep_3 Data_Merge Liv_Index_Calc

The Figure 3 illustrates the breakdown of each Pillar score (equation 4)between the districts and gives clear overview of their performance. The district ranked as the most livable, *Mitte*, scores very highly in all of the Pillars, other than Environmental.As Mitte is the most central district, its poor

environmental score is not surprising. The cumulative total Livability Index score (equation 5) has been illustrated in Figure 4. The difference between the best (Mitte 3.95) and the worst (Neukolln 2.37) district is 1.58 score, which is quite high. The central districts have much better infrastructure, higher level of economic activity and cultural engagement, which makes them score pretty high in the Index. Neukolln is known to be the "Hipster" district of Berlin and the Hub for artists. It is not surprising therefore, to see that it scored very poorly in Economic Pillar. The neighborhood is also very grey and high level of crime is recorded. Consequently, it obtained the lowest score in the Index. Another district known for its bad reputation, Marzahn-Hellersdorf, has scored only slightly better. The next step is the comparison of the Livability's Index results with the living costs in each of the districts.

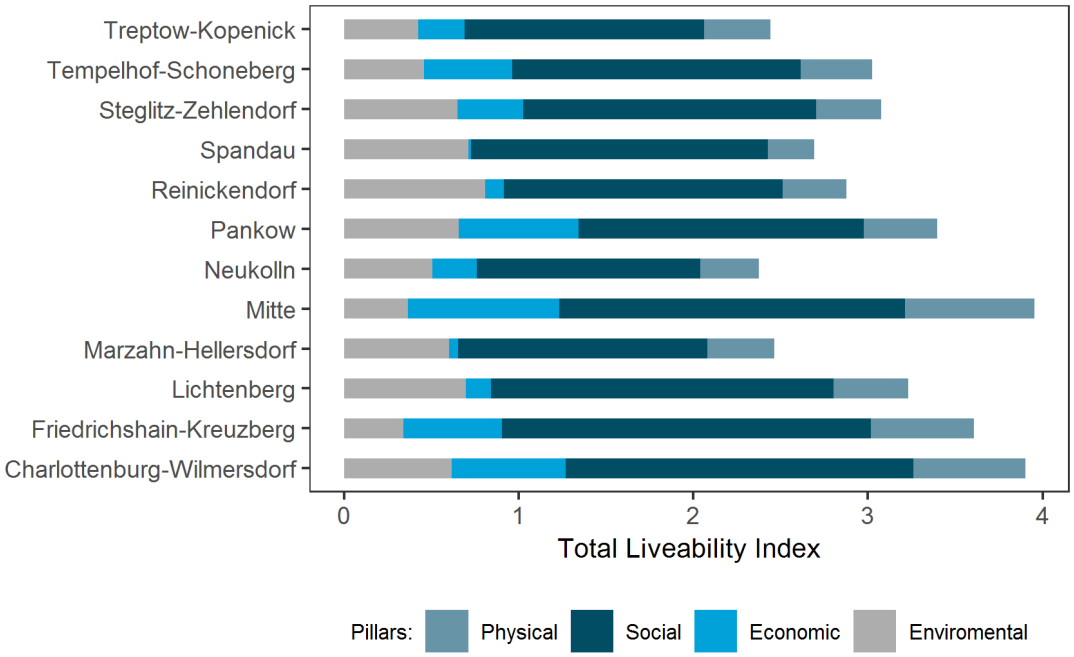

Figure 4: *Cumulative Total Livability Index*

See Quantles: Data_Prep_1 Data_Prep_2 Data_Prep_3 Data_Merge Liv_Index_Calc

5 Visualization of results and Rent Analysis

The Analysis framework was based on the data for 12 Berlin districts.

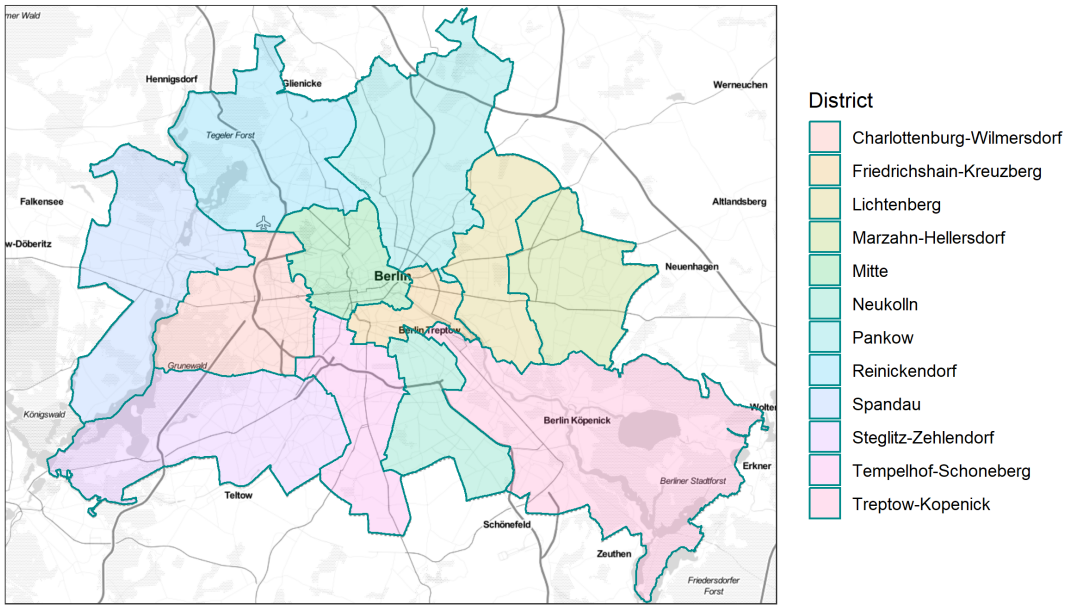

Figure 5: *Districts of Berlin*

See Quantlets: Rent_Analysis Liv_Index_Calc

For the purpose of the analysis we used the following data:

- Livability index and subindexes calculated in previous chapters
- Average rent per m^2 for each district: https://de.statista.com/statistik/daten/studie/259905/umfrage/mietpre in-berlin-nach-bezirken/
- Data polygons for borders of each district: https://data.technologiestiftung-berlin.de/dataset/bezirksgrenzen
- Data polygon of historical Berlin wall: https://opendata-esri-de.opendata.arcgis.com/
- Stamen-maps

```
# Download map-teils from Stamen Maps
map = get_stamenmap(ber_adj, maptype="toner-lite", zoom = 11)

# Download polygons with districts borders
test = readOGR("./06_Results_Visualization/bezirksgrenzen.kml")
Bezirk = fortify(test)

# Merge Index and Rent Data with District borders
IndRntDt = Bezirk %>%
```

```
    merge(IndDt, by = "District") %>%
    merge(RntDt, by = "District")
```

Within this section we are going to answer 3 questions: 1) How is the livability of the district connected to the average rent in the district? 2) Which districts are overpriced and which ones are underpriced? 3) If there still a difference among East and West parts of Berlin as it used to be in the past?

In order to answer those questions, we performed visual analysis and applied basic statistical concepts, including correlation and linear regression analysis. It should be mentioned that statistical analysis and validity of its results are highly constrained by the small sample size (12 observations). That is why we rather focus on the visual interpretations of calculated results.

5.1 Analysis of Rent vs. Livability Index

According to (Schultheis 2019), Berlin has one of the highest rates of rent increase among other German cities. Because of the increase in residents and tourism, Berlin recently transformed from one of the cheapest cities to live to one of the most expensive. The graph below shows the distribution of rent per sq.m. over 12 districts together with calculated livability index.

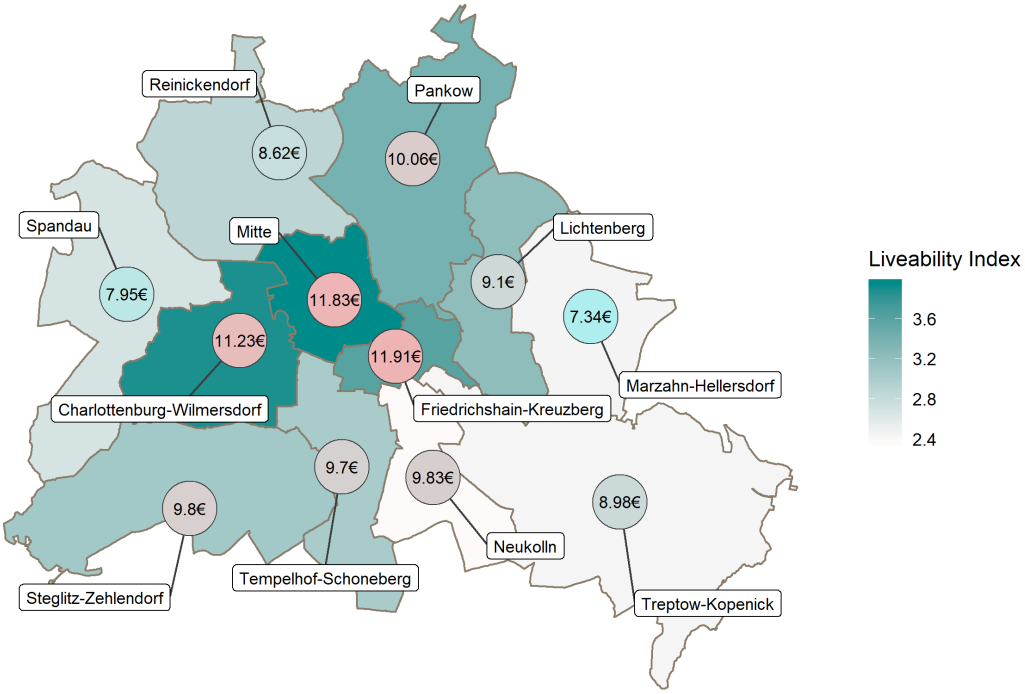

Figure 6: *Visual Comparison of Livability Index with Rent*
See Quantlets: Rent_Analysis Liv_Index_Calc

The map was produced in R with the following script:

```
# Plot liveability Index heatmap with rent labeled on it
```

```
p2 = ggplot(IndRntDt,  # Plot districts
            aes(long, lat, group = group))+
  geom_polygon(aes(fill = TotalIn),   # Color polygons according to index value
               colour = "bisque4")+   # Color of the border
  scale_fill_gradient(low = "snow1",  # Set color gradient scale
                      high = "darkcyan",
                      name = "Liveability␣Index") +
  theme_bw() +
  theme(panel.border = element_blank(),  # No graph border
        panel.grid.major = element_blank(),  # No grid
        panel.grid.minor = element_blank(),
        axis.line  = element_blank(),
        axis.ticks = element_blank(),
        axis.text  = element_blank(),
        axis.title = element_blank())+
  scale_color_gradient(low = "paleturquoise",  # Set colors for Rent labels
                       high = "rosybrown2",
                       guide=FALSE) +
  geom_label_repel(data = LablesDt,  # Add district labels
             aes(long, lat, label = District, group = District),
             size = 3,
             box.padding = 2,
             segment.color = 'grey24') +
  geom_point(data = LablesDt,  # Add round balls for rent labels
             aes(long, lat,
                 colour = Rent,
                 group = District),
             size = 13) +
  geom_point(data = LablesDt,  # Add rent balls borders
             aes(long, lat,
                 group = District),
             size = 13,
             shape = 1,
             colour = "grey24") +
  geom_text(data = LablesDt,  # Add Rent Numbers
            aes(long, lat, label = paste0(Rent,"   "), group = District),
            size = 3)
```

As we can observe, districts with high livability index usually also have high average rent. It is also clear that among districts with low livability index the dependence is weaker. The dependence structure of average district rent and the livability index is clearer illustrated on the scatter plot below.

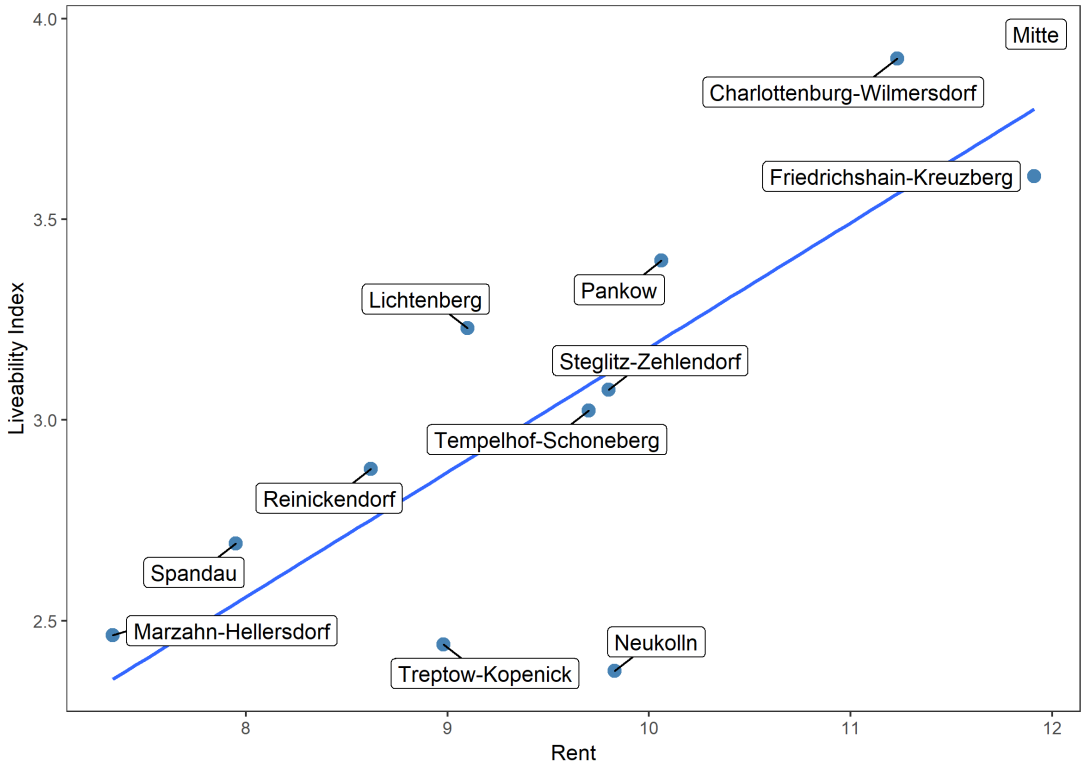

Figure 7: *Rent and Livability Index Dependence*
See Quantlets: Rent_Analysis Liv_Index_Calc

The graph was created in R with the following script:

```
ggplot(LablesDt, aes(x=Rent, y=TotalIn)) +
  geom_point(size = 3,
             colour = "steelblue") +  # Use hollow circles
  geom_smooth(method = lm,  # Add linear regression line
              se = FALSE) +  # Don't add shaded confidence region
  theme_bw()+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(y = "Liveability Index") +
  geom_label_repel(data = LablesDt,
                   aes(Rent, TotalIn, label = District, group = District),
                   size = 4,
                   box.padding = 0.5)
```

As we can see from the plot above, there is a clear positive dependence between rent and livability index. The pattern can be well approximated by a straight line, which is why we can use Pearson correlation coefficient to measure the degree of dependence. The correlation coefficient is equal to 0.812, which means that rent and livability index are highly linearly dependent. Determination coefficient R^2 is equal to 0.659 which means that livability index explains 66% of rent variance. Regression coefficient is significant on 0.1% significance level. Described statistics were calculated with the following R script:

```
#calculate correlation between Rent and Total Index
cor(LablesDt$TotalIn, LablesDt$Rent)

#run regression of rent on calculated total index and print regression summary
LablesDt %>%
  lm(Rent~TotalIn, .) %>%
  summary()
```

5.2 Analysis of historical differences between East and West Berlin

The last question we wanted to address in our analysis is the difference of historically separated west and east parts of Berlin. Is there still a difference in development of those two parts? To answer this question we created a heatmap of the livability index and overlaid the historical border of the Berlin Wall.

Figure 8: *Livability in East and West Berlin*
See Quantlets: Rent_Analysis Liv_Index_Calc

The graph does not show any clear dependence between livability and historical separation. To get a full picture we also considered rent and subindexes (housing, infrastructure, social, economic and ecological with regard to historical border)

(a) *Average rent*

(b) *Housing*

(c) *Infrastructure*

(d) *Social Index*

(e) *Economic Index*

(f) *Environment*

Figure 9: *Comparison of East and West Berlin by multiple parameters*

As there is no clear dependence, we conclude that all the historical differences we suspected at the beginning of our analysis disappeared during the last 30 years (Berlin was united in 1989).

Maps were produced by means of an R function:

```
# For repeated plots I write a function with all the settings
EastWest = function(variable, title, name, leg = TRUE) {
    # Author: Aleksandra Kudaeva
```

```
    # Description: function creates a heatmap of Berlin with highlighted
    #              Berlin Wall border according to the given parameter
    # Input: Table and column, which will determine coloring,
    #        Title of legend
    #        Name of png file which is going to be saved
    #        Legend: by default - TRUE
    # Output: Heatmap, printed and saved in png
    plot = ggplot(IndRntDt,
                  aes(long, lat, group = group)) +
           geom_polygon(aes(fill = variable),
                        colour = "darkcyan",
                        show.legend = leg) +
           geom_path(data = Wall, aes(x = long, y = lat),
                     color="bisque4",
                     size = 1.5) +
           scale_fill_gradient(low = "snow1",
                     high = "darkcyan",
                     name = title) +
           theme_bw() +
           theme(panel.border = element_blank(),
                 panel.grid.major = element_blank(),
                 panel.grid.minor = element_blank(),
                 axis.line  = element_blank(),
                 axis.ticks = element_blank(),
                 axis.text  = element_blank(),
                 axis.title = element_blank()) +
           geom_label(data = LablesCnt,
                      aes(long, lat, label = Part, group = Part),
                      size = 6)
    ggsave(paste0(name, ".png"),
           plot = plot,
           scale = 1,
           device = "png",
           path = "SPL_BerlinDst_Rent_Analysis/")

    return(plot)
}
```

6 Principle Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables, which are called principal components.

It is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. We can chose certain number of principle components from the original dataset, without losing too much information from the original dataset. By doing that, we reduce the dimension of variables, which lowers the data complicity and allow us to do some data visualization.

PCA can be done by eigenvalue decomposition of a data correlation matrix of a data matrix, usually after a normalization step of the initial data. There are different packages and functions in R for principle components Analysis. Here we use prcomp. And we choose the indicator score to do the analysis. The R code is shown below:

```
data.pca = prcomp(select(IndexFull,ends_with('Scr')),
                  center = TRUE,
                  scale. = TRUE)
summary(data.pca)  # Importance of components
# part code of plotting
fviz_pca_ind(data.pca,
                  col.ind = "cos2", # Color by the quality of representation
                  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
                  repel = TRUE) +     # Avoid text overlapping
                  geom_text_repel(aes(label=IndexFull$District),size=3.5)+
                  theme_bw()  +
                  theme(panel.grid.minor = element_blank(),
                  panel.grid.major = element_blank(),
                  panel.border = element_blank())
```

Center means whether the variables should be shifted to be zero centered, and scale means whether the variables should be scaled to have unit variance before the Analysis take place,which is particularly recommended when variables are measured in different scales, otherwise, the PCA outputs obtained will be severely affected. here we set both to be True.

Figure 10 shows us how much variance of the data has been explained by each component. We could see from the graph that the first four components already contains most of the information. Furthermore, if we want to get a closer look, we could call *summary* in R to see the cumulative proportion of principle components.

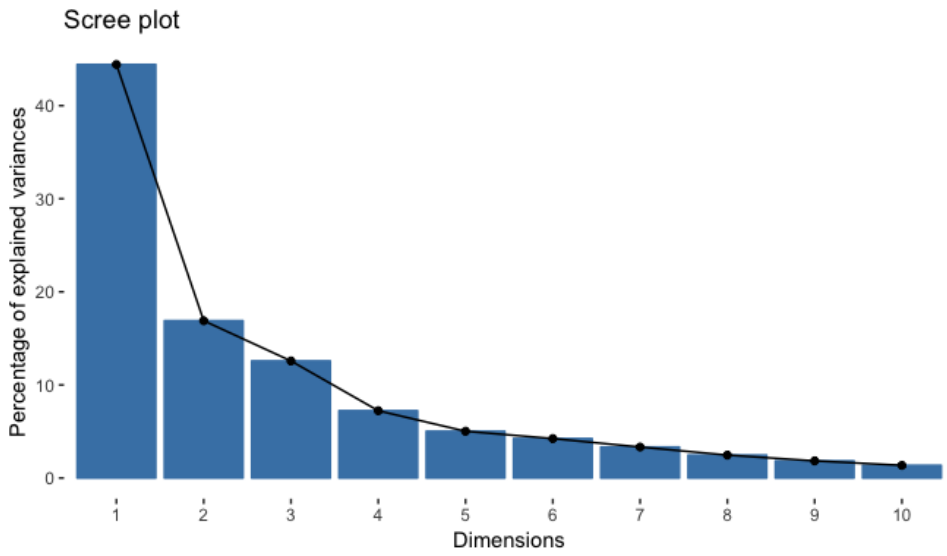

Figure 10: *Explained Variance of Each Component*
See Quantlets: PCA and Cluster

By choosing the first two principle components, we are able to do some visualization of variables and individuals. And for convenience ,here we chooses only ten most important variables to get an overview, since we also care more about variables with higher contribution to our principle components.

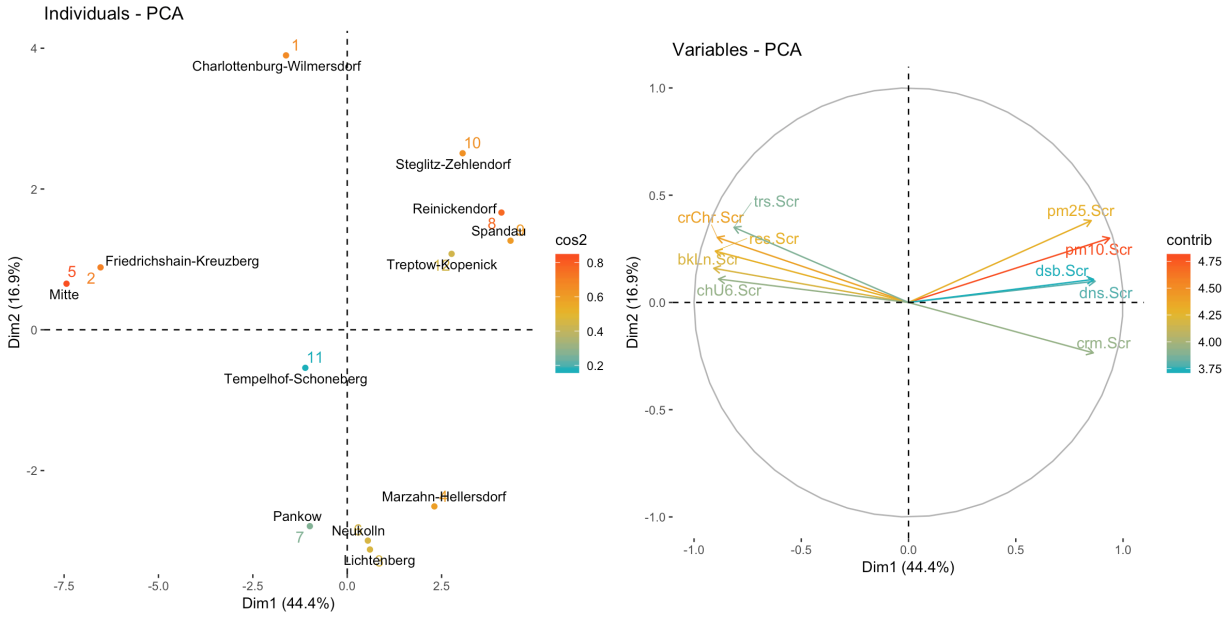

Figure 11: *Individual and Variables PCA*
See Quantlets: PCA and Cluster Liv_Index_Calculation

6.1 Distance of Individuals and Correlation of Variables

From Figure 11, the first graph on the left, also known as the Quality of representation. We could see the 'distance' of individuals, based on x and y axis being the first and second principle components. This distance indicates the similarity or difference between districts, and how significant those

difference are.

On the second graph on the right, which is also known as variable correlation plots. we see the relationships between ten variable-vectors we chose. Positively correlated variables are grouped together, and negatively correlated variables has opposite directions. e.g., PM10.Scr and PM25.Scr are positively correlated with each other and negatively correlated with e.g. res.Scr. For a deeper understanding of those directions, we see that they mostly point closer to x-axis than y-axis,because those variable-vectors are most important variables we chose for contributing the first and second components, since the first component contains more data variation, they are supposed to point closer to x-axis.

6.2 Contribution of Indivisuals and Variables

Secondly, the color of individuals and variables indicates their different levels . For the variables, it means the level of contribution from a variable to the component, while on the other hand, the level of different data point shows the importance of a principal component for a given observation (vector of original variables). From the graph on the left, we see that the component are most important to Mitte, which means it loses least information while doing the transformation via principle component analysis.And from the Graph on the right, We see that PM10.Scr has the highest contribution to component one. Which means it is highly correlated with PC1. We could also get a closer look for that:

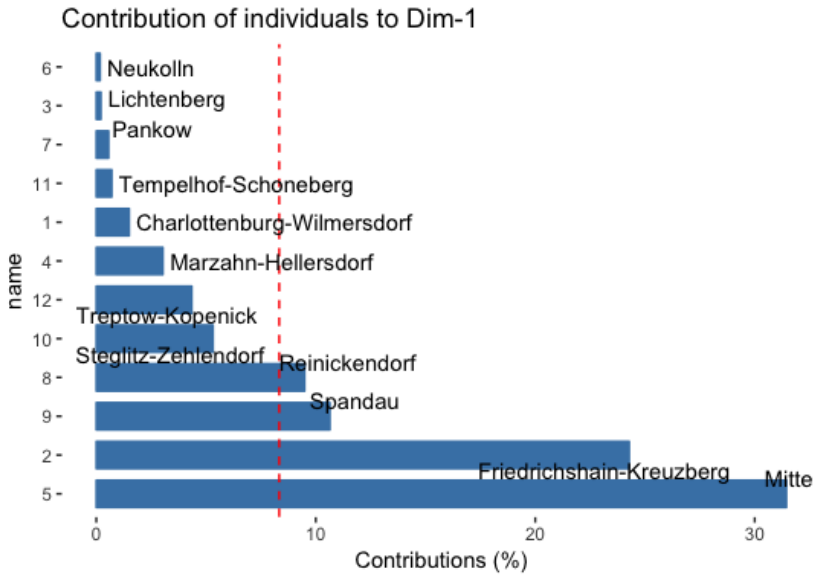

Figure 12: *Importance of First Component to Individuals*

See Quantlets: PCA and Cluster

Lastly, some critic and thoughts about principle component in this project is that we don't have a very clear clue what those numbers in principle component exactly represent, but rather only a rough understanding of which variable contributes how much percentage for the component. So we dropped the possibility to calculate our livability based on components, since we are not able to determine what exactly makes a district 'better' than another one.

7 K Means Clustering

Out of historical reason, east and west Berlin have been having some different characteristic. We want to see if these difference would still be obvious by clustering districts according to our indicators to two set. And we also calculate the optimal number of clustering

7.1 Choosing cluster dataset

We chose 4 groups of variables for cluster, i.e.

1. 5 Index: physical1,physical2, social, economical and environmental index
2. 4 Pillar: physical, social, economical and environmental pillar, which are weighted indexes
3. 9 index : 5 index together with 4 pillars
4. 33 score, which are normalized raw data

And the purpose of this grouping is to see how different data processing are going to influence the result of clustering.

7.2 Optimal number of k

Before performing clustering, we also want to have a look at the optimal number of cluster. If it's not two, then we could chose to perform clustering for both number of cluster, and compare them, which is dropped in this report. There are also several methods to calculate the optimal number of k-fold cluster, here we use average Silhoutte method. First we define a function to calculate the average Silhoutte width. and The optimal number of clusters k is the one that maximizes the average silhouette over a range of possible values for k.

```
#Average Silhouette Method

AvgSil = function(k, df) {
    # Function computes the average Silhouette width
    # The optimal number of clusters k is the one that maximizes
    # the average silhouette over a range of possible values for k
    #
    # Arg:
    # k is number of clusters; df is the dataframe that we apply kmean cluster
    #
    # Output:
    # mean value of silhouette width for one specific k value
    km = kmeans(df, centers = k, nstart = 25)
    s = silhouette(km$cluster, dist(df))
    # silhouette : compute Silhouette Information from Clustering
    # returns 3 columns : cluster, neighbor, sil_width
    mean(s[, 'sil_width'])
}
```

We could call the function to see the optimal number:

```
k = c(2:10)  # for function 'silhouette' we need cluster number to be at least 2
df = CIndex
#df = CPilar
#df = CInPi
#df = CFull
AvgSil_values = sapply(k, FUN = AvgSil, df)
k[which(AvgSil_values == max(AvgSil_values))]  # output:2
```

Alternativaly we could also visualize the Optimal Number of Clusters by calling function in package:

```
fviz_nbclust(CIndex, kmeans, method = "silhouette")
fviz_nbclust(CPilar, kmeans, method = "silhouette")
fviz_nbclust(CInPi, kmeans, method = "silhouette")
fviz_nbclust(CFull, kmeans, method = "silhouette")

ggarrange(f1,f2,f3,f4,labels = c('1','2','3','4'))
```

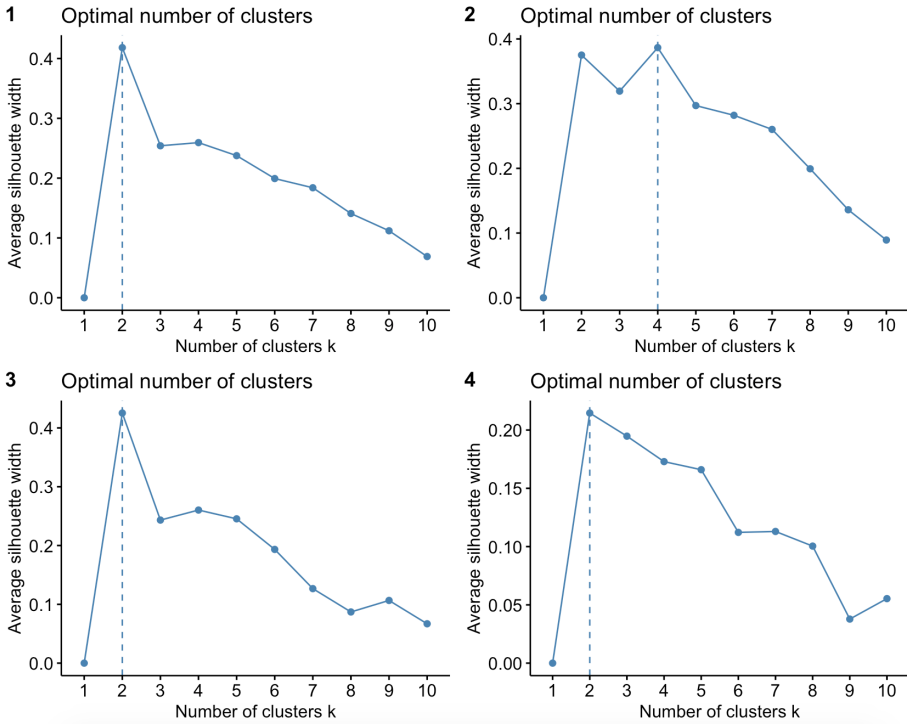

Figure 13: *Clustering results*

See Quantlets: PCA and Cluster

7.3 Comparative analysis

As mentioned in the chapter before, with the help of principle component, we are able to visualize the cluster result, with the first component being x axis and the second the y axis, which makes much more sense than randomly choosing two variables as x and y axis, where we couldn't tell how much percent variance of the data each randomly chosen variable could explain. Now let's take a look at our cluster result according to the optimal number of k. It also shows us depends on different data set, how much percent does the component explains the total variance of the original data.

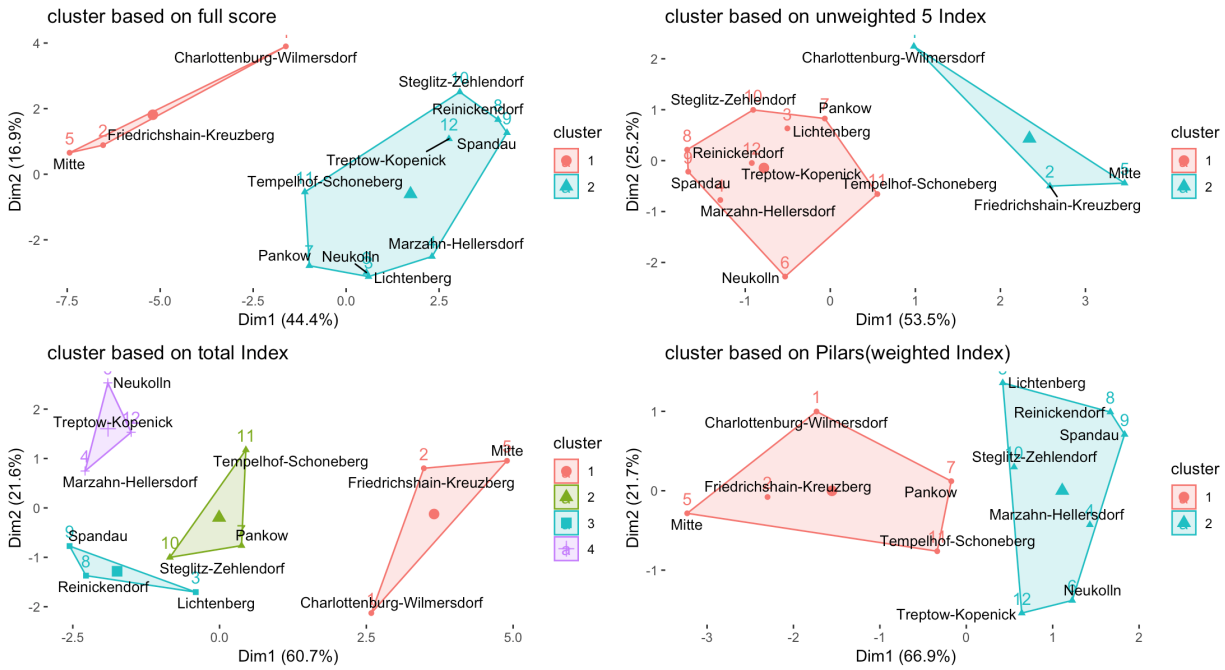

Figure 14: *Optimal number of k*

See Quantlets: PCA and Cluster Rent_Analysis Liv_Index_Calc

By comparing the first and second Graph above, we see high similarity, possible reason is that the second graph on the right, i.e. Cluster based on unweighted five Index, is just the unweighted sum of the full score. Unweighted sum indicate invariant variance of data before and after linear transformation. On the other hand, if we compare the first two graph above with the two below, we do see some changes in the cluster, the changes seems significant at the beginning, but if we take a closer look, we still see the similar pattern, similar districts still lie nearby each other and are tend to be grouped together. which means linear combination of variables wouldn't change much of the cluster result, but when we weight the variables differently, we could still get some difference.

Recall that our original purpose was to see if there still exist significant different between east and west Berlin via our livability index calculation based on the 33 indicators we have. The result of clustering give us a negative answer.

Although from the cluster result we couldn't directly see which district are better, but since our indicators have a build-in ranking characteristic, we could conjecture that good districts tends to be grouped together and bad ones as well. To confirm that, we could compare the cluster result to livability index ranking. i.e. Comparing figure3 and figure14.

8 Conclusion

This report presents a description of results of an R-programming project devoted to statistical description of Berlin's districts. The main idea of the project was to calculate index, which could describe the quality of life (livability) in different districts of Berlin.

Project results help to answer our initial questions. According to the project results the best district to live in Berlin is "Mitte", which is not surprising considering that city center has the best social, economic and infrastructural parameters. The worst to live is "Neukoeln": the district which does not correspond to the common criteria of prosperity and is generally famous for that. Close results were obtained by clustering methods. Comparison of calculated livability index with average rent per district helps us to see that some districts are overestimated (Neukoeln, Treptow-Kopenick) and some underestimated (Charlottenburg-Wilmersdorf, Lichtenberg). At the same time, neither values of livability index nor subindixes contain any spatial pattern. Historical border separating East and West Berlin seems to have no effect on the current development level of the districts. That means that last 30 years managed to erase all the differences in development of those two parts. The same result was proved by conducted cluster analysis.

The structure of report was determined by the sequence of steps and challenges we meet in programming. That is why, big part is devoted to the data sources and possible ways of data acquisition, as it was one of the main challenges of the project. The choice of factors and further calculation of the Index were performed according to the adjusted methodology for livability index calculation. Statistical methods were chosen according to initial questions and data constraints.

9 Declaration of Authorship

We hereby confirm that we have authored this Seminar paper independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin , 15.03. 2018, Names of Group members:

- Malgorzata Paulina Olesiewicz 598939
- Qi Wu 601623
- Aleksandra Kudaeva 599943

Name	Report Sections	Quantlets
Olesiewicz	Motivation, Methodology, Data Preparation 3.1, Livability Index	Data_Prep_1, Liv_Index
Wu	Data Preparation 3.2, PCA and cluster	Data_Prep_2, PCA_Cluster
Kudaeva	Data Preparation 3.3 - 3.4, Rent Analysis, Conclusion	Data_Prep_3, $\text{Data}_{M}erge, Rent_{A}nalysis$

Table 3: *Distribution of work*

References

Agency, E. E. (2016), 'Air quality standards under the air quality directive, and who air quality guidelines'.
URL: *https://www.eea.europa.eu/data-and-maps/figures/air-quality-standards-under-the*

Cooke, B. & Zaby, A. (2015), 'Skill gaps in business education: fulfilling the needs of tech startups in berlin', *Journal of Higher Education Theory and Practice* **15**(4), 97.

Diener, E. & Suh, E. (1997), 'Measuring quality of life: Economic, social, and subjective indicators', *Social indicators research* **40**(1-2), 189–216.

Giap, T. K., Thye, W. W. & Aw, G. (2014), 'A new approach to measuring the liveability of cities: the global liveable cities index', *World Review of Science, Technology and Sustainable Development* **11**(2), 176–196.

Kuhn, T. (2019), 'Google erÖffnet neues bÜro in berlin anders als gedacht'.
URL: *https://www.wiwo.de/technologie/digitale-welt/google-eroeffnet-neues-buero-in-berlin-anders-als-gedacht/23896914.html*

Ministry of Urban Development, G. o. I. (2017), 'Methodology for collection and computation of liveability standards in cities'.
URL: *smartcities.gov.in/upload/uploadfiles/files/MethodologicalReportFinal.pdf*

Okulicz-Kozaryn, A. (2013), 'City life: Rankings (livability) versus perceptions (satisfaction)', *Social Indicators Research* **110**(2), 433–451.
URL: *https://doi.org/10.1007/s11205-011-9939-x*

Schultheis, E. (2019), 'Berlins radical plan to stop rocketing rents'.
URL: *http://www.bbc.com/capital/story/20190226-berlins-radical-plan-to-stop-rocketing-rents*

tagesspiegel (2019), 'Pro jahr sterben 13.000 deutsche vorzeitig durch verkehrsabgase'.
URL: *https://www.tagesspiegel.de/wirtschaft/studie-zu-luftverschmutzung-pro-jahr-sterben-13-000-deutsche-vorzeitig-durch-verkehrsabgase/24046760.html*

thelocal.de (2019), 'Berlin clubs brought city 1.5 billion in 2018: study'.
URL: *https://www.thelocal.de/20190213/new-research-highlights-the-economic-value-of-berlins-nightlife-and-why-it-might-be-under-threat*

Unit, E. I. (2011), 'Liveanomics–urban liveability and economic growth', *London: The Economist* .

10 Annex A: Table of Indicators

Pillar	Sub.Index	Weight.	Category	Group of Indicators	Indicators	Name in the code	Value	Unit
Physical	1. Housing	0.15	1. Housing	1.1 Housing availability	1.1.1 Living space	lvSpc	Living space per capita	m2/capita
					1.1.2. Housing available	hsAv	Number of flats per Ha	nr/Ha
					1.1.3 Density of population	dns	Population per Ha	nr/Ha
				1.2 Housing Inclusiveness	1.2.1. Housing Benefits	hsAl	Housing allowance households	nr
	2. Infrastructure	0.10	2. Infrastructure	2.1 Public Transportation Network	2.1.1 Density of Public Transportation Network	trnDn	Number of Public Transportation Stops per Ha	nr/Ha
				2.2 Cycling lines availability	2.2.1 Length of cycling lines	bkLn	Meters of cycling lines available per Ha	m/Ha
				2.3 Electric car readiness	2.3.1 Number of charging stations	crChr	Number of charging stations per Ha	nr/Ha
				2.4 Number of parking spaces	2.4.1 Number of parking places	prkSp	Number of parking spaces per Capita	nr/capita
Social	3. Social	0.25	3. Culture & Tourism	3.1 Tourist attractiveness	3.1.1 Tourist guests	trs	Tourist guests per Capita	nr
					3.1.2 Hotel occupancy	htlOc	Overnight stays/Hotel beds *365 days	%
				3.2 Culture	3.2.1 Number of restaurants	res	Number of Restaurants Clubs per Ha	nr
					3.2.2 Number of Sport Clubs	sprCl	Number of Sport Clubs per Ha	nr
			4. Education	4.1 Schools	4.1.1 Number of pupils	std	Pupils/K	nr
					4.1.2 Average grade size	grdSz	Pupils/ grades	nr
				4.2 Early child care avaiability	4.2.1 Children (<3 years old) in the daycare	chU3	% of kids < 3y/o in daycare	%
					4.2.2 Children (3 -6 years old) in the daycare	chU6	% of kids 3-6y/o in daycare	%
			5. Health	5.1 Health care availability	5.1.1 Number of house doctors	doc	Number of house doctors	nr/capita
				5.2 Activeness of citizens	5.2.1 Active seniors (61+)	actSn	% of Seniors as Members of Sport Clubs	%
					5.2.2 Active youth (<18)	actJn	% of Juniors as Members of Sport Clubs	%
			6. Safety	6.1 Street Safety	6.1.1 Street Traffic Accidents	trf	Street Traffic Accidents per Capita	nr/capita
					6.1.2 Number of Street Crossings	strCr	Street Crossings per Ha	nr/capita
				6.2 Crime	6.2.1 Criminal offenses	crm	Choose criminal offences per 10K ppl	nr/10K ppl
			7. Social Support	7.1 Amount of Social Support	7.1.1 Social Help	socHl	Social Help Recipients per Capita	nr/capita
				7.2 Handicaps inclusiveness	7.2.1 Handicaps inclusiveness	dsb	Severely handicapped per 1K ppl	nr/capita
Economic	4. Economic	0.25	8. Economic	8.1 Level of Employment	8.1.1 Employment	emp	Nr of Employed per 1K capable of employment	nr/ 1K ppl
				8.2 Local Economic Growth	8.2.1 Number of registered companies	comp	Nr of Companies	nr
					8.2.2 Taxable Revenue	txRv	Taxable Revenue from the companies	Euro
					8.2.3 Bankruptcies	bnk	Nr of bankruptcies per nr of companies	nr
Environmental	5. Environmental	0.25	9. Environmental	9.1 Greenness	9.1.1 Green Open Space	grSp	Green Space per Ha	nr
					9.1.2 Land usage	agrRe	Agricultural surface to residential and traffic surface ratio	nr
					9.1.3 Tree per km	tr	Trees per km of road	nr/km
				9.2 Pollution	9.2.1 PM 25	pm25	Average concentration	microgram/m3
					9.2.2 PM 10	pm10	Average concentration	microgram/m3

10.1 Table of Scores

District	Charlottenburg-Wilmersdorf	Friedrichshain-Kreuzberg	Lichtenberg	Marzahn-Hellersdorf	Mitte	Neukolln	Pankow	Reinicke-ndorf	Spandau	Steglitz-Zehlendorf	Tempelhof-Schoneberg	Treptow-Kopenick
lvSpc.Scr	0.783	0.657	0.509	0.785	0	0.19	0.056	0.934	1	0.999	0.301	0.963
hsAv.Scr	1	0.54	0.627	0.319	0.363	0	0.58	0.157	0.015	0.464	0.318	0.604
dns.Scr	0.722	0	0.687	0.781	0.352	0.529	0.812	0.891	0.914	0.891	0.592	1
hsAl.Scr	0.627	0.482	0.374	0.156	0.974	0.524	1	0.152	0	0.324	0.662	0.19
trnDn.Scr	0.298	1	0.279	0.291	0.842	0.513	0.069	0.154	0.116	0.212	0.509	0
bkLn.Scr	0.441	1	0.224	0.219	0.965	0.348	0.111	0.116	0.185	0.216	0.497	0
crChr.Scr	0.564	0.756	0.261	0	1	0.034	0.075	0.048	0.179	0.028	0.302	0.002
prkSp.Scr	0.884	0.052	0.616	0.676	1	0.497	0.909	0.689	0	0.23	0.154	0.685
trs.Scr	0.569	0.379	0.096	0	1	0.059	0.06	0.038	0.052	0.023	0.137	0.059
htlOc.Scr	0.816	0.978	0.695	0.301	1	0.273	0.876	0.502	0.725	0	0.929	0.119
sprCl.Scr	0.26	1	0.168	0	0.573	0.163	0.049	0.123	0.127	0.094	0.241	0.01
res.Scr	0.343	1	0.017	0.02	0.654	0.125	0.124	0.031	0.027	0.051	0.21	0
std.Scr	0.257	0.083	0.316	0.264	0	0.108	0.33	0.979	0.573	1	0.073	0.17
grdSz.Scr	0.428	0.699	0.882	0.733	0.605	1	0.55	0.304	0.508	0	0.372	0.42
chU3.Scr	0.303	0.652	0.766	0.507	0.701	0.06	1	0	0.169	0.537	0.632	0.716
chU6.Scr	0.48	1	0.485	0.087	0.983	0.079	0.629	0.083	0	0.389	0.603	0.454
doc.Scr	1	0.68	0.191	0.68	0.132	0.155	0	0.801	0.944	0.6	0.444	0.391
actSn.Scr	0.977	0.324	0.46	0	0.614	0.256	0.295	0.574	1	0.761	0.358	0.67
actJn.Scr	1	0.215	0.164	0	0.749	0.219	0.284	0.454	0.355	0.729	0.232	0.658
trf.Scr	0.162	0	1	0.976	0.087	0.856	0.95	0.471	0.325	0.744	0.712	0.25
strCr.Scr	0.204	1	0.134	0.256	0.376	0.212	0.193	0.084	0.028	0.065	0.097	0
crm.Scr	0.555	0.352	0.89	0.905	0	0.71	0.887	0.792	0.823	1	0.796	0.913
socHl.Scr	0.021	0.096	1	0.235	0.183	0.158	0.034	0.16	0.22	0	0.075	0.072
dsb.Scr	0.575	0	0.577	0.739	0.257	0.695	0.27	1	0.912	0.726	0.7	0.573
emp.Scr	0.407	0.508	0.336	0.206	0.678	0.386	1	0.06	0	0.25	0.55	0.261
comp.Scr	1	0.591	0.026	0.001	0.909	0.212	0.69	0.1	0	0.317	0.512	0.158
txRv.Scr	0.243	0.161	0.007	0.005	1	0.033	0.063	0.115	0	0.033	0.136	0.051
bnk.Scr	0.963	0.989	0.207	0	0.884	0.381	1	0.146	0.064	0.895	0.829	0.597
grSp.Scr	0.381	0.577	0.844	0.955	0.991	1	0.487	0.358	0.344	0.196	0.436	0
agrRe.Scr	0.007	0	0.528	0.051	0	0.048	1	0.195	0.511	0.015	0.039	0.064
tr.Scr	1	0.708	0.708	0.458	0.354	0.042	0.292	0.729	0	0.812	0.625	0.125
pm10.Scr	0.538	0.081	0.392	0.538	0	0.582	0.368	0.964	1	0.856	0.425	0.841
pm25.Scr	0.547	0	0.321	0.407	0.122	0.354	0.48	0.992	1	0.726	0.304	0.672

11 Annex B: R code

11.1 Quantlet 1: SPL_BerlinDst_Data_Prep_1

Data_Prep_1

```
#=========================PREPARING THE ENVIROMENT============================

#setwd("~/SPL_BerlinDst")
# The code was tested at HU PC Pool 25

install.packages("rvest")
install.packages("magrittr")
install.packages("rlang")
install.packages("dplyr")
install.packages("tidyr")
install.packages("rgdal")
install.packages("xlsx")
install.packages("maptools")# Install required pakages

library(rvest)
library(magrittr)
library(rlang)
library(dplyr)
library(tidyr)
library(rgdal)
library(xlsx)
library(maptools) # Load required packages


#========FUNCTIONS FOR CLEANING AND ARRANGING THE DATA=====================

GetDataUnderURL = function(URL){
    # Function returns the data table from under the URL address.
    # Firstly, the URL address is read (webpage). Secondly, the data of
    # the type "table" is extracted from the HTML document with the CSS
    # selector (?html_nodes). The data table is read, stored and
    # returned as a table.
    #
    # Args:
    #   URL: URL address under which the required data is
    #        stored. The table which contains the data must
    #        be the first table [[1]] on the page.
    #
    # Returns:
    # Matrix with the data stored in the first table under from
    # the URL address

    webpage = read_html(URL)
    table = html_table(html_nodes(webpage, "table")[[1]],
                       fill = TRUE,trim = TRUE)
}

DataToNumeric = function(column){
```

```
    # Function cleans up data in chosen column by replacing "," with "." as a
    # decimel symbol, removes empty space between numbers and converst the
    # data to numeric type
    #
    # Args:
    #     column: name of the column in the dataframe which values suppose to
    #     be cleaned up and converted to numeric type
    #
    # Returns:
    # Vector with numeric values of converted data
    commatodot = (gsub(",", ".", column))
    num = as.numeric(gsub("[[:space:]]", "", commatodot))
    return(num)
}

DistricToFullName = function (column){
    # Function re-names the districts of Berlin with their full names
    # without special signs. The function identifies the district by
    # three or four letters of its name.
    #
    # Args:
    #      column: vector or a column containing names of Berlin
    #      districts.Name can have any form, special signs can be
    #      use and additional information can be added to it. Vector
    #      must be character type.Only first 3-4letters need to be correct.
    #
    # Returns:
    #        Vector of replaced district names by its official names
    #        without special signs.

    column[grepl("mit",column, ignore.case = TRUE)] = "Mitte"
    column[grepl("fri",column,
                 ignore.case = TRUE)] = "Friedrichshain-Kreuzberg"
    column[grepl("pank",column,ignore.case = TRUE)] = "Pankow"
    column[grepl("mar",column,ignore.case = TRUE)] = "Marzahn-Hellersdorf"
    column[grepl("char",column,
                 ignore.case = TRUE)] = "Charlottenburg-Wilmersdorf"
    column[grepl("spa",column,ignore.case = TRUE)] = "Spandau"
    column[grepl("ste",column,ignore.case = TRUE)] = "Steglitz-Zehlendorf"
    column[grepl("tem",column,ignore.case = TRUE)] = "Tempelhof-Schoneberg"
    column[grepl("trep",column,ignore.case = TRUE)] = "Treptow-Kopenick"
    column[grepl("neu",column,ignore.case = TRUE)] = "Neukolln"
    column[grepl("lich",column,ignore.case = TRUE)] = "Lichtenberg"
    column[grepl("rein",column,ignore.case = TRUE)] = "Reinickendorf"

    return(column)
}

NrofStops= function(district){
    # Function checkes, based on the geografical coordinates, how many public
    # transportation stops (in file bsStp) are in the chosen Berlin district
    # and counts their number.
    #
    # Args:
```

```
    #     district: names of the polygon file with the coordinates of the
    #     given Berlin district.The longitude coordinates  must be the
    #     first column of the file, the latitude the second column.
    #
    # Returns:
    # Number of public transportation stops in the given district
    # or on its border ( in whichstops function 1 = in the
    # polygon, 2 = on the border of the polygon ,0 = not in the
    # polygon).
    whichstops = point.in.polygon(bsStp$stop_lon, bsStp$stop_lat,
                                  district[, 1],district[, 2])
    nrstops = length(whichstops[whichstops != 0])

    return(nrstops)
}

#========ACCESING THE DATA FROM STATISTIK BERLIN BRANDENBURG WEBSITE==========

# The data on Statistik Berlin Brandenburg Website are stored under several
# URL adresses. Creat a list with each URL as its element:

lstUrl = list(pplt = paste0("https://www.statistik-berlin-brandenburg.de/re",
                            "gionalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sa",
                            "geb=13003&creg=BBB&anzwer=6"),

              sz   = paste0("https://www.statistik-berlin-brandenburg.de/regi",
                            "onalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=",
                            "33000&creg=BBB&anzwer=8"),

              srf  = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=33",
                            "000&creg=BBB&anzwer=8"),

              hshl = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=12",
                            "011&creg=BBB&anzwer=5"),

              stdt = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=21",
                            "001&creg=BBB&anzwer=5&fbclid=IwAR0d7Ebm0uLtl5cc81",
                            "O49tvFXeW0PRU_jEFh2qaCLWl1G2XSFv7mnb-SmwU"),

              immb = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=31",
                            "000&creg=BBB&anzwer=0"),

              trsm = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=45",
                            "005&creg=BBB&anzwer=7"),

              socl = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=22",
                            "001&creg=BBB&anzwer=5"),
```

```
              chld = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=22",
                            "005&creg=BBB&anzwer=9"),

              hndy = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=22",
                            "007&creg=BBB&anzwer=10"),

              cmp  = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=52",
                            "001&creg=BBB&anzwer=5"),

              bnkr = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=52",
                            "004&creg=BBB&anzwer=7"),

              accd = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=46",
                            "002&creg=BBB&anzwer=6"),

              allw = paste0("https://www.statistik-berlin-brandenburg.de/regio",
                            "nalstatistiken/r-gesamt_neu.asp?Ptyp=410&Sageb=22",
                            "003&creg=BBB&anzwer=7"))


# Apply the DatafromURL function to all the previously listed URL adresses and
# split each data table from the tblLst into individual data frame in the
# Global Enviroment

tblLst = lapply(lstUrl, GetDataUnderURL) %>%
    list2env(envir = .GlobalEnv)


#======== PREPARING DATA FRAME FROM STATISTIK WEBSITE FOR MERGING =========

# The data tables aquired from the website are very poorly structured and
# columns are not directly linked with their names. Therefore hardcoding of
# numbers of rows and columns is required.

X1 = c(pplt[5:16, 1])  # choose district name data
X2 = c(pplt[5:16, 2])  # choose population  size data
X3 = c(sz[4:15, 2])  # choose district size data
X4 = c(srf[4:15, 3])  # choose residential and traffic surface data
X5 = c(srf[4:15, 4]) %>%  # choose agricultural surface data
    replace(1, "0")  # replace the first element "-" with 0
X6 = c(pplt[5:16, 3])  # choose employment data
X7 = c(pplt[5:16, 5])  # choose non-professional persons data
X8 = c(hshl[4:15, 2])  # choose  nr of private hauseholds data
X9 = c(hshl[4:15, 3])  # choose persons/household data
X10 = c(stdt[4:15, 2])  # choose nr of schools data
X11 = c(stdt[4:15, 3])   # choose  nr of grades data
X12 = c(stdt[4:15, 4])  # choose  nr of pupils data
X13 = c(stdt[4:15, 5])  # choose  nr of pupils dat / K
X14 = c(immb[4:15, 2])  # choose  nr of residential buildings data
```

```
X15 = c(immb[4:15, 3])  # choose  nr of flats
X16 = c(immb[4:15, 4])  # choose  total living space
X17 = c(immb[4:15, 5])  # choose living space per capita
X18 = c(trsm[4:15, 3])  # choose  nr of hotel beds
X19 = c(trsm[4:15, 4])  # choose  nr of tourist guests
X20 = c(trsm[4:15, 5])  # choose  nr of overnight stays
X21 = c(socl[5:16, 2])  # choose  nr of social help recipients
X22 = c(chld[4:15, 3])  # choose  nr of children in daycare
X23 = c(chld[4:15, 4])  # choose % of children in daycare under 3
X24 = c(chld[4:15, 5])  # choose  nr of %Children in daycare 3 - under 6
X25 = c(hndy[4:15, 2])  # choose  nr of severely handicapped
X26 = c(hndy[4:15, 3])  # choose  nr of severely handicapped/1K
X27 = c(cmp[4:15, 2])  # choose  nr of companies
X28 = c(cmp[4:15, 3])  # choose  nr of taxable revenues
X29 = c(bnkr[5:16, 2])  # choose  nr of bankruptcies
X30 = c(accd[4:15, 4])  # choose  nr of street traffic accidents /10K
X31 = c(allw[5:16, 2])  # choose  nr of housing allowance households


# Merge the data vectors together into one data frame and split the number
# and the name of the district in X1 into two seperate columns

wbsDt = data.frame(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16,
                   X17,X18,X19,X20,X21,X22,X23,X24,X25,X26,X27,X28,X29,X30,X31,
                   stringsAsFactors = FALSE) %>%
    separate(col = X1,into = c("X0", "X1"),sep = "␣")%>%
    mutate_at(vars(X1), DistricToFullName) %>%
    mutate_at(vars("X2":"X31"),DataToNumeric)


# Name the columns of wbsDt
colnames(wbsDt) = c("Nr",
                    "District",
                    "Population",
                    "Size",
                    "RsSurface",
                    "AgrSurface",
                    "Employment",
                    "NonProf",
                    "Hausehold",
                    "PplHausehold",
                    "Schools",
                    "Grades",
                    "Pupils",
                    "Puplis1000",
                    "Buildings",
                    "Flats",
                    "Space",
                    "SpacePC",
                    "Hotel",
                    "Tourists",
                    "Stays",
                    "SocHelp",
                    "Daycare",
```

```
                    "Child3",
                    "Child6",
                    "Disabled",
                    "Disabled1000",
                    "Company",
                    "Revenue",
                    "Bankruptcy",
                    "Accidents",
                    "Allowance")

#==================READING IN THE SPORT DATA FILES=========================

# Data source paset0("https://www.statistik-berlin-brandenburg.de/Statistiken/",
#"statistik_SB.asp?Ptyp=700&Sageb=21006&creg=BBB&anzwer=10")

# Read in sport clubs memberships data and convert columns to numeric.

sprtMb = read.xlsx("SPL_BerlinDst_Data_Prep_1/SB_B05-01-00_2018j01_BE.xls",
                   sheetName = "T9", startRow = 4, encoding = "UTF-8",
                   as.data.frame = TRUE) %>%
    mutate_at(vars("bis6":"mehrund61"),DataToNumeric) %>%
    mutate_at(vars("Bezirk"),as.character)

# Read in sport clubs numbers data and rename the columns
# Convert columns to numeric and name the districts properly

# Ignore Warning message: "In (function (column)  : NAs introduced by coercion"

sprtCl = read.xlsx("SPL_BerlinDst_Data_Prep_1/SB_B05-01-00_2018j01_BE.xls",
                   sheetName = "G3", encoding = "UTF-8", startRow = 2,
                   as.data.frame = TRUE) %>%
    mutate_at(vars("Sportvereine"),DataToNumeric) %>%
    mutate_at(vars("Bezirk"),as.character)

#==================== PREPARING SPORT DATA FOR MERGING ======================

# Sport Club Members

mbDst = c(sprtMb$Bezirk[33:44])%>%  # Chose district names
    DistricToFullName(.)  # Uniformise the disctrict names


# Take mean of 3 age groups (up to 6, 7-14 and 15-18) to obtain average % of
# active sport club members age 0-18

jnrMb = c(rowMeans(sprtMb[33:44, 4:6]))


snrMb = c(sprtMb$mehrund61[33:44])  # Choose active sport club memb. age 61+

spMbDt = data.frame(mbDst, jnrMb, snrMb,
                    stringsAsFactors = FALSE)  # create a data frame

colnames(spMbDt) = c("District","JunSport","SenSport")  # name the columns
```

```
# Sport Clubs

clbDst = c(sprtCl$Bezirk[1:12]) %>%  # Chose sport club district name data
    DistricToFullName(.)

clbNm = c(sprtCl$Sportvereine[1:12])


clbsDt = data.frame(clbDst,
                    clbNm,
                    stringsAsFactors = FALSE)  # Create data frame


colnames(clbsDt) = c("District", "Sport")  # Name the column


#====================== READING IN BUS STOP DATA  ==========================


# Read in district borders coordinates
# Data source: https://data.technologiestiftung-berlin.de/dataset/bezirksgrenzen
# Ignore warnings: no altitude values for KML object 1 - 16


dstrBrd = getKMLcoordinates("SPL_BerlinDst_Data_Prep_1/bezirksgrenzen.kml")

# Read in bus stops data
# Source: https://daten.berlin.de/datensaetze/vbb-fahrplandaten-gtfs

bsStp = read.csv("SPL_BerlinDst_Data_Prep_1/public␣transportation␣stops.csv")


#=================== PREPARING BUS STOP DATA FOR MERGING ===================

# create a list of all the polygons assigned to corresponding districts

dstLst = list( rein = dstrBrd [[1]],  # chose polygon of Reinickendorf
               char = dstrBrd[[2]],  # chose polygon of Charlottenburg-Wilmersd.
               trep = dstrBrd[[3]],  # chose polygon of Treptow-K penic
               pank = rbind(dstrBrd[[4]], dstrBrd[[5]], dstrBrd[[6]],
                            dstrBrd[[7]], dstrBrd[[8]]), # chose polys.of Pankow
               #
               neu  = dstrBrd[[9]],  # chose polygon of Neukolln
               lich = dstrBrd[[10]],  # chose polygon of Lichtenberg
               marz = dstrBrd[[11]],  # chose polygon of Marzahn-Hellersdorf
               spa  = dstrBrd[[12]],  # chose polygon of Spandau
               steg = dstrBrd[[13]],  # chose polygon of Steglitz-Zehlendorf
               mit  = dstrBrd[[14]],  # chose polygon of Mitte
               fri  = dstrBrd[[15]],  # chose polygon of Friedrichshain-Kreu.
               tem  = dstrBrd[[16]])  # chose polygon of Tempelhof-Sch neberg


# Apply NrofBusStops function to all the districts:
```

```
bsLst = lapply(dstLst, NrofStops)

# Creat data frame from list names and computed elements

bsStpDt= data.frame(DistricToFullName(names(bsLst)),
                    matrix(unlist(bsLst)),
                    stringsAsFactors = FALSE) %>%
    mutate_at(vars("matrix.unlist.bsLst.."),DataToNumeric)


colnames(bsStpDt)= c("District",
                     "Transport") # name the columns

#=========================READING IN CRIME DATA============================
# There is problem with encoding in this file as special signs make the code
# not replicable on every computer when calling columns by their name
# (even if encoding is set to "UTF-8" ). Therefore, maching of columns
# names by only couple of (first) latters in the name is required
# Columns of interst start with:

crmCl= c("LOR", "Bezei", "Straft") # Vector with begining of column names

crm = read.xlsx("SPL_BerlinDst_Data_Prep_1/Fallzahlen&HZ␣2012-2017.xls",
                sheetName = "HZ_2017", encoding = "UTF-8",
                startRow = 3, as.data.frame = TRUE) %>%
    select(grep(paste(crmCl,collapse="|"), names(.))) %>%  # select columns
    mutate_at(grep("Bezei", names(.)), as.character)


#=======================PREPARING CRIME DATA FOR MERGING================

# Select data based on their "LOR Schlussel"[, 1] where districts are coded
# with the last 4 numbers being "0000". Select only required data. Convert
# names of the districts to full names, the rest of the data to numeric

crmDt = crm[grep("0000", crm[, 1]), ] %>%
    select(2:3) %>%
    mutate_at(grep("Bezei", names(.)), DistricToFullName) %>%
    mutate_at(grep("Straft", names(.)), DataToNumeric)


colnames(crmDt) = c("District", "Crime")  # name the columns


#======================= READING IN THE PARKING SPACES DATA================

# Writte in manualy numbers of parking spaces from mobility report. See:
# paset0("https://www.berlin.de/senuvk/verkehr/politik_planung/zahlen_",
#"fakten/download/Mobilitaet_dt_komplett.pdf")


rprtDt = c(26488, 4090, 26000, 20500, 2726, 7400, 7150)  # Avaiable data
```

```
#====================== PREPARING PARKING DATA FOR MERGING=================

# For the remaining districts the average of avaiable data has been used, not to
# impact the index calculation. Create a vector with observations for all the
# districts

prkNr = c(rprtDt, rep(mean(rprtDt), 12 - length(rprtDt)))

prkDst = c("Mitte",
           "Friedrichshain-Kreuzberg",
           "Pankow",
           "Charlottenburg-Wilmersdorf",
           "Spandau",
           "Steglitz-Zehlendorf",
           "Tempelhof-Sch neberg",
           "Neukolln",
           "Treptow-K penick",
           "Marzahn-Hellersdorf",
           "Lichtenberg",
           "Reinickendorf") # create vector with district names

prkDt = data.frame(prkDst, prkNr,
                   stringsAsFactors = FALSE) %>%  # merge data frame
    mutate_at(vars(prkDst),DistricToFullName) %>%
    mutate_at(vars(prkNr),DataToNumeric)   # convert data to numeric

colnames(prkDt) = c("District", "Parking")  # name columns

#========================READING IN TREES DATA==============================

# Data Source: paste0("https://de.statista.com/statistik/daten/studie/652680/",
#"umfrage/strassenbaeume-in-berlin-nach-bezirken/")
# Read in the trees data.
# Ignore Warning: Warning message: In DataToNumeric(NA..1) : NAs introduced
# by coercion

trNm = paste0("SPL_BerlinDst_Data_Prep_1/",
              "statistic_id652680_strassenbaeume-in-",
              "berlin-nach-bezirken-2017.xls")

tr = read.xlsx(trNm, sheetName = "Daten", as.data.frame = TRUE ,
               encoding = "UTF-8") %>%
    mutate_at(grep("Berlin", names(.)), as.character) %>%  # Con. Dist. Names
    mutate_at(-grep("Berlin", names(.)), DataToNumeric)  # Con.other data to num.

#====================PREPARING TREES DATA FOR MERGING=====================

trDst = tr[3:14, grep("Berlin", names(tr))] %>%
    DistricToFullName(.) # select district names and mutate them  to full form

trKM = tr[3:14, 3] # select total number of trees per km of the road
```

```
trDt = data.frame(trDst,
                  trKM,
                  stringsAsFactors = FALSE) # merge data frame


colnames(trDt) = c("District", "Trees") # name columns

#=========READING IN  AND PREPARING FOR MERGING GREEN SPACE DATA ===========

# Read in green space data
# Data source: Source of data : paste0("https://de.statista.com/statistik/",
# "daten/studie/652716/umfrage/oeffentliche-gruenflaechen-in-berlin-nach-",
# "bezirken/")

# Again problem with encoding here.
# Intrested in "  ffentliche  .Gr nfl chen.in.ha" use "lich" to identify column


grSpNm = paste0("SPL_BerlinDst_Data_Prep_1/statistic_id652716_oeffentliche-",
                "gruenflaechen-in-berlin-nach-bezirken-2017.xlsx")

grSpDt = read.xlsx(grSpNm, sheetName = "Daten", startRow = 5,
                   as.data.frame = TRUE, encoding = "UTF-8") %>%
    mutate_at(vars(grep("lich", names(.))),DataToNumeric) %>%
    mutate_at(vars(- grep("lich", names(.))), as.character) %>%
    mutate_at(vars(- grep("lich", names(.))), DistricToFullName)

colnames(grSpDt) = c("District", "GreenSp")

#=============== MERGING DATA SETS INTO ONE DATA FRAME ====================


FnlDt = merge(wbsDt,spMbDt, by.y = "District") %>%
    merge(.,clbsDt, by.y = "District") %>%
    merge(.,crmDt,by.y = "District") %>%
    merge(.,prkDt,by.y = "District")%>%
    merge(.,bsStpDt,by.y = "District") %>%
    merge(.,trDt,by.y = "District") %>%
    merge(.,grSpDt,by.y = "District")  # Merge all the data sets by the district


write.csv2(FnlDt, "SPL_BerlinDst_Data_Prep_1/SPL_BerlinDst_Data_Prep_1.csv")

# Read in best with:
# read.csv("SPL_BerlinDst_Data_Prep1/SPL_BerlinDst_Data_Prep_1.csv",
# sep = ";", dec = ",", row.names = 1, stringsAsFactors = FALSE)

# ======================END OF THE SCRIPT=================================
# ========================================================================
```

11.2 Quantlet 2: SPL_BerlinDst_Data$_{prep_2}$

Data_Prep_2

```
#======================== Set working directory==============================
setwd("~/SPL_BerlinDst")
#========================install and run packages============================

packages = c('rvest','readxl','magrittr','dplyr')
install.packages('rvest')
install.packages('readxl')
install.packages('magrittr')
install.packages('dplyr')
install.packages("xlsx")
library(rvest)
library(readxl)
library(magrittr)
library(dplyr)
library(xlsx)

#============================list district names================================
Dstc = c("Mitte",
         "Friedrichshain-Kreuzberg",
         "Pankow",
         "Charlottenburg-Wilmersdorf",
         "Spandau",
         "Steglitz-Zehlendorf",
         "Tempelhof-Schoeneberg",
         "Neukoelln",
         "Treptow-Koepenick",
         "Marzahn-Hellersdorf",
         "Lichtenberg",
         "Reinickendorf"
)
Dstc = as.data.frame(Dstc)
colnames(Dstc) = c("District"); Dstc

#====================Ortsteil in every district ============================

# Ortsteil is the German word for subdistricts

# using package rvest to read internet page
Oteil = read_html(paste0("https://de.wikipedia.org/wiki/",
                         "Liste_der_Bezirke_und_Ortsteile_Berlins"))

Oteil = Oteil %>%
    html_nodes("table")%>%
    .[[3]] %>%  #the third table in this website give us a dataframe
    # lists all Orststeile in Berlin and their corresponding districts
    html_table()

# define a function to replace all german letter which could causes issues
Replace = function(clmn) {
    # author: Aleksandra Kudaeva
    # Input:  column where you want to replace umlauts
    # Output: column without umlauts
    # check if at least one element of a vector has any umlauts in it
    # replaces umlauts until there are no one left
```

```
    while(any(grepl("  |  |  |  |␣",clmn)) == TRUE) {
        clmn  %<>%
            sub("  ", "ae", .) %<>%
            sub("  ", "oe", .) %<>%
            sub("  ", "ue", .) %<>%
            sub("  ", "ss", .) %<>%
            sub("␣", "-", .) # also replace all space with '-'
    }
    return(clmn)
}

# replace all the german letters in the dataframe
Oteil$Bezirk = Replace(Oteil$Bezirk)
Oteil$Ortsteil = Replace(Oteil$Ortsteil)

# create a dataframe OteilD which contains Ortsteil in 12 districts
OteilD = data.frame(matrix(ncol = length(Dstc$District),
                           nrow = 20))
# nrow can be any number bigger than the maxi number of Ortsteil in district

j = 1  # start with the first observation in dataframe Oteil

for (i in 1:length(Dstc$District)){  # a loop with number of districts

    j = j  # every loop start with current j value where last while loop ends
    k = 1  # k start from 1 for every column in the new dataframe OteilD

    while(Oteil[j,'Bezirk'] == Dstc[i,"District"]){
        OteilD[k,i] = Oteil[j,'Ortsteil']
        k = k+1
        j = j+1
    }
}  # ignore the error message hier : missing value where TRUE/FALSE needed

colnames(OteilD) = Dstc$District
View(OteilD)

#=========================charging station================================

Cgst = read_excel("SPL_BerlinDst_Data_Prep_2/ladestationen.xls")
# if error appears here, then use:
#read.xlsx("SPL_BerlinDst_Data_Prep_2/ladestationen.xls",
#          encoding = "UTF-8",
#          as.data.frame = TRUE)

# extract post code in column 'Cgst$Adresse',
# where information listed as e.g."Malteserstrasse 136<U+2013>138, 12249 Berlin"
# and create a new column only with post code

Adss = strsplit(Cgst$Adresse,"[,]")  # returns a list

# next step:extract info we need from the list and create a vector to store them

Adsscl = c()  # create a column to store the postcode together with 'Berlin'
```

```
for(i in 1:length(Adss)) {
    Adsscl[i]= Adss[[i]][2]  # the second element has the postcode information
}   ;  Adsscl

Adsscl = as.character(Adsscl)  # need charachter for fct strsplit

Adsscl = strsplit(Adsscl,"␣") # split postcode and 'Berlin', return a list

# same as before extract info from list and create a vector to store them
Pcd = c()
for(i in 1:length(Adss)) {
    Pcd[i] = Adsscl[[i]][2] # the second element has the postcode information
}

#create new dataframe with name CgstN with the new column of post code
CgstN = data.frame(Cgst,Pcd)

# extract post code of district from file 'ZuordnungderBezirkezuPostleitzahlen'

Pscd = read_excel(paste0("SPL_BerlinDst_Data_Prep_2/",
                         "ZuordnungderBezirkezuPostleitzahlen.xls"))
Pscd

# if error appears here, then use:
#read.xlsx(paste0("SPL_BerlinDst_Data_Prep_2/",
#                 "ZuordnungderBezirkezuPostleitzahlen.xls"),
#          encoding = "UTF-8",
#          as.data.frame = TRUE)


# create 12 vector to store the Post code for each district
# 3:12 are the columns with post code infomation
Dstc01 = as.numeric(unlist(as.list(Pscd[6:8,3:12])))
Dstc02 = as.numeric(unlist(as.list(Pscd[10:11,3:12])))
# ignore warning of NA since it doesn't effect our use of the data
Dstc03 = as.numeric(unlist(as.list(Pscd[13:15,3:12])))
Dstc04 = as.numeric(unlist(as.list(Pscd[17:20,3:12])))
Dstc05 = as.numeric(unlist(as.list(Pscd[22:23,3:12])))
Dstc06 = as.numeric(unlist(as.list(Pscd[25:27,3:12])))
Dstc07 = as.numeric(unlist(as.list(Pscd[29:32,3:12])))
Dstc08 = as.numeric(unlist(as.list(Pscd[34:36,3:12])))
Dstc09 = as.numeric(unlist(as.list(Pscd[38:39,3:12])))
Dstc10 = as.numeric(unlist(as.list(Pscd[41:42,3:12])))
Dstc11 = as.numeric(unlist(as.list(Pscd[44:45,3:12])))
Dstc12 = as.numeric(unlist(as.list(Pscd[47:48,3:12])))

# ignore warning since we don't mind repetition of same value in the same column
Pscd= as.data.frame(cbind(Dstc01,Dstc02,Dstc03,Dstc04,
                          Dstc05,Dstc06,Dstc07,Dstc08,
                          Dstc09,Dstc10,Dstc11,Dstc12))

# assign districts to every charging station and count for each districs
```

```
# compare post code of each Charging station, assgin a district to each
DsLd = rep(0,length(CgstN$Pcd))

for(i in 1:length(CgstN$Pcd)){
    for(j in 1:12){
        if(CgstN$Pcd[i] %in% Pscd[,j]){
            DsLd[i]=j
        }
    }
}
# check if all charging stations is assigned to a district
(DsLd !=0) == length(CgstN$Pcd) # the result should all be False

# count the number of charging station in each district
Nrct=rep(0,12)
for(i in 1:12){
    for(j in 1:length(DsLd)){
        if(DsLd[j]==i)
            Nrct[i]=Nrct[i]+1
    }
};Nrct


#==================================restaurant==================================

Steil = read_html("https://www.berlin.de/restaurants/stadtteile/")

SteilList = Steil %>% html_nodes("br+ .decoda-list a") %>% html_text()

SteilList = Replace(SteilList); SteilList

R_html = c()
for (i in 1: length(SteilList)){
    R_html[i] = paste0("https://www.berlin.de/restaurants/stadtteile/",
                       SteilList[i],"/")
}
Nrrs = rep(0,length(SteilList))

system.time(  # this takes a short while so I calculated the system time
    for (i in 1: length(SteilList)){
        Nrrs[i] = length(read_html(R_html[i]) %>%
                             html_nodes(".main-content .list--arrowlist a") %>%
                             html_text() )
        if(Nrrs[i]==0){
            Nrrs[i] = length(read_html(R_html[i]) %>%
                                 html_nodes(".basis .heading a") %>%
                                 html_text() )
        }
        if(Nrrs[i]==1){
            Nrrs[i] = length(read_html(R_html[i]) %>%
                                 html_nodes(".basis .heading a") %>%
                                 html_text() )
        }
    }); Nrrs  # Nrrs gives the number of each restaurant in each Ortsteil
```

```
RestO = cbind(SteilList, Nrrs)

# calculate nr. of restaurants in every district
Rest = numeric(length = 12)
for(i in 1:length(Nrrs)){
    for(j in 1:12){
        if(RestO[i,'SteilList'] %in% OteilD[,j])
            Rest[j] = Rest[j] + as.numeric(RestO[i,'Nrrs'])
    }
}; Rest


#==============================cycling length==============================

# data obtained from <http://www.stadtentwicklung.berlin.de//geoinformation/fis-broker/

Rad = read_excel(paste0("SPL_BerlinDst_Data_Prep_2/Radverkehrsanlagen.xls"))

# if error appears here, then use:
#read.xlsx(paste0("SPL_BerlinDst_Data_Prep_2/",
#                 "Radverkehrsanlagen.xls"),
#          encoding = "UTF-8",
#          as.data.frame = TRUE)

View(Rad)  # have a look at the data
unique(Rad$`RVA-Typ`)  # have a better understanding of the data

# select the column in dataset we need
Rad = Rad %>% select('Bezirk','L nge␣[m]')

Rad$Bezirk = Replace(Rad$Bezirk)  # replace all German letters in Bezirk

CyllVec = list()  # define a list to store all cycling length in districts

for(i in 1:length(Dstc$District)){
    CyllVec[[i]] = Rad$`L nge [m]`[which(Rad$Bezirk == Dstc$District[i])]
}

Cyll = sapply(CyllVec,sum)  # total length in each districts by summing up
Cyll

#==============================nr.doctor==============================

# Nr. of house doctor per 10,000 people
# data obtained from
# <https://www.berlin.de/ba-lichtenberg/auf-einen-blick/buergerservice/gesundheit/artik

Nrdr = c(66.7,66.6,65.3,86.7,63.2,67.3,73.5,59.5,53.2,61.8,51.8,64.2)

#==========================street crossings==============================

# data obtained from
# <http://www.stadtentwicklung.berlin.de//geoinformation/fis-broker/>
Nrsc = c(33, 39, 54, 35, 22, 31, 19, 25, 32, 39, 22, 30)
```

```
#============================merging the data================================

Nr = 1:12
QiDt = data.frame(Nr,Dstc$District, Nrct, Rest, Cyll, Nrdr, Nrsc)
colnames(QiDt) = c("Nr",
                   "District",
                   "Charging",
                   "Restaurants",
                   "Cycle",
                   "Doctors",
                   "Crossings")

QiDt = as.data.frame(QiDt); QiDt

write.csv(QiDt,"SPL_BerlinDst_Data_Prep_2.csv")
```

11.3 Quantlet 3: SPL$_{B}erlinDst_Data_Prep_3$

Data_Prep_3

```
# Working directory should be set to the project folder (normally by default)
#setwd("")
#=========================RUN HELPFUL FUNCTIONS==============================
#replace all the Umlauts by latin equivalents
ReplaceUmlauts = function(clmn){
  #Description: Replaces german special symbols and turns to lower case
  #Author: Aleksandra Kudaeva
  #Input:  column where you want to replace umlauts
  #Output: column without umlauts (lower case)

  clmn = tolower(clmn)  #all strings to lower case

  #check if at least one element of a vector has any umlauts in it
  #replaces umlauts until there are no one left
  while(any(grepl("  |  |  |  ",clmn)) == TRUE) {
    clmn %<>%
      sub("  ", "ae", .) %<>%
      sub("  ", "oe", .) %<>%
      sub("  ", "ue", .) %<>%
      sub("  ", "ss", .)
  }
  return(clmn)
}

#count NAs
CountMissings=function(column){
  #Description: count NAs
  #Author: Aleksandra Kudaeva
  #Input:  column where you want to count missings
  #Output: number of missings
  sum(ifelse(is.na(column),1,0))
```

```
}

#============================LOAD PACKAGES=====================================
# Install packages
if(!require("rvest")){install.packages("rvest")}
if(!require("readxl")){install.packages("readxl")}
if(!require("magrittr")){install.packages("magrittr")}
if(!require("dplyr")){install.packages("dplyr")}

# Load libraries
library("rvest")
library("readxl")
library("magrittr")
library("dplyr")

#==============================================================================
#============================DOWNLOAD THE DATA=================================
#==============================================================================

#====================1.Street-Index Matching Table============================

# Download list of districts in Berlin
dstr = read.csv2("./SPL_BerlinDst_Data_Prep_3/List_of_districts.csv",
                 header = TRUE,
                 sep    = ";",
                 dec    = ",",
                 stringsAsFactors = FALSE)

# Create table with data for the whole Berlin
StrMtch = data.frame()

# Download street-index matching table from web
for (i in 1:dim(dstr)[1]) {
  # Generate a link to data for all the districts and sub-districts
  link=paste0("https://berlin.kauperts.de/Bezirke/",
              dstr$District[i],
              "/Ortsteile/",
              dstr$Sub.district[i],
              "/Strassen")

  # Download the data from the web-page with generated link
  webpage = read_html(link)
  tbls    = html_nodes(webpage, "table")
  tab     = html_table(tbls)[[1]]

  # Add columns for district and sub-district
  tab$District    = dstr$District[i]
  tab$SubDistrict = dstr$Sub.district[i]

  # Add created table to the table for the whole Berlin
  StrMtch = rbind(StrMtch, tab)
}

# Save resulting table to csv
```

```
write.csv2(StrMtch,
           "./SPL_BerlinDst_Data_Prep_3/Street_Index_Matching.csv",
           row.names = FALSE)


#============================2.Air-Pollution Data==============================

# Read air-pollution data from excel
ap15 = read_excel("./SPL_BerlinDst_Data_Prep_3/Air_Pollution_2015.xls",
                  sheet = 1)

# Original names are too long and contain special symbols and spaces
mtch = read.csv2("./SPL_BerlinDst_Data_Prep_3/matching.csv",
                 sep = ";",
                 stringsAsFactors = FALSE)  # Matching table for short names

names(ap15) = mtch$new[match(names(ap15), mtch$old)]  # Rename variables

#==============================================================================
#============================PROCESS THE DATA==================================
#==============================================================================

#=================1.Street-Index Matching Table=============================

# Street name formatting (in order to merge with air pollution data)
StrMtch$str = StrMtch$Stra e %>%
  ReplaceUmlauts() %>%  # Replace umlauts and switch to lower case
  sub("str.$|str$|-strasse|strasse|-str.$", "", .)  # Delete street indicator

#=========================2.Air-Pollution Data==============================

ap15$Nr = as.numeric(ap15$Nr)  # Reformat steet section number to "numeric"

# Street name formatting (in order to merge with street-index matching table)
ap15$str = ap15$Street %>%
  ReplaceUmlauts() %>%  # Replace umlauts and switch to lower case
  sub("str.$|str$|-strasse|strasse|-str.$", "", .) %>%  # Delete street ind.
  sub("ak␣|as␣|ad␣", "", .)  # Delete AK, AS, AD in the beginning

#==============================================================================
#====================MERGE AND SUMMARIZE THE DATA==============================
#==============================================================================

# Find PM10 and PM15 weighted averages for each street (by length of str.section)
Pltn = ap15 %>%
  group_by(str) %>%
  summarize(L = sum(Length),
            MinPM10 = min(PM10_yearly),
            MaxPM10 = max(PM10_yearly),
            AvgPM10 = weighted.mean(PM10_yearly, Length),
            MinPM25 = min(PM25_yearly),
            MaxPM25 = max(PM25_yearly),
            AvgPM25 = weighted.mean(PM25_yearly, Length)) %>%
  arrange(str)
```

```
# Merge averaged air pollution tables with
PltnM = merge(Pltn, StrMtch, by="str", all.x = TRUE, all.y = FALSE)

# Check how many streets were not merged
CountMissings(PltnM$PLZ)  # 51 streets were not in database for postal indexes

# Some streets are allocated to more than one district
# We assume here that such streets are equally devided among all districts
# to which they belong

ap = PltnM %>%
  filter(!is.na(PLZ)) %>%  #filter NAs
  group_by(str) %>%
  mutate(DistrictLength = L/n()) %>% #appr. length of street in the district
  group_by(District) %>%
  summarize(PM10 = weighted.mean(AvgPM10, DistrictLength),  # av. PM10
            PM25 = weighted.mean(AvgPM25, DistrictLength)) %>%  # av. PM25

  arrange(desc(PM10))

# Save final result as csv
write.csv2(ap,
           "./SPL_BerlinDst_Data_Prep_3/SPL_BerlinDst_Data_Prep_3.csv",
           row.names = FALSE)
```

11.4 Quantlet 4: SPL$_{B}erlinDst_Data_Merge$


```
#======================LOAD PACKAGES=======================================
# Install packages if not installed before
if(!require("pastecs")){install.packages("pastecs")}
if(!require("dplyr")){install.packages("dplyr")}

# Load libraries
library(pastecs)
library(dplyr)

#====================RUN HELPFULL FUNCTIONS ==============================
# Use function from Malgorzata Olesiewitz
DistricToFullName = function (column){
  # Author: Malgorzata Olesiewitz
  # Function re-names the districts of Berlin with their full names
  # without special signs. The function identifies the district by
  # three or four letters of its name.
  #
  # Args:
  #      column: vector or a column containing names of Berlin
  #      districts.Name can have any form, special signs can be
  #      use and additional information can be added to it. Vector
  #      must be character type.Only first 3-4letters need to be correct.
```

```
  #
  # Returns:
  #         Vector of replaced district names by its official names
  #         without special signs.

  column[grepl("mit", column,
               ignore.case = TRUE)] = "Mitte"
  column[grepl("fri", column,
               ignore.case = TRUE)] = "Friedrichshain-Kreuzberg"
  column[grepl("pank", column,
               ignore.case = TRUE)] = "Pankow"
  column[grepl("mar", column,
               ignore.case = TRUE)] = "Marzahn-Hellersdorf"
  column[grepl("char", column,
               ignore.case = TRUE)] = "Charlottenburg-Wilmersdorf"
  column[grepl("spa", column,
               ignore.case = TRUE)] = "Spandau"
  column[grepl("ste", column,
               ignore.case = TRUE)] = "Steglitz-Zehlendorf"
  column[grepl("tem", column,
               ignore.case = TRUE)] = "Tempelhof-Schoneberg"
  column[grepl("trep", column,
               ignore.case = TRUE)] = "Treptow-Kopenick"
  column[grepl("neu", column,
               ignore.case = TRUE)] = "Neukolln"
  column[grepl("lich", column,
               ignore.case = TRUE)] = "Lichtenberg"
  column[grepl("rein", column,
               ignore.case = TRUE)] = "Reinickendorf"

  return(column)
}

#================ READING IN ALL PREPARED DATA SETS ======================
# Working directory is a project directory (if not automatic)
# setwd("")

# Read data from the previous quantlets
dt1 = read.csv2("./SPL_BerlinDst_Data_Merge/SPL_BerlinDst_Data_Prep_1.csv",
                sep = ";",
                dec = ",",
                row.names = 1,
                stringsAsFactors = FALSE)

dt2 = read.csv2("./SPL_BerlinDst_Data_Merge/SPL_BerlinDst_Data_Prep_2.csv",
                sep = ",",
                dec = ".",
                row.names = 1,
                stringsAsFactors = FALSE)

dt3 = read.csv2("./SPL_BerlinDst_Data_Merge/SPL_BerlinDst_Data_Prep_3.csv",
                sep = ";",
                dec = ",",
                stringsAsFactors = FALSE)
```

```
#=================== MERGE THE DATA SETS ================================
# Change districts name to the standard writing
dt2$District = DistricToFullName(dt2$District)
dt3$District = DistricToFullName(dt3$District)

# Merge loaded data sets
lvbInDt = merge(dt1,
                dt2[,-1],  #delete district number in the first column
                by = "District") %>%
    merge(dt3, by = "District")

# Save file to csv

write.csv2(lvbInDt, "./SPL_BerlinDst_Data_Merge/SPL_BerlinDst_Data_Merge.csv")

#==================== DATA ANALYSIS =======================================
# Produce main statistics of the data

stats = lvbInDt %>%
    select(-c(District, Nr)) %>%
    stat.desc()  #aggregated statistics
```

11.5 Quantlet 5: SPL$_{B}erlinDst_Liv_Index_Calc$

Liv_Index_Calc

```
#====================== PREPARING THE ENVIROMENT ================================
#setwd("~/SPL_BerlinDst")
# The code was tested at HU PC Pool 25

install.packages("ggplot2")
install.packages("reshape2")
install.packages("rlang")
install.packages("dplyr")
install.packages("tidyr")
install.packages("xlsx")
install.packages("ggthemes")
install.packages("xtable")
install.packages("magrittr")   # Install pakages required

library(ggplot2)
library(magrittr)
library(rlang)
library(dplyr)
library(tidyr)
library(ggthemes)
library(xlsx)
library(reshape2)
library(xtable)  # Read in required packages

options(xtable.floating = FALSE)
```

```
options(xtable.timestamp = "")
#==================== READING IN THE  DATA SETS ================================

lvbInDt = read.csv("SPL_BerlinDst_Liv_Index_Calc/SPL_BerlinDst_Data_Desc.csv",
                sep = ";", dec = ",", row.names = 1, stringsAsFactors = FALSE)


#====================PREPERATION OF USEFUL FUNCTIONS ============================

PerCapita = function(x){
    # Function devides the data per number of inhabitants of given district to
    # obtain per capita value
    #
    #Args:
    #    x: the column of which the data should devided per number of citizens
    #
    # Returns:
    # The vector of data x devided per number of inhabitants of given district

  x/lvbInDt$Population
}

PerHa = function(x){
    # Function devides the data by size in ha of given district to obtain
    # per ha value
    #
    #Args:
    #    x: the column of which the data should devided by size of the district
    #
    # Returns:
    # The vector of data x devided by size of the district in ha

  x/lvbInDt$Size
}

NormalizePositive = function(x){
    # Function normalizes the data in the vector x , by assigning the values
    # between 0 and 1 for every observation. The highest value among the
    # vector recives 1, the lowest 0.
    #
    # Args:
    #      x: the vectors of which values are to be normalized. The data type
    #      must be numeric.
    #
    # Returns:
    # Vector of normalized data of vector x
    (x-min(x))/(max(x)-min(x))
}

NormalizeNegative=function(x){
    # Function normalizes the data in the vector x , by assigning the values
    # between 0 and 1 for every observation.. The highest value among the vector
    # recives 0, the lowest 1.
    #
```

```
    # Args:
    #      x: the vectors of which values are to be normalized. The data type
    #      must be numeric.
    #
    # Returns:
    # Vector of normalised data of vector x
    (max(x)-x)/(max(x)-min(x))
}

IsOutlier =  function(y) {
    # Function checks if the given observation in vector y is an outlier or not.
    # The outlier is defined as an observation smaller than the first quantile
    # minus 1,5 times the interquartile range of the vector OR bigger than than
    # the third quantile plus the 1.5 times the interquartile range of the vector.
    # The interquartile range is calculated using IQR function.
    # For more information see ?IQR.
    #
    # Agrs:
    #      y: vector of data for which outliers are to be found. The type of
    #         data must be numeric.
    #
    # Returns: Logical statment "TRUE" for the outlier, "FALSE"  otherwise

    return(y < quantile(y,
                        0.25) - 1.5 * IQR(y) |
               y > quantile(y, 0.75) + 1.5 * IQR(y))
}
#======================CALCULATING INDEX INDICATORS============================

#Create indicators data frame

indDt = data.frame(lvSpc = PerCapita(lvbInDt$SpacePC),  # Calc. liv space/cap.
                   hsAv   = PerCapita(lvbInDt$Flats),  # Calc. nr of flats/cap.
                   dns    = PerHa(lvbInDt$Population),  # Calc. population per ha
                   hsAl   = lvbInDt$Household, # Choose hous. all.
                   trnDn  = PerHa(lvbInDt$Transport),  # Calc. den. of pub. trans.
                   bkLn   = PerHa(lvbInDt$Cycle),  # Calc. den. of cycling lines
                   crChr  = PerHa(lvbInDt$Charging),  #  Calc.e-car char.stat./ha
                   prkSp  = PerCapita(lvbInDt$Parking),  # Calc.nr park. spc./ cap
                   trs    = PerCapita(lvbInDt$Tourists),  # Calc.nr of tour. /cap.
                   # Calculate the hotel occupancy by deviding the total num.
                   # of stays by total capacity; number of beds multiplied
                   # by 365 days in the year
                   htlOc  = lvbInDt$Stays/(lvbInDt$Hotel*365),
                   sprCl  = PerHa(lvbInDt$Sport),  # Choose nr of sport clubs/ ha
                   res    = PerHa(lvbInDt$Restaurants),  # Cal.nr of restaur./ha
                   std    = lvbInDt$Puplis1000,  # Choose nr of pupils/ K of ppl
                   grdSz  = lvbInDt$Pupils/lvbInDt$Grades,  # Cal. avg. grade size
                   chU3   = lvbInDt$Child3,  # Choose % of kids < 3y/o in daycare
                   chU6   = lvbInDt$Child6,  # Choose % of kids 3-6y/o in daycare
                   doc    = PerCapita(lvbInDt$Doctors),  # Cal.num. of doctors/cap
                   actSn  = lvbInDt$SenSport,  # Choose % of active seniors
                   actJn  = lvbInDt$JunSport,  # Choose % of active juniors
                   trf    = PerCapita(lvbInDt$Accidents),  # Calc. traff. acc/cap
```

```
                  strCr  = PerHa(lvbInDt$Crossings),  # Calc.nr street cross/ha
                  crm    = lvbInDt$Crime,  # Choose criminal offences per 10K ppl
                  socHl  = PerCapita(lvbInDt$SocHelp),  # Cal.soc. help rec. /cap
                  dsb    = lvbInDt$Disabled1000, # Choose sev. handicapped/1K ppl
                  emp    = lvbInDt$Employment,  # Choose emp./ 1K ppl cap. of emp.
                  comp   = lvbInDt$Company,  # Choose nr of companies
                  txRv   = lvbInDt$Revenue,  # Choose taxable revenue
                  # Cal. nr bankr./nr.comp.
                  bnk    = (lvbInDt$Bankruptcy)/(lvbInDt$Company),
                  grSp   = PerHa(lvbInDt$GreenSp),  #Cal. green space / ha
                  # Cal. ratio of agr surface vs. residential & traffic surf.
                  agrRe  = (lvbInDt$AgrSurface)/(lvbInDt$RsSurface),
                  tr     = lvbInDt$Trees, # Choose nr of trees per km of the road
                  pm10   = lvbInDt$PM10,  # Choose PM10 avg. level
                  pm25   = lvbInDt$PM25)  # # Choose PM2.5 avg. level


#============================NEGATIVE INDICATORS================================

# Most of the indicators contribute positively to the index, the higher the
# value the better the district should be rank. Here are 7 indicators, which
# contribute negatively to the index calculation:


ngtInd = list(a = which(colnames(indDt)=="dns"),
              b = which(colnames(indDt)=="trf"),
              c = which(colnames(indDt)=="bnk"),
              d = which(colnames(indDt)=="crm"),
              e = which(colnames(indDt)=="grdSz"),
              f = which(colnames(indDt)=="pm25"),
              g = which(colnames(indDt)=="pm10"))


#=========================NORMALIZING THE DATA ================================

# Normalize the data according to whether the indicators contribute positivly or
# negatively to the Index. For each indicatore, strore the score in new column
# where letters "Scr" are added to the original name of the indicator.

for (i in 1:(ncol(indDt))){  # Run function for every column in the indDt file
    if ((i) %in% ngtInd){
        indDt[,paste(colnames(indDt[i]),"Scr")] =  # create new col. with "score"
             NormalizeNegative(indDt[i])  # If indicator in ngtInd use Neg. fun
    } else {
          indDt[,paste(colnames(indDt[i]),"Scr")] =
               NormalizePositive(indDt[i]) # If not in ngtInd use Pos. fun
    }
}

#==============  CREATING FINAL DATA FRAME FOR INDEX CALCULATION ==============

Nr = lvbInDt$Nr  # Choose district numbers

District = lvbInDt$District  # Choose district names
```

```
# Create the final data frame and select only the variables' scores and not
# absolute values

indScrDt = data.frame(Nr, District,indDt,stringsAsFactors = FALSE) %>%
    select("Nr", "District", grep("Scr", names(.)))

# Write csv file with the data

write.csv2(indScrDt,"SPL_BerlinDst_Liv_Index_Calc/Index_Sore_Data.csv")

# Read in best with:
# read.csv("SPL_BerlinDst_Liv_Index_Calc/Index_Sore_Data.csv",
# sep = ";", dec = ",", row.names = 1, stringsAsFactors = FALSE)

#======================== WEIGHTS OF EACH SUB-INDEX ============================

phys1W = c(0.10)
phys2W = c(0.15)
socW   = c(0.25)
ecoW   = c(0.25)
envW   = c(0.25)

#==================DEVIDING INDICATORS ACCORDING TO THE PILLARS================


phy1Ind = indScrDt %>%
    select("lvSpc.Scr",
           "hsAv.Scr",
           "dns.Scr",
           "hsAl.Scr")  # Select Indicators of Physical 1 Sub-Index

phy2Ind = indScrDt %>%
    select( "trnDn.Scr",
            "bkLn.Scr",
            "crChr.Scr",
            "prkSp.Scr") # Select Indicators of Physical 2 Sub-Index

socInd = indScrDt %>%
    select("trs.Scr",
           "htlOc.Scr",
           "sprCl.Scr",
           "res.Scr",
           "std.Scr",
           "grdSz.Scr",
           "chU3.Scr",
           "chU6.Scr",
           "doc.Scr",
           "actSn.Scr",
           "actJn.Scr",
           "trf.Scr",
           "strCr.Scr",
           "crm.Scr",
           "socHl.Scr",
```

```
           "dsb.Scr")  # Select Indicators of Social Sub-Index

ecoInd = indScrDt %>%
    select("emp.Scr",
           "comp.Scr",
           "txRv.Scr",
           "bnk.Scr")  # Select Indicators of Economic Sub-Index

envInd = indScrDt %>%
    select("grSp.Scr",
           "agrRe.Scr",
           "tr.Scr",
           "pm10.Scr",
           "pm25.Scr")  # Select Indicators of Enviromental Sub-Index

# ======================CALCULATING SUB-INDEXES================================

Phy1In = apply(phy1Ind, 1, sum)
Phy2In = apply(phy2Ind, 1, sum)
SocIn  = apply(socInd, 1, sum)
EcoIn  = apply(ecoInd, 1, sum)
EnvIn  = apply(envInd, 1, sum)


# ======================CALCULATING LIVIBILITY INDEX ==========================

# Calculate the contribution of each pillar to the Liveability Index

Pllrs= data.frame(PhyPl   = (Phy1In*phys1W + Phy2In*phys2W),
                  SocPl = SocIn*socW,
                  EcoPl = EcoIn*ecoW,
                  EnvPl = EnvIn*envW)

# Calculate total Liveability Index

TotalIn = apply(Pllrs,1, FUN = sum)

# Creat final data frame with the results

RsltDt = data.frame(District,
                    Phy1In,
                    Phy2In,
                    SocIn,
                    EcoIn,
                    EnvIn,
                    Pllrs,
                    TotalIn)
# Write csv file

write.csv2(RsltDt, "SPL_BerlinDst_Liv_Index_Calc/SPL_BerlinDst_Liv_Index.csv")

# Read in best with:
# read.csv("SPL_BerlinDst_Liv_Index_Calc/SPL_BerlinDst_Liv_Index.csv",
# sep = ";", dec = ",", row.names = 1, stringsAsFactors = FALSE)
```

```
#====================== CALCULATE MAXIMUM SCORE ==============================

MaxScoreIndex = sum(length(phy1Ind)*phys1W,length(phy2Ind)*phys2W,
                    length(socInd)*socW, length(ecoInd)*ecoW,
                    length(envInd)*envW)  # Calculate max score of Total Index

# Creat data frame with max score for each result

MaxScore = data.frame("Max Score",
                      length(phy1Ind),
                      length(phy2Ind),
                      length(socInd),
                      length(ecoInd),
                      length(envInd),
                      (length(phy1Ind)*phys1W + length(phy2Ind)*phys2W),
                      length(socInd)*socW,
                      length(ecoInd)*ecoW,
                      length(envInd)*envW,
                      MaxScoreIndex,
                      stringsAsFactors = FALSE)

colnames(MaxScore) = colnames(RsltDt)  #  sure that the col. names are the same

#============================== RESULTS ========================================
# See results for sub-Indexes, Pilars and TotaL Liveability Index

View(RsltDt)

# Get the latex code for the report

xtable(RsltDt)
xtable(MaxScore)


#============================== BOXPLOT ========================================

subMlt = melt(RsltDt[, -c(7:11)],id.vars = "District")  # Melt Sub-Index Results

Districtlable = as.character(subMlt$District)  # Convert district names to chr.

# Create final data frame for plotting, group the data according to the
# Sub-Index (variable) and creat new column which checks for outliers and
# if "TRUE" returns the name of the District

subMltDt = data.frame(Districtlable,subMlt[, -1],stringsAsFactors = FALSE)%>%
    group_by(variable) %>%
    mutate(Outlier = ifelse(IsOutlier(value), Districtlable , ""))


ggplot(subMltDt, aes(x=variable,y=value, group = variable)) +  # Create ggplot
    geom_boxplot(aes(fill = variable)) +  # Create boxplot for every Sub-Index
    # Add District lebels for outliers
    geom_text(aes(label = Outlier), size = 3,  vjust = - 0.5) + # Plot the outliers
```

```
    theme_bw() +  # Choose black and white theme
    scale_fill_economist(labels = c("Housing","Infrastructure",  # Choose color palet
                                    "Social", "Economic",
                                    "Enviromental")) +  # Names the labes
    labs(x = "Sub-Index", y = "Score") +  # Name the axises
    theme(panel.grid.minor = element_blank(),  # Remove the minor grid
          panel.grid.major = element_blank(),  # Remove the major grid
          axis.title.x     = element_text(size = 10), # X axis lebel font size
          axis.title.y     = element_text(size = 10), # Y axis lebel font size
          axis.text.x      = element_blank(),  # Remove x axis labels
          legend.title     = element_blank(), # Remove legent title
          legend.text      = element_text(size = 9),  # Legend font size
          legend.position  = "bottom",  # Place the legend on the graph
          legend.box       = "horizontal")  # Horizontally

# Print the plot

ggsave("Sub-Index Boxplot.png", plot = last_plot(),scale = 1, device = "png",
       path = "SPL_BerlinDst_Liv_Index_Calc/")

dev.off()

#============================ BAR PLOTS =======================================

TtlMltDt = melt(RsltDt[, -c(2:6,11)],id.vars = "District") # Melt Pillars Data


# Creat showing all individual pillars

ggplot(TtlMltDt, aes(x = District,y = value, fill = variable)) +  # Creat ggplot
    geom_bar(stat = "identity", position = "dodge", width = 0.8)+  # Creat barplot
    theme_bw() +  # Use black and white theme
    scale_fill_economist(labels = c("Physical","Social",# Chose color palet
                                    "Economic", "Enviromental")) +  # Chose color palet
    labs(y = "Liveability Index: Pillars Comparison") +  # Add x axis label
    theme(panel.grid.minor = element_blank(),  # Remove the minor grid
          panel.grid.major = element_blank(),  # Remove the major grid
          legend.position  = "bottom",  # Place the legend on the graph
          legend.box       = "horizontal",  # Horizontally
          axis.title.x     = element_text(size = 10), # X axis lebel font size
          legend.title     = element_text(size = 8),  # Legend title font size
          legend.text      = element_text(size = 8),  # Legend font size
          axis.title.y     = element_blank()) +  # Remove y axis labels
    guides(fill=guide_legend(title="Pillars:")) +  # Add legent title
    coord_flip()  # Flip the chart to be horizontal

# Print the plot

ggsave("Per Pillar BarPlot.png", plot = last_plot(),scale = 1, device = "png",
       path = "SPL_BerlinDst_Liv_Index_Calc/")

dev.off()

# Creat Cummulative Index Plot
```

```
ggplot(TtlMltDt, aes(x = District,y = value, fill = variable)) +  # Creat ggplot
    geom_bar(stat = "identity", width = 0.5)+  # Creat barplot
    theme_bw() +  # Use black and white theme
    scale_fill_economist(labels = c("Physical", "Social",  # Chose color palet
                                    "Economic", "Enviromental")) +  # Name labels
    labs(y = "Total␣Liveability␣Index") +  # Add x axis label
    theme(panel.grid.minor = element_blank(),  # Remove the minor grid
          panel.grid.major = element_blank(),  # Remove the major grid
          legend.position  = "bottom",  # Place the legend on the graph
          legend.box       = "horizontal",  # Horizontally
          axis.title.x     = element_text(size = 10), # X axis lebel font size
          legend.title     = element_text(size = 8),  # Legend title font size
          legend.text      = element_text(size = 8),  # Legend font size
          axis.title.y     = element_blank()) +  # Remove y axis labels
    guides(fill=guide_legend(title="Pillars:")) +  # Add legent title
    coord_flip()  # Flip the chart to be horizontal

# Print the plot

ggsave("Total␣Index␣BarPlot.png", plot = last_plot(),scale = 1, device = "png",
       path = "SPL_BerlinDst_Liv_Index_Calc/")

dev.off()

# ========================END OF THE SCRIPT=====================================
# ==============================================================================
```

11.6 Quantlet 6: SPL_BerlinDst_Rent_Analysis


```
#=======================LOAD PACKAGES=========================================
# Install packages if not installed before
if(!require("ggplot2")){install.packages("ggplot2")}
if(!require("sp")){install.packages("sp")}
if(!require("mapproj")){install.packages("mapproj")}
if(!require("maps")){install.packages("maps")}
if(!require("rgdal")){install.packages("rgdal")}
if(!require("ggthemes")){install.packages("ggthemes")}
if(!require("ggmap")){install.packages("ggmap")}
if(!require("osmdata")){install.packages("osmdata")}
if(!require("ggrepel")){install.packages("ggrepel")}

# Load libraries
library("ggplot2")
library("ggmap")
library("sp")
library("maps")
library("mapproj")
library("osmdata")
library("rgdal")
```

```
library("dplyr")
library("readxl")
library("ggthemes")
library("ggrepel")

#====================RUN HELPFULL FUNCTIONS ============================
# Use function from Malgorzata Olesiewitz
DistricToFullName = function (column){
    # Author: Malgorzata Olesiewitz
    # Function re-names the districts of Berlin with their full names
    # without special signs. The function identifies the district by
    # three or four letters of its name.
    #
    # Args:
    #      column: vector or a column containing names of Berlin
    #      districts.Name can have any form, special signs can be
    #      use and additional information can be added to it. Vector
    #      must be character type.Only first 3-4letters need to be correct.
    #
    # Returns:
    #        Vector of replaced district names by its official names
    #        without special signs.

    column[grepl("mit", column,
                 ignore.case = TRUE)] = "Mitte"
    column[grepl("fri", column,
                 ignore.case = TRUE)] = "Friedrichshain-Kreuzberg"
    column[grepl("pank", column,
                 ignore.case = TRUE)] = "Pankow"
    column[grepl("mar", column,
                 ignore.case = TRUE)] = "Marzahn-Hellersdorf"
    column[grepl("char", column,
                 ignore.case = TRUE)] = "Charlottenburg-Wilmersdorf"
    column[grepl("spa", column,
                 ignore.case = TRUE)] = "Spandau"
    column[grepl("ste", column,
                 ignore.case = TRUE)] = "Steglitz-Zehlendorf"
    column[grepl("tem", column,
                 ignore.case = TRUE)] = "Tempelhof-Schoneberg"
    column[grepl("trep", column,
                 ignore.case = TRUE)] = "Treptow-Kopenick"
    column[grepl("neu", column,
                 ignore.case = TRUE)] = "Neukolln"
    column[grepl("lich", column,
                 ignore.case = TRUE)] = "Lichtenberg"
    column[grepl("rein", column,
                 ignore.case = TRUE)] = "Reinickendorf"

    return(column)
}

#====================DOwNLOAD AND TRANSFORM DATA=========================
# Get Berlin Boundaries from OSM data
ber = getbb("berlin")
```

```
# Automatic bbox cut some parts of berlin out, that is why I manually adjust x
# coordinate. Adjustments are defined manually.
ber_adj = ber
ber_adj["x","min"] = ber_adj["x","min"]-0.18
ber_adj["x","max"] = ber_adj["x","max"]+0.22
ber_adj["y","min"] = ber_adj["y","min"]-0.02

# Download map teils from stamen maps
map = get_stamenmap(ber_adj, maptype="toner-lite", zoom = 11)

# Download and transform polygon data for district borders
test = readOGR("./SPL_BerlinDst_Rent_Analysis/bezirksgrenzen.kml")
Bezirk = fortify(test)

# Read and transform KML border of West Berlin
wall = readOGR(paste0("./SPL_BerlinDst_Rent_Analysis",
                      "/Berliner_Mauer_Hinterlandmauer.kml"))
Wall = fortify(wall)

# Assign names of districts
Bezirk$District = case_when(Bezirk$id == "0" ~ "Reinickendorf",
                            Bezirk$id == "1" ~ "Charlottenburg-Wilmersdorf",
                            Bezirk$id == "10" ~ "Friedrichshain-Kreuzberg",
                            Bezirk$id == "11" ~ "Tempelhof-Schoneberg",
                            Bezirk$id == "2" ~ "Treptow-Kopenick",
                            Bezirk$id == "3" ~ "Pankow",
                            Bezirk$id == "5" ~ "Lichtenberg",
                            Bezirk$id == "4" ~ "Neukolln",
                            Bezirk$id == "6" ~ "Marzahn-Hellersdorf",
                            Bezirk$id == "7" ~ "Spandau",
                            Bezirk$id == "8" ~ "Steglitz-Zehlendorf",
                            Bezirk$id == "9" ~ "Mitte",
                            TRUE ~ "other")

# Download livability index table created in previous quantlet
IndDt = read.csv2("./SPL_BerlinDst_Rent_Analysis/SPL_BerlinDst_Liv_Index.csv",
                  sep = ";",
                  dec = ",")

# Download rent data
RntDt = read_excel(paste0("./SPL_BerlinDst_Rent_Analysis/statistic_id259905",
                          "_mietpreise-in-berlin-2017-nach-bezirken.xlsx"),
                   sheet = "Daten",
                   skip = 5,
                   col_names = FALSE) %>%
  setNames(c("District", "Rent")) %>%
  filter(!District == "Berlin␣Durchschnitt")

# Change districts name to the standard writing
RntDt$District = DistricToFullName(RntDt$District)

# Merge polygon data with index and rent
IndRntDt = Bezirk %>%
```

```
    merge(IndDt, by = "District") %>%
    merge(RntDt, by = "District")

# Calculate coordinates of centroids for each district
LablesDt = aggregate(cbind(long, lat) ~ District,
                     data=Bezirk,
                     mean) %>%
  merge(IndDt, by = "District") %>%
  merge(RntDt, by = "District")

#======================DISTRICTS MAP=======================================
p1 <- ggmap(map, extent = "normal", maprange = FALSE) +
  geom_polygon(data = IndRntDt,
               aes(long, lat, group = group, fill=District),
               colour = "red",
               alpha = 0.2) +
  theme_bw() +
  coord_map(projection="mercator",
            xlim = c(attr(map, "bb")$ll.lon, attr(map, "bb")$ur.lon),
            ylim = c(attr(map, "bb")$ll.lat, attr(map, "bb")$ur.lat))

# Save the plot
ggsave("BerlinDistrictsMap.png",
       plot = p1,
       scale = 1,
       device = "png",
       path = "SPL_BerlinDst_Rent_Analysis/")

dev.off()

#====================TOTAL INDEX - RENT HEATMAP============================
p2 = ggplot(IndRntDt,
            aes(long, lat, group = group))+
  geom_polygon(aes(fill = TotalIn),
               colour = "bisque4")+
  scale_fill_gradient(low = "snow1",
                      high = "darkcyan",
                      name = "Liveability␣Index") +
  theme_bw() +
  theme(panel.border = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line  = element_blank(),
        axis.ticks = element_blank(),
        axis.text  = element_blank(),
        axis.title = element_blank())+
  scale_color_gradient(low = "paleturquoise",
                       high = "rosybrown2",
                       guide=FALSE) +
  geom_label_repel(data = LablesDt,
             aes(long, lat, label = District, group = District),
             size = 3,
             box.padding = 2,
             segment.color = 'grey24') +
```

```
  geom_point(data = LablesDt,
             aes(long, lat,
                 colour = Rent,
                 group = District),
             size = 13) +
  geom_point(data = LablesDt,
             aes(long, lat,
                 group = District),
             size = 13,
             shape = 1,
             colour = "grey24") +
  geom_text(data = LablesDt,
            aes(long, lat, label = paste0(Rent,"   "), group = District),
            size = 3)

# Save the plot
ggsave("IndexRent.png",
       plot = p2,
       scale = 1,
       device = "png",
       path = "SPL_BerlinDst_Rent_Analysis/")

dev.off()

#====================INDEX RENT ANALYSIS==================================

ggplot(LablesDt, aes(x=Rent, y=TotalIn)) +
  geom_point(size = 3,
             colour = "steelblue") +  # Use hollow circles
  geom_smooth(method = lm,  # Add linear regression line
              se = FALSE) +  # Don't add shaded confidence region
  theme_bw()+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(y = "Liveability␣Index") +
  geom_label_repel(data = LablesDt,
                   aes(Rent, TotalIn, label = District, group = District),
                   size = 4,
                   box.padding = 0.5)

# Save the plot
ggsave("IndexRentScatter.png",
       plot = last_plot(),
       scale = 1,
       device = "png",
       path = "SPL_BerlinDst_Rent_Analysis/")

dev.off()


#calculate correlation between Rent and Total Index
cor(LablesDt$TotalIn, LablesDt$Rent)

#run regression of rent on calculated total index and print regression summary
```

```
LablesDt %>%
  lm(Rent~TotalIn, .) %>%
  summary()

#==================EAST-WEST BERLIN COMPARISON==============================

# Define lists of districts which belonged to east and west Berlin
# (Omit Mitte and Fridrichshein-Kreuzberg as they are in both East and West)
east = c("Pankow",
         "Lichtenberg",
         "Treptow-Kopenick",
         "Marzahn-Hellersdorf")
west = c("Tempelhof-Schoneberg",
         "Reinickendorf",
         "Charlottenburg-Wilmersdorf",
         "Steglitz-Zehlendorf",
         "Neukolln")

# Define centroids of east and west Berlin
LablesCnt = LablesDt %>%
  mutate(Part = ifelse(District %in% east, "East Berlin",
                       ifelse(District %in% west, "West Berlin", NA))) %>%
  group_by(Part) %>%
  summarize(lat = mean(lat),
            long = mean(long)) %>%
  filter(!is.na(Part))

# For repeated plots I write a function with all the settings
EastWest = function(variable, title, name, leg = TRUE) {
    # Author: Aleksandra Kudaeva
    # Description: function creates a heatmap of Berlin with highlighted
    #              Berlin Wall border according to the given parameter
    # Input: Table and column, which will determine coloring,
    #        Title of legend
    #        Name of png file which is going to be saved
    #        Legend: by default - TRUE
    # Output: Heatmap, printed and saved in png
    plot = ggplot(IndRntDt,
                  aes(long, lat, group = group)) +
           geom_polygon(aes(fill = variable),
                        colour = "darkcyan",
                        show.legend = leg) +
           geom_path(data = Wall, aes(x = long, y = lat),
                     color="bisque4",
                     size = 1.5) +
           scale_fill_gradient(low = "snow1",
                     high = "darkcyan",
                     name = title) +
           theme_bw() +
           theme(panel.border = element_blank(),
                 panel.grid.major = element_blank(),
                 panel.grid.minor = element_blank(),
                 axis.line  = element_blank(),
                 axis.ticks = element_blank(),
```

```
                axis.text  = element_blank(),
                axis.title = element_blank()) +
          geom_label(data = LablesCnt,
                     aes(long, lat, label = Part, group = Part),
                     size = 6)
    ggsave(paste0(name, ".png"),
           plot = plot,
           scale = 1,
           device = "png",
           path = "SPL_BerlinDst_Rent_Analysis/")

    return(plot)
}

# Plot Index together with Berlin Wall
EastWest(IndRntDt$TotalIn, "Liveability␣Index", "IndexEastWest", leg = TRUE)

# Plot Rent with Berlin wall, without legend
EastWest(IndRntDt$Rent, "Rent", "RentEastWest", leg = FALSE)

# Plot subindixes and Berlin Wall, without legend
EastWest(IndRntDt$Phy1In, "Housing", "HousingEastWest", leg = FALSE )
EastWest(IndRntDt$Phy2In, "Infrastructure", "InfrEastWest",leg = FALSE)
EastWest(IndRntDt$SocIn,  "Social␣Index", "SocEastWest", leg = FALSE)
EastWest(IndRntDt$EcoIn,  "Economic␣Index", "EconEastWest", leg = FALSE)
EastWest(IndRntDt$EnvIn,  "Environmental␣Index", "EcolEastWest", leg = FALSE)
```

11.7 Quantlet 7: SPL_BerlinDst_PCA_Cluster


```
#========================= Set working directory============================

setwd("~/SPL_BerlinDst")

#================================install packages==========================
install_github("vqv/ggbiplot")
# error: "failed to set default locale"
# solution: system('defaults write org.R-project.R force.LANG en_US.UTF-8'),
# and then restart R
install.packages("devtools")
install.packages("factoextra")
install.packages("ggthemes")
install.packages("readr")
install.packages("ggplot2")
install.packages("ggrepel")
install.packages("purrr")
install.packages("ggpubr")
install.packages("dplyr")
install.packages("cluster")
install.packages("xlsx")
```

```
library(ggbiplot)
library(devtools)
library(factoextra)
library(ggthemes)
library(readr)
library(ggplot2)
library(ggrepel)
library(purrr)
library(ggpubr)
library(dplyr)
library(cluster)
library(xlsx)

#======================== Set working directory=============================

setwd("~/SPL_BerlinDst")

#============================Import the data================================
options(digits=6)

# Physical1; Physical2; Social; Economical; enviromental Index
# and the respectively weighted Index

Index = read.csv(paste0("SPL_BerlinDst_Liv_Index_Calc/",
                        "SPL_BerlinDst_Liv_Index.csv"),
                 sep = ";", dec = ",", row.names = 1,
                 stringsAsFactors = FALSE)
View(Index)
#Full data set with normalised data as individual index
IndexFull = read.csv("SPL_BerlinDst_Liv_Index_Calc/Index_Score_Data.csv",
                     sep = ";", dec = ",", row.names = 1,
                     stringsAsFactors = FALSE)
View(IndexFull)

#==========================principle component============================

data.pca = prcomp(select(IndexFull,ends_with('Scr')),
                  center = TRUE,
                  scale = TRUE)
#centure: whether the variables should be shifted to be zero centered
#scale: whether the variables should be scaled to have unit variance
summary(data.pca)  # Importance of components

# each component explains a percentage of the total variation in the dataset.

# Extract and visualize the eigenvalues/variances of dimensions,
# Show the percentage of variances explained by each principal component.
fviz_eig(data.pca)  +
  theme_bw()  +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_blank(),
        panel.border = element_blank())
```

```
# Graph of observations(individuals), use the first 2 Principle compoonent
# to visualize the 'distance' between our 12 obsersions
p1 = fviz_pca_ind(data.pca,
                  col.ind = "cos2", # Color by the quality of representation
                  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
                  repel = TRUE) +     # Avoid text overlapping
                  geom_text_repel(aes(label=IndexFull$District),size=3.5)+
                  theme_bw()  +
                  theme(panel.grid.minor = element_blank(),
                  panel.grid.major = element_blank(),
                  panel.border = element_blank())

# Graph of variables. Positive correlated variables point to the same side
# Negative correlated variables point to opposite sides of the graph.
# chosed only 10 most important variables for visualization
p2 = fviz_pca_var(data.pca,
             col.var = "contrib", # Color by contributions to the PC
             select.var = list(contrib = 10),
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)+      # Avoid text overlapping
             theme_bw()  +
             theme(panel.grid.minor = element_blank(),
             panel.grid.major = element_blank(),
             panel.border = element_blank())

#Graph of variables and observations with only the first 10 important variables
p3 = fviz_pca_biplot(data.pca, label="var",
                col.var = "contrib",
                select.var = list(contrib = 10),
                repel = TRUE,
                gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))  +
                geom_text_repel(aes(label=IndexFull$District),size=3.5)+
                theme_bw()  +
                theme(panel.grid.minor = element_blank(),
                panel.grid.major = element_blank(),
                panel.border = element_blank())

#Plot all graphs
ggarrange(p1,p2)
p3

# get a closer look at the diffrent level of contributions of individuals
fviz_contrib(data.pca, choice = "ind", axes = 1) +
  coord_flip() +
  geom_text_repel(aes(label=IndexFull$District))+
  theme_bw()  +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_blank(),
        panel.border = element_blank())

#======================prepare data for Clustering=========================

# select colomn of 5 original Index, i.e. not weighted
# exclude Total Index, since it is just weighted sum of Index
```

```
CIndex = select(Index,ends_with('In'))[,-grep('TotalIn',
                                        colnames(select(Index,ends_with('In'))))]

# select column of 4 weighted Index, which are weighted with
# phys1W = 0.10, phys2W = 0.15,socW = 0.25, ecoW = 0.25, envW = 0.25
CPilar = select(Index,ends_with('Pl'))

# every column in Index
CInPi = Index[,-grep('District',colnames(Index))]

# select column from the full Index dataset, the score of each variable
# exclude the first 2 column with district names and number
CFull = select(IndexFull,ends_with('Scr'))

CGropu = c(CIndex,CPilar,CInPi,CFull)

# have a look at the distance Matrix
get_dist(CIndex, method = "euclidean")
get_dist(CPilar, method = "euclidean")
get_dist(CInPi, method = "euclidean")
get_dist(CFull, method = "euclidean")

# visualization of distance Matrix
fviz_dist(get_dist(CFull),
          gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))


#====================Optimal Cluster numbers================================

#Average Silhouette Method

AvgSil = function(k, df) {
    # Function computes the average Silhouette width
    # The optimal number of clusters k is the one that maximizes
    # the average silhouette over a range of possible values for k
    #
    # Arg:
    # k is number of clusters; df is the dataframe that we apply kmean cluster
    #
    # Output:
    # mean value of silhouette width for one specific k value
    km = kmeans(df, centers = k, nstart = 25)
    s = silhouette(km$cluster, dist(df))
    # silhouette : compute Silhouette Information from Clustering
    # returns 3 columns : cluster, neighbor, sil_width
    mean(s[, 'sil_width'])
}

k = c(2:10)  # for function 'silhouette' we need cluster number to be at least 2
df = CIndex
#df = CPilar
#df = CInPi
#df = CFull
AvgSil_values = sapply(k, FUN = AvgSil, df)
```

```
k[which(AvgSil_values == max(AvgSil_values))]  # optimal number of cluster is 2

fviz_nbclust(CIndex, kmeans, method = "silhouette") + theme(title = element_blank())

# Visualizing the Optimal Number of Clusters
f1 = fviz_nbclust(CIndex, kmeans, method = "silhouette")  # optimal cluster is 2
f2 = fviz_nbclust(CPilar, kmeans, method = "silhouette") # optimal cluster is 2
f3 = fviz_nbclust(CInPi, kmeans, method = "silhouette") # optimal cluster is 4
f4 = fviz_nbclust(CFull, kmeans, method = "silhouette") # optimal cluster is 2

ggarrange(f1,f2,f3,f4,labels = c('1','2','3','4'))

#==============================k mean clustering=========================

kmCIndex = kmeans(CIndex,
            centers = 2, # we set cluster number to be 2 first,
            # to see if there would be an east-west Berlin cluster
            iter.max = 20, nstart = 12,
            algorithm = "Hartigan-Wong",trace=FALSE)

kmCFull = kmeans(CFull,
           centers = 2,
           iter.max = 20, nstart = 12,
           algorithm = "Hartigan-Wong",trace=FALSE)

kmCInPi = kmeans(CInPi,
                 centers = 4,
                 iter.max = 20, nstart = 12,
                 algorithm = "Hartigan-Wong",trace=FALSE)

kmCPilar = kmeans(CPilar,
                  centers = 2,
                  iter.max = 20, nstart = 12,
                  algorithm = "Hartigan-Wong",trace=FALSE)


cluster = cbind(Index$District, kmCIndex$cluster,kmCFull$cluster,
                kmCInPi$cluster,kmCPilar$cluster)
colnames(cluster) = c('dis','unweighted Index','Full Score',
                      'Index+ weighted Index','weigted Index')

#write.csv(cluster,'cluster.csv')

# plot the data points according to the first two principal components
# that explain the majority of the variance

colnames = c('dis',
             'unweighted Index',
             'Full Score',
             'Index+ weighted Index',
             'weigted Index')

pc1 = fviz_cluster(kmCFull, data = CFull,
             geom = c("point", "text")) +
```

```
              geom_text_repel(aes(label=IndexFull$District),size=3.5) +
              theme_bw()  +
              theme(panel.grid.minor = element_blank(),
              panel.grid.major = element_blank(),
              panel.border = element_blank())+
              ggtitle("cluster␣based␣on␣full␣score")

pc2 = fviz_cluster(kmCIndex, data = CIndex,
              geom = c("point", "text")) +
              geom_text_repel(aes(label=IndexFull$District),size=3.5) +
              theme_bw()  +
              theme(panel.grid.minor = element_blank(),
              panel.grid.major = element_blank(),
              panel.border = element_blank())+
              ggtitle("cluster␣based␣on␣unweighted␣5␣Index")

pc3 = fviz_cluster(kmCInPi, data = CInPi,
              geom = c("point", "text")) +
              geom_text_repel(aes(label=IndexFull$District),size=3.5) +
              theme_bw()  +
              theme(panel.grid.minor = element_blank(),
              panel.grid.major = element_blank(),
              panel.border = element_blank()) +
              ggtitle("cluster␣based␣on␣total␣Index")

pc4 = fviz_cluster(kmCPilar, data = CPilar,
              geom = c("point", "text")) +
              geom_text_repel(aes(label=IndexFull$District),size=3.5) +
              theme_bw()  +
              theme(panel.grid.minor = element_blank(),
              panel.grid.major = element_blank(),
              panel.border = element_blank()) +
              ggtitle("cluster␣based␣on␣Pilars(weighted␣Index)")

ggarrange(pc1,pc2,pc3,pc4)
pc1
```