Data Wizard Take Home Exercise - Assignment Part 1

Author: Giacomo Dalla Chiara

The objective of this exercise is to estimate the average hourly pay and visualize it using a heatmap, taking into account errors, outliers, edge cases, lack of data, extreme events, and robustness of the estimated averages. I first generate a synthetic dataset of drivers' trips to work on. I then propose four different models, ordered by level of complexity, to approach the estimation probem:

- V0: non-parametrics model using the mean function,
- V1: non-parametric model using the median function,
- V2: parametrics model using linear regression,
- V3: parametric model using mixed effect regression.

In [147…
```
rm(list=ls()) #clean the console
library(ggplot2) #package used for plotting
library(repr) #package used to control plot size in notebook
#install.packages("lme4", repos='http://cran.us.r-project.org')
#library(lme4) #package for estimating mixed-effect models
```

Synthetic data generation

In this first step, I generate a synthetic dataset of 1000 trips performed by 50 drivers. Each observation is a trip, characterized by the following variables:

- driver_id: assuming we have 50 unique driver IDs
- time_of_day: assuming we have data from midnight (0) to 11 pm (23)
- pick_up_date: we assume data is from a week in February 2023
- earnings_pay: assuming earnings are exponentially distributed with rate=1 (parameter of the exp distirbution)
- day_of_week: Monday, Tuesday, ...
- special_event: =1 whenever a trip is considered belonging to a "special event", e.g. a sport event

By generating synthetic data we can test how different models behave under certain conditions (e.g. outliers, extreme values). Synthetic data can also be used for simulation (e.g. we could simulate a entire population of drivers, or simulate drivers in a location for which we have little data).

In [148…
```
# Fix a seed number for random data generation to ensure replication
set.seed(61)

# Fix the total size of the sample to generate
n <- 1000

# Generate synthetic job dataset
job <- data.frame("driver_id"=sample(1:50, n, replace=TRUE),
                  "time_of_day"=sample(0:23, n, replace=TRUE),
                  "pick_up_date"=sample(seq(as.Date("2023/02/13"), as.Date("2023/02/19"), by="day")
                  "earnings_pay"=rexp(n, rate = 0.045))

# Adding day of the week
job$day_of_week <- weekdays(job$pick_up_date)

# Adding special events (e.g. super bowl)
job$special_event <- 0
job[c(1,50,467,835,456,23,788),"special_event"] <- 1
job[c(1,50,467,835,456,23,788),"earnings_pay"] <- job[c(1,50,467,835,456,23,788),"earnings_pay"]+40
job$special_event <- as.factor(job$special_event)

str(job)
```

```
'data.frame':   1000 obs. of  6 variables:
 $ driver_id    : int  12 7 23 42 42 41 20 34 11 33 ...
 $ time_of_day  : int  9 8 14 4 6 17 0 17 20 1 ...
 $ pick_up_date : Date, format: "2023-02-17" "2023-02-14" ...
 $ earnings_pay : num  426.48 14.69 3.45 2.25 38.04 ...
 $ day_of_week  : chr  "Friday" "Tuesday" "Monday" "Sunday" ...
 $ special_event: Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...
```

The data generated is per trip. I then compute the total hourly earnings per driver, per time of day (of the pick-up) and day of the week. NOTE: I'm assuming here that trips are cotnained within the pick-up hour.

In [149…
```
# Compute the hourly pay, from the trips data
hour_pay <- aggregate(job$earnings_pay, list(job$time_of_day, job$day_of_week, job$driver_id), FUN=
names(hour_pay) <- c("time_of_day", "day_of_week", "driver_id", "hourly_earnings")

str(hour_pay)
```

```
'data.frame':   943 obs. of  4 variables:
 $ time_of_day    : int  17 5 9 8 0 17 8 11 1 2 ...
 $ day_of_week    : chr  "Friday" "Monday" "Monday" "Saturday" ...
 $ driver_id      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ hourly_earnings: num  24.1 47.6 29.8 23.6 12.6 ...
```

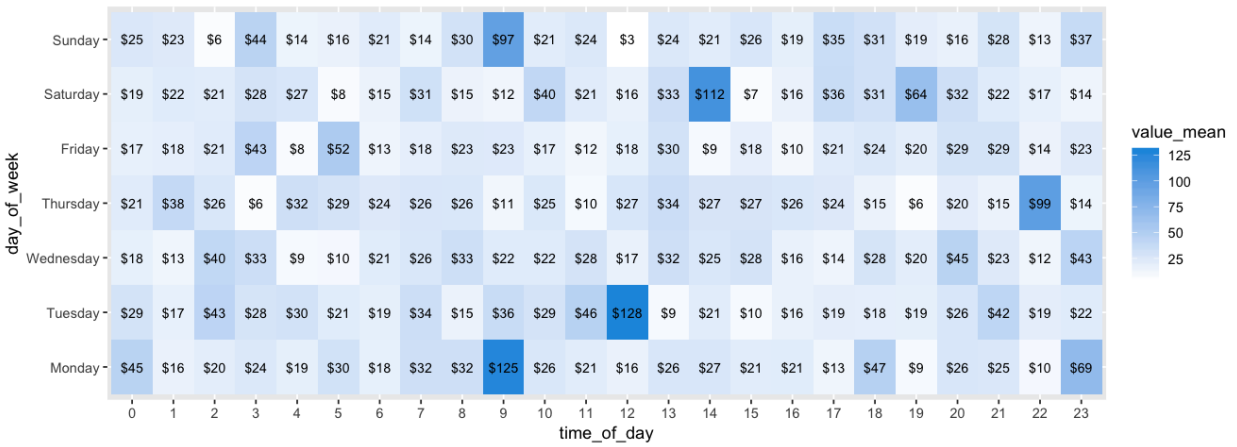
V0. Non-parametric model: Mean

In the first model V0, I estimate average hourly pay using the mean function

In [150…
```
# Use the aggregate() function to compute the mean pay for each combination of day and time
avg_pay <- aggregate(hour_pay$hourly_earnings, list(hour_pay$time_of_day, hour_pay$day_of_week), FU
names(avg_pay) <- c("time_of_day", "day_of_week", "value_mean")
```

Then, I use ggplot to create a heatmap of the mean hourly pay

In [151…
```
avg_pay$lab_mean <- paste0("$",as.character(floor(avg_pay$value_mean)))
avg_pay$time_of_day <- as.factor(avg_pay$time_of_day)
avg_pay$day_of_week <- as.factor(avg_pay$day_of_week)
levels(avg_pay$day_of_week) <- c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sun
hm_mean <- ggplot(avg_pay, aes(time_of_day, day_of_week)) +
      geom_tile(aes(fill = value_mean)) +
      geom_text(aes(label = lab_mean), size=3) +
      scale_fill_gradient(low = "white", high = "#1b98e0")
options(repr.plot.width=11, repr.plot.height=4)
hm_mean
```

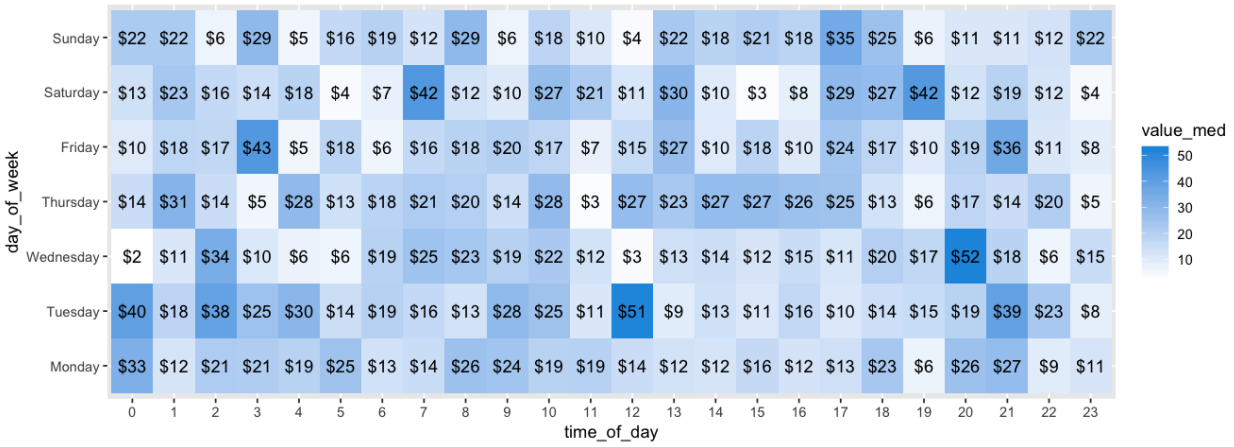


Note that the mean function is not robust to extreme values and outliers. Cells representing average hourly pay for Monday, 9 am, Tuesday, 12 pm, and Saturday, 2 pm, show skewed averages due to special event observations.

V1. Non-parametric model: Median

In the second model V1, I use the median funciton to estimate average hourly pay. The median function is more rovust to extreme values and outliers than the mean function.

In [152…
```
# Add column to the avg_pay dataset reporting the median hourly pay
avg_pay$value_med <- aggregate(hour_pay$hourly_earnings, list(hour_pay$time_of_day, hour_pay$day_of

# Same as above, I use ggplot to plot a heatmap of avrage hourly earnings
avg_pay$lab_med <- paste0("$",as.character(floor(avg_pay$value_med)))
avg_pay$time_of_day <- as.factor(avg_pay$time_of_day)
hm_med <- ggplot(avg_pay, aes(time_of_day, day_of_week)) +
  geom_tile(aes(fill = value_med)) +
  geom_text(aes(label = lab_med)) +
  scale_fill_gradient(low = "white", high = "#1b98e0")
options(repr.plot.width=11, repr.plot.height=4)
hm_med
```



Note that, in comparison with the heatmap generated by V0, the large average hourly pays for Monday 9 am and Saturday 2 pm disappear. Now the Tuesday 12 pm and Saturday 7 am show the highest averages.

Lack of data and confidence of the estimates

While the median function take care of extreme values and outliers, lack of observations for certain combination of days and times might still be a problem. In the plots above I only reported the point estimates for the average hourly pay, but I do not show how confident I am that these point estimates are as close as possible to the "true" average hourly pay. To tackle this problem, I would us: 1) a rule base method, i.e. I would identify how often and when/where we have less than, for instance, 5 observations 2) compute 95% confidence interval for the estimate

The narrower the confidence interval around the point estimate, the more precise the estimate is. Lack of data would then be reflected by a wider confidence interval. Confidence intervals could be added somehow to the visualization to aid decision making. Or it could be used in the background, for instance to identify point estimates where the CI is too wide.

In the following code I compute first the sample size that contributes to each point estimates, then the confidence interval for just one point estimate as an example.

In [153…

```
# Add column to avg_pay containing sample size
avg_pay$sample_size <- aggregate(hour_pay$hourly_earnings, list(hour_pay$time_of_day, hour_pay$day_

# Display the point estimates with small sample size using a rule-based method
avg_pay[avg_pay$sample_size<5,]

# Compute confidence interval for the median hourly pay for Tuesday 11 am
wilcox.test(hour_pay[hour_pay$time_of_day==12 & hour_pay$day_of_week=="Tuesday","hourly_earnings"],
```

	time_of_day	day_of_week	value_mean	lab_mean	value_med	lab_med	sample_size
3	2	Monday	20.209514	$20	21.661586	$21	4
5	4	Monday	19.249510	$19	19.249510	$19	2
18	17	Monday	13.024267	$13	13.961362	$13	4
21	20	Monday	26.851343	$26	26.051626	$26	3
22	21	Monday	25.687073	$25	27.851747	$27	4
25	0	Tuesday	29.606296	$29	40.230553	$40	3
28	3	Tuesday	28.548463	$28	25.661461	$25	3
29	4	Tuesday	30.203213	$30	30.203213	$30	2
33	8	Tuesday	15.772899	$15	13.048394	$13	3
35	10	Tuesday	29.148743	$29	25.268841	$25	3
37	12	Tuesday	128.671484	$128	51.755472	$51	4
38	13	Tuesday	9.636809	$9	9.636809	$9	2
39	14	Tuesday	21.410615	$21	13.686273	$13	4
41	16	Tuesday	16.887585	$16	16.887585	$16	2
43	18	Tuesday	18.985550	$18	14.188585	$14	3
46	21	Tuesday	42.537447	$42	39.127481	$39	3
52	3	Wednesday	33.855431	$33	10.865091	$10	4
54	5	Wednesday	10.191207	$10	6.364244	$6	4
55	6	Wednesday	21.699484	$21	19.319365	$19	4
61	12	Wednesday	17.037145	$17	3.745407	$3	3
65	16	Wednesday	16.948235	$16	15.232957	$15	4
70	21	Wednesday	23.378795	$23	18.678058	$18	4
71	22	Wednesday	12.979949	$12	6.346315	$6	4
74	1	Thursday	38.867939	$38	31.071062	$31	4
76	3	Thursday	6.430548	$6	5.641970	$5	4
80	7	Thursday	26.519510	$26	21.832280	$21	3
82	9	Thursday	11.507121	$11	14.146257	$14	3
84	11	Thursday	10.181336	$10	3.554036	$3	4
87	14	Thursday	27.712570	$27	27.712570	$27	2
89	16	Thursday	26.054552	$26	26.018165	$26	4
92	19	Thursday	6.433588	$6	6.433588	$6	2
98	1	Friday	18.894450	$18	18.055512	$18	4
99	2	Friday	21.991279	$21	17.064795	$17	4
100	3	Friday	43.224626	$43	43.224626	$43	2
113	16	Friday	10.821055	$10	10.588692	$10	3
119	22	Friday	14.954982	$14	11.192073	$11	3
124	3	Saturday	28.790955	$28	14.153939	$14	4
128	7	Saturday	31.699634	$31	42.919022	$42	3
130	9	Saturday	12.906509	$12	10.716854	$10	3
132	11	Saturday	21.899035	$21	21.899035	$21	2
134	13	Saturday	33.697137	$33	30.501217	$30	4
135	14	Saturday	112.236858	$112	10.097720	$10	4
140	19	Saturday	64.611315	$64	42.453055	$42	3
141	20	Saturday	32.888508	$32	12.908583	$12	3
142	21	Saturday	22.725838	$22	19.306658	$19	4
147	2	Sunday	6.552647	$6	6.552647	$6	2
150	5	Sunday	16.363175	$16	16.363175	$16	1

	time_of_day	day_of_week	value_mean	lab_mean	value_med	lab_med	sample_size
155	10	Sunday	21.455414	$21	18.242860	$18	4
161	16	Sunday	19.113806	$19	18.482689	$18	4
162	17	Sunday	35.692359	$35	35.692359	$35	2
163	18	Sunday	31.383464	$31	25.833059	$25	3

1. 1.14134671166539
2. 54.8265688507234

V2. Parametric model: Linear regression

The above models are non-parametric, i.e. they do not make any assumption on how the true average hourly pay are generated. In model V2 we impose a regression formula for the generation of the average hourly pay. In the regression model below, the average hourly pay is the dependent variable, and it is explained by the following predictors:

- time_of_day
- day_of_week
- special_event

I estimate the regression formula unknown parameters using the sample data (job).

In [154…

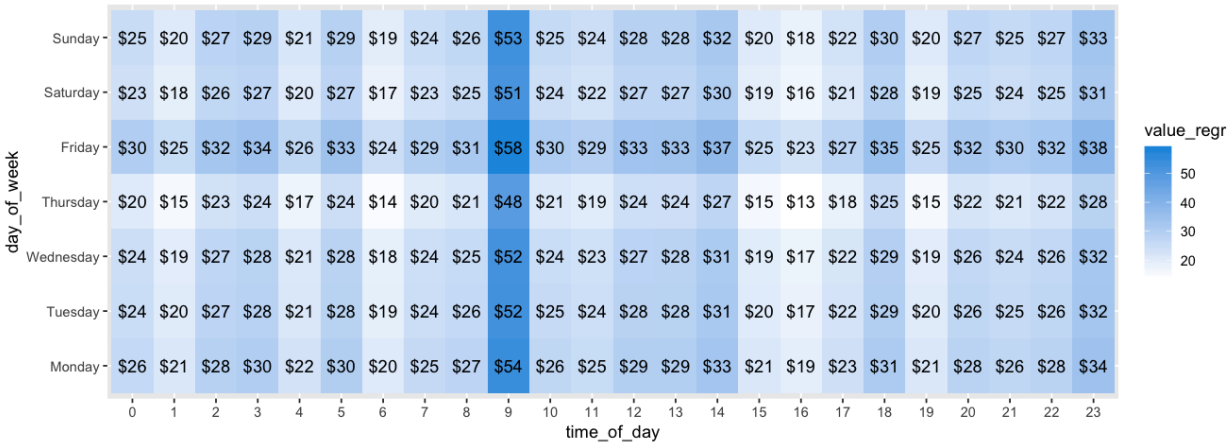
```
tmp <- hour_pay # I define a new temporary dataset tmp
tmp$day_of_week <- as.factor(tmp$day_of_week)
tmp$time_of_day <- as.factor(tmp$time_of_day)

# I estimate the regression parameters using the lm() function
modv2 <- lm(hourly_earnings~day_of_week+time_of_day, data=tmp)
summary(modv2)
```

```
Call:
lm(formula = hourly_earnings ~ day_of_week + time_of_day, data = tmp)

Residuals:
   Min     1Q Median     3Q    Max
-52.81 -19.50 -10.14   7.58 495.36

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          30.1603     7.2227   4.176 3.25e-05 ***
day_of_weekMonday    -3.9631     5.3880  -0.736  0.46219
day_of_weekSaturday  -6.3956     5.1784  -1.235  0.21712
day_of_weekSunday    -4.9686     5.2593  -0.945  0.34505
day_of_weekThursday  -9.6917     5.0828  -1.907  0.05687 .
day_of_weekTuesday   -5.3128     5.2821  -1.006  0.31476
day_of_weekWednesday -5.7965     5.2600  -1.102  0.27075
time_of_day1         -4.7735     9.1501  -0.522  0.60202
time_of_day2          2.7750     9.4913   0.292  0.77007
time_of_day3          3.9055    10.5502   0.370  0.71133
time_of_day4         -3.2592     9.8389  -0.331  0.74052
time_of_day5          3.8276     9.3045   0.411  0.68090
time_of_day6         -5.7672     9.5266  -0.605  0.54508
time_of_day7         -0.3475     9.4797  -0.037  0.97076
time_of_day8          1.3299     9.3991   0.141  0.88751
time_of_day9         28.0182     9.1786   3.053  0.00233 **
time_of_day10         0.5512     9.5965   0.057  0.95421
time_of_day11        -0.8065     9.2990  -0.087  0.93091
time_of_day12         3.5933     9.2732   0.387  0.69848
time_of_day13         3.7136     9.1276   0.407  0.68421
time_of_day14         6.8680     9.6966   0.708  0.47895
time_of_day15        -4.6468     9.1750  -0.506  0.61266
time_of_day16        -6.9306     9.9379  -0.697  0.48574
time_of_day17        -2.3337     9.5613  -0.244  0.80722
time_of_day18         4.9740     9.2237   0.539  0.58984
time_of_day19        -4.7606     9.6282  -0.494  0.62111
time_of_day20         1.9512     9.4819   0.206  0.83701
time_of_day21         0.6176     9.7520   0.063  0.94952
time_of_day22         2.0090     9.7662   0.206  0.83707
time_of_day23         8.0263     9.2880   0.864  0.38773
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 43.43 on 913 degrees of freedom
Multiple R-squared:  0.03103,   Adjusted R-squared:  0.0002492
F-statistic: 1.008 on 29 and 913 DF,  p-value: 0.4547
```

Note in the table of estimated parameters above that we have one parameters estimated for 6 days of the week (Friday is used as reference level) and 23 hours of the day (midnight is used as reference level). Using the estimated regression model, I then predict the average hourly pay for each compbination of inputs.

In [155…

```
# Predict values using estimated regression model
avg_pay$value_regr <- predict(modv2, avg_pay)

# I use ggplot to visualize the new results
avg_pay$lab_regr <- paste0("$",as.character(floor(avg_pay$value_regr)))
hm_regr <- ggplot(avg_pay, aes(time_of_day, day_of_week)) +
  geom_tile(aes(fill = value_regr)) +
  geom_text(aes(label = lab_regr)) +
  scale_fill_gradient(low = "white", high = "#1b98e0")
options(repr.plot.width=11, repr.plot.height=4)
hm_regr
```



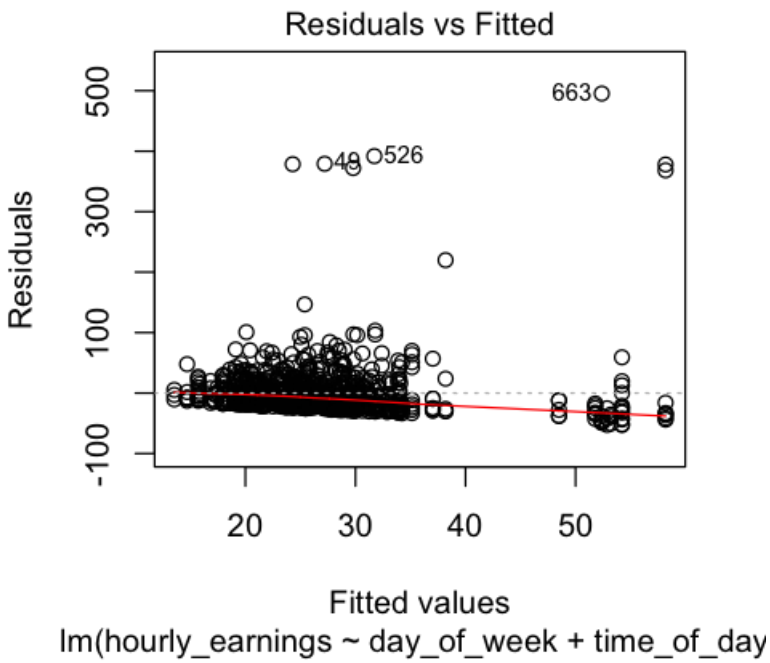
Note that the new heatmap show values that are much more closer to the overall median for the whole sample than the heatmaps generated with the mean and median functions. This is reasonable as the data was generated using a uniform distribution, and we would not expect strong differences between different times of the day and days of the week.

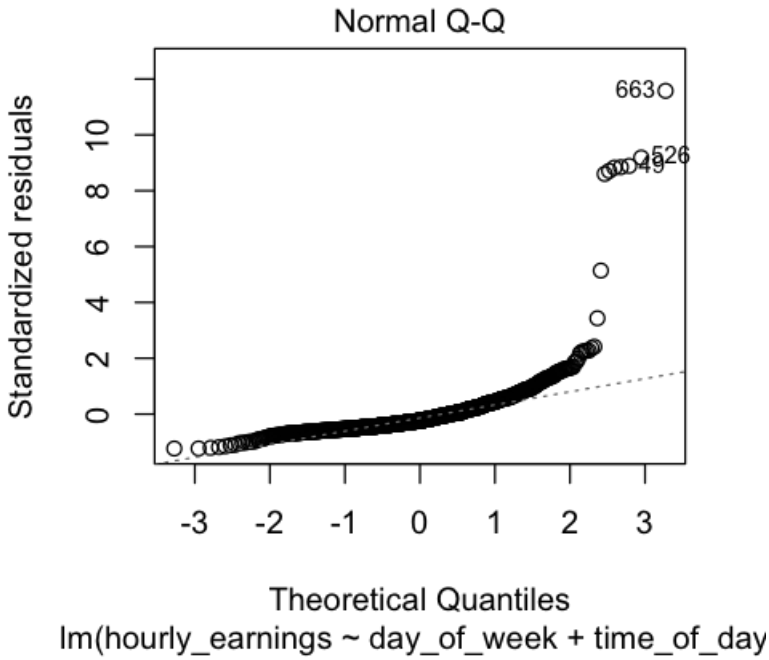
A parametric model has other advantages:

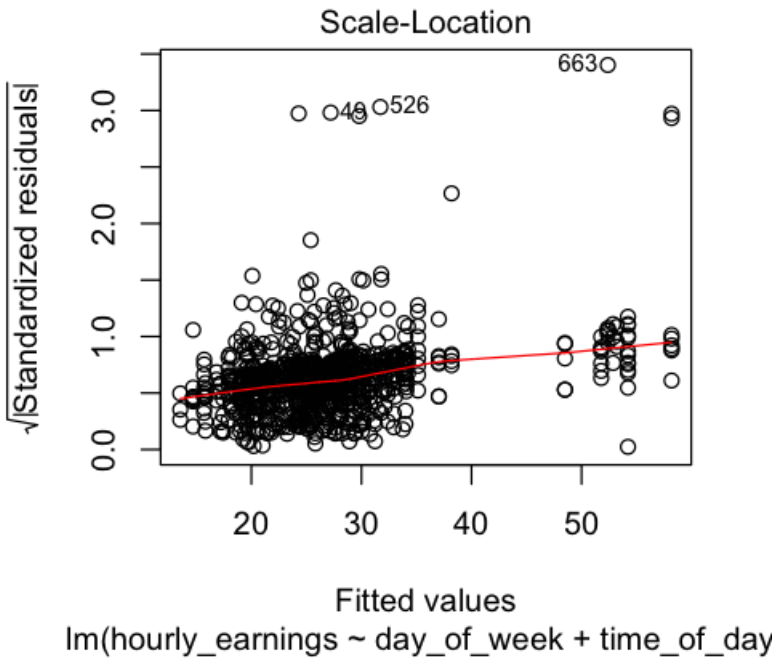
- it uses the whole dataset to predict average hourly pay
- it allows to use multiple predictors simultanously. For instance, in the model we also use special events. If for some reason we are interested to produce the same visualization only for special events, then we only need to change the special_event column from 0 to 1 and predict the new values
- it allows to use new variables (e.g. we can introduce spacial predictors, or individual-specific characteristics, for instance type of vehicle used, user rating etc.)
- model diagnostic can be used to identify outliers and extreme values (see below). While linear regression is still influenced by outliers, there are more advanced regression methods that can further increase robustness of the model

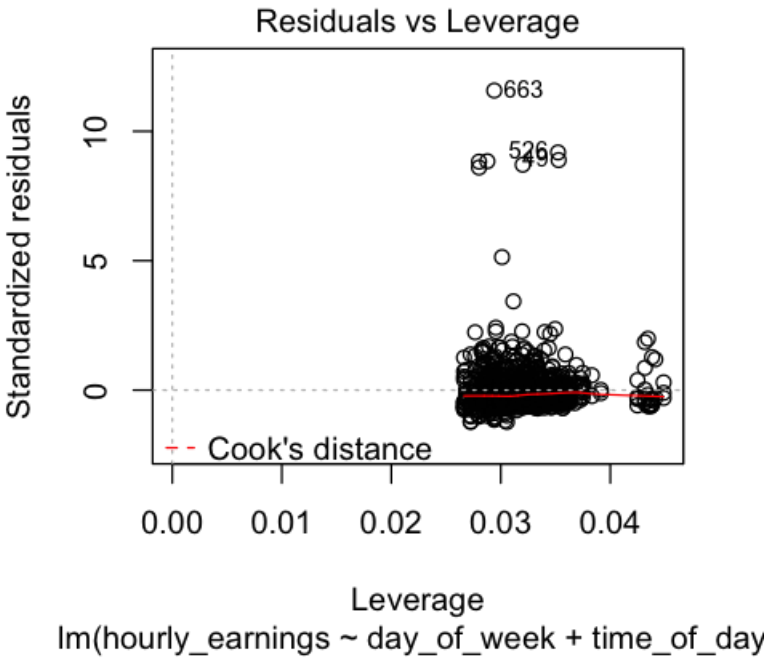
In [156…

```
# Run model diagnostic to identify potential outliers and extreme values
options(repr.plot.width=4, repr.plot.height=4)
plot(modv2)
```




Normal Q-Q
Standardized residuals
663
526
49
0 2 4 6 8 10
-3 -2 -1 0 1 2 3
Theoretical Quantiles
lm(hourly_earnings ~ day_of_week + time_of_day


Scale-Location
√|Standardized residuals|
663
49
526
0.0 1.0 2.0 3.0
20 30 40 50
Fitted values
lm(hourly_earnings ~ day_of_week + time_of_day



V3. Parametric model: Mixed effet model

A problem with model V2 is that it assumes that observations are independently generated. This is not the case, since different trips performed by the same driver over time might be correlated. For instance, a driver might have a stronger preference for longer trips, hence his/her earnings will be larger than other drivers. Or a driver might be seen as more friendly by customers, and therefore she/he will be able to earn larger amount of tips.

In this last model V3 I will take this problem into account, and include driver_id as a random factor in the regression model. The obtained regression is a "mixed-effect model".

In [157…

```
tmp <- hour_pay # I define a new temporary dataset tmp
tmp$day_of_week <- as.factor(tmp$day_of_week)
tmp$time_of_day <- as.factor(tmp$time_of_day)
tmp$driver_id <- as.factor(tmp$driver_id)

# I estimate the regression parameters using the lmer() function
modv3 <- lmer(hourly_earnings ~ day_of_week +
                time_of_day +
                (1|driver_id), data=tmp)
summary(modv3)
```

```
Error in lmer(hourly_earnings ~ day_of_week + time_of_day + (1 | driver_id), : could not find funct
ion "lmer"
Traceback:
```

Somehow I was not able to load the lme4 package in jupyter. Moreover, I would need to generate data from multiple weeks. Therefore, at this point, this model is not estimable. It is meant only as an example of more complex analysis that could be done.

Conclusion

I wanted to show 4 different models, as they approach the problem in different ways and each has some advantages and disadvantages. Moreover, I usually operate by starting from a "baby model" and then progressively test and compare more complex models.

Here, the median is defiantely more robust than the mean function, in estimating the average hourly pay. However, regression models are more flexible and can make use of multiple variables simultanously, as well as make use of the whole dataset.

Here I tested a few classic statistical models, which are simpler but of easy interpretation. Going forward, more complex machine learning models can be tested, especially given that the goal here is not understanding the

relationship between the dependent variable (hourly pay) and the input data, but rather have the best prediction.