Data Science with R – Bike Sharing Demand – Solution Sketch

TechAcademy e.V.

Sommersemester 2019

Inhalt

What's this document about?

Before we start, a few comments on this document.
This is one starting point for the Bike Sharing Project. I include all code for the plots from the project guideline ("Projektleitfaden"). Also, I included a very simple regression model for the Kaggle challange. This obviously predicts very poorly. The main focus is not on building the best possible model, but on getting the project up and running.
Refer to this document if the participants have technical (i.e. code-related) questions regarding the Exercises.
If you have questions regarding this project, please contact Lukas Jürgensmeier at *lukas.juergensmeier@tech-academy.io* or via TechAcademy's Slack channel at https://techacademyffm.slack.com.

Packages

```
library(GGally) # for corrplot()
library(ggplot2) # for visualization
library(lubridate) # deals with dates
library(dplyr) # for data wrangling
library(scales)
library(stargazer) # for nice LaTeX tables
library(car) #for vif()
library(ggmap) # for creating a map
library(magick) # for including a .png in plots
```

Explorative Datenanalyse – Lerne den Datensatz kennen

1. Plot des Ausleihverhaltens über die Zeit (train data set)

At this point, only importing the train data set is necessary. However, for the sake of completeness, I also import the test data set at this point.

```
train <- read.csv("BikeSharing_data.csv")
test <- read.csv("BikeSharing_data_test.csv")
head(train)
```

```
##              datetime season holiday workingday weather temp  atemp
## 1 2011-01-01 00:00:00      1       0          0       1 9.84 14.395
## 2 2011-01-01 01:00:00      1       0          0       1 9.02 13.635
## 3 2011-01-01 02:00:00      1       0          0       1 9.02 13.635
## 4 2011-01-01 03:00:00      1       0          0       1 9.84 14.395
## 5 2011-01-01 04:00:00      1       0          0       1 9.84 14.395
## 6 2011-01-01 05:00:00      1       0          0       2 9.84 12.880
##   humidity windspeed casual registered count
## 1       81    0.0000      3         13    16
## 2       80    0.0000      8         32    40
## 3       80    0.0000      5         27    32
## 4       75    0.0000      3         10    13
## 5       75    0.0000      0          1     1
## 6       75    6.0032      0          1     1
```

```
str(train)
```

```
## 'data.frame':    10886 obs. of  12 variables:
##  $ datetime  : Factor w/ 10886 levels "2011-01-01 00:00:00",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weather   : int  1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
##  $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
##  $ humidity  : int  81 80 80 75 75 75 80 86 75 76 ...
##  $ windspeed : num  0 0 0 0 0 ...
##  $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
##  $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
##  $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
```

```
summary(train)
```

```
##                  datetime         season          holiday       
##  2011-01-01 00:00:00:    1   Min.   :1.000   Min.   :0.00000  
##  2011-01-01 01:00:00:    1   1st Qu.:2.000   1st Qu.:0.00000  
##  2011-01-01 02:00:00:    1   Median :3.000   Median :0.00000  
##  2011-01-01 03:00:00:    1   Mean   :2.507   Mean   :0.02857  
##  2011-01-01 04:00:00:    1   3rd Qu.:4.000   3rd Qu.:0.00000  
##  2011-01-01 05:00:00:    1   Max.   :4.000   Max.   :1.00000  
##  (Other)            :10880                                    
##    workingday        weather           temp           atemp      
##  Min.   :0.0000   Min.   :1.000   Min.   : 0.82   Min.   : 0.76  
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:13.94   1st Qu.:16.66  
##  Median :1.0000   Median :1.000   Median :20.50   Median :24.24  
##  Mean   :0.6809   Mean   :1.418   Mean   :20.23   Mean   :23.66  
##  3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:26.24   3rd Qu.:31.06  
##  Max.   :1.0000   Max.   :4.000   Max.   :41.00   Max.   :45.45  
##                                                                  
##     humidity        windspeed          casual         registered   
##  Min.   :  0.00   Min.   : 0.000   Min.   :  0.00   Min.   :  0.0  
##  1st Qu.: 47.00   1st Qu.: 7.002   1st Qu.:  4.00   1st Qu.: 36.0
##  Median : 62.00   Median :12.998   Median : 17.00   Median :118.0
##  Mean   : 61.89   Mean   :12.799   Mean   : 36.02   Mean   :155.6
##  3rd Qu.: 77.00   3rd Qu.:16.998   3rd Qu.: 49.00   3rd Qu.:222.0
##  Max.   :100.00   Max.   :56.997   Max.   :367.00   Max.   :886.0
##
##      count
##  Min.   :  1.0
##  1st Qu.: 42.0
##  Median :145.0
##  Mean   :191.6
##  3rd Qu.:284.0
##  Max.   :977.0
##
```

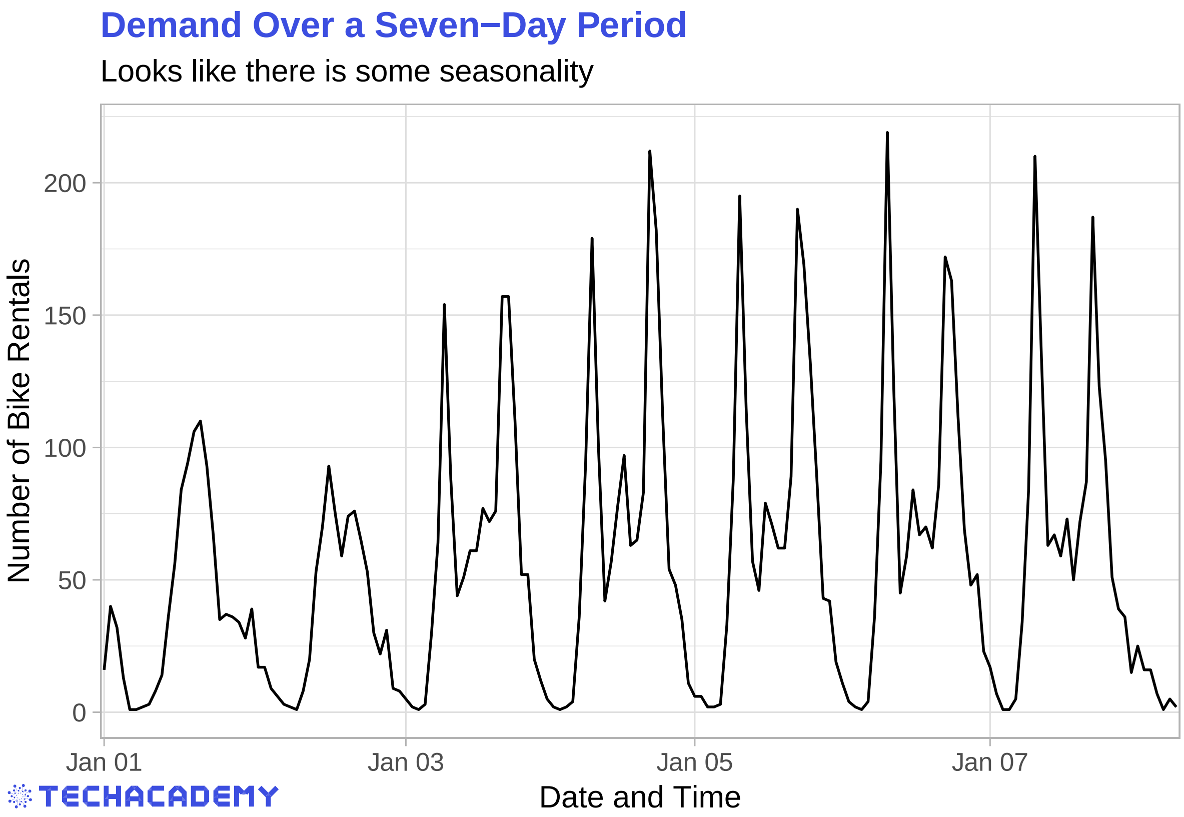
Transformations: We'll need to do some data wrangling here. Most of those transformations (except *hour*() and *weekday*()) are not necessary for plotting. At this point the participants only do those for the train data set and at a later stage repeat those steps for the train data set. This solution includes a creation of the variable *jitter_times*. This one is not required for the prediction, but for the plot by hour of the day. If we didn't include that jitter, there would only be data points on full hours.

```
#transformation for train dataset
train$hour  <- hour(ymd_hms(train$datetime))
train$times <- as.POSIXct(strftime(ymd_hms(train$datetime), format="%H:%M:%S"), format="%H:%M
train$jitter_times <- train$times+minutes(round(runif(nrow(train),min=0,max=59)))
train$day <- wday(ymd_hms(train$datetime), label=TRUE)

#transformation for test dataset
test$hour  <- hour(ymd_hms(test$datetime))
test$times <- as.POSIXct(strftime(ymd_hms(test$datetime), format="%H:%M:%S"), format="%H:%M:%
test$jitter_times <- test$times+minutes(round(runif(nrow(test),min=0,max=59)))
test$day <- wday(ymd_hms(test$datetime), label=TRUE)

#for train dataset
train$season <- as.factor(train$season)
train$holiday <- as.factor(train$holiday)
train$workingday <- as.factor(train$workingday)
train$weather <- as.factor(train$weather)
train$day <- factor(train$day, ordered = FALSE)

#for test data set
test$season <- as.factor(test$season)
```

```
test$holiday <- as.factor(test$holiday)
test$workingday <- as.factor(test$workingday)
test$weather <- as.factor(test$weather)
test$day <- factor(test$day, ordered = FALSE)
```

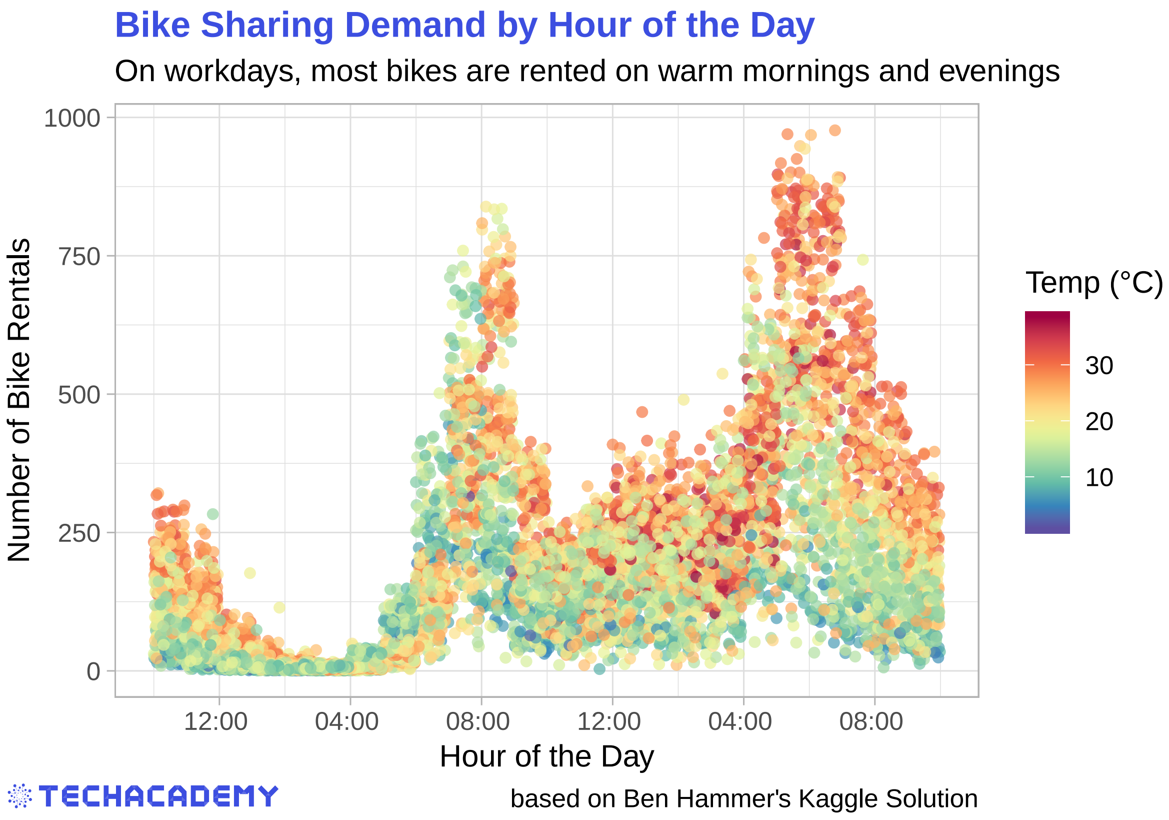
Create the demand vs date plot during the first 7 days.

```
ggplot(train[1:168,], aes(datetime, count)) +
  geom_line(group = 24) +
  scale_x_discrete(breaks=c("2011-01-01 00:00:00","2011-01-03 00:00:00", "2011-01-05 00:00:00
  theme_light(base_size=12) +
  labs(title = "Demand Over a Seven-Day Period",
       subtitle = "Looks like there is some seasonality",
       x = "Date and Time",
       y = "Number of Bike Rentals") +
  theme(plot.title = element_text(color = "#3c4ee0", face = 'bold'))

# assigning the TechAcademy Logo to an object to include it in the plots
TA_logo <- image_read("TA_logo.png")
# add TechAcademy Logo
grid::grid.raster(TA_logo, x = 0.01, y = 0.02, just = c('left', 'bottom'), width = unit(1.5,
```



The following one is a more advanced plot, which is not required. It should only serve as an inspiration for the participants and show what *ggplot*2 is capable of. Data set is filtered to working days only, since this particular pattern is only visible on working days. On weekends, the distribution looks completely different.

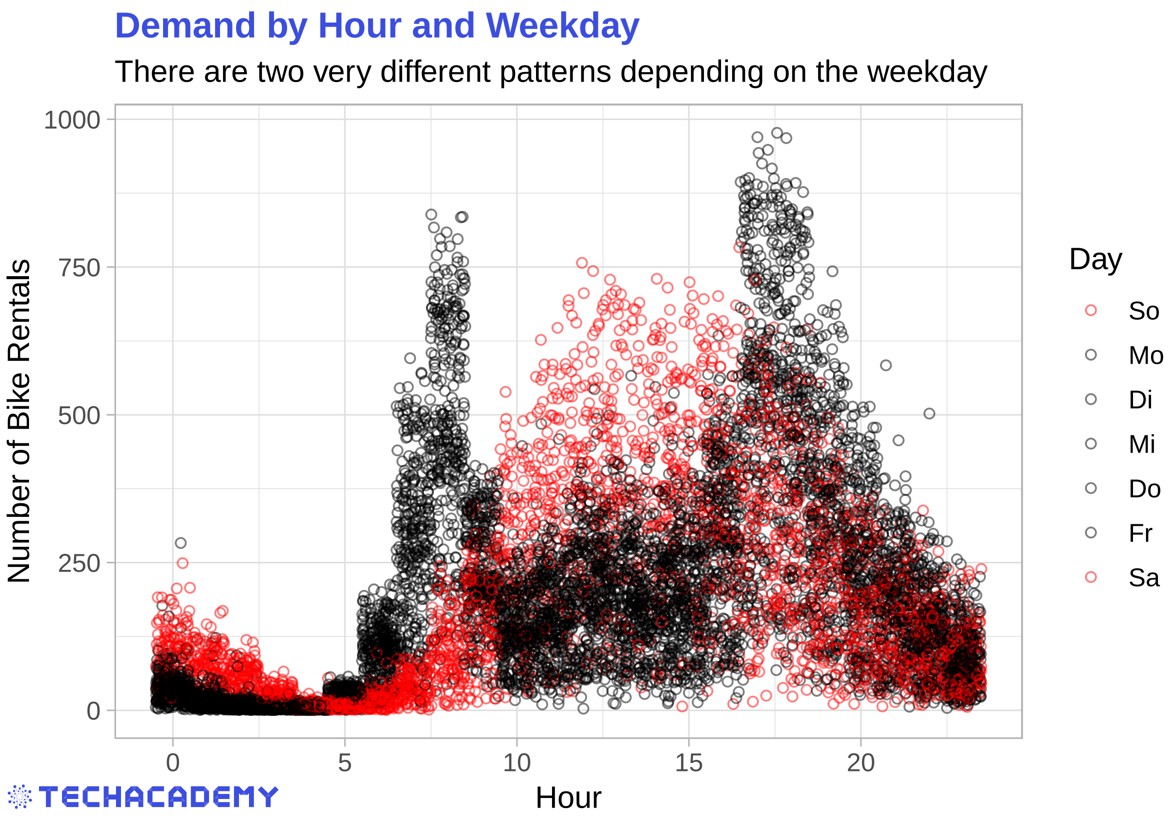
```
ggplot(train[train$workingday==1,], aes_string("jitter_times", "count", color="temp")) +
     geom_point(position=position_jitter(w=0.0, h=0.4), alpha = 0.7) +
     theme_light(base_size=12) +
     scale_x_datetime(breaks = date_breaks("4 hours"), labels=date_format("%I:%M %p")) +
     labs(title = 'Bike Sharing Demand by Hour of the Day',
          subtitle = "On workdays, most bikes are rented on warm mornings and evenings",
          caption = "based on Ben Hammer's Kaggle Solution",
          x = "Hour of the Day",
          y = "Number of Bike Rentals") +
     scale_colour_gradientn("Temp (°C)", colours=c("#5e4fa2", "#3288bd", "#66c2a5", "#abdda4"
      theme(plot.title = element_text(color = "#3c4ee0", face = 'bold'))

# add TechAcademy Logo
grid::grid.raster(TA_logo, x = 0.01, y = 0.02, just = c('left', 'bottom'), width = unit(1.5,
```



2. Ausleihverhalten nach Wochentag (train data set)

This exercise should make clear that there is a very different pattern in *count* depending on day of the week. This can be achieved by grouping the different week days by color in a scatterplot.

```
ggplot(train, aes(hour, count)) +
  geom_jitter(aes(color = day), shape = 1, width = 0.5, alpha = 0.5) +
  theme_light(base_size=12) +
  labs(title = "Demand by Hour and Weekday",
       subtitle = "There are two very different patterns depending on the weekday",
       x = "Hour",
       y = "Number of Bike Rentals")+
  theme(plot.title = element_text(color = "#3c4ee0", face = 'bold')) +
  scale_colour_manual(name="Day", values= c("red", "black", "black", "black", "black", "black

# add TechAcademy Logo
grid::grid.raster(TA_logo, x = 0.01, y = 0.02, just = c('left', 'bottom'), width = unit(1.5,
```



3. Wie lange werden die Räder ausgeliehen? (data_month)

Important: We're switching data sets here.
First, import and glimpse into the monthly data set:

```
monthly <- read.csv("BikeSharing_data_201902.csv")
head(monthly)
```

```
##   Duration          Start.date            End.date Start.station.number
## 1      206 2019-02-01 00:00:20 2019-02-01 00:03:47                31509
## 2      297 2019-02-01 00:04:40 2019-02-01 00:09:38                31203
## 3      165 2019-02-01 00:06:34 2019-02-01 00:09:20                31303
## 4      176 2019-02-01 00:06:49 2019-02-01 00:09:45                31400
## 5      105 2019-02-01 00:10:41 2019-02-01 00:12:27                31270
## 6      757 2019-02-01 00:12:37 2019-02-01 00:25:14                31503
##                                  Start.station End.station.number
## 1                     New Jersey Ave & R St NW              31636
## 2                   14th & Rhode Island Ave NW              31519
## 3 Tenleytown / Wisconsin Ave & Albemarle St NW              31308
## 4               Georgia & New Hampshire Ave NW              31401
## 5                                8th & D St NW              31256
## 6                        Florida Ave & R St NW              31126
##                          End.station Bike.number Member.type
## 1 New Jersey Ave & N St NW/Dunbar HS      W21713      Member
## 2                      1st & O St NW      E00013      Member
## 3                39th & Veazey St NW      W21703      Member
## 4             14th St & Spring Rd NW      W21699      Member
## 5                     10th & E St NW      W21710      Member
## 6                11th & Girard St NW      W22157      Member
```

```
str(monthly)
```

```
## 'data.frame':    158130 obs. of  9 variables:
##  $ Duration            : int  206 297 165 176 105 757 844 313 152 1935 ...
##  $ Start.date          : Factor w/ 147706 levels "2019-02-01 00:00:20",..: 1 2 3 4 5 6 7 8
##  $ End.date            : Factor w/ 147561 levels "2019-02-01 00:03:47",..: 1 3 2 4 5 8 9 6
##  $ Start.station.number: int  31509 31203 31303 31400 31270 31503 31324 31241 31020 31104
##  $ Start.station       : Factor w/ 523 levels "10th & E St NW",..: 390 38 473 284 162 274
##  $ End.station.number  : int  31636 31519 31308 31401 31256 31126 31107 31245 31030 31108
##  $ End.station         : Factor w/ 523 levels "10th & E St NW",..: 388 91 127 44 1 11 325
##  $ Bike.number         : Factor w/ 4366 levels "51020","51033",..: 2389 22 2380 2376 2386
##  $ Member.type         : Factor w/ 2 levels "Casual","Member": 2 2 2 2 2 2 2 2 2 2 ...
```

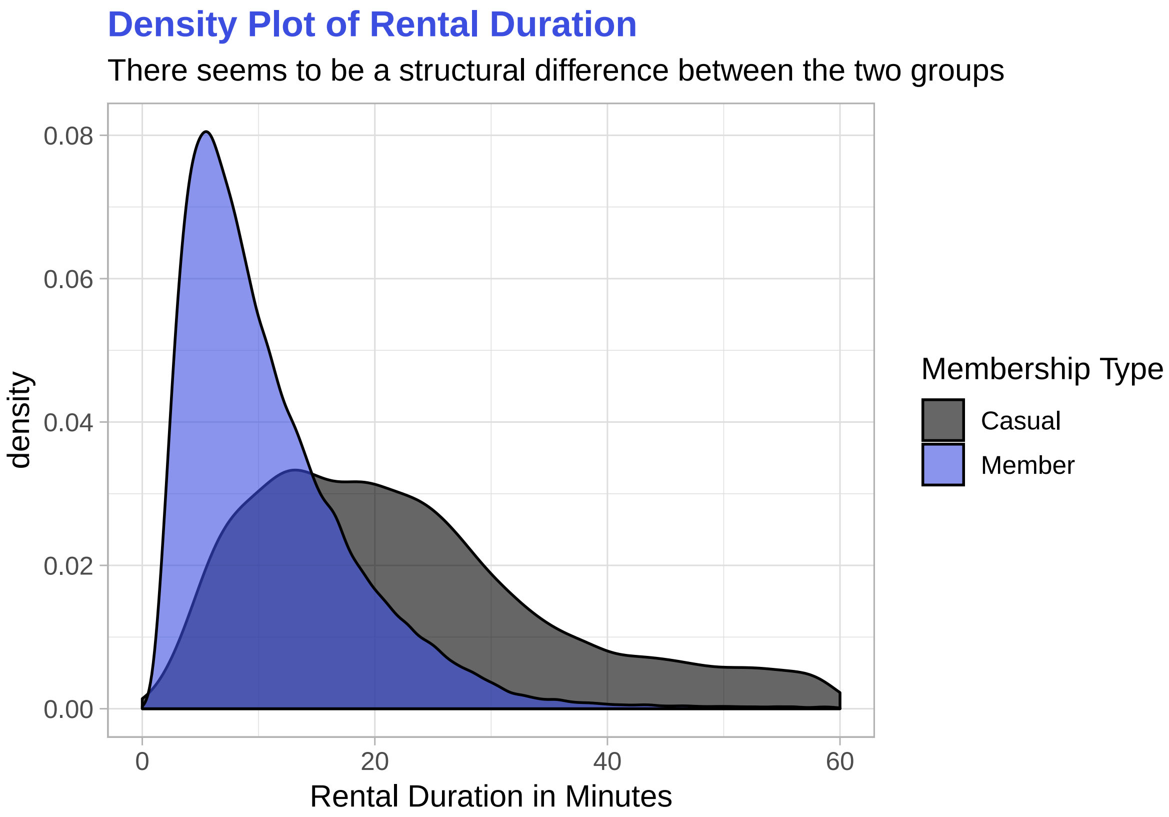
Duration is in seconds. For better readability (esp. in plots), transform it to minutes.

```
monthly$Duration <- monthly$Duration/60
```

Now, participants should create something like a Density Plot of Rental Duration, distinguished by *Membership.type*. It's important to specify the limits of the x-axis or deal with the outliers in some other way. Else, it is very hard to interpret the density plot.

```
ggplot(monthly, aes(Duration)) +
  geom_density(aes(y=..density.., fill = Member.type), alpha = 0.6) +
  xlim(0, 60) +
  theme_light(base_size=12) +
  labs(title = 'Density Plot of Rental Duration',
       x = "Rental Duration in Minutes",
       subtitle = "There seems to be a structural difference between the two groups") +
  theme(plot.title = element_text(color = "#3c4ee0", face = 'bold')) +
  scale_fill_manual(name= "Membership Type", values = c("black", "#3c4ee0"))
```

```
## Warning: Removed 3255 rows containing non-finite values (stat_density).
```



4. Mergen & Karten zeichnen (data_month)

This part consists of two tasks. First, we need to merge two data sets to find out the coordinates for the start and end stations. The second task introduces the Google Maps API and shows how we can visualize coordinates on a map. The second one is not mandatory, since you need to enter credit card information to gain access to the API.

4.1 Datensätze mergen

We now merge two data sets in order to get the coordinates of the start and end station into the *monthly* data set. This will probably cause a lot of headaches among the students, but the solution is quite simple (in hindsight).

First, import the *stations* data set. This serves as a lookup table.

```
stations <- read.csv("BikeSharing_data_stations.csv")
head(stations)
```

```
##   X Station.number                                 Station.name       lon
## 1 0          31303 Tenleytown / Wisconsin Ave & Albemarle St NW -77.07934
## 2 1          31400              Georgia & New Hampshire Ave NW -77.02465
## 3 2          31270                                8th & D St NW -77.02324
## 4 3          31503                           Florida Ave & R St NW -77.01344
## 5 4          31324               18th & New Hampshire Ave NW -77.04183
## 6 5          31241                                Thomas Circle -77.03250
##        lat
## 1 38.94757
## 2 38.93604
## 3 38.89486
## 4 38.91254
## 5 38.91128
## 6 38.90590
```

```
summary(stations)
```

```
##        X          Station.number                   Station.name
##  Min.   :  0.0   Min.   :31000   Kennedy Center        :  2
##  1st Qu.:102.8   1st Qu.:31224   10th & E St NW        :  1
##  Median :205.5   Median :31513   10th & Florida Ave NW:  1
##  Mean   :205.5   Mean   :31552   10th & G St NW        :  1
##  3rd Qu.:308.2   3rd Qu.:31919   10th & Monroe St NE   :  1
##  Max.   :411.0   Max.   :32407   10th & U St NW        :  1
##                                  (Other)               :405
##       lon              lat
##  Min.   :-77.37   Min.   :38.78
##  1st Qu.:-77.08   1st Qu.:38.89
##  Median :-77.04   Median :38.90
##  Mean   :-77.06   Mean   :38.92
##  3rd Qu.:-77.01   3rd Qu.:38.94
##  Max.   :-76.83   Max.   :39.12
##
```

```
# First, merge the the two data sets by Start.station.number
monthly <- merge(monthly, stations, by.x ="Start.station.number", by.y = "Station.number")

#then rename the variables including coordinates to reflect the starting station
monthly <- monthly %>%
  rename(start.station.name = Station.name,
         start.lon = lon,
         start.lat = lat)

# then do the same for the end stations
```

```
monthly <- merge(monthly, stations, by.x ="End.station.number", by.y = "Station.number")
monthly <- monthly %>%
  rename(End.station.name = Station.name,
         End.lon = lon,
         End.lat = lat)
```

Now check if everything went well. We need to have four more variables in the data set, each two coordinates for the start as well as the end station. Note that the first few entries belong to the same station. It is therefore okay, if the first few coordinates are the same.

```
head(monthly)
```

```
##   End.station.number Start.station.number  Duration          Start.date
## 1              31000                31071  1.733333 2019-02-05 13:06:50
## 2              31000                31230 38.233333 2019-02-18 12:47:01
## 3              31000                31064 25.166667 2019-02-15 18:01:05
## 4              31000                31321 23.850000 2019-02-06 16:02:54
## 5              31000                31218 26.800000 2019-02-17 18:31:14
## 6              31000                31000 16.416667 2019-02-03 13:17:30
##              End.date                  Start.station         End.station
## 1 2019-02-05 13:08:35            Eads St & 12th St S Eads St & 15th St S
## 2 2019-02-18 13:25:16  Metro Center / 12th & G St NW Eads St & 15th St S
## 3 2019-02-15 18:26:15                 Gravelly Point Eads St & 15th St S
## 4 2019-02-06 16:26:46  15th St & Constitution Ave NW Eads St & 15th St S
## 5 2019-02-17 18:58:03 L'Enfant Plaza / 7th & C St SW Eads St & 15th St S
## 6 2019-02-03 13:33:55            Eads St & 15th St S Eads St & 15th St S
##   Bike.number Member.type X.x             start.station.name start.lon
## 1      W22530      Member 275            Eads St & 12th St S -77.05428
## 2      W23519      Member 129  Metro Center / 12th & G St NW -77.02787
## 3      W00697      Member 123                 Gravelly Point -77.03951
## 4      W21917      Member 119  15th St & Constitution Ave NW -77.03324
## 5      W00765      Member  79 L'Enfant Plaza / 7th & C St SW -77.02224
## 6      W22916      Member 297            Eads St & 15th St S -77.05323
##   start.lat X.y    End.station.name   End.lon  End.lat
## 1  38.86276 297 Eads St & 15th St S -77.05323 38.85898
## 2  38.89836 297 Eads St & 15th St S -77.05323 38.85898
## 3  38.86504 297 Eads St & 15th St S -77.05323 38.85898
## 4  38.89225 297 Eads St & 15th St S -77.05323 38.85898
## 5  38.88627 297 Eads St & 15th St S -77.05323 38.85898
## 6  38.85898 297 Eads St & 15th St S -77.05323 38.85898
```

```
str(monthly)
```

```
## 'data.frame':    100558 obs. of  17 variables:
```

```
##  $ End.station.number  : int  31000 31000 31000 31000 31000 31000 31000 31000 31000 31000
##  $ Start.station.number: int  31071 31230 31064 31321 31218 31000 31071 31071 31071 31071
##  $ Duration            : num  1.73 38.23 25.17 23.85 26.8 ...
##  $ Start.date          : Factor w/ 147706 levels "2019-02-01 00:00:20",..: 20881 92048 793
##  $ End.date            : Factor w/ 147561 levels "2019-02-01 00:03:47",..: 20741 92008 793
##  $ Start.station       : Factor w/ 523 levels "10th & E St NW",..: 253 358 297 54 323 254
##  $ End.station         : Factor w/ 523 levels "10th & E St NW",..: 254 254 254 254 254 254
##  $ Bike.number         : Factor w/ 4366 levels "51020","51033",..: 2990 3756 535 2531 580
##  $ Member.type         : Factor w/ 2 levels "Casual","Member": 2 2 2 2 2 2 2 2 2 2 ...
##  $ X.x                 : int  275 129 123 119 79 297 275 275 275 275 ...
##  $ start.station.name  : Factor w/ 411 levels "10th & E St NW",..: 202 286 239 45 260 203
##  $ start.lon           : num  -77.1 -77 -77 -77 -77 ...
##  $ start.lat           : num  38.9 38.9 38.9 38.9 38.9 ...
##  $ X.y                 : int  297 297 297 297 297 297 297 297 297 297 ...
##  $ End.station.name    : Factor w/ 411 levels "10th & E St NW",..: 203 203 203 203 203 203
##  $ End.lon             : num  -77.1 -77.1 -77.1 -77.1 -77.1 ...
##  $ End.lat             : num  38.9 38.9 38.9 38.9 38.9 ...
```

4.2 Koordinaten-Einträge mit Google Maps visualisieren (Bonus-Aufgabe)

Configure your own Google Maps API first. You'll find more information how to configure the API at https://lucidmanager.org/geocoding-with-ggmap/

```
# The API Key is confidential, since it is linked to a credit card
# To run this on your machine, create your own account with Google
# and replace this fake key with your real one.
# You'll then be able to create your own map.
library(mapsapi) # for configuring the Google Maps API
register_google(key = "INSERT YOUR KEY HERE")
```

Then download the map from Google Maps with this function. I set the location based on the mean values of our data set's coordinates. You can also just type in the city as a string (i.e. "Washington, DC"). The following chunks for the maps are causing problems with RMarkdown, so I set eval = FALSE.

```
DC_map <- get_map(location = c(lon = mean(monthly$start.lon),
                               lat = mean(monthly$start.lat)),
                  zoom = 13,
                  maptype = "roadmap", scale = 2)
```

This draws all start locations on the map of Washington, DC.

```
ggmap(DC_map) +
  geom_point(data = monthly,
             aes(x = start.lon, y = start.lat, alpha = 0.8),
             color = "#3c4ee0",
             size = 1,
             shape = 3) +
  theme_light(base_size=12) +
  labs(title = 'Location of Rental Bike Stations',
       x = "Longitude",
       y = "Latitude",
       subtitle = "Stations seem to be clustered in the city center") +
  theme(plot.title = element_text(color = "#3c4ee0", face = 'bold'),
        legend.position = "none")
```

You can also play around with densities. This map shows the density of stations additionally to the single stations.

```
ggmap(DC_map) +
  guides(fill=FALSE, alpha=FALSE, size=FALSE) +
  stat_density2d(data = monthly,
                 aes(x = start.lon, y = start.lat, fill = ..level.., alpha = ..level..),
                 size = 0.01,
                 bins = 16,
                 geom = "polygon") +
  scale_fill_gradient(low = "grey", high = "red") +
  geom_point(data = monthly,
             aes(x = start.lon, y = start.lat, alpha = 0.8),
             color = "blue", size = 1, shape = 3) +
  theme_light(base_size=12) +
  labs(title = 'Density of Rental Bike Stations',
       x = "Longitude",
       y = "Latitude",
       subtitle = "Stations seem to be clustered in the city center") +
  theme(plot.title = element_text(color = "#3c4ee0", face = 'bold'),
        legend.position = "none")
```

A very nice additional visualisation would be to show the densities of *count* by station.

Nachfrage-Prognose – Wende statistische Methoden an

1. Untersuche den Zusammenhang zwischen den Variablen näher

First, check correlations

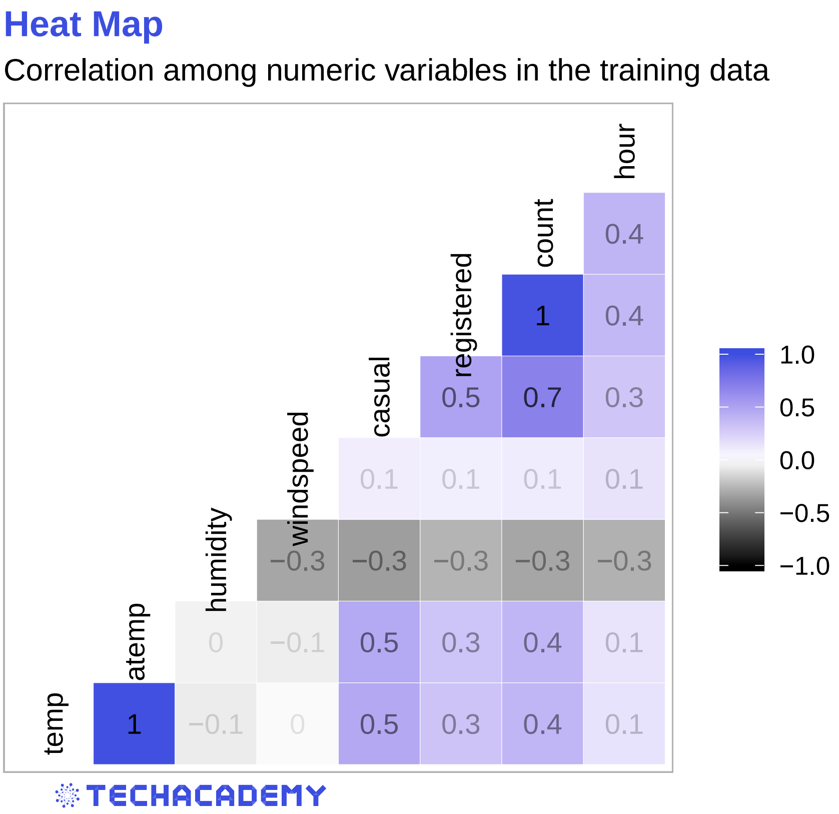
```
# in a table
round(cor(train[sapply(train, is.numeric)]),2)
```

```
##               temp atemp humidity windspeed casual registered count  hour
## temp         1.00  0.98     -0.06     -0.02   0.47       0.32  0.39  0.15
## atemp        0.98  1.00     -0.04     -0.06   0.46       0.31  0.39  0.14
## humidity   -0.06 -0.04      1.00     -0.32  -0.35      -0.27 -0.32 -0.28
## windspeed  -0.02 -0.06     -0.32      1.00   0.09       0.09  0.10  0.15
## casual       0.47  0.46     -0.35      0.09   1.00       0.50  0.69  0.30
## registered  0.32  0.31     -0.27      0.09   0.50       1.00  0.97  0.38
## count        0.39  0.39     -0.32      0.10   0.69       0.97  1.00  0.40
## hour         0.15  0.14     -0.28      0.15   0.30       0.38  0.40  1.00
```

```r
# in a heat map
ggcorr(train, low = "black", mid = "white", high = "#3c4ee0",
       label = TRUE, label_alpha = TRUE, angle = 90) +
  theme_light(base_size=12) +
     labs(title = 'Heat Map',
          subtitle = "Correlation among numeric variables in the training data ",
          caption = " ") +
      theme(plot.title = element_text(color = "#3c4ee0", face = 'bold'))
```

```
## Warning in ggcorr(train, low = "black", mid = "white", high = "#3c4ee0", :
## data in column(s) 'datetime', 'season', 'holiday', 'workingday', 'weather',
## 'times', 'jitter_times', 'day' are not numeric and were ignored
```

```r
# Add TechAcademy Logo
grid::grid.raster(TA_logo, x = 0.19, y = 0.02,
                  just = c('left', 'bottom'), width = unit(1.5, 'inches'))
```



2. Teste verschiedene Modelle und deren Qualität

Then, set up simple models. Note: this is just a starting point. Those models are obviously very much improvable.

```
model1 <- lm(count ~ temp , data = train)
model2 <- lm(count ~ temp + hour, data = train)
```

Very important here is that *hour* is a numeric variable. It doesn't make too much sense to include it like that. Transform it to a factor for a better model.

```
stargazer(model1, model2,
          type = "latex",
          column.labels = c(),
          title = "Model Comparison",
          style = "default",
          font.size = "small",
          header = FALSE)
```

4. Lade den Datensatz auf Kaggle hoch

Create new prediction data set

```
# extract predicted values from model and fit them to test df
predictions_model2 <- predict(model2, test)
```

Table 1: Model Comparison

	Dependent variable:	
	count	
	(1)	(2)
temp	9.171***	7.985***
	(0.205)	(0.192)
hour		9.185***
		(0.216)
Constant	6.046	$-$75.973***
	(4.439)	(4.541)
Observations	10,886	10,886
R^2	0.156	0.276
Adjusted R^2	0.156	0.276
Residual Std. Error	166.464 (df = 10884)	154.152 (df = 10883)
F Statistic	2,005.529*** (df = 1; 10884)	2,073.866*** (df = 2; 10883)
Note:		*p<0.1; **p<0.05; ***p<0.01

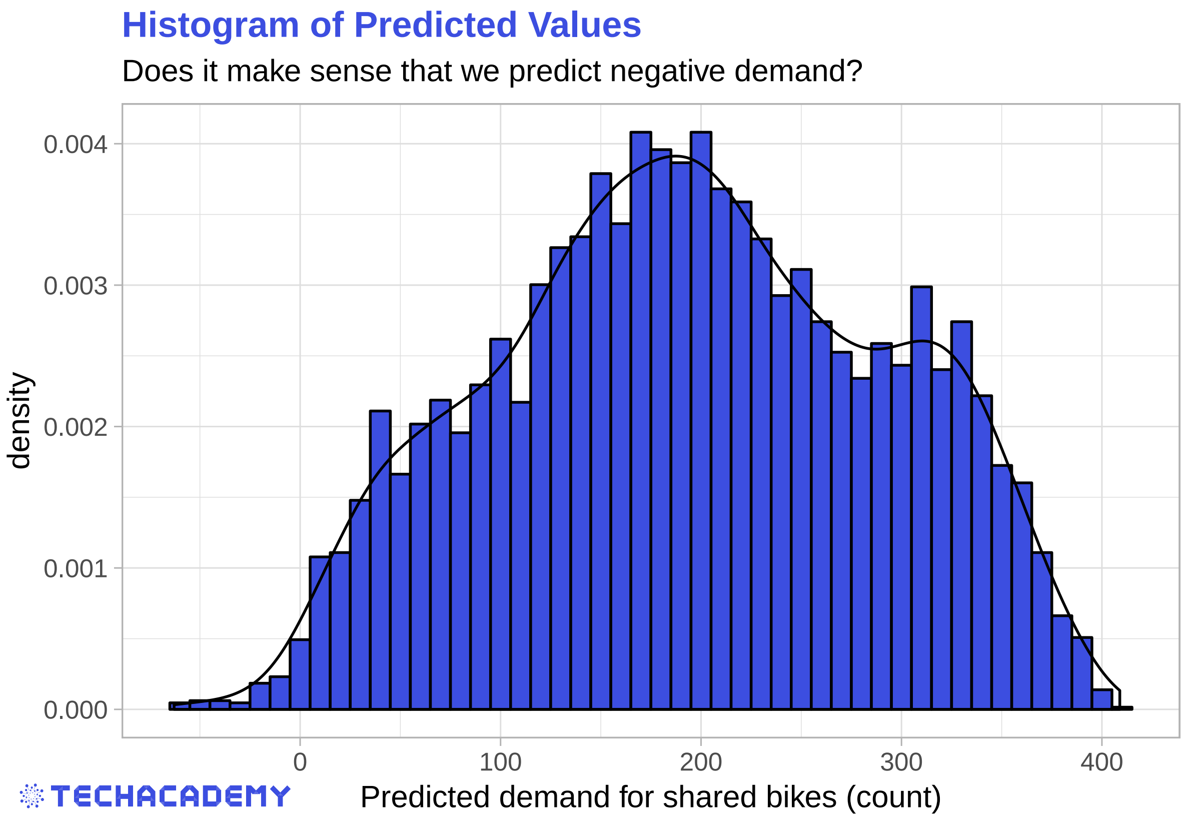
```r
# create new df with only datetime and count
submit_model2 <- data.frame(datetime = test$datetime, count = predictions_model2)
```

Then, check the submission data set

```r
head(submit_model2)
```

```
##              datetime     count
## 1 2011-01-20 00:00:00  9.146889
## 2 2011-01-20 01:00:00 18.331415
## 3 2011-01-20 02:00:00 27.515941
## 4 2011-01-20 03:00:00 36.700467
## 5 2011-01-20 04:00:00 45.884992
## 6 2011-01-20 05:00:00 48.521845
```

```r
ggplot(submit_model2, aes(count)) +
  geom_histogram(aes(y=..density..), binwidth = 10, color = "black", fill = "#3c4ee0") +
  geom_density(aes(y=..density..)) +
  theme_light(base_size=12) +
     labs(title = 'Histogram of Predicted Values',
          x = "Predicted demand for shared bikes (count)",
          subtitle = "Does it make sense that we predict negative demand?") +
      theme(plot.title = element_text(color = "#3c4ee0", face = 'bold'))

# Add TechAcademy Logo
grid::grid.raster(TA_logo, x = 0.02, y = 0.02, just = c('left', 'bottom'),
                  width = unit(1.5, 'inches'))
```



Refine the predicions based on that evaluation. A very simple fix is to eliminate the negative predictions. However, it might be better to choose a different model

```
# negative count values don't make sense; replace them with 0
submit_model2$count[submit_model2$count<0] <- 0
```

As a last step, write the results to a .csv file for submission

```
write.csv(submit_model2, file="submit_model2.csv", row.names=FALSE)
```

Then, upload this file to Kaggle.